Emanuele T (EcoTan)

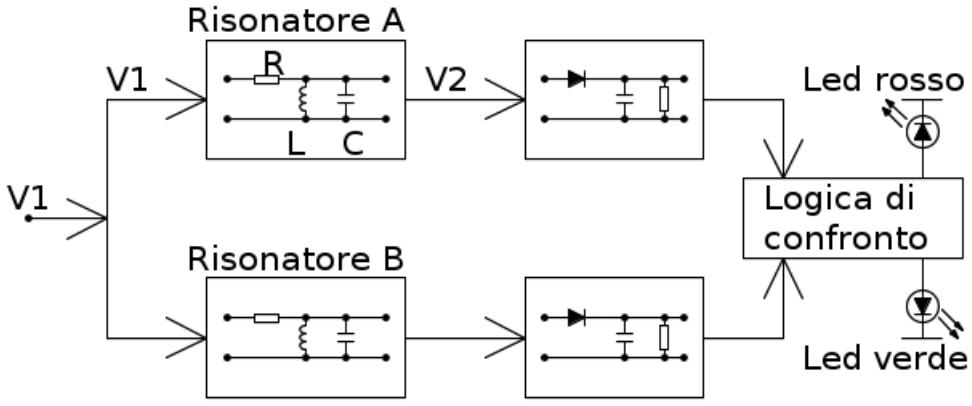
REALIZZAZIONE PRATICA DI UN SEMPLICE FILTRO DIGITALE

1 February 2014

Generalità

Mediante un microcontrollore opportunamente programmato, si possono svolgere le stesse funzioni di un circuito elettronico analogico. Mi riferisco alla possibilità di campionare i segnali in ingresso e simulare il circuito mediante operazioni alle differenze finite.
Svolgeremo un esempio pratico descrivendo una realizzazione che comprende un paio di semplici filtri, due rivelatori a valor massimo e una logica di confronto. Il tutto fa parte di un ricevitore per radiocomando a 3 canali On-Off in tecnica AFSK, ma è chiaro che non voglio fare concorrenza alla Futaba bensì soltanto servirmi di un esempio per mostrare come si possa simulare un filtro.
Non tratteremo l'argomento dei filtri in generale nè dei quadripoli nè tanto meno dei filtri digitali. Invece presupponiamo di avere un filtro già definito, nel senso che abbiamo già lo schema elettrico coi valori dei componenti o abbiamo già la funzione di trasferimento con poli e zeri, e desideriamo soltanto trovare il modo per simulare questo determinato filtro.
Assunte queste premesse, l'esempio che svolgeremo ha valore generale e può costituire un metodo valido per simulare qualsiasi filtro. I filtri di ordine superiore converrà considerarli come costituiti da più celle in cascata.

Il caso in esame



Con riferimento allo schema a blocchi quì sopra riportato, il problema che risolveremo è quello di simulare il filtro indicato come "Risonatore A".
Abbiamo lo schema elettrico e ne calcoliamo la funzione di trasferimento:

$$\frac{V_2}{V_1} = \frac{SL}{R + SL + S^2RLC}$$

Applichiamo "brutalmente" la seguente sostituzione bilineare, in cui T è l'intervallo di campionamento:

$$S = \frac{2}{T}\frac{(1 - Z)}{(1 + Z)}$$

Effettuata la sostituzione, l'operatore S sparisce e compare in sua vece l'operatore Z.
Poi portiamo a primo membro dell'eguaglianza la sola V_2, cioè portiamo a secondo membro tutti i termini che contengono V_1, e anche quelli che contengono V_2 accompagnata dal simbolo operatoriale Z.
Ci vuole un paio di pagine di calcoli, ma si tratta di passaggi algebrici elementari e non è richiesta alcuna notazione o conoscenza superiore. Il risultato è costituito dalla seguente formula, dall'aspetto alquanto cabalistico:

$$V_2 = \frac{-2RT^2 - 2LT + 4RLC}{RT^2 + 2LT + 4RLC} ZV_2 +$$
$$+ \frac{-3RT^2 + 2LT + 4RLC}{RT^2 + 2LT + 4RLC} Z^2V_2 +$$
$$+ \frac{-RT^2 + 2LT - 4RLC}{RT^2 + 2LT + 4RLC} Z^3V_2 +$$
$$+ \frac{2LT}{RT^2 + 2LT + 4RLC}(1 + Z - Z^2 - Z^3)V_1$$

