5. Las funciones de transcodificación.

El siguiente paso en el diseño de OperAIT es conseguir una herramienta *software* que nos permita tratar los datos a transferir por la red.

Los datos recibidos, aunque podrían ser utilizados por una aplicación genérica, simplemente se descartan en OperAIT. Esto es así porque el usuario final se comunica con el programa mediante la pulsación de teclas, y la detección de estas pulsaciones la lleva a cabo el propio controlador de emulación CAPI 2.0.

Las herramientas de transcodificación están incluidas en un módulo escrito en lenguaje C llamado `isdn2wav`. Este módulo no sólo proporciona funciones de transcodificación: además incluye funciones sencillas para escribir y leer cabeceras WAV, de forma que los datos recibidos puedan almacenarse en ficheros de este tipo y los datos a enviar puedan leerse de estos ficheros.

5.1. El formato de datos vocales RDSI.

Los datos vocales RDSI utilizados tienen las siguientes características:

- Cada muestra ocupa 8 bits.
- La codificación utilizada es Ley-A.
- El número de canales enviados es uno (monoaural).
- El número de **muestras** por segundo es 8000.
- Las muestras se envían comenzando por el bit menos significativo.
- Los bits impares se complementan, es decir, se cambia su valor lógico.

En definitiva, se transmiten 8000 muestras de 8 bits por segundo, utilizando codificación mediante Ley-A, comenzando por el bit menos significativo y complementando los bits impares.

5.1.1. La codificación en Ley-A.

La codificación en Ley-A es una forma de representación de datos que se conoce con el nombre de *compansión*. Esta forma de codificación permite que las muestras más probables en una comunicación vocal se describan con mayor detalle que las muestras menos probables.

De esta forma se consigue -en media- una mayor calidad en la comunicación. De hecho, la codificación mediante Ley-A consigue una calidad similiar a la que ofrece una codificación utilizando modulación por codificación de pulsos -PCM- de 12 bits. Por supuesto, la calidad será mayor si utilizamos PCM de 12 bits, pero Ley-A nos permite acercarnos mucho a esta calidad -es decir, tener pocas pérdidas de compresión- reduciendo el tamaño de muestra en un 33% y con unos *codecs* muy sencillos.

La siguiente tabla muestra la relación entre la codificación PCM de 12 bits y la codificación en Ley-A de 8 bits. El sistema más sencillo para crear un codificador de Ley-A es utilizar un codificador PCM de 12 bits y crear las muestras en Ley-A según esta tabla. Del mismo modo, si se desea obtener el valor PCM de 12 bits correspondiente a una muestra en Ley-A -por ejemplo, para guardar las muestras en un fichero PCM- hay que consultar la tabla 4 y rellenar con ceros los bits menos significativos de la muestra PCM, si corresponde (bloques 2 al 7).

PCM (12 bits)												***Ley-A (8 bits)***		
Signo	10	9	8	7	6	5	4	3	2	1	0	Signo	Bloque	Datos
s	1	X	X	X	X							s	111	XXXX
s	0	1	X	X	X	X						s	110	XXXX
s		0	1	X	X	X	X					s	101	XXXX
s			0	1	X	X	X	X				s	100	XXXX
s				0	1	X	X	X	X			s	011	XXXX
s					0	1	X	X	X	X		s	010	XXXX
s						0	1	X	X	X	X	s	001	XXXX
s							0	X	X	X	X	s	000	XXXX

Tabla 4. Relación PCM 12 bits - Ley-A.

La ilustración 6 muestra el tamaño del paso de cuantización -la inversa de la resolución- de la muestra codificada en Ley-A en relación a la amplitud de la señal a codificar, siendo el valor máximo de esta última 2047 unidades.

