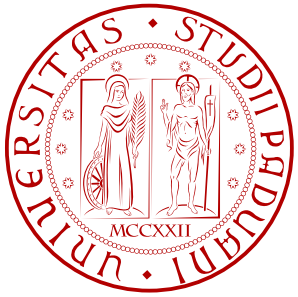


Università degli studi di Padova

Facoltà di Ingegneria

Tesi di Laurea Triennale in

Ingegneria dell'Automazione

Generazione del codice per una scheda dSPACE di Fast Control Prototyping mediante Matlab-Simulink.

RELATORE

Prof. Silverio Bolognani

CORRELATORE

Luca Sgarbossa

LAUREANDO

Matteo De Bernardin

Anno Accademico 2010/2011

Prefazione

Questa tesi, realizzata nel laboratorio di azionamenti elettrici dell'Università di Padova, è il frutto di un lavoro di ricerca nel campo degli Azionamenti Elettrici durato sei mesi.
In questa trattazione verrà descritto come è stato possibile far lavorare l'ambiente Matlab/Simulink in parallelo al sistema prototipale della dSPACE (scheda DS1104) e il relativo software di gestione "ControlDesk", per realizzare dei controlli sui motori a magnete permanente. Tale metodo, proposto dal costruttore, è detto "Real Time Prototyping".

Nel primo capitolo verranno effettuati dei richiami sui motori a magnete permanente (brushless) e sulla struttura di un azionamento per il controllo di questo tipo di motore. Come prima cosa saranno presentate le equazioni caratteristiche del motore a rotore anisotropo e da esse sarà ricavato lo schema generale di un azionamento. In seguito, analizzando lo schema dell'azionamento, si parlerà del controllo di corrente e della modulazione vettoriale per controllare l'inverter di tensione che regola il motore.

Nel secondo capitolo sarà presentata la scheda dSPACE modello DS1104 fornendo una breve panoramica sull'architettura e sul funzionamento della stessa. Per poter utilizzare la scheda saranno fatti dei richiami sul software ControlDesk che permette la gestione del sistema di controllo.

Nel terzo capitolo verrà introdotto il concetto di progettazione Real-Time. In seguito saranno presentati tutti i passi necessari per affacciarsi a questo tipo di progettazione.
Dopo aver mostrato come configurare la piattaforma sarà presentato un semplice esempio realizzativo. Infine verranno analizzati i blocchi fondamentali delle librerie Simulink aggiuntive fornite dalla dSPACE.

Il quarto capitolo è dedicato alla parte sperimentale del lavoro condotto in laboratorio.
Riferendosi agli schemi presentati nel primo capitolo, verrà creato il modello del controllo di corrente di un motore. Una volta descritti tutti i passi realizzativi e verificato il funzionamento offline, si collegherà un carico all'inverter e si passerà alla fase finale di test dove viene generata la modulazione PWM che comanda gli stati dell'inverter e misurare le correnti sinusoidali prodotte.

Indice

Capitolo 1

Controllo di un motore brushless PM

1.1 Richiami sul motore brushless

I motori sincroni a magnete permanente, detti anche brushless sinusoidali, sono impiegati sempre più diffusamente per svariate applicazioni: l'aereonautica, l'automobilismo, l'ambito navale e nei sempre più diffusi veicoli elettrici o ibridi.
Recentemente, grazie alla spinta commerciale legata alla richiesta di prodotti ad elevata efficienza energetica nel settore civile, tali motori sono stati introdotti sul mercato del condizionamento e della refrigerazione, dove possono portare elevati benefici in termini di consumi a favore dei grandi utilizzatori.
Il motore brushless è un motore elettrico che a differenza di un motore a spazzole, non necessita di contatti elettrici striscianti sull'albero motore (da qui il nome brushless, senza spazzole), di conseguenza la commutazione della corrente non avviene più per via meccanica, ma elettronicamente, con i benefici di una minore resistenza meccanica.
Un altro vantaggio presentato dai motori brushless è il tempo atteso di vita: le spazzole sono gli elementi che si degradano più velocemente in un motore. Essendo privi di spazzole, i motori a magneti permanenti garantiscono una maggior durata e di conseguenza richiedono minore manutenzione, inoltre l'assenza delle spazzole elimina anche la principale fonte di rumore elettromagnetico presente nei motori con commutazione meccanica (ad esempio il DC-MOTOR).
Per contro, uno svantaggio di questo tipo di motori sta nel costo. Infatti il controllo del motore deve essere effettuato elettronicamente da un controllore che viene fornito dal costruttore o da terzi, incidendo sul costo finale; inoltre anche i magneti hanno un prezzo sostenuto perchè sono composti generalmente da materiali pregiati.

Per quanto riguarda l'efficienza, i motori brushless dissipano molto meno calore di un equivalente motore in corrente alternata, in aggiunta il riscaldamento complessivo del motore risulta essere diminuito in quanto, le perdite Joule sono attribuite solamente agli avvolgimenti di statore, visto che quelli di rotore sono sostituiti dai magneti.
La conversione elettromeccanica che essi attuano segue il principio di funzionamento dei sistemi elettrodinamici che si basa sull'iterazione fra conduttori percorsi da corrente e campi magnetici prodotti da altri conduttori o da magneti permanenti; i conduttori dove agiscono le forze risiedono nello statore (parte fissa), mentre i magneti permanenti stanno nel rotore (parte mobile). Il rotore e lo statore sono entrambi a forma di corona cilindrica di materiale ferromagnetico e sono separati da un traferro in aria. I magneti presentano solitamente una permeabilità magnetica differenziale molto simile a quella dell'aria, a seconda della forma del rotore e della loro disposizione si possono ottenere due strutture di rotore:

- strutture isotrope: che caratterizzano i motori brushless SPM (*surface permanent magnet*).

- strutture anisotrope: che caratterizzano i motori brushless IPM (*interior permanent magnet*).

L'avvolgimento dello statore è trifase, dove ognuna delle fasi presenta lo stesso numero e distribuzione dei conduttori e risulta sfasata di $2\pi/3$ dalle altre, ciascuna fase fa capo ad una coppia di morsetti attraverso i quali ci si collega all' alimentazione da una sorgente trifase esterna.
In Fig.1.1 è visualizzata una rappresentazione schematica di un motore sincrono a magneti permanenti a due poli: a sinistra è schematizzata l'induzione al traferro prodotta dal magnete permanente di rotore, mentre nelle figura di destra è schematizzata l'induzione prodotta dall' avvolgimento statorico della sola fase a.

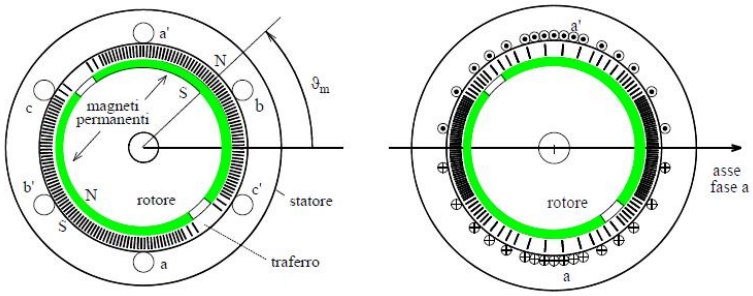


Fig. 1.1: Rappresentazione di un motore SPM a due poli

1.2 Equazioni caratteristiche motore sincrono anisotropo

In seguito verranno ricavate e analizzate le equazioni caratteristiche di un motore sincrono IPM dato che quelle di un motore di tipo SPM possono essere viste come un caso particolare delle precedenti.
Le equazioni del bilancio delle tensioni u_a, u_b, u_c delle rispettive fasi a, b, c sono:

$$\begin{aligned} u_a(t) &= R \cdot i_a(t) + \frac{d\lambda_a(t)}{dt} \\ u_b(t) &= R \cdot i_b(t) + \frac{d\lambda_b(t)}{dt} \\ u_c(t) &= R \cdot i_c(t) + \frac{d\lambda_c(t)}{dt} \end{aligned} \tag{1.1}$$

dove i_a, i_b, i_c sono le correnti che percorrono le tre fasi, R è la resistenza di fase (supposta uguale per tutte e tre le fasi), λ_a, λ_b, λ_c sono i flussi magnetici concatenati con ciascuna fase.
Ipotizzando l'assenza di saturazione dei circuiti magnetici, i flussi concatenati di ciascuna fase possono essere scritti come la somma di un flusso concatenato prodotto dal magnete con il flusso dovuto alle correnti di fase:

$$\begin{aligned} \lambda_a(t) &= \lambda_{a,mg}(t) + \lambda_{a,i}(t) \\ \lambda_b(t) &= \lambda_{b,mg}(t) + \lambda_{b,i}(t) \\ \lambda_c(t) &= \lambda_{c,mg}(t) + \lambda_{c,i}(t) \end{aligned} \tag{1.2}$$

Supponiamo di non inviare alcuna corrente all'avvolgimento statorico. La combinazione di un opportuna sagomatura del magnete e della distribuzione circonferenziale non uniforme dei conduttori di ciascuna fase, consentono di ottenere flussi concatenati dovuti al magnete pressochè sinusoidali. Scegliendo arbitrariamente come coordinata di riferimento l'angolo elettrico ϑ_{me} tra l'asse della fase a e quello del campo prodotto dal magnete permanente (posizione elettromeccanica o meccanica-elettrica) si può scrivere:

$$\begin{aligned} \lambda_{a,mg} &= \Lambda_{mg}\cos(\vartheta_{me}) \\ \lambda_{b,mg} &= \Lambda_{mg}\cos(\vartheta_{me} - 2\pi/3) \\ \lambda_{c,mg} &= \Lambda_{mg}\cos(\vartheta_{me} - 4\pi/3) \end{aligned} \tag{1.3}$$

dove Λ_{mg} rappresenta il massimo flusso concatenato. La terna di grandezze espressa in (1.3) può essere espressa dal vettore spaziale[1]:

$$\lambda^s_{mg} = \Lambda_{mg} e^{j\vartheta_{me}} \tag{1.4}$$

[1]Il vettore spaziale è uno strumento matematico simbolico che consente di studiare il comportamento dei sistemi trifase applicando alle terne di grandezze una trasformazione che permette di semplificare significativamente le equazioni in gioco, in particolare in un piano complesso con asse reale detto asse α e immaginario detto β rappresenta un vettore di modulo e fase variabili nel tempo $\overline{g}(t) = g_\alpha(t) + g_\beta(t)$

dove l'apice s indica che ci stiamo riferendo ad un sistema di riferimento stazionario (solitamente indicato come sistema $\alpha - \beta$) e si nota che il vettore spaziale del flusso concatenato da statore e dovuto al magnete permanente è allineato con l'asse polare, se si fa coincidere l'asse α con l'asse della fase a (Fig.1.2).

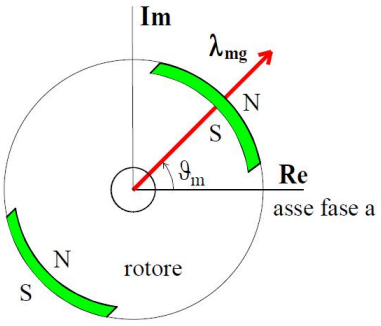


Fig. 1.2: Vettore spaziale del flusso magnete (p=1)

Le equazioni nel sistema di riferimento rotante sincrono[2] con il rotore (d-q) sono:

$$\begin{aligned} u_d &= Ri_d + L_d\frac{di_d}{dt} - \omega_{me}L_q i_q \\ u_q &= Ri_q + L_q\frac{di_q}{dt} - \omega_{me}L_d i_d + \omega_{me}\Lambda_{mg} \end{aligned} \tag{1.5}$$

dove si è definita l'induttanza sincrona diretta L_d percorsa dalla corrente diretta i_d e l'induttanza sincrona in quadratura L_q percorsa dalla corrente in quadratura i_q.
Le espressioni (1.5) permettono di tracciare il bilancio energetico nel sistema di riferimento sincrono in modo da poter ricavare dopo alcuni passaggi matematici qui omessi, l'espressione della coppia meccanica sviluppata dal motore.

$$m = \frac{3}{2}p\Lambda_{mg}i_q + \frac{3}{2}p(L_d - L_q)i_d i_q \tag{1.6}$$

da tale relazione si può notare che il valore delle coppia dipende, per il primo termine, dalla componente della corrente i_q (cioè della componente in quadratura in relazione al flusso concatenato che viene generato dai magneti permanenti). Mentre il secondo termine, detto anche coppia di riluttanza, coinvolge entrambe le correnti.
Una volta definito il carico meccanico con l'equazione:

$$m = m_L + B\omega_m + j\frac{d\omega_m}{dt} \tag{1.7}$$

[2]In un sistema rotante sincrono il vettore spaziale $\overline{g}(t) = g_\alpha(t) + g_\beta(t)$ associato ruota con velocità angolare $\omega_{dq}(t)$ rispetto al sistema stazionario $\alpha - \beta$

applicando le trasformate di Laplace alle equazioni (1.5), (1.6) e (1.7) si può infine tracciare lo schema a blocchi per il motore sincrono a magneti permanenti di tipo anisotropo (Fig.1.3).

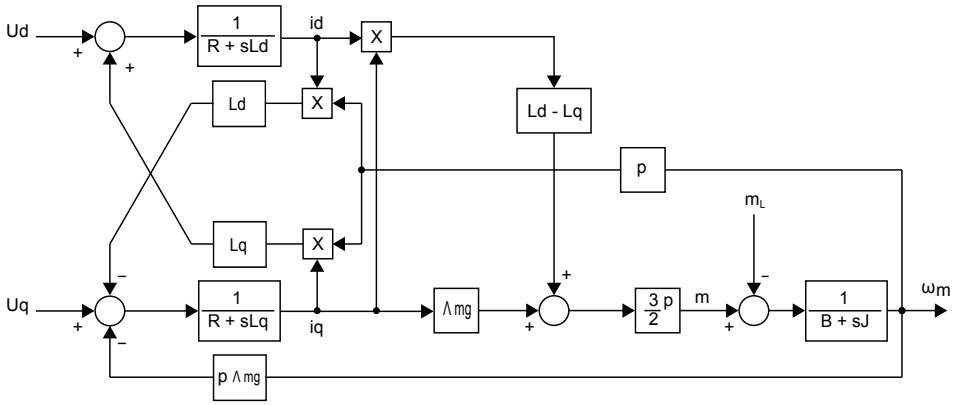


Fig. 1.3: Schema a blocchi motore sincrono a magneti permanenti anisotropo

1.3 Struttura di un azionamento per motore IPM

La struttura base di un azionamento con motore sincrono a magnete permanente si presenta come in Fig.1.4
Dalla sorgente d'alimentazione, che di solito è la rete trifase di distribuzione, si ricava, mediante un raddrizzatore a diodi, la tensione continua (V_{dc}) che alimenta l'invertitore trifase, fornendo un bus di tensione fissa (DC-bus) da cui viene prelevata energia. I comandi per gli interruttori arrivano direttamente dal controllo di coppia dell'azionamento. Il controllo in questo caso un controllo vettoriale di corrente e verrà illustrato in seguito.
Il controllo di coppia necessita, come parametri in ingresso delle tre correnti di fase[3], la posizione assoluta del rotore ϑ_m dalla quale si ricava facilmente la ϑ_{me}.
Sul bus DC, tra il raddrizzatore e l'inverter sono collegati un condensatore C di livellamento e un sistema di dissipazione dell'energia di frenatura. L'intervento del sistema di frenatura si rende necessario quando il motore funziona da freno e la potenza di frenatura viene trasferita, mediante il motore e l'invertitore, dal carico meccanico al bus in continua.
Il riferimento di coppia proviene generalmente da un anello di velocità che elabora l'errore tra la velocità di riferimento e quella misurata sul motore o sul carico.

