UNIVERSITÀ DEGLI STUDI DI PADOVA
FACOLTÀ DI INGEGNERIA

Tesi di Laurea / Relazione di Tirocinio

CANCELLAZIONE DELL'ECO ACUSTICA MEDIANTE FILTRAGGIO ADATTATIVO NLMS A FILTRI FIR E IMPLEMENTAZIONE SU DSP

Acoustic Echo Cancellation by Adaptive Filter FIR with NLMS

and Implementation about DSP

17 Aprile 2012

Relatore
Prof. Stefano Tomasin

Laureando
Patrik Celin
588572-IL

Dipartimento di Ingegneria
dell'Informazione

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Corso di Laurea Specialistica
in Ingegneria Elettronica

Anno Accademico 2011 – 2012

Ai miei genitori Luigi e Nadia

che mi hanno sempre sostenuto con affetto nelle gioie e nei dolori in questi anni

Al mio Maestro

che è sempre stato una guida per superare le difficoltà e ha sempre creduto in me

Ringrazio il Prof. Stefano Tomasin
per il suo prezioso aiuto nell'affiancarmi durante la stesura di questa tesi.

Ringrazio l'Ing. Riccardo Sala, Roberto Fabbian, e tutti i miei colleghi del Laboratorio di Ricerca e Sviluppo della CAME Domotic System s.r.l. e l'azienda CAME s.p.a.
per la loro grande disponibilità, supporto e cortesia durante il periodo di stage.

Ringrazio i miei amici più stretti e le persone più care
(che non nomino ma sono certo non sia doveroso), che mi vogliono bene e mi sono sempre rimasti vicino

A tutti Voi un Grazie di cuore. Patrik.

Indice

Introduzione

Il problema dell'eco acustica è un aspetto rilevante nella progettazione di un sistema di comunicazione in ambienti fortemente riverberanti o nel caso di sistemi realizzati con componenti ed elementi che per le loro caratteristiche intrinseche inneschino facilmente fenomeni di eco. La presenza di eco infatti, non solo può compromettere la stabilità dei sistemi operanti in anello chiuso (*full-duplex*), innescando fenomeni di feedback acustico come l'effetto Larsen che impedisce la comunicazione, ma genera anche disturbi che diminuiscono l'intellegibilità delle conversazioni; è pertanto auspicabile che tali sistemi prevedano un dispositivo adibito alla cancellazione dell'eco acustica (*Acoustic Echo Canceller*, *AEC*). Poiché il comportamento acustico del sistema è in generale di tipo tempo-variante, le tecniche numeriche utilizzate per realizzare un AEC si basano sul filtraggio adattativo, tipicamente, per motivi di semplicità ed efficienza, tra i principali algoritmi impiegati vi sono l'*LMS* (*Least-Mean-Square*), l'*NLMS* (*Normalized Least-Mean-Square*) e l'*RLS* (*Recursive Least Squares*).
A fronte di queste considerazioni, le prestazioni di un *AEC* sono determinate, a parità di massima attenuazione dell'eco ottenibile, dalla velocità di convergenza e dal costo computazionale. Essendo la voce umana un segnale colorato, per ottenere prestazioni soddisfacenti in termini di velocità di convergenza, l'algoritmo *LMS*, per esempio, deve essere impiegato all'interno di schemi di filtraggio sofisticati, o viceversa l'algoritmo *RLS* deve essere impiegato in schemi altamente performanti.
In questa tesi si analizzeranno le caratteristiche di riverberazione acustica degli ambienti chiusi, si tratteranno i fondamenti teorici della cancellazione dell'eco acustica, si presenterà una panoramica sulla realizzazione di un *AEC* basato su *LMS* e *NLMS* e facendo riferimento a simulazioni in linguaggio *MATLAB*, si confronteranno le prestazioni tra i diversi algoritmi.
Vedremo una possibile implementazione del progetto realizzata su scheda a *DSP*, in particolare faremo riferimento alla TLV320AIC3254EVM-U della Texas Instruments interamente programmabile con un linguaggio di sviluppo su ambiente grafico analogo al Labview, il PurePath Studio Graph Portable Audio.
Il filtro adattativo implementato è basato sull'algoritmo *NLMS* che corregge echi di lunghezza massima pari a 32 ms con una frequenza operativa di campionamento di 8 kHz.

Capitolo 1

Teoria del Suono

1.1 Percezione sonora

La percezione sonora è la capacità di distinguere e riconoscere differenti pressioni acustiche che giungono al timpano dell'orecchio e vengono riconosciute come vibrazioni di questo. Queste vibrazioni, sono l'equivalente fisico del suono e sono provocate da piccole variazioni di pressione dell'aria. In un altoparlante, il movimento verso l'esterno della membrana determina un aumento di pressione spingendo lontano le molecole d'aria, invece il movimento verso l'interno della membrana comporta una diminuzione di pressione delle molecole d'aria, attraendole. Analogamente la membrana del timpano presenta lo stesso comportamento della membrana dell'altoparlante: un incremento di pressione dell'aria la spinge verso l'interno, mentre una diminuzione la attrae verso l'esterno.

1.2 Il Suono

Il suono è la sensazione percepita dall'organo dell'udito sottoposto ad una sollecitazione acustica esterna. Le principali caratteristiche del suono sono: *l'intensità del suono*, che caratterizza i suoni in funzione dell'energia e della forza con le quali giungono al timpano, e l'*altezza del suono*, la quale ci permette di distinguere i suoni da più o meno acuti, a più o meno gravi.

1.3 Il Rumore Acustico

Il rumore acustico è un fenomeno sonoro per il quale valgono tutte le osservazioni viste per il suono, è quindi opportuno, per distinguerlo dal suono stesso, indicarne una caratterizzazione in maniera non fisica ma funzionale, semplicemente come *suono non voluto*.
Viene classificato in due principali tipologie :

- *rumore impulsivo*: determinato da rapide variazioni di pressione con un'estensione temporale che si aggira intorno pochi millisecondi.
- *rumore stazionario*: presenta un'ampia estensione temporale, è privo di regolarità e ne viene, tipicamente, descritto l'andamento di pressione da un punto di vista statistico.

1.4 Caratteristiche del Suono

La vibrazione di corpi elastici presenta un andamento ondulatorio, descritto dalle seguenti grandezze:

- la frequenza f, espressa in $Hertz$ $[Hz]$, definisce l'altezza del suono: in particolare suoni con frequenze elevate sono più acuti, suoni con frequenze basse sono più gravi.
- l'ampiezza dell'oscillazione, espressa in $Watt$ $[W]$, definisce l'intensità del suono, ed è proporzionale all'energia dell'onda sonora, tipicamente tale grandezza viene espressa in $Decibel$ $[dB]$ in quanto la percezione uditiva dell'orecchio non segue un andamento lineare bensì logaritmico.
- Il timbro sonoro, ci fornisce informazioni sul tipo di sorgente sonora che stiamo ascoltando, è legato a variazioni regolari di pressione cioè al carattere di periodicità di un'onda sonora.

1.5 Suoni Periodici

Un suono periodico, di periodo T, caratterizzato dall'andamento di pressione acustica in un mezzo trasmissivo $p(t)$, è scomponibile, per il teorema della scomposizione in serie di Fourier, nella somma di n sinusoidi pesate dai termini a_n:

$$p(t) = a_0 + \sum_{n=1}^{\infty} a_n \, sin(2\pi ntT + \varphi_n)$$

dove φ_n è la fase iniziale di ciascuna sinusoide; si possono trovare unità di misura assolute come il watt oppure relative espresse come watt per unità di superficie. Tipicamente, se si considera, come mezzo di propagazione l'aria, questa presenta già un termine costante dato dalla pressione atmosferica, pertanto il termine a_0 viene usualmente trascurato. In *Figura 1.1* vengono riportati, come esempio, gli andamenti delle forme d'onda rispettivamente associate a una sinusoide, ad un segnale periodico costituito da una somma di 16 sinusoidi in rapporto armonico tra loro e ad un segnale rumoroso stazionario.

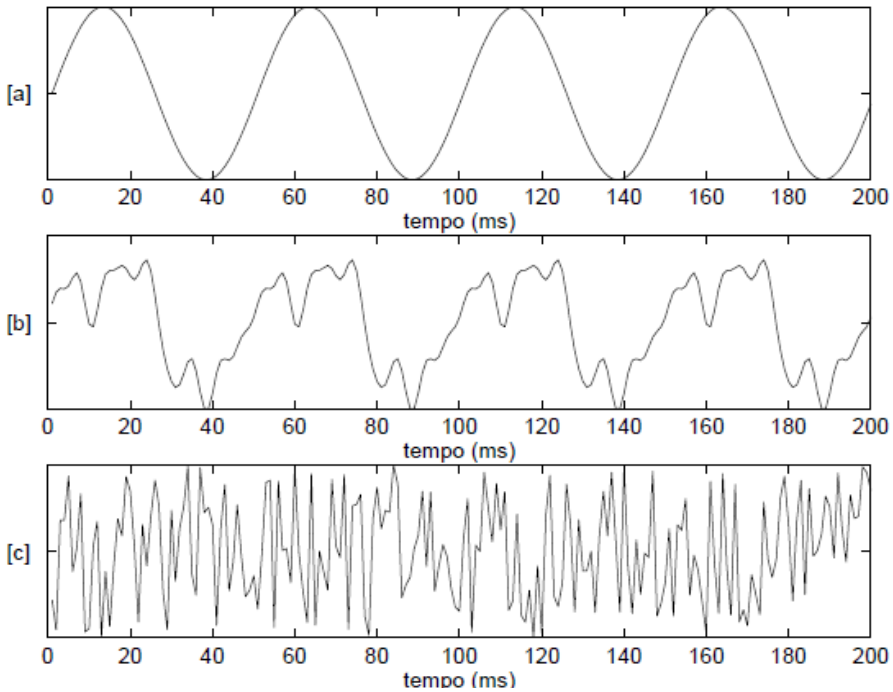


Figura 1.1: Andamento nel tempo di tre segnali rispettivamente con andamento: [a] sinusoidale, [b]periodico (somma di 16 sinusoidi), [c] aperiodico

La frequenza è direttamente proporzionale alla sensazione di altezza (*pitch*) di un suono: pertanto un suono risulterà acuto se la sua frequenza è elevata e grave se la sua frequenza è bassa. L'organo uditivo umano è in grado di percepire suoni nell'intervallo di frequenze che va da circa 20 Hz a circa 20 kHz: per frequenza inferiori a tale intervallo, le variazioni di pressione vengono percepite come una successione di impulsi; le frequenze superiori (ultrasuoni), invece, non vengono percepite.
Quindi il limite in frequenza della soglia di udibilità umana è lecito fissarlo a 20 kHz.

1.6 Suoni Reali

I suoni naturali non si presentano mai come sinusoidi pure, bensì sono costituiti da un insieme di armoniche, che contribuiscono a dare profondità e ricchezza alla qualità sonora e il cui rapporto non è esattamente un numero intero.
Inoltre, in natura, i suoni reali non sono mai esattamente periodici, bensì le forme d'onda, nonostante si ripetano nel tempo, assumono andamenti simili ma non uguali.
L'orecchio percepisce quindi un andamento approssimativamente periodico con delle variazioni che contribuiscono a dare dinamicità al suono ascoltato; d'altra parte un'eccessiva regolarità nello sviluppo temporale di un suono (come per esempio i suoni di sintesi), si traduce in una progressiva perdita di interesse per l'ascoltatore nei confronti del suono udito.

1.7 Inviluppo del Suono

Un suono matematicamente periodico, non consentirebbe alcuna forma di comunicazione, la quale invece è composta da un'articolata organizzazione di suoni.
La struttura intrinseca di un generico suono percepito da un ascoltatore, può essere schematizzata da un inizio, un'evoluzione temporale e una fine.
Si definisce fase di attacco (*attack*), la fase iniziale durante la quale l'ampiezza cresce progressivamente, generalmente si estende al più per qualche decina di millisecondi, in relazione al tipo di suono considerato. La fase successiva è definita decay, si nota un rapido assestarsi dell'ampiezza ad un valore pressoché costante. Spesso fra la fase di attack e la fase di decay può verificarsi una sovraelongazione dell'ampiezza dell'onda. Con la fase decay anch'essa piuttosto rapida, si esaurisce il transitorio iniziale di un suono. La fase che segue si chiama sustain, può durare anche diversi secondi, ed è caratterizzata dal fatto che l'ampiezza del suono non varia significativamente. L'ultima fase, viene denominata release e corrisponde al tempo che intercorre fra il termine dell'emissione sonora da parte della sorgente audio considerata e lo stato di quiete. In *Figura 1.2* sono illustrate le quattro fasi descritte per un generico suono; sarebbe stato più preciso rappresentarne l'andamento dell'inviluppo di ampiezza in forma esponenziale anziché con delle spezzate, tuttavia l'errore che si commette con queste approssimazioni non è significativo.

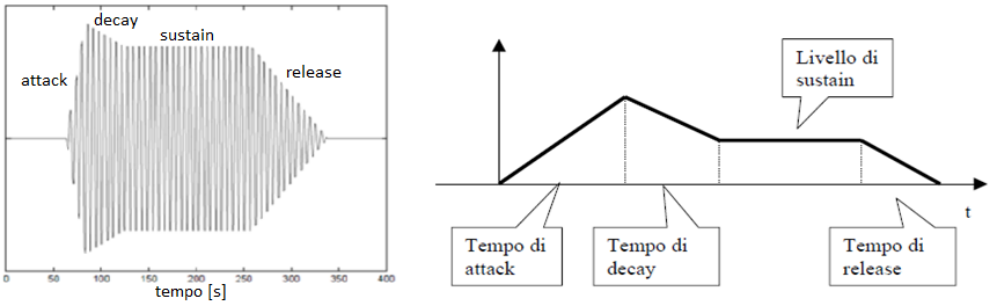


Figura 1.2: Fasi temporali di un segnale audio: attack, decay, sustain e release

1.8 Propagazione del Suono

Analizziamo la propagazione del suono in un mezzo di natura elastica come l'aria: questa è costituita da molecole unite fra di loro da legami inter-molecolari. La vibrazione di un corpo, trasferisce energia cinetica alle molecole d'aria che sono a contatto direttamente con la superficie stessa del corpo; queste molecole iniziando a vibrare, trasferiscono a loro volta il movimento alle molecole adiacenti, per mezzo dei legami inter-molecolari, le quali faranno lo stesso e così via. Un istante successivo i legami inter-molecolari modellabili come interazioni elastiche, "richiamano" indietro le molecole nelle loro posizioni iniziali di equilibrio, dove però, per effetto d'inerzia, le molecole non torneranno esattamente al punto di equilibrio, bensì lo supereranno, raggiungendo posizioni quasi speculari rispetto al punto di massima escursione raggiunto precedentemente.
Queste oscillazioni, corrispondenti a variazioni di pressione acustica, identificano nella direzione di propagazione del suono, delle zone in cui vi sarà compressione dell'aria e delle altre in cui vi sarà rarefazione.

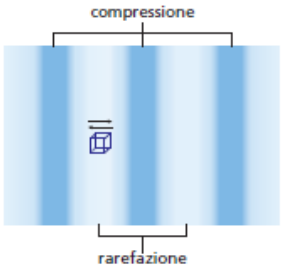


Figura 1.3: Zone di compressione e rarefazione molecolari dell'aria

Se le direzioni di oscillazione e di propagazione coincidono si parla di *onda longitudinale*, se invece sono perpendicolari si parla di *onda trasversale*.
Misurando la pressione dell'aria, di un suono periodico, lungo la direzione di propagazione, si può notare una successione di aumenti e diminuzioni di pressione denominata onda sonora. La più piccola distanza tra due punti corrispondenti dell'onda sonora (ad esempio tra due massimi consecutivi) è detta *lunghezza d'onda* λ ed è legata al periodo e alla velocità di propagazione nel mezzo:

$$\lambda = c\,T = \frac{c}{f} \qquad [m]$$

La velocità del suono in aria viene solitamente indicata con la lettera c e, a temperatura ambiente ($25\ °C$), è di circa $344\ m/s$, ovvero $1238\ km/h$, è indipendente dalla frequenza del suono considerato, bensì è legata alla pressione atmosferica e alla temperatura: indicativamente aumenta di circa $0{,}6\ m/s$ per grado centigrado.
Nello spettro delle frequenze udibili da 20 Hz a 16 kHz, le lunghezze d'onda dei suoni udibili vanno da circa $17\ m$ (suono grave) a $21\ cm$ (suono acuto).
Sperimentalmente si è osservato che il suono si propaga in *aria secca* (alla pressione atmosferica normale di $1{,}01 \cdot 10^5$ Pa e alla temperatura di 0 °C) con la velocità di 331,4 m/s cioè 1193 km/h.

In *Tabella 1.1* vengono riportate le velocità di propagazione per differenti mezzi trasmissivi:

VELOCITÀ DEL SUONO IN DIVERSI MEZZI MATERIALI		
Mezzo	**Temperatura (°C)**	**Velocità (m/s)**
Aria	0	331,4
Acqua	15	1450
Piombo	20	1230
Ferro	20	5130
Granito	20	4000
Gomma vulcanizzata	0	54

Tabella 1.1: Velocità di propagazione del suono a 0 °C in alcuni mezzi trasmissivi [4]

1.9 Intensità del Suono

Dal punto di vista fisico, il suono è una variazione di pressione dell'aria (che viene misurata in Pascal, $[Pa]$), ed è direttamente proporzionale alla percezione di volume sonoro dell'ascoltatore (*loudness*). Invece di osservare la variazione istantanea della pressione è più significativo fare riferimento alla pressione efficace p_{eff}, cioè alla media quadratica delle variazioni di pressione in un intervallo di tempo:

$$p_{eff} = \frac{1}{(t_2 - t_1)} \sqrt{\int_{t_1}^{t_2} p(t)^2 dt} \qquad [Pa]$$

dove l'integrazione è opportuno considerarla su un periodo per suoni periodici e su un intervallo idealmente infinito per suoni non periodici.
Nel caso specifico di un andamento sinusoidale della pressione:

$$p(t) = P_0 \sin\frac{2\pi t}{T} \qquad \text{si ha} \qquad p_{eff} = \frac{P_0}{\sqrt{2}}$$

La minima pressione efficace che può essere percepita dall'organo uditivo umano è di $2{\cdot}10^{-5}\ Pa$, mentre la soglia del dolore critica si assesta intorno ai $20\ Pa$.
La pressione efficace di un suono dipende dalla distanza tra l'ascoltatore e la sorgente che lo ha generato e varia in funzione delle diffrazioni e riflessioni che si vengono a creare nel mezzo di propagazione.
Questi effetti complicano la misurazione di questa grandezza, rendendola sensibile sia alla distanza, che alla posizione rispetto alla sorgente, ma anche alla presenza di ostacoli ed elementi che alterano la propagazione dell'onda sonora.
Per ovviare a queste difficoltà, le sorgente sonore vengono caratterizzate in base alla propria potenza acustica, misurata in $Watt$ $[W]$.
In *Tabella 1.2* viene riportata la potenza acustica di diverse sorgenti sonore.

Sorgente sonora	**Potenza (W)**
Parlato (normale)	10^{-5}
Parlato (litigio)	10^{-3}
Cantante lirico	0.03
Clarinetto	0.05
Tromba	0.3
Pianoforte	0.4
Trombone	6
Orchestra	60

Tabella 1.2: Potenza massima prodotta da alcune sorgenti sonore. [2]

Si definisce *intensità acustica* I, la potenza media trasmessa per unità di superficie nella direzione di propagazione dell'onda. Si può dimostrare che, per onde piane e onde sferiche, vale la relazione:

$$I = \frac{{p_{eff}}^2}{\rho\, c} \qquad [\frac{W}{m^2}]$$

dove ρ è la densità del mezzo trasmissivo (in aria, a temperatura ambiente e a pressione atmosferica standard ρ = 1,21 kg/m^3), p_{eff} è la pressione efficace e c è la velocità del suono nel mezzo.
Considerando l'intervallo di valori assunti dalla pressione efficace, si nota che l'intensità acustica assume valori in un range molto elevato, andando da circa 10^{-12} $[\frac{W}{m^2}]$ per la soglia di udibilità a circa 1 $[\frac{W}{m^2}]$ per la soglia del dolore.
La potenza trasmessa dalla sorgente audio al mezzo elastico, è detta *potenza sonora* W che nel caso specifico di onde piane in mezzi in quiete non viscosi vale:

$$W = I \cdot S \qquad [W]$$

con S la superficie attraverso cui si trasmette la potenza sonora.

1.10 Misura del Suono

La misura più usata in acustica è il *decibel* $[dB]$, esso esprime il rapporto fra due potenze su scala logaritmica con base 10.
Tipicamente si ricorre all'utilizzo della scala logaritmica per esprimere grandezze i cui valori si distribuiscono su intervalli molto estesi, ma la motivazione per la quale viene usata in acustica, consiste nel fatto che presenta un andamento più prossimo a quello delle scale percettive.
Si definisce *livello di pressione* (*Pressure Level,* PL) il logaritmo del rapporto tra la pressione misurata p e una pressione di riferimento p_{ref}:

$$PL = 20\, log_{10} \frac{p}{p_{ref}} \qquad [dB]$$

dove si fa implicitamente riferimento alla pressione efficace.
Talvolta, può risultare conveniente esprimere questo rapporto utilizzando una pressione efficace di riferimento minima udibile p_{ref} pari a $2{\cdot}10^{-5}Pa$, si parla quindi di *livello di pressione sonora* (*Sound Pressure Level,* SPL) definito come:

$$SPL = 20\, log_{10} \frac{p}{p_{ref}} = 20\, log_{10} \frac{p}{2{\cdot}10^{-5}} \quad \Rightarrow \quad SPL = 20\, log_{10}\, p + 94 \qquad [dB]$$

Analogamente, anche per la potenza e per l'intensità sonora si parla rispettivamente di:
livello di potenza sonora (Soundpower Level, L_W) :

$$L_W = 10\, log_{10} \frac{P}{P_{ref}} \qquad [dB]$$

dove con P si è indicata la potenza acustica misurata in *Watt [W]* e con P_{ref} *la* potenza di riferimento di $1 \cdot 10^{-12} [W]$*;*
livello di intensità acustica (Intensity Level, IL) è definito dalla formula:

$$IL = 10\ log_{10} \frac{I}{I_{ref}} \qquad [dB]$$

dove l'intensità di riferimento I_{ref} è assunta pari a $1 \cdot 10^{-12}\ [W/m^2]$.
In *Tabella 1.3* vengono riportati, indicativamente, utilizzando come riferimento la minima intensità udibile, i valori in $[dB]$ prodotti da alcuni esempi di sorgenti sonore; da considerare però, che nonostante la soglia di dolore si aggiri intorno ai $140\ [dB]$, un'esposizione prolungata, a intensità sonore elevate seppur inferiori alla soglia del dolore, possono causare danni permanenti all'organo uditivo.

Livello di intensità dB	Condizione ambientale	Effetto sull'uomo
140	Soglia del dolore	Lesioni dell'orecchio nel caso di ascolto prolungato
120	Clacson potente, a un metro	
110	Picchi d'intensità di una grande orchestra	Zona pericolosa per l'orecchio
100	Interno della metropolitana	
90	Picchi di intensità di un pianoforte	
80	Via a circolazione media	Zona di fatica
75	Voce forte, a un metro	
70	Conversazione normale, a un metro	
60	Ufficio commerciale	
50	Salotto calmo	Zona di riposo (giorno)
40	Biblioteca	
30	Camera da letto molto calma (notte)	Zona di riposo (notte)
20	Studio di radiodiffusione	
0	Soglia di udibilità	

Tabella 1.3: Livello di intensità associato ad alcune sorgenti sonore. [2]

I livelli di intensità e di pressione acustica variano con la distanza tra sorgente e ascoltatore e dipendono dal numero di sorgenti sonore che si considerano.
Nel caso specifico di due differenti sorgenti sonore, di pari intensità, tra loro non correlate, si può dimostrare che, l'intensità acustica totale ottenuta come somma delle singole intensità acustiche sarà incrementata al massimo di una quantità pari a $3\ [dB]$.
In *Figura 1.4* viene illustrato, nel caso di due sorgenti audio non correlate, l'andamento dell'incremento del livello di intensità sonora, in funzione dell'intensità di una delle due sorgenti mentre l'altra viene mantenuta costante a $60\ [dB]$.
È evidente che nel caso di sorgenti con intensità molto differenti, risulta significativo il contributo della sorgente a maggiore intensità.
In generale considerati due suoni non coerenti, con livelli di intensità IL_1 e IL_2, il livello di intensità totale si calcola come somma delle rispettive potenze acustiche:

$$IL_{tot} = 10\ log_{10}\left(\frac{P_1 + P_2}{P_{ref}}\right) = 10\ log_{10}\left(10^{IL_1/10} + 10^{IL_2/10}\right) \qquad [dB]$$

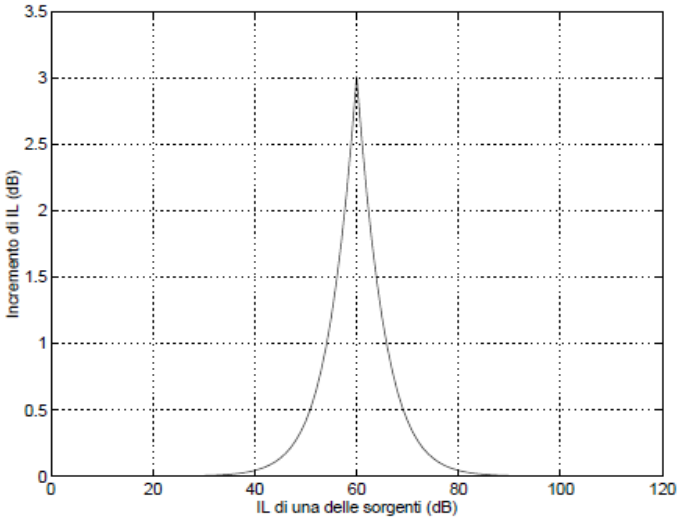


Figura 1.4: Incremento del livello di intensità, rispetto la sorgente di intensità maggiore, nel caso di due sorgenti, la prima fissa a 60 dB e la seconda variabile da 0 dB a 120 dB. [2]

In situazioni non ideali, la propagazione del suono in un mezzo ha andamenti piuttosto complessi e articolati che ne complicano il calcolo delle grandezze precedentemente esposte. In particolare, la propagazione del suono in un mezzo non omogeneo è interessata da alcuni fenomeni analoghi a quelli riscontrati nella propagazione di un'onda elettromagnetica, come per esempio la luce: tra questi i principali sono la diffrazione e la riflessione.

1.11 La Diffrazione

La diffrazione è un fenomeno che consiste nella deviazione della traiettoria di propagazione di un'onda, quale per esempio il suono o la luce, quando questa incontra un ostacolo sul proprio cammino.
Gli effetti della diffrazione sono considerevoli quando la lunghezza d'onda λ dell'onda è comparabile con le dimensioni dell'ostacolo, in particolare per una sorgente sonora reale, λ è legata all'efficienza di irradiamento acustica: se la dimensione di una sorgente audio (per esempio il raggio di un altoparlante) è ridotta rispetto λ, l'effetto della diffrazione può essere trascurato e si può considerare la sorgente come se fosse ideale quindi puntiforme e irradiante in tutte le direzioni con la stessa efficienza; nel caso λ sia confrontabile con le dimensioni della sorgente, il suono verrebbe irradiato con efficienza diversa a seconda della direzione; infine se λ è minore rispetto la dimensione della sorgente vi è un angolo al di sopra del quale non c'è praticamente irradiamento.
In *Figura 1.5* viene illustrata la diversa diffrazione di un'onda che incontra come ostacolo, nel cammino di propagazione, un'apertura D di dimensione, rispettivamente, minore o maggiore della lunghezza d'onda λ.

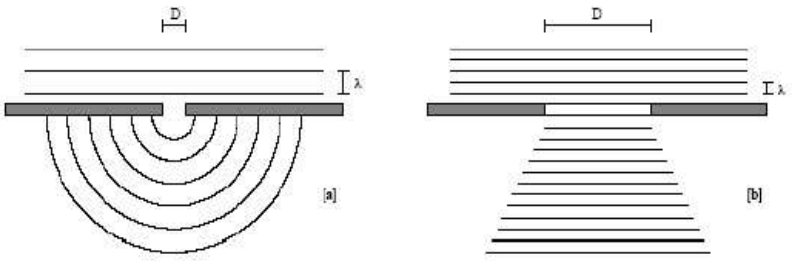


Figura 1.5: Effetto della diffrazione nei casi (a) $\lambda > D$, (b) $\lambda < D$

Uno degli effetti della diffrazione consiste nel favorire la direzionalità dei suoni più acuti rispetto a quelli più gravi, tuttavia, i suoni con frequenze basse vengono più facilmente percepiti anche in presenza di ostacoli che non ne consentono la propagazione diretta. Infatti, quando un'onda incontra un ostacolo di dimensioni piccole rispetto alla lunghezza d'onda, il suono viene diffratto e riesce a superare l'ostacolo, viceversa il suono non riesce a superare l'ostacolo e si crea una zona d'ombra.

1.12 Riflessione ed Eco

La riflessione è un fenomeno che si verifica quando nel cammino di propagazione di un'onda cambiano le caratteristiche del mezzo trasmissivo; la causa più comune è la presenza di una discontinuità dell'indice di rifrazione, dovuta al fatto che l'onda propagandosi incontra un ostacolo.
Ponendosi nel caso, più semplice, in cui l'ostacolo è per esempio una parete liscia di una stanza, di dimensioni sufficientemente grandi rispetto alla lunghezza d'onda λ dell'onda, si avrà che l'onda verrà parzialmente assorbita e parzialmente riflessa, con un angolo di riflessione pari all'angolo di incidenza; invece nel caso in cui l'onda si rifletta su di una superficie che presenta delle irregolarità di dimensioni paragonabili a λ, si avrà un tipo di riflessione detta *eco diffuso*, dove la direzione di propagazione varia in funzione di λ e della geometria fisica dell'ostacolo.
L'effetto della riflessione è estremamente importante nella progettazione di ambienti in cui il comportamento acustico sia determinante, infatti a causa della riflessione alle pareti, all'ascoltatore giunge sia il suono diretto proveniente dalla sorgente sonora, sia una successione di onde riflesse che, a causa della maggiore distanza percorsa, giungono all'ascoltatore in ritardo rispetto al suono diretto.
Questo fenomeno è noto come *riverberazione*: in prima analisi, considerando una singola riflessione totale, senza perdite, e in condizioni di idealità, ai fini del calcolo del tempo di riverberazione o della lunghezza d'eco di un'onda sonora, che incontra un singolo ostacolo, è sufficiente conoscere la distanza d fra la sorgente e l'ostacolo. L'onda sonora percorrerà due volte la distanza d fra la sorgente e l'ostacolo (percorso di andata e di ritorno *Figura 1.6*), impiegando un tempo Δt:

$$\Delta t = \frac{2d}{v} \quad [s]$$

dove v è la velocità del suono in aria secca a 25 °C pari a 344 [m/s].

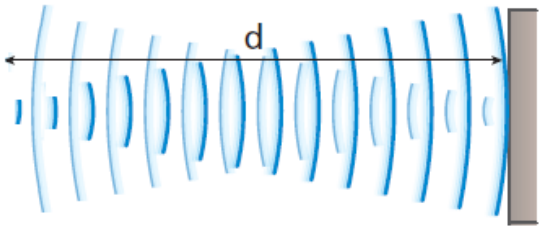


Figura 1.6: Distanza percorsa dall'onda sonora

Capitolo 2

Elementi di acustica ambientale

L'ambiente è l'elemento principale che modifica le caratteristiche di propagazione dell'onda sonora emessa da una sorgente audio.
In un ambiente esterno, le onde sonore (che non colpiscono direttamente la zona di ascolto), vengono disperse, senza nessun effetto indesiderato, mentre in un ambiente chiuso la situazione è ben diversa (*Figura 1.7*): in quanto un'onda sonora che raggiunge una parete o un ostacolo può venire riflessa totalmente, oppure venire parzialmente riflessa e/o parzialmente trasmessa e/o parzialmente assorbita (*Figura 1.8*).

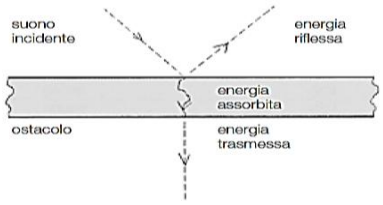


Figura 1.7: Propagazione energia sonora

2.1 Effetti della Riflessione, dell'Assorbimento e della Diffrazione

Gli effetti provocati dalle riflessioni sono principalmente:

- *Attenuazione di livello,* dove una porzione di energia emessa in direzione dell'ascoltatore viene deviata verso altre direzioni;
- *Interferenza,* dove porzione di energia emesse in direzioni divergenti da quella utile vengono ridirette verso la direzione utile, andandosi a sovrapporre, con un certo ritardo, alla porzione di emissione "utile".

Quando l'emissione sonora si infrange su un ostacolo assorbente, solo una porzione di essa è in grado di attraversarlo, con la conseguente attenuazione di livello dell'emissione.
Generalmente quando un'emissione sonora incontra un ostacolo, tende ad aggirarlo, se però sono presenti delle aperture, una porzione di energia dell'emissione (corrispondente alle componenti sonore di lunghezza d'onda del medesimo ordine di grandezza delle dimensioni delle aperture), attraverserà l'apertura venendo inoltre deviata rispetto la direzione di propagazione originale. La diffrazione può in tal modo determinare fenomeni di interferenza, con conseguenti attenuazioni o amplificazioni di livello.

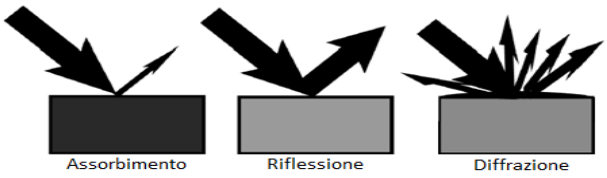


Figura 1.8: Tipologie di propagazione

In generale, quando un'onda elettromagnetica incontra un ostacolo, una parte dell'energia acustica viene trasmessa attraverso il materiale, una parte ne viene assorbita e la rimanente viene riflessa nell'ambiente: la composizione dei materiali e le proprietà geometriche degli ostacoli determineranno le rispettive quantità di energia alle diverse frequenze.
L'energia acustica non assorbita o trasmessa viene riflessa nell'ambiente e ad ogni differente posizione dell'ascoltatore corrisponde una precisa quantità di suono indiretto: costituito dalla somma delle riflessioni provenienti dalle superfici circostanti, che giungono alla posizione di ascolto con ritardi temporali diversi e tendendo a persistere (*riverberazione*) anche dopo che l'energia acustica diretta si sia esaurita.
Lo studio delle caratteristiche acustiche di un ambiente può risultare molto articolato e complesso: si valuta la geometria dell'ambiente stesso, i materiali che lo rivestono, gli oggetti presenti, la tipologia di sorgenti sonore, la posizione di queste, la potenza sonora sviluppata e quella irradiata, si ricorre pertanto all'utilizzo di software dedicati in grado di simulare il comportamento dell'onda sonora, ai fini di ottimizzare l'acustica nell'ambiente simulato.
Le riflessioni dell'onda acustica (*Figura 1.9*), dovute all'ambiente, ne elevano la sonorità, in quanto vanno a rafforzare il segnale originario talvolta degenerando in un fastidioso feedback che compromette l'informazione sonora trasmessa.

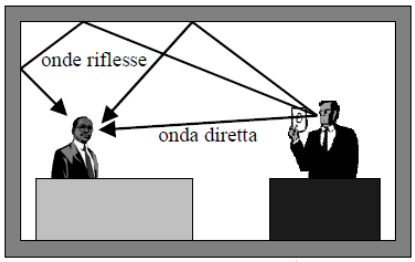


Figura 1.9: Acustica Ambientale

Con una serie di accorgimenti si cerca di eliminare o di limitare questo rientro acustico: queste soluzioni spaziano dall'impiego di equalizzatori parametrici, alla distribuzione di diffusori a potenze inferiori, all'utilizzo di pannelli fonoassorbenti oppure alla progettazione di filtri digitali dedicati.
Se un suono riflesso, come per esempio il parlato in una stanza, raggiunge l'ascoltatore con un tempo di ritardo inferiore ai $30\ [ms]$, il cervello lo percepisce come un rinforzo di timbro, invece se il ritardo si allunga dai 30 ai $50\ [ms]$, si percepisce il suono "opaco" (a causa degli sfasamenti), superato tale limite si ottiene un vero e proprio eco che rende la comunicazione inintelligibile (nel caso di musica i tempi di allungano e si raggiungono anche gli $80\ [ms]$ di ritardo).

2.2 Feedback Acustico: l'Effetto Larsen

Quando un'onda sonora diretta (proveniente da un diffusore acustico) o indiretta (riverberata), viene percepita dal microfono e nuovamente amplificata e riprodotta dal diffusore, si crea un anello di feedback acustico nominato *effetto Larsen* (dal nome del fisico Soren Absalon Larsen che per primo ne scoprì il principio), in altre parole, quando un'onda sonora si riflette in un ambiente acusticamente riflessivo, o quando il microfono è troppo vicino ai diffusori, questo capta una frequenza che in un preciso istante è più forte delle altre, la quale viene amplificata e riprodotta a sua volta con ampiezza via via crescente, virtualmente illimitata, se non fosse che l'amplificatore entra in saturazione. L'effetto Larsen appare all'ascoltatore come un fastidiosissimo ronzio o fischio riprodotto dai diffusori

presenti nell'anello di retroazione (*Figura 1.10*); l'effetto Larsen si instaura se il suono emesso dai diffusori raggiunge il microfono a un'intensità pressoché simile a quella del suono originale, percepito dal microfono all'inizio della "catena".

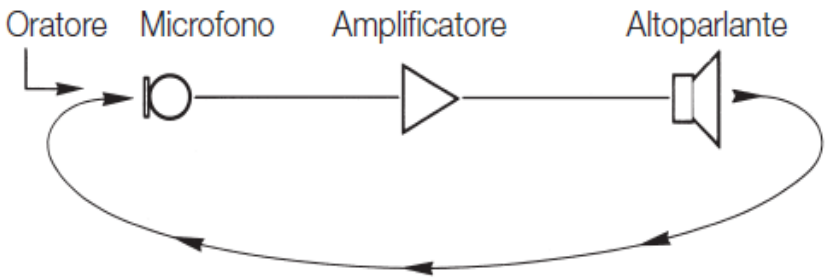


Figura 1.10: Feedback acustico effetto Larsen

Si cerca di ovviare a tale inconveniente in svariati modi: riposizionando i diffusori perché non diffondano in direzione del microfono; intervenendo mediante equalizzazione sui controlli di sensibilità, volume e tonalità dell'amplificatore; utilizzando un numero maggiore di diffusori acustici così da poter lavorare a volumi acustici inferiori; ricorrendo a componenti altamente direttivi, quali microfoni unidirezionali (cardioide); oppure mediante la progettazione di filtri digitali che intervengono per eliminare il feedback, senza aggravare però le prestazioni del sistema, in termini di pressione acustica sviluppata.

Vediamo come calcolare il guadagno G_r richiesto dal sistema di amplificazione in $[dB]$, affinché un suono possa venire udito, almeno a una certa intensità minima, da tutti gli ascoltatori presenti. Consiste nel considerare ΔL_o la distanza tra sorgente sonora e ascoltatore più lontano in [m], in funzione di questa trovare il guadagno corrispondente in [dB] (*Figura 1.11*) al quale si sottraggono 12 [dB] che è un guadagno di riferimento corrispondente a una distanza convenzionale di 1,2 [m] tra sorgente sonora e ascoltatore alla quale non è necessario l'uso di un rinforzo acustico per la comunicazione.

