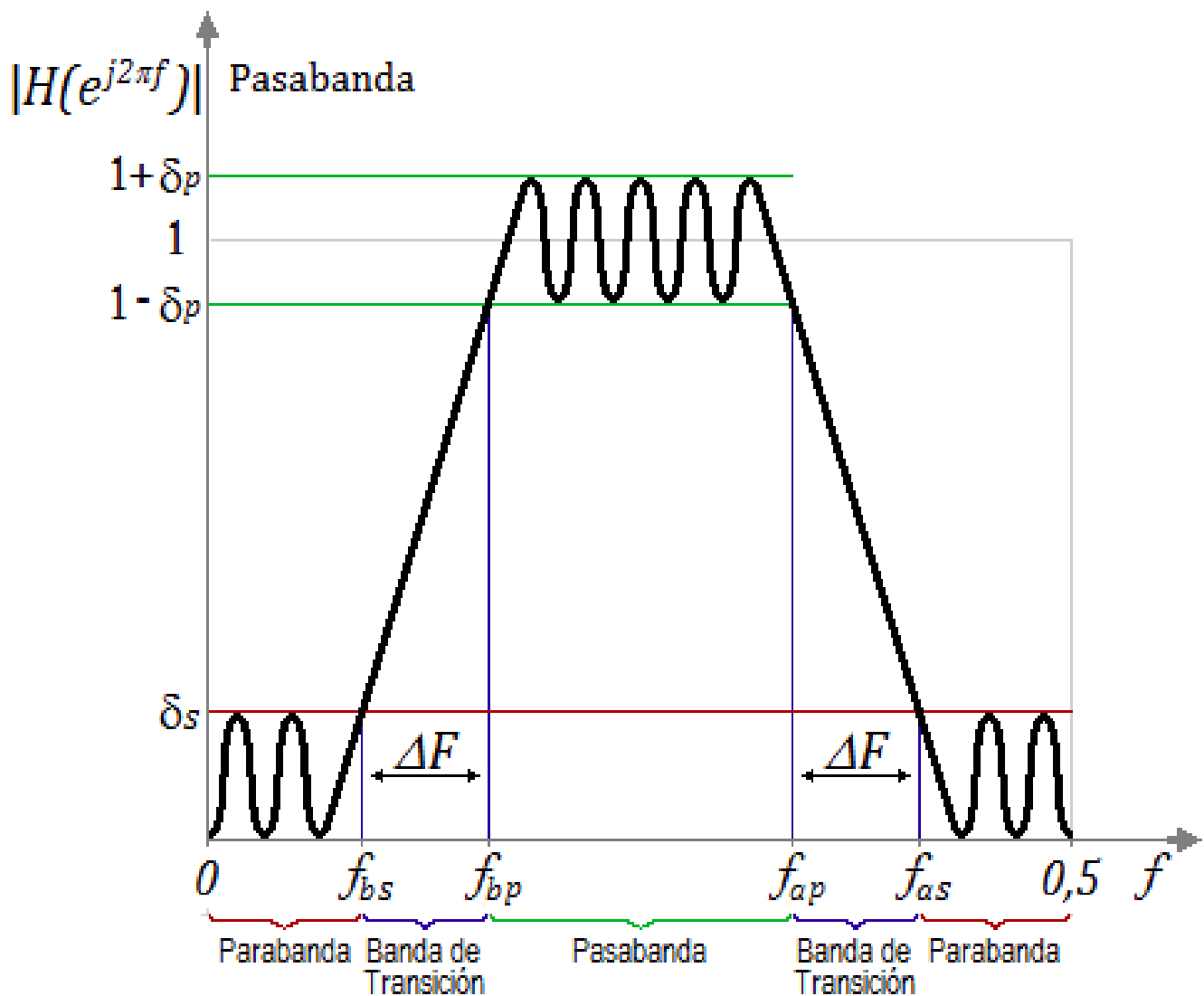


FILTRO FIR

Arquitecturas Avanzadas y de Propósito Específico



Kapil Ashok Melwani Chugani
alu0100883473@ull.edu.es
Universidad de La Laguna (ETSII)

ÍNDICE

Introducción: Página 3

Explicación Filtro FIR: Página 4

Desarrollo: Página 5

Medición de Ciclos: Páginas 6 – 7

Ejecuciones: Paginas 8-10

Valoración Personal: Página 11

INTRODUCCIÓN

Filtro Digital FIR – Finite Impulse Response (Respuesta al Impulso Finita)

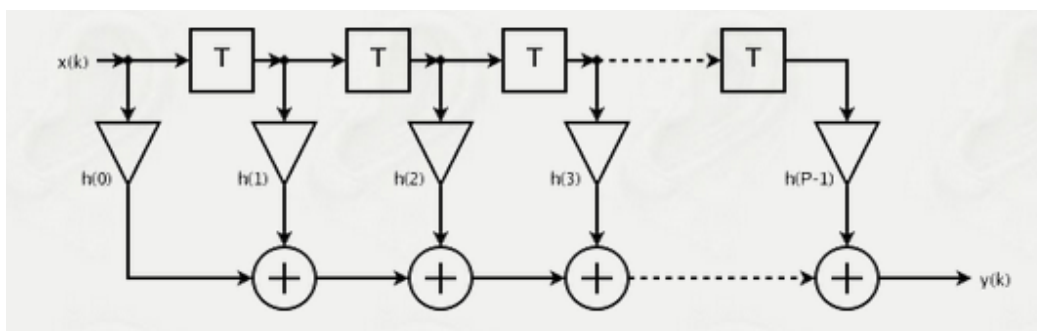
Es un tipo de filtro digital que si su entrada es un impulso (una delta de Kronecker) la salida será un numero limitado de términos no nulos. Para obtener la salida solo se emplean valores de la entrada actual y anteriores. También se llaman filtros digitales no recursivos. Su expresión en el dominio discreto es:

$$y_n = \sum_{k=0}^{N-1} b_k x(n-k)$$

El orden del filtro está dado por N, es decir, el numero de coeficientes. También la salida puede ser expresada como la convolución de una señal de entrada $x[n]$ con un filtro $h[n]$:

$$y_n = \sum_{k=0}^{N-1} h_k x_{n-k}$$

La estructura de un filtro FIR por tanto es la siguiente:



