Theremino

FILTRI DIGITALI

7 May 2017

Semplice implementazione dei filtri digitali

A differenza di quanto può apparire leggendo le pubblicazioni su questo argomento, si possono implementare i filtri digitali con poche righe di software. Abbiamo quindi preparato una applicazione efficiente sotto l'aspetto computazionale ma nel contempo semplice da leggere e facile da capire.

La applicazione **Theremino Filters** è minimale e didattica. L'algoritmo del filtro sta tutto in una ventina di righe e il resto della applicazione (interfaccia utente e comunicazione con le altre applicazioni del sistema Theremino) in poche pagine.

Questa applicazione non è uno dei soliti esempi teorici che si trovano nelle pubblicazioni in rete, ma è realmente utilizzabile per filtrare i segnali provenienti dai sensori. In questo modo si possono fare esperimenti su dati reali e non solo teoria matematica, la quale spesso porterebbe a sovrastimare alcuni aspetti e trascurarne altri.

Infine la applicazione Theremino Filters è scritta in un linguaggio moderno (VbNet) e può quindi facilmente essere portata su altri linguaggi come CSharp, Java e Python.

Trattazione matematica dei filtri digitali

Esistono centinaia di pubblicazioni in italiano e migliaia in inglese su questo argomento, inutile scriverne altre.

Per chi fosse interessato alla trattazione matematica consiglio documenti come i seguenti:

- http://infocom.uniroma1.it/~parisi/Cap2_Uncini.pdf
- http://lesc.det.unifi.it/sites/default/files/Documenti/Progetto%20IIR.pdf
- http://lesc.det.unifi.it/sites/default/files/Documenti/Progetto%20FIR.pdf
- http://risorse.dei.polimi.it/dsp/courses/fens/books/rocca-fens.pdf
- http://ens.di.unimi.it/dispensa/cap8.pdf
- http://www-lar.deis.unibo.it/people/crossi/files/Filtri-digitali-col.pdf

Oppure su ElectroYOU:

- http://www.electroyou.it/g.schgor/wiki/articolo21
- http://www.electroyou.it/g.schgor/wiki/articolo22
- http://www.electroyou.it/g.schgor/wiki/articolo23

