UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

IMPLEMENTACIÓN DE UN CÓDEC DE VOZ PROPIETARIO PARA APLICACIONES DE CODIFICACIÓN DE VOZ EN BANDA ANCHA CON BAJA RELACIÓN SEÑAL A RUIDO

Jesús Marcos Zamarreño

JULIO 2009

Implementación de un códec de voz para aplicaciones de codificación de voz en banda ancha con baja relación señal a ruido

AUTOR: Jesús Marcos Zamarreño
TUTOR: Doroteo Torre Toledano

Área de Tratamiento de Voz y Señales - ATVS
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2009

RESUMEN

Este proyecto tiene como finalidad el desarrollo de un códec de voz propietario para satisfacer las necesidades de una empresa que desea sustituir el uso del GSM-Full Rate por uno más moderno, con mejor comportamiento frente al ruido, que permita la utilización de grabaciones de voz en banda ancha y manteniendo o bajando la tasa binaria empleada durante la codificación. Ésta implementación debe también poderse ejecutar en un sistema embebido de reducida capacidad computacional y sin unidad de punto flotante.

Para ello, se ha realizado un estudio comparativo del comportamiento del GSM-Full Rate y dos posibles candidatos a formar la base del códec propietario, el AMR-WB y el SPEEX 1.2b3. En este estudio se han realizado mediciones de calidad objetiva, coste computacional y procesamiento posterior (reconocimiento de locutor, habla, idioma y filtrado de ruido).

A continuación se realizaron los cambios necesarios de cara a permitir la ejecución en el sistema embebido en los tiempos requeridos por el cliente y las modificaciones de formato llevadas a cabo para conseguir un formato propietario.

Por último se desarrolló un códec-driver ACM que permite codificar, decodificar y reproducir archivos del formato desarrollado desde las aplicaciones de Windows.

PALABRAS CLAVE

Codificación, Códec, GSM-Full Rate, AMR-WB, SPEEX, Audio Compression Manager, ACM

ABSTRACT

This Project aims to develop a proprietary voice codec to meet the needs of a company that wants to replace the use of GSM-Full Rate by a newer one with better performance in noisy scenarios, allowing the use of bandwidth voice recordings while maintaining or lowering the bitrate used during encoding. This implementation must work properly in an embedded system with small computational capabilities and without floating-point unit.

In order to achieve this goal, we made a comparative study between GSM-Full Rate and two potential candidates, the AMR-WB and Speex 1.2b3. In this study, we measured objective quality on codification as well as computational cost and the effects on post processing (speaker, language and speech recognition and noise filtering).

Once we selected the basis coded, we describe the changes made to achieve the proper execution on the embedded system satisfying time requisites exposed by clients as well as the format adaptation to produce a proprietary format.

Finally, we developed an ACM codec-driver than enables encoding, decoding and playback of these proprietary files on windows applications.

KEYWORDS

Codification, Codec, GSM-Full Rate, AMR-WB, SPEEX, Audio Compression Manager, ACM

AGRADECIMIENTOS

En primer lugar quisiera agradecer a Joaquín González la oportunidad de trabajar en el ATVS y a mi tutor Doroteo Torre Toledano su confianza en mí para llevar a cabo este proyecto. Trabajar con él durante estos meses ha sido, además de un honor, un placer. Muchas gracias por su tiempo, dedicación y apoyo.

A todos los profesores que han pasado por mi vida académica desde que entré en el colegio hasta ahora, especialmente a los de la Escuela Politécnica Superior.

A todos los miembros del ATVS, por su apoyo y ayuda cuando ha sido necesario.

A todos mis compañeros y amigos que he conocido durante estos años de carrera. A los que conocí en el colegio y fuera de él, a todos gracias por estar ahí.

A toda mi familia, en especial a mis padres José María y Fermina y mi hermana Virginia a los que les debo todo lo que soy. Por ser siempre el mejor referente y ejemplo a seguir. Por todo su amor, tan grande como el que siento yo por ellos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

1.1 Motivación

El tratamiento de voz en su forma analógica se remonta casi un siglo y cuarto en la historia con los primeros tiempos de la telefonía. Sin embargo, fue hace unos 40 años cuando apareció el interés por el tratamiento digital de voz al iniciar la industria de las telecomunicaciones un proyecto global para digitalizar la red telefónica. Las demandas del mercado llevaron a una preferencia por aumentar la capacidad de transmisión en lugar de buscar métodos para reducir la tasa binaria de la voz, motivo por el que la tecnología de compresión de voz estuvo bastante tiempo dormida.

Con el nacimiento de los sistemas digitales y de las posibilidades que éstos ofrecían, pronto se pasó a las implementaciones digitales. Fue en 1939 cuando se produjo el primer desarrollo importante en la historia de la compresión de voz con la invención de los vocoders, codificadores paramétricos basados en la producción de la señal de voz. Durante la década de los 40 hubo también una gran actividad en la codificación por modulación de impulsos (PCM).

Gracias a la flexibilidad de los sistemas digitales se pudo experimentar con formas más sofisticadas de representación de la voz, dando lugar a finales de los cincuenta al modelo lineal de producción de voz.

El surgimiento de la tecnología VLSI durante los 60 y 70 permitió nuevas soluciones al problema de la codificación de la voz como la basada en la transformada rápida de Fourier (FFT) de Flanagan y Golden.

Desde entonces, dentro de este campo se ha desarrollado un interés creciente sobre las tecnologías de codificación de voz, fomentado en gran medida por el desarrollo de la telefonía móvil, la disminución de la capacidad disponible para las comunicaciones en ciertos mercados, la aparición en los últimos años de nuevos escenarios y aplicaciones como la voz sobre IP y la cada vez más imprescindible exigencia de alta calidad.

Uno de los codificadores de audio digital más conocido y empleado en redes de todo el mundo es el GSM Full Rate, usado en el sistema digital de telefonía móvil GSM. Para la época en que se desarrolló (principios de los años 90) suponía un buen compromiso entre calidad y complejidad computacional. Sin embargo, a lo largo de estos años se han venido desarrollando estándares más modernos que mejoran la calidad del audio codificado, por lo que éste códec está siendo sustituido gradualmente por otros nuevos como el GSM EFR (Enhaced Full Rate) y el AMR (Adaptative Multi-Rate) que ofrecen una mayor calidad de sonido con igual o inferior tasa binaria. En general, los codificadores de voz asumen voz de banda estrecha (muestreo a 4 u 8 kHz) pero se han desarrollado también extensiones que permiten muestrear a mayor frecuencia (16 y 32 kHz).

1.2 Objetivos

Este proyecto tiene como objetivo cumplir las necesidades de una empresa que desea sustituir el uso del codificador GSM Full Rate por uno con un mejor comportamiento frente al ruido y que ofrezca una mayor calidad manteniendo o bajando la tasa binaria empleada durante la codificación. Para ello, implementaremos un códec de voz propietario para aplicaciones en las que la voz se graba en banda ancha (frecuencias de muestreo de 16 kHz o mayores) y en las que la relación señal a ruido es baja.

Se tendrá en cuenta la capacidad de codificar razonablemente efectos de sonido y ruidos.

La implementación se realizará en un sistema embebido de reducida capacidad computacional y sin unidad de cálculo en punto flotante. Deberá además poder mantenerse dos codificaciones simultáneas en tiempo real.

Para ello, partiremos de algunos codificadores estándares de los cuales se dispone del código fuente en C. Se realizarán sobre ellos distintas pruebas de cara a evaluar su comportamiento y rendimiento ante distintos escenarios (baja relación señal a ruido, amplitud reducida, codificación de ruidos y efectos de sonido…).

Una vez elegido el codificador cuyos resultados satisfagan en mayor medida nuestros requisitos, implementaremos el códec en tres etapas:

En primer lugar desarrollaremos una implementación sin requisitos especiales en cuanto a funcionamiento. En este punto debemos realizar las adaptaciones necesarias para mejorar el comportamiento del códec frente a grabaciones en las que la relación señal a ruido es baja.

En segundo lugar desarrollaremos una implementación que opere en un sistema embebido desprovisto de unidad de punto flotante. En este punto debemos adaptar el códec anteriormente desarrollado para su correcto funcionamiento en la unidad y optimizarlo con el objetivo de que se puedan correr dos codificaciones independientes en tiempo real.

En tercer lugar, realizaremos distintas pruebas que permitan comprobar el buen funcionamiento del sistema de codificación desarrollado, tanto para la primera implementación como para la que opera en la unidad empotrada.

Desarrollaremos también un “códec driver” para el Audio Compression Manager de Microsoft con el objetivo de poder reproducir y decodificar en Windows los ficheros generados por el codificador tanto en Linux como en el sistema embebido. Los ficheros codificados en Windows deberán también poder ser decodificados desde Linux y el sistema embebido. Veremos que para ello deberemos modificar el códec desarrollado en primera instancia generando una nueva versión del mismo ahora si compatibles entre sí.

1.3 Planificación

El proyecto se ha realizado en aproximadamente un año de duración. Durante este tiempo se han llevado a cabo distintas tareas de cara a satisfacer las tres entregas planificadas con el cliente:

- **Primera entrega:** Evaluación de códecs candidatos (Julio 2008).
- **Segunda entrega:** Presentación de la primera versión del codificador para Linux y sistema embebido ATVS-1.0 (Diciembre 2008).
- **Tercera entrega:** Presentación códec-driver ACM para Windows. Fue necesario readaptar el códec anteriormente entregado para obtener versiones totalmente compatibles ATVS-2.0 (Mayo 2009).

En la siguiente tabla podemos ver cada una de las tareas con el momento de su realización. Marcamos en azul las distintas entregas planificadas con los clientes:

Id.	Tarea	Comienzo	Fin
0	Lectura documentación codificación audio	Abril	Mayo
1	Evaluación códecs candidatos	Mayo	Junio
2	Generación documentación evaluación. Presentación evaluación códecs	Junio	Julio
3	Estudio Speex 1.2b3 y modificación	Julio	Octubre
4	Adaptación mipsel ATVS-1.0	Octubre	Diciembre
5	Pruebas. Generación manuales y documentación. Presentación ATVS-1.0	Diciembre	Diciembre
6	Adaptación formato WAV	Enero	Enero
7	Estudio documentación ACM	Febrero	Febrero
8	Desarrollo códec-driver ACM	Febrero	Abril
9	Readaptación limitaciones ACM → ATVS-2.0	Abril	Abril
10	Pruebas. Generación manuales y documentación. Presentación ATVS-2.0 y códec driver ACM	Abril	Mayo

Tabla 1. Planificación tareas del proyecto

1.4 Organización de la memoria

El cuerpo de la memoria está dividido en tres apartados claramente diferenciados:

ESTADO DEL ARTE: Introduciremos conceptos sobre el sonido y la voz, una visión de los distintos enfoques en codificación de voz, sus aplicaciones y los distintos sistemas y estándares existentes.

EVALUACIÓN DE CODIFICADORES: Describiremos los códecs objeto de evaluación, las pruebas llevadas a cabo y las conclusiones extraídas de ellas, que nos permitirán elegir un códec candidato para formar la base del códec propietario

DESARROLLO VERSIONES CÓDEC: Describiremos las adaptaciones llevadas a cabo en el códec base para obtener por un lado el códec propietario funcional en Linux y por el otro, la correcta ejecución en los tiempos requeridos en la unidad empotrada. Detallaremos también el desarrollo del códec-driver ACM y la readaptación de los primeros para obtener la compatibilidad entre los tres. Incluye también las pruebas de calidad realizadas para comprobar el efecto de las modificaciones introducidas y las pruebas de requisitos de tiempos.

En último lugar incluiremos las conclusiones extraídas del proyecto, las líneas propuestas como trabajo futuro, las referencias consultadas, glosario y anexos.

2 ESTADO DEL ARTE

2.1 Fundamentos de la percepción y producción de audio y voz

Este apartado tiene como finalidad conocer el sonido desde un punto de vista físico y entender tanto la manera de percibir los sonidos por los seres humanos como de producir aquellos que nos permiten comunicarnos.

Para tratar el sonido, bien de manera analógica o bien de forma digital debemos conocerlo físicamente. El análisis de la percepción de los sonidos (la psicoacústica) nos permite extraer conocimientos esenciales a la hora de procesar digitalmente los sonidos de forma más eficaz, lo que tiene su aplicación en codificación de audio y reconocimiento automático de sonidos y voz.

Por otro lado, el análisis de los mecanismos de producción de voz nos permitirá extraer conocimientos esenciales para procesar y reconocer un tipo esencial de sonido: la señal de voz. Estos conocimientos tienen su aplicación en codificación de voz, reconocimiento automático de voz y en la síntesis de voz.

2.1.1 El sonido

Desde un punto de vista físico el sonido es una perturbación de la presión en un medio que se transmite en una forma de onda de presión a través del medio. Aunque generalmente dicho medio será el aire (medio gaseoso), el sonido puede transmitirse también por medios líquidos (como el sónar o la comunicación entre ballenas) y por medios sólidos (como la transmisión de ruido a través de paredes o cristales) donde no podemos hablar de transmisión por presión sino de vibraciones de estructuras sólidas.

La unidad de presión en el Sistema Métrico Internacional es el Pascal donde

$$1\ Pa = 1\ N/m^2$$

La velocidad de transmisión del sonido en el aire se puede calcular como:

$V = \sqrt{\frac{\gamma RT}{M}}$, donde

$\gamma = constante\ adiabática\ (1.4\ para\ el\ aire)$
$R = constante\ gas\ (8.31\ JK^{-1}mol^{-1})$
$T = temperatura\ absoluta\ (K)$
$M = peso\ molecular\ (kgmol^{-1})$

La velocidad de transmisión del sonido en un gas depende por tanto exclusivamente de dos factores:

- Del peso molecular medio del gas (que es proporcional a la densidad del gas), de forma inversamente proporcional a la raíz cuadrada. Al disminuir la densidad del gas el sonido viaja más rápidamente.
- De la temperatura del gas, de forma directamente proporcional a la raíz cuadrada. (explica que la velocidad del sonido sea más lenta a gran altitud). La variación de la velocidad del sonido debida a la temperatura es despreciable con las variaciones de temperatura que experimentamos en la vida diaria.

Si asumimos que el aire es 21 % O_2, 78 % N_2, 1 % A_r su peso molecular será: $M = 21\% \times 16 \times 2 + 78\% \times 14 \times 2 + 1\% \times 18 \times 1 = 2.87 \times 10^{-2} kgmol^{-1}$ y la velocidad de transmisión será:

$$V = \sqrt{\frac{\gamma RT}{M}} = \sqrt{\frac{1.4 \times 8.31T}{2.87 \times 10^{-2}}} = 20.1\sqrt{T}$$

$$a\ 20ºC\ ;\ T = 293K;\ V = 20.1\sqrt{293} = 344\ ms^{-1}$$

La velocidad de transmisión del sonido en un gas es esencialmente igual para todos los sonidos, por lo que puede resultar más interesante analizar propiedades que nos permitan diferenciar unos sonidos de otros como el nivel de presión del sonido o la periodicidad.

El nivel de presión del sonido se mide, para el sistema métrico internacional, en Pascales (Pa). Sin embargo, dado el margen de variación de presión del sonido que es capaz de manejar el oído humano, conviene expresar el nivel de presión del sonido en otras unidades.

El oído humano no es capaz, en general, de detectar variaciones de presión inferiores a 2×10^{-5} Pascales r.m.s. ni siquiera para las frecuencias en las que es más sensible. Las mayores variaciones de presión que es capaz de soportar el oído humano, en general, sin llegar a sentir dolor son del orden de 10^2 Pascales r.m.s. Para representar ese rango dinámico de valores tan amplio de niveles de presión se emplean las unidades logarítmicas conocidas como *decibelios* (dB) y definidas como:

$$SPL = 20\log_{10}\left(\frac{p}{p_0}\right), p_0 = 2 \times 10^{-5} Pa = 20\mu Pa$$

En ocasiones se emplea también la intensidad de la onda sonora (*I*), definida como la potencia por unidad de área en la onda sonora. Dado que la intensidad de la onda sonora es proporcional a la presión al cuadrado, podemos expresar *SPL* en función de *I* como:

$$SPL = 10\log_{10}\left(\frac{I}{I_0}\right), I_0 = 10^{-12}\ W/m^2$$

En cualquier caso, el rango dinámico del oído en dB excede los 130dB con el umbral mínimo (sonidos apenas perceptibles) en 0 dB y el umbral máximo (umbral del dolor) en 134 dB.

2.1.2 Percepción del sonido

En el apartado anterior hemos descrito las características físicas de los sonidos, sin embargo, lo que realmente caracteriza al audio es que son sonidos destinados a ser percibidos mediante el sentido del oído humano. Estudiando el funcionamiento del oído humano podremos extraer nuevas conclusiones de la psicoacústica, disciplina científica que se encarga de estudiar estadísticamente la sensación que producen los sonidos en los humanos.

2.1.2.1 Funcionamiento del oído

El sentido del oído humano depende de una serie de procesos acústicos, mecánicos, hidráulicos nerviosos y mentales que tienen lugar en el oído y cerebro humanos. El oído es lo primero que se encuentra el sonido en su camino hacia la percepción. Estructuralmente se descompone en tres partes: oído externo, que pre-procesa y acondiciona la señal acústica, oído medio, que se encarga de transformar esa señal acústica en nerviosa y oído interno, parte en la que se realiza el procesamiento sensorial del sonido.

- **Oído externo**: Se encarga de recoger el sonido y conducirlo a través del canal auditivo hacia el tímpano haciendo que vibre siguiendo las ondas de presión produciendo a su vez cierto filtrado del mismo. El conducto auditivo se puede modelar como un tubo de longitud 2 cm abierto por un único lado. Un tubo de estas características tiene una frecuencia de resonancia de 4 kHz, frecuencia para la que el oído humano tiene mayor sensibilidad.

- **Oído medio:** Convierte los movimiento de aire del canal auditivo en movimientos del fluido de la cóclea. Para ello utiliza una estructura ósea formada por el tímpano, martillo, yunque, estribo y ventana oval que transforma grandes movimientos de aire con poca

energía en pequeños movimientos del fluido de la cóclea de mayor energía. Realiza por tanto una labor de adaptación de impedancias, que es máxima para frecuencias de alrededor de 1 kHz. Alrededor de esta frecuencia la transmisión de energía será máxima y contaremos por tanto con una buena sensibilidad.

- **Oído interno:** Formado por la cóclea, realiza un análisis espectral de las vibraciones de su fluido interno transformándolas en impulsos nerviosos que se envían a través del nervio auditivo hacia el cerebro. La cóclea es un tubo largo y fino que se enrolla sobre sí mismo de forma espiral unas dos veces y media. Dentro de ella hay tres canales rellenos de líquidos denominados escalas:
  - **La escala vestibular:** que está en contacto directo con el oído medio a través de la ventana oval.
  - **La escala media:** que está separada de la escala vestibular por una membrana muy fina conocida como membrana de Reissner.
  - **La escala timpánica:** que está separada de la escala media por la membrana basilar y que está en contacto con la ventana circular.

Los efectos más importantes funcionalmente son los movimientos del fluido a lo largo de la membrana basilar. Ésta tiene unos 32 mm de longitud, es relativamente ancha cerca de la ventana oval, y llega a ser sólo de un tercio de dicha anchura en el otro extremo de la cóclea, el *ápex*, donde la escala timpánica está en contacto directo con la escala vestibular a través del *helicotrema*. La membrana basilar sirve como base al órgano de Corti, que contiene las células sensoriales capilares que transforman movimientos del fluido en impulsos eléctricos que van al nervio auditivo.

El funcionamiento de la cóclea sigue el siguiente esquema:

- El fluido de la cóclea se desplaza en la escala media y vestibular por los movimientos de la ventana oval producidos por el oído medio.
- Estos desplazamientos de fluido se compensan con movimientos de la membrana basilar, o, para frecuencias muy bajas, por desplazamientos de fluido hacia la escala timpánica a través del helicotrema.
- Finalmente el movimiento de fluido en la escala timpánica se compensa mediante movimientos de la ventana circular, localizada en la base de la escala timpánica.

Georg von Békésy estudió experimentalmente los movimientos de los fluidos en el oído interno y llegó a una verdaderamente remarcable conclusión, ya propuesta con anterioridad por Helmholtz: **la cóclea actúa como un analizador espectral**:

- Los sonidos de una frecuencia particular hacen que la membrana basilar vibre con amplitud máxima en un punto particular entre la ventana oval y el helicotrema.
  Los sonidos de bajas frecuencias producen vibraciones con amplitud máxima cerca del ápex de la cóclea, cerca del helicotrema. Los sonidos de altas frecuencias, en cambio, producen vibraciones con amplitud máxima cerca de la ventana oval.

2.1.2.2 Procesamiento mental del sonido

El proceso auditivo no termina en el análisis espectral realizado por el oído interno. Ésta información se transmite al cerebro que se encarga de analizarla para:

- Localizar la fuente sonora
- Separar las fuentes sonoras en el caso de que exista más de una.
- Segmentar los sonidos
- Clasificar e identificar los sonidos
- En el caso de que los sonidos sean compatibles con la señal de voz:
    - Combinar los sonidos para formar sílabas, palabras y frases.
    - Realizar un análisis semántico y pragmático de lo entendido.
    - Integrar los resultados con el contexto discursivo y sensorial.
- Y por último, producir una respuesta apropiada en cada situación.

2.1.2.3 Sensación Sonora y umbral auditivo

La energía o potencia de un sonido se mide físicamente mediante el nivel de presión sonoro (*sound pressure level*, SPL) en decibelios. La percepción de la energía o potencia de un sonido depende del nivel de presión sonoro, pero la relación entre ambas no es estrictamente lineal, sino que depende de forma compleja del nivel de presión sonoro y de la frecuencia.

El término de *sensación sonora* (*loudness*) se introdujo en la década de 1920 para describir la intensidad sonora percibida y se define como el nivel de presión acústica de un tono de 1kHz que se percibe tan fuerte como el sonido cuya sensación sonora queremos medir. Se mide con sonidos producidos por fuentes lejanas e incidentes frontalmente y su unidad de medida es el fon (phon).

La experimentación ha permitido establecer una serie de curvas que indican la sensación sonora que experimentan, en media, los sujetos al presentarles un tono con una determinada frecuencia y nivel de presión sonoro. Estas curvas cambian individualmente para cada persona y son variables con la edad.

Del estudio de estas curvas se han extraído observaciones interesantes:

- Para bajas frecuencias una gran variación en el nivel de presión sonoro produce una pequeña variación de sensación sonora.
- Para frecuencias muy altas el oído se vuelve sordo de forma brusca.
- Alrededor de 4kHz existe un máximo de la sensación sonora debido a la frecuencia de resonancia del canal auditivo.

El umbral auditivo representa el nivel de presión sonoro más bajo que puede ser oído a cada frecuencia por el oído humano (se corresponde con la curva de sensación sonora de 0 fons). Es una curva extremadamente importante en codificación de audio ya que las componentes de una señal de audio que caigan bajo este umbral son inaudibles y no se necesitan codificar ni transmitir. Además el ruido introducido en una señal, por ejemplo el ruido de codificación, será inaudible siempre que se encuentre por debajo del umbral auditivo.
Dada su importancia en la codificación se ha estudiado ampliamente la dependencia del umbral auditivo con la frecuencia.

- Típicamente es de unos 50dB en 50Hz.
- Baja a casi 0dB en 500 Hz y se mantiene hasta los 2000 Hz.
- Entre 2 y 5 kHz baja por debajo de 0dB.
- Por encima de 5kHz hay picos y valles que varían mucho con el sujeto.
- A partir de 16kHz el umbral sube muy rápidamente.

En la siguiente imagen mostramos unas curvas de sensación acústica (en la izquierda) y una representación del umbral auditivo (derecha) basado en la siguiente aproximación: $A(f) = 3.64f^{-0.8} - 6.5e^{-0.6f^2} + 10^{-3}f^4$ donde $A(f)$ está en dB y la frecuencia f en kHz:

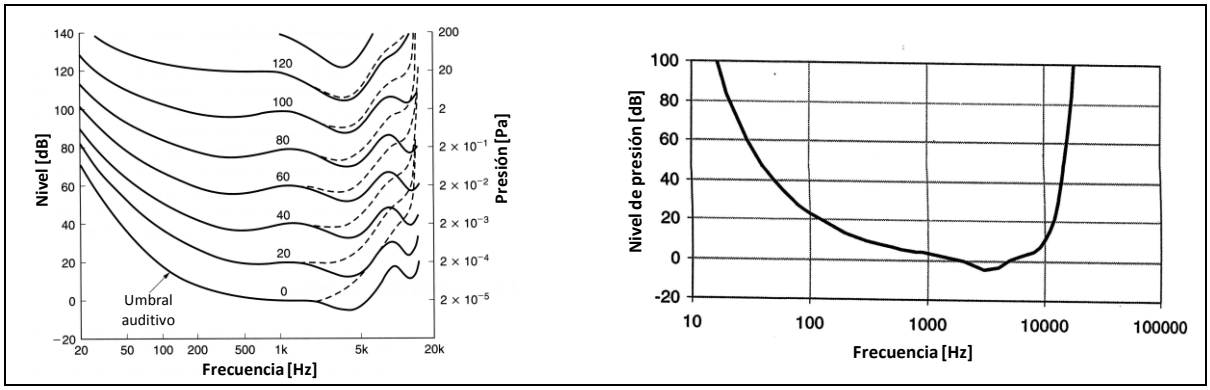


Figura 1. Curvas de sensación acústica y umbral auditivo

2.1.2.4 Enmascaramiento

El enmascaramiento es un fenómeno por el cual los sonidos débiles son anulados por otros más fuertes. Existen dos tipos de enmascaramiento: el simultáneo o frecuencial y el temporal.

Al que estamos más habituados es al simultáneo o frecuencial en el que el sonido enmascarador (un tono de alta energía, por ejemplo) excita la misma región de la membrana basilar de la cóclea que el sonido enmascarado (un tono próximo y en frecuencia y de menor energía, por ejemplo). Esto se produce por ejemplo cuando estamos hablando y un ruido no nos deja escuchar lo que dice nuestro interlocutor. El umbral de enmascaramiento en frecuencias indica el nivel de presión mínimo que debe tener una señal para ser audible en presencia del tono enmascarador.

Podemos calcular la relación entre señal y máscara (*signal to mask ratio)* como la diferencia de niveles de presión en decibelios entre una componente de una señal y el umbral de enmascaramiento en la frecuencia de dicha componente. Si el valor obtenido es negativo dicha componente será inaudible. Si por el contrario es positiva será audible, y tanto más cuanto mayor sea.

A modo de ejemplo, en la siguiente figura podemos observar dos señales que aparecen por debajo del umbral de enmascaramiento. Estas señales serán por tanto inaudibles:

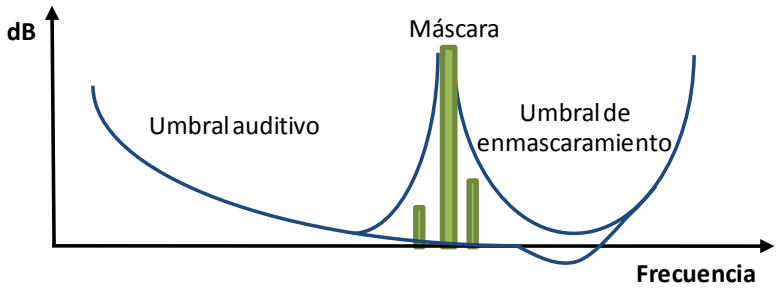


Figura 2. Ejemplo enmascaramiento frecuencial

El efecto del enmascaramiento en frecuencias se puede emplear en la codificación de igual manera que el umbral auditivo:

- Las señales enmascaradas totalmente no son audibles y por tanto no tienen por qué ser codificadas y transmitidas.
- Si el ruido queda por debajo del umbral de enmascaramiento tampoco será audible y no supondrá un problema.

Por otro lado el enmascaramiento temporal se produce entre sonidos que no están presentes simultáneamente y se explica por la inercia de los elementos físicos involucrados en el proceso auditivo.

- Si el tono enmascarado se empieza a producir antes de que comience la señal enmascarada estamos hablando de pre-enmascaramiento. Comienza varias decenas de milisegundos antes y es fuerte sólo varios milisegundos después.
- Si por el contrario se deja de producir después de que termine la señal enmascaradora estamos hablando de post-enmascaramiento. Puede durar hasta varios cientos de milisegundos y es un fenómeno más fuerte que el pre-enmascaramiento.

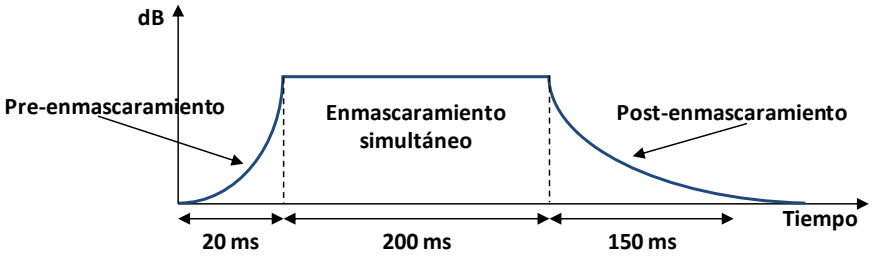


Figura 3. Enmascaramiento temporal

Estudiando las curvas de enmascaramiento frecuencial se descubrió la existencia de un pequeño margen de frecuencia alrededor de la frecuencia enmascaradora en el que el umbral de enmascaramiento en lugar de decrecer es plano. A este rango de frecuencias se le denomina **banda crítica**. Esto es válido para cualquier señal enmascarante y enmascarada aunque con distinto nivel de enmascaramiento según el caso.

Esta conclusión llevó a un modelo auditivo como un banco de filtros paso banda que se superponían en frecuencia y con anchos de banda iguales a las bandas críticas. Aunque existen varias expresiones y fórmulas para las bandas críticas, la más aceptada es una expresión que describe la variación de las bandas críticas como una función de la frecuencia central enmascaradora: $\Delta f = 25 + 75[1 + 1.4 f_c^2]^{0.69}$ donde f_c está en kHz y Δf en Hz.

Del enmascaramiento sabemos por tanto que:

- Su valor máximo se produce cerca de la frecuencia de la señal enmascaradora.
- Se mantiene constante en una pequeña banda (banda crítica) alrededor de la frecuencia central de la señal enmascaradora.
- Decrece rápidamente al alejarnos de la frecuencia central de la señal enmascaradora
- La forma de sus curvas depende de forma compleja de la señal enmascaradora, frecuencia y nivel.
- Comienza algunos milisegundos antes de que comience la señal enmascaradora (pre-enmascaramiento) y puede terminar cientos de milisegundos después de que termine la señal enmascaradora (post-enmascaramiento).

2.1.3 Producción de voz. Modelos

La voz, siendo el medio de comunicación más utilizado entre las personas, es una de las señales de audio más importantes. Por ello no es de extrañar el interés existente en el procesamiento digital de la señal de voz manifestado especialmente en sistemas de comunicación como la telefonía (tradicional, móvil o de voz sobre IP) o la comunicación con ordenadores.

La señal de voz es una señal de audio y como tal, se puede tratar como audio genérico. Sin embargo podemos explotar el conocimiento sobre su percepción y, sobre todo, su producción para procesarla de manera más eficiente. Dedicaremos entonces un pequeño espacio en este apartado a analizar los mecanismos de producción de voz que nos permiten aumentar esta eficiencia y a presentar distintos modelos digitales de producción de voz.

La señal hablada se produce cuando expelimos el aire de los pulmones a través de la tráquea. En el habla existen segmentos de naturaleza distinta:

- **Sonidos sonoros:** Presentan carácter periódico. En ellos, las cuerdas vocales, que están ubicadas en la laringe se hallan en tensión y vibran cuando las atraviesa el flujo de aire proveniente de los pulmones.

- **Sonidos sordos:** Tienen apariencia ruidosa. En ellos, las cuerdas vocales están en relajación, y el flujo de aire las atraviesa libremente. El sonido se produce por la turbulencia de aire generada en una constricción del tracto vocal.

Los órganos más importantes del aparato fonador humano son:

- **Tracto Vocal:** comienza en las cuerdas vocales (glotis) y termina en los labios. Está formado por la laringe y la cavidad oral. Mide unos 17 cm de longitud de media en el caso de los hombres y tiene una sección transversal que va desde 20 cm^2 a cero en el caso de cierre completo.
- **Tracto Nasal:** Comienza en el velo y termina en la nariz. El velo puede bajar conectando acústicamente el tracto nasal con el tracto vocal para producir sonidos nasales.

En la producción de sonidos hablados, la laringe excita estas cavidades, produciendo determinadas frecuencias de resonancia, denominadas formantes. Los distintos fonemas se realizan cambiando la forma del tracto vocal (a través del movimiento de los articuladores como lengua, labios, dientes, etc.), caracterizando a los distintos fonemas de una configuración de formantes distinta (y también un modo de excitación distinto).

Los formantes desempeñan por tanto un papel fundamental en la diferenciación de los sonidos. La detección de estas frecuencias de resonancia se realiza en la envolvente espectral, constituyendo los formantes los máximos relativos de dicha envolvente.

Para que un modelo fuera completo debería incluir los efectos de variación con el tiempo de la forma del tracto vocal, las pérdidas debidas a la transmisión de calor, la fricción viscosa en las paredes del tracto vocal, la falta de rigidez del tracto vocal, la radiación del sonido en los labios y/o la nariz, el acoplamiento acústico nasal y la excitación del tracto vocal. Sin embargo se realizan importantes simplificaciones dado que la formulación y solución de las ecuaciones que definirían los modelos más completos serían excesivamente complejas.

Un modelo muy utilizado es el que considera el tracto vocal como un tubo de sección transversal constante por tramos y sin pérdidas como el de la siguiente figura:

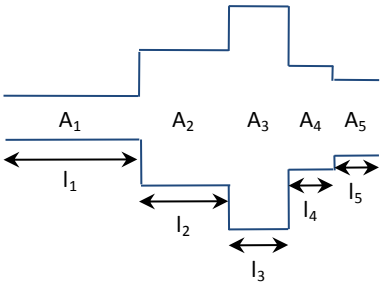


Figura 4. Modelo de tubo de sección constante por tramos

Las áreas de las secciones transversales se toman de tal forma que se aproxime la forma del tracto vocal. Si se elige un número de tubos suficientemente grande, de forma que los tubos sean de longitud suficientemente pequeña, la aproximación resulta razonable. De esta forma podemos conseguir que las frecuencias de resonancia del tubo coincidan con las del tracto vocal.

Este modelo considera tubos sin pérdidas, por lo que los anchos de banda de las resonancias obtenidas son menores que los reales. No se tienen en cuenta las pérdidas por vibración de las paredes, por fricción viscosa y por transmisión de calor. Estas pérdidas, sin embargo, se podrán incluir en los modelos de la glotis y la radiación en los labios.