[3]solitamente la misura delle correnti viene eseguita solo per due fasi, potendosi ricavare la corrente mancante con il principio di Kirchhoff, come somma delle prime due cambiata di segno

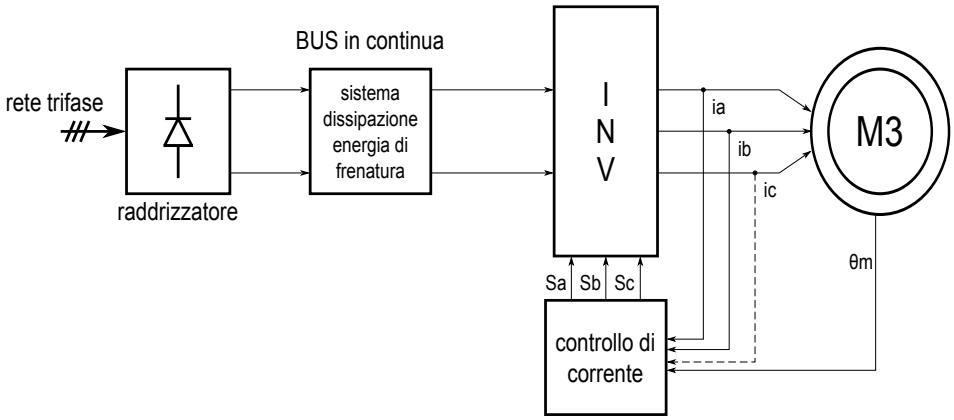


Fig. 1.4: Schema di principio di un azionamento brushless

Controllo in d-q con regolatori PI

Una tecnica realizzativa del controllo di corrente evidenziato nello schema generale di un azionamento di Fig.1.4 è la tecnica del controllo in d-q con regolatori di tipo proporzionale integrale (PI).

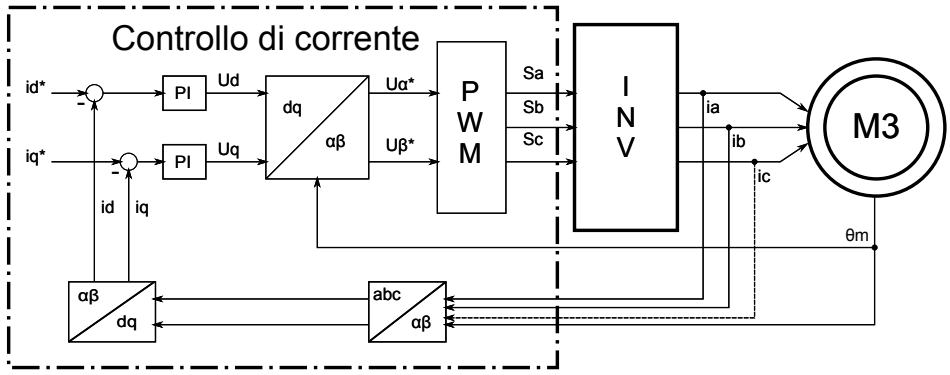


Fig. 1.5: Controllo di corrente

Nel rettangolo tratteggiato in Fig.1.5 si possono visualizzare i blocchi che realizzano questo tipo di controllo di corrente; nel quale si assume un sistema di riferimento d-q in rotazione rispetto allo statore con velocità angolare del tipo $\omega_x = d\vartheta_x/dt$ che nel caso degli azionamenti brushless è $\omega_{me} = d\vartheta_{me}/dt$. In questa tecnica di controllo le uscite dei regolatori sono rispettivamente i riferimenti di tensione per gli assi diretto e in quadratura, che vengono successivamente trasformati dai blocchi di conversione $\mathrm{T}_{dq/\alpha\beta}$ e $\mathrm{T}_{\alpha\beta/abc}$ in segnali logici di controllo.
I vantaggi di questa tecnica di controllo rispetto al controllo in abc o in $\alpha\beta$ sono i seguenti:

- riduzione del numero di variabili da controllare,
- i riferimenti che arrivano ai controllori sono costanti nel tempo.

Questo schema è particolarmente adatto ad una realizzazione digitale, nella quale di solito le funzioni trascendenti vengono memorizzate in apposite tabelle.

1.4 Invertitore trifase di tensione

Lo schema di un invertitore trifase a tensione impressa è illustrato in Fig.1.6 Esso è composto da tre rami (insiemi di due interruttori bidirezionali collegati in serie) alimentati in parallelo da una sorgente in continua (*dc link*), solitamente ottenuta dalla rete di alimentazione tramite un raddrizzatore ed un condensatore di livellamento di appropriata capacità, atto a mantenere pressochè costante la tensione continua (V_{dc}) ai suoi capi. A ciascun ramo fa capo un morsetto del carico trifase, alimentato dal punto centrale tra i due interruttori. Dal punto di vista funzionale, esso è un convertitore DC/AC,

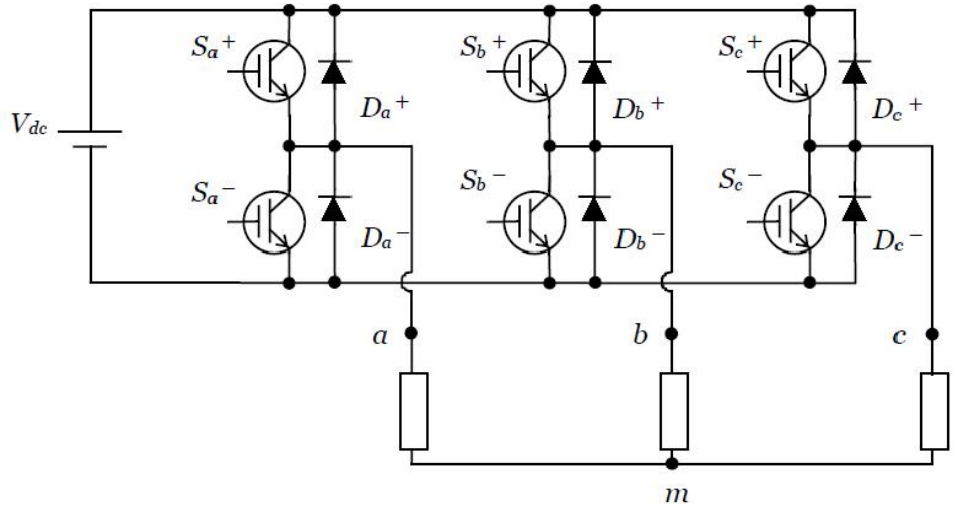


Fig. 1.6: Schema dell'inverter trifase

in grado di trasformare, con opportuno comando degli interruttori di ramo, la grandezza continua in ingresso in grandezze trifase alternate in uscita.

Comandando la chiusura dell'interruttore superiore di un ramo si connette la fase del motore al positivo dell'alimentazione; per evitare il corto circuito della sorgente continua, il comando dei due interruttori dello stesso ramo deve essere di tipo complementare, come illustrato in Fig.1.6
Dall'ultima condizione descritta risultano sei possibili stati assunti dagli interruttori, che possono essere rappresentati con una parola binaria a tre

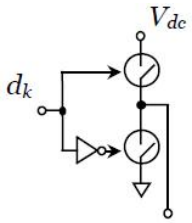


Fig. 1.7: Comando di ramo

bit (uno per fase) utilizzando la convenzione di indicare con "1" la chiusura dell'interruttore superiore di un ramo e con "0" quella dell'interruttore inferiore.

Applicando la definizione di vettore spaziale, si trova che 6 degli 8 stati sono rappresentati da una stella di sei vettori di stato (detti *state vectors*) di ampiezza $\frac{2}{3}Vdc$ e sfasati rispettivamente di $\pi/3$, occupando i vertici di un esagono centrato nell' origine degli assi, come mostra la Fig.1.8

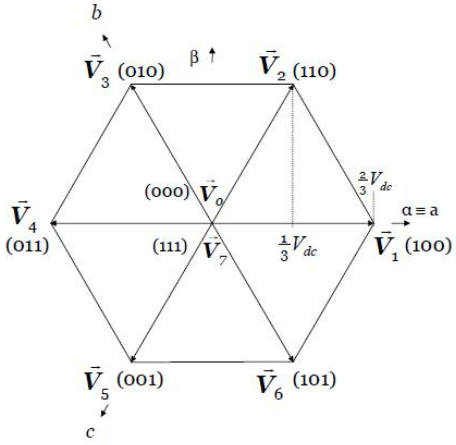


Fig. 1.8: Esagono delle tensioni d'uscita dell'inverter

I due stati 111 e 000, che corrispondono alle configurazioni con tutti gli interruttori inferiori o superiori chiusi, compongono il relativo vettore spaziale nullo, ed occupano l'origine degli assi del piano complesso (*stati nulli*).

La stella dei vettori individua sei regioni o settori, convenzionalmente numerati in senso antiorario; anche i vettori spaziali di stato vengono numerati alla stessa maniera e l'espressione del generico vettore di stato è data da:

$$\mathbf{V}m = \frac{2}{3} \cdot Vdc \cdot e^{(j(m-1)\cdot\frac{\pi}{3})} \tag{1.8}$$

1.5 Tecniche di generazione dei segnali PWM

La soluzione al problema della regolazione dell'ampiezza della tensione in uscita dell'invertitore, assieme ad un miglioramento del contenuto armonico a bassa frequenza, si ottengono distribuendo in modo appropiato le inversioni di tensione di fase dell'invertitore all'interno di ciascun semiperiodo, ovvero modulando opportunamente la durata di ciascun impulso.

Questa tecnica di controllo della tensione di uscita dell'invertitore è comunemente nota come modulazione della larghezza degli impulsi o PWM (*Pulse Width Modulation*).
Mediante le diverse tecniche di PWM si può far in modo che, diversamente da quanto accade per il metodo in onda quadra, le armoniche di ampiezza maggiore abbiano frequenza molto più alta di cinque volte la fondamentale e siano quindi soggette ad una più energica azione di filtraggio da parte dell'induttanza del motore. Una tensione PWM produce pertanto una corrente di fase pressochè sinusoidale con un carico RLC formata dalla componente fondamentale e da oscillazioni di modesta ampiezza sovrapposte ad essa.
La possibilità di variare la componente fondamentale U_1 della tensione d'uscita modificando la larghezza degli impulsi, senza dover variare la tensione continua che lo alimenta, permette di alimentare l'invertitore tramite un raddrizzatore a diodi, evitando quindi ulteriori conversioni in cascata.
I principali vantaggi sono una semplificazione ed un minor ritardo della risposta della sezione di potenza ad un comando di tensione, con benefici sulle prestazioni dinamiche dell'azionamento. Per contro, il funzionamento in PWM presenta alcuni svantaggi rispetto a quello in onda quadra: le ripetute inversioni di tensione causano un aumento delle perdite di commutazione negli interruttori di potenza e il circuito di controllo dell'invertitore risulta più complesso, in quanto deve calcolare e produrre più istanti d'inversione nel periodo della forma d'onda in uscita, aumentata rispetto al controllo ad onda quadra.
I benefici apportati dalla PWM nel funzionamento di un motore dipendono dal numero e dalla posizione delle inversioni. Per quanto riguarda il numero, risulta conveniente che sia il più alto possibile, compatibilmente con le limitazioni poste dalla massima frequenza di lavoro dei componenti di potenza e dalla capacità di calcolo del sistema di controllo dell'inverter.
Per la scelta della posizione delle inversioni esistono diverse tecniche di modulazione sia analogiche che digitali; in seguito verranno trattate le tecniche di modulazione vettoriale.

1.5.1 Modulazione vettoriale

In tempi abbastanza recenti è stata sviluppata una nuova generazione di tecniche, di tipo intrinsecamente digitale, che va sotto il nome di modulazione vettoriale (*space vector modulation, SVM*).
Essenzialmente, la *SVM* è una tecnica digitale che fa uso della rappresentazione vettoriale delle tensioni trifase da generare, utilizzando la definizione di vettore spaziale citata nel paragrafo 1.2.
Si supponga ad esempio di voler produrre una terna sinusoidale simmetrica di tensioni di fase, con pulsazione angolare ω ed ampiezza U_1:

$$\begin{aligned} U_a^* &= U_1 cos(\omega t) \\ U_b^* &= U_1 cos(\omega t - \frac{2\pi}{3}) \\ U_c^* &= U_1 cos(\omega t - \frac{4\pi}{3}) \end{aligned} \tag{1.9}$$

Applicando la definizione di vettore spaziale alla terna in esame, si ottiene il vettore di ampiezza pari a quella di ciascuna delle grandezze della terna trifase e rotante nel piano complesso con velocità angolare ω.

$$\mathbf{u}^* = u_\alpha^* + j u_\beta^* = U_1 e^{j\omega t} \tag{1.10}$$

La sua traiettoria è dunque una circonferenza di raggio U_1 e rappresenta la traiettoria del riferimento del vettore spaziale $\mathbf{u}^*$ da produrre.

Si suddivida ora la scala dei tempi in intervalli di durata T_c, chiamato tempo di modulazione (*switching period*), mentre il reciproco di T_c definisce la frequenza di modulazione.
Durante ogni intervallo T_c la terna trifase di tensioni da generare è assunta costante, per esempio pari al valore che assume all'inizio dell'intervallo in questione. Ogni fase è rappresentata, dunque, da un fasore spaziale (vedi par.1.2) costante in modulo (pari al valore di tensione raggiunto) e fase (corrispondente all'angolo assunto dalle sinusoidi nell'intervallo T_c attuale).
Consideriamo ad esempio di voler riprodurre un vettore di riferimento $\mathbf{u}^*$ che giace nel primo settore dell'esagono di Fig.1.8, compreso tra i due vettori di stato 100 e 110 come riportato in Fig.1.9.
Il vettore $\mathbf{u}^*$, formato da parte reale u_α^* e immaginaria u_β^* può essere scomposto secondo la formula (1.10) nelle due componenti $\mathbf{u}_1$ e $\mathbf{u}_2$, secondo le direzioni dei vettori di stato ad esso adiacenti.

$$\mathbf{u}^* = \mathbf{u}_1 + \mathbf{u}_2 \tag{1.11}$$

Si arriva così a generalizzare l'espressione per un vettore di riferimento che

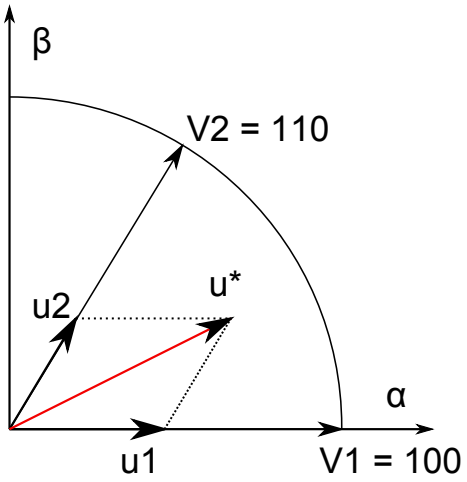


Fig. 1.9: Esempio di sintesi di un vettore spaziale

si trova nel settore m (m∈[1..6]):

$$\langle \mathbf{u}^* \rangle = \frac{T_m}{T_c}\mathbf{v}_m + \frac{T_{m+1}}{T_c}\mathbf{v}_{m+1} \tag{1.12}$$

dove si è indicato con T_m l'intervallo durante il quale l'inverter produce il vettore $\mathbf{v}_m$ e con T_{m+1} l'intervallo nel quale produce il vettore $\mathbf{v}_{m+1}$ che precede il vettore $\mathbf{v}_m$ in senso orario.
L'espressione dell'm-esimo vettore di stato è la seguente:

$$\mathbf{v}_m = \frac{2}{3} \cdot U_{dc} \cdot e^{j(m-1)\pi/3} \tag{1.13}$$

e per l'indice aumentato vale:

$$\mathbf{v}_{m+1} = \frac{2}{3} \cdot U_{dc} \cdot e^{jm\pi/3} \tag{1.14}$$

Sostituendo le (1.13) e (1.14) nella (1.12) e separando parte reale e immaginaria si ottengono due equazioni, che possono essere risolte nelle incognite T_m e T_{m+1}.
Adottando la (1.10) dopo qualche passaggio matematico si ottengono i due tempi:

$$\begin{aligned} T_m &= \sqrt{3}\frac{T_c}{U_{dc}}\left[u^*_\alpha \sin(m\pi/3) - u^*_\beta \cos(m\pi/3)\right] \\ T_{m+1} &= \sqrt{3}\frac{T_c}{U_{dc}}\left[-u^*_\alpha \sin[(m-1)\pi/3)] + u^*_\beta \cos[(m-1)\pi/3)]\right] \end{aligned} \tag{1.15}$$

Nella rimanente parte del periodo di commutazione viene applicato il vettore

nullo (000) oppure (111):

$$T_0 = T_c - T_m - T_{m+1} \tag{1.16}$$

Una volta calcolati i due tempi T_m e T_{m+1} di applicazione dei vettori di stato all'interno di un dato intervallo di commutazione, è necessario generare i comandi ai dispositivi di commutazione dell'inverter con la corretta tempistica.
A questo scopo sono state sviluppate diverse tecniche realizzative e nel prossimo paragrafo verrà analizzata una specifica tecnica: la modulazione vettoriale simmetrica.