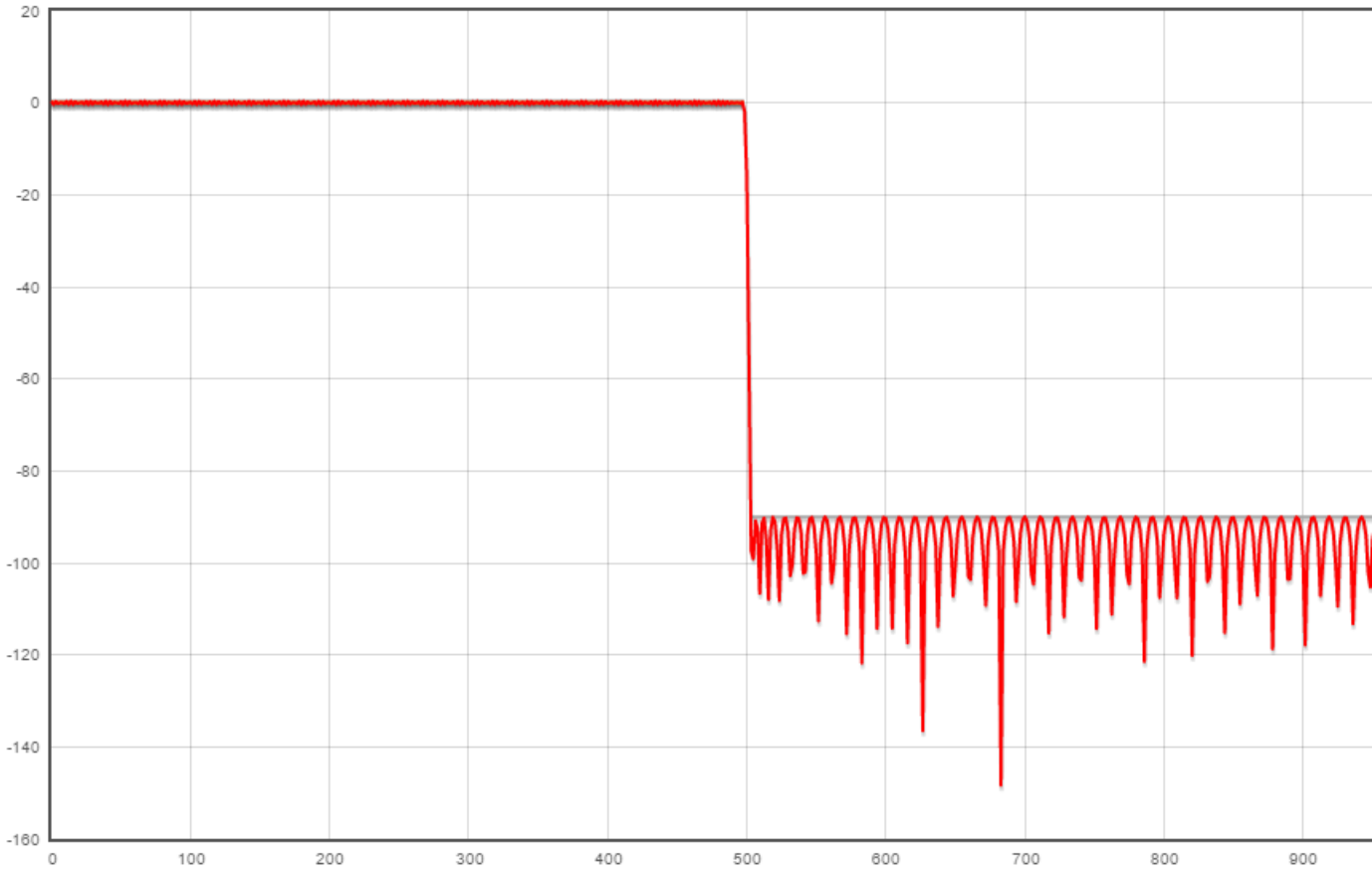
Oppure per chi preferisce il formato "a schede" (tipo Power Point):

- http://www2.units.it/marsi/filtri/Filtri%20digitali%20IIR.ppt
- http://www2.units.it/marsi/filtri/Filtri%20FIR.ppt

Però anche con questa dovizia di articoli (e forse proprio per la loro quantità) quando qualcuno chiede come implementare un filtro digitale in software gli viene sostanzialmente detto "arrangiati". Le risposte sono così tante e l'argomento così complesso, da scoraggiare ogni ulteriore sperimentazione.

Io non sarei in grado di descrivere bene l'aspetto matematico e non ne avrei nemmeno il tempo. Ma c'è un aspetto che tutti gli autori trascurano, cioè mostrare (con un esempio realmente funzionante) che la effettiva implementazione di questi filtri può essere semplicissima, basta sapere "come si fa". E in questo posso sicuramente dare un contributo utile.

Caratteristiche dei filtri digitali



Qui si vede la curva di risposta di un filtro FIR con banda passante da 0 a 498 Hz entro un decibel e che da 502 Hz in su attenua di almeno 90 decibel. Per chi era abituato ai filtri analogici caratteristiche del genere sarebbero state impensabili e fuori da ogni logica. Una implementazione hardware di un filtro del genere sarebbe "mostruosa", richiederebbe chili di materiale (oltre 2000 componenti) e non produrrebbe la curva di risposta calcolata a causa delle tolleranze dei componenti fisici.

Filtri IIR e FIR

Sempre senza addentrarci nei particolari matematici ecco le principali differenze tra questi due tipi di filtri digitali.

Un filtro FIR è sempre causale e stabile. In parole semplici questo vuol dire che, dato un qualunque insieme di specifiche, esiste sempre un insieme di coefficienti che possono implementarlo e che in tutti i casi i valori prodotti giungono alla stabilità in un tempo finito. Questo però non vuol dire che

si possono specificare parametri a caso o transizioni con ripidezza infinita. Perché in alcuni casi il numero di coefficienti potrebbe diventare molto alto, o addirittura infinito se si specificassero transizioni con larghezza zero.