Si consideramos un tubo como el anterior con *N* secciones de longitud constante, $\Delta x = l/N$, la onda sonora recorrerá la longitud de un tubo en $\tau = x/c$ (*c* = velocidad del sonido).

Si consideramos ahora la respuesta de este sistema a un impulso en la glotis, $\delta(t)$:

- El impulso se propaga inicialmente hacia la derecha.
- Al llegar a cada unión de dos tubos parte se continúa propagando hacia la derecha (se presentará en los labios con un retardo $N\tau$) y parte se refleja a la izquierda. Esta parte reflejada a la izquierda podrá volverse a reflejar en otra unión llegando a los labios con un retardo adicional de 2τ.
- Este proceso puede repetirse más veces llegando impulsos con retardos adicionales de $2k\tau$.

Tras estas consideraciones, la respuesta al impulso del sistema puede escribirse como:

$$h_r(t) = \alpha_0\delta(t - N\tau) + \sum_{k=1}^{\infty} \alpha_k\delta(t - N\tau - 2k\tau)$$

Tomando la transformada de Fourier de la respuesta al impulso anterior obtenemos:

$$h_r(t) = \alpha_0\delta(t - N\tau) + \sum_{k=1}^{\infty} \alpha_k\delta(t - N\tau - 2k\tau) \overset{FT}{\leftrightarrow} H_r(j\Omega) = \sum_{k=0}^{\infty} \alpha_k e^{-j\,\Omega(N\tau + 2k\tau)}$$

$$= e^{-j\,\Omega N\tau} \sum_{k=0}^{\infty} \alpha_k e^{-j\,\Omega 2k\tau}$$

Esto nos permite dividir el sistema en un retardo constante Nτ más un sistema lineal e invariante con respuesta en frecuencia:

$$H(j\Omega) = \sum_{k=0}^{\infty} \alpha_k e^{-j\,\Omega 2k\tau}$$

El retardo se suele ignorar por no tener consecuencias prácticas en muchas aplicaciones. Además, si la entrada al sistema (la excitación) es de banda limitada a frecuencias por debajo de p/2t podemos muestrear la entrada con periodo $T = 2\tau$, filtrar la señal muestreada con un filtro con respuesta al impulso: $h[n] = \begin{cases} \alpha_n & n \geq 0 \\ 0 & n < 0 \end{cases}$, y reconstruir la señal de salida a partir de la del filtro digital.

De esta forma modelamos el tracto vocal como la siguiente secuencia, en la que el filtro digital tiene respuesta en frecuencia y función de transferencia dadas por:

$$H\left(e^{j\omega}\right) = \sum_{k=0}^{\infty} \alpha_k e^{-j\,\omega \mathrm{k}}\ ;\ H(z) = \sum_{k=0}^{\infty} \alpha_k z^{-\mathrm{k}}.$$

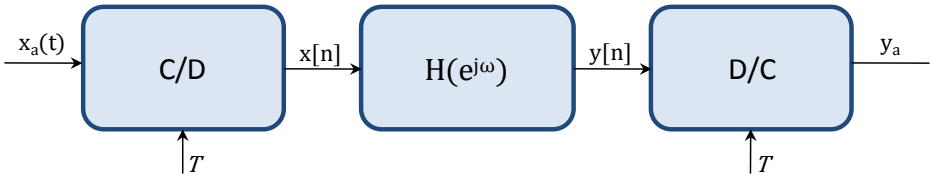


Figura 5. Modelo del tracto vocal

En la práctica resulta más sencillo trabajar con señales digitales, por lo que en lugar de emplear el modelo que incluye conversión de analógico a digital y de digital a analógico se considera únicamente el filtro digital.

Para llegar a una expresión más compacta en función de los componentes de reflexión, vemos que el tubo completo se puede representar como un filtro digital con la siguiente estructura:

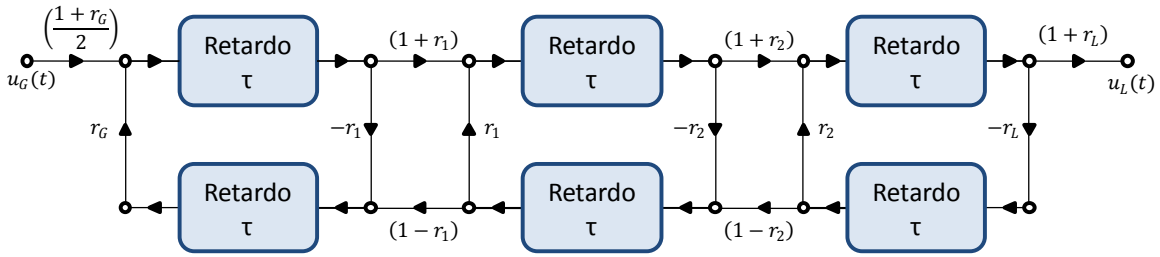


Figura 6. Estructura filtro digital

Analizando una de las secciones que forman este filtro, y considerando posteriormente N secciones conectadas en cascada llegamos a la conclusión de que la función de transferencia (ignorando el retardo) se puede escribir de la forma:

$$H(z) = \sum_{k=0}^{\infty} \alpha_k z^{-\mathrm{k}} = \frac{G}{1 - \sum_{k=0}^{\mathrm{N}} \alpha_k z^{-\mathrm{k}}}$$

Esta función de transferencia cuenta con N polos y ningún cero. Al tener N polos puede haber como máximo N/2 pares de polos conjugados, que corresponden a las resonancias del tubo (los formantes del tracto vocal). Esto no es del todo consistente con el funcionamiento del tracto vocal. El tracto vocal no permanece fijo con el tiempo, sino que varía para producir los diferentes sonidos. Sabemos también que para las nasales el acoplamiento de la cavidad vocal introduce ceros en la función de transferencia. Además, también debemos incorporar un modelo digital de radiación en los labios y uno de excitación del tracto vocal.

A continuación mostramos el modelo digital de producción de voz que combina en un único modelo los sistemas lineales e invariantes del pulso glotal, el tracto vocal y el modelo de radiación:

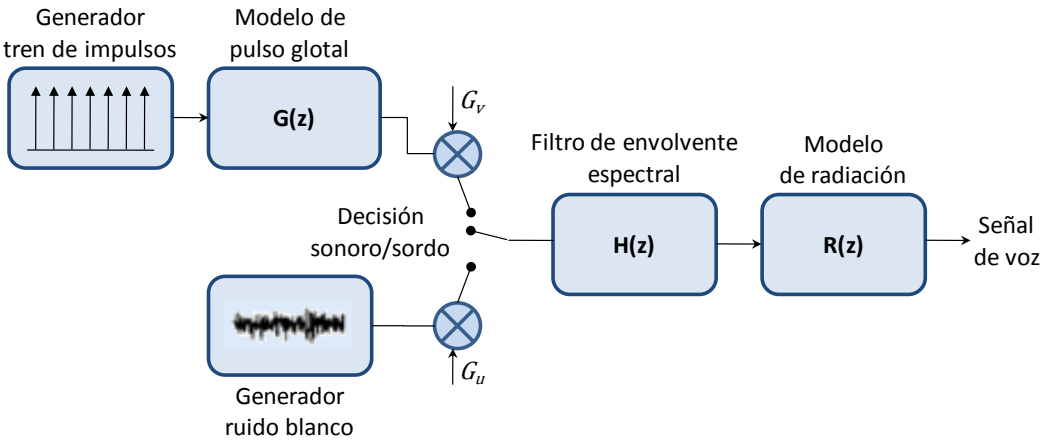


Figura 7. Modelo digital de producción de voz

La función de transferencia de este modelo es: $H(z) = G(z)\,V(z)R(z)$

El pulso glotal describe las diferencias en la calidad de la voz afectada por la variación de los pliegues de las cuerdas vocales al hablar. En términos de mecánica, un pulso glotal se produce por el movimiento de tejido en la región de las cuerdas vocales y el hueco entre ellas, lo que comúnmente conocemos como glotis. La frecuencia del pulso glotal es el resultado de la vibración de la resonancia de las cuerdas vocales contra la laringe.

El modelo de radiación en los labios $R(z)$ reproduce el efecto de la impedancia de radiación que el medio opone a la salida del habla desde la boca. Un modelo de radiación aproximado viene dado por la ecuación $R(z) = R_0(1 - z^{-1})$

El modelo excitación del tracto vocal se modela por tanto como un tren de impulsos glotales para sonidos sonoros y como ruido aleatorio para sonidos sordos:

- El generador de impulsos genera un tren de impulsos unitarios separados entre sí el periodo de la frecuencia fundamental.
- Este tren de impulsos ataca un sistema lineal e invariante cuya respuesta al impulso es la forma del pulso glotal:

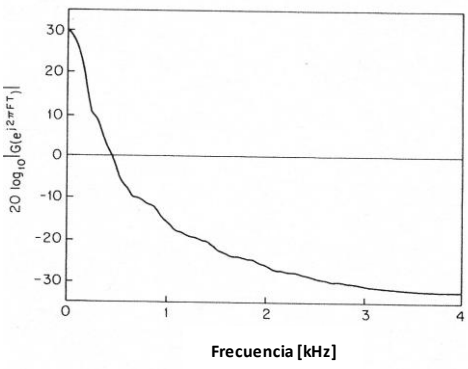


Figura 8. Pulso glotal

Este modelo nos proporciona información acerca de la estructura de la señal de voz, así como sobre la forma en que se codifica la información de los fonemas en la señal de voz. Este modelo cuenta con las siguientes limitaciones:

- **Variación en el tiempo:** La cuasi-estacionariedad de la señal de voz aproximadamente correcta en sonidos sin transiciones. Con transiciones el modelo es peor, pero todavía adecuado.
- **Falta de ceros en la función de transferencia**: Esto supone una deficiencia para nasales. Es posible, aunque inusual, añadir ceros al modelo. Otra forma de compensar esta limitación es añadiendo más polos de los necesarios.
- **Falta de excitación combinada:** La dicotomía sordo/sonoro no es tan estricta y también hay fricativas sonoras.

A pesar de estas limitaciones, este modelo supone la base del análisis de predicción lineal de voz, que a su vez es la base de todas las técnicas modernas de codificación de voz. Sirve también de base para modelos de síntesis de voz y es empleado, aunque no directamente, en técnicas de reconocimiento de voz.

2.2 Codificación de audio y voz

Como comentamos en el apartado anterior de este documento, cualquier sonido en la naturaleza tiene características similares. Desde que vivimos en una era computacional, deseamos disponer de dicha información en forma digital de modo que podamos grabar, procesar, transmitir y reproducir dicha información de forma digital.

Un codificador de audio digital, o códec (por encoder-decoder de su denominación inglesa) es un sistema o dispositivo que toma como entrada señales de audio analógico y las transforma temporalmente en una representación digital conveniente de dicha señal de entrada. Esta transformación tiene lugar en la etapa de codificación del códec.

Una vez que tenemos la señal representada como series de números podemos almacenarla, procesarla o transmitirla. En algún punto desearemos poder escuchar de nuevo dicha señal de audio. Para ello necesitamos transformar de nuevo la señal desde su representación digital a una señal analógica. Dicha transformación inversa desde la representación digital a la analógica tiene lugar en la etapa de decodificación del códec (o decodificador).

Cabe destacar que la primera etapa de la cadena de codificación de audio la forma la fuente de sonido y la última el oído humano. Ambas juegan un papel clave en el diseño del codificador. Aplicando los conocimientos de la psicoacústica podremos reducir la cantidad de información presente en nuestra representación a aquella que es relevante para nuestra percepción. Si además podemos desarrollar un buen entendimiento de la fuente podremos optimizar la forma en que representamos la señal de audio (podremos usar una descripción más compacta del sonido).

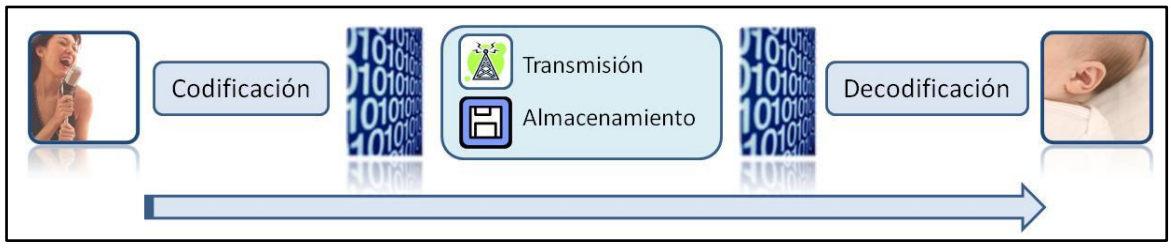


Figura 9. Cadena de codificación

A la hora de obtener una representación digital de una señal de audio, entran en juego ciertos compromisos a la hora de producir dicha transformación. En general desearemos maximizar la calidad percibida, pero también minimizar la cantidad de información necesaria para representar

dicha señal. El reto a la hora de diseñar un codificador de audio es por tanto, buscar un equilibrio entre ambos objetivos enfrentados manteniendo a su vez un coste aceptable del sistema.

Algunos de los factores más importantes que debemos tomar en consideración a la hora de diseñar o evaluar un codificador son la fidelidad, la tasa de información, la complejidad o el retardo. El equilibrio entre estos factores vendrá determinado por las aplicaciones y tecnologías que deba soportar. La fidelidad o calidad global es el atributo más importante de cualquier sistema de codificación. Sin embargo, dependiendo de la aplicación de destino la consideración de "calidad aceptable" varía significativamente. Lo que es aceptable en aplicaciones en las que únicamente se requiere inteligibilidad del mensaje como la llamada "calidad telefónica" puede no ser suficiente en muchas otras como en la distribución electrónica de música en la que desearíamos tener "calidad CD" o similar. Desafortunadamente una mayor fidelidad requiere tasas de información más altas, sistemas más complejos y mayores retardos.

Aunque los codificadores de audio genéricos pueden emplearse también para codificar voz, lo harán de manera menos eficiente que los dedicados. En cambio, un codificador de voz puede no producir buenos resultados a la hora de codificar audio distinto de la voz. De hecho uno de los objetivos secundarios de los codificadores de voz es que su comportamiento con señales que no sean de voz sea aceptable.

Los objetivos principales de la codificación de voz son los mismos que los de la codificación de audio, pero particularizados para la señal de voz:

- Conseguir una **buena calidad** en la voz decodificada.
- Conseguir una **reducida tasa binaria** (*bitrate*) para la voz codificada.
- Conseguir una **baja complejidad** en la codificación y decodificación.

Además, los codificadores de voz buscan otros objetivos más específicos:

- **Robustez frente a voces de distintos locutores e idiomas:** Tarea difícil por la diferencia entre hombres, mujeres, niños y a la distinta codificación de la información de los fonemas en distintas partes de la señal de voz dependiendo del idioma (por ejemplo hay lenguaje tonales como el chino en el que el tono es una característica de los fonemas que puede hacer cambiar el significado de una palabra).
- **Bajo retardo de codificación:** esencial en aplicaciones de comunicación en tiempo real como la telefonía.

2.2.1 Clasificación de codificadores de audio y voz.

Podemos hacer una clasificación general de los codificadores de audio y voz en función de las suposiciones que hacen acerca de la señal codificada:

- **Codificadores de forma de onda:** No hacen ninguna suposición sobre la señal codificada. Intentan reproducir lo más fielmente posible la forma de onda de la señal codificada. Los codificadores PCM forman parte de este tipo.

- **Codificadores perceptuales de audio:** Suponen que la señal codificada está destinada finalmente a ser escuchada por humanos. Emplean conocimientos sobre percepción del audio por parte de los humanos para reducir la tasa binaria con el menor impacto sobre la calidad percibida y sin intentar que la forma de onda se reproduzca fielmente. Ejemplos de este tipo de codificadores son MPEG-1 y MPEG-2 de audio.

- **Codificadores paramétricos de voz:** Al igual que los perceptuales suponen que la señal codificada está destinada a ser escuchada por los humanos y emplean conocimientos sobre la percepción de audio. Además emplean conocimientos de producción de voz y

todo ello sin intentar que ni la forma de onda ni cualquier sonido distinto de la voz se reproduzcan fielmente. Entre los codificadores de este tipo se encuentran el GSM-FR, GSM-HR, GSM-EFR, AMR y SPEEX.

En los siguientes subapartados de esta sección presentaremos estos tipos, profundizando especialmente en los codificadores paramétricos de voz, tipo a los que pertenecen los codificadores utilizados durante el desarrollo del proyecto:

2.2.2 Codificadores de audio de forma de onda

Los codificadores de forma de onda son aquellos que no hacen ningún tipo de suposición acerca de la naturaleza de la señal a codificar. Simplemente consideran que deben codificar una forma de onda de modo que la forma de onda decodificada se parezca lo más posible a la original. El método más sencillo de codificación de forma de onda es el Pulse Code Modulation (PCM).

Básicamente el codificador PCM transforma una señal analógica en una secuencia de bits mediante muestreo uniforme y cuantificación. El decodificador opera de manera inversa:

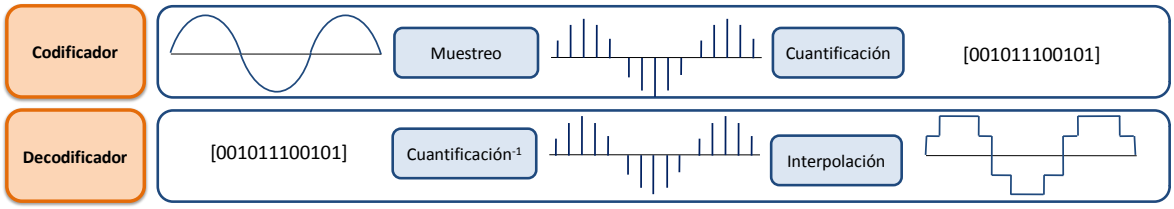


Figura 10. Esquema de funcionamiento codificación PCM

Existe una variante al PCM denominado DPCM (PCM diferencial) que explota el hecho de que las muestras de las señales de voz no cambian mucho de una muestra a otra, sino que están correladas. Basándose en la muestra anterior de la señal, predice el valor actual cometiendo un error de predicción. Al ser el margen dinámico de la señal de error menor que el de la señal original se cuantifica dicha señal de error empleando un número inferior de bits manteniendo el ruido de codificación bajo. Es una idea similar a la predicción lineal que veremos en profundidad en los codificadores paramétricos de voz.

2.2.3 Codificadores perceptuales de audio

Los codificadores perceptuales de audio explotan el conocimiento del oído humano para optimizar la codificación de todo tipo de señales de audio. Los ejemplos típicos de este tipo de codificadores son los estándares MPEG 1 Audio, MPEG-2 Audio y MPEG-2 AAC. El grupo MPEG (Moving Picture Experts Group) comenzó a trabajar en 1988 con el objetivo de desarrollar estándares de codificación de audio y video.

Los estándares MPEG-1 y 2 estandarizan codificación de video, audio, sincronización y otros aspectos que afectan tanto al audio como al video. Típicamente estos estándares incluyen tres partes: una primera que describe los aspectos del sistema completo (audio y video), una segunda que describe los elementos de codificación de vídeo y la tercera, que describe los elementos de codificación de audio. Las terceras parte del MPEG 1 y 2 también se conocen como MPEG-1 Audio y MPEG-2 Audio.
MPEG-1 Audio fue el primer estándar de codificación de audio de alta calidad con el objetivo de mantener la misma calidad de CD reduciendo la tasa binaria. Es válido para cualquier tipo de audio. El audio de entrada es audio codificado en PCM compatible con sistemas como CD

muestreados a 32, 44.1 o 48 kHz (aunque puede soportar otras). Dicha señal puede incluir uno o dos canales que se podrán codificar conjuntamente o de forma independiente. La tasa binaria de salida varía entre 32 y 224 Kb/s. El estándar incluye tres capas con distinto nivel de complejidad:

- MPEG-1 Audio Layer I: La más sencilla con tasas binarias entre 32 y 224 Kb/s por canal.
- MPEG-1 Audio Layer II: De complejidad media tiene tasas binarias entre 32 y 160 Kb/s por canal.
- MPEG-1 Audio Layer III: La más compleja con tasas binarias entre 32 y 160 Kb/s por canal. El tan conocido formato MP3 es una modificación de esta capa para menores frecuencias de muestreo.

MPEG-2 Audio es una familia de codificadores de audio que extiende la funcionalidad de MPEG-1 Audio en dos direcciones:

- Por un lado, proporcionar tasas binarias más bajas (y frecuencias de muestreo también inferiores). Entre ellos están MPEG-2 LSF, MPEG-2.5 y MP3.
- Por otro, soportar audio multicanal. Entre ellos están MPEG-2 BC (Backwards Compatible) y MPEG-2 AAC (Advance Audio Coding) que sin las ataduras de ser compatible hacia atrás consigue codificar 5.1 con solo 320 Kb/s con una calidad igual o superior a él. Para ello combina un banco de filtros de alta resolución, técnicas de predicción y codificación Huffman.

En audio multicanal existen otros estándares muy importantes como el Dolby-AC-3 o Dolby Digital que también se basan en conceptos de codificación perceptual de audio.

2.2.4 Codificadores paramétricos de voz

Los codificadores paramétricos de voz codifican una señal de voz destinada a ser escuchada por humanos. Pueden, por tanto, hacer uso de los conocimientos de producción de voz humana basándose en un modelo de producción de la misma. Los más avanzados emplean, además del modelo de producción, un modelo de la percepción de voz humana.

Estos codificadores son fundamentales para los sistemas de telefonía digital, en particular para los sistemas de telefonía móvil digital, donde el ancho de banda de transmisión es un recurso escaso.

Comenzaremos este apartado con una clasificación de estos codificadores. A continuación introduciremos la codificación de voz por predicción lineal, modelo fundamental de la mayoría de los codificadores de voz. Posteriormente veremos modificaciones sobre este sistema de codificador que darán lugar a los codificadores Regular Pulse Excitation Coders como el ETSI GSM 6.10 (GSM-FR).

Por último veremos los codificadores de predicción lineal excitados por código (CELP) y la modificación algebraica de los mismos (ACELP), utilizada en los codificadores de voz más avanzados como el ETSI GSM Enhanced Full Rate (GSM-EFR), ETSI Adaptive Multirate (AMR) o SPEEX.

Podemos realizar varias clasificaciones de los codificadores de voz atendiendo a distintos factores:

- Según la tasa binaria de la señal codificada:
  - **Tasa binaria alta** (> 15 Kb/s): Normalmente son codificadores de forma de onda.
  - **Tasa binaria media** (5 – 15 Kb/s) – Normalmente se trata de codificadores paramétricos híbridos.
  - **Tasa binaria baja** (2 – 5 Kb/s) – Normalmente son codificadores paramétricos puros.
  - **Tasa binaria muy baja** (<2 Kb/s) – Reconocimiento / síntesis de voz.

- En función del tipo de técnica de compresión:
  - **Paramétricos puros**: La voz está generada por un modelo con ciertos parámetros, que se estiman a partir de la voz y se transmiten al decodificador. Entre ellos se encuentra la codificación por predicción lineal (LPC). Son de baja tasa binaria y no funcionan bien con señales que no son de voz.
  - **Paramétricos híbridos**: Similares a los anteriores pero se añaden parámetros adicionales que intentan que la forma de onda recuperada sea lo más parecida a la original. Entre ellos están los codificadores de predicción lineal excitados por código (CELP). Tienen una tasa binaria media y funcionan razonablemente bien con señales que no son voz.

- En función de si se usa un único modo de codificación o varios con distintas tasas binarias:
  - **Codificadores mono-modo:** Emplean siempre el mismo tipo de codificación con la misma tasa binaria. Ejemplos: LPC, CELP.
  - **Codificadores multi-modo:** Emplean distintos tipos de codificación con diferente tasa binaria. El cambio del tipo de codificación puede estar controlado:
    - **Por la señal de voz:** Controlados por fuente
    - **Exteriormente:** Controlados por la red de comunicaciones.

    Un ejemplo de estos codificadores es el Adaptative Multi Rate (AMR).

2.2.4.1 Codificador de voz por predicción lineal (LPC)

La codificación por predicción lineal se basa en una simplificación del modelo digital de producción de voz que presentamos al finalizar el apartado anterior y que reproducimos a continuación:

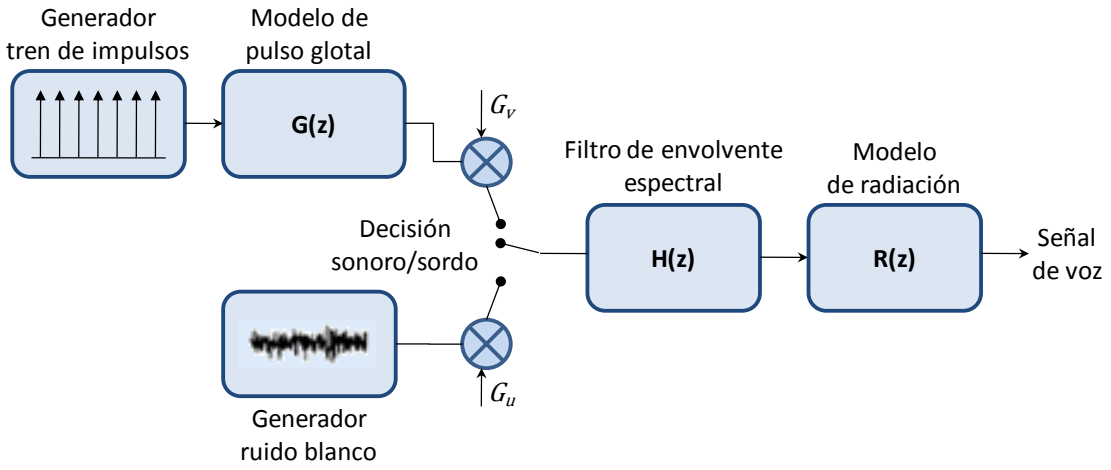


Figura 11. Modelo digital de producción de voz

Agrupando en un único filtro digital de síntesis con coeficientes variables en el tiempo todos los filtros digitales presentes en el modelo $(G(z), V(z)\ y\ R(z))$ y aplicando las ganancias (también variables con el tiempo) justo antes del filtro de síntesis llegamos al modelo simplificado de la figura, que es la base de la codificación por predicción lineal:

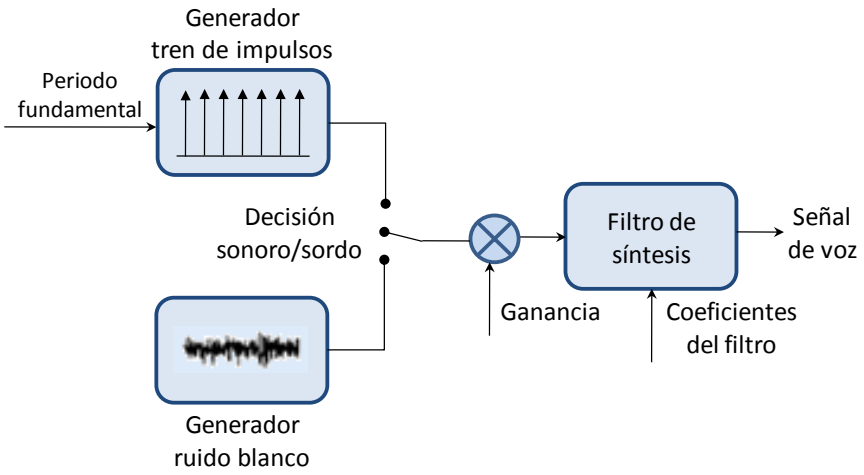


Figura 12. Modelo simplificado de producción de voz. Base codificación LPC

El codificador por predicción lineal, en lugar de transmitir la voz codificada en PCM, divide la voz en segmentos, obtiene los parámetros del anterior modelo para cada uno de ellos y transmite al decodificador esos parámetros exclusivamente. El decodificador empleará el modelo de producción de voz anterior y los parámetros transmitidos por el codificador para sintetizar voz para cada uno de los segmentos y posteriormente los agrupará para obtener la señal de voz decodificada.

Para la utilización de este modelo de codificación debemos por tanto asumir que podemos separar la señal de voz en segmentos (frames) suficientemente cortos como para que los parámetros del modelo no varíen. Debemos además estimar dichos parámetros del modelo para cada uno de esos segmentos:

- Coeficientes del filtro de síntesis.
- Ganancia: dependiendo de la energía de la excitación,
- Sonoridad: sordo o sonoro.
- Periodo fundamental (pitch period): periodo de la señal de excitación en sonidos sonoros.

La codificación de voz por predicción lineal supuso en 1982 un gran avance, pudiendo codificar voz a tasas binarias tan bajas como 2.4 Kb/s (53.3 veces menos tasa binaria que los 128 Kb/s de la voz digital). Sin embargo, la calidad ofrecida es muy reducida: La voz decodificada es inteligible, pero suena muy sintética y poco natural. Este tipo de codificadores encontró aplicación en el campo militar, donde la calidad es sólo un factor secundario.

La idea fundamental consiste en predecir una muestra de voz como una combinación lineal de muestras anteriores. Si denotamos la señal discreta de voz como *x*[*n*], la predicción de voz trata de encontrar los coeficientes a_k que permiten expresar aproximadamente una muestra en función de *P* muestras anteriores (donde *P* es el orden de la predicción lineal).

$$x[n] \approx \sum_{k=1}^{P} a_k x[n-k] \Rightarrow x[n] = \sum_{k=1}^{P} a_k x[n-k] + e[n]$$

La predicción con un *P* pequeño no será perfecta, y cometeremos un error de predicción, $e[n]$.

Para cada segmento de voz los pesos empleados para la combinación lineal se eligen de forma que se minimice el error cuadrático medio (energía media) del error de predicción. Los pesos resultantes se denominan *coeficientes de predicción lineal.*

La justificación para la predicción lineal se basa en el modelo digital del tracto vocal, $V(z)$, basado en una concatenación de secciones de tubos acústicos que vimos en la sección anterior.

Analizando este modelo, llegábamos a la siguiente función de transferencia donde *N* era el número de secciones de tubos acústicos empleados en el modelo:

$$V(z) = \frac{G}{1 - \sum_{k=1}^{N} \alpha_k z^{-\mathrm{k}}}$$

Esta función de transferencia define la siguiente relación entre la entrada $e'[n]$ y la salida $x[n]$ de la señal acústica en el tracto vocal:

$$V(z) = \frac{E'(z)}{X(z)} = \frac{G}{1 - \sum_{k=1}^{N} a_k z^{-\mathrm{k}}} \Rightarrow x[n] = \sum_{k=1}^{N} a_k x[n-k] + G e'[n]$$

Como vemos, la función de transferencia del modelo digital del tracto vocal, $V(z)$, que obtuvimos al analizar el mecanismo de producción de voz nos está diciendo que cada muestra de salida de la señal de voz depende linealmente de las *N* muestras anteriores (siendo *N* el número de secciones consideradas en el modelo del tracto vocal), así como de la excitación multiplicada por una ganancia.

La coincidencia con la fórmula de predicción lineal es evidente haciendo que:

- El orden de predicción lineal, *P*, coincida con el número de secciones transversales empleadas en el modelo de producción de voz, *N*.
- El error de predicción $e[n]$, sea igual a la ganancia *G* por la excitación del tracto vocal, $e'[n]$.
- $x[n]$ sea la señal acústica a la salida del tracto vocal.

En definitiva, la predicción lineal de voz es una forma de estimar los coeficientes del filtro digital que modela el tracto vocal, $V(z)$, y que también denominaremos filtro de síntesis.

Sin embargo, en el modelo digital de producción de voz había un modelo para la radiación en los labios, $R(z)$, justo después del filtro que modelaba el tracto vocal, $V(z)$. Para compensarlo, se emplea un *filtro de pre-énfasis* sobre la señal de voz, *y*[*n*] que básicamente aumenta las altas frecuencias para compensar el efecto de la radiación en los labios. Se suele tomar un valor de 0.9 para el parámetro α:

$$P(z) = \frac{X'(z)}{Y(z)} = 1 - \alpha z^{-1} \Rightarrow x'[n] = y[n] - \alpha y[n-1]$$

Con el pre-énfasis se elimina o atenúa el efecto de la radiación en los labios, $R(z)$, y se consigue una señal casi auto regresiva, $x'[n]$, similar a la que se obtiene a la salida del tracto vocal, $x[n]$.
Es sobre esta señal pre-enfatizada sobre la que se aplica la predicción lineal que permitirá calcular los coeficientes del filtro de síntesis, a_k.

Estos coeficientes son los que se transmiten al decodificador, donde se emplean para sintetizar una señal de voz. Sin embargo, la señal de voz sintética será parecida a la señal $x'[n]$, en lugar de parecerse a la señal de voz, $y[n]$. La diferencia entre ambas se debe al pre-énfasis, que habrá que eliminar mediante una operación de de-énfasis para obtener una señal parecida a la señal de voz original, $y[n]$.

$$\frac{1}{P(z)} = \frac{Y(z)}{X'(z)} = \frac{1}{1 - \alpha z^{-1}} \Rightarrow y[n] = x'[n] + \alpha y[n-1]$$

El cálculo de los coeficientes de predicción lineal parte de la expresión del error de predicción:

$$e[n] = x[n] - \sum_{k=1}^{P} a_k x[n-k]$$

Se trata de minimizar el error cuadrático medio de predicción (energía del error):

$$J = E\{e^2[n]\} = E\left\{\left(x[n] - \sum_{k=1}^{P} a_k x[n-k]\right)^2\right\}$$

Para ello tomamos derivadas parciales respecto a cada coeficiente e igualamos a 0:

$$\frac{dJ}{da_k} = -2E\left\{\left(x[n] - \sum_{i=1}^{P} a_i x[n-i]\right)x[n-k]\right\} = 0$$

Reorganizando la ecuación y escribiéndola en función de la autocorrelación de la señal $x[n]$, $Rx[k] = E\{x[n]x[n-k]\}$

$$-2E\{x[n]x[n-k]\} + 2\sum_{i=1}^{P} a_i E\{x[n-1]x[n-k]\} = 0 \Rightarrow \sum_{i=1}^{P} a_i R_x[i-k] = R_x[k]$$

La ecuación anterior se puede escribir en forma matricial como $R_x a = r_x$, donde:

$$R_x = \begin{pmatrix} R_x[0] & \cdots & R_x[P-1] \\ \vdots & \ddots & \vdots \\ R_x[P-1] & \cdots & R_x[0] \end{pmatrix} \quad a = \begin{pmatrix} a_1 \cdots a_p \end{pmatrix}^T \qquad r_x = (R_x[1] \cdots R_x[P])^T$$

Asumiendo que existe la inversa de la matriz la solución es trivial. Sin embargo, el cálculo de la matriz inversa es costoso. Existen algoritmos recursivos que permiten solucionar esta ecuación de forma más eficiente como el algoritmo de Levinson-Durbin o el de Leroux-Gueguen.