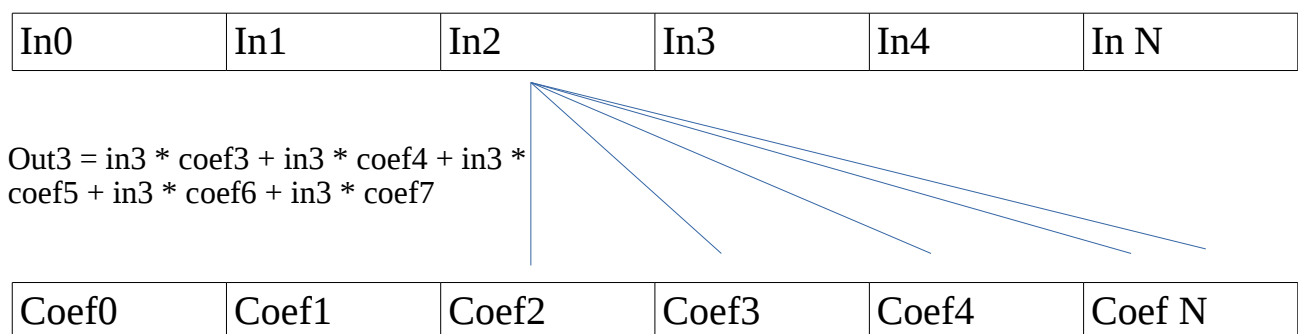
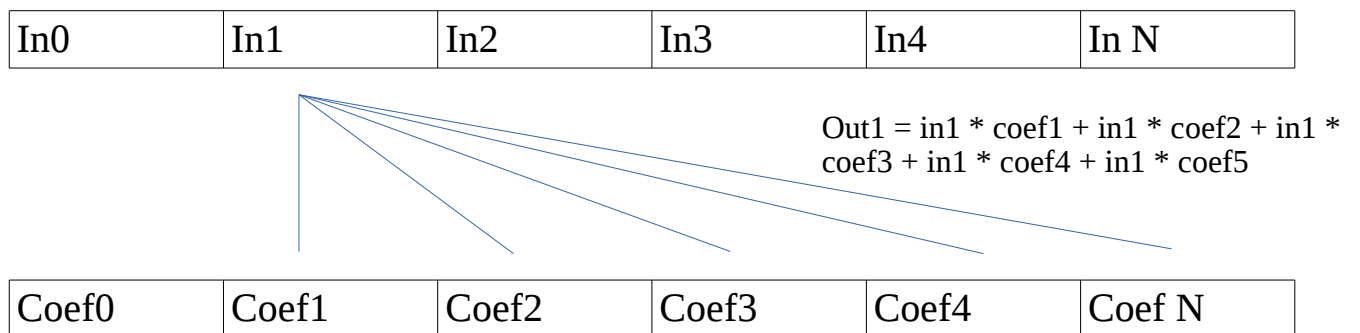
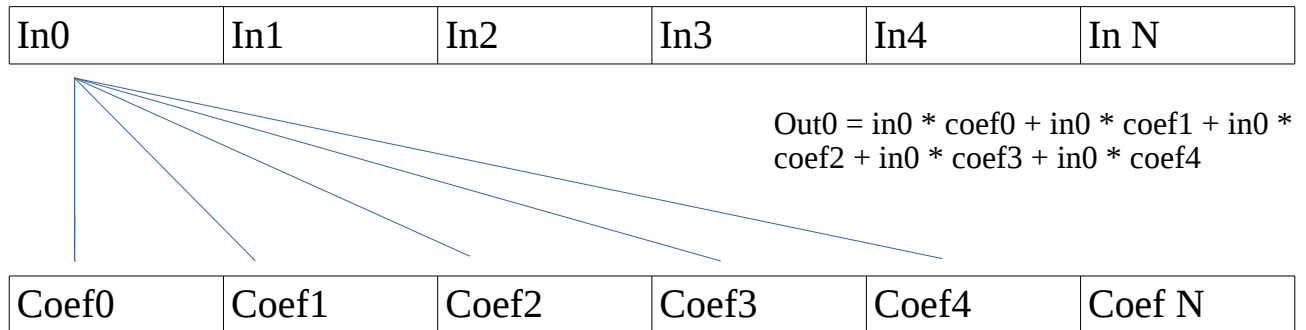
La cual puede verse reflejada en la aplicación de la transformada Z:

$$H(z) = \sum_{k=0}^{N-1} h_k z^{-k} = h_0 + h_1 z^{-1} + \dots + h_{N-1} z^{-(N-1)}$$

Los filtros FIR son estables puesto que solo tienen polo, es decir, elementos en el numerador en su función de transferencia. También tienen la ventaja que pueden diseñarse para ser de

fase lineal, es decir, no introducen desfases en la señal, a diferencia de los IIR o los filtros analógicos. Por ese motivo tienen interés en el audio.