I filtri IIR vengono spesso consigliati per la loro più semplice implementazione, e per le minori necessità di memoria e di potenza computazionale. Questo era vero in passato e lo è ancora quando si lavora con micro poco potenti che non hanno un coprocessore matematico. Ma quando si lavora con veri sistemi operativi (PC) e con linguaggi attuali, questi "vantaggi" sono talmente piccoli da non comportare nessuna differenza pratica.

Nei prossimi capitoli mostreremo che sui sistemi attuali, ad esempio Windows10 e su computer nemmeno molto potenti, ad esempio un Tablet quad core, un filtro FIR, implementato con algoritmi ben scritti, costituisce un carico trascurabile sia per la CPU che per il sistema operativo.

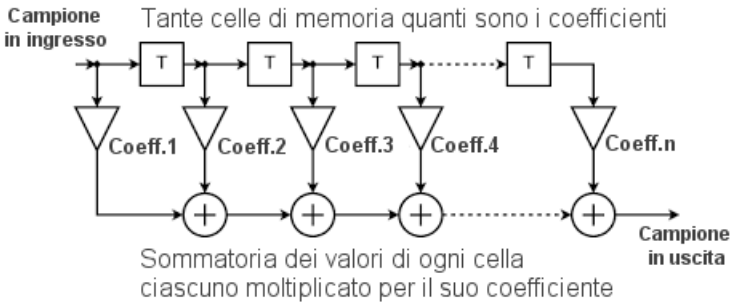
Per cui in questo articolo, da ora in poi, tralasceremo i filtri IIR e descriveremo solo i filtri FIR.

I coefficienti di un filtro FIR

Come sempre in programmazione lo stesso risultato si può ottenere in molti modi per cui esistono implementazioni dei filtri FIR complesse e difficili da capire. Spesso queste implementazioni sono anche poco efficienti perché comportano la copia di blocchi di memoria ad ogni nuovo campione. Ma è possibile implementare il filtro con un efficiente buffer circolare a due indici. Quindi un filtro FIR ben scritto può essere un algoritmo di appena una trentina di righe.

E' anche possibile descrivere questo algoritmo con parole semplici:

- Si prepara un area di memoria, che chiameremo "coda" con tanti posti quanti sono i coefficienti che si intendono utilizzare.
- Ad ogni nuovo campione in ingresso.
- Si aggiunge il nuovo campione in coda.
- Si fa uscire il campione più vecchio dalla testa della coda.
- Il campione di uscita è semplicemente la somma di tutti i campioni della coda moltiplicati per i rispettivi coefficienti.



Quindi per un filtro con 100 coefficienti la coda è lunga 100 posti e si eseguono 100 moltiplicazioni e 100 somme.

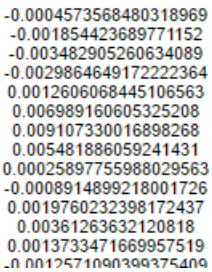
Tutta la magia di questi filtri non risiede nell'algoritmo ma nei coefficienti. O, più propriamente, nei loro valori numerici. La tabella dei coefficienti in se non è niente di speciale, sono un elenco di numeri positivi e negativi in virgola mobile.

Esistono algoritmi ed applicazioni molto comode per calcolare i coefficienti. Con queste applicazioni si possono specificare i parametri desiderati, vedere se la curva risultante risponde alle specifiche e infine copiare i coefficienti e inserirli nel filtro. Alcune di queste applicazioni utilizzano anche finestre di campionamento (Hamming, Hanning, Blackman ecc..) che migliorano alcune caratteristiche della funzione di trasferimento. Si possono anche specificare i tipi di filtri desiderati (Passa banda, Passa basso, Passa alto o Elimina banda), la attenuazione minima nelle bande proibite, l'ondulazione massima nella banda passante, le frequenze di inizio e fine transizione, e altri parametri.