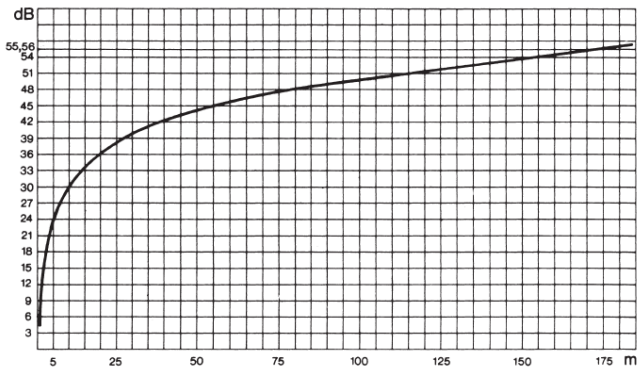


Figura 1.11: Guadagno in [dB] in funzione della distanza dal microfono [2]

Vediamo come si calcola il guadagno massimo G_{m0} di un sistema di amplificazione:

$$G_{m0} = [\Delta L_o + \Delta L_1 - \Delta L_2 - \Delta L_m]_{dB} \qquad [dB]$$

dove tutte le distanze lineari presenti nella formula sono convertite in $[dB]$ con l'ausilio del grafico della *Figura 1.11*, ΔL_o è la distanza tra sorgente sonora e ascoltatore più lontano, ΔL_1 è la distanza tra microfono e diffusore più vicino, ΔL_2 è la distanza tra ascoltatore più

lontano e diffusore a lui più vicino, e ΔL_mè la distanza tra sorgente sonora e microfono (*Figura 1.12*).

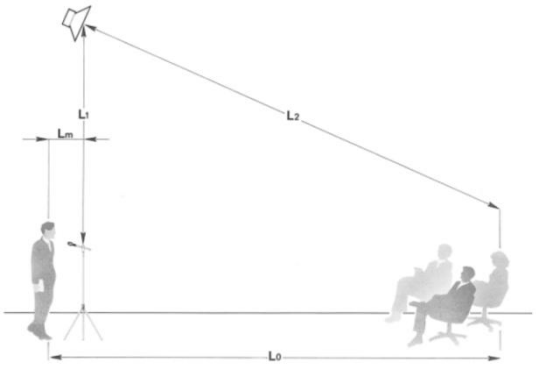


Figura 1.12: Schema per lo studio dell'effetto Larsen

Se il guadagno richiesto dal sistema supera il guadagno massimo ottenibile, si innesca l'effetto Larsen e occorre apportare delle variazioni alla struttura del sistema stesso o adottare uno degli accorgimenti sopra citati.
Tipicamente in sistemi di amplificazione non equalizzati correttamente, si riesce a raggiungere, nel migliore dei casi, prima che si inneschi l'effetto Larsen, metà del guadagno massimo ottenibile.

I parametri progettuali comunemente utilizzati in acustica ambientale sono:

- i tempi di riverberazione (*Reverberation Time, Early Decay Time*)
- i criteri energetici (*Clarity, Definition e* $ST1$)

a questi si aggiungono, per le applicazioni che lo richiedono:

- i parametri di intelligibilità del parlato (*Speech Transmission Index* STI*, RApid Speech Transmission Index* $RASTI$).

2.3 La Riverberazione

Il fenomeno acustico della riverberazione descrive la persistenza del suono dopo che la sorgente sonora ha terminato di irradiare energia ed è causato dalla riflessione continua delle onde sonore sulle superfici dell'ambiente di propagazione. Si consideri che, in ogni riflessione, l'intensità del suono venga ridotta di un fattore $x < 1$, dopo n riflessioni l'intensità risulterà diminuita di un fattore x^n; sia ΔT l'intervallo di tempo medio tra due riflessioni consecutive, allora l'intensità all'istante t sarà [2]:

$$I = I_0 x^{t/\Delta T} = I_0 e^{-t/\tau} \qquad [\frac{W}{m^2}]$$

che presenta un andamento esponenziale decrescente, con $\tau = -\Delta T/\log(x)$ costante di tempo caratteristica legata alla geometria dell'ambiente e ai materiali con cui sono rivestite le superfici.
Il suono riflesso, rispetto a quello diretto, giungerà quindi con un ritardo che dipende dalla distanza tra le pareti e dalla posizione dell'ascoltatore; mentre la durata totale del riverbero ed il suo comportamento in frequenza dipenderanno dalle dimensioni dell'ambiente e dalla proprietà di assorbimento sonoro delle superfici.

2.3.1 Assorbimento del Suono

L'assorbimento del suono è il processo mediante il quale l'energia di un'onda sonora, che si propaga in un mezzo, si trasferisce sotto forma di energia interna, con aumento della temperatura, e di energia meccanica, con aumento della vibrazione, alle particelle del materiale stesso su cui il suono incide.
Dopo una singola riflessione dell'onda sonora su di un materiale, si definisce *coefficiente di assorbimento* α del materiale, il rapporto adimensionale tra l'energia che perde l'onda incidendosi e l'energia totale prima della riflessione:

$$\alpha = \frac{Energia\ Persa}{Energia\ Incidente}$$

In acustica ambientale, comunemente si fa riferimento all'assorbimento di una superficie, che viene determinato come prodotto della superficie stessa per il suo coefficiente di assorbimento:

$$A = S \cdot \alpha \qquad [Sabin]$$

Il [Sabin] dimensionalmente è corretto considerarlo una misura di superficie.
Si introduce poi il concetto di assorbimento totale A, di un ambiente chiuso con pareti di diversi materiali:

$$A = S_1\alpha_1 + S_2\alpha_2 + \cdots + S_n\alpha_n = \sum_{i=1}^{n} S_i\alpha_i$$

dove S_i è l'area di ciascuna superficie e α_i il coefficiente di assorbimento del rispettivo materiale che la riveste (*Figura 1.13*).

Coefficienti di Assorbimento						
Tipo di superficie	**125Hz**	**250Hz**	**500Hz**	**1kHz**	**2kHz**	**4kHz**
Mattoni a vista	0.03	0.03	0.03	0.04	0.05	0.07
Intonaco dipinto	0.01	0.02	0.02	0.03	0.04	0.05
Cemento	0.01	0.02	0.04	0.06	0.08	0.10
Moquette su cemento	0.02	0.06	0.14	0.37	0.60	0.65
Marmo	0.01	0.01	0.01	0.01	0.02	0.02
Linoleum	0.02	0.03	0.03	0.03	0.03	0.02
Parquet	0.04	0.04	0.07	0.06	0.06	0.07
Porte in legno	0.1	0.07	0.05	0.04	0.04	0.04
Vetro	0.35	0.25	0.18	0.12	0.07	0.04
Tendaggi	0.05	0.07	0.13	0.22	0.32	0.35
Griglie di ventilazione	0.3	0.4	0.5	0.5	0.5	0.4
Superficie dell'acqua	0.01	0.01	0.01	0.02	0.02	0.03
Sedie in legno vuote	0.04	0.05	0.06	0.1	0.1	0.08
Sedie in legno occupate	0.3	0.41	0.49	0.84	0.87	0.84
Panche in legno vuote	0.1	0.09	0.08	0.08	0.08	0.08
Panche in legno occupate	0.5	0.56	0.66	0.76	0.8	0.76
Poltroncine vuote	0.49	0.66	0.8	0.88	0.82	0.7
Poltroncine occupate	0.6	0.74	0.88	0.96	0.93	0.85
Area con pubblico	0.25	0.35	0.42	0.46	0.5	0.5

Figura 1.13: Tabella dei coefficienti di assorbimento [2]

Infine ā, il coefficiente di assorbimento medio di un ambiente chiuso, è espresso da:

$$\bar{a} = \frac{A}{S}$$

con S la superficie totale dell'ambiente.

2.3.2 Tempo di Riverberazione, Reverberation Time ($RT60$)

Si consideri una sorgente audio in un ambiente chiuso, e che a un certo istante, interrompa di emettere segnale, il tempo impiegato dalla densità di energia dell'onda sonora per decrescere di $60\ [dB]$, rispetto il valore originario, è definito *tempo di riverberazione* ($RT60$ - *Reverberation Time*).
La relazione, seppur approssimata, che lega l'$RT60$ alle caratteristiche fisiche e geometriche di un ambiente, è la legge di Sabine (Wallace C. Sabine, 1898):

$$RT60 = 0{,}161\ \frac{V}{A} \qquad [s]$$

dove V è il volume dell'ambiente in $[m^3]$, A è l'assorbimento totale del suono in $[Sabin]$ e $0{,}161$ è una costante sperimentale in $[s/m]$.
Esistono, tuttavia, leggi più precise come per esempio la formula di Eyring impiegata per materiali con coefficiente di assorbimento maggiore di $0{,}2$:

$$RT60 = 0{,}161\ \frac{V}{-S\ln(1-A)} \quad [s]$$

Nel caso ideale di una distribuzione sonora uniforme e stazionaria in un ambiente chiuso, è possibile definire il concetto di *cammino libero medio* mfp come la distanza che il suono percorre mediamente prima di incontrare un ostacolo [2]:

$$mfp = \frac{4V}{S} \qquad [m]$$

a cui corrisponde un *tempo libero medio* mft pari a [2]:

$$mft = \frac{mfp}{v} \qquad [s]$$

con v la velocità del suono nel mezzo di propagazione.
È possibile perciò esprimere l'$RT60$ con le grandezze appena viste:

$$RT60 \ \cong\ -\ \frac{60 \cdot mft}{(1-\bar{a})_{dB}}$$

La formula di Sabine è ancora utilizzata per determinare i coefficienti di assorbimento dei materiali e per stimare l'$RT60$ in ambienti chiusi, tuttavia vanno imposti alcuni importanti vincoli:

- le tre dimensioni spaziali dell'ambiente non devono essere troppo dissimili;
- l'assorbimento dell'aria va considerato trascurabile, in realtà per grandi ambienti, alle alte frequenze, presenta un contributo significativo;
- l'attenuazione dell'onda sonora segua esattamente l'andamento esponenziale decrescente, mentre ciò è attendibile solo nei primi istanti.

Ai fini del calcolo e dell'individuazione dei tempi di riverberazione ottimali, in funzione degli eventi acustici considerati, si svilupparono una serie di misure caratteristiche: che evidenziarono come il parlato presenti meno riverberazione rispetto alla musica, e come opportuni accorgimenti ambientali valorizzino determinati generi musicali.

2.3.3 Tempo di Riverberazione Ottimale per un Ambiente Chiuso

Il fenomeno della riverberazione presenta un duplice effetto: da una parte, innalza l'intensità sonora rispetto a quella del singolo suono diretto; dall'altra, aumenta la dispersione temporale, degradando così l'intellegibilità del segnale audio.
Mantenendo controllato, entro certi limiti, il tempo di riverberazione, è possibile trovare un compromesso piuttosto efficiente fra i due effetti: in merito si utilizzano le seguenti formule empiriche che forniscono una stima del tempo di riverberazione ottimale in funzione del volume dell'ambiente, nel caso di eventi acustici differenti come musica e parlato:

$$RT60_{parlato} = 0{,}1\, V^{1/3} \qquad [s]$$

$$RT60_{musica} = 0{,}5 + \ 10^{-4}\, V \qquad [s]$$

2.3.4 Early Decay Time (EDT)

Uno dei parametri utilizzati per stimare la qualità acustica di un ambiente è l'EDT (Early Decay Time) che è pari a sei volte il tempo di riverberazione stimato sui primi $10\ [dB]$ di decadimento sonoro; tipicamente è inferiore rispetto il valore di $RT60$ e l'entità della discrepanza è un buon indice del grado di imperfezione della diffusione acustica; in generale è considerato uno dei parametri che meglio riescono a esprimere le sensazioni psicoacustiche soggettive.

2.4 Criteri Energetici

L'organo uditivo umano non riesce a distinguere chiaramente eventi sonori che si verificano a istanti ravvicinati nel tempo (circa $50\ [ms]$ per il parlato; $80\ [ms]$ per la musica), bensì considera le riflessioni come parte del suono diretto.
L'energia sonora percepita prima dei $50\ [ms]$ è definita energia utile perché i contributi delle riflessioni si aggiungono, enfatizzando, il suono diretto, mentre l'energia percepita oltre tale limite è considerata dannosa in quanto crea disturbi alla percezione sonora.

2.4.1 Definition – $D50$, Clarity – $C80$ e $ST1$

L'indice di definizione ($Definition - D50$) misura la chiarezza con la quale l'ascoltatore percepisce un messaggio audio, ed è definito nel seguente modo:

$$D50 = \frac{\int_{0\ ms}^{50\ ms} p^2(t)\, dt}{\int_{0\ ms}^{+\infty} p^2(t)\, dt} = \frac{Energia\ Utile}{Energia\ Totale} \qquad [dB]$$

L'indice di chiarezza ($Clarity\ -\ C80$), invece, valuta la possibilità per un ascoltatore, di percepire nitidamente suoni differenti in rapida successione o riprodotti contemporaneamente; è definito come il rapporto tra l'energia sonora percepita entro i primi $80\ [ms]$ e quella che giunge successivamente:

$$C80 = 10 \log \frac{D80}{(1 - D80)} = 10\ log \frac{\int_{0\ ms}^{80\ ms} p^2(t)\, dt}{\int_{80\ ms}^{+\infty} p^2(t)\, dt} \qquad [dB]$$

Infine l'$ST1$ descrive la facilità con la quale si percepiscono suoni differenti a una distanza pari a un metro dalla sorgente sonora:

$$ST1 = 10\,log\frac{\int_{10\,ms}^{100\,ms} p^2(t)\,dt}{\int_{0\,ms}^{10\,ms} p^2(t)\,dt} \qquad [dB]$$

2.5 Parametri di Intellegibilità

Alcuni dei parametri illustrati precedentemente vengono utilizzati anche come indici di riferimento per l'intelligibilità del parlato (*speech intelligibility*): infatti se l'ambiente acustico preso in considerazione ha una funzione strettamente inerente il parlato piuttosto che alla musica, è opportuno analizzare la situazione con dei criteri specifici di intelligibilità come l'STI (Speech Transmission Index) e il $RASTI$ (RApid Speech Transmission Index), che permettono inoltre di misurare la “distanza critica" oltre la quale viene compromessa la qualità dell'intelligibilità del parlato.

2.5.1 Distanza Critica

In un ambiente, a posizioni differenti corrispondono combinazioni tra suono diretto e riflesso differenti, si definisce *Distanza Critica* Dc la distanza fra la sorgente audio e la posizione spaziale di un ascoltatore caratterizzata dal fatto che il livello del suono diretto equivale il livello del suono riflesso:

$$Dc = 0{,}057\sqrt{(Q \cdot V/RT60)} \qquad [m]$$

Q è l’indice di direttività della sorgente sonora ed è definito come il rapporto tra il quadrato del livello di pressione sonora misurato sull’asse sorgente-ascoltatore, e il valore medio del quadrato del livello di pressione sonora, rilevato alla stessa distanza per tutte le direzioni di emissione:

$$Q = \frac{SPL^2_{asse}}{SPL^2_{medio}}$$

L’indice di direttività è direttamente proporzionale alla frequenza del segnale sonoro riprodotto: per tanto i suoni a frequenze più elevate saranno irradiati in prossimità dell’asse sorgente-ascoltatore.

2.6 Elementi di un Sistema Elettroacustico

I tre elementi che costituiscono la base di qualunque sistema sonoro sono: il microfono, l'altoparlante e l'amplificatore.

2.6.1 Il Microfono

Il microfono è un dispositivo che trasforma le variazioni di pressione acustica in corrispondente energia elettrica, permettendo così la trasmissione e l'amplificazione del suono. Questa conversione di energia opera sulla base di fenomeni induttivi, nel caso dei microfono magnetodinamici, o di variazioni di capacità nel caso di microfoni a condensatore.

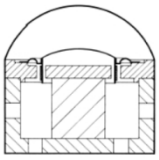


Figura 1.14: Sezione di una capsula microfonica dinamica

Microfoni magnetodinamici
Nei microfoni magnetodinamici il diaframma è solidale a una bobina, che scorre nel traferro di un magnete permanente, sotto l'azione della pressione sonora. Alle estremità della bobina si forma una tensione proporzionale all'intensità della pressione sonora che agisce sul diaframma stesso. È la tipologia di microfoni più diffusa: grazie alle buone prestazioni, alla solida struttura (poco sensibile all'umidità) e al basso costo che lo caratterizzano.

Microfoni elettrostatici o a condensatore
In questi microfoni il diaframma è l'armatura di un condensatore, il principio di funzionamento è basato sul fatto che la pressione sonora va a modificare la distanza fra le armature inducendo una variazione di capacità.
Presentano un livello di tensione in uscita piuttosto basso, quindi tipicamente sono accoppiati a un semplice preamplificatore interno, che ne amplifica la caratteristica I/O ottenendo ottimi livelli di sensibilità. Tuttavia, questo richiede la necessità di un'alimentazione a batteria o, come si trova comunemente, tramite i collegamenti di segnale con l'amplificatore (*phantom supply*).

Le prestazioni di un microfono vanno valutate in funzione della sensibilità, della qualità e della direttività. La qualità è proporzionale alla banda in frequenza, mentre la direttività è la capacità di selezionare direzioni preferenziali come sorgenti sonore. Una delle principali tipologie è la cardioide la quale è sensibile esclusivamente alle pressioni sonore frontali, limita così, la probabilità di captare suoni indesiderati o riflessi, riducendo in questo modo il pericolo di innescare l'effetto Larsen. Gli omnidirezionali o panoramici, invece, raccolgono tutti i suoni che li circondano su di un ampio fronte (*Figura 1.15*).[2]

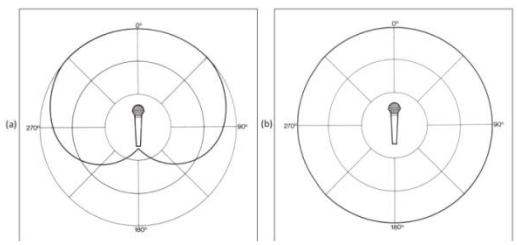


Figura 1.15: Risposta polare di un microfono cardioide (a sinistra) e un microfono omnidirezionale (a destra)

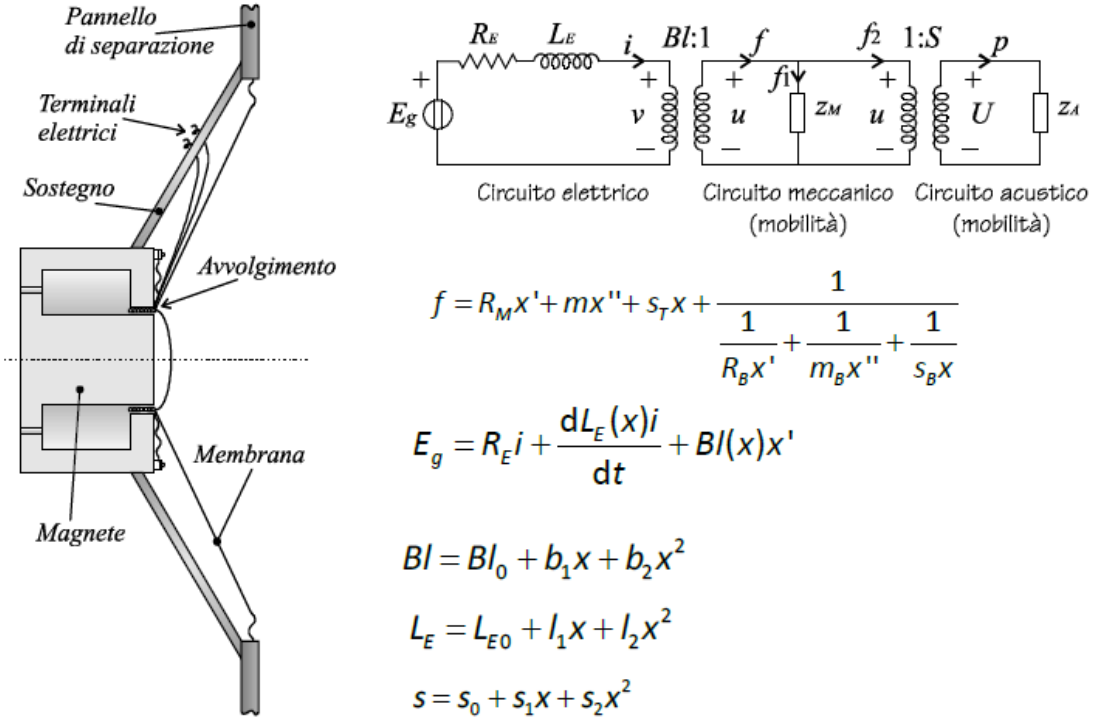
La sensibilità è la capacità di fornire, a parità di pressione sonora agente sulla membrana, una tensione d'uscita maggiore, è espressa in mV/Pa dove $20\ uPa = 0\ dB$.

2.6.2 L'amplificatore

Questo dispositivo ha la capacità di pilotare un altoparlante nonostante in ingresso il livello della sorgente sonora sia basso, in quanto ne eleva il segnale sufficientemente per l'ascolto su altoparlante. La scelta dell'amplificatore verte principalmente nella valutazione del numero di ingressi che si vogliono gestire, della potenza che si vuole sviluppare in funzione della tipologia di diffusori del sistema, della risposta in frequenza e della distorsione massima accettabile.

2.6.3 L'altoparlante

L'altoparlante converte energia elettrica in energia acustica, le tipologie di altoparlanti esistenti in commercio, in base al tipo di meccanismo di conversione sono molteplici. Il più diffuso è il magnetodinamico, nel quale vi è una bobina mobile percorsa da corrente immersa nel campo magnetico generato da un magnete permanente; la membrana o cono dell'altoparlante, è solidale con l'avvolgimento e vibrando genera una pressione sonora proporzionale alla tensione applicata ai capi della bobina stessa (*Figura 1.16*).
Le non linearità che introducono i trasduttori vengono generate principalmente dall'altoparlante e fissano limiti importanti alle prestazioni acustiche di un sistema.
Rappresentiamo, qualitativamente, uno dei modelli più diffusi (*modello di Klippel*) di queste non linearità:



$$f = R_M x' + m x'' + s_T x + \frac{1}{\frac{1}{R_B x'} + \frac{1}{m_B x''} + \frac{1}{s_B x}}$$

$$E_g = R_E i + \frac{\mathrm{d}L_E(x) i}{\mathrm{d}t} + Bl(x) x'$$

$$Bl = Bl_0 + b_1 x + b_2 x^2$$

$$L_E = L_{E0} + l_1 x + l_2 x^2$$

$$s = s_0 + s_1 x + s_2 x^2$$

Figura 1.16: Sezione altoparlante magnetodinamico a cono e relativo circuito equivalente (modello di Klippel)

Capitolo 3

AEC (Acoustic Echo Cancellation)

Il problema dell'eco acustica è un aspetto non trascurabile nella progettazione di un sistema di comunicazione in ambienti fortemente riverberanti o nel caso di sistemi realizzati con componenti ed elementi che per le loro caratteristiche intrinseche inneschino facilmente fenomeni di eco; i quali non solo possono compromettere la stabilità dei sistemi operanti ad anello chiuso (full-duplex), innescando l'effetto Larsen, ma anche generano disturbi che ne diminuiscono l'intellegibilità della comunicazione; è pertanto auspicabile che tali sistemi prevedano un dispositivo adibito alla cancellazione dell'eco acustica (Acoustic Echo Canceller, AEC). In generale, il comportamento acustico di un sistema è tempo-variante e le tecniche numeriche utilizzate per realizzare un AEC si basano sul filtraggio adattativo basato su algoritmi come l'LMS (Least-Mean-Square), l'$NLMS$ (Normalized Least-Mean-Square) e l'RLS (Recursive-Least-Squares).
A fronte di queste considerazioni, le prestazioni di un AEC sono determinate, a parità di massima attenuazione ottenibile, dalla velocità di convergenza e dal costo computazionale. Essendo la voce umana un segnale colorato, per ottenere prestazioni soddisfacenti in termini di velocità di convergenza, l'algoritmo LMS, per esempio, deve essere impiegato all'interno di schemi di filtraggio sofisticati, o viceversa RLS deve essere impiegato in schemi altamente performanti.
Si tratteranno i fondamenti teorici della cancellazione dell'eco acustica, presenteremo una panoramica sulla realizzazione di un AEC basato sugli algoritmi LMS e $NLMS$ e facendo riferimento a simulazioni in linguaggio $MATLAB$, si confronteranno le prestazioni tra le due soluzioni.
Analizzeremo, un modello semplificato, sufficiente per comprendere gli aspetti principali della cancellazione dell'eco acustica, ma si considererà il problema della cancellazione dell'eco solo lato terminale ricevente; verrà inoltre trascurato il ritardo introdotto dalla rete di comunicazione fra i terminali presenti nel sistema, al fine di concentrarsi solo su quello acustico e quello introdotto dalle operazioni numeriche di filtraggio.

Sotto queste ipotesi è possibile rappresentare il problema della cancellazione dell'eco mediante un semplice modello come quello in *Figura 3.1*. Il sistema è costituito da:

- una sorgente sonora lontana (*far end*), che in assenza dell'AEC è infastidita dall'ascoltare l'eco della propria voce
- una seconda sorgente sonora vicina (*near end*), che si trova nel terminale ricevente caratterizzato da un determinato comportamento acustico
- almeno due coppie microfono-altoparlante, una per ogni sorgente sonora.

Il compito dell'*AEC* è quello di rimuovere dal segnale catturato dal microfono la voce della sorgente sonora lontana.

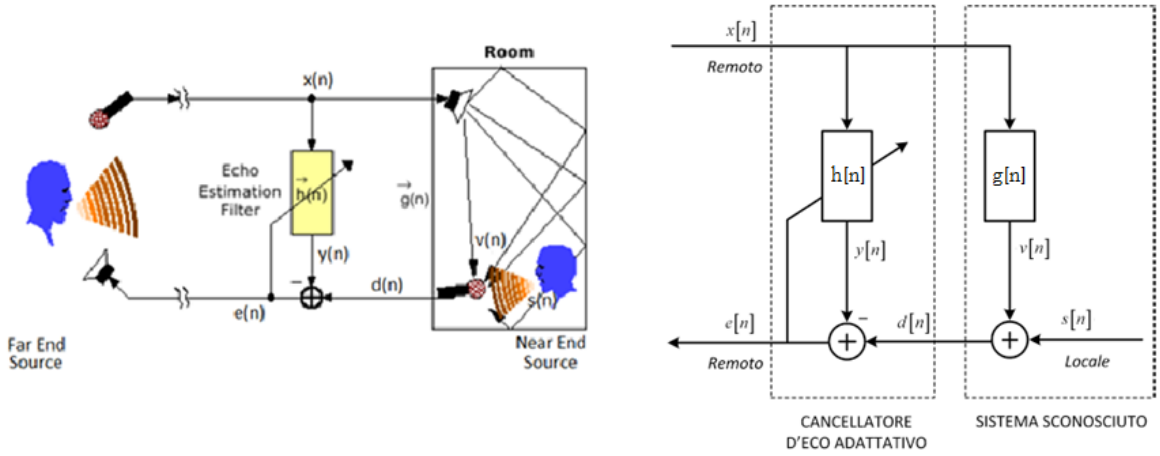


Figura 3.1: Modello di un sistema per la cancellazione d'eco acustica

Come anticipato, la tecnica di cancellazione dell'eco acustica che analizzeremo consiste nell'utilizzo di un filtro numerico adattativo basato su algoritmo LMS (*Least Mean Square*) in particolare, data la natura non stazionaria della voce, si utilizzerà la variante normalizzata $NLMS$ (*Normalized LMS*); l'idea è quella di utilizzare il blocco adattativo $NLMS$ per identificare un sistema incognito (*Unknown System o Plant*), che nella fattispecie corrisponde al cammino acustico tra l'altoparlante e il microfono nel terminale di ricezione. Più precisamente, il compito dell'*AEC* nel terminale ricevente è rimuovere dal segnale totale catturato dal microfono $d(n)$ (*Desired Signal*), quella porzione $v(n)$ di segnale che presenta correlazione con il segnale di far end $x(n)$ (*Reference Signal*), convoluto con la risposta impulsiva del sistema fisico reale $g(n)$ corrispondente al cammino acustico fra l'altoparlante e il microfono. Così facendo, sì eviterà che alla sorgente di far end si ripresenti, insieme al segnale utile $s(n)$, prodotto dalla sorgente di near end, il proprio segnale ritardato e filtrato. Il cancellatore deve pertanto emulare il comportamento filtrante della stanza $v(n)$, generandone una stima $y(n)$, a partire da $x(n)$ e dal segnale di errore $e(n)$ (ottenuto come differenza tra il segnale totale ricevuto al microfono $d(n)$ e la stima stessa), quindi sottrarla al segnale microfonico $d(n)$, realizzando di fatto la cancellazione dell'eco (*Figura 3.2*). Infine il segnale "pulito" $e(n)$ verrà spedito alla sorgente di far end.
Le relazioni matematiche fra i vari segnali e una trattazione matematica rigorosa dell'algoritmo verrà affrontata nel prossimo capitolo.

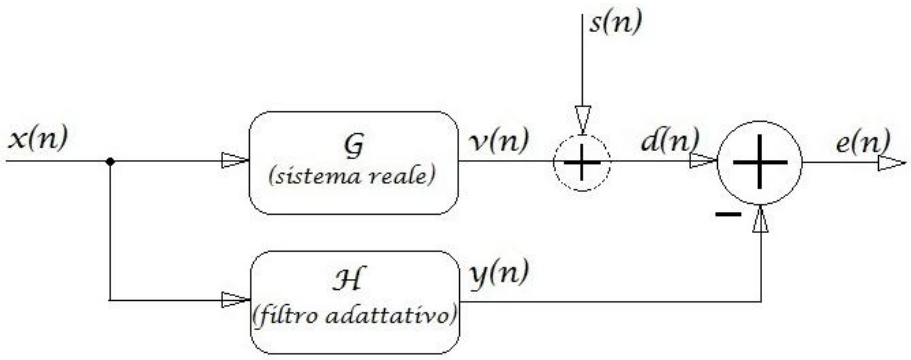


Figura 3.2: Modello del sistema

Affinché il filtro adattativo possa riprodurre la risposta impulsiva del cammino acustico, l'ordine del filtro deve essere comparabile con quello della risposta impulsiva desiderata.
La conseguenza principale di questo fatto è che la lunghezza del filtro adattativo cresce proporzionalmente al massimo ritardo che si desidera coprire, pertanto in applicazioni caratterizzate da ambienti con tempi di riverbero significativi la lunghezza richiesta al filtro potrebbe essere proibitiva dal punto di vista del costo computazionale.

3.1 Misurazione delle Prestazioni di un AEC

3.1.1 Echo Return Loss Enhancement (ERLE)

L'$ERLE$ misura la quantità di perdita qualitativa introdotta nel processo di cancellazione dell'eco ed è definito come il rapporto fra la potenza istantanea del segnale desiderato $d(n)$ e la potenza istantanea dell'eco residuo $v(n)$ dopo la cancellazione:

$$ERLE = 10\log\frac{P_d(n)}{P_v(n)} = 10\log\frac{E[d^2(n)]}{E[v^2(n)]} \qquad [dB]$$

3.1.2 Disallineamento Normalizzato

Questo criterio viene utilizzato per valutare la fedeltà del filtro adattativo stimato rispetto alla prestazione ottimale:

$$\mathfrak{M} = 20\log\left(\frac{\|\boldsymbol{g}_n - \boldsymbol{h}_n\|_2}{\|\boldsymbol{g}_n\|_2}\right) \qquad [dB]$$

3.2 Sistemi LEMS Reali

I sistemi presi in considerazione finora vengono denominati ***LEMS***, *Loudspeaker Enclosure Microphone System* e sono caratterizzati dal modello semplificato di *Figura 3.3*, il quale rispetto al caso reale non tiene in considerazione una serie di limitazioni che ne complicano di gran lunga il comportamento: come la presenza di rumore (acustico, ambientale, termico e circuitale), la precisione numerica finita dei *DSP* che ne implementano la realizzazione, il troncamento e il sotto-modellamento della funzione di trasferimento acustica, gli effetti delle vibrazioni sull'intero sistema, le non linearità introdotte dagli altoparlanti, il compromesso fra la convergenza e la precisione dell'*AEC*, …

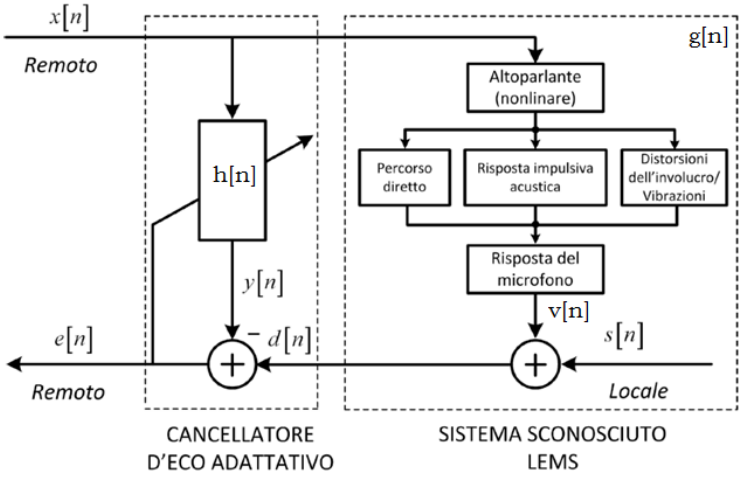


Figura 3.3: Modello LEMS Reale

In particolare, il sotto-modellamento della funzione di trasferimento acustica si verifica quando la lunghezza del filtro adattativo è minore della lunghezza della risposta impulsiva dell'ambiente, in tali condizioni la rimanente porzione di risposta impulsiva non cancellata si manifesta sotto forma di errore all'uscita del cancellatore. Tuttavia l'aumento "alla cieca" della lunghezza del filtro comporta una complessità e un rumore dell'algoritmo notevoli e una velocità di convergenza piuttosto modesta.
In questi termini viene definito il massimo $ERLE$ raggiungibile dal sistema come il rapporto TIP/TP (*Total Impulse Power to Tail Portion*):

$$\frac{TIP}{TP} = 10\log\frac{\sum_{i=0}^{M-1} h^2(i)}{\sum_{i=N}^{M-1} h^2(i)} \qquad [dB]$$

dove h è la risposta impulsiva del sistema di lunghezza pari a M, e N è il punto di partenza della "coda" della risposta impulsiva.

3.3 Filtraggio Adattativo

In generale, la risposta di un canale di comunicazione non è nota a priori con precisione e quindi l'eliminazione di un eventuale eco non può essere realizzata con filtri di ricezione a coefficienti costanti o con l'impiego di filtri analogici: i quali non avendo memoria non riescono a tenere traccia del segnale in questione.
Viene, dunque, impiegato un filtro digitale adattativo: i cui coefficienti sono modificati e aggiornati, mediante algoritmi specifici, in funzione del comportamento caratteristico del canale; inoltre questi filtri sono parecchio flessibili in quanto possono venire riprogrammati in tempo reale, via software, senza intervenire fisicamente sull'hardware.

3.4 Filtri Digitali

Un filtro digitale è un sistema lineare stazionario (SLS) a tempo discreto, la cui caratteristica ingresso-uscita è esprimibile come convoluzione discreta:

$$y(n) = \sum_{k=-\infty}^{+\infty} x(k)\, h(n-k) = x(n) * h(n)$$

dove $y(n)$ è l'uscita del sistema, $x(n)$ l'ingresso e $h(n)$ la risposta impulsiva cioè l'uscita del sistema nel caso l'ingresso sia un impulso discreto $\delta(n)$.

Figura 3.4: Filtraggio digitale

La risposta in frequenza del sistema $H(f)$ è la trasformata di Fourier della risposta impulsiva:

$$H(f) = \sum_{n=-\infty}^{+\infty} h(n)\, e^{-j2\pi nfT} df$$

Un'importante classe di filtri digitali è quella che caratterizza i sistemi $ARMA$ (Auto Regressive Moving Average:

- Filtri AR = Auto Regressive
- Filtri MA = Moving Average

Matematicamente, questi sistemi sono descritti da equazioni alle differenze, dove le operazioni realizzate sono moltiplicazione per una costante, introduzione di un ritardo elementare e la somma algebrica.

3.4.1 Filtri MA (Moving Average)

Vediamo la struttura dei filtri *Moving Average*, a cui appartengono i filtri FIR (Finite Impulse Response), che costituiscono l'elemento base del cancellatore d'eco realizzato.
Un'espressione di un filtro Moving Average è del tipo:

$$y(n) = \frac{1}{N} \sum_{k=n-N+1}^{n} x(k)$$

schematizzata nel modo seguente:

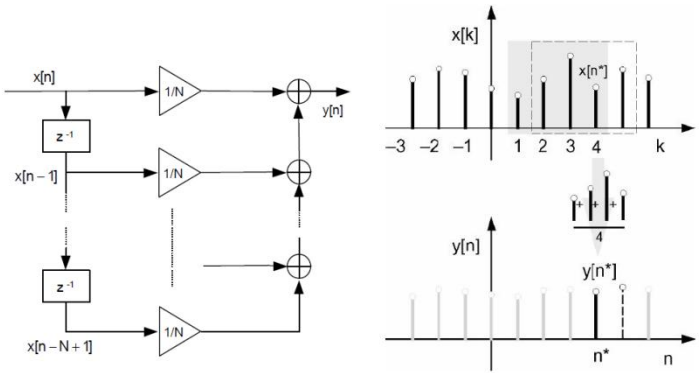


Figura 3.5: Schematizzazione di un sistema MA con N = 4 campioni

dove sono presenti rispettivamente $N-1$ ritardi, N moltiplicatori e $N-1$ sommatori.
Il filtro effettua la media aritmetica degli ultimi N campioni dell'ingresso (nell'esempio di *Figura 3.5* sono 4 campioni). Al variare di n variano i campioni che vengono mediati, con la conseguenza che la media risulterà mobile, da cui il nome filtri MA (*Moving Average*).
Applicando in ingresso al sistema un impulso $\delta(k)$ è possibile calcolarne la risposta impulsiva:

$$h(n) = \frac{1}{N} \sum_{k=n-N+1}^{n} x(k) = \frac{1}{N} \sum_{k=n-N+1}^{n} \delta(k) = \frac{1}{N} \sum_{k=0}^{N-1} \delta(n-k)$$

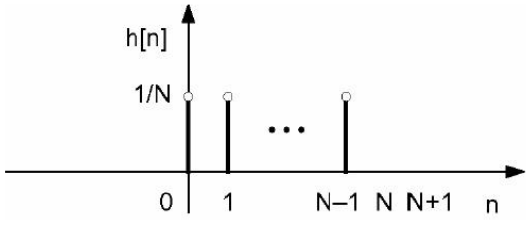


Figura 3.6: Risposta impulsiva di un sistema MA

La lunghezza della risposta impulsiva è finita e pari a N campioni, questa caratteristica identifica una classe di filtri digitali detti ***FIR* (*Finite Impulse Response*)**.
In generale la forma di un filtro FIR di ordine N è la seguente:

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

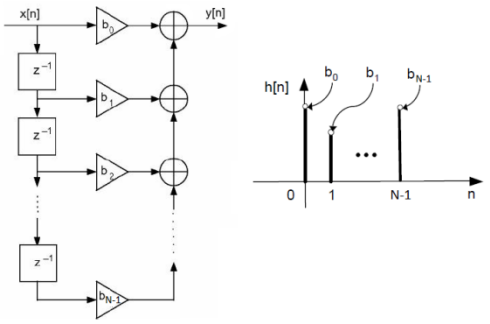


Figura 3.7: Struttura generica di un filtro FIR

La cui uscita è ottenuta come media ponderata dei pesi $b_0, b_1, b_2, \dots, b_{N-1}$ degli ultimi N campioni del segnale d'ingresso.
I filtri *FIR* sono sempre stabili, per qualunque scelta dei coefficienti.