EXPLICACIÓN FILTRO FIR



DESARROLLO

Se han realizado seis ficheros los cuales contienen:

- Filtro FIR Desenrollado.
- Filtro FIR con dos bucles Anidados en el main.
- Filtro FIR con dos bucles Anidados en una funcion.
- Filtro FIR Desenrollado con uso de Restrict.
- Filtro FIR con dos bucles Anidados en el main con uso de Restrict.
- Filtro FIR con dos bucles Anidados en una funcion con uso de Restrict.

Desenrollado

```
84
85 //APLICACION DEL UNROLLED FIR FILTER
86 vector_out[0] += vector_coef[0] * vector_in[0] + vector_coef[1] * vector_in[0] + vector_coef[2] * vector_in[0] + vector_coef[3] * vector_in[0] + vector_coef[4]
87 vector_out[1] += vector_coef[1] * vector_in[1] + vector_coef[2] * vector_in[1] + vector_coef[3] * vector_in[1] + vector_coef[4] * vector_in[1] + vector_coef[5]
88 vector_out[2] += vector_coef[2] * vector_in[2] + vector_coef[3] * vector_in[2] + vector_coef[4] * vector_in[2] + vector_coef[5] * vector_in[2] + vector_coef[6]
89 vector_out[3] += vector_coef[3] * vector_in[3] + vector_coef[4] * vector_in[3] + vector_coef[5] * vector_in[3] + vector_coef[6] * vector_in[3] + vector_coef[7]
90 vector_out[4] += vector_coef[4] * vector_in[4] + vector_coef[5] * vector_in[4] + vector_coef[6] * vector_in[4] + vector_coef[7] * vector_in[4] + vector_coef[8]
91 vector_out[5] += vector_coef[5] * vector_in[5] + vector_coef[6] * vector_in[5] + vector_coef[7] * vector_in[5] + vector_coef[8] * vector_in[5] + vector_coef[9]
92 vector_out[6] += vector_coef[6] * vector_in[6] + vector_coef[7] * vector_in[6] + vector_coef[8] * vector_in[6] + vector_coef[9] * vector_in[6] + vector_coef[10]
93 vector_out[7] += vector_coef[7] * vector_in[7] + vector_coef[8] * vector_in[7] + vector_coef[9] * vector_in[7] + vector_coef[10] * vector_in[7] + vector_coef[11]
94 vector_out[8] += vector_coef[8] * vector_in[8] + vector_coef[9] * vector_in[8] + vector_coef[10] * vector_in[8] + vector_coef[11] * vector_in[8] + vector_coef[12]
95 vector_out[9] += vector_coef[9] * vector_in[9] + vector_coef[10] * vector_in[9] + vector_coef[11] * vector_in[9] + vector_coef[12] * vector_in[9] + vector_coef[13]
96
97
```

Dos Bucles Anidados en Main:

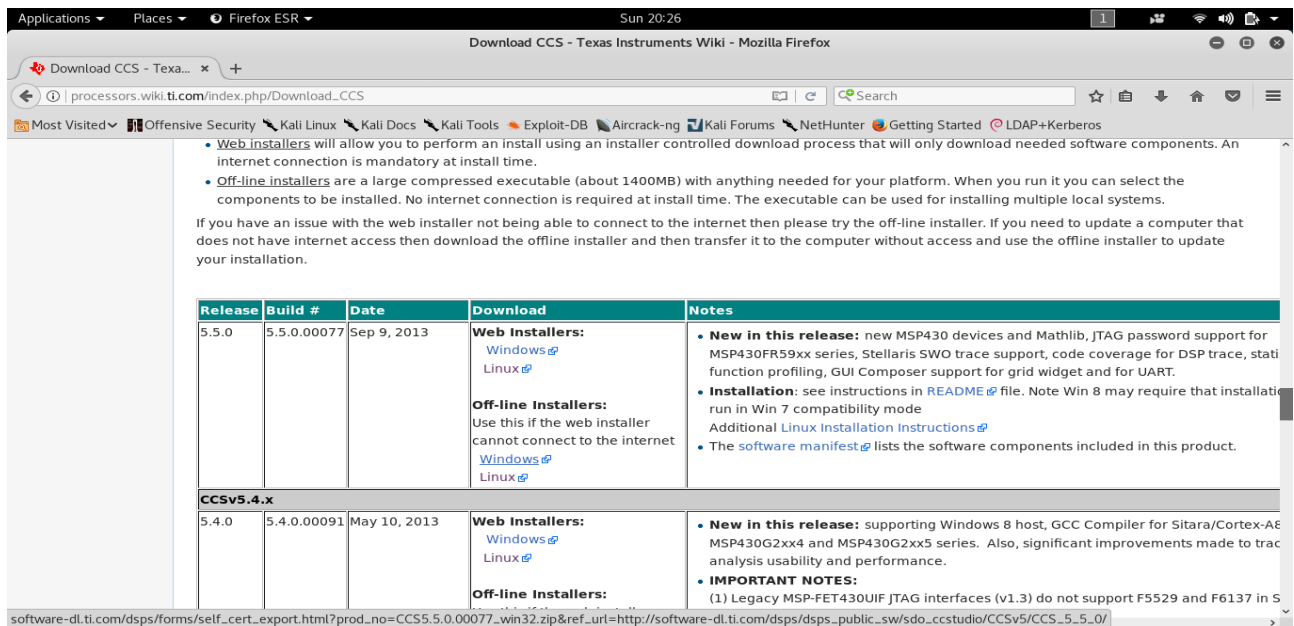
```
84
85 //APLICACION DEL FIR FILTER (CON DOS BUCLES ANIDADOS EN EL MAIN)
86 for(i=0; i<SAMPLES+COEF-1; i++){
87     vector_out[i] = 0;
88     for(int j=0; j<COEF ; j++){
89         vector_out[i] += vector_coef[i+j] * vector_in[i];
90     }
91 }
```

