Departamento de Ingeniería Electrónica
E.T.S.I. de Telecomunicación
Universidad Politécnica de Madrid

Laboratorio de Sistemas Electrónicos Digitales

Enunciado de la práctica estándar del Laboratorio de Sistemas Electrónicos Digitales (LSED)

Sistema de procesamiento digital de señal para el MCF5272

Plan 94. Curso 2011-2012. Versión 1.4

Autores: equipo docente de LSED

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Introducción

El objetivo del *Laboratorio de Sistemas Electrónicos Digitales* es que el alumno adquiera práctica en el desarrollo de sistemas con interacción entre HW y SW y con requisitos de tiempo real, aplicando y consolidando de una manera práctica los conocimientos adquiridos principalmente en las asignaturas de tercer curso *Sistemas Electrónicos Digitales* y *Laboratorio de Circuitos Electrónicos*, partiendo de la base sobre programación adquirida en *Fundamentos de la Programación* y *Laboratorio de Programación*.

Para ello deberá seguir las instrucciones aquí incluidas, que implicarán diversas fases de diseño, análisis, implementación y medida de los circuitos y programas propuestos. Igualmente se hará especial énfasis en que los alumnos adquieran una visión práctica de los problemas con los que se encuentra el diseño del hardware (HW) y el software (SW) de sistemas electrónicos a la hora de implementar prototipos reales de Laboratorio.

El resultado del trabajo realizado debe quedar reflejado en **una memoria final** que contenga los detalles del proceso, así como los resultados obtenidos y todas aquellas cuestiones específicas que se indiquen en el enunciado. **Es obligatorio entregar electrónicamente el programa en desarrollo a través del portal** http://lsed.die.upm.es de acuerdo con las normas generales del LSED [1] publicadas previamente.

Salvo que se indique lo contrario, la práctica propuesta contiene las **especificaciones mínimas obligatorias** que deben cumplir los sistemas y serán valoradas con un máximo de **8 puntos si se realiza en C y 8,5 puntos si se realiza en ensamblador**. **Esta nota máxima sólo se conseguirá si se cumplen todas las especificaciones, y la memoria, el código, el hardware y las respuestas en el examen oral son perfectos**. **La descripción del sistema de menús de la interfaz de usuario es orientativa,** aunque en la evaluación se valorará la calidad de la interfaz desarrollada. Las desviaciones respecto a la interfaz de este enunciado deben ser explicadas en la memoria final.

Adicionalmente a las especificaciones mínimas, se presentarán sugerencias de **mejoras opcionales**, dejando a los alumnos la libertad para que añadan nuevas mejoras o esquemas alternativos. Con estas mejoras o montajes alternativos añadidos al prototipo básico, y dependiendo de su dificultad y realización, se podrá alcanzar la máxima nota, 10 puntos. Añadir una mejora no garantiza que la nota final esté por encima de **8 puntos si se realiza en C y 8,5 puntos si se realiza en ensamblador**; sólo ofrece la posibilidad de sumar puntos sobre la nota de la parte básica de su sistema, de su examen oral y de su memoria final.

Sobre otras modalidades de práctica distintas de la estándar, remitimos al alumno a la información general de la asignatura [1]. Aquellos alumnos que deseen realizar una **práctica innovadora** basada en un problema o un diseño propios (o propuesto por un profesor), deberán hablar con alguno de los profesores de la asignatura y presentarle un listado de notas y una propuesta de práctica donde describan en 2 o 3 páginas cuáles son los objetivos del sistema propuesto, qué recursos son necesarios para llevarlo a cabo, así como las arquitecturas HW y SW propuestas para la resolución del problema. Para poder realizar la práctica será necesario contar con la aprobación de dicho profesor. No será admitida ninguna práctica (por muy compleja o perfecta que sea) que no se ajuste a estas normas.

Sobre las consideraciones éticas del trabajo en el laboratorio, remitimos al alumno al documento publicado sobre la filosofía de los laboratorios LCEL y LSED, disponible a través del portal de la asignatura [2].

Para cualquier consulta, no duden en dirigirse al coordinador Roberto Barra Chicote (B-112, barra@die.upm.es), o a cualquiera de los profesores del LSED en sus respectivos horarios de tutoría. Podrá encontrar éste y otros documentos relacionados, así como información actualizada sobre la asignatura, en http://lsed.die.upm.es/. Adicionalmente, el alumno dispondrá de un conjunto de vídeos explicativos y demostrativos sobre esta práctica y este enunciado.

2. Aviso inicial

Cuando aborde la lectura de este documento, hágalo con tranquilidad y detenimiento. Frases o comentarios que no se entiendan en una primera lectura pueden encerrar avisos y recomendaciones que le serán útiles a lo largo del desarrollo del Laboratorio.

No se preocupe si no alcanza a comprender todos los términos, conceptos y detalles que se discuten. Todos ellos se irán aclarando a medida que avance en la lectura de este documento. Por supuesto, asuma que necesitará varias lecturas y una reflexión a fondo sobre todo esto.

Preste especial atención a todas las referencias explícitas a aspectos que se indican como obligatorios para incluir en la memoria: no quiere decir que algo no referenciado explícitamente no tenga que ser tratado, sino que nuestra experiencia demuestra que algunos de esos aspectos no son considerados por un cierto número de alumnos, lo que da lugar a desagradables sorpresas en los exámenes.

A lo largo de este enunciado irá descubriendo que multitud de detalles imprescindibles de solventar en un sistema real, se dejan de lado o se simplifican notablemente. Es fundamental que tenga en cuenta que esta práctica pretende ser un ejercicio de diseño, implementación y prueba de sistemas electrónicos digitales (con una pequeña componente analógica), con lo que es seguro que encontrará decisiones de diseño y recomendaciones que harían imposible que el prototipo final construido pudiera llegar a formar parte de un sistema real.

3. Especificación de requisitos del sistema

3.1 Antecedentes

Son numerosas las aplicaciones de procesado digital de señal relacionadas con el procesamiento de señales de audio y voz. Existen numerosos sistemas electrónicos para la adquisición y manipulación de audio, tales como tarjetas de adquisición de alta calidad, amplificadores, sistemas de mezclas, sistemas de ecualización, generadores de efectos de sonido, etc. En el diseño abordaremos un procesador orientado a la banda de audio, aunque los conocimientos que el alumno adquiera serían aplicables a otros campos como la automoción o las telecomunicaciones digitales.

Uno de los elementos básicos de los sistemas de tratamiento de audio son los sistemas de ecualización. Un ecualizador es un módulo que permite modificar la respuesta en frecuencia de un sistema de audio. Es habitual realizar esta modificación mediante la variación de los parámetros de

un conjunto de filtros, modificando así la señal de entrada y adaptando ésta a las condiciones acústicas deseadas (local, tipo de música, etc). Existen diversos tipos de ecualizadores, de entre los que destacan:

- Los ecualizadores paramétricos, aquellos en los que se pueden modificar la frecuencia central, ancho de banda, selectividad, etc, además de la propia ganancia.
- Los ecualizadores gráficos, los cuales emplean un banco de filtros cuyo único parámetro ajustable es la ganancia de cada filtro. Reciben el nombre de "gráficos" debido a que la disposición los controles permiten visualizar rápidamente la respuesta deseada.

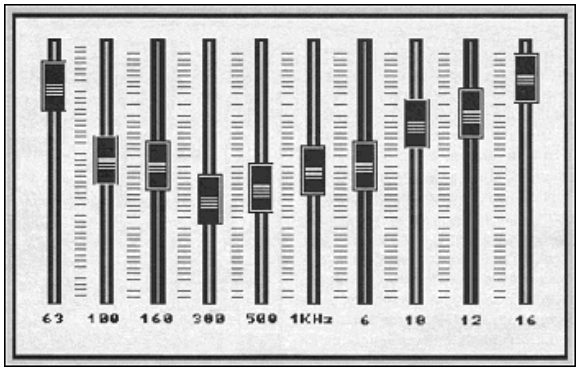

Figura 1: Ejemplo de ecualizador gráfico

En la presente práctica supondremos que un cliente (el Departamento de Ingeniería Electrónica) nos ha contratado para desarrollar un prototipo de un sistema de ecualización de audio y generador de efectos de bajo coste, simplificado e inspirado en los sistemas de ecualización que incorporan los sistemas de audio comerciales. Dado que el propósito de este laboratorio es eminentemente docente (con especial énfasis en la interacción HW-SW, así como en los requisitos de tiempo real), el objetivo será implementar un prototipo funcional completo. Este enunciado constituye una guía para su diseño e implementación.

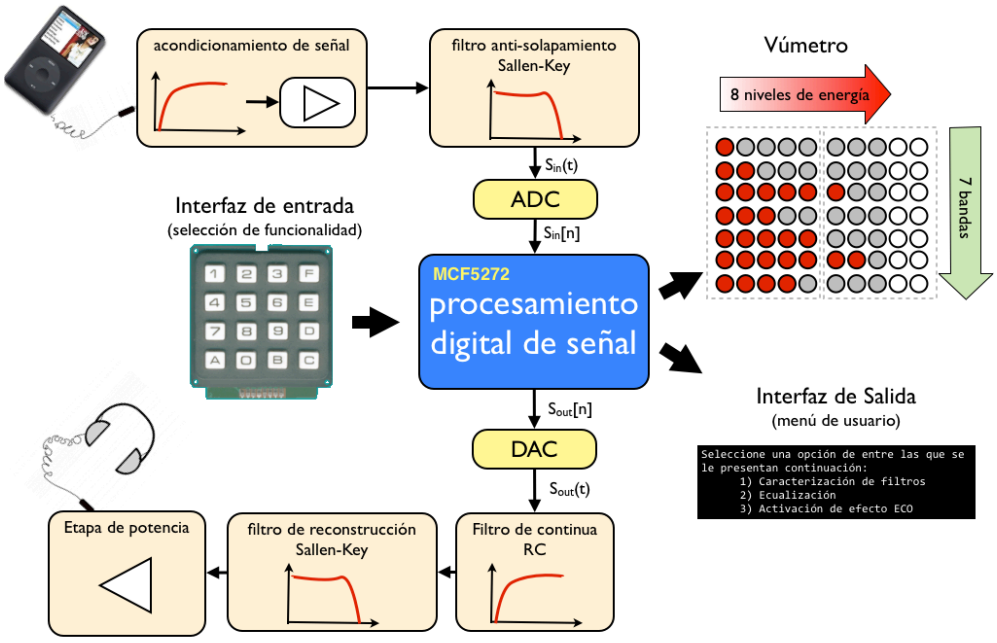

Figura 2: Arquitectura general del sistema de ecualización y aplicación de efectos de audio propuesto.

3.2 Objetivo general

El objetivo de la práctica es mostrar la viabilidad de la idea de diseño básica desarrollando un prototipo plenamente funcional. El programa que ejecutará el micro se realizará en **ensamblador o en C** (ambos están incorporados en el entorno de desarrollo EDColdFire; la calificación máxima de la práctica básica es **8 puntos si se realiza en C y 8,5 puntos si se realiza en ensamblador**). El sistema digital basado en un microprocesador será la plataforma ENT2005CF disponible en el laboratorio B-043 (construida en torno a un MCF5272). Para más detalles relacionados con el sistema de desarrollo, consulte la publicación [4].

En los apartados siguientes se detallarán las arquitecturas HW y SW, haciendo énfasis en la **descomposición modular** del sistema, tarea clave para abordar con éxito el diseño de cualquier sistema HW o SW medianamente complejo.

El objetivo general lo podemos resumir de la siguiente manera, desarrollar "un sistema de procesado digital de audio basado en la plataforma ENT2005CF" que permita:

- La **ecualización** de la señal de audio de entrada mediante la modificación de los niveles de energía de 7 bandas de frecuencia. Dicha modificación se llevará a cabo mediante el procesamiento digital de la señal analógica de entrada (tras su previa conversión a digital mediante el conversor analógico-digital, disponible en la plataforma ENT2004CF[3]) mediante el filtrado digital por medio de 7 filtros paso banda de tipo IIR (*Infinite Impulse Response*) implementados digitalmente. El sistema debe muestrear, procesar y reproducir señales de audio de hasta 3,4 kHz, de una manera indefinida y síncrona en muestra.
- La **caracterización** de cada uno de los **filtros** digitales implementados: se implementará la posibilidad de emplear cada filtro de forma aislada.
- La activación de un **efecto de reverberación simple**, que se aplicará a la señal de entrada mediante procesado digital de señal. Los parámetros de entrada configurables de este efecto serán el nivel de señal de dicha reverberación y su retardo.
- La visualización del nivel de energía de cada banda a través de un **vúmetro** implementado con la matriz de LEDS disponible en cada uno de los puestos del LSED. Dicha visualización estará activa en todo momento, independientemente de la funcionalidad seleccionada (caracterización, ecualización o aplicación de la reverberación simple).
- La comunicación con el usuario del sistema para seleccionar una de las tres funcionalidades disponibles: caracterización de filtros, ecualización y efecto de reverberación simple. La **interfaz de usuario** se implementará por medio del control del teclado matricial disponible en la plataforma ENT2004CF y la realimentación de información al usuario a través de la pantalla del ordenador.

4. Sistema de procesamiento digital de audio

4.1 Justificación y esquema general

Un sistema de procesamiento digital de señal dispone de entradas y salidas que serán señales analógicas, pero realiza el procesamiento intermedio de modo digital (en un microprocesador como el del LSED, por ejemplo). Las ventajas de dicho procesado son conocidas:

- **Menor complejidad** frente al procesado analógico. El problema se reduce a programar un procesador (sea de propósito general, como el de los ordenadores PC, o diseñado específicamente para esta tarea, caso de los DSPs).
- Es **menos sensible** a las tolerancias de los componentes hardware.
- Las señales son **almacenadas sin apenas pérdidas de calidad**.
- El **procesado** digital puede ser realizado de manera **off-line**, a partir de las muestras almacenadas (en disco duro, CD-ROM...).

Figura 3: Esquema general de un sistema de procesamiento digital de señal.

Un sistema de procesamiento digital de señal constará de tres componentes:

- Un subsistema de conversión Analógico/Digital
- Un módulo que realice el procesado digital (basado en un microprocesador, un microcontrolador o un DSP)
- Un subsistema de conversión Digital/Analógico

A continuación daremos detalles sobre cada una de los bloques del sistema. Nuestro enfoque será práctico, orientado al LSED. En cuanto a teoría de procesado de señal y sistemas lineales digitales, recomendamos al alumno la lectura de las referencias bibliográficas (A. Oppenheim et al, 1997) o (A. Oppenheim et al, 1999)

4.2 Subsistema de conversión Analógico/Digital

Las etapas básicas de un subsistema de conversión Analógico/Digital son:

- una etapa de acondicionamiento de señal formada por una etapa de adaptación de impedancias eliminación de continua y una etapa pre-amplificadora;
- un filtro anti-solapamiento (o *anti-aliasing*),
- un circuito de retención y muestreo (*Sample & Hold)* y
- un convertidor analógico digital (ADC).

