UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

SISTEMA DE ADQUISICIÓN Y CODIFICACIÓN DE SEÑALES DE AUDIO

INFORME DE SUFICIENCIA

PARA OPTAR AL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

Franca Alina Giacchetti Burrei

PROMOCIÓN
1994-I

LIMA-PERÚ
2005

A mis padres
y a mi hermana Silvana

Sistema de Adquisición y Codificación de Señales de Audio

SUMARIO

Desde la aparición de los sistemas de grabación digital de audio, como el PCM de Sony, el mundo discográfico ha vivido una impresionante mejora de la calidad de sonido. Las copias digitales suenan exactamente como el original, dado que no se producen las típicas pérdidas de las copias analógicas (aumento de ruido de fondo y pérdida de agudos en las cintas de cassette). El único inconveniente de la grabación digital es el espacio que ocupa: unos 10 MB por minuto en estéreo, para obtener la calidad de un disco compacto. De ahí que distintos fabricantes hayan buscado soluciones que permitan almacenar el sonido en menos espacio, pero manteniendo un nivel de calidad semejante al del CD. Los ficheros de información son susceptibles de ser recodificados o comprimidos por medio de algoritmos y procedimientos especiales, de forma que ocupen menos bytes y por lo tanto menos espacio en los discos. Por eso se desarrollan programas de compresión y descompresión que facilitan estas tareas. Tenemos el formato WAV que guarda el sonido tal como viene, graba todas las notas, mientras que el formato MIDI graba el sonido nota por nota, o sea, la computadora sabe lo que graba, y el MP3 que comprime el sonido reduciendo su calidad muy ínfimamente, casi imperceptible para nuestro oído mediante un complejo algoritmo de compresión. Justamente el trabajo se centrará en este

importante formato MP3, analizándolo en sus aspectos con profundidad.

INDICE

Página

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

Se sabe que los archivos de audio y video, o imágenes son los más consultados y buscados en la Web. Pero también es cierto que todos son demasiado pesados y a veces difíciles de transferir.

Un archivo de audio digital es un sonido o secuencia de sonidos que ha sido convertido a un formato numérico para poder ser almacenado en un computador.

Por audio digital entendemos la grabación (digitalización) de sonido real, ya sea procedente de voces, instrumentos musicales acústicos o electrónicos, grabaciones, etc., en ordenador, en forma de ficheros informáticos.

Esta digitalización, también denominada *muestreo* (sampling), se realiza mediante los denominados ADC, o conversores de analógico a digital, circuitos que, a una determinada frecuencia, toman "fotografías" del sonido, que convierten en números que después son almacenados en la memoria del ordenador (RAM o disco duro, dependiendo de su tamaño).

Pero al hablar de audio digital no hay que olvidar que no sirve de nada tener el sonido digitalizado en el ordenador si no podemos escucharlo. Para ello, necesitamos hacer el proceso inverso al del muestreo: la conversión de digital a analógica, encargada a los circuitos DAC. Además de convertir los

números almacenados en el ordenador a una señal eléctrica, se debe filtrar ésta para obtener una señal válida. En la calidad de dichos filtros reside, en muchas ocasiones, la calidad de sonido de una tarjeta de muestreo, obteniendo en algunas un nivel de ruido de fondo que las hace inútiles para usuarios exigentes.

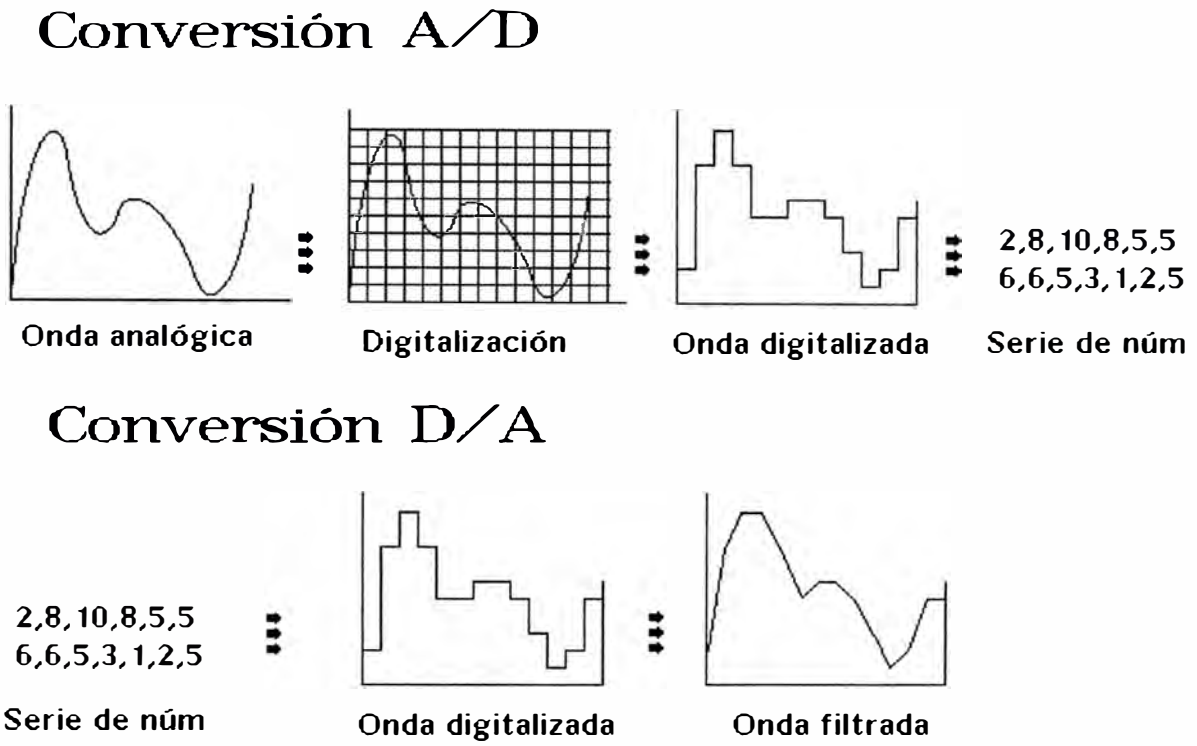


Figura 1: Proceso que convierte una señal de analógica a digital y viceversa.

Como es de suponer, entre los procesos de digitalización y escucha, tenemos acceso a una variada gama de manipulaciones del sonido, que nos permiten obtener resultados imposibles, o, al menos, muy difíciles de realizar por otros métodos, sin necesidad de usar un caro equipo especializado.

Para medir la calidad del muestreo, debemos referirnos a dos parámetros: la frecuencia de muestreo y la resolución.

La frecuencia de muestreo se refiere al número de mediciones que se realizan por segundo. Cuanto mayor sea esta frecuencia, más parecido será el resultado obtenido al sonido original. Según el teorema de Nyquist, la frecuencia mínima de muestreo debe ser el doble del ancho de banda de la señal original. En términos más sencillos: si el sonido original llega, en la zona de los agudos, pongamos, 10000 hertzios, debemos muestrear a partir de 20000 hertzios, esto es debido a que cuando la onda es más aguda es mucho más corta y son necesarias más particiones en tiempo para representarla.

Dado que el oído humano es capaz de escuchar sonidos en el rango de 20 a 20000 hertzios, aproximadamente, se ha elegido como frecuencia de muestreo más adecuada, la de 44.1 KHz, es decir, aproximadamente el doble de la frecuencia más aguda que podemos escuchar.

Para calidad de CD se hace un muestreo de 44.1 KHz, o lo que es lo mismo, 44100 muestras en 1 segundo, lo que es suficiente para "engañar" al oído humano, ya que el ancho de banda de este es de 20 KHz, por consiguiente un muestreo mayor sería imperceptible, ya que el oído no notaría la diferencia.

Si deseamos digitalizar sonidos acústicos, no es necesario alcanzar esas frecuencias de muestreo, ya que ni las voces ni los instrumentos acústicos producen frecuencias relevantes por encima de los 10 KHz, aproximadamente (de hecho, un sistema de reducción de ruido de Philips, y la codificación usada en los DCC - Digital Compact Cassette, se basa en esta realidad, por lo que corta todas aquellas frecuencias por encima de

dicho límite). Así pues, en estos casos se puede utilizar una frecuencia de 32 KHz, o incluso de 22 KHz.

Si reducimos la frecuencia de muestreo, podemos apreciar que el sonido es menos nítido, más apagado, porque perdemos frecuencias agudas.

La resolución hace referencia a la exactitud de las medidas efectuadas. Se mide en bits: si la resolución es de 8 bits tenemos 256 niveles posibles. Si ampliamos a 16 bits, cada medida puede estar en un rango de 0 a 65535. Como se ve, la precisión en este último caso es mucho mayor.

Para tener un punto de referencia, se puede decir que los discos compactos graban la música, como se mencionó anteriormente, con una frecuencia de muestreo de 44.1 KHz, y una resolución de 16 bits [4]. Como se puede comprobar, esos parámetros ofrecen una calidad de sonido realmente buena.

Es curioso comprobar cómo el ojo es más fácil de engañar que el oído: para dar sensación de movimiento, el cine y la televisión usan una frecuencia de entre 24 y 30 fotogramas por segundo [4], mientras que ¡la música necesita 44100!

Este proceso que se acaba de explicar se denomina PCM (Pulse Code Modulation) y el tamaño de los ficheros así grabados es realmente monstruoso.

El tener música con calidad digital no es algo nuevo, de hecho los Compact Disc (CD) y más recientemente los Digital Audio Tape (DAT) son una viva demostración de ello, en el ámbito de la computación la idea tampoco es nueva, de hecho es posible, en una computadora, almacenar la

información que conforma el sonido digital... desde hace mucho tiempo, ¿y entonces porque no se había hecho?... pues principalmente no se hacía por el "tamaño" (aquí el tamaño sí importa), veremos por qué con un pequeño ejemplo, digamos que queremos guardar en nuestro disco duro una muestra de nuestra canción favorita de 1 minuto, como queremos calidad de CD, necesitamos que se haga la muestra a 44.1 KHz, estéreo y con 16 bits por muestra. 44.1 KHz significa que tendremos 44100 valores por segundo viniendo de nuestra tarjeta de sonido (o de nuestro archivo), multipliquemos esto por 2 puesto que tenemos dos canales (por ser estéreo), multipliquemos esto otra vez por 2 (eso es lo que significa 16 bits) y entonces tenemos:

$$44100\,\frac{muestras}{segundo} \times 2\,canales \times 2\,\frac{bytes}{muestra} \times 60\,segundos = 10584000\,bytes = 10.0936\,MB \approx 10\,MB$$

En otras palabras, que 1 minuto de sonido nos ocupa más de 10 MB de disco duro. Si tenemos en cuenta que en 1995 el tamaño promedio de un disco duro era de 1.2 GB, podríamos entonces haber copiado a nuestro disco duro tal vez nuestros dos discos compactos favoritos (un CD tiene capacidad para 650 MB o 74 minutos). En la actualidad con los discos de 10, 20, 30 y más Gigas, podríamos copiar algo más de 10, 20, 30 y más de nuestros discos favoritos respectivamente.

Cuando se codifica en PCM se almacena toda la información obtenida del muestreo, lo que trae como consecuencia archivos demasiado extensos y redundancia de información, lo que hace que se utilicen anchos de banda grandes lo que ocasiona un aumento en el costo de la transmisión.

	Rango de frecuencia (Hz)	Frecuencia de muestreo (KHz)	Bits por muestra	Velocidad de bits en Kbps
"Calidad telefónica"	300-3400	8	8	64
"Voz"	100-10000	16	8	128
"Audio de bandamedia"	10-11000	24	16	384
"Audio de bandaancha"	10-22000	48	16	768
"CD estéreo"	10-22000	44.1	16	1410

Tabla 1: Parámetros básicos para la codificación PCM con señales de audio.

Si se trata de descargar archivos de Internet, se comprueba que 30 megabytes son enormes. Si se está usando un módem conectado a Internet, 30 megabytes de datos tomarán varias horas de descarga.

Este volumen de información no puede transmitirse de forma práctica en Internet. Debemos pues, recurrir por un lado a grabaciones con un nivel de calidad un tanto inferior, y por otro al empleo de sistemas de compresión de sonido. Una grabación a 16 KHz, 8 bits y estéreo ocupa 32 KB/seg; y una grabación a 8 KHz, 8 bits y mono ocupa 8 KB/seg. Estos tamaños ya son más aceptables para su transmisión en Internet. La diferencia de calidad entre los distintos valores de grabación es apreciable, pero si tratamos únicamente de transmitir sonido o palabras que puedan ser entendidos, pueden emplearse los valores más bajos, usar frecuencias de muestreo más bajas y resolución de 8 bits para aquellos trabajos que no requieran tanta calidad (juegos, enciclopedias, etc.). Sin embargo, si intentamos transmitir sonido de más calidad o bien transmitir sonido en tiempo real, habrá que recurrir a métodos de compresión. Existen sistemas de compresión que

almacenan y leen en tiempo real los ficheros de audio, con la ayuda de chips DSP (procesadores de señal), consiguiendo razones de 1:4 o superiores.

Un módem de 33,6 Kbps en condiciones óptimas ideales puede transmitir 4,1 KB/seg de información. En la práctica, para transmitir sonido en tiempo real sin interrupciones a través de un módem convencional deben emplearse compresiones que permitan reducir la secuencia sonora a 2 KB/seg como máximo.

Como se dijo al principio, estos enormes volúmenes de información hacen difícil su tratamiento en un ordenador personal, sin mencionar la dificultad de transportarlos en soportes físicos y sobre todo enviarlos a través de Internet, por ello, el propósito de este Informe de Suficiencia es tratar con las formas que existen para que los ficheros de información sean susceptibles de ser recodificados o comprimidos por medio de algoritmos y procedimientos especiales, de forma que ocupen menos bytes. Así se consiguen algunas ventajas, por ejemplo que los archivos ocupen menos espacio en los discos. Por eso se desarrollan programas de compresión y descompresión que facilitan estas tareas.

La compresión es posible porque normalmente en una fuente (el código, los caracteres que componen un fichero), además de información hay también redundancia, es decir, datos que no aportan más información, en general porque pueden obtenerse a partir de los datos anteriores.

¿Existe un límite en cuanto a lo que podemos llegar a comprimir un fichero, un punto a partir del cual no se pueda reducir ya más el tamaño de la fuente, un punto en el que se haya eliminado toda la redundancia y ya

sólo quede información? La intuición nos dice que sí. Si llegáramos a comprimirlo hasta 0 ó 1 byte, parece difícil que se pueda sacar toda la información de ahí.

Dentro de la Teoría de la Información, que es la rama de la ciencia que se ocupa de la compresión de datos, hace tiempo que se ha demostrado que ese límite, efectivamente, existe, y recibe el nombre de *entropía* (según definición de Mp3-tech.org's glossary: "Medida de la información contenida en un mensaje, es el compactado más bajo que se consigue a causa de la compresión").

Todos los ficheros no son compresibles en la misma proporción. En líneas generales podemos decir que los ficheros de texto pueden comprimirse razonablemente, que los ficheros de programas pueden comprimirse poco, y que otros ficheros (por ejemplo gráficos BMP) pueden comprimirse mucho más. También podemos asegurar que la compresión será menos eficaz si el fichero es pequeño.

Aunque parezca raro, los archivos de audio son difíciles de comprimir. Para datos de 8 bits se han obtenido sistemas sencillos y exitosos, pero para 16 bits tanto Philips como Sony han gastado millones para obtener sus formatos de compresión.

Las formas mencionadas anteriormente se denominan *formatos de audio*, que, desde antes de la aparición de Internet, había muchos y distintos, pero eran específicos de una plataforma.

Hay diferentes tipos de formatos:

- Los formatos de onda de audio guardan la información captada por un

micrófono, almacenando la amplitud del sonido y su frecuencia cada cierto período de tiempo, ejemplo de ello es el formato básico WAV.

- Los formatos de secuencia almacenan las notas, leyéndolas desde algún tipo de entrada MIDI; se graban varias secuencias que se ponen en determinados canales. Se deja al computador y a un estándar internacional (que define, por ejemplo, que en el canal 0 siempre va el piano), el definir la forma en que se tocará cada canal. El ejemplo típico es precisamente, MIDI (extensión .mid).
- Los formatos mixtos almacenan al comienzo un ejemplo de cómo sonará cada canal, de una manera similar a los formatos de onda de audio, y luego graban una secuencia de las notas para cada canal. El ejemplo típico de este formato es MOD, el cual es ampliamente implementado en varios sistemas por su capacidad de generar una excelente calidad de sonido y al mismo tiempo caber en un espacio de disco muy pequeño.
- Los formatos que utilizan la compresión de audio explotando las características "imperfectas" de nuestros oídos (es decir, el como percibimos el sonido). La compresión de audio en realidad consiste en dos partes, la primera llamada "codificación" (o para los hispanohablantes "compresión") que sería la transformación de la señal original de audio, que reside digamos en un archivo de onda (WAV) en una forma de archivo altamente comprimido llamado "bitstream" (o datos de audio codificados). La segunda parte, el reproducir el sonido en la tarjeta de sonido de nuestra PC, llamada "decodificación", simplemente toma el archivo "bitstream" y lo reconstruye en un archivo WAV. Para

lograr la codificación se usan los mentados algoritmos (que dicho sea de paso consumen mucho tiempo del procesador o pongámoslo de esta forma, hacen que nuestra máquina trabaje como no lo hará en ningún momento nuestro procesador de palabras), estos explotan las redundancias y las irrelevancias en un dominio de frecuencias basado en el modelo auditivo humano, esto básicamente significa que la señal original contiene elementos que quedan fuera de la capacidad del oído humano para ser escuchados, además de tener señales repetitivas las cuales por decirlo de una forma sencilla se guardan como en los cancioneros; "se repite"; y no una y otra vez físicamente. Un ejemplo de esto es el formato MP3.

Trataré en los capítulos posteriores una descripción general de los formatos más conocidos, dedicando un capítulo especial al formato WAV, otro al formato MIDI y otros dos capítulos explicando la Psicoacústica y el formato MP3, dándole mayor importancia a este último.

Con el formato MP3 se describirán aspectos técnicos y se explicará cómo funciona el esquema de codificación MP3 y cual es su relación con la Psicoacústica.

Las limitaciones encontradas en el estudio del MP3, se deben a la dificultad de encontrar información útil que explique su algoritmo en profundidad ya que, principalmente debido que el estándar ISO/IEC 11172-3 (que describe la manera de generar flujos de audio MP3 válidos), así como la tecnología MP3, tienen copyright.

Debo, finalmente, agradecer la existencia de aquella maravilla que nos

acompaña cada día más y más que es la red de redes o sea el Internet, convertido en la mayor biblioteca y enciclopedia que jamás se hubiera imaginado, ya que sin su existencia habría sido muy difícil tratar con un tema como éste, y encontrar la suficiente información, profunda y actualizada, para poder hacer lo que logré, un Informe de Suficiencia para optar al Título de Ingeniero Electrónico de la Universidad Nacional de Ingeniería.

CAPÍTULO I
DESCRIPCIÓN GENERAL DE LOS FORMATOS DE AUDIO MÁS CONOCIDOS

De alguna manera, la historia de la grabación de audio y video es la carrera por conseguir sistemas y formatos cada vez más pequeños que precisen menores velocidades para funcionar, que permitan un mayor tiempo de grabación-reproducción y, todo esto sin menguar la calidad.

Así como imágenes, archivos de sonido también vienen en una variedad de formatos, de los cuales no todos pueden ser usados en toda plataforma. AIFF es un formato de audio nativo para la Macintosh; WAV es nativa de Windows. Ambas plataformas pueden tocar y guardar sonidos en formatos de audio AU, SND y MPEG. Ambas también usan el formato de MIDI, pero solo para música.

Históricamente, casi todo tipo de plataforma usaba su propio formato de audio. Los formatos son independientes de las características del equipamiento.

Al surgir el Web, las nuevas capacidades de los browser (programa de aplicación utilizado para ubicar y hojear páginas del www en el Internet), dieron soporte a muchos formatos distintos por medio de plug-ins que se encargan de interpretar la información separadamente, o capacidades multimediales incorporadas.

Actualmente se puede encontrar multitud de formatos de audio digital, algunos con mayor implantación en la PC y otros verdaderamente desconocidos que, bien por ser específicos de algunos programas o estar regulados por complicadas patentes, nunca tendrán una aceptación entre las masas.

1.1. Soportes digitales de audio

En la actualidad, enumerados por orden de aparición en el mercado, existen los siguientes sistemas de grabación de audio digitales:

DAT:

Simboliza Digital Audio Tape. Los estudios de grabación vieron las conveniencias de almacenar las grabaciones maestras y demás mezclas en un formato digital, que no perdiera calidad al realizar sucesivas copias (para los artistas, el productor, las emisoras, etc.).

Para este tipo de funciones se desarrolló el DAT. Este sistema digitaliza el sonido a muy alta calidad (de 32 KHz - modo larga duración - a 44 ó 48 KHz) y lo almacena en una cinta magnética, similar a los populares "cassettes". También es capaz de grabar directamente el formato digital CD Audio a 44.1 KHz. Es una cinta similar a video de 8 mm., 2 canales, hasta 180 minutos.

No se ha popularizado debido a su alto precio; y está dirigido, de hecho, a profesionales. No sólo lo utilizan para grabar el sonido en sí, sino también como copa de seguridad de todo tipo de archivos informáticos.

Al igual que el CD Audio, el DAT se basa en una grabación digital sin procesado posterior, es decir, se almacena en la cinta toda la información generada en la grabación, por lo que se obtiene la mayor calidad posible.

CDA:

Simboliza Compact Disc Audio o sea el formato de disco compacto de audio que ha supuesto la posibilidad de escuchar grabaciones de muy alta fidelidad en un soporte muy barato de fabricar. De lo que cuesta un disco, sólo un escaso 10% se debe al costo del material; el resto se dedica a los derechos de autor, gastos de distribución, promoción, y margen comercial.

El CD Audio se basa en una grabación digital sin procesado posterior, es decir, se almacena en el disco compacto toda la información generada en la grabación, lo que representa una gran cantidad de información (aproximadamente 10 MB por minuto).

Dado que un disco compacto es capaz de almacenar más de 650 MB de información, esto nos da unos 74 minutos de tiempo máximo por CD, una buena cantidad (hora y cuarto) donde cabe sobradamente la duración usada en los discos de vinilo L.P. (long play - larga - duración).

El problema del CD Audio es que no podemos grabar nueva música en dichos discos, porque vienen prensados de fábrica, como los antiguos L.P. Así pues, en aquella época, la única alternativa de grabación doméstica era la cinta de cassette.

DCC:

Simboliza Digital Compact Cassette. Philips ha desarrollado un sistema que pretendía invadir el mercado con un producto que hace la tecnología de grabación digital asequible en precio. Las siglas DCC amplían el nombre CC de las super conocidas cintas analógicas, que también fueron inventadas por Philips.

Una pletina (dispositivo para grabar y reproducir cintas magnetofónicas en cassette) DCC sirve para reproducir y grabar cintas DCC, pero también puede reproducir (no grabar) las cintas analógicas.

A diferencia del DAT o el CD Audio, el DCC utiliza un sistema llamado PASC (Precision Adaptive Sub Coding - subcodificación de precisión adaptiva) que, al no grabar la información de las frecuencias que no es capaz de percibir el oído humano normal, le permite almacenar más minutos de música en menos espacio.

Este sistema produce una ligerísima pérdida de calidad que la inmensa mayoría de oyentes no serán capaces de detectar, aunque lo hace inadecuado para su uso profesional en estudios de grabación.

Otro problema es el elevado precio de las cintas digitales vírgenes.

CDR:

Simboliza Compact Disc Rom. En los últimos dos años, han salido al mercado unidades de CD-ROM capaces de grabar cualquier tipo de información digital en *discos compactos vírgenes*.

Existen dos tipos de dispositivos: Las grabadoras (CDR), que permiten grabar un CD virgen una sola vez (tras lo cual, se usaría sólo para escucharlo, como un disco compacto prensado en fábrica, pero no se podría grabar encima). Por su parte, las regrabadoras (CDRW), además de lo anterior, pueden usar unos discos vírgenes especiales para regrabación, en los cuales pueden grabar, borrar y grabar encima hasta unas 1000 veces.

Aunque el precio era prohibitivo al principio, en la actualidad, se puede conseguir una grabadora por un precio accesible, y una regrabadora por una

cantidad prácticamente igual.

Este sistema permite grabaciones de alta calidad a un costo muy bajo (se acerca al de una cinta cassette virgen). Sin embargo, estos sistemas sólo funcionan como parte de un sistema informático.

La extracción de audio de un CD-ROM es diferente que la de un CD-Audio, la cual requiere un software especial (CDCopy, por ejemplo).

Hasta que Philips sacó al mercado un equipo independiente, con el tamaño adecuado para incluir como un componente más de una cadena de alta fidelidad, que funciona como grabadora y regrabadora, sin necesidad de usar un ordenador personal.

Aunque aún es muy caro en comparación con una pletina de cassette (a la que supera ampliamente en calidad de sonido), se prevé que los precios bajarán en el futuro, hasta convertirlo en una opción interesante para cualquier aficionado.

MD:

Simboliza Mini Disc. Es un disco magnético de 140 MB, de tamaño similar a los disquetes de 3.5 pulgadas, 2-6 canales, donde podemos grabar unos 74 minutos de música estéreo.

Del mismo modo que el DCC, utiliza un sistema de compresión de datos que le permite ahorrar espacio a costa de una pequeña pérdida de calidad. En este caso, se llama ATRAC (siglas de Adaptive Transform Acoustic Coding - codificación acústica de transformación adaptiva), con una tasa de compresión de 5:1, memoria resistente a golpes y conexiones digitales.

Aunque Sony lo está promocionando de manera exhaustiva, fabricando

incluso reproductores portátiles (tipo disc-man), hasta el momento no ha despegado, y parece difícil que lo haga, dado su elevado precio.

Comparte con el DCC el alto precio de los discos vírgenes.

CDRW:

Simboliza Compact Disc ReWritable. Ha sido explicado en el CDR.

DVD:

Simboliza Digital Versatile Disc o Digital Video Disc. Es un soporte de memorización que tiene el aspecto y las dimensiones de un CD-ROM (12 cm.), pero una capacidad muy superior.

Un disco DVD puede almacenar una película entera con video de altísima calidad (formato MPEG2) y audio estereofónico en varios idiomas diferentes.

Si es usado como soporte para datos, tiene una capacidad variable de 4.7 a 17 GB, el equivalente de cerca de 27 CD-ROM. La variabilidad depende del hecho que un DVD es grabable en ambas caras. Además, en cada carta están presentes dos niveles (capas) grabables separadamente, el primero de los cuales es semi transparente. La capacidad máxima se alcanza cuando son utilizadas todas las cuatro capas grabables. Más que al número de capas grabables, la notable capacidad del DVD se debe al tipo de laser utilizado, cuya longitud de onda es sensiblemente inferior a aquella de los laser utilizados en los lectores de CD, permitiendo "surcos" más delgados.

Para poder leer este tipo de soporte es necesario un lector apropiado, que puede leer también los CD-ROM normales. Como para los CD-ROM, existen también DVD grabables, DVD reescribibles y masterizadores DVD. Las capacidades comúnmente disponibles en la actualidad son:

- DVD-5: 4.7 GB (1 lado, 1 capa)
- DVD-9: 8.5 GB (1 lado, 2 capas)
- DVD-10: 9.4 GB (2 lados, 1 capa por lado)
- DVD-18: 17.0 GB (2 lados, 2 capas por lado)
- DVD-R: 4.7 GB (1 lado, 1 capa) (escribible una vez)
- DVD-RAM: 2.6 GB (por lado, 1 capa) (reescribible)
- DVD-RAM: 4.7 GB (por lado, 1 capa) (reescribible)

SA: [11]

Simboliza Silicon Audio. El último sistema de grabación digital ideado hasta el momento es mucho más avanzado que los anteriores, y presenta considerables ventajas, dado que el soporte de la grabación no posee elementos mecánicos, fuente de problemas, imprecisiones y consumo eléctrico.

El Silicon Audio, como su nombre indica, está basado en chips de silicio, es decir, memoria en circuitos integrados, donde se consigue almacenar ¡hasta *192 minutos*! de música con calidad CD en una sola tarjeta, del tamaño de las tarjetas telefónicas.

El sistema de compresión MPEG Layer 3 (Motion Picture Expert Group - grupo de expertos en imagen y cine) es el utilizado para reducir el tamaño de la información, sin perder calidad de sonido distinguible.

Existen dos tipos de tarjetas Silicon Audio: las RAM y las ROM; las primeras permiten escuchar y grabar música, mientras que las ROM sólo permiten escucharlas (como un disco compacto).

En 1993, la desconocida empresa Meister Electronics anunció la creación

de estos dispositivos, tomando Siemens y actualmente NEC el relevo. Ya existen varios prototipos totalmente operativos, e incluso han salido al mercado dictáfonos que usan este sistema, como el ICD-50 de Sony (16 minutos de sonido con calidad de cinta de cassette en un aparatito de sólo 80 gr.).

El único inconveniente de este sistema es el precio de las tarjetas de alta capacidad, aunque el avance en este campo es tan rápido, que en un par de años, llegará a un nivel muy competitivo.

Para entonces, tras un interregno de las grabadoras y regrabadoras de discos compactos, algunos afirman que el Silicon Audio desbancará a los demás formatos digitales.

1.1.1. Tipos de archivos de sonido

WAV:

Síntesis de ondas de sonido reales. Es el formato de audio de más calidad y el más universal, pues permite todo tipo de conversiones. Se ha convertido en el formato estándar de la Industria Discográfica y de las Librerías Musicales Profesionales, y de la postproducción de audio, por su máxima calidad de sonido. Creado por Microsoft en 1987, WAV viene de Wave, que significa "onda" en inglés. Por ejemplo, la librería Music Lab Collection permite a partir de sus CD Audio la conversión a formatos WAV, respetando así toda la calidad del sonido original. En el Capítulo II se tratará con amplitud este formato. Tiene extensión .wav.

AIFF:

Simboliza Audio Interchange File Format, un formato común para

almacenar y transmitir sonido muestreado. El formato fue desarrollado por la Apple Computer y es el formato de audio estándar para las computadoras Macintosh. Es también usado por Silicon Graphics Incorporated (SGI). Es medianamente conocido fuera de estos dos ambientes, pero bastante más que el formato AU. En general las páginas Web que incluyen sonidos los colocan en formato AIFF para las personas que tengan Macintosh, y WAV para los usuarios de PC. El formato codifica datos de audio en formas de onda en mono de 8 bits o estéreo. Los archivos AIFF generalmente terminan con una extensión *.aif o *.ief. El formato AIFF no soporta compresión de datos así que los archivos AIFF tienden a ser grandes. Sin embargo, hay otro formato llamado AIIF-Compressed (AIFF-C o AIFC) que soporta razones de compresión tan altas como 6:1 y tienen extensión .aifc.

SND:

Es la abreviatura de sound y son sonidos Macintosh System 7 (y arriba) que pueden ser ejecutados en PC's usando convertidores tales como SOX. La extensión de estos archivos es .snd.

MIDI:

Simboliza Music Instruments Digital Interface. Es un estándar para transmitir información musical entre instrumentos electrónicos y computadoras. El tamaño de un archivo MIDI es pequeño, y no toma mucho espacio en el disco duro. La calidad de la música es muy buena. Tiene extensión .mid. En el Capítulo III se hablará más ampliamente de este formato.

1.2. Formatos de audio en Internet

1.2.1. Formatos con compresión sin pérdida ("loseless")

- Formatos de propósito general: ZIP, ARJ, RAR. Reducción máxima de 1:0.7.
- Formatos específicos para audio: WaveZIP, WavPAC, ZAP, Monkey. Reducción máxima 4:1. Es necesario reconvertir a WAV o AIFF para reproducirlos.

1.2.2. Formatos con compresión con pérdida ("lossy")

- Se basan en "codificación perceptual" para eliminar componentes de energía que teóricamente no se escuchan.
- Medida en Kilobits por segundo (Kbps).
- Aún usando el mismo sistema, dos programas diferentes pueden dar lugar a versiones diferentes.

MP3:

Son las siglas de MPEG-1 Layer 3. Es el formato más popular de compresión de audio. Permite almacenar canciones de 3 minutos en archivos de 4 Megabytes, un tamaño razonable para enviar por la Red y que superó por doce veces a todos los sistemas de codificación conocidos hasta el momento. Su extensión es .mp3. En el Capítulo V se tratará con amplitud este popularísimo formato.

Beatnik (Rich Media Format):

Es un formato desarrollado por y para diseñadores de sonido, tiene excelentes capacidades interactivas y una codificación complicada. Incorpora un sintetizador propio, para asegurar que la música MIDI suene igual en todas las plataformas. Tiene una tasa de compresión mayor incluso

que MP3.

AAC:

Simboliza Advanced Audio Coding (Codificación Avanzada de Audio), es un formato de audio comprimido de alta calidad diseñado por AT&T, Dolby Laboratories, Fraunhofer Institute for Integrated Circuits (Fraunhofer IIS), Sony Corporation, la Universidad de Hannover y NEC. Es un formato de compresión similar al MP3, que comprime la música que proviene de cualquier CD, en archivos mucho más pequeños. Es uno de los formatos de compresión de audio definidos por el estándar MPEG-2, sus siglas provienen del MPEG-2 Layer 1 y es mucho más eficiente que el formato MP3. Fue declarado estándar en abril de 1997. Es capaz de codificar un total de 5 canales (izquierdo, derecho, central, surround izquierdo y surround derecho). En sus estadios tempranos de desarrollo el AAC fue llamado NBC (Non-Backward-Compatible), porque no es compatible con los formatos de audio MPEG-1. MPEG-2 también definió otro formato de audio llamado *MPEG-2 Multichannel* o *MPEG-2 BC* (Backward Compatible), el cual es compatible con MPEG-1.

Compresión y calidad: Un archivo digital de un CD puede codificarse en formato AAC en 128 Kbps a 44 KHz y además tiene mejor calidad que un MP3 a 128 Kbps. Por lo tanto, un archivo de audio WAV de 44 MB de 16 bits y a 44 KHz puede ser comprimido a un AAC de 3.9 MB aproximadamente, dependiendo del reproductor que se utilice.

Ventaja:

- *Compresión:* Es un formato muy eficiente para la compresión, y además

puede bajarse a una velocidad mayor, y con mejor calidad.

Desventajas:

- *Los derechos de autor:* Como con los demás formatos, los usuarios lo utilizan para distribuir material con derecho de autor; y a pesar que el sistema de decodificación puede crear archivos de audio seguros contra la violación de la propiedad intelectual, las dos aplicaciones gratis para codificar audio en este formato, no son tan seguras y hacen que el formato sea tan abierto como lo es el MP3.
- *Diferentes variaciones del AAC:* Existen varias versiones del AAC. La primera fue desarrollada por AT&T para ser utilizada como un método de compresión de alta calidad para vender música a través de Internet. Luego, Liquid Audio tomó esta idea y creo otra variación comercial. El primer software codificador fue diseñado por Homeboy Software basándose en los prototipos de la ISO MPEG-4 (es decir, MP4).

  Por lo tanto, lo que se puede decir de este formato es que su tecnología recién comienza a utilizarse, y sus programas todavía están en desarrollo.
- *Dificultades para utilizar los programas:* Algunos de los reproductores y las aplicaciones de codificación son complicados de utilizar y producen resultados no muy claros.
- *Se requieren computadoras de mejor calidad:* A pesar que los archivos pueden reproducirse utilizando cualquier sistema Pentium, algunos reproductores consumen mucha CPU y producen algunas deficiencias.
- *No hay demasiados archivos disponibles en este formato:* Por lo tanto, el

MP3 continua siendo el más popular.

VQF (TwinVQ o TVQ):

Simboliza Twin Vector Quantization o Transform-domain Weighted Interleave Vector Quantization que es algo así como Transformada de dominio de intervalos de vectores cuantificados (a trozos) con peso.

Formato originalmente atribuido a Yamaha y desarrollado por los estudios NTT Human Interface Laboratories de Japón. La extensión de los ficheros generados por el TVQ es .vqf.

Su método, parecido al MPEG, no va comprimiendo trocito a trocito, sino que se van comparando por vectores o segmentos (de ahí su nombre). Los ficheros generados son más pequeños (en tamaño) a los MP3, pero "absorben" más recursos (tanto al comprimir como al descomprimir). Este formato consigue una compresión del orden de 1:10 a 1:20, lo que resulta en un archivo 35% menor en promedio que un archivo MP3, manteniendo entre tanto la misma calidad. La compresión con cualquier ordenador es mucho más lenta... consume 3 veces más que un MP3 (aproximadamente).

Compresión y calidad: Para notar la diferencia con los anteriores, un archivo WAV de 44.1 MB de 16 bits, y a 44 KHz pueden comprimirse dentro de un archivo VQF de 3 MB.

Ventajas:

- *Excelente compresión con una calidad de sonido espectacular:* Mucho más eficiente que el MP3, por eso muchos creen que será el próximo grande en el mundo del sonido.
- *Software gratis:* A pesar de ser diseñado con fines comerciales, la

mayoría del software, tanto para reproducir o codificar es gratis.

- *Variedad musical:* A pesar que este formato es relativamente nuevo, los sitios que se dedican a su promoción están surgiendo rápidamente y cada vez son más.

Desventajas:

- *Dificultades con las copias:* Como ocurre con los archivos MP3, muchos aficionados están codificando su música y distribuyéndola "on line", y esto puede ser un problema para las compañías grabadoras.
- *El equipo recomendado:* Para poder reproducir este tipo de formatos se necesita una computadora de mejor calidad que las requeridas para los formatos anteriores, por lo tanto se recomienda que quienes deseen escuchar archivos VQF posean una Pentium II, o una AMD K6 como requerimiento mínimo.

Veamos algunas características de cada uno:

Formato	Tamaño	Bitrate
MP3	3.99 MB	128 Kbps
VQF	3.0MB	96 Kbps
AAC	3.96MB	128 Kbps

1.2.3. Formatos para "streaming"

- *Streaming:* el audio se reproduce a medida que fluye desde el servidor hasta el cliente, en lugar de empezar cuando el archivo está completamente descargado.
- Generalmente se requiere un servidor especializado en generar y

controlar el flujo de datos hacia el cliente.

- Para modems lentos, hay que codificar a algo menos de 28 Kbps.
- MP3 y Beatnik también pueden utilizarse en streaming con ayuda de un metafile (al igual que Real Audio).

Real Audio:

Real Networks desarrolló un sistema de compresión de sonido digital, con la intención de aplicarlo a Internet. El objetivo era poder transmitir sonido en tiempo real, como una emisora de radio. Los sistemas de compresión han de adaptarse por tanto a la capacidad de transmisión de datos de los modems actuales. Muchas emisoras de radio convencional, emplean este software, para transmitir su programación en directo a través de Internet. Es el formato de streaming de audio de Real Networks. Con cualquier modem de 33.6; 28.8 e incluso de 14.4 puede recibirse sonido en tiempo real.

Es necesario un reproductor (plug-in), que incorporado a nuestro navegador (browser) de Internet realiza las funciones de recepción de sonido. Este reproductor puede obtenerse de forma gratuita y es el Real Audio Player.

La grabación de sonidos en este formato se realiza con un programa también de distribución gratuita denominado Real Audio Encoder.

Es capaz de reproducir sonido con fidelidad graduada, es decir, uno puede decidir que tan alta sea la calidad del sonido para así encontrar un equilibrio entre un archivo muy grande con alta fidelidad a uno muy pequeño de baja fidelidad. Los codecs propios de la empresa, permiten definir las tasas de compresión de sonido para adaptarlas a las condiciones de transmisión de

Internet. Es posible utilizar tasas tan reducidas como 5 Kbits por segundo.

Con este software se generan ficheros del tipo .ra a partir de ficheros sonoros WAV, o bien incluso, a partir de sonido en directo desde un micrófono conectado al ordenador. Los archivos también terminan en extensión .rm y .ram.

Este formato no es utilizado por la industria discográfica por su pobre calidad de sonido (muy inferior al formato CD Audio y WAV). Reduce el tamaño de los ficheros originales de música y sonido, lo que implica un empobrecimiento notable de la calidad. Solo aconsejable para hacer "demos" de CD o para insertar en páginas Web. Los ficheros en CD Audio o WAV pueden convertirse a Real Audio para facilitar su escucha en Internet. Se espera que este formato sea el que adquiera un mayor nivel de estandarización en el futuro.

Otras características:

- En realidad el sistema actualmente se llama G2 y ofrece audio de calidad aceptable incluso con modems de 28 Kbps.
- Permite sincronizar con fotos y textos.
- La calidad se "negocia" dinámicamente (si disminuye el ancho de banda, se utiliza menor calidad -si está disponible- pero sin necesidad de detener y volver a descargar).
- Es el formato de streaming más popular.

WMA:

Significa Windows Media Audio y es el formato de audio de Microsoft. Se trata de otro formato cada vez más extendido, que en un primer momento

solo podía ser reproducido por Windows Media Player, lo que limitó su implantación. Actualmente es admitido y puede reproducirse en los reproductores más conocidos.

Características:

- Calidad aproximada a CD codificando a 64 Kbps.
- Tiempo de codificación: 4 minutos de canción tardan 20 segundos a 128 Kbps, 44 KHz estéreo.
- Se puede sincronizar con imágenes.
- No funciona en servidores Mac.

Este formato fue creado para disputarle mercado al MP3, pero pruebas realizadas por revistas americanas constataron que la calidad del audio en WMA, cuando se utilizan tasas altas de bitrate (encima de 96 Kbps), es bastante inferior al MP3. El WMA distorsiona todas las frecuencias audibles y malogra las demás del sonido. En archivos de bitrate bajos, donde no es necesaria tanta definición de sonido (como en una conversación, por ejemplo), consigue una tasa de compactación mayor que el MP3, con una diferencia de calidad casi imperceptible.

Quick Time Audio:

Es un formato multimedia de Apple Computer y usado en los computadores Macintosh. Este estándar lleva mucho más tiempo vigente que el estándar MPEG. Los archivos quicktime tienen extensión .qt y .mov.

Por su ductilidad, este, formato fue adoptado por la ISO (International Organization for Standardization) como base del MPEG-4. El "revolucionario" Windows Media Player es una simple implementación del MPEG-4 estándar

a caballo de un formato propietario de Microsoft, sin poseer la integración que ofrece Quick Time de audio, video, MIDI, texto, animación Flash, sprites, fotos panorámicas y objetos 3D. Esta integración nos permite presentar una gran diversidad de contenidos en el sitio, sin que tenga que bajarse un nuevo plug-in a cada rato.

En el caso del audio, el codec QDesign del QT3 parte de la misma filosofía que el MP3, pero con una serie de optimizaciones acústicas que permiten trabajar a bitrates mucho más bajos, y obtener archivos que pesan entre 3 y 10 veces menos.

Para escuchar los temas se necesita una máquina con procesador Pentium o PowerPC, y bajarse e instalar el QuickTime (shareware).

Características:

- Permite codificar una variedad extrema de formatos (actúa como "envoltorio").
- Permite escuchar desde un punto sin recargar el archivo.
- No funciona en servidores Windows.
- Los codecs pueden irse descargando según se necesiten (sistema originalmente compacto).

Liquid Audio:

Formato de compresión de sonido solo soportado por el reproductor Liquid Player. Es un servicio que -mediante la incorporación de un código de seguridad- pelea por convertirse en el estándar para la transmisión de música legal por Internet. Los usuarios pueden comprar las canciones en formato de Liquid Audio y escucharlas con el reproductor de la compañía o

con el Real Player.

Es utilizado por algunas casas discográficas para la distribución de música por la Red. De similar calidad que Real Player. Para su escucha es necesario descargar el software Liquid Player en el disco duro del usuario.

Características:

- Formato desarrollado de cara a la industria musical profesional.
- Permite codificar "marcas de agua" para proteger derechos intelectuales de creación y de ejecución.
- Permite transferir meta-datos y archivos relacionados con el contenido sonoro (portadas de discos, letras, etc.).
- Las herramientas de codificación son caras.
- Sólo sirve para audio, no permite sincronizar imágenes.

Shockwave:

Es una nueva tecnología, creada por Macromedia, que permite la reproducción de archivos multimedia en la Red.

Es una pequeña herramienta de fácil instalación e integración con los dos navegadores más populares de la Red (Internet Explorer de Microsoft y Navigator de Netscape) que permite al usuario visualizar contenidos en la Web de alto carácter interactivo.

Para tener acceso a estos archivos en la Red, es necesario "bajar" al disco duro el plug-in de Shockwave e instalarlo en el buscador que utiliza para navegar en Internet. Este plug-in es gratuito.

Centenares de páginas web utilizan los archivos en formato Shockwave para ofrecer juegos, presentaciones de productos, música u otros servicios

de entretenimiento para ser consultados directamente desde un navegador.

Está preparado para distintas plataformas como Windows, Macintosh o America Online.

Características:

- Ofrece audio altamente comprimido y de buena calidad sin necesidad de tener un servidor especializado en streaming, pero no es tan flexible como otras soluciones.
- Requiere *Director* (es caro).
- Permite interactividad compleja gracias al lenguaje de programación Lingo (pero no tan compleja para audio).
- Requiere un cierto ancho de banda para funcionar bien.

Flash:

Flash debe sus raíces a una pequeña compañía llamada FutureSplash que fue adquirida por Macromedia en 1997 para complementar su programa Director que sirve para la creación de producciones multimedia interactivas, títulos de CD/DVD, etc., cuando deseaban darle un enfoque para el Web. Esta aplicación es una mezcla de un editor de gráficas y de un editor de películas.

Flash diseña gráficas de vectores; gráficas definidas como puntos y líneas en lugar de pixeles. Es decir que los vectores son como un conjunto de instrucciones matemáticas que por medio de valores le dan forma a una imagen. Así, un círculo vectorial, puede ser ampliado al tamaño que se desee y siempre seguirá siendo un círculo perfecto, cosa que no se lograría en una gráfica de pixeles y que rellena cada punto de la imagen con un color

para darle forma. Además de las gráficas vectoriales, Flash permite incluir audio comprimido en diversos formatos como el MP3, importar gráficas creadas con otros programas, formularios y algo de programación. Todo esto definido al igual que los vectores por un conjunto de instrucciones que mueven los objetos de posición y forma, y que dan como resultado archivos muy pequeños que se cargan en poco tiempo.

Es un programa en el que se pueden diseñar animaciones audiovisuales, pero que se comprimen en forma de texto para que el reproductor la decodifique y las presente tal como fueron creadas. Flash es independiente del navegador y el plug-in es universal, por lo que las animaciones diseñadas con este programa se verán casi idénticamente en cualquier plataforma y navegador.

La única desventaja que tienen las películas Flash, es que para poder visualizarlas, es necesario tener instalado el plug-in. Aunque, por el impacto que ha tenido esta tecnología, a partir de la versión 4.0 de los navegadores, el plug-in ya se incluye dentro de la instalación.

Flash es una tecnología con mucho futuro por su funcionamiento y facilidad de uso.

Características:

- Buena solución para páginas multimedia de gran impacto y con fragmentos sonoros cortos.
- Permite codificar audio en MP3 para descargarlo por streaming.
- También se integra bien con RealMedia (el audio puede ser RealAudio).

1.3. Formatos de audio multicanal

Aquí se describirá brevemente los principales sistemas de audio multicanal, haciendo notar las diferencias básicas entre ellos.

Dolby Surround Prologic:

Es un procesamiento de sonido cinematográfico, uno de los primeros sistemas multicanal que fue inventado por los laboratorios Dolby. Consiste en la separación de 4 canales de audio a partir de los dos canales principales (izquierdo y derecho). Los canales obtenidos son izquierdo, derecho, central e I y D trasero. Los canales traseros tienen la particularidad de que trabajan en mono, o sea reproducen una versión monofónica de las diferencias entre las dos señales de los canales originales con un leve retraso temporal que crea el efecto espacial. Tienen menos potencia de salida que los delanteros, y trabajan a partir de 100 Hz hasta 7000 Hz, por lo que la sensación de sonido envolvente es menos espectacular que los nuevos sistemas digitales como el AC3 o el DTS.

Este sistema se puede hallar en las cintas VHS de los videoclubs o en algunas películas transmitidas por las plataformas digitales como Canal Satélite Digital, Vía Digital, etc.

Dolby Digital o AC3:

Es el formato más corriente para el procesamiento de sonido Surround, es un sistema mejorado de Dolby Prologic. Se introdujo por primera vez en las salas de cine en el año 1992 pasando poco después al ámbito doméstico en los procesadores y receptores de alta gama.

La característica peculiar del AC3 es que todos los canales de audio son discretos, de esa forma hay un sentido más claro de dirección y una mejor

localización de efectos sonoros y del diálogo.

El software de Dolby Surround AC3 tiene seis canales discretos de sonido digital, cinco con ancho de banda completo y uno especial para reforzar los sonidos graves (izquierdo frontal, derecho frontal, centro, izquierdo trasero, derecho trasero, graves), o sea consta de 5 canales de sonido digital de alta calidad mas uno para bajas frecuencias (también llamados 5.1). Así pues nos encontramos con dos canales delanteros, uno central, dos traseros, que cubren todo el rango de frecuencias audibles (20 Hz - 20000Hz) y el canal para graves.

El sonido es mucho más espectacular que el Dolby Prologic, puesto que tenemos la misma potencia de salida por los 5 canales principales.

Este sistema de sonido se encuentra codificado en prácticamente todos los discos DVD y en algunas transmisiones digitales por satélite de última generación.

DTS:

Son las iniciales de Digital Theater System. Es un sistema de sonido multicanal de alta calidad. Es una tecnología inventada para transformar los sistemas existentes del "cine en casa" en seis canales discretos con un funcionamiento de audio digital de excepcional claridad, mejor que el disco compacto.

Descodifica 5.1 canales de audio a 20 bits en comparación con Dolby Digital que lo hace a 16 bits. Esto supone un notable incremento de la resolución en la reproducción sonora de la película. Como los otros sistemas, fue desarrollado para las salas de cine y así hizo su debut en el

año 1993 con la película Jurassic Park, de Steven Spielberg.

El DTS Digital Surround es el único formato de sonido Surround 5.1 que puede ofrecer esta experiencia tridimensional revolucionaria, mediante las nuevas grabaciones musicales en los discos compactos, pistas sonoras originales 5.1 en los discos láser de películas y, en poco tiempo, una amplia variedad de productos en DVD. Los fabricantes de equipos ya han comenzado a integrar esta nueva tecnología DTS en sus mejores procesadores Surround de A/V. También, se encuentra disponible un catálogo excepcional de discos compactos y discos láser de películas codificados DTS que pueden disfrutarse en cualquier reproductor con un conector de salida digital incorporado.

El sistema DTS no está plenamente desarrollado, puesto que aunque en la zona 1 (Estados Unidos) hay gran variedad de títulos, para la zona 2 (Europa, Japón ...) son escasos por el momento.

THX:

Sistema de sonido desarrollado por George Lucas, llamado así en honor de THX 1138, su primer largometraje, y se instalaron sistemas THX en dos cines para el estreno de "El regreso del Jedi". Hoy día, más de 2000 cines y salas de copiado de todo el mundo cumplen con los requisitos de THX.

Fue inventado por Lucas para conseguir un nivel de calidad superior en las salas donde proyectaban sus películas y como con los otros sistemas, mas tarde pasó al ámbito doméstico.

Lucasfilm comenzó la investigación del sistema de audio Home THX en 1986; le llevó cuatro años de trabajo hasta que pudo lanzar el Programa

Home THX. Hoy, existen más de 45 fabricantes autorizados que producen componentes con Certificación THX para Home Theater.

Otra definición dice que es un conjunto de normas exigentes que los equipos con esta certificación deben de cumplir, que no es propiamente un sistema de sonido multicanal. Este conjunto de normas incluye la corrección tonal y espacial del sonido ya que el sonido de las películas ha sido concebido para ser reproducido en salas de proyección grandes. De este modo los diálogos son más claros y el sonido se vuelve más nítido. También dispone normas sobre el tipo de altavoces que se emplean, así como amplificadores, procesadores, receptores, etc. Por lo general los equipos con certificación THX constituyen la gama alta de cualquier marca.

Existen dos clasificaciones dependiendo del nivel de exigencia, ULTRA y SELECT.

THX también certifica software de video para el hogar a través del Programa THX Digital Mastering, asegurando que el software DVD, discos láser y videocassettes VHS brinden la mejor calidad de imagen y sonido posible mediante el uso de la tecnología de THX patentada y muestren pericia excepcional en control de calidad.

MPEG 2:

Es otro formato multicanal pensado para la zona 2 y en la actualidad prácticamente en desuso. Se basa en la compresión de audio variable mediante algoritmos (método similar al MPEG 2 video por defecto en todos los discos DVD). Como consecuencia de esta compresión variable se obtiene un mayor espacio libre en el disco, pudiendo contener información

para configurar hasta 7.1 canales de sonido digital y varios idiomas.

Este formato carece del apoyo de la industria cinematográfica y es probable que pase sin pena ni gloria a una vida mejor.

CAPÍTULO II
EL FORMATO WAV

Los ficheros WAV de sonido digital (WAVE, Waveform Audio File Format) son el formato de audio creado por Microsoft y empleado con su sistema operativo Windows, de ahí su amplia difusión. Permite almacenar sonido con una calidad muy elevada, aunque lo hace a costa de ocupar un gran espacio en el disco. Prácticamente cualquier software de edición de audio para Windows soporta estos archivos y pueden ser reproducidos en otros sistemas operativos. Permiten definir en la grabación la frecuencia de muestreo que puede ser 8 KHz, 11 KHz, 22 KHz y 44 KHz, y el número de bits de información por muestra que puede ser 8 bits o 16 bits en mono o estéreo. Con valores mayores, la calidad será más alta, pero el tamaño del fichero generado también. Según el número de "muestras" que tomemos por segundo del sonido analógico, el sonido digital tendrá una mayor o menor calidad.

Un minuto de sonido con calidad CD (44.1 KHz / 16 bits estéreo) pesa 10 MB, y aún comprimido sigue pesando demasiado, por lo que la única opción que queda es bajar la resolución y cantidad de canales.

Es el formato de archivos de sonido digitalizado más utilizado en la PC y tiene como extensión .WAV que significa "wave" u "onda", porque el sonido se ha representado gráficamente como una curva u "onda" parecida a la

gráfica de la función matemática y = f(x) = sin(x), en el rango comprendido entre -360 < x < 360; -1 < y < 1. Estos archivos pueden contener música, voces, ruidos, efectos sonoros, etc. Se generan a través de la digitalización de sonidos reales realizada a través de una plaqueta de sonido.

Se trata de almacenar las muestras una tras otra (a continuación de la cabecera del fichero, que entre otras cosas indica la frecuencia de muestreo), sin ningún tipo de compresión de datos, con cuantificación uniforme. La sencillez de este formato lo hace ideal para el tratamiento digital del sonido.

El formato de los ficheros .WAV es el siguiente:

Bytes	Contenido Usual	Propósito/Descripción
00 - 03	"RIFF"	Bloque de identificación (sin comillas).
04 - 07	???	Entero largo. Tamaño del fichero en bytes, incluyendo cabecera.
08 - 11	"WAVE"	Otro identificador.
12 - 15	"fmt "	Otro identificador
16 -19	16, 0, 0, 0	Tamaño de la cabecera hasta este punto.
20 - 21	1, 0	Etiqueta de formato. (Algo así como la versión del tipo de formato utilizado).
22 - 23	1, 0	Número de canales (2 si es estéreo).
24 - 27	???	Frecuencia de muestreo (muestras/segundo).
28 - 31	???	Número medio de bytes/segundo.
32 - 33	1, 0	Alineamiento de bloque.
34 - 35	8, 0	Número de Bits por muestra (normalmente 8, 16 ó 32).
36 - 39	"data"	Marcador que indica el comienzo de los datos de las muestras.
40 - 43	???	Número de bytes muestreados.
resto	???	Muestras (cuantificación uniforme)

Tabla 2.1: Formato de los ficheros WAV.

Los datos numéricos que ocupan más de un byte se representan de la siguiente forma: primero están los bytes menos significativos y a continuación los más significativos (convenio "extermista menor", también

conocido como "formato Intel").

2.1. Ventajas del formato waveform

- Son más transportables que los archivos MIDI (ya que los archivos .wav pueden sonar prácticamente igual en cualquier plaqueta de sonido).
- Al ser un formato estándar para archivos de sonido digital, pueden ser incluidos en prácticamente cualquier aplicación que soporte elementos multimedia, lo cual no ocurre con formatos menos populares (como los .voc).
- Pueden ser incorporados como pista de audio en archivos .avi de "Video for Windows", pudiendo ser sincronizados con la secuencia de video en forma bastante exacta.
- Pueden ser reproducidos por el parlante de la PC si se cuenta con un driver apropiado.

2.2. Propiedades que caracterizan a un archivo .wav

❖ *Frecuencia a la cual fue muestreada o sampleada la onda sonora:*

Al digitalizar el sonido, las muestras son tomadas a la misma frecuencia de manera tal que la onda queda dividida en porciones de idéntico tamaño. Cuantas más porciones o muestras se tomen por segundo (es decir, a mayor frecuencia de muestreo) mayor es la calidad del sonido y mayor el espacio de almacenamiento requerido. Una mayor cantidad de muestras por segundo implica que también podrán ser recolectados tonos más altos. Con una frecuencia de 11025 KHz, que permite tomar 11025 muestras por segundo, sólo podrán ser capturados tonos correspondientes a frecuencias inferiores a 5513 KHz. Todas

aquellas frecuencias mayores que estuvieran presentes en el sonido original se percibirán en el sonido digitalizado como una distorsión. Es por esta razón que la frecuencia de sampleo está tan ligada a la calidad sonora. Las frecuencias de sampleo estándares para archivos .wav son 44.1 KHz, 22.05 KHz y 11.025 KHz.

- *Tamaño de la muestra:*

  La cantidad de información almacenada por sample, determina la precisión con la cual cada muestra es medida. La información correspondiente a cada sample, surge de dividir verticalmente cada muestra de la onda sonora en unidades equivalentes. Tomar muestras de 8 bits significa que cada muestra va a tomar un valor de las 256 unidades en las cuales se subdivide la onda sonora; en cambio si se toman muestras de 16 bits cada muestra, va a poder tomar uno de los 65536 valores posibles. Cuanto mayor sea el número de subdivisiones verticales utilizadas para describir las características de la forma de onda en cada muestra, mayor similitud ofrecerá dicha representación con respecto a la onda de sonido analógica original y, por supuesto, más espacio ocupará el archivo .wav correspondiente.

- *Número de canales grabados:*

  El número de canales sonoros especifica si la grabación produce una forma de onda (grabación mono o monoaural) o si produce dos formas de onda (grabación estéreo). El sonido estéreo ofrece mayor calidad en la reproducción pero requiere el doble de espacio de almacenamiento que el mono. Ya se han establecido los parámetros que determinan el tamaño

de un archivo .wav sin ningún tipo de compresión. Una forma de calcular dicho espacio es multiplicar la frecuencia de muestreo por la cantidad de bytes que ocupa cada muestra, y al resultado multiplicarlo por la cantidad de segundos que dura la secuencia de sonido. En el caso de que sea una grabación estéreo, al resultado se lo debe volver a multiplicar por dos. Por ejemplo, un archivo de sonido estéreo de un minuto de duración muestreado a una frecuencia de 44.1 KHz y a 16 bits por muestra (calidad de compact disc) ocuparía aproximadamente 10 MB.

2.3. Almacenamiento de archivos .wav

❖ *Método PCM (Pulse Code Modulation):*

- Las muestras son almacenadas en forma lineal.
- Consiste en tomar muestras del sonido a intervalos regulares dados por la frecuencia de muestreo y asignarle un número entero a cada muestra. La cantidad de valores distintos que puede tomar cada muestra depende de la resolución que se use (8 ó 16 bits).

❖ *Método ADPCM (Adaptive Delta Pulse Code Modulation):*

- Emplea mecanismos de compresión delta entre una muestra y otra.
- Comprime la información reduciendo el tamaño del archivo a 4 bits por sample. El método de compresión (compresión delta) consiste básicamente en almacenar sólo la diferencia entre dos samples en lugar de almacenar íntegramente cada uno de ellos.

2.4. Programas de edición de sonido

Al elegir un formato de sonido hay que tener en cuenta las posibilidades que se tiene de modificar el sonido original. En el caso de los archivos .wav,

estas posibilidades están dadas por los programas de edición de sonido digital. Estos programas se caracterizan por permitir grabar, editar, mezclar y reproducir archivos .wav, a la vez que ofrecen una representación gráfica del archivo a editar que puede ser ampliada para observar en detalle parte de la secuencia. Estas son las opciones básicas que pueden encontrarse en los programas de edición de sonido rudimentarios que suelen venir incluidos en programas más grandes.

Es el caso del SoundRecorder provisto con Windows 3.1, y del WaveEditor, que incluye "Video for Windows" como parte de sus Multimedia Tools. No obstante, existen programas específicos de edición de sonido como el Wave, de Turtle Beach, o el Sound Impression, de Microsoft, que presentan características mucho más avanzadas.

CAPÍTULO III
EL FORMATO MIDI

3.1. Descripción general

M.I.D.I. son las siglas de *Musical Instrument Digital Interface*, es decir, *Interconexión Digital para Instrumentos Musicales*.

Fue desarrollado en el año 1981 por varios fabricantes de sintetizadores, y orientado al mercado profesional. Hasta entonces, cada marca creaba modelos sólo compatibles con su propia gama. Pensando acertadamente que la estandarización ampliaría el mercado, desde hace muchos años, el M.I.D.I. se ha convertido en un estándar absoluto e indiscutible, si bien ampliado a cuentagotas con nuevas especificaciones. Hoy en día, incluso los teclados domésticos de gama media-baja incorporan conectores M.I.D.I.

Estos archivos se emplean para melodías y fondos musicales y ocupan muy poco espacio, pero no son válidos para la grabación y transmisión de voz, pues se oye como una computadora, pero son muy eficientes para muchos tipos de música y algunos tipos de sonidos.

Aún así, este formato no es apoyado suficientemente en la Web como los otros formatos.

Cada vez que una tecla del teclado de control es tocada, se transmite un mensaje digital, este mensaje define que nota fue tocada, que tan duro fue tocada (esto es llamado velocidad), y que nota fue tocada del canal MIDI.

MIDI tiene 16 canales y esta característica permite tocar a la vez múltiples instrumentos.

Los números usados en los mensajes MIDI cubren 128 valores en un rango de 0 a 127. Este número tiene lugar porque MIDI es un sistema de 8 bits. Por lo tanto, el más grande número binario disponible es 1111111 (o 127 en el sistema decimal). Estos números pueden ser usados para controlar muchos parámetros diferentes de las expresiones digitales incluyendo volumen, tremolo, pitch bend, etc.

Cuando las notas son tocadas en un teclado de control MIDI, los mensajes MIDI son enviados a la computadora y a los módulos de sonido uno después otro en forma serial. Incluso si las notas son tocadas al mismo tiempo, los mensajes son enviados uno a la vez con un retardo muy breve entre las notas. Este retardo es imperceptible bajo la mayoría de circunstancias.

El programa secuenciador recuerda cuales notas son grabadas y reproducidas de vuelta al mismo tiempo que usted graba notas adicionales. Cualquier nota grabada puede ser editada con gran flexibilidad.

Un módulo de sonido multitimbral es necesario para reproducir los mensajes MIDI en múltiples canales así que usted puede oír múltiples instrumentos a la vez.

Bajo las siglas M.I.D.I. se cobijan dos elementos: *un hardware y un software de comunicaciones*. En definitiva, se trata de un método estándar para interconectar dispositivos musicales electrónicos de todo tipo: sintetizadores, teclados maestros, módulos de sonido, samplers, cajas de ritmo, secuenciadores, mesas de mezcla, etc.

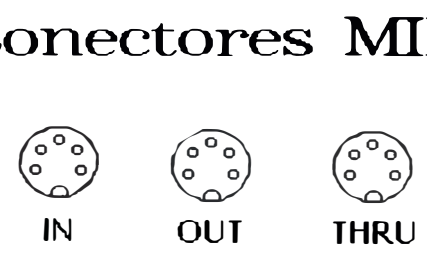


Figura 3.1: Conectores que debe poseer la interfase serie.

Por la parte física, tenemos una interfase serie de 32.5 Kbaudios, dotada de uno, dos o tres conectores DIN de 5 puntas. Todo aparato debe incorporar al menos uno de estos conectores, sea el de entrada (MIDI IN), salida (MIDI OUT) y/o retransmisión (THRU), éste último usado para conectar varios dispositivos en cadena (de OUT a IN, THRU a IN, de THRU a OUT, ...), de modo que la información del primer equipo se transmita a todos los demás.

En cuanto al software, se trata de un protocolo de comunicaciones muy simple, de 8 bits, en los que se codifican los aspectos más importantes de la interpretación musical. Es un sistema de tiempo real, por lo que la sincronización resulta fundamental. Para simplificar, los mensajes MIDI vienen a simbolizar *la misma información que una partitura*.

El primer bit toma valor 1 en los bytes de status, que indican el tipo de mensaje, y 0 en los de datos, que contienen los valores correspondientes. Los mensajes estándar usan de 1 a 4 bytes, y los de "sistema exclusivo" son de duración ilimitada (varias decenas de miles, por ejemplo).

3.2. Canales M.I.D.I.

El sistema M.I.D.I. estructura la información en *16 canales diferentes simultáneos*. Cada mensaje lleva implícito el número del canal al que afecta, de modo que no hay confusión posible.

Los canales vienen a significar diferentes instrumentos. Dado que los sintetizadores son multitímbricos (capaces de generar varios sonidos -timbres- diferentes a la vez), cada una de las partes del sintetizador debe saber qué notas debe interpretar, y con qué sonido. Así pues, podíamos considerar un sintetizador multitímbrico como un televisor capaz de dividir la pantalla en 16 partes, cada una de las cuales sintonizando una emisora diferente (o no). Cada parte puede interpretar la "partitura" de uno de los instrumentos de la pieza a interpretar: violín, piano, bajo eléctrico, batería, etc., simulando así una orquesta de 16 músicos.

3.3. Códigos M.I.D.I.

Figura 3.2: Cuadro sintético.

Hay mensajes MIDI encargados de la sincronización, de la interpretación, etc. No se citará todos, sino sólo los más habituales, que son los siguientes:

❖ ***Mensajes de canal***

Son independientes para cada uno de los 16 canales MIDI.

- *Nota pulsada/liberada (Note ON y Note OFF):* Cuando se pulsa una nota, se transmite la información de qué canal MIDI se está usando, qué nota fue pulsada, y con qué fuerza (velocidad MIDI). Al soltarla, se envía un mensaje indicando el canal MIDI y el número de la nota liberada. Algunos teclados avanzados (y muy caros) pueden también transmitir la velocidad con la que se ha soltado la tecla, lo que regula el *tiempo de release* (desvanecimiento) del sonido.

  Se nota que, a diferencia de la partitura, en MIDI *la duración de las notas no se envía como tal*, ya que la transmisión sucede en tiempo real; una nota dura el tiempo que transcurre entre la recepción de los mensajes de pulsación y liberación. Tampoco existe el concepto de acorde en sentido estricto: se produce un acorde cuando se reciben pulsaciones nuevas sin que se hayan liberado las anteriores notas.
- *Postpulsación (aftertouch):* También denominada *presión*, ya que sirve para expresar la mayor o menor presión aplicada sobre las teclas después de haberlas pulsado inicialmente. Esta característica es muy útil para simular el sonido de trompetas u otros instrumentos de viento, que producen una variación periódica de tono (vibrato) poco después del soplido inicial. También se usa para que el instrumentista pueda variar a su conveniencia el volumen de la nota mientras la interpreta.
- *Cambio de programa (Program change):* Se refiere a la posibilidad de indicar a cada parte del sintetizador multitímbrico qué sonido se le desea

asignar a partir de ahora. Así pues, una composición no se debe limitar a 16 instrumentos, aunque ése sea el límite de instrumentos simultáneos.

- *Cambio de control (Control change):* MIDI también permite transmitir información sobre la forma de la interpretación, así como datos adicionales. Existen 128 parámetros de control (controladores), y cada uno de ellos puede adoptar un valor de 0 a 127. Algunos de los controladores son estándar y otros quedan a disposición de los fabricantes para ser usados según sus necesidades. Sin embargo, normalmente los equipos sólo usan unos 15 o 20 diferentes. El controlador nº 7 es el volumen, independiente para cada canal MIDI, al igual que el nº 10 (panorama estéreo). Con estos dos se puede ajustar al gusto propio la mezcla final de todos los sonidos, al igual que se haría con una mesa de mezclas. Otros controladores sirven para indicar el uso de los diferentes pedales del piano, datos sobre el soplo en instrumentos de viento, etc.
- *Inflexión de tono (Pitch Bend):* Sirve para desafinar el sonido, simulando así el estiramiento de las cuerdas de una guitarra o similar.

❖ ***Mensajes de tiempo real***

Son aquellos destinados a la sincronización de secuenciadores, cajas de ritmo, y cualquier aparato capaz de grabar y reproducir una interpretación musical.

START (comenzar desde el principio de la "canción")

STOP (parar)

CONTINUE (continuar donde se paró la última vez)

CLOCK (este mensaje se envía 24 veces en la duración de una nota negra)

- ***Mensajes de sistema exclusivo***

No son estándar, aparte de la obligación de comenzar por el valor F0 (hexadecimal) y terminar por F7, así como no contener valores mayores de 127 (7 bits). Sirven para que los fabricantes codifiquen información específica de sus productos, como por ejemplo, la configuración de los parámetros de los sonidos, muestras digitalizadas (en los samplers), etc. A cada fabricante se le asigna un código distinto, y cada modelo tiene a su vez otro número identificativo. De este modo, si un aparato recibe un mensaje no destinado a él, lo ignora completamente.

3.4. Ampliaciones del M.I.D.I. original

Como es lógico, un estándar creado en 1981 necesita ciertas actualizaciones según pasa el tiempo, para adecuarlo a los nuevos medios aparecidos en el mercado. En particular, la explosión de los ordenadores personales, que ha dado un vuelco al mercado musical. Se comentará a continuación algunas de las ampliaciones a las especificaciones MIDI 1.0 aún vigentes.

- ***General MIDI (GM)***

Cada fabricante de sintetizadores utiliza las tecnologías que ha desarrollado o aquellas de las que dispone licencia. Así pues, en muchos casos, cada sintetizador suena distinto, y unos reproducen mejor determinado tipo de sonido que otros. Además, se fabrican equipos orientados a un determinado tipo de música (clásica, tecno, pop, etc.).

Por ambas causas, los sintetizadores organizan sus bancos de sonidos de diferente modo. Esto hace que el sonido nº 1, por ejemplo, pueda ser un piano en un determinado sintetizador, mientras que en otro es un oboe, una guitarra eléctrica, o un efecto especial.

Con la popularización de las secuencias y de los ficheros estándar MIDI, al reproducir una canción en un sintetizador distinto, el mapa de sonidos utilizado originalmente no era válido en otro equipo, por lo que la secuencia sonaba mal. En especial, cuando un ritmo, originalmente destinado a un sonido de batería, es reproducido en un instrumento afinado (por ejemplo, piano), el resultado es "torturador".

Para solucionar este problema, los fabricantes de sintetizadores decidieron adoptar un estándar consistente en que todos los sintetizadores fabricados bajo el sello GM usarían un conjunto común de 128 sonidos, organizados en 16 grupos de 8: pianos y órganos, guitarras, percusión afinada, cuerdas, viento madera, viento de metal, sintéticos, efectos especiales, etc. Además, crea un mapa de sonidos de batería (usando el canal MIDI nº 10) según el cual las diferentes partes de una batería y las percusiones también mantendrían la misma numeración en todos los equipos compatibles.

De este modo, si un compositor usa el banco de sonidos General MIDI, puede estar seguro de que su creación sonará de modo coherente en cualquier equipo GM.

El General MIDI no implica que todos los sintetizadores suenen igual, sino simplemente que la numeración de los sonidos es la misma. Es

decir, un equipo de mayor calidad tendrá mejores sonidos, más limpios, más potentes, más parecidos a los instrumentos que intenta simular, etc.

- ***General Standard (GS)***

La empresa Roland, una de las más importantes del mundo musical, considerando que el GM se quedaba corto, y a pesar de que la clasificación de los sonidos acordada en el GM era prácticamente la misma que había usado en sus últimos productos, desarrolló un nuevo estándar, llamado GS.

Este sistema es compatible con el GM en el sentido de que respeta todas sus normas, aunque añade nuevas características. Por ejemplo, para ampliar el abanico de sonidos y matices, admite variaciones de los 128 sonidos GM, (por ejemplo, añade instrumentos como el ukelele, y otros). También establece que todos los equipos GS deben incorporar un generador de efectos con reverberación y chorus, así como el control de algunos de los más importantes parámetros de los sintetizadores (ataque, decay, sustain, release, frecuencia de corte, etc.).

- ***Standard MIDI File (Fichero MIDI estándar)***

Dado que al principio cada programa de ordenador utilizaba su propio formato para almacenar las secuencias, no existía un método fácil para intercambiar piezas musicales entre un usuario y otro, ni entre un programa secuenciador y un editor de partituras, por ejemplo.

Esta carencia fue cubierta por la aparición del SMF (fichero estándar MIDI), que no sólo compatibiliza ficheros dentro de un tipo de ordenador, sino que se impone en todos los sistemas operativos, incluyendo el PC,

Mac, Amiga, Atari ST, e incluso la mayoría de los "secuenciadores hard" (es decir, no basados en ordenadores, sino en equipos independientes).

Así una secuencia creada en un sintetizador Korg 01WFD, dotado de una unidad de disquete de 3.5 pulgadas (formato PC de 720 KB) puede ser leída desde un PC, un Mac (usando Apple File Exchange, Access PC o cualquier software similar), o un secuenciador Roland.

- ***Sample Dump Standard (Volcado de muestras estándar)***

Los usuarios de samplers (aparatos que permiten digitalizar sonido para usarlo como base de sus instrumentos), debían perder todo su trabajo cuando compraban otro equipo, dado que el formato de codificación y almacenamiento de las muestras no era compatible.

Tras la aparición del SDS, todos los sintetizadores compatibles pueden intercambiar muestras, vía disquete o vía MIDI. Esto ha ampliado considerablemente la oferta comercial de disquetes y CD-ROM de muestras para todo tipo de samplers.

- ***MIDI Show Control (Control MIDI de los espectáculos)***

Después de controlar el equipo musical MIDI en directo, el siguiente paso ha sido hacerse cargo de los equipos musicales no MIDI (mesas de mezclas, magnetófonos, etc.), e incluso los no musicales (iluminación, proyectores, fuegos artificiales, telones, plataformas, etc.) que intervienen en un musical o teatral.

Así, músicos como Prince, Madonna, U2, Pink Floyd y otros, que realizan impresionantes y complejos espectáculos multimedia, necesitan de un sistema capaz de coordinar y sincronizar todos estos elementos.

Para ello se creó el MIDI Show Control, cuyas especificaciones contienen un lenguaje capaz de abarcar todos estos campos.

CAPÍTULO IV
LA PSICOACÚSTICA Y SU APROVECHAMIENTO EN EL PROCESO DE COMPRESIÓN DE SEÑALES DE AUDIO

4.1. Definición

La Psicoacústica puede ser definida como el estudio psicológico de la audición. También se define como la ciencia que se ocupa de la percepción auditiva del hombre.

El objetivo de la investigación psicoacústica es averiguar cómo funciona la audición, en otras palabras, el objetivo es descubrir cómo los sonidos que entran al oído son procesados por éste y el cerebro, con el fin de dar a la persona que escucha información útil acerca del mundo exterior.

La audición humana es un proceso extraordinariamente complejo, que recién comienza cuando el sonido golpea el tímpano y es convertido de variaciones en la presión del aire a impulsos nerviosos. De ahí en adelante, es asunto de la mente, y la Psicología se convierte en factor importante para estudiar y analizar los sonidos, así como las reacciones de las personas ante éstos.

La conexión de la Psicoacústica con la Psicología puede ser confusa. Muchos problemas tratados por los psicoacústicos tienen muy poco que ver con la concepción popular de la Psicología. Algunas investigaciones concernientes a la sonoridad y cómo ésta es representada por las células

nerviosas en el oído, podrían hacer pensar a algunas personas que es una materia concerniente a la Neurofisiología, y de hecho lo es. La Psicoacústica abordaría el problema midiendo la capacidad del oyente para hacer discriminaciones entre sonidos escogidos cuidadosamente.

Se considera a la Psicoacústica una rama de la Psicología básicamente porque trata de medir las reacciones en el comportamiento de las personas oyentes. La Psicoacústica no se ocupa sobre cómo los sonidos producen una respuesta emocional o cognoscitiva, sino que, como es un área muy amplia, existe un gran solapamiento con la Fisiología por un lado y se tiene que recurrir algunas veces a la Psicología clásica por el otro, para poder explicar los resultados experimentales más complejos.

Algunas de las investigaciones psicoacústicas más interesantes son:

- ¿Cuál es el mecanismo usado para lograr distinguir sonidos que ocurren simultáneamente?
- ¿Cómo se localiza el origen espacial de un sonido?
- ¿Cómo se determina el tono de, por ejemplo, un instrumento musical?

Para entender la forma en que los sonidos son procesados es necesario determinar las capacidades y limitaciones del oído humano. Cualquier dispositivo que produzca sonido con el propósito del disfrute humano debería tomar en cuenta lo que los oídos harán con ese sonido. Algunos de estos imperativos de diseño son mucho más que sentido común, y tampoco pueden ser analizados satisfactoria y completamente a través de una prueba de escucha casual. En muchos casos se requiere un conocimiento más profundo y real del problema.

¿Cómo diseñar un dispositivo de compresión de datos que reduzca la cantidad de información digital viajando por una línea telefónica sin afectar la calidad del sonido percibido?, ¿Cómo se determina un nivel seguro o confortable de exposición al ruido en una fábrica?, ¿Cómo diseñar un sistema de advertencia auditivo que sea claramente audible sobre el ruido de fondo sin ser causa de distracción?. Un buen conocimiento en Psicoacústica puede ser de gran ayuda en todos estos problemas de diseño.

Los estudios de la Psicoacústica revelan que nuestro oído, no está capacitado para percibir frecuencias "débiles" adyacentes a frecuencias "fuertes", en tanto que estas últimas cubren (en términos técnicos se dice: "enmascaran") las primeras. Las informaciones inherentes a las frecuencias más débiles por lo tanto, dado que estas últimas no serían percibidas por el oído humano, vienen eliminadas por el sistema MPEG durante la fase de compresión. De esta manera se obtiene una notable reducción del archivo de audio en términos de espacio físico ocupado.

Además existen varios niveles de compresión utilizados por el MPEG. Utilizando una capa de compresión más alta (por ejemplo 64 Kb/seg), el MPEG eliminará, además de las informaciones no audibles, también aquellas audibles pero menos importantes. Usando una compresión "más ligera" (por ejemplo 128 Kb/seg) no será posible percibir la diferencia entre la pieza con compresión en MPEG y el original.

4.2. Aspectos relevantes

Para conseguir tal reducción de la cantidad de datos, el formato MP3 usa unas pocas técnicas y trucos.

La percepción de la intensidad del sonido con respecto a la frecuencia no es lineal, lo que se aprovecha para la codificación, además el sistema auditivo puede ser descrito como un filtro pasabanda, que se subdivide en 26 frecuencias críticas hasta 24 KHz, estas bandas no son lineales, porque el ancho de banda varía de unos 50 a 100 Hz en frecuencias por debajo de 500 Hz y de hasta 5000 Hz en frecuencias altas.

El enmascaramiento simultáneo es un efecto en la frecuencia donde una señal de bajo nivel ("la enmascarada") puede ser inaudible (enmascarada) por una señal simultánea de más alto nivel ("la enmascaradora") tanto como estas dos señales estén tan cerca en la frecuencia. Esto es aprovechado por el codificador para eliminar estas bandas del espectro, sobreviviendo solamente la enmascaradora, lo que hace que se presente una compresión inicial.

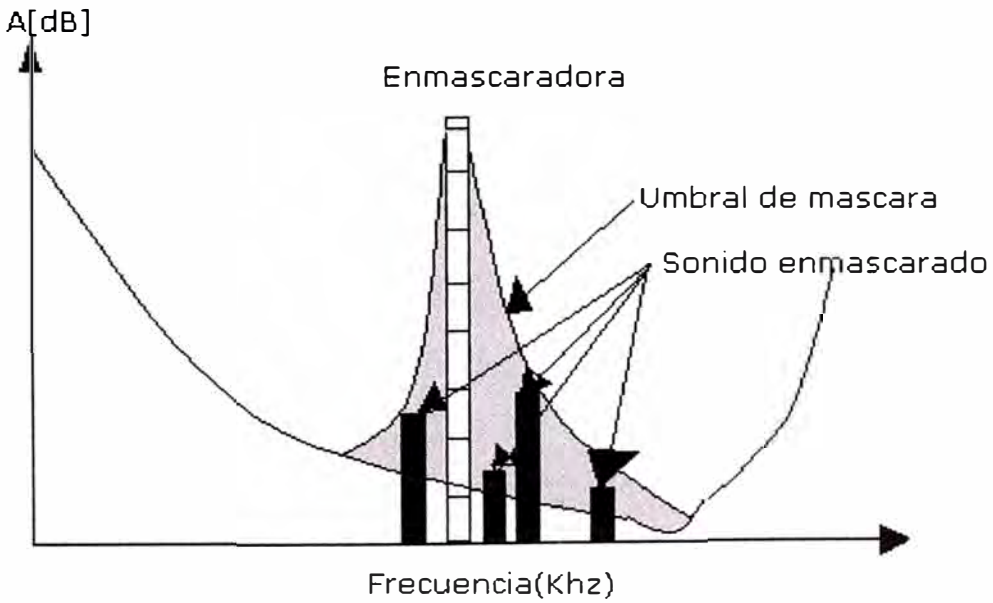


Figura 4.1: Enmascaramiento simultáneo.

Hay 4 aspectos que son los más interesantes y útiles para el desarrollo de un codificador MP3.

4.2.1. Mínimo umbral auditivo

Como se dijo anteriormente, el mínimo umbral auditivo no es lineal. Es representado, de acuerdo a la ley de Fletcher y Munson, por una curva de Intensidad (dB) contra Frecuencia (Hz), que posee niveles mínimos entre 2 KHz y 5 KHz, los cuales corresponden a la parte más sensitiva del oído humano. Este umbral es también conocido como umbral absoluto y corresponde al sonido de intensidad más débil que se puede escuchar en un ambiente silencioso. Por lo tanto, en los sistemas de compresión de audio que sacan provecho de la Psicoacústica, no es necesario codificar los sonidos situados bajo este umbral (el área por debajo de la curva), ya que éstos no serán percibidos.

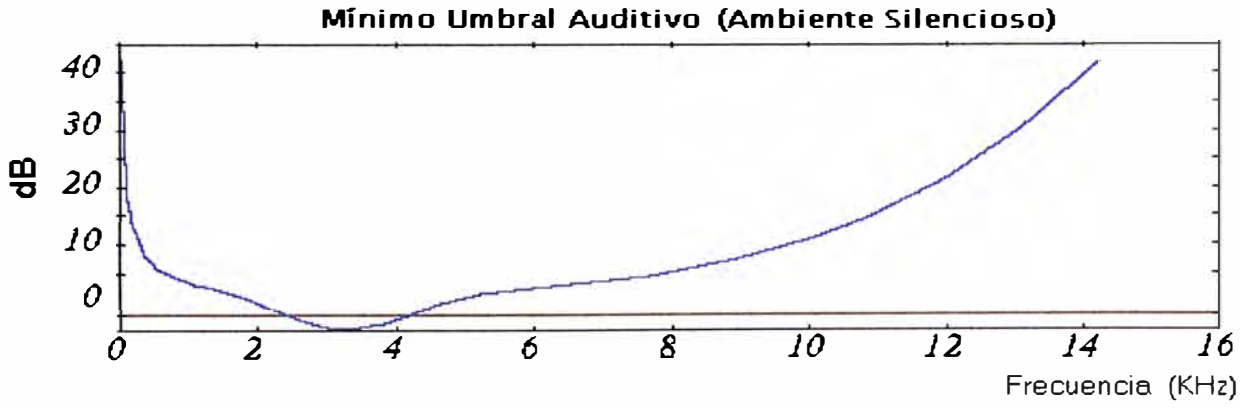


Figura 4.2: Mínimo umbral auditivo o umbral absoluto.

Qué sucede:

Frecuencias inferiores a los 2 KHz necesitarán de un volumen más alto para poder ser percibidas por el oído humano. La mayor sensibilidad está

entre los 2 KHz y los 4 KHz. Para poder oír las frecuencias desde los 6 KHz hacia arriba, se tendrá que incrementar su volumen según la curva mostrada en la figura 4.2.

4.2.2. Enmascaramiento

Partamos del asunto que, para el oído humano y la voz:

- El oído humano percibe las frecuencias que van desde 20 Hz hasta 20 KHz, y es más sensible entre los 2 KHz y los 4 KHz.
- El rango dinámico, desde el sonido más bajo percibido hasta el más alto, es de 96 dB.
- El rango de la voz humana varía desde los 80 Hz hasta los 10000 Hz.
- Vocales y tonos graves son bajas frecuencias.
- Consonantes son altas frecuencias.

El efecto de enmascaramiento se basa en las limitaciones del oído humano para responder a todas las componentes de un sonido complejo. Durante los sonidos fuertes, no se pueden oír los sonidos más débiles. Por ejemplo, cuando un músico organista no está tocando, se puede escuchar el resoplido de los tubos; y cuando el músico toca, se pierde el sonido de éstos porque ha sido enmascarado.

No es necesario por lo tanto codificar todos los sonidos. Esta es la primera propiedad usada por el formato MP3 para ganar algo de espacio. Para esto, el codificador MP3 usa un modelo psicoacústico para modelar el comportamiento del oído humano.

4.2.2.1. Enmascaramiento en frecuencia

Funciona de manera que un sonido en determinada frecuencia puede

enmascarar o disminuir el nivel de otro sonido en las frecuencias adyacentes, siempre y cuando el nivel del sonido enmascarante sea más alto (un sonido más intenso, más fuerte) que el nivel del sonido adyacente.

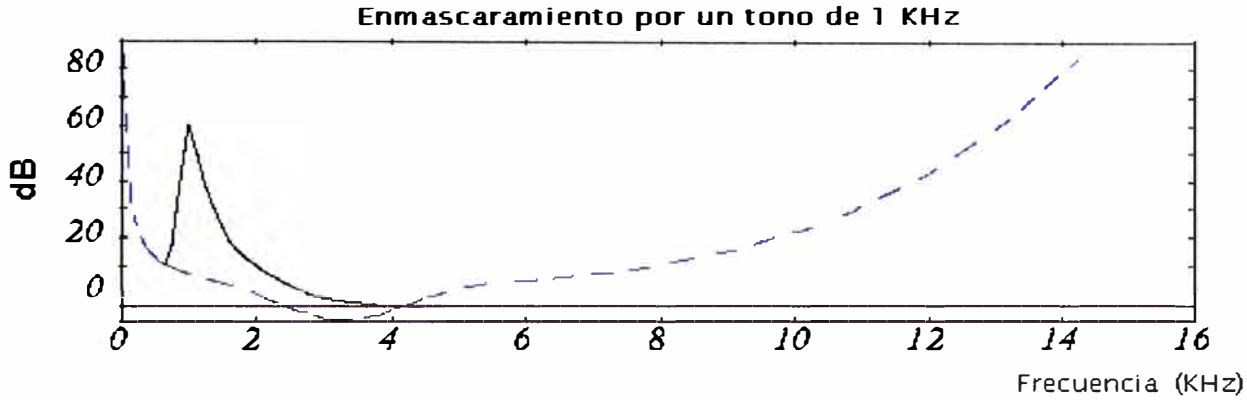


Figura 4.3: Enmascaramiento en frecuencia.

Qué sucede:

El tono fijo de 60 dB cubre el tono de prueba en las frecuencias inmediatamente anteriores y sobre todo en las frecuencias sucesivas. En condiciones normales de hecho, la percepción del tono de prueba sería igual al de la figura 4.2; la introducción de un segundo tono fijo a un volumen más alto, obliga a aumentar el volumen del tono de prueba a un cierto porcentaje para poder ser escuchado junto al tono fijo (tono enmascarante).

4.2.2.2. Enmascaramiento temporal

Se presenta cuando un tono suave está muy cercano en el dominio del tiempo (unos cuantos milisegundos) a un tono fuerte. Si se está escuchando un tono suave y aparece un tono fuerte, el tono suave será enmascarado por el tono fuerte, antes de que el tono fuerte efectivamente aparezca (pre-enmascaramiento). Posteriormente, cuando el tono fuerte desaparece, el

oído necesita un pequeño intervalo de tiempo (entre 50 y 300 ms) para que se pueda seguir escuchando el tono suave (post-enmascaramiento).

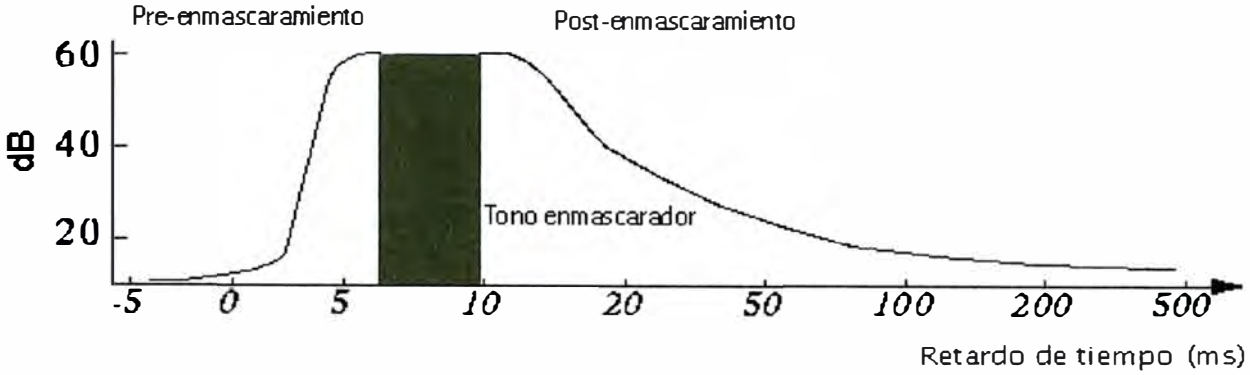


Figura 4.4: Enmascaramiento temporal.

Qué sucede:

Obtenemos que un tono más fuerte enmascara los tonos adyacentes más bajos pero igualmente audibles, dentro de un umbral de tiempo variable (igual a 5 ms, en la figura 4.4).

Con el post-enmascaramiento no hay problemas; pero el pre-enmascaramiento sugiere que un tono será enmascarado por otro tono, antes de que el tono enmascarador realmente aparezca, atentando contra el buen juicio de cualquier oyente. Para este fenómeno, se han presentado dos explicaciones:

1) El cerebro integra el sonido sobre un período de tiempo, y procesa la información por ráfagas en la corteza auditiva, o
2) Simplemente, el cerebro procesa los sonidos fuertes más rápido que los sonidos suaves.

Sin importar el mecanismo, el caso es que el pre-enmascaramiento

temporal en verdad existe, así sea exageradamente pequeño (se ha calculado con un valor aproximado de 30 ms).

En un sonido cualquiera, se presentan ambos tipos de enmascaramiento. El enmascaramiento en frecuencia es mucho más importante que el enmascaramiento temporal; aunque en ciertos dispositivos para compresión de audio se tienen en cuenta ambos tipos de enmascaramiento, con lo cual se logra mejor compresión de datos. Superponiendo ambas gráficas en una sola que presente tres ejes, se puede ver una curva bajo la cual están todos los sonidos que no pueden ser escuchados.

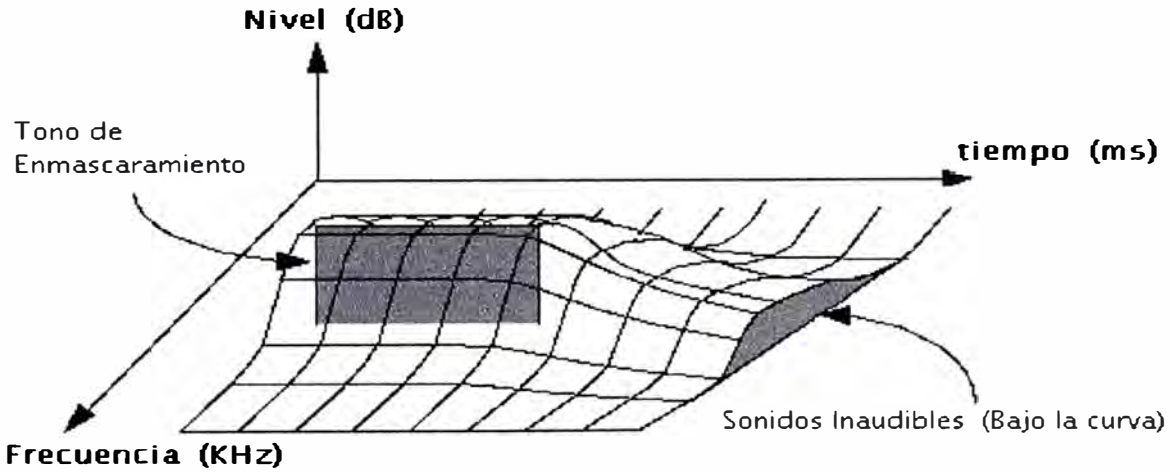


Figura 4.5: Superposición de los enmascaramientos en frecuencia y temporal.

Conclusión: Los factores que intervienen en el enmascaramiento son dos: enmascaramiento en frecuencia y enmascaramiento temporal.

4.2.3. Joint stereo o estéreo conjunto

En el caso de una señal de audio estereofónica, se puede lograr comprimirla, con base en la irrelevancia o redundancia entre ambos canales. Estas herramientas están referidas en la codificación Joint Stereo (JS), para

favorecer el encogimiento del tamaño del archivo comprimido.

En muchos aparatos de alta fidelidad de rango medio, hay un único subwoofer (alto parlante para sonidos bajos, en el equipo estereofónico). Sin embargo, se tiene la impresión de que el sonido proviene de diferentes fuentes como si existieran parlantes en todas las direcciones. En realidad, para frecuencias muy bajas y muy altas, el oído humano ya no es capaz de localizar el origen espacial de los sonidos con precisión total. El formato MP3 puede en consecuencia (opcionalmente) revertir usando un truco el cual es llamado Intensity Stereo (IS). De esta manera, algunas frecuencias se pueden grabar como señal monofónica seguida por un pequeño código para lograr restaurar un pequeño porcentaje de estereofonía en la decodificación.

La segunda herramienta del Joint Stereo es llamada Mid/Side (M/S) stereo. Cuando los canales izquierdo y derecho son muy similares, entonces los canales centro (L+R) y lateral (L-R) son codificados en vez del izquierdo y derecho. Esto permite reducir el tamaño del archivo final usando menos bits para el canal lateral. Durante la reproducción, el decodificador MP3 reconstruirá los canales izquierdo y derecho.

4.2.4. Las bandas críticas y el bark

4.2.4.1. Bandas críticas

Estudios de la discriminación en frecuencia del oído han demostrado que en las bajas frecuencias, tonos con unos cuantos Hertz de separación pueden ser distinguidos; sin embargo, en las altas frecuencias para poder discriminar los tonos se necesita que estén separados por cientos de Hertz. En cualquier caso, el oído responde al estímulo más fuerte que se presente

en sus diferentes regiones de frecuencia; a este comportamiento se le da el nombre de bandas críticas. Los estudios muestran que las bandas críticas son mucho más estrechas en las bajas frecuencias que en las altas; el 75% de las bandas críticas están por debajo de los 5 KHz, lo que implica que el oído recibe más información en las bajas que en las altas frecuencias. Las bandas críticas tienen un ancho de aproximadamente 100 Hz para las frecuencias de 20 a 400 Hz; este ancho aumenta de manera logarítmica a medida que aumenta la frecuencia. Se ha comprobado que el ancho de las bandas críticas se puede aproximar con la fórmula:

$$\textit{Ancho de la banda crítica (Hz)} = 24.7(4.37F + 1) \qquad \text{(ec. 4.1)}$$

F es la frecuencia central en KHz.

Las bandas críticas son comparables a un analizador de espectro con frecuencia central variable. Más importante aún es el hecho de que las bandas críticas no son fijas; son continuamente variables en frecuencia y cualquier tono audible creará una banda crítica centrada en él. Mirado desde otro punto de vista, el concepto de la banda crítica es un fenómeno empírico: una banda crítica es el ancho de banda al cual las respuestas subjetivas cambian abruptamente.

4.2.4.2. El bark

El bark (en honor al físico alemán Georg Heinrich Barkhausen) es la unidad de frecuencia perceptual; específicamente, un bark mide la tasa de banda crítica, o sea, una banda crítica tiene un ancho de un bark. La escala bark relaciona la frecuencia absoluta (en Hz) con las frecuencias medidas

perceptualmente (el caso de las bandas críticas). Usando el bark, un sonido en el dominio de la frecuencia puede ser convertido a sonido en el dominio psicoacústico. De esta manera, un tono puro (representado por una componente en el dominio de la frecuencia) puede ser representado como una curva de enmascaramiento psicoacústico. Eberhard Zwicker modeló el oído con 24 bandas críticas arbitrarias para frecuencias por debajo de 15 KHz, con una banda adicional que ocupa la región entre 15 y 20 KHz. El bark (ancho de una banda crítica) puede calcularse con las siguientes fórmulas:

$$\begin{aligned} 1\ \text{bark (Hz)} &\cong f/100 && \text{para} \quad f < 500\ \text{Hz} \\ 1\ \text{bark (Hz)} &\cong 9 + 4\log\left(f/1000\right) && \text{para} \quad f > 500\ \text{Hz} \end{aligned} \qquad \text{(ec. 4.2)}$$

f = frecuencia

Se hace notar que, para determinar el ancho de una banda crítica, se pueden usar tanto la fórmula 4.1 como las 4.2.

De las consideraciones anteriores, se deduce que el umbral de enmascaramiento es diferente cuando se tienen en cuenta las bandas críticas. El umbral sin tener en cuenta las bandas críticas, sería:

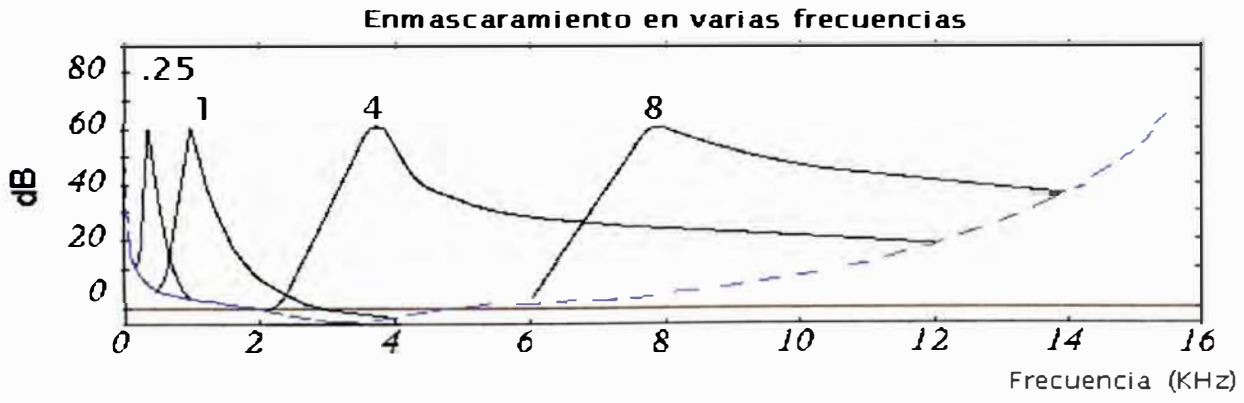


Figura 4.6: Enmascaramiento en varias frecuencias.

En esta gráfica se evidencia el hecho de que existen múltiples efectos enmascarantes.

Y teniendo en cuenta las bandas críticas:

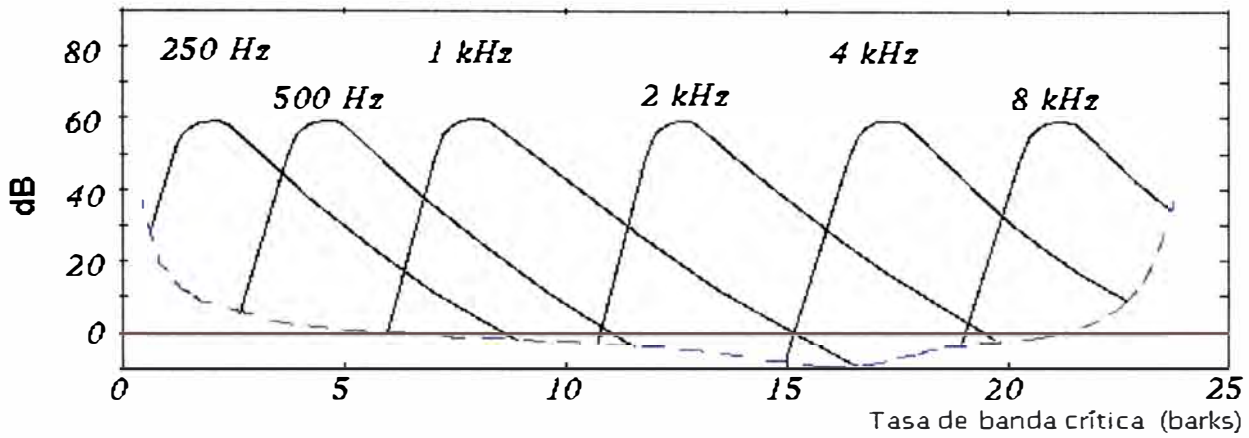


Figura 4.7: Umbral teniendo en cuenta las bandas críticas.

4.3. Conclusión

Con lo mostrado acerca de la Psicoacústica, se concluye que no todos los sonidos tienen la misma relevancia. Estas propiedades son usadas por los mecanismos de compresión de audio para disminuir la cantidad de datos necesarios para representar un sonido, basándose en un modelo psicoacústico que simula el comportamiento del oído humano. Se toma ventaja de las limitaciones del oído (así como del cerebro) para responder a todas las componentes en una onda de audio compleja y, de esta manera, lograr que los mecanismos de compresión calculen lo que se oirá de un sonido particular, descartando el material indetectable o codificándolo con menos precisión, idealmente sin cambiar la calidad del sonido percibido.

La Psicoacústica es la disciplina que estudia la forma en que las

vibraciones que llegan al cuerpo humano afectan a la mente y su respuesta. De ahí viene el primer punto importante, el hecho de que todo el cuerpo es capaz de percibir vibraciones. Y es importante, porque si bien nuestro órgano dedicado a la recepción de vibraciones del aire (sonido) es el oído, en muchísimas ocasiones son más impactantes los efectos generados en otras partes del cuerpo.

La Psicoacúistica existe como fenómeno desde siempre. Incluso sus raíces vienen aún antes de la existencia del hombre y de ahí, que las reacciones que podamos tener son reflejos que llevamos grabados genéticamente. Esto es, claro, bajo el ejemplo del sonido de un león. Este sonido aparte de poder alcanzar niveles de presión sonora impresionantes, tiene componentes de baja frecuencia que provocan resonancia en el pecho y las zonas viscerales, creándonos una reacción instantánea de miedo y esto debido a que nuestro sistema neurológico compara inmediatamente este sonido con los que tenemos codificados en nuestras "librerías" de sonidos que, como se mencionó, tenemos grabadas genéticamente y que vamos aumentando durante toda nuestra vida. Los primeros indicios de la búsqueda de manera consciente de ciertas reacciones humanas a través de vibraciones, se remontan a las épocas tribales, donde las danzas que surgen como reacción a sonidos rítmicos, estimulaban fuertemente a los individuos de la tribu para las diferentes actividades que desempeñaban, como darse valor para ir de cacería.

La Psicoacústica como disciplina de estudio, tiene fecha reciente y marca el estudio de la recepción de vibraciones, interpretación de las mismas, etc.

CAPÍTULO V
EL FORMATO MP3

5.1. Definición

MP3 es la abreviatura de MPEG 1 layer 3, o lo que es lo mismo: Algoritmo de codificación perceptual desarrollado por el consorcio MPEG (Moving Picture Expert Group), junto con el Instituto Tecnológico Fraunhofer que finalmente se ha estandarizado como norma ISO-MPEG Audio Layer 3 (IS 11172-3 y IS 13818-3) y que viene a ser un avance importante sobre los anteriores (Layer 1 y 2).

En pocas palabras; el MP3 es un algoritmo de compresión de audio con el que se puede grabar sonido en calidad CD con una compresión de 1 a 12.

Su finalidad es la de comprimir lo máximo posible un archivo de audio, manteniendo inalterable, en lo que sea posible, la calidad. De esta manera los archivos se vuelven fácilmente transportables en disquettes, o transferibles a través de Internet.

El algoritmo de compresión del MP3 se basa en las limitaciones del oído humano, que sólo es capaz de captar frecuencias entre 20 Hz y 20 KHz (es más sensible entre 2 y 4 KHz), y elimina las frecuencias inaudibles conservando la esencia del sonido. Al obtener un MP3 es posible seleccionar el nivel de codificación y compresión que se desea. Obviamente, a mayor compresión, menor calidad. A 128 Kbits/seg y 44 KHz estéreo se

consigue un buen equilibrio entre compresión y calidad y es el nivel que encontraremos por defecto en los compresores y en las canciones disponibles en la red.

En otras palabras, el algoritmo de compresión MP3 descompone la onda en sus armónicos elementales y destruye aquellos menos significativos. La destrucción es selectiva y determinada por el usuario a través del bitrate que se quiera obtener. El bitrate como unidad de caudal, indica el número de bits por segundo que se van a transmitir por un medio de transmisión. Cuanto menor sea el bitrate, menor será el tamaño de la señal ya comprimida, menos selectiva será la destrucción de armónicos y menor calidad tendrá el sonido obtenido.

Es el formato más difundido de compresión de archivos de audio en Internet, el cual garantiza una alta calidad de sonido. Este formato de audio se ha usado normalmente para "mover" canciones enteras en Internet en un espacio de tiempo muy reducido. El éxito del formato está basado sobre tres pilares:

- *Capacidad de compresión:* El MP3 es un formato de compresión de audio que logra reducir hasta 12 veces el tamaño de una canción. Un tema de 5 minutos sin comprimir ocupa 60 Megabytes (un tamaño bastante molesto para cualquier disco duro); si se convierte esa canción en un archivo MP3 ocupará solo 5 Megabytes. Otro ejemplo: mientras que en un disco duro de 1 GB entrarían 16 canciones sin comprimir, en la misma cantidad de espacio caben 200 archivos MP3. Es posible crear un disco DVD con más de 80 horas de música y un CD grabable puede contener

once CD's de audio (unas 150 canciones).

- *Calidad de compresión:* Un método de compresión no es otra cosa que un algoritmo (serie de ecuaciones matemáticas) capaz de simplificar la información. Pero compresión indica, inevitablemente, una pérdida de calidad. Pues bien, el MP3 logra un equilibrio casi perfecto: logra reducir el tamaño de una canción con una mínima pérdida de calidad sonora.
- *Facilidad de distribución:* La tercera clave fundamental para entender el éxito y la aceptación mundial del MP3 es su facilidad de distribución. Gracias a Internet se puede enviar o recibir un archivo de este tipo sin demasiado esfuerzo. Una canción de tres minutos (con un tamaño promedio de 3 megabytes) se puede distribuir por Internet sin mayor problema (incluso con conexiones lentas).

Debido a la cantidad y tipo de cálculos necesitados para llevar a cabo la compresión, el mínimo para trabajar con MP3 es un procesador Pentium a 133 MHz o superior con 32 MB de RAM. Para reproducirlo es necesario un ordenador como mínimo Pentium de 75 MHz y 16 MB de RAM, aunque últimamente han aparecido unos aparatos similares a los walkman, que son capaces de reproducir MP3.

5.2. Un poco de historia

Dada la existencia de un estándar JPEG (Join Photographic Experts Group) para la reducción de la rata de bits de imágenes estáticas y su adecuado uso en las imágenes en movimiento, se crea MPEG (Moving Pictures Expert Group) para idear un esquema de codificación conveniente para la transmisión y grabación de estas imágenes en un formato digital,

como son CD-ROM, CD-i y Video CD.

En el año 1987 los investigadores del Instituto Fraunhofer, en Erlangen, Alemania, comenzaron a trabajar en la codificación perceptual de audio en el marco del proyecto EUREKA EU147, Digital Audio Broadcasting (DAB). En cooperación con la Universidad de Erlangen (Prof. Dieter Seitzer), el IIS (*Fraunhofer Institut Integrierte Schaltungen,* Instituto para el Circuito Integrado Fraunhofer) finalmente desarrolló un poderoso algoritmo de compresión que fue estandarizado como ISO-MPEG Audio Layer 3.

El Instituto Fraunhofer se propuso desarrollar un método para transmitir audio en un formato digital comprimido. Idearon un algoritmo (codec) capaz de comprimir el sonido sin una pérdida de calidad apreciable. En 1992, el MPEG aprobó la tecnología y nació el MP3 (MPEG1 Audio Layer 3 - 3er nivel de compresión del MPEG1). La verdadera revolución llegó con el crecimiento espectacular de Internet: cualquiera podía descargar una canción a su PC ignorando lo que se conoce como *copyright*.

Del encuentro entre la ISO (International Standards Organisation) y la IEC (International Electrotechnical Commission) en noviembre de 1991 resulta un estándar para la codificación de audio y video, conocida como MPEG 1 (ISO/IEC 11172). Posteriormente, en noviembre de 1994 nace MPEG 2 (ISO/IEC 13818) como un nuevo estándar, después del encuentro de ISO e IEC en Singapore.

MPEG 1: Este estándar consiste de cuatro partes: Sistema, video, Audio y Adecuación. La parte de audio, ISO/IEC 11172-3, define tres algoritmos: Capa 1, 2 y 3 para la codificación de señales de audio PCM con frecuencias

de muestreo de 32, 44.1 y 48 KHz, en ratas de bits desde 32 hasta 448 Kbits/seg.

5.3. Como se crea un archivo MP3

Podemos convertir a un archivo MP3 cualquier canción o sonido que deseemos. Para ello, debemos partir de cualquiera de las siguientes fuentes:

- Una pista de CD-Audio
- Un fichero ya digitalizado en formato WAV.

En este segundo caso, la fuente original podría ser un disco LP, una cinta de cassette, la radio o banda sonora de una cinta o cámara de video, o lo que estemos diciendo en un micrófono.

Para pasar cualquiera de estas fuentes de sonido a un archivo en formato WAV, nos basta con usar un programa de grabación de sonido, como la misma grabadora de Sonidos de Windows, o programas shareware de calidad como Cool Edit, Goldwave, u otros.

Existen programas que permiten comprimir a MP3 pistas de CD-Audio directamente, o mediante el uso intermedio temporal de un archivo WAV. En el contexto del MP3, a estos programas se les denomina habitualmente *"Todo en Uno"*.

Otros programas sólo permiten comprimir a MP3 partiendo del archivo WAV, son los llamados *compresores (coder)*.

Para complementar a los anteriores, se usan programas que convierten una pista de audio de un CD a un archivo en formato WAV, y reciben el nombre de *extractores de audio digital (ripper)*.

En la siguiente figura se puede ver un esquema de todas estas

posibilidades:

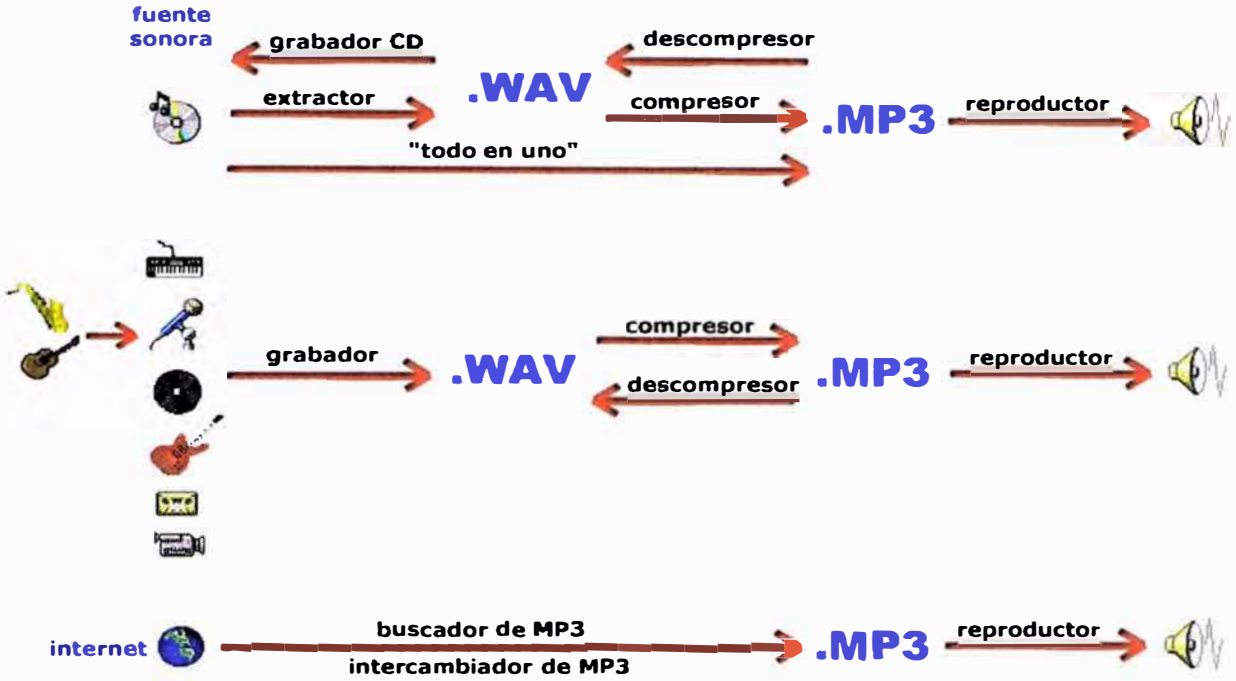


Figura 5.1: Maneras de crear archivos MP3.

5.4. Audio MPEG

Estándar MPEG-1:

PARTE	DESCRIPCIÓN
Parte 1	Sistema (Multiplexación y control para sincronización del video, el audio y la información secundaria).
Parte 2	Codificación del video.
Parte 3	Codificación del audio.
Parte 4	Pruebas del sistema.
Parte 5	Simulación por *software*.

Tabla 5.1: Estándar MPEG-1.

Estándar MPEG-2:

PARTE	DESCRIPCIÓN
Parte 1	Sistema (Multiplexación y control para sincronización del audio y video).
Parte 2	Codificación del video.
Parte 3	Codificación del audio (*Backwards Compatible*, Compatible con el audio MPEG-1).
Parte 4	Pruebas del sistema.
Parte 5	Reportes técnicos.
Parte 6	DSM-CC: *Digital Storage Media-Command and Control*, Medios de Almacenamiento Digital-Comando y Control.
Parte 7	AAC: *Advanced Audio Coding*, Codificación Avanzada de Audio (No compatible con el audio MPEG-1).
Parte 8	Fue abandonada cuando se comprobó que no había interés de la industria. Intentó codificar el video cuando las muestras de entrada son 10 bits.
Parte 9	RTI: *Real Time Interface*, Interface en tiempo real.
Parte 10	Pruebas del DSM-CC.

Tabla 5.2: Estándar MPEG-2.

En el Anexo B se han reseñado y ampliado todas las partes de MPEG-1 y MPEG-2 que corresponden a estándares internacionales, pero al audio sólo corresponden las siguientes partes: la parte tres de ambos estándares; y la parte siete del MPEG-2, Codificación Avanzada de Audio (AAC, Advanced Audio Coding), a la que también se refieren como MPEG-2 Capa 7 o MPEG-2 NBC (Non-Backwards Compatible) ya que es incompatible con el MPEG-1 y con los otros formatos del MPEG-2. La parte 3 está organizada como se muestra en la siguiente figura:

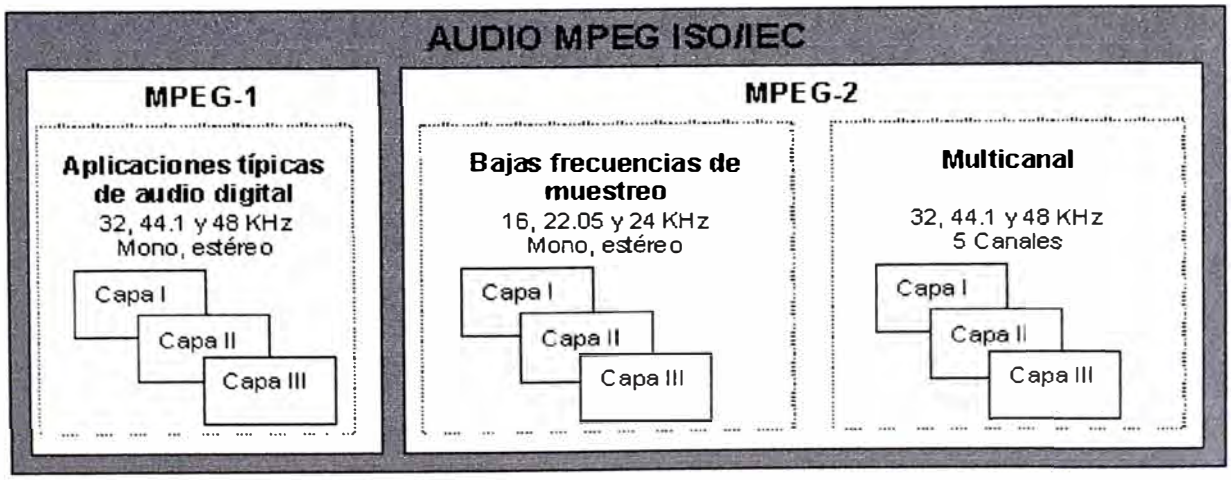


Figura 5.2: Organización de la parte 3 del MPEG-1 y MPEG-2.

En la tabla que se muestra a continuación, se resumen las características de cada nivel para el audio MPEG-1:

CAPA	FRECUENCIA DE MUESTREO	TASA DE BITS
Capa I	32, 44.1 ó 48 KHz	De 32 a 448 Kbps
Capa II		De 32 a 384 Kbps
Capa III		De 32 a 320 Kbps

Tabla 5.3: Características de cada nivel.

En la siguiente tabla comparativa, donde "layer" es sinónimo de capa, se aprecian las tasas de compresión para obtener sonido estéreo en calidad "casi CD":

Formato	Compresión	Kb/seg
Layer 1	4 a 1	384
Layer 2	6 a 1	256
	8 a 1	192
Layer 3	10 a 1	128
	12 a 1	112

Tabla 5.4: Tasas de compresión.

El audio de los CD's (frecuencia de 44.1 KHz, codificación a 16 bits, modo estéreo) requiere de una tasa de transferencia (ancho de banda) muy cercano a 1.5 Mbps:

$$44100 \frac{muestras}{segundo} x\ 16 \frac{bits}{muestra} x\ 2\ canales = 1411200\ bps = 1.34582519\ Mbps$$

Sólo ocho segundos de audio consumen aproximadamente 1.5 MB de espacio de almacenamiento. Empleando el primer esquema de codificación, Capa I, la calidad CD se alcanza con una tasa de apenas 384 Kbps; éste fue

el esquema usado en el DCC (*Digital Compact Cassette,* Casete Compacto Digital) de la firma Philips. Adopta exclusivamente el método de eliminación de las frecuencias enmascaradas, derivado de los estudios de Psicoacústica. Como se explicó en el capítulo anterior, eso significa que elimina aquellas frecuencias que son escondidas debajo de otras que tienen más amplitud de potencia.

El siguiente esquema, Capa II, lograba la calidad CD con apenas 192 Kbps; su uso más común fue en estaciones de radio digital en Norteamérica, donde se conoció como MUSICAM. Adopta métodos de filtrado de señales de audio más avanzados respecto al primer modelo. Fue mejorado el método de escogencia y de eliminación de las frecuencias que no son necesarias. Codificando a 160 Kbps se obtiene una buena calidad de sonido; a 192 Kbps resulta difícil notar la diferencia del original; a 256 Kbps también los más fanáticos oyentes serán satisfechos.

Pero la popularidad del audio comprimido explotó con la aparición de la Capa III; el famoso MP3, que sólo necesita 128 Kbps para lograr la calidad de audio CD. Así que un minuto de alta calidad requiere apenas 1 MB de espacio de almacenamiento; en 650 MB (la capacidad de un CD) se pueden almacenar hasta 11 horas de música.

Es el modelo más complejo de los tres. No solo adopta filtrados más macizos respecto a la Capa II, sino que utiliza un coder extremadamente complejo (Huffman). Codificando la señal a 112 Kbps se obtiene un sonido suficientemente bueno; a 128 Kbps se está muy cerca del original; a 160 Kbps y a 192 Kbps, no es posible percibir diferencias con el original.

Las principales diferencias entre las tres capas son:

- El incremento en la complejidad del codificador y el decodificador (Capa I más simple, Capa III más compleja), especialmente verdadero para el codificador de la Capa III.
- El mayor tiempo de respuesta al hacer la evaluación del codificador-decodificador (codec). La Capa III consume más tiempo para realizar el proceso de generar y/o leer los flujos de bits codificados.
- Y el incremento en el desempeño (Capa I más bajo desempeño, Capa III más alto desempeño; teniendo en cuenta que la tasa de bits permanece constante). Es decir, el audio codificado a 96 Kbps usando el esquema de la Capa III tiene más calidad que si se codifica a 96 Kbps con el esquema de la Capa II, o con el de la Capa I.

La figura 5.3 muestra la vecindad de las diferentes capas a la calidad CD (100%).

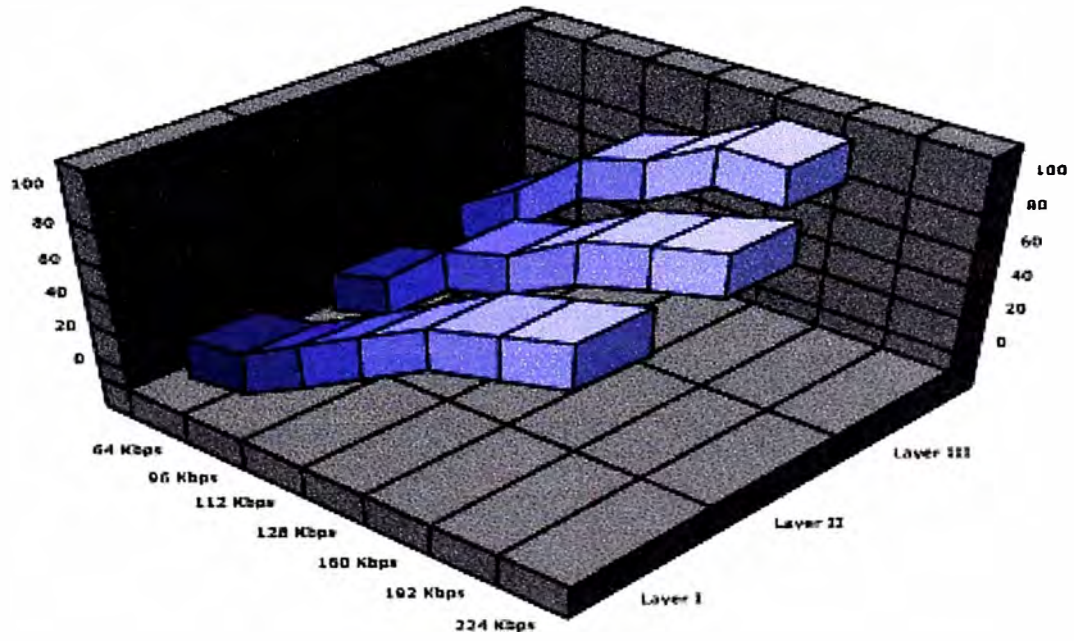


Figura 5.3: Cercanía de las diferentes capas a la calidad CD.

Y por otro lado, podemos conseguir un nivel de reducción como el siguiente para diversas calidades:

calidad del sonido	ancho de banda	modo	bitrate	ratio de compresión
sonido telefónico	2.5 kHz	mono	8 kbps	96:1
mejor que onda corta	4.5 kHz	mono	16 kbps	48:1
mejor que radio AM	7.5 kHz	mono	32 kbps	24:1
similar a radio FM	11 kHz	estéreo	56...64 kbps	26...24:1
cercano al CD	15 kHz	estéreo	96 kbps	16:1
CD	>15 kHz	estéreo	112..128kbps	14..12:1

*8 Kbps: El Instituto Fraunhofer usa una extensión NO-ISO en el MPEG Capa III (a la que bautizó como "MPEG-2.5" para mejorar el desempeño.

Tabla 5.5: Tabla tomada del Instituto Tecnológico Fraunhofer.

Para la segunda generación de los estándares, MPEG-2, se introdujo la extensión para las bajas tasas de muestreo, que apuntan a las aplicaciones con muy baja tasa de bits y con requerimientos limitados de ancho de banda (las nuevas frecuencias de muestreo son 16, 22.05 y 24 KHz; mientras que las tasas de bits disminuyen hasta 8 Kbps).

Además de lo anterior, también se añadió una extensión multicanal a las frecuencias de muestreo tradicionales (32, 44.1 y 48 Khz) con capacidad para cinco canales principales de audio (izquierdo, centro, derecho, surround izquierdo y surround derecho); opcionalmente se puede añadir un canal adicional que mejora la calidad en las bajas frecuencias que manejan las señales subwoofer.

Y adicionalmente, se incluyó una extensión pensada para múltiples idiomas que permite siete canales más comentarios (sólo se envía un canal de video junto con siete diferentes lenguajes), ahorrando, de esta manera, gran cantidad de ancho de banda. A los canales de comentarios les es permitido tener una tasa de muestreo correspondiente a la mitad de la tasa de muestreo usada en el canal de alta fidelidad.

También es importante mencionar que un decodificador MPEG-1 creado para la Capa III, debe ser capaz de manejar flujos de bits codificados con los esquemas de las Capas I y II. Y debido a que MPEG-2, en su primera fase, usa la misma familia de codificadores y decodificadores de audio, entonces puede decodificar flujos de bits MPEG-1, sin importar el esquema (capa) de codificación.

La compresión MPEG-1 Capa III consta de las siguientes fases:

- *Audición mínima:* Según la ley de Fletcher y Munsen no hace falta codificar los sonidos entre 2 KHz y 5 KHz puesto que no se perciben correctamente.
- *Efecto de Máscara:* Como se dijo anteriormente, cuando se oyen sonidos fuertes, estos enmascaran sonidos débiles.
- *Reserva de bytes:* Muchas veces, algunas partes de una canción no pueden ser codificadas a cierta velocidad sin alterar la calidad musical. El MP3 entonces utiliza un reservorio corto de bytes que actúa al usar partes que pueden ser codificadas a una velocidad inferior del flujo de corriente como un buffer.
- *Codificación tipo Huffman:* Codificación por el algoritmo Huffman. Por

tabla, como muchos sonidos son iguales es una gran codificación, logrando reducir un 20% del espacio. Gracias a que los sonidos están enmascarados son más fáciles aun de usar con este tipo de codificación.

5.4.1. Codificador psicoacústico

Un codificador psicoacústico por subbandas, del mismo tipo que se usa en el audio MPEG-1, se muestra en el siguiente diagrama de bloques:

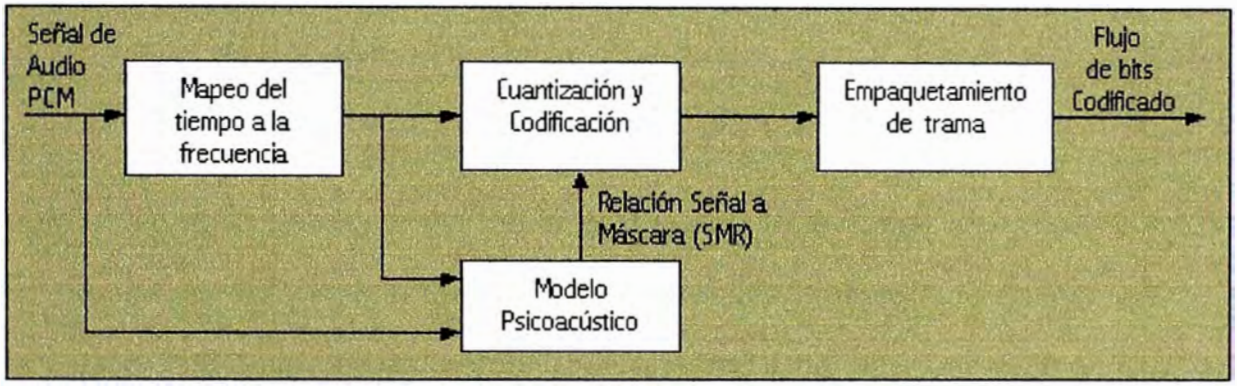


Figura 5.4: Codificador psicoacústico por subbandas.

MPEG-1 Capa I: Para la Capa I, el bloque "MAPEO DEL TIEMPO A LA FRECUENCIA" es un banco de filtros polifásico basado en la DCT (Transformada Discreta del Coseno), el cual divide el audio en 32 subbandas igualmente espaciadas en frecuencia (es equivalente a implementar filtros pasabanda), cada una aportando 12 muestras para un total de 384 muestras, las cuales se incluyen en lo que ha recibido el nombre de "TRAMA".

El "MODELO PSICOACÚSTICO" sólo usa enmascaramiento en frecuencia, por medio de una FFT de 512 puntos. La salida de la FFT se usa para encontrar tanto el enmascaramiento tonal (sinusoidal) como el no-tonal

(ruido) de la señal. Cada componente de enmascaramiento produce un umbral de enmascaramiento dependiente de su frecuencia, intensidad y tonalidad. Para cada subbanda, los umbrales de enmascaramiento individuales se combinan para formar el umbral de enmascaramiento global. El umbral de enmascaramiento se compara con el máximo nivel de señal para la subbanda, produciendo una relación señal a máscara (SMR, Signal-to-Mask Ratio), que es la entrada al siguiente bloque (cuantizador).

El "CUANTIZADOR/CODIFICADOR" primero examina las muestras de cada subbanda, encontrando el valor máximo absoluto de esas muestras, y realizando la cuantización con seis bits. Estos seis bits se llaman "FACTOR DE ESCALA PARA LA SUBBANDA". Luego se determina la repartición de bits para cada subbanda minimizando la SMR total con respecto a los bits repartidos para cada subbanda. Por último, las muestras subbanda son linealmente cuantizadas teniendo en cuenta la repartición de bits para esa subbanda.

El "EMPAQUETAMIENTO DE TRAMA" se encarga de formar un flujo MPEG válido. Cada trama comienza con información del encabezado para sincronización y control, además de un CRC (Código de Redundancia Cíclica) para detección y corrección de errores. Cada una de las 32 subbandas usa cuatro bits para describir la repartición de bits (nivel de cuantización) y seis bits para el factor de escala. Los bits restantes en la trama se usan para las muestras subbanda, con un espacio opcional para información extra.

A 48 KHz, cada trama en la Capa I lleva información de 8 ms de audio. La

calidad más alta se logra a 384 Kbps. Aplicaciones típicas de la Capa I incluyen grabación digital en cintas, discos duros o discos magneto-ópticos, los cuales soportan la alta tasa de bits.

MPEG-1 Capa II: El "MAPEO DEL TIEMPO A LA FRECUENCIA" es el mismo de la Capa I, un banco de filtros polifásicos tipo DCT con 32 subbandas; pero usa tres conjuntos de muestras en el filtro (anterior, actual, próximo) logrando 36 muestras por subbanda para un total de 1152 muestras subbanda.

El "MODELO PSICOACÚSTICO" es similar al de la Capa I, pero usa una FFT de 1024 puntos para una mejor resolución en frecuencia, y añade un poco de enmascaramiento temporal. El procedimiento es igual al de la capa anterior para producir las SMR para cada una de las 32 subbandas.

El "CUANTIZADOR/CODIFICADOR" genera factores de escala de seis bits para cada subbanda, igual que en la Capa I. Sin embargo, las tramas de la Capa II son tres veces más largas que aquellas de la Capa I, así que la Capa II permite tres factores de escala sucesivos para cada subbanda, y el codificador usa 1, 2 o los 3 factores, dependiendo de cuánto difieran entre sí. De esta manera se logra, en promedio, reducir a la mitad la cantidad de bits que se usan para los factores de escala, en comparación con el consumo de bits para los factores de escala usado con el esquema de la Capa I. La repartición de bits se computa de manera similar a la repartición que se realiza en la Capa I.

El "EMPAQUETAMIENTO DE TRAMA" usa la misma estructura de encabezado y CRC de la Capa I. Sin embargo, el número de bits usados

para describir la repartición de bits varía con la subbanda: cuatro (4) bits para las subbandas bajas, tres (3) bits para las subbandas medias y dos (2) bits para las subbandas altas (debido a que se tiene en cuenta el ancho de las bandas críticas). Los factores de escala (1, 2 ó 3 dependiendo de los datos) se codifican junto con un código de 2 bits que describe cuál combinación de factores de escala se está usando. Las muestras subbanda son cuantizadas de acuerdo a la repartición de bits, y luego combinadas en grupos de tres (llamados gránulos). Cada gránulo se codifica con un código especial. Esto permite a la Capa II capturar mucha más información irrelevante en la señal, en comparación con la Capa I.

A 48 KHz, cada trama en la Capa II lleva 24 ms de audio. La calidad más alta se alcanza a tasas de 256 Kbps, aunque a 64 Kbps tiene un nivel aceptable. Las aplicaciones de esta capa incluyen radiodifusión de audio, grabación profesional y multimedia.

MPEG-1 Capa III: La Capa III es mucho más compleja que la Capa II. Usa un banco de filtros híbridos conmutado (conformado por un filtro polifásico DCT similar al de la Capa II y por una transformación MDCT (Transformada Discreta del Coseno Modificada)) que ayuda a incrementar la resolución en frecuencia, permitiendo dividir el audio en bandas que se ajustan a las bandas críticas del oído (no hay igual espaciamiento entre subbandas). Emplea un modelo psicoacústico que incluye los efectos totales del enmascaramiento tanto en la frecuencia como en el tiempo. Utiliza un sofisticado esquema de codificación por entropía y cuantificación no uniforme donde se involucran la redundancia estéreo y los códigos de

Huffman, permitiendo crear tramas de longitud variable. El empaquetamiento de trama incluye el bit reservoir, que permite usar más bits en partes de la señal que lo necesiten. Además, permite alta calidad en el audio a tasas tan bajas como 64 Kbps.

5.5. La Capa III

Ya que ésta es la capa que corresponde al formato MP3, se intentará dar una mejor descripción de la manera en que se realiza el proceso para generar flujos de audio MP3 válidos. En la siguiente figura se muestra un diagrama en bloques más detallado del codificador psicoacústico que se usa en la Capa III.

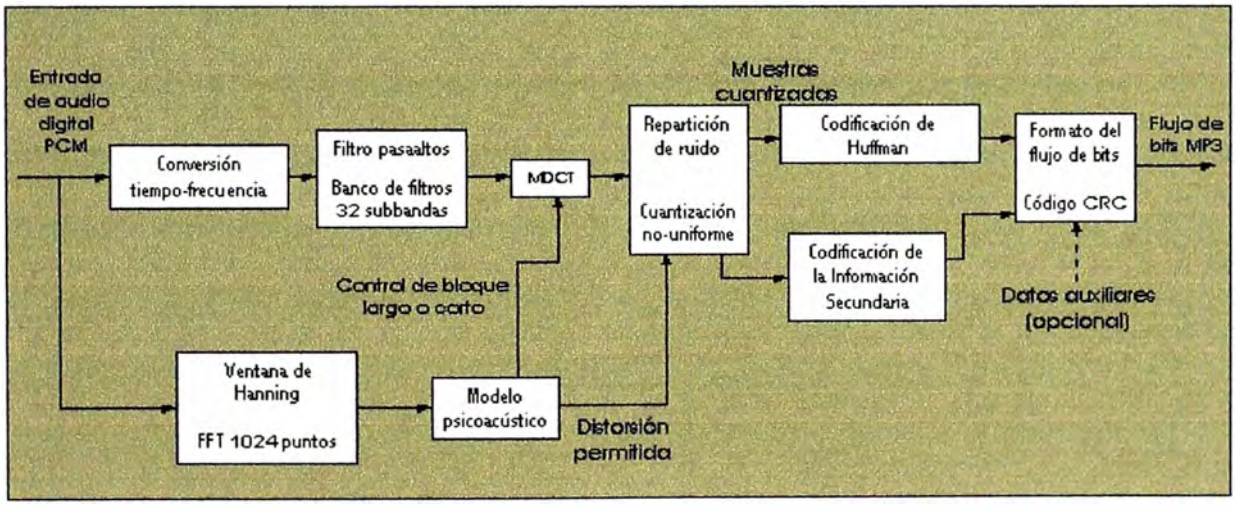


Figura 5.5: Diagrama en bloques más detallado del codificador psicoacústico para la Capa III.

En resumen, el proceso es el siguiente: el flujo de audio a la entrada pasa a través de un banco de filtros que divide la señal en múltiples subbandas. El filtraje se hace paralelamente al análisis psicoacústico que determina el ruido (léase distorsión permitida) en cada subbanda. La etapa "REPARTICIÓN DE RUIDO" usa las distorsiones permitidas para decidir cómo dividir el número

total de bits de código disponibles. Finalmente, la última etapa toma las muestras codificadas con Huffman, realizando el formato para entregar un flujo de bits MP3 válido.

5.6. Análisis psicoacústico

El modelo psicoacústico II que se usa en la Capa III tiene mejoras adicionales que se adaptan mejor a las propiedades del oído humano, en comparación con el modelo empleado en las otras dos capas (modelo I). El análisis psicoacústico tiene dos tareas que cumplir: decidir qué tipo de bloque usar, y calcular la distorsión máxima permitida.

Primero el modelo convierte el audio al dominio espectral, usando una FFT de 1024 puntos para conseguir una buena resolución de frecuencia y poder calcular correctamente los umbrales de enmascaramiento. Antes de la FFT, se aplica una ventana de Hanning convencional para evitar las discontinuidades en los extremos de la señal. La salida de la FFT se usa primero para analizar qué tipo de señal está siendo procesada: una señal estacionaria hace que el modelo escoja bloques largos, y una señal con muchos transitorios da como resultado bloques cortos. El tipo de bloque se usa luego en la parte MDCT del algoritmo. Después de esto, el modelo psicoacústico calcula el mínimo umbral de enmascaramiento para cada subbanda. Estos valores de umbral se usan luego para calcular la distorsión permitida. El modelo pasa entonces las distorsiones permitidas a la sección "REPARTICIÓN DE RUIDO" en el codificador para uso posterior.

El estándar 11172-3 proporciona dos modelos psicoacústicos; el modelo I es menos complejo que el modelo psicoacústico II y simplifica mucho los

cálculos. Ambos modelos trabajan para cualquiera de las capas, aunque requieren adaptaciones específicas para el esquema de la Capa III. Existe considerable libertad en la implementación del modelo psicoacústico; la precisión que se requiera del modelo es dependiente de la aplicación y de la tasa de bits que se quiere lograr. Para bajos niveles de compresión, donde hay un número generoso de bits para realizar la codificación, el modelo psicoacústico puede ser completamente omitido; en cuyo caso, sólo se calcula la relación señal a ruido (SNR) más baja, y con este valor se realiza el proceso de repartición de ruido para la subbanda. A continuación se muestran los pasos generales para el cálculo psicoacústico de la señal.

1) ***Alineación en tiempo.*** Se debe tener en cuenta que cuando se hace la evaluación psicoacústica, los datos de audio que son enviados al modelo deben ser concurrentes con los datos de audio a ser codificados. El modelo psicoacústico debe tener en cuenta el retardo de los datos al pasar por el banco de filtros y aplicar un desplazamiento adicional, de tal manera que los datos relevantes queden centrados en la ventana del análisis psicoacústico. Por ejemplo, usando el modelo I para la Capa I, el retardo a través del banco de filtros es 256 muestras y el desplazamiento necesario para centrar las 384 muestras, dentro de la FFT de 512 puntos, es:

   (512 - 384) / 2 = 64 puntos

   El desplazamiento requerido es, entonces, de 320 puntos para alinear los datos del modelo I con la salida del banco de filtros polifásico.

2) ***Representación espectral.*** El modelo psicoacústico realiza una

conversión del tiempo a la frecuencia totalmente independiente del mapeo realizado por el banco de filtros porque necesita una mejor resolución en frecuencia para calcular con gran precisión los umbrales de enmascaramiento. Ambos modelos usan una transformada de Fourier para realizar el mapeo.

El modelo I usa una FFT de 512 puntos para la Capa I y una FFT de 1024 puntos para Capas II y III. Debido a que el análisis se realiza para 384 muestras en la Capa I, la FFT de 512 puntos proporciona la cobertura adecuada. El análisis psicoacústico para las Capas II y III se realiza sobre 1152 muestras, así que la FFT de 1024 puntos no proporciona cobertura total. Idealmente, la FFT debería cubrir todas las 1152 muestras; aunque 1024 puntos es un compromiso razonable ya que las muestras que se omiten, no tienen mayor impacto en el análisis psicoacústico.

El modelo II usa una FFT de 1024 puntos para todas las capas. En la Capa I, el modelo centra las 384 muestras dentro de la FFT de 1024 puntos. Para las Capas II y III, el modelo ejecuta dos cálculos psicoacústicos de 1024 puntos. El primer cálculo se encarga de las 576 muestras iniciales, y el segundo cálculo se realiza sobre las últimas 576 muestras. El modelo II combina los resultados de ambos cálculos, de tal manera que el resultado total implique la selección del umbral de enmascaramiento de ruido (Noise Masking Threshold) más bajo en cada subbanda. Para simplificar los cálculos, ambos modelos procesan los valores espectrales en unidades perceptuales (el bark, relacionado con el

ancho de las bandas críticas).

3) ***Componentes tonales y no-tonales.*** Ambos modelos identifican y separan las componentes tonales y las componentes de ruido en la señal de audio. Esto se debe a que cada componente presenta un tipo de enmascaramiento diferente.

   El modelo I identifica las componentes tonales basado en los picos locales del espectro de potencias. Después de procesar todas las componentes tonales, el modelo concentra los valores espectrales restantes en una única componente no-tonal por banda crítica. El índice de frecuencia de cada una de estas componentes no-tonales es el valor más cercano a la media geométrica de la banda crítica a la cual pertenece cada componente no-tonal.

   El modelo II realmente nunca separa las componentes tonales ni las no-tonales, sino que calcula un índice de tonalidad en función de la frecuencia, el cual mide el comportamiento que presenta cada tipo de componente. El modelo II usa este índice para interpolar entre valores puros TMN (Tone Masking Noise) y valores puros NMT (Noise Masking Tone). El índice de tonalidad es en realidad una medida anticipada (la cual es llamada "PREDICTABILITY MEASURE", medición de la predecibilidad): el modelo II usa datos de los dos cálculos anteriores para predecir, a través de una extrapolación lineal, los valores de la componente que está siendo procesada. Las componentes tonales son más predecibles y, por lo tanto, tienen índices de tonalidad más altos. Este método de discriminación es mejor que el usado por el modelo I.

4) ***Función de dispersión.*** La capacidad enmascarante de una componente determinada se distribuye por toda la banda crítica que la rodea. Ambos modelos determinan el umbral de enmascaramiento de ruido para ambos tipos de componentes; para lograr esto, el modelo I compara con un enmascaramiento determinado empíricamente, mientras que el modelo II aplica una función de dispersión.

5) ***Umbral de enmascaramiento individual.*** Para poder calcular el umbral de enmascaramiento global (paso 6), el modelo I debe calcular primero los umbrales de enmascaramiento que cada componente tonal o no-tonal genera sobre la señal de audio (llamados "UMBRALES DE ENMASCARAMIENTO INDIVIDUALES"). Debe tenerse en cuenta que antes de esto se realiza un proceso conocido como "DECIMATION OF MASKERS" (disminución en la cantidad de componentes enmascarantes). Este proceso consiste en escoger únicamente las componentes tonales y no-tonales que verdaderamente enmascaran el sonido (cuya magnitud y distancia en barks debe ser apropiada), desechando el resto de componentes computadas en el paso anterior.

   Después de realizada esta escogencia, el modelo I calcula el efecto de enmascaramiento que cada componente enmascarante (tonal o no-tonal) tiene sobre las líneas de frecuencia adyacentes a ella. Este análisis sólo es necesario hacerlo para las líneas de frecuencia que se encuentran entre -3 y +8 barks a partir de la componente enmascarante.

   O sea, el análisis abarca todas las líneas de frecuencia que se encuentren a tres (3) bandas críticas a la izquierda (hacia las bajas

frecuencias), y ocho (8) bandas críticas a la derecha (hacia las altas frecuencias) de la componente enmascarante. Esto se debe a que el efecto de enmascaramiento de la componente tonal o no-tonal que está siendo analizada (por más intensidad que ésta tenga) es demasiado tenue por fuera de este rango.

Como el modelo II nunca separa las componentes no-tonales y tonales, sino que calcula el índice de tonalidad (en función de la frecuencia) que presenta cada componente enmascarante, entonces no es necesario hacer el cálculo de los umbrales de enmascaramiento individuales.

6) ***Umbral de enmascaramiento global.*** Ambos modelos psicoacústicos incluyen un umbral de enmascaramiento absoluto, el cual ha sido determinado empíricamente: el mínimo umbral auditivo en un ambiente silencioso. Se debe recordar que ésta es la intensidad del sonido más débil que se puede escuchar cuando no hay más sonidos presentes.

   Usando el modelo I, este umbral absoluto se combina con los umbrales individuales calculados en el paso anterior para determinar el umbral de enmascaramiento global sobre toda la banda de audio.

   El modelo II no calcula el umbral de enmascaramiento global, sino que trabaja todos los datos dentro de cada subbanda, de acuerdo con el índice de tonalidad que tenga cada componente enmascarante en esa subbanda.

7) ***Umbral de enmascaramiento mínimo.*** Ambos modelos psicoacústicos seleccionan el mínimo umbral de enmascaramiento en cada subbanda. Con el modelo I, para encontrar el umbral de enmascaramiento mínimo

en cada subbanda, simplemente se extrae el mínimo valor del espectro global incluido entre las dos frecuencias límites de cada subbanda, o sea, el valor extraído del umbral global debe ser el valor mínimo de enmascaramiento en la subbanda. Este método se comporta bien para las subbandas más bajas donde la subbanda es estrecha con respecto a las bandas críticas, pero se vuelve inadecuado para las subbandas altas porque una banda crítica en esta frecuencia se distribuye sobre varias subbandas. Esta imprecisión se incrementa todavía más, debido a que el modelo I concentra todas las componentes no-tonales, dentro de cada banda crítica, en un único valor para una sola frecuencia.

El modelo II selecciona el mínimo de todos los umbrales de enmascaramiento en cada subbanda sólo para regiones de frecuencia donde el ancho de la subbanda es amplio comparado con el ancho de la banda crítica. Si el ancho de la subbanda es estrecho en comparación con el ancho de la banda crítica, el modelo realiza un promedio entre todos los umbrales de enmascaramiento en esa subbanda. El modelo II es más preciso para las subbandas altas, ya que éste no concentra las componentes de ruido.

8) ***Relaciones señal a máscara.*** Los dos modelos computan la relación señal a máscara, SMR, como la relación entre la energía de la señal en la subbanda (para la Capa III, un grupo de bandas) y el mínimo umbral de enmascaramiento para esa subbanda. El modelo psicoacústico pasa este valor a la sección "REPARTICIÓN DE RUIDO" (para las Capas I y II, "REPARTICIÓN DE BITS") para uso posterior. En la Capa III, el valor que

se entrega no es la SMR, sino un valor equivalente llamado "DISTORSIÓN PERMITIDA" o "RUIDO PERMITIDO". Este valor determina cuál es la cantidad máxima de ruido de cuantización que se permite en el bloque "REPARTICIÓN DE RUIDO".

5.7. Filtro híbrido

1) ***Filtro pasaaltos.*** El estándar ISO/IEC 11172-3 proporciona respuesta en frecuencia hasta el nivel de DC (corriente directa ó 0 Hz). Sin embargo, para ciertas aplicaciones, se puede incluir un filtro pasaaltos a la entrada del codificador, con su frecuencia de corte ubicada entre 2 y 10 Hz. La aplicación de tal filtro evita el innecesario requerimiento de una alta tasa de bits para la subbanda más baja y aumenta la calidad total en el sonido.

2) ***Banco de filtros polifásicos.*** Bautizado como "FILTRO ANÁLISIS", es uno de los bloques más importantes, usado en todas las capas del codificador de audio MPEG.

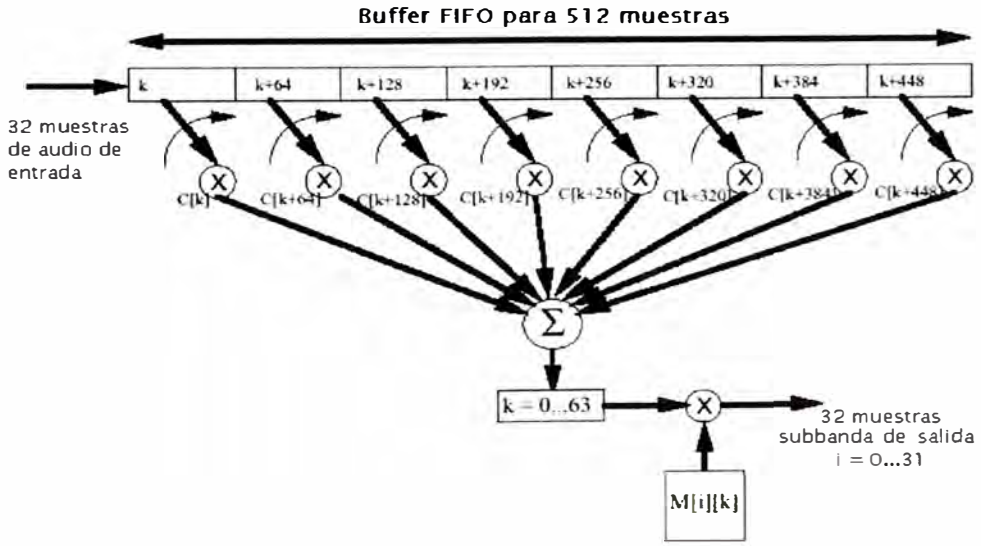


Figura 5.6: Banco de filtros polifásicos también llamado "Filtro Análisis".

