POLITECNICO DI MILANO

SCUOLA DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Specialistica in
Ingegneria delle Telecomunicazioni
Indirizzo Segnali

SVILUPPO DI UN'APPLICAZIONE DI RICONOSCIMENTO VOCALE LOW COST PER IMPIANTI DOMOTICI

Relatore:
Prof. Carlo Maria Prati

Tesi di Laurea di:
Andrea Laini
Matr. 755233

Anno Accademico 2010/2011

“L’importante è dove sei diretto”

“E tu, dove sei diretto?”

“Ovunque abbia voglia”

dal film Nemico Pubblico

Un particolare ringraziamento al Professor Claudio Prati, relatore di questo elaborato, per la disponibilità e l'attenzione mostrata nel corso dello svolgimento del progetto.

Un grazie sentito ai colleghi del reparto di Ricerca e Sviluppo Elettronica di Gewiss s.p.a., per il supporto e l'aiuto costante in questi mesi di tirocinio: Valerio (a cui va il ringraziamento più grande), Mauro, la coppia Massimo & Massimo (i due elettronici) e Giuseppe.

INDICE

INDICE DELLE FIGURE

INDICE DELLE TABELLE

PREMESSA

La scelta di affrontare la tematica del riconoscimento vocale è stata dettata dalla curiosità di approfondire le conoscenze in questo ramo del processing di segnali e dalla volontà di Gewiss s.p.a. di valutare la possibilità di applicare questi metodi in un contesto quotidiano.

L'intento è stato da subito quello di progettare un'applicazione dimostrativa, che prendesse in considerazione tutti i possibili ambiti di lavoro e applicabilità (in generale un impianto, classico o domotico, casalingo): dalle prove eseguite con quanto realizzato nel corso del progetto, si è valutata la fattibilità di poter creare un possibile futuro dispositivo a basso consumo energetico, basso costo di produzione e che prevedesse un impiego sia in impianti domotici (come quello effettivamente realizzato), quindi con comunicazione su bus, ed uno da integrare nei classici impianti.

Indubbiamente, pur trattandosi di una "novità" (il riconoscimento vocale è diffuso in molti impieghi, basti pensare ai cellulari, alle automobili, ma ancora poco diffuso in situazioni più quotidiane) nel campo dell'impiantistica e dei sistemi domotici, si tratta di un particolare tipo di applicazione che potrebbe faticare a sfondare il mercato: è dimostrata, infatti, la difficoltà con cui gli uomini parlino con le macchine. Riconoscimento di impronte, di retina, dispositivi touch screen, wireless vengono impiegati senza grossi problemi, ma quando si tratta di dare comandi vocali ad un'apparecchiatura, le persone si "fermano", un po' titubanti e quasi con la paura di essere considerati dei pazzi.

Non si può negare, però, come un'applicazione come quella descritta in questo elaborato, se opportunamente studiata, progettata e realizzata, può portare a dei grandi vantaggi, soprattutto per quelle persone che hanno difficoltà motorie, a cui verrebbe permesso di compiere alcune operazioni quotidiane con l'ausilio della voce.

1. STORIA

Realizzare un'apparecchiatura che fosse in grado di simulare il comportamento del linguaggio umano, sia nella capacità di emettere suoni con un preciso significato lessicale, che di capire ed interpretare una particolare parola pronunciata, ha incuriosito ingegneri e scienziati per diversi decenni, soprattutto (in modo più concreto e realizzabile) a partire dall'inizio del secolo scorso.

Infatti, benchè le tecniche di riconoscimento vocale abbiano avuto larga diffusione solamente negli ultimi due decenni, lo studio e l'impiego di questo tipo di applicazioni risale agli anni Venti; l'approccio nei confronti di questi sistemi è stato, quantomeno nei primi tempi, progressivo e graduale, senza grandi innovazioni che cambiassero radicalmente le conoscenze e il campo di applicabilità di dette tecniche di riconoscimento. Da semplici macchine capaci di rispondere ad un numero molto limitato di suoni (singole parole o semplici cifre) si è giunti, con un forte exploit negli Anni Ottanta, a sistemi sofisticati in grado di interpretare ed elaborare il normale parlato umano, fluente, continuo e senza la necessità di pronunciare esclusivamente i soli termini che il sistema può riconoscere, distinguendo anche il parlante in base alle caratteristiche statistiche (del linguaggio) e fisiche (del suono) della frase pronunciata.

Inizialmente, l'interesse degli studiosi non era rivolto alla possibilità di riconoscere e capire delle parole pronunciate, ma di poter realizzare macchine che fossero "capaci di parlare", di produrre suoni che avessero un significato (sempre dal punto di vista del linguaggio umano): praticamente creare sistemi che approssimassero il tratto vocale dell'uomo. Questi tentativi ebbero inizio già nella seconda metà del XVIII secolo, grazie a Christian Kratzestein, professore russo di fisiologia dell'Università di Copenhagen, che nel 1773 riuscì a produrre una vocale usando una cavità di risonanza connessa con le canne di un organo.

Vent'anni più tardi, Wolfgang von Kempelen costruì una "macchina parlante acustica-meccanica", che venne ripresa e migliorata a metà del 1800 da Charles Wheastone, il quale impiegò un risonatore in pelle: in questo modo, era possibile modificarlo e configurarlo manualmente, in modo che il dispositivo potesse produrre differenti suoni che somigliassero a parole.

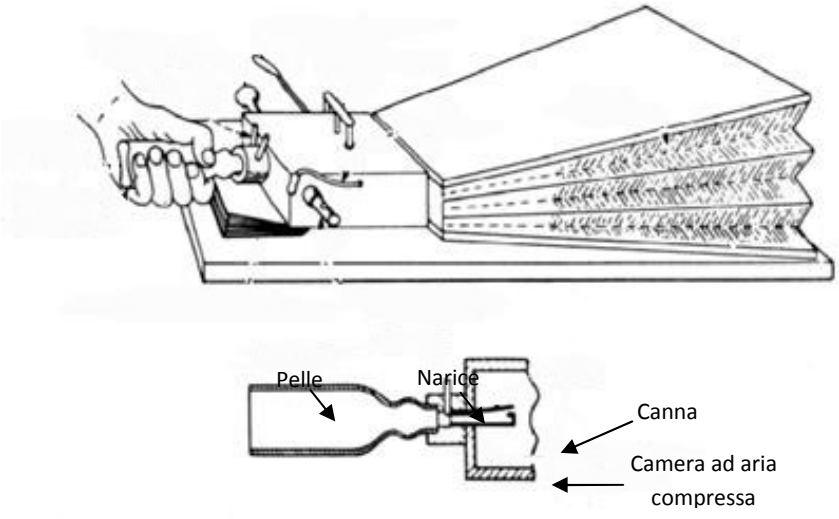


Figura 1.1 Versione di Wheastone della macchina parlante di von Kempeler

Il passo significativo, sia per quanto riguarda i sintetizzatori vocali che per le tecniche di riconoscimento, *fu però segnato dagli studi* (e dai successivi articoli pubblicati su riviste specializzate) *di Fletcher e di altri studiosi dei Bell Laboratories, i quali riuscirono a dimostrare il* ***legame tra lo spettro del parlato e le caratteristiche del suono pronunciato***, ovvero i parametri che permettono all'orecchio umano di interpretare l'onda acustica come un particolare suono. Conseguenza di questa scoperta fu la realizzazione, negli Anni Trenta, del *primo sintetizzatore vocale* di Homer Dudley: il *VODER* (Voice Operating Demonstrator); non era altro che l'equivalente elettrico (ovviamente, con un controllo meccanico) della "macchina parlante" costruita da Wheatstone qualche decennio prima.

I primi tentativi di progettazione di sistemi per il riconoscimento vocale erano per lo più basati sugli studi di Fletcher e sulla teoria acustica-fonetica, la quale è basata sui fonemi, ovvero gli elementi (suoni) elementari che danno origine prima alle parole e poi all'intero linguaggio. In particolare, veniva prestata particolare attenzione al modo in cui queste unità vocali vengono prodotte dal tratto vocale: come verrà chiarito meglio in seguito, la generazione di un'onda acustica è possibile grazie alla vibrazione di una sorgente, che nel caso dell'essere umano è rappresentata dalle corde vocali, la quale dà luogo ad un suono avente dei modi di risonanza naturali; questi ultimi sono definiti formanti o frequenze formanti, e sono zone dello spettro di potenza del segnale acustico in cui l'energia è maggiormente concentrata.

Da considerazioni analoghe a quelle appena descritte, con l'ausilio di conoscenze più fisiche e meccaniche del fenomeno della propagazione delle onde sonore, è nato il primo prototipo di speech

recognition: un semplicissimo giocattolo, denominato ***Radio Rex***. Esso consisteva in un cane di cellulosa con base metallica, in grado di muoversi grazie all'ausilio di una molla (alla quale è collegato) e mantenuto all'interno della sua cuccia per mezzo di un magnete. Il magnete è alimentato da una corrente di energizzazione che scorre all'interno di una barra metallica, costruita in modo tale da formare un ponte; quest'ultimo è sensibile a onde acustiche di frequenza 500 Hz: interrompendo la corrente circolante viene rilasciata la molla, e quindi anche il cane. In particolare, quest'energia è contenuta nella vocale della parola "Rex", cosicchè sia possibile liberare il cane quando viene pronunciato il suo nome.

Figura 1.2 Immagini della Radio Rex

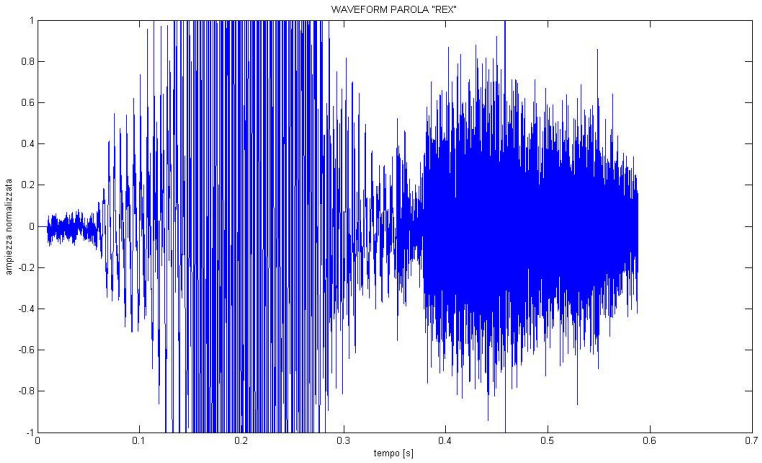


Figura 1.3 Forma d'onda della parole "Rex" (in ascissa il tempo e in ordinata l'ampiezza normalizzata). Il codice Matlab per generare questa immagine è riportato in appendice

Fino agli Anni Sessanta, l'impiego delle tecniche di riconoscimento vocale venne adottato solamente in applicazioni ad hoc: processing ed estrazione delle caratteristiche di semplici segnali, rilevamento dell'energia alle varie frequenze o della frequenza dominante relativamente ad un dizionario molto limitato, come le cifre da 0 a 9, le parole "yes"/"no", le vocali. Si trattava di sistemi testati su un numero molto limitato di utenti, spesso inferiore alla decina, mostrando probabilità di errore generalmente inferiori al 10%.

Nel frattempo iniziarono a svilupparsi le tecniche di analisi impiegate nelle applicazioni attuali, come il *training statistico* e i *modelli di linguaggio* (language modeling).

Nel 1952, presso i Bell Laboratories, Davis, Biddulph e Balashek realizzarono il primo sistema per il riconoscimento di cifre (da 0 a 9), pronunciate singolarmente da un unico utente. La tecnica era fondata sull'analisi delle frequenze formanti misurate (o stimate) durante le regioni delle vocali di ciascuna parola. In particolare, veniva effettuata un'analisi sullo spettro diviso in due bande: frequenze superiori e inferiori a 900 Hz (prima e seconda formante rispettivamente); dai risultati di queste osservazioni, vengono definite delle "traiettorie", ognuna diversa e caratteristica della particolare cifra pronunciata. All'atto del riconoscimento, viene effettuato un confronto tra i parametri (traiettorie) calcolati dal suono e quelli memorizzati per ciascuno dei vocaboli del dizionario: il reference pattern che maggiormente rispecchia quello del suono in ingresso identifica la cifra che (con maggiore probabilità) è stata pronunciata.

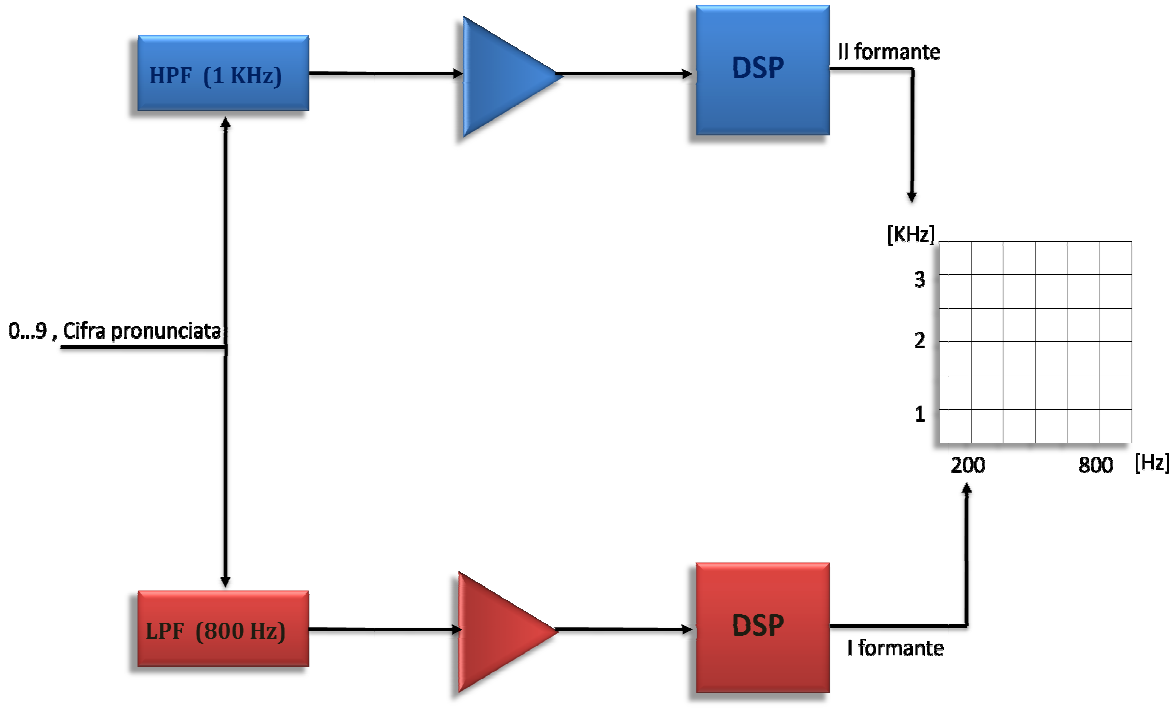


Figura 1.4 Schema a blocchi del funzionamento del sistema di Davis, Biddulph e Balashek

Con questa nuova applicazione si riuscì ad ottenere un'accuratezza del 93% nel riconoscimento e identificazione di un termine, arrivando a raggiungere anche una percentuale di errore inferiore al 2% nel caso in cui l'utente non muovesse la testa durante la pronuncia.
Questi risultati vennero pubblicati dagli stessi autori sulla rivista The Journal of the Acoustical Society of America, nell'articolo Automatic Recognition of Spoken Digits.
In altri sistemi di riconoscimento degli Anni Cinquanta, Olson e Belar degli RCA Laboratories realizzarono un'applicazione in grado di riconoscere 10 sillabe pronunciate da un singolo utente, mentre al MIT Lincon Lab, Forgie implementò il primo sistema cosiddetto speaker independent, cioè capace di riconoscere il suono indipendentemente dal parlatore, con un dizionario di 10 vocali: per la prima volta, si è in grado di estrarre dei parametri della voce che non dipendano da chi li ha pronunciati.
Una decina di anni più tardi, molti laboratori giapponesi realizzarono hardware per il riconoscimento vocale destinati all'impiego in applicazioni specifiche. In particolare, si ricorda il lavoro svolto da Suzuki e Nakata presso il Radio Research Lab di Tokyo, il riconoscitore di fonemi di Sakai e Doshita alla Kyoto University e un altro digit recognizer prodotto dai NEC Laboratories. Tra questi, la novità principale fu introdotta dai secondi, che furono i primi ad usare la segmentazione del parlato per l'analisi e l'identificazione delle parole pronunciate (questo concetto verrà chiarito nel paragrafo ad esso dedicato). Il lavoro della Kyoto University si può considerare il precursore del "continuos speech recognition".
Sempre in questi anni, Fry e Denes, alla University College in Inghilterra, costruirono un sistema capace di riconoscere 4 vocali e 9 consonanti. Incorporando informazioni statistiche sulle possibili sequenze di fenomeni nella lingua inglese (assegnando una probabilità praticamente nulla a quelle successioni che nel dizionario della lingua anglosassone non sono previste e attribuendo valori maggiori a quelle di più largo uso nel parlato comune), furono in grado di incrementarne l'accuratezza. Questo lavoro segnò il primo passo dell'impiego della sintassi statistica (statistical syntax) per l'ASR.
Verso la fine degli Anni Sessanta, Atal e Itakura formularono i primi concetti fondamentali del Linear Predictive Coding (LPC), che ha enormemente semplificato la stima del tratto vocale dalla forma d'onda del suono da esso generata.
A partire dalla metà degli Anni Settanta, iniziarono a diffondersi sempre più le idee di utilizzare il pattern recognition (basato sui metodi LPC) per il riconoscimento vocale. Sempre in questo periodo, Tom Martin fondò la *prima compagnia commerciale che producesse sistemi e applicazioni inerenti l'ASR*: la *Threshold Technology Inc*. e sviluppò il primo prodotto con finalità di mercato,

chiamato ***VIP-100 System***. Si trattava di un dispositivo usato in poche e semplici applicazioni, ma la sua importanza è dovuta all'influenza che ebbe sull'Advanced Research Projects Agency (ARPA) del Dipartimento di Difesa Americano nel fondare il programma Speech Understanding Research (SUR) .

Tra i partecipanti al progetto ARPA, si ricorda in particolare la Carnegie Mellon University, la quale realizzò "Harpy", un progetto di SR in grado di riconoscere un vocabolario di 1011 parole con una discreta accuratezza. Il contributo essenziale di questo sistema allo sviluppo delle tecniche di riconoscimento vocale è il concetto di ricerca a grafo, in cui il riconoscimento viene rappresentato come una rete di nodi connessi fra di loro in base alla connessione lessicale che lega le parole tra di loro, tenendo conto di precise regole sintattiche. La voce in ingresso, dopo un'analisi parametrica, viene segmentata e la sequenza così ottenuta viene sottoposta ad un confronto con tutte le osservazioni delle 1011 parole memorizzate. Usando come discriminante la distanza di Itakura, viene scelto il template che ha la minor distanza dal pattern di riferimento, ovvero quello che meglio rappresenta il segnale in ingresso.

Harpy fu molto probabilmente la prima macchina a trarre vantaggio dall'impiego di una rete a stati finiti nella modellizzazione del sistema, riducendo la complessità computazionale e, contemporaneamente, aumentando l'efficienza. Tuttavia, i metodi in grado di ottimizzare il risultato di un rete a stati finiti (Finit State Network, FSN) non furono realizzati prima degli Anni Novanta. Altre tecniche vennero sviluppate grazie al progetto dell'ARPA, tra cui si ricordano il CMU e il BBN: entrambe, nonostante le novità teoriche implementate, basate sul processare in modo parallelo il suono d'ingresso ed estrarne le caratteristiche da parametri di basso livello, tenendo presente vincoli e regole lessicali, non riuscirono a raggiungere lo scopo che l'associazione si era prefissata, tanto che nel 1976 essa cessò di esistere a causa della mancanza di fondi.

Contemporaneamente, oltre all'agenzia del Dipartimento di Difesa Statunitense, anche IBM e AT&T Bell Laboratories si dedicarono allo studio e allo sviluppo di applicazioni commerciali di speech recognition, anche se perseguendo obiettivi differenti e percorrendo strade diverse.

Scopo della IBM era creare una macchina da scrivere a comando vocale (VAP, Voice Activated Typewriter), in modo da poter convertire una frase pronunciata dall'utente in una sequenza di lettere e parole che potessero essere visualizzate su uno schermo o stampate su carta. In questo caso, il sistema di riconoscimento, chiamato Tangora, era di tipo speaker dependent, ovvero l'utilizzatore doveva preventivamente memorizzare il vocabolario con le parole da riconoscere. Il lavoro della IBM si focalizzò quindi sulla necessità di disporre di un vocabolario il più ampio possibile e sulla struttura del linguaggio, ovvero sulla sua rappresentazione statistica, in senso probabilistico, di una

sequenza di simboli (fonemi o parole). A questo genere di SR si fa spesso riferimento con il termine di "transcription". L'insieme delle regole sintattiche e delle probabilità statistiche della grammatica prende il nome di language model, la cui variante di uso più frequente è l'n-gram, che definisce la probabilità di occorrenza di una sequenza ordinata di n parole: si tratta di una semplificazione efficiente e molto potente della grammatica "classica", adattata per le macchine.

Ben differente ero l'obiettivo dei AT&T Bell Laboratories, che con il loro programma di ricerca intendevano fornire dei servizi di telecomunicazioni automatizzati per il pubblico, con comandi e controlli vocali per la telefonia. Era quindi necessario che questi sistemi potessero funzionare correttamente con un numero molto elevato di utenti (teoricamente decine di milioni, considerando la diffusione della telefonia fissa) senza la necessità che ognuno di essi dovesse preventivamente memorizzate i templates. I Bell Laboratories si focalizzarono quindi sulla tecnica opposta a quella sviluppata dalla IBM, ovvero sul modello speaker independent, che potesse lavorare anche in presenza di una elevata variabilità delle caratteristiche intrinseche della voce, come il timbro (uomo/donna), agli accenti regionali e a molti altri fattori. Questo portò alla creazione di un insieme di algoritmi per generare dei pattern sonori in modo che potessero essere usati per molti utenti e per differenti accenti.

Negli Anni Ottanta, le ricerche nel campo dello speech recognition si spostarono gradualmente verso metodologie differenti da quelle fino ad allora impiegate, basate quasi esclusivamente sul solo pattern recognition: iniziarono a prendere piede rigorosi modelli statistici. In particolare, quello che ebbe più successo furono le catene markoviane a stati finiti (*Hidden Markov Model*, HMM). Benché l'idea circa l'impiego delle tecniche appena nominate in applicazioni di riconoscimento risale già agli anni '60, la metodologia e la teoria non fu completa e pienamente compresa fino a metà degli Anni '80, a seguito di una larga serie di pubblicazioni sul tema. La popolarità e l'uso dell'HMM in applicazioni di automatic speech recognition è praticamente rimasta costante per due decenni, soprattutto grazie ai continui miglioramenti e perfezionamenti della tecnologia.

Un'altra tecnologia introdotta (o, meglio, reintrodotta) nei tardi Anni Ottanta, furono le *Reti Neurali Artificiali* (ANN, Artificial Neural Network). In realtà, questa tecnica era già stata impiegata a metà del XX secolo, ma i risultati furono molto deludenti e con probabilità di errore molto elevate. L'avvento di sistemi di processing paralleli e distribuiti (PDP, Parallel Distributed Processing), costituiti da un'elevata interconnessione di semplici elementi computazionali (perceptron), e conseguenti metodi di apprendimento (training), ridiedero interesse alle reti neurali.

Una particolare forma di PDP, il Multilayer Perceptron, fu il centro dell'attenzione degli studi e delle nuove implementazioni, grazie alla sua capacità di approssimare praticamente ogni funzione

con una precisione arbitraria. La grossa limitazione delle reti neurali dell'epoca, nonostante garantisca una ottima probabilità di corretto riconoscimento di fonemi o parole, era la scarsa capacità a trattare input variabili nel tempo, come la voce; ai giorni nostri, questo ostacolo è stato largamente superato e le ANN presentano una buona dinamicità di impiego anche con segnali non costanti.

Nel corso degli Anni Novanta si è avuto uno sviluppo e una diffusione esponenziale dei metodi di riconoscimento vocale, soprattutto quelli basati sulle catene di Markov, grazie al sempre maggiore interesse verso gli algoritmi di ottimizzazione e alla possibilità di realizzare queste applicazioni con costi di produzione minori, utilizzandole anche per dispositivi di uso comune, e quindi con un vastissimo bacino di potenziali client (come sempre, il ritorno economico non è un fattore trascurabile). Di seguito vengono illustrate in maniera schematica le tappe fondamentali dell'evoluzioni dei sistemi di riconoscimento vocale:

- ***Anni Venti***
  - *1920's* : Radio Rex
- ***Anni Cinquanta***
  - *1952* : Digit Recognizer
  - *1955* : primo sistema di riconoscimento indipendente dal parlatore
- ***Anni Sessanta***
  - *1960's* : Fast Fourier Transform (FFT)
  - *1966* : introduzione delle Hidden Markov Model (HMM)
- ***Anni Settanta***
  - *1971* : inizio studi da parte dell'ARPA
  - *1972* : Dynamic Time Warping
  - *1973* : Algoritmo di Viterbi
  - *1974* : modelli markoviani impiegati appositamente per il riconoscimento
  - *1975* : Linear Predictive Coding (LPC)
- ***Anni Ottanta***
  - *1980's* : Cepstrum e Mel-cepstrum, PLP
  - *1988* : second progetto ARPA
- ***Anni Novanta***
  - *1990's* : diffusione e miglioramento degli algoritmi basati su modelli markoviani

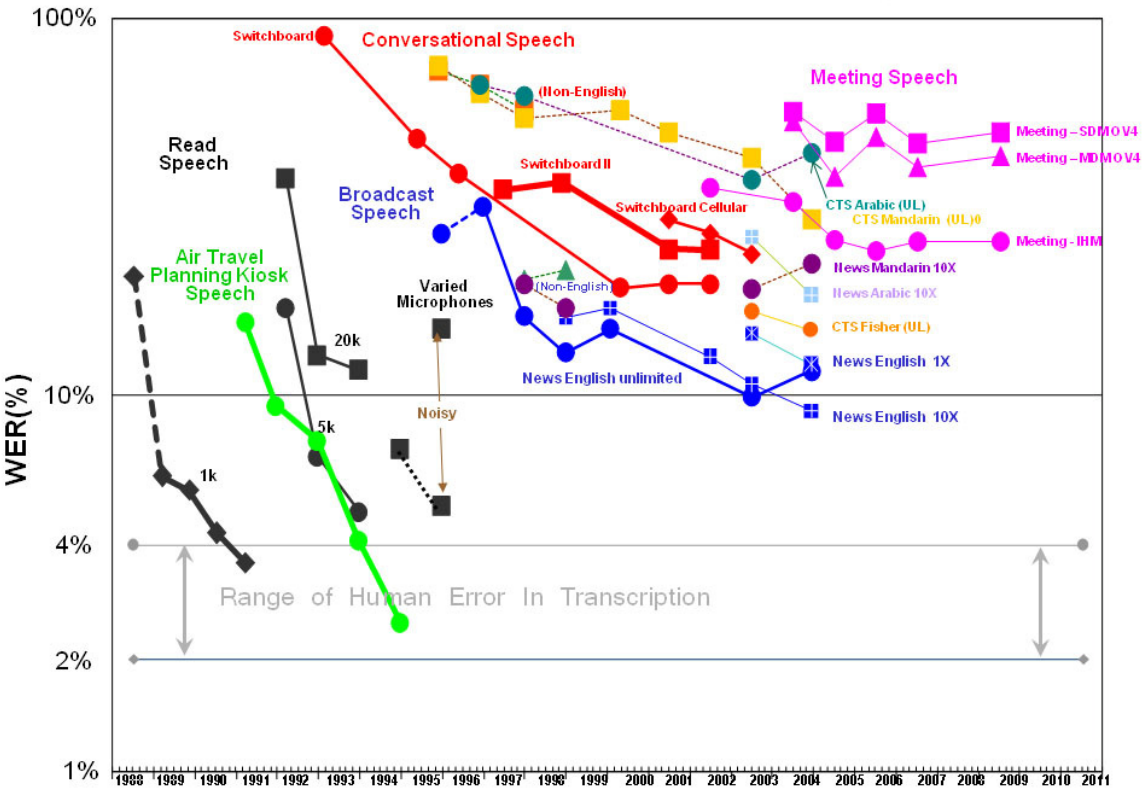


Figura 1.5 Evoluzione dei sistemi ASR negli ultimi due decenni (aggiornata Maggio 2009); si noti che in ordinata è rappresentata la probabilità d'errore

2. CENNI TEORICI

Sino ad ora si è solamente fatto un breve (e generale) accenno alla nascita e all'evoluzione dei sistemi di riconoscimento vocale nel corso dell'ultimo secolo, periodo durante il quale l'interesse verso queste possibili applicazioni è cresciuto sempre più. Per comprendere veramente i metodi, statistici e non che permettono di identificare una parola (o una sequenza di parole) è necessario analizzare anche da un punto di vista fisico il *fenomeno che permette all'uomo di comunicare e parlare*: la **voce**.

2.1 IL SUONO E L'ONDA ACUSTICA

Contrariamente a quanto spesso si è solito pensare, *il suono non è altro che una sensazione percepita dall'orecchio umano, generata dalla vibrazione di un corpo in oscillazione*. L'orecchio, grazie alla sua particolare conformazione, dà luogo ad una sensazione uditiva correlata alla vibrazione percepita, che può essere interpretata dal cervello, il quale le attribuisce un particolare significato, a seconda del tipo di linguaggio che il parlante e l'ascoltatore stanno adottando (generalmente si tratterà della lingua che accomuna i due soggetti). Si nota come l'elemento centrale perchè si possa produrre un suono è la vibrazione di un corpo: questo fenomeno è l'origine delle onde acustiche e, più in generale, del campo acustico, il quale è descritto dalla variazione della pressione e della velocità delle particelle del mezzo in cui avviene la propagazione.

*Con il termine **onda** si fa riferimento ad una perturbazione che si propaga nello spazio e che può trasportare energia* (e quantità di moto) *da un punto ad un altro*. Questa perturbazione è costituita dalla variazione di qualunque grandezza fisica (ad esempio pressione, temperatura, posizione, intensità del campo elettrico,…) attorno alla sua posizione di equilibrio, ovvero quella corrispondente all'assenza di oscillazione. È opportuno mettere in evidenza come ad essere trasportate non siano le particelle del mezzo, ma la variazione della particolare grandezza fisica che esse subiscono.

L'***onda acustica*** è un particolare tipo di onda meccanica longitudinale: essa *si propaga in tutte le direzioni dello spazio, generando, lungo il raggio di propagazione* (direzione ortogonale al fronte d'onda) *una vibrazione delle particelle investite dalla perturbazione (ossia la variazione di pressione dell'onda stessa)*. Questo fenomeno dà luogo a fasi alternate di compressione (alta densità e pressione) e di rarefazione (bassa densità e pressione) nel mezzo.

Le onde meccaniche, dette anche elastiche in quanto traggono origine dalle proprietà elastiche del corpo che le ha generate, possono propagarsi in tutti i tipi di mezzi, siano essi solidi che fluidi (gas e liquidi).

Possiamo chiamare ***sorgente sonora*** un *qualsiasi corpo che emette un suono*; in particolare, è utile ricordare che l'origine di un qualsiasi evento acustico è la vibrazione della sorgente stessa. Inoltre, dato che la propagazione di tale onda può avvenire solo grazie all'azione esercitata sulle molecole costituenti il mezzo in cui essa viaggia, è chiaro che un'onda acustica non può essere trasmessa nel vuoto (si dice che si propaga soltanto nella materia). Il massimo spostamento di una particella dalla sua posizione di equilibrio prende il nome di ampiezza, la quale tende a diminuire man mano che ci si allontana dalla sorgente sonora.

Tra le onde elastiche, spicca una particolare classe: le ***onde sonore***, le quali hanno *frequenze comprese tra 20 Hz e 20 KHz*, cioè nell'intervallo delle *frequenze udibili dall'orecchio umano*; infatti, l'uomo è in grado di percepire una sensazione sonora solamente per onde elastiche comprese tra 16 e 12000 Hz. Le onde al di fuori di questa banda giungono al nostro orecchio, ma non vengono percepite. La pulsazione determina l'altezza del suono: onde con frequenze inferiori ai 20 Hz si dicono infrasuoni, mentre quelle superiori a 20 KHz sono dette ultrasuoni.

Ricordando che la lunghezza d'onda è la distanza tra due compressioni (rarefazioni) successive, ovvero la distanza percorsa dalla perturbazione in un periodo, si ottiene il primo parametro fisico della stessa con la seguente equazione

$$\lambda = \frac{v}{f} \tag{2.1}$$

in cui, come è noto, con la lettera v si indica la velocità dell'onda nel mezzo, mentre f ne rappresenta la frequenza.

Considerando che la velocità dell'onda sonora, nell'aria, è di circa 331.4 m/s, assumendo i valori limiti di udibilità, si ottengono le lunghezze d'onda udite dall'orecchio umano:

$$\lambda_{max} = \frac{331.4}{16} = 20m \qquad \lambda_{min} = \frac{331.4}{12000} = 0.03m \tag{2.2}$$

Gli altri parametri caratteristici, sempre a livello fisico, sono comuni a tutti gli altri tipi di onde: frequenza, velocità di propagazione, costante di propagazione, polarizzazione. Accanto a questi, però, nei problemi di riconoscimento è necessario definire altri attributi che indicano la tipologia e

la qualità del suono. Questi termini, tuttavia, anche quando hanno lo stesso nome di una grandezza fisica (come accade ad esempio per l'intensità), non si riferiscono a caratteristiche oggettive del suono, ma a diversi modi di percepire il fenomeno stesso.

Il primo, e forse più conosciuto, di questi parametri è quello che comunemente (e volgarmente) chiamiamo *volume*, a cui si dovrebbe fare riferimento con il termine ***intensità***, grazie al quale è possibile distinguere suoni intensi da suoni più deboli. Vi è poi l'***altezza*** del suono, che consente di stabilire se un suono è acuto o grave (molto usata dai musicisti, perché essa permette di distinguere le diverse note), più precisamente quelli molto profondi hanno frequenze di poche decine di Hz, mentre quelli più acuti hanno pulsazioni di diverse migliaia di Hz. Poiché la lunghezza d'onda è inversamente proporzionale alla frequenza (a meno di un fattore costante pari alla velocità di propagazione nel mezzo), si può anche affermare che l'orecchio umano percepisce un suono tanto più alto quanto minore è la sua lunghezza d'onda. Ultimo, ma non meno importante, è il ***timbro***, forse la caratteristica più difficile da comprendere pienamente: *esso*, infatti, *indica la qualità percepita del suono e ci permette di distinguere due suoni, identici per altezza e intensità, che provengono da sorgenti differenti*. Si possono distinguere, ad esempio, la voce maschile da quella femminile o da quella di un bambino, o la melodia di un pianoforte da quella di una chitarra.

È importante dedicare un po' di tempo alla discussione di questo attributo del suono. Il timbro, infatti, è un *fenomeno multidimensionale*: non può essere espresso con un singolo numero in una qualche unità di misura, dato che prende in considerazione più parametri.

A ciascuna di queste peculiarità del suono corrisponde una ben precisa grandezza fisica relativa all'onda acustica che la definisce e la caratterizza. In particolare, dalla tabella si osserva come intensità ed altezza dipendano dalla forma d'onda temporale del segnale sonoro, mentre il timbro sia indipendente (quantomeno direttamente) dalla stessa

Parametro del suono	Grandezza fisica relativa all'onda
Altezza	Frequenza
Intensità (o volume)	Ampiezza al quadrato
Timbro	Composizione spettrale

Tabella 1.1: Parametri del suono e grandezze fisiche

Identificate e descritte le onde acustiche, e sonore in particolare, risulta opportuno e necessario mettere in evidenza la differenza fra ciò che comunemente indichiamo come rumore e ciò che è suono: in generale, *si definiscono **suoni** le onde acustiche generate da una qualsiasi vibrazione*

meccanica periodica, mentre i ***rumori*** sono *perturbazioni acustiche alle quali manca un preciso carattere di periodicità*.

Definite le caratteristiche principali di un'onda sonora, di seguito viene riportato un esempio di quello che è il diagramma temporale di una parola: la cosiddetta waveform (in italiano, forma d'onda):

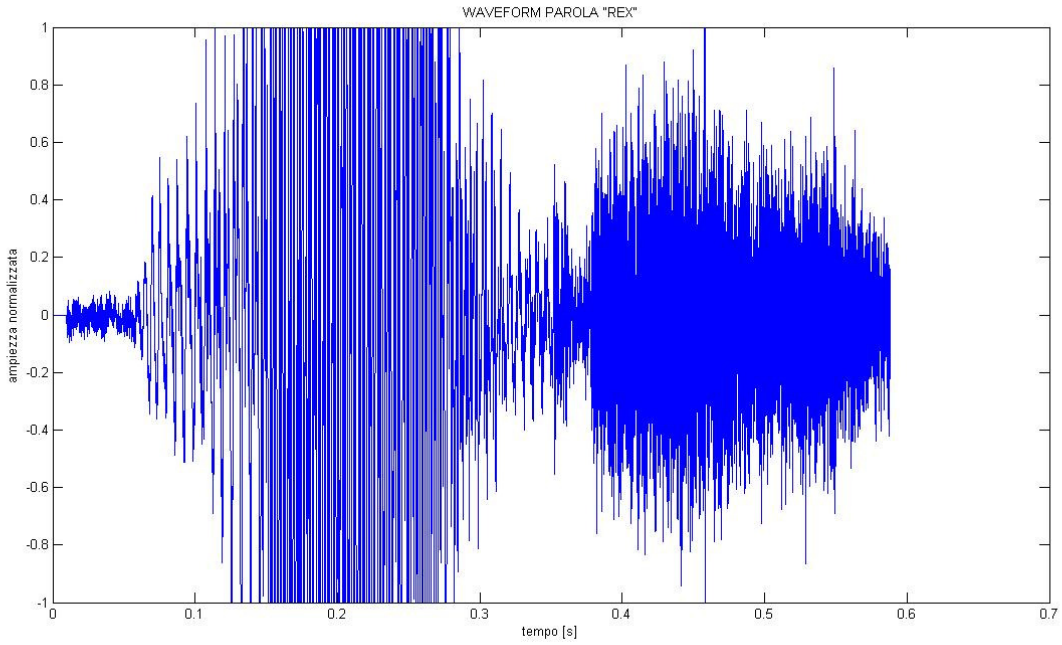


Figura 2.1: Forma d'onda della parola "Rex" (normalizzata)

2.2 LA VOCE E IL PARLATO

Il segnale vocale viene formato nel ***tratto vocale***, che va dalle corde vocali fino alle labbra, strutturato in modo tale da formare un sistema riverberante (ricordando che il riverbero non è altro che il suono che permane in un determinato ambiente quando il segnale diretto che lo ha generato si è esaurito, ed è quindi sostenuto unicamente dalle riflessioni dello stesso), la cui conformazione dipende dalla posizione della bocca e della lingua. Il segnale d'ingresso è generato dalle corde vocali, le quali sono due lamelle messe in vibrazione dal passaggio dell'aria. Non essendo particolarmente morbide, la loro oscillazione è di tipo impulsivo: producono una serie di "picchi" seguiti da spazi vuoti; questi ultimi vengono "riempiti" dalla risonanza, ossia una piccola serie di echi generati dal tratto vocale che agisce come una cassa armonica. Infatti, *se non vi fosse la risonanza, lo spettro delle corde vocali sarebbe semplicemente una lunga serie di armonici*.

Il suono prodotto dalle corde vocali può essere di due tipi: ***vocalizzato*** (quando viene generata una serie periodica di impulsi acustici) o ***non vocalizzato*** (quando le corde producono un fruscio con una distribuzione praticamente gaussiana e spettro abbastanza uniforme, assimilabile ad un rumore). La sequenza di impulsi che rappresenta l'origine del segnale vocalizzato ha una frequenza di ripetizione che va da circa 100 Hz (voce roca, maschi) ai 250/300 Hz (voce melodiosa, tipica dei bambini). Ciascuno di questi impulsi ha una durata di qualche millisecondo, quindi lo spettro che si ricava ha un'estensione che raggiunge gli 8/10 KHz.

Il segnale appena descritto attraversa il tratto vocale e viene sagomato spettralmente dalla funzione di trasferimento (tutti poli) corrispondente alla particolare conformazione assunta dalla cassa armonica.

Si osserva come si possano avere al massimo 5 frequenze di risonanza, anche se, nella maggior parte dei casi, per ogni vocale, se ne contano 3 o, in rari casi, 4. Queste frequenze di risonanza sono chiamate formanti e la loro posizione caratterizza le vocali.

Si presti particolare attenzione al fatto che quanto enunciato fino ad ora non garantisce la comprensione del parlato, anzi, l'intelligibilità dello stesso è soprattutto legata alle consonanti, ossia ai transitori tra una vocale ed un'altra.

L'emissione dei suoni vocalici può essere preceduta, interrotta o seguita da occlusioni o restringimenti del canale vocale, determinati dai movimenti articolatori. Questi suoni non vocalizzati, vengono detti ***consonanti***. Essi possono essere di vario tipo: le *consonanti sibilanti*, ad esempio, sono prodotte dal fruscio emesso dalle corde vocali (in modo analogo a quanto accade per i suoni non vocalizzati), mentre le consonanti di carattere *plosivo* (quali "b","p","t","g") sono caratterizzate da repentini e bruschi cambiamenti nella conformazione del tratto vocale.

Bisogna ricordare, poi, che non sempre le corde vocali entrano in azione: la maggior parte delle consonanti è prodotta mediante suoni generati dal passaggio dell'aria nel tratto vocale, opportunamente configurato dalla posizione della mandibola, della lingua e della labbra. Al contrario, esse risultano molto importanti per la generazione delle vocali. Ne deriva la distinzione tra suoni vocalizzati, che rappresentano le vocali, e non vocalizzati, che sono relativi alle consonanti.

Analizzando i suoni vocalizzati, per prima cosa si deve evidenziare come le vocali, a livello fonetico, siano ben più di 5: nelle varie forme, aperte, chiuse, mezze-aperte, mezze-chiuse, la lingua inglese ne conta da 10 a 14, mentre l'italiano di può arrivare a contarne fino a 10.

Le ***vocali*** sono *generate grazie all'effetto di cassa armonica del tratto vocale sulla vibrazione generata dalle corde, filtrando il suono e facendo in modo che esso sia caratterizzato da più*

formanti (fino ad un massimo di 5): infatti, ogni suono vocalizzato ha una propria configurazione di formanti, che è l'unico modo che consente di distinguere le varie vocali. Le prime tre formanti sono quelle che assicurano la comprensibilità della vocale, mentre il quarto e il quinto danno luogo al rinforzo formantico.

Vocale		Frequenza	Frequenza	Frequenza
Rappresentazione	**Fonema**	**I Formante**	**II Formante**	**III Formante**
[a]	Cava	520	1190	2380
[é]	Séta	530	1840	2480
[è]	Sèrpe	660	1720	2410
[i]	Lino	270	2290	3010
[o]	Sole	730	1090	2440
[ò]	Tòro	570	840	2410
[u]	Lupo	440	1020	2240

Tabella 1.2: Vocali italiane

Più in particolare, con il termine ***formante*** si fa riferimento alla *concentrazione di energia acustica in una certa banda frequenziale*: una formante può essere caratterizzata da una o più armoniche adiacenti. Il tratto vocale umano è l'unica forma di cassa di risonanza variabile conosciuta: esso può variare la sua configurazione, in modo da mettere certe frequenze della vibrazione prodotta dalle corde in fase e altre in controfase; in sostanza, il tratto vocale si comporta come un filtro che con la sua azione crea le formanti. *La posizione delle formanti gioca un ruolo importante nel riconoscimento del timbro*.