La ganancia de predicción lineal nos indica la eficiencia del predictor lineal. Se define como: $PG = 10\log_{10}\left(\frac{E\{x^2[n]\}}{E\{e^2[n]\}}\right)$.

La figura representa un ejemplo típico de evolución de la ganancia de predicción en función del orden. Podemos ver que salvo para fonemas sonoros, en los que se produce un importante aumento al alcanzar el orden de predicción el periodo fundamental, a partir de órdenes de predicción de 6-8 coeficientes no se producen ganancias importantes:

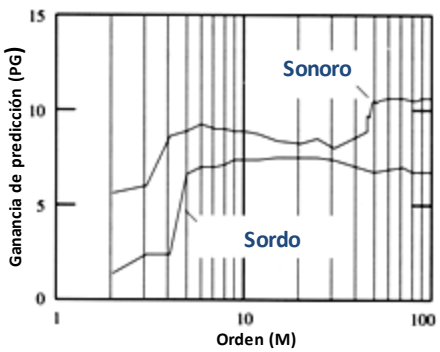


Figura 13. Evolución de ganancia de predicción en función del orden

La predicción lineal es tan interesante en codificación de voz porque típicamente permite codificar un segmento de alrededor de 200 muestras con 8 o 10 coeficientes.

Para sacar partido de la capacidad de predicción de la muestra equivalente en el periodo anterior en señales sonoras sin aumentar mucho el número de coeficientes de predicción se añade al predictor normal un predictor a largo plazo que necesita dos coeficientes: el periodo fundamental, *T*, y el coeficiente de predicción lineal a largo plazo, *b*, con lo que añadiendo un parámetro mas podemos aumentar la ganancia de predicción para sonoras.

$$x[n] \approx \sum_{k=1}^{P} a_k x[n-k] + bx[n-T] \Rightarrow x[n] = \sum_{k=1}^{P} a_k x[n-k] + bx[n-T] + e[n]$$

Una vez obtenidos los coeficientes de predicción lineal para un segmento, se calcula el error de predicción:

$$e[n] = x[n] - \sum_{k=1}^{P} a_k x[n-k] - \langle bx[n-T] \rangle$$

A la hora de sintetizar la voz en el decodificador se partirá de un tren de impulsos o de ruido blanco, ambos de cierta potencia constante. Para que la energía de la señal sintética sea la de la señal original es necesario calcular la ganancia que se debe aplicar a las excitaciones del decodificador para obtener una señal de la misma energía que el error de predicción que se obtiene en el codificador. Esa ganancia se debe enviar al codificador junto a los coeficientes de predicción lineal y el resto de parámetros.

Decisión de sonoridad

En la codificación LPC es necesario determinar si el segmento es sordo o sonoro. Observando la forma de onda se podría hacer de manera sencilla:

- Si el segmento muestra una señal cuasi-periódica, es sonoro.
- Si el segmento muestra una señal similar a un ruido, es sordo.

Sin embargo, muchas veces no se puede tomar una decisión tan estricta.

El detector de sonoridad es por tanto uno de los componentes más críticos del codificador LPC ya que una clasificación errónea lleva a una señal sintética desastrosa. Por ello, se suelen combinar varios parámetros para tomar la decisión sobre la sonoridad. Entre los parámetros más comunes están:

- **Energía del segmento:** los sonidos sonoros suelen ser mucho más energéticos. se puede emplear la suma de los valores absolutos de cada muestra del segmento (magnitude sum function, MSF).
- **Tasa de cruces por cero (ZCR):** el número de veces que la señal cruza el eje horizontal, mucho mayor en sonidos sordos.
- **Ganancia de predicción lineal (PG):** Como explicamos anteriormente, es el ratio entre la energía de la señal y la energía del error de predicción lineal. Los sonidos sonoros suelen tener mayor ganancia de predicción porque una muestra es más fácil de predecir en función de las anteriores.

Por último, si el segmento era sonoro se debe estimar el periodo fundamental o *pitch period*. Se puede estimar a partir de la señal de error de predicción de manera más precisa que a partir de la señal de voz, ya que la predicción lineal consigue eliminar de la señal el efecto del tracto vocal.

Existen muchos métodos para estimar el periodo fundamental de una señal como por ejemplo:

- **Método de la autocorrelación:** calcular la autocorrelación de la señal y calcular su primer máximo (que debe corresponder con la frecuencia fundamental). *N* es el tamaño del segmento.

$$R[l,m] = \sum_{n=m-N-1}^{m} e[n]e[n-l]$$

- **Método de la función del valor absoluto de la diferencia**: (*magnitude difference function*, MDF). Consiste en minimizar esta función en *l*.

$$MDF[l,m] = \sum_{n=m-N-1}^{m} |e[n] - e[n-l]|$$

La estructura del decodificador LPC es el modelo de producción de voz que vimos en el apartado anterior en el que las informaciones provenientes del codificador se desempaquetan y se decodifican antes de ser utilizadas. Además, se aplica un filtro de deénfasis final para compensar el filtro de pre-énfasis del codificador.

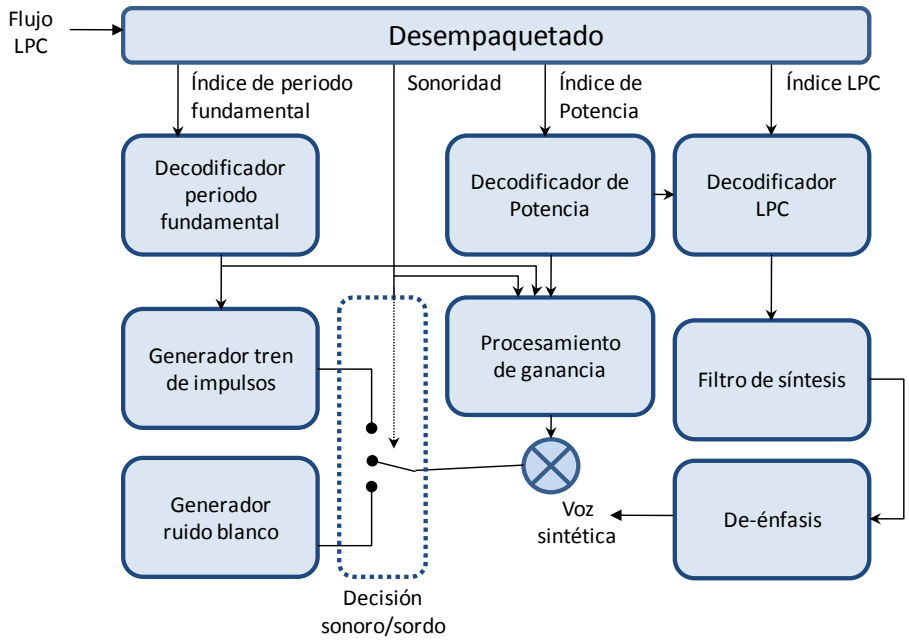


Figura 14. Estructura decodificador LPC

2.2.4.2 Codificador de excitación multipulso

Los codificadores de excitación multipulso (multipulse excitation coders) están basados en la codificación de predicción lineal pero evitando la generación de señales de excitación como ruido blanco o como trenes de impulsos. Para ello, si en lugar de indicar únicamente la sonoridad, periodo fundamental y ganancia de la señal de excitación se enviase la señal completa. Con ella, se podría recuperar a la perfección la señal original.

Esta técnica aplicada a la señal de excitación no sería práctica porque la tasa binaria aumentaría considerablemente. Sin embargo, se puede aplicar a la señal de error, que tiene un margen dinámico mucho menor. Es más, se observó que no hacía falta enviar todas las muestras de la señal de error de predicción: típicamente con enviar el 10% de las mismas y suponer que el resto eran nulas se conseguía una buena calidad.

El problema está en determinar qué muestras enviar y cuáles no para conseguir la mínima distorsión. Una vez determinadas las muestras (pulsos) a enviar, éstas se cuantifican y se envían junto con la posición que ocupan (la posición es necesaria al no transmitir todas las muestras del error de predicción).

La selección de las muestras puede realizarse de dos maneras distintas:

- **Selección de pulsos en lazo abierto:** En este caso se seleccionan los pulsos (las muestras) de acuerdo a cierto criterio. Por ejemplo se pueden seleccionar los pulsos de mayor amplitud. El problema de esta aproximación es que es necesario transmitir tanto los pulsos como sus posiciones. Para evitarlo, es muy común emplear lo que se conoce como Regular Pulse Excitation (RPE) coders. Éstos codificadores eligen una muestra de cada N. Así son posibles N elecciones distintas de pulsos pudiéndose optar por la que mayor energía tenga. De este modo sólo es necesario transmitir las amplitudes y la posición del primer pulso (número entre 0 y N-1). Un ejemplo de este tipo de codificadores es el GSM-FR.

- **Selección de pulsos en lazo cerrado:** En este caso en el codificador se realiza la síntesis de la señal a partir de los pulsos seleccionados y se modifica la selección de pulsos para minimizar la diferencia entre la señal original y la señal sintética. Éste principio es conocido como "análisis por síntesis" (analysis by synthesis) y también es esencial en otros codificadores de voz modernos. En general la selección de pulsos en lazo cerrado proporciona mayor calidad, pero con un mayor coste computacional.

Predicción lineal excitada por código (CELP)

La predicción lineal excitada por código (codificación CELP) se basa en el siguiente modelo de producción de voz:

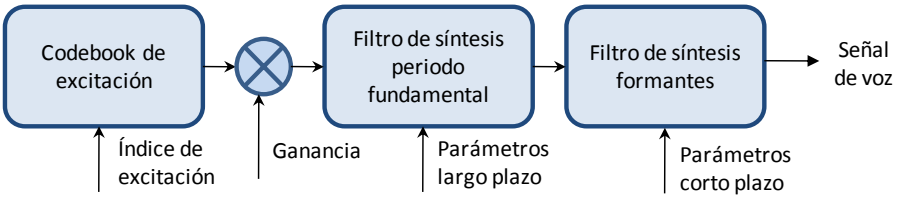


Figura 15. Modelo de producción de voz CELP

El codificador y el decodificador comparten un *codebook* de secuencias de excitación, de modo que el codificador sólo tiene que transmitir el índice de la secuencia de excitación a emplear en el decodificador. Por ello recibe el nombre de predicción lineal excitada por código.

Después de obtener la excitación del *codebook*, se le aplica una ganancia y se pasa por un filtro de síntesis de la frecuencia fundamental (predicción lineal a largo plazo), y finalmente por el filtro de síntesis de formantes o envolvente espectral (predicción lineal a corto plazo).

Para seleccionar las secuencias de excitación a emplear para un segmento de voz, se prueba con cada una de las secuencias de excitación, se decodifica (en el codificador) el bloque de voz con cada una de las secuencias de excitación, y se elige la secuencia de excitación que hace que la señal decodificada sea más parecida a la señal original. Sigue por tanto el modelo de análisis por síntesis que comentábamos en la selección de pulsos en lazo cerrado.

Operación de un codificador CELP

En la siguiente figura mostramos el diagrama de bloques de un codificador CELP. El proceso seguido en la codificación es el siguiente:

- Se divide la señal de voz en segmentos (frames) y subsegmentos (subframes).
- Se realiza el análisis de predicción lineal a corto plazo.
- Se realiza el análisis de predicción lineal a largo plazo a partir del error de predicción del anterior.
- A continuación se busca la secuencia de excitación óptima y la ganancia a aplicar.
- Se codifican el índice del vector de excitación, la ganancia y los coeficientes de predicción lineal a largo y corto plazo

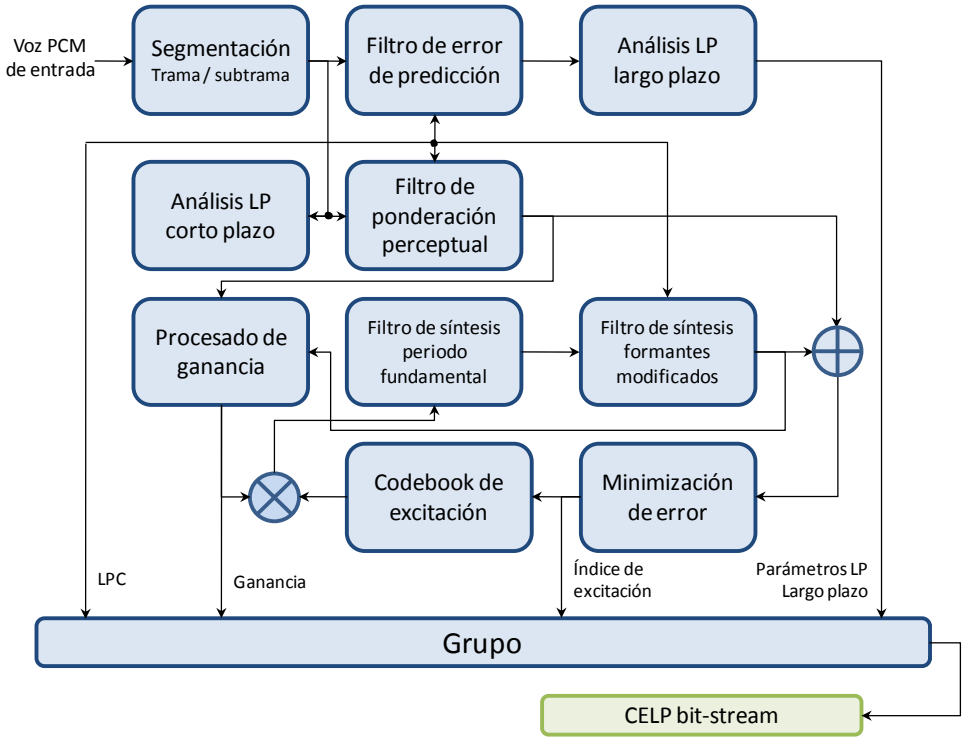


Figura 16. Diagrama de bloques codificador CELP

El decodificador CELP es básicamente el modelo de producción de voz CELP. En él, se decodifican las informaciones producidas por el decodificador y se aplican al modelo de producción de voz. Se añade un post-filtrado al final para mejorar la calidad de la señal resultante:

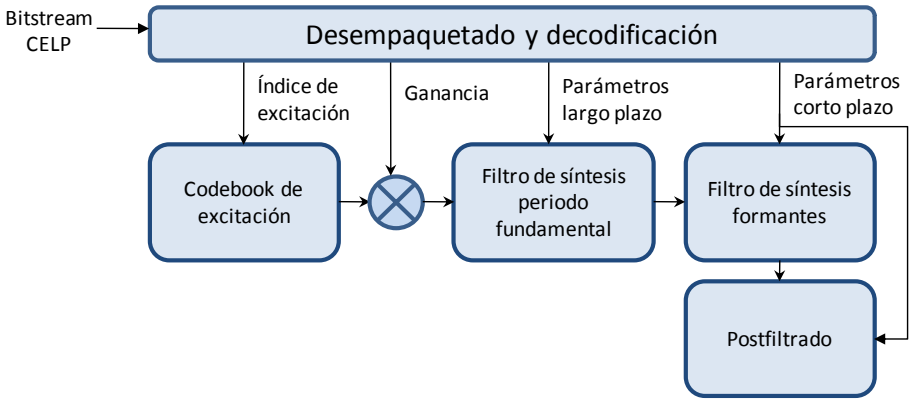


Figura 17. Diagrama de bloques decodificador CELP

La codificación CELP es la base de todos los codificadores de voz modernos. La búsqueda del vector de excitación es lo más costoso computacionalmente, aunque se han desarrollado muchas optimizaciones en la búsqueda, lo que lo ha convertido en un algoritmo práctico.

CELP Algebraico (ACELP)

El CELP algebraico o ACELP es un codificador CELP en el que el vector de excitación se genera por medio de operaciones algebraicas sencillas y reglas matemáticas definidas a partir de un índice de vector de excitación de cierto número de bits.

En la práctica el vector de excitación se define como la suma de un número fijo de pulsos desplazados. La posición de cada uno de los pulsos y su signo vienen definidos por los bits del índice del vector de excitación.

Existen multitud de codificadores modernos basados en ACELP:

- ITU-T G.723.1 Multipulse Maximum Likelihood Quantization (MP-MLQ)/ACELP, 1995, 8 kb/s.
- ITU-T G.729 Conjugate Structure (CS)-ACELP, 1995. Se emplea para videollamadas con muy baja tasa binaria, el audio se codifica a 5.3 Kb/s y 6.3 Kb/s.
- TIA IS641 ACELP, 1996.
- ETSI GSM Enhanced Full Rate (EFR) ACELP, 1996.
- ETSI Adaptive Multirate (AMR) ACELP, 1996.

El ITU-T G.723.1 genera el vector de excitación de 40 muestras como la suma de cuatro pulsos, de acuerdo con la figura inferior (en la que las celdas azules representan las posibles posiciones de cada pulso).

- Para los pulsos 0 a 2 son necesarios 3 bits para especificar la posición del pulso.
- Para el pulso 3 son necesarios 4 bits para especificar la posición.
- Por cada pulso se debe especificar el signo, pues el pulso puede tomar el valor +1 o –1.
- En total, un vector de excitación queda determinado por 17 bits.

Los demás estándares ACELP son muy similares en este aspecto, si bien cambia el número de pulsos y las posibles posiciones de los mismos.

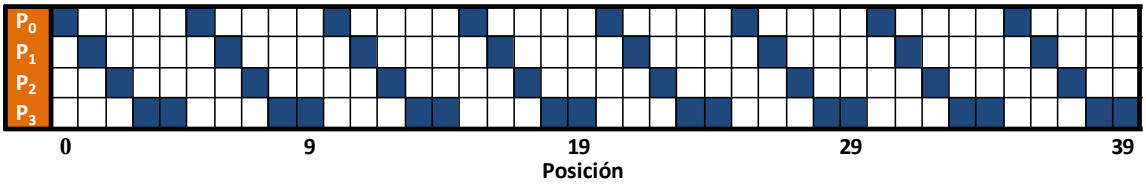


Figura 18. Esquema funcionamiento ITU-T G.723.1

3 EVALUACIÓN DE CODIFICADORES

3.1 Diseño y desarrollo

El primer paso en nuestro camino hacia el desarrollo de nuestro códec propietario consiste en la evaluación de tres codificadores, el ETSI GSM 6.10 (GSM-FR), ETSI Adaptive Multirate Wideband v7.0.0 (AMR-WB) y el Speex 1.2b3. El GSM-FR es el códec que queremos sustituir y el AMR-WB y el Speex 1.2b3 son los candidatos a convertirse en la base de nuestro codificador propietario.

Llevaremos a cabo distintas pruebas, que hemos organizado en tres bloques:

- **Pruebas de calidad:** Evaluaremos la calidad objetiva ofrecida en la codificación por los tres códecs evaluados en tres escenarios de interés para el cliente: escenario original, con ficheros de audio sin tratar por nosotros, escenario de baja amplitud o bajo volumen en el que los ficheros han visto rebajada su amplitud y escenario ruidoso en el que se les añade ruido blanco y gaussiano.
- **Pruebas de coste computacional:** Evaluaremos el tiempo de codificación de cada uno de ellos.
- **Pruebas de procesamiento posterior:** Evaluaremos el efecto introducido por la codificación sobre sistemas de reconocimiento de locutor, idioma, habla y reducción de ruido

3.1.1 Descripción de los Códecs evaluados.

A continuación presentamos un breve resumen de las características de los tres códecs evaluados:

3.1.1.1 ETSI GSM 6.10 (GSM-FR)

Introducido en 1987 su última publicación es de Junio de 2001. Fue el primer estándar de codificación digital de voz usado en sistemas GSM de telefonía móvil. Está basado en el modelo de codificación CELP RPE-LTP (Regular Pulse Excitation – Long Term Prediction) con una predicción lineal de orden 8. Únicamente opera en banda estrecha (8 KHz) y con una tasa binaria de 13 Kbps

El predictor es implementado como una conexión en cascada de predictores a corto y largo plazo. La inclusión de los predictores de largo plazo mejora notablemente la ganancia en la predicción.

De la señal de error de predicción lineal se elige una de cada tres muestras. De entre las tres opciones posibles se elige la de mayor energía. La señal obtenida se cuantifica con un codificador PCM de ganancia adaptativa APCM.

El modelo de codificación incluye enventanado en tramas de 160 muestras (20ms) y subtramas de 40 muestras (5 ms), preprocesado, análisis de predicción lineal por cada trama, cuantificación LPC e interpolación. A continuación se genera el filtro de predicción de error, se realiza el análisis de predicción lineal a largo plazo y la codificación. El descodificador sigue una estructura similar con operaciones inversas a las que realiza el codificador.

3.1.1.2 ETSI ADAPTIVE MULTIRATE WIDEBAND v7.0.0 (AMR-WB)

Estandarizado por la ETSI/3GPP en Diciembre de 2001 y aprobado por la ITU en Enero de 2002. Su última publicación es de Junio de 2007. Está basado en el modelo de codificación ACELP (Algebraic code-excited linear prediction) con una predicción lineal de orden 16. Presenta la posibilidad de adaptación dinámica a las condiciones de la red y cuenta con implementación en punto flotante y aritmética entera. Opera en banda estrecha (8 kHz) y banda ancha (16 kHz) con tasas binarias de 1.75 a 23.85 Kbps según el modo de operación seleccionado. Todos los modos son muestreados a 16 kHz y procesados a 12.8 kHz.

Modo	0	1	2	3	4	5	6	7	8	9
Bit-rate [Kbps]	1.75	6.60	8.85	12.65	14.25	15.85	18.25	19.85	23.05	23.85

Tabla 2. Modos de operación / tasas binarias AMR-WB v7.0.0

Como hemos vimos en el Estado del Arte, el modelo de codificación ACELP surgió con la intención de reducir el coste computacional de los codificadores CELP estándar donde la operación más intensa es la búsqueda del codebook de excitación. El término "algebraico" simboliza esencialmente la utilización de reglas matemáticas o álgebra simple para crear los codevectors de excitación, utilizando las reglas de desplazamiento y suma. Esto permite un ahorro significativo de memoria al no tener que almacenar físicamente la totalidad del codebook.

El códec AMR-WB consiste por tanto en nueve codificadores con tasas binarias de 23.85, 23.05, 19.85, 18.25, 15.85, 14.25, 12.65, 8.85 y 6.60 Kbps. La señal de entrada se pre-enfatiza y a continuación se aplica el modelo CELP a dicha señal. Se utiliza un filtro de síntesis o predicción lineal de orden 16.

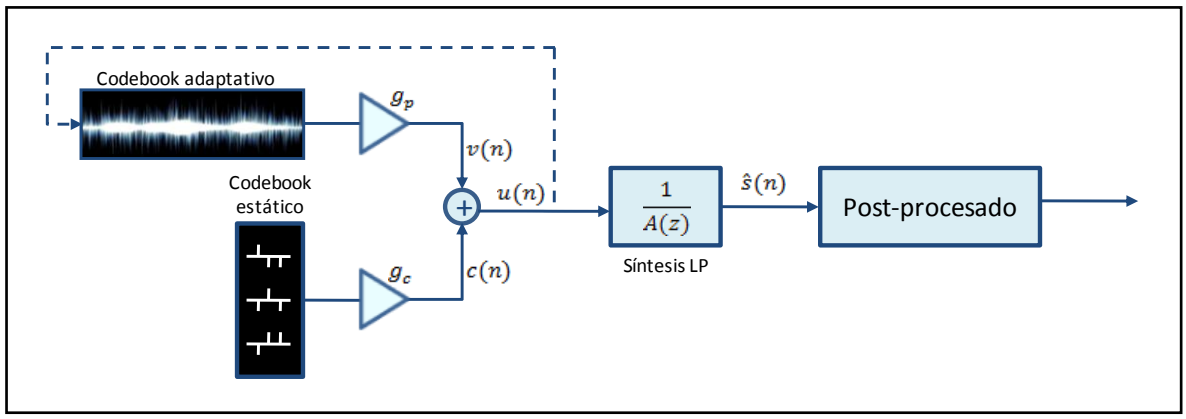


Figura 19. Diagrama de bloques simplificado del modelo de codificación CELP

En la figura podemos observar el modelo CELP de síntesis de voz. En este modelo, la señal de excitación a la entrada del filtro de predicción lineal a corto plazo se construye sumando dos vectores de excitación. Uno desde el codebook estático y otro desde el adaptativo.

La voz se sintetiza pasando los dos vectores elegidos de estos codebooks por el filtrado a corto plazo. La secuencia óptima de excitación en un codebook se elige a través de una búsqueda de *análisis por síntesis* en la que el error entre la voz original y la sintetizada se minimiza de acuerdo a una medida de distorsión perceptual.

El codificador lleva a cabo el análisis LPC, LPT y de los parámetros del codebook estático a una frecuencia de muestreo de 12.8 kHz. El codificador opera con tramas de 20 ms. En cada trama, se analiza la señal vocal para extraer los parámetros del modelo CELP (coeficientes del filtro de predicción lineal, índices y ganancias de los codebooks fijo y adaptativo). Además de estos parámetros se computan los índices de ganancia de la parte superior de la banda en modo de 23.85 Kbps. Estos parámetros son codificados y transmitidos. En el descodificador estos parámetros se descodifican y la voz es sintetizada filtrando la señal de excitación reconstruida a través del filtro de síntesis de predicción lineal.

3.1.1.3 SPEEX 1.2b3

Se trata de un códec de software libre, sin patentes ni royalties. Su última publicación es de Julio de 2008. Su diseño fue orientado hacia redes de intercambio de paquetes (aplicaciones VoIP y similares) y no hacia telefonía móvil lo que implica una mayor robustez frente a la pérdida de paquetes aunque menor frente a paquetes corruptos. Está basado en el modelo de codificación CELP (code-excited linear predictive). Presenta la posibilidad de adaptación dinámica a las condiciones de la red (VBR), detección de actividad vocal (VAD), transmisión discontinua (DTX) e implementación en punto flotante y aritmética entera. Integra banda estrecha (8 kHz), banda ancha (16 kHz) y banda ultra-ancha (32 kHz) con tasas binarias de 2.15 a 44 Kbps según el modo de operación seleccionado y la posibilidad de codificar audio en estéreo (modalidad de intensidad).

Modo	0	1	2	3	4	5	6	7	8	9	10
Bit-rate [Kbps]	3.95	5.75	7.75	9.8	12.8	16.8	20.6	23.8	27.8	34.2	42.2

Tabla 3. Modos de operación / tasas binarias SPEEX 1.2b3

El códec además de las características antes descritas incluye los siguientes módulos adicionales:

Preprocesado

Diseñado para la preparación del audio como paso previo a la codificación. Entre sus funciones está la eliminación de ruido, el control automático de ganancia (AGC) y la detección de actividad vocal (VAD). La reducción de ruido permite rebajar el ruido de fondo reduciendo el efecto de amplificación de ruido que se produce en todos los codificadores de voz. El control automático de ganancia (AGC) permite ajustar la amplitud de la señal a un volumen de referencia. El detector de actividad vocal (VAD) presente en el preprocesado es más avanzado que el proporcionado directamente en el codificador.

Buffer/Jitter adaptativo

Cuando se transmite voz sobre redes UDP o RTP los paquetes pueden perderse, llegar con distinto retardo o incluso desordenados. La función de este buffer es precisamente la reordenación de los paquetes para su posterior envío al descodificador.

Cancelador de eco

En los sistemas de comunicaciones “manos libres” el sonido de la voz se propaga a través de la habitación y el micrófono capta también las reverberaciones. El cancelador de eco permite eliminar el eco acústico producido antes de enviar la señal al descodificador.

Remuestreo

En algunos casos resulta útil poder convertir audio entre diferentes tasas binarias, por ejemplo para poder reproducir audio con tasas binarias que no soporta una tarjeta de sonido, mezclar audio con diferentes tasas binarias o realizar una transcodificación. Este remuestreo se puede usar para convertir entre dos tasas binarias arbitrarias (siempre que la relación entre ellas sea un número racional). Existe un control entre la relación entre calidad y complejidad.

Speex ha sido compilado y probado con éxito en las siguientes plataformas: x86, x86-64, Power, SPARC, ARM, Blackfin, Coldfire (familia de 68k), TI C54XX, C55XX, C6XXX y TriMedia (experimental).

Los sistemas operativos en los que SPEEX ha sido probado con éxito son: Linux, µClinux, MacOS X, BSD, Symbian y otras variantes UNIX/POSIX.

En la siguiente tabla mostramos una comparativa de los tres códecs bajo estudio:

	GSM-FR 6.10	AMR-WB v7.0.0	SPEEX 1.2b3
Fecha	Junio 2001	Junio 2007	Diciembre 2007
Tipo	RPE-LTP	ACELP	CELP
Tasa muestreo (kHz)	8	8, 16	8, 16, 32
Tasas binarias (Kbps)	13	1.75 – 23.85	2.15 – 24.6 (NB) 4 – 44.2 (WB)
Packet Loss Cancelling	NO	SI	SI
DTX	NO	SI	SI
VBR	NO	SI	SI
Estéreo/Mono	Mono	Mono	Mono/ Estéreo de Intensidad
Aspectos legales	Patentado	Patentado	Software Libre

Tabla 4. Comparativo códecs bajo evaluación

3.1.2 Bases de datos empleadas en la evaluación.

En los experimentos realizados se ha empleado audio de distintas bases de datos que nos permiten evaluar la calidad de los codificadores en distintos escenarios y el efecto de éstos sobre el procesamiento posterior (reconocimiento de locutor, idioma, habla y filtrado de ruido).

3.1.2.1 Bases de datos para la prueba de calidad objetiva.

Para la evaluación de calidad en los distintos escenarios se ha empleado la base de datos ALBAYZIN que consta de 3 corpus de señales grabadas a 16 kHz y producidas por 304 locutores de la variedad central del castellano:

- **Corpus fonético**: 6800 elocuciones de frases equilibradas fonéticamente. 1000 de ellas han sido segmentadas por fonemas.
- **Corpus geográfico:** 6800 elocuciones de frases correspondientes a la consulta de una base de datos geográfica española.
- **Corpus "Lombard":** 2000 elocuciones de los corpus anteriores producidas en condiciones adversas.

En los experimentos de evaluación de calidad hemos empleado:

- **Como conjunto de datos reducido:** Una selección de 18 grabaciones de voz, la mitad de hombres y la mitad de mujeres, 3 de cada franja de edad (18-30, 31-40 y 41-55 años) del subcorpus de prueba del corpus fonético de ALBAYZIN.
- **Como conjunto de datos ampliado:** Todas las grabaciones del subcorpus de prueba del corpus fonético de la base de datos ALBAYZIN.
- **Como conjunto de datos de ruido:** 10 grabaciones de “efectos de sonido” o “ruidos”, fragmentos procedentes de la compilación “Efectos de sonido II” de la editorial Doblón.

3.1.2.2 Bases de datos para el análisis del coste computacional.

Para el análisis del coste computacional se ha empleado el conjunto de datos reducido descrito en la sección anterior.

3.1.2.3 Bases de datos para la prueba de procesamiento posterior.

Para el efecto introducido sobre el reconocimiento del locutor se han empleado los datos de la última evaluación de reconocimiento de locutor NIST SRE 2008 compuestos únicamente por voz telefónica espontánea y locutores masculinos.

Para el reconocimiento del habla se han empleado todas las grabaciones del subcorpus de prueba del corpus fonético de la base de datos ALBAYZIN, es decir el conjunto de datos ampliado utilizado en la prueba de calidad objetiva.

Para el efecto del ruido se ha empleado el conjunto de datos reducido descrito en la sección de calidad objetiva.

Para el reconocimiento del idioma se ha empleado audio de la base de datos CallFriend que es ampliamente utilizada en el campo de reconocimiento automático de idioma. De hecho, se ha utilizado varias veces en las propias evaluaciones NIST de reconocimiento de idioma, en concreto esta base de datos supuso el corpus principal de las evaluaciones de NIST 1996 y NIST 2003. Esta base de datos tiene 11 idiomas, en los experimentos realizados se utilizaron 4 de ellos. La base de datos está compuesta por 3 grupos distintos de ficheros de audio: “train”, “development” y “test”, de estos grupos únicamente se emplearon el conjunto de train y parte del conjunto de test, lo que supone aproximadamente 27 horas de audio.

En la base de datos CallFriend no existen ficheros de audio de Ruso, con lo que para este idioma se empleó, en su lugar, una base de datos muy similar a ella llamada RusTen.

3.2 Pruebas y resultados

Este apartado detalla las pruebas y resultados obtenidos en la evaluación de los tres codificadores

3.2.1 Pruebas de calidad.

Las pruebas de calidad se han desarrollado sobre los conjuntos de datos reducido, ampliado y ruido descritos en la sección anterior de este documento.

La recomendación ITU-T P.563, método basado en un sólo extremo para la evaluación objetiva de la calidad vocal, nos permitirá evaluar la bondad en la codificación/decodificación ofrecida por cada uno de ellos. El enfoque empleado en esta Recomendación constituye el primer método recomendado para realizar medidas no intrusivas basadas en un sólo extremo que tiene en cuenta toda la gama de distorsiones que se producen en las redes telefónicas públicas conmutadas, y permite predecir la calidad vocal sobre una escala de percepción MOS-LQO de conformidad con la Rec. UIT-T P.800.1. La puntuación así calculada es comparable a la calidad percibida por un oyente humano, que escucha en ese punto utilizando un auricular convencional.
La evaluación de tecnologías de codificación forma parte de los factores para los que la p.563 ha demostrado una precisión aceptable formando parte de los escenarios de prueba recomendados.

El enfoque de P.563 predice los resultados que se obtendrían con experimentos de calidad de audición subjetiva ACR (LQS, *listening quality subjective*) calculando un valor de calidad de audición objetiva (LQO, *listening quality objective*) en la escala MOS de 1 a 5. Por tanto, esta Recomendación debe considerarse relacionada principalmente con la escala de opinión de la calidad de audición ACR.

El algoritmo empleado en la recomendación está diseñado exclusivamente para evaluar la voz humana, por lo que no ha sido utilizada para evaluar la calidad en la codificación de ruidos o

efectos *de sonido*. En ese caso ha sido necesario realizar escuchas de los ficheros obtenidos para las distintas codificaciones.

Las codificaciones y decodificaciones para los códecs AMR-WB y SPEEX se han llevado a cabo utilizando banda ancha (16 kHz) mientras que para las de GSM se ha utilizado banda estrecha (única opción posible). A la hora de evaluar la calidad mediante la recomendación p.563 ha sido necesario pasar los ficheros resultado de codificar y descodificar con AMR-WB y SPEEX a 8 kHz de muestreo por ser un requisito de la recomendación.

