PRÁCTICAS DE PROCESAMIENTO DE SEÑALES EN TIEMPO REAL

G. FARIAS[1], M. SANTOS[2]
[1]*Departamento de Informática y Automática.*
Escuela Técnica Superior de Ingeniería Informática. UNED. España
[2]*Departamento de Arquitectura de Computadores y Automática.*
Facultad de Informática. Universidad Complutense de Madrid. España

En este trabajo se exponen prácticas que realizan los alumnos de Ingeniería Electrónica de la UCM que facilitan la comprensión de los principales conceptos del procesamiento digital de señales de audio. Las prácticas se dividen en dos etapas, en la primera se utiliza Matlab como herramienta de diseño, en la segunda se procede a la implementación en tiempo real de los algoritmos en una DSP. La comprobación visual y auditiva del procesamiento aplicado permite la auto evaluación y motivación del alumno.

1. Introducción

La información, desde el siglo pasado, es la herramienta más poderosa de nuestra sociedad, tanto a nivel cultural como político, económico, técnico, científico, etc. Pero la información se encuentra hoy en día en unos formatos y se transmite con unas técnicas que son producto de la más reciente tecnología. De hecho, somos testigos del avance vertiginoso de las comunicaciones debido a la incorporación de nuevas estrategias y dispositivos que hacen más eficiente su transmisión.

Junto con esto, hemos detectado en los alumnos de la Universidad un interés creciente por saber trabajar con información que puede provenir de fuentes muy diversas. Los estudiantes de áreas experimentales y técnicas deben ser capaces de procesar las señales físicas independientemente del ámbito en el que se hayan generado. Así, ya sea una imagen de satélite o tomada con una cámara digital, ya sea un registro que recoge una serie de operaciones bancarias, bien un encefalograma que representa un tumor cerebral, o una señal de voz, etc., todas estas señales pueden aportar una información muy valiosa, en algunos casos crucial, que hay que saber tratar.

En la actualidad el personal docente cuenta con múltiples alternativas de software para la realización de prácticas de simulación de procesamiento de señales, que proporcionan gran cantidad de funciones y algoritmos. Pese a las ventajas de estos tipos de herramientas, presentan una carencia importante que las aleja de la situación real, ya que la ejecución de las mismas tiene lugar en entornos controlados o simulados, con señales en general pre-procesadas, y sin considerar restricciones temporales, saturaciones, otras no linealidades, etc., en la respuesta de los sistemas hardware.

El alcance de este aspecto descrito anteriormente puede ser considerable para estudiantes de áreas técnicas. Es claro además que cuantos más elementos de realidad incorporen las prácticas, mayor será la experiencia y confianza que tenga el alumno al enfrentarse su futuro laboral.

Este trabajo tiene como objetivo mostrar el desarrollo de algunas prácticas de tratamiento de señales en tiempo real que incorporan elementos de la realidad. Para ello se utilizan tarjetas de procesamiento digital de señales (DSP) como plataformas de soporte y desarrollo. Se están realizando actualmente en los estudios de Ingeniería Electrónica de la Universidad Complutense de Madrid, y los alumnos se han mostrado muy receptivos a este tipo de iniciativas.

La utilización de este tipo de estrategias docente persigue la implantación de metodologías más activas y participativas orientadas a facilitar y mejorar el proceso del aprendizaje de los estudiantes así como su consolidación.

2. Objetivos y prácticas

El alumno puede realizar las experiencias propuestas para incorporar adecuadamente las enseñanzas teóricas de las materias relacionadas, que en algunas ocasiones requieren mucho tratamiento matemático, lo que les puede hacer perder la visión global o la utilidad de esos conocimientos. El alumno al realizar las prácticas puede combinar el entorno de simulación con el de la realización hardware en una placa DSP (Fig. 1).

Las prácticas propuestas están orientadas a la introducción de plataformas de desarrollo para el tratamiento digital de señales en tiempo real en los estudios de los alumnos. Específicamente el desarrollo aquí reseñado permite un primer acercamiento a los algoritmos de procesamiento de señales en una tarjeta DSP DSK6713 [1]. La elección de esta tarjeta se basó en consideraciones de coste, potencialidad en tiempo real (soporte Multi-hebra, biblioteca de funciones propia, CODEC, 16MB de RAM, etc.) y sobretodo en las facilidades otorgadas por su ambiente de desarrollo y depuración (Code Composer Studio). Se presentan a continuación tres de los proyectos prácticos que realizan los alumnos con este fin.