3.4.2 Filtri AR (Auto Regressive)

Per completezza, riportiamo l'analoga struttura dei filtri *Auto Regressive*, a cui appartengono i filtri IIR (I*nfinite Impulse Response*).
Un'espressione di un filtro *Auto Regressive* è del tipo:

$$y(n) = a\, y(n-1) + b\, x(n)$$

schematizzata nel modo seguente:

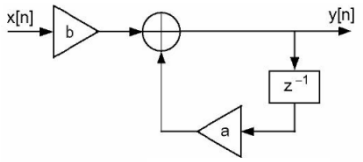


Figura 3.8: Schematizzazione di un sistema AR

Il legame tra ingresso e uscita non è stato esplicitato e compare la relazione ricorsiva che caratterizza questa tipologia di filtri. Ricavando la risposta impulsiva del filtro e fissando come condizione iniziale del sistema $h(n)|_{n<0} = 0$ troviamo ricorsivamente:

$$h(n)_{n\geq 0} = a^n b\, \delta(n)$$

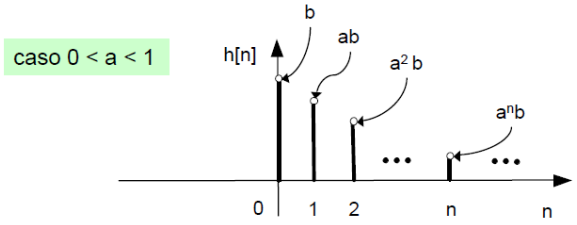


Figura 3.9: Coefficienti di un filtro AR

Con $0 < a < 1$ otteniamo una sequenza esponenziale unilatera e si può notare che la lunghezza della risposta impulsiva è infinita, questa caratteristica identifica una classe di filtri digitali detti ***IIR* (*Infinite Impulse Response*)**.

In generale la forma di un IIR è la seguente:

$$y(n) = -\sum_{k=1}^{N} a_k\, y(n-k) + b_0\, x(n)$$

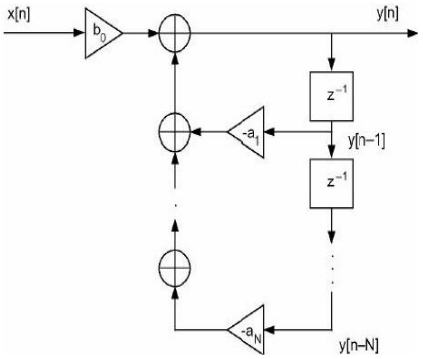


Figura 3.10: Struttura generica di un filtro IIR

L'uscita all'istante n dipende dall'ingresso allo stesso istante e dagli N valori precedenti dell'uscita stessa. La stabilità dei filtri IIR è legata alla scelta dei coefficienti $a_0, a_1, a_2, \dots, a_N$, tuttavia con questa tipologia di filtri si può realizzare lo stesso tipo di filtraggio dei filtri *FIR* con un numero minore di componenti elementari (moltiplicatori, ritardi e sommatori).

Una caratterizzazione, generale, dei filtri FIR consiste nel considerarli come sistemi LTI causali con risposta finita all'impulso, la cui funzione di trasferimento è un polinomio in z^{-1}; analogamente i filtri IIR si possono vedere come sistemi LTI causali con risposta anche infinita all'impulso, la cui funzione di trasferimento è razionale in z^{-1}.
L'impiego di filtri FIR è preferito rispetto i filtri IIR, in tutte quelle applicazioni dove vi siano specifiche restrittive inerenti la distorsione di fase: come per esempio l'esigenza di preservare la linearità di fase. Tuttavia se ciò non è richiesto l'utilizzo dei filtri IIR rispetto i FIR comporta notevoli vantaggi: un minor numero di parametri, un risparmio di memoria e una minor complessità computazionale.

3.4.3 Filtri FIR e Filtri a Fase Lineare

Un sistema LTI causale a tempo discreto con risposta all'impulso unitario $h(n)$ finita ($h(n) = 0$, per $n < 0$ e per $n \geq M$ con $M > 0$ intero), si definisce filtro FIR e la caratteristica I/O è descritta dalla seguente equazione alle differenze finite:

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k)$$

Passando alla trasformata z e applicando la proprietà della traslazione temporale, si ottiene:

$$Y(z) = H(z)\, X(z)$$

con $H(z) = \sum_{k=0}^{M-1} h(k)z^{-k}$, e $X(z)$, $Y(z)$ le trasformate z rispettivamente di $x(n)$ e $y(n)$.
Le caratteristiche più importanti dei filtri *FIR* sono:

- causalità e stabilità: infatti $H(z)$ è un polinomio in z^{-1}, e quindi ha un solo polo in $z = 0$ interno alla circonferenza di raggio unitario.

- fase lineare: se la funzione $h(n)$ è simmetrica o antisimmetrica rispetto a $(M-1)/2$ (cioè $h(k) = h(M-1-k)$ oppure $h(k) = -h(M-1-k)$), la fase $\measuredangle H(e^{jw})$ risulta lineare.
  Infatti, se $h(k) = h(M-1-k)$:

$$\begin{aligned} H(e^{jw}) &= \sum_{k=0}^{M-1} h(k)e^{-jkw} \\ &= \sum_{k=0}^{M-1} h(k)\frac{e^{-jkw} + e^{-j(M-1-k)w}}{2} \\ &= e^{-j\frac{M-1}{2}w} \sum_{k=0}^{M-1} h(k)\frac{e^{j(\frac{M-1}{2}-k)w} + e^{-j(\frac{M-1}{2}-k)w}}{2} \\ &= e^{-j\frac{M-1}{2}w} \sum_{k=0}^{M-1} h(k)\cos\left(\frac{M-1}{2}-k\right)w \end{aligned}$$

  Poiché $\sum_{k=0}^{M-1} h(k)\cos\left(\frac{M-1}{2}-k\right)w$ è un numero reale, la fase risulta lineare:

$$\measuredangle H(e^{jw}) = -\frac{M-1}{2}w$$

  Analogamente, se $h(k) = -h(M-1-k)$ si ottiene:

$$H(e^{jw}) = je^{-j\frac{M-1}{2}w} \sum_{k=0}^{M-1} h(k)\sin\left(\frac{M-1}{2}-k\right)w$$

  osservando che $j = e^{j\frac{\pi}{2}}$, si ottiene:

$$\measuredangle H(e^{jw}) = -\frac{M-1}{2}w + \frac{\pi}{2}$$

In *Figura 3.11* sono rappresentate le risposte all'impulso di due filtri *FIR* a fase lineare, uno antisimmetrico, l'altro simmetrico.

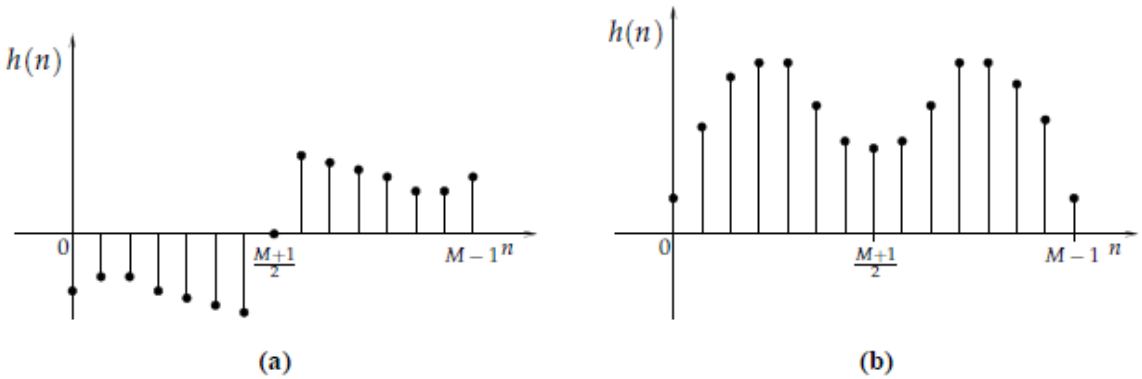


Figura 3.11: Risposta di un filtro FIR a fase lineare, (a) antisimmetrico, (b) simmetrico

3.4.4 Filtri IIR

Si consideri ora un sistema LTI in cui la caratteristica I/O soddisfi la seguente equazione:

$$\sum_{k=0}^{L-1} a_k y(n-k) = \sum_{k=0}^{M-1} b_k y(n-k) \qquad (a_{L-1} \neq 0)$$

Per $L = 1$ la relazione precedente è la stessa di un filtro FIR, invece per $L > 1$ non riesce a specificare univocamente il sistema: infatti passando alle trasformate z e applicando la proprietà della trasformata temporale, si ottiene:

$$Y(z) = H(z)\, X(z)$$

dove $H(z) = \frac{\sum_{k=0}^{M-1} b_k z^{-k}}{\sum_{k=0}^{L-1} a_k z^{-k}}$ e $X(z)$, $Y(z)$ sono le trasformate z rispettivamente di $x(n)$ e $y(n)$; poiché $H(z)$ è una funzione razionale in z^{-1} dotata di poli non nulli, possiamo descrivere più sistemi caratterizzati dalla stessa funzione $H(z)$, ma con differenti corone di convergenza. Un sistema LTI causale è definito filtro IIR, se la risposta $h(n)$ all'impulso unitario è nulla per $n < 0$, ma diversa da 0 per infiniti n positivi.
Una delle più importanti caratteristiche dei filtri IIR consiste nel fatto che la fase non può mai essere lineare, pertanto sono soggetti a problemi di stabilità; tuttavia presentano una maggior semplicità realizzativa rispetto ai filtri FIR e migliori caratteristiche di attenuazione a parità di ordine del filtro

3.5 Progettazione di Filtri Digitali

Le caratteristiche che deve avere un filtro digitale, tipicamente sono espresse nel dominio delle frequenze tramite uno schema di tolleranza: cioè una maschera applicata al guadagno in frequenza che ne specifica le proprietà.
Per esempio, gli elementi principali di uno schema di tolleranza di un filtro passa-basso (*Figura 3.12*) sono:

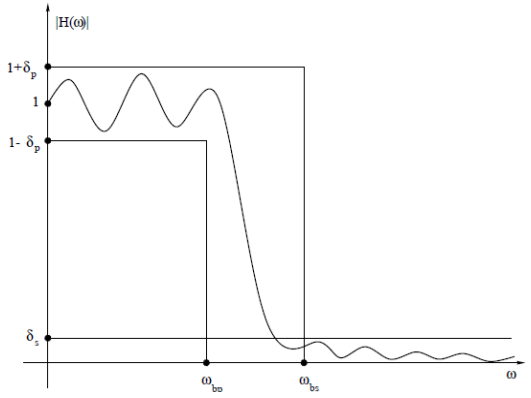


Figura 3.12: Specifiche di un filtro passa basso

- le frequenze w_{bp} e w_{bs} che definiscono rispettivamente i vincoli per la banda passante e la banda di transizione, secondo la relazione: $w_{bp} \leq w_c < w_s \leq w_{bs}$ con w_c la frequenza di taglio e w_s la frequenza di stop del filtro.
- le dimensioni massime δ_p e δ_s permesse alle oscillazioni in banda passante e banda proibita, le quali vengono espresse nel seguente modo:

$$A_p = 20\ log \frac{1+\ \delta_p}{1-\delta_p} \approx 17{,}4\ \delta_p\ [dB] \qquad A_s = -20\ log\ \delta_s \quad [dB]$$

dove A_p denota l'oscillazione in banda passante, mentre A_s denota l'attenuazione in banda proibita.

È consuetudine trovare le frequenze indicate normalizzate rispetto la frequenza di campionamento w_0, mentre per quanto riguarda la risposta in fase, usualmente sono sufficienti specifiche di tipo qualitativo sulla linearità.

3.5.1 Accorgimento sul Progetto di Filtri FIR: l'utilizzo delle Finestre

L'utilizzo della tecnica delle finestre è basata sull'idea di approssimare un filtro desiderato, eventualmente non causale e con risposta impulsiva $h_d(n)$ di durata infinita, azzerando tale risposta al di fuori di una finestra temporale di ampiezza N, nella speranza che l'approssimazione sia tanto migliore quanto più la dimensione N della finestra sia ampia.
Più precisamente:

1. Si consideri il filtro desiderato con risposta all'impulso $h_d(n)$ eventualmente di durata infinita;
2. Fissato un intero N, si costruisce il filtro FIR con risposta all'impulso $h_N(n)$ tale che:

$$h_N(n) = \begin{cases} h_d(n), & se\ |n| \le (N-1)/2 \\ 0, & altrimenti \end{cases}$$

3. Si ottiene, infine, il filtro FIR causale con risposta all'impulso $h_N(n - N/2)$

Considerando la *finestra rettangolare* $rett_N(n)$ data da:

$$rett_N(n) = \begin{cases} 1, & se\ |n| \le (N-1)/2 \\ 0, & altrimenti \end{cases}$$

possiamo scrivere:

$$h_N(n) = rett_N(n)\ h_d(n)$$

In altre parole, $h_N(n)$ è ottenuta moltiplicando la risposta all'impulso del filtro che si desidera approssimare per la finestra rettangolare $rett_N(n)$ di durata finita.
Studiamo, ora come il filtro con risposta $h_N(n)$ approssima il filtro $h_d(n)$, al variare di N.
Per semplicità assumiamo che il filtro passa-basso ideale caratterizzato dalla seguente risposta in frequenza:

$$H_d(e^{jw}) = \begin{cases} 1, & se\ |w| \le 1 \\ 0, & se\ 1 < |w| \le \pi \end{cases}$$

Denotiamo con $H_N(e^{jw})$ la risposta in frequenza del filtro $h_N(n)$.
Si possono osservare due fenomeni:

- $H_N(e^{jw})$ ha una banda di transizione non nulla, la cui ampiezza converge a 0 quando N diverge;
- $|H_N(e^{jw})|$ presenta delle oscillazioni, sia in banda passante che in quella proibita, la cui ampiezza massima è indipendente dalla dimensione temporale N della finestra.

Il secondo fenomeno è particolarmente negativo, ed è legato al tipo di convergenza della serie di Fourier (*fenomeno di Gibbs*); esso tuttavia si può limitare, scegliendo opportune

finestre $w_N(n)$: diverse da 0 nell'intervallo $-(N-1)/2 \leq n \leq (N-1)/2$, simmetriche rispetto all'origine e non dovranno presentare discontinuità di salto come quella rettangolare.
La relazione tra il filtro desiderato e il filtro FIR ottenuto con la finestra $w_N(n)$ è:

$$h_N(n) = w_N(n)\, h_d(n)$$

Vediamo alcuni esempi di finestre comunemente impiegate :
Finestra Triangolare (o Bartlett):

$$w_N(n) = \begin{cases} 1 + \dfrac{2n}{N}, & se\ -(N-1)/2 \leq n < 0 \\ 1 - \dfrac{2n}{N}, & se\ 0 \leq n \leq (N-1)/2 \end{cases}$$

Finestra di Hanning:

$$w_N(n) = \frac{1}{2}\left[1 + cos\frac{2\pi n}{N}\right], \qquad |n| \leq (N-1)/2$$

Finestra di Hamming:

$$w_N(n) = 0{,}54 + 0{,}46\ cos\frac{2\pi n}{N}, \quad |n| \leq (N-1)/2$$

In *Figura 3.13* sono illustrate graficamente le finestre sopra elencate:

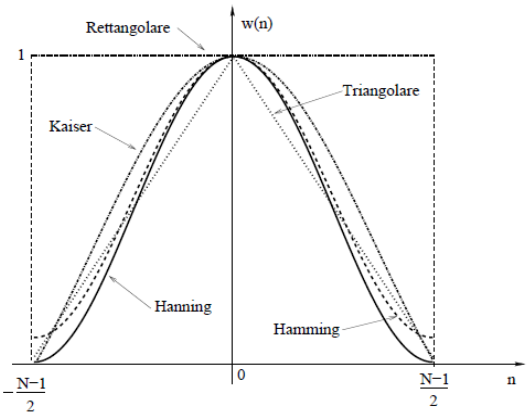


Figura 3.13: Finestre comunemente usate per la progettazione di filtri FIR

La progettazione mediante finestre richiede di operare due scelte, la tipologia di finestra e la dimensione N dell'intervallo, tenendo in considerazione che:

- la deviazione massima delle oscillazioni dipende dalla scelta della finestra
- l'estensione della banda di transizione è inversamente proporzionale a N e legata alla tipologia di finestra scelta.

La *Tabella 3.1* riporta l'ampiezza delle oscillazioni e la banda di transizione (normalizzata alla frequenza di campionamento) per la finestra rettangolare, quella di Hanning e quella di Hamming:

Finestra	*Osc.* A_p (dB)	*Att.* A_s (dB)	*Amp. trans.* (Hz)
Rettangolare	0.74	21	$1/N$
Hanning	0.05	44	$3.1/N$
Hamming	0.02	53	$3.3/N$

Tabella 3.1: Ampiezza delle oscillazioni e banda di transizione (normalizzata) di alcune finestre

Si può notare che solo la finestra di Hamming permette un'attenuazione di almeno $50\ dB$.

3.5.2 Rumore nel Disegno di Filtri Digitali

L'elaborazione di segnali analogici mediante sistemi digitali richiede la conversione di valori reali in numeri digitali, rappresentabili con un numero finito di bit: implicando così un approssimazione nella rappresentazione con l'introduzione di errori inevitabili. Presentiamo qui una breve rassegna sui diversi tipi di errori che si vengono a creare nella realizzazione di filtri mediante sistemi digitali; si analizzeranno in particolare gli errori dovuti alla quantizzazione dei coefficienti che entrano nella specifica del filtro.

Rumore per quantizzazione del segnale.
Tutti i convertitori analogico-digitale modificano il segnale di ingresso, introducendo quindi un errore di quantizzazione detto *rumore granulare del quantizzatore*: proporzionale al numero di bit del segnale in uscita.

Rumore per quantizzazione dei coefficienti.
Un filtro digitale viene specificato attraverso l'algoritmo (o attraverso la rete) che lo implementa dove le operazioni di moltiplicazione per una costante richiedono a loro volta l'assegnazione di opportuni coefficienti reali che vengono programmati come parametri. L'implementazione dell'algoritmo su un processore con parola di lunghezza fissata (tipicamente 16 o 32 bit), pone dei limiti all'accuratezza con cui possono essere specificati i parametri: viene quindi introdotto un errore, detto *errore di quantizzazione dei coefficienti*, per cui il filtro implementato non coincide in generale con quello specificato.
Osservazioni:

- Gli effetti della quantizzazione dei coefficienti dipendono dall'aritmetica di macchina del processore su cui il filtro è implementato: la rappresentazione in virgola fissa è generalmente più sensibile a tali errori, a causa della propagazione dell'errore nella moltiplicazione. Su DSP (Digital Signal Processor) a 32 bit o più, con rappresentazione in virgola mobile, questo tipo di errore può invece essere trascurato.
- I filtri IIR, rispetto ai filtri FIR, sono più sensibili all'errore di quantizzazione dei coefficienti, a causa della struttura intrinsecamente ricorsiva.
- Lo stesso filtro può essere realizzato con diverse architetture di rete: la sensibilità all'errore di quantizzazione dei coefficienti è fortemente legata al tipo di rete realizzata. Ad esempio, la sensibilità all'errore di reti che realizzano un filtro in forma diretta è generalmente più alta rispetto alle reti che realizzano lo stesso filtro in forma di cascata o parallelo.

Rumore per troncamento.
L'implementazione di un algoritmo che realizza un filtro digitale richiede l'esecuzione di varie operazioni di somma e prodotto. Ipotizziamo di utilizzare una rappresentazione in virgola fissa, dove l'ingresso, l'uscita e i coefficienti del filtro sono numeri rappresentabili con n bit, mantenere nei calcoli questa accuratezza richiede una precisione maggiore poiché, tipicamente, la moltiplicazione di due numeri di n bit produce un numero rappresentabile con $2n$ bit. La necessità di arrotondare i risultati intermedi produce inevitabilmente un errore detto *rumore di troncamento*. Una delle tecniche per controllare l'errore di troncamento è quella di spostare l'operazione di arrotondamento il più possibile nella parte finale del calcolo. A tal riguardo, risulta molto utile implementare un filtro digitale su processori che, lavorando con parole di n bit, hanno registri come l'accumulatore (ACC) o il prodotto (P) di dimensione doppia, cioè $2n$ bit.

3.5.3 Funzionalità Aggiuntive Presenti in un AEC

Un AEC oltre all'elemento principale, il filtro adattivo, può contenere numerosi altri componenti che ne sviluppano le funzionalità a discapito di una maggiore complessità circuitale e un costo più elevato, alcune di queste caratteristiche sono:

- *Far End Speech Detector (FESD)*
- *Near End Speech Detector (NESD)*
- *Double Talk Detector (DTD)*
- *Control Unit (CU)*
- *Non Linear Processor (NLP)*
- *Noise Reduction Unit (NR)*
- *Automatic Gain Control (AGC)*

Questi componenti si integrano al filtro adattivo in modo tale da ottimizzare e migliorare la comunicazione.

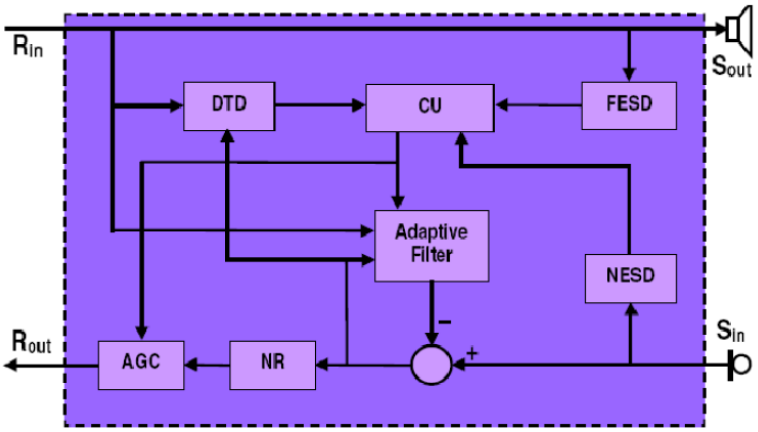


Figura 3.19: Schema di un AEC avanzato

Capitolo 4

Algoritmi per il Filtraggio Adattativo

Vediamo ora, di analizzare in dettaglio, alcuni dei principali algoritmi, che implementano il filtraggio adattativo, che nello schema a blocchi del sistema (*Figura 4.1*) vengono rappresentati dal blocco H, i quali chiaramente devono emulare il sistema fisico reale G soggetto all'eco acustico.
Ricordiamo che: $x(n)$ è il segnale proveniente dal far end e lo ritroviamo allo speaker del near end; $v(n)$ è il contributo di segnale, che raggiunge il microfono del near end, dovuto all'accoppiamento acustico fra speaker e microfono; $s(n)$ è la sorgente sonora lato near; $d(n)$ è il segnale totale che viene ricevuto al microfono; $y(n)$ è l'uscita del filtro adattativo, rappresenta un'emulazione del segnale $v(n)$; $e(n)$ è il segnale di errore ottenuto come differenza fra $d(n)$ e $y(n)$, è il segnale che viene spedito al far end.

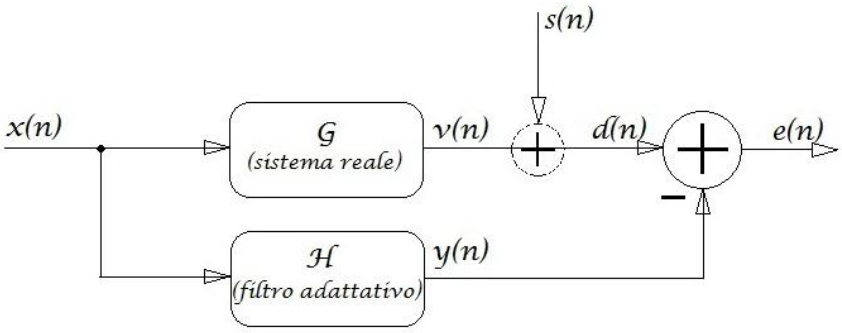


Figura 4.1: Schema a blocchi del filtraggio adattativo

4.1 Principio di Ortogonalità

La relazione ingresso-uscita di un generico filtro adattativo H (*Figura 4.1*) è la seguente:

$$y(n) = \sum_{k=0}^{M} h_k^* \; x(n-k)$$

dove h_k sono i pesi del filtro e si assume che sia l'ingresso $x(n)$ che la risposta $d(n)$ siano processi congiuntamente stazionari in senso lato con media nulla.
L'errore da minimizzare (nel caso di assenza di una sorgente sonora al near end) è $e(n) = d(n) - y(n)$, si sceglie di cercare la soluzione ottima del problema minimizzando la funzione costo:

$$J = E\big(e(n)e^*(n)\big) = E[|e(n)|^2]$$

Calcoliamo il minimo della funzione costo:

$$\nabla_k J = -2E[x(n-k)\, e^*(n)]$$

Imponiamo quindi il valore di gradiente nullo e otteniamo $E[x(n-k)\, e_0^*(n)] = 0$, questo risultato è denominato *principio di ortogonalità* e in base a questo possiamo scrivere:

$$E\left[x(n-k)\left(d^*(n) - \sum_{i=0}^{\infty} w_{oi} x^*(n-i)\right)\right] = 0 \quad k = 0,1,\dots$$

$$\sum_{i=0}^{\infty} h_{oi}\, E[x(n-k)\, x^*(n-i)] = E[x(n-k)\, d^*(n)]$$

$$\sum_{i=0}^{\infty} h_{oi}\, r(i-k) = p(-k) \qquad k = 0,1,\dots$$

L'estensione al caso di filtri FIR è immediata ponendo l'indice superiore della sommatoria a un valore finito M.

4.2 Matrice di Wiener-Hopf

Si definisce:

$$\boldsymbol{R} = E[\boldsymbol{x}(n)\boldsymbol{x}^H(n)]$$

la matrice di autocorrelazione dei campioni in ingresso $\boldsymbol{x}(n)$:

$$\boldsymbol{x}(n) = [x(n), x(n-1), \dots, x(n-M+1)]^T$$

$$\boldsymbol{R} = \begin{bmatrix} r(0) & r(1) & \dots & r(M-1) \\ r^*(1) & r(0) & \dots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r^*(M-1) & r^*(M-2) & \dots & r(0) \end{bmatrix}$$

$$\boldsymbol{p} = [p(0), p(-1), \dots, p(1-M)]^T$$

$$\boldsymbol{h_o} = [h_{o0}, h_{o1}, \dots, h_{oM-1}]^T$$

dove $\boldsymbol{h_o}$, è la successione ottima dei coefficienti del filtro e $\boldsymbol{p}$ il vettore di crosscorrelazione. Possiamo riscrivere, così, le equazioni del principio di ortogonalità in forma matriciale (*equazione di Wiener-Hopf*):

$$\boldsymbol{h_o} = \boldsymbol{R^{-1}p}$$

4.3 Superficie di Errore

Siamo interessati a studiare la superficie di errore, cioè il valore della funzione costo J al variare dei pesi $\boldsymbol{h}$:

$$\begin{aligned} J(\boldsymbol{h}) &= E[e(n)e^*(n)] \\ &= E[|d(n)|^2] - \sum_{k=0}^{M-1}\{h_k^*\, E[x(n-k)d^*(n)] - h_k E[x^*(n-k)d(n)]\} + \\ &+ \sum_{k=0}^{M-1}\sum_{i=0}^{M-1}\{h_k^*\, h_i\, E[x(n-k)x^*(n-1)]\} = \sigma_d^2 - \boldsymbol{h}^H\boldsymbol{p} - \boldsymbol{p}^H\boldsymbol{h} + \boldsymbol{h}^H\boldsymbol{R}\boldsymbol{h} \end{aligned}$$

Definiamo $J_{min} = \sigma_d^2 - \boldsymbol{h}_o^H\boldsymbol{R}\boldsymbol{h}_o$ il valore minimo assunto dalla funzione $J(\boldsymbol{h})$ per $\boldsymbol{h_o}$, la quale può essere riscritta nel modo seguente:

$$J(\boldsymbol{h}) = J_{min} + (\boldsymbol{h} - \boldsymbol{h}_o)^H\boldsymbol{R}(\boldsymbol{h} - \boldsymbol{h}_o)$$

Infine, utilizzando un cambio di base si può esprimerla così:

$$J = J_{min} + \sum_{k=1}^{M}\lambda_k v_k v_k^*$$

4.4 Metodo del Gradiente (Steepest Descent)

Si vuole cercare una soluzione all'equazione numerica di Wiener-Hopf: $\boldsymbol{h_o} = \boldsymbol{R}^{-1}\boldsymbol{p}$.
Una soluzione deterministica al problema può essere data dal metodo *Steepest Descent*.
Il metodo usa un gradiente deterministico per avvicinarsi a $\boldsymbol{h_o}$, consiste nella creazione di una successione di vettori $\boldsymbol{h}(n)$ che tendono alla successione ottima tale che $J(\boldsymbol{h_o}) \leq J(\boldsymbol{h})$.
La successione $\{\boldsymbol{h}(n)\}$ dell'algoritmo è generata nel modo seguente:

$$\boldsymbol{h}(n+1) = \boldsymbol{h}(n) - \frac{1}{2}\mu\boldsymbol{g}(n)$$

dove $\boldsymbol{g}(n) = \nabla J(\boldsymbol{h})$ e μ è chiamato *step-size parameter* ed è utilizzato per controllare il passo di avanzamento del metodo nonché la velocità di convergenza dello stesso.

4.5 LMS (Least Mean Square)

Il *Least-Mean-Square* (*LMS*) è un algoritmo stocastico per la soluzione del problema di Wiener in modo adattativo: i valori dei pesi $\widehat{h_l}(n)$, ottenuti dall'*LMS*, rappresentano una stima che si avvicina per $n \to \infty$ alla soluzione ottima di Wiener (per un ambiente stazionario in senso lato).
La principale differenza rispetto il metodo *Steepest Descent* consiste nel fatto che il calcolo del gradiente viene realizzato svincolandosi dal concetto di speranza matematica.
Non sono note a priori né la matrice di autocorrelazione $\boldsymbol{R}$ né il vettore di crosscorrelazione $\boldsymbol{p}$, pertanto il gradiente $\nabla J(n) = -2\boldsymbol{p} + 2\boldsymbol{R}\boldsymbol{h}(n)$ viene stimato dai dati disponibili.
Si approssimano quindi i valori di $\boldsymbol{R}$ e $\boldsymbol{p}$ come segue:

$$\widehat{\boldsymbol{R}}(n) = \boldsymbol{x}(n)\,\boldsymbol{x}^H(n)$$
$$\widehat{\boldsymbol{p}}(n) = \boldsymbol{x}(n)\,d^*(n)$$

Riscriviamo l'approssimazione del gradiente e la nuova relazione ricorsiva:

$$\widehat{\nabla}J(n) \quad = -2\boldsymbol{x}(n)d^*(n) + 2\boldsymbol{x}(n)\boldsymbol{x}^H(n)\widehat{\boldsymbol{h}}(n)$$

$$\widehat{\boldsymbol{h}}(n+1) = \widehat{\boldsymbol{h}}(n) + \mu\boldsymbol{x}(n)\left[d^*(n) - \boldsymbol{x}^H(n)\widehat{\boldsymbol{h}}(n)\right]$$

L'algoritmo *LMS* può quindi venire riassunto come segue:

$$y(n) = \widehat{\boldsymbol{h}}^{\boldsymbol{H}}(\boldsymbol{n})\boldsymbol{x}(n)$$

$$e(n) = d(n) - y(n)$$

$$\widehat{\boldsymbol{h}}(n+1) = \widehat{\boldsymbol{h}}(n) + \mu\boldsymbol{x}(n)e^*(n)$$

Per quanto riguarda la complessità computazionale stiamo parlando di $2M+1$ moltiplicazioni e $2M$ addizioni per ogni iterazione.

La condizione di stabilità del filtro è legata allo *step-size parameter* μ vediamone il comportamento teorico.

4.5.1 Comportamento Teorico dell'Algoritmo LMS

Il comportamento teorico dell'algoritmo *LMS* dipende in sostanza dagli autovalori della matrice di autocorrelazione dei campioni di ingresso $\boldsymbol{R}$. Indichiamo gli autovalori in ordine crescente con $\lambda_1, \dots, \lambda_M$. Si può dimostrare che, l'errore medio di un generico coefficiente $E[\boldsymbol{h}_m(n) - \boldsymbol{h}_m]$ con $m = 0, 1, \dots, \mathrm{M}-1$, è dato dalla somma di contributi esponenziali del tipo $(1-\mu\lambda_i)^n$. Dunque il valor medio dell'errore converge a zero se:

$$|1 - \mu\lambda_i| < 1$$

o, equivalentemente:

$$0 < \mu < \frac{2}{\lambda_M}$$

Il valore di μ che massimizza la velocità di convergenza è:

$$\mu_{opt} = \frac{2}{\lambda_1 + \lambda_M}$$

In tal caso:

$$\left|1 - \mu_{opt}\lambda_1\right| = \left|1 - \mu_{opt}\lambda_M\right| = \left|\frac{\lambda_M - \lambda_1}{\lambda_M + \lambda_1}\right|$$

La convergenza del valore medio dei coefficienti a zero non implica la convergenza dell'errore quadratico medio dell'uscita del filtro, il quale converge se e solo se:

$$0 < \mu < \frac{2}{\sum_{i=1}^{M}\lambda_i} = \frac{2}{traccia(\boldsymbol{R})}$$

La traccia di $\boldsymbol{R}$ è la somma degli elementi sulla diagonale principale che, poiché la matrice è hermitiana, sono tutti reali e gli autovalori reali e positivi:

$$traccia(\boldsymbol{R}) = \sum_{k=0}^{m-1} E[|r_+((nT - mT)f_s/f_k)|^2]$$

dove f_s = 1 / T è la *simbol rate* del segnale in ingresso e f_k la frequenza dei campioni, chiaramente il rapporto f_k/f_s deve essere maggiore o uguale a 2 per evitare problemi di aliasing.

Nonostante la derivazione dell'*LMS* sia basata su segnali stazionari in senso lato, l'algoritmo è applicabile anche ad ambienti deterministici in cui il vettore d'ingresso $\boldsymbol{x}(n)$ e la risposta $d(n)$ siano entrambi segnali deterministici e ad ambienti non stazionari in cui ci sia la necessità di inseguimento del comportamento del sistema (*tracking*).

Ai fini di descrivere le proprietà del sistema e stimarne il comportamento, si ricorre spesso all'impiego di grandezze specifiche, tra cui le medie delle curve di apprendimento:

1. *Excess Mean-Square Error* $J_{ex}(\infty)$ definito come la differenza tra $J(\infty)$ e il minimo valore ottenuto dalla soluzione ottima J_{min}:

$$J_{ex}(\infty) = J(\infty) - J_{min}$$

2. Misadjustment $\boldsymbol{\mathcal{M}}$ definito come il rapporto tra $J_{ex}(\infty)$ e J_{min}, questa grandezza è legata al valore di μ e influisce sulla retroazione come un filtro passabasso la cui costante di tempo è inversamente proporzionale a μ:

$$\boldsymbol{\mathcal{M}} = \frac{J_{ex}(\infty)}{J_{min}}$$

3. *Mean-Square Error MSE*:

$$J(n) = E[|e(n)|^2]$$

4. *Mean-Square Deviation MSD*:

$$\mathfrak{D}(n) = E[\|\epsilon(n)\|]$$

dove $\epsilon(n) = \boldsymbol{h}_o - \widehat{\boldsymbol{h}}(n)$

5. Average Eigenvalue λ_{av}, l'autovalore medio:

$$\lambda_{av} = \frac{1}{M}\sum_{k=1}^{M} \lambda_k$$

(nell'ipotesi che μ sia piccolissima si può esprimere il *misadjustment* $\boldsymbol{\mathcal{M}} = \mu M \lambda_{av}/2$)

6. Settling Time $\tau_{mse,av}$, la costante di decadimento media:

$$\tau_{mse,av} = \frac{1}{2\mu\lambda_{av}}$$

Dalla formula di aggiornamento dei coefficienti del filtro si può notare che l'algoritmo presenta diversi inconvenienti: innanzitutto la necessità di dimensionare il *step-size parameter* μ in funzione del sistema che si sta analizzando, cosa non immediata in quanto da un lato agisce sulla velocità e rapidità di convergenza dell'algoritmo, quindi alla dinamica del sistema e dall'altra è legato alla stabilità del filtro. Poi, la correzione del peso dei coefficienti è direttamente proporzionale al vettore $\boldsymbol{x}(n)$ perciò tanto più è grande, tanto più l'*LMS* risente dell'amplificazione del rumore nel calcolo del gradiente.
Per far fronte a queste difficoltà, si è sviluppato un algoritmo più evoluto:
l'*NLMS (Normalized LMS)*.

4.6 NLMS (Normalized Least Mean Square)

Questo algoritmo è basato sul concetto di "minimo disturbo": cioè da un'iterazione all'altra i pesi dei coefficienti devono variare il meno possibile, sotto la condizione imposta dall'uscita del filtro; vediamo in termini rigorosi le differenze rispetto all'algoritmo *LMS*.
Possiamo scrivere che:

$$\delta\widehat{\boldsymbol{h}}(n+1) = \widehat{\boldsymbol{h}}(n+1) - \widehat{\boldsymbol{h}}(n)$$

$$\widehat{\boldsymbol{h}}^{\boldsymbol{H}}(n+1)\,\boldsymbol{x}(n) = d(n)$$

Applichiamo adesso il metodo di Lagrange alla funzione costo:

$$J(n) = \left\|\delta\widehat{\boldsymbol{h}}(n+1)\right\|^2 + \Re\left[\lambda^*\left(d(n) - \widehat{\boldsymbol{h}}^{\boldsymbol{H}}(n+1)\,\boldsymbol{x}(n)\right)\right] =$$

$$= \left(\widehat{\boldsymbol{h}}(n+1) - \widehat{\boldsymbol{h}}(n)\right)^{\boldsymbol{H}}\left(\widehat{\boldsymbol{h}}(n+1) - \widehat{\boldsymbol{h}}(n)\right) + \Re\left[\lambda^*\left(d(n) - \widehat{\boldsymbol{h}}^{\boldsymbol{H}}(n+1)\,\boldsymbol{x}(n)\right)\right]$$

Deriviamo e uguagliamo a 0 il risultato:

$$\frac{\partial J(n)}{\partial\widehat{\boldsymbol{h}}(n+1)} = 2\left(\widehat{\boldsymbol{h}}(n+1) - \widehat{\boldsymbol{h}}(n)\right) - \lambda^*\boldsymbol{x}(n) = 0$$

$$\widehat{\boldsymbol{h}}(n+1) = \widehat{\boldsymbol{h}}(n) + \frac{1}{2}\lambda^*\boldsymbol{x}(n)$$

Sostituiamo questa considerazione all'interno dell'espressione di $d(n) = \widehat{\boldsymbol{h}}^{\boldsymbol{H}}(n+1)\,\boldsymbol{x}(n)$ e otteniamo:

$$d(n) = \widehat{\boldsymbol{h}}^{\boldsymbol{H}}(n)\,\boldsymbol{x}(n) + \frac{1}{2}\lambda\,\|\boldsymbol{x}(n)\|^{\boldsymbol{2}}$$

$$\lambda = \frac{2e(n)}{\|\boldsymbol{x}(n)\|^{\boldsymbol{2}}}$$

$$\widehat{\boldsymbol{h}}(n+1) = \widehat{\boldsymbol{h}}(n) + \frac{\tilde{\mu}}{\|\boldsymbol{x}(n)\|^2 + \delta}\boldsymbol{x}(n)e^*(n)$$

Il fattore di scala $\tilde{\mu}$ è una costante di adattamento adimensionale, mentre la costante μ nell'algoritmo LMS corrisponde dimensionalmente all'inverso della potenza.