3. ANALISI DELL'ONDA SONORA

Sino a questo punto si è trattato il problema del riconoscimento vocale solamente da un punto di vista prettamente teorico, soprattutto approfondendo alcuni dettagli circa l'origine e la natura fisica del suono, accennando ad alcuni suoi parametri e definendo il loro legame con le grandezze fisiche dell'onda. Ora occorre entrare nel dettaglio delle operazioni di riconoscimento, cercando di capire quali siano le caratteristiche che meglio ci permettono di discriminare suoni diversi, e, da queste, costruire degli algoritmi che portino all'identificazione effettiva della parola pronunciata.

È evidente come *l'obiettivo primario in un problema di riconoscimento sia quello di estrarre informazioni dal suono, scartando ciò che non è rilevante in modo da rendere più agevole l'analisi e l'elaborazione*: si parla allora di ***estrazione delle caratteristiche*** (***features extraction***).

I passi principali per l'estrazione dell'informazione da un particolare segnale sono fondamentalmente tre:

- pre-elaborazione del suono;
- divisione del segnale in blocchi (anche parzialmente sovrapposti) e determinazione delle caratteristiche degli stessi;
- post-elaborazione;

La prima fase consiste nel modificare il segnale, in modo da facilitarne l'estrazione delle ***features*** con gli appositi algoritmi: con questo termine inglese si intendono *tutti gli aspetti distintivi, caratteristici, di un particolare oggetto, di una sequenza di dati in ingresso, e possono essere sia simboliche che numeriche*.

Alcune di queste operazioni sono, per esempio, la riduzione del rumore, l'equalizzazione, il filtraggio (generalmente di tipo passa basso, dato che questo è l'andamento spettrale tipico della voce umana, come si è visto nel capitolo precedente). Il secondo step comporta la suddivisione temporale del suono in segmenti (denominati ***frames***) di opportuna lunghezza, meglio se parzialmente sovrapposti per garantire continuità ed evitare che un evento di breve durata, ma di grande impatto sul segnale, possa concentrarsi in un unico frame, rischiando in questo modo di degradare la qualità dell'estrazione delle caratteristiche. Infine, con l'operazione di post-processing, si manipolano i risultati ottenuti per ottimizzarli e renderne più agevole l'interpretazione.

Nella figura è rappresentato lo schema a blocchi elementare per l'estrazione e caratterizzazione delle features:

Figura 3.1: Schema a blocchi del processo di estrazione delle caratteristiche

I metodi di analisi del suono e di estrazione dei suoi parametri derivano direttamente dal segnale rappresentato nel tempo oppure dallo studio dello stesso nel dominio coniugato, derivando lo spettro di potenza. Nel proseguio del capitolo verranno inizialmente illustrati i metodi base per la stima degli attributi nel dominio temporale, successivamente si discuterà l'importante problema della stima dell'inviluppo spettrale e dell'analisi dello stesso.

Verranno quindi introdotti e descritti vari parametri (appartenenti a differenti livelli di studio) ricavabili da queste rappresentazioni ed utilizzabili per la descrizione dei suoni e per un'ulteriore analisi volta a separare e comprendere gli stessi e la loro organizzazione.

3.1 PARAMETRI NEL DOMINIO DEL TEMPO

L'uso dei parametri che descrivono alcune delle caratteristiche fondamentali di un segnale è molto diffuso nell'ambito dell'elaborazione dei segnali, soprattutto in quei casi (come quello del riconoscimento vocale) in cui sia necessario memorizzare e/o trattare delle grandezze che, considerate così come le si osservano nella realtà, richiederebbero una mole enorme di dati e di memoria. Per rendere più agevole ed immediato lo studio di particolari segnali, si ricorre quindi a valutare dei precisi attributi che li descrivono pienamente.

Bisogna sin da subito dire che *questi metodi* (usati, per esempio, per discriminare il rumore di fondo dal parlato, oppure distinguere suoni vocalizzati e non, come vedremo in seguito) *non danno risultati assolutamente certi sull'informazione che il segnale porta con sé* e che *spesso vengono usati in combinazione*, in modo da incrementare l'accuratezza e l'affidabilità sull'identificazione. Tuttavia, il loro grande vantaggio risiede nella relativa ***semplicità d'implementazione*** e nelle ***modeste capacità di calcolo*** richieste (tant'è che uno degli obiettivi di questa tesi è valutare la possibilità, ed eventualmente le modalità, di poter realizzare dispositivi stand-alone a basso costo che siano in grado di implementare speech recognition con una discreta affidabilità).

Ritornando ai parametri che descrivono le principali caratteristiche di un suono, è importante far notare come spesso essi siano *varianti nel tempo*: ciò richiede di prestare particolare attenzione alla loro manipolazione. Generalmente, per meglio affrontare lo studio del segnale audio, si ipotizza che le sue proprietà cambino lentamente nel tempo (quantomeno rispetto al periodo di campionamento): questo ci permette di *elaborare i segmenti in cui viene diviso il segnale originario come se fossero suoni con caratteristiche costanti all'interno di un singolo frame*. Tale supposizione è valida per il segnale vocale, dato che la generazione delle parole, a cui contribuiscono sia le corde vocali che l'intero apparato fonatorio (laringe, lingua, bocca), avviene con una rapidità non molto elevata, tanto che le proprietà del suono posso essere considerate pressoché costanti entro i 100/200 ms (quindi con una frequenza di circa 10 Hz, nettamente inferiore a quella di campionamento).

La ***finestratura temporale*** viene impiegata per segmentare il segnale audio, determinando la durata del singolo frame in funzione della larghezza della finestra stessa. Questo valore deriva dal compromesso tra più fattori: deve essere sufficientemente breve, dimodochè le proprietà del suono non varino al suo interno, ma al tempo stesso deve essere abbastanza lungo da permettere il calcolo del parametro che si vuole studiare; infine, il susseguirsi delle finestre dovrebbe ricoprire interamente il segnale (in questo caso di dovrà verificare che il frame rate del parametro che si sta valutando, ovvero la frequenza con cui esso si manifesta, sia almeno pari all'inverso della durata della finestra: ciò significa che tutti i campioni della grandezza originaria vengono analizzati; qualora questa condizione non fosse verificata, si perderebbe alcuni valori, in quanto la finestratura non copre tutto il segnale).

La finestra per eccellenza è quella ***rettangolare***, la più semplice ed intuitiva:

$$w(n) = \begin{cases} 1 & per\ 0 \leq n \leq N-1 \\ 0 & altrimenti \end{cases} \tag{3.1}$$

Benché di facile implementazione, presenta però un grosso limite: traslando la finestra nel tempo (di un passo opportuno) per analizzare frames successivi di segnale, qualora vi siano grandi variazioni dei parametri in esame, con la finestra rettangolare non è possibile sopperire a questa discordanza di valori, che, anzi, viene fatta risaltare. Si pensi, ad esempio, alla misura dell'energia, ottenuta sommando il quadrato dell'ampiezza dei campioni contenuti nella finestra: qualora vi siano delle oscillazioni di notevole portata tra due campioni adiacenti, l'energia misurata sarà notevolmente maggiore o minore a seconda che il campione con ampiezza elevata venga incluso o meno.

Per questo motivo, si fa spesso ricorso a finestre "smussate", i cui fronti di transizione non siano immediati, repentini, ma più dolci, in modo da limitare il contributo dei campioni limite, ovviando alle problematiche appena descritte. Con queste tecniche di windowing si privilegiano dunque i valori centrali; la soluzione a cui immediatamente si pensa è una finestra triangolare, che presenta una "smussatura" lineare a partire dal punto di massima ampiezza. Molto più spesso, però, si impiega la ***finestra di Hamming***, definita nel seguente modo:

$$w(n) = \begin{cases} 0.54 - 0.46 \cdot \cos\left(\frac{2\pi n}{N-1}\right) & per\ \ 0 \leq n \leq N-1 \\ 0 & altrimenti \end{cases} \tag{3.2}$$

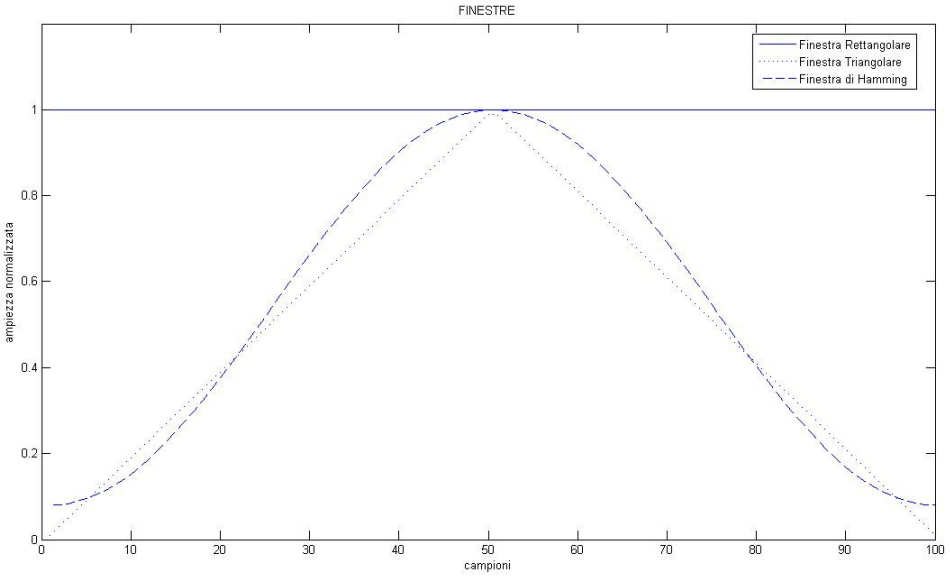


Figura 3.2: Finestra rettangolare, triangolare e di Hamming, aventi ampiezza pari a 100 campioni (il codice è in appendice)

Ora che sono state definite le modalità secondo cui avviene la segmentazione del parlato in frames dalle caratteristiche costanti, è possibile introdurre i veri e propri attributi del suono determinabili dalla conoscenza della forma d'onda temporale del segnale.

3.1.1 SHORT-TIME AVERAGE ENERGY e MAGNITUDE

Il primo parametro che viene spontaneo analizzare in un segnale, è rappresentato dalla sua ***energia***. Si definisce dunque l'energia media dei campioni contenuti all'interno di una generica finestra di lunghezza N, detta anche ***Short Time Average Energy***, che per un segnale discreto si calcola con la seguente formula:

$$E(n) = \frac{1}{N} \cdot \sum_{i=n-N+1}^{n} s(i)^2 \tag{3.3}$$

in cui si nota come l'energia media del campione n-esimo dipenda dagli N-1 campioni precedenti (ovvero il campione n è l'ultimo racchiuso all'interno della finestra). Questo parametro è un buon modo per determinare se il suono analizzato è vocalizzato, nel qual caso avrà un'energia molto elevata, dovuta alla periodicità del segnale, oppure non vocalizzato, in cui l'STE sarà basso.

Un *inconveniente della misura di questo parametro è la sua sensibilità a grandi ampiezze di segnale* (i campioni sono elevati al quadrato): per questo motivo spesso si tende ad ovviare a questo problema impiegando la ***Short Time Average Magnitude***, con la quale non si considera più l'energia, ma il modulo dell'ampiezza del campione, così da rendere il parametro meno suscettibile a grandi variazioni del segnale in ingresso:

$$M(n) = \frac{1}{N} \cdot \sum_{i=n-N+1}^{n} |s(i)| \tag{3.4}$$

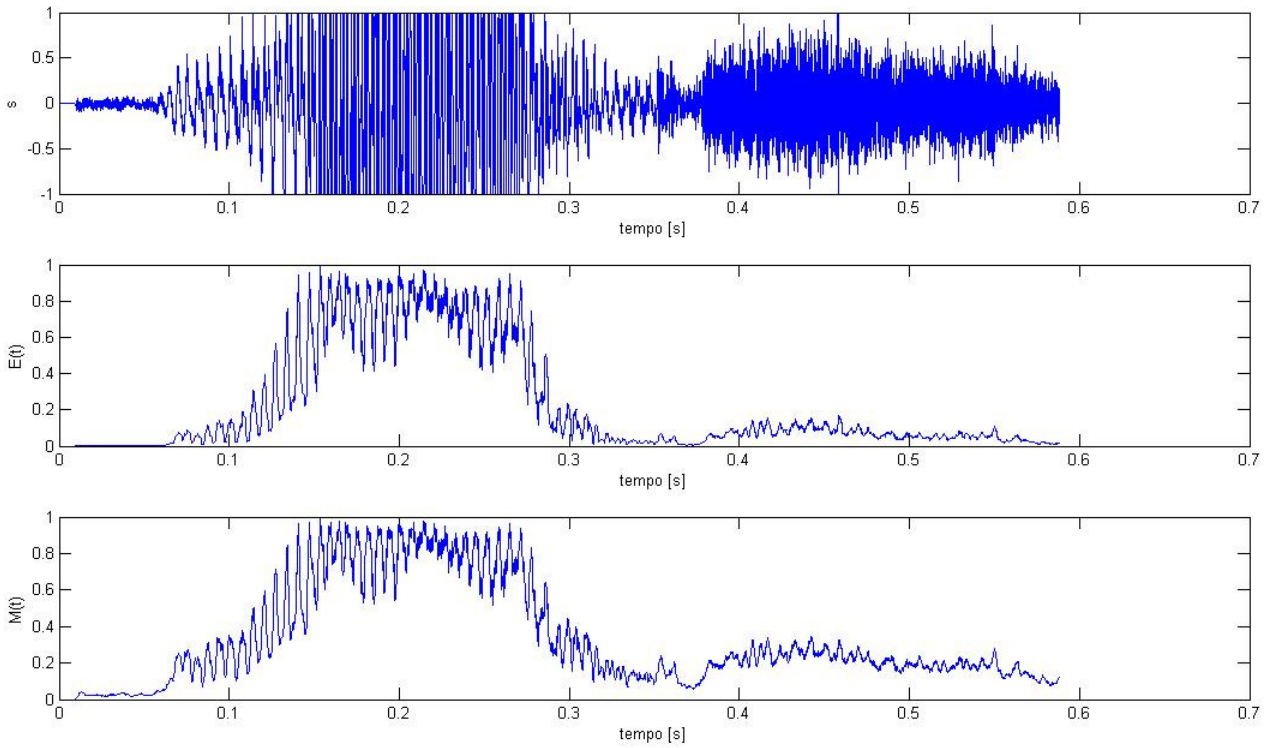


Figura 3.3: In alto l'espressione "Rex", mentre di seguito sono riportare le corrispondenti Short Time Average Energy e Short Time Average Magnitude normalizzate al valore massimo, calcolato usando finestre rettangolari da $N = 100$ campioni e con frame rate pari alla frequenza di campionamento ($44.1\ KHz$)

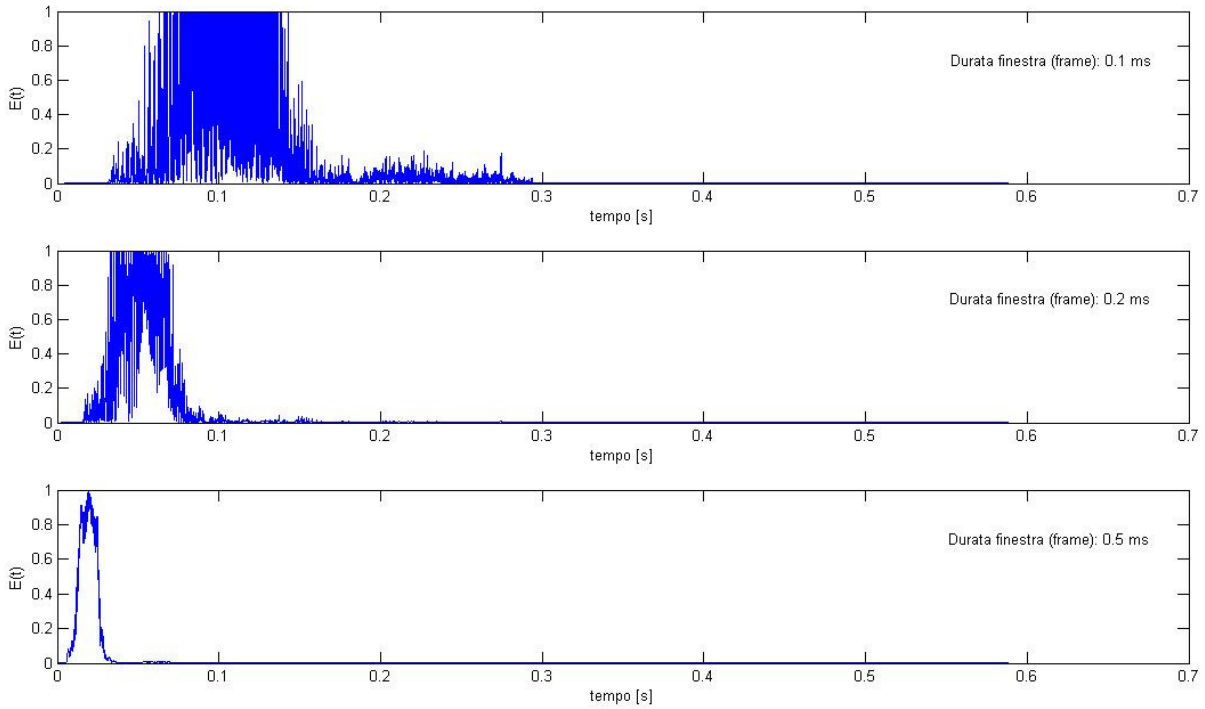


Figura 3.4: Short Time Average Energy nel caso in cui cambi la durata della finestra. Si noti come all'aumentare di questo valore, la STE risulti sempre più smussata

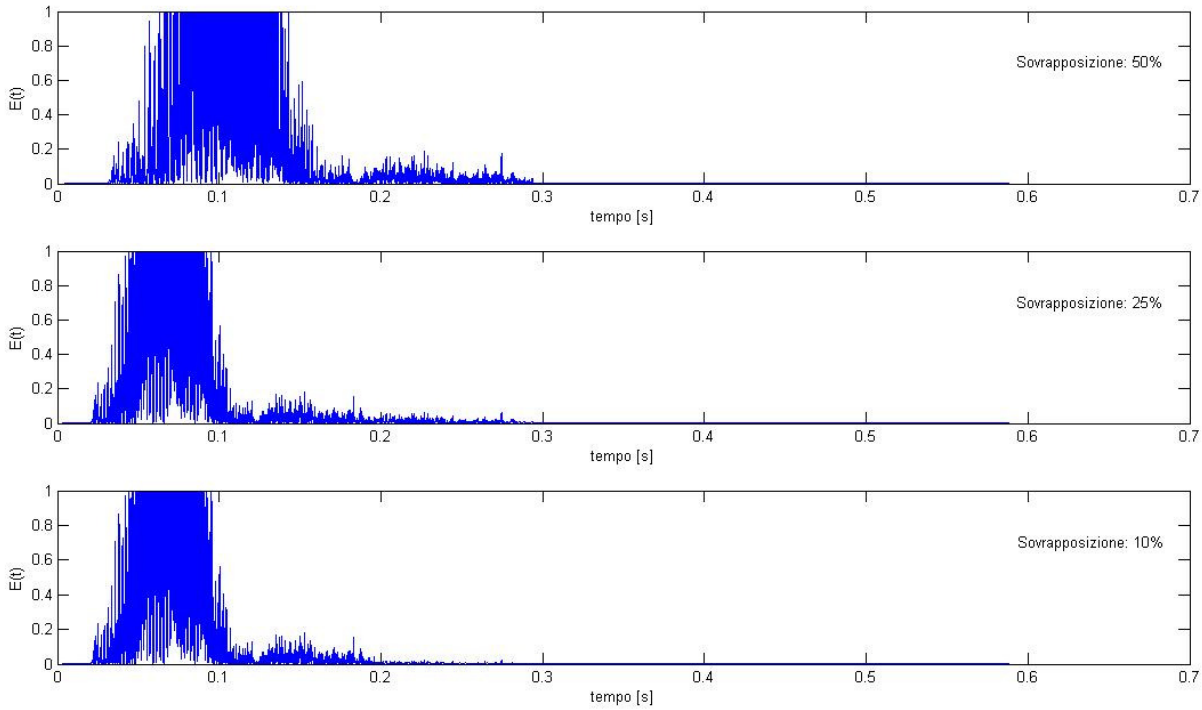


Figura 3.5: Short Time Average Energy qualora a variare sia la percentuale di sovrapposizione tra un frame ed il successivo. Si osserva che la differenza tra le varie casistiche non è così sostanziale come nel caso precedente

3.1.2 SHORT-TIME AVERAGE ZERO CROSSING RATE

Lo ***Zero Crossing Rate*** (ZCR) indica il numero di passaggi per lo zero del segnale che, da un punto di vista matematico, è rappresentato dal cambiamento di segno tra due campioni successivi. Si tratta di un parametro che *permette, senza l'impiego della trasformata di Fourier, di dare una discreta informazione spettrale* (ma la misura viene effettuata nel dominio del tempo) *ad un basso costo computazionale*: infatti, il suo valore non è altro che l'indicazione della frequenza attorno alla quale è maggiormente concentrata l'energia del segnale stesso. Nell'analisi vocale, lo Zero Crossing Rate *permette di determinare se il suono* (o parte di esso) che si sta analizzando *è vocalizzato oppure no*: infatti, il modello della generazione della voce indica che l'energia della componente vocalizzata è concentrata sotto i 3 KHz, mentre quella della componente non vocalizzata a frequenze maggiori. Quindi, *ad alte ZCR corrispondo suoni non vocalizzati* (unvoiced speech), *mentre a valori bassi di questo parametro suoni vocalizzati* (voiced speech).

Indicando con $sign(m)$la funzione segno, lo ZCR può essere calcolato come:

$$Z(n) = \frac{1}{N} \cdot \sum_{i=n-N+1}^{n} \frac{|sign[s(i)] - sign[s(i-1)]|}{2} \tag{3.5}$$

Affiancata alla Short-Time Average Energy consente di individuare, con buona precisione, l'inizio e la fine delle parole.

Per segnali a banda stretta, inoltre, conoscendo il rate con il quale avviene l'attraversamento dello zero è possibile calcolare la frequenza fondamentale degli stessi:

$$f_0 = \frac{ZCR \cdot f_s}{2} \tag{3.6}$$

avendo indicato con f_s la frequenza di campionamento del segnale e con ZCR lo Zero Crossing Rate per campione. È tuttavia molto sensibile al rumore (sia quello degli ADC, sia quello dei sistemi digitali, ma anche dei 50 Hz della rete di alimentazione).

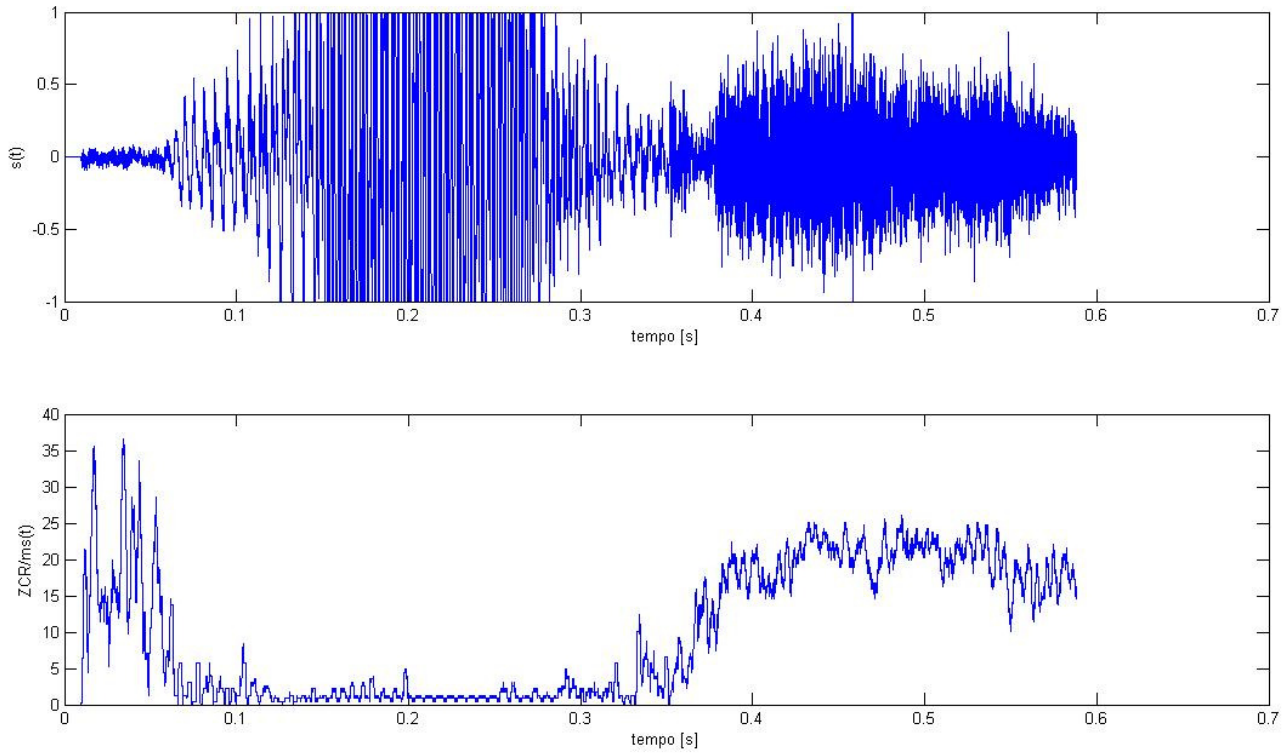


Figura 3.6: Zero Crossing Rate (al millisecondo) dell'espressione "Rex", calcolato con una finestra rettangolare di $N = 100$ campioni e frame rate parti a quello della frequenza di campionamento; si noti come sia possibile discriminare la /e/ (suono vocalizzato) dal resto della parola

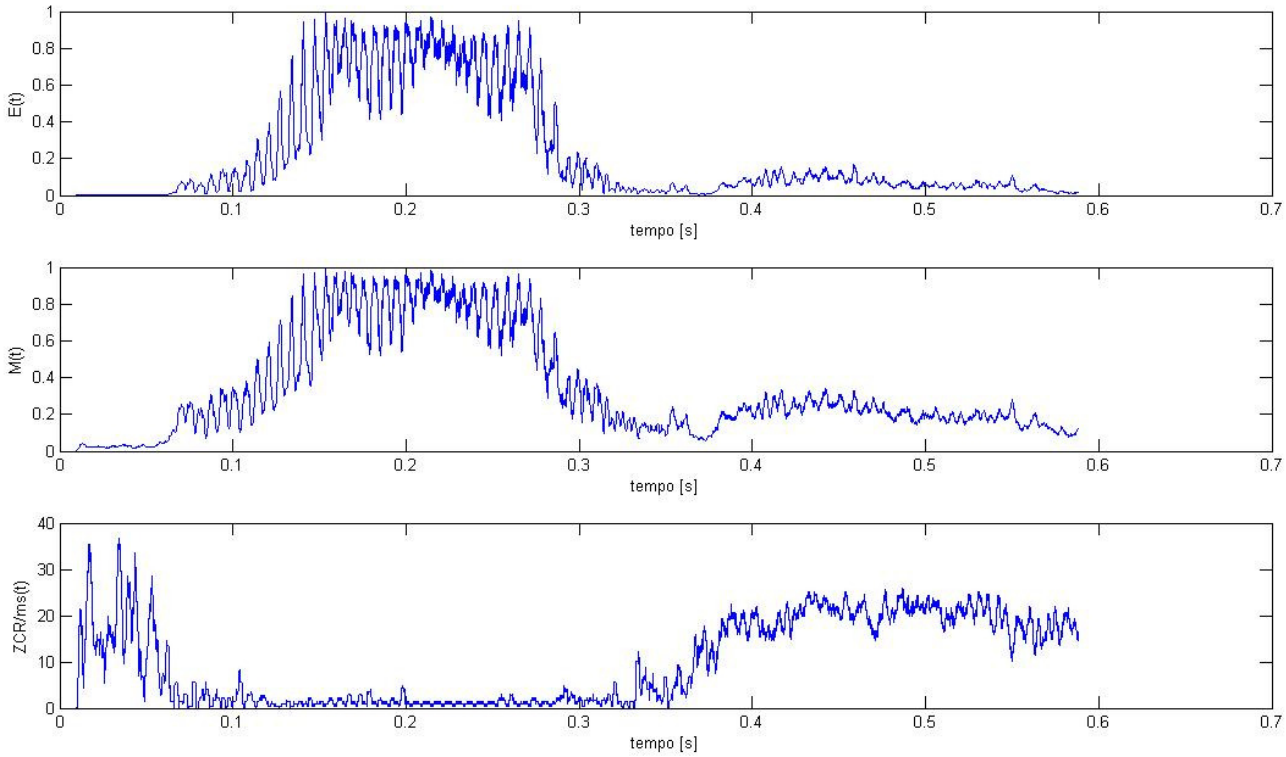


Figura 3.7: Confronto dei vari parametri analizzati a parità di condizioni di calcolo (stessa ampiezza della finestra e frame rate)

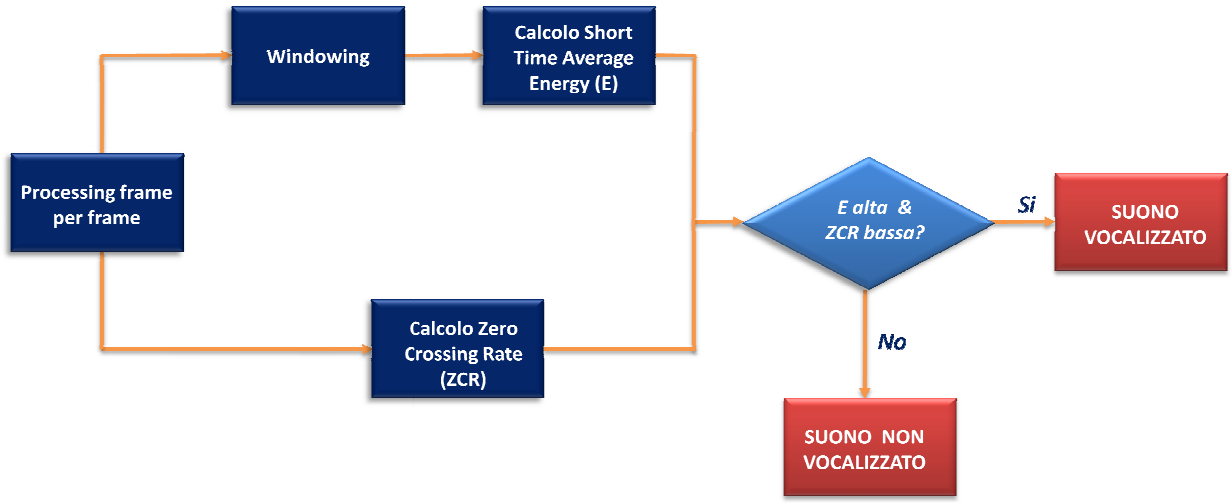


Figura 3.8: Schema per verificare se un suono è vocalizzato o meno

3.1.3 SHORT-TIME AUTOCORRELATION FUNCTION

Un altro parametro molto utilizzato nel campo del signal processing, che può essere ripreso nello studio e nell'analisi del parlato per ottenere delle informazioni importanti circa il tipo di suono, ed in particolare la sua periodicità, è l'***autocorrelazione***, ossia l'anti-trasformata di Fourier della densità spettrale di energia $C_s(f)$:

$$R(k) = F^{-1}[C_s(f)] = F^{-1}[|S(f)|^2] \tag{3.7}$$

Essa è anche definita come la correlazione del segnale con sé stesso o, meglio, con una sua versione traslata di un certo intervallo temporale, in modo che sia possibile individuare al suo interno eventuali pattern che si ripetono periodicamente. Si ricorda che, se è dimostrata l'autocorrelazione tra due valori (campioni) di un segnale (o, più in generale, di una variabile), al variare di uno cambierà anche l'altro. Rispettando questa definizione, per un segnale discreto la formula matematica per il calcolo dell'attributo in analisi è la seguente:

$$R(k) = \sum_{m=-\infty}^{+\infty} s(m) \cdot s(m+k) \tag{3.8}$$

Questo parametro conserva le informazioni relative alle armoniche del segnale, all'ampiezza delle formanti e alla loro frequenza. La *Short-Time Autocorrelation Function trova applicazione nell'estrazione del pitch (timbro), oltre che nella discriminazione tra voiced e unvoiced speech*. Inoltre, grazie alla proprietà di cui gode l'autocorrelazione, ovvero che, se il segnale è periodico di

periodo T, allora anche l'autocorrelazione ha lo stesso periodo, è possibile determinare la frequenza fondamentale f_0.

Tuttavia, se la prima formante è molto vicina, o addirittura più bassa, della frequenza fondamentale, questo può interferire con la stima, oppure, se l'energia della prima formante è particolarmente elevata, si crea una periodicità in competizione con quella di f_0. Inoltre, il segnale vocale nella realtà è quasi-periodico, per cui i picchi della funzione di autocorrelazione non sono elevatissimi, con possibili difficoltà nella loro individuazione.

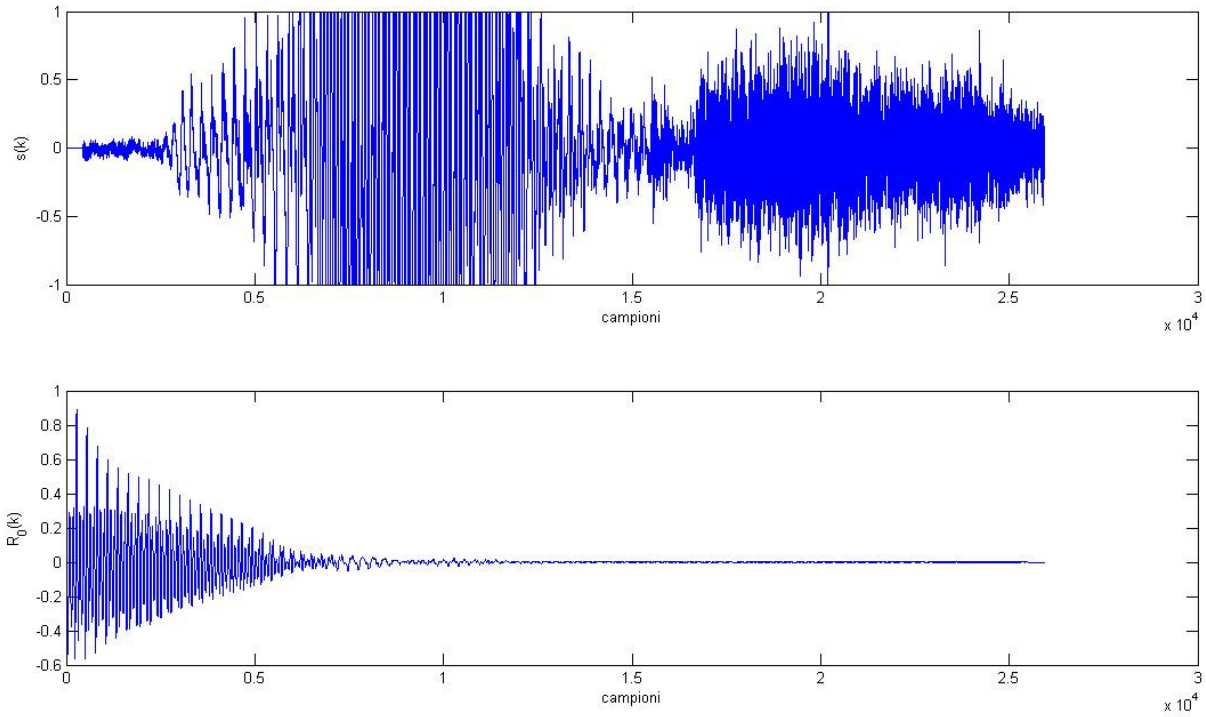


Figura 3.9: Funzione di autocorrelazione relativa alla solita waveform

3.1.4 STIMA DEL PITCH

In molte applicazioni è importante potere determinare la ***frequenza fondamentale*** (f_0), detta anche **pitch**, ovvero la *tonalità del suono*. Il suono vocalizzato viene generato dalla vibrazione delle corde vocali e il pitch si riferisce alla frequenza di questa vibrazione. Dalla Short Time Autocorrelation Function si ricava l'informazione sulla periodicità del segnale, determinando il campione in corrispondenza del quale si trova il primo massimo della funzione dopo quello all'istante iniziale[1]:

[1] una delle proprietà fondamentali della funzione di autocorrelazione afferma che il massimo della stessa si ha in corrispondenza dell'origine; di conseguenza, se la funzione è periodica, il massimo si ripresenterà ad ogni istante multiplo del periodo

$$f_0 = \frac{f_s}{k_m} \tag{3.9}$$

avendo indicato con f_s la frequenza di campionamento, mentre k_m è l'indice temporale in cui si è registrato il primo massimo dopo quello per $k = 0$.

Va tenuto in considerazione che, talvolta, il terzo massimo ha ampiezza maggiore del secondo: si rischia un errore nella stima (il periodo calcolato sarebbe quello di un'armonica della frequenza fondamentale); per questo motivo, spesso si affiancano altri metodi che permettano di evitare questi errori.

3.2 STIMA DELL'INVILUPPO SPETTRALE

Oltre all'analisi temporale dei parametri e delle caratteristiche del segnale sonoro, molto importante nella caratterizzazione dello stesso è lo studio dello spettro armonico, in particolar modo dell'inviluppo spettrale, soprattutto nel caso in cui si abbia a che fare con la voce umana. *Le bande di frequenza in cui è maggiormente concentrata l'energia rappresentano le principali risonanze del tratto vocale umano e sono uno degli strumenti principali per determinare dei ben specifici attributi* (come, ad esempio, l'individuazione delle vocali).

Il primo semplice metodo per l'analisi nel dominio coniugato è l'impiego di un ***banco di filtri***: lo spettro in ampiezza del segnale viene approssimato mediante segmenti; in particolare, vengono selezionati i massimi, oppure si considerano punti equispaziati nelle frequenze, e si congiungono con delle linee rette: una sorta di discretizzazione nelle pulsazioni dello spettro ed interpolazione delle rispettive ampiezze. Si tratta di un metodo flessibile, ma poco preciso. Dal punto di vista computazionale, i filtri possono essere realizzati mediante la Fast Fourier Transform (FFT, cioè la trasformata di Fourier implementata dai processori digitali), calcolando prima lo spettro del segmento di segnale in ingresso, e poi sommando i contributi frequenziali pesandoli per la risposta in frequenza del filtro stesso.

Ad esempio, nel caso in cui i filtri adottati siano passabanda rettangolari, è sufficiente sommare i contributi dei bin appartenenti alla banda di tale filtro (infatti, per definizione, il filtro rettangolare ha una risposta in frequenza unitaria per tutti i campioni compresi nella banda, nulla al suo esterno): si ricava che l'energia $E(j)$ del j-esimo frame vale

$$E(j) = \frac{1}{N} \cdot \sum_{k \in B} \left| s_j(k) \right|^2 \tag{3.10}$$

Dove B rappresenta la banda del filtro, N la dimensione della FFT, s_j i campioni di segnale appartenenti al j-esimo frame.

3.2.1 STIMA DELL'INVILUPPO SPETTRALE MEDIANTE PREDIZIONE LINEARE (LPC)

La ***codificazione per predizione lineare*** (*Linear Predictive Coding*) fa parte di una famiglia di tecniche per la sintesi/risintesi del linguaggio umano. È tuttora una delle tecniche più utilizzate per questo scopo. *La LPC si basa sulla stima dello spettro di un suono, grazie alla definizione dei coefficienti di filtri che sarebbero necessari per sintetizzarlo se applicati ad una fonte di eccitazione.*

Il Linear Predictive Coding è quindi un metodo che consente di predire il campione di un segnale qualora siano noti tutti (teoricamente), o parte (nella pratica), i campioni precedenti, appartenente al ramo della cosiddetta analisi spettrale parametrica, la quale consente di determinare i parametri di un generico filtro, in modo che lo spettro in uscita dallo stesso assomigli quanto più possibile a quello del segnale da analizzare (nel caso oggetto di questo elaborato, si tratterà della voce).

La LPC si fonda sulla visione del segnale vocale come costituito da due componenti che possono essere separate, ossia ciò che comunemente viene definito come ***modello sorgente-filtro*** del parlato: esso consiste nel "dividere" la voce in una *funzione di trasferimento*, la quale porta con sé l'*informazione sulla qualità del parlato*, e nell'*eccitazione*, caratterizzata dal *pitch* e dal suono.

Si suppone quindi che il generico campione $i-esimo$ sia dato dalla combinazione lineare degli N precedenti (N rappresenta l'ordine del predittore); quanto detto si esprime matematicamente con la seguente formula:

$$\hat{s}_i = \sum_{k=1}^{N} a_k \cdot s_{i-k} \tag{3.11}$$

dove $\hat{s}$ indica il campione stimato, mentre a_k i coefficienti con cui pesare ciascun campione già noto (s_{i-k}). Ovviamente, all'aumentare dell'ordine, l'accuratezza del filtro aumenta sempre più, fino ad essere in grado di determinare esattamente il valore del campione qualora N tenda ad infinito. Nella realtà ciò è irrealizzabile: *la pratica ha dimostrato che ordini compresi tra 10 e 20 garantiscono una buona precisione ad un costo computazionale limitato*. Trattandosi di un predittore, e quindi di un sistema che implementa una stima, la scelta dei coefficienti viene fatta in modo da minimizzare l'errore quadratico medio tra il valore effettivo del campione e quello stimato; si definisce dunque un errore di predizione, dato da

$$e_i = s_i - \hat{s}_i = s_i - \sum_{k=1}^{N} a_k \cdot s_{i-k} \tag{3.12}$$

$$E[|e_i|^2] = E[(s_i - \sum_{k=1}^{N} a_k \cdot s_{i-k}) \cdot (s_i - \sum_{k=1}^{N} a_k \cdot s_{i-k})^*] \tag{3.13}$$

L'errore della stima è minimo quando esso risulta incorrelato con i dati, ovvero quando non vi è più alcuna possibilità di ridurlo ulteriormente; quest'ultimo si ricava dalla *condizione di ortogonalità* (in realtà si tratta di un insieme di N equazioni, ossia tante quante l'ordine del predittore):

$$\frac{\partial E[|e_i|^2]}{\partial a_k} = 0 \qquad k = 1, \dots, N \tag{3.14}$$

Le equazioni 3.14 vengono dette di Yule-Walker, dagli studiosi che per primi le utilizzarono; esse possono essere riscritte nel seguente modo:

$$E\left[\varepsilon_p(n) s^*(n-i)\right] = r_i + \sum_{k=1}^{p} a_k \cdot r_{i-k} = 0 \qquad i = 1, \dots, p \tag{3.15}$$

in cui r_i è la funzione di autocorrelazione del segnale originario.
È possibile anche calcolare la trasformata z dell'errore di predizione, risultando:

$$E(z) = S(z) - \sum_{k=1}^{N} a_k \cdot S(z) \cdot z^{-k} = S(z)[1 - \sum_{k=1}^{N} a_k \cdot z^{-k}] = S(z)A(z) \tag{3.16}$$

In cui si nota come l'errore commesso dal sistema sia rappresentabile come prodotto della trasformata z del segnale originale $S(z)$ e la funzione di trasferimento $A(z)$, la quale non è altro che un filtro digitale a tutti zeri (FIR), i cui coefficienti corrispondo agli zeri del filtro stesso nel piano z. In modo analogo, è possibile esprimere il segnale noto, s, in funzione dell'errore e della funzione di trasferimento, ottenendo

$$S(z) = \frac{E(z)}{A(z)} \tag{3.17}$$

Nella quale la componente $1/A(z)$ è un filtro a tutti poli (IIR), dove ciascuno dei coefficienti a_k rappresenta un polo nel piano z; ovviamente, affinchè il sistema sia in equilibrio, questi devono giacere sulla circonferenza a raggio unitario.