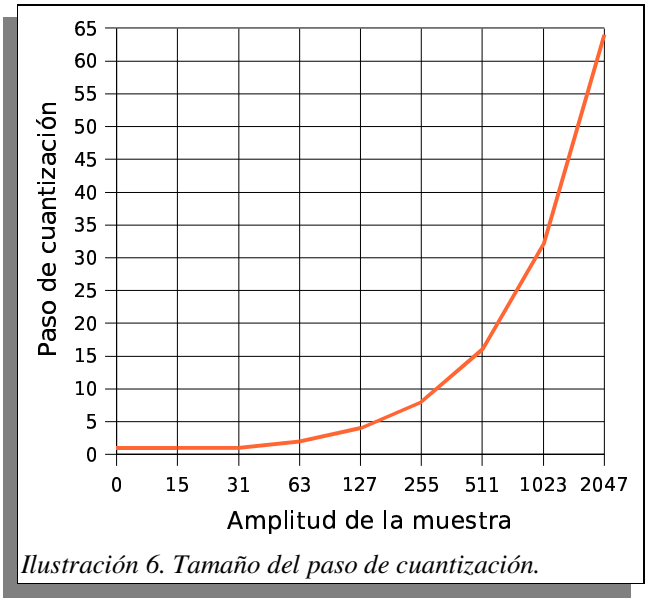

Ilustración 6. Tamaño del paso de cuantización.

Podemos ver que el paso de cuantización es menor -y por lo tanto, la resolución es mayor- para muestras de menor amplitud. En cambio las muestras de mayor amplitud tienen un paso de cuantización mayor, lo que equivale a una menor resolución.

La ilustración 7 representa el valor de salida del cuantizador en función de la amplitud de la entrada, para valores positivos de ésta. Para valores negativos, la gráfica sería simétrica respecto de ambos ejes.

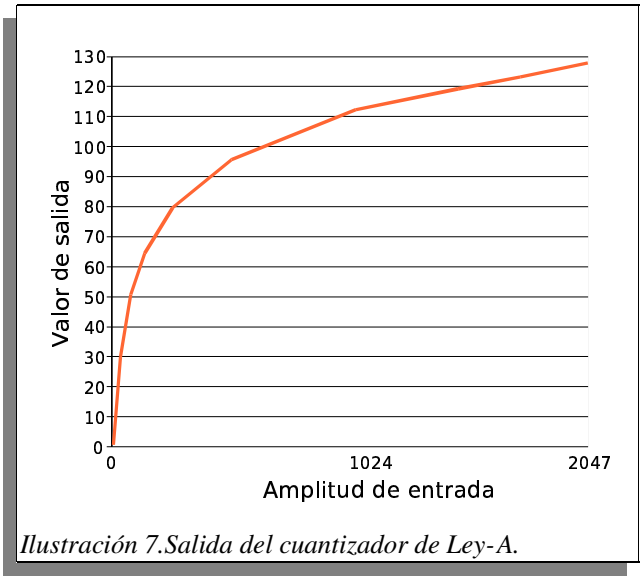

Ilustración 7.Salida del cuantizador de Ley-A.

En este caso puede verse con claridad la mayor concentración de valores de salida para bajas amplitudes, lo que en la gráfica aparece como una mayor pendiente para los valores pequeños de la entrada, frente a una pendiente baja para valores mayores. Vemos que, para amplitudes de entrada grande, la variación de la amplitud debe ser grande para causar una variación de una unidad en el valor de salida. En cambio, para amplitudes de entrada

pequeñas, para conseguir un cambio de una unidad en el valor de salida la variación de la entrada será también pequeña. Todo esto equivale a tener una resolución grande para amplitudes pequeñas y una resolución pequeña para amplitudes grandes.

5.2. Implementación de los *codecs* de audio.

Las funciones que realizan la transcodificación de audio se encuentran agrupadas en un módulo C llamado `isdn2wav`. El fichero de cabecera `isdn2wav.h` contiene los prototipos de todas las funciones exportadas. Además de las dos funciones que realizan la transcodificación en uno y otro sentido, se incluyen tres funciones más para tratar con ficheros WAV.

Puesto que hemos visto que las transcodificaciones de Ley-A a PCM 12 bits y viceversa son directas, es posible escribir dos funciones sencillas que transformen una muestra de una a otra codificación. Existe un pequeño problema: PCM de 12 bits no es un formato de audio muy utilizado, al no ser el número de bits un múltipo de 8. Por lo tanto lo que se ha implementado es las funciones de transformación entre Ley-A y PCM de 16 bits -este formato sí es ampliamente utilizado en archivos WAV-.