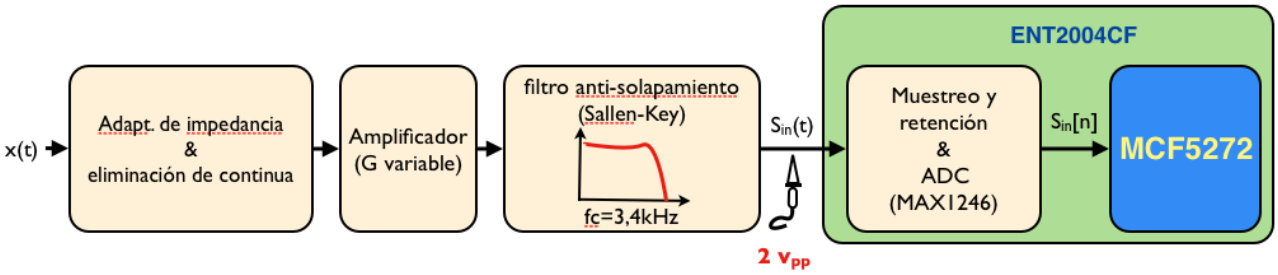

Figura 4: Diagrama de bloques del subsistema de conversión analógico/digital basado en la plataforma ENT2004CF

4.2.1 Acondicionamiento de señal

En nuestro caso, la señal de entrada *x(t)* será la señal que proviene de un reproductor externo (mp3 o similar) que incorporaremos a nuestro sistema conectando la salida de auriculares mediante un cable con conectores de tipo "*jack*" a la entrada de nuestro circuito. Conecte una resistencia de R=10Ω (entre la entrada del sistema y masa) para simular la impedancia de los auriculares.

Monte un filtro RC paso alto con una frecuencia de corte f_c que evite el paso de posible corriente continua y atenúe lo menos posible el resto de frecuencias. Estime finalmente el valor de dicha frecuencia de corte empleando el osciloscopio y el generador de funciones.

A la salida del filtro RC conecte un amplificador no inversor de ganancia variable mediante el uso de un potenciómetro y un amplificador operacional del tipo TL082. Deberá ajustar dicho potenciómetro para que la señal $\mathbf{s_{in}[n]}$ presente un valor de $\mathbf{2V_{pp}}$ una vez que haya montado el filtro anti-solapamiento. Calcule previamente, y estime finalmente, el valor de la ganancia de dicho amplificador.

4.2.2 Filtro anti-solapamiento

Este filtro anti-solapamiento será un filtro paso bajo con una frecuencia de corte f_c=3,4KHz (dado que los filtros no son ideales, optaremos por una f_c menor que 4kHz con el fin de atenuar lo suficiente las componentes de frecuencia por encima de la frecuencia de *Nyquist*. Para la implementación de este filtro paso-bajo se propone un filtro de 2º orden Sallen-Key, similar a los desarrollados en LCEL (consultar [14], apartado 4.1.1). Estime la respuesta en frecuencia (tanto en amplitud como en fase) del conjunto formado por la etapa de acondicionamiento y el filtro anti-solapamiento, introduciendo señales sinusoidales de amplitud $2V_{pp}$.

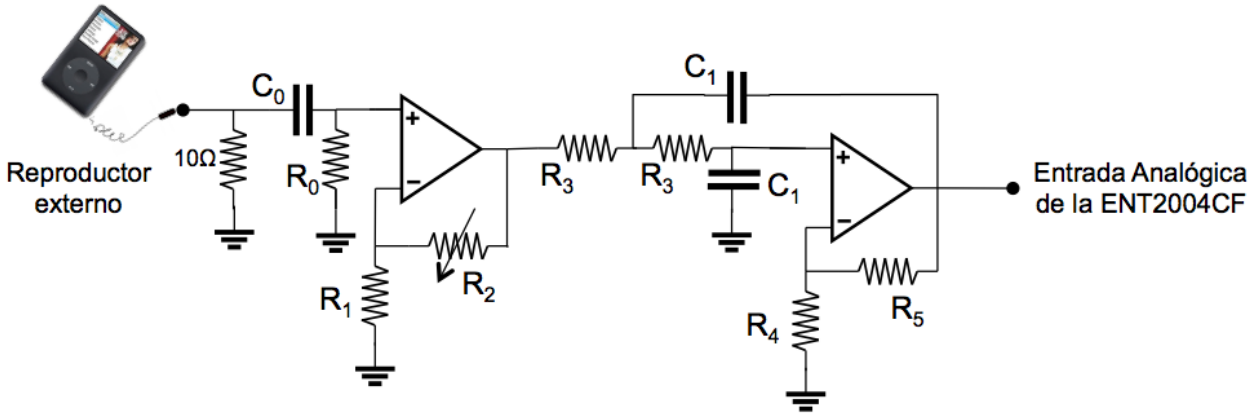

Figura 5: Esquema de adaptación de la señal de entrada y filtro paso bajo anti-solapamiento

4.2.3 Conversor Analógico/Digital

En nuestro caso, la etapas de muestreo y retención y ADC se implementan a través del ADC MAX1246 incorporado en la plataforma ENT2004CF y controlado por el MCF5272. Se deberá configurar el ADC en modo asimétrico para procesar señales bipolares (entre ±2,5V) con una frecuencia de muestreo de 8kHz. Consulte [3] o la hoja del fabricante del ADC MAXIM1246 para configurar apropiadamente el byte de control.

Importante, el esquema de protecciones de la placa limita la tensión de entrada a 4V, no pudiéndose medir correctamente tensiones por debajo de -1,3V. Por ello, se configurará la ganancia G del amplificador de la etapa de acondicionamiento para que introduzcamos una señal sin(t) en la entrada analógica de la plataforma ENT2004CF de 2Vpp. Se recomienda al alumno leer la documentación disponible respecto a los conceptos generales de conversión analógico/digital en el apartado 2.2.1 de [3] y las características y posibles modos de configuración del ADC MAX1246 en el apartado 2.2.2.1 de [3]. Así mismo, se recomienda al alumno familiarizarse con el control del ADC realizando el tutorial disponible en el Capítulo 8 de [3].

4.3 Subsistema de conversión Digital/Analógico

A la hora de reconstruir una señal analógica a partir de la secuencia de códigos binarios que la representa se empleará el siguiente esquema de la Figura 6:

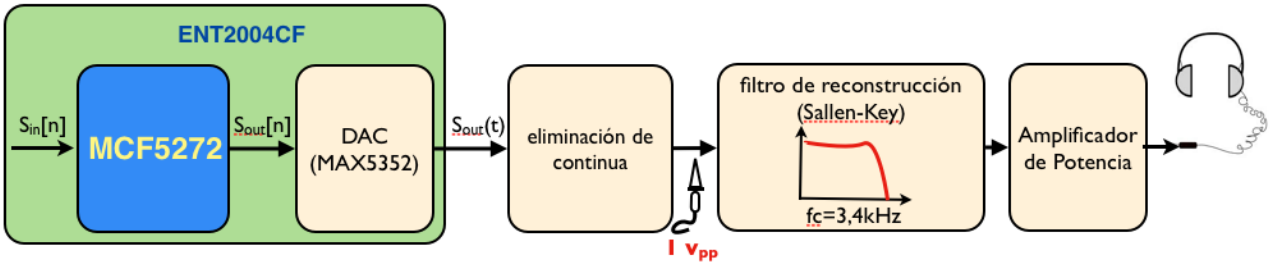

Figura 6: Diagrama de bloques del subsistema de conversión digital/analógico basado en la plataforma ENT2004CF

4.3.1 Conversor Digital/Analógico

La plataforma ENT2004CF dispone de un DAC MAX 5352, un conversor digital-analógico de 12 bits con interfaz serie, capaz de ofrecer tensiones de salida en un rango entre 0 y 2,5V, y cuya salida está disponible (convenientemente opto-acoplada) en uno de los conectores BNC hembra debidamente etiquetado en la caja.

Debido al rango de tensiones del DAC, la relación entre las tensiones de entrada del ADC y de salida del DAC es de 2; por lo que se deberá comprobar con el osciloscopio que a la salida del DAC obtenemos una señal de aproximadamente $1V_{pp}$ cuando a la entrada del ADC introducimos una señal de $2V_{pp}$.

Se deberá eliminar la componente continua de la señal que proviene del DAC mediante un filtro paso alto RC, con una frecuencia de corte f_c posible, que minimice la atenuación del resto de bajas frecuencias.

4.3.2 Filtro de reconstrucción

En la plataforma ENT2004CF no se ha implementado filtro de reconstrucción alguno por lo que, para poder reconstruir la señal analógica correspondiente al efecto de sonido a partir de la secuencia de códigos binarios que lo representa (i.e. muestras), es preciso disponer detrás del DAC (en nuestro caso detrás del filtro paso bajo RC) un filtro (equivalente al filtro anti-solapamiento del apartado 4.2.2) que vuelva a cumplir las restricciones del teorema de *Nyquist* para eliminar las copias espectrales en múltiplos de la frecuencia de muestreo que se producen a la salida del DAC (para más detalles generales acerca de la conversión entre los mundos digital y analógico y en particular del conversor DAC de la plataforma se puede consultar el apartado 2.2 de [3]).

4.3.3 Amplificador de potencia

La salida de los circuitos digitales (y de muchos amplificadores operacionales con los que se construyen los filtros) tiene una limitación importante en relación con la corriente que pueden ofrecer. En muchos casos, esta corriente no es suficiente para excitar unos altavoces. Por esta razón, es necesario incluir un módulo de amplificación de potencia previo a los altavoces.

Para la implementación de este módulo se recomienda utilizar el amplificador de potencia LM386 [13] utilizando alguno de los montajes que propone el fabricante (es necesario eliminar la componente continua antes de amplificar). En la Figura 7 se muestra el montaje más sencillo.

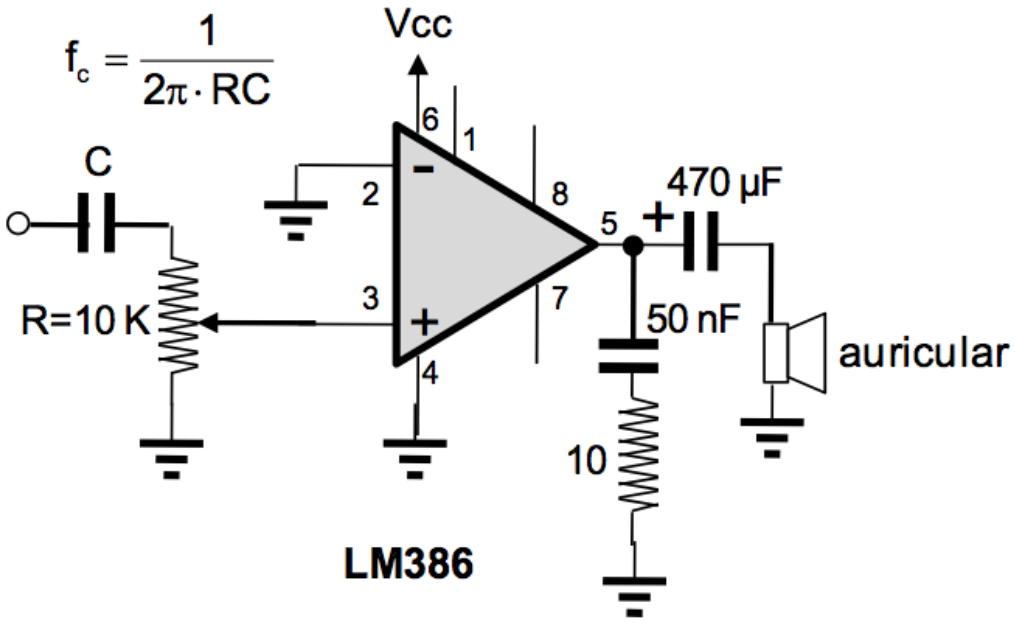

Figura 7: Amplificador de potencia basado en un LM386 en configuración de ganancia = 20

Un aspecto importante que el alumno debe tener en cuenta es que este amplificador presenta una ganancia entre 20 y 200. Por esta razón, se debe incluir una etapa de atenuación que reduzca el margen dinámico de la señal. Al realizar el montaje se recomienda al alumno que consulte las hojas de especificaciones del LM386 con el fin de ajustar correctamente los niveles de las señales y de la alimentación. La impedancia del altavoz se puede modelar aproximadamente como una resistencia de unos 10 ohmios.

También es necesario recordar la importancia del filtrado de alimentación típico del LCEL, apartado 5.10 de [14], especialmente cuando se conecta el amplificador de potencia, que puede requerir incluso una alimentación independiente. Si no se hace, el filtro puede funcionar mal debido al ruido que le entre por su alimentación.

Aunque la salida se oiga razonablemente bien es importante comprobar con el osciloscopio el correcto funcionamiento tanto del filtro paso-bajo como del amplificador de potencia. Cuando se compruebe el funcionamiento del amplificador de potencia de forma aislada es importante probarlo

utilizando señales sinusoidales (obtenidas del generador de funciones) con el fin de detectar posibles problemas de saturación.

4.4 Subsistema de procesamiento digital de audio

El procesado de la señal digital leída de la salida del ADC MAX1246 se llevará a cabo empleando el micro-controlador MCF5272. El alumno desarrollará un sistema software que lea la señal $S_{in}[n]$ del ADC, realice un filtrado en bandas de dicha señal, pudiendo variar la ganancia de cada uno de los filtros con el fin de ecualizarla y ofreciendo la posibilidad de añadir un eco. La señal digital resultante $S_{out}[n]$ se entregará en la salida analógica mediante el DAC MAX5352. La Figura 8 muestra el esquema general del procesamiento digital a implementar.

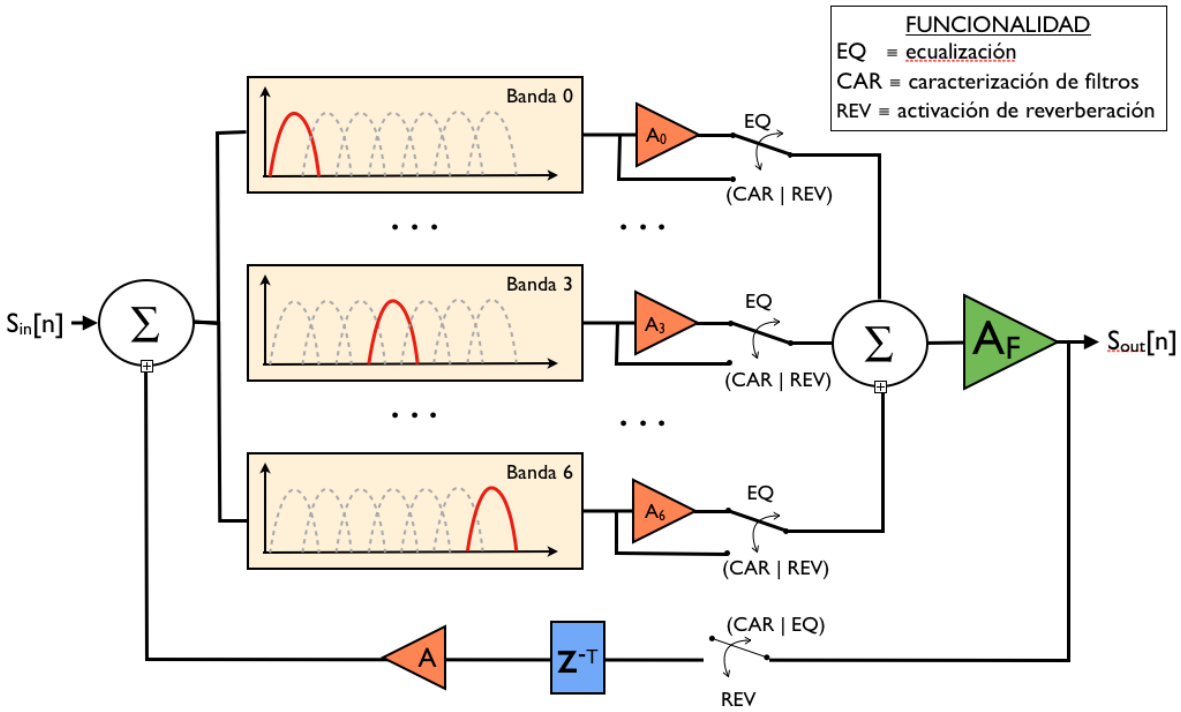

Figura 8: Esquema de filtrado, ecualización y adición de la reverberación simple a la señal de entrada

4.4.1 Caracterización de filtros digitales

El usuario tendrá la opción de filtrar la señal de entrada $S_{in}[n]$ seleccionando uno de los 7 filtros paso banda que estarán disponibles. Los filtros disponibles serán filtros paso-banda IIR de orden dos. La función de transferencia de cada filtro en términos de la transformada z es:

$$H[z] = \frac{G \cdot \left(1 + \sum_{k=1}^{k=2} b_i \cdot z^{-k}\right)}{1 + \sum_{k=1}^{k=2} a_i \cdot z^{-k}}$$

dónde G es un factor de ganancia (derivado de normalizar los coeficientes del numerador[1]), b_i y a_i son los coeficientes del numerador y denominador respectivamente. La Figura 9 presenta la respuesta en frecuencia teórica de cada filtro.