1.5.2 Modulazione vettoriale simmetrica

Malgrado vi siano numerose possibili sequenze per applicare i vettori di stato, è facile intuire che il numero minimo di commutazioni degli interruttori che comandano l'inverter si ha quando si fa commutare un solo ramo del dispositivo da uno stato all'altro. Inoltre, dato che in ciascuno dei sei settori dell'esagono ci sono dei vettori nulli da applicare, conviene che ogni ciclo di modulazione inizi e termini con lo stesso vettore nullo.
Si è dimostrato che se si utilizzano vettori di stato adiacenti per la sintesi del vettore di riferimento si ottiene il contenuto armonico più favorevole possibile. La tecnica di modulazione vettoriale che rispetta tutti questi principi è la modulazione simmetrica (*double edge modulation*), in pratica, gli stati dell'inverter vengono sequenzialmente posizionati in modo tale da avere una simmetria delle commutazioni all'interno di un periodo di modulazione.
Ad esempio, in Fig.1.10 viene rappresentato il funzionamento nel primo settore: si riportano i comandi degli switch superiori di ogni ramo (S_a, S_b, S_c di Fig.1.6).
In questo esempio si è considerato un periodo di commutazione nel primo settore, quando gli stati attivi sono 100 per la durata T_1 e 110 di durata T_2 e vale l'equazione 1.16.

Gli impulsi sono centrati nella mezzeria del periodo di commutazione e lo stato nullo viene ricavato in parte agli estremi con il vettore 000 e in parte al centro con il vettore 111.
Dalla Fig.1.10 è immediato ricavare i tempi di accensione e di spegnimento di ciascuno switch, naturalmente essi variano con il variare del settore dove giace il vettore spaziale di riferimento di tensione:

$$\begin{aligned} T_{a,ON} &= \frac{T_0}{4} \\ T_{b,ON} &= \frac{T_0}{4} + \frac{T_1}{2} \\ T_{c,ON} &= \frac{T_0}{4} + \frac{T_1}{2} + \frac{T_2}{2} \end{aligned} \tag{1.17}$$

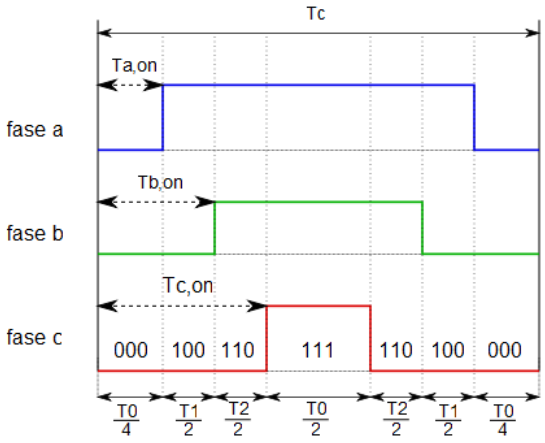


Fig. 1.10: Modulazione vettoriale simmetrica nel primo settore

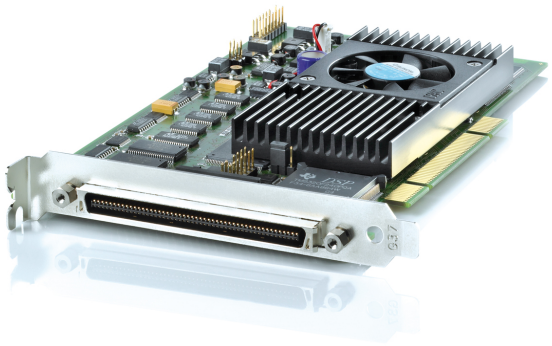
Risulta evidente che per una corretta applicazione della tecnica occorre stabilire per ogni intervallo di commutazione, il settore di appartenenza della tensione di riferimento $\mathbf{u}^*$.
Per ricavare il settore esistono numerosi metodi come ad esempio far ricorso a tabelle pre calcolate e poste in memoria dati del microprocessore, oppure calcolando il settore tramite un apposita routine di riconoscimento.

Capitolo 2

Programmazione del sistema prototipale dSPACE

2.1 La scheda di controllo DS1104

La scheda di controllo DS1104 R&D è una fast prototyping board prodotta dalla ditta tedesca dSPACE, progettata e utilizzata principalmente per lo sviluppo di controllori digitali multi variabili ad alta velocità e per simulazioni real-time in svariati campi, sia nel campo della ricerca che dello sviluppo e ingegnerizzazione di prototipi.



2.1.1 Architettura del sistema prototipale

Si tratta di un sistema completo per il controllo real-time basato sul processore floatingpoint 603 della famiglia PowerPC prodotto da Motorola, con frequenza di clock pari a 250 MHz. La scheda include un sottosistema, detto DSP basato sul microcontrollore TMS320F240 DSP della Texas Instrument per applicazioni I/O, con riferimento al controllo di motori elettrici: generazione di segnali PWM, misura della frequenza, I/O digitale, convertitori

A/D.
Tale sottosistema ha una frequenza di clock di 20 MHz ed è dotato di una memoria dual-port 4Kx16 bit usata per la comunicazione con il master PPC.

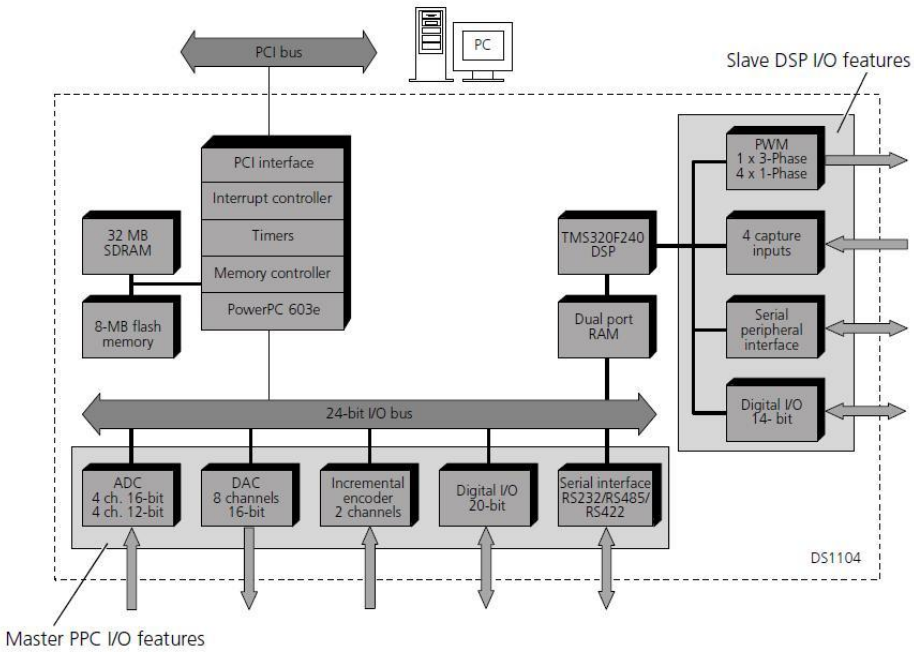


Fig. 2.1: Panoramica della scheda DS1104

Lo **slave DSP** è dotato di:

- Slave DSP Bit I/O Unit;
- Slave DSP Serial Peripheral Interface;
- Slave DSP Timing I/O Unit.

In particolare quest'ultima unità fornisce:

- uscite per la generazione di segnali PWM monofasi con *duty-cycle*, frequenze PWM e polarità variabili;
- uscite invertite e non invertite per la generazione di segnali PWM trifase con *duty-cycle*, frequenze PWM e tempi morti variabili;
- uscite invertite e non invertite per la generazione di segnali Space Vector-PWM (SVPWM) trifase;
- uscite per la generazione di segnali ad onda quadra con frequenza variabile;

- ingressi per la misura dei *duty-cycle* e della frequenza dei segnali PWM;

Per applicazioni di prototipizzazione rapida, appositi connettori di interfaccia e un pannello connettore consentono facili accessi a tutti i segnali di ingresso e di uscita della scheda.
Il pannello connettore CP1104 (Fig.2.2) consente semplici connessioni tra la scheda DS1104 R&D e i dispositivi ad essa connessi. In aggiunta al CP1104, il CLP1104 (*Connector/Led Combi Panel*) fornisce una serie di 54 LED indicanti lo stato dei segnali digitali.

Il pannello CP1104 contiene:

- 4 connettori A/D a 12 bit e 4 entranti in un unico convertitore a 16 bit mediante multiplexer, con tempi di campionamento rispettivamente pari a 800 ns e 2 μs;
- 8 uscite D/A con risoluzione a 16 bit e tempo di ricostruzione pari a 10 μs;
- un connettore digitale I/O;
- 2 connettori seriali UART (*Universal Asynchronous Receiver and Transmitter*);
- 2 connettori di interfaccia per encoder incrementale;
- un connettore di I/O Slave DSP digitale.

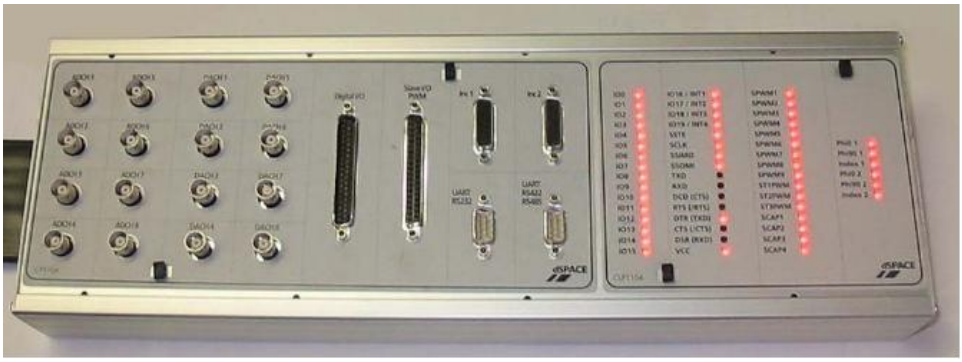


Fig. 2.2: Visuale pannello CP1104

2.2 L'ambiente dSPACE ControlDesk

Il software ControlDesk della dSPACE è l'interfaccia grafica a PC tra la scheda DS1104 R&D e l'utente.
Esso è un un pacchetto di software che consiste nei seguenti moduli:

- Experiment management, che consente di controllare tutti i dati relativi ad un esperimento. L'esperimento può essere caricato con la sola operazione di compilazione del modello in Simulink mediante il comando *Build* della finestra *Simulation* → *Simulation parameters* → *Real-Time Workshop*.
- Hardware management, che permette di configurare l'hardware della dSPACE e manipolare le applicazioni mediante una interfaccia grafica.
- Instrumentation kits, che offre una varietà di strumenti virtuali e di acquisizione dati.

Tra le innumerevoli applicazioni il programma ControlDesk permette di memorizzare l'andamento delle variabili presenti nell'applicazione real-time, mettendole a disposizione del Workspace di Matlab, di visualizzare su schermo l'andamento delle grandezze desiderate mediante uno strumento utilizzato come oscilloscopio o display numerico, di modificare in tempo reale i parametri dell'applicazione e di realizzare dei sinottici grafici come interfaccia utente per i programmi real-time in esecuzione sulla scheda DS1104.
Le funzioni del sistema prototipale sono innumerevoli e per ragioni di spazio vengono tralasciate. La guida a ControlDesk riporta tutte le funzioni spiegate in dettaglio.

2.2.1 Descrizione dell'ambiente di lavoro

L'ambiente di sviluppo in ControlDesk in Fig.2.3 si divide in tre parti fondamentali:

1. **Navigator**: è la sezione evidenziata in rosso in Fig.2.3 e raccoglie tre visualizzazioni:
   - Experiment: descrive e tiene traccia di tutti i programmi e layout[1] associati ad ogni esperimento.
   - Platform: descrive tutte le piattaforme simulative con cui può interfacciarsi il programma ControlDesk. Nel nostro caso contiene le icone del sistema DS1104.
   - Instrumentation: descrive tutti i layout aperti e per ciascun di essi tiene traccia degli strumenti utilizzati.

[1] Il layout costituisce l'insieme degli strumenti virtuali utilizzati nella work area e verrà descritto meglio nel prossimo paragrafo.

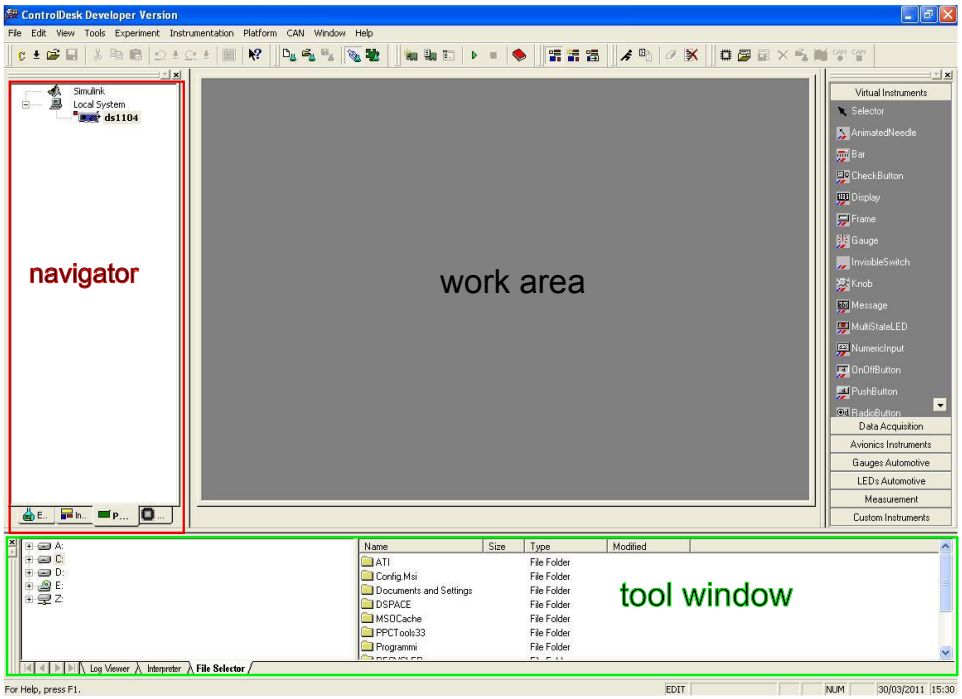


Fig. 2.3: Ambiente di sviluppo ControlDesk

2. **Tool window**: è la parte evidenziata in verde in Fig.2.3 che racchiude contenuti differenti.

   - Log viewer: finestra di visualizzazione di messaggi d'errore o di avvisi.
   - File selector: permette di selezionare l'applicazione da scaricare sulla scheda in modo grafico ed interattivo (*drag and drop*).
   - Interpreter: finestra dei messaggi dell'interprete Python utilizzabile da ControlDesk.
   - Variable manager: nella finestra del Variable Manager (vedi Fig.2.4) vengono visualizzate sotto forma di elenco ad albero tutte le variabili del programma caricato sulla scheda e le loro proprietà (per la maggior parte le variabili corrispondono ai blocchi *Matlab/Simulink*).