Imponendo quindi:

$$\mu(n) = \frac{\tilde{\mu}}{\|\boldsymbol{x}(n)\|^2}$$

si può pensare all'$NLMS$ come a una variante dell'LMS dove il passo è tempo invariante.
Si evince che l'$NLMS$ possiede una velocità di convergenza potenzialmente più elevata rispetto a quella dell'LMS sia in presenza di ingressi correlati che non correlati.
Nelle applicazioni dove è richiesta una lunghezza notevole del filtro da adattare per far fronte alla risposta impulsiva coinvolta, corrispondono richieste di memoria considerevoli, l'adattamento in frequenza può fornire, in questi casi, un valido aiuto per far fronte al problema.
Sfruttando infatti le proprietà di ortogonalizzazione della trasformata di Fourier e di altre trasformate è possibile aumentare la velocità di convergenza dei convenzionali metodi *LMS* impiegando tecniche come la *BLMS* (*Block adaptive LMS*) e la *FBLMS* (*Fast Block adaptive LMS*) ma la trattazione rigorosa di tali metodi esula dagli obiettivi prefissati da questo lavoro.
Tuttavia esiste una variante del tradizionale *LMS* che sfrutta una tecnica denominata *self-orthogonalizing*, finalizzata a incrementare notevolmente la velocità di convergenza dell'algoritmo, svincolandola dalla statistica degli ingressi, l'equazione di aggiornamento dei pesi è:

$$\widehat{\boldsymbol{h}}(n+1) = \widehat{\boldsymbol{h}}(n) + \alpha \boldsymbol{R}^{-1}\boldsymbol{x}(n)e(n)$$

dove $\boldsymbol{R}^{-1}$ è l'inversa della matrice di autocorrelazione $\boldsymbol{R}$ e $\alpha = \frac{1}{2\,M}$ è una costante $\in (0,1)$.

Capitolo 5

ADC, DAC e Implementazione su DSP

I segnali del mondo reale sono analogici, mentre un elaboratore digitale è in grado di memorizzare e trattare esclusivamente sequenze finite di bit. Per trattare con tecniche digitali i segnali analogici è allora necessario, in via preliminare, approssimare quest'ultimi con segnali digitali. I sistemi che trasformano un segnale analogico nel corrispondente digitale sono detti *convertitori analogico-digitali (ADC)*, mentre quelli che realizzano l'operazione inversa di trasformare un segnale digitale in un segnale analogico sono detti *convertitori digitali-analogici (DAC)*. I principi di base per la conversione sono quelli relativi alle operazione di campionamento e la quantizzazione. Campionare un segnale a tempo continuo significa rilevare le ampiezze del segnale su un insieme discreto di tempi: i “frames" catturati da una telecamera che inquadra una scena reale, ne costituiscono un esempio. Viene qui discusso il fondamentale teorema del campionamento: un segnale a banda limitata da W [Hz] può essere perfettamente ricostruito dai suoi campioni presi con una frequenza di almeno $2W$ $[Hz]$; viene inoltre sommariamente descritto il fenomeno dell'equivocazione (*aliasing*), che si verifica quando la frequenza di campionamento è inferiore a $2W$. La quantizzazione permette di trasformare un segnale a valori continui in un segnale a valori su un insieme finito. Questa operazione in generale introduce un errore irreversibile nel segnale quantizzato: dato il segnale quantizzato, non è in generale possibile ricostruire il segnale originale. È tuttavia possibile controllare come il segnale quantizzato risulti una buona approssimazione di quello analogico: un tipico indice di qualità è il *rapporto segnale-rumore SQNR*. I sistemi che realizzano l'operazione di quantizzazione sono chiamati quantizzatori: in particolare il quantizzatore uniforme a n bit, ha la caratteristica che ad ogni bit aggiunto al quantizzatore si ha un miglioramento di 6 dB nel rapporto segnale-rumore. Successivamente si motivano e si introducono modelli di quantizzatori non uniformi. Un segnale di durata limitata nel tempo, campionato e quantizzato, può essere trattato con tecniche digitali: i principali sistemi di conversione analogico-digitale (ADC) e digitale-analogico (DAC), costituiscono il “ponte" tra il mondo analogico e l'elaboratore digitale.

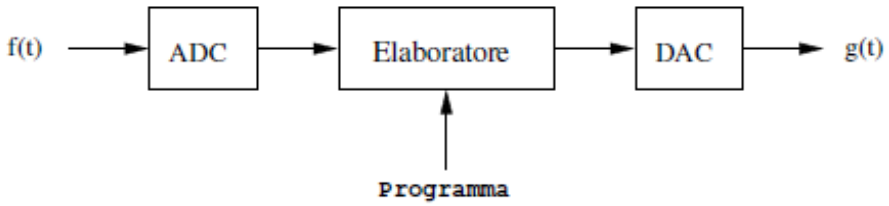


Figura 5.1: Elaborazione di segnali analogici

5.1 Campionamento

Campionare un segnale a tempo continuo significa rilevare le ampiezze del segnale su un insieme discreto di tempi. Ad esempio, fissato un intervallo di tempo di ampiezza τ, un campionamento uniforme con periodo τ di un segnale $f(t)$ corrisponde all'osservazione del segnale ai tempi $n\tau$ $(-\infty < n < \infty)$; il segnale campionato può essere interpretato come il segnale a tempo discreto $f(n\tau)$.
Il sistema *campionatore uniforme* a frequenza di campionamento $F_s = {}^1/_\tau$, trasforma quindi un segnale a tempo continuo $f(t)$ nel segnale a tempo discreto $f(n\tau)$, come mostrato in *Figura 5.2*.

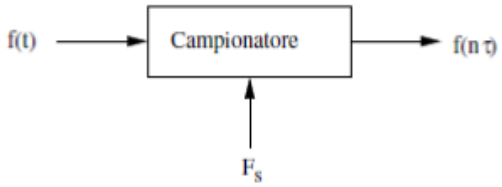


Figura 5.2: Campionatore

Si osservi che il sistema campionatore è un sistema lineare.
Per il teorema del campionamento: un segnale $f(t)$ è ricostruibile da $f(n\tau)$, se le componenti armoniche contenute nel segnale hanno frequenze inferiori a $F_s/2$, dove F_s è la frequenza di campionamento; quindi dato un segnale $f(t)$, è possibile stabilire con quale frequenza deve essere campionato affinché il segnale campionato $f(n\tau)(-\infty < n < \infty)$ contenga la stessa informazione di $f(t)$, cioè sia possibile ricostruire $f(t)$ a partire dalla sequenza $f(n\tau)$.

5.1.1 Teorema del Campionamento

Un segnale $f(t)$ a banda limitata da f_{max} $[Hz]$, la cui trasformata di Fourier $F(\omega)$ è quindi nulla per $|\omega| > 2\pi f_{max}$ $[rad/sec]$, può essere univocamente ricostruito dai suoi campioni $f(n\tau)(-\infty < n < \infty)$ presi a frequenza $F_s = {}^1/_\tau$, se $F_s \geq 2f_{max}$.
La frequenza $2f_{max}$ è detta tasso di Nyquist.

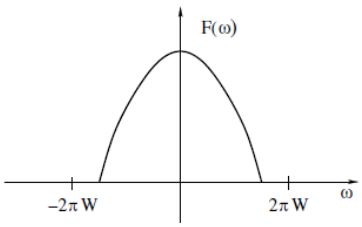


Figura 5.3: Esempio di un segnale a banda limitata

Supponendo di campionare un segnale a banda limitata B con una frequenza $W < 2B$, il metodo di ricostruzione garantito dal Teorema del campionamento non lavora più correttamente, a causa delle sovrapposizioni che si creano nella ripetizione periodica del segnale trasformato, come mostrato in *Figura 5.4*.

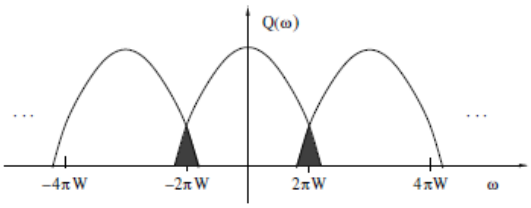


Figura 5.4: Manifestazione del fenomeno dell'aliasing

A causa di questo effetto, chiamato *equivocazione* o *aliasing*, non è in generale possibile ricostruire il segnale di partenza sulla base del segnale campionato. Per evitare questo fenomeno, un sistema campionatore a frequenza F_s viene normalmente fatto precedere da un filtro passa-basso con frequenza di taglio f_{max} al più $F_s/2$, detto filtro anti-aliasing, come mostrato in *Figura 5.5*.

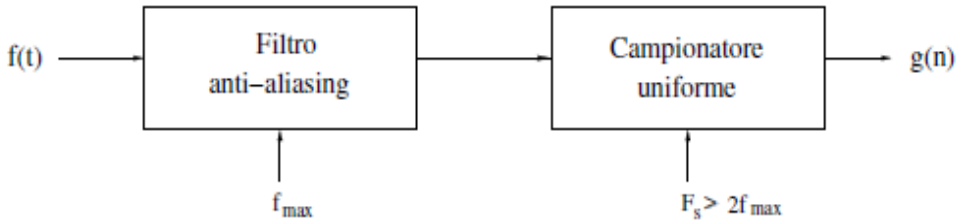


Figura 5.5: Sistema campionatore con filtro anti-aliasing

Per esempio: in telefonia, per esigenze del canale di comunicazione o della qualità dei segnali audio in gioco, la banda di frequenze di interesse è quella fonica, la cui frequenza più alta si aggira intorno i 3.4 [kHz].
Una conversazione può tuttavia contenere frequenze anche superiori ai 10 [kHz], è pertanto indispensabile premettere al campionatore, il quale può benissimo lavorare a una frequenza di campionamento canonica per sistemi di questa tipologia a 8 [kHz], un filtro passa-basso con una frequenza di taglio fissata a 3 [kHz].

5.2 Caratteristiche dei Filtri Anti-aliasing

Un filtro anti-aliasing dovrebbe rimuovere tutte le componenti del segnale a frequenze superiori o uguali alla metà della frequenza di campionamento: in linea di principio questo può essere ottenuto da un filtro ideale passa-basso con frequenza di taglio pari alla metà della frequenza di campionamento. Tali filtri non sono tuttavia realizzabili e possono solo essere approssimati con filtri “adeguati" alla particolare applicazione: discutiamo l'adeguatezza di un filtro anti-aliasing rispetto alle caratteristiche di un convertitore. Supponiamo che il convertitore sia composto da un campionatore a frequenza F_s e da un quantizzatore che approssima i valori del segnale utilizzando m bit; consideriamo inoltre filtri passa-basso caratterizzati dalla loro frequenza di taglio f_c a 3 [dB] e frequenza di stop f_s.
Il nostro obiettivo consiste nel determinare f_s, f_c e F_s in modo tale da garantire la corretta conversione di segnali con limite di banda f_{max}.
Le soluzioni possibili sono due:

1. Scegliere la frequenza di taglio f_c a 3 [dB] pari a f_{max}

2. Scegliere la frequenza di stop in modo tale che la massima oscillazione in banda proibita sia confrontabile con l'errore introdotto dalla quantizzazione o, equivalentemente, richiedendo che l'attenuazione $-g(f_s)$ sia uguale al rapporto segnale rumore SQNR, che per un quantizzatore a m bit vale $6m + 1{,}7$:

$$-20log_{10}|H(f_s)| = 6m + 1{,}7$$

5.3 Quantizzazione

La quantizzazione è il processo che permette di trasformare un segnale a valori continui in un segnale che assume un numero finito di valori. Un modo semplice di quantizzare consiste nel prefissare un insieme finito di l valori numerici $\{x_1, \dots, x_l\}$ e di associare ad ogni numero x il valore numerico x_k che è più vicino a x.
Se i segnali che prendiamo in considerazione hanno ampiezze comprese tra $-\frac{V}{2}$ e $\frac{V}{2}$, questo può essere ottenuto dividendo l'insieme $\left[-\frac{V}{2}, \frac{V}{2}\right]$ in l intervalli, detti livelli, ed attribuendo ad un punto $x \in \left[-\frac{V}{2}, \frac{V}{2}\right]$ il centro del livello in cui x cade. Detti $\{x_1, \dots, x_l\}$ i centri dei vari livelli, l'operazione di quantizzazione può essere allora descritta dalla funzione Q che ad ogni x associa il centro più vicino:

$$Q(x) = \underset{x_i \in \{x_1,\dots,x_l\}}{\arg\min} |x - x_i|$$

Il sistema che realizza l'operazione di quantizzazione è detto quantizzatore.
Poiché $\{x_1, \dots, x_l\}$ non è uno spazio vettoriale, il quantizzatore non è in generale un sistema lineare. Poiché inoltre la quantizzazione Q è una funzione molti-uno, essa introduce un errore irreversibile nel segnale quantizzato: dato il segnale quantizzato, non è possibile ricostruire in modo esatto il segnale d'origine.

5.3.1 Quantizzatore Uniforme e Rumore di Quantizzazione

Un sistema quantizzatore in cui l'intervallo $\left[-\frac{V}{2}, \frac{V}{2}\right]$ è suddiviso in l livelli di uguale ampiezza $\frac{V}{l}$ è detto *quantizzatore uniforme*; $\Delta = \frac{V}{l}$ è anche chiamato passo di quantizzazione.
Se $l = 2^m$, gli elementi $\{x_1, \dots, x_l\}$ possono essere codificati con parole di m bit:

$$x_i = b_{i1} \cdots b_{im}, \qquad b_{ik} \in \{0, 1\}\ (1 \le i \le l)$$

Il sistema in questo caso è detto quantizzatore uniforme a m bit.
La *Figura 5.6* mostra il risultato del campionamento (indicatore bianco) e campionamento più quantizzazione uniforme a quattro livelli (indicatore nero) di un segnale $f(t)$.

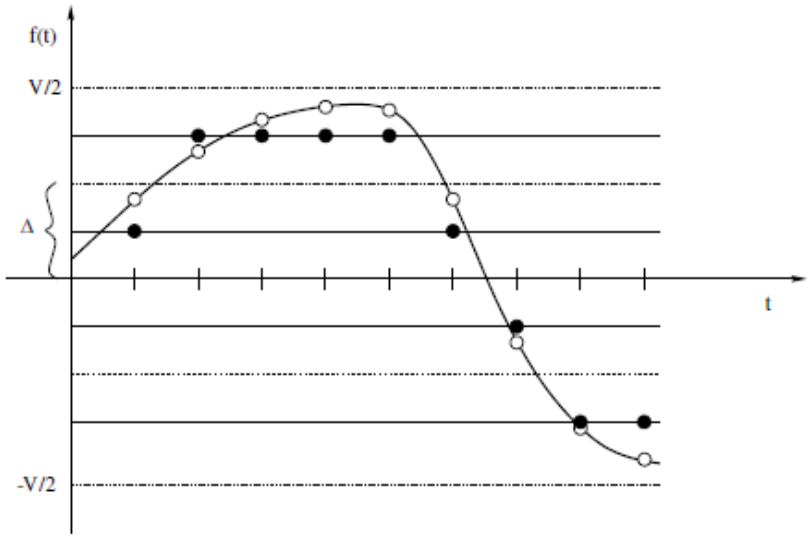


Figura 5.6: Campionamento più quantizzazione uniforme a quattro livelli di un segnale f(t)

Come ben evidenziato, la quantizzazione Q è una funzione molti-uno che introduce un errore irreversibile nel segnale quantizzato. Una naturale misura dell'errore sul numero x è la seguente:

$$e(x) = Q(x) - x$$

La *Figura 5.7* mostra l'errore di quantizzazione per un quantizzatore uniforme di due bit (quattro livelli).

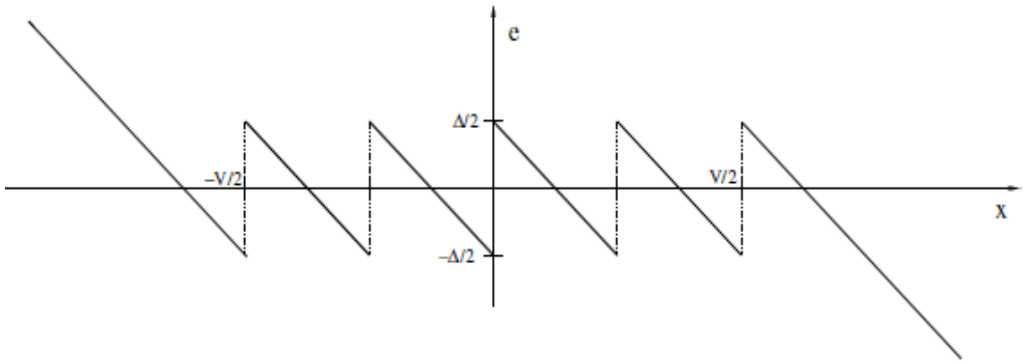


Figura 5.7: Errore di quantizzazione introdotto dal quantizzatore uniforme a quattro livelli

L'errore di quantizzazione ha un comportamento ben differenziato in due zone:

1. Se $x < -\frac{V}{2}$ oppure $x > \frac{V}{2}$, l'errore può essere arbitrariamente grande: in questo caso l'errore è detto *distorsione da overload* e lo si controlla cercando di garantire che i valori del segnale $f(t)$ in ingresso al quantizzatore rientrino nel range del quantizzatore, cioè che $-\frac{V}{2} \leq f(t) \leq \frac{V}{2}$.
2. Se x è invece interno all'intervallo $-\frac{V}{2} \leq x \leq \frac{V}{2}$, l'errore $e(x)$ si mantiene in valore assoluto minore o uguale a $\frac{\Delta}{2}$; tale errore è detto *rumore granulare*.

In seguito supporremo che l'unica sorgente di errore sia il rumore granulare.
Una misura di prestazione del quantizzatore è data dal rapporto segnale-rumore di quantizzazione SQNR (Signal Quantization to Noise Ratio), misurato in decibel [dB]:

$$SQNR = 10\, log_{10} \frac{\sigma^2}{\sigma_e^2}$$

dove σ^2 è la varianza del segnale e σ_e^2 l'errore di quantizzazione quadratico medio.
Osserviamo che nelle nostre ipotesi l'errore di quantizzazione è sempre limitato:

$$-\frac{\Delta}{2} \leq errore \leq \frac{\Delta}{2}$$

Per molti segnali deterministici inoltre l'errore è uniformemente distribuito in $\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right]$.
Questo significa che la probabilità che l'errore sia compreso fra e ed $e + de$ è $\frac{de}{\Delta}$. L'errore quadratico medio è allora:

$$\sigma_e^2 = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} e^2 \frac{de}{\Delta} = \frac{\Delta^2}{12}$$

Ipotizziamo che il segnale di riferimento sia $\frac{A}{2}\sin(t)$. La media di tale segnale è 0, poiché:

$$\lim_{T\to\infty} \frac{\int_{-T}^{T} \sin(t)\, dt}{2T} = 0$$

La varianza σ^2 di tale segnale è invece $\frac{A^2}{8}$.
Infatti:

$$\sigma^2 = \lim_{T\to\infty} \frac{\int_{-T}^{T} (\frac{A}{2}\sin(t) - 0)^2\, dt}{2T} = \frac{A^2}{8}$$

In tal caso:

$$SQNR = 10\log_{10}\frac{A^2/8}{\Delta^2/12} = 10\log_{10}\frac{V^2A^2}{\Delta^2V^2} + \log_{10}\frac{3}{2} = 20\log_{10} l\frac{A}{V} + 1{,}76$$

Per quantizzatori a m bit vale $l = 2^m$, quindi:

$$SQNR = 6{,}02m + 20\log_{10}\frac{A}{V} + 1{,}76$$

Pertanto, si ottiene che in un quantizzatore ogni bit aggiunto comporta un incremento di 6,02 [dB] al rapporto segnale rumore. Se inoltre il range dinamico A del segnale sfrutta tutto il range V del quantizzatore risulta $SQNR \approx 6{,}02m + 1{,}76$ [dB].

5.4 Convertitore Analogico-Digitale (ADC)

Applicando un campionatore a frequenza F_s e consecutivamente un quantizzatore a m bit, un segnale $f(t)$ osservato per un tempo T può essere trasformato in un vettore di TF_s componenti a m bit: esso può quindi essere memorizzato in forma digitale usando TF_sm bit ed eventualmente modificato.
Il sistema che realizza questa trasformazione è detto *Convertitore Analogico-Digitale (ADC)* e può essere descritto come in Figura 5.10.
Il filtro anti-aliasing in figura è un filtro passa-basso ed ha la funzione di porre in ingresso al campionatore un segnale a banda limitata la cui frequenza di Nyquist non superi la frequenza di campionamento.
Esistono essenzialmente due differenti tipologie di convertitori analogico-digitale.

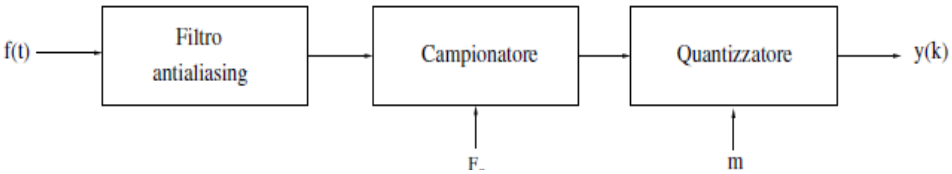


Figura 5.10: Convertitore Analogico-Digitale (ADC)

- Nel primo tipo il campionatore opera vicino alla frequenza di Nyquist del segnale e il quantizzatore è un quantizzatore ad m bit ($m \gg 1$)
- Nel secondo tipo si usa un campionatore a frequenza molto superiore al tasso di Nyquist (sovracampionamento), un quantizzatore a 1 bit e varie operazioni digitali.

Nei convertitori analogico-digitali del primo tipo, l'elemento critico è il quantizzatore. Fra i vari modelli disponibili, presentiamo il flash ADC (*Figura 5.11*) con il quantizzatore a 2 bit.

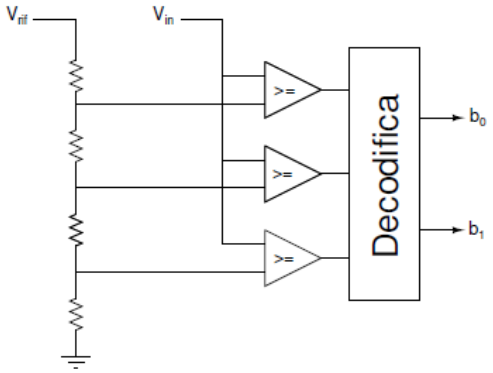


Figura 5.11: Flash ADC a 2 bit

Nel caso generale di un quantizzatore a m bit, la tensione del segnale di ingresso V_{im} viene confrontata con $2^m - 1$ tensioni di riferimento ottenute con un sistema di 2^m resistenze uguali poste in serie. Le uscite binarie dei comparatori vengono poi trasformate negli m bit di codifica da un opportuno circuito booleano di decodifica. Il Flash ADC è sicuramente il più veloce convertitore disponibile, ma è costituito da un numero elevato (2^m) di resistenze che devono essere molto accurate: questo rende difficile e costosa la realizzazione per alti valori di m ($m \gg 8$).

5.5 Convertitore Digitale-Analogico (DAC)

Il *Convertitore Digitale-Analogico* (DAC) trasforma un segnale digitale, a tempo e valori discreti, in un segnale analogico. Un modo semplice per convertire un segnale digitale $x(n)$ a frequenza F_s (i cui valori sono specificati da parole di m bit) è quello di trasformarlo nel segnale analogico $g(t)$, dove:

$$g(t) = x(n) \qquad n\tau \leq t < (n+1)\tau \qquad \tau = {}^{1}\!/_{F_s}$$

Questo tipo di convertitore è detto di tipo *ZOH (Zero-Order-Hold)*: la parola binaria al tempo $n\tau$ è convertita nel corrispettivo valore analogico, e tale valore viene mantenuto per tutto l'intervallo seguente di ampiezza τ. Il segnale ottenuto è descritto da una funzione a scala, che può essere "lisciata" applicando un opportuno filtro passa-basso, come mostrato in *Figura 5.16*.

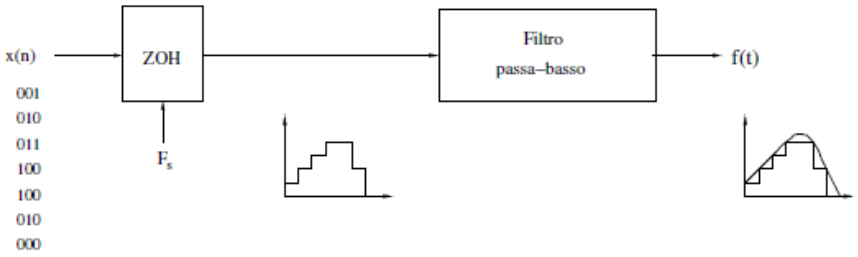


Figura 5.16: Convertitore Digitale-Analogico DAC

5.6 Analisi in Frequenza di un Convertitore ZOH

Un segnale digitale $x(n)$ può essere interpretato come segnale analogico $\sum_{n=-\infty}^{\infty} x(n)\delta(n\tau)$, in cui tutta l'energia del segnale è concentrata ai tempi discreti $n\tau$. Da questo punto di vista, il convertitore ZOH può essere visto come un sistema lineare tempo-invariante in cui la risposta all'impulso $\delta(t)$ è il rettangolo $\frac{1}{2} rect_{\frac{\tau}{2}}(t - \frac{\tau}{2})$ (*Figura 5.17*).

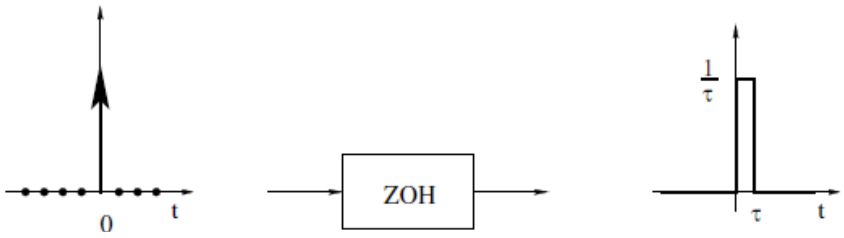


Figura 5.17: Risposta all'impulso di un DAC ZOH

La funzione di trasferimento $H(\omega)$ di questo sistema è dunque la trasformata di Fourier di $\frac{1}{2} rect_{\frac{\tau}{2}}(t - \frac{\tau}{2})$, cioè:

$$H(\omega) = e^{-i\frac{\tau}{2}\omega} 2 \frac{\sin(\frac{\tau}{2}\omega)}{\tau\omega}$$

Il modulo e la fase della funzione di trasferimento del DAC di tipo ZOH a frequenza $F_s = {}^{1}/_{\tau}$ risultano allora:

$$|H(\omega)| = \left| \frac{2\sin(\frac{\tau}{2}\omega)}{\tau\omega} \right|$$

$$\measuredangle H(\omega) = -\frac{\tau}{2}\omega$$

Osserviamo come prima cosa che la fase è lineare, con coefficiente angolare $-\frac{\tau}{2}$.
Rappresentiamo il grafico del modulo, limitato alla frequenza di Nyquist πF_s $[rad/s]$:

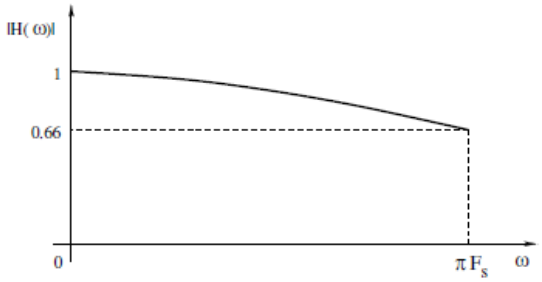


Figura 5.18:Modulo della funzione di trasferimento del DAC

Si osserva che il guadagno, per le componenti ad alta frequenza ($\omega \approx \pi F_s$) è significativamente minore che per quelle a bassa frequenza ($\omega \approx \pi 0$): questo fatto può provocare notevoli distorsioni nel segnale. Per ricostruire il segnale in modo fedele è utile mettere in sequenza al DAC un sistema lineare tempo invariante E con funzione di trasferimento $\frac{1}{H(\omega)}$, in modo che il sistema complessivo ($DAC + E$) risulti avere guadagno $G(\omega)$, con:

$$G(\omega) = \left| H(\omega) \frac{1}{H(\omega)} \right|^2 = 1$$

Il circuito che realizza E viene detto *equalizzatore* ed è caratterizzato da una funzione di trasferimento il cui modulo, *Figura 5.19,* è $\frac{\tau\omega}{2\,sin\frac{\tau}{2}}$.

Realizzare un equalizzatore, in questa applicazione, equivale quindi a determinare un sistema LTI il cui modulo, nella zona di lavoro, sia almeno approssimativamente $\frac{1}{|H(\omega)|}$.

Il sistema complessivo necessario ad elaborare digitalmente i segnali viene mostrato in *Figura 5.20*. Ricordiamo che se in una sequenza di sistemi LTI modifichiamo l'ordine dei sottosistemi, il risultato non cambia: in molte applicazioni risulta utile anteporre l'equalizzatore al DAC.

Nelle applicazioni risultano allora possibili due approcci.

1. Il filtro equalizzatore viene posto in uscita del DAC

2. Il filtro equalizzatore precondiziona il segnale prima che esso entri nel DAC.
   In tal caso l'equalizzatore sarà realizzato da un filtro digitale.

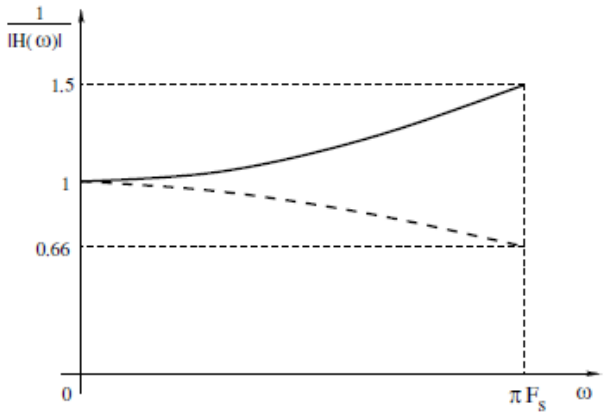


Figura 5.19:Modulo della funzione di trasferimento dell'equalizzatore

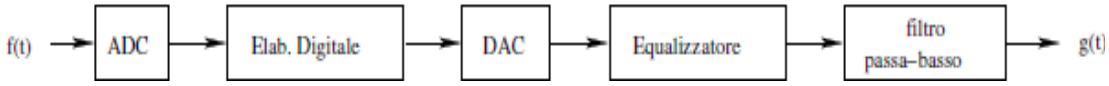


Figura 5.20:Elaborazione digitale di segnali

5.7 Architetture DSP

Un convertitore analogico-digitale (ADC) trasforma un segnale a tempo continuo in una sequenza di bit; viceversa un convertitore digitale-analogico (DAC) trasforma una sequenza di bit in un segnale a tempo continuo. Avendo a disposizione un elaboratore che abbia in input l'uscita di un ADC e dia la sua uscita in input a un DAC, si può pensare di emulare lato software a tutti gli effetti un generico sistema S, caratterizzato dalla sua relazione input-output. A tutti gli effetti pratici, il sistema S viene realizzato attraverso l'implementazione di un opportuno algoritmo, riducendo in linea di principio la sintesi di sistemi ad uno speciale problema di programmazione su architetture eventualmente specializzate: questa area di attività è detta *elaborazione numerica di segnali*.
Analizziamo, di seguito, i diversi approcci (software e hardware) all'elaborazione digitale dei segnali, discutendo vantaggi e svantaggi che l'elaborazione digitale ha rispetto a quella analogica. Un ampio spettro di applicazioni riguarda l'elaborazione in tempo reale: questo tipo di applicazioni richiede generalmente alta velocità di elaborazione, da cui la necessità di specializzare l'architettura degli elaboratori.

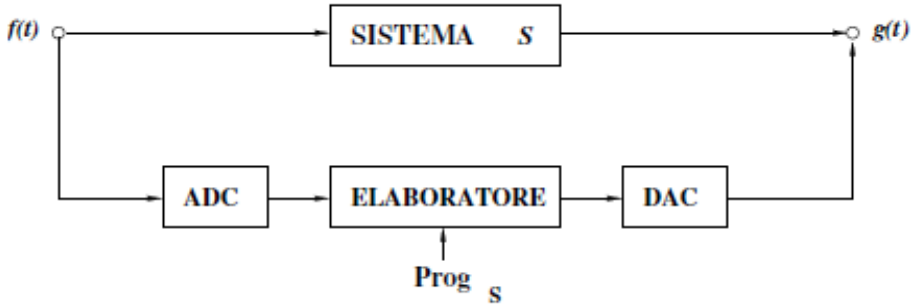


Figura 5.21: Elaborazione di segnali mediante conversione A/D e D/A

5.8 Elaborazione Digitale dei Segnali: Approcci

In generale, le tipologie fondamentali di approccio all'elaborazione digitale dei segnali sono mediante:

- *Programmazione*: in questo caso le caratteristiche di sistemi che trasformano segnali vengono simulate da programmi che vengono eseguiti da un elaboratore digitale. La velocità di esecuzione del programma dipende dal tipo di elaboratore ed in particolare dal livello di parallelismo e dalla velocità con cui vengono eseguite le istruzioni. Questo tipo di approccio risulta di grande flessibilità: sistemi totalmente diversi vengono simulati sulla stessa macchina, semplicemente eseguendo algoritmi diversi.

- *Realizzazione* h/w: in questo caso gli algoritmi che simulano i sistemi vengono realizzati in h/w con circuiti dedicati ad ad applicazioni specifiche (ASIC) o mediante circuiti programmabili (PLD). La velocità di elaborazione dipende dalla frequenza di clock e dalla programmazione dei ritardi nei circuiti logici. Questo approccio permette realizzazioni più efficienti delle soluzioni s/w, con velocità di elaborazione maggiori anche di un paio di ordini di grandezza, risultando tuttavia molto meno flessibile, in quanto la simulazione di un nuovo sistema costringe a riprogettare buona parte del circuito.

Entrambi gli approcci portano a soluzioni più precise, affidabili e stabili delle equivalenti realizzazioni analogiche. Inoltre, l'approccio mediante programmazione può ulteriormente essere agevolato fornendo l'elaboratore di un minimo di s/w di base, per esempio per l'auto-diagnosi e per la gestione delle periferiche. Infine, la progettazione di sistemi con tecniche digitali allarga la fascia dei potenziali progettisti, in quanto richiede meno competenze matematiche e fisiche che non la controparte analogica: grazie all'uso di strumenti s/w di ausilio alla programmazione, l'uso di questa tecnologia richiede solo una conoscenza di base dei principi di elaborazione dei segnali e qualche abilità di programmazione.
In conclusione, le soluzioni digitali offrono vari vantaggi rispetto a quelle analogiche. Per prima cosa, i circuiti analogici sono affetti dalla cosiddetta variabilità: lo stesso circuito ha comportamenti diversi in dipendenza dalla temperatura di lavoro o dal tempo di funzionamento, inoltre circuiti con lo stesso disegno hanno caratteristiche diverse a causa dell'intrinseca imprecisione delle componenti. Per contro, soluzioni digitali sono caratterizzate dalla loro ripetibilità: circuiti digitali correttamente disegnati produrranno gli stessi risultati in ogni tempo e per ragionevoli range di temperatura. L'approccio mediante programmazione porta inoltre a vantaggi di flessibilità: lo stesso h/w può supportare una infinita gamma di potenziali applicazioni. Va tuttavia segnalato che per particolari applicazioni le soluzioni analogiche hanno minor costo e maggior semplicità, oltre a non incorrere nel rumore di quantizzazione.

5.9 Architettura Von Neumann e Harvard

I microprocessori programmabili per l'elaborazione digitali dei segnali possono essere raggruppati nelle seguenti categorie:

Microprocessori general-purpose:
Questi microprocessori devono supportare le più disparate applicazioni, quindi la loro architettura viene progettata per l'ottimizzazione della gestione della memoria; le loro prestazioni nell'elaborazione digitale dei segnali risultano tuttavia mediocri.

Microcontrollori:
Questi strumenti implementano singole parti di un elaboratore, ad esempio apparecchi per l'input-output, memorie RAM e ROM; l'architettura è generalmente funzionale all'ottimizzazione delle caratteristiche input-output.

Processori specializzati all'elaborazione dei segnali (Digital Signal Processing DSP):
Questi microprocessori sono appositamente studiati per ottimizzare le prestazioni nell'elaborazione dei segnali. Poiché gli algoritmi per la simulazione di sistemi per segnali consistono spesso nella iterazione di sequenze di semplici operazioni aritmetiche, grande attenzione è posta nell'ottimizzazione dell'unità aritmetico-logica (ALU): i primi DSP sono addirittura stati motivati dalla necessità di accelerare l'esecuzione dell'operazione di moltiplicazione, rispetto agli usuali microprocessori. La necessità di grande velocità di elaborazione imposte dalle applicazioni in tempo-reale richiede inoltre l'introduzione di architetture che sfruttino l'inerente parallelismo di alcune funzionalità, pur sacrificando la semplicità realizzativi e la flessibilità rispetto alle applicazioni.

Per quanto detto, i DSP richiedono architetture di calcolo piuttosto differenti rispetto agli usuali microprocessori general-purpose: discutiamo brevemente qui le principali diversità.

Un elaboratore programmabile riceve in ingresso dati di due diversi tipi: istruzioni per il programma e dati veri e propri. La maggior parte dei microprocessori general-purpose è basata sull'architettura proposta da Von Neumann e realizzata per la prima volta nel 1951 (*Figura 5.22*), nella quale dati e programmi vengono memorizzati nella stessa area di memoria.

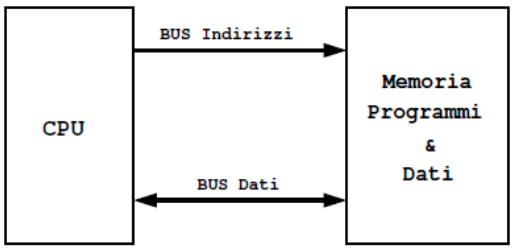


Figura 5.22: Architettura Von Neumann

Come si può osservare, esiste un'unica area di memoria per dati e programmi, ed il processore usa gli stessi bus dati e indirizzi per accedervi. Le unità basilari sono l'unità aritmetico-logica (ALU) e l'unità di input-output (IO). La ALU permette l'esecuzione di operazioni aritmetiche, mentre l'unità di input-output gestisce il flusso di dati esterni alla macchina. Per questo tipo di macchina, i programmi sono sequenze di istruzioni e la singola istruzione generalmente contiene un comando di operazione e l'indirizzo del dato su cui il comando deve essere eseguito,

Tipicamente l'esecuzione di una istruzione prevede tre passi:

1. Fetch (preleva l'istruzione)
2. Decode (decodifica)
3. Execute (esegui)

Poiché dati e programmi sono memorizzati nello stesso spazio di memoria, non è possibile prelevare (fetch) l'istruzione seguente mentre l'istruzione corrente è in esecuzione, poiché entrambe le operazioni prevedono un accesso a memoria: questo comporta un collo di bottiglia, che rallenta la velocità di esecuzione (*Figura 5.23*).