La vera magia è che alla fine tutte queste caratteristiche vengono "compresse" in una unica lista di coefficienti. Questa lista sarà più o meno lunga a seconda delle caratteristiche desiderate. Si copia la lista nel software e il filtro avrà una curva di risposta "esattamente" uguale a quella progettata. Per chi, come me, ha passato i primi venti anni della vita a fare filtri con il saldatore questa è fantascienza. Le prime volte che ho scritto un filtro FIR non pensavo che potesse funzionare davvero, ci ho messo un po' a crederci.

Quanti coefficienti usare?

La quantità di coefficienti è grossolanamente paragonabile al numero di condensatori (o induttori) che costituirebbero un filtro hardware. Un centinaio di coefficienti potrebbero quindi essere equivalenti a un filtro con 100 celle RC (o RL) che potrebbero produrre una ripidezza fino a 600 decibel per ottava. Di fatto il numero di coefficienti non viene utilizzato solo per ottenere ripidezza ma anche per altre caratteristiche desiderabili, come la piattezza nella banda passante e la attenuazione nella banda proibita. Ma in tutti i casi già con filtri da un centinaio di coefficienti si possono ottenere prestazioni notevoli.



Ho fatto misure con filtri "esagerati" che comportano migliaia di coefficienti e il sistema (Windows10 su un semplice dual core) mi ha indicato un carico di lavoro per le CPU dello 0%

(cioè inferiore allo 0.1%). Per iniziare a notare qualche effetto sul carico (circa lo 0.5%) ho dovuto superare i diecimila coefficienti.

Anche l'occupazione di memoria è minima, si parla di decine di kbyte che sono niente per i computer attuali. Quindi il numero dei coefficienti non è limitato dal consumo di risorse.

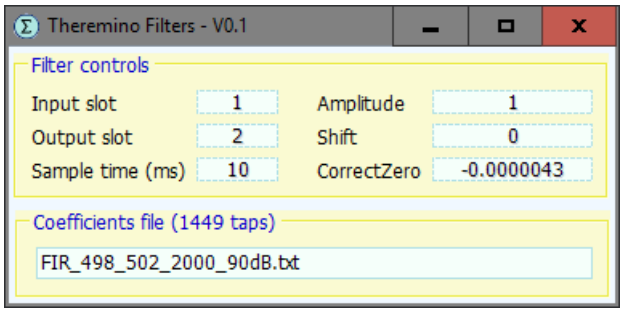
Il vero limite al numero di coefficienti è dato dal tempo di ritardo tra ingresso e uscita del filtro. Il ritardo cresce linearmente con il numero di coefficienti e si calcola, in modo grossolano e semplice, come il tempo di campionamento moltiplicato per la metà del numero di coefficienti.

Alcune applicazioni potrebbero tollerare lunghi tempi di ritardo ma in genere è sempre bene avere ritardi più bassi possibile. Per cui si cerca sempre di utilizzare il minimo numero di coefficienti che permette di ottenere le caratteristiche richieste.

La applicazione Theremino Filters

Molte applicazioni del sistema Theremino utilizzano filtri digitali, ad esempio **Theremino Geiger** e **Theremino MCA**, ma soprattutto **Theremino SDR** che ne utilizza vari tipi e calcola da sola i coefficienti. Ma sono applicazioni troppo complesse per utilizzarle come esempio. Per cui abbiamo scritto una piccola applicazione didattica.

Questa applicazione non è solo un esempio teorico ma è anche realmente utilizzabile come blocchetto modulare per filtraggi generici nel nostro sistema. I campioni di ingresso vengono letti da uno "Slot" e i campioni di uscita vengono scritti in un altro "Slot". In questo modo si possono filtrare segnali in arrivo dai sensori, nonché i segnali che transitano da una all'altra delle varie applicazioni.



Come si può vedere dalla interfaccia utente questa è una applicazione molto semplice. La implementazione software è altrettanto semplice ma nel contempo è quanto di meglio si può fare in termini di precisione ed efficienza. Tutti i calcoli vengono effettuati in virgola mobile con quindici cifre di precisione che corrispondono a oltre cinquanta bit o, in altri termini, a oltre trecento decibel di dinamica.