In base a quanto detto in precedenza, circa le modalità con cui viene generato il suono, si ha che lo ***spettro dell'errore*** nella stima può avere una struttura differente, a seconda che il suono che si sta analizzando sia vocalizzato o meno. Nel primo caso, lo spettro risulterà periodico e sarà evidente il pitch oppure una struttura armonica, mentre nel secondo sarà assimilabile ad un rumore bianco, in cui non è possibile individuare alcuna periodicità.

Nel caso particolare del riconoscimento vocale, questo algoritmo viene impiegato per calcolare i coefficienti del filtro a partire da quelli del segnale (che, ovviamente, è noto, trattandosi della parola pronunciata dall'utente e che deve essere riconosciuta), *avendo come obiettivo quello di ricostruire A(z)* che, in base a quanto detto all'inizio del paragrafo, è la funzione di trasferimento tra il segnale d'ingresso $s(n)$ e l'eccitazione $e(n)$. Essa descrive la qualità del parlato, permettendo di capire la voce dell'utente, evitando equivocazioni tra più persone; al contrario, l'eccitazione identifica il particolare suono e la precisa parola pronunciata.

Come si può vedere dallo schema a blocchi illustrato di seguito, *ponendo in ingresso al filtro LPC il segnale originario, in uscita si ottiene l'eccitazione, mentre se si considera il filtro inverso, esso restituirà il segnale originale se come input ha la componente di eccitazione*:

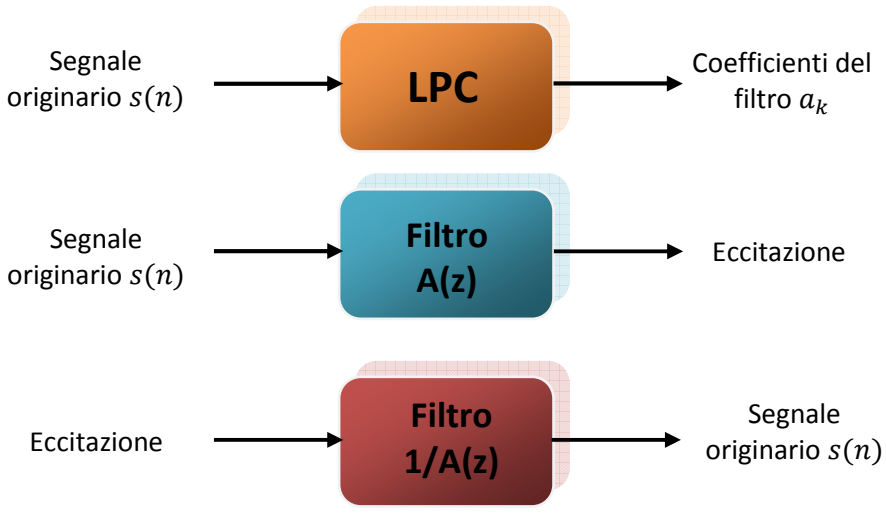


Figura 3.10: Schema a blocchi della logica che sta alla base dell'algoritmo LPC

Spesso, nei sistemi di speech processing, si rivela molto utile effettuare una pre-enfasi sul segnale in ingresso prima che esso venga sottoposto a Predizione Lineare. Questa manipolazione si rende necessaria a causa del fatto che l'energia dello spettro della voce tende a diminuire con l'aumentare

della frequenza: la pre-enfasi non fa altro che aumentare l'energia dello spettro di un valore proporzionale alla frequenza associata a quel particolare valore. Il risultato di questa operazione è appiattire lo spettro, in modo che esso risulti composto da un elevato numero di formanti aventi praticamente la stessa ampiezza. In questo modo è possibile effettuare un'analisi LPC del segmento del parlato molto più accurata. Senza questa operazione, la predizione lineare si concentrerebbe sulle componenti del parlato alle frequenze più basse, perdendo quindi tutta l'informazione del suono.

In un processo di sintesi con LPC, sia l'onda di eccitazione, sia i coefficienti di filtro possono essere modificati per creare varianti del suono originale.

Spesso però, per migliorare l'efficienza, si impiegano metodi non lineari: il cepstrum e la sua variante mel-cepstrum, che verranno illustrati di seguito.

3.2.2 STIMA DELL'INVILUPPO SPETTRALE MEDIANTE CEPSTRUM

Il ***metodo del cepstrum*** consente la separazione di un segnale $y(n) = x(n) * h(n)$, basato sul modello sorgente-filtro, in cui la sorgente $x(n)$ passa attraverso un filtro descritto dalla risposta all'impulso $h(n)$. Lo spettro del segnale $y(n)$ risulta essere $Y(k) = X(k) \cdot H(k)$, che è il prodotto di due spettri (k è l'indice per le frequenze discrete): il primo della sorgente, il secondo del filtro. È difficile separare questi due spettri (nel senso della loro parte reale, ovvero lo spettro d'ampiezza, e quella immaginaria, la fase), mentre è più semplice estrarre l'inviluppo (reale) del filtro dal resto dello spettro, attribuendo tutta la fase alla sorgente.

L'idea del cepstrum si basa sulla proprietà del logaritmo $log(a \cdot b) = log(a) + log\,(b)$. Prendendo il logaritmo del modulo dello spettro $Y(k)$ si ricava

$$log|Y(k)| = log(|X(k) \cdot H(k)|) = log|X(k)| + log|H(k)| \qquad (3.18)$$

Considerando il grafico di $log|Y(k)|$ *come un segnale nel tempo* (ma ricordando che in realtà è in frequenza), *si possono distinguere due componenti: un'oscillazione veloce, dovuta alla struttura armonica (righe) dell'eccitazione, e un andamento più lento, corrispondente alle risonanze del filtro (inviluppo spettrale).* Con questa "visione" del problema, si possono separare le due componenti, la variazione veloce e quella lenta del segnale $log|Y(k)|$ (sempre interpretato come segnale nel tempo), mediante un filtro passa alto e uno passa basso, rispettivamente.

Un metodo per separare le due componenti consiste nell'usare la trasformata (nel nostro caso inversa) di Fourier. Pertanto

$$DFT^{-1}(log|Y(k)|) = DFT^{-1}(log|X(k)|) + DFT^{-1}(log|H(k)|) \quad (3.19)$$

La parte di $DFT^{-1}(log|Y(k)|)$ verso l'origine descrive l'inviluppo spettrale, quella distante l'eccitazione (in base alle considerazioni fatte precedentemente). In particolare, si noterà una specie di riga in corrispondenza della periodicità del $log|Y(k)|$ e quindi del periodo del suono.

A questo punto è possibile capire il *gioco di parole che sta alla base del termine* ***cepstrum***: infatti, questo termine *corrisponde a spec-trum con la prima parte letta all'inverso*. Analogamente si chiama **quefrency** l'ascissa di $DFT^{-1}(log|Y(k)|)$ invece che frequency.

Normalmente la DFT^{-1} produce un segnale nel tempo, ma qui invece va interpretata come frequenza (per la "supposizione" fatta in precedenza, considerando lo spettro come segnale temporale, e quindi la sua trasformata, benché in realtà nel dominio del tempo, viene vista come uno spettro nel dominio coniugato). In definitiva, il cepstrum è dato da

$$c(n) = DFT^{-1}(log|Y(k)|) \quad (3.20)$$

L'indice n di $c(n)$ è chiamato quefrency, dove ad alta quefrency (variazioni rapide nello spettro in dB) corrispondono valori di n grandi e viceversa. Pertanto, in base alle considerazioni precedenti, *si osserva come i valori bassi delle quefrency descrivano l'****inviluppo spettrale****, mentre quelli alti corrispondo all'****eccitazione*** *(sorgente).*

La separazione è ottenuta moltiplicando il cepstrum per una finestra passa basso $w_{LP}(n)$ nel dominio del cepstrum.

Ad esempio, per i suoni vocalizzati possiamo considerare che la formante più bassa F1 di un maschio adulto sia circa $F_1 = 270\ Hz$. Le oscillazioni dello spettro corrispondenti all'inviluppo non devono avere componenti sopra la quefrency $q_p = 3.7\ ms = 1/270\ Hz$. In definitiva, per suoni periodici, $n_c < n_p$, con n_p periodo in campioni. Per una frequenza di campionamento $f_s = 44.1\ KHz$, risulta $n_p = f_s/F_1 = 163\ campioni$. In pratica verrà scelto come soglia un valore leggermente inferiore. Si noti che per la voce femminile la separazione è più difficile: infatti, l'altezza media della voce femminile è di circa 256 Hz, mentre il formante più basso è a 310 Hz. Questi valori sono piuttosto vicini e, quindi, meno facilmente separabili.

3.2.3 ANALISI MEDIANTE MEL CEPSTRUM

Studi di psicoacustica hanno mostrato che la percezione umana del contenuto frequenziale del suono non segue una scala lineare, ma all'incirca logaritmica. Infatti, per ogni tono di f (misurata in Hz) corrisponde un'altezza soggettiva misurata su una scala chiamata ***scala di mel***.

Per ricavare i valori di questa scala si sfrutta una trasformazione non lineare che permette di determinare i valori in mel corrispondenti ai più utilizzati Hertz; i riferimenti per effettuare questo cambio di scala sono i punti 0 e 1000 Hz che non vengono alterati dalla trasformazione e corrispondono a 0 e 100 mel rispettivamente. Di seguito, la formula per effettuare la conversione

$$\begin{cases} f & per \;\; f \leq 1\,KHz \\ 2595 \cdot log\left(1 + \frac{f}{700}\right) & per \;\; f \;>\; 1\,KHz \end{cases} \tag{3.21}$$

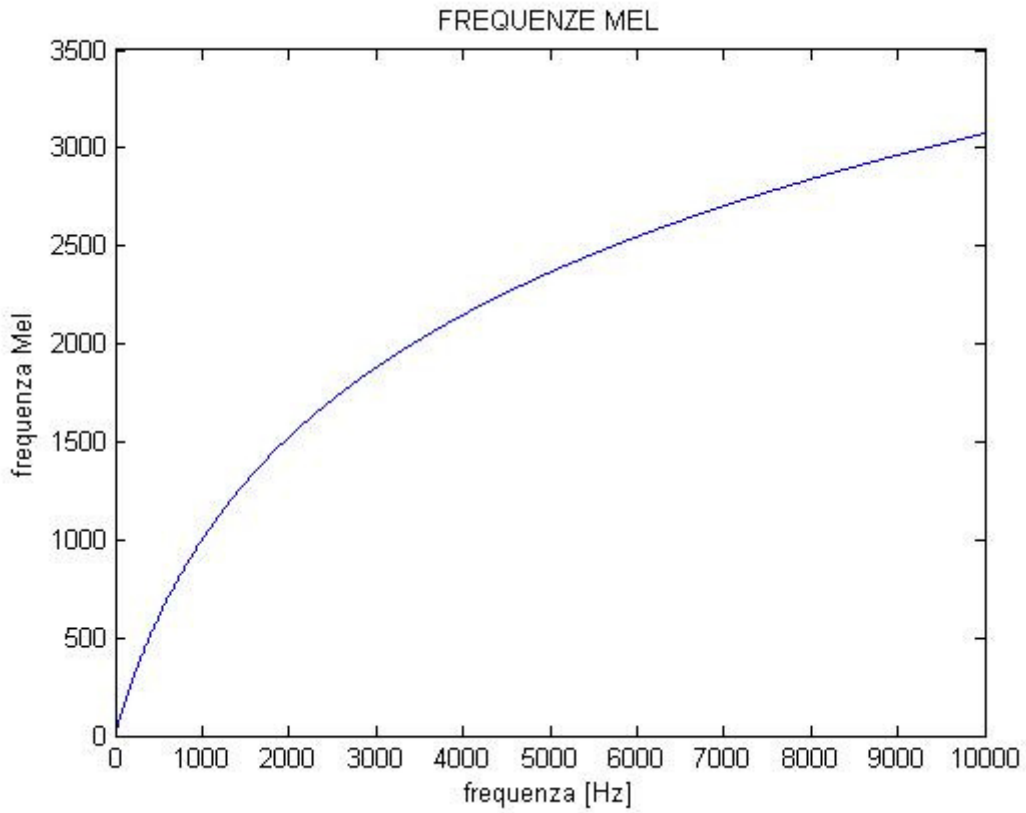


Figura 3.11: Grafico conversione scala tradizionale delle frequenze e Mel

Per applicare la scala mel al cepstrum, si utilizza un banco di filtri triangolari passabanda che riproducono in modo approssimato e digitale la funzione di trasferimento associata ad una

particolare conformazione dell'apparato fonatorio. I filtri sono disposti in modo da risultare equispaziati nel dominio delle frequenze Mel, mentre la loro larghezza di banda è pari alla distanza dalla frequenza centrale del filtro precedente moltiplicata per due. Il primo filtro parte da 0, pertanto la larghezza di banda dei filtri sotto 1000 Hz sarà 200 Hz, poi essa crescerà esponenzialmente.

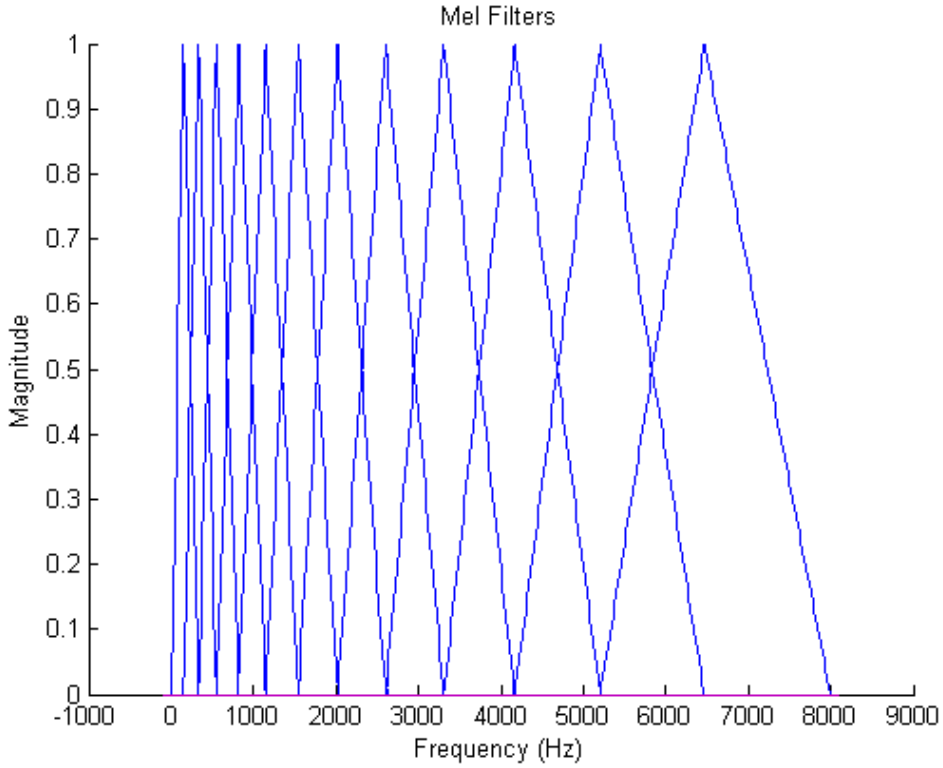


Figura 3.12: Banco di filtri per l'analisi del mel-cepstrum

Il mel-cepstrum vuole stimare l'inviluppo spettrale dell'uscita di questo banco di filtri. Sia quindi Y_n il logaritmo dell'energia in uscita dal canale n, attraverso la trasformata coseno discreta (DCT), ottengo i ***coefficienti mel-cepstrali MFCC*** (Mel Frequency Cepstral Coefficient) mediante l'equazione

$$c_k = \sum_{n=1}^{N} Y_n \cdot cos\left[k\left(n-\frac{1}{2}\right)\frac{\pi}{N}\right] \qquad k = 0, \dots, K \tag{3.22}$$

Si ricostruisce un inviluppo spettrale semplificato usando i primi K_m coefficienti, con $K_m < K$, analogamente a quanto visto per la stima dell'inviluppo con il cepstrum

$$\tilde{C}(mel) = \sum_{k=1}^{K_m} c_k \cdot cos\left(2\pi k \cdot \frac{mel}{B_m}\right) \quad (3.23)$$

dove B_m è la larghezza della banda analizzata, espressa in mel. Un tipico valore di K_m usato nella caratterizzazione e classificazione della musica è $K_m = 20$. Si noti che il coefficiente c_0 è il valore medio dei valori (in dB) dell'energia dei canali del banco di filtri. Inoltre, normalmente, viene trascurato quando si vuole fare un confronto della forma dell'inviluppo, normalizzato in energia, di vari suoni, ad esempio nei problemi di riconoscimento.

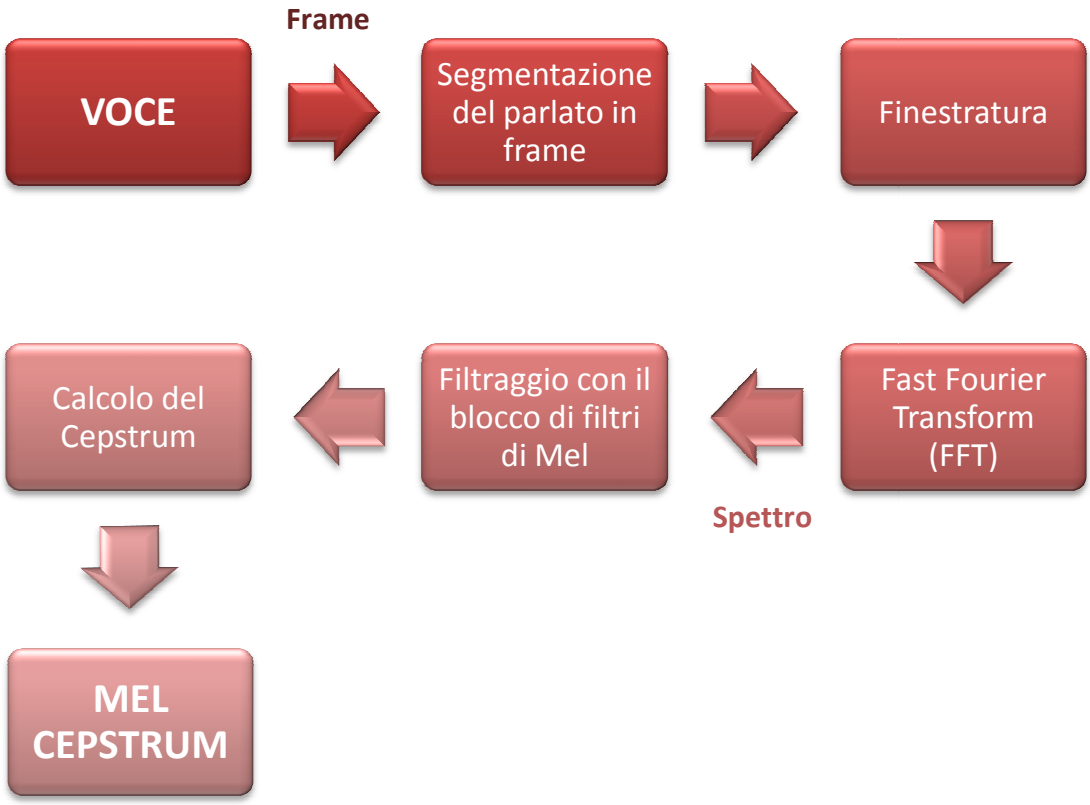


Figura 3.13: Diagramma logico delle operazioni di estrazione dei coefficienti mel-cepstrali

Il numero di filtri da utilizzare è un parametro del sistema, ma l'esperienza indica che una dozzina è sufficiente per garantire una buona qualità nel riconoscimento.

3.3 ATTRIBUTI A MEDIO E BASSO LIVELLO

Dall'analisi spettrale del suono vengono ricavati dei parametri che danno una descrizione delle caratteristiche fisiche a basso livello dello stesso. In particolare, si ricavano ampiezza, fase e frequenza istantanee di ogni parziale. A partire da questi parametri di basso livello è possibile dedurre una descrizione ad un livello di astrazione più alto, che possa servire per riconoscere la sorgente (in questo caso sono importanti anche l'armonicità, la rumorosità e la brillantezza), per ricavare altre informazioni da essa trasmesse o anche per sintetizzare il suono.

Oltre agli attributi istantanei del suono, spesso sono utili le loro ***derivate***: *la derivata prima descrive la tendenza dell'evoluzione temporale in quell'istante*; talvolta viene presa in considerazione anche la derivata seconda, che rappresenta l'accelerazione dell'evoluzione temporale. Nei segnali discreti, la derivata viene sostituita dal calcolo della differenza tra il valore corrente ed il precedente $d(n) = p(n) - p(n-1)$: si tratta però di un metodo abbastanza rumoroso; per questo motivo, ad ogni istante, si determina la parabola che meglio approssima, in termini di minimi quadrati, il valore del parametro che si sta analizzando in tre punti adiacenti. Si farà dunque riferimento alla derivata (prima, ed eventualmente seconda) della parabola calcolata: analizzando i parametri e la loro variazione nel tempo è possibile rendere più agevole la segmentazione del segnale in ingresso in frame che siano il più omogenei possibile.

Le caratteristiche definite con il termine "***attributi a basso livello***" sono quei parametri che si ricavano dalla rappresentazione sinusoidale del suono, con l'aggiunta di una componente di rumore, spesso definita con il termine residuo. Definendo con $x(n)$ il suono originario, esso, secondo la rappresentazione precedentemente indicata, può essere scomposto nelle sue due componenti: sinusoidale, $x_S(n)$, e rumore, $x_R(n)$; si avrà quindi $x(n) = x_S(n) + x_R(n)$, in cui $x_S(n) = \sum_{i=1}^{I} a_i \cdot cos[2\pi n \cdot f_i(n)/F_S + \Phi_i(n)]$. In questa espressione a_i rappresenta l'ampiezza, in scala lineare , della i-esima armonica, f_i la sua frequenza e Φ_i la fase istantanea. Tutti questi parametri sono tempo varianti: essi vengono ricavati dall'analisi di un singolo frame e generalmente vengono riferiti temporalmente al centro del frame stesso oppure alla fine (nel caso in cui si abbia un'analisi real time). Si tratta quindi di attributi istantanei del suono in esame. Tra questi, si ricordano

- *Ampiezza* totale *della componente sinusoidale*, espressa in decibel ed ottenuta sommando tutte le parziali contenute in un singolo frame

$$A_{Stot} = 20log(\sum_{i=1}^{I} a_i) \tag{3.24}$$

- *Ampiezza complessiva del residuo*, anch'essa espressa in dB e ricavata dalla somma di tutti i valori assoluti del rumore nel frame

$$A_{Rtot} = 20log(\sum_{n=0}^{M-1}|x_R(n)|) \tag{3.25}$$

- Peso dell'armonica i-esima rispetto al valore complessivo della componente sinusoidale (ovvero A_{Stot} espresso in scala lineare)

$$w_i = \frac{a_i}{\sum_{i=1}^{I} a_i} \tag{3.26}$$

- La frequenza fondamentale, che può anche essere ottenuta come media pesata delle frequenze (delle armoniche) normalizzate di tutte le armoniche

$$f_0 = \sum_{i=1}^{I} \frac{f_i}{i} \cdot w_i \tag{3.27}$$

Nel caso in cui il suono sia perfettamente periodico, ciò implica che tutte le frequenze delle armoniche risultano essere multiple della fondamentale, ovvero, per la generica frequenza i-esima, $f_i = i \cdot f_0$. Questa considerazione, per suoni reali, è valida solo in modo approssimativo.

3.4 ATTRIBUTI A LIVELLO SUPERIORE

Attributi di livello superiore rispetto a quelle descritte al paragrafo precedente sono le caratteristiche spettrali del suono, anch'esse ricavate dal singolo frame e quindi considerate istantanee, come fatto in precedenza per quelle di basso livello. Tra questi i più utilizzati vengono illustrati di seguito:

- *Disarmonicità*, ovvero la differenza, pesata, tra la frequenza di un'armonica e il valore che essa avrebbe nel caso in cui il suono in ingresso fosse esattamente periodico, ovvero

$$HD = \sum_{i=1}^{I} |f_i - i \cdot F0| \cdot w_i \tag{3.28}$$

- *Rumorosità* (noiseness), data dal rapporto tra l'energia della componente rumorosa e l'energia complessiva del segnale in ingresso

$$Noiseness = \frac{\sum_{n=0}^{M-1}|x_R(n)|}{\sum_{n=0}^{M-1}|x(n)|} \quad (3.29)$$

- *Deviazione spettrale delle armoniche* (Harmonic Spectral Deviation)

$$HDEV = \frac{1}{I}\sum_{i=1}^{I}[a_i - spec\_env(f_i)] \quad (3.30)$$

Dove $spec\_env(f_i)$ è l'inviluppo spettrale stimato con uno dei metodi visti sopra, calcolato alla frequenza della i-esima armonica

- Tristimulus, coefficienti introdotti per pesare in modo differente le armoniche appartenti a diverse zone del dominio coniugato: fondamentale (T1), dalla seconda alla quarta (T2), le rimanenti (T3); questi parametri sono definiti dalla seguenti relazioni

$$T1 = \frac{a_1}{\sum_i a_i} \qquad T2 = \frac{a_2+a_3+a_4}{\sum_i a_i} \qquad T3 = \frac{\sum_{i=5}^{I} a_i}{\sum_i a_i} = 1 - T1 - T2 \quad (3.31)$$

- *Spectral roll-off*, definito come la frequenza, generalmente indicata con R_s, al di sotto della quale è concentrata l'85% della distribuzione delle ampiezze. Si tratta di una misura alternativa che consente di determinare, seppur in modo indicativo, la forma dello spettro

$$roll - off = \sum_{k=1}^{R_s}|X(k)| = 0.85 \cdot \sum_{k=1}^{N-1}|X(k)| \quad (3.32)$$

3.5 ATTRIBUTI DEL SEGMENTO SONORO

Oltre al calcolo e all'analisi dei parametri in un determinato istante temporale, ovvero relativamente ad un certo frame, è molto importante riuscire a caratterizzare anche l'evoluzione nel tempo di questi elementi, ricavando in questo modo un attributo del frame. Con il termine segmento si fa riferimento ad una porzione di suono avente caratteristiche omogenee.

Per poter effettuare questa segmentazione del suono, è necessario esaminare i parametri ad un livello di segnale, cercando di individuarne la traiettoria, cioè, come già detto in precedenza, la variazione di un particolare elemento caratteristico. Indicano con par il parametro che si sta analizzando, è importante calcolarne

- La derivata al frame j-esimo

$$der[par(j)] = \frac{par(j) - par(j-1)}{H/f_s} \tag{3.33}$$

con H l'hop size e f_s la frequenza di campionamento;

- Media pesata del parametro su tutti i frame considerati

$$media(par) = \frac{\sum_j par(j) \cdot amp(j)}{\sum_j amp(j)} \tag{3.34}$$

- Varianza pesata del parametro, sempre relativamente a tutti i frame

$$var(par) = \frac{\sum_j [par(j) - media(par)]^2 \cdot amp(j)}{\sum_j amp(j)} \tag{3.35}$$

4. ALGORITMI

Il *riconoscimento automatico del parlato* (ASR) *può essere visto come un compito di* ***pattern recognition***, applicando le tecniche sviluppate nel settore della robotica (identificazione di immagini) o comunicazioni di dati (conversione di segnali analogici in informazione digitale). In pratica, *il riconoscimento della voce richiede una mappatura tra il parlato e il testo,* in modo che ogni possibile forma d'onda in ingresso sia identifica con il suo testo corrispondente. Relativamente a questo ambito, il termine testo viene usato per identificare parole, frasi o altre unità linguistiche che il parlante pensa e verbalizza.

Tutte le possibili varianti di pattern recongition, incluso quelle adottate nello speech recognition, *prevedono due fasi: il* ***training*** *e il* ***riconoscimento*** vero e proprio. La prima fase stabilisce una memoria di referenza o un dizionario di modelli del parlato, che sono assegnati alle etichette del testo (cioè alle rappresentazioni delle parole). In sistemi indipendenti dal parlante, la fase di addestramento è prodotta durante lo sviluppo dell'applicazione, mentre in quelli più diffusi nel mercato, dipendenti dall'utente, consentono all'utilizzatore finale di addestrare lui stesso il dispositivo. La fase di riconoscimento automatico, invece, tenta di assegnare un'etichetta ai modelli di input sconosciuti.

Nel proseguio del capitolo verranno illustrati i concetti generali che stanno alla base dei metodi di pattern recognition, tecnologia su cui si fondano la quasi totalità degli algoritmi e dei sistemi di riconoscimento del parlato. Si procederà quindi con un'analisi più dettagliata delle tecniche che la demo board è in grado di implementare, ossia le catene di Markov nascoste (Hidden Markov Model) e le reti neurali (Artificial Neural Network).

4.1 PATTERN RECONGNITION

*Il riconoscimento di pattern è dunque una sottoarea dell'****apprendimento automatico***: esso consiste nell'analisi e individuazione di un ***pattern*** (ossia un *insieme di oggetti, processi, eventi, caratteristiche, che hanno componenti sia deterministiche che aleatorie, e che permettono di identificare in modo univoco un particolare campione o una sequenza in ingresso* al generico sistema) all'interno di dati grezzi al fine di identificarli.

Il pattern recognition ha come obiettivo quello di apprendere un classificatore di dati, basato sulla conoscenza a priori della classe di appartenenza dell'input oppure su informazioni statistiche da esso estratte. Con ***classe*** (o categoria) si fa riferimento ad un *insieme di differenti pattern*

accomunati da parametri simili, spesso indicazione che la sorgente che ha originato i dati è la medesima. Gli oggetti da classificare sono tipicamente gruppi di misure od osservazioni, che definiscono dei punti in un appropriato spazio multidimensionale.

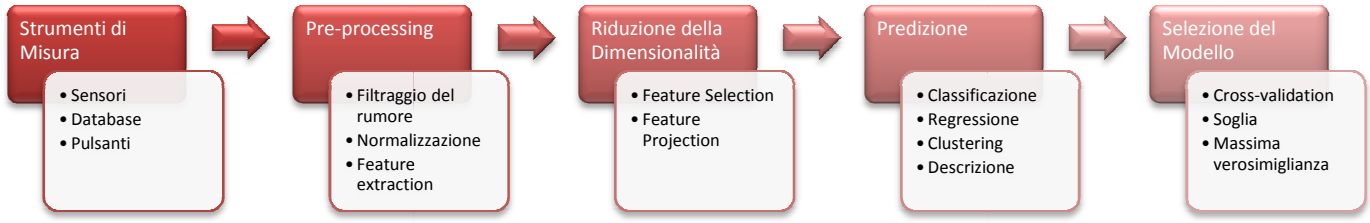


Figura 4.1: Schema elementare del funzionamento delle tecniche di pattern recognition

Questo spazio viene chiamato ***feature space***, mentre i punti che in esso rappresentano i particolari oggetti sono definiti scatter plot. Le coordinate sono le cosiddette ***features***, ovvero *tutti gli aspetti distintivi, caratteristici, di un particolare oggetto, di una sequenza di dati in ingresso, e possono essere sia simboliche che numeriche*. Una loro combinazione viene generalmente rappresentata con un vettore colonna di dimensione d (nel caso in cui lo spazio sia $d-dimensionale$) a cui si fa riferimento con il termine inglese ***feature vector***.

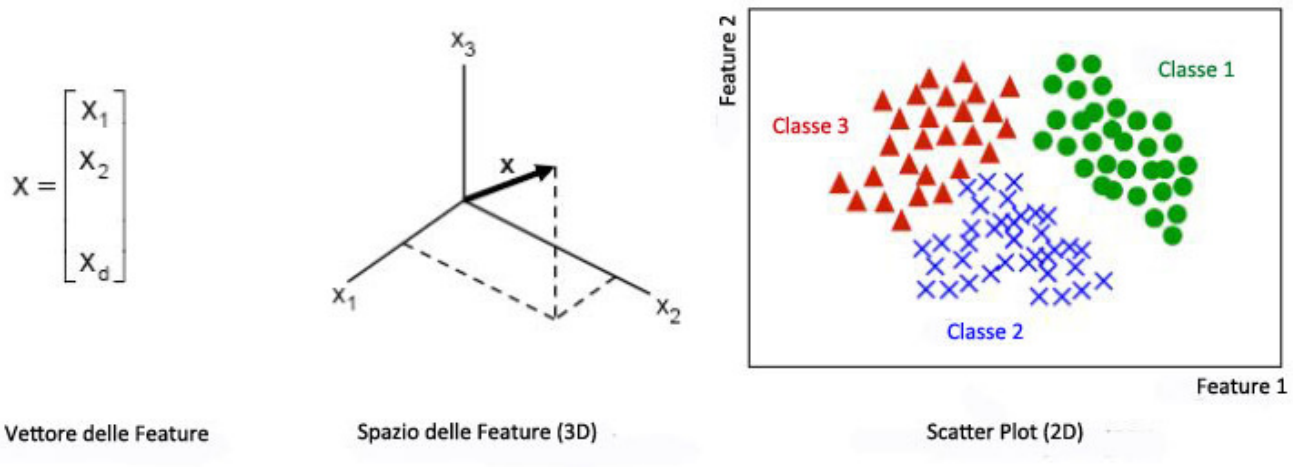


Figura 4.2: Illustrazione dello spazio delle feature e dei concetti ad esso associati

La qualità del vettore delle features è legata alla (relativa) semplicità con cui è possibile discriminare punti appartenenti a categorie differenti, come si può notare dalla figura che segue

Figura 4.3: Differenza tra la scelta di parametri e caratteristiche che consentono una buona discriminazione degli stessi e feature che non permettono una corretta identificazione

I pallini verdi e le "x" blu della figura 4.3 rappresentano due classi distinte: nel caso di sinistra, tutti i punti disegnati con una "x" sono accorpati e ben distinti da quelli indicati col pallino. Ciò significa che le features estratte dal segnale in ingresso permettono di discriminare con una buona efficienza campioni appartenenti a categorie differenti, cosa che non si verifica nella figura di destra, in cui pallini verdi e "x" blu si mischiano e quindi il sistema di riconoscimento (che, salvo in rari casi, non conosce a priori la categoria di appartenenza di un determinato pattern) non è in grado di assegnare ad ogni campione la sua classe, oppure lo farà con una probabilità dì errore molto elevata.
Il riconoscimento ha quindi il compito di associare ad ogni pattern una categoria, compresa in un insieme di classi predefinite; un termine molto usato al posto di riconoscimento, infatti, è ***classificazione*** (che rende anche più chiaro il concetto che sta alla base di questa operazione).

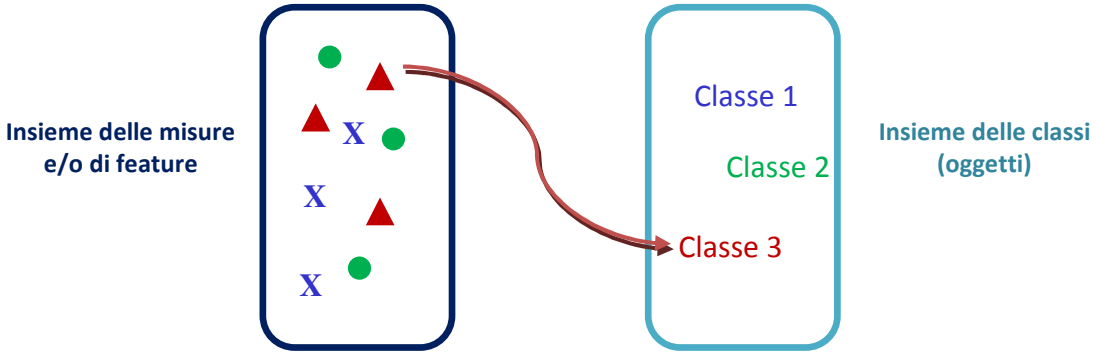


Figura 4.4: Processo di classificazione

La catena di elaborazione dei dati in ingresso è analoga a quella descritta all'inizio del capitolo 3: innanzitutto si procede con la pre-elaborazione della sequenza di input, proseguendo con l'estrazione delle features ed infine si applica la classificazione. Di seguito lo schema a blocchi più semplice del funzionamento logico di un sistema di pattern recogniton:

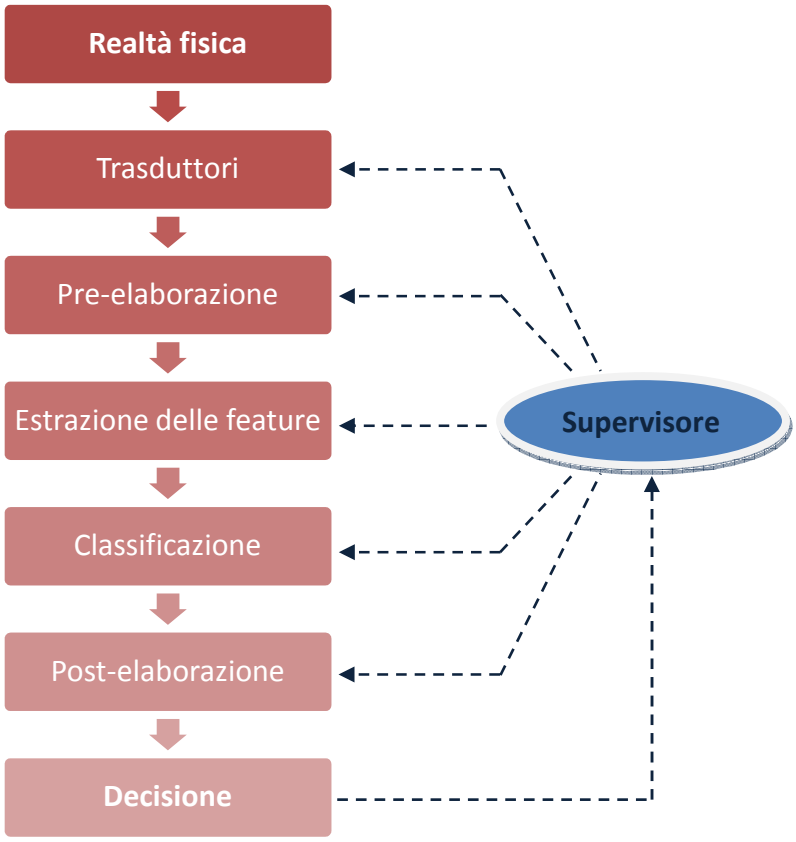


Figura 4.5: Schema del riconoscimento (la parte tratteggiata è opzionale, a seconda del particolare sistema scelto)

Fino ad ora si è fatto riferimento al solo obiettivo di classificare dei dati sconosciuti, ma nella realtà i ***problemi di predizione*** possono essere differenti. In particolare, si ricordano

- **Classificazione**

  Il problema di PR consiste nell'assegnare un determinato oggetto ad una ben precisa classe; il risultato di questa operazione sarà dunque un'*etichetta* (ad esempio, definire se un prodotto è "accettabile" o "da scartare" in un test di qualità)

- **Regressione**

  Si tratta di una generalizzazione del caso precedente, in cui l'uscita restituita dal sistema è un valore reale (un caso intuitivo è la previsione dell'andamento del fatturato di un'azienda, basandosi sulle performance precedenti e sugli indicatori di mercato)

- **Clustering**

  L'obiettivo è raggruppare i dati di input in gruppi caratterizzati dal fatto che gli oggetti che vi appartengono hanno delle caratteristiche significative comuni; il sistema restituisce un raggruppamento (a volte gerarchico) dei dati (ad esempio, la classificazione delle specie animali)

- **Descrizione**

  Utilizzato quando si rende necessario rappresentare la sequenza originale in funzione di una serie di parametri "primitivi" (il caso più evidente e di ambito comune è il segnale dell'ECG, espresso in funzione di alcuni parametri tipici: l'onda P, il complesso QRS e l'onda T)

Un'ulteriore distinzione che si può fare circa i sistemi PR è relativa al *tipo di addestramento*: questo può essere ***supervisionato***, nel qual caso viene fornita, unitamente a ciascun campione di training, anche la sua etichetta (classe di appartenenza), cercando in questo modo di ridurre l'errore di classificazione per l'intero insieme dei dati di addestramento. L'efficacia dell'algoritmo di training si misura in termini di capacità di raggiungere, in un tempo minimo, una soluzione semplice, ottima e soprattutto stabile (robusta al rumore e a piccole variazioni temporali del segnale).
Al contrario, l'addestramento ***non supervisionato*** non richiede campioni di training e consiste nell'identificazione, nello spazio delle features, di gruppi omogenei dei dati di ingresso: in sostanza, esegue un auto-apprendimento.
Lo scopo di un classificatore è dunque quello di partizionare lo spazio delle features in regioni di decisioni che permettano di determinare le varie classi di appartenenza degli oggetti in ingresso.
Esistono tre principali approcci di PR:

- **Statistico**

  I campioni sono classificati rispettando un definito modello statistico delle features; quest'ultimo è costituito da una famiglia di densità di probabilità condizionata delle

classi, indicate con $P(x|c_i)$, cioè la probabilità di avere il feature vector x data la classe c_i.

- **Neurale**

  La classificazione viene effettuata in base alla risposta di una rete neurale ad uno stimolo in input (sequenza di campioni, pattern); la "conoscenza" è memorizzata nelle connessioni e nei pesi tra le varie sinapsi. Una rete neurale per il pattern recognition è addestrabile, non algoritmica e funzionante secondo il modello della black-box.

- **Sintattico**

  I patterns sono classificati secondo delle misure sulla somiglianza e la "knowledge" è rappresentata da delle relazioni descritte mediante grafi. È il metodo più utilizzato nel caso di descrizione.

4.2 HIDDEN MARKOV MODEL

I modelli Markoviani sono metodi statistici che hanno visto una larga diffusione in questi ultimi anni, in particolare a partire dagli Anni Novanta, grazie al fatto che possono essere utilizzati in moltissime applicazioni. Questa ampia varietà di impieghi è resa possibile dalla solida base matematica su cui questi metodi si fondano.

Nonostante la loro diffusione sia un fenomeno abbastanza recente, essi furono introdotti già nel 1907 dal matematico russo Andrei Andreevich Markov. *Un **modello di Markov**, detto anche **catena di Markov** (Markov Chain), è un metodo matematico usato per rappresentare la tendenza (probabilità) che un determinato evento avvenga dopo che se ne sia verificato un altro o una serie di altri eventi ben precisi*. In origine, il matematico ideò questa tecnica per poter dedurre delle considerazioni circa l'ortografia russa.

I modelli Markoviani sono molto utili per rappresentare delle famiglie di sequenze caratterizzate da particolari proprietà statistiche. Elemento fondamentale è la ***matrice di transizione***, qui indicata con la lettera A, *rappresentante la probabilità, per ogni stato del modello, di passare ad uno stato successivo o di rimanere nello stesso*:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \tag{4.1}$$

In cui i singoli coefficienti a_{ij} sono ovviamente tutti positivi (o al più nulli) e godono della proprietà per cui $\sum_j a_{ij} = 1$.

Un ***processo stocastico markoviano*** *è un processo stocastico nel quale la probabilità di transizione*, che determina il passaggio da uno stato del sistema ad un altro, *dipende unicamente dallo stato immediatamente precedente* e non dalla "storia" che ha permesso di giungere alle condizioni attuali: questa è la proprietà fondamentale che caratterizza il modello. Qualora ciò non fosse verificato, si tratterebbe di un processo non markoviano.

In modo formale, questa proprietà (definita anche ***assenza di memoria***) si può esprimere con la seguente equazione

$$P[X(t_{n+1}) \leq x_{n+1} \mid X(t_n) = x_n, X(t_{n-1}) = x_{n-1}, \cdots, X(t_0) = x_0] = P[X(t_{n+1}) \leq x_{n+1} \mid X(t_n) = x_n] \tag{4.2}$$

Una ***catena di Markov*** *è un processo markoviano con spazio degli stati discreto* e che gode della proprietà appena descritta. L'insieme degli stati può essere finito (numerabile), e in questo caso si parla di catena a stati finiti, oppure infinito (continuo). Inoltre, vi posso essere catene tempo-continue o tempo-discrete, a seconda del dominio di definizione della variabile.