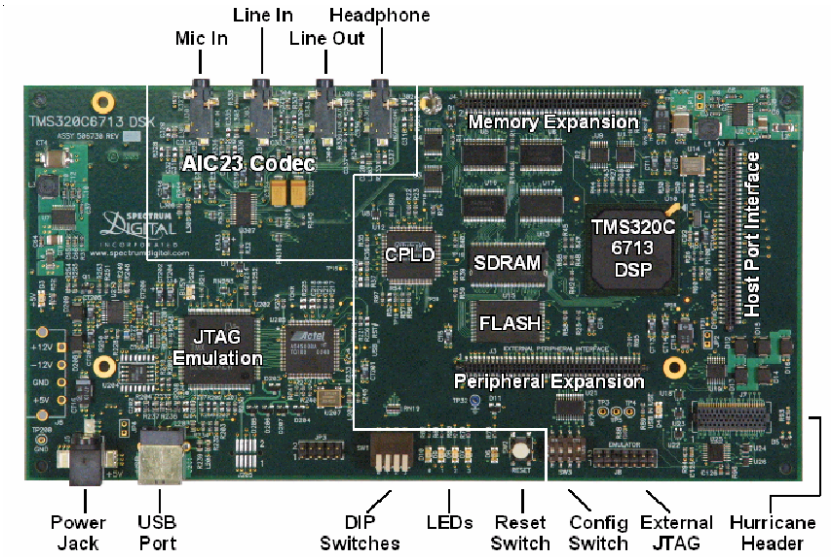


Figura 1. Vista de la DSP TMS320C6713 DSK.

2.1. Generación de señales musicales en tiempo discreto

El objetivo de esta práctica es implementar en la tarjeta DSK6713 una función para la obtención de notas musicales implementadas como señales digitales y posteriormente la ejecución de una partitura. Debido a que las notas musicales son señales oscilantes es posible reproducirlas como tonos puros mediante la utilización de formas de onda sinusoidales, cuyas frecuencias serán las mismas que la nota. La amplitud de la onda evidentemente permitirá establecer el volumen del tono.

La síntesis de notas descritas anteriormente sólo permite la generación de tonos puros, es decir, señales ideales cuyo realismo sonoro es bastante pobre. Para mejorar la calidad del tono se sugiere implementar una amplitud dependiente del tiempo o la incorporación de armónicos con un volumen ligeramente inferior. Con estas consideraciones es posible variar la forma de onda de la señal, como si se tratara de un instrumento musical, y por tanto mejorar la calidad sonora de ésta.

En esta práctica el alumno tiene acceso mediante bibliografía específica y a través de internet a frecuencias y partituras que ejecutará. El diseño inicial se realiza en Matlab [2], mediante el uso de funciones propias, como ***sin(2*pi*fnota)*** o ***sound(nota,fmuestreo)***, y posteriormente la

implementación se ejecuta sobre la DSP utilizando el ambiente de desarrollo y depuración Code Composer Studio (Fig. 2).

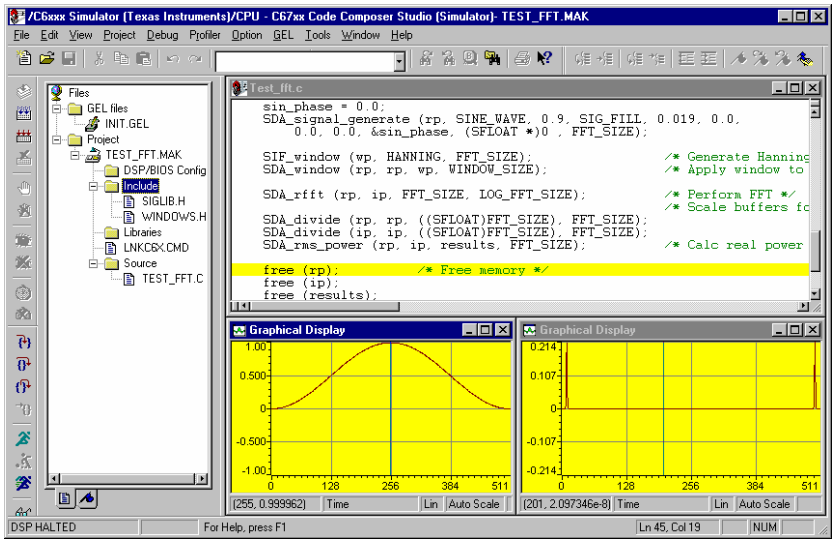


Figura 2. Interfaz gráfica del Code Composer Studio.