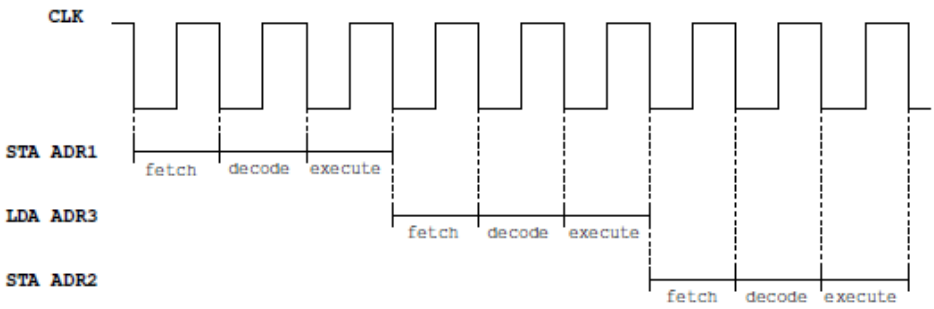


Figura 5.23: Esempio di esecuzione di tre istruzioni in un'architettura Von Neumann

Poiché la velocità di esecuzione è un elemento critico nell'elaborazione dei segnali, l'architettura Von Neumann non è adatta per questo tipi di applicazioni. Va per contro segnalato che la presenza di un'unica area per dati o programmi permette una buona flessibilità nell'uso della memoria: se l'applicazione cambia, il sistema può con facilità riallocare le risorse di memoria per permettere la nuova applicazione.
Un'architettura alternativa che permette un aumento di velocità, come richiesto dalle applicazioni concernenti l'elaborazione dei segnali, e l'architettura di Harvard: che prevede

due spazi di memoria, uno per i dati, l'altro per i programmi, e corrispondentemente diversi bus dati e bus indirizzi per accedervi (*Figura 5.24*).

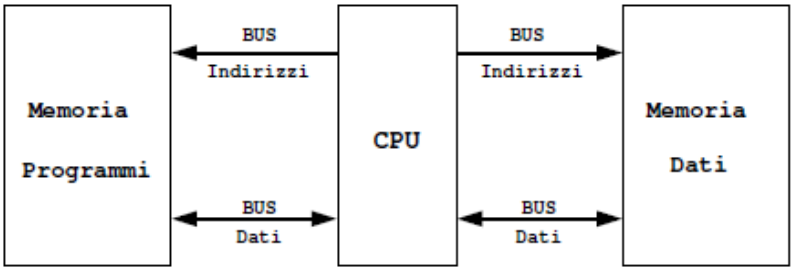


Figura 5.24: Architettura Harvard

Questa architettura permette di accedere contemporaneamente sia ai dati che alle istruzioni, ottenendo migliori prestazioni in velocità. Infatti la fase di fetch dell'istruzione seguente effettuabile in parallelo all'esecuzione dell'istruzione corrente, come mostrato in *Figura 5.25*.

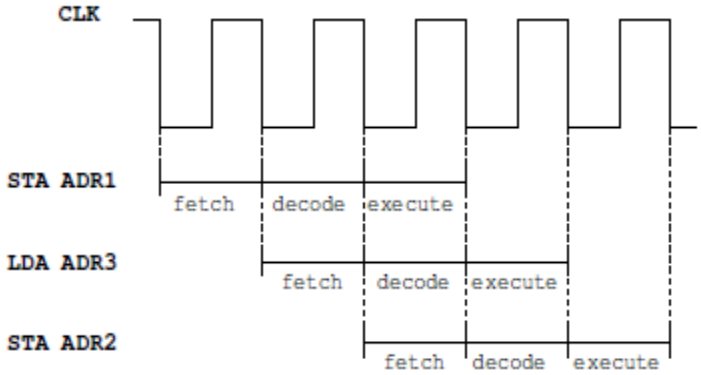


Figura 5.25: Esempio di esecuzione di tre istruzioni in un'architettura Harvard

L'architettura Harvard aumenta la velocità di esecuzione, ma introduce alcuni svantaggi:

- Le componenti h/w necessarie alla realizzazione di una architettura Harvard sono più complesse che per l'architettura Von Neumann, e questo comporta un considerevole aumento di prezzo.

- Le due distinte aree di memoria per dati e programmi tipiche dell'architettura Harvard non possono essere riallocate con la flessibilità usuale nell'architettura Von Neumann.

- Per ottimizzare le prestazioni, il processore deve prelevare ed eseguire le istruzioni in un unico ciclo macchina: questo pone vincoli e costi significativi.

- Scrivere programmi per una architettura Harvard è più complesso rispetto l'architettura Von Neumann.

Per aumentare flessibilità nell'allocazione di spazio di memoria, alcuni processori prevedono speciali blocchi di memoria che possono essere riconfigurati o come memoria per dati o come memoria per programmi. Questo porta a processori con una architettura di *Harvard modificata*: dove un singolo bus è usato esternamente per dati e indirizzi, mentre due o più bus separati sono usati internamente per dati e programmi.

In *Figura 5.26* si mostra il diagramma a blocchi del DSP TMS320C203 (prodotto dalla Texas Instruments) basato su un'architettura Harvard modificata.
Osserviamo infine che l'architettura Harvard può essere ulteriormente estesa, permettendo al processore di accedere a varie aree di memoria per dati, ognuna con un proprio bus dati e bus indirizzi.

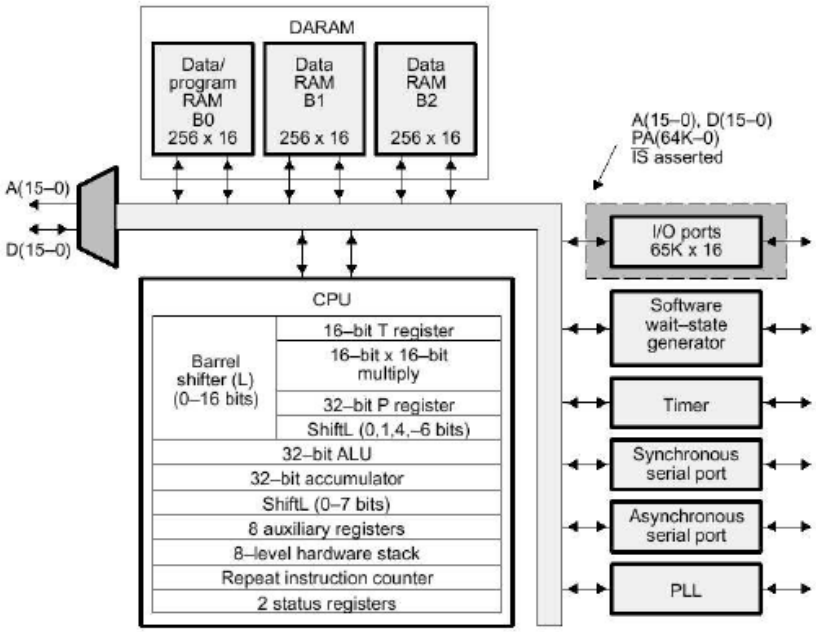


Figura 5.26: Diagramma a blocchi del DSP Texas Instruments TMS320C203

5.10 Istruzioni di Base di un DSP

Per ottenere alta velocità di esecuzione è necessario prelevare, decodificare ed eseguire ogni istruzione nel più breve tempo possibile. Gli approcci-limite tradizionali che vengono seguiti nella progettazione di un microprocessore a questo riguardo sono:

Approccio VLIW (Very Long Instruction Word).
Viene previsto un set di istruzioni di base molto ampio, in modo che risolvere un problema richieda programmi con poche istruzioni. Per contro, questa scelta richiede di codificare ogni istruzioni con molte parole nell'area programmi, così che prelevare, decodificare ed eseguire una istruzione richiedono molti cicli di clock.

Approccio RISC (Reduced Instruction Set).
Si sceglie di implementare in questo caso solo un ristretto set di istruzioni di base. In questo modo la scrittura di un programma richiede numerose istruzioni, ma ogni istruzione può essere codificata con una sola parola dell'area programmi. Conseguentemente, basta un ciclo di clock per prelevare ed eseguire una istruzione.

Generalmente, algoritmi per elaborazione dei segnali sono implementati con programmi dotati di un modesto numero di istruzioni e risulta invece critica la velocità di esecuzione della singola istruzione: per questa ragione, l'approccio RISC risulta effettivamente utile, anche se varianti dell'approccio VLIW possono accelerare la velocità di esecuzione mediante la gestione parallela di pacchetti di istruzioni.
La richiesta di poter prelevare ed eseguire una istruzione in un singolo ciclo di clock comporta che la lunghezza della parola che codifica l'istruzione non superi la dimensione del bus di programma. Tale parola contiene due parti: la codifica dell'operazione che deve

essere applicata (parte operazione) e l'indirizzo della locazione di memoria cui l'operazione va applicata (parte operando). I bit della parte operazione limitano il numero di istruzioni possibili; la parte operando può essere definita usando differenti modi di indirizzamento, ognuno dei quali consuma diversi bit nella parola che codifica l'istruzione. A causa dei limiti alla lunghezza di tale parola, non tutti i modi risultano disponibili nei processori specializzati per l'elaborazione dei segnali; vediamone alcuni.

Indirizzamento diretto: in questo caso l'operando specifica direttamente l'indirizzo della locazione di memoria cui applicare l'operazione. Questo modo richiede generalmente un elevato numero di bit nella parte operando dell'istruzione, spesso in conflitto col vincolo sulla lunghezza della parola. In questo caso, si utilizza una memoria detta registro di pagina, in cui vengono memorizzati i bit più significativi dell'indirizzo: la parte operando conterrà di conseguenza solo i bit meno significativi dell'indirizzo. L'indirizzamento diretto è quindi ottenuto combinando l'operando col registro di pagina.
Indirizzamento indiretto: in questo caso l'operando specifica quale registro-indirizzi dell'unità centrale punta alla locazione di memoria cui applicare l'operazione. La parte operando dell'indirizzamento indiretto usa generalmente un limitato numero di bit.
Indirizzamento indicizzato: in questo caso l'operando specifica un valore di riferimento v e un registro-indice dell'unità centrale. L'indirizzo della locazione di memoria cui applicare l'operazione si ottiene sommando v al contenuto del registro-indice. La parte operando dell'indirizzamento indicizzato usa generalmente un ragionevole numero di bit.
Indirizzamento circolare: algoritmi per l'elaborazione dei segnali fanno spesso uso di buffer first-in first-out (code FIFO). Per semplificare l'uso di tali strutture dati, molti DSP contengono un registro-puntatore che supporta il modo di indirizzamento *modulo*: il contenuto di tale registro è un intero modulo m, cioè resto della divisione di tale intero con m, in modo tale che il registro punti sempre a uno di m indirizzi consecutivi.
Altri modi di indirizzamento: nell'elaborazione dei segnali ci sono algoritmi di base che vengono richiamati frequentemente; può essere allora utile prevedere modi di indirizzamento orientati all'esecuzione efficiente di tali algoritmi. Senza entrare nel merito, segnaliamo ad esempio l'indirizzamento *bit-reversed* utile ad una implementazione efficiente della FFT.

5.11 Livello di integrazione delle Componenti di un DSP

La richiesta di eseguire ogni istruzione in un singolo ciclo di clock comporta la necessità di realizzare su un singolo chip di silicio il processore, l'area di memoria per i dati, l'area di memoria per i programmi e i rispettivi bus. Nei DSP vengono così integrate su un singolo chip molte delle componenti di un tradizionale computer. Questo pone severe limitazioni alla capacità di memoria dei DSP; fortunatamente, un gran numero di applicazioni richiede programmi con solo poche migliaia di istruzioni e dati per poche migliaia di parole.
Tra le apparecchiature fondamentali che rendono possibile l'elaborazione digitale vanno ricordati i convertitori ADC e DAC: interfacciare in modo efficiente il processore digitale con i convertitori ADC e DAC è di grande importanza per una efficiente elaborazione.
Sono proposte due soluzioni h/w:

1. integrazione dei convertitori ADC e DAC nello stesso chip del DSP;

2. presenza di h/w speciale per interfacciare efficientemente convertitori e DSP.

Nel primo caso, all'attuale livello di miniaturizzazione, i DSP non possono includere altre componenti, visto il consumo complessivo dell'area di silicio. Questo comporta attualmente forti limiti alle prestazioni di questi DSP.
Nel secondo caso, il processore include una speciale porta seriale per lo scambio di dati, con un numero veramente ridotto di pin di I/O. Questa soluzione permette di interfacciare DSP integrati su un unico chip con potenti convertitori a 24 bit.

5.12 H/W Specializzato: Moltiplicazione Veloce

La realizzazione circuitale h/w di una funzione permette tempi di calcolo un paio di ordini di grandezza inferiori della corrispondente realizzazione in software o firmware. Non è pertanto sorprendente che i DSP, nella perenne ricerca di velocità, contengano molte parti di h/w specializzato.
Vediamone alcuni importanti esempi.

Moltiplicazione h/w
Tipiche applicazioni dei DSP, come l'implementazione di un filtro, richiedono di eseguire un ordine di $10^6 \cdots 10^9$ moltiplicazioni nell'unità di tempo. La velocità di esecuzione della singola moltiplicazione è allora un parametro estremamente critico per la possibilità di elaborazione in tempo reale: soluzioni firmware, che calcolano la moltiplicazione eseguendo un programma sequenziale per la somma iterata, hanno velocità di esecuzione troppo bassa per molte applicazioni. Risulta allora importante sviluppare circuiti dedicati per la moltiplicazione, che ne permettano l'esecuzione in un singolo ciclo di clock.
In *Figura 5.27* viene presentato un tipico h/w dedicato *moltiplicatore-accumulatore* (MAC). In questa configurazione, il circuito moltiplicatore possiede una coppia di registri di input a 16 bit ed un registro di output a 32 bit.

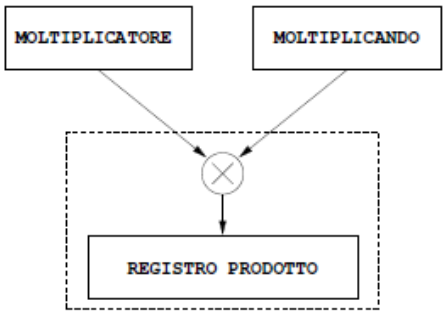


Figura 5.27: Moltiplicatore-Accumulatore (MAC)

Esistono essenzialmente due tipi di unità di moltiplicazione:

Unità per la moltiplicazione a virgola fissa (fixed point): sono caratterizzate dal rappresentare un numero in un intervallo fisato con precisione finita, in questo caso tutte le cifre del numero moltiplicato sono significative, e quindi questa unità ha eccellente velocità e precisione. Valori tipici per la rappresentazione ad interi sono 16, 24 e 32 bit.
L'intervallo di rappresentazione varia per un 16 bit da -32768 a 32767 (e per grandezze *unsigned* da 0 a 65535), per un 32 bit da 0 a $4 \cdot 10^9$.
In termini di precisione è opportuno definirne il concetto:

$$precisione = \frac{|\max valore\ rappresentato|}{|\max errore\ commesso\ in\ quantizzazione|}$$

Al numeratore c'è il massimo valore rappresentato (diverso da quello rappresentabile).
In virgola fissa la precisione cambia, perché non si riesce a sfruttare appieno la dinamica dei registri. Il rimedio per curare l'errore di precisione è quello di operare un rescaling: si fa coincidere il massimo valore rappresentato con il massimo valore rappresentabile.

Unità per la moltiplicazione a virgola mobile (floating point): la rappresentazione numerica mantissa-esponente permette di processare un grande range di numeri, tuttavia il circuito per la moltiplicazione in virgola mobile è più complesso e costoso. I vantaggi sono rappresentati da una maggiore precisione e semplicità del s/w che richiede effettivamente meno operazioni e tempi di sviluppo. L'errore che si commette è proporzionale alla grandezza dei numeri. L'intervallo di rappresentazione varia per un 32 bit (24 di mantissa e 8 di esponente) da $1{,}2 \cdot 10^{-38}$ a $3{,}4 \cdot 10^{38}$.
In virgola mobile la precisione è costante, perché i gap fra due numeri consecutivi sono proporzionali alla grandezza dei numeri in questione. Questa caratteristica è dovuta al fatto che i valori rappresentati in virgola mobile non sono ugualmente spaziati fra gli estremi dell'intervallo di rappresentazione, come avviene invece per i fixed point.

Accumulatori speciali: gli accumulatori sono un elemento critico dell'Unità Centrale; in un DSP, in particolare, è spesso necessario utilizzare numeri complessi oppure, per evitare errori di arrotondamento, risulta opportuno in certe fasi del calcolo considerare un numero maggiore di bit significativi. Questo può essere ottenuto con particolari accumulatori, come gli accumulatori duali che permettono il trattamento di numeri complessi in singolo ciclo macchina, o accumulatori con extra-bit. Poiché inoltre i calcoli causano spesso overflow, è necessario gestire particolari flags che rilevano tale situazione.

Stack in hardware: lo stack è una struttura dati per la gestione delle procedure, dovendosi memorizzare gli indirizzi di ritorno delle subroutine e interrupt. L'architettura di molti DSP offre una soluzione h/w alla gestione dello stack, includendo una speciale area di memoria separata per tale gestione, cosa che accelera i tempi di chiamata. Lo svantaggio è legato al fatto che, a causa della piccola dimensione di questo spazio di memoria, c'è un severo limite al numero di procedure innestate.

5.13 Il Rumore di Quantizzazione

Abbiamo già visto che quando un segnale analogico, continuo e variabile con precisione infinita, viene convertito in digitale, in una serie discreta di misure, la precisione è limitata dal numero di bit disponibile per rappresentare tali misure con valori digitali, introducendo degli errori non lineari e dipendenti dal segnale. A tutto ciò si devono aggiungere gli errori dovuti alla precisione limitata nei calcoli aritmetici all'interno del processore.
Un modo di descrivere il problema è considerare un'aggiunta di rumore al segnale, attribuendo gli errori di quantizzazione alla presenza di random-noise.

5.14 Elaborazione Parallela: SIMD e Superscalari

Avendo come obiettivo miglioramenti significativi nelle prestazioni, l'attuale tendenza nelle architetture per DSP è di aumentare sia il numero di operazioni compiute per istruzione che il numero di istruzioni eseguite per ogni ciclo. Di conseguenza, le architetture dei nuovi DSP devono saper sfruttare in modo intelligente le tecniche di elaborazione parallela. Analizziamo qui brevemente due approcci: *architetture SIMD* e *architetture superscalari*.
Sistemi con architettura SIMD (Single Instruction Multiple Data) devono permettere l'esecuzione parallela della stessa operazione su differenti dati. Per ottenere questo obiettivo, il DSP deve contenere diversi bus dati e diverse unità di elaborazione: in questo modo in un ciclo la stessa istruzione può essere eseguita dalle varie unità di elaborazione su differenti dati.
Di seguito, (*Figura 5.28*) evidenziamo la presenza di due unità di elaborazione e di due bus dati nello stesso DSP.

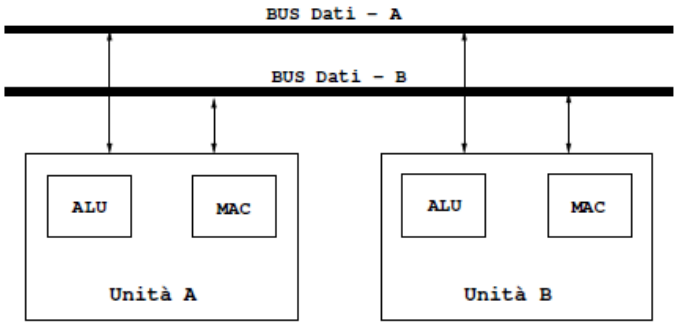


Figura 5.28: Architettura SIMD

L'elaborazione superscalare è una tecnica per aumentare la velocità di calcolo sfruttando invece il potenziale parallelismo a livello di istruzioni. È noto infatti che, in certe circostanze, alcune istruzioni possono essere eseguite in modo indipendente: avendo a disposizione un adeguato numero di unità di elaborazione, la loro esecuzione in parallelo sulle varie unità accelera il calcolo.
Per chiarire con un esempio, si consideri la sequenza di istruzioni:

$$\begin{aligned} & increm(x) \\ & y = x \end{aligned}$$

in questo caso le istruzioni non possono essere eseguite in modo indipendente, perché in particolare il risultato dipende dall'ordine di esecuzione. Se invece consideriamo la sequenza:

$$\begin{aligned} x &= x + 1 \\ y &= z \end{aligned}$$

le due istruzioni possono essere eseguite in maniera indipendente: attribuendo a due distinte unità di elaborazione le due istruzioni ed eseguendole in parallelo, si ottiene il risultato corretto diminuendo il tempo di esecuzione.
Questo è il principio generale: in una architettura superscalare, differenti unità concorrono all'avanzamento del calcolo eseguendo differenti istruzioni in parallelo. Va da sé che nella programmazione superscalare, a tempo di compilazione, deve essere effettuata in modo statico, una schedulazione delle istruzioni che eviti i problemi di conflitto creati dalle dipendenze nei dati e nel controllo.

5.15 Decimazione e Interpolazione

I moderni sistemi digitali possono processare ed elaborare i dati a più di una frequenza di campionamento. Le due operazioni base che permettono di modificare digitalmente le frequenze sono la *decimazione* e l'*interpolazione*: l'interpolazione aumenta la frequenza di campionamento, la decimazione la riduce, comprimendo di fatto i dati.
Naturalmente questi obiettivi devono essere ottenuti senza introdurre effetti indesiderati, come errori di quantizzazione o aliasing.
La decimazione $\downarrow M$ di un fattore M è ottenuta da un sistema con la seguente relazione ingresso-uscita:

$$M(x(n)) = x(Mn)$$

In *Figura 5.29* è rappresentato la relazione ingresso-uscita per la decimazione con $M = 3$:

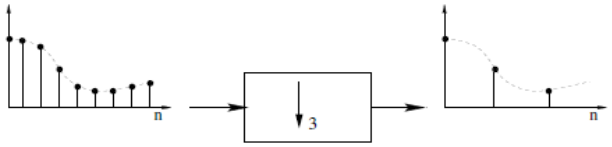


Figura 5.29: Operazione di decimazione

Tale sistema è chiaramente lineare, ma non tempo invariante: vogliamo qui studiarne la risposta in frequenza. A tal riguardo, consideriamo $X(z)$ e $Y(z)$ le trasformate zeta dell'ingresso $x(n)$ e dell'uscita $y(n) = x(Mn)$ e $F(\omega) = X(e^{i\omega})$, $G(\omega) = Y(e^{i\omega})$, le trasformate di Fourier, 2π-periodiche, di $x(n)$ e $y(n)$.
Se $F(\omega)$ e $G(\omega)$ sono lo spettro di frequenza di segnali rispettivamente in ingresso e uscita all'operazione di decimazione con fattore M, vale:

$$G(\omega) = \frac{1}{M}(F(\omega) + F\left(\omega - \frac{2\pi}{M}\right) + \cdots + F(\omega - \frac{2(M-1)\pi}{M})$$

Osserviamo che la funzione $G(\omega)$ è periodica di periodo $2\pi M$: se F_s è la frequenza di campionamento del segnale di ingresso, la frequenza del segnale di uscita risulta $\frac{F_s}{M}$.
Osserviamo inoltre che il segnale di ingresso può essere perfettamente ricostruito dal segnale di uscita se si evita il fenomeno dell'aliasing, e cioè se il limite di banda del segnale di ingresso è $\frac{\pi}{M}$. Se quindi desideriamo che l'operazione di decimazione non crei perdita di informazione rispetto al segnale di ingresso, tale operazione dovrà essere preceduta da un filtro passa-basso, con frequenza di taglio $\frac{\pi}{M}$. In *Figura 5.30* viene mostrato il sistema Decimatore, composto da un filtro anti-aliasing e dall'operazione di decimazione.

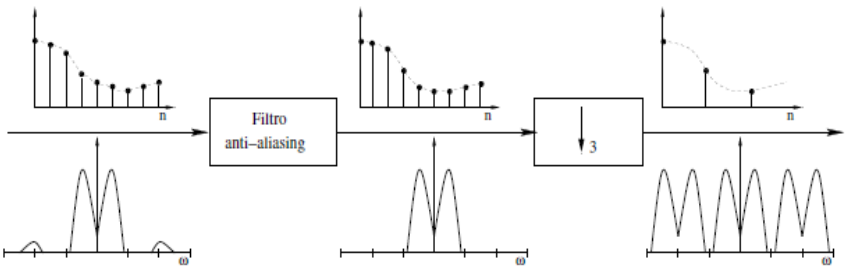


Figura 5.30: Sistema Decimatore

L'interpolazione di segnali digitali ha forti similitudini con la conversione digitale-analogica: in questo caso, tuttavia, il segnale di uscita continua ad essere digitale, anche se "campionato" a frequenza più alta. Più precisamente, un interpolatore con fattore L trasforma un segnale campionato con frequenza F_s in uno campionato con frequenza LF_s.
In *Figura 5.31* è rappresentato un sistema Interpolatore (con fattore 3): costituito da un'operazione di interpolazione $\uparrow L$ che inserisce, tra $x(n)$ e $x(n+1)$, $L-1$ campioni a valore 0. Il nuovo segnale viene poi filtrato con un filtro digitale passabasso a frequenza di taglio $\frac{F_s}{2L}$, dove F_s è la frequenza di campionamento di $x(n)$. Poiché l'inserzione di $L-1$ zeri "distribuisce" l'energia di un campione su L campioni, l'uscita $y(n)$ risulta attenuata di un fattore $\frac{1}{L}$: questo fatto può essere compensato moltiplicando per L il valore dell'uscita.
Come è ben evidenziato dalla rappresentazione in frequenza, il Decimatore e l'Interpolatore rappresentano sistemi l'uno inverso dell'altro.

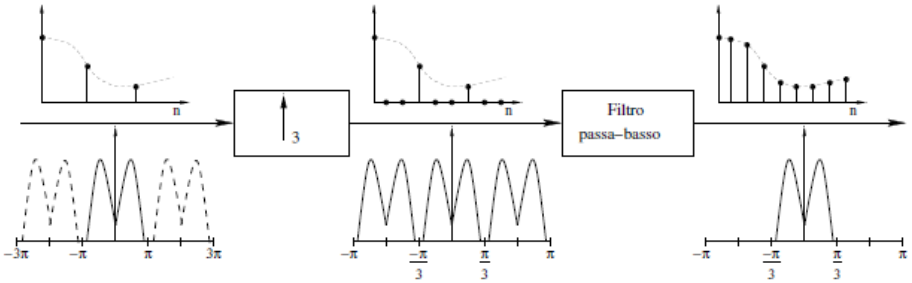


Figura 5.31: Sistema Interpolatore

Capitolo 6

Sistema Analizzato e Programmazione dell'Anti-Larsen

I sistemi di comunicazione presi in considerazione, soggetti al fenomeno Larsen, appartengono al vasto mondo della domotica. Nella realizzazione di impianti finalizzati all'automazione degli edifici vengono impiegati numerosi dispositivi atti a mettere in comunicazione audio e video, zone o aree differenti all'interno dello stesso edificio.
Questi dispositivi sono per esempio videocitofoni oppure schermi touchscreen da parete nei quali sono integrati opportuni microfoni e speaker gestiti via software. Nel dettaglio, gli apparati analizzati sono connessi in rete tramite ethernet e le informazioni audio vengono trasmesse mediante pacchetti PCM codificati da un codec audio che dispone anche di un controller per touchscreen l'UCB1400BE della Philips.
Vediamo in *Figura 6.1* uno schema del sistema analizzato dove nel dettaglio si può notare che i touchscreen e i videocitofoni si interfacciano tramite Ethernet (linea rossa):

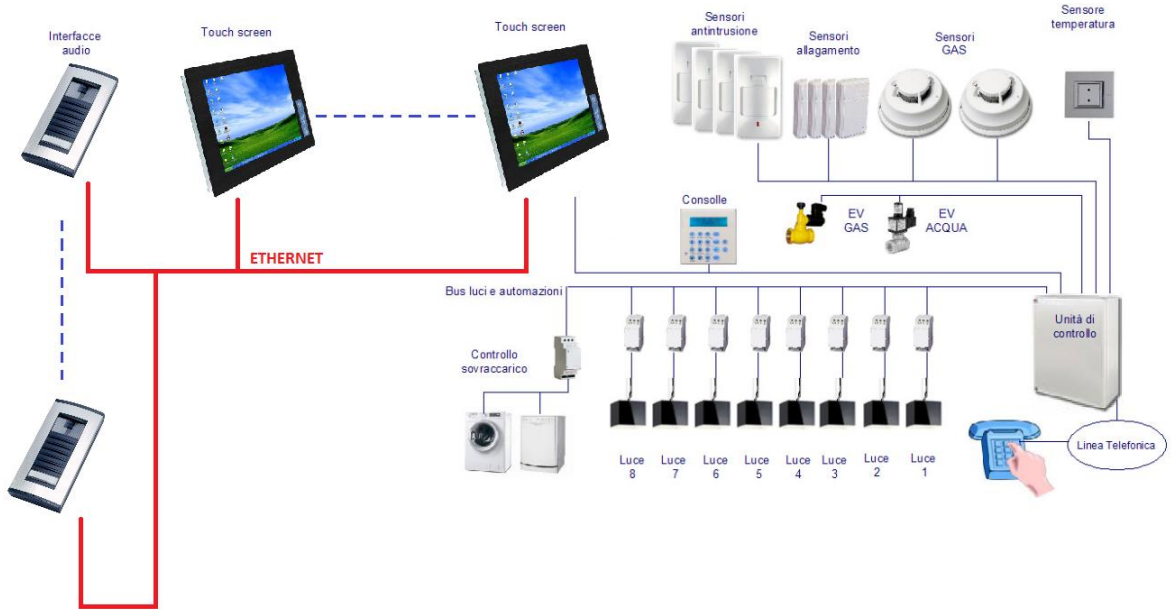


Figura 6.1: Schema di un sistema domotico

Su ciascun dispositivo, che sia touchscreen o videocitofono, sia il microfono che lo speaker vengono installati seguendo accorgimenti e criteri atti a sfavorire l'innesco dell'effetto Larsen; infatti l'installazione è realizzata su lati opposti del dispositivo, in modo da sfruttare la massima distanza tra i due.
L'orientamento opportuno sia dello speaker sia del microfono riduce la probabilità che una parte della pressione sonora emessa dallo speaker venga captata dal microfono; inoltre si

impiegano materiali come la spugna in prossimità del microfono per limitarne la sensibilità alle vibrazioni del telaio sul quale è agganciato. Talvolta infine il microfono è incapsulato in una protezione di materiale plastico, la quale ha una piccola fessura che va ad agire sulla direttività del microfono.
Nonostante tutte queste strategie sul piano fisico e meccanico, l'installazione in ambienti chiusi che per loro intrinseca natura sono propensi alla propagazione dell'eco, fa si che s'inneschi ugualmente l'effetto Larsen.

6.1 Sistema Full-Duplex Analizzato

Tipicamente un sistema domotico permette la comunicazione fra molteplici dispositivi audio/video in modalità *conference*, per semplicità consideriamo la comunicazione esclusivamente attiva tra due dispositivi: il *Far-End* (*FE*) e il *Near-End (NE)*.
Come abbiamo già visto, il *Far-End* è la sorgente lontana mentre il *Near-End* è la sorgente vicina, rispettivamente a seconda del verso della comunicazione avremmo due percorsi dell'informazione audio tra le due sorgenti sonore quello di *Downlink* da *FE* a *NE* e quello di *Uplink* da *NE* a *FE*. (*Figura 6.2*).
Se è attivo solo uno dei due percorsi si parla di una comunicazione *Half-Duplex* invece se entrambi i percorsi sono aperti contemporaneamente si parla di una comunicazione *Full-Duplex*.
Per ipotesi, supponiamo che la coppia speaker/microfono lato *Far-End* sia priva di accoppiamento acustico e che quindi non contribuisca ai fini dell'eco acustico nella comunicazione.
Viceversa lato Near-End supponiamo un significativo contributo di eco acustico dovuto all'accoppiamento tra microfono e speaker presenti sul touchscreen.

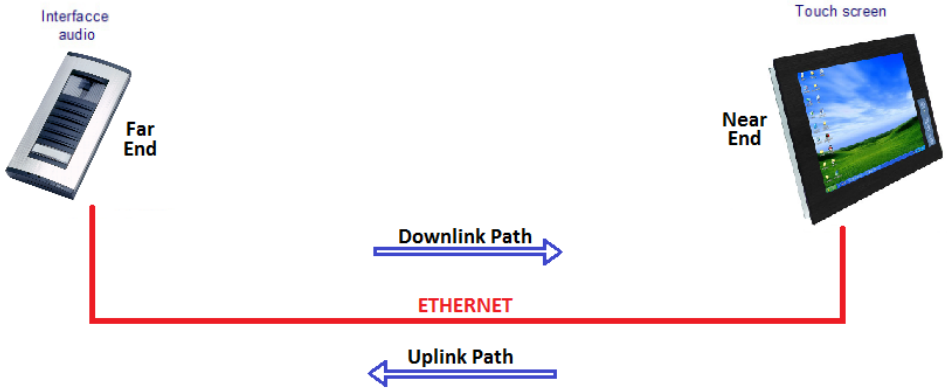


Figura 6.2: Sistema Full-duplex

6.2 Misure Realizzate

Ai fini di quantificare l'entità di accoppiamento acustico tra speaker e microfono lato *Near-End* (touchscreen) sono stati iniettati dei segnali campione con l'obiettivo di misurare e analizzare, i tempi di ritardo e i livelli di segnale presenti direttamente ai terminali dello speaker e del microfono del touchscreen.
Per realizzare tali misure sono state considerate condizioni operative dei dispositivi reali, pertanto è stata tenuta in considerazione un'opportuna ubicazione dei dispositivi, il fissaggio di questi alle pareti, l'applicazione dell'intelaiatura e della cornice. Il *Far-End* è stato isolato acusticamente in un ambiente silenzioso mentre il *Near-End* sottoposto a test, è stato studiato con svariate configurazioni ambientali e in diverse posizioni logistiche e per ciascuna di queste le misure realizzate sono state molteplici.

Sul touchscreen microfono e speaker sono disposti in posizioni opposte, verticalmente, rispetto lo schermo del touch, il quale è gestito da una board di controllo dedicata che è l'Orchid V.1.4 della Toradex.
Il microfono si interfaccia direttamente con questa board invece lo speaker viene pilotato da una scheda di controllo intermedia, proprietaria, la cui funzione, per quanto riguarda l'audio, consiste nell'amplificare il segnale proveniente dall'Orchid (*Far-End Signal*) per inviarlo allo speaker.
Il microfono è di tipo elettrostatico o a condensatore modello AMF-O97A38-NWH2-LF, da specifiche presenta:

- Sensitività di -38 $\pm$ $3dB$ ($0dB = \frac{1V}{Pa}$ @ $1kHz$) garantita dai $300Hz$ ai $3kHz$
- Banda passante dai $70Hz$ ai $20kHz$
- Impedenza $2{,}2k\Omega$
- Direttività Omnidirezionale
- Tensione operativa da $4{,}5V$ a $10V$
- Massima corrente assorbita $0{,}5\ mA$
- SNR Ratio $> 58dB$

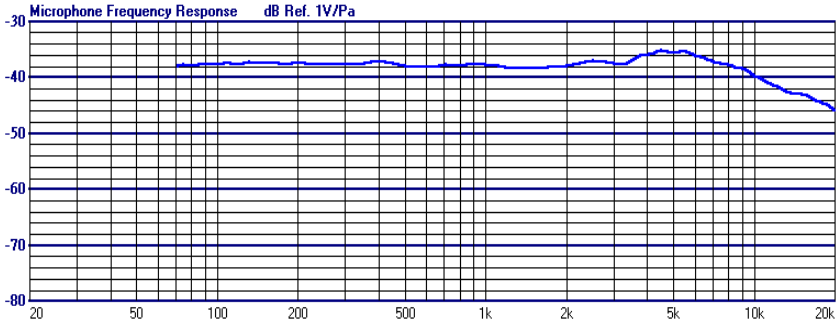


Figura 6.2: Risposta in frequenza del microfono

Dalle misure realizzate operativamente il dispositivo lavora con una *phantom supply* che varia dai $2{,}8V$ ai $3{,}3V$ inferiore rispetto le specifiche la quale comporta una riduzione della sensitività di diversi dB, segnali tipici hanno un'ampiezza massima di $400mV\ picco-picco$.
Lo speaker è di tipo dinamico modello AK-251508EC-1-LF, da specifiche presenta:

- Massima potenza $0{,}5W$
- Impedenza $8\Omega \pm 15\%$ @ $2kHz$
- Sensitività di $77dB\left(\frac{W}{m}\right) \pm 3dB, 90dB\left(\frac{0.25W}{0.1m}\right) \pm 3dB$) garantita da $1kHz$ a $2kHz$
- Banda passante dai $900Hz$ ai $15kHz$
- THD $< 10\%$ @$1kHz$, $1V$

Segnali tipici hanno un'ampiezza massima di $5V\ p-p$
Pilotato lo speaker con un segnale campione si è osservato in svariate condizioni ambientali il livello di segnale ricevuto al microfono e il tempo di ritardo di quest'ultimo.
Si è osservato sperimentalmente come w*orst-case* la seguente situazione:

Ritardo massimo pari a 14ms tra il segnale riprodotto dallo speaker e lo stesso segnale avvertito dal microfono, attenuato di circa $-21,67\ dB$.

Per completezza, ai fini della caratterizzazione del sistema, sono stati stimati i tempi di propagazione dei vari segnali fra le sorgenti di *Far-End* e *Near-End*, sia in *Uplink* che in *Downlink*: mediamente si è riscontrata una variabilità considerevole di queste tempistiche, tuttavia la media si assestava intorno ai *2* secondi.

La spiegazione di ritardi così considerevoli è dovuta a diversi fattori, innanzitutto l'audio processato dal touchscreen viene bufferizzato, per costruzione delle librerie che ne gestiscono il flusso audio/video, a 1 secondo, in secondo luogo la rete Ethernet introduce una latenza non costante legata al flusso e al traffico istantaneo variabile presente in rete. Infine c'è da considerare che il software del touch deve elaborare contemporaneamente al flusso audio/video altre richieste legate alla ordinaria gestione dell'impianto che introducono ritardi inevitabili all'elaborazione audio/video.

6.3 Simulazione del Sistema

Dai risultati sperimentali rilevati, si è costruito un modello matematico del sistema in Matlab, e si è simulata l'applicazione di un algoritmo $NLMS$ ai fini della cancellazione dell'eco. Riportiamo lo schema a blocchi dell'algoritmo di cancellazione dell'eco:

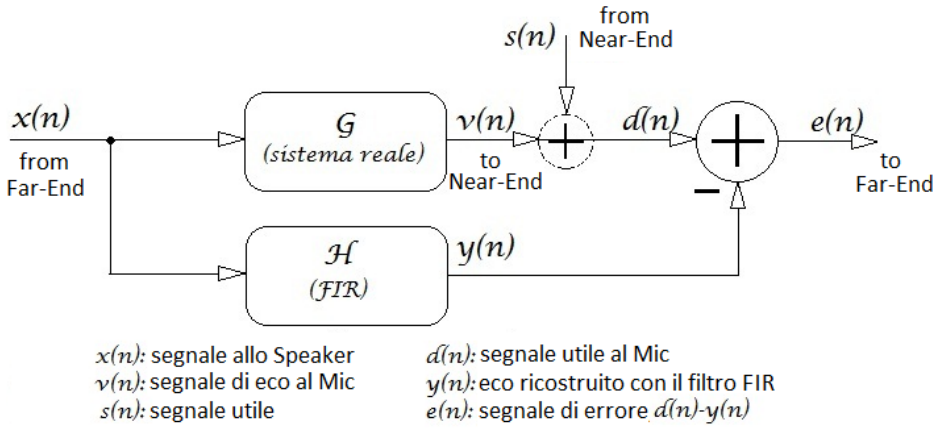


Figura 6.3: Schema a blocchi del sistema con il filtro FIR

In $Appendice\ A$ si riportano i programmi Matlab realizzati per la stesura dell'algoritmo $NLMS$, e per la simulazione del sistema, di seguito riportiamo i risultati grafici della simulazione.
La frequenza di campionamento f_k utilizzata nelle seguenti simulazioni in Matlab è di $24kHz$, in ogni caso era ampiamente sufficiente anche un *8kHz* perché considerata come banda di interesse quella fonica a $3{,}4kHz$, è sufficiente evitare aliasing campionando a una frequenza superiore ai $6{,}8kHz$.
Fissata f_k a $24\ kHz$ avremmo un campione ogni $41{,}6\ \mu s$, quindi per coprire il ritardo di $14\ ms$ del sistema significa che la lunghezza N del filtro FIR deve essere di almeno:

$$N = f_k \cdot 14\ ms = 24000 \cdot 0{,}014 = 336$$

È stato costruito un segnale campione generato da una funzione *randn* generante variabili casuali con densità di probabilità normale e si è fissata arbitrariamente la lunghezza del filtro $N = 450$ tappi.
Si evince dai risultati grafici che l'errore, come mi aspetto, diventi infinitesimo, in particolare è stato rappresentato l'MSE che si nota si assesta intorno ai $-130\ dB$.