Dos Bucles Anidados en Funcion:

```
63
64 float* firfilter(float* vector_coef, float* vector_in){
65     float* vector_salida = (float *) malloc((SAMPLES+COEF-1) * sizeof(float));
66     int i;
67     int j;
68     for(i=0; i<SAMPLES+COEF-1; i++){
69         vector_salida[i] = 0;
70         for(j=0; j<COEF ; j++){
71             vector_salida[i] += vector_coef[i+j] * vector_in[i];
72         }
73     }
74     return vector_salida;
75 }
76
```

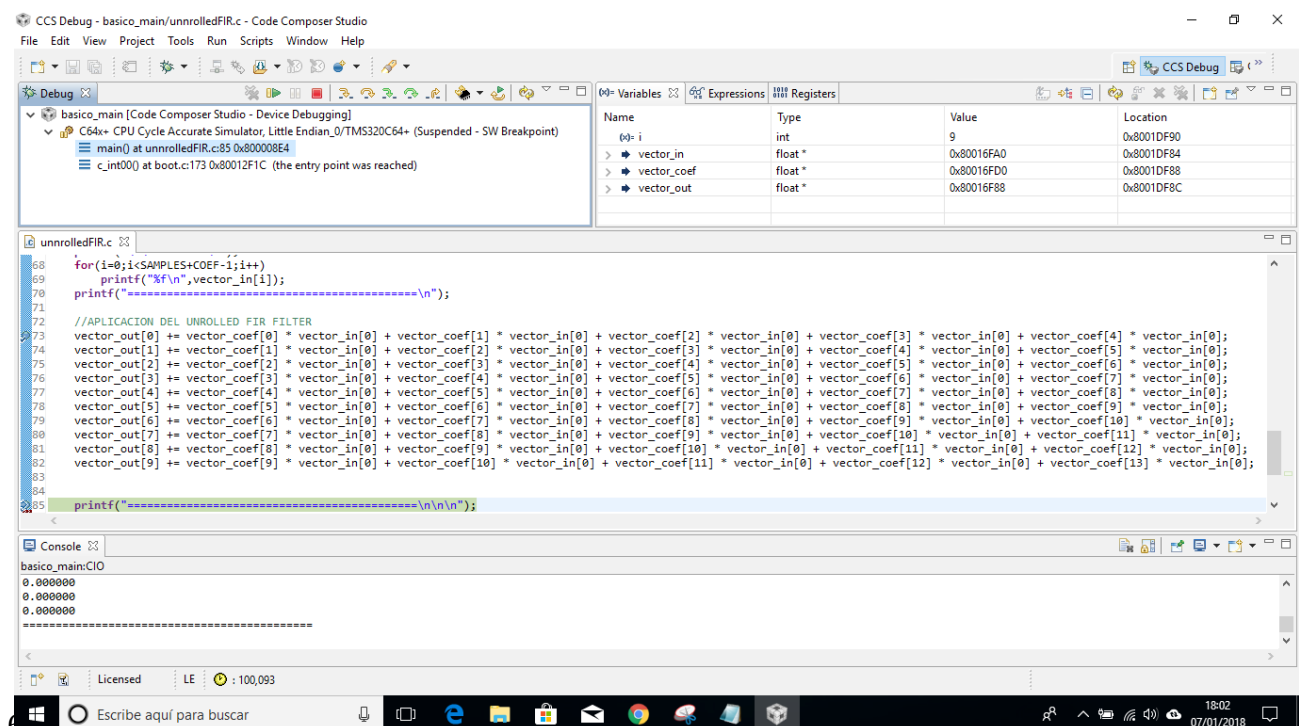
MEDICIÓN DE CICLOS

Para la medición de Ciclos hemos tenido que descargarnos el programa “Code Composer Studio” CCSv5.5