El proceso seguido es el siguiente:

- Codificación de los ficheros de interés mediante el códec bajo estudio.
- Decodificación de los ficheros de interés mediante el códec bajo estudio.
- Conversión a 8 kHz de muestreo (en el caso de AMR-WB y SPEEX) y formato raw (el sonido se representa como datos modulados PCM).
- Paso del ejecutable de la recomendación p.563 con los ficheros de interés bajo estudio, obteniendo una puntuación numérica entre 1 (mínima calidad) y 5 (calidad máxima).

3.2.1.1 Conjunto Original

Para las codificaciones del conjunto de datos original podemos apreciar como SPEEX y AMR-WB mejoran la calidad de GSM aunque las diferencias entre ellos son reducidas. Los resultados para el conjunto de datos reducido son los siguientes:

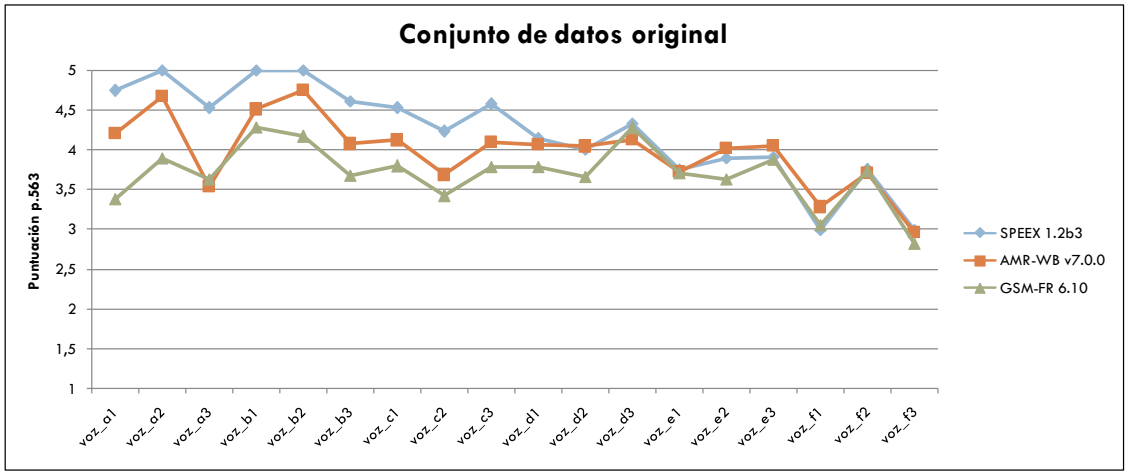


Figura 20. Puntuación p.563 conjunto reducido original

	SPEEX 1.2b3 12,8 Kbps	AMR-WB v7.0.0 12,65 Kbps	GSM-FR 6.10 13 Kbps
Media	4,22	3,98	3,7
Desv típica	0,61	0,45	0,37

Tabla 5. Puntuación p.563 conjunto reducido original

A continuación mostramos los resultados obtenidos para el conjunto de datos ampliado:

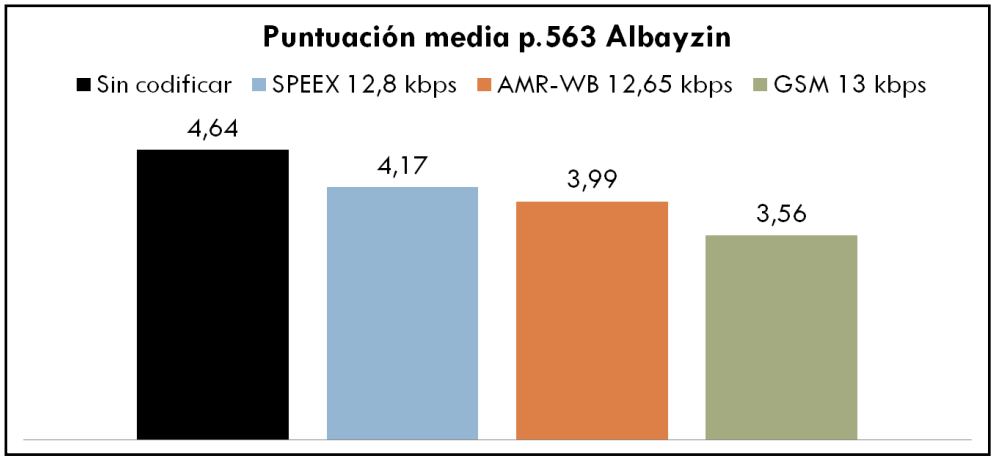


Figura 21. Puntuación p.563 conjunto ampliado original

	SIN CODIFICAR	GSM-FR 6.10 13 Kbps	AMR-WB v7.0.0 12,65 Kbps	SPEEX 1.2b3 12,8 Kbps
Media	4,64	3,56	3,99	4,17
Desv típica	0,40	0,39	0,43	0,47

Tabla 6. Puntuación p.563 conjunto ampliado original

3.2.1.2 Conjunto con ruido

Para la prueba de calidad de los códecs en escenarios ruidosos hemos empleado el mismo conjunto de datos anteriormente descrito, al que se ha añadido ruido blanco y gaussiano con una SNR de 30 dB, antes de aplicar las codificaciones y descodificaciones correspondientes. A la vista de los resultados, podemos concluir que SPEEX obtiene una mayor puntuación seguido de AMR, aunque las diferencias entre ambos son reducidas.

En el caso del conjunto reducido con ruido las puntuaciones obtenidas son:

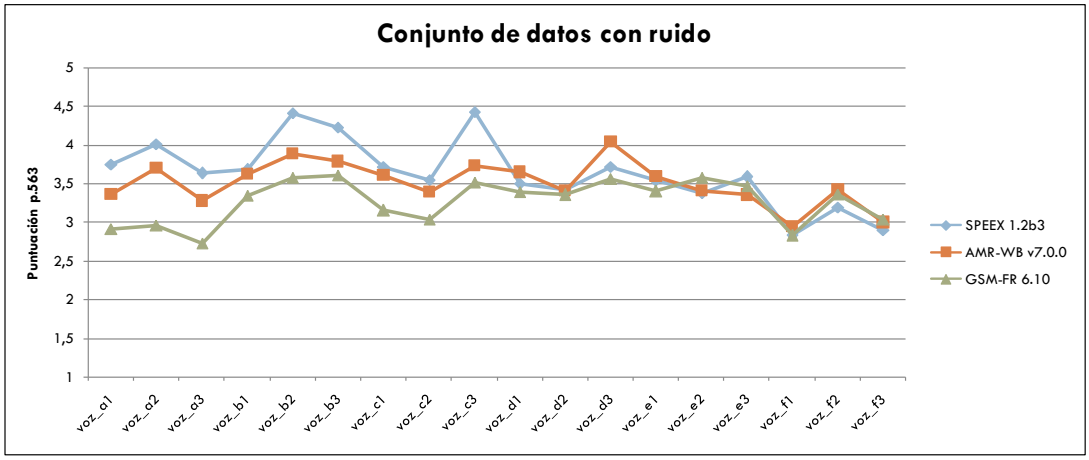


Figura 22. Puntuación p.563 conjunto reducido con ruido blanco añadido

	SPEEX 1.2b3 12,8 Kbps	AMR-WB v7.0.0 12,65 Kbps	GSM-FR 6.10 13 Kbps
Media	3,64	3,51	3,27
Desv típica	0,44	0,28	0,29

Tabla 7. Puntuación p.563 conjunto reducido con ruido blanco añadido

Para el conjunto de datos ampliado:

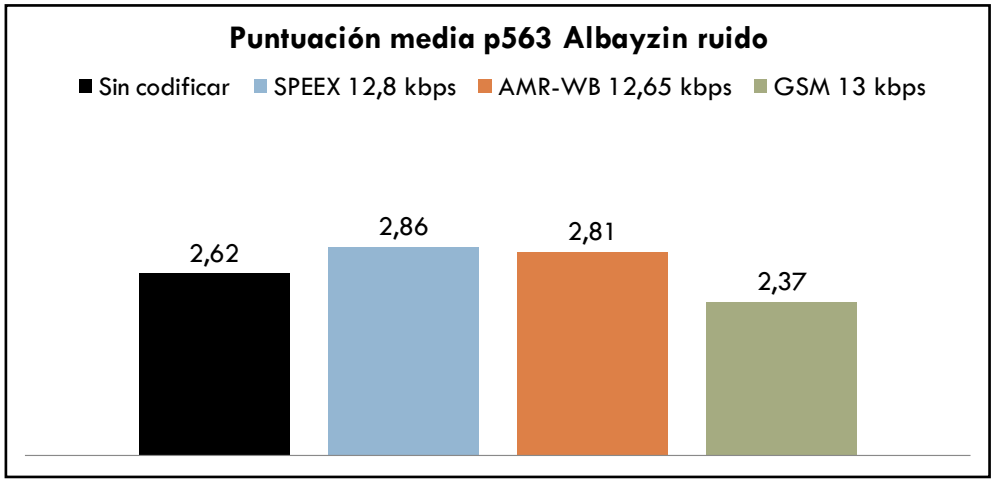


Figura 23. Puntuación p.563 conjunto ampliado con ruido blanco añadido

	SPEEX 1.2b3 12,8 Kbps	AMR-WB v7.0.0 12,65 Kbps	GSM-FR 6.10 13 Kbps
Media	3,64	3,51	3,27
Desv típica	0,44	0,28	0,29

Tabla 8. Puntuación p.563 conjunto ampliado con ruido blanco añadido

3.2.1.3 Conjunto atenuado

Para la prueba de los codificadores en condiciones de bajo volumen se ha atenuado los ficheros de los conjuntos de datos reducido y ampliado 10, 20, 30 y 40 dB antes de aplicar las codificaciones/descodificaciones correspondientes. Los ficheros rebajados 40 dB no han podido ser evaluados mediante la Recomendación p.563 por no cumplir el margen dinámico especificado en sus requerimientos.

Presentamos los resultados obtenidos para el conjunto rebajado 30 dB (los obtenidos para 20 dB son similares) donde podemos apreciar que la calidad ofrecida por SPEEX es claramente superior a la proporcionada por AMR-WB y GSM.

Para el conjunto de datos reducido:

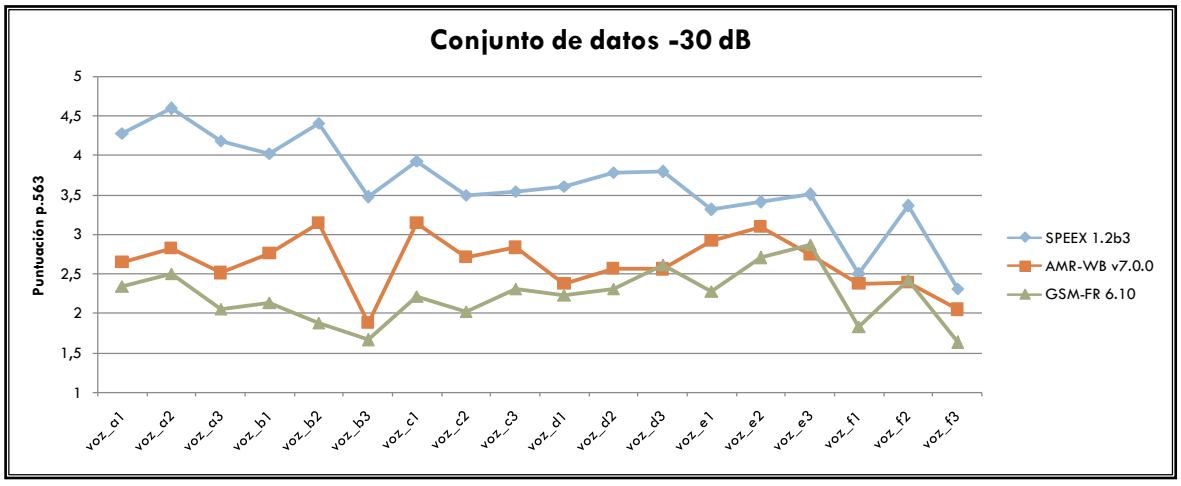


Figura 24. Puntuación p.563 conjunto reducido atenuado 30 dB

	SPEEX 1.2b3 12,8 Kbps	AMR-WB v7.0.0 12,65 Kbps	GSM-FR 6.10 13 Kbps
Media	3,64	2,64	2,22
Desv típica	0,59	0,35	0,34

Tabla 9. Puntuación p.563 conjunto reducido atenuado 30 dB

Para el conjunto de datos ampliado:

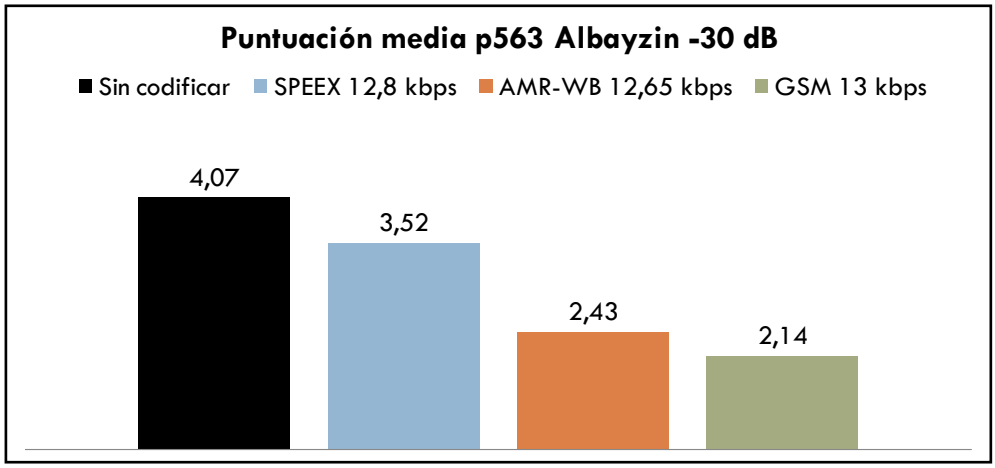


Figura 25. Puntuación p.563 conjunto ampliado atenuado 30 dB

	SIN CODIFICAR	GSM-FR 6.10 13 Kbps	AMR-WB v7.0.0 12,65 Kbps	SPEEX 1.2b3 12,8 Kbps
Media	4,07	2,14	2,43	3,52
Desv típica	0,60	0,47	0,49	0,55

Tabla 10. Puntuación p.563 conjunto ampliado atenuado 30 dB

Se ha realizado además para el conjunto de datos reducido una comparativa de la degradación de calidad sufrida por los archivos codificados mediante AMR-WB y SPEEX al bajar la tasa binaria de codificación. Para AMR-WB:

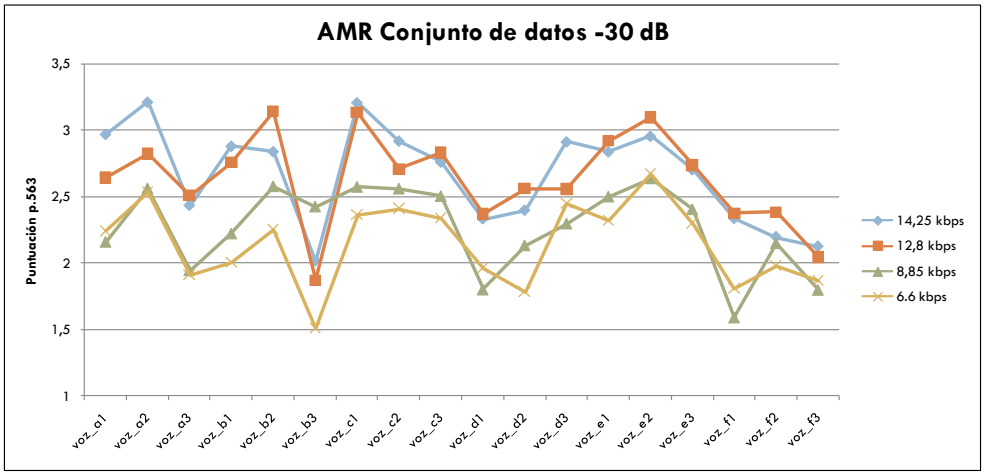


Figura 26. Puntuación p.563 conjunto reducido atenuado 30 dB. Degradación de calidad AMR-WB

	14,25 kbps	12,8 kbps	8,85 kbps	6.6 kbps
Media	2,67	2,64	2,27	2,15
Desv típica	0,37	0,35	0,32	0,31

Tabla 11. Puntuación p.563 Prueba degradación calidad AMR-WB

Para SPEEX 1.2b3:

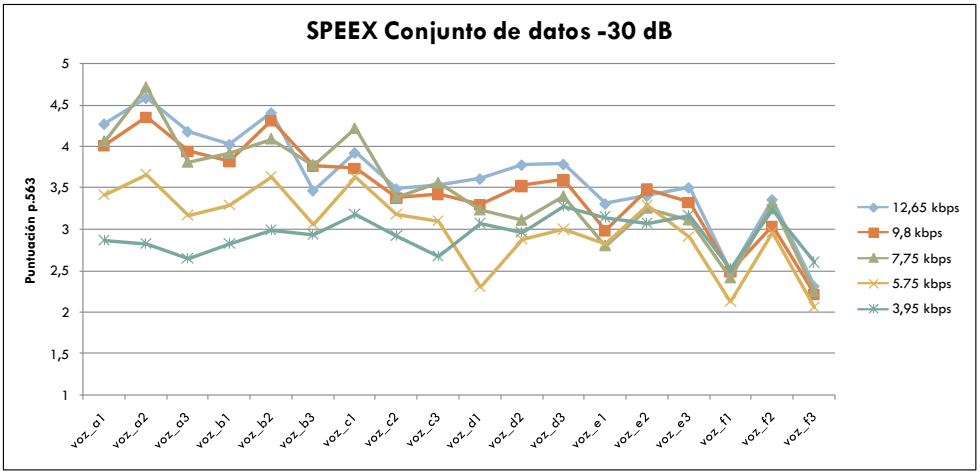


Figura 27. Puntuación p.563 conjunto reducido atenuado 30 dB. Degradación de calidad SPEEX

	12,65 kbps	9,8 kbps	7,75 kbps	5.75 kbps	3,95 kbps
Media	3,64	3,48	3,47	3,03	2,94
Desv típica	0,58	0,56	0,63	0,48	0,23

Tabla 12. Puntuación p.563 Prueba degradación calidad AMR-WB

Podemos comprobar que la calidad ofrecida por Speex en esta prueba supera claramente a la ofrecida por AMR-WB para las distintas tasas binarias comprobadas superando incluso las puntuaciones obtenidas por Speex para tasas binarias más bajas a las empleadas en AMR-WB.

3.2.1.4 Efectos de sonido

En cuanto a la codificación de *ruidos* o *efectos de sonido*, ha sido necesario escuchar los ficheros obtenidos para evaluar la calidad ofrecida ya que el algoritmo empleado en la Recomendación p.563 está diseñado exclusivamente para la evaluación de voz humana. La diferencia de calidad entre los códecs se muestra poco apreciable aunque el GSM parece ser más agresivo con el ruido.

3.2.2 Prueba de coste computacional.

En cuanto a coste computacional medido como el tiempo de codificación en el sistema probado, encontramos que SPEEX es algo menos de un 50% más costoso que GSM-FR mientras que AMR-WB es casi un 800% más costoso que GSM-FR.

A continuación mostramos una gráfica que muestra la relación entre el tiempo de codificación y la duración del fichero obtenida para el conjunto de datos reducido. Estos resultados se han obtenido sobre un ordenador portátil Toshiba Satellite A50-522 con procesador Intel Centrino M715 a 1.50 GHz con 512 MB DDR RAM y con sistema operativo Kubuntu 7.10.

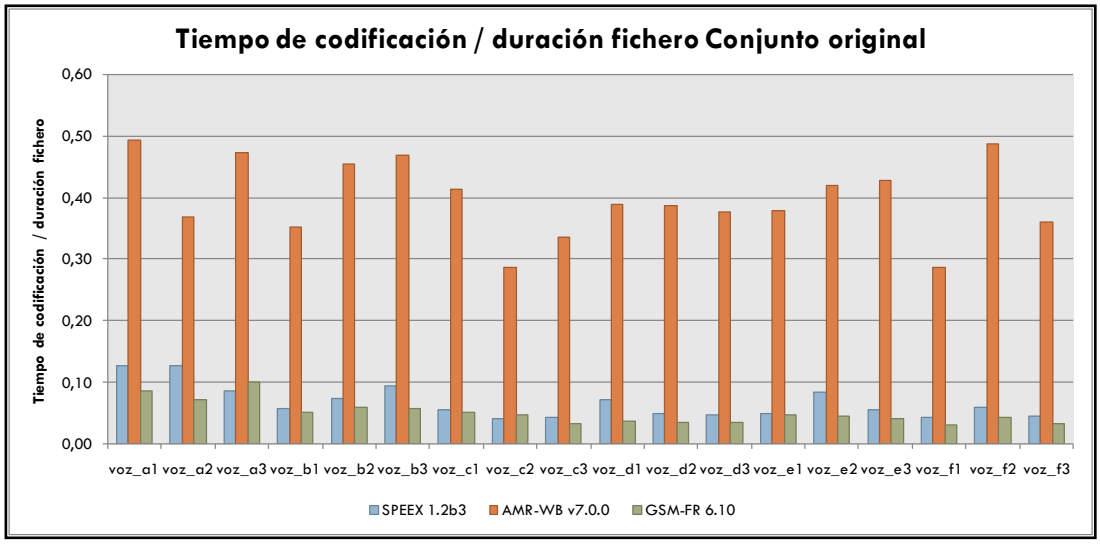


Figura 28. Tiempo de codificación sobre duración de fichero original conjunto reducido

	SPEEX 1.2b3 12,8 Kbps	AMR-WB v7.0.0 12,65 Kbps	GSM-FR 6.10 13 Kbps
Media	0,067	0,398	0,051
Desv típica	0,03	0,06	0,02

Tabla 13. Tiempo de codificación sobre duración de fichero original conjunto reducido.

3.2.3 Prueba sobre procesamiento posterior.

En esta sección se analiza la influencia de los distintos códecs en el procesamiento automático posterior del habla codificada y descodificada. En concreto, se analiza la influencia de la codificación en el reconocimiento automático del locutor, el reconocimiento automático del idioma, el reconocimiento automático del habla y el filtrado de ruido sin referencia.

Para analizar estos comportamientos se ha contado con herramientas desarrolladas en el grupo ATVS-UAM. En todos los casos se ha puesto el énfasis en comparar las influencias de los códecs más que en conseguir resultados óptimos en cada una de las tareas. En ese sentido se ha tendido a elegir las técnicas usadas de un modo más generalizado en cada uno de los campos, tratando de mantener los sistemas relativamente sencillos.

Todas las técnicas aquí probadas se pueden adaptar a las características del habla procesada por un determinado códec, sin embargo en estas pruebas se ha preferido no realizar esa adaptación a fin de evaluar la degradación introducida por el códec en un sistema automático sin especial adaptación. De este modo es también más justa la comparación.

En todos los casos se comparan los resultados obtenidos con la voz sin codificar y la voz obtenida tras el proceso completo de codificación y descodificación con cada uno de los tres códecs analizados.

3.2.3.1 Reconocimiento automático de locutor

En reconocimiento automático del locutor se ha empleado la técnica GMM-UBM (Gaussian Mixture Model y Universal Background Model). Esta es la técnica más exitosa en reconocimiento automático del locutor a lo largo de la historia del mismo, y la que mayor desarrollo e implantación ha tenido.

En los últimos cuatro años los resultados de estas técnicas se han visto notablemente mejorados con la aplicación de técnicas de compensación de canal avanzadas. El sistema empleado en esta prueba emplea una de estas novedosas técnicas de compensación de canal denominada Nuissance Attribute Projection (NAP) o proyección de atributos molestos, que consisten en determinar las direcciones de mayor variabilidad inter-canal en el GMM y anularlas.

La prueba se ha realizado sobre los datos de la última evaluación competitiva internacional de reconocimiento de locutor (Speaker Recognition Evaluation, SRE) organizada por el National Institute of Standards and Technology (NIST) en la que ATVS-UAM ha participado en el primer semestre de 2008.

En concreto de la base de datos de evaluación se ha seleccionado el subconjunto correspondiente únicamente a voz telefónica espontánea y con locutores masculinos. Éste subconjunto incluye varios cientos de locutores distintos, varios miles de segmentos de test y varias decenas de miles de enfrentamientos de verificación entre segmento de test y locutor. Dado el tamaño de la prueba, los resultados obtenidos en la misma son bastante significativos, resultando complicado en la actualidad realizar evaluaciones mucho más extensas que la descrita.

En todas las pruebas el modelo del locutor se ha entrenado con una conversación telefónica (un canal de la conversación) con aproximadamente 2.5 minutos (en media) de habla neta del locutor. El entrenamiento siempre se ha realizado con voz sin codificar.

El reconocimiento en todos los casos se ha realizado con una conversación telefónica de aproximadamente 2.5 minutos (en media) de habla neta de un locutor. En el caso del reconocimiento, además de emplear la voz sin codificación empleamos los mismos segmentos de test codificados y descodificados con los distintos codificadores de voz empleados. Como los algoritmos de reconocimiento de locutor empleados están desarrollados para voz telefónica con frecuencia de muestreo de 8 KHz, en el caso de emplear codificadores con mayor ancho de banda se ha procedido a pasar los ficheros a 8 KHz de frecuencia de muestreo antes de realizar el reconocimiento de locutor.

Los resultados se presentan en forma de curvas DET, que representan gráficamente cómo evoluciona el porcentaje de falsa aceptación (FA) frente al de falso rechazo (FR) al modificar el umbral de reconocimiento. El sistema es tanto mejor cuanto más abajo y hacia la izquierda está la curva DET. También se representan en forma de Equal Error Rate (EER), que se calcula como la intersección de la curva DET con la diagonal FA=FR. El EER es el porcentaje en el que se igualan los porcentajes de falso rechazo y de falsa aceptación, y es tanto mejor cuanto menor es. Finalmente también se representan los resultados en forma de Detection Cost Function (DCF). Esta medida, definida por NIST en las evaluaciones de reconocimiento del locutor como:

$$C_{Det} = C_{Miss} \times P_{Miss|Target} \times P_{Target} + C_{FalseAlarm} \times P_{FalseAlarm\ |NonTarget} \times (1 - P_{Target})$$

depende tanto de la probabilidad de falso rechazo ($P_{Miss|Target}$) como de la probabilidad de falsa aceptación ($P_{FalseAlarm|NonTarget}$), que se combinan de acuerdo a unos costes de falso rechazo (C_{Miss}) y falsa aceptación ($C_{FalseAlarm}$), así como de acuerdo a una probabilidad a priori de que el locutor sea el correcto (P_{Target}). En concreto para las evaluaciones NIST se emplean los siguientes valores:

C_{Miss}	$C_{FalseAlarm}$	P_{Target}
10	1	0.01

La siguiente figura representa las curvas DET obtenidas con los distintos codificadores empleados:

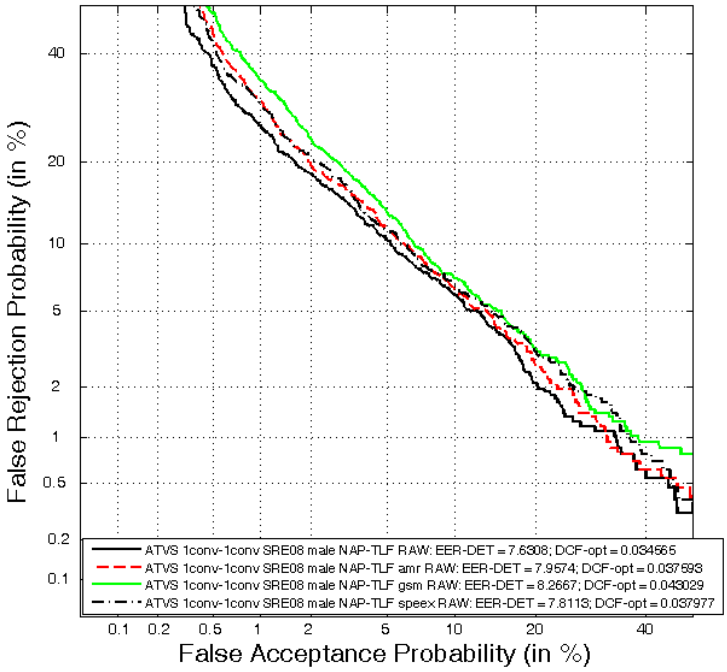


Figura 29. Curvas DET obtenidas en la prueba de reconocimiento de locutor con voz codificada y descodificada con los distintos codificadores así como con voz sin codificar (RAW).

En estas curvas se aprecia que los mejores resultados se consiguen con la voz sin codificar, algo que era de esperar. Entre la voz codificada y descodificada, se aprecia que GSM-FR es el codificador que más afecta a los resultados de reconocimiento de locutor, mientras que AMR-WB y Speex tienen un comportamiento que podría caracterizarse de intermedio. En cualquier caso las curvas DET están muy próximas, lo que demuestra que la influencia de la codificación en los resultados del reconocimiento de idioma es limitada, en parte debido a los sofisticados mecanismos de compensación de la influencia del canal que se emplean en el sistema de reconocimiento del locutor empleado.

Podemos realizar también comparaciones en función del EER y del DCF. Ambas medidas se muestran de forma gráfica en la siguiente figura:

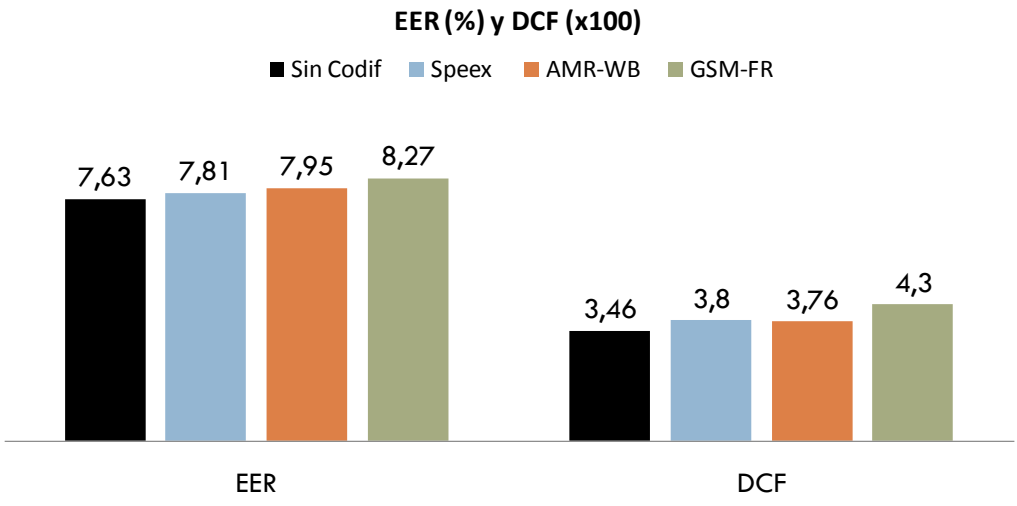


Figura 30. EER y DCF obtenidos en la prueba de reconocimiento de locutor con voz codificada y descodificada con los distintos codificadores así como con voz sin codificar.

Los resultados muestran lo mismo que ya observábamos en la curva DET. Todos los codificadores introducen pequeñas pérdidas en reconocimiento de locutor frente a la opción de no codificar. Estas pérdidas son pequeñas probablemente debido a los mecanismos de compensación de canal empleados en el reconocedor de locutor. GSM-FR es el que mayores pérdidas introduce. AMR-WB y Speex introducen pérdidas muy similares entre sí, y próximas ambas a la mitad de las introducidas por GSM-FR. Estas son las principales conclusiones que podemos obtener respecto a la influencia del códec en el reconocimiento de locutor.

3.2.3.2 Reconocimiento del idioma

En reconocimiento automático del idioma se ha empleado la técnica PPRLM (Parallell Phone Recognition followed by Language Modelling). Esta técnica es la técnica más frecuentemente empleada en reconocimiento de idioma a lo largo de la historia del mismo, y la que ha dado resultados más satisfactorios en general. Debemos decir, no obstante, que en los últimos años se han desarrollado sistemas que igualan o incluso superan esta técnica, así como mejoras sobre esta técnica que aquí no se han tenido en cuenta. El motivo es mantener el sistema lo más estándar posible ya que el objetivo es comparar los resultados obtenidos con y sin codificar, así como con los distintos codificadores. En concreto, el sistema PPRLM empleado utiliza un conjunto de 7 reconocedores fonéticos en los siguientes idiomas: español, inglés, alemán, ruso, francés, árabe y euskera. Una de las ventajas de la técnica PPRLM es que permite reconocer idiomas para los que no se dispone de un reconocedor fonético, por lo que con este conjunto de 7 reconocedores fonéticos se puede reconocer, en principio, cualquier conjunto de idiomas del mundo.

La prueba que hemos realizado utiliza habla telefónica espontánea grabada de conversaciones normales entre personas. Este tipo de habla es muy espontánea y resulta una de las más complicadas de tratar. El habla ha sido obtenida de bases de datos CallFriend. Este tipo de habla era el empleado habitualmente en las evaluaciones NIST hasta 2007. A partir de entonces se pasó a emplear habla también conversacional telefónica pero de otras bases de datos.

La prueba la hemos realizado en 5 idiomas: alemán, árabe, francés inglés y ruso.

La prueba consiste en entrenar un modelo de cada uno de los idiomas a reconocer y posteriormente tratar de detectar si el idioma está o no presente en una grabación.

Para entrenar los modelos de reconocimiento de idioma empleamos 20 horas de voz de cada uno de los idiomas a reconocer. El entrenamiento de los modelos se realiza siempre con voz sin codificar. Una vez seleccionado el codificador a emplear lo lógico sería entrenar los modelos con la voz codificada con el codificador seleccionado, y eso daría lugar a mejoras, pero por ahora nos centramos en seleccionar un codificador y para esta tarea resulta más adecuado entrenar los modelos con voz sin codificar.

Para el reconocimiento se emplean 7 horas de voz por cada idioma, divididas en segmentos de 30 segundos de duración (para decidir si el idioma está presente o no se emplean únicamente 30 segundos). Estos segmentos de 30 segundos se emplean sin codificar o codificados con los tres codificadores estudiados. En el caso de los códecs de ancho de banda extendido, antes de reconocer se pasan los audios codificados y descodificados a 8 KHz de frecuencia de muestreo.

Siguiendo en este caso también los métodos de evaluación establecidos por NIST en las evaluaciones competitivas internacionales de reconocimiento de idioma (Language Recognition Evaluation, LRE), evaluamos el reconocimiento de idioma como una tarea de detección (debemos decidir si un idioma está presente en un segmento de 30 segundos de audio), al igual que

hacíamos con el reconocimiento de locutor. De este modo, en reconocimiento de idioma también presentaremos curvas DET y EERs. El DCF aunque aparece en las gráficas aquí presentadas no se emplea en reconocimiento de idioma como tal.