Fuente: PAN, Davis. A tutorial on MPEG/Audio compression. En: IEEE Multimedia Journal. Vol. 2 No. 2 (Summer 1995).

Su función es dividir la señal de audio en 32 subbandas; estas subbandas están igualmente espaciadas en frecuencia, y no reflejan exactamente las bandas críticas del oído.

El oído tiene una limitada selectividad en frecuencia que varía en exactitud desde menos de 100 Hz para las frecuencias más bajas hasta un poco más de 4 KHz para las frecuencias más altas. El ancho de banda que proporcionan los filtros es demasiado amplio para las bajas frecuencias, y demasiado estrecho para las altas frecuencias; así que el número de bits del cuantizador no se puede optimizar para la sensibilidad al ruido dentro de cada banda crítica. Entonces, lo mejor es que al espectro audible se le hagan particiones en bandas críticas (por medio de la transformada MDCT) que reflejen la selectividad en frecuencia del oído humano.

El filtro es relativamente simple, pero da una buena resolución en el tiempo con una aceptable resolución en frecuencia. El banco de filtros polifásico presenta pérdidas; inclusive sin cuantización no hay posibilidad de recuperar exactamente la señal de entrada. Afortunadamente, el oído humano no es capaz de percibir el error introducido por el banco de filtros. También existe solapamiento en frecuencia entre bandas adyacentes del filtro; por lo tanto, una señal en una frecuencia particular puede afectar las dos salidas adyacentes en el banco de filtros.

3) ***Transformada discreta del coseno modificada.*** La Capa III procesa las

salidas del banco de filtros con una DCT Modificada de 6 ó 18 puntos y 50% de solapamiento, con el fin de compensar la falta de precisión del banco de filtros, logrando subdividir la salida espectral en frecuencias que proporcionen mejor resolución con respecto a las bandas críticas. Usando 18 puntos, el número máximo de componentes frecuenciales es:

$$23 \times 18 = 576$$

Dando lugar a una resolución frecuencial de:

$$24000/576 = 41.67 \text{ Hz} \ (\text{si } f_s = 48 \text{ Khz})$$

Si se usan 6 líneas de frecuencia la resolución frecuencial es menor, pero la temporal es mayor, y se aplica en aquellas zonas en las que se espera efectos de preeco (transiciones bruscas de silencio a altos niveles energéticos, como por ejemplo justo antes de un sonido percusivo). En estos casos se produce un transitorio con elevados errores de cuantización, debido a la saturación del cuantizador. Al realizar la decodificación, el error se distribuye por toda la trama, ocasionando que las partes de silencio ya no sean silencio, sino que presenten parte de la energía de las otras regiones de la trama. Esto obliga al uso de ventanas MDCT temporales más pequeñas que limitan el efecto de preeco a un número menor de muestras, en comparación con el uso de ventanas grandes; logrando de esta manera, reducir la distorsión. El preenmascaramiento temporal evita que la distorsión restante sea audible. La transformación MDCT no presenta pérdidas comparada con el banco de filtros análisis.

En la figura siguiente se aprecia la configuración del sistema que se está

explicando:

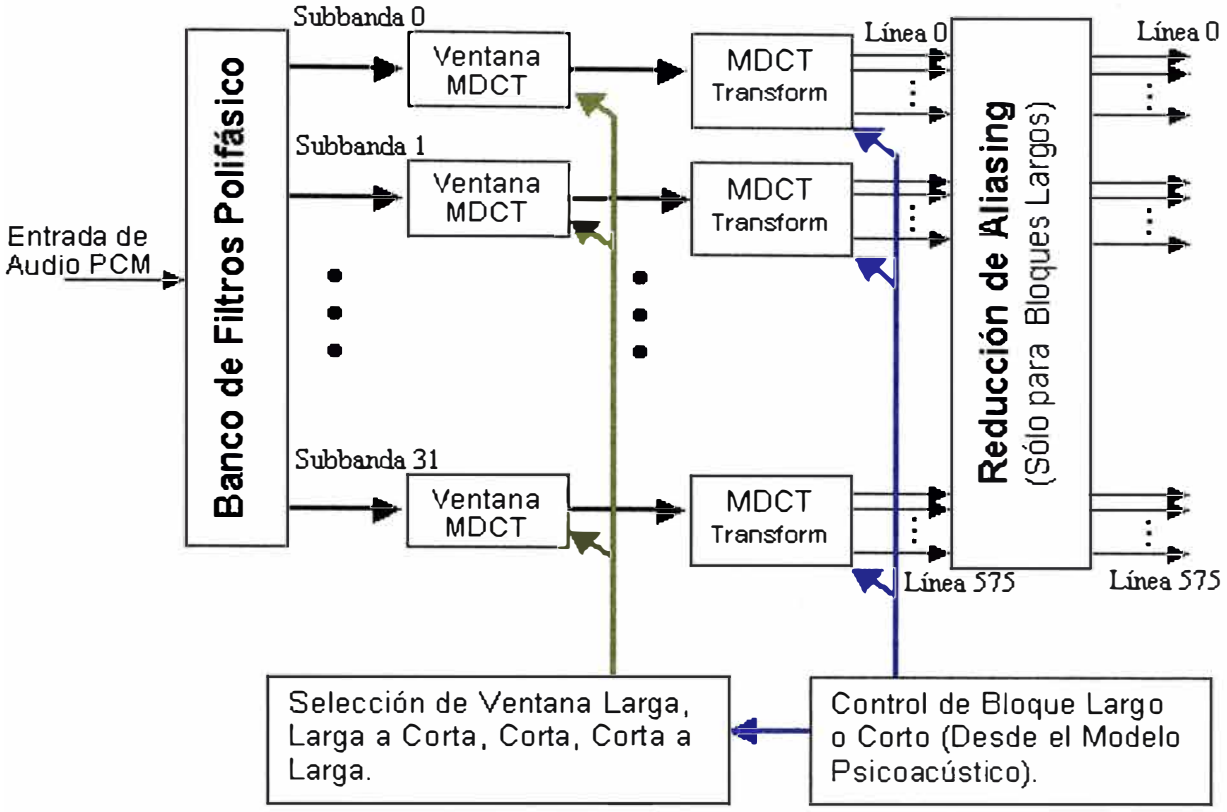


Figura 5.7: Configuración del sistema.

La Capa III tiene tres modos de bloque: dos modos donde las 32 salidas del banco de filtros pueden pasar a través de las ventanas y las transformadas MDCT; todas las salidas con la misma longitud de bloque. Y un modo de bloque mixto donde las dos bandas de frecuencia más baja usan bloques largos y las 30 bandas superiores usan bloques cortos. La decisión del modo de bloque a ser usado recae sobre el modelo psicoacústico: si la señal presenta muchos transitorios se debe usar bloque corto, correspondiente a tres ventanas cortas; pero si la señal es más estacionaria, se debe usar bloque largo, correspondiente a una ventana larga. El cambio entre modos no es instantáneo; un bloque

largo con una ventana de datos especializada (ventana larga a corta o, ventana corta a larga) proporciona el mecanismo de transición entre modos. En la siguiente figura se muestran los cuatro (4) tipos de ventana que se usan durante el proceso MP3: (a) NORMAL, (b) transición de ventana larga a corta (START), (c) 3 ventanas cortas (SHORT), y (d) transición de ventana corta a larga (STOP).

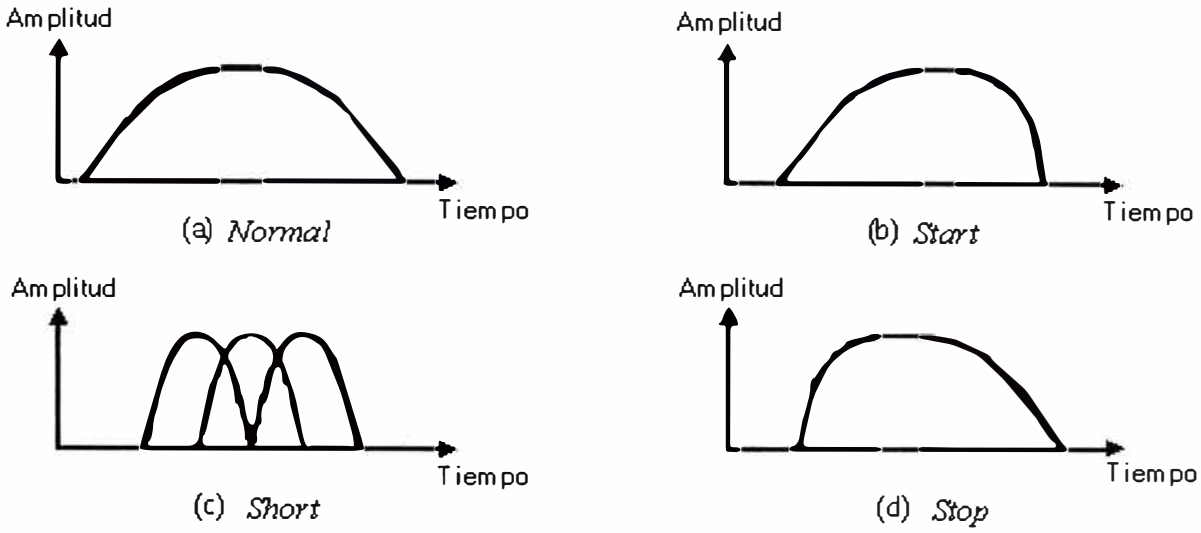


Figura 5.8: Tipos de ventana que se usan durante el proceso MP3.

Si se ejecuta la MDCT sobre cualquiera de las ventanas largas (NORMAL, START, o STOP), se producirán 18 líneas de frecuencia debido al 50% de solapamiento. Cuando se usan las tres ventanas cortas se producirán 3 grupos; cada grupo con 6 líneas de frecuencia que pertenecen a diferentes intervalos de tiempo. El proceso de la transformación MDCT sobre cualquier tipo de bloque producirá, entonces, 576 líneas de frecuencia referidas como "GRÁNULO" (subdivisión de una trama).

En resumen, el proceso que se ha mostrado es: 576 muestras PCM de entrada se convierten en 576 muestras subbanda. El solapamiento, antes de la MDCT, ocasiona que esta cantidad se duplique: en este punto son 1152 muestras subbanda, las cuales finalmente producen 576 coeficientes MDCT (líneas de frecuencia) de salida.

Antes de continuar, se realiza la reducción del aliasing introducido por el filtro análisis. Este proceso se realiza aquí, para lograr reducción en la cantidad de información a ser codificada y transmitida. La reducción se logra a través de cálculos mariposa de 256 puntos como se ve en la siguiente figura:

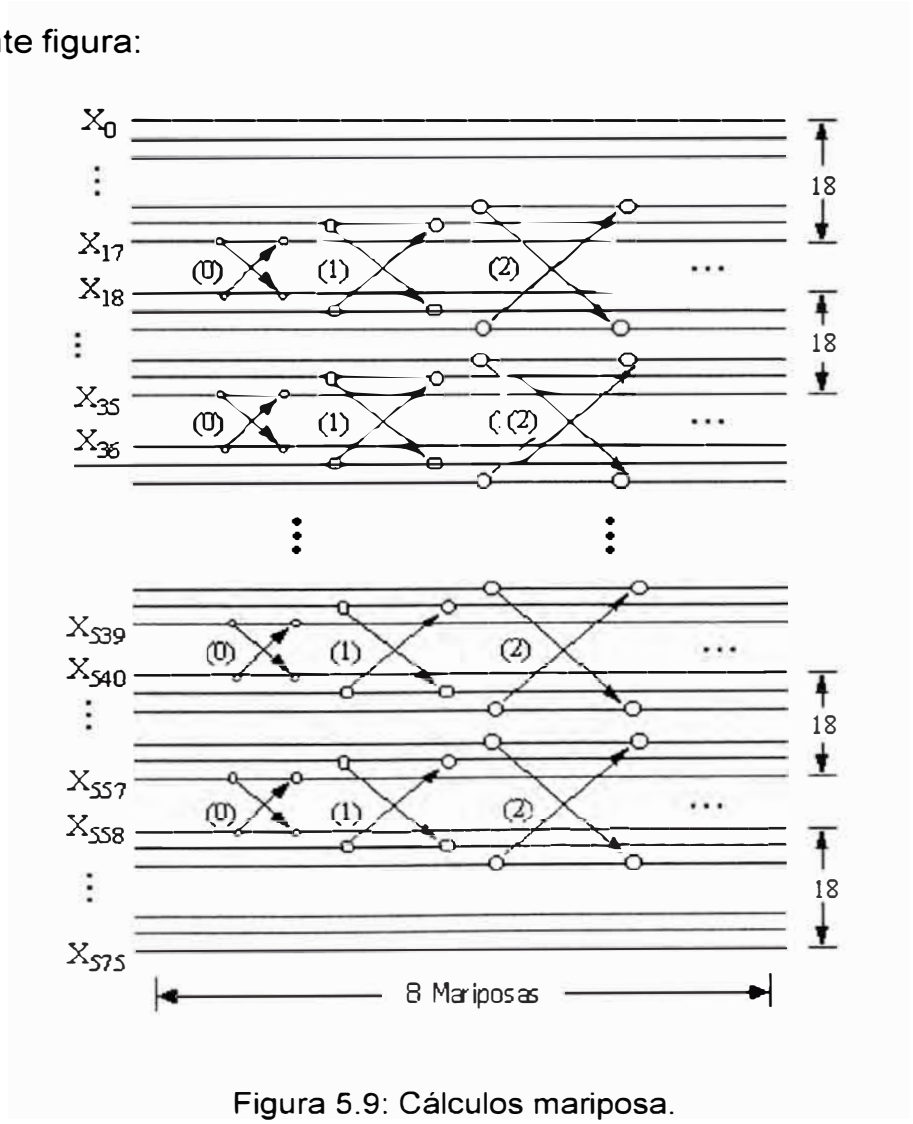


Figura 5.9: Cálculos mariposa.

Cada cálculo mariposa se computa como se muestra en la siguiente figura, donde los valores ca_i y cs_i son dos coeficientes proporcionados por el estándar internacional ISO/IEC 11127-3:

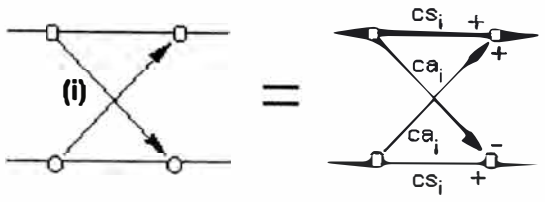


Figura 5.10: Forma de tratar cada cálculo mariposa.

5.8. Repartición de ruido

Esta se hace en la Capa III, mientras las Capas I y II usan repartición de bits. La repartición de bits únicamente aproxima la cantidad de ruido causado por la cuantización, mientras la repartición de ruido verdaderamente calcula el ruido. La repartición se hace en un ciclo de iteración que consiste de un ciclo interno y uno externo.

1) ***Ciclo interno.*** El ciclo interno realiza la cuantización no-uniforme de acuerdo con el sistema de punto flotante verdadero (cada valor espectral MDCT se eleva a la potencia 3/4). El ciclo escoge un determinado intervalo de cuantización (quantization step size), y a los datos cuantizados se les aplica codificación Huffman. Si al realizar este proceso se encuentra que el número de bits requerido para codificar los valores excede la cantidad de bits disponibles, de acuerdo con la tasa de bits escogida, entonces el ciclo comienza otra vez con un nuevo intervalo de cuantización, ejecutando la cuantización y la codificación de Huffman otra vez. El ciclo termina cuando los valores cuantizados que han sido

codificados con Huffman usan menor o igual número de bits que la máxima cantidad de bits permitida.

2) ***Ciclo externo.*** Ahora el ciclo externo se encarga de verificar si el factor de escala para cada subbanda tiene más distorsión de la permitida (ruido en la señal codificada), comparando cada banda del factor de escala (scalefactor band) con los datos previamente calculados en el análisis psicoacústico. Si cualquiera de las bandas del factor de escala tiene más ruido que el máximo permitido, el ciclo amplifica esa banda del factor de escala y ejecuta ambos ciclos (el interno y el externo) de nuevo. El ciclo externo termina cuando una de las siguientes condiciones se cumple:
   - Ninguna de las bandas del factor de escala tiene mucho ruido.
   - La próxima iteración amplificaría una de las bandas más de lo permitido.
   - Todas las bandas han sido amplificadas al menos una vez.

   Ya que el ciclo consume mucho tiempo, una aplicación en tiempo real debe tener en cuenta una cuarta condición, que tenga el ciclo evitando que la codificación se ejecute fuera de tiempo.

3) ***Codificación de Huffman.*** El MP3 también emplea la clásica técnica del algoritmo de Huffman. Actúa al final de la compresión para codificar la información; por lo tanto, no es un algoritmo de compresión, sino más bien un método de codificación.

   Esta técnica crea códigos de longitud variable sobre un número total de bits, donde los símbolos con más alta probabilidad tienen códigos más cortos. Los códigos de Huffman tienen la propiedad de poseer un único

prefijo y por lo tanto, pueden ser decodificados correctamente a pesar de su longitud variable; el proceso de la decodificación es muy rápido, a través de una tabla de correspondencias. Este tipo de codificación permite ahorrar, en promedio, aproximadamente un 20% en espacio de almacenamiento.

Las técnicas que se han mostrado son el complemento ideal para la codificación psicoacústica: durante gran polifonía, muchos sonidos están enmascarados o disminuidos, logrando que la codificación psicoacústica sea muy eficiente; y debido a que hay poca información idéntica, entonces el algoritmo de Huffman presenta poca eficiencia. Pero durante los sonidos "puros" hay muy pocos efectos de enmascaramiento, y es aquí donde la codificación de Huffman se vuelve muy eficiente debido a que los sonidos puros, cuando se digitalizan, contienen gran cantidad de bytes redundantes, que entonces serán reemplazados por códigos más cortos.

5.9. Formato del flujo de bits

El último paso en el proceso de codificación es producir un flujo de bits MP3 válido. Este bloque es el encargado de almacenar el audio codificado y algunos datos adicionales en tramas, donde cada trama contiene información de 1152 muestras de audio. Una trama consiste de encabezado y datos de audio junto con el chequeo de errores y los datos auxiliares, estos dos últimos opcionales. El encabezado describe, entre otros, cuál capa, tasa de bits y frecuencia de muestreo se están usando para el audio codificado. Los datos codificados con Huffman y su información secundaria están

localizados en la parte de los datos de audio, donde la información secundaria dice qué tipo de bloque, tablas de Huffman y factores de ganancia deben ser usados.

Las tramas de audio. El audio en un flujo MPEG-1 está organizado de tal manera que cada fragmento del audio codificado (llamado trama) sea decodificable por sí mismo, con una posible excepción para la Capa III. La trama está constituida por las muestras de audio y por la información secundaria. Esta última sirve de control, además de proporcionar información del archivo. Para la Capa III, la trama está constituida por:

1 trama = 1152 muestras de audio + información de la trama

A la salida del banco de filtros polifásico, las muestras de audio se dividen por subbandas de la manera mostrada en la siguiente figura. Como se ve, cada subbbanda aporta 12 muestras para un total de 384 muestras de audio, en la Capa I; mientras que para la Capa III, cada subbanda aporta 36 muestras de audio para un total de 1152 muestras subbanda por trama.

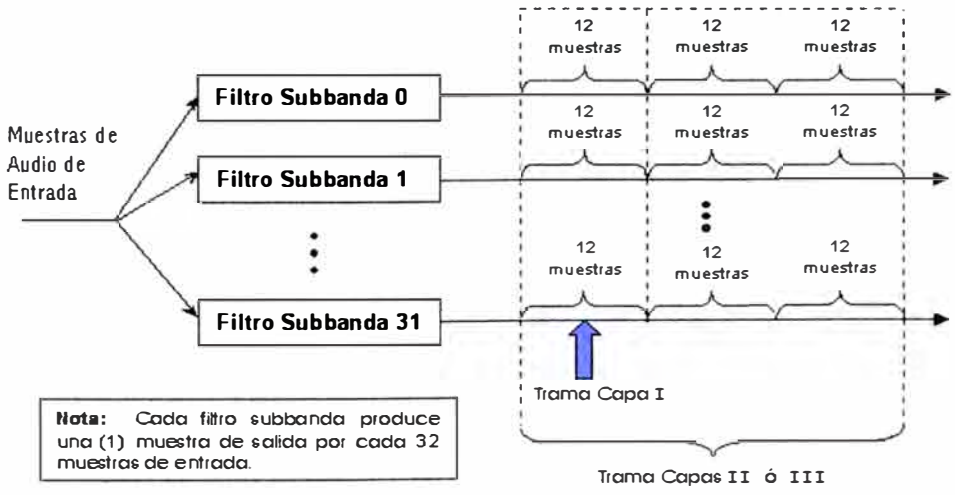


Figura 5.11: División por subbandas de las muestras de audio.

Una trama es un bloque de datos con su propio encabezado e información de audio. En el caso de las Capas I ó II, las tramas son elementos totalmente independientes, así que se puede extraer cualquier fragmento de datos del archivo MPEG y decodificarlo correctamente. Sin embargo, en el caso de la Capa III, las tramas no son totalmente independientes siempre: debido al posible uso del *bit reservoir*, que es una especie de buffer, las tramas son a menudo dependientes unas de otras. En el peor caso, se pueden necesitar hasta nueve tramas antes de poder realizar la decodificación de una sola. La siguiente figura puede servir como idea general.

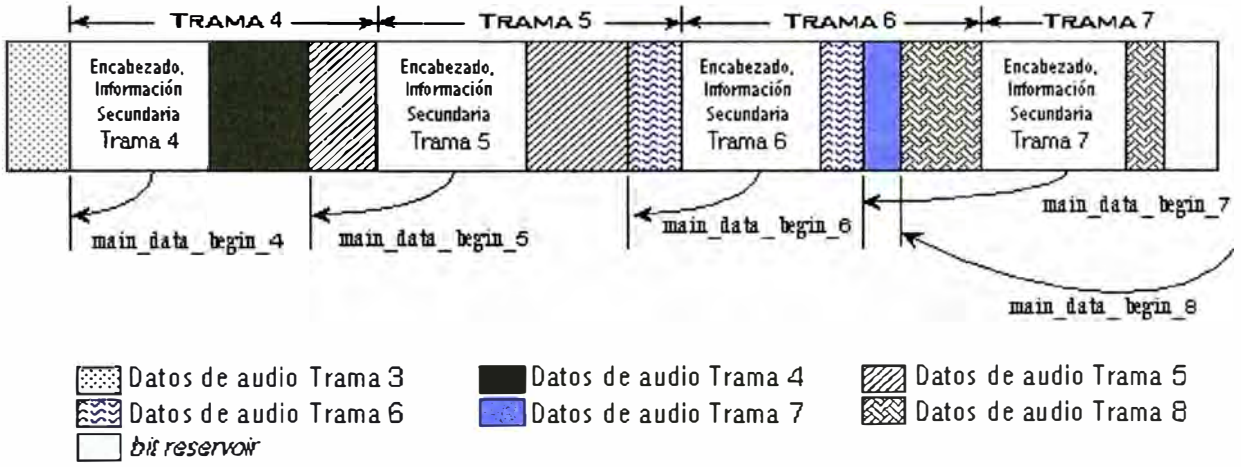


Figura 5.12: Tramas en el caso de la Capa III.

main_data_begin es un puntero de ajuste negativo, incluido dentro de la información secundaria, que indica el inicio de la información de audio dentro de cada trama. Por ejemplo, main_data_begin_4 es igual a cero, indicando que los datos de audio empiezan inmediatamente después de la información secundaria. Para indicar que el audio de la trama 5 se inicia en la trama 4, se especifica main_data_begin_5 como un ajuste negativo que indica el

desplazamiento en bytes hacia la izquierda para encontrar el primer dato de audio de la trama 5.

En el ejemplo se ve como cada trama permite el uso del *bit reservoir*. En el caso de la trama 7, el proceso empieza codificando la información de audio de su propia trama; como los datos requieren muy pocos bits, y la trama 6 tenía espacio disponible, entonces todos los datos de audio de la trama 7 se incluyen en la trama 6, pero la trama 6 sigue con espacio para bit reservoir, que se usa para datos de la trama 8; por lo que gracias al bit reservoir, la trama 6 incluye los datos de audio de tres (3) tramas: las tramas 6, 7 y 8. El audio de la trama 8 se reparte entre las tramas 6 y 7; sin embargo, éste no alcanza a ocupar todo el espacio disponible en la trama 7, así que el bit reservoir de la trama 7 se usa para la trama 9, y así sucesivamente, teniendo en cuenta que los datos de audio de una determinada trama no pueden estar desplazados más de nueve (9) tramas.

Este caso puede ocurrir en una señal de audio MPEG-1 estéreo, si la frecuencia de muestreo es 48 KHz y la tasa de transferencia deseada es 32 Kbps. En este caso, cada trama consume 768 bits, donde 304 bits (32 bits para el encabezado, 16 bits para el chequeo de errores, 256 bits para la información secundaria) son fijos. Por lo tanto, quedan 464 bits disponibles para los datos codificados con Huffman, y debido a que el valor de main_data_begin puede apuntar máximo 511 bytes (4088 bits) hacia atrás, entonces es posible que main_data_begin apunte sobre más de ocho (8) tramas (no se cuenta ninguno de los bits usados para el encabezado y la información secundaria de ninguna trama).

También es importante mencionar que el bit reservoir sólo puede originarse de tramas que ya han sido codificadas; para este buffer no es posible usar tramas para las que todavía no se haya hecho la repartición de los bits disponibles (repartición de ruido).

El formato que tiene cada trama se muestra en la siguiente figura, en la cual se puede ver el encabezado de trama que posee 32 bits (cuatro bytes) de longitud; los primeros 12 bits siempre se ponen en '1', se llaman "*FRAME SYNC*", y se usan para sincronización de la trama.

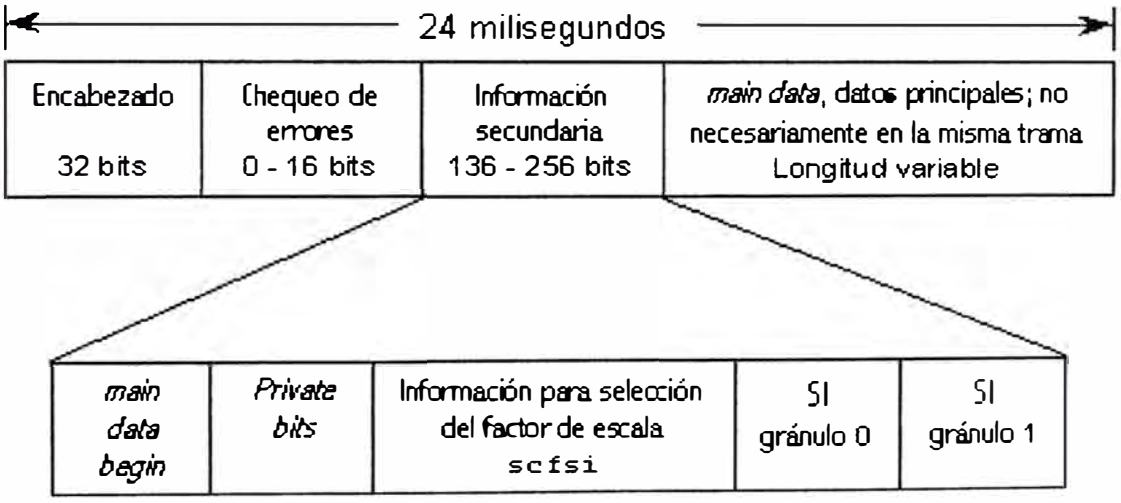


Figura 5.13: Formato de cada trama.

Las tramas pueden tener opcionalmente un CRC para chequeo de errores. Su longitud es de 16 bits, y si existe, se pone después del encabezado. Volviendo a calcular el CRC se puede comprobar si la trama ha sido alterada durante la transmisión del flujo de bits de audio MP3. A continuación sigue la información secundaria (*Side Information*) que indica cómo se realizó la codificación, y por lo tanto, cómo debe realizarse la decodificación. En el último bloque sí vienen los datos de audio (main data), repartidos entre dos

(2) gránulos.

Dentro de la información secundaria, que usa 136 bits en modo monofónico y 256 bits en los otros modos, se incluye el main_data_begin, que es el puntero ya visto. Los bits privados están a disposición del usuario. Después viene la información que indica cuál combinación de factores de escala se está usando (scfsi, *scalefactor selection information).* Los últimos dos subbloques corresponden a la información secundaria (*Side Info*, SI) para cada cada uno de los dos gránulos (subtramas) en los que se divide una trama.

El último bloque, *main data*, es el que lleva la información de audio; las muestras MDCT codificadas con Huffman, repartidas entre dos gránulos. Cada gránulo contiene información de 576 muestras de audio (exactamente la mitad de la información total de la trama). Además, en este mismo bloque se incluyen los factores de escala de la trama y la información auxiliar, siendo esta última opcional.

5.10. El encabezado de las tramas

No existe encabezado principal de archivo en el formato de audio MPEG. En éste el encabezado es individual para cada trama (fragmento de archivo).

Cuando se quiere leer información de un archivo MP3, usualmente es suficiente encontrar la primera trama, leer su encabezado y asumir que las otras tramas son iguales. Pero éste no es siempre el caso; por ejemplo, existen algunos archivos con tasas de bits variables, donde cada trama posee su propia tasa de bits. Esto se hace con el fin de mantener constante la calidad del sonido durante todo el archivo. Otro método usado para

mantener constante la calidad de sonido es emplear más bits (con ayuda del buffer bit reservoir) en las partes donde se necesite.

El encabezado de la trama tiene la siguiente presentación, con las posiciones para cada uno de los 32 bits:

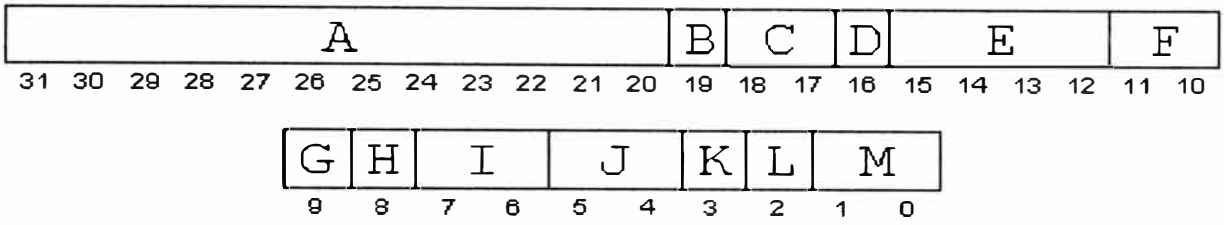


Figura 5.14: Presentación del encabezado de la trama.

Campo	Comentario	N° de bits
syncword	tren 1111 1111 1111 (FFF hex)	12
ID	siempre a "1" para MPEG-1 de audio	1
layer	11=I, 10=II, 01=III, 00 reservado (capa)	2
protection_bit	0 si se añade redundancia, 1 sino	1
bitrate_index	15 valores (0000=flujo libre, 1111=prohibido)	4
sampling_frequency	00=44.1 KHz, 01=48, 10=32, 11=reservado	2
padding_bit	1=ajuste (necesario para Fmuestreo=44.1 KHz)	1
private_bit	no especificado, uso libre	1
mode	00=stereo, 01=joint, 10=dual, 11=mono	2
mode_extension	margen de las sub-bandas en intensity_stereo	2
copyright	1=copyright, 0=libre	1
original/copy	1=original, 0=copia	1
emphasis	00=no, 01=50/15 μs, 10=reservado, 11=J.17	2

Tabla 5.6: Campos de la cabecera de una trama MPEG-1 de audio.

A: ***syncword.*** Con 12 bits de longitud, todos en '1' para identificar el comienzo de la trama.

B: ***ID.*** Un (1) bit usado para identificación del audio. Siempre en '1', para indicar que se trata de audio MPEG-1.

C: ***Layer.*** Dos (2) bits usados para descripción de la capa. Para identificar cuál esquema (léase capa) fue usado durante la codificación del audio.

00	Reservado
01	Capa III
10	Capa II
11	Capa I

Tabla 5.7: Descripción del esquema usado durante la codificación.

D: ***protection_bit.*** Un (1) bit de protección. Si está en '0' indica que la trama está protegida por un código de redundancia cíclica para detección de errores.

E: ***bitrate_index.*** Cuatro (4) bits para proporcionar el índice de la tasa de bits, de acuerdo con la siguiente tabla:

Código	Tasa de bits MPEG-1 (Kbps)		
	Capa I	Capa II	Capa III
0000	Formato libre	Formato libre	Formato libre
0001	32	32	32
0010	64	48	40
0011	96	56	48
0100	128	64	56
0101	160	80	64
0110	192	96	80
0111	224	112	96
1000	256	128	112
1001	288	160	128
1010	320	192	160
1011	352	224	192
1100	384	256	224
1101	416	320	256
1110	448	384	320
1111	No permitido	No permitido	No permitido
Nota: Si la trama usa formato libre (una tasa de bits diferente a las listadas), la tasa debe permanecer constante, y debe ser menor a la máxima tasa de bits permitida (320 Kbps para la Capa III).			

Tabla 5.8: Indice de la tasa de bits.

F: ***sampling_frequency.*** Dos (2) bits que indican la tasa de muestreo.

00	44.1 KHz
01	48 KHz
10	32 KHz
11	Reservado

Tabla 5.9: Tasas de muestreo.

G: *padding_bit.* Un (1) bit usado para relleno. Si está en '1' la trama se rellena con una ranura extra. Únicamente se usa para frecuencias de 44.1 KHz. Por ejemplo, un sonido de 128 Kbps y 44.1 KHz Capa II usa muchas tramas de 418 bytes de largo y unas pocas de 417 bytes para cumplir exactamente la tasa de transferencia de 128 Kbps. La ranura consume 8 bits (1 byte) para las Capas II y III.

H: *private_bit.* Un (1) bit para uso privado. No se usa generalmente.

I: *mode.* Dos (2) bits que indican el modo de canal, tal y como se muestra a continuación:

00	*Stereo*
01	*Joint Stereo*
10	*Dual Channel* (2 canales monofónicos independientes)
11	*Single Channel* (1 canal monofónico)

Tabla 5.10: Indicación del modo de canal.

En el modo *Stereo* indica que el canal comparte bits, pero no usa codificación *Joint Stereo*. En el modo *Joint Stereo* sí se saca provecho de la correlación existente entre los dos canales para representar más eficientemente la señal. El modo *Dual Channel* está conformado por dos canales mono totalmente independientes (cada uno es un archivo de audio diferente); cada canal usa exactamente media tasa de bits del archivo. La

mayoría de los decodificadores los procesan como estéreo, pero no es siempre el caso. *Single Channel* consiste en un único canal de audio.

J: *mode_extension.* Dos (2) bits indicando extensión al modo; sólo se usa en modo *Joint Stereo*. La extensión al modo se usa para información que no es de ninguna utilidad en el efecto estéreo. Estos bits se determinan dinámicamente por un codificador en el modo *Joint Stereo*, y este modo puede cambiar entre tramas, o incluso se puede dejar de usar en algunas tramas. En la Capa III, estos dos bits indican qué tipo de codificación *Joint Stereo* se está usando, Intensidad estéreo o Estéreo M/S. Estéreo M/S se refiere a transmitir los canales normalizados *Middle/Side* (Suma/Diferencia) de los canales izquierdo y derecho en lugar de los habituales Izquierdo/Derecho. En el lado del codificador los canales habituales se reemplazan usando la fórmula:

$$M_i = \frac{\sqrt{2}}{2}(L_i + R_i) \quad y \quad S_i = \frac{\sqrt{2}}{2}(L_i - R_i)$$

M_i = Middle; S_i = Side; L_i = Izquierdo; R_i = Derecho

Los valores M_i se transmiten por el canal izquierdo y los valores S_i se transmiten por el canal derecho.

En el lado del decodificador los canales izquierdo y derecho se reconstruyen así:

$$L_i = \frac{M_i + S_i}{\sqrt{2}} \quad y \quad R_i = \frac{M_i - S_i}{\sqrt{2}}$$

Intensidad estéreo se refiere a retener en las frecuencias superiores a 2 Khz sólo la envolvente de los canales izquierdo y derecho.

El código indica que tipo de extensión al modo se está usando de la siguiente manera:

Código para la Capa III		
Código	*Intensity stereo*	*M/S stereo*
00	no	no
01	sí	no
10	no	sí
11	sí	sí

Tabla 5.11: Código indicativo del tipo de extensión al modo.

K: *copyright.* Un (1) bit usado para *copyright*. Tiene el mismo significado que el bit de *copyright* en CD y cintas DAT, indicar que es ilegal copiar el contenido del archivo si el bit está en '1'.

L: *original/copy.* Un (1) bit usado para indicar si se trata de un medio original, si el bit está puesto en '1'. En '0' indica que es una copia del medio original.

M: *emphasis.* Dos (2) bits usados para información del énfasis. Le indica al decodificador que el sonido debe ser "re-ecualizado" después de una supresión de ruido tipo *Dolby*. Se usa raramente.

00	Ninguna
01	50/15 µs
10	Reservado
11	CCITT J.17

Tabla 5.12: Bits usados para información del énfasis.

5.10.1. Chequeo de errores

Si el bit de protección en el encabezado es igual a '0', se incluye un CRC de 16 bits después del encabezado. Si el bit de protección está en '1', no hay chequeo de errores y estos bits pueden ser usados para los datos de audio. El método para detección de errores que se utiliza es CRC-16, cuyo polinomio generador es:

$$CRC - 16 = x^{16} + x^{15} + x^2 + 1$$

5.11. Información secundaria

Ésta consta de 17 bytes para el modo monofónico, y de 32 bytes en cualquier otro modo. La información que contiene, consiste de cuatro partes: el puntero main_data_begin, información secundaria para ambos gránulos (scfsi y private_bits), información secundaria para el gránulo 0, e información secundaria del gránulo 1.

main_data_begin (9)	private_bits (5,3)	scfsi (4,8)	SI gránulo 0 (59,118)	SI gránulo 1 (59,118)

Figura 5.15: Distribución de la información secundaria.

- **main_data_begin:** El campo main_data no está necesariamente localizado justo después de la información secundaria. main_data_begin es un puntero que usa 9 bits, indicando la localización donde está el primer byte del main_data de la trama actual. La localización está especificada como un desplazamiento negativo en bytes desde el encabezado actual (bytes a la izquierda, antes del primer bit del

encabezado).

La información secundaria (SI) común a ambos gránulos se muestra a continuación:

- **private_bits:** El número de private_bits para la información secundaria depende del número de canales (5 para mono y 3 para estéreo). El número de bits reservados para private_bits es definido por el usuario.
- **scfsi:** La variable scfsi (información para selección del factor de escala) determina si los factores de escala se envían para cada gránulo, o si son comunes para ambos gránulos, por canal. Se transmiten cuatro (4) bits por canal, cada bit perteneciente a un grupo de bandas del factor de escala diferente. Un '0' para un grupo específico de bandas del factor de escala, indica que los factores de escala para ese grupo en particular, se transmiten para cada gránulo. Un '1' indica que se usan los mismos factores de escala para ambos grupos; por lo tanto, sólo se transmiten los factores de escala correspondientes al grupo de bandas del primer gránulo.

Después de la información secundaria para ambos gránulos, sigue la información secundaria para cada gránulo:

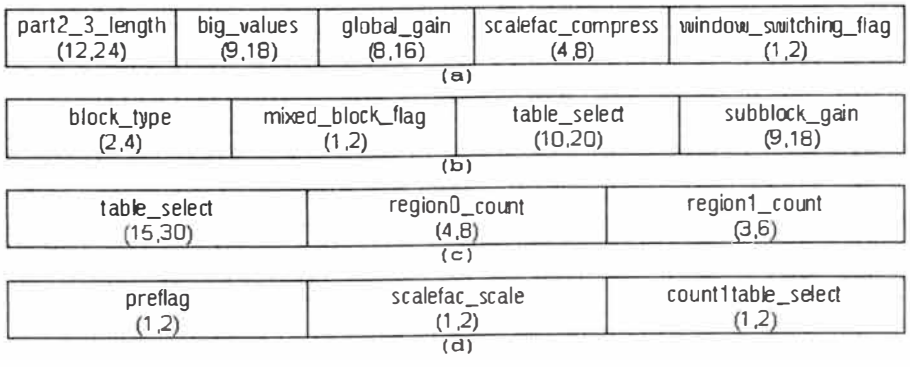


Figura 5.16: Información secundaria para cada gránulo.

En el caso de bloques largos, la información secundaria para cada gránulo es:

- **part2_3_length:** Denota el número de bits que son usados en main_data para los factores de escala y los datos codificados con Huffman. Se usan 12 bits en modo mono y 24 en los otros modos. Como la cantidad de bits usados para la información secundaria es constante, part2_3_length puede usarse para calcular el comienza del próximo gránulo.
- **big_values:** Después de la cuantización, las 576 muestras MDCT cuantizadas están organizadas en un orden determinado (de menor a mayor frecuencia). Luego, estos valores se dividen en tres particiones consecutivas: rzero, count1 y big_values. La primera partición, rzero, se localiza en las altas frecuencias y consiste en pares de ceros. La partición de la mitad, count1, consiste de cuádruplos cuyo valor es -1, 0 ó 1. La última partición, big_values, se localiza en las bajas frecuencias extendiéndose hasta el nivel de directa (frecuencia de 0 Hz) y se compone de pares de valores restringidos a una amplitud máxima absoluta de 8206 (8191+15, el cual es el máximo valor cuantizado permitido). El campo big_values indica la cantidad de pares cuantizados que pertenecen a esta partición. Nueve (9) bits se usan para big_values en modo mono y 18 en los otros modos.
- **global_gain:** Contiene información acerca del intervalo usado en el cuantizador, donde la cuantización se hace logarítmicamente. La variable global_gain usa 8 bits en modo mono y 16 bits para los otros modos.
- **scalefact_compress:** Es una variable de 4 bits (en modo mono),

transmitida para cada gránulo, la cual determina el número de bits usados para la transmisión de los factores de escala. Cada gránulo se divide en 12 ó 21 bandas del factor de escala dependiendo del tipo de ventana que se esté usando. Estas bandas del factor de escala se dividen de nuevo en dos grupos (0-10 y 11-20 para ventanas largas; 0-5 y 6-11 en el caso de ventanas cortas). La variable scalefac_compress se usa como índice a una tabla proporcionada en el estándar ISO 11172-3, la cual retorna dos variables llamadas "slen1" y "slen2", que indican la cantidad de bits usados para los factores de escala del primer y segundo grupo de bandas, respectivamente.

- **window_switching_flag:** Un (1) bit por canal que señala si una ventana diferente del tipo *NORMAL* se está usando. Este valor determina los siguientes 22 bits en la información secundaria: si está en '1' se añaden los bits de la figura 5.16(b); si está en '0', se añaden los bits de la figura 5.16 (c).
- **table_select:** Habilita el uso de 32 diferentes tablas para el código de Huffman, dependiendo de las estadísticas de la señal. Se usan 15 bits por canal (5 bits por región) para indicar cuáles de las 32 tablas han sido seleccionadas.
- **region0_count:** Para mejorar el desempeño en la codificación, la partición big_values se subdivide en tres regiones llamadas region0, region1 y region2. Cada región se codifica con una de las 32 tablas de Huffman (seleccionada con table_select). La variable region0_count especifica el límite entre region0 y region1. Esta variable de 4 bits (en

modo mono) especifica la cantidad de bandas del factor de escala incluidas en esta región, pero disminuidas en 1.

region0_count = bandas del factor de escala en region0 - 1

- **region1_count:** Especifica el límite entre region1 y region2. Esta variable de 3 bits por canal indica las bandas del factor de escala incluidas en region1, disminuidas en 1.

region1_count = bandas del factor de escala en region1 - 1

- **preflag:** Un (1) bit por canal, indicando que se usó preénfasis (o sea, amplificación adicional en las altas frecuencias). Este valor apunta a una tabla en el estándar ISO 11172-3, cuyos 21 valores son sumados a los factores de escala. Para bloques cortos, no se usa preénfasis.
- **scalefac_scale:** Los factores de escala están cuantizados de manera logarítmica con un intervalo de 2 ó $2^{\frac{1}{2}}$, dependiendo del valor de scalefact_scale, que usa 1 bit por canal.
- **count1table_select:** Esta variable, que usa 1 bit por canal, indica cuál de dos (2) posibles tablas de Huffman fue usada para codificar la partición count1.

En el caso de bloques cortos, la información secundaria sólo cambia en las variables mostradas en la figura 5.16(c), las cuales son reemplazadas por aquellas de la figura 5.16(b). Las otras variables mostradas en la figura 5.16 no cambian.

- **block_type:** Indica el tipo de ventana que se usa en un gránulo particular. La variable block_type consume 2 bits por canal.
- **mixed_block_flag:** Esta variable, que consume 1 bit por canal, indica

que se usan diferentes tipos de ventana en las bajas y en las altas frecuencias. Si esta variable está en '1', las dos subbandas más bajas usan ventana *NORMAL*, y las 30 subbandas restantes usan el tipo de ventana especificado por block_type.

- **table_select:** En este caso, table_select usa 10 bits por canal, debido a que, para bloques cortos, la partición big_values sólo se subdivide en dos (2) regiones.
- **subblock_gain:** Habilita una ganancia por un factor de 4 para un subbloque particular. Esta variable usa 3 bits por canal.

5.12. Datos principales

En esta parte del flujo de bits de la Capa III, están incluidos los campos mostrados en la siguiente figura:

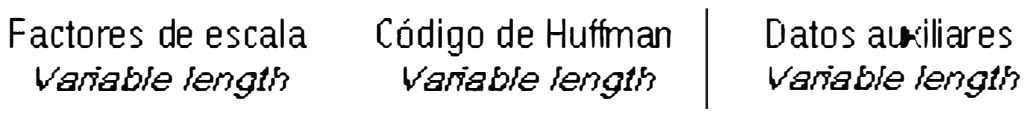


Figura 5.17: Campos incluídos en los datos principales.

- **Factores de escala:** Éstos se usan para colorear el ruido de cuantización. Los factores de escala se transmiten para cada grupo de líneas de frecuencia (bandas del factor de escala) de cada gránulo, dependiendo del valor de scfsi para ese grupo particular de líneas de frecuencia. La cantidad de factores de escala realmente transmitidos, también depende de block_type, window_switching_flag y mixed_block_type. Los factores de escala consumen entre 0 y 74 bits.
- **Código de Huffman:** Las líneas de frecuencia de cada gránulo se

dividen en tres particiones (rzero, count1 y big_values). La partición rzero no se codifica, ya que sólo contiene valores iguales a cero (0). La partición count1 contiene cuádruplos de valores iguales a -1, 0 ó 1, que se codifican usando una de 2 posibles tablas de Huffman, la cual ha sido especificada por count1table_select. Para cada valor diferente de cero, se agrega un bit que indica el signo ('0' si es positivo). La partición big_values fue subdividida en tres regiones, las cuales se codifican separadamente, usando una de 32 posibles tablas de Huffman (numeradas de 0 a 31, pero en realidad son 30, ya que las tablas 4 y 14 no existen), o sea, una tabla por región. Dentro de la partición big_values, los pares de líneas de frecuencia con valor absoluto menor que 15, se codifican directamente. Para cada valor absoluto mayor o igual a 15, se agregan 1 ó 2 campos extras llamados "linbitsx" o "linbitsy" dependiendo de cuál es el valor del par (x,y) que es mayor o igual a 15. Este campo extra usa de 0 a 13 bits, dependiendo del parámetro "linbits", el cual se calcula con base en el valor máximo de la región, como se muestra en la siguiente fórmula:

$$\text{linbits} = \log_2(\text{máximo valor cuantizado} - 14) \Rightarrow \text{Se redondea por exceso}$$

De nuevo, para cada valor diferente de cero, se agrega bit de signo ('0' si es positivo). Por ejemplo: asúmase, primero que la tabla de Huffman ya ha sido seleccionada, y también:

Par de valores cuantizados (x,y) = (0,15)
Máximo valor cuantizado de la región = 1039
Código de Huffman para el par (0,15) = '01101'
Valor adicional para 'y' = linbitsy = 15-15 = 0

$$linbits = \log_2(1039 - 14) \cong 10{,}0014 \Rightarrow linbits = 11$$

$$linbitsy = 15 - 15 = 0 = '00000000000'$$

$$Codificación\ del\ par\ (0,15) = Codificación\ del\ par\ (0,15) + linbitsy$$

$$Codificación\ del\ par\ (0,15) = '01101'\ '00000000000'$$

$$bits\ necesarios\ para\ codificar\ el\ par\ (0,15) = 16\ bits$$

(x,y) 5 bits	linbitsx 0 bits	linbitsy 11 bits	signx 0 bits	signy 1 bit
Flujo de bits '01101' '00000000000' '0'				

Figura 5.18: Obtención del flujo de bits.

En el caso de que 'x' también sea mayor que 14, se debe buscar el código de Huffman para el par (15,15), y además también se debe codificar un valor adicional llamado "linbitsx", que indica la diferencia entre 15 (máximo valor de las tablas) y el valor verdadero de 'x'. Adicionalmente, por cada valor diferente de cero se debe agregar un bit de signo ('0' si es positivo, '1' si es negativo). En el ejemplo, la cantidad total de bits que se necesita para codificar el par es 17 bits, ya que se debe agregar un bit para indicar que 'y' es diferente de cero (0).

- **Datos auxiliares:** Éstos son opcionales, y la cantidad de bits repartidos para este campo, se define por el usuario.

5.13. Ventajas y desventajas del Mp3

Ventajas:

- *Excelente compresión:* Pueden bajarse diferentes canciones de la Web sin ningún tipo de inconvenientes.
- *Amplia variedad de software:* La tecnología ha estado disponible durante

varios años, y de esta manera el software se desarrolló cada vez más.

- *Amplia variedad de canciones:* Encontrar este tipo de archivos en la Web no es nada difícil, lo único que se debe hacer es buscar los temas favoritos y bajarlos al disco duro.
- *No exige procesadores de muy alta calidad:* Para reproducir este tipo de archivos no se necesita una supercomputadora, ya que el mínimo requerimiento es un procesador Pentium 100.

Desventajas:

- *Dificultades con los derechos de autor:* la piratería con este tipo de archivos es muy frecuente, y a pesar que muchos de los codificadores están disponibles gratis para bajar en la Web, algunos aún siguen discutiendo acerca de los derechos sobre los algoritmos de dichas aplicaciones.
- *Tecnologías competidoras:* Con la creación del AAC y el VQF, el MP3 ya no es el formato de audio más eficiente, ya que estos dos pueden crear archivos más pequeños con la misma calidad de sonido.

Los efectos audibles producidos por una mala codificación o decodificación son muy diferentes de los típicos efectos de ruido de fondo o distorsión armónica que se producen al procesar audio analógicamente. Defectos típicos de la codificación perceptiva pueden manifestarse como:

- Distorsión no armónica.
- Ruido restringido a determinadas bandas frecuenciales (a veces en forma de preecos).
- Una especie de ligeros ronquidos o asperezas.

- Voz doble: Parece que una voz se haya grabado superpuesta a sí misma. Este efecto se produce cuando se utilizan bajas frecuencias de muestreo y mucha compresión (el AAC corrige este defecto empleando la técnica *Temporal Noise Shaping*).

Algunas limitaciones del audio analógico quedan definitivamente superadas por el MP3, por lo que no es necesario buscar sistemas para medir su calidad. Este es el caso del margen dinámico (o simplemente dinámica) y de la respuesta en frecuencia. Si el sistema está bien diseñado, la dinámica supera las capacidades de un conversor A/D de 24 bits (más de 120 dB) y la respuesta en frecuencia es completamente plana (o dB de desviación para cualquier frecuencia).

Ante la imposibilidad de aplicar los mismos criterios de evaluación de la calidad sonora que en el mundo analógico, se han realizado algunos sistemas nuevos de medición como el PEAQ (*Perceptual Evaluation of Audio Quality*). Sin embargo, el método más fiable sigue siendo el Test de Escucha. En audio, un oído educado es siempre el test definitivo.

5.14. Aspectos legales del MP3

Sobre los aspectos legales del MP3 se ha escrito mucho. La ley prohibe comerciar con música que no haya sido escrita por nosotros, pero descargar un tema de la Web en formato zip no está cubierto por ninguna ley, aunque la cuestión es: ¿qué pasa si yo comienzo a crear CD's desde MP3? O sea, el proceso inverso. Primero copiaría los archivos a mi disco duro, luego los decodifico para después quemar un compacto con música en formato CD-Audio, que puede ser reproducida en cualquier dispositivo.

Todo este asunto ha dejado la plataforma de investigación científica, para pasar a formar parte de un divertimento casero. El software se consigue en Internet, así como la música y con una lectora y una grabadora de CD montadas en una PC, la copia de música ha vuelto a hacerse tan común como en los tiempos de la doble cassettera.

El MP3 no es ilegal. No es posible sentar a un algoritmo para comprimir archivos en el banquillo. Se puede comprimir todas las canciones de su discoteca (incluidos LP's y cassettes) y almacenarlas digitalmente, en su disco duro o creando sus propios CD's. Otra cosa es que luego se nos ocurra vender esos CD's, colocar las canciones en su Web o distribuirlas por correo. Tendrá que pagar los derechos de autor. Es también legal si se compran los CD's del artista y hacemos una copia en MP3 de respaldo para uso estrictamente personal.

Estos problemas de copias ilegales han llamado la atención de las compañías grabadoras y editoras de música, ya que la distribución de MP3 crece a ritmos agigantados y amenaza en convertirse en un nuevo estándar de distribución masiva. Como ejemplo está la demanda entablada por la Asociación de Sellos Fonográficos estadounidenses a la compañía Diamond Multimedia por su reproductor MP3 portátil Río PMP3000, el cual reproduce archivos que se guardan en una tarjeta de memoria flash y se pueden reproducir al estilo Walkman. Diamond salió libre de culpas ya que se alegó que dicho dispositivo reproduce archivos de computadora, los que eventualmente se decodifican en sonido audible, pero también se pueden transportar de esta manera archivos zip, de imágenes, etc.

¿Son legales las webs que tienen archivos MP3?

- Algunas sí y otras no. Me explico:
- Son **ilegales** las que ofrecen archivos MP3 de temas publicados (en disco, LP, cassette o el medio que sea), sin pagar los derechos correspondientes a la institución equivalente del país que se trate, puesto que están vulnerando los derechos de autor, que están fuertemente protegidos en la legislación. Hay montones de éstas webs, como el sitio "pirata" http://mp3.box.sk (en realidad un buscador de mp3 en otras webs), y otras muchas. Entre ellas cabe distinguir la presencia de una empresa totalmente seria como el buscador Lycos, que ha tenido problemas legales al crear su buscador de MP3: http://mp3.lycos.com, ya que en su motor de búsqueda, junto a mp3 con licencia, contenía también otros sin licencia.
- Son **"bastante legales"** las que ofrecen esos mismos archivos con licencia de la Sociedad General de Autores o la de los países en donde uno se encuentre, a la cual pagan los derechos correspondientes. Las empresas que gestionan estas webs cobran a sus visitantes unas cuotas por descargar a su ordenador las canciones que desee. Pero parece ser que no son del todo legales, porque también deberían pagar derechos a la AIE (Asociación de Intérpretes y Ejecutantes), y quizá a algunas otras entidades similares. Web de este tipo en España es: www.weblisten.com.
- **Casi Totalmente legales** son aquellas webs que ofrecen sólo música que ha sido ofrecida por sus propios autores con el objeto de promocionarse, o incluso venderla. Las más conocidas son

www.mp3.com en USA y www.vitaminic.com en España y otros países de Europa, pero existen otras muchas. Estas webs son altamente recomendables, especialmente mp3.com y las que mantienen su misma filosofía.

- www.mp3.com ofrece un montón de archivos completos que puede descargar legalmente, catalogados en estilos musicales, y junto a información detallada de sus autores e intérpretes. Si un grupo le gusta, tiene la opción de comprar sus CDs en la misma web, generalmente a un precio inferior a 1.500 pts (cambio aproximado).
- www.vitaminic.com tiene un planteamiento algo diferente: en vez de poner muchas canciones a libre disposición, sólo tiene algunas, y la mayoría deben ser compradas directamente antes de poder descargarlas. Por otro lado, ofrece a los autores el 50% de lo que paguen los visitantes por sus canciones, lo que puede ser una buena fuente de ingresos si su canción resulta ser un "best-seller". Los precios son variables alrededor de 150 pts por canción.

CAPÍTULO VI
LOS FORMATOS QUE PRETENDEN SUPLIR AL MP3

Hay pocos formatos que puedan competir hoy en día con los MP3. Uno de los más populares es el VQF que obtiene una mejor compresión.

Además del MP3 existen diversos formatos de audio como se vio anteriormente, el gran problema es que ninguno, hasta hoy pudo superar la popularidad de este formato. Esto sin contar que a nadie se le pasaría, por ahora por la cabeza, cambiar el formato de miles y miles de archivos.

Entre estos formatos el que más se destaca es el MSaudio el cual, dependiendo del tipo de sonido, puede llegar a "superar" al famoso MP3 solo en una compresión mayor. Esto se debe a que el MSaudio tan solo necesita 64 KHz para una calidad un poco inferior al estándar del MP3 que es de 128 KHz, además un MP3 a 64 KHz por mejor que sea el codificador que se use, no llegaría nunca a la calidad de este formato. Pero este codec (programa codificador/decodificador) no esta aún perfeccionado, ya que no tiene la cantidad de variantes del MP3.

El MSaudio, es por ahora uno de los mejores formatos de compresión de sonido, por lo menos superior al MP3 a un muestreo más bajo.

El problema principal con el MP3 es aquel de la piratería, y a causa de esto se puede pensar que se optará por algunas soluciones como la que el Liquid Audio propone. Nace de una casa productora de software

norteamericana justamente para contrarrestar el fenómeno de la piratería y por lo tanto, de la distribución ilegal de piezas en MP3 todavía protegidas por el copyright. La característica es la de garantizar la protección del copyright, poniendo criptografía en cada archivo de audio, de modo que se pueda leer solamente desde el programa que posee el que lo adquiere. Así tendremos tantas piezas musicales criptografiadas, para poder ser leídas por una única copia de Liquid Audio, y también, si un hacker logra manipular el programa reproductor, sería siempre posible averiguar el nombre de la persona que adquirió, y se presume que difundió, un archivo ilegalmente. Es como si en la copia permaneciese escrito el nombre del comprador.

Seguramente la cosa más interesante ligada al Liquid Audio, sería que quedase la posibilidad de que finalmente se ofrezca al usuario escoger y pagar solamente algunas piezas musicales de un álbum sin tener que comprarlo enteramente. También este tipo de distribución ofrece la esperanza que en el futuro quizás no muy lejano, los costos para adquirir música serían derrumbados desde el momento que desapareciese la impresión de CD's y su distribución a través de los canales clásicos.

También está el a2b que es un formato nuevo y que parece no tener nada que ver con el MP3. Por lo que se puede saber, no tiene una gran diferencia de compresión con el MP3 y además el carácter legal que posee hará que se quede un poco apartado del mundo del "internauta de a pie".

La música a2b no utiliza la tecnología del MP3, no es el algoritmo de compresión basado en el MP3, es, según AT&T un sucesor de él. La compresión de sonido a2b es MPEG-2 AAC Low Complexity Profile Audio

Coding. El AAC no está basado en layer 3 como parece dar a entender la denominación de "MP4" que se le ha venido dando a este nuevo formato.

Según los creadores del a2b, el MP3 tiene ciertas limitaciones que el nuevo formato supera y mejora con amplitud.

MP3Pro es otro formato nuevo aparecido en escena, cuyo primer codec apareció el 14 de Junio del 2001, es responsabilidad de Coding Technologies, una empresa con sede en Estocolmo, que trabajó ayudada por 12 desarrolladores del Instituto Fraunhofer. Los derechos de explotación pertenecen a Thomson Multimedia.

Básicamente, es lo mismo que su hermano mayor, con la diferencia de que su capacidad de compresión lo convierte en el algoritmo más poderoso existente al día de hoy en razón de calidad a tamaño. MP3Pro consigue meter 2 minutos de audio en un megabyte con una calidad similar a la de un disco compacto.

Para conseguirlo separa las frecuencias altas y bajas, las bajas las codifica de la manera tradicional y con las altas emplea las nuevas mejoras. Los nuevos reproductores que vayan saliendo tendrán en cuenta ambas frecuencias, y los viejos reproductores de MP3 ignorarán las altas, por lo que un archivo MP3Pro no sonará con toda la calidad posible, pero la diferencia será inapreciable.

Una prueba casera ha arrojado estos resultados: La canción Wild Horses de los Rolling Stones que en formato WAV (el formato empleado en los CD's) ocupa 57.2 MB, en MP3 se queda en 7.7 MB, y en MP3Pro en 4.2 MB.

Esto significa, entre otras cosas, que todos esos discos duros repletos de

canciones MP3, ahora podrán albergar el doble de títulos. Y esto no sólo es positivo para los usuarios habituales de música digital, que ahorrarán sobremanera en recursos de sus computadoras, también es beneficioso para los sistemas de intercambio de música P2P como el sitiado Napster o el reciente AudioGalaxy. Todo indica que si MP3Pro se impone, habrá más canciones que compartir y las transmisiones serán mucho más rápidas.

Pero ni MP3 ni MP3Pro son software libre, por lo que sus padres Thomson Multimedia y el Instituto Fraunhofer pueden ejercer sus derechos de explotación. Sin entrar en la polémica de las patentes de software, de si coartan el progreso tecnológico o de si un algoritmo es patentable, lo cierto es que es un derecho que ejercitan legítimamente.

Con una diferencia de pocos días después de la aparición de MP3Pro, irrumpió la versión 1.0 de OGG Vorbis, un formato de compresión creado por un grupo de desarrolladores independientes que no sólo es bueno, también es "opensource", por lo que no está sujeto a "royalties" como MP3Pro. Si alguien quiere programar un reproductor (por ejemplo) nadie tiene que desembolsar un solo dólar.

Empresas de software como Sonic Foundry o WinAmp (de AOL Time Warner), además de varios sitios web, ya lo han adoptado. OGG Vorbis se encuentra con el mismo freno a su expansión que los competidores de MP3.

El MP3 es, con diferencia, mucho más popular que cualquier otro formato, y hay millones de reproductores de MP3 que son compatibles con MP3Pro.

Superar al MP3 es difícil, el único que podría robarle el puesto a este ubicuo archivo es su descendiente, MP3Pro. El software de manejo del MP3

es diverso y de muy buena calidad, invita a ser usado y a divertirse con él. El MP3 es el estándar que ha llegado para quedarse y aún va por más. A medida que el hardware multimedia sea cada vez más accesible al usuario casero, la proliferación del audio comprimido también será más accesible, mejorando en calidad. Solo nos resta esperar qué nos deparan los genios de la compresión. ¿Cuál será el formato definitivo?, Lo nuevo está a la vuelta de la esquina, la multimedia a través de la red Internet llega a nuestros hogares sin intermedio ninguno, una vez más es el público en línea, consumidor potencial, quien decide el rumbo del diseño e implementación informáticos. La pesadilla de las discográficas continúa.

PRUEBAS

Las pruebas han sido realizadas con una computadora IBM 2170, modelo 13T con procesador AMD K6-II de 450 MHz y 56 MB de RAM.

Están divididas en tres partes; la primera parte corresponde a "Canciones" divididas en cuatro géneros, una canción por cada género. La segunda parte corresponde a "Señales Sinusoidales", de 20 Hz hasta 20480 Hz y por último "Voces", tres archivos de voces masculina y femeninas.

Se trabajó con archivos monofónicos a 44100 Hz, como es la condición para poder utilizar el codificador MP3 que se está estudiando y probando.

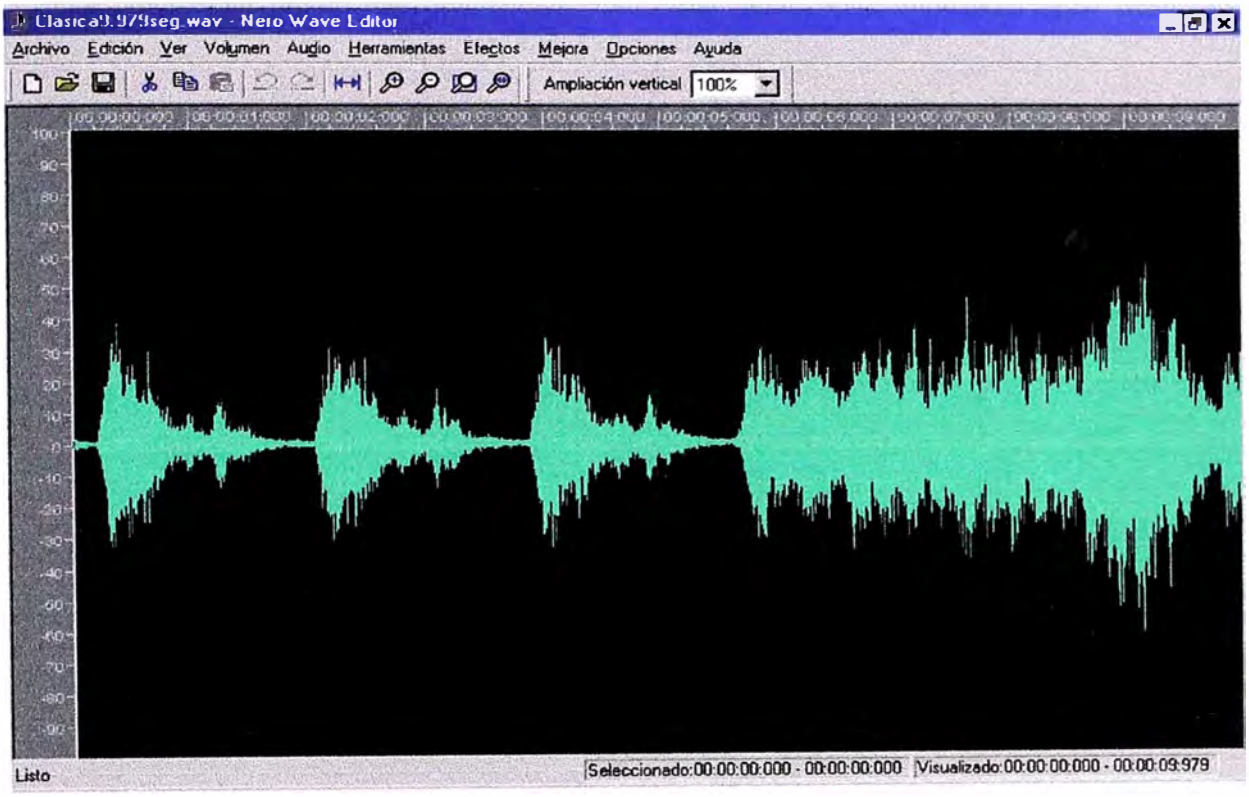
Se utilizaron archivos de 8 bits (calidad telefónica) y de 16 bits (calidad CD y DAT) para poder ver el comportamiento y resultados del codificador así como el peso de los archivos resultantes.

A continuación se podrá apreciar los gráficos de las ondas WAV originales, editados para tener la misma longitud de tiempo que los archivos codificados a partir de ellos, para poder ver en el mismo lapso de tiempo los resultados y poder hacer las comparaciones.

Por cada señal existe un gráfico de onda y un espectrograma donde se verán las variaciones sufridas por la señal WAV original al codificarla a ocho diferentes tasas de bits en Kbps.

Canciones

Visualización de Onda



Visualización de Espectrograma

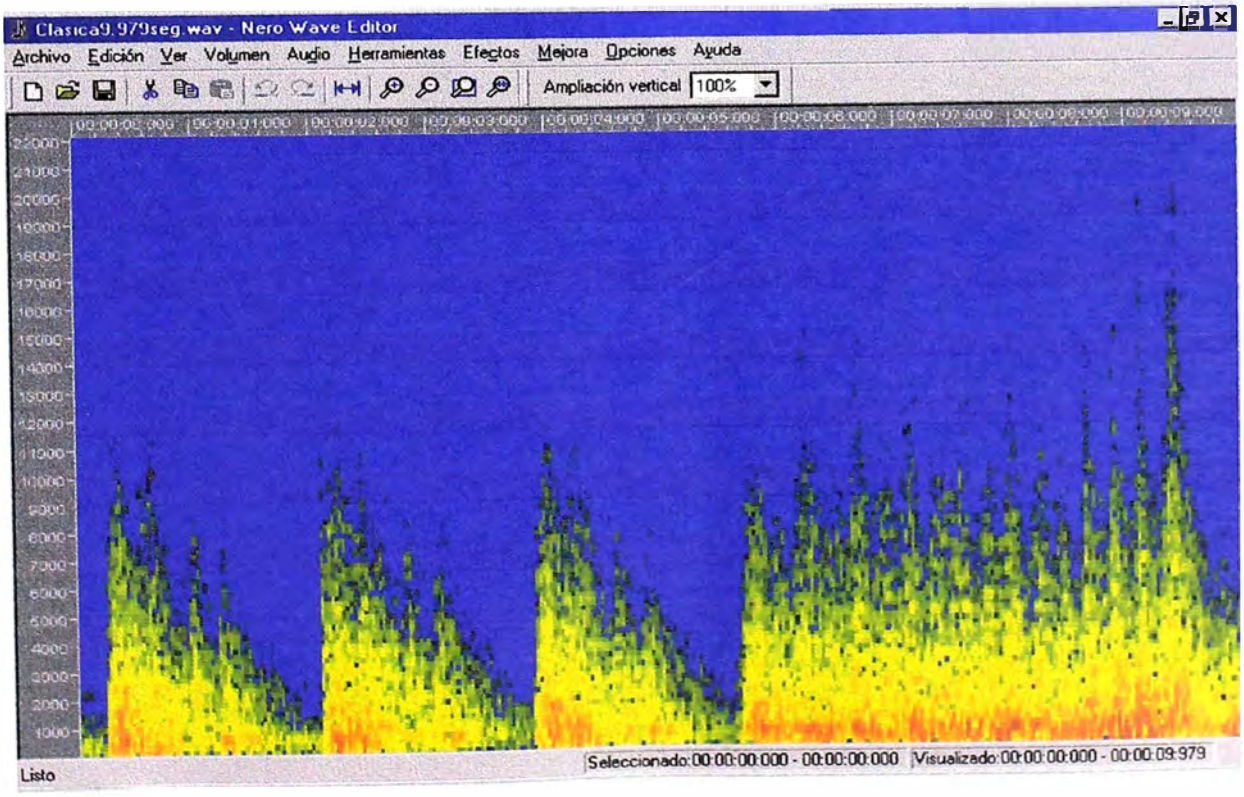
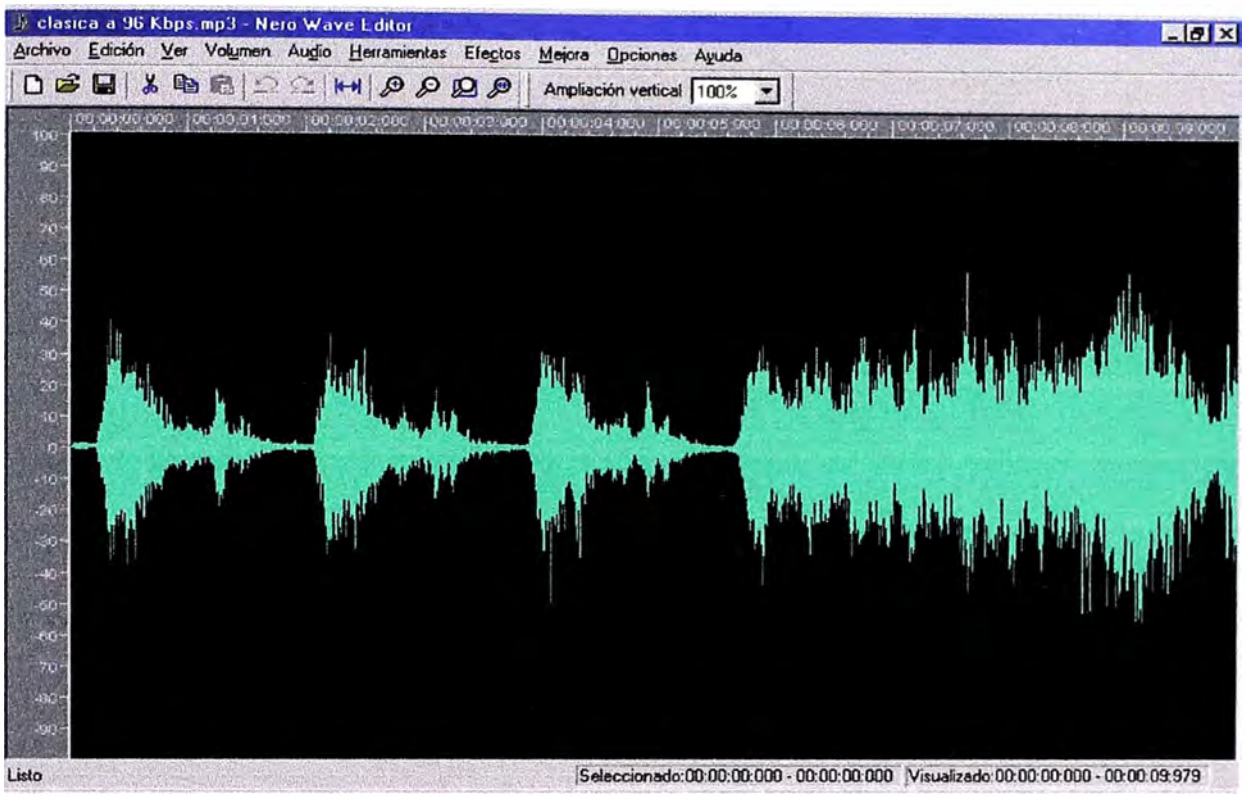


Figura PC1: **Clasica9.979seg.wav**

Visualización de Onda



Visualización de Espectrograma

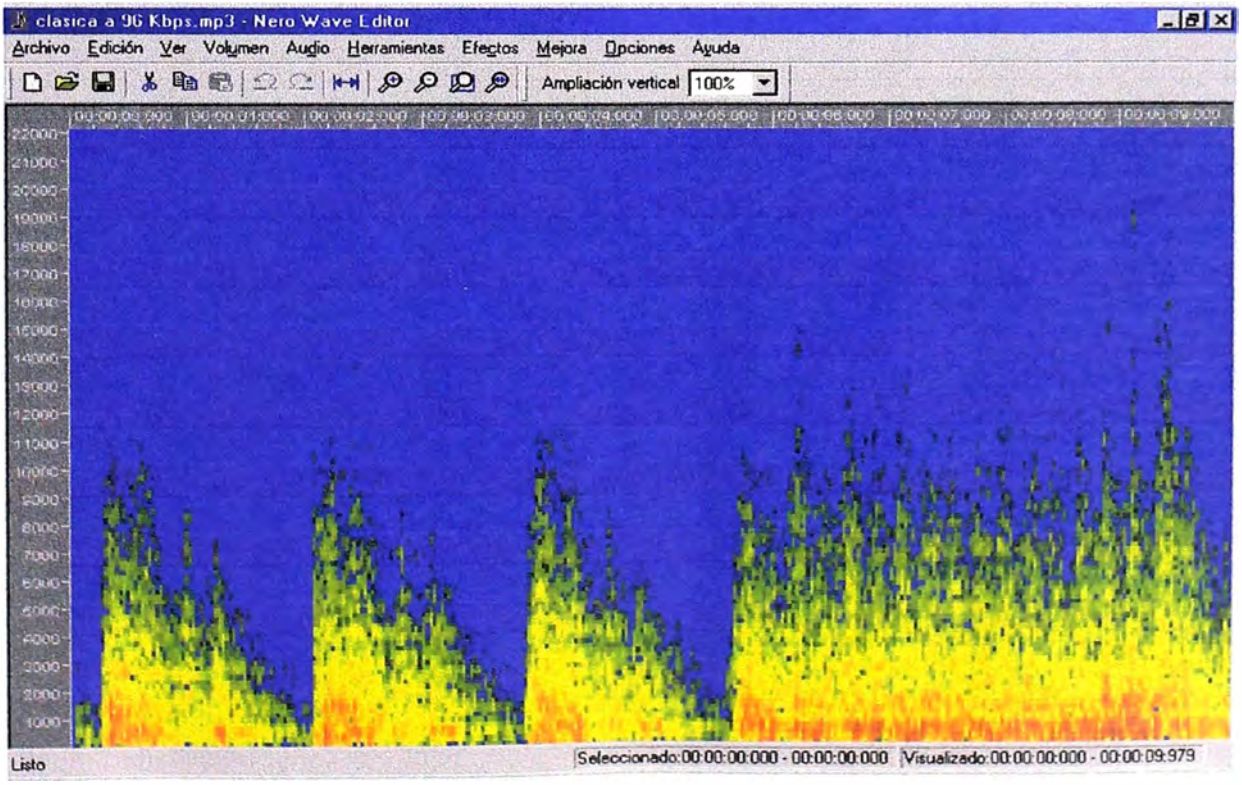
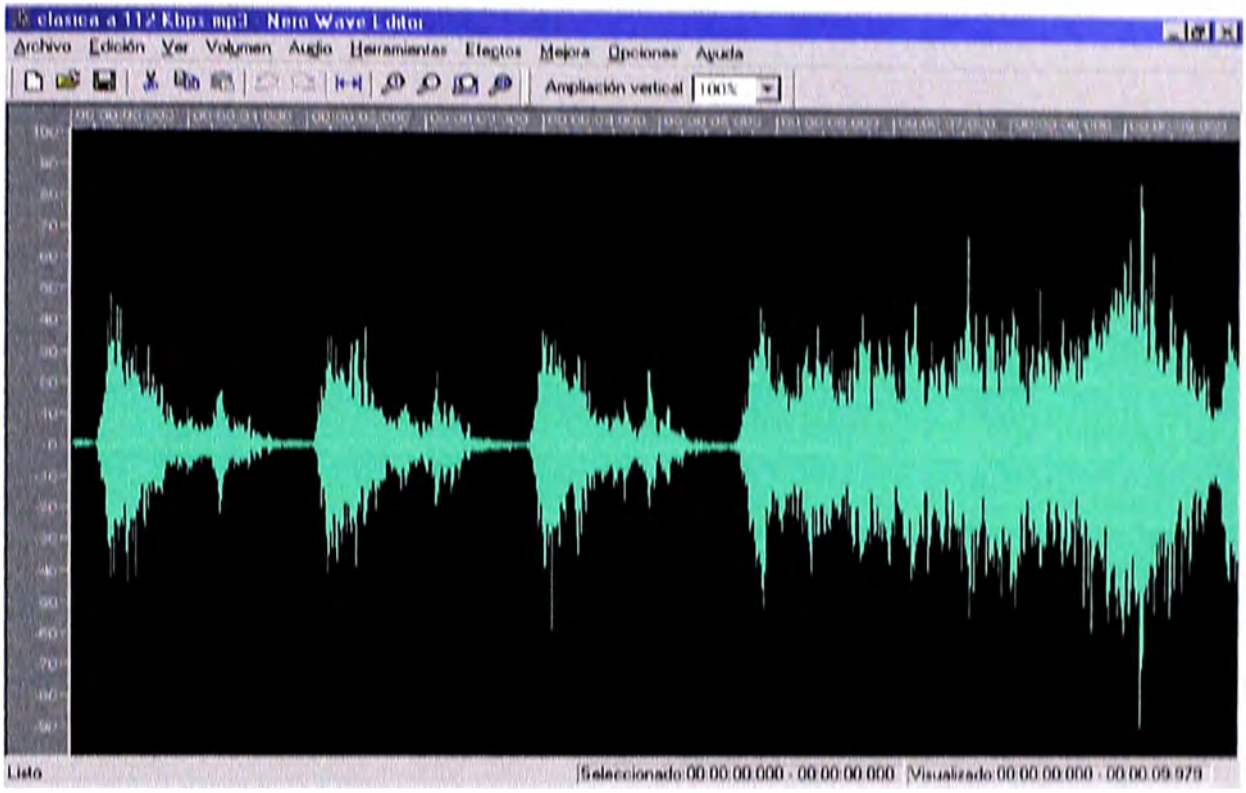


Figura PC2: **Clásica a 96 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

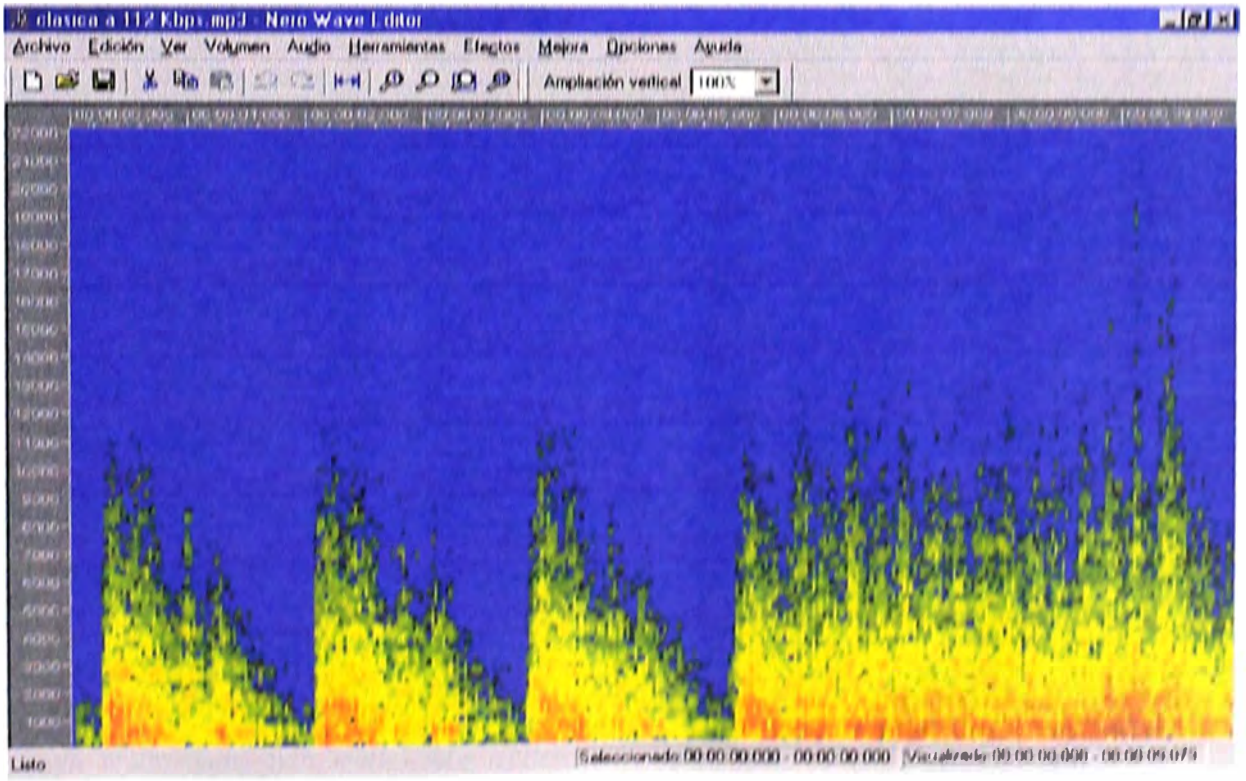
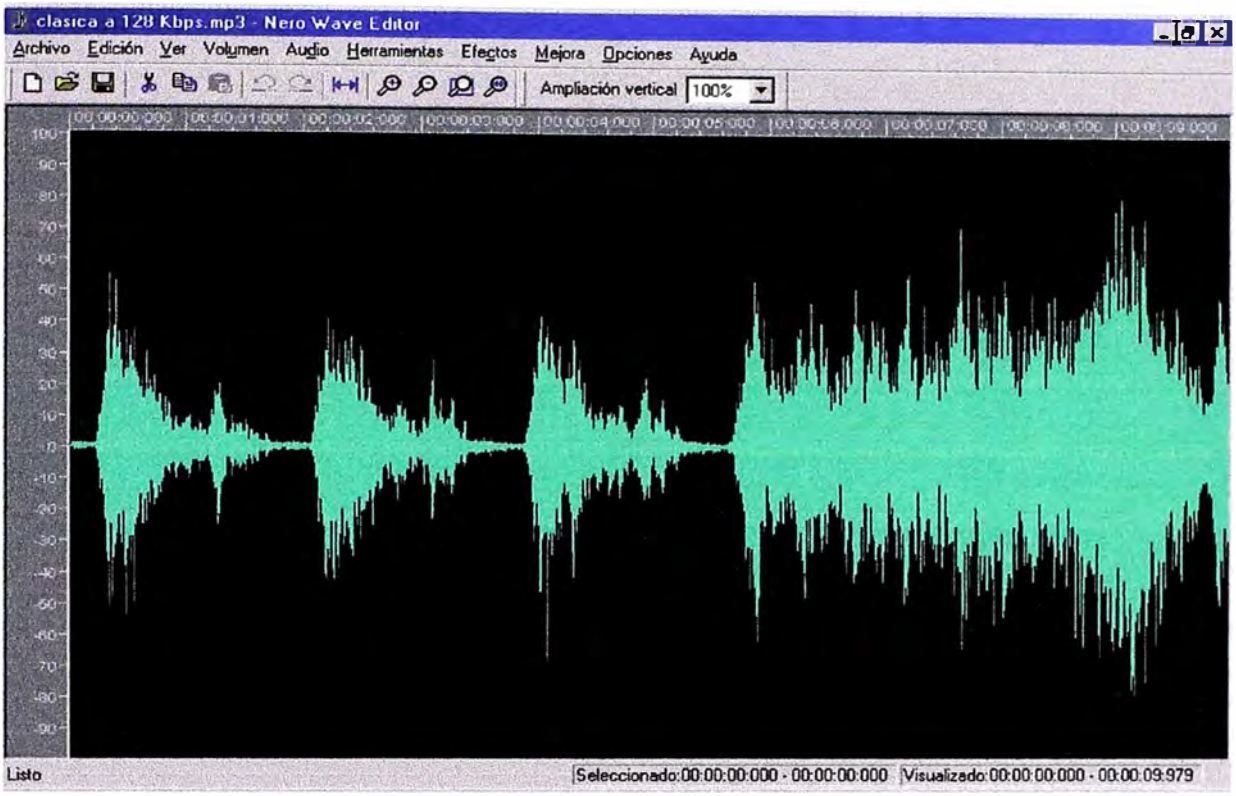


Figura PC3: **Clásica a 112 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

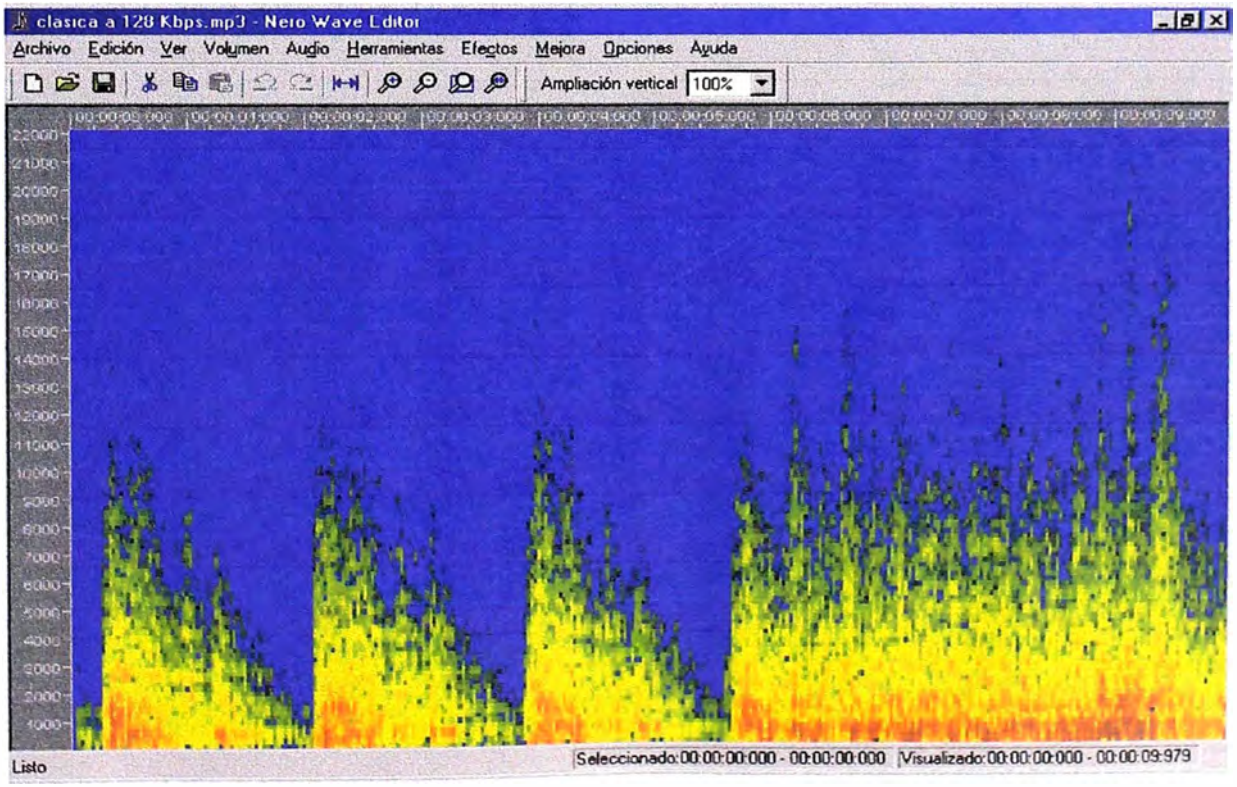
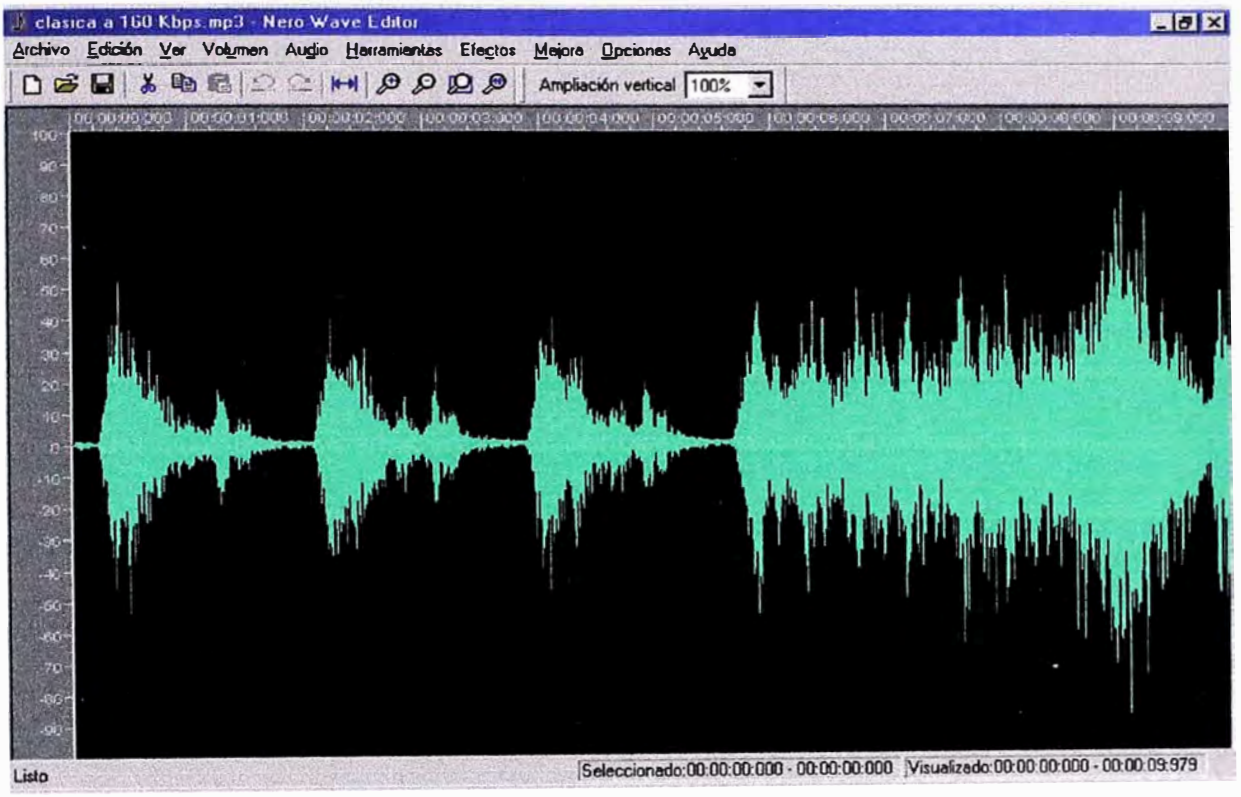


Figura PC4: **Clásica a 128 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

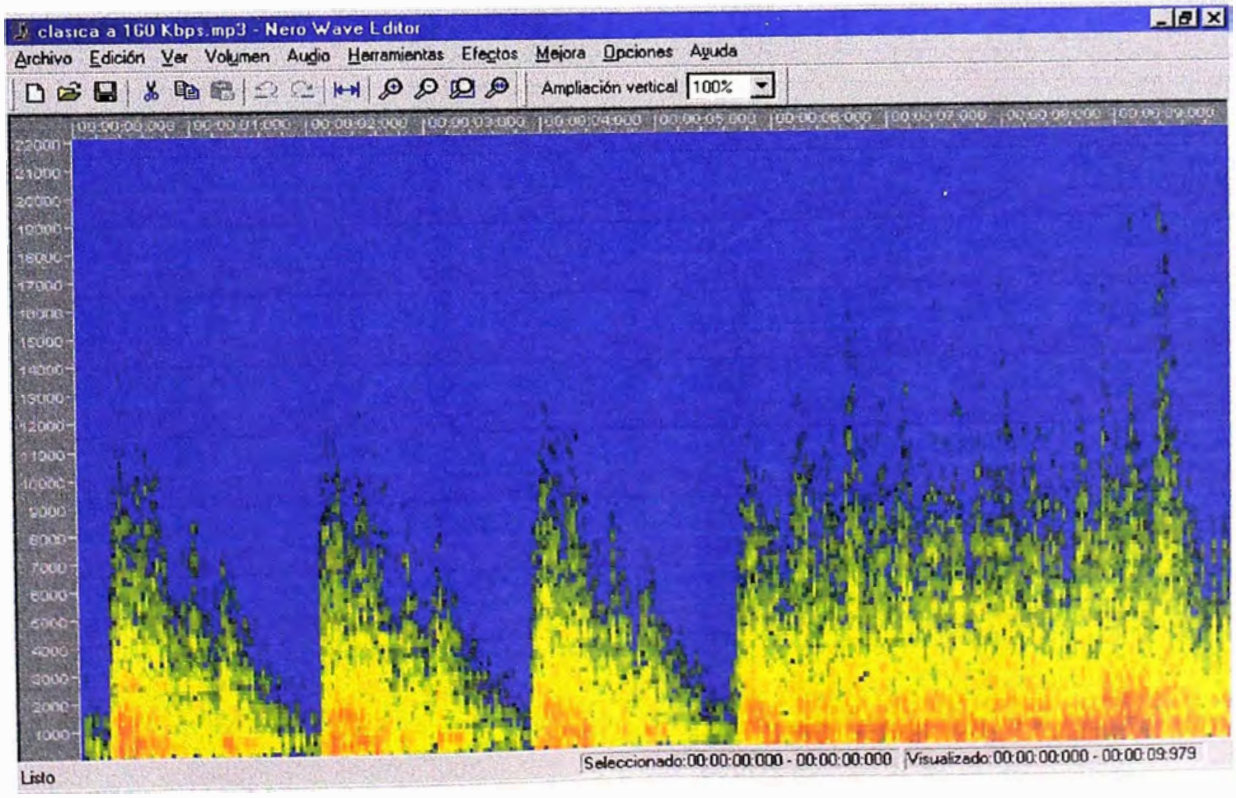
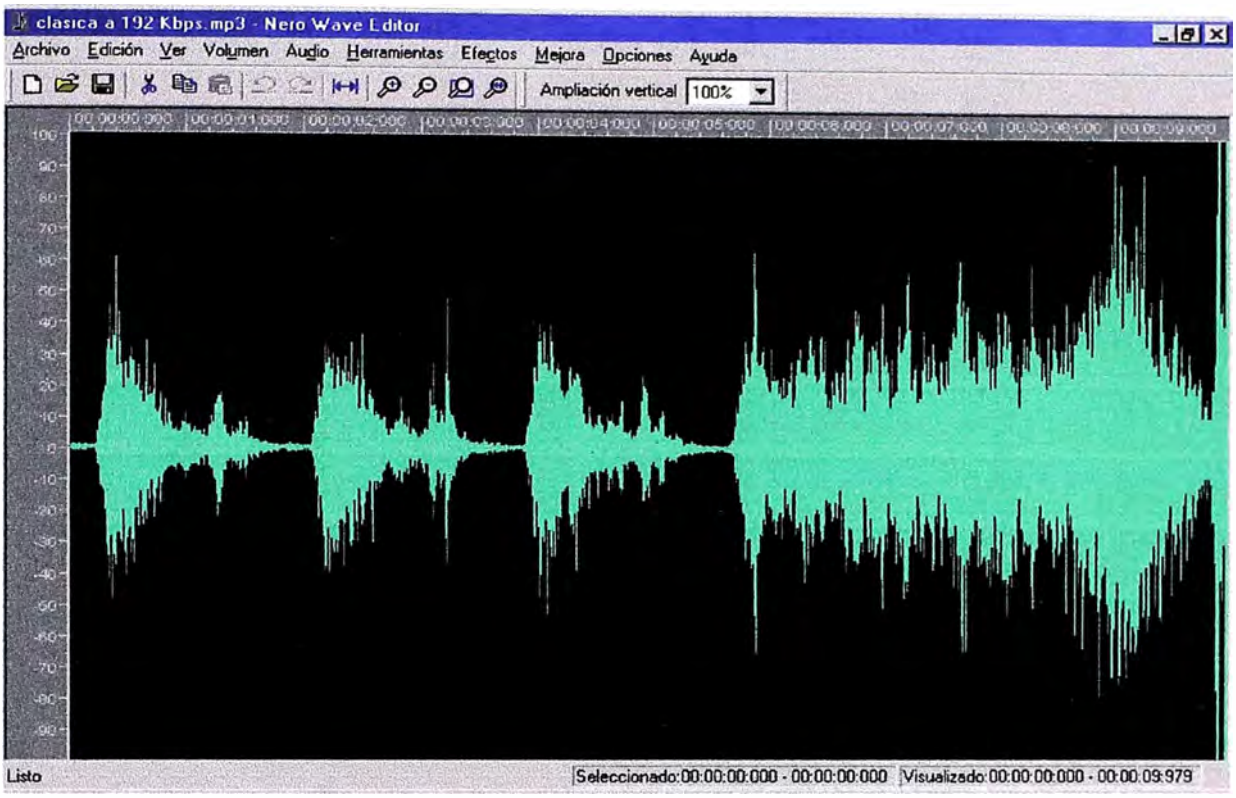


Figura PC5: **Clásica a 160 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

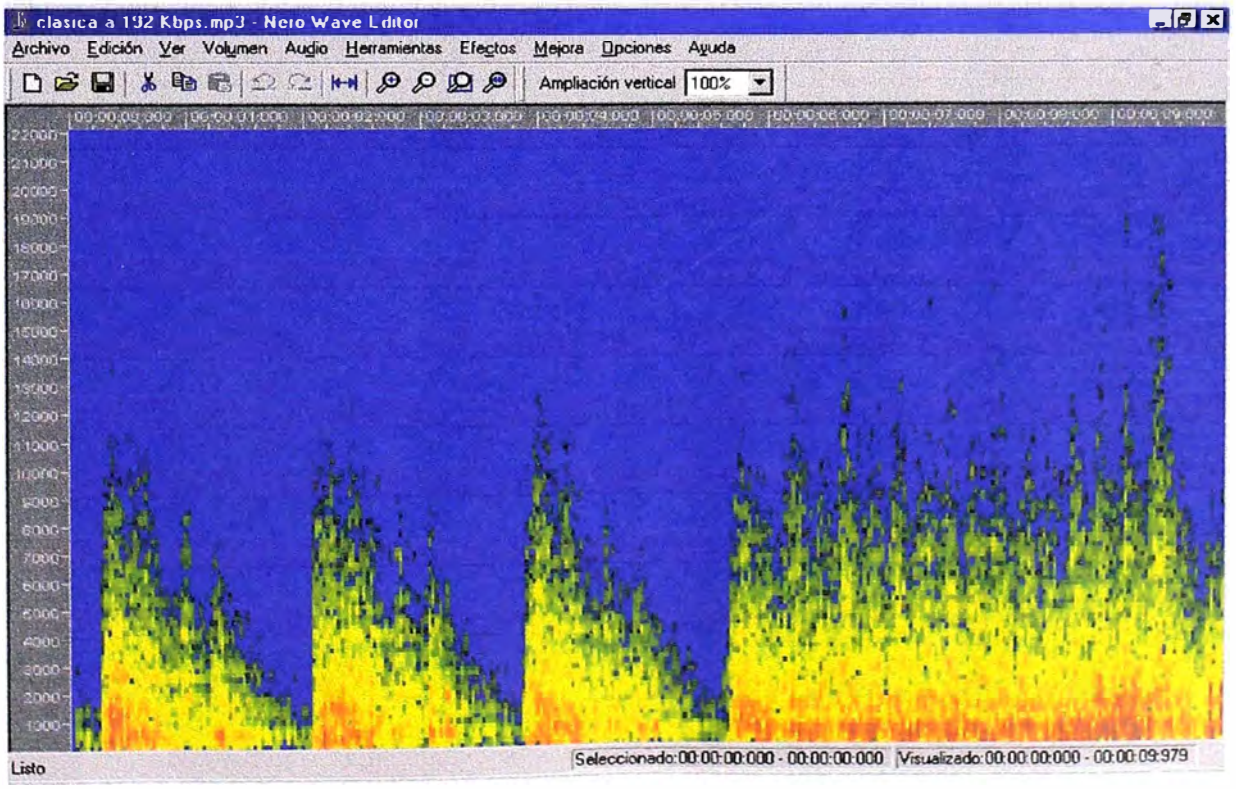
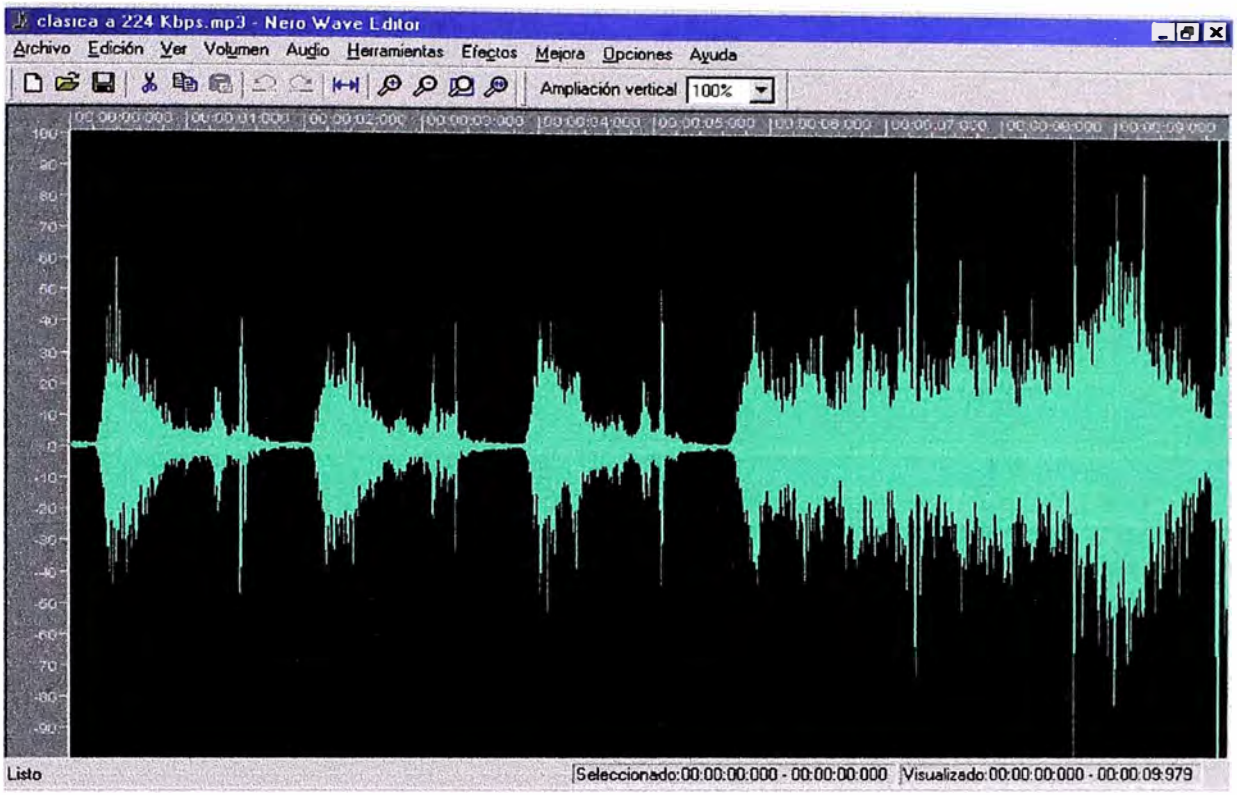


Figura PC6: **Clásica a 192 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

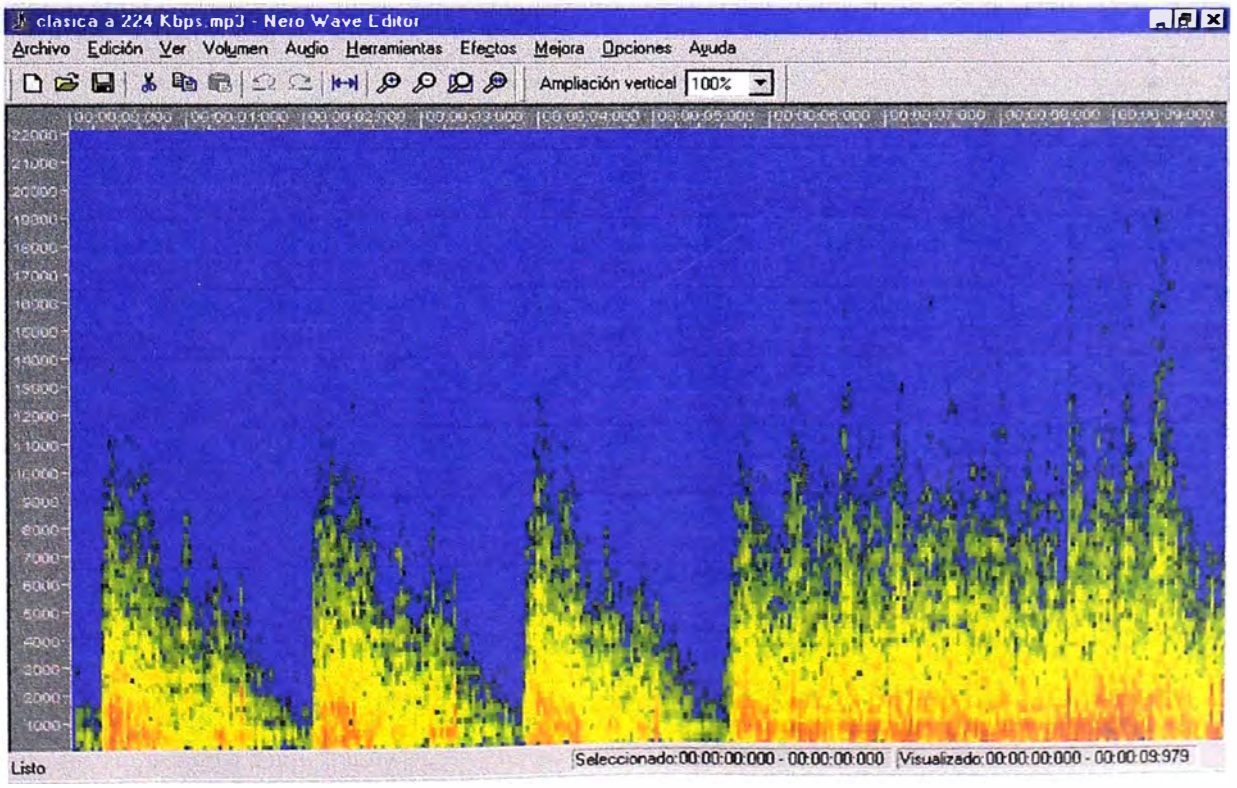
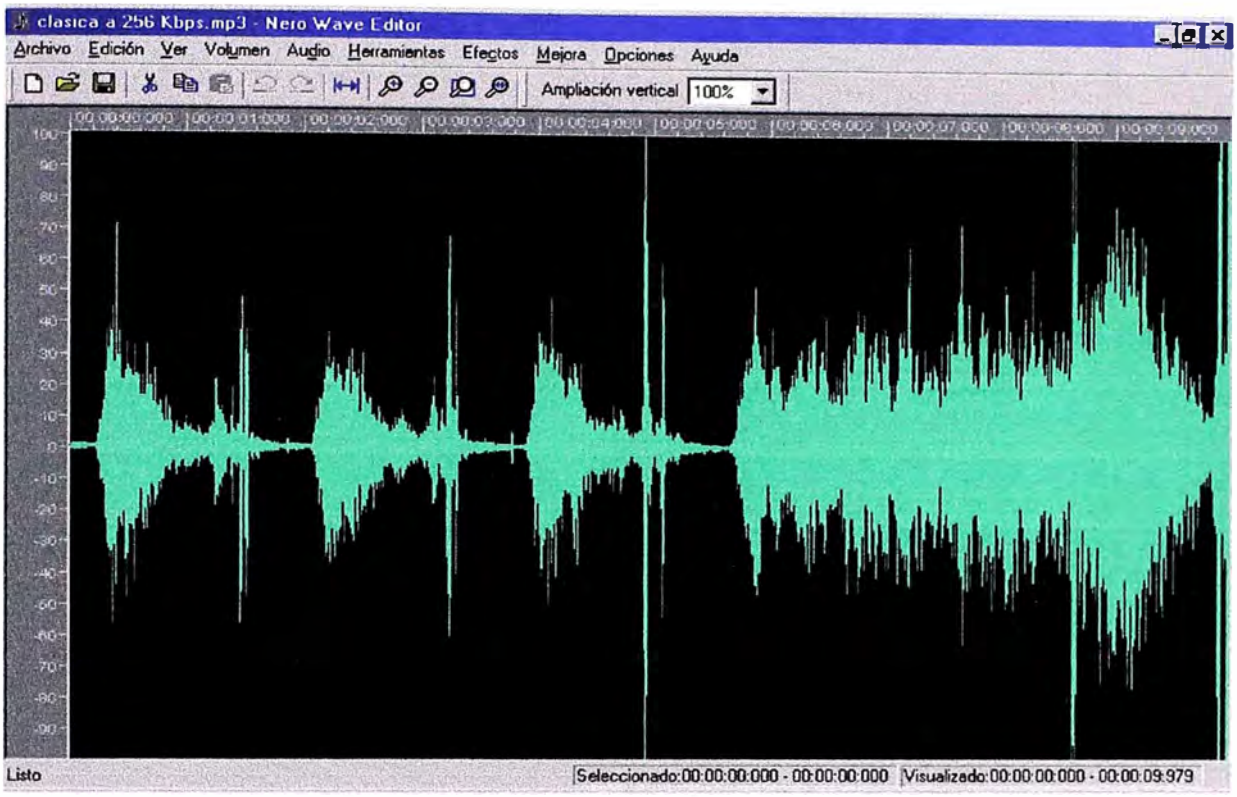


Figura PC7: **Clásica a 224 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

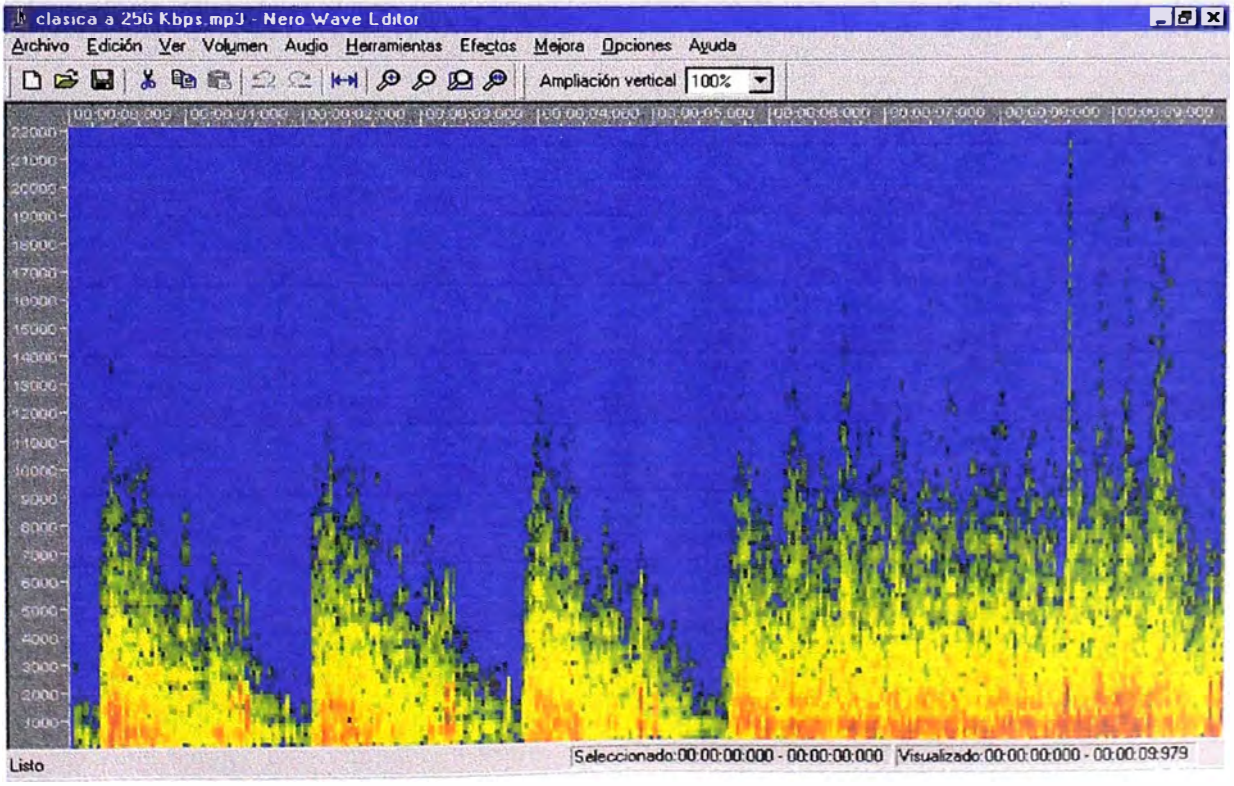
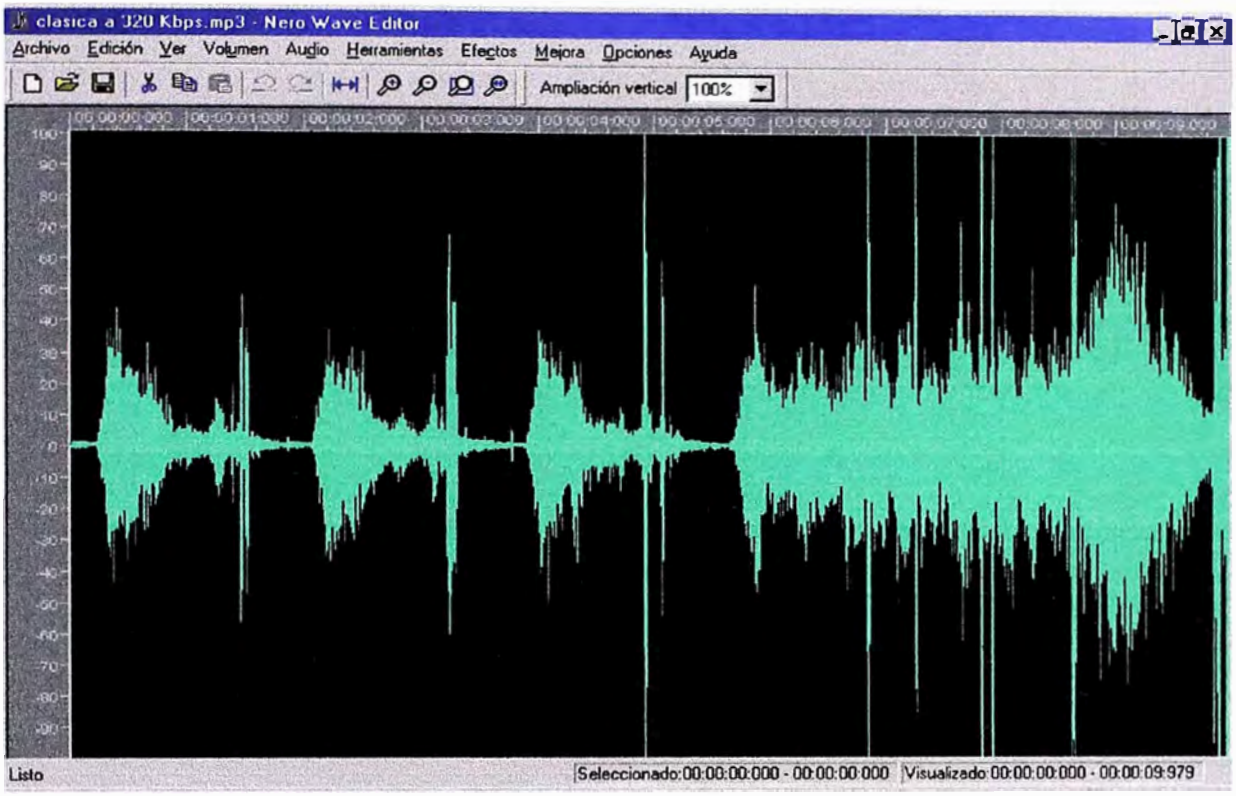


Figura PC8: **Clásica a 256 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

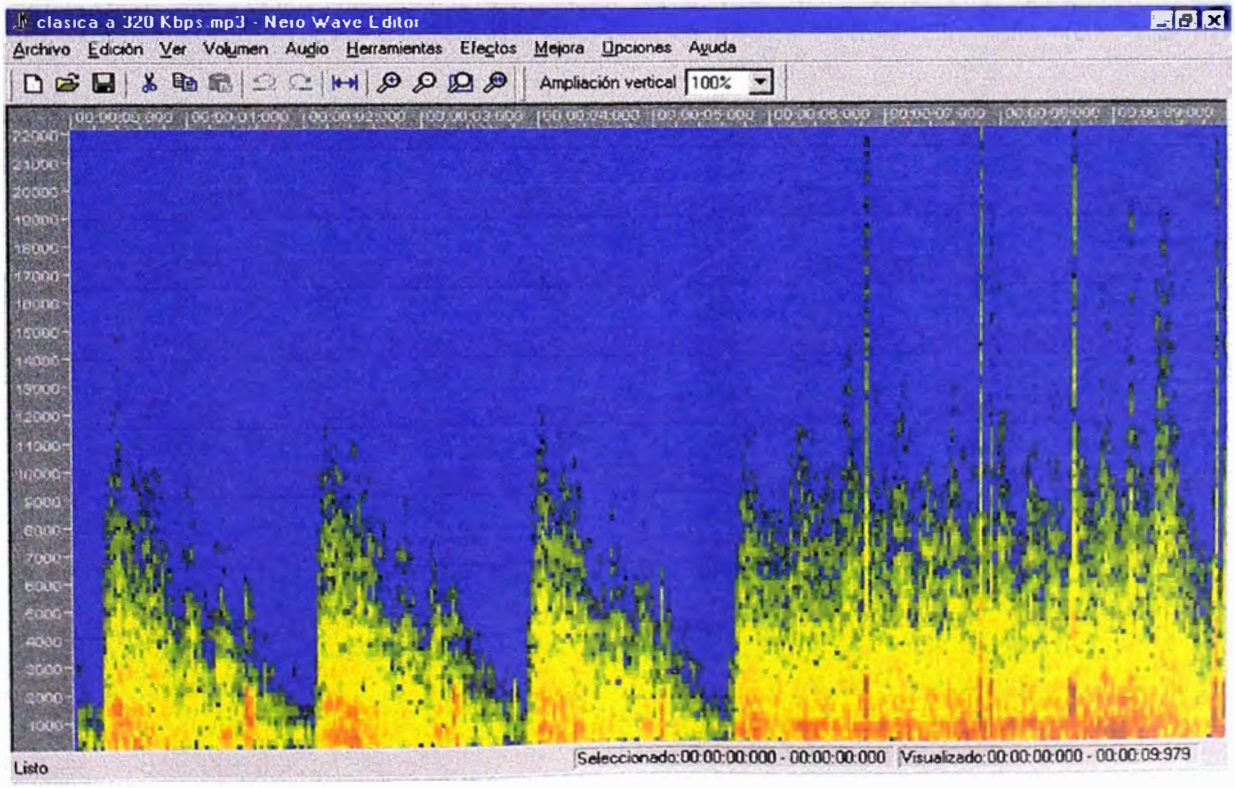
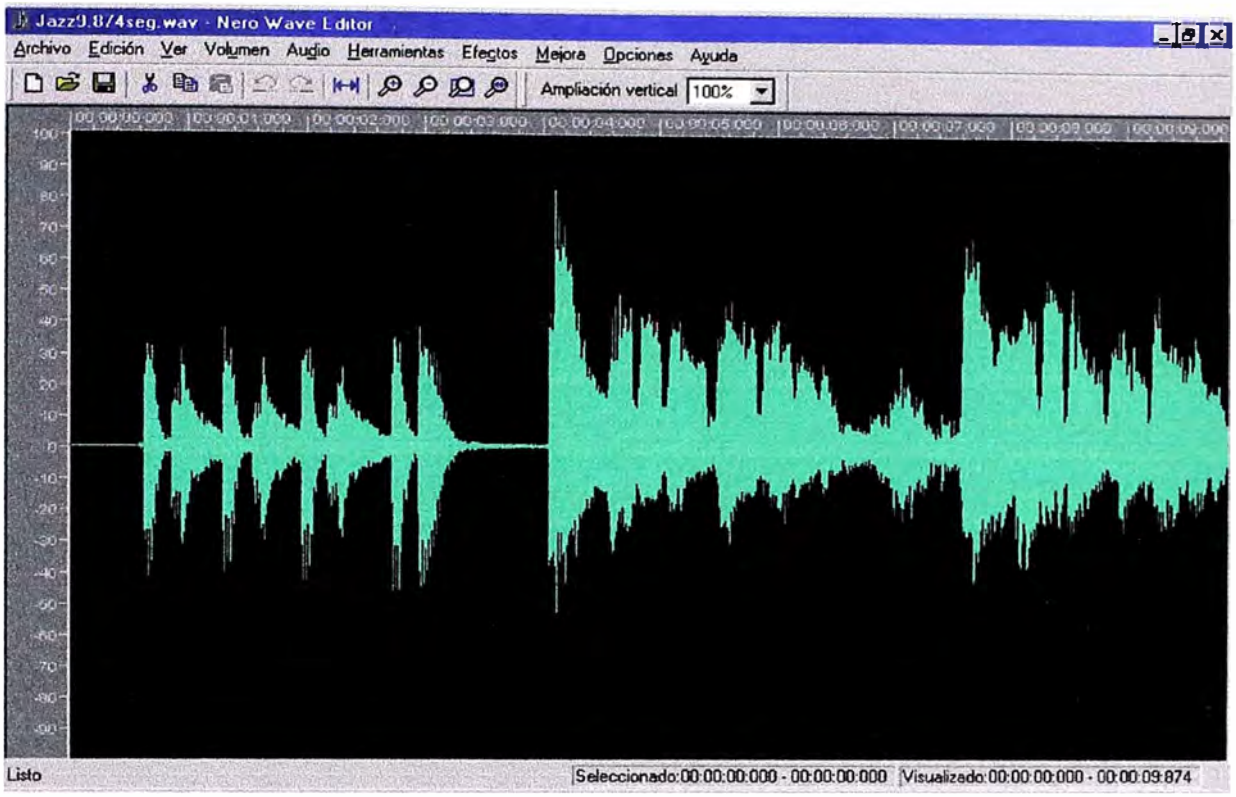


Figura PC9: **Clásica a 320 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

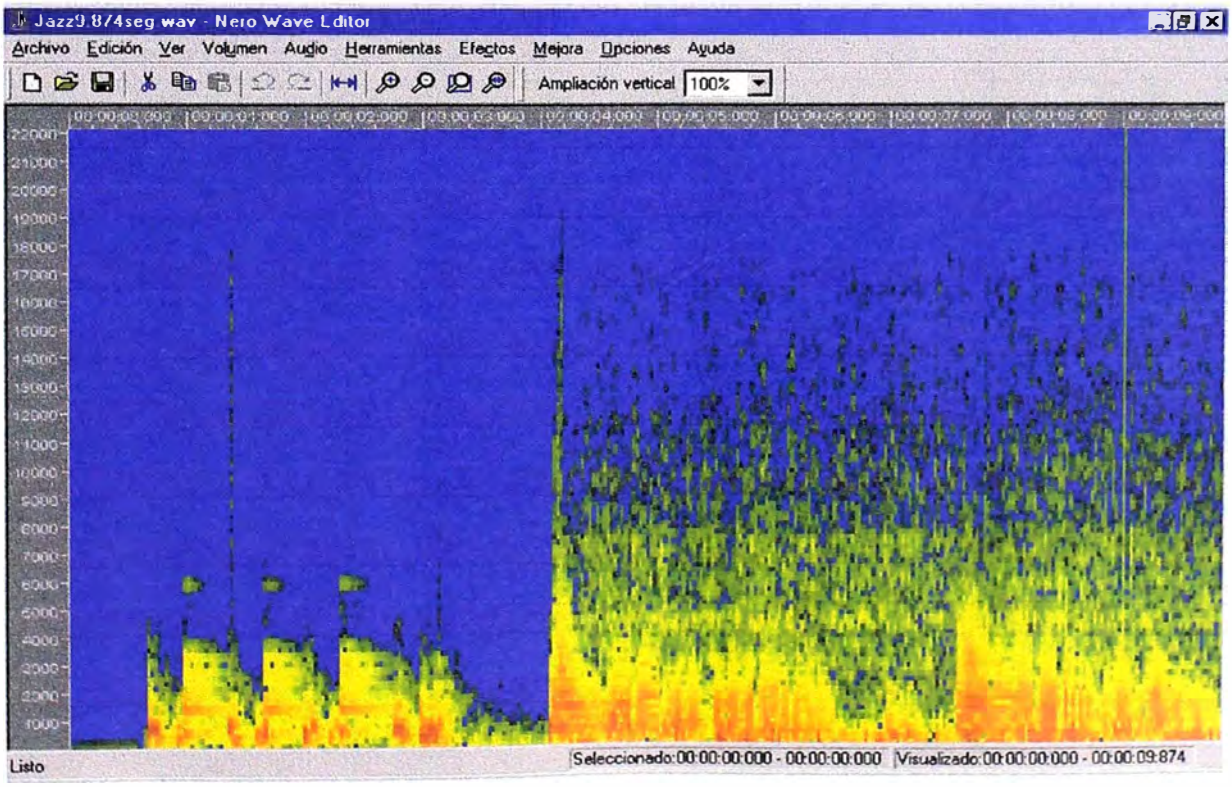
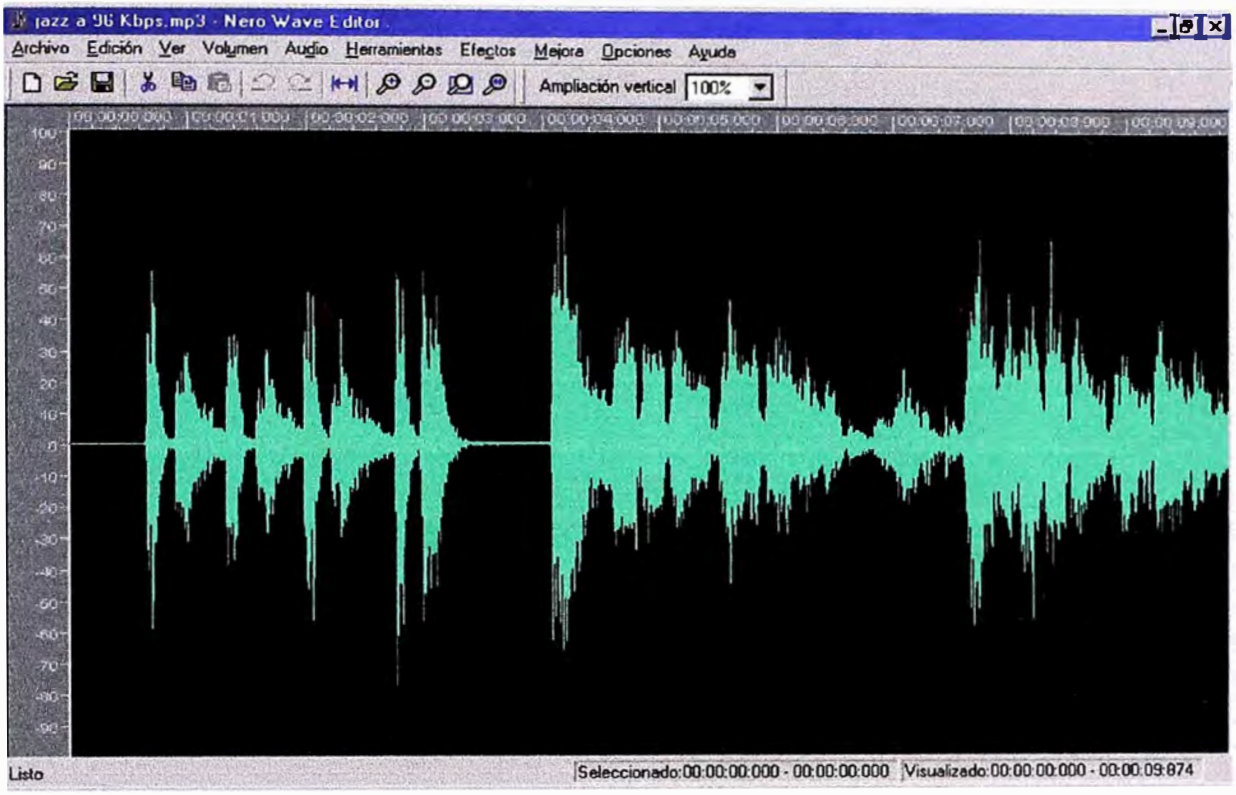


Figura PC10: **Jazz9.874seg.wav**

Visualización de Onda



Visualización de Espectrograma

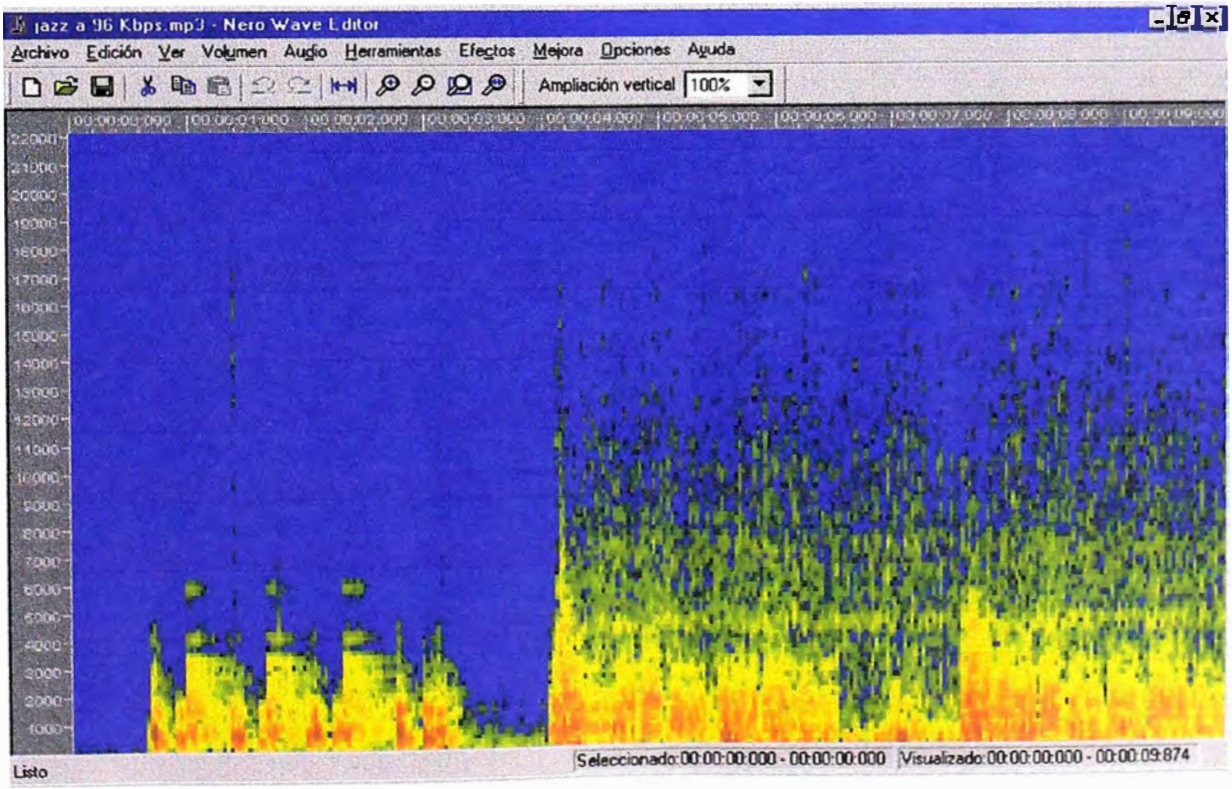
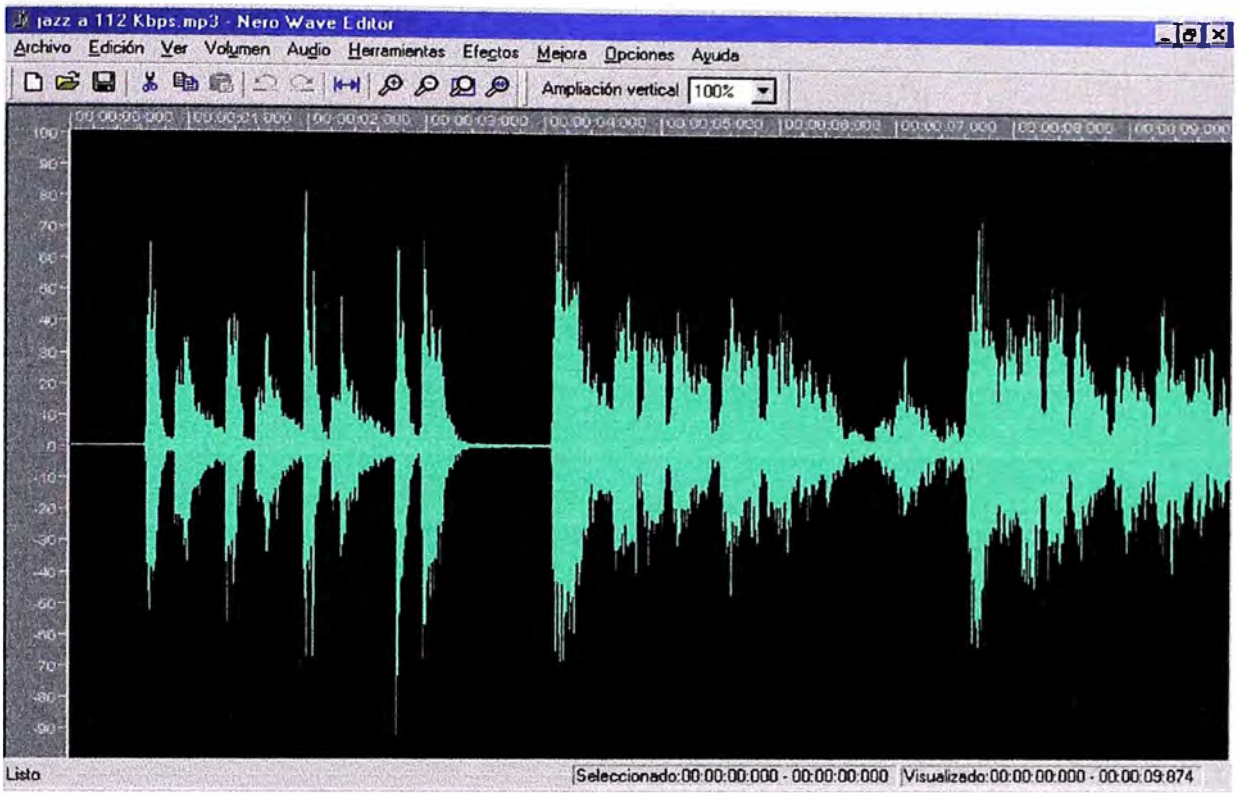


Figura PC11: **Jazz a 96 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

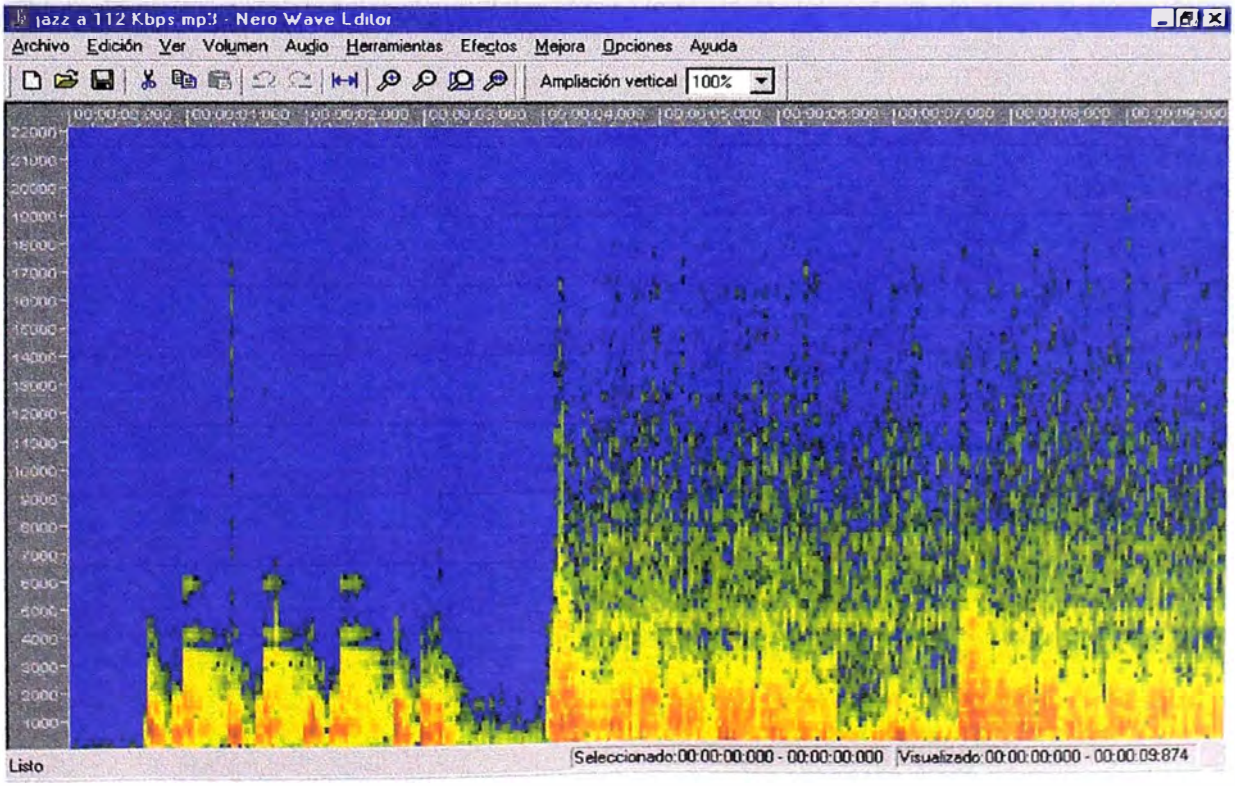
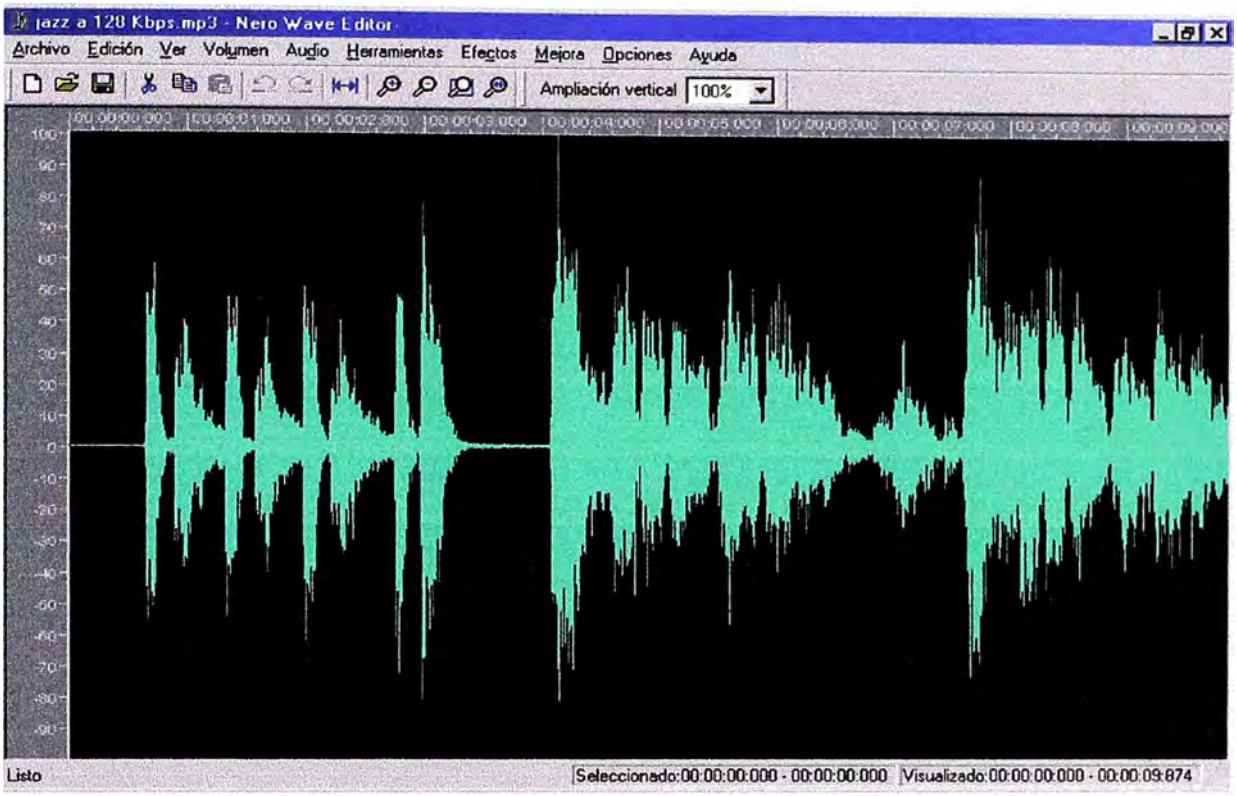


Figura PC12: **Jazz a 112 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

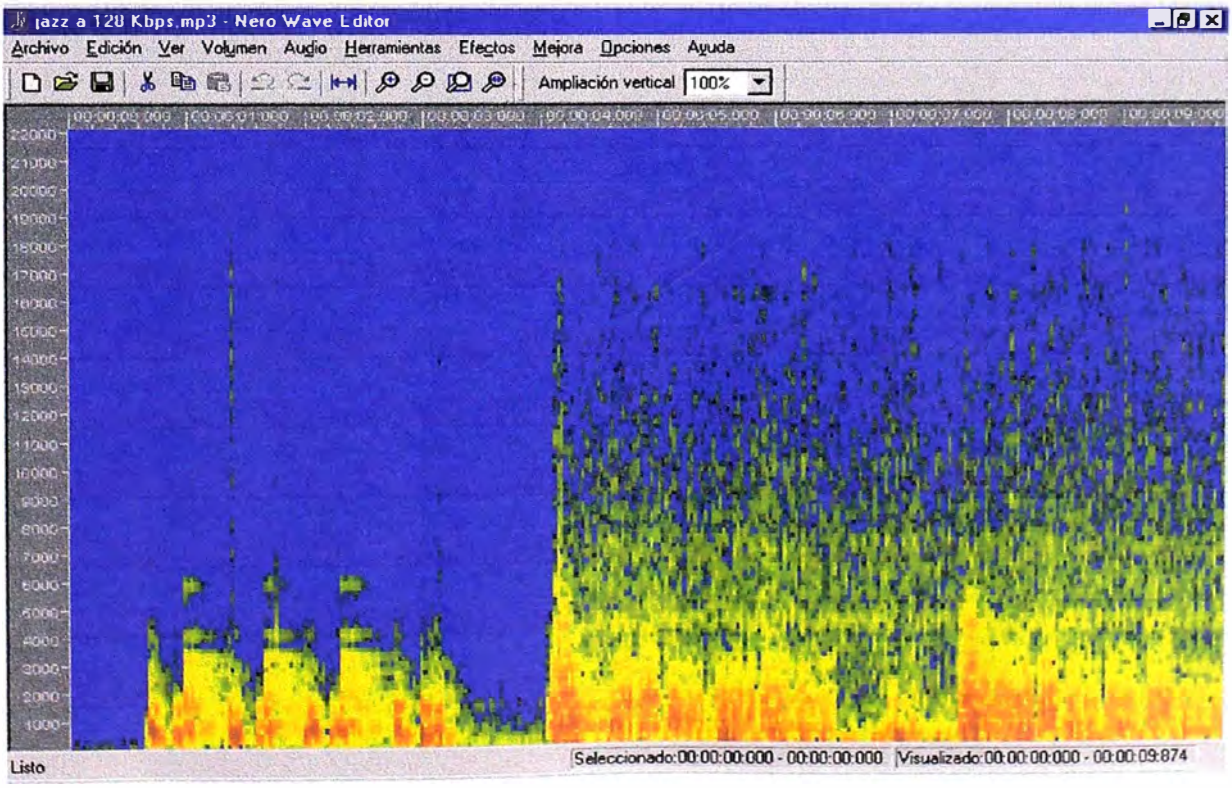
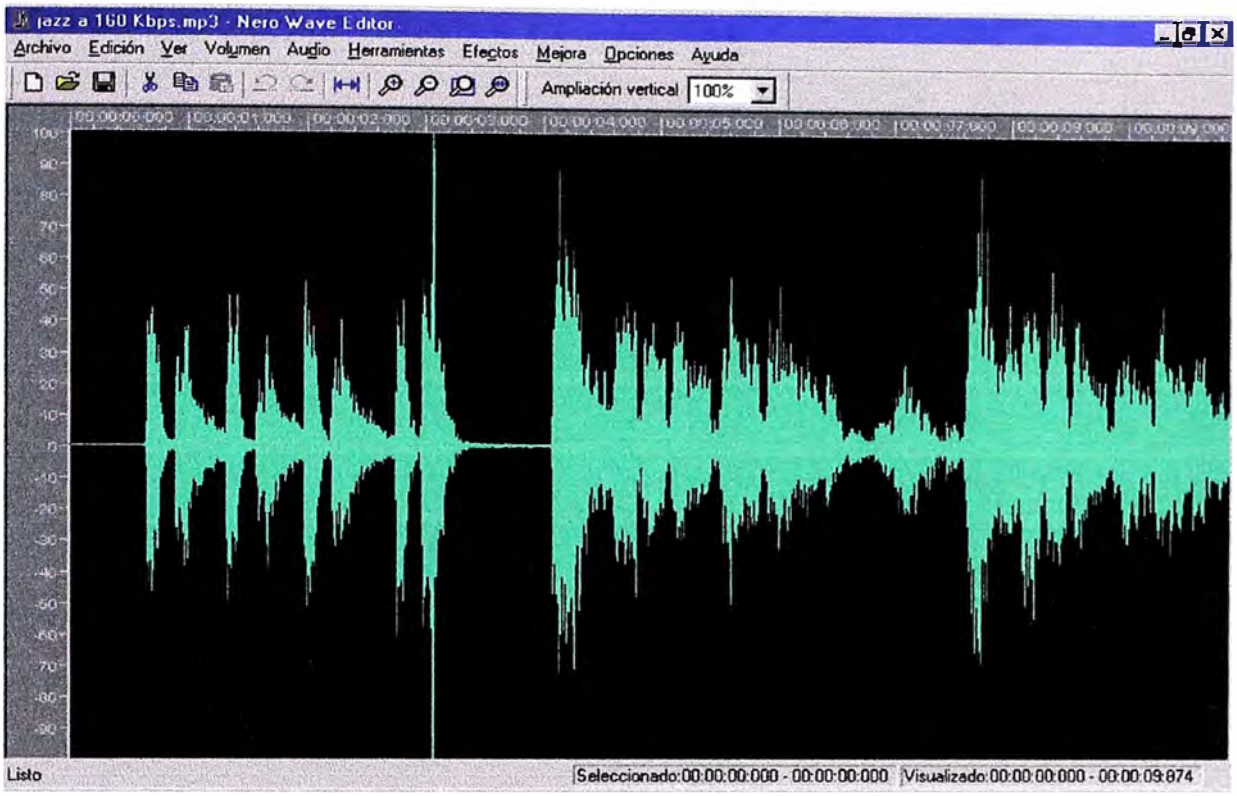