Con el desarrollo de la práctica, mediante la audición y sobre todo la visualización de las formas de ondas de las señales, permiten al alumno observar y tratar con conceptos básicos como frecuencia de muestreo, volumen o frecuencia de la nota.

La DSP TMS320C6713 proporciona un codificador / decodificador (CODEC AIC23) análogo digital que permite interactuar con las líneas de audio de entrada y salida.

Para manejar el CODEC se pueden utilizar las funciones proporcionadas por las bibliotecas de soporte de la tarjeta [1,3].

Los elementos más importantes para la realización del código de todas las prácticas son:

- **dsk6713.h**: Biblioteca general para el manejo de la tarjeta.
- **dsk6713_aic23.h**: Biblioteca específica para el manejo del CODEC.
- **DSK6713_init()**: Función para inicializar de la tarjeta.
- **hCodec = DSK6713_AIC23_openCodec(0, &config)**: Función para inicializar el CODEC con una configuración particular. La variable *hcodec* tendrá el valor que identifica el CODEC.
- **DSK6713_AIC23_write(hCodec, output)**: Función que escribe un valor entero en el CODEC y que será transformado en un voltaje en un canal de audio de salida de la tarjeta (*Headphone*).
- **DSK6713_AIC23_read(hCodec, &input))**: Función similar a la anterior que obtiene un valor entero representativo del voltaje aplicado en la señal de audio de entrada de la tarjeta (*Line In*).
- **math.h**: Biblioteca general de funciones matemáticas.
- **sin(2*pi*nota*sample)**: Función seno proporcionada por la Biblioteca *Math.h*.

A continuación se presenta comentado el programa principal completo que ejecuta una melodía en el procesador TMS320C6713 de la tarjeta. Evidentemente los alumnos no disponen del código

sombreado que define y ejecuta la partitura (Fig. 3 y 4). Obsérvese la configuración del CODEC, y especialmente la frecuencia de muestreo seleccionada (48KHz).

Como elementos opcionales de la práctica se proponen dos de los elementos mencionados anteriormente para incorporar mayor realismo en las notas.

1 Volumen variable: Para que la amplitud de la onda varíe puede incorporarse fácilmente una función como la Ec. 1:

$$V(t) = A \cdot e^{-t/\tau} \tag{1}$$

Donde A es la amplitud inicial, t el tiempo y τ la constante de decaimiento.

2 Incorporación de armónicos: Si se incorporan dos o más señales sinusoidales cuya frecuencia sea múltiplo de la nota es posible mejorar la calidad sonora.

```
#include "melodiacfg.h"

/* Cargar bibliotecas de la tarjeta y módulo*/

#include "dsk6713.h"
#include "dsk6713_aic23.h"

/* Cargar biblioteca general de funciones matemáticas*/

#include "math.h"

/*Frecuencias de Notas*/

#define fdo 523
#define fre 587
#define fmi 659
#define ffa 698
#define fsol 783
#define fla 880
#define fsi 987
#define fdo6 1046
#define fre6 1174

/* Duración de Notas */

#define c 0.5  //corchea
#define n 1.0  //negra
#define b 2.0 //blanca
#define r 4.0  //redonda

/* Configuración del Codec */

DSK6713_AIC23_Config config = { \
    0x0017,  /* 0 DSK6713_AIC23_LEFTINVOL  Volumen del canal izquierdo entrada*/ \
    0x0017,  /* 1 DSK6713_AIC23_RIGHTINVOL Volumen del canal derecho entrada */\
    0x00d8,  /* 2 DSK6713_AIC23_LEFTHPVOL  Volumen del canal izquierdo headphone */  \
    0x00d8,  /* 3 DSK6713_AIC23_RIGHTHPVOL Volumen del canal derecho headphone */ \
    0x0011,  /* 4 DSK6713_AIC23_ANAPATH    Control de audio análogo */      \
    0x0000,  /* 5 DSK6713_AIC23_DIGPATH    Control de audio digital */     \
    0x0000,  /* 6 DSK6713_AIC23_POWERDOWN  Control power down */             \
    0x0043,  /* 7 DSK6713_AIC23_DIGIF      Formato de audio digital */ \
    0x0081,  /* 8 DSK6713_AIC23_SAMPLERATE Tasa de muestreo 48K[Hz] */           \
    0x0001   /* 9 DSK6713_AIC23_DIGACT     Activación de la interface digital */   \
};
```