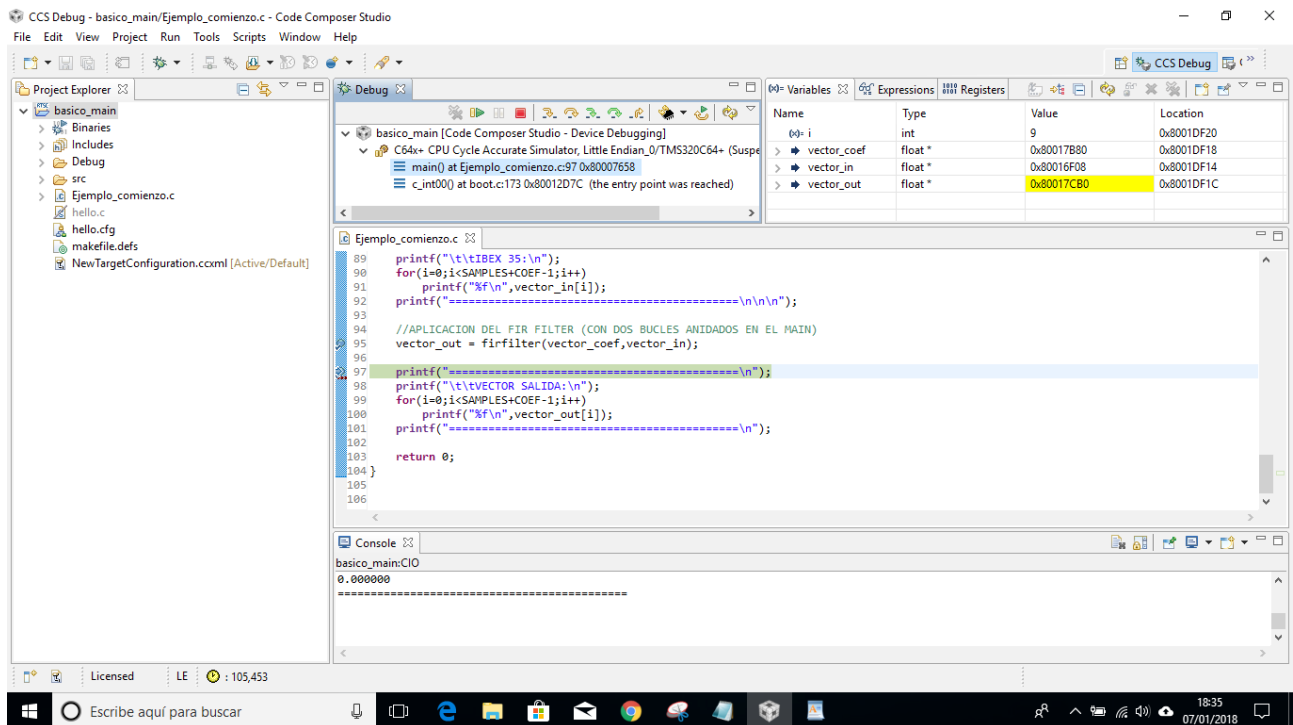


Para descargarlo también hemos tenido que registrarnos en: <http://www.ti.com>

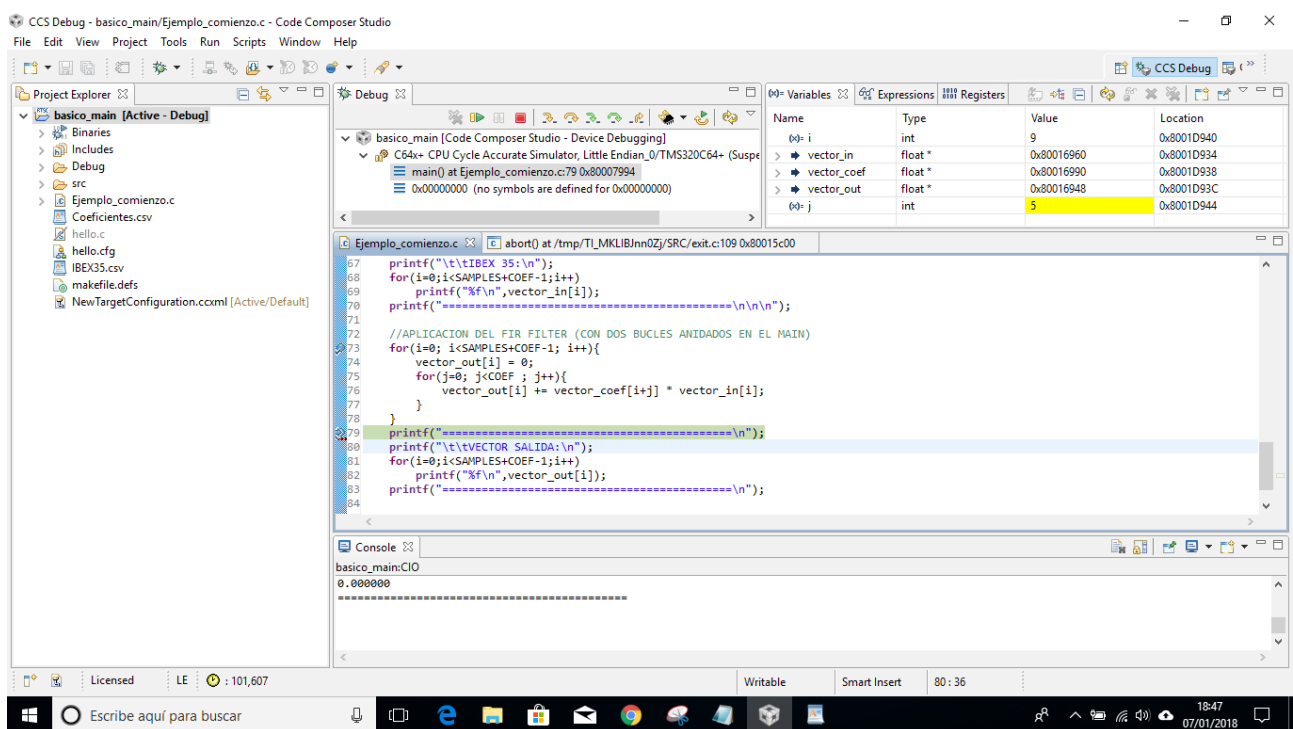
Desenrollado: 100.093 ciclos de reloj la ejecución del main hasta el final del filtro FIR.



Dos bucles Anidados en el main: 101,607 ciclos de reloj desde el main hasta el final de la ejecución del filtro FIR



Dos bucles Anidados en una funcion: 101.607 ciclos de reloj desde el main hasta el final de la ejecucion del filtro fir en la funcion.