L'algoritmo del filtro FIR

```
Public Class FirFilter

   Private mCoeffs() As Double
   Private mNumCoeffs As Int32
   Private mHistory() As Double
   Private mCurrentIndex As Int32

   Friend Sub Init(ByVal Coeffs() As Double)
       mCoeffs = Coeffs
       mNumCoeffs = mCoeffs.Length
       mCurrentIndex = 0
       ReDim mHistory(mNumCoeffs - 1)
   End Sub

   Friend Sub PutSample(ByVal value As Double)
       mHistory(mCurrentIndex) = value
       mCurrentIndex += 1
       If mCurrentIndex = mNumCoeffs Then mCurrentIndex = 0
   End Sub

   Friend Function GetSample() As Double
       Dim output As Double = 0
       Dim index As Integer = mCurrentIndex
       For i As Int32 = 0 To mNumCoeffs - 1
           If index <> 0 Then
               index -= 1
           Else
               index = mNumCoeffs - 1
           End If
           output += mHistory(index) * mCoeffs(i)
       Next
       Return output
   End Function

End Class
```

L'algoritmo per implementare un filtro FIR è tutto qui.

- **Init** - Prima di utilizzare il filtro è necessario "inizializzarlo" e passargli un array che contiene i coefficienti. La funzione Init si occupa di individuare il numero di coefficienti dell'array e preparare tutto il necessario per il funzionamento del filtro.

- **PutSample** - Con questa funzione si aggiunge un nuovo campione alla "coda" dei campioni. La coda è implementata con un buffer circolare chiamato "History". L'indice di scrittura "CurrentIndex" viene incrementato di un posto e quando arriva alla fine viene riportato all'inizio (buffer circolare).
- **GetSample** - Con questa funzione si estrae un campione filtrato dalla "coda" dei campioni. L'indice di lettura "Index" viene incrementato lungo tutta la coda e quando arriva alla fine viene riportato all'inizio (buffer circolare). Vengono fatte tutte le moltiplicazioni e le somme e il dato filtrato viene riportato (con Return) alla funzione chiamante.

Per utilizzare il filtro prima lo si inizializza con i coefficienti desiderati e poi, ad intervalli di tempo regolari pari al tempo di campionamento, si inserisce un nuovo campione con "PutSample" e si legge un campione filtrato con "GetSample". Queste operazioni vengono eseguite nella classe "AccurateTimer" della nostra piccola applicazione.

Gli altri moduli della applicazione si occupano di presentare all'utente una interfaccia grafica, comunicare con le altre applicazioni del sistema Theremino, leggere il file dei coefficienti e leggere e scrivere i file di servizio.

Applicazioni per sperimentare con i filtri digitali

Con le applicazioni **Theremino SignalScope** e **Theremino WaveGenerator** è possibile provare i filtri, con segnali che viaggiano sugli Slot da una applicazione del sistema a un altra.

Nella immagine seguente si vede il WaveGenerator che invia un'onda quadra da un hertz allo Slot 1, e poi Theremino Flter che invia il segnale filtrato allo Slot 2. Infine il Signal Scope visualizza i due segnali.

Sperimentare l'effetto dei filtri sui campioni provenienti dai sensori e sui dati che le applicazioni del sistema si scambiano tra di loro, permette di acquisire esperienza pratica. Conoscendo solo la teoria si potrebbero sovrastimare alcuni aspetti e trascurarne altri, ma fare esperimenti su dati reali immunizza da questi pericoli.

E' importante rendersi conto che sovradimensionare alcune caratteristiche può produrre effetti negativi su altre, principalmente le seguenti:

- Overshoot (sovraelongazione) e ringing (oscillazioni) nella risposta ad una eccitazione a gradino.
- Tempo di ritardo (tempo occorrente perché l'uscita raggiunga il 50% del valore finale).
- Tempo di salita (tempo occorrente perché l'uscita passi dal 10 al 90 % del valore finale).
- Tempo di assestamento (tempo occorrente perché l'uscita si stabilizzi entro il +/-5% del valore finale).