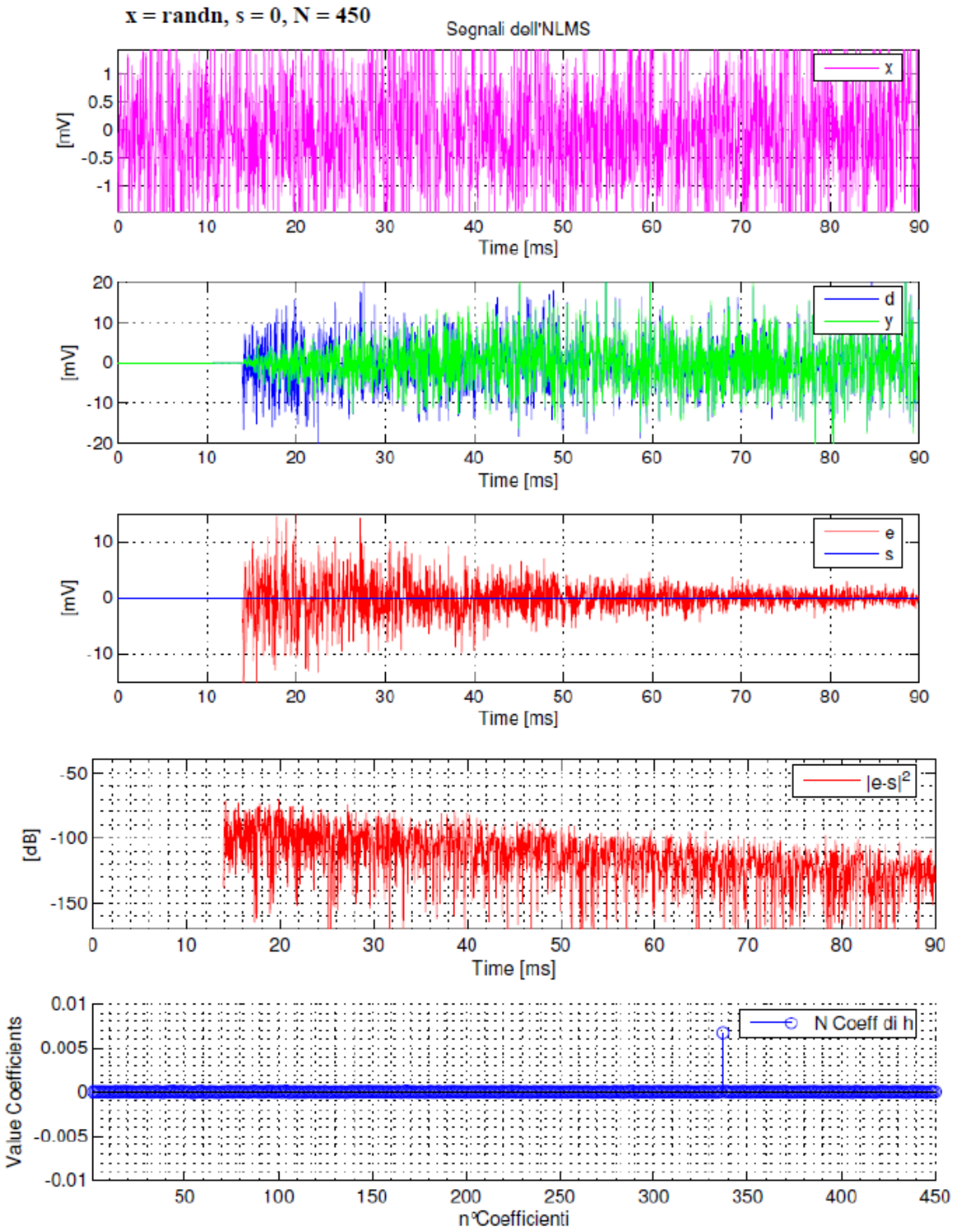


Figura 6.4: Segnali in simulazione dell'algoritmo NLMS

6.4 Board Evaluation Module TLV320AIC3254EVM-U T.I.

Il TLV320AIC3254 è un dispositivo programmabile tramite USB e basato su un codec audio stereo AIC3254 i cui ingressi e uscite sono interamente mappabili, lavora a basse potenze e tensioni, e dispone di 2 miniDSP completamente programmabili mediante delle routine software di signal processing precostruite e parametrizzabili, finalizzate ad applicazioni specifiche in ambito di registrazione/riproduzione audio, cancellazione di rumore, AEC e filtraggio digitale.
La connessione USB provvede sia all'alimentazione della scheda stessa, che alla sua programmazione, inoltre tra le feature disponibili è possibile indirizzarne eventuali flussi audio in modalità streaming.
Dispone di 2 software: uno di programmazione (*PurePath Studio*) che permette la gestione di ogni singolo dispositivo presente sulla board; e uno di controllo (*Control Software*) con il quale si possono mappare facilmente gli ingressi e le uscite e caricare delle feature pre-programmate, tuttavia questo rimane più limitato a livello di funzionalità rispetto al precedente.

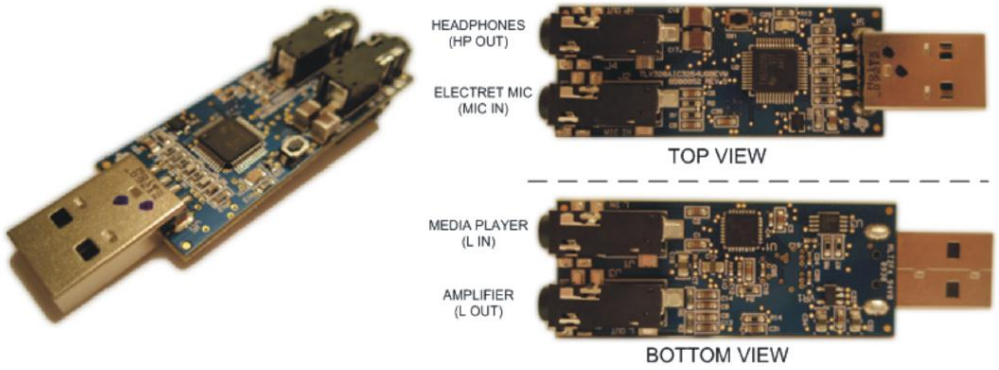


Figura 6.5: TLV320AIC3254EVM-U

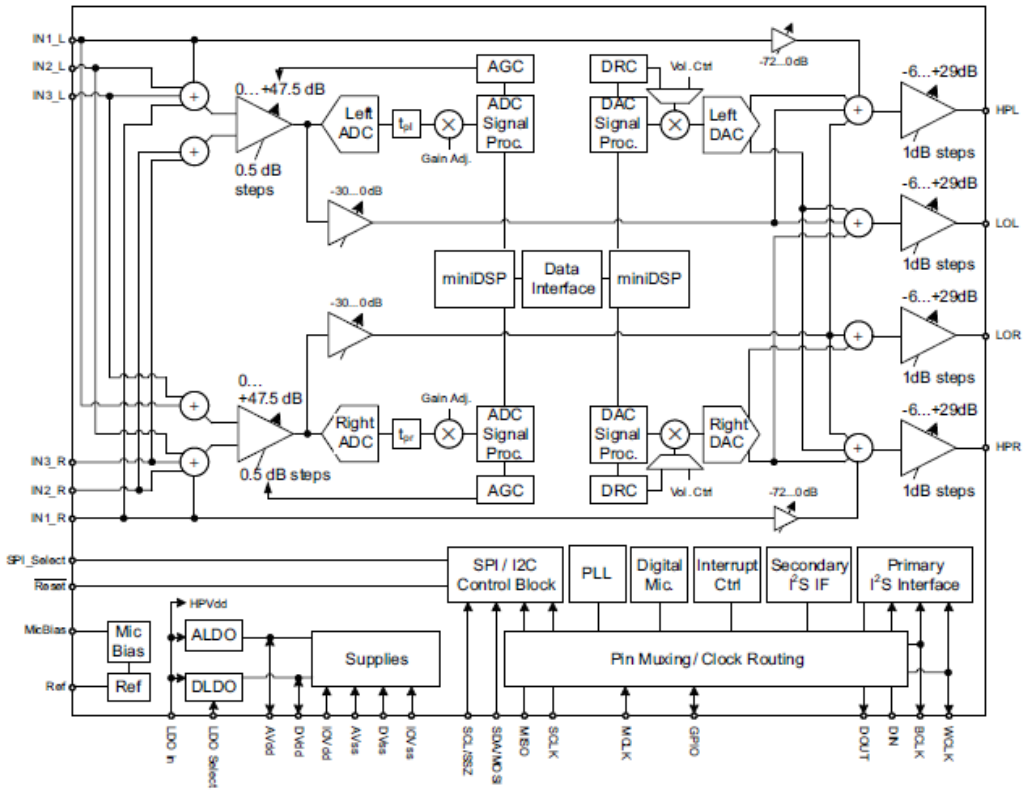


Figura 6.5: Diagramma a blocchi del TLV320AIC3254EVM-U

Il TLV320AIC3254EVM-U dispone di 2 connettori d'ingresso audio analogici stereo, il Mic-in (IN 3) e il Line-in (IN 2) configurabili come single-ended o come differenziali; 2 connettori d'uscita audio analogici stereo, l'Headphone-out (HO) e il Line-out (LO); e di un controller per lo streaming audio il TAS1020B che gestisce l'interfaccia USB-board, e comunica con il codec audio AIC3254, a livello di controllo tramite il bus I2C, a livello di dati audio tramite il bus I2S.
La board può lavorare, a seconda dell'applicazione per la quale viene programmata, a frequenze che spaziano dagli $8kHz$ mono fino alla frequenza di $192kHz$ stereo.
Tra i dispositivi disponibili sulla board è configurabile un preamplificatore stereo microfono ed è possibile gestire l'eventuale alimentazione richiesta da un microfono collegato esternamente qualora fosse necessaria.
Le uscite Headphone sono amplificate e fully-differential mentre il range di alimentazione per la TLV320AIC3254EVM-U varia da 1,5V - 1,95V per applicazioni analogiche e da 1,26V – 1,95V per applicazioni digitali grazie agli LDO integrati che riescono a generare l'appropriata alimentazione.
Dispone inoltre di un PLL programmabile, con clock che variano nel range da $512kHz$ a $50MHz$.

6.4.1 Setting Iniziale della TLV320AIC3254EVM-U

Procediamo con il setting iniziale della board per poi andare a realizzare la programmazione vera e propria definendone le funzionalità.
Accedendo con il Control Software la prima operazione è andare a mappare quali ingressi vogliamo utilizzare e per ciascuno di essi abbiamo la possibilità di scegliere il valore resistivo con il quale andare a contattarli (*Figura 6.5*).

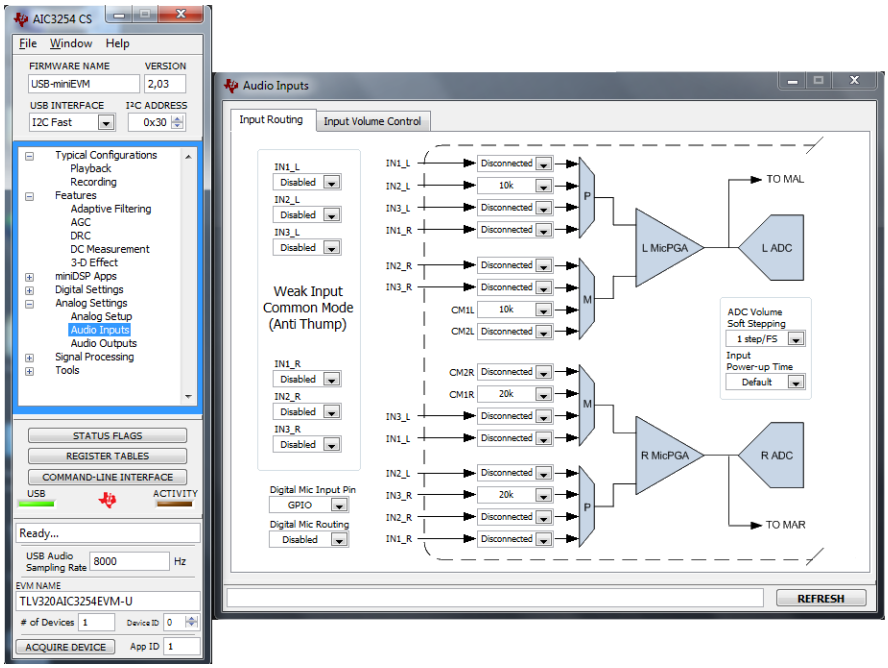


Figura 6.5: Control Software – Gestione degli ingressi

Come è possibile notare dalla precedente figura, tutti i canali left d'ingresso e tutti i canali right d'ingresso vengono processati parallelamente, ma tramite due percorsi distinti incontrando due preamplificatori microfonici (*MicPGA*) e due convertitori analogico/digitale (*ADC*) diversi.

Successivamente andiamo a configurare per ciascuno di questi dispositivi i livelli di volume a seconda dell'applicazione e delle esigenze operative (*Figura 6.6*).

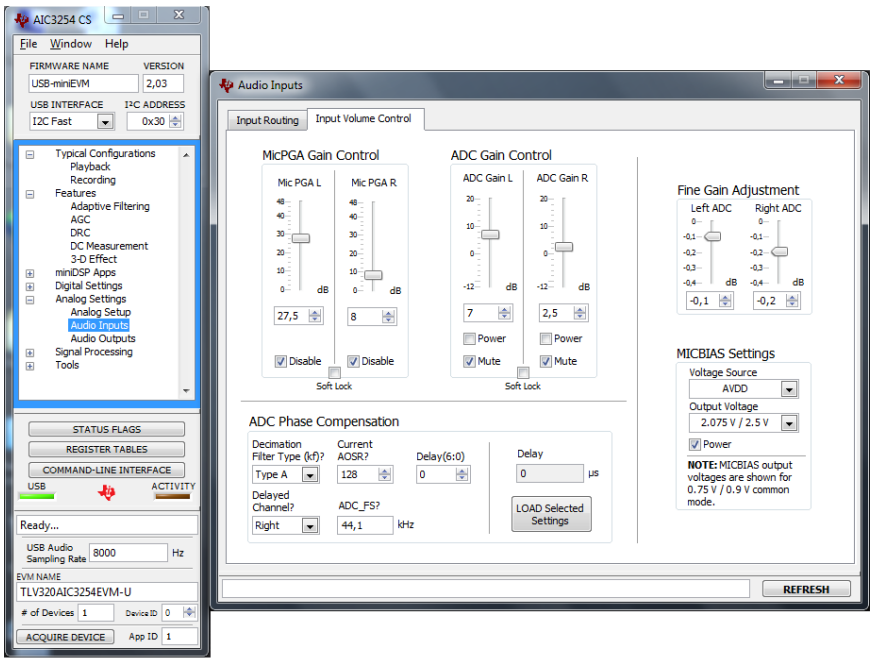


Figura 6.6: Control Software – Controllo del volume degli ingressi

In questa fase abbiamo la possibilità di stabilire l'appropriata alimentazione per il canale microfonico (*MICBIAS*), ed eventualmente di spegnere un canale, nel momento in cui, non fosse più necessario.
Analogamente andiamo a mappare le uscite, si osservi (*Figura 6.7*), la possibilità di processare i flussi in uscita dai convertitori digitale/analogico (*DAC*) quello del canale left e quello del canale right, parallelamente su entrambe le uscite dello stesso canale.

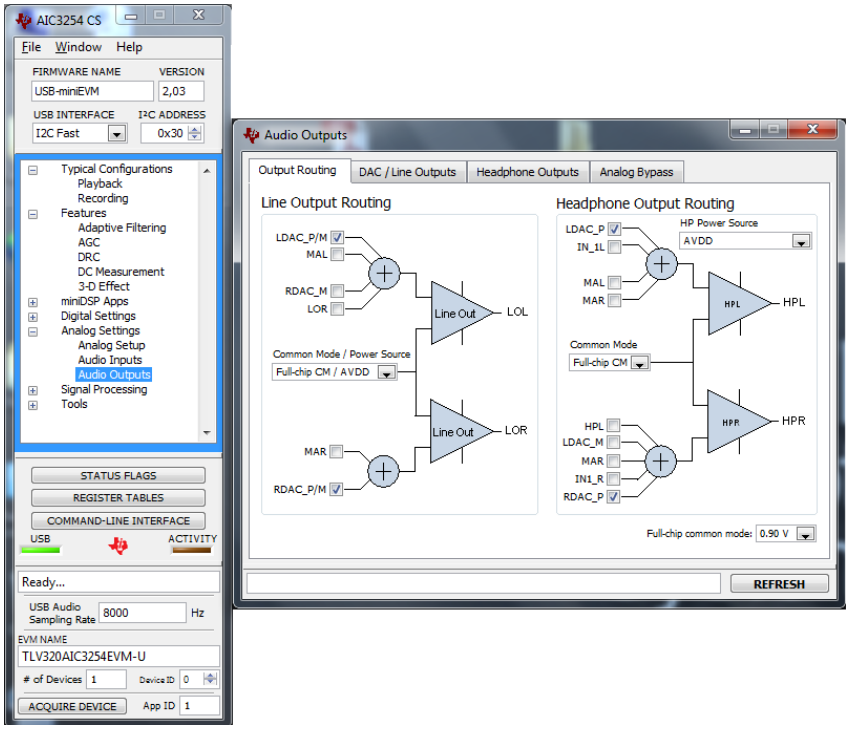


Figura 6.7: Control Software – Gestione delle uscite

Anche in questo caso andiamo a configurare per ciascuno di questi dispositivi i livelli di volume a seconda dell'applicazione e delle esigenze operative (*Figura 6.8*).

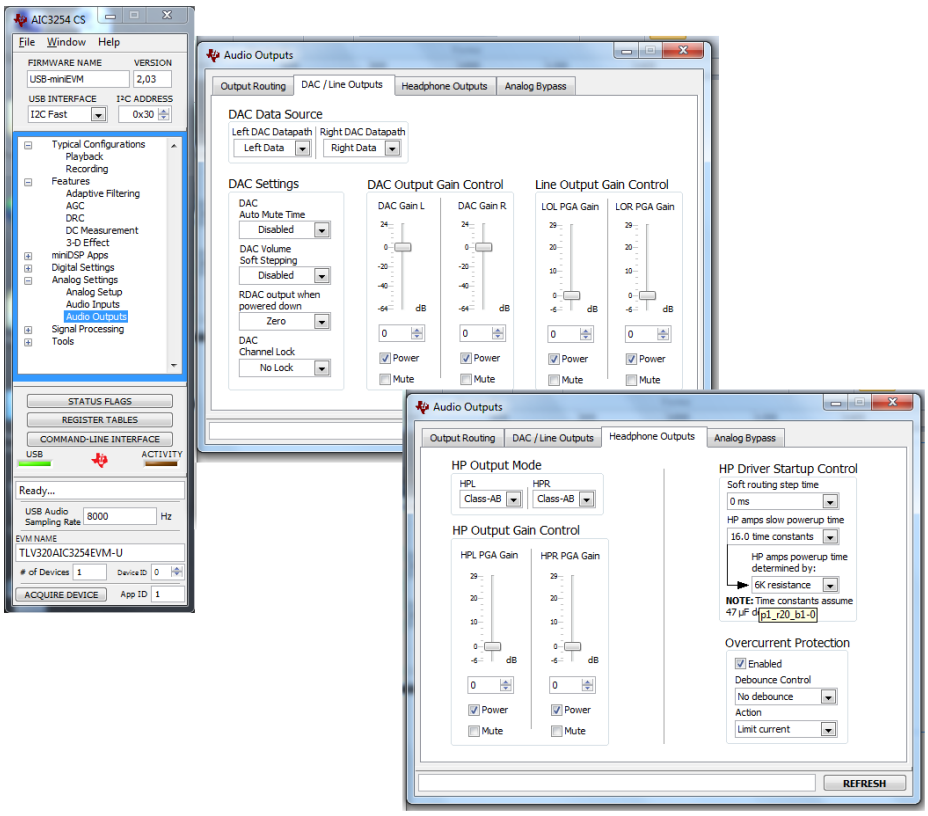


Figura 6.8: Control Software – Controllo del volume delle uscite

Nel dettaglio per quanto riguarda le uscite Headphone possiamo definire il tipo di amplificazione dello stadio d'uscita, a seconda del carico che andremo a pilotare, possiamo indicare sia di classe AB oppure di classe D.
Si nota la possibilità di attivare un controllo di protezione per la sovracorrente che limita la corrente d'uscita, preservando l'integrità della scheda, nell'eventualità la connessione di un carico errato provochi un eccessivo assorbimento.
Infine, tra le funzionalità a disposizione nel Control Software, utili dopo la programmazione che andremo a realizzare con il PurePath Studio, ci sono (*Figura 6.9*):

- *Status Flag*: questa utility monitora i flag già definiti nella TLV320AIC3254EVM-U, quali per esempio il controllo se è stato programmato il guadagno per ciascun dispositivo sulla scheda, oppure eventuali segnalazioni di overflow degli ADC, dei DAC, l'attivazione della protezione per la sovracorrente.

- *Register Table*: questa utility consente di andare a leggere e/o scrivere dei valori sulle pagine dei registri della scheda, in modo da poter analizzare in fase di debug come variano i dati nei vari registri.

- *Command-Line Interface*: questa funzionalità consiste nell'avere a disposizione una riga di comando con la quale è possibile eseguire degli script.

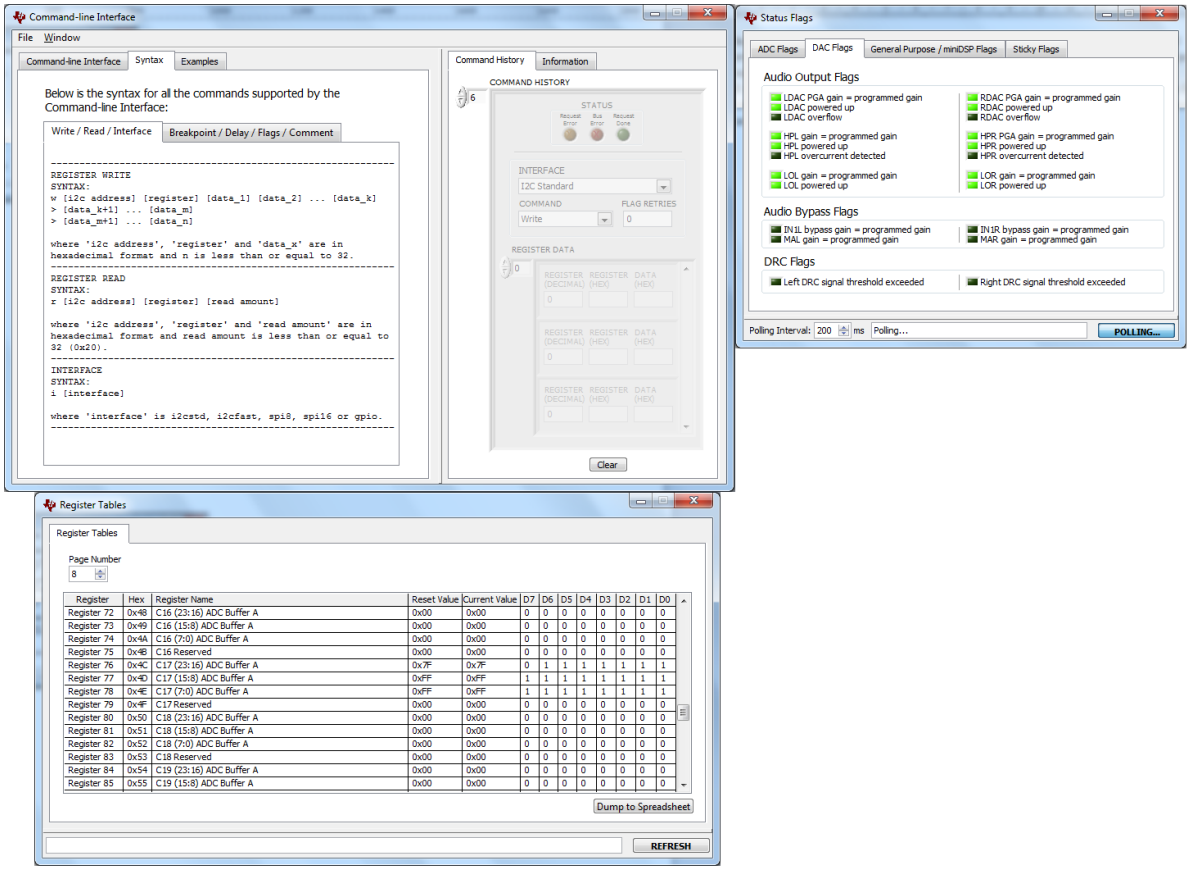


Figura 6.8: Control Software – Tools dei registri

6.4.2 Connessione Elettrica della TLV320AIC3254EVM-U al Touchscreen

La connessione della TLV320AIC3254EVM-U al touchscreen viene realizzata nel seguente modo:

- Si disconnettono elettricamente il microfono e lo speaker del touch e li si pilotano direttamente dalla TLV320AIC3254EVM-U rispettivamente: il microfono sarà connesso all'ingresso microfonico canale sinistro Mic-L (IN3_L) e lo speaker sarà connesso all'uscita Headphone canale sinistro HPL, amplificata con classe AB

- Il connettore che gestiva lo speaker sulla scheda del touch va collegato all'ingresso Line canale destro LI-R (IN2_R) della TLV320AIC3254EVM-U

- Il connettore che gestiva il microfono sulla scheda del touch va collegato all'uscita Line canale destro LO-R della TLV320AIC3254EVM-U, nel caso specifico questa connessione è stata disaccoppiata in continua con un condensatore da 100nF, in quanto il connettore microfonico della scheda del touch presenta una tensione continua a circa 3V, polarizzazione che era necessaria per l'alimentazione del microfono, ma che adesso disaccoppiamo, affinché non carichi l'uscita della TLV320AIC3254EVM-U.

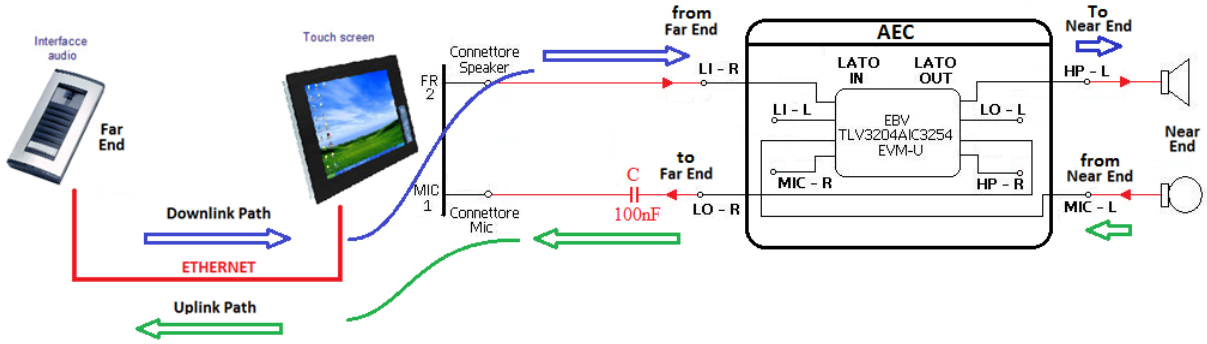


Figura 6.9: Schema delle connessioni elettriche fra touchscreen e AEC

6.4.3 Struttura Interna della TLV320AIC3254EVM-U

Prima di procedere a esporre la modalità di programmazione della scheda come AEC, è opportuno capirne la struttura interna per sapersi orientare successivamente ai fini di una corretta programmazione.

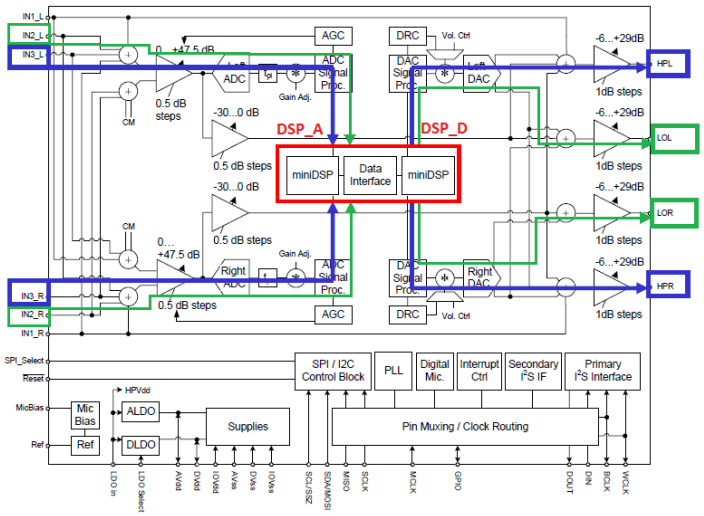


Figura 6.10: Dettaglio del diagramma a blocchi del TLV320AIC3254EVM-U

Si nota (*Figura 6.10*) come gli ingressi e le uscite siano state mappate verso il core dell'AIC3254 (*Figura 6.11*), il quale è caratterizzato da due miniDSP distinti: il DSP_A, lato ingressi o meglio lato ADC e il DSP_D, lato uscite o meglio lato DAC, entrambi programmabili tramite il PurePath Studio (*PPS*).

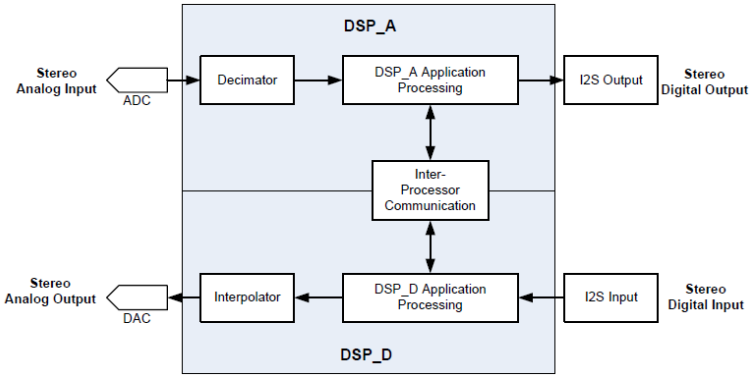


Figura 6.11: Architettura del core

6.5 Introduzione al PurePath Studio (PPS)

Il PurePath Studio è un ambiente di sviluppo integrato della Texas Instruments, basato su un linguaggio di programmazione visuale grafico, *Graphical Development Environment* (*GDE*), che permette la programmazione di dispositivi basati su architetture a miniDSP per applicazioni audio e più in generale tutto ciò che concerne l'elaborazione e il processo di segnali, è l'analogo del conosciutissimo LabVIEW della National Instruments in ambito dell'automazione industriale.
L'ambiente mette a disposizione un completo e ampio pacchetto di componenti elementari che realizzano e implementano le più svariate funzioni a livello di elaborazione dei segnali. Combinando insieme questi componenti è possibile realizzare algoritmi di una considerevole complessità a livello computazionale. La programmazione si distingue dai linguaggi tradizionali in quanto non viene realizzata in forma di testo e si svincola dalla stesura di codice a basso livello ma si sposta su un piano grafico (*Graphic Language, G-Language*), combinando mediante logica *drag & drop* i blocchi dei componenti elementari disponibili sotto forma di icone e altri oggetti grafici, su una finestra di diagramma del progetto, poi vanno uniti da linee di collegamento (*wire*), in modo da formare una sorta di diagramma di flusso, andando così a realizzare il processo desiderato.
Tale linguaggio viene definito *dataflow* (flusso di dati) in quanto la sequenza di esecuzione è definita e rappresentata dal flusso dei dati stessi attraverso i fili monodirezionali che collegano i blocchi funzionali. Poiché i dati possono anche scorrere in parallelo attraverso blocchi e fili non consecutivi, il linguaggio realizza spontaneamente il *multithreading* senza bisogno di esplicita gestione da parte del programmatore.
Un programma in G-Language, viene denominato **VI** (*Virtual Instrument*) questo non esiste sotto forma di testo, ma viene salvato come file binario interpretato correttamente solo dall'ambiente di sviluppo dedicato, è composto da tre componenti principali: *il pannello componenti, la finestra diagramma e il riquadro proprietà*.

Il pannello componenti è il set di blocchetti elementari disponibile, suddivisi per tipologia di elaborazione audio che realizzano; la finestra di diagramma del progetto è la destinazione dove collocare i componenti e le connessioni tra di essi; mentre il riquadro proprietà permette la configurazione e il setting del componente selezionato mediante parametri editabili o, per alcuni componenti, tracciando graficamente la caratteristica I/O voluta.

Infine è sufficiente generare il codice e inviarlo alla scheda per l'esecuzione, vediamo in *Figura 6.12* un diagramma a blocchi dell'intero processo di programmazione:

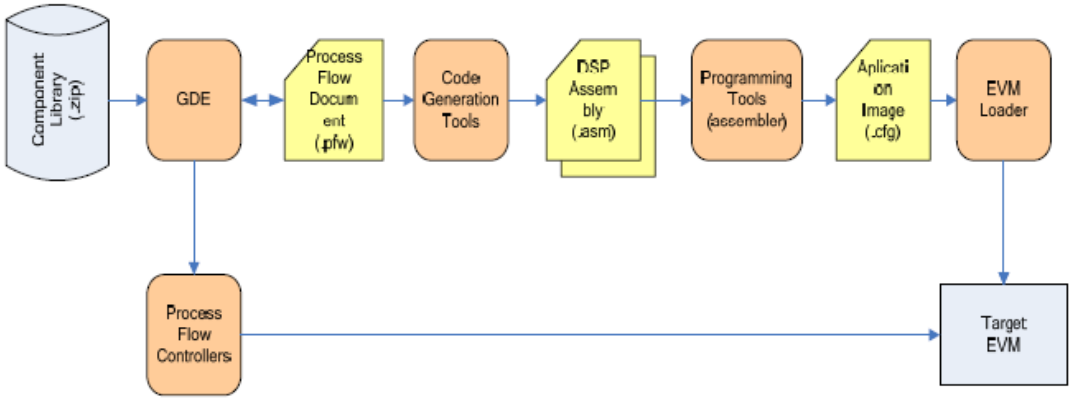


Figura 6.12: Processo di programmazione mediante PPS

6.6 Programmazione di Test: Streaming Audio

Per meglio comprendere il processo di programmazione della board abbiamo realizzato un semplice progetto di test, con l'obiettivo di programmare la scheda perché si comporti in modalità streaming audio, quindi che acquisisca un flusso audio in ingresso e lo riproduca *real-time* su un canale d'uscita.

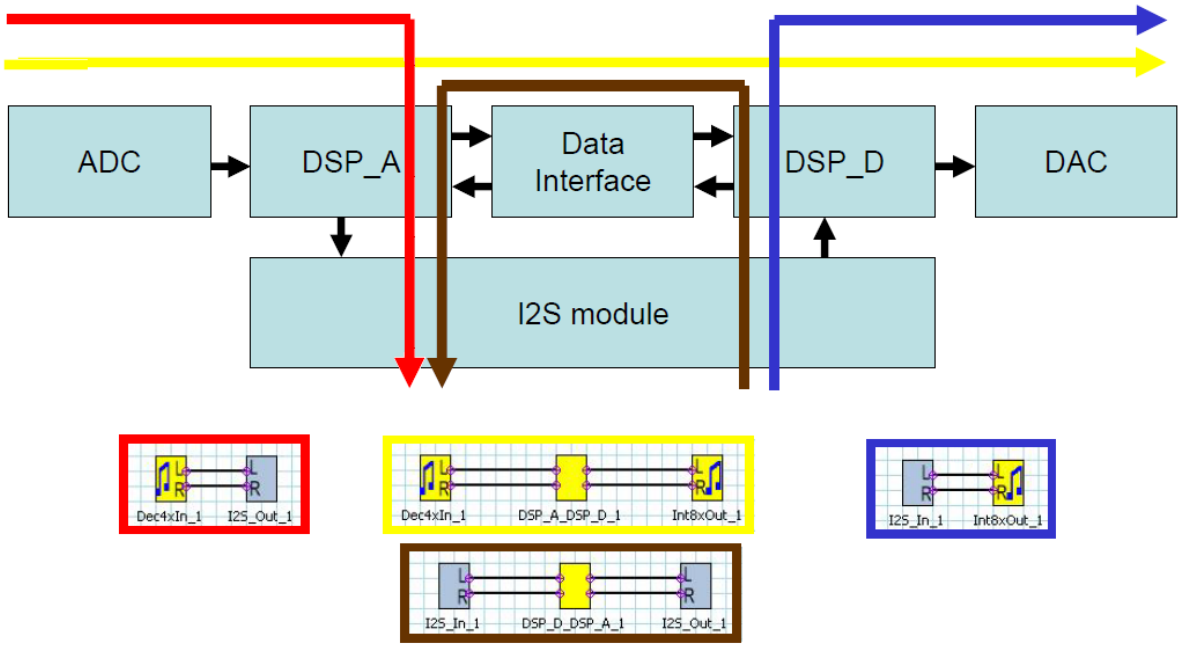


Figura 6.13: Flusso di progetto playback e programmazione G-Language

In *Figura 6.13* si notano 4 differenti flussi audio di streaming:

- **ADC - DAC** (colore giallo): il flusso è acquisito stereofonicamente dai connettori d'ingresso della TLV320AIC3254EVM-U, processato e riprodotto dai connettori d'uscita della board.
- **ADC - I2S_Out** (colore rosso): il flusso è acquisito stereofonicamente dai connettori d'ingresso della TLV320AIC3254EVM-U, processato e spedito in uscita sul bus I2S, indirizzato su USB.
- **I2S_In – DAC** (colore blu): il flusso è acquisito stereofonicamente dal bus I2S, indirizzato su USB, processato e riprodotto dai connettori d'uscita della TLV320AIC3254EVM-U.
- **I2S_In – I2S_Out** (colore marrone): il flusso è acquisito stereofonicamente dal bus I2S, indirizzato su USB, processato e rispedito in uscita sul bus I2S.

Per ciascun flusso, è rappresentata l'equivalente programmazione in G-Language che lo implementa, si notano la semplicità e l'intuitività caratteristici di questo genere di linguaggio. Osservando per esempio il flusso ADC –DAC (colore giallo), il miniDSP_A verrà programmato per eseguire i seguenti processi: applicare ai dati in uscita dall'ADC, un decimatore stereo $4x$ (ogni 4 campioni in ingresso ne fornisce 1 in uscita), gestire l'interfaccia *inter-processor* con il miniDSP_D; il miniDSP_D invece verrà programmato per eseguire i seguenti processi: gestire l'interfaccia *inter-processor* con il miniDSP_A e inviare i dati al DAC, antecedendo un interpolatore stereo $8x$ (ogni campione in ingresso ne fornisce 8 in uscita).
A questo punto non ci resta che mettere insieme i 4 processi in un unico progetto (*Figura 6.14*), generarne il codice associato e caricarlo sulla board.