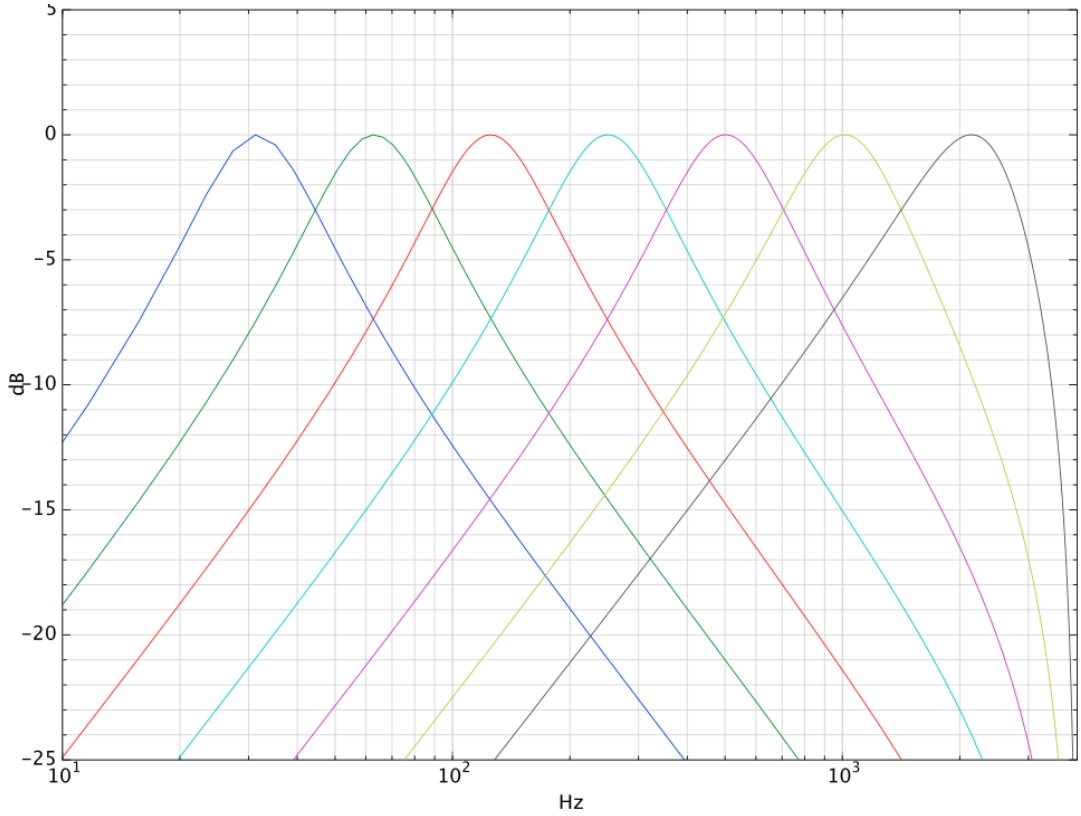

Figura 9: Módulo de la respuesta en frecuencia de los 7 filtros paso banda IIR de orden 2

Sin embargo el alumno deberá realizar el filtrado de la señal de entrada en el dominio temporal. La ecuación característica de estos filtros en dicho dominio es:

$$s_{out}[n] = G \cdot (b_0 \cdot s_{in}[n] + b_1 \cdot s_{in}[n-1] + b_2 \cdot s_{in}[n-2]) - a_1 \cdot s_{out}[n-1] - a_2 \cdot s_{out}[n-2]$$

La Tabla 1 detalla la ganancia y los coeficientes de cada filtro. Los valores de los coeficientes y ganancias están escalados con un factor 1024 (multiplicados) y truncados con el fin de poder trabajar en entero (trabajar en flotante haría inviable el procesado en tiempo real en el MCF5272, que no dispone de unidad aritmética en punto flotante). El alumno deberá tener en cuenta este escalado a la hora de implementar la rutina de filtrado, dividiendo por el mismo factor (realice la división mediante un desplazamiento).

[1] Esta normalización es necesaria para que el filtro sea estable. En la implementación que el alumno realice de cada filtro considere aplicar esta ganancia después de filtrar. Si incluye dicha ganancia des-normalizando los coeficientes el filtro podría saturar en puntos intermedios.

Tabla 1: Detalle de los valores de la frecuencia central (f_o), frecuencias de corte (fc_1, fc_2), ganancia (G) y coeficientes de cada uno de los 7 filtros considerados

BANDA	f_o	fc_1	fc_2	GANANCIA (G)	COEFICIENTES				
					b_0	a_1	a_2	b_1	b_2
0	31,25	22,10	44,19	8	1024	-2029	1006	0	-1024
1	62,50	44,19	88,39	17	1024	-2011	988	0	-1024
2	125	88,39	176,78	34	1024	-1970	955	0	-1024
3	250	176,78	353,55	66	1024	-1878	890	0	-1024
4	500	353,55	707,11	125	1024	-1660	772	0	-1024
5	1000	707,11	1414,21	227	1024	-1115	569	0	-1024
6	2000	1414,21	2828,43	392	1024	141	239	0	-1024

Una posible representación gráfica de esta relación entre s_{in} y s_{out} es la que se presenta en la Figura 10. La figura representa la realización de un filtro en lo que se conoce como forma directa II (mediante la transformación de la realización que representaría la ecuación característica anterior). En este caso, en lugar de deshacer el escalado cada vez que multiplicamos por uno de los coeficientes, se recomienda realizarlo después de cada sumatorio y tras la aplicación de la ganancia G. No deshacer el escalado de forma correcta traerá consigo el desbordamiento de las variables, complicando la depuración del programa.

Esta estructura tan sólo es una alternativa a las diversas variantes de realización de filtros digitales, pudiendo el alumno optar por otras alternativas. No obstante, dadas las exigencias de tiempo real y recursos del sistema, esta realización nos permite minimizar el número de búferes.

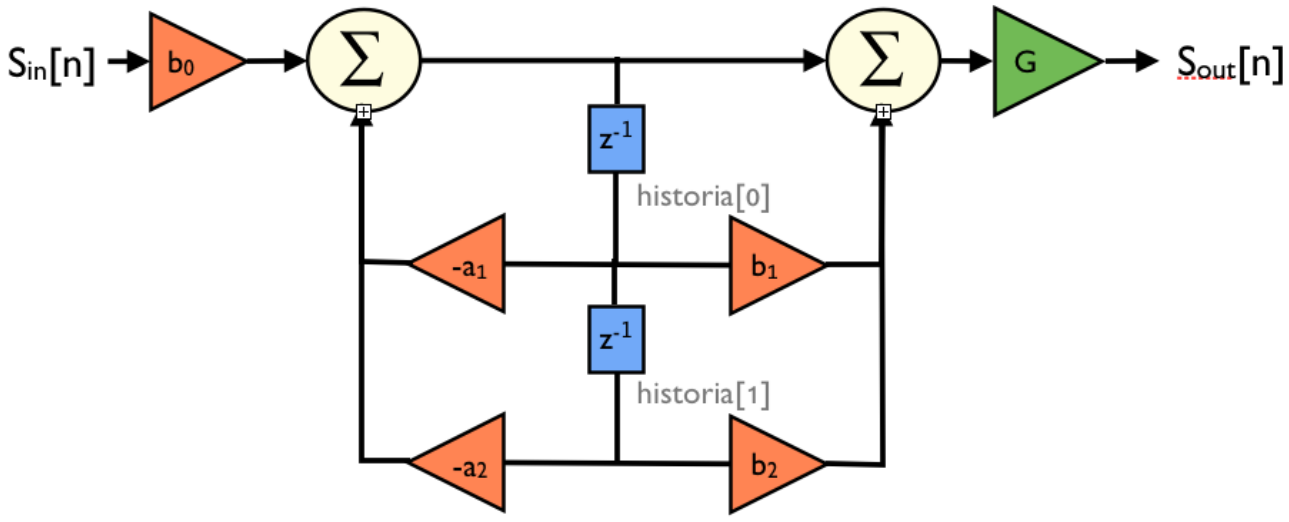

Figura 10: Realización de un filtro IIR de orden 2 mediante una estructura conocida como forma directa II

El objetivo de la práctica en ningún caso es profundizar en la teoría de filtros digitales, dejando al alumno la libertad de estimar teóricamente y optar por otros tipos de realizaciones a partir de las especificaciones de los parámetros de cada filtro especificados en la Tabla 1 (frecuencia central y frecuencias de corte inferior y superior).

4.4.2 Ecualización de audio

En la práctica implementaremos un ecualizador de octava empleando el conjunto de filtros descrito en el apartado anterior. La frecuencia central de cada filtro está a la distancia de una octava de sus adyacentes (lo que equivale a multiplicar o dividir por 2 cada frecuencia central). La Figura 11 muestra la respuesta global al emplear todo el banco de filtros tal y como muestra la Figura 8 asumiendo que la opción EQ está seleccionada.

El rizado en la banda de paso empleando este tipo de ecualizador, con los filtros presentados, es de 1,76 dB. Note que la ganancia global introducida por el banco de filtros es de 3dB tomando como referencia cualquiera de las frecuencias centrales de cada filtro. Compense esta ganancia en su sistema con el fin de obtener una señal $s_{out}[n]$ lo más parecida a la señal de entrada $s_{in}[n]$.

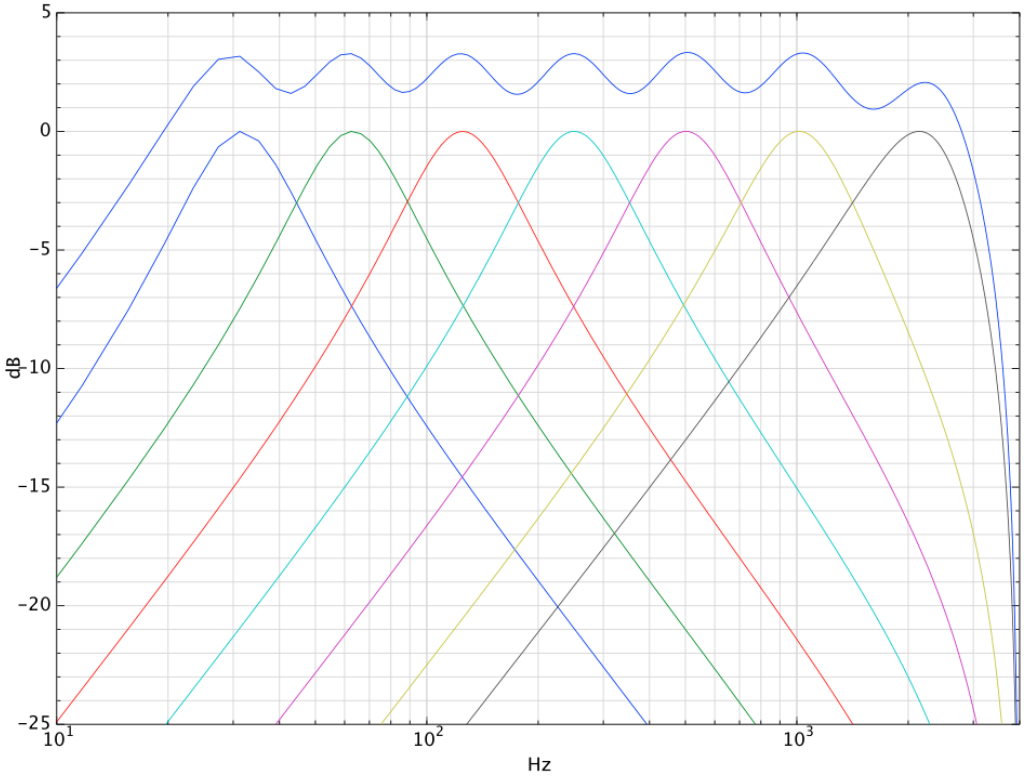

Figura 11: Módulo de la respuesta global del banco de filtros junto con el módulo de la respuesta en frecuencia de cada filtro por separado

El margen dinámico útil para disponer de un buena relación S/N aceptable hace que tan sólo podamos realzar cada banda algunos dB. Es por esto por lo que plantearemos la ecualización en términos de atenuación de unas bandas respecto a otras.

Para establecer las posibles atenuaciones a aplicar, necesitamos definir N niveles de energía (en nuestro caso se han definido 8 niveles, más el nivel por debajo del umbral de ruido, nivel 0). Siendo el nivel máximo cuando disponemos de una señal de amplitud 1,5Vp (señales con una amplitud mayor se ha comprobado que producen saturación a la salida del DAC). Como nivel mínimo consideraremos un umbral de ruido de 25mV. Los 8 niveles de energía se han establecido siguiendo una escala logarítmica (Tabla 2).

Tabla 2 Niveles de energía definidos para cubrir todo el margen dinámico de Energía de la señal de entrada Sin

NIVEL	Amplitud (V_{max}) de $S_{in}[n]$		Energía de $S_{in}[n]$	
	(Vp)	(valor en entero)	($V_{max}^2/2$)	log($Vmax^2/2$)
8	1,500	1228	753992	5,88
7	0,895	733	269354	5,43
6	0,535	438	96223	4,98
5	0,320	262	34374	4,54
4	0,191	156	12280	4,09
3	0,114	93	4386	3,64
2	0,067	55	1567	3,20
1	0,040	33	559	2,75
0	0,025	20	200	2,30

De la tabla anterior, podemos comprobar que la ganancia G_n entre niveles continuos de energía es 1,68. Con el fin de establecer mayor granularidad en la ecualización estableceremos una ganancia de $\sqrt{Gn}$, que nos permitirá disponer de 16 niveles de ecualización. Una alternativa de implementación del sistema de ecualización consiste en preestablecer una ganancia asociada al nivel de ecualización que el usuario seleccione mediante la interfaz de usuario. De esta forma podemos definir las ganancias asociadas a cada nivel de ecualización **(estas ganancias están multiplicadas por un factor 1024, tenga esto en cuenta a la hora de aplicarlas)**.

Tabla 3: Factores de ganancia a aplicar según el nivel de ecualización seleccionado por el usuario.

NIVEL	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Ganancia	1024	790	610	471	364	281	217	167	129	99	77	59	46	35	27	21

4.4.3 Sistema de visualización de energía en bandas: vúmetro

El alumno implementará un vúmetro (si bien su implementación será tan sólo aproximada, dado que no se ajusta estrictamente a los parámetros establecidos en la norma ANSI C16.5-1942) con el que visualizar la energía estimada en cada banda de frecuencias mediante el banco de filtros descrito en el apartado 4.4.1. Dicho vúmetro se implementará mediante hardware empleando la matriz de leds disponible en cada uno de los puestos del LSED y siguiendo el esquema que se muestra en la Figura 12. Emplearemos las 7 filas de la matriz de leds (una por cada banda de frecuencias) y 8 columnas (correspondientes a los 8 niveles de energía que detallábamos en Tabla 2). Si aplicamos 0 voltios a una fila y 5 a las demás, y si aplicamos 0 voltios a las columnas que no queramos iluminar, y 5 a las que deseemos iluminar, conseguiremos ir mostrando los niveles de energía de cada banda (distinto número de leds encendidos en cada fila, dependiendo del nivel de energía de cada banda).