A questo punto assumiamo che l'operatore Z significa "precedente" cioè ZV è il campione di segnale che precede V, Z^2V precede ZV e così via.
In sostanza, lo ADC (analog to digital converter) del microcontrollore è programmato per campionare il segnale di ingresso V_1 ad intervalli di tempo T, e la relativa ISR (Interrupt Servicing Routine) non deve fare altro che applicare la formula quì sopra scritta.
I coefficienti frazionari sono costanti (notiamo anche che hanno tutti lo stesso denominatore), dunque conviene calcolarli una volta per tutte all'inizio del Main Program, poi la ISR dovrà soltanto moltiplicare questi coefficienti costanti per le variabili, rappresentate dai segnali V1 campionati e V2 calcolati, e sommare; queste sono operazioni veloci fattibili in tempo reale.
Infine la ISR dovrà fare scorrere i valori delle variabili in memoria: ZV_1 diventa Z^2V_1 per il passo successivo, V_1 diventa ZV_1 e così via.
Ce la caviamo con pochissimi microsecondi di tempo di CPU, il microcontrollore resta sostanzialmente libero per fare anche altre cose.

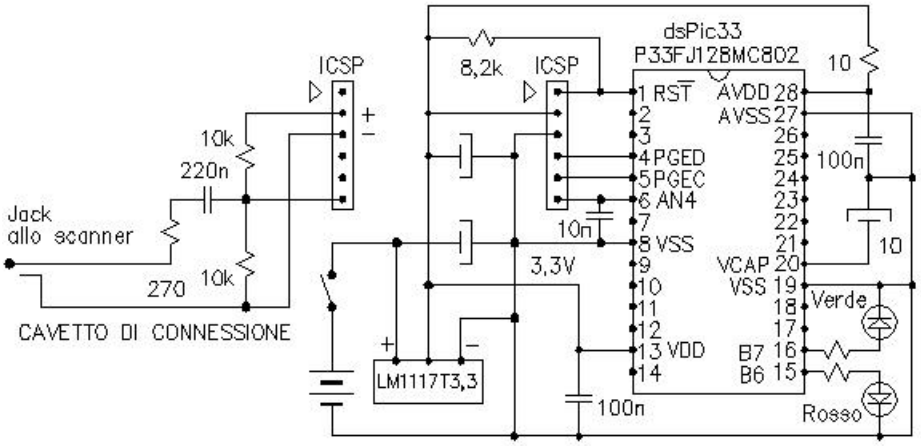
Formato dei dati numerici

I coefficienti costanti vengono calcolati in formato floating point e quì non ci sono problemi. Poi vanno convertiti in formato Integer per essere utilizzati dalla ISR, e quì qualche problema ci può essere. Se il valore è minore di uno, convertito in Integer diventerebbe zero e questo non va bene. Io ho usato il seguente metodo: esaminare il valore di ogni coefficiente (questo si può fare con lo stesso programma in fase di debug, oppure calcolandolo a parte), quindi moltiplicarlo per una opportuna potenza di 2 per scalarlo fino a un valore che sia comodo da rappresentare in formato Integer a 16 bit. Possiamo anche aggiungere 0.5 in floating al modulo per arrotondare la conversione in linguaggio C. Quando la ISR eseguirà la moltiplicazione di un coefficiente per una variabile, essendo due fattori entrambi a 16 bit, utilizzerà una variabile ausiliaria di appoggio Long Int a 32 bit. A questo punto bisogna dividere il risultato della moltiplicazione per la stessa potenza di 2 già utilizzata per scalare quel coefficiente, ricordando che la divisione può farsi velocemente con un semplice shift in memoria. Se tutto quadra, siamo tornati in rappresentazione a 16 bit e non siamo incorsi in nessun overflow, tenendo anche conto che lo ADC fornisce il segnale V_1 a 12 bit.
La resistenza R del filtro ha semplicemente valore R=1, T si assume circa pari a 73 microsecondi cioè 14 kSample/sec.