Figura PC13: **Jazz a 128 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

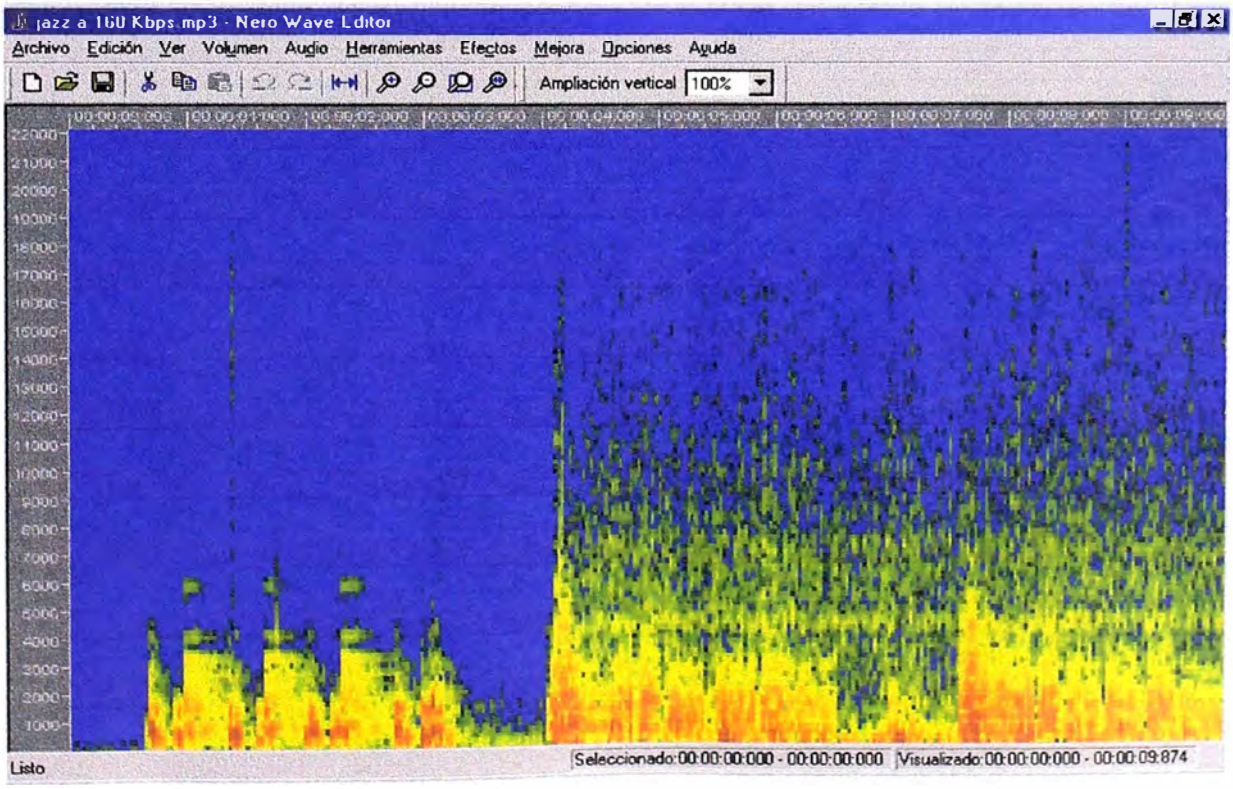
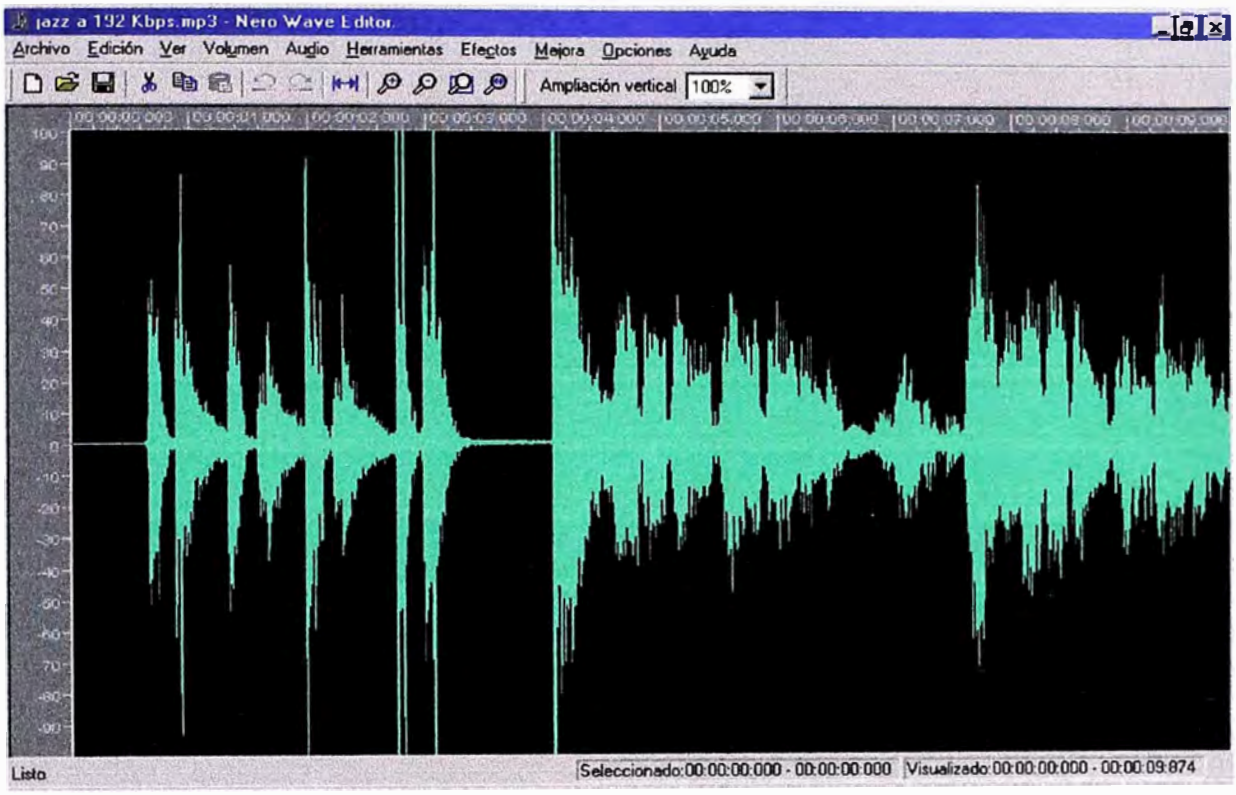


Figura PC14: **Jazz a 160 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

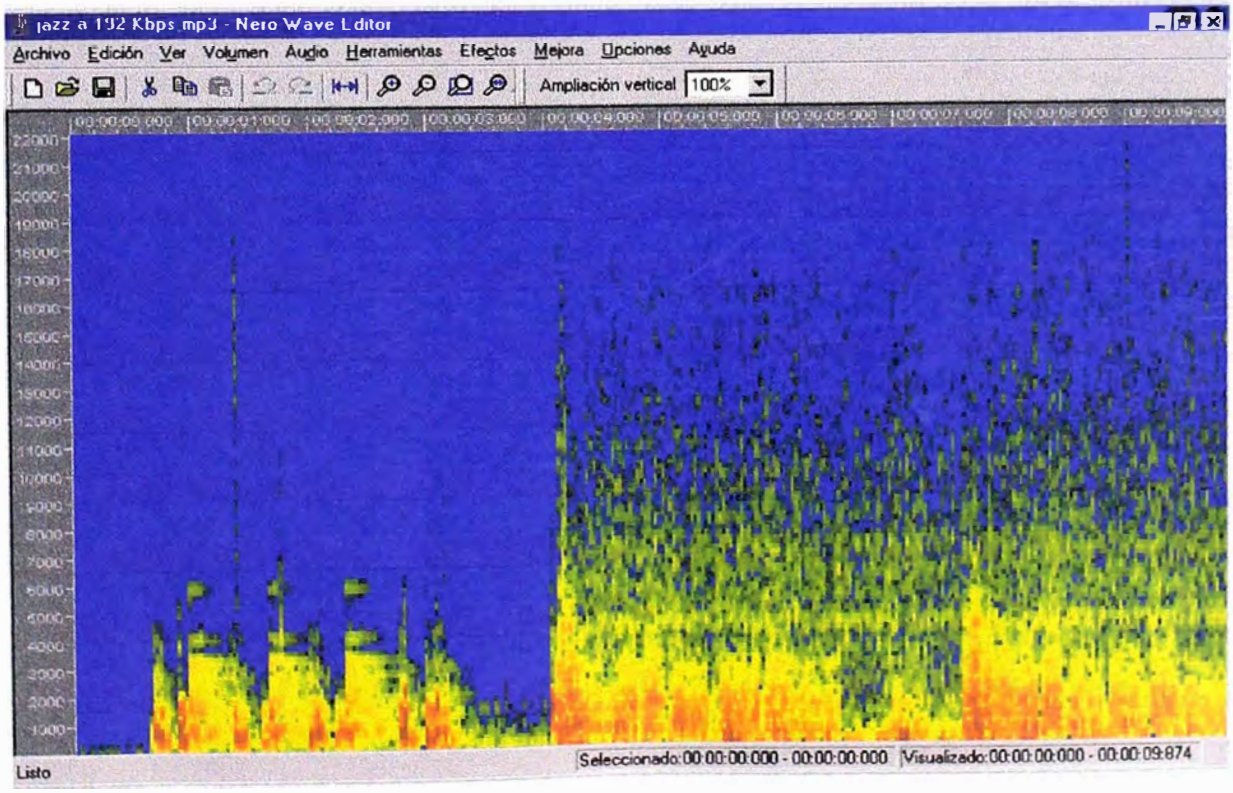
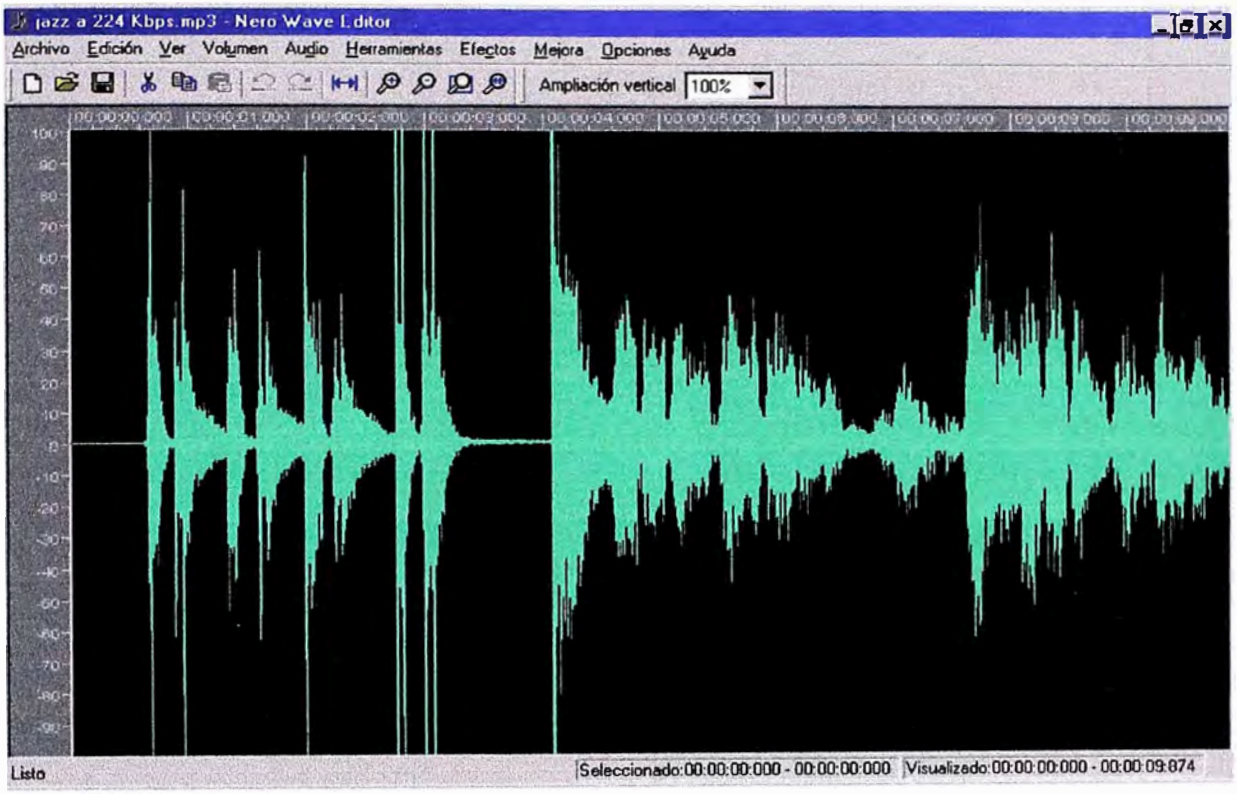


Figura PC15: **Jazz a 192 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

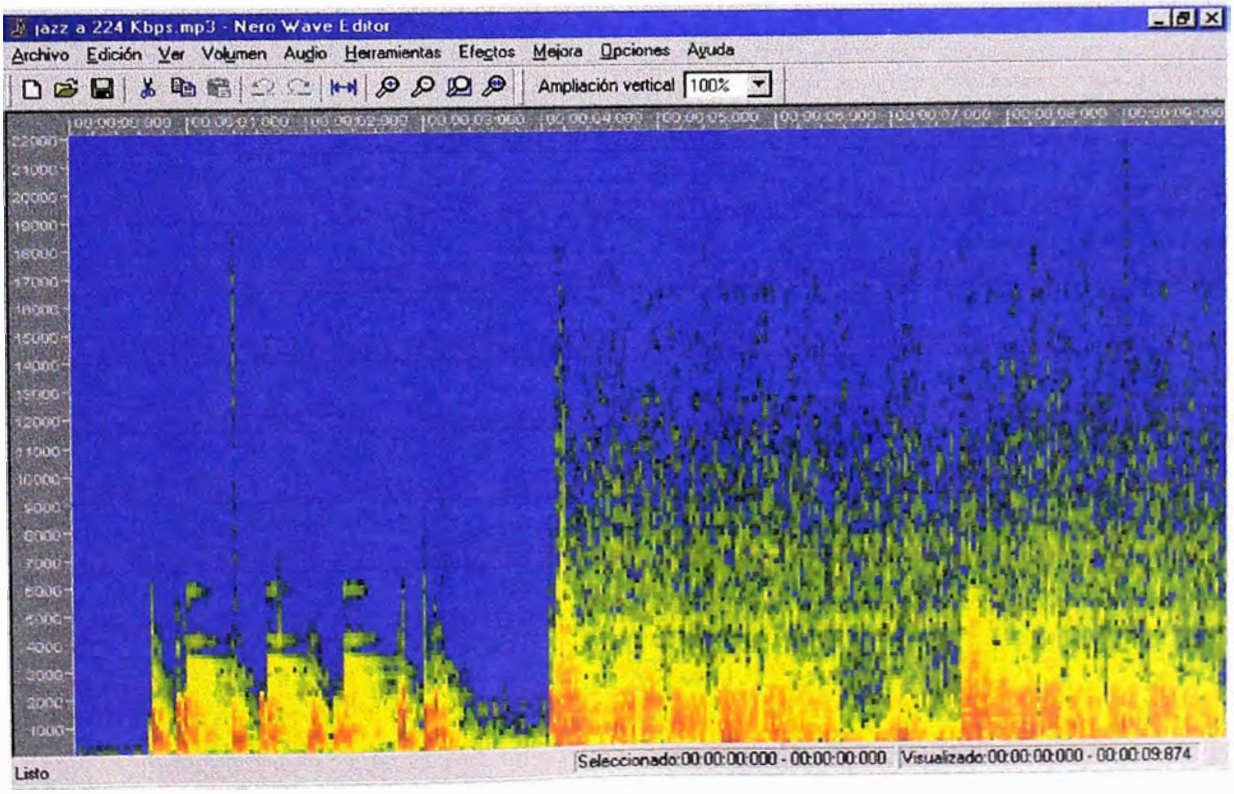
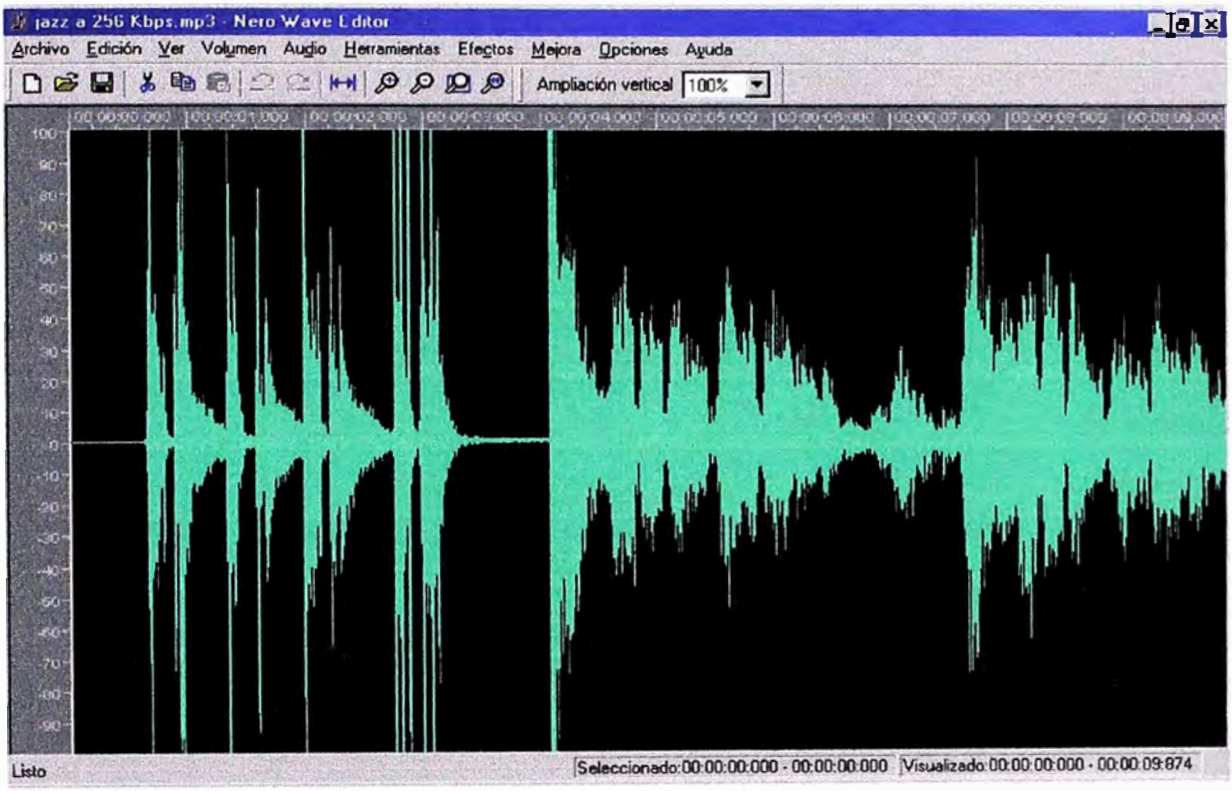


Figura PC16: **Jazz a 224 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

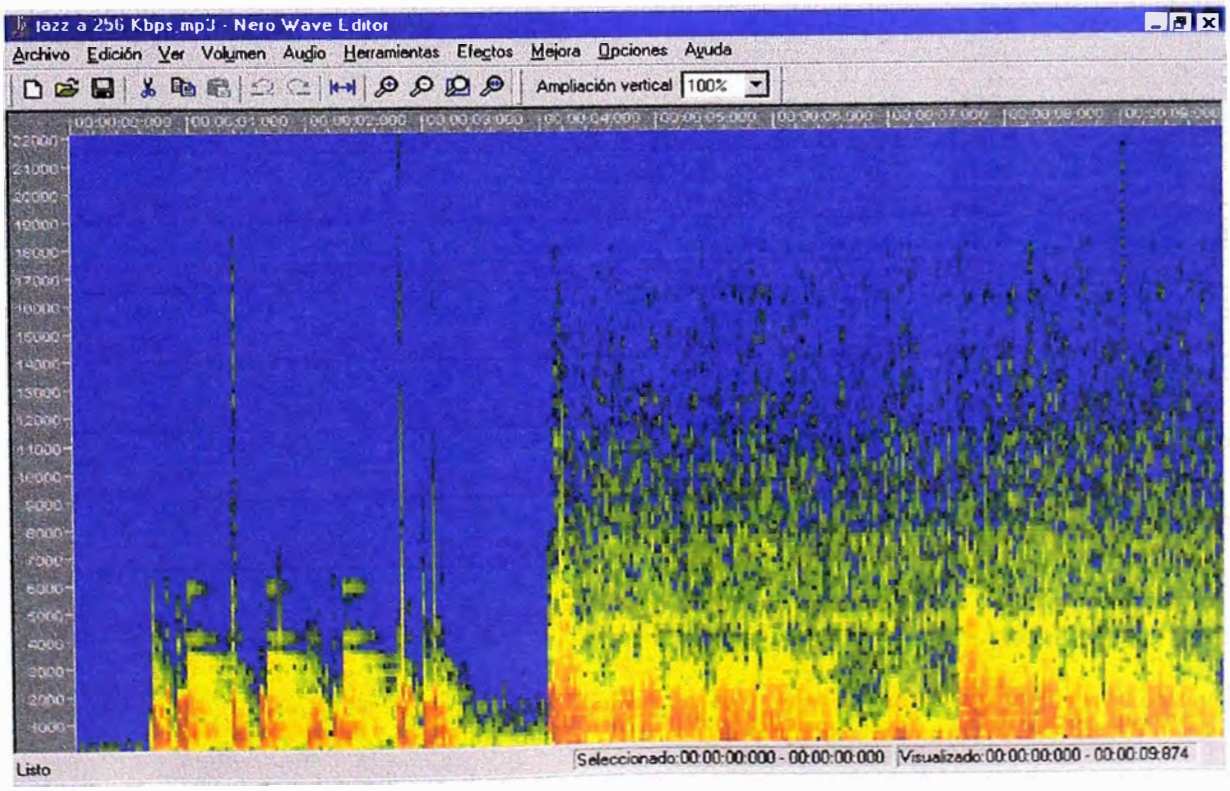
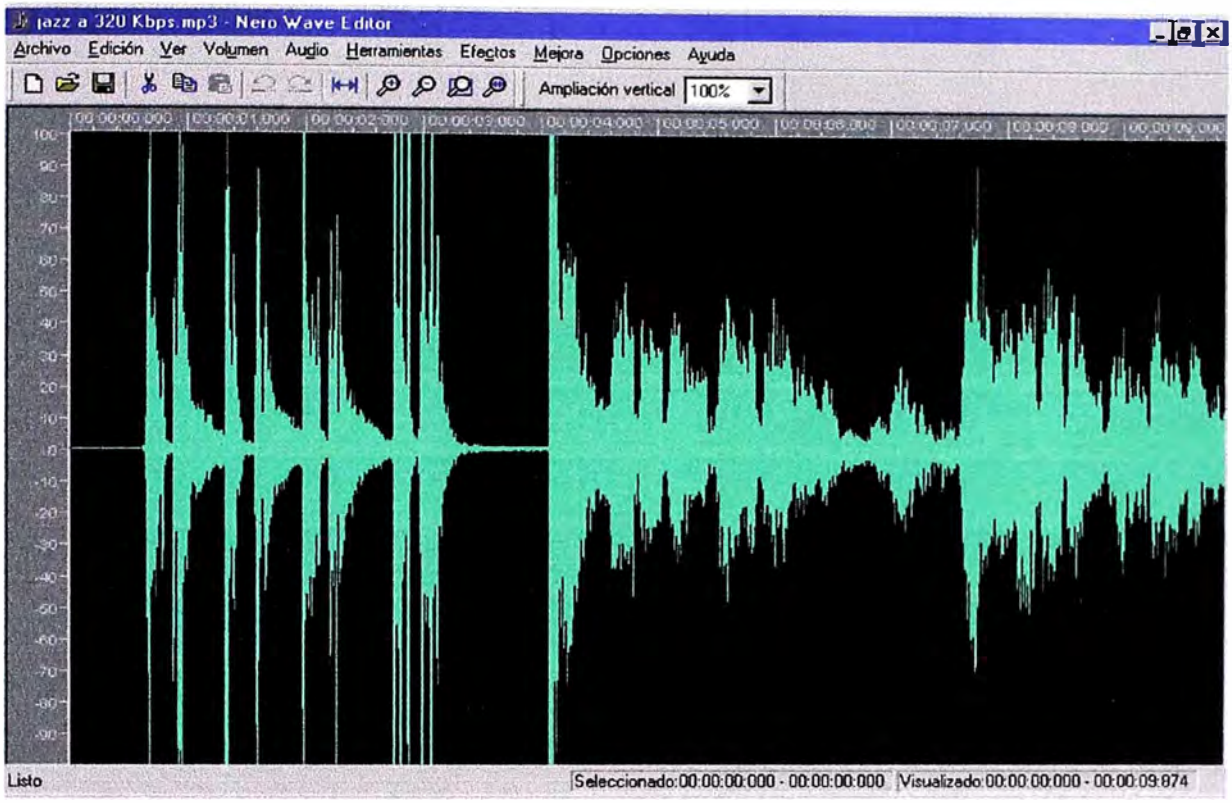


Figura PC17: **Jazz a 256 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

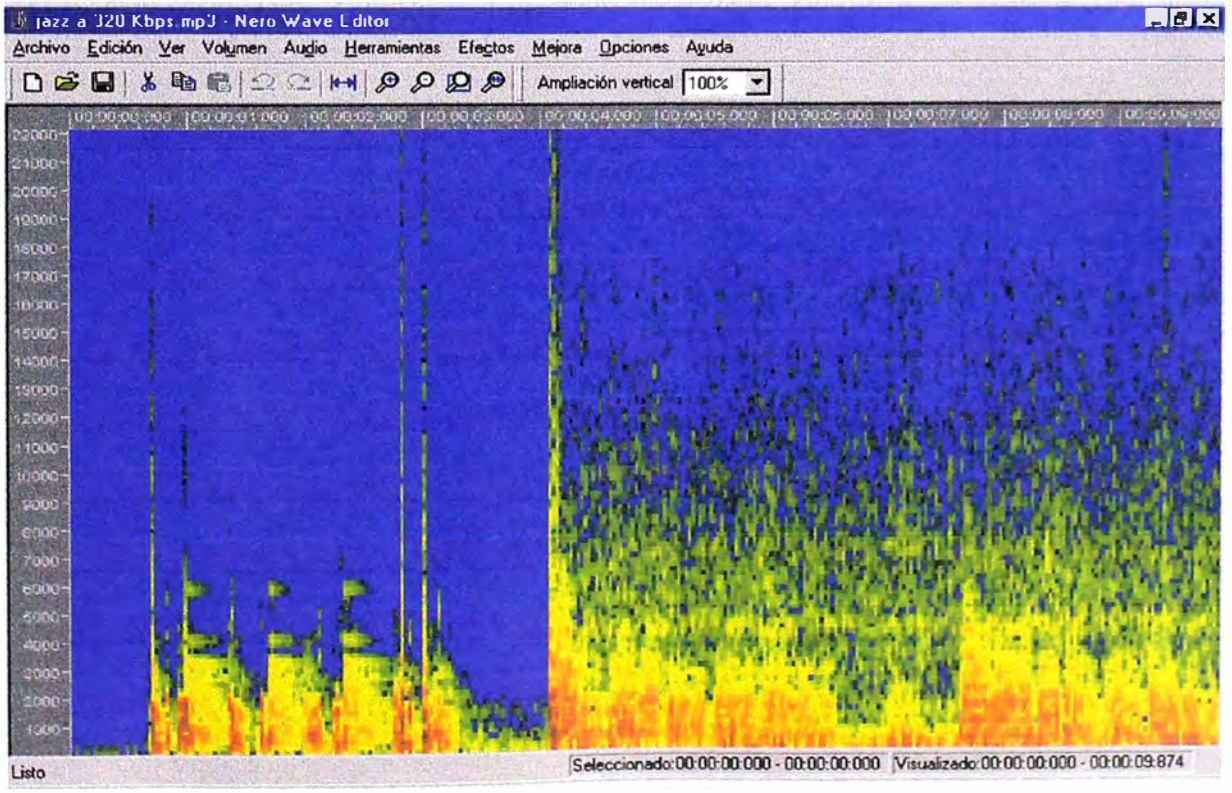
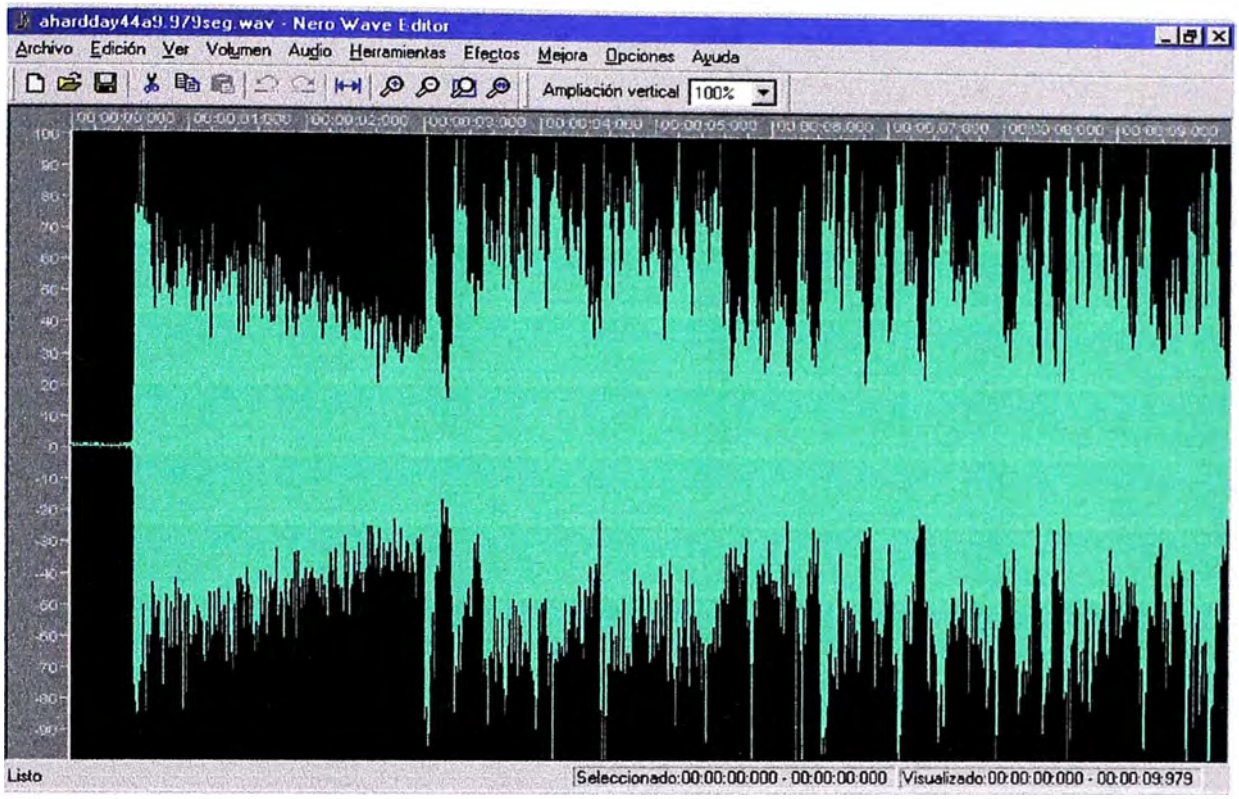


Figura PC18: **Jazz a 320 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

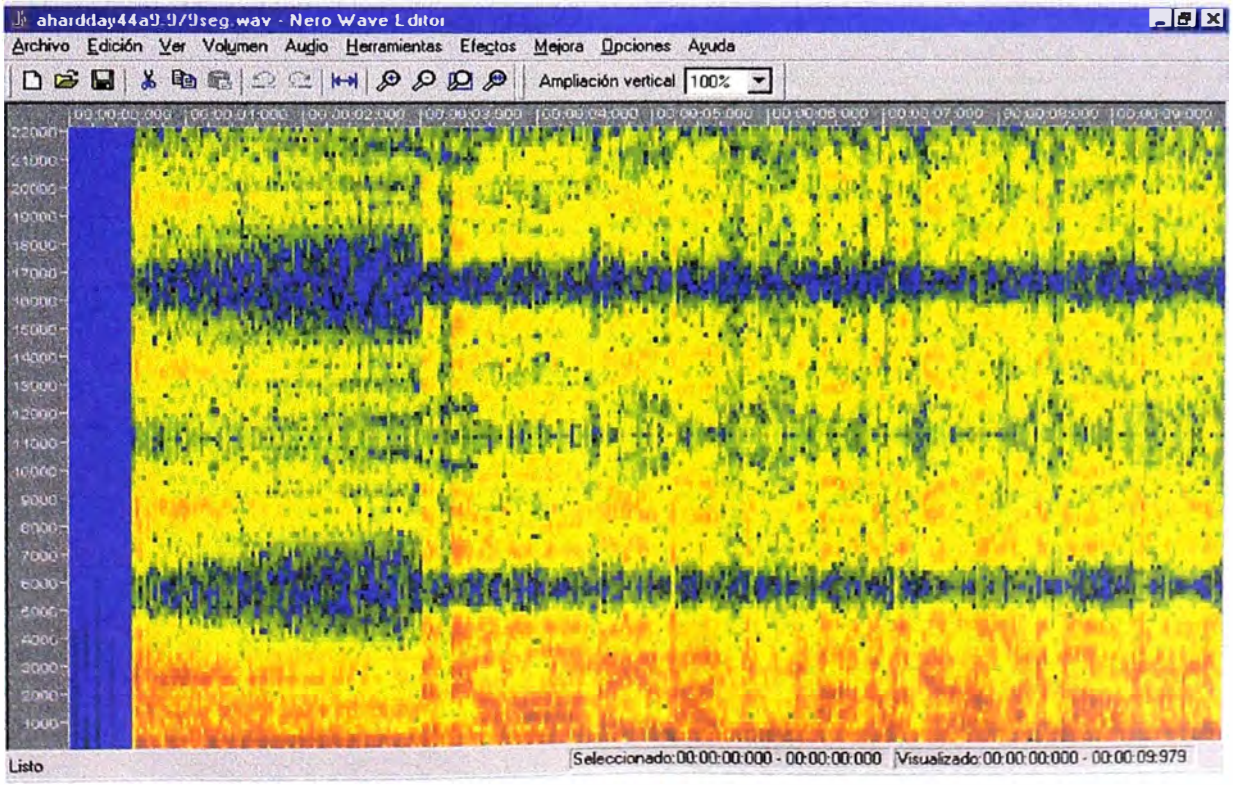
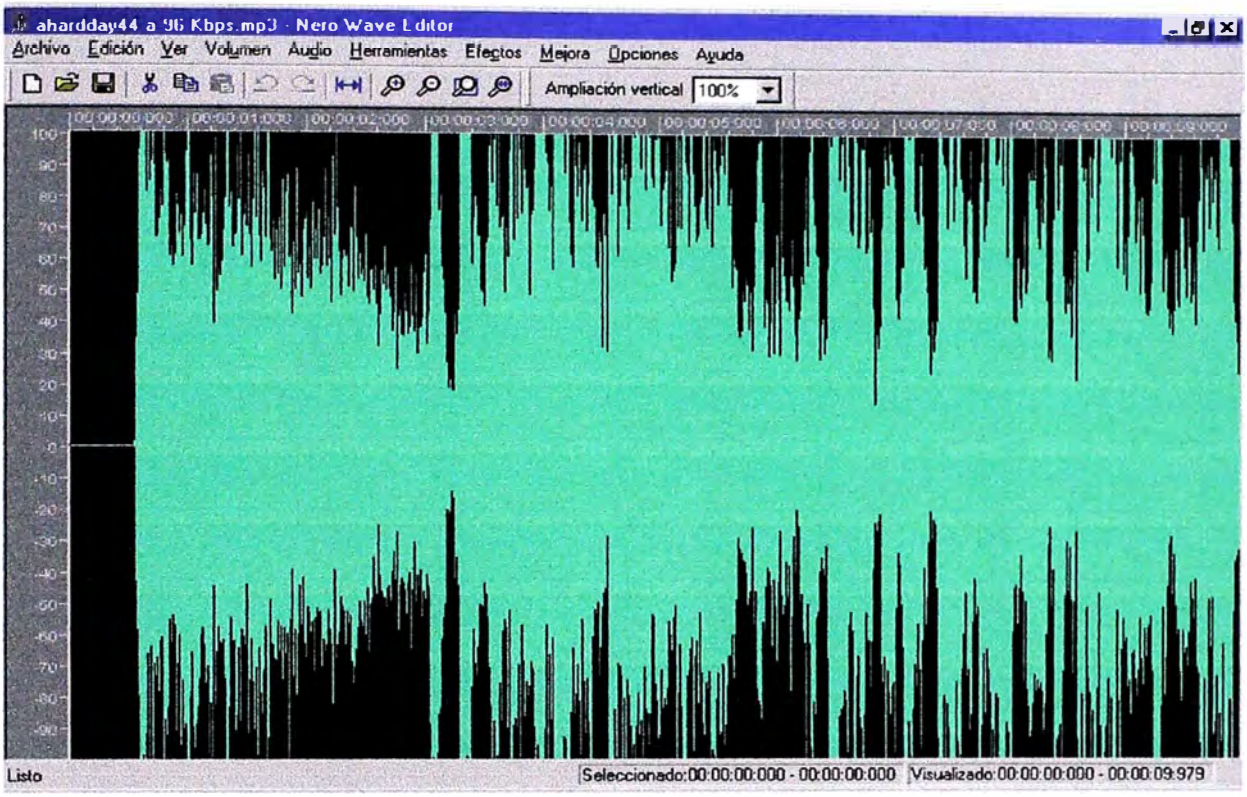


Figura PC19: **Ahardday44a9.979seg.wav**

Visualización de Onda



Visualización de Espectrograma

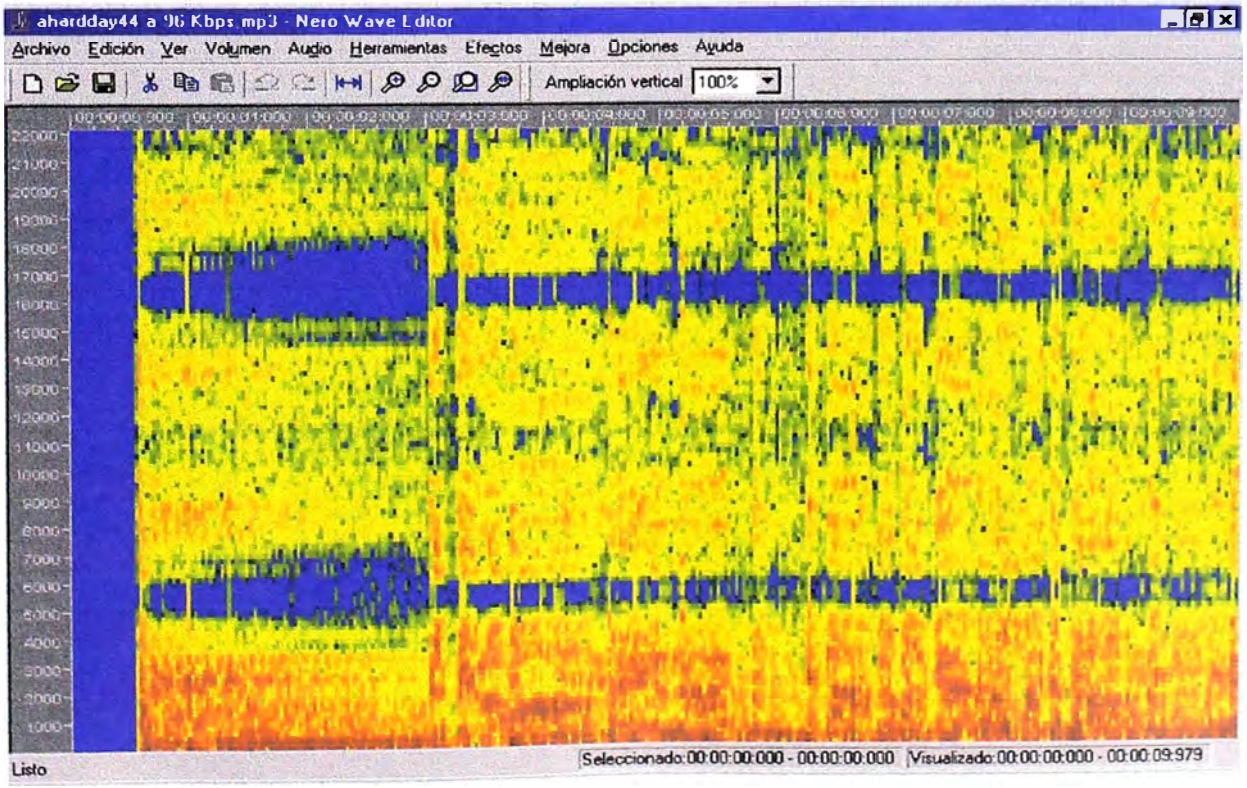
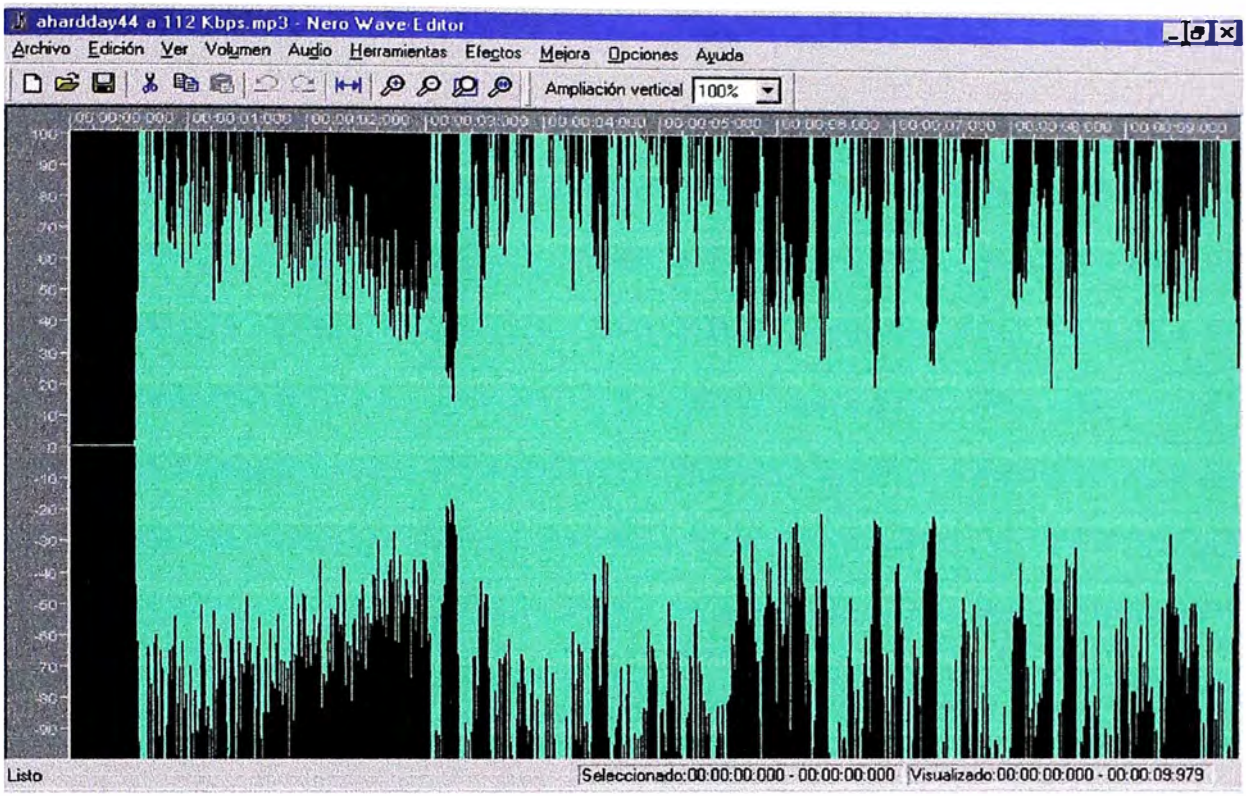


Figura PC20: **Ahardday44 a 96 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

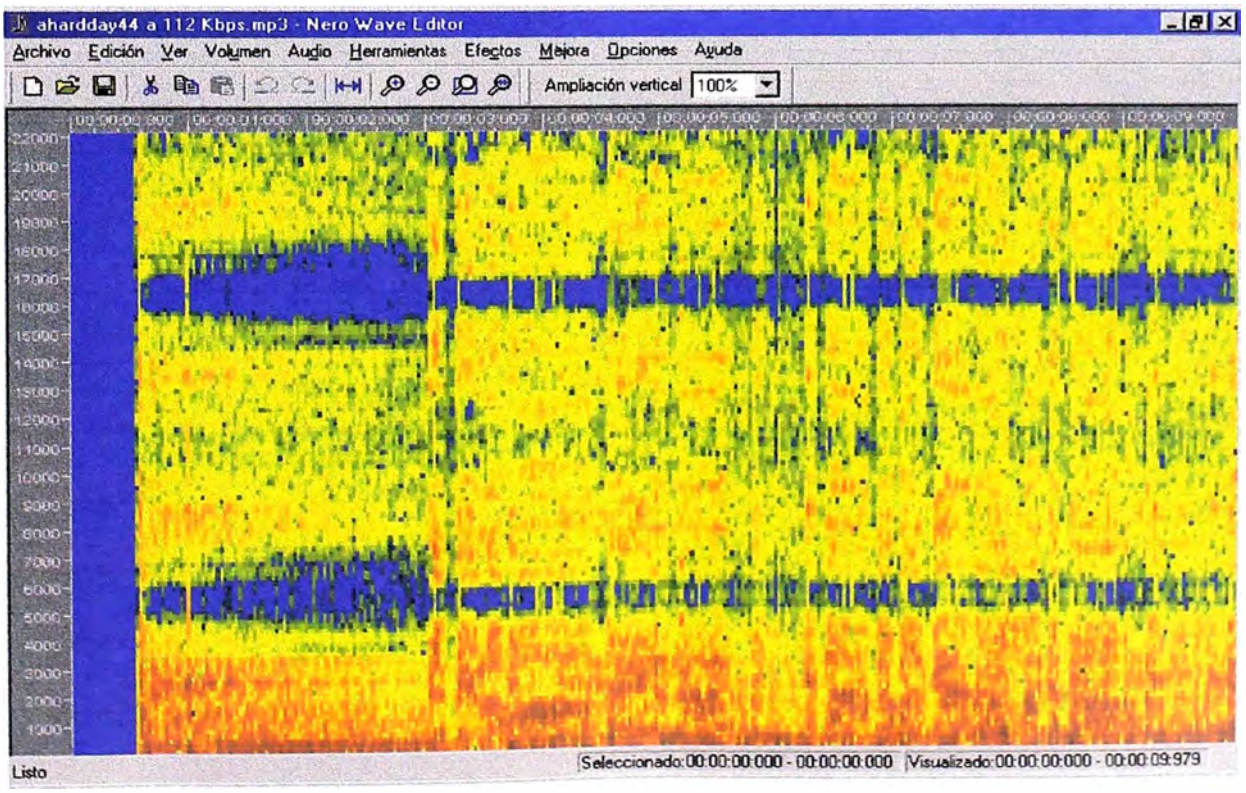
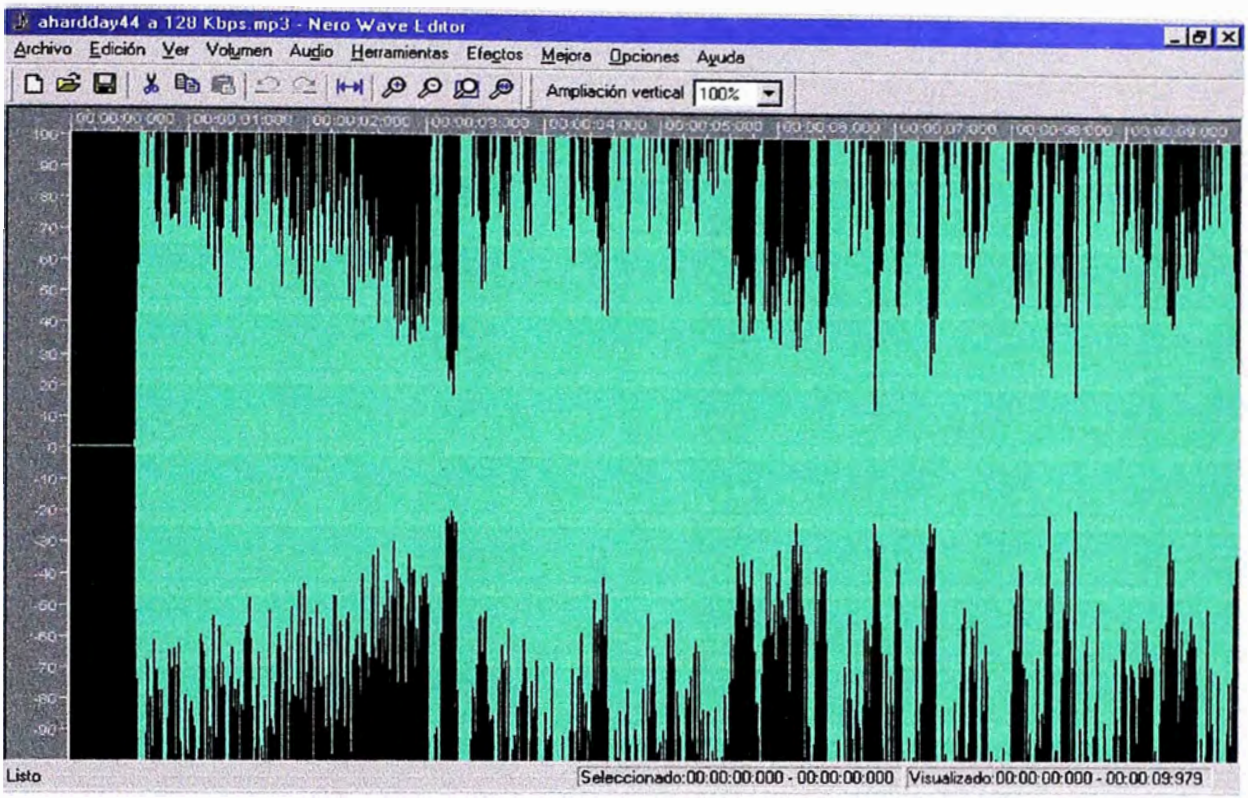


Figura PC21: **Ahardday44 a 112 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

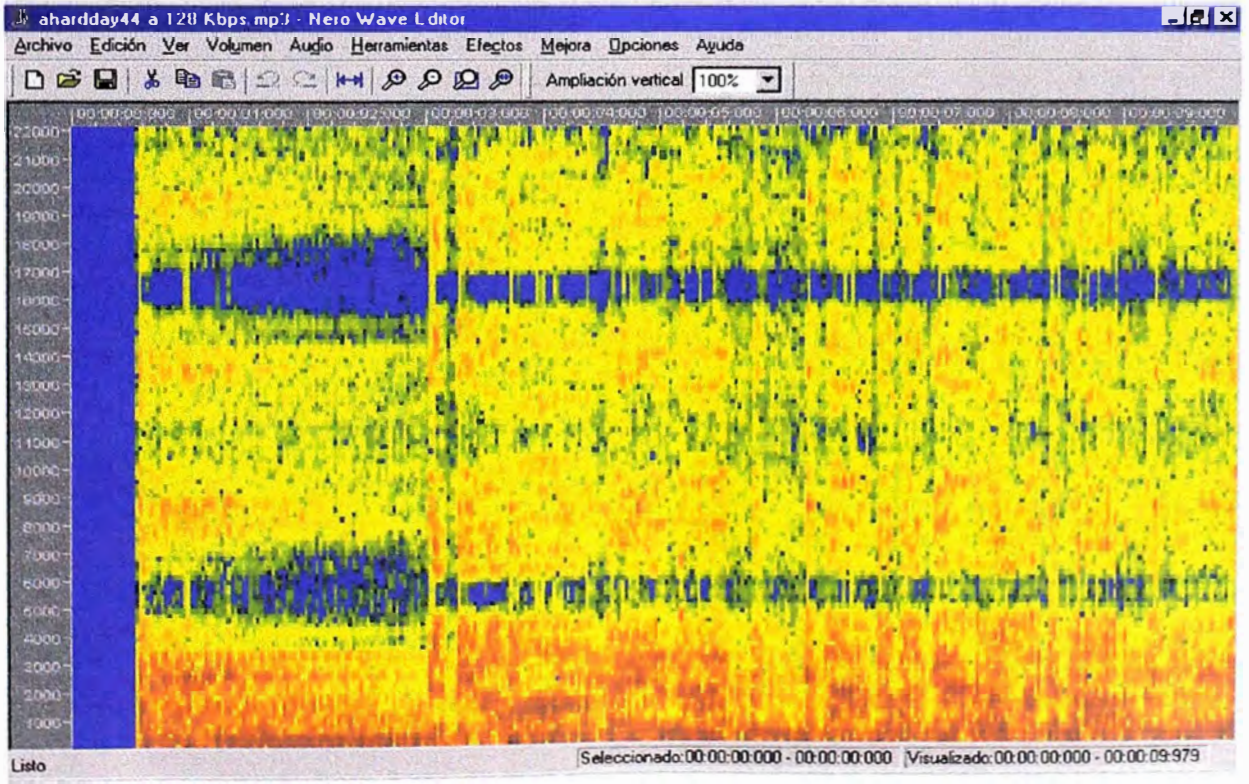
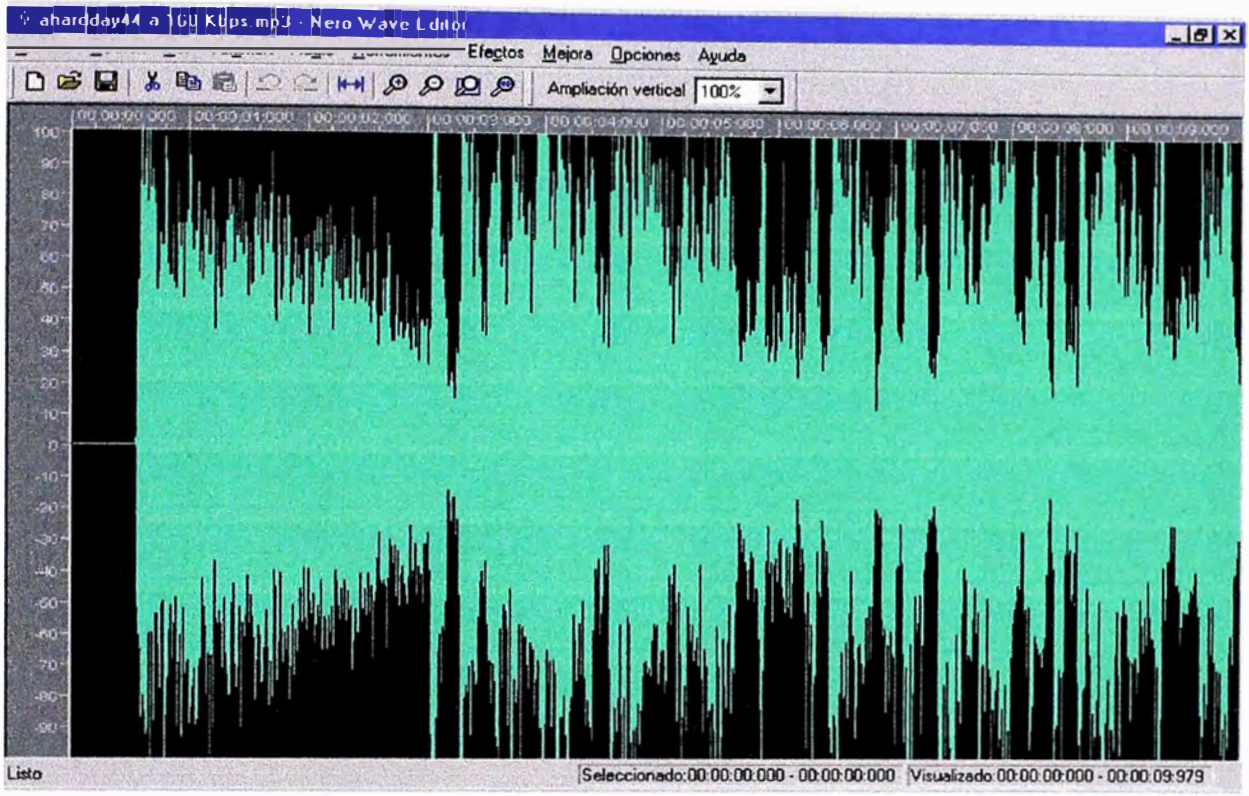


Figura PC22: **Ahardday44 a 128 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

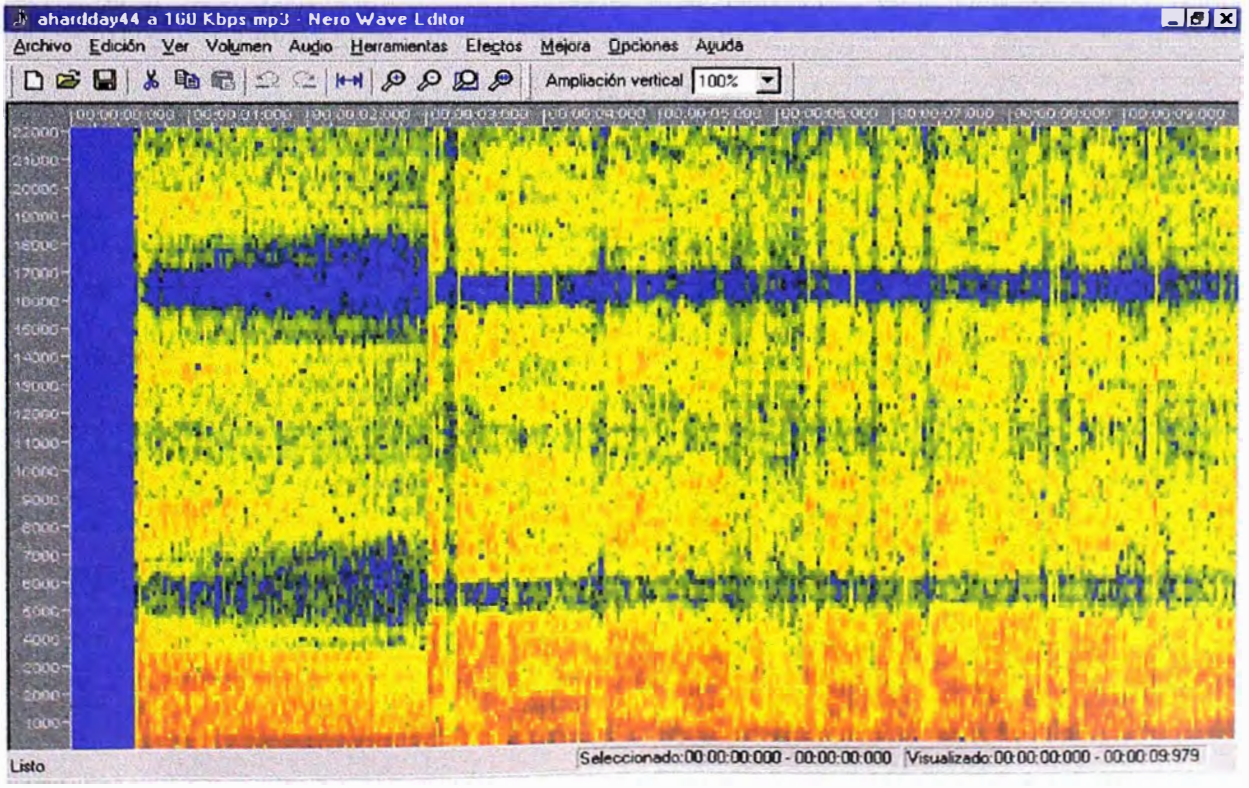
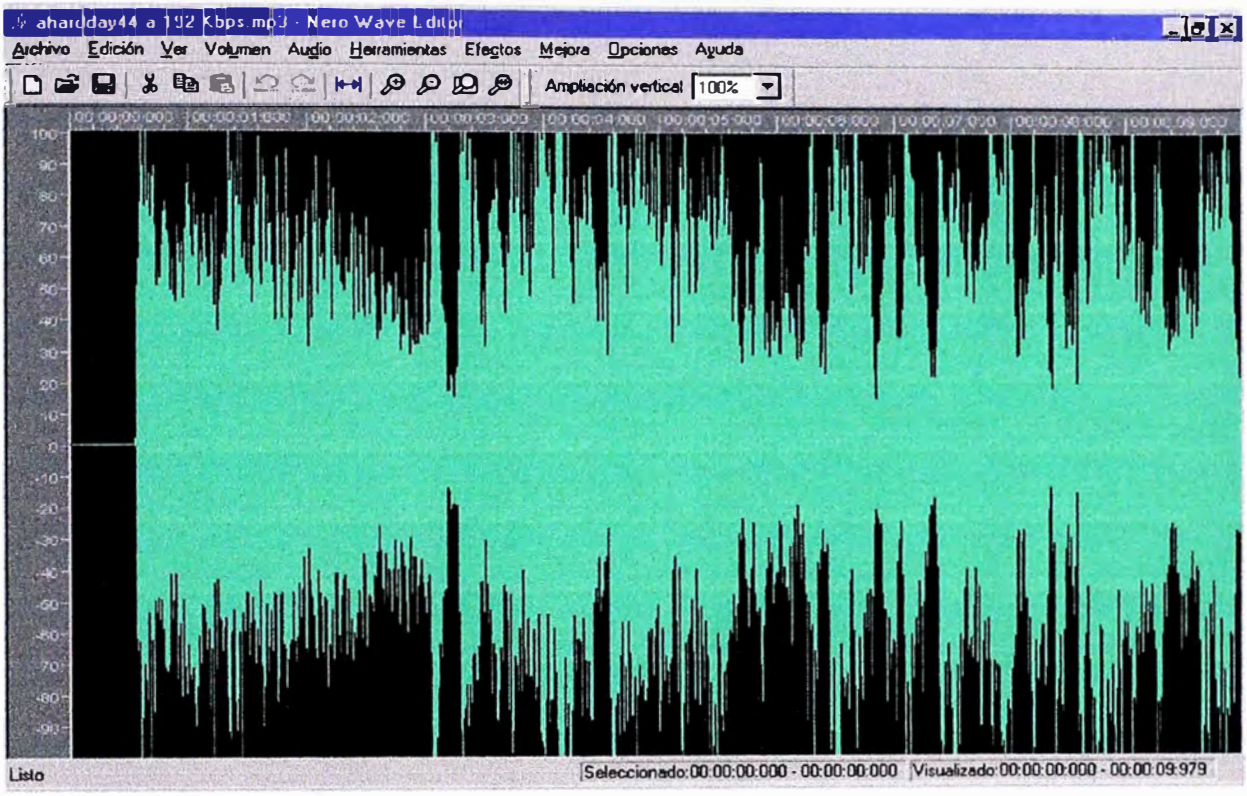


Figura PC23: **Ahardday44 a 160 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

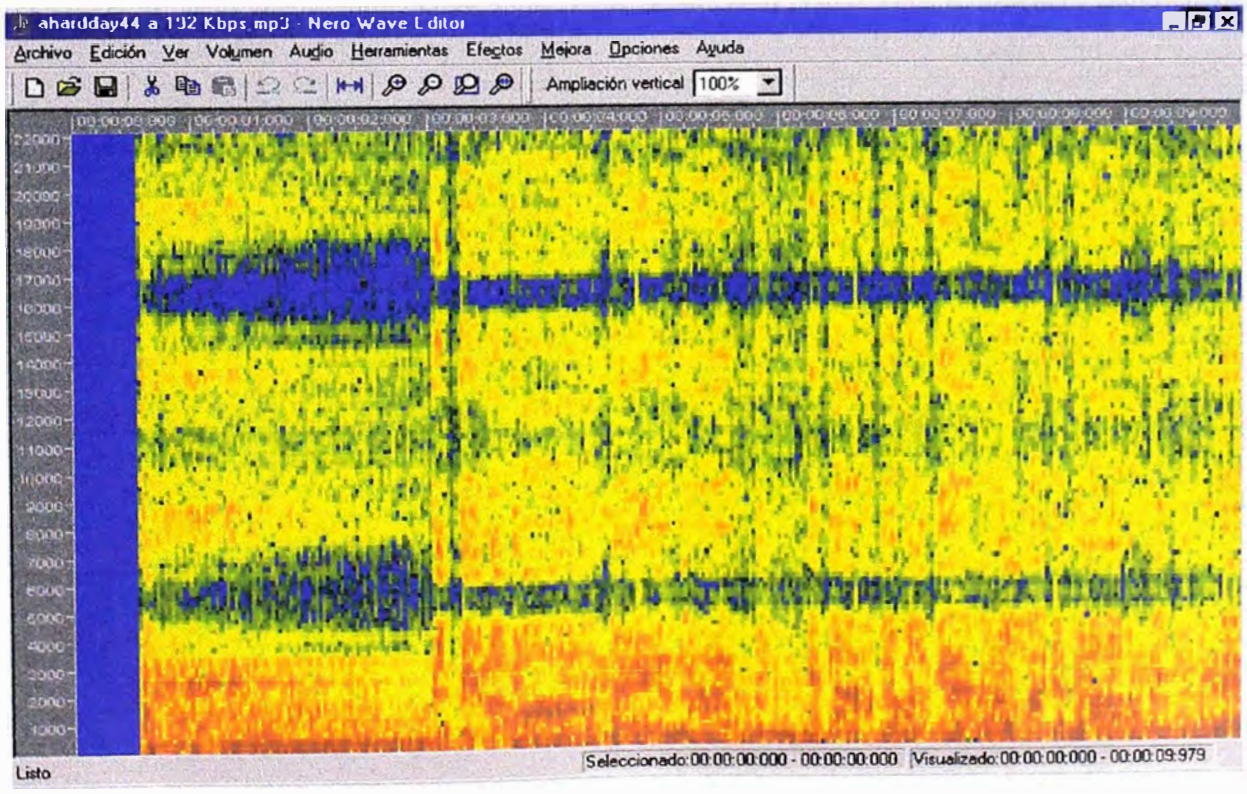
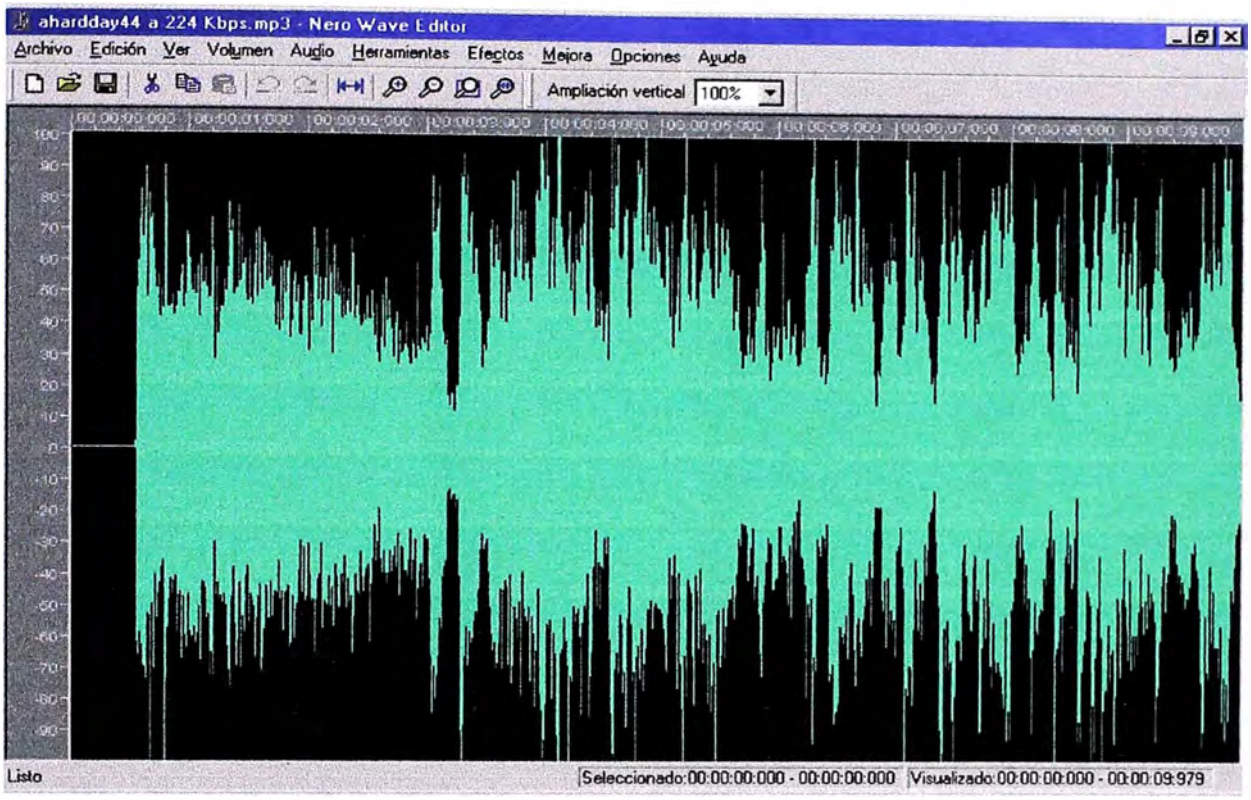


Figura PC24: **Ahardday44 a 192 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

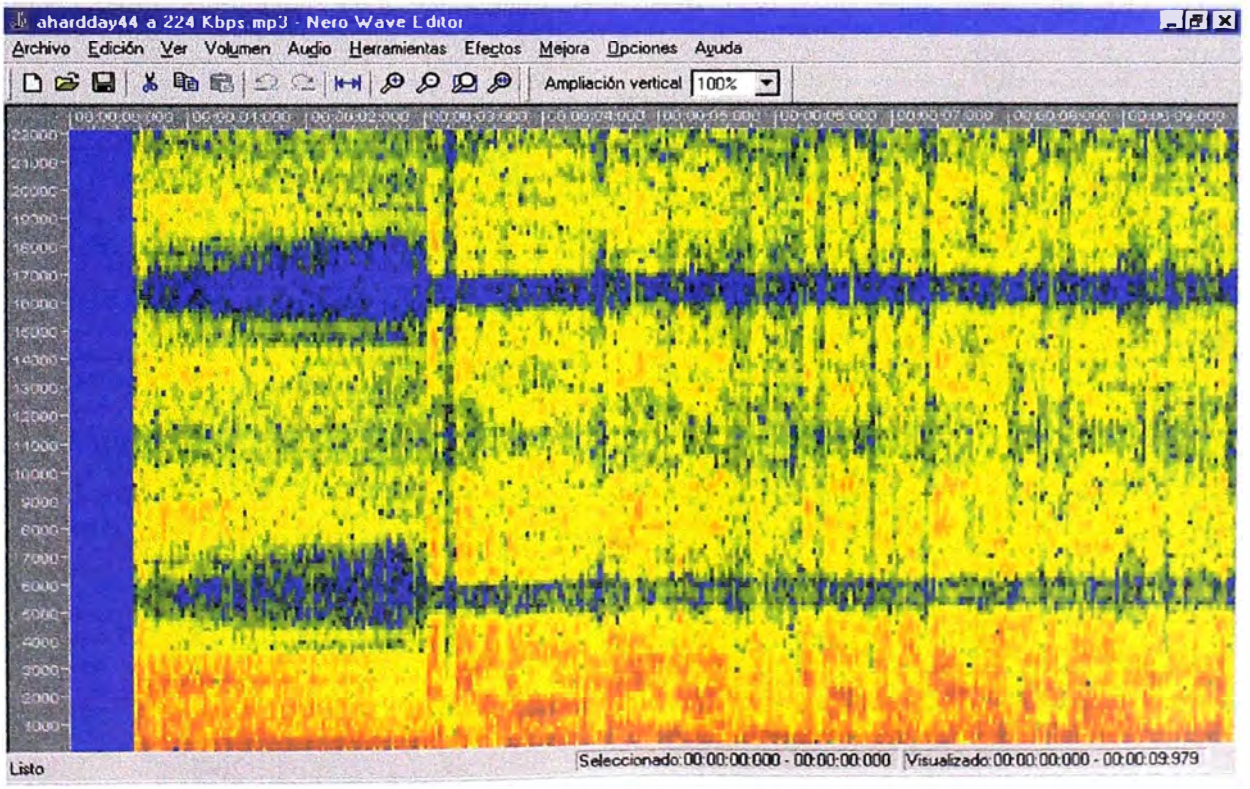
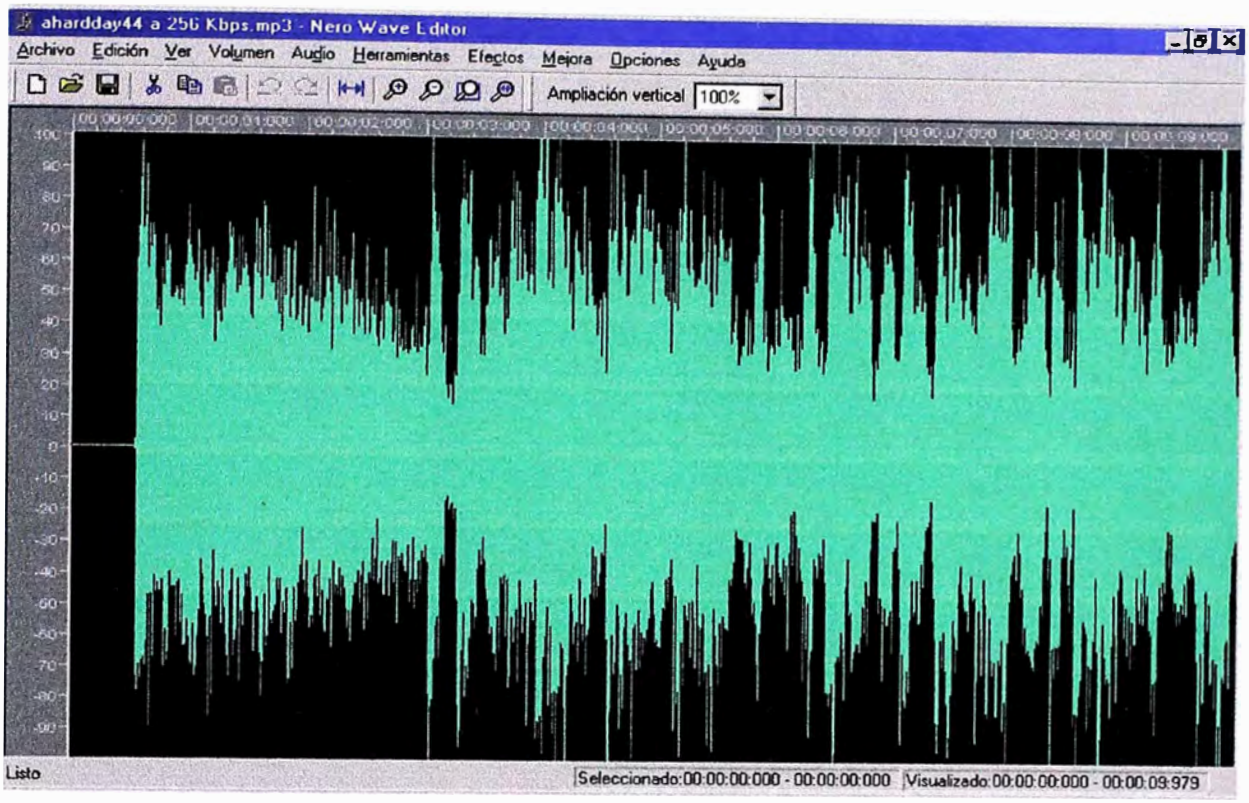


Figura PC25: **Ahardday44 a 224 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

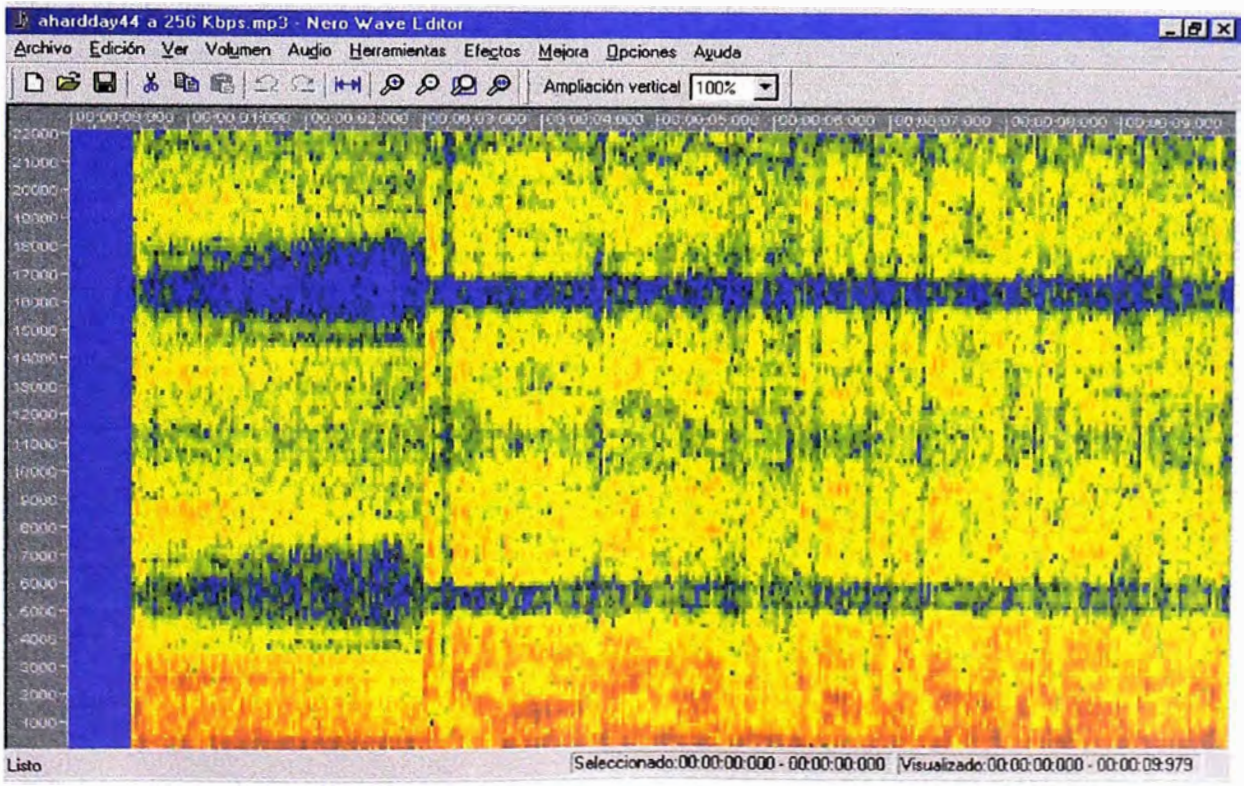
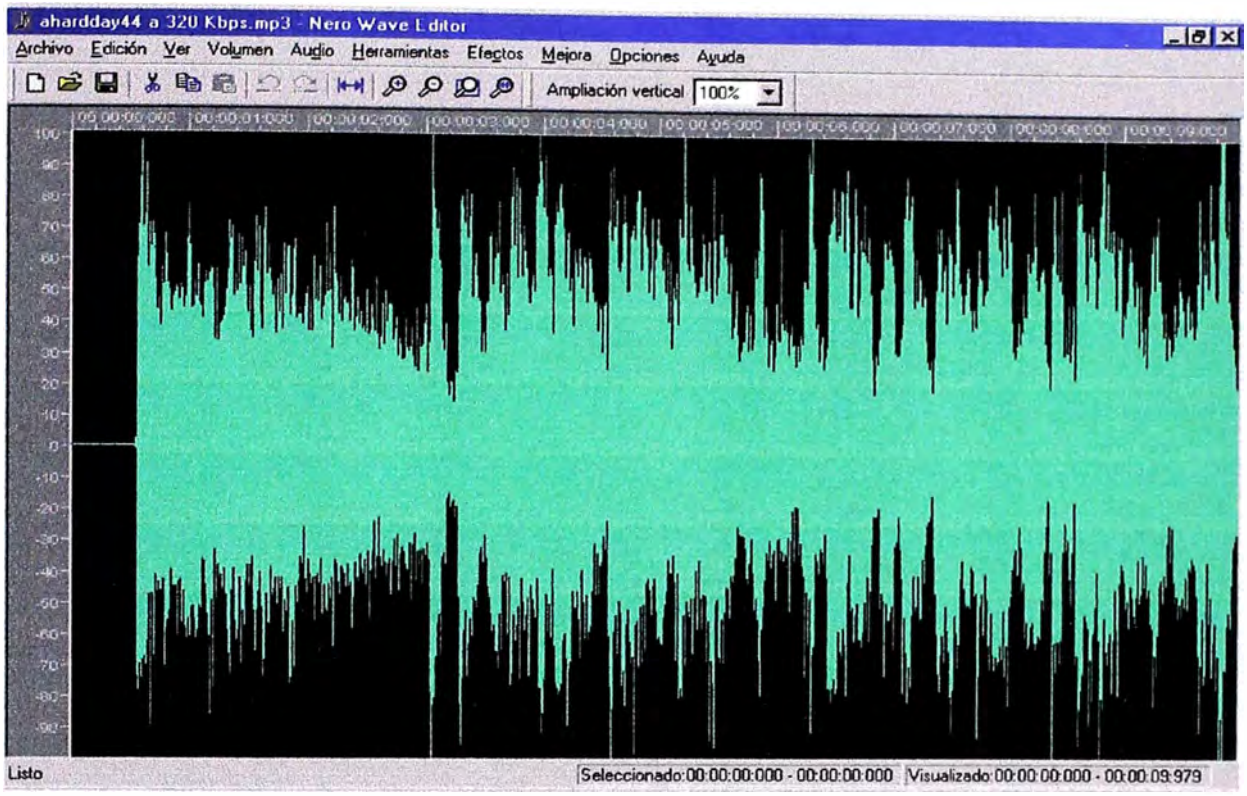


Figura PC26: **Ahardday44 a 256 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

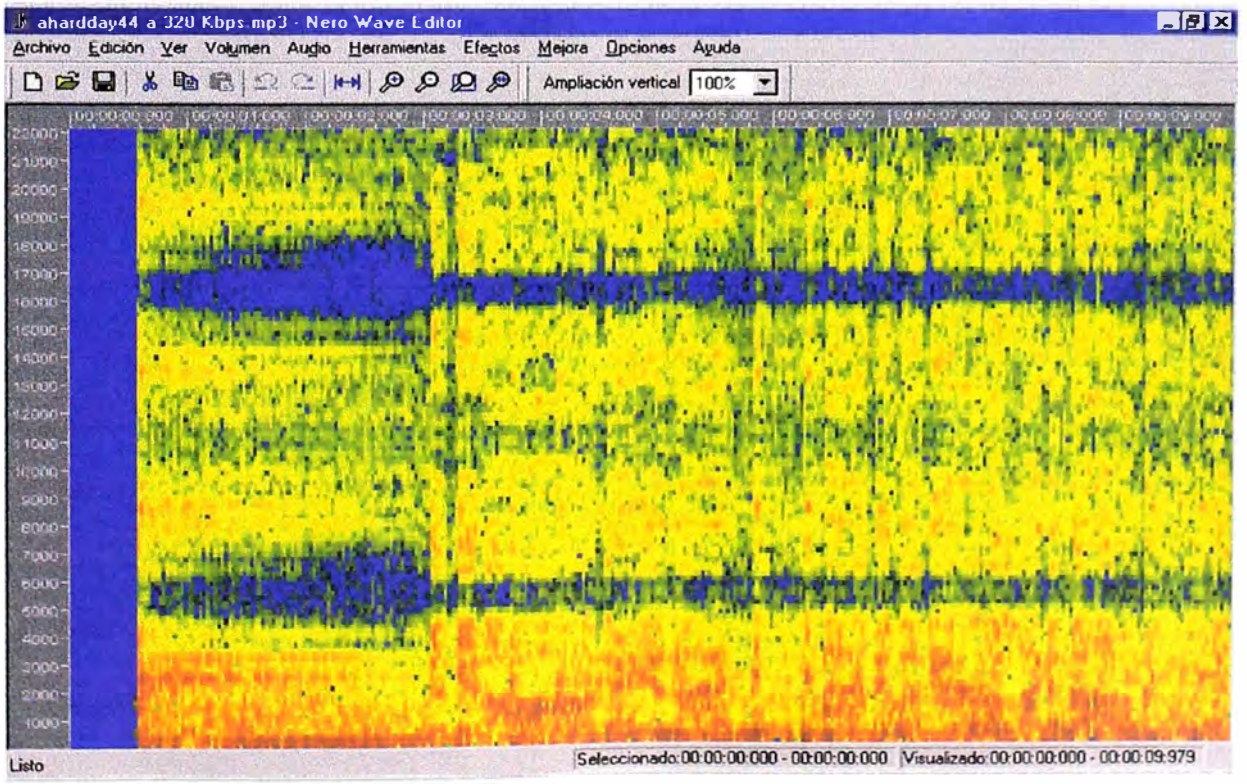
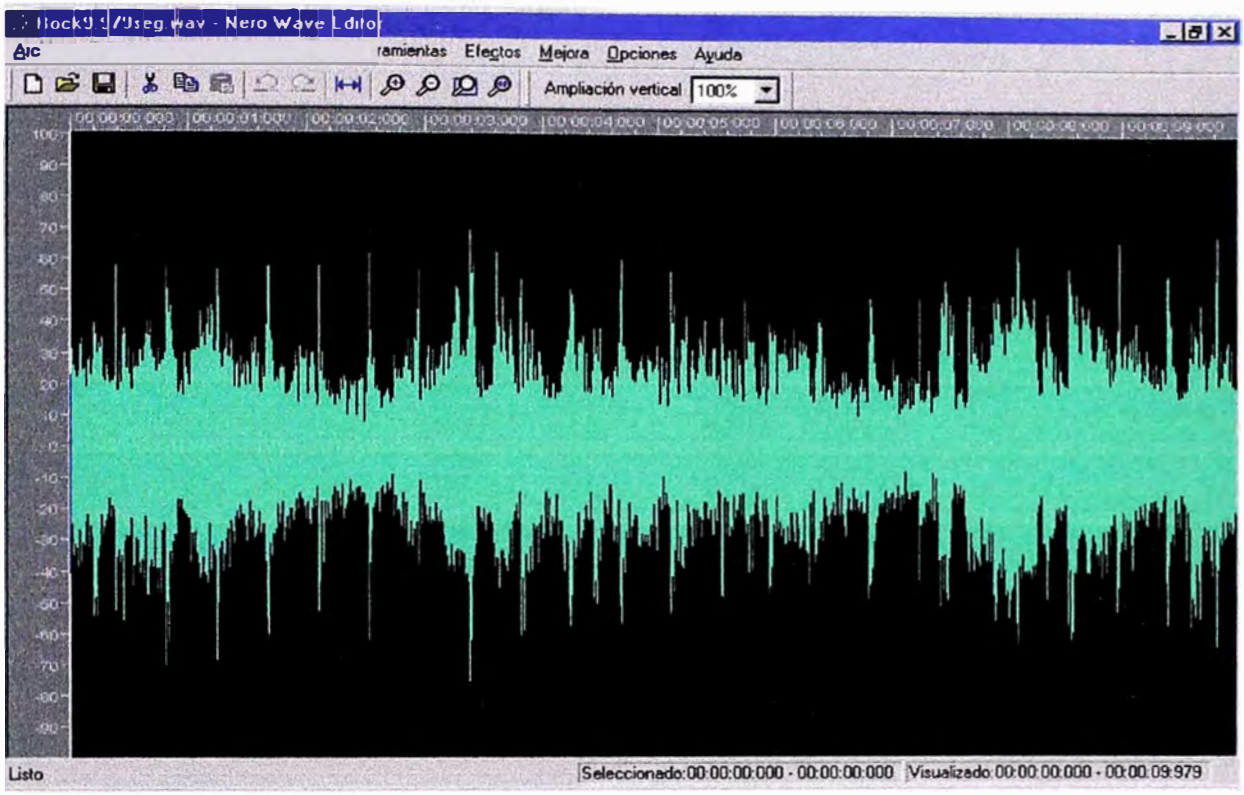


Figura PC27: **Ahardday44 a 320 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

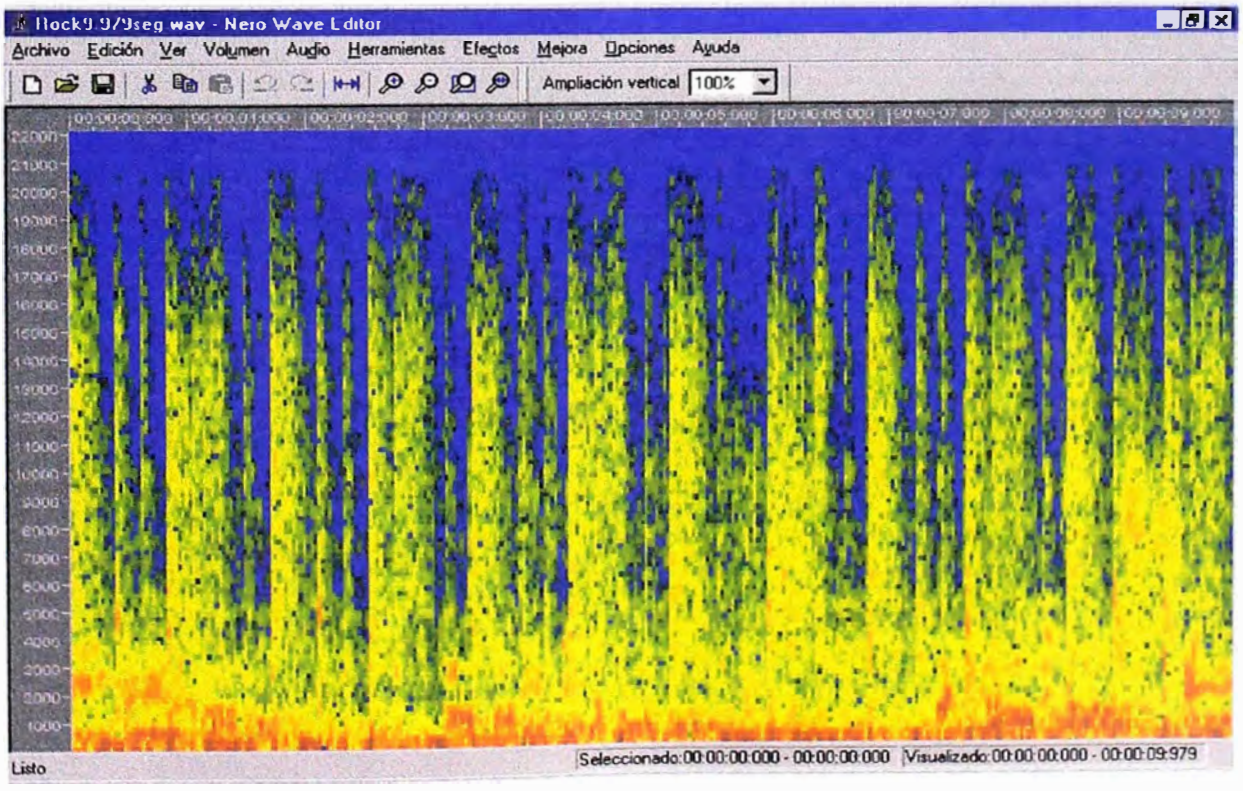
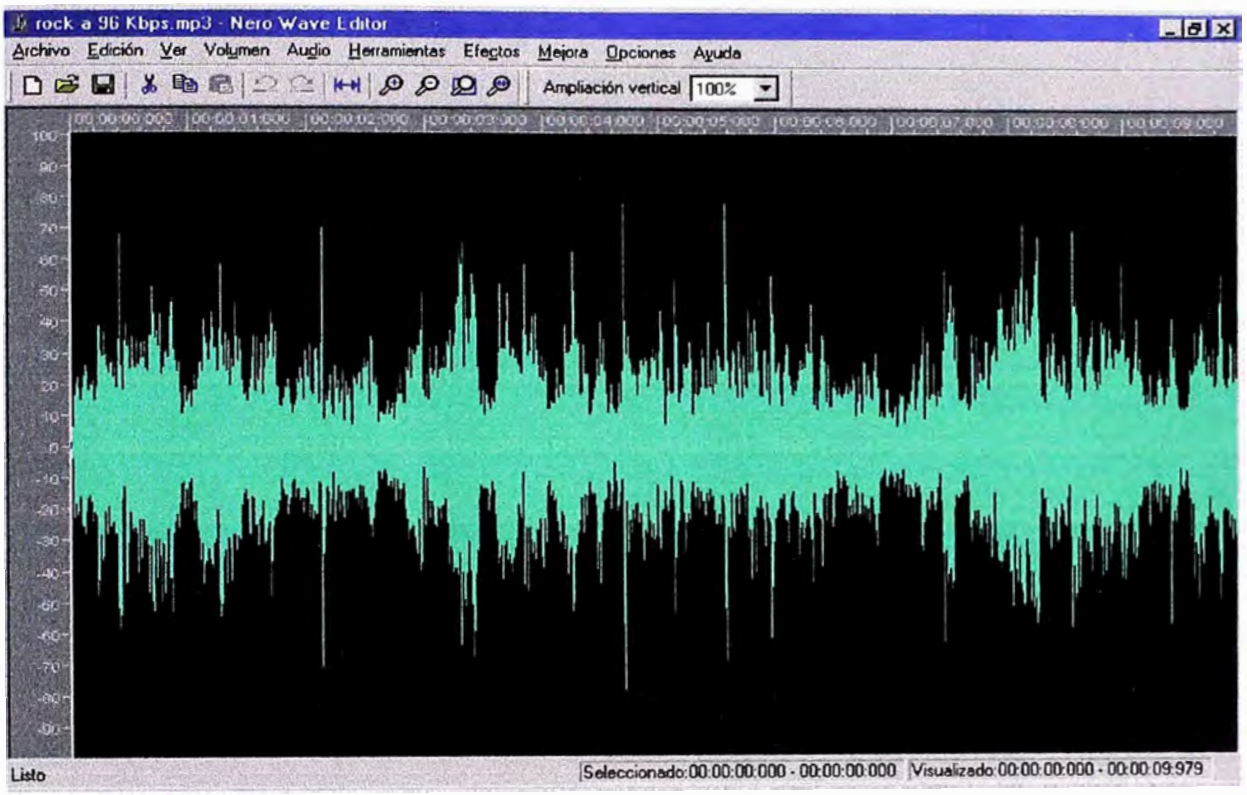


Figura PC28: **Rock9.979seg.wav**

Visualización de Onda



Visualización de Espectrograma

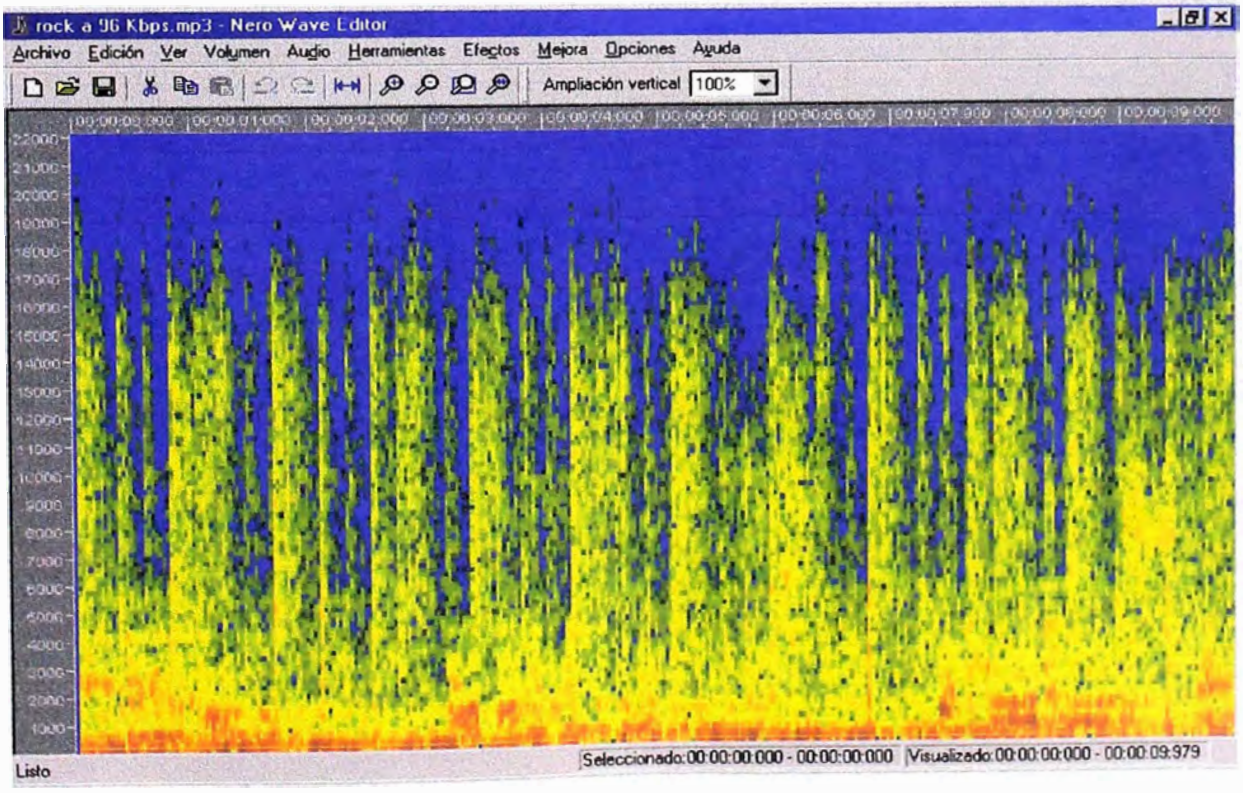
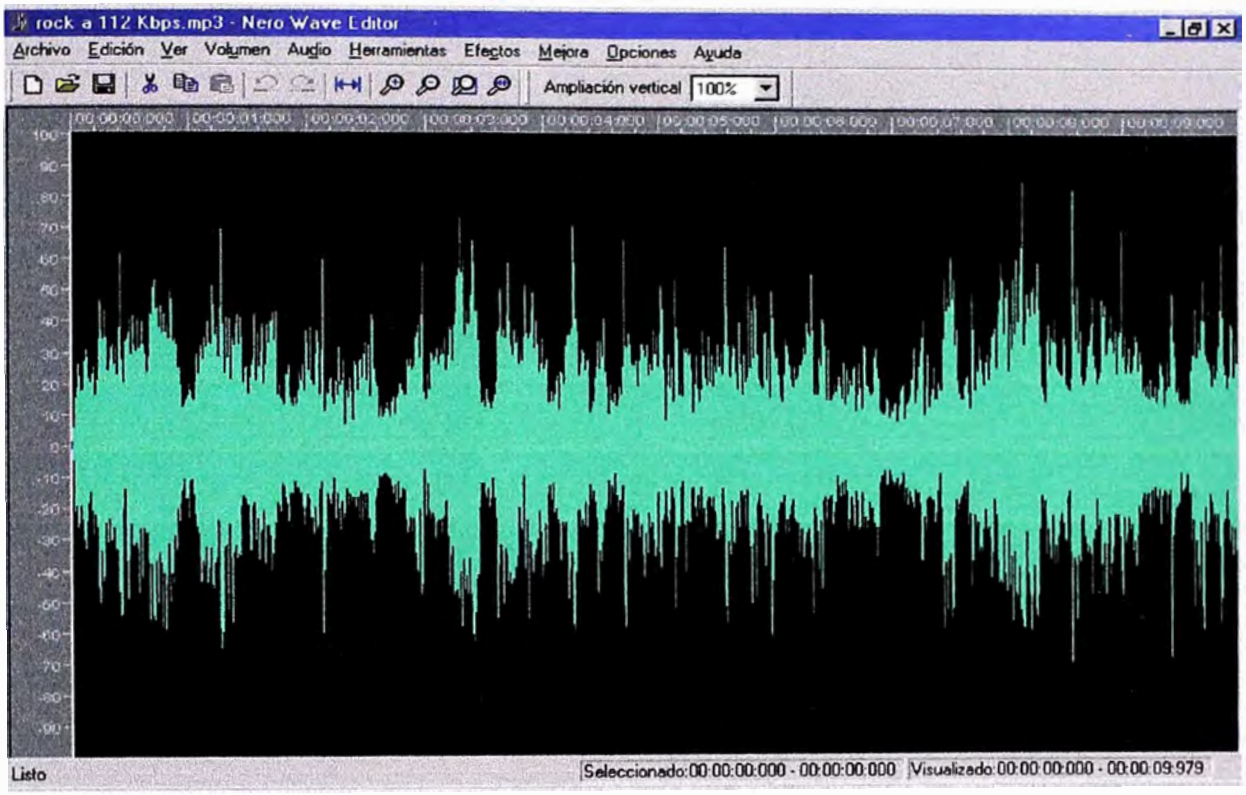


Figura PC29: **Rock a 96 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

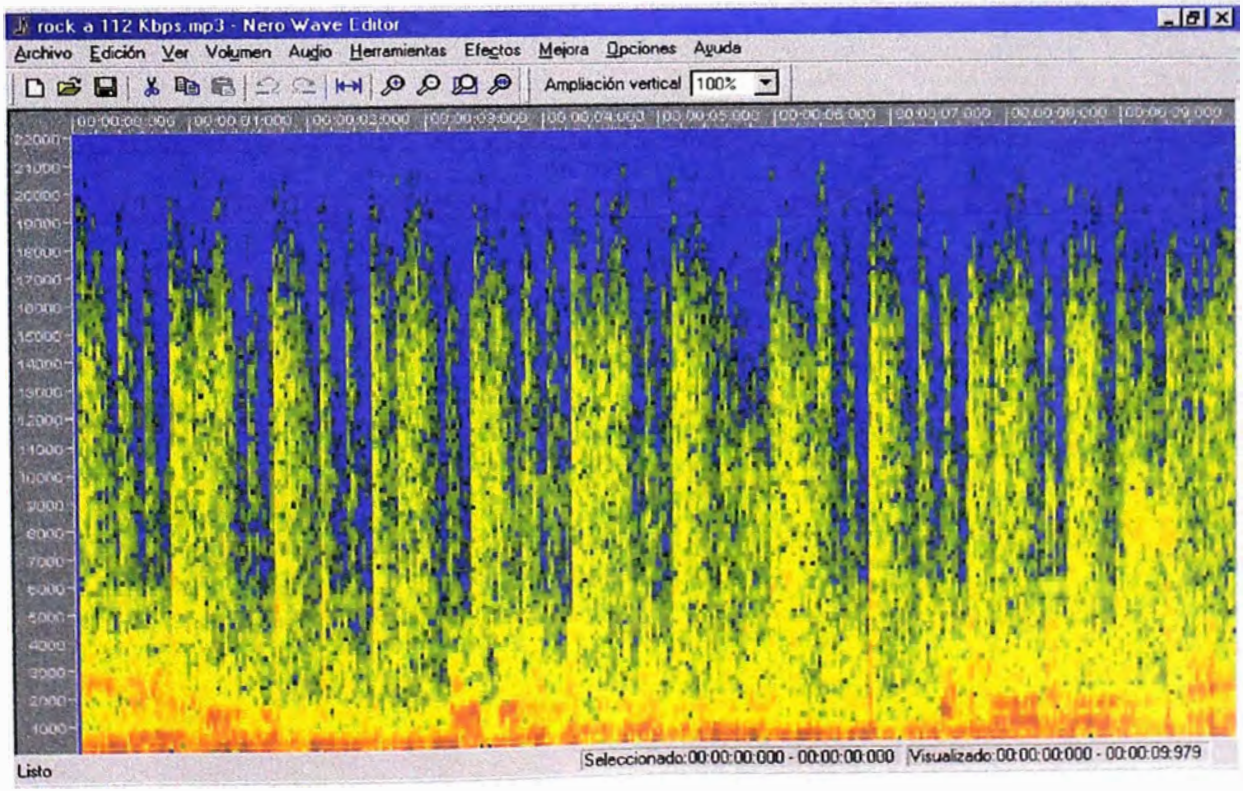
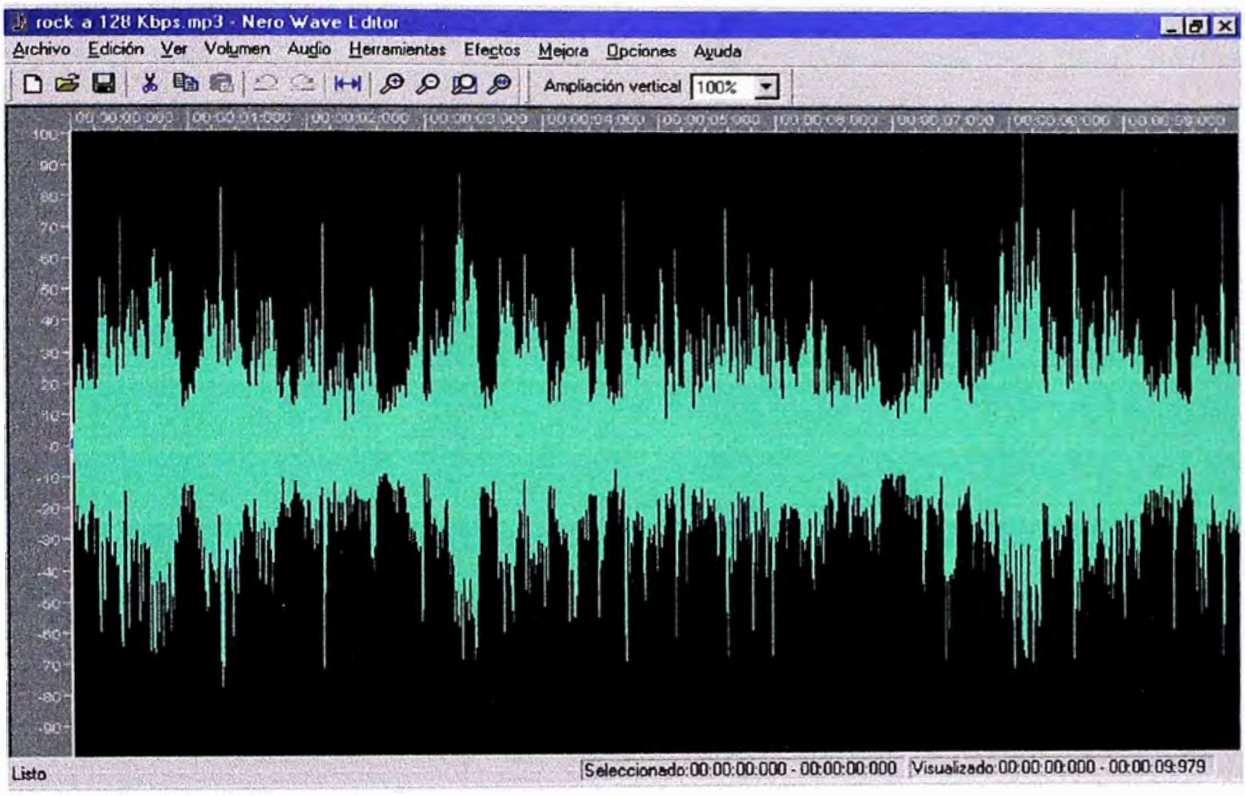


Figura PC30: **Rock a 112 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

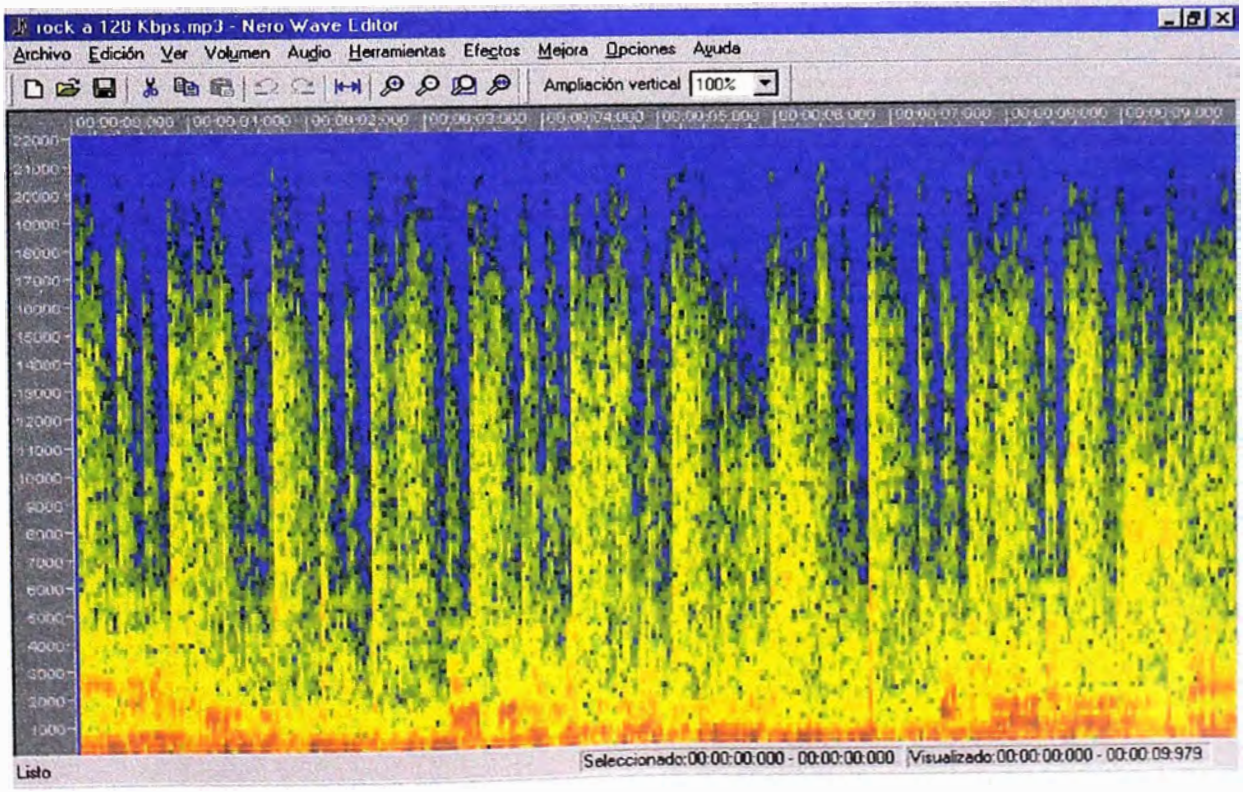
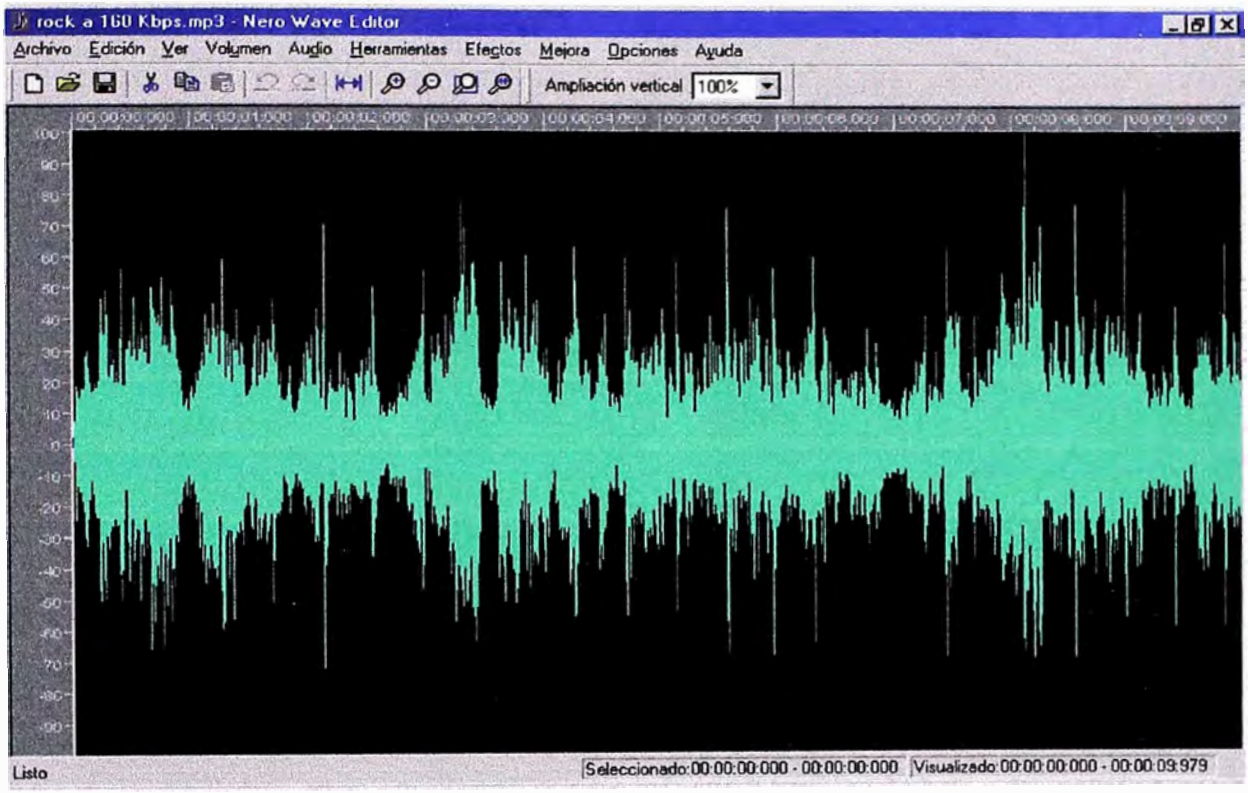


Figura PC31: **Rock a 128 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

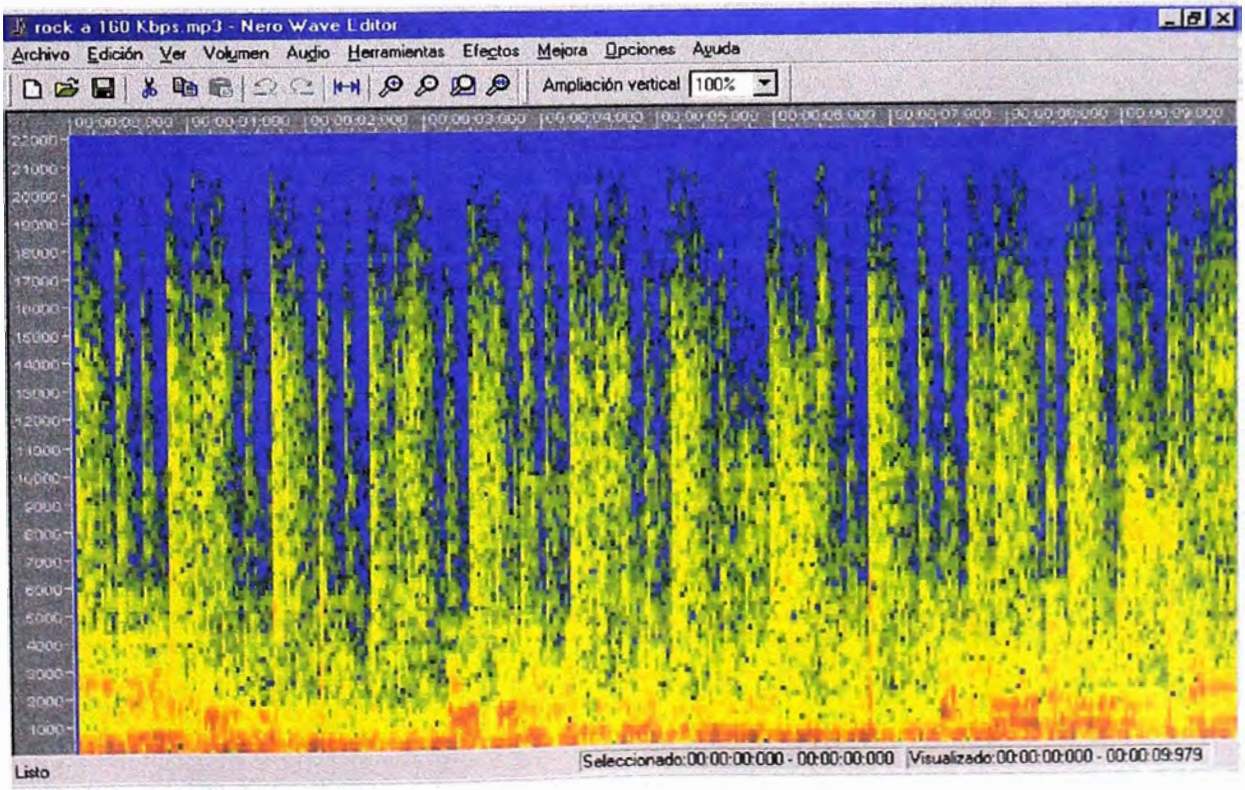
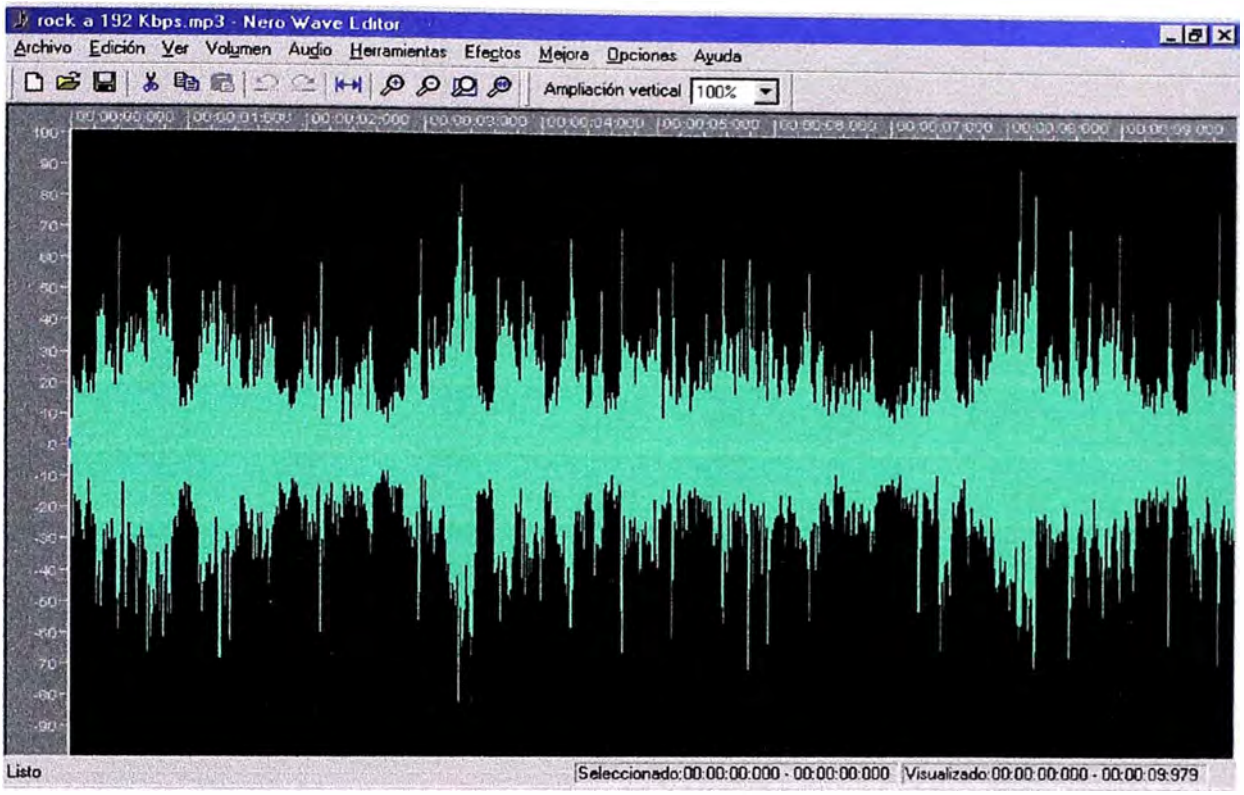


Figura PC32: **Rock a 160 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

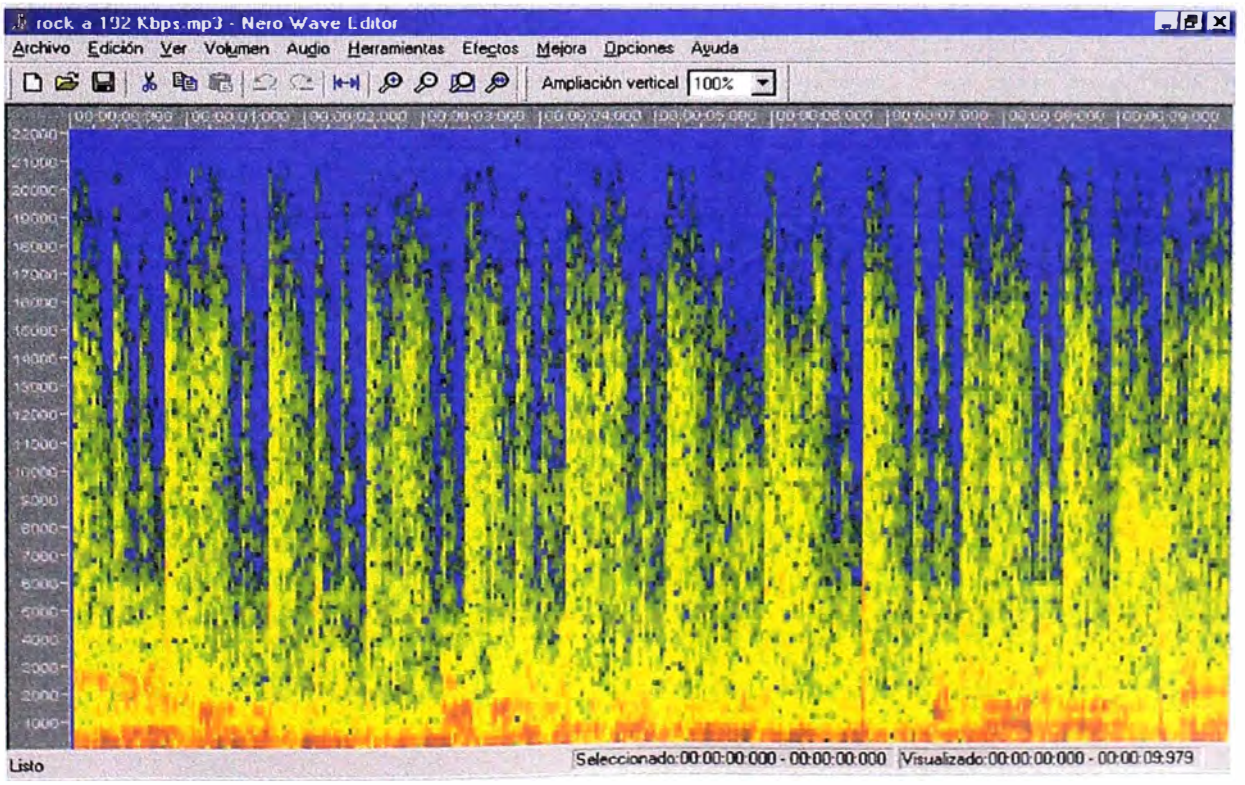
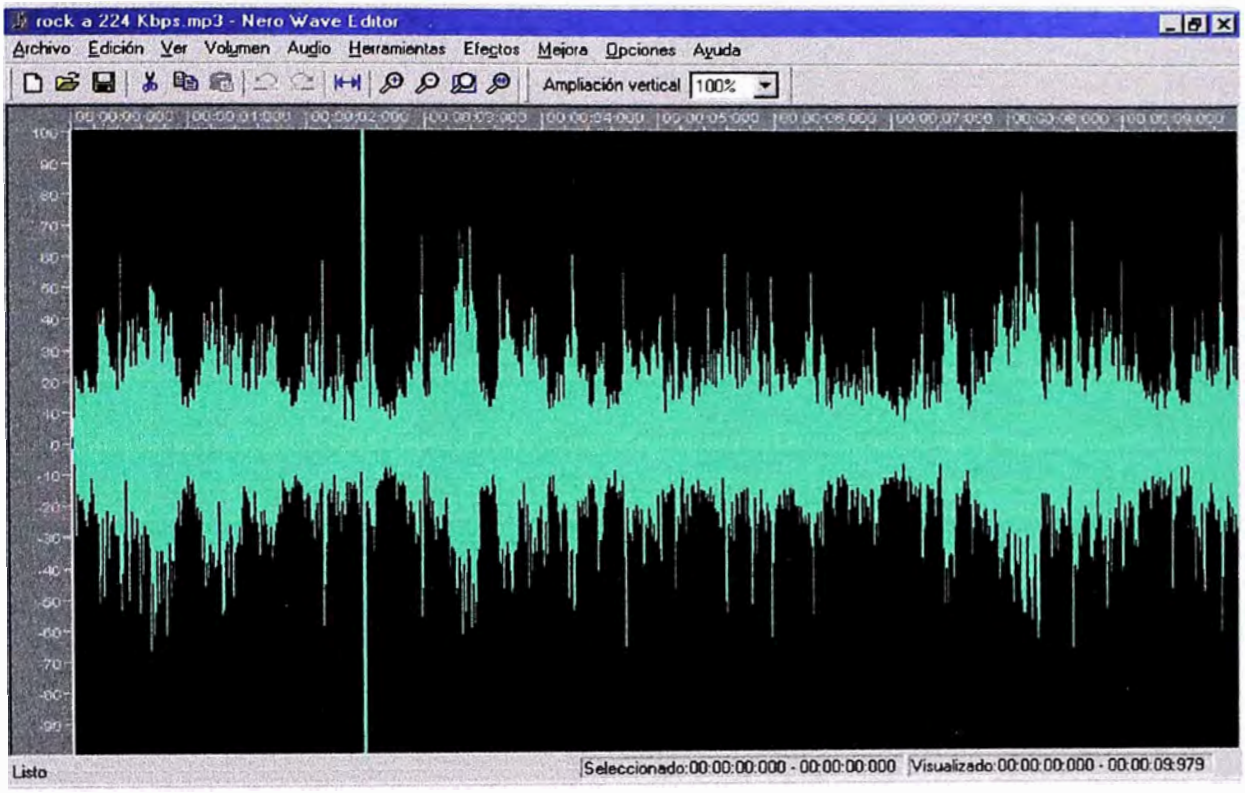


Figura PC33: **Rock a 192 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

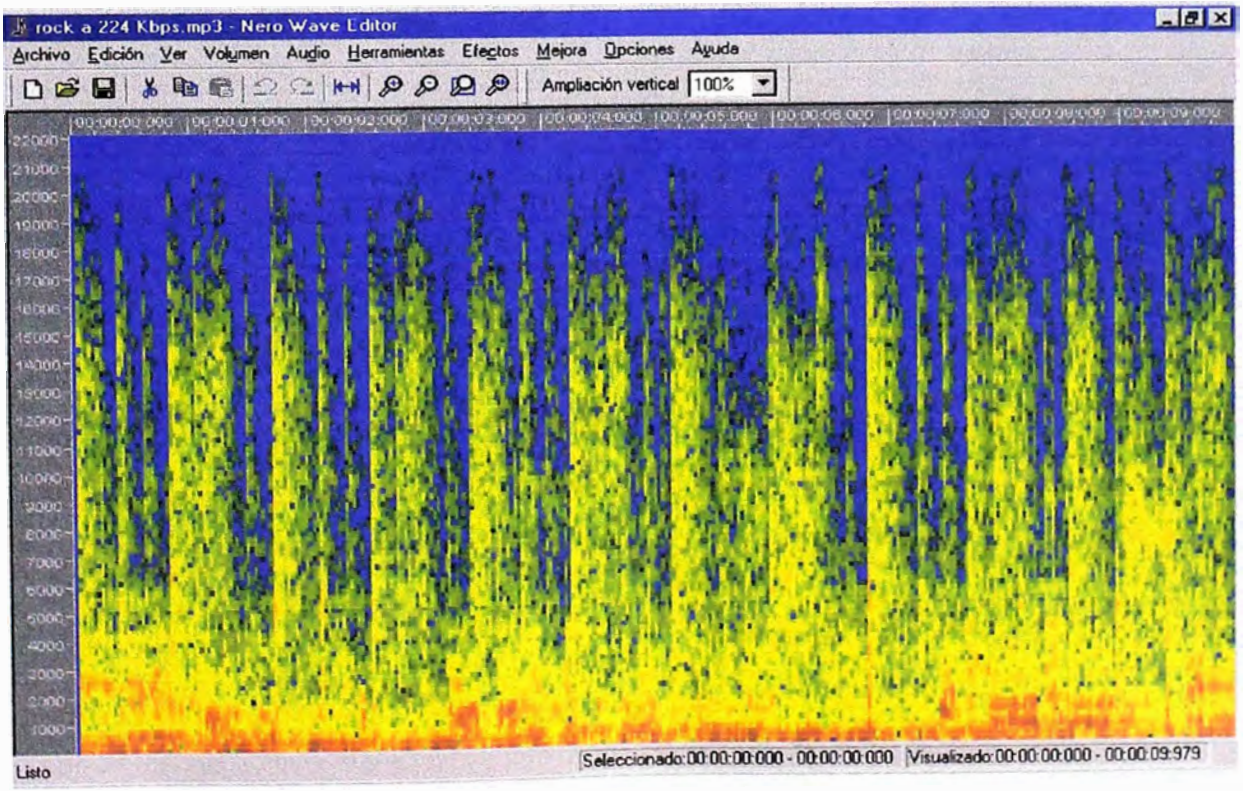
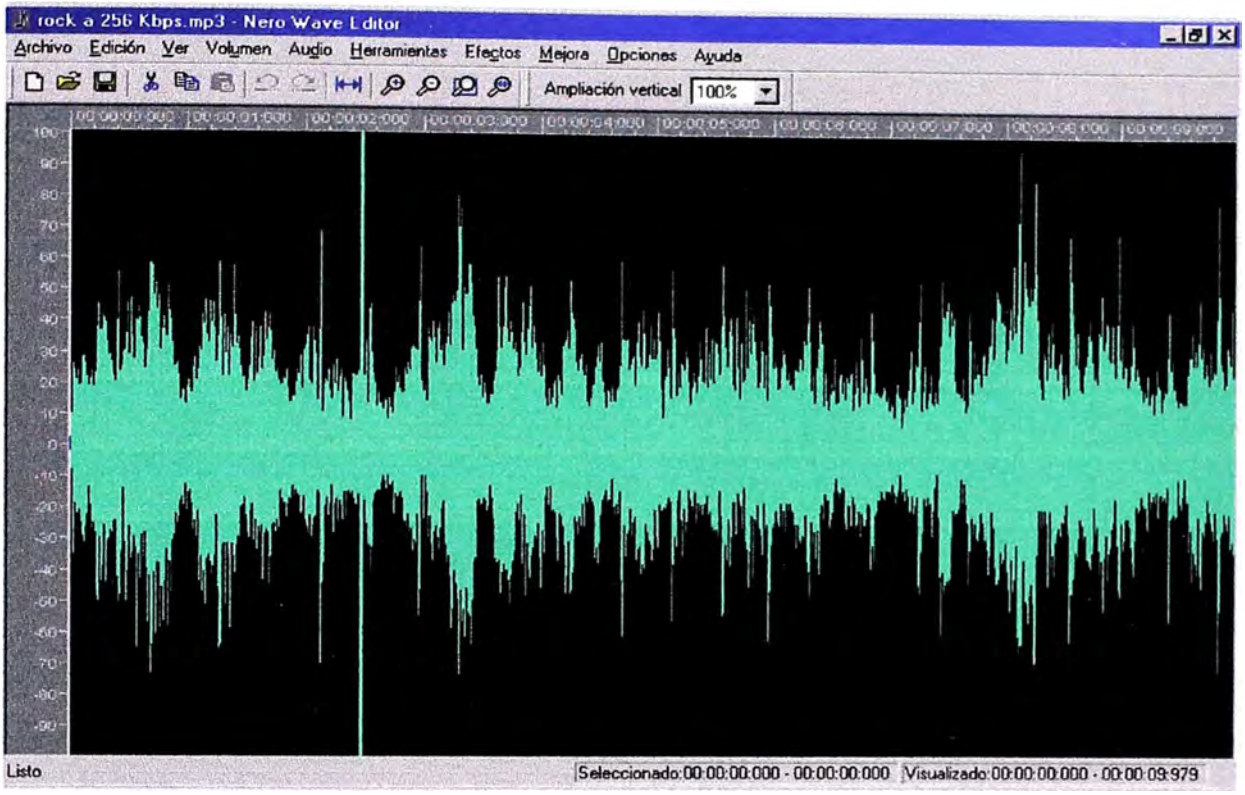


Figura PC34: **Rock a 224 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

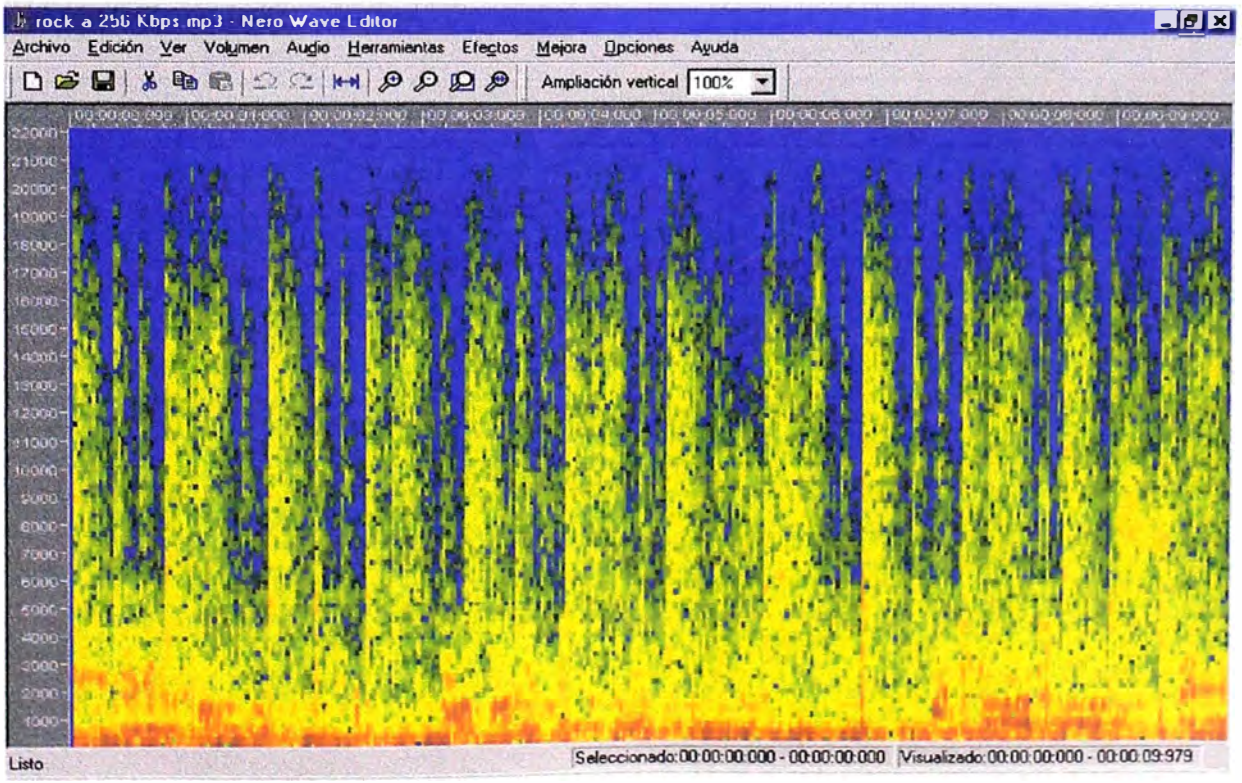
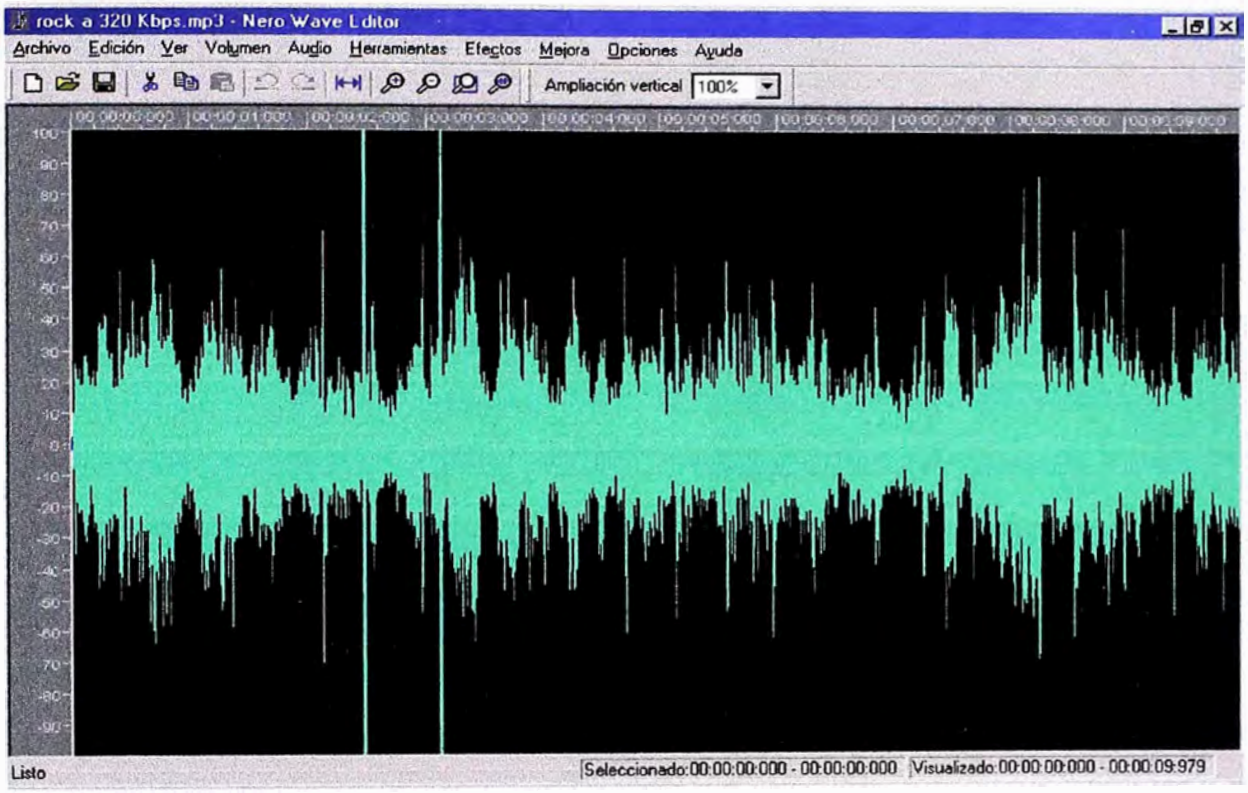


Figura PC35: **Rock a 256 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

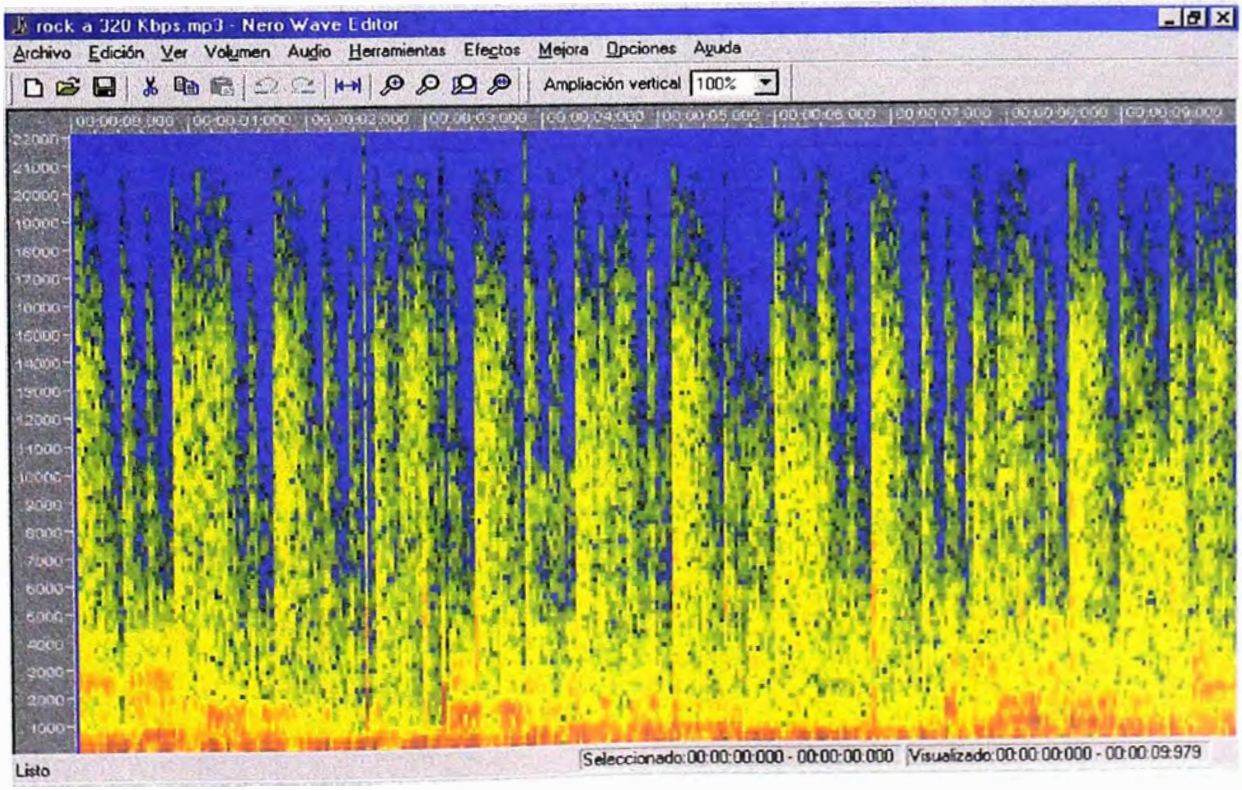


Figura PC36: **Rock a 320 Kbps.mp3**

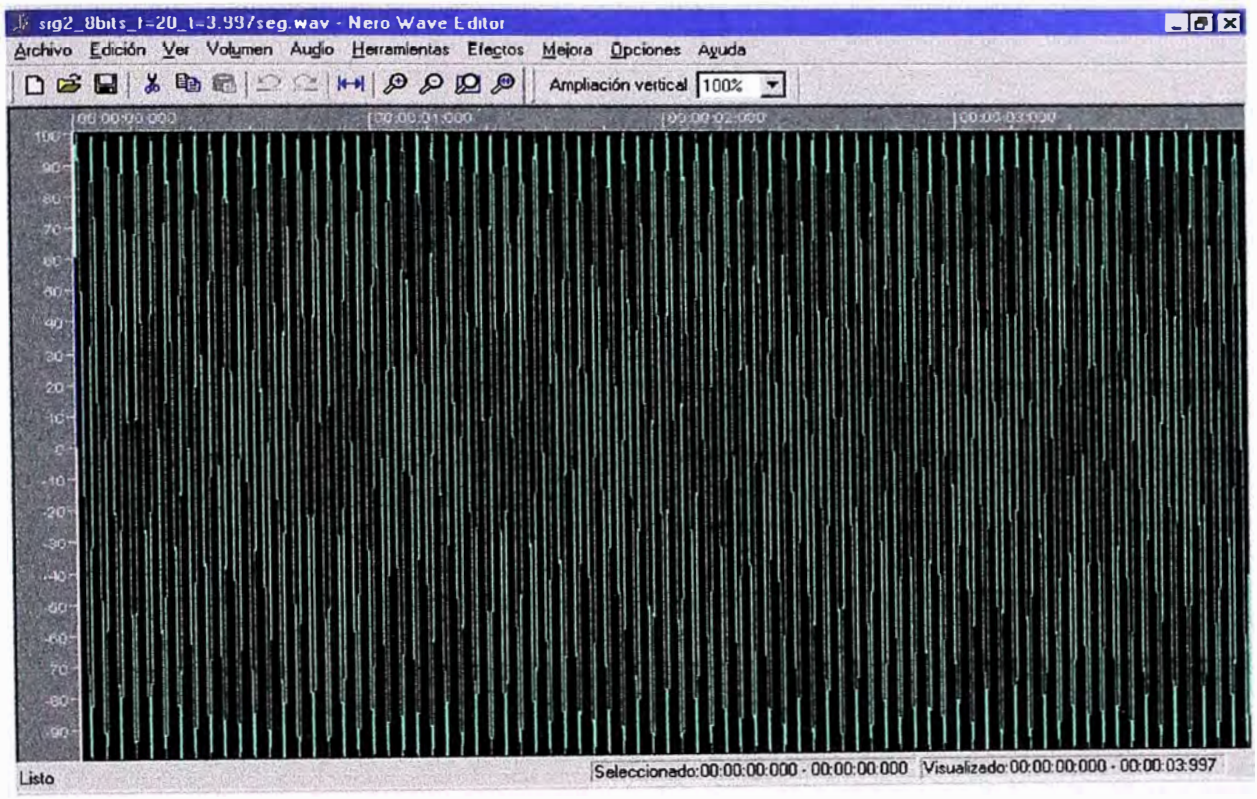
Canciones (con MP3 codificado a 10 segundos (excepto Jazz a 9.8743 segundos))				
Canción	**Clásica**	**Jazz**	**Pop (ahardday44)**	**Rock**
Tamaño original (Wav)	1064 KB	852 KB	6460 KB	949 KB
Duración (Wav)	12 347 segundos	9 88 segundos	2 minutos 29 segundos	11.013 segundos
Bitrate (Wav)	706 Kbps	706 Kbps	353 Kbps	706 Kbps
Resolución (Número de bits)	16 bits	16 bits	8 bits	16 bits
Tamaño de la muestra para comparación (Wav)	860 KB	851 KB	430 KB	860 KB
Duración de la muestra para comparación (Wav)	9.979 segundos	9.874 segundos	9.979 segundos	9.979 segundos

Tasas de bits en Kbps para codificación en MP3:	**MP3 (Tamaño en KB)**				**Duración del archivo en seg**				**Ejecución en minutos**				**Tasa de Compresión**			
	Clásica	**Jazz**	**Pop**	**Rock**	**Clásica**	**Jazz**	**Pop**	**Rock**	**Clásica**	**Jazz**	**Pop**	**Rock**	**Clásica**	**Jazz**	**Pop**	**Rock**
96	117	116	117	117	9.979	9.874	9.979	9.979	38'	42'	40'	46'	1:7.3504	1:7.3362	1:3.6752	1:7.3504
112	137	135	137	137	9.979	9.874	9.979	9.979	44'	48'	45'	53'	1:6.2773	1:6.3037	1:3.1386	1:6.2773
128	156	155	156	156	9.979	9.874	9.979	9.979	53'	48'	52'	1h 1'	1:5.5128	1:5.4903	1:2.7564	1:5.5128
160	195	193	195	195	9.979	9.874	9.979	9.979	1h 1'	1h 5'	1h 9'	1h 1'	1:4.4102	1:4.4093	1:2.2051	1:4.4102
192	234	232	234	234	9.979	9.874	9.979	9.979	1h 19'	1h 10'	1h 15'	1h 17'	1:3.6752	1:3.6681	1:1.8376	1:3.6752
224	273	270	273	273	9.979	9.874	9.979	9.979	1h 25'	1h 15'	1h 38'	1h 23'	1:3.1501	1:3.1518	1:1.5750	1:3.1501
256	312	309	312	312	9.979	9.874	9.979	9.979	1h 39'	1h 32'	1h 36'	1h 43'	1:2.7564	1:2.7540	1:1.3782	1:2.7564
320	390	386	390	390	9.979	9.874	9.979	9.979	2h 23'	1h 59'	2h 12'	2h	1:2.2051	1:2.2046	1:1.1025	1:2.2051

Tabla Pruebas 1: Resultados de las pruebas para las canciones.