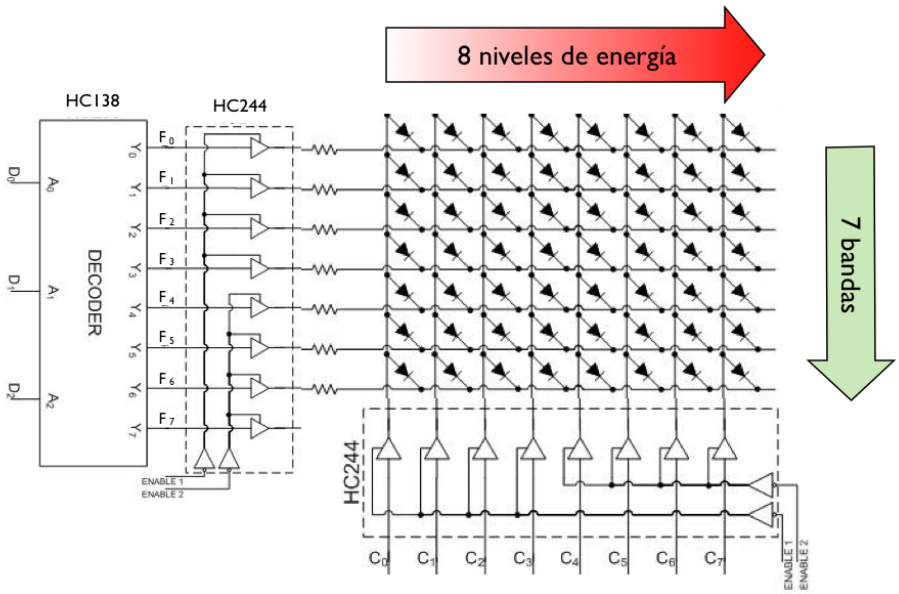

Figura 12: Detalle del Hardware asociado al vúmetro

El procedimiento para conseguir una visualización correcta del vúmetro consistirá en aplicar sucesivamente el barrido de excitaciones necesario para que cada fila de leds pueda encenderse y apagarse hasta aproximadamente 47 veces por segundo (i.e. 1 vez cada 3ms x 7 filas = 21ms), produciéndose la ilusión de que los leds de todas las filas están encendidos a la vez, cuando realmente en cada instante de tiempo, sólo están siendo excitados los leds de una determinada fila.

En particular, será necesario recorrer cíclicamente un autómata en anillo de 7 estados (7 es el número de filas de leds). Cada fila estará iluminada durante 3 ms (i.e. 24 interrupciones si configuramos un temporizador con una frecuencia de interrupción de 8 KHz) y a continuación se pasará a iluminar la siguiente fila. Una vez finalizado el periodo de excitación correspondiente a la última fila daremos comienzo nuevamente a un nuevo periodo de excitación de la primera.

Finalmente, el conjunto de leds de cada fila (número de columnas) que es preciso iluminar dependerá de la energía estimada (únicamente la energía de la banda asociada a la fila activa en cada periodo de excitación) en cada momento.

El alumno deberá implementar una rutina que emplee el puerto de salida para controlar la fila y aquellas columnas de la matriz de leds que se iluminan en cada instante al igual, sin modificar por ello los valores del puerto asociados al teclado matricial de la TL04.

4.4.4 Incorporación de efecto reverberación simple

La reverberación es un fenómeno acústico por el cual una onda se refleja y regresa a la fuente, con una determinada atenuación y fase, a través de caminos distintos (varias reflexiones). El efecto propuesto consiste en la simplificación de un reverberación. Implementaremos únicamente una reflexión, que de forma iterativa vuelve a ser reflejada una y otra vez (hasta que la atenuación la hace despreciable). Los parámetros que definen esta reverberación son su atenuación A y el retardo T. La Figura 13 muestra el esquema general del efecto propuesto.

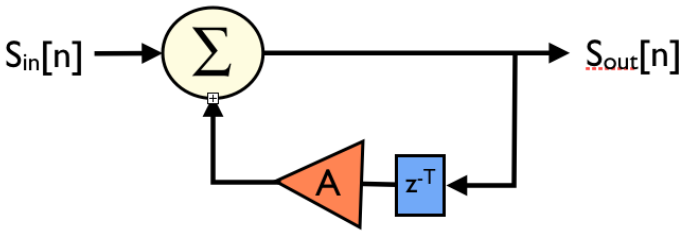

Figura 13: Esquema general de la reverberación simple

El alumno deberá implementar este efecto almacenando las muestras por medio de un búfer circular (empleando un tamaño de búfer de 1 segundo, por ejemplo).

5. Requisitos y consideraciones de implementación

5.1 ¿Cómo conecto el reproductor externo a mi circuito?

El reproductor de sonido que se empleará en esta práctica será por lo general estereofónico. Deberá conectar dicho reproductor mediante un cable Jack "macho"-"macho" equivalente al montado en LCEL. En este caso, basta con emplear uno de los dos canales (izquierdo o derecho). Para ello, en el jack de conexión del auricular, suelde dos cables: uno en el terminal de masa (GND) y otro en cualquiera de los otros dos. NO CORTOCIRCUITE los terminales L y R porque puede dañar el reproductor de sonido. NO UTILICE UN JACK MONOAURAL YA QUE CORTOCIRCUITA INTERNAMENTE AMBOS CANALES.

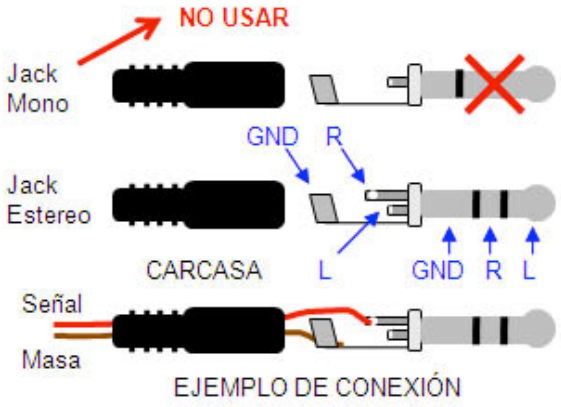

Figura 14:Especificaciones de montaje del conector de cable de conexión del reproductor de sonido

5.2 ¿Cómo escucho la señal de salida?

La reproducción de la señal procesada se realizará utilizando auriculares (**nunca altavoces**) y con un volumen de reproducción bajo, sólo audible para las personas que tienen puesto el auricular. El motivo es no incrementar innecesariamente el ruido del laboratorio, evitando molestar al resto de compañeros.

5.3 Estructura obligatoria de procesos

Frecuentemente, los sistemas electrónicos digitales precisan realizar diversas acciones de una manera paralela (concurrente en varios procesadores) o aparentemente paralela (ambas acciones se ejecutan entrelazadamente, produciéndose la apariencia de paralelismo si la frecuencia de conmutación entre ambas es elevada). Son los denominados procesos. La creación, ejecución, sincronización y destrucción entre procesos suele ser facilitada por el sistema operativo a través de diversas primitivas. Pero incluso en ese caso, el diseñador debe definir qué tareas o rutinas serán concurrentes, elegir los mecanismos de comunicación entre ellas, los tiempos de vida, etc.

En el LSED no disponemos de un gran sistema operativo en nuestra placa de desarrollo, pero podemos definir varios procesos por medio de interrupciones periódicas, interrupciones externas y el programa principal.

En la Figura 15 se muestra el esquema de procesos propuesto para esta práctica. El sistema implementado por el alumno comprenderá 2 procesos: el programa principal y una interrupción periódica. **Este esquema de procesos es obligatorio.**

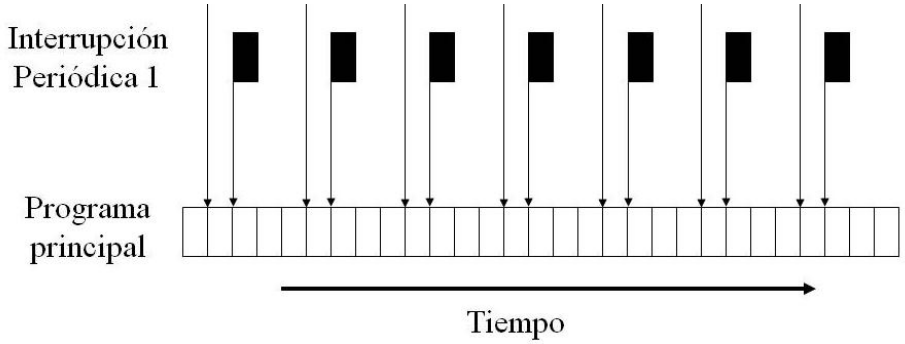

Figura 15: Representación temporal de los 2 procesos

Las funciones asociadas a un mismo objeto no se ejecutan necesariamente en un mismo proceso. Sobre los mecanismos de sincronización en el LSED, recomendamos la lectura del apartado 4.1.6 "Desarrollo de software de tiempo real" de [5]. Como en nuestro caso las interrupciones periódicas pueden interrumpir la ejecución del programa principal en instantes de tiempo que resultan imprevisibles cuando se escribe el código, si el programa principal necesita modificar un objeto compartido, procederá a **deshabilitar las interrupciones**, realizar la modificación y volver a **habilitar las interrupciones.** Todo este conjunto de operaciones puede encapsularse en el objeto puerto. Como el programa principal no puede interrumpir a las propias interrupciones, estas últimas no necesitan realizar ninguna operación especial. Como las TRAP son excepciones software, se ejecutan en instantes de tiempo predecibles y se puede evitar que interrupciones y programa principal las usen a la vez.

En cualquier caso, **el nivel de las interrupciones temporizadas debe ser inferior a 7**, porque el nivel 7 no es enmascarable y dificulta mucho la depuración (la ejecución de la rutina de atención a la interrupción puede ser a su vez interrumpida por una nueva petición de interrupción).

Si el nivel es 7, aunque pongamos un punto de parada en la rutina de atención a la interrupción y el programa aparentemente se detenga al llegar a él, la generación y atención de interrupciones no se detiene y nuestro programa sigue ejecutándose de una manera indeseada y difícil de predecir.

5.3.1 Proceso principal

El proceso principal siempre existe y debe: inicializar los objetos del sistema, inicializar el HW, habilitar la aparición de interrupciones e inicializar ambos procesos de interrupción periódica (*_init*). Los sistemas empotrados de tiempo real suelen contener un bucle indefinido (*bucleMain*) que puede llevar a cabo tareas sin requisitos temporales estrictos, como puede ser en nuestro caso la lectura del teclado y la presentación en pantalla de los menús. En nuestro caso el bucle del programa principal podría ser aproximadamente:

```
BucleMain:
  while (true)
    switch (estado)
      case INICIO:
       GestionMenuPrintipal();
```

```
        break;
      case CARACTERIZACION:
        GestionCaracterizacion ();
        break;
      case ECUALIZACION:
        GestionEcualizacion ();
        break;
      case REVERB:
        GestionReverb ();
        break;
       default:
         _printf ("\n*** Opción elegida errónea. Vuelva a intentarlo. ***\n\n");
         estado = INICIO;
         break;
endwhile
```

En cada una de las opciones del menú se gestionará el valor de las variables relacionadas con cada una de las funciones disponibles, como por ejemplo:

- filtro seleccionado para caracterizar
- niveles de ecualización en cada banda
- reverberación activa o inactiva
- valor de atenuación y retardo de la reverberación
- etc

La rutina general de inicialización sería similar a:

```
Init:
  swInit()        * inicializa todos los objetos/variables SW
  hwInit()        * inicializa las interrupciones y los puertos
```

Este proceso principal es el menos prioritario de los tres porque el teclado o la escritura en la pantalla no son acciones tan críticas en el tiempo.

5.3.2 Interrupción periódica

La interrupción periódica contendrá las funciones críticas en el tiempo relacionadas con el refresco de los leds y el procesamiento síncrono en muestra (frecuencia de interrupción igual a 8kHz) de la señal analógica de entrada. Por esta razón, **este proceso debe tener mayor prioridad que el programa principal**. Su ejecución es cíclica, discontinua y sucede a intervalos regulares de tiempo; es concurrente con el proceso principal.

5.4 ¿Cómo leo una señal bipolar empleando el ADC?

Las rutinas proporcionadas en el tutorial para el manejo del ADC están configuradas para leer datos de señales unipolares entre 0 y 5V.

Será necesario modificar dichas rutinas para que las muestras de tensión negativa sean leídas correctamente. Para ello será necesario realizar una extensión de signo (de 12 a 32 bits), dado que el dato leído se encuentra en complemento a dos de 12 bits, pero las operaciones aritméticas las haremos en 32 bits con signo.

En el caso de que el número de 12 bits sea positivo la extensión se realizaría rellenando con 0's los 20 bits más significativos. En caso de que el número de 12 bits sea negativo, la extensión se realizaría rellenando con 1's los 20 bits más significativos.

5.5 ¿Cómo convierto una señal bipolar en una señal unipolar del mismo número de bits para sacarla por el DAC?

La señal de entrada $S_{in}[n]$ es una señal bipolar de media 0. Será necesario adaptar el rango de esta señal a los valores admitidos por el DAC (valores sin signo entre 0 y 2^{12}-1), para lo cual será necesario añadirle a la señal la mitad del fondo de escala (2^{11}, 0x800 en hexadecimal).

5.6 ¿Cómo se estima la energía en una banda de frecuencias a partir de N muestras?

La energía E_{Bi} en una banda i de la señal $S_{Bi}[n]$ ($S_{in}[n]$ tras el filtro B_i), estimada dentro de una cierta ventana de N muestras, se define como:

$$E_{Bi} = \frac{1}{N} \sum_{n=0}^{n=N-1} s_{Bi}[n]^2$$

El alumno deberá estimar la energía asociada a la banda correspondiente en cada periodo de excitación del vúmetro, coincidiendo N con el número de interrupciones que corresponden a cada periodo de excitación (N =24, 3ms empleando una frecuencia de muestreo de 8 kHz). Deberá ir acumulando el valor de la energía de la muestra procesada en cada interrupción en una variable.

En la práctica, en lugar de dividir por N en el cálculo de E_{Bi}, multiplique por dicho factor N los valores de los umbrales que el alumno puede deducir de la Tabla 2 (los cuales serán necesarios para mostrar el nivel de energía a través del vúmetro). De esta forma se evita tener que realizar una operación de división en cada periodo de excitación.

5.7 ¿Cómo modifico uno o varios bits del puerto de salida?

En esta práctica, ambos procesos deben usar el puerto de salida: el programa principal y la primera interrupción periódica encargada del refresco de los leds (aunque usan bits diferentes de dicho puerto). Como los dos procesos se ejecutan aparentemente en paralelo, si no se tiene cuidado, puede haber conflictos entre ellos, llegando al extremo de que un proceso modifique sin querer los bits o los datos correspondientes a otro.

Tabla 4: Detalle de las diferentes conexiones al puerto de salida

Puerto de salida	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Conexión	C7	C6	C5	C4	C3	C2	C1	C0	-	D2	D1	D0	T3	T2	T1	T0

En los tutoriales de C se emplea la rutina set16_puertoS(), para enviar un valor de 16 bits al puerto de salida (la rutina equivalente en ensamblador hará aproximadamente un MOVE.W del dato de 16 bits a la dirección del puerto). Sin embargo, como los puertos son unidireccionales, el puerto de salida no puede ser leído. Esto implica que en ensamblador no se puede modificar correctamente un bit del puerto de salida directamente haciendo un BSET, BCLR o BCHG de un bit del puerto, porque estas instrucciones implican leer y escribir en el puerto. En C, no se puede aplicar una

máscara AND (operador &) u OR (operador |) directamente al puerto para modificar un bit. En ambos casos es necesario emplear una variable auxiliar (lo que denominamos objeto Puerto).