Las siguientes gráficas muestran las curvas DET obtenidas por cada uno de los idiomas estudiados con cada uno de los diferentes códecs analizados.

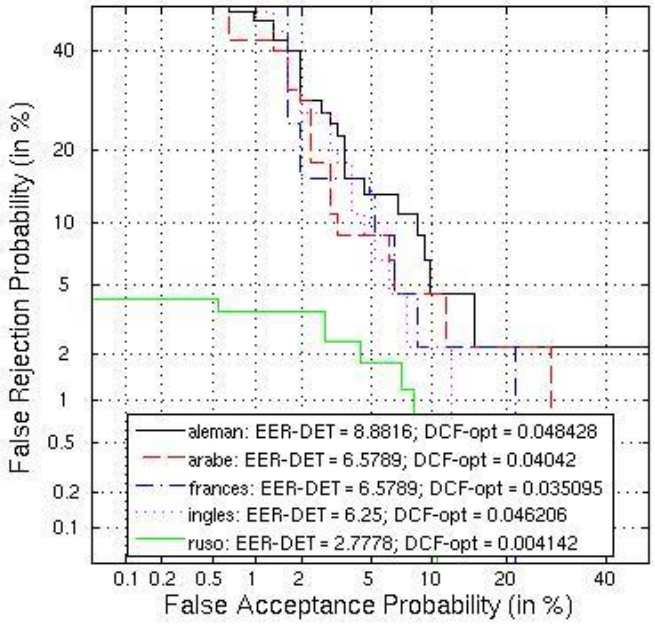


Figura 31. Curvas DET obtenidas en la prueba de reconocimiento de idioma con voz sin codificar

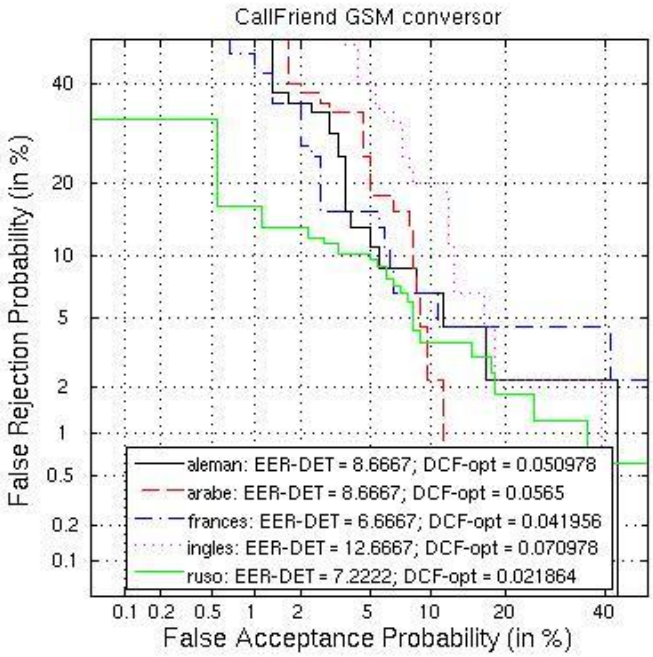


Figura 32. Curvas DET obtenidas en la prueba de reconocimiento de idioma con voz codificada y descodificada con GSM-FR

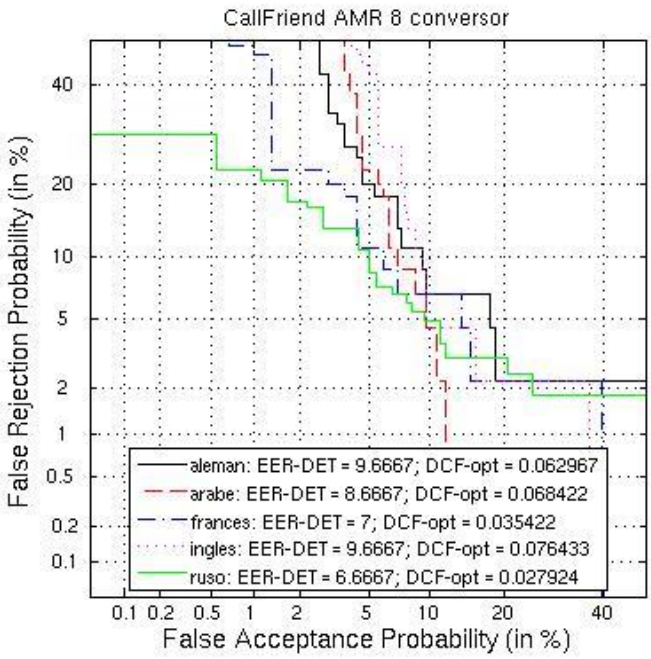


Figura 33. Curvas DET obtenidas en la prueba de reconocimiento de idioma con voz codificada y descodificada con AMR-WB (pasada luego a 8KHz).

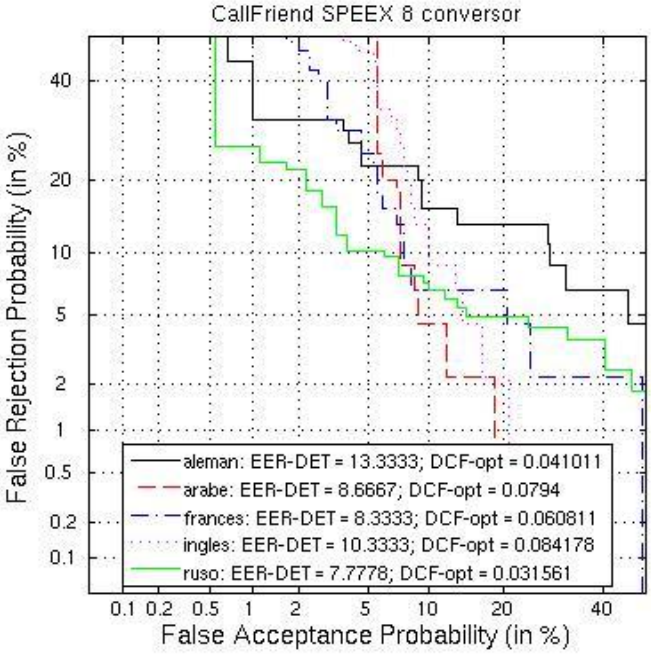


Figura 34. Curvas DET obtenidas en la prueba de reconocimiento de idioma con voz codificada y descodificada con Speex (pasada luego a 8KHz)

Dado que es necesario generar una curva DET por cada uno de los idiomas a analizar y que es complicado representarlas todas en la misma figura resulta complicado analizar los resultados en forma de curvas DET. La siguiente gráfica representa los resultados en forma de EERs:

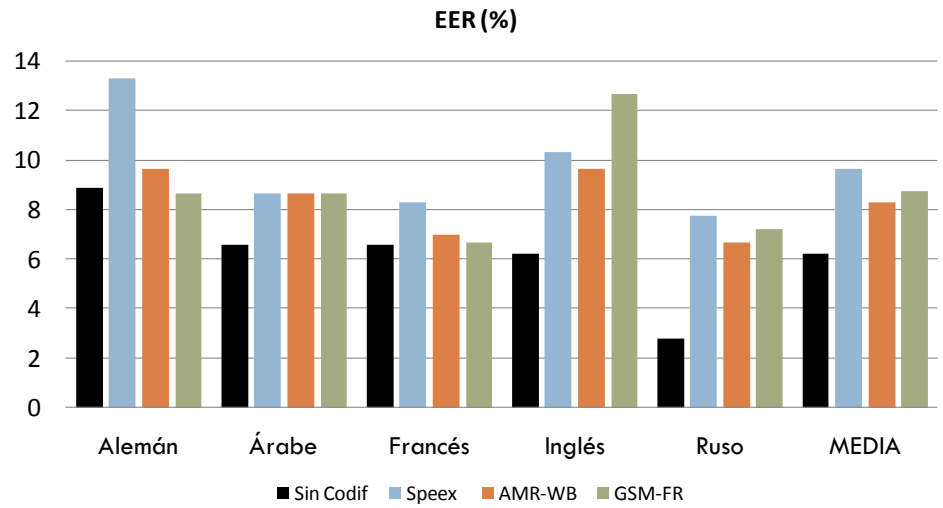


Figura 35. EER obtenidos en la prueba de reconocimiento de idioma con voz codificada y descodificada con los distintos codificadores así como voz sin codificar

La Figura 35 muestra más claramente lo siguiente:

- En general todos los códecs tienden a degradar los resultados. Esto en parte es debido a que los modelos de cada idioma se han entrenado con voz sin codificar por lo que la voz más parecida a los datos de entrenamiento es la voz sin codificar.
- El efecto de los códecs en reconocimiento de idioma es muy dependiente del idioma que se pretenda reconocer. Así en el caso del alemán el Speex es el peor y GSM-FR el mejor, mientras que en el inglés el GSM es el peor y AMR-WB el mejor.
- En general AMR-WB es el que mejores resultados obtiene en reconocimiento de idioma.

3.2.3.3 Reconocimiento del habla

Para esta prueba se ha empleado un reconocedor fonético en lugar de un reconocedor de palabras completo. La diferencia básica entre uno y otro es que un reconocedor de palabras está compuesto por un reconocedor fonético más un conjunto de restricciones léxicas (palabras que puede reconocer) y un conjunto de restricciones gramaticales (secuencias de palabras que puede reconocer). Este conjunto de restricciones facilita el reconocimiento porque limita el número de posibles secuencias de fonemas reconocidas. El reconocimiento de fonemas sin restricciones adicionales es una tarea muy compleja en la que los mejores resultados publicados en inglés americano sobre habla leída, sin ruidos y microfónica rondan el 75% de precisión fonética. Esto da una idea de la dificultad de la tarea. En esta misma tarea, con un reconocedor de palabras se obtienen fácilmente tasas de reconocimiento de palabras superiores al 95%.

Las pruebas de reconocimiento fonético se emplean habitualmente para comprobar la influencia del ruido o del canal en el reconocimiento, evitando introducir efectos de las restricciones léxicas y gramaticales que son muy dependientes del vocabulario y de la gramática que emplea el reconocedor. De este modo se evalúa únicamente lo que se conoce como la parte del modelado acústico del reconocedor (la parte que modela cómo suena cada fonema).

En la prueba que se ha realizado se ha empleado la base de datos ALBAYZIN. Se trata de una base de datos en español, de voz microfónica capturada con un micrófono de habla cercana y sin ruidos. La voz es producida leyendo una serie de frases escritas (no es espontánea en absoluto).

Para realizar la prueba se ha entrenado un conjunto de 23 modelos fonéticos para los distintos fonemas del castellano, más 3 modelos para silencios. Estos modelos son modelos ocultos de Markov (Hidden Markov Models, HMMs), que son los modelos empleados habitualmente en todos los reconocedores de voz comerciales. Los modelos son relativamente sencillos, pues son independientes del contexto (usamos el mismo modelos para el fonema /a/ independientemente de los fonemas que lo preceden o lo siguen) y también independientes del locutor. Los modelos utilizan 3 estados para modelar cada fonema y modelan cada estado como un modelo de mezcla de gaussianas (GMM) como los empleados en reconocimiento de locutor, pero con 65 Gaussianas únicamente. Con esto el total de Gaussianas empleadas en todos los estados de todos los modelos es de 5070.

Estos modelos se entrenan sobre el subconjunto de entrenamiento definido en la base de datos ALBAYZIN, que consta de 164 locutores que pronuncian un total de 4800 frases cortas. El entrenamiento se ha realizado únicamente con voz sin codificar, como en las pruebas anteriores.

Una vez entrenados los modelos se realiza la prueba de reconocimiento fonético sobre el subconjunto de evaluación definido en ALBAYZIN, que consta de 40 locutores que pronuncian un total de 2000 frases. La prueba de reconocimiento se ha realizado tanto con voz sin codificar como con voz codificada y descodificada con cada uno de los tres códecs analizados.

Tanto en entrenamiento como en evaluación es necesario parametrizar la voz, es decir convertir la voz en un conjunto de vectores de parámetros que resumen las características de la voz. En ambos casos se ha empleado una parametrización definida por el estándar ETSI ES 202 050. Este estándar de la ETSI (European Telecummunications Standards Institute) define una parametrización compuesta por Mel-Frequency Cepstral Coefficients (MFCCs), que es la parametrización empleada por casi todos los reconocedores de voz comerciales. Sin embargo, el estándar incluye una serie de mecanismos para el control del ruido ambiente y de la variabilidad del canal, como son un doble filtrado de Wiener y una ecualización ciega de canal.

Para evaluar los resultados se han empleado dos medidas empleadas habitualmente en reconocimiento fonético:

- El porcentaje de fonemas reconocidos correctamente (%CORR). Se calcula como el número de fonemas correctamente reconocidos dividido entre el número de fonemas originales.
- La precisión fonética (phonetic accuracy, %ACC). Se calcula como el número de fonemas correctamente reconocidos menos el número de fonemas insertados por error, dividido entre el número de fonemas originales. Esta segunda medida es más adecuada ya que tiene en cuenta los errores de inserción, que de otro modo no se penalizan.

La siguiente figura muestra los resultados obtenidos en esta prueba.

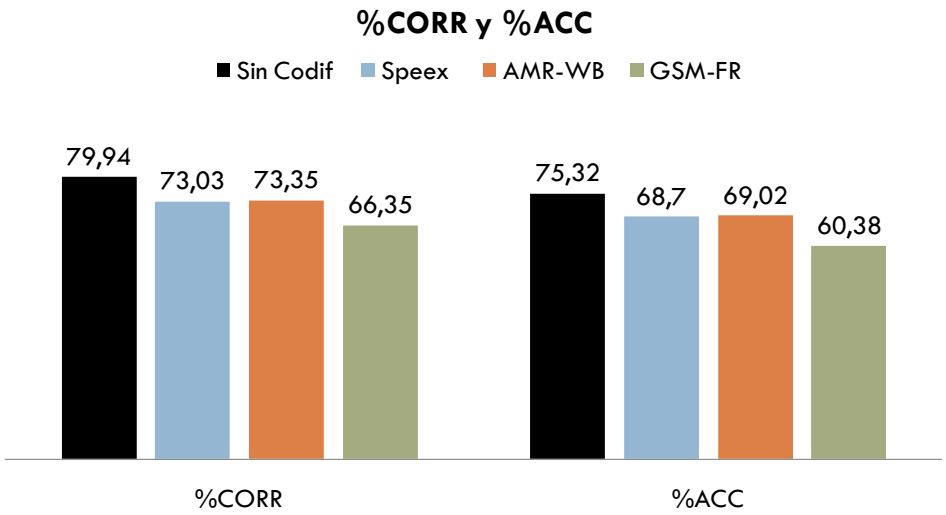


Figura 36. Resultados de reconocimiento fonético (%CORR y %ACC) obtenidos con voz sin codificar y codificada con los tres códecs analizados

Los resultados muestran un comportamiento similar al que observábamos en reconocimiento de locutor:

- Al codificar la voz siempre se producen degradaciones en el reconocimiento fonético, que llegan a ser muy importantes en el caso del GSM, pasando de un 75% de precisión a un 60%. Estas degradaciones pueden reducirse parcialmente entrenando los modelos con voz codificada.
- Los códecs AMR-WB y Speex afectan menos al resultado del reconocimiento fonético que el GSM, produciendo degradaciones intermedias entre el caso sin codificar y el GSM.
- Los resultados obtenidos por los códecs AMR-WB y Speex son muy similares entre sí.

3.2.3.4 Filtrado del ruido sin referencia

Para evaluar el impacto del codificador en las técnicas de filtrado de ruido sin referencia se ha realizado una prueba empleando el mecanismo más estándar de filtrado de ruido sin referencia: el filtrado de Wiener.

La prueba se ha realizado sobre el conjunto de datos reducido sobre el que se realizaron las medidas de calidad de la voz codificada. Este conjunto de datos consiste en 18 grabaciones de voz de la base de datos ALBAYZIN con ruido (blanco y Gaussiano) añadido.

La prueba consiste en codificar y descodificar la voz (con ruido añadido) con cada uno de los códecs, a continuación medir la SNR (Signal to Noise Ratio, Relación Señal a Ruido) de la voz con ruido sin codificar y codificada y descodificada con cada uno de los códecs. De este modo tenemos la SNR de la voz sin codificar y tras pasar por cada uno de los codificadores. A partir de este momento procesamos la voz tanto sin codificar como tras pasar por cada uno de los codificadores mediante un filtro de Wiener y evaluamos cómo aumenta la SNR con este proceso.

A continuación mostramos los resultados obtenidos en esta prueba:

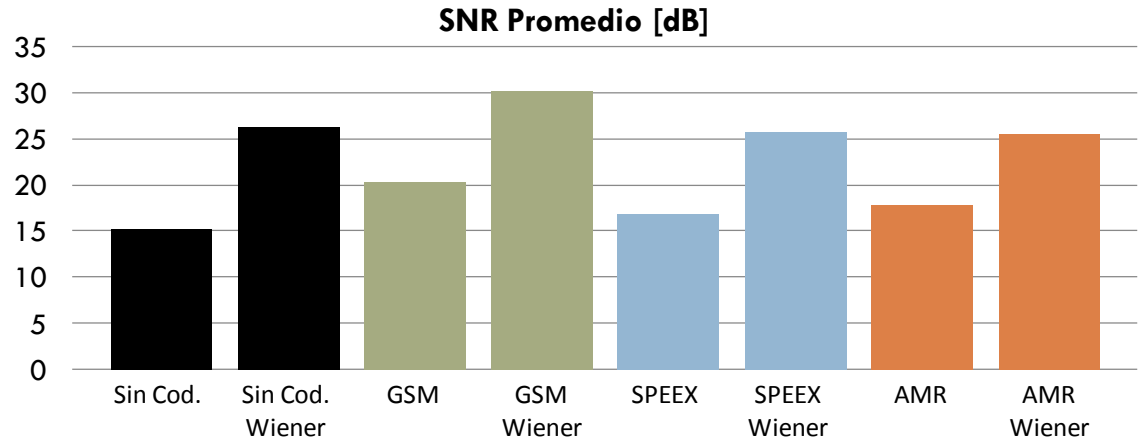


Figura 37. Relación Señal a Ruido (SNR) obtenida antes y después del filtro Wiener con voz sin codificar y codificada con cada uno de los tres códecs analizados.

	SIN CODIFICAR	GSM-FR 6.10 13 Kbps	SPEEX 1.2b3 12,8 Kbps	AMR-WB v7.0.0 12,65 Kbps
Sin filtrar	18.98	20.39	21.24	20.52
Filtrado	28.93	30.24	29.34	28.22
Mejora	9.95	9.86	8.10	7.69

Tabla 14. Valores numéricos correspondientes a la Figura 17 y mejora de la SNR con el filtrado de Wiener.

Observando los resultados de esta prueba podemos extraer las siguientes conclusiones:

- El mero proceso de codificar y descodificar la voz tiende a producir una mejora en la relación señal a ruido de entre 1 y 2 dB. Idealmente el códec no debería modificar la relación señal a ruido, pero parece que todos eliminan algo de ruido por defecto.
- En general la mejora producida en la SNR sobre la voz codificada al aplicar el filtrado de Wiener es algo inferior (hasta unos 2 dB inferior) a la conseguida sobre la voz sin codificar.
- Los efectos anteriores se compensan entre sí, de modo que al final la SNR obtenida tras el filtrado de Wiener es muy parecida sobre la voz codificada y sobre la voz sin codificar, variando la SNR entre -1 dB y +1 dB sobre la obtenida con la voz sin codificar.
- La conclusión principal es que el efecto del códec sobre los mecanismos de filtrado de ruido sin referencia (en particular sobre el filtrado de Wiener) es muy pequeño y prácticamente despreciable.

3.3 Conclusiones de la evaluación.

3.3.1 Sobre las medidas directas de calidad y coste

De acuerdo con las medidas de calidad directa parece claro que Speex es el códec que proporciona la mayor calidad de los tres analizados. AMR-WB le sigue de cerca salvo con bajo volumen, donde Speex es muy superior. Por otro lado, AMR-WB y Speex son claramente superiores a GSM-FR, incluso trabajando a tasas binarias inferiores a la de GSM-FR.

Además, tanto AMR-WB como Speex ofrecen ventajas de orden práctico, por ejemplo, ambos permiten codificar a 16 kHz y Speex incluso a 32 kHz. También ambos permiten codificar a tasas binarias variables para ajustarse a distintos requisitos de espacio o ancho de banda.

En cuanto a coste computacional, y aunque esta medida se deba tomar con cautela porque depende del grado de optimización del código utilizado, en las pruebas realizadas Speex se mostraba sólo un poco más costoso que GSM-FR, mientras que AMR-WB parece ser mucho más costoso.

3.3.2 Sobre el procesamiento posterior

En cuanto al procesamiento posterior, los resultados que se han obtenido son muy similares en reconocimiento de locutor y en reconocimiento del habla. En ambos casos, AMR-WB y Speex tienen un comportamiento casi idéntico e intermedio entre el mejor caso (voz sin codificar) y el peor (GSM-FR).

En reconocimiento de idioma se observa que los resultados son muy variables dependiendo del idioma que se intente reconocer, pero en general los resultados son favorables al codificador AMR-WB.

Finalmente, en las pruebas realizadas para analizar el comportamiento de los tres códecs en filtrado de ruido sin referencia (filtrado de Wiener) los resultados han mostrado un comportamiento prácticamente idéntico de los tres códecs analizados, que tienen en todos los casos una influencia muy pequeña en la reducción de ruido conseguida con el filtrado de Wiener.

En definitiva, en todo este análisis AMR-WB y Speex empatan, salvo en reconocimiento de idioma, donde AMR-WB resulta el ganador.

3.3.3 Conclusión final de la prueba

Teniendo en cuenta todas las comparaciones realizadas entre los tres codificadores elegidos parece que **Speex es la opción más adecuada para servir de códec base para el desarrollo del códec propietario** subsiguiente. Es el códec que parece aportar mayor calidad, tiene un comportamiento similar al códec AMR-WB en procesamiento posterior (salvo en reconocimiento de idioma), y es mucho menos costoso computacionalmente.

4 DESARROLLO VERSIONES CÓDEC

Una vez seleccionado el códec Speex como la opción más adecuada, debemos modificarlo para convertirlo en un códec propietario, portándolo al mismo tiempo a Linux y al sistema embebido MIPSEL (400 MHz), cumpliendo unos requisitos de tiempo de codificación inferior a tiempo real.

4.1 Diseño y desarrollo

En este apartado veremos en primer lugar las modificaciones llevadas a cabo para conseguir ejecutar el códec Speex original en el sistema embebido (carente de unidad de coma flotante) cumpliendo los requisitos de tiempo establecidos por el cliente. A continuación veremos los cambios introducidos en el codificador para generar un formato propietario tanto para Linux como para el sistema embebido, el desarrollo del códec-driver para el Audio Compression Manager de Microsoft y por último la readaptación del códec de Linux y sistema embebido debido a las limitaciones introducidas por el Audio Compression Manager.

4.1.1 Desarrollo códec Linux

4.1.1.1 Implementación en el sistema embebido

El primer reto consiste por tanto en portar el codificador elegido al sistema embebido que cuenta con las siguientes características hardware:

- Microprocesador MIPSEL32 corriendo a 395 Bogomips.
- 128 MB de RAM.
- 32 MB de memoria flash donde reside el cargador de arranque Yamon (Bootloader), núcleo (Kernel) y sistema de ficheros (RootFS).
- **No dispone de unidad de punto flotante (Floating Point Unit).**
- Unidad de gestión de memoria MMU (Memory Management Unit) muy sencilla.

Además se nos entregó con el siguiente S.O. instalado:

- Kernel 2.4.21 proporcionado por el fabricante (fuentes incluidas)
- Distribución basada en busybox, con libc 2.3.2
- Herramientas de adquisición de audio

El toolchain proporcionado por el fabricante era muy antiguo y el codificador Speex original no compilaba en él. Tras parchear el toolchain se consiguió compilar con éxito el codificador Speex original.

Las primeras pruebas de tiempos realizadas en el dispositivo mostraban una velocidad de codificación cercana a 0.1x con el 99% sistema, lo que supone un tiempo de codificación 10 veces más lento que tiempo real.

Además las herramientas originales nos impedían depurar las causas al no existir ninguna para hacer análisis de caja negra (llamadas de sistema y/o librerías externas) ni permitir el uso de elementos para hacer profiling (gprof).

Fue necesario por tanto compilar un toolchain más actual. OpenEmbedded soportaba el dispositivo y era capaz de ofrecer un toolchain para él basado en versiones más modernas de gcc/libc.

Para realizar un estudio de las causas se crea también un programa por encima de libspeex que simplemente codifica audio localizado en RAM para evitar considerar los tiempos de entrada/salida y se estudia con strace y gprof los lugares en los que se pierde mayor cantidad de tiempo.

Las conclusiones a las que se llegaron mediante este estudio fueron:

- Una mala configuración del compilador puede dar resultados mucho más lentos. Se mejora la compilación modificando los makefiles del codificador Speex.
- Hay que tener especial cuidado al revisar que el código a optimizar no hace uso del emulador de coma flotante del sistema. El Speex seguía empleándolo a pesar de indicarle que no lo usase: se cambia el código para que no emplee dicho emulador.

Estas dos mejoras conseguían acelerar notablemente el funcionamiento en el sistema embebido y además eran portables al toolchain antiguo (modificado).

4.1.1.2 Diseño formato de archivo

Para conseguir diferenciar el formato propietario, y hacer que los archivos generados por el codificador no pudieran ser decodificados por el codificador Speex original, se introdujeron los siguientes cambios en el formato del codificador ATVS-1.0:

- Se modificó la cabecera, eliminando los parámetros innecesarios y desordenando aquellos que permanecían.

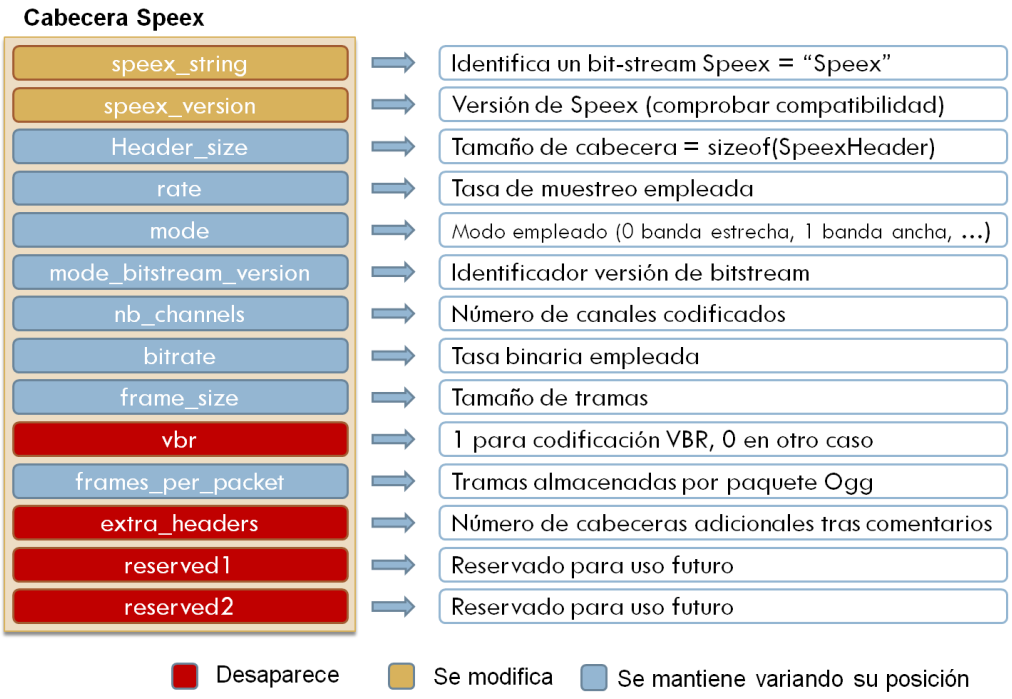


Figura 38. Modificaciones sobre cabecera Speex

- Se modificó la estructura de "paquete" eliminando el parámetro begin_of_stream por ser su información redundante. El control se realiza ahora a través del número de paquete.

- Se modificaron los programas de codificación y descodificación (speexenc y speexdec, ahora ATVSenc y ATVSdec) eliminando la estructura contenedora ogg así como la

utilización de las funciones de la librería ogg para la composición y almacenamiento del bitstream resultante. En su lugar se ha definido un formato propio de fichero y se han incluido diversas funciones como las encargadas de leer y escribir los paquetes en el nuevo formato y la desordenación y ordenación del bitstream.

- Se han realizado las modificaciones necesarias en la librería libspeex (ahora libatvs) para su adaptación a las nuevas estructuras de cabecera, datos y nuevo formato de fichero.

El formato de archivo resultado de una codificación con el códec ATVS 1.0 sigue el siguiente esquema:

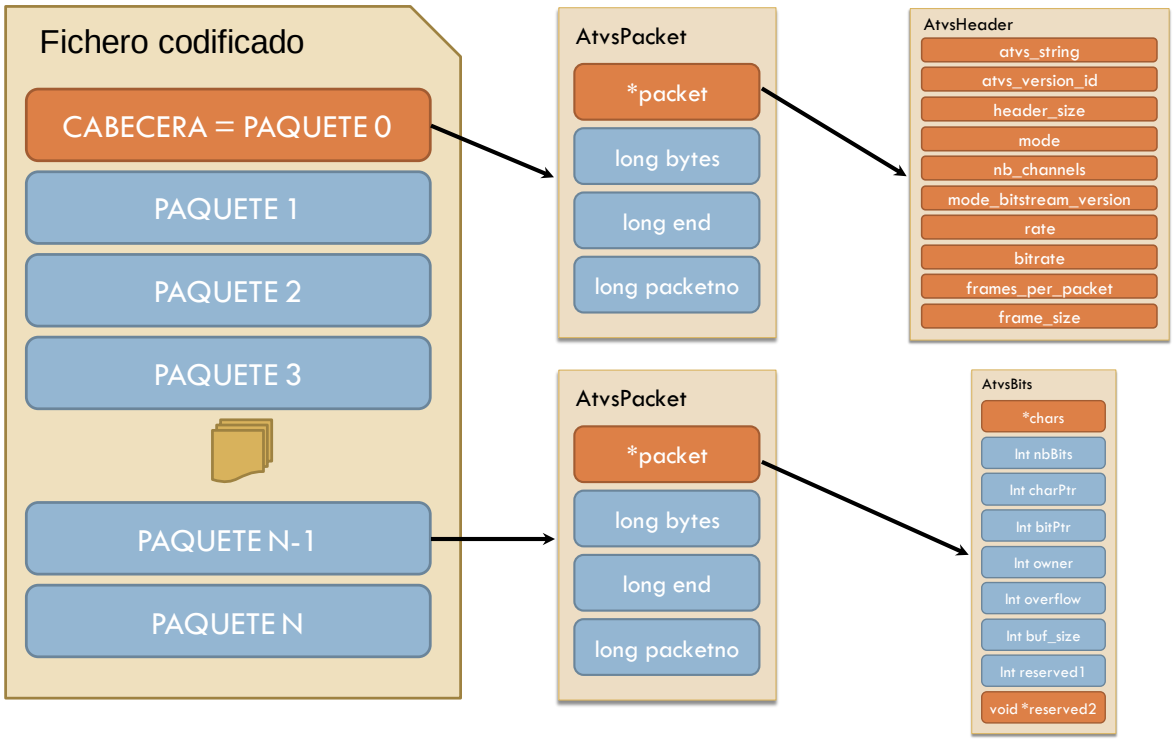


Figura 39. Formato de archivo de salida codificador versión 1.0

Un fichero codificado mediante ATVSenc está formado por una sucesión de paquetes, desde el primero en el que se incluye la cabecera (paquete número 0) hasta el último (paquete N). Cada uno de esos paquetes son elementos del tipo estructura de datos ATVSPacket.

En el primer paquete, el puntero *packet apunta a una estructura de cabecera (AtvsHeader) mientras que para el resto de paquetes apuntará a una estructura ATVSBits cuyo puntero *chars apunta a su vez a los bits codificados pertenecientes a la trama o tramas codificadas en ese paquete.

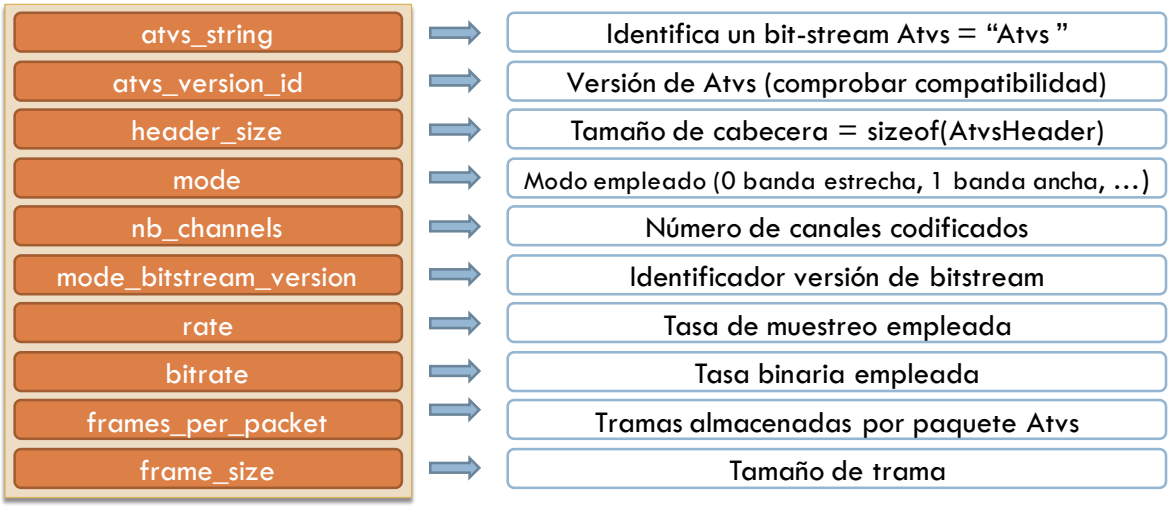


Figura 40. Formato de cabecera. Estructura AtvsHeader versión 1.0

Sobre la estructura ATVSBits, se lleva a cabo una desordenación a nivel de Byte una vez finalizada su creación y antes de ser insertada en el paquete en el que será almacenado en el fichero. Se desordena por tanto la información "raw" y cursores y demás elementos presentes en la estructura.