El modo de transcodificar se modifica entonces para ambos casos:

- PCM a Ley-A: primero se toma la muestra PCM de 16 bits y se desplazan todos los bits 4 posiciones a la derecha, lo que equivale a dividir la muestra por 16 -es decir, realizar una atenuación-. Se toman entonces los 12 bits menos significativos de esta operación y se realiza la transcodificación usual sobre ellos, obteniéndose la muestra en Ley-A correspondiente.
- Ley-A a PCM: se transcodifica la muestra de Ley-A a PCM de 12 bits, y se desplazan los bits del resultado 4 posiciones a la izquierda, lo que equivale a multiplicar la muestra por 16 -amplificarla en este caso-. El resultado serán 16 bits cuyos 4 bits menos significativos son ceros, y este es precisamente el valor PCM de 16 bits asociado a la muestra en Ley-A.

Las dos funciones que realizan la transcodificación tienen los siguientes prototipos:

```
void RDSIaPCM(unsigned char *buffer, short *bufsal, int longitud);
void PCMaRDSI(short *buffer, unsigned char *bufsal, int longitud);
```

En ambas, el parámetro `buffer` debe apuntar a la zona de memoria donde se almacenan las muestras a convertir, `bufsal` debe apuntar a la zona de memoria donde se almacenarán las muestras transformadas y `longitud` indica el número de muestras a transformar -no la longitud de ninguno de los *buffers*, como podría pensarse-.

Las otras funciones que están implementadas en el archivo `isdn2wav.c` son:

```
void ReservaCabecera(int fd);
void ReescribeCabecera(int fd, unsigned long longitud);
unsigned CompruebaWAV(int fd);
```

Las dos primeras permiten escribir una cabecera RIFF al inicio de un fichero. `ReservaCabecera` simplemente escribe una cabecera vacía, mientras que `ReescribeCabecera` rellena todos los campos de esta cabecera con los valores adecuados al formato de las muestras -PCM de 16 bits monoaural a 8 KHz- indicando que el número de muestras es igual al parámetro `longitud`.

5.2.1. Excepciones en la transcodificación.

Hay que destacar dos hechos curiosos que se producen en la transcodificación, ambos debidos a que Ley-A utiliza codificación en signo-magnitud y PCM utiliza complemento a dos.

- El máximo -en valor absoluto- valor negativo que puede ser representado en PCM de 16 bits, -32768, no puede ser representado -previo desplazamiento de bits- mediante Ley-A. Esto es así porque el valor binario PCM de 12 bits correspondiente, al ser transformado usando la tabla 4, se convierte en el valor 0 en Ley-A. Este hecho se produce porque, al utilizar la codificación en Ley-A una representación en signo-magnitud, el valor 0 puede representarse de dos formas distintas y una de ellas ocupa el valor asociado por la tabla 4 a -32768. La solución consiste en modificar la muestra PCM para que sature a -32767, que sí puede representarse en signo magnitud con 12 bits, antes de realizar la transcodificación.
- Al tener el valor 0 dos representaciones distintas en Ley-A, hay que decidir cuál de ellas utilizar al convertir el valor PCM 0. La función PCMaRDSI codifica siempre el 0 sin signo, es decir, con el bit de signo a cero. Si tomamos entonces un bloque de datos en Ley-A genérico, en el cual pueden aparecer los valores 0 con signo y sin signo, los transcodificamos a PCM, volvemos a transcodificar a Ley-A y comparamos la salida con los datos originales, pueden existir diferencias en las muestras que codifiquen un 0. Es importante reseñarlo para que nadie piense que la transcodificación ha sido incorrecta si realiza dicha prueba.

Ambos efectos son totalmente imperceptibles para el oído, por lo que no producen una pérdida de calidad en absoluto, pero es necesario tenerlos en cuenta a la hora de diseñar los *codecs*.