Filtro digital programable

Juan Pablo Moreno – 20142005231, Sebastián Bejarano – 20142005016

*Resumen: Cuando se habla de filtros tenemos dos grandes posibilidades, trabajar con filtros analógicos o con filtros digitales. En este reporte hablaremos del diseño de un filtro digital programable de orden 50 con efecto pasa bajas, pasa altas o pasa banda utilizando la metodología de diseño de ventaneo de respuesta al impulso para filtros FIR.*

***Palabras clave: Psoc5LP, Filtro digital, Respuesta en frecuencia, Ventaneo de Respuesta al Impulso, Filtros FIR.***

**1. INTRODUCCIÓN**

Si tenemos una señal cualquiera, esta tiene una composición espectral, es decir, su energía está dispersa (Por el teorema de Rayleigh) a lo largo de un ancho de banda.

Las señales son el corazón de la información, por lo tanto, es de vital importancia que se encuentren apegadas de manera correcta al fenómeno que representa. Sin embargo, esto puede verse afectado por distintos factores, por ejemplo, en potenciales evocados, el ruido que introduce la red eléctrica ya sean 50 Hz o 60 Hz o el ruido miográfico que se produce por el movimiento de los músculos al respirar. Todo esto mencionado anteriormente puede ser removido utilizando filtros, ahora en este informe lo que haremos será describir detalladamente la implementación de un filtro digital programable tipo FIR (Finite Impulse Response) de orden 50 utilizando un teclado matricial para elegir entre pasa bajas, pasa altas o pasa banda.

**2. FORMULACIÓN DEL PROBLEMA**

La problemática a resolver fue:

“*Diseñar un filtro digital programable utilizando el ADC del PSoC5LP.*”

*2.1 El filtro debe ser de tercer orden como mínimo*

El orden del filtro implementado debe tercer orden o más, en este caso más debido a que el filtro que se implementará es FIR y requiere un orden mucho más alto para realizar la misma tarea que un filtro analógico que se llevaría a un filtro tipo IIR en el dominio discreto (utilizando retenedor de orden cero y discretizando la función de transferencia). Lo elegimos de orden 50 porque así teníamos las ganancias unitarias y de cero en las bandas de paso y de rechazo, y la atenuación del pasa banda no sería tan alta según las pruebas hechas en Matlab.

*2.2 Debe ser programable*

El usuario debe ser capaz de ingresar la frecuencia de corte del filtro, entonces para esto se propuso utilizar un teclado matricial de 4x3, cuyo funcionamiento se explicará con detalle un poco más adelante.

*2.3 Se debe poder configurar el filtro en tres posibles configuraciones*

El usuario debe poder escoger en cual configuración desea usar el filtro digital:

* Pasa Bajo
* Pasa Alto
* Pasa Banda

*2.4Frecuencia de Corte ira por lo menos 1KHz*

La frecuencia de corte programada por el usuario debe ser menor o igual a 1KHz

*2.5 La señal podrá ser bipolar*

La señal puede tener valores positivos o negativos.

**3. DISEÑO Y MODELO DE SOLUCIÓN**

Como primera medida se decide implementar un filtro FIR, al escoger ese tipo de filtro se notó que el orden mínimo para su implementación era 50 debido al rango de frecuencias pedidas en la guía de laboratorio, así que este fue el orden escogido.

Antes de empezar a trabajar en la discretización de la señal es necesario acondicionar la señal es decir dejar sus niveles de tensión ente 0V y 5V para ello se usó el circuito mostrado en la Figura 2.

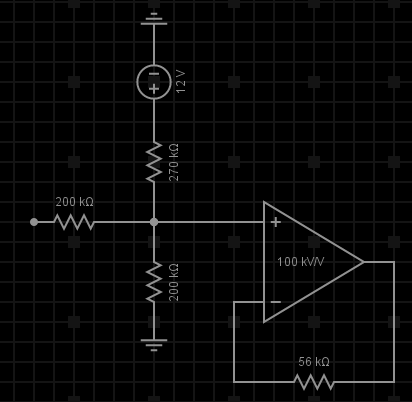


Figura 2: Circuito de Acondicionamiento de la señal de entrada

Se escogió un ADC\_SAR con la siguiente configuración

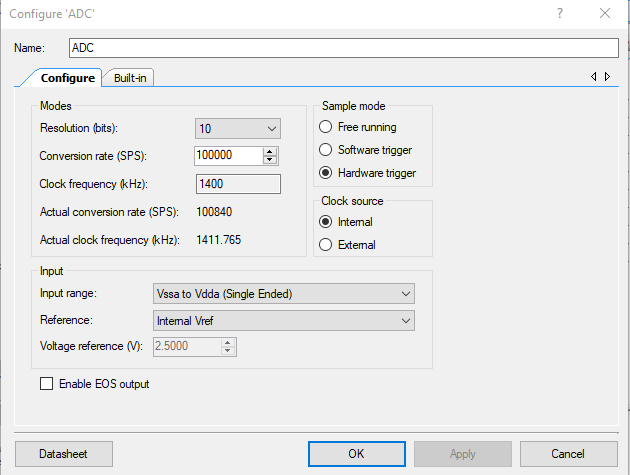


Figura 3: Configuración del ADC\_SAR

Para la frecuencia de muestreo se usó un Timer con la siguiente configuración

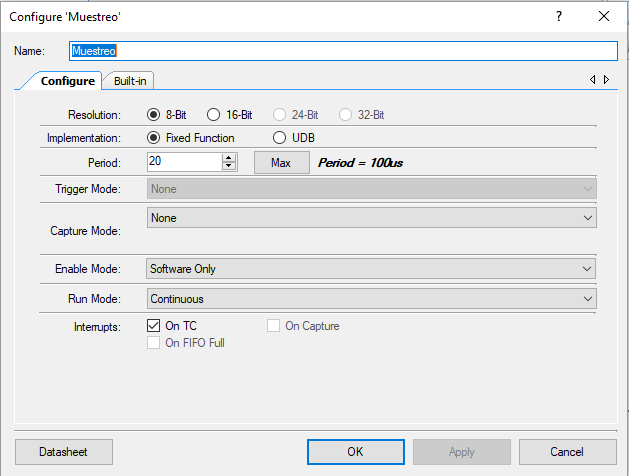


Figura 4. Configuración Timer, para frecuencia de muestreo

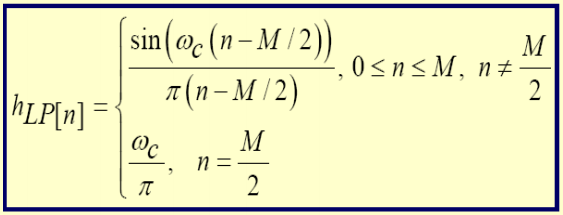
Una vez digitalizada la señal de entrada se continuó con la selección del filtro, para esto se utilizó un teclado matricial de 4x3 de membrana como el que se muestra en la figura 5, para hacer funcionar este teclado, lo que teníamos que hacer era poner un ‘1’ lógico a recorrer las filas, y recibirlo en las columnas, es decir, ponemos una base de tiempo por ejemplo de un mili segundo, y cada que pase un mili segundo escribimos en el puerto de las filas 0001, luego que pase otro milisegundo 0010, y así sucesivamente hasta 1000. Esto es para poder discriminar la fila de la cuál viene el dígito, ya que sin esto no podríamos distinguir, por ejemplo, entre el 1, el 4, el 7, o el asterisco. Finalmente como es un ‘1’ lo que está recorriendo las filas ponemos resistencias de Pull Down en el puerto de las columnas para activar una interrupción por flanco de subida y así lograr determinar cuál tecla fue oprimida.



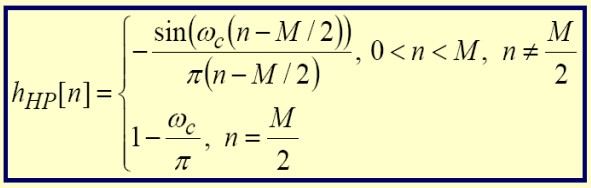
Figura 5. Teclado Matricial 4x3 de Membrana (Tomada de internet)

La construcción de los filtros fue hecha con las siguientes funciones y la ayuda de la librería “math”:

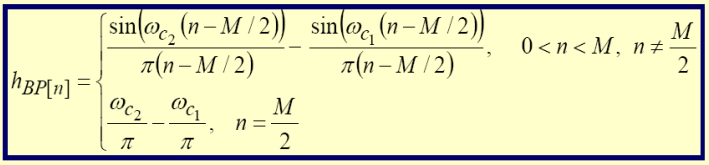
* Pasa Bajo



* Pasa Alto



* Pasa Banda



Las anteriores fórmulas (tomadas de [1]) surgen de la IDTFT (Inverse Discrete Time Fourier Transform) del análisis de la respuesta al impulso ideal en frecuencia, es decir, tomamos la respuesta ideal en frecuencia y le aplicamos transformada inversa. Esto genera unas respuestas infinitas y no causales por lo cual dentro de las mismas fórmulas se contempla un acotamiento de la señal, y también un desplazamiento de la misma, esto con el fin de aplicar una “ventana” (lo que llamamos ventaneo) que consiste en multiplicar la sección tomada de la respuesta al impulso y multiplicarla por otra función que nos permita suavizar el resultado en frecuencia o bajar la energía aportada por los lóbulos laterales de la misma al acotarla.

El usuario digita la frecuencia de corte tanto para el pasa bajos como para el pasa altos, mientras que para el filtro pasa banda debe digitar la frecuencia central, con un ancho de banda prestablecido de 100Hz para posteriormente hacer el cálculo de los coeficientes de la señal haciendo uso de las ecuaciones mostradas anteriormente como se muestra en el código al final del documento.

El espectro de tiempo discreto tiene una periodicidad de donde la mitad de éste, representa la mitad de la frecuencia de muestreo o frecuencia de Nyquist (frecuencia máxima de la señal a procesar), entonces para relacionar la frecuencia de corte discreta del filtro con una frecuencia lineal analógica, utilizamos la siguiente expresión:

Para realizar la implementación de la convolución, decidimos no utilizar un ciclo for debido a las sucesivas comparaciones que debía hacer, lo que volvería el proceso un poco más lento

Para la salida de la señal se toma la función donde se almacenan los coeficientes calculados y se envían al DAC que se configuro de la siguiente manera

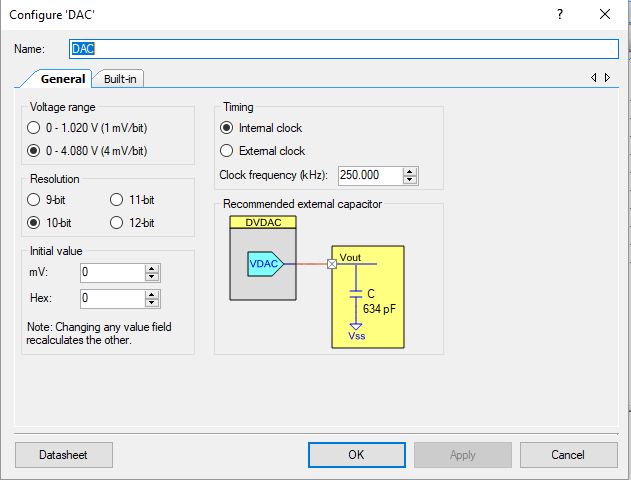


Figura 6. Configuración del DAC

A la salida del PSOC se acondiciono nuevamente la señal para dejarla nuevamente en sus niveles de voltaje iniciales. El circuito cumple la función inversa del circuito en la Figura 2.

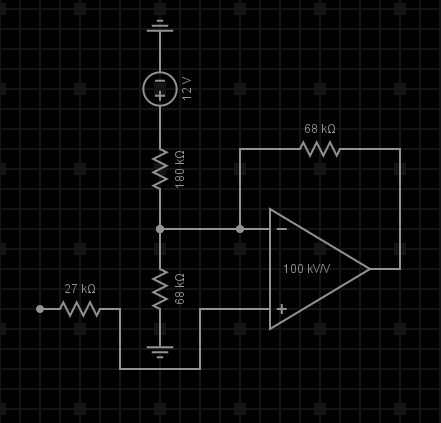


Figura 7. Reacondicionamiento de la señal

**4. RESULTADOS**

Primero se muestra la “interfaz” con la que interactúa el usuario

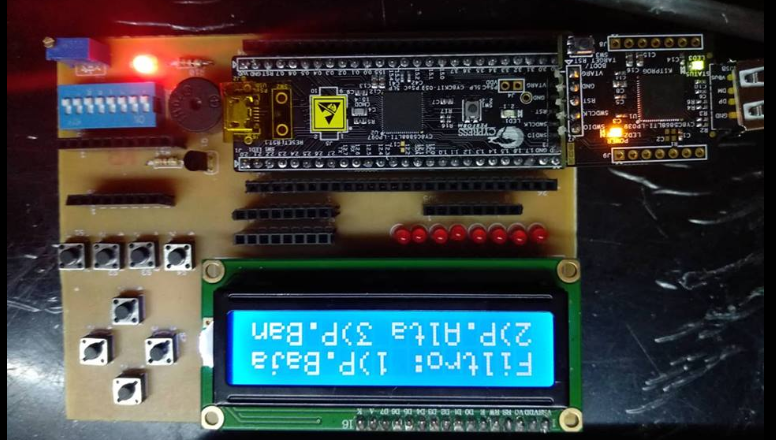


Figura 6.1 Interfaz

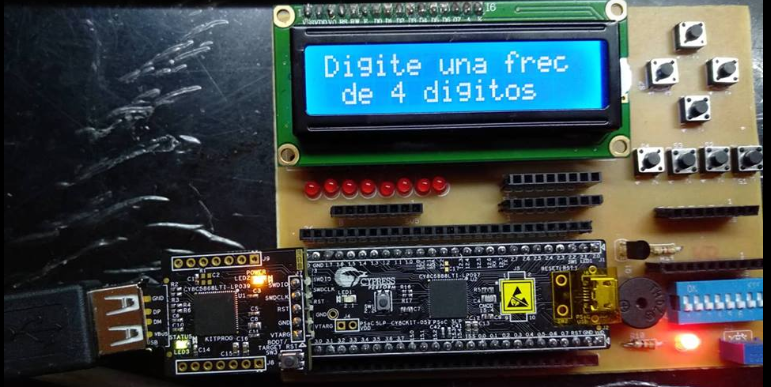


Figura 6.2 Interfaz

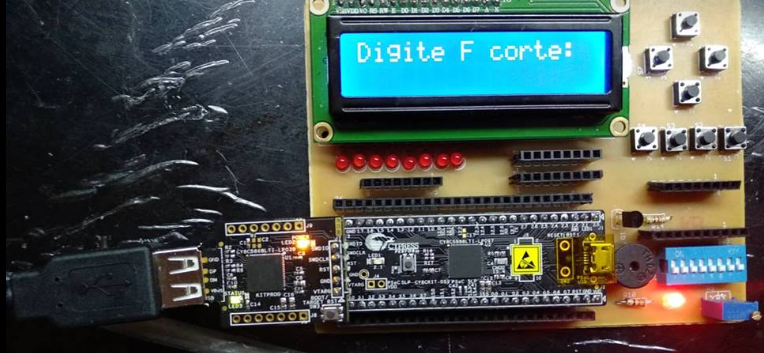


Figura 6.3 Interfaz

Dentro de los resultados se encontró que el filtro pasa bajo y el pasa alto frecuencia de corte dada por el filtro era prácticamente la misma que la digitada por el usuario. Para el filtro pasa banda se encontró que la frecuencia central del filtro se encontraba 15Hz por encima de la digitada por el usuario.

**5. CONCLUSIONES**

* Se encontró que entre mayor sea la frecuencia de muestreo del filtro la señal emulada o señal filtrada será más suave
* Para los Filtros FIR, entre más bajo sea el rango de funcionamiento (en frecuencia) mayor debe ser el orden del filtro para poder ser implementado
* Para este laboratorio en especial, no es posible utilizar otro tipo de técnicas de diseño de filtros digitales, debido a que debía ser programable. La transformación bilineal de funciones de transferencia requiere una frecuencia definida para obtener una función de transferencia por cada frecuencia definida lo que nos hubiese obligado a generar N ecuaciones de diferencias para implementar un filtro para cada frecuencia.
* A pesar de utilizar una frecuencia de 74MHz (máxima frecuencia permitida por el IMO) para el clock master del PSoC, nos vimos limitados a utilizar una frecuencia máxima de muestreo de 3KHz, lo que nos deterioraba demasiado la señal a frecuencias a más de 800Hz.

**6. REFERENCIAS**

[1]http://www.fimee.ugto.mx/profesores/arturogp/documentos/Filtrado%20Digital/Lectura%203\_Filtrado\_Digital.pdf

[2] <http://www.cypress.com/file/376411/download> Datasheet del timer del PSoC

[3] <http://www.cypress.com/file/131731/download> Datasheet del ADC del PSoC

[4] [http://www.cypress.com/file/141831/download](http://www.cypress.com/file/131731/download) Datasheet del DAC del PSoC

**Código utilizado en la tarea:**

#include "project.h"

#include <math.h>

//Creación de variables h para los coeficientes del filtro, x para las muestras, y para la salida, i como variable de conteo

//tecla para guardar lo recibido del teclado

volatile double h[51];

uint16 x[51];

uint16 frec\_final;

volatile uint8 i,cont\_char,u\_frec,d\_frec,c\_frec,um\_frec;

volatile char fila, tecla, columna,paso,filtro;

volatile char frec[4];

CY\_ISR(llegaMuestra)

{

x[50] = x[49];

x[49] = x[48];

x[48] = x[47];

x[47] = x[46];

x[46] = x[45];

x[45] = x[44];

x[44] = x[43];

x[43] = x[42];

x[42] = x[41];

x[41] = x[40];

x[40] = x[39];

x[39] = x[38];

x[38] = x[37];

x[37] = x[36];

x[36] = x[35];

x[35] = x[34];

x[34] = x[33];

x[33] = x[32];

x[32] = x[31];

x[31] = x[30];

x[30] = x[29];

x[29] = x[28];

x[28] = x[27];

x[27] = x[26];

x[26] = x[25];

x[25] = x[24];

x[24] = x[23];

x[23] = x[22];

x[22] = x[21];

x[21] = x[20];

x[20] = x[19];

x[19] = x[18];

x[18] = x[17];

x[17] = x[16];

x[16] = x[15];

x[15] = x[14];

x[14] = x[13];

x[13] = x[12];

x[12] = x[11];

x[11] = x[10];

x[10] = x[9];

x[9] = x[8];

x[8] = x[7];

x[7] = x[6];

x[6] = x[5];

x[5] = x[4];

x[4] = x[3];

x[3] = x[2];

x[2] = x[1];

x[1] = x[0];

x[0] = ADC\_GetResult16();

if(filtro == 2 || filtro == 3)

{

double y = (double)x[0]\*h[0]+(double)x[1]\*h[1]+(double)x[2]\*h[2]+(double)x[3]\*h[3]+(double)x[4]\*h[4]+(double)x[5]\*h[5]+(double)x[6]\*h[6]+(double)x[7]\*h[7]+(double)x[8]\*h[8]+(double)x[9]\*h[9]+(double)x[10]\*h[10]+(double)x[11]\*h[11]+(double)x[12]\*h[12]+(double)x[13]\*h[13]+(double)x[14]\*h[14]+(double)x[15]\*h[15]+(double)x[16]\*h[16]+(double)x[17]\*h[17]+(double)x[18]\*h[18]+(double)x[19]\*h[19]+(double)x[20]\*h[20]+(double)x[21]\*h[21]+(double)x[22]\*h[22]+(double)x[23]\*h[23]+(double)x[24]\*h[24]+(double)x[25]\*h[25]+(double)x[26]\*h[26]+(double)x[27]\*h[27]+(double)x[28]\*h[28]+(double)x[29]\*h[29]+(double)x[30]\*h[30]+(double)x[31]\*h[31]+(double)x[32]\*h[32]+(double)x[33]\*h[33]+(double)x[34]\*h[34]+(double)x[35]\*h[35]+(double)x[36]\*h[36]+(double)x[37]\*h[37]+(double)x[38]\*h[38]+(double)x[39]\*h[39]+(double)x[40]\*h[40]+(double)x[41]\*h[41]+(double)x[42]\*h[42]+(double)x[43]\*h[43]+(double)x[44]\*h[44]+(double)x[45]\*h[45]+(double)x[46]\*h[46]+(double)x[47]\*h[47]+(double)x[48]\*h[48]+(double)x[49]\*h[49]+(double)x[50]\*h[50];

y+=512;

DAC\_SetValue((int)trunc(y));

}

else

{

int y = (double)x[0]\*h[0]+(double)x[1]\*h[1]+(double)x[2]\*h[2]+(double)x[3]\*h[3]+(double)x[4]\*h[4]+(double)x[5]\*h[5]+(double)x[6]\*h[6]+(double)x[7]\*h[7]+(double)x[8]\*h[8]+(double)x[9]\*h[9]+(double)x[10]\*h[10]+(double)x[11]\*h[11]+(double)x[12]\*h[12]+(double)x[13]\*h[13]+(double)x[14]\*h[14]+(double)x[15]\*h[15]+(double)x[16]\*h[16]+(double)x[17]\*h[17]+(double)x[18]\*h[18]+(double)x[19]\*h[19]+(double)x[20]\*h[20]+(double)x[21]\*h[21]+(double)x[22]\*h[22]+(double)x[23]\*h[23]+(double)x[24]\*h[24]+(double)x[25]\*h[25]+(double)x[26]\*h[26]+(double)x[27]\*h[27]+(double)x[28]\*h[28]+(double)x[29]\*h[29]+(double)x[30]\*h[30]+(double)x[31]\*h[31]+(double)x[32]\*h[32]+(double)x[33]\*h[33]+(double)x[34]\*h[34]+(double)x[35]\*h[35]+(double)x[36]\*h[36]+(double)x[37]\*h[37]+(double)x[38]\*h[38]+(double)x[39]\*h[39]+(double)x[40]\*h[40]+(double)x[41]\*h[41]+(double)x[42]\*h[42]+(double)x[43]\*h[43]+(double)x[44]\*h[44]+(double)x[45]\*h[45]+(double)x[46]\*h[46]+(double)x[47]\*h[47]+(double)x[48]\*h[48]+(double)x[49]\*h[49]+(double)x[50]\*h[50];

DAC\_SetValue(trunc(y));

}

// x[0] = ADC\_GetResult16();

// y = x[0];

}

void Calc\_coef()

{

//Por estandar de C, el valor decimal de los dígitos en ASCII es el dígito en ASCII - '0', entonces eso es lo que hacemos

um\_frec = frec[0] - '0';

c\_frec = frec[1] - '0';

d\_frec = frec[2] - '0';

u\_frec = frec[3] - '0';

//Ahora acá procedemos a volver estos un número en frec\_final

frec\_final = (um\_frec)\*1000+(c\_frec)\*100+(d\_frec)\*10+u\_frec;

//miramos si frec\_final está dentro del rango

if(frec\_final>1000 || frec\_final == 0)

{

//Si no, le decimos al usuario que resetee el sistema

LCD\_ClearDisplay();

LCD\_Position(0,0);

LCD\_PrintString("Fuera de Rango");

LCD\_Position(1,0);

LCD\_PrintString("Presione Reset");

}

else

{

//Si sí viene la tarea engorrosa de calcular los coeficientes

volatile double wc = 2\*3.14159265\*(double)frec\_final/3000;

if(filtro == 1)

{

for(i = 0; i < 51; i++)

{

if(i != 25)

{

double temp = wc\*(i-25);

h[i] = (sin(temp));

h[i] /= (3.14159265\*(i-25));

}

else

{

h[i] = wc/3.14159265;

}

}

}

if(filtro == 2)

{

for(i = 0; i < 51; i++)

{

if(i != 25)

{

double temp = wc\*(i-25);

h[i] = -(sin(temp));

h[i] /= (3.14159265\*(i-25));

}

else

{

h[i] = 1 - wc/3.14159265;

}

}

}

if(filtro == 3)

{

double wc2 = ((double)frec\_final+100)\*3.14159265/1500;

double wc1 = ((double)frec\_final-100)\*3.14159265/1500;

for(i = 0; i < 51; i++)

{

if(i != 25)

{

double temp2 = wc2\*(i-25);

double temp1 = wc1\*(i-25);

h[i] = (sin(temp2));

h[i] /= (3.14159265\*(i-25));

double temp3 = (sin(temp1));

temp3 /= (3.14159265\*(i-25));

h[i] -= temp3;

}

else

{

h[i] = wc2/3.14159265 - wc1/3.14159265;

}

}

}

//Ahora después de calculados notificamos al usuario que fueron calculados para esa frecuencia

LCD\_ClearDisplay();

LCD\_Position(0,0);

LCD\_PrintString("Coef Calculados");

LCD\_Position(1,0);

LCD\_PrintString("Para F:");

LCD\_PutChar(frec[0]);

LCD\_PutChar(frec[1]);

LCD\_PutChar(frec[2]);

LCD\_PutChar(frec[3]);

CyDelay(3000);

LCD\_ClearDisplay();

LCD\_Position(0,0);

LCD\_PrintString("Inicia");

LCD\_Position(1,0);

LCD\_PrintString("Filtrado");

ADC\_Start();

DAC\_Start();

Muestreo\_Start();

Recorre\_Uno\_Stop();

int\_EndOfConv\_StartEx(llegaMuestra);

int\_CambioUno\_Stop();

int\_TeclaOprimida\_Stop();

}

}

//Esta función se ejecutará cada que se termine una conversión del ADC;

//Con este método identificaremos la tecla oprimida del teclado matricial

CY\_ISR(identificarTecla)

{

columna = Columnas\_Read();

if(fila == 1)

{

switch(columna)

{

case 0b00000001:

tecla = '1';

break;

case 0b00000010:

tecla = '2';

break;

case 0b00000100:

tecla = '3';

break;

}

}

if(fila == 2)

{

switch(columna)

{

case 0b00000001:

tecla = '4';

break;

case 0b00000010:

tecla = '5';

break;

case 0b00000100:

tecla = '6';

break;

}

}

if(fila == 4)

{

switch(columna)

{

case 0b00000001:

tecla = '7';

break;

case 0b00000010:

tecla = '8';

break;

case 0b00000100:

tecla = '9';

break;

}

}

if(fila == 8)

{

switch(columna)

{

case 0b00000001:

tecla = '\*';

break;

case 0b00000010:

tecla = '0';

break;

case 0b00000100:

tecla = '#';

break;

}

}

//Empezamos a mirar la lógica del algoritmo de visualización

if(paso == 3)

{

//Procedemos a calcular los coeficientes luego de colocar \*

if(tecla == '\*')

{

LCD\_Position(0,0);

LCD\_PrintString(" ");

LCD\_Position(0,0);

LCD\_PrintString("Calculando Coef");

LCD\_Position(1,0);

LCD\_PrintString(" ");

LCD\_Position(1,0);

LCD\_PrintString("...");

Calc\_coef();

}

}

if(paso == 2)

{

//Digitamos cada uno de los valores de unidades de mil de frecuencia, centenas de frecuencia, decenas de frecuencia y unidades de frecuencia

if((tecla - '0') != filtro)

{

frec[cont\_char] = tecla;

LCD\_PutChar(tecla);

cont\_char++;

if(cont\_char == 4)

{

LCD\_Position(0,0);

LCD\_PrintString(" ");

LCD\_Position(0,0);

LCD\_PrintString("Presione \*");

CyDelay(2000);

paso = 3;

}

}

else

{

tecla = filtro + '0';

frec[cont\_char] = tecla;

LCD\_PutChar(tecla);

cont\_char++;

if(cont\_char == 4)

{

LCD\_Position(0,0);

LCD\_PrintString(" ");

LCD\_Position(0,0);

LCD\_PrintString("Presione \*");

CyDelay(2000);

paso = 3;

}

}

}

if(paso == 1)

{

//Miramos qué tipo de filtro digita

switch(tecla)

{

case '1':

filtro = 1;

paso = 2;

break;

case '2':

filtro = 2;

paso = 2;

break;

case '3':

filtro = 3;

paso = 2;

break;

}

}

CyDelay(250);

Columnas\_ClearInterrupt();

}

//Con este método hacemos que el uno recorra las filas para recibirlo en las columnas

CY\_ISR(cambiarUno)

{

switch(fila)

{

case 1:

fila = 2;

break;

case 2:

fila = 4;

break;

case 4:

fila = 8;

break;

case 8:

fila = 1;

break;

}

Filas\_Write(fila);

}

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

/\* Place your initialization/startup code here (e.g. MyInst\_Start()) \*/

//Inicializamos los componentes

LCD\_Start();

Recorre\_Uno\_Start();

//Fin inicialización componentes

//Inicializamos las interrupciones

int\_TeclaOprimida\_StartEx(identificarTecla);

int\_CambioUno\_StartEx(cambiarUno);

//Fin inicialización de interrupciones

//Inicializamos variables

fila = 1;

paso = 1;

cont\_char = 0;

u\_frec = 0;

d\_frec = 0;

c\_frec = 0;

um\_frec = 0;

frec\_final = 0;

//Fin inicialización variables

for(;;)

{

//En el for infinito vamos a lidiar con la visualización

if(paso == 1)

{

//Preguntamos por el tipo de filtro

LCD\_PrintString("Filtro: 1)P.Bajas");

LCD\_Position(1,0);

LCD\_PrintString("2)P.Alta 3)P.Banda");

}

if(paso == 2)

{

//Pedimos la frecuencia del filtro intermitentemente

if(cont\_char == 0)

{

LCD\_ClearDisplay();

int\_TeclaOprimida\_Stop();

LCD\_PrintString("Digite una frec");

LCD\_Position(1,0);

LCD\_PrintString("de 4 digitos");

CyDelay(2500);

LCD\_Position(0,0);

LCD\_PrintString(" ");

LCD\_Position(0,0);

switch(filtro)

{

case 1:

LCD\_PrintString("Digite F corte");

break;

case 2:

LCD\_PrintString("Digite F corte");

break;

case 3:

LCD\_PrintString("Digite F central");

break;

}

LCD\_Position(1,0);

LCD\_PrintString(" ");

LCD\_Position(1,0);

LCD\_PrintString("Frec:");

int\_TeclaOprimida\_StartEx(identificarTecla);

}

}

CyDelay(2500);

}

}