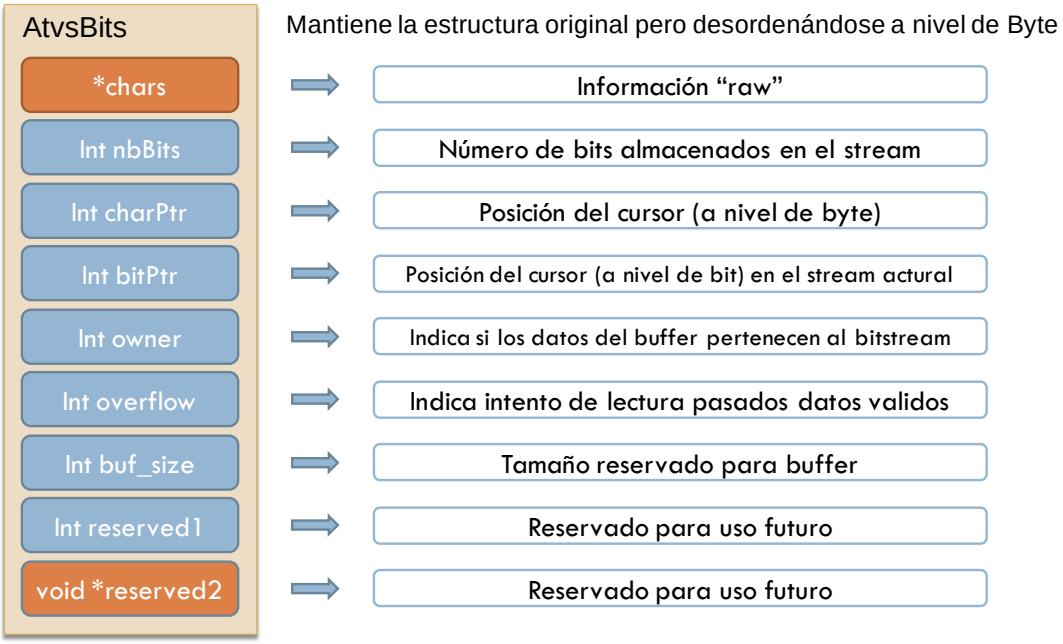


Figura 41. Formato de bit-stream. Estructura AtvsBits versión 1.0

Para esta desordenación se ha utilizado un algoritmo determinista sencillo de entrelazado. La ordenación del bitstream en el decodificador se lleva a cabo tras la lectura del paquete y antes de su procesamiento.

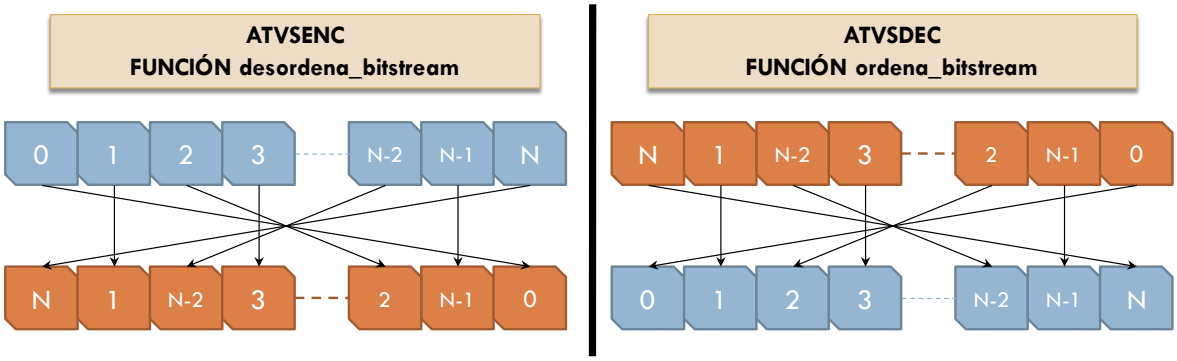


Figura 42. Desordenación estructura AtvsBits

4.1.2 Desarrollo Códec-Driver ACM

4.1.2.1 Introducción al Audio Compression Manager

La interfaz de programación de aplicaciones (API) Win32 de Microsoft proporciona un conjunto de funciones estándar para dispositivos de sonido.

Estas funciones gestionan las llamadas a los drivers encargados del control de dichos dispositivos. Para controlar los dispositivos instalados, el sistema cuenta con un módulo llamado mapeador (mapper) que se encarga de vigilar la interfaz de los drivers y de actuar como un intermediario entre el sistema y los dispositivos instalados en él. Es por tanto el responsable de satisfacer las peticiones de acceso a los dispositivos por parte de las aplicaciones, encontrando aquel dispositivo que cumpla las necesidades de sonido requeridas por la aplicación. El sistema incluye por defecto mapeadores para drivers estándar como el de forma de onda (waveform-audio) o MIDI (Musical Instrument Digital Interface).

El ACM es una extensión del sistema multimedia básico y se instala como un mapeador adicional. Al igual que el mapeador estándar, vigila la interfaz de los drivers para dispositivos de audio lo que le permite decodificar o codificar información de audio antes de pasarla hacia o desde un driver de dispositivo.

La principal diferencia entre el ACM y el mapeador estándar del sistema es que el ACM es capaz de buscar un dispositivo de sonido que soporte cierto formato o una combinación de un dispositivo de audio y un códec que permita codificar o decodificar dicha información.

Más concretamente, cuando una aplicación lanza una petición al sistema de apertura de un dispositivo de audio para entrada o salida, en dicha petición especifica el formato y el dispositivo deseados. Cuando el dispositivo deseado es el mapeador, éste debe encontrar entre los instalados otro capaz de manejar el formato también indicado. Si el ACM no encuentra tal dispositivo, buscará una combinación de un dispositivo de audio y un códec que en conjunto permitan manejar el formato deseado. Esto permitirá satisfacer la petición de la aplicación aunque no exista ningún dispositivo en el sistema que soporte por si mismo ese formato.

Además de conversión entre formatos, el Audio Compression Manager añade soporte a nivel de sistema para otros servicios. Los servicios que soporta son:

- Compresión y descompresión de audio de manera transparente.
- Selección de formato de información de sonido.
- Selección de filtrado de información de sonido.
- Conversión de formato de información de sonido.
- Filtrado de información de sonido.

4.1.2.2 Tipos de controladores

Como hemos comentado, el ACM intercepta las peticiones de apertura de dispositivos del mapeador original. Una vez interceptadas, el ACM puede llevar a cabo distintas tareas de procesado de la información de audio. Por ejemplo la inserción de un codificador o decodificador en la secuencia. Para ello, el ACM cuenta con distintos tipos de controladores:

- **Controladores de compresión y descompresión (códecs):** Los controladores de compresión y descompresión transforman tanto el formato como el tipo de información. Por ejemplo, un códec puede transformar un fichero de codificación PCM (Pulse code modulation) en otro con codificación ADPCM (Adaptative Differential Pulse Code Modulation). El driver AtvsACM desarrollado es un controlador de este tipo.

- **Controladores de conversión de formato:** Los controladores de conversión de formato cambian el formato pero no el tipo de información (conservan la codificación original). Por ejemplo puede transformar información de 16 bits y 44 kHz a 8 bits y 44 kHz.

- **Controladores de filtros:** Los controladores de filtros no cambian el formato de la información pero modifican la forma de onda en cierto aspecto. Por ejemplo, un filtro podría combinar un flujo de información y una versión retrasada (eco) de sí mismo.

Un único driver ACM o etiqueta de filtrado o de formato en un driver puede estar formado por una combinación de los tipos de drivers anteriores.

4.1.2.3 Esquema simplificado de conversión de formato

Imaginemos que deseamos reproducir en una aplicación un fichero de audio codificado Atvs (queremos obtener una salida de forma de onda). El ACM le pasará al conversor (códec) cada buffer de información según le llega. El conversor decodificará la información y la devolverá al ACM en un buffer llamado “de sombra”. El ACM le pasa ahora esta información decodificada al driver de forma de onda. El ACM reserva los buffers de sombra cuando recibe un mensaje de preparación.

Si lo que deseamos es convertir una forma de onda en un fichero Atvs, el ACM pasará los buffers de sombra vacíos al driver. Una tarea en segundo plano le enviará una notificación una vez que el driver haya llenado el buffer de sombra. Entonces el ACM pasa dichos buffers al driver para su codificación. Una vez completada, el driver le entrega la información a la aplicación.

De manera más concreta y dejando a un lado los mensajes de configuración del códec, una solicitud de conversión de formato a nuestro driver seguirá el siguiente esquema:

- El sistema o la aplicación interesada en reproducir o convertir desde o hacia el formato Atvs producirá el mensaje DRV_OPEN. El ACM, funcionando como mapeador, detectará esta petición y llamará a la función acmDriverOpen que se encargará de la apertura del driver adecuado (en nuestro caso Atvs-2.0) como preparación a los trabajos de conversión.

- A continuación, si la apertura del driver fue correcta se producirá un mensaje ACMDM_STREAM_OPEN. Al interceptarlo ACM lanzará la función acmdStreamOpen que iniciará un nuevo flujo de conversión, determinará la rutina adecuada de conversión y reservará el buffer de tamaño necesario para la conversión.

- Si la apertura del flujo también se realizó de manera adecuada, a continuación se producirán uno o varios mensajes ACMDM_STREAM_CONVERT. ACM iniciará por cada uno de ellos la función acmdStreamConvert que se encargará de la conversión de cada una de las partes del flujo de datos. (Utilizando la rutina adecuada determinada por acmdStreamOpen).

- Con la finalización del flujo de información la aplicación lanzará un mensaje ACMDM_STREAM_CLOSE. ACM llamará a la función acmdStreamClose que se encargará de liberar los recursos utilizados durante la conversión.

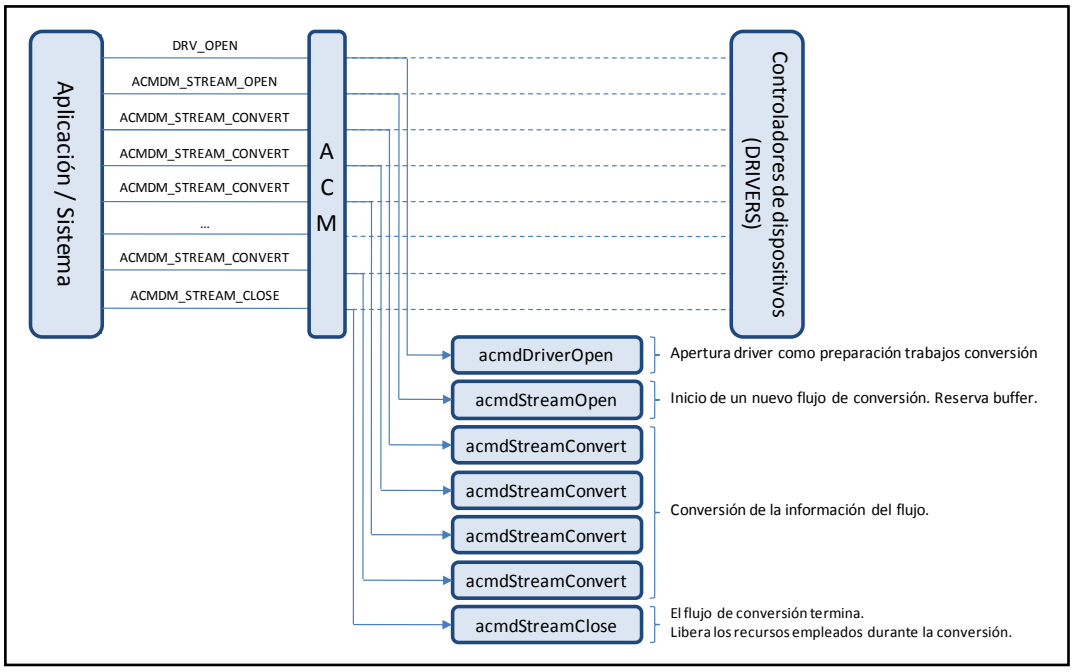


Figura 43. Esquema simplificado de conversión de formato.

4.1.2.4 Introducción al formato WAV

El formato de archivo WAV es un subconjunto de la especificación RIFF (Resource interchange file format) de Microsoft. Este formato RIFF o formato de archivo informático para intercambio de recursos, es un formato genérico que almacena información en bloques etiquetados. Fue presentado por Microsoft e IBM en 1991 y Microsoft lo expuso como el formato predeterminado de los archivos multimedia de Windows 3.1. Está basado en el formato de archivo para intercambio (IFF) de Electronic Arts, que se presentó en 1985. La diferencia consiste en que los enteros de más de un byte se representan en el sistema little-endian de la serie de procesadores 80x86 que se utiliza en PC compatibles IBM, en lugar del sistema big-endian de la serie de procesadores 68k presentes en los computadores Amiga y Apple Macintosh, donde los archivos IFF eran muy frecuentes.

Además del formato WAV (audio de Windows) otros de los formatos más comunes que emplean RIFF como base son:

- AVI (audio y vídeo de Windows).
- RMI ("archivo MIDI basado en RIFF" de Windows).
- CDR (archivo de gráficos vectoriales de CorelDRAW).

Los archivos RIFF están formados en su totalidad por "bloques", cada uno de ellos con su propia cabecera y bytes de datos. Ésta organización permite a los programas que no utilizan o pueden reconocer ciertos bloques poder saltarlos y continuar procesando aquellos que sí son capaces de reconocer.

El formato general se mantiene respecto al IFF, salvo por el orden de bytes, como se expuso anteriormente, y el significado distinto de los nombres de los bloques. Todos los bloques tienen la siguiente estructura:

- 4 bytes: un identificador ASCII del bloque, por ejemplo "fmt " o "data".
- 4 bytes: un entero sin signo de 32 bits, little-endian, con la longitud del bloque (sin contar este mismo campo y el identificador del bloque).
- Campo de longitud variable: la información del bloque, del tamaño especificado en el campo anterior.
- Un byte de relleno, si la longitud del bloque no es par.

Dos identificadores, "RIFF" y "LIST", introducen un bloque que puede a su vez contener subbloques (por ejemplo, los bloques “fmt” y “data” son en realidad subbloques del bloque “RIFF”). Su información de bloque, tras el identificador y la longitud, tiene la siguiente composición:

- 4 bytes: un identificador ASCII para este bloque en particular (en el caso del bloque RIFF: para el archivo completo, como "AVI " o "WAV").
- resto: subbloques.

Un punto a tener en cuenta es que los bloques RIFF deben estar alineados por palabras. Esto significa que su longitud total debe ser un múltiplo de 2 bytes. Si uno de los bloques contiene un número impar de bytes de datos, perdiendo así la alineación por palabras, se debe añadir un byte de relleno de valor cero al final del último byte de datos. Éste byte de relleno no se cuenta en el tamaño de bloque, por lo que los programas deben siempre ajustar el tamaño de cabecera a valores alineados de palabra para calcular de manera correcta el tamaño de dicho bloque.

Cabecera WAV:

La cabecera de los ficheros wav sigue el estándar RIFF anteriormente descrito. Los primeros 8 bytes los componen la cabecera del bloque RIFF que contiene el identificador “RIFF” y un tamaño de bloque igual al tamaño total del archivo menos los 8 bytes utilizados en la cabecera. Los 4 primeros bytes de datos del bloque RIFF determinan el tipo de recurso almacenado en el bloque. Los ficheros wav siempre utilizan el identificador “WAV”. Tras éste identificador vendrán los distintos bloques del fichero wav que definen la forma de onda.

Descripción	Offset	Tamaño	Valor
ID de bloque	0x00	4	“RIFF”
Tamaño de bloque	0x04	4	Tamaño de archivo – 8
ID tipo de RIFF	0x08	4	“WAV”
Bloques WAV	0x10	-	-

Tabla 15. Bloques de cabecera WAV

Bloques WAV:

Aunque existen varios tipos de bloques definidos para los ficheros WAV, muchos de ellos contienen únicamente dos de ellos: el de formato y el de datos.

Estos dos bloques son los mínimos necesarios para describir tanto el formato de las muestras de audio digital como las propias muestras. Aunque no está indicado en la especificación oficial de

los ficheros wav, es una buena práctica colocar el bloque de formato antes del de datos. Muchos programas esperan que los bloques estén almacenados en este orden y adquiere especial importancia cuando se transmite audio digital desde una fuente lenta y lineal como internet. Si el bloque de formato se incluyese tras los datos, deberíamos esperar a recibir todos los datos y luego el formato para poder reproducirlos de manera correcta. Todos los bloques RIFF y por tanto también los WAV son almacenados siguiendo este formato:

Descripción	Offset	Tamaño
ID de bloque	0x00	4
Tamaño de bloque	0x04	4
Bytes de Datos	0x08	-

Tabla 16. Formato bloques RIFF

Bloque de Formato "fmt":

El bloque de formato contiene información acerca de cómo ha sido almacenada la información del sonido y como debe ser reproducida incluyendo el tipo de compresión utilizada, el número de canales, frecuencia de muestreo, bits por muestra y otros atributos.

Descripción	Offset	Tamaño	Valor
ID de bloque	0x00	4	"fmt "
Tamaño de bloque	0x04	4	16 + bytes de formato adicionales
Etiqueta de formato	0x08	2	1 – 65.535
Número de canales	0x0a	2	1 – 65.535
Frecuencia de muestreo	0x0c	4	1 – 0xFFFFFFFF
Bytes por segundo en media	0x10	4	1 – 0xFFFFFFFF
Alineamiento de bloque	0x14	2	1 – 65.535
Bits por muestra	0x16	2	1 – 65.535
Bytes de formato adicionales	0x18	2	1 – 65.535
Bytes extras	0x1a	-	-

Tabla 17. Formato bloques "fmt"

Etiqueta de formato: Especifica el tipo de compresión empleada en la información incluida en el bloque wav. Los códigos más comunes son:

Código	Descripción
0 (0x0000)	Desconocido
1 (0x0001)	PCM / sin compresión
2 (0x0002)	Microsoft ADPCM
6 (0x0006)	ITU G.711 a-law
7 (0x0007)	ITU G.711 Âμ-law
17 (0x0011)	IMA ADPCM
20 (0x0016)	ITU G.723 ADPCM (Yamaha)
49 (0x0031)	GSM 6.10
64 (0x0040)	ITU G.721 ADPCM
80 (0x0050)	MPEG

Tabla 18. Códigos comunes etiqueta de formato

- **Número de canales:** Especifica cuántas señales de audio separadas van codificadas en el bloque wav. Un valor de "1" indica una señal "mono", "2" indica "estéreo", etc.

- **Frecuencia de muestreo:** Indica la tasa de muestreo en muestras por segundo (Hercios). Para PCM los valores comunes son 8 kHz, 11.025 kHz, 22.05 kHz, y 44.1 kHz.

- **Bytes por segundo en media:** Indica cuantos bytes de datos deben transmitirse a un conversor D/A por segundo para poder reproducir el archivo. Esta información resulta de utilidad cuando deseamos comprobar si la información puede ser transmitida desde la fuente de una manera lo suficientemente rápida para permitir su reproducción. Se puede calcular de manera sencilla:

$$Bytes\ por\ segundo\ en\ media = Frecuencia\ de\ muestreo \times Alineamiento\ de\ bloque$$

- **Alineamiento de bloque:** Indica el número de bytes por muestra. Es por tanto la unidad atómica mínima de datos para el tipo de codificación indicada en la etiqueta de formato. Para el caso de PCM este valor se calcula como:

$$Alineamiento\ de\ bloque = \frac{Bits\ por\ muestra \times Número\ de\ canales}{8}$$

- **Bits por muestra:** Especifica el número de bits usados para definir cada muestra. Típicamente son 8, 16, 24 o 32. Si el número de bits no está alineado a nivel de byte (múltiplo de 8) se redondea al valor alineado superior más próximo y los bits sin uso son puestos a cero e ignorados. Algunos sistemas de compresión no pueden definir un valor constante para este campo por lo que puede ser cero.

- **Bytes de formato adicionales:** Especifica el tamaño en bytes de la información adicional de formato incluida. Si no se requiere de dicha información, debe indicarse con un valor nulo. Para el código de compresión 0 (desconocido) este campo no existe. Para el formato PCM, su valor es ignorado.

En nuestro caso, hemos utilizado la estructura WAVFORMATEX definida en Windows que incluye la información común para todos los ficheros WAV. Para formatos que requieren información adicional (como es nuestro caso), ésta estructura se incluye como primer miembro de otra estructura mayor que incluye dicha información adicional.

Bloque de Información "data":

Contiene la información/muestras de audio digital que pueden ser decodificadas utilizando el formato y método de compresión indicados en el bloque de formato. Si el código de compresión es 1 (PCM sin compresión), el bloque contendrá valores de muestras en bruto (raw).

Los ficheros wav generalmente contienen un único bloque de datos, pero pueden contener más si van incorporados en una lista de bloques "wavl"

Descripción	Offset	Tamaño	Valor
ID de bloque	0x00	4	"data"
Tamaño de bloque	0x04	4	Depende de la longitud de la muestra y compresión utilizada
Datos	0x08	-	Datos

Tabla 19. Formato bloques "data"

Un punto que puede generar confusión es que a la hora de representar las muestras utilizando un único byte (8 bits), se definan como valores sin signo. Para cualquier otro tamaño se definen como valores con signo. Por ejemplo, una muestra de 16 bits puede variar desde -32.768 a +32.768 con un valor central (silencio) en 0.

En el caso de ficheros de audio multicanal, sus muestras son almacenadas de manera entrelazada, es decir se almacenan las muestras del instante actual de cada uno de los canales antes de pasar a las muestras del siguiente instante de tiempo. Esto permite que los ficheros puedan ser reproducidos o emitidos antes de ser leídos completamente, algo especialmente interesante a la hora de reproducir grandes archivos almacenados en el disco que no caben en memoria o en la transmisión de estos ficheros por internet. En la siguiente figura mostramos las posibles muestras almacenadas en un fichero wav procedentes de una señal de audio estéreo.

Tiempo	Canal	Valor
0	1 (izquierdo)	0x0053
	2 (derecho)	0x0024
1	1 (izquierdo)	0x0057
	2 (derecho)	0x0029
2	1 (izquierdo)	0x0063
	2 (derecho)	0x003C

Tabla 20. Muestras procedentes de una señal de audio estéreo almacenadas en un fichero WAV

Como hemos comentado previamente, todos los bloques RIFF (incluidos los bloques de datos wav) deben estar alineados a nivel de palabra. Si la información está formada por un número impar de bytes, debemos añadir un byte de relleno de valor 0 al final de este bloque. El tamaño de bloque en la cabecera no debe tener en cuenta el byte de relleno introducido.

Bloque "fact":

El bloque "fact" es obligatorio para todos los formatos de compresión distintos a PCM (por ejemplo en nuestro caso). Incluye información dependiente del formato de compresión. En la práctica y por el momento, sólo hay definido un único campo. Se trata de un valor con tamaño 4 bytes que especifica el número de muestras contenido en el bloque de datos. Éste valor puede emplearse junto con el de muestras por segundo (presente en el bloque de formato) para calcular la longitud en segundos del fichero en reproducción.

Según se introduzcan nuevos formatos, el bloque "fact" se expandirá añadiendo campos tras el del número de muestras. Las aplicaciones pueden usar el tamaño de este bloque para determinar los campos presentes en él.

Descripción	Offset	Tamaño	Valor
ID de bloque	0x00	4	"fact"
Tamaño de bloque	0x04	4	Depende del formato
Info. dependiente de formato	0x08	-	-

Tabla 21. Formato bloques "fact"

Otros tipos de bloque:

Los ficheros generados por Atvs-2.0 están formados por los bloques antes descritos, sin embargo, un fichero wav puede contener otros tipos de bloques entre los que están:

- Bloque de "wavl"
- Bloque de "slnt"
- Bloque de indicación "cue"
- Bloque de lista de reproducción "plst"
- Bloque de lista de datos asociados "list"
- Bloque de etiqueta "labl"
- Bloque de texto etiquetado "ltxt"

- Bloque de nota “note”
- Bloque de muestra “smpl”
- Bloque de instrumento “inst”

Variaciones de formato:

Una de las desventajas de la gran popularidad del formato wav es que, dentro de los cientos de programas que lo utilizan puede que se utilice de manera incorrecta, bien por errores de programación o por contar con una documentación insuficiente acerca de este formato. En algunas ocasiones algunos de estos programas se hicieron muy populares, generando miles de archivos con un formato wav incorrecto, por lo que la industria se vio forzada a adaptar sus programas para ser capaces de leer archivos generados de manera “incorrecta”. Las excepciones introducidas dentro del formato original/estricto wav son:

- **Valor incorrecto del campo Alineamiento de bloque:** Puede ser solventado empleando la fórmula antes expuesta para su cálculo.

- **Valor incorrecto del campo Muestras por segundo en media:** Puede resolverse calculando dicho valor con la fórmula antes indicada.

- **Falta de byte de relleno para alineamiento:** Este problema puede ser difícil de resolver. Una opción es mostrar al usuario una advertencia cuando bloques que no se reconocen, se consiguen reconocer al aplicar un offset de 1 byte en la lectura. No se trata de una solución concreta pero puede funcionar incluso aunque el programa no disponga de una lista de identificadores legales.

4.1.2.5 Rediseño formato de archivo

Un fichero codificado por la anterior versión del codificador (ATVS-1.0) estaba formado por una sucesión de paquetes, desde el primero en el que se incluía la cabecera (paquete número 0) hasta el último (paquete N). Cada uno de esos paquetes eran elementos del tipo estructura de datos AtvsPacket.

Esta estructura de paquete se trataba de una alternativa a la paginación Ogg del códec Speex original y el control de los paquetes se realizaba a través de su numeración.

En el primer paquete, el puntero *packet apuntaba a una estructura de cabecera (AtvsHeader) mientras que para el resto de paquetes lo hacía a una estructura AtvsBits cuyo puntero *chars apuntaba a su vez a los bits codificados pertenecientes a la trama o tramas codificadas en ese paquete.

Debido a la inclusión de la cabecera wav y la necesidad de satisfacer los requisitos del Audio Compression Manager, el formato de archivo ha sido modificado.

- Para el codificador ACM, se ha incorporado una tabla de información de calidad *QualityInfo* que indica el número de bits necesarios para codificar una trama de sonido, el número de tramas por bloque y la tasa de codificación efectiva para minimizar la pérdida por el relleno de alineamiento. Los índices de esta tabla son la tasa de muestreo, el número de canales y el factor de calidad.

- Esta tabla ha sido llevada también al codificador de Linux para satisfacer la concordancia necesaria entre ambos sistemas forzando los valores de codificación a uno de los disponibles en la tabla y por tanto entendibles por el driver ACM.

Frecuencia de muestreo	Número de canales	Calidad	Bits por trama	Tamaño de trama	Tramas por bloque	Bitrate efectivo
8000	1	0	43	160	8	2150
8000	1	1	79	160	8	3950
8000	1	2	119	160	8	5950
8000	1	3	160	160	1	8000
8000	1	4	160	160	1	8000
8000	1	5	220	160	2	11000
8000	1	6	220	160	2	11000
8000	1	7	300	160	2	15000
8000	1	8	300	160	2	15000
8000	1	9	364	160	2	18200
8000	1	10	492	160	2	24600
8000	2	0	60	160	2	3000
8000	2	1	96	160	1	4800
8000	2	2	136	160	1	6800
8000	2	3	177	160	7	8857
8000	2	4	177	160	7	8857
8000	2	5	237	160	8	11850
8000	2	6	237	160	8	11850
8000	2	7	317	160	8	15850
8000	2	8	317	160	8	15850
8000	2	9	381	160	3	19066
8000	2	10	509	160	3	25466
16000	1	0	79	320	8	3950
16000	1	1	115	320	8	5750
16000	1	2	155	320	8	7750
16000	1	3	196	320	2	9800
16000	1	4	256	320	1	12800
16000	1	5	336	320	1	16800
16000	1	6	412	320	2	20600
16000	1	7	476	320	2	23800
16000	1	8	556	320	2	27800
16000	1	9	684	320	2	34200
16000	1	10	844	320	2	42200
16000	2	0	96	320	1	4800
16000	2	1	132	320	2	6600
16000	2	2	172	320	2	8600
16000	2	3	213	320	8	10650
16000	2	4	273	320	7	13657
16000	2	5	353	320	6	17666
16000	2	6	429	320	3	21466
16000	2	7	493	320	3	24666
16000	2	8	573	320	3	28666
16000	2	9	701	320	3	35066

16000	2	10	861	320	3	43066
32000	1	0	83	640	8	4150
32000	1	1	151	640	8	7550
32000	1	2	191	640	8	9550
32000	1	3	232	640	1	11600
32000	1	4	292	640	2	14600
32000	1	5	372	640	2	18600
32000	1	6	448	640	1	22400
32000	1	7	512	640	1	25600
32000	1	8	592	640	1	29600
32000	1	9	720	640	1	36000
32000	1	10	880	640	1	44000
32000	2	0	100	640	2	5000
32000	2	1	168	640	1	8400
32000	2	2	208	640	1	10400
32000	2	3	249	640	7	12457
32000	2	4	309	640	8	15450
32000	2	5	389	640	3	19466
32000	2	6	465	640	6	23266
32000	2	7	529	640	6	26466
32000	2	8	609	640	5	30480
32000	2	9	737	640	5	36880
32000	2	10	897	640	5	44880

Tabla 22. Información de calidad QualityInfo

El nuevo formato de archivo está formado por la cabecera WAV, la cabecera Atvs-2.0 y una serie de tramas con la información codificada. Para ello se hace uso de la estructura ATVSWAVEFORMAT compuesta por:

- **Estructura WAVEFORMATEX:** la estructura WAVEFORMATEX definida en el archivo MMsystem.h del Windows SDK incluye la información común para todos los ficheros WAV: Etiqueta de formato, número de canales, frecuencia de muestreo, Bytes por segundo en media, alineamiento de bloque, bits por muestra y bytes extra (diferencia entre el tamaño de la estructura ATVSWAVEFORMATEX y WAVEFORMATEX.

- Identificador de versión ACM formado por dos bytes: acm_version_major y acm_version_minor.

- Cabecera AtvsHeader: También ha sufrido modificaciones. Se han desordenado sus elementos y se han añadido campos nulos para satisfacer los requisitos de alineamiento.

En cuanto a las tramas de información codificada se ha eliminado la estructura de paquetes. Ahora en el bloque de datos "data" se encuentra la información de las tramas codificadas sin empaquetar.

Este cambio se debe a que, dado que los datos necesarios para la decodificación se encuentran en las cabeceras WAV y Atvs y que el control de las tramas se puede llevar a cabo con la información de alineamiento de bloque, esta estructura suponía un trabajo y espacio ocupados sin funcionalidad.

Se mantiene la desordenación/ordenación a nivel de byte, aunque ahora se lleva a cabo en las propias funciones de la librería Libatvs-2.0 atvs_bits_write, atvs_bits_read_from y atvs_bits_read_whole_bytes, por motivos de comodidad en su integración con el driver ACM.

4.1.2.6 Funciones códec-driver

Las funciones ACM están, en general, organizadas en varias categorías. Su nomenclatura nos permite identificarlas de manera sencilla. Los nombres de las funciones (con la excepción de dos de ellas) siguen el siguiente esquema:

$$acmGrupoFunción$$

donde:

- ***Grupo*:** define la categoría ACM: "Driver," "Format," "FormatTag," "Filter," "FilterTag," or "Stream".
- ***Función***: define la utilidad desarrollada por dicha función.

Las funciones en los grupos "Filter" y "Format" son muy similares y casi todas las funciones que actúan sobre filtros cuentan con una equivalente que actúa sobre los formatos.

Las funciones en el grupo de flujo "Stream" representan todos los pasos involucrados en una conversión (apertura de instancia de conversión, preparación para la conversión, conversión, limpieza una vez la conversión ha concluido y cierre de instancia de conversión).

En el caso de nuestro códec-driver AtvsACM hemos organizado las funciones en dos archivos: codec.c y atvs_acm.c. En el primero (codec.c) contamos con funciones ACM de tipo Driver, Format, FormatTag, y Stream:

Driver	Format	FormatTag	Stream	Atvs
acmDriverOpen acmDriverClose acmDriverConfigure acmDriverDetails acmDriverAbout	acmFormatSuggest acmdFormatDetails	acmFormatTagDetails	acmStreamOpen acmStreamClose acmdStreamSize acmdStreamConvert	atvsBlockAlign atvsSamplesPerBlock atvsAvgBytesPerSec atvsIsValidFormat

Tabla 23. Funciones ACM fichero "codec.c"

En el segundo (atvs_acm) se encuentran las rutinas propias de la conversión:

Descripción	Funcionalidad
pcmM08BytesToSamples pcmM16BytesToSamples pcmS08BytesToSamples pcmS16BytesToSamples	Encargadas de identificar el número de muestras presentes en un buffer
atvsEncode_M08 atvsEncode_M16 atvsEncode_S08 atvsEncode_S16	Rutinas de codificación. Convierten un buffer de información PCM a ATVS
atvsDecode_M08 atvsDecode_M16 atvsDecode_S08 atvsDecode_S16	Rutinas de decodificación. Convierten un buffer de información ATVS a PCM

Tabla 24. Funciones ACM fichero "atvs_acm.c"

En el anexo (apartado 8.1) se adjunta una descripción en detalle de estas funciones.

4.1.3 Readaptación Códec

4.1.3.1 Cambios Introducidos. Motivación de los cambios

Para cumplir la compatibilidad con el Microsoft Audio Compression Manager y para adaptarse al nuevo formato de archivo descrito en apartados anteriores de este documento se han llevado a cabo los siguientes cambios en el codificador:

- Como hemos comentado anteriormente la desordenación a nivel de byte del bitstream la realiza ahora la función atvs_bits_write por motivos de comodidad en su integración con el driver ACM.
- Por motivos de compatibilidad con el driver ACM y que éste sea capaz de decodificar los archivos generados por el codificador, portamos la tabla QualityInfo del driver ACM. Recordemos que ésta tabla indica el número de bits necesarios para codificar una trama de sonido, el número de tramas por bloque y la tasa de codificación efectiva para minimizar la pérdida por el relleno de alineamiento. Los índices son la tasa de muestreo, el número de canales y el factor de calidad.
- Se obligan las condiciones de codificación a una de las recogidas por la tabla QualityInfo y que son entendidas por el driverACM de Windows. En caso de que el bitstream indicado no se encuentre entre los disponibles, se utilizará el inferior más próximo. Por defecto se utilizará 12800 bps.
- La elección del número de tramas por bloque se hace por tanto de manera automática dejando de ser un parámetro modificable por el usuario.
- Se compone la cabecera WAV con los datos de codificación.
- Se ha eliminado la estructura de paquetes. Ahora en el bloque de datos “data” se encuentra la información de las tramas codificadas sin empaquetar. La lectura se controla ahora gracias al valor de alineamiento de bloque proporcionado en la cabecera.
- Al finalizar la codificación se deben actualizar los campos de la cabecera longitud total de fichero (desplazamiento 4), número de bytes de datos (desplazamiento 168) y número de tramas codificadas (desplazamiento 160). Sin embargo, para poder realizar una grabación continua y que frente a una interrupción o corte de la codificación se pueda reproducir el archivo codificado, estos campos se actualizarán cada vez que se produzca un almacenado de datos en el fichero.