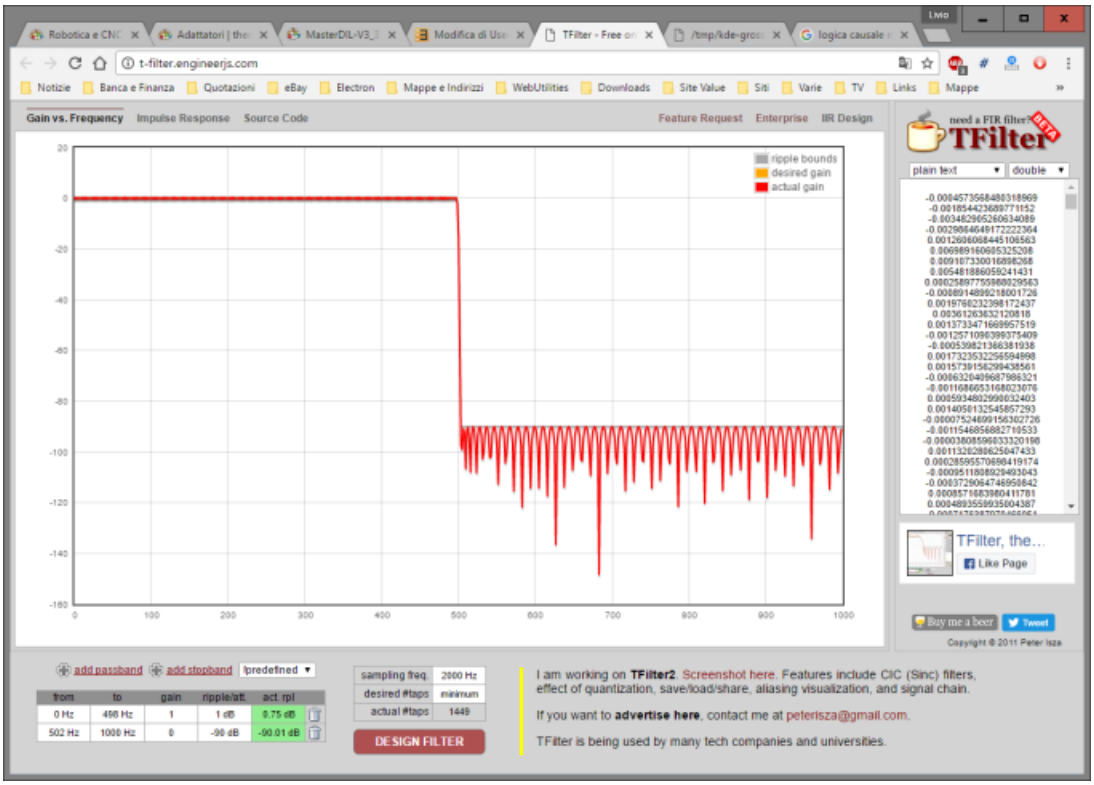
Generare i coefficienti

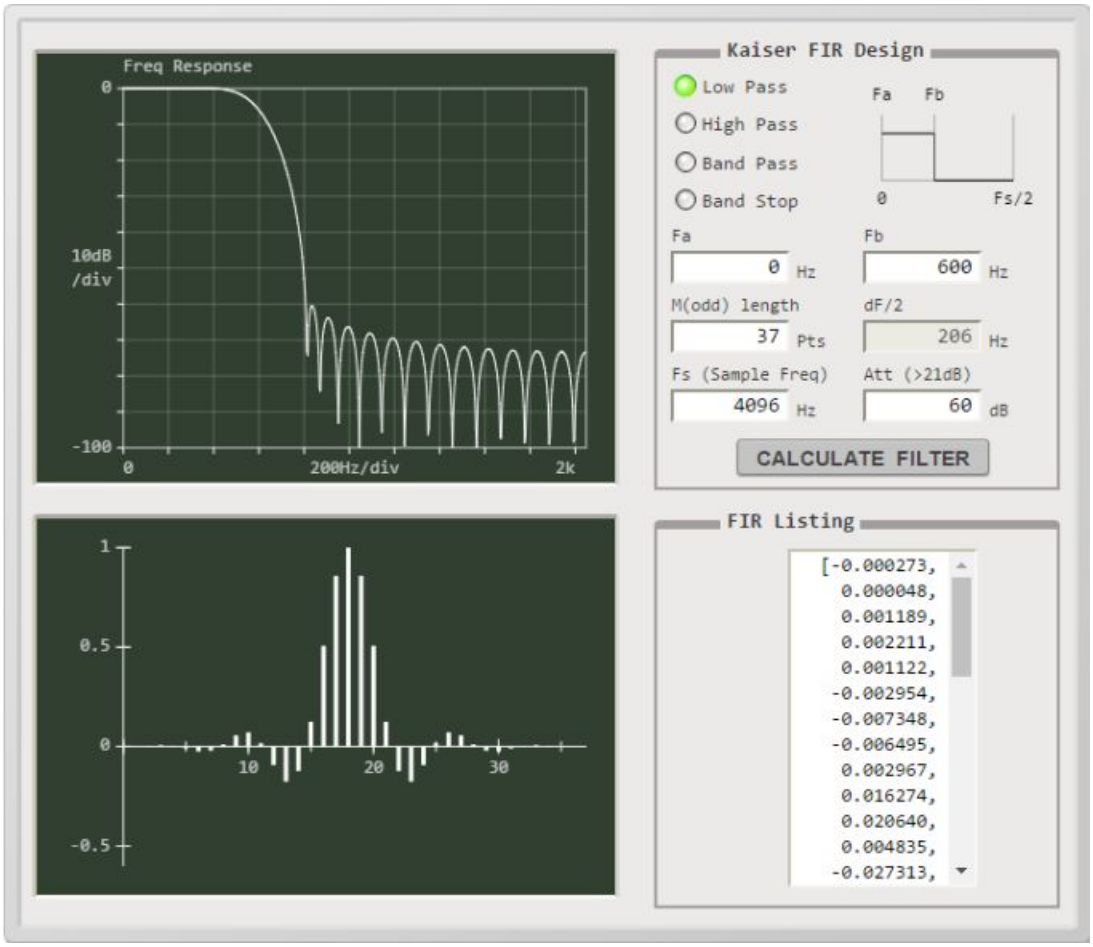
I coefficienti vanno scritti in semplici file di testo nella cartella "Coeffs" della applicazione "Theremino Filters". In questo modo sarà possibile sceglierli per nome e provare velocemente il comportamento di ogni versione e le differenze tra di esse.

Per generare i coefficienti si usano apposite applicazioni. Esistono applicazioni costose e difficili da utilizzare ma esistono anche ottime applicazioni gratuite che si possono utilizzare in rete senza nemmeno scaricarle.

In questi anni (2017) le applicazioni seguenti sono probabilmente le migliori:
http://t-filter.engineerjs.com
http://www.arc.id.au/FilterDesign.html

Ed ecco le immagini:


t-filter.engineerjs.com
Gain vs. Frequency
Impulse Response
Source Code
Feature Request
Enterprise
IIR Design
ripple bounds
desired gain
actual gain
TFilter
plain text
double
add passband
add stopband
predefined
from
to
gain
ripple/att.
act. rpl
0 Hz
486 Hz
1
1 dB
0.75 dB
502 Hz
1000 Hz
0
-90 dB
-90.01 dB
sampling freq.
2000 Hz
desired #taps
minimum
actual #taps
1449
DESIGN FILTER
I am working on TFilter2. Screenshot here. Features include CIC (Sinc) filters, effect of quantization, save/load/share, aliasing visualization, and signal chain.
If you want to advertise here, contact me at peterisza@gmail.com.
TFilter is being used by many tech companies and universities.
TFilter, the...
Like Page
Buy me a beer
Tweet
Copyright © 2011 Peter Isza



Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Theremino:filtri-digitali"