3. **Work area**: è l'area grafica a disposizione per realizzare interfacce grafiche (layout) per i programmi in esecuzione sulla scheda.
   Tramite i layout è possibile interagire con la scheda ad esempio variare parametri del programma mentre è in esecuzione, visualizzare l'andamento temporale di variabili d'interesse e gestirne la cattura dell'evoluzione temporale.

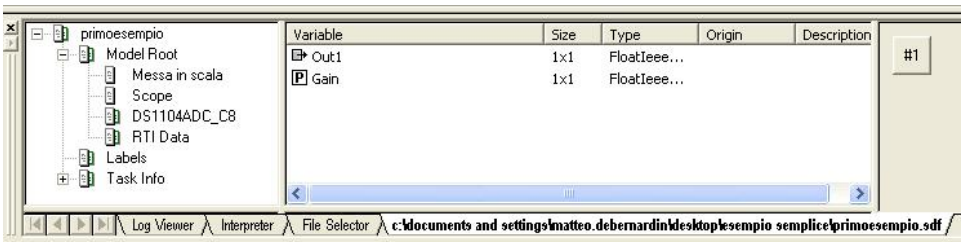


Fig. 2.4: Elenco ad albero delle variabili

- **Toolbar**: una volta caricato il programma nella scheda è possibile avviare e terminare l'esecuzione dello stesso attraverso i comandi di *Run, Stop* evidenziati in rosso in Fig.2.5.
  Nell'altro riquadro colorato in blu di Fig.2.5 sono inoltre presenti tre comandi per tre diverse modalità di utilizzo di ControlDesk:
- La modalità **Edit mode** nella quale la work area viene utilizzata per creare il layout.
- La modalità **Test mode** utilizzata per testare le funzionalità del layout creato.
- La modalità **Animation mode** nella quale il layout viene utilizzato dall'applicazione in esecuzione sulla scheda.

Fig. 2.5: Toolbar per controllo simulazione

2.2.2 Creare un layout

Per creare un nuovo layout (*Menu* → *File* → *New* → *New Layout*) compare in automatico nell'angolo in alto a destra del workspace un'altra finestra: quella della strumentazione virtuale (vedi Fig.2.6).
Il layout é inizialmente un'area di lavoro dove vanno inseriti gli strumenti desiderati, per fare questo è sufficiente selezionare dall'elenco della strumentazione lo strumento desiderato e in seguito "disegnare" con il mouse un rettangolo nel layout delle dimensioni che si desidera conferire allo strumento.

- Quando uno strumento non è associato a nessuna variabile, il rettangolo che lo racchiude è di colore rosso.
- Per associare una varibile ad uno strumento è sufficiente selezionare con il mouse la variabile dalla finestra *Variable Manager* e trascinarla dentro al rettangolo che racchiude lo strumento.

- Una volta fatta l'associazione il rettangolo diventa di colore grigio e porta il nome della variabile associata (nell'esempio in Fig.2.6 è stata assegnato un valore di guadagno ad uno strumento di input numerico).

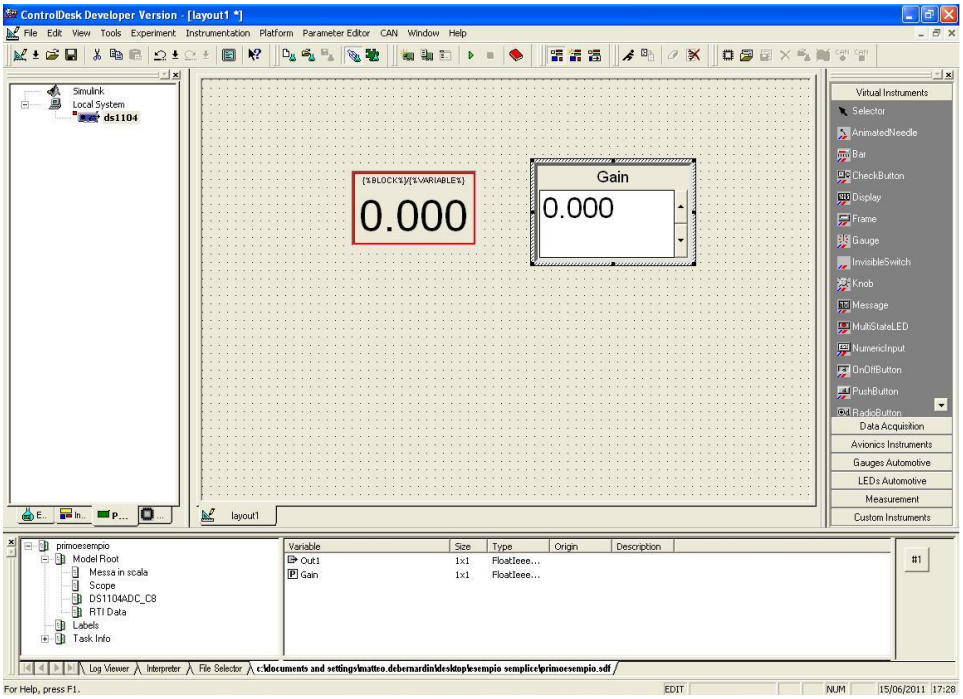


Fig. 2.6: Esempio di un layout

Tra la strumentazione virtuale esistono dispositivi di aquisizione dati tramite i quali è possibile visualizzare l'andamento temporale delle variabili. Gli strumenti che permettono di realizzare queste funzioni sono il *Plotter Array* che funge da oscilloscopio, mentre con lo strumento *Capture Settings* è possibile salvare i valori delle variabili d'interesse e memorizzarli su file, in modo da poterli rielaborare successivamente.

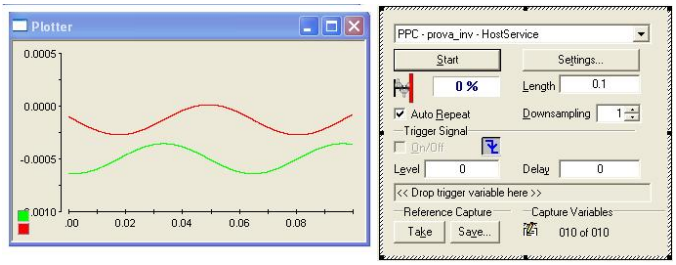


Fig. 2.7: Strumenti virtuali per l'acquisizione dati

Capitolo 3

Metodo d'implementazione di un progetto Real Time

3.1 Introduzione all'interfacciamento RTI

Nel campo ingegneristico difficilmente un controllore viene progettato senza simulazione. La tipica procedura di sviluppo è di creare un modello dell'impianto e simularlo al computer. Solamente in un secondo tempo viene aggiunto il controllore del processo e ci si occupa della sua ottimizzazione. Nella prima fase di sviluppo la simulazione è condotta mediante Simulink: la caratteristica principale di questo tipo di simulazione è che, essendo il computer molto rapido nei calcoli, se il modello del processo da controllare è semplice il risultato può essere calcolato velocemente. Se il modello risulta più complesso, è necessario molto più tempo per ricavare i dati richiesti. Tuttavia, visto che non è necessario avere un ristretto tempo di simulazione in Simulink, non è necessario ridurre la complessità del modello e si mantiene, così, una buona approssimazione.
Una volta che il modello simulato rispetta le specifiche richieste si inserisce il controllore nel sistema attuale e si testano le sue prestazioni nel complesso. Questo procedimento tiene conto che il controllore potrebbe richiedere ulteriori modifiche, le quali devono essere fatte prima di passare alla realizzazione hardware. Per fare tutto questo si connette il sistema reale ad un controllore simulato in real-time mediante la scheda prototipale. Questa tecnica viene chiamata Rapid Control Prototyping (RCP).
Quando il controllore simulato è in grado di controllare il modello reale si può passare alla fase finale di costruzione passando alla realizzazione hardware. Per il test finale si connette il controllore reale ad un modello dell'impianto simulato in real-time, in questo modo si può verificare che il controllore non contiene errori e non può danneggiare il sistema fisico. Questa tecnica è definita Hardware-In-the-loop Simulation (HIL).
La simulazione in tempo reale rappresenta un aspetto fondamentale dal

punto di vista del controllo, la stessa importanza vale per la generazione automatica del codice real-time che deve essere in seguito implementata nell'hardware.
Nei sistemi dSPACE l'interfaccia real time (RTI/RTI-MP) fornisce un collegamento tra queste due funzioni principali quando viene affiancata al real-time Workshop di Matlab[1], infatti facendo lavorare assieme le due piattaforme risulta possibile generare il codice real-time da utilizzare nel sistema dSPACE direttamente dal modello Simulink e farlo caricare automaticamente nell'hardware.
I vantaggi di questa generazione automatica del codice sono i seguenti:

- Non è necessario convertire manualmente il modello Simulink in un altro linguaggio come ad esempio il C;
- Non bisogna preoccuparsi di creare delle strutture di programma real-time e realizzare delle chiamate a funzione di I/O;
- Non bisogna preoccuparsi di implementare e scaricare il codice all' interno dell' hardware dSPACE.

[1] ©The MathWorks

3.2 Configurare Matlab

3.2.1 Selezione della piattaforma

Le librerie RTI supportano tutte le piattaforme dSPACE in commercio. Se si è selezionata più di una piattaforma durante l'installazione bisogna scegliere quale piattaforma utilizzare visto che non è possibile lavorare con più piattaforme contemporaneamente.
Per selezionare la piatttaforma all'avvio di Matlab una finestra di dialogo come quella in Fig.3.1 verrà aperta e sarà possibile scegliere con quale scheda lavorare.

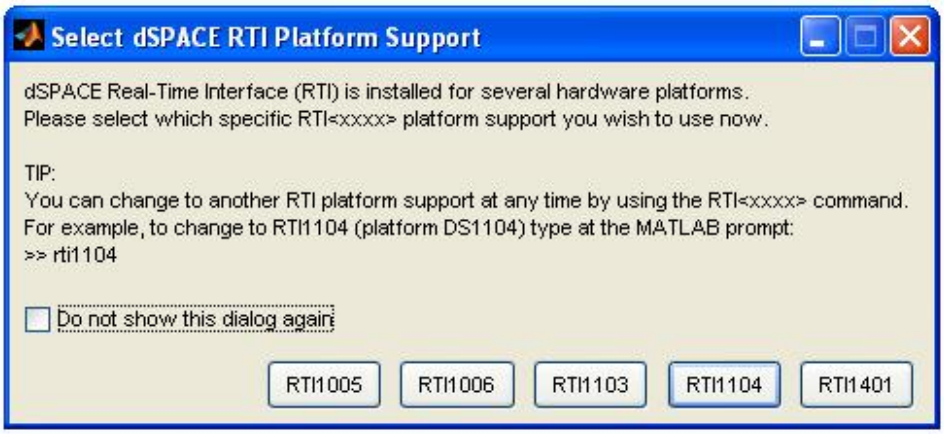


Fig. 3.1: Finestra di selezione piattaforma

3.2.2 Parametri di simulazione

Dal menu principale di Simulink si seleziona la voce *Simulation* → *Configuration Parameters* (Fig.3.2) e si settano i seguenti dati[2]:

- **Tempo di simulazione** (Simulation Time)
  - Inizio esecuzione (Start time): 0.
  - Fine Esecuzione (Stop time): valore numerico oppure *inf* (esecuzione senza fine).
- **Campionamento** (Solver options)
  - Motore di risoluzione: a passo fisso (Fixed step).

[2]Le impostazioni indicate devono essere sempre utilizzate per un corretto funzionamento del modello in fase di progettazione.

  - Ampiezza del passo: selezionare la modalità assegnata (es. *Fixed step size* = 2e-5).
  - Modalità: si può scegliere tra *single tasking* o *multi tasking* (selezionare auto).

- **Real Time Workshop**
  - System target file: selezionare rti1104.tlc
  - Language: C.
  - Makefile configuration: selezionare *generate makefile* e deselezionare l'opzione *generate code only*.

Per quanto riguarda la configurazione degli altri settaggi la scelta è dell'utente in base al lavoro che deve essere fatto, si consiglia comunque di scorrere e leggere tutte le opzioni per poter scegliere di adottare le prestazioni ottimali.

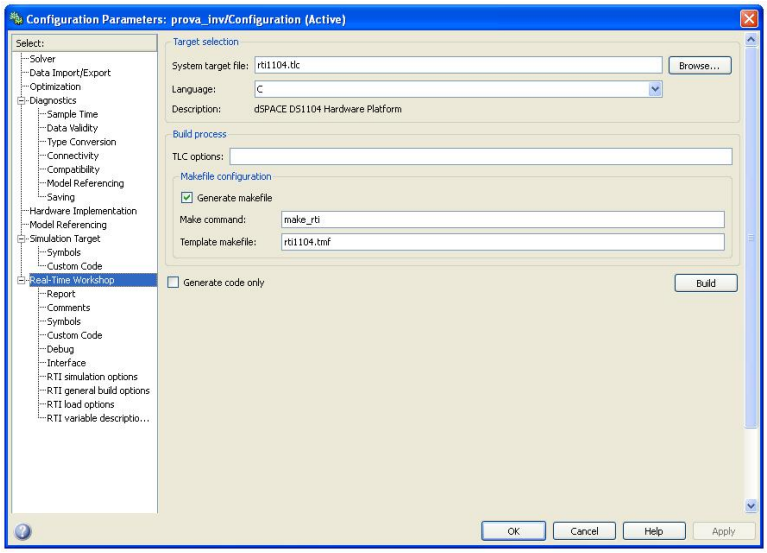


Fig. 3.2: Finestra di configurazione parametri

3.3 Come generare un'applicazione real-time

L'esempio seguente rappresenta un primo semplice lavoro eseguito per prendere dimestichezza nella creazione di un applicazione real-time.
Si utilizza l'ambiente Matlab e tutta la libreria di blocchi Simulink di I/O, messi a disposizione dalla dSPACE, per realizzare una particolare applicazione real-time e per configurare l'hardware della scheda.

Esempio: acquisizione di un segnale

Il primo semplice modello che si costruisce accede al canale di input analogico (uno degli ADC della scheda) e permette di visualizzare l'andamento temporale di un segnale generato con un generatore di funzioni esterno.
Il sistema Simulink descritto deve contenere un blocco che permetta l'accesso ad uno degli ADC della scheda (Fig.3.12), per poi elaborare il segnale acquisito, opportunamente scalato, e infine inviarlo ad un blocco di visualizzazione (in realtà è un blocco utilizzatore dei segnali, in seguito sarà chiarito questo punto).

Realizzazione passo-passo:

1. Si crea il modello inserendo il blocco per la lettura dell ADC settando il canale che vogliamo utilizzare in ingresso, (in questo caso il canale 8), un blocco *Gain* di guadagno per la messa in scala del segnale e un blocco *Scope* per poter vedere il segnale (Fig.3.3).

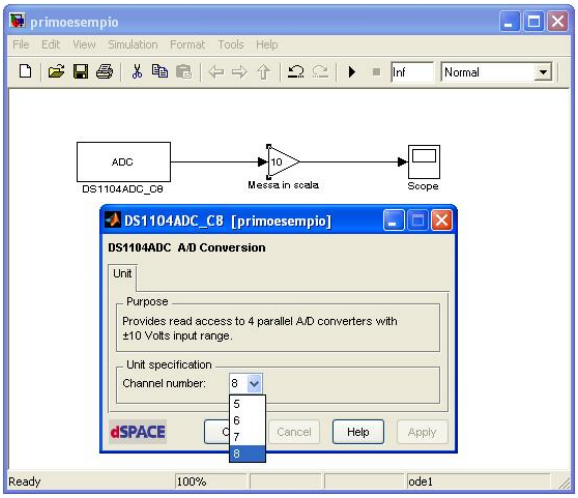


Fig. 3.3: Modello Simulink

2. Una volta completato il modello si salva il file in una cartella creata dove si desidera che vengano inseriti tutti i file generati in automatico dopo la compilazione.