El decodificador varía drásticamente su estructura. Ahora se encarga de ir leyendo y comprobando cada uno de los bloques wav que forman la cabecera. Está preparado para manejar el formato de archivo wave broadcast, descartando los bloques de información adicional innecesarios para la decodificación del fichero.

Una vez alcanzado el bloque de información codificada “data” se procede a la lectura de las tramas según el valor del alineamiento de bloque y a su decodificación.

Por último actualiza los campos de tamaño de fichero y de datos de la cabecera wav del fichero decodificado.

Desaparecen las funciones atvs_read_packet y order_bitstream. La primera desparece al eliminar la estructuración por paquetes. La ordenación sigue manteniéndose pero se realiza ahora en las funciones atvs_bits_read_from y atvs_bits_read_whole_bytes de la librería Libatvs-2.0

4.2 Pruebas y resultados

Las pruebas desarrolladas durante este apartado tienen como finalidad:

- Por un lado comprobar que las modificaciones llevadas a cabo, tanto para acelerar la ejecución en el sistema embebido, eliminar el uso de la unidad de punto flotante o el formato de archivo, no alteran la calidad ofrecida por el codificador de manera significativa
- Por otro lado comprobar que el codificador cumple los requisitos de tiempo establecidos por el cliente (tiempo de codificación del orden de 30% para permitir dos codificaciones simultáneas en tiempo real).

4.2.1 Prueba de calidad objetiva

Esta primera prueba trata de evaluar el efecto sobre la calidad ofrecida por la eliminación del uso de la unidad de punto flotante durante la codificación. Para ello repetiremos las pruebas de calidad objetiva llevadas a cabo en la evaluación de los códecs candidatos, pero esta vez comparando los resultados entre el codificador ATVS-2.0 y el Speex original.

CONJUNTO ORIGINAL

Para las codificaciones del conjunto de datos original podemos apreciar como prácticamente la puntuación obtenida por el códec ATVS en la unidad frente a la obtenida por Speex se igualan diferenciándose en media un 0.8 %.

Los resultados para el conjunto de datos reducido son los siguientes:

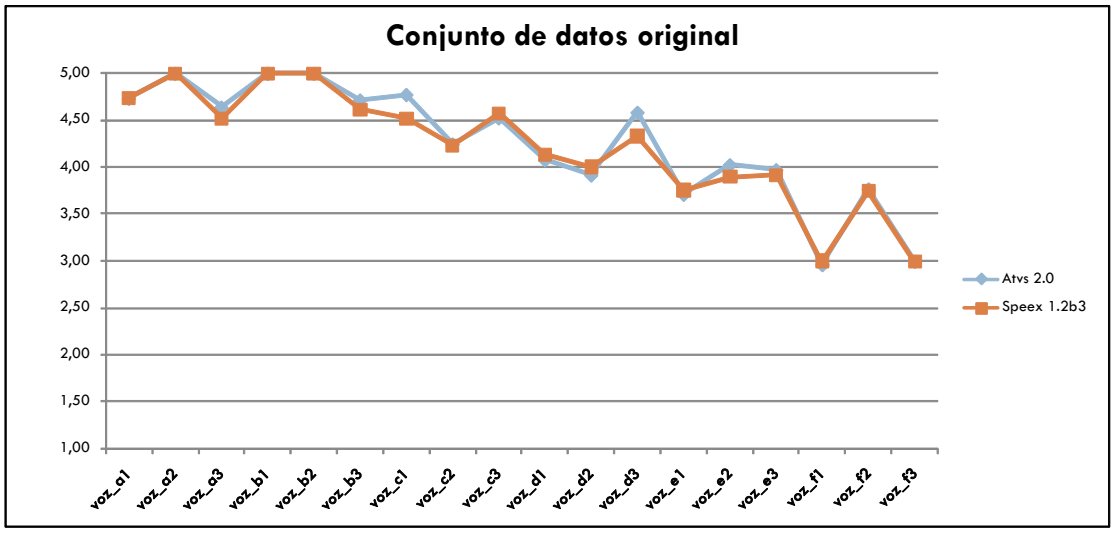


Figura 44. Puntuación p.563 conjunto reducido original

	SPEEX 1.2b3 12,8 Kbps	ATVS 2.0 12,8 Kbps
Media	4,22	4,26
Desv típica	0,61	0,63

Tabla 25. Puntuación p.563 conjunto reducido original

A continuación mostramos los resultados obtenidos para el conjunto de datos ampliado. Podemos comprobar que las modificaciones realizadas para acelerar la codificación así como la eliminación del uso de la unidad de punto flotante no han afectado a la calidad media ofrecida para el conjunto original:

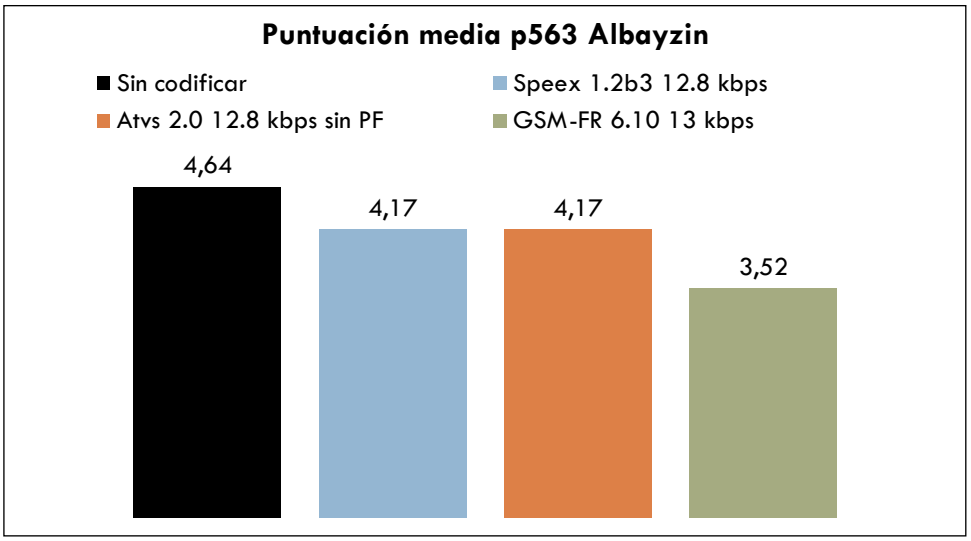


Figura 45. Puntuación p.563 conjunto ampliado original

	SIN CODIFICAR	SPEEX 1.2b3 12,8 Kbps	ATVS 2.0 12,8 Kbps	GSM-FR 6.10 13 Kbps
Media	4,64	4,17	4,17	3,56
Desv típica	0,40	0,47	0,47	0,39

Tabla 26. Puntuación p.563 conjunto ampliado original

Para el conjunto de datos ampliado original, se ha repetido la prueba codificando mediante GSM-FR 6.10 los ficheros de entrada a 16 kHz, obteniendo una puntuación de 3.51, lo que supone unos resultados prácticamente iguales (un 1% peor) a los obtenidos con el GSM-FR 6.10.

CONJUNTO CON RUIDO

Para la prueba de calidad de los códecs en escenarios ruidosos hemos empleado el mismo conjunto de datos que utilizamos en la primera evaluación. Recordemos que eran los ficheros del conjunto original, al que se les había añadido ruido blanco y gaussiano con una SNR de 30 dB, antes de aplicar las codificaciones y descodificaciones correspondientes.

En el caso del conjunto ampliado con ruido las puntuaciones obtenidas son:

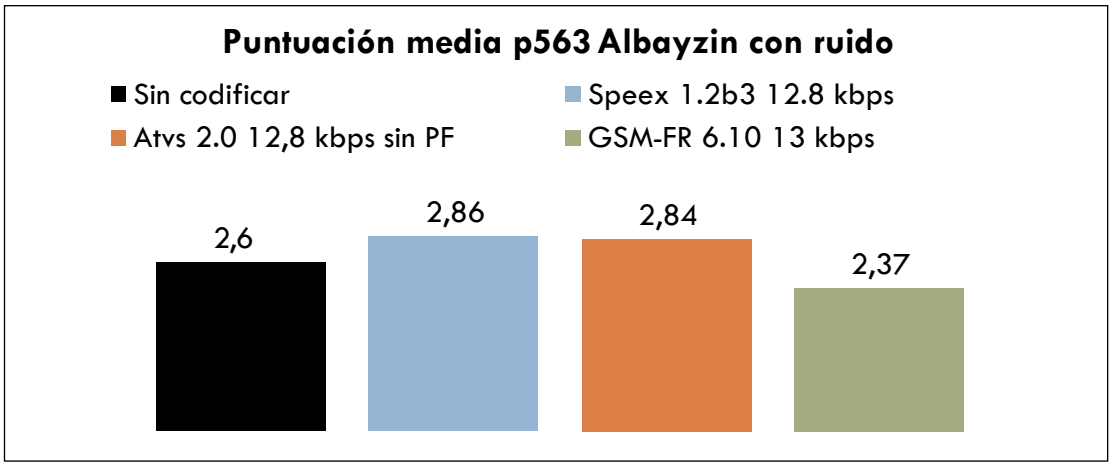


Figura 46. Puntuación p.563 conjunto ampliado con ruido blanco añadido

	SIN CODIFICAR	SPEEX 1.2b3 12,8 Kbps	ATVS 2.0 12,8 Kbps	GSM-FR 6.10 13 Kbps
Media	2,62	2,86	2,84	2,37
Desv típica	0,44	0,58	0,64	0,74

Tabla 27. Puntuación p.563 conjunto ampliado con ruido blanco añadido

Podemos observar cómo, para el conjunto con ruido blanco añadido, las modificaciones realizadas para adaptar el códec a la unidad apenas han afectado a la calidad obtenida, obteniendo una variación del 0.4 % en la escala de calidad ofrecida por la recomendación p.563.

CONJUNTO ATENUADO

Para la prueba de los codificadores en condiciones de bajo volumen recordemos que se atenuaron los ficheros de los conjuntos de datos reducido y ampliado 10, 20, 30 y 40 dB antes de aplicar las codificaciones/descodificaciones correspondientes. Los ficheros rebajados 40 dB no pueden ser evaluados mediante la Recomendación p.563 por no cumplir el margen dinámico especificado en sus requerimientos.

Del mismo modo que en la primera evaluación, presentamos los resultados obtenidos para el conjunto rebajado 30 dB (los obtenidos para 20 dB son similares) donde podemos apreciar que la calidad ofrecida por ATVS es prácticamente la misma proporcionada por SPEEX y bastante superior a la ofrecida por GSM.

Para el conjunto de datos ampliado los resultados obtenidos son los siguientes:

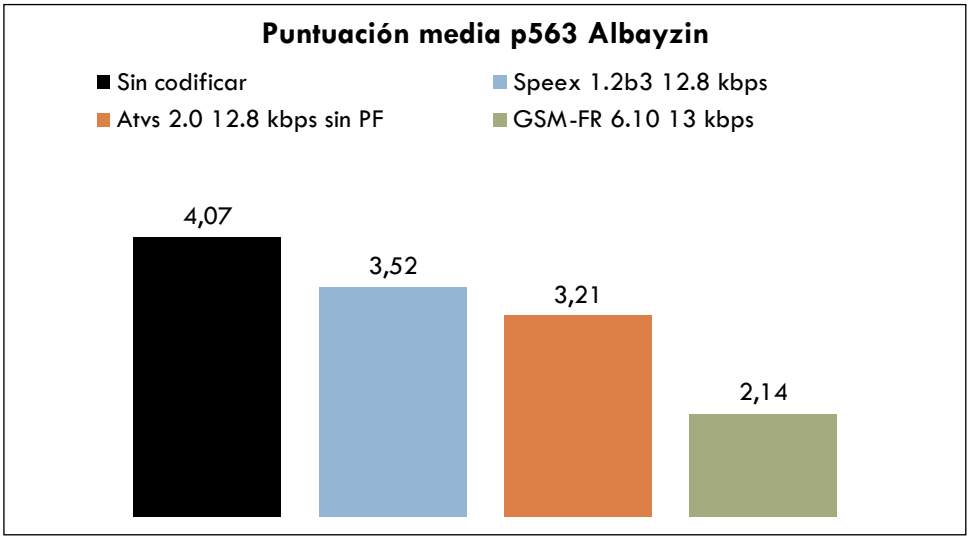


Figura 47. Puntuación p.563 conjunto ampliado atenuado 30 dB

	SIN CODIFICAR	SPEEX 1.2b3 12,8 Kbps	ATVS 2.0 12,8 Kbps	GSM-FR 6.10 13 Kbps
Media	4,07	3,52	3,21	2,14
Desv típica	0,6	0,55	0,45	0,47

Tabla 28. Puntuación p.563 conjunto ampliado atenuado 30 dB

Podemos observar cómo, para el conjunto atenuado, las modificaciones realizadas para adaptar el códec a la unidad han afectado en pequeña medida a la calidad obtenida. La variación es tan solo del 6.2 %. Aún así, la calidad proporcionada sigue siendo bastante superior a la ofrecida por el GSM-FR.

CALIDAD EN EFECTOS DE SONIDO

En cuanto a la codificación de *ruidos* o *efectos de sonido*, ha sido necesario escuchar los ficheros obtenidos para evaluar la calidad ofrecida ya que el algoritmo empleado en la Recomendación p.563 está diseñado exclusivamente para la evaluación de voz humana. No se perciben diferencias significativas entre el Speex y el códec ATVS.

4.2.2 Prueba requisitos de tiempo

Esta prueba tiene como finalidad comprobar que el codificador cumple los requisitos de tiempo establecidos por el cliente (tiempo de codificación del orden de 30% para permitir dos codificaciones simultáneas en tiempo real).

Para la evaluación del tiempo de codificación, tanto en el PC como en la unidad MIPSEL, se han utilizado tres archivos de voz de 153, 82 y 41 segundos.

Dichos archivos han sido traducidos a formato de audio wav, entendible por el codificador, utilizando frecuencias de muestreo de 8 kHz (banda estrecha), 16 kHz (banda ancha) y 32 kHz (ultra wide-band) con el objetivo de probar los distintos modos de codificación del códec.

A continuación se han codificado los ficheros mediante el códec, tanto en el PC como en la unidad MIPSEL, empleando la complejidad de compilación por defecto (3).
El tiempo de ejecución en el PC se ha medido utilizando la instrucción time.

En la unidad, dado que el núcleo de Linux instalado no dispone de la instrucción time o una equivalente, la medición de tiempos se ha llevado a cabo utilizando el comando date antes y después de la instrucción de codificación. Dicha instrucción proporciona la fecha y hora del sistema. Es por esto que las mediciones aquí presentadas tienen una precisión de un segundo.

Ejemplo:

```
date; speexenc -w -bitrate 12800 falonso_europa16.wav falonso_europa_16.cod; date
```

A continuación presentamos los resultados obtenidos en la prueba. En la primera columna se muestra el archivo utilizado y su duración. En las siguientes, la frecuencia de muestreo, tasa binaria empleada, el modo/calidad equivalente a esa tasa binaria y el tiempo de codificación. En la última columna se muestra el porcentaje de tiempo empleado en la codificación sobre la duración total del fichero.

Fichero	FRECUENCIA DE MUESTREO [HZ]	TASA BINARIA [BPS]	MODO / CALIDAD	TIEMPO CODIFICACIÓN PC [S]	% TIEMPO REAL PC	TIEMPO CODIFICACIÓN MIPSEL [S]	% TIEMPO REAL MIPSEL
falonso_canada Duración: 41 segundos	8k	**15000**	7-8	1,33	3%	9	22%
		11000	5-6	1,59	4%	11	27%
		8000	3-4	1,09	3%	8	20%
		5950	2	0,71	2%	5	12%
	16k	**12800**	4	1,86	5%	13	32%
		9800	3	1,38	3%	10	24%
		7750	2	0,99	2%	7	17%
		5750	1	1,13	3%	8	20%
	32k	**12800**	4	1,85	5%	14	34%
		9800	3	1,4	3%	10	24%
		7750	2	1,6	4%	10	24%
		5750	1	1,16	3%	8	20%

falonso_australia Duración: 82 segundos	8k	**15000**	7-8	1,20	1%	18	22%
		11000	5-6	1,55	2%	22	27%
		8000	3-4	1,12	1%	16	20%
		5950	2	0,72	1%	11	13%
	16k	**12800**	4	1,8	2%	26	32%
		9800	3	1,39	2%	20	24%
		7750	2	1	1%	14	17%
		5750	1	1,16	1%	16	20%
	32k	**12800**	4	1,76	2%	26	32%
		9800	3	1,40	2%	20	24%
		7750	2	1,58	2%	21	26%
		5750	1	1,16	1%	15	18%
falonso_europa Duración: 153 segundos	8k	**15000**	7-8	4,25	3%	33	22%
		11000	5-6	5,64	4%	41	27%
		8000	3-4	4,03	3%	31	20%
		5950	2	2,54	2%	20	13%
	16k	**12800**	4	6,64	4%	47	31%
		9800	3	4,98	3%	37	24%
		7750	2	3,46	2%	26	17%
		5750	1	4,16	3%	29	19%
	32k	**12800**	4	6,61	4%	47	31%
		9800	3	5,38	4%	36	24%
		7750	2	5,73	4%	39	25%
		5750	1	4,17	3%	31	20%

Tabla 29. Resultados prueba de tiempos desarrollo códec

En la prueba realizada se han obtenido tiempos de codificación entre el 1 y el 5% de tiempo real en el PC y entre el 13 % y el 34 % de tiempo real en la unidad MIPSEL, siendo la media un 2,67 % en PC y un 23% en MIPSEL.

Si atendemos únicamente al modo de operación (para las diferentes tasas binarias evaluadas) los resultados son en media (para la unidad MIPSEL) del 20% para banda estrecha (8kHz), 23% para banda ancha (16kHz) y 25 % para banda ultra ancha (32 kHz).

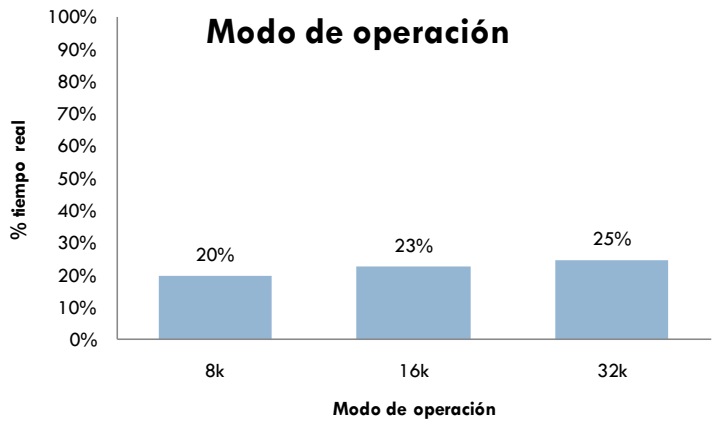


Figura 48. Porcentaje de tiempo real en media para cada modo de operación

Cumple por tanto los requisitos de tiempo establecidos por el cliente ya que los tiempos de ejecución máximos son del orden del 30% lo que permite mantener dos codificaciones simultáneas en tiempo real en el sistema embebido.

5 CONCLUSIONES

El objetivo básico del proyecto consistía en desarrollar un códec de voz propietario para el cliente basándonos como punto de partida en códecs ya desarrollados y centrándonos en mejorar la calidad del GSM-FR 6.10, especialmente en condiciones de ruido, bajo volumen y no sólo voz sino también ruidos, tratando de que el códec influyese poco en el procesamiento posterior (reconocimiento de locutor, idioma, voz, reducción de ruido, etc.)

Para ello, comparamos el códec de partida (GSM-FR 6.10) con otros teóricamente mejores en cuanto a capacidad de codificar: el ETSI AMR-WB y el Speex 1.2b3. De estas pruebas extrajimos que:

- Speex y AMR-WB conseguían mejor calidad y eran más versátiles que GSM-FR 6.10
- En medidas de calidad subjetiva el vencedor era Speex
- En procesamiento posterior Speex y AMR-WB resultaban similares
- En coste computacional el mejor fue Speex

Finalmente elegimos Speex como códec base.

Modificamos el códec seleccionado para convertirlo en un códec propietario

- Eliminando el recubrimiento Ogg.
- Alterando las cabeceras.
- Alterando el orden de los bits.

Portamos el nuevo códec (llamado ATVS-1.0) a Linux y al sistema embebido (MIPSEL 400MHz). La eliminación del uso de coma flotante no afectó significativamente a la calidad ofrecida por el codificador. Además se cumplía para varios modos de codificación los requisitos de velocidad establecidos (tiempo inferior a RT (en torno a 30% CPU)). Para modos de codificación más exigentes (especialmente los de ancho de banda muy extendido) los requisitos de velocidad no se llegaban a cumplir.

Se desarrolló un driver de un códec ACM para windows que se instala en el sistema y permite emplear la codificación y descodificación a todas las aplicaciones del sistema operativo.

Por último se rediseñó el formato del códec así como las versiones para Linux y sistema embebido MIPSEL ya que el formato de la primera versión no era compatible ni se podía encapsular en el formato WAV necesario para desarrollar un códec ACM. Además tuvimos que eliminar algunas opciones de codificación pues no era posible mantenerlas en ACM como la codificación con bitrate variable. Se generó así la versión ATVS-2.0 del códec.

Durante el desarrollo del proyecto y de cara a satisfacer las distintas entregas planificadas con el cliente se generaron diversos entregables. En la sección 8.2 del anexo se incluye un listado de dichos entregables y se adjunta a modo de ejemplo uno de ellos en la sección 8.3 (Manual sobre estructuración del código fuente, instalación y funcionamiento del codificador y driver ACM).

6 REFERENCIAS

[1] Wai C. Chu, **"Speech coding algorithms: foundation and evolution of standardized coders**", J. Wiley, 2003.

[2] Bosi, Marina, **"Introduction to digital audio coding and standards**", Kluwer Academic, 2003.

[3] J.Watkinson, **"The Art of Digital Audio",** Ed. Focal Press, 3ª edición.

[4] L.R. Rabienr, R.W. Schafer, **"Digital Processing of Speech Signals"**, Prentice Hall

[5] D. Torre Toledano, "**Fundamentos de la percepción y generación de audio y voz"**, Material docente de Tratamiento Digital de Audio. 2008-2009.

[6] D. Torre Toledano, **"Compresión, Codificación y Descripción de Señales de Audio y Voz"**, Material docente de Tratamiento Digital de Audio. 2008-2009.

[7] J. Ortega García, **"La Señal de voz"**, Material docente de Ampliación de Señales Aleatorias. 2007-2008.

[8] J. Ortega García, **"Codificación de Señales de Voz"**, Material docente de Ampliación de Señales Aleatorias. 2007-2008.

[9] Jean-Marc Valin, **"The Speex Codec Manual Version 1.2 Beta 3"**, Xiph.org Foundation, Diciembre 2007.

[10] Jean-Marc Valin, **"Speex API Reference Manual 1.2-beta 2"**, Xiph.org Foundation, Mayo 2007.

[11] Jean-Marc Valin, **"Speex: A Free Codec For Free Speech",** CSIRO ICT Centre Australia, Xiph.org Foundation.

[12] ETSI Technical Specification **TS 100 961 v8.2.0 "GSM-FR"**, Junio 2001.

[13] ETSI Technical Specification **TS 126 171 v7.0.0 "AMR-WB"**, Enero 2007.

[14] Recomendación UIT-T P.563 **"Método basado en un solo extremo para la evaluación objetiva de la calidad vocal en aplicaciones de banda estrecha",** Mayo 2004.

[15] Microsoft Corporation, **"Microsoft Developer Network (MSDN) Library**", Microsoft, 2009.

[16] P.Laben, **"Specification of the Broadcast Wave Format"**, European Broadcasting Union, Julio 1997.

[17] R.Chalmers, "**The Broadcast Wave Format",** EBU Technical Review, Invierno 1997.

[18] Xiph.org, **"The Ogg Skeleton Metadata",** http://www.xiph.org/ogg, Xiph.org Foundation, 2008.

[19] Scott Wilson, **"WAVE PCM soundfile format",** http://ccrma.stanford.edu/courses.

[20] The Sonic Spot, **"WAVE File Format",** http://www.sonicspot.com/guide/wavefiles.htm

[21] Xiph Open Source Community, **"Direct Show Filters",** http://www.xiph.org/dshow

7 GLOSARIO

ACELP Algebraic code excited linear prediction (Predicción lineal algebraica excitada por código).

ACM Audio Compression Manager.

ADPCM Adaptative Differential Pulse Code Modulation (Modulación diferencial adaptativa por impulsos codificados).

AGC Automatic Gain Control (Control automático de ganancia).

AMR Adaptative Multi-rate Compression (Codificación adaptativa multitasa).

API Application Programming Interface (Interfaz de programación de aplicaciones).

CELP Code Excited Linear Prediction (Codificador de predicción lineal excitada por código).

CODEC Codificador de audio digital (por encoder-decoder).

DCF Detection Cost Function (Función de detección de coste).

DDK Driver Development Kit (Kit de desarrollo de controladores).

DET Curvas DET. Representan gráficamente cómo evoluciona el porcentaje de falsa aceptación frente al de falso rechazo al modificar el umbral de reconocimiento.

DPCM Differential Pulse Code Modulation (Modulación diferencial por impulsos codificados).

DTX Discontinuous Transmission (Transmisión discontinua).

EER Equal Error Rate (Tasa de equierror).

ETSI European Telecommunications Standards Institute (Instituto Europeo de estándares de Telecomunicaciones)

GMM Gaussian Mixture Model (Modelo de mezcla de Gaussianas).

HMMs Hidden Markov Models (Modelos ocultos de Markov).

IFF Interchange File Format (Formato intercambiable de fichero)

LPC Linear Predictive Coding (Codificación por predicción lineal).

LQO Listening Quality Objective (Calidad objetiva de escucha).

LQS Listening Quality Subjective (Calidad subjetiva de escucha).

LTP Long Term Prediction (Predicción a largo plazo).

MDF Magnitude Difference Function (Función del valor absoluto de la diferencia).

MFCC Mel-Frequency Cepstral Coefficients (Coeficientes Cepstrales en escala de frecuencias MEL).

MIDI Musical Instrument Digital Interface (Interfaz digital de gestión musical).

MPEG Moving Picture Experts Group (Grupo de expertos sobre imágenes en movimiento).

MSF Magnitude Sum Function (Función suma de los valores absolutos).

NAP Nuissance Attribute Projection (Proyección de atributos molestos).

NIST National Institute of Standard Technology.

PCM Pulse Code Modulation (Modulación por impulsos codificados).

PG Prediction Gain (Ganancia de predicción).

PPRLM Parallell Phone Recognition followed by Language Modelling.

RIFF Resource Interchange File Format.

RPE Regular Pulse Excitation Códecs.

SDK Software Development Kit (Kit de desarrollo de software).

SPE Speaker Recognition Evaluation (Evaluación competitiva internacional de reconocimiento de locutor).

SPL Sound Pressure Level (Nivel de presión sonoro).

UBM Universal Background Model.

VAD Voice Activity Detection (Detección de actividad vocal).

VBR Variable bit-rate (Tasa de información binaria variable).

VoIP Voice over IP (Voz sobre IP).

VSELP Vector Sum Excited Linear Prediction (Predicción lineal excitada por suma de vectores)

ZCR Zero Crossing Rate (Tasa de cruces por cero).

8 ANEXOS

8.1 Descripción funciones Códec-Driver ACM

acmDriverOpen

Función encargada de manejar los mensajes DRV_OPEN del driver ACM. El driver puede "abrirse" por varios motivos, siendo el más común la apertura como preparación a los trabajos de conversión. Es muy importante que el driver sea capaz de manejar correctamente múltiples instancias de apertura.

Argumentos

HDRVR hdrvr: Operador de driver que será devuelto al invocador de la función de apertura de driver. Normalmente será el ACM aunque podría ser el sistema.

LPACMDRVOPENDESC paod: Descripción abierta de cómo está siendo abierto el driver ACM. Este argumento puede ser nulo.

Devuelve

(LRESULT): 0 si el driver no puede abrirse. Otro valor en caso contrario.

acmDriverClose

Función encargada de manejar los mensajes DRV_CLOSE del driver ACM. El driver recibe un mensaje DRV_CLOSE por cada mensaje DRV_OPEN que haya resultado en una apertura válida.

Argumentos

PDRIVERINSTANCE pdi: Puntero a estructura de instancia de driver. Esta estructura es reservada opcionalmente durante el mensaje de apertura DRV_OPEN que maneja la función acmdDriverOpen.

Devuelve

(LRESULT): 0 si no puede cerrarse la instancia abierta. Otro valor en caso contrario.

acmDriverConfigure

Función encargada de manejar los mensajes de configuración DRV_[QUERY]CONFIGURE. Normalmente son para hardware, como por ejemplo la indicación de que debe mostrarse un diálogo para configurar puertos, mapas de memoria etc., aunque un driver software ACM también puede necesitar configuración para definir por ejemplo qué es tiempo real u otros aspectos de calidad frente a tiempo.

Argumentos

PDRIVERINSTANCE pdi: Puntero a estructura de instancia de driver. Esta estructura es reservada opcionalmente durante el mensaje de apertura DRV_OPEN que maneja la función acmdDriverOpen.

HWND hwnd: Indicador de ventana "padre" a utilizar cuando se muestra un cuadro de diálogo. Un driver ACM debe mostrar los cuadros de dialogo utilizando éste parámetro como "padre". Si el driver está siendo llamado para labores de configuración este parámetro tendrá un valor de (HWND)-1.

LPDRIVERCONFIGINFO pdci: Puntero a una estructura DRVCONFIGINFO opcional. Si este parámetro es nulo, el driver debe crear su propia localización de almacenamiento.

Devuelve

(LRESULT): Si el driver se llama para labores de configuración (hwnd=(HWND)-1) se devuelve un valor no nulo para indicar que el driver soporta un diálogo de configuración o cero si no lo soporta.
Si el driver se llama para mostrar el diálogo de configuración (hwnd!=(HWND)-1) se devolverá uno de los siguientes valores:

- **DRVCNF_CANCEL (0x0000):** Indica que se mostró el diálogo de configuración y el usuario o bien lo canceló o no modificó ninguno de los parámetros.
- **DRVCNF_OK (0x0001):** Indica que se mostró el diálogo de configuración y el usuario pulsó la opción ACEPTAR. Éste valor se devuelve aunque el usuario no hubiera modificado nada. De otro modo el driver no se instalará de manera adecuada.
- **DRVCNF_RESTART (0x0002):** Indica que se mostró el diálogo de configuración y el usuario introdujo cambios en la configuración que requieren reiniciar Windows antes de que los cambios tengan efecto. El driver mantendrá los valores anteriores de configuración hasta que se produzca el reinicio.

acmDriverDetails

Función encargada de manejar los mensajes ACMDM_DRIVER_DETAILS y de completar la información presente en la estructura ACMDRIVERDETAILS.

Argumentos

PDRIVERINSTANCE pdi: Puntero a estructura de instancia de driver. Esta estructura es reservada opcionalmente durante el mensaje de apertura DRV_OPEN que maneja la función acmdDriverOpen.

LPACMDRIVERDETAILS padd: Puntero a la estructura a rellenar por la función. Esta estructura puede tener un tamaño superior o inferior al definido. El campo cbStruct de la estructura indica el tamaño correcto.

Devuelve

(LRESULT): Cero (MMSYSERR_NOERROR) en el caso de que no haya ningún problema. Cualquier otro valor si no se pudieron obtener los detalles del driver. Ésta función no debería fallar en nunca. Los dos posibles puntos de error pueden ocurrir si:

- El puntero a la estructura es nulo.
- Si el campo cbStruct es menor que cuatro. En este caso no hay espacio para devolver el número de bytes escritos.

Dado que estos errores se encuentran identificados, ACM es capaz de detectarlos y el driver NO necesita comprobar estas condiciones.
Si el driver se llama para labores de configuración (hwnd=(HWND)-1) se devuelve un valor no nulo para indicar que el driver soporta un diálogo de configuración o cero si no lo soporta.
Si el driver se llama para mostrar el diálogo de configuración (hwnd!=(HWND)-1) se devolverá uno de los siguientes valores:

- **DRVCNF_CANCEL (0x0000):** Indica que se mostró el diálogo de configuración y el usuario o bien lo canceló o no modificó ninguno de los parámetros.

- **DRVCNF_OK (0x0001):** Indica que se mostró el diálogo de configuración y el usuario pulsó la opción ACEPTAR. Éste valor se devuelve aunque el usuario no hubiera modificado nada. De otro modo el driver no se instalará de manera adecuada.
- **DRVCNF_RESTART (0x0002):** Indica que se mostró el diálogo de configuración y el usuario introdujo cambios en la configuración que requieren reiniciar Windows antes de que los cambios tengan efecto. El driver mantendrá los valores anteriores de configuración hasta que se produzca el reinicio.

acmDriverAbout

Función encargada de manejar los mensajes ACMDM_DRIVER_ABOUT. Un driver ACM puede mostrar su propia ventana de "Información acerca de…" o dejar que ACM o la aplicación interesada muestre una por sí misma. La última opción es la recomendada, y así se ha seguido en este driver (AtvsACM) al estar incluida toda la información necesaria en la estructura ACMDRIVERDETAILS.

Argumentos

PDRIVERINSTANCE pdi: Puntero a estructura de instancia de driver. Esta estructura es reservada opcionalmente durante el mensaje de apertura DRV_OPEN que maneja la función acmdDriverOpen.

HWND hwnd: Indicador de ventana "padre" a utilizar cuando se muestra un cuadro de diálogo. Un driver ACM debe mostrar los cuadros de dialogo utilizando éste parámetro como "padre". Si el driver está siendo llamado para labores de configuración este parámetro tendrá un valor de (HWND)-1.

Devuelve

(LRESULT): MMSYSERR_NOT_SUPPORTED para indicar que el driver no mostrará una ventana propia. En este caso, ACM o la aplicación interesada postrará un cuadro genérico con la información contenida en la estructura ACM_DIRVER_DETAILS devuelta por el mensaje ACMDM_DRIVER_DETAILS.

acmFormatSuggest

Función encargada de manejar los mensajes ACMDM_FORMAT_SUGGEST. Su propósito es proporcionar de manera rápida al ACM o a la aplicación interesada un formato de destino al que podamos convertir la fuente. El ACM o la aplicación interesada pueden indicar restricciones en el formato que puede ser sugerido. El miembro fdwSuggest de la estructura padfs contendrá los bits de restricción. El driver soporta los siguientes:

- **ACM_FORMATSUGGESTF_WFORMATTAG:** La etiqueta de formato de destino debe ser igual al miembro "Etiqueta de formato" (wFormatTag) presente en la cabecera del formato de destino.
- **ACM_FORMATSUGGESTF_NCHANNELS:** El número de canales de destino debe ser el mismo que el especificado en el miembro de la cabecera (nChannels) del formato de destino.
- **ACM_FORMATSUGGESTF_NSAMPLESPERSEC:** El número de muestras por segundo debe ser el mismo que el especificado en el miembro de la cabecera (nSamplesPerSec) del formato de destino.