Molto importante e molto utilizzata in varie applicazioni, è una particolare classe di ***catene markoviane***: quelle ***omogenee***. Questi particolari processi sono caratterizzati dal fatto che *la probabilità di transizione è indipendente dal tempo, ma solo ed esclusivamente dallo stato del sistema al tempo immediatamente precedente*. In conclusione, la probabilità di transizione dipende solamente dalla distanza temporale tra due punti, indipendentemente dall'origine dell'asse dei tempi (ossia non conta il valore assoluto del tempo, ma quello relativo all'ultimo stato in cui si trovava la catena). Da un punto di vista matematico, questa definizione può essere espressa con la seguente equazione

$$P(X_{n+1} = x_{n+1} \mid X_n = x_n) = P(X_n = x_n \mid X_{n-1} = x_{n-1}) \tag{4.3}$$

Tuttavia, i sistemi reali che possono essere modellati con catene markoviane omogenee sono molto difficili da trovare: la probabilità di transizione da uno stato ad un altro è fortemente condizionata da molti fattori nella realtà; limitando però l'osservazione del sistema ad un intervallo di tempo abbastanza ristretto, sufficientemente piccolo in modo che le caratteristiche della variabile

considerata (e quindi del segnale in esame) possano considerarsi costanti, può essere applicato questo modello.

Come già osservato all'inizio del paragrafo, *elemento basilare nella descrizione di una catena di Markov (omogenea) a stati finiti, in cui l'insieme al quale quest'ultimi appartengono ha cardinalità pari ad N, è la matrice di transizione* $A \in \mathbb{R}^{N\,x\,N}$; tuttavia, essa non è sufficiente a descrivere l'intera catena: è necessario introdurre un ***vettore delle probabilità iniziali***, rappresentante la *probabilità che inizialmente* (all'istante $t = 0$ in cui si inizia l'osservazione) il *modello si trovi in ciascuno degli N stati*. Tale vettore è indicato con $\pi_0 \in \mathbb{R}^N$.

Una MC omogenea è univocamente definita dalla coppia $(\boldsymbol{A, \pi_0})$. Infatti, una volta noti questi due parametri, è possibile determinare la distribuzione di probabilità al generico istante t_n, con l'espressione

$$\pi_n = A^n \cdot \pi_0 \tag{4.4}$$

Di conseguenze, il modello risulta essere definito da

- un set di N stati, $S = \{s_1, s_2, s_3, \cdots, s_N\}$;
- un insieme di probabilità di transizione, raggruppate nella matrice di transizione, in cui sono riportate le probabilità che avvenga il passaggio dallo stato s_i a s_j. Si ha quindi che $a_{ij} = P[x(t) = j \mid x(t-1) = i]$;
- una distribuzione di probabilità, indicata con la lettera π.

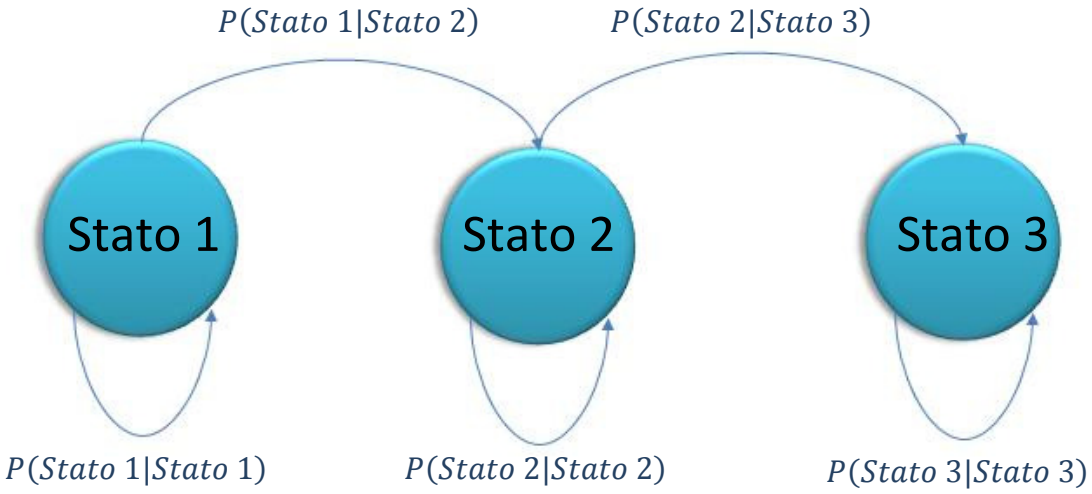


Figura 4.6: Catena di Markov

Fatta questa premessa sulle caratteristiche generali delle catene markoviane, verrà ora introdotta la variante che viene maggiormente impiegata nei dispositivi di riconoscimento vocale: i ***modelli di Markov nascosti*** (***Hidden Markov Model***, ***HMM***); *si tratta di comuni catene markoviane, con la solo differenza che gli stati non sono direttamente osservabili*, da cui il termine "nascosti", in particolare:

- la catena ha un certo numero di stati;
- ogni stato genera un evento con una certa distribuzione di probabilità che dipende unicamente dallo stato attuale;
- ***l'evento è osservabile ma lo stato no***.

I modelli HMM, che sono *processi doppiamente stocastici* (infatti sono previste due probabilità: una di transizione tra stati della catena e una distribuzione di probabilità tipica di ogni singola unità), descrivono le variazioni intrinseche del segnale vocale (e delle sue feature), dando vita ad un modello statistico del parlato.

Quando le persone pronunciano una stessa parola, il segnale acustico non è identico per ognuna di esse, anche se la struttura linguistica che sta alla base, ovvero la pronuncia, la sintassi e la grammatica, sono le medesime. Dunque, *i modelli di Markov nascosti restituiscono una misura probabilistica che sfrutta una catena markoviana per rappresentare la struttura linguistica del parlato e un set di distribuzioni di probabilità per tenere in considerazione la variabilità del suono che dà luogo a quella particolare parola*.

Dato un insieme di termini noti, in grado di approssimare tutte le possibili variazioni delle parole di interesse, necessarie per l'addestramento (cioè il ***training set***), è possibile determinare il miglior modello (set di parametri) che identifica una ben precisa parola o frase (in generale, una ***utterance***[2]) del vocabolario; questo modello verrà poi impiegato in fase di riconoscimento per valutare la verosimiglianza di un suono sconosciuto (segnale in ingresso) con tutti gli elementi del training set, consentendo quindi di determinare quale sia la realizzazione che meglio rappresenta il suono stesso.

La misura statistica restituita dal modello nascosto markoviano è la componente essenziale di ciascun sistema di riconoscimento vocale basato sull'approccio di pattern recognition statistico ed ha la sua origine nel famoso teorema di Bayes.

[2] Termine inglese che letteralmente significa "espressione" e che verrà usato spesso nell'elaborato per indicare una generico suono, sia esso una singola sillaba, una parola o una frase intera

Analizzando la catena che rappresenta il modello di una utterance, si nota come essa sia costituita da molte sotto-unità, ognuna delle quali rappresenta un particolare fonema di cui è costituita la parola (compresi gli intervalli di silenzio, che possono manifestarsi all'inizio o alla fine di una frase, ma anche all'interno di essa, qualora, ad esempio, l'utente esiti a parlare), e per ogni sotto-sistema è prevista una particolare distribuzione di probabilità per descrivere le varie possibilità con cui può essere pronunciata una stessa parola, necessaria per determinare la verosimiglianza complessiva del modello.

Quest'ultima viene calcolata come somma di tutte le singole verosimiglianze relative a ciascuna unità, dopo che è stato effettuato un preventivo allineamento temporale (definito *Dynamic Time Warping*), in modo che si abbia il miglior matching possibile tra la sequenza del modello e quella derivata dalle features estratte dal segnale in ingresso.

Ad ogni istante, quindi, è possibile scegliere l'unità che maggiormente rispecchia il segmento di parlato tra un certo numero delle stesse, basandosi sulla massima verosimiglianza: *poiché il numero di possibili alternative può essere molto elevato, è necessario adottare particolari algoritmi computazionali per risolvere il problema di riconoscimento*.

Per i modelli di Markov nascosti si possono individuare tre problemi estremamente interessanti (definendo X una particolare sequenza e M il modello):

- data la sequenza X, qual è la probabilità che essa sia osservata sul modello M?
- nota X, qual è la sequenza ottima di stati del modello M che ha prodotto X?
- dato un training set X, come possiamo stimare i parametri del modello M per massimizzare $P(X|M)$, ovvero la verosimiglianza?

È nel nostro interesse, ai fini dell'elaborato, analizzare più in dettaglio l'ultimo di questi possibili impieghi delle HMM: in questo caso, la *problematica* è l'***addestramento del modello***. Si tratta di stimare i parametri dello stesso A, B, π_0 (con A matrice delle probabilità di transizione, B l'insieme delle distribuzioni di probabilità per ciascuno stato e π_0 la distribuzione iniziale) che, data una sequenza finita di osservazioni O (ad esempio, il training set stesso), massimizzano $P(O|modello)$.

Per migliorare le performance dell'addestramento talvolta, in caso di modelli complessi, si ricorre al cosiddetto ***Viterbi learning***: l'idea di base è quella di calcolare un solo path, o un gruppo dei più probabili, invece di valutare tutte le possibili sequenze del modello, come avverrebbe normalmente. Tuttavia, la semplificazione introdotta con l'algoritmo di Viterbi, porta sì ad un aumento delle

prestazioni (nella pratica si aumenta la velocità di calcolo), ma comporta anche una qualità dell'addestramento inferiore rispetto al caso in cui tutti i path vengano presi in considerazione.

4.2.1 ALGORITMO DI VITERBI

Nella sua forma più generale, l'algoritmo di Viterbi può essere visto come una *soluzione del problema della stima a massima probabilità a posteriori* (*MAP*) di una sequenza di stati di un processo stocastico markoviano con un numero finito di stati e a tempo discreto. In altre parole, esso viene utilizzato per trovare la miglior sequenza di stati (detta *Viterbi path*) in una sequenza di eventi osservati in un processo di Markov.

Il problema della decisione della parola pronunciata, come già accennato in precedenza, si basa sul criterio della massima verosimiglianza: la soluzione, infatti, consistere nello scegliere, tra tutte le possibili sequenze di fonemi, quella che rende massima la probabilità a posteriori. Si noti che non tutte le sequenze di fonemi sono possibili e, ovviamente, il loro numero cresce in modo esponenziale con l'aumentare del numero di possibili parametri in ingresso. Pertanto, la complessità computazionale diventa ben presto insostenibile, a meno che non si adottino dei meccanismi che consentano di ridurre la complessità di calcolo a valori accettabili.

Per facilitare la comprensione del problema, si introduce il ***diagramma a traliccio***, *una rappresentazione grafica bidimensionale (in funzione anche del tempo) del classico processo di Markov*: ad ogni nodo corrisponde uno stato e le transizioni vengono indicate con dei collegamenti tra i rispettivi nodi a cui si riferiscono, in base al diagramma a stati associato alla HMM; inoltre, tutti i nodi relativi ad uno stesso istante temporale sono allineati, come si osserva nella figura

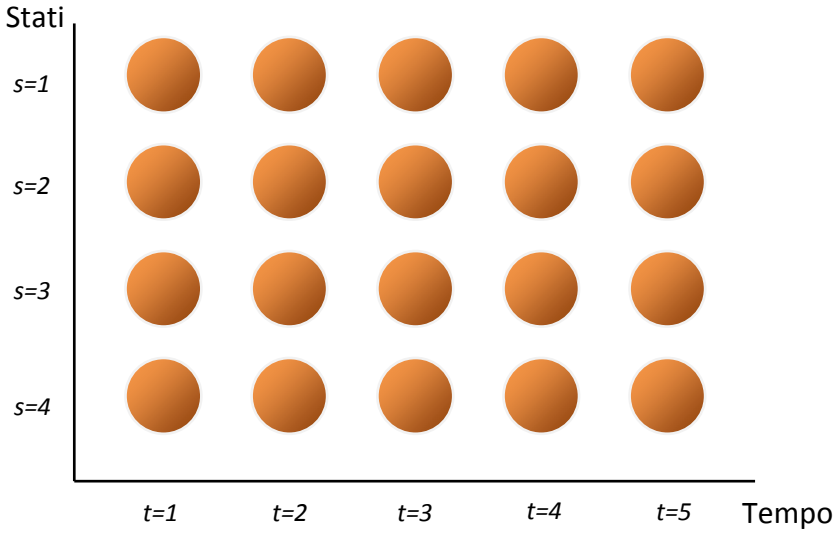


Figura 4.7: Grafico a traliccio (vuoto)

Il problema MAP si rivela essere identico al problema di ricerca del cammino minimo: l'algoritmo di Viterbi risulterà quindi essere una soluzione ricorsiva.

La proprietà fondamentale del traliccio è che ad una possibile sequenza di stati X corrisponde un unico cammino attraverso il grafo, e viceversa.

Data un'osservazione o, ad ogni cammino può essere associata una lunghezza proporzionale a $-\ln\big(P(x,o)\big)$, dove x è la sequenza di stati associata a quel determinato cammino. Questo consente di risolvere il problema di ricerca della serie di stati per cui $P(x|o)$ è massima, o, equivalentemente, per la quale $P(x,o) = P(x|o)P(o)$ (probabilità congiunta) è massima, trovando il cammino la cui lunghezza è minima[3].

Assegnando ad ogni arco (transizione) la lunghezza

$$\lambda(\xi_{k+1}) \triangleq -lnP(x_{k+1}|x_k) - lnP(o_k|\xi_{k+1}) \tag{4.5}$$

avendo definito, per comodità ξ_k la transizione dallo stato k a $k+1$. Allora la lunghezza totale del cammino corrispondente ad una specifica sequenza vale

$$-\ln P(x,o) = \sum_{k=0}^{K-1} \lambda(\xi_k) \tag{4.6}$$

Prima di illustrare l'algoritmo vero e proprio è necessaria un'ulteriore osservazione: denotiamo con x_0^k un segmento che consiste degli stati fino all'istante k (ossia la sequenza $x = (x_0, x_1, \dots, x_k)$), il quale corrisponde ad una porzione di cammino che inizia nel nodo x_0 e termina in x_k. Per ogni nodo x_k corrispondente ad un certo istante k, in generale, ci sono diversi segmenti di cammino, ciascuno con una certa lunghezza: la porzione di percorso minimo viene chiamata ***segmento sopravvissuto***, e verrà denotato con $\hat{x}(x_k)$. Per ogni istante $k > 0$ ci sono N sopravvissuti (con N pari alla cardinalità degli stati), uno per ogni x_k. Quindi, per ogni tempo discreto, è sufficiente memorizzare solo i sopravvissuti $\hat{x}(x_k)$ e le loro lunghezze $\Gamma(x_k)$.

Per passare all'istante successivo $k+1$ è necessario calcolare le lunghezze dei segmenti e, per ogni nodo x_{k+1}, selezionare il segmento il cui cammino è minimo, diventando quindi il sopravvissuto di $k+1$.

[3] infatti $\ln\big(P(x,o)\big)$ è una funzione monotona su $P(x,o)$ e c'è una corrispondenza uno a uno attraverso i cammini e la sequenza

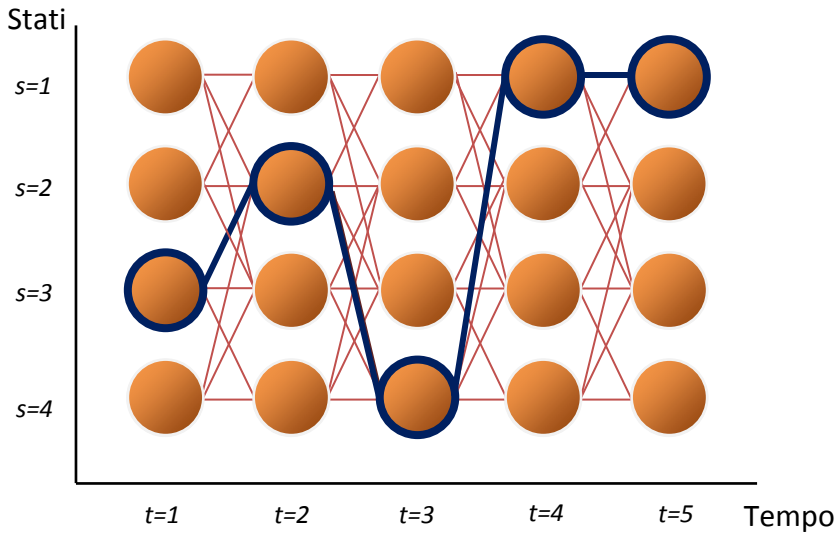


Figura 4.8: Esempio di grafico a traliccio, in cui gli stati e le transizioni evidenziate in blu rappresentano il cammino minimo

Formalmente, l'algoritmo di Viterbi può essere definito come segue:

- **Memorizzazione**:

  k (indice temporale)

  $\hat{x}(x_k)\ ,\ 1 \le x_k \le N$ (sopravvissuto terminante in x_k)

  $\Gamma(x_k)\ ,\ 1 \le x_k \le N$ (lunghezza del sopravvissuto)

- **Inizializzazione**:

  $k = 0$

  $\hat{x}(x_0) = x_0\ ;\ \hat{x}(x_m)$ arbitrario , per $m \neq 0$

  $\Gamma(x_0) = 0\ ;\ \Gamma(x_m) = \infty$, per $m \neq 0$

- **Ricorsione**:

  Calcola $\Gamma(x_{k+1}, x_k) \triangleq \Gamma(x_0) + \lambda(\xi_{k+1})$ per ogni ξ_{k+1}

  Trova $\Gamma(x_{k+1}) = min_{x_k}\, \Gamma(x_{k+1}, x_k)$ per ogni x_{k+1}

  Memorizza $\Gamma(x_{k+1})$ ed il corrispondente sopravvissuto $\hat{x}(x_{k+1})$

  Incrementa k e ripeti fino a quando $k = K$

Con una sequenza finita di stati, l'algoritmo termina all'istante K, e il cammino minimo è stato memorizzato in $\hat{x}(x_K)$. Nella pratica, però, sono necessarie alcune semplici modifiche, soprattutto nel caso in cui la sequenza degli stati sia molto lunga, o addirittura infinita: è necessario troncare i sopravvissuti ad una certa lunghezza, ovvero non valutando tutti i k precedenti a quello corrente, ma solamente gli ultimi. Inoltre, se k diventa grande, è necessario normalizzare le lunghezze $\Gamma(x_m)$ sottraendo una costante; infine, lo stato iniziale potrebbe non essere noto: in questo caso si può ragionevolmente assegnare, per ogni m, $\Gamma(x_m) = 0$ oppure $\Gamma(x_m) = -\ln \pi_m$, se gli stati hanno a priori probabilità π_m. Solitamente, dopo un momentaneo periodo iniziale, c'è un'alta possibilità che tutti i sopravvissuti si riuniscano nel cammino corretto.

Data la sua struttura fortemente parallela ed il solo uso di operazioni di somma, confronto e selezione, l'algoritmo di Viterbi è adatto per applicazioni ad alta velocità.

4.3 RETI NEURALI

Un'altra tecnologia che venne reintrodotta alla fine degli Anni Ottanta furono le ***Artificial Neural Network*** (ANN), grazie all'avvento di modelli di processing distribuito e parallelo (Parallel Distributed Processing, PDP), costituiti da un numero molto elevato di unità computazionali elementari, densamente interconnesse tra di loro. Inoltre, lo sviluppo di metodi di training basati su quello che comunemente viene definito error back propagation ridiedero interesse nei confronti di questi sistemi, il cui obiettivo originario era quello di realizzare elettronicamente dei meccanismi in grado di riprodurre il comportamento delle sinapsi umane.

Una particolare forma di PDP, il ***multiplayer perceptron***, attirò l'attenzione dei progettisti, non soltanto perché risultava essere la rete neurale che meglio simulava il comportamento umano, ma soprattutto grazie alla sua *capacità di approssimare praticamente ogni funzione sul segnale in ingresso, con una precisione arbitraria* (dipendente dalla configurazione dei suoi parametri), permettendo quindi di eseguire delle analisi e delle simulazioni senza limitazioni a livello di complessità delle operazione da implementare.

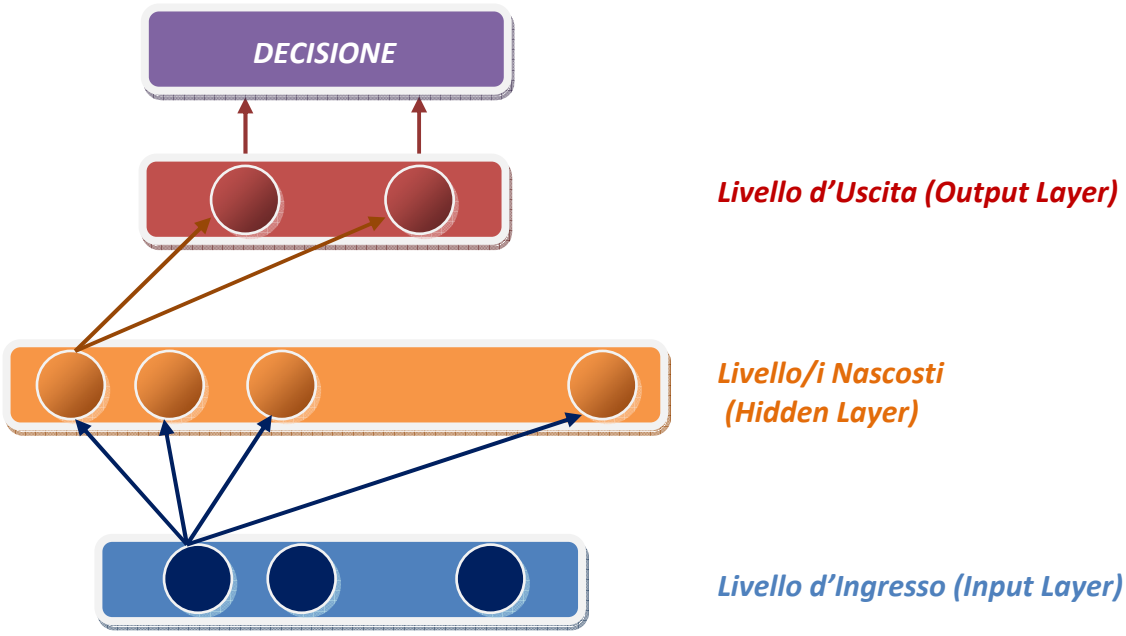


Figura 4.9: Esempio di multiplayer perceptron

Una rete neurale artificiale è un modello computazionale parallelo, costituito da numerose unità elaborative omogenee, raggruppate in differenti livelli (da qui il termine multi-layer), fortemente interconnesse mediante collegamenti di diversa intensità. Ogni neurone artificiale esegue la somma degli input pesati con il valore delle interconnessioni, per poi effettuare una trasformazione con una funzione spesso non lineare.

Uno degli aspetti più interessanti è che una rete neurale non viene progettata per compiere una particolare attività, ma essa dipende dal particolare algoritmo di apprendimento automatico adottato.

Questa qualità ne consente l'applicazione in svariati campi quali l'intelligenza artificiale, il trattamento dei segnali (voce, immagini, etc.), il controllo di robot o di processi industriali o la soluzione di problemi complessi raggiungendo, in molti casi, le prestazioni di altre tecniche più complesse e difficili da progettare. Nel corso degli anni si sono sperimentate diverse strutture, tra cui quella in assoluto più utilizzata è quella di tipo *feed-forward*, di solito con più strati.

Andando ad analizzare più in dettaglio la struttura di un singolo neurone (perceptron) possiamo concettualmente dividerlo negli ingressi, nei ***pesi*** (dendriti), nel ***corpo*** (cellula) e in una ***funzione di uscita*** (sinapsi). *I pesi saranno i responsabili, assieme al tipo di funzione di uscita, del comportamento del singolo neurone e della rete in generale*. Saranno loro che dovranno essere correttamente modificati in fase di addestramento, al fine di garantire un comportamento corretto al modello.

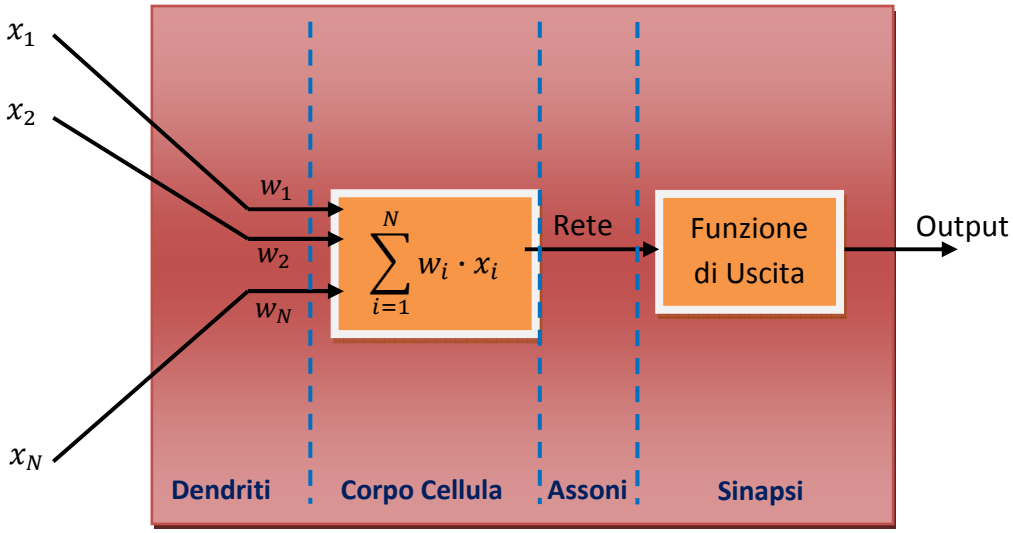


Figura 4.10: Schema del principio di funzionamento di una rete neurale

Definito per ogni singola unità i un vettore d'ingresso $x_i = (x_1, x_2 \dots, x_n)$ e un vettore di pesi $w_i = (w_{i1}, w_{i2} \dots, w_{in})$ il corpo del neurone compie una somma pesata degli ingressi:

$$in_i = \sum_{j=1}^{n} w_{ij} \cdot x_j \tag{4.9}$$

Tra i pesi ne esiste uno "particolare", che prende il nome di *bias*; esso non è legato a nessun'altra unità di rete ed è come se avesse ingresso sempre uguale a 1. Considerando anche questo peso, l'equazione precedente risulta nella forma:

$$in_i = \left(\sum_{j=1}^{n} w_{ij} \cdot x_j\right) + b_i \tag{4.10}$$

A questo valore viene quindi applicata una trasformazione al fine di generare l'uscita dell'unità elementare. Questa funzione può teoricamente essere di qualsiasi tipo, ma generalmente si utilizzano la sigmoide, il gradino, la tangente iperbolica, la funzione segno o la funzione lineare, anche se la trasformazione più adottata è la prima, così definita:

$$out_i = \frac{1}{1+e^{-in_i}} \tag{4.11}$$

A questo punto il funzionamento della rete neurale può essere spiegato senza difficoltà. La rete è semplicemente formata da un insieme di unità elementari. Quando arriva un nuovo segnale da analizzare esso viene caricato nei dati di input del primo livello di neuroni i quali, tramite le formule precedentemente descritte, generano un output. Questo risultato, assieme all'uscita dei neuroni dello stesso livello, costituirà un nuovo input per le unità al livello superiore. Il processo sarà così iterato fino all'ultimo layer in cui si otterrà la probabilità per ogni singola uscita.

Una volta capito il funzionamento, *l'**aspetto più importante e difficile** da affrontare consiste nell'**addestramento della rete**: nel cercare cioè i pesi e i bias che le permettano di essere il più possibile aderente al fenomeno che deve modellare*.

Senza questa fase, la rete non potrebbe essere utilizzata per il riconoscimento. A tale scopo esistono degli appositi algoritmi, di cui il più importante (ed utilizzato in questo progetto) è l'algoritmo di *back-propagation*. Esso venne proposto nel 1986 da Rumelhart, il cui scopo è quello di addestrare una rete neurale di tipo multi-layer perceptron basandosi sul criterio della discesa del gradiente per minimizzare l'errore quadratico medio tra gli output desiderati e quelli ottenuti, ovvero:

$$E = \frac{1}{2} \cdot \sum_{i=1}^{N} (d_i - o_i)^2 \tag{4.12}$$

dove d_i sono gli output desiderati, mentre o_i quelli restituiti dalla rete. L'aggiornamento dei pesi, uniche variabili modificabili nella ANN, è realizzato con la seguente formula:

$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}(t)} + \alpha \cdot \Delta w_{ij}(t-1) \tag{4.13}$$

nella quale:

$w_{ij}(t)$ = peso tra il nodo i e j all'istante t

η = *learning rate*, o *fattore di apprendimento* della rete; si tratta di un parametro chiave in quanto determina il giusto compromesso tra la capacità di apprendere nuovi pattern e non dimenticare quelli già memorizzati

α = momento, necessario per evitare che i pesi oscillino, durante il training, attorno ad un valore medio senza alcun miglioramento

In modo analogo a quanto visto per le tecniche di pattern recognition, anche per le ANN esistono due possibili tipi di training: ***supervised learning*** (addestramento supervisionato) e ***unsupervised learning*** (addestramento non supervisionato).

Nel primo caso, si danno in input alla rete dei patterns con associato la classe di appartenenza (target), nota a priori. Il risultato dell'elaborazione di questi dati viene analizzato mediante la back-propagation, un blocco di feedback che consente di apportare una correzione dei pesi e dei bias al fine di rendere minimo l'errore complessivo.

Un addestramento di tipo unsupervised viene invece utilizzato tutte le volte in cui, per qualche motivo, non si ha a disposizione il target. Questo metodo ha risultati peggiori rispetto al precedente ma ha due particolari pregi: può essere utilizzato in modo real time, cioè durante il normale funzionamento della rete in riconoscimento, e permette di addestrare la rete senza nessun intervento umano. Nel metodo supervised, infatti, c'è bisogno di un intervento esterno per dare al target il giusto valore e per consentire l'apprendimento del pattern.

Il funzionamento del metodo non supervisionato è abbastanza intuitivo, anche se all'inizio presuppone la presenza di una rete già semi-addestrata, che possa cioè generare un riconoscimento accettabile. Questa è una grossa limitazione, perché ne vede impossibile l'utilizzo nelle prime fase di addestramento, in cui si necessita di un sistema supervisionato.

Come è abbastanza intuitivo capire, l'identificazione di un determinato oggetto da parte di un'applicazione unsupervised non sarà sempre corretta e, in quei casi, si opererà un addestramento sbagliato alla rete. Ciò è compensato dal fatto che la rete tenderà naturalmente ad evolvere verso uno stadio di ottimo, anche se in modo meno preciso e più lento rispetto all'addestramento supervisionato.

4.4 MODELLI IBRIDI HMM-ANN

L'applicazione delle reti neurali nel campo del riconoscimento vocale è stata notevolmente ostacolata dalla natura del segnale vocale: esso, infatti, ha un comportamento di tipo sequenziale e temporale, mentre le ANN sono più adatte al riconoscimento di pattern statici. Solo nel 1988 si sono cominciati a compiere i primi studi su ***modelli ibridi markoviani-neurali*** (HMM-NN).

Questi ultimi ereditano dall'Hidden Markov Model la capacità di gestire fenomeni temporali attraverso un modellamento ad automa a stati finiti delle parole che consente di utilizzare l'algoritmo di Viterbi *per trovare la sequenza di stati ottima*. Dall'altro lato i vantaggi della tecnologia neurale sono molteplici: *le reti neurali sono, per loro definizione discriminative, di modo*

che durante il training le varie classi da riconoscere siano messe in competizione l'una contro l'altra, creando una più precisa separazione tra di loro e dando risultati migliori rispetto all'utilizzo di misture di gaussiane.

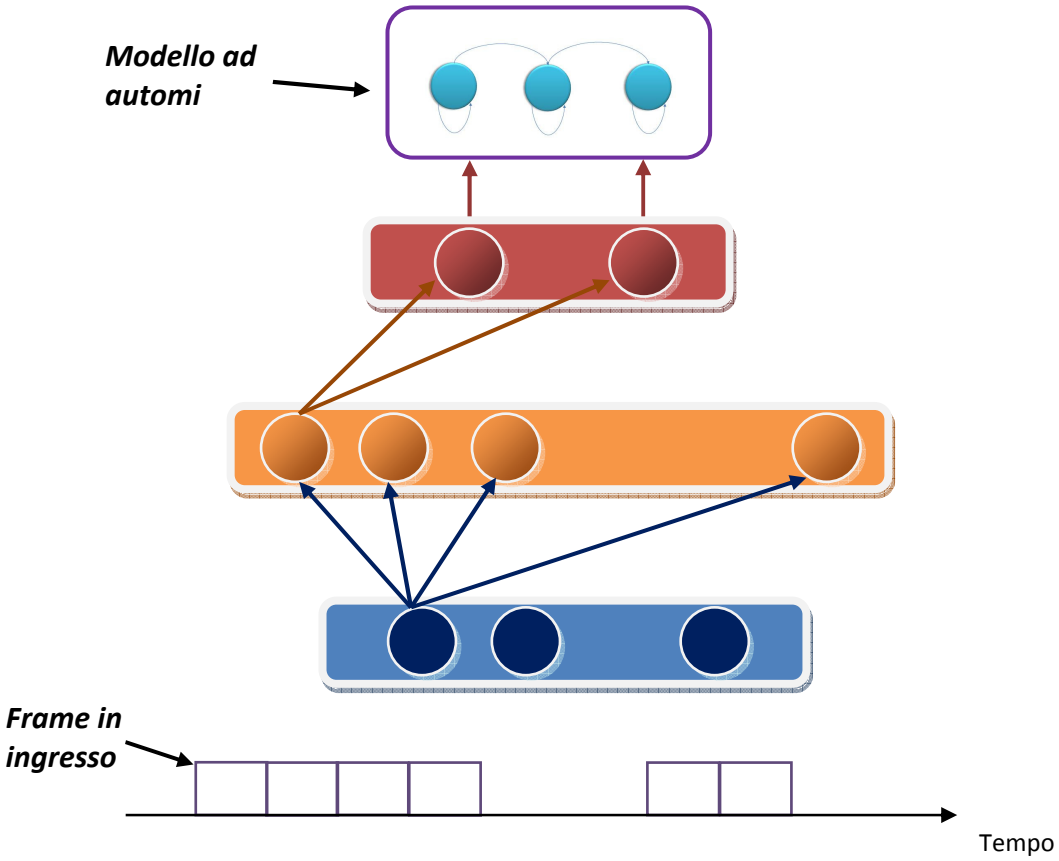


Figura 4.11: Schema a blocchi elementare di un modello ibrido HMM-ANN

L'input alla rete è una finestra di frames che ad ogni iterazione si sposta di un frame verso destra: una volta caricati i valori nel livello di ingresso della rete neurale (input layer) vi è una propagazione dei dati fino ai nodi di uscita (output layer). A questo punto si ottiene una probabilità che può essere utilizzata come probabilità di emissione nel modello ad automi, simile a quello descritto per l'Hidden Markov Model.

Come per le reti neurali generiche, anche per quelle finalizzate al riconoscimento vocale, la *fase di training ha un'importanza essenziale* per un adeguato funzionamento. Sebbene ci siano molte similitudini con le ANN classiche, l'addestramento di un modello ibrido markoviano-neurale richiede qualche attenzione in più, dovuto proprio alla natura temporale del segnale che deve

riconoscere: gli ingressi dipendenti dal tempo non creano alcun problema, lo stesso vale per i target; l'unico accorgimento da prendere è che siano sincronizzati con l'ingresso. Dando, ad esempio, in input il frame relativo al fonema “a” si dovrà confrontarlo con un target “a” e, nell’istante in cui avverrà il passaggio ad un altro fonema del segnale in ingresso, anche il modello in quel momento dovrà cambiare.

Il processo che porta alla determinazione del momento in cui avviene il passaggio tra un target e il successivo si chiama *segmentazione*. Il metodo più usato, visto gli ottimi risultati che si riescono ad ottenere e i tempi lunghissimi di un approccio manuale, è la ***segmentazione automatica***. Sapendo quali sono le utterance pronunciate, quindi la sequenza di simboli che le compongono, si utilizza la rete neurale e l'algoritmo di Viterbi per segmentare.

Quando bisogna addestrare una rete completamente nuova, sorgono due problemi:

- trovare una segmentazione iniziale, poi raffinarla
- addestrare nel modo migliore la rete a discriminare i fonemi

5. APPLICATION BOARD

La linea di componenti integrati (IC, Integrated Circuit) e software sviluppata da Sensory™ è una gamma di prodotti orientati alla sintesi e al riconoscimento vocale, che hanno come obiettivo principale quello di realizzare *tecnologie cost-sensitive*, ovvero di poter garantire un prezzo modesto, tale da poter impiegare tutti questi prodotti in applicazioni di uso comune, come elettrodomestici, home automation (domotica), giocattoli, comunicazioni, e molto altro ancora. L'intera gamma è basata sul ***microcontrollore*** della ***famiglia RSC-4x***: un componente a basso costo, ottimizzato per lavorare con segnali vocali, completamente integrato e con circuiti ausiliari per l'implementazione di ben precise operazioni, utili nell'ambito del processing dei segnali, quali convertitori analogico-digitali (ADC), preamplificatori, convertitori digitali-analogici (DAC), possibilità di interagire con memorie RAM e ROM esterne.

Per realizzare il progetto dell'elaborato è stata utilizzata una particolare application-board (sempre basata sulla tecnologia Sensory™ appena descritta): il ***VR Stamp Toolkit***™. Si tratta di un modulo in grado di realizzare il riconoscimento vocale ("VR" sta per Voice Recognition) in tutte le sue forme, il cui nucleo centrale è il *microcontrollore RSC-4128*, il quale è stato volutamente realizzato sottoforma di componente integrato DIP (Dual In-line Package, un particolare tipo di package per circuiti integrati che prevede la disposizione dei pin su due file parallele) a 40 pin, quindi un valore standard nell'industria elettronica.

Oltre all'unità hardware, il toolkit prevede tutto ciò che è necessario per sviluppare un'applicazione perfettamente funzionante: *VR Stamp™, Module Programming Board (ossia una scheda elettronica sulla quale può essere montato il micro in modo che si possa interfacciare direttamente con il PC, per la programmazione, o con altri dispositivi di input/output), un ambiente di sviluppo integrato (IDE, Integrated Development Environment) per poter scrivere e compilare il codice in C++, il Project SE della Phyton™, unitamente ad un compilatore appositamente realizzato per le librerie e le funzioni della Sensory™, il MCC-SE-LTV; infine, sono disponibili due applicativi software interamente dedicati alla parte vocale, ovvero QuickSynthesis™4 e Quick T2SI-Lite™*.

L'intero Software Development Kit (SDK) è basato sulla ***tecnologia FluentSoft***™, un insieme di librerie che permettono di implementare un elevato numero di funzioni per gestire e realizzare applicazioni di riconoscimento vocale: in grado di garantire una buona robustezza nei confronti del rumore, grazie alla loro dinamicità di configurazione, consentono di realizzare le modalità tipiche di speech recognition: speaker dependent, independent e verification, continuos speaking e word spotting (questi termini verranno chiariti più avanti nel testo).

La tecnologia FluentSoft™ è stata realizzata in modo da essere compatibile anche su processori non appartenenti a famiglie della Sensory™, tra i quali si ricordano Intel XScale, TI OMAP e piattaforme basate su ARM9 (quest'ultima è una delle più diffuse). Inoltre, supportano vari Sistemi Operativi, come Microsoft Windows, Linux e Symbian.

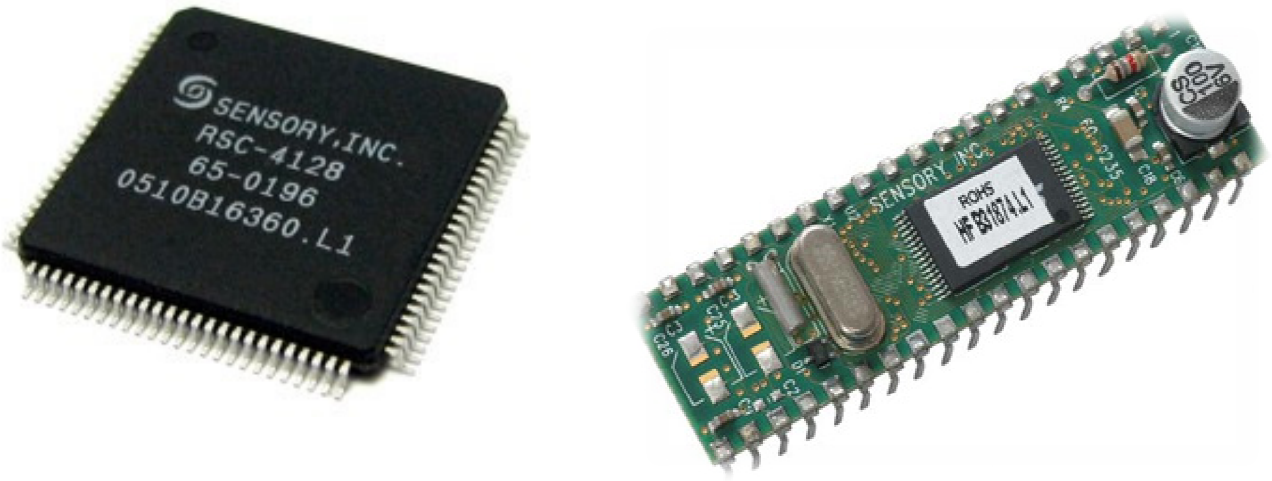


Figura 5.1: A destra il microcontrollore RSC-4128; a sinistra, lo stesso montato sul DIP a 40 pin del VR Stamp™

Nel seguito del capitolo verrà descritta prima la parte hardware e la logica di basso livello, per poi passare alla parte di programmazione e quindi di analisi dei software necessari per l'implementazione dell'applicazione.

5.1 HARDWARE: VR STAMP DEMO BOARD

RSC-4128 è uno dei processori della linea di prodotti Interactive Speech della Sensory™, *disegnato con l'obiettivo di realizzare un componente che racchiudesse la necessità di elevata integrazione e versatilità di utilizzo con la possibilità di essere low-cost e power-sensitive* (cioè basso consumo energetico). Proprio grazie all'elevata compattezza dell'integrato, oltre al microcontrollore sono presenti altre unità on-chip, per l'elaborazione di segnali analogici e digitali (tecnologia mixed-signal), ed il processing del segnale audio.

RSC-4128 dispone di 8 bit per il bus dati e 20 per gli indirizzi, che gli permettono di gestire e di interfacciarsi con eventuali dispositivi di storage esterni, siano essi ROM, RAM, EPROM o Flash.