3. Si procede nella compilazione mediante i tasti *Tools* → *Real Time Workshop* → *Build Model*(Fig.3.4).

4. Durante la compilazione nel Workspace di Matlab è possibile visualizzare le operazioni che vengono compiute dal compilatore e analizzare eventuali errori (Fig.3.5).

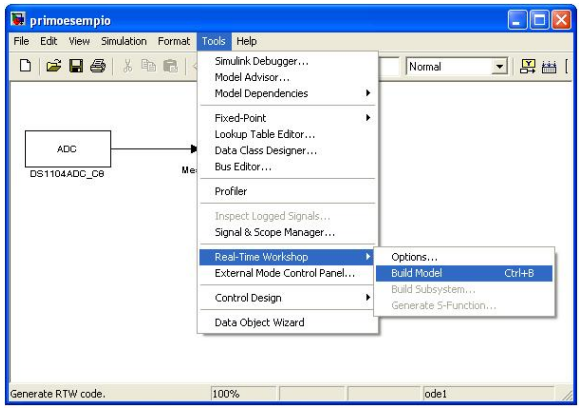


Fig. 3.4: Compilazione del modello

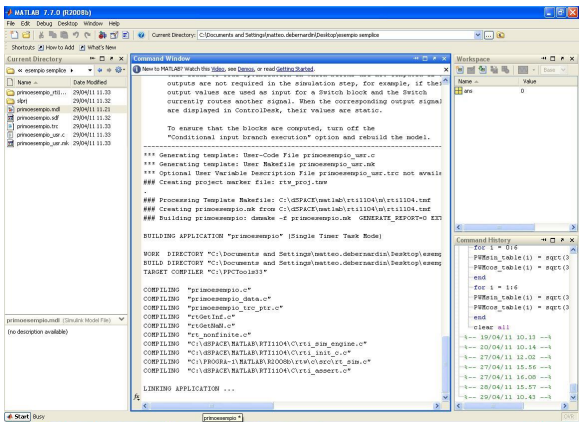


Fig. 3.5: Analisi del Workspace durante la compilazione

5. Una volta ultimata la compilazione, in assenza di errori, vengono generati in automatico il codice C dell'applicazione eseguita e i file con estensione come segue:

   - PPC: file che contiene codice eseguibile da caricare sul DSP.
   - SDF: file descrittore del programma RT, da utilizzare col programma ControlDesk (System Description File).
   - TRC: Il Trace File è il descrittore delle variabili presenti nel programma RT. Si presenta come un elenco ad albero di tutte le variabili relative ai blocchi Matlab/Simulink inseriti. Questo file viene aperto in automatico nel *Variable Manager* se ControlDesk viene avviato avviato dopo che è terminata la generazione del codice in Matlab col caricamento del programma sulla scheda e la sua esecuzione, altrimenti si attiva col comando *File* → *Open Variable File*.

- MAP: file descrittore delle variabili con le informazioni relative all'indirizzamento di ciascuna variabile nella RAM della scheda DSP (Map File).

Successivamanete il codice generato dalla compilazione viene caricato automaticamanete sulla scheda e viene eseguito. A questo punto per osservare i risultati o variare i parametri si utilizza l'interfaccia fornita da ControlDesk.

6. Si apre ControlDesk e si crea un nuovo esperimento. Se nella *Platform navigator* l'icona della scheda è in pausa si effettua nuovamente la compilazione del modello Simulink finchè nel *Variable Manager* situato nella *Tool Window*, non apparirà il file di descrizione del sistema (*nomefile.sdf*) e l'icona della scheda passerà dalla modalità pausa a play (riquadro rosso in Fig.3.6).

7. Dal file *sdf* si trascina la variabile chiamata scope all'interno di uno strumento di acquisizione dati creato (plotter array) e si passa in configurazione *animation mode* per visualizzare in tempo reale il segnale sinusoidale acquisito dall'ADC (Fig.3.6).

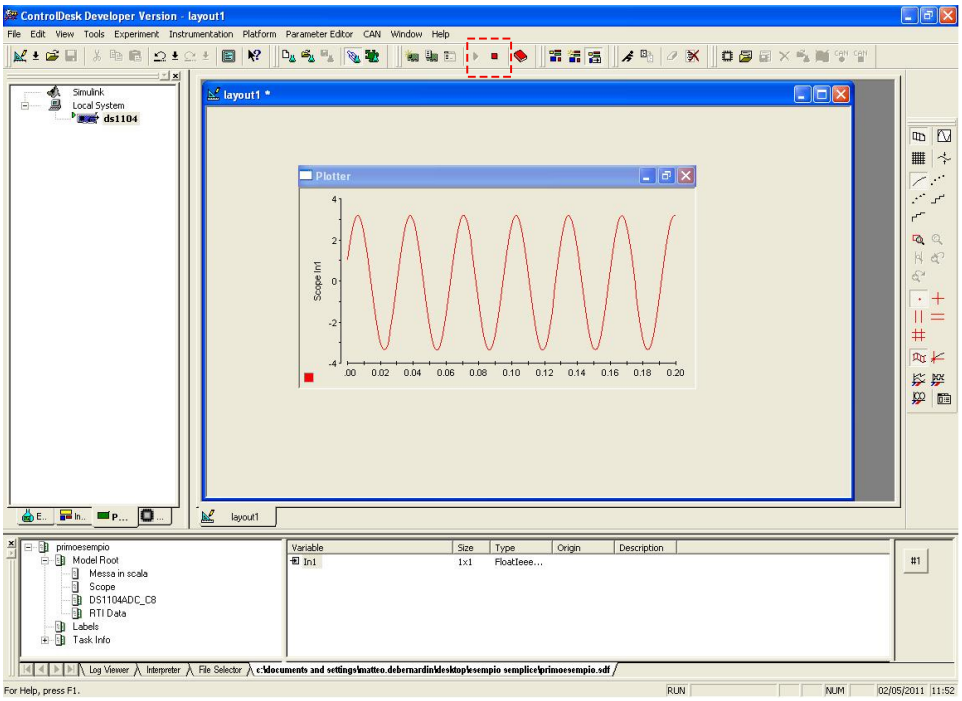


Fig. 3.6: Acquisizione del segnale sinusoidale via Controldesk

3.3.1 Ottimizzare la fase di progettazione

- Gli ingressi e le uscite che non vengono utilizzati vanno sempre connessi ai blocchi Simulink di terminazione, per evidenziare il loro non utilizzo in modo che non venga generato codice inutile per l'elaborazione di quei segnali.

- Nel modello Simulink il blocco RTIData (Fig.3.7) viene generato automaticamente durante la prima generazione del codice per la scheda DSP, esso contiene informazioni relative all'intervallo di campionamento, alla durata dell'esecuzione sulla scheda DSP, all'algoritmo d'integrazione numerica scelto e per nessun motivo deve essere tolto dal modello o copiato in altri modelli.

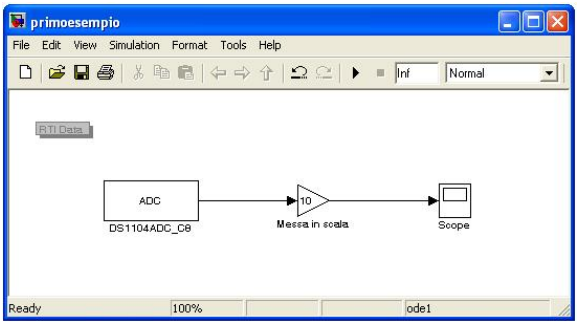


Fig. 3.7: Blocco RTIData

- I blocchi Simulink di tipo scope non generano codice da eseguire sulla scheda DSP.

3.3.2 Escludere sottosistemi dal file TRC

Escludere un sottosistema dal file descrittore delle variabili (TRC) è una delle tecniche di ottimizzazione implementate dalla RTI.
Ci sono due principali motivi per escludere un sottosistema dalla generazione del file TRC: il primo è di ridurre la dimensione del file TRC stesso: normalmente un file TRC generato contiene tutti i segnali definiti nel modello che si progetta, questo significa che nel caso di modelli molto grandi e complicati il file TRC abbia dimensioni elevate; di conseguenza il file richiede maggiore tempo per essere generato e questo va ad intaccare i tempi di caricamento in ControlDesk.
Il secondo motivo è quello di nascondere delle parti del modello ad altri utenti.

Per escludere un insieme di blocchi dal file le operazioni da fare sono:

1. si accede alla libreria TaskLib e si seleziona il blocco TrcExclusion(Fig.3.8) dalla libreria Extras.
2. si aggiunge il blocco nel sottosistema da escludere.

Per default il blocco è settato a 1 (attivo) e risulta abilitata l'esclusione del sottosistema dove è inserito, è possibile però modificare lo stato del blocco e portarlo a 0 (inattivo) per generare i segnali nel file TRC.

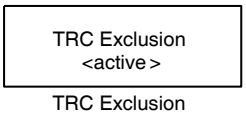


Fig. 3.8: Blocco per l'esclusione del codice dal file TRC

3.3.3 Priorità d'esecuzione dei blocchi

Tra le varie operazioni che si possono svolgere accedendo alle proprietà di un blocco con il tasto destro del mouse, vi è la possibilità di configurare l'ordine d'esecuzione del blocco all'interno del flusso complessivo di esecuzione del modello, come se non ci fossero dipendenze topologiche.
Per influenzare l'esecuzione del blocco occorre:

Fig. 3.9: Settare l'ordine d'esecuzione di un blocco

1. nel campo *priory* si specifica un valore intero, anche negativo e zero, minore è il numero assegnato e maggiore è la priorità del blocco (Fig.3.9).

2. se si sta lavorando con un applicazione Real-Time-Multiple-Processing (RTI-MP), nella quale si ha a che fare con più processi da gestire, una volta definita la priorità di ogni processo bisogna controllare che non ci siano conflitti in esecuzione con gli altri processi.

3.4 Librerie RTI1104

La libreria d'interfaccia real-time (RTI) per la scheda DS1104 fornisce i blocchi RTI che realizzano le capacità in input-output della DS1104 nei modelli Simulink.
Questi blocchi sono progettati per specificare la progettazione hardware nelle applicazioni real time; inoltre la rtlib1104 fornisce blocchi RTI addizionali, modelli demo e informazioni utili.
Per accedere alla libreria si digita rti1104 nella Comand-Window di Matlab e appare una finestra come quella in Fig.3.10 dove si possono distinguere i vari componenti disponibili.

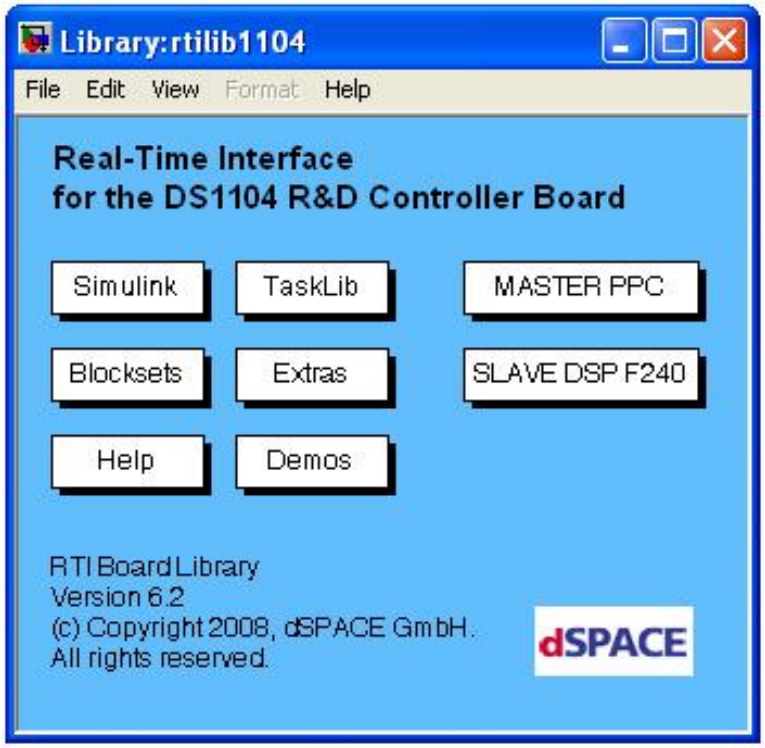


Fig. 3.10: Panoramica delle librerie RTI

- **Simulink**: chiama la libreria Simulink dei blocchi standard.
- **Blocksets**: include blocchi RTI opzionali per la DS1104.

- **Help**: visualizza il manuale per le informazioni.
- **Readme**: visualizza le informazioni dell' ulttimo minuto e le note importanti.
- **TaskLib**: fornisce i blocchi RTI per le interruzioni in Simulink.
- **Extras**: fornisce i blocchi RTI per funzioni speciali.
- **Demos**: visualizza modelli esempio.
- **Master PPC**: è una sotto libreria che comprende i blocchi RTI per le unità input-output fornite dal Power PC Processor.
- **Slave DSP F240**: è una sotto libreria che comprende i blocchi RTI per le unità di input-output fornite dalla Texas Instruments TMS320F240 DSP.

Nei paragrafi successivi verranno esaminati i blocchi fondamentali utilizzati nella fase sperimentale in laboratorio, la descrizione di ogni singolo blocco è effettuata in modo che leggendo questo lavoro si possa prendere una certa famigliarità nel creare dei progetti utilizzando Simulink, per un maggiore approfondimento dei blocchi si consiglia la visione del manuale dSPACE.

3.4.1 RTILib Master ppc

La libreria relativa al Master contiene i blocchi necessari per interfacciare la scheda con l'esterno e per sincronizzare le varie unità di input/output. Verranno in seguito presi in considerazione i blocchi fondamentali che verranno utilizzati in seguito nella fase sperimentale nel capitolo 4.

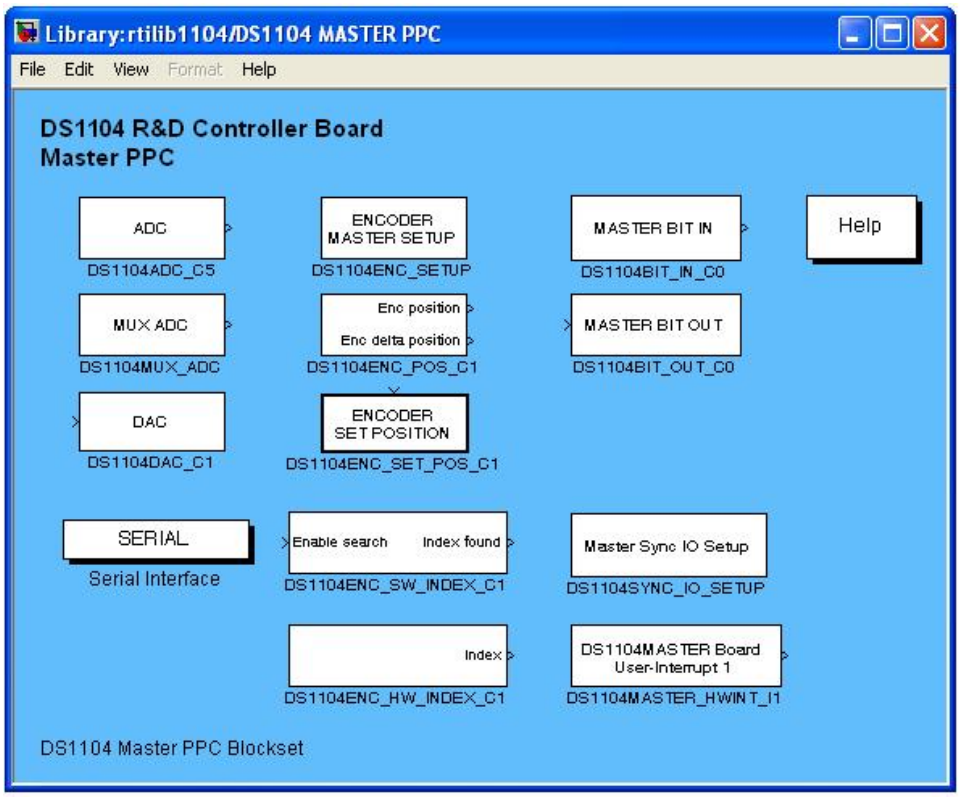


Fig. 3.11: Sottolibreria MasterPPC

- **adc**: è utilizzato per l'acquisizione di segnali analogici in ingresso come ad esempio correnti o tensioni. Esso consente di leggere selezionare uno dei canali disponibili del pannello connettore, per selezionare il canale è sufficiente modificare le proprietà del blocco.