Señales Sinusoidales

Visualización de Onda



Visualización de Espectrograma

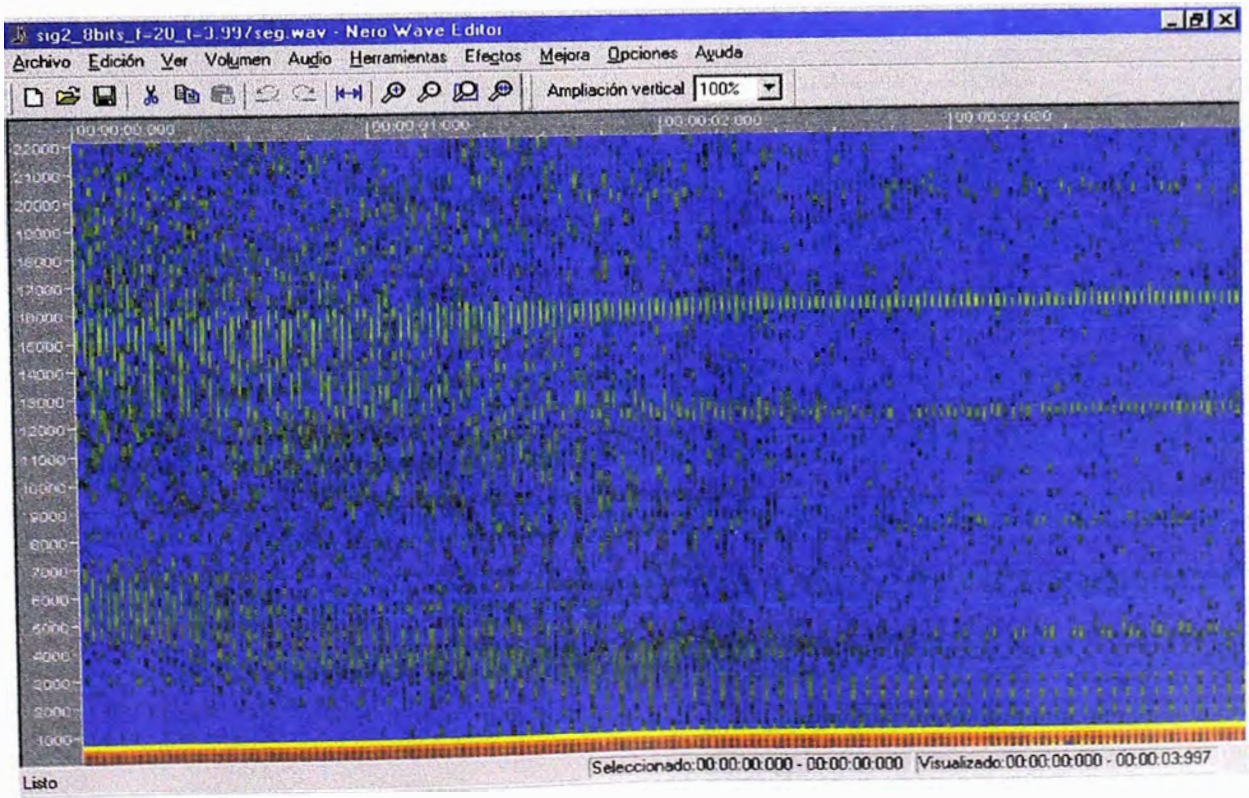
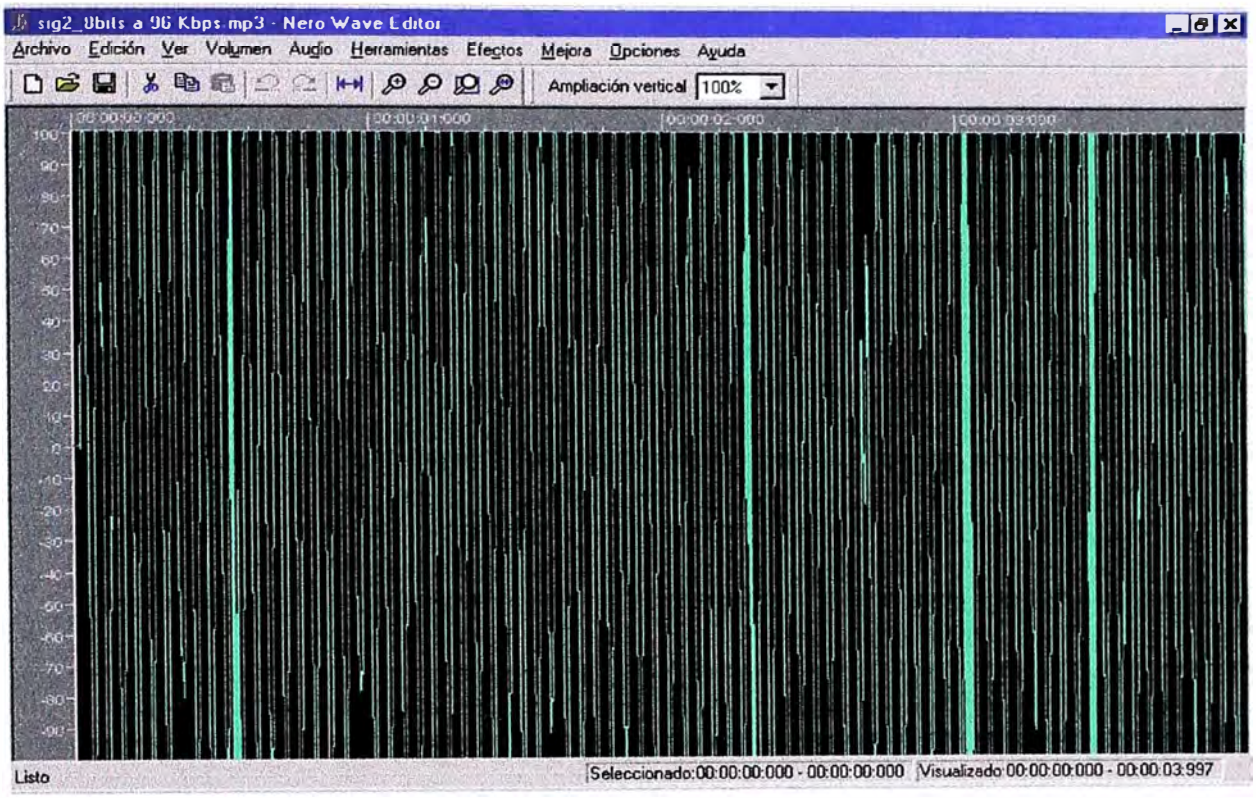


Figura PS1: **sig2_8bits_f=20_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

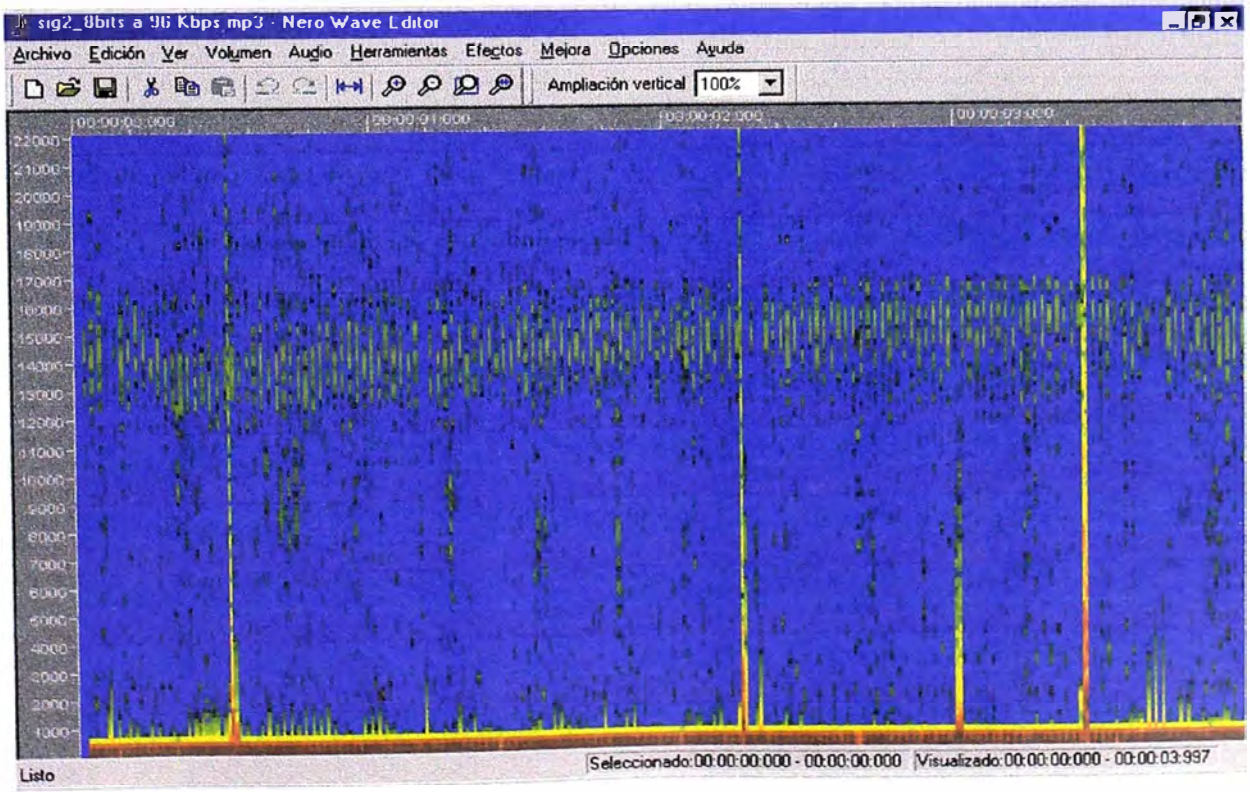
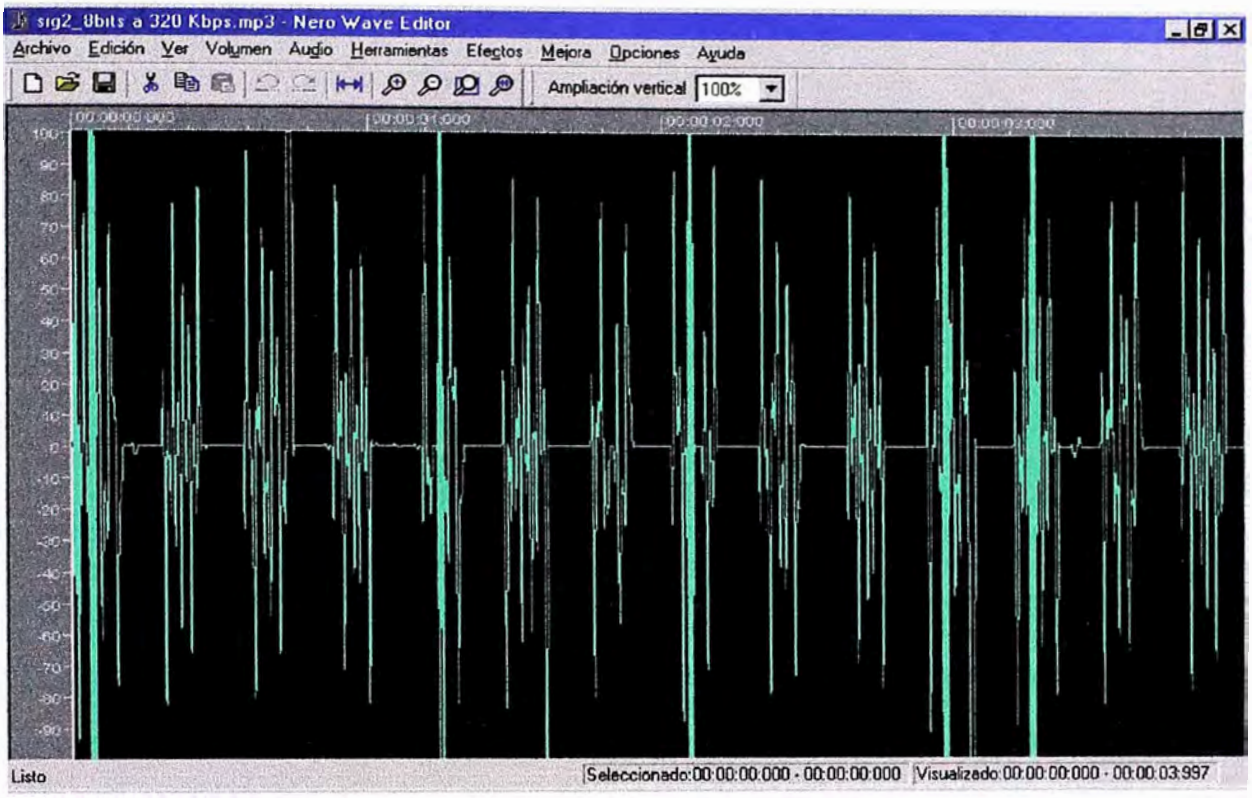


Figura PS2: **sig2_8bits a 96 Kbps_f=20_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

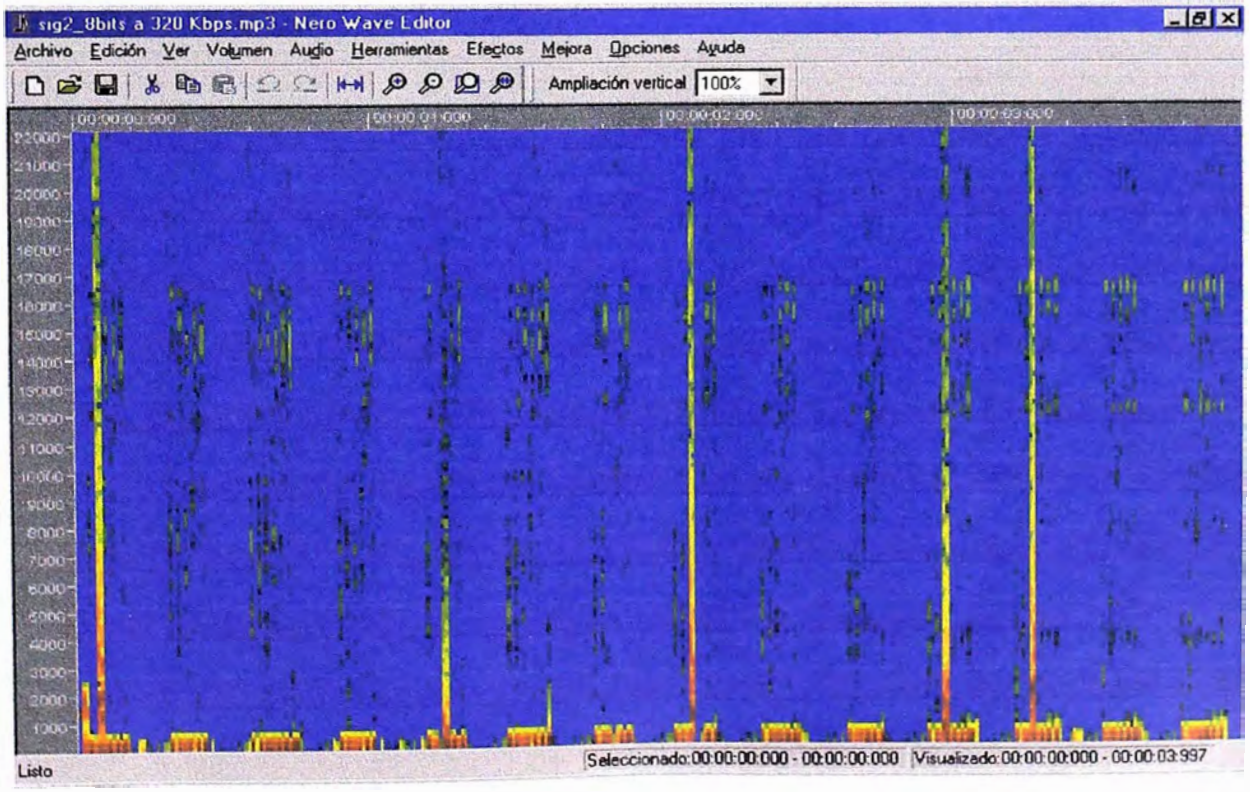
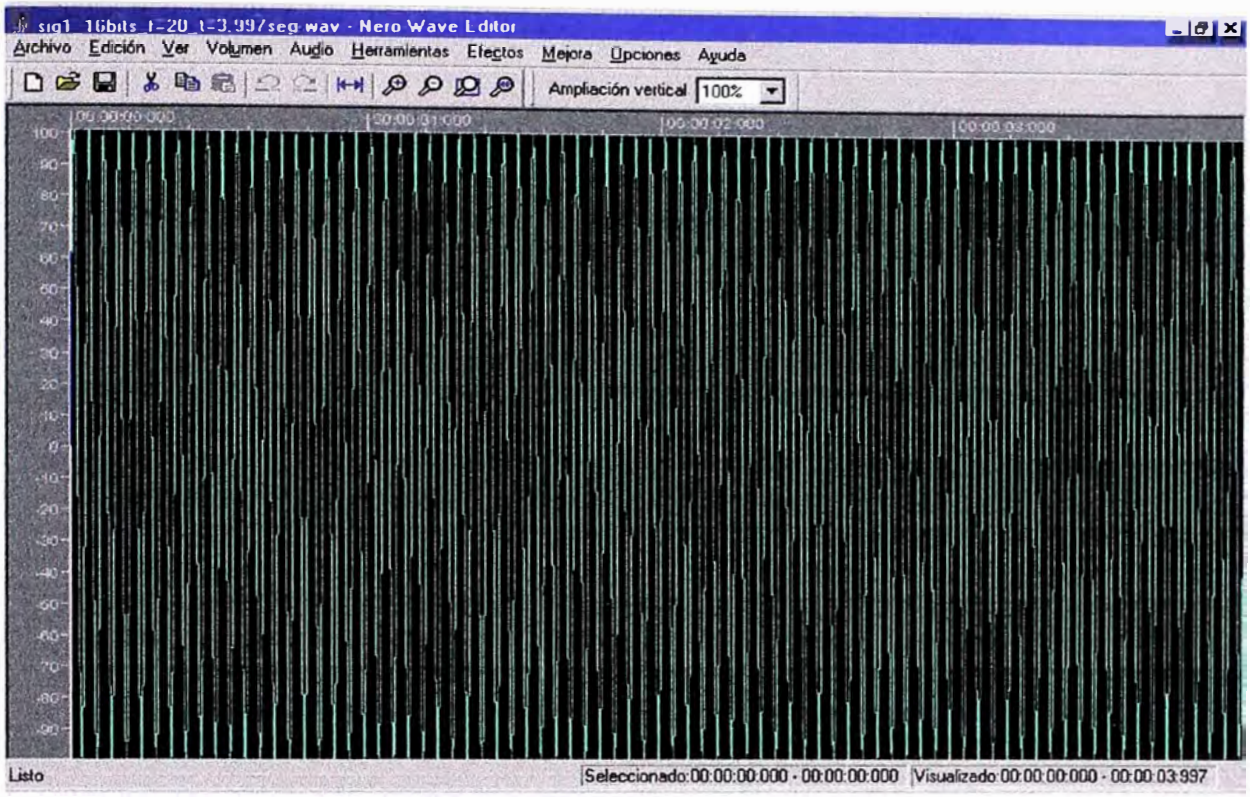


Figura PS3: **sig2_8bits a 320 Kbps_f=20_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

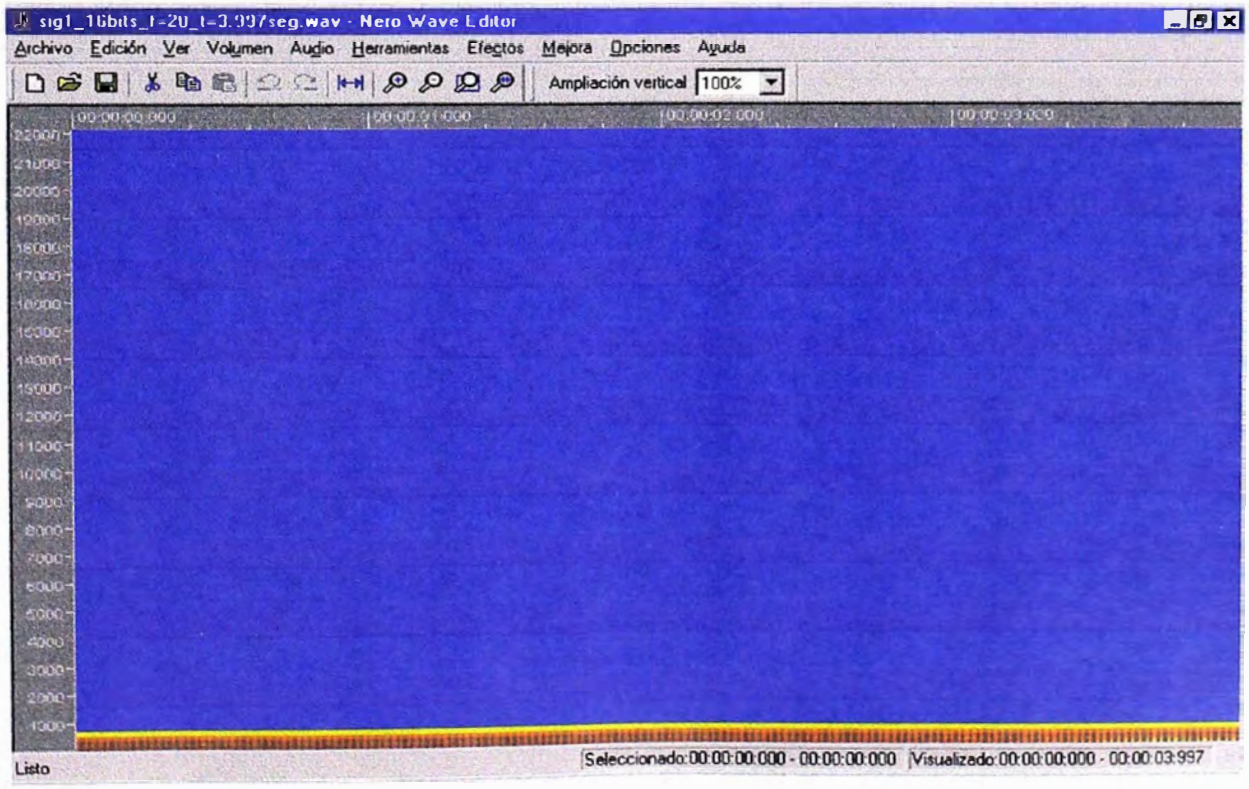
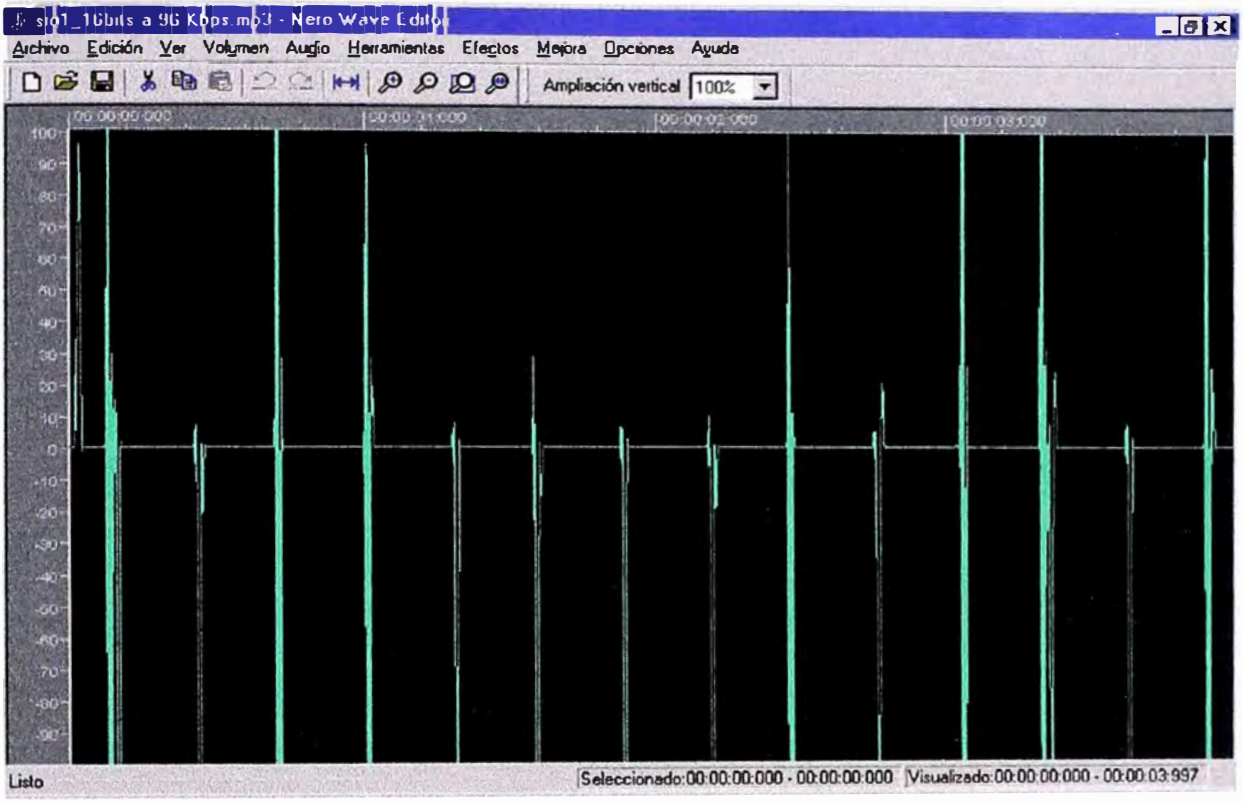


Figura PS4: **sig1_16bits_f=20_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

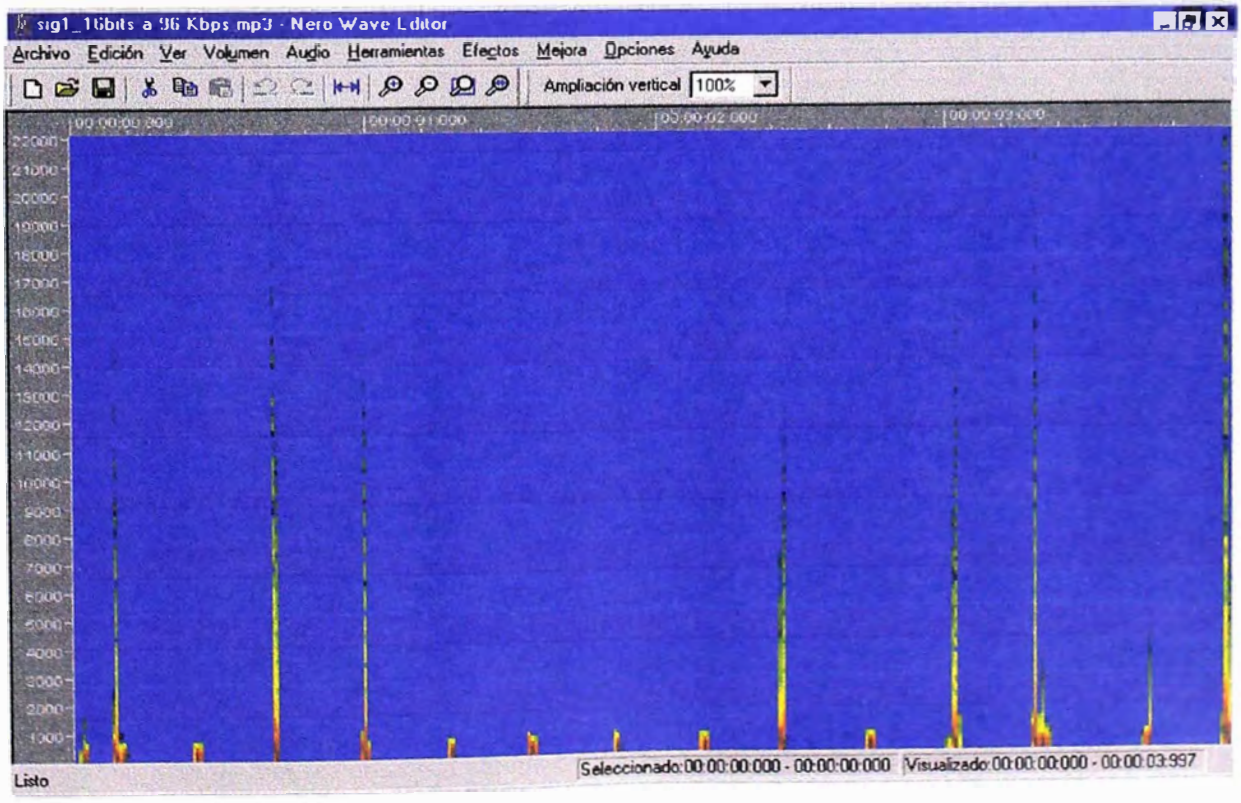
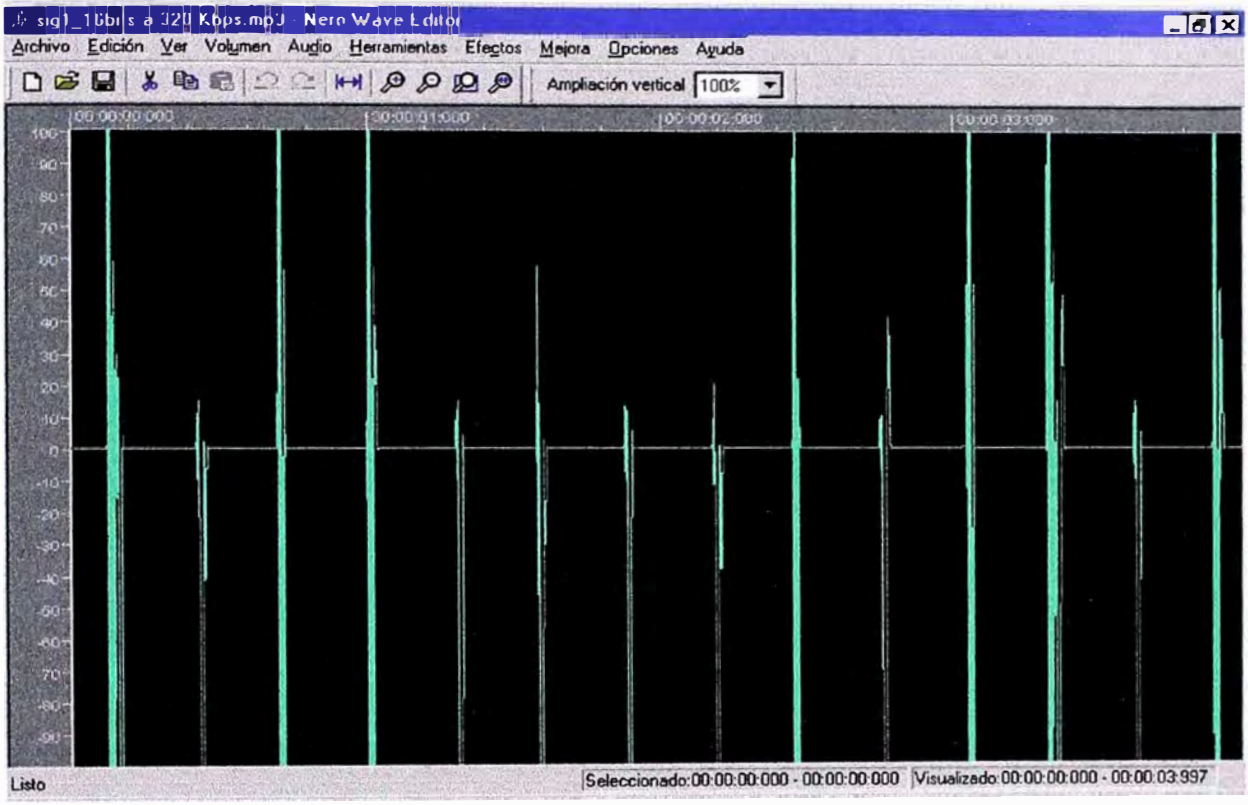


Figura PS5: **sig1_16bits a 96 Kbps_f=20_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

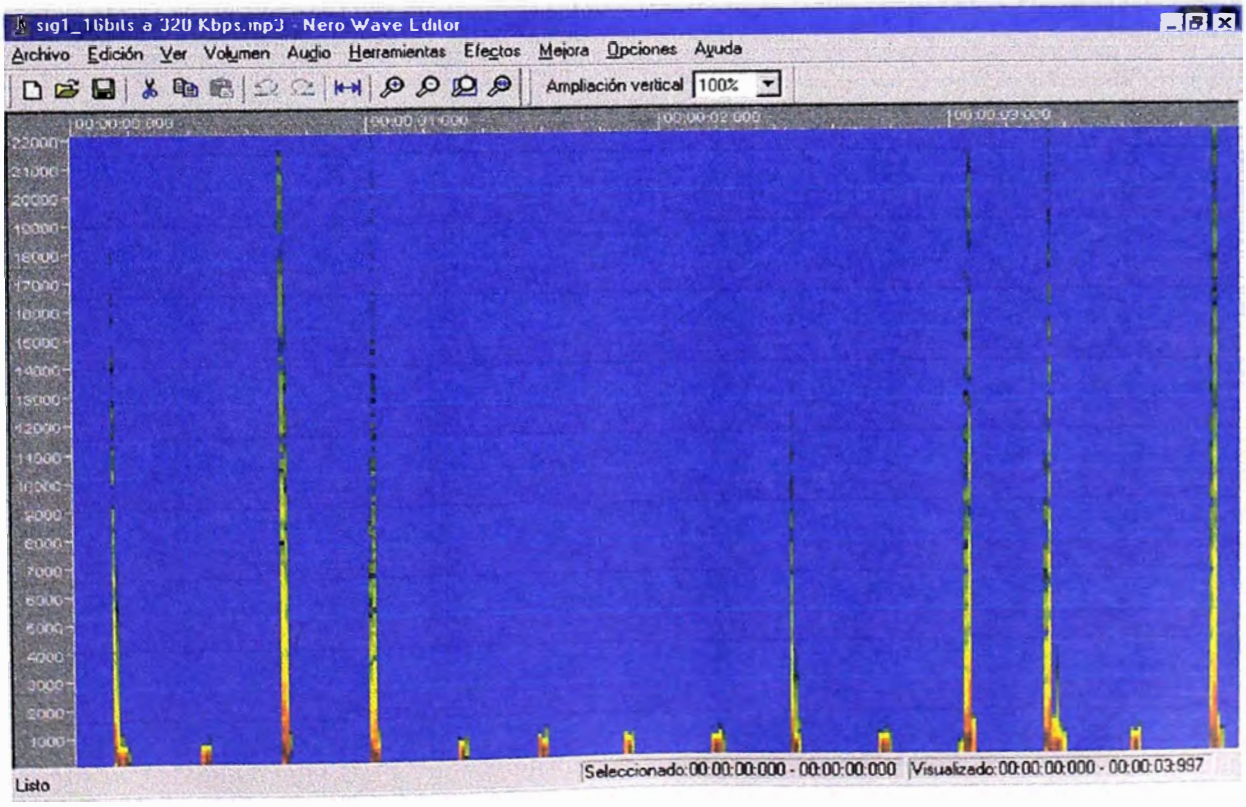
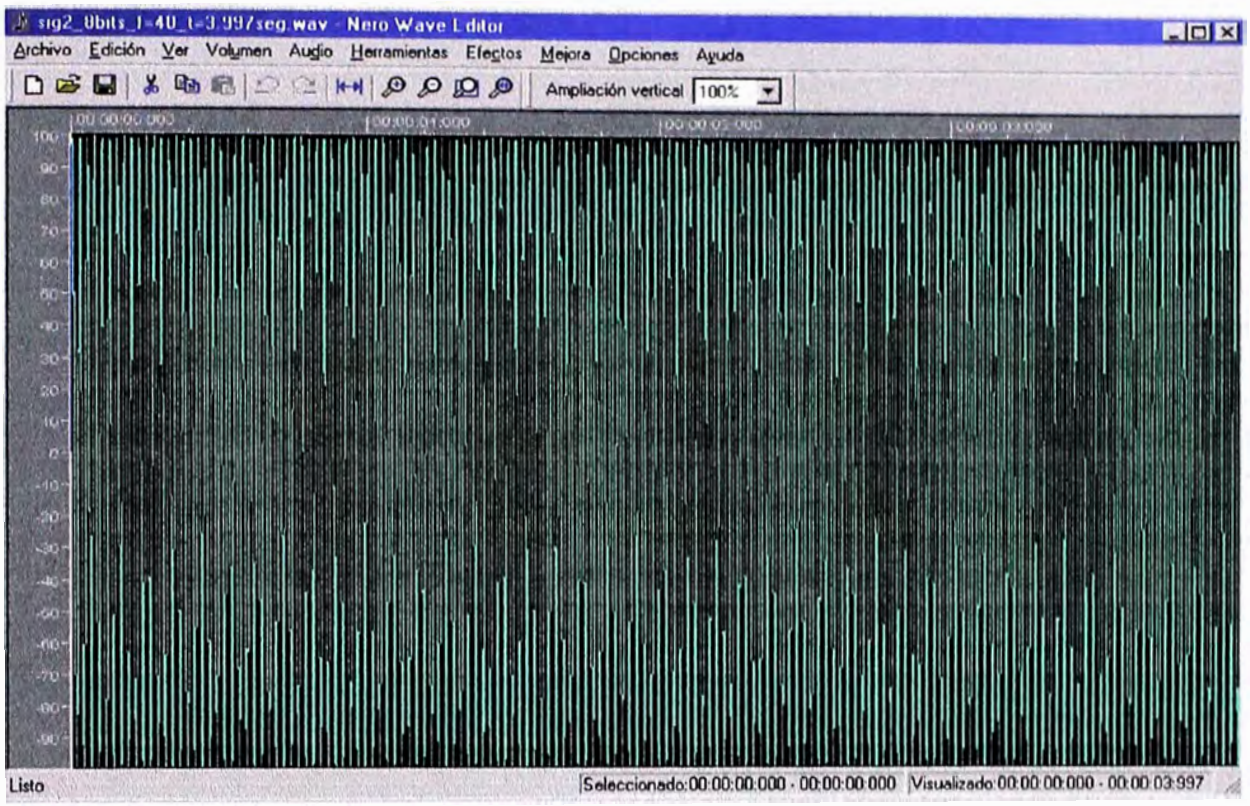


Figura PS6: **sig1_16bits a 320 Kbps_f=20_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

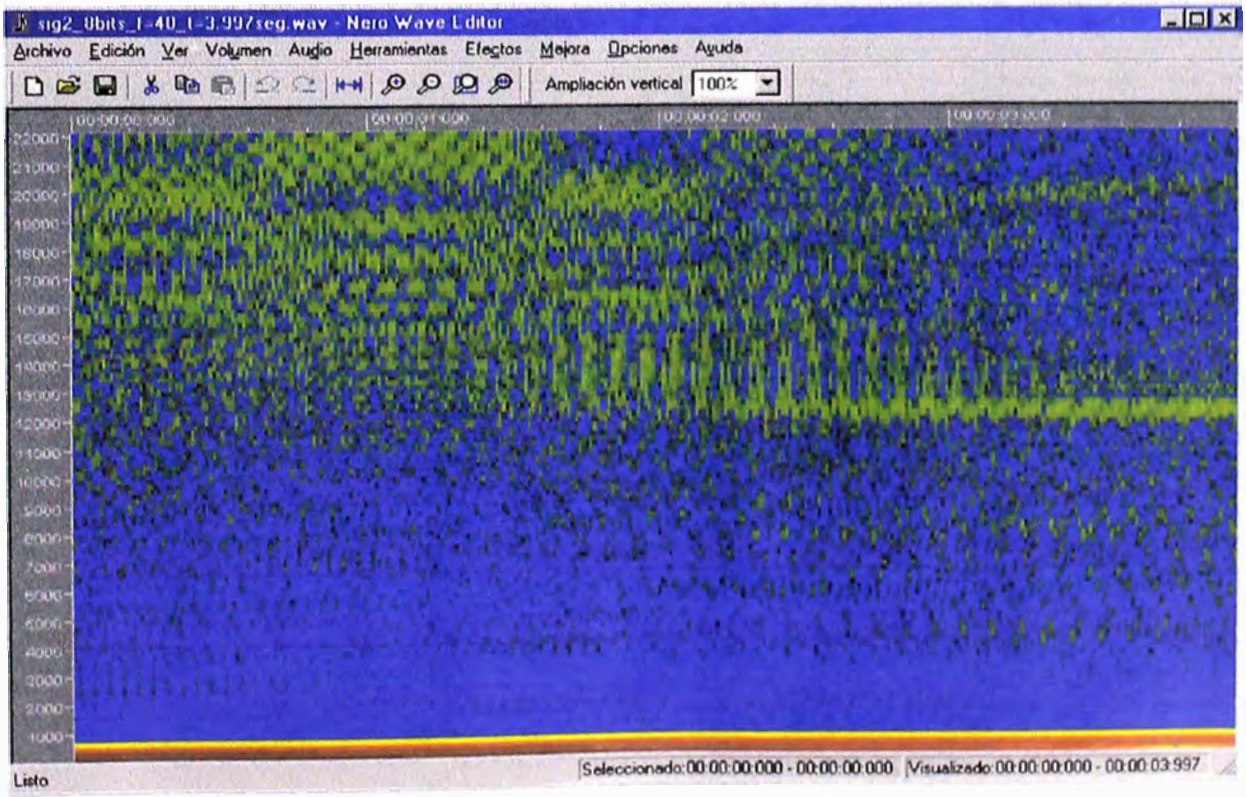
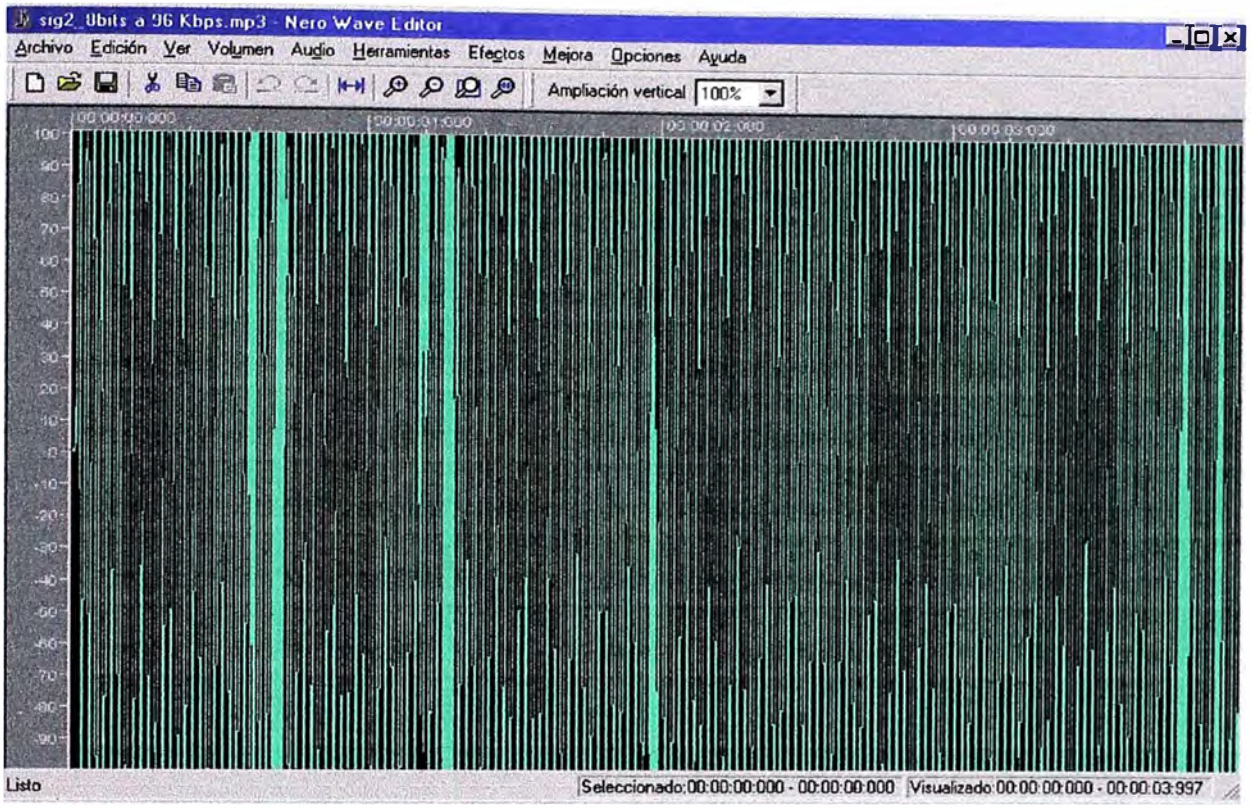


Figura PS7: **sig2_8bits_f=40_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

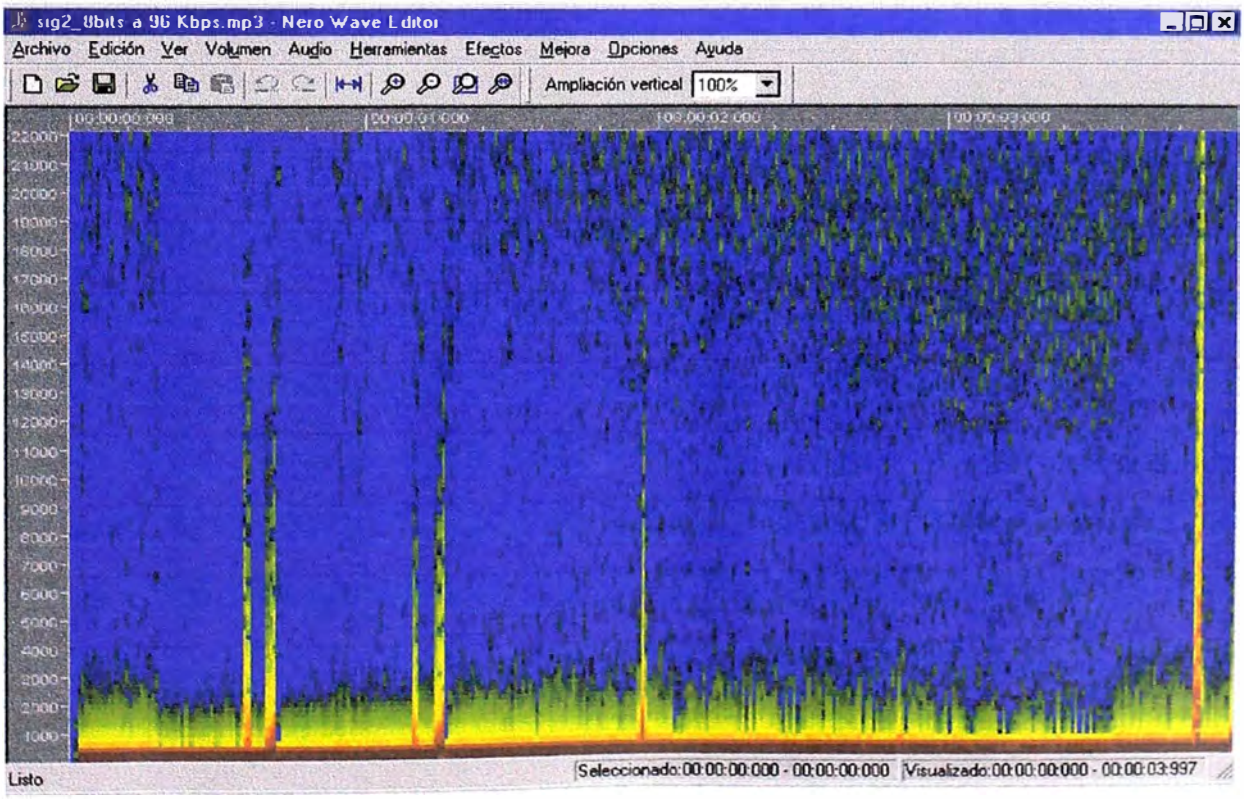
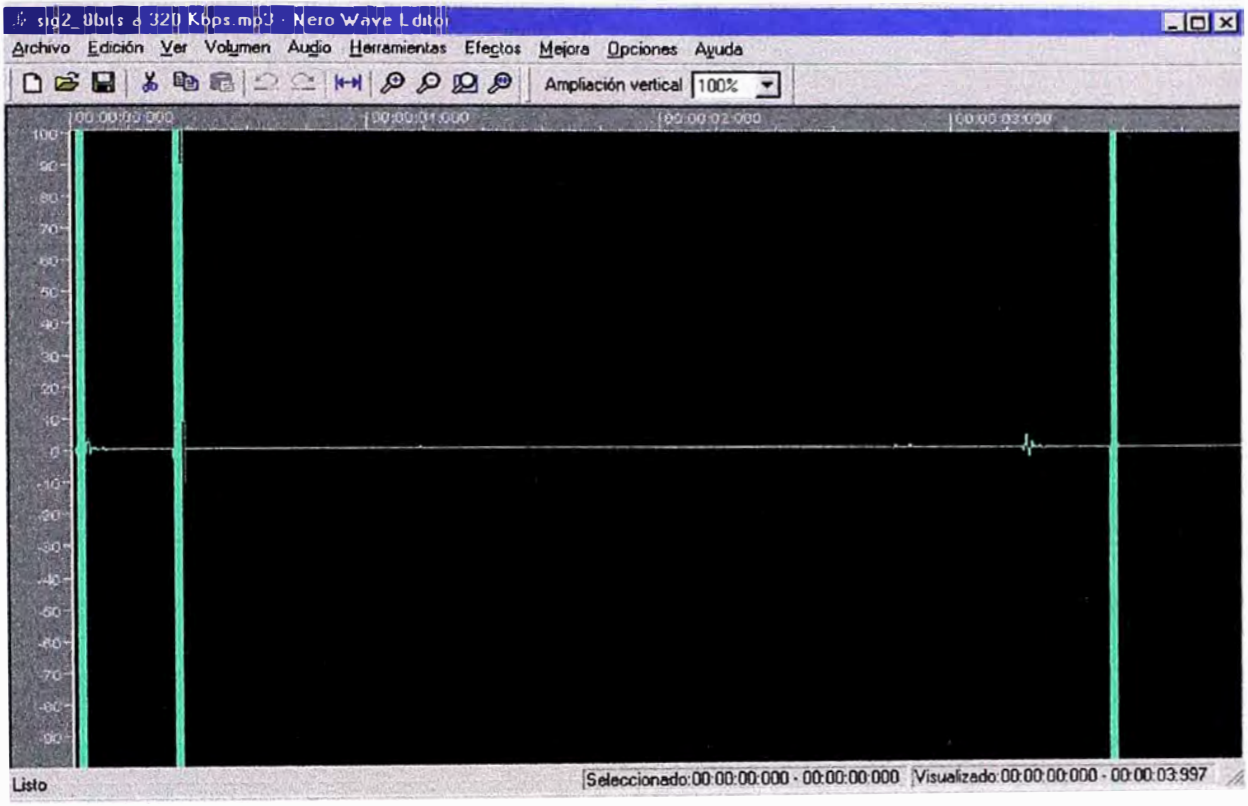


Figura PS8: **sig2_8bits a 96 Kbps_f=40_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

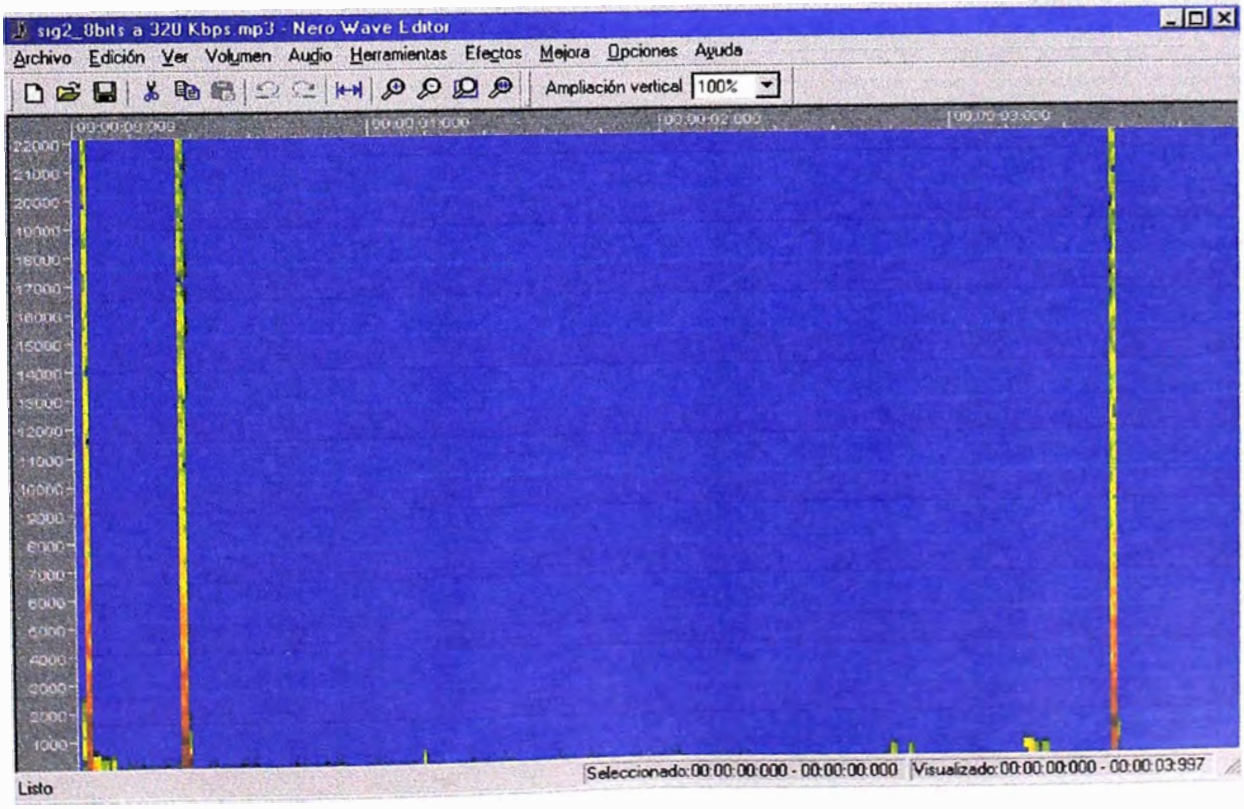
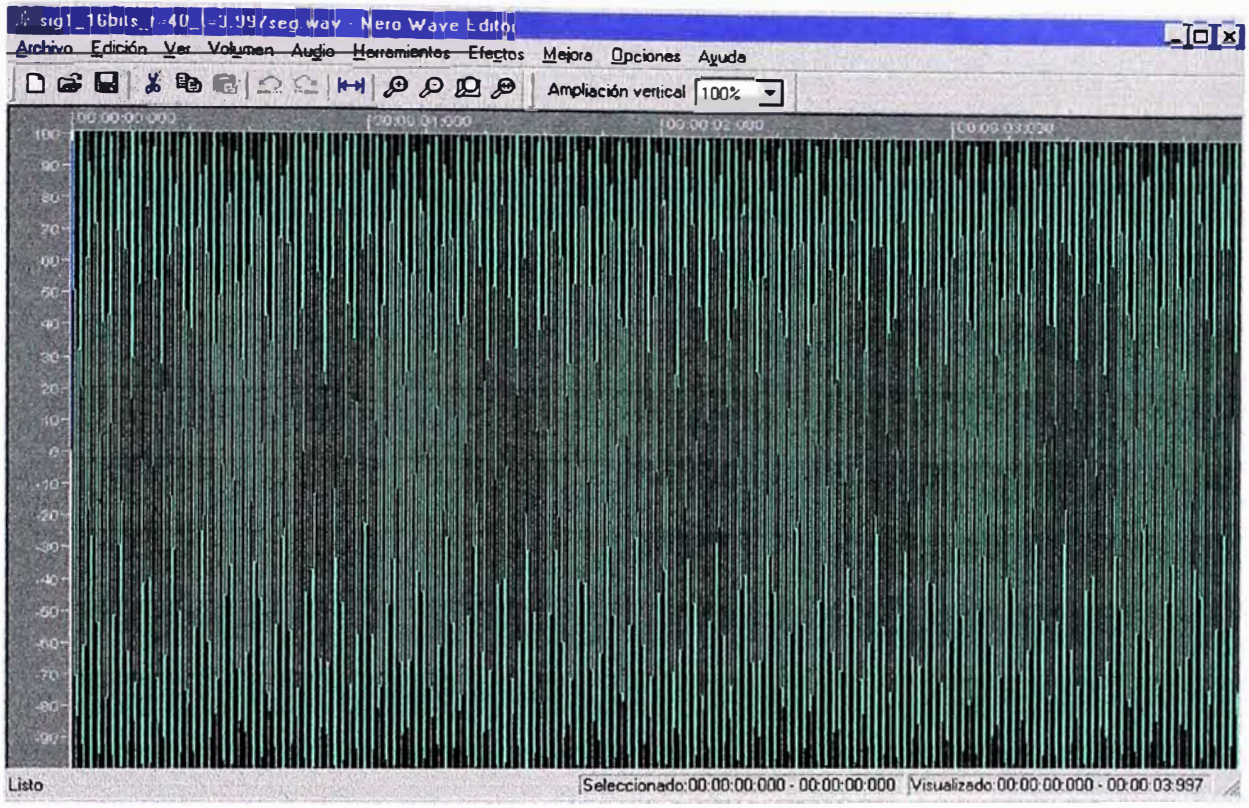


Figura PS9: **sig2_8bits a 320 Kbps_f=40_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

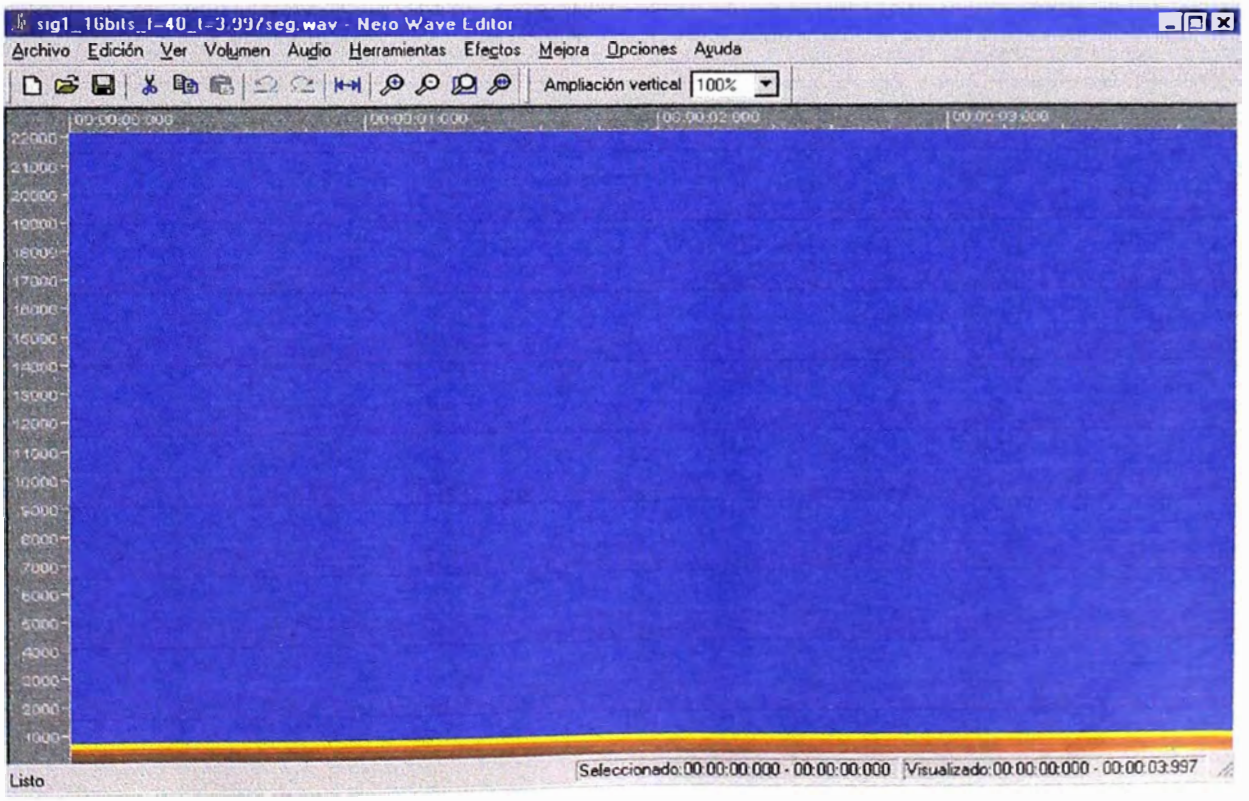
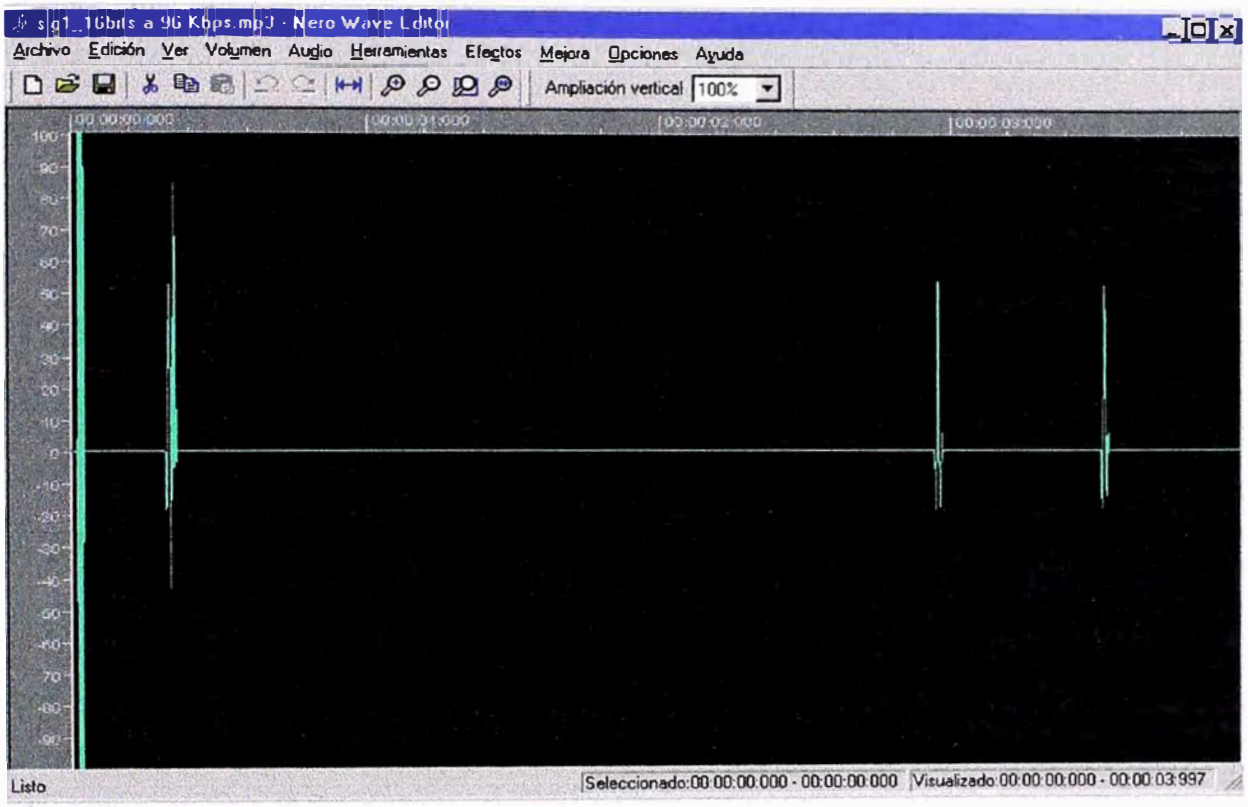


Figura PS10: **sig1_16bits_f=40_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

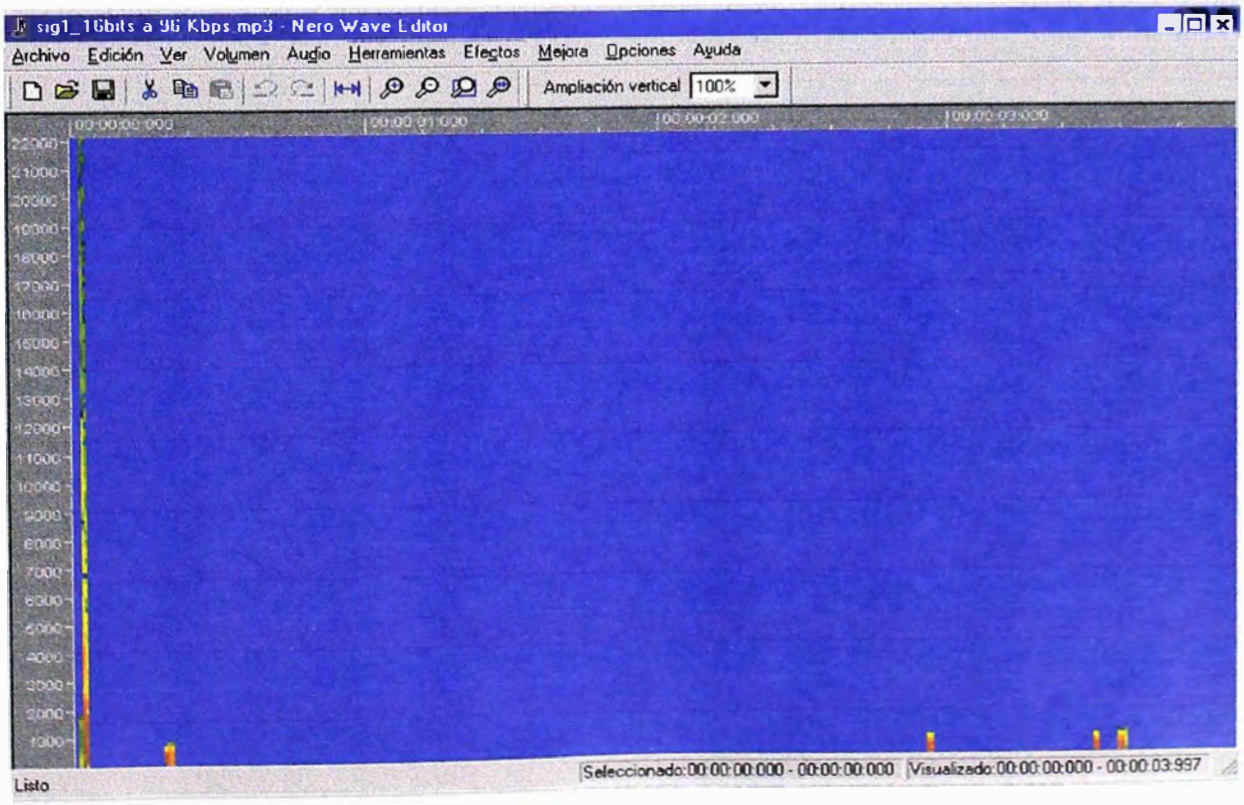
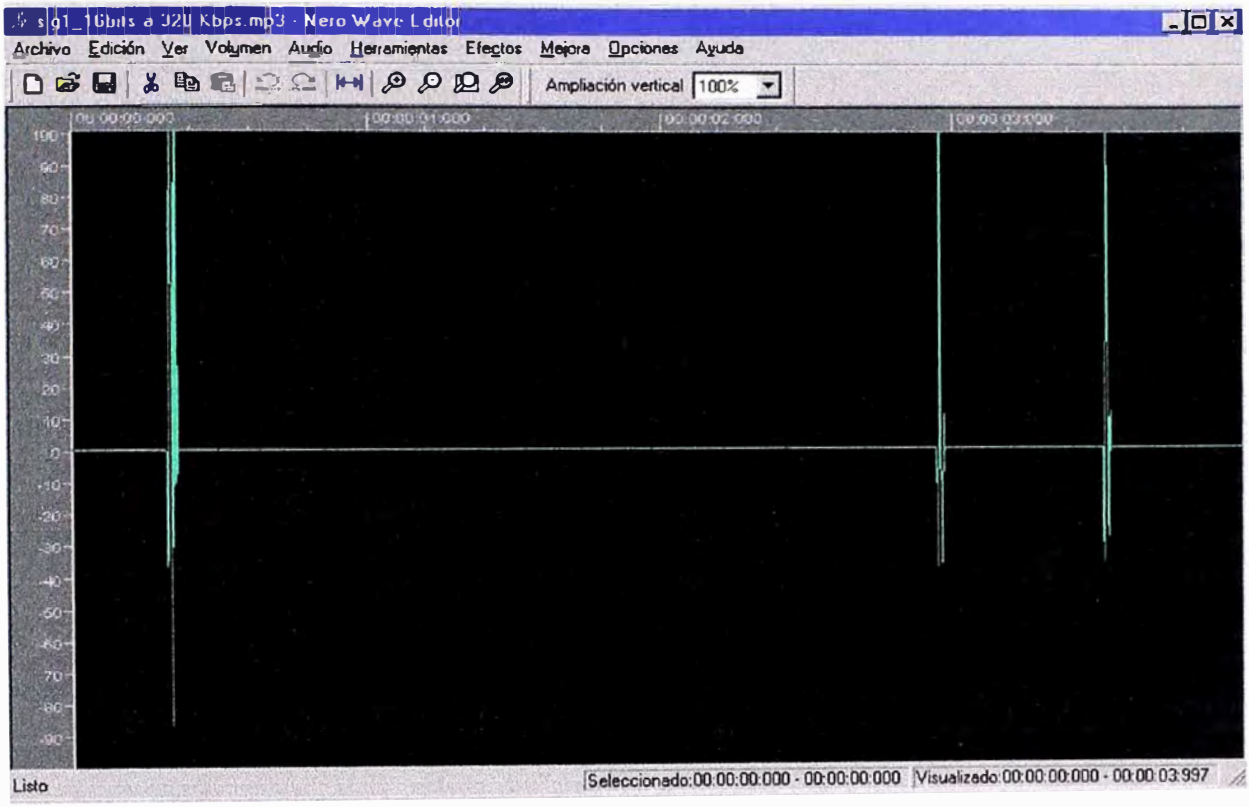


Figura PS11: **sig1_16bits a 96 Kbps_f=40_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

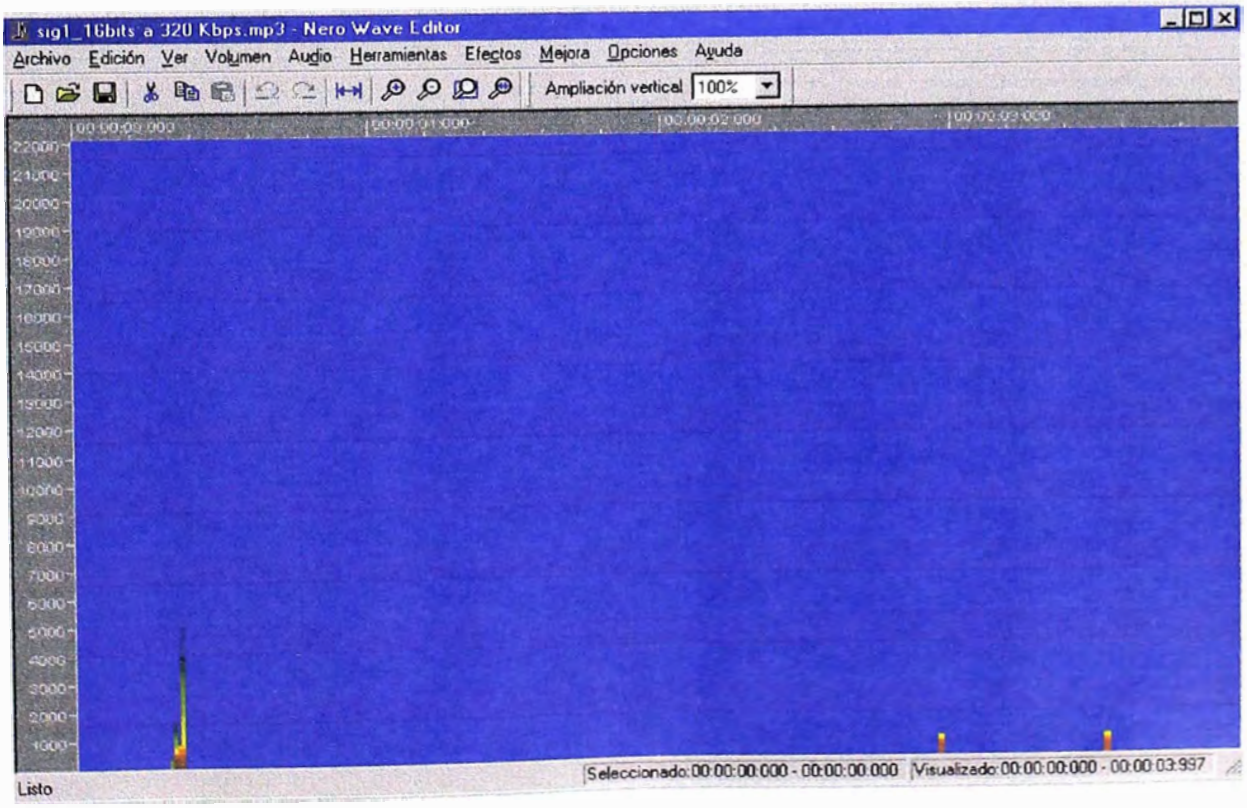
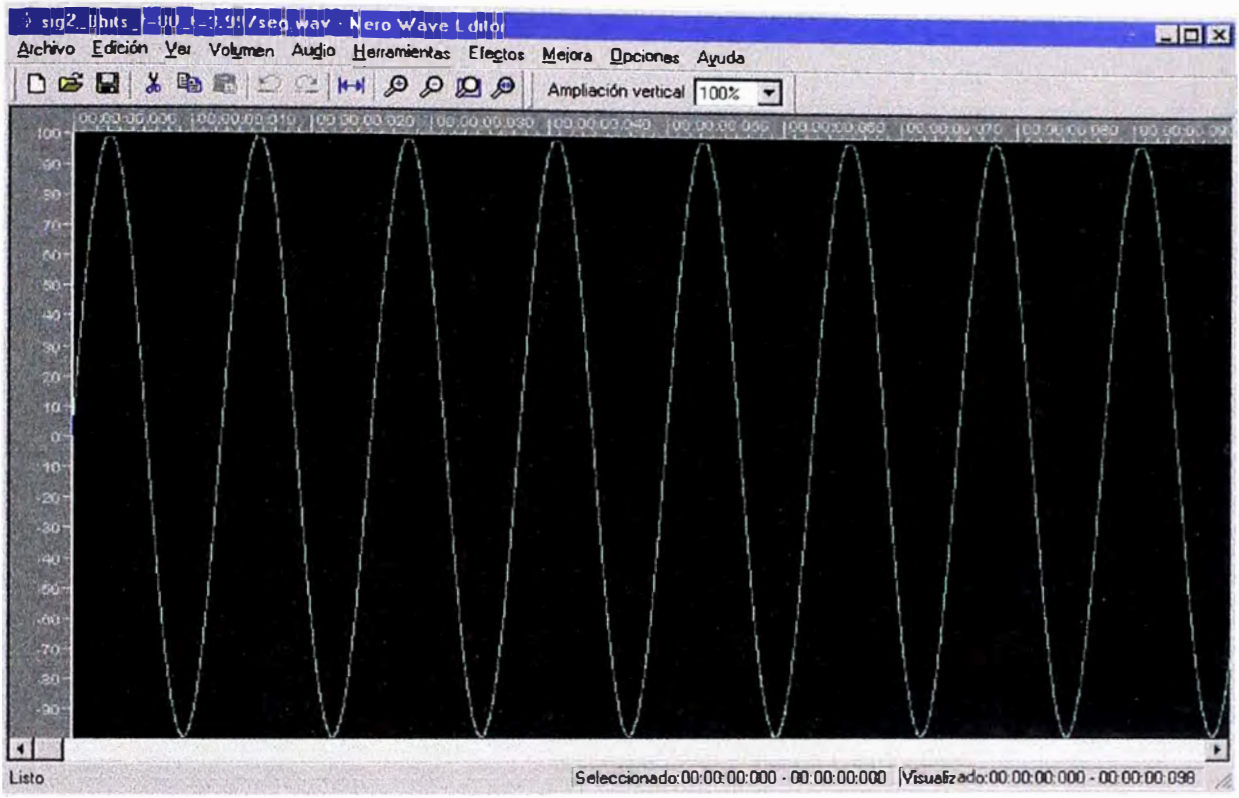


Figura PS12: **sig1_16bits a 320 Kbps_f=40_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

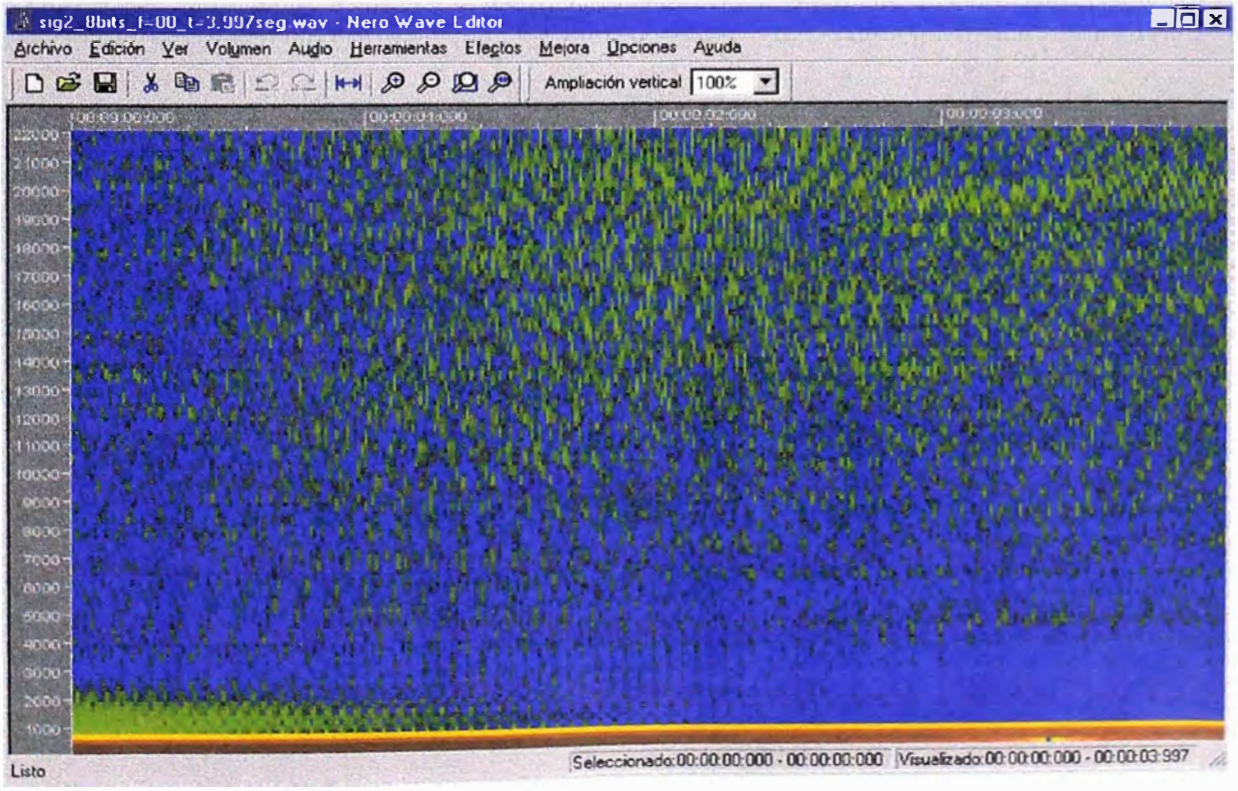
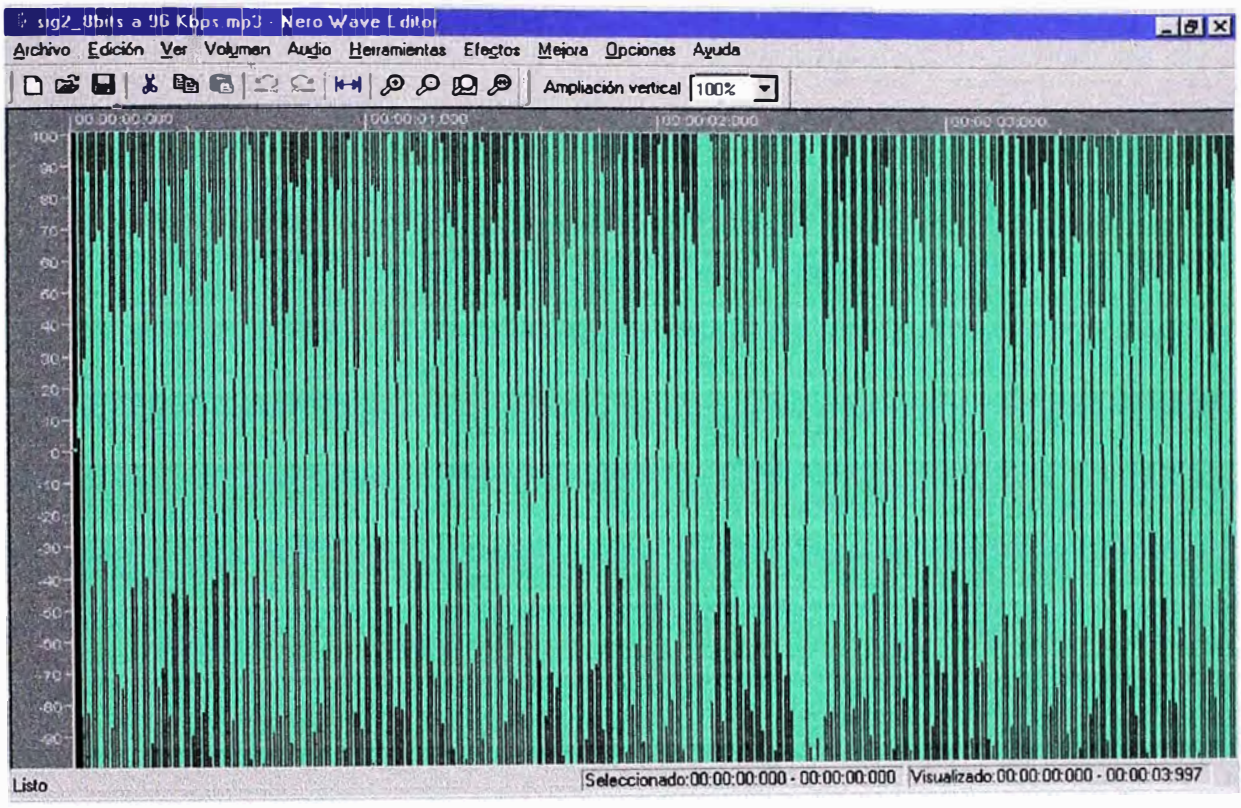


Figura PS13: **sig2_8bits_f=80_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

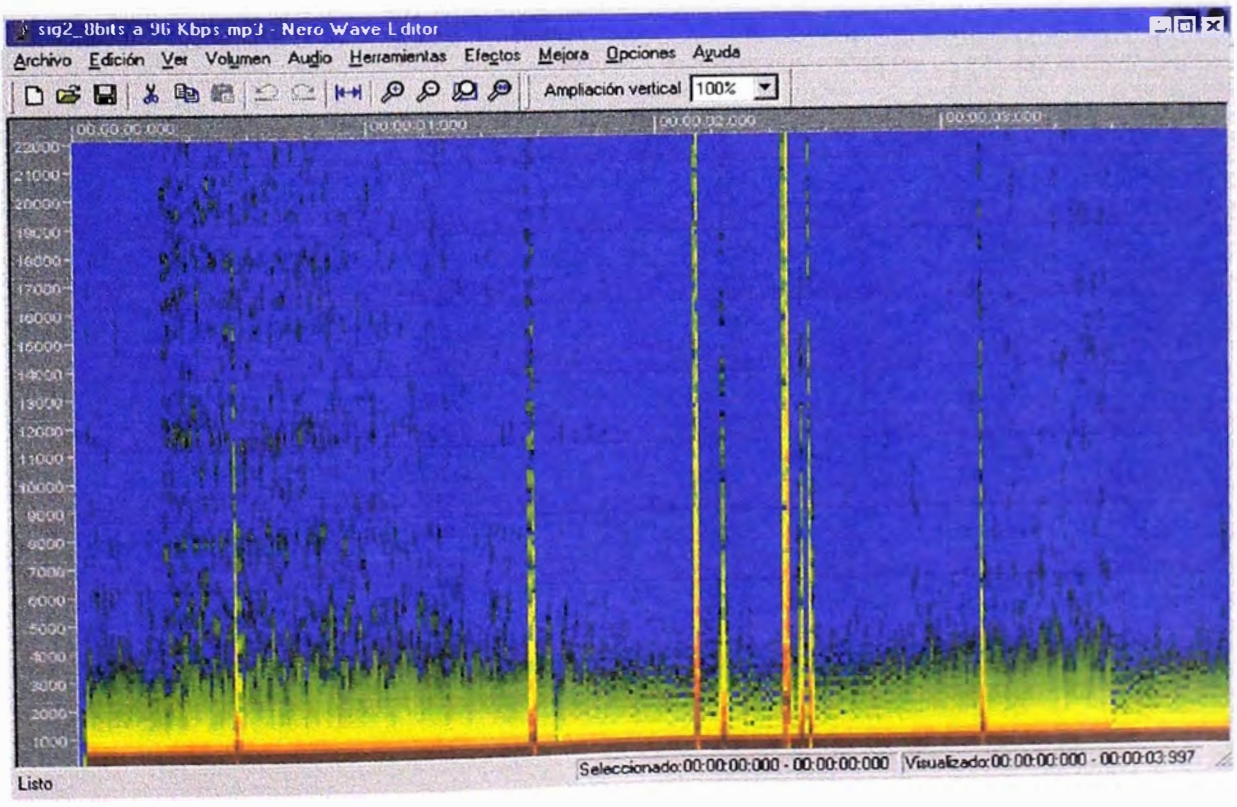
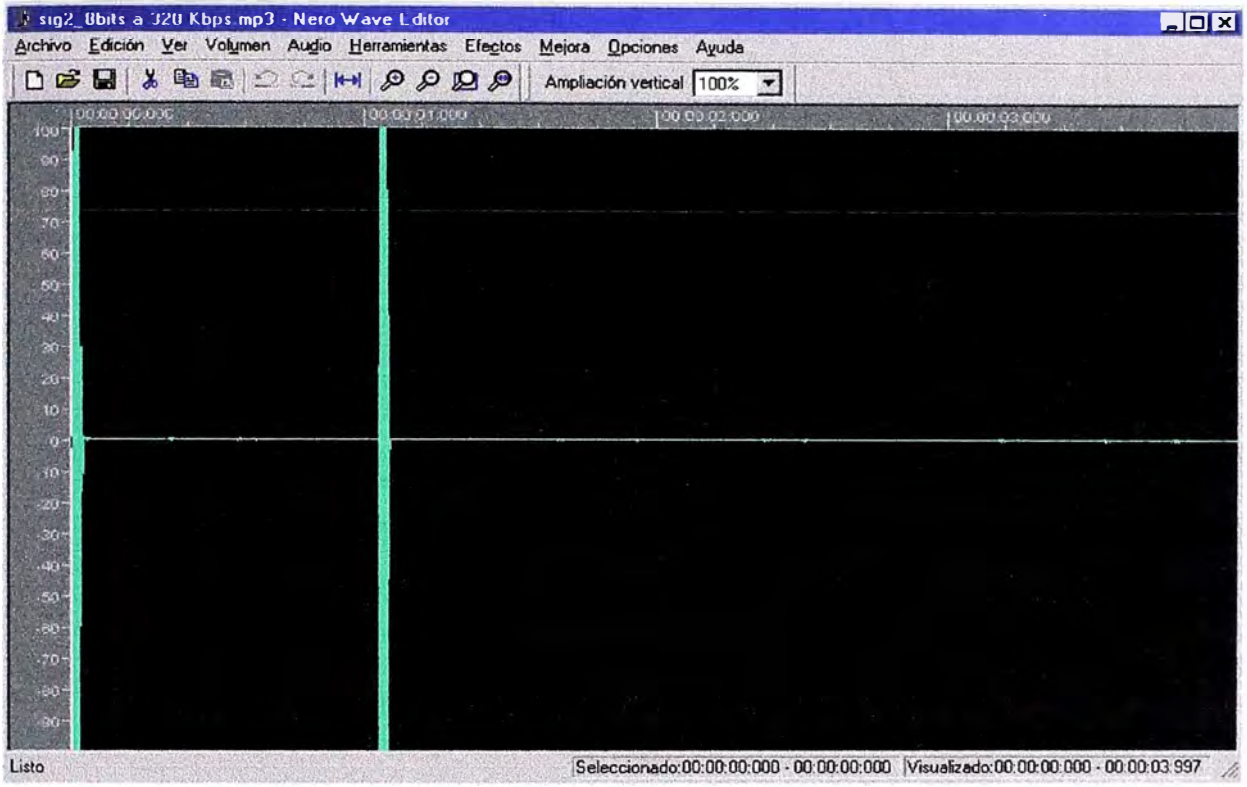


Figura PS14: **sig2_8bits a 96 Kbps_f=80_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

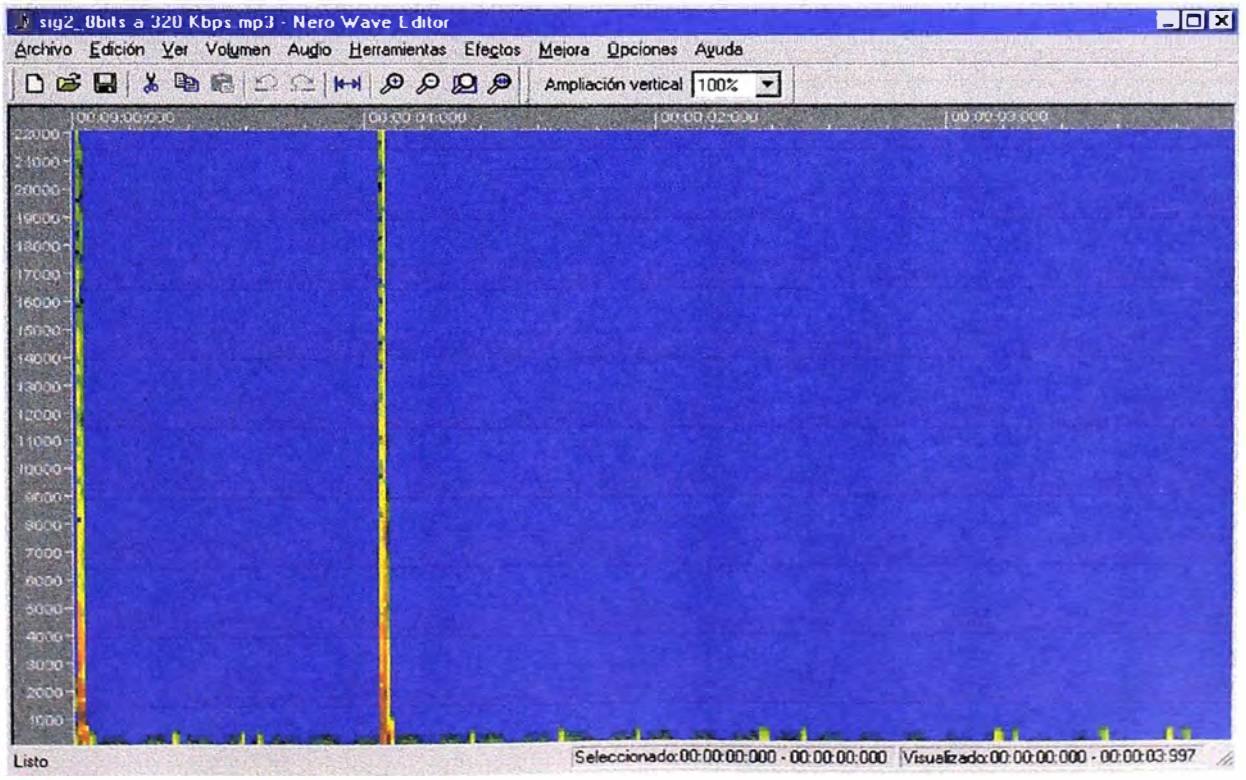
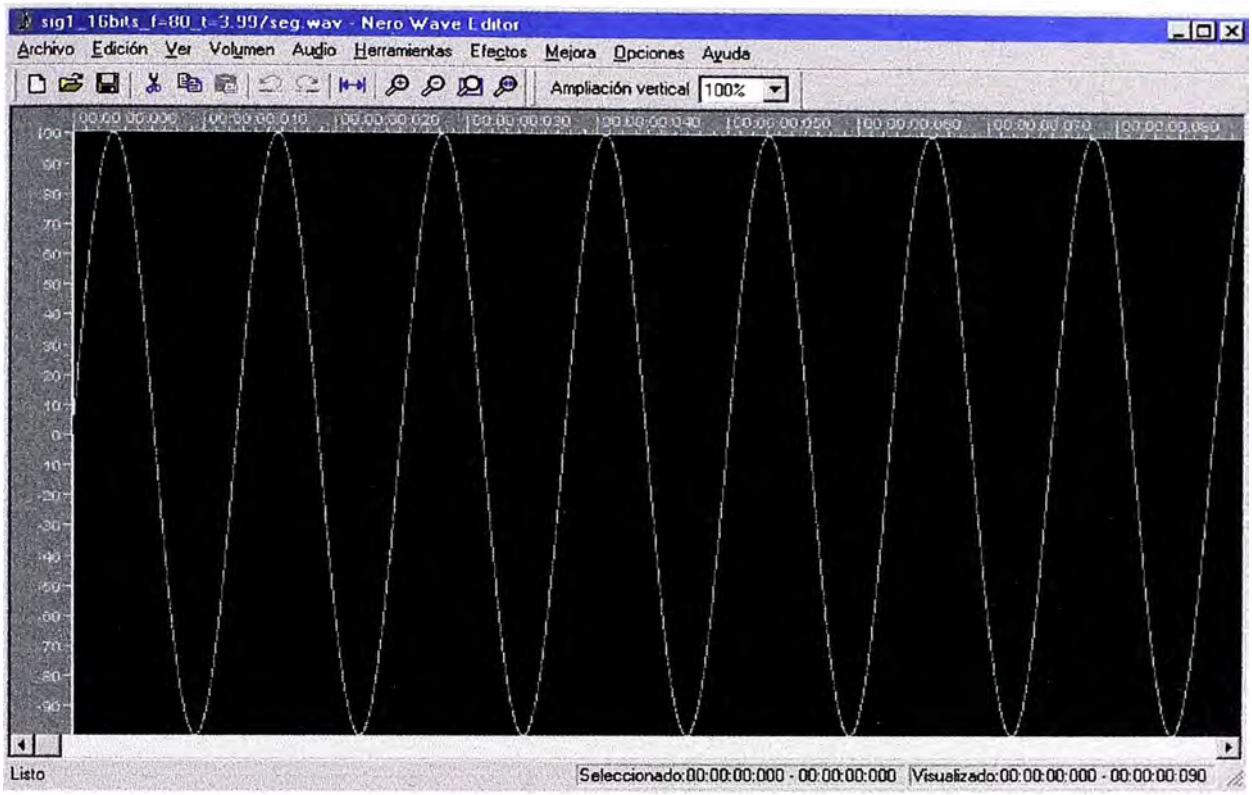


Figura PS15: **sig2_8bits a 320 Kbps_f=80_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

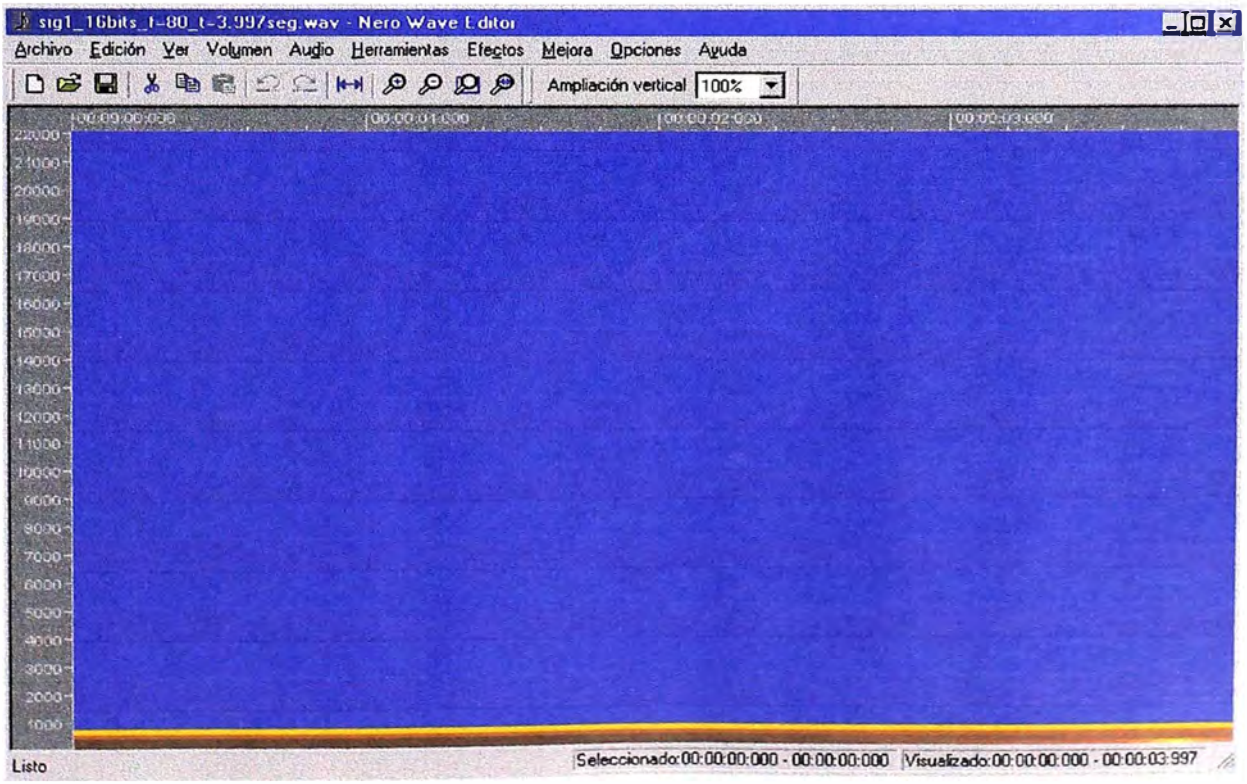
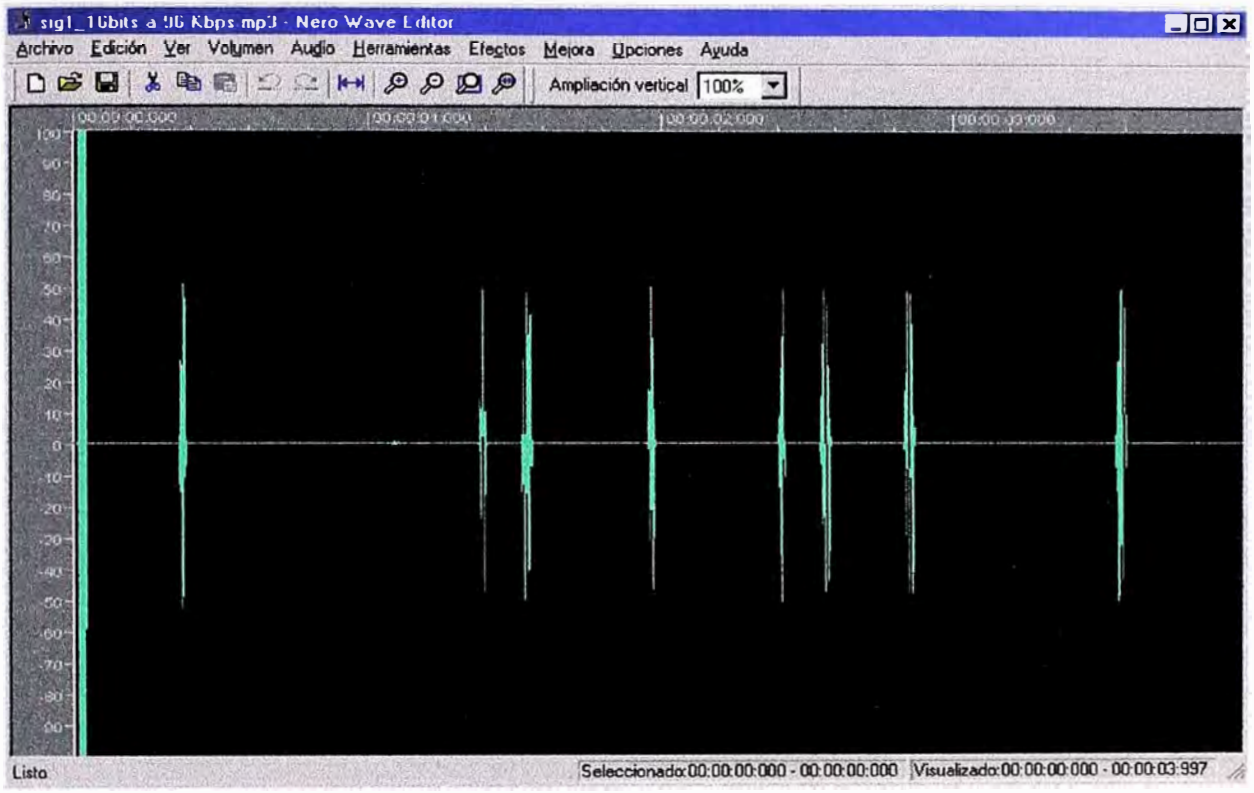


Figura PS16: **sig1_16bits_f=80_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

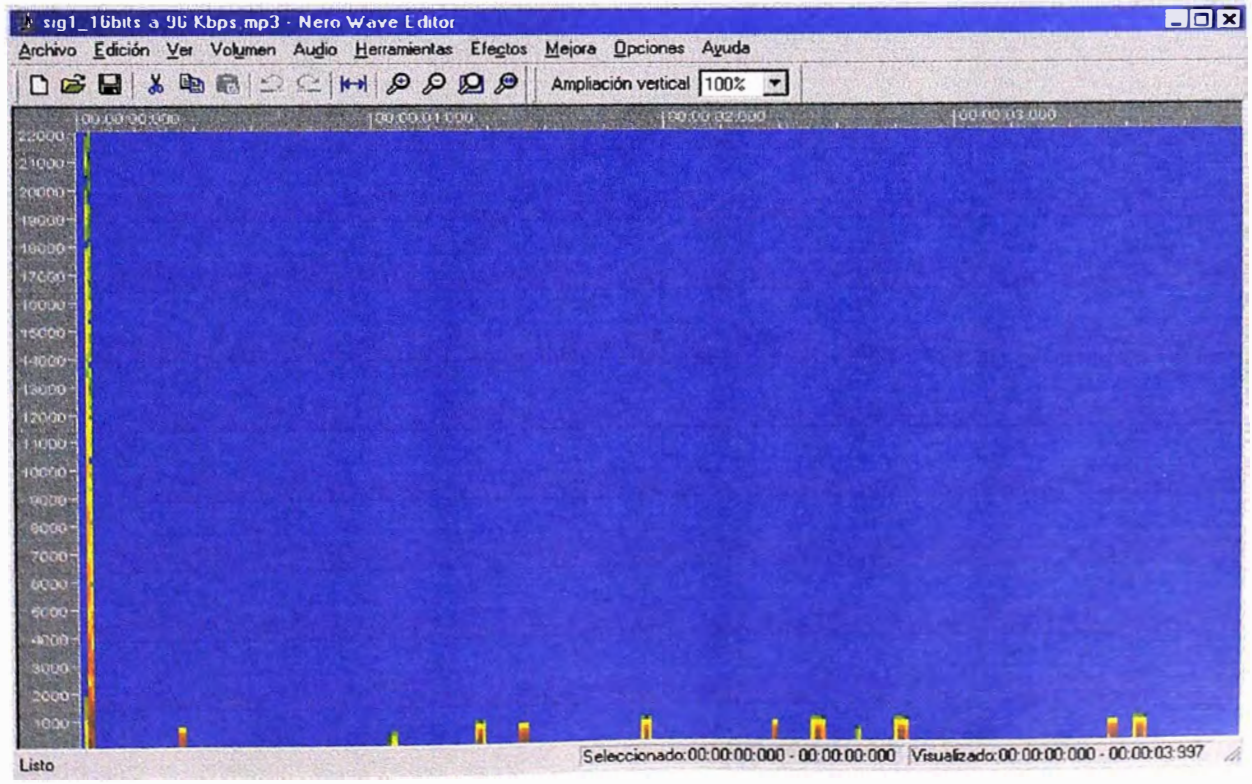
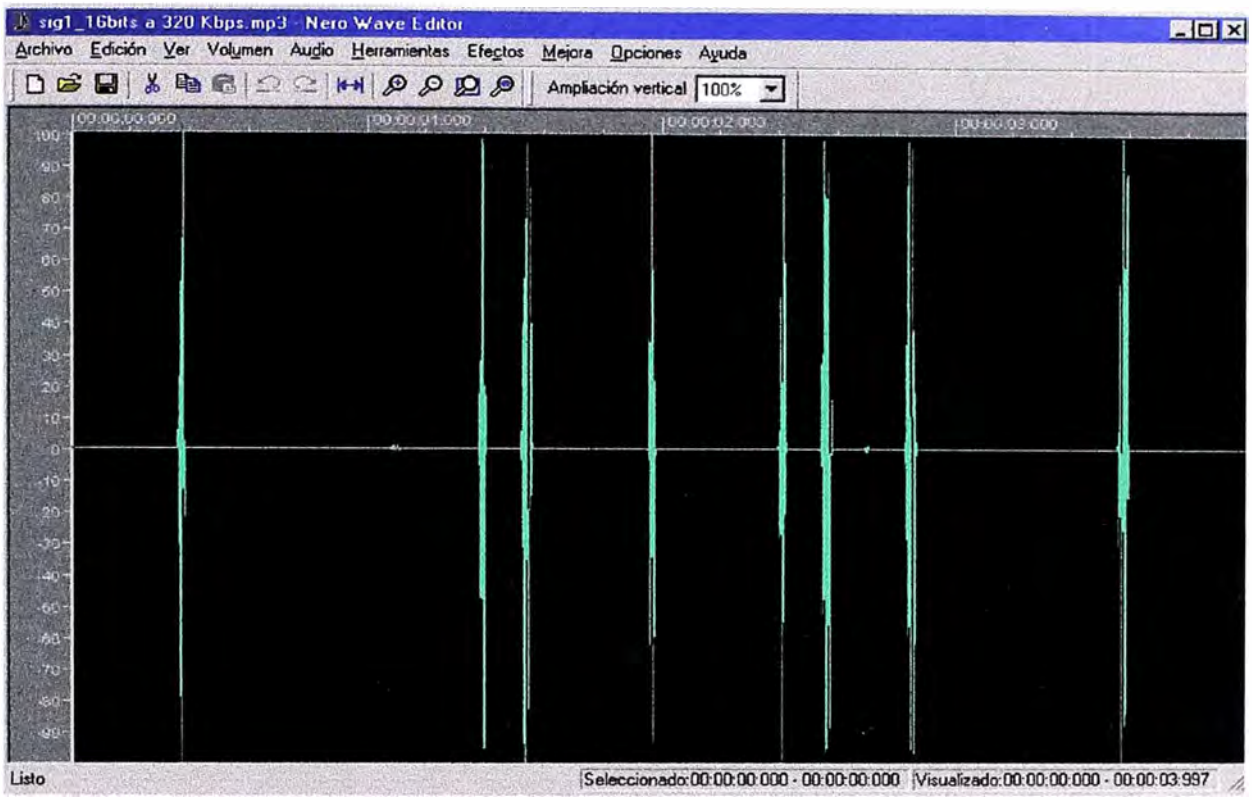


Figura PS17: **sig1_16bits a 96 Kbps_f=80_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

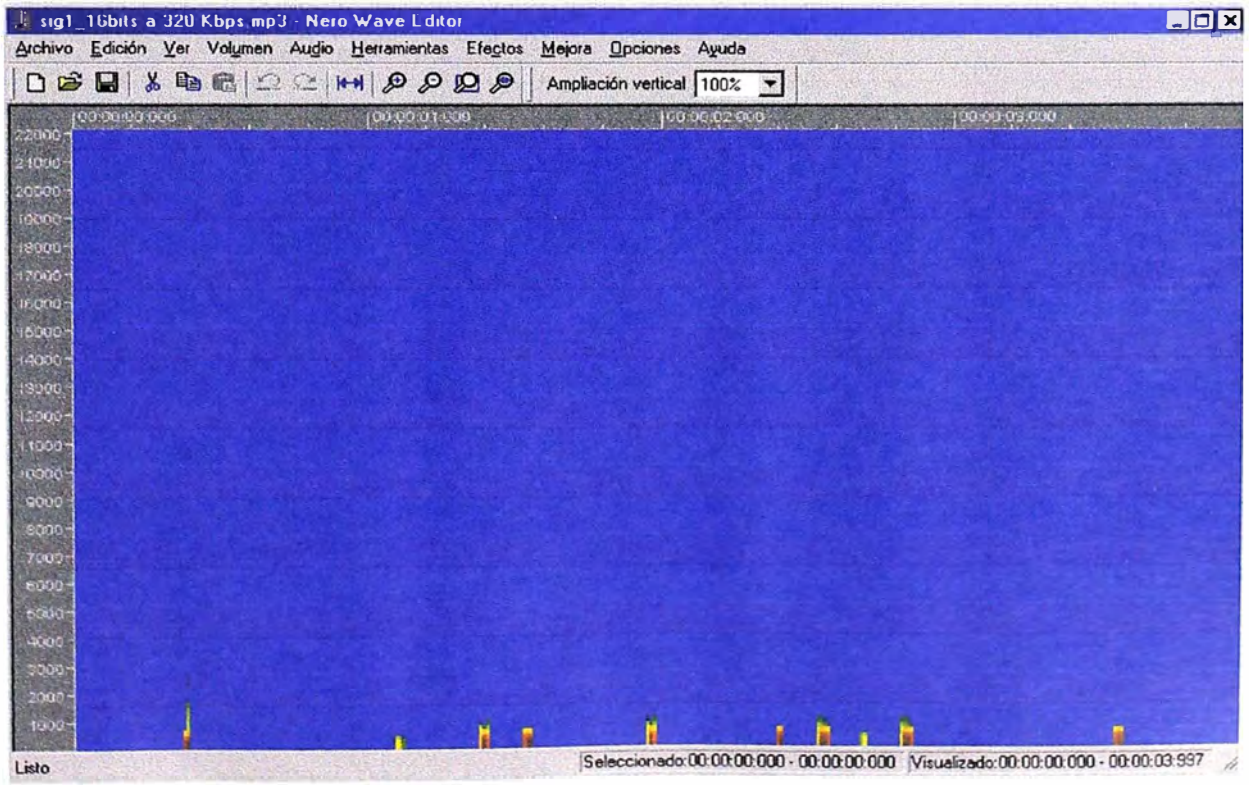
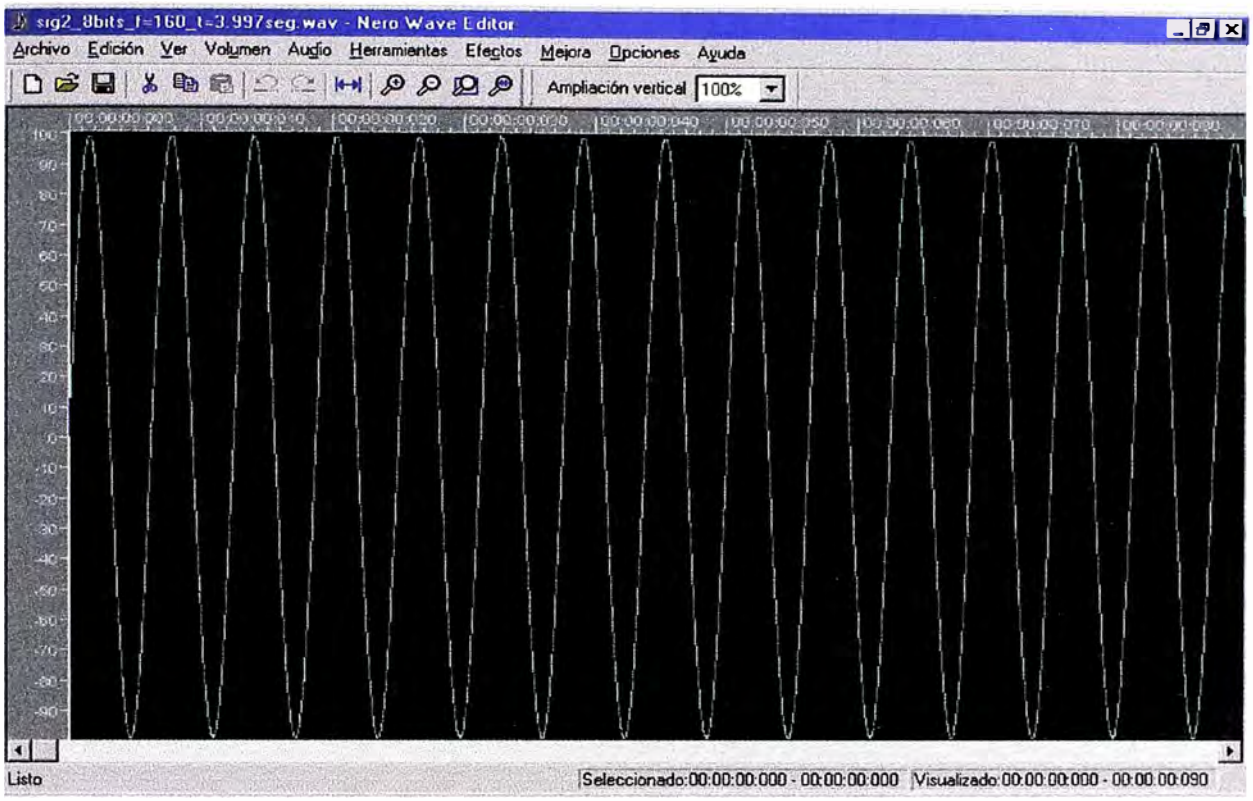


Figura PS18: **sig1_16bits a 320 Kbps_f=80_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

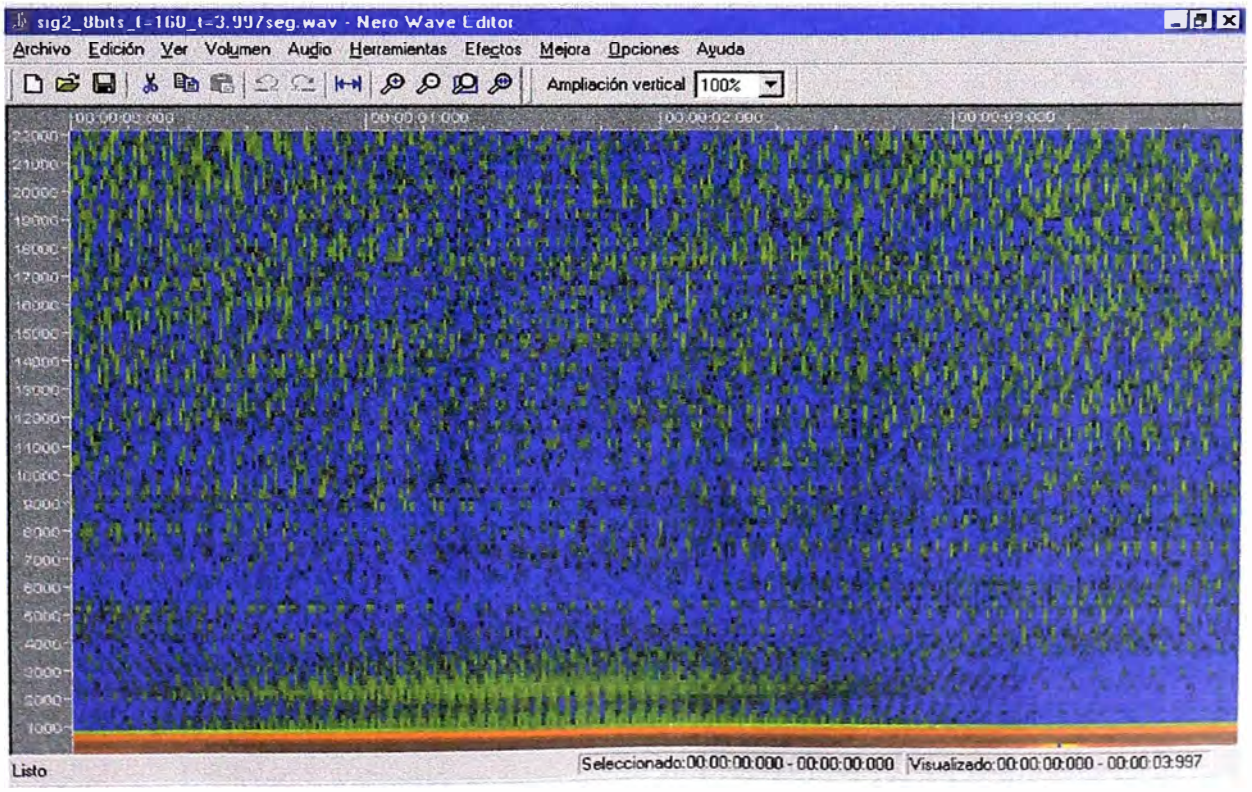
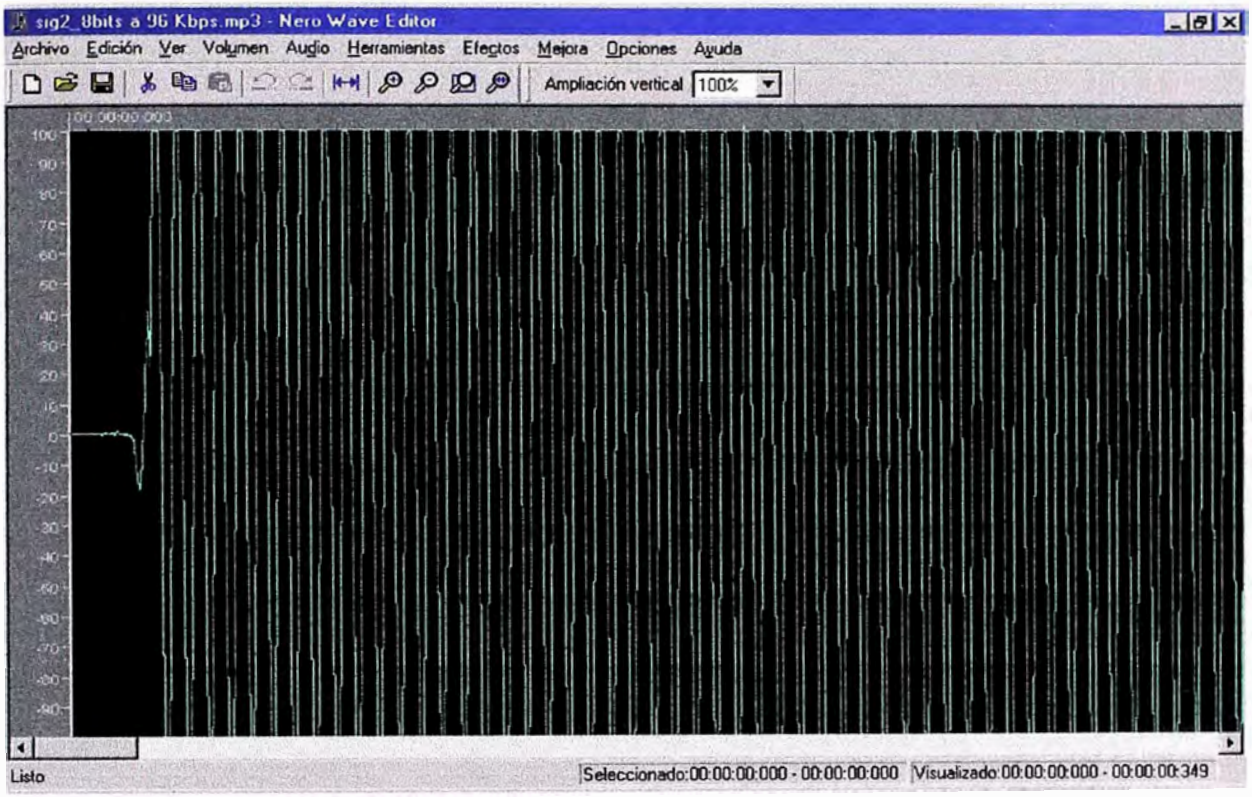


Figura PS19: **sig2_8bits_f=160_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

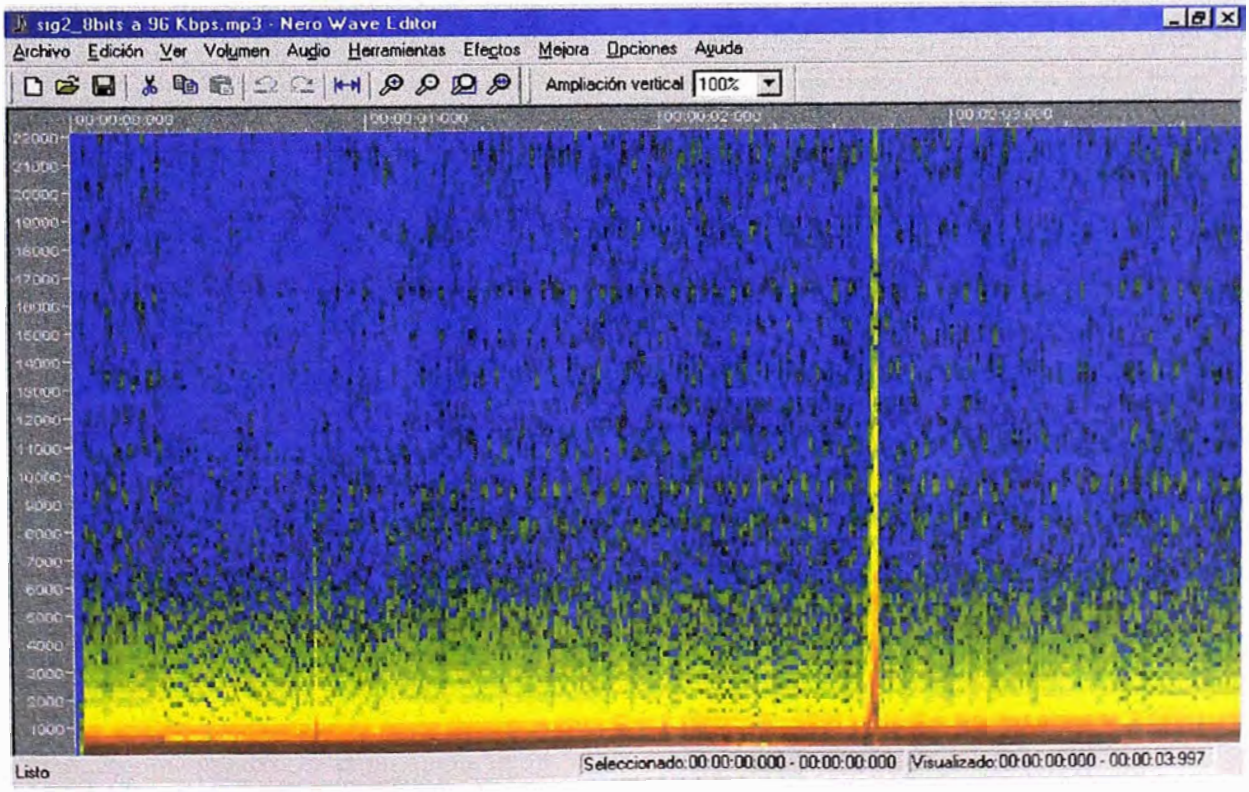
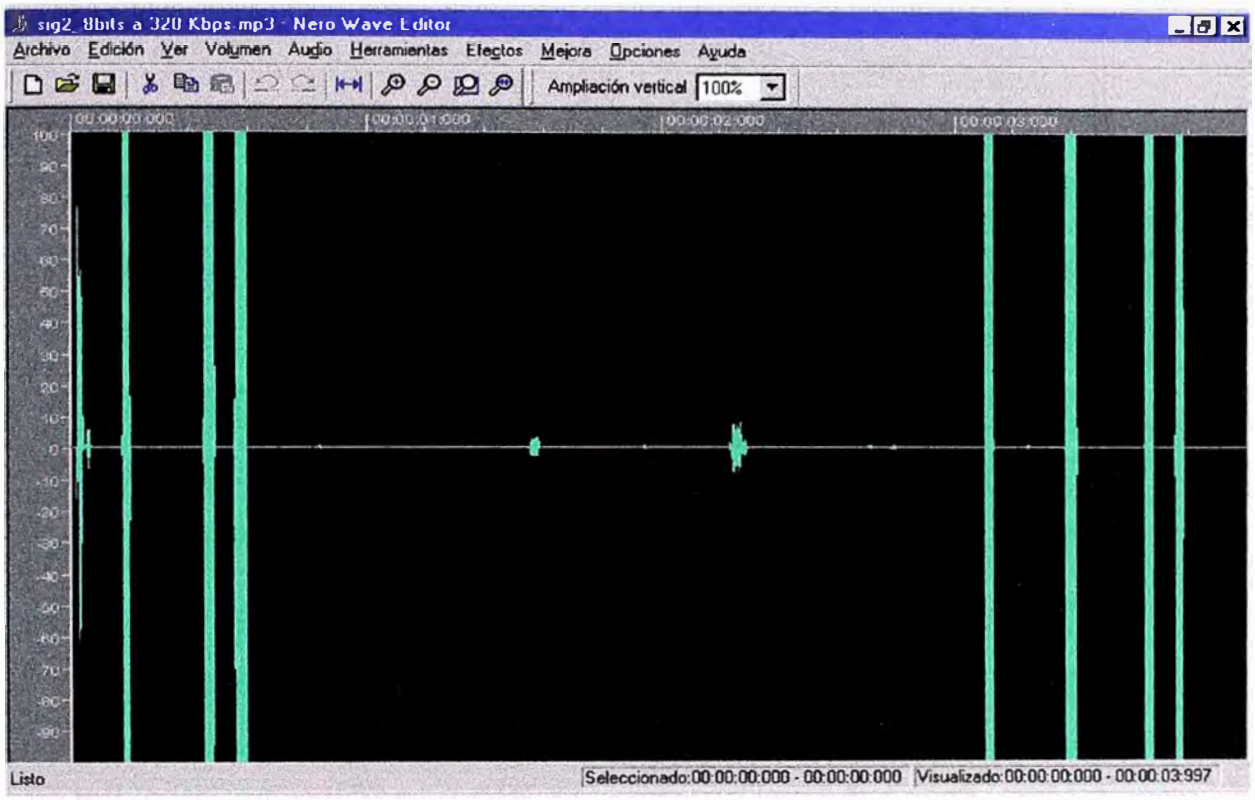


Figura PS20: **sig2_8bits a 96 Kbps_f=160_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

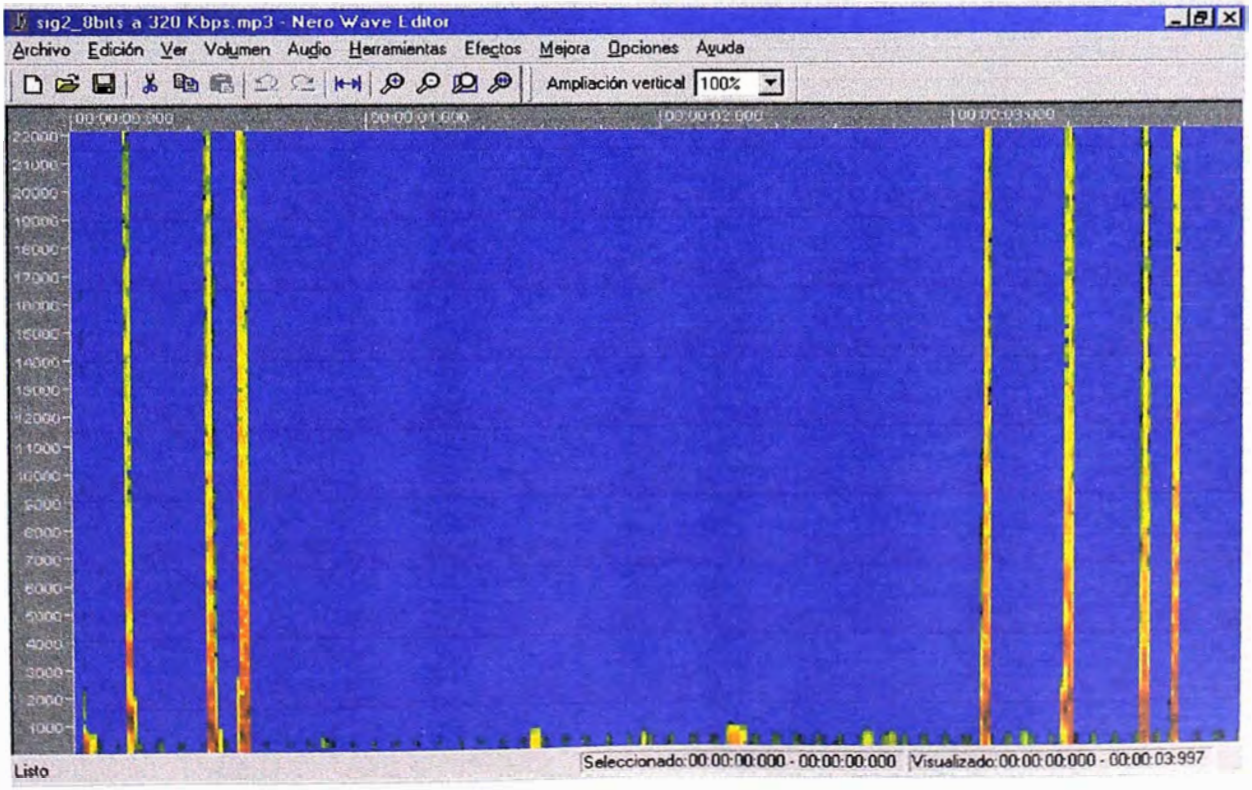
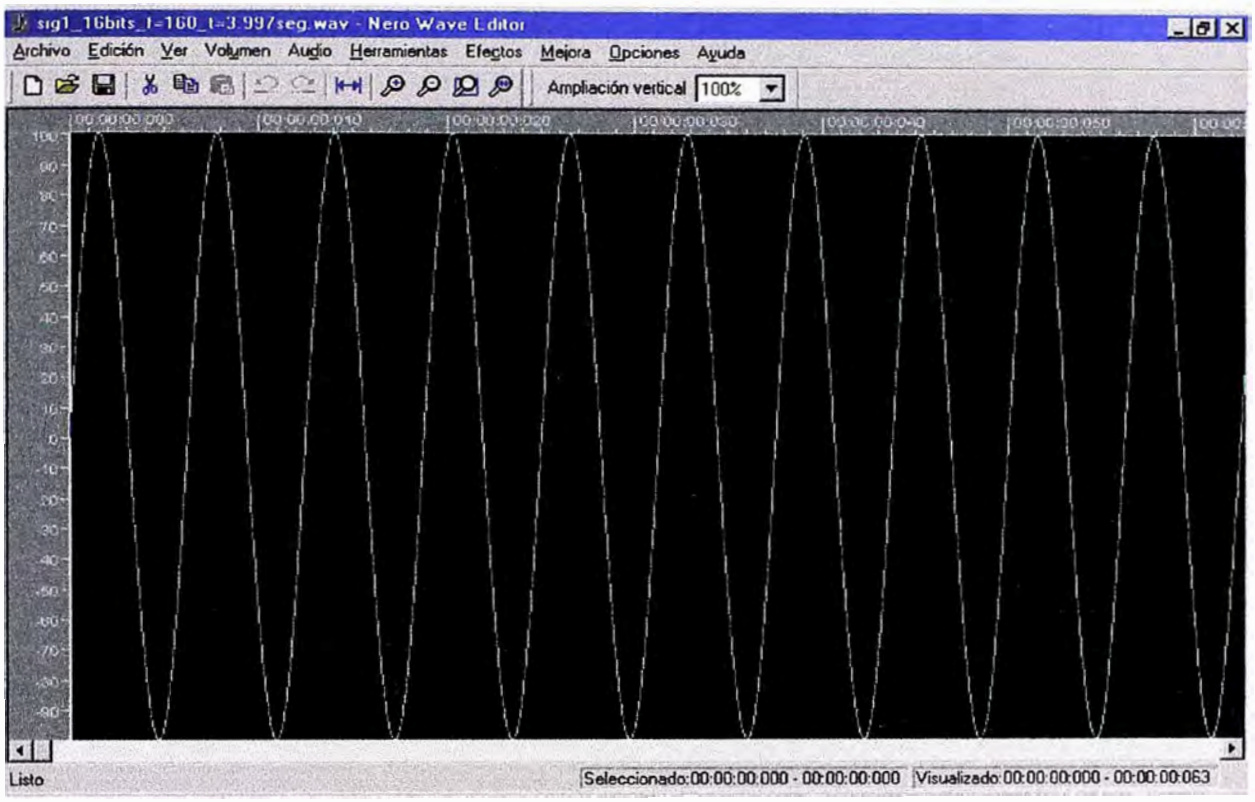


Figura PS21: **sig2_8bits a 320 Kbps_f=160_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

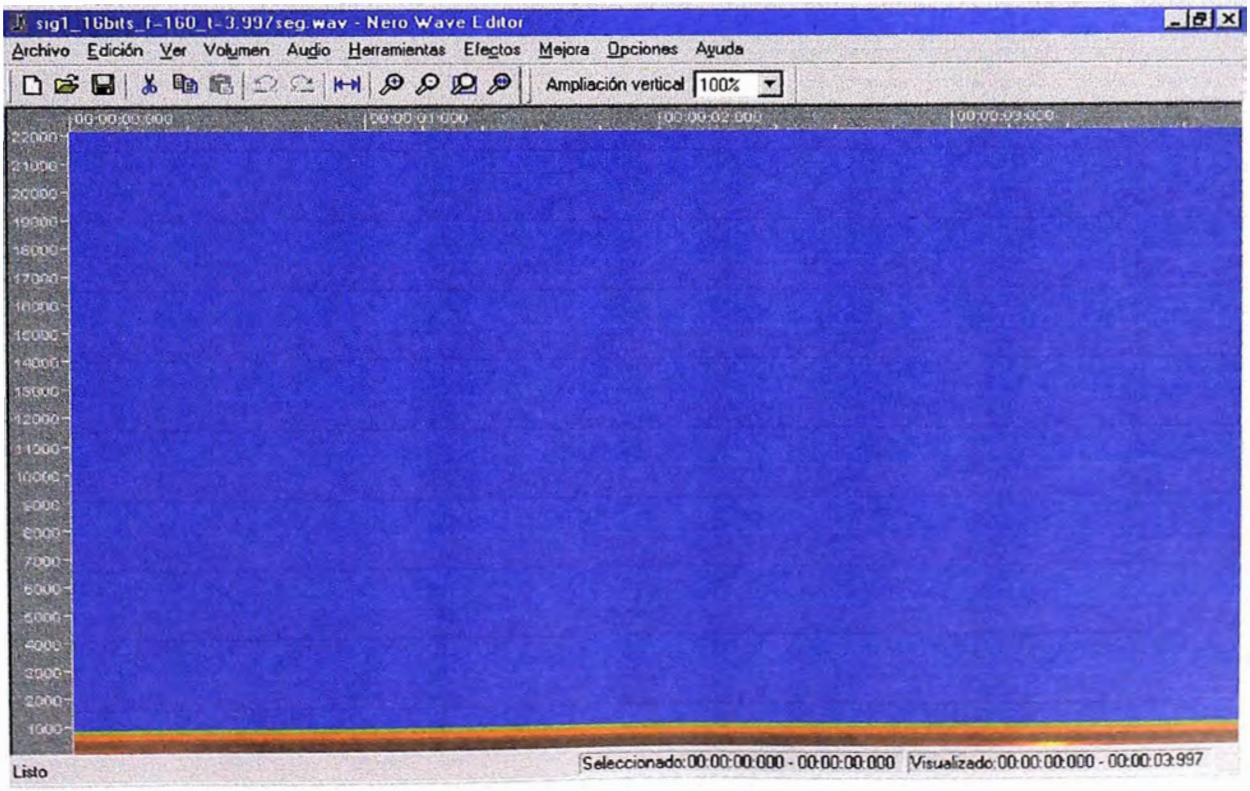
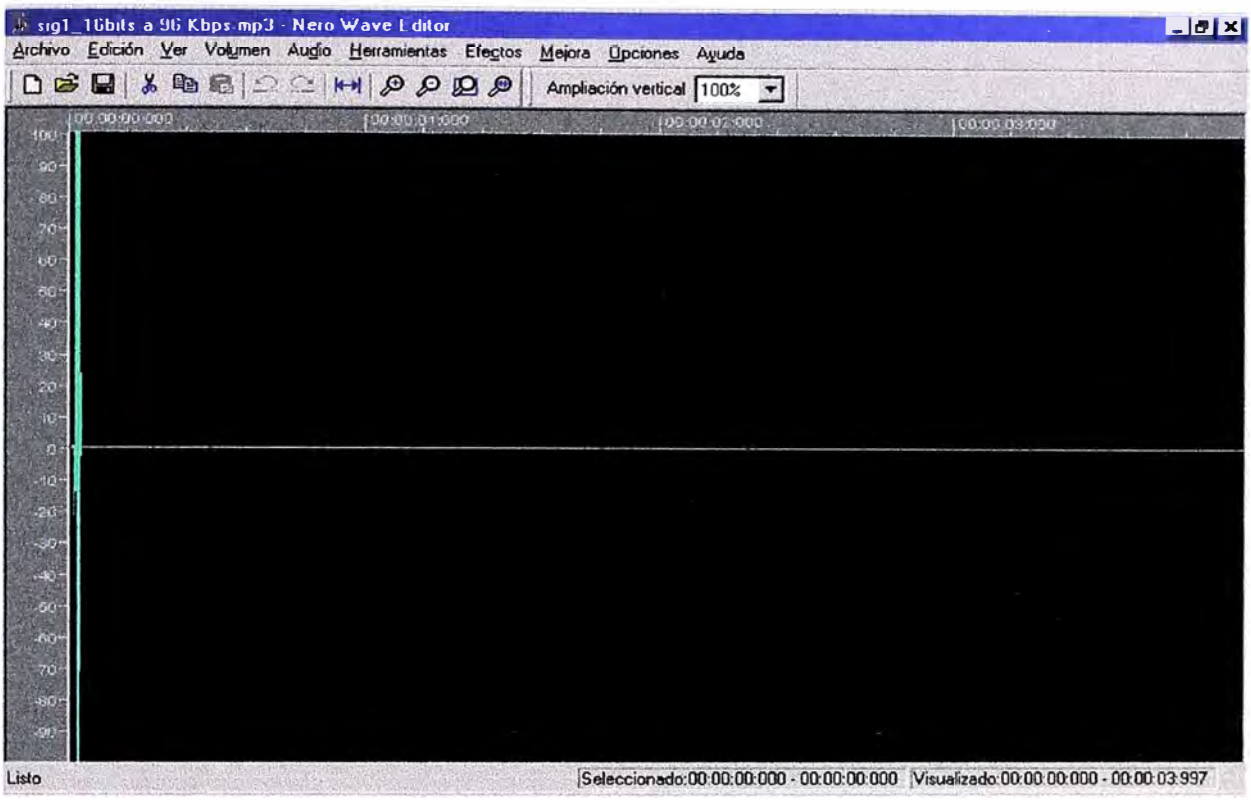


Figura PS22: **sig1_16bits_f=160_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

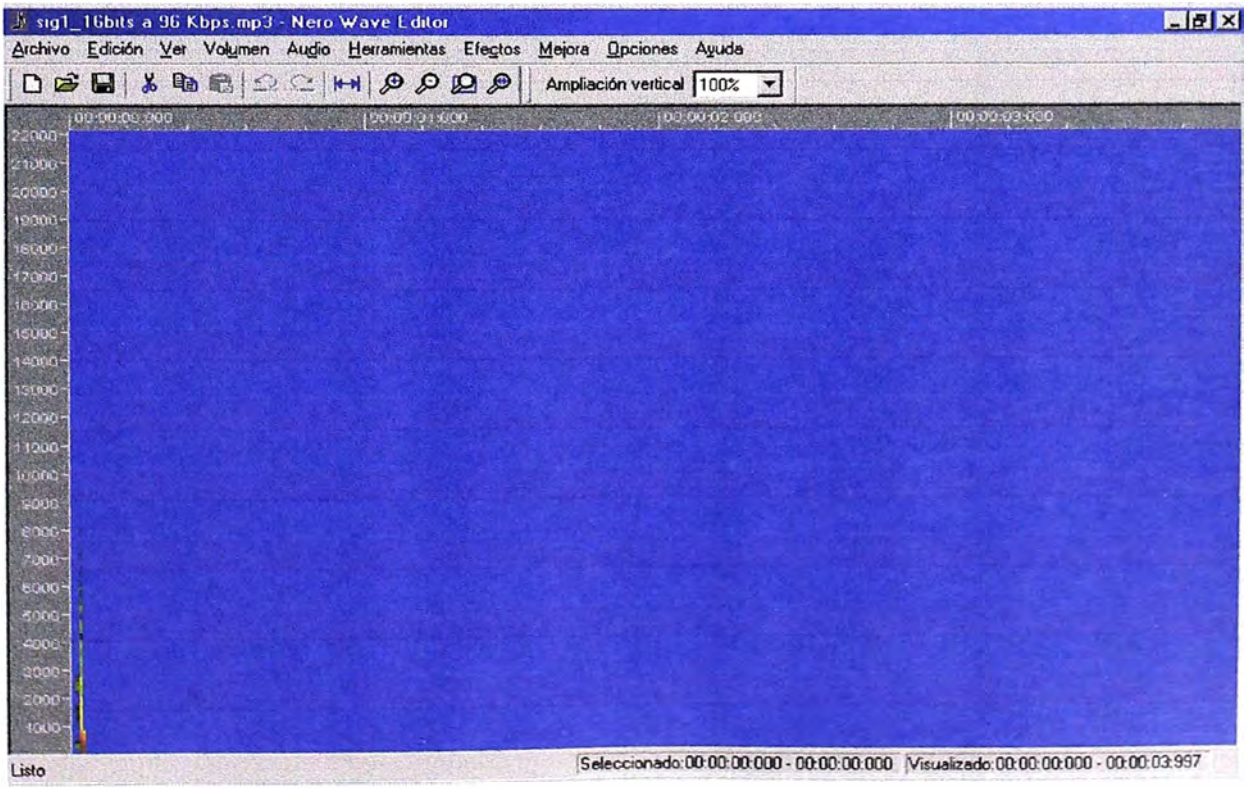
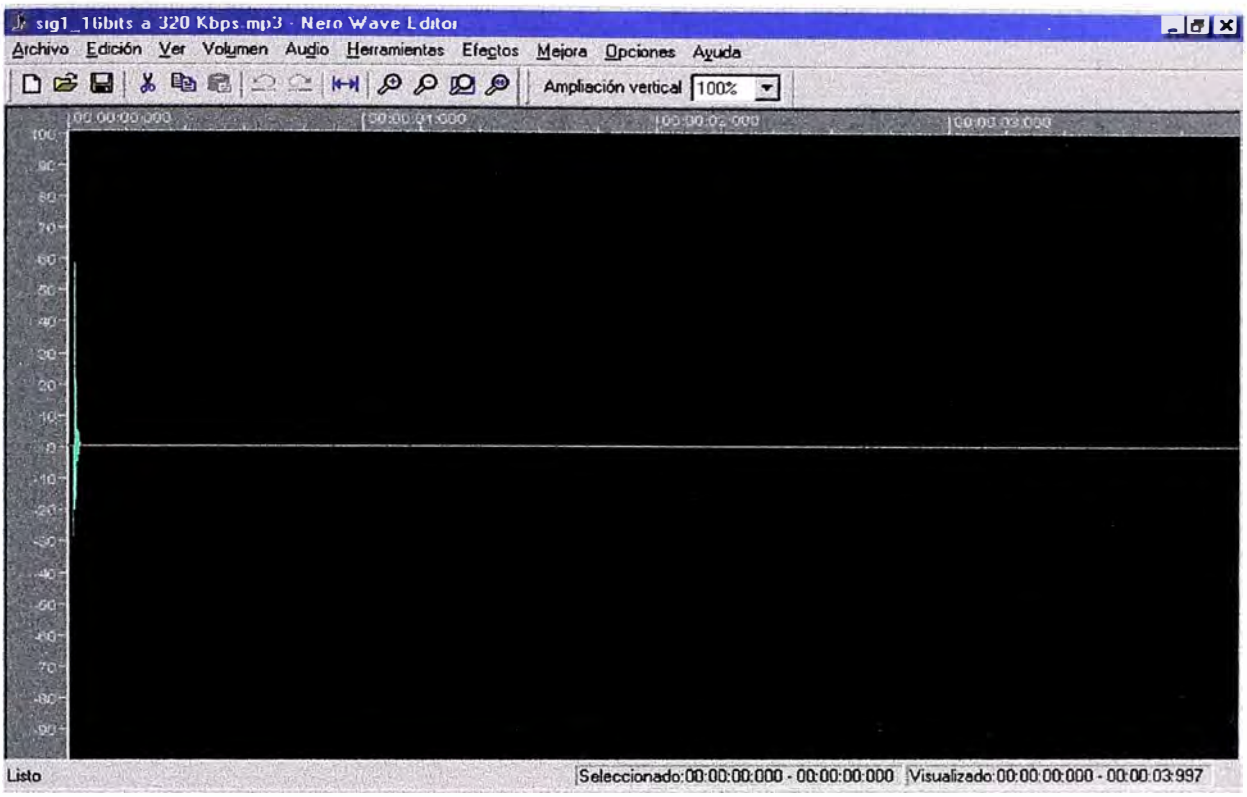


Figura PS23: **sig1_16bits a 96 Kbps_f=160_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

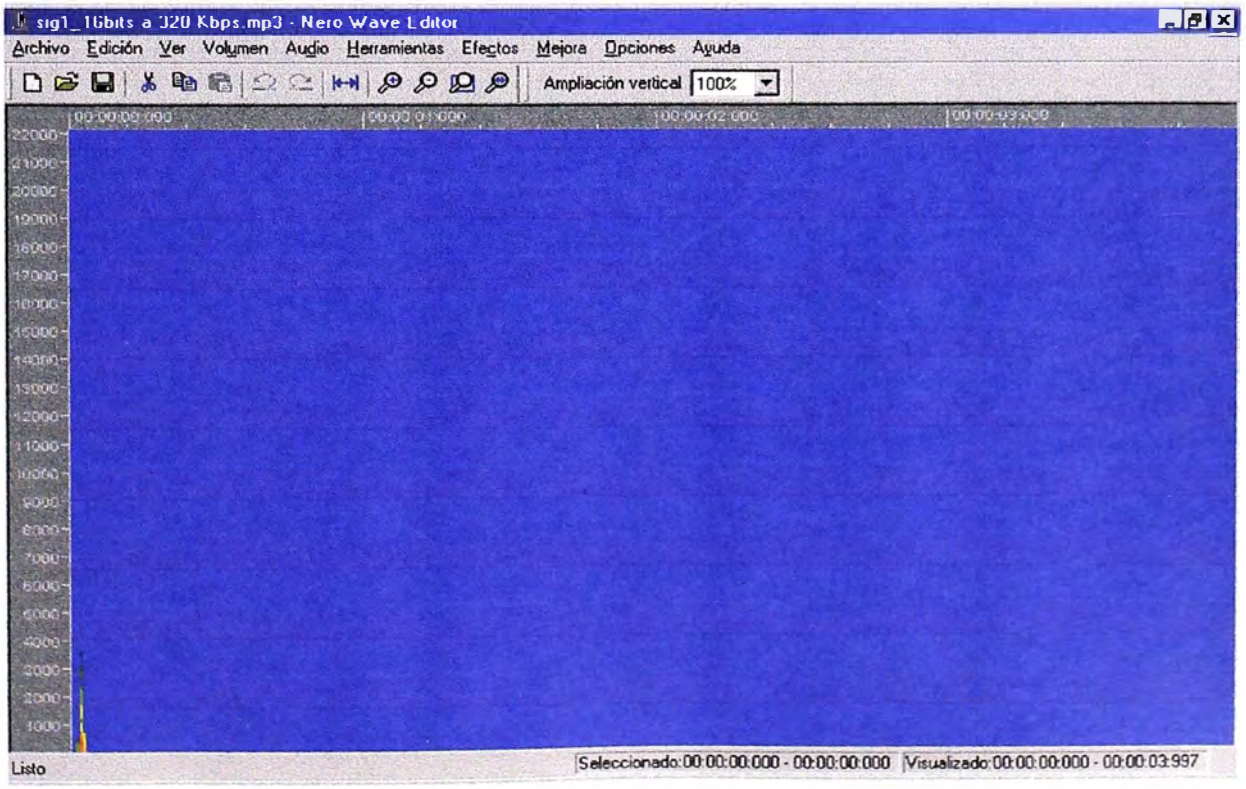
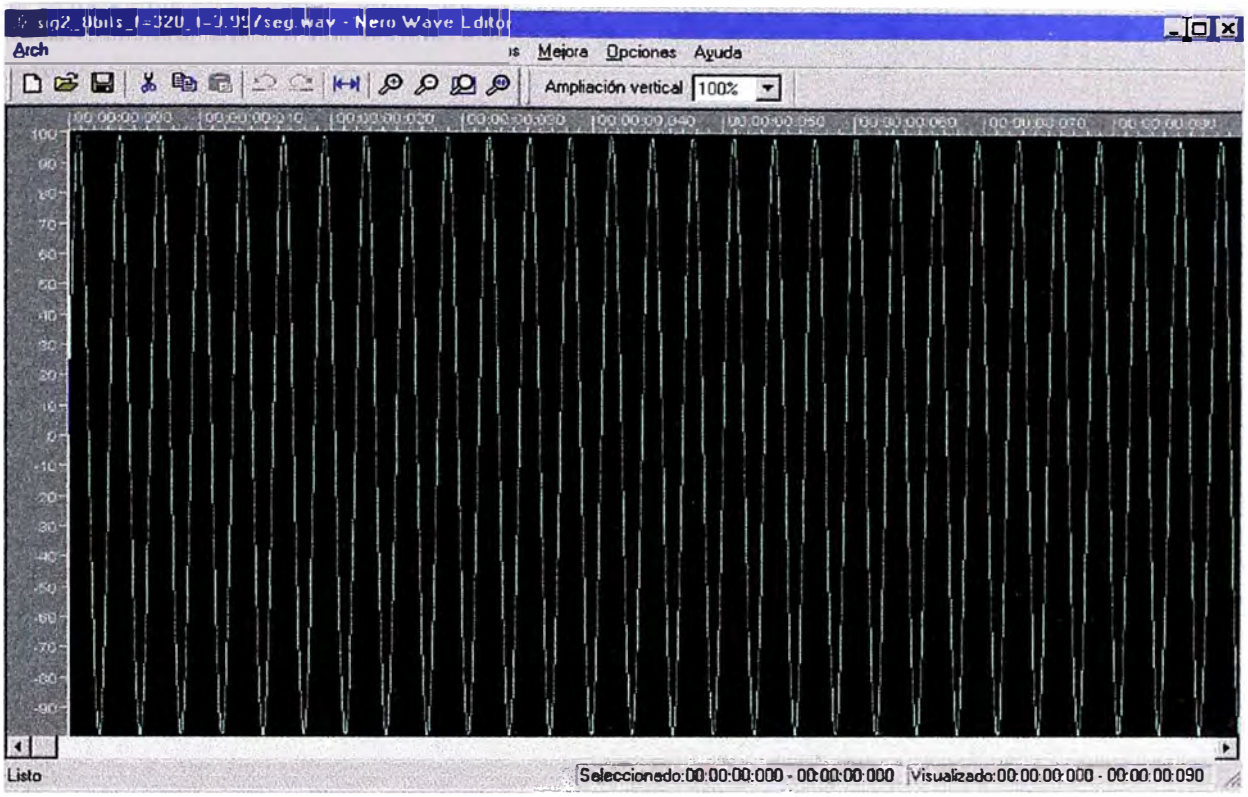