Figura 3. Primera parte del código para generar la melodía.

```
/*  main() - Rutina principal, Inicializa BSL y Genera melodía */

void main()
{
    DSK6713_AIC23_CodecHandle hCodec;
    Int16 fseno,buffindex;
    float nota,duracion,sample;
    float pi=3.141592;

    /* Define partitura, los elementos del array son frecuencia y duración de la nota*/
    float partitura[110]={fsi,n,fdo6,n,fre6,n,fre6,n,fdo6,n,fsi,n,fla,n,fsol,n,
        fsol,n,fla,n,fsi,n,fsi,b,fla,b,fsi,b,fdo6,n,fre6,n,fre6,n,fdo6,n,
       fsi,n,fla,n,fsol,n,fsol,n,fla,n,fsi,n,fla,b,fsol,b,fla,b,fsi,n,fsol,n,
       fla,n,fsi,c,fdo6,c,fsi,n,fsol,n,fla,n,fsi,c,fdo6,c,fsi,n,fla,n,fsol,n,
       fla,n,fre,b,fsi,b,fdo6,n,fre6,n,fre6,n,fdo6,n,fsi,n,fla,n,fsol,n,fsol,n,
       fla,n,fsi,n,fla,b,fsol,b};

    /* Inicializa tarjeta */
    DSK6713_init();

    /* Inicializa Codec */
    hCodec = DSK6713_AIC23_openCodec(0, &config);

    /* Ejecuta melodia definida por la partitura */
    for (buffindex=0;buffindex<110;buffindex=buffindex+2)
    {
       nota=partitura[buffindex];
       duracion=partitura[buffindex+1];

        for (sample = 0.0; sample <(duracion/2.0); sample=(48000*sample+1)/48000)
        {

            fseno=(Int16)(32000*sin(2*pi*nota*sample));

            /* Enviar voltaje al canal izquierdo */
            while (!DSK6713_AIC23_write(hCodec, fseno));

            /* Enviar voltaje al canal derecho */
            while (!DSK6713_AIC23_write(hCodec,fseno));

        }

    }
       /* Cierra el codec*/
    DSK6713_AIC23_closeCodec(hCodec);
 }
```

Figura 4. Segunda parte del código para generar la melodía.

2.2. Generación de efectos digitales de audio

El objetivo de esta práctica es ejecutar mediante la DSP TMS320C6713 funciones que implementan efectos digitales de audio, como vibrato y tremelo, que permiten mejorar la calidad de la música. Los efectos musicales que se agregan utilizan operaciones matemáticas simples que se proporcionan al alumno. El vibrato y tremelo son dos efectos muy utilizados que añaden distorsiones sobre las notas musicales.

La realización de estas experiencias tiene un gran alcance pedagógico: el alumno tiene que salir al paso de problemas que atañen distintos aspectos de la realización que suelen surgir al trabajar con sistemas físicos. Nos parece fundamental que los estudiantes tengan contacto con los sistemas reales con los que se enfrentan en el mundo laboral. El procedimiento para la realización de la práctica es similar a la anterior, es decir el análisis y diseño se realiza en Matlab, y posteriormente se debe implementar un algoritmo similar en la tarjeta DSP. Evidentemente el código de la práctica 1 sirve de base para la práctica 2. Para evitar describir todo el código nuevamente, se presentará sólo la parte que corresponde a la implementación de los efectos. En el ciclo *for* más interno del código de la práctica 1 (Fig. 4) el alumno debe introducir un conjunto de instrucciones similar al sombreado (Fig. 5):

```
for (sample = 0.0; sample <(duracion/2.0); sample=(48000*sample+1)/48000) {
            switch (tipo efecto)
            {
               // Vibrato
               case 1:
               fm=10;
               efecto =sin(2*pi*nota*sample)*sin(2*pi*fm*sample);
               break;
              // Tremelo
              case 2:
               fm=5;
               m=0.01;
               efecto=sin(2*pi*nota*sample + nota*m*sin(2*pi*fm*sample));
               break;
              // Tono puro
               default:
               efecto=sin(2*pi*nota*sample);
            }
            fseno=(Int16)(32000*efecto);

            /* Enviar voltaje al canal izquierdo */
            while (!DSK6713_AIC23_write(hCodec, fseno));
            /* Enviar voltaje al canal derecho */
            while (!DSK6713_AIC23_write(hCodec,fseno));
 }
```