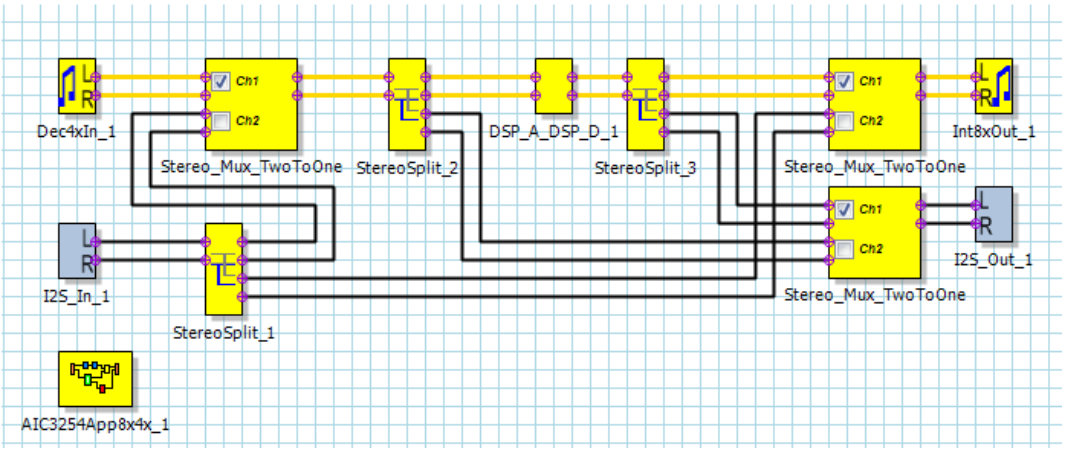


Figura 6.14: Progetto Streaming Audio

In *Figura 6.14,* per chiarezza è stato evidenziato uno dei flussi ADC – DAC (colore giallo), si nota che nel progetto è stato necessario aggiungere dei componenti quali gli *splitter*, per la duplicazione delle linee di connessione e i *multiplexer*, per selezionare i percorsi attivi.
Notiamo l'aggiunta di un componente (*in basso a sinistra*) l'AIC3254App8x4x è un *framework*, cioè una struttura di supporto, che indica all'ambiente di sviluppo su quali board e codec stiamo lavorando, settando così, direttamente il clock e il numero massimo di cicli macchina di cui disponiamo per il progetto: ovviamente questo comporta una limitazione dei progetti e delle applicazioni realizzabili, legata alla capacità computazionale di cui disponiamo. Tipicamente la board TLV320AIC3254EVM-U è configurata per supportare frequenze di campionamento che vanno dagli $8kHz$ ai $48\ kHz$.

6.7 Programmazione Anti-Larsen: AEC basato su Algoritmo NLMS

A livello di funzionalità, disponibili nelle librerie dell'ambiente di sviluppo, sfruttiamo il blocchetto configurabile che implementa l'algoritmo adattativo NLMS per la cancellazione dell'eco acustico. L'algoritmo NLMS è basato su una procedura iterativa che va a fare continue correzioni al peso dei coefficienti del filtro adattativo in funzione dell'errore misurato, con l'obiettivo infine di minimizzare l'errore quadratico medio (*MSE, Mean Square Error*).
È stato previsto, per limitazioni dovute alla capacità di elaborazione del core AIC3254, che il filtro NLMS operi a una frequenza di campionamento di $8kHz$ e agisca al più su ritardi d'eco di lunghezza pari a 32ms.

L'applicazione del filtro NLMS, richiede l'impiego di ulteriori 3 componenti:

- *NLP Controller* (*Non-Linear Processor Controller*): la cui funzione è provvedere alla soppressione dell'eco residuo là dove l'NLMS non fosse sufficiente da solo.
- *CNG* (*Comfort Noise Generator*): la cui utilità consiste nel "mascherare" gli effetti e i cambiamenti che vengono introdotti dall'NLP Controller quando quest'ultimo è attivo. Inoltre ovvia all'effetto "*dead air*" in quanto elimina il rischio di un'interruzione involontaria del flusso riprodotto.
  Questo effetto se non configurato correttamente può compromettere significativamente la comunicazione tra Far-End e Near-End.
- *CC* (*Center Clipper*): la cui finalità è valutare quali campioni audio presentano una piccola ampiezza, inferiore rispetto un livello di soglia configurabile, e li sostituisce con dei campioni nulli; in questo modo anche un persistente residuo eco a bassissimo livello viene reso inudibile.

Vediamo in *Figura 6.15* uno schema a blocchi dell'azione di questi componenti nel nostro sistema:

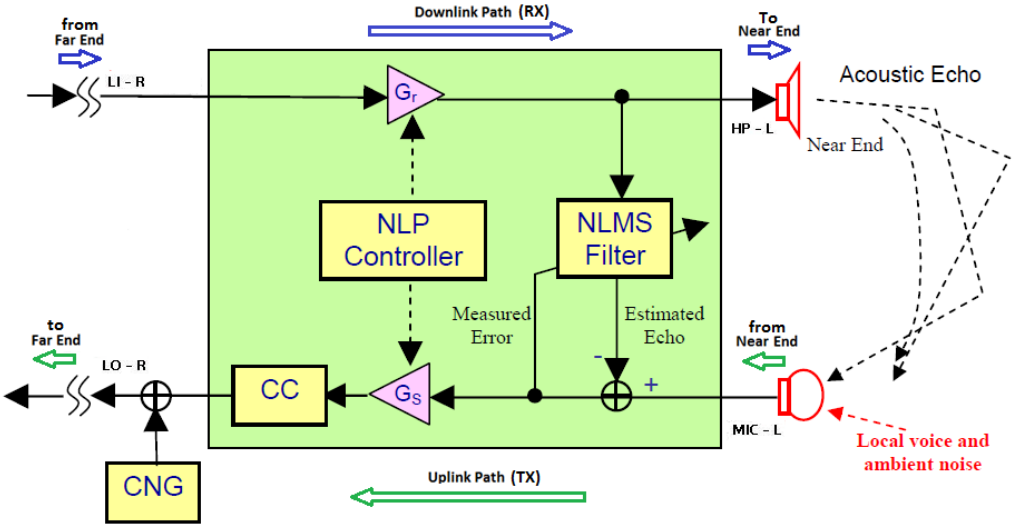


Figura 6.15: Schema a blocchi della programmazione dell'AEC

Il segnale d'ingresso proveniente dal Far-End è diretto, tramite il percorso di Downlink (RX), al Near-End e anche al modulo che realizza il filtro NLMS, dove ne viene stimato l'eco e sottratto al segnale in ingresso al cammino di Uplink (TX), proveniente dal Near-End.
Idealmente, una fedele stima dell'eco, fa si che l'eco venga completamente rimosso e venga inviato al Far-End soltanto il segnale utile presente in ingresso al Near-End.
In realtà, vanno considerati i cambiamenti dinamici ai quali è soggetto il segnale di Uplink (prima di essere analizzato), come la distorsione non-lineare introdotta dallo speaker, i cambiamenti dell'ambiente acustico al Near-End, e la distorsione dovuta al microfono.
In queste condizioni, la stima dell'eco effettuata dall'NLMS, potrebbe non corrispondere esattamente all'eco legato dalle condizioni reali del sistema.
Per errore si intende la differenza tra l'ingresso del percorso di Uplink (from Near-End) e l'eco stimato dall'algoritmo NLMS, questo segnale di errore viene utilizzato per correggere i coefficienti del filtro adattativo NLMS con l'obiettivo di far convergere esattamente l'eco stimato all'eco reale del sistema.
La velocità di questa convergenza, è legata alla configurazione di due specifici parametri:

- *faster step size*: questo parametro viene applicato automaticamente se il segnale microfonico presente al Near-End è costituito solo da eco acustico.
- *slower step size*: questo parametro interviene se il segnale microfonico presente al Near-End è costituito sia da eco acustico che da parlato (*double talk*)

Ovviamente, se non viene rilevato nessun contributo di eco acustico all'ingresso microfonico non verrà fatto nessun adattamento dei coefficienti del filtro.
Laddove l'NLMS non riesce a rimuovere completamente l'eco, l'NLP Controller va ad agire rimuovendo l'eco residuo: inizialmente il controller va a classificare la comunicazione in atto secondo la seguente tipologia, e successivamente applica l'appropriata attenuazione ai segnali di Uplink e/o Downlink rendendo l'eco residuo il più inudibile possibile.

- *Idle*: non si rilevano segnali significativi né al Far-End né al Near-End
- *Double Talk*: si rilevano segnali significativi sia al Far-End sia al Near-End
- *Far-End*: si rilevano ampiezze di segnale significative solo lato Far-End
- *Near-End*: si rilevano ampiezze di segnale significative solo lato Near-End

L'aggressività dell'NLP Controller è gestita da alcuni parametri editabili pre-configurati (Gs e Gr, *Figura 6.15*) nelle diverse modalità di comunicazione sopra citate; la modifica di tali parametri può indurre il controller a una carente aggressività inibendone l'effetto e lasciando inalterato un eventuale eco residuo; oppure può comportare un'eccessiva aggressività, invalidando la comunicazione full-duplex, rendendola half-duplex: in quanto non permetterà alla sorgente lato Near-End di intervenire nella comunicazione fintanto che la sorgente Far-End sta comunicando. Il componente che implementa il filtro NLMS, a livello di progetto in PPS, e le sue proprietà sono rappresentate in *Figura 6.16*, dove si possono configurare anche l'*NLP Controller* e il Center Clip (*CC)*:

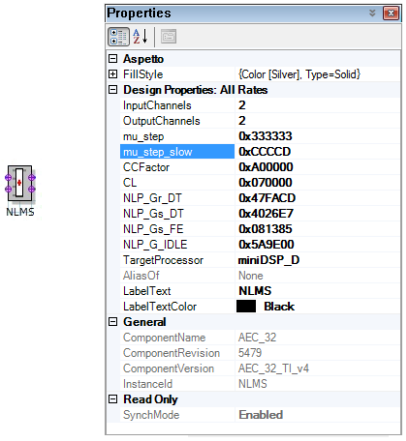


Figura 6.16: Componente NLMS

Il *CNG* (*Comfort Noise Generator*) applicato al percorso di Uplink, genera rumore "*rosa*", che a differenza del rumore bianco in cui la potenza è uguale per qualsiasi componente in frequenza, questo è caratterizzato dal fatto che le componenti a bassa frequenza hanno potenze maggiori. In acustica questo tipo di rumore è strutturato in modo tale da compensare la sensibilità dell'orecchio umano alle varie frequenze, e viene utilizzato per l'equalizzazione del suono in ambito professionale. Il CNG *(Figura 6.17*) fa una stima del livello di rumore di fondo lato Near-End e genera del rumore rosa da aggiungere al segnale che viene spedito al Far-End con l'obiettivo di renderlo più "morbido". Inoltre, per limitate variazioni dei segnali, maschera i transitori del NLP Controller quando questo cambia la modalità di funzionamento. Infine, elimina l'effetto "*dead air*" causato dall'eliminazione di tutto il rumore residuo presente in linea. Il filtro del rumore rosa è un tempo invariante del secondo ordine i cui poli sono completamente allocabili.

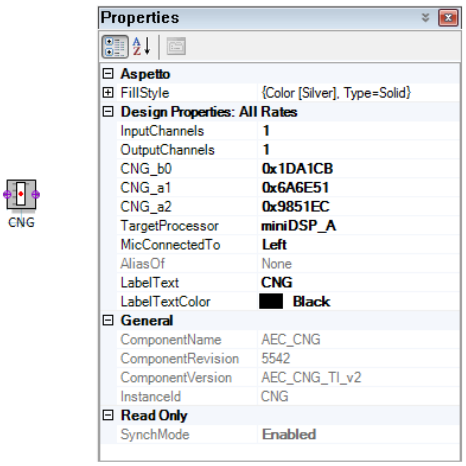


Figura 6.17: Componente CNG

Vediamo ora la realizzazione definitiva del progetto dell'AEC completa, *Figura 6.18*:

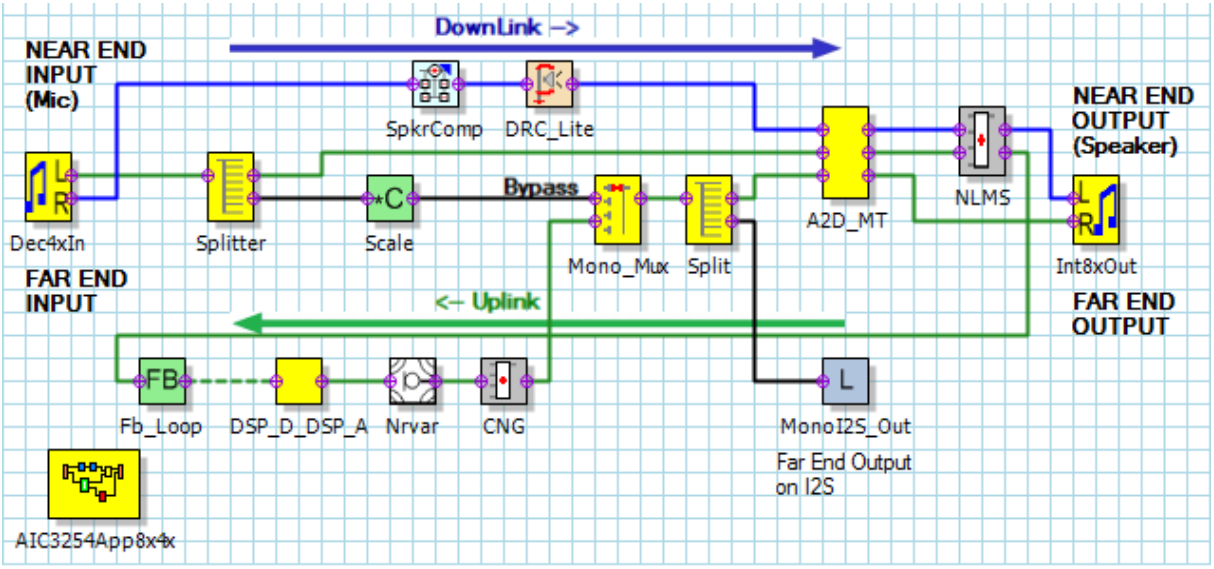
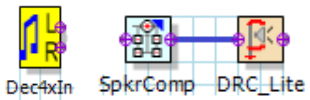


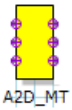
Figura 6.18: Progetto AEC

Il percorso di Downlink (colore blu) riceve il flusso audio proveniente dal Far-End collegato al canale right del Line-In, si effettua:

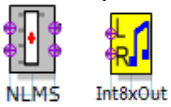
una decimazione $4X$ e un'equalizzazione per la riproduzione su altoparlante,



successivamente viene indirizzato all'interfaccia dati inter-processore per il trasferimento dei dati dal miniDSP_A al miniDSP_D,



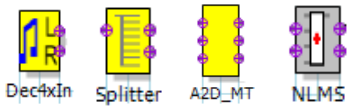
e infine passato sia al filtro $NLMS$, sia all'interpolatore $8X$,



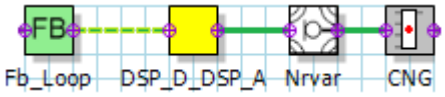
per venire riprodotto dallo speaker del Near-End collegato al canale left d'uscita Headphone.

Il percorso di Uplink (colore verde) riceve il flusso audio proveniente dal Near-End collegato al canale left del Mic, si effettua:

una decimazione $4X$, uno sdoppiamento (*split*), viene indirizzato all'interfaccia dati inter-processore e all'$NLMS$,

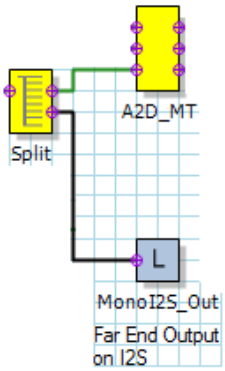


l'uscita del filtro $NLMS$, che è il segnale di errore, prima di venire spedita al Far-End, viene opportunamente trattata e le viene aggiunto del rumore rosa con il CNG,



successivamente, un multiplexer (Mux) permette di selezionare se si vuole applicare o meno l'AEC all'anello, nell'eventualità questo venisse escluso, si può ugualmente regolare il livello del segnale grazie al componente *Scale*,

in serie al multiplexer, uno splitter invia i dati sia all'interfaccia dati inter-processore che all'$I2S$,

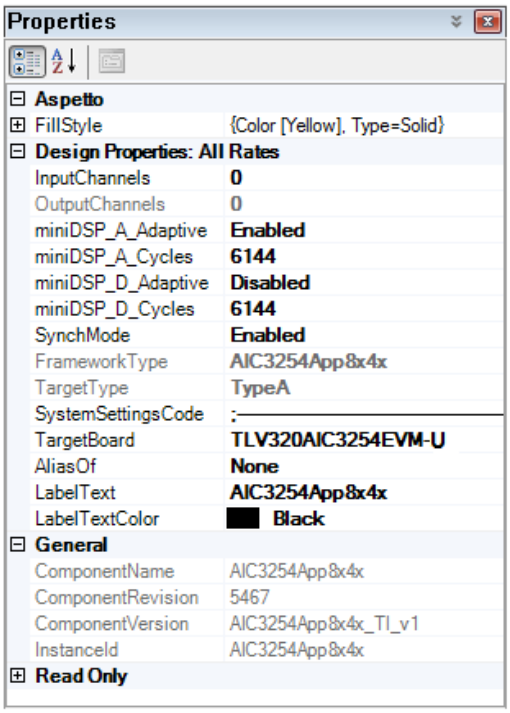


e infine il flusso viene processato da un interpolatore $8X$ che lo indirizza verso il Far-End collegato al canale right del Line-out.

6.7.1 Proprietà e Specifiche dei Componenti

Vediamo nel dettaglio le specifiche e il setting delle proprietà per ciascun componente significativo di progetto:

"AIC3254App8x4x" Framework AIC3254 con Decimatore $4X$ e Interpolatore $8X$

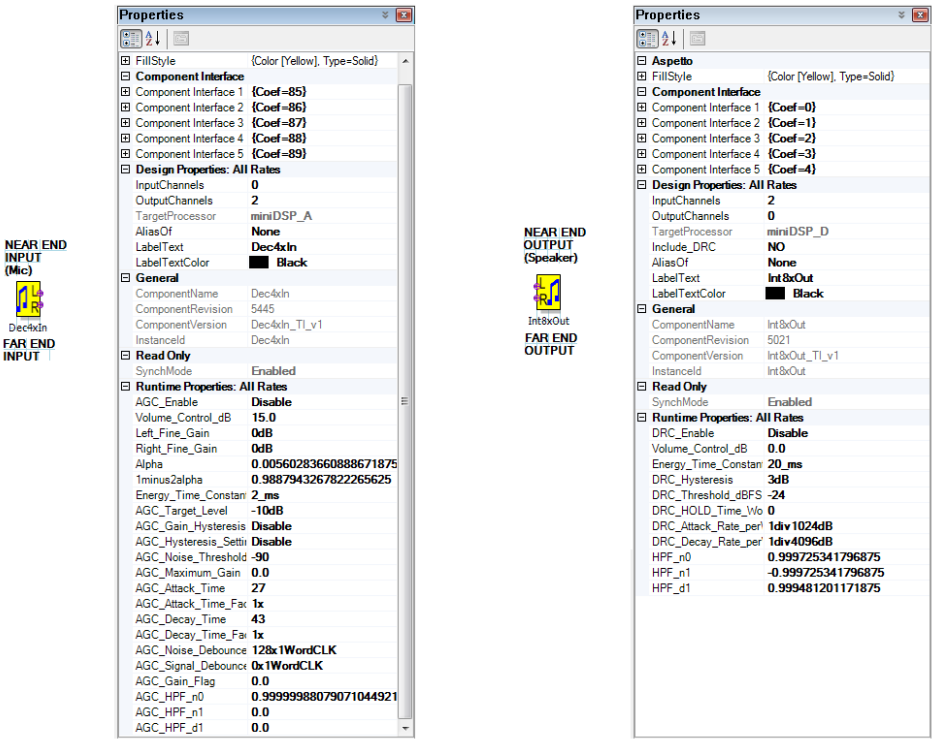


La scelta del framework appropriato, determina la configurazione dei registri dell'AIC3254 necessari per la specifica operazione che si va ad implementare; inclusi i registri di setting del clock, degli I/O, e dell'alimentazione dei dispositivi presenti sulla board. I Framework disponibili sono $8x4x$, $4x2x$, $2x1x$. Ogni framework supporta un range specifico di frequenze di campionamento e soprattutto capacità differenti di cicli istruzione disponibili. Per questo motivo la scelta del framework è legata alla tipologia di decimazione, interpolazione e interfaccia dati inter-processore che caratterizzano il progetto.
Si può notare *Tabella 6.1* le risorse di cui disponiamo nel nostro progetto, pertanto le funzionalità, i componenti delle librerie e la complessità degli algoritmi che andremo a sviluppare, saranno limitati, in quanto non potranno eccedere tali specifiche.

AEC_32	Resource Available on AIC3254	
miniDSP_A	Instruction Words	1024
	Data RAM	896
	Coefficient RAM	512
	Cycles/Frame	904
miniDSP_D	Instruction Words	1024
	Data RAM	896
	Coefficient RAM	512
	Cycles/Frame	6144

Tabella 6.1: Risorse disponibili sull'AIC3254

"Dec4xIn" Decimatore $4X$ e "*Int8xOut*" Interpolatore $8X$



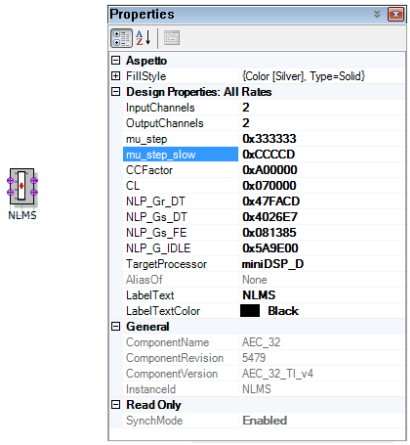
Questi componenti implementano rispettivamente: una funzione di ADC/decimazione gestita dal miniDSP_A e una funzione di DAC/interpolazione gestita dal miniDSP_D.
L'opportuna scelta dei fattori di decimazione e interpolazione, come abbiamo visto, è definita a livello di framework.

In particolare, si può notare la possibilità di gestire un *AGC* (*Automatic Gain Control*) per regolare automaticamente l'amplificazione dei segnali a un livello desiderato, nel nostro progetto tale funzionalità non la utilizziamo, ma amplifichiamo costantemente a $15\ dB$ i segnali in ingresso tramite il parametro *Volume_Control_dB*.
C'è la possibilità di configurare un *DRC* (*Dynamic Range Compression*) la cui funzione consiste nel limitare il guadagno, fornito ai dati processati dal DAC, quando questi superano una soglia specificata. Il tasso al quale il DRC diminuisce o aumenta il guadagno, in funzione delle variazioni di ampiezza dell'ingresso, è legato ai tempi di intervento desiderati, gestiti rispettivamente dai parametri "*Attack*" e "*Decay*".

Property	Permitted Values	Description
AGC_Enable	Enable / Disable	AGC Enable or Disable
Volume_Control_dB	-12dB to 20dB	PGA value. Effective only when AGC_Enable is set to Disable.
Left_Fine_Gain	0dB to 0.4dB and Muted	Fine gain on left channel
Right_Fine_Gain	0dB to 0.4dB and Muted	Fine gain on Right channel. Not present in Mono configuration.
Energy_Time_Constant_ms	1 to 350 ms	Energy time constant
AGC_Target_Level	-5.5dB to -24dB	Target AGC level
AGC_Gain_Hyteresis	0.5dB, 1dB, 1.5dB and disable	AGC hysteresis
AGC_Hyteresis_Setting	1.0dB, 2.0dB, 4dB and disable	AGC hysteresis setting
AGC_Noise_Threshold	-30dB to -90dB	AGC noise threshold
AGC_Maximum_Gain	0dB to 58dB, in steps of 0.5dB	Maximum gain applied by AGC
AGC_Attack_Time	1 to 63, times 32/Fs	AGC attack time
AGC_Attack_Time_Factor	1 to 128 attack time scale factor	AGC attack time scale factor
AGC_Decay_Time	1 to 63, times 512/Fs	AGC decay time
AGC_Decay_Time_Factor	1 to 128	AGC decay time scale factor
AGC_Noise_Debounce_Time	0 to 21*4096 1/Fs	AGC noise debounce time
AGC_Signal_Debounce_Time	0 to 6*2048 1/Fs	AGC signal debounce time
AGC_HPF_n0	-1 to 0.9999998807907104492187 5	High pass filter coefficient n0
AGC_HPF_n1	-1 to 0.9999998807907104492187 5	High pass filter coefficient n1
AGC_HPF_d1	-1 to 0.9999998807907104492187 5	High pass filter coefficient d0

Property	Permitted Values	Description
DRC_Enable	YES / NO	DRC Enable or Disable.
Volume_Control_dB	-63.5dB to 24dB	Digital Volume Control.
Energy_Time_Constant_ms	1 to 350 ms	Used in the signal energy calculations by DRC.
DRC_Hyteresis	0dB, 1dB, 2dB, 3 dB	It is a programmable window around the programmed DRC Threshold that must be exceeded for a disabled DRC to become enabled, or an enabled DRC to become disabled.
DRC_Threshold_dBFS	-24dB to -3dB	DRC threshold. Level of signal at which gain compression becomes active.
DRC_HOLD_Time_WordCLK	0, 32, 64, 128,.....,128x4096	DRC hold time in word clocks.
DRC_Attack_Rate_perWCLK	4dB, 2dB, 1dB, 1/2dB,....., 1/4096dB	DRC attack rate per DAC word clock.
DRC_Decay_Rate_perWCLK	1/64dB, 1/128dB, 1/256dB,....., 1/2097152dB	DRC decay rate per DAC word clock.
HPF_n0	-1 to 0.9999998807907104492187 5	High pass filter coefficient n0
HPF_n1	-1 to 0.9999998807907104492187 5	High pass filter coefficient n1
HPF_d1	-1 to 0.9999998807907104492187 5	High pass filter coefficient d0

"NLMS" Filtro NLMS – AEC $32\ ms$



Il componente viene completamente gestito dal miniDSP_D e implementa: un filtro adattativo NLMS di 256 coefficienti (pari a una lunghezza di $32\ ms$), le funzionalità di NLP e CC. I parametri configurabili del filtro NLMS sono il "*mu_step*" e il "*mu_step_slow*" corrispondenti ai parametri già discussi, *faster step size* e *slower step size*.
Il "*mu_step*" determina la rapidità con la quale l'algoritmo NLMS convergerà, andando così a cancellare l'eco, quando è attiva la modalità Far End (*FE*). L'impostazione di un valore troppo elevato indurrà tempi di intervento rapidissimi ma degraderà le prestazioni del filtro in termini di stabilità invece l'impostazione di un valore troppo basso porterà a tempi di convergenza lunghi con il rischio di compromettere la cancellazione d'eco.
Il valore desiderato μ va convertito nel formato a virgola fissa nel seguente modo:

$$mu\_step\ =\ round\ \{\mu\ *\ 2^{25}\}$$

A un valore di $\mu = 0.1$ corrisponde $mu\_step\ = 0x333333$
Il "*mu_step_slow*" va impostato a un valore inferiore rispetto il *mu_step*, determina la rapidità con la quale l'algoritmo NLMS convergerà, andando così a cancellare l'eco, quando è attiva la modalità Double Talk (*DT*). Analogamente al *mu_step* l'entità di questo parametro è legata alla convergenza e alla stabilità del filtro durante la modalità *DT*.
Il valore desiderato μ va convertito nel formato a virgola fissa nel seguente modo:

$$mu\_step\_slow\ =\ round\ \{\mu\ *\ 2^{25}\}$$

A un valore di $\mu = 0.025$ corrisponde $mu\_step\_slow\ = 0xCCCCD$
Il "*CCfactor*" (*Center Clipper threshold*) interviene per livelli di segnali molto bassi e rimuove l'eco residuo ponendolo a zero fintanto che l'eco rimane al di sotto della soglia indicata da questo parametro. È configurabile in un range che va da 0.0 a -1.0 dove con 0.0 la funzionalità è disabilitata invece con -1.0 è configurata per agire con il massimo effetto.
Il valore desiderato CC va convertito nel formato a virgola fissa nel seguente modo:

$$CCfactor\ =\ round\ \{CC\ *\ -2^{23}\}$$

A un valore di $CC = -0.75$ corrisponde $CCfactor\ = 0xA00000$
Il "*CL*" (*Coupling Loss*) va a configurare in modo appropriato l'NLP Controller per il sistema che si sta utilizzando: questo parametro rappresenta il rapporto tra il livello digitale d'uscita Rx e il corrispondente livello digitale di eco acustico all'ingresso Tx. Se lo speaker e il

microfono sono acusticamente ben disaccoppiati, il *CL* assumerà un valore significativo perché il livello di eco risulterà molto ridotto. Tipici valori di *CL* spaziano dai 10 ai $20\ dB$ per applicazioni in ambito mobile phone e dai -15 a $0\ dB$ per applicazioni audio in banda fonica.
Il valore desiderato CL_dB va convertito nel formato a virgola fissa nel seguente modo:

$$CL\ =\ round\ \{10^{(-0.1\ *\ CL\_dB)}\ *\ 2^{17}\}$$

A un valore di $CL\_dB\ =0\ dB$ corrisponde $CL\ =0x020000$
Il "*NLP Gr DT*" configura il livello di attenuazione nel cammino di downlink (Rx) in modalità *DT*, il "*NLP Gs DT*" configura il livello di attenuazione nel cammino di uplink (Tx) in modalità *DT,* il "*NLP Gs FE*" configura il livello di attenuazione nel cammino di uplink (Tx) in modalità *FE*, e il "*NLP G IDLE*" configura il rapporto tra il livello di attenuazione nel cammino di downlink (Rx) e quello di uplink (Tx) in modalità *IDLE.*
I valori desiderati sono convertiti nei formati a virgola fissa nei seguenti modi:

$$NLP\_Gr\_DT\ =\ round\ \{10^{(0.05*Grx\_DT\_dB)}\ *\ 2^{23}\}$$

A un valore di $Grx\_DT\_dB\ =-3\ dB$ corrisponde $NLP\_Gr\_DT\ =\ 0x5A9DF8$

$$NLP\_Gs\_DT\ =\ round\ \{10^{(0.05*Gtx\_DT\_dB)}\ *\ 2^{23}\}$$

A un valore di $Gtx\_DT\_dB\ =-3\ dB$ corrisponde $NLP\_Gs\_DT\ =\ 0x5A9DF8$

$$NLP\_Gs\_FE\ =\ round\ \{10^{(0.05*Gtx\_FE\_dB)}\ *\ 2^{23}\}$$

A un valore di $Gtx\_FE\_dB\ =-15\ dB$ corrisponde $NLP\_Gs\_FE\ =\ 0x4026E7$

$$NLP\_G\_IDLE\ =\ round\ \{10^{(0.05*G\_IDLE\_dB)}\ *\ 2^{23}\}$$

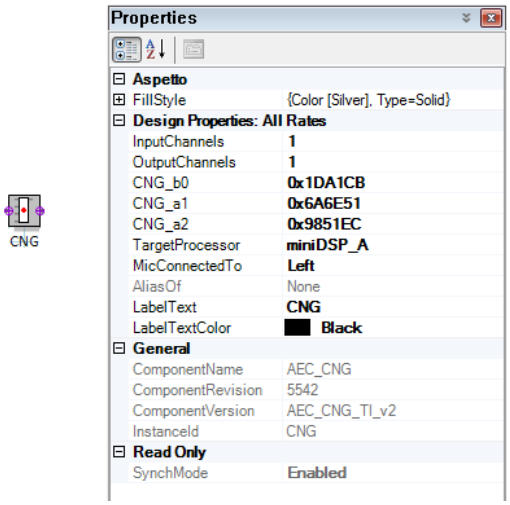
A un valore di $G\_IDLE\_dB\ =-3\ dB$ corrisponde $NLP\_G\_IDLE\ =\ 0x5A9DF8$

Una linea guida per orientarsi nel dimensionare inizialmente questi parametri di attenuazione è mostrata in *Tabella 6.2*, dove i valori rappresentati sono da considerarsi puramente indicativi.

Terminal Type	Attenuation Level in double-talk		Attenuation Level in far-end only
	Rx	Tx	Tx
1 (full-duplex)	≤ 3 dB	≤ 3 dB	6 ~ 15 dB
2a (partial-duplex)	≤ 5 dB	≤ 6 dB	~ 24 dB
2b (partial-duplex)	≤ 8 dB	≤ 9 dB	~ 36 dB
2c (partial-duplex)	≤ 10 dB	≤ 12 dB	~ 48 dB
3 (half-duplex)	> 10 dB	> 12 dB	~ 60 dB

Tabella 6.2: Linea guida dei livelli di attenuazione

"CNG" Comfort Noise Generator



Il CNG è gestito dalle risorse del miniDSP_A e i parametri sui quali si può agire sono: "*CNG b0*", "*CNG a1*" e "*CNG a2*". Questi parametri, modificabili in modalità runtime, definiscono la caratteristica di rumore che andiamo a generare per mascherare i transitori dell'NLP Controller e per eliminare l'effetto "dead air": descrivono il comportamento di un filtro passa-tutto del 2° ordine caratterizzato dalla seguente equazione:

$$y(n) = b0 * x(n) + a1 * y(n-1) + a2 * y(n-2)$$

Operativamente, per generare del rumore rosa, la soluzione ottimale di dimensionamento di questi parametri, in banda fonica con il sistema analizzato, prevede di generare del rumore con un picco alla frequenza di 500 Hz e un polo di raggio 0,9 alla frequenza di 8 kHz:

$$b0 = 0.2315$$
$$a1 = 2 * 0.9 * cos\left(2 * \pi * \frac{500}{8000}\right)$$
$$a2 = -(0.9)^2$$

I cui valori rappresentati in virgola fissa sono:

$$b0_{fix} = round\{b0 * 2^{23}\} = 0x1DA1CA$$

$$a1_{fix} = round\{a1 * 2^{22}\} = 0x6A6E51$$

$$a2_{fix} = round\{a2 * 2^{23}\} = 0x9851EC$$

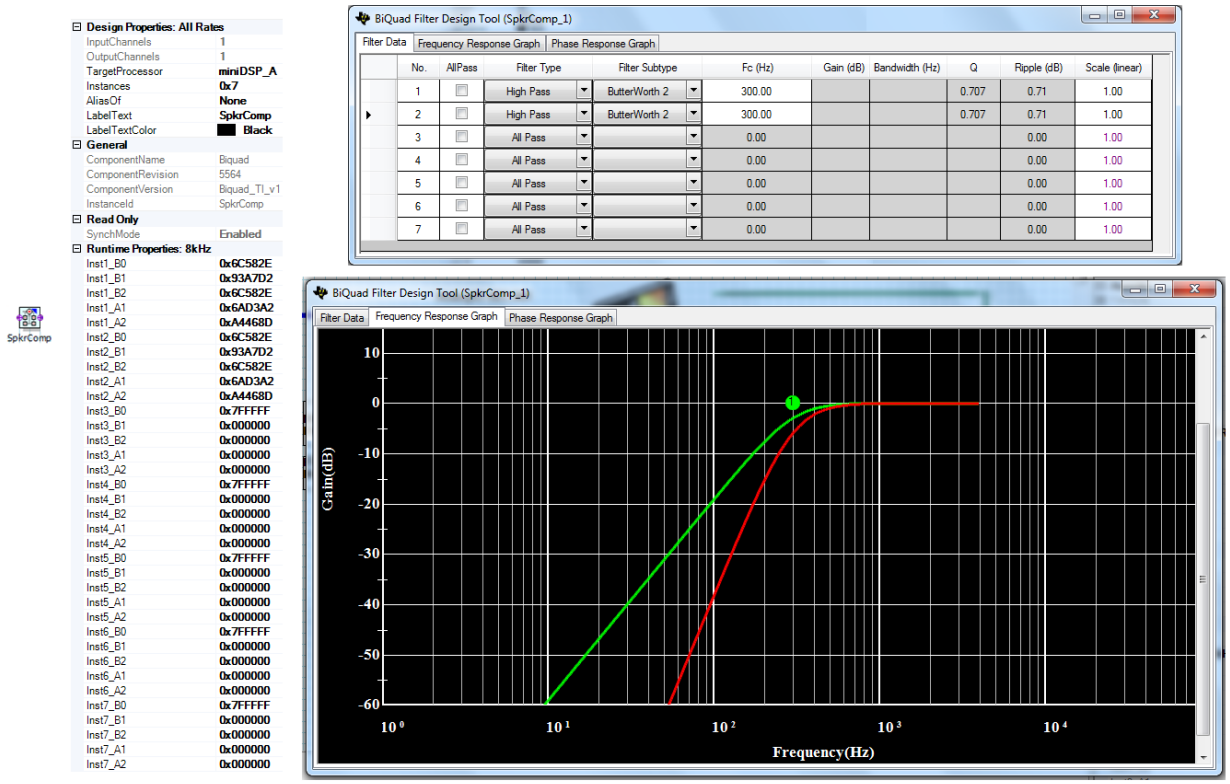
L'applicazione del CNG introduce inevitabilmente un disturbo in continua che deve essere necessariamente bilanciato intervenendo su 3 parametri presenti nel decimatore al quale è collegato il microfono, questi modellano un filtro passa-alto che rimuove tale disturbo in continua, il setting di questi parametri è il seguente:

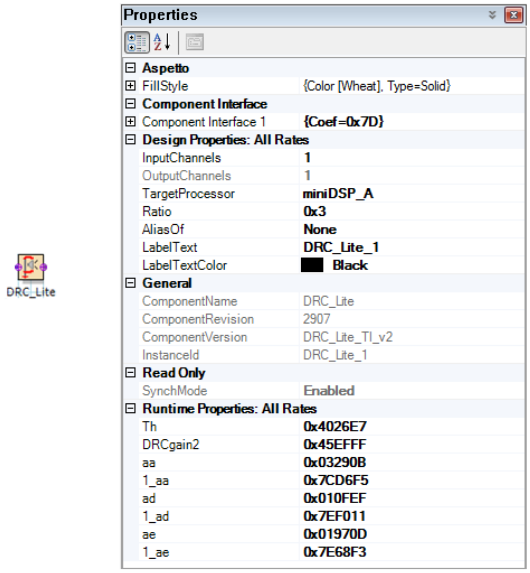
$$AGC\_HPF\_n0 = 0x7B2900$$

$$AGC\_HPF\_n1 = 0x84D700$$

$$AGC\_HPF\_d1 = 0x765200$$

"*SpkrComp*" e "*DRC_Lite*" Equalizzazione per lo Speaker





Lo "SpkrComp" è gestito dalle risorse del miniDSP_A e implementa l'iterazione di n ("*Instances*") filtri IIR in cascata, in particolare questo componente è predisposto alla progettazione di filtri digitale biquadratici che sono filtri del secondo ordine, lineari e ricorsivi, la cui funzione di trasferimento nel dominio Z presenta due poli e due zeri:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

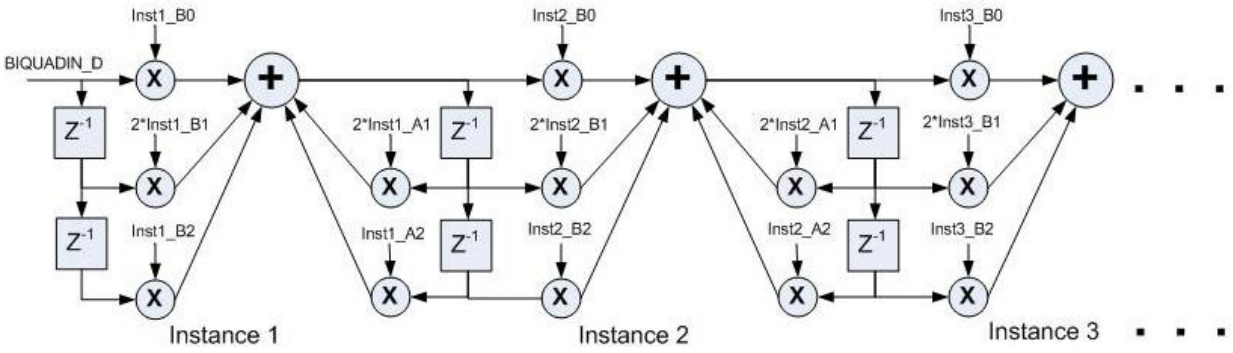


Figura 6.18: Schema a blocchi filtro biquadratico

Scelto dove allocare poli e zeri è sufficiente inserire i coefficienti rispettivi facendo attenzione (*Figura 6.18*) che i coefficienti a_1 e b_1 delle varie iterazioni vengono di default duplicati pertanto vanno inseriti dimezzati rispetto il valore desiderato. Il dispositivo è dotato di un'utility che ne setta automaticamente poli, zeri e quindi i coefficienti a seconda della tipologia di filtro che si desidera realizzare, inoltre visualizza graficamente la caratteristica del guadagno e fase in frequenza.