EJECUCIONES

- Desenrollado

```
gcc unnrrolledfir.c  
./a.out
```

```
Applications ▾ Places ▾ Terminal ▾ Sun 19:44 1
root@kali: ~/Desktop/FILTRO FIR
File Edit View Search Terminal Help
root@kali:~/Desktop/FILTRO FIR# gcc unnrrolledfir.c
root@kali:~/Desktop/FILTRO FIR# ./a.out
=====
COEFICIENTES:
=====
0.004913
0.019350
0.022430
-0.009160
-0.029652
=====
=====
IBEX 35:
=====
11333.900391
11446.400391
11425.700195
11161.000000
11179.400391
11240.200195
11375.000000
11537.299805
11535.200195
=====
FILTRO FIR
=====
VECTOR SALIDA:
=====
89.324768
245.572189
24.040848
-226.857346
391.667969
1246.400879
1576.606201
2132.135986
=====
root@kali:~/Desktop/FILTRO FIR#
```

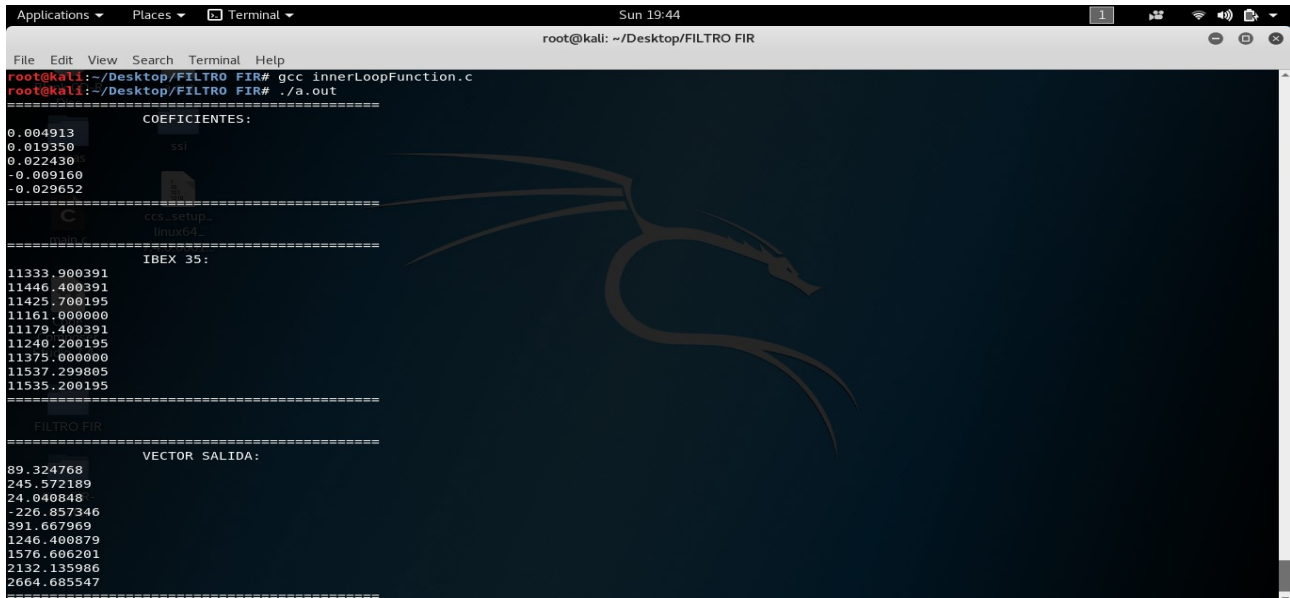
- Dos Bucles Anidados en el Main

```
gcc innerLoopMain.c  
./a.out
```

```
Applications ▾ Places ▾ Terminal ▾ Sun 19:44 1
root@kali: ~/Desktop/FILTRO FIR
File Edit View Search Terminal Help
root@kali:~/Desktop/FILTRO FIR# gcc innerLoopMain.c
root@kali:~/Desktop/FILTRO FIR# ./a.out
=====
COEFICIENTES:
=====
0.004913
0.019350
0.022430
-0.009160
-0.029652
=====
=====
IBEX 35:
=====
11333.900391
11446.400391
11425.700195
11161.000000
11179.400391
11240.200195
11375.000000
11537.299805
11535.200195
=====
FILTRO FIR
=====
VECTOR SALIDA:
=====
89.324768
245.572189
24.040848
-226.857346
391.667969
1246.400879
1576.606201
2132.135986
2664.685547
=====
root@kali:~/Desktop/FILTRO FIR#
```