Figura PS24: **sig1_16bits a 320 Kbps_f=160_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

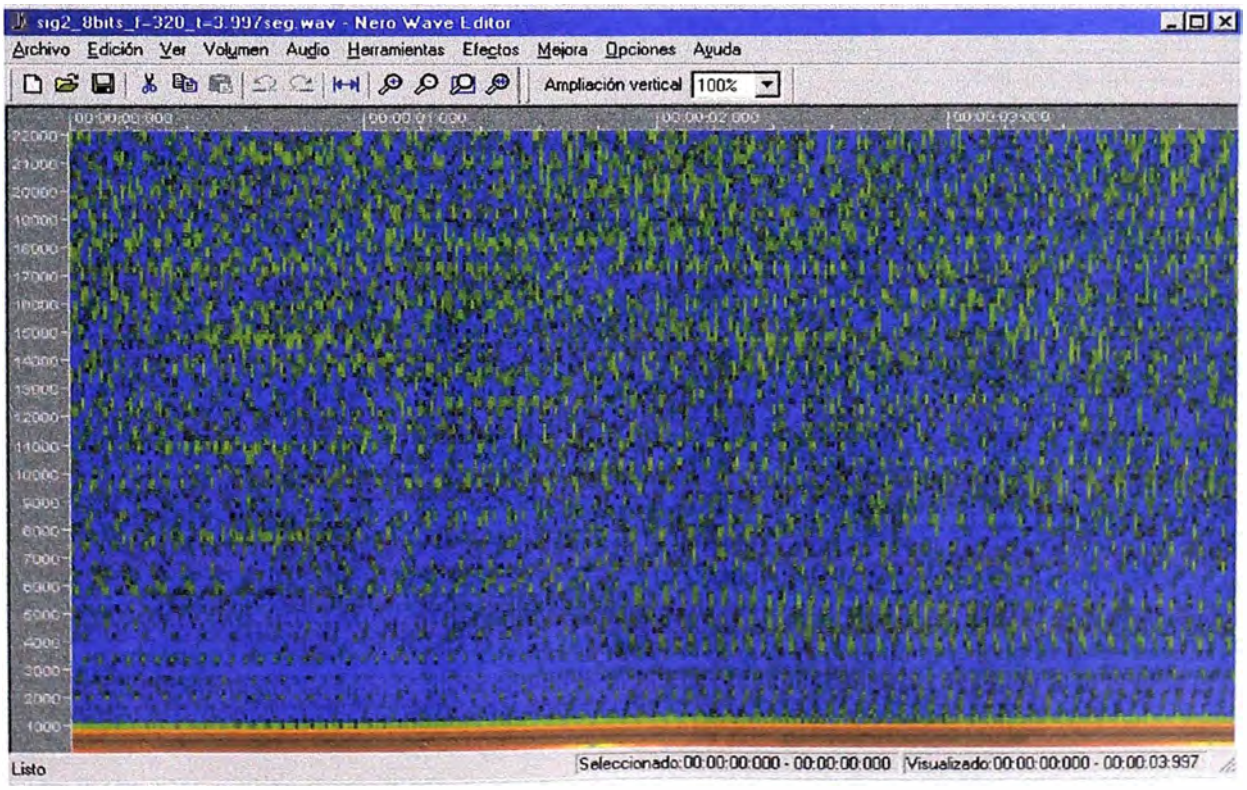
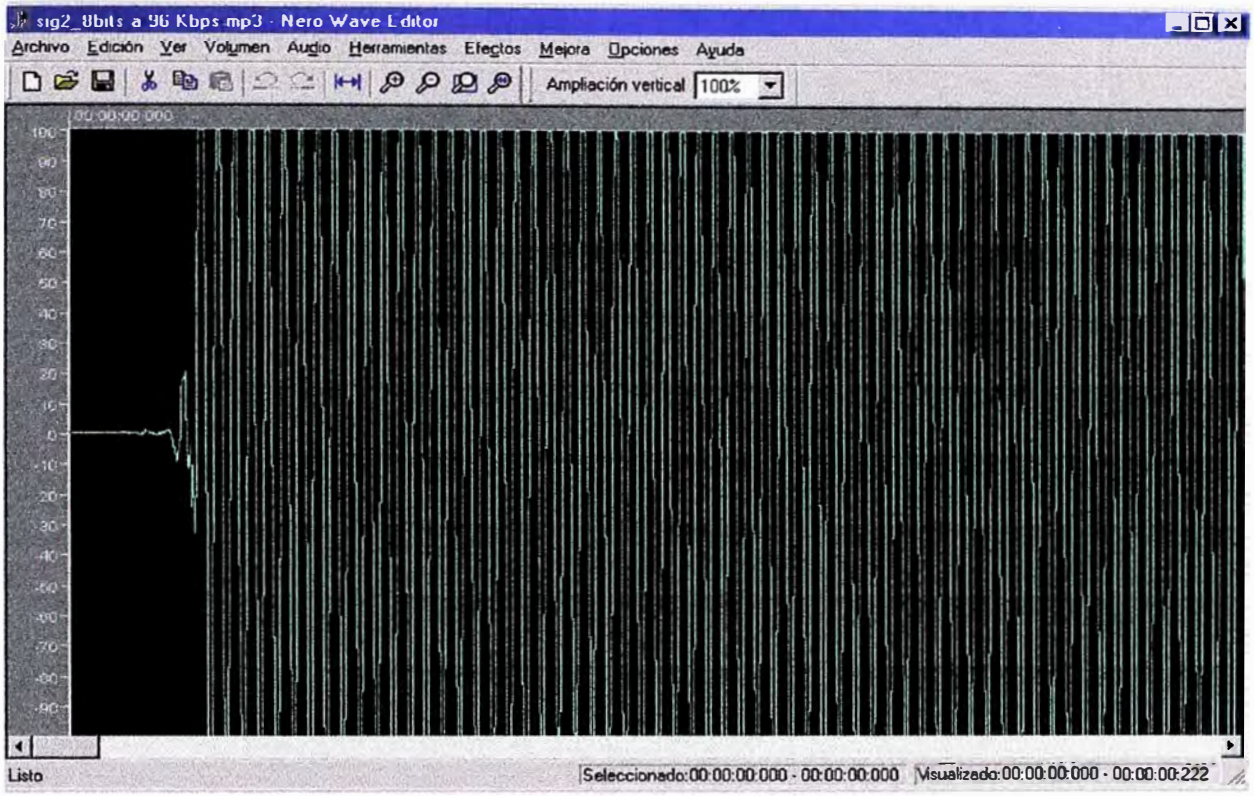


Figura PS25: **sig2_8bits_f=320_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

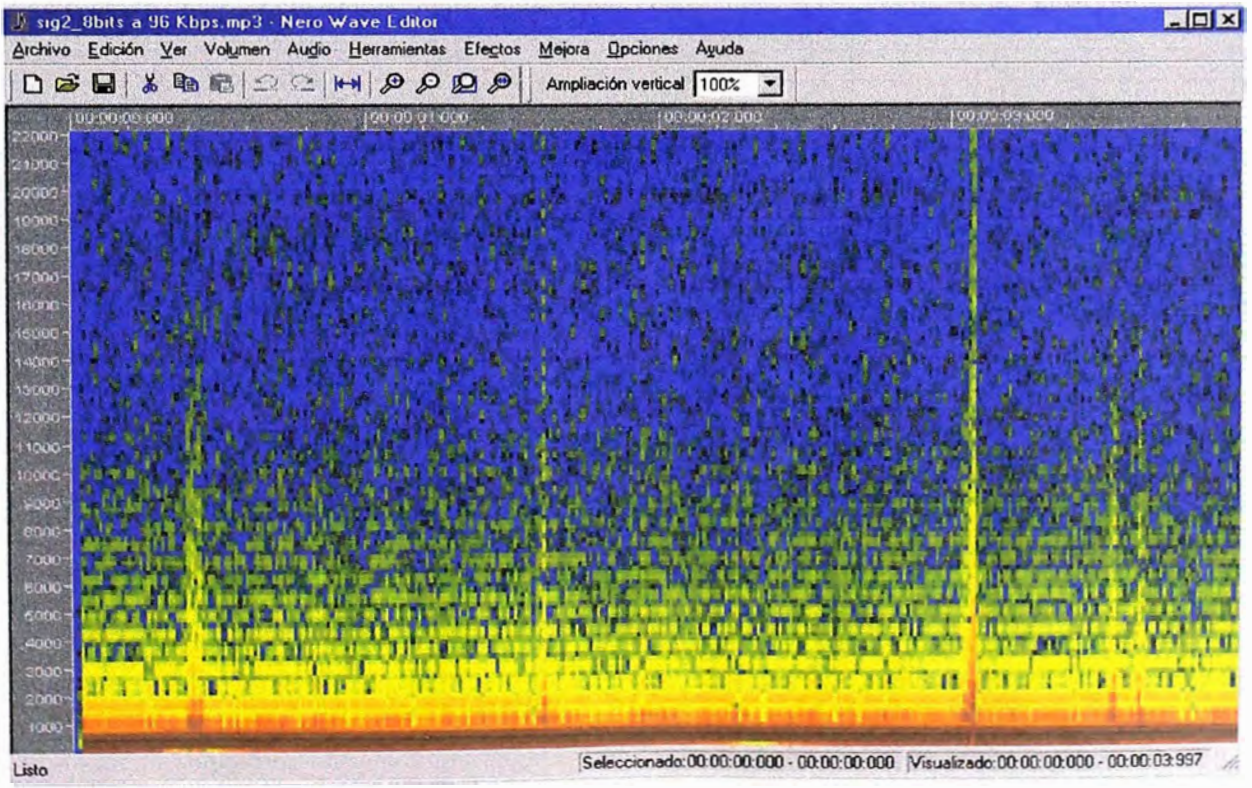
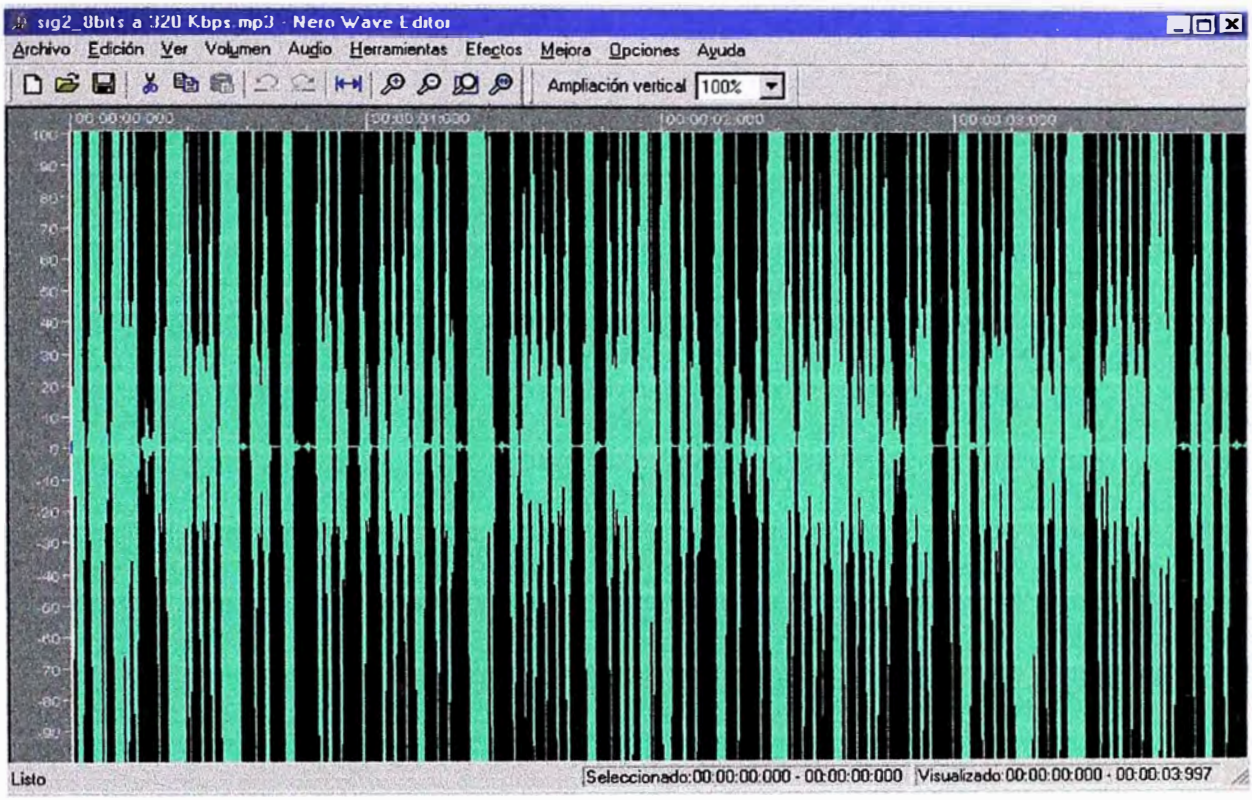


Figura PS26: **sig2_8bits a 96 Kbps_f=320_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

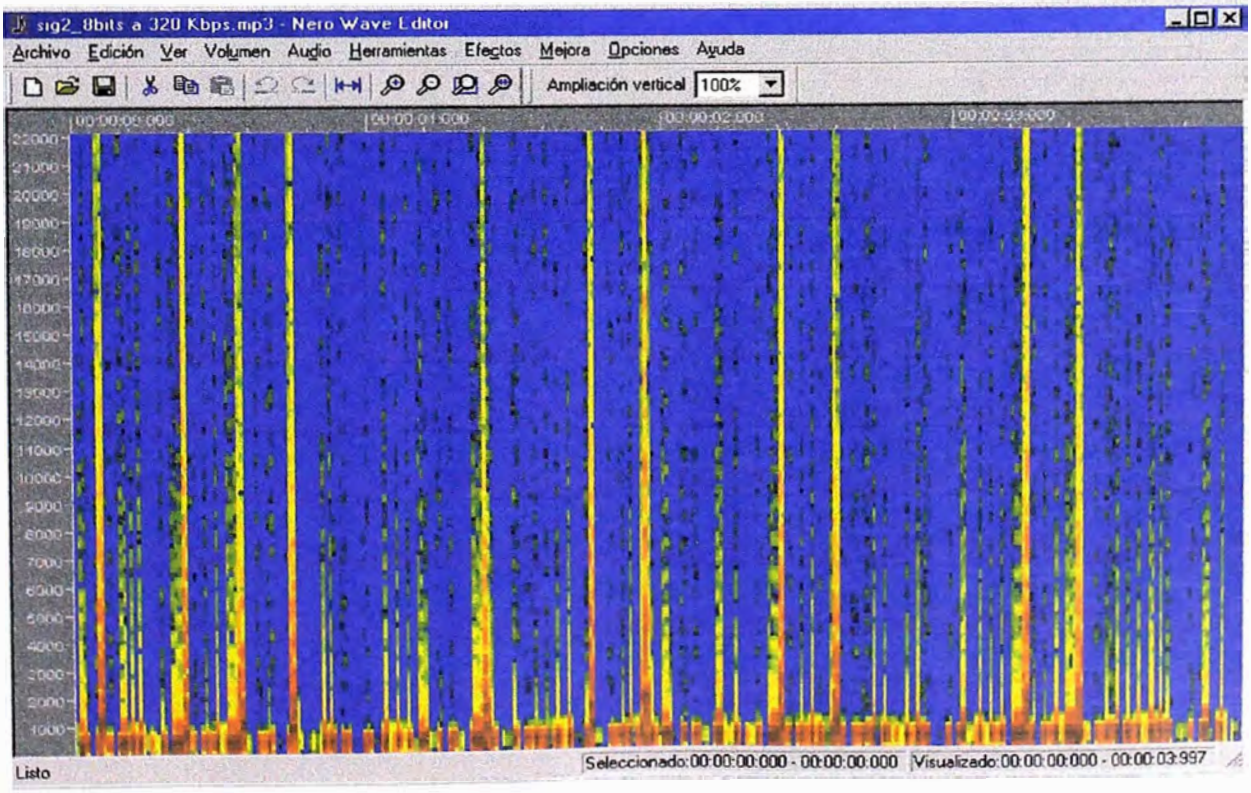
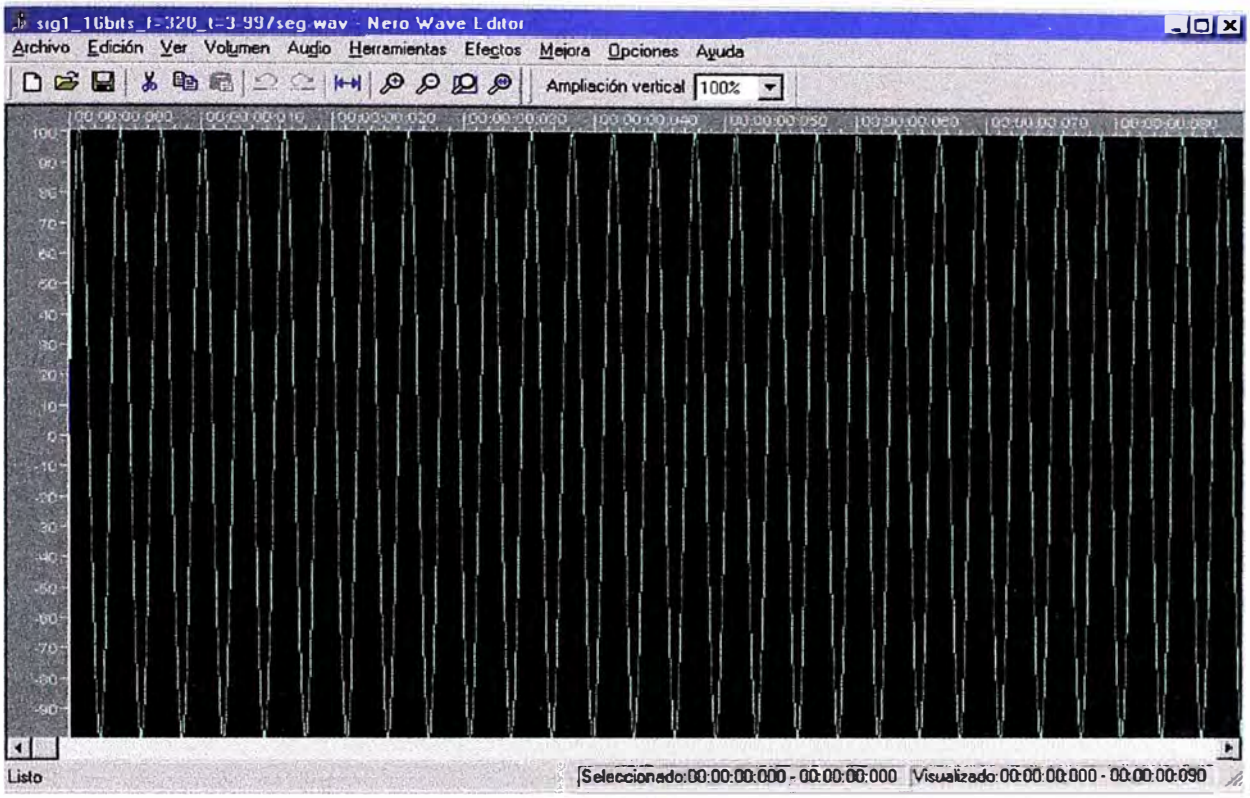


Figura PS27: **sig2_8bits a 320 Kbps_f=320_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

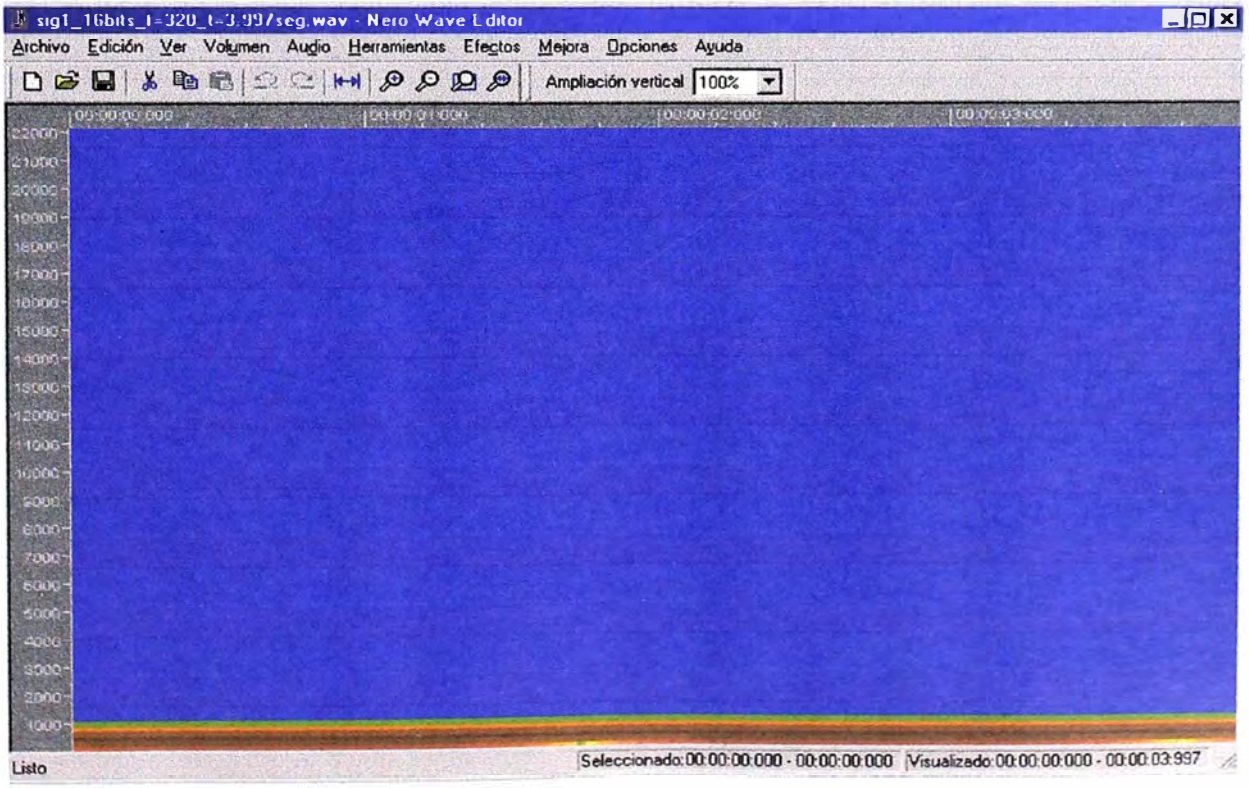
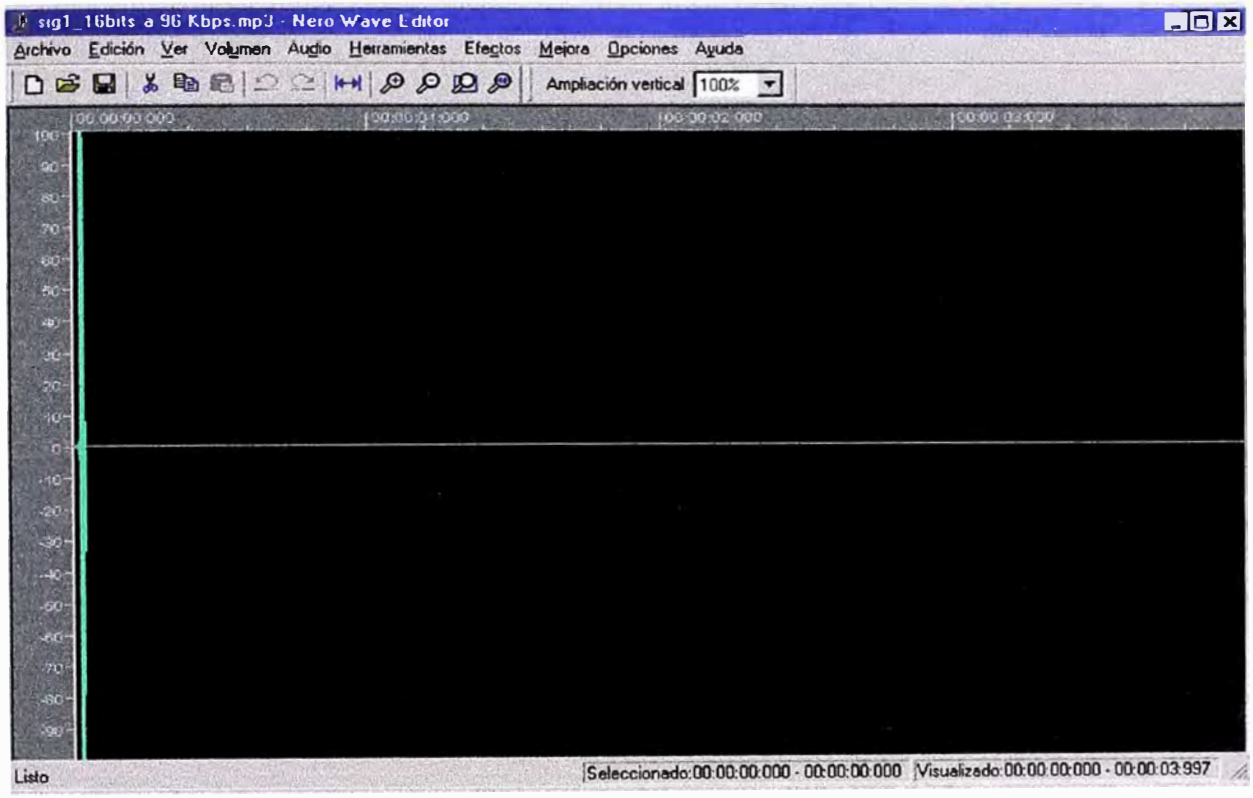


Figura PS28: **sig1_16bits_f=320_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

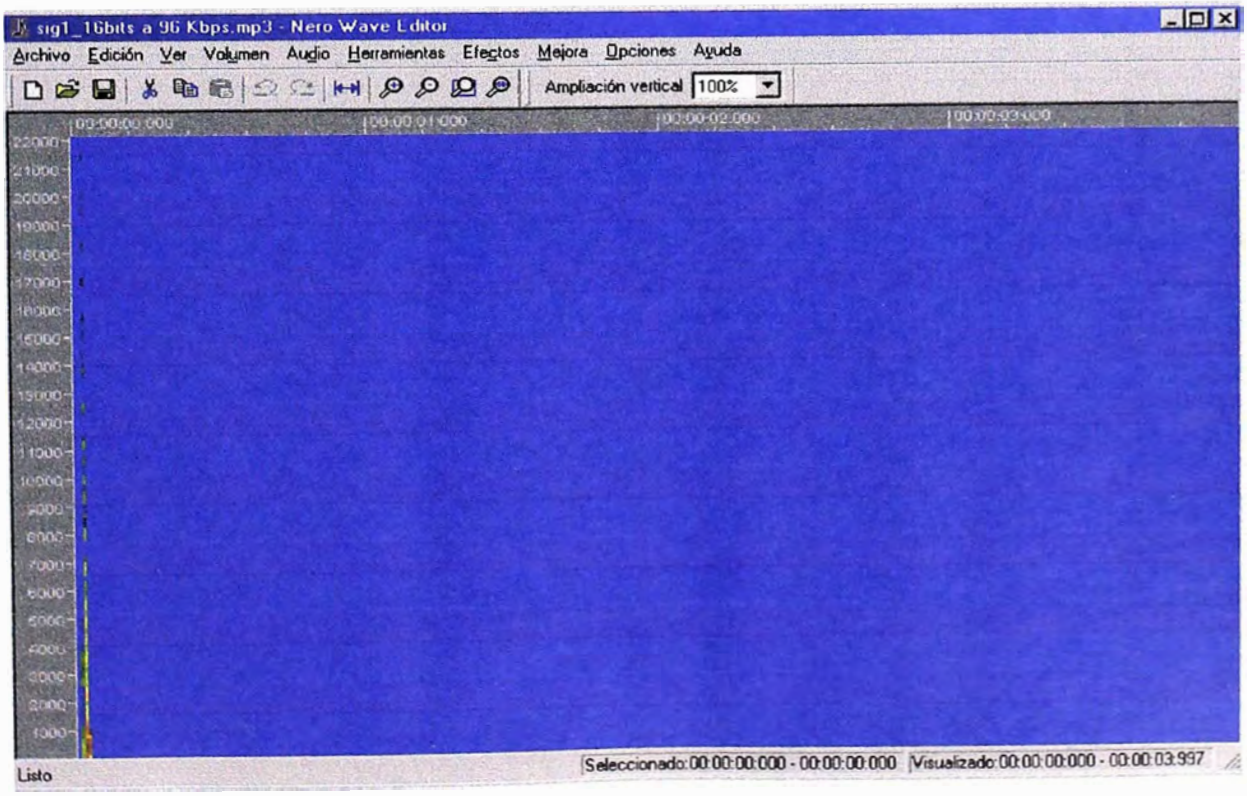
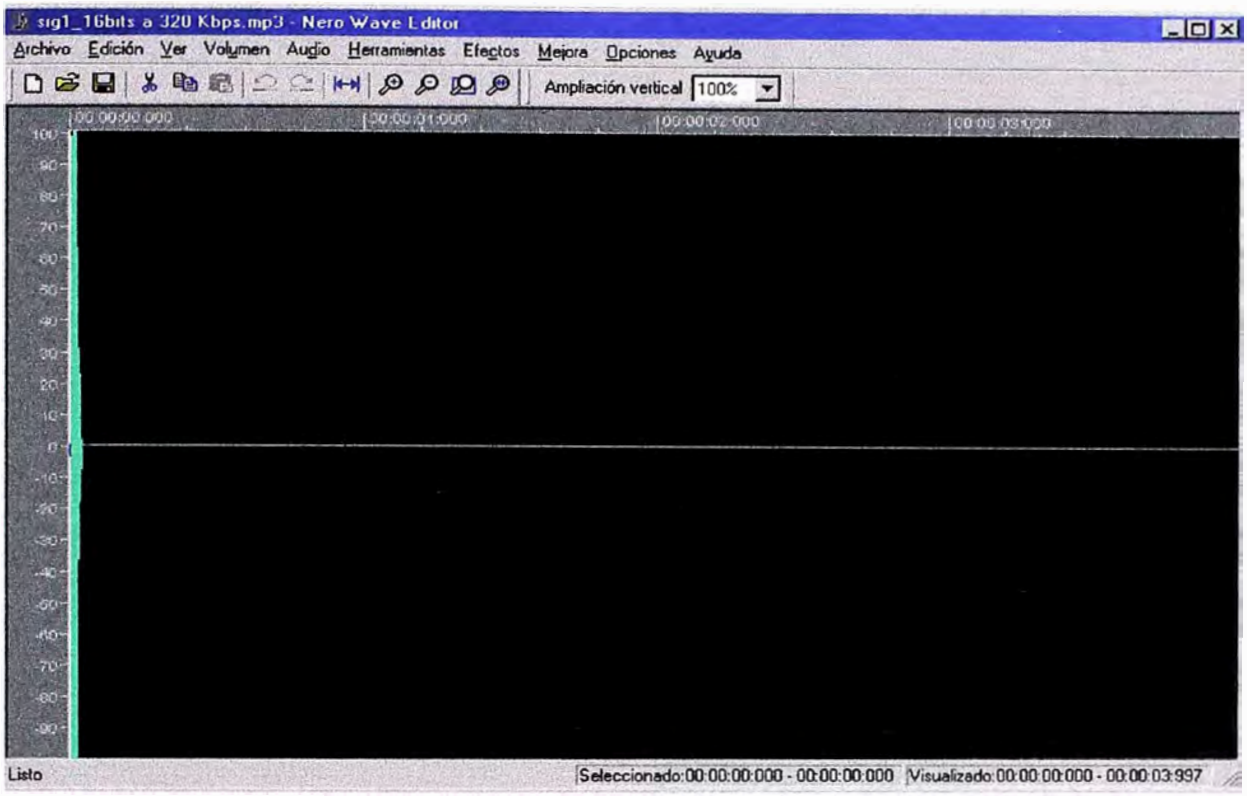


Figura PS29: **sig1_16bits a 96 Kbps_f=320_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

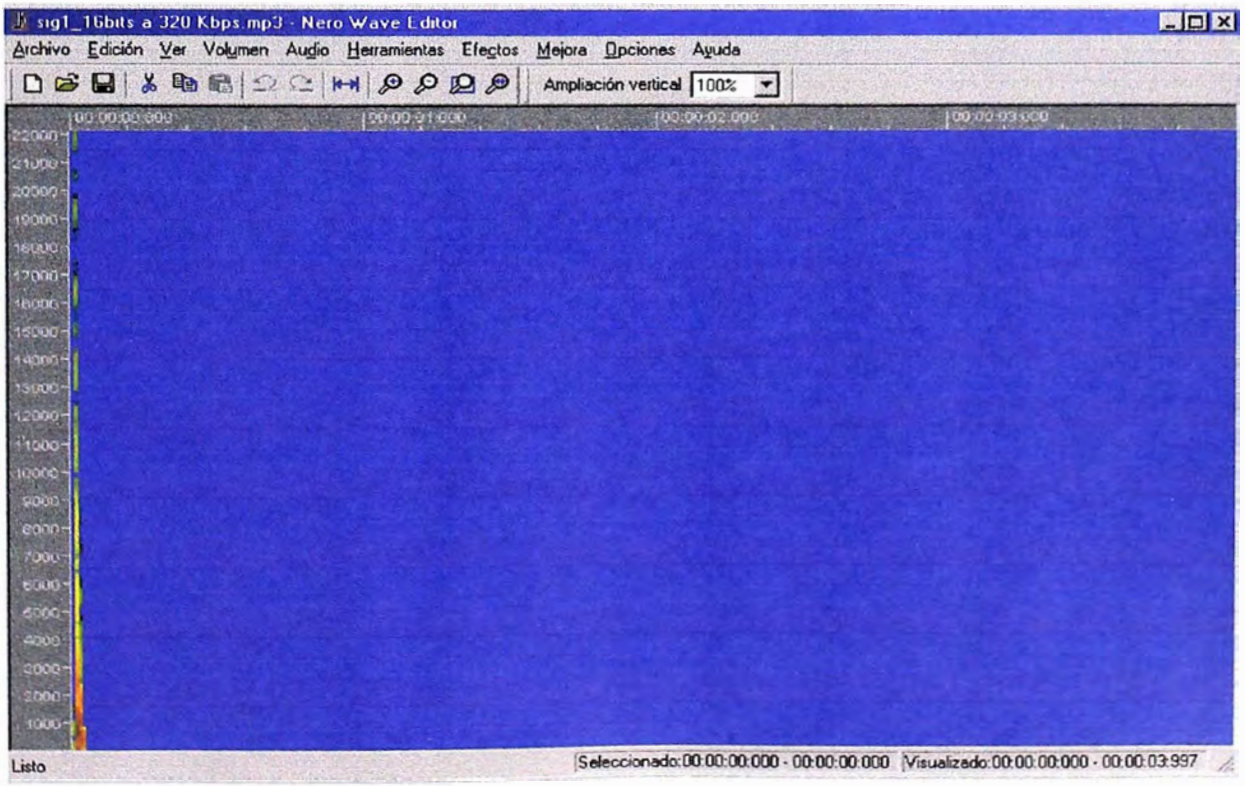
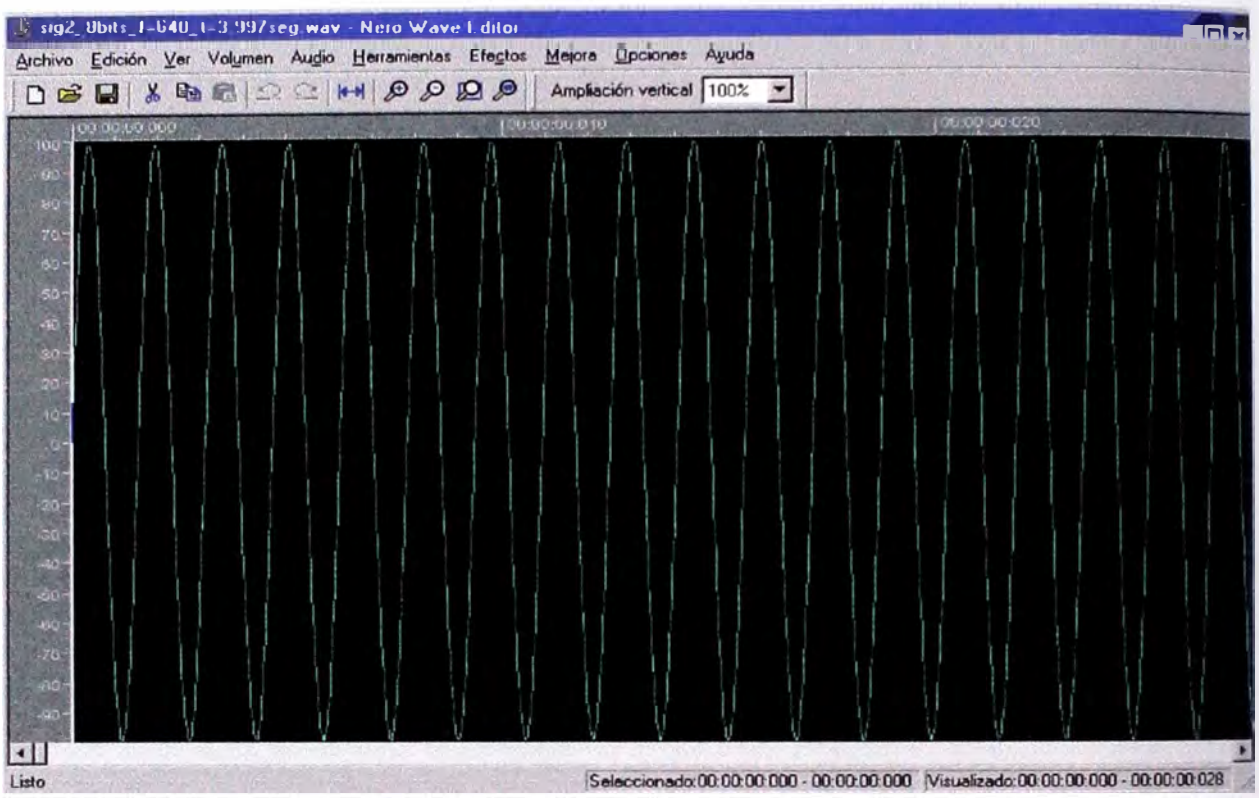


Figura PS30: **sig1_16bits a 320 Kbps_f=320_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

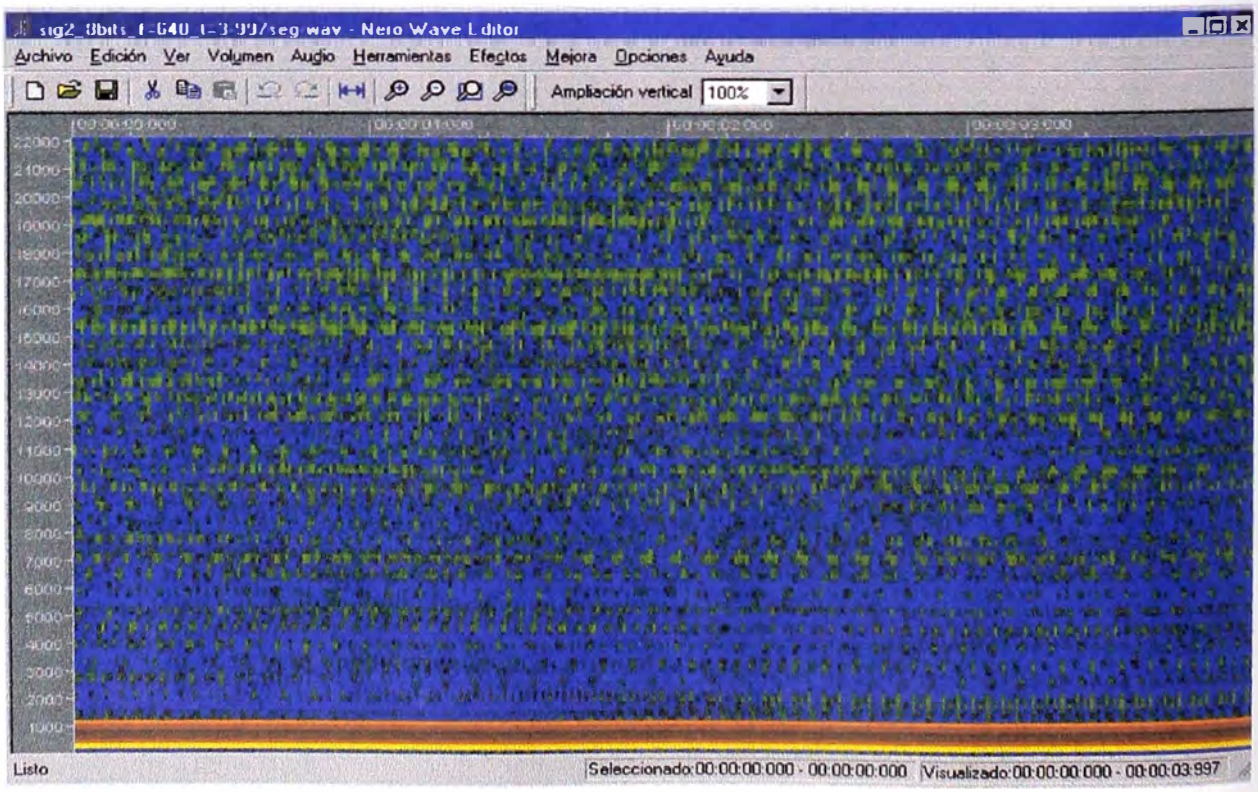
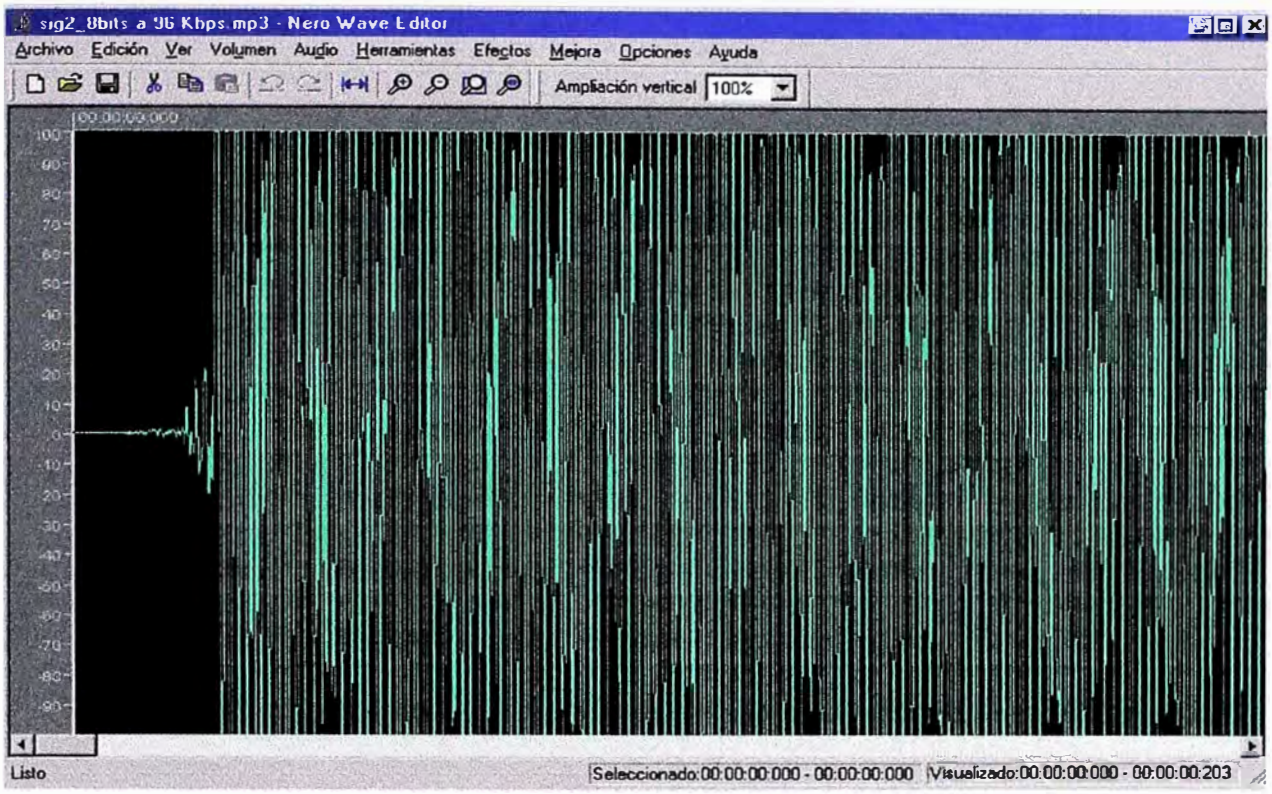


Figura PS31: **sig2_8bits_f=640_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

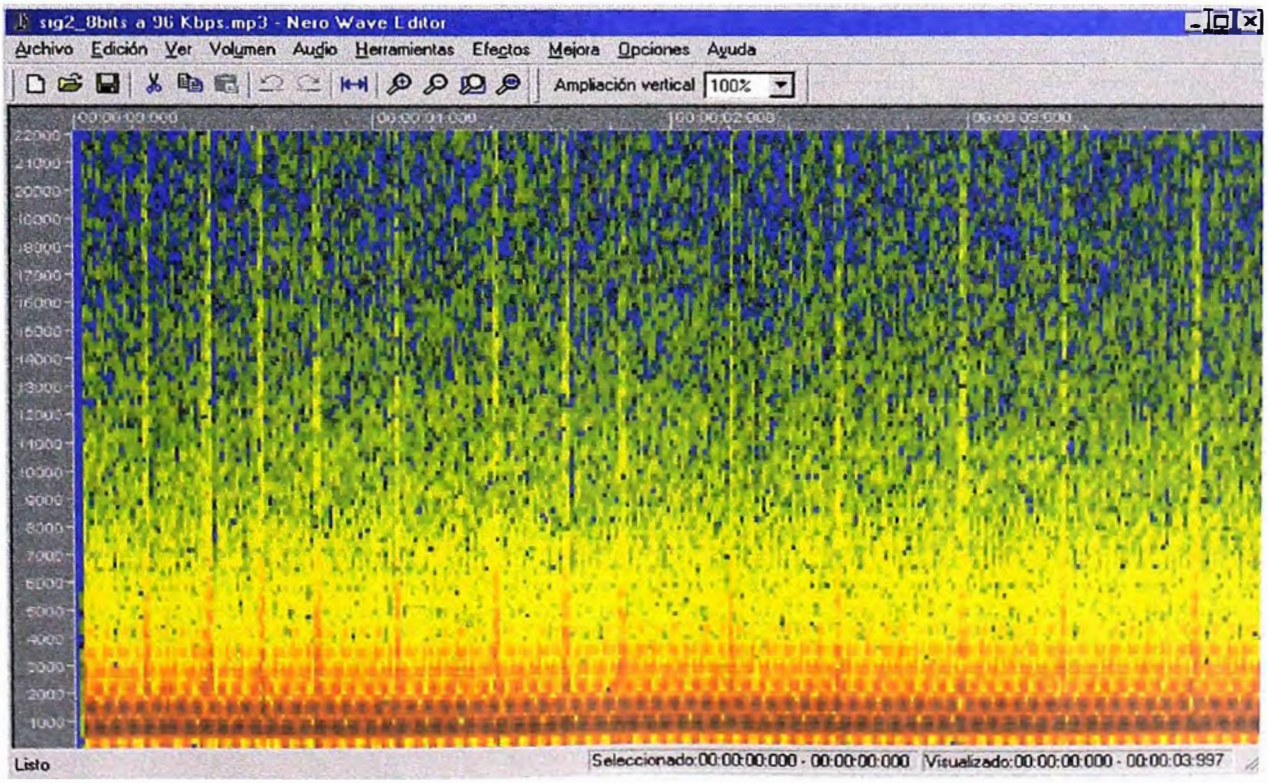
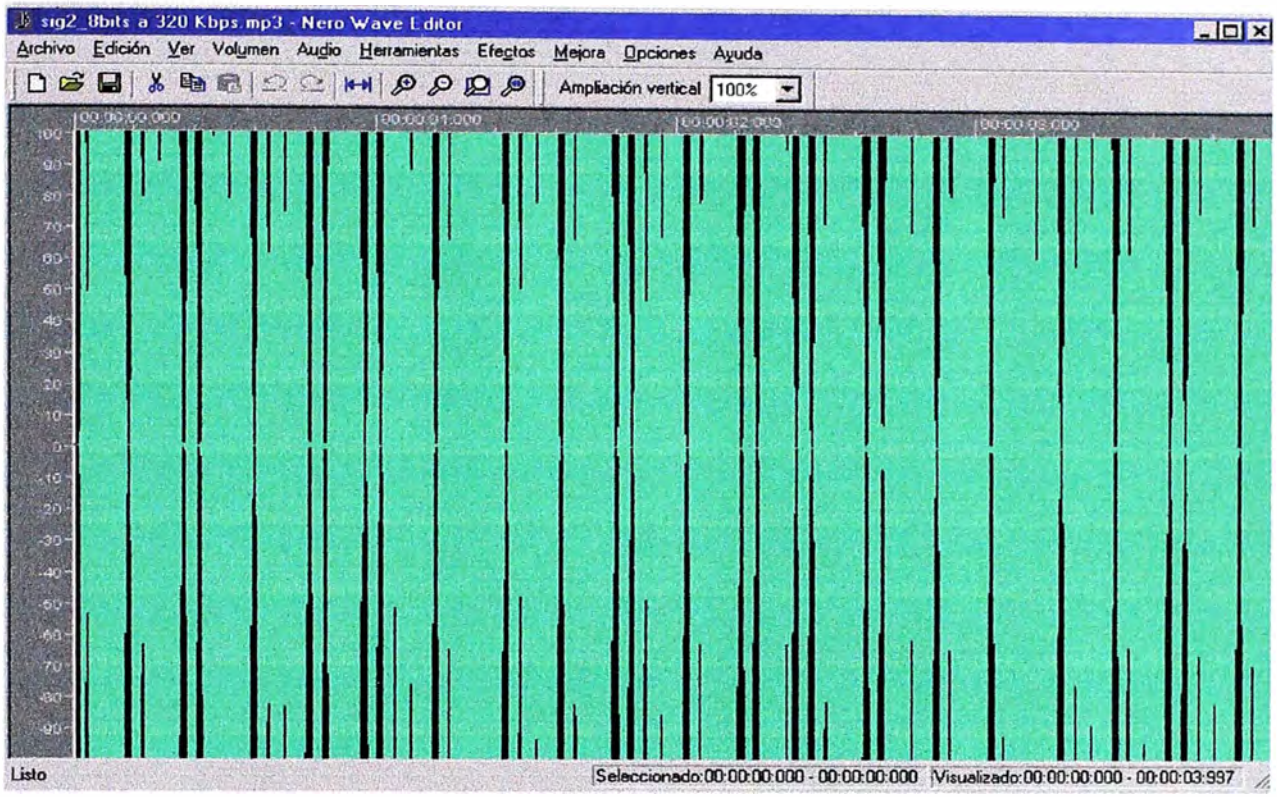


Figura PS32: **sig2_8bits a 96 Kbps_f=640_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

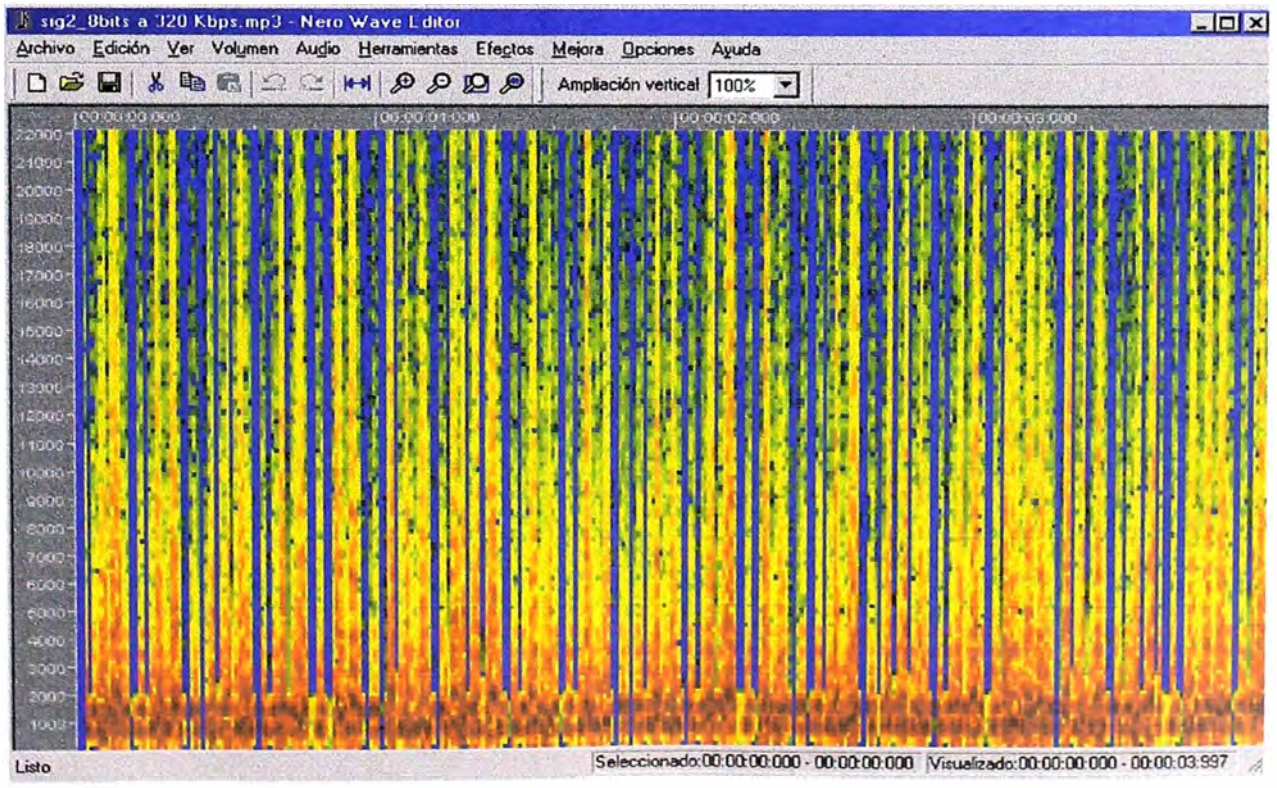
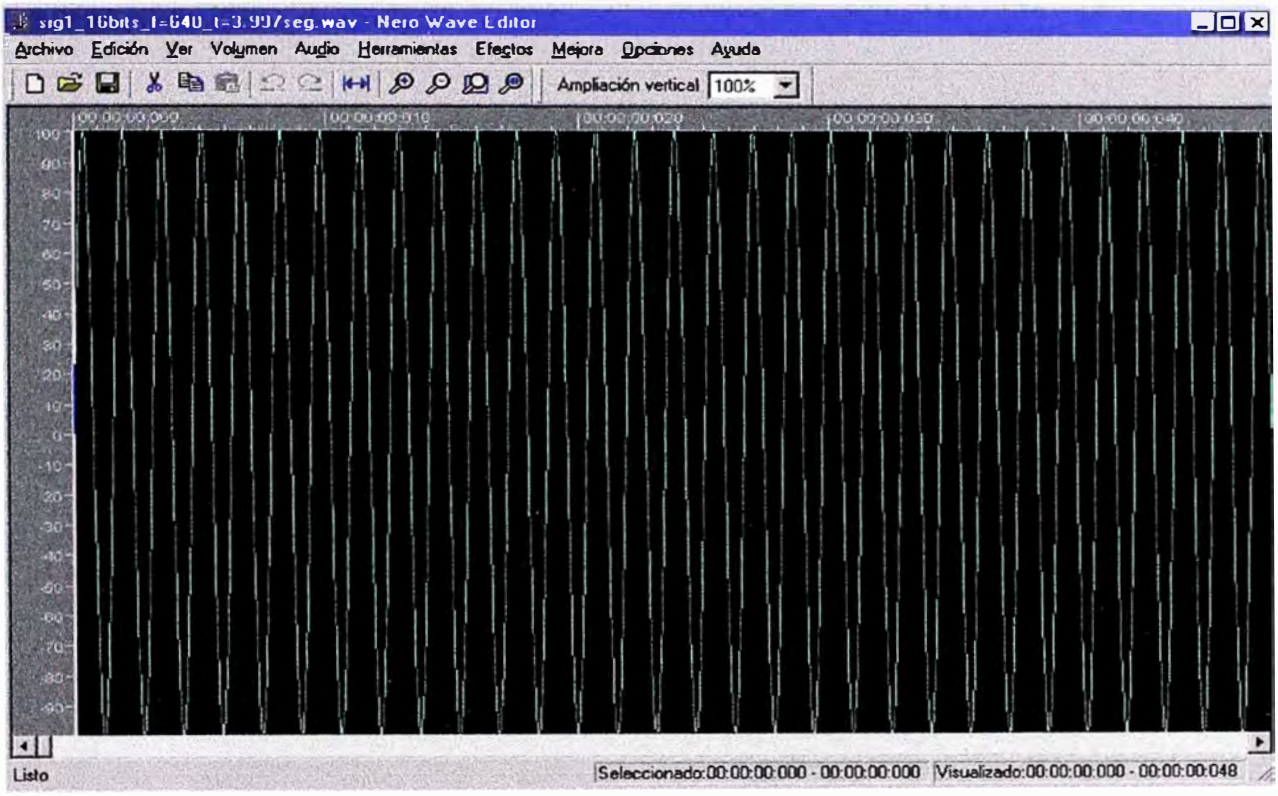


Figura PS33: **sig2_8bits a 320 Kbps_f=640_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

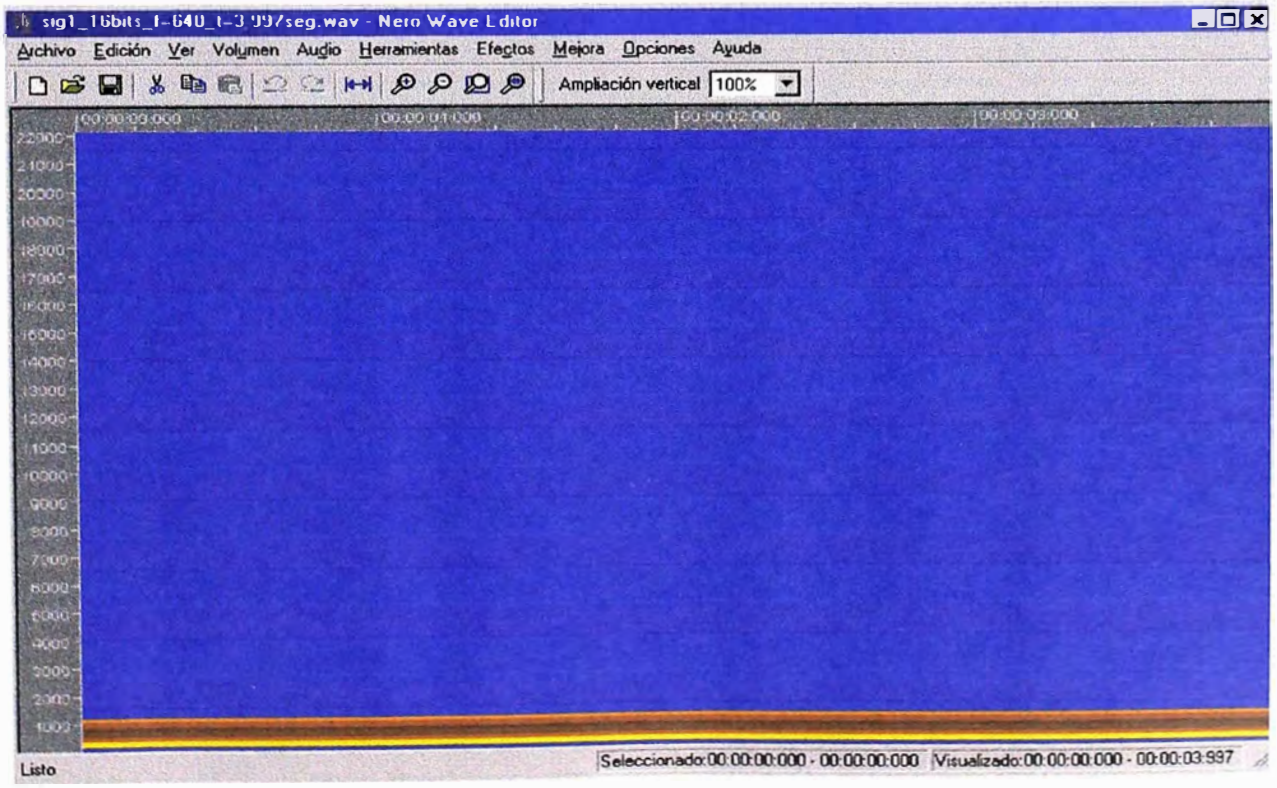
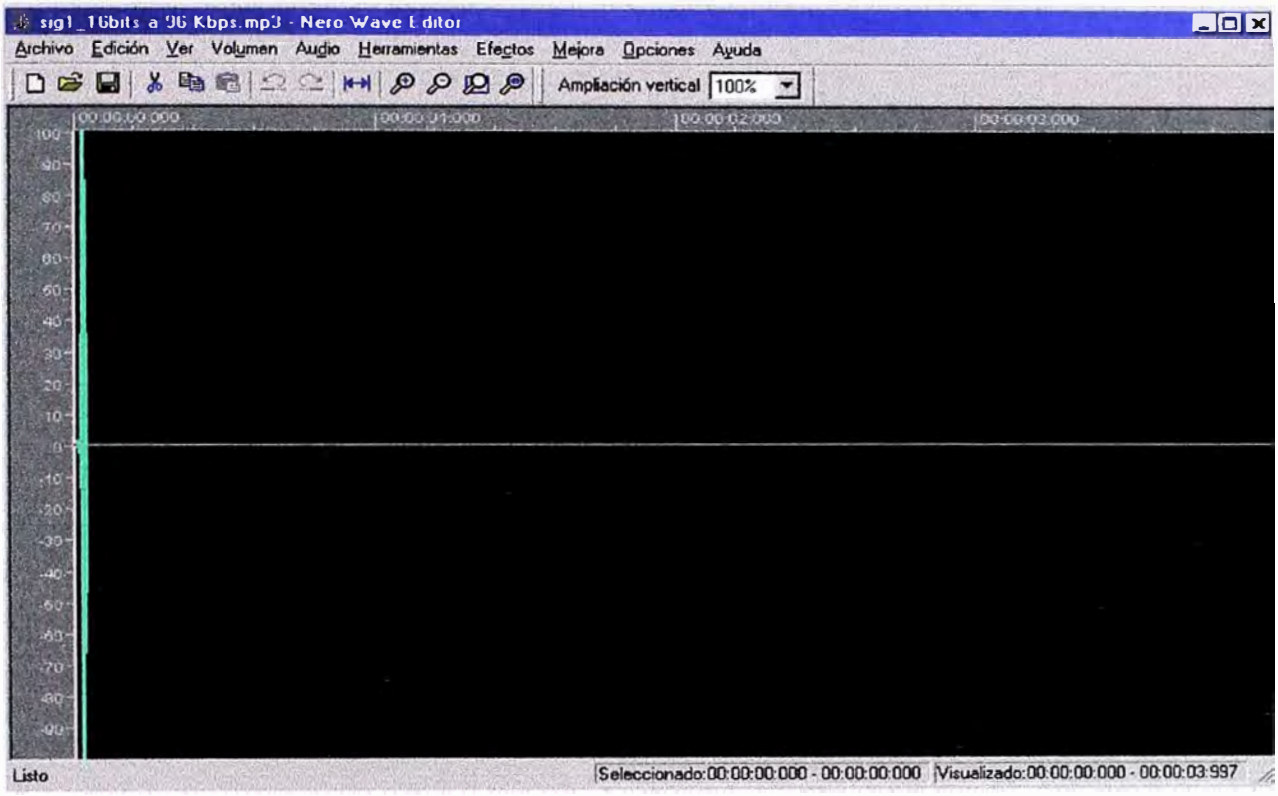


Figura PS34: **sig1_16bits_f=640_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

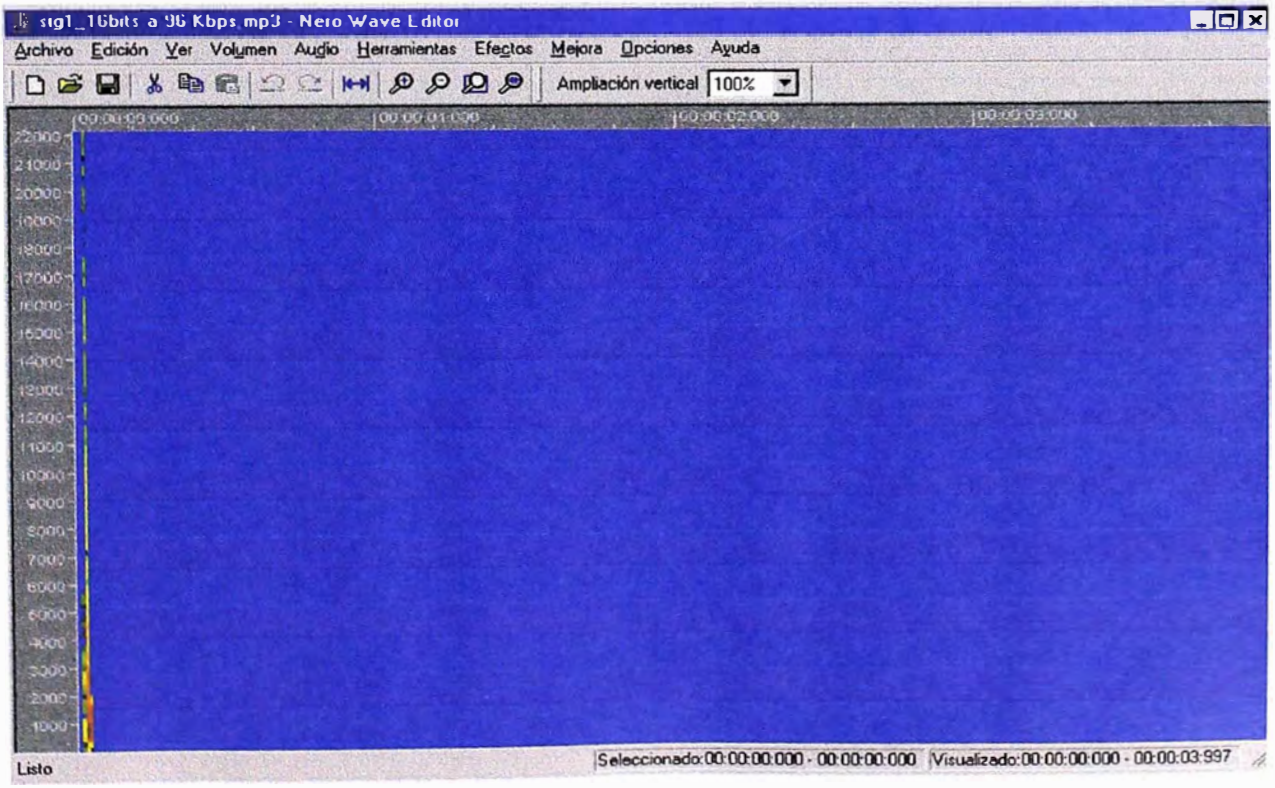
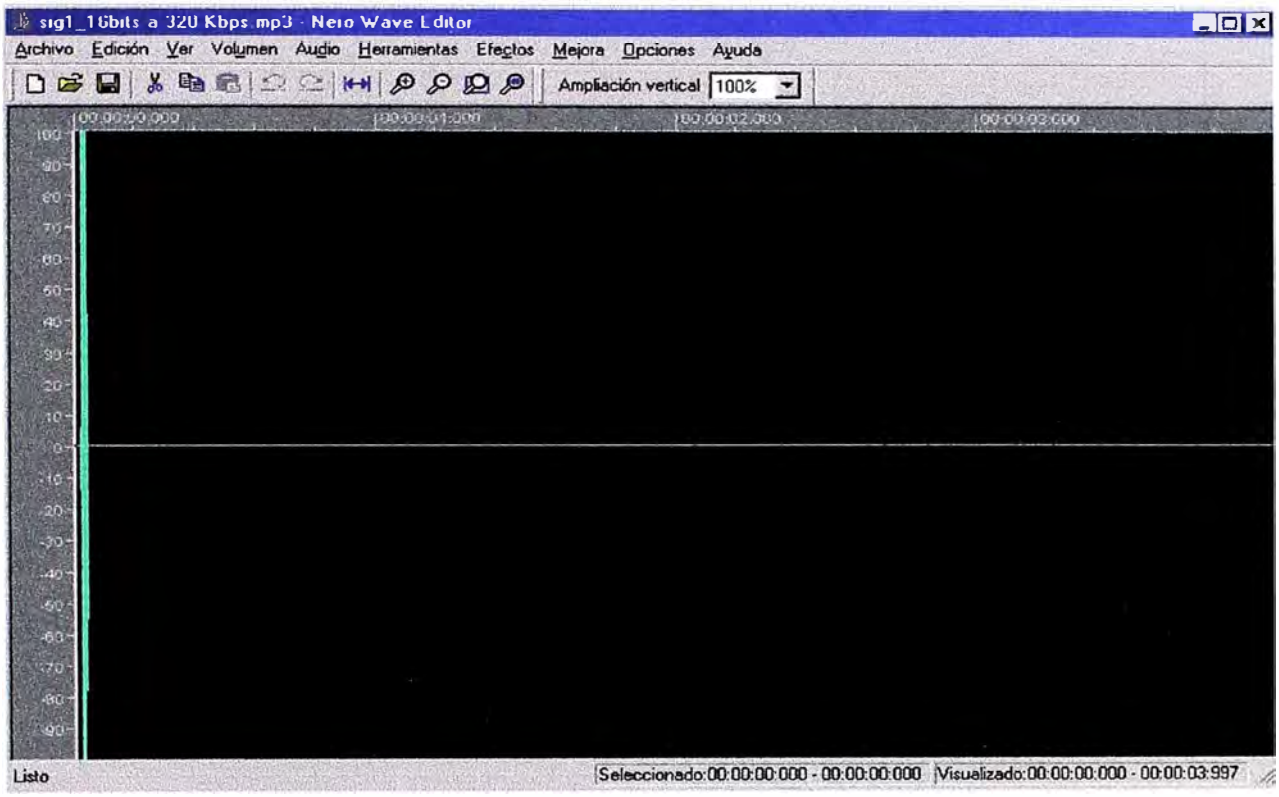


Figura PS35: **sig1_16bits a 96 Kbps_f=640_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

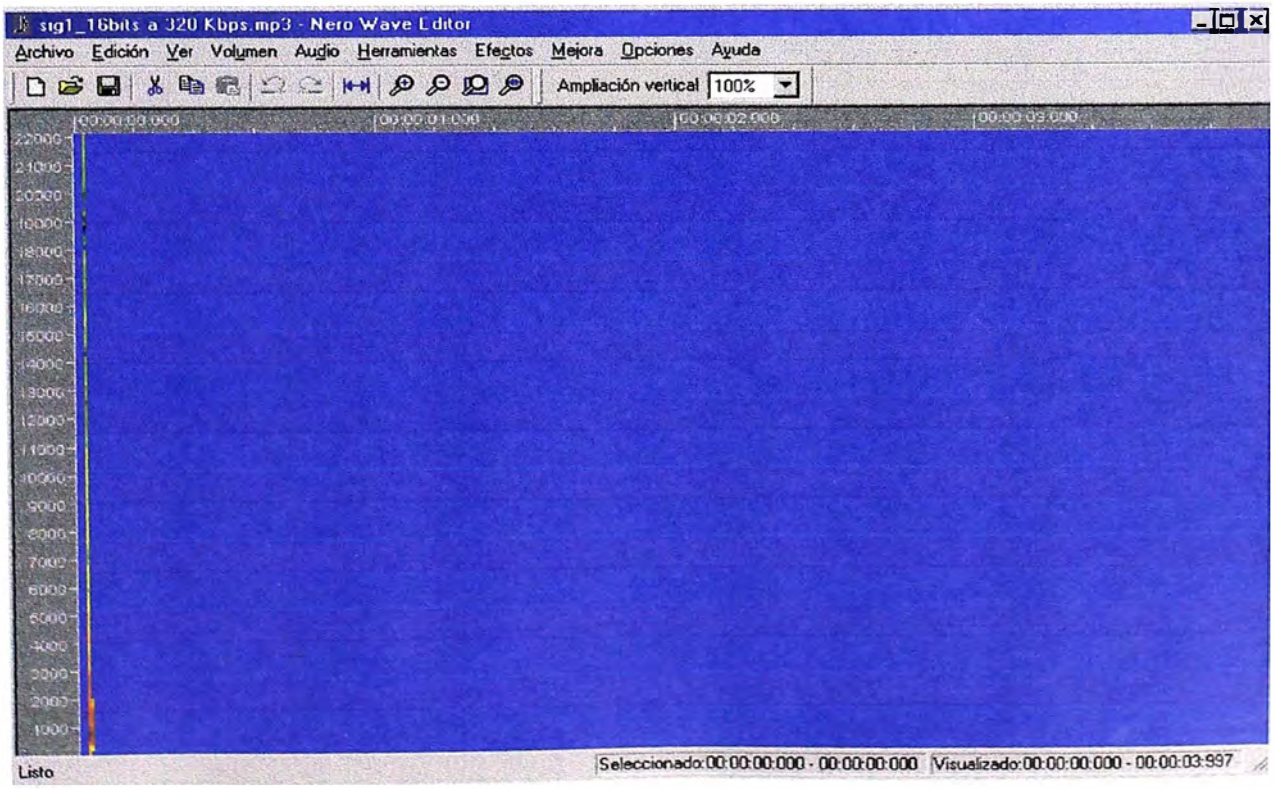
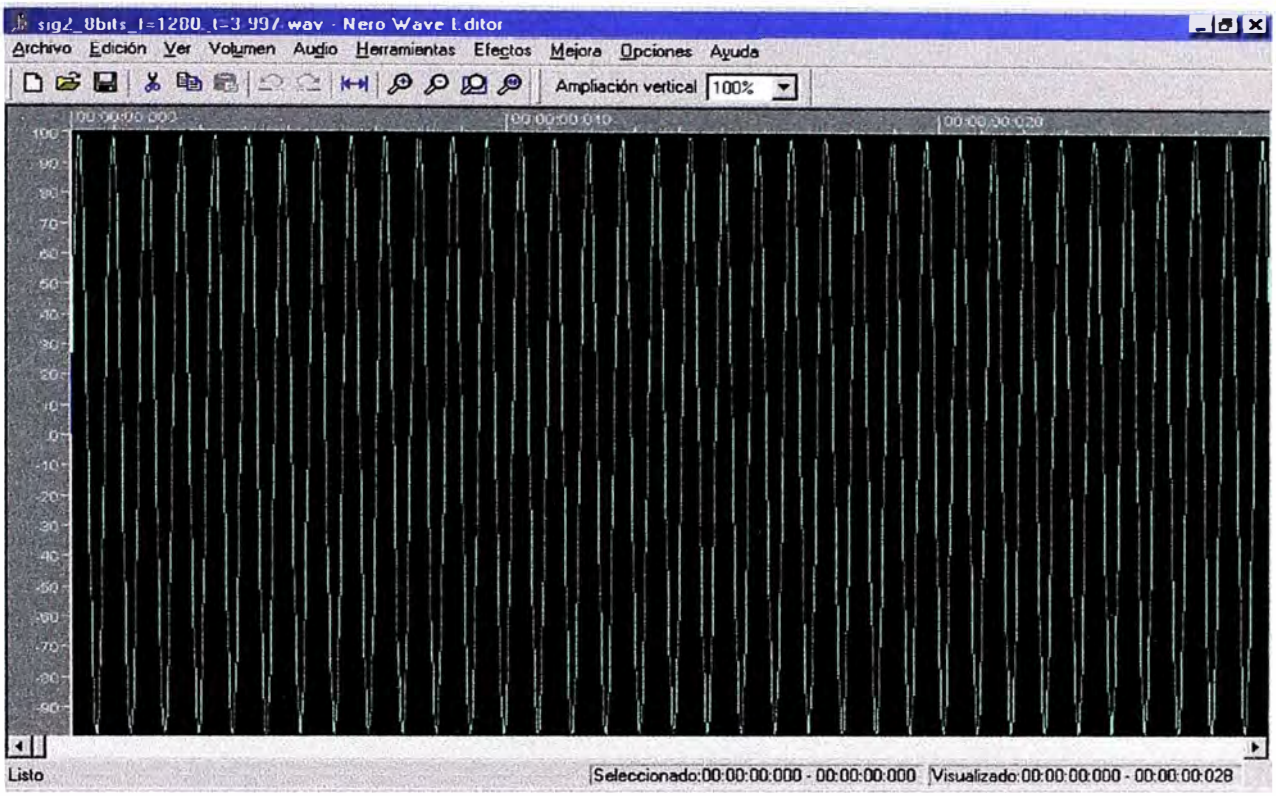


Figura PS36: **sig1_16bits a 320 Kbps_f=640_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

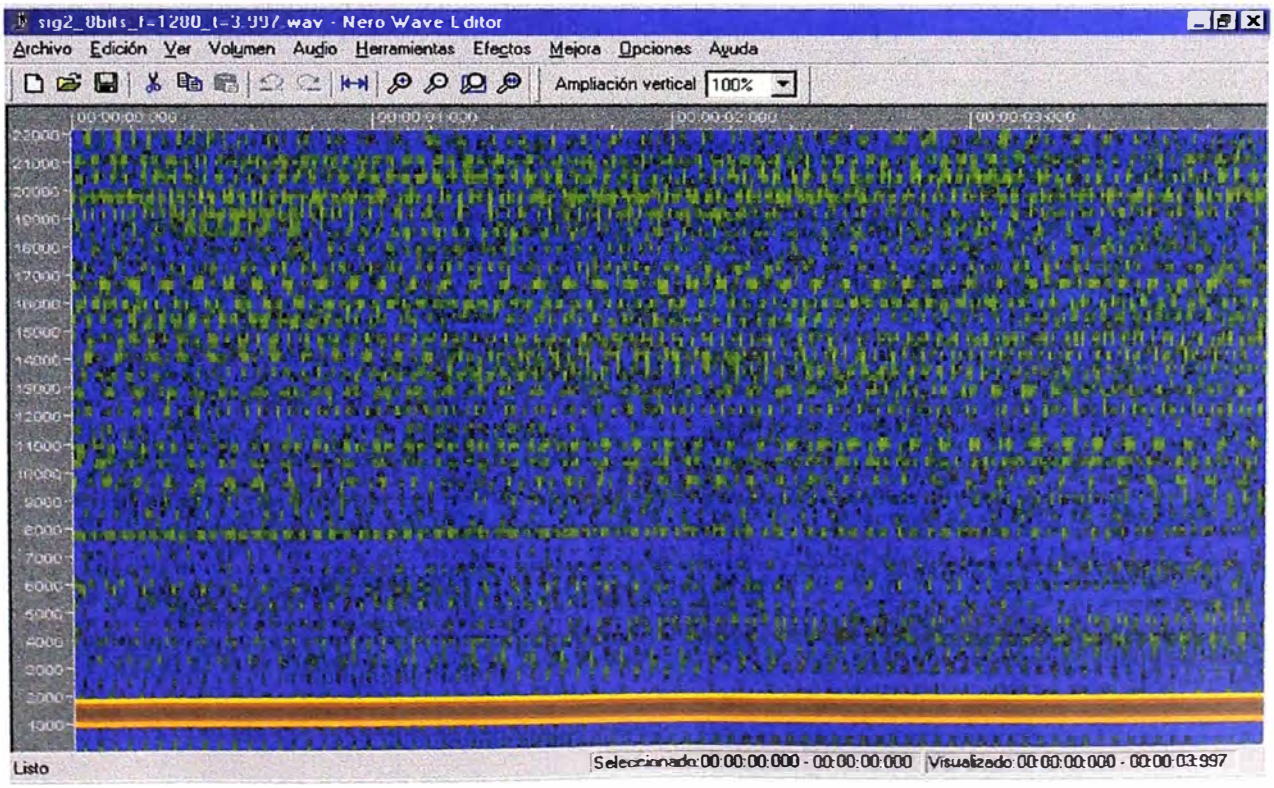
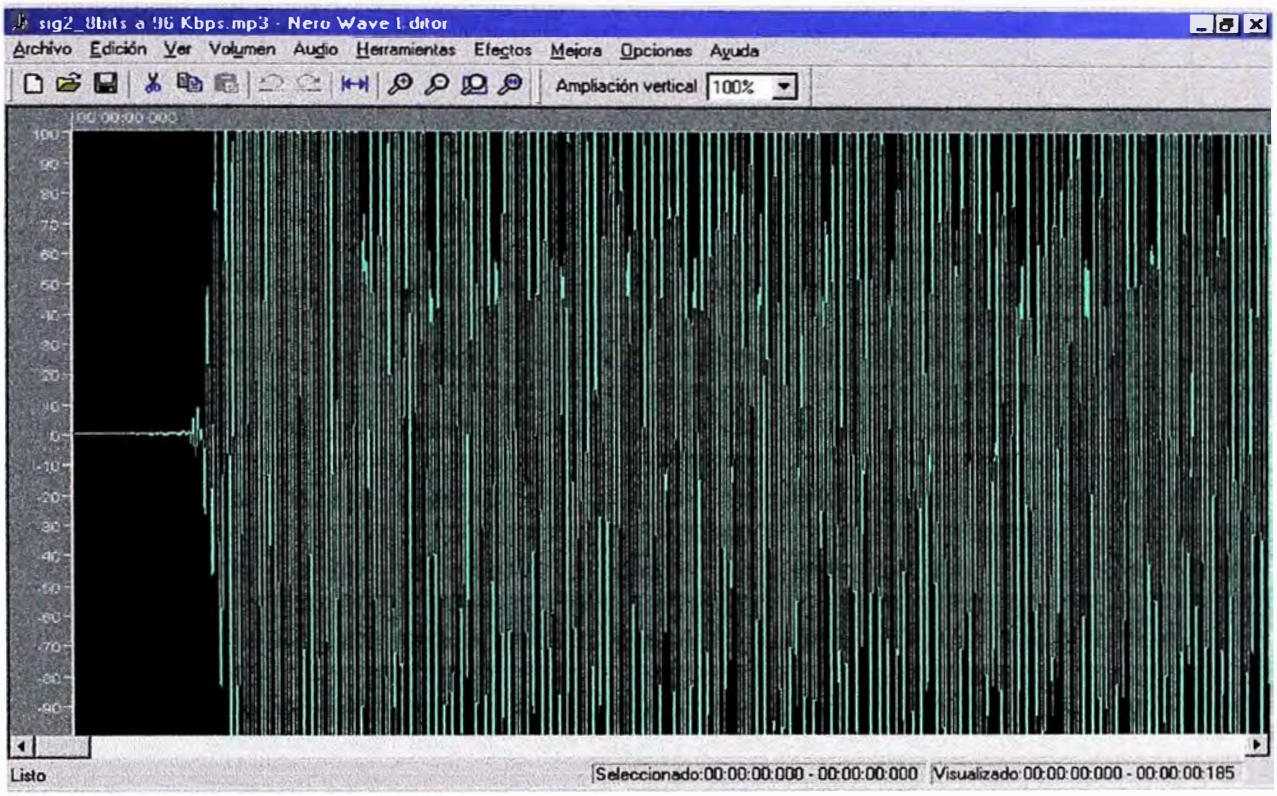


Figura PS37: **sig2_8bits_f=1280_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

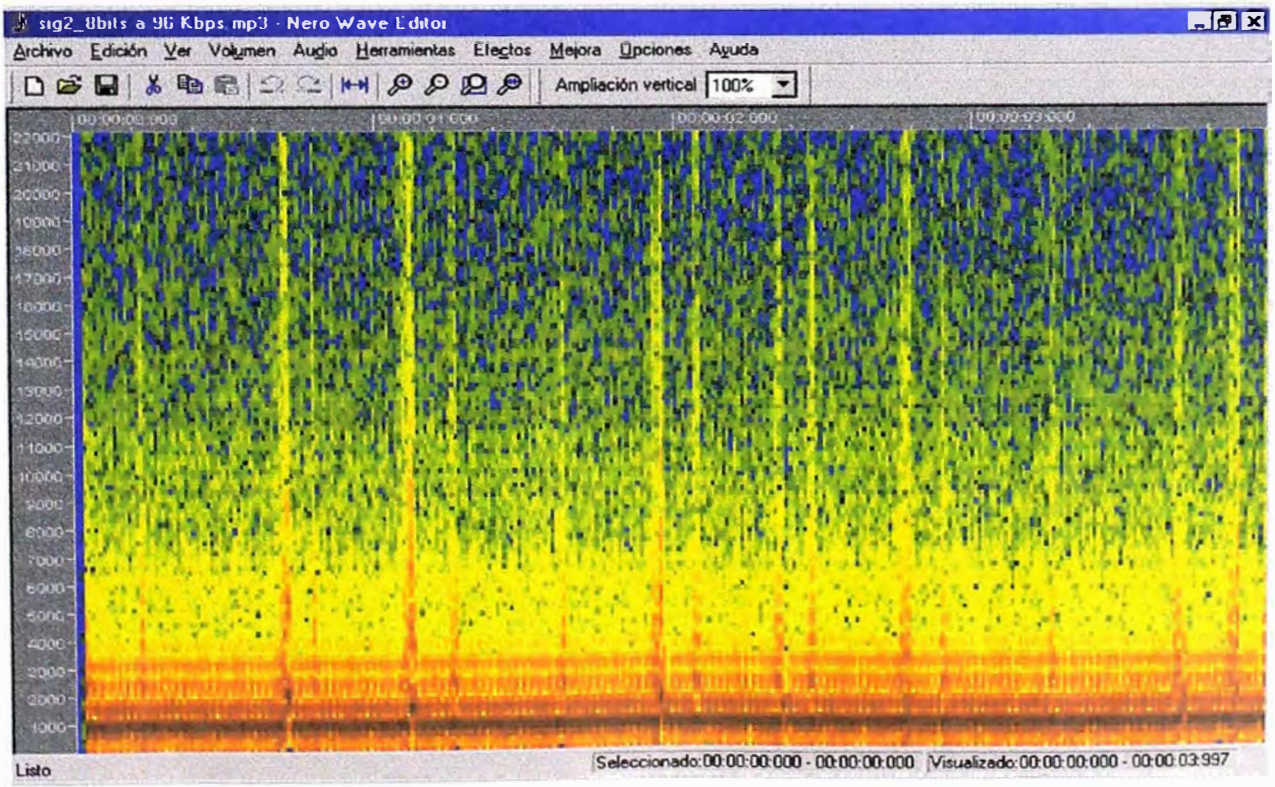
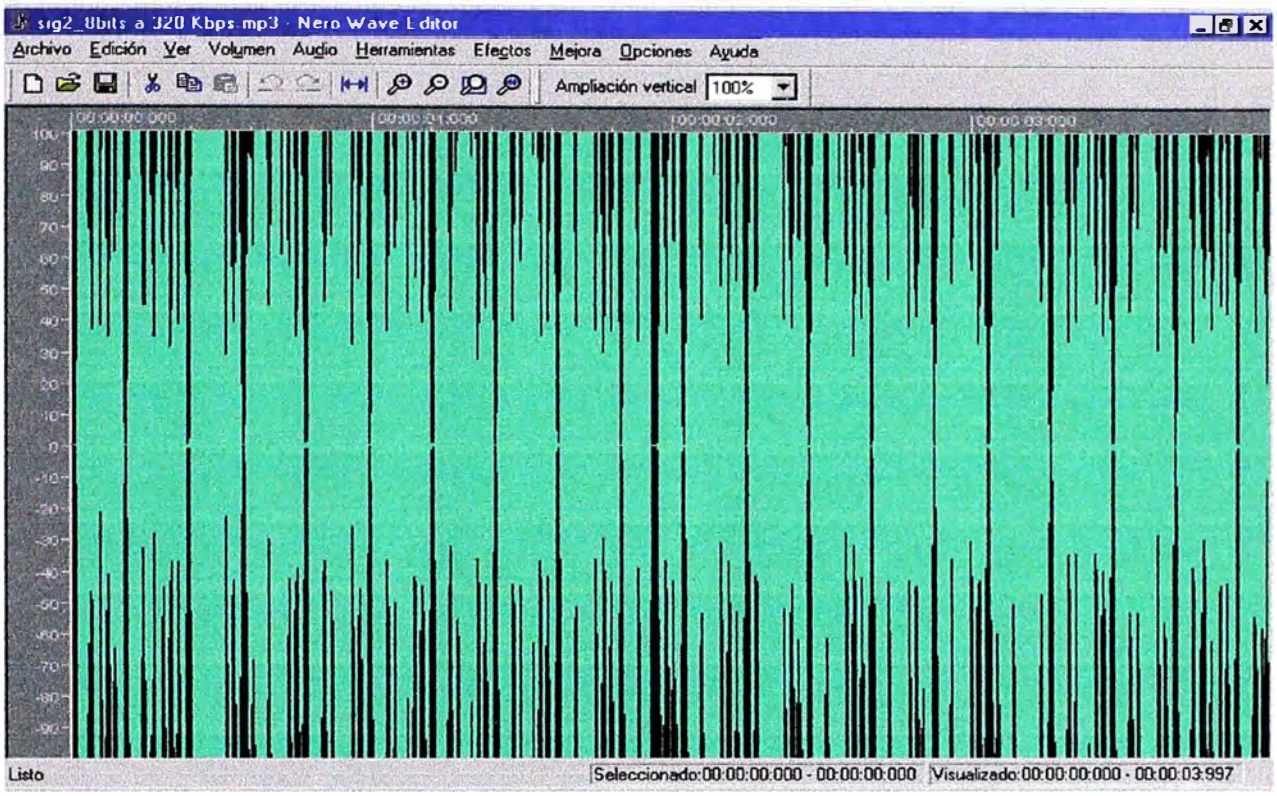


Figura PS38: **sig2_8bits a 96 Kbps_f=1280_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

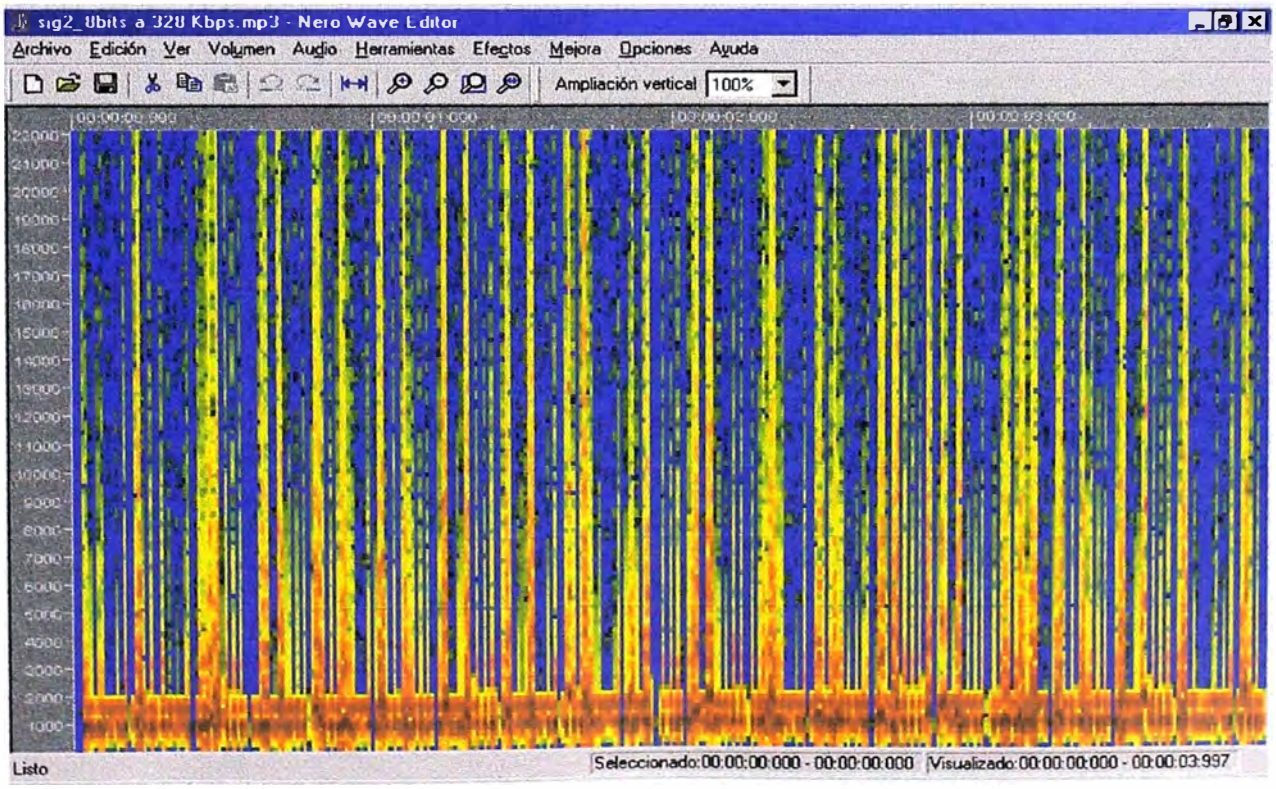
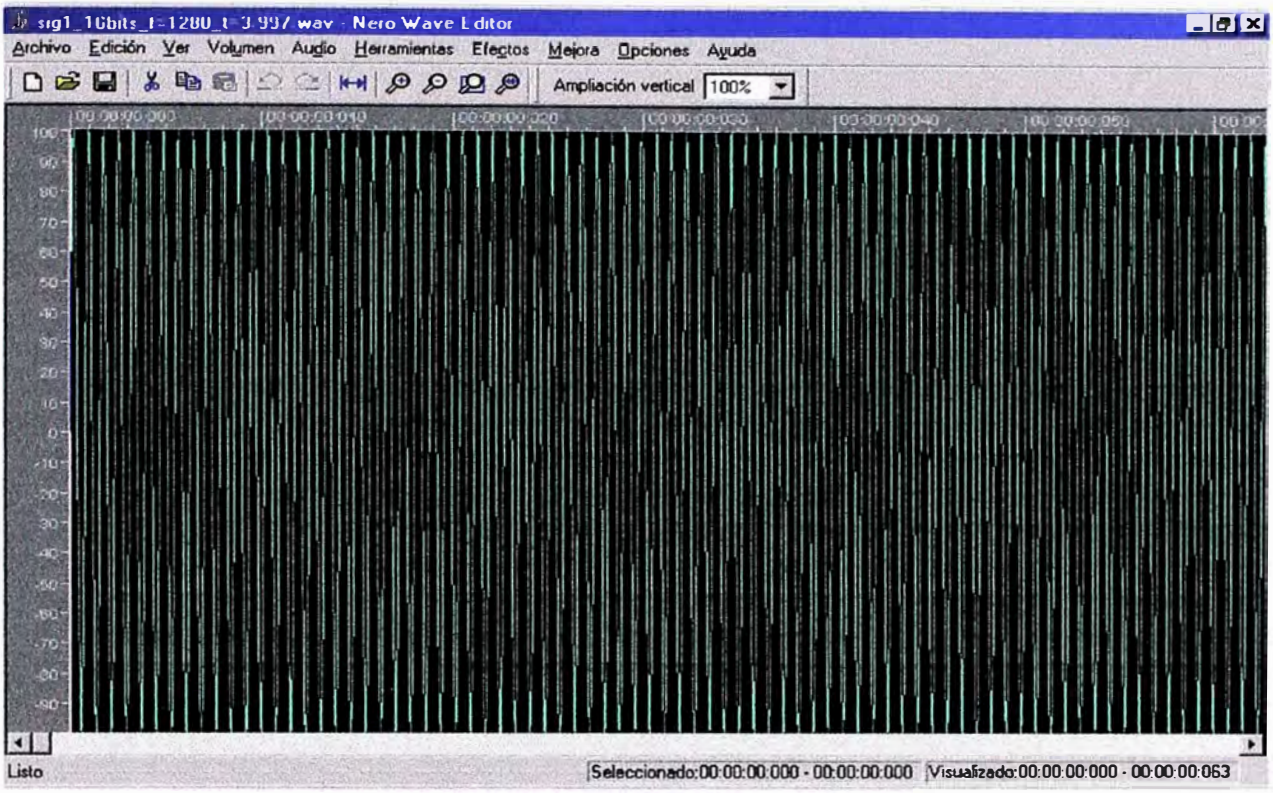


Figura PS39: **sig2_8bits a 320 Kbps_f=1280_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

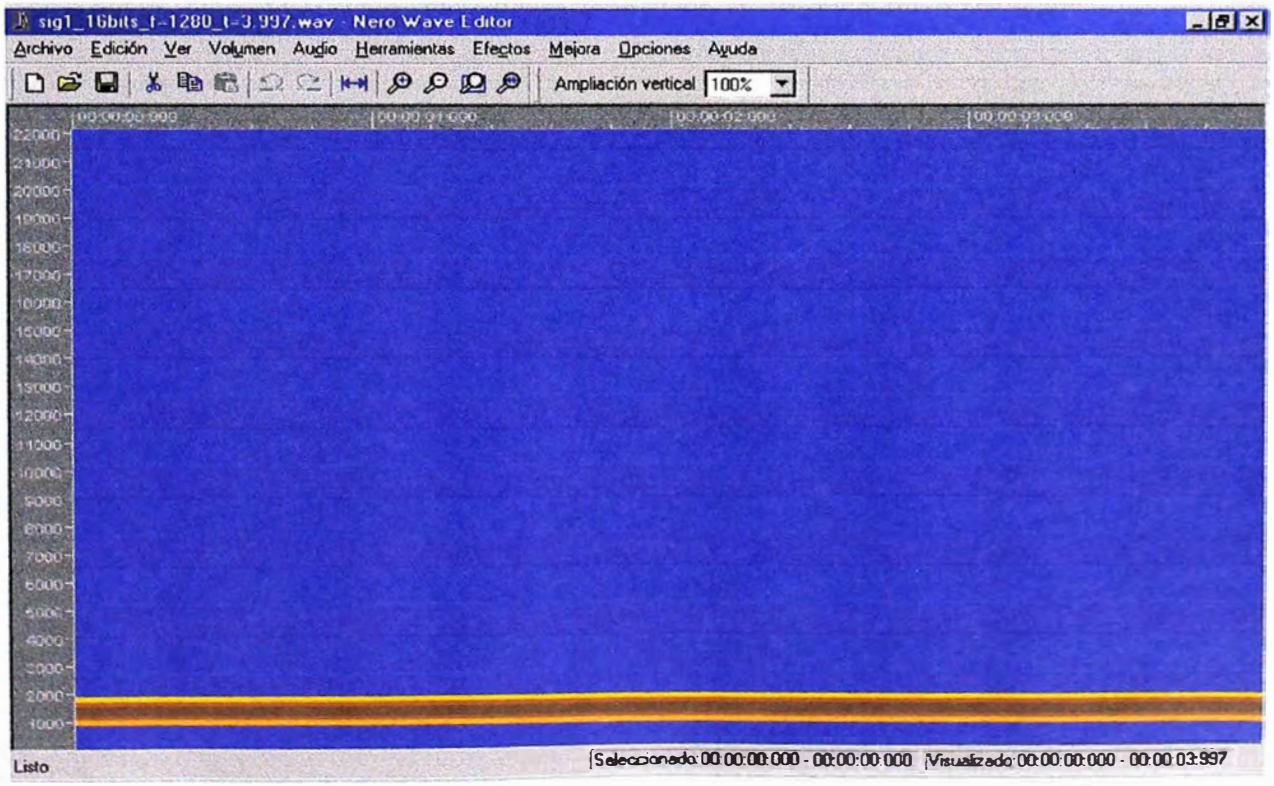
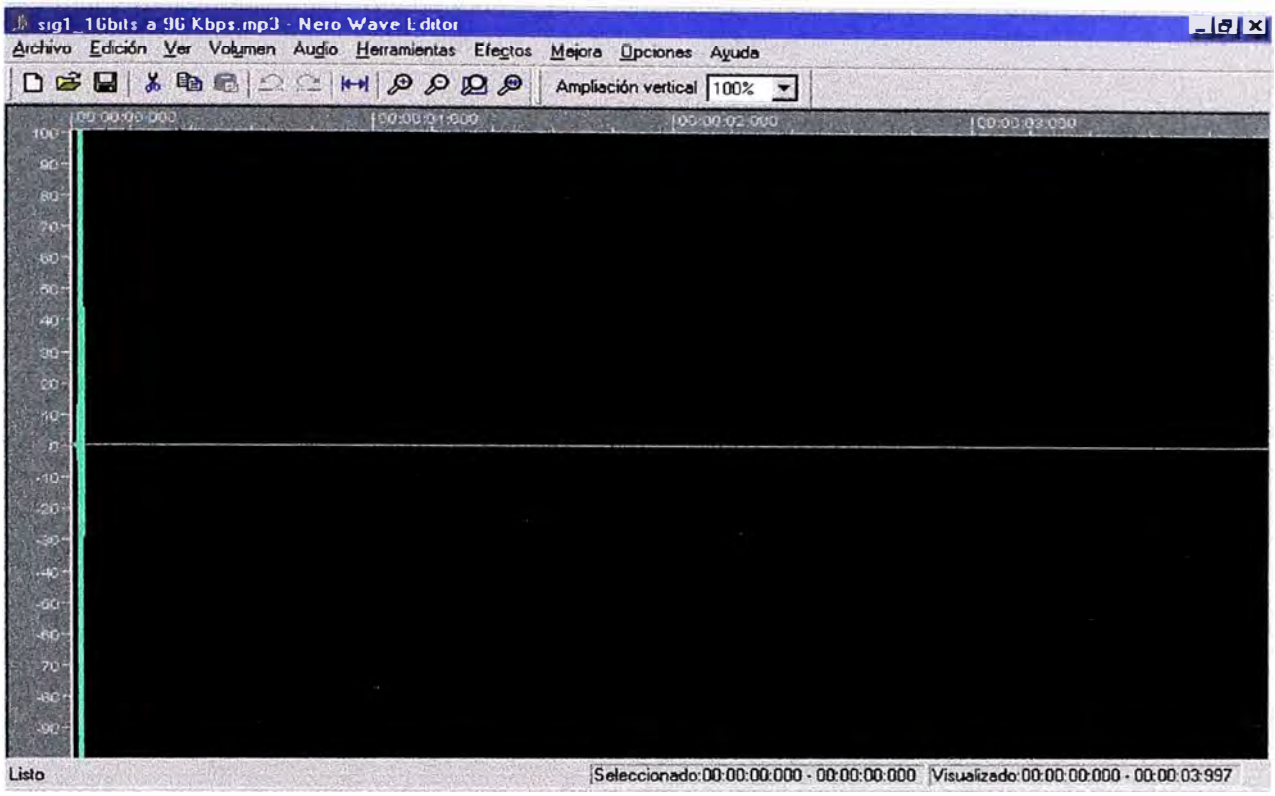


Figura PS40: **sig1_16bits_f=1280_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

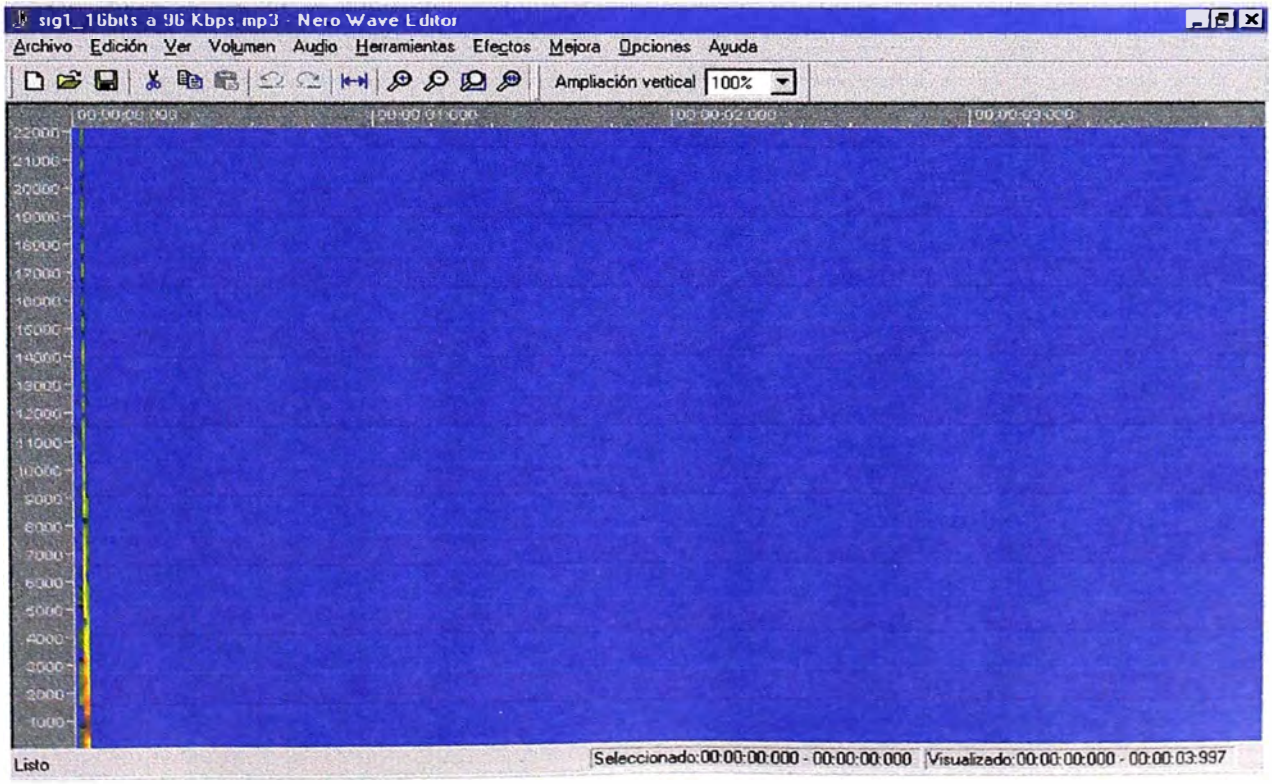
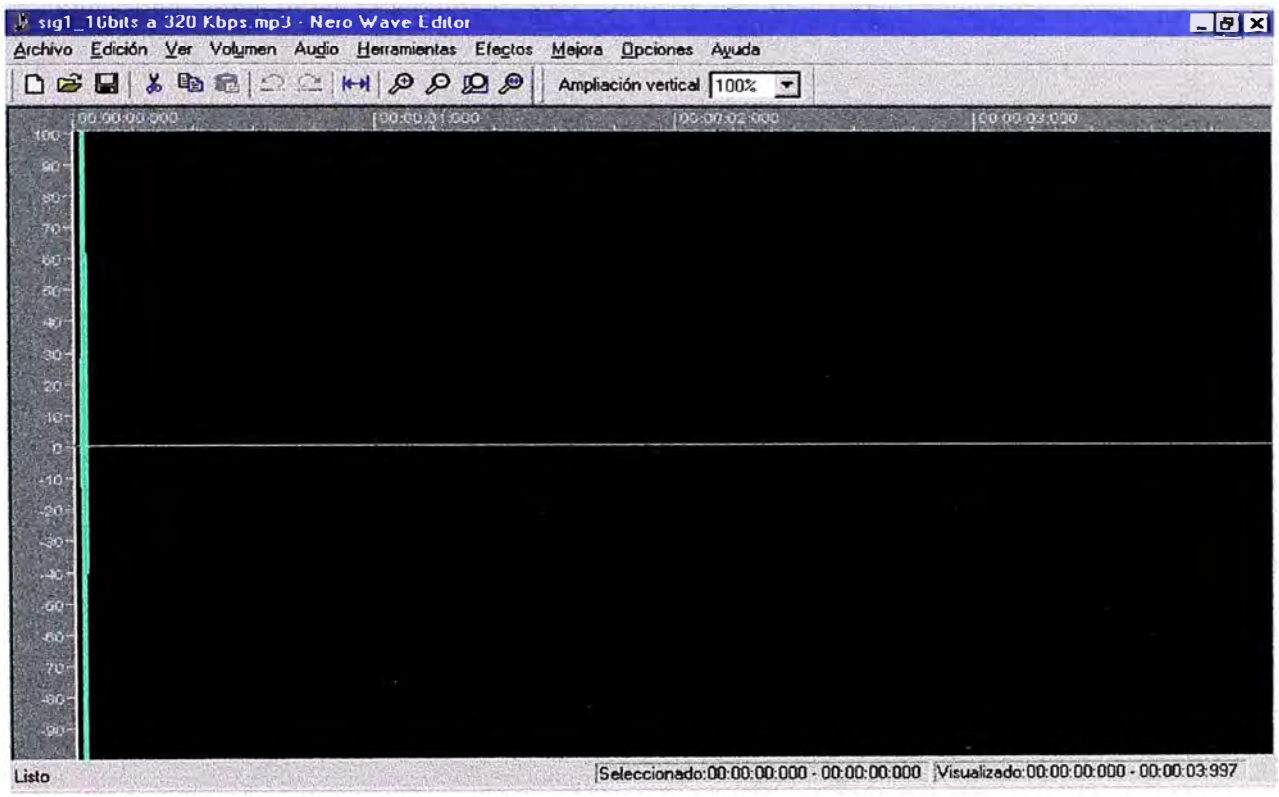


Figura PS41: **sig1_16bits a 96 Kbps_f=1280_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

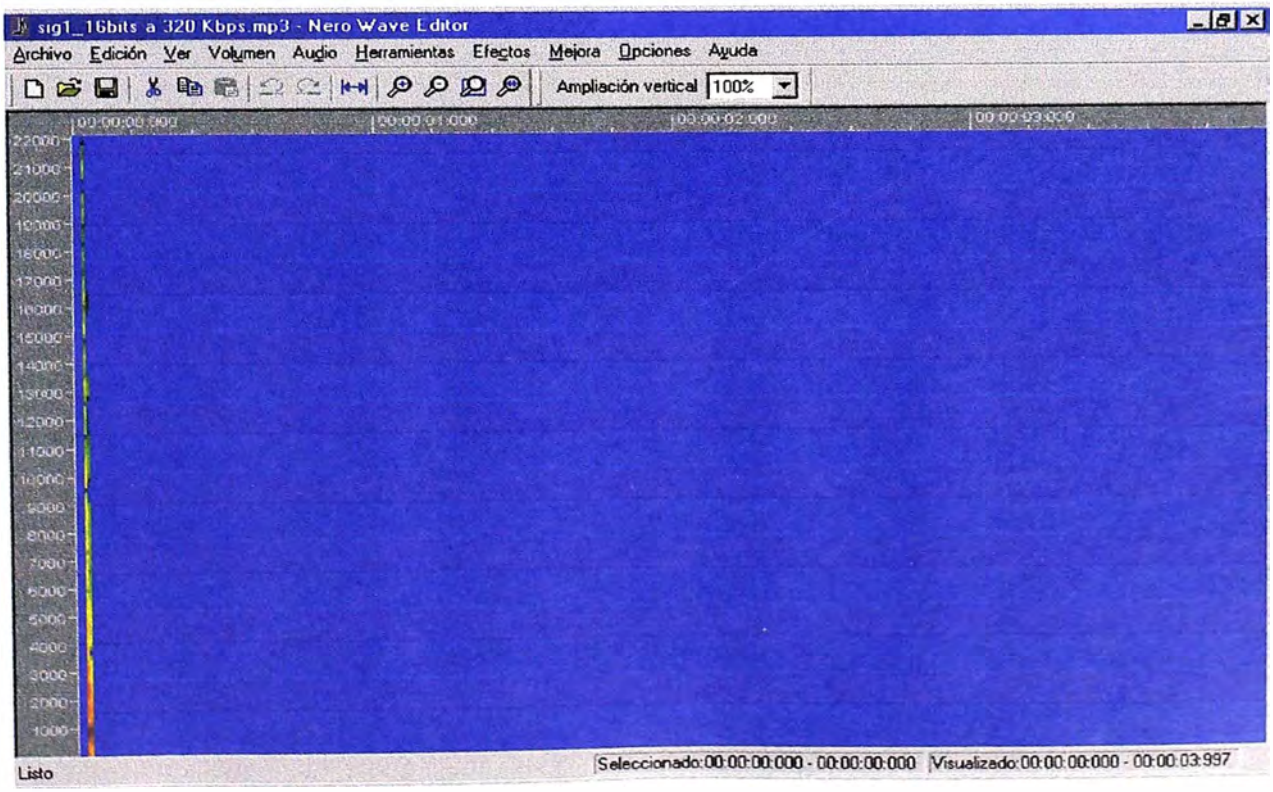
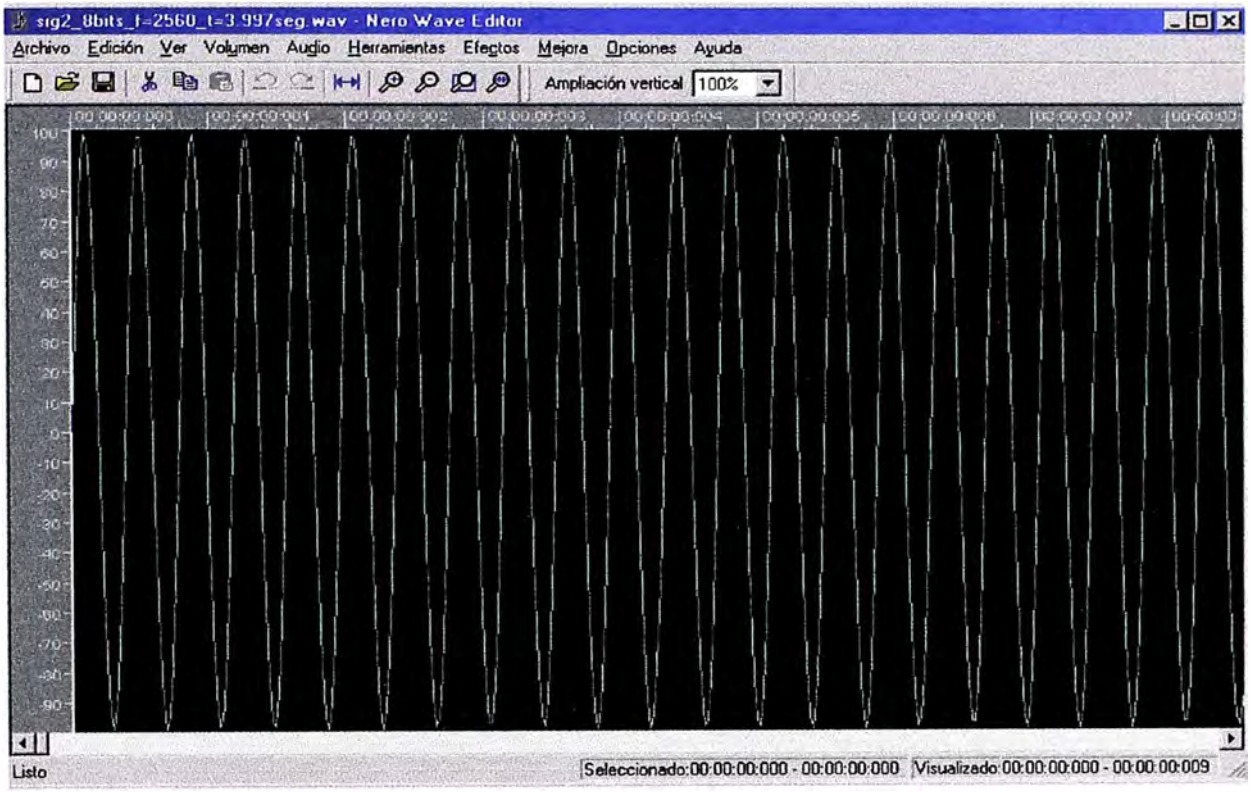


Figura PS42: **sig1_16bits a 320 Kbps_f=1280_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

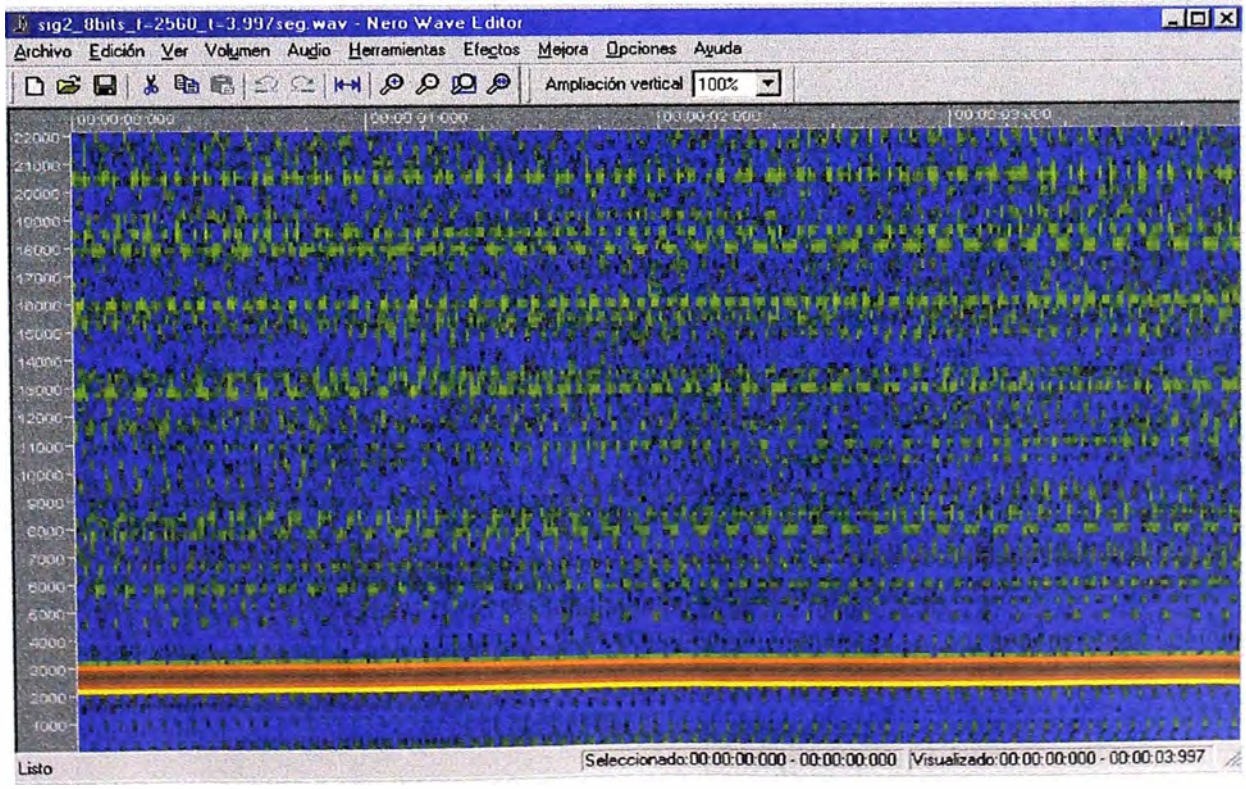
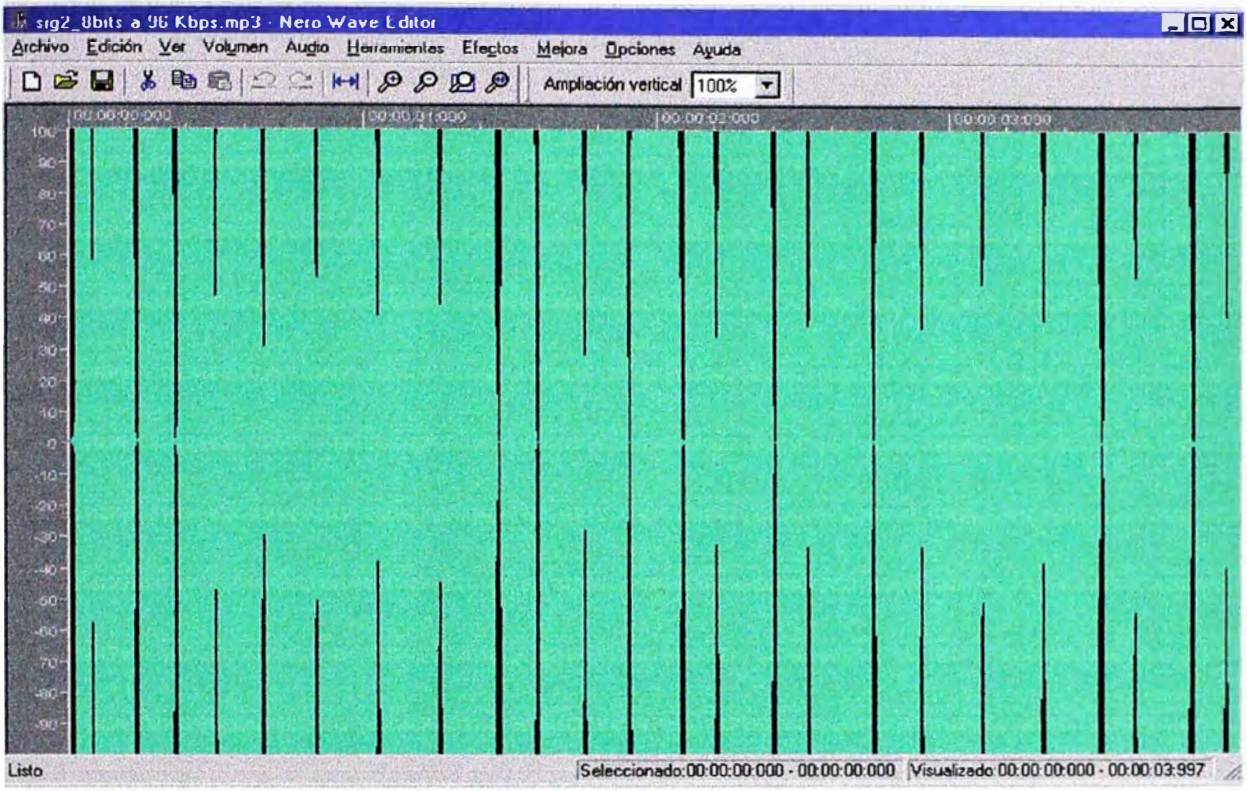


Figura PS43: **sig2_8bits_f=2560_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

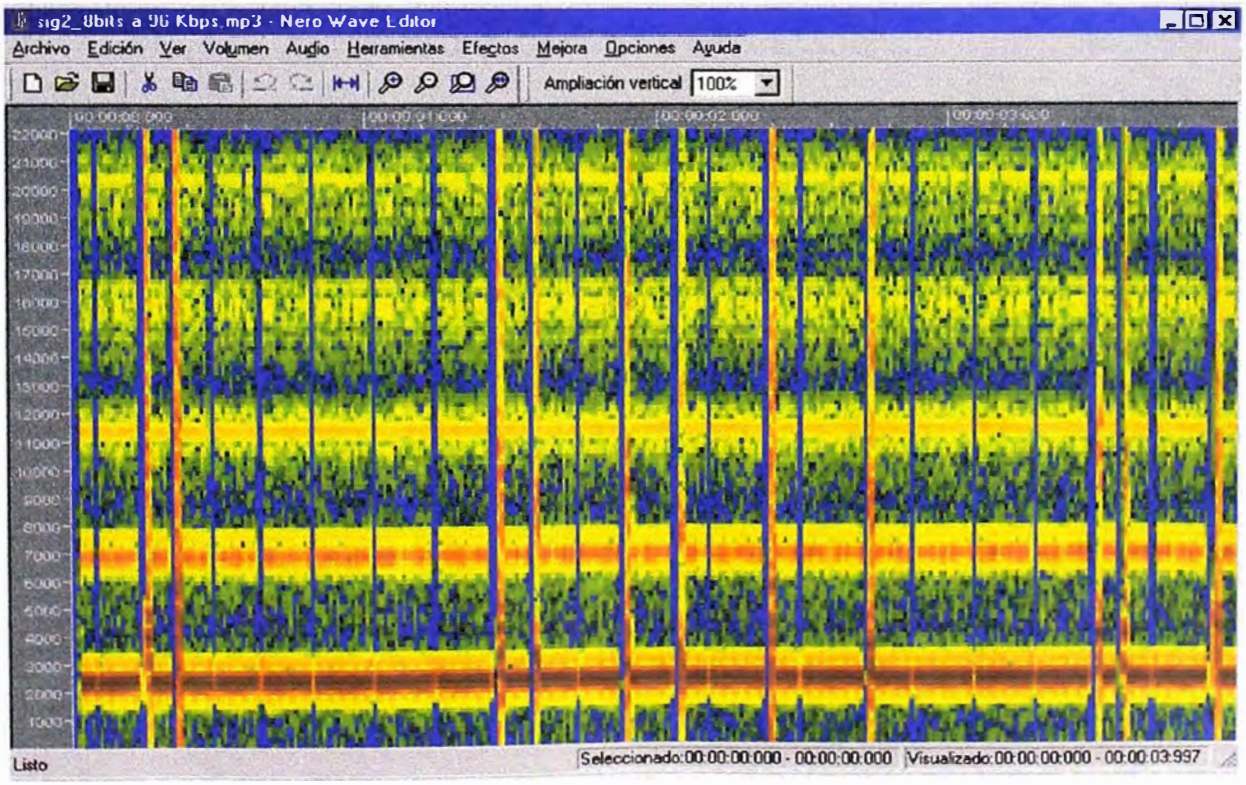
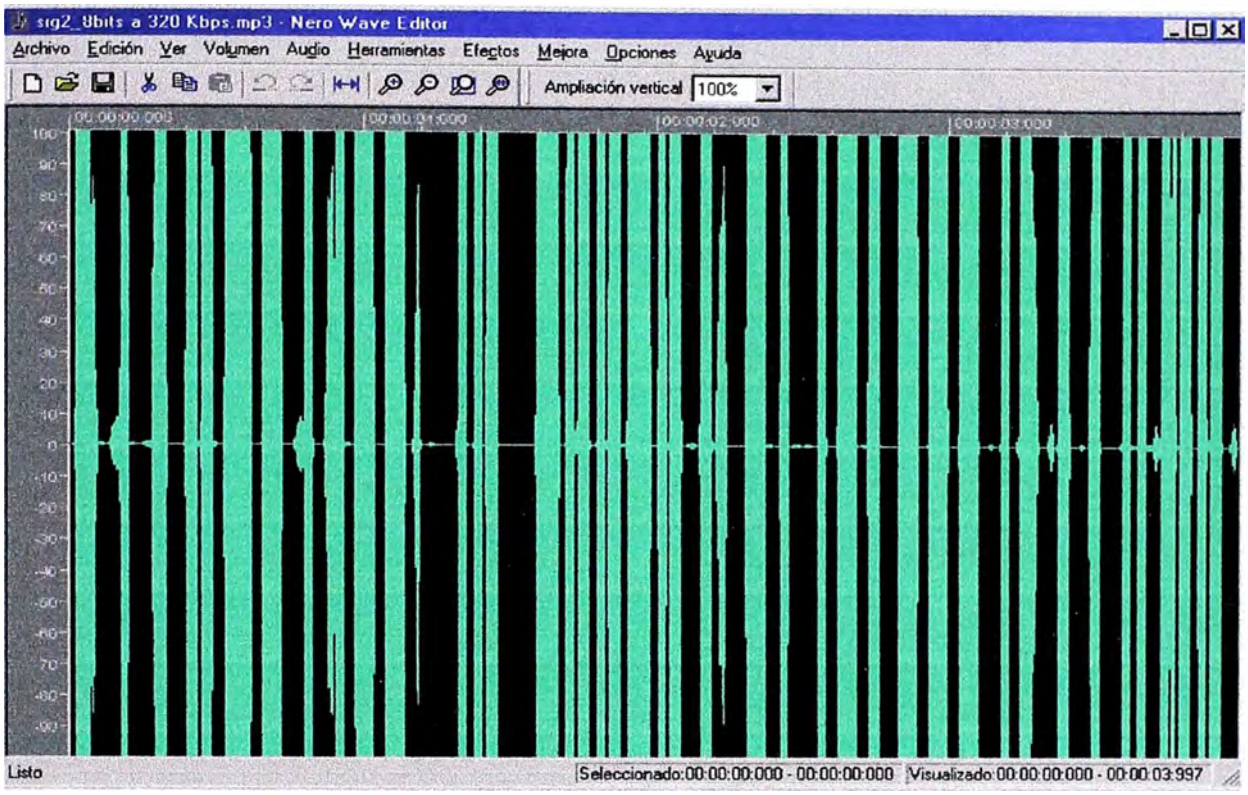


Figura PS44: **sig2_8bits a 96 Kbps_f=2560_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

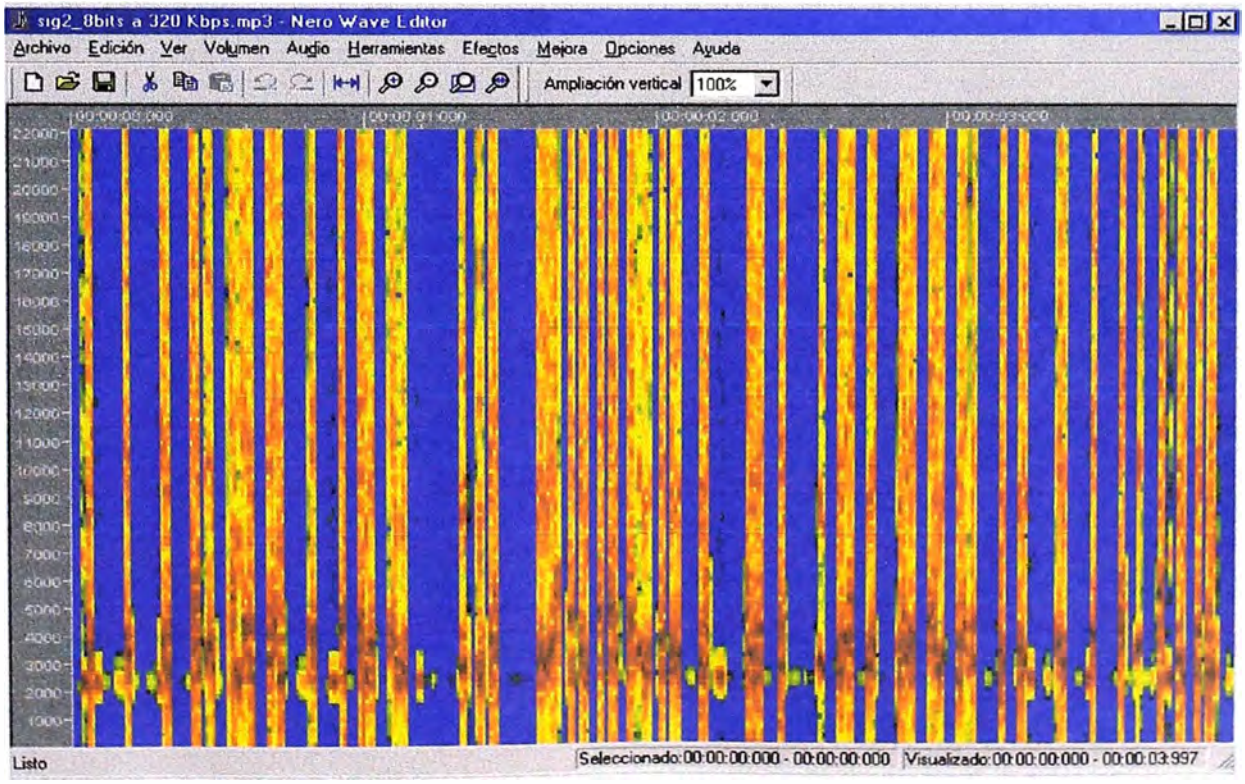
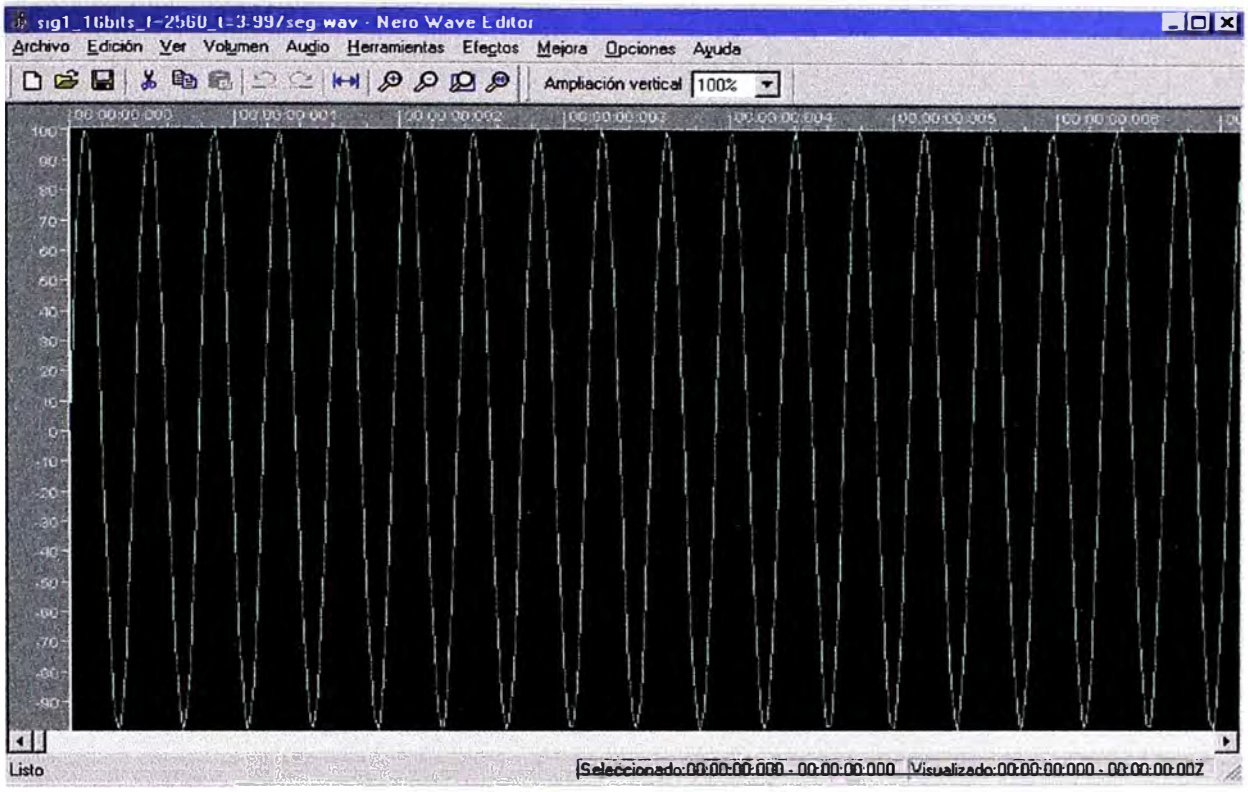


Figura PS45: **sig2_8bits a 320 Kbps_f=2560_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

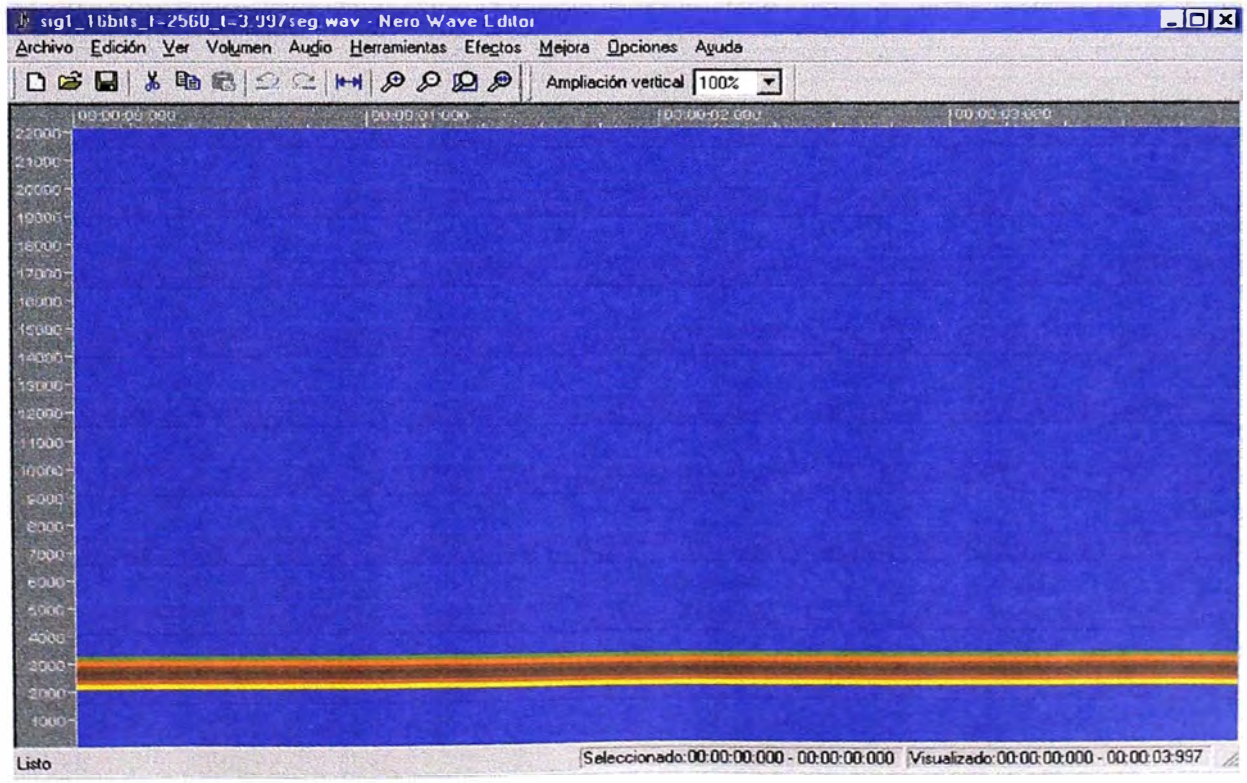
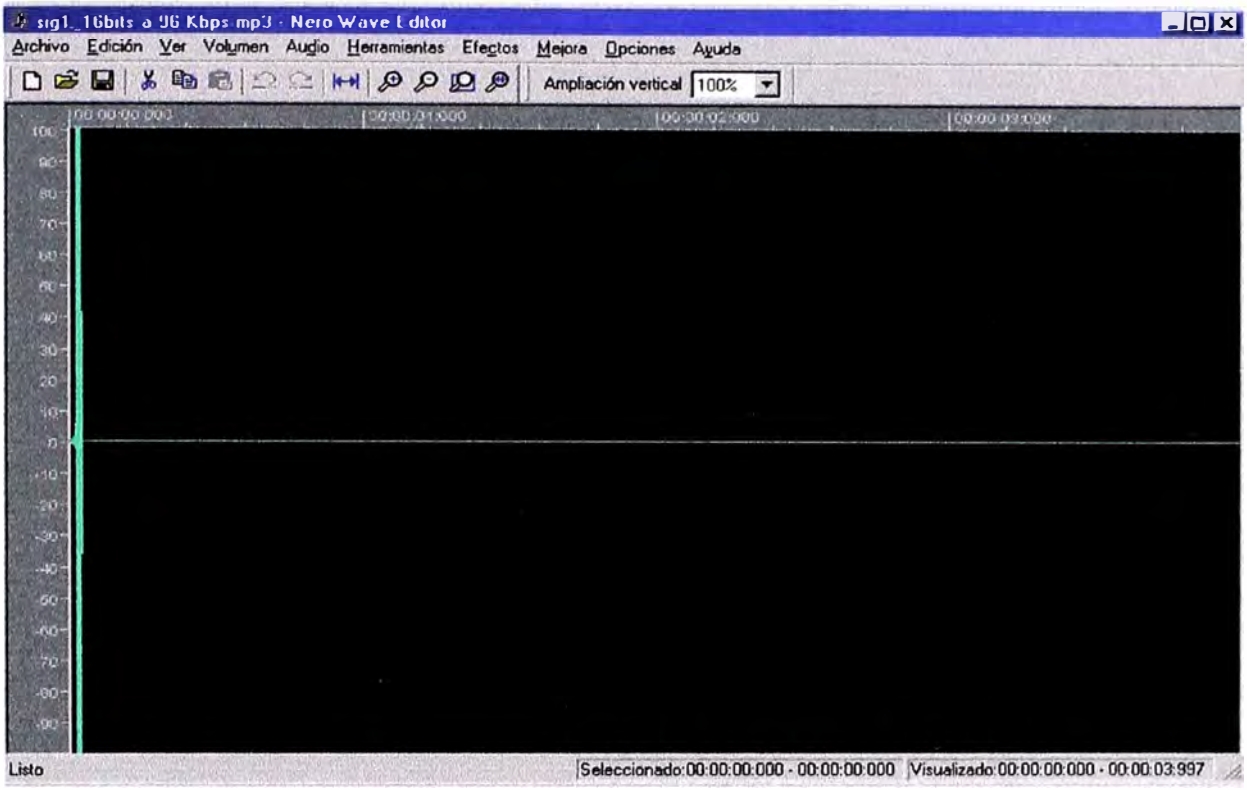


Figura PS46: **sig1_16bits_f=2560_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

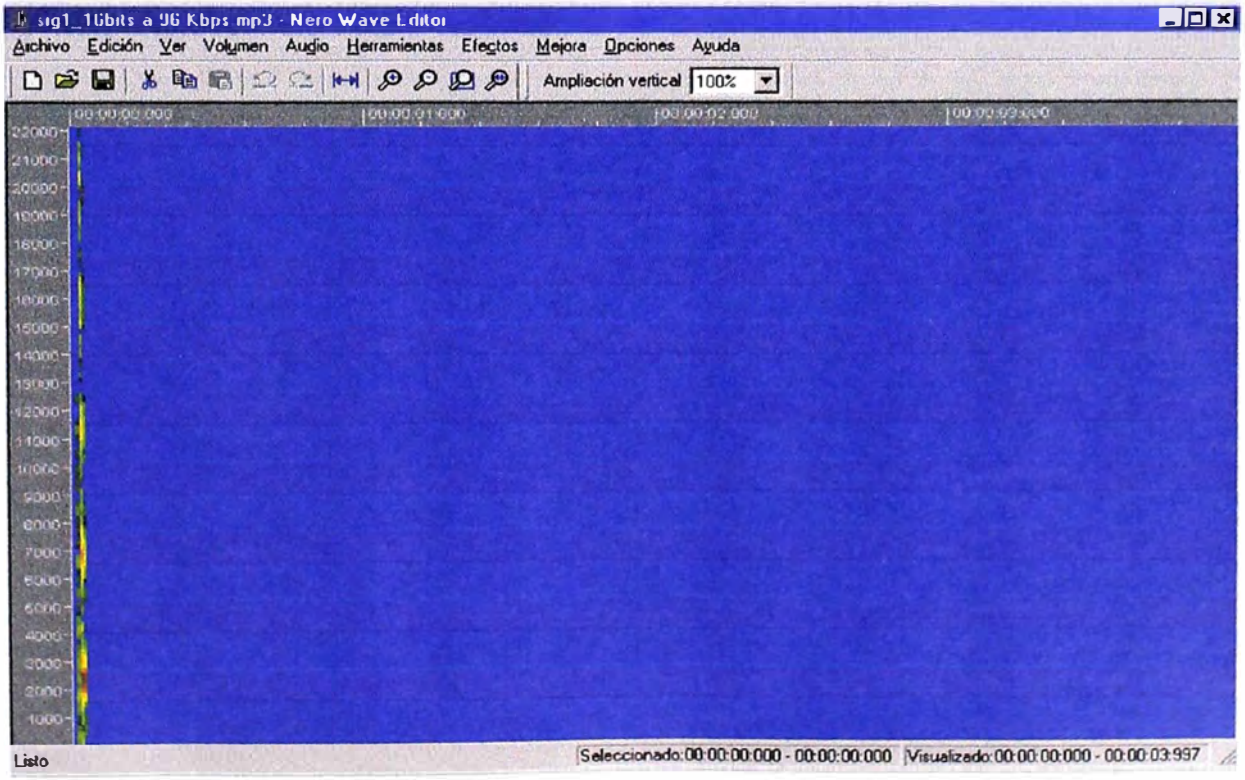
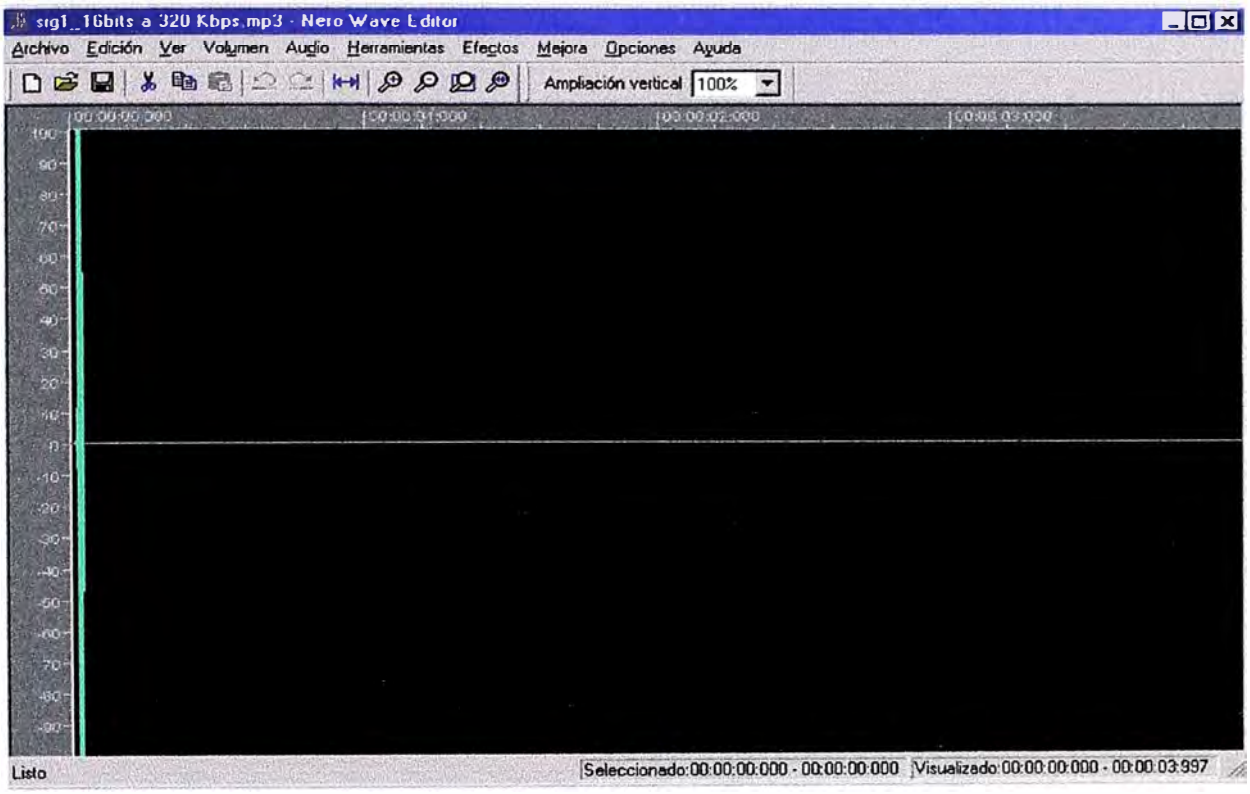


Figura PS47: **sig1_16bits a 96 Kbps_f=2560_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