Inoltre, per rendere l'integrato ancor più dinamico, sono presenti tre porte bidirezionali, ciascuna delle quali dotata di 8 pin per operazioni generiche (General Purpose pin), configurabili a livello software come input o come output: in totale sono quindi disponibili 24 pin per scopi generici.
Per la gestione del clock e la sincronizzazione, sono previsti due oscillatori: uno ad alta frequenza, 14.32 MHz, ed uno a bassa frequenza, 32.768 KHz. Il timing del processore può essere estratto da entrambi i quarzi, grazie alla possibilità di settare opportuni registri; tuttavia, ***solamente con un clock di sistema originato dai 14.32 MHz è possibile implementare le funzioni di riconoscimento del parlato***.
Per consentire operazioni di conteggio, sono previsti anche tre timer programmabili (nel senso che si può determinare l'intervallo temporale dopo il quale viene generato un impulso di conteggio ed il numero massimo di impulsi, al cui raggiungimento viene generato un interrupt): Timer1, Timer2 e Timer3. In aggiunta, è disponibile un timer per il Watchdog, importante per consentire al programma di uscire da eventuali stati sconosciuti o condizioni indesiderate durante l'esecuzione dello stesso. Infine, per permettere al microprocessore di gestire più operazioni in parallelo, è previsto un Multi-Tasking Timer.
Accanto a questi elementi, tutti legati alle funzionalità dell'unità logica centrale, è presente anche un preamplificatore in ingresso per modificare e gestire in modo opportuno il segnale audio (voce o generico suono che sia), a valle del quale vi è un convertitore analogico-digitale a 16 bit.
Nel front-end di uscita, invece, sono presenti un DAC a 10 bit e un modulatore di larghezza d'impulso (PWM): entrambi i dispositivi generano un segnale analogico che può direttamente pilotare l'output, che tipicamente sarà un altoparlante.
La realizzazione on-chip di una serie di blocchi di filtri digitali consente l'estrazione delle feature del segnale vocale, che verranno poi rese disponibili alle librerie contenenti le funzioni (a livello software) che implementano le varie fasi del riconoscimento.
Infine, sempre *per poter raggiungere gli obiettivi di risparmio energetico, sono previste delle modalità di lavoro a basso consumo*, ovvero è possibile avere un ***power-down*** dell'application board, mantenendo attive solamente quelle unità strettamente necessarie al funzionamento della stessa.
Un ultimo accenno è relativo agli interrupt, che l'RSC-4128 è in grado di gestire se opportunamente configurato. Nel proseguio del capitolo verranno descritti in modo più approfondito i principali blocchi cui si è accennato in questo paragrafo e che vengono presenti nella figura

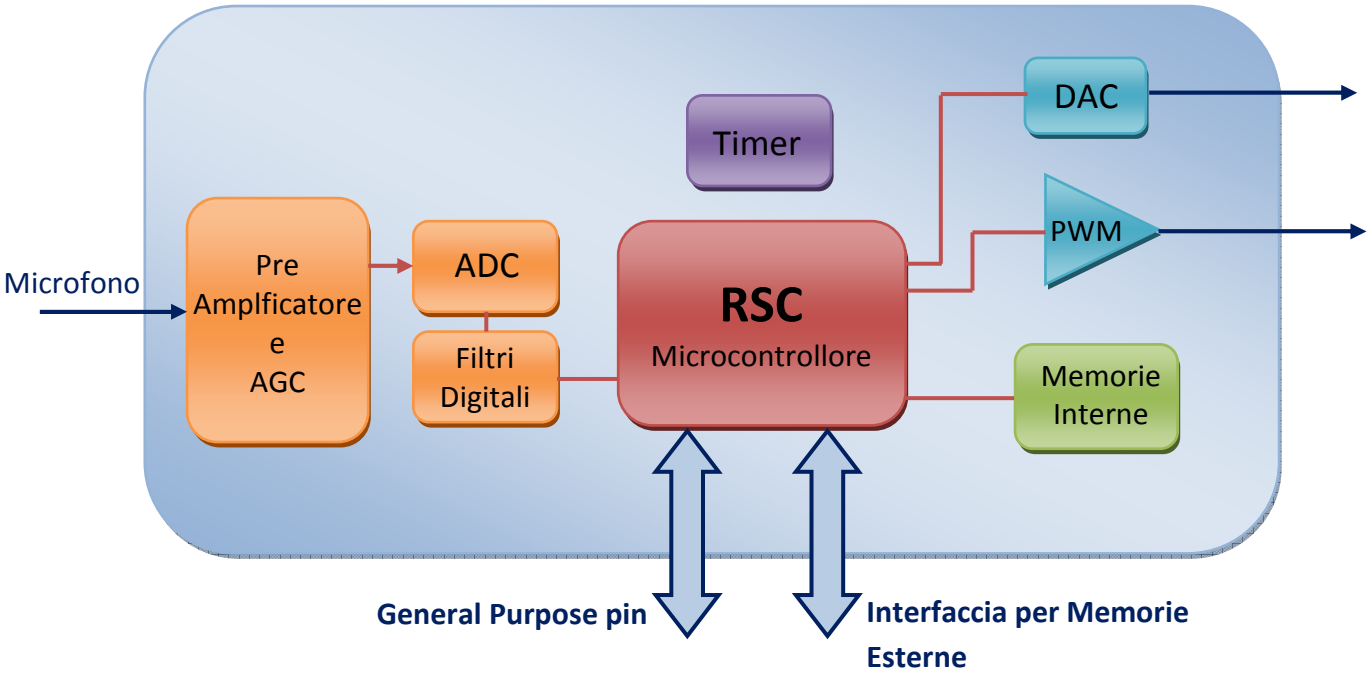


Figura 5.2: Schema a blocchi dell'application board

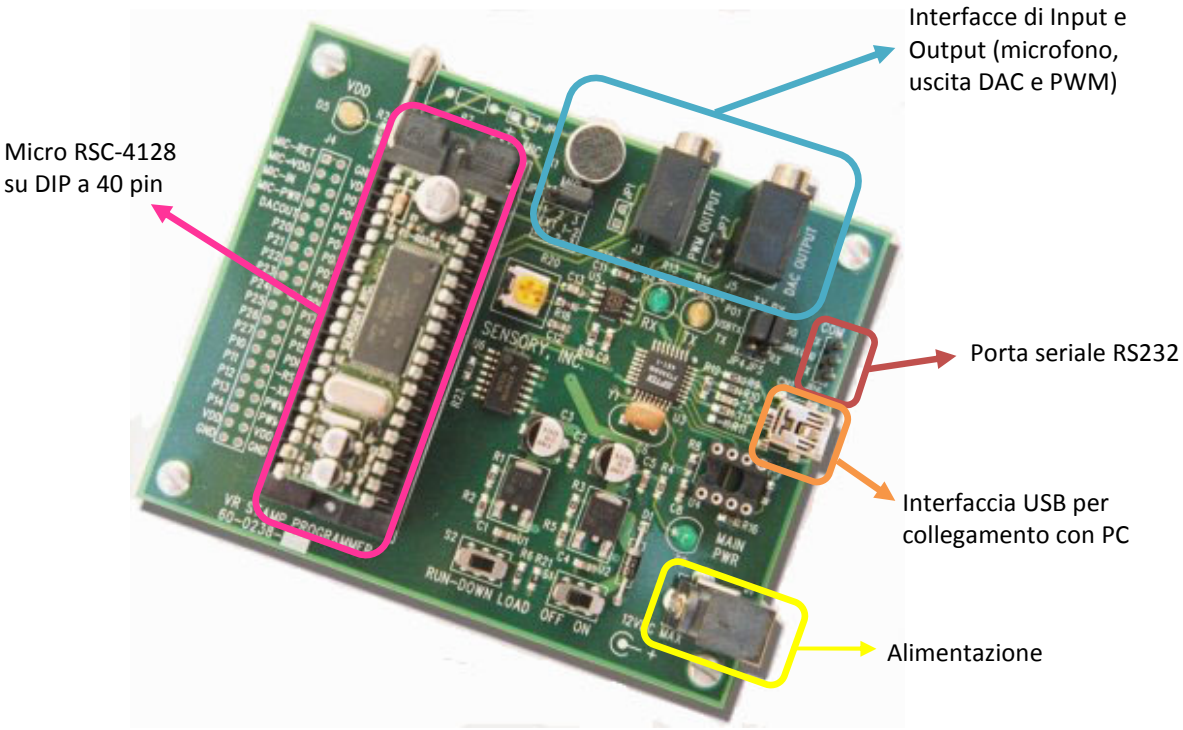


Figura 5.3: Modulo VR Stamp™ e parti principali

5.1.1 GENERAL PURPOSE I/O

Come già accennato in precedenza, l'RSC-4128 dispone di 3 porte, ciascuna delle quali dotata di 8 pin per generici I/O: P0.0-P0.7 per la porta 0, P1.0-P1.7 per la 1 e P2.0-P2.7 per la porta 2.

Ognuno di questi pin è ***configurabile a livello software*** *come input o come output*, grazie ad opportune direttive nel file di configurazione del micro. Gli input possono essere di tre tipi e si differenziano a seconda della resistenza di pull-up: con questo termine si fa riferimento ad una tecnica impiegata nei circuiti logici elettronici (in particolare microcontrollori ed integrati) per garantire che gli ingressi siano ad un prestabilito livello logico anche quando sono scollegati (fluttuanti) oppure ad alta impedenza. Trattandosi di pull-up, e non del suo opposto pull-down, il pin si porterà ad un livello alto, dato che internamente l'input è collegato alla tensione di riferimento V_{cc}.

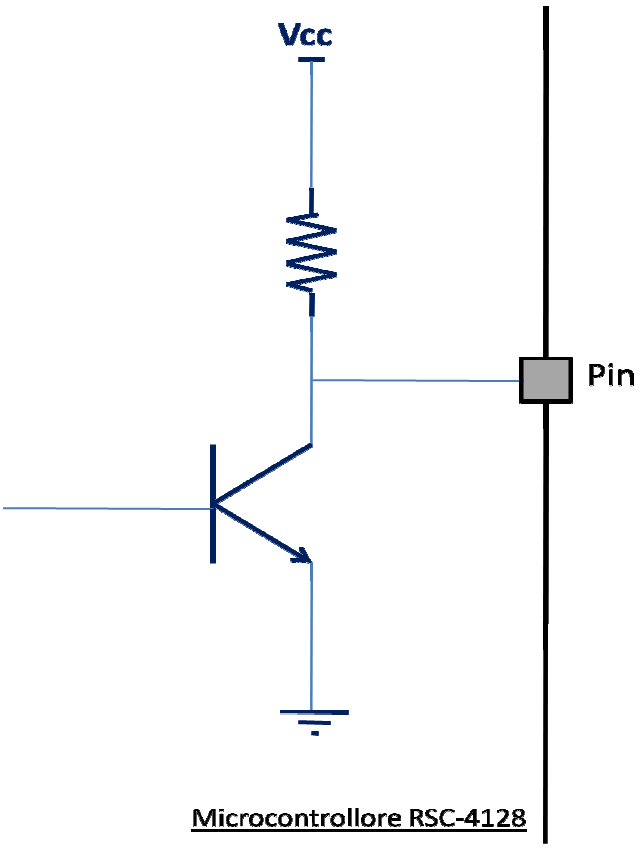


Figura 5.4: Circuito interno di pull-up di un pin GPIO

Il valore della resistenza può essere più o meno alto, a seconda della corrente che si vuole fornire al carico collegato a valle del pin. Nel caso particolare della RSC-4128 si possono avere ingressi con *pull-up debole*, la cui resistenza vale 200 kΩ (e, di conseguenza, permette la circolazione di una

corrente bassa), *pull-up forte* (10 kΩ), con una corrente che può fluire verso il carico molto più alta del caso precedente, e input *senza* alcun *pull-up*, in cui il pin si trova in una condizione cosiddetta "fluttuante" (né alto, né basso) ed è consigliabile utilizzarla qualora esternamente il componente sia connesso a massa oppure a V_{cc}, cioè quando lo stato del pin venga stabilito esternamente.
La quarta configurazione possibile è impostare il pin dell'integrato come output: in questa modalità non vi sono problemi di resistenza di pull-up, semplicemente i GPIO sono stati realizzati in modo tale che abbiano una corrente sufficiente per pilotare un diodo LED.
Due registri di controllo, A e B, sono utilizzati per impostare la natura degli input e degli output di ciascuna porta. I due registri, insieme, determinano la funzione dei pin come segue

Registro B	Registro A	Configurazione pin
0	0	Input, weak pull-up
0	1	Input, strong pull-up
1	0	Input, no pull-up
1	1	Output

Tabella 5.1: Configurazione pin micro RSC-4128

La condizione di default, che si verifica la prima volta che si accende il dispositivo oppure dopo ogni reset, prevede che i pin P0.0-P0.7, P1.0-P1.7, P2.5-P2.7 siano impostati come input aventi un debole pull-up, mentre i restanti (P2.0-P2.4) sono sempre input, ma senza pull-up.
Inoltre, P0.0 e P0.2 possono essere configurati come ingressi sui quali attendere degli interrupt esterni, mentre P0.1 come pin per ricevere un evento del contatore.

5.1.2 INDIRIZZAMENTO DELLA MEMORIA

L'RSC-4128 è in grado di indirizzare fino a 2 Mbytes, comprensivi dei 128 Kbit della ROM interna e/o di altri eventuali device di storage esterni, senza che sia necessaria l'aggiunta di ulteriori circuiti di decodifica. Indipendentemente dal tipo di memorie usate e dal fatto che siano interne (on-chip) o esterne (off-chip), lo spazio a disposizione viene diviso (sia in senso "logico", che fisico) in due differenti sottospazi, ciascuno avente una dimensione massima di 1 Mbyte: *Constant/Code Space* e *Data Space*.
Il primo può essere salvato sia sulla memoria interna che su dispositivi esterni, ed è di sola lettura (read only). Questa porzione viene tipicamente usata per il salvataggio del codice dell'applicazione,

oppure del vocabolario nel caso in cui si voglia implementare un riconoscimento di tipo speaker independent. Al contrario, il Data Space è sempre salvato su memorie esterne al chip (tenendo in considerazione che può avere un dimensione massima di 1 Mbyte, indipendentemente da quella del dispositivo) e viene usato, nella maggior parte dei casi, per memorizzare i templates, nel caso di riconoscimento dipendente dal parlatore, oppure le registrazioni audio qualora si usi anche una tecnica Record and Playback. In sostanza, si nota come *il Data Space sia caratterizzato da dati volatili*, che vengono memorizzati per un periodo di tempo limitato o che possono essere modificati direttamente dall'utente.

In alcuni casi è possibile che, a causa di mancanza di spazio sulla memoria on-chip, sia necessario salvare porzioni del programma su dispositivi esterni, nel qual caso questi dati non saranno più volatili, ma esclusivamente di sola lettura. La comunicazione con device esterni può avvenire sia in maniera seriale, che parallela.

5.1.3 OSCILLATORI

L'RSC-4128 ha integrati ***due oscillatori indipendenti***, realizzati con tecnologie differenti, come diverse sono le pulsazioni che generano: *OSC1 è ad alta frequenza, sorgente dei 3.58 MHz che vengono sfruttati per originare il clock di sistema, e OSC2, operante a frequenze più basse, 32.768 KHz*. Il primo è costituito da un cristallo (elemento attivo) e da un circuito risonatore LC (con condensatore ceramico), mentre il secondo prevede sempre un quarzo ma con un circuito interno RC.

È utile soffermarsi un istante su questo argomento per descrivere brevemente le caratteristiche degli oscillatori utilizzati nel campo elettronico. Il principio di funzionamento può essere schematizzato come illustrato di seguito:

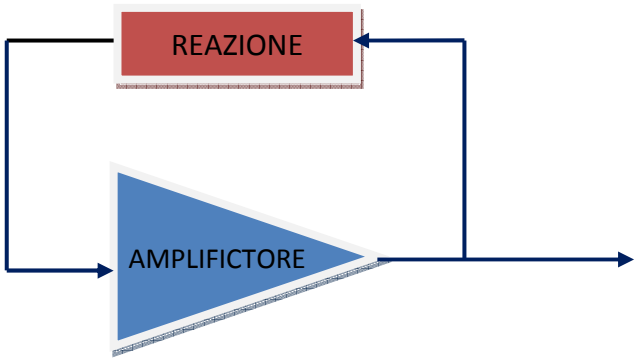


Figura 5.5: Schema concettuale del funzionamento di un oscillatore

Sono quindi previste due unità: un *amplificatore* e un *blocco di reazione* (o retroazione, in inglese feedback, l'elemento che accomuna la maggior parte degli oscillatori impiegati per scopi elettrici). Lo schema può apparire strano o addirittura errato, dato che, come si nota, non è presente alcun segnale esterno applicato in ingresso al sistema, ma vi è solamente un collegamento diretto tra uscita ed ingresso, per mezzo di un ramo di retroazione. In realtà, anche se non rappresentata nella figura precedente, è presente un'alimentazione, necessaria al funzionamento dell'amplificatore che è un componente attivo (per semplicità ed immediatezza di "lettura" non è stata disegnata nell'immagine).

Supponiamo che sull'amplificatore sia presente un segnale d'uscita di qualunque tipo, per esempio il transitorio derivante dall'accensione dell'apparecchiatura. Una porzione di questo segnale viene riportato immediatamente all'ingresso tramite il ramo di feedback, e quindi viene nuovamente amplificato e in uscita si trova ad interferire con il segnale già presente. Come è ben noto, nel campo dell'acustica, come in quello dell'ottica e in altri ancora, l'interferenza non necessariamente è un fenomeno negativo, ovvero non sempre può essere distruttiva, ma vi sono dei casi in cui si ha interferenza costruttiva (o positiva): più precisamente, se due segnali sono in fase (stessa frequenza e stessa posizione temporale) si sommano, in caso contrario tendono a degradarsi, fino ad annullarsi nel caso di fase opposta. ***La reazione positiva è quella che sta alla base degli oscillatori***: di tutte le componenti (pulsazioni) del segnale riportate in ingresso, solamente alcune verranno esaltate dal circuito di retroazione, tutte le altre verranno attenuate. Quali componenti vengono privilegiate e quali ridotte, dipende dalla fase con cui il segnale si presenta in ingresso all'amplificatore. L'operazione viene ripetuta un numero elevato di volte, fino a che si ottiene un segnale di frequenza unica e fissa (o, meglio, con una variazione, cioè una tolleranza, accettabile), per la quale si ha la massima esaltazione dovuta alla particolare rete di reazione impiegata e opportunamente calibrata per questa pulsazione.

In sostanza, quindi, *il blocco di reazione ha il compito di sfasare ed attenuare in modo opportuno il segnale d'uscita, in modo da ottenere la giusta ampiezza e fase per la frequenza desiderata*.

Tipicamente, sono due le modalità principali con cui viene realizzato il blocco di retroazione: RC (resistenza e capacità) o LC (induttanza e capacità); il primo tipo è utilizzato per costruire oscillatori di bassa frequenza, mentre il secondo usa un circuito accordato ad induttanza e capacità e trova largo impiego nella realizzazione di segnali a radiofrequenza.

In entrambi i casi, il parametro fondamentale per fare in modo che l'oscillatore possa lavorare è il cosiddetto ***guadagno d'anello***, ovvero il prodotto tra il guadagno dell'amplificatore e l'attenuazione del blocco di retroazione: per avere oscillazioni persistenti, questo valore deve essere maggiore di 1,

ovvero l'amplificatore deve essere in grado di fornire al segnale l'energia persa a causa delle dissipazioni nei vari passaggi del circuito di feedback, ma non deve essere nemmeno troppo elevato da comportare fenomeni di saturazione.

Gli oscillatori con parte passiva costituita da resistenze e capacità (RC, definito anche circuito a sfasamento) sono generalmente utilizzati per generare basse frequenze, con un range che varia da 1 Hz a 1 MHz: la pulsazione che viene privilegiata è determinata dai valori dei componenti passivi, i quali introducono uno sfasamento sul segnale di 180°. In questo modo si avrebbero segnali in opposizione di fase in uscita all'amplificatore, che tenderebbero ad annullarsi: nella pratica, è uso comune utilizzare un amplificatore invertente, ossia un componente attivo che di suo sfasa il segnale in ingresso di 180°; di conseguenza non si introduce alcun sfasamento sul segnale (per la frequenza specifica che si vuole generare).

Il limite principale di questi sistemi è la loro intrinseca instabilità e il fatto che i valori di resistenze e condensatori è fisso, ovvero si può generare un'unica pulsazione.

Molto più diffusi (e importanti) sono gli oscillatori a radiofrequenza (RF), che nella maggior parte dei casi sono basati su un circuito accordato (detto anche risonante), formato da un condensatore e da una bobina (LC): esso ha la proprietà di lasciar passare una corrente la cui frequenza coincide con quella di risonanza del circuito stesso, mentre attenua fortemente tutte le correnti aventi pulsazioni differenti. La rete di reazione è costituita da un circuito accordato che seleziona la frequenza e da un elemento di accoppiamento, che può essere capacito o induttivo a seconda della configurazione.

Nella seguente tabella è presente un breve riepilogo delle caratteristiche dei quarzi installati sulla demo board

Oscillatore	Frequenza	PLL	Sorgenti
1	3.58 MHz	4X	Cristallo, risonatore ceramico LC
2	32.768 KHz	/	Cristallo esterno o risonatore RC interno

Tabella 5.2: Oscillatori

Come per i pin di input/output, anche per gli oscillatori è possibile impostarne l'abilitazione a livello software. OSC1 deve lavorare a 3.58 MHz qualora si stiano usando le librerie della FluentChip™; se, invece, l'application board viene impiegata come piattaforma per scopi generici è

possibile (e consigliabile) ridurre la frequenza dell'oscillatore. Ovviamente, se quest'ultimo è disabilitato, anche il PLL non è attivo.

OSC2 si differenzia dal precedente perché è possibile scegliere se adottare la sorgente on-chip oppure optare per una esterna: questa scelta è sempre possibile configurando opportunamente i registri di sistema. Anche se può sembrare scontato, è utile rimarcare come in caso di sorgente esterna la frequenza da essa generata deve essere sempre 32.768 KHz.

La scelta di utilizzare un quarzo esterno o il circuito risonatore interno è dettata dalla precisione del timing che si vuole avere: nel caso in cui questa debba essere elevata oppure il clock abbia un ruolo fondamentale nell'esecuzione dell'applicazione, è preferibile adottare un cristallo esterno. Con questa configurazione, infatti, è possibile ridurre la percentuale di errore fino a 20 ppm. Qualora si opti per la sorgente interna, si deve tenere in considerazione che *il circuito RC on-chip ha delle variazioni non trascurabili, dipendenti dalla temperatura, dal particolare processo in esecuzione e dalle oscillazioni della tensione di alimentazione*; tutti questi fattori causano una tolleranza che può raggiungere anche il $\pm 30\%$ rispetto al valore nominale. È facile intuire come questa scelta sia conveniente quando il timing non è un parametro critico del sistema, permettendo di mantenere un costo di realizzazione dell'applicazione basso.

I più precisi e diffusi, anche se spesso più costosi, sono gli oscillatori al quarzo: il loro funzionamento è basato sul fatto che certi cristalli producono cariche di segno opposto quando si esercita uno sforzo tra due superfici; viceversa, si deformano quando si applica ad essi una tensione mediante due elettrodi. Questo fenomeno è chiamato ***effetto piezoelettrico***. In altri termini, ciò consiste nel fatto che tali cristalli subiscono deformazioni meccaniche se sottoposti ad un campo elettrico e, viceversa, generano un campo elettrico se sottoposti a deformazioni meccaniche.

Perché di un cristallo piezoelettrico si possa sfruttarne le caratteristiche, questo deve essere tagliato in lamine parallele a determinati assi cristallografici e inserito fra due elettrodi: il tutto costituisce un sistema elettromeccanico che vibra se opportunamente eccitato. La frequenza di risonanza f è legata allo spessore d dalla relazione

$$f = \frac{1666}{d} \tag{5.1}$$

avendo espresso f in MHz e d in mm.

Per ottenere un oscillatore al quarzo realmente funzionante, occorre inserire il cristallo in un circuito reazionato positivamente, in modo da farlo risonare nella sua zona induttiva.

5.1.4 CLOCKS

Ora che sono stati descritti i due oscillatori presenti sulla RSC-4128, è utile illustrare brevemente i segnali di clock presenti. Innanzitutto, è buona cosa definire il range di frequenze entro il quale il microcontrollore è in grado di lavorare correttamente: esso si estende da 1 KHz fino ai 14.32 MHz.
Il clock del processore può essere scelto tra due opzioni possibili: CLK1, generato da un PLL che quadruplica la frequenza generata da OSC1 fino a raggiungere i 14.32 MHz, oppure CLK2, determinato direttamente da OSC2. La selezione viene sempre fatta settando i bit dei registri di sistema. Considerando le frequenze di lavoro, CLK2 è consigliabile utilizzarlo quando il processore abbia un tempo di attività limitato durante l'esecuzione del programma, riducendo ulteriormente i consumi. Al contrario, CLK1 è conveniente, data la pulsazione elevata, quando l'attività del microcontrollore occupa gran parte del tempo in cui l'applicazione è attiva.
La condizione di default prevede che a generare il clock del processore sia CLK1.
Una piccola considerazione sul perché dell'utilizzo di un PLL a valle dell'oscillatore ad alta frequenza è d'obbligo: un quarzo che generasse direttamente i 14.32 MHz avrebbe avuto un costo abbastanza elevato, ma con il circuito ad aggancio di fase è possibile sfruttare un cristallo a pulsazione inferiore, quindi meno costoso, perseguendo uno degli obiettivi principali della linea RSC-4x: essere low-cost.
Volendo, è possibile utilizzare un timing di sistema esterno, avendo premura di verificare che esso abbia valori accettabili per il corretto funzionamento del micro.
Accanto al clock di processore appena descritto, ve ne sono presenti altri nel sistema, sempre originati da uno dei due oscillatori:

- **OSC1**: *CLK1, clock dei filtri digitali, master clock del front-end analogico, Timer1, Timer3, Multi-Task Timer*;
- **OSC2**: *CLK2, Timer2, Watchdog*;

Nel caso in cui, per qualsiasi motivo, venga a mancare il clock di sistema, ciò non causa perdita di informazioni o errori, dato che la RSC-4128 adotta una ***logica fully static***, permettendo in questo modo che il processo possa essere arrestato e riattivato senza che si sia verificato un reset generale o che siano stati persi i contenuti nei registri interni.

5.1.5 TIMER

L'application board è dotata di più unità di conteggio e timing, alcune delle quali con valori impostabili direttamente dall'utente ed altre, invece, con parametri fissi.

Alla prima "categoria" appartengono ***Timer1*** e ***Timer3***, entrambi *aventi come sorgente un clock corrispondente a CLK1 diviso per un fattore pari a 16* (con una frequenza di $14.32/16 = 0.895\ MHz$) e caratterizzati da 3 elementi: un registro a 8 bit nel quale memorizzare il valore da cui partire con il conteggio, definito ***reload value register***, un ***up-counter*** a 8 bit e un blocco che consente di effettuare operazioni di pre-scalatura a 4 bit. Nella seguente tabella vengono indicati tutti i possibili divisori

Valore di Scala	Divisore	Valore di Scala	Divisore
0000	0	**1000**	256
0001	2	**1001**	512
0010	4	**1010**	1024
0011	8	**1011**	2048
0100	16	**1100**	4096
0101	32	**1101**	8192
0110	64	**1110**	16384
0111	128	**1111**	32768

Tabella 5.3: Divisori timer

Il reload register è sia leggibile che scrivibile dal processore (l'operazione di scrittura può essere ripetuta più volte nel codice del programma), mentre il registro contenente il valore attuale del conteggio è di sola lettura.

Si noti che, se l'applicazione tenta di scrivere nel contatore, il dato che si vuole immettere viene ignorato, ma l'operazione di scrittura comporta la ripartenza del conteggio dal valore iniziale.

Il più grande intervallo temporale che può essere conteggiato da T1 e T3 vale

$$T_{max} = 2^{23} \cdot \frac{1}{CLK1} = 2^{23} \cdot \frac{1}{16 \cdot OSC1} = 2^{23} \cdot \frac{1}{16 \cdot 14.32 \cdot 10^{-6}} = 9.3\ s \qquad (5.2)$$

In aggiunta alla sua funzione di timing, ***il Timer3 può essere configurato come contatore di eventi esterni***, valutando il fronte di salita o di discesa di un segnale applicato al pin P0.1 (da specifica la

transizione del fronte è internamente sincronizzata con il CLK1 e può avere, esternamente, un count rate massimo di 447 KHz).

Il ***Timer2*** (T2) ha una frequenza di conteggio più bassa dei precedenti, avendo come sorgente CLK2 diviso per un fattore costante pari a 128 (dunque, $32768/128 = 256\ Hz$). È caratterizzato dal fatto che, una volta terminato il conteggio, ***può dare luogo ad una richiesta di interrupt*** generica, utilizzabile, se configurato nel modo opportuno, ***per avviare un'operazione di Wakeup da una condizione di powerdown del sistema***.

Nella tabella che segue sono illustrati i due registri principali per ciascun timer, il *reload register* (*txr*) e il *registro contenente il valore attuale del contatore* (*txv*):

Registro	Stato	Valore
t1r	Lettura/Scrittura	Timer1 counted Reload
t1v	Lettura Scrittura	Valore corrente Timer1 Forzatura per riportare il contatore al valore del reload register
t2r	Lettura/Scrittura	Timer2 counted Reload
t2v	Lettura Scrittura	Valore corrente Timer2 Forzatura per riportare il contatore al valore del reload register
t3r	Lettura/Scrittura	Timer3 counted Reload
t3v	Lettura Scrittura	Valore corrente Timer3 Forzatura per riportare il contatore al valore del reload register

Tabella 5.4: Registri timer

A fianco di questi contatori programmabili, il sistema prevede anche dei timer fissi, le cui frequenze di lavoro non sono modificabili in nessun modo. Il primo di questi è il cosiddetto ***Multi-Tasking Timer*** (MT), il quale consente un conteggio con un intervallo temporale prefissato pari a 858.1 µs, ricavabile dividendo CLK1 per un fattore di scala fisso, 12288. Questo timing permette di avere una pulsazione regolare (a volte la si indica con il termine "heartbeat", ovvero battito cardiaco, per rendere meglio l'idea) sfruttata per eseguire più operazioni in parallelo nello stesso istante, dato che con le librerie della FluentChip™ è realizzabile il mlti-tasking. Anche per l'MT è possibile generare un interrupt che venga opportunamente interpretato dal sistema una volta terminato il conteggio.

Infinte, è presente un timer che accomuna molte applicazioni: il ***Watchdog Timer*** (WDT). A causa dell'elettricità statica, dei glitch di tensione o di altri particolari condizioni ambientali, nonché bug nella scrittura del firmware/sofware, il programma potrebbe lavorare in modo non corretto, rimanendo bloccato in particolari stati non previsti e sconosciuti all'applicazione. Il WDT garantisce una protezione nei confronti di queste problematiche.

Esso ha come sorgente OSC2, di conseguenza l'accuratezza del timing dipende da quale circuito risonatore si sia utilizzato per l'oscillatore, se quello interno (meno costoso ma anche meno preciso) o quello esterno (genericamente di qualità migliore). Sempre grazie ai registri di sistema, appositamente previsti per questa funzione, è possibile determinare quale debba essere la durata del timeout prima che venga generato un impulso di watchdog: il range va da 15.6 ms a 4 secondi.

È importante far notare come l'abilitazione di questo timer avvenga tramite un'apposita istruzione, e venga disabilitato ogni volta che avviene un reset del sistema, ma anche quando esso si porta in condizione di powerdown,. Una volta avvenuto il ripristino delle normali condizioni di funzionamento, il WDT resta disabilitato e si rende necessario riattivarlo ogni volta.

Inoltre, dato che la sorgente di questo timer è l'oscillatore OSC2, il sistema è realizzato in modo tale che, una volta abilitato il WDT, il bit di registro che determina l'attivazione di OSC2 non possa più essere modificato: ciò si rende necessario per evitare che, accidentalmente, il programma in esecuzione possa andare a modificare il bit, disabilitando il Timer2 e, di conseguenza, anche il Watchodg, rendendo l'applicazione priva della protezione nei confronti degli stati indesiderati.

Perché il sistema funzioni correttamente, è necessario che il programma richiami l'istruzione di abilitazione del Watchdog ad un rate più veloce del periodo di timeout dello stesso, altrimenti allo scadere del conteggio del WDT verrebbe generato un impulso, il TimeOut Reset, che resetta l'intera RSC-4128.

5.1.6 POWER E WAKEUP CONTROL

Per poter valutare le prestazioni e la capacità del sistema di lavorare a bassi consumi, è necessario prima di tutto definire quale sia il valore tipico di corrente di alimentazione nel caso di funzionamento in condizioni normali. Questo accade quando la demo board viene alimentata a 3V e il microcontrollore ha un clock rate di 14.32 MHz, con tutti gli I/O configurati alti (è possibile farlo con un pull-up software di ciascuno dei 24 GPIO): si ha una corrente di circa 12 mA. Ovviamente, frequenze di lavoro più basse consentono di ridurre il consumo energetico del dispositivo, ma la tecnologia FluentChip™ richiede tipicamente un clock di 14.32 MHz.

*Il **power-down** può avvenire in due modalità differenti, che si differenziano per le unità che rimangono abilitate anche in condizioni di basso consumo: **Sleep Mode** e **Idle Mode***. Nel primo caso, tutti i blocchi sono spenti e l'unica via per poter riattivare (o, traducendo letteralmente il termine inglese wake-up, risvegliare) il normale funzionamento del sistema è un evento di input/output: si tratta della modalità di risparmio energetico più efficiente, con correnti di alimentazione inferiori ad 1 µA. In condizioni di Idle, invece, OSC2 e Timer2 continuano a rimanere operativi e il Wakeup può essere determinato da un evento di I/O oppure da un interrupt generato dal Timer2 a seguito di un overflow (cioè dopo aver raggiunto il termine del conteggio), consumando in media 7 µA.

È previsto anche un *modo a bassa potenza, usando l'Idle con un Audio Wakeup*. In questa condizione è il circuito di Audio Wake up che determina il risveglio del chip dal suo stato di power-down. La differenza rispetto all'Idle mode classico è che il risveglio, oltre che da un evento I/O o dall'interrupt di Timer2, può essere generato anche da un evento sonoro. La corrente consumata dalla sola logica di Audio Wakeup è tipicamente 40 µA, mentre considerando anche il quarzo, il Timer2 e il microfono attivo, il consumo totale è di circa 150 µA.

Per minimizzare il consumo molti blocchi operazionali sul chip sono dotati di enable, in modo che possano essere abilitati o meno direttamente dal programmatore in base alle direttive memorizzate nell'applicazione.

5.1.7 WAKEUP DA POWERDOWN

Contrariamente a quanto si potrebbe pensare (e spesso viene frainteso), il powerdown del dispositivo non comporta alcuna perdita di contenuti e di informazioni: semplicemente, si pone in una condizione tale da dover alimentare unicamente quei circuiti strettamente necessari al corretto funzionamento e alla gestione logica dell'applicazione. Si capisce quindi come powerdown e reset siano concetti completamente differenti: il primo comporta solo uno stato di basso consumo del sistema, il secondo lo riporta alle condizioni di default, eliminando qualsiasi informazione aggiuntiva o differente da quelle di fabbrica. Insomma, ***una volta "risvegliato" il dispositivo è nelle stesse condizioni in cui era prima di "congelarsi".***

Per quanto riguarda gli eventi che danno luogo ad un *Wakeup*, quelli *generati da I/O e dal Timer2* si comportano allo stesso modo, cambia solamente la sorgente che li ha originati: in entrambi i casi, *viene rilevato il fronte d'onda del segnale con una certa polarità*.

Molto simile è il risveglio a seguito di un interrupt generato dal Timer2; la differenza è che non è necessario configurare su quale pin e con quale polarità ci si attende il fronte del segnale di

riattivazione delle normali condizioni di funzionamento, ma si deve solamente indicare se questo tipo Wakeup è abilitato o meno (sempre configurando i corrispondenti registri). È importante far notare come, trattandosi di un interrupt, ossia di un evento che può accadere in un istante qualsiasi durante l'esecuzione del programma, una volta terminata la routine ad esso corrispondente è necessario cancellare il flag corrispondente alla richiesta di interrupt da parte del Timer2.
L'Audio Wakeup si differenzia dalle due modalità precedenti in quanto richiede la presenza di un apposito circuito per determinare se un particolare tipo di evento sonoro, impostato originariamente dal programmatore, viene rilevato.

5.1.8 INTERRUPTS

L'application board consente ***8 possibili richieste di interrupt***, ognuna delle quali generata da sorgenti differenti. *Sono tutte asincrone, rilevate sul fronte di salita dell'impulso, ad eccezione delle richieste provenienti dall'esterno (della RSC-4128), le quali, come abbiamo visto, hanno un edge programmabile*. Ciascuno di questi possibili interrupt è attivabile configurando opportunamente il corrispondente bit nel registro di sistema ed è prevista anche un'abilitazione generale: il Global Interrupt Enable flag (GIE). Se questo bit è posto a 0, indipendentemente dai valori dei registri dei singoli interrupt, essi sono disabilitati.
Di seguito vengo descritti i ***registri imr*** e ***irq***: il primo rappresenta la cosiddetta maschera, ossia l'abilitazione, mentre il secondo indica se è in corso una richiesta di interrupt o meno.

REGISTRO IMR		
Bit 7:	1	Abilito richiesta di Interrupt #7 (Overflow del MultiTasking Timer)
Bit 6:	1	Abilito richiesta di Interrupt #6 (P0.2)
Bit 5:	1	Abilito richiesta di Interrupt #6 (Riservato)
Bit 4:	1	Abilito richiesta di Interrupt #6 (Overflow del Timer3)
Bit 3:	1	Abilito richiesta di Interrupt #6 (P0.0)
Bit 2:	1	Abilito richiesta di Interrupt #6 (Riservato)
Bit 1:	1	Abilito richiesta di Interrupt #6 (Overflow del Timer2)
Bit 0:	1	Abilito richiesta di Interrupt #6 (Overflow del Timer1)

Tabella 5.5: Registro abilitazione interrupt

REGISTRO IRQ		
Bit7:	1	Richiesta Interrupt #7 (Overflow MultiTasking Timer)
Bit6:	1	Richiesta Interrupt #7 (P0.2)
Bit5:	1	Richiesta Interrupt #7 (Riservato)
Bit4:	1	Richiesta Interrupt #7 (Overflow Timer3)
Bit3:	1	Richiesta Interrupt #7 (P0.0)
Bit2:	1	Richiesta Interrupt #7 (Riservato)
Bit1:	1	Richiesta Interrupt #7 (Overflow Timer2)
Bit0:	1	Richiesta Interrupt #7 (Overflow Timer1)

Tabella 5.6: Registro richiesta interrupt

Se una richiesta di interrupt si verifica mentre un'istruzione del programma sta settando il valore del GIE a 0, questa operazione verrà eseguita solamente dopo che la routine di interrupt è stata completata.

5.1.9 AUDIO WAKEUP

L'unità Audio Wakeup è un circuito analogico/digitale che può essere configurato in modo da dare luogo al "risveglio" dell'application board a seguito di uno di questi eventi audio:

- *doppio battito di mani*, o altri due suoni secchi e a breve distanza (temporale) uno dall'altro;
- *triplo battito di mani*, o altri tre suoni secchi, sempre a breve distanza;
- *fischio*;
- *ogni suono "forte" sopra una certa soglia di ampiezza*, della durata opzionale di 1 o 2 secondi

Poiché questa unità deve essere in ascolto a livelli di potenza (e quindi consumo) molto bassi, è necessario che la rilevazione di uno di questi quattro eventi avvenga ***senza l'interazione di un processore***. La configurazione delle modalità con cui deve avvenire il Wakeup viene effettuata tramite apposite istruzioni nel file di configurazione del codice, mentre l'abilitazione è opera del programma in esecuzione, che ha il compito di attivare l'opzione di Audio Wakeup appena prima di andare in Idle mode. Non è possibile adottare questa metodologia in stato di Sleep, perché è necessario che il CLK2 sia abilitato, cosa che in questo stato non accade.

Le librerie FluentChip™ contengono routines per rilevare ognuno dei quattro eventi elencati in precedenza.

5.1.10 INPUT ANALOGICI

Il front-end analogico (AFE, Analog Front End) rappresenta l'interfaccia di ingresso tra il mondo reale e l'RSC-4128: esso implementa la conversione da analogico a digitale di segnali di basso livello, che vengono derivati dal segnale elettrico generato dal microfono come trasduzione del suono. I blocchi principali di questa unità sono il convertitore analogico-digitale a 16 bit, il decimatore e il banco di filtri di canale.

Il segnale in ingresso viene amplificato da un preamplificatore che consente di avere quattro livelli di guadagno, selezionabili impostando l'apposito registro.

Ogni singolo circuito del front-end è dotato di enable, in modo che possa essere singolarmente spento, consentendo di utilizzare solamente ciò che è necessario per la particolare applicazione realizzata.

Il segnale così amplificato viene processato da un ***convertitore A/D*** tipo ***sigma-delta***, il quale genera un segnale discreto codificato con un bit. Questa stringa digitale è filtrata e decimata in modo da ottenere campioni a 16 bit, con un rate fisso pari a 18636 campioni al secondo.

L'ADC di tipo sigma-delta usa un convertitore analogico-digitale a 1 bit (che di fatto è un comparatore) e un DAC in retroazione; la presenza di un anello di feedback fa sì che il sistema si porti in equilibrio solo quando la tensione in ingresso all'integrato ha media nulla: questa condizione si verifica se la sequenza di bit prodotta dall'ADC ha media pari alla tensione in ingresso (questo valore viene estratto mediante un filtro passa basso digitale posto a valle del comparatore).

Il risultato finale è una parola a n bit (il numero di bit finale può essere anche relativamente elevato, fino a 20). Questo particolare convertitore sfrutta il principio del *sovracampionamento*: il convertitore analogico-digitale interno produce una sequenza di bit a frequenza molto alta, spesso maggiore della banda del segnale in ingresso (decine di MHz), consentendo di ottenere ottimi rapporti segnale-rumore nonostante il campionatore sia ad un solo bit. Il fattore di sovra campionamento, quindi, può variare da alcune centinaia ad alcune migliaia.

Il recupero dell'informazione associato al segnale richiede un filtro passa basso, il quale è integrato nel convertitore stesso.

Negli ADC sigma delta si riduce la complessità delle parti analogiche, complicando le parti digitali per ottenere le prestazioni volute; inoltre, i circuiti digitali sono meno ingombranti e quindi più

economici da produrre. Anche con filtri digitali di modesta complessità è possibile sintetizzare un numero di bit finale piuttosto alto, ricorrendo a livelli elevati di sovracampionamento (ad es. 10 bit con fattori di sovracampionamento dell'ordine del migliaio).

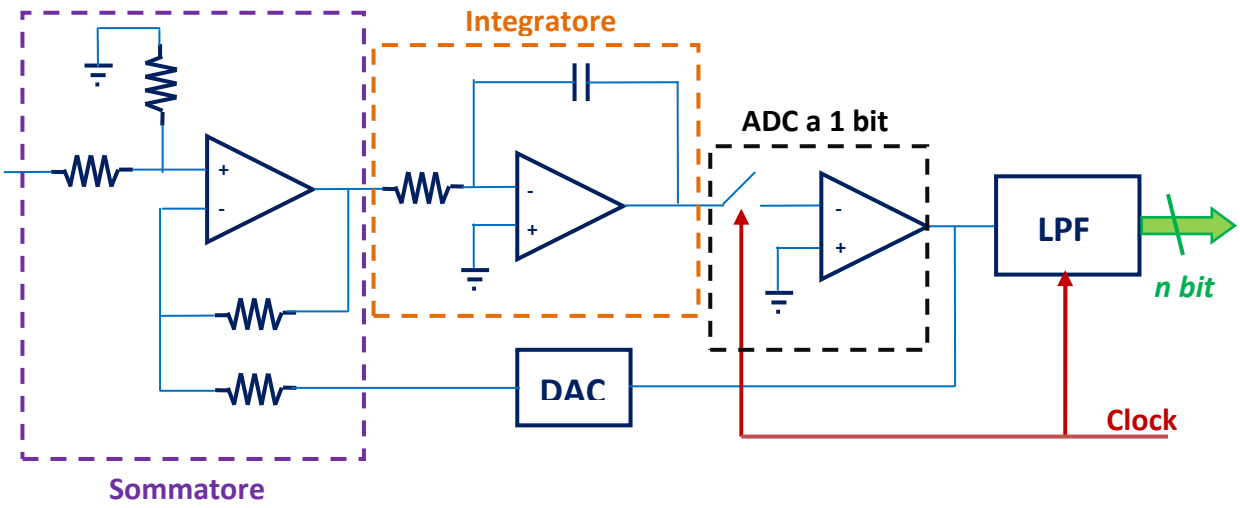


Figura 5.6: Circuito di un convertitore analogico-digitale di tipo sigma-delta

Il convertitore digitale analogico consiste in una rete R-2R con una risoluzione a 10 bit. Per poter pilotare l'altoparlante è richiesto l'uso di un amplificatore esterno: le sue caratteristiche determinano la miglior scelta in termini di impedenza dello speaker, e, di conseguenza, nell'adattamento e nel volume.