Figura 5. Código para el desarrollo de los efectos de audio vibrato y tremelo.

Los efectos de coro y eco utilizan la retroalimentación, la amplificación y el retardo para elaborar una salida auditiva proveniente de la distorsión tanto en el tiempo como en la amplitud de la señal original. Se propone en forma opcional a los alumnos la incorporación de este tipo de efectos en la práctica desarrollada. Otro desarrollo que se propone a los alumnos es la implementación de filtros digitales, específicamente se propone la incorporación de un filtro FIR para el procesamiento de las señales de audio sintéticas.

2.3. Aplicación de un filtro a señales de audio en tiempo real

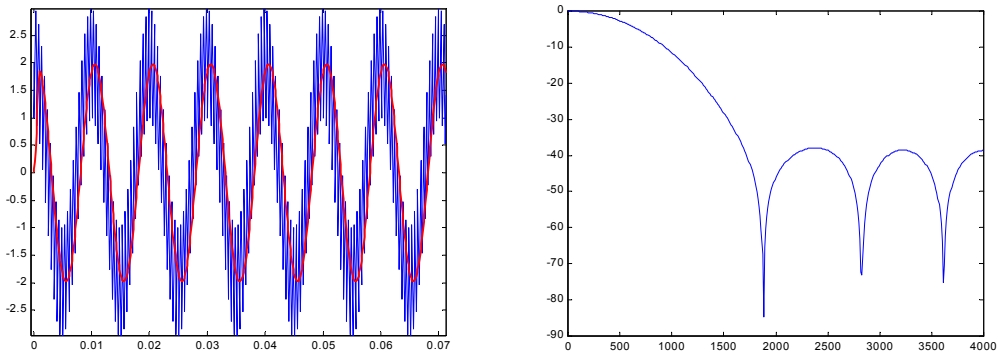
Esta práctica tiene por objetivo el procesamiento mediante un filtro FIR de una señal de audio en tiempo real. Al igual que en las prácticas anteriores será necesario utilizar funciones propias de Matlab para el diseño y generación de los coeficientes del filtro. Posteriormente el alumno deberá desarrollar la función de filtrado y utilizarla adecuadamente en el proyecto de la DSP proporcionado para tal efecto.

```
%Parámetros de Diseño
fm=8000; %Frecuencia de Muestreo
fcorte=300; %Frecuencia de Corte
ncoeficientes=10;%Número de Coeficientes

%Señal de Prueba
t=0:1/fm:1; %Tiempo
senal=2*cos(2*pi*t*100)-sin(2*pi*t*2000);
figure(1); plot(t,senal);

%Obtener Coeficientes Filtro FIR
bfir=fir1(10,(2/fm)*fcorte);
f=1:10:(fm/2);
hfir=freqz(bfir,[1,zeros(1,length(bfir)-1)],f,fm);
figure(2); plot(f,20*log10(abs(hfir)));%Respuesta en Frecuencia del Filtro

%Comprobar Filtro
senalfir=filter(bfir,[1,zeros(1,length(bfir)-1)],senal);%senal filtrada FIR

%senal original y filtrada
figure(1); hold on; plot(t,senalfir,'r');
```

Figura 6. Código utilizado para el diseño y análisis del filtro FIR a implementar.

Para el análisis y diseño del filtro se proporciona a los alumnos de un *script* desarrollado en Matlab (Fig. 6).