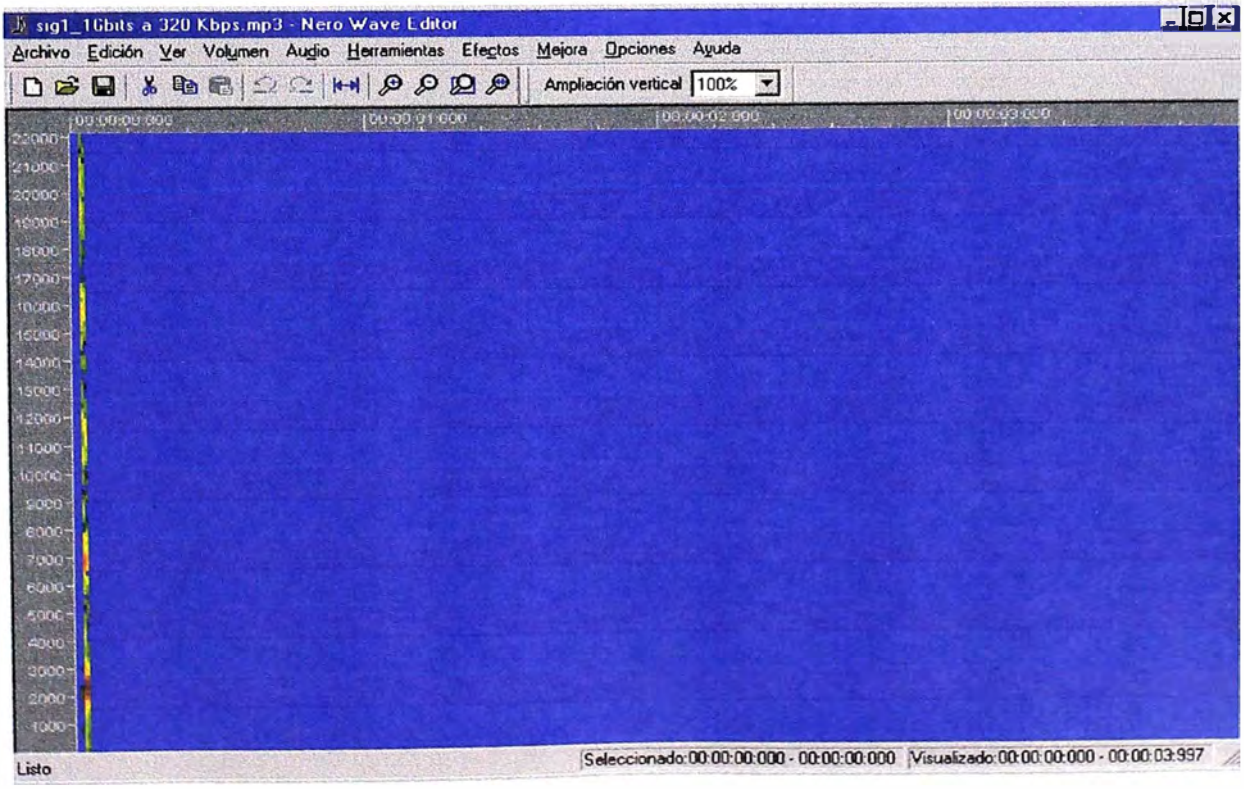
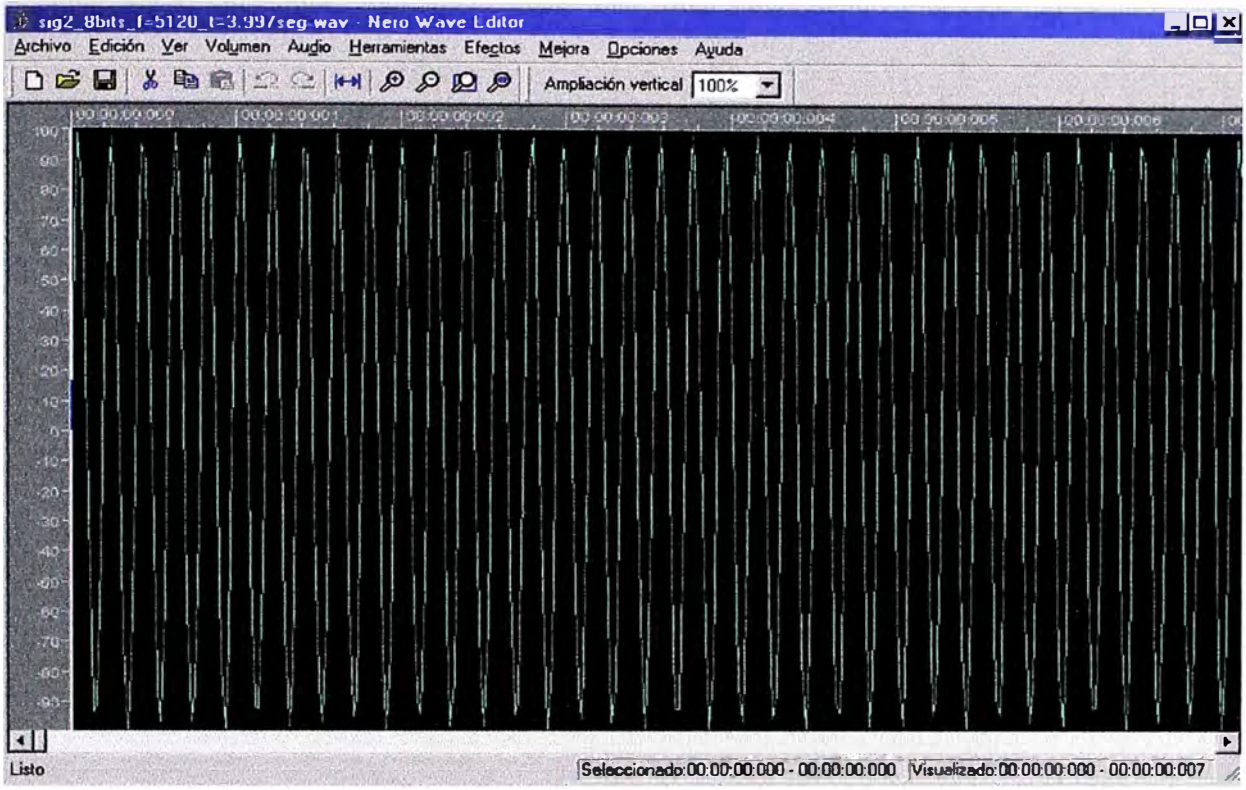


Figura PS48: **sig1_16bits a 320 Kbps_f=2560_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

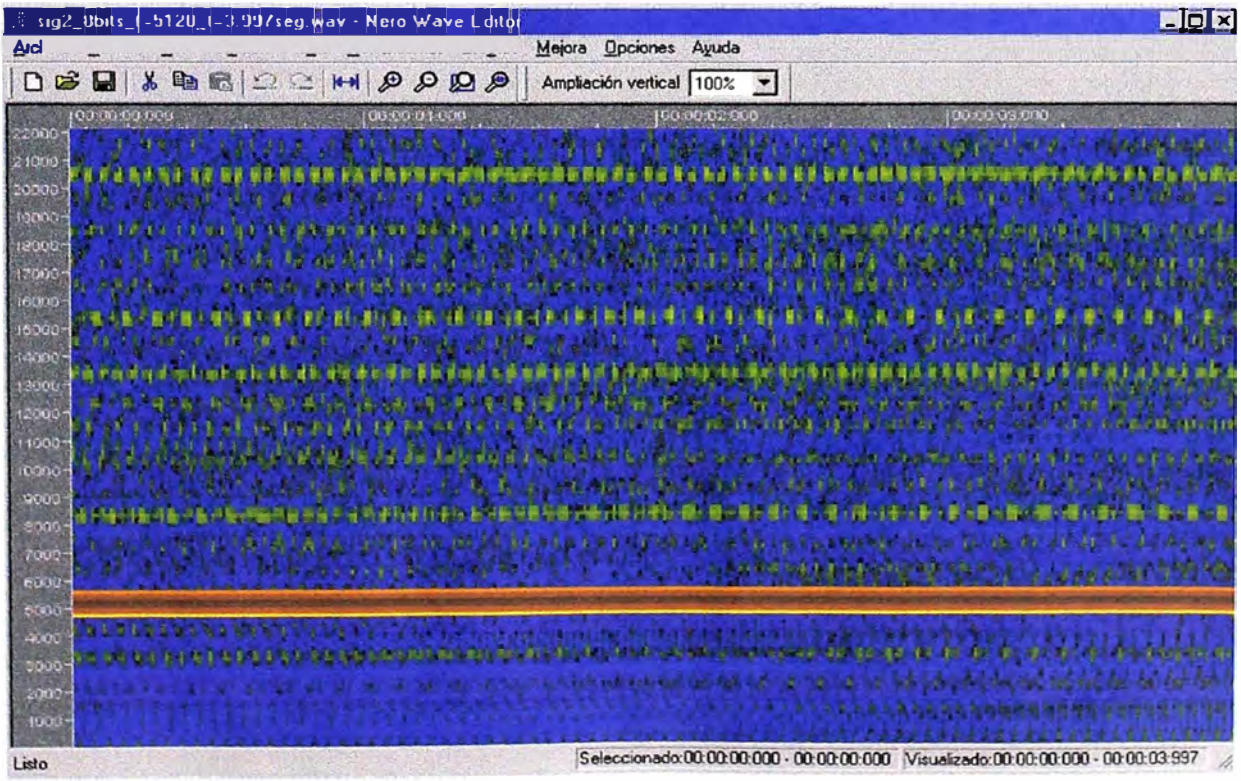
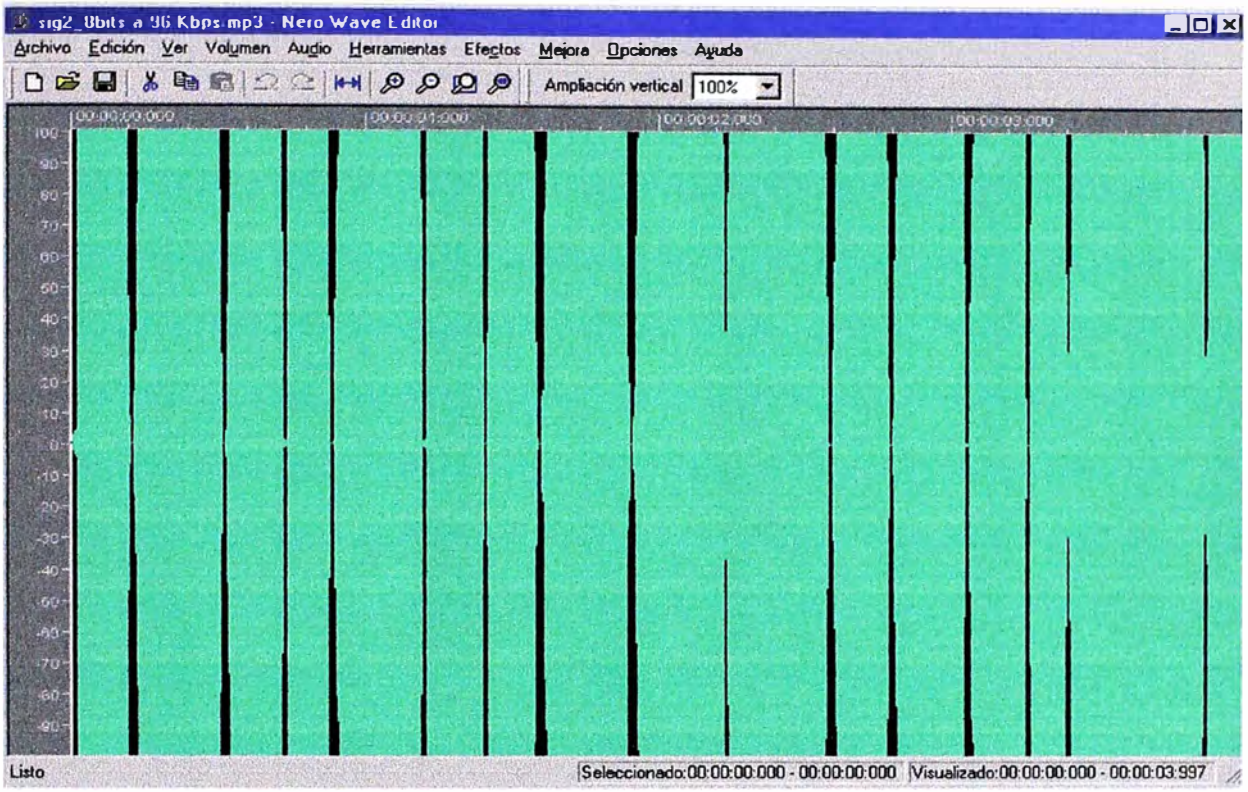


Figura PS49: **sig2_8bits_f=5120_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

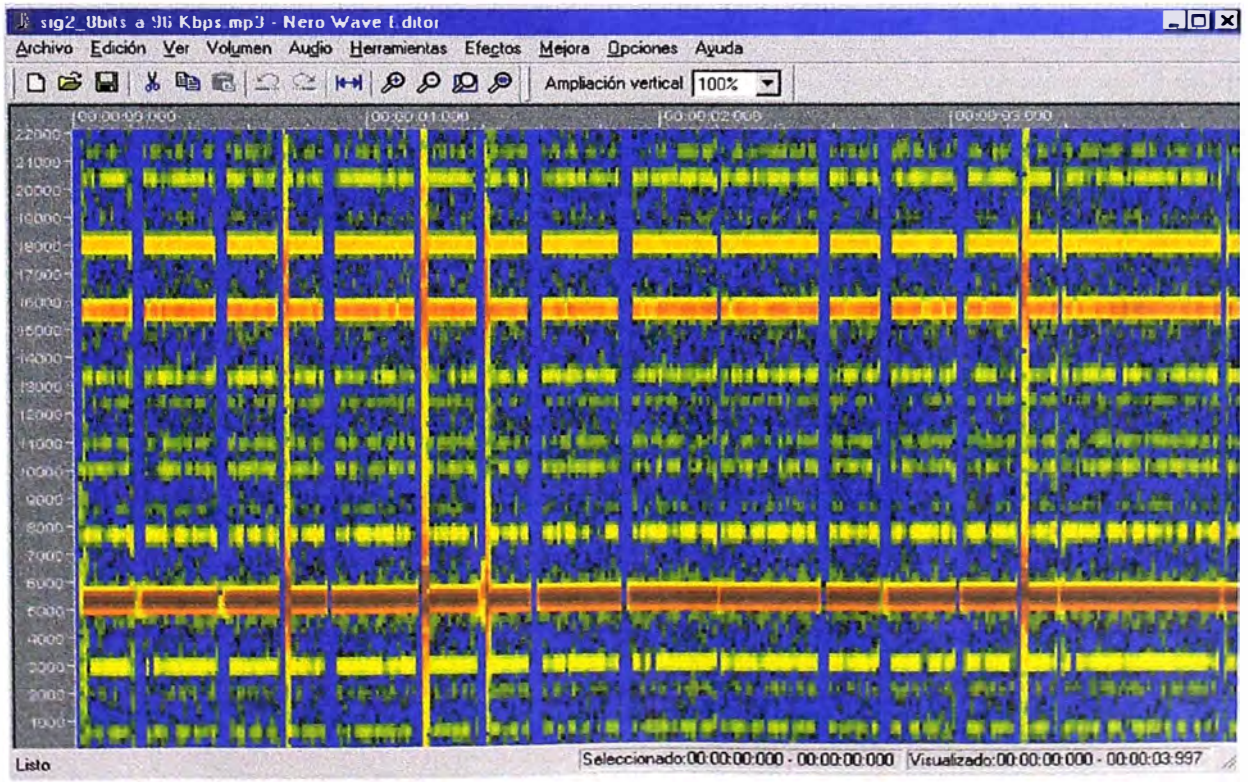
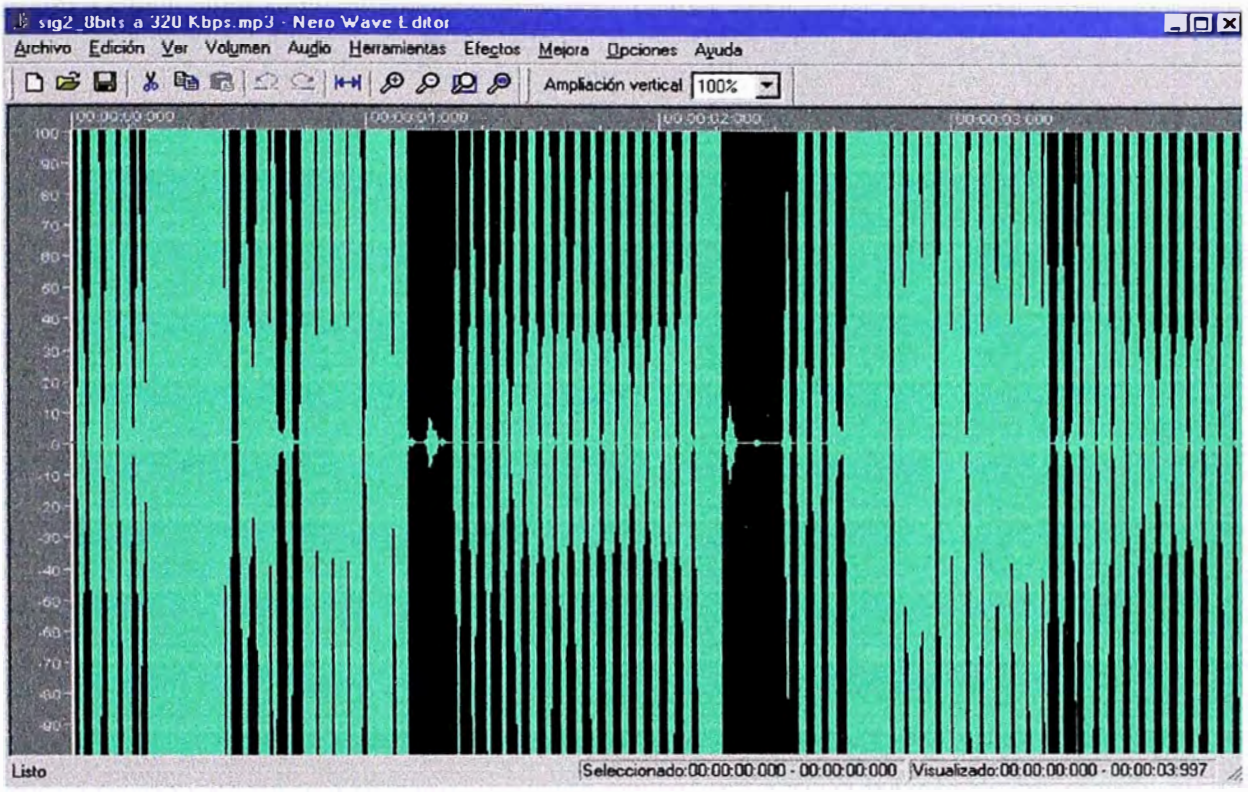


Figura PS50: **sig2_8bits a 96 Kbps_f=5120_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

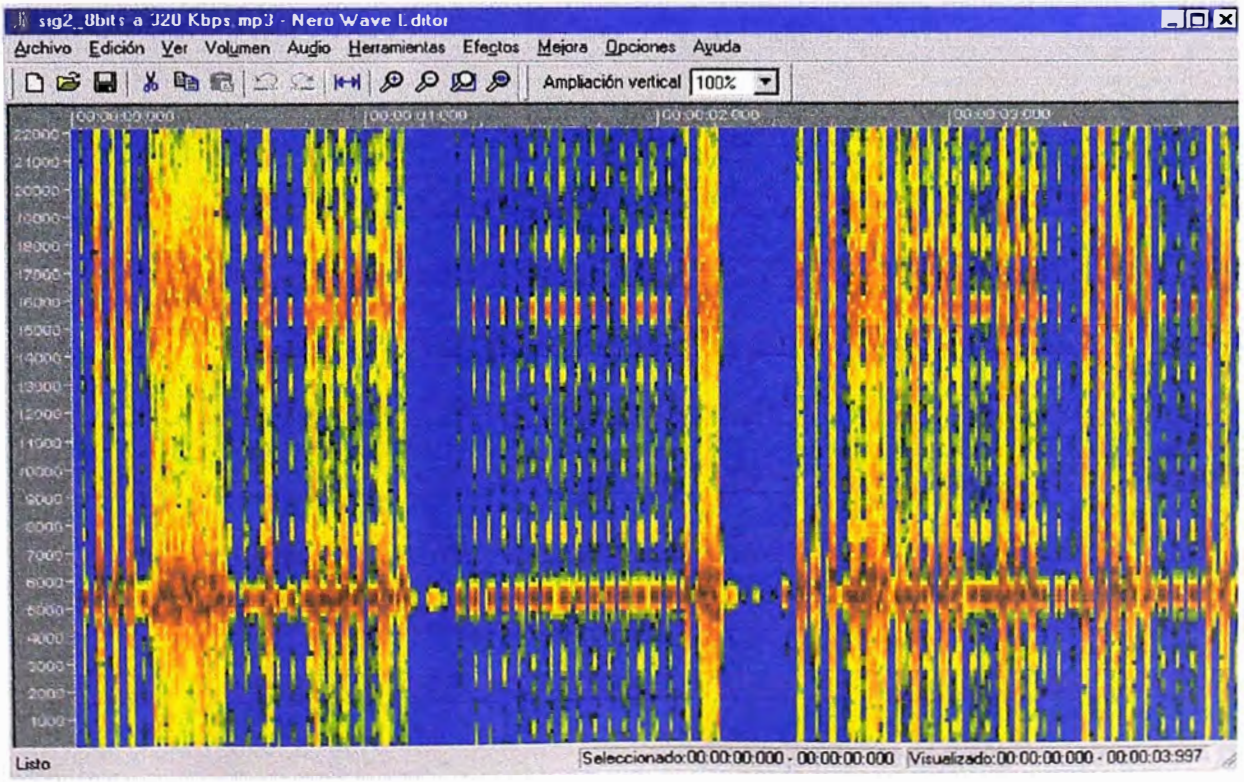
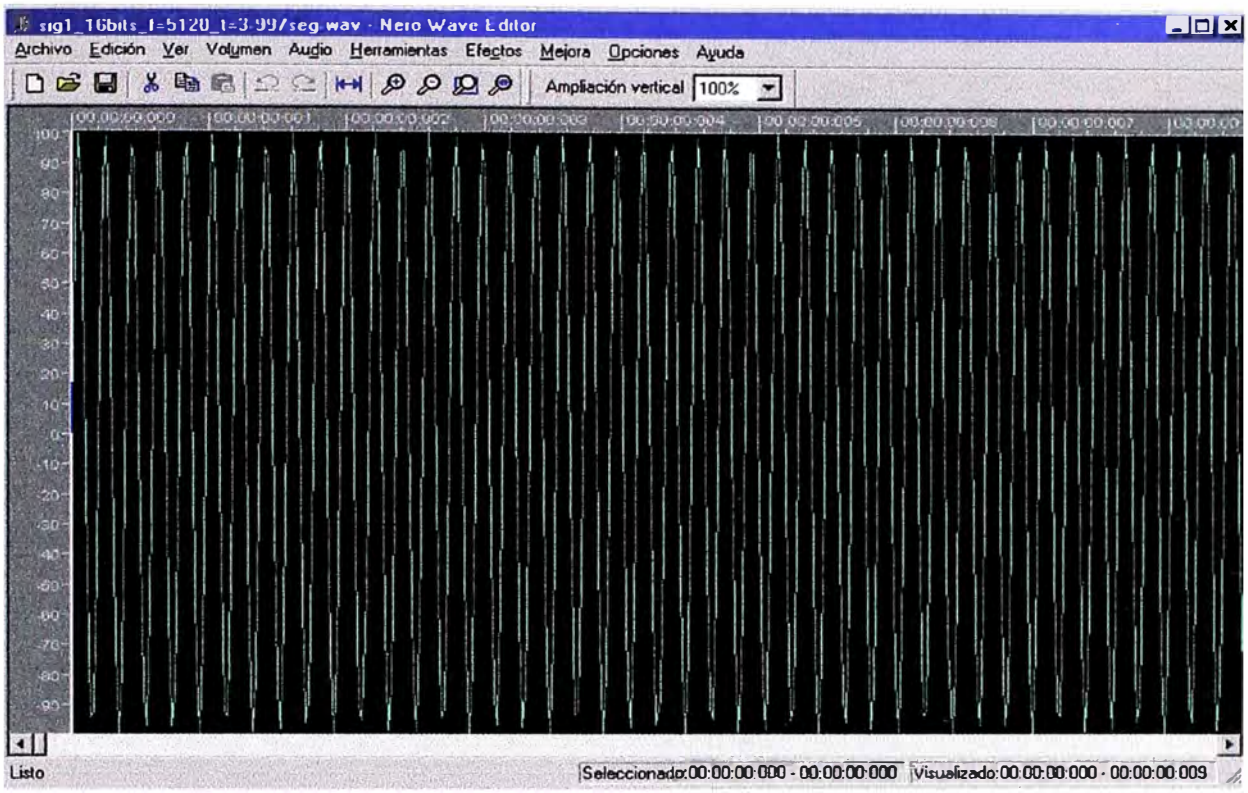


Figura PS51: **sig2_8bits a 320 Kbps_f=5120_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

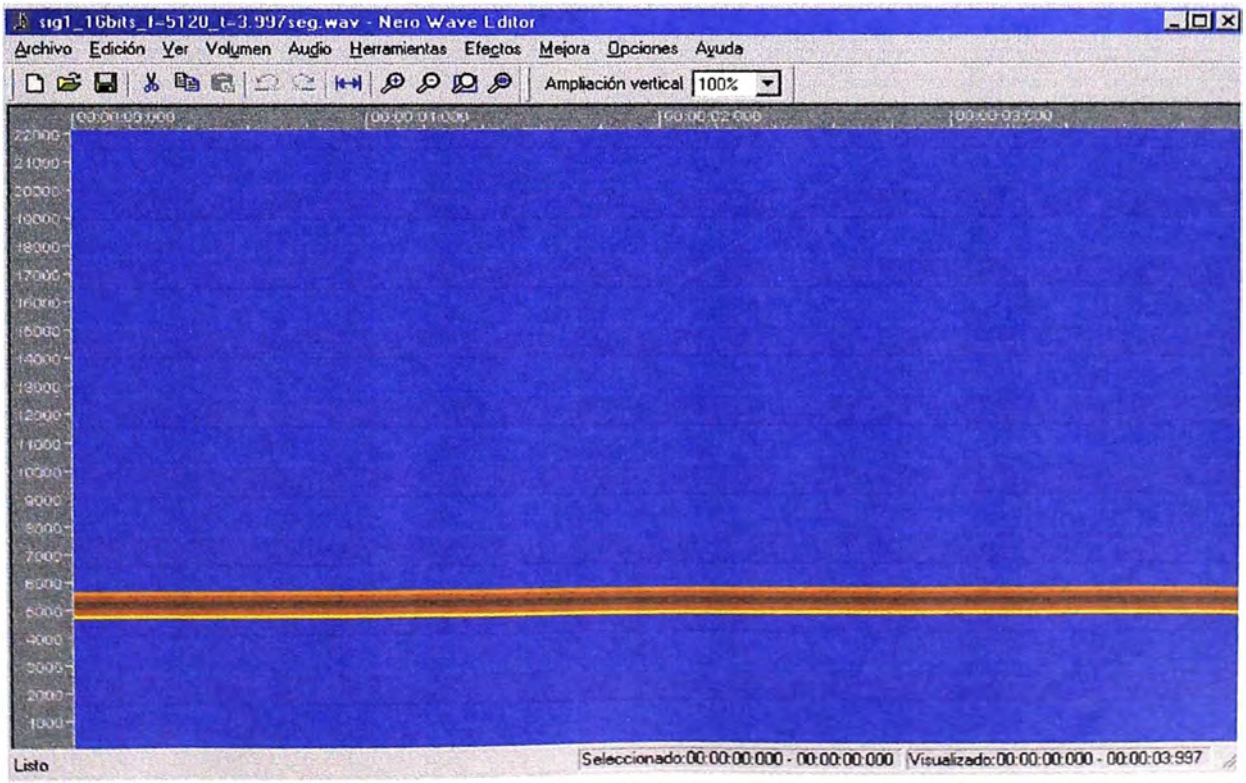
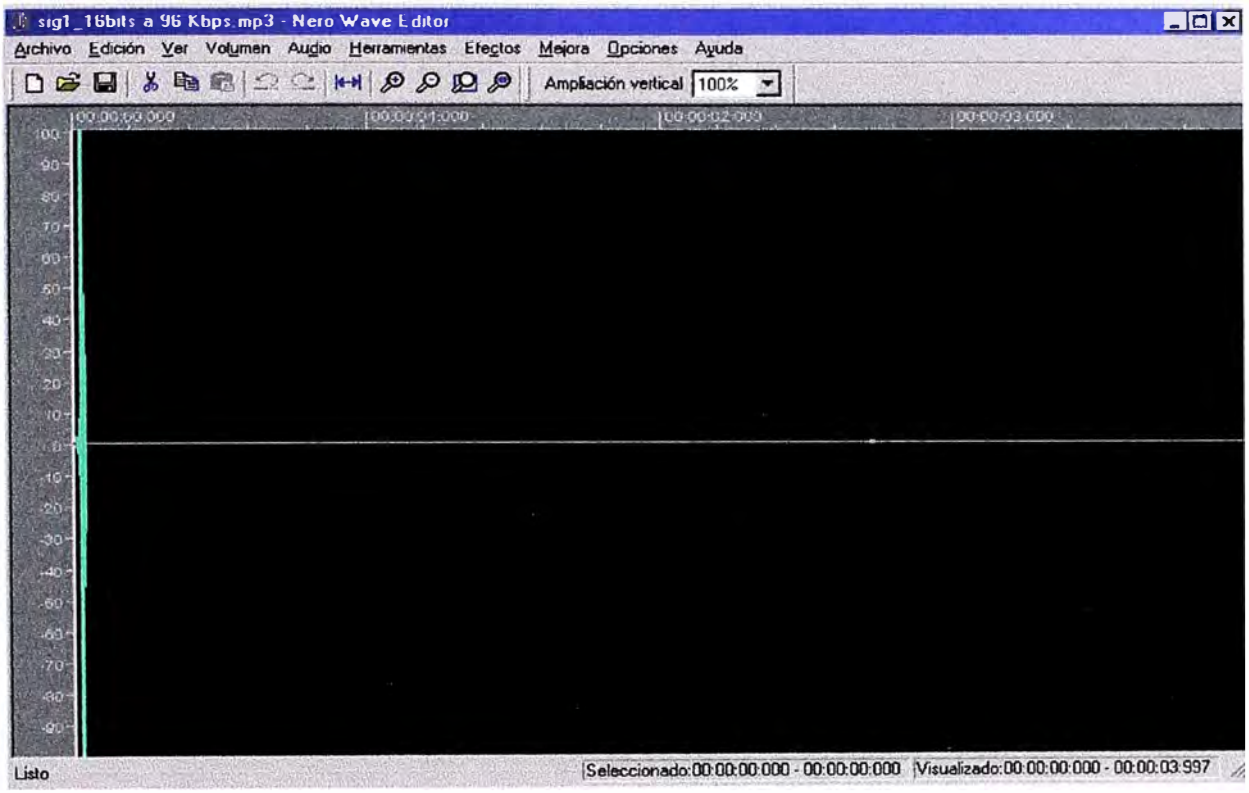


Figura PS52: **sig1_16bits_f=5120_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

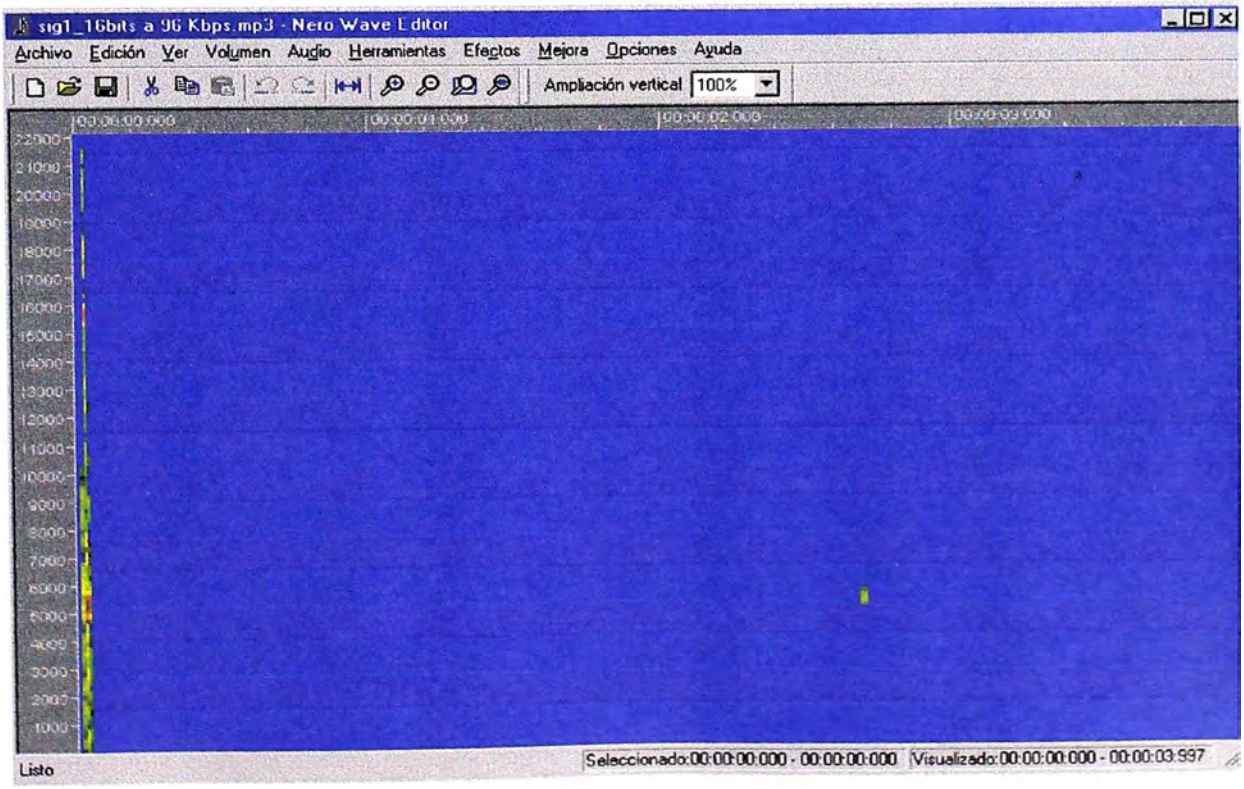
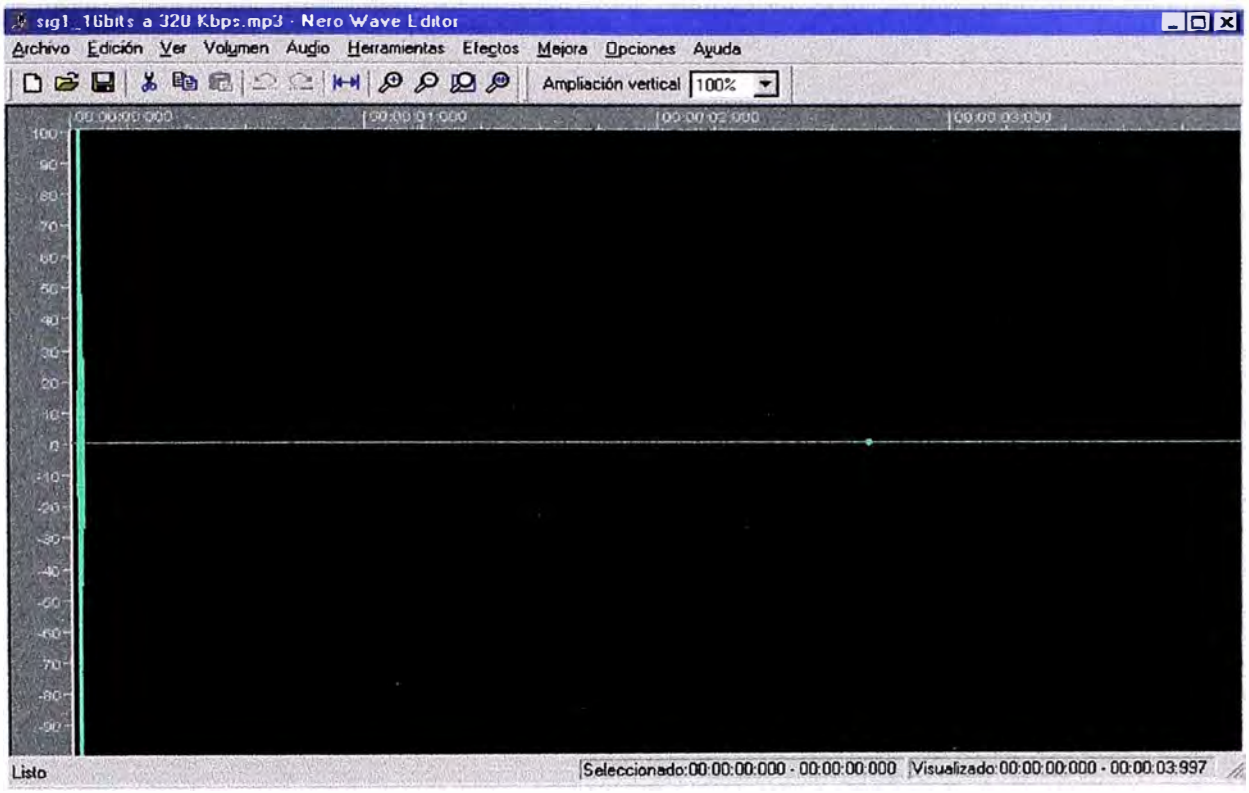


Figura PS53: **sig1_16bits a 96 Kbps_f=5120_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

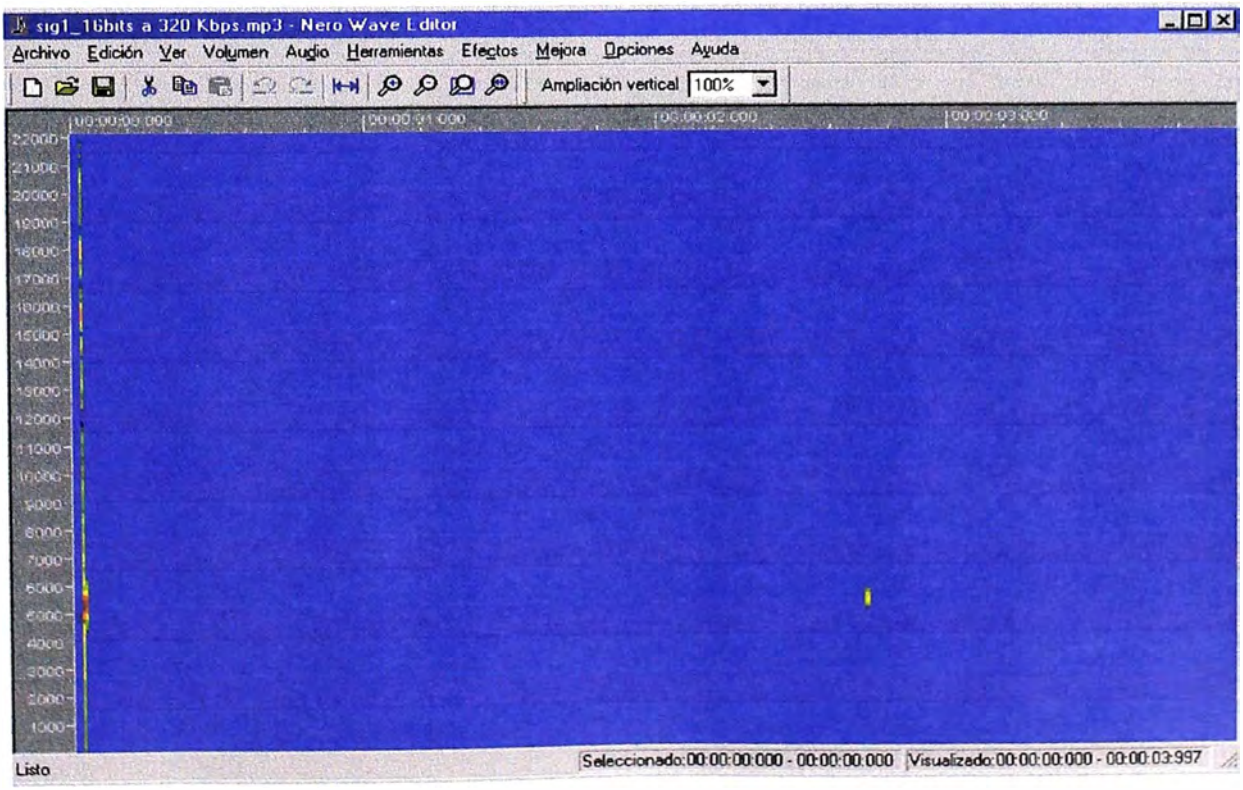
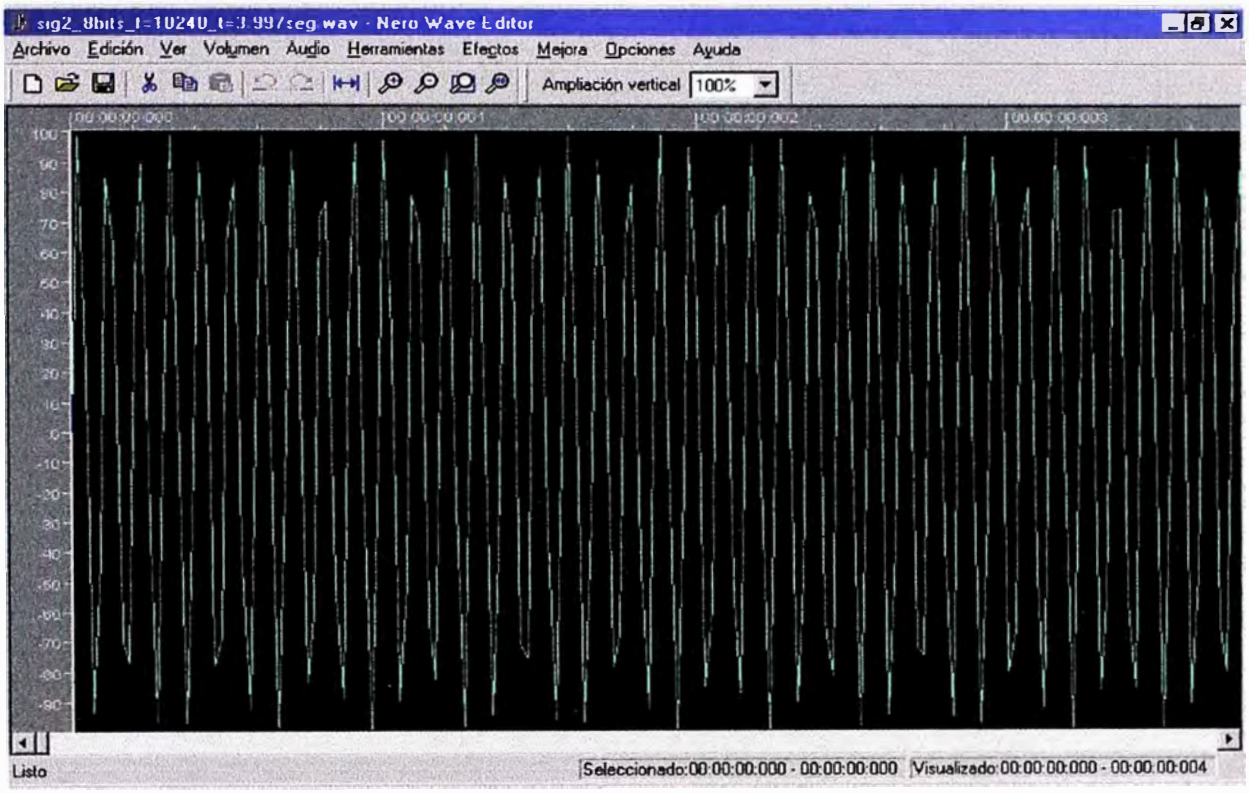


Figura PS54: **sig1_16bits a 320 Kbps_f=5120_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

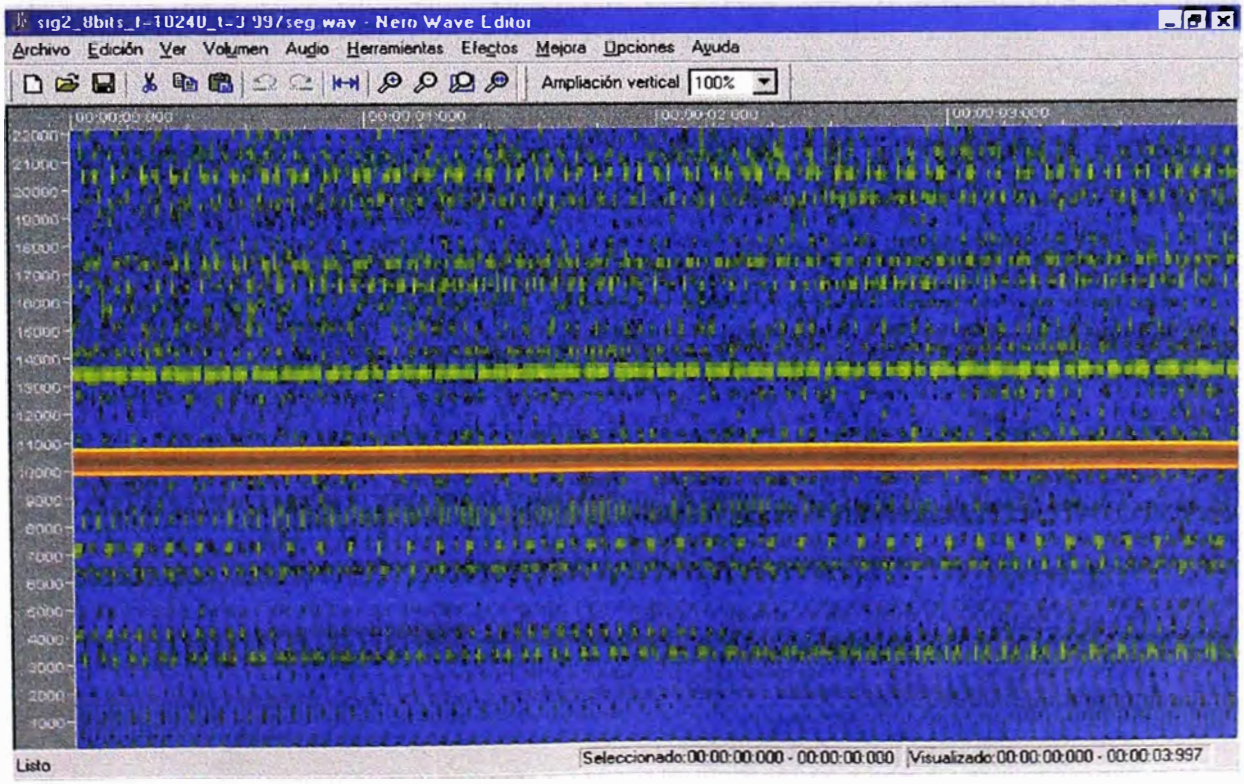
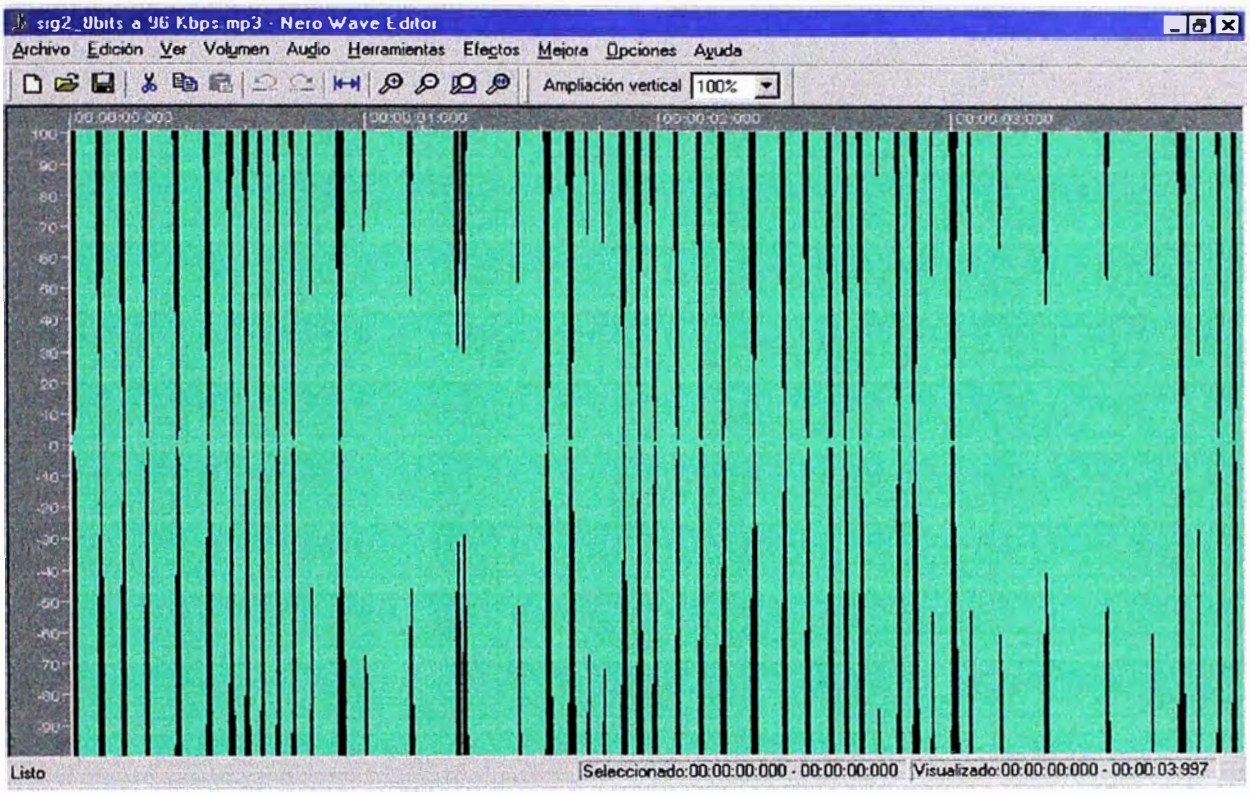


Figura PS55: **sig2_8bits_f=10240_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

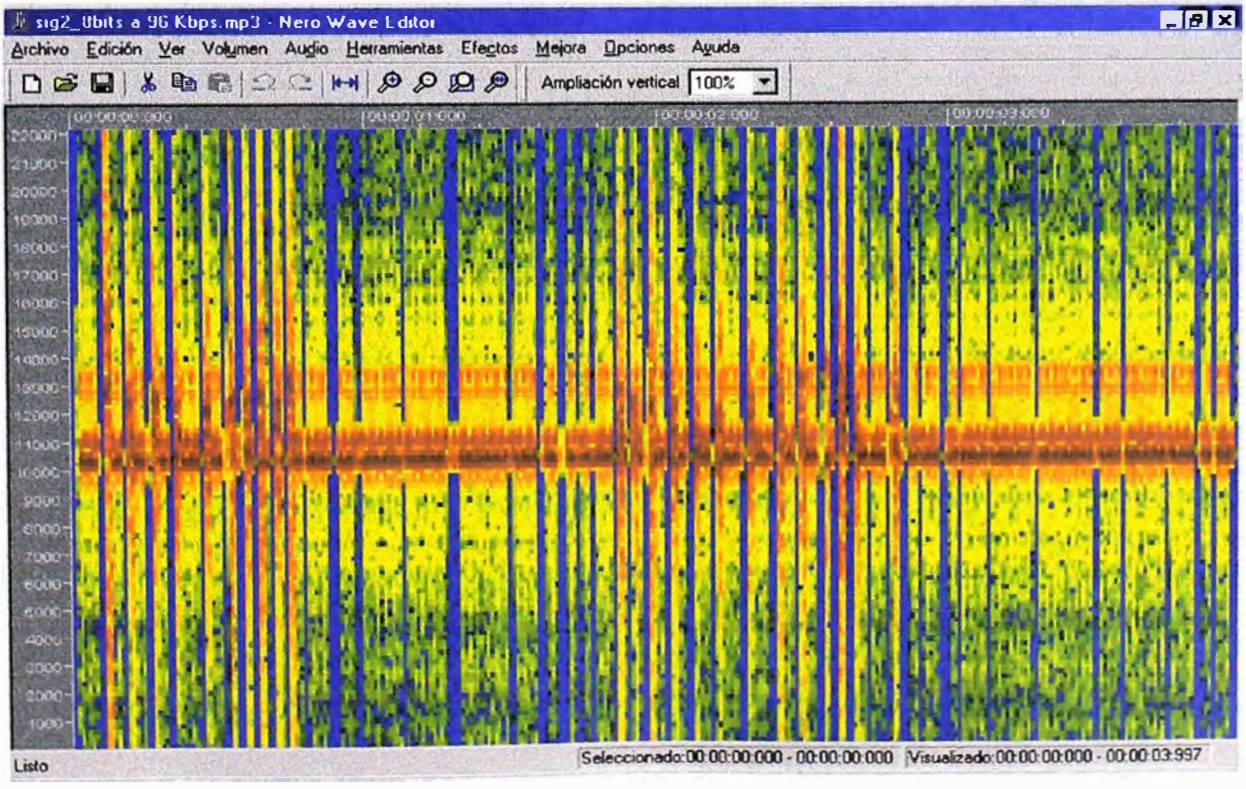
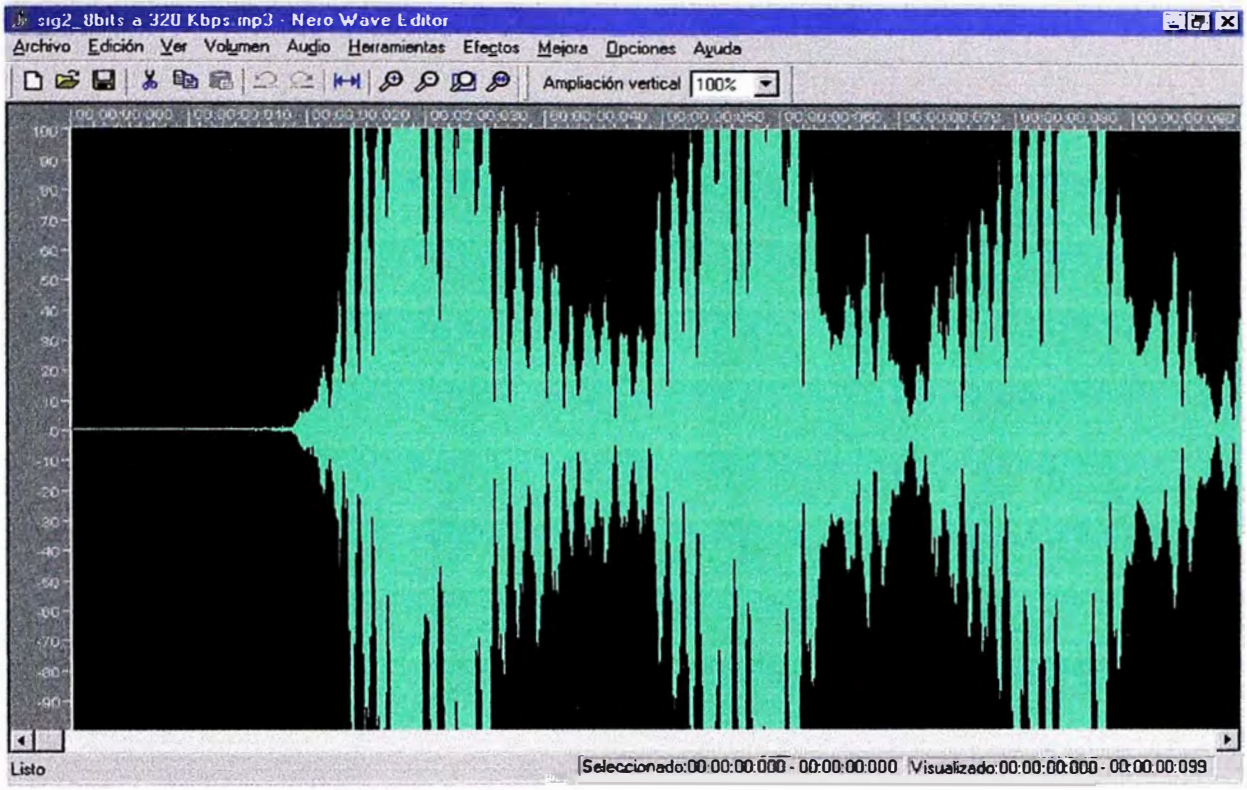


Figura PS56: **sig2_8bits a 96 Kbps_f=10240_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

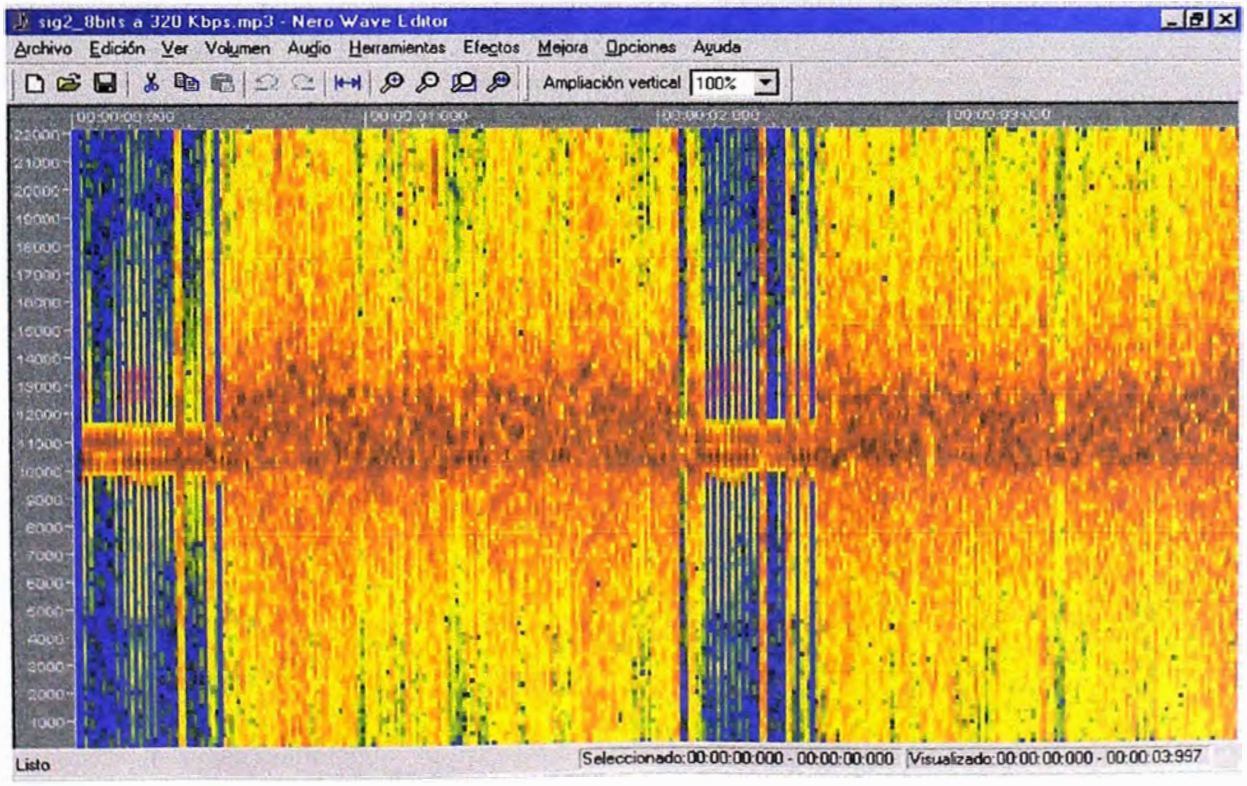
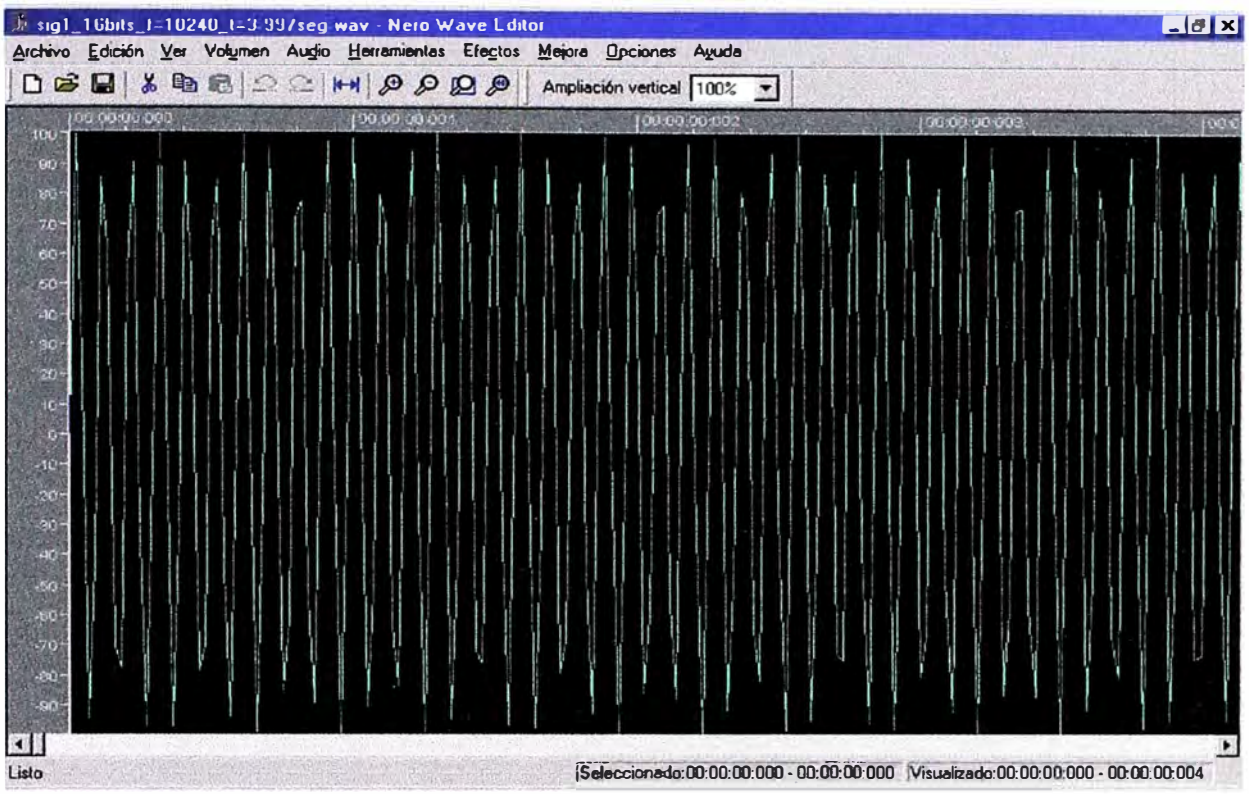


Figura PS57: **sig2_8bits a 320 Kbps_f=10240_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

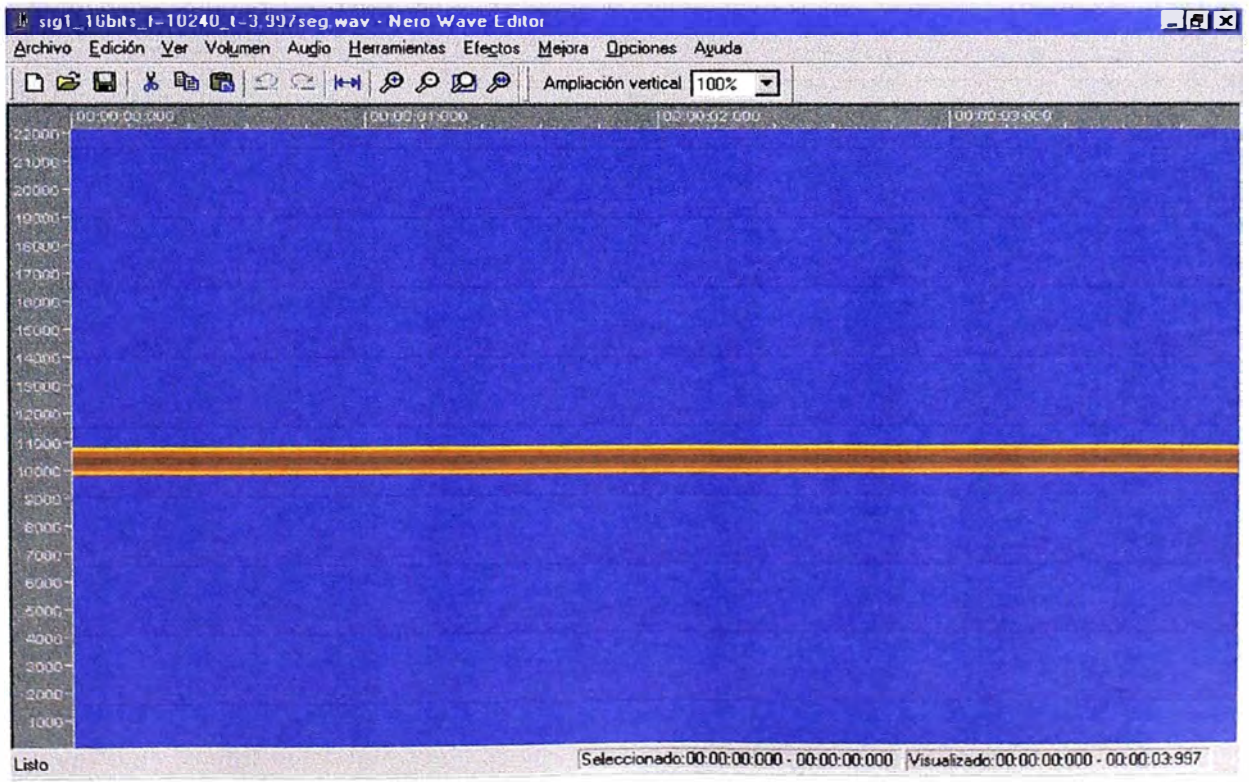
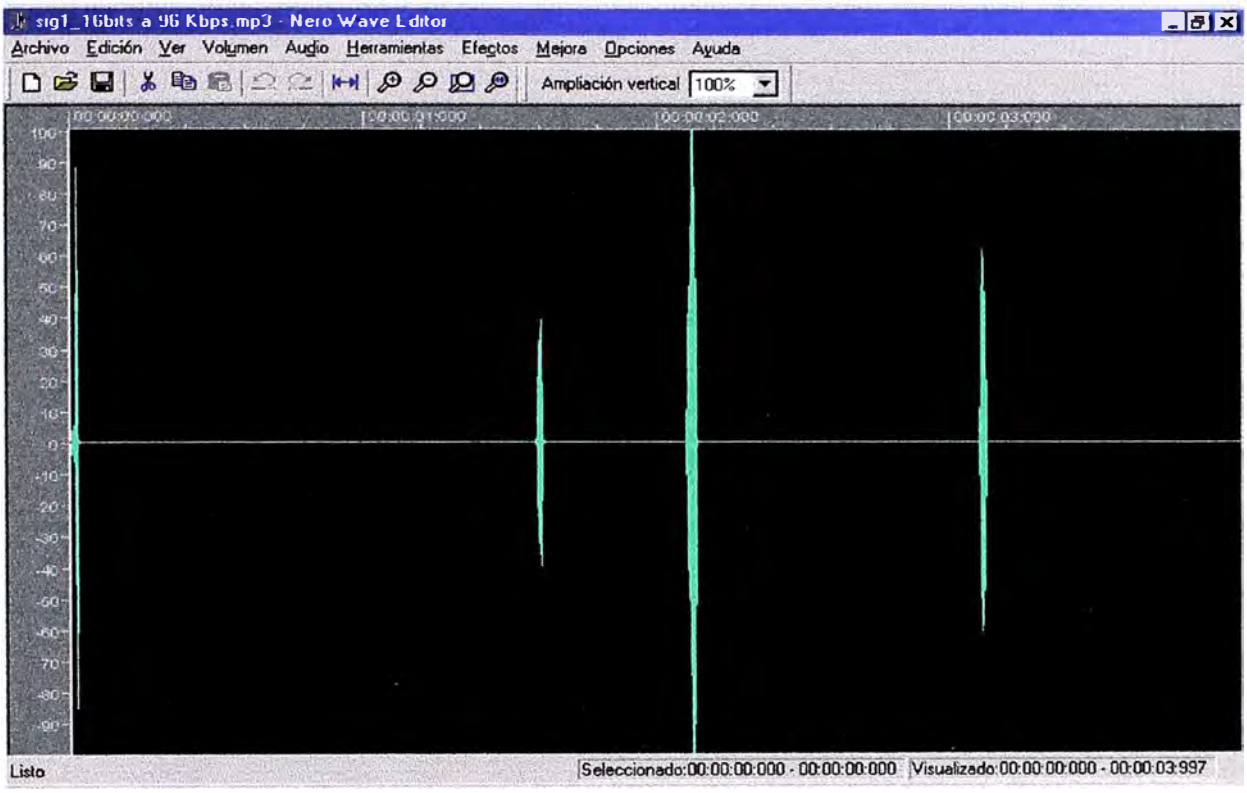


Figura PS58: **sig1_16bits_f=10240_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

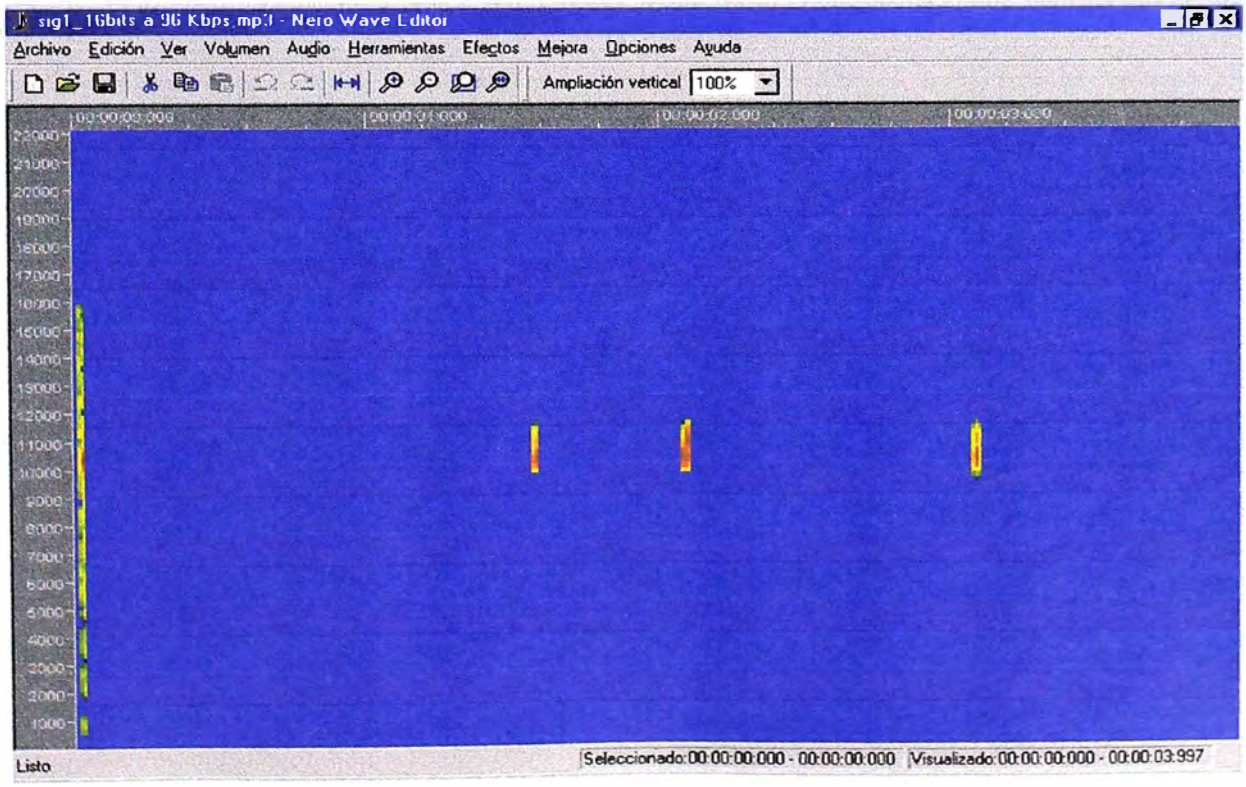
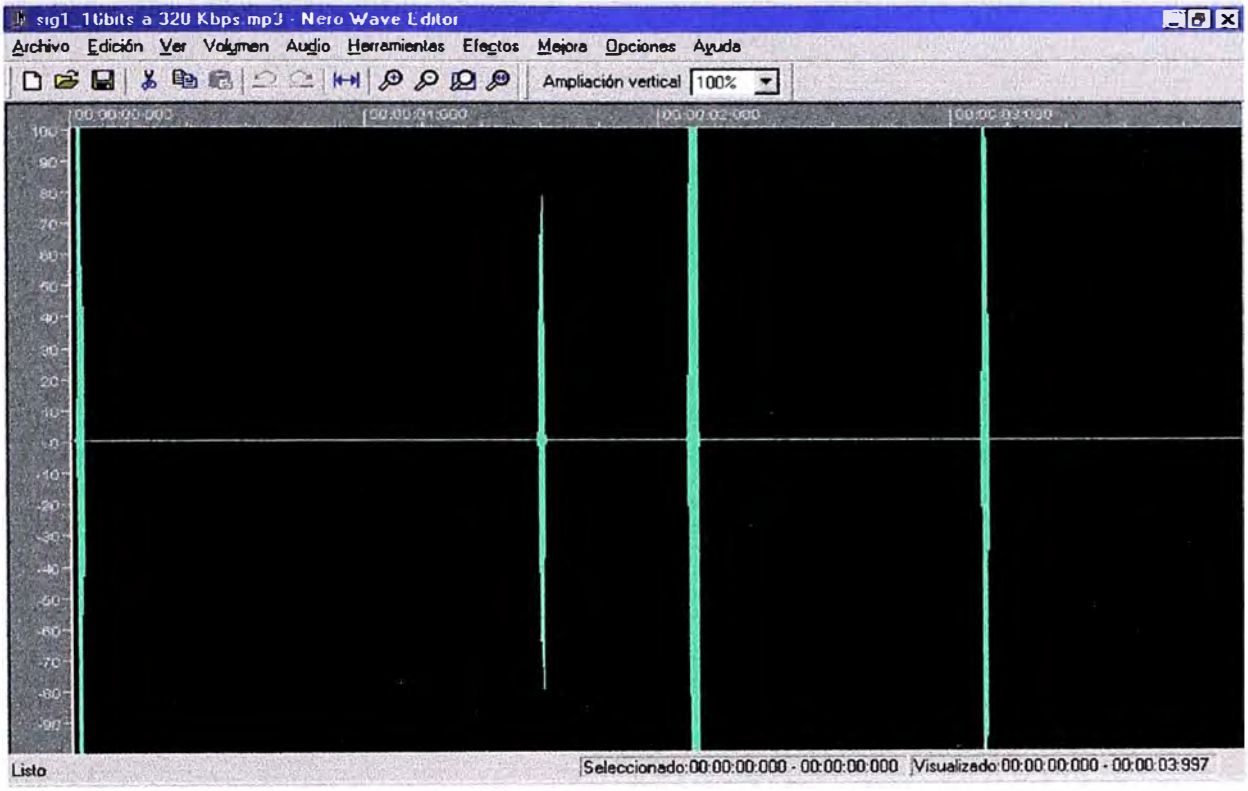


Figura PS59: **sig1_16bits a 96 Kbps_f=10240_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

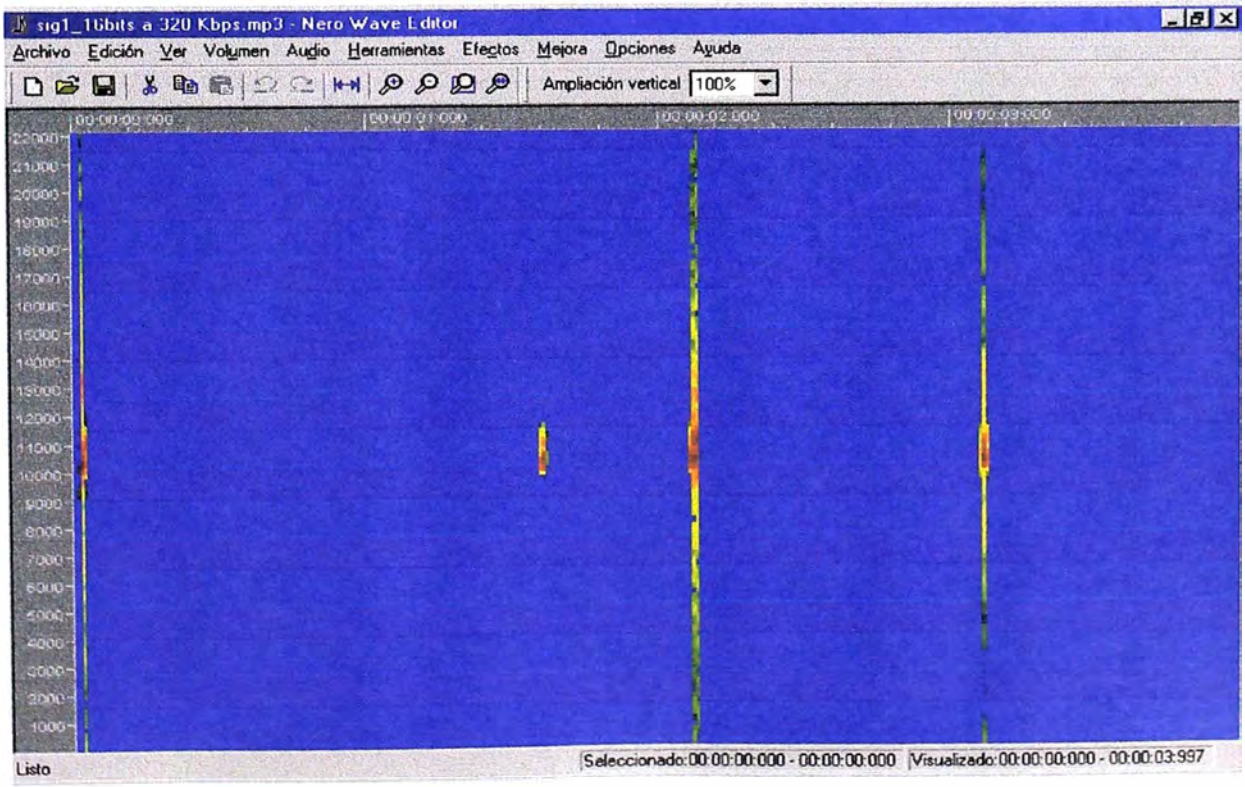
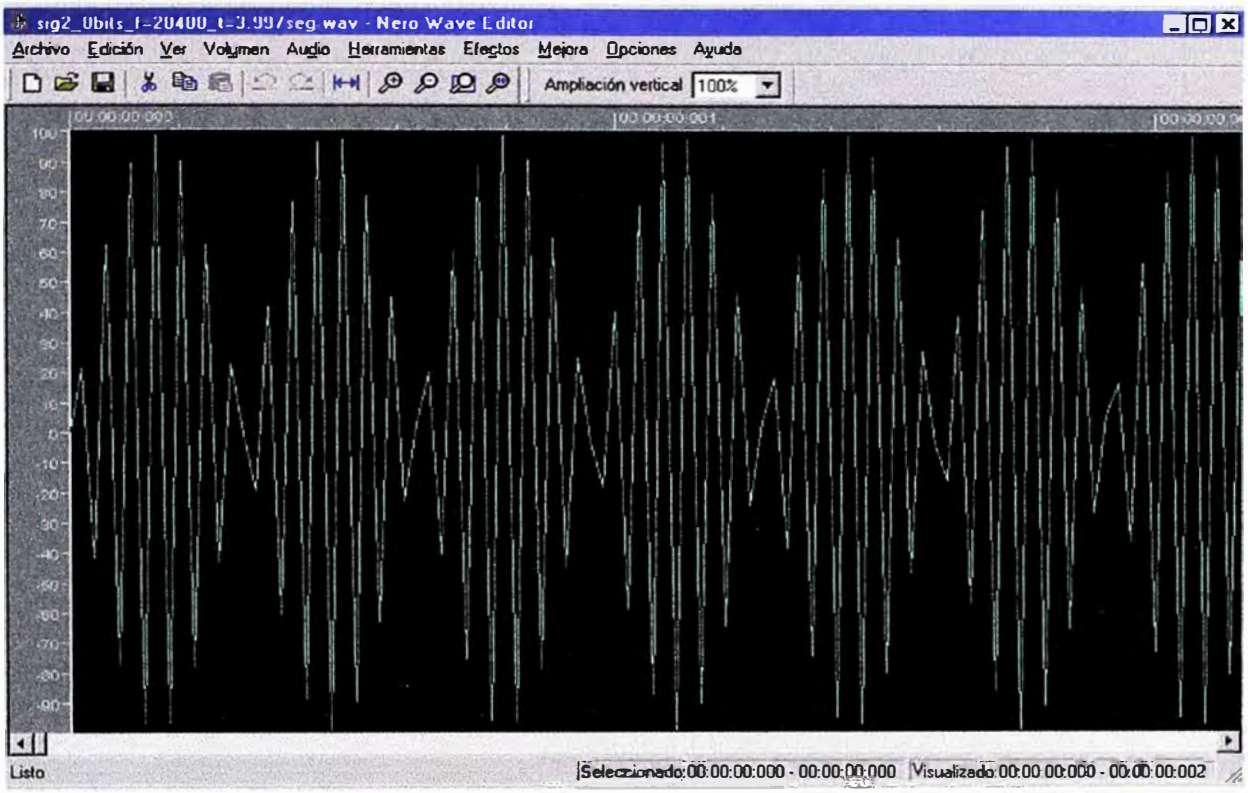


Figura PS60: **sig1_16bits a 320 Kbps_f=10240_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

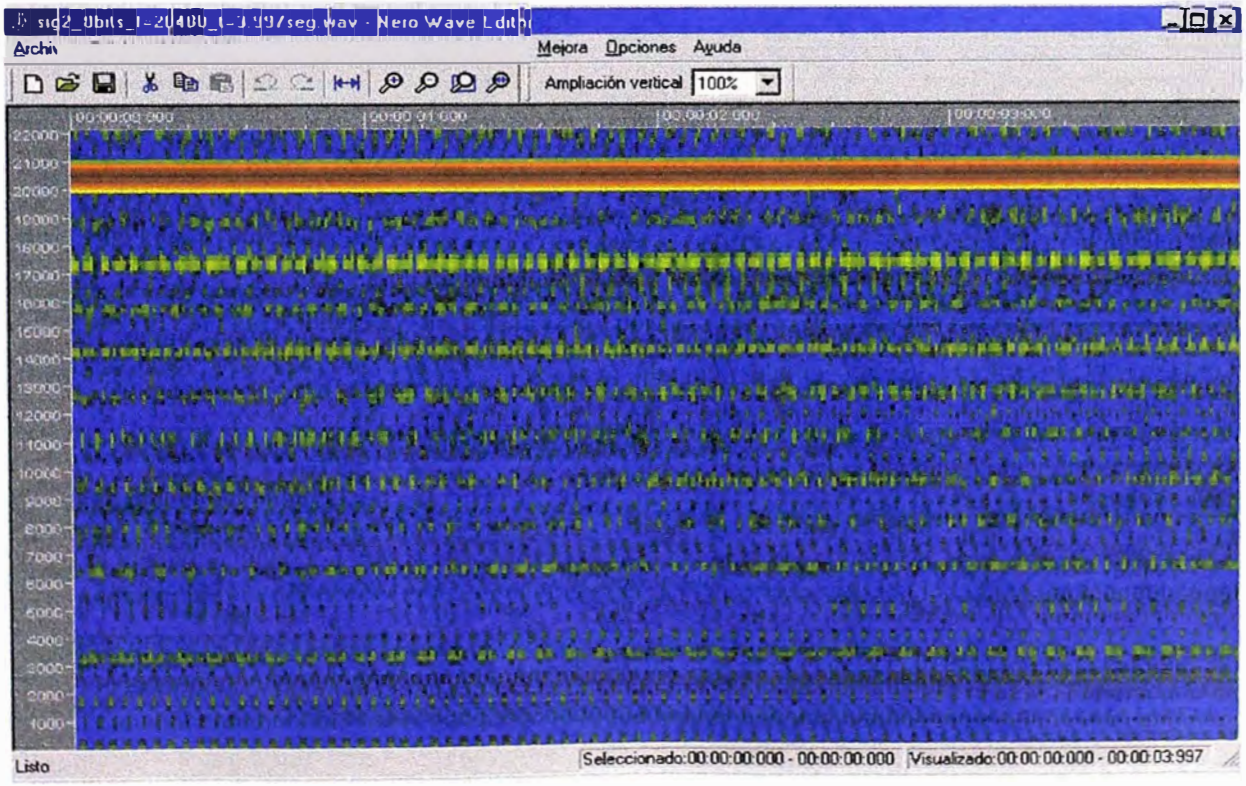
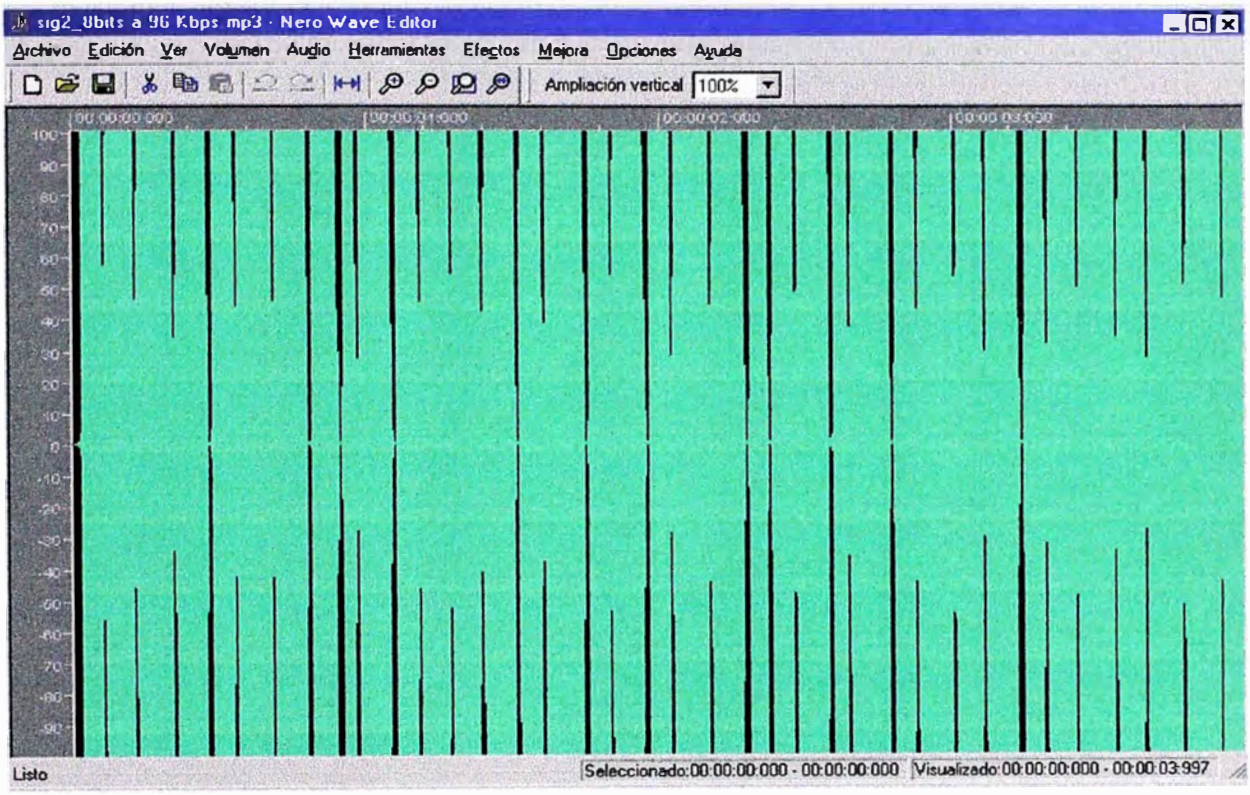


Figura PS61: **sig2_8bits_f=20480_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

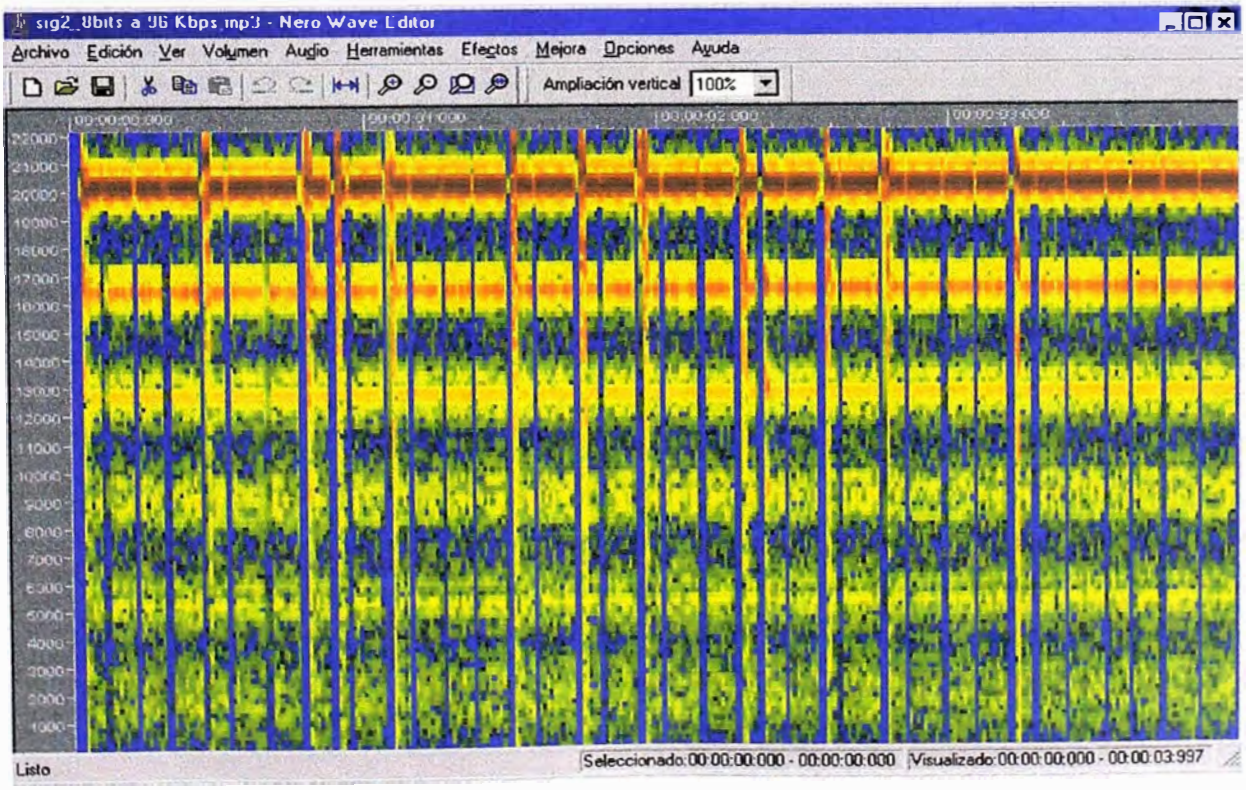
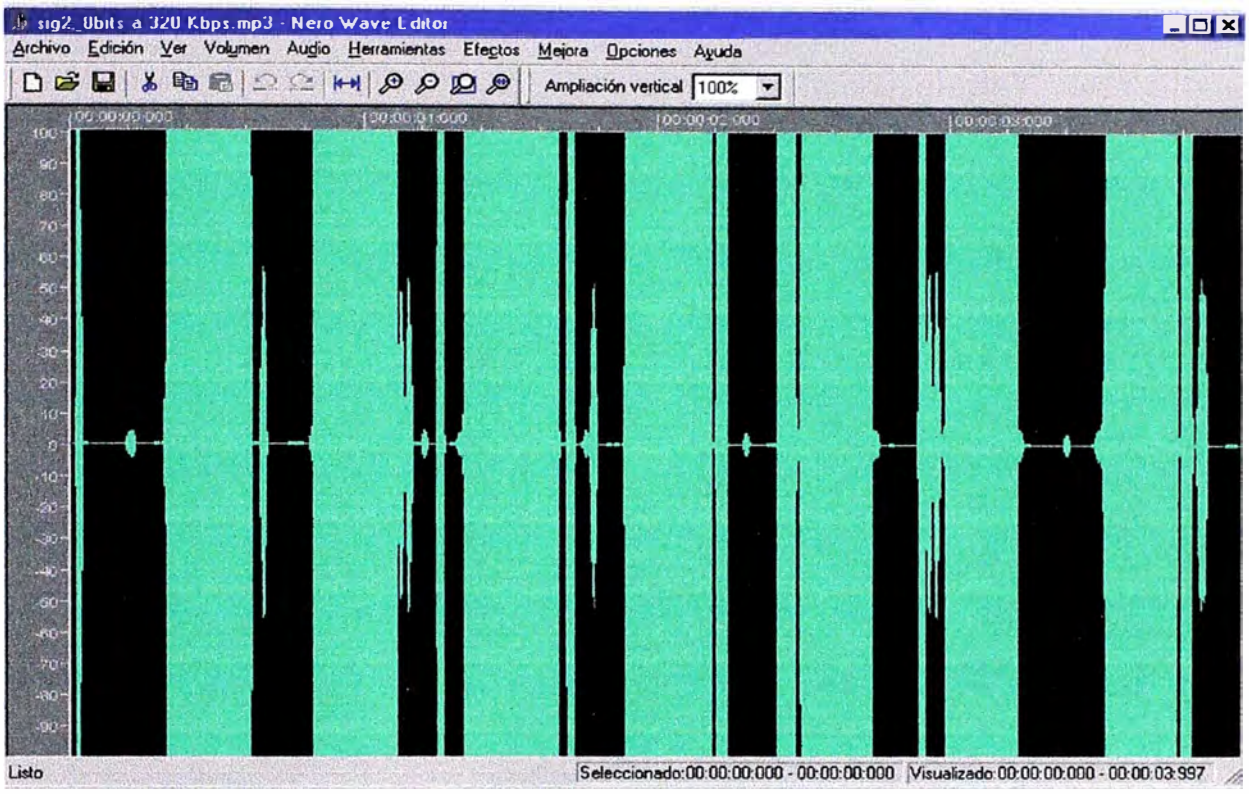


Figura PS62: **sig2_8bits a 96 Kbps_f=20480_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

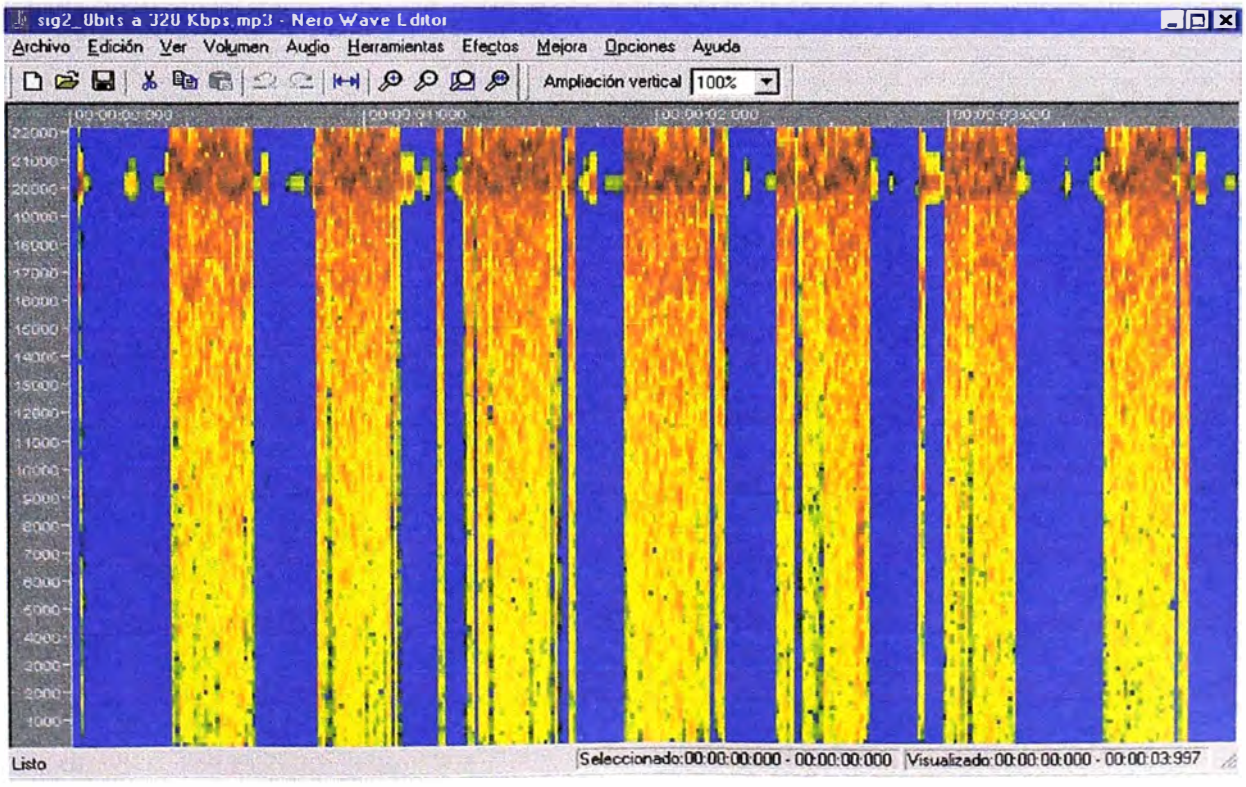
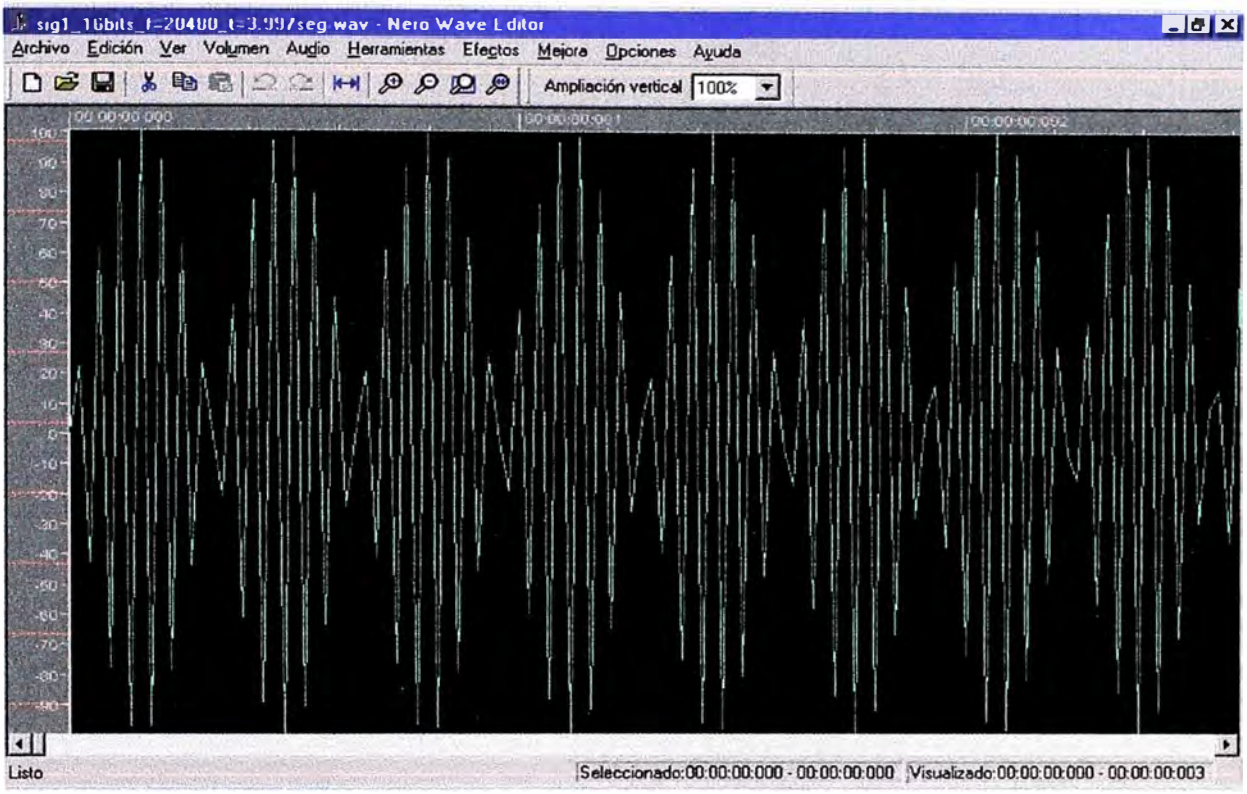


Figura PS63: **sig2_8bits a 320 Kbps_f=20480_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

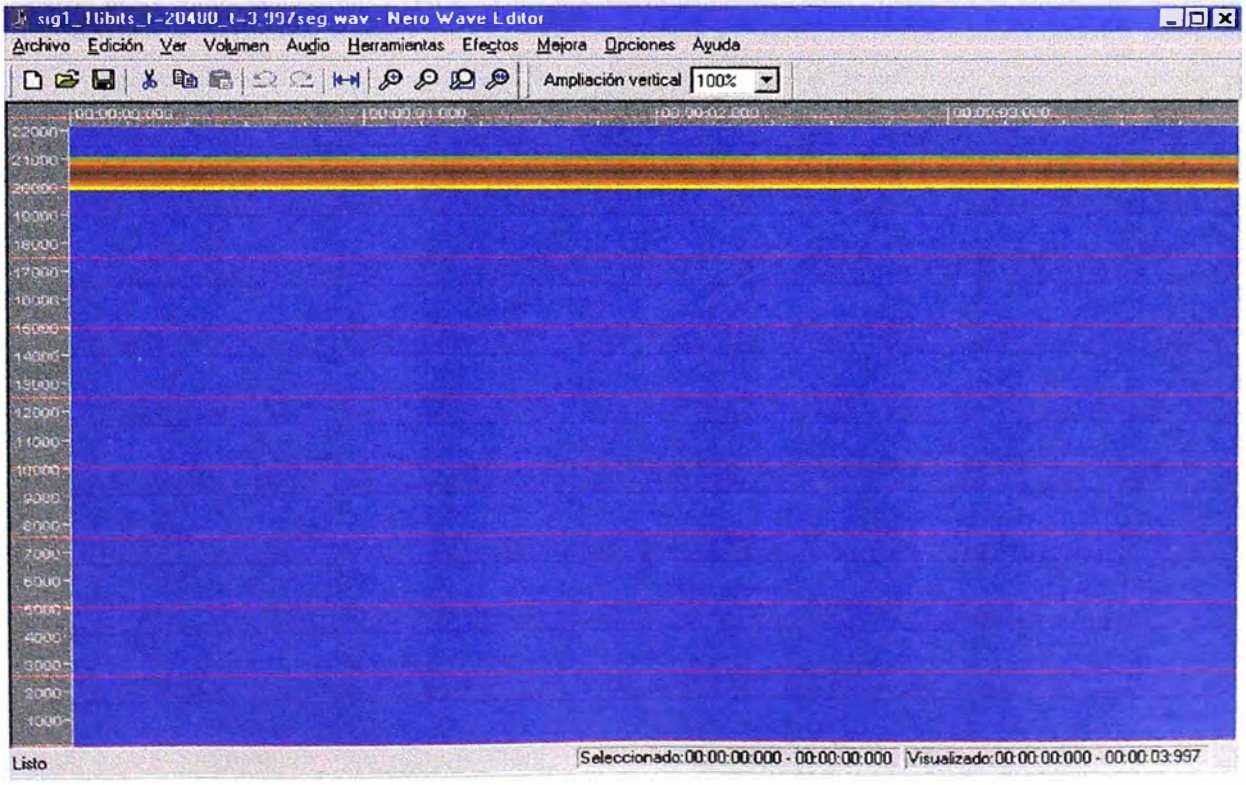
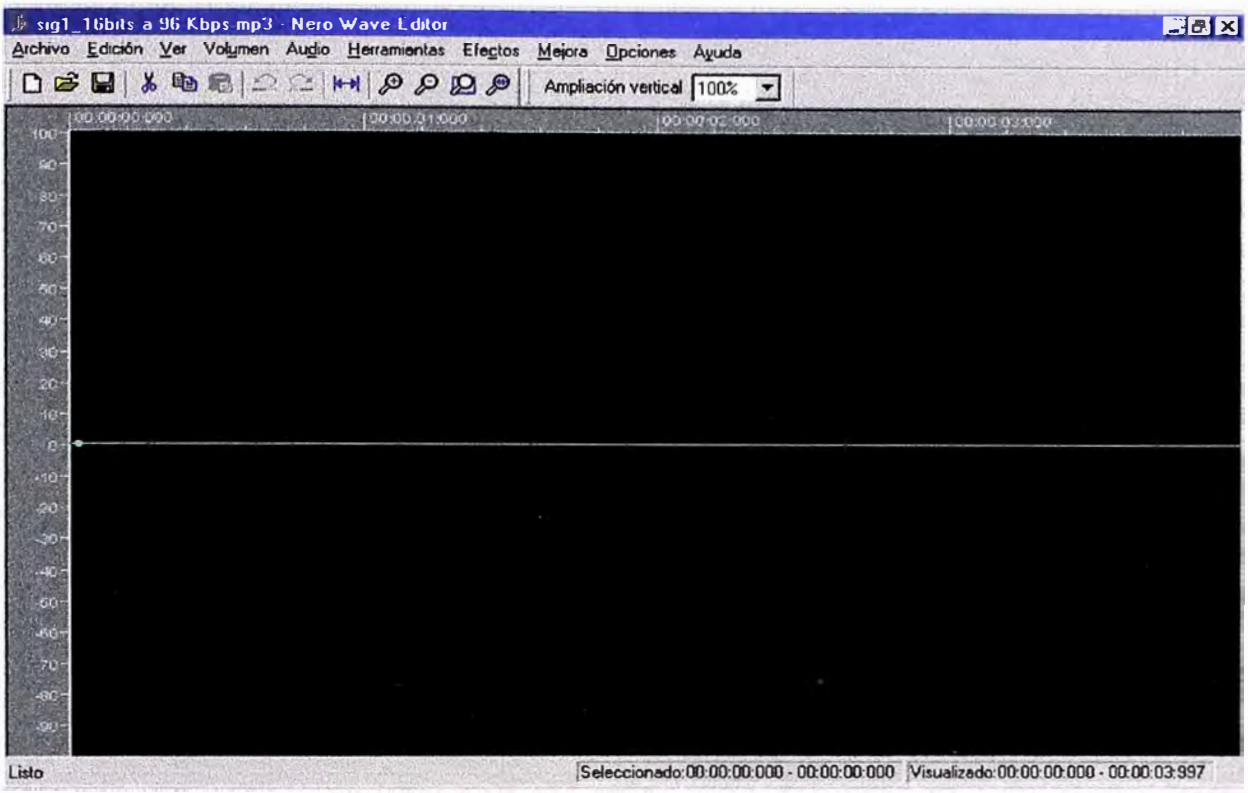


Figura PS64: **sig1_16bits_f=20480_t=3.997seg.wav**

Visualización de Onda



Visualización de Espectrograma

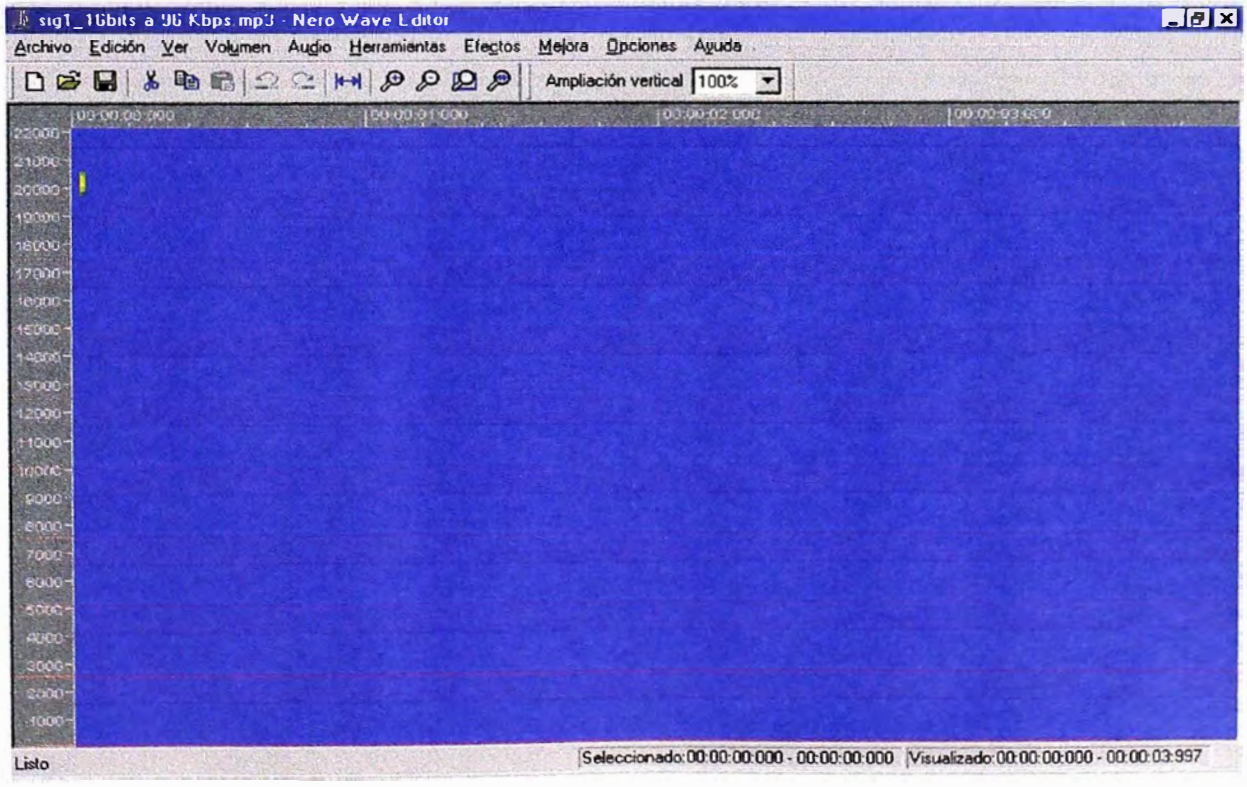
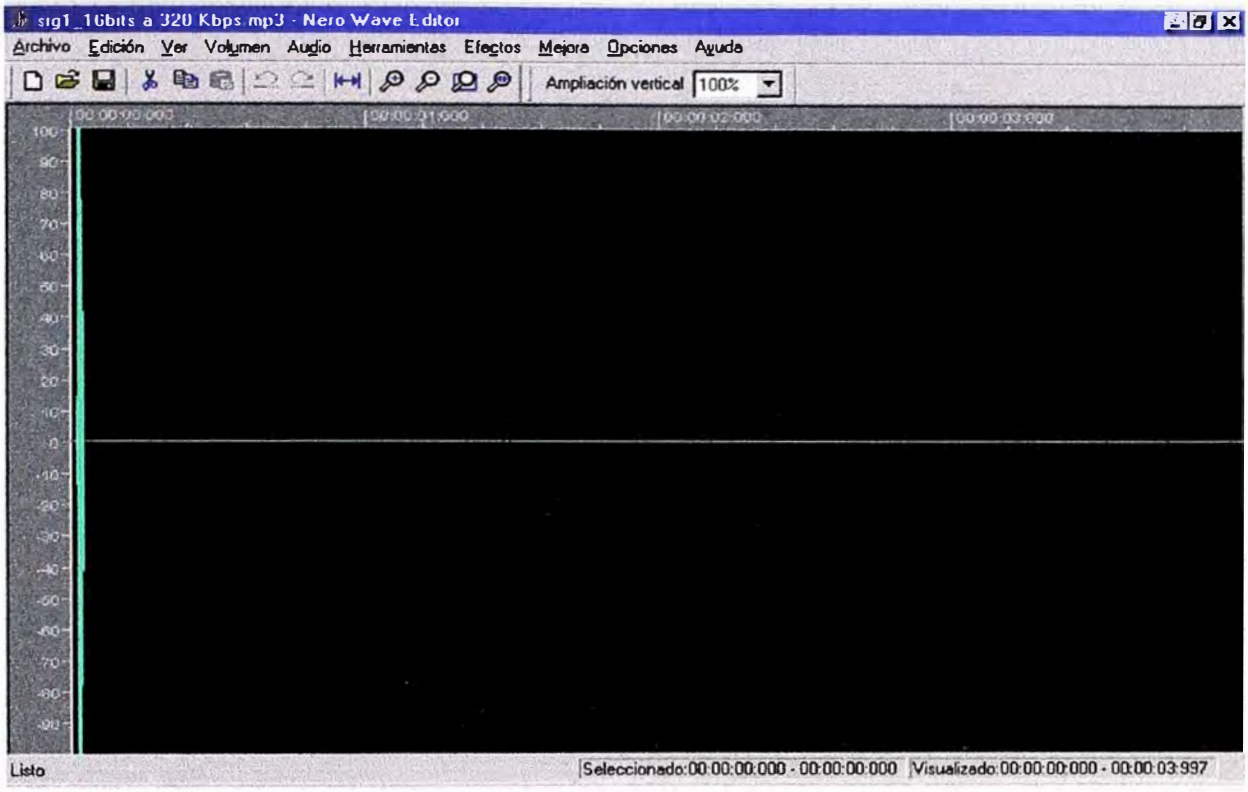


Figura PS65: **sig1_16bits a 96 Kbps_f=20480_t=3.997seg.mp3**

Visualización de Onda



Visualización de Espectrograma

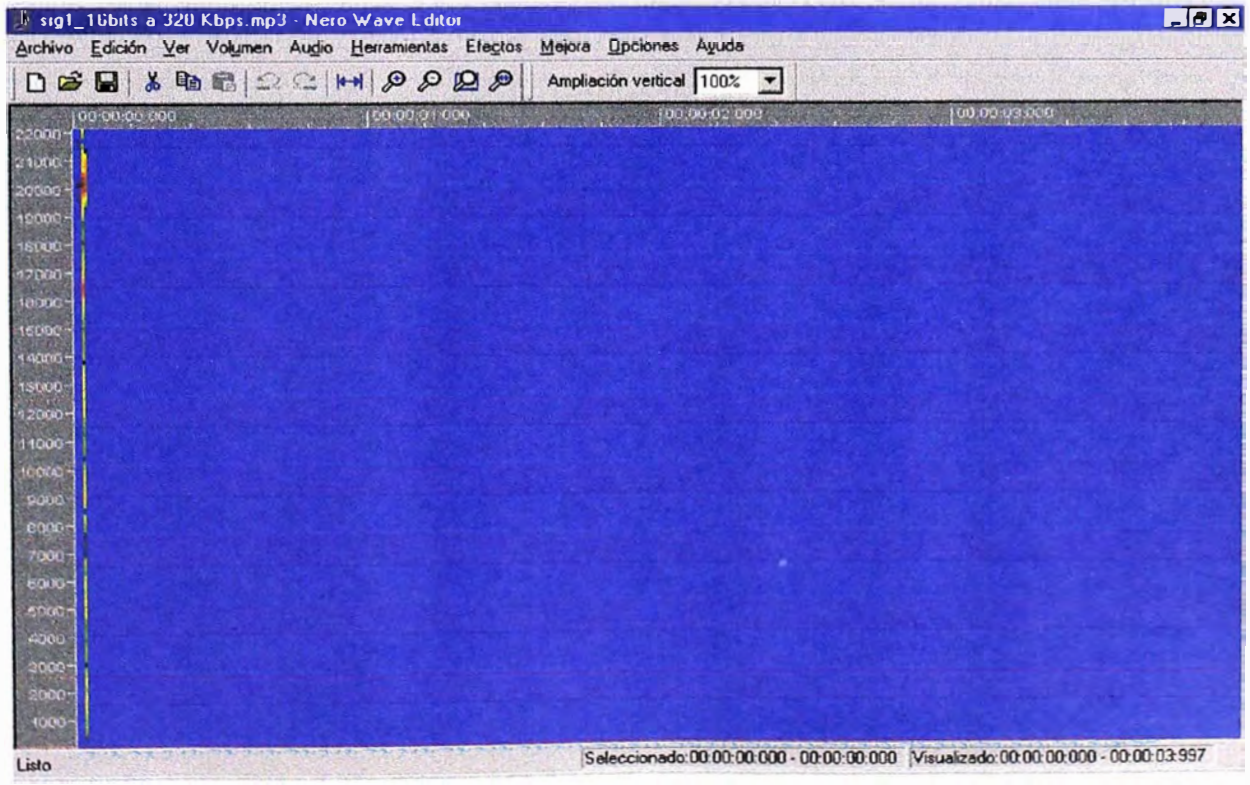


Figura PS66: **sig1_16bits a 320 Kbps_f=20480_t=3.997seg.mp3**

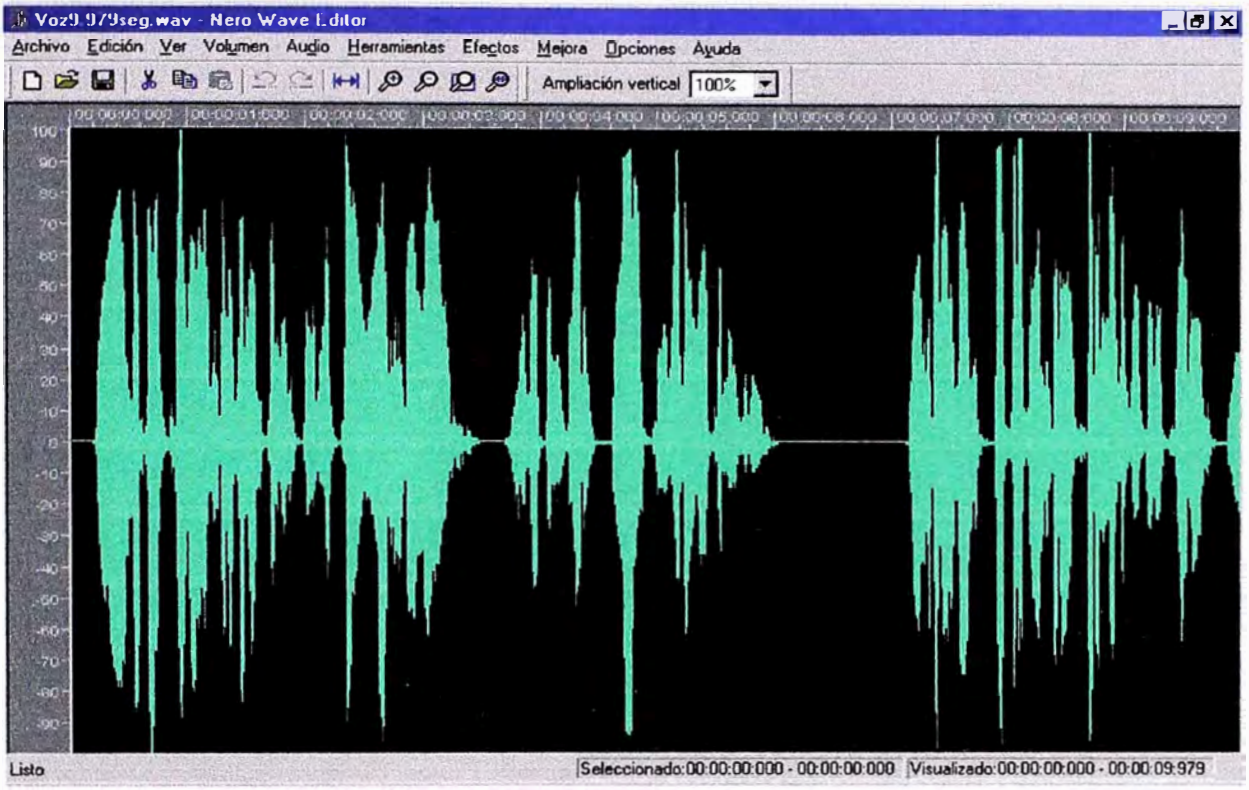
Señales Sinusoidales de 5 segundos de duración (con MP3 codificado a 4 segundos (3.997 segundos efectivos))												
Frecuencia	**WAV a 8 bits (KB)**		**MP3 (tamaño en KB)**		**Ejecución en minutos**		**WAV a 16 bits (KB)**		**MP3 (tamaño en KB)**		**Ejecución en minutos**	
(Hz)	**(5 seg)**	**(3.997 seg)**	**96 Kbps**	**320 Kbps**	**96 Kbps**	**320 Kbps**	**(5 seg)**	**(3.997 seg)**	**96 Kbps**	**320 Kbps**	**96 Kbps**	**320 Kbps**
20	216	173	47	157	11	28	431	345	47	157	10	21
40	216	173	47	157	14	24	431	345	47	157	9	20
80	216	173	47	157	10	28	431	345	47	157	10	21
160	216	173	47	157	10	24	431	345	47	157	9	20
320	216	173	47	157	10	26	431	345	47	157	10	22
640	216	173	47	157	11	24	431	345	47	157	10	20
1280	216	173	47	157	10	27	431	345	47	157	10	20
2560	216	173	47	157	10	24	431	345	47	157	12	23
5120	216	173	47	157	11	23	431	345	47	157	12	23
10240	216	173	47	157	11	23	431	345	47	157	11	22
20480	216	173	47	157	10	25	431	345	47	157	15	21
Para 20, 40, 80 y 20480 Hz no hubo sonido audible.												

Tasas de Compresión
Wav a 8 bits
Para 96 Kbps: 173/47 = 3.6808 ==> (1:3.6808)
Para 320 Kbps: 173/157 = 1.1019 ==> (1:1.1019)
Wav a 16 bits
Para 96 Kbps: 345/47 = 7.3404 ==> (1:7.3404)
Para 320 Kbps: 345/157 = 2.1974 ==> (1:2.1974)

Tabla Pruebas 2: Resultados de las pruebas para las señales sinusoidales.

Voces

Visualización de Onda



Visualización de Espectrograma

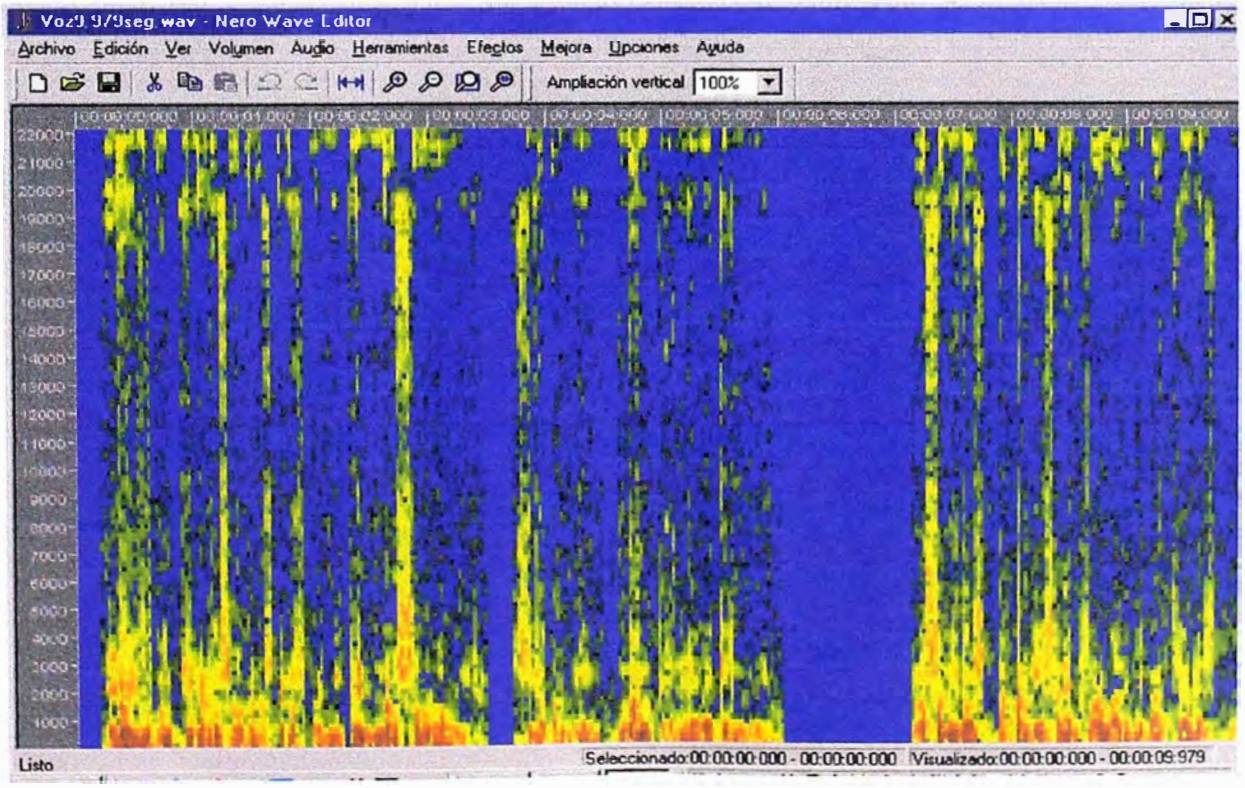
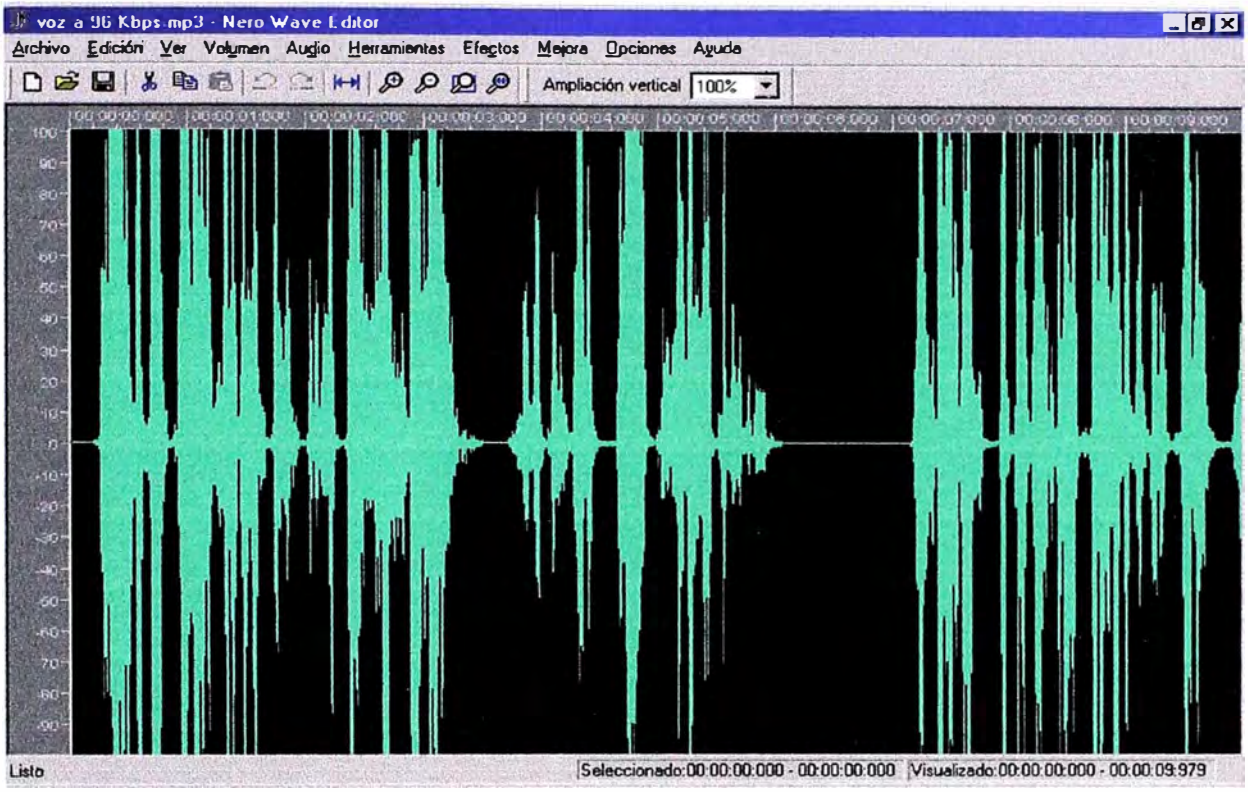


Figura PV1: **Voz9.979seg.wav**

Visualización de Onda



Visualización de Espectrograma

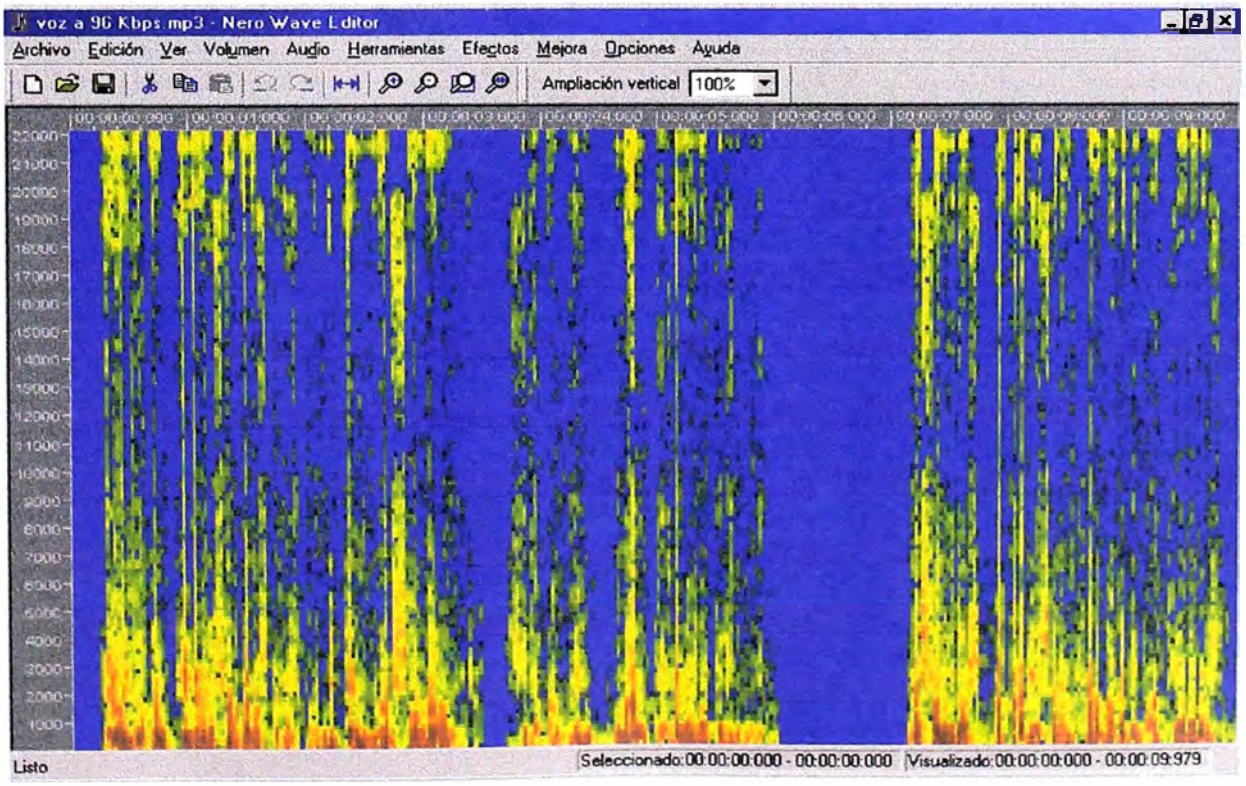
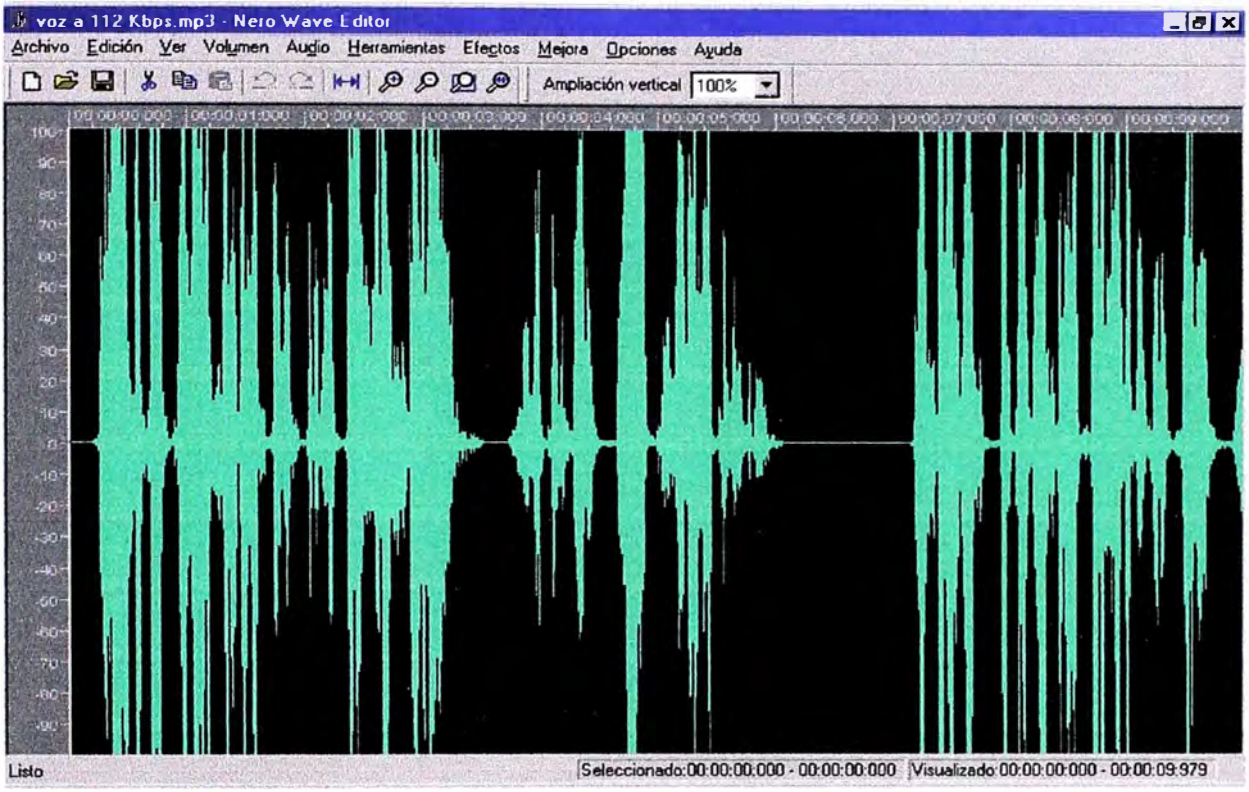


Figura PV2: **Voz a 96 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

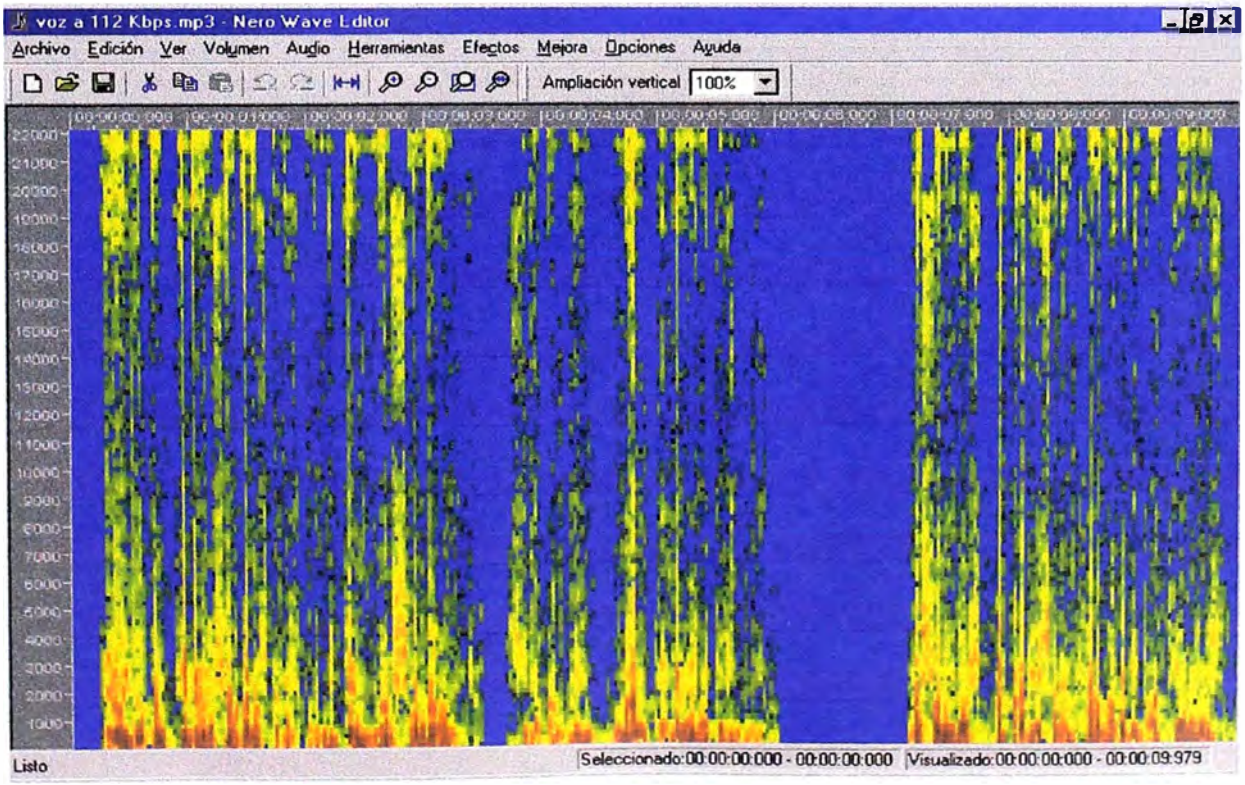
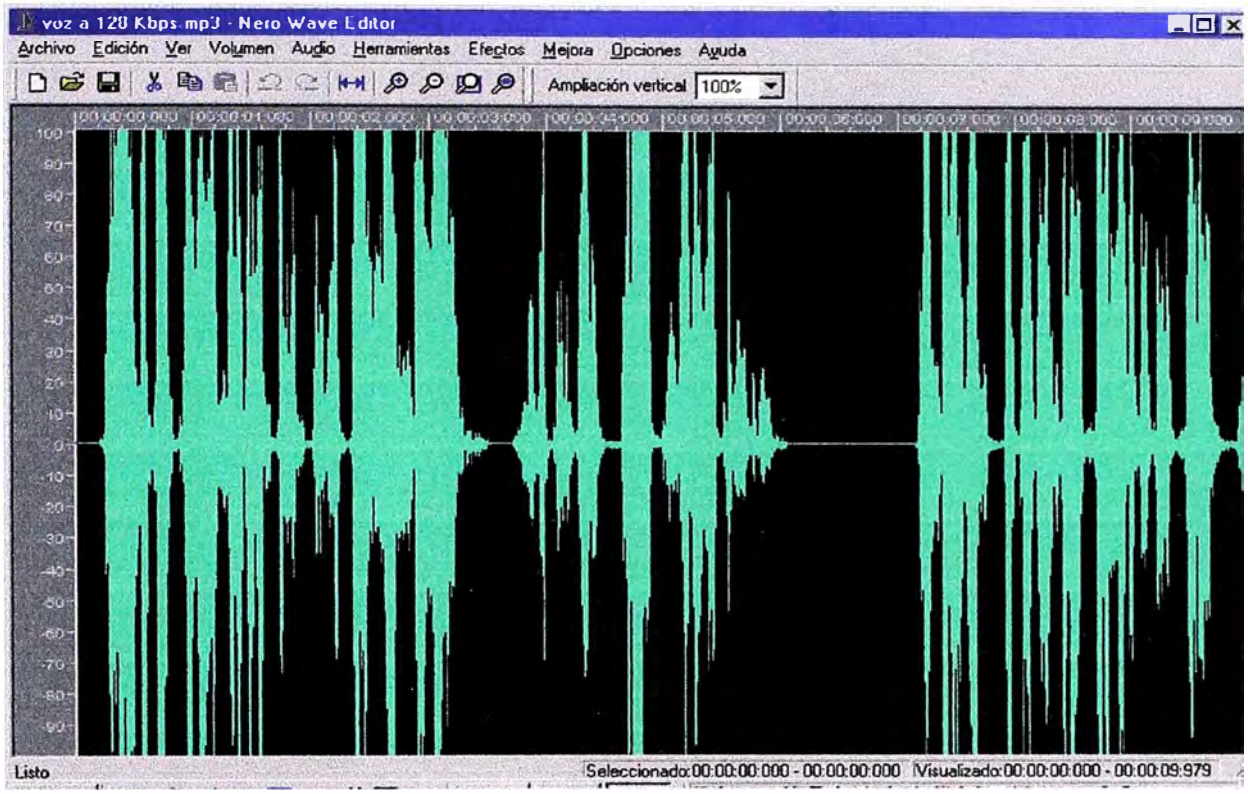


Figura PV3: **Voz a 112 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

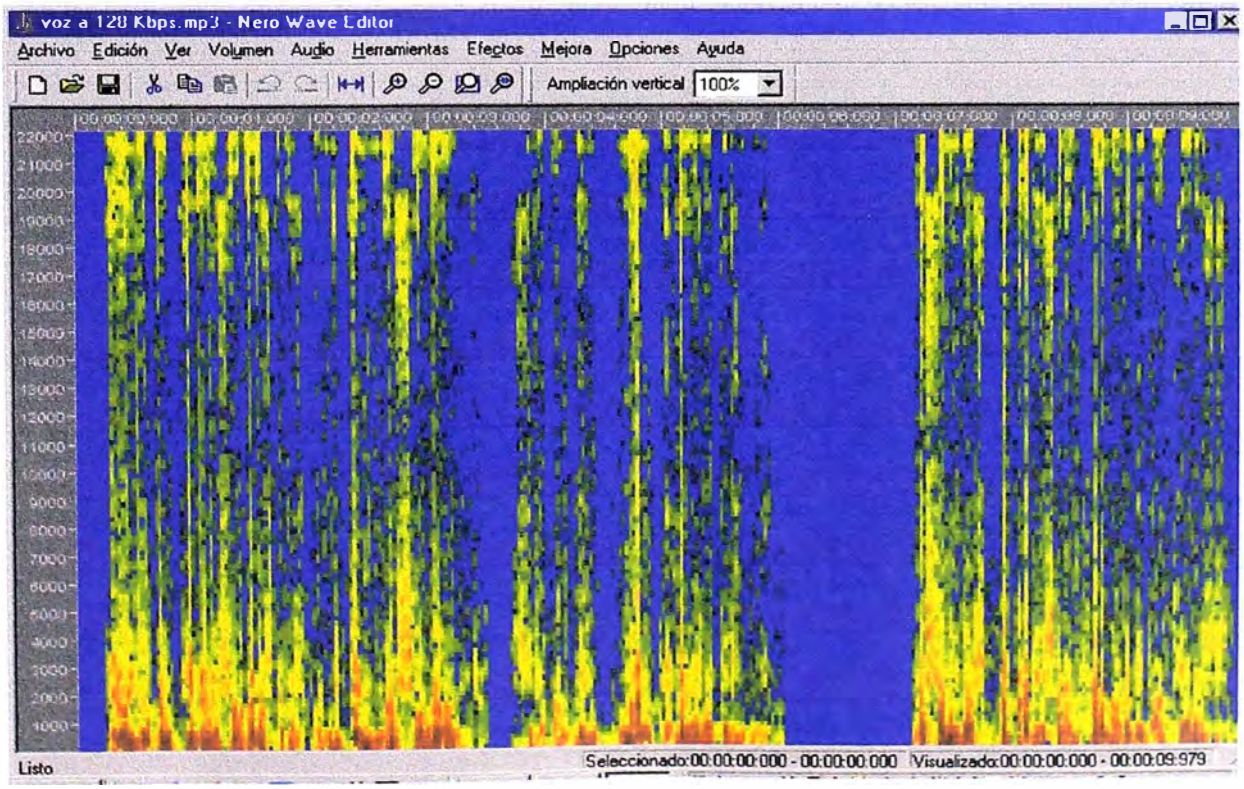
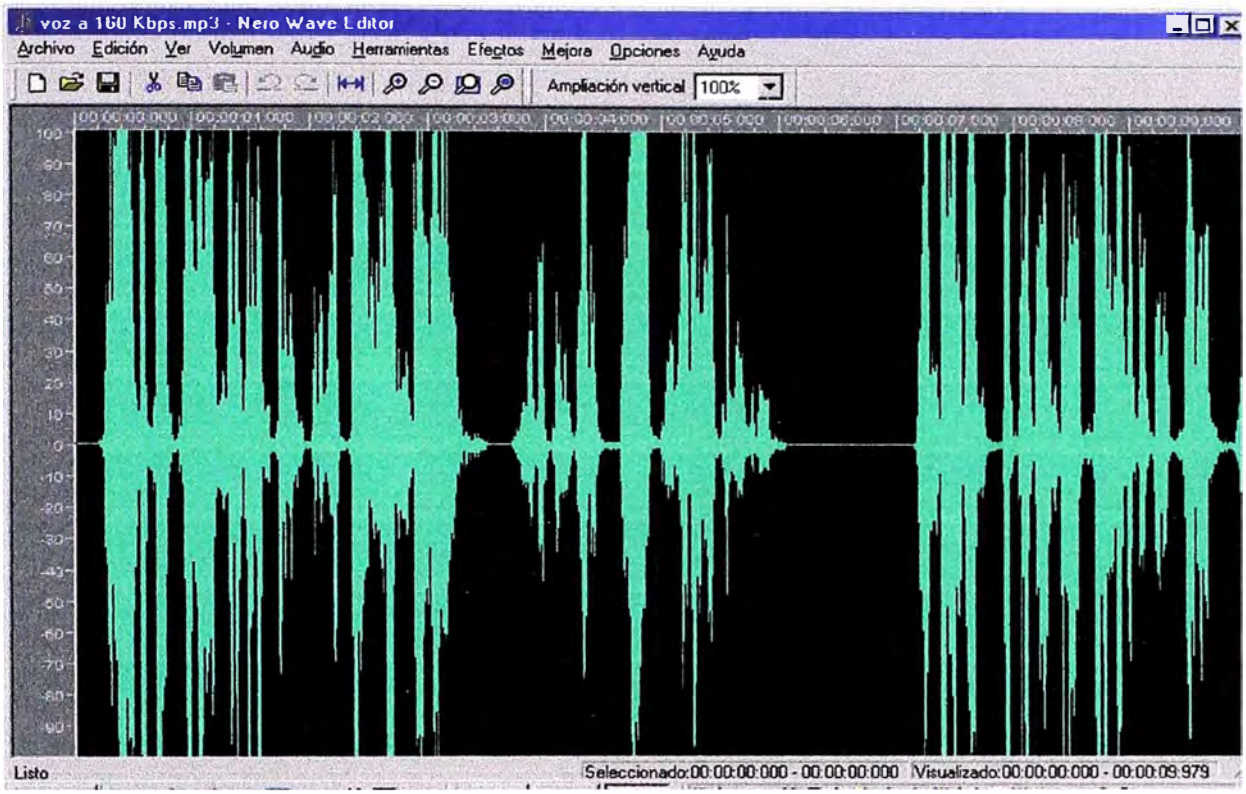


Figura PV4: **Voz a 128 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

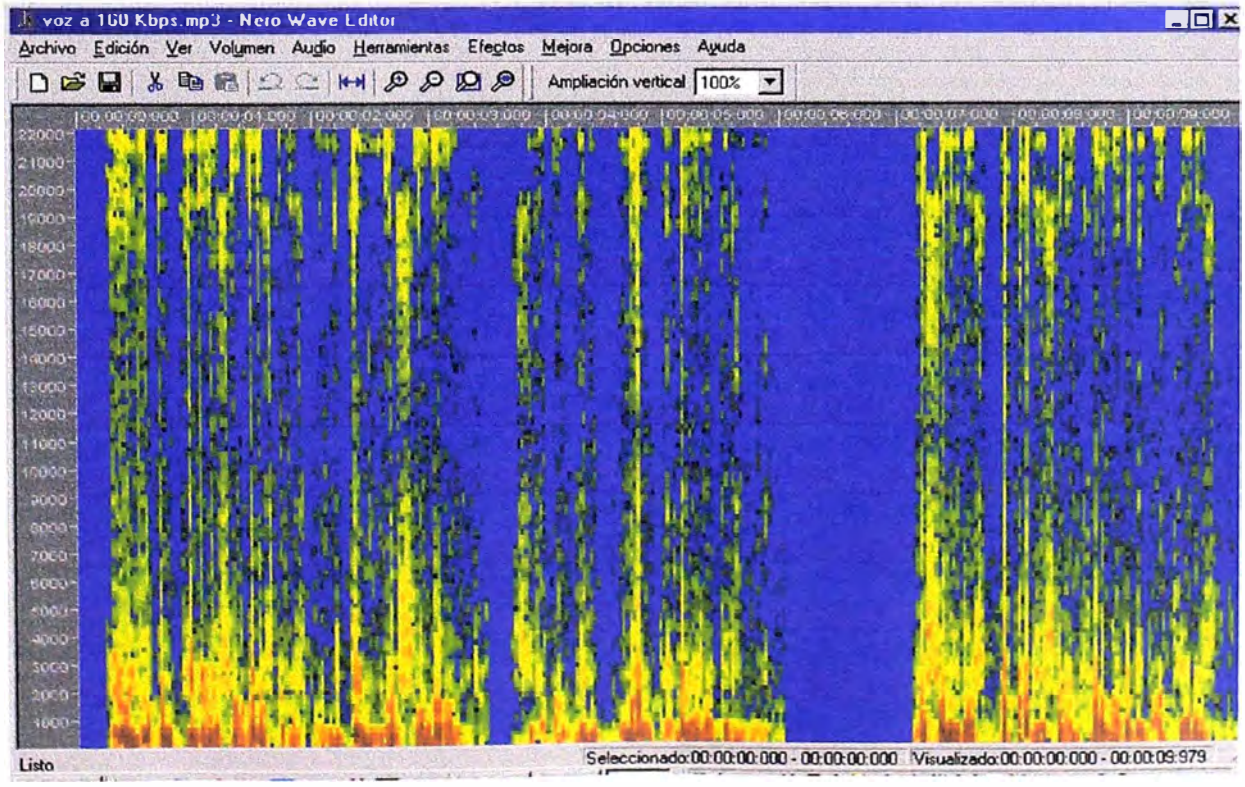
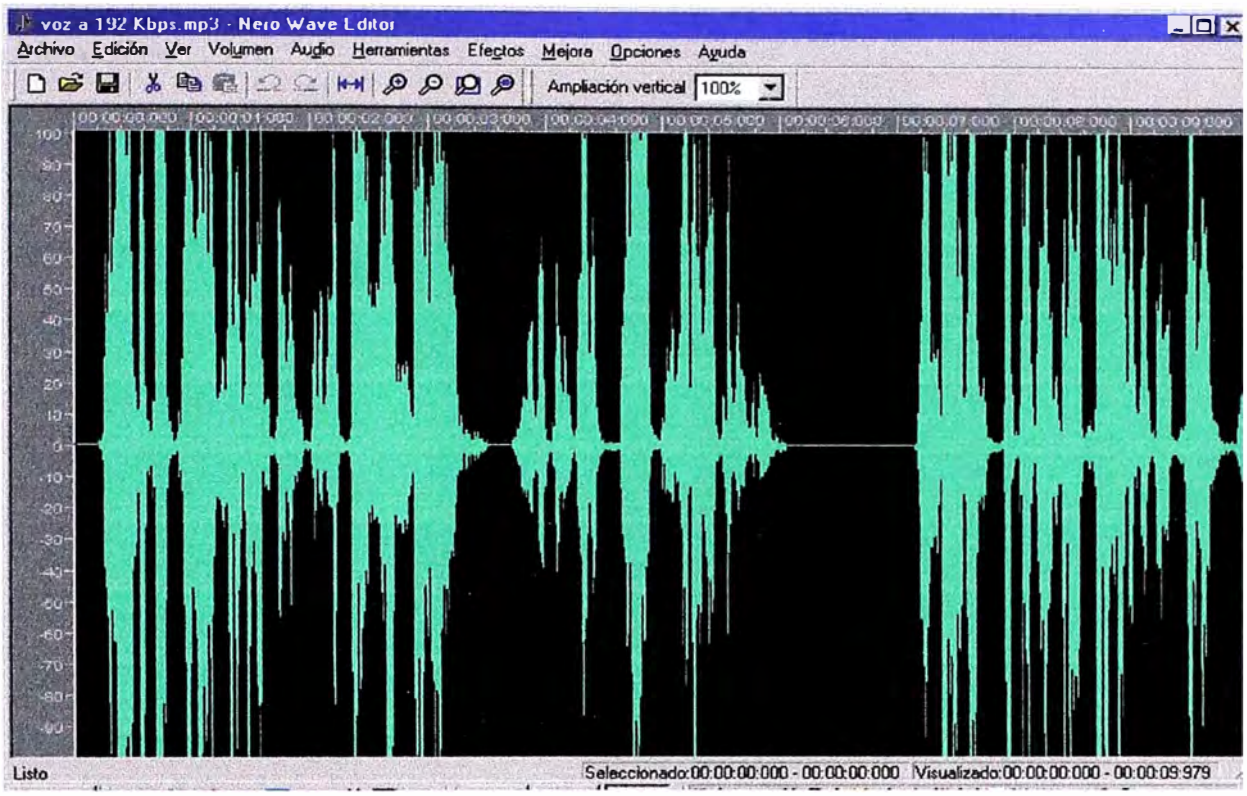