Con 10 bit di risoluzione, si ha che il valore del LSB (ovvero l'unità minima) è pari a $V_{fondo\_scala}/2^{10} = 3/1024 \cong 3\,mV$. Inoltre, per l'uscita digitale, vi sono due possibilità: full scale, nel qual caso la tensione può variare da 0 V a $V_{dd} - 1LSB$, o half scale, con un range compreso tra $V_{dd}/4$ e $3V_{dd}/4 - 1LSB$.

5.1.11 USCITA ANALOGICA PWM

La modulazione di larghezza d'impulso (dall'inglese Pulse Width Modulation) è un particolare metodo di *modulazione che permette di ottenere valori di tensione media differenti, agendo sulla durata dell'impulso positivo e di quello negativo*. Generalmente, infatti, la larghezza dell'impulso viene espressa in rapporto al periodo che intercorre tra due fronti di salita (o discesa), ricorrendo quindi al ben noto concetto di duty cycle: un dc pari a 0% indica un impulso di durata nulla, ossia

assenza di segnale, mentre un valore del 100% indica un impulso completamente positivo, trasferendo tutta l'energia disponibile. In varie applicazioni, si rende necessario l'impiego di un clock per poter determinare la posizione (inizio) degli impulsi, anche se in molte altre questo timing non è necessario, dato che al valore 0 (teoricamente con duty cycle dello 0%) viene assegnato un piccolo impulso, ossia un dc con percentuale bassissima ma non nulla.

Il vantaggio di questa tecnica è di riuscire ad ottenere le stesse prestazioni riducendo il consumo di potenza.

Sulla RSC-4128 è previsto un circuito che consente di regolare l'ampiezza di un impulso su due outputs, PWM0 e PWM1, riferendosi ad un periodo di durata predefinita e programmabile. Una delle due uscite viene mantenuta a massa mentre l'altra viene modulata. È inoltre presente un divisore che scala il clock di riferimento (CLK1, 14.32 MHz) per fattori costanti, pari a 4,6 o 7, risultando in segnali PWM di frequenza pari a 27.9 KHz, 18.6 KHz e 15.97 KHz rispettivamente.

5.2 MICROFONO

Uno degli elementi più importanti di un'applicazione per il riconoscimento vocale è indubbiamente l'interfaccia tra il mondo reale e quello digitale, ovvero il dispositivo in grado di captare i suoni e trasdurli in segnali elettrici: il ***microfono***.

Nella RSC-4128, un singolo microfono può essere usato sia come ingresso del front-end analogico (per il riconoscimento vocale), sia come "sensore" per captare gli eventi sonori per l'unità di Audio Wakeup.

Sulla linea del microfono è presente un condensatore che ha il compito di filtrare il segnale da componenti rumorose, la cui capacità può variare tra 33µF e 220µF: maggiore è il valore, più verranno ridotte le componenti del disturbo a basse frequenze (le quali non incidono in modo notevole sulla qualità del riconoscimento vocale, originate tipicamente dal ronzio della tensione di rete a 50/60 Hz oppure dalle fluttuazioni della tensione di alimentazione); al contrario, capacità ridotte agiscono sulle frequenze elevate.

I microfoni adottati in queste applicazioni possono essere sia omni-direzionali che uni-direzionali. Nella maggior parte dei casi, i primi, che risultano essere meno costosi dato che presentano caratteristiche standard, sono sufficienti per implementare con buona accuratezza il sistema che si vuole realizzare. Tuttavia, alcuni particolari campi di applicazione potrebbero richiedere una maggiore robustezza al rumore e, quindi, una migliore affidabilità nel riconoscimento: l'uso di microfoni uni-direzionali può garantire questo incremento di prestazioni, dato che essi hanno una

risposta (diagramma a lobi) che dipende dalla particolare direzione spaziale considerata, consentendo quindi di attenuare notevolmente eventuali componenti di disturbo provenienti da zone che vengono scarsamente considerate dalla risposta del microfono stesso. Questi dispositivi, però, richiedono uno studio e un'analisi del campo/ambiente di applicazione più accurato prima di procedere con l'installazione, onde evitare di trasformare i loro vantaggi in limiti di funzionamento, che possono pregiudicare, anche in maniera importante, il funzionamento dell'applicazione nel suo complesso.

In ultima analisi, si osservi come nel caso in cui il tipo di riconoscimento implementato sia lo speaker verification, non si dovrebbero utilizzare microfoni con un diametro inferiore ai 10 mm: è un'accortezza necessaria dato che la tecnologia SV è molto più suscettibile a piccole variazioni delle caratteristiche del suono descritte nei paragrafi iniziali della tesi e i microfoni con un diametro molto ridotto hanno una risposta fortemente dipendente dall'angolazione con cui viene ricevuta l'onda acustica, rivelandosi dunque inadeguati ai fini dello speaker verification. Sono però utilizzabili per tutte le altre tecnologie implementate dagli hardware e software della Sensory™.

5.2.1 SEDE DEL MICROFONO

È utile, al fine dell'applicazione oggetto di questo elaborato, dedicare alcune righe a considerazioni sulle *modalità di progettazione e realizzazione dell'alloggio del microfono*: *un corretto posizionamento dello stesso garantisce una migliore qualità ed accuratezza anche dal punto di vista acustico, e quindi del riconoscimento del parlato*.

Innanzitutto, la prima attenzione va prestata al fatto che ***il microfono dovrebbe essere posizionato in modo tale che sia in linea con la superficie su cui è montato e che non vi sia aria tra lo stesso e la sezione del case in cui esso è posto***: se quest'ultimo accorgimento non dovesse essere rispettato, si rischiano fenomeni di risonanza e di leggero eco, portando ad una degradazione del segnale audio in ingresso.

Figura 5.7: A sinistra, la posizione corretta del microfono in linea con la superficie, mentre a destra la modalità sbagliata (o, meglio, che potrebbe causare un deterioramento delle condizioni) di posizionamento

In secondo luogo, ***l'area di fronte al microfono deve essere libera da ostacoli***, per evitare che si creino riflessioni (echi, generando un'ulteriore interferenza che si aggiunge al segnale desiderato) oppure si dia luogo ad un'attenuazione che riduce l'ampiezza del suono stesso, rischiando di portarlo ad un valore tale per cui il riconoscimento si fa difficoltoso. ***Il diametro dell'alloggio del microfono dovrebbe essere di almeno 5 mm***, mentre, nel caso non sia possibile rimuoverla del tutto, ***qualsiasi superficie posta davanti al trasduttore stesso deve essere il più fine possibile***, preferibilmente non superiore agli 0.7 mm.

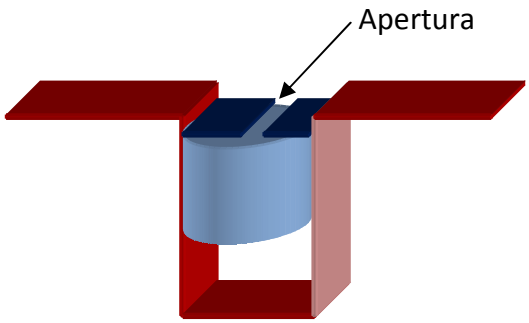


Figura 5.8: Microfono con una superficie di plastica posta davanti alla membrana (disegnata in blu scuro)

Infine, ***l'elemento dovrebbe essere isolato acusticamente il più possibile dalla sede***: questo può essere realizzato ricoprendo lo stesso (ovviamente, ad eccezione della membrana) con un materiale quale gomma o, in alcuni casi, anche un'apposita schiuma. Lo scopo è quello di prevenire eventuali disturbi dovuti al rumore generato dall'attrito del microfono stesso contro le pareti della sede.

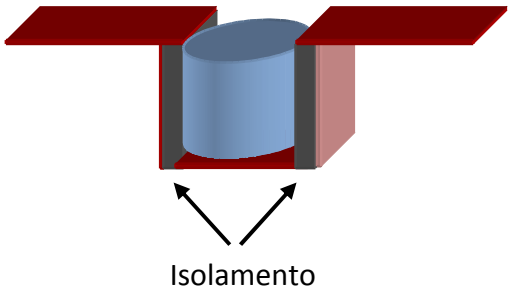


Figura 5.9: Microfono posizionato nella sede con l'apposito isolamento acustico per evitare echi e disturbi

5.2.2 GUADAGNO

Oltre a queste accortezze dal punto di vista meccanico, un altro elemento importante per un buon recognition è il calcolo del guadagno e del corretto valore della resistenza per il particolare tipo di microfono scelto. *L'amplificazione è determinata soprattutto da come viene configurato il programma, nel quale è possibile impostare le tipiche modalità di utilizzo dell'applicazione per il riconoscimento*: parlatore a distanza ravvicinata (***headset***, circa 20/30 cm), a media distanza (***arms_length***, un metro) o lontano (***far_mic***, per distanze superiori ai 2 metri). Nella seguente tabella sono riportati i valori del guadagno complessivo del sistema

Configurazione	Guadagno (dB)
Headset	-49
Arms_length	-44
Far_mic	-43

Tabella 5.7: Configurazione microfono

Un altro fattore importante è il valore della ***resistenza di bias*** del microfono, *necessaria per generare una tensione in continua che serve per "alimentare" il microfono stesso* (o, meglio, il condensatore presente al suo interno, facendone caricare le armature): questa differenza di potenziale subirà delle variazioni in funzione delle vibrazioni della membrana, ovvero dell'audio in ingresso. La relazione che permette di derivare il valore del componente resistivo è la seguente

$$R_s = I \cdot 10^{\frac{G-S}{20}} \tag{5.3}$$

con

G : guadagno complessivo del sistema espresso in dB

S : sensibilità del microfono usato (indicata nel datasheet ed espressa in dB)

I : impedenza del microfono (in ohm)

Un piccolo appunto riguardo la ***sensibilità***: un valore di 0 dB corrisponde alla generazione di 1V di tensione quando viene applicata una pressione di 1Pa da un'onda acustica alla frequenza di 1 KHz

$$\text{Sensibilità} : 0\ dB \rightarrow 1\ V/Pa\ \ @\ 1\ KHz$$

I valori di questo parametro richiesti per l'uso di tecnologie Sensory™ vanno dai -40 dB ai -46 dB. Nella tabella riportata sono indicati i valori consigliati per la resistenza di bias nei vari casi possibili, per due valori tipici di impedenza del microfono: 2.2 KΩ e 2.0 KΩ

Impedenza 2.2 KΩ	-38 dB	-39 dB	-40 dB	-41 dB	-42 dB	-43 dB	-44 dB	-45 dB	-46 dB
Far	1.1KΩ	1.2KΩ	1.3KΩ	1.5KΩ	1.8KΩ	2.0KΩ	2.2KΩ	2.4KΩ	2.7K Ω
Arms	1.0KΩ	1.1KΩ	1.2KΩ	1.3KΩ	1.5KΩ	1.8KΩ	2.0KΩ	2.2KΩ	2.4KΩ
Headset	620Ω	620Ω	680Ω	750Ω	910Ω	1.0KΩ	1.1KΩ	1.2KΩ	1.3KΩ

Impedenza 2.0 KΩ	-38 dB	-39 dB	-40 dB	-41 dB	-42 dB	-43 dB	-44 dB	-45 dB	-46 dB
Far	1.0KΩ	1.1KΩ	1.2KΩ	1.5KΩ	1.5KΩ	1.8KΩ	2.0KΩ	2.2KΩ	2.4K Ω
Arms	910Ω	1.0KΩ	1.1KΩ	1.2KΩ	1.5KΩ	1.5KΩ	1.8KΩ	2.0KΩ	2.2KΩ
Headset	510Ω	580Ω	620Ω	710Ω	820Ω	910Ω	1.0KΩ	1.1KΩ	1.2KΩ

Tabella 5.8: Valori impedenza bias

6. SOFTWARE

Ora che la parte hardware del progetto è stata descritta, si passa ad un livello più alto di analisi del sistema di riconoscimento. In particolare, verranno studiate le varie modalità con cui è possibile realizzare questa applicazione.

Speaker Independent (SI) : questo metodo di riconoscimento è caratterizzato dal cosiddetto pre-training, ovvero il vocabolario è pre-definito da chi ha realizzato l'applicazione e non può essere modificato dall'utente (a meno che, ovviamente, non si intervenga a livello software correggendo il codice). *Il prodotto con tecnologia SI è pronto all'uso, senza alcun bisogno che l'utente debba memorizzare i vocaboli da riconoscere*. Questa modalità viene spesso impiegata in dispositivi di nuova generazione in cui i comandi (ovvero un insieme definito di parole) vengono pronunciati a voce e non più inseriti manualmente: si pensi ai navigatori satellitari, ai cellulari e a molte altre apparecchiature.
Per quanto riguarda il caso particolare della RSC-4128, i vocaboli speaker independent sono tipicamente memorizzati nella memoria on-chip dell'integrato ed è prevista un'apposita funzione nelle librerie fornite dalla FluentChip™ che consente di effettuare questo tipo di riconoscimento: essa è chiamata ***Text-to-SI*** (***T2SI***) poiché l'insieme dei vocaboli SI viene generato con un opportuno programma (fornito dalla Sensory™ unitamente a tutta l'application board), il QuickT2SI. Questo tool permette di generare i comandi (memorizzati come fonemi) semplicemente selezionando la lingua desiderata e scrivendo tutti i termini del vocabolario.

Speaker Dependent (SD) : *con questa tecnologia, si rende necessario eseguire una fase di addestramento (training) prima di procedere con il riconoscimento vero e proprio*. L'utente deve quindi memorizzare tutti i vocaboli desiderati, cercando di fare in modo che ciò avvenga in una situazione in cui non vi siano rumori e disturbi che possano degradare la qualità dell'audio, e quindi la capacità del sistema di estrarre le caratteristiche che si renderanno poi necessarie per lo speech recognition.
Relativamente alla demo board della Sensory™ i templates SD possono essere salvati sulla memoria RAM interna (con un limite massimo) oppure su dispositivi off-chip. La scelta tra le due opportunità è quali esclusivamente dettata da un tradeoff tra il costo e il numero di vocaboli da memorizzare.

Con la tecnologia FluentChip™ si può implementare l'operazione di training grazie alla funzione di ***MakeTmpltSD***, mentre quella di riconoscimento con ***SDRecog***.

SD&SI : in molte applicazioni si rende necessario un compromesso tra le due tecniche illustrate in precedenza, in cui, accanto ad un set di vocaboli predefiniti, l'utente ha l'opportunità di customizzare il sistema, potendo memorizzare degli ulteriori templates che andranno ad aggiungersi al vocabolario SI pre-esistente. Generalmente, la parte speaker independent rappresenta le voci di comando o di menù del dispositivo.
Per implementare questa tecnologia sui dispositivi della Sensory™, è nuovamente disponibile un'apposita funzione nelle librerie della FluentChip™: per simboleggiare ancora più l'unione dei due metodi, prende il nome di ***T2SISD***, Text-to-SI+SD. *Non viene fatta alcuna distinzione tra i vocaboli SI e quelli salvati dall'utente (SD): si tratta di una normalissima operazione di best-matching tra il suono pronunciato dall'utente e il template che meglio lo rappresenta.*
Si noti come questa funzione non è solo una comodità per poter implementare dependent e independent speaker recognition con un unico codice, ma è indispensabile se si considerano sullo stesso livello i vocaboli pre-impostati e quelli definiti dall'utente, ovvero se il vocabolario delle caratteristiche con il quale viene fatto il confronto per riconoscere un suono è composto sia da parole SI che SD.

Speaker Verification (SV) : *è una variante del riconoscimento speaker dependent, utilizzata per realizzare delle soluzioni di sicurezza vocali, come le password*. Si tratta a tutti gli effetti di un riconoscimento SD, in cui l'utente deve compiere inizialmente una fase di training per memorizzare la parola, con la sola differenza che in questo caso viene data più rilevanza ai parametri biometrici, ovvero quelli che identificano la voce del particolare parlante (è importante sì che il dispositivo "capisca" quale termine sia stato pronunciato e che questo effettivamente coincida con quello memorizzato, ma ciò deve verificarsi solamente se chi ha parlato è la stessa persona che in precedenza ha memorizzato il tutto). Scopo principale è dunque quello di identificare il parlante.
Sulla RSC-4128, trattandosi di una tecnologia praticamente identica allo speaker dependent, la password può essere salvata sia sulla RAM interna che su eventuali memorie esterne.
Sfruttando la tecnologia della Sensory™ è possibile avere insiemi multipli (più di un vocabolario) di templates SI, SD o SV, senza che essi tendano a "mescolarsi" e mantenendosi sempre indipendenti gli uni dagli altri. L'unica limitazione è ovviamente dovuta alla memoria disponibile.

6.1 ACCURATEZZA DEL RICONOSCIMENTO

Finora si è discusso circa le modalità di implementazione del riconoscimento vocale, valutando vantaggi e limitazioni di ciascuna di esse. Considerando un generico contesto di utilizzo, senza che siano richieste particolari specifiche, è utile descrivere quali possano essere gli aspetti da valutare per garantire il miglior riconoscimento possibile.

Indubbiamente, il primo elemento che viene naturale prendere in considerazione è il ***vocabolario***, ovvero l'insieme di tutte le parole (speaker independent o dependent che siano) che sono state memorizzate e che possono essere riconosciute. Considerando che, in linea teorica, il numero massimo delle stesse è determinato dalla memoria disponibile, è opportuno limitare il numero di parole di ciascun set: quest'ultimo deve comprendere tutti i templates utili, ma escludere ogni vocabolo superfluo, rendendo in questo modo l'operazione di recognition meno complessa. Infatti, ogni volta che il sistema "sente" un suono, lo va a comparare con tutti quelli presenti in memoria, verificando se vi è corrispondenza o meno: la presenza, nel vocabolario, di parole indesiderate dà luogo a tempi di calcolo maggiori (avendo più parole da confrontare) e ad una probabilità più elevata di falsi riconoscimenti, soprattutto in condizioni rumorose.

Si rivela un ottimo accorgimento, quando possibile, ***separare parole simili a livello di pronuncia, ma anche di contenuto fonetico, in più insiemi distinti*** (ovvero non riconoscibili allo stesso istante), dimodochè si riducano gli *errori* cosiddetti di sostituzione, cioè di scambiare un vocabolo per un altro appartenente allo stesso set.

In aggiunta a quanto detto finora, sono da ***evitare parole*** (o frasi) ***troppo lunghe***, contenenti più di 4/5 sillabe, dato che portano con sé un'informazione fonetica molto elevata, e quindi una maggiore difficoltà durante la fase di confronto e di matching con i termini del vocabolario. Queste parole sono generalmente meno soggette ai falsi riconoscimenti, cioè i casi in cui un suono viene erroneamente associato ad una parola del set, ma più propense a falsi rigetti, ossia il vocabolo viene pronunciato correttamente dall'utente, ma a causa delle sue numerose caratteristiche e del suo elevato contenuto fonetico dovuto alla sua lunghezza, non viene riconosciuto. ***Parole molto corte*** (una sola sillaba) *presentano vantaggi e svantaggi opposti a quanto descritto in precedenza*, risultando quindi con una probabilità di falsi riconoscimenti molto più elevata, mentre la percentuale di rigetti è limitata.

Ora è utile soffermarsi un istante sull'aspetto più fonetico del linguaggio: infatti, nel parlato, alcuni suoni sono più facilmente udibili ed individuabili in presenza di rumore. La distinzione tra vocali e consonanti, ad esempio, è molto importante ed è utile riprendere il modo in cui esse vengono distinte (quantomeno dal punto di vista articolatorio): le prime vengono prodotte semplicemente

grazie all'aria proveniente dai polmoni, senza che vi sia alcuna ostruzione del canale fonatorio, condizione indispensabile per produrre una consonante. L'assenza di ostruzioni dà luogo alla maggiore intensità sonora (espressa in dB) che caratterizza i suoni vocalici rispetto a quelli consonantici. Inoltre, l'intensità di una vocale è dovuta anche all'effetto della cassa di risonanza, la cui forma determina le varie vocali.

Per quanto riguarda le consonanti, l'ostruzione del canale fonatorio, con il conseguente blocco o costrizione della colonna d'aria originata dai polmoni, può essere di gradi diversi: con un'ostruzione totale si ottengono le cosiddette consonanti occlusive (a volte dette anche plosive), mentre quando l'aria è costretta a passare attraverso una piccola fessura causando una turbolenza si originano le fricative. Questi due tipi sono detti ostruenti, mentre tutte le altre consonanti sono dette sonoranti, in quanto presentano una sonorità spontanea.

Tra le varie categorie, l'ultima è quella che presenta una maggiore intensità: ***la presenza in una parola (o frase) si sole consonanti sonoranti e di vocali, senza che vi siano plosive e fricative, rendono molto più difficile il riconoscimento dell'intero vocabolo***. La miglior combinazione sarebbe quella di avere sia fricative/plosive e sonoranti/vocali.

Un metodo implementabile con la tecnologia FluentChip™ che consente di ridurre in modo drastico i falsi riconoscimenti è l'utilizzo della ***trigger word***. La sua utilità è ben chiara già dal nome, dato che "trigger" in inglese significa "innescare": *pronunciare la trigger word in un sistema di speech recongition consente di abilitare a tutti gli effetti il dispositivo, permettendo di riconoscere ed attuare tutti i comandi del vocabolario*. In caso contrario, il sistema rimane in uno stato di attesa, senza poter eseguire alcuna operazione fin quando non viene pronunciata la parola necessaria per abilitarlo (trigger). Ultimamente, questa tecnica si sta diffondendo anche per identificare i vari dispositivi di un sistema più ampio, come ad esempio un "centro multimediale" comprendente Tv, lettori dvd, stereo e altre apparecchiature, in cui per poter comandare vocalmente ciascuna di esse si rende prima necessario abilitarle, pronunciando il loro nome.

Per quanto riguarda l'application board della Sensory™ sono disponibili tre modalità di utilizzo della trigger word: speaker independent, speaker dependent e l'unione dei due. Nel primo caso la trigger è facilmente realizzabile, in quanto si sfrutta sempre il T2SI, ovvero la possibilità di creare templates grazie al QuickT2SI™, il quale dà anche la possibilità di fare una verifica, in modo da constatare se la parola scelta può essere o meno una trigger word (e quindi non dia luogo a dei falsi riconoscimenti anch'essa).

Al contrario, nel caso SD è necessaria una fase preliminare di training, per memorizzare la trigger, come del resto tutti gli altri termini.

Come si è visto fino ad ora, è evidente come ***uno dei principali limiti di un sistema di riconoscimento è la percentuale di falsi validi e di falsi rigetti***. L'ottimo (che è anche l'obiettivo che il progettista si prefigge) sarebbe quello di riuscire a ridurre entrambi i rate di errore, ma purtroppo *essi sono correlati in modo inverso*: migliorando l'affidabilità dell'applicazione nei confronti di una delle due problematiche, si riduce l'altra. Questa richiesta può non essere fondamentale in alcuni dispositivi, per cui è si tende a privilegiare la robustezza nei confronti dei falsi validi o dei falsi rigetti, ma generalmente è richiesto un buon compresso tra i due.

Per questi motivi, *spesso i dispositivi prevedono un livello di confidenza con il quale effettuare l'operazione di riconoscimento*: si tratta di un modo per stabilire la qualità dei suoni che si vogliono identificare, determinando la percentuale delle utterance che devono essere riconosciute in modo corrette, e, quindi, impostando implicitamente a livello software la probabilità di false accept e reject. Le tecnologie FluentChip™ adottano livelli di differenti a seconda del tipo di riconoscimento implementato. Nel caso particolare di SI, il bilanciamento tra falsi validi e falsi rigetti viene impostato direttamente nel tool Quick T2SI™ dal progettista, nella voce "Out of Vocabulary Sensitivity"

6.2 TECNOLOGIA FLUENTCHIP™

Le librerie della FluentChip™ sfruttano due algoritmi per effettuare lo speech recognition: le catene di Markov, in particolare il già descritto Hidden Markov Model (HMM) e il pattern recognition. Il primo metodo viene impiegato nel riconoscimento indipendente dal parlante, mentre il secondo per lo speaker dependent o verification.

Tipicamente, il modello HMM genera un database contenente le informazioni acustiche estratte mentre l'utente sta parlando, per poi analizzarle e decidere quale vocabolo sia stato pronunciato, decisione che è nota non appena l'utente conclude la frase. Con la funzione Text-to-Speaker Independent, il confronto con i termini del vocabolario viene fatto basandosi sulla grammatica (infatti, il tool disponibile per il T2SI, ovvero il QuickT2SI, consente di scegliere la lingua dei potenziali utenti, in modo che si selezionino implicitamente anche le regole grammatiche da seguire per il confronto).

Nel caso in cui l'algoritmo sia basato sul pattern, le operazioni sono leggermente differenti: innanzitutto si ha una fase cosiddetta di block collection, ossia di raccolta delle caratteristiche fonetico-acustiche del suono, seguita da un pattern generation (patgen), che porta a determinare un possibile "percorso" tra i vari blocchi, definendo quindi una possibile parola pronunciata, ed infine

il riconoscimento vero e proprio. Generalmente queste operazioni sono svolte in sequenza dal programma, ma, in alcuni casi particolari, possono avvenire (quasi) contemporaneamente.

Nello SD e SV si fa riferimento ai vocaboli con il termine template, ovvero un percorso tipico (costituito anche da due o più pattern) che rappresenta una specifica parola memorizzata nel vocabolario.

Descritte brevemente le modalità con cui le varie tecniche di riconoscimento vengono implementate dalle librerie della FluentChip™, e dato che i campi di applicazioni usuali per ciascuna di esse sono già stati specificati in precedenza, ora verranno illustrate alcune delle caratteristiche principali che hanno le funzioni

FUNZIONI PER RICONSOCIMENTO SI		
Fattore	**T2SI**	**T2SISD**
Massimo numero di parole del vocabolario	30*	30+8SD
Sillabe per utterance	1-8	1-8**
Robustezza al rumore	Ottima	Buona
Trigger word	Si	Si
Wordspot	Si*	No
Fase di training (necessità di memorizzate i template)	No	Si
Costo del vocabolario	Bassa	Bassa
Tempo di vocabolario	Veloce	Veloce

** T2SI Wordspot è limitato ad un vocabolario con al massimo 7 parole*

*** I template SD sono limitati a 4 sillabe per utterance*

FUNZIONI PER RICONSOCIMENTO SD		
Fattore	**SD**	**T2SISD**
Massimo numero di parole del vocabolario	64	8+30SD
Robustezza al rumore	Buona	Ottima
Utilizzo di risorse (relativo)	Basso	Alto
Sillabe per utterance	1-4	1-8*
Trigger word	No	Si
Wordspot	No	No
On-chip templates	7	7

** per i vocaboli SD il limite di sillabe per utterance è 4*

Riassumendo, le funzioni implementabili con le librerie della FlunetChip™ sono le seguenti

Speaker Independent (SI)

- Text-to-Speaker Independent (T2SI)
- Text-to-Speaker Independent/Dependent (T2SISD)

Speaker Dependent (SD)

- Speaker Dependent (SD)
- Fast Speaker Dependent (SDF)
- SD con Continuos Listening (SDCL)
- Text-to-Speaker Independent/Speaker Dependent (T2SISD)

Speaker Verification (SV)

- Speaker Verification (SV)
- Fast Speaker Verification (SVF)

6.2.1 TEXT-TO-SPEAKER INDEPENDENT RECOGNITION (T2SI)

Come già detto in precedenza, il T2SI è una tecnologia di riconoscimento del parlato indipendente dall'utente, che viene utilizzata dai microcontrollori della famiglia RSC-4x, basata sul Hidden Markov Model, sfruttando come algoritmo di ricerca quello studiato e ideato da Viterbi. In fase di sviluppo dell'applicazione, il vocabolario è definito inserendo (scrivendo) tutte le parole in un apposito tool, il già nominato QuickT2SI™, il quale permette di creare un file contenente i dati relativi alle HMM, chiamato "grammars" (definizione dei singoli suoni e fonemi di cui è composta ogni singola parola del vocabolario e loro relazione), e un altro in cui viene memorizzato il modello acustico che sta alla base della rete neurale di identificazione del suono (per questo motivo viene anche definito "net"). Sempre a livello software, può essere impostato il tradeoff tra false accept/reject, a seconda del campo di utilizzo ultimo del dispositivo.

Inoltre, la funzione T2SI può operare sia in trigger mode, che in command mode: il tool di sviluppo dovrà quindi essere in grado di generare entrambi i file "grammars" (due e non uno solo in quanto la trigger word non è trattata esattamente come un generico comando, per cui è preferibile tenere separate anche le loro caratteristiche), che dovranno successivamente essere inclusi nel codice sorgente dell'applicazione.

In fase di esecuzione, il T2SI usa il modello acustico per stimare i suoni pronunciati e, una volta estratti i parametri caratteristici, con le catene di Markov determina, con una stima statistica, quale termine del vocabolario corrisponde ai suoni. Il risultato di questa operazione è un numero intero,

chiamato classID, con il quale si fa riferimento ad una precisa word memorizzata; infatti, per semplicità, il sistema salva ogni vocabolo associando ad esso un numero, in modo ordinale. È poi previsto un particolare valore, identificato da "NOTA" (None Of The Above), che viene restituito ogni volta che l'applicazione ha riconosciuto che è stata pronunciata una parola, ma non è in grado di determinare quale.

Ad ogni operazione di riconoscimento, inoltre, il T2SI genera anche un codice di errore, indicante se la procedura di recognition ha avuto successo e con quale livello di confidenza (alto, medio o basso), oppure se non è stato possibile identificare il suono, nel qual caso può anche restituire alcune cause del possibile errore (come, ad esempio, la pronuncia con un volume troppo basso, oppure un ambiente disturbato, e altri ancora).

Il tool permette dunque di generare questi piccoli database contenenti le informazioni necessarie agli algoritmi per funzionare, ma è compito dello sviluppatore quello di importarli e linkarli nel proprio codice sorgente. Di seguito vengono elencati tutti i possibili file generati quando si preme il pulsante "build" dopo avere completato un progetto con QuickT2SI™:

comm_rscApp_<project>.inc	File in linguaggio assembly da includere nel progetto; definisce i valori della classID per ogni comando del vocabolario e il grammar
comm_rscApp_<project>.h	File in linguaggio C da includere nel progetto; definisce i valori della classID per ogni comano del vocabolario e il grammar
Trig_rscApp_<project>.inc	File in linguaggio assembly da includere nel progetto; definisce il valore di NOTA per la trigger word e il grammar
Trig_rscApp_<project>.h	File in linguaggio C da includere nel progetto; definisce il valore di NOTA per la trigger word e il grammar
Comm_rscGram_<project>.mco	File contenente la grammar e linkabile dal codice (in linguaggio assembler)
Trig_rscGram_<project>.mco	File contenente la grammar e linkabile dal codice (in linguaggio assembler)
Rscnet_<project>.mco	File contenente la grammar e linkabile dal codice (in linguaggio assembler)

Un'applicazione in linguaggio C (o C++) necessita dei files .h e .mco per importare correttamente tutti i parametri necessari alle HMM per lavorare correttamente. Al contrario, in linguaggio assembly, servono i .inc e .mco.

Il software QuickT2SI™ può quindi realizzare files per un sistema avente un solo vocabolario per la trigger e/o per i comandi. Tuttavia, alcune applicazioni richiedono più di un vocabolario: per sopperire a questa esigenza, è previsto un ***Acoustic Model Combiner***, ovvero un particolare *tool che consente di creare un vocabolario che sia il risultato della condivisione di più vocabolari*. Il grande vantaggio di questa tecnica è la possibilità di ridurre l'occupazione di zone di memoria a parità di accuratezza nel riconoscimento. Come si può intuire, questo metodo deve essere utilizzato dallo sviluppatore solamente in ultima fase, quando tutti i vocabolari "inferiori" sono già stati creati.
Per chiudere questo paragrafo, verrà presentata la funzione T2SI da utilizzare nel codice

_T2SI(acousticModel, grammar, knob, timeout, trailing, *pRes)

PARAMETRI	
acousticModel	Indirizzo fisico in cui memorizzato il file generato con il tool QuickT2SI™ e contenente il modello acustico
Grammar	Indirizzo in cui è memorizzato il file necessario alle Hidden Markov Model
Knob	Livello di confidenza con cui deve avvenire il riconoscimento (0…4) 0 : più risultati con alta confidenza, meno con media e bassa 2 : valore tipico, non si privilegia nessuno dei due casi 4 : meno risultati con alta confidenza, più con media e bassa N.B. : questo parametro è ignorato qualora si debba riconoscere la trigger word
Timeout	Massimo tempo di ascolto, espresso in secondi, con un range che può variare da 1 a 254. 0 : nessun timeout, il sistema rimane in ascolto finchè non identifica una parola 255 : valore di default, imposta a 3 secondi il timeout per i comandi e non lo prevede per la trigger
Trailing	Minima durata del silenzio dopo l'ultimo suono percepito. Espresso in unità di 0.025 s, con valori che posso andare da 4 a 36 (ovvero, in secondi, da 0.1s a 0.9s) Questo parametro determina fortemente la latenza del sistema, ovvero l'intervallo che intercorre tra l'ultimo suono "sentito" dal dispositivo e il tempo in cui esso restituisce il risultato dell'operazione di riconoscimento
pRes	Puntatore al struttura contente i risultati

Un po' di attenzione va prestata al parametro "trailing", che, come detto, influisce sul tempo di latenza del sistema. Valori piccoli comportano una più veloce risposta, e quindi un riconoscimento più immediato del suono, ma possono causare errori nel caso in cui il vocabolario contenga parole che, al loro interno, presentano dei tempi di silenzio importanti. Di default, trailing è impostato a 15 (400 ms), un tempo che garantisce delle buone prestazioni per la maggior parte dei vocabolari, ma rallenta leggermente la velocità di risposta dell'applicazione. Questo valore si può ridurre, mantenendo la stessa affidabilità, qualora i vocaboli presentino, al loro interno, tempi molto bassi di

silenzio o addirittura essi siano assenti (questo fatto non implica necessariamente che anche la parola sia corta). Infine, si osservi come questo parametro vada calibrato sul caso peggiore tra tutti i possibili termini del vocabolario.

6.2.2 SPEAKER DEPENDENT (SD)

Come è ben noto, la tecnologia SD consiste in una prima fase di addestramento, per la creazione del vocabolario definito dall'utente e la sua memorizzazione, e la successiva operazione di riconoscimento dell'insieme dei vocaboli appena definiti.

La procedura di training si basa sulle tecniche di pattern generation (patgen) per collezionare una serie di blocchi di informazioni acustiche e creare un pattern specifico per una determinata parola: tutto ciò è realizzato mediante la routine MakeTmpltSd. Ogni singolo template, tipicamente, ha una dimensione di circa 256 byte.

_MakeTmpltSd(timeout, sepSil, maxWords)

PARAMETRI	
Timeout	Tempo Massimo di attesa per il parlato, in unità di 1 secondo (0..255, 0 : nessun timeout)
sepSil	Intervallo di tempo entro cui aspettare una seconda utterance dopo che la prima è terminata, in unità di 0.25 s
maxWords	Massimo numero di utterance per ogni template

L'utilità del sepSil sta nel fatto che tra due utterance successive l'utente può esitare, oppure un altro suono può verificarsi.

_RecogSd (classes, knob, timeout, sepSil, maxWords, *pRes)

PARAMETRI	
Classes	Numero di tremplates memorizzati
knob	Livello di riconoscimento
Timeout	Tempo Massimo di attesa per il parlato, in unità di 1 secondo (0..255, 0 : nessun timeout)
sepSil	Intervallo di tempo entro cui aspettare una seconda utterance dopo che la prima è terminata, in unità di 0.25 s
maxWords	Massimo numero di utterance per ogni template
pRes	Puntatore al struttura contente i risultati

6.2.3 TEXT-TO-SPEAKER INDEPENDENT/SPEAKER DEPENDENT (T2SISD)

T2SISD è una tecnologia con la quale una singola utterance può essere riconosciuta sia a partire da un vocabolario realizzato con l'apposito tool (quindi SI) oppure confrontandola con dei templates memorizzati dall'utente (ossia SD).

Questa funzione non è altro che la combinazione delle tecniche che implementano il normale riconoscimento speaker dependent e speaker indpendent in modo singolo, con la presenza di un supervisore che consente di coordinare le due operazioni e di valutare quale è il risultato più soddisfacente. Quando il sistema che utilizza T2SISD riconosce che è stata pronunciata una parola, per prima cosa esegue il confronto con il vocabolario SI, restituendo un risultato relativo al migliore termine speaker independent che rappresenta il suono sentito. Fatto questo, viene generato un particolare pattern utilizzato per il confronto con i templates SD, dal quale si ricava ancora un nuovo score, questa volta relativo allo speaker dependent. Compito del supervisore è quello di valutare quale dei due risultati ottenuti sia il migliore. Due sono le funzioni principali per implementare questo tipo di speech recongition: la prima, _MakeTmpltT2SISD, consente di effettuare la fase di training per le parole dipendenti dall'utente, la seconda, _T2SISD, per procedere alla vera e propria identificazione delle stesse.

_MakeTmpltT2SISD(acousticModel, grammarModel, knob, timeout, trailing, *pRes)

PARAMETRI	
acousticModel	Indirizzo del modello acustico da usare per il riconoscimento T2SISD
grammarModel	Indirizzo del grammar model da usare per il riconoscimento T2SISD
knob	Inutilizzato
timeout	Massimo tempo di ascolto, espresso in secondi, con un range che può variare da 1 a 254. 0 : nessun timeout, il sistema rimane in ascolto finchè non identifica una parola 255 : è un valore di default, che imposta a 3 secondi il timeout per i comandi e non lo prevede per la trigger
Trailing	Minima durata del silenzio dopo l'ultimo suono percepito. Espresso in unità di 0.025 s, con valori che posso andare da 4 a 36 (ovvero, in secondi, da 0.1s a 0.9s) Questo parametro determina fortemente la latenza del sistema, ovvero l'intervallo che intercorre tra l'ultimo suono "sentito" dal dispositivo e il tempo in cui esso restituisce il risultato dell'operazione di riconoscimento
pRes	Puntatore al struttura contente i risultati

_T2SISD(acousticModel, grammar, knob, timeout, trailing, sdClasses *pRes)

PARAMETRI	
acousticModel	Vedi _T2SI
grammarModel	Vedi _T2SI
knob	Vedi _T2SI
timeout	Vedi _T2SI
Trailing	Vedi _T2SI
sdClasses	Numero di templates SD memorizzati (0 è il caso in cui si implementi solamente il normale riconoscimento speaker independent T2SI
pRes	Vedi _T2SI

6.2.4 FAST SPEAKER DEPENDENT (SDF)

Fast recognition è una variante delle tecnologie tradizionali, che consente di effettuare l'operazione di generazione del pattern e di riconoscimento in modalità multi-tasking, ovvero in parallelo: questa modalità di funzionamento implica un utilizzo maggiore dell'unità centrale, rendendo quindi impossibile l'impiego di questa tecnica in applicazioni in cui sono necessarie anche altre operazioni di background durante il riconoscimento (come, ad esempio, monitorare eventuali segnali esterni). La fase di addestramento e di memorizzazione dei template è la medesima dello SD e SI, mentre l'operazione di identificazione del suono avviene in contemporanea con la collezione di blocchi del parlato del segnale sconosciuto, necessari per poter estrarre le features dello stesso. Quando viene rilevato un intervallo di silenzio di durata sufficiente, allora il sistema considera "conclusa" l'utterance e procede alla scelta del template che meglio la rappresenta.

6.2.5 SPEAKER VERIFICATION

Tecnologia dipendente dal parlatore, viene impiegata in applicazioni di sicurezza vocali. SV è molto simile allo Speaker Dependent, anche se viene prestata più attenzione al modo in cui viene pronunciata una particolare parola, rendendo questa modalità fortemente legata alla voce della persona e alle caratteristiche che la contraddistinguono dalle altre. Come per lo SD, sono previste tre fasi: addestramento, memorizzazione dei template e conseguente riconoscimento.

Speaker Verification può effettuare il riconoscimento di una sola parola oppure di una sequenza (fino ad un massimo di quattro), in modo che possa essere utilizzata per applicazioni con password multiple.

7. BUS KNX

Con il termine ***sistema di automazione*** si intende *l'insieme di sensori, attuatori, controllori, unità centrali (se presenti) e periferiche, opportunamente interconnessi tra loro ed in grado di condividere stati, eventi e variabili relativi ai processi controllati nell'edificio.*

I sistemi HBES (Home and Building Electronic System) sono sistemi dedicati al controllo ed alla automazione degli edifici ad uso residenziale civile, terziario e industriale. Essi sono nati per:

- integrare tutte le funzioni ed i servizi presenti negli edifici in un unico impianto e/o progetto (ad esempio, illuminazione, sicurezza, termoregolazione, accessi,…);
- ottimizzare le interfacce tra i vari sottosistemi o impianti;
- supervisionare e controllare l'edificio.

