Sistemas Digitales II

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

Memoria del proyecto desarrollado en Sistemas Digitales II (SDII)

Curso 2014/2015

Título del proyecto desarrollado "Sistema de Visualización de Señales en el Dominio de la Frecuencia basado en el MCF5272"

Autores (orden alfabético): Francisco García de la Corte

Carlos Santos Rancaño

Código de la pareja: LT-07

ÍNDICE GENERAL

1 Introducción

Tras la primera parte del proyecto donde tanto el hardware como el software funcionaron y fuimos capaces de visualizar la DFT en el osciloscopio, en esta segunda parte quisimos hacer nuestro sistema mucho más **completo** aumentando sus prestaciones, sencillo e **intuitivo** a la hora de manejarlo y **eficiente** en cuanto a recursos de ejecución.

En cuanto a las **prestaciones** del sistema las aumentamos añadiendo las mejoras del afinador de frecuencia con una resolución de hasta 5Hz, entrada de la señal a través de un micrófono, hallar la fase de la DFT para complementar al módulo y analizador de circuitos.

Para que sea un sistema intuitivo y fácil de manejar implementamos una interfaz de usuario utilizando el teclado matricial y el LCD tanto estático como dinámico.

Debido al número de mejoras añadidas tuvimos que optimizar todo el código reduciendo el número de bucles y llamadas a funciones, sustituyendo las divisiones por desplazamientos lógicos cuando fuese posible, reducción del número de variables creadas, búsqueda binaria… De esta forma conseguimos que en cada interrupción dé tiempo a realizar todas las operaciones.

Por todo lo anterior consideramos que nuestro sistema actual además de funcional es **variado** y completo.

2 Descripción de las mejoras

2.1 Fase DFT

Paralelo al cálculo del módulo de la DFT, la fase sigue una estrategia bastante similar: una vez se han hallado las partes real e imaginaria de 80 muestras leídas por el ADC, en vez de calcular el módulo al cuadrado, aplicamos un algoritmo desarrollado por nosotros, que sigue la estrategia de "Divide Y Vencerás" para mayor eficiencia, detallada en otro apartado. Para reusar el array de valores de escalado del DAC, tendremos 32 niveles de decisión para el cálculo de la fase, y redondeamos el resultado hacia abajo para evitar posibles ruidos, por lo que hay un error máximo de 360/32 = 11.5 grados.

El algoritmo que hemos implementado para calcular la fase consiste en varios pasos:

1. Determinamos en qué cuadrante del plano real-imaginario se situaría el valor complejo de la DFT, simplemente comprobando sus signos, para después hallar el valor de la fase si trasladásemos el valor complejo al primer cuadrante. Dependiendo de en qué cuadrante se encontraba en un primer momento, se sumarán 0, 90, 180 o 270 grados.
2. Simplificado el cálculo al primer cuadrante del plano real-imaginario, el valor de la fase se situará por encima de los 45 grados si la parte imaginaria es mayor que la real, y si no estará por debajo de los 45 grados.

A partir de aquí nos vimos obligados a usar la definición de la tangente para seguir profundizando en niveles de decisión más precisos, la cual dice:

$$\Psi = \arctan(\text{parte imaginaria}/\text{parte real})$$

De forma que si queremos ver la relación que tendrán las partes real e imaginaria para un ángulo dado (nivel de decisión), sería:

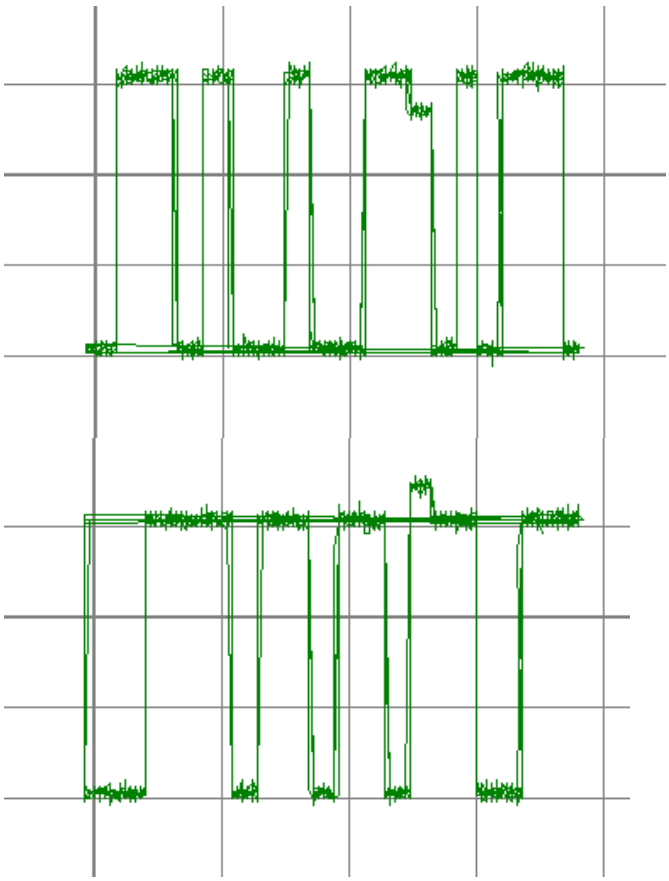
$$\text{Parte imaginaria} = \tan(\text{ángulo}) \cdot \text{Parte real}$$

Como salen números con decimales y queremos evitarlos en el programa por el coste de operaciones, reorganizamos las ecuaciones para cada condición de decisión. Detallamos a continuación los pasos en el caso de que el valor de la fase se sitúe por debajo de los 45 grados y las condiciones resulten en el valor más bajo, ya que por encima las condiciones son iguales e invertidas*:

3. Imaginario = tan(22.5º)*Real: comprobamos si 10*Imaginario < 4*Real; en tal caso, la fase estará entre 0 y 22.5 grados; si no, entre 22.5 y 45 grados.
4. Imaginario = tan(11.25º)*Real: comprobamos si 10*Real < 2*Imaginario; en tal caso, la fase estará entre 0 y 11.25 grados; si no, entre 11.25 y 22.5 grados.
5. Una vez sabemos que la va a estar entre dos valores diferenciados en 11.25 grados (límite de precisión en nuestro algoritmo), asignamos el nivel más bajo para evitar posibles ruidos.

Una vez detallado el algoritmo, pasamos a calcular respuestas en fase de filtros usando el analizador de circuitos, detallado en otro apartado de mejoras.

Para comprobar el correcto funcionamiento de este módulo representamos la fase de un seno y un coseno a 1200Hz, cuyas gráficas deberían salir inversas:



Fase de un coseno a 1200Hz

Fase de un seno a 1200Hz

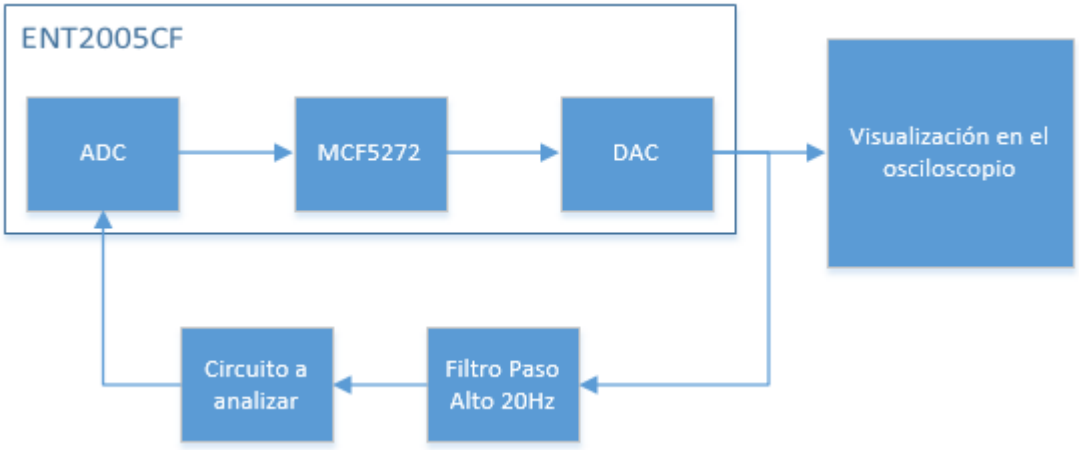
```
Paco@PacoLaptop ~/SDG2
$ ./fase.exe
100
100
Parte real: 100
Parte imaginaria: 100
Fase = 45.000000

Paco@PacoLaptop ~/SDG2
$ ./fase.exe
100
1
Parte real: 1
Parte imaginaria: 100
Fase = 78.750000

Paco@PacoLaptop ~/SDG2
$ ./fase.exe
1
100
Parte real: 100
Parte imaginaria: 1
Fase = 0.000000

Paco@PacoLaptop ~/SDG2
$ ./fase.exe
-100
-100
Parte real: -100
Parte imaginaria: -100
Fase = 225.000000

Paco@PacoLaptop ~/SDG2
$ ./fase.exe
-100
100
Parte real: 100
Parte imaginaria: -100
Fase = 315.000000
```

Así mismo realizamos pruebas en ordenador local para comprobar el correcto funcionamiento de este módulo antes de probarlo en el laboratorio.

*Todos los números están justificados en los comentarios en el código de la función estimaFase0a90, y se ha comprobado que el algoritmo da valores coherentes al simularlo en nuestros ordenadores y pasándole partes real e imaginaria.

2.2 Analizador de circuitos

Esta mejora trata de obtener la respuesta en frecuencia (módulo y fase) de un circuito hardware. Abordamos el problema con conocimientos de teoría de señales y aprovechando el funcionamiento del módulo básico, lo que nos llevó a la siguiente estrategia:

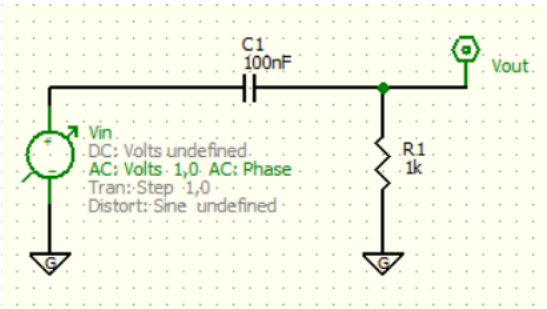


En primer lugar, usamos las muestras de la sinusoide de las que disponemos en el programa para generar picos de frecuencia a través del DAC y barrer en este dominio el circuito a analizar, de forma que hacemos una DFT de cada pico de frecuencia y guardamos el valor a esa frecuencia en un array de valores llamado Hw. Esto equivale en teoría de señales a hacer la convolución de una delta de Dirac y la respuesta en frecuencia del circuito.

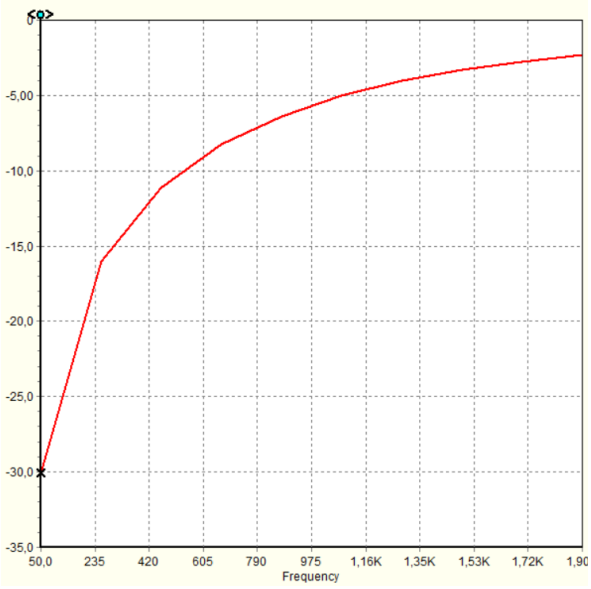
Después, una vez se ha llenado el array Hw con los resultados de barridos desde 50Hz hasta 1900Hz (cuando han pasado 20 cálculos de la DFT, equivalente a 1600 interrupciones), se para la generación de senos y se saca por el DAC este array de la misma forma que se sacaba cualquier DFT. Por supuesto, se puede conmutar entre mostrar el módulo o la fase de la respuesta en frecuencia, que sólo cambia las operaciones que se hacen con las muestras tomadas por el ADC.