El *script* anterior genera los coeficientes a partir de las especificaciones de los Parámetros de Diseño introducidos por los alumnos. Mediante el código se obtiene un filtro FIR paso-bajo, aunque es posible obtener uno distinto a través de leves modificaciones. Mediante el *script* proporcionado se obtienes imágenes de la señal original, la filtrada y la respuesta en frecuencia del filtro (Fig. 7). Evidentemente el ajuste fino en cuanto a las características del filtro deseado dependerán de la señal de audio y de los requerimientos que el alumno persiga. Posteriormente la elección de los parámetros del filtro deberán ser justificadas y contrastadas con la respuesta de la sistema (DSP + señal de audio).



a) b)

Figura 7. En a) se observa la señal de prueba en azul y la señal filtrada en rojo. En b) se presenta la respuesta en frecuencia del filtro.

Una vez definido el filtro requerido el alumno debe utilizar el proyecto *FiltroAudio* para introducir el algoritmo y los coeficientes del filtro FIR. El algoritmo de un filtro de este tipo se puede implementar mediante el uso de un vector o arreglo circular en el que se acumulan N muestras de la señal de audio [4-5]. La idea básica consiste en que primero se reemplace la muestra más antigua por la más reciente y posteriormente se aplique la convolución del vector con los N coeficientes del filtro (Fig. 8).

```
vector[indice_muestra_antigua++]= muestra_entrada;
if (indice_muestra_antigua > N-1)  indice_muestra_antigua = 0;
muestra_salida=0; indice_coeficiente=0; indice_buffer= indice_muestra_antigua;

while (indice_coeficiente < N)
{
 muestra_salida = muestra_salida + buffer[indice_bufer++]*filtro[indice_coeficiente++];
 if (indice_buffer > N-1) indice_buffer =0;
}
```

Figura 8. Ejemplo de una función de filtrado utilizando un vector circular.

El programa principal del proyecto *FiltroAudio* presenta dos etapas (Fig. 9). En la primera se realiza la lectura de una muestra de audio mediante sondeo (polling) desde la entrada *Line In*. La segunda etapa realiza la escritura de la misma muestra de audio en la salida *Headphone*. Evidentemente el alumno deberá introducir la función de procesamiento entre las dos etapas para realizar el filtrado.

Como señal de audio se puede utilizar la salida de audio del lector de CD o la de los altavoces del ordenador. En cualquier caso en la etapa de depuración se recomienda utilizar música compuesta de sonidos agudos y bajos de modo que se pueda percibir claramente el efecto del filtrado.

Un aspecto relevante a considerar es la frecuencia de muestreo, en el caso del proyecto *FiltroAudio* ésta esta definida en la configuración del CODEC como de 8KHz (*0x0d1*) y puede ser modificada siguiendo las especificaciones proporcionadas por el manual técnico del CODEC AIC23 [3].

```
/* Polling y Filtro FIR */
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#include "math.h"
/* Configuración del Codec */
DSK6713_AIC23_Config config = { \
    0x0017,  /* 0 DSK6713_AIC23_LEFTINVOL  Volumen del canal izquierdo entrada */ \
    0x0017,  /* 1 DSK6713_AIC23_RIGHTINVOL Volumen del canal derecho entrada  */\
    0x00d8,  /* 2 DSK6713_AIC23_LEFTHPVOL  Volumen del canal izquierdo headphone */  \
    0x00d8,  /* 3 DSK6713_AIC23_RIGHTHPVOL Volumen del canal derecho headphone */ \
    0x0011,  /* 4 DSK6713_AIC23_ANAPATH    Control de audio análogo */      \
    0x0000,  /* 5 DSK6713_AIC23_DIGPATH    Control de audio digital */     \
    0x0000,  /* 6 DSK6713_AIC23_POWERDOWN  Control power down  */             \
    0x0043,  /* 7 DSK6713_AIC23_DIGIF      Formato de audio digital  */ \
    0x000d,  /* 8 DSK6713_AIC23_SAMPLERATE Tasa de muestreo  8K[Hz] */            \
    0x0001   /* 9 DSK6713_AIC23_DIGACT     Activación de la interfase digital  */   \
};

/* main() - Rutina principal, Inicializa BSL, Genera Audio Procesado */
void main()
{
    DSK6713_AIC23_CodecHandle hCodec;
    Uint32 muestra_entrada, muestra_salida;

    /* Inicializa tarjeta */
    DSK6713_init();
    /* Inicializa Codec */
    hCodec = DSK6713_AIC23_openCodec(0, &config);

    for (;;){
         //Leer muestra
         while (!DSK6713_AIC23_read(hCodec, & muestra_entrada));
         muestra_salida = muestra_entrada;
         /* Enviar muestra al canal izquierdo */
         while (!DSK6713_AIC23_write(hCodec, (Int16)muestra_salida));
         /* Enviar muestra al canal derecho */
         while (!DSK6713_AIC23_write(hCodec,(Int16)muestra_salida));
        }

    /* Cierra el codec*/
    DSK6713_AIC23_closeCodec(hCodec);
}
```