Grazie a questi sistemi è possibile realizzare un'unica rete di comunicazione (cablata o wireless) in cui tutti i dispositivi (o i sottosistemi in generale) si scambiano le informazioni relative a dati, eventi, misure, allarmi…

Tipicamente, l'architettura di rete può essere rappresentata con la classica piramide, distinguendo, all'interno di un impianto di automazione, tre livelli caratteristici in base alla distribuzione dell'intelligenza (ovvero la capacità di calcolo e controllo) ed alle funzionalità specifiche dei vari componenti HW e SW:

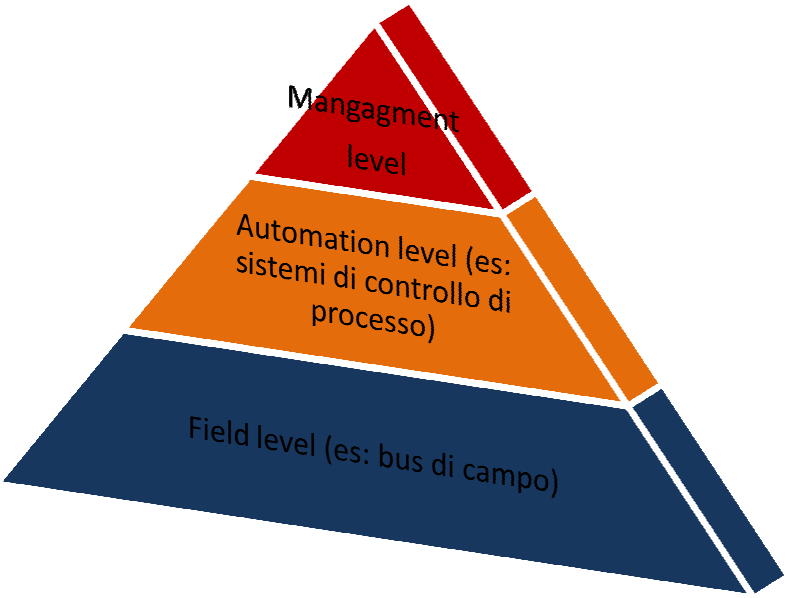
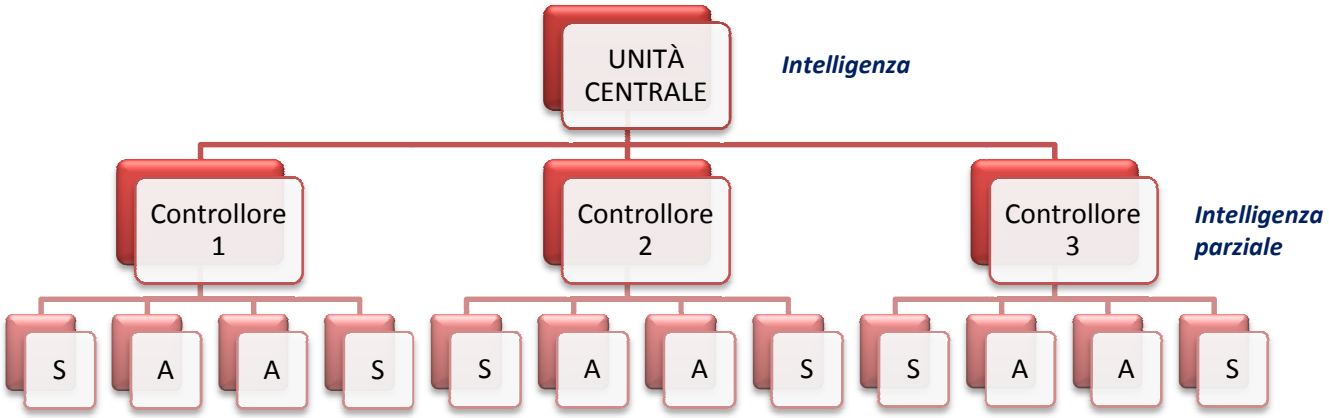


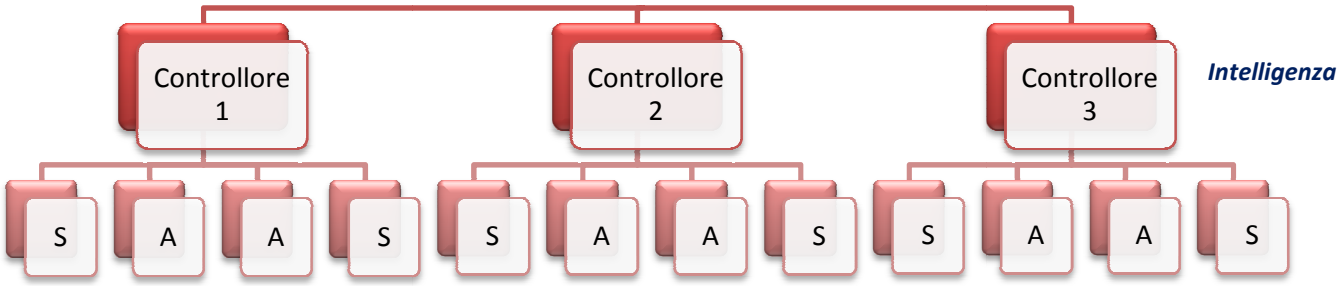
Figura 7.1: Livelli architettura di rete del bus KNX

Nel particolare, si possono definire tre tipi differenti di architetture, che si differenziano per il livello in cui viene concentrata l'intelligenza:

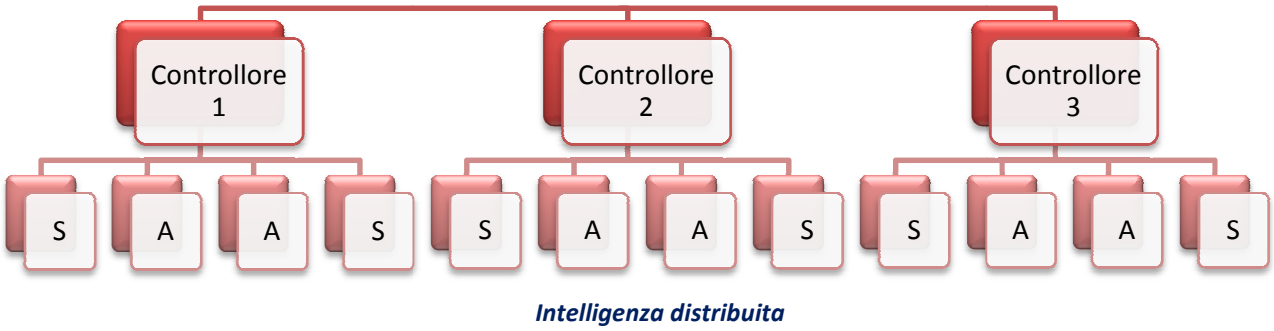
- **Architettura gerarchica**:



- **Architettura piatta**:



- **Architettura ad intelligenza distribuita**:



I sistemi HBES si distinguono principalmente in due classi, a seconda della complessità/dimensione del progetto ed alla sua funzionalità/costo: la ***Home Automation*** (*Domotica*) e la ***Building Automation***. Come si può dedurre anche dai termini che identificano le due tipologie, si nota come la prima sia destinata all'uso in abitazioni o complessi residenziali, mentre la seconda per edifici di uso civile che richiedono un'impiantistica studiata appositamente. La figura seguente illustra in modo schematico ed intuitivo ciò che contraddistingue e differenzia le due categorie:

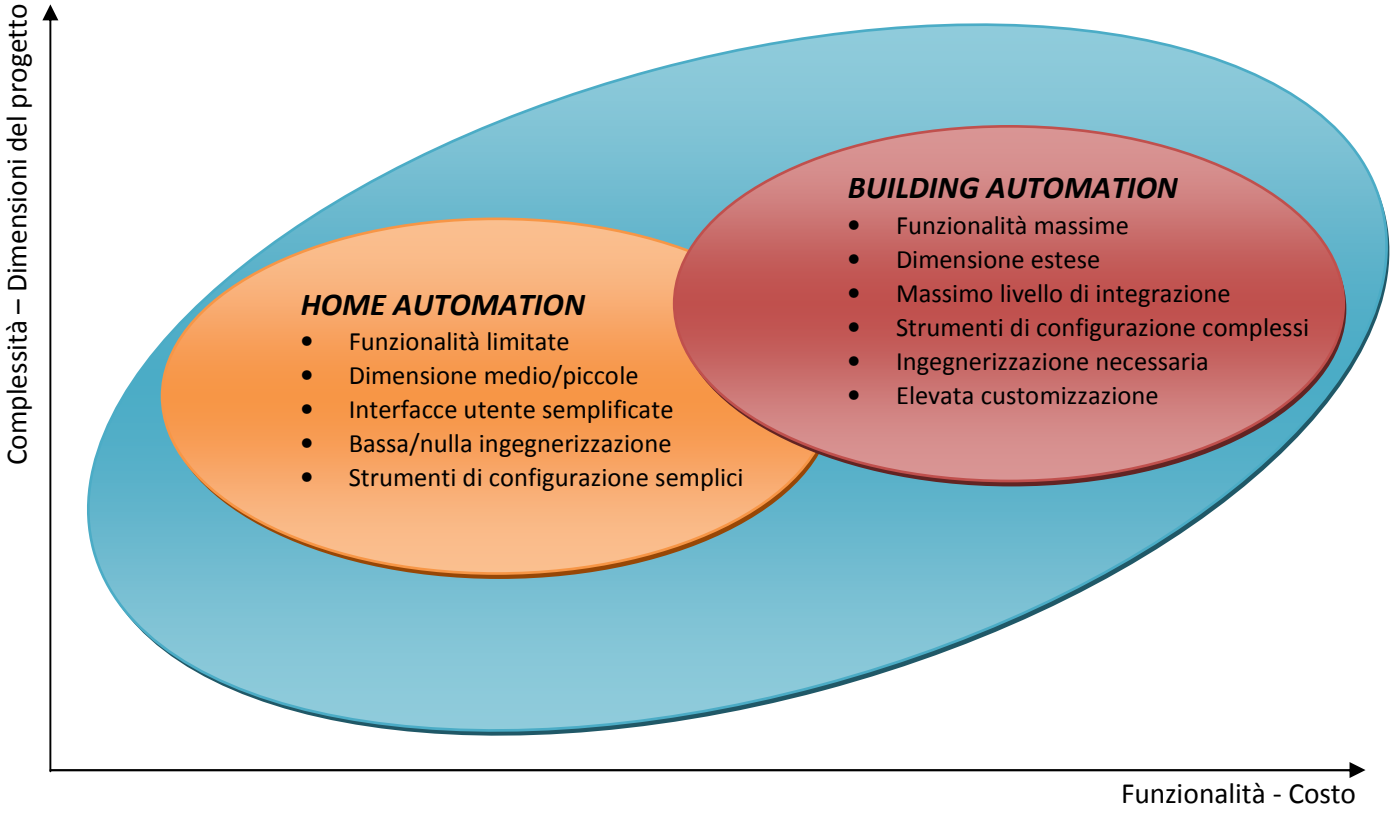


Figura 7.2: Ambiti di utilizzo dei sistemi domotici a seconda della loro complessità

Il mercato di questi sistemi si segmenta in funzione di vari fattori:

- ***Destinazione d'uso finale***

  Vengono offerte funzioni specifiche in base alla destinazione d'uso, la quale identifica i mercati di riferimento; da questa caratteristica si possono distinguere chiaramente i sistemi dedicati alla Home Automation (palazzi, complessi residenziali, case, ville) rispetto alla Building Automation (banche, edifici scolastici, ospedali, hotel, centri commerciali, industrie, musei, chiese, teatri, centri sportivi…)

- ***Contenuto tecnologico o del servizio***

  I sistemi sono stati sviluppati per supportare specifiche funzioni relative al controllo di una singola applicazione o servizio (impianto elettrico, illuminazione, riscaldamento, antincendio, sicurezza, comunicazione…)

- ***Dimensione del sistema***

  Vengono proposti in base al numero di nodi da controllare (o dispositivi) e alle distanze che possono essere raggiunte dalla rete di comunicazione. Generalmente, per sistemi con meno di 1000 nodi è sufficiente un installatore, mentre per impianti di dimensioni maggiori è richiesta la presenza e la progettazione da parte di un sistemista.

7.1 L'ASSOCIAZIONE KONEX E LO STANDARD KNX: BREVE STORIA

L'associazione ***Konnex*** è stata fondata nel Maggio 1999 a Bruxelles ed è nata dalla fusione di tre aziende europee impegnate nella promozione di sistemi intelligenti per abitazioni ed edifici, nello specifico:

- BCI (Francia) attiva nella diffusione del sistema BatiBUS;
- EIB Association (Belgio), produttrice di dispositivi domotici basati sul bus EIB;
- European Home Systems Association, EHSA, (Olanda), specializzata nella powerline con tecnologia ad onda convogliata (dati spediti sul cavo dell'alimentazione 230V);

Essa persegue i seguenti obiettivi:

- definizione di un nuovo standard "KNX" realmente aperto per la gestione intelligente di abitazioni ed edifici;
- affermazione del marchio KNX come simbolo di qualità ed interoperabilità di prodotti di diversi costruttori (multi-vendor interworking);
- diffusione di KNX come standard europeo e mondiale.

Fin quando sarà necessario, l'associazione offrirà la propria assistenza anche per i sistemi antecedenti, compresa la certificazione conforme a tali standard. Dal momento che il sistema EIB è totalmente compatibile con KNX la maggior parte degli apparecchi potrà riportare entrambi i loghi.

Al momento della sua istituzione, l'Associazione KNX era costituita da 9 membri, ma questo numero è aumentato, ed oggi si contano più di 217 associati (situazione a Marzo 2011); l'elenco aggiornato dei membri può essere consultato all'indirizzo web www.knx.org.
Alla fine del 2003 lo Standard KNX è stato approvato dal CENELEC (European Committee of Electrotechnical Standardization) quale standard europeo per l'automazione di case ed edifici (HBES, Home and Building Electronic Systems). Conforme alla normativa europea EN 50090.
Nel 2007, invece, è stato approvato come standard mondiale (ISO/IEC 14543-3); nello stesso anno, la traduzione in cinese dello standard internazionale ha ottenuto l'approvazione come norma cinese GB/Z 20965.

7.2 TECNOLOGIA E TIPOLOGIE DI CONFIGURAZIONE

Per quanto riguarda l'utilizzo del mezzo di comunicazione, il più diffuso è il cosiddetto ***twisted pair*** (*cavo ritorto schermato*): esso svolge la funzione di cavo di controllo, posato parallelamente a quello per l'alimentazione da rete (230V, 50 Hz). Ciò si traduce in

- notevole riduzione dell'estensione del cablaggio rispetto ad una tecnologia di installazione convenzionale, perché i dispositivi bus vengono disposti in modo decentralizzato;
- maggior numero delle possibili funzioni del sistema;
- migliore trasparenza dell'installazione.

Le funzioni di questo cavo sono le seguenti:

- Collegare carichi e comandi;
- Fornire ai dispositivi bus l'alimentazione (nella maggior parte dei casi).

Non è necessaria la presenza di un'unità di controllo centrale (ad esempio, il PC) perché tutti i dispostivi bus hanno ciascuno la propria intelligenza autonoma. È anche possibile implementare KNX sul cavo da 230V (mezzo trasmissivo ***Powerline***, a cui si fa riferimento con *Powerline 110*), via radio (mezzo trasmissivo ***Radio Frequency***) e tramite Ethernet (***KNX IP***). Usando appropriati gateway è anche possibile trasmettere telegrammi (termine con cui, in questo Standard, si fa riferimento ai pacchetti) KNX su altri mezzi, come le fibre ottiche. Il particolare mezzo usato in un dispositivo è visibile nell'etichetta del prodotto.

Mezzo	*Trasmissione via*	*Aree preferite di applicazione*
Twisted pair	Cavo di comando separato	Nuovi impianti e ampi lavori di ristrutturazione, livello alto di affidabilità di trasmissione
Powerline	Reste esistente *	Nei luoghi nei quali non è necessario che venga installato un cavo di controllo supplementare ed è disponibile quello della rete da 230 V
Radio frequenza	Linea radio	Nei luoghi ove non sono presenti cavi o non sono richiesti
IP	Ethernet	Installazioni di grandi dimensione dove è necessaria una dorsale veloce

* la trasmissione avviene tra fase e neutro nella PL110

Tabella 7.1: Mezzi trasmissivi per bus KNX

In base a quanto riportato sull'etichetta del dispositivo, inoltre, questo può essere configurato (ovvero collegato e parametrizzato) tramite:

S-Mode: Modalità System

È la modalità più completa per affrontare impianti complessi in edifici di media/grande dimensione. Presenta un elevato livello di flessibilità nella definizione delle funzionalità e dell'indirizzamento dei dispositivi; necessita del programma ETS, quindi di un PC, per la configurazione dell'impianto: per questi motivi, S-Mode è orientato a professionisti con una buona formazione e preparazione in materia.

E-Mode: Easy Mode

Questa modalità si applica a dispositivi dotati di funzioni preprogrammate dal costruttore e con alcuni parametri di funzionamento già predisposti in fabbrica. La possibilità di indirizzamento è limitata, come del resto il set di funzioni disponibili, a beneficio però di una semplificazione nella procedura di programmazione e messa in servizio. Non necessita di mezzi informatici per la configurazione delle apparecchiature, ma di altri meccanismi di programmazione (come, ad esempio, il configuratore del sistema Chorus Easy).

Caratteristiche	EASY Mode	SYSTEM Mode
Numero massimo dispositivi	255	Oltre 64000
Topologia rete	Una sola linea (max $1000\ m$)	15 linee x 15 aree
Programmazione	Configuratore GW90837 oppure software ETS	Solo con software ETS
Protocollo	KNX	KNX
Cablaggio	Bus KNX	Bus KNX
Funzioni programmabili sui dispositivi	Solo le "principali"	"Molte" e con svariati parametri impostabili
Indirizzamento	Riconoscimento automatico	Manuale, con pulsante di programmazione

Tabella 7.2: Modalità programmazione dispositivi EIB

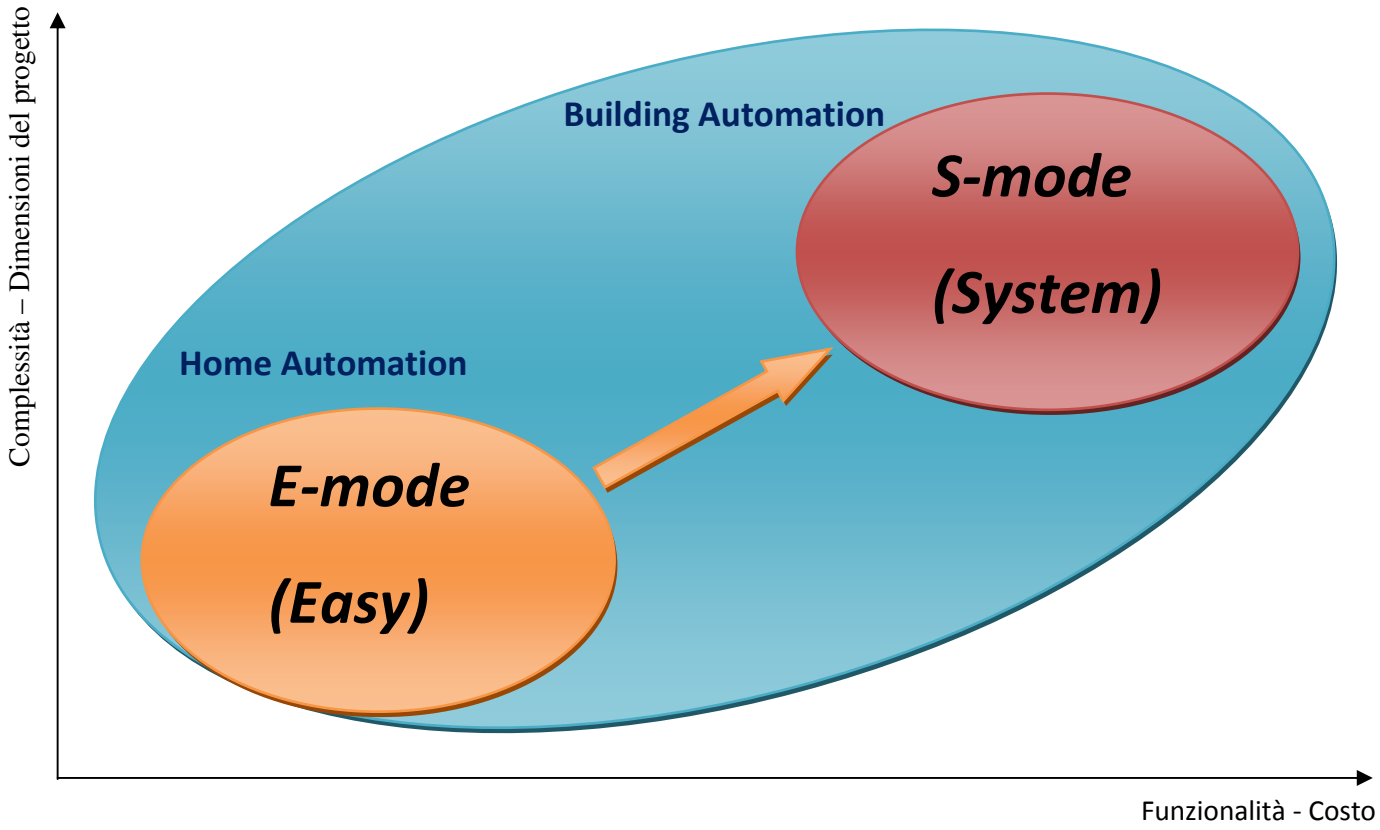


Figura 7.3: Confronto grafico prestazioni/costi tra le due modalità di programmazione

7.3 TOPOLOGIA E COMUNICAZIONE

La differenza fondamentale tra una tecnologia bus e un impianto tradizionale sta nel fatto che in quest'ultimo il comando e l'attuazione del comando sul carico sono integrati nello stesso componente e realizzati tramite una interruzione o deviazione di una linea di energia (230V).
Al contrario, *in un sistema bus, normalmente il comando è separato funzionalmente dall'attuatore ed i due dispositivi si trasmettono le informazioni utilizzando la rete*.
Questa nuova modalità impiantistica porta notevoli vantaggi:

- collegamento libero su un mezzo di comunicazione condiviso, il bus, contrariamente a quanto accade in un sistema tradizionale in cui si hanno collegamenti punto-punto;
- dispositivi intercomunicanti tramite interfacce native;
- cablaggio semplificato;
- separazione elettrica tra comando ed attuatore (il che implica maggiore sicurezza e più funzioni implementabili);
- facile espandibilità/riconfigurabilità dell'impianto, diversamente da quando accadeva nell'impiantistica classica in cui il cablaggio era rigido e spesso complesso;
- facile interfacciamento verso l'esterno.

Il bus veicola le informazioni in strutture di dati di tipo binario (sequenze di bit strutturate in pacchetti, nel protocollo EIB chiamate ***telegramma***) in cui sono riportati indirizzo del mittente, del destinatario e contenuto dell'informazione stessa (dati).
L'architettura del sistema KNX è ad intelligenza distribuita, ovvero il controllo è distribuito su ogni singolo componente periferico, sia esso un sensore o un attuatore, e ciascuno di questi gestisce autonomamente tutta (o in parte) la sovrintendenza della sua specifica funzione. In particolare, ***la topologia prevede una dorsale principale alla quale si collegano, tramite appositi dispositivi di accoppiamento, delle linee secondarie***.
Vi sono anche dei ***vincoli per*** quanto riguarda il ***dimensionamento di una linea bus*** (segmento di linea):

- *massimo 64 apparecchi per segmento di linea*;
- *massimo 2 alimentatori* (con distanza minima tra loro di 200 m);
- *massimo 1000 m di cavo bus*;

Tramite un ripetitore di segnale è possibile estendere una linea con altri 1000 m e, di conseguenza, altri 64 dispositivi. Sono poi previsti al massimo 4 segmenti di linea, e quindi un massimo di 3 ripetitori di segnale, ovvero non si possono installare più di 255 dispositivi (indirizzi fisici) su uno stesso segmento di linea.
Inoltre, tra le singole apparecchiature e alimentatore devono essere rispettate delle distanze, per poter garantire la corretta comunicazione sul bus:

- distanza dispositivo-dispositivo max. 700 m
- distanza alimentatore-dispositivo max. 350 m

L'***indirizzo fisico*** di ciascun dispositivo viene rappresentato in base alla disposizione topologica dello stesso nella rete bus ed alla sua progressione numerica nella linea di appartenenza.

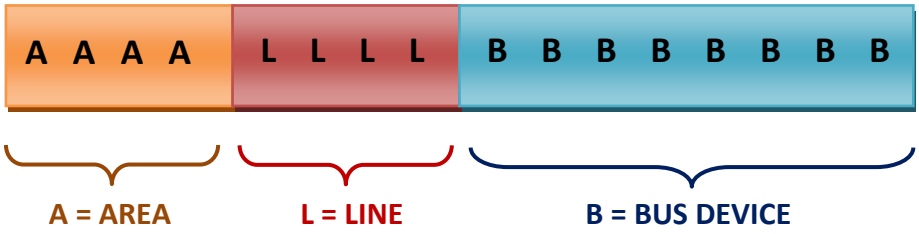


Figura 7.4: Indirizzo fisico del bus KNX

Tutti i dispositivi sviluppati per essere configurati in Easy o System devono poter essere configurati anche con il software unico di progettazione ETS.
Le funzioni supportate da un dispositivo KNX sono configurate nel programma applicativo ETS certificato e ad esse associato. Un singolo dispositivo può avere più programmi applicativi associati. L'insieme dei programmi applicativi certificati da Konnex ed associati a ciascun prodotto viene rilasciato dal costruttore sottoforma di un file (database) importabile da ETS4. La conoscenza dei programmi applicativi e del set di variabili od oggetti di comunicazione gestiti, nonché dei relativi parametri ad essi associati, permette di sapere effettivamente le funzioni svolte dal dispositivo sia internamente che durante la comunicazione via bus.
Gli oggetti di comunicazione o datapoint sono standardizzati da KNX sia nella loro sintassi, sia nella semantica per garantire così l'interoperabilità tra dispositivi di differenti costruttori.
Il formato di un *telegramma*, ovvero del pacchetto scambiato sul bus di comunicazione, è quello rappresentato in figura

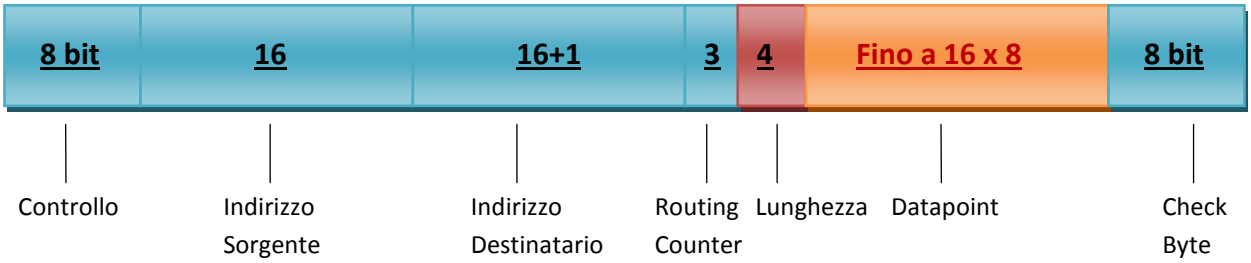


Figura 7.5: Telegramma

Elemento fondamentale nella comunicazione KNX sono gli ***oggetti di comunicazione*** (o ***datapoint***): essi rappresentano i dati (o, meglio, i tipi) veri e propri. Come si può notare dalla figura in cui è riportata la rappresentazione del telegramma, essi hanno una dimensione variabile tra 1 bit e 16 byte, a seconda della complessità del dato e della funzionalità che quest'ultimo può implementare.

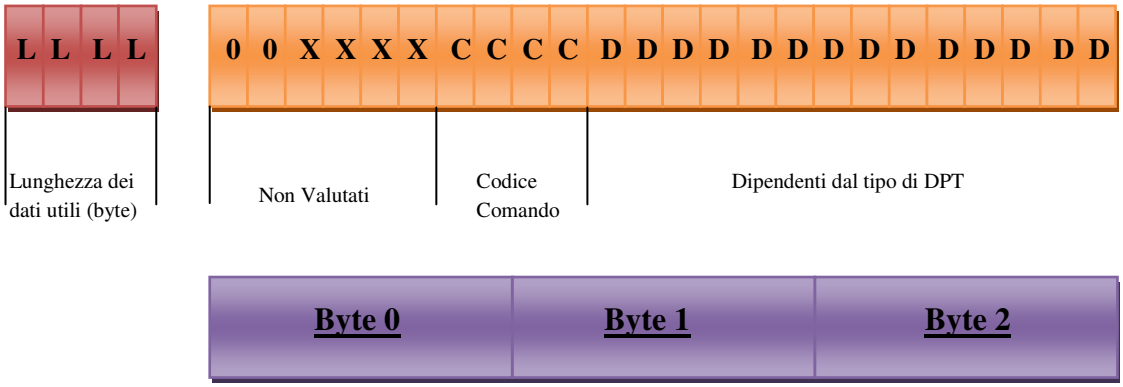


Figura 7.6: Bit del blocco dedicato al datapoint (si noti come i colori rosso e arancione corrispondano a quelli della figura 7.5)

A differenza degli indirizzi fisici, i quali vengono utilizzati per identificare in modo univoco ciascuna dispositivo, in fase di comunicazione vengono impiegati i cosiddetti ***indirizzi di gruppo***: essi non dipendono dalla disposizione topologica del dispositivo, ma solamente dagli oggetti di comunicazione ad essa associati. Si tratta dunque di valori che hanno un significato funzionale, in base allo specifico controllo che gli oggetti associati (datapoint) attuano sui dispositivi (ingressi/sensori e uscite/attuatori).

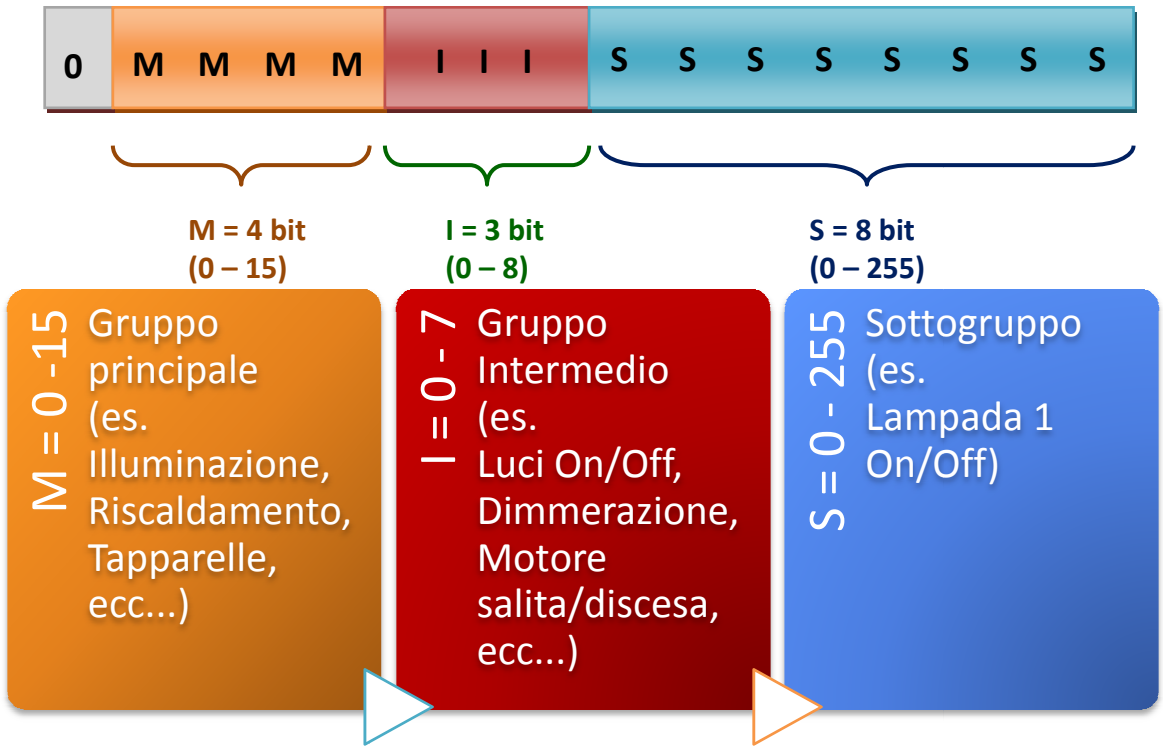


Figura 7.7: Indirizzo di gruppo

ID	*Nome*	*Dimensione*	*Funzione*
1.001	Switch (Commutazione)	1 bit	Commutazione
1.008	Up/Down	1 bit	Movimentazione
2.001	Switch Control (Priorità)	2 bit	Comando Prioritario
5.001	Scaling (valore %)	1 byte	Invio Valore in %
3.007	Dimming (Regolazione)	4 bit	Regolazione Luce
9.001	Value Temp (Temperatura)	2 byte	Misura di Temperatura
10.001	Time	3 byte	Ora
11.001	Date	3 byte	Data
14.xyz	4 Octet Float Value	4 byte	Contatore a 4 byte
15.000	Access	4 byte	Transito Controllo Accessi
16.000	String	14 byte	Stringa di Caratteri

Tabella 7.3: Alcuni oggetti di comunicazione standard KNX: ad un tipo possono essere associati differenti sottotipi aventi funzioni diverse

8. PROGETTO

L'idea che ha ispirato questo progetto è stata la possibilità di realizzare un'applicazione dimostrativa, a partire dalla demo board descritta nei capitoli precedenti, che permettesse di valutare la miglior configurazione, i potenziali impieghi e l'affidabilità della stessa nelle operazioni di riconoscimento del parlato. In particolare, lo studio è stato effettuato integrando la scheda della Sensory™ in un sistema domotico, con comunicazione su bus KNX, interfacciandolo con più dispositivi (attuatori) EIB di Gewiss s.p.a.

Come ultimo passo, dai risultati di queste osservazioni e sperimentazioni si è considerata la possibilità di implementare le stesse funzioni in un dispositivo stand-alone, ovvero utilizzabile in un impianto tradizionale (senza bus), riducendo l'occupazione di spazio (possibilmente ad un modulo DIN) ed il costo di produzione, mantenendo inalterate le prestazioni.

Prima di procedere con la parte pratica del progetto, è stato dedicato del tempo allo studio teorico delle tecniche di automatic speech recognition, per chiarire le possibili metodologie d'impiego e i concetti teorici che stanno alla base del processo di riconoscimento, in modo da poter già delineare, seppur in modo approssimato, quali modalità potessero essere più convenienti ai nostri fini. In particolare, è stata prestata particolare attenzione alle varie tecniche di recognition, dipendenti o meno dal particolare utente.

Successivamente si è passati alla fase di sviluppo vero e proprio: partendo dall'analisi dell'application board, ed in particolare dai codici degli esempi applicativi forniti con il toolkit, è stato scritto il nucleo centrale del codice dell'applicazione, il quale, a seguito di numerose prove e modifiche (che hanno permesso di approfondire la conoscenza delle tecnologie di riconoscimento, rendendo evidenti limiti e potenzialità di ciascuna di essa) è stato migliorato fino alla sua versione finale. Da sola, però, la demo board non ha grandi potenzialità, se non quella di implementare i semplici esempi applicativi di cui si è accennato: grazie alla porta seriale di cui dispone (RS232), è stata interfacciata con un dispositivo EIB di Gewiss (il ricevitore di segnale a radiofrequenza, GW14831) opportunamente modificato, sia a livello hardware che software, per poter comunicare con il VR Stamp™. Con questa configurazione, quindi, la scheda dimostrativa può gestire diversi attuatori, grazie al ricevitore RF che funge da interfaccia per il bus KNX.

Per ricreare il sistema domotico di un'abitazione, oltre all'unità centrale rappresentata dalla scheda della Sensory™ e al modulo RF, sono presenti anche diversi attuatori, ognuno dei quali comanda uno (o più) led a 230V, simulando l'accensione e lo spegnimento delle luci di casa oppure il comando per aprire/chiudere le tapparelle.

Di seguito vengono elencati e brevemente descritti tutti i dispositivi impiegati per realizzare il pannello dimostrativo.

Alimentatore da guida DIN (GW90702)

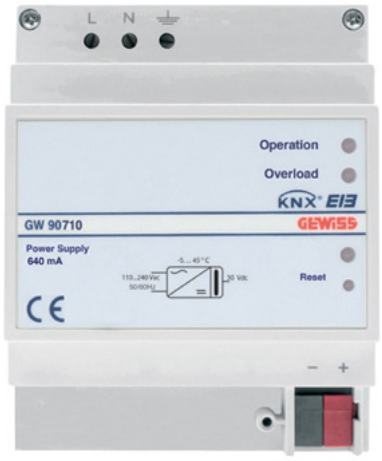


Figura 8.1: Alimentatore bus EIB GW90702

Alimentatore autoprotetto per dispositivi EIB impiegato per fornire e controllare la tensione del bus. È alimentato da rete (230V, 50/60 Hz) e in uscita fornisce una tensione in continua pari a $29V \pm 1V$ con una corrente massima di 320 mA.

Presenta una bobina integrata in modo da evitare che la tensione di rete interferisca con la linea bus ed un pulsante di reset per il ripristino di tutti i dispositivi collegati alla linea dati stessa.

Il numero massimo di apparecchiature alimentabili va calcolato in funzione del loro assorbimento, ricordando che ciascuna di esse, secondo gli standard KNX, *può consumare al massimo 10 mA*.

È prevista la possibilità di collegarvi una batteria da 6-15 Ah a $12\,V_{DC}$, in modo che possa alimentare il bus (per un limitato lasso di tempo) anche nel caso in cui venga a mancare la tensione di rete.

Dispone inoltre di due led utilizzati per segnalare condizioni di funzionamento anomale del dispositivo: la prima soglia è definita ***overcurrent*** (o *overload*) ed indica che il carico complessivo a valle dell'alimentatore sta consumando (complessivamente) più di 320 mA; qualora questa corrente dovesse continuare ad aumentare a tal punto da superare un'ulteriore soglia (impostata a livello hardware tramite dei partitori di tensione), il dispositivo entra in ***protezione***, ovvero smette di alimentare il bus onde evitare dei cortocircuiti.

Interfaccia USB da guida DIN (GW90706A)

Figura 8.2: Interfaccia USB per bus EIB GW90706A

Dispositivo a due moduli impiegato per collegare un PC al bus KNX. In questo modo è possibile configurare l'impianto tramite software ETS, ma anche controllare il traffico, in modo da poter verificare gli oggetti di comunicazione che i dispositivi si stanno effettivamente scambiando.
L'unico vincolo è la lunghezza del cavo USB che non può superare i 5 m.

Attuatore 4 canali 16AX Easy da guida DIN (GW90836)

Figura 8.3: Attuatore 4 canali KNX GW90836

Attuatore a 4 canali per l'attivazione di carichi attraverso contatti (relè) in uscita (normalmente aperti, NA) privi di potenza, ciascuno dei quali ha una corrente massima di commutazione pari a 16

A. È utilizzato in molte applicazioni, grazie alle numerose funzioni che esso può implementare: commutazione on/off, comandi temporizzati e prioritari, gestione di scenari.
Nel caso in cui venga a mancare la tensione di alimentazione del bus, al suo ripristino i contatti di uscita rimangono aperti.
Tra i carichi che esso può pilotare, si ricordano lampade a incandescenza e alogene ($230\ V_{AC}$) fino ad un massimo di 3000 W, trasformatori toroidali con una potenza di 3000 W o trasformatori elettronici non superiori ai 2000 W, ma anche lampade a basso consumo. È dotato di pulsanti frontali per il comando locale delle uscite e di led di segnalazione della chiusura del contatto.

Attuatore dimmer 500W resistivo/induttivo da guida DIN (GW90844)

Figura 8.4: Dimmer per la regolazione della luminosità GW90844

Attuatore dimmer per la regolazione di luminosità di lampade ad incandescenza e lampade alogene controllate da trasformatori ad avvolgimento, con funzione soft-start di protezione delle lampade stesse (l'accensione non avviene istantaneamente, ma con una rampa graduale fino a raggiungere la tensione di lavoro nominale). Il dispositivo viene alimentato dal bus per quanto riguarda la scheda logica, mentre necessita della tensione di rete per la parte di potenza.
Il suo funzionamento è basato sui concetti di zero-crossing e di energia media di un segnale, nel caso particolare una sinusoide: l'intensità della lampada viene regolata aumentando o diminuendo l'energia media in uscita dal dimmer; questa operazione è resa possibile "tagliando" la semionda della tensione di rete ad un istante proporzionale alla percentuale di dimmerazione che si vuole avere (100% significa che non vi è alcun taglio, mentre al 50% la semionda viene interrotta a metà del periodo). Quanto detto risulta più chiaro osservando l'immagine che segue e la nota

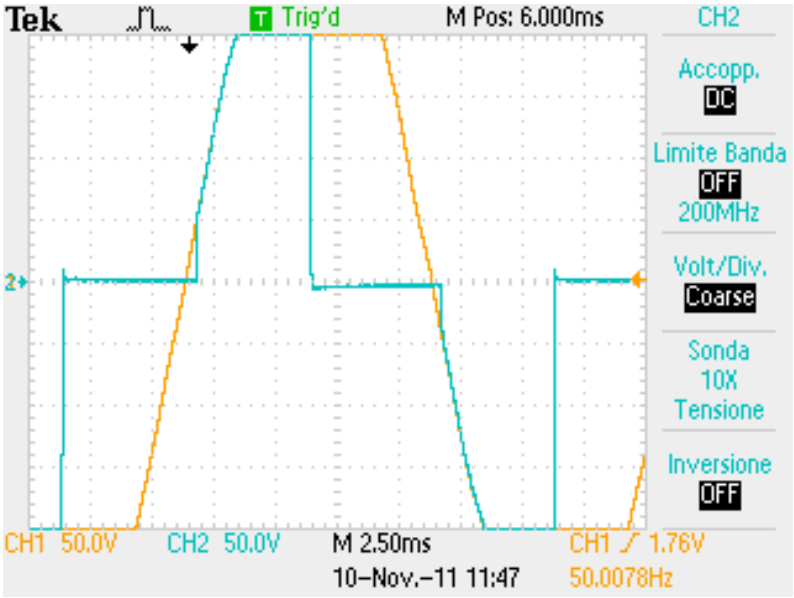


Figura 8.5: Taglio dell'onda in uscita dal dimmer. Questa immagine è stata "catturata" con un oscilloscopio: la sinusoide arancio rappresenta la tensione di rete (si noti, in basso a destra, la frequenza della stessa, pari a circa $50\ Hz$), mentre quella azzurra è la forma d'onda in uscita dal dispositivo. Come si nota essa è troncata a metà del semiperiodo (lo si confronti con l'onda d'ingresso arancio), il che significa che il carico disporrà del 50% dell'energia totale, per cui si ha una dimmerazione del 50%

Perché non si abbiano problemi è necessario stimare correttamente i punti in cui l'onda attraversa lo zero, in modo che essi vengano presi come riferimento per effettuare il taglio della stessa. Qualora questa operazione venga eseguita in modo sbagliato, si può incorrere in condizioni dannose per il dispositivo, soprattutto se il carico è induttivo.

Come per l'attuatore a 4 canali, anche il dimmer può essere impiegato per l'esecuzione di comandi on/off e per la gestione di scenari. Inoltre, è presente un pulsante frontale per il comando locale e un led per la segnalazione dello stato del carico.

Nel caso in cui venga a mancare la tensione del bus, il dispositivo porta la fase dimmerata in "off" (0%), ma al suo ripristino l'uscita si riporta al valore precedente la caduta di tensione.

Attuatore per tapparelle 1 canale 6A Easy da incasso (GW12767)

Figura 8.6: Attuatore per tapparelle ad 1 canale GW12767

Attuatore a un canale per l'attivazione di un motore per la movimentazione di tapparelle, tende, veneziane, avvolgibili, ecc, attraverso due contatti di uscita (NA) interbloccati e privi di potenziale; in uscita restituisce la tensione di rete con una corrente massima di 6 A.
Viene utilizzato per l'esecuzione di comandi di movimentazione/regolazione/arresto, comandi prioritari e scenari.
Sono previsti anche in questo caso due pulsanti frontali di comando locale e due led di segnalazione chiusura contatto di uscita. Se l'alimentazione del bus dovesse mancare, non viene apportata alcuna modifica allo stato dell'uscita.

Ricevitore RF 8 canali Easy (GW14776)

Figura 8.7: Ricevitore RF EIB GW14776

Ricevitore radio a 8 canali per interfacciare gli apparecchi di comando del sistema "RF Comando e Controllo" con gli attuatori del sistema KNX Easy; ad ogni canale di ingresso radio possono essere associati fino a 4 dispositivi di comando.

Consente l'invio di comandi on/off con gestione dei fronti, di inversione, temporizzati, prioritari, comandi per la gestione di tapparelle, dimmer o scenari (memorizzazione e attivazione).

Questo dispositivo andrebbe utilizzato con la serie di apparecchi RF di Gewiss, ma nel progetto ha un'altra funzione rispetto a quella per cui è stato originariamente pensato: esso funge da interfaccia tra la demo board e il bus KNX. Il perché sia stato scelto proprio il ricevitore RF è da ricercare nel fatto che la realizzazione di un'interfaccia apposita avrebbe richiesto un costo eccessivo in termini di tempo e costo; inoltre, il GW14776 si presta alle modifiche (sia hardware, che firmware) necessarie per farlo comunicare su seriale con il VR Stamp™. Si è sfruttato il microcontrollore interno, MSP430 (il tipico processore utilizzato nei dispositivi dell'ambito domotico di Gewiss), ed il suo firmware, opportunamente modificato grazie all'apposito ambiente di sviluppo integrato IAR Embedded Workbench, per far trasmettere i comandi vocali interpretati dalla scheda della Sensory™ su bus, in modo che potessero essere interpretabili dagli attuatori.

Grazie all'apposita debug-interface (illustrata in figura 8.8) è possiible connette il calcolatore al micro, in modo da scaricare il codice compilato nella memoria del dispositivo.