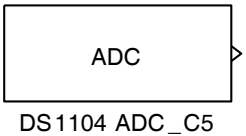


Fig. 3.12: ADC

- **mux adc**: ha le stesse funzionalità del blocco ADC, l'unica differenza è che fornisce l'accesso in lettura a quattro canali selezionati con un multiplexer.

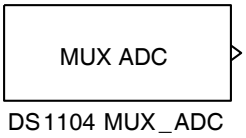


Fig. 3.13: ADC

- **dac**: permette di scrivere in uno solo degli otto canali convertitori d/a del pannello connettore. Esso converte il segnale d'ingresso in una tensione analogica in uscita.

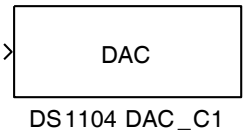


Fig. 3.14: DAC

- **encoder master setup**: setta i parametri globali dei due canali dell'encoder. Esso deve essere inserito nel modello per poter utilizzare tutti gli altri blocchi d'interfaccia dell'encoder.
  Per i due canali possono essere scelti due tipi di segnali:
  - Differential (RS422)
  - Single-ended (TTL)

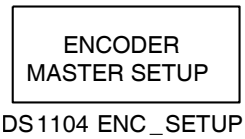


Fig. 3.15: ENCODER MASTER SETUP

- **encoder pos**: legge le posizione e la variazione della posizione di uno dei due canali dell'encoder.
  La posizione viene espressa sotto forma di impulsi (encoder lines) e se si vuole utilizzare questo blocco nel modello è necessario inserire anche il blocco MASTER SETUP (Fig.3.15).
  I parametri del blocco sono:
  - **unit spcification**: per specificare quale canale di encoder si utilizza.

– **position value**: permette d'inizializzare la posizione dell' encoder all' inizio della simulazione dando come valore di riferimento la posizione sotto forma di numero di impulsi (lines), ad esempio il valore di posizione 1000.75 corrisponde a 1000 impulsi e 3/4 in quanto ogni impulso vale 0.25.

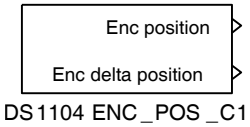


Fig. 3.16: ENCODER POSITION

- **encoder set position**: quando il blocco è triggerato con un fronte di salita alto durante la simulazione, il conteggio del canale specifico viene settato al valore scelto.

  – **channel number**: seleziona il canale dell' encoder.

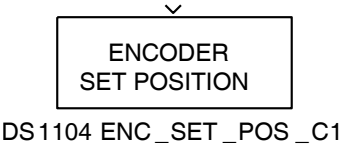


Fig. 3.17: ENCODER SET POSITION

- **master bit in**: fornisce l'accesso in lettura a uno dei 20 bit in ingresso del MASTER PPC.

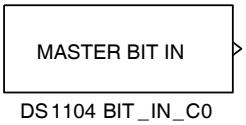


Fig. 3.18: MASTER BIT IN

- **master bit out**: fornisce l'accesso alla scrittura a un output a 20 bit del MASTER PPC.

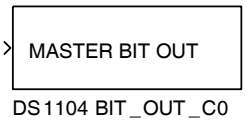


Fig. 3.19: MASTER BIT OUT

- **master sync io setup**: blocco che deve essere inserito per configurare il trigger nelle varie unità e i parametri attraverso i quali si desidera vengano triggerate.
  I parametri del blocco sono:
  - **trigger source**: per selezionare il segnale di trigger, esso può essere un segnale interno o esterno.
  - **input units**: si selezionano le unità di input che si vogliono sincronizzare e il fronte di salita e discesa per il trigger.
  - **output units**: si selezionano le unità di output che si vogliono sincronizzare e il fronte di salita e discesa per il trigger.

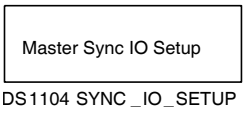


Fig. 3.20: MASTER SYNC IO SETUP

- **master board user interrupt**: abilita e imposta l'interruzione hardware del microprocessore.

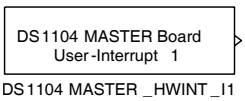


Fig. 3.21: MASTER BOARD USER INTERRUPT

- **serial interface**: I blocchi della libreria serial interface vengono utilizzati per realizzare comunicazione seriali.

3.4.2 RTILib Slave dsp

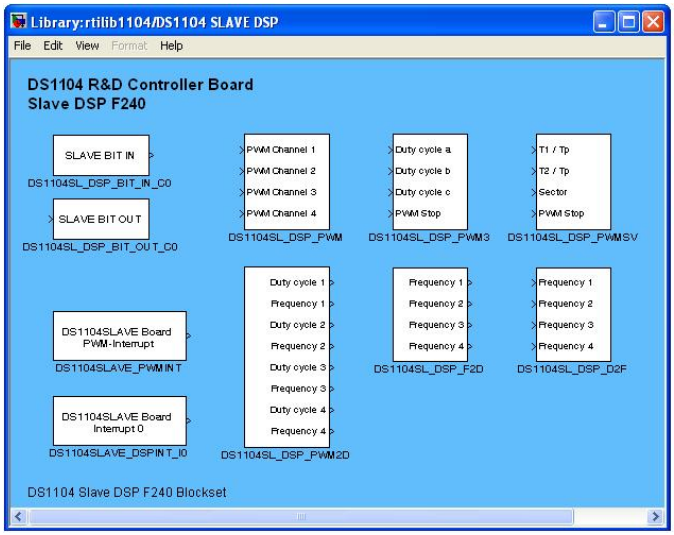


Fig. 3.22: Sottolibreria Slave DSP

La libreria relativa allo SLAVE DSP è composta dai seguenti blocchi necessari per l'interruzione dello SLAVE e per la generazione della PWM.

- **slave bit in**: fornisce l'accesso in lettura a un singolo bit dei 14 bit in ingresso.

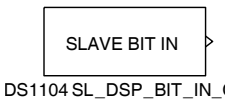


Fig. 3.23: SLAVE BIT IN

- **slave bit out**: fornisce l'accesso in scrittura a uno dei 14 bit d'uscita.

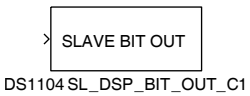


Fig. 3.24: SLAVE BIT OUT

- **slave board interrupt**: permette l'interruzione dello SLAVE da parte dell'utente.

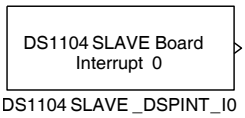


Fig. 3.25: SLAVE BOARD INTERRUPT

- **slave board pwm interrupt**: rende disponibile l'interrupt PWM dello slave dsp della scheda come sorgente di trigger per far si che possano essere calcolati i tempi necessari per la creazione della PWM. Esso è generato nel mezzo della fronte alto o del fronte basso del segnale (settandolo inizialmente) del periodo dell'onda PWM.

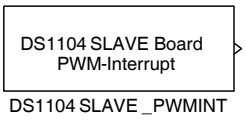


Fig. 3.26: SLAVE BOARD PWM INTERRUPT

- **pwmsv**: genera i segnali trifase PWM con uscite invertite e non invertite. Si possono impostare l'enable e i tempi morti variabili.
  Per utilizzare il blocco deve essere inviato un interrupt PWM dallo slave dsp al master ppc attraverso il blocco PWM INTERRUPT.
  In ingresso al blocco si devono inserire i due tempi T1 e T2 (1.15), il settore dove è localizzato il vettore spaziale e il valore booleano per lo stop della pwm (1 se si desidera interrompere la generazione e 0 per riprenderla).
  I parametri del blocco sono:
  - **pwm frequency**: si setta la frequenza della pwm in un range 1.5 - 5 MHz.
  - **deadband**: si inserisce il tempo morto tra il segnale originale e il suo negato in uscita.
    Il valore massimo ammesso è di 100μs e per nessun motivo deve essere maggiore del 50% del periodo di PWM

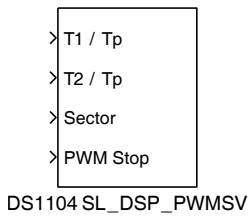


Fig. 3.27: PWMSV

Capitolo 4

Fase sperimentale del controllo RT

4.1 Realizzazione del modello Simulink per il controllo

Nei capitoli precedenti è stata presentata una panoramica del funzionamento della scheda DS1104 e dell'utilizzo delle librerie RTI Simulink, in questo capitolo si passerà alla fase di sperimentazione nella quale si cercherà, come primo obiettivo, di generare una PWM trifase sinusoidale per comandare un inverter di tensione.
Come prima sarà analizzato in dettaglio lo schema presentato nel primo capitolo (Fig.1.5) raffigurante un controllo in corrente; esso può essere suddiviso in sotto sistemi di dimensioni ridotte, in modo tale da poter descrivere in maniera accurata il comportamento di ogni parte e il modo in cui è stata realizzate nel modello Real-time.
Il modello utilizzato può essere in prima analisi suddiviso in due parti fondamentali:

1. la prima, è la sezione principale del programma, racchiude i blocchi necessari per le seguenti operazioni:

   - chiamata alla sotto-routine di generazione della PWM;
   - implementazione della protezione dell'inverter nel caso di loop con il software di controllo (Watchdog);
   - comandare l'abilitazione e la disabilitazione dell'inverter (Enable inverter);

2. la seconda, invece, è composta dai blocchi contenuti nella routine d'interruzione per la generazione della PWM, in essa devono essere implementate le seguenti azioni:

- lettura delle correnti di fase, della tensione continua U_{dc} e della posizione angolare;
- trasformazione delle correnti nel sistema $\alpha - \beta$ e successivamente nel sistema d-q;
- controllo delle correnti i_d, i_q con regolatori PI;
- nuova trasformazione delle tensioni in uscita dei controllori da d-q in $\alpha - \beta$;
- algoritmo di decisione del settore per il vettore spaziale relativo alla tensione;
- routine per la creazione degli intervalli temporali della PWM;
- routine per generazione PWM con vettore spaziale.

4.1.1 Sezione principale

Come prima operazione per la generazione della PWM è stato necessario realizzare un modello Simulink per permettere l'interruzione del microprocessore di controllo motore (Slave), vedi Fig.4.1. L'interruzione genera il periodo di PWM nel quale vengono chiamate le subroutine di misura delle correnti e viene effettuato il calcolo dei periodi di commutazione. Il blocco d'interruzione PWM (*Slave Board PWM-Interrupt*) è stato settato allineando l'interruzione a metà periodo mentre nel blocco di sincronizzazione degli ingressi (*Master Sync IO Setup*) sono stati abilitati i convertitori analogico-digitali (ADC), in particolare il canale 5 è stato impostato per la lettura della tensione del bus in continua mentre canali 6 e 7 per la lettura delle correnti di fase i_a, i_b.
Per implementare la chiamata alla funzione di calcolo e generazione PWM è stato creato un sottosistema (interrupt driven subsystem) che conterrà al suo interno il modello dove verranno realizzate tutte le operazioni per la generazione della PWM. Tali operazioni saranno analizzate in dettaglio nei paragrafi successivi.
Nel modello principale del progetto sono stati inseriti inoltre due blocchi *master bit out* (Fig.3.19) con i quali si accede in scrittura al bit 16 e 18 che sono i rispettivi bit degli ingressi/uscite IO16 e IO18 per realizzare la protezione del circuito (watchdog e enable inverter).
Per realizzare la funzione di enable inverter, che permette di attivare e disattivare manualmente l'inverter durante la fase di simulazione, è stata inserita una porta nand che riceve in ingresso un onda quadra di periodo $2 \cdot 10^{-4}$ sec e un valore booleano impostato a "0", in modo che l'inverter rimanga disabilitato finchè il valore non viene modificato dall'utente e portato al valore logico "1" tramite un pulsante (Enable Inverter) comandato dal layout del progetto.
La soluzione adottata per proteggere il circuito di potenza è il watchdog:

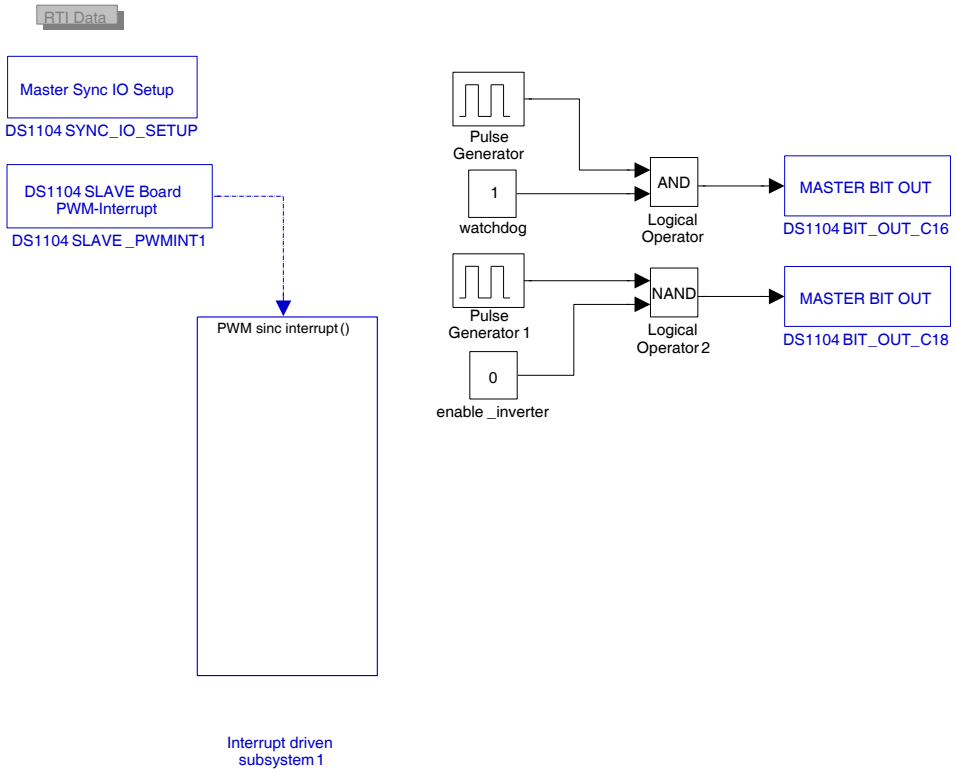


Fig. 4.1: Modello principale

essenzialmente è la realizzazione Simulink di un multivibratore monostabile che richiede in ingresso un segnale oscillante ad una frequenza di 10 KHz: se per qualche motivo il programma si blocca in un loop l'oscillazione smette di esistere e l'inverter viene immediatamente disabilitato.