Figura 9. Código para generación del sondeo y escritura de la señal de audio.

Al evaluarse la función de filtrado se debe tener en cuenta que probablemente los resultados no serán lo suficientemente satisfactorios como en la práctica anterior, principalmente porque los tipos de señales de audio son diferentes. Sin embargo enfrentarse a tales situaciones es en sí parte de los objetivos perseguidos por las prácticas. Al finalizar la práctica se proponen en forma opcional el desarrollo del procesamiento de la señal de audio mediante un filtro IIR.

3. Metodología

Las prácticas se realizan en el laboratorio de Sistemas y Automática de la Facultad de CC. Físicas, de la UCM, que consta de 15 puestos, de los cuales hay 5 equipados con las placas DSP. Los alumnos trabajan en grupos de 3 para realizar la implementación sobre las mismas y observar los resultados. El número de alumnos oscila alrededor de 30, por lo que se turnan los dos grupos en los que se divide la clase para hacerlas.

Previamente o mientras la mitad de la clase realiza las prácticas sobre las DSP, cada alumno de forma individual desarrolla el código de las prácticas en Matlab para su simulación. Para ello cuentan con un aula de Informática en la Facultad de CC. Físicas, UCM, con 40 puestos en los que está disponible este software. De esta forma, cada uno de ellos ha generado previamente el código de las mismas, depurándolo si es necesario.

El trabajo en grupo consiste entonces en traducir ese código al entorno Code Composer Studio, en C/C++, y realizar las conexiones apropiadas para su funcionamiento. Tal como se ha comentado, la parte del código que gestiona el acceso a las direcciones de la placa se facilita a los alumnos para que sólo tengan que añadir lo que hace referencia a las funciones de procesamiento.

Las prácticas se estructuran en varias fases, cada una de ellas con distintos objetivos:

1 Primero, la realización de la práctica en su formulación básica, para adquirir los conceptos fundamentales necesarios para llevarla a cabo y entender su funcionamiento; así el alumno comprende cómo funciona, la finalidad de la práctica, los conceptos que se quieren resaltar con ella, etc.; sería una labor de síntesis.
2 Segundo, el trabajo de laboratorio propiamente dicho, donde el alumno puede modificar varios parámetros de la práctica, observar su funcionamiento y sacar conclusiones que le ayuden a comprender las explicaciones teóricas; en definitiva, una labor de análisis.
3 Finalmente se puede también entender como un tercer paso la auto evaluación por parte del alumno de su aprendizaje y el asesoramiento del profesor para resolver dudas, etc.

Estas prácticas son una ayuda a las clases teóricas y a los problemas vistos en clase, de forma que el profesor puede hacer referencia a las prácticas existentes, realizar demostraciones o utilizarlas como explicación en ciertos momentos de las clases de teoría [6-7].

Así, la metodología propuesta, junto con la supervisión del profesor, proporciona un modo de aprendizaje rápido y eficiente, sobre todo para establecer relaciones entre conceptos vistos en teoría y los experimentos. Además permite la repetición por parte del alumno de los ejemplos (las veces necesarias), y le ayuda a avanzar en su estudio de una forma natural, motivándole a preguntar el por qué y cómo del funcionamiento de los distintos tipos de señales, el conocimiento que se puede extraer de su procesamiento, etc.