Supongamos que los 4 bits menos significativos (0-3) se emplean para excitar las columnas T0-T3 del teclado (como se explica en el tutorial del mismo), los bits 4-6 para excitar las filas F0- F6 de la matriz de leds (recuerde que emplearemos un decoder de 3 a 8 líneas, por lo que realmente lo que codificamos con esos 3 bits, D0-D2, es la entrada de dicho decoder), y que se emplean los bits 8-15 del puerto para excitar las columnas C0-C7 de la matriz.

Cada vez que el teclado quisiese excitar una de sus columnas (si se realizase el barrido), debería poner un 1 en el bit correspondiente a esa columna del teclado, y un 0 en los otros 3 bits relacionados con el teclado, respetando el valor que tuviesen los bits relativos a los leds. Si, por ejemplo, se quiere excitar la columna 0 del teclado (T0), y se envía un 0x0001 al puerto, se comete un error: además de excitar la columna adecuada, se están poniendo a 0 todos los bits que tienen que ver con los leds, con la posible alteración del buen funcionamiento de la parte de visualización.

Para poner a 1 el bit de una columna del teclado (T0-T3), lo recomendado en ensamblador sería realizar el BSET, BCLR o BCHG, sobre un registro que contiene el valor de la variable auxiliar Puerto_variable (que contiene el último valor enviado al puerto), para a continuación actualizar la variable auxiliar y enviar esa variable modificada al puerto. Tenga en cuenta que el puerto (y por lo tanto la variable auxiliar) es de 16 bits, pero las funciones BSET, BCLR o BCHG, no pueden modificar directamente una variable .W en una sola instrucción:

```
PUERTO_S EQU $40000002 * DIRECCION DEL PUERTO S

Puerto_variable DC.W 0
* esta rutina recibe a través de D0 el número del bit que poner a 1
PUERTO_SET_BIT: MOVE.L D1,-(A7)
MOVE.W Puerto_variable,D1
BSET.L D0,D1
MOVE.W D1,Puerto_variable
MOVE.W D1,PUERTO_S
MOVE.L (A7)+,D1
RTS
```

Para poner a 1 el bit de una columna del teclado (T0-T3), lo recomendado en C sería:

```
#define MASCARA_TECLADO 0xFFF0 // asigna 0 a cada bit del teclado
#define EXCIT_TECLADO 0x0001

typedef struct {
WORD variableAux; // variable básica del objeto puerto
...
}
 TpuertoSalida;

TpuertoSalida puerto;
  // numColumna es la columna del teclado que excitar: 0, 1, 2 o 3

void puerto_excitaColumnaTeclado(int numColumna)
 {
 ...
  // falta comprobar que numColumna<4
 static WORD excitaTeclado[]={0x1,0x2,0x4,0x8};
  // suponemos que variableAux contiene el último dato enviado al puerto
  // borra sólo los bits del teclado
```

```
    puerto.variableAux= puerto.variableAux & MASCARA_TECLADO;
     // pone a 1 el bit adecuado en función de numColumna

    puerto.variableAux=puerto.variableAux|excitaTeclado[numColumna];
    set16_puertoS(puerto.variableAux); // envía el nuevo valor al puerto ...
    }
```

El alumno deberá implementar la rutina o rutinas necesarias para escribir en el puerto de salida los valores de los bits relacionados con las columnas (C0-C7) y la fila (D0-D2) que se desean iluminar, sin por ello modificar los valores de los bits relativos al teclado (T0-T3).

5.8 ¿Qué es un búfer circular?

Un búfer circular es una estructura de datos en la que la forma en la que se introducen y sacan los datos hace que conceptualmente sus extremos estén conectados. De tal forma que cuando la posición de escritura o lectura llegan al final del búfer éstas vuelven al inicio de éste. Un ejemplo del pseudocódigo de un posible búfer circular sería:

```
int BufferCircular[MAX_LONGITUD];
int *UltimaMuestra = BufferCircular;
int DatoLeido;

BUCLE INFINITO
{
   DatoLeido = LeeDato(ADC);
   *UltimaMuestra = DatoLeido;

   UltimaMuestra++;

   if (UltimaMuestra >= (BufferCircular + sizeof(BufferCircular)))
      Ultimuestra = BufferCircular;
}
```

5.9 Observaciones adicionales sobre el HW de entrada/salida

Es necesario recordar que los puertos de entrada y salida disponibles en la plataforma ENT2004CF están formados por buffers digitales con resistencias de pull-down de 10 KΩ. Si no se tiene en cuenta este hecho, se podrían producir efectos de carga no deseados al conectar el HW a la plataforma.

Igualmente importante es el hecho de conocer y considerar la impedancia de salida del subsistema conversor, cuya señal de salida es entregada mediante una resistencia de 100Ω al terminal BNC hembra (fundamentalmente para aumentar la protección ante posibles errores de utilización de la plataforma).

5.10 Alimentación de los subsistemas

Los subsistemas se alimentarán con tensiones simétricas o asimétricas, según convenga. Es muy importante desacoplar las alimentaciones y alimentar en estrella; remitimos al alumno a la referencia bibliográfica [5].

Si algún elemento puede demandar picos fuertes de corriente (por ejemplo, un amplificador de potencia), puede ser muy importante filtrar de manera extraordinaria su alimentación para que estos picos no afecten al resto de la circuitería (por ejemplo, un filtro de audio).

5.10.1 Optimización

Aunque la optimización del código (en cuanto a consumo de memoria o de tiempo de CPU) no se puede llevar a cabo hasta la fase de implementación, un mal diseño o una mala arquitectura pueden hacer imposible esta labor en la fase de implementación, obligando a una reconsideración del diseño. Así, si nuestra arquitectura intentase actualizar el visualizador o generar las muestras de audio desde el programa principal, los requisitos de temporización de estas tareas serían muy difíciles de cumplir y se podría producir un funcionamiento incorrecto: al tratarse de un defecto estructural, cualquier optimización puede estar condenada al fracaso si no revisamos la arquitectura.

Aunque éste no es nuestro caso (y la arquitectura propuesta puede satisfacer los requisitos del enunciado), dado que en el LSED las prácticas son de tiempo real, no es extraño que el alumno se vea obligado a emplear una o varias de estas estrategias de optimización para conseguir solucionar el problema planteado por el enunciado.

5.10.1.1 Optimización de pequeñas rutinas en C

Si una rutina es pequeña se pierde más tiempo en el proceso de entrar y salir de la rutina, que en ejecutar la rutina propiamente dicha. En estos casos puede ser interesante definirla como *inline y activar un nivel de optimización del compilador distinto de 0*, de tal manera que el compilador no genera una verdadera rutina y una verdadera llamada a rutina (con paso de datos a través de registros o de la pila), sino que simplemente inserta el código equivalente en el lugar donde se llamaría a la rutina, con el consiguiente ahorro de tiempo de ejecución. Por ejemplo:

```
// declaración del tipo (o clase)
typedef struct {
int BufferCircular[MAX_LONGITUD];
int *UltimaMuestra;
...
} TBufferCircular;

TBufferCircular bufferCircular; // declaración del objeto (o variable)

// rutina que deseamos ejecutar rápidamente
inline int buffer_escribir(TBufferCircular *pBC, int dato)
{
*(pBC->UltimaMuestra) = dato;

(pBC->UltimaMuestra)++;

if (pBC->UltimaMuestra >= (&(pBC->BufferCircular)+sizeof(pBC>BufferCircular))
pBC->UltimaMuestra= &(pBC->BufferCircular);
}

int buffer_init(TBufferCircular *pBC)
{
pBC->UltimaMuestra = &(pBC->BufferCircular);
}

void __init()
{
...
buffer_init(&bufferCircular)
...
}
```

```
  void otra_rutina()
  {
  ...
  {
  // un bloque entre llaves permite definir nuevas variables locales a ese
bloque, como DatoLeido
  int DatoLeido = LeeDato(ADC);
  buffer_escribir(&bufferCircular, DatoLeido);
  }
  ...
  }
```

El código optimizado generado por el compilador será equivalente a haber escrito directamente:

```
  // código equivalente "generado" por el compilador
  TBufferCircular bufferCircular; // declaración del objeto

  void __init()
  {
  ...
  buffer_init(&bufferCircular)
  ...
  }

  void otra_rutina()
  {
  ...
  {
  int DatoLeido = LeeDato(ADC);
  bufferCircular.UltimaMuestra++;

  if (bufferCircular.UltimaMuestra >=
(&(bufferCircular.BufferCircular)+sizeof(bufferCircular.BufferCircular))
  bufferCircular.UltimaMuestra= &(bufferCircular.BufferCircular);
  }
  ...
  }
```

A lo largo de la práctica seguramente será necesario activar la optimización del compilador a nivel 2 ó 3. Se deberá considerar esta opción especialmente cuando se estén aplicando todo el banco de filtros, gestionando el teclado y el vúmetro, momento en el que el sistema necesita la mayor potencia de proceso.

Debido a la necesidad de tiempo de procesamiento, será necesario reducir en la medida de lo posible el número de operaciones que realizará el sistema; por ello, se recomienda substituir las divisiones por números equivalentes a potencias de 2, en desplazamientos del número equivalente de bits.

6. Descripción de la sesión -1 (antes de ir al laboratorio B-043)

Aunque al principio le resulte tedioso, recomendamos al alumno que se tome su tiempo para formarse con los libros de apoyo al laboratorio publicados. Concretando algo más, debería leer, antes de empezar las sesiones del Laboratorio, al menos:

- El tema 6 (de temporizadores) del libro de SEDG [9]. Prestar especial atención a la generación de interrupciones en tiempo real.
- El libro de tutoriales de manejo de la plataforma ENT2004CF [3]. Como mínimo los tutoriales relacionados directamente con la práctica (temas 1, 2, 3, 4, 5, 7 y 8).
- El tutorial de utilización y la ayuda para el entorno de desarrollo EDColdFire v3.0 [4], en especial el tema 3.
- El capítulo 4 de [5]: con el epígrafe 4.1 y 4.2 de dicho manual le bastará.
- El breve manual del Osciloscopio HAMEG HM-407 [6], en el caso de no haberlo manejado anteriormente.
- Los documentos de Dudas y Problemas Frecuentes en ensamblador y en C [7][8].

También debería bajar de Internet el software EDColdFire (que incluye el ensamblador del MCF5272, y también el compilador GCC integrado para desarrollar la práctica en lenguaje C), y aprender a manejarlos en su casa. Estos programas, el manual del osciloscopio y el documento de dudas se encuentran disponibles a través del portal del LSED (http://lsed.die.upm.es/).

Con el fin de aprovechar al máximo las sesiones en el Laboratorio, que son un recurso muy limitado, el alumno debe traerse los programas ya escritos de casa, donde los habrá ensamblado con EDColdFire para comprobar que el código no tiene errores de sintaxis. Durante la sesión de Laboratorio, el alumno volverá a ensamblar el programa y lo ejecutará en la plataforma ENT2004CF. Encontrará errores que ya no son de sintaxis, sino de ejecución, que corregirá en el Laboratorio: la labor de depuración es fundamental.

No terminaremos este apartado sin mencionar algo obvio: el alumno debería conocer la arquitectura y el juego de instrucciones del procesador MCF5272. Recomendamos que para ello emplee la bibliografía de apoyo a la asignatura Sistemas Electrónicos Digitales como [9].

6.1 Comentarios sobre las conexiones

En relación a las diferentes conexiones con el puerto de salida deben tenerse en cuenta los siguientes aspectos:

- En el tutorial de utilización del EDColdFire [4] y en la ayuda del programa está dibujado el patillaje de los puertos; es decir, qué significa cada terminal de la plataforma (el dibujo está hecho tal como se ve el conector de la plataforma desde fuera, no hay que imaginárselo al revés o en espejo).

- En el tutorial de la placa auxiliar del teclado y del LCD (TL04) (tema 7 de [3]) están igualmente disponibles los 16 terminales de entrada y los 16 de salida. Éstos están también disponibles en la web de la asignatura [18].

- En la placa auxiliar TL04, las entradas y salidas están numeradas de la misma manera que en los esquemas de entradas y salidas disponibles en web y en [3]. Uno de los conectores negros hembra de la placa TL04 está conectado a los pines 1-13 (**ambas filas del mismo conector negro están redundantemente conectadas a los mismos pines**) y el otro está conectado a los pines 14-25, también de manera redundante (el 26 del conector negro no está conectado).

- En el futuro va a ser muy importante que la masa de la plataforma (disponible en los conectores DB25 y en la placa auxiliar) esté conectada a la masa de la fuente de alimentación (esto es, **la masa debe ser única**). Todos los terminales que se indican como masa en los conectores de la plataforma son el mismo punto (sólo necesitamos conectar uno de ellos).

- Para las conexiones se deben usar tiras de 8x1 y 4x1 pines macho-macho planos, sin tornear, largos (25 pines), de 2,54 mm de paso. En la web de la asignatura se han publicado fotos sobre cómo conectar el HW de la práctica a la placa TL04 [19]. Estos sencillos conectores macho debéis soldarlos a vuestros cables y facilitarán las conexiones que hay que realizar cuando se llega al B-043 y que desconectar al irse. **Conectar cables rígidos directamente a los conectores negros está totalmente desaconsejado, porque la conexión no es buena y provocará numerosos problemas.**

7. Descripción detallada de la sesión 0

Para comenzar a abordar la Práctica en sí, el alumno tendrá dos primeras sesiones de Laboratorio introductorias en las cuales se familiarizará con el entorno de desarrollo Hardware (la plataforma ENT2004CF, aunque por motivos de seguridad, el alumno no conecta directamente su sistema al MCF5272, sino a una tarjeta de interfaz compuesta de un primer buffer digital, un conjunto de leds, una etapa de optoacopladores, y un segundo buffer digital), desarrollará el hito 1 y terminará de adquirir habilidad con el entorno de desarrollo EDColdFire (que, como bien sabe, debe haber usado previamente antes de ir al laboratorio B-043). Las partes de que se compone esta primera sesión de laboratorio son:

- Realizar un pequeño **tutorial inicial** descrito en el capítulo 3 de [3].

- Realizar el **tutorial de manejo del teclado del capítulo 7 [3]**. Con este tutorial se pretende que el alumno conozca y aprenda el manejo del teclado disponible en la placa TL04 que utilizaremos como subsistema hardware de entrada de datos. De las 3 partes de las que se compone el tutorial de manejo de la placa TL04 sólo hay que realizar el tutorial de manejo del teclado, que permite imprimir en la pantalla del ordenador.

- Ejecutar, depurar y modificar un **primer programa**; este programa debe pedir al usuario que seleccione un tipo de funcionalidad (caracterización de filtros, ecualización o incorporación de reverberación simple) pulsando la tecla asignada correspondiente (i.e. 1, 2 o 3), y el programa debe generar un mensaje en pantalla informando de la opción seleccionada. A continuación se solicitará al usuario los parámetros necesarios

dependiendo de la funcionalidad previamente seleccionada. Si la tecla pulsada fuese distinta de las válidas en ese momento del diálogo con el usuario, deberá presentarse un mensaje de error por pantalla advirtiendo al usuario de que la tecla que debe pulsada es errónea, recordándole las opciones posibles. Veamos un ejemplo parcial de funcionamiento del programa.

Sistema: "Seleccione una de las siguientes opciones:"
" 1) Caracterización de filtros"
" 2) Ecualización Gráfica"
" 3) Incorporación de Reverberación Simple"

Usuario: Pulsa 1.

Sistema: "CARACTERIZACIÓN DE FILTROS"

Sistema: " Seleccione el filtro que quiere caracterizar (1-7) o 'A' para volver al menú de inicio:"

Usuario: Pulsa 8.