4.1.2 Lettura correnti, posizione angolare e tensione di bus

La sezione per la lettura dei dati è rappresentata in Fig.4.2; i canali di ADC 6, 7 sono utilizzati per la lettura delle correnti i_a, i_b dalle quali si ricava, attraverso Kirchhoff[1], la corrente i_c (questo metodo di misura della corrente i_c garantisce realmente che la somma delle correnti sia nulla, evitando così il problema dell'offset). La tensione del bus in continua viene invece ottenuta dal canale 5 di ADC.
Le correnti e la tensione V_{dc} vengono in seguito moltiplicate per un fattore di scala. Le correnti vengono inoltre limitate con il blocco di saturazione per evitare che la corrente massima superi il valore limite di sicurezza; inoltre correnti e tensione possono essere private di un eventuale errore di offset

[1]Il primo principio di Kirchhoff dice che la somma delle correnti entranti in un nodo è nulla

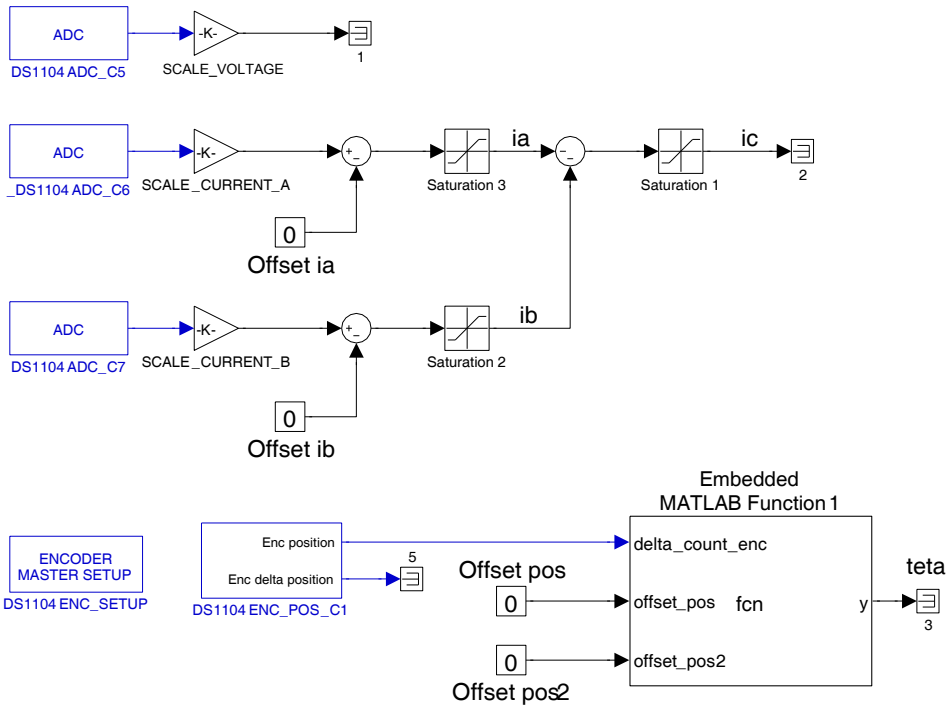


Fig. 4.2: Blocchi fase input

in fase di prova attraverso un nodo sommatore settato inizialmente ad un valore di offset nullo.
Per quanto riguarda la posizione angolare è richiesto l'uso del blocco *encoder setup* (Fig.3.15) e di quello *encoder position* (Fig.3.16): quest'ultimo fornisce in uscita un valore di posizione angolare espresso come conteggio di linee da parte dell'encoder (lines). Per avere la posizione in radianti è stata inserita un *embedded Matlab function*; le istruzioni relative a tale funzione sono riportate in appendice A.1.
Il codice inserito nella embedded Matlab function 1 per il calcolo della posizione angolare contiene delle costanti: ad esempio il numero di coppie polari del motore o la risoluzione dell'encoder.
In fase di progettazione deve essere tenuto conto di questi valori se, ad esempio, si decide di utilizzare il modello per un motore con un diverso numero di coppie polari, oppure si vuole modificare la risoluzione dell'encoder. Si potrebbero usare delle variabili di ambiente di Matlab, da inizializzare prima della compilazione.

4.1.3 Trasformazioni e controllo di corrente

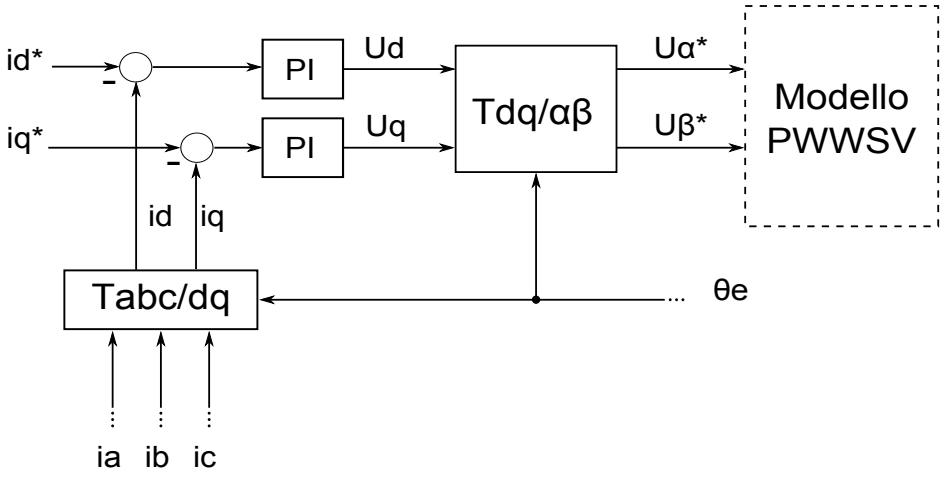


Fig. 4.3: Schema di un controllo di corrente in dq con regolatori PI

Una volta acquisite le correnti di fase, la tensione V_{dc} e la posizione elettrica, si passa alla fase di trasformazione dei riferimenti e al controllo di corrente in dq realizzato con regolatori di tipo proporzionale integrale descritto nel paragrafo 1.3.
Riprendendo in esame lo schema di un controllo di corrente (Fig.4.3) introdotto nel primo capitolo è possibile implementare i vari blocchi e creare il modello Simulink di Fig.4.4.
Nello schema del controllo di corrente si distinguono due blocchi relativi ai

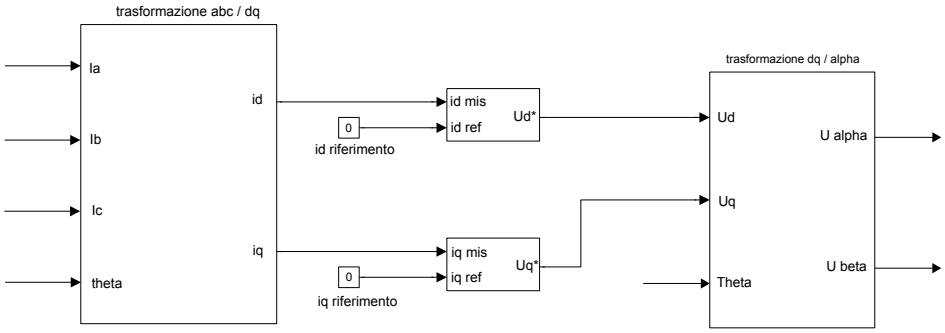


Fig. 4.4: Realizzazione Simulink dello schema in Fig.4.3

sottosistemi che realizzano le trasformazioni delle coordinate $T_{abc/dq}$ e $T_{dq/\alpha\beta}$ descritti dalle matrici (4.1) e (4.2) e i due regolatori di tipo proporzionale-integrale.
Il blocco di trasformazione $T_{abc/dq}$ realizza la trasformazione espressa dalla

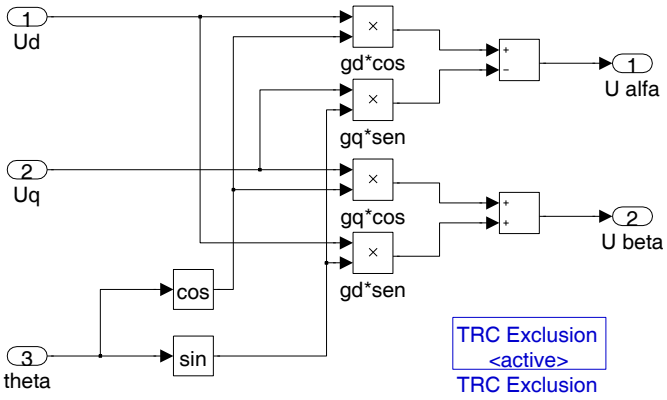


Fig. 4.5: Sottosistema per la trasformazione dq/$\alpha\beta$

relazione matriciale:

$$T_{abc\to dq} = \frac{2}{3}\begin{vmatrix} \cos(\vartheta_{dq}) & \cos(\vartheta_{dq} - 2\pi/3) & \cos(\vartheta_{dq} - 4\pi/3) \\ -\sin(\vartheta_{dq}) & -\sin(\vartheta_{dq} - 2\pi/3) & -\sin(\vartheta_{dq} - 4\pi/3) \end{vmatrix} \quad (4.1)$$

Mentre il blocco di trasformazione $T_{dq/\alpha\beta}$ viene realizzato mediante una funzione descritta dalla matrice:

$$T_{dq\to\alpha\beta} = \begin{vmatrix} \cos(\vartheta_{dq}) & -\sin(\vartheta_{dq}) \\ \sin(\vartheta_{dq}) & \cos(\vartheta_{dq})) \end{vmatrix} \quad (4.2)$$

Nella Fig.4.5 si può vedere un esempio di come è stata implementata la matrice di trasformazione[2] che permette la trasformazione $T_{dq/\alpha\beta}$, semplicemente utilizzando dei blocchi di moltiplicazione e somma della libreria Simulink; anche per quanto riguarda la trasformazione $T_{abc/dq}$ è stato utilizzato un approccio del tutto simile. Infine, i due regolatori inseriti sono

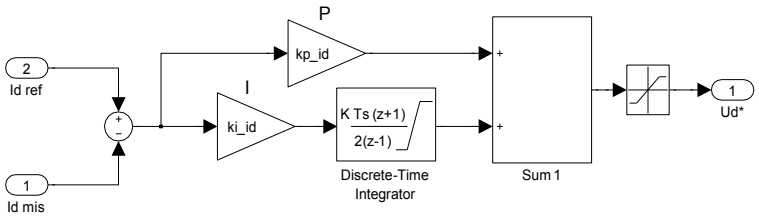


Fig. 4.6: Regolatore PI

stati realizzati secondo il modello in Fig.4.6: essi ricevono in ingresso i riferimenti di corrente i_d^* e i_q^*, entrambi settati con un valore costante nullo che

[2]In entrambi i sottosistemi che svolgono le trasformazioni dei riferimenti è stato inserito un blocco di esclusione *Trace Exclusion* (*TRC*) per evitare la generazione di variabili inutili nel *trace file* durante la fase di compilazione del modello

in seguito sarà possibile modificare dal layout ControlDesk attraverso uno strumento di input numerico.

4.1.4 Generazione della modulazione Space Vector

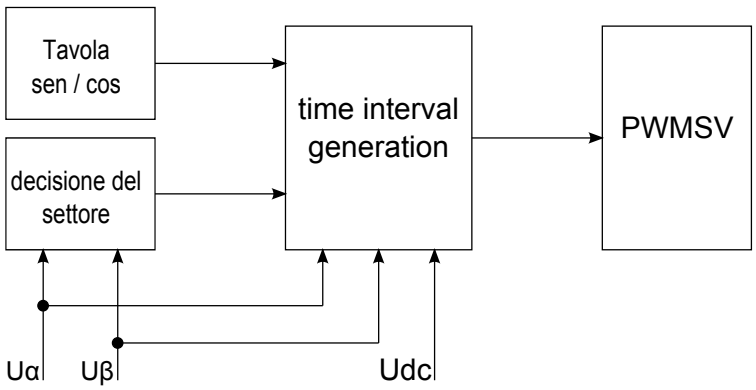


Fig. 4.7: Schema generazione PWM

La generazione della modulazione vettoriale PWMSV, descritta nel paragrafo 1.5.1, può essere schematizzata secondo la Fig.4.7 ed è stata realizzata con il modello Simulink di Fig.4.8.

Per generare la PWMSV si è inserito il blocco *PWMSV* (Fig.3.27) inizializ-

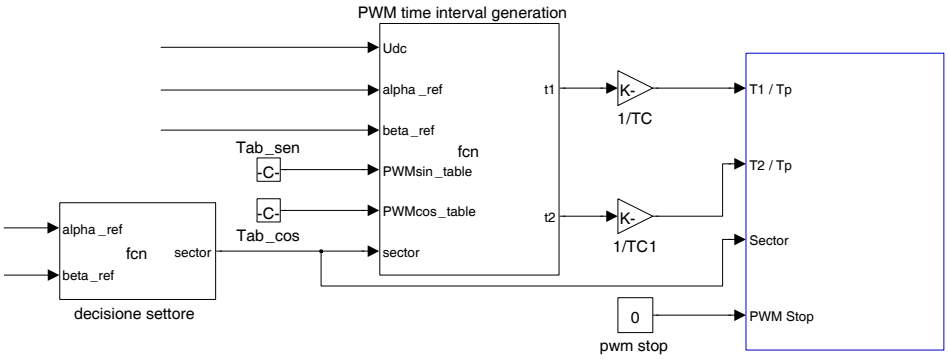


Fig. 4.8: Modello Simulink per generazione PWM

zato con i seguenti valori: deadband di 10μs e frequenza di PWM di 10KHz. Inoltre il blocco PWMSV richiede in ingresso i seguenti parametri:

- Due tempi T_1/T_p e T_2/T_p, dove T_1 e T_2 sono stati definiti nel par.1.15.
- Il settore dell'esagono dove si trova il vettore spaziale di tensione, che deve essere un numero intero tra 1 e 6.

- Un segnale booleano per fermare la generazione della PWM (gestione allarmi).

Il blocco per la decisione del settore, realizzato con una *s-function* riportata in appendice A.2, riceve in ingresso le tensioni di riferimento trasformate nel sistema $\alpha\beta$ (vedi 4.1.3) e restituisce il settore dell'esagono nel quale si trova il vettore spaziale descritto dalle due tensioni U_α e U_β.
Il blocco *PWM time interval generation*, implementato con una s-function definita dalle equazioni di Clark Park e riportata in appendice A.3, genera i due tempi T_1, T_2 descritti dalle formule citate al paragrafo 1.15.
I parametri da fornire al blocco ad ogni iterazione di PWM sono:

- il settore dove risiede il vettore spaziale.
- La tensione V_{dc} (vedi par.4.1.2).
- Le due tensioni U_α e U_β (vedi par.4.1.3).
- Una tabella contenente i valori che assume il seno calcolato nel settore.
- Una tabella contenete i valori assunti dal coseno calcolato nel settore.

Il vantaggio di passare come ingressi al blocco di generazione dei tempi due vettori di valori pre calcolati di seno e coseno è fornire un rapido accesso ai dati in essi contenuti, evitando così al processore di calcolare ad ogni modifica del vettore spaziale (in particolare del settore dove giace) il relativo valore assunto dal seno e dal coseno, risparmiando memoria e diminuendo così i tempi di calcolo di ogni iterazione di PWM.
Nel M-file di Matlab *Costanti.m* riportato in appendice, è inserito il codice utilizzato per generare le due tavole di valori[3]:
Una volta che la *s-function* ha effettuato il calcolo di di T_1 e T_2 i due valori vengono divisi per il periodo di modulazione dell'invertitore T_c e vengono inviati in ingresso al blocco *PWMSV* che in automatico genera i tre segnali di comando PWM.

4.1.5 Test del modello: generazione PWM

Ad ultimazione avvenuta del modello Simulink per il controllo di corrente è cominciata una fase graduale di test per verificare il coretto funzionamento di tutti i blocchi che compongono il modello finale ricavato nei paragrafi

[3]quando si realizza una struttura di tipo array in Matlab bisogna tenere conto che l'array viene indicizzato partendo dall'indice 1 e proseguendo. Al contrario, quando si realizza una struttura array in C la prima cella d'array ha indice 0. Nell'implementare le formule per il calcolo dei tempi e per il calcolo delle tabelle del seno e coseno è stato tenuto conto di questa differenza nei linguaggi: sommando al settore un offset di 1 in tutte le equazioni, in modo che ci sia una corrispondenza biunivoca nei due linguaggi, ho ovviato al problema.