- Dos Bucles Anidados en la Funcion

```
gcc innerLoopFunction.c
./a.out
```



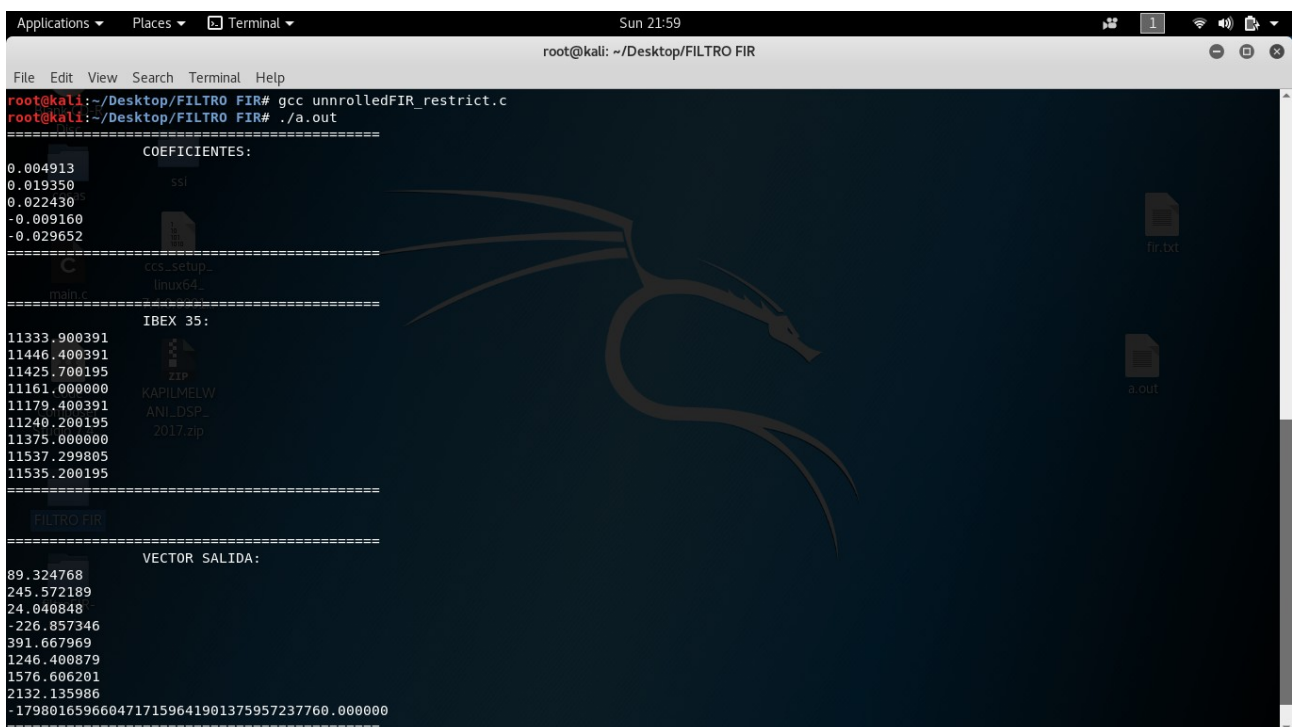
```

root@kali: ~/Desktop/FILTRO FIR
root@kali:~/Desktop/FILTRO FIR# gcc innerLoopFunction.c
root@kali:~/Desktop/FILTRO FIR# ./a.out
=====
COEFICIENTES:
=====
0.004913
0.019350
0.022430
-0.009160
-0.029652
=====
IBEX 35:
=====
11333.900391
11446.400391
11425.700195
11161.000000
11179.400391
11240.200195
11375.000000
11537.299805
11535.200195
=====
FILTRO FIR
=====
VECTOR SALIDA:
=====
89.324768
245.572189
24.040848
-226.857346
391.667969
1246.400879
1576.606201
2132.135986
2664.685547
=====

```

- Desenrollado Restrict

```
gcc unnrrolledfir_restrict.c
./a.out
```



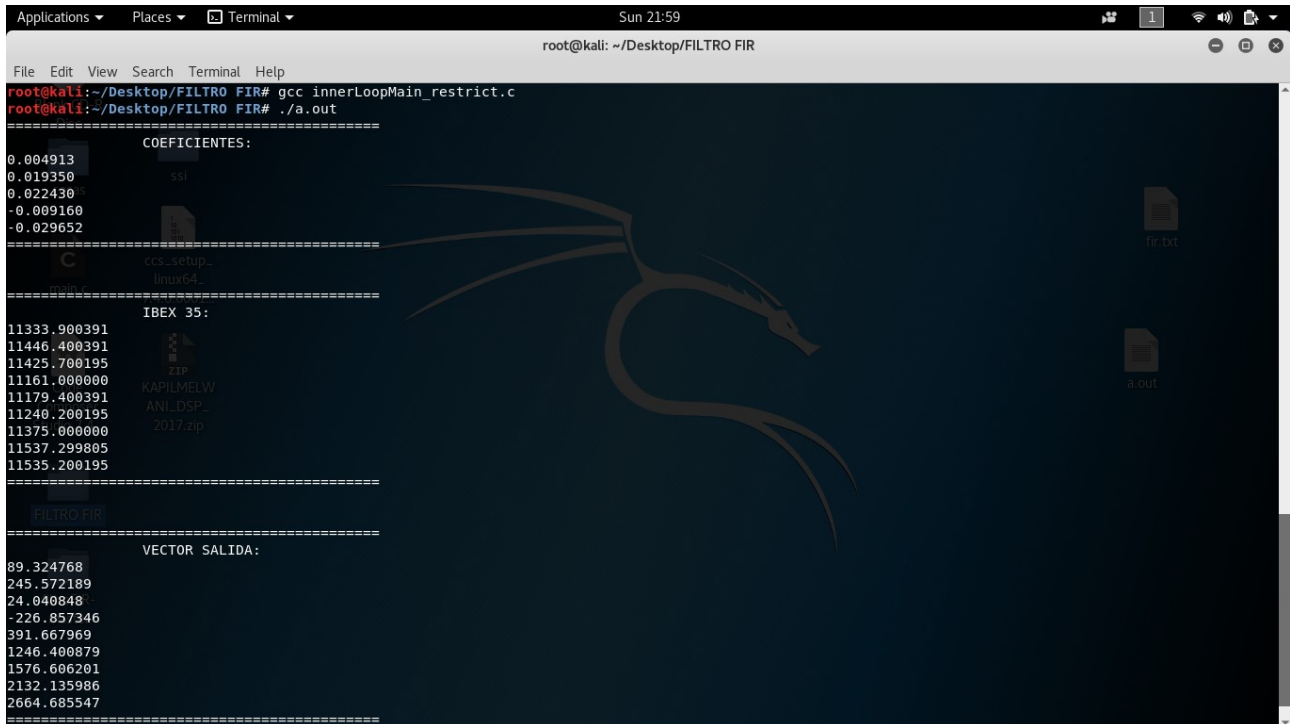
```

root@kali: ~/Desktop/FILTRO FIR
root@kali:~/Desktop/FILTRO FIR# gcc unnrrolledfir_restrict.c
root@kali:~/Desktop/FILTRO FIR# ./a.out
=====
COEFICIENTES:
=====
0.004913
0.019350
0.022430
-0.009160
-0.029652
=====
IBEX 35:
=====
11333.900391
11446.400391
11425.700195
11161.000000
11179.400391
11240.200195
11375.000000
11537.299805
11535.200195
=====
FILTRO FIR
=====
VECTOR SALIDA:
=====
89.324768
245.572189
24.040848
-226.857346
391.667969
1246.400879
1576.606201
2132.135986
-1798016596604717159641901375957237760.000000
=====