Sistema: "Tecla no válida. Por favor, pulse de 1 a 7 para seleccionar…"

En el caso de que seleccionemos la funcionalidad "ecualización", cada vez que ajustemos el nivel de una banda de frecuencias, será necesario mostrar por pantalla el nivel de atenuación de todas las bandas, a fin de poder deducir la respuesta en frecuencia del ecualizador. El nivel de la banda seleccionada se podrá subir o bajar pulsando las teclas '8' y '9' respectivamente. Un ejemplo aproximado de visualización sería:

```
BANDA:    32 Hz  64 Hz  125 Hz  250 Hz  500 Hz  1kHz  2kHz
Ganancia  1024   1024    471     281     610    790   1024
(Nivel):   (0)    (0)    (3)     (6)     (2)    (1)   (0)
```

7.1 Método de trabajo

En el PC de laboratorio, deberá crear un directorio de trabajo colgando de la carpeta "c:\alumnos" (carpeta para la que dispondrá de permisos de escritura). Ponga a ese directorio el nombre de su código de pareja de laboratorio (por ejemplo, MT23). Dentro de él vaya creando un directorio por cada sesión de acuerdo con el plan propuesto en el enunciado (sesion_1, sesion_2, etc.).

Asegúrese de grabar los programas modificados en una memoria FLASH, o enviarlos por email al final de la sesión de trabajo. **Para evitar problemas de copias no deseadas de su SW por parte de otros alumnos, al final de cada sesión debe borrar los ficheros .ASM, .C o .H que constituyan su práctica.** Recuerde que se realizarán entregas electrónicas del SW y habrá verificación de plagios. Debe guardar al menos 2 copias en algún soporte (correo electrónico, memoria flash...).

Recuerde que el contenido de los discos duros puede ser borrado como consecuencia de las tareas periódicas de mantenimiento del laboratorio.

Trabajar directamente en un dispositivo de tipo pendrive con el EDColdFire en el B-043 puede dar problemas de compilación en algunos casos. Si sucede, se recomienda copiar los ficheros al disco duro (carpeta "c:\alumnos"), trabajar con ellos, y al acabar la sesión, copiarlos en el pendrive y borrarlos del disco duro. En todo caso, usar directamente este tipo de dispositivos ralentiza el proceso de compilación y ejecución.

El modo correcto de abrir el programa EDColdFire es desde el menú de Inicio / Todos los programas, o desde el icono que se crea en el escritorio al instalar.

7.2 Tutorial inicial

Viene descrito en el capítulo 3 de [4]. Fuera del laboratorio B-043 puede hacer algunos de los pasos que figuran en dicho libro.

Como parte de este tutorial inicial se recomienda también la ejecución del programa **Cochefantastico.c**.

En el laboratorio, el código fuente original de Cadena.c y CocheFantástico.c no se puede modificar, está protegido contra escritura. Para trabajar con él, haga una copia en su directorio de trabajo. Todos los tutoriales se encuentran en el directorio c:\archivos de programa\DIE-UPM\EDColdFire\tutoriales...

7.3 Tutorial del manejo del teclado de la placa TL04

Este tutorial viene descrito en el capítulo 7 de [3]. Con este tutorial se pretende que el alumno conozca y aprenda el manejo del teclado disponible en la placa TL04 que utilizaremos como subsistema hardware de entrada de datos. De las 3 partes de las que se compone el tutorial de manejo de la placa TL04 sólo hay que realizar el tutorial de manejo del teclado, que implementa la exploración del teclado y presenta en la pantalla del ordenador la tecla pulsada.

7.4 Diseño, implementación y prueba de un primer programa

En la primera sesión el alumno debe completar su primer programa que permite elegir uno de los 3 funcionalidades y configurar cada uno de los parámetros de la funcionalidad deseada. Para ello debe seguir la estructura que se comenta en los siguientes apartados.

7.4.1 Estructura básica de los programas que desarrolle

A continuación incluimos el esqueleto del programa a desarrollar. Éste contiene las 5 partes de las que se habla en el apartado 4.1.2 de [5] y en [3]. Además, la nomenclatura que se sigue para nombrar las variables y subrutinas es la misma que la comentada en dicho manual. Esa estructura es independiente de que la codificación se realice en ensamblador o lenguaje C.

Con el símbolo <...> se representa algo que puede bien estar vacío, o ser una instrucción o un grupo de instrucciones que han sido omitidas.

```
************************************************
*   ZONA DE CONSTANTES
    <...>
    ORG <valor>

************************************************
*   ZONA DE VARIABLES GLOBALES
    <...>
sinusoide DC.W …

************************************************
```

```
*   PROGRAMA PRINCIPAL
*************************************************
PPAL      <...>             * Modo supervisor. Inhabilita interrupciones
          <...>             * Mueve el puntero de pila a una región válida
          BSR swInit        * Subrutina configuración completa del SW
          BSR hwInit        * Subrutina configuración completa del HW
          <...>             * Habilita las interrupciones CPU

BUCLE     BSR bucleMain     * Bucle de programa principal.
          BRA BUCLE
    END   PPAL


***********************************************
*   ZONA DE SUBRUTINAS
    <...>
* Subrutina de configuración completa del hardware
hwInit:   <...>
          RTS

* Subrutina de inicialización de las variables que lo necesiten
swInit:   <...>
          RTS

* Subrutina que contiene el programa principal
* cuya ejecución es típicamente periódica en el LSED
bucleMain:     <...>
          BRA FIN_MAIN
ERROR:    <...>
FIN_MAIN
          RTS

***************************************************
*   ZONA DE RUTINAS DE ATENCIÓN A LAS INTERRUPCIONES
***************************************************
* Rutina de atención a las interrupciones en el caso de
* que sean necesarias
INTERR:   <...>
FININTER  RTE
```

7.4.2 Subrutina para configurar el HW (hwInit)

En esta subrutina se deben incluir las acciones necesarias para configurar cada uno de los recursos hardware que se vayan a utilizar: interrupciones, puertos,... Hay que prestar especial atención a la configuración de recursos que comparten los mismos registros de configuración: por ejemplo las interrupciones periódicas y las interrupciones externas. Sea muy cuidadoso en la inicialización de estos registros.

En relación con la frecuencia de las interrupciones periódicas se recomienda la configuración de estas interrupciones a una frecuencia de, por ejemplo, 8 KHz, a partir de la cual se puedan obtener fácilmente frecuencias menores para implementar los diferentes relojes que se necesitan a lo largo de la práctica.

7.4.3 Si tiene dudas o problemas…

Se recomienda al alumno que, antes de empezar a trabajar, lea el documento de *Dudas y Problemas Frecuentes del LSED*, o su versión para programación en C.

7.5 Nociones sobre depuración de programas

En este apartado se comentan algunas nociones interesantes para la depuración de programas. La depuración consiste en la búsqueda de errores durante el funcionamiento del programa, no durante la fase de ensamblado o compilación. En relación con los errores vamos a considerar cuatro tipos característicos:

- **NO sistemáticos y ocasionales**: son errores que no siguen una sistemática a la hora de producirse, lo hacen de manera ocasional. Estos errores son los más complicados de detectar puesto que son difíciles de reproducir.

- **NO sistemáticos pero frecuentes:** aunque no siguen una sistemática, al ser frecuentes permiten reproducirlos fácilmente y definir un proceso de búsqueda.

- **Sistemáticos pero no frecuentes:** en este caso al producirse bajo unas mismas condiciones, son errores fácilmente reproducibles y por tanto depurables.

- **Sistemáticos y periódicos:** se producen de forma periódica y en la mayoría de los casos tienen que ver con el funcionamiento de las interrupciones periódicas.

En los apartados siguiente describiremos un protocolo (o conjunto de pautas a seguir) de depuración o protocolo de búsqueda de errores aplicables en el caso de tener algún error **de los últimos 3 tipos.** Para el caso de errores no sistemáticos y ocasionales se ofrecerán algunos comentarios o pautas sobre las acciones de depuración que se pueden hacer para localizar el error.

7.5.1 PASO 1: Obtención de una prueba fallida y previsión de la salida

En este primer paso se trata de verificar los casos de uso propuestos en el enunciado y obtener un conjunto de pruebas lo más concretas posibles para las cuales el sistema no se está comportando correctamente (según las especificaciones). Si el número de pruebas asociado a un caso de uso es pequeño (ej: pulsaciones sobre todas las teclas) se pueden realizar todas las pruebas (ej: pulsar todas las teclas y analizar comportamientos similares: entre teclas de la misma fila o columna por ejemplo). La información del tipo de error que producen unas u otras pruebas nos puede dar información muy valiosa a aplicar posteriormente en el segundo paso: localización del error.

Una prueba fallida está compuesta de dos elementos imprescindibles:

Un conjunto de **entradas** o acciones concretas (ej: pulsa la tecla C) que originan o producen una salida o un comportamiento erróneo.

Realizar una **previsión** de cómo se debería comportar el sistema ante esas entradas o acciones (considerando el comportamiento correcto). Esta fase es la más laboriosa porque la previsión se debe realizar a bajo nivel (a nivel de registro, variable o cable), pero es imprescindible para garantizar el éxito en la localización posterior del error.

7.5.2 PASO 2: Localización del error

Con la prueba fallida obtenida en el paso anterior, ejecutamos el programa (o circuito) y observamos algún efecto de salida del error (algún comportamiento erróneo). Un comportamiento erróneo se produce cuando hay un desajuste en ese punto del programa (o circuito) entre la previsión realizada y el valor obtenido o medido en una variable, registro o señal.

Una vez detectado el comportamiento erróneo disponemos de dos puntos principales para comenzar el proceso de localización del error (búsqueda):

- **El punto de inicio**: comienzo del programa, inicio de las señales de entrada,...
- **El punto final:** lugar donde se detectó un desajuste entre la previsión para esa prueba y los valores reales de comportamiento.

El proceso de búsqueda puede comenzar desde el punto de inicio (Forwardtracking) o desde el punto final (Backtracking). En ambos casos es necesario:

- **Analizar las entradas y salidas de cada módulo**. En el caso de "forwardtracking" si tanto las entradas como las salidas están bien (coinciden con la previsión) pasamos al siguiente módulo. En el caso de "backtracking" si tanto las entradas como las salidas están mal (no coinciden con la previsión) pasamos a analizar el módulo anterior.

Cuando se detecta el módulo que está produciendo el error se comienza a analizar las diferentes partes del módulo utilizando las mismas estrategias de "forwardtracking" o "backtracking" en cada una de las partes hasta localizar una unidad atómica (indivisible, por ejemplo **instrucción** en programación o integrado en HW) donde las entradas están bien y las salidas mal. En este punto hemos localizado el error.

7.5.2.1 Comentarios adicionales

Algunos comentarios adicionales aplicables al proceso de búsqueda del error:

El proceso de localización del error requiere realizar varias ejecuciones del programa (o circuito) hasta determinar el módulo o bloque donde se encuentra dicho error. Es muy importante repetir las ejecuciones del programa realizando **siempre la misma prueba**. En algunos casos es difícil realizar la misma prueba pero se debe intentar reproducir las condiciones más parecidas posibles: por ejemplo, en el caso de probar pulsaciones largas de las teclas es complicado precisar en todos los casos la misma duración de la pulsación, pero se debe intentar que sea lo más parecida posible.

Cuando se están analizando bloques grandes, la acción de depuración suele ser, ejecutar con **puntos de parada** (situados al comienzo y al final del bloque) e inspección de variables, registros o señales.

Para analizar dentro de un bloque, la acción de depuración suele ser ejecutar con puntos de parada, paso a paso, e inspección de variables, registros o señales.

El proceso natural es BACKTRACKING pero en muchos casos es muy difícil identificar el punto de final (punto donde la previsión y lo ocurrido son diferentes): un ejemplo es cuando un programa genera un *segmetation fault* sin poder ver dónde ocurre. En estos casos no hay más remedio que plantear una estrategia **FORWARDTRACKING**.

Nota importante para el LSED: cuando ocurre un error hardware que deja al programa bloqueado, la opción más interesante para identificar el punto del final es mirar el contador de programa. En el caso de que este contador tenga un valor que no corresponde con ninguna dirección de nuestro programa, lo más probable es que el error se haya producido en la recuperación del contador de programa de la pila: tras RTS o RTE.

La depuración de **BUCLES** puede ser muy laboriosa si el número de iteraciones del bucle es muy elevado. En primer lugar hay que ver si el error se produce en el bucle. En ese caso, el consejo es avanzar de varias en varias iteraciones utilizando la posibilidad que nos ofrecen los puntos de parada. A medida que nos vamos acercando al error vamos reduciendo el número de iteraciones que avanzamos y así hasta localizar la pasada del bucle que generó el error.

7.5.3 PASO 3: Determinación de la causa del error

Una vez detectado el error se debe detectar la causa u origen de dicho error: una instrucción no válida, un modo de direccionamiento incorrecto, una variable sin inicializar,...

En este paso, al igual que en los anteriores, puede **consultar al profesor,** quien le dará las orientaciones oportunas para encontrar la explicación del error.

7.5.4 PASO 4: Corregir el error

Una vez detectada la causa del error se debe corregir dicho error realizando un rediseño o re-implementación de esa parte del programa o circuito.

Una vez realizada la modificación es necesario **volver a probar el sistema con la misma prueba** que nos permitió localizar el error. Si el error ha sido correctamente corregido y no había más errores adicionales, la prueba debe dejar de ser fallida y el sistema funcionará correctamente.

NOTA: en el caso de que el alumno se encuentre bloqueado en la búsqueda del error se propone la posibilidad de cambiar de prueba fallida y repetir los 4 pasos que se proponen. Recuerde que en cualquier paso puede consultar con el profesor.

7.5.5 Consejos para errores no sistemáticos y ocasionales

Este tipo de errores son los más complicados de detectar puesto que al ser ocasionales y no seguir un patrón conocido, cualquier proceso de localización nos puede llevar mucho tiempo. Ante esta situación, y en base a la experiencia de los profesores a lo largo de los últimos años, se proponen las siguientes recomendaciones:

- Un primer consejo es que sean otros ojos (diferentes a los que han implementado el sistema) los que evalúen la posible causa del error. En este punto se aconseja a los alumnos que pregunten a los instructores y a los profesores.
- Cuando se dispone de versiones anteriores del programa o circuito se propone realizar algunas de las acciones siguientes:
    - Analizar las diferencias con la última versión del programa o circuito que funcionaba correctamente.

  - Deshacer los últimos cambios e ir introduciéndolos poco a poco comprobando, en cada momento, si aparece o no el error.

- En el caso de no disponer de una versión anterior, una estrategia interesante es modificar de manera controlada el sistema (programa o circuito) desarrollado. Esta modificación puede consistir en substituir un módulo por otro (e.g. si se usase el LCD, una alternativa sería imprimir en la pantalla del EDColdFire) o bien desactivar alguno de los módulos:

  - El último módulo implementado, el más complejo o el menos probado hasta la fecha.

  - El módulo relacionado con el origen de la acción que produce el fallo o el módulo donde se detecta algún desajuste entre el comportamiento real y el esperado.

- Otra acción interesante es analizar puntos intermedios que permitan detectar desajustes entre el comportamiento real y el esperado.

- Para los errores de los programas que producen un error del hardware de la plataforma, es muy recomendable analizar los principales registros (PC, D0-D7 y A0-A7) con el fin de extraer información útil que nos permita detectar el error. Algunas causas muy frecuentes de este tipo de errores son:

  - **Variables no inicializadas** que producen comportamientos aleatorios.

  - Corrupción de la pila por mal uso de las instrucciones RTS y RTE.