Figura 8.8: Interfaccia di comunicazione tra MSP430 e software per la compilazione del codice (IAR)

Lo schema a blocchi funzionale dell'impianto realizzato risulta essere il seguente

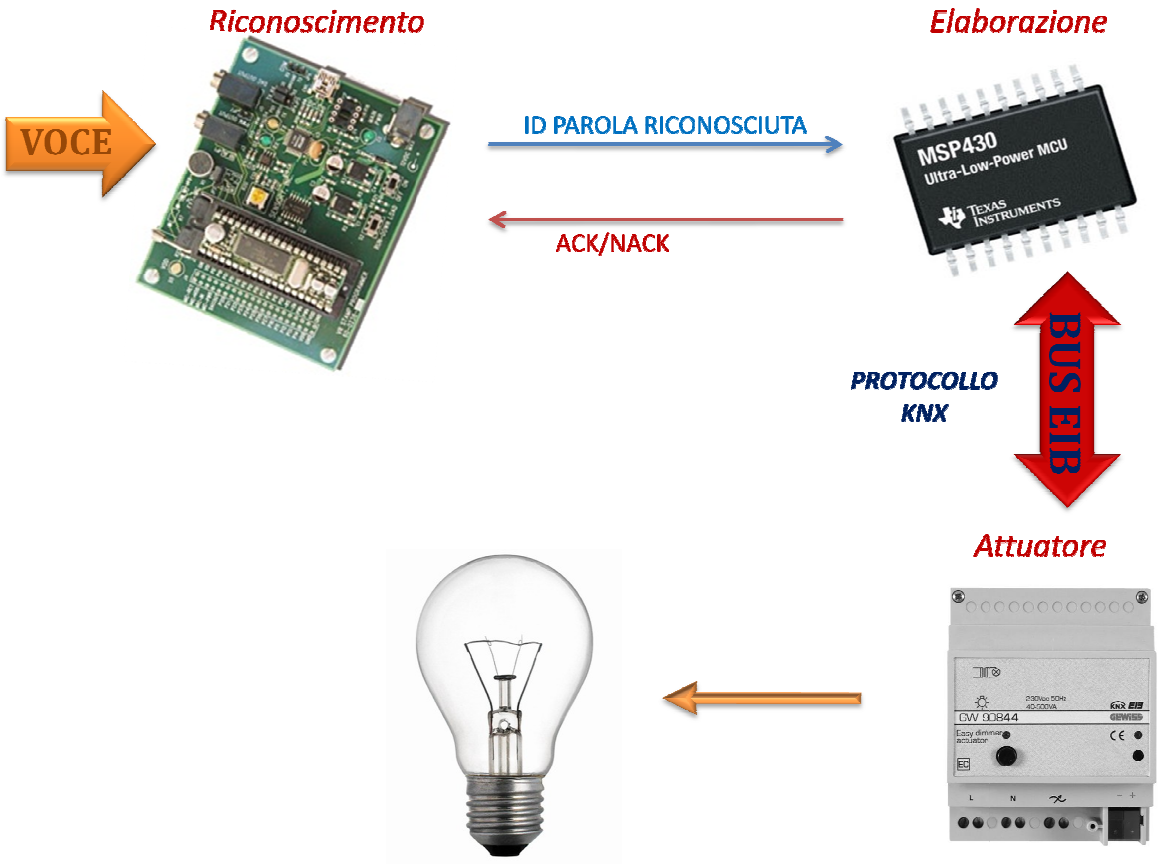


Figura 8.9: Schema a blocchi del funzionamento del progetto

Inizialmente, si era scelta come tecnica di riconoscimento quella indipendente dal particolare utente, in modo da rendere l'applicazione già pronta all'uso, con i comandi memorizzati a livello software direttamente dal produttore.

Le prove effettuate con questa configurazione hanno però dimostrato una forte dipendenza della qualità del riconoscimento dall'ambiente in cui questo avveniva: in condizioni di silenzio o di rumori molto deboli, il sistema presentava un'elevata efficienza, con un ridottissimo numero di falsi riconoscimenti. Tuttavia, non appena il rumore di fondo aumenta, si verifica una radicale diminuzione della capacità di identificazione: il numero di mancati riconoscimenti aumenta in modo drastico, rendendo praticamente inutilizzabile l'applicazione con questa configurazione in un ambiente domotico, quindi domestico, in cui suoni che fungano da disturbi ci sono in ogni istante (porte che sbattono, aspirapolvere, televisione, radio, caldaia…).

Inoltre, in questo modo è praticamente impossibile realizzare un dispositivo che abbia un accesso limitato, ovvero che possa essere gestito e comandato da un limitato numero di persone (ad esempio, i proprietari della casa), oppure che identifichi le parole pronunciate solamente in ben determinati casi (i faretti si devono accendere solamente quando l'utente vuole compiere questa operazione, non ogni volta che chiunque sia presente all'interno della casa, e a portata del microfono, pronunci la parola "luce" o "faretti").

I termini del vocabolario devono essere pronunciati esattamente: qualsiasi esitazione o modifica della pronuncia (si pensi al semplice raffreddore o mal di gola) può compromettere la capacità di funzionamento del sistema.

Quindi, benché il metodo SI garantisca la memorizzazione di vocaboli (e, di conseguenza, modelli di addestramento a cui si farà riferimento in ogni operazione di recongition) non modificati dal rumore, essendo stati realizzati a livello software con gli appositi tool, non è adatto per applicazioni di ambito domotico.

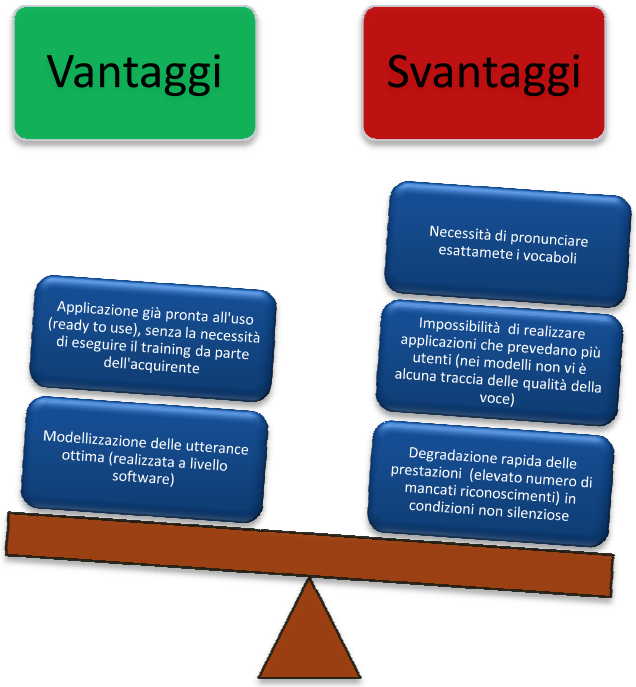


Figura 8.10: Vantaggi e svantaggi dell'utilizzo della tecnica speaker independent

Essendosi dimostrato la modalità speaker independent inadeguata alle finalità d'impiego dell'applicazione, si è modificato il codice in modo da implementare una tecnologia SD (lo speaker

verification e le altre modalità precedentemente descritte non risultano funzionali per l'obiettivo di questo progetto). La differenza sostanziale con il caso precedente è che ora si ha necessità di eseguire una preventiva operazione di training del dispositivo, per poter creare e memorizzare i modelli necessari per implementare il riconoscimento: non si avrà più, quindi, un sistema ready to use come in modo SI. Questa limitazione è però compensata dal fatto con la tecnologia speaker dependent è possibile limitare l'utilizzo dell'applicazione ad un gruppo di persone, trattandosi di un metodo in grado di discriminare la voce dei vari utenti.

Inoltre, i vari test eseguiti hanno consentito di verificare che, con un'opportuna scelta del livello di confidenza delle funzioni della FluentChip™, si ottiene un buon compromesso tra falsi riconoscimenti e mancati riconoscimenti anche in condizioni di ambiente rumoroso. Soglie troppo alte garantiscono una buona robustezza del sistema nei confronti dei disturbi, ma limitano il numero delle parole identificate. Viceversa, livelli di confidenza bassi assicurano che tutti i vocaboli vengano riconosciuti, ma può causare la rilevazione di suoni indesiderati che vengono erroneamente scambiati per template (quindi si può verificare l'accensione della luce quando non desiderato).

Come già accennato in precedenza nel paragrafo 6.1, è buona norma evitare parole troppo brevi (cioè scarse di contenuto fonetico) ed eseguire l'operazione di addestramento in presenza di basso rumore di fondo.

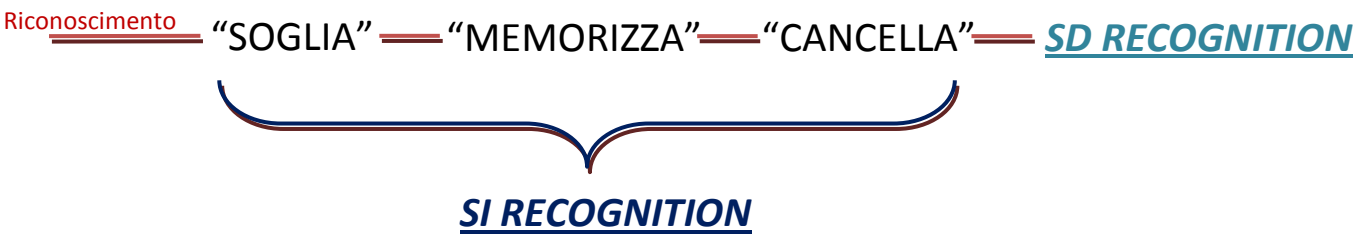
Figura 8.11: Vantaggi e svantaggi dell'utilizzo della tecnica speaker independent

Nonostante la tecnologia SD garantisca delle buone prestazioni per l'impiego finale dell'applicazione, non permetteva di soddisfare completamente le esigenze che si erano prefissate: infatti, oltre ai vocaboli si rendono necessarie delle "voci di menù" per effettuare delle operazioni di gestione del sistema, come ad esempio l'addestramento, l'eliminazione di un vocabolario, la regolazione della soglia e altro ancora. Per realizzarle si potrebbero prevedere dei pulsanti, ad ognuno dei quali (o ad ogni pressione di uno stesso pulsante) è possibile associare una specifica funzione, ma questo renderebbe praticamente inutile il lavoro svolto fino ad ora: progettare un dispositivo in grado di riconoscere dei comandi vocali, ma poi gestirlo ancora manualmente, è una contraddizione e un prodotto con praticamente nessuna possibilità di mercato.

È quindi evidente come anche le voci del menù debbano essere implementate in modo vocale: avendo deciso di adottare una tecnica speaker dependent per quanto riguarda i comandi, la prima idea è stata quella di adottare questa modalità anche per la gestione del sistema. Ben presto, è emerso un limite evidente: trattandosi di riconoscimento dipendente dal parlatore, si sarebbe dovuta prevedere una fase di addestramento anche per il menù, ma questo sarebbe il male minore. Infatti, la problematica più grossa è il fatto che l'inizio dell'operazione di training deve essere stabilita ancora una volta a mano: l'apparecchiatura, con le impostazioni iniziali, non ha alcuna template memorizzato, per cui è necessario un pulsante (o un altro input meccanico) per avviare l'addestramento e salvare comandi e voci del menù.

Inoltre, fatto questo, la gestione del dispositivo sarebbe possibile solo se a parlare è un utente che ha già addestrato il sistema con la sua voce. Questa tecnica, con le voci del menù dipendenti dal parlatore, non è quella adeguata.

L'ultima opzione disponibile, che è quella adottata nel progetto, prevede due modalità di utilizzo dell'applicazione: SI per le voci del menù, che vengono preventivamente salvate in memoria a livello software, grazie alle quali è possibile gestire il sistema, e SD per i comandi. La struttura del riconoscimento vocale implementato è più chiara se illustrata in maniera schematica come di seguito



In conclusione, è stata adottata la tecnica SI/SD che prevede l'uso in contemporanea delle due modalità di riconoscimento: questo consente di gestire il dispositivo solamente con la voce (handless), avendo la possibilità di modificare i livelli di confidenza, di eliminare un vocabolario già memorizzato e di effettuare l'addestramento del sistema.

In particolare, pronunciando la parola "soglia", si richiama una funzione apposita che permette di variare la soglia di riconoscimento in modo ciclico: il valore impostato di default è 2 (in modo da ottenere un compromesso tra falsi riconoscimenti e mancati riconoscimenti), ma può assumere valori che vanno da 0 (privilegio le utterance identificate con un'elevata confidenza) a 4 (vengono accettate anche le parole riconosciute con una confidenza medio/bassa). Essendo ciclica questa modifica, ogni volta che il dispositivo sente il termine "soglia", aumenta questo dato di una unità, tranne nel caso in cui abbia già valore 4 e ritorna a 0.

Pronunciando "memorizza", invece, si procede alla fase di addestramento; prima di procedere con la descrizione di questa operazione è utile notare come, contrariamente a quanto si possa pensare, il fatto che i comandi siano speaker dependent non significa che ogni utente possa definire delle funzioni diverse: infatti, ad ogni vocabolo del dizionario è associata una ben precisa operazione (ad esempio, al primo template l'accensione della luce, al secondo lo spegnimento della stessa, al terzo l'apertura della porta e così dicendo) che è stabilita preventivamente (anche perché, in base a queste impostazioni, deve essere configurato il bus e gli oggetti di comunicazione dei singoli device). L'utilizzatore può solo associare una parola differente a quella funzione.

In poche parole, al primo vocabolo corrisponde una certa funzione, al secondo un'altra, al terzo un'operazione differente dalle altre due, indipendentemente da quale parola sia necessario riconoscere per attivarle.

Una volta avviata la fase di training, il dispositivo prevede una voce guida che richiede di pronunciare una ben precisa parola; è anche presente un'indicazione luminosa (dei led), che indicano se l'applicazione è ancora in attesa del vocabolo (led giallo acceso), oppure se esso è stato riconosciuto correttamente (led verde e conferma della voce guida) o meno (led rosso).

In particolare, il progetto è stato strutturato in modo che sia possibile aver memorizzati contemporaneamente tre utenti, ognuno dei quali può salvare 14 comandi più il proprio "userID": infatti, per evitare che il dispositivo posso riconoscere erroneamente dei disturbi, confondendoli con i template memorizzati, prima che i comandi vengano attuati è necessario che l'utente si identifichi. Una volta compiuta questa operazione, ha a disposizione 10 secondi (questo tempo può essere modificato o anche tolto a seconda delle esigenze) per pronunciare la particolare azione da eseguire. Scaduto questo periodo, è nuovamente necessario autenticarsi.

Nel caso del progetto dimostrativo, i vocaboli sono memorizzati come segue:

NOME UTENTE:

ANDREA

SD TEMPLATES:

BAGNO

BAGNO SPENTO

CAMERA

CAMERA SPENTA

CUCINA

CUCINA SPENTA

PIÙ LUCE

MENO LUCE

STOP

SCENARIO (ENTRATA)

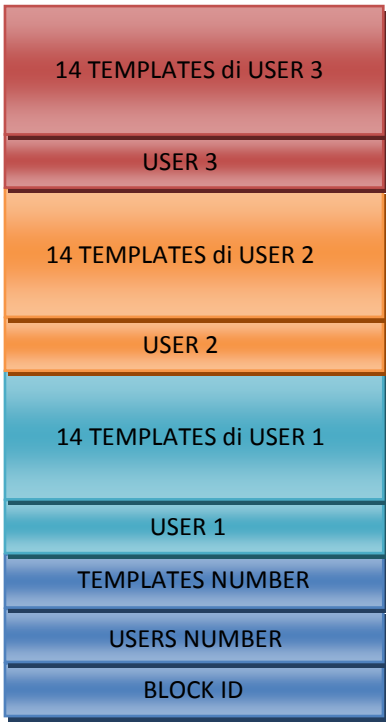
ENTRATA

ALZA TAPPARELLA

ABBASSA TAPPARELLA

FERMA TAPPARELLA

Inoltre, a livello fisico, la memoria del dispositivo è così strutturata



Infine, riconoscendo la parola "elimina", il sistema chiede quale utente si vuole eliminare: il parlatore dovrà quindi pronunciare lo userID corrispondente, e, se identificato correttamente, ripeterlo un'altra volta per confermare l'eliminazione dalla memoria.
Riassumendo:

Fase di Training:

1. *Pronunciare "Memorizza"*
2. *La voce guida chiederà di pronunciare il nome dell'utente, necessario in seguito per identificarlo (se entro 5 secondi non viene rilevato alcun suono, la demo ritorna nello stato di "ascolto", uscendo da quello di training)*
3. *Una volta memorizzato l'ID, verranno chiesti in ordine tutti i 14 vocaboli da salvare (se si verificano errori durante il training di una parola, come ad esempio eccessivo rumore di fondo, questa verrà chiesta nuovamente)*

Riconoscimento:

1. *Identificarsi pronunciando il proprio nome utente*
2. *Una volta riconosciuto (l'avvenuta identificazione è segnalata dalla voce guida e dall'accensione del led verde) pronunciare uno dei 14 vocaboli memorizzati durante la fase precedente*
3. *Attuazione del comando corrispondente alla parola pronunciata*

***N.B.** : l'identificazione dell'utente ha una validità di 10 secondi, scaduti i quali si rende necessario identificarsi nuovamente*

Eliminazione:

1. *Pronunciare "Elimina"*
2. *La voce guida chiederà il nome utente da eliminare (unitamente ai relativi templates)*
3. *Pronunciare il nome utente da cancellare*
4. *Confermare l'operazione ribadendo il nome*

Come si può notare, nell'applicazione sviluppata non è stata adottata la tecnica di Audio Wakeup (l'unico metodo di powerdown implementabile, volendo mantenere a tutti gli effetti il dispositivo handless) per un semplice motivo: si è verificato che, impostando come evento di risveglio un suono di forte entità, basta un rumore leggermente più forte del normale (porta che sbatte o oggetti che cadono per terra) per riattivare la demo board. È ora spontaneo chiedersi perché non utilizzare il battito di mani come risveglio, ma si è evitato per rimanere sempre nell'ottica di un dispositivo che possa essere comandato interamente a voce.

In base alle considerazioni dei capitoli precedenti, a seguito delle varie prove effettuate in fase di progetto, e dopo una valutazione di tutte le componenti necessarie per il corretto funzionamento dell'applicazione dimostrativa, è possibile affermare che ***la realizzazione*** della stessa ***come dispositivo stand-alone ad un modulo*** (e quindi integrabile in un impianto tradizionale) ***è fattibile***: infatti, i componenti elettronici richiesti sono per lo più elementi passivi (resistenze, condensatori, diodi) o piccoli dispositivi attivi (quali regolatori, buffer, amplificatori), occupando nel complesso una superficie tale da poter garantirne l'integrazione in un modulo DIN.

Si noti come il disegno elettronico e lo sbroglio delle piste di un eventuale dispositivo stand-alone non è stato fatto nel corso di questo progetto, richiedendo tempi e costi che non erano previsti e che spettano alla fase successiva.

Ovviamente, particolare attenzione dovrà essere prestata al posizionamento del microfono nella plastica, ricordando i consigli riportati nel paragrafo 5.2 per quanto riguarda la sede dello stesso, e cercando di non installarlo vicino ad altri dispositivi che possano creare dei disturbi significativi (grossi relè, quantità elevata di cavi nei quali scorre la corrente di rete,…). Inoltre, dovrà essere studiata più approfonditamente e progettata con maggiore attenzione la parte di processing a valle del microfono, che determina la qualità con cui il suono giunge al microcontrollore, e, quindi, la possibilità di discriminare quanto "sentito" dal sistema.

A livello di costo, ***l'applicazione realizzata è sicuramente low-cost***: il componente che maggiormente influisce sul prezzo complessivo (limitatamente all'elettronica) è senza dubbio il microcontrollore della Sensory™, ossia l'RSC-4128. Il suo prezzo, acquistando un singolo pezzo come utente privato, è di circa 20 euro; ovviamente, qualora le quantità aumentino e ad acquistarlo sia un'azienda, il costo dello stesso diminuisce drasticamente: per uno stock di 500 pezzi, il prezzo unitario non supera i 5 euro, garantendo una spesa complessiva per la parte elettronica che si attesta attorno ai 10/15 euro.

Figura 8.12: Vista frontale del pannello dimostrativo; si notano gli attuatori nella parte inferiore, mentre in quella superiore è presente la piantina e i led per la simulazione, nonché la lista dei comandi; le due "scatolette" sono una l'altoparlante, e l'altro il microfono con i led

Figura 8.13: Dispositivi Gewiss impiegati nella realizzazione del pannello, da sinistra verso destra: attuatore 4 canali (GW90836), dimmer (GW90844), alimentatore bus KNX (GW90702), interfaccia usb (GW90706A), attuatore tapparella (GW12767)

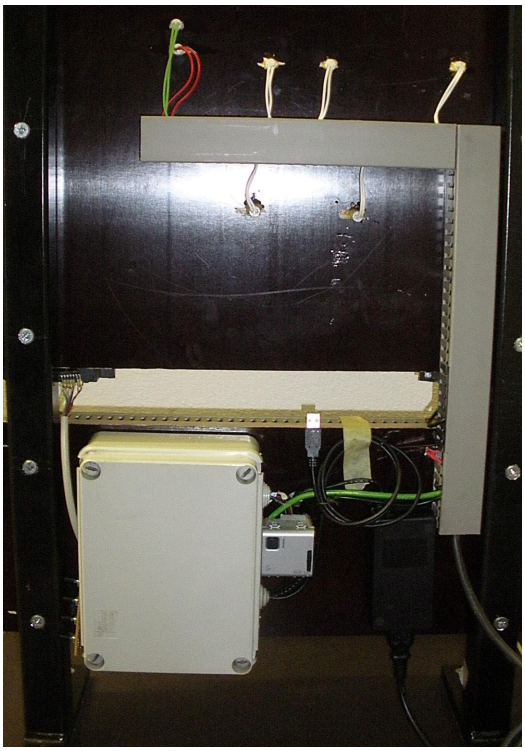


Figura 8.14: Vista posteriore del pannello; si nota il ricevitore RF (GW14776) al quale è collegato il cavo del bus (verde)

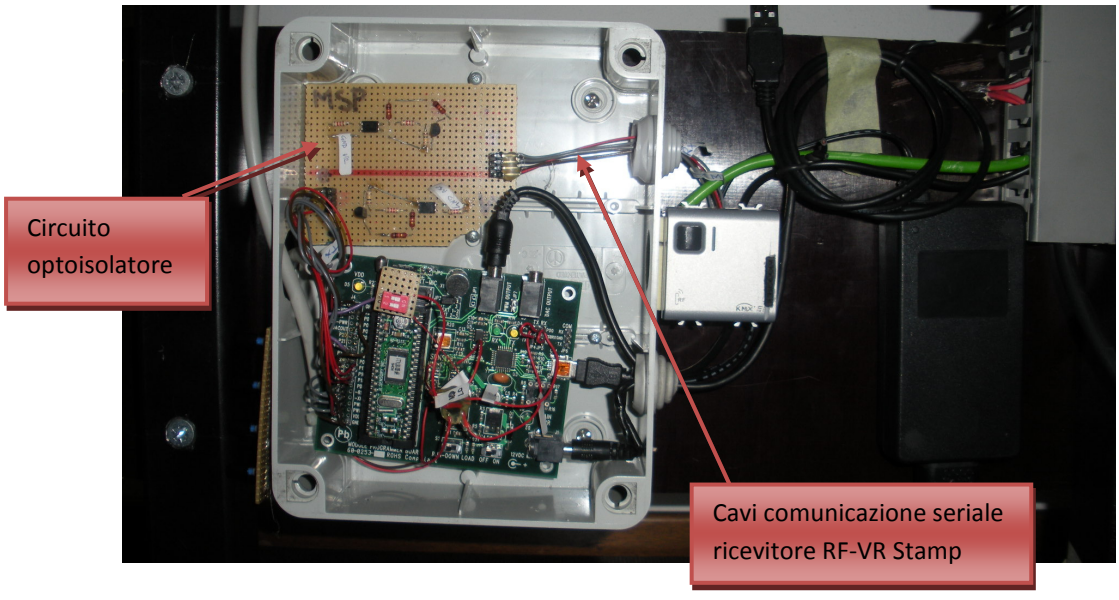


Figura 8.15: Vista posteriore con scatola di derivazione aperta

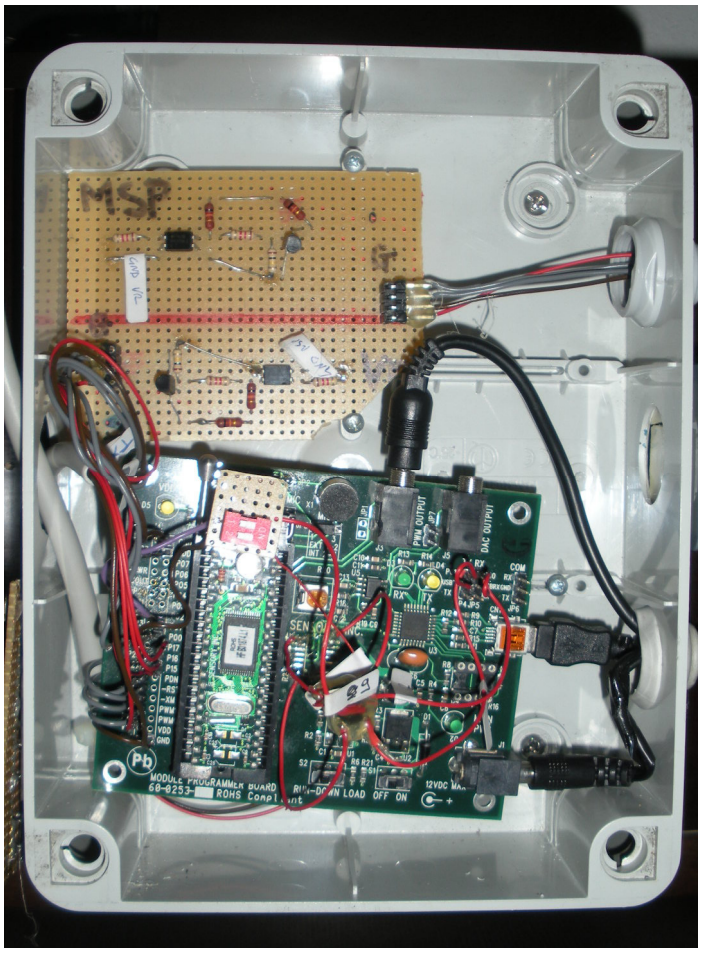


Figura 8.16: VR Stamp™

APPENDICE A

BREVE RIEPILOGO DEI CONCETTI PRINCIPALI DI CAMPIONAMENTO E QUANTIZZAZIONE

Prima di procedere con l'elaborazione digitale, è essenziale discretizzare il segnale in ingresso. L'onda acustica, trattandosi di un segnale reale (naturale) rappresentante una grandezza fisica (nel caso trattato in questo elaborato sarà la pressione), è ovviamente un segnale analogico: esso può assumere infiniti valori, essendo continuo sia nei tempi, che nelle ampiezze. Come è ormai ben noto, soprattutto ai nostri giorni in cui vi è una vastissima diffusione di apparecchiature elettroniche, trattare un segnale analogico non è affatto semplice ed è molto sconveniente in quanto comporta tempi di elaborazione e calcolo elevatissimi (se non infiniti, nel qual caso diventerebbe impossibile da analizzare). Si rende quindi necessaria la *digitalizzazione del suono*, ovvero la sua *discretizzazione nei tempi e nelle ampiezze*: tale operazione è resa possibile dal **campionamento** e dalla **quantizzazione**.

La prima operazione, il sampling, consiste nel "fotografare" con una certa frequenza il valore istantaneo del segnale analogico: il risultato sarà una sequenza di impulsi la cui ampiezza coinciderà con quella del segnale originario.

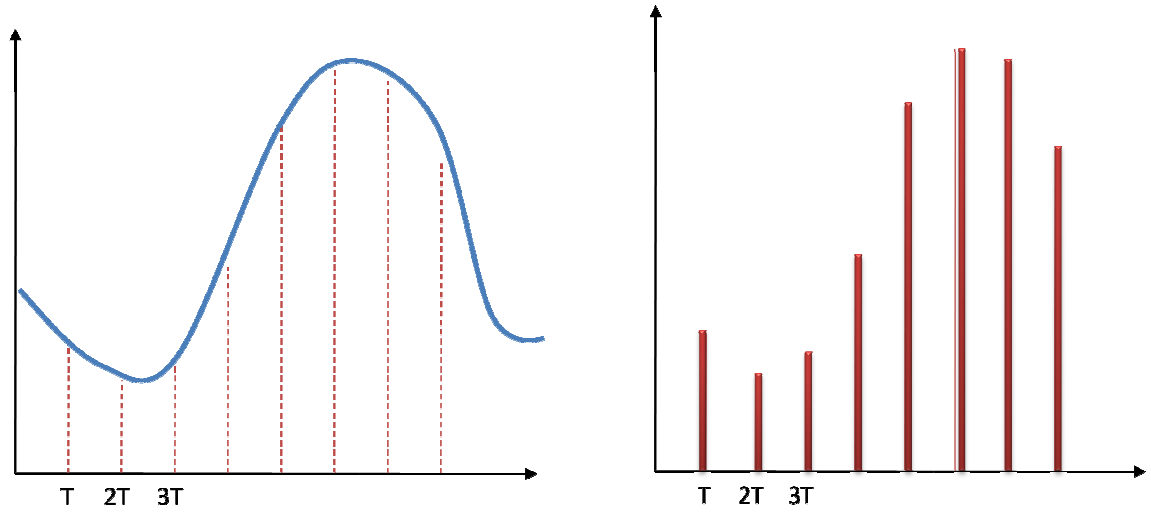


Figura A.1: Campionamento di un segnale analogico

Tuttavia, la cadenza con cui catturare i valori non può essere scelta in modo arbitrario, ma deve rispettare il ben noto ***criterio di Nyquist***, una delle pietre miliari su cui si poggia tutta la teoria dei segnali (branca che generalmente prende il nome di Digital Signal Processing, DSP): questo teorema mette in relazione il contenuto del segnale campionato con la frequenza di campionamento e le componenti spettrali del segnale analogico originario, definendo la *minima frequenza necessaria per discretizzare temporalmente il segnale senza che si abbia perdita di informazioni*, e quindi *in modo tale che il segnale originario possa essere ricostruito a partire da quello*

campionato. In particolare, il teorema afferma che la minima frequenza con cui campionare un generico segnale analogico a banda finita, deve essere almeno pari al doppio della sua frequenza massima (che, per un segnale in banda base, ovvero non soggetto a modulazione, coincide con la banda dello stesso), in modo da evitare qualsiasi forma di interferenza (aliasing) o perdita di informazione. In formule:

$$f_c|_{min} = 2 \cdot f_{max} \tag{A.1}$$

Dove con $f_c|_{min}$ si indicata la minima frequenza di campionamento, mentre f_{max} rappresenta la massima frequenza contenuta nello spettro del segnale originale.

Questo criterio ha validità generale, per un qualsiasi segnale reale. Nel caso di interesse dell'elaborato, trattandosi di un sistema di riconoscimento vocale, il segnale originale sarà rappresentato dalla voce: per essa è stato definito uno standard, derivante dalla telefonia. Il canale telefonico utilizza una banda di circa 3.1 KHz (infatti, si assume che lo spettro della voce abbia frequenze comprese tra 300 e 3100 Hz), dunque, per il teorema di Nyquist, si deduce che è necessaria una frequenza di campionamento pari ad almeno 6.2 KHz ; nella pratica, tuttavia, non si considera la banda netta, ma (a causa della non idealità dei dispositivi, in particolar modo dei filtri) quella cosiddetta lorda, pari a 4 KHz. In questo modo, lo standard prevede che la voce sia rappresentata digitalmente con 8000 campioni al secondo.

Affinchè una grandezza sia trasmissibile e codificabile con un numero finito di bit, ossia possa essere rappresentata in forma numerica, è necessario che essa possa assumere solo un numero limitato di valori: l'operazione necessaria per giungere ad un segnale di questo tipo è la ***quantizzazione***, ovvero il *processo non lineare di discretizzazione dell'intensità della grandezza fisica d'ingresso, in modo che possa essere rappresentata con un numero finito L di livelli*.

Per ottenere ciò, vengono definiti un valore massimo e un minimo entro i quali il segnale numerico (a valle della quantizzazione) può oscillare, permettendo di definire le regioni di decisione e la dinamica del quantizzatore stesso: in questo modo il valore analogico della grandezza originaria (in corrispondenza dell'istante di campionamento) verrà ricondotto al più prossimo dei valori discreti preventivamente definiti. Qualora l'input superi i limiti entro cui è limitata la dinamica del blocco di quantizzazione, si ha una condizione di saturazione, in cui l'output assume il valore massimo (o minimo).

Si tratta quindi di un *processo deterministico*, poiché noto il valore in ingresso si può stabilire quale sia il rispettivo valore in uscita, *senza memoria*, in quanto l'output dipende esclusivamente dall'input attuale, e *non invertibile*, a causa della non linearità.

La fase di discretizzazione è seguita da quella di assegnamento dei livelli discreti al codice corrispondente; tale procedura viene chiamata codifica.

Trattandosi di un sistema distribuito, e non di una cascata di blocchi uniformi, non è possibile determinare in modo esatto il numero di poli necessari per rappresentare in modo equivalente il sistema reale, ma è necessario fare delle approssimazioni. Considerando che, in media, il tratto vocale ha una lunghezza di circa20 cm e che la velocità del suono è 330 m/s, si deduce che il ritardo complessivo di propagazione al suo interno vale $0.2/330 \approx 6\,ms$. Inoltre, si osserva come il fenomeno fonatorio abbia uno spettro in cui la quasi totalità dell'energia è concentrata nella banda compreso tra 0 e 4000 Hz, per cui si può adottare come frequenza di campionamento un valore almeno pari a 8 KHz, al quale corrisponde un intervallo di campionamento inferiore (o, al più, pari) a 125 μs. Il ritardo unitario di ciascuna cella risulta quindi essere inferiore a 62.5 μs ed il numero minimo delle stesse superiore a 10; tipicamente, si sceglie di approssimare il tratto vocale con un filtro IIR a 10 celle, ovvero con 10 poli.

APPENDICE B

CODICI MATLAB

Figura 1.3 – Forma d'onda della parola "Rex"

```
% La funzione wavread estrae le informazioni dal segnale audio (in questo
% caso nel formato ".wav"; in particolare, è possibile conoscerne i valori
% dei campioni (normalizzati, quindi tra -1 e +1), la frequenza di
% campionamento del segnale e il numero di bit per campione
[s,fS,b]=wavread('rex.wav');

% s --> vettore dei campioni del segnale
% fS --> frequenza di campionamento
% b --> numero di bit per campione

% Calcolo la durata del segnale (moltiplico il periodo di campionamento per
% il numero di campiono del segnale
t = (1/fS)*[1:max(size(s))];

% Disegno della forma d'onda
plot(t,s);

xlabel('tempo [s]');
ylabel('ampiezza normalizzata');
title('WAVEFORM PAROLA "REX"');
```

Figura 3.2 - Finestre

```
% Finestre di 100 campioni
window_rec = rectwin(100);
window_tri = triang(100);
window_ham = hamming(100);

% Rappresentazione grafica
plot(window_rec);
hold on;
plot(window_tri,':');
hold on;
plot(window_ham,'--');

xlabel('campioni');
ylabel('ampiezza normalizzata');
title('FINESTRE');
ylim([0 1.2]);
legend('Finestra Rettangolare','Finestra Triangolare','Finestra di Hamming');
```

Figura 3.3 – Short Time Average Energy e Magnitude

```
% Lettura del segnale audio
[s,fS,b]=wavread('rex.wav');

Nframe=100;        % Numero di campioni per frame
Ns=max(size(s));   % Numero di campioni del segnale

% Calocolo della Short Time Average Energy
for n=1:Ns;
   E(n,1)=sum(s(max(1,n-Nframe+1):n).*s(max(1,n-Nframe+1):n))/Nframe;
end

% Calocolo della Short Time Average Magnitude
for n=1:Ns;
   M(n,1)=sum(abs(s(max(1,n-Nframe+1):n)))/Nframe;
end

% Rappresentazione grafica
t = (1/fS)*[1:max(size(s))];
subplot(3,1,1);
plot(t,s);
xlabel('tempo [s]');
ylabel('s');
title('WAVEFORM, SHORT TIME AVERAGE ENERGY e SHORT TIME AVERAGE MAGNITUDE');

E=E/max(E);        % normalizza E(t)
t = (1/fS)*[1:max(size(E))];
subplot(3,1,2);
plot(t,E);
xlabel('tempo [s]');
ylabel('E(t)');

M=M/max(M); % normalizza M(t)
t = (1/fS)*[1:max(size(M))];
subplot(3,1,3);
plot(t,M);
xlabel('tempo [s]');
ylabel('M(t)');
```

Figura 3.4 – Confronto STD variando la durata della finestra

```
% Lettura del segnale audio
[s,fS,b]=wavread('rex.wav');

frame = 0.0001;
overlap = 50;

Ns = max(size(s));                  % numero di campioni di s
Nframe = floor(fS * frame);         % numero di campioni per frame
Ndiff = floor(Nframe * (1 - overlap/100));   % numero di campioni tra frames
L = floor((Ns-Nframe)/Ndiff);       % numero di finestre
window_ham = hamming(Nframe);

for n=1:L
inizio = (n-1) * Ndiff + 1;         % inizio della finestra
tempi(n,1) = n* Ndiff/fS;
Q(n,1) = sum(s(inizio:inizio+Nframe-1,1) .* window_ham);
end
% Calocolo della Short Time Average Energy
for n=1:L;
   E(n,1)=sum(Q(max(1,n-Nframe+1):n).*Q(max(1,n-Nframe+1):n))/Nframe;
end

E=E/max(E);        % normalizza E(t)
t = (1/fS)*[1:max(size(E))];
subplot(3,1,1);
plot(t,E,'-');
xlabel('tempo [s]');
ylabel('E(t)');

frame = 0.0002;
overlap = 50;

Ns = max(size(s));                  % numero di campioni di s
Nframe = floor(fS * frame);         % numero di campioni per frame
Ndiff = floor(Nframe * (1 - overlap/100));   % numero di campioni tra frames
L = floor((Ns-Nframe)/Ndiff);       % numero di finestre
window_ham = hamming(Nframe);

for n=1:L
inizio = (n-1) * Ndiff + 1;         % inizio della finestra
tempi(n,1) = n* Ndiff/fS;
Q(n,1) = sum(s(inizio:inizio+Nframe-1,1) .* window_ham);
end

% Calocolo della Short Time Average Energy
```

```
for n=1:L;
   E(n,1)=sum(Q(max(1,n-Nframe+1):n).*Q(max(1,n-Nframe+1):n))/Nframe;
end

E=E/max(E);        % normalizza E(t)
t = (1/fS)*[1:max(size(E))];
subplot(3,1,2);
plot(t,E,'-');
xlabel('tempo [s]');
ylabel('E(t)');

frame = 0.0005;
overlap = 50;

Ns = max(size(s));                    % numero di campioni di s
Nframe = floor(fS * frame);           % numero di campioni per frame
Ndiff = floor(Nframe * (1 - overlap/100));   % numero di campioni tra frames
L = floor((Ns-Nframe)/Ndiff);         % numero di finestre
window_ham = hamming(Nframe);

for n=1:L
inizio = (n-1) * Ndiff + 1;           % inizio della finestra
tempi(n,1) = n* Ndiff/fS;
Q(n,1) = sum(s(inizio:inizio+Nframe-1,1) .* window_ham);
end

% Calocolo della Short Time Average Energy
for n=1:L;
   E(n,1)=sum(Q(max(1,n-Nframe+1):n).*Q(max(1,n-Nframe+1):n))/Nframe;
end

E=E/max(E);        % normalizza E(t)
t = (1/fS)*[1:max(size(E))];
subplot(3,1,3);
plot(t,E,'-');
xlabel('tempo [s]');
ylabel('E(t)');
```

Figura 3.4 – Confronto STD variando l'overlap delle finestre

Il codice è lo stesso del precedente, l'unica differenza è che si varia il valore di "overlap" invece di "frame"

Figura 3.6 – Zero Crossing Rate

```
% Lettura del segnale audio
[s,fS,b]=wavread('rex.wav');

Nframe = 100;            % numero di campioni per frame
Ns = max(size(s));       % numero di campioni del segnale

% Calcolo dello Zero Crossing Rate
for n = 1+Nframe:Ns;
   ZCR(n,1) = sum(abs(sign(s(n-Nframe+1:n))-sign(s(n-Nframe:n-1)))/2)/Nframe;
end

% Calcolo degli attraversamenti dello zero ogni millisecondo
ZCR=ZCR*fS/1000;

% disegna Z(t)
t = (1/fS)*[1:max(size(ZCR))];
plot(t,ZCR);
 xlabel('time (s)');
ylabel('ZCR/ms(t)');

% Rappresentazione grafica
t = (1/fS)*[1:max(size(s))];
subplot(2,1,1);
plot(t,s);
xlabel('tempo [s]');
ylabel('s(t)');
%title('WAVEFORM e ZERO CROSSING RATE');

t = (1/fS)*[1:max(size(ZCR))];
subplot(2,1,2);
plot(t,ZCR);
xlabel('tempo [s]');
ylabel('ZCR/ms(t)');
```

Figura 3.7- Complessivio

```
% Lettura del segnale audio
[s,fS,b]=wavread('rex.wav');

Nframe = 100;            % numero di campioni per frame
Ns = max(size(s));       % numero di campioni del segnale

% Calocolo della Short Time Average Energy
```

```
for n=1:Ns;
   E(n,1)=sum(s(max(1,n-Nframe+1):n).*s(max(1,n-Nframe+1):n))/Nframe;
end

% Calocolo della Short Time Average Magnitude
for n=1:Ns;
   M(n,1)=sum(abs(s(max(1,n-Nframe+1):n)))/Nframe;
end

% Calcolo dello Zero Crossing Rate
for n = 1+Nframe:Ns;
   ZCR(n,1) = sum(abs(sign(s(n-Nframe+1:n))-sign(s(n-Nframe:n-1))))/2)/Nframe;
end

% Calcolo degli attraversamenti dello zero ogni millisecondo
ZCR=ZCR*fS/1000;

% Rappresentazione grafica
E=E/max(E);        % normalizza E(t)
t = (1/fS)*[1:max(size(E))];
subplot(3,1,1);
plot(t,E);
xlabel('tempo [s]');
ylabel('E(t)');
%title('CONFRONTO DEI PARAMETRI');

M=M/max(M); % normalizza M(t)
t = (1/fS)*[1:max(size(M))];
subplot(3,1,2);
plot(t,M);
xlabel('tempo [s]');
ylabel('M(t)');

t = (1/fS)*[1:max(size(ZCR))];
subplot(3,1,3);
plot(t,ZCR);
xlabel('tempo [s]');
ylabel('ZCR/ms(t)');
```

BIBLIOGRAFIA

Dispense di Michael Picheny, Ellen Eide, Stanley F. Chen, ricercatori presso IBM T.J. Watson Research Center (Columbia University, Yorktown Heights, NY), 2005

Dispense del corso Speech, Audio Processing and Recognition dei professori Dan Ellis e Mike Mandel del Dipartimento di Ingegneria Elettronica della Columbia University, 2009

Dispense del Prof. Bryan Pellom del Dipartimento di Computer Science Center for Spoken Language Research presso la University of Colorado

Dispense del corso Speech Recognition, Synthesis and Dialogue del professor Dan Jurafsky presso la Stanford University, 2007

Capitolo "Sound modeling: signal-based approach" dell'articolo di Giovanni de Poli e Federico Avanzini, 2009

Capitolo "Elementi di Acustica e Psicoacustica" di Carlo Drioli e Nicola Orio

Articolo "An Introduction to Hybrid HMM/Connectionist Continuos Speech Recognition" di Nelson Morgan e Hervé Boulard dell'International Computer Science Institute. 2005

Articolo "Automatic Speech Recognition – A Brief History of the Technology Development" di B.H. Juang e Lawrance R.Rabiner del Georgia Institute of Technology, 2004

Articolo "Separation of voiced and unvoiced using zero crossing rate and energy of the speech signal" di Bachu R.G., Kopparthi S., Adapa B., Barkana B.D., 2008

Dispense interne relative al Bus KNX/EIB di Gewiss Professional, 2011

VR Stamp with Serial EEPROM Module Data Sheet di Sensory™, 2006

RSC-4128_Datasheet di Sensory™, 2006

Design Note, Microphone Housing di Sensory™, 2006

FluentChip™ Reference di Sensory™, 2007