La programmazione

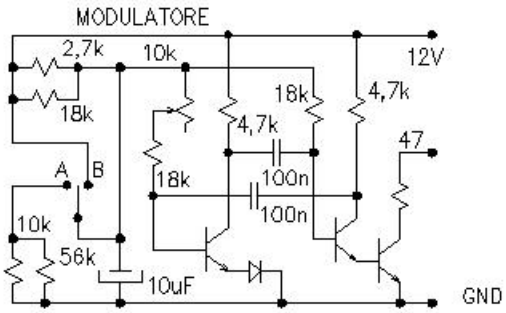
Fornisco il programma in linguaggio C: **electro.c**
Questo programma è stato compilato con il compilatore C30 versione 3.25 lite student free che ho scaricato dal sito Microchip, e poi riversato con il programmatore Pickit3 tramite il connettore ICSP indicato nello schema, sotto Mplab 8.60.
La ISR dello ADC simula i due filtri A e B rappresentati nello schema a blocchi, e simula anche i rispettivi rivelatori a valor massimo.
Il Main Program configura lo HardWare del microcontrollore, poi calcola i coefficienti costanti, avvìa lo ADC e infine si pone in loop per confrontare le uscite dei due filtri A e B. Il risultato del confronto viene segnalato tramite due Led, ma potrebbero anche essere due transistor che azionano dei relè o altro.

L'hardware

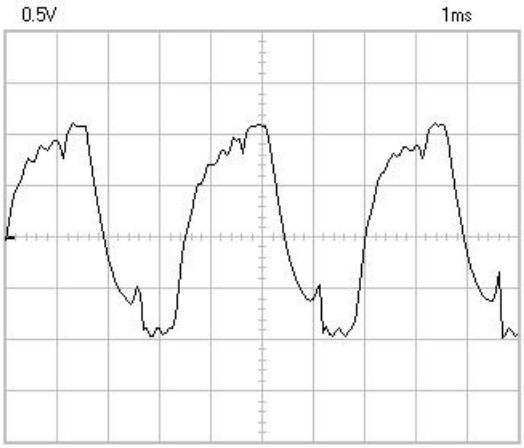


ecotan.JPG

Nella realizzazione pratica da me fatta, il segnale V_1 viene prelevato tramite un jack dalla presa auricolare di un radio ricevitore Yaesu del tipo scanner, ma potrebbe provenire da qualsiasi altra fonte purchè non dia più di 3 volt picco/picco. L'altra estremità del cavetto di connessione si inserisce sullo stesso connettore ICSP usato per la programmazione, che quindi acquista una funzione alternativa.
Il trasmettitore per questo rudimentale "radiocomando AFSK" è capace di inviare un segnale modulato con 3 diverse frequenze audio: 263, 313 o 372 Hz che rappresentano i 3 canali di comando On-Off. E' chiaro che le due frequenze estreme coincidono con le frequenze di risonanza dei due filtri A e B; se le ampiezze all'uscita dei rispettivi rivelatori sono circa eguali vuol dire che la frequenza inviata è quella intermedia. Non mi sono curato granchè dello warping tipico dei filtri digitali, eventualmente si può ritarare il tutto ma non ce ne è stato bisogno se non per altri motivi (componentistica del modulatore).
Riporto quì sotto lo schema del circuito che genera le tre frequenze suddette e funge da modulatore, e riporto anche un oscillogramma del segnale audio all'uscita dello scanner: la forma d'onda è distorta ma il funzionamento del radiocomando è affidabile grazie ai filtri.
Il fattore di merito dei risonatori, Q = 2 pigreco F R C, è stato fissato ad un valore di 15 che va bene per questa applicazione ma si può aumentare se serve per altri scopi.



ecotan1.JPG



elec.JPG

Estratto da "https://www.electroyou.it/mediawiki/index.php?title=UsersPages:Ecotan:realizzazione-pratica-di-un-semplice-filtro-digitale"