7.5.6 Comentarios adicionales sobre depuración de programas

- **Antes de usar una función (o método) es necesario haberla declarado.** Si no se hace, el compilador supone que esa función va a devolver un 'int' (declaración implícita) y nos dará avisos como los siguientes:

  - warning: type mismatch with previous implicit declaration

  - warning: previous implicit declaration of `funcionX'

  - warning: `funcionX' was previously implicitly declared to return `int'

- **Editar los ficheros fuentes (.c o .h) usando un editor que no sea el EDColdFire puede provocar problemas de compilación**, debido a una diferente gestión de los retornos de carro por parte del compilador (caracteres que normalmente no se ven al editar). El fichero afectado parece ser correcto (el EDColdFire lo muestra bien, pero el compilador no lo procesa bien, pero se producen errores de compilación imprevisibles (el EDColdFire lo muestra bien, pero el compilador no lo procesa bien). Se puede intentar corregir el problema copiando el texto desde el EDColdfire, pegándolo sin formato en el bloc de notas, volviendo a copiarlo y, finalmente, pegándolo en el EDColdFire.

- Si la mayoría de los datos de un “array” o “struct” son constantes y no cambian durante la ejecución del programa, **se pueden inicializar directamente al declararlos, siempre y cuando se declaren como 'static'** (respuesta 32 de [8]). Para declarar e inicializar una variable de tipo 'struct', se puede consultar el ejemplo 6.17 de [20] (no el 6.16, que es

muy poco recomendable). El ejemplo 6.18 explica cómo inicializar un 'array', lo cual es extrapolable a 'arrays' contenidos en 'structs'.

- **Si se activan las optimizaciones del compilador, las variables globales** que pueden ser accedidas desde el programa principal y desde las interrupciones, **deberían declararse como volatile**. Esta palabra clave indica al compilador que a la hora de optimizar trate de manera especial estas variables porque pueden ser modificadas por las interrupciones en instantes de tiempo impredecibles para el compilador. Ejemplo: *'volatile BOOL flag;'* (explicación adicional en el capítulo 8 de [20]).

- Se ha publicado una implementación en C incompleta (pero funcional) de la **función printf,** adaptada al EdColdFire por Víctor Miguel Morales [21], la cual facilitará enormemente la implementación de la interfaz de usuario.

8. Desarrollo recomendado

En este apartado se da una división y planificación orientativa del trabajo a realizar en el laboratorio, organizados en forma de hitos que conseguir y sesiones de laboratorio semanales (en la tabla se muestra la correspondencia entre sesiones e hitos):

Tabla 5. Relación entre sesiones de laboratorio e hitos a conseguir.

SESIÓN	HITO QUE CONSEGUIR
0	
1	
2	
3	1
4	REVISIÓN INTERMEDIA
5	
6	2
7	REVISIÓN INTERMEDIA
8	
9	
10	3

8.1 Hitos y sesiones orientativas

Esta es una planificación orientativa en relación al posible desarrollo de la práctica. En caso de que los alumnos superen los objetivos planteados para una determinada sesión antes de la finalización de la misma, **se recomienda al alumno tratar de abordar los objetivos y tareas planteados para la siguiente sesión**.

Igualmente, y en relación con la redacción de la memoria final, **se recomienda que la realización de la memoria se vaya realizando a medida que se van consiguiendo cada uno de los hitos descritos**.

8.1.1 Sesión -1

Antes de asistir a su primera sesión de laboratorio **es necesario preparar el trabajo fuera del laboratorio**, como se ha mencionada en un apartado previo. A través del portal web y de los libros [3] y [4] encontrará recomendaciones y materiales que le resultarán útiles en esta labor.

8.1.2 Hito 1: Menús y filtrado

8.1.2.1 Sesión 0

Un primer objetivo ligado al hito 1 es **desarrollar el sistema de menús completo** para la funcionalidad del sistema de procesamiento digital de audio que queremos aplicar.

El alumno comenzará a diseñar, implementar y depurar el primer prototipo del sistema. **Una vez completados los tutoriales del entorno y del teclado**, el primer paso será implementar el sistema de menús, para una implementación eficiente será **IMPRESCINDIBLE** una labor previa de **diseño**. Para ello elabore (fuera del laboratorio) un **flujograma** de la interfaz de usuario que deberá presentar al profesor el día de la evaluación del Hito 1 e incluirla en la memoria final.

El siguiente paso consistirá en agregar interrupciones temporizadas a los menús. **Una vez completados los tutoriales de los temporizadores**, deberá conseguir provocar interrupciones a 8 KHz, interrupciones que deberán habilitarse y contabilizarse mediante la oportuna actualización de un contador.

Es muy importante traer programas diseñados y escritos en casa, para aprovechar al máximo las horas de laboratorio disponibles.

El **plan de pruebas** propuesto en esta sesión consistiría en probar cada una de las posibilidades del sistema de menús, que las teclas no útiles son ignoradas y no provocan ningún problema, que el objeto estado se actualiza correctamente y que los mensajes en pantalla son adecuados a cada prueba. En relación a las interrupciones se deberá comprobar, **haciendo uso de puntos de parada**, la correcta atención de las mismas prestando especial atención a la actualización del mencionado contador.

8.1.2.2 Sesión 1

El objetivo de esta sesión consiste en **incorporar el empleo del DAC** al sistema. **Una vez completado el tutorial del DAC**, deberá conseguir sacar por la salida analógica de la plataforma ENT400CF una señal almacenada en memoria.

El **plan de pruebas** propuesto en esta sesión consistiría en escribir en el DAC, por medio de la rutina de atención a la interrupción, las muestras de una señal almacenada en un array. Sólo se escribirá una muestra por interrupción. Deberá visualizar con el osciloscopio la señal, medir y justificar la amplitud de ésta. Un ejemplo de señal de test sería:

```
  SeñalTest[NUM_MUESTRAS] =
[128,150,172,194,216,194,172,150,
 128,105,83,61,39,61,83,105,
 128,150,172,194,216,194,172,150,
 128,105,83,61,39,61,83,105,
 128,150,172,194,216,194,172,150,
 128,105,83,61,39,61,83,105,
```

```
128,150,172,194,216,194,172,150,
128,105,83,61,39,61,83,105,
128,150,172,194,216,194,172,150,
128,105,83,61,39,61,83,105]
```

8.1.2.3 Sesión 2

El objetivo de esta sesión consiste en **incorporar el empleo del ADC** al sistema. **Una vez completado el tutorial del ADC**, deberá conseguir leer de la entrada analógica de la plataforma ENT400CF (una muestra en cada interrupción) una señal sinusoidal de 2Vpp (±1V).

El **plan de pruebas** propuesto en esta sesión consistiría en introducir en el ADC, varios valores señal continua y comprobar que los valores entregados por el ADC son correctos. Seguidamente, deberá ser capaz de introducir una señal sinusoidal de 2Vpp (±1V) del generador de funciones y obtener en la salida analógica la misma señal (atenuada por un factor ½ , debido a la diferencia de rangos entre el ADC y el DAC) sobre un valor de continua (que eliminaremos en una sesión posterior vía hardware)

8.1.2.4 Sesión 3

Finalmente, se propone como objetivo último del hito **implementar la funcionalidad completa de caracterización de filtros**.

Para ello el alumno deberá emplear las interrupciones temporizadas añadidas durante la primera sesión, para leer una muestra del ADC, filtrarla con el filtro seleccionado, y enviarla al DAC.

Adicionalmente se implementará la opción de aplicar todos los filtros y posteriormente sumar sus salidas para poder evaluar la respuesta en frecuencia global. Recuerde compensar la ganancia de 3 dB de la respuesta global del banco de filtros a la hora de enviar las muestras al DAC.

El **plan de pruebas** propuesto en esta sesión consistirá en realizar un barrido de frecuencias, introduciendo al ADC varias sinusoides mediante el generador de funciones. Estime el módulo de la respuesta en frecuencia de cada uno de los filtros junto con el módulo de la respuesta global midiendo en la salida analógica con el osciloscopio. Debe incluir al menos en el barrido, las frecuencias centrales y de corte de cada filtro.

8.1.3 Hito 2: Sistema de Ecualización Gráfico

8.1.3.1 Sesión 4

El objetivo de esta sesión consiste en **incorporar el subsistema hardware del vúmetro** basado en la matriz de leds disponible en cada puesto del LSED.

Para ello se debe diseñar, implementar y probar dicho HW de visualización conectándolo al puerto digital de salida. En este caso, el programa cocheFantastico.c puede servir para una prueba básica **incompleta** de las conexiones. **Será obligatorio modificar el programa cocheFantastico.c para comprobar de forma automática el correcto funcionamiento de cada uno de los leds**.

8.1.3.2 Sesión 5

El objetivo de esta sesión consiste en implementar el subsistema software del vúmetro. Para ello, deberá implementar la rutina o rutinas necesarias para escribir el valor del nivel (columnas que deben iluminarse) proporcional a la energía de banda correspondiente a la fila activa en cada periodo de excitación de la matriz de leds.

El **plan de pruebas** para verificar el correcto funcionamiento consiste en comprobar que todos y cada uno de los leds de las filas y las columnas se encienden correctamente al enviar al puerto de salida los valores apropiados, sin que el teclado deje de funcionar. En este sentido, será importante comprobar con el osciloscopio la correcta generación de las señales de excitación aplicadas a las diferentes filas de la matriz de leds, señales necesarias para su oportuno refresco.

8.1.3.3 Sesión 6

El objetivo de esta sesión es doble: implementar la **funcionalidad de ecualización** e implementar el **hardware adicional del subsistema de conversión digital/analógico** (filtro de continua, filtro de reconstrucción y etapa de potencia).

Respecto a la ecualización, el alumno deberá aplicar el factor de atenuación para cada banda establecido mediante la interfaz de usuario y el Teclado de la TL04. A través de dicha interfaz, una vez seleccionada una banda, podremos subir (tecla '8' o bajar de nivel (tecla '9'). La configuración de ecualización sólo se aplicará cuando el usuario se encuentre dentro de esta funcionalidad, nunca cuando nos encontremos en la función de *"caracterización de filtros"* o *"incorporación de reverberación simple"*.

El **plan de pruebas** se compone de:

- Inicialmente, verificar el correcto funcionamiento de la ecualización consistirá en introducir a la entrada del sistema una señal sinusoidal de 1Vp y frecuencia igual al valor de la frecuencia central de cada filtro, ecualizar la señal mediante la interfaz de usuario, que presentará por pantalla cada uno de los valores de atenuación en cada banda (lea la descripción detallada de la sesión 0), y comprobar con el osciloscopio que la señal de salida presenta la amplitud deseada.
- Seguidamente, verificar el correcto funcionamiento de cada módulo hardware del subsistema conversor de forma aislada y en conjunto (empleando señales sinusoidales del generador de funciones).
- Finalmente, verificar que es posible escuchar una sinusoide introducida en la entrada analógica.

8.1.3.4 Sesión 7

El objetivo de esta sesión consiste en **incorporar el hardware adicional del subsistema de conversión analógico/digital** y conectar el reproductor externo a la entrada de nuestro sistema a fin de ser capaces de reproducir un ejemplo de audio (grabe un ejemplo sólo con voz, sin música) y escucharlo mediante auriculares a la salida de nuestro sistema.

Respecto al hardware del subsistema de conversión analógico/digital, diseñe, monte y evalúe cada módulo por separado, previamente a añadir cada módulo a la etapa anterior.

8.1.3.5 Sesión 8

El objetivo de esta sesión consiste en la **implementación de un efecto de reverberación simple** a añadir a la señal de entrada. El usuario podrá introducir el valor de la atenuación (por ejemplo de 2 a 9) y de retardo (en unidades de 100 ms, entre 100 ms y 900 ms).

El **plan de pruebas** en este caso consistirá en introducir una señal cuadrada de 1 Hz al sistema y aplicar una reverberación simple de T=200 ms y atenuación A=2. Y comprobar con el osciloscopio cómo la señal retardada y atenuada se va sumando a la entrada. Finalmente, compruebe el efecto introduciendo la señal del reproductor externo.

8.1.4 Hito 3: Sistema final

8.1.4.1 Sesiones 9 y 10

Durante esta sesión está prevista la realización de la **memoria final** y opcionalmente de **mejoras**.

Es importante que no se deje para el final la redacción completa de la memoria, sino que se vaya escribiendo conforme se van construyendo y probando los diferentes subsistemas.

9. Mejoras

Se trata de modificaciones sobre las especificaciones básicas que pueden permitir al alumno alcanzar una calificación superior a **8 puntos si se realiza en C y 8,5 puntos si se realiza en ensamblador** en el LSED. En este apartado se incluyen algunas mejoras con una indicación sobre la máxima calificación que se podría obtener en cada una de ellas. No obstante, el alumno puede proponer a los profesores otras mejoras no sugeridas en este documento. Las mejoras se valorarán en función de su dificultad. Se proponen tres tipos de mejoras:

- Dificultad BAJA: hasta 0,3 puntos
- Dificultad MEDIA: hasta 0,6 puntos
- Dificultad ALTA: hasta 1 punto
- Dificultad MUY-ALTA: hasta 2 puntos

9.1 Definición de distintas reverberaciones simples seleccionables (“pre-sets”) mediante la interfaz de usuario.

Se implementará un módulo de la interfaz avanzado, mediante el cual se puedan almacenar en memoria los parámetros de distintas reverberaciones para posteriormente seleccionar cuál de ellos aplicar.

Dificultad estimada: BAJA

9.2 “Presets” de ecualización

Se implementará la posibilidad de aplicar modos de ecualización preestablecidos, así como la posibilidad de almacenar en memoria configuraciones de ecualización introducidas por el usuario.

Dificultad estimada: BAJA

9.3 Efecto trémolo

Este efecto consiste en hacer variar el nivel de intensidad de una señal de entrada rítmicamente. La variación del nivel debe de seguir una determinada función. Los parámetros configurables de esta función deben de ser su amplitud (o “profundidad”) y su frecuencia de oscilación. La función que gobernará la variación de la intensidad será una función triangular implementada con un contador por medio de la interrupción periódica.

Dificultad estimada: BAJA

9.4 Efecto reverberación compleja

El efecto de es un fenómeno acústico de reflexión que se produce en un recinto, cuando una onda incide sobre paredes y techos de esté y vuelve parcialmente reflejada. Se implementará el efecto de reverberación incorporando la simulación varias reflexiones pudiendo ajustar la amplitud y retardo de cada una de ellas.

Dificultad estimada: BAJA

9.5 Reducción del rizado de la banda de paso

Esta mejora consistirá en rediseñar el conjunto de filtros para que estos se corten a -6dB manteniendo las frecuencias centrales distribuidas logarítmicamente, a fin de reducir el rizado de la banda de paso al emplear todos en paralelo.

Dificultad estimada: BAJA

9.6 Visualización de los niveles de ecualización en la matriz de leds

Se implementarán el conjunto de rutinas que permitan visualizar los niveles de ecualización seleccionados para cada una de las bandas en la matriz de leds, es decir se tratará de visualizar la respuesta en frecuencia deseada de forma similar a la Figura 1.

Dificultad estimada: MEDIA

9.7 Visualización de la respuesta en frecuencia de la señal de entrada o la señal de salida

Se implementará la posibilidad de elegir mediante la interfaz de usuario qué respuesta en frecuencia deseamos visualizar en el vúmetro: si la de la señal de entrada o la de la señal de salida.

Dificultad estimada: MEDIA

9.8 Construcción de un prototipo del HW en PCB (máximo 0,5 puntos)

En la realización de esta mejora es obligatorio utilizar un programa comercial para el diseño de la PCB. Remitimos al alumno interesado a lo que se comenta en el enunciado y la página web de la asignatura LCEL, destacando que esta mejora es más asequible en LSED porque el circuito a diseñar y construir es más pequeño.

Dificultad estimada: MEDIA

9.9 Mezclador

Esta mejora consistirá en leer adicionalmente la entrada analógica adicional disponible en la plataforma ENT400CF y mezclarla digitalmente con la entrada analógica 1 antes de enviarlas al DAC. La mejora incluye el hardware de acondicionamiento y anti-solapamiento necesario equivalente al hardware de la entrada analógica 1.