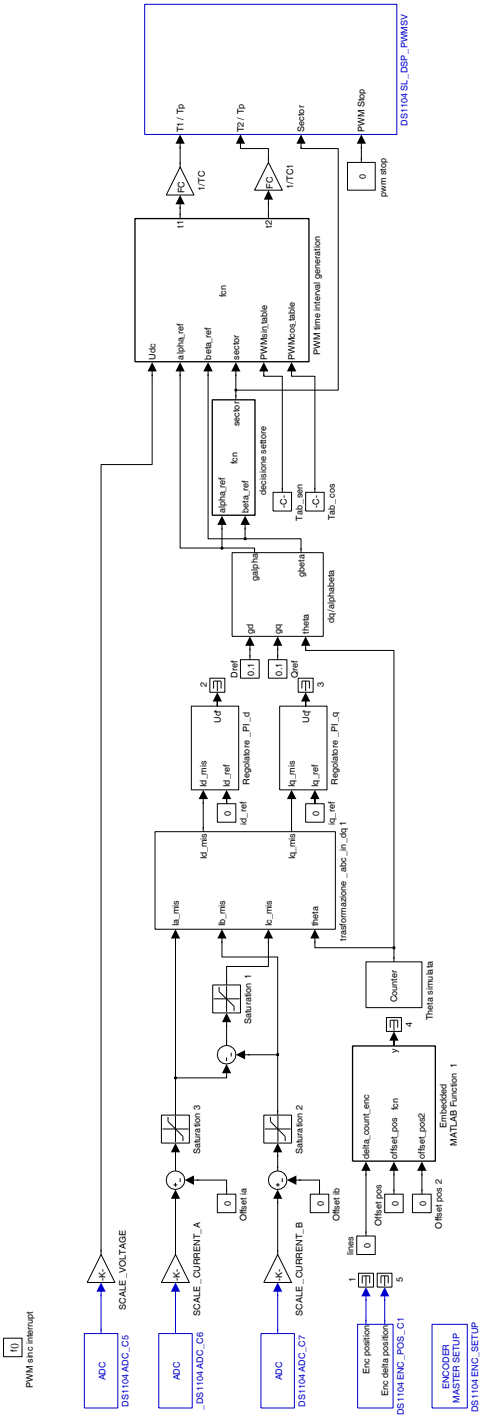


Fig. 4.9: Modello Simulink per il test della routine di calcolo PWM

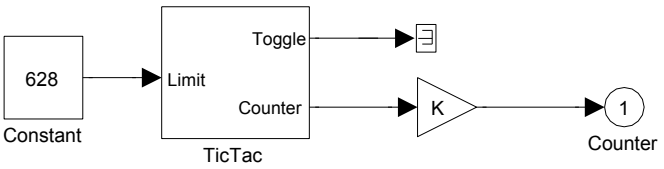


Fig. 4.10: Blocco simulazione posizione angolare

precedenti. La prima verifica è stata: controllare il corretto funzionamento dei trasduttori per la lettura delle correnti e del bus-DC; per fare questo è stato collegato in ingresso agli ADC un generatore di segnale e si sono visualizzate sul layout di ControlDesk le onde generate, in modo da poter tarare i fattori di scala presenti per avere una corrispondenza precisa tra i segnali in ingresso e quelli visualizzati al pc.
Una volta testato il corretto funzionamento dell'encoder per la lettura della posizione angolare, (dente di sega che varia tra 0 e 2π), è stato disconnesso il relativo blocco **Embedded Matlab Function 1** (Fig.4.2) sostituendolo con il sotto sistema chiamato **Theta simulata** (Fig.4.10) che simula ininterrottamente la lettura di un encoder fatta su un motore in movimento, per ricavare infine lo schema presentato in Fig.4.9.
Per realizzare il dente di sega tipico della lettura di un encoder è stato utilizzato il blocco *Tic Tac* fornito dalle librerie Simulink: composto da un contatore che incrementa il suo valore ogni volta che il blocco viene eseguito e si azzera quando viene raggiunto il limite fissato come ingresso[4]; con il contatore settato con un limite di 628 come in Fig.4.10 si ricava una posizione angolare misurata di periodo T $\approx$ 0.06 sec. corrispondente ad una frequenza f$\approx$ 16.7 Hz (Fig.4.11).

Per generare la PWM è stato disconnesso l'anello di corrente, e si sono inseriti dei valori costanti di riferimento di tensione: Dref = 0.1 V, Qref = 0.1 V (rispettivamente per l'asse d e l'asse q) e tensione del bus in continua V_{dc} = 100 V.
Con un oscilloscopio digitale sono stati misurati i comandi di modulazione, come mostrato nella Fig.4.12. Si possono distinguere i segnali di comando dei rami superiori dell'inverter; la modulazione è simmetrica, come si vede in figura dove è riportato un periodo di modulazione (T_c = 100 μs).
Per verificare la misura delle correnti di fase è stato collegato, come carico RL, un motore asincrono trifase 'Lafert' con potenza di 1.5 Hp a frequenza 50 Hz.
Utilizzando il blocco per simulare la posizione angolare descritto preceden-

[4]Dato che il conteggio viene incrementato a passi di uno, per far arrivare il dente di sega al valore 2π si è impostato un valore intero del limite del conteggio di 628 che viene poi diviso dal guadagno K, con questo metodo è possibile aumentare o diminuire la frequenza del dente di sega generato modificando il numero di cifre del limite e il guadagno dell'amplificatore.

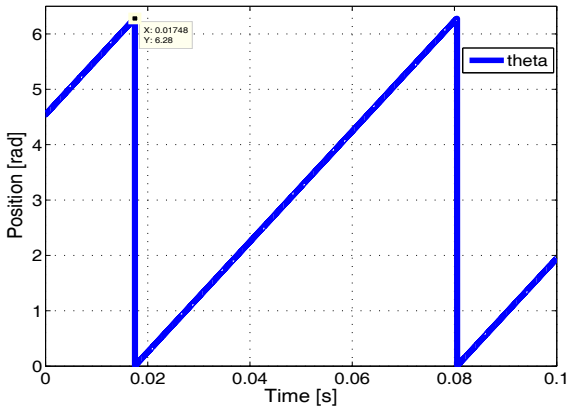


Fig. 4.11: Theta simulata

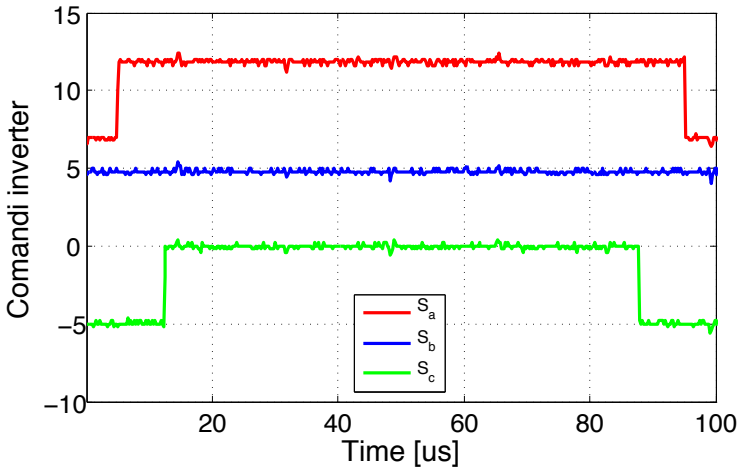


Fig. 4.12: Segnali di comando dei rami alti dell'inverter

temente è stato implementato un azionamento Volt-Hertz; lo scopo è di verificare che il sistema di controllo progettato fornisca una forma d'onda d'uscita desiderata.
Attraverso oscilloscopio digitale sono state misurate le correnti entranti in ogni fase del carico RL, è stato sufficiente misurate solamente due delle tre correnti in quanto la terza si ricava facilmente dalle precedenti.
Le forme d'onda ricavate sono visibili in Fig.4.13, si può notare la forma sinusoidale delle correnti e, di conseguenza, delle fem (non visualizzate). Il periodo vale $T_{mis} \approx 0.06$ sec e corrisponde ad una frequenza di 16.6 Hz.
Dato che la verifica ha riscosso esito positivo, il sistema è pronto per lo stadio successivo di chiusura degli anelli di corrente e verifica del controllo.

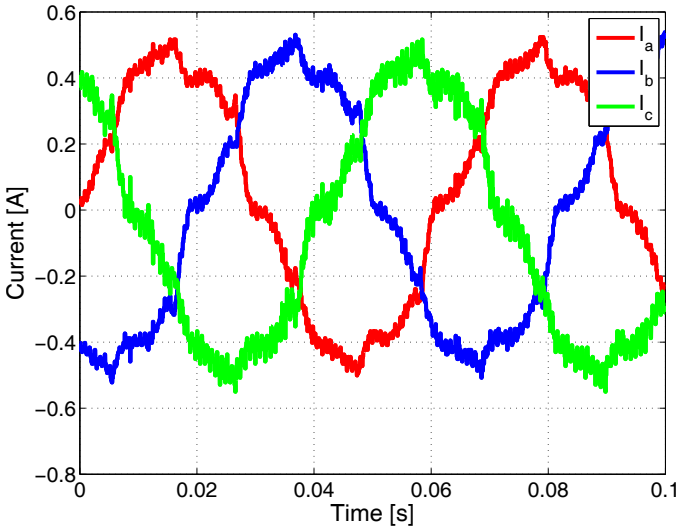


Fig. 4.13: Correnti di fase misurate

Conclusioni

Con questa esperienza si è dimostrato come sia possibile far operare dei modelli sulla scheda dSPACE DS1104 attraverso l'interfacciamento in tempo reale (RTI); è stato verificato che è possibile configurare tutti gli input/output graficamente, inserendo i blocchi delle librerie Simulink nel modello del progetto. Inoltre, la generazione del codice da caricare nella scheda attraverso Real-Time Workshop© avviene quando il modello Simulink viene compilato, scaricato e avviato sulla scheda, tutto in automatico.
Questo innovativo modo di realizzare prototipi minimizza i tempi in fase di progettazione e rende più semplice la fase di programmazione da parte dell'utente: per esempio, invece di scrivere diverse righe di codice in linguaggio C per implementare la lettura di un ADC è sufficiente inserire il blocco relativo nel modello e successivamente la compilazione il codice viene creato in automatico.
Come sviluppo futuro è possibile passare al controllo motore ad anello chiuso, per poi provare altri blocchi sperimentali, ad esempio controlli senza sensore di posizione.

Appendice A

Codici Matlab

Codice A.1: Codice embedded Matlab function 1 per determinare la posizione angolare

```
function y = fcn(delta_count_enc,offset_pos,offset_pos2)
PP = 2;         //numero di coppie polari
PPR_ENC=2000;  //linee encoder
GEAR_RATIO=1;

pulse2rad =pi*2/(PPR_ENC*GEAR_RATIO);
enc_pulse=0;
enc_pulse=enc_pulse - delta_count_enc;
 while (enc_pulse >= PPR_ENC)
  enc_pulse = enc_pulse-PPR_ENC;
 end

 while (enc_pulse < PPR_ENC)
   enc_pulse =enc_pulse + PPR_ENC;
 end
ang_m = enc_pulse*pulse2rad-offset_pos2-offset_pos;
 while (ang_m>pi*2)
   ang_m=ang_m-pi*2;
 end

 while (ang_m<0)
    ang_m=ang_m+pi*2;
 end
ang_e=PP*ang_m;
 while (ang_e>pi*2)
    ang_e=ang_e-pi*2;
 end
 while (ang_e<0)
    ang_e=ang_e+pi*2;
 end
y = ang_e;
```

Codice A.2: Codice per il riconoscimento del settore dove si trova il vettore spaziale di tensione

```
function sector  = fcn(alpha_ref,beta_ref)
ualpharef_rad3 = alpha_ref * sqrt(3);

if (beta_ref >= 0)
  if (alpha_ref >= 0)
    if (ualpharef_rad3 > beta_ref)
       sector = 1;
    else
       sector = 2;
    end
  else
    if (- ualpharef_rad3 <= beta_ref)
          sector = 2;
     else
      sector = 3;
     end
    end
 else
    if (alpha_ref < 0)
     if (- ualpharef_rad3 > - beta_ref)
       sector = 4;
      else
       sector = 5;
      end
    else
      if (ualpharef_rad3 <= -beta_ref)
       sector = 5;
      else
       sector = 6;
      end
    end
end
```

Codice A.3: codice del blocco che realizza la generazione dei tempi T_1 e T_2

```
t1 = (1/( Udc))*(alpha_ref*PWMsin_table(sector+1)-
   - beta_ref*PWMcos_table(sector+1));

t2 = (1/( Udc))*(-alpha_ref*PWMsin_table(sector)+
   + beta_ref*PWMcos_table(sector));
```

File ‘costanti.m’ per il modello Matlab/Simulink:

```
clear all
close
%Numero di coppie polari
 PP = 2;
%Conversione radianti --> rpm
VEL_ME2RPM  =   60/(2*pi*PP); %velocità meccanica in rpm
RAD_S2RPM  = 60/2*pi; %trasformazione da rad/s in rpm
RPM2RAD_S = 2*pi/60; %trasformazione da rpm in rad/s

%Parametri controllori PI
kp_id=60;
ki_id=0.5;
kp_iq=60;
ki_iq=0.5;

%Limiti di tensione
Ud_lim=50;  %limite Ud
Uq_lim =50; %limite Uq

%Parametri encoder
PPR_ENC=2000;
GEAR_RATIO=1;
PULSE2RAD =pi*2/(PPR_ENC*GEAR_RATIO);

%Fattore di scala DC bus
SCALE_VOLTAGE   = 510.0  ;
%Fattori di scala correnti di fase */
SCALE_CURRENT_A  = 14.5;
SCALE_CURRENT_C  = 14.44;
SCALE_CURRENT_B  = 14.5;
I_MAX =5;  %limite di corrente

%Periodo e frequenza PWM */
TC =100.0e-6;   %Periodo di modulazione dell'inverter
TCMEZZI =50.0e-6; %Tc/2
FC =10.0e+3;
DEADBAND = 4.0e-6;
PWMsin_table=[0 0 0 0 0 0 0];
PWMcos_table=[0 0 0 0 0 0 0];

%Generatore di seni e coseni
for i = 1:7
 count= i-1;
    PWMsin_table(i) = sqrt(3)*TC*sin(count*pi/3);
    PWMcos_table(i) = sqrt(3)*TC*cos(count*pi/3);
end
```

Bibliografia

[1] Syed A. Nasar and F. C. Trutt, *Electric Power Systems*, department of electrical engineering, university of Kentucky.

[2] Silverio Bolognani, *Dispense di Azionamenti elettrici*, Università degli studi di Padova, 2006.

[3] A. E. Fitzgerald - Charles Kingsley Jr. - D. Sthephen, *Electric machinery*, Mc Graw hill, sesta edizione.

[4] M. E. El-Hawary, *Principles of electric machines with power electronic applications*, Reston book, Prentice-Hall.

[5] Bin Wu, *High-Power Converters and AC drives*, IEEE press.

[6] M. Andriollo - G. Martinelli - A. Morini, *Macchine elettriche rotanti*, libreria internazionale Cortina Padova, seconda edizione.

[7] D. Grahame Holmes - Thomas A. Lipo, *Pulse Width Modulation for power converters*, Wiley-interscience, 2003.

[8] W. Shepherd - L. N. Hulley - D. T. W. Liang, *Power electronics and motor control*, Cambridge University Press, second edition.

Ringraziamenti

Vorrei concludere questa trattazione ringraziando il prof. Silverio Bolognani, relatore di questa tesi, per la grande disponibilità e la cortesia dimostratemi e per tutto l'aiuto durante la stesura.

Un doveroso ringraziamento va a tutti i dottorandi e ai componenti dello staff del laboratorio di Azionamenti elettrici di Padova, in particolare a Luca e a Mattia, che si sono sempre dimostrati pazienti e disponibili a dirimere i miei dubbi durante ogni attività di quest'esperienza.

Un sentito ringraziamento va ai miei genitori, che, con il loro incrollabile sostegno morale ed economico, mi hanno permesso di raggiungere questo traguardo.

Un ultimo grazie va rivolto a tutti i miei amici, che mi hanno sempre sostenuto in questi anni di studi standomi vicini nei momenti di difficoltà. Un particolare pensiero va rivolto a Ivan, che non può essere qui fisicamente a festeggiare questo passo significativo della mia vita, ma che porto sempre nel cuore.