Il *DRC_Lite* può essere gestito dalle risorse di entrambi i miniDSP, analizzando l'ampiezza di un segnale di controllo fornisce una regolazione automatica della dinamica di un segnale audio presente in ingresso con l'obiettivo di contenerla entro dei limiti specifici. Viene programmata una soglia, al di sotto della quale, il segnale d'uscita segue fedelmente quello in ingresso con un rapporto $1:1$, superata tale soglia, invece, il segnale d'uscita viene riprodotto rispetto il segnale d'ingresso con un rapporto di compressione $m:1$ con $m = 2,4\ e\ 8$ configurabile. Viene stimata, inizialmente, la media del segnale audio in ingresso, *Average Absolute Value* (AAV): una costante di tempo t_{energy} espressa in secondi $[s]$, permette di configurare l'estensione della finestra temporale che si osserva ai fini del calcolo del AAV, questa costante a una determinata frequenza di campionamento F_s è legata al parametro settabile a_e dalla seguente relazione:

$$t_{energy} = -\frac{1}{F_s \ln(1 - a_e)}$$

Successivamente si attua il controllo di compressione: si analizzano i livelli del segnale d'ingresso, in funzione di una soglia definita dal parametro "*Th*", e si calcola un opportuno coefficiente di guadagno da passare al controllo successivo l'*attack/decay time*. Fintanto che il segnale d'ingresso $x(n)$ rimane al di sotto della soglia il guadagno g rimane unitario non appena la soglia viene superata il guadagno utilizzato è il seguente:

$$g(n) = \frac{1}{x(n)^{\left(1-\frac{1}{m}\right)}}\ 10^{\left(1-\frac{1}{m}\right)\frac{Th\_dB}{20}}$$

con $x(n)$ l'ingresso, m il rapporto di compressione e Th_dB il valore della soglia espressa in $[dB]$, regolabile tra $-24\ dB$ e $0\ dB$. L'uscita sarà data da $y(n) = g * x(n)$; a livello di parametrizzazione la prima parte di g viene calcolata ad ogni campione automaticamente all'interno del componente, mentre la seconda parte è una costante configurabile tramite il

parametro "*DRCgain2*". Ogni qualvolta si inseriscono dei parametri, il PPS automaticamente, va a troncarli in funzione della lunghezza della parola binaria che può essere contenuta in memoria, la quale dipende, chiaramente, dal processore che si sta utilizzando. L'ultimo controllo è l'*attack/decay time*, il quale regola i tempi di transizione (*Figura 6.19*) alle variazioni del guadagno g calcolato precedentemente, le relazioni che legano il tempo di *attack* (compressione del guadagno) e il tempo di *decay* (espansione del guadagno) rispettivamente ai parametri "a_a" e "a_d" sono:

$$t_{attack} = -\frac{1}{F_s \ln(1 - a_a)}$$

$$t_{decay} = -\frac{1}{F_s \ln(1 - a_d)}$$

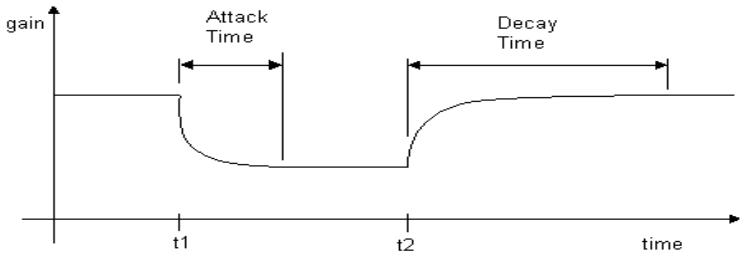
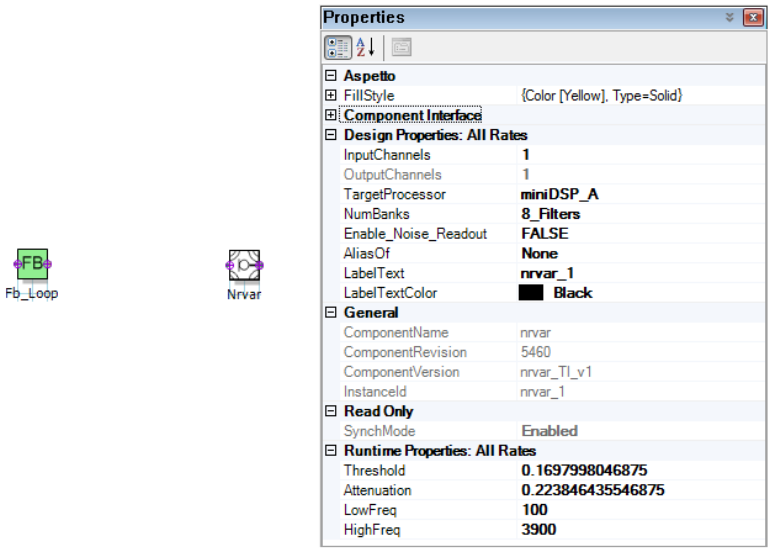


Figura 6.19: Tempi di regolazione del DRC

"*Fb_Loop*" e "*Nrvar*"



Questi componenti sono gestiti rispettivamente dalle risorse del miniDSP_D e miniDSP_A. Il *Fb_Loop* è necessario prima di un componente, che per sua natura introduce dei ritardi, come per esempio un inter-processore: gestisce un anello di retroazione nel flusso di progetto e in assenza di tale componente verrebbe generato un errore in fase di compilazione.

Il *Nrvar* è un filtro multibanda che suddivide la banda audio in diverse sottobande tante quante il valore indicate nel parametro "*NumBanks*", e ad ogni sottobanda sono poi applicate un'attenuazione di rumore e una soglia variabili. Un'interfaccia grafica permette la configurazione della soglia di distinzione tra segnale e rumore, dell'attenuazione e della banda di frequenze da considerarsi.

6.8 Procedura Operativa di Setting

Programmata la scheda, la configurazione in modalità runtime, per la cancellazione di eco è abbastanza immediata, tuttavia la configurazione del dispositivo ai fini di avere un buon rapporto segnale rumore, non è per niente semplice; senza contare poi, che nello stesso impianto, touchscreen o videocitofoni differenti presentano quantità di rumore in linea completamente diverse. Se infine consideriamo un nuovo impianto, il valore intrinseco dei parametri è qualcosa di davvero poco conto, invece è ben più utile indicare una procedura da seguire, ai fini del setting valida in generale, che si svincoli dal caso specifico, ripetibile e riproducibile, da attuare su ciascuna coppia di terminali audio del sistema.

- Programmata la scheda, si disabilitino tutti i parametri relativi l'AEC, nel seguente modo:

Faster Step Size (mu_step)	0x000000
Slower Step Size (mu_step_slow)	0x000000
CC threshold (CC Threshold)	0x000000
Coupling Loss (CL)	-
NLP Rx Attenuation in DT (NLP_Gr_DT)	0x7FFFFF
NLP Tx Attenuation in DT (NLP_Gs_DT)	0x7FFFFF
NLP Tx Attenuation in FE (NLP_Gs_FE)	0x7FFFFF
NLP Rx/Tx Attenuation in IDLE (NLP_G_IDLE)	0x7FFFFF
CNG filter B0 (CNG_b0)	0x000000
CNG filter A1 (CNG_a1)	-
CNG filter A2 (CNG_a2)	-

Tabella 6.3: Parametri AEC disabilitati

- Configurare il guadagno del DAC e del PGA (Control Software) affinché non sia udibile rumore lato speaker, tipicamente è opportuno settarlo, inizialmente, a circa $-6\ dB$.

- Configurare il guadagno dell'ADC e del PGA (Control Software) affinché il livello di segnale in uscita verso il Far End (rilevabile anche dall'uscita dell'$I2S$) sia sufficientemente elevato, tipicamente è opportuno settarlo, inizialmente, a circa $30\ dB$.

- Abilitare il parametro "*mu step*" cioè il fast step size del componente AEC al valore $0x199900$.

- Testare la comunicazione, in un ambiente silenzioso, senza parlato al lato Near End, e osservare la componente di eco, verso il Far End, se diverge: nell'eventualità lo facesse, ridurre a piccole variazioni il parametro "*mu step*" e ripetere questa operazione fintanto che l'eco non diverge.

- Abilitare il parametro "*CL*" Coupling Loss dell'AEC al valore $0x020000$, successivamente in questa procedura, torneremo a modificarlo ai fini della qualità della comunicazione.

- Abilitare i parametri dell'NLP Controller: impostando "*NLP_Gr_DT*", "*NLP_Gs_DT*" e "*NLP_G_IDLE*" a $0x5a9e00$, "*NLP_Gs_FE*" a $0x402700$ e "*CC*" a $0xa00000$.

- Testare la comunicazione, in un ambiente silenzioso, senza parlato al lato Near End e osservare la componente di eco residuo, verso il Far End: se risulta significativa, aumentare a piccole variazioni il parametro "*NLP_Gs_FE*" e ripetere questa operazione fintanto che l'eco residuo risulta inudibile o al massimo accettabile.

- Testare la comunicazione in modalità Double Talk, cioè parlando anche dal lato Near End e osservo l'eco verso il Far End: se il parlato del Near End viene udito lato Far End, in modo increspato o con buchi nella riproduzione Double Talk, decrementare il valore del parametro "*CL*", invece se persiste dell'eco residuo questo valore va aumentato. La soluzione tipicamente è un compromesso fra qualità del parlato trasmesso e livello dell'eco residuo in Double Talk.

- Terminato il setting del "*CL*", se l'eco residuo risulta ancora considerevole, intervenire, a piccole variazioni, sui parametri "*NLP_Gr_DT*" e"*NLP_Gs_DT*" incrementandoli: rappresentiamo alcuni esempi di applicazione:

	Attenuazione in DT		Attenuazione in FE
	Rx	Tx	Tx
(full-duplex)	0x5A9DF8 (-3dB)	0x5A9DF8 (-3dB)	0x4026E7 (-6dB)
(partial-duplex)	0x47FACD (-5dB)	0x4026E7 (-6dB)	0x081385 (-2dB)
(partial-duplex)	0x32F52D (-8dB)	0x2D6A86 (-9dB)	0x020756 (-36 dB)
(partial-duplex)	0x287A27 (-10dB)	0x2026F3 (-12dB)	0x008274 (-48 dB)
(half-duplex)	0x7FFFFF (0dB)	0x000000	0x000000

Tabella 6.4: Parametri AEC di attenuazione

- Testare accuratamente la minima variazione del parametro "mu step slow" in quanto può migliorare o degradare la comunicazione in Double Talk.

- Abilitare i parametri del CNG: "*b0*", "*a1*" e "*a2*" rispettivamente $0x213F00$, $0x6A6E51$ e $0X9851EC$.
  Il livello di rumore del CNG viene controllato prevalentemente dal parametro "*b0*".

Conclusioni

Per il progetto realizzato, si possono osservare le percentuali di utilizzo delle risorse a disposizione: per le capacità computazionali della board utilizzata, il progetto visto, rappresenta un'applicazione limite, in quanto la sola introduzione di un ulteriore filtro o di una semplice elaborazione di qualsiasi genere non sarebbe possibile perché non riuscirebbe a venire gestita.

Resource	Usage
miniDSP_A_coeff (words)	236 (92,19%)
miniDSP_A_cycles	740 (12,04%)
miniDSP_A_cycles_alloc	6144 (100,00%)
miniDSP_A_data (words)	339 (37,83%)
miniDSP_A_instr (words)	815 (79,59%)
miniDSP_A_instr_alloc (words)	833 (81,35%)
miniDSP_D_coeff (words)	389 (76,42%)
miniDSP_D_cycles	2933 (47,74%)
miniDSP_D_cycles_alloc	6144 (100,00%)
miniDSP_D_data (words)	456 (51,01%)
miniDSP_D_instr (words)	975 (95,21%)
miniDSP_D_instr_alloc (words)	1010 (98,63%)

Dai test operativi dell'AEC, si è riscontrata una discreta e immediata risposta in termini di cancellazione dell'eco, risulta tuttavia sempre presente una componente seppur minima di rumore di background che in alcuni casi, rende instabile la comunicazione.
A questo proposito, ritengo che in fase di sviluppo e perfezionamento futuro del lavoro svolto, sia necessario operare cercando un compromesso in termini di setting, fra i parametri del CNG e dell'AGC, facendo attenzione di ottimizzare il filtro multibanda presente nel percorso di uplink.
A livello di configurazione, risulta oneroso, a seconda dell'impianto in considerazione, trovare una soluzione di compromesso, valida in generale, bensì volta per volta è necessario effettuare una regolazione fine dei parametri affinché si ottengano prestazioni soddisfacenti. Questa difficoltà è legata alla forte sensibilità dell'applicativo al rumore termico dei dispositivi connessi all'impianto.
Le limitate capacità computazionali della board, impediscono di poter garantire la cancellazione di echi la cui tempistica superi i *32ms*, per questo motivo si è vagliata l'esigenza di considerare a livello di sviluppo l'impiego di una board più prestante la T.I. TMX320C5505 eZDSP.
Inoltre, è stata vagliata una strada alternativa: quella di gestire, interamente lato software, la comunicazione mediante un algoritmo, implementato nel codice del software dei touchscreen. A grandi linee questo algoritmo analizzando i livelli dei segnali microfonico e altoparlante lato Near-End trasforma la comunicazione in Half-Duplex. Si sono ottimizzati i tempi di attivazione sia del percorso di uplink che di downlink, privilegiando, in concomitanza di comunicazione, il percorso di uplink (da touchscreen verso videocitofono), tale specifica è dettata dal fatto che l'installazione dei touchscreen tipicamente è in ambienti chiusi poco soggetti a disturbi di rumore ambientale mentre l'installazione dei videocitofoni avviene all'aperto con forti componenti di rumore ambientale.
Tale soluzione ai fini della cancellazione d'eco è tuttavia un artifizio, interessante però a livello di costo in quanto non richiede l'impiego di hardware dedicato.

Il lavoro svolto mi ha permesso di acquisire competenze che spaziano dall'analisi e dall'applicazione di filtri digitali, alla struttura di impianti domotici.
Mi sono state di grande aiuto l'esperienza e la professionalità dei colleghi, che mi hanno aiutato nelle varie fasi di tirocinio dalle misure di analisi del sistema fino ai test operativi, a cui va la mia gratitudine e riconoscenza.

Bibliografia

[1] Pietro Di Mascolo – *Fonica*

[2] Carlo Drioli, Nicola Orio – *Elementi di Acustica e Psicoacustica*

[3] Francesco Mangione – *Costruire il suono, manuale pratico per musicisti e sound engineers*

[4] Gian Antonio Mian – *Elaborazione Numerica Dei Segnali*

[5] Donato Masci – *Parametri fisici dell'acustica ambientale*

[6] Amaldi – *Onde e Induzione Elettromagnetica*

[7] Simone Secchi – *La qualità acustica delle sale*

[8] RCF – *Guida alla progettazione e all'installazione dei sistemi di diffusione sonora*

[9] Paolo Annibale – *Cancellazione dell'eco acustica mediante filtraggio adattativo LMS e decomposizione del segnale in sottobande*

[10] Danilo Comminiello – *La cancellazione di eco acustica*

[11] Livio Tenze – *Filtri Adattativi*

[12] Radhika Chinaboina – *Adaptive Algorithms For Acoustic Echo Cancellation in Speech Processing*

[13] Mohamed Al Mahdi Eshtawie – *An Algorithm Proposed for FIR Filter Coefficients Representation*

[14] Elena Punskaya – *Design of FIR Filters*

[15] Alessandro Bardine – *Filtraggio Digitale*

Appendice A:

Scripts in MATLAB

```
%NLMS (Normalized Least Mean Square) Algoritmo Adattivo FIR
%Rel. 4.0 14/10/2011
%Celin Patrik
%------------------------------------------------------------------------
%Input
%x_in : segnale/vettore d'ingresso              (Segnale allo SPK)
%v_in : segnale/vettore d'eco                   (Eco&Noise al MIC)
%s_in : segnale/vettore utile                   (Segnale utile al MIC)
%d_in : segnale/vettore in ricezione: v_in + s_in (Eco&Noise + utile al MIC)
%N    : lunghezza del filtro FIR (numero di tappi/coefficienti del filtro)
%      (ordine del filtro è pari a N-1)
%Output
%errore_exit : segnale/vettore d'errore
%OutFIR      : segnale/vettore d'uscita del FIR
%Variabili Locali
%x          : vettore d'ingresso di lunghezza utile   (Segnale allo SPK)
%d          : vettore in ricezione di lunghezza utile (Segnale al MIC)
%h          : vettore con gli N coefficienti del filtro
%y          : vettore d'uscita del filtro FIR
%e          : vettore d'errore d(n)-y(n)
%camp_consid : vettore di sottocampioni considerati di lunghezza pari a N
%------------------------------------------------------------------------
function [e y Ee_s]= NLMS (x_in,v_in,s_in,N)
warning off all; % disattivazione dei warning sul clipping dei file wave e log 0
fk   = 8000;    % Frequenza di Sampling 8000 campioni al secondo,
%------------------------------------------------------------------------
%Acquisizione Audio dei file
%------------------------------------------------------------------------
%x_in = wavread(x_in);
%v_in = wavread(v_in);
%s_in = wavread(s_in);
%------------------------------------------------------------------------
%Controllo lunghezza utile dei file
%------------------------------------------------------------------------
d_in = v_in + s_in;
xtemp = 0;
dtemp = 0;
for (i=1:1:length(x_in))
   if x_in(i)~=0   %trova il momento in cui si annulla per tagliarlo
      xtemp=i;
   end
end
for (t=1:1:length(d_in))
```

```
   if d_in(t)~=0   %trova il momento in cui si annulla per tagliarlo
      dtemp=t;
   end
end
%--------------------------------------------------------------------------
%Adattam. della lunghezza utile dei file in base alla più corta tra x e d
%--------------------------------------------------------------------------
if xtemp > dtemp %se d_in è il segnale più corto
   M = dtemp;   %M il numero di campioni è pari al numero di camp di d_in
else           %se x_in è il segnale più corto
   M = xtemp;   %M il numero di campioni è pari al numero di camp di x_in
end
x = x_in(1:M).';
d = d_in(1:M).';
if s_in == 0
   s = zeros(1,M);
else
   s = s_in(1:M);
end
%--------------------------------------------------------------------------
%Test dell'algoritmo nlms già implementato in Matlab
%--------------------------------------------------------------------------
ha    = adaptfilt.nlms(N,1,1,1); %l'ultimo parametro è l'offset per evitare div*0
[Out_T,Err_T] = filter(ha,x,d);
%--------------------------------------------------------------------------

[y e h tot]       = CORE(x,d,M,N);      %CORE NLMS
Ee_s              = sum(abs(e-s).^2);    %Verifica l'energia di e-s
[massimi ind_maxs] = max(tot');
hmax              = max(massimi)
%--------------------------------------------------------------------------
%Debug di Cross Correlation
%--------------------------------------------------------------------------
% xc = (xcorr(e,s));
% [value_xcmax,position_xcmax] = max(xc); %Guardo il massimo della cross corr
% offset = position_xcmax-M;
% offset_time = offset/fk;
% for i=1:M
%    if (i+offset)<1
%       e_corr(i)=0;
%    else
%       if (i+offset)<M+1
%         e_corr(i)=e(i+offset);
%       else
%         e_corr(i)=0;
%       end
%    end
% end
% Eecorr_s    = sum(abs(e_corr-s).^2);
%--------------------------------------------------------------------------
%Visualizzazione Grafica Ingresso
%--------------------------------------------------------------------------
Tmax = M / fk;                % Durata totale = num di campioni/fk
deltaT = Tmax / M;              % DeltaT
step_graf = 0:deltaT:Tmax-deltaT; %indice per visualizzare i grafici in [s]
ampiezza = 1.44;
subplot(3,1,1);
plot(step_graf*1000,x,'m');
title('Segnali dell''NLMS');
```

```
legend('x');
xlabel('Time [ms]');
ylabel('[mV]');
grid on;
%--------------------------------------------------------------------------
%Visualizzazione Grafica Desiderato e Uscita
%--------------------------------------------------------------------------
subplot(3,1,2);
plot(step_graf*1000,d*1000,'b',step_graf*1000,y*1000,'g');
legend('d','y');
% plot(step_graf*1000,d*1000,'b')
% legend('d');
%axis([0 Tmax*1000 -20 20]);
xlabel('Time [ms]');
ylabel('[mV]');
grid on;
%--------------------------------------------------------------------------
%Visualizzazione Grafica Segnale Originale
%--------------------------------------------------------------------------
% subplot(3,1,1);
% plot(step_graf*1000,s*1000,'k');
% axis([0 Tmax*1000 -200 200]);
% legend('s');
% xlabel('Time [s]');
% ylabel('V');
% grid on;
%--------------------------------------------------------------------------
%Visualizzazione Grafica Errori (Linear Scale)
%--------------------------------------------------------------------------
% subplot(3,1,3);
% plot(step_graf*1000,e*1000,'r');                %Errore
% axis([0 Tmax*1000 -5 5]);
% legend('e nlms');
% xlabel('Time [ms]');
% ylabel('[mV]');
% grid on;

% subplot(3,1,2);
% plot(xc*1000,'r');                %Crosscorrelation
% axis([1 1439 -45 45]);
% legend('xcorrelation e,s');
% xlabel('Time [ms]');
% grid on;

subplot(3,1,1);
plot(step_graf*1000,e*1000,'r',step_graf*1000,1000*s);%Errore e utile
%axis([Tmax*1000-500 Tmax*1000 -250 250]);
legend('e','s');
xlabel('Time [ms]');
ylabel('[mV]');
grid on;

% Errore, Utile e Errore corretto in fase

% subplot(3,1,3);
% plot(step_graf*1000,e*1000,'r',step_graf*1000,s*1000,step_graf*1000,e_corr*1000);
% axis([0 Tmax*1000 -10 10]);
% legend('e','s','e corr');
% xlabel('Time [ms]');
```

```
% ylabel('[mV]');
% grid on;

% subplot(3,1,1);
% plot(step_graf*1000,e*1000,'r',step_graf*1000,Err_T*1000); %Errore e errore Matlab
% axis([0 Tmax*1000 -15 15]);
% legend('e','Err T nlms Matlab');
% xlabel('Time [ms]');
% ylabel('[mV]');
% grid on;

% subplot(3,1,3);
% plot(step_graf*1000,1000*(Err_T-e.'),'r');
% axis([0 Tmax*1000 -15 15]);
% legend('Err_T-e');
% xlabel('Time [ms]');
% ylabel('[mV]');
% grid on;

% subplot(3,1,1);
% plot(step_graf*1000,abs(e-s),'m');
% %axis([0 Tmax*1000 0 12]);
% legend('|e-s|');
% xlabel('Time [ms]');
% ylabel('[mV]');
% grid on;

% Errore corretto in fase - s

% subplot(3,1,2);
% plot(step_graf*1000,1000*abs(e_corr-s),'m');
% axis([0 Tmax*1000 0 20]);
% legend('|e corr-s|');
% xlabel('Time [ms]');
% ylabel('[mV]');
% grid on;

% subplot(3,1,3);
% plot(step_graf*1000,(1000*abs(e-s)).^2,'m');  % MSE [mV]^2
% legend('|e-s|^2');
% axis([0 Tmax*1000 0 150]);
% xlabel('Time [ms]');
% ylabel('[mV]^2');
% grid on;
%--------------------------------------------------------------------------
%Visualizzazione Grafica Errori (Logarithmic Scale)
%--------------------------------------------------------------------------
% subplot(3,1,1);
% plot(step_graf*1000,20*log10(abs(e*1000)),'r'); %Errore
% axis([0 Tmax*1000 -40 15]);
% legend('|e|');
% xlabel('Time [ms]');
% ylabel('[dBm]');
% grid on;

% subplot(3,1,2);
% plot(step_graf*1000,40*log10(abs(e*1000)),'r'); %Errore quadratico medio
% axis([0 Tmax*1000 -100 20]);
% legend('|e|^2');
```

```
% xlabel('Time [ms]');
% ylabel('[dBm]');
% grid on;

% subplot(3,1,3);
% plot(step_graf*1000,40*log10(abs(e*1000)),'r',step_graf*1000,40*log10(abs(Err_T*1000)),'y');
% axis([0 Tmax*1000 -80 50]);
% legend('|e|^2','|Err_T|^2');
% xlabel('Time [ms]');
% ylabel('[dBm]');
% grid on;

% subplot(3,1,1);
% plot(step_graf*1000,20*log10(abs(e-s)),'r'); %Diff. errore-utile
% axis([0 Tmax*1000 -120 -40]);
% legend('|e-s|');
% xlabel('Time [ms]');
% ylabel('[dB]');
% grid on;

% subplot(3,1,1);
% plot(step_graf*1000,20*log10(1000*abs(e-s)),'r'); %Diff. errore-utile
% axis([0 Tmax*1000 -40 15]);
% legend('|e-s|');
% xlabel('Time [ms]');
% ylabel('[dBm]');
% grid on;

% subplot(3,1,2);
% plot(step_graf*1000,40*log10(abs(e-s)),'r');   %MSE [dB]
% axis([0 Tmax*1000 -170 -40]);
% legend('|e-s|^2');
% xlabel('Time [ms]');
% ylabel('[dB]');
% grid on;
% grid minor;

% subplot(3,1,3);
% plot(step_graf*1000,40*log10(1000*abs(e-s)),'r');%MSE [dBm]
% axis([0 Tmax*1000 -100 20]);
% legend('|e-s|^2');
% xlabel('Time [ms]');
% ylabel('[dBm]');
% grid on;

% subplot(3,1,2);
% semilogy(step_graf*1000,((1000*abs(e-s)).^2),'r'); %MSE [Semilogarithmic]
% axis([0 Tmax*1000 10^(-6) 10^(1)]);
% legend('|e-s|^2');
% xlabel('Time [ms]');
% ylabel('[mV]^2');
% grid on;
%--------------------------------------------------------------------------
%Visualizzazione Grafica Coefficienti del filtro
%--------------------------------------------------------------------------
subplot(3,1,3);
stem([h.',ha.coefficients.']);
%stem(h.');
%stem(ha.coefficients.');
```

```
%axis([1 N -0.008 0.008]);
legend('N Coeff di h','N Coeff di ha');
xlabel('n°Coefficienti');
ylabel('Value Coefficients');
grid minor;
grid on;
%--------------------------------------------------------------------------
%Rendering Audio in wave
%--------------------------------------------------------------------------
% wavwrite (d,fk,'desiderato');
% wavwrite (e,fk,'errore_exit');
% wavwrite (s,fk,'sample_exit');
% wavwrite (Err_T,fk,'Err_T'); %wave dell'errore ottenuto con Matlab
end %end LMS

%--------------------------------------------------------------------------
%ALGORITMO NLMS
%--------------------------------------------------------------------------
function [y e h tot] = CORE(x,d,M,N)
%--------------------------------------------------------------------------
%Inizializzazione vettori dei segnali
%--------------------------------------------------------------------------
y = zeros (1,M);    %inizializza il vettore d'uscita
e = zeros (1,M);    %inizializza il vettore di errore
h = zeros (1,N);    %inizializza il vettore dei coefficienti del filtro FIR
u = zeros (1,M);    %inizializza il vettore degli step-size
tot = zeros (M,N);  %inizializza la matrice traccia dei coefficienti
%--------------------------------------------------------------------------
%CORE DELL'ALGORITMO NLMS
%--------------------------------------------------------------------------
%N : numero di coefficienti
%M : numero totale di campioni
for n = 1:M                          %n varia da 1 a M
   if n < N
      %Inizio la scansione dei primi n campioni, sono meno di N!
      %Li carico in ordine cronologico inverso (dal +recente al +vecchio)
      camp_consid = [x(n:-1:1);zeros(1,N-n).'];%n<N quindi completo con 0
   else
      %Carico una porzione (N) di campioni di M in ordine cronologico inverso
      camp_consid = x(n:-1:n-N+1);          %dal +recente al +vecchio
   end
   tot(n,:) = h;                        % traccia la variazione dei coefficienti
   y(n)    = h * camp_consid;                %segnale d'uscita
   e(n)    = (d(n) - y(n).');                %segnale di errore
   u(n)    = 1/((camp_consid. * camp_consid)+1); %vettore di autocorrelazione
   h       = h + u(n) * e(n) * camp_consid.';    %aggiornamento coefficienti NLMS
end  %end for
end  %end CORE
```

```
% MODELLO DEL SISTEMA
% Rel. 3  03/10/2011
% Celin Patrik
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
function [output] = system (input);
   amp      = 0.0068;
   ampiezza = 1.44;    %adattamento SPK
   tempo    = 0.3;
```

```
   delay    = 0.014; %[s]
   fk       = 8000;   % frequenza di campionamento / n° campioni al [s]
   tk       = 1/fk;
   celle    = delay/tk;
   x = 0:tk:(tempo-tk);  % [s] secondi length (x) = tempo * campioni
%--------------------------------------------------------------------------
% Relazione I/O nel tempo
   output = [zeros(1,celle),amp*input(1:(length(input)-celle))];
   output2 = filter([zeros(1,celle),amp,zeros(1,length(input)-celle-1)],1,input);
%--------------------------------------------------------------------------
% FdT e Bode
%--------------------------------------------------------------------------
   fmin = 300;
   fmax = 16000;
   freq = (fmin:fmax-1);
   resp = amp .* exp(-i*2*pi*freq*delay);
   sys  = frd(resp,freq);
%   subplot(1,1,1);
%   bode(sys);
%   grid on;
%--------------------------------------------------------------------------
%Visualizzazione Grafica I/O
%--------------------------------------------------------------------------
   subplot(2,1,1);
   plot(x*1000,input);
   %axis([0 tempo*1000 -ampiezza ampiezza]);
   legend('input');
   xlabel('[ms]');
   ylabel('[V]');
   grid on;

   subplot(2,1,2);
   plot(x*1000,1000*output,x*1000,1000*output2);
   %axis([0 tempo*1000 -15 15]);
   legend('output','output2');
   xlabel('[ms]');
   ylabel('mV');
   grid on;
%--------------------------------------------------------------------------
%Rendering Audio
%--------------------------------------------------------------------------
%   wavwrite (input,campioni,'input.wav');
end

function fin = rect_fin(fmin,fmax,funct)
   fin = [zeros(1,(fmin)),funct.*ones(1,(fmax-fmin)),zeros(1,(2*fmin))];
end
```

```
% GENERATORE DI FUNZIONI
% Rel. 3  30/09/2011
% Celin Patrik
% ATTENZIONE SIN E RECT FUNZIONANO IN MODO DIVERSO
% NEI SIN  LA f_sin  REGOLA I PERIODI IN UN S
% NEI RECT LA f_rect REGOLA I PERIODI NEL TEMPO TOTALE
% L'amp dello SPK varia tra 2,48V e -2,48V
% L'amp del MIC varia tra 200mV e -200mV con un bias di 2.8V
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
```

```
function [randn_SPK sin_MIC] = genwav
   campioni = 8000;  % frequenza di campionamento / n° campioni al secondo
   %DTMF freq
   fa =  697;  fb = 770; fc = 852; fd =  941;
   fA = 1209; fB = 1336; fC =1477; fD = 1633;
   f_sin = 100 ; % indica il n° di periodi in 1 s
   f_rect = 3;
   tempo = 0.3;                % [s] secondi
   step = 1/campioni;
   x = 0:step:(tempo-step);  % [s] secondi length (x) = tempo * campioni
   ampiezza = 1.44;         % adatt. SPK è in Volt corrisponde al driver matlab 0.005
   %-------------------------------------------------------------------------
   %Costruzione randn
   %-------------------------------------------------------------------------
   randn_signal = randn(1,length(x));
   randn_SPK    = randn_signal;          %adatt. in amp per SPK
   randn_utile  = 0.0068*randn(1,length(x));
   randn_MIC    = 0.1*randn_SPK;              %adatt. in amp per MIC
   randn_DRIVER = 0.005*(2*randn(1,length(x))-1);   %adatt. per pilotare 2 touch
   lpf = fir1 (100, 0.9, hamming(101));
   randn_lpf = filter(lpf,1,4*randn_SPK);
   randn_lpf_MIC = filter(lpf,1,0.0068*randn_SPK);
   %-------------------------------------------------------------------------
   %Costruzione sin
   %-------------------------------------------------------------------------
   sin_signal  = sin(2*pi*f_sin*x);
   sin_SPK     = ampiezza*sin_signal;              %adatt. in amp per SPK
   sin_MIC     = 0.02*sin_signal;                   %adatt. in amp per MIC
   sin_DRIVER  = 0.005 * sin_signal;               %adatt. per pilotare 2 touch
   %sin per la durata di 'tempo' e poi silenzio per un altro 'tempo'
   sin_dead    = ampiezza*[sin_signal,zeros(1,1*length(sin_signal))];
   sin_dead_inv= ampiezza*[zeros(1,1*length(sin_signal)),sin_signal];
   %-------------------------------------------------------------------------
   %Costruzione rect
   %-------------------------------------------------------------------------
%    if f_rect == 1
%       rect_signal = [ones(1,(length(x))/(f_rect*2)),zeros(1,(length(x))/(f_rect*2))];
%    else
%       temp = [ones(1,(length(x))/(f_rect*2)),zeros(1,(length(x))/(f_rect*2))];
%       for t=1:f_rect-1
%          temp = [temp, [ones(1,(length(x))/(f_rect*2)),zeros(1,(length(x))/(f_rect*2))]];
%       end
%       rect_signal = temp;
%    end
%
%    rect_SPK    = ampiezza*rect_signal;          %adatt. in amp per SPK
%    rect_MIC    = 0.0068*rect_SPK;              %adatt. in amp per MIC
%    rect_DRIVER = 0.005*rect_signal;             %adatt. per pilotare 2 touch
%-------------------------------------------------------------------------
%Costruzione Combinazioni
%-------------------------------------------------------------------------
 comb1 = randn_SPK + sin_SPK;
 comb2 = (ampiezza/3)*(sin(2*pi*2*f_sin*x)+sin(2*pi*3*f_sin*x)+sin(2*pi*5*f_sin*x));
 comb3 = (ampiezza/3)*(sin(2*pi*fa*x)+sin(2*pi*fc*x)+sin(2*pi*fA*x));

%-------------------------------------------------------------------------
%Visualizzazione Grafica sin
%-------------------------------------------------------------------------
%    subplot(3,1,1);
```

```
%    plot(x*1000,sin_signal);
%    axis([0 tempo*1000 -1 1]);
%    legend('sin signal');
%    xlabel('[ms]');
%    ylabel('Signal Value');
%    grid on;

%    subplot(3,1,2);
%    plot(x,sin_SPK);
%    axis([0 tempo -1.5 1.5]);
%    legend('sin SPK');
%    xlabel('[s]');
%    ylabel('Signal Value');
%    grid on;

   subplot(3,1,2);
   plot(x*1000,sin_MIC*1000);
   %axis([0 tempo*1000 -200 200]);
   legend('sin MIC');
   xlabel('[ms]');
   ylabel('mV');
   grid on;

%    subplot(3,1,1);
%    plot(x*1000,sin_DRIVER);
%    axis([0 tempo*1000 -0.005 0.005]);
%    legend('sin DRIVER');
%    xlabel('[ms]');
%    ylabel('Signal Value');
%    grid on;

%    subplot(3,1,1);
%    plot(sin_dead_inv);
%    %axis([0 1440 -0.005 0.005]);
%    legend('sin dead');
%    xlabel('[campioni]');
%    ylabel('Signal Value');
%    grid on;

%    subplot(3,1,2);
%    plot(x*1000,sin_signal);
%    axis([0 1000/f_sin -1 1]);
%    legend('sin signal zoom');
%    xlabel('[ms]');
%    ylabel('Signal Value');
%    grid on;

%    subplot(3,1,2);
%    plot(x*1000,randn_lpf);
%    axis([0 tempo*1000 -1.5 1.5]);
%    legend('randn lpf');
%    xlabel('[ms]');
%    ylabel('Signal Value');
%    grid on;
%------------------------------------------------------------------------
%Visualizzazione Grafica rect
%------------------------------------------------------------------------
%    subplot(3,1,2);
%    plot(0:step:(tempo-step),rect_signal);
```

```
%    axis([0 tempo 0 1.5]);
%    legend('rect signal');
%    xlabel('[s]');
%    ylabel('Signal Value');
%    grid on;
%
%    subplot(3,1,2);
%    plot((0:step:(tempo-step))*1000,rect_signal);
%    axis([0 1000*tempo/f_rect 0 1.5]);
%    legend('rect signal zoom');
%    xlabel('[ms]');
%    ylabel('Signal Value');
%    grid on;

%    subplot(3,1,2);
%    plot(x*1000,randn_signal);
%    punti = 200;                % scelgo di osservare 200 punti nel grafico
%    axis([0 1000*step*punti -3 3]);
%    legend('randn signal zoom');
%    xlabel('[ms]');
%    ylabel('Signal Value');
%    grid on
%--------------------------------------------------------------------------
%Visualizzazione Grafica randn
%--------------------------------------------------------------------------
%    subplot(3,1,1);
%    plot(x*1000,randn_signal);
%    axis([0 tempo*1000 -2.5 2.5]);
%    legend('randn signal');
%    xlabel('[ms]');
%    ylabel('Signal Value');
%    grid on;
   subplot(3,1,1);
   plot(x,randn_SPK);
   axis([0 tempo -2.5 2.5]);
   legend('randn SPK');
   xlabel('[s]');
   ylabel('Signal Value');
   grid on;
%
%    subplot(3,1,3);
%    plot(x*1000,1000*randn_MIC);
%    axis([0 tempo*1000 -12 12]);
%    legend('randn MIC');
%    xlabel('[ms]');
%    ylabel('mV');
%    grid on;
%--------------------------------------------------------------------------
%Rendering Audio
%--------------------------------------------------------------------------
%[signal,F,nbits] = wavread('Single.wav');
%wavwrite (sin_signal,campioni,exit_sin);
%wavwrite (sin_signal2,campioni,exit_sin);
%wavwrite (rect_signal,campioni,exit_rect);
End
```