- **ACM_FORMATSUGGESTF_WBITSPERSAMPLE:** El número de bits por muestra debe ser el mismo que el especificado en el miembro de la cabecera (wBitsPerSample) del formato de destino.

Argumentos

PDRIVERINSTANCE pdi: Puntero a estructura de instancia de driver. Esta estructura es reservada opcionalmente durante el mensaje de apertura DRV_OPEN que maneja la función acmdDriverOpen.

LPACMDRVFORMATSUGGEST padfs: Puntero a la estructura AACMDRVFORMATSUGGEST que describe la fuente y el destino (posiblemente con restricciones) para la conversión.

Devuelve

(LRESULT): Cero (MMSYSERR_NOERROR) si la función termina sin errores. En caso contrario devuelve cualquier otro valor. Si no se puede sugerir ningún formato devolverá ACMERR_NOTPOSSIBLE.

acmFormatTagDetails

Función encargada de manejar los mensajes ACMDM_FORMATTAG_DETAILS. Su propósito es obtener detalles acerca de una etiqueta de formato soportada por el driver. El tipo de solicitud se indica en el atributo fdwDetails. Los tipos de solicitud soportados por AtvsACM para esta función son:

- **ACM_FORMATTAGDETAILSF_INDEX:** Indica que se ha proporcionado un índice de etiqueta de formato en el miembro dwFormatTagIndex de la estructura ACMFORMATTAGDETAILS. La función se encargará de completar la etiqueta y los detalles de formato en la estructura apuntada por el parámetro padft.
- **ACM_FORMATTAGDETAILSF_FORMAT:** Indica que se ha proporcionado una etiqueta de formato en el miembro dwFormatTagIndex de la estructura ACMFORMATTAGDETAILS. La función se encargará de completar los detalles de formato en la estructura apuntada por el parámetro padft.
- **ACM_FORMATTAGDETAILSF_LARGESTSIZE:** Indica que debe devolverse el mayor tamaño en bytes de la etiqueta de formato proporcionada.

Argumentos

PDRIVERINSTANCE pdi: Puntero a estructura de instancia de driver. Esta estructura es reservada opcionalmente durante el mensaje de apertura DRV_OPEN que maneja la función acmdDriverOpen.

LPACMFORMATTAGDETAILS padft: Puntero a la estructura ACMFORMATTAGDETAILS que describe etiqueta de la que deseamos obtener la información.

DWORD fdwDetails: Flags que definen la etiqueta de la que deseamos la información.

Devuelve

(LRESULT): Cero (MMSYSERR_NOERROR) si la función termina sin errores. En caso contrario devuelve cualquier otro valor. Si no se puede proporcionar información adicional para ese formato o el tipo de solicitud indicado en el atributo fdwDetails no puede atenderse devolverá ACMERR_NOTPOSSIBLE.

acmdFormatDetails

Función encargada de manejar los mensajes ACMDM_FORMAT_DETAILS. Su propósito es obtener detalles acerca de un formato en concreto soportado por el driver. El tipo de solicitud se indica en el atributo fdwDetails. Los tipos de solicitud soportados por AtvsACM para esta función son:

- **ACM_FORMATDETAILSF_INDEX:** Indica que se ha proporcionado un índice de etiqueta de formato en el miembro dwFormatIndex de la estructura ACMFORMATTAGDETAILS. La función se encargará de completar los detalles de formato en la estructura apuntada por el parámetro padf.
- **ACM_FORMATDETAILSF_FORMAT:** Indica que el miembro pwfx de la estructura ACMFORMATDETAILS apunta a una estructura WAVEFORMATEX de la que debe devolver los valores restantes. El miembro dwFormatTag de la estructura ACMFORMATDETAILS se inicializa con la misma etiqueta de formato indicada por pwfx. Esta llamada se suele realizar para obtener una cadena con la descripción de un formato arbitrario.

Esta función es la encargada de mostrar las opciones disponibles de formato en una cadena de texto del tipo mostrado en la selección de formato en la grabadora de sonido de Windows XP.

Argumentos

PDRIVERINSTANCE pdi: Puntero a estructura de instancia de driver. Esta estructura es reservada opcionalmente durante el mensaje de apertura DRV_OPEN que maneja la función acmdDriverOpen.

LPACMFORMATDETAILS padf: Puntero a la estructura ACMFORMATDETAILS que describe el formato (para determinada etiqueta de formato) del que deseamos obtener la información.

DWORD fdwDetails: Flags que definen la etiqueta de la que deseamos la información.

Devuelve

(LRESULT): Cero (MMSYSERR_NOERROR) si la función termina sin errores. En caso contrario devuelve cualquier otro valor. Si no se puede proporcionar información adicional para ese formato o el tipo de solicitud indicado en el atributo fdwDetails no puede atenderse devolverá ACMERR_NOTPOSSIBLE.

acmStreamOpen

Función encargada de manejar los mensajes ACMDM_STREAM_OPEN lanzados para iniciar un nuevo flujo de conversión. Si la función se ejecuta de manera correcta se generarán uno o más mensajes ACMDM_STREAM_CONVERT (llamadas a la función acmStreamConvert). Si el bit de flag ACM_STREAMOPENF_NONREALTIME del parámetro fdwDetails no está activo, la conversión debe hacerse en tiempo real.

Antes de realizar ninguna operación se verifica que el códec puede realizar la conversión del flujo que se solicita. Para ello:

- Verifica que los formatos de fuente y destino son aceptables. Para ello hace uso de las funciones pcmIsValidFormat y atvsIsValidFormat descritas en la siguiente subsección.
- Determina en función del número de bits por muestra y el número de canales la rutina de conversión adecuada.
- Verifica que el número de canales *nChannels* y el número de muestras por segundo *nSamplesPerSec* son iguales en los formatos fuente y destino.

Una vez comprobado que puede llevar a cabo la conversión, reserva una pequeña estructura con el puntero a la función de conversión a utilizar en cada una de las conversiones de este flujo y una copia de la configuración actual del driver que no variarán durante la vida del flujo de conversión.

Por último, completa la información de la instanciación en la estructura ACMDRVSTREAMINSTANCE que se pasará a todos los mensajes que componen el flujo.

Argumentos

PDRIVERINSTANCE pdi: Puntero a estructura de instancia de driver. Esta estructura es reservada opcionalmente durante el mensaje de apertura DRV_OPEN que maneja la función acmdDriverOpen.

LPACMDRVSTREAMINSTANCE padsi: Puntero de instanciación del flujo de información. Esta estructura la reserva y rellena ACM con la información básica necesaria para la conversión. No variará durante el tiempo de vida del flujo y se pasará a los siguientes mensajes de flujo si la apertura se realiza de manera correcta.

DWORD fdwDetails: Flags que definen la etiqueta de la que deseamos la información.

Devuelve

(LRESULT): Cero (MMSYSERR_NOERROR) si la función termina sin errores. En caso contrario devuelve cualquier otro valor.

Si no se puede producir la conversión debido a incompatibilidades entre el formato fuente y destino devolverá ACMERR_NOTPOSSIBLE.

Si la conversión no puede realizarse en tiempo real y la petición de apertura no activa el flag ACM_STREAMOPENF_NONREALTIME devolverá MMSYSERR_NOTSUPPORTED.

acmStreamClose

Función encargada de manejar los mensajes ACMDM_STREAM_CLOSE lanzados cuando un flujo de conversión termina. El driver libera los recursos utilizados por el flujo durante la conversión.

Argumentos

LPACMDRVSTREAMINSTANCE padsi: Puntero de instanciación del flujo de información. Esta estructura la reserva y rellena ACM con la información básica necesaria para la conversión. Contiene la misma información que tenía durante la apertura del flujo (ACMDM_STREAM_OPEN).

Devuelve

(LRESULT): Cero (MMSYSERR_NOERROR) si la función termina sin errores. En caso contrario devuelve cualquier otro valor.

acmdStreamSize

Función encargada de manejar los mensajes ACMDM_STREAM_SIZE. Su propósito es doble:

- Calcular el tamaño máximo en bytes que el buffer de destino necesita dados el formato de entrada y salida y el tamaño en bytes del buffer de fuente. Es decir calcula cómo de grande debe ser el buffer destino para ser capaz de almacenar la información codificada. (ACM_STREAMSIZEF_SOURCE)
- Calcular el tamaño máximo en bytes del buffer de fuente dado el tamaño del buffer destino. (ACM_STREAMSIZEF_DESTINATION)

Argumentos

LPACMDRVSTREAMINSTANCE padsi: Puntero de instanciación del flujo de información. Esta estructura la reserva y rellena ACM con la información básica necesaria para la conversión. Contiene la misma información que tenía durante la apertura del flujo (ACMDM_STREAM_OPEN).

LPACMDRVSTREAMINSTANCE padss: Puntero a la estructura ACMDRVSTREAMSIZE que define los valores de tamaños proporcionados para el flujo de conversión.

Devuelve

(LRESULT): Cero (MMSYSERR_NOERROR) si la función termina sin errores. En caso contrario devuelve cualquier otro valor. Devuelve MMSYSERR_NOTSUPPORTED si no entiende la petición. Si la conversión produjera valores fuera de rango devolverá ACMERR_NOTPOSSIBLE.

acmdStreamConvert

Función encargada de manejar los mensajes ACMDM_STREAM_CONVERT que se produzcan tras la apertura correcta de un flujo de conversión (ACMDM_STREAM_OPEN). Es la encargada de la conversión de la información del flujo.

Argumentos

PDRIVERINSTANCE pdi: Puntero a estructura de instancia de driver. Esta estructura es reservada opcionalmente durante el mensaje de apertura DRV_OPEN que maneja la función acmdDriverOpen.

LPACMDRVSTREAMINSTANCE padsi: Puntero de instanciación del flujo de información. Esta estructura la reserva y rellena ACM con la información básica necesaria para la conversión. Contiene la misma información que tenía durante la apertura del flujo (ACMDM_STREAM_OPEN).

LPACMDRVSTREAMHEADER padsh: Puntero a la estructura de cabecera del flujo que define la fuente de información y el buffer destino de la conversión.

Devuelve

(LRESULT): Cero (MMSYSERR_NOERROR) si la función termina sin errores. Un código de error distinto de cero en caso contrario.

Además de las funciones básicas de ACM, para el funcionamiento del códec driver AtvsACM de han desarrollado las siguientes:

atvsBlockAlign

Función que calcula el alineamiento de bloque que debe ser utilizado dada la estructura WAVEFORMATEX.

Para su cálculo, utiliza los campos nChannels, nBitsPerSample y nSamplesPerSec presentes en la estructura de cabecera por lo que se asume que el formato es un formato ATVSFORMATEX válido y que sus valores en la estructura son correctos.

Además obtiene los datos nBitsPerFrame y nFramesPerBlock de la tabla de información de calidad.

Con ellos se calcula el valor de alineamiento de bloque empleando la fórmula descrita en la sección que describe el formato de archivo WAVE:

$$uBlockAlign = \frac{nBitsPerFrame \times nFramesPerBlock + 7}{8}$$

Argumentos

LPWAVEFORMATEX pwfx: Puntero al formato de cabecera.

Devuelve

(UINT): Valor de alineamiento de bloque.

atvsSamplesPerBlock

Función que calcula el número de muestras por bloque que debe ser utilizado dada la estructura WAVEFORMATEX.
Para su cálculo, utiliza los campos nChannels, nBitsPerSample y nSamplesPerSec presentes en la estructura de cabecera por lo que se asume que el formato es un formato ATVSFORMATEX válido y que sus valores en la estructura son correctos.
Además obtiene los datos nFrameSize y nFramesPerBlock de la tabla de información de Calidad.
Con ellos se calcula el valor del número de muestras que irán codificadas en cada bloque ATVS:

$$uSamplesPerBlock = nFrameSize \times nFramesPerBlock$$

Argumentos

LPWAVEFORMATEX pwfx: Puntero al formato de cabecera.

Devuelve

(DWORD): Número de muestras por bloque.

atvsAvgBytesPerSec

Función que calcula el número de bytes por segundo en media que debe ser utilizado dada la estructura WAVEFORMATEX.
Para su cálculo, utiliza los campos nChannels, nBitsPerSample y nSamplesPerSec presentes en la estructura de cabecera por lo que se asume que el formato es un formato ATVSFORMATEX válido y que sus valores en la estructura son correctos.
Además obtiene el dato nEffectiveBitrate de la tabla de información de Calidad.
Con ellos se calcula el valor medio del número de bytes por segundo:

$$dwAvgBytesPerSec = \frac{nEffectiveBitrate + 7}{8}$$

Argumentos

LPWAVEFORMATEX pwfx: Puntero al formato de cabecera.

Devuelve

(DWORD): Número de muestras por bloque.

atvsIsValidFormat

Función encargada de verificar que la cabecera wave pasada como argumento se trata de una cabecera ATVSFORMATEX válida que éste driver puede manejar.
Las comprobaciones realizadas para verificar que se trata de una cabecera válida son:

- Puntero a cabecera no nulo.
- Comprobación que la etiqueta de formato coincide con la asignada a Atvs-2.0.
- Comprobar que el número de canales no sea inferior a 1 ni superior al número de canales que el códec puede manejar (2 en Atvs-2.0).
- Comprobamos que la versión ACM de la cabecera es la misma que la del códec (1.0 en Atvs-2.0).
- Verificamos que el campo *Bytes de formato adicionales* es lo suficientemente grande para albergar la cabecera Atvs.

En el caso de que se cumplan todos los requisitos anteriores inicializamos la cabecera Atvs empleando la función atvs_packet_to_header presente en la librería Atvs-2.0 del códec.
Si se cumplen todos los pasos la función retorna devolviendo TRUE. En caso de fallar en algún requisito lo hará devolviendo FALSE.

Argumentos

LPWAVEFORMATEX pwfx: Puntero al formato de cabecera a verificar.

Devuelve

(BOOL): 0 si la cabecera NO es válida. Otro valor en caso contrario.

Funciones propias de la conversión

Las siguientes funciones se encargan de identificar el número de muestras presente en un buffer y la conversión de dichas muestras. La determinación de la función de conversión que se utilizará la realiza la función de inicialización del flujo acmdStreamOpen descrita en la subsección anterior.

pcmM08BytesToSamples **pcmM16BytesToSamples**
pcmS08BytesToSamples **pcmS16BytesToSamples**

Devuelven el número de muestras en un buffer PCM del formato especificado.

Argumentos

HDRVR cb: Tamaño del buffer en bytes.

Devuelve

(DWORD): La longitud del buffer en muestras.

atvsEncode_M08 **atvsEncode_M16**
atvsEncode_S08 **atvsEncode_S16**

Rutinas de codificación. Codifican un buffer de información de PCM a ATVS en el formato indicado. Se llama a la función de codificación apropiada una vez por cada mensaje ACMDM_STREAM_CONVERT recibido.
Dado que las funciones deben tener, por requisitos de ACM el mismo prototipo que las funciones de codificación, no todos los argumentos o parámetros son usados en ellas.

Argumentos

HPBYTE pbSrc: Puntero al buffer fuente (información codificada PCM).
DWORD cbSrcLength: Longitud en bytes del buffer fuente.
HPBYTE pbDst: Puntero al buffer destino (información codificada ATVS).
UINT nBlockAlignment: Alineamiento de bloque en bytes de la información ATVS. Parámetro no utilizado en la codificación.

Devuelve

(DWORD): Número de bytes utilizados en el buffer destino.

atvsDecode_M08 **atvsDecode_M16**
atvsDecode_S08 **atvsDecode_S16**

Rutinas de decodificación. Decodifican un buffer de información de ATVS a PCM en el formato indicado. Se llama a la función de decodificación apropiada una vez por cada mensaje ACMDM_STREAM_CONVERT recibido.

Argumentos

HPBYTE pbSrc: Puntero al buffer fuente (información codificada ATVS).
DWORD cbSrcLength: Longitud en bytes del buffer fuente.
HPBYTE pbDst: Puntero al buffer destino (información codificada PCM).
UINT nBlockAlignment: Alineamiento de bloque en bytes de la información ATVS.

Devuelve

(DWORD): Número de bytes utilizados en el buffer destino.

8.2 Listado de entregables generados

A lo largo del desarrollo del proyecto y de cara a satisfacer las tres entregas planificadas con el cliente, se han generado los siguientes entregables:

Entrega 1 **EVALUACIÓN DE DIFERENTES CODIFICADORES**	Julio de 2008

- **Entregable 1:** Informe sobre las pruebas de diferentes codificadores.
- **Entregable 2:** Presentación entrega 1.

Entrega 2 **CODIFICADOR PROPIETARIO SOBRE MIPSEL Y LINUX**	Diciembre de 2008

- **Entregable 1:** Informe sobre profiling del código y pruebas del decodificador.
- **Entregable 2:** Manual sobre estructuración del código fuente, instalación y formato de salida del codificador.
- **Entregable 3:** Presentación entrega 2.
- **Entregable 4:** Software ATVS-1.0

Entrega 3 **CODIFICADOR PROPIETARIO SOBRE MIPSEL, LINUX Y CÓDEC-DRIVER ACM**	Mayo de 2008

- **Entregable 1:** Descripción formato de archivo ATVS-2.0.
- **Entregable 2:** Descripción códec-driver ACM ATVS.
- **Entregable 3:** Cambios en ATVS-Linux y ATVS-Mipsel para su adaptación al formato 2.0.
- **Entregable 4:** Manual sobre estructuración del código fuente, instalación y funcionamiento del codificador y driver ACM.
- **Entregable 5:** Presentación entrega 3.
- **Entregable 6:** Software ATVS-2.0

A continuación incluimos el manual de instalación y funcionamiento (Entrega 3, Entregable 4) como ejemplo más interesante de los documentos entregados.

MANUAL SOBRE ESTRUCTURACIÓN DEL CÓDIGO FUENTE, INSTALACIÓN Y FUNCIONAMIENTO DEL CODIFICADOR Y DRIVER ACM

Descripción del documento

Este documento detalla la estructuración del código fuente del códec y el driver ACM desarrollado, un pequeño manual de instrucciones para su compilación e instalación tanto en un PC como en una unidad MIPSEL y las distintas opciones y modos de funcionamiento por consola del codificador Linux/mipsel.

ÍNDICE

DESCRIPCIÓN DE LA ESTRUCTURA DEL CD

El CD contiene los ficheros necesarios para la instalación completa del códec tanto en un PC equipado con linux como en una unidad MIPSEL, así como el Driver ACM para Windows. Los distintos ficheros se han organizado según la siguiente estructura:

- 🗁 **Fuentes:** Contiene los distintos códigos fuente desarrollados.
  - 🗁 **AtvsACM_libatvs-2.0:** Contiene la solución atvsacm de Visual Studio 2008 con los proyectos y código fuente necesarios para la compilación del driver, así como instrucciones para su compilación/instalación (README.txt).
  - 🗁 **AtvsLinux_libatvs-2.0**: Contiene el código fuente del códec ATVS-2.0 para Linux e instrucciones para su compilación/instalación (README.txt).
  - 🗁 **AtvsMipsel_libatvs-2.0**: Contiene el código fuente del códec ATVS-2.0 para Linux incorporado en la estructura de directorios necesaria para su instalación en la unidad MIPSEL, así como instrucciones para su compilación/instalación (README.pdf)
- 🗁 **Binarios:** Contiene los distintos códigos ejecutables desarrollados
  - 🗁 **AtvsACM_libatvs-2.0:** Contiene el driver ACM compilado, un fichero de entradas de registro e instrucciones para su instalación (README.txt).
  - 🗁 **AtvsLinux_libatvs-2.0**: Contiene los binarios atvsenc y atvsdec, librerías necesarias para su funcionamiento e instrucciones para su instalación (README.txt).
  - 🗁 **AtvsMipsel_libatvs-2.0**: Contiene los archivos que se copian en la tarjeta Compact Flash tras la instalación, así como instrucciones para su funcionamiento en la unidad MIPSEL (README.pdf).

ESTRUCTURACIÓN DEL CÓDIGO FUENTE

Los archivos relativos al códec desarrollado se encuentran dentro de la carpeta 🗁 **fuentes/ AtvsLinux_libatvs-2.0** del CD entregado. Dentro de ella el código se estructura de la siguiente manera:

🗁 **include:** Contiene los ficheros de cabecera necesarios.
🗁 **src**: Contiene el código fuente del codificador (atvsenc) y decodificador (atvsdec) por línea de comandos.
🗁 **libatvs:** Librería C necesaria para el funcionamiento del codificador.

COMPILACIÓN E INSTALACIÓN DEL CÓDEC EN UN PC

1. Copiamos la carpeta 🗁 **Fuentes/AtvsLinux_libatvs-2.0** en el PC destino de la instalación. A continuación, situándonos en la carpeta copiada, ejecutamos el siguiente script de configuración con privilegios de usuario **root** o **sudo** para compilar e instalar en Linux la versión en el PC equivalente a la que se instala en la unidad MIPSEL:

```
./COMPILAR_E_INSTALAR
```

   Este script se encarga de configurar e instalar tanto la librería libogg (necesaria para el funcionamiento del códec) como el propio códec. En el proceso de instalación se generan varias librerías dinámicas que se instalan por defecto en el directorio 🗁**/usr/local/lib**. También se copian varios ficheros de cabecera a 🗁**/usr/local/include** y se generan varios ejecutables que se guardan en 🗁**/usr/local/bin.**

   Es importante asegurarse de que el LD_LIBRARY_PATH incluye /usr/local/lib y que el PATH incluye /usr/local/bin para asegurar su correcto funcionamiento.

⚠ En la carpeta 🗁**Binarios/AtvsLinux_libatvs-2.0** se pueden encontrar los archivos que se generan durante la instalación en las carpetas 🗁bin (binarios atvsenc y atvsdec), 🗁lib (librerías necesarias para el funcionamiento del códec) e 🗁include (archivos de cabecera generados durante la instalación) en 🗁/usr/local. Podrían copiarse manualmente en caso de ser necesario. Ver README en la carpeta para más información.

COMPILACIÓN E INSTALACIÓN DEL CÓDEC EN LA UNIDAD MIPSEL

1. El entorno de compilación necesita conocer la localización del toolchain (carpeta 🗁 **toolchain** presente en la raíz del CD). Para ello debemos crear un enlace simbólico a él. Un enlace simbólico en sistemas Unix o Linux, indica un acceso a un directorio o fichero que no es real (se trata de un enlace) y se encuentra en un lugar distinto dentro de la estructura de directorios. Una modificación realizada utilizando este enlace se reflejará en el original, pero por el contrario, si se elimina el enlace, el original no será eliminado. Para crearlo no hace falta privilegios de root.

```
ln -s /toolchain/mipsel-linux   /usr/mipsel-linux
```

2. A continuación debemos modificar en el archivo path.sh presente dentro de la carpeta 🗁 **tools** [/tools/path.sh] las líneas que indican el directorio donde se encuentra el toolchain.

```
ROOT=/home/atvs/toolchain/
export PATH=$PATH:/home/atvs/toolchain/bin/
```

3. A continuación, una vez adecuado el fichero path.sh, para compilar el código ejecutamos el **script de bash** compile.sh presente en la carpeta 🗁 **tools.**

 Para el correcto funcionamiento es necesario ejecutar el script como bash y no como sh

```
bash compile.sh
```

4. Para proceder a la instalación debemos introducir la tarjeta Compact Flash que irá dentro de la unidad MIPSEL en un lector de tarjetas conectado al ordenador y montarlo en /mnt. Si el punto de montaje es diferente, puede ser modificado en el archivo install.sh modificando la línea TARGET=/mnt por la ruta deseada.

 La tarjeta compact flash debe estar en blanco. Si no lo está su contenido será sobreescrito.

5. Para proceder a la instalación ejecutamos el script de bash install.sh

```
bash install.sh
```

 En la carpeta 🗁**Binarios/AtvsMipsel_libatvs-2.0** se pueden encontrar los archivos que se copian en la tarjeta Compact Flash tras la instalación. Podrían copiarse manualmente en caso de ser necesario. Ver README en la carpeta para más información.

6. Desmontamos la tarjeta compact flash del ordenador y la introducimos en la unidad MIPSEL.

7. Si es la primera vez que se instala el códec en la unidad, deberá realizar las siguientes operaciones en la propia unidad MIPSEL (conectándose por el puerto Ethernet o por el puerto serie):

    a. Mover el archivo /mnt/atvs al directorio /etc/init.d

```
mv /mnt/atvs    /etc/init.d
```

    b. Crear el siguiente enlace simbólico:

```
ln –s   /etc/init.d/atvs    /etc/init.d/S99atvs
```

    c. Para poder ejecutar atvsenc y atvsdec sin enlazar con /mnt/bin podemos copiar los archivos binarios:

```
cp /mnt/bin/atvs*  /bin/
```

8. Con esto la unidad quedará lista para su utilización. Por defecto con el arranque de la unidad se arrancará también el demonio de grabación continua y el servidor web. Para terminar su procesamiento, iniciarlo ó reiniciarlo (parada e inicio) basta con ejecutar:

```
/etc/init.d/atvs {stop | start | restart}
```

UTILIZACIÓN POR LÍNEA DE COMANDO

El codificador tiene dos ejecutables, atvsenc y atvsdec, para codificar y descodificar, respectivamente. Los dos permiten manejar ficheros con cabeceras WAV tanto codificados como descodificados (descodificados es equivalente en este contexto a ficheros con codificación PCM).

Estos dos ejecutables tienen una ayuda en línea que aparece al llamar al programa sin argumentos de línea de comando.

CODIFICADOR ATVSENC

El codificador por línea de comando atvsenc tiene como estructura de llamada la siguiente:

```
atvsenc [OPCIONES] fichero_entrada fichero_salida
```

- Como fichero de entrada admite los formatos:
  - – (stdin). Entrada estándar si se emplea en pipes de linux. En este caso se supone formato raw PCM.
  - Fichero wav (extensión .wav).
  - Fichero raw PCM (se supone este formato para ficheros con cualquier otra extensión).
- Como nombre del fichero de salida (que en todos los casos tendrá el formato ATVS 2.0) admite:
  - – (stdout).
  - Fichero codificado con cabecera wav y de cualquier extensión.

Estos ficheros se podrán decodificar mediante atvsdec y reproducir y decodificar en Windows gracias al driver ACM.

- Las opciones disponibles para el codificador son las siguientes:

OPCIÓN	SIGNIFICADO
-n –narrowband	Fichero de entrada banda estrecha (8kHz)
-w –wideband	Fichero de entrada banda ancha (16kHz)
-u –ultra-wideband	Fichero de entrada banda ultra ancha (32kHz)
--quality n	Calidad de codificación (0-10). 8 por defecto
--bitrate n	Tasa binaria de codificación.
--comp n	Complejidad de codificación (0-10). 3 por defecto
--denoise	Filtrado de ruido antes de codificar
--agc	Aplica control de ganancia adaptativo (AGC)
-h –-help	Lista opciones disponibles
-v –version	Información sobre versión del códec
-V	Muestra el bitrate (modo verbosa)

- Las opciones disponibles relativas a la información necesaria para el procesamiento de los ficheros raw PCM son:

INFO FICHERO RAW	SIGNIFICADO
--rate n	Tasa de muestreo del fichero de entrada
--stereo	Considera fichero de entrada como estéreo
--le	Fichero raw de entrada little endian
--be	Fichero raw de entrada big endian
--8bit	Fichero raw de entrada es 8 bits sin signo
--16bit	Fichero raw de entrada es 16 bits con signo

DECODIFICADOR ATVSDEC

El decodificador por línea de comando atvsdec tiene como estructura de llamada la siguiente:

```
atvsdec [OPCIONES] fichero_entrada fichero_salida
```

- Como nombre del fichero de entrada (en todos los casos con formato ATVS 2.0) admite los formatos:
  - – (stdin)
  - Cualquier fichero codificado mediante atvsenc o el driver ACM de Windows.
- Como fichero de salida admite:
  - Fichero wav (extensión .wav).
  - Fichero raw PCM con cualquier extensión.
  - Nada (reproducción mediante tarjeta de sonido).
  - – (stdout): Salida estándar si se emplea en pipes de linux. En este caso el formato de salida es raw PCM.

- Las opciones disponibles para el decodificador son las siguientes:

OPCIÓN	SIGNIFICADO
--enh	Habilitar mejora perceptual
--no-enh	Deshabilita mejora perceptual
--force-nb	Forzar decodificación en banda estrecha
--force-wb	Forzar decodificación en banda ancha
--force-uwb	Forzar decodificación en banda ultra ancha
--mono	Forzar decodificación en mono
--stereo	Forzar decodificación en estéreo
--rate n	Forzar decodificación con tasa de muestreo n
-h –-help	Lista opciones disponibles
-v –version	Información sobre versión del códec
-V	Muestra el bitrate (modo verbose)

DRIVER AUDIO COMPRESSION MANAGER (ENTORNO WINDOWS)

Estructuración del código fuente

Los archivos relativos al driver desarrollado se encuentran dentro de la carpeta 🗁 **Fuentes/AtvsACM_libatvs-2.0** del CD entregado. Dentro de ella el código se estructura de la siguiente manera:

🗁 **ACM:** Contiene el código fuente del driver.
🗁 **atvs-2.0**: Se hace uso de la librería libatvs y de los archivos de cabecera del códec Atvs-2.0 presentes en la carpeta. Son los mismos que se utiliza en los codificadores Linux y mipsel.

Compilación e instalación del driver.

Los archivos relativos al driver desarrollado se encuentran dentro de la carpeta 🗁 **Fuentes/ AtvsACM_libatvs-2.0**

Contiene la solución atvsacm de Visual Studio 2008 con los proyectos necesarios para la compilación del driver: **libatvs** (que contiene la librería atvs-2.0) y **atvs** que contiene el código del propio driver.
Una vez generada la solución (compilar ambos proyectos), obtendremos el archivo **atvs32.acm** en la carpeta 🗁**ACM/atvs/Release**.

Para instalar el driver debemos copiar el archivo atvs32.acm obtenido en 🗁**C:/Windows/System32.**
A continuación ejecutaremos el fichero **acm_atvs.reg** que se encargará de añadir las siguientes entradas al registro de Windows:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Drivers32]
"msacm.atvsacm"="atvs32.acm"
```

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\drivers.desc]
"atvs32.acm"="Atvs ACM"

 En la carpeta 🗁**Binarios/AtvsACM_libatvs-2.0** se puede encontrar el archivo atvs32.acm ya generado y el fichero **acm_atvs.reg** encargado de añadir las entradas al registro.

Utilización del driver.

Una vez instalado, el driver funcionará en todas aquellas aplicaciones que utilicen los drivers ACM instalados en el equipo.

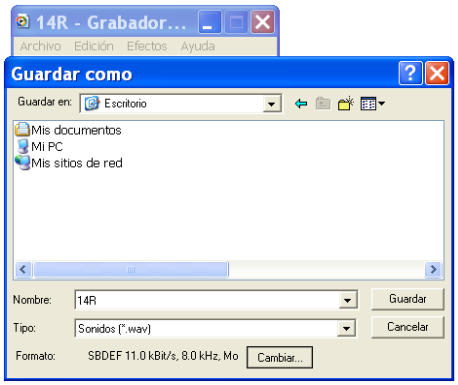
Se podrán reproducir los archivos codificados por Atvs-2.0 tanto en Linux como en el sistema empotrado en aplicaciones como el reproductor de Windows media:

También se puede realizar conversiones de formato empleando por ejemplo la grabadora de sonido de Windows:

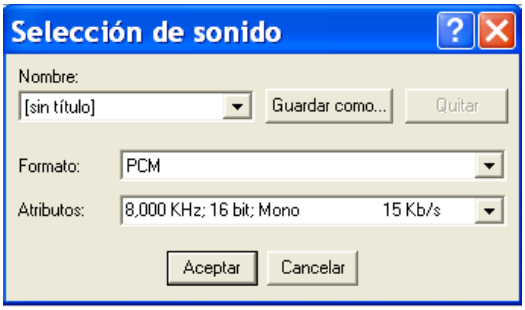
a. Abrimos un archivo codificado en el sistema empotrado:

b. Podemos reproducir el archivo. Si deseamos convertirlo a otro formato, pinchamos en Archivo ➔ Guardar como:

c. A continuación en Formato vemos la información del formato actual de codificación. En este caso ATVS de 11kbps y 8kHz de frecuencia de muestreo. (parte inferior de la ventana). Para cambiar el formato pinchamos sobre Cambiar…:

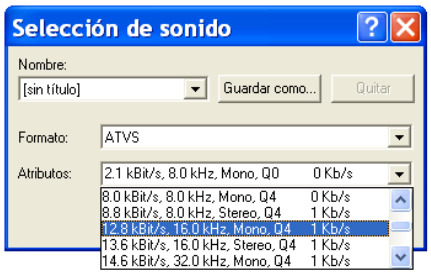


d. Elegimos el formato deseado entre las opciones disponibles. En este caso PCM a 8kHz.



e. Aceptando y guardando el fichero ya lo tenemos con codificación PCM.

El mismo procedimiento se puede seguir para codificar un archivo de PCM a Atvs-2.0, eligiendo en la ventana de selección de sonido la opción de formato deseada de entre las disponibles:



PRESUPUESTO

1 Ejecución Material

- Compra de ordenador personal (Software incluido)..... 2.000 €
- Alquiler de impresora láser durante 6 meses..... 50 €
- Material de oficina..... 150 €
- Total de ejecución material ..... 2.200 €

2 Ejecución Material

- 16 % sobre Ejecución Material..... 352 €

3 Beneficio Industrial

- 6 % sobre Ejecución Material..... 132 €

4 Honorarios Proyecto

- 640 horas a 15 € / hora..... 9600 €

5 Material fungible

- Gastos de impresión ..... 60 €
- Encuadernación ..... 200 €

6 Subtotal del presupuesto

- Subtotal Presupuesto..... 12060 €

7 I.V.A. aplicable

- 16% Subtotal Presupuesto..... 1929.6 €

8 Total presupuesto

- Total Presupuesto..... 13989,6 €

Madrid, Junio de 2009

El Ingeniero Jefe de Proyecto

Fdo.: Jesús Marcos Zamarreño
Ingeniero Superior de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un códec de voz propietario. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.
Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad “Presupuesto de Ejecución de Contrata” y anteriormente llamado ”Presupuesto de Ejecución Material” que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.