Dificultad estimada: MEDIA

9.10 Ampliación del subsistema de visualización

Utilización del LCD (Liquid Cristal Display) de la placa TL04 en lugar de la pantalla del ordenador para mostrar mensajes. Le será necesario adaptar el HW básico porque los terminales usados para manejar el LCD están siendo usados para iluminar los leds.

Un ejemplo de utilización del LCD se puede consultar en el ejemplo “teclado_lcd.asm” que se muestra en el tutorial de manejo de la placa TL04 (capítulo 7 de [3]).

Se valorarán detalles como:

- Incluir símbolos especiales desarrollados por el alumno empleando las posibilidades que ofrece el LCD.
- Implementar una barra de desplazamiento (*scroll*) por medio de un búfer que guarde las últimas líneas impresas (el LCD sólo puede mostrar a la vez 2 líneas de texto).
- HW añadido.

Dificultad estimada: MEDIA

9.11 Comunicación con una PSP, con una PDA o un iPod

En este caso se propone utilizar alguna de las UARTs disponibles en el ColdFire para comunicarse con una PSP, con un iPod o con una PDA (quizá con módulo GPS). El material necesario para realizar esta mejora será facilitado por los profesores.

Dificultad estimada: ALTA

9.12 Control por voz del sistema

Esta mejora consistirá en manejar por voz las opciones del menú. Según el interés mostrado por los alumnos, a lo largo del curso se proporcionarán una o más clases de orientación e introducción al tema, creándose una lista de distribución de correo electrónico para las personas interesadas.

Dificultad estimada: MUY-ALTA

10. Evaluación

Como ya se indicó en la introducción, salvo que se haya indicado lo contrario, la práctica estándar propuesta contiene las **especificaciones mínimas obligatorias** que deben cumplir los sistemas y serán valoradas con un máximo de **8 puntos si se realiza en C y 8,5 puntos si se realiza en ensamblador**. **Esta nota máxima sólo se conseguirá si se cumplen todas las especificaciones, y la memoria, el código, el hardware y las respuestas en el examen oral son perfectos**.

La descripción del sistema de menús de la interfaz de usuario es orientativa, aunque en la evaluación se valorará la calidad de la interfaz desarrollada. Las desviaciones respecto a la interfaz de este enunciado deben ser explicadas en la memoria final.

Obsérvese que se podría dar el caso de que suspenda alguien a quien la práctica le funciona, pero presenta graves deficiencias en los demás apartados; aunque será extremadamente improbable que esto suceda a una pareja de alumnos que hayan realizado la práctica y la memoria basándose en su propio esfuerzo y conocimiento; tenga en cuenta que en una práctica docente es tan importante cómo se hagan las cosas como el hecho mismo de hacerlas.

Además de las especificaciones obligatorias se ofrece al alumno la posibilidad de realizar mejoras con puntuación máxima definida y otras mejoras con puntuación por determinar.

Sobre las consideraciones éticas del trabajo en el laboratorio, remitimos al alumno a la normativa de la UPM y al documento publicado sobre la filosofía de los laboratorios LCEL y LSED, disponible a través del portal de la asignatura. **Tenga en cuenta que se aplicarán métodos automáticos de detección de copias o plagios.**

10.1 Funcionamiento y examen oral (<=4,5 puntos o <=4 puntos)

En este apartado se pueden obtener hasta 4,5 puntos (si se implementa en ensamblador) o 4 puntos (si se implementa en C). Estos puntos provendrán (como máximo) de la comprobación por parte del profesor de que:

- el prototipo desarrollado responde completamente a las especificaciones obligatorias contenidas en este enunciado,
- el alumno responda correctamente a las **preguntas** que le formule el profesor durante el **examen oral**.
- la nota obtenida en la **revisiones intermedias obligatorias**. Dichas revisiones forma parte de la evaluación de la asignatura, de acuerdo con el apartado 11 de las normas [1] y

el apartado “Evaluación” del programa de la asignatura. La nota máxima en cada prueba es 0,5 puntos y 1 punto respectivamente, que forman parte de los 4 o 4,5 puntos que se pueden obtener, como máximo, por “Funcionamiento y examen oral”. La calificación obtenida será comunicada al realizar el examen oral final.

Se recuerda que para aprobar el examen oral es necesario un **buen dominio individual del depurador del EDColdFire** (cargar, ejecutar y parar un programa, establecer puntos de parada, visualizar variables o zonas de memoria, realizar modificaciones sobre un programa, etc.) de acuerdo con lo publicado en el documento de “Principios y valores” [2], especialmente en el **apartado 4.4, aplicable tanto a C como a ensamblador**.

10.2 Memoria final (<=1,5 punto)

La memoria final puede suponer hasta 1,5 puntos de la nota final:

- la calidad de la redacción, la estructuración y la presentación del documento
- las justificaciones técnicas de los valores y decisiones adoptados,
- la claridad y la completitud en las explicaciones técnicas de cada uno de los módulos HW o SW
- etc.

10.3 Calidad del SW (<=2 puntos)

La calidad del SW desarrollado, explicado y documentado puntuará hasta 2 puntos:

- La **estructura general del programa** en ensamblador debe responder a lo comentado en la sección 4.1.2 de [5].
- Cada subrutina debería tener **un único punto de comienzo y un único punto de terminación**, y no debería entrelazarse con otras subrutinas (según se comenta en el documento “Dudas y preguntas frecuentes del LSED”). Sí que es conveniente que las rutinas se aniden (una rutina llame a otra para realizar una cierto cálculo y al terminarse la segunda rutina, pueda continuar la ejecución de la primera).
- Debería emplear **un buen número de rutinas** (subrutinas, métodos o funciones), que le ayuden a dividir cada problema en sub-problemas, y que no resulten muy extensas (y por lo tanto difíciles de entender y depurar). Algunas rutinas recomendadas se han ido incluyendo a lo largo de este enunciado. Si pensamos en una media de casi 3 métodos por objeto, 25 rutinas parece un número mínimo exigible para esta práctica.
- Debe **emplear constantes** EQU o #define que permitan reconfigurar fácilmente el sistema (las instrucciones deberían no usar números mágicos). Ejemplos típicos podrían ser:
    - Una constante que permita fácilmente variar la frecuencia de las interrupciones temporizadas (FREQ_INT EQU ... o bien #define FREQ_INT...) .

  - Una constante para modificar el retardo antirrebotes del teclado (RET_REBOT EQU... o bien #define RET_REBOT...).

- Debe emplear **modos de direccionamiento variados y apropiados** (indexado, indirecto, post-incremento...) y usar estructuras de datos que agrupen datos interrelacionados y que faciliten la implementación (búferes, tablas de valores...). Emplear estas técnicas suele producir código más compacto y elegante, evitándose frecuentemente el abuso de la desaconsejable técnica de "copiar, pegar y modificar".

- Debe incluir **comentarios orientativos** en el código (comentarios que aporten información al que los lea, no que se limiten a parafrasear el código). Un ejemplo paradigmático de un mal comentario es:

  - ADD.L #1,D0 * incrementa el registro D0

- Debe emplear typedef, struct, switch para producir soluciones elegantes, en C. Por cada objeto debería haber al menos un typedef y un struct. Por ejemplo:

```
typedef struct {
int BufferCircular[MAX_LONGITUD];
int *UltimaMuestra;
...
} TBufferCircular;
```

- Un **empleo abundante y adecuado de #include** para dividir el código en subsistemas coherentes, en C. Al menos debería haber al menos un #include por objeto.

- Debe emplear **pocas variables globales**, sobre todo en C. Sólo parece haber 4 objetos globales: estado, relojes, puerto y resultados. Para minimizar las variables globales debe:

  - usar variables static locales a los métodos o funciones en C, para que esas variables sólo puedan ser usadas por ese método, pero no sean variables locales que se pierdan al terminar la ejecución de la rutina, y que así en la siguiente ejecución de la rutina, mantengan el valor que tenían al acabar la anterior ejecución:

```
void rutinaInterrupcionConStaticLocal()
  {
  static TSeñal SeñalTest=...;
...
  }
```

  - pasar parámetros a los métodos o funciones en C: por valor (si el parámetro no va a ser variado por la rutina) o por referencia (en caso contrario):

```
inline int buffer_escribir(TBufferCircular *pBC, int dato)
{
 *(pBC->UltimaMuestra) = dato;
  ...
}
```

- El empleo de punteros y arrays en C es recomendable para acceder a estructuras de datos complejas de una manera elegante.

```
int buffer_init(TBufferCircular *pBC)
{
  pBC->UltimaMuestra = &(pBC->BufferCircular);
}
...
TBufferCircular bufferCircular;
...
buffer_init(&bufferCircular);
...
```

- No se debe emplear goto, en C.
- Etc.

10.4 Calidad del HW (<=0,5 puntos)

La calidad del HW básico puede puntuar un máximo de 0,5 puntos:

- estructura,
- niveles de ruido presentes, etc.

11. Matrículas y diplomas de honor

De acuerdo con la legislación vigente, en esta asignatura sólo se pueden asignar hasta un 5% de matrículas de honor. Como habitualmente el número de prácticas destacadas del LSED supera este límite legal tan estricto, se podrían conceder hasta 7 diplomas de honor a las mejores prácticas (o aspectos de una práctica) de alumnos que no obtengan matrícula de honor:

- Diploma de honor por lo innovador que fue el sistema desarrollado.
- Diploma de honor por la excelente calidad del código escrito en C.
- Diploma de honor por la excelente calidad del código escrito en ensamblador.
- Diploma de honor por lo innovadora que fue una de sus mejoras.
- Diploma de honor por la excelente calidad del subsistema hardware.
- Diploma de honor por la excelente calidad de la memoria presentada.
- Diploma de honor por la calidad del vídeo explicativo del sistema desarrollado.

Este año además se podrá optar a diploma en dos categorías relacionadas con prácticas que incorporen en modo alguno algún dispositivo móvil tipo iPhone/iPad/Ipod Touch/Móvil Android:

- Diploma de honor por lo innovador que fue el sistema/aplicación desarrollado con dispositivos móviles.

- Diploma de honor por el excelente uso de las posibilidades/recursos ofrecidos por los dispositivos móviles (ej.: acelerómetros, GPS, brújula, etc.).

A las prácticas que opten a matrícula de honor o a alguno de los diplomas (a excepción de las categorías relacionadas con la calidad SW y calidad de la memoria), se les pedirá que graben uno o varios vídeos explicativos de su práctica. Estos vídeos podrían ser incluidos en el canal de la asignatura en YouTube, si los alumnos implicados no se oponen a su difusión por este medio. El Departamento dispone de un par de cámaras con capacidad para grabar vídeos, que pueden ser usadas en el B-043.

Los alumnos que reciban estas matrículas y diplomas tendrán prioridad en la concesión de Proyectos Fin de Carrera en el Departamento de Ingeniería Electrónica.

La ceremonia de entrega de los Diplomas de Honor tendrá lugar en acto público celebrado en el Salón de Grados de la ETSIT. Durante la misma, se proyectará un vídeo-resumen que recoja las mejores prácticas del laboratorio.

12. Bibliografía

[1] *Información general del Laboratorio de Sistemas Electrónicos Digitales (LSED).*

- ✓ http://lsed.die.upm.es

[2] *Principios, valores, objetivos y aspectos éticos del LCEL y el LSED.*

- ✓ http://lsed.die.upm.es/public/docs/MetaLaborariosLCELLSED.pdf

[3] *R. San Segundo et al*, "Introducción a los Sistemas Digitales con el micro-controlador MCF5272", Ed. Marcombo S.A., 2006a.

- ✓ Básico para el desarrollo de la práctica del MCF5272, dado que describe varios tutoriales para el manejo de los principales recursos de la plataforma ENT2004CF.

[4] *R. San Segundo et al, "Entorno de desarrollo EDColdFire V3.0" Ed. ETSIT, 2006.*

- ✓ Básico para el desarrollo de la práctica del MCF5272, dado que describe el manejo del programa EDColdFire y contiene ejemplos sencillos para el manejo de la plataforma ENT2005CF.
- ✓ http://lsed.die.upm.es/public/docs/tutorialBasicoEDColdFire_v7.htm
- ✓ http://lsed.die.upm.es/public/docs/EDColdFire_ayuda.htm

[5] *Javier Ferreiros et al, "Aspectos Prácticos de Diseño y Medida en Laboratorios de Electrónica", Ed. ETSIT. 2002*

- ✓ Básico para el desarrollo de la práctica (sobre todo, el capítulo 4).

[6] *J.M. Montero et al. "How-To de los equipos del laboratorio" Disponible en Internet a través de http://lsed.die.upm.es/ 2006.*

- ✓ Básico para el desarrollo del HW.
- ➢ http://lsed.die.upm.es/

[7] *J.M Montero et al. "Dudas y preguntas frecuentes del LSED" Disponible en Internet a través del portal del LSED http://lsed.die.upm.es/ 2006.*

[8] *J.M Montero et al. "Dudas y preguntas frecuentes del LSED Programación en C".*

- ✓ Especialmente útiles en las primeras fases del desarrollo.
- ➢ http://lsed.die.upm.es/

[9] *C. Carreras et al "Sistemas Electrónicos Digitales" Ed. ETSIT. 2005*

[10] *Manuales del MCF5272 de Freescale.*

- ✓ Sobre todo si se implementa en lenguaje ensamblador o se usan recursos para los cuales no hay tutoriales disponibles

[11] *J.M Montero et al. "Transparencias sobre programación en C para alumnos de Java".*

- ✓ Sobre todo si se carece de experiencia en lenguaje C.
- ➢ http://lsed.die.upm.es/

Básicos para el desarrollo del HW:

[12] *"Hojas de características del 74HC244".*

- ➢ http://lsed.die.upm.es/

[13] *"Hojas de características del amplificador LM386".*

- ➢ http://lsed.die.upm.es/

[14] *Enunciado de la Práctica del Laboratorio de Circuitos Electrónicos (LCEL), Curso 2011-2012*

- ➢ http://neirol-vm-50.die.upm.es/~profes/lcel/1112/public/docs/enunciado1112.pdf

[15] *Enunciado de la práctica estándar del Laboratorio de Sistemas Electrónicos Digitales (LSED), Curso 2008-2009, "*Piano Hero*: un juego musical basado en el MCF5272".*

- ➢ http://neirol.die.upm.es/~profes/lsed/0809/public/docs/Enunciado_lsed0809-v1_4.pdf

[16] *David Brown "Objects in Plain English" Ed Wiley & Sons, 1998*

- ✓ Documento riguroso y ameno, que compendia las metodologías expuestas por diversos expertos en el análisis y en el diseño orientado a objetos.

[17] *Plantillas para la creación de un nuevo archivo .c o de un nuevo archivo .h.*

- ✓ http://lsed.die.upm.es/public/sw/objeto.c
- ✓ http://lsed.die.upm.es/public/sw/objeto.h

[18] *Esquemas de salida y entrada del entrenador.*

- ✓ http://lsed.die.upm.es/public/docs/salidasEntrenador.jpg
- ✓ http://lsed.die.upm.es/public/docs/entradasEntrenador.jpg

[19] *Fotos sobre cómo conectar el HW de la práctica a la placa TL04.*

- ✓ http://lsed.die.upm.es/public/fotos/hw1.jpg
- ✓ http://lsed.die.upm.es/public/fotos/hw2.jpg

[20] *The C book, online version of The C Book, second edition by Mike Banahan, Declan Brady and Mark Doran, originally published by Addison Wesley in 1991 (freely available).*

- ✓ http://publications.gbdirect.co.uk/c_book/

[21] *Implementación en C incompleta (pero funcional) de la función printf, adaptada al EdColdFire por Víctor Miguel Morales.*

- ✓ http://lsed.die.upm.es/public/sw/printf.zip

Esta bibliografía se puede encontrar fácilmente en Publicaciones de la ETSI Telecomunicación de la UPM, en su biblioteca o ciberteca, o en el laboratorio B-043.