Como problemas a destacar en el desarrollo de esta mejora, tuvimos que adaptar los senos que salen por el ADC y excitan al circuito, ya que por el ADC no se pueden generar señales bipolares: esto se soluciona con un filtro paso bajo que elimine la continua. Después nos dimos cuenta de que los valores negativos de las muestras del seno no casaban bien con los valores escalados del DAC. Haciendo un estudio de estos valores, concluimos en sumar un offset de 2500 para hacer que lo que los senos varíen entre 3319 y 1681, de forma que al salir por el DAC y pasar por el filtro paso alto queda una sinusoide de aproximadamente 0.5 Vpp. Elegimos este valor para que las posibles ganancias de los filtros a estudiar no se pasen de los niveles medibles por el ADC.

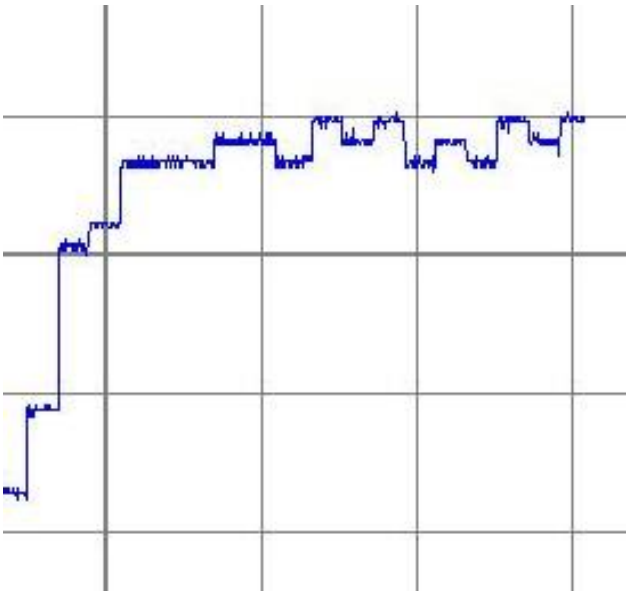
Como ejemplo de resultado de esta mejora mostramos el estudio de un filtro paso alto cuya frecuencia de corte es 200Hz:



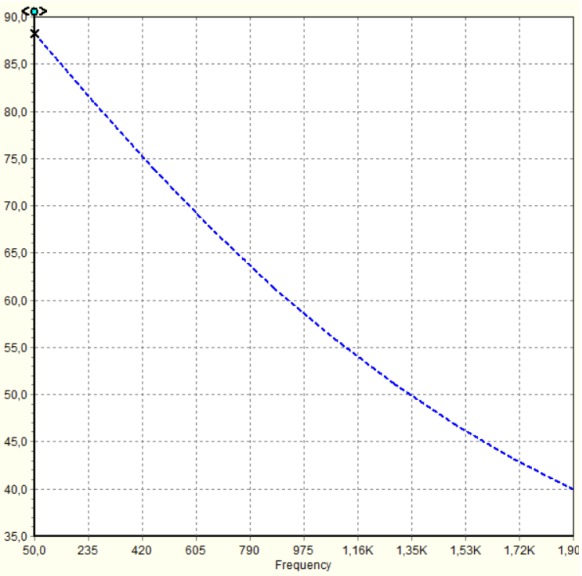
Esquemático del circuito a evaluar



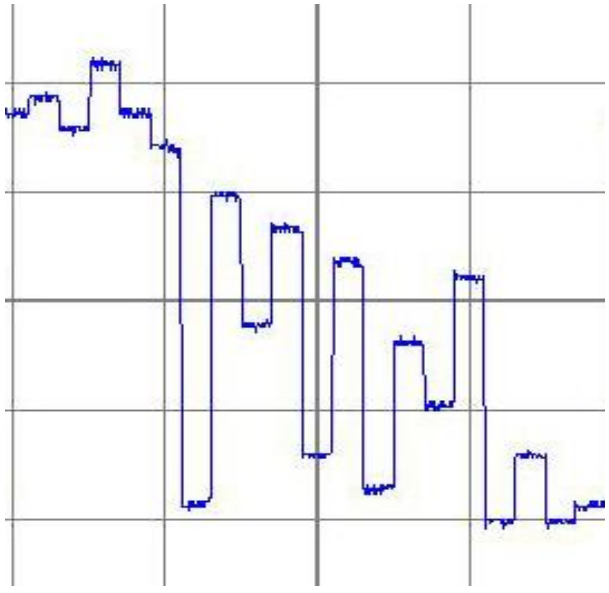
Simulación de la respuesta en módulo



Respuesta experimental en módulo



Simulación de la respuesta en fase



Simulación de la respuesta en fase

2.3 Afinador

Trabajando de forma similar a un afinador real de instrumentos, esta funcionalidad de nuestro proyecto obtiene de la DFT calculada el pico de frecuencia con la mayor potencia, para después hacer una búsqueda más exhaustiva y centrada en esa frecuencia de forma que obtenemos el valor máximo de frecuencia con un error máximo de 2.5Hz (dentro del rango de frecuencias analizables del proyecto).

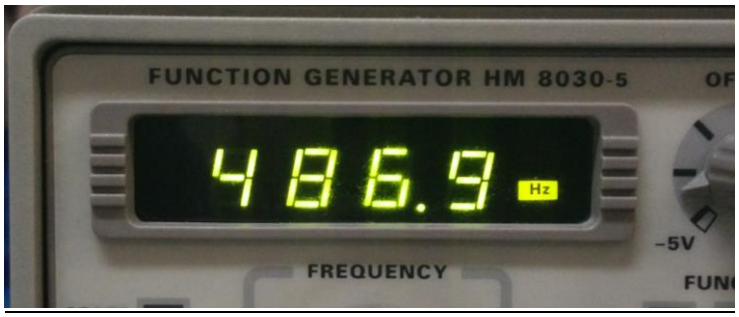
Seleccionable gracias a nuestra interfaz de usuario para cambiar entre mejoras, el programa calcula automáticamente el pico en frecuencia cada segundo y lo muestra por la segunda línea del LCD, de forma que se puede ver como varía esta frecuencia si excitamos el DAC con una señal y vamos cambiando su frecuencia.

En cuanto a cómo funciona esta mejora en más detalle, dividimos el proceso de cálculo del pico en frecuencia y su muestra por el LCD en varios pasos que se ejecutan cada segundo:

1. Cálculo de la DFT (funcionalidad básica del proyecto) y obtención del pico de frecuencia. Este número de frecuencia pertenecerá al array principal de frecuencias que se muestrean normalmente, es decir: 50, 100, 200, 300… 1900.
2. Se llama a la función cambiaRangoExploracion, que establecerá el rango de frecuencias en los que se calculará la próxima DFT: centrado en el pico calculado en el paso anterior, desde 50Hz antes hasta 50Hz después.
3. En este nuevo rango, calculamos la DFT de 5Hz en 5Hz (volviendo a tener 20 componentes). Obtenemos el pico de frecuencia definitivo, ahora con mayor precisión.
4. Nueva llamada a la función cambiaRangoExploracion, que esta vez restablecerá todas las variables para que 1 segundo después se vuelva al paso 1. Se muestra por el LCD el pico obtenido en el paso 3, llamando a una función que adapta el formato de entero a chars de la forma “XXXX Hz”.

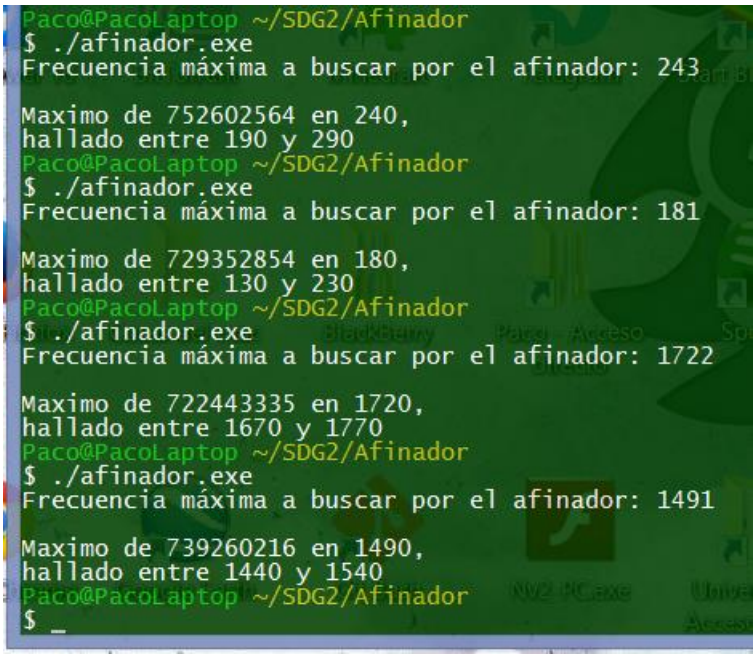
Es notable en esta mejora en particular la dificultad de los problemas que tuvimos a la hora de su depuración de errores y puesta en funcionamiento. En primer lugar, una vez esbozamos un primer algoritmo que calculaba correctamente frecuencias con un seno generado internamente y en nuestro ordenador, al probarlo en el laboratorio sobrecargaba de operaciones al microcontrolador y no funcionaba ninguna otra mejora. Probamos varias formas distintas de hacerlo, intentando que fuesen más eficientes, pero todas acabaron en fracaso, así que decidimos usar el primer código y parar la interrupción, que recoge muestras y calcula la DFT, a la hora de hacer los cálculos pertinentes: actualizar el rango de frecuencias en cambiaRangoExploracion y convertir el número de la frecuencia a chars a sacar por el LCD. Una vez se terminan estos cálculos se vuelve a activar la interrupción.

Aparte de la dificultad del código que cambia los rangos de exploración en frecuencia, otro problema que tuvimos que abordar fue la visualización de estos picos de frecuencia por el LCD, teniendo que hacer una función independiente (llamada stringFrecuencia) que dado un número devuelve los chars “XXXX Hz”, adaptados para usarlos en el LCD. Consigue esto obteniendo los millares, centenas, decenas y unidades del número pasado como parámetro, convirtiendo cada uno a char y colocándolos en posiciones de memoria consecutivas, seguidas de “ Hz”. Devuelve un puntero al primer char.



Podemos comprobar el buen funcionamiento del afinador introduciendo un seno de 486,9Hz y viendo cómo la frecuencia detectada es 485Hz

Otra dificultad de esta mejora ha sido tener que generar un array de 800 muestras del periodo de un seno, de forma que si se usan esas muestras una a una se simula una sinusoide a 5Hz, fundamental para la exploración en una ventana de frecuencias más concretas. Tuvimos que generar este array mediante Matlab e incorporarlo a nuestro programa. Por comodidad, hemos cambiado todas las variables necesarias para usar este array para todo el proyecto, de forma que prescindimos del array del seno a 10Hz proporcionado para el Hito 1.



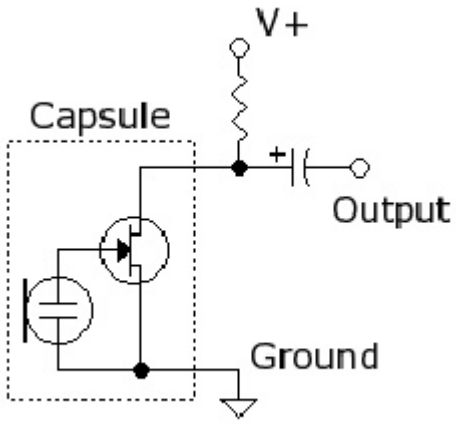
Así mismo realizamos pruebas en ordenador local para comprobar el correcto funcionamiento de este módulo antes de probarlo en el laboratorio.

El código utilizado se muestra en el anexo al final del documento.

2.4 Micrófono

Como entrada auxiliar para calcular la DFT añadimos un micrófono a nuestro sistema, de esta forma somos capaces de analizar las frecuencias de la voz.

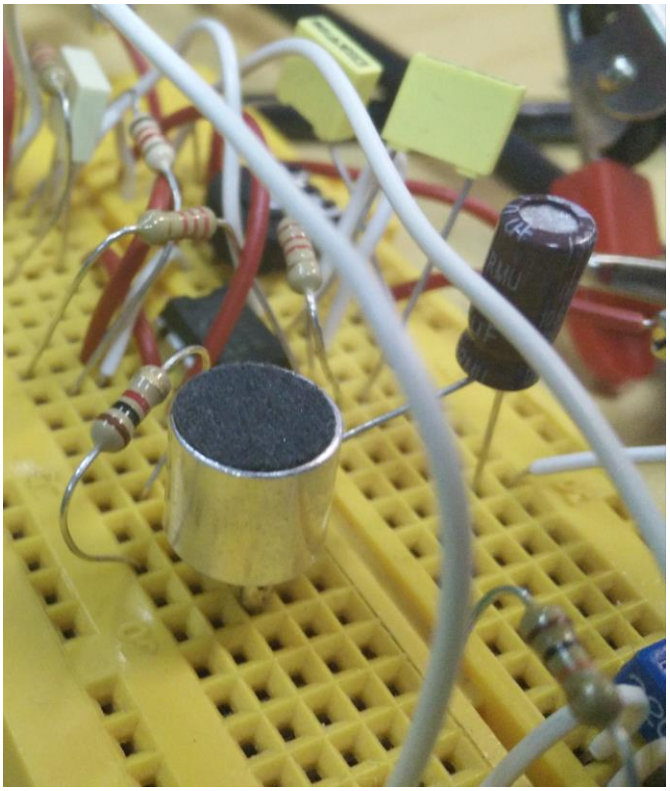
Para conseguir un correcto funcionamiento del mismo añadimos una resistencia de pull up conectada a Vcc y un condensador para eliminar la componente continua de la señal de salida del micrófono. El esquema eléctrico es el mostrado a continuación:



Los valores óptimos de resistencia y condensador son 1kΩ y 10µF respectivamente.

Para poder conmutar entre entrada del generador de funciones y entrada del micrófono utilizamos un multiplexor conectando dichos dispositivos a 2 canales de entrada y utilizando el puerto 5 podemos conmutar entre ambas entradas.

Para esta mejora no fue necesaria ninguna modificación adicional en el código al tratarse de un esquema puramente de hardware e independiente de la plataforma ENT2005CF.



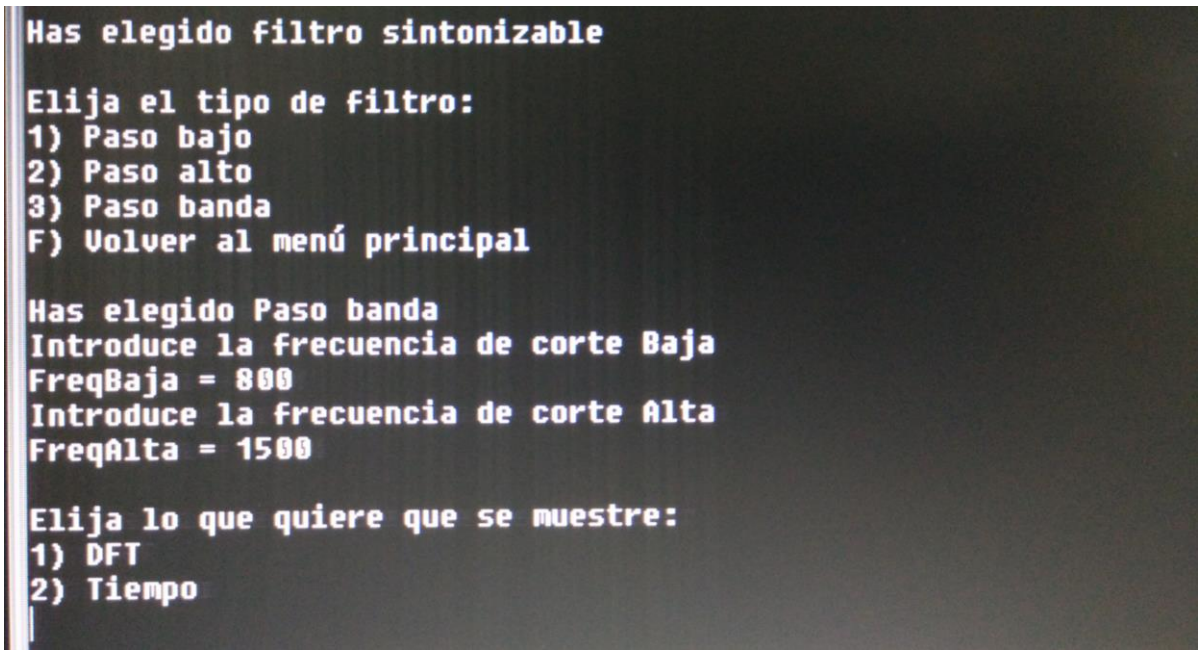
El código utilizado se muestra en el anexo al final del documento.

2.5 Sistema de filtrado digital y reconstrucción de la señal

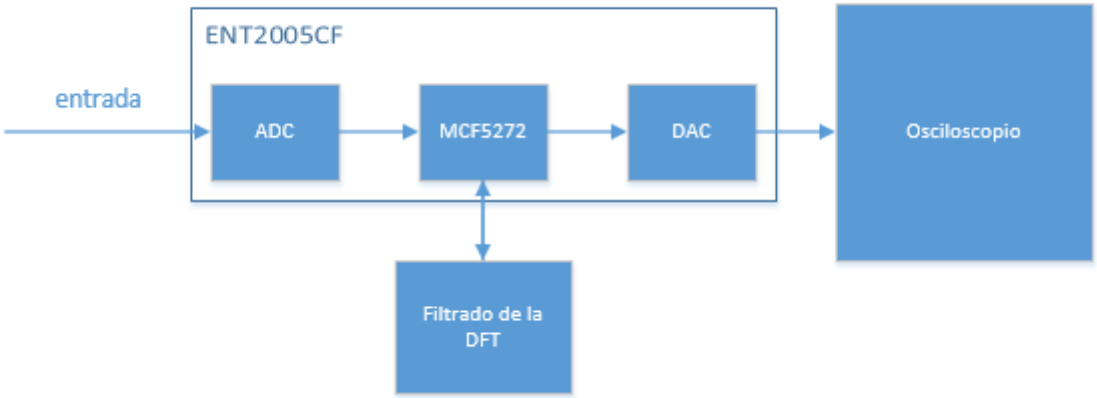
Se trata de una mejora propia que podemos dividir en dos grandes subapartados, el código utilizado se muestra en el anexo al final del documento bien documentado.

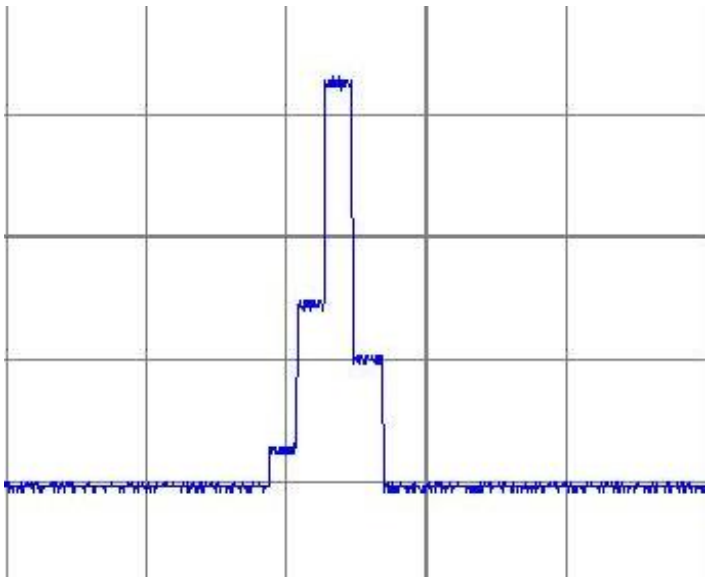
2.5.1 Filtrado sintonizable digital de la señal

Esta mejora nos permite filtrar las frecuencias que queramos de nuestra señal de entrada a la plataforma. Para ello podemos elegir si queremos un filtro paso alto, paso bajo o paso banda especificando las frecuencias de corte todo a través de una interfaz muy sencilla e intuitiva.



A continuación se muestra el esquema software de dicha mejora:





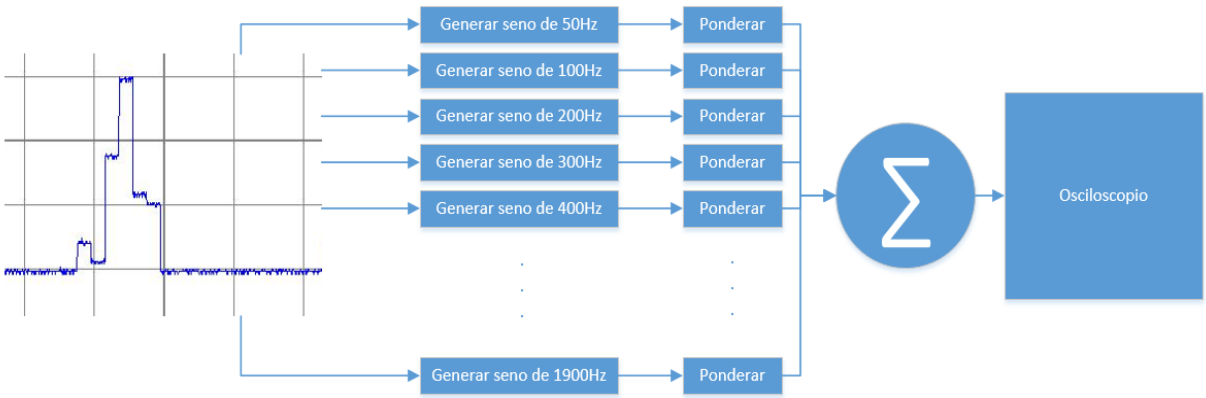
Podemos apreciar la DFT de la señal de entrada filtrada paso banda entre 600 y 900Hz

Tras elegir el tipo de filtro y las frecuencias de corte nos permite elegir si queremos mostrar la DFT filtrada a tiempo real o la señal filtrada en función del tiempo

2.5.2 Reconstrucción de la señal filtrada, DFT inversa

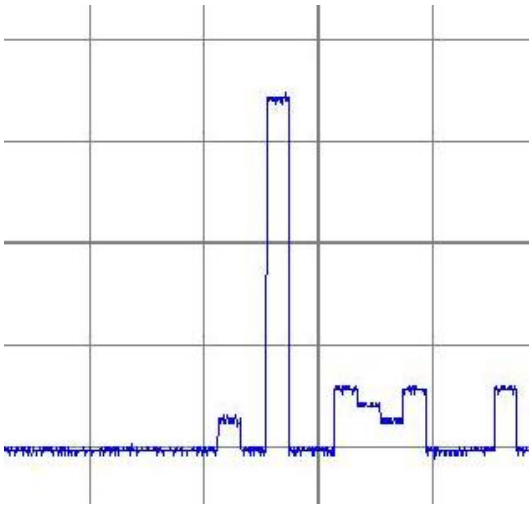
Ya que teníamos la DFT filtrada pensamos en implementar la DFT inversa, reconstruir la señal a partir de su DFT. El procedimiento software consiste en generar senos de cada frecuencia, normalizar su amplitud según los pesos de la DFT y sumar todas las componentes formando la señal de salida.

A continuación detallamos su esquema software.

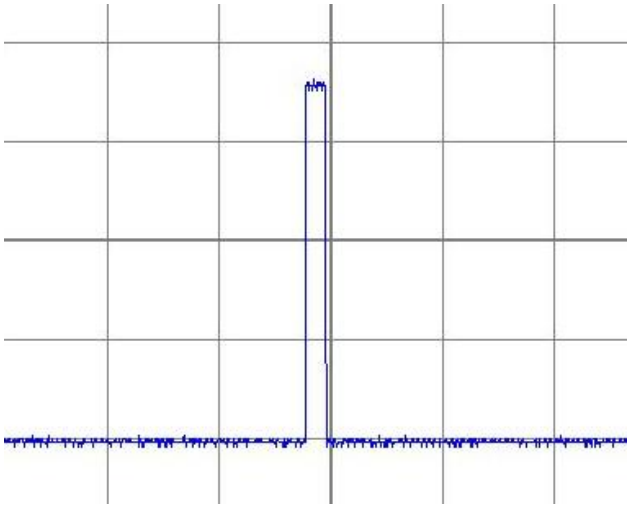


Debido a la frecuencia de la interrupción (4kHz) y al teorema de Nyquist por el que la máxima frecuencia a generar es la mitad de lafrecuencia de la interrupción, la máxima frecuencia que podríamos generar sería teóricamente 2kHz lo cual no es un problema en nuestro sistema pues la DFT tiene una máxima frecuencia de 1900Hz.

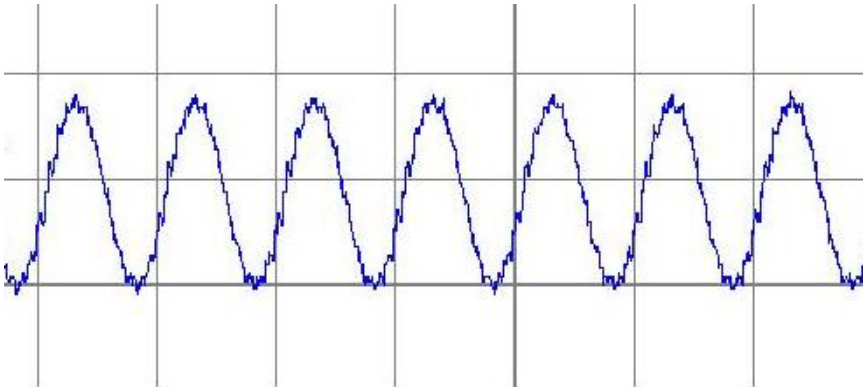
Para comprobar el buen funcionamiento de esta mejora introdujimos una señal de entrada con varias componentes a la plataforma ENT2005CF, filtramos una sola frecuencia (1000Hz) y posteriormente visualizamos la DFT inversa en el osciloscopio:



DFT de la señal de entrada

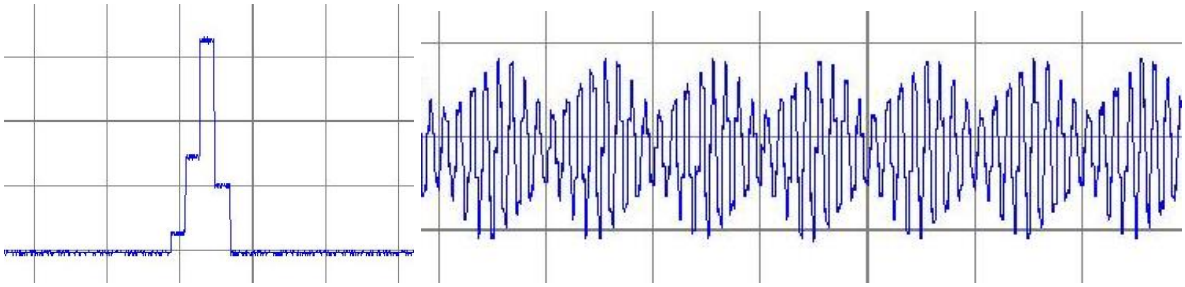


DFT dejando pasar 1000Hz



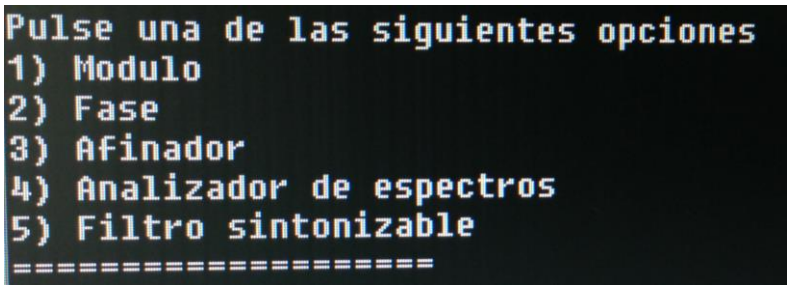
Señal a 1000Hz reconstruida

Para el caso de reconstruir más de una componente no hay problema, a continuación mostramos el ejemplo de un filtro paso banda de 600Hz a 900Hz.



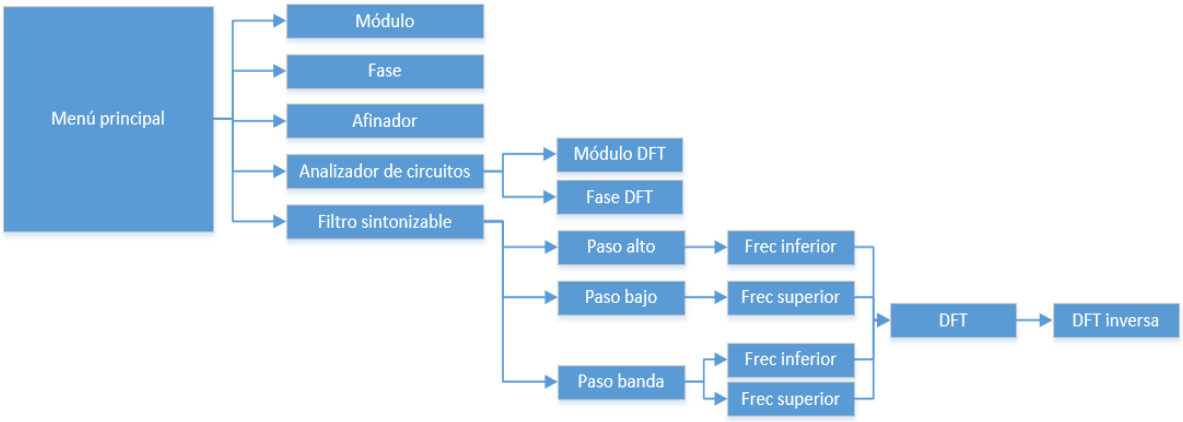
2.6 Interfaz

Con el objetivo de que fuese fácil manejar todas las mejoras que implementamos y poder pasar de una a otra implementamos una interfaz sencilla e intuitiva a través del teclado matricial, LCD y monitor.



Nada más iniciar el programa nos permite elegir qué mejor mostrar de entre todas las posibles. Al seleccionar la mejora elegida se desplegaría otro menú con las opciones particulares de cada mejora y en cualquier momento pulsando la tecla 'F' volveríamos al menú principal.

El diagrama de navegación de nuestro sistema se detalla a continuación:



2.6.1 Teclado matricial

Para seleccionar la mejora, introducir datos como las frecuencias de los filtros, volver al menú principal… Implementamos el teclado matricial que usa los puertos del 0 al 3.

Basándonos en el código proporcionado teclado.c, añadimos en nuestro código la función teclado() que nos devuelve en formato char la tecla pulsada. Tuvimos que usar una máscara para que no hubiese conflictos entre los 16 puertos.

En aquellos casos en los que necesitamos un número en vez de carácter (como por ejemplo al elegir las frecuencias de los filtros) hicimos la función strToInt(char str) que nos devuelve el entero.

El código utilizado se muestra en el anexo al final del documento.

2.6.2 LCD

Implementamos la pantalla LCD con el objetivo de poder visualizar datos de salida de la plataforma. Empezamos mostrando un mensaje estático en la primera línea nada más iniciar el programa

para posteriormente mostrar mensajes estáticos en ambas líneas y por último conseguimos un LCD dinámico donde muestra la mejora elegida, la submejora o datos concretos como en el caso del afinador.

2.7 Optimización

Al ir añadiendo nuevas mejoras y funcionalidades a nuestro sistema llegó un momento en el que no daba tiempo a realizar todas las operaciones en cada interrupción de 0,25ms por lo que tuvimos que optimizar todo el código posible.

Dicha optimización la realizamos principalmente sustituyendo las divisiones y multiplicaciones de múltiplos de 2 por desplazamientos lógicos, trabajar con int en vez de doublé cuando no necesitemos demasiada precisión, reducir el número de bucles o sustituirlos en algunos casos por múltiples ifs, crear nuevas estructuras…

Tras todas las optimizaciones conseguimos que nuestro sistema funcionase de un modo muy fluido a 4kHz e incluso podríamos aumentar la frecuencia de la interrupción lo que nos proporciona un pequeño margen de mejora de funcionamiento.

2.8 Divide y vencerás

El algoritmo "Divide Y Vencerás" (DYV de ahora en adelante) trata de dividir un problema en varios subproblemas, y cada uno de esos subproblemas en otros más simples todavía, repitiendo el proceso hasta que se llega a una solución directa del primer problema a resolver.

En nuestro caso, aplicamos el algoritmo DYV para la conversión a decibelios de los valores del módulo y fase de la DFT, haciendo una búsqueda binaria de los valores finales a sacar por el osciloscopio. En ambos casos partimos de 32 posibles valores, que dependiendo de ciertas condiciones se van reduciendo a 16, 8, 4…. Hasta que llegamos a una única posibilidad, convirtiéndose en el valor definitivo.

En el caso de la conversión a decibelios del módulo de la DFT, el algoritmo DYV ha consistido en asignar los valores calculados a los valores escalados dados en la tabla del enunciado mediante 5 niveles de sentencias if-else:

- Si el valor es mayor que el nivel nº15 del array de umbrales, se sigue buscando en los 16 niveles superiores; si no, se busca en los 16 inferiores
- En el nuevo rango de búsqueda (16 niveles): si el valor es mayor que el nivel nº7 del array de los umbrales, se sigue buscando en los 8 niveles superiores; si no, se busca en los 8 inferiores.
- …
- Último rango de búsqueda (2 niveles): se coge el nivel más cercano, y termina la búsqueda binaria.

Paralelamente, en el caso del cálculo de la fase también se va dividiendo el número de posibles valores, hasta llegar al resultado definitivo. Sin embargo, no se usa una estrategia de tablas como en el módulo, sino una basada en cuadrantes del plano real-complejo en una primera división de los problemas (aquí se dividen los problemas en 4 subproblemas, por lo que no es una búsqueda binaria propiamente dicha en este primer paso). Una vez encontrado el cuadrante donde se sitúa el valor complejo de la DFT, vamos reduciendo posibilidades al usar la definición de la tangente.

- Si el valor complejo se encuentra en el cuadrante X, descartamos los otros tres. Dividimos los 32 casos posibles entre 4, quedando 8 posibilidades.
- Situándonos en ese cuadrante y usándolo de referencia, el valor complejo tendrá una fase mayor de 45º si la parte imaginaria es mayor que la real; si no, será menor de 45º. Quedan 4 posibilidades.
- Usando la definición de la tangente, se dividen los casos dos veces más, obteniendo el valor definitivo.

Esta mejora nos ha ayudado mucho para todo el proyecto en general, ya que el número de operaciones (y por tanto, el tiempo de ejecución) era un aspecto crítico para el funcionamiento de las demás mejoras. Anteriormente, para la conversión a decibelios, se necesitaban 32 iteraciones en el peor de los casos, y era imposible tener todas las mejoras funcionando al mismo tiempo.

El principal problema del algoritmo DYV ha sido la complejidad del código en términos de trabajo, pero para el programa es un código muy simple, al pasar de 32 operaciones a sólamente 5. Previamente probado el correcto funcionamiento del algoritmo en nuestros ordenadores, hemos adaptado estas nuevas formas de cálculo al código principal del proyecto en las funciones conversion_dB, y para la fase una comprobación previa de cuadrantes y después la función estimaFase0a90.

3 Principales problemas encontrados

A lo largo de la implementación de las nuevas mejoras fueron surgiendo diversos problemas, a continuación explicamos los principales y cómo los solucionamos:

- El bit de rampa lo pusimos en el puerto 0 con lo que nada más implementar el teclado dejó de funcionar. Esto lo solucionamos pasando el bit de rampa al puerto 4 y poniendo las correspondientes máscaras al teclado.

- Con el afinador tuvimos el principal problema de que saturábamos de operaciones al microcontrolador así que para realizar estos cálculos paramos la interrupción y una vez calculado todo la reanudamos.

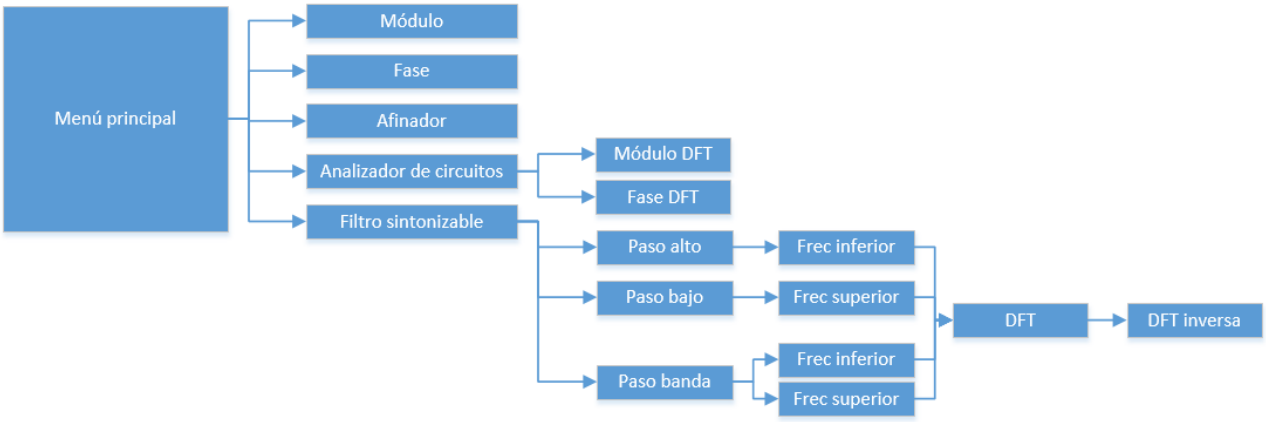
- En cuanto al analizador de circuitos tuvimos que poner un filtro paso alto a 20Hz para eliminar la continua que introducíamos a los circuitos a evaluar, de esta forma introducimos sólo las componentes alternas obteniendo medidas mucho más precisas.

- Cuando calculamos la DFT inversa teníamos un problema de saturación del DAC cuando sumábamos varias componentes, por ello tuvimos que normalizar la salida a la máxima amplitud del DAC ponderando cada componente por su amplitud de la DFT.

- Al implementar tantas mejoras no daba tiempo a que se realizasen todas las operaciones en cada interrupción, la solución a este problema ha sido detallada en el apartado de la mejora "Optimización".

4 Manual de usuario

En todo momento hemos intentado que todo nuestro sistema sea intuitivo y sencillo de utilizar y para ello implementamos la interfaz cuyo esquema se muestra a continuación:



En cada mejora elegida se despliega un submenú en el monitor donde aparecen las opciones propias de cada mejora (detalladas en el esquema superior). La entrada de datos se realiza con el teclado matricial y la salida entre monitor del ordenador y LCD.

Más detalles sobre la utilización de nuestro sistema se han explicado en el apartado de la mejora "Interfaz".

5 Bibliografía

DATASHEETS DE COMPONTENTES

Para saber cómo conectar los componentes analógicos (sobre todo los transistores) recurrimos a las datasheets de los mismos (www.alldatasheet.com). También las utilizamos para conocer las entradas/salidas del entrenador.

TEORÍA DE OTRAS ASIGNATURAS

Usamos los apuntes de otras asignaturas como ayuda adicional, sobre todo de Teoría Digital de Señales para la mejora que hace la DFT inversa.

Después, para el diseño de circuitos usamos los apuntes de Circuitos Electrónicos, ya que necesitamos filtros para probar el analizador de circuitos.

TUTORIALES DE LA ASIGNATURA

Fundamentales para LCD y teclado matricial, usamos sus respetivos tutoriales para añadir el uso de estos accesorios en nuestro proyecto

USO DEL MICRÓFONO ELECTRET

Gracias a manuales y tutoriales encontrados por internet

http://electronicayciencia.blogspot.com.es/2010/06/utilizar-un-microfono-electret.html

6 ANEXO I: Código del programa del proyecto final

```
//Francisco García de la Corte
//Carlos Santos Rancaño
#include "m5272.h"
#include "m5272lib.c"
#include "m5272adc_dac(mod).c"
#include "m5272gpio.c"

//--------------Interfaz---------------
typedef enum {INICIO=-1, MODULO=0, FASE=1, AFINADOR=2, ANALIZADOR_HW=3, FILTRO_SINTONIZABLE=4} ESTADO;
ESTADO estado = INICIO;


//Constantes para la interrupción ----------------------------------------------------------------------------
#define FONDO_ESCALA 0xFFF       // Valor de lectura mÃ¡xima del ADC
#define V_MAX 5
#define V_BASE 0x40                          // DirecciÃ³n de inicio de los vectores de interrupciÃ³n
#define DIR_VTMR0 4*(V_BASE+5)                       // DirecciÃ³n del vector de TMR0
#define FREC_INT 4000                                // Frec. de interr. TMR0 = 4000 Hz (cada 0.25ms)
#define PRESCALADO 2                                         //parece que el preescalado no hace nada
#define CNT_INT1 MCF_CLK/(FREC_INT*PRESCALADO*16)      // Valor de precarga del temporizador de interrupciones TRR0
#if CNT_INT1>0xFFFF
#error PRESCALADO demasiado pequeÃ±o para esa frecuencia (CNT_INT1>0xFFFF)
#endif
#define BORRA_REF 0x0002                     // Valor de borrado de interr. pendientes de tout1 para TER0
#define FONDO_ESCALA 0xFFF       // Valor de lectura mÃ¡xima del ADC
#define V_MAX 5
volatile ULONG cont_retardo;
```

```
//Constantes y variables para el cálculo de la DFT y las mejoras ----------------------------------------------
#define NUM_MUESTRAS_PERIODO_5HZ 800 //nueva dimensión para el array de muestras de la sinusoide, ahora a 5Hz.
#define ESC_ADC 819
#define N 80
#define DESESCALADO 20 //Al cuadrado, en realidad es 10
//Orden de la DFT: 20 muestras de la TF
#define N_FRECS 20
//32 niveles para los umbrales
#define NIVELES 32

static int umbrales[NIVELES]={63095, 84924, 114304, 153849, 207075, 278715, 375140, 504923,
  679607, 914724, 1231182, 1657123, 2230422, 3002059 , 4040653 , 5438559 , 7320085,
  9852544 ,13261133, 17848959, 24023991, 32335339, 43522083, 58578997, 78845006,
  106122250, 142836338, 192252044, 258763624, 348285572, 468778561, 686492401};

static int escalado[NIVELES]={122, 245, 368, 491, 614, 737, 860, 983, 1105, 1228,
  1351, 1474, 1597, 1720, 1843, 1966, 2088, 2211, 2334, 2457, 2580,
  2703, 2826, 2949, 3072, 3194, 3317, 3440, 3563, 3686, 3809, 3932};

static int sin5Hz[NUM_MUESTRAS_PERIODO_5HZ] = {0, 6, 13, 19, 26, 32, 39, 45, 51, 58, 64, 71, 77, 83, 90, 96, 103,
  109, 115, 122, 128, 134, 141, 147, 153, 160, 166, 172, 179, 185, 191, 197, 204, 210,
  216, 222, 228, 235, 241, 247, 253, 259, 265, 271, 277, 283, 289, 296, 301, 307, 313,
  319, 325, 331, 337, 343, 349, 355, 360, 366, 372, 378, 383, 389, 395, 400, 406, 411,
  417, 422, 428, 433, 439, 444, 450, 455, 460, 466, 471, 476, 481, 487, 492, 497, 502,
  507, 512, 517, 522, 527, 532, 537, 542, 546, 551, 556, 561, 565, 570, 575, 579, 584,
  588, 593, 597, 601, 606, 610, 614, 619, 623, 627, 631, 635, 639, 643, 647, 651, 655,
  659, 663, 666, 670, 674, 677, 681, 685, 688, 692, 695, 698, 702, 705, 708, 711, 715,
  718, 721, 724, 727, 730, 733, 735, 738, 741, 744, 746, 749, 752, 754, 757, 759, 761,
  764, 766, 768, 771, 773, 775, 777, 779, 781, 783, 785, 786, 788, 790, 792, 793, 795,
  796, 798, 799, 801, 802, 803, 804, 806, 807, 808, 809, 810, 811, 812, 813, 813, 814,
  815, 815, 816, 816, 817, 817, 818, 818, 818, 819, 819, 819, 819, 819, 819, 819, 819,
```

819, 818, 818, 818, 817, 817, 816, 816, 815, 815, 814, 813, 813, 812, 811, 810, 809,
808, 807, 806, 804, 803, 802, 801, 799, 798, 796, 795, 793, 792, 790, 788, 786, 785,
783, 781, 779, 777, 775, 773, 771, 768, 766, 764, 761, 759, 757, 754, 752, 749, 746,
744, 741, 738, 735, 733, 730, 727, 724, 721, 718, 715, 711, 708, 705, 702, 698, 695,
692, 688, 685, 681, 677, 674, 670, 666, 663, 659, 655, 651, 647, 643, 639, 635, 631,
627, 623, 619, 614, 610, 606, 601, 597, 593, 588, 584, 579, 575, 570, 565, 561, 556,
551, 546, 542, 537, 532, 527, 522, 517, 512, 507, 502, 497, 492, 487, 481, 476, 471,
466, 460, 455, 450, 444, 439, 433, 428, 422, 417, 411, 406, 400, 395, 389, 383, 378,
372, 366, 360, 355, 349, 343, 337, 331, 325, 319, 313, 307, 301, 296, 289, 283, 277,
271, 265, 259, 253, 247, 241, 235, 228, 222, 216, 210, 204, 197, 191, 185, 179, 172,
166, 160, 153, 147, 141, 134, 128, 122, 115, 109, 103, 96, 90, 83, 77, 71, 64, 58, 51,
45, 39, 32, 26, 19, 13, 6, 0, -6, -13, -19, -26, -32, -39, -45, -51, -58, -64, -71, -77,
-83, -90, -96, -103, -109, -115, -122, -128, -134, -141, -147, -153, -160, -166, -172,
-179, -185, -191, -197, -204, -210, -216, -222, -228, -235, -241, -247, -253, -259, -265,
-271, -277, -283, -289, -296, -301, -307, -313, -319, -325, -331, -337, -343, -349, -355,
-360, -366, -372, -378, -383, -389, -395, -400, -406, -411, -417, -422, -428, -433, -439,
-444, -450, -455, -460, -466, -471, -476, -481, -487, -492, -497, -502, -507, -512, -517,
-522, -527, -532, -537, -542, -546, -551, -556, -561, -565, -570, -575, -579, -584, -588,
-593, -597, -601, -606, -610, -614, -619, -623, -627, -631, -635, -639, -643, -647, -651,
-655, -659, -663, -666, -670, -674, -677, -681, -685, -688, -692, -695, -698, -702, -705,
-708, -711, -715, -718, -721, -724, -727, -730, -733, -735, -738, -741, -744, -746, -749,
-752, -754, -757, -759, -761, -764, -766, -768, -771, -773, -775, -777, -779, -781, -783,
-785, -786, -788, -790, -792, -793, -795, -796, -798, -799, -801, -802, -803, -804, -806,
-807, -808, -809, -810, -811, -812, -813, -813, -814, -815, -815, -816, -816, -817, -817,
-818, -818, -818, -819, -819, -819, -819, -819, -819, -819, -819, -819, -818, -818, -818,
-817, -817, -816, -816, -815, -815, -814, -813, -813, -812, -811, -810, -809, -808, -807,
-806, -804, -803, -802, -801, -799, -798, -796, -795, -793, -792, -790, -788, -786, -785,
-783, -781, -779, -777, -775, -773, -771, -768, -766, -764, -761, -759, -757, -754, -752,
-749, -746, -744, -741, -738, -735, -733, -730, -727, -724, -721, -718, -715, -711, -708,
-705, -702, -698, -695, -692, -688, -685, -681, -677, -674, -670, -666, -663, -659, -655,
-651, -647, -643, -639, -635, -631, -627, -623, -619, -614, -610, -606, -601, -597, -593,
-588, -584, -579, -575, -570, -565, -561, -556, -551, -546, -542, -537, -532, -527, -522,

```
  -517, -512, -507, -502, -497, -492, -487, -481, -476, -471, -466, -460, -455, -450, -444,
  -439, -433, -428, -422, -417, -411, -406, -400, -395, -389, -383, -378, -372, -366, -360,
  -355, -349, -343, -337, -331, -325, -319, -313, -307, -301, -296, -289, -283, -277, -271,
  -265, -259, -253, -247, -241, -235, -228, -222, -216, -210, -204, -197, -191, -185, -179,
  -172, -166, -160, -153, -147, -141, -134, -128, -122, -115, -109, -103, -96, -90, -83,
  -77, -71, -64, -58, -51, -45, -39, -32, -26, -19, -13, -6};

int muestra;

int dft[N_FRECS]; // = dftCos^2 + dftSin^2
int dftCos[N_FRECS]; //parte real
int dftSin[N_FRECS]; //parte imaginaria

int i, n, k, j; //iteradores para bucles
int cont160Inter;
int calculo; //Si calculo o no la DFT en función de si me quedan por sacar por pantalla

static int saltoPrincipal[N_FRECS] = {10, 20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300, 320, 340, 360, 380};
int salto[N_FRECS];

int *pSin[N_FRECS];
int *pCos[N_FRECS];
```

```
//--------Analizador de circuitos---------
int indice = 0;
int cont1600ints = 0;
int barrido = 0; //contador de frecuencias: aumenta en 1 con cada dft calculada
int mostrarHw = 0; //a activar cuando se ha calculado
int Hw[N_FRECS];

int modulo_fase = 0; //si es uno, se visualizará la fase

//--------Sistema de filtrado digital y reconstrucción de la señal-------
int freqBaja=0;
int freqAlta=0;

int salidaAnalogica=0;

int DFTlista=0;
int nFiltro=0;
int coef[N_FRECS];
int sumaCoef=0;
int salida[N];

int salidaAnalogicaCalculada = 0;

int *indicesSenos[N_FRECS];

//------------Teclado------------
#define NUM_FILAS 4
#define NUM_COLS 4
#define EXCIT 1
UWORD pSal=0x0000;
char opcion;
```

```
char tecla;

//---------------LCD-------------
#include "m5272lcd(mod).c"

//-------------Afinador----------
int freqAntes=0;
int freqMax=0;
char *maximoString;
int picoMax=0;
int explorar = 0; //1 si voy de 100 en 100 (como siempre sin el afinador) y 2 para la búsqueda precisa
int contador1seg=0; //para que recalcule solo la frecuencia maxima cada 4000 interrupciones (25 dfts, contando las que se desprecian)
```

```
//PROGRAMA----------------------------------------------------------------------------------------------------

//-----------------------------------------------------
// char teclado(void)
//
// Descripción:
// Función basada en el tutorial teclado.c. Excita el teclado en busca
// de respuestas para obtener las teclas pulsadas por el usuario.
// Devuelve un char correspondiente a la tecla pulsada.
//-----------------------------------------------------
char teclado(void) {
  BYTE fila, columna, fila_mask;
  static char teclas[4][4] = {{"123C"},
                              {"456D"},
                              {"789E"},
                              {"A0BF"}};
  // Bucle de exploración del teclado
  while(TRUE){
    // Excitamos una columna
    for(columna = NUM_COLS - 1; columna >= 0; columna--){
      pSal = pSal & 0xFFF0;
      pSal= (EXCIT << columna) | pSal;
      set16_puertoS(pSal);          // Se envía la excitación de columna
      retardo(1150);                              // Esperamos respuesta de optoacopladores
      // Exploramos las filas en busca de respuesta
      for(fila = NUM_FILAS - 1; fila >= 0; fila--){
        fila_mask = EXCIT << fila;                // Máscara para leer el bit de la fila actual
        if(lee16_puertoE() & fila_mask){          // Si encuentra tecla pulsada,
          while(lee16_puertoE() & fila_mask);     //   Esperamos a que se suelte
          retardo(1150);                          //   Retardo antirrebotes
          return teclas[fila][columna];           //   Devolvemos la tecla pulsada
```

```
            }
          } // Siguiente columna
        }// Exploración finalizada sin encontrar una tecla pulsada
      }// Reiniciamos exploración
    }
```

```
//-------------------------------------------------------
// void sacaLCD(char* sacar, int linea)
//
// Descripción:
// Función basada en el tutorial lcd.c. Escribe por el LCD el array de
// chars pasado como parámetro, refrescando la pantalla y colocando el
// cursor al principio. Además, se puede seleccionar en cuál de las dos
// líneas del LCD se desea escribir mediante el segundo parámetro.
//-------------------------------------------------------
void sacaLCD(char* sacar, int linea) {

  if (linea==1) {
   LCD_inst(CLR_DISP);                    // Limpiamos display
   LCD_inst(LIN_1LCD); // Movemos el cursor a la 1ª línea

  }
  if (linea==2)
   LCD_inst(LIN_2LCD); // Movemos el cursor a la 2ª línea

  while(*sacar){  // Imprime "HOLA" en el display
    LCD_dato(*sacar++);// carácter a carácter
  }
  //No se borrará el mensaje hasta que llamemos a esta función de nuevo
}
```

```
//-----------------------------------------------------
// int strToInt (char str)
//
// Descripción:
// Simple rutina auxiliar para nuestro sistema de filtrado digital.
// Se llama a esta función para pasar a enteros los chars intruducidos
// desde el teclado.
// Devuelve un char equivalente al int pasado como parámetro
//-----------------------------------------------------
int strToInt (char str) {
  switch(str) {
                        case '0': return 0;
                        case '1': return 1;
                        case '2': return 2;
                        case '3': return 3;
                        case '4': return 4;
                        case '5': return 5;
                        case '6': return 6;
                        case '7': return 7;
                        case '8': return 8;
                        case '9': return 9;
                        default: break;
              }
  return 0;
}
```

```
//-------------------------------------------------------
// char * stringFrecuencia(int f)
//
// Descripción:
// Esta función tiene como objetivo adaptar los números de frecuencias hallados
// por el afinador a un formato de array de chars "XXXX Hz", de forma que se
// pueda sacar por el LCD. Halla los millares, centenas, decenas y unidades del
// número, y los guarda en posiciones de memoria consecutivas seguidas de " Hz".
// Devuelve un puntero a la primera posición en memoria de estos chars, hallados
// a partir del número pasado como parámetro.
//-------------------------------------------------------
char * stringFrecuencia(int f) { //Devuelve un puntero a un array de 7 chars
  //Generar string a sacar:
  static char freq[7];
  i=0;
  k=0;
  //Cada if a partir de ahora, obtiene el numero a la izquierda,
  //lo quita y deja los numeros de la derecha
  if (f>=1000) {
          freq[k] = '1';
          f -=1000;

  } else {
          freq[k] = ' ';
  }
  k++;

  if (f>=100) {
          while ((f-100) >= 0) {
                  f-=100;
                  i++;
```

```
		}
		switch(i) {
			case 0: freq[k] = '0'; break;
			case 1: freq[k] = '1'; break;
			case 2: freq[k] = '2'; break;
			case 3: freq[k] = '3'; break;
			case 4: freq[k] = '4'; break;
			case 5: freq[k] = '5'; break;
			case 6: freq[k] = '6'; break;
			case 7: freq[k] = '7'; break;
			case 8: freq[k] = '8'; break;
			case 9: freq[k] = '9'; break;
			default: break;
		}

	}
	if(i == 0 ){
		if (freq[k-1]==' '){
			freq[k] = ' ';
		} else {
			freq[k] = '0';
		}
	}
	i=0;
	k++;
	if (f>=10) {
		while ((f-10) >= 0) {
			f-=10;
			i++;
		}
		switch(i) {
			case 0: freq[k] = '0'; break;
```

```
                case 1: freq[k] = '1'; break;
                case 2: freq[k] = '2'; break;
                case 3: freq[k] = '3'; break;
                case 4: freq[k] = '4'; break;
                case 5: freq[k] = '5'; break;
                case 6: freq[k] = '6'; break;
                case 7: freq[k] = '7'; break;
                case 8: freq[k] = '8'; break;
                case 9: freq[k] = '9'; break;
                default: break;
        }
}
if(i == 0){
        if(freq[k-1] == ' '){
                freq[k] = ' ';
        } else{
                freq[k]='0';
        }
}
i=0;
k++;
switch(f) {
                case 0: freq[k] = '0'; break;
                case 1: freq[k] = '1'; break;
                case 2: freq[k] = '2'; break;
                case 3: freq[k] = '3'; break;
                case 4: freq[k] = '4'; break;
                case 5: freq[k] = '5'; break;
                case 6: freq[k] = '6'; break;
                case 7: freq[k] = '7'; break;
                case 8: freq[k] = '8'; break;
                case 9: freq[k] = '9'; break;
```

```
                default: break;
            }

  freq[k+1] = ' ';
  freq[k+2] = 'H';
  freq[k+3] = 'z';
  return freq;
}
```

```
//-----------------------------------------------------
// void cambiaRangoExploracion(void)
//
// Descripción:
// Gestiona el cambio de ventana de análisis en frecuencia, de acuerdo con el pico
// máximo detectado en el cálculo de la DFT. Controla los casos límites y saca por
// el LCD la frecuencia hallada
//-----------------------------------------------------
void cambiaRangoExploracion(void) {
  cli(); //Parar la interrupción para los siguientes cálculos costosos
  explorar++;
  switch (explorar) {
        case 1: //Nuevo rango de exploración: ventana de 100Hz y paso de 5Hz
              if (freqMax<=55)
                    freqAntes = 5;
              else if (freqMax>=1945)
                    freqAntes = 1895
              //Si no entramos en ningún caso limite
              else freqAntes = freqMax-50;

              for (i=0; i<N_FRECS; i++) { //con sumar uno a cada salto, se aumenta la frec en 5Hz
                    salto[i] = (freqAntes/5) + i;
                    //Nunca se van a pasar de las 800 muestras del seno
                    pSin[i] = sin5Hz + salto[0];
                    pCos[i] = sin5Hz + salto[0] + (NUM_MUESTRAS_PERIODO_5HZ>>2);
              }
              picoMax=0;
        break;

        case 2://Vuelta al calculo de la DFT original
              for(i=0; i<N_FRECS; i++) {
```

```
                        salto[i] = saltoPrincipal[i];
                    }
                    maximoString = stringFrecuencia(freqMax);
                    sacaLCD(maximoString, 2);

                    explorar=0;
                    picoMax=0;
                    freqAntes=0;;
                    freqMax=0;

                    contador1seg=0;
            break;
            default: break;
    }
    sti(); //habilitar interrupciones
}
```

```
//-----------------------------------------------------
// int * estimaFase0a90(int re, int im)
//
// Descripción:
// Función auxiliar para el cálculo de la fase. Una vez se ha detectado
// en qué cuadrante se encuentra la fase, se llama a esta función para
// que de un valor más preciso dentro de ese cuadrante. Se sigue la
// estrategia "divide y vencerás".
// Devuelve un puntero a un valor del array de escalado, como representación
// de los grados como un nivel de tensión en el osciloscopio
//-----------------------------------------------------
int * estimaFase0a90(int re, int im) {
  int *pfase0a90;
  if (re<0) re*=-1;
  if (im<0) im*=-1;

  if (re>im) {//0 a 45
        //22.5 cuando aprox. cuando Im = 0.414*Re
        // o aprox 10*Im = 4*Re => 21.8
        if (10*im < 4*re) {//0 a 22.5 (21.8)
              //11.25 cuando Im = 0.19*Re
              // o 10*Im = 2*Re
              if (10*im < 2*re) //0 a 11.25 (11.3)
                    pfase0a90 = escalado; //0
              else //11.25 a 22.5
                    pfase0a90 = escalado+1; //11.25
              }
        else {//22.5 a 45
              //33.75 si Im = 0.668*Re
              if (10*im < 7*re) //22.5 a 33.75 (35)
                    pfase0a90 = escalado+2; //22.5
```

```
                else //33.75 a 45
                        pfase0a90 = escalado+3; //33.75
        }
    }
    else {//45 a 90
        //22.5 cuando aprox. cuando Im = 0.414*Re
        // o aprox 10*Im = 4*Re => 21.8
        if (10*re > 4*im) {//45 a 67.5
            if (10*re > 2*im) //45 a 56.25
                pfase0a90 = escalado+4; //45
            else //56.25 a 67.5
                pfase0a90 = escalado+5; //56.25
            }
        else {//67.5 a 90
            if (10*re > 7*im) //67.5 a 78.75
                pfase0a90 = escalado+6; //67.5
            else //78.75 a 90
                pfase0a90 = escalado+7; //78.75
        }
    }
    return pfase0a90;
}
```

```
//-----------------------------------------------------
// int conversion_dB(int x)
//
// Descripción:
// Convierte los valores calculados de la DFT a decibelios
// de una forma eficiente al seguir la estrategia "divide
// y vencerás".
// Devuelve un valor de el array de escalado del DAC
//-----------------------------------------------------
int conversion_dB(int x) {
   if (x>umbrales[31]) return escalado[31];
   if (x<umbrales[0]) return escalado[0];

   if (x>umbrales[16]) { // 16 - 31
      if (x>umbrales[24]) { // 24 - 31
         if (x>umbrales[28]) { // 28 -31
            if (x>umbrales[30]) { // 30 - 31
               if (x>umbrales[31]) return escalado[31];
               else return escalado[30];
            }
            else { // 28 - 29
               if (x>umbrales[29]) return escalado[29];
               else return escalado[28];
            }
         }
         else { // 16 - 19
            if (x>umbrales[26]) { // 26 - 27
               if (x>umbrales[27]) return escalado[27];
               else return escalado[26];
            }
            else { // 24 - 25
```

```
            if (x>umbrales[25]) return escalado[25];
            else return escalado[24];
         }
      }
   }
   else { // 16 - 23
      if (x>umbrales[20]) { // 20 - 23
         if (x>umbrales[22]) { // 22 - 23
            if (x>umbrales[23]) return escalado[23];
            else return escalado[22];
         }
         else { // 20 - 21
            if (x>umbrales[21]) return escalado[21];
            else return escalado[20];
         }
      }
      else { // 16 - 19
         if (x>umbrales[18]) { // 18 - 19
            if (x>umbrales[19]) return escalado[19];
            else return escalado[18];
         }
         else { // 16 - 17
            if (x>umbrales[17]) return escalado[17];
            else return escalado[16];
         }
      }
   }
}

else{ // 0 - 15
   if (x>umbrales[8]) { // 8 - 15
      if (x>umbrales[12]) { // 12 - 15
```

```
        if (x>umbrales[14]) { // 14 - 15
            if (x>umbrales[15]) return escalado[15];
            else return escalado[14];
        }
        else { // 12 - 13
            if (x>umbrales[13]) return escalado[13];
            else return escalado[12];
        }
    }
    else { // 8 - 11
        if (x>umbrales[10]) { // 10 -11
            if (x>umbrales[11]) return escalado[11];
            else return escalado[10];
        }
        else { // 8 - 9
            if (x>umbrales[9]) return escalado[9];
            else return escalado[8];
        }
    }
}
else { // 0 - 7
    if (x>umbrales[4]) { // 4 - 7
        if (x>umbrales[6]) { // 6 - 7
            if (x>umbrales[7]) return escalado[7];
            else return escalado[6];
        }
        else { // 4 - 5
            if (x>umbrales[5]) return escalado[5];
            else return escalado[4];
        }
    }
    else { // 0 - 3
```

```
            if (x>umbrales[2]) { // 2 - 3
               if (x>umbrales[3]) return escalado[3];
               else return escalado[2];
            }
            else { // 0 - 1
               if (x>umbrales[1]) return escalado[1];
               else return escalado[0];
            }
         }
      }
   }
   return 0;
}
```

```
//-----------------------------------------------------
// void DFT_Inversa(void)
//
// Descripción:
// Esta rutina reconstruye la señal que pasa por el filtro sintonizable,
// es decir, genera en el tiempo la señal filtrada. Calcula los pesos
// de cada componente en frecuencia y devuelve una suma de senos
// ponderados por esos pesos, para posteriormente sacar esta suma por
// el DAC. Es necesario parar la interrupción para estos cálculos.
//-----------------------------------------------------
void DFT_Inversa(void) {
  cli(); //Parar interrupción para el cálculo de la DFT inversa
  //Cálculo de coeficientes para ponderar los senos
  sumaCoef=0;
  for(i=0; i<N_FRECS; i++) {
          coef[i] = dft[i]/125; //dft ya en decibelios
          sumaCoef+= coef[i];
  }
  sumaCoef++;
  //Cálculo de las muestras de salida
  for (i=0; i<N; i++){
          for(j=0; j<N_FRECS; j++) {
                  salida[i]+= coef[j]*(*indicesSenos[j] + 2500);
                  indicesSenos[j] += salto[j];
                  if((indicesSenos[j] - sin5Hz) >= NUM_MUESTRAS_PERIODO_5HZ)
                          indicesSenos[j] -= NUM_MUESTRAS_PERIODO_5HZ;
          }
          salida[i] = salida[i]/sumaCoef;
  }
  sti(); //habilitar interrupciones
}
```

```
//-----------------------------------------------------
// void rutina_tout0(void)
//
// Descripción:
// Interrupción que originalmente sólo muestraba la señal de entrada
// y calculaba su DFT. Ahora se ocupa de todas las mejoras,
// haciendo las operaciones pertinentes de acuerdo con la mejora
// seleccionada.
//-----------------------------------------------------
void rutina_tout0(void) {

  mbar_writeShort(MCFSIM_TER0,BORRA_REF);    // Reset del bit de fin de cuenta

  n++;
  //Obtenemos muestra
  muestra = ADC_dato();

  if(estado == ANALIZADOR_HW) {
        if (mostrarHw) //Ya calculada H(w), la sacamos
              DAC_dato(Hw[cont160Inter>>3]);
        else { //Mientras se esté calculando, se barre en frecuencia
              DAC_dato(sin5Hz[indice] + 2500);
              barrido = cont1600ints/80; //misma frecuencia durante 80 interrupciones
              indice = (indice + salto[barrido])%NUM_MUESTRAS_PERIODO_5HZ;

              cont1600ints++;
              if(cont1600ints == 1600) { //Cuando se termina de calcular H(W)
                    cont1600ints = 0;
                    mostrarHw = 1;
                    barrido = 0;
              }
```

```
		}
	}
	else if(salidaAnalogica && DFTlista) {
		DAC_dato(salida[nFiltro]);
		nFiltro++;
		if (nFiltro == N) nFiltro=0;
	}
	else {
		DAC_dato(dft[cont160Inter>>3]);

		if (lcd_matriz%2)
			sacaLEDs();
	}

	cont160Inter++;

	if(cont160Inter == 160) {
		for (k=0; k<N_FRECS; k++) {
			dftCos[k] = 0;
			dftSin[k] = 0;
			pSin[k] = sin5Hz;
			pCos[k] = sin5Hz + (NUM_MUESTRAS_PERIODO_5HZ>>2); //empieza en sin5Hz[200]
		}
		calculo = 1;  //Vuelvo a calcular dft al sacar las 20 componentes (calculo = 1)
		cont160Inter = 0;
		n = 0;
		pSal = pSal | 0x0010;
		set16_puertoS(pSal);//activar rampa
	}
	else {
		pSal = pSal & 0xFFEF;
		set16_puertoS(pSal);
```

```
	}

	//Calcular aportaciones real e imaginaria de la muestra
	for (k=0; k<N_FRECS; k++) {
		dftCos[k] += muestra*(*pCos[k]);
		dftSin[k] += muestra*(*pSin[k]);
		//Actualizamos los Índices
		pSin[k] += salto[k];
		pCos[k] += salto[k];
		//Buffer circular
		if((pSin[k] - sin5Hz) >= NUM_MUESTRAS_PERIODO_5HZ)
			pSin[k] -= NUM_MUESTRAS_PERIODO_5HZ;
		if((pCos[k] - sin5Hz) >= NUM_MUESTRAS_PERIODO_5HZ)
			pCos[k] -= NUM_MUESTRAS_PERIODO_5HZ;
	}

	//Calculo del modulo de la dft
	if(calculo && (n == N)){
		//Calculo del mÃ³dulo al cuadrado
		for (k=0; k<N_FRECS; k++) {
			switch(estado) {

			case MODULO:
				dft[k] = ((dftCos[k]>>DESESCALADO) * (dftCos[k]))
						+ ((dftSin[k]>>DESESCALADO) * (dftSin[k]));
				dft[k] = conversion_dB(dft[k]);
			break;

			case FASE:
				if (dftCos[k]>0 && dftSin[k]>0) //0 a 90
					dft[k] = *estimaFase0a90(dftCos[k],dftSin[k]);
				else if (dftCos[k]<0 && dftSin[k]>0) //90 a 180
```

```
                dft[k] = *(estimaFase0a90(dftCos[k],dftSin[k])+8); //+90
        else if (dftCos[k]<0 && dftSin[k]<0) //180 a 270
                dft[k] = *(estimaFase0a90(dftCos[k],dftSin[k])+16); //+180
        else /*if (dftCos[k] > 0 && dftSin[k] < 0)*/ //270 a 360
                dft[k] = *(estimaFase0a90(dftCos[k],dftSin[k])+24); //+270
break;

case AFINADOR:
        dft[k] = ((dftCos[k]>>DESESCALADO) * (dftCos[k]))
                        + ((dftSin[k]>>DESESCALADO) * (dftSin[k]));

        if (dft[k] > picoMax) {
                picoMax= dft[k];
                freqMax = salto[k]*5;
        }
        dft[k] = conversion_dB(dft[k]);
break;

case ANALIZADOR_HW:
        if (!modulo_fase) { //Calcular módulo
                dft[k] = ((dftCos[k]>>DESESCALADO) * (dftCos[k]))
                                + ((dftSin[k]>>DESESCALADO) * (dftSin[k]));
                dft[k] = conversion_dB(dft[k]);
        }
        else { //Calcular fase
                if (dftCos[k]>0 && dftSin[k]>0) //0 a 90
                        dft[k] = *estimaFase0a90(dftCos[k],dftSin[k]);
                else if (dftCos[k]<0 && dftSin[k]>0) //90 a 180
                        dft[k] = *(estimaFase0a90(dftCos[k],dftSin[k])+8); //+90
                else if (dftCos[k]<0 && dftSin[k]<0) //180 a 270
                        dft[k] = *(estimaFase0a90(dftCos[k],dftSin[k])+16); //+180
                else /*if (dftCos[k] > 0 && dftSin[k] < 0)*/ //270 a 360
```

```
                                dft[k] = *(estimaFase0a90(dftCos[k],dftSin[k])+24); //+270
                        }
                break;

                case FILTRO_SINTONIZABLE:

                        if (k<=freqBaja/100 || k>=freqAlta/100)
                                dft[k]=escalado[0];
                        else {
                                dft[k] = ((dftCos[k]>>DESESCALADO) * (dftCos[k]))
                                        + ((dftSin[k]>>DESESCALADO) * (dftSin[k]));
                                dft[k] = conversion_dB(dft[k]);
                        }

                break;

                default: break;
                }
                if (estado==ANALIZADOR_HW) Hw[barrido] = dft[barrido];
                //Reiniciamos acumuladores
                dftCos[k] = 0;
                dftSin[k] = 0;
                //Reiniciamos indices a seno y coseno
                pSin[k] = sin5Hz;
                pCos[k] = sin5Hz + (NUM_MUESTRAS_PERIODO_5HZ>>2); //empieza en sin5Hz[200]
        }
        calculo = 0;
        n=0;
        if (estado==AFINADOR) {
                contador1seg++;
                if (contador1seg >= 25) {
                        cambiaRangoExploracion();
```

```
                }
            }
            if (estado==FILTRO_SINTONIZABLE) {
                DFTlista = 1;
                if (salidaAnalogica && !salidaAnalogicaCalculada) {
                    DFT_Inversa();
                    salidaAnalogicaCalculada=1;
                }
            }
        }
    }
```

```
//-----------------------------------------------------
// void __init(void)
//
// Descripción:
// Rutina de inicialización del programa. Valor inicial de
// variables, punteros, inicialización de la interrupción,
// DAC/ADC y LCD.
//-----------------------------------------------------
void __init(void) {
  calculo = 1;
  n = 0;
  cont160Inter = 0;

  for (i=0; i<N_FRECS; i++) {
        dft[i] = escalado[0]; //inicializado al valor que DAC reconoce como cero
        Hw[i] = escalado[0];

        dftSin[i] = 0;
        dftCos[i] = 0;
        salto[i] = saltoPrincipal[i];
        pSin[i] = sin5Hz;
        pCos[i] = sin5Hz + (NUM_MUESTRAS_PERIODO_5HZ>>2); //empieza en sin5Hz[200]

        indicesSenos[i] = sin5Hz;
  }

  LCD_reset();    // Reseteamos el LCD
  LCD_init();     // e inicializamos el display

  DAC_ADC_init();
  mbar_writeByte(MCFSIM_PIVR,V_BASE);                    // Fija comienzo de vectores de interrupción en V_BASE.
```

```
  ACCESO_A_MEMORIA_LONG(DIR_VTMR0)= (ULONG)_prep_TOUT0; // Escribimos la direcciÃ³n de la función para TMR0
  output("COMIENZA EL PROGRAMA\r\n");
  mbar_writeShort(MCFSIM_TMR0, (PRESCALADO-1)<<8|0x3D);          // TMR0: PS=1-1=0 CE=00 OM=1 ORI=1 FRR=1 CLK=10 RST=1
  mbar_writeShort(MCFSIM_TCN0, 0x0000);            // Ponemos a 0 el contador del TIMER0
  mbar_writeShort(MCFSIM_TRR0, CNT_INT1);       // Fijamos la cuenta final del contador
  mbar_writeLong(MCFSIM_ICR1, 0x8888C888);       // Marca la interrupción del TIMER0 como no pendiente y de nivel 4
  sti();     // Habilitamos interrupciones
 }
```

```
//-------------------------------------------------------
// void bucleMain(void)
//
// Descripción:
// Bucle principal del programa. Gestiona la interfaz de usuario,
// leyendo el teclado y actuando en consecuencia al elegir
// menús o submenús. Conmuta entre mejoras según las opciones
// elegidas por el usuario a través del teclado matricial.
//-------------------------------------------------------
void bucleMain(void) {

  output("Pulse una de las siguientes opciones\n");
   output("1) Modulo\n");
   output("2) Fase\n");
   output("3) Afinador\n");
   output("4) Analizador de espectros\n");
  output("5) Filtro sintonizable\n");
   output("====================\n");

  while(1){
            opcion = teclado();

     switch (opcion){
       case '1': output("Has elegido modulo\n");
                        estado = MODULO;
                                    sacaLCD("MODULO",1);
                                break;
       case '2': output("Has elegido fase\n");
                        estado = FASE;
                        sacaLCD("FASE",1);
                                break;
```

```
case '3': output("Has elegido afinador\n");
		estado = AFINADOR;
				sacaLCD("AFINADOR", 1);
		//sacaremos el maximo por el LCD una vez se calcule
			break;
case '4': output("Has elegido analizador de circuitos\n");
				if (estado!=ANALIZADOR_HW)
					estado = ANALIZADOR_HW;
		else mostrarHw = 0; //para que recalcule la Hw si se ha pulsado de nuevo
				indice=0;

		sacaLCD("ANALIZADOR",1);
				sacaLCD("DE ESPECTROS",2);
				output("\nElija lo que quiera visualizar: \n");
				output("1) Modulo\n");
				output("2) Fase\n");
				output("F) Volver al menú principal\n");

				while(1) {
					opcion = teclado();
					if (opcion=='1') {
						modulo_fase = 0;
						mostrarHw = 0;
				}

					if (opcion=='2') {
						modulo_fase = 1;
						mostrarHw = 0;
				}
					if (opcion == 'F') {
						modulo_fase=0;
```

```
                                        mostrarHw=0;
                                        break;
                                }
                        }

                break;
case '5': output("Has elegido filtro sintonizable\n");
                estado = FILTRO_SINTONIZABLE;
                        sacaLCD("FILTRO",1);
                        sacaLCD("SINTONIZABLE",2);
                        output("\nElija el tipo de filtro: \n");
                    output("1) Paso bajo\n");
                    output("2) Paso alto\n");
                    output("3) Paso banda\n");
                        output("F) Volver al menú principal\n");

                        indice=0;
                        freqAlta = 0;
                        freqBaja = 0;

                        while(1){

                                        opcion = teclado();

                                        switch (opcion){
                                          case '1': output("\nHas elegido Paso bajo\n");
                                                    freqAlta = 0;
                                                    freqBaja = 0;
                                                  sacaLCD("FILTRO",1);
                                                            sacaLCD("PASO BAJO",2);
                                                            output("Introduce la frecuencia de corte\n");
```

```
                    opcion=teclado();
                    freqAlta += strToInt(opcion)*1000;
                    opcion=teclado();
                    freqAlta += strToInt(opcion)*100;
                    opcion=teclado();
                    freqAlta += strToInt(opcion)*10;
                    opcion=teclado();
                    freqAlta += strToInt(opcion);
                    output("FreqAlta = ");
                    outNum(10, freqAlta, 0);
                    output("\n");

        break;
case '2': output("\nHas elegido Paso alto\n");
          freqAlta = 2000;
   freqBaja = 0;

                    sacaLCD("FILTRO",1);
        sacaLCD("PASO ALTO",2);
                    output("Introduce la frecuencia de corte\n");

                    opcion=teclado();
                    freqBaja += strToInt(opcion)*1000;
                    opcion=teclado();
                    freqBaja += strToInt(opcion)*100;
                    opcion=teclado();
                    freqBaja += strToInt(opcion)*10;
                    opcion=teclado();
                    freqBaja += strToInt(opcion);
                    output("FreqBaja = ");
                    outNum(10, freqBaja, 0);
                    output("\n");
```

```
                break;

        case '3': output("\nHas elegido Paso banda\n");
                freqAlta = 0;
                freqBaja = 0;

                        sacaLCD("FILTRO",1);
                        sacaLCD("PASO BANDA", 2);
                        output("Introduce la frecuencia de corte Baja\n");
                        opcion=teclado();
                        freqBaja += strToInt(opcion)*1000;
                        opcion=teclado();
                        freqBaja += strToInt(opcion)*100;
                        opcion=teclado();
                        freqBaja += strToInt(opcion)*10;
                        opcion=teclado();
                        freqBaja += strToInt(opcion);
                        output("FreqBaja = ");
                        outNum(10, freqBaja, 0);
                        output("\n");

                        output("Introduce la frecuencia de corte Alta\n");
                        opcion=teclado();
                        freqAlta += strToInt(opcion)*1000;
                        opcion=teclado();
                        freqAlta += strToInt(opcion)*100;
                        opcion=teclado();
                        freqAlta += strToInt(opcion)*10;
                        opcion=teclado();
                        freqAlta += strToInt(opcion);
                        output("FreqAlta = ");
                        outNum(10, freqAlta, 0);
```

```
                                                output("\n");

                                        break;

                                  default: break;

                                }

                                while(1){
                                        output("\nElija lo que quiere que se muestre: \n");
                                        output("1) DFT\n");
                                        output("2) Tiempo\n");
                                        opcion = teclado();
                                        if(opcion=='1')
                                                salidaAnalogica = 0;
                                        if (opcion == '2') {
                                                salidaAnalogica = 1;
                                                salidaAnalogicaCalculada = 0;
                                        }

                                        if (opcion == 'F') break;
                                }


                                if (opcion == 'F') {
                                        salidaAnalogica = 0;
                                        break;
                                }
                        }
case '6': output("LCD <--> Matriz de LEDS\n");
                        lcd_matriz++;
                        //Si es par, lcd, y si es impar se visualizará la DFT en la matriz de LEDS
```

```
                    estado = MODULO;
                        break;

        default:
                    output("Elije mejora a probar\n");
      }
    }
  }

//---------------------------------------------------------
// Definición de rutinas de atención a las interrupciones
// Es necesario definirlas aunque estén vacías
void rutina_int1(void){}
void rutina_int2(void){}
void rutina_int3(void){}
void rutina_int4(void){}
void rutina_tout1(void){}
void rutina_tout2(void){}
void rutina_tout3(void){}
```