Università degli Studi di Padova
Facoltà di Ingegneria

UNIVERSITÀ DEGLI STUDI DI PADOVA
FACOLTÀ DI INGEGNERIA

—

DIPARTIMENTO DI INNOVAZIONE MECCANICA E GESTIONALE

—

TESI DI LAUREA TRIENNALE IN INGEGNERIA DELL'AUTOMAZIONE

PROGETTAZIONE E REALIZZAZIONE DI DRIVER PER MOTORE IN CORRENTE CONTINUA

RELATORE: CH.MO PROF. ING. GIULIO ROSATI

LAUREANDO: DI VITTORIO ALBERTO

ANNO ACCADEMICO 2009-2010

“ E quindi uscimmo a riveder le stelle ”

Dante - Divina Commedia (Inferno XXXIV, 139)

Indice

Sommario

Uno SCARA-robot è un manipolatore a tre gradi di libertà, capace di una movimentazione completa sul piano orizzontale.
Nella configurazione più comune esso presenta due *link*, movimentati da altrettanti motori in corrente continua.

Al DIMEG (Dipartimento di Ingegneria Meccanica E Gestionale) di Padova è stato realizzato un manipolatore avente anche una base mobile, la quale fornisce un grado di libertà ridondante. Grazie a questa opzione è possibile creare delle alternative alle configurazioni non singolari del robot.
Per questo manipolatore la rotazione consentita ai due link senza azionare i finecorsa induttivi è rispettivamente $\pm 125°$ e $\pm 155°$. E' quindi fondamentale la realizzazione dei due driver per i motori in continua.

Introduzione

Questo elaborato è volto a presentare la progettazione e la realizzazione del driver per un motore in corrente continua.
La realizzazione di due driver in continua è parte fondamentale di un progetto ideato dallo scrivente congiuntamente ad altri quattro studenti del corso di Ingegneria dell'Automazione dell'Università di Padova, il quale prevede la costruzione e il controllo di un manipolatore planare a tre gradi di libertà (quattro considerando anche il movimento dell'$end\text{-}effector$).

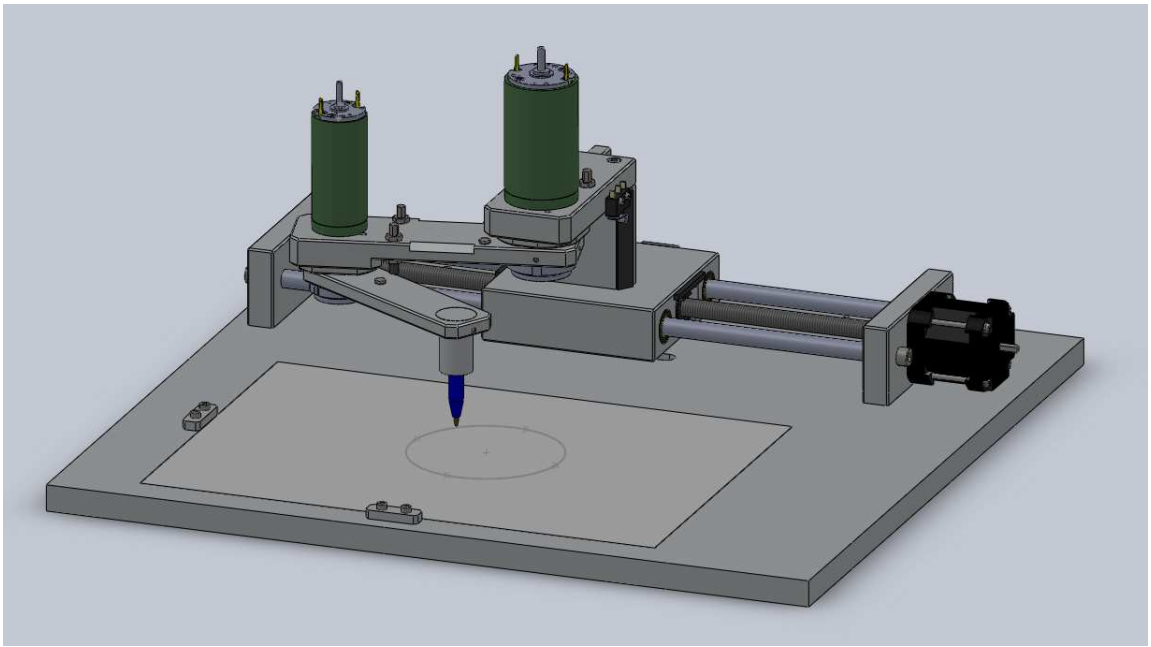


Figura 1: Manipolatore progettato.

Capitolo 1

Introduzione al progetto

L'obiettivo dell'intero progetto è quello di realizzare un manipolatore di tipo SCARA con base mobile.

Lo SCARA (Selective Compliant Assembly Robot Arm) è un manipolatore a tre gradi di libertà, sempre più usato all'interno dei progetti industriali. Esso presenta non solo la possibilità di compiere qualsiasi movimento sul piano orizzontale, ma anche quella di effettuare un movimento (di tipo rigido) sul piano verticale. Quest'ultimo consiste solitamente nell'alzare e abbassare l'$end-effector$, in modo da effettuare un qualsiasi tipo di interazione con l'ambiente circostante.
Nel nostro caso l'$end-effector$ è una matita, attraverso la quale il robot deve essere in grado di tracciare una desiderata traiettoria passante per un numero qualsiasi di punti, nei limiti di un determinato spazio di lavoro (un foglio A4). I punti vengono definiti dall'utente tramite un'apposita interfaccia grafica, direttamente collegata al controllo del manipolatore.

A differenza di un normale SCARA il nostro manipolatore presenta, come citato precedentemente, una base mobile. Questa opzione introduce quindi un grado di libertà aggiuntivo e ridondante, grazie al quale è possibile ottenere configurazioni non singolari alternative. Proprio per questo negli ultimi tempi si è visto crescere notevolmente il numero di manipolatori con un grado di libertà ridondante.

Figura 1.1: Manipolatore Scara[1].

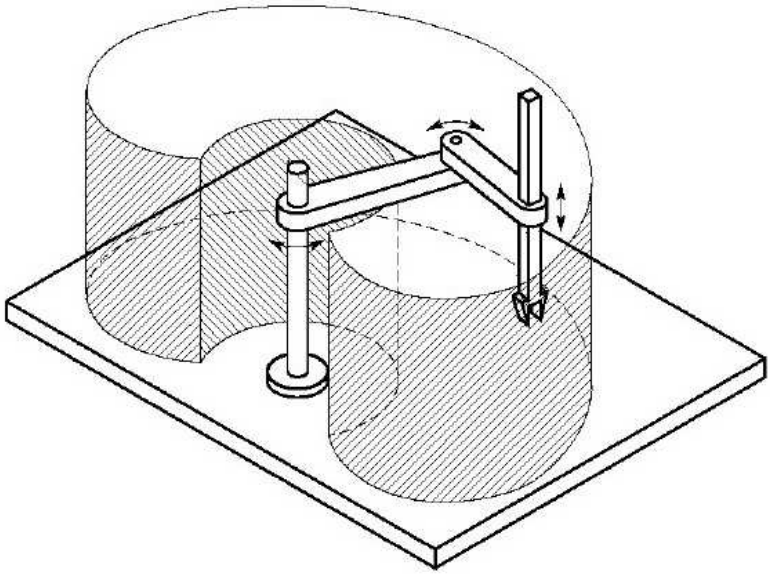


Figura 1.2: Area di lavoro di un manipolatore Scara.

La realizzazione del progetto implica il portare a termine tutte le seguenti fasi:

1. Scegliere, in base ai costi e ai consumi, i motori da utilizzare.
2. Progettare (utilizzando *Power Logic*© e *Power PCB*©) e realizzare i driver dei tre motori e la scheda per interfacciarli con il manipolatore, quindi programmare i microcontrollori.
3. Creare una *GUI* (*Graphic User Interface*) per permettere all'utente di definire i punti attraverso i quali dovrà passare la traiettoria.
4. Effettuare la pianificazione delle traiettorie che il manipolatore deve seguire, per soddisfare le rischieste dell'utente e ottimizzare le prestazioni.
5. Programmare, con Matlab©, il controllo del manipolatore.
6. Realizzare il quadro elettrico, contenente il circuito di emergenza e tutte le parti di elettronica realizzate.
7. Effettuare con *AutoCad*© e *SolidWorks*©, il disegno 3D del robot.

1.1 Meccanica del robot

Per quanto riguarda la parte meccanica, si è deciso di posizionare i motori al di sopra dei rispettivi link. Inizialmente si era pensato di realizzare i membri del robot in acrilonitrile-butadiene-stirene (ABS), ma successivamente si è deciso di utilizzare l'alluminio, poiché essendo più leggero permette delle migliori prestazioni nei movimenti. Il peso dei motori stessi infatti non è trascurabile e utilizzando l'ABS si sarebbe venuto a formare un peso complessivo decisamente troppo elevato.
I due link del robot vengono messi in movimento da due motori in corrente continua; in particolare il motore sul secondo membro è stato scelto il più leggero possibile, sempre per ragioni di carico. Non sono stati utilizzati motoriduttori, perciò il sistema lavora in presa diretta. La posizione assoluta dei due motori in continua viene letta da due encoder incrementali, che fungono quindi da trasduttori di posizione.

Per quanto riguarda il movimento della base si è scelto un motore stepper (passo-passo) e una vite a ricircolo di sfere, che trasformasse il movimento rotatorio del motore in un movimento lungo un unico asse longiudinale. Inizialmente si era pensato di utilizzare una guida lineare dove far scorrere la chiocciola, ma successivamente si è optato per l'impiego di due sbarre cilindriche. In questo caso non è previsto l'utilizzo di alcun encoder, quindi la base mobile lavora in catena aperta.

Per scegliere i motori si è utilizzato il software Matlab©, implementando un programma in grado di calcolare la coppia necessaria al motore per seguire una determinata traiettoria. Pianificando la traiettoria dell'ipotetico caso peggiore (*worst-case*) si è riuscito a determinare la stima della coppia massima richiesta per il movimento. Procedendo in questa maniera in modo decentralizzato (cioè separatamente per ogni link) si è riuscito a scegliere i motori in maniera opportuna. I motori, scelti osservando la coppia di stallo e la coppia massima continua dei vari modelli in commercio, risultano quindi dimensionati in modo otiimizzato per il nostro progetto.

Successivamente si è passato alla realizzazione, con SolidWorks©, del disegno 3D del manipolatore, in modo da determinare precisamente le dimensioni di ogni componente facente parte del progetto.

1.2 Pianificazione delle traiettorie e Controllo del robot

Per quanto riguarda la movimentazione del robot, è assolutamente necessario effettuare la pianificazione delle traiettorie, per fare in modo che siano ottimizzati i consumi e che il manipolatore si muova in maniera corretta seguendo la traiettoria desiderata. Nel nostro caso è stato utilizzato un computer con sistema operativo Windows XP. Si è quindi implementata la pianificazione di moti punto-punto nello spazio dei giunti e, tramite definizioni delle leggi orarie, di moti nello spazio operativo. Nel primo caso le leggi di moto usate sono:

- legge ad accelerazione costante (triangolare);
- legge ad accelerazione costante tagliata (trapezoidale);

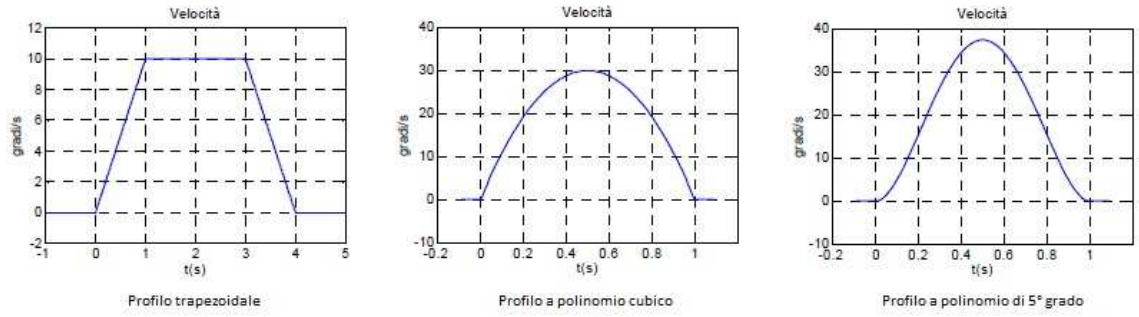


Figura 1.3: Esempi di diagrammi di velocità per varie traiettorie.

- legge a polinomio cubico (di terzo grado) o di quinto grado;

La pianificazione comunica direttamente con il controllo tramite una connessione di rete con protocollo UDP. Si è scelto di usare un protocollo UDP sostanzialmente per motivi legati alla velocità: infatti il protocollo utilizzato più comunemente, il TCP, è di tipo con connessione con riscontro, e quindi è più sicuro dal punto di vista dell'integrità dei pacchetti, ma risulta effettivamente più lento. Infatti con esso è necessario, prima di utilizzare il pacchetto ricevuto, inviare un pacchetto-conferma di avvenuta ricezione. Con l'UDP invece è possibile, dopo averne verificata la correttezza, utilizzare direttamente i pacchetti ricevuti senza dover inviare dei dati di conferma. Nel nostro caso, essendo impiegato per la connessione un cavo ethernet di lunghezza molto ridotta, è corretto supporre l'arrivo dei pacchetti senza gravi perdite di informazioni dovute all'attenuazione della trasmissione dei dati, e perciò il protcollo UDP risulta adattarsi maggiormente alle esigenze del progetto.

Il controllo del manipolatore avviene su un altro calcolatore; per fare in modo che il sistema riesca a lavorare a frequenze di campionamento nell'ordine dei KHz si è scelto di utilizzare il sistema real-time Windows Target di Matlab©, in modo da evitare i ridardi e i problemi di mancanza di velocità dovuti a un eventuale sistema operativo multi-tasking. Per quanto riguarda l'implementazione, il sistema è basato su una macchina a stati, che si occupa di tutte le fasi di controllo. La prima di esse è obbligatoriamente la calibrazione del robot, necessaria poiché come detto in precedenza non vengono utilizzati degli encoder assoluti (che quin-

di sono in grado di conoscere la posizione degli assi a priori), ma degli encoder di tipo incrementale. La fase di calibrazione consiste nel muovere singolarmente ogni membro sino a che esso non giunge ad uno dei finecorsa induttivi, il quale permette quindi di determinare la posizione del manipolatore in quell'istante. Questo procedimento fa sì che il robot non entri in contatto con i finecorsa meccanici, posizionati subito dopo quelli induttivi, i quali determinerebbero un blocco permanente del manipolatore. Precedentemente era stata progettata anche un'altra soluzione (poi scartata), che non prevedeva l'utilizzo dei finecorsa induttivi, poi preferiti soprattutto per questioni di ripetibilità: in questo caso il controllo avrebbe dovuto disabilitare l'elettronica dei finecorsa nella fase di calibrazione, in modo da non mandare in blocco il manipolatore.
Determinata la posizione dei link e della base il robot viene portato nella posizione di post-calibrazione, ossia con la base posta esattamente nel centro della rotaia e i due membri completamente distesi. In sostanza, chiamando θ_2 l'angolo tra i due link del manipolatore, nella posizione di post-calibrazione risulterà $\theta_2 = \pi$ [rad].
Terminata la fase di calibrazione e decisa la traiettoria che si vuole seguire, il controllo riceve dalla pianificazione il riferimento di posizione che sarà necessario raggiungere. Il riferimento può essere dato nello spazio dei giunti oppure nello spazio operativo, a seconda del tipo di implementazione effettuato. Per poter procedere, nel primo caso sarà necessario conoscere la cinematica diretta, mentre nel secondo caso bisognerà trovare la cinematica inversa del robot. Eseguendo la differenza tra questo riferimento (θ_{rif}) e quello ottenuto dgli encoder (θ_{enc}) si ottiene il segnale di errore (e) che dev'essere inviato al controllore.

$$\theta_{rif} - \theta_{enc} = e \tag{1.1}$$

Il controllo può avvenire in due differenti modalità:

a) decentralizzato;

b) centralizzato.

Il controllo di tipo decentralizzato considera ogni link (e la base) un sistema indipendente dagli altri e perciò non influenzato dal movimento degli altri membri

del robot; in ingresso infatti si ricevono le traiettorie separate dei singoli membri. Questo tipo di controllo è più semplice di quello centralizzato, ma in alcuni casi risulta troppo costoso in termini di coppia.
Il controllo centralizzato invece tiene conto del movimento complessivo del robot e perciò, a differenza del caso precedente, riceve in ingresso la traiettoria totale da seguire. Nel caso centralizzato si deve calcolare la dinamica inversa, al fine di determinare la coppia necessaria ad ogni link per portare a termine correttamente il movimento richiesto.

Il controllo comunica inoltre con gli encoder e con i driver dei singoli motori, interfacciandosi tramite la scheda *Sensoray*626[2]. Essa presenta:

- 48 I/O digitali, 20 dei quali con possibilità di gestire gli interrupt;
- 6 contatori a 24 bit;
- Possibilità di selezionare varie modalità (1x, 2x, 4x) per gli encoder incrementali;
- 16 ingressi analogici differenziali (14 bit di risoluzione);
- 4 uscite analogiche (13 bit di risoluzione) con ingressi *remote sense* per compensare l'eventuale resistenza esterna di uscita.

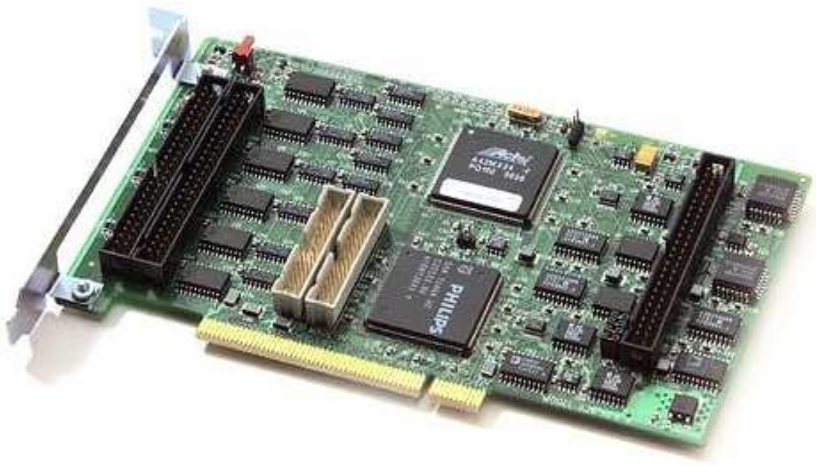


Figura 1.4: Scheda d'interfaccia Sensoray626[3]

1.3 Driver ed Elettronica

Per quanto riguarda la realizzazione della parte elettronica e di azionamento, è necessario partire dallo schema generale del driver.

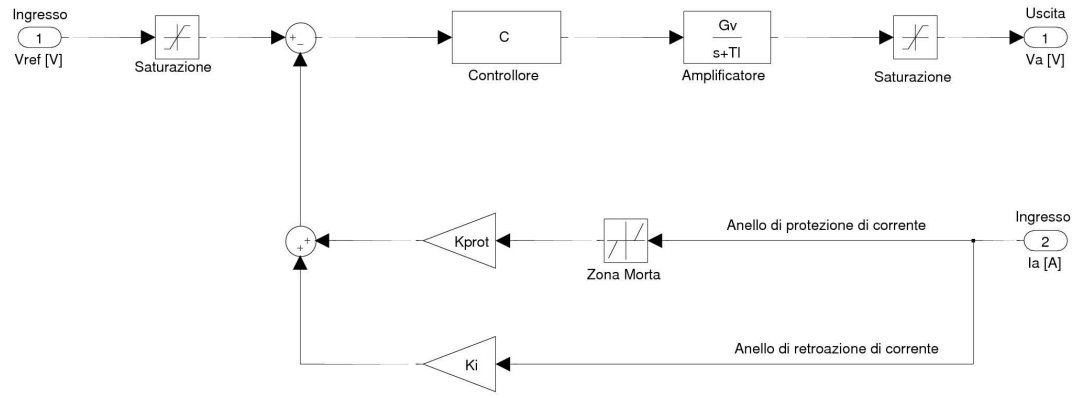


Figura 1.5: Schema simulink© di un driver.

La tensione di riferimento proveniente dal controllo viene inviata ad un primo saturatore, necessario per non superare i limiti di tensione imposti dal driver. Successivamente il segnale di riferimento viene regolato tramite un controllore e quindi amplificato. Anche in questo caso, per simulare i limiti di amplificazione è necessario introdurre un secondo saturatore, avente limiti pari alla tensione di alimentazione. Il segnale uscente può essere inviato e applicato direttamente al motore. Vi sono inoltre due anelli in retroazione, relativi alla corrente: il primo, l'anello di retroazione di corrente, permette tramite il guadagno K_i di confrontare la tensione ai capi del motore con quella di riferimento, generando un segnale di errore che andrà in ingresso al controllore. Il secondo anello, quello di protezione di corrente, entra in funzione se il sistema cerca di superare il limite della corrente massima, oppure se si supera in modo continuativo la corrente massima continua. E' presente infatti una zona morta con estremi variabili, poiché per brevi istanti (nell'ordine dei ms) è consentito superare la corrente massima continua senza bruciare il motore. Per questo brevissimo tempo i limiti della *deadzone* saranno calcolati sulla corrente massima totale, mentre successivamente essi faranno riferimento alla corrente massima continua.

Il driver può essere implementato tramite controllo in tensione oppure con controllo in corrente. Nel caso del driver controllato in tensione, si assume il controllore $C(s) = 1$ e il guadagno $K_i = 0$. Nel caso non dovessero intervenire le saturazioni quindi il driver in tensione si riduce ad un semplice guadagno amplificatore. Esso permette un'ottima reiezione dei disturbi e quindi ottime prestazioni. Per calcolare la capacità di reiezione del driver in tensione è necessario partire dal modello del motore in corrente continua:

$$M(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K_m}{1 + sT_m} \tag{1.2}$$

In particolare, volendo ottenere dei valori numerici, possiamo effettuare i calcoli sul motore 35NT2R82-426SP della *Portescap*, avente costante di tempo $T_m = 12ms$ $K_m = 1/k_v = 19.23$ $[A/Nm]$. Collegando il driver in tensione al motore otteniamo la seguente equazione:

$$\dot{\Theta}(s) = \frac{\frac{G_v}{K_v}}{1 + sT_m} V_{ref} - \frac{\frac{R_a}{K_t K_v}}{1 + sT_m} C_r \tag{1.3}$$

Possiamo quindi calcolare la funzione di trasferimento tra il disturbo e l'uscita. Perché sia ottima la reiezione ai disturbi, bisogna che sia minimo il coefficiente di C_r, e quindi che risulti il più piccolo possibile il fattore $\frac{R_a}{K_t K_v}$, che in questo caso risulta pari a $1.87 \cdot 10^4$. E' un numero molto grande, ma bisogna tenere presente che le coppie di disturbo sono nell'ordine dei mNm, quindi in realtà questo coefficiente comporta una buona reiezione.

Nel caso del driver controllato in corrente invece abbiamo $K_i \neq 0$, poiché questo guadagno permette di convertire la tensione di riferimento nella corrente utile per il comando del motore.

Per calcolare la capacità di reiezione del driver in corrente è necessario partire da un diverso modello del motore in corrente continua, poiché è necessario pilotare il motore non in tensione, ma, appunto, in corrente:

$$\frac{\dot{\Theta}(s)}{C(s)} = \frac{\frac{1}{F_m}}{1 + s\frac{I_m}{F_m}} \tag{1.4}$$

Collegando il driver in corrente al motore otteniamo la seguente equazione:

$$\dot{\Theta}(s) = \frac{\frac{K_t}{K_i F_m}}{1 + s\frac{I_m}{F_m}} V_{ref} - \frac{\frac{1}{F_m}}{1 + s\frac{I_m}{F_m}} C_r \tag{1.5}$$

Prendendo in esame i dati del medesimo motore precedentemente utilizzato, troviamo una costante di tempo di circa $407ms$ (il sistema è quindi più lento rispetto al caso precedente, nel quale avevamo una costante di tempo di $12ms$) e un valore di $F_m = 6,68 \cdot 10^5$, che presuppone quindi una peggior reiezione alle coppie di disturbo.
Il driver controllato in corrente tuttavia risulta molto indicato quando si vuole comandare una precisa quantità di corrente (nota quindi a priori), ancor di più quando è ben noto il modello delle coppie fungenti da disturbo.

Come accennato in precedenza, per quanto riguarda la parte elettronica sarà necessario realizzare:

- una scheda contenente il driver del motore stepper;
- una scheda contenente i due driver dei motori in corrente continua;
- un quadro elettrico, contenente tutte le componenti di elettronica e il circuito di emergenza;
- una scheda di interfaccia che permetta di collegare i due driver dei motori in continua, il driver del motore passo-passo, il circuito di emergenza, il calcolatore dal quale viene effettuato il controllo del manipolatore e ovviamente gli encoder incrementali.

Per comandare ogni driver in corrente continua dev'essere impiegato un microcontrollore, mentre per realizzare lo stadio in potenza è necessario un apposito integrato - l'LMD18200 della *National Semiconductor* - che verrà ampiamente descritto nel Capitolo 2.
Infine per fare in modo che i driver comunichino in modo corretto con il computer dal quale si effettua il controllo del robot è necessario l'integrato MAX232, della *Maxim*. La comunicazione avviene tramite porta seriale.

1.4 Obiettivi del progetto

Per quanto riguarda il progetto appena introdotto, l'obiettivo dello scrivente è stato in particolare quello di progettare e realizzare la scheda contenente i due

driver dei motori in corrente continua. Di seguito troviamo gli schemi di configurazione della parte elettronica con i relativi motori, l'elaboratore da dove si effettua il controllo e il robot:

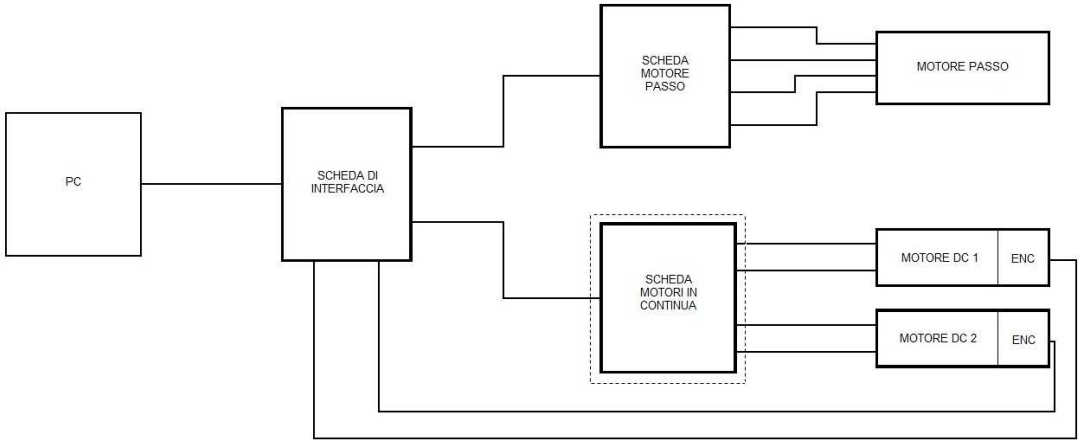


Figura 1.6: Schema di configurazione delle schede - 1

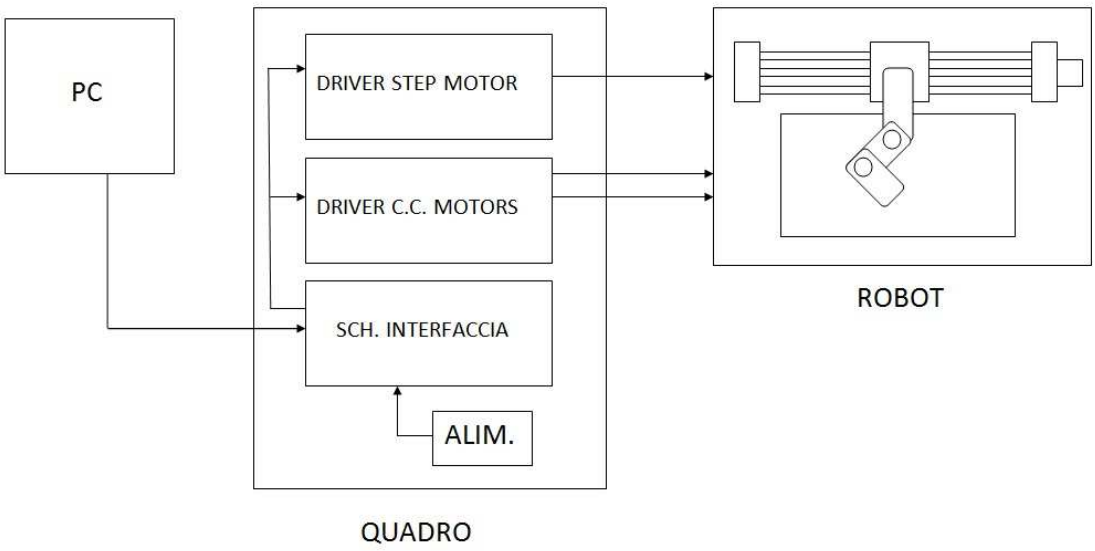


Figura 1.7: Schema di configurazione delle schede - 2

Capitolo 2

Driver di un motore in corrente continua

Il driver del motore in corrente continua ha il compito di tradurre la tensione proveniente dal controllo in un segnale applicabile direttamente al motore. Per far ciò si è implementato un controllo sia in tensione che in corrente. Sulla stessa scheda sono stati realizzati entrambi i driver (uno per ogni motore), anche se poi ovviamente ciascuno riceverà segnali di ingresso differenti, in base alla desiderata movimentazione di ogni asse del robot. Per poter implementare sia il controlllo in tensione che il controllo in corrente, in ogni driver è presente un microcontrollore, nel quale è implementato il software, che assieme ai microcontrollori degli altri assi comunica con un elaboratore tramite la porta seriale. E' necessario inoltre uno stadio in potenza, che piloti correttamente il motore sfruttando la PWM. E' stata garantita una possibile espansione futura del driver, poiché è inoltre presente un connettore che permette la comunicazione tra più microcontrollori.

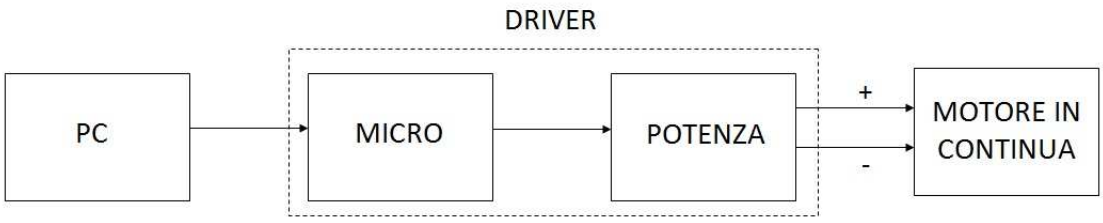


Figura 2.1: Configurazione della scheda

2.1 Microchip dsPic30F4012

Per quanto riguarda il microcontrollore si è scelto un dsPic della Microchip, più precisamente il dsPic30F4012[4,5].

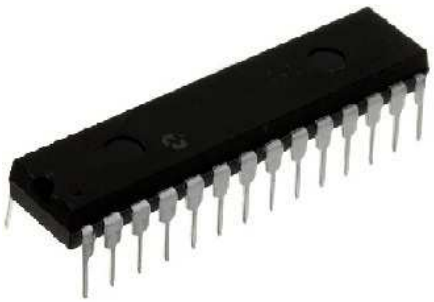


Figura 2.2: MicroChip dsPic30F4012

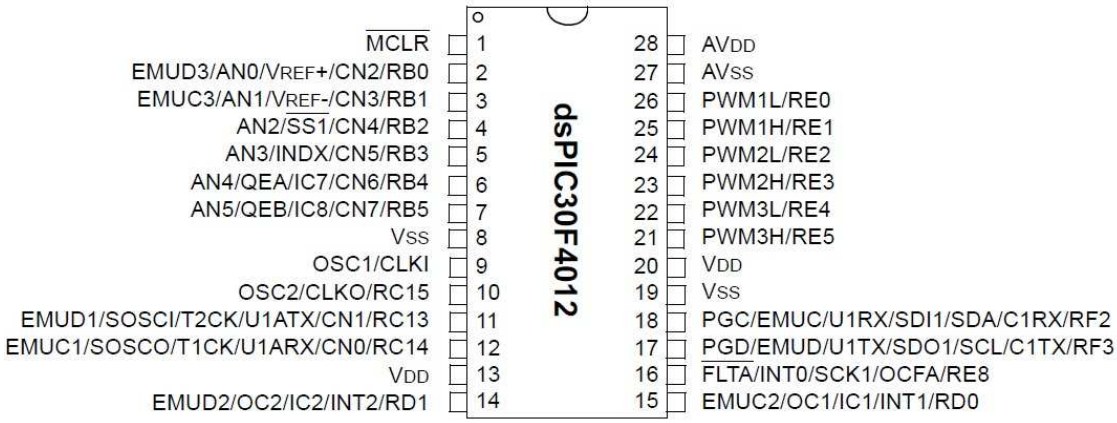


Figura 2.3: MicroChip dsPic30F4012

Esso presenta un'architettura Harvard-modified e quindi, grazie a 4 bus separati può accedere contemporaneamente:

- alla memoria volatile;
- alla memoria non-volatile;
- all'ALU (Arithmetic Logic Unit);
- alle I/O ports.

La CPU è di tipo $High\ Performance\ Modified\ RISC$ a 16 bit, con un *instruction set* di 83 istruzioni. Questo numero ridotto di istruzioni permette di avere maggiori prestazioni.
Per programmare si può usare il linguaggio Assembly, il linguaggio C (la stessa microChip ne fornisce un compilatore) oppure il linguaggio Matlab. Esistono infatti delle apposite librerie Simulink©, di Lubin Kerhuel.
Le altre principali caratteristiche del dsPIC30F4012 sono:

a) 58 Kbyte di Flash Program Memory;

b) 2,048 bytes di RAM;

c) 1 Kbyte di data EEPROM Memory;

d) Possibilità di arrivare alla velocità di 30 MIPs, usando un'oscillatore 4-10 MHz con PLL interna attiva (4x, 8x, 16x), in modo da arrivare a una frequenza massima di 120 MHz;

e) Possibilità inoltre di usare un oscillatore interno con un un clock (meno performante) di 7.37 MHz;

f) 28 pin totali, di cui 20 pin dedicati alle operazioni I/O (scegliendo l'architettura DIP).

I 20 I/O pins presentano le seguenti caratteristiche:

- 20 canali digitali;

- 6 canali analogici con ADC a 10 bit;

- 6 canali PWM Motor Control;

- 1 modulo UART (con *alternate*);

- 1 modulo SPI;

- 1 modulo I2C;

- 1 modulo CAN;

- 1 interfaccia QEI (Quadrature Encoder Interface);

In tutte le porte I/O di input vengono utilizzati Trigger di Schmitt in ingresso, per migliorare la reiezione ai disturbi. Per quanto riguarda le porte I/O digitali, nella figura sottostante troviamo lo schema di funzionamento:

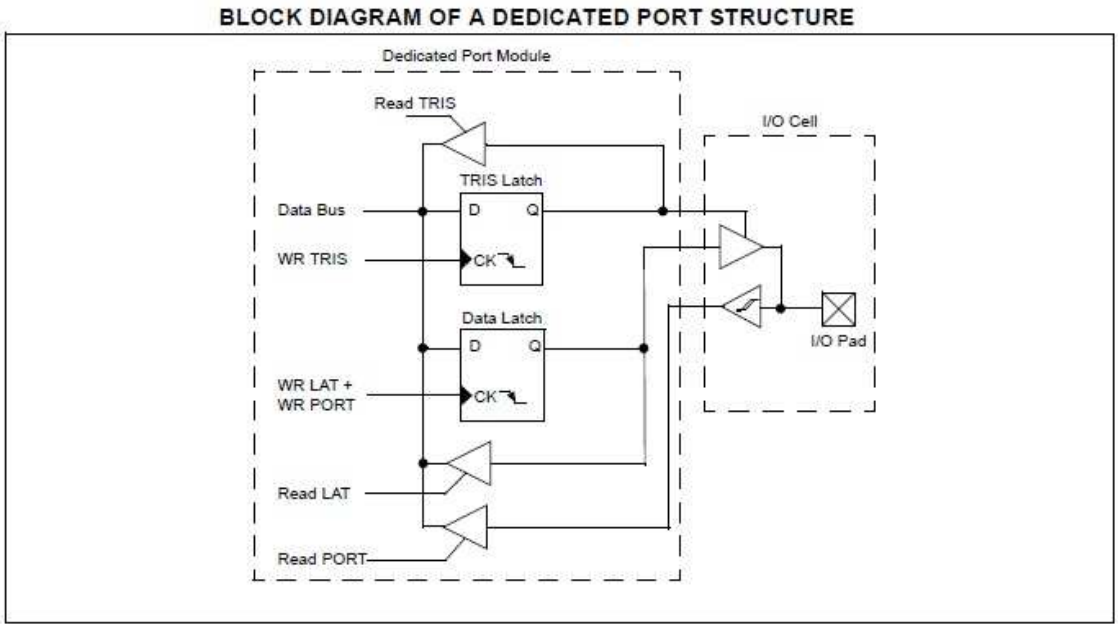


Figura 2.4: Schema di funzionamento dei canali digitali.

Ogni porta digitale ha tre registri associati direttamente al funzionamento del pin:

1. Il Data Direction Register (TRISx) determina se il pin in esame è utlizzato come ingresso (Input) o come uscita (Output). In particolare, se il Data Direction Register ha un livello logico alto ('1') il pin sarà un ingresso, mentre se il TRISx è a '0' la porta sarà un'uscita. Dopo un'azione di Reset tutti i pin sono inizializzati come ingressi, quindi con un livello logico alto;

2. il LAT Register (LATx), che ha due azioni: *read* (legge il *latch*) e *write* (scrive il *latch*). Con l'azione *write* si può imporre un determinato livello logico sul pin desiderato;

3. il PORT Register (PORTx), tramite il quale è possibile leggere lo stato logico effettivo della porta in questione.

Per quanto riguarda invece le 6 porte I/O analogiche, come precedentemente accennato il dsPic presenta un ADC a 10 bit, il cui schema di funzionamento è rappresentato nella figura sottostante:

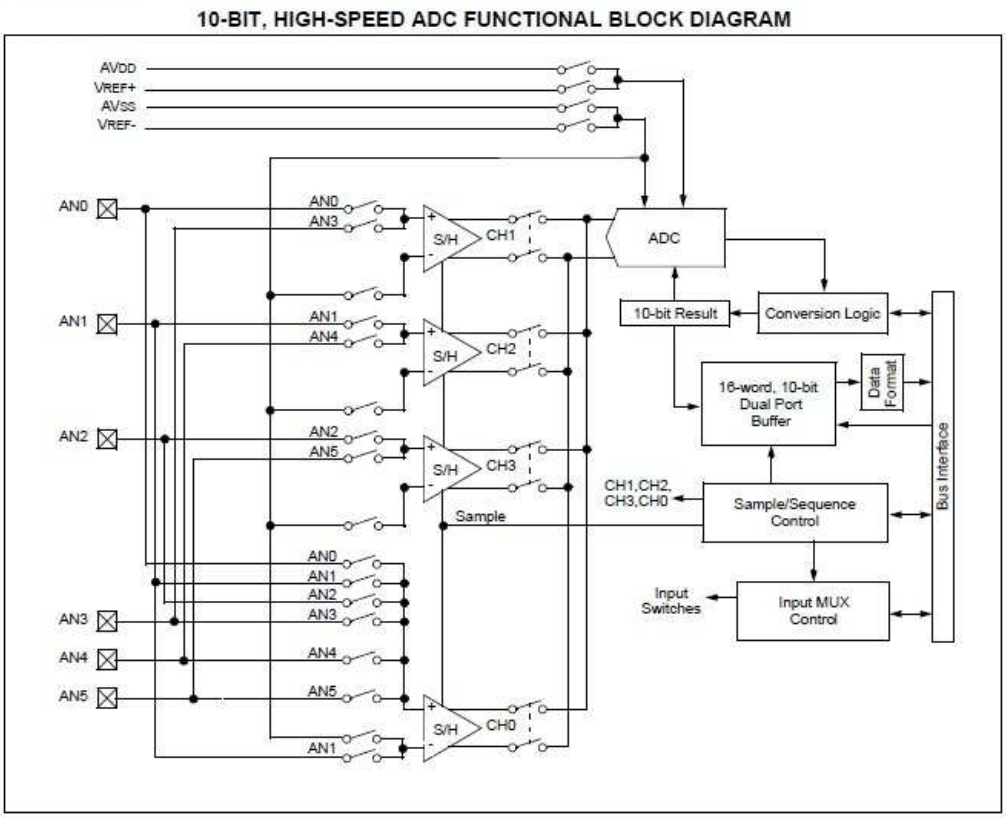


Figura 2.5: Schema di funzionamento dell'ADC.

L'ADC a 10 bit *high speed* permette la conversione di un ingresso di tipo analogico in numero digitale a 10 bit (cioè compreso tra 0 e 1023). Il modulo ADC è basato su un'architettura di tipo SAR (Successive Approximation Register), ed ha 16 *analog inputs* che vengono multiplexati in 4 amplificatori 'Sample and Hold'. Vi sono ben 6 registri da impostare per il funzionamento dell'ADC:

- 3 registri *A/D Control Register* (ADCON1, ADCON2 e ADCON3) che controllano il funzionamento del modulo ADC. In particolare essi stabiliscono la configurazione del formato-dati (es. intero con/senza segno, frazionario con/senza segno, ecc.), determinano quali riferimenti di tensione adoperare per la conversione e controllano il segnale di clock (prescaler e postscaler);

- 1 *A/D Channel Select Register* (ADCHS), che seleziona gli *input channels* che devono essere convertiti;
- 1 *A/D Port Configuration Register* (ADPCFG), che configura i pin come ingressi analogici o come I/O digitali;
- 1 *A/D Input Scan Selection Register* (ADCSSL), che permette di impostare la lettura sequenziale degli ingressi, in modo da selezionare quali di essi sono la leggere o meno.

Per eseguire una corretta conversione Analogico/Digitale bisogna portare a termine i seguenti punti:

1. Configurare il modulo ADC:
   - Configurare i pin analogici, il riferimento di tensione e gli I/O digitali;
   - Selezionare i canali di input;
   - Selezionare la frequenza del clock e di trigger;
   - Attivare il modulo ADC.
2. Configurare gli A/D interrupt (se necessario):
   - Cancellare il bit ADIF;
   - Selezionare la priorità del A/D interrupt.
3. Iniziare il campionamento.
4. Attendere il tempo di acquisizione necessario.
5. Terminata l'azquisizione dei trigger, iniziare le conversione.
6. Quando la conversione è stata completata, attendere che venga settato a '1' il bit DONE del registro ADCON1
7. Leggere il risutato della conversione nel buffer.

Il modulo PWM del dsPic ha le seguenti caratteristiche principali:

a) 6 pin I/O per PWM (PWM1H/PWM1L e PWM2H/PWM2L e PWM3H/PWM3L), con 3 generatori di *duty-cycle*;

b) Risoluzione di 16 bit;

c) Possibilità di variare la frequenza del PWM durante l'esecuzione del programma ('*On-The-Fly*');

d) Sono presenti due diverse modalità di uscite, *Center-Align* e *Edge-Align*;

e) Possibilità per i *Fault Pin* di settare i pin delle uscite PWM a un determinato stato logico.

La base dei tempi del PWM viene determinata da un timer a 15 bit, con *prescaler* e *postscaler*. Si può agire sulla base dei tempi tramite diversi registri:

- il registro PTMR, il cui 15esimo bit è detto PTDIR. In relazione allo stato di questo bit PTMR<14:0> 'conta' in avanti o all'indietro. Più precisamente: se PDIR ha un livello logico basso ('0'), PTMR conta in avanti. Se invece PTDIR è settato a '1' PTMR conta all'indietro;

- il registro PTCON, che permette di abilitare/disabilitare la base dei tempi tramite il set/clear del registro;

- il registro PTPER, che seleziona il *counting period* del PTMR. Quando i valori di PTMR<14:0> corrispondono a quelli di PTPER<14:0> la base dei tempi resettata entrambi a '0'.

La base dei tempi può essere configurata per quattro differenti modalità di funzionamento:

1. Free-Running mode;

2. Single-Shot mode;

3. Continuous Up/Down Count mode;

4. Continuous Up/Down Count mode con interrupt (Double updates mode).

Queste quattro configurazioni possono essere selezionate dalla combinazione dei due bit PTMOD<1:0> nel registro PTCON ('00' = Free-Running, '01' = Single-Shot, '10' = Continuous Up/Down Count e '11' = Double updates).

Se la configurazione scelta è una delle prime due siamo in modalità *Edge-Aligned*; per calcolare il periodo del PWM (T_{PWM})si può usare questa relazione:

$$T_{PWM} = \frac{T_{CY} \cdot (PTPER + 1)}{(PTMR\ Prescale\ Value)} \tag{2.1}$$

dove T_{CY} è il tempo di ciclo per un'istruzione base.
Se invece abbiamo una delle due configurazioni *Continuous Up/Down Count* siamo in modalità *Center-Aligned*; in questo caso la relazione da usare per il calcolo del periodo è la seguente:

$$T_{PWM} = \frac{2T_{CY} \cdot PTPER + 0.75}{(PTMR\ Prescale\ Value)} \tag{2.2}$$

Infine, da questi dati è possibile calcolare la risoluzione del PWM:

$$Resolution = \frac{log(2 \cdot T_{PWM}/T_{CY})}{log(2)} \tag{2.3}$$

La differenza tra le due modalità *Edge* e *Center* si può notare confrontando i grafici sottostanti:

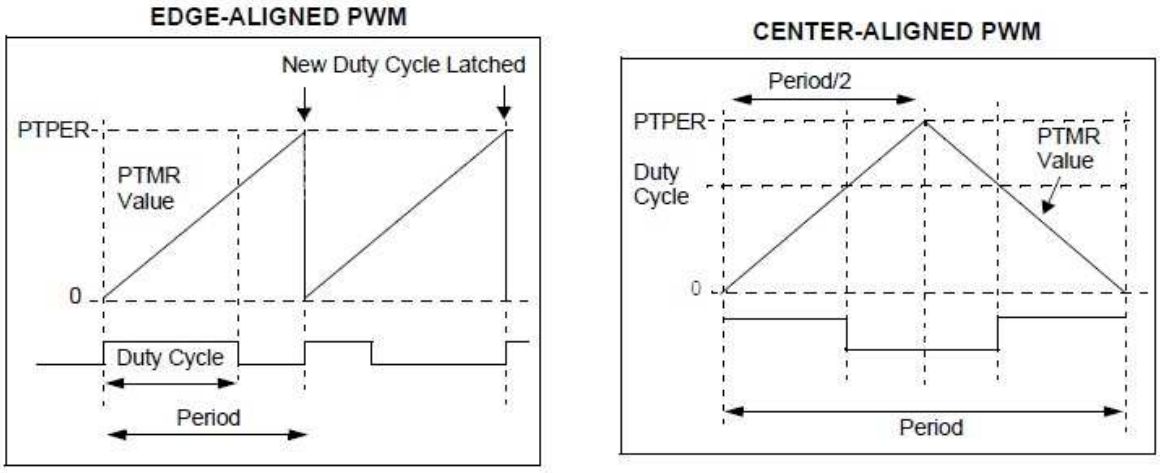


Figura 2.6: Confronto tra le modalità PWM *Edge-Aligned* e *Center-Aligned*

Nel primo caso il PWM viene attivato all'inizio del periodo e viene disattivato quando il valore nel *duty-cycle register* arriva a quello presente in PTMR; nel secondo caso invece il PWM viene settato a '1' quando il valore nel *duty-cycle register* è uguale a quello in PTMR e la base dei tempi sta contando in avanti (PTDIR='1'), e viene resettato a 0 quando i valori nei registri sono nuovamente equivalenti nel mentre che la base dei tempi sta contando all'indietro (PTDIR='0').

I 2 canali UART (1+*altenate*) del dsPic30F4012 presentano le seguenti caratteristiche principali:

- Comunicazione Full-Duplex a 8 o 9 bit, che permette contemporanemente di ricevere e trasmettere dati. Il controller è detto NRZ (Non Return Zero);
- 1 bit per indicare l'inizio del pacchetto e 1 o 2 bit per indicarne la fine (STOP bit);
- Possibilità di impostare un bit di parità (parity bit);
- Interrupt separati per il trasmettitore e il ricevitore.

La configurazione più utilizzata è la '(8,N,1)', con 8 bit di dati, 1 Stop-bit e 0 bit di parità (*No Parity*). E' possibile modificare la configurazione agendo sui registri UxMODE e UxSTA.
L'abilitazione e la disabilitazione del modulo UART avviene agendo sul bit UARTEN, presente nel registro UxMODE.

Nei due seguenti schemi è illustrato il funzionamento del modulo UART.
Sono descritte sia la fase di trasmissione dei dati (si osservi fig. 2.7) che la fase di ricezione dei dati (in fig. 2.8):

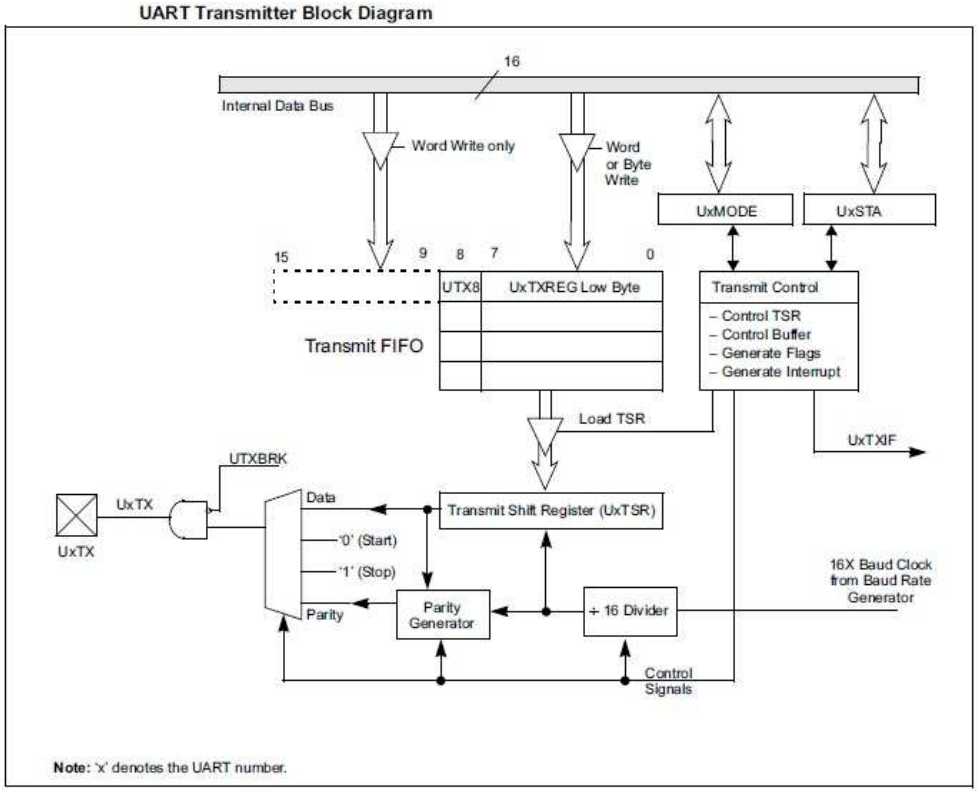


Figura 2.7: Schema a blocchi trasmettitore UART.

Per una trasmissione con 8 bit di dati è necessario:

1. Eseguire il *set up* del modulo UART;

2. Abilitare il UART tramite il bit UARTEN (UxMODE<15>);

3. Settare il bit UTXEN per abilitare la trasmissione;

4. Scrivere il dato che dev'essere trasmesso nel bit meno significativo del registro UxTXREG. Il valore viene quindi trasmesso al $Transmit\ Shift\ Register$ (UxTSR), il quale non viene letto finché non viene spedito lo Stop-bit. A questo punto se nel registro è presente un dato, esso viene caricato e quindi trasmesso;

5. In base al controllo dell'*interrupt control bit* UTXISEL (UxSTA<15>) può essere generato un interrupt.

Lo schema del ricevitore è simile a quello del trasmettitore. Lo troviamo nella seguente figura:

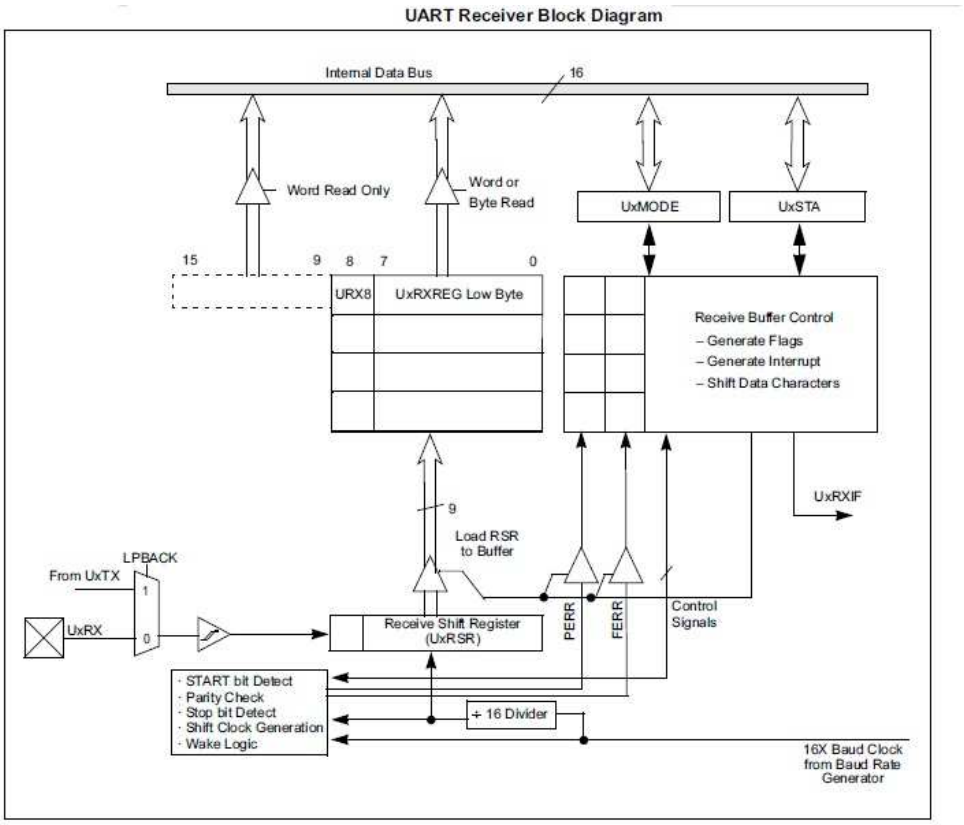


Figura 2.8: Diagramma a blocchi ricervitore uart.

Per una ricezione con 8 bit di dati è necessario:

1. Eseguire il *set up* del modulo UART;
2. Abilitare il UART tramite il bit UARTEN (UxMODE<15>);
3. In base al controllo degli *interrupt control bits* UTXISEL (UxSTA<7:6>) può essere generato un interrupt;
4. I dati vengono ricevuti dal pin apposito e vengono inviati al *Receive Serial Shift Register* (UxRSR). Quando si riceve lo Stop-bit i dati vengono spostati nell'apposito registro FIFO, dove è possibile leggere l'intero byte.

Il modulo SPI è un modulo per la comunicazione seriale di tipo full-dulpex. Esso consiste in un registro a 16 bit (SPI1SR), impiegato per trasferire dati in ingresso e in uscita, e in un registro di buffer (SPI1BUF). Il registro di controllo (SPI1CON) configura il modulo. L'interfaccia seriale è composta da 4 pin:

- SDI1(Serial Data Input) ingresso per il *master* ed uscita per lo *slave*;
- SDO1(Serial Data Output) uscita per il *master* ed ingresso per lo *slave*;
- SCK1, ossia il segnale di clock che viene emesso dal *master*;
- SS1 (Active-low Slave Select), ossia il segnale emesso dal *master* per determinare con quale degli *slave* interagire.

Il modulo I2C è un interfaccia anch'essa seriale, largamente impiegata nel campo della comunicazione tra microcontrollori, display, memorie, ecc. A differenza dell'SPI l'I2C è di tipo half-duplex, e il sistema di comunicazione previsto è di tipo bifilare (*master-slave*). Nonostante esistano ugualmente architetture *multimaster*, di solito ci si trova in presenza di un solo *master* e molti *slave*. Vi sono due canali: l'SDA, dove vengono scambiati i dati, e l'SCL che funge da segnale di clock.

Il modulo CAN è sempre un modulo di comunicazione seriale che viene largamente utilizzato in ambienti rumorosi. Esso è in grado di trasmettere vari tipi di dati (*data messages*). Sono supportati i seguenti tipi di $frame$:

- Standard data frame;
- Extended data frame;
- Remote frame;
- Error frame;
- Overload frame.

Per concludere la breve spiegazione delle interfacce, nel dsPic30F4012 è implementato anche un modulo per la lettura degli encoder in quadratura (QEI). Le caratteristiche di funzionamento del modulo QEI includono:

- 3 canali di ingresso;
- Un contatore di posizione a 16 bit up/down;
- Misure di posizione con modalità di conteggio 2x e 4x;
- QEI interrupts.

2.2 National Semiconductor LMD18200

Per il pilotaggio dei motori è necessario impiegare un ponte H (*h-bridge*). Esso non è altro che una quaterna di interruttori, collegati in modo da pilotare un carico sfruttando la Pulse Width Modulation (modulazione della durata d'impulso). In questa tipologia di driver i 4 interruttori sono posti a formare 2 colonne, ed essi possono essere realizzati tramite transistor bipolari, mosfet di potenza, IGBT, ecc. Il carico (il motore) viene posto nel centro, in modo da collegare i due rami: così facendo la struttura forma un'ipotetica 'H', da cui il nome del circuito.

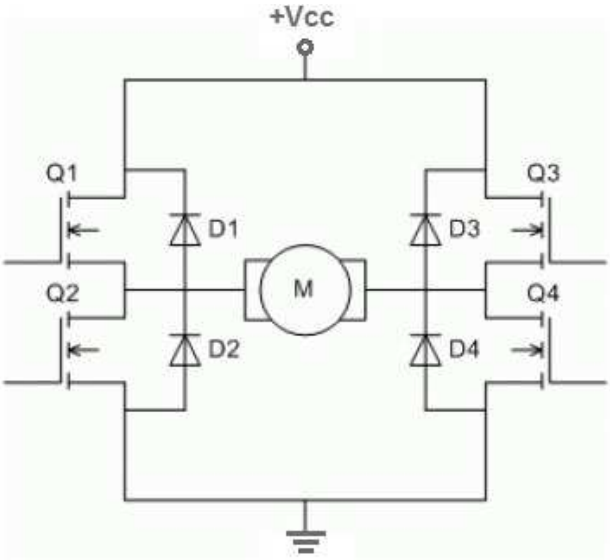


Figura 2.9: Schema di un ponte H[6].

I due transistor inferiori (Q2 e Q4 in figura) sono detti 'di *sink*', in quanto assorbono la corrente proveniente dal motore, mentre i due transistor connessi direttamente alla V_{CC} sono detti 'di *source*'.
A seconda di quali transistor vengono attivati si possono avere diversi percorsi di corrente:

a) Se è attivo un transistor di *sink* ed uno di *source* appartenenti a lati opposti del ponte, si ha passaggio di corrente nel motore, il quale è quindi in rotazione in un verso. La corrente raggiunge il valore di regime in un certo tempo, secondo una curva esponenziale dipendente dalla costante di tempo del circuito;

b) Se è attivo un transistor di *sink* ed uno di *source* appartenenti allo stesso lato del ponte, si ha un corto circuito. Questa configurazione deve essere quindi evitata nel modo più assoluto, in quanto porterebbe alla distruzione del ponte (o dell'alimentatore) in tempi brevissimi. Per evitare questo tipo di problema nei transistor sono presenti degli inverter, che non permettono di cortocircuitare il ramo del ponte;

c) Se tutti i transistor sono spenti, non abbiamo maglie in cui possa passare la corrente fornita dall'alimentatore. Tuttavia bisogna tenere conto dell'eventuale corrente accumulata nell'avvolgimento del motore in continua, il quale è sostanzialmente un induttore, cioè un elemento che tende a mantenere costante la corrente che scorre in esso. Quando un transistor si apre infatti, la corrente dovrebbe andare a zero in modo istantaneo; tuttavia l'induttore tende ad impedire questa repentina diminuzione, provocando inevitabilmente un notevole aumento della tensione sul collettore del transistor (il transistor in questo caso è assimilabile a una resistenza molto elevata, in cui l'induttore tenta di far passare una corrente: per la legge di ohm, V=RI quindi la tensione cresce inevitabilmente). Arrivando facilmente a centinaia di volt, tale tensione (detta 'di fly-back') danneggerebbe irrimediabilmente il transistor stesso.
Per evitare questo fenomeno distruttivo vengono inseriti in parallelo alla

bobina del motore quattro diodi, che forniscono alla corrente una via alternativa a quella del transistor, nel momento in cui questo si apre. E' necessario che questi diodi siano molto rapidi nei passaggi di stato (interdizione/conduzione e conduzione/interdizione), e che siano in grado di sopportare correnti di svariati ampere.
Terminata la scarica dell'induttore non si ha più passaggio di corrente, e se il motore era precedentemente in moto si arresta lentamente a causa degli attriti meccanici;

d) Se è attivo almeno uno dei transistor di *source* e nessuno di quelli di *sink* anche in questo caso non vi sono percorsi in cui passa la corrente fornita dall'alimentatore. La differenza rispetto alla situazione precedente è il sostanziale cortocircuito che si viene a creare ai capi del motore: infatti la tensione ai capi del motore è pari alla tensione diretta del diodo sommata a quella di conduzione del transistor.
L'effetto risultante è una vigorosa azione frenante.

La scelta dello stadio in potenza ha portato all'impiego dell'integrato LMD18200[7,8], della *National Semiconductor*.

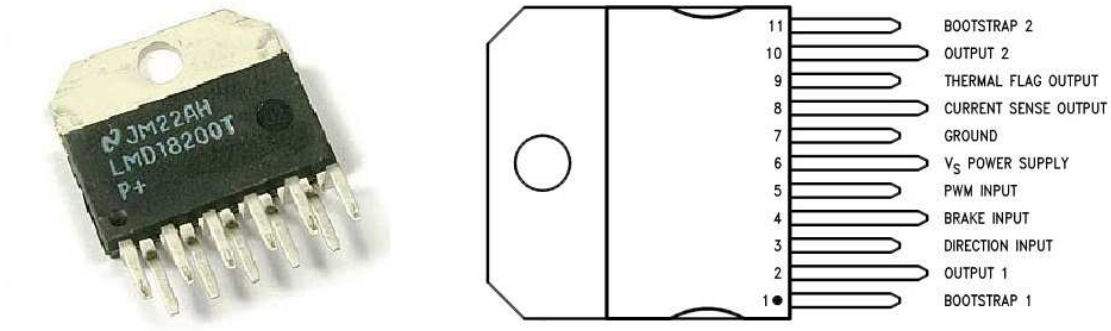


Figura 2.10: LMD 18200.

Le principali caratteristiche che esso presenta sono:

- Possibilità di raggiungere una corrente massima continua di 3A e una corrente massima (di picco) di 6A;

- Tensione massima di alimentazione consentita pari a 55V;
- Compatibilità sia con la tecnologia CMOS che con la tecnologia TTL;
- Funzione *thermal flag*, che quando la temperatura supera i 145°C setta ad un livello logico alto il relativo pin;
- Funzione *thermal shutdown* (che porta a off le uscite) se la temperatura supera i 170°C;
- Possibilità di lettura della corrente, dal piedino *current sensing*;
- 3 ingressi:
  - ingresso DIR, che determina la direzione del flusso della corrente e quindi il verso di rotazione del motore;
  - ingresso BRAKE, che offre la possibilità di frenare elettronicamente il motore;
  - ingresso PWM, che riceve il segnale utile per pilotare il motore.

Nella seguente figura troviamo lo schema logico dell'LMD18200:

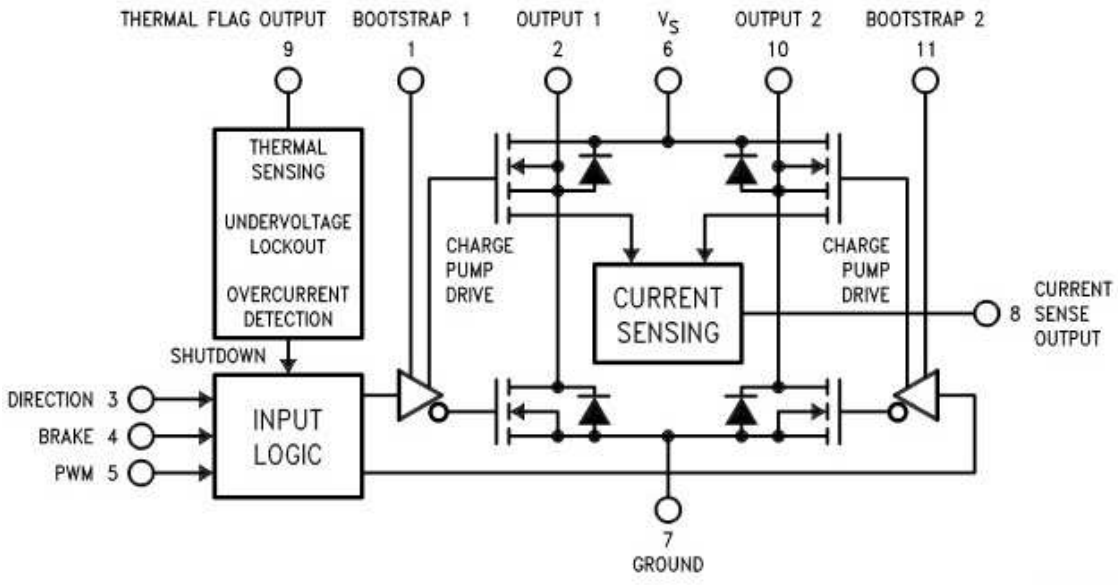


Figura 2.11: LMD18200 - Schema logico.

Questo integrato permette di interfacciarsi con vari tipi di segnali PWM. Per il pilotaggio del motore, le forme più diffuse sono le seguenti:

- *Locked anti-phase* PWM, che consiste nell'inviare il segnale modulato al piedino DIR (pin 3) e inviare al piedino PWM (pin 5) solamente un segnale di *enable* (livello logico alto). In questo modo esiste un solo segnale (con *duty-cycle* variabile) che contiene sia le informazioni di direzione che quelle di ampiezza (fig. 2.12). Quando il *duty-cycle* è del 50% il motore risulta fermo, mentre con un *duty-cycle* del 100% e dello 0% è possibile ottenere il motore con una rotazione a velocità massima, nell'uno o nell'altro verso;

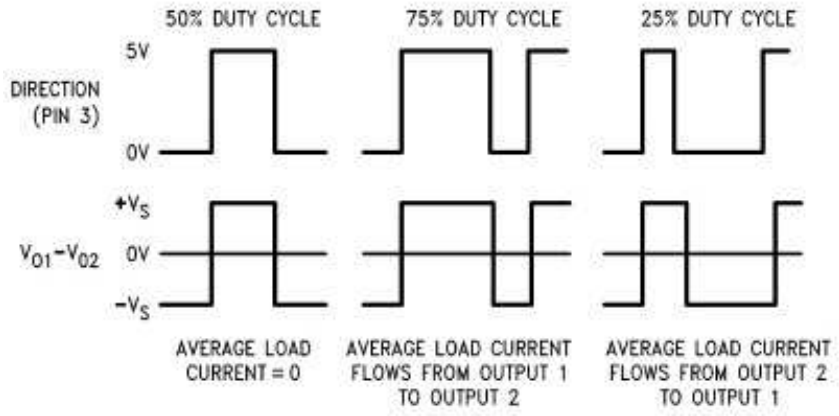


Figura 2.12: Controllo Locked anti-phase PWM.

- *Sign/Magnitude* PWM, che consiste nell'inviare il segnale modulato (valore assoluto) al piedino PWM, ed inviare al piedino DIR un segnale che determini il verso di rotazione della corrente. In questo modo un segnale contiene le informazioni sull'ampiezza (*magnitude*), mentre un altro segnale (*sign*) separato contiene quelle riguardanti la direzione (fig 2.13). In assenza di segnale, ossia con un *duty-cycle* dello 0%, il motore risulta fermo, mentre con un *duty-cycle* del 100% il motore ruota con velocità massima, nel verso determinato in base al piedino DIR.

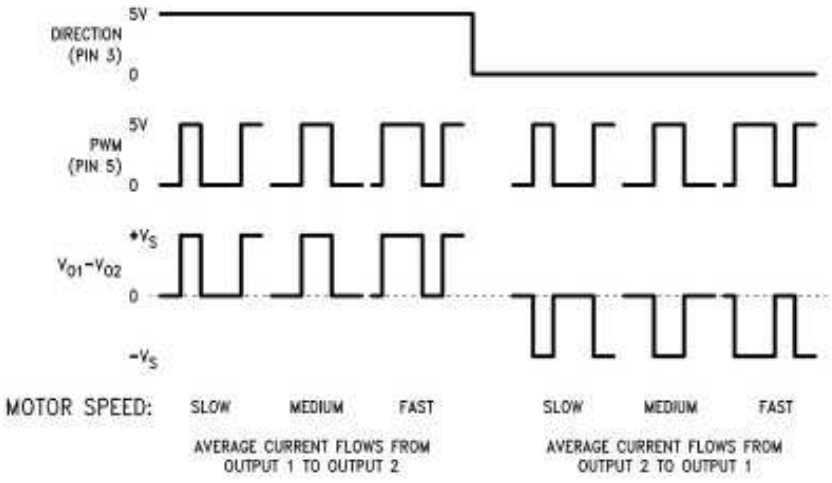


Figura 2.13: Controllo Sign/Magnitude PWM.

L'LMD18200 permette inoltre di effettuare una lettura di corrente dal piedino CURRENT. Il *current sense output* (pin 8) ha una risoluzione di $377\mu A/A$. Per rispettare i limiti di tensione in ingresso del dsPIC ([0 5]V), la tensione del monitor di corrente viene letta ai capi di una resistenza da $2,21k\Omega$, inserita in uscita al piedino CURRENT. Questo valore, sapendo che la corrente di picco dell'LMD18200 è $6A$, può essere calcolato applicando la legge di Ohm:

$$V_{max} = R * I_{max} \longrightarrow R = V_{max}/I_{max} \tag{2.4}$$

Sostituendo i valori numerici si ottiene

$$R = 5/(377 \cdot 10^{-6} \cdot 6) = 2210 \ \ [\Omega] \tag{2.5}$$

cioè proprio il limite massimo di tensione utilizzabile in ingresso al microcontrollore dsPIC30F4012.

Per concludere la descrizione dell'integrato, è d'obbligo considerare che l'LMD18200 è presente in due *packages* differenti.
Per il nostro progetto abbiamo deciso di utilizzare il modello ad 11 pin TO-220, illustrato in figura 2.10.

2.3 Schemi di collegamento

In questa sezione troviamo gli schemi di collegamento del microcontrollore e dello stadio in potenza per il motore in continua, progettati con *PowerLogic*©.

2.3.1 Microcontrollore

Come già detto in precedenza, per il nostro progetto è necessario comandare due diversi omotori in continua. Tuttavia, essendo perfettamente identici gli schemi dei microcontrollori, è possibile analizzarne solamente uno. Lo schema del microcontrollore utilizzato lo troviamo nella seguente figura:

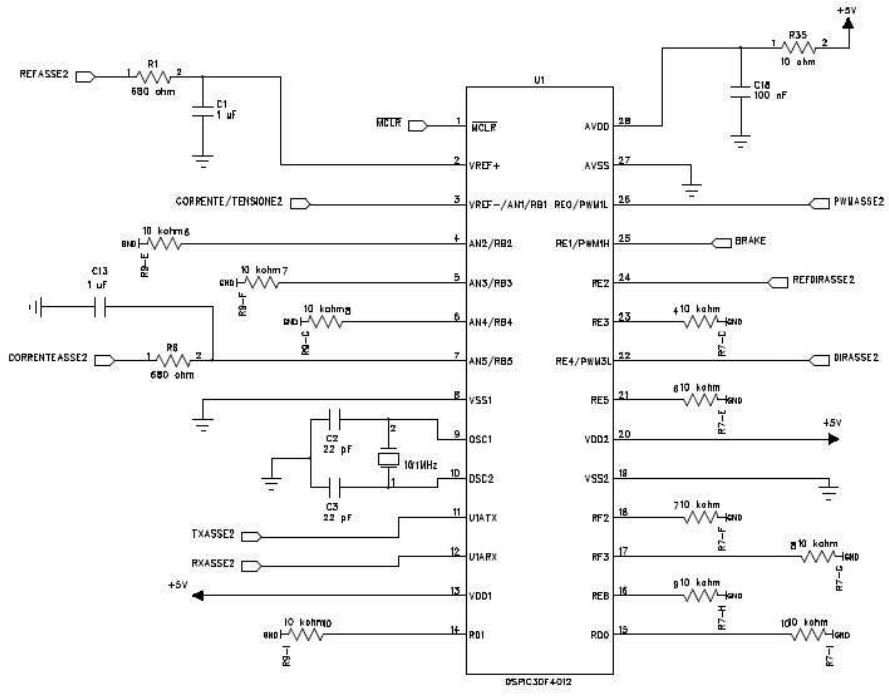


Figura 2.14: Schema del microcontrollore dsPIC30F4012.

1. Il piedino 1 è il cosiddetto MCLR (*Master Clear Reset*) che, essendo un ingresso attivo-basso, permette di resettare il pic se portato a massa. Esso è comune a tutti e tre i microcontrollori (i due per motori in continua più uno per il motore stepper), e nella scheda del driver per il motore passo è stato posizionato un interruttore che permetta il reset di tutti i pic. Dopo una

prima configurazione, poi scartata, si è deciso per un circuito anti-rimbalzo (fig. 2.15), con un filtro passa basso che permetta di eliminare in gran parte le oscillazioni scaturenti dalla pressione del pulsante. E' presente inoltre un diodo in modo da proteggere l'alimentazione dalla tensione di $13.5V$, che viene applicata al piedino quando si programma il microcontrollore.

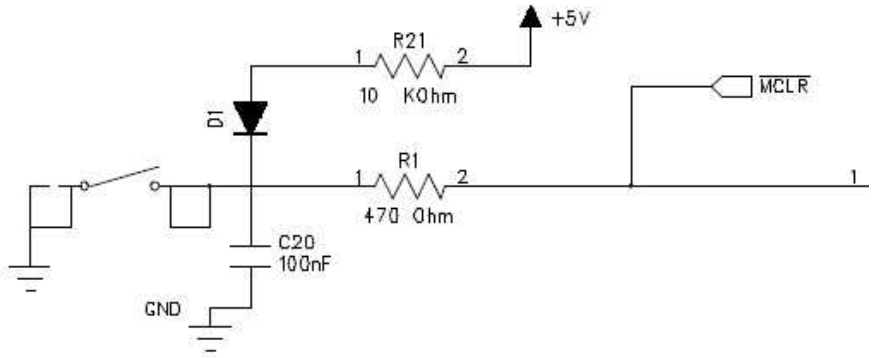


Figura 2.15: circuito del piedino MCLR.

2. Il piedino 2 è adibito alla funzione di ingresso analogico (AN0); ad esso viene inviato (dal controllo) il riferimento di tensione utile alla movimentazione del *link* corrispondente (REFASSEx). La tensione dovrà essere compresa tra 0 e 5 volt, cove $0V$ corrispondono allo stato di quiete e $5V$ alla movimentazione con velocità massima. E' presente inoltre un filtro RC con frequenza di taglio $234Hz$, per eliminare il più possibile le componenti di rumore.

3. Il piedino 3 è usato come ingresso digitale (RB1); esso riceve un segnale digitale alto ('1') se si vuole utilizzare il controllo in tensione, oppure un segnale digitale basso ('0') se si sceglie di utilizzare il driver con un controllo di corrente.

4. Il piedino 7 è usato come ingresso analogico, e più precisamente come monitor di corrente; ad esso giunge il segnale (in volt) contenente il valore di corrente in uscita dal piedino CURRENT dell'LMD18200, utile per effet-

tuare il controllo di corrente. Anche in questo caso è presente un filtro passa basso, con gli stessi parametri del piedino 2.

5. Ai piedini 9 e 10 è collegato un circuito che permette di generare il clock; per far ciò, come spiegato nel datasheet del dsPic30F4012, si utilizza un cristallo al quarzo con frequenza compresa tre 4 e 10Mhz come oscillatore. La frequenza poi potrà essere moltipilcata per un fattore 4,8 o 16, grazie ad uno stadio PLL. Il tempo di *start-up* dipende da molti fattori, come la qualità e la frequenza del quarzo, il rumore, il *rise-time* della tensione di alimentazione e il valore dei condensatori. Per quanto riguarda quest'ultimo, nel data-sheet è indicato di scegliere due condensatori con valori compresi tra $15pF$ e $33pF$. Se da un lato l'aumento di questi valori influisce negativamente sulle prestazioni aumentando il tempo di *settling*, d'altra parte un valore maggiore fa rende il sistema più stabile. La nostra scelta è caduta su due condensatori da $22pF$, ossia un ottimo compromesso tra velocità e stabilità.

6. I piedini 11 e 12 (ingressi/uscite U1ATX e U1ARX) sono collegati al circuito per la comunicazione tramite porta seriale.

7. Il piedino 22, DIRASSEx, può essere usato come uscita PWM (PWM3L) oppure come uscita digitale (RE4), a seconda che si voglia utilizzare rispettivamente la configurazione *locked-antiphase* oppure *sign/magnitude*. A seconda di questa scelta dipenderà ovviamente anche il tipo di segnale da inviare, se un'onda quadra o un valore logico digitale.

8. Il piedino 24, REFDIRASSEx, viene utilizzato come ingresso digitale (RE2). Il controllo invia a questo piedino un valore logico alto o basso, a seconda della direzione del movimento del rispettivo link. Precisamente, '1' significa 'movimento in avanti' e '0' significa 'movimento all'indietro'. La velocità della movimentazione, come descritto in precedenza, è data dal voltaggio inviato al REFASSEx.

9. Il piedino 25, BRAKE, è normalmente ad un livello logico basso. Il controllo ha la possibilità di settarlo ad '1' e quindi di frenare elettricamente il motore.

Il piedino è collegato automaticamente anche all'LMD18200, colui che mette in funzione la frenata.

10. Il piedino 26, PWMASSEx, può anch'esso essere usato come uscita PWM (PWM1L) oppure come uscita digitale (RE0) a seconda del tipo di controllo che si vuole ottenere. Nel caso della configurazione *locked-antiphase* esso viene settato costantemente a 1, mentre nel caso *sign/magnitude* ad esso viene mandato un segnale di tipo onda quadra, con *duty-cycle* variabile.

11. Per concludere, ai piedini 27 e 28 sono collegati i riferimenti di tensione per l'ADC. E' presente un filtro passa-basso con frequenza di taglio di circa $16KHz$, per ridurre al minimo il rumore sui $+5V$ di alimentazione.

2.3.2 Stadio in potenza

Lo stadio in potenza è basato sull'integrato precedentemente descritto, l'LMD18200. Lo schema logico è il seguente:

Come già precedentemente accennato, questo integrato supporta una tensione di alimentazione da 12 a $55V$, permette una corrente massima continua di $3A$ ed una corrente di picco di $6A$.
Analizziamo nel dettaglio lo schema di fig. 2.16:

- tra il piedino 1 e il piedino 2, come tra il piedino 10 e il piedino 11, è presente un condensatore da $10nF$. La capacità di *bootstrap* ha la funzione di *charge-pump*, ossia ha il compito di pilotare il gate dei DMOS interni all'LMD. Per collegarsi al motore sono stati impiegati dei connettori a morsetto.

- il piedino 3, DIR, determina la direzione del flusso della corrente e quindi il verso di rotazione del motore. Ad esso è collegato un diodo-LED verde (tramite una resistenza) che, in base al tipo di controllo utilizzato, risulterà acceso, spento o lampeggiante alla frequenza di PWM impostata.

- il piedino 4, BRAKE, offre la possibilità di frenare elettronicamente il motore. Ad esso è collegato un diodo-LED rosso, ad indicare lo stato di emergenza del motore in frenatura. La frenata elettronica agisce quando il PWM è settato a '1', altrimenti il motore cesserà il suo movimento solo per inerzia.

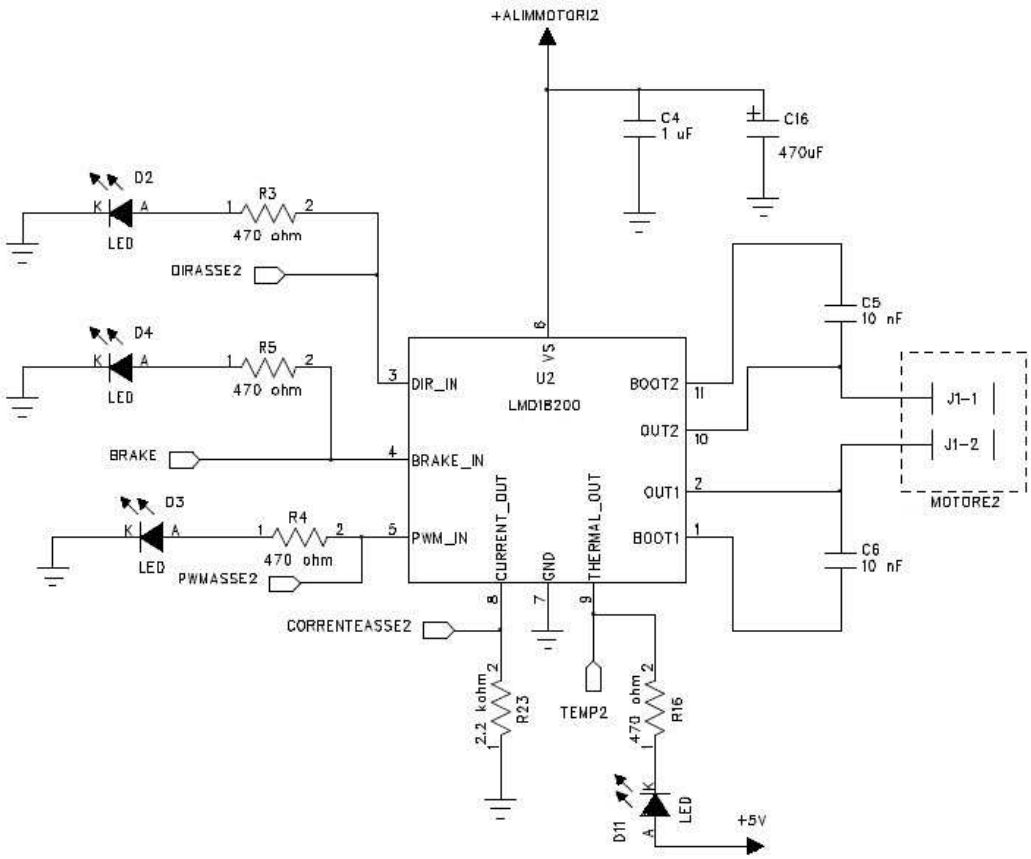


Figura 2.16: schema logico dello stadio in potenza

- il piedino 5, PWM, riceve anch'esso, come DIR, un segnale utile a pilotare il motore (in base al tipo di controllo utilizzato). Il diodo-LED corrispondente è di colore verde.

- il piedino 6 V_s, è il pin di alimentazione. Per ridurre il più possibile il rumore, in modo da ricevere un'alimentazione il più possibile 'pulita', si è scelto di inserire un condensatore a monte del piedino.

- il piedino 8, CURRENT, permette la lettura della corrente presente nell'avvolgimento del motore, fornendo una tensione proporzionale alla suddetta corrente circolante. Per ogni *Ampere* di corrente presente nel motore vengono forniti in uscita $377\mu A$. Per leggere il più correttamente possibile il monitor di corrente è necessario usare una resistenza di precisione, del valore di 2210Ω (calcolato a pag. 29).

- il piedino 9, THERMAL, fornisce la lettura sulla temperatura. In particolare, essendo un pin attivo-basso, esso presenta un livello logico basso quando la temperatura dell'integrato supera la temperatura critica di $140°C$. Comunque, in caso di superamento della temperatura massima consentita (cioè $170°C$), l'integrato si spegne automaticamente in modo da non danneggiare le componenti collegate ad esso.

2.4 I^a versione software: controllo in tensione sign/magnitude

Per fare in modo che il driver agisca in modo corretto, è necessario realizzare un software che permetta al microcontrollore di portare a termine le operazioni richieste. E' possibile procedere con tre differenti metodi di programmazione:

a) usando codice assembly. Questo tipo di programmazione è sicuramente il più performante, ma d'altra parte, oltre a dover conoscere perfettamente l'architettura interna del microcontrollore, il codice risulta poco comprensibile e poco portabile;

b) usando codice C: vi sono infatti vari compilatori a disposizione per programmare microcontrollori in questo linguaggio. Tuttavia esso risulta meno performante del codice assembly, nonostante sia più leggibile e portabile.

c) usando Simulink©. Pur essendo meno performante dell'assembly, il software in Simulink© è sicuramente il più comprensibile. Vi sono a disposizione delle apposite librerie scaricabili da Internet che permettono di programmare i microcontrollori. Tuttavia, a differenza dei casi precedenti, nel caso si volesse utilizzare questo linguaggio, è bene tenere presente che la versione gratuita delle librerie permette di creare e compilare programmi utilizzando fino a 6 ingressi/uscite digitali. Questo limite fa si che questo linguaggio si applichi in maniera brillante solo a programmi con limitate funzioni.

Dopo un'approfondita analisi, si è notato che gli I/O necessari al nostro scopo erano proprio 6. Di conseguenza per i motori in continua si è deciso di program-

mare i microcontrollori utilizzando Simulink©.
Per far ciò è stato necessario scaricare dal sito di Lubin Kerhuel[9] le librerie *dsPIC Toolbox* per *Matlab*©*R*2010*a*, nonché installare il compilatore C30 della *MPLAB distribution*, il tutto gratuitamente.

La prima versione del software prevede l'implementazione del solo controllo in tensione per il driver, utilizzando un controllo PWM di tipo *sign/magnitude*. Il programma è il seguente:

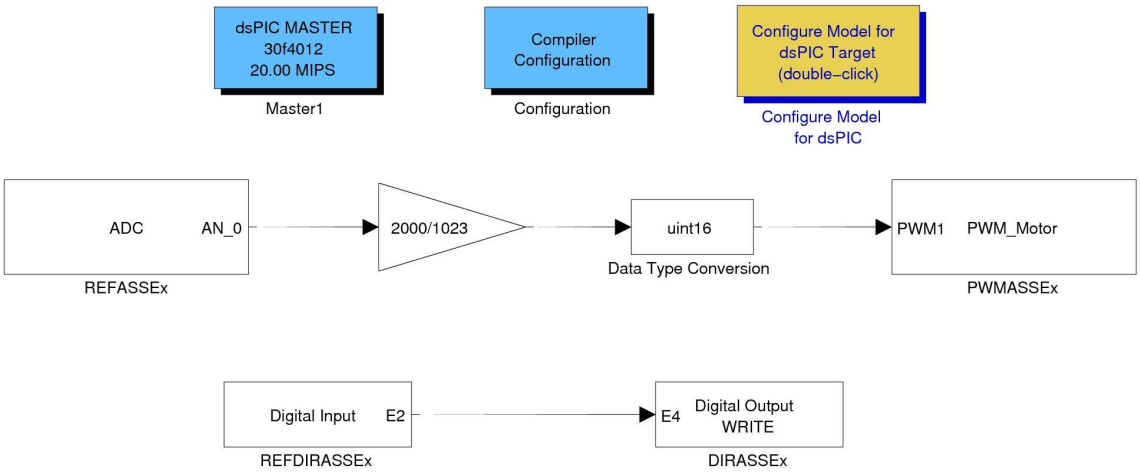


Figura 2.17: Controllo in tensione *sign/magnitude* per driver in continua.

Consideriamo prima di tutto i blocchetti per la configurazione del modello:

- Il blocchetto *Master* permette di configurare il tipo di *dsPic* che si intende utilizzare e tutte le sue opzioni di configurazione. Nel nostro caso è necessario impostare l'MCLR a *enable*, l'oscillatore XT (cristallo) a $10MHz$, il tipo di PLL utilizzato a $PLL - 8x$ e il numero di istruzioni per secondo a $20 \cdot 10^6$.

- I blocchetti *Compiler configuration* e *Configure Model for dsPic target* permettono di configurare correttamente il modello per il compilatore usato (tramite doppio click).

- Il blocchetto ADC permette di configurare correttamente il convertitore analogico-digitale. Nel nostro caso esso viene impostato a 10 bit di risoluzione, con *sample time* pari a quello della simulazione (tipicamente 10^{-5} secondi). E' necessario anche impostare il corretto canale di lettura; nel caso in considerazione l'ingresso analogico in questione è $AN0$, cioè proprio $REFASSEx$.

- Il blocchetto PWM_Motor permette invece di configurare la frequenza desiderata e il canale PWM del microcontrollore adibito alla funzione. Nel caso in questione la frequenza scelta è $20KHz$ e il canale è il $PWM1L$, ossia proprio $PWMASSEx$, collegato al piedino PWM dell'LMD.

- I blocchetti $DigitalInput$ e $DigitalOutput$ permettono di identificare quali canali digitali si stanno utilizzando.

Analizziamo ora il programma vero e proprio:
Per quanto riguarda la *magnitude*, il software parte dalla lettura del riferimento di tensione dall'ADC. Quest'ultimo, avendo una risoluzione di 10 bit, convertirà automaticamente il riferimento di tensione (compreso tra 0 e 5 volt) in un numero compreso tra 0 e 1023. Inoltre dal datasheet è possibile ricavarsi il valore massimo del PWM, che con una frequenza di $20KHz$ risulta 2000. Ciò significa che in ingresso al bloccheto PWM_Motor sarà necessario avere un numero intero (a 16 bit) compreso tra 0 e 2000. Al segnale uscente dall'ADC viene infatti applicato un guadagno proporzionale alla scalatura, e successivamente viene convertito - tramite il blocchetto *data-type conversion* - in un intero a 16 bit. La u davanti a *int* sta ad indicare che è stata scelta una conversione in un intero privo del bit con le informazioni sul segno.
Ora resta da decidere se il movimento dev'essere effettuato in un verso oppure nell'altro. Per far ciò, viene letto il valore digitale proveniente dal controllo al piedino 24 del microcontrollore (RE2), e viene scritto nell'uscita digitale RE4, collegata al piedino DIRECTION dell'LMD.
In questo modo il controllo in tensione del driver agisce correttamente, secondo la configurazione *sign/magnitude*.

2.5 II^a versione software: controllo in tensione e corrente

La prima realizzazione del software, nonostante fosse perfettamente funzionante, non permette di sfruttare al meglio le potenzialità del microcontrollore. Per questo si è deciso di implementare anche un controllo in corrente, dimodoché sia possibile scegliere ad ogni movimentazione che tipo di controllo utilizzare.
Per implementare questo nuovo tipo di controllo è necessario utilizzare il monitor di corrente dell'LMD18200, ossia il segnale di tensione uscente dal piedino CURRENT SENSING, misurato ai capi di una resistenza di precisione da 2210Ω.
Purtroppo la configurazione *sign/magnitude* non permette una corretta lettura del monitor, poiché restituisce sempre il modulo della corrente misurata. Per questo è necessario utilizzare un controllo PWM *locked-antiphase* il quale, nonostante dimezzi la risoluzione della scala di valori, permette di ricavarsi il segno della corrente. Ricordiamo che la configurazione *locked-antiphase* prevede di mantenere sempre ad un livello logico alto il piedino PWM dell'LMD18200 e di inviare al piedino DIRECTION un segnale di tipo onda quadra con *duty-cycle* variabile. In questo modo un *duty-cycle* dello 100% corrisponde ad una rotazione con velocità massima in un verso, un *duty-cycle* dello 0% corrisponde ad una rotazione con velocità massima nel verso opposto e un *duty-cycle* del 50% mantiene fermo il motore.

Questo è lo schema Simulink© del programma:

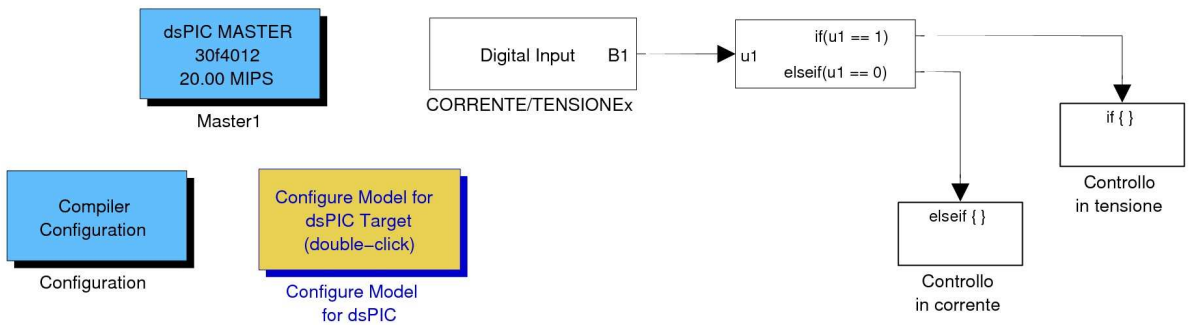


Figura 2.18: Seconda versione software per motore in continua

Come si osserva, il primo step è determinare se effettuare il controllo in tensione o in corrente. La decisione, presa a priori dall'utente che gestisce il controllo del robot, viene passata al piedino 3 del microcontrollore (RB1). Esso, in base all'ingresso digitale ricevuto, seleziona la tipologia di controllo.
Analizziamo prima il controllo in tensione.
L'implementazione è la seguente:

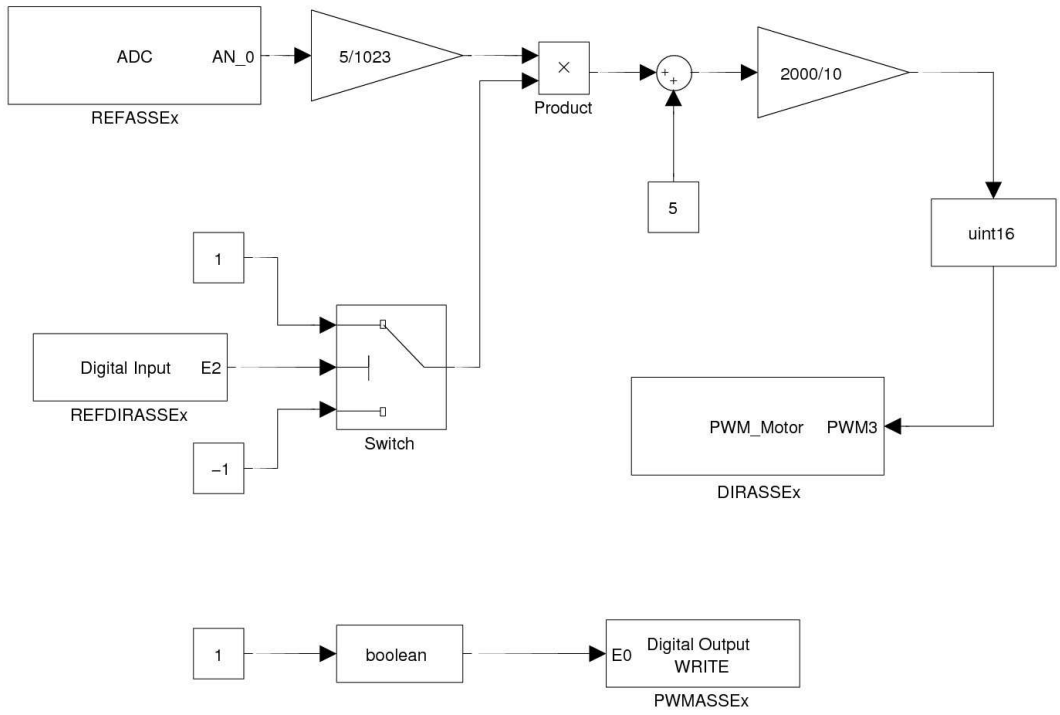


Figura 2.19: Controllo in tensione *locked-antiphase* per driver in continua.

Come si nota, sono presenti alcune sostanziali variazioni rispetto alla versione precedente del software. Dopo la lettura dall'ADC della tensione di riferimento, essa viene riscalata tra 0 e $5V$ (sostanzialmente si riporta il riferimento al valore dato in ingresso dal controllo, come se non fosse intervenuto l'ADC). Il segnale viene poi moltiplicato per il segno della direzione, determinato in base a un blocchetto *switch*: se il segnale di riferimento digitale della direzione è un '1' logico, allora il segnale viene moltiplicato per 1; se invece il riferimento digitale è basso, cioè uno '0' logico, il valore di tensione viene moltiplicato per -1. In questo modo

si possono ottenere valori di riferimento su una scala da -5 a 5. Per trasformare questo valore in un segnale utile a pilotare il PWM è necessario riscalarlo tra 0 e 2000. Per questo si aggiunge una costante 5 al riferimento ottenuto, portando il range a [0 10], e successivamente si moltiplica per un guadagno pari a 2000/10. Così facendo è possibile, tramite un'opportuna conversione a $uint16$, pilotare correttamente il PWM.
Proviamo a verificare il funzionamento del programma con alcuni esempi:

$ex1$. REFASSEx = 5, REFDIRASSEx = 0. Ci aspettiamo un valore di PWM con *duty-cycle* pari a 0, cioè velocità massima di rotazione all'indietro.

$$5 \longrightarrow 1023 \cdot 5/1023 = 5 \cdot (-1) = -5 + 5 = 0 \cdot 2000/10 = 0 \tag{2.6}$$

$ex2$. REFASSEx = 5, REFDIRASSEx = 1. Questa volta ci aspettiamo un PWM pari a 2000 (*duty-cycle* del 100%), cioè velocità di rotazione massima in avanti.

$$5 \longrightarrow 1023 \cdot 5/1023 = 5 \cdot 1 = 5 + 5 = 10 \cdot 2000/10 = 2000 \tag{2.7}$$

$ex3$. REFASSEx = 0, REFDIRASSEx = $0 \vee 1$. Questa volta ci aspettiamo un PWM pari a 1000 (*duty-cycle* del 50%), cioè velocità di rotazione nulla.

$$0 \longrightarrow 0 \cdot 5/1023 = 0 \cdot (\pm 1) = 0 + 5 = 5 \cdot 2000/10 = 1000 \tag{2.8}$$

$ex4$. REFASSEx = 2.5, REFDIRASSEx = 0 . Questa volta ci aspettiamo un PWM pari a 500 (*duty-cycle* del 25%), cioè velocità di rotazione all'indietro pari a metà del valore massimo.

$$2.5 \longrightarrow 512 \cdot 5/1023 = 2.5 \cdot (-1) = -2.5 + 5 = 2.5 \cdot 2000/10 = 500 \tag{2.9}$$

Il programma funziona correttamente. Per fare in modo che la configurazione *locked-antiphase* sia corretta, è necessario inoltre settare a '1' il piedino PWM dell'LMD18200. Per questo si effettua la conversione booleana di una costante '1' e, tramite l'apposito blocchetto *digital output* $WRITE$, si imposta il registro corrispettivo del microcontrollore (RE0).

Analizziamo ora il controllo in corrente. Per semplicità separiamo due stadi:

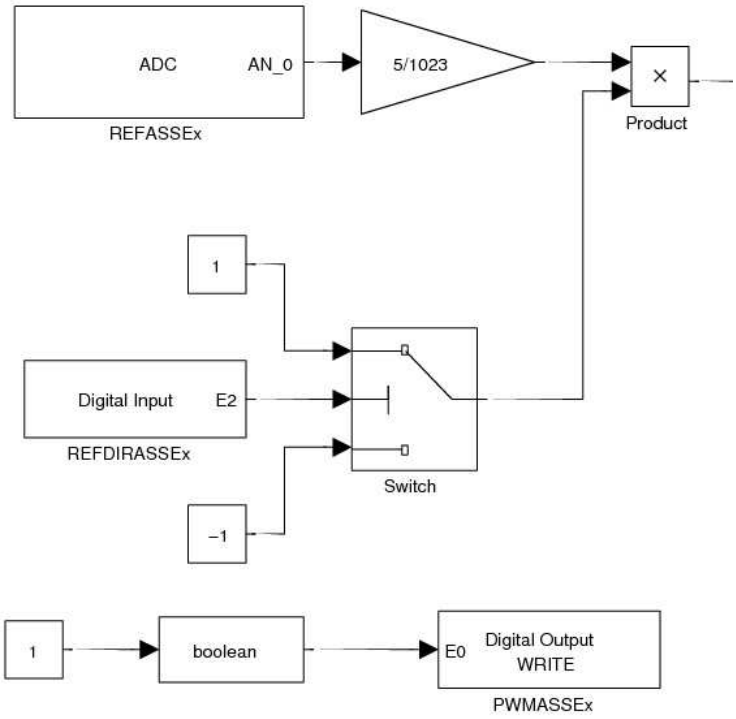


Figura 2.20: controllo in corrente per driver in continua - primo stadio

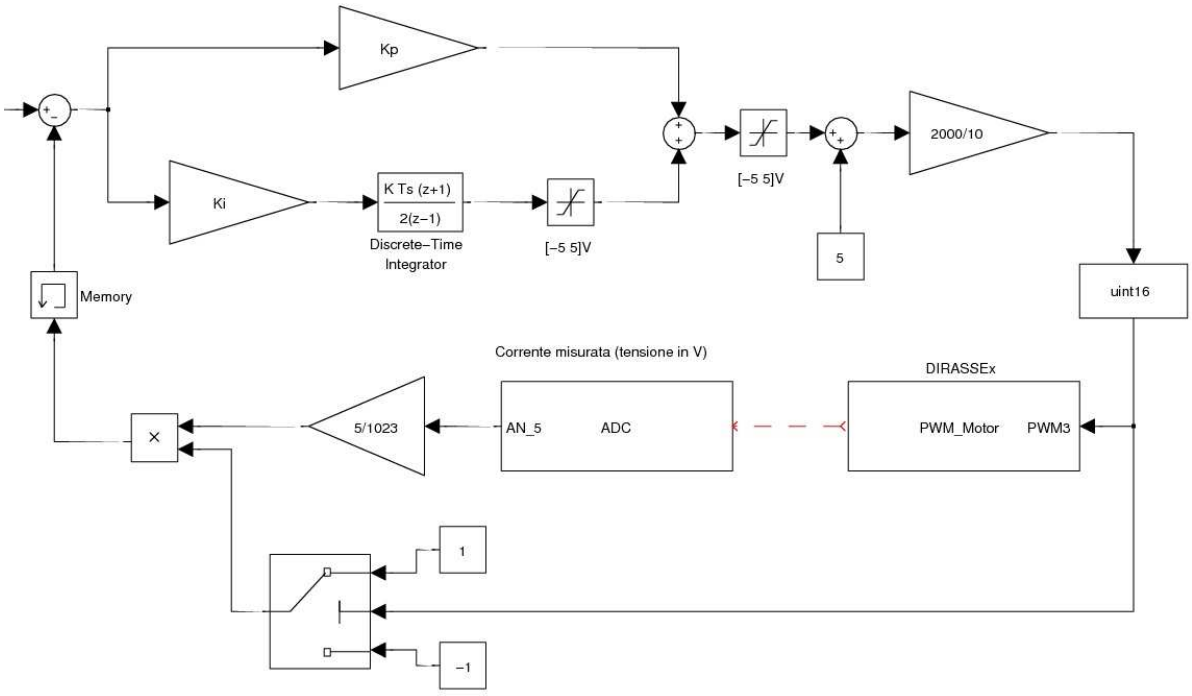


Figura 2.21: controllo in corrente per driver in continua - secondo stadio

- il primo stadio consiste nella lettura dell'ADC, nella scalatura del segnale e nella moltiplicazione per il segno della direzione. Inoltre si provvede a settare ad un livello logico alto il registo RE0, cioè il piedino PWM dell'LMD.
  Il tutto avviene come nel caso precedentemente descritto.

- il secondo stadio (fig 2.22) consiste nel vero e proprio controllo di corrente. Per controllare l'errore si è scelto di usare un regolatore PI con *anti-windup*. Come nel caso del controllo in tensione, in uscita dal blocchetto prodotto abbiamo un segnale di riferimento in volt compreso tra -5 e 5. Ad esso viene sottratto il valore misurato della corrente (sempre in volt), proveniente dall'uscita del piedino 8 dell'LMD18200. E' importante ricordare che dal piedino 8 dell'LMD18200 esce un segnale contenente solo il modulo della corrente, e proprio per questo non era stato possibile implementare il controllo in corrente con il metodo *sign-magnitude*, poiché in caso di diversità di segno tra la corrente di riferimento e quella misurata non si sarebbe potuta effettuare un'azione di controllo corretta.
  Con la configurazione *locked-antiphase* invece è possible risolvere il problema con un semplice blocchetto *switch*, che si può notare in figura: esso infatti permette di determinare qual è il segno della corrente misurata in base al verso di rotazione del motore. Ad esempio, supponendo positivo il verso orario, se il motore gira nel verso antiorario (il che significa un PWM compreso tra 0 e 1000) il valore della corrente misurata viene moltiplicato per -1, viceversa se il motore gira in senso orario (cioè si ha un PWM compreso tra 1000 e 2000) il modulo resta inalterato tramite una moltiplicazione per la costante 1.
  In uscita al sommatore quindi abbiamo il segnale di errore che dovrà essere controllato e portato a 0. Per far ciò è stato scelto di utilizzare un regolatore proporzionale/integrativo, provvedendo ad implementare anche la desaturazione dell'azione integrale (*anti-windup*). Allo scopo di regolare correttamente l'errore in tempi brevi, è necessario un tempo di campionamento di 10^{-5} secondi, poiché la corrente presenta variazioni molto rapide, che causerebbero, con un tempo di campionamento troppo elevato, una continua saturazione dell'azione di comando. I valori scelti per i parametri del

PI sono $Kp = 0.5$ e $Ki = 0.1$. Il metodo di integrazione scelto è quello trapezoidale, ossia secondo la trasformata di $Tustin$.
L'uscita del regolatore, che avrà un *rate* di $[-5\ +5]$ viene successivamente trasformata in un segnale utile a pilotare il PWM nello stesso identico modo utilizzato per il controllo in tensione, ossia aggiungendo una costante pari a 5 e moltiplicando per un guadagno di 2000/10. In uscita al blocchetto *data-type conversion* avremo un intero compreso tra 0 e 2000, che potrà quindi pilotare l'uscita PWM del microcontrollore desiserata (DIRASSEx). In base alla percentuale di *duty-cycle* che viene inviata al piedino DIR la corrente cambierà, avvicinandosi sempre più al valore di riferimento (REFASSEx), sino ad avere un errore pari a 0.

2.6 Layout della scheda

In questa sezione vengono presentati gli schemi integrali della scheda contenente i due driver per i motori in continua. E' presente sia lo schema logico (realizzato tramite $PowerLogic$©), sia il layout PCB (realizzato tramite $PowerPCB$©).
Si può notare, nel layout PCB, la quasi totale simmetria nella disposizione di piste e componenti dei due diversi driver. Questa scelta facilita in modo considerevole la fase di foratura della scheda e quella di disposizione dei componenti, e inoltre, nel caso di non funzionamento o mal funzionamento di uno dei due driver, permette di effettuare un *debug* assai più rapido rispetto ad una scheda dove le piste e i componenti sono disposti in maniera non simmetrica.

Per fare in modo che la comunicazione tra i tre driver e quella tra i driver e la *Sensoray*626 avvenissero in maniera corretta, è stato necessario realizzare una scheda di interfaccia. Essa permette il collegamento con la *Sensoray* non solo tramite porte analogiche e digitali, ma anche tramite gli encoder dei motori. E' proprio su questa scheda che sono stati implementati sia il circuito di alimentazione e stabilizzazione della tensione che quello per la gestione dei fine-corsa induttivi e dell'emergenza.