Por otro lado, la metodología específica consiste en:

1. Para que el número de horas que el alumno pase en el laboratorio sea de la máxima utilidad, y teniendo en cuenta que este tiempo suele ser bastante reducido, resulta fundamental el disponer de un guión que describa los experimentos con detalle, y que dé ideas de tipo práctico. En dicho guión conviene indicar los objetivos a conseguir con esa práctica, así como los aspectos que se quieren resaltar en cada una, destacando los aspectos del funcionamiento correcto o incorrecto de la misma –si fuera el caso- para que el alumno compruebe el comportamiento del sistema al variar los parámetros de éste.
2. Las prácticas de laboratorio son esenciales como actividades para introducir al alumno en técnicas de medida, obtención, tratamiento, representación e interpretación de datos, etc. Por eso es importante también describir con claridad el funcionamiento cada uno de los elementos que aparecen en el sistema, de forma que el alumno tenga una visión global del problema, pueda interpretar correctamente su funcionamiento y lo pueda resolver con eficacia.
3. El alumno puede variar distintos parámetros del sistema, modificar algunas características de las señales y observar la respuesta. Además, el alumno puede repetir la práctica cuantas veces desee y variar los parámetros como quiera.

4. Otro aspecto importante, eminentemente práctico, es que con esta herramienta todos los alumnos podrán acceder a la práctica simultáneamente, con lo que se podrían hacer más prácticas o realizar aclaraciones o explicaciones de distintos aspectos del problema que de otra forma sería muy costoso en tiempo. Además cada práctica puede replicarse las veces necesarias, ya que no se consume material.
5. Por último, también forma parte de la metodología la relación que se establezca entre el tutor y los alumnos, así como el seguimiento del aprendizaje tanto por parte del mismo alumno como por parte del profesor.

El seguimiento del proceso de aprendizaje tiene un doble objetivo: guiar al alumno en los aspectos que debe hacer hincapié al realizar la práctica, puesto que en ocasiones los alumnos no aprecian ciertos aspectos de la realización de las mismas que son importantes; y permitir al profesor conocer en qué grado cada alumno va dominando los conceptos impartidos y comprende el funcionamiento del sistema de modulación presentado.

La idea de estas herramientas docentes es incentivar al alumno a que pregunte y participe en las clases, a raíz de las dudas que surjan al realizar las prácticas, y que supongan una motivación para su estudio y profundización, y fomenten su relación con otros alumnos y con el profesor. Este último punto nos parece también importante ya que facilita la consulta de dudas, acudir a tutorías, etc.

4. Conclusiones

El desarrollo de las experiencias descritas tiene como principal objetivo reducir la brecha existente entre la teoría y la práctica real del procesamiento digital de señales. Para ello se utiliza una tarjeta DSP que proporciona diversas funcionalidades para la implementación de algoritmos de procesamiento de señales en tiempo real.

Los alumnos trabajan con una señal desde su implementación inicial hasta la fase final en la que pueden comprobar los efectos de los algoritmos de tratamiento aplicados mediante la audición de la misma en tiempo real.

Se ha generado un material de prácticas que constituye un laboratorio de apoyo a asignaturas de gran importancia para las carreras científicas para las que se propone. Además, por la metodología seguida para las prácticas utilizando estas herramientas, y por la interactividad que permiten, el alumno puede autoevaluarse y por lo tanto aprender de su propio trabajo, analizar resultados, sacar conclusiones, etc. También se dota al profesor de mecanismos para el seguimiento del aprendizaje de los alumnos.

Por último, hacer hincapié en que la transferibilidad del proyecto se ve facilitada porque este conjunto de experiencias se proponen en un entorno estándar, por lo que pueden funcionar en cualquier máquina con pocas prestaciones, y no tienen especiales requerimientos. Añadir también que por su diseño modular, se presta a futuras mejoras y ampliaciones.

Referencias

[1] Spectrum Digital Incorporated. *TMS320C6713 DSK Technical Reference*. (2003).
[2] The Mathworks Inc. *The Student Edition of MATLAB.* Prentice Hall (2002).
[3] Digital Audio Products. *Data Manual TLV320AIC23B.* Texas Instruments (2002).
[4] Chassaing, R. *Digital Signal Processing. Laboratory Experiments Using C and the TMS320C31 DSK.* Wiley-Interscience (1999)
[5] Chassaing, R. *Digital Signal Processing and Applications with the C6713 and C6416 DSK.* Wiley-Interscience (2005)
[6] Oppenheim, A. and R. Schafer. *Discrete Time Signal Processing.* Prentice Hall (1999)
[7] Ifeachor, E.C., Jervis, B.W. *Digital Signal Processing.* Ed. Addison-Wesley (1993)