Figura PV5: **Voz a 160 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

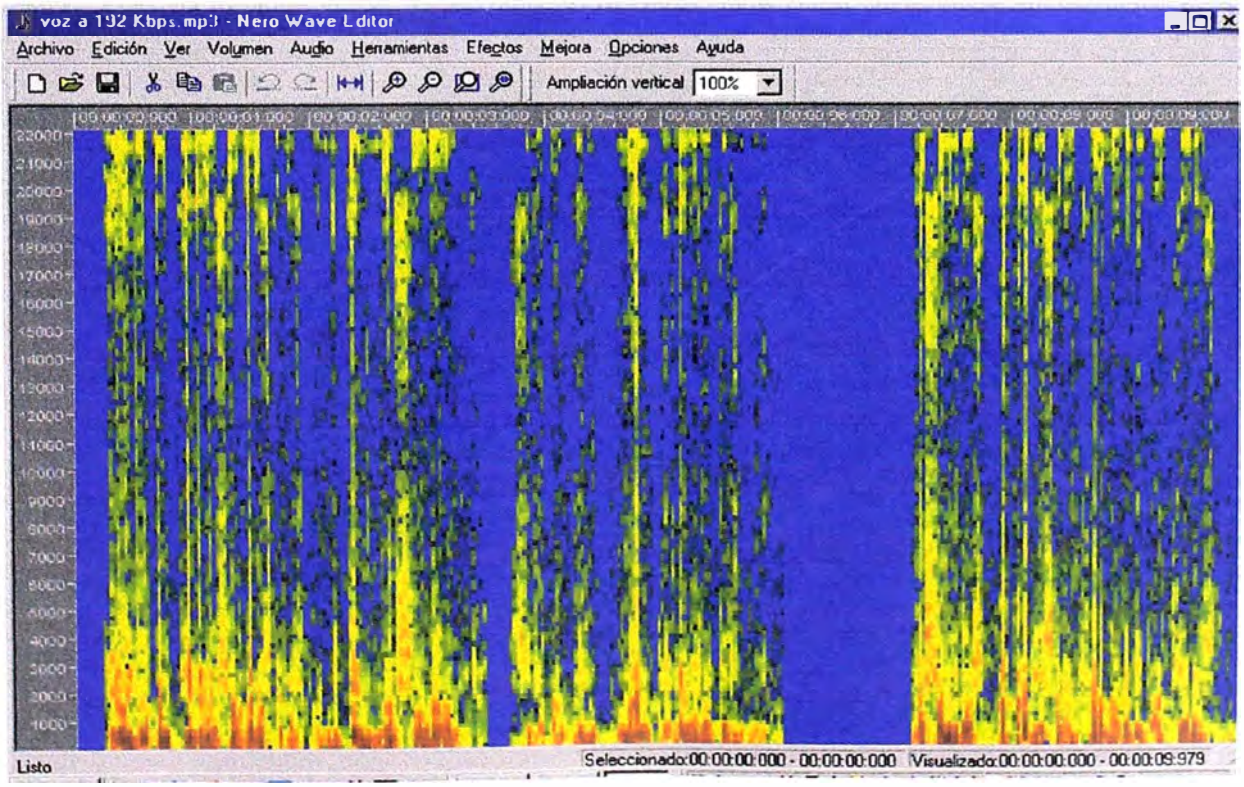
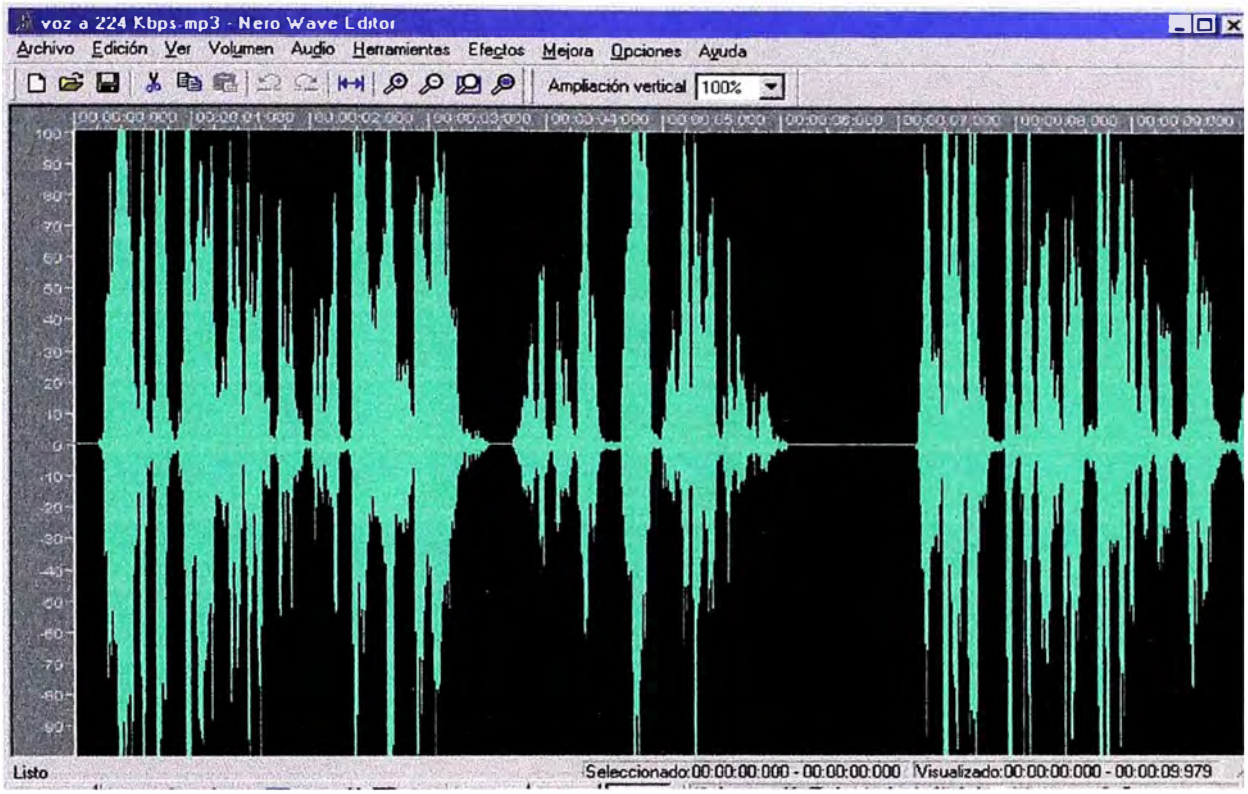


Figura PV6: **Voz a 192 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

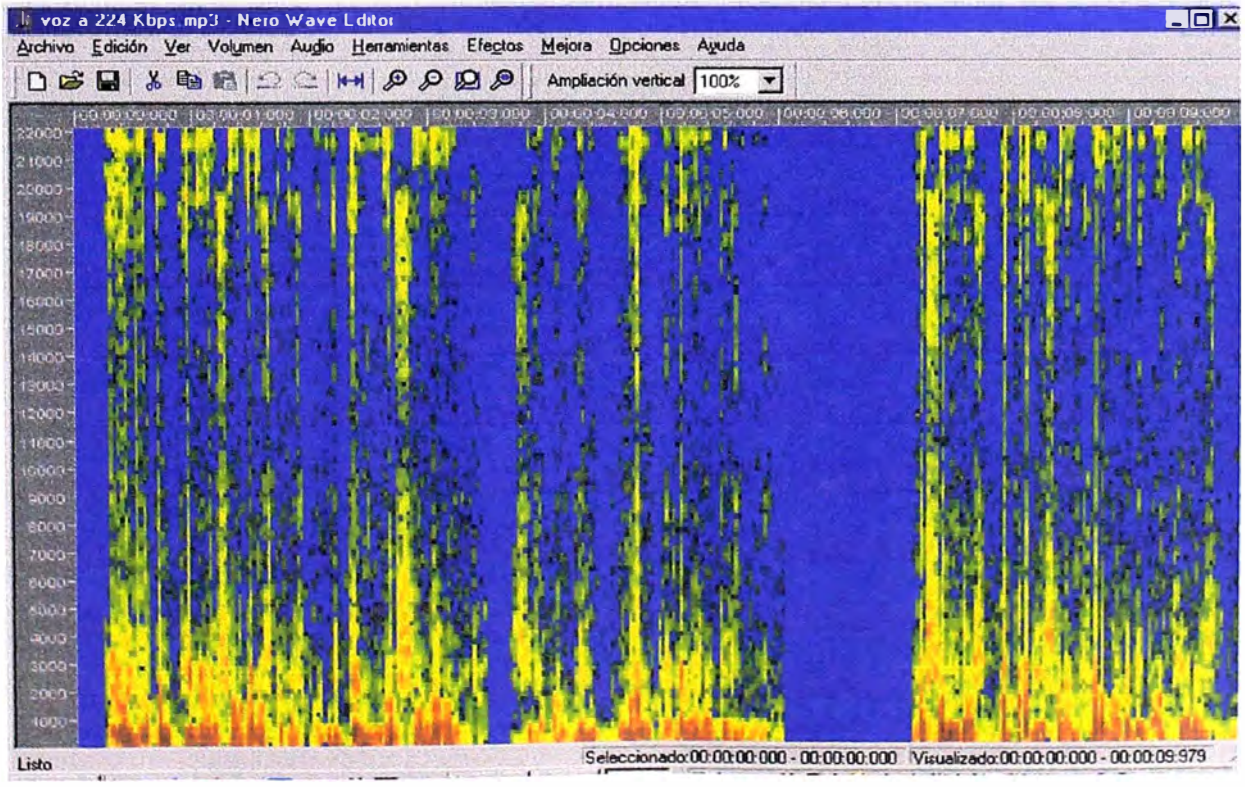
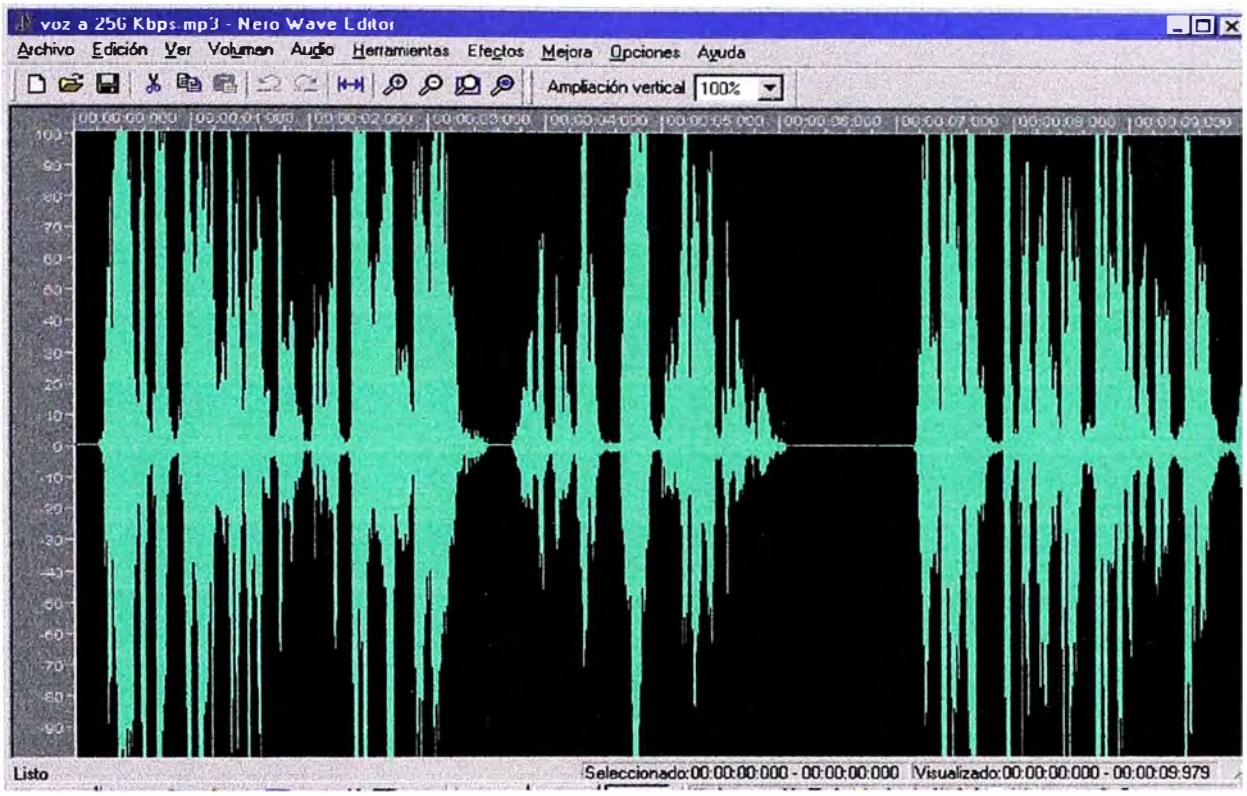


Figura PV7: **Voz a 224 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

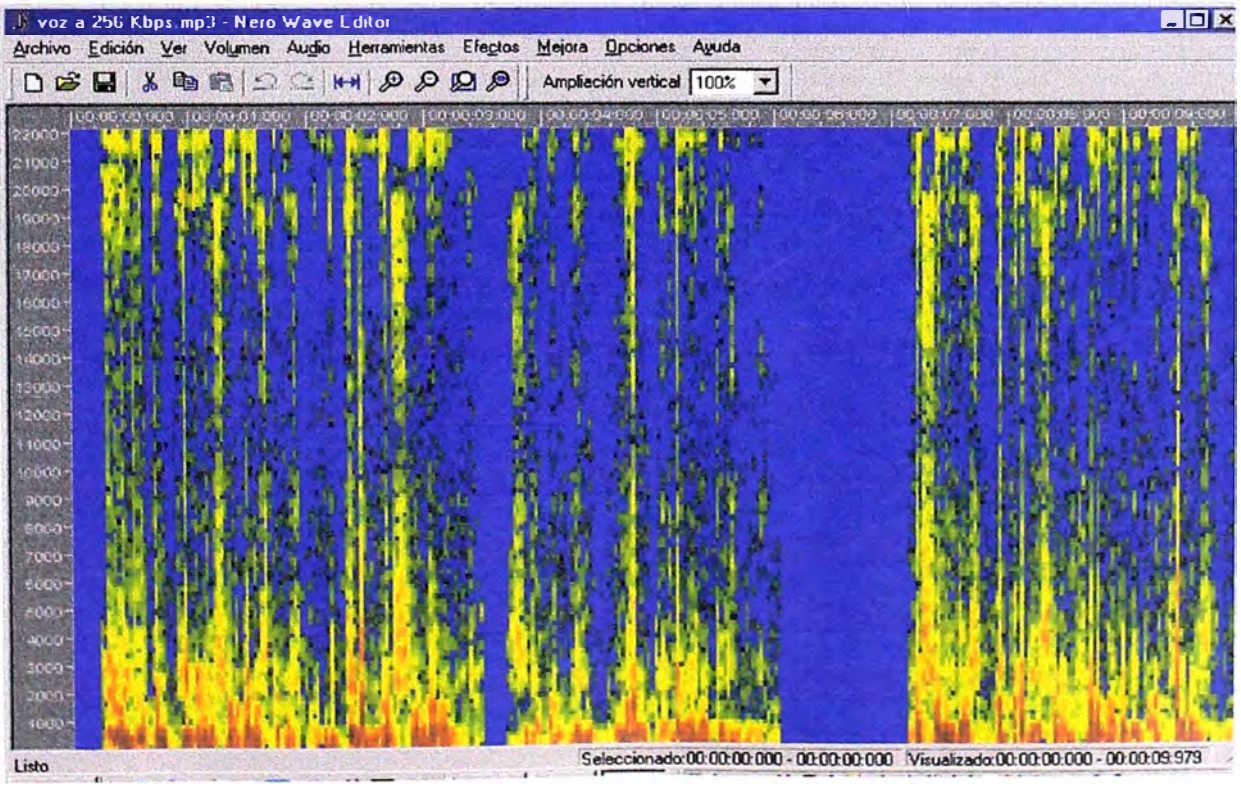
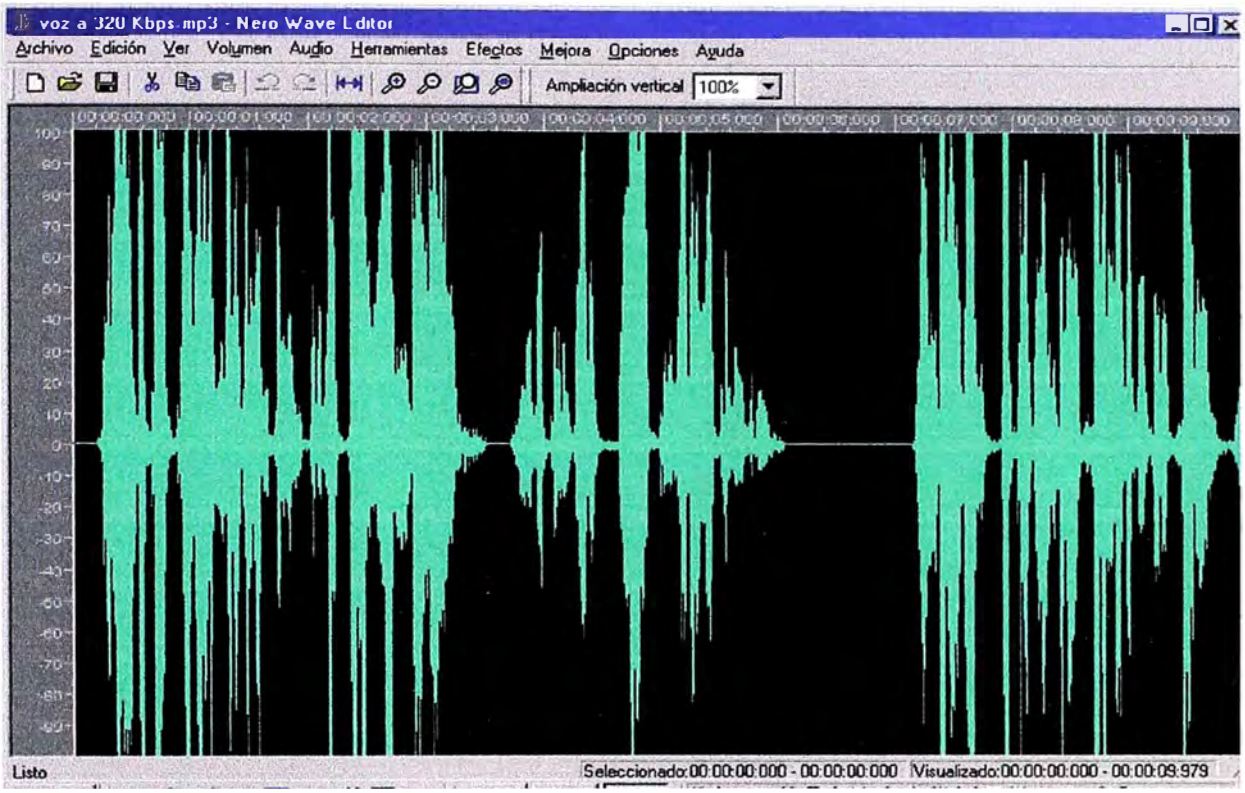


Figura PV8: **Voz a 256 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

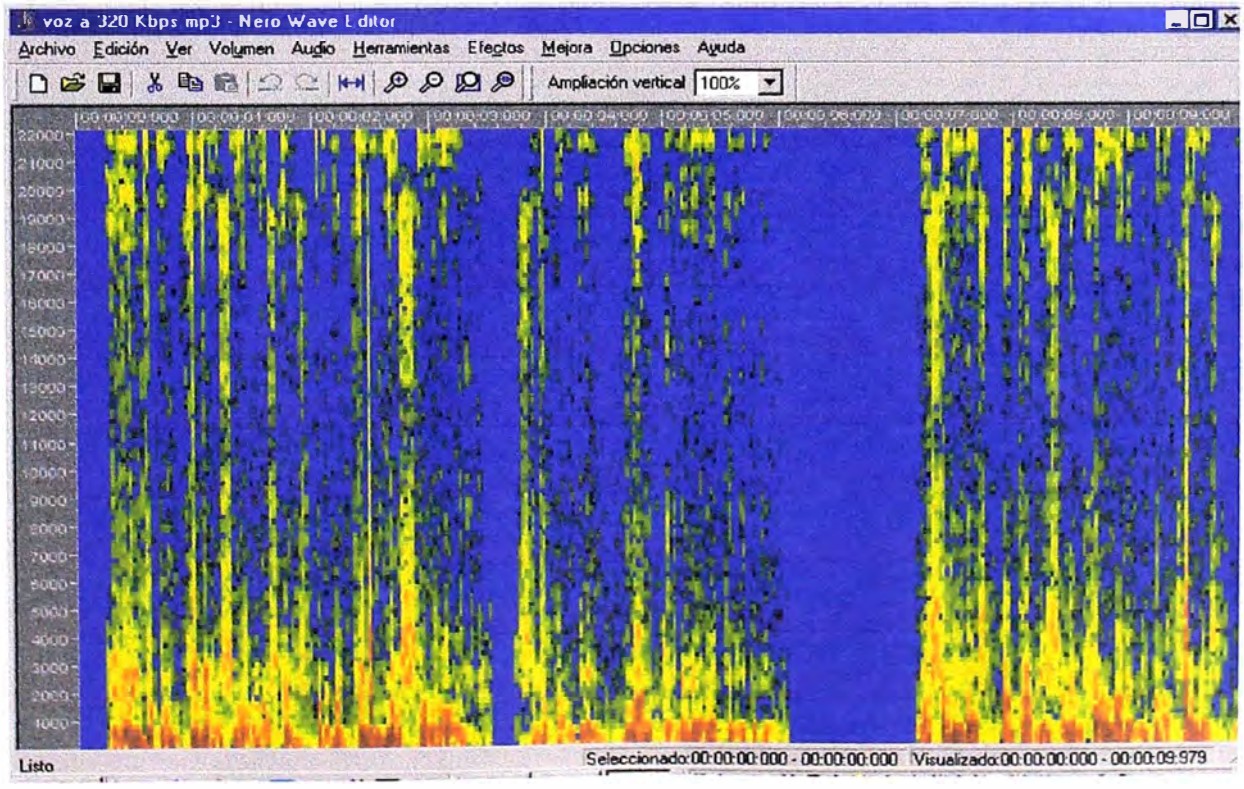
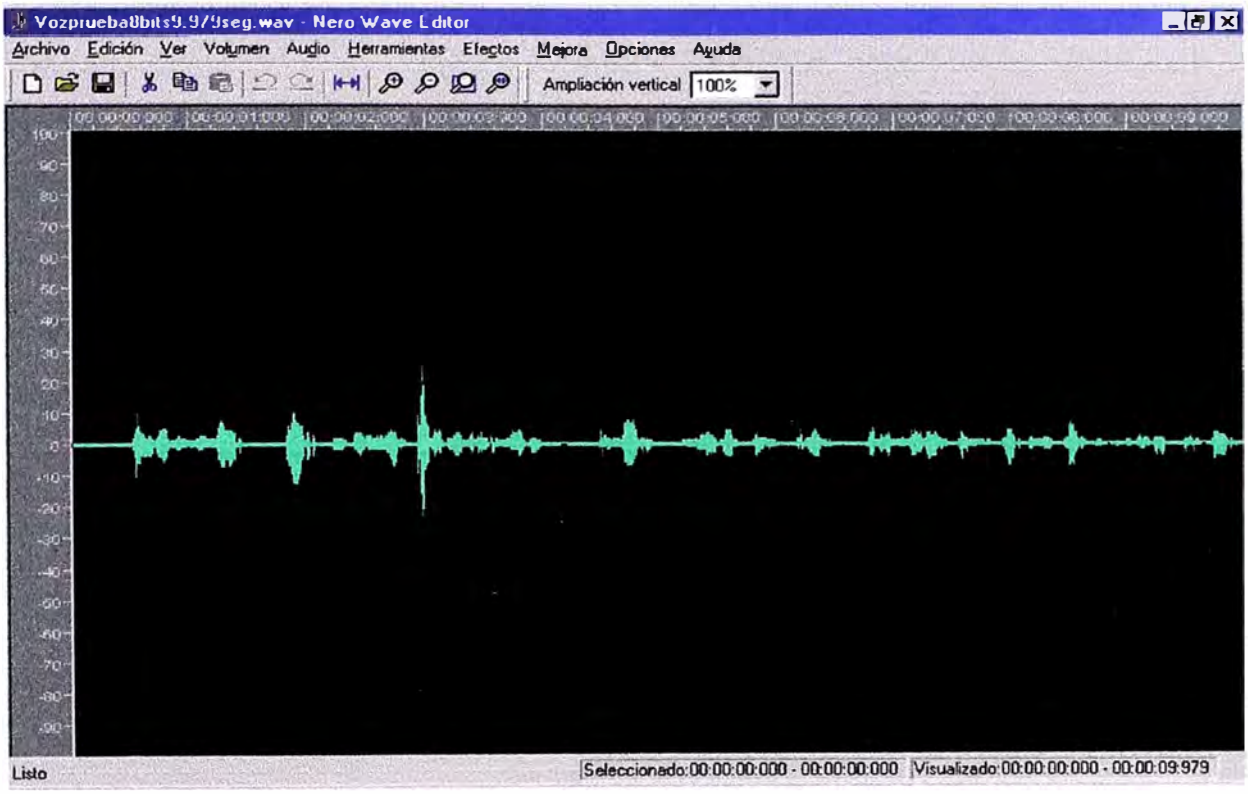


Figura PV9: **Voz a 320 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

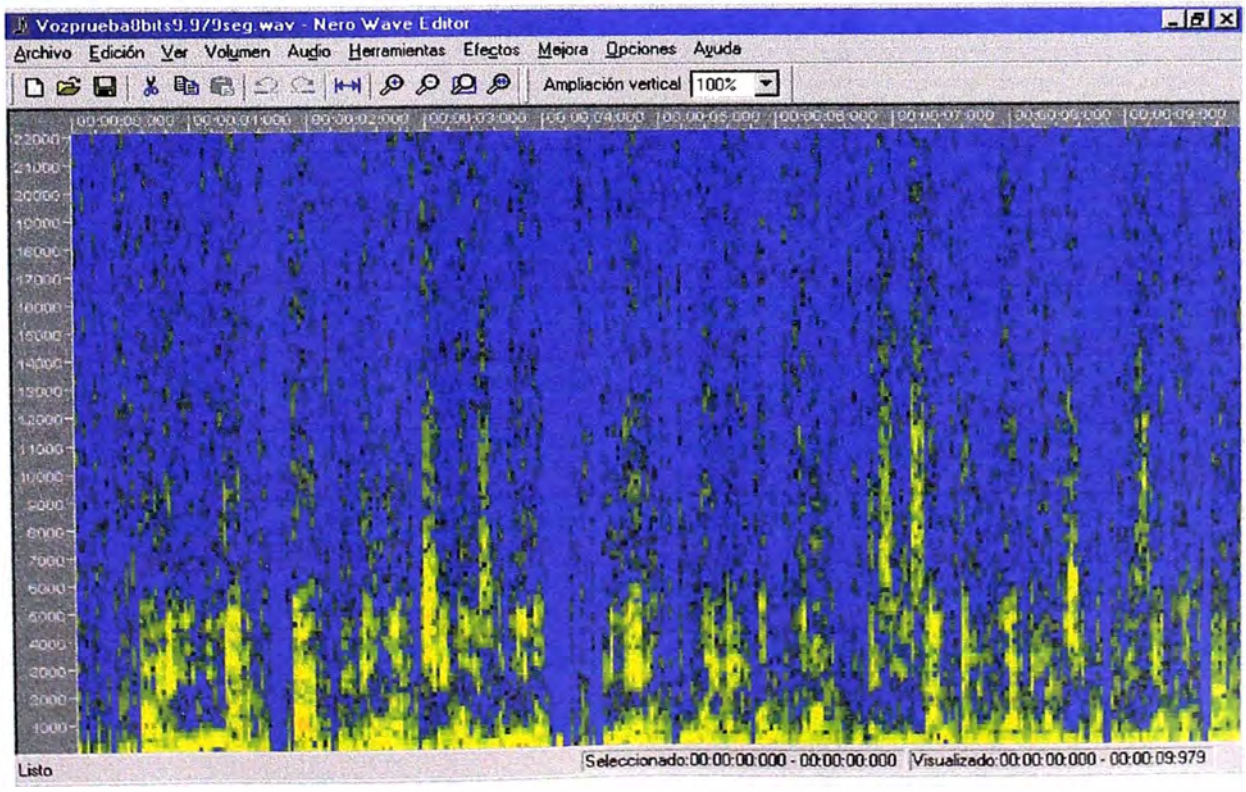


Figura PV10: **Vozprueba8bits9.979seg.wav**

Visualización de Onda

Visualización de Espectrograma

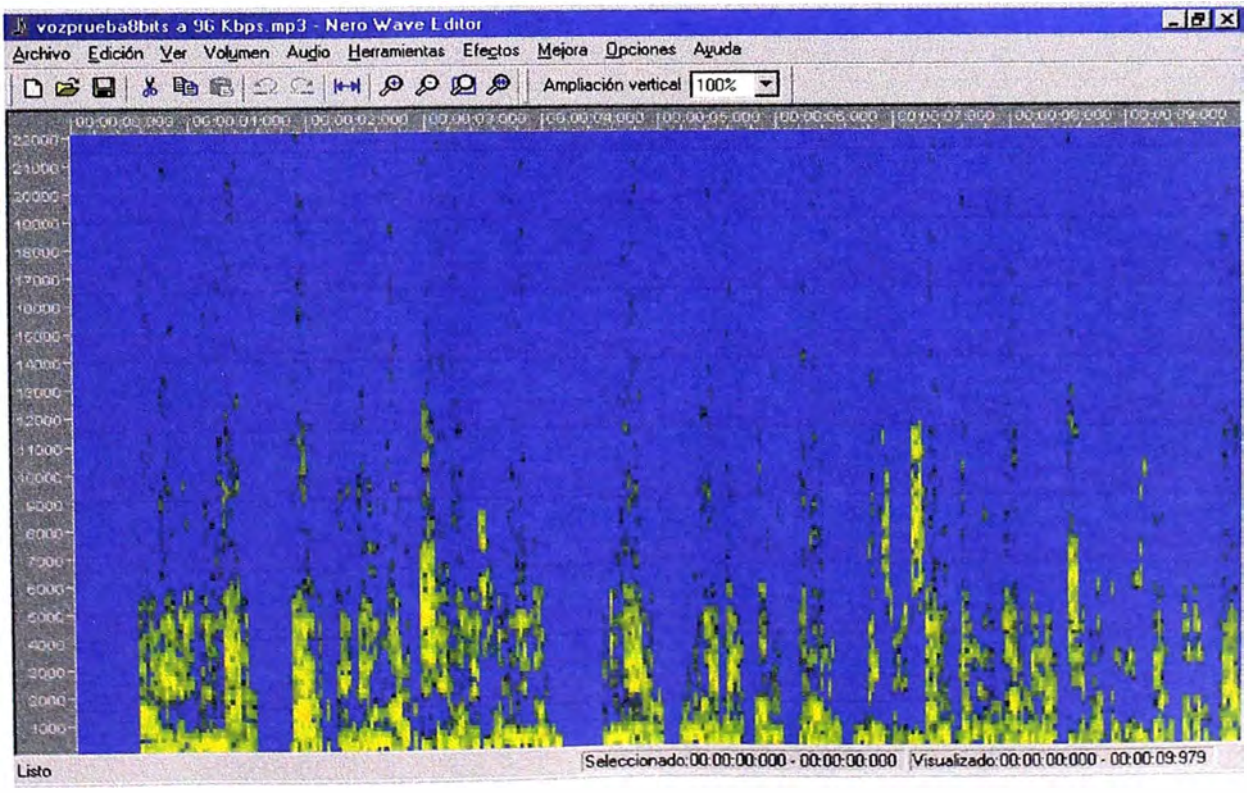
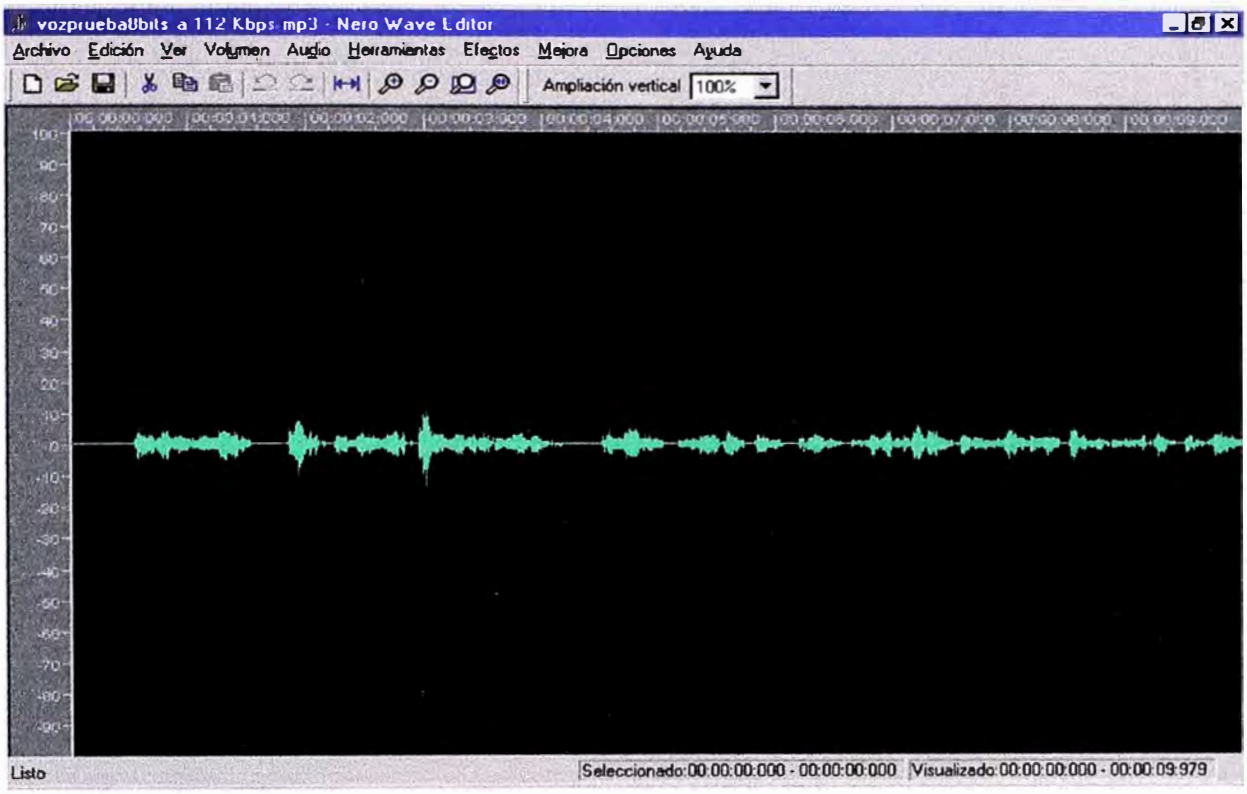


Figura PV11: **Vozprueba8bits a 96 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

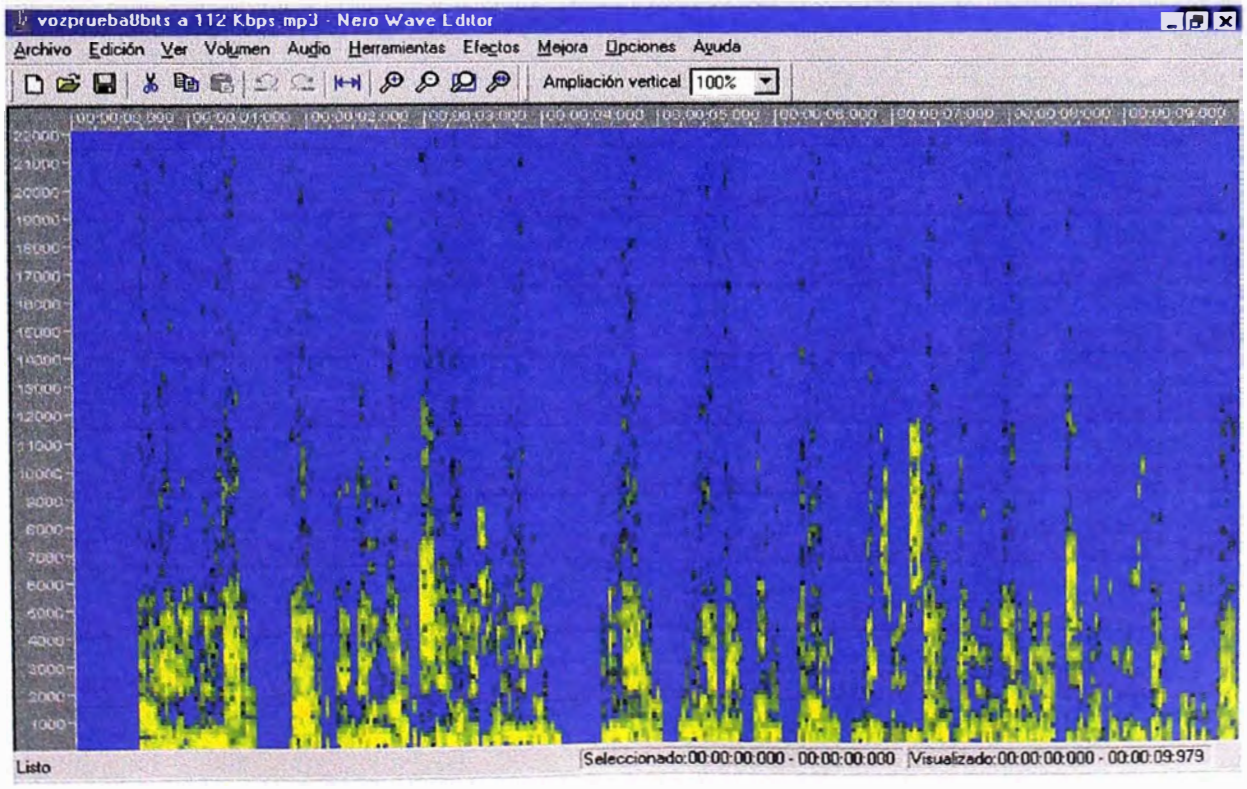
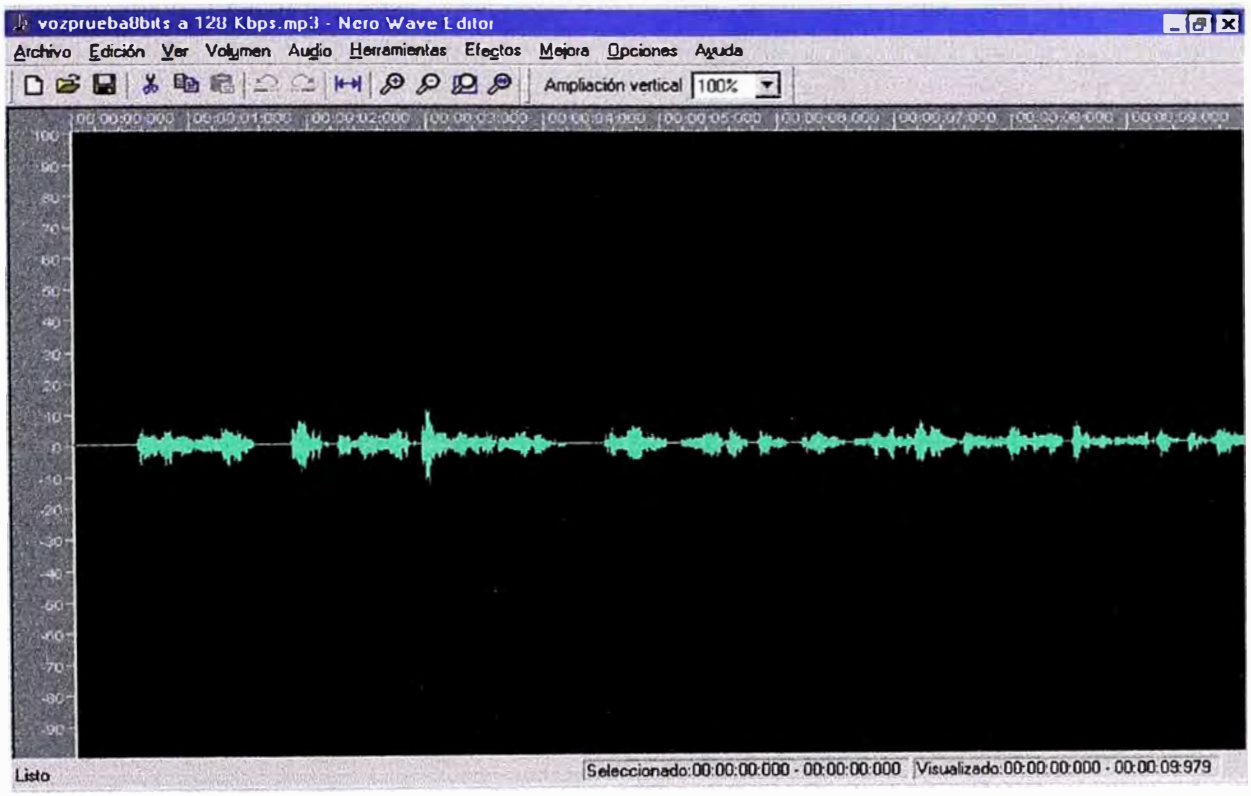


Figura PV12: **Vozprueba8bits a 112 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

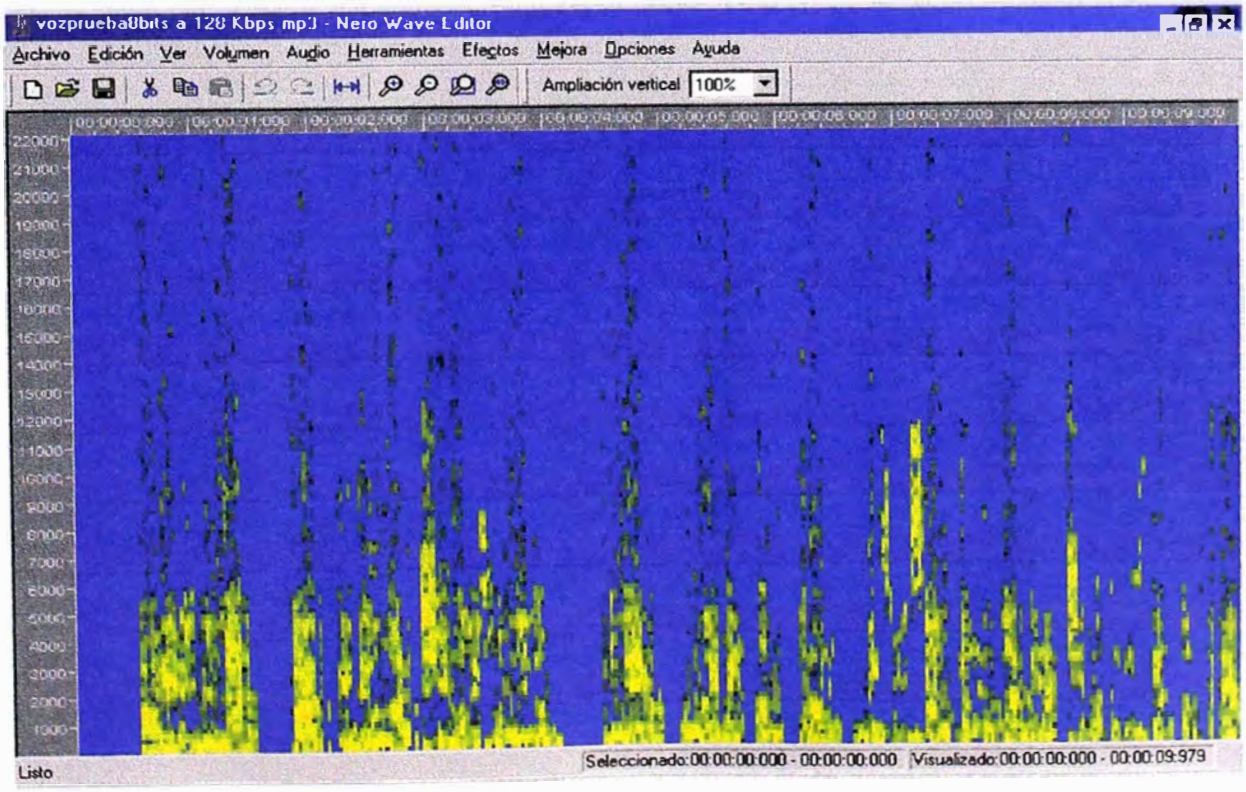


Figura PV13: **Vozprueba8bits a 128 Kbps.mp3**

Visualización de Onda

Visualización de Espectrograma

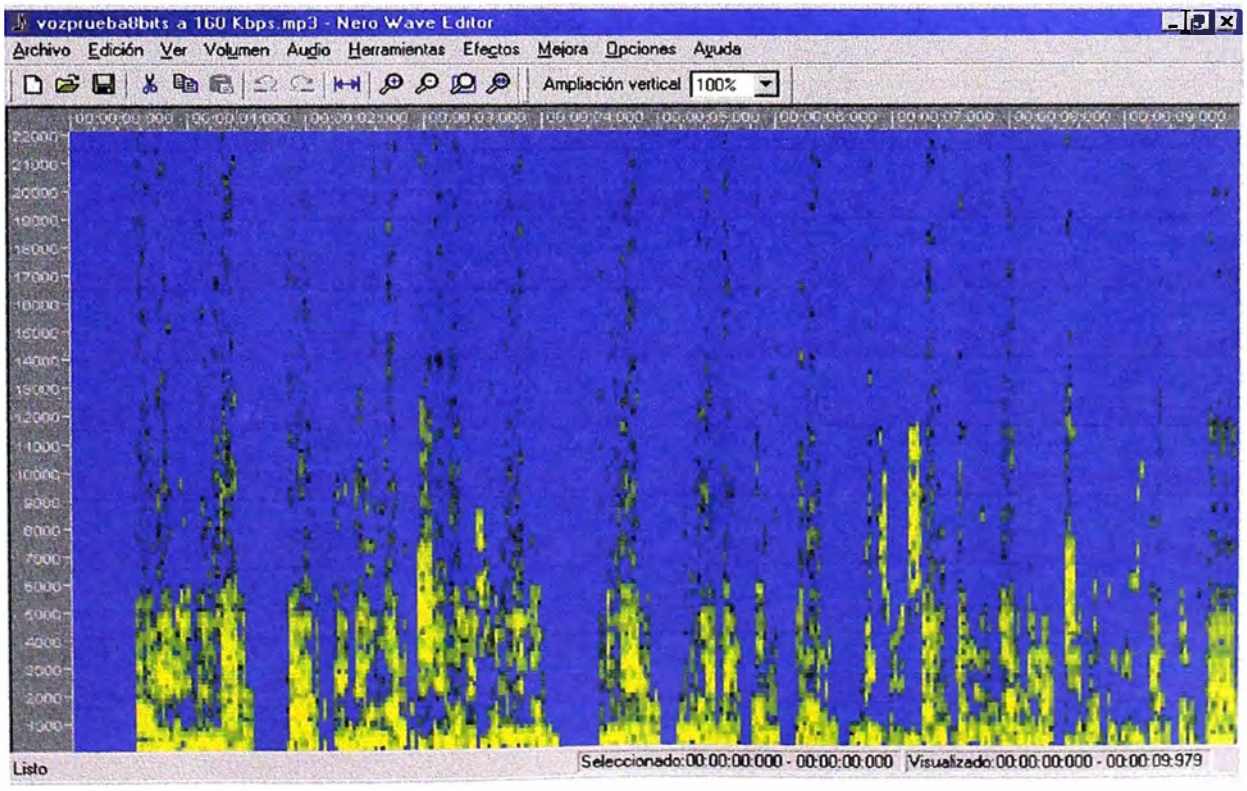
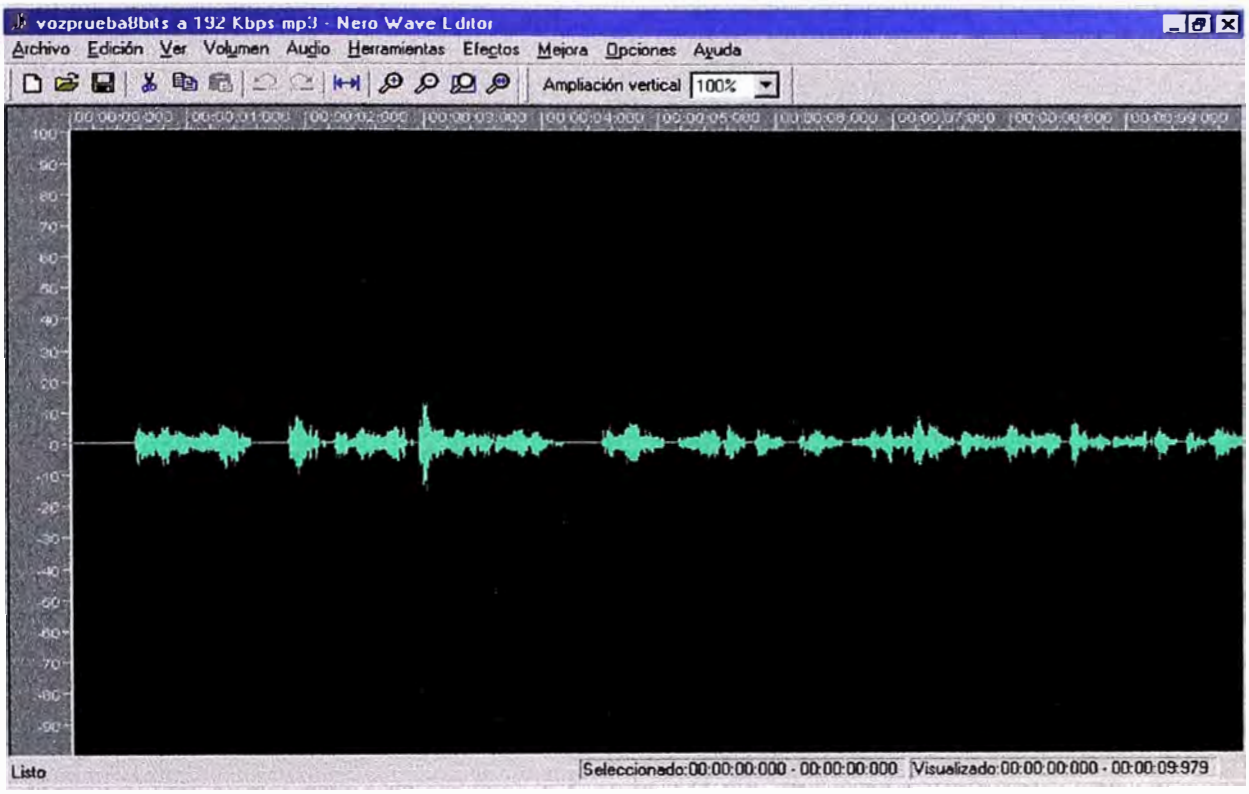


Figura PV14: **Vozprueba8bits a 160 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

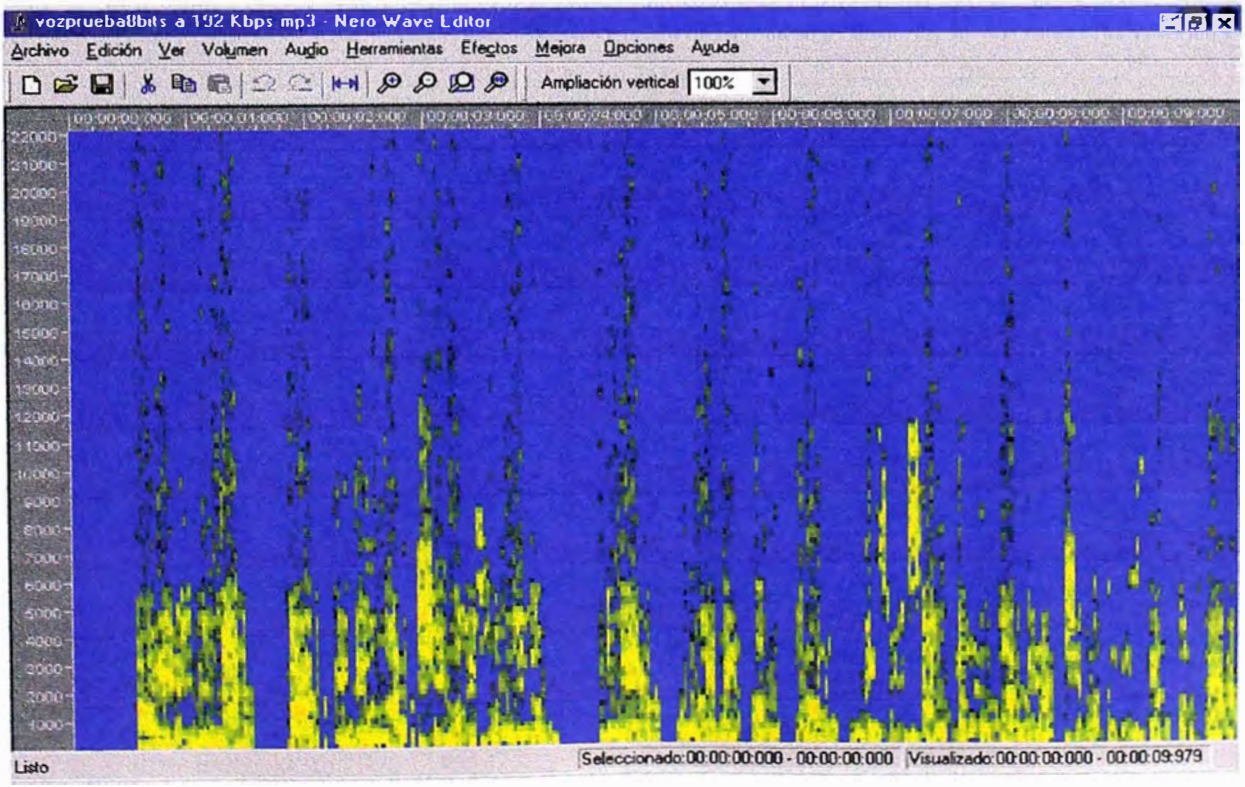
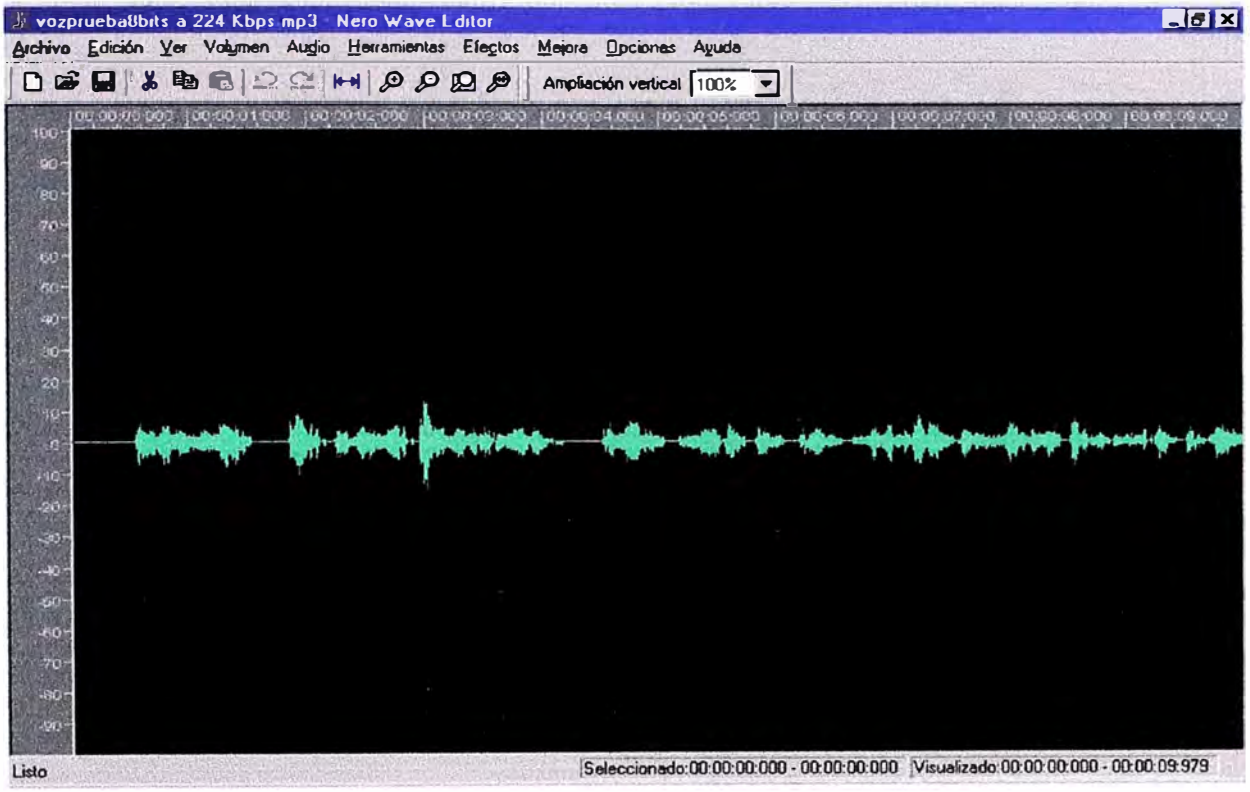


Figura PV15: **Vozprueba8bits a 192 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

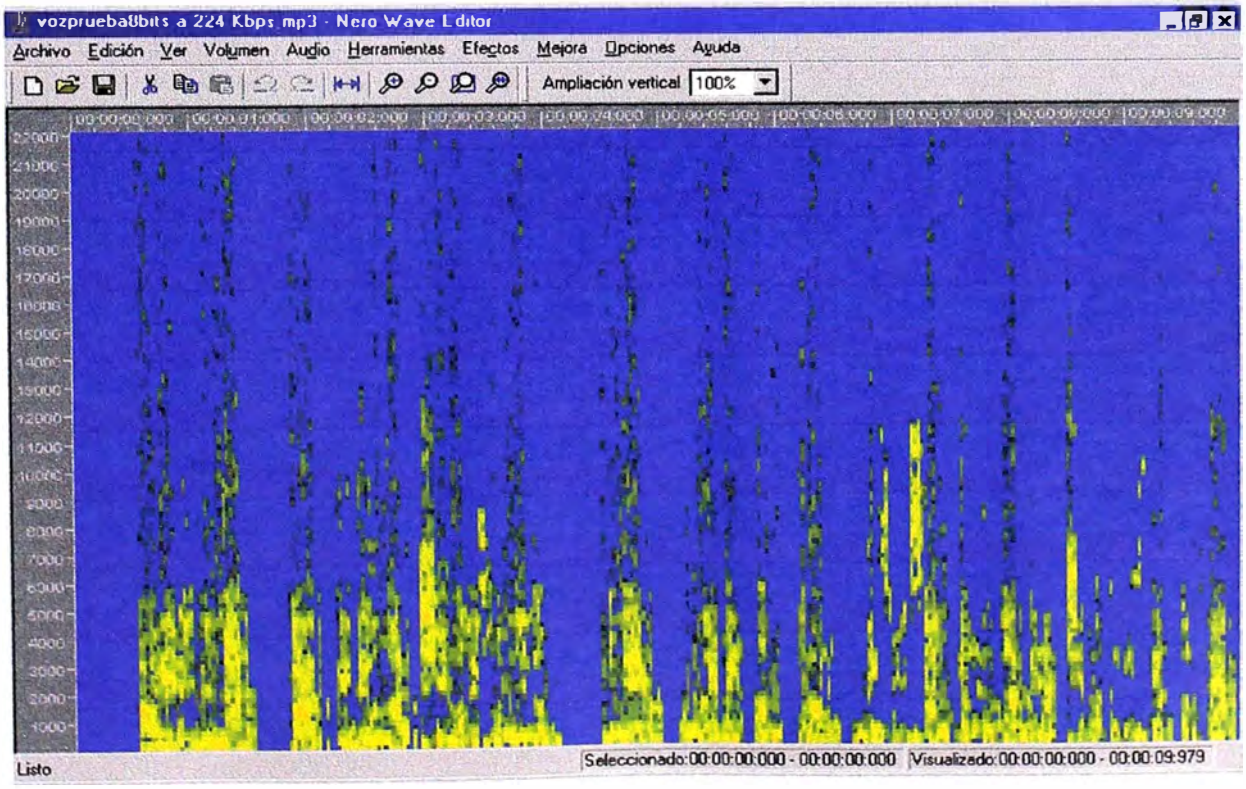
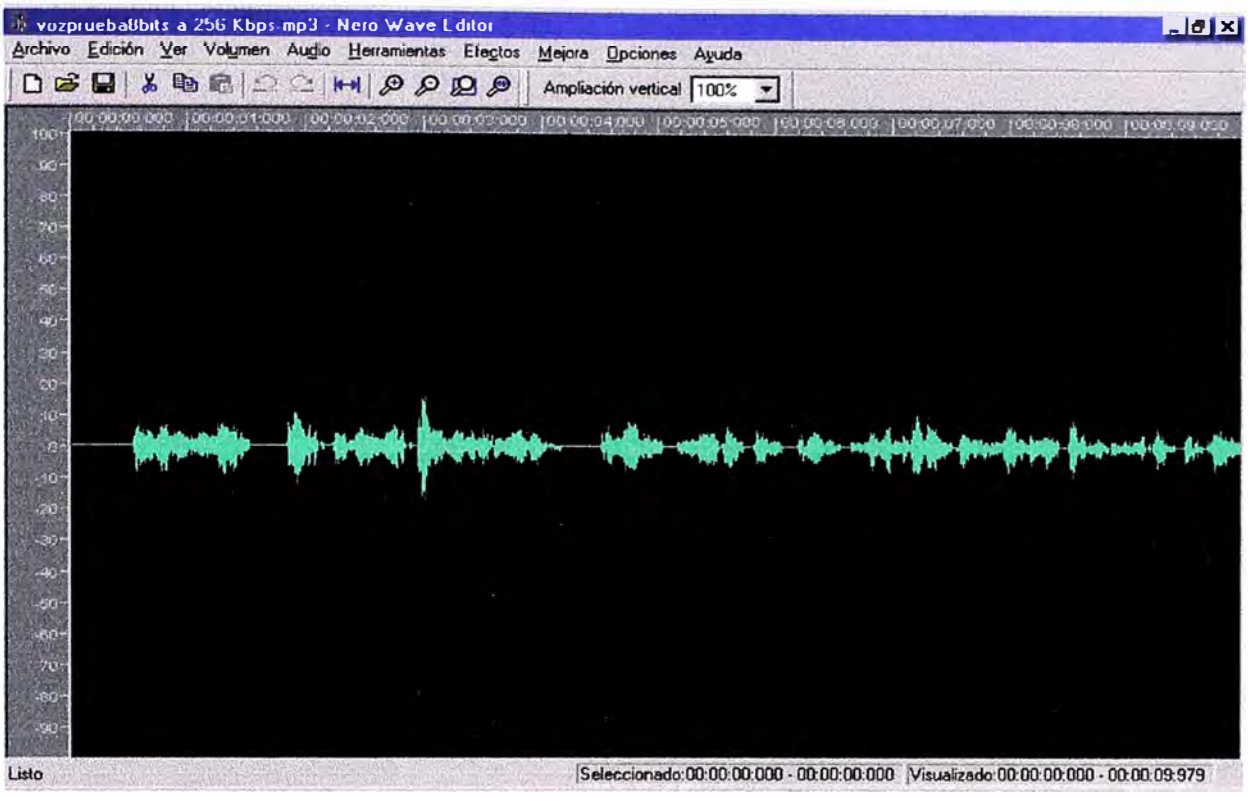


Figura PV16: **Vozprueba8bits a 224 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

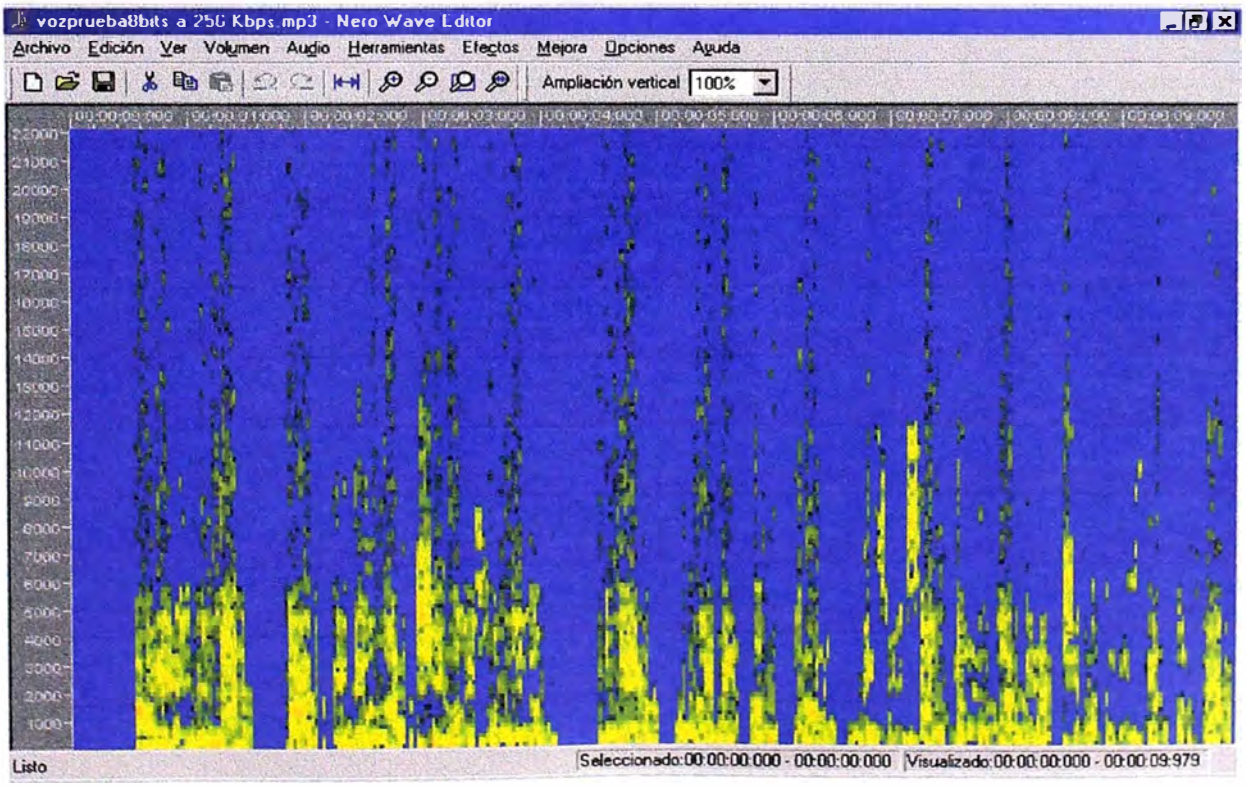
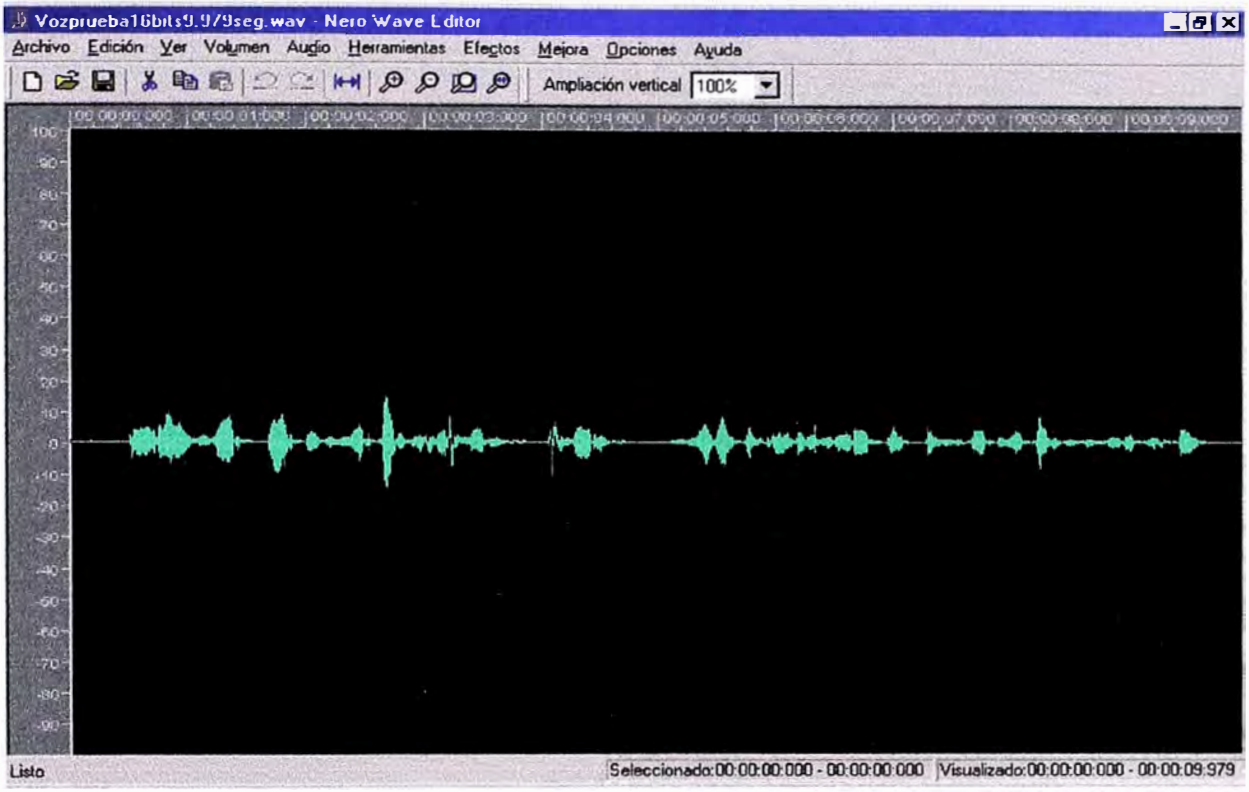


Figura PV17: **Vozprueba8bits a 256 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

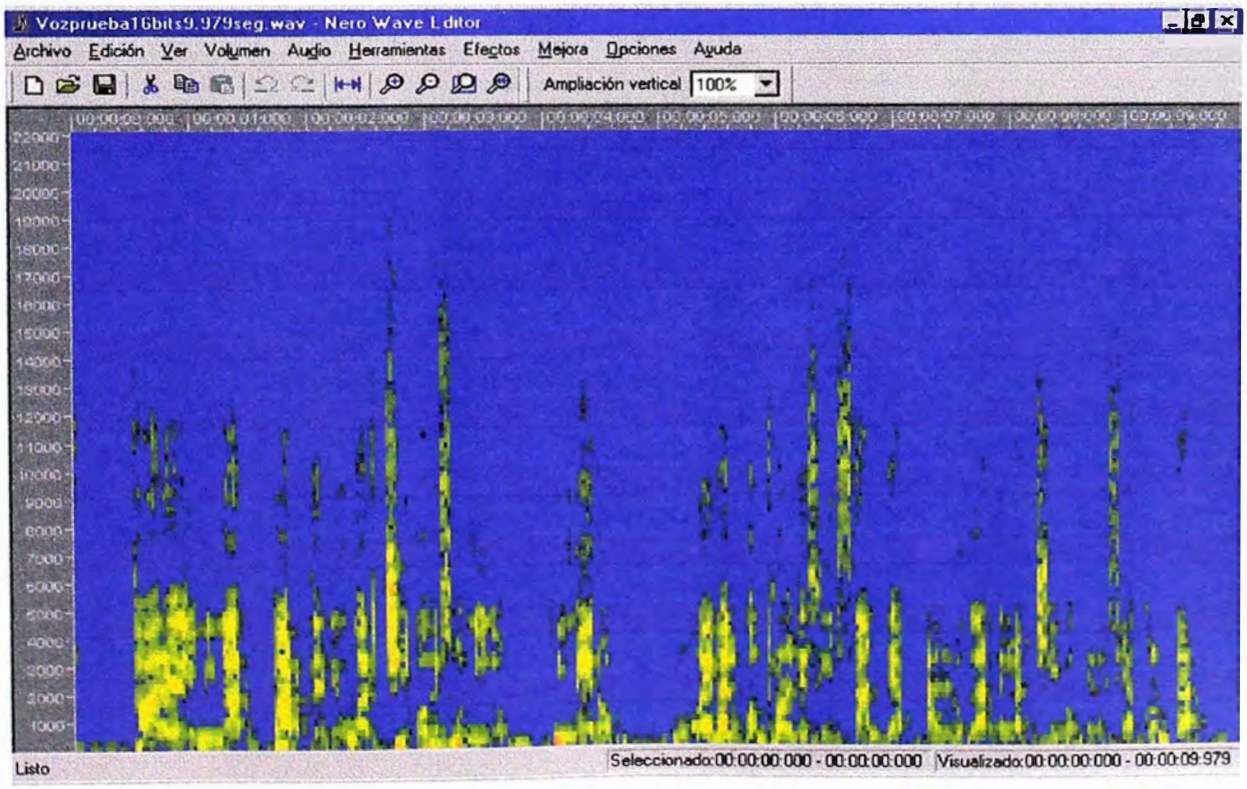
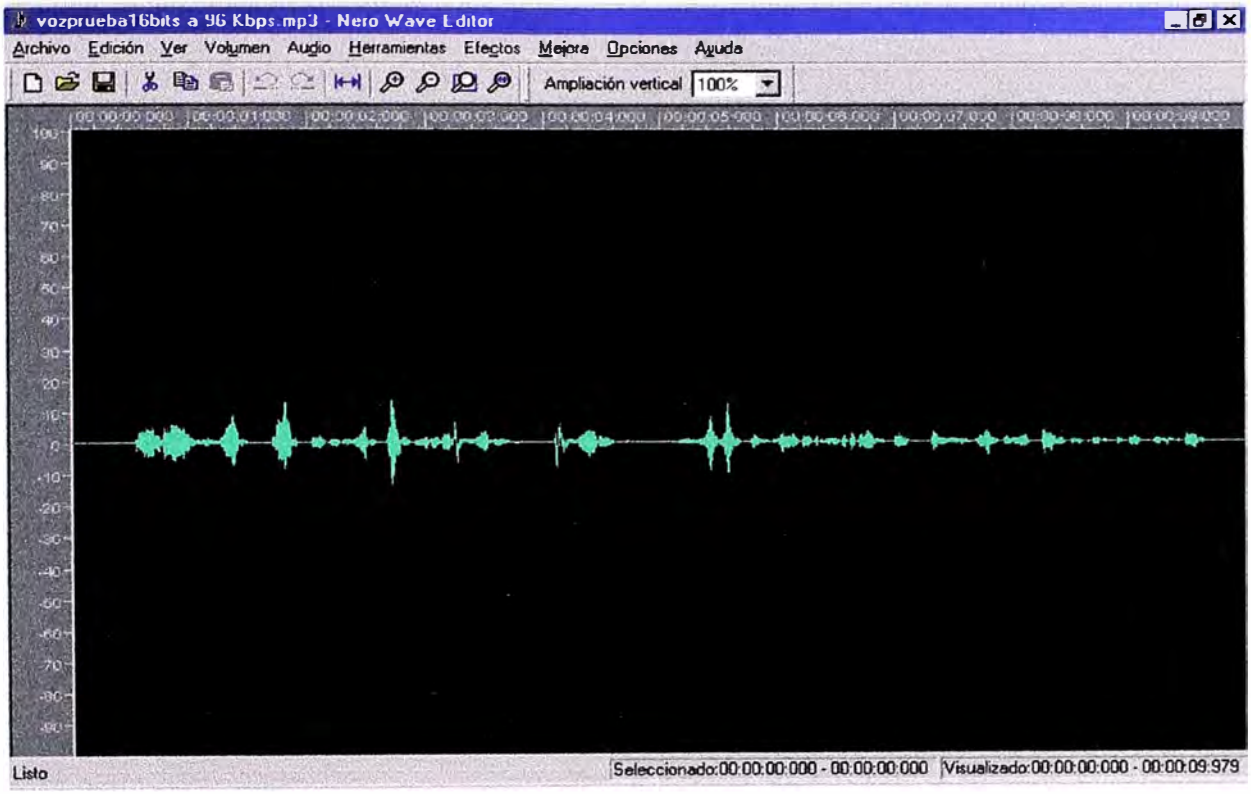


Figura PV18: **Vozprueba16bits9.979seg.wav**

Visualización de Onda



Visualización de Espectrograma

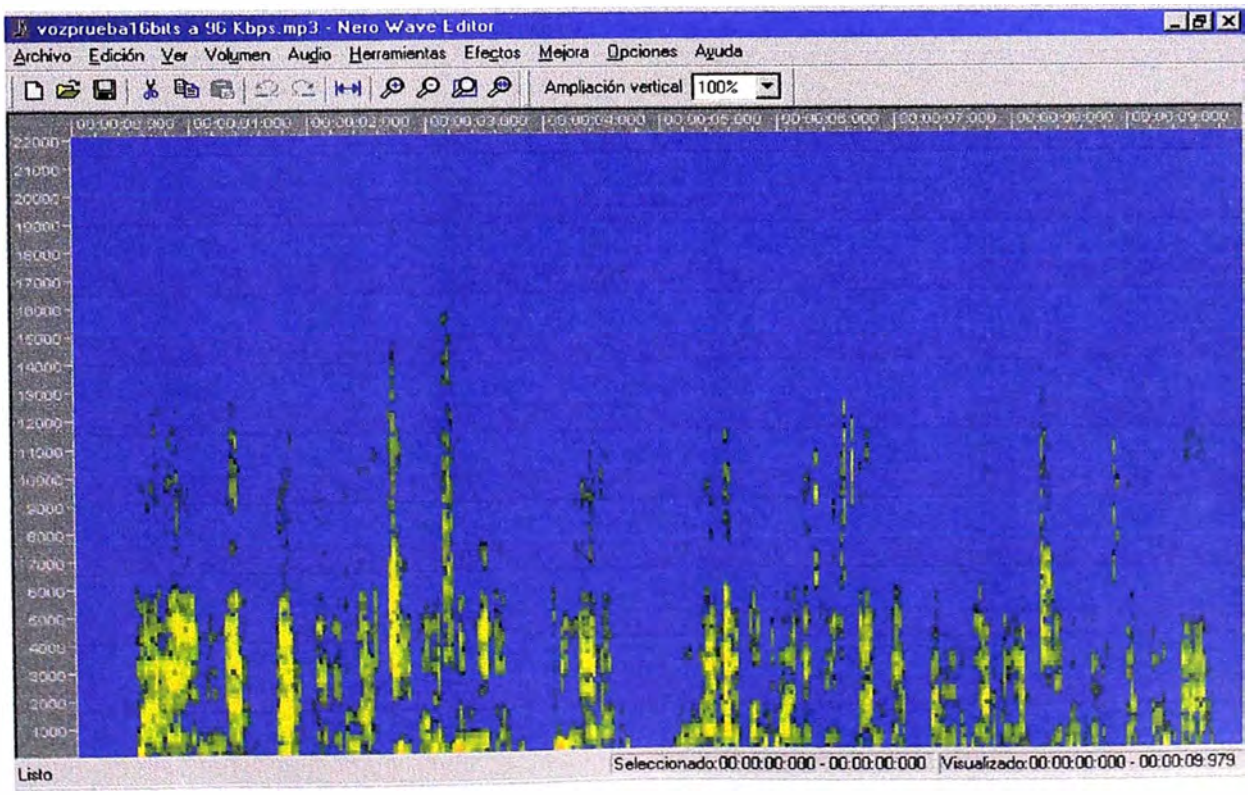
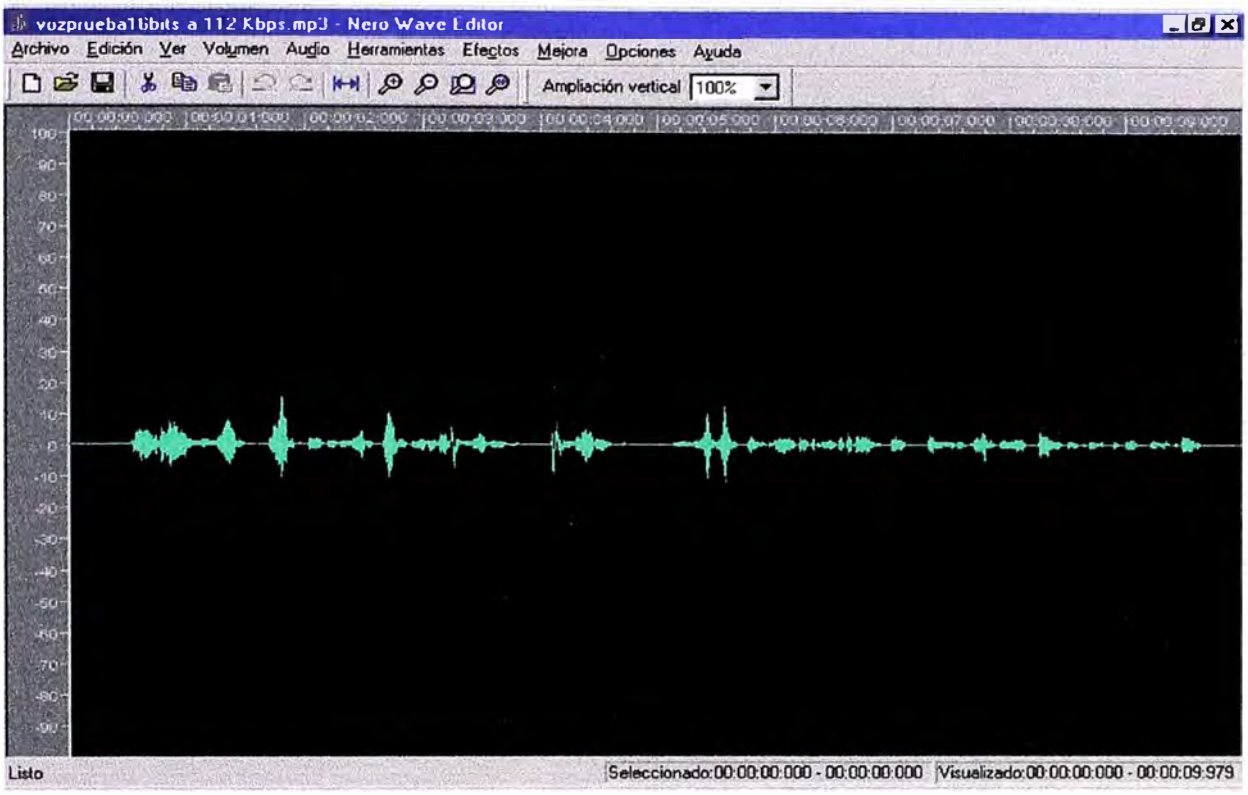


Figura PV19: **Vozprueba16bits a 96 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

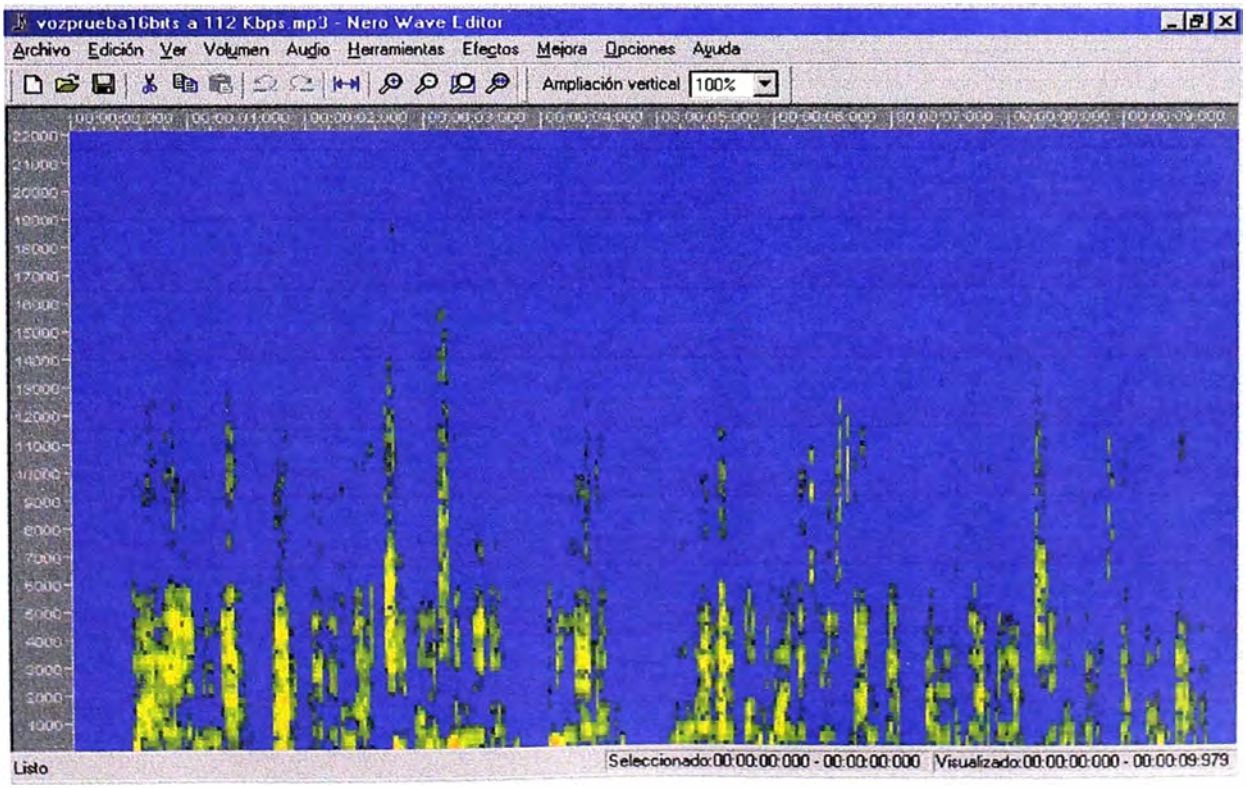
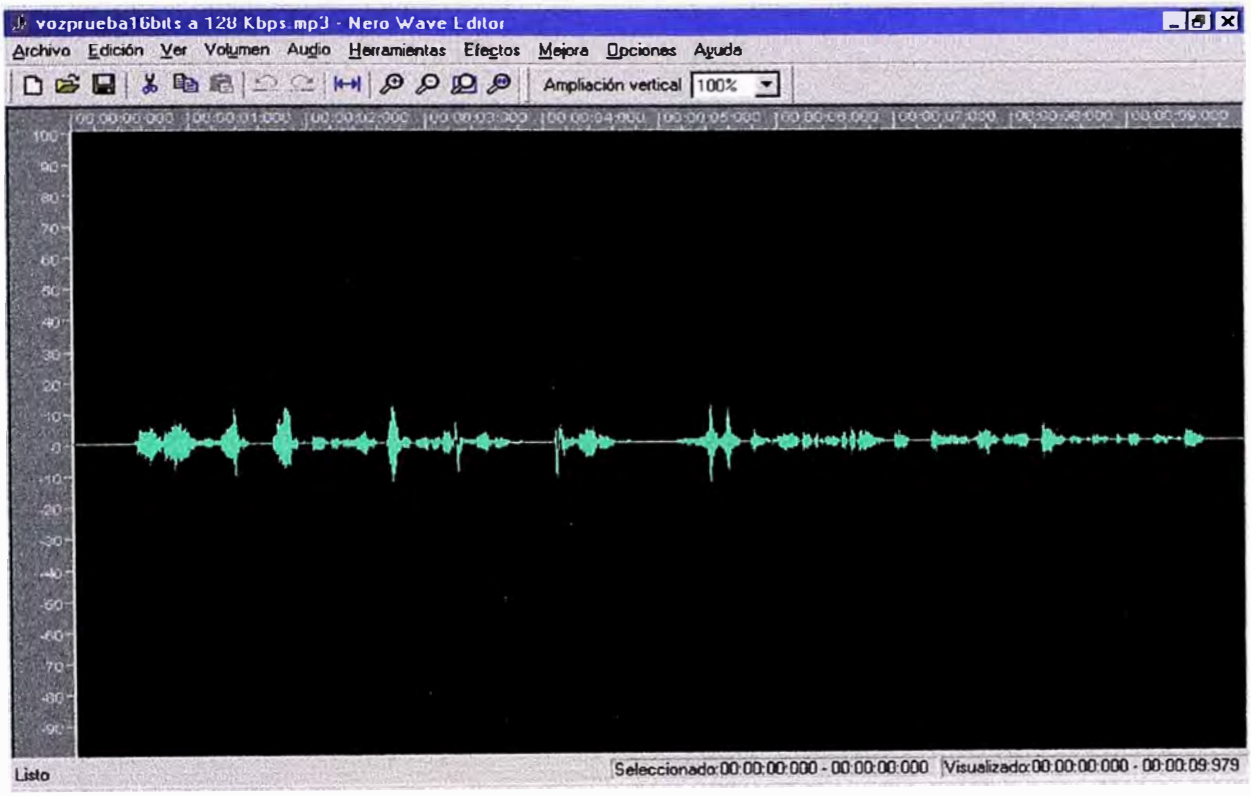


Figura PV20: **Vozprueba16bits a 112 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

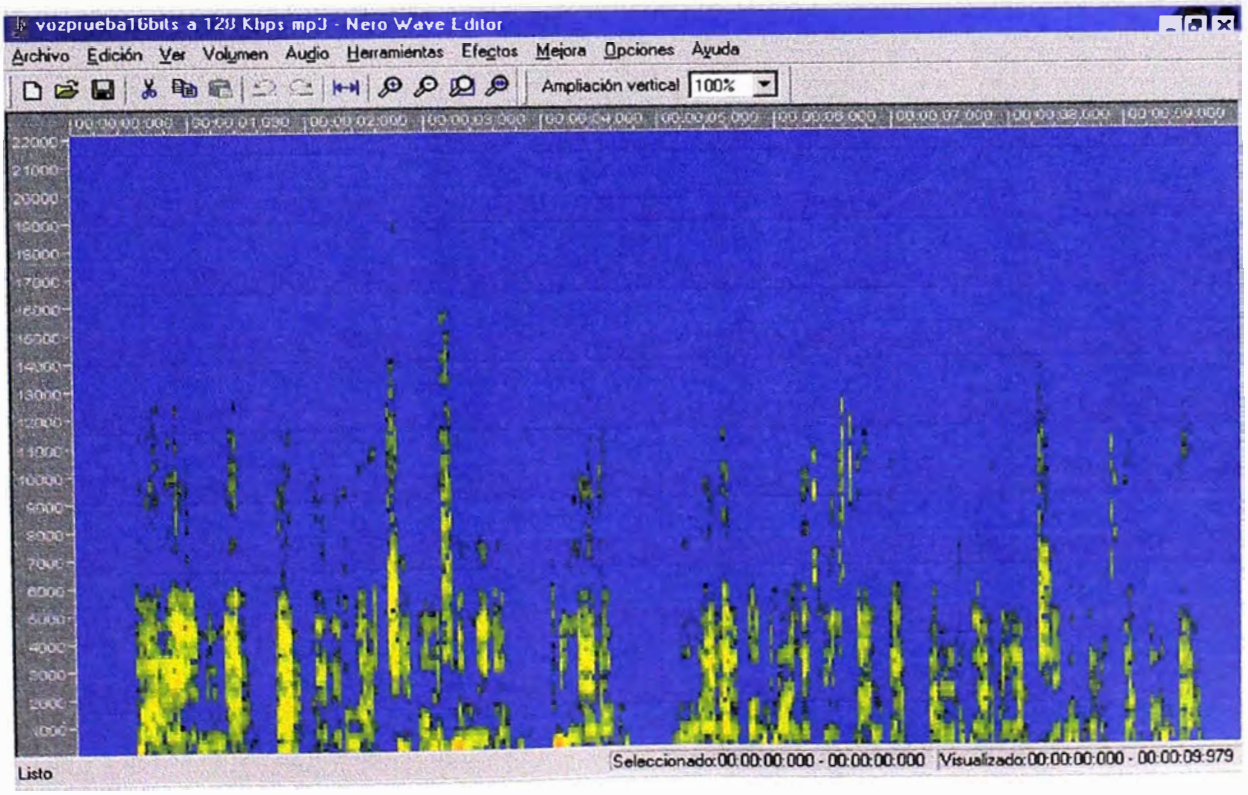
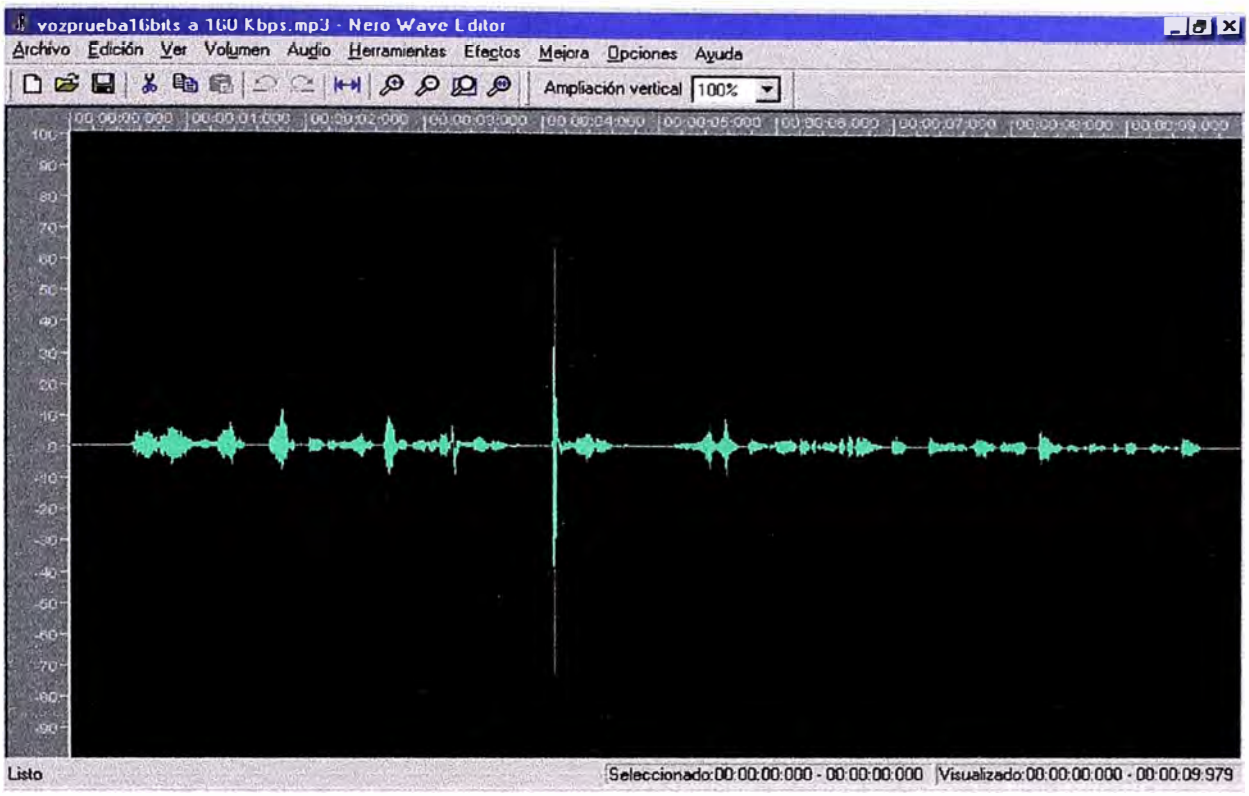


Figura PV21: **Vozprueba16bits a 128 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

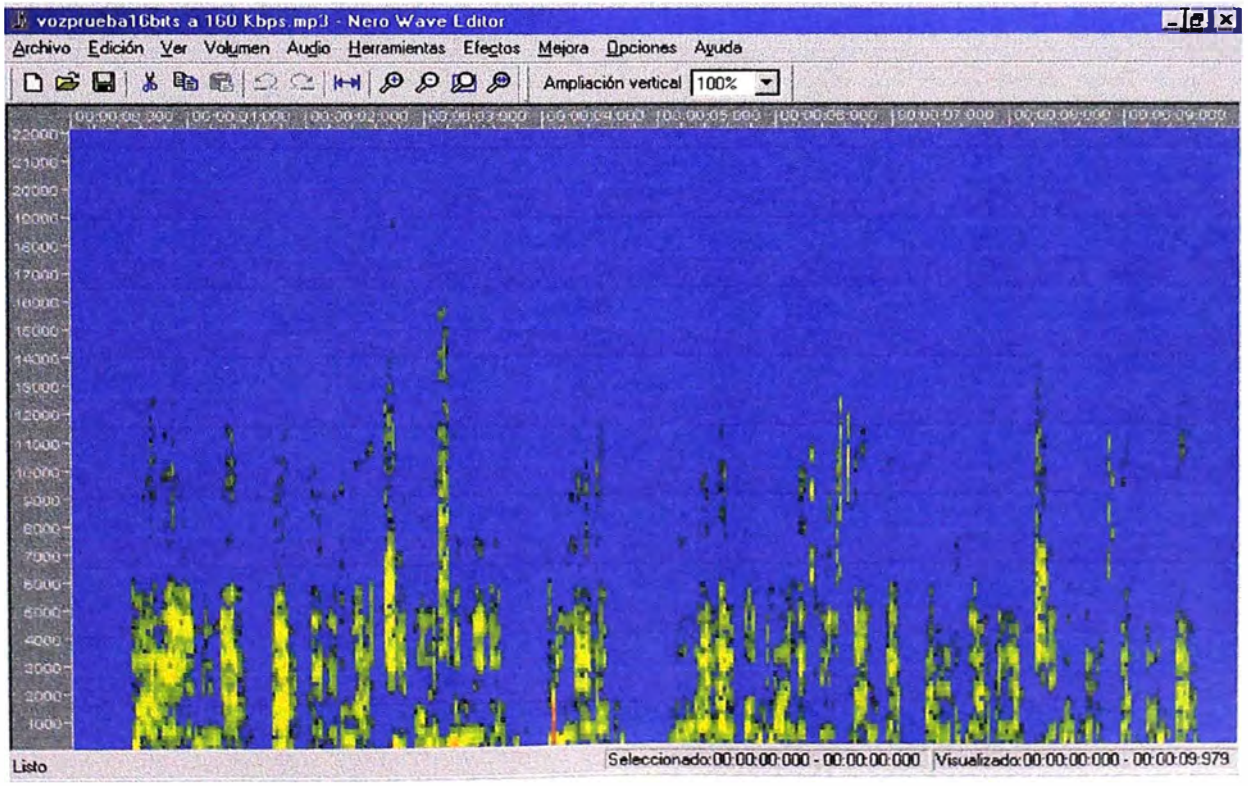
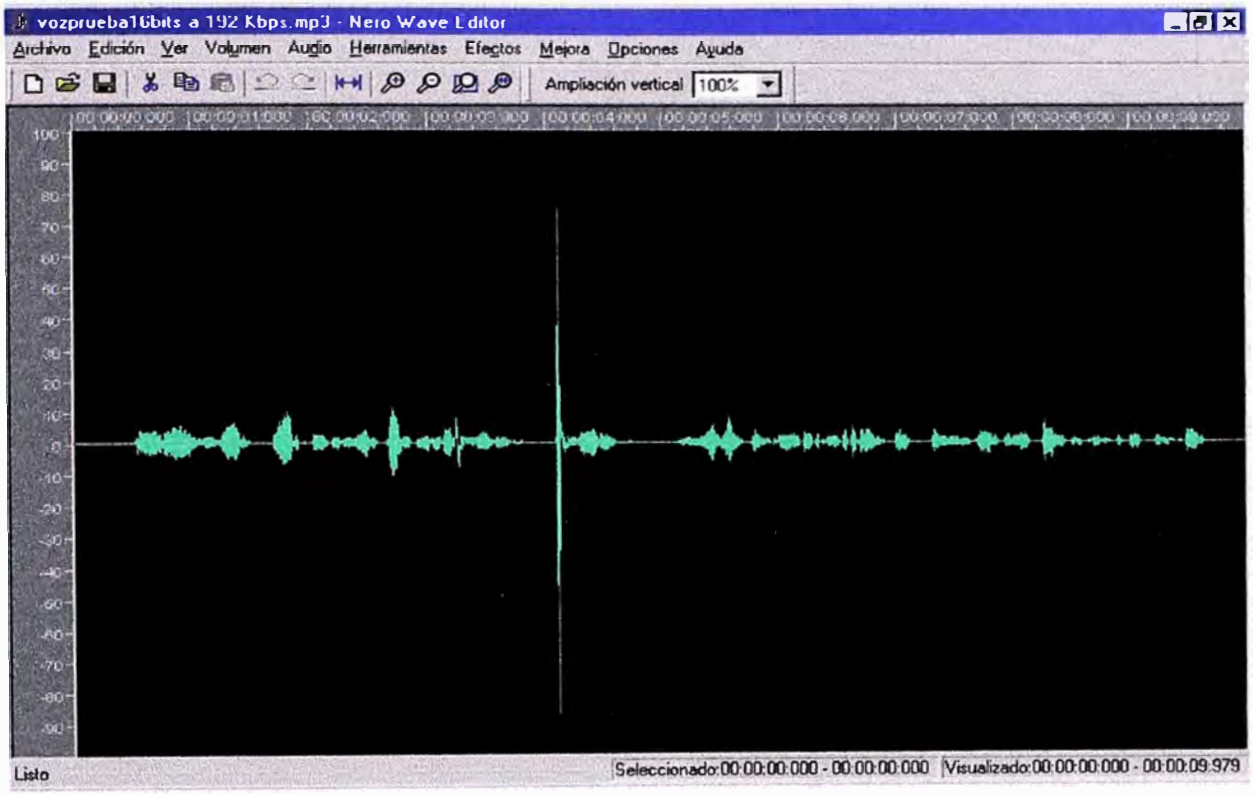


Figura PV22: **Vozprueba16bits a 160 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

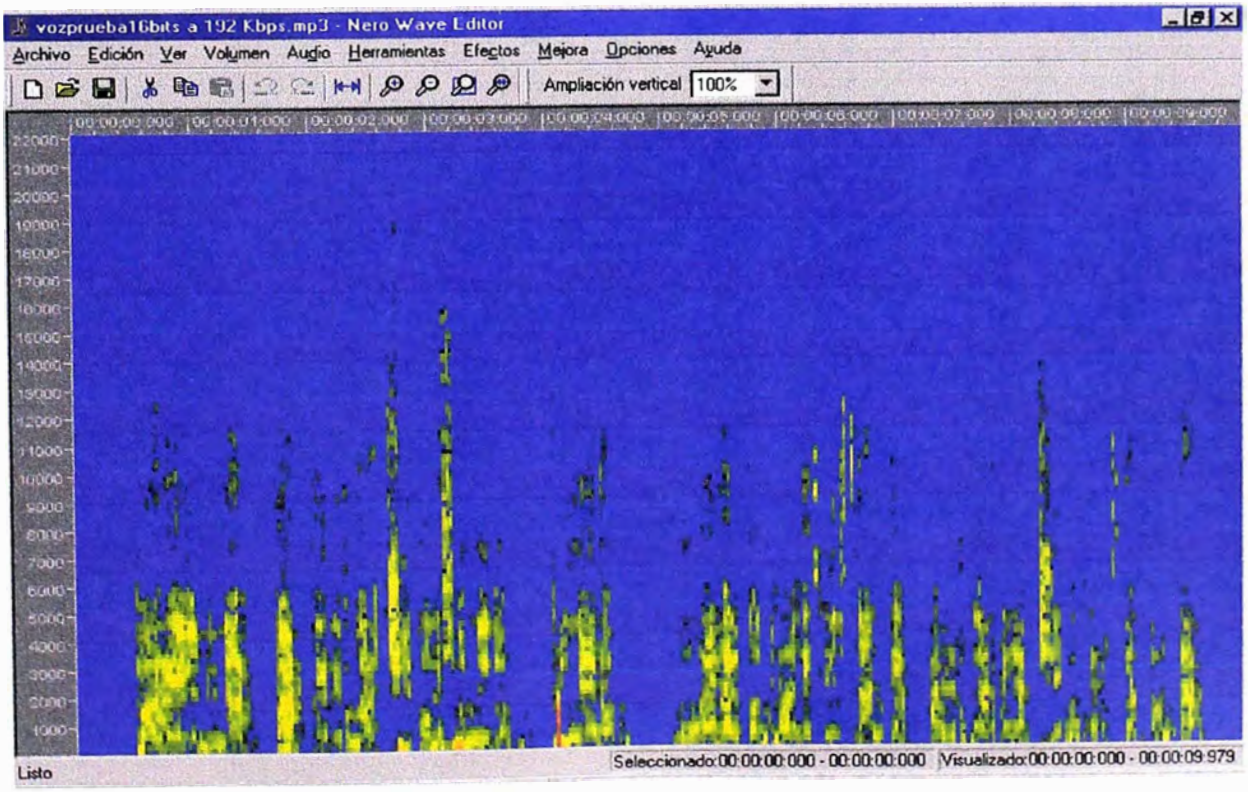


Figura PV23: **Vozprueba16bits a 192 Kbps.mp3**

Visualización de Onda

Visualización de Espectrograma

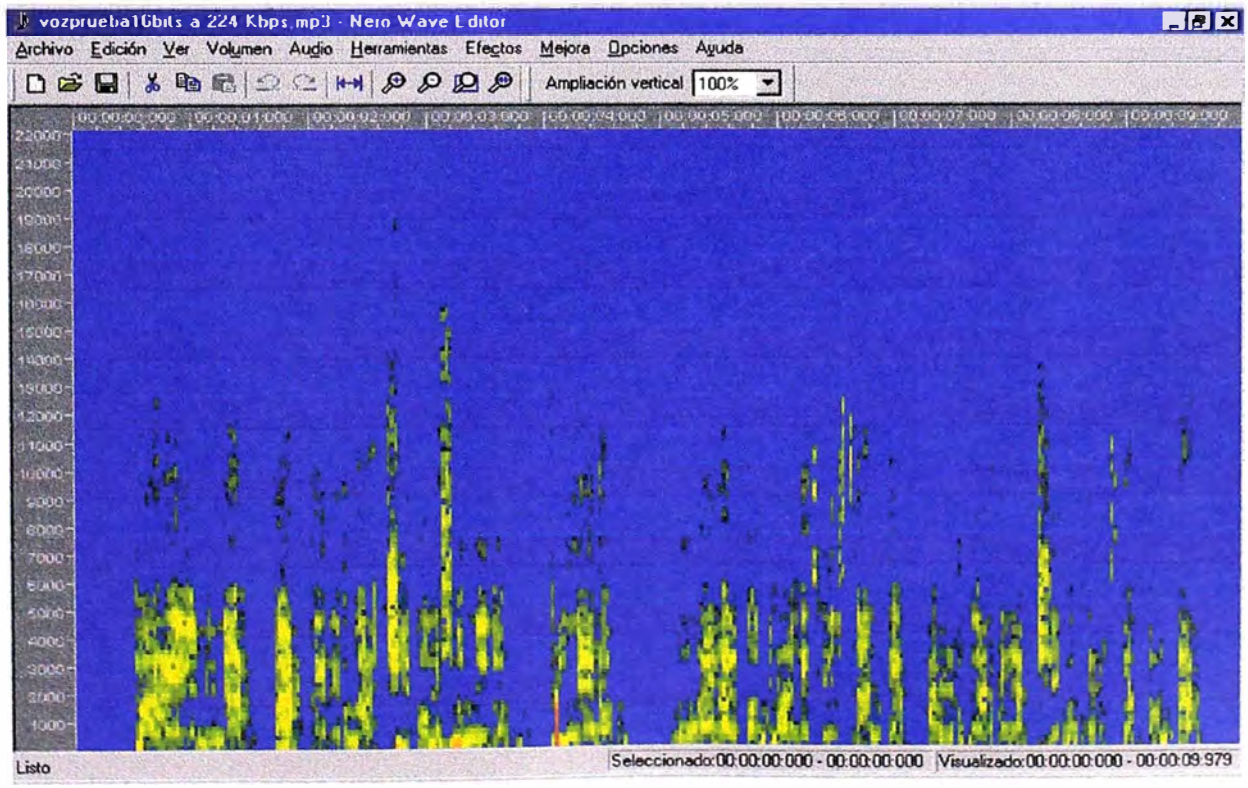
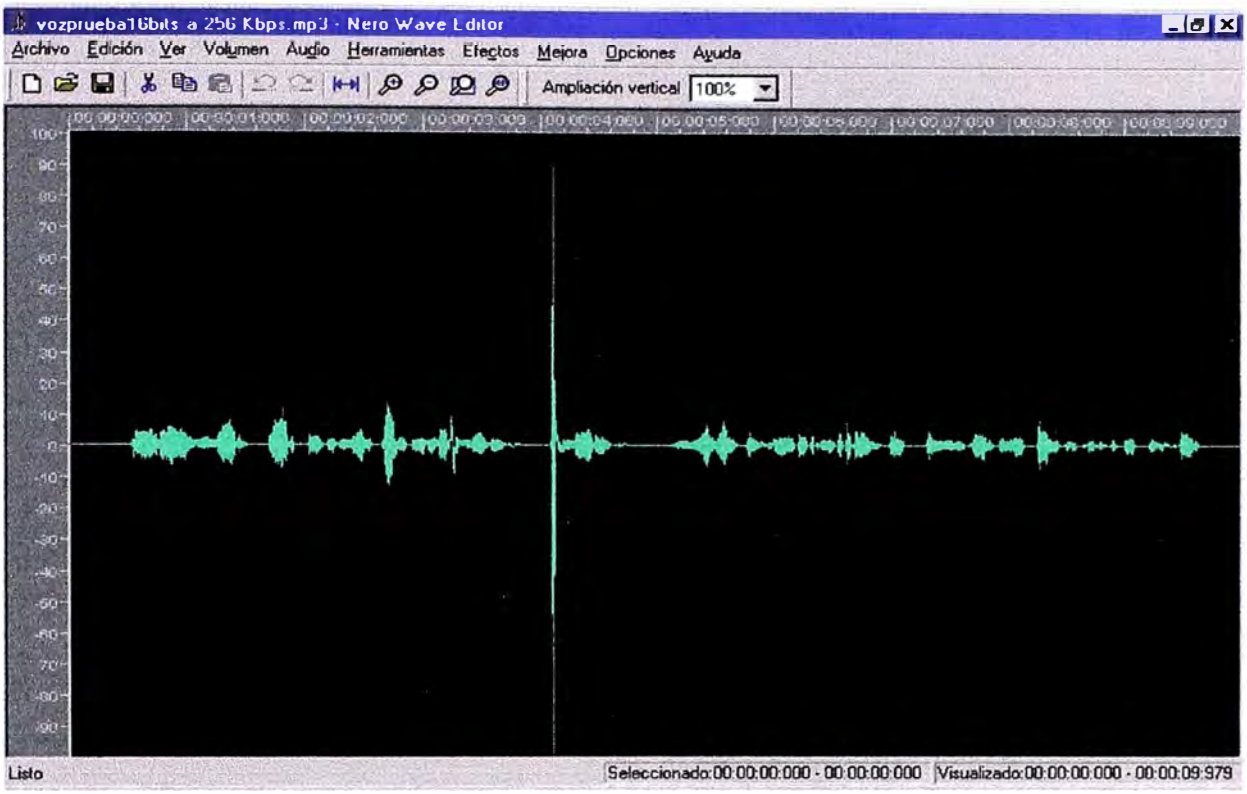


Figura PV24: **Vozprueba16bits a 224 Kbps.mp3**

Visualización de Onda



Visualización de Espectrograma

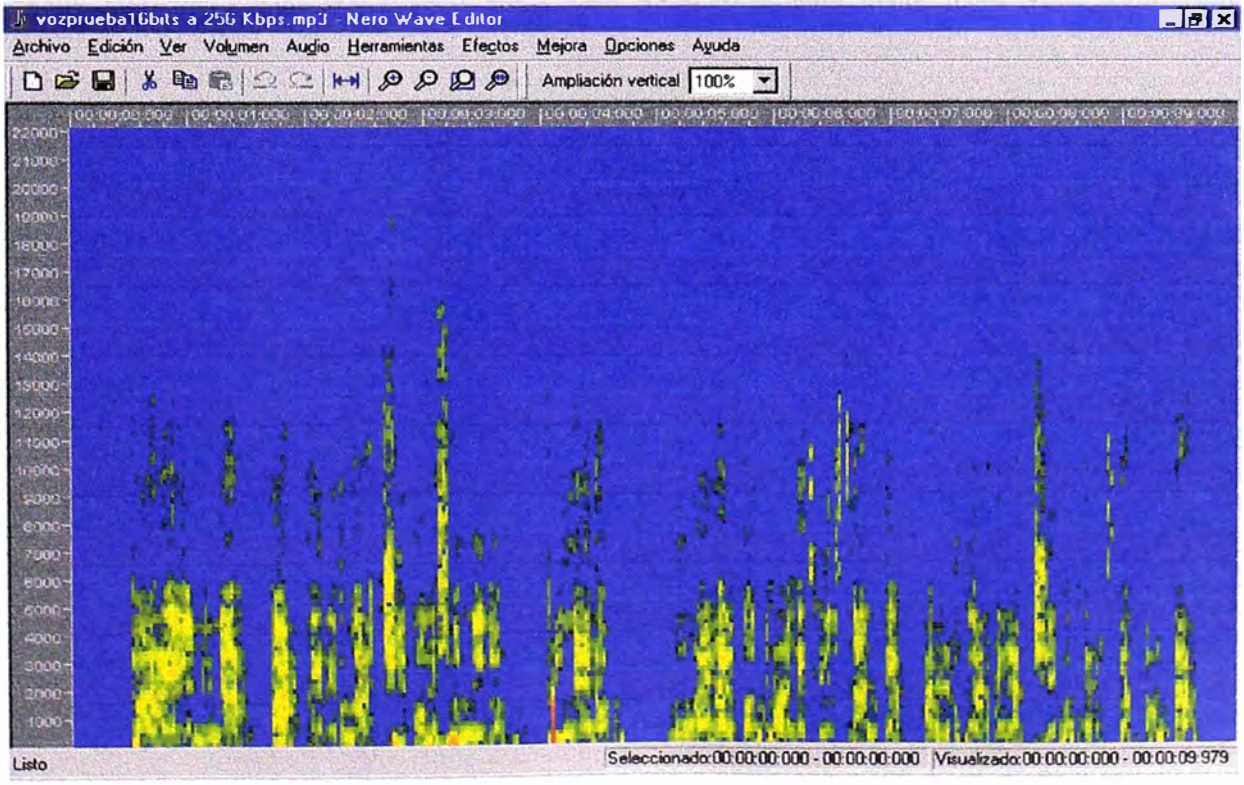


Figura PV25: **Vozprueba16bits a 256 Kbps.mp3**

Voces (con MP3 codificado a 10 segundos)			
Nombre del archivo de Voz	**Voz**	**Vozprueba16bits**	**Vozprueba8bits**
Tamaño original (Wav)	971 KB	1712 KB	891 KB
Duración (Wav)	11.265 segundos	19.867 segundos	20.667 segundos
Bitrate (Wav)	706 Kbps	706 Kbps	353 Kbps
Resolución (Número de bits)	16 bits	16 bits	8 bits
Tamaño de la muestra para comparación (Wav)	860 KB	860 KB	430 KB
Duración de la muestra para comparación (Wav)	9.979 segundos	9.979 segundos	9.979 segundos

Tasas de bits en Kbps para codificación en MP3:	**MP3 (Tamaño en KB)**			**Duración del archivo en segundos**			**Ejecución en minutos**			**Tasa de Compresión**		
	Voz	**Vozprueba16bits**	**Vozprueba8bits**	**Voz**	**Vozprueba16bits**	**Vozprueba8bits**	**Voz**	**Vozprueba16bits**	**Vozprueba8bits**	**Voz**	**Vozprueba16bits**	**Vozprueba8bits**
96	117	117	117	9.979	9.979	9.979	40'	36'	38'	1:7.3504	1:7.3504	1:3.6752
112	137	137	137	9.979	9.979	9.979	46'	43'	48'	1:6.2773	1:6.2773	1:3.1386
128	156	156	156	9.979	9.979	9.979	57'	52'	44'	1:5.5128	1:5.5128	1:2.7564
160	195	195	195	9.979	9.979	9.979	1h 19'	57'	53'	1:4.4102	1:4.4102	1:2.2051
192	234	234	234	9.979	9.979	9.979	1h 9'	1h 10'	1h 8'	1:3.6752	1:3.6752	1:1.8376
224	273	273	273	9.979	9.979	9.979	1h 27'	1h 22'	1h 16'	1:3.1501	1:3.1501	1:1.5750
256	312	312	312	9.979	9.979	9.979	1h 31'	1h 34'	1h 24'	1:2.7564	1:2.7564	1:1.3782
320	390	———	———	9.979	———	———	2h	———	———	1:2.2051	———	———

Tabla Pruebas 3: Resultados de las pruebas para voces.

CONCLUSIONES DE LAS PRUEBAS

En las pruebas de las canciones, se puede apreciar que en "Clásica" al ir aumentando la tasa de bits, se adquieren distorsiones audibles y visibles en los gráficos, como saltos que perjudican la calidad del sonido. Esto se aprecia sobre todo a partir de los 160 Kbps y aumentan conforme aumenta la tasa de bits. La calidad en general de las compresiones no es buena, se oyen ruidos de fondo.

En "Jazz" se puede apreciar lo anterior, o sea, comportamiento similar a "Clásica", con el sonido de las compresiones de mala calidad y con saltos a partir de 160 Kbps.

Los mejores resultados se han obtenido con "Rock" y "Pop" (Ahardday44), el sonido es mejor que en los géneros anteriores. En "Rock" los saltos empiezan a los 224 Kbps y en "Pop" no se producen estos.

Según lo que se sabe acerca de la tasa de bits, la calidad y el tamaño de un archivo serán bajos si esta tasa es baja y serán altos si ella lo es, pero según los resultados observados de las pruebas para canciones, el codificador estudiado no mejora la calidad de los archivos comprimidos conforme aumenta la tasa de bits, el raspado del ruido de fondo persiste y la mayor tasa adjunta saltos que perturban más la señal. Los saltos solo se presentan en los archivos de 16 bits.

De la tabla de resultados que se muestran en la Tabla Pruebas 1 (página 168), se observa que sí se cumple lo de *a mayor tasa de bits, mayor tamaño del archivo*. Tanto para una canción wav de 8 bits o 16 bits, el tamaño del archivo comprimido es el mismo, como se puede apreciar de las características originales, donde se ve que tres de ellas son de 16 bits y "Pop" es de 8 bits pero los archivos codificados tienen el mismo tamaño a la misma tasa de bits.

La tasa de compresión es del orden de mas de 7 a 1 (para menores tasas de bits) hasta mas de 2 a 1 (para mayores tasas de bits) para los archivos wav de 16 bits, y de mas de 3 a 1 hasta mas de 1 a 1 respectivamente para los archivos wav de 8 bits, concluyendo que se obtienen mayores compresiones a 96 Kbps y con archivos wav de 16 bits.

Cuando se trata con señales de una sola frecuencia o frecuencia uniforme, o sea señales sinusoidales, las pruebas nos muestran que el codificador estudiado no muestra eficiencia.

Las señales de 8 bits a 96 Kbps son deformaciones de la señal wav original y esto se da con mayor incidencia conforme se aumenta la frecuencia de la señal wav. Para las señales de 8 bits a 320 Kbps, la deformación es más acentuada, el resultado es peor que el caso anterior. En ambos casos la señal original sinusoidal wav ha sido totalmente deformada.

Las señales de 16 bits a 96 Kbps se reducen a pulsos o saltos que van disminuyendo en número conforme aumenta la frecuencia de la señal original, reduciéndose poco a poco a un solo pulso a partir de 160 Hz, este pulso está situado al inicio. A la frecuencia de 10240 Hz, aparte del pulso

inicial aparecen otros tres pulsos para luego, a la frecuencia de 20480 Hz, desaparecer casi por completo la señal. Los resultados son por lo tanto malísimos, para nada se acercan a la señal wav original.

Resultados similares al caso anterior se aprecian para las señales de 16 bits a 320 Kbps, excepto la señal no desaparece en la frecuencia de 20480 Hz, sino que existe todavía el pulso inicial. Se concluye como el caso anterior que el resultado es pésimo.

Este análisis basado en los gráficos concluye que, el codificador no se recomienda para señales sinusoidales. El mejor resultado se consigue con señales de 8 bits a 96 Kbps, pero aún así los resultados son muy malos. Para este tipo de señales, el codificador deforma la señal original cambiándole su frecuencia y adjuntándole armónicos que la hacen irreconocible y ruidosa.

De los resultados que se muestran en la Tabla 2 (página 236), se concluye que la compresión es mayor para las señales de 16 bits (de mas de 7 a 1 hasta de mas de 2 a 1 contra la compresión para 8 bits que es de mas de 3 a 1 hasta de mas de 1 a 1) y para una tasa de bits de 96 Kbps. Se cumple también que a mayor tasa de bits hay mayor tamaño del archivo.

En el caso de las pruebas con voces, para la primera prueba que es "Voz" se notó un resultado homogéneo para las codificaciones en todas las tasas de bits, el resultado es medianamente bueno puesto que a pesar de ser inteligible la voz y ser bastante parecida al original, se oye ruido de fondo, y aparecen las distorsiones de sonido como pulsos o saltos a partir de la codificación a 224 Kbps (un solo salto), aumentando estas distorsiones en la

codificación a 320 Kbps.

Para la segunda prueba de voz que corresponde a una voz femenina a 8 bits, la grabación original en wav presenta poca claridad, oyéndose sonido de raspado en el fondo, y los resultados de las codificaciones a las diferentes tasas de bits presentan adición de sonidos que se mezclan con la voz. A mayor tasa de bits, mayor cantidad de sonidos intrusos. Por lo tanto no mejora la calidad al aumentar la tasa de bits.

Para la tercera prueba de voz que corresponde a la misma voz femenina que la anterior pero a 16 bits, el original grabado en wav tiene mucha mejor calidad que el grabado a 8 bits siendo esto obvio porque la calidad de 16 bits es mejor, como es bien sabido. Los resultados de las codificaciones son mejores que las de 8 bits, pero son de poca calidad porque no son claras y se oye como una "segunda voz" lo que le quita limpieza a la señal, aparte de oírse también raspado. A los 160 Kbps aparece un solo salto o pulso de distorsión de la señal que se repite para las tasas de bits superiores.

De los resultados que se pueden ver en la Tabla 3 (página 263), se aprecia que también se cumple que a mayor tasa de bits hay mayor tamaño del archivo, que tanto para archivos de 8 bits como de 16 bits los archivos comprimidos o codificados a la misma tasa de bits tienen el mismo tamaño, igual que para las canciones y que también como estas para los archivos a 16 bits la compresión va de mas de 7 a 1 para 96 Kbps hasta mas de 2 a 1 para 320 Kbps y para el archivo de 8 bits de mas de 3 a 1 para 96 Kbps hasta mas 1 a 1 para 256 Kbps. Para los archivos "Vozprueba16bits" y "Vozprueba8bits" no se pudo obtener la codificación a 320 Kbps por

deficiencias que presentó la computadora.

Las conclusiones generales para las pruebas se resumirían en lo siguiente: contrariamente al codificador profesional, este codificador no da mejores resultados conforme aumenta la tasa de bits, para archivos wav de mejor calidad (los de 16 bits) no funciona bien (le adjunta saltos o pulsos), todos los archivos wav, tanto de 8 bits como de 16 bits, producen el mismo tamaño de archivo resultante de la codificación a la misma tasa de bits y si se cumple que a mayor tasa de bits hay mayor tamaño de archivo. Las mayores tasas de compresión se dan a menores tasas de bits y en archivos de 16 bits.

Tomando en cuenta que se trabajó con un codificador artesanal, es de mi opinión que los resultados son bastante aceptables para esa condición, dado que si se quiere trabajar con canciones o voz y a una tasa de bits no muy alta (como por ejemplo 128 Kbps que es la tasa recomendada para producir archivos con calidad cercana a la calidad del CD a la vez que se equilibra tamaño y calidad), se pueden obtener archivos con algunos defectos pero inteligibles a los cuales se puede someter a procesos para mejorar su calidad. Como las señales sinusoidales sirven solo de prueba y nada más y para ellas el codificador demostró su ineficiencia, se pueden obviar y se puede considerar que este codificador artesanal como se dijo antes, cumple mas o menos su cometido, aunque otro escollo grande es que no se pueden codificar archivos de alrededor de 20 segundos para adelante.

CONCLUSIONES

El MP3 no es mas que un archivo WAV de una calidad cercana al CD (pero no mayor que la del CD). El formato WAV es un formato de sonido (de onda) que abarca distintos grados de calidad de sonido en base al "bitrate", a la frecuencia de muestreo (Hz) así como al sonido estéreo o mono. Con un archivo WAV podemos conseguir una calidad de reproducción igual a la obtenida con un CD, el problema es el tamaño de estos archivos que hacen completamente impracticable no sólo la transmisión por Internet, sino el almacenamiento de canciones completas en el disco duro. Por medio de compresores especiales se consigue reducir el tamaño de estos archivos de una forma asombrosa y así, con el formato comprimido MP3, podemos tener en un poco mas de tres megabytes una canción que nos ocuparía 30, el porcentaje de reducción de estos archivos es de 10 a 12 veces, lo que hace muy practicable su distribución y almacenamiento a gran escala, y con calidad cercana al CD.

La calidad de los archivos MP3 sin duda alguna, no es comparable a la calidad del CD. Este es el mayor mito relativo al MP3. Estos ficheros sufren pérdidas de calidad notables sobre todo en los agudos y los graves.

Los archivos de formato WAV son muy básicos, son solamente muestras de sonido digitalizadas. Ellas son voluminosas pero simples; cualquier

computadora puede reproducirlos, y se oyen muy bien, y aunque ambos, WAV y MP3 suenen bien, como se pudo apreciar, las diferencias son profundas.

La cabecera del formato WAV está formada por 44 bytes, en cambio, la cadena de bits a la salida del codificador MPEG-1 está formada por frames (cuadros) que cuentan con un encabezamiento; un control de errores por medio del método de redundancia cíclica (CRC), los datos correspondientes al audio codificado e información auxiliar. Los mencionados frames o cuadros se almacenan en un archivo de computadora de manera secuencial. Estos archivos, los MP3, no contienen ningún tipo de encabezamiento general, simplemente son grandes "paquetes" de frames. No obstante esto, es posible encontrar al final del archivo, información sobre el origen del audio comprimido a manera de etiqueta ("tag" en inglés); de esta manera el dispositivo o programa que comprime una grabación musical puede adicionar el nombre del autor, de la obra, del álbum al que corresponde (con 30 caracteres asignados a cada uno), año de realización (en 4 bytes), comentarios (30 caracteres), y un índice numérico correspondiente al género musical (1 byte). El campo (si es especificado y es correcto) tiene como identificación inicial la palabra "TAG". Con estos tres caracteres totaliza 128 bytes.

WAV no requiere software adicional para ser reproducido, en cambio MP3 requiere reproductores especiales tales como Napster, WinAmp o Yahoo!Player por ejemplo.

Tanto el MP3, el VQF y el AAC son formatos que producen calidades de

audio variadas. Todos utilizan algoritmos de compresión acústicos diseñados según las propiedades del oído humano, ya que las frecuencias o los sonidos que no pueden ser percibidos por el oído humano se eliminan.

A pesar que muchos aficionados de la música creen que la calidad del audio es la característica que define perfectamente a dichos formatos, también deben tenerse en cuenta otros factores importantes, como por ejemplo, la facilidad para conseguirlos en la Web, su costo, el software para manipularlos, la cantidad de archivos disponibles en los diferentes sitios, etc.

Los nuevos formatos de audio digital MP3 y VQF, sustituirán los viejos formatos MIDI y WAV, y con muchas probabilidades, los formatos físicos de CD, vinilo o cassette, con el creciente desarrollo de interfaces y unidades de reproducción de estos formatos de audio. En estos momentos, en audio se está imponiendo el formato MP3, pero el formato VQF permite comprimir 1 minuto de audio en 545 KB cuando con MP3 serían 819 KB y una sensible pérdida de calidad. Quizás nos encontremos ante otro nuevo y futuro ejemplo del caso sucedido con los sistemas Betamax y Laser Disc, pero es algo difícil encontrar webs, encoders, y line-kits sobre el formato VQF.

El uso del formato MP3 se ha extendido por todo el mundo. Podemos encontrar multitud de servidores en todo el mundo que nos deleitan con los últimos éxitos del momento en formato MP3 para que podamos bajárnoslos a nuestras casas o a servidores de radio en formato MP3 para poderlas escuchar en cualquier lugar del mundo.

El audio nunca ha sido tan fácil de distribuir puesto que nunca antes ha ocupado tan poco espacio, esto ha hecho que la música sea recibida antes

en formato MP3 que en el propio CD de audio. Esto ha hecho que haya una auténtica revolución para las discográficas o los propios autores de música ya que el conseguir una copia de su música de forma pirata nunca ha sido tan fácil.

Por esto hay unos movimientos para poder codificar el audio en formato MP3, para que sea de pago. Este movimiento esta llevado por la RIAA (Recording Industria Asociation of America) y la SDMI (Secure Digital Music Iniciative) para proteger el copyright de las canciones y los derechos de los músicos, que ha tenido muy mala aceptación por parte de los "internautas". Aunque muchos músicos desconocidos y no tanto, ofrecen su música por Internet, evitando a las casas discográficas, y así se dan a conocer de forma más económica.

Otra de las consecuencias que ha tenido la expansión de este formato ha sido los lectores. Normalmente para poder disfrutar de la música comprimida en MP3 es necesario tener un ordenador con una tarjeta de sonido que se encargue de descomprimir en tiempo real el fichero MP3.

Esto es una desventaja, puesto que para oír música en formato MP3 necesitábamos un ordenador, algo que no todo el mundo puede tener en algunas situaciones (nadie lleva un ordenador dentro de un automóvil, o se lleva su portátil por la calle para escuchar su música favorita). Por ello se están creando unos descompresores de MP3 en formato hardware del tamaño de walkmans para poder llevar. Por ello se dice que el futuro del MP3 es muy alentador, ya que cada día existen más empresas que lanzan al mercado estos reproductores.

Aunque últimamente se está hablando mucho sobre el MP4, mayor calidad de sonido y menor tamaño que el MP3, así como del formato propuesto por Microsoft, el WMA, la mitad de un MP3 y que preserva los derechos de autor, el futuro tecnológico del MP4 aún es algo incierto. Lo que está claro es que independientemente del fundamento tecnológico seguirá habiendo compartición de música a través de Internet. Nadie podrá evitar el intercambio de canciones, y si algún formato tecnológico lo impide, simplemente no será aceptado.

Lo que sí es cierto es que Internet está cambiando al mundo. Y como no iba a ser menos también está cambiando la forma de generar dinero. Durante los próximos años veremos como muchas formas tradicionales de hacer negocio evolucionan o simplemente desaparecen.

Y debido al empuje del MP3, de aquí a unos años la industria discográfica no será la misma, habrá tenido que cambiar radicalmente.

BIBLIOGRAFÍA

Introducción

[1] Tratamiento de Señales en Tiempo Discreto. 2da edición.
Alan V. Oppenheim, Ronald W. Schafer.

[2] Tratamiento Digital de Señales. Principios, algoritmos y aplicaciones.
3era. edición.
John G. Proakis, Dimitris G. Manolakis.

[3] Señales y Sistemas. Segunda edición.
Alan V. Oppenheim, Alan S. Willsky, S. Hamid Nawab.

[4] Audio Digital.
www.terra.es/personal/fcyborg/musica/audiodig.html

[5] Compresión de Audio, por Mikel.
www.hispamp3.com/tallermp3/tutoriales/mp3profundidad/2.shtml

[6] Codificación de Audio en PCM.
Glosarios de términos técnicos y electrónicos.
www.babylon.com

[7] CYBORG Informática Musical.
www.terra.es/personal/fcyborg/mp3/mp3-crear.html

[8] Codificación de Audio en PCM.
www.geocities.com/TimesSquare/Battlefield/1401/mp32.htlm

[9] Música Digital.
www.geocities.com/Tokio/Towers/1811/mp3.htm

[10] MP3 a fondo.
www.gui.uva.es/~tuti/mp3/mp3afondo.html

Capítulo I

[11] CYBORG Informática Musical.

www.terra.es/personal/fcyborg/mp3/sa.html

[12] Formatos de Archivo.
www.musiclab.es/formatos.html

[13] Formatos para el audio.
www.lacompu.com/mp3/notas/formatos

[14] La fuente de la música.
www.gui.uva.es/~tuti/mp3/mp3.html

[15] Formatos de sonido digital.
www.conganat.org/iicongreso/comunic/008/sonido.htm

[16] Formatos de audio multicanal.
www.audiolav.com/noticias/formatos/formatosaudiomulticanal.htm

[17] Tutoriales.
www.logueres.com/dvdxplorer/info.htm

[18] Formatos de audio en Internet.
www.iua.upf.es/~perfe/cursos/eines/internetaudio.html

[19] Formatos de audio.
www.tejedoresdelweb.com/307/article-1053.html

Capítulo II

[20] Tecnología: Sonido.
www-lifia.info.unlp.edu.ar/tmm/teo2.html

[21] Formato de los ficheros de Sonido WAV.
www.upv.es/protel/usr/jotrofer/sonido/sound.htm

[22] WAV
Glosarios técnicos y electrónicos.
www.babylon.com

[23] Ficheros WAV de sonido.
www.conganat.org/iicongreso/comunic/008/wav.htm

Capítulo III

[24] MIDI
Glosarios técnicos y electrónicos.
www.babylon.com

[25] MIDI
www.terra.es/personal/fcyborg

[26] El Rincón Musical Peruano.
MIDI: Preguntas y respuestas.
www.musicaperuana.com/espanol/ayuda.htm

[27] MIDI 101
www.omnisphere.com/macsupp/MIDI101.html

Capítulo IV

[28] Audio Compression-Psychoacoustics.
www.cs.sfu.ca/undergrad/CourseMaterials/CMPT479/material/notes/Chap4/Chap4.3/Chap4.3.html

[29] Guida agli MP3 - Aspetti Tecnici
- Effetto "Frecuenze Maschera"
  www.euro2001.com/mp3/mp3/asptecn2.htm
- Effetto "Maschera temporale"
  www.euro2001.com/mp3/mp3/asptecn3.htm
- Metodo di compressione dell'MPEG1 Layer III
  www.euro2001.com/mp3/mp3/asptecn4.htm

[30] Apéndice 2: El modelo Psicoacústico, por Mikel.
www.hispamp3.com/tallermp3/tutoriales/mp3profundidad/7.shtml

Capítulo V

[31] Compresión de Audio.
www.fuac.edu.co/autonoma/pregrado/ingenieria/ingelec/proyectosgrado/compresvideo/compresion-audio.htm

[32] ¿Qué es un MP3?, por JohnnyB.
www.hispamp3.com/tallermp3.como/queesunmp3.shtml

[33] MP3 en profundidad, por Mikel.
www.hispamp3.com/tallermp3/tutoriales/mp3profundidad/mp3profundidad.shtml

[34] ¿Qué es ISO MPEG 1 Capa 3 (MP3)?
Codificación & decodificación en MP3.
www.angelfire.com/co/mp3Colombia

[35] Overview of the MP3 techniques.
www.mp3-tech.org/tech.htlm

[36] MP3 software.
utenti.tripod.it/nonsolomp3/mp3soft.htm

[37] El MP3 inicia la revolución de la música digital.
www.baquia.com/com/legacy/8441.html

[38] Aspectos legales del MP3.
www.terra.es/personal/fcyborg/mp3/mp3-legalidad.html

[39] ¿Qué es el MP3?
www.audiotest.org/esp/mp3.htm

[40] El ABC del MP3.
www.terra.com.ar/canales/informesespeciales/0/166.html

[41] Detalles técnicos del MP3.
new.sonico.com/educacion/avanzado/3.htm

[42] www.noticiasMP3.com

[43] www.mp3-tech.org

[44] Organización MPEG.
www.mpeg.org/MPEG/

[45] Documentos técnicos sobre MP3, AAC, MPEG4, etc.
www.mp3-tech.org/programmer/docs/index.html

[46] MDCT (Modified Discrete Cosine Transform) and MPEG audio encoding.
www.uq.net.au/~zzmcheng/mdct/mdct/mdct.html

[47] DCT
Tratamiento de Señales en Tiempo Discreto. 2da edición.
Alan V. Oppenheim, Ronald W. Schafer.

[48] Official MPEG audio.
www.tnt.uni-hannover.de/project/mpeg/audio/

[49] Codificador MP3 en Matlab.
members.fortunecity.com/alex1944/mp3coding

[50] Subband coding.
www.otolith.com/pub/u/howitt/sbc.tutorial.html

[51] Audio compression.
www.cs.sfu.ca/undergrad/CourseMaterials/CMPT479/material/notes/Chap4/Chap4.3/Chap4.3.html

[52] Guida agli MP3 - MPEG audio Layers.
www.euro2001.com/mp3/mp4/mpeg-audio-layers.htm

Capítulo VI

[53] Nuevos formatos de compresión.
www.gui.uva.es/~tuti/mp3/mp3.html

[54] Guida agli MP3 - Presente e Futuro.
www.euro2001.com/mp3/mp7/presefut.htm

[55] MP3Pro, la pesadilla continúa.
www.baquia.com/com/20010620/art00012.print.html

[56] Otros formatos de audio.
members.es.tripod.de/liquid0/mp3/mp_otro.htm

[57] Noticias sobre "MP4"/a2b.
www.zoom.es/~loboshs/mp3.html

[58] MP3 de principio a fin.
www.mp3.com.org/queesmp3.html

Anexo

CODIFICADOR MP3 EN MATLAB

Este es un programa que convierte un archivo de sonido WAV de Microsoft, en un archivo de sonido MP3, de acuerdo a las especificaciones dadas en el estándar internacional ISO/IEC 11172-3.

El archivo WAV debe ser monofónico, con frecuencia de muestreo igual a 44100 Hz y en formato PCM.

El archivo resultante es creado con el mismo nombre del archivo WAV original, y está codificado en formato MPEG-1 Capa III, con las siguientes características:

- Tasas de transferencia = 96...320 Kbps.
- Frecuencia de muestreo = 44100 Hz.
- Modo = Monofónico.

El proyecto consiste de varios archivos-m relacionados entre sí, pensados para funcionar como un todo, y no para su utilización independiente; aunque después de entender el código, pueden ser personalizados. Los archivos que se incluyen en este proyecto son:

1) Archivo principal:
   - Wav2mp3.m
2) Filtro híbrido:
   - Ci.mat

- Filtro_subbanda.m
- Transf_discreta_coseno.m
- Aliasing.m

3) Modelo Psicoacústico I:

- Analisis_fft.m
- Limites_banda_critica.m
- Componentes_tonales.m
- Reduccion.m
- Umbrales_enmasc_individual.m
- Umbral_enmasc_global.m

4) Cuantización/Codificación:

- Distorsion_permitida.m
- Ciclo_interno.m
- Huffman.m
- Ciclo_externo.m

5) Formato:

- Encabezado.m
- Info_secundaria.m
- Datos_principales.m

Los archivos son totalmente modificables, de acuerdo con lo que cada usuario desee visualizar del programa. Adicionalmente, se incluyen tres (3) archivos que pueden ser usados por cualquier persona interesada en ampliar el codificador hacia las Capas I y II del formato de audio MPEG-1. Estos archivos extra intentan facilitar la ampliación y mejoramiento del

codificador. Los 3 archivos adicionales son (todos pertenecientes al modelo psicoacústico I):

- Factores_escala.m
- Nivel_presion_sonora.m
- Umbral_enmasc_minimo.m

El código ha sido probado en MATLAB 5.2 corriendo sobre Windows 98. Para empezar a utilizar el programa, se deben copiar todos los archivos en un directorio del disco duro que esté incluido en el PATH de Matlab y luego, llamar la rutina "Wav2mp3" desde la ventana de comandos.

Este programa está pensado de una manera didáctica, y el objetivo principal no es un desempeño óptimo en cuanto a velocidad o calidad en el sonido, sino mostrarlo más detalladamente posible cómo funciona el esquema de codificación MP3.

En cuanto a la velocidad de codificación, lo más recomendable es codificar tiempos pequeños, ya que el codificador en Matlab no fue pensado para trabajar en tiempo real, además de que la manera en que Matlab está construido, no lo permite.

Este programa está basado en:

Information Technology -- Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s -- Part 3: Audio. ISO/IEC. Genève, Switzerland. First Edition. 1993-08-01. ISO/IEC IS 11172-3.

Autores:

Alejandro Duque González

Erwin Alexander Vargas Restrepo

Universidad Pontificia Bolivariana, Facultad de Ingeniería Electrónica.

Directora del trabajo de grado:

Gloria Elena Cárdenas Soto. Especialista en Telemática. Universidad de Antioquía.

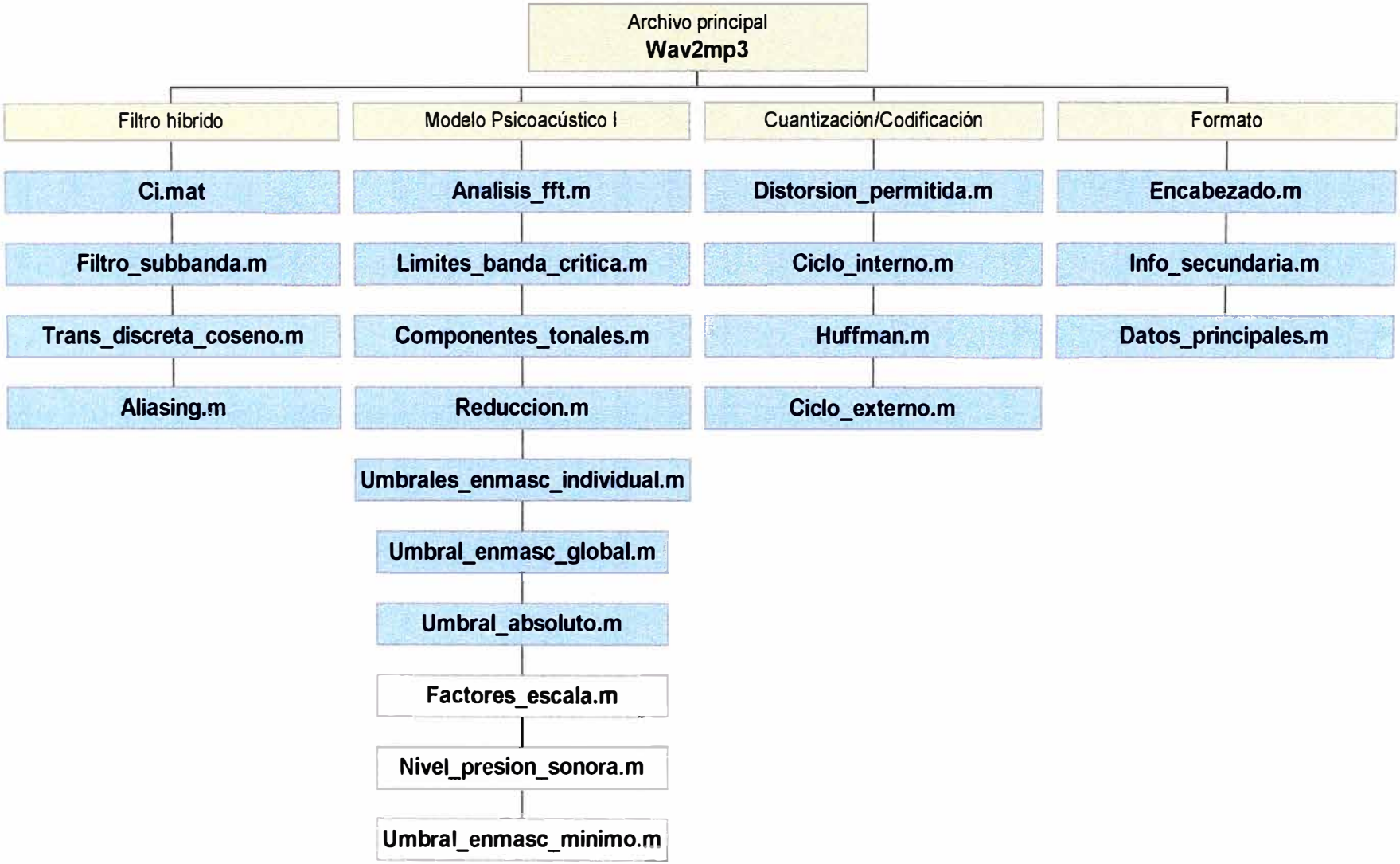
Agradecimientos a:

Ricardo García, Magister in Music Engineering, por la idea original de este proyecto y por facilitar la consecución del estándar internacional en la biblioteca del MIT.

Gloria Cárdenas, Especialista en Telemática, por su trabajo de revisión y por sus sugerencias.

Jorge Londoño, Magister in Computer Science, por la asesoría prestada.

Diagrama de bloques del Codificador MP3



```
%WAV2MP3 Convierte un archivo WAV de Microsoft en un archivo MP3.
%
%   Convierte un archivo de sonido WAV de Microsoft, en un archivo de
%   sonido MP3, de acuerdo a las especificaciones dadas en el estándar
%   ISO/IEC 11172-3.
%
%   El archivo WAV debe ser monofónico, con frecuencia de muestreo igual
%   a 44100 Hz y en formato PCM.
%
%   El archivo resultante es creado con el mismo nombre del archivo WAV
%   original, y está codificado en formato MPEG-1 Capa III, con las
%   siguientes características:
%
%   - Tasas de transferencia = 96...320 Kbps.
%   - Frecuencia de muestreo = 44100 Hz.
%   - Modo = Monofónico.
%
%   Noticia Legal:
%   El estándar ISO/IEC 11172-3 es propiedad de la Organización Internacional
%   para la Estandarización ISO. Todos los derechos reservados.

clear all
clc
disp(' ')
disp('                    CODIFICADOR MP3 EN MATLAB')
disp(' ')
disp('    Este programa convierte un archivo de sonido WAV de Microsoft, en un')
disp('    archivo de sonido MP3, de acuerdo a las especificaciones dadas en el')
disp('    estándar internacional ISO/IEC 11172-3.')
disp(' ')
disp('    El archivo WAV debe ser monofónico, con frecuencia de muestreo igual')
disp('    a 44100 Hz y en formato PCM.')
disp(' ')
disp('    El archivo resultante es creado con el mismo nombre del archivo WAV')
disp('    original, y está codificado en formato MPEG-1 Capa III, con las')
disp('    siguientes características:')
disp(' ')
disp('    - Tasas de transferencia = 96...320 Kbps.')
disp('    - Frecuencia de muestreo = 44100 Hz.')
disp('    - Modo = Monofónico.')
disp(' ')
disp(' ')

% Declaración de variables globales, las cuales son:
% X: Búfer FIFO del filtro subbanda.
% XR: Vector de muestras subbanda.
% gr: Gránulo (escalar).
% xmin: Vector de distorsiones permitidas.
% scalefac_l: Vector que contiene los factores de escala para bloques largos.
% bin_str: Cadena de caracteres, en la cual se almacenan todos los datos del
%          archivo MP3 (en formato binario) que deben ser escritos en el disco
%          duro.
global X XR gr xmin scalefac_l bin_str

% Inicializa el búfer FIFO (vector X) del filtro subbanda para el análisis.
% Inicializa la matriz S que almacena los valores del filtro subbanda.
X = zeros(1,512);
```

```
S = zeros(36,32);

% Asigna el archivo WAV de entrada a codificar.
archivo = input('Ingrese el nombre del archivo WAV que desea codificar\n','s');

% Obtiene la información del archivo WAV a codificar.
[PMA,Fs,bits] = wavread(archivo,1); % Sólo lee la primera muestra.
SIZ = wavread(archivo,'size'); % Obtiene el tamaño total del archivo.

% Si la frecuencia de muestreo no es 44100 Hz, se interrumpe la codificación.
if Fs ~= 44100
  fserror = ['Archivo WAV a ',int2str(Fs),' Hz no soportado.'];
  error(fserror)
end

% Si el archivo no es monofónico, se procesa el canal izquierdo.
if SIZ(2) == 2
  warning('El archivo WAV es estereofónico, se procesará el canal izquierdo.')
end

% Asigna la tasa de transferencia.
disp(' ')
disp('Las siguientes tasas de bits están disponibles (en Kbps):')
disp('96, 112, 128, 160, 192, 224, 256, ó 320')
tasa = input('Ingrese la tasa de bits deseada\n');
disp(' ')

% Inicialización de variables, de acuerdo con la tasa de bits escogida.
switch tasa
case 96
  bitrate_index = 7; % Índice de la tasa de bits.
  ajuste = 1; % Para ajustar el tamaño entero en bytes de la trama.
  ajgg = 11; % Para ajustar global_gain.
case 112
  bitrate_index = 8;
  ajuste = 2;
  ajgg = 10;
case 128
  bitrate_index = 9;
  ajuste = 3;
  ajgg = 9;
case 160
  bitrate_index = 10;
  ajuste = 1;
  ajgg = 9;
case 192
  bitrate_index = 11;
  ajuste = 3;
  ajgg = 8;
case 224
  bitrate_index = 12;
  ajuste = 1;
  ajgg = 8;
case 256
  bitrate_index = 13;
  ajuste = 3;
  ajgg = 7;
```

```
case 320
   bitrate_index = 14;
   ajuste = 3;
   ajgg = 7;
otherwise
   error(['La tasa de bits escogida (', num2str(tasa),' Kbps) no es soportada.'])
end

% Asigna el tiempo de codificación.
max_rsize = fix(SIZ(1)/1152)*1152;
seg = max_rsize/44100;
tiempo = input(['¿Cuántos segundos desea codificar (mínimo 0.10449, máximo ',...
      num2str(seg),' seg.)?\n(ENTER para codificar todo el archivo).\n']);

% Obtiene el último valor del número de muestras del archivo WAV que sea múltiplo
% de 1152. En otras palabras, se obliga a que la cantidad de muestras del archivo
% WAV sea múltiplo de 1152. En este paso, se eliminan entre 1 y 1151 muestras PCM
% (son las últimas muestras del archivo WAV y, por lo tanto, no son procesadas).
if isempty(tiempo)
   rsize = max_rsize;
else
   rsize = fix(tiempo*44100/1152)*1152;
end
if rsize < 4608
   error('La cantidad de tiempo es insuficiente')
elseif rsize > max_rsize
   warning(['El archivo WAV sólo dura ', num2str(seg),...
         ' segundos, procesando todo el archivo.'])
   rsize = max_rsize;
end

disp('Okay, espere unos minutos...')

% Carga las tablas necesarias para el análisis psicoacústico.
[UA,MAP,UES] = Umbral_absoluto;
LBC = Limites_banda_critica;

% Carga las tablas necesarias para los códigos de Huffman.
% Tablas A y B para los cuádruplos count1, proporcionadas
% por el estándar ISO 11172-3.
tabla_0 = [1 4 4 5 4 6 5 6 4 5 5 6 5 6 6 6; % hlen.
   1 5 4 5 6 5 4 4 7 3 6 0 7 2 3 1]; % hcod en formato decimal.
tabla_1 = [ones(1,16)*4; % hlen.
   15:-1:0]; % hcod en formato decimal.

% Carga los coeficientes de la ventana del análisis (vector C).
load('Ci.mat')

% Tabla de las bandas del factor de escala (Para bloques largos y 44.1 kHz).
SFBT = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21;
   4 4 4 4 4 4 6 6 8 8 10 12 16 20 24 28 34 42 50 54 76;
   1 5 9 13 17 21 25 31 37 45 53 63 75 91 111 135 163 197 239 289 343;
   4 8 12 16 20 24 30 36 44 52 62 74 90 110 134 162 196 238 288 342 418];

% Tabla para scalefac_compress.
SFC = [0 0 0 0 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4; % slen1.
   0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3; % slen2.
```

```
  0 1 2 3 5 5 6 7 8 8 9 10 4 11 12 13 14 14 14 15]; % scalefac_compress.

% Calcula el número promedio de bits disponibles para main_data, por gránulo
% (sin incluir el bit_reservoir).
% Inicializa las variables usadas para el control del formato.
init_mean_bits = fix(tasa*1000*(1152/44100)/2)-(84+ajuste);
bit_reservoir = 0;
padding_bit = 0;
rest = 0;

% Inicialización de variables para el formato del flujo de bits, de acuerdo con
% la tasa de bits escogida. Se inicializan:
% main_data_begin: Indica dónde empiezan los datos principales de la primera trama.
% mbr: Máximo permitido para bit_reservoir.
% mmdb: Máximo permitido para main_data_begin.
% mdt: Cantidad máxima de bits para main_data, por trama.
main_data_begin = 0;
if tasa == 96 | 112 | 128 | 160
  mbr = init_mean_bits*2;
else
  mbr = 4088;
end
mmdb = mbr/8;
mdt = init_mean_bits*2;

% Ciclo Principal. Analiza grupos de 1152 muestras PCM.
% Cada grupo se convierte en una trama MP3.
for a = 1:1152:rsize,

  % Almacena los últimos 576 valores de la trama anterior (matriz S).
  U576 = S(19:36,:);
  S = []; % Reinicia la matriz del filtro subbanda para cada gránulo.
  MDCT = []; % Reinicia la matriz MDCT.
  system_const = 8; % Reinicia el valor de la constante del sistema.

  % Lee el valor de 1152 muestras PCM desde el archivo de audio WAV.
  ENT = wavread(archivo,[a a+1151]);

  %%% ANÁLISIS PSICOACÚSTICO (Modelo I), Parte 1.

  % Realiza un análisis FFT para calcular la densidad espectral de potencia.
  F = Analisis_fft(ENT);

  % Encuentra las componentes tonales (senoidales) y no-tonales (ruidosas)
  % de la señal de audio.
  [BS,LT,LNT] = Componentes_tonales(F,UA,MAP,LBC);

  % Reduce las componentes enmascarantes: elimina todas las componentes
  % enmascarantes irrelevantes.
  [BSR,LTR,LNTR] = Reduccion(LT,LNT,BS,UA,MAP);

  % Calcula los umbrales de enmascaramiento individual.
  [UET,UENT] = Umbrales_enmasc_individual(F,LTR,LNTR,UA,MAP);

  % Calcula el umbral de enmascaramiento global.
  UEG = Umbral_enmasc_global(UES,UET,UENT);
```

```
% Determina el umbral de enmascaramiento mínimo en cada subbanda.
% Este paso no se realiza para la Capa III. Se puede usar para ampliar
% el codificador a las Capas I y/o II.
%UEM = Umbral_enmasc_minimo(UEG,MAP);

%%% FILTRADO SUBBANDA, Parte 1.

% Filtrado subbanda para el análisis. En la capa III se obtienen 36
% muestras subbanda consecutivas en el tiempo para cada una de las
% 32 subbandas (matriz S de 1152 valores).
for b = 1:32:1152,
  S = [S; Filtro_subbanda(ENT(b:b+31),C)];
end

%%% ANÁLISIS PSICOACÚSTICO (Modelo I), Parte 2.

% Los siguientes pasos no se realizan para la Capa III, por lo tanto, han sido
% desactivados. Pueden ser usados por cualquier persona interesada en el proyecto,
% para ampliar el codificador a las Capas I y/o II.

% Calcula los factores de escala.
%FDE = Factores_escala(S);

% Determina el nivel de presión sonora en cada subbanda.
%NPS = Nivel_presion_sonora(F,FDE);

% Calcula la relación señal a máscara.
%SMR = NPS-UEM;

for gr = 1:2,

  %%% FILTRADO SUBBANDA, Parte 2.

  % Calcula la MDCT con 50% de solapamiento.
  for sb = 1:32,
    MDCT = [MDCT; Transf_discreta_coseno(S,U576,sb,gr)];
  end

  % Reduce el aliasing introducido por el 50% de solapamiento de la MDCT.
  XR = Aliasing(MDCT(1+576*(gr-1):576*gr));

  %%% CUANTIZACIÓN Y CODIFICACIÓN

  % Calcula los bits disponibles para cada gránulo (incluye el bit_reservoir).
  mean_bits = init_mean_bits + bit_reservoir;
  % Chequea la máxima cantidad de bits permitida para cada gránulo.
  if mean_bits > 4095
    mean_bits = 4095;
  end

  % Si no hay datos de audio, entonces devuelve ciertos valores por
  % defecto. Si se trata del primer gránulo, entonces se procesa el
  % segundo gránulo. Si está procesando el segundo gránulo, entonces
  % la ejecución continúa en la etapa de formato.
  if XR == 0,
    scalefac_scale(gr) = 0;
    scalefac_l(:,gr) = zeros(21,1);
```

```
    IX(:,gr) = zeros(576,1);
    IS(:,gr) = zeros(576,1);
    count1table_select(gr) = 0;
    big_values(gr) = 0;
    region0_count(gr) = 0;
    region1_count(gr) = 0;
    table_select(gr,:) = zeros(1,3);
    preflag = 0;
    global_gain(gr) = 210 - 8; % Es decir, 210-system_const.
    scalefac_compress(gr) = 0;
    part2_3_length(gr) = 0;
    bit_reservoir = mean_bits;
    S = zeros(36,32);
    SHT = zeros(256,6,gr);
    slen1(gr) = 0;
    slen2(gr) = 0;
    rlb(gr,:) = zeros(1,3);
    fe(gr,:) = zeros(1,3);
    ff(gr,:) = zeros(1,3);
    count1(gr) = 0;
  else
    % Inicialización de variables.
    scalefac_scale(gr) = 0;
    scalefac_l(:,gr) = zeros(21,1);

    % Calcula la distorsión permitida, según el modelo psicoacústico.
    xmin = Distorsion_permitida(SFBT,UA,UEG);

    % Ciclo interno. Chequea la tasa de bits. Si el vector IX requiere más
    % bits de los permitidos para ser codificado, repite el ciclo interno;
    % hasta que la cantidad disponible de bits sea suficiente para codificar
    % el espectro cuantizado (vector IX).
    [IX(:,gr),IS(:,gr),SHT(:,:,gr),overall_bitsum,count1table_select(gr),...
        big_values(gr),region0_count(gr),region1_count(gr),table_select(gr,:),...
        qquant,quantanf,rlb(gr,:),count1(gr),fe(gr,:),ff(gr,:)] = ...
      Ciclo_interno(XR,system_const,SFBT);
    while mean_bits - 74 < overall_bitsum,
      system_const = system_const - 1;
      [IX(:,gr),IS(:,gr),SHT(:,:,gr),overall_bitsum,count1table_select(gr),...
          big_values(gr),region0_count(gr),region1_count(gr),table_select(gr,:),...
          qquant,quantanf,rlb(gr,:),count1(gr),fe(gr,:),ff(gr,:)] = ...
        Ciclo_interno(XR,system_const,SFBT);
    end

    % Ciclo externo. Chequea la distorsión. Si la distorsión máxima
    % es excedida, vuelve a llamar el ciclo interno, de acuerdo con
    % los requerimientos del estándar internacional ISO/IEC 11172-3.
    % En este paso se incluyen las condiciones para terminar los ciclos;
    % si alguna de ellas se cumple, la ejecución de los ciclos se detiene
    % y los datos obtenidos hasta ese momento son usados para la etapa de
    % formato.
    [XFSF,preflag] = Ciclo_externo(SFBT,IX(:,gr),qquant,quantanf,...
      scalefac_scale(gr));
    % Si alguna banda excede la distorsión permitida, entonces se chequean
    % las condiciones para terminar los ciclos. En el caso de que las
    % condiciones para terminación de los ciclos no se cumplan, entonces se
    % repite el ciclo interno.
```

```
while length(find(xmin < XFSF)) > 0,
  % Chequea si todas las bandas del factor de escala ya han sido amplificadas,
  % en cuyo caso se terminan los ciclos.
  if length(find(scalefac_l(:,gr) ~= 0)) == 21
    break
  end
  % Chequea el máximo de los factores de escala, teniendo en cuenta
  % el campo scalefac_scale.
  if (max(scalefac_l(1:11,gr))==15 | max(scalefac_l(12:21,gr))==7)...
      & scalefac_scale(gr)== 0
    % scalefac_scale se pone en '1' y scalefac_l se pone en '0'.
    scalefac_l(:,gr) = zeros(21,1);
    scalefac_scale(gr) = 1;
  end
  if (max(scalefac_l(1:11,gr))==15 | max(scalefac_l(12:21,gr))==7)...
      & scalefac_scale(gr)==1
    % Se terminan los ciclos.
    break
  end
  % Si no se cumple ninguna condición para la terminación de los ciclos,
  % entonces se repite el ciclo interno.
  [IX(:,gr),IS(:,gr),SHT(:,:,gr),overall_bitsum,count1table_select(gr),...
      big_values(gr),region0_count(gr),region1_count(gr),table_select(gr,:),...
      qquant,quantanf,rlb(gr,:),count1(gr),fe(gr,:),ff(gr,:)] = ...
    Ciclo_interno(XR,system_const,SFBT);
  while mean_bits - 74 < overall_bitsum,
    system_const = system_const - 1;
    [IX(:,gr),IS(:,gr),SHT(:,:,gr),overall_bitsum,count1table_select(gr),...
        big_values(gr),region0_count(gr),region1_count(gr),...
        table_select(gr,:),qquant,quantanf,rlb(gr,:),count1(gr),...
        fe(gr,:),ff(gr,:)] = Ciclo_interno(XR,system_const,SFBT);
  end
  [XFSF,preflag] = Ciclo_externo(SFBT,IX(:,gr),qquant,quantanf,...
    scalefac_scale(gr));
end

% Calcula la ganancia global del sistema.
% Primero, calcula el vector IX sin realizar el redondeo (especialmente
% usado sólo para este cálculo).
% IXGG: IX para global_gain.
IXGG = (abs(XR)/(2^((qquant+quantanf)/4))) .^ 0.75 - 0.0946;
% Después, calcula la información del intervalo de cuantización (vector iic)
% para cada una de las 576 muestras de audio del gránulo.
iic = zeros(576,1);
ifqstep = 2^(0.5*(1+scalefac_scale(gr)));
for h = SFBT,
  for q = h(3):h(4),
    iic(q) = 4*log2(abs(XR(q))*ifqstep^scalefac_l(h(1),gr)/(abs(IXGG(q))^...
      (4/3)));
  end
end
iic(419:576)=4.*log2(abs(XR(419:576)).*ifqstep./abs(IXGG(419:576)).^(4/3));
% Por último, calcula global_gain.
global_gain(gr) = round(mean(iic))+210-system_const-qquant-ajgg;

% Chequea el valor de global_gain (se debe tener en cuenta que este campo
% se escribe en la información secundaria, y consume 8 bits).
```

```
    if global_gain(gr) < 0
      global_gain(gr) = 0;
    end
    if global_gain(gr) > 255
      global_gain(gr) = 255;
    end

    % Calcula la cantidad de bits necesaria para codificar los
    % factores de escala.
    if max(scalefac_l(1:11,gr)) ~= 0
      slen1(gr) = fix(log2(max(scalefac_l(1:11,gr))))+1;
    else
      slen1(gr) = 0;
    end
    if max(scalefac_l(12:21,gr)) ~= 0
      slen2(gr) = fix(log2(max(scalefac_l(12:21,gr))))+1;
    else
      slen2(gr) = 0;
    end
    scalefac_compress(gr) = SFC(3,4*slen1(gr)+slen2(gr)+1);

    % Corrige los valores de slen1 y slen2, para evitar una escritura
    % incorrecta del flujo de bits, ocasionada por valores de slen1 y
    % slen2 que no fueron incluidos en la tabla de scalefac_compress
    % proporcionada por el estándar ISO/IEC 11172-3.
    if scalefac_compress(gr) == 5
      slen1(gr) = 1;
      slen2(gr) = 1;
    end
    if scalefac_compress(gr) == 8
      slen1(gr) = 2;
      slen2(gr) = 1;
    end
    if scalefac_compress(gr) == 14
      slen1(gr) = 4;
      slen2(gr) = 2;
    end

    % Calcula la cantidad total de bits que usa el espectro cuantizado
    % (vector IX más los factores de escala), y se determina cuántos bits
    % quedan disponibles para el próximo gránulo.
    part2_3_length(gr) = overall_bitsum + slen1(gr)*11 + slen2(gr)*10;
    bit_reservoir = mean_bits - part2_3_length(gr);
  end

end

%%% FORMATO DEL FLUJO DE BITS.

% Para cada trama, determina si es necesario activar padding_bit con el fin de
% ajustar la tasa de bits promedio. Antes, se calcula la cantidad de tramas.
frames = (a+1151)/1152;
if frames > 1
  dif = rem(144000*tasa,Fs);
  rest = rest - dif;
  if rest < 0
    padding_bit = 1;
```

```
      rest = rest + Fs;
    else
      padding_bit = 0;
    end
  end

  % Escribe el encabezado como una cadena binaria de texto (ASCII).
  Encabezado(padding_bit,bitrate_index);

  % Escribe la información secundaria como una cadena binaria de texto (ASCII).
  Info_secundaria(main_data_begin,part2_3_length,big_values,...
    global_gain,scalefac_compress,table_select,region0_count,region1_count,...
    preflag,scalefac_scale,count1table_select);

  % Escribe los códigos de Huffman y los factores de escala como una cadena
  % binaria de texto (ASCII).
  Datos_principales(IX,IS,SHT,slen1,slen2,big_values,table_select,rlb,...
    fe,ff,tabla_0,tabla_1,count1,count1table_select,main_data_begin,padding_bit,mdt);

  % Chequea el valor de bit_reservoir, de tal manera que nunca permite que
  % main_data esté repartido entre más de 2 tramas. Además, redondea el valor
  % de main_data_begin, de tal manera que sea múltiplo de 8 (main_data_begin
  % se especifica en bytes).
  if tasa == 320
    main_data_begin = 0;
    bit_reservoir = 0;
  else
    if bit_reservoir > mbr
      main_data_begin = mmdb;
      bit_reservoir = mbr;
    else
      main_data_begin = fix(bit_reservoir/8);
      bit_reservoir = main_data_begin*8;
    end
  end
end

% Transforma la cadena binaria de texto ASCII (bin_str) en un vector de valores
% binarios (unos y ceros) en formato decimal, y escribe este vector como un archivo
% binario, de acuerdo con los requerimientos del estándar ISO/IEC 11172-3 (usando
% el formato de máquina big-endian).
fid = fopen([archivo '.mp3'],'wb','b');
fwrite(fid,bin_str-48,'ubit1');
fclose(fid);

% Indica que el archivo MP3 ya ha sido creado.
disp([sprintf('\n'),'El archivo ',archivo,'.mp3 ha sido terminado.'])

clear Fs IXGG MAP SIZ PMA X a b bits dif fid gr h iic q rest ifqstep mdt
clear rsize sb v bit_reservoir bitrate_index init_mean_bits mean_bits mbr mmdb ajgg
clear archivo ENT LBC SFBT SFC SHT U576 UA UES tabla_0 tabla_1 C ans bin_str tasa
clear tiempo seg max_rsize ajuste
```

```
function MDCTB = Aliasing(MDCT)
%ALIASING Reduce el aliasing introducido por la MDCT.
%
%   MDCTB=ALIASING(MDCT)
%   Retorna en el vector MDCTB de 576 valores el resultado de la reducción del
%   aliasing introducido por el 50% de solapamiento usado al aplicar la MDCT o
%   transformada discreta del coseno modificada.
%
%   El vector MDCT contiene los 576 valores de la transformación obtenida a
%   través de TRANSF_DISCRETA_COSENO.
%
%   Ver también TRANSF_DISCRETA_COSENO.

MDCTB = [];

% Constantes para el cálculo mariposa, proporcionadas por el estándar
% ISO/IEC 11172-3 (vectores c, cs y ca).
% c = [-0.6; -0.535; -0.33; -0.185; -0.095; -0.041; -0.0142; -0.0037];
% cs = 1 ./ sqrt(1+c .^ 2);
% ca = c ./ sqrt(1+c .^ 2);
cs = [0.85749292571254; 0.88174199731771; 0.94962864910273; 0.98331459249179;
  0.99551781606759; 0.99916055817815; 0.99989919524445; 0.99999315507028];
ca = [-0.51449575542753; -0.47173196856497; -0.31337745420390; -0.18191319961098;
  -0.09457419252642; -0.04096558288530; -0.01419856857247; -0.00369997467376];

% Reducción del aliasing o solapamiento a través de 8 cálculos mariposa. La
% reducción se hace en grupos de 18 líneas de frecuencia (a las 16 primeras se
% les hace reducción y a las 2 últimas no), empezando en la muestra 11.
MDCTcs = zeros(8,2);
MDCTca = zeros(8,2);
for i = 11:18:551,

  % Las líneas de frecuencia se multiplican por cs y ca.
  MDCTcs = [[MDCT(i+7:-1:i) .* cs] [MDCT(i+8:i+15) .* cs]];
  MDCTca = [[MDCT(i+7:-1:i) .* ca] [MDCT(i+8:i+15) .* ca]];

  % El concepto mariposa involucra sumar y restar las líneas de frecuencia en
  % pares, entregando dos nuevas líneas de frecuencia por cada par. En total, se
  % reemplazan 16 líneas de frecuencia (vector MDCTBparcial).
  MDCTBparcial = [MDCTcs(8:-1:1,1)+MDCTca(8:-1:1,2); MDCTcs(:,2)-MDCTca(:,1)];

  % Concatenación del resultado anterior con los dos valores de MDCT a los que
  % no se les aplica la reducción del aliasing.
  MDCTB = [MDCTB; MDCTBparcial; MDCT(i+16:i+17)];

end

% Los primeros 10 y los últimos 8 valores de MDCT (que no son incluidos en el
% cálculo mariposa) se conservan para MDCTB.
MDCTB = [MDCT(1:10); MDCTB; MDCT(569:576)];
```

```
function F = Analisis_fft(ENT)
%ANALISIS_FFT Análisis Transformada Rápida de Fourier.
%
%   F=ANALISIS_FFT(ENT)
%   Calcula el espectro auditivo usando la Transformada rápida de Fourier.
%   El espectro (vector F) se expresa en decibeles, y sólo tiene 512 valores
%   debido a la simetría de la FFT. La cantidad de puntos de la transformada
%   es 1024 (Capa II). La variable de entrada (vector ENT) equivale a
%   1152 muestras de audio PCM. Con el fin de usar la FFT de 1024 puntos para
%   las 1152 muestras, se escogen las 1024 muestras centrales de las 1152.
%
%   A las 1024 muestras centrales se les aplica una ventana de Hanning convencional
%   antes de calcular su FFT, para suavizar los extremos del intervalo de señal.

% Escoge las 1024 muestras centrales del vector ENT y las almacena en el vector s.
s = ENT(65:1088);

% Sólo se realizan los cálculos si el vector s no es un vector de ceros. De lo
% contrario, el espectro auditivo (vector F) es un vector de -INF.
if s ~= 0
   % Calcula la ventana de Hanning de 1024 puntos (vector h).
   h = sqrt(8/3) * hanning(1024);

   % Obtiene la densidad espectral de potencia (vector F) a través de la FFT.
   % F se reduce a la mitad de componentes, por la simetría de la FFT.
   F = max(20*log10(abs(fft(s .* h))/1024),-200);
   F = F(1:512);
else
   F = zeros(512,1)-200; % -200 dB corresponde a -INF.
end
```

```
function [XFSF,preflag] = Ciclo_externo(SFBT,IX,qquant,quantanf,scalefac_scale)
%CICLO_EXTERNO Realiza el ciclo para el control de la distorsión.
%
%
[XFSF,PREFLAG]=CICLO_EXTERNO(SFBT,IX,QQUANT,QUANTANF,SCALEFAC_SCALE)
%   XFSF es el vector de la distorsión en cada una de las bandas del factor de
%   escala. PREFLAG indica si la opción de preénfasis ha sido utilizada o no.
%
%   SFBT es la matriz de las bandas del factor de escala, obtenida en WAV2MP3.
%   IX es el vector de valores espectrales cuantizados, obtenido con CICLO_INTERNO.
%   QQUANT y QUANTANF son los valores usados para el intervalo de cuantización,
%   obtenidos con CICLO_INTERNO. SCALEFAC_SCALE es el factor logarítmico de
%   cuantización para los factores de escala, obtenido en WAV2MP3.
%
%   Ver también WAV2MP3, CICLO_INTERNO.

global XR gr xmin scalefac_l

% La opción de preénfasis no se implementa.
preflag = 0;

% Calcula la distorsión en las bandas del factor de escala (vector XFSF), y luego
% amplifica las bandas del factor de escala que exceden el umbral de enmascaramiento.
ifqstep = 2^(0.5*(1+scalefac_scale));
XFSF = zeros(1,21);
for sb = SFBT,
  for i = sb(3):sb(4),
    XFSF(sb(1)) = XFSF(sb(1))+(abs(XR(i))-IX(i)^(4/3)*2^((qquant+quantanf)/4))^2/sb(2);
  end
  if xmin(sb(1)) < XFSF(sb(1))
    scalefac_l(sb(1),gr) = scalefac_l(sb(1),gr) + 1;
    xmin(sb(1)) = xmin(sb(1))*ifqstep^(2*scalefac_l(sb(1),gr));
    for i = sb(3):sb(4),
      XR(i) = XR(i)*ifqstep^scalefac_l(sb(1),gr);
    end
  end
end
```

```
function [IX,IS,SHT,overall_bitsum,count1table_select,big_values,region0_count, ...
    region1_count,table_select,qquant,quantanf,rlb,count1,fe,ff] = ...
  Ciclo_interno(XR,system_const,SFBT)
%CICLO_INTERNO Realiza el ciclo para el control de la tasa de bits.
%
%
[IX,IS,SHT,OVERALL_BITSUM,COUNT1TABLE_SELECT,BIG_VALUES,REGION0_COUNT,...
%     REGION1_COUNT,TABLE_SELECT,QQUANT,QUANTANF,RLB,COUNT1,FE,FF]=...
%       CICLO_INTERNO(XR,SYSTEM_CONST,SFBT)
%
%   IX es el vector de valores espectrales cuantizados (sin tener en cuenta el
%   signo de XR). IS es el vector de valores espectrales cuantizados (teniendo
%   en cuenta el signo de XR). SHT es la matriz que contiene las 3 tablas de
%   Huffman seleccionadas para codificar las 3 subregiones de la región
%   big_values. OVERALL_BITSUM representa la cantidad de bits que se usan para
%   codificar los valores cuantizados. COUNT1TABLE_SELECT indica la tabla de
%   Huffman escogida para codificar los cuádruplos de valores pertenecientes a
%   la región count1. BIG_VALUES es la cantidad de valores espectrales, contados
%   por pares, ubicados en las bajas frecuencias. REGION0_COUNT es la cantidad de
%   bandas del factor de escala (disminuidas en 1) incluidas en region0.
%   REGION1_COUNT es la cantidad de bandas del factor de escala (disminuidas
%   en 1) incluidas en region1. TABLE_SELECT es el vector que incluye el número
%   de las tablas de Huffman usadas para codificar region0, region1 y region2.
%   QQUANT y QUANTANF son los valores usados para el intervalo de cuantización.
%   El vector RLB indica cuántos bits deben usarse para codificar cada uno de los
%   valores adicionales en cada una de las 3 subregiones en que se divide big_values
%   (el valor adicional es necesario si el valor máximo de la subregión es mayor
%   que 15). COUNT1 es la cantidad de cuádruplos incluidos en la región count1
%   (región intermedia del vector IX). El vector FE indica el principio de cada
%   una de las 3 subregiones en que se divide big_values. El vector FF indica el
%   final de cada una de las 3 subregiones en que se divide big_values.
%
%   XR es el vector de muestras subbanda obtenido originalmente con ALIASING, y
%   más tarde modificado en CICLO_EXTERNO. SYSTEM_CONST es la constante del
sistema,
%   obtenida en WAV2MP3. SFBT es la matriz que proporciona las bandas del factor de
%   escala, obtenida en WAV2MP3.
%
%   Ver también WAV2MP3, ALIASING, CICLO_EXTERNO, HUFFMAN.

% Inicialización de variables.
sfm = exp(sum(log(XR.^2))/576)/(sum(XR.^2)/576); % Calcula la planura espectral.
quantanf = system_const*log(sfm); % Selección del intervalo de cuantización.
qquant = 0; % Selección del intervalo de cuantización.

% Cuantiza los valores del vector XR. En IX se almacenan los valores
% absolutos y en IS se almacenan los valores signados.
IX = round((abs(XR)/(2^((qquant+quantanf)/4))) .^ 0.75 - 0.0946);
IS = sign(XR).*IX;

% Cuando todos los valores del vector IX son cero, se termina el Ciclo Interno.
if IX == 0
   count1table_select = 0;
   big_values = 0;
   region0_count = 0;
   region1_count = 0;
   table_select = zeros(1,3);
```

```
  overall_bitsum = 0;
  SHT = zeros(256,6);
  rlb = zeros(1,3);
  count1 = 0;
  fe = ones(1,3);
  ff = ones(1,3);
  return
else
  % Limita a (8191 + 15) el valor máximo de los componentes del vector IX.
  while max(IX) > 8206,
    qquant = qquant+1;
    IX = round((abs(XR)/(2^((qquant+quantanf)/4))) .^ 0.75 - 0.0946);
    IS = sign(XR).*IX;
  end

  % Calcula el número de pares de ceros (rzero) en el extremo superior
  % del vector IX.
  rzero = 0;
  for i = 576:-1:1,
    if IX(i) == 0
      rzero = rzero + 0.5;
    else
      break
    end
  end
  rzero = fix(rzero);

  % Calcula el número de cuádruplos de valores (count1) en el vector IX cuyo valor
  % absoluto no es mayor que uno, y que siguen luego de los valores rzero (en la
  % parte intermedia del vector IX).
  count1 = 0;
  for i = 576-rzero*2:-1:1,
    if IX(i) == 0 | IX(i) == 1
      count1 = count1+0.25;
    else
      break
    end
  end
  count1 = fix(count1);

  % Calcula el número de pares de valores (big_values) en el extremo inferior
  % del vector IX.
  big_values = (576-rzero*2-count1*4)/2;

  % Almacena en bitsum_count1 el número de bits necesario para codificar los
  % valores count1. Además, determina la tabla de Huffman más apropiada para
  % dicha codificación (count1table_select).
  count1table_0 = [1 4 4 5 4 6 5 6 4 5 5 6 5 6 6 6];
  % Calcula los bits que se consumen con la tabla B.
  bitsum_table1 = count1*4;
  % Calcula los bits que se consumen con la tabla A.
  bitsum_table0 = 0;
  for k = big_values*2+1:4:big_values*2+count1*4,
    bitsum_table0 = bitsum_table0+count1table_0(8*IX(k)+4*IX(k+1)+2*IX(k+2)+IX(k+3)+1);
  end
  % Calcula cuántos bits de signo se usan para la región count1.
  count1_signbits = length(find(IX(big_values*2+1:576-rzero*2)));
```

```
% Calcula la cantidad total de bits que consume la región count1,
% usando la tabla de Huffman que usa menos bits para codificar los
% cuádruplos count1.
bitsum_count1 = min(bitsum_table0,bitsum_table1) + count1_signbits;
if bitsum_table0 < bitsum_table1
  % Escoge la tabla A, proporcionada por el estándar ISO 11172-3.
  count1table_select = 0;
else
  % Escoge la tabla B, proporcionada por el estándar ISO 11172-3.
  count1table_select = 1;
end

% Divide la región de los valores big_values en 3 subregiones, cuyos límites
% deben coincidir con los límites de las bandas del factor de escala incluidas
% en el tamaño de dicha región. Los valores de region0_count y region1_count
% obtenidos aquí, no están regidos por los requerimientos del estándar ISO 11172-3.
if big_values == 0
  bvscfb = 0;
else
  for i = SFBT,
    if big_values*2 <= i(4)
      break
    end
  end
  bvscfb = i(1);
end
region0_count = fix(bvscfb/2);
region2_count = fix(bvscfb/4);
region1_count = bvscfb - region0_count - region2_count;

% Encuentra el máximo valor cuantizado de cada subregión (max_region), el
% cual sirve para calcular linbits (vector rlb). Estos dos valores permiten
% seleccionar las tablas de Huffman apropiadas (matrices HCTN y HCT) para
% codificar cada una de las subregiones de big_values. En este paso, se puede
% dar la posibilidad de hasta tres (3) tablas de Huffman por subregión, para un
% caso máximo de nueve (9) tablas seleccionadas para big_values. Posteriormente,
% se realiza la selección de la tabla más apropiada para cada subregión.
% Adicionalmente, se inicializa apropiadamente la variable bitsum_table, de
% acuerdo con la cantidad de subregiones en que se divide big_values.
rlb = zeros(1,3);
HCTN = zeros(3);
HCT = zeros(256,18);
if bvscfb > 3
  % Caso 1. La región big_values se ha dividido en tres (3) subregiones.
  %       Máximo nueve (9), mínimo tres (3) tablas para big_values.
  fe = [1,SFBT(4,region0_count)+1,SFBT(4,region0_count+region1_count)+1];
  ff = [SFBT(4,region0_count),SFBT(4,region0_count+region1_count),big_values*2];
  max_region0 = max(IX(1:ff(1)));
  max_region1 = max(IX(fe(2):ff(2)));
  max_region2 = max(IX(fe(3):ff(3)));
  if max_region0 > 15
    rlb(1) = ceil(log2(max_region0 - 14)); % Cálculo de linbits, region0.
  end
  if max_region1 > 15
    rlb(2) = ceil(log2(max_region1 - 14)); % Cálculo de linbits, region1.
  end
  if max_region2 > 15
```

```
    rlb(3) = ceil(log2(max_region2 - 14)); % Cálculo de linbits, region2.
  end
  % Selección de la(s) tabla(s) de Huffman por subregión.
  [HCTN(1,:),HCT(:,1:6)] = Huffman(rlb(1),max_region0);
  [HCTN(2,:),HCT(:,7:12)] = Huffman(rlb(2),max_region1);
  [HCTN(3,:),HCT(:,13:18)] = Huffman(rlb(3),max_region2);
  bitsum_table = ones(3)*inf;
elseif bvscfb == 1 | bvscfb == 2
  % Caso 2. La región big_values se ha dividido en una (1) subregión.
  %         Máximo tres (3), mínimo una (1) tabla para big_values.
  fe = [1, 1, 1];
  ff = [1, big_values*2, 1];
  max_region1 = max(IX(1:ff(2)));
  if max_region1 > 15
    rlb(2) = ceil(log2(max_region1 - 14)); % Cálculo de linbits, region1.
  end
  % Selección de la(s) tabla(s) de Huffman por subregión.
  [HCTN(2,:),HCT(:,7:12)] = Huffman(rlb(2),max_region1);
  bitsum_table = [zeros(1,3); ones(1,3)*inf; zeros(1,3)];
elseif bvscfb == 3
  % Caso 3. La región big_values se ha dividido en dos (2) subregiones.
  %         Máximo seis (6), mínimo dos (2) tablas para big_values.
  fe = [1, 5, 1];
  ff = [4, big_values*2, 1];
  max_region0 = max(IX(1:4));
  max_region1 = max(IX(5:ff(2)));
  if max_region0 > 15
    rlb(1) = ceil(log2(max_region0 - 14)); % Cálculo de linbits, region0.
  end
  if max_region1 > 15
    rlb(2) = ceil(log2(max_region1 - 14)); % Cálculo de linbits, region1.
  end
  % Seleccion de la(s) tabla(s) de Huffman por subregión.
  [HCTN(1,:),HCT(:,1:6)] = Huffman(rlb(1),max_region0);
  [HCTN(2,:),HCT(:,7:12)] = Huffman(rlb(2),max_region1);
  bitsum_table = [ones(2,3)*inf; zeros(1,3)];
elseif bvscfb == 0
  % Caso 4. La región big_values no existe, y por lo tanto, no hay subdivisión.
  fe = [1, 1, 1];
  ff = [1, 1, 1];
  bitsum_table = zeros(3);
end

% Ajusta los valores de region0_count y region1_count, de acuerdo con los
% requerimientos del estándar ISO 11172-3.
if region0_count ~= 0
  region0_count = region0_count - 1;
end
if region1_count ~= 0
  region1_count = region1_count - 1;
end

% A partir de las tablas de Huffman seleccionadas anteriormente (mínimo 1,
% máximo 3 tablas para cada subregión), escoge aquella que usa menos bits
% en la codificación (table_select y la matriz SHT), y calcula los bits
% necesarios para codificar los valores en cada subregión. Por último,
% calcula overall_bitsum, que es la cantidad total de bits que se usan
```

```
  % para codificar los valores cuantizados. Primero, se cargan los valores
  % máximos de las tablas de Huffman.
  maxth = [1 2 3 3 0 4 4 6 6 6 8 8 8 ones(1,19)*16];
  % Después, se realiza el conteo de bits con todas las tablas de Huffman, y
  % almacena el resultado en bitsum_table.
  for j = 1:3,
    for t = 1:3,
      if HCTN(j,t)
        bs = 0;
        for k = fe(j):2:ff(j),
          bs = bs + HCT(maxth(HCTN(j,t)+1)*min(15,IX(k))+ ...
            min(15,IX(k+1))+1,2*t+6*j-7);
        end
        bitsum_table(j,t) = bs;
      end
    end
    % Selecciona la tabla de Huffman que consume menos bits.
    [min_bitsum,min_bitsum_index] = min(bitsum_table(j,:));
    % Guarda el número de la tabla, en el vector table_select.
    table_select(j) = HCTN(j,min_bitsum_index);
    % Almacena en bitsum_region, la cantidad de bits que consume la subregión.
    bitsum_region(j) = min_bitsum;
    % Calcula la cantidad de bits que consumen los valores mayores que 14.
    rlbsum = length(find(IX(fe(j):ff(j)) > 14))*rlb(j);
    region_linbits_sum(j) = rlbsum;
    % Almacena en la matriz SHT, la tabla de Huffman seleccionada.
    SHT(:,2*j-1:2*j) = HCT(:,2*min_bitsum_index+6*j-7:2*min_bitsum_index+6*j-6);
  end
  % Calcula cuántos bits de signo se usan para la región big_values.
  big_values_signbits = length(find(IX(1:big_values*2)));

  % Calcula la cantidad de bits que consume el espectro cuantizado (vector IX).
  overall_bitsum = sum(bitsum_region) + sum(region_linbits_sum) + ...
    big_values_signbits + bitsum_count1;
end
```

```
function [BS,LT,LNT] = Componentes_tonales(F,UA,MAP,LBC)
%COMPONENTES_TONALES Encuentra las componentes tonales y no-tonales de la señal
%   de audio.
%
%   [BS,LT,LNT]=COMPONENTES_TONALES(F,UA,MAP,LBC)
%   La matriz LT lista las componentes tonales y la matriz LNT lista las
%   componentes no-tonales; ambas matrices se componen de dos columnas, la
%   primera proporciona el índice de la línea de frecuencia y la segunda, el
%   nivel de presión sonora de esa línea. El vector BS proporciona las banderas
%   para cada una de las 512 líneas de frecuencia, tal que:
%   - Componente no examinada = 0.
%   - Componente tonal = 1.
%   - Componente no-tonal = 2.
%   - Componente irrelevante = 3.
%
%   F es el vector de densidad espectral de potencia normalizada, obtenido con
%   ANALISIS_FFT. UA es la matriz de umbral absoluto y MAP es el vector de mapeo
%   entre las líneas de frecuencia y su índice para la matriz UA; ambos obtenidos
%   con UMBRAL_ABSOLUTO. LBC es la matriz que contiene los límites de las bandas
%   críticas, obtenida con LIMITES_BANDA_CRITICA.
%
%   Ver también ANALISIS_FFT, UMBRAL_ABSOLUTO, LIMITES_BANDA_CRITICA.

% Inicializa el vector de banderas BS para las 512 líneas de frecuencia.
BS = zeros(512,1);

% Determina la lista de los máximos locales (matriz LML) para las líneas de
% frecuencia. La primera columna de LML son los índices, y la segunda corresponde
% al nivel de presión sonora. El análisis sólo es necesario hacerlo para las
% líneas de frecuencia con índice entre 3 y 500, de acuerdo con los requerimientos
% del estándar ISO 11172-3, página 112 (mirar los intervalos de k usados para
% generar los intervalos de J, más adelante en este mismo archivo; éstos obligan a
% la exclusión de índices k menores que 2 y mayores que 501).
LML = [];
c = 1;
for k = 3:500,
  if (F(k)>F(k-1) & F(k)>=F(k+1))
    LML(c,1) = k;
    LML(c,2) = F(k);
    c = c+1;
  end
end

% Determina cuáles de los máximos locales son componentes tonales y calcula su
% nivel de presión sonora (columnas de la matriz LT). Además, asigna para cada
% una de las líneas de frecuencia analizadas su respectivo valor de bandera en
% el vector BS.
LT = [];
c = 1;
if not(isempty(LML))
  for i = 1:length(LML(:,1)),
    k = LML(i,1);
    tonal = 1;

    % Determina el intervalo J que indica cuántas frecuencias adyacentes
    % deben ser examinadas, de acuerdo con la posición de la línea de
    % frecuencia, o sea, de acuerdo con el índice k.
```

```
    if (2<k & k<63)
      J = [-2,2];
    elseif (63<=k & k<127)
      J = [-3,-2,2,3];
    elseif (127<=k & k<255)
      J = [-6:-2,2:6];
    elseif (255<=k & k<=500)
      J = [-12:-2,2:12];
    else
      tonal = 0;
    end

    % Examina las frecuencias adyacentes de acuerdo con el intervalo definido
    % por J, y determina si el máximo local es definitivamente una componente
    % tonal.
    for j = J,
      tonal = tonal & (F(k)-F(k+j) >= 7);
    end

    % Si F(k) es realmente una componente tonal, entonces lo siguiente se lista:
    %  - Índice k de la línea de frecuencia.
    %  - Nivel de presión sonora.
    %  - Bandera tonal.
    if tonal
      LT(c,1) = k;
      LT(c,2) = 10*log10(10^(F(k-1)/10)+10^(F(k)/10)+10^(F(k+1)/10));
      BS(k) = 1; % Bandera de componente tonal = 1.
      for j = [J,-1,1],
        BS(k+j) = 3; % Bandera de componente irrelevante = 3.
      end
      c = c+1;
    end
  end
end

% Dentro de cada banda crítica, analiza las restantes líneas de frecuencia y suma
% sus potencias para determinar el nivel de presión sonora de cada componente
% no-tonal (columnas de la matriz LNT). Además, asigna para cada una de estas
% líneas de frecuencia su respectivo valor de bandera en el vector BS.
LNT = [];
for i = 1:26,

  % Para cada banda crítica, calcula la potencia en las componentes no-tonales.
  POT = -200;
  PESO = 0; % Usado para calcular la media geométrica de la banda crítica.
  for k = UA(LBC(i,1),1):UA(LBC(i+1,1),1)-1,
    if (BS(k) == 0) % Bandera de componente no examinada = 0.
      POT = 10*log10(10^(POT/10)+10^(F(k)/10));
      PESO = PESO+10^(F(k)/10)*(UA(MAP(k),2)-i);
      BS(k) = 3; % Bandera de componente irrelevante = 3.
    end
  end

  % El índice de la componente no-tonal es el más cercano a la media geométrica
  % de la banda crítica.
  if (POT <= -200)
    IND = round(mean(UA(LBC(i,1),1)+UA(LBC(i+1,1),1)));
```

```
    else
      IND = UA(LBC(i,1),1)+round(PESO/10^(POT/10)*...
        (UA(LBC(i+1,1),1)-UA(LBC(i,1),1)));
    end

    % Chequea que el índice de la componente no-tonal esté dentro de los valores
    % permitidos, o sea, entre 1 y 512.
    if (IND<1)
      IND = 1;
    end
    if (IND>512)
      IND = 512;
    end

    % El índice de la componente no-tonal no puede coincidir con el índice
    % de alguna componente tonal.
    if (BS(IND) == 1)
      IND = IND+1;
    end

    % Dentro de cada banda crítica, para su componente no-tonal se lista lo
    % siguiente:
    %  - Índice de la línea de frecuencia.
    %  - Nivel de presión sonora.
    %  - Bandera no-tonal.
    LNT(i,1) = IND;
    LNT(i,2) = POT;
    BS(IND) = 2;

  end
```

```
function Datos_principales(IX,IS,SHT,slen1,slen2,big_values,table_select,rlb,...
  fe,ff,tabla_0,tabla_1,count1,count1table_select,main_data_begin,padding_bit,mdt)
%DATOS_PRINCIPALES Escritura de los datos principales.
%
%
DATOS_PRINCIPALES(IX,IS,SHT,SLEN1,SLEN2,BIG_VALUES,TABLE_SELECT,RLB,FE,FF,...
%
TABLA_O,TABLA_1,COUNT1,COUNT1TABLE_SELECT,MAIN_DATA_BEGIN,PADDING_BIT,
MDT)
%
%   Realiza la escritura de los datos principales.
%
%   IX es el vector de valores espectrales cuantizados (sin tener en cuenta
%   el signo de XR), obtenido con CICLO  INTERNO. IS es el vector de valores
%   espectrales cuantizados (teniendo en cuenta el signo de XR), obtenido
%   con CICLO_INTERNO. SHT es la matriz que contiene las 3 tablas de Huffman
%   seleccionadas para codificar las 3 subregiones de big_values, obtenida
%   con CICLO_INTERNO. SLEN1 es la cantidad de bits que consumen los factores
%   de escala para las primeras 11 bandas del factor de escala, y slen2 es la
%   cantidad de bits que consumen los factores de escala para las últimas
%   10 bandas del factor de escala, ambos obtenidos en WAV2MP3. BIG_VALUES es
%   la cantidad de valores espectrales, contados por pares, ubicados en las
%   bajas frecuencias, obtenido con CICLO_INTERNO. TABLE_SELECT es el vector
%   que incluye el número de las tablas de Huffman usadas para codificar region0,
%   region1 y region2, obtenido con CICLO_INTERNO. RLB indica la cantidad de
%   bits usados para codificar el valor adicional en cada una de las 3 subregiones
%   en que se divide big_values (el valor adicional es necesario si el valor
%   máximo de la subregión es mayor que 14), obtenida con CICLO_INTERNO. El
%   vector FE indica el principio de cada una de las 3 subregiones en que se
%   divide big_values; y el vector FF indica el final de cada una de las
%   3 subregiones en que se divide big_values, ambos obtenidos con CICLO_INTERNO.
%   TABLA_0 es la tabla A; y TABLA_1 es la tabla B, usadas para codificar los
%   cuádruplos count1, y ambas obtenidas en WAV2MP3. COUNT1 es la cantidad de
%   cuádruplos incluidos en la región count1 (región intermedia del vector IX),
%   obtenida con CICLO_INTERNO. COUNT1TABLE_SELECT indica la tabla de Huffman
%   escogida (tabla A o tabla B) para codificar los cuádruplos de valores
%   pertenecientes a la región count1, obtenida con CICLO_INTERNO.
%   MAIN_DATA_BEGIN es el puntero que indica el comienzo de la información de
%   audio (main_data) de cada trama, obtenido en WAV2MP3. PADDING_BIT es la
%   bandera que indica si la trama usa padding_bit, obtenida en WAV2MP3. MDT es
%   la cantidad máxima de bits para main_data, por trama, obtenida en WAV2MP3.
%
%   Ver también WAV2MP3, CICLO INTERNO.

global scalefac_l bin_str

% Carga los valores máximos de las 30 tablas de Huffman usadas para
% big_values, e inicializa la cadena binaria de audio.
maxth = [1 2 3 3 0 4 4 6 6 6 8 8 8 ones(1,19)*16];
aud_str = '';

for gr = 1:2,

   % Escribe los factores de escala en la cadena binaria de audio.
   if slen1(gr) ~= 0
      for sfb = 1:11,
         aud_str = [aud_str dec2bin(scalefac_l(sfb,gr),slen1(gr))];
```

```
    end
  end
  if slen2(gr) ~= 0
    for sfb = 12:21,
      aud_str = [aud_str dec2bin(scalefac_l(sfb,gr),slen2(gr))];
    end
  end

  % Escribe los códigos de Huffman para los pares big_values en la cadena
  % binaria de audio.
  for region = 1:3,
    if table_select(gr,region) ~= 0
      if table_select(gr,region) > 15

        %Se usan las tablas de Huffman que incluyen linbits (linbits>0).
        for i = fe(gr,region):2:ff(gr,region),
          x = IX(i,gr);
          y = IX(i+1,gr);
          if x > 14
            linbitsx = x-15;
            x = 15;
          end
          if y > 14
            linbitsy = y-15;
            y = 15;
          end

          % El campo hcod([x][y]) se extrae de la tabla SHT, con ayuda del
          % campo table_select, y se escribe en la cadena binaria de audio
          % empezando por el bit más a la izquierda; la cantidad de bits es
          % hlen([x][y]).
          aud_str = [aud_str dec2bin(SHT(maxth(table_select(gr,region)+1)*...
              x+y+1,region*2,gr),SHT(maxth(table_select...
              (gr,region)+1)*x+y+1,region*2-1,gr))];

          % Escribe linbitsx en la cadena binaria de audio; la cantidad
          % de bits es linbits (vector rlb).
          if x > 14
            aud_str = [aud_str dec2bin(linbitsx,rlb(gr,region))];
          end

          % Escribe signx en la cadena binaria de audio. Se escribe '0'
          % si es positivo, y '1' si es negativo. La cantidad de bits es 1.
          if IS(i,gr) > 0
            aud_str = [aud_str dec2bin(0,1)];
          elseif IS(i,gr) < 0
            aud_str = [aud_str dec2bin(1,1)];
          end

          % Escribe linbitsy en la cadena binaria de audio; la cantidad
          % de bits es linbits (vector rlb).
          if y > 14
            aud_str = [aud_str dec2bin(linbitsy,rlb(gr,region))];
          end

          % Escribe signy en la cadena binaria de audio. Se escribe '0' si es
          % positivo, y '1' si es negativo. La cantidad de bits es 1.
```

```
        if IS(i+1,gr) > 0
          aud_str = [aud_str dec2bin(0,1)];
        elseif IS(i+1,gr) < 0
          aud_str = [aud_str dec2bin(1,1)];
        end
      end
    else
      % Se usan las tablas de Huffman que no incluyen linbits (linbits=0).
      for i = fe(gr,region):2:ff(gr,region),
        x = IX(i,gr);
        y = IX(i+1,gr);

        % El campo hcod([x][y]) se extrae de la tabla SHT, con ayuda del
        % campo table_select, y se escribe en la cadena binaria de audio
        % empezando por el bit más a la izquierda, la cantidad de bits es
        % hlen([x][y]).
        aud_str = [aud_str dec2bin(SHT(maxth(table_select(gr,region)+1)*...
            x+y+1,region*2,gr),SHT(maxth(table_select...
            (gr,region)+1)*x+y+1,region*2-1,gr))];

        % Escribe signx en la cadena binaria de audio. Se escribe '0'
        % si es positivo, y '1' si es negativo. La cantidad de bits es 1.
        if IS(i,gr) > 0
          aud_str = [aud_str dec2bin(0,1)];
        elseif IS(i,gr) < 0
          aud_str = [aud_str dec2bin(1,1)];
        end

        % Escribe signy en la cadena binaria de audio. Se escribe '0'
        % si es positivo, y '1' si es negativo. La cantidad de bits es 1.
        if IS(i+1,gr) > 0
          aud_str = [aud_str dec2bin(0,1)];
        elseif IS(i+1,gr) < 0
          aud_str = [aud_str dec2bin(1,1)];
        end
      end

    end
  end
end

% Escribe los códigos de Huffman para los cuádruplos count1 en la
% cadena binaria de audio.
for k = big_values(gr)*2+1:4:big_values(gr)*2+count1(gr)*4,
  if count1table_select(gr) == 0
    % Escribe hcod([v][w][x][y]) en la cadena binaria de audio, usando los
    % datos de tabla_0 (tabla A), empezando por el bit más a la izquierda,
    % la cantidad de bits es hlen([v][w][x][y]).
    aud_str = [aud_str dec2bin(tabla_0(2,8*IX(k,gr)+4*IX(k+1,gr)+2*IX(k+2,gr)+...
        IX(k+3,gr)+1),tabla_0(1,8*IX(k,gr)+4*IX(k+1,gr)+2*IX(k+2,gr)+...
        IX(k+3,gr)+1))];
  else
    % Escribe hcod([v][w][x][y]) en la cadena binaria de audio, usando
    % los datos de tabla_1 (tabla B), empezando por el bit más a la
    % izquierda, la cantidad de bits es 4.
    aud_str = [aud_str dec2bin(tabla_1(2,8*IX(k,gr)+4*IX(k+1,gr)+2*IX(k+2,gr)+...
        IX(k+3,gr)+1),4)];
```

```
      end

      % Se escribe los bits de signo en la cadena binaria de audio. Se escribe
      % '0' si es positivo, y '1' si es negativo. La cantidad de bits es 1.
      if IS(k,gr) > 0
        aud_str = [aud_str dec2bin(0,1)];
      elseif IS(k,gr) < 0
        aud_str = [aud_str dec2bin(1,1)];
      end
      if IS(k+1,gr) > 0
        aud_str = [aud_str dec2bin(0,1)];
      elseif IS(k+1,gr) < 0
        aud_str = [aud_str dec2bin(1,1)];
      end
      if IS(k+2,gr) > 0
        aud_str = [aud_str dec2bin(0,1)];
      elseif IS(k+2,gr) < 0
        aud_str = [aud_str dec2bin(1,1)];
      end
      if IS(k+3,gr) > 0
        aud_str = [aud_str dec2bin(0,1)];
      elseif IS(k+3,gr) < 0
        aud_str = [aud_str dec2bin(1,1)];
      end

    end
  end

  % Escribe un byte adicional en la cadena binaria de audio, para ajustar la tasa
  % de bits promedio.
  if padding_bit == 1
    aud_str = [aud_str dec2bin(0,8)];
  end

  % Calcula las longitudes de las cadenas binarias (aud_str y bin_str).
  largo = length(aud_str);
  LBS = length(bin_str);
  mdbx8 = main_data_begin*8;

  % Une la cadena binaria de audio (aud_str) con la cadena binaria (bin_str),
  % de acuerdo con los valores de main_data_begin y bit_reservoir, y teniendo en
  % cuenta las longitudes de ambas cadenas.
  if main_data_begin == 0
    % Todos los datos de audio (vector aud_str) se escriben en la trama actual.
    % Si aud_str no consume todos los bits disponibles para el audio, entonces
    % una parte de la trama actual se deja como bit_reservoir para la siguiente
    % trama.
    bin_str = [bin_str aud_str dec2bin(0,mdt+padding_bit*8-largo)];
  else
    if mdbx8 <= largo
      % Los datos de audio (vector aud_str) se escriben entre dos tramas: la
      % trama actual y la anterior. En el caso de que aud_str no consuma todos
      % los bits que habían disponibles, entonces una parte de la trama actual
      % se deja como bit_reservoir para la siguiente trama.
      bin_str(LBS-168-mdbx8+1:LBS-168) = aud_str(1:mdbx8);
      bin_str = [bin_str aud_str(mdbx8+1:largo) dec2bin(0,mdt+padding_bit*8-largo+mdbx8)];
    else
```

```
    % Todos los datos de audio (vector aud_str) se escriben en la trama anterior.
    % Si aud_str consume menos bits de los que había en la trama anterior como
    % bit_reservoir, entonces ese fragmento de la trama anterior que no se usó
    % y toda la trama actual se dejan como bit_reservoir para la siguiente trama.
    % NOTA: Debido a que para esta implementación los datos principales se reparten
    % apenas entre 2 tramas, entonces el fragmento de la trama anterior no será
    % sobreescrito y quedará como bits de relleno (stuffing_bits).
    bin_str(LBS-168-mdbx8+1:LBS-168-mdbx8+largo) = aud_str;
    bin_str = [bin_str dec2bin(0,mdt+padding_bit*8)];
  end
end
```

```
function xmin = Distorsion_permitida(SFBT,UA,UEG)
%DISTORSION_PERMITIDA Calcula la distorsión permitida, según el modelo psicoacústico.
%
%   XMIN=DISTORSION_PERMITIDA(SFBT,UA,UEG)
%   XMIN es el vector que contiene las distorsiones permitidas, en cada
%   banda del factor de escala, según las salidas del modelo psicoacústico.
%
%   SFBT es la tabla de bandas del factor de escala, obtenida en WAV2MP3. UA es
%   la matriz de umbral absoluto obtenida con UMBRAL_ABSOLUTO. El vector UEG es
%   el umbral de enmascaramiento global obtenido con UMBRAL_ENMASC_GLOBAL.
%
%   Ver también WAV2MP3, UMBRAL_ENMASC_GLOBAL, UMBRAL_ABSOLUTO.

% Almacena en el vector UEGMAP el resultado del mapeo entre 512 líneas de frecuencia
% y los 130 valores del vector UEG. Primero, se mapea el valor inicial.
UEGMAP(1) = UEG(1);

% Después, se mapean los valores finales.
UEGMAP(UA(130,1):512) = UEG(130);

% Por último, se mapean todos los demás valores.
for i = 2:129,
  UEGMAP(UA(i,1):UA(i+1,1)) = UEG(i);
end

% Encuentra el mínimo umbral de enmascaramiento (vector MINTHR) para cada una
% de las 21 bandas del factor de escala.
for n = 1:21,
  MINTHR(n) = min(UEGMAP(SFBT(3,n):SFBT(4,n)));
end

% Ajuste del umbral mínimo para la Capa III. Con este cambio, se notan leves
% mejoras en el sonido de algunos archivos.
MINTHR(1:7) = MINTHR(1:7)+12;
MINTHR(8:14) = MINTHR(8:14)+5;

% Calcula la distorsión permitida (vector xmin) para las 21 bandas del factor
% de escala.
xmin = 10.^(MINTHR/20)./SFBT(2,:);
```

```
function Encabezado(padding_bit,bitrate_index)
%ENCABEZADO Escritura del encabezado.
%
%   ENCABEZADO(PADDING_BIT,BITRATE_INDEX)
%   Realiza la escritura del encabezado para cada trama del archivo MP3.
%
%   PADDING_BIT es la bandera que indica la escritura de un byte de relleno
%   dentro del flujo de bits para ajustar la tasa de transferencia promedio,
%   obtenida en WAV2MP3. BITRATE_INDEX es un valor que indica la tasa de bits
%   del archivo, obtenido en WAV2MP3.
%
%   Ver también WAV2MP3.

global bin_str

% Inicialización de variables.
syncword = 4095; % 12 bits en '1', para sincronización de la trama.
ID = 1; % Indica que es audio MPEG.
layer = 1;  % El esquema usado es la Capa III.
protection_bit = 1; % No se incluye CRC.
sampling_frequency = 0; % La frecuencia de muestreo es 44100 Hz.
private_bit = 0; % No se usan private_bits dentro del flujo de bits.
mode = 3; % Modo monofónico.
mode_extension = 0; % No se usa extensión del modo.
copyright = 0; % El archivo MP3 no tiene copyright.
original_or_copy = 0; % El archivo MP3 es una copia.
emphasis = 0; % No se usa ningún tipo de preénfasis.

% Escribe el encabezado en la cadena binaria bin_str.
bin_str = [bin_str dec2bin(syncword,12)];
bin_str = [bin_str dec2bin(ID,1)];
bin_str = [bin_str dec2bin(layer,2)];
bin_str = [bin_str dec2bin(protection_bit,1)];
bin_str = [bin_str dec2bin(bitrate_index,4)];
bin_str = [bin_str dec2bin(sampling_frequency,2)];
bin_str = [bin_str dec2bin(padding_bit,1)];
bin_str = [bin_str dec2bin(private_bit,1)];
bin_str = [bin_str dec2bin(mode,2)];
bin_str = [bin_str dec2bin(mode_extension,2)];
bin_str = [bin_str dec2bin(copyright,1)];
bin_str = [bin_str dec2bin(original_or_copy,1)];
bin_str = [bin_str dec2bin(emphasis,2)];
```

```
function FDE = Factores_escala(S)
%FACTORES_ESCALA Calcula los factores de escala
%
%   FDE=FACTORES_ESCALA(S)
%   Obtiene los factores de escala (matriz FDE) para cada una de las subbandas en
%   la capa II. S es la matriz de 36x32 muestras subbanda obtenidas con
%   FILTRO_SUBBANDA.
%
%   Se determina el máximo valor absoluto para grupos de 12 muestras subbanda y
%   se compara con los valores de la tabla de factores de escala; eligiendo su
%   valor más cercano por exceso como el factor de escala asociado a cada grupo.
%
%   Ver también FILTRO_SUBBANDA

% Tabla de Factores de Escala proporcionada por el estándar ISO 11172-3.
TFE = [
   2.00000000000000; 1.58740105196820; 1.25992104989487; 1.00000000000000;
   0.79370052598410; 0.62996052494744; 0.50000000000000; 0.39685026299205;
   0.31498026247372; 0.25000000000000; 0.19842513149602; 0.15749013123686;
   0.12500000000000; 0.09921256574801; 0.07874506561843; 0.06250000000000;
   0.04960628287401; 0.03937253280921; 0.03125000000000; 0.02480314143700;
   0.01968626640461; 0.01562500000000; 0.01240157071850; 0.00984313320230;
   0.00781250000000; 0.00620078535925; 0.00492156660115; 0.00390625000000;
   0.00310039267963; 0.00246078330058; 0.00195312500000; 0.00155019633981;
   0.00123039165029; 0.00097656250000; 0.00077509816991; 0.00061519582514;
   0.00048828125000; 0.00038754908495; 0.00030759791257; 0.00024414062500;
   0.00019377454248; 0.00015379895629; 0.00012207031250; 0.00009688727124;
   0.00007689947814; 0.00006103515625; 0.00004844363562; 0.00003844973907;
   0.00003051757813; 0.00002422181781; 0.00001922486954; 0.00001525878906;
   0.00001211090890; 0.00000961243477; 0.00000762939453; 0.00000605545445;
   0.00000480621738; 0.00000381469727; 0.00000302772723; 0.00000240310869;
   0.00000190734863; 0.00000151386361; 0.00000120155435; 1E-20
];

FDE = [];

for a = 1:12:36,
   for i = 1:32,

      % Determina el máximo valor absoluto de 3 grupos (índice a) de 12 muestras
      % subbanda para cada una de las 32 subbandas (índice i) y lo almacena en
      % MAXS.
      MAXS = max(abs(S(a:a+11,i)));

      % Realiza un barrido de la tabla de factores de escala hasta encontrar en
      % ella un valor mayor que el valor de MAXS.
      j = 0;
      while (j<64 & MAXS>TFE(64-j))
            j = j+1;
      end

      % Crea el vector MAXT, que es compuesto de cada factor de escala elegido de
      % la tabla (para cada i) en el paso anterior.
      MAXT(i) = TFE(64-j);

    end
```

```
    % Almacena en la matriz FDE los valores que va tomando MAXT (para cada a);
    % resultando una matriz de 3x32, o sea, 3 factores de escala elegidos para
    % cada subbanda.
    FDE = [FDE; MAXT];
end
```

```
function S = Filtro_subbanda(MAE,C)
%FILTRO_SUBBANDA  Filtro subbanda para el análisis.
%
%   S=FILTRO_SUBBANDA(MAE,C)
%   Obtiene 32 muestras subbanda (vector S), a partir de 32
%   muestras de audio de entrada PCM (vector MAE).
%   El vector C, almacenado en el archivo Ci.mat, contiene los
%   coeficientes de la ventana del análisis.

global X

% Búfer FIFO (vector X) de 512 elementos con desplazamientos de a 32 elementos.
X(512:-1:33) = X(480:-1:1);

% Introduce 32 muestras de audio (vector MAE) al búfer FIFO, la primera muestra
% en la posición 32 y la última muestra en la posición 1.
X(32:-1:1) = MAE;

% Multiplica al vector X por los coeficientes de la ventana del análisis (vector C),
% para obtener el vector Z.
Z = C .* X;

% Cálculo parcial para obtener el vector Y de 64 coeficientes.
Y = zeros(1,64);
for i = 1:64,
   Y(i) = sum(Z(i:64:512));
end

% Calcula las 32 muestras subbanda de salida (vector S).
% Haciendo la analogía con el estándar ISO 11172-3, la variable Mik (no calculada
% explícitamente en este código), sería Mik = cos((2*i-1).*fcos).
S = zeros(1,32);
fcos = (-16:47)*pi/64;
for i = 1:32,
   S(i) = sum(cos((2*i-1).*fcos)*Y(:));
end
```

```
function [HCTN,HCT] = Huffman(region_linbits,max_region)
%HUFFMAN Tablas del código de Huffman.
%
%   [HCTN,HCT]=HUFFMAN(REGION_LINBITS,MAX_REGION)
%   Selecciona las posibles tablas del código de Huffman para codificar
%   una subregión de los valores big_values. El vector HCTN contiene los
%   números de las tablas de Huffman seleccionadas para la subregión. La
%   matriz HCT contiene las tablas de Huffman para codificar la subregión.
%
%   REGION_LINBITS es la cantidad de bits necesarios para codificar
%   los valores mayores que 14 pertenecientes a la subregión. MAX_REGION
%   es el mayor valor cuantizado de la subregión. Ambas variables son
%   obtenidas en CICLO_INTERNO.
%
%   Ver también CICLO_INTERNO.

% Valores del código binario de Huffman, pero expresados en formato
% decimal. Los valores se encuentran en el estándar ISO 11172-3.
hcod_t1 = [1; 1; 1; 0];
hcod_t2 = [1; 2; 1; 3; 1; 1; 3; 2; 0];
hcod_t3 = [3; 2; 1; 1; 1; 1; 3; 2; 0];
hcod_t5 = [1; 2; 6; 5; 3; 1; 4; 4; 7; 5; 7; 1; 6; 1; 1; 0];
hcod_t6 = [7; 3; 5; 1; 6; 2; 3; 2; 5; 4; 4; 1; 3; 3; 2; 0];
hcod_t7 = [1; 2; 10; 19; 16; 10; 3; 3; 7; 10; 5; 3; 11; 4; 13; 17; 8; 4; 12; 11; 18;
           15; 11; 2; 7; 6; 9; 14; 3; 1; 6; 4; 5; 3; 2; 0];
hcod_t8 = [3; 4; 6; 18; 12; 5; 5; 1; 2; 16; 9; 3; 7; 3; 5; 14; 7; 3; 19; 17; 15; 13;
        10; 4; 13; 5; 8; 11; 5; 1; 12; 4; 4; 1; 1; 0];
hcod_t9 = [7; 5; 9; 14; 15; 7; 6; 4; 5; 5; 6; 7; 7; 6; 8; 8; 8; 5; 15; 6; 9; 10; 5;
        1; 11; 7; 9; 6; 4; 1; 14; 4; 6; 2; 6; 0];
hcod_t10 = [1; 2; 10; 23; 35; 30; 12; 17; 3; 3; 8; 12; 18; 21; 12; 7; 11; 9; 15; 21;
            32; 40; 19; 6; 14; 13; 22; 34; 46; 23; 18; 7; 20; 19; 33; 47; 27; 22; 9;
         3; 31; 22; 41; 26; 21; 20; 5; 3; 14; 13; 10; 11; 16; 6; 5; 1; 9; 8; 7; 8;
         4; 4; 2; 0];
hcod_t11 = [3; 4; 10; 24; 34; 33; 21; 15; 5; 3; 4; 10; 32; 17; 11; 10; 11; 7; 13; 18;
            30; 31; 20; 5; 25; 11; 19; 59; 27; 18; 12; 5; 35; 33; 31; 58; 30; 16; 7; 5;
         28; 26; 32; 19; 17; 15; 8; 14; 14; 12; 9; 13; 14; 9; 4; 1; 11; 4; 6; 6; 6;
         3; 2; 0];
hcod_t12 = [9; 6; 16; 33; 41; 39; 38; 26; 7; 5; 6; 9; 23; 16; 26; 11; 17; 7; 11; 14; 21;
            30; 10; 7; 17; 10; 15; 12; 18; 28; 14; 5; 32; 13; 22; 19; 18; 16; 9; 5; 40;
         17; 31; 29; 17; 13; 4; 2; 27; 12; 11; 15; 10; 7; 4; 1; 27; 12; 8; 12 ; 6; 3;
         1; 0];
hcod_t13 = [1; 5; 14; 21; 34; 51; 46; 71; 42; 52; 68; 52; 67; 44; 43; 19; 3; 4; 12; 19;
            31; 26; 44; 33; 31; 24; 32; 24; 31; 35; 22; 14; 15; 13; 23; 36; 59; 49; 77;
            65; 29; 40; 30; 40; 27; 33; 42; 16; 22; 20; 37; 61; 56; 79; 73; 64; 43; 76;
            56; 37; 26; 31; 25; 14; 35; 16; 60; 57; 97; 75; 114; 91; 54; 73; 55; 41; 48;
            53; 23; 24; 58; 27; 50; 96; 76; 70; 93; 84; 77; 58; 79; 29; 74; 49; 41; 17;
            47; 45; 78; 74; 115; 94; 90; 79; 69; 83; 71; 50; 59; 38; 36; 15; 72; 34; 56;
            95; 92; 85; 91; 90; 86; 73; 77; 65; 51; 44; 43; 42; 43; 20; 30; 44; 55; 78;
            72; 87; 78; 61; 46; 54; 37; 30; 20; 16; 53; 25; 41; 37; 44; 59; 54; 81; 66;
            76; 57; 54; 37; 18; 39; 11; 35; 33; 31; 57; 42; 82; 72; 80; 47; 58; 55; 21;
              22; 26; 38; 22; 53; 25; 23; 38; 70; 60; 51; 36; 55; 26; 34; 23; 27; 14; 9; 7;
            34; 32; 28; 39; 49; 75; 30; 52; 48; 40; 52; 28; 18; 17; 9; 5; 45;  21; 34; 64;
            56; 50; 49; 45; 31; 19; 12; 15; 10; 7; 6; 3; 48; 23; 20; 39; 36; 35; 53; 21;
              16; 23; 13; 10; 6; 1; 4; 2; 16; 15; 17; 27; 25; 20; 29; 11; 17; 12; 16; 8; 1;
            1; 0; 1];
hcod_t15 = [7; 12; 18; 53; 47; 76; 124; 108; 89; 123; 108; 119; 107; 81; 122; 63; 13; 5;
         16; 27; 46; 36; 61; 51; 42; 70; 52; 83; 65; 41; 59; 36; 19; 17; 15; 24; 41;
```

```
                34; 59; 48; 40; 64; 50; 78; 62; 80; 56; 33; 29; 28; 25; 43; 39; 63; 55; 93;
                76; 59; 93; 72; 54; 75; 50; 29; 52; 22; 42; 40; 67; 57; 95; 79; 72; 57; 89;
                69; 49; 66; 46; 27; 77; 37; 35; 66; 58; 52; 91; 74; 62; 48; 79; 63; 90; 62;
                40; 38; 125; 32; 60; 56; 50; 92; 78; 65; 55; 87; 71; 51; 73; 51; 70; 30; 109;
                53; 49; 94; 88; 75; 66; 122; 91; 73; 56; 42; 64; 44; 21; 25; 90; 43; 41; 77;
                73; 63; 56; 92; 77; 66; 47; 67; 48; 53; 36; 20; 71; 34; 67; 60; 58; 49; 88;
                76; 67; 106; 71; 54; 38; 39; 23; 15; 109; 53; 51; 47; 90; 82; 58; 57; 48; 72;
                57; 41; 23; 27; 62; 9; 86; 42; 40; 37; 70; 64; 52; 43; 70; 55; 42; 25; 29; 18;
                11; 11; 118; 68; 30; 55; 50; 46; 74; 65; 49; 39; 24; 16; 22; 13; 14; 7; 91;
                44; 39; 38; 34; 63; 52; 45; 31; 52; 28; 19; 14; 8; 9; 3; 123; 60; 58; 53; 47;
                43; 32; 22; 37; 24; 17; 12; 15; 10; 2; 1; 71; 37; 34; 30; 28; 20; 17; 26; 21;
                16; 10; 6; 8; 6; 2; 0];
hcod_t16 = [1; 5; 14; 44; 74; 63; 110; 93; 172; 149; 138; 242; 225; 195; 376; 17; 3; 4;
                12; 20; 35; 62; 53; 47; 83; 75; 68; 119; 201; 107; 207; 9; 15; 13; 23; 38;
                67; 58; 103; 90; 161; 72; 127; 117; 110; 209; 206; 16; 45; 21; 39; 69; 64;
                114; 99; 87; 158; 140; 252; 212; 199; 387; 365; 26; 75; 36; 68; 65; 115; 101;
                179; 164; 155; 264; 246; 226; 395; 382; 362; 9; 66; 30; 59; 56; 102; 185; 173;
                265; 142; 253; 232; 400; 388; 378; 445; 16; 111; 54; 52; 100; 184; 178; 160;
                133; 257; 244; 228; 217; 385; 366; 715; 10; 98; 48; 91; 88; 165; 157; 148;
                261; 248; 407; 397; 372; 380; 889; 884; 8; 85; 84; 81; 159; 156; 143; 260;
                249; 427; 401; 392; 383; 727; 713; 708; 7; 154; 76; 73; 141; 131; 256; 245;
                426; 406; 394; 384; 735; 359; 710; 352; 11; 139; 129; 67; 125; 247; 233; 229;
                219; 393; 743; 737; 720; 885; 882; 439; 4; 243; 120; 118; 115; 227; 223; 396;
                746; 742; 736; 721; 712; 706; 223; 436; 6; 202; 224; 222; 218; 216; 389; 386;
                381; 364; 888; 443; 707; 440; 437; 1728; 4; 747; 211; 210; 208; 370; 379; 734;
                723; 714; 1735; 883; 877; 876; 3459; 865; 2; 377; 369; 102; 187; 726; 722;
                358; 711; 709; 866; 1734; 871; 3458; 870; 434; 0; 12; 10; 7; 11; 10; 17; 11;
                9; 13; 12; 10; 7; 5; 3; 1; 3];
hcod_t24 = [15; 13; 46; 80; 146; 262; 248; 434; 426; 669; 653; 649; 621; 517; 1032; 88;
                14; 12; 21; 38; 71; 130; 122; 216; 209; 198; 327; 345; 319; 297; 279; 42; 47;
                22; 41; 74; 68; 128; 120; 221; 207; 194; 182; 340; 315; 295; 541; 18; 81; 39;
                75; 70; 134; 125; 116; 220; 204; 190; 178; 325; 311; 293; 271; 16; 147; 72;
                69; 135; 127; 118; 112; 210; 200; 188; 352; 323; 306; 285; 540; 14; 263; 66;
                129; 126; 119; 114; 214; 202; 192; 180; 341; 317; 301; 281; 262; 12; 249; 123;
                121; 117; 113; 215; 206; 195; 185; 347; 330; 308; 291; 272; 520; 10; 435; 115;
                111; 109; 211; 203; 196; 187; 353; 332; 313; 298; 283; 531; 381; 17; 427; 212;
                208; 205; 201; 193; 186; 177; 169; 320; 303; 286; 268; 514; 377; 16; 335; 199;
                197; 191; 189; 181; 174; 333; 321; 305; 289; 275; 521; 379; 371; 11; 668; 184;
                183; 179; 175; 344; 331; 314; 304; 290; 277; 530; 383; 373; 366; 10; 652; 346;
                171; 168; 164; 318; 309; 299; 287; 276; 263; 513; 375; 368; 362; 6; 648; 322;
                316; 312; 307; 302; 292; 284; 269; 261; 512; 376; 370; 364; 359; 4; 620; 300;
                296; 294; 288; 282; 273; 266; 515; 380; 374; 369; 365; 361; 357; 2; 1033; 280;
                278; 274; 267; 264; 259; 382; 378; 372; 367; 363; 360; 358; 356; 0; 43; 20;
                19; 17; 15; 13; 11; 9; 7; 6; 4; 7; 5; 3; 1; 3];

% Cantidad de bits (longitud) usados por el código de Huffman. Los valores se
% encuentran en el estándar ISO 11172-3.
hlen_t1 = [1; 3; 2; 3];
hlen_t2 = [1; 3; 6; 3; 3; 5; 5; 5; 6];
hlen_t3 = [2; 2; 6; 3; 2; 5; 5; 5; 6];
hlen_t5 = [1; 3; 6; 7; 3; 3; 6; 7; 6; 6; 7; 8; 7; 6; 7; 8];
hlen_t6 = [3; 3; 5; 7; 3; 2; 4; 5; 4; 4; 5; 6; 6; 5; 6; 7];
hlen_t7 = [1; 3; 6; 8; 8; 9; 3; 4; 6; 7; 7; 8; 6; 5; 7; 8; 8; 9; 7; 7; 8; 9; 9; 9;
           7; 7; 8; 9; 9; 10; 8; 8; 9; 10; 10; 10];
hlen_t8 = [2; 3; 6; 8; 8; 9; 3; 2; 4; 8; 8; 8; 6; 4; 6; 8; 8; 9; 8; 8; 8; 9; 9; 10;
           8; 7; 8; 9; 10; 10; 9; 8; 9; 9; 11; 11];
hlen_t9 = [3; 3; 5; 6; 8; 9; 3; 3; 4; 5; 6; 8; 4; 4; 5;  6;7; 8; 6; 5; 6; 7; 7; 8;
```

```
                7; 6; 7; 7; 8; 9; 8; 7; 8; 8; 9; 9];
hlen_t10 = [1; 3; 6; 8; 9; 9; 9; 10; 3; 4; 6; 7; 8; 9; 8; 8; 6; 6; 7; 8; 9; 10; 9;
                    9; 7; 7; 8; 9; 10; 10; 9; 10; 8; 8; 9; 10; 10; 10; 10; 10; 9; 9; 10; 10;
                    11; 11; 10; 11; 8; 8; 9; 10; 10; 10; 11; 11; 9; 8; 9; 10; 10; 11; 11; 11];
hlen_t11 = [2; 3; 5; 7; 8; 9; 8; 9; 3; 3; 4; 6; 8; 8; 7; 8; 5; 5; 6; 7; 8; 9; 8; 8;
                    7; 6; 7; 9; 8; 10; 8; 9; 8; 8; 8; 9; 9; 10; 9; 10; 8; 8; 9; 10; 10; 11;
                    10; 11; 8; 7; 7; 8; 9; 10; 10; 10; 8; 7; 8; 9; 10; 10; 10; 10];
hlen_t12 = [4; 3; 5; 7; 8; 9; 9; 9; 3; 3; 4; 5; 7; 7; 8; 8; 5; 4; 5; 6; 7; 8; 7; 8;
                    6; 5; 6; 6; 7; 8; 8; 8; 7; 6; 7; 7; 8; 8; 8; 9; 8; 7; 8; 8; 8; 9; 8; 9;
                    8; 7; 7; 8; 8; 9; 9; 10; 9; 8; 8; 9; 9; 9; 9; 10];
hlen_t13 = [1; 4; 6; 7; 8; 9; 9; 10; 9; 10; 11; 11; 12; 12; 13; 13; 3; 4; 6; 7; 8; 8;
                    9; 9; 9; 9; 10; 10; 11; 12; 12; 12; 6; 6; 7; 8; 9; 9; 10; 10; 9; 10; 10;
                    11; 11; 12; 13; 13; 7; 7; 8; 9; 9; 10; 10; 10; 10; 11; 11; 11; 11; 12; 13;
                    13; 8; 7; 9; 9; 10; 10; 11; 11; 10; 11; 11; 12; 12; 13; 13; 14; 9; 8; 9;
                    10; 10; 10; 11; 11; 11; 11; 12; 11; 13; 13; 14; 14; 9; 9; 10; 10; 11; 11;
                    11; 11; 11; 12; 12; 12; 13; 13; 14; 14; 10; 9; 10; 11; 11; 11; 12; 12; 12;
                    12; 13; 13; 13; 14; 16; 16; 9; 8; 9; 10; 10; 11; 11; 12; 12; 12; 12; 13;
                    13; 14; 15; 15; 10; 9; 10; 10; 11; 11; 11; 13; 12; 13; 13; 14; 14; 14; 16;
                    15; 10; 10; 10; 11; 11; 12; 12; 13; 12; 13; 14; 13; 14; 15; 16; 17; 11; 10;
                    10; 11; 12; 12; 12; 12; 13; 13; 13; 14; 15; 15; 15; 16; 11; 11; 11; 12; 12;
                    13; 12; 13; 14; 14; 15; 15; 15; 16; 16; 16; 12; 11; 12; 13; 13; 13; 14; 14;
                    14; 14; 14; 15; 16; 15; 16; 16; 13; 12; 12; 13; 13; 13; 15; 14; 14; 17; 15;
                    15; 15; 17; 16; 16; 12; 12; 13; 14; 14; 14; 15; 14; 15; 15; 16; 16; 19; 18;
                    19; 16];
hlen_t15 = [3; 4; 5; 7; 7; 8; 9; 9; 9; 10; 10; 11; 11; 11; 12; 13; 4; 3; 5; 6; 7; 7; 8;
                    8; 8; 9; 9; 10; 10; 10; 11; 11; 5; 5; 5; 6; 7; 7; 8; 8; 8; 9; 9; 10; 10; 11;
                    11; 11; 6; 6; 6; 7; 7; 8; 8; 9; 9; 9; 10; 10; 10; 11; 11; 11; 7; 6; 7; 7; 8;
                    8; 9; 9; 9; 9; 10; 10; 10; 11; 11; 11; 8; 7; 7; 8; 8; 8; 9; 9; 9; 9; 10; 10;
                    11; 11; 11; 12; 9; 7; 8; 8; 8; 9; 9; 9; 9; 10; 10; 10; 11; 11; 12; 12; 9; 8;
                    8; 9; 9; 9; 9; 10; 10; 10; 10; 10; 11; 11; 11; 12; 9; 8; 8; 9; 9; 9; 9; 10;
                    10; 10; 10; 11; 11; 12; 12; 12; 9; 8; 9; 9; 9; 9; 10; 10; 10; 11; 11; 11;
                    11; 12; 12; 12; 10; 9; 9; 9; 10; 10; 10; 10; 10; 11; 11; 11; 11; 12; 13; 12;
                    10; 9; 9; 9; 10; 10; 10; 10; 11; 11; 11; 11; 12; 12; 12; 13; 11; 10; 9; 10;
                    10; 10; 11; 11; 11; 11; 11; 11; 12; 12; 13; 13; 11; 10; 10; 10; 10; 11; 11;
                    11; 11; 12; 12; 12; 12; 12; 13; 13; 12; 11; 11; 11; 11; 11; 11; 11; 12; 12;
                    12; 12; 13; 13; 12; 13; 12; 11; 11; 11; 11; 11; 11; 12; 12; 12; 12; 12; 13;
                    13; 13; 13];
hlen_t16 = [1; 4; 6; 8; 9; 9; 10; 10; 11; 11; 11; 12; 12; 12; 13; 9; 3; 4; 6; 7; 8; 9; 9;
                    9; 10; 10; 10; 11; 12; 11; 12; 8; 6; 6; 7; 8; 9; 9; 10; 10; 11; 10; 11; 11;
                    11; 12; 12; 9; 8; 7; 8; 9; 9; 10; 10; 10; 11; 11; 12; 12; 12; 13; 13; 10; 9;
                    8; 9; 9; 10; 10; 11; 11; 11; 12; 12; 12; 13; 13; 13; 9; 9; 8; 9; 9; 10; 11;
                    11; 12; 11; 12; 12; 13; 13; 13; 14; 10; 10; 9; 9; 10; 11; 11; 11; 11; 12; 12;
                    12; 12; 13; 13; 14; 10; 10; 9; 10; 10; 11; 11; 11; 12; 12; 13; 13; 13; 13;
                    15; 15; 10; 10; 10; 10; 11; 11; 11; 12; 12; 13; 13; 13; 13; 14; 14; 14; 10;
                    11; 10; 10; 11; 11; 12; 12; 13; 13; 13; 13; 14; 13; 14; 13; 11; 11; 11; 10;
                    11; 12; 12; 12; 12; 13; 14; 14; 14; 15; 15; 14; 10; 12; 11; 11; 11; 12; 12;
                    13; 14; 14; 14; 14; 14; 14; 13; 14; 11; 12; 12; 12; 12; 12; 13; 13; 13; 13;
                    15; 14; 14; 14; 14; 16; 11; 14; 12; 12; 12; 13; 13; 14; 14; 14; 16; 15; 15;
                    15; 17; 15; 11; 13; 13; 11; 12; 14; 14; 13; 14; 14; 15; 16; 15; 17; 15; 14;
                    11; 9; 8; 8; 9; 9; 10; 10; 10; 11; 11; 11; 11; 11; 11; 11; 8];
hlen_t24 = [4; 4; 6; 7; 8; 9; 9; 10; 10; 11; 11; 11; 11; 11; 12; 9; 4; 4; 5; 6; 7; 8; 8;
                    9; 9; 9; 10; 10; 10; 10; 10; 8; 6; 5; 6; 7; 7; 8; 8; 9; 9; 9; 9; 10; 10; 10;
                    11; 7; 7; 6; 7; 7; 8; 8; 8; 9; 9; 9; 9; 10; 10; 10; 10; 7; 8; 7; 7; 8; 8; 8;
                    8; 9; 9; 9; 10; 10; 10; 10; 11; 7; 9; 7; 8; 8; 8; 8; 9; 9; 9; 9; 10; 10; 10;
                    10; 10; 7; 9; 8; 8; 8; 8; 9; 9; 9; 9; 10; 10; 10; 10; 10; 11; 7; 10; 8; 8;
                    8; 9; 9; 9; 9; 10; 10; 10; 10; 10; 11; 11; 8; 10; 9; 9; 9; 9; 9; 9; 9; 10;
                    10; 10; 10; 11; 11; 8; 10; 9; 9; 9; 9; 9; 9; 10; 10; 10; 10; 10; 11; 11; 11;
```

```
                8; 11; 9; 9; 9; 9; 10; 10; 10; 10; 10; 10; 11; 11; 11; 11; 8; 11; 10; 9; 9;
                9; 10; 10; 10; 10; 10; 10; 11; 11; 11; 11; 8; 11; 10; 10; 10; 10; 10; 10; 10;
                10; 10; 11; 11; 11; 11; 11; 8; 11; 10; 10; 10; 10; 10; 10; 10; 11; 11; 11;
                11; 11; 11; 11; 8; 12; 10; 10; 10; 10; 10; 10; 11; 11; 11; 11; 11; 11; 11;
                11; 8; 8; 7; 7; 7; 7; 7; 7; 7; 7; 7; 7; 8; 8; 8; 8; 4];

% Selecciona los números de las tablas de Huffman (vector HCTN) apropiadas para
% codificar una subregión, teniendo en cuenta el valor de linbits y el máximo
% valor de la subregión.
HCTN = zeros(1,3);
switch region_linbits
case 1
  HCTN(1) = 16;
case 2
  HCTN(1) = 17;
case 3
  HCTN(1) = 18;
case 4
  HCTN(1) = 19;
  HCTN(2) = 24;
case 5
  HCTN(1) = 25;
case 6
  HCTN(1) = 20;
  HCTN(2) = 26;
case 7
  HCTN(1) = 27;
case 8
  HCTN(1) = 21;
  HCTN(2) = 28;
case 9
  HCTN(1) = 29;
case 10
  HCTN(1) = 22;
case 11
  HCTN(1) = 30;
case {12,13}
  HCTN(1) = 23;
  HCTN(2) = 31;
otherwise
  switch max_region
  case 0
    HCTN(1) = 0;
  case 1
    HCTN(1) = 1;
  case 2
    HCTN(1) = 2;
    HCTN(2) = 3;
  case 3
    HCTN(1) = 5;
    HCTN(2) = 6;
  case {4,5}
    HCTN(1) = 7;
    HCTN(2) = 8;
    HCTN(3) = 9;
  case {6,7}
    HCTN(1) = 10;
```

```
    HCTN(2) = 11;
    HCTN(3) = 12;
  otherwise
    HCTN(1) = 13;
    HCTN(2) = 15;
  end
end

% Selecciona las tablas de Huffman (vector HCT) apropiadas para codificar una
% subregión, teniendo en cuenta el valor de linbits y el máximo valor de la
% subregión.
HCT = zeros(256,6);
switch region_linbits
case {1,2,3,10}
  HCT(:,1:2) = [hlen_t16,hcod_t16];
case {4,6,8,12,13}
  HCT(:,1:2) = [hlen_t16,hcod_t16];
  HCT(:,3:4) = [hlen_t24,hcod_t24];
case {5,7,9,11}
  HCT(:,1:2) = [hlen_t24,hcod_t24];
otherwise
  switch max_region
  case 0
    HCT(:,1) = 0;
  case 1
    HCT(:,1:2) = [[hlen_t1; zeros(252,1)][hcod_t1; zeros(252,1)]];
  case 2
    HCT(:,1:2) = [[hlen_t2; zeros(247,1)][hcod_t2; zeros(247,1)]];
        HCT(:,3:4) = [[hlen_t3; zeros(247,1)][hcod_t3; zeros(247,1)]];
  case 3
    HCT(:,1:2) = [[hlen_t5; zeros(240,1)][hcod_t5; zeros(240,1)]];
        HCT(:,3:4) = [[hlen_t6; zeros(240,1)][hcod_t6; zeros(240,1)]];
  case {4,5}
    HCT(:,1:2) = [[hlen_t7; zeros(220,1)][hcod_t7; zeros(220,1)]];
    HCT(:,3:4) = [[hlen_t8; zeros(220,1)][hcod_t8; zeros(220,1)]];
        HCT(:,5:6) = [[hlen_t9; zeros(220,1)][hcod_t9; zeros(220,1)]];
  case {6,7}
    HCT(:,1:2) = [[hlen_t10; zeros(192,1)][hcod_t10; zeros(192,1)]];
        HCT(:,3:4) = [[hlen_t11; zeros(192,1)][hcod_t11; zeros(192,1)]];
        HCT(:,5:6) = [[hlen_t12; zeros(192,1)][hcod_t12; zeros(192,1)]];
  otherwise
    HCT(:,1:2) = [hlen_t13,hcod_t13];
    HCT(:,3:4) = [hlen_t15,hcod_t15];
  end
end
```

```
function Info_secundaria(main_data_begin,part2_3_length,big_values,...
  global_gain,scalefac_compress,table_select,region0_count,region1_count,...
  preflag,scalefac_scale,count1table_select)

%!INFO_SECUNDARIA Escritura de la información secundaria.
%
%   INFO_SECUNDARIA(MAIN_DATA_BEGIN,PART2_3_LENGTH,BIG_VALUES,...
%
GLOBAL_GAIN,SCALEFAC_COMPRESS,TABLE_SELECT,REGION0_COUNT,REGION1_
COUNT,...
%      PREFLAG,SCALEFAC_SCALE,COUNT1TABLE_SELECT)
%
%   Realiza la escritura de la información secundaria para cada trama del MP3.
%
%   MAIN_DATA_BEGIN es el puntero que indica el comienzo de la información de
%   audio (main_data) de cada trama, obtenido en WAV2MP3. PART2_3_LENGTH es la
%   cantidad de bits usados para codificar los valores cuantizados, obtenido
%   en WAV2MP3. BIG_VALUES es la cantidad de valores espectrales, contados por
%   pares, ubicados en las bajas frecuencias, obtenido con CICLO_INTERNO.
%   GLOBAL_GAIN es la información del intervalo de cuantización, obtenida en
%   WAV2MP3. SCALEFAC_COMPRESS es un índice a una tabla que proporciona la
%   cantidad de bits que consume la codificación de los factores de escala,
%   obtenido en WAV2MP3. TABLE_SELECT es el vector que incluye el número de las
%   tablas de Huffman usadas para codificar region0, region1 y region2, obtenido
%   con CICLO_INTERNO. REGION0_COUNT es la cantidad de bandas del factor de
%   escala (disminuidas en 1) incluidas en region0, obtenida con CICLO_INTERNO.
%   REGION1_COUNT es la cantidad de bandas del factor de escala (disminuidas en 1)
%   incluidas en region1, obtenida con CICLO_INTERNO. PREFLAG es la bandera que
%   indica el uso de la opción de preénfasis, obtenida con CICLO_EXTERNO.
%   SCALEFAC_SCALE es el escalamiento para los factores de escala, obtenido en
%   WAV2MP3. COUNT1TABLE_SELECT indica la tabla de Huffman escogida para
codificar
%   los cuádruplos de valores pertenecientes a la región count1, obtenida con
%   CICLO_INTERNO.
%
%   Ver también WAV2MP3, CICLO INTERNO, CICLO EXTERNO.

global bin_str

% Inicialización de variables.
private_bits = 0; % 5 bits que no se usan. Se ponen en '0'.
scfsi = 0; % Los factores de escala se transmiten para cada gránulo.
window_switching_flag = 0; % Sólo se usan bloques largos.

% Escritura de la información secundaria en la cadena binaria.
bin_str = [bin_str dec2bin(main_data_begin,9)];
bin_str = [bin_str dec2bin(private_bits,5)];
bin_str = [bin_str dec2bin(scfsi,4)];
for gr = 1:2,
   bin_str = [bin_str dec2bin(part2_3_length(gr),12)];
   bin_str = [bin_str dec2bin(big_values(gr),9)];
   bin_str = [bin_str dec2bin(global_gain(gr),8)];
   bin_str = [bin_str dec2bin(scalefac_compress(gr),4)];
   bin_str = [bin_str dec2bin(window_switching_flag,1)];
   for region = 1:3,
      bin_str = [bin_str dec2bin(table_select(gr,region),5)];
   end
```

```
    bin_str = [bin_str dec2bin(region0_count(gr),4)];
    bin_str = [bin_str dec2bin(region1_count(gr),3)];
    bin_str = [bin_str dec2bin(preflag,1)];
    bin_str = [bin_str dec2bin(scalefac_scale(gr),1)];
    bin_str = [bin_str dec2bin(count1table_select(gr),1)];
end
```

```
function LBC = Limites_banda_critica
%LIMITES_BANDA_CRITICA Límites de las Bandas Críticas.
%
%   LBC=LIMITES_BANDA_CRITICA
%   Proporciona, en la matriz LBC, los valores de la tabla "LÍMITES DE LAS BANDAS
%   CRÍTICAS" tal y como está dada en el estándar ISO 11172-3; para la Capa II,
%   y una frecuencia de muestreo de 44.1 kHz.
%
%   Ver también UMBRAL_ABSOLUTO

% En la matriz LBC se almacena la tabla "Límites de las Bandas Críticas" para la
% Capa II a 44.1 kHz, proporcionada por el estándar ISO 11172-3. Las frecuencias
% corresponden al límite superior de cada banda crítica.

% Índice  | Frecuencia | Tasa de Banda Crítica
%                (Hz)                 (z)
LBC = [
 1              43.066              0.425
 2              86.133              0.850
 3             129.199              1.273
 5             215.332              2.112
 7             301.465              2.934
10             430.664              4.124
13             559.863              5.249
16             689.063              6.301
19             818.262              7.274
22             947.461              8.169
26            1119.727              9.244
30            1291.992             10.195
35            1507.324             11.232
40            1722.656             12.125
46            1981.055             13.042
51            2325.586             14.062
56            2756.250             15.100
62            3273.047             16.110
69            3875.977             17.079
74            4478.906             17.904
79            5340.234             18.922
85            6373.828             19.963
92            7579.688             20.971
99            9302.344             22.074
105          11369.531             22.984
117          15503.906             24.013
130          19982.813             24.574
];
```

```
function NPS = Nivel_presion_sonora(F,FDE)
%NIVEL_PRESION_SONORA Nivel de presión sonora en cada subbanda.
%
%   NPS=NIVEL_PRESION_SONORA(F,FDE)
%   Determina el nivel de presión sonora para cada subbanda (vector NPS).
%   El vector F es la densidad espectral de potencia normalizada.
%   FDE es la matriz de factores de escala (3 por subbanda).
%
%   Ver también ANALISIS_FFT, FACTORES_ESCALA.

% Halla el máximo entre los 3 factores de escala de cada subbanda.
MAXFDE = max(FDE);

% Encuentra el valor máximo de las líneas de frecuencia para cada subbanda
% (componentes del vector XKI).
XKI = [];
for k = 1:16:512,
   Xk = max(F(k:k+15));
   XKI = [XKI; Xk];
end

% Calcula el nivel de presión sonora por subbanda (vector NPS).
for i = 1:32,
   NPS(i) = max(XKI(i),20*log10(MAXFDE(i)*32768)-10);
end
```

```
function [BSR,LTR,LNTR] = Reduccion(LT,LNT,BS,UA,MAP)
%REDUCCION Reduce la cantidad de componentes enmascarantes tonales y no-tonales
%   de la señal de audio.
%
%   [BSR,LTR,LNTR]=REDUCCION(LT,LNT,BS,UA,MAP)
%   La matriz LTR lista las componentes tonales y la matriz LNTR lista las
%   componentes no-tonales; ambas después de disminuir la cantidad de
%   componentes enmascarantes. Las componentes que se encuentran por debajo
%   del mínimo umbral auditivo, o que su distancia con respecto a otra componente
%   es menor a medio Bark, son eliminadas. Las matrices LTR y LNTR se componen
%   de dos columnas, la primera proporciona el índice de la línea de frecuencia
%   y la segunda el nivel de presión sonora de esa línea. El vector BSR
%   proporciona las banderas para cada una de las 512 líneas de frecuencia,
%   después de realizar la disminución sobre LT y LNT, tal que:
%   - Componente no examinada = 0.
%   - Componente tonal = 1.
%   - Componente no tonal = 2.
%   - Componente irrelevante = 3.
%
%   UA es la matriz de umbral absoluto y MAP es el vector de mapeo entre las
%   líneas de frecuencia y su índice para la matriz UA; ambos obtenidos con
%   UMBRAL_ABSOLUTO. LBC es la matriz de límites de las bandas críticas
%   obtenida con LIMITES_BANDA_CRITICA. LT y LNT son las matrices de
%   componentes tonales y de componentes no-tonales, y BS es el vector de
%   banderas, todos obtenidos con COMPONENTES_TONALES.
%
%   Ver también UMBRAL_ABSOLUTO, LIMITES_BANDA_CRITICA,
COMPONENTES_TONALES.

BSR = BS;

% a) Caso no-tonal y caso tonal, parte I:
%    Elimina las componentes tonales (filas de la matriz LT) o las componentes
%    no-tonales (filas de la matriz LNT) en las cuales el nivel de presión sonora
%    esté por debajo del mínimo umbral auditivo (tercera columna de la matriz
%    UA); y fija en 3 su valor de bandera en el vector BSR.

% Primero, se analiza el caso no-tonal.
LNTR = [];
if not(isempty(LNT))
      for i = 1:26,
         k = LNT(i,1);
         if (LNT(i,2) < UA(MAP(k),3))
            BSR(k) = 3;
            else
               LNTR = [LNTR; LNT(i,:)];
          end
      end
end

% Después, se analiza el caso tonal, parte I.
LTR = [];
if not(isempty(LT))
      for i = 1:length(LT(:,1)),
      k = LT(i,1);
      if (LT(i,2) < UA(MAP(k),3))
               BSR(k) = 3;
```

```
        else
            LTR = [LTR; LT(i,:)];
        end
      end
end

% b) Caso tonal, parte II:
%    Elimina dos o más componentes tonales (filas de la matriz LT) cuya distancia
%    respecto a otra componente es menor que medio bark, y fija en 3 su valor de
%    bandera en el vector BSR. Se elimina(n) la(s) componente(s) con menor nivel
%    de presión sonora y se conserva la componente con mayor nivel.
if not(isempty(LTR))
   i = 1;
   while (i < length(LTR(:,1))),
      k = LTR(i,1);
      sigk = LTR(i+1,1);
      if (UA(MAP(sigk),2) - UA(MAP(k),2) < 0.5)
         if (LTR(i,2) < LTR(i+1,2))
            % Elimina la componente con índice k.
            LTR = LTR([1:i-1,i+1:length(LTR(:,1))],:);
            BSR(k) = 3;
         else
            % Elimina la componente con índice k+1.
            LTR = LTR([1:i,i+2:length(LTR(:,1))],:);
            BSR(sigk) = 3;
         end
      end
      i = i+1;
   end
end
```

```
function MDCT = Transf_discreta_coseno(S,U576,sb,gr)
%TRANSF_DISCRETA_COSENO Calcula la transformada discreta del coseno modificada.
%
%   MDCT=TRANSF_DISCRETA_COSENO(S,U576,SB,GR)
%   Retorna los 18 valores (vector MDCT) correspondientes al resultado de aplicar
%   la transformada discreta del coseno modificada, con 50% de solapamiento,
%   sobre 36 muestras subbanda consecutivas en el tiempo.
%
%   S es la matriz de 1152 muestras subbanda obtenidas con FILTRO_SUBBANDA. La
%   matriz U576 proporciona las últimas 576 muestras subbanda de la matriz S
%   anterior, SB es el número de subbanda a procesar, GR es el gránulo actual,
%   los cuales fueron obtenidos en WAV2MP3.
%
%   Ver también FILTRO_SUBBANDA, WAV2MP3.

% En cada subbanda, los 18 valores de salida del gránulo anterior (almacenados en
% U576) y los 18 valores de salida del gránulo actual se ensamblan en un bloque
% de 36 muestras (vector xp). En caso de que el gránulo sea el segundo, para cada
% subbanda, la matriz S es procesada directamente.
if gr == 1
   xp = [U576(1:18,sb); S(1:18,sb)];
else
   xp = S(1:36,sb);
end

% Calcula el vector z de 36 valores para el tipo de bloque NORMAL.
z = xp' .* sin((pi/36)*(0.5:35.5));

% Calcula el vector MDCT de 18 valores para bloques largos.
MDCT = zeros(18,1);
fMDCT = pi/72*(19:2:89);
for i = 1:18,
   MDCT(i) = sum(z.*cos((2*i-1)*fMDCT));
end
```

```
function [UA,MAP,UES] = Umbral_absoluto
%UMBRAL_ABSOLUTO Mínimo Umbral Auditivo.
%
%   [UA,MAP,UES]=UMBRAL_ABSOLUTO
%   Proporciona en la matriz UA, los valores de:
%   - Índices de líneas de frecuencia (columna 1)
%   - Tasa de Banda Crítica (columna 2)
%   - Umbral Absoluto (columna 3)
%   para la capa II con una frecuencia de muestreo de 44.1 kHz. En el vector MAP se
%   proporciona el mapeo entre las líneas de frecuencia y su índice para la matriz UA.
%   El vector UES contiene solamente el Umbral Absoluto, también llamado Umbral en
%   Silencio.

% Almacena en la matriz TablaUA la tabla denominada "Frecuencias, Tasas de Banda
% Crítica y Umbral Absoluto" para la Capa II a 44.1 kHz, proporcionada por el
% estándar ISO 11172-3.
% ----------------  ------------
% Frecuencia | Tasa de Banda Crítica | Umbral Absoluto
%   (Hz)                 (z)                 (dB)
TablaUA = [
   43.07                0.425               45.05
   86.13                0.850               25.87
  129.20                1.273               18.70
  172.27                1.694               14.85
  215.33                2.112               12.41
  258.40                2.525               10.72
  301.46                2.934                9.47
  344.53                3.337                8.50
  387.60                3.733                7.73
  430.66                4.124                7.10
  473.73                4.507                6.56
  516.80                4.882                6.11
  559.86                5.249                5.72
  602.93                5.608                5.37
  646.00                5.959                5.07
  689.06                6.301                4.79
  732.13                6.634                4.55
  775.20                6.959                4.32
  818.26                7.274                4.11
  861.33                7.581                3.92
  904.39                7.879                3.74
  947.46                8.169                3.57
  947.46                8.450                3.40
 1033.59                8.723                3.25
 1076.66                8.987                3.10
 1119.73                9.244                2.95
 1162.79                9.493                2.81
 1205.86                9.734                2.67
 1248.93                9.968                2.53
 1291.99               10.195                2.39
 1335.06               10.416                2.25
 1378.13               10.629                2.11
 1421.19               10.836                1.97
 1464.26               11.037                1.83
 1507.32               11.232                1.68
 1550.39               11.421                1.53
 1593.46               11.605                1.38
```

1636.52	11.783	1.23
1679.59	11.957	1.07
1722.66	12.125	0.90
1765.72	12.289	0.74
1808.79	12.448	0.56
1851.86	12.603	0.39
1894.92	12.753	0.21
1937.99	12.900	0.02
1981.05	13.042	-0.17
2024.12	13.181	-0.36
2067.19	13.317	-0.56
2153.32	13.578	-0.96
2239.45	13.826	-1.38
2325.59	14.062	-1.79
2411.72	14.288	-2.21
2497.85	14.504	-2.63
2583.98	14.711	-3.03
2670.12	14.909	-3.41
2756.25	15.100	-3.77
2842.38	15.284	-4.09
2928.52	15.460	-4.37
3014.65	15.631	-4.60
3100.78	15.796	-4.78
3186.91	15.955	-4.91
3273.05	16.110	-4.97
3359.18	16.260	-4.98
3445.31	16.406	-4.92
3531.45	16.547	-4.81
3617.58	16.685	-4.65
3703.71	16.820	-4.43
3789.84	16.951	-4.17
3875.98	17.079	-3.87
3962.11	17.205	-3.54
4048.24	17.327	-3.19
4134.38	17.447	-2.82
4306.64	17.680	-2.06
4478.91	17.905	-1.32
4651.17	18.121	-0.64
4823.44	18.331	-0.04
4995.70	18.534	0.47
5167.97	18.731	0.89
5340.23	18.922	1.23
5512.50	19.108	1.51
5684.77	19.289	1.74
5857.03	19.464	1.93
6029.30	19.635	2.11
6201.56	19.801	2.28
6373.83	19.963	2.46
6546.09	20.120	2.63
6718.36	20.273	2.82
6890.63	20.421	3.03
7062.89	20.565	3.25
7235.16	20.705	3.49
7407.42	20.840	3.74
7579.69	20.972	4.02
7751.95	21.099	4.32
7924.22	21.222	4.64

```
   8096.48          21.342          4.98
   8268.75          21.457          5.35
   8613.28          21.677          6.15
   8957.81          21.882          7.07
   9302.34          22.074          8.10
   9646.88          22.253          9.25
   9991.41          22.420         10.54
  10335.94          22.576         11.97
  10680.47          22.721         13.56
  11025.00          22.857         15.31
  11369.53          22.984         17.23
  11714.06          23.102         19.34
  12058.59          23.213         21.64
  12403.13          23.317         24.15
  12747.66          23.415         26.88
  13092.19          23.506         29.84
  13436.72          23.592         33.05
  13781.25          23.673         36.52
  14125.78          23.749         40.25
  14470.31          23.821         44.27
  14814.84          23.888         48.59
  15159.38          23.952         53.22
  15503.91          24.013         58.18
  15848.44          24.070         63.49
  16192.97          24.125         68.00
  16537.50          24.176         68.00
  16882.03          24.225         68.00
  17226.56          24.271         68.00
  17571.09          24.316         68.00
  17915.63          24.358         68.00
  18260.16          24.398         68.00
  18604.69          24.436         68.00
  18949.22          24.473         68.00
  19293.75          24.508         68.00
  19638.28          24.542         68.00
  19982.81          24.574         68.00
];

% Convierte la primera columna de la matriz TablaUA de 130 valores de frecuencia
% a sus equivalentes 130 índices de líneas de frecuencia.
UA = [round(TablaUA(:,1)/44100*1024) TablaUA(:,2:3)];

% Almacena en el vector MAP el resultado del mapeo entre 512 líneas de frecuencia
% y sus 130 índices para la matriz UA. Primero, se mapea el valor inicial.
MAP(1) = 1;

% Después, se mapean los valores finales.
MAP(UA(130,1):512) = 130;

% Por último, se mapean todos los demás valores.
for i = 2:129,
  MAP(UA(i,1):UA(i+1,1)) = i;
end

% Como la mínima tasa de bits es 96 Kbps, se disminuyen en 12 dB los valores
% del Umbral Absoluto, almacenados en la tercera columna de la matriz UA.
UA(:,3) = UA(:,3)-12;
```

```
% Almacena en el vector UES la tercera columna de la matriz UA que contiene
% los valores de Umbral en Silencio o Umbral Absoluto para la Capa II.
UES = UA(:,3);
```

```
function  UEG = Umbral_enmasc_global(UES,UET,UENT)
%UMBRAL_ENMASC_GLOBAL Calcula el umbral de enmascaramiento global.
%
%   UEG=UMBRAL_ENMASC_GLOBAL(UES,UET,UENT)
%   Calcula el umbral de enmascaramiento global (vector UEG) para el subconjunto
%   de líneas de frecuencia definidas en la tabla denominada "Frecuencia, Tasa
%   de Banda Crítica y Umbral Absoluto" para la Capa II a 44.1 kHz, proporcionada
%   por el estándar ISO 11172-3. Éste es la suma (en la escala normal de amplitud
%   cuadrada del espectro) de los umbrales de enmascaramiento individual y del
%   umbral absoluto o umbral en silencio.
%
%   La matriz UET es el efecto enmascarante de las componentes tonales y la matriz
%   UENT es el efecto enmascarante de las componentes no tonales, ambas obtenidas
%   con UMBRALES_ENMASC_INDIVIDUAL. El vector UES es el umbral en silencio,
%   proporcionado por UMBRAL_ABSOLUTO.
%
%   Ver también UMBRALES_ENMASC_INDIVIDUAL, UMBRAL_ABSOLUTO.

% El umbral de enmascaramiento global (vector UEG) se calcula para el subconjunto
% de frecuencias definidas en la tabla denominada "Frecuencia, Tasa de Banda
% Crítica y Umbral Absoluto" para la Capa II a 44.1 kHz, proporcionada por el
% estándar ISO 11172-3. Éste es la suma de las potencias de los correspondientes
% umbrales de enmascaramiento individual (matrices UET y UENT) y el umbral en
% silencio (vector UES).
if not(isempty(UET))
   m = length(UET(:,1));
end
if not(isempty(UENT))
   n = length(UENT(:,1));
end

for i = 1:130

   % Para el umbral en silencio.
   t = 10^(UES(i)/10);

   % Contribución de las componentes tonales.
   if not(isempty(UET))
         for j = 1:m,
            t = t+10^(UET(j,i)/10);
      end
   end

   % Contribución de las componentes no tonales.
   if not(isempty(UENT))
      for j = 1:n,
         t = t+10^(UENT(j,i)/10);
      end
   end

   % Umbral de enmascaramiento global.
   UEG(i) = 10*log10(t);

end
```

```
function UEM = Umbral_enmasc_minimo(UEG,MAP)
%UMBRAL_ENMASC_MINIMO Calcula el umbral de enmascaramiento mínimo.
%
%   UEM=UMBRAL_ENMASC_MINIMO(UEG,MAP)
%   Retorna en el vector UEM el valor mínimo del umbral de enmascaramiento
%   global para cada una de las 32 subbandas.
%
%   El vector UEG es el umbral de enmascaramiento global obtenido con
%   UMBRAL_ENMASC_GLOBAL. El vector MAP es el mapeo entre las líneas de
%   frecuencia y su índice para la matriz UA en la función UMBRAL_ABSOLUTO.
%
%   Ver también UMBRAL_ENMASC_GLOBAL, UMBRAL_ABSOLUTO.

% Extrae del umbral de enmascaramiento global (vector UEG), su valor mínimo
% en cada subbanda (vector UEM).
for n = 1:32,
   UEM(n) = min(UEG(MAP((n-1)*16+1):MAP((n-1)*16+16)));
end
```

```
function [UET,UENT] = Umbrales_enmasc_individual(F,LTR,LNTR,UA,MAP)
%UMBRALES_ENMASC_INDIVIDUAL Calcula los umbrales de enmascaramiento
individual.
%
%   [UET,UENT]=UMBRALES_ENMASC_INDIVIDUAL(F,LTR,LNTR,UA,MAP)
%   Calcula el efecto enmascarante de las componentes tonales (matriz UET) y
%   de las componentes no tonales (matriz UENT) sobre las líneas de frecuencia
%   vecinas. Para esto, el nivel de presión sonora de cada componente enmascarante
%   es sumado con su índice de enmascaramiento y con su función de enmascaramiento.
%
%   F es el vector de densidad espectral de potencia normalizada, obtenido con
%   ANALISIS_FFT. UA es la matriz de umbral absoluto y MAP es el vector de mapeo
%   entre las líneas de frecuencia y su índice para la matriz UA; ambos obtenidos
%   con UMBRAL_ABSOLUTO. La matriz LTR es la lista reducida de las componentes
%   tonales y la matriz LNTR es la lista reducida de las componentes no-tonales,
%   ambas obtenidas con REDUCCION.
%
%   Ver también ANALISIS_FFT, COMPONENTES_TONALES, UMBRAL ABSOLUTO,
REDUCCION.

% Los umbrales de enmascaramiento individual para las componentes tonales
% y no tonales se fijan en -INF, ya que la función de enmascaramiento tiene
% atenuación infinita más allá de -3 y de +8 Barks, o sea, la componente no
% tiene efecto enmascarante sobre frecuencias más allá de aquellos rangos.
if isempty(LTR)
  UET = [];
else
  UET = zeros(length(LTR(:,1)),130) - 200; % -200 dB equivale a -INF.
end
if isempty(LNTR)
  UENT = [];
else
  UENT = zeros(length(LNTR(:,1)),130) - 200; % -200 dB equivale a -INF.
end

% Sólo un subconjunto de las muestras son consideradas para el futuro cálculo
% del umbral de enmascaramiento global. El número de estas muestras depende
% de la tasa de muestreo y de la capa de codificación. Toda la información
% necesaria está en la matriz UA, la cual contiene las frecuencias, las tasas
% de banda crítica y el umbral absoluto.
if not(isempty(LTR)) & not(isempty(LNTR))
  for i = 1:130,
    zi = UA(i,2);  % Tasa de banda crítica de la frecuencia considerada.
    if not(isempty(LTR))

      % Para las componentes tonales.
      for k = 1:length(LTR(:,1)),
        j  = LTR(k,1);
        zj = UA(MAP(j),2); % Tasa de banda crítica de la componente enmascarante.
        dz = zi-zj; % Distancia en Barks hasta la componente enmascarante.

        % Como la componente tonal tiene atenuación infinita más allá de -3 y
        % de +8 Barks, entonces los cálculos sólo se realizan para el rango
        % dz = {-3...+8}.
        if (dz>=-3 & dz<8)

          % Índice de Enmascaramiento.
```

```
            avtm = -1.525-0.275*zj-4.5;

            % Función de Enmascaramiento.
            if (-3<=dz & dz<-1)
              vf = 17*(dz+1)-(0.4*F(j)+6);
            elseif (-1<=dz & dz<0)
              vf = (0.4*F(j)+6)*dz;
            elseif (0<=dz & dz<1)
              vf = -17*dz;
            elseif (1<=dz & dz<8)
              vf = -(dz-1)*(17-0.15*F(j))-17;
            end

            % Umbral de enmascaramiento, componentes tonales.
            UET(k,i) = F(j)+avtm+vf;
          end
        end
      end

      % Para las componentes no tonales.
      if not(isempty(LNTR))
        for k = 1:length(LNTR(:,1)),
          j  = LNTR(k,1);
          zj = UA(MAP(j),2); % Tasa de banda crítica de la componente enmascarante.
          dz = zi-zj; % Distancia en Barks hasta la componente enmascarante.

          % Como la componente no-tonal tiene atenuación infinita más allá
          % de -3 y de +8 Barks, entonces los cálculos sólo se realizan para
          % el rango dz = {-3...+8}.
          if (dz>=-3 & dz<8)

            % Índice de Enmascaramiento.
            avnm = -1.525-0.175*zj-0.5;

            % Función de Enmascaramiento.
            if (-3<=dz & dz<-1)
              vf = 17*(dz+1)-(0.4*F(j)+6);
            elseif (-1<=dz & dz<0)
              vf = (0.4*F(j)+6)*dz;
            elseif (0<=dz & dz<1)
              vf = -17*dz;
            elseif (1<=dz & dz<8)
              vf = -(dz-1)*(17-0.15*F(j))-17;
            end

            % Umbral de enmascaramiento, componente no-tonal.
            UENT(k,i) = F(j)+avnm+vf;

          end
        end
      end
    end
  end
end
```

Especificaciones ISO MPEG

MPEG-1 : El estándar internacional ISO/IEC 11172, más conocido como MPEG-1 (Codificación de imágenes en movimiento y el audio asociado para medios de almacenamiento digital a una tasa cercana a 1.5 Mbps). Cinco partes, las tres primeras estandarizadas desde 1992.
Terminado.

- **IS-11172-1** (Sistema) - Describe la sincronización y multiplexación de señales de audio y video.
- **IS-11172-2** (Video) - Describe la compresión de señales de video, centrándose en el escaneo progresivo y considerando especialmente las aplicaciones de video en CD.
- **IS-11172-3** (Audio) - Describe una familia genérica de codificación de audio, con tres miembros jerárquicamente compatibles, denominados esquema-1, esquema-2 y esquema-3. Contempla la codificación de uno (mono) o dos (estéreo/dual) canales de audio digital con frecuencias de muestreo de 32, 44.1 ó 48 KHz. La velocidad de transmisión varía entre 32 y 448 Kbits/seg en la Capa I, entre 32 y 384 Kbits/seg en la Capa II, y entre 32 y 320 Kbits/seg en la Capa III.
- **IS-11172-4** (Test de conformidad) - Describe los procedimientos para determinar las características de los bitstreams codificados y el

proceso de decodificación, así como los test de conformidad con los requerimientos establecidos en las otras partes.

- **DTR-11172-5** (Simulación por software) - No se trata de un estándar, sino de un reporte técnico. Es un informe técnico sobre la implementación por software de las tres primeras partes de MPEG-1.

MPEG-2 : Numerado de manera formal como ISO/IEC MPEG 13818 (Codificación genérica para información de imágenes en movimiento y el audio asociado). Nueve partes, las tres primeras estandarizadas desde 1994, con algunos añadidos posteriores. En diferentes estados de acabado. En su primera fase fue una simple extensión en las capacidades de compresión y codificación del primer estándar, mientras que en la segunda se desarrollaron nuevos algoritmos que implicaban métodos para realizar la compresión tanto de video como de audio, pero en este punto se sacrificó la compatibilidad con el estándar anterior en algunas áreas.

- **IS-13818-1** (Sistema) - Describe la sincronización y multiplexación de señales de audio y video; estandarizado por ITU-T como H.222.
- **IS-13818-2** (Video) - Describe un conjunto genérico de herramientas para codificación de video; estandarizado por ITU-T como H.262.
- **IS-13818-3** (Audio) - MPEG-2 BC. Describe una extensión de MPEG-1 compatible hacia atrás (BC=Backwards Compatible), para codificación de audio multicanal, esto es, admite hasta 5 canales principales (sonido envolvente, sonido multilingüe) y una extensión no compatible hacia atrás (un canal LFE; Low Frecuency

Enhancement = refuerzo de bajas frecuencias) para frecuencias de muestreo inferiores, para soportar aplicaciones de sonido con requerimientos de ancho de banda limitado. Aumento de la velocidad de transmisión hasta aproximadamente 1 Mbit/seg. Permite el uso de frecuencias de muestreo menores: 16, 22.05 y 24 KHz para velocidades de transmisión entre 32 y 256 Kbits/seg (Capa I), y entre 8 y 160 Kbits/seg (Capa II y Capa III).

- **IS-13818-4** (Test de concordancia) - Describe los procedimientos para determinar las características de los bitstreams codificados y el proceso de decodificación, así como los test de conformidad con los requerimientos establecidos en las otras partes.
- **DTR-13818-5** (Simulación por software) - Es un informe técnico sobre la implementación por software de las tres primeras partes de MPEG-2.
- **IS-13818-6** (Extensiones de sistema Control y comandos para medios de almacenamiento digital) - Describe un conjunto de protocolos para aplicaciones cliente-servidor.
- **CD-13818-7** (Audio, codificación no compatible hacia atrás (NBC)) - MPEG-2 AAC. Describe un esquema de codificación de audio mejorado para señales mono y estéreo, así como para sonido multicanal. Es una norma de compresión de muy alta calidad. Admite hasta 48 canales de audio y frecuencias de muestreo desde 8 hasta 96 KHz con capacidad multicanal, multiidioma y multiprograma. Las velocidades de transmisión van desde 8 Kbits/seg (señal vocal

monofónica) hasta más de 160 Kbits/seg/canal para señales de muy alta calidad que permiten ciclos múltiples de codificación-decodificación.

- **IS-13818-8** (Video, extensión para muestras de entrada de 10 bits) - Se ha retirado, debido al escaso interés.
- **IS-13818-9** (Especificaciones del interface en tiempo real para aplicaciones low-jitter) - Define las restricciones temporales para el envío en tiempo real de bitstreams MPEG-2.
- **WD-13818-10** (Extensiones de concordancia - DSM-CC) - Describe los añadidos a IS-13818-4 para DSM-CC.

MPEG-4 (ISO/IEC 14496-3):

Se basa en el concepto de Audio Estructurado que permite representar sonidos naturales (como la voz y la música) y sintetizar cualquier tipo de sonido basándose en descripciones estructuradas. Esto permitirá, por ejemplo, transmitir un texto junto con la prosodia deseada (entonación, duración de los fonemas, etc.) y generar en destino la voz correspondiente. Una de las muchas ventajas es que se podrá transmitir voz con velocidades de transmisión inferiores a 1 Kbit/seg (y posiblemente elegir en qué idioma queremos escucharla).

La norma contempla:

- Codificación y composición de objetos sonoros tanto naturales como sintéticos.
- Escalado de la velocidad de transmisión de la señal de audio

digital.

- Escalado de la complejidad tanto del codificador como del decodificador.
- Audio Estructurado para síntesis sonora controlada por guiones.
- Interfaz TTSI para sistemas de conversión de texto en diálogos.

MPEG-7 (ISO/IEC 15938):

Esta norma, que ha diferencia de las anteriores no define algoritmos de compresión, proporcionará:

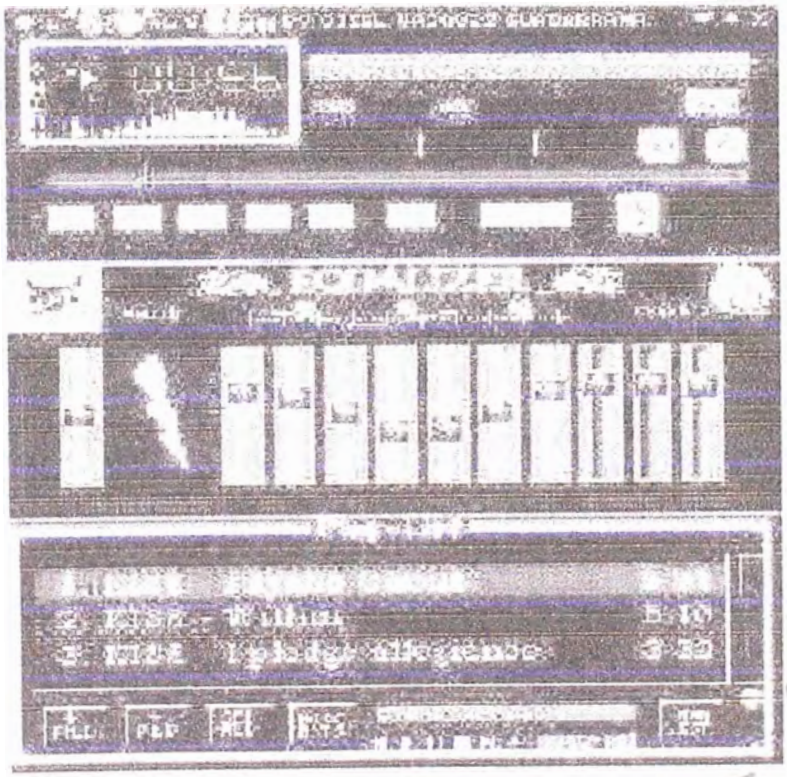
- Descripciones normalizadas y esquemas descriptivos de estructuras de audio y contenidos sonoros.
- El lenguaje para implementar estas descripciones y esquemas descriptivos.

Se aprobó en Julio del año 2001.

Escuela Superior de Ingenieros
Injineruen Goimailako Eskola
UNIVERSIDAD DE NAVARRA - NAFARROAKO UNIBERTSITATEA

Trabajo de Matlab 2001-2002

Compresión de ficheros de sonido



Javier Atencia - Miguel Ángel Aybar - Aiert Amundarain
Iker Aguinaga - Diego Borro - Gonzalo Martínez

San Sebastián, Octubre 2001

Trabajo de Matlab 2001-2002

Índice

1. Introducción

El trabajo de Matlab para el Curso 2000-01 consiste en la creación de un software específico para la compresión de ficheros de sonido. Habrá que desarrollar una aplicación de tal manera que, a partir de un fichero de sonido (***.wav**), que se obtendrá por medio de un CD de audio, se pueda generar otro fichero, también de sonido, que además de tener un tamaño menor, tenga la misma (o casi la misma) calidad de sonido que el fichero ***.wav** o la pista del CD original.

1.1 La compresión de ficheros

Los ficheros de información son susceptibles de ser recodificados o comprimidos por medio de algoritmos y procedimientos especiales, de forma que ocupen menos bytes. Así se consiguen algunas ventajas, por ejemplo que los archivos ocupen menos espacio en los discos. Por eso se desarrollan programas de compresión y descompresión que facilitan estas tareas.

La compresión es posible porque normalmente en una fuente (el código, los caracteres que componen un fichero), además de información hay también redundancia, es decir, datos que no aportan más información, en general porque pueden obtenerse a partir de los datos anteriores.

¿Existe un límite en cuanto a lo que podemos llegar a comprimir un fichero, un punto a partir del cual no se pueda reducir ya más el tamaño de la fuente, un punto en el que se haya eliminado toda la redundancia y ya sólo quede información? La intuición nos dice que sí. Si llegáramos a comprimirlo hasta 0 ó 1 byte, parece difícil que se pueda sacar toda la información de ahí.

Dentro de la Teoría de la Información, que es la rama de la ciencia que se ocupa de la compresión de datos, hace tiempo que se ha demostrado que ese límite, efectivamente, existe, y recibe el nombre de *entropía*.

Todos los ficheros no son comprimibles en la misma proporción. En líneas generales podemos decir que los ficheros de texto pueden comprimirse razonablemente, que los ficheros de programas pueden comprimirse poco, y que otros ficheros (p. ej. gráficos BMP) pueden comprimirse mucho más. También podemos asegurar que la compresión será menos eficaz si el fichero es pequeño.

1.2 Compresión de ficheros de audio

1.2.1 Sonido analógico y sonido digital

Hay dos formas de almacenar sonidos: la analógica y la digital. Las grabaciones analógicas son, por ejemplo, las que podemos escuchar en una cinta de casete. Son señales continuas, impresas en un soporte magnético. Un disco compacto es una grabación digital. En ellas, la señal de sonido no es continua. Está dividida en pequeños intervalos de tiempo. En cada intervalo el sonido es constante. Juntando todos los pequeños intervalos se consigue una curva que se aproxima a la del sonido real. Los intervalos de tiempo que se emplean en este proceso son tan pequeños que el oído humano no es capaz de distinguir esos saltos discretos de tono entre un intervalo y otro.

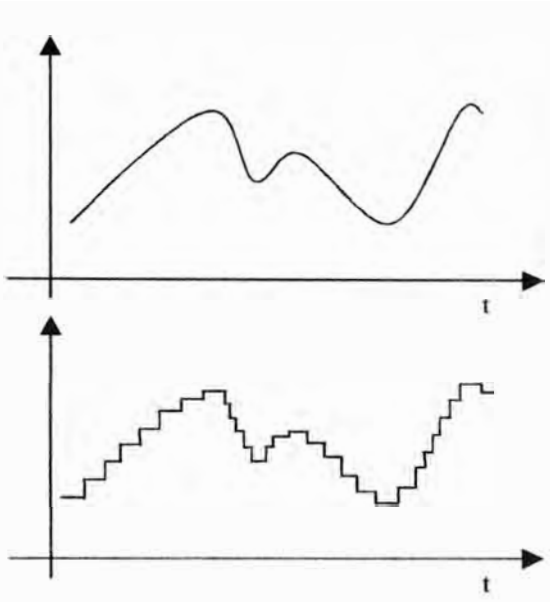


Fig. 1: sonidos analógico y digital

1.2.2 Problemas de espacio: ficheros **.wav* y **.mp3*

En los últimos años se ha intentado comprimir ficheros de sonido al máximo y mantener una alta fidelidad al momento de reproducirlo, sin embargo esa calidad no ha sido lograda hasta la aparición de nuevos formatos de audio.

¿Qué es el MP3 en definitiva? Pues nada más que un método de almacenamiento capaz de almacenar música siguiendo unas especificaciones muy concretas que dan como resultado una calidad sonora equiparable a la de los CDs pero con un tamaño hasta 11 veces menor. El MP3 se basa en un algoritmo que, al contrario de los compact discs, no procura la máxima calidad de sonido cueste lo que cueste, sino proporcionar la máxima calidad de sonido que nuestros oídos son capaces de percibir. Es decir, su principal misión es hacer que la señal final suene igual que la señal original en base a la capacidad auditiva de los humanos, y no por lo que indican los osciloscopios. Como esta capacidad está entre los 20 y 20.000 Hz, en el momento en que se genera un MP3, el algoritmo elimina todas aquellas frecuencias que estén fuera de estos límites, partiendo de la base de que como no vamos a oírlas no son necesarias para el resultado final. Esta técnica, junto a otras argucias de compresión consiguen reducir a menos de un décimo el tamaño de los ficheros MP3 respecto a su equivalente en calidad CD, y todo sin que el oído humano pueda notar apenas diferencias entre uno y otro.

Generalmente, el sonido se almacena en ficheros ****.wav***. Estos ficheros ocupan una gran cantidad de espacio en los discos duros, ya que almacenan la información completa de los CDs. El formato MP3 no es más que un ****.wav*** (wave) con una calidad muy elevada de compresión.

Para obtener un archivo MP3 con calidad CD se deben hacer dos cosas. Primero debe extraerse la información del CD-Audio y pasarla al disco duro. Esto lo hace un programa denominado extractor (*ripper* o *ripeador*). De esta manera ya tenemos la información del CD-Audio en el disco duro, generalmente en un voluminoso fichero ****.wav***.

Segundo, debido al gran tamaño de estos ficheros, necesitamos comprimirlos, y aquí es donde surge el compresor, que nos pasará este fichero ***.wav*** a un fichero MP3, el cual contiene prácticamente la misma calidad de sonido pero ocupa 11 veces menos espacio.

Finalmente, para poder escuchar la canción necesitaremos un reproductor de MP3 (p. ej. el *Winamp*), capaz de reproducir este tipo de ficheros.

La casi milagrosa capacidad para comprimir sin perder calidad es lo que ha hecho del MP3 el formato de distribución musical por excelencia de la Red, donde la escasez de ancho de banda obliga a pensárselo mucho antes de enviar o descargar ficheros de varios megas.

1.2.3 Actualidad. Napster. Gnutella. Bajada de ficheros a través de Internet

El futuro del MP3 es muy alentador, ya que cada día más empresas lanzan al mercado reproductores portátiles y para poder oírlos sin necesidad de computadoras, ya sea como complemento a una cadena de música, como una especie de walkman o como reproductor para portátil.

Fig. 2: logotipo de Napster

Empresas como Napster y Gnutella, dedicadas al intercambio de ficheros multimedia (sobre todo MP3) a través de la red, están creando una gran polémica. De hecho, son bien conocidos los pleitos que mantienen con casas discográficas y autores. Entre los artistas también hay división de opiniones en cuanto a la legalidad de sus actividades. Hay cantantes y grupos que manifiestan abiertamente su apoyo a compartir ficheros MP3, y de hecho colocan sus canciones en la red para que todo el mundo pueda disfrutar de ellas gratuitamente; y hay artistas que no sólo se declaran en contra de estas prácticas, sino que incluso mantienen litigios con estas empresas, ya que consideran violados sus derechos de autor.

Fig. 3: logotipo de Gnutella

2. Compresión de ficheros de audio. Series de Fourier. Descomposición en armónicos

Como ya hemos dicho, el MP3 es un sistema de compresión de archivos de audio que permite reducir hasta 11 veces un archivo de sonido ***.wav***. Para ello el algoritmo de compresión MP3 descompone la onda en sus armónicos elementales y destruye aquellos menos significativos, dando buenos resultados prácticamente sin pérdida de calidad.

2.1 Descomposición en forma analógica o continua

El sonido que escuchamos procedente de cualquier fuente es una función continua en el tiempo. El tono que escuchamos depende de la frecuencia con la que vibra el medio por donde se propaga el sonido (normalmente el aire).

Una función continua y periódica se puede descomponer en una suma de infinitas funciones sinusoidales.

Supongamos una función periódica $y = f(x)$, con período T, como la de la Fig. 4. La función $f(x)$ puede tener cualquier expresión, con tal de que cumpla ser *periódica, continua y acotada* en todos los puntos de su período.

Se puede demostrar:

Si se escoge un período cualquiera de la función, es decir, cualquiera de los intervalos $[x_1, x_2]$, $[x_3, x_4]$, etc., existe una serie infinita de funciones sinusoidales, una serie de la forma:

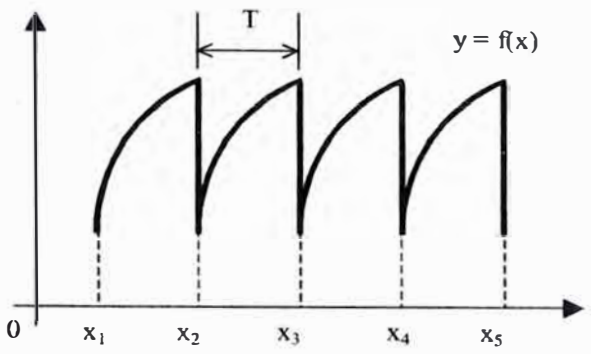


Fig. 4: $y=f(x)$ periódica

$$S(x) = a_0 + \sum_{n=1}^{\infty}\left(a_n \cos\frac{n\pi x}{T} + b_n sen\frac{n\pi x}{T}\right) \qquad (1)$$

O lo que es lo mismo:

$$S(x) = a_0 + a_1\cos\frac{\pi x}{T} + b_1 sen\frac{\pi x}{T} + a_2\cos\frac{2\pi x}{T} + b_2 sen\frac{2\pi x}{T} + a_3\cos\frac{3\pi x}{T}3x + b_3 sen\frac{3\pi x}{T} + \ldots \qquad (2)$$

Que es equivalente a la función $y = f(x)$ en el intervalo considerado, es decir:

$$S(x) = f(x) \qquad \forall x \in \text{intervalo} \qquad (3)$$

Además, siempre es posible escribir la expresión (1) en la forma:

$$S(x) = a_0 + \sum_{n=1}^{\infty} c_n sen\left(\frac{n\pi x}{T} + \varphi_n\right) = a_0 + c_1 sen(x+\varphi_1) + c_2 sen(2x+\varphi_2) + c_3 sen(3x+\varphi_3) + \ldots \qquad (4)$$

Donde los c_n son coeficientes escalares y los φ_n son ángulos de desfase entre el origen y cada uno de los sumandos. Éstos se obtienen mediante fórmulas matemáticas que se pueden consultar en cualquier libro de cálculo.

Ésta es la ***descomposición en serie de Fourier*** de la función $y = f(x)$.

Los términos $c_n sen\left(\frac{n\pi x}{T} + \varphi_n\right)$ son los ***armónicos elementales*** de la serie. El término $c_1 sen\left(\frac{\pi x}{T} + \varphi_1\right)$ es el armónico de primer orden, $c_2 sen\left(\frac{2\pi x}{T} + \varphi_2\right)$ es el armónico de 2º orden, $c_3 sen\left(\frac{3\pi x}{T} + \varphi_3\right)$ es el de tercer orden, etc.

Se cumple además que el período del armónico de primer orden dura lo mismo que el intervalo considerado (T). El 2º armónico tiene frecuencia doble, el 3º frecuencia triple, etc.

Todo esto se puede aplicar a una grabación sonora de la siguiente manera: se tiene un sonido grabado, de una duración determinada. Supongamos que ese sonido se repitiese de manera periódica. Se tendría grabado uno de los períodos de esa función sonora, una grabación de T segundos (Fig. 5).

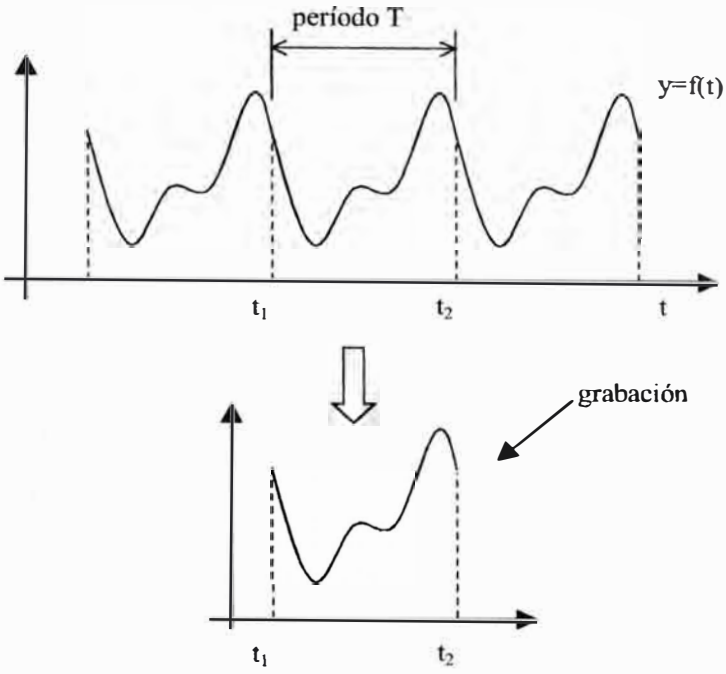


Fig. 5: se escoge un período T

Según lo visto anteriormente, para ese intervalo T, se cumple:

$$y = f(t) = a_0 + \sum_{n=1}^{\infty} c_n sen\left(\frac{n\pi t}{T} + \varphi_n\right) \qquad (5)$$

El cociente $\frac{n\pi}{T}$ es la frecuencia del armónico n.

Se tiene entonces que esa curva de sonido es una suma de infinitas curvas senoidales. Ese sonido grabado es una suma de infinitos sonidos, cada uno con una frecuencia mayor que la del anterior, llegando hasta una frecuencia infinita.

2.2 Descomposición en forma discreta o digital

Lo explicado antes se puede aplicar también al caso discreto. En MATLAB las curvas se representan uniendo puntos discretos mediante tramos rectos. Son aproximaciones a la curva continua que se quiere representar. La aproximación será más exacta cuantos más puntos se tengan.

En el caso discreto la información es limitada (según el número de puntos), por lo que no se puede obtener una suma de infinitos senos. Lo mismo que se aproximan los infinitos puntos de una curva continua por un número finito de puntos, al hacer la descomposición en serie de Fourier en el caso discreto se tiene que aproximar la serie infinita por un número finito de armónicos.

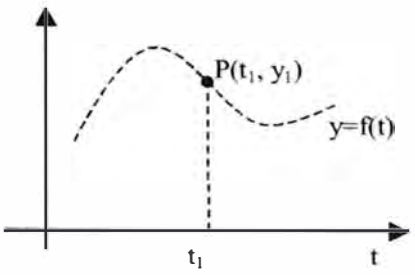


Fig. 6: curva definida con 1 punto

Veámoslo con un par de ejemplos. Si tuviésemos una curva *y=f(t)* definida con un sólo punto, no podríamos aproximarla por una función sinusoidal de la forma:

$$y(t) \approx c_1 sen\left(\frac{\pi t}{T} + \varphi_1\right) \qquad (6)$$

Para hallar función sinusoidal buscada necesitamos hallar c_1 y φ_1, pero sólo se dispone de un punto como fuente de información. Hay infinitos senos de una determinada frecuencia que pasan por el punto P (Fig. 6). La mínima información necesaria la proporcionan 2 puntos.

Con 2 puntos, sólo hay una función sinusoidal de una determinada frecuencia, definida por su amplitud (c_1) y su ángulo de desfase (φ_1) que pase por P_1 y P_2. Pero al igual que sólo con P_1 y P_2 aproximamos de forma bastante imprecisa la curva *y=f(t)*, únicamente con una función sinusoidal, sólamente con un armónico, tampoco nos acercamos mucho a la curva verdadera.

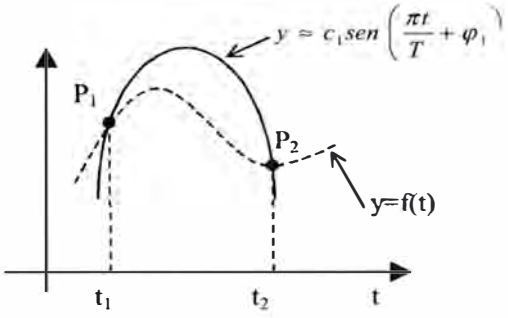


Fig. 7: curva definida con 2 puntos

Cuantos más puntos de la curva se conozcan, la aproximación será mejor. También se podrá obtener un mayor número de armónicos, es decir, se podrá obtener un mayor número de sumandos de la serie de Fourier.

3. MATLAB

El oído humano sólo puede escuchar sonidos dentro de un determinado rango de frecuencias, más o menos entre los 20 y los 20.000 Hz. Cualquier sonido fuera de este rango no somos capaces de escucharlo. Entonces, si se conservan sólamente las frecuencias audibles, si de esos infinitos sumandos se eliminan los de frecuencias muy altas, se tendría una suma con un número finito de términos, y el sonido que produciría sería exactamente igual para nuestros oídos. Se habría eliminado toda la información sonora que no somos capaces de escuchar, y que por lo tanto no no interesa. Ésta es la base del formato MP3 y del método que se va a emplear para realizar este trabajo.

3.1 Conversión del sonido a datos manipulables mediante MATLAB

A partir de un CD de audio, se generará un fichero de sonido ***.wav***, de una duración determinada. Será un fichero voluminoso, con gran cantidad de información (los ***.wav*** ocupan mucha memoria). Será necesario transformar la información contenida en archivos ***.wav*** en datos con los que

MATLAB pueda trabajar. También hará falta escuchar la información sonora de esos datos. Para todo esto MATLAB dispone de una serie de funciones que son las siguientes:

- *wavread( )* y *wavwrite( )*
- *wavplay( )* y *wavrecord( )*
- *sound( )* y *soundsc( )*

En el ***Help*** de MATLAB se explica detalladamente para qué sirve y cómo se emplea cada una de estas funciones.

3.2 Desarrollo en serie de Fourier

Teniendo una función *x(t)* con N puntos conocidos, en un vector *x*, se usa la función *fft( )* de MATLAB, que proporciona la transformada de Fourier discreta (otro vector de N elementos, números complejos).

$$X = fft(x) \qquad \text{ó} \qquad X = fft(x, N)$$

$$X(k) = \sum_{j=1}^{N} x(j) e^{-\frac{2\pi i}{N}(j-1)(k-1)} \qquad 1 < k < N \tag{7}$$

$$x(n) = \frac{1}{N} \sum_{k=1}^{N} X(k) e^{\frac{2\pi i}{N}(k-1)(n-1)} \qquad 1 < n < N$$

Si se desarrolla:

$$x(n) = a_0 + \sum_{k=1}^{N/2} a_k \cos\frac{2\pi t(n)}{Ndt} + b_k \operatorname{sen}\frac{2\pi t(n)}{Ndt} \tag{8}$$

$$a_0 = \frac{X(1)}{N} \qquad a_k = 2 \cdot real\frac{X(k+1)}{N} \qquad b_k = 2 \cdot imag\frac{X(k+1)}{N}$$

En las expresiones, t(n) representaría el vector de tiempos , y dt sería el intervalo de tiempo entre t(n) y t(n-1).

Interesa obtener los coeficientes a_k y b_k. Pero es más interesante tener un solo coeficiente para cada armónico, y para ello se puede hacer un arreglo matemático:

$$a\cos(x) + b\operatorname{sen}(x) = c\operatorname{sen}(x + \varphi)$$

$$\frac{a}{c}\cos(x) + \frac{b}{c}\operatorname{sen}(x) = \operatorname{sen}(x + \varphi) \tag{9}$$

$$\operatorname{sen}(x)\cos(\varphi) + \cos(x)\operatorname{sen}(\varphi) = \operatorname{sen}(x + \varphi)$$

$$\frac{a}{c} = \operatorname{sen}(\varphi) \quad \frac{b}{c} = \cos(\varphi) \qquad c = \frac{a}{\operatorname{sen}(\varphi)} = \frac{b}{\cos(\varphi)}$$

Y entonces:

$$x(n) = a_0 + \sum_{k=1}^{N/2} c_k \operatorname{sen}\left(\frac{2\pi k t(n)}{Ndt} + \varphi_k\right)$$

$$c_k = \frac{a_k}{\operatorname{sen}\varphi_k} = \frac{b_k}{\cos\varphi_k} = \frac{2 \cdot realX(k+1)}{N\operatorname{sen}\varphi_k} = \frac{-2 \cdot imagX(k+1)}{N\cos\varphi_k} \tag{10}$$

$$\varphi_k = \operatorname{arctg}\left(\frac{a_k}{b_k}\right)$$

La función *fft( )* devuelve un vector de números complejos. El cociente a_k/b_k se obtiene del ángulo de cada número complejo.

$$\frac{a_k}{b_k} = \frac{\frac{1}{N} 2realX(k+1)}{-\frac{1}{N} 2imagX(k+1)} = \frac{realX(k+1)}{-imagX(k+1)} \qquad (11)$$

Finalmente, se obtienen las incógnitas c_k, las componentes de los armónicos, y los ángulos φ_k, sus ángulos de desfase.

Supongamos un vector x con 1000 puntos. Al escribir:

```
fft(x)
```

se obtendrá un vector X con 1000 números complejos. El primer elemento, *X(1)*, será la componente a_0 del desarrollo en serie. Los elementos desde el 2 al 501 serán números complejos, y los elementos desde el 502 al 1000 seran los complejos conjugados de los anteriores.

De la misma manera que mediante la función *fft( )* se obtiene un determinado número de armónicos a partir de un vector (que define una función), se puede seguir el paso inverso. A partir de un determinado número de armónicos se puede obtener la función que éstos definen.

Esto se consigue con la función *ifft( )* de MATLAB, que realiza el paso inverso a *fft( )*. Para ver en detalle cómo se utiliza, lo mejor es consultar el ***Help*** de MATLAB.

4. Posibles pasos a seguir en la realización del trabajo

*4.1 Reducción del tamaño de archivos *.wav*

Partiendo de un archivo ****.wav***:

- Transformarlo a un vector manipulable desde MATLAB.
- Analizarlo mediante las funciones *fft( )*. Desechar armónicos innecesarios.
- Reconstruir el vector y escucharlo.

4.2 División en intervalos

Partiendo también del archivo ****.wav***:

- Transformarlo a vector, y dividirlo en pequeños intervalos con Δt constante.
- Analizar cada intervalo aplicando *fft( )* y obtener una matriz de coeficientes y ángulos de desfase.
- Desechar armónicos no deseados, y reconstruir el vector de sonido.
- Escuchar el vector intervalo a intervalo.

4.3 Creación de un interface *de usuario*

Desarrollo de un *interface* de usuario, capaz de generar un archivo de sonido comprimido, con formato propio de cada grupo, y que sea ejecutable bien desde MATLAB o bien desde otro programa elaborado por el grupo.

NOTA IMPORTANTE: ***Estas explicaciones son indicativas y se irán completando durante las clases teóricas, con ayudas en el WEB de la asignatura y con instrucciones más detalladas sobre cómo se realizará la entrega y la presentación del trabajo.***

```
» help fft

 FFT Discrete Fourier transform.
    FFT(X) is the discrete Fourier transform (DFT) of vector X.  If the
    length of X is a power of two, a fast radix-2 fast-Fourier
    transform algorithm is used.  If the length of X is not a
    power of two, a slower non-power-of-two algorithm is employed.
    For matrices, the FFT operation is applied to each column.
    For N-D arrays, the FFT operation operates on the first
    non-singleton dimension.

    FFT(X,N) is the N-point FFT, padded with zeros if X has less
    than N points and truncated if it has more.

    FFT(X,[],DIM) or FFT(X,N,DIM) applies the FFT operation across the
    dimension DIM.

    For length N input vector x, the DFT is a length N vector X,
    with elements
                    N
      X(k) =        sum  x(n)*exp(-j*2*pi*(k-1)*(n-1)/N), 1 <= k <= N.
                   n=1
    The inverse DFT (computed by IFFT) is given by
                    N
      x(n) = (1/N) sum  X(k)*exp( j*2*pi*(k-1)*(n-1)/N), 1 <= n <= N.
                   k=1

    The relationship between the DFT and the Fourier coefficients a and b in
              N/2
    x(n) = a0 + sum a(k)*cos(2*pi*k*t(n)/(N*dt))+b(k)*sin(2*pi*k*t(n)/(N*dt))
              k=1
    is
      a0 = X(1)/N, a(k) = 2*real(X(k+1))/N, b(k) = -2*imag(X(k+1))/N,
    where x is a length N discrete signal sampled at times t with spacing dt.

    See also IFFT, FFT2, IFFT2, FFTSHIFT.

» help ifft

 IFFT Inverse discrete Fourier transform.
    IFFT(X) is the inverse discrete Fourier transform of X.

    IFFT(X,N) is the N-point inverse transform.

    IFFT(X,[],DIM) or IFFT(X,N,DIM) is the inverse discrete Fourier
    transform of X across the dimension DIM.

    See also FFT, FFT2, IFFT2, FFTSHIFT.

 Overloaded methods
    help uint8/ifft.m
```