```

- Dos Bucles Anidados en el Main

```
gcc innerLoopMain_restrict.c
./a.out
```



```

Applications ▾ Places ▾ Terminal ▾ Sun 21:59
root@kali: ~/Desktop/FILTRO FIR

File Edit View Search Terminal Help

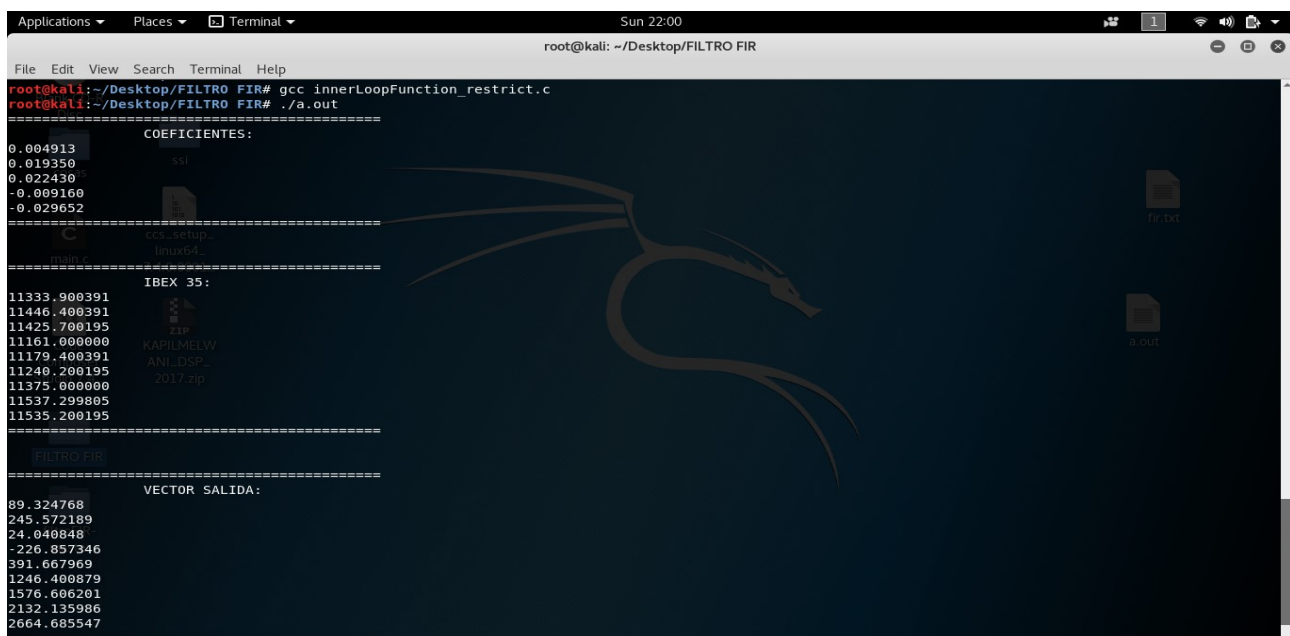
root@kali:~/Desktop/FILTRO FIR# gcc innerLoopMain_restrict.c
root@kali:~/Desktop/FILTRO FIR# ./a.out

=====
COEFICIENTES:
0.004913
0.019350
0.022430
-0.009160
-0.029652
=====
main.c
=====
IBEX 35:
11333.900391
11446.400391
11425.700195
11161.000000
11179.400391
11240.200195
11375.000000
11537.299805
11535.200195
=====
FILTRO FIR
=====
VECTOR SALIDA:
89.324768
245.572189
24.040848
-226.857346
391.667969
1246.400879
1576.606201
2132.135986
2664.685547
=====

```

- Dos Bucles Anidados en la Funcion

```
gcc innerLoopFunction_restrict.c
./a.out
```



```

Applications ▾ Places ▾ Terminal ▾ Sun 22:00
root@kali: ~/Desktop/FILTRO FIR

File Edit View Search Terminal Help

root@kali:~/Desktop/FILTRO FIR# gcc innerLoopFunction_restrict.c
root@kali:~/Desktop/FILTRO FIR# ./a.out

=====
COEFICIENTES:
0.004913
0.019350
0.022430
-0.009160
-0.029652
=====
main.c
=====
IBEX 35:
11333.900391
11446.400391
11425.700195
11161.000000
11179.400391
11240.200195
11375.000000
11537.299805
11535.200195
=====
FILTRO FIR
=====
VECTOR SALIDA:
89.324768
245.572189
24.040848
-226.857346
391.667969
1246.400879
1576.606201
2132.135986
2664.685547
=====

```

Valoración Personal

Una vez adquiridos todos los conceptos del Filtrado FIR, haber realizado los correspondientes programas en lenguaje c, podemos concluir tras haber analizado el número de ciclos que tarda el FIR en ejecutarse que el más optimo es el Desenrollado, tiene la gran desventaja de que en nuestro caso el numero de Muestras y Coeficientes era 5, pero en los casos en los que aumente a tamaños mucho mayores, la extension del codigo se alargaria demasiado. Por tanto, para que nuestro programa se muestre de una forma más eficiente, utilizaria el Filtro Fir con dos bucles anidados en una funcion con restrict puesto que la diferencia con el de dos bucles anidados en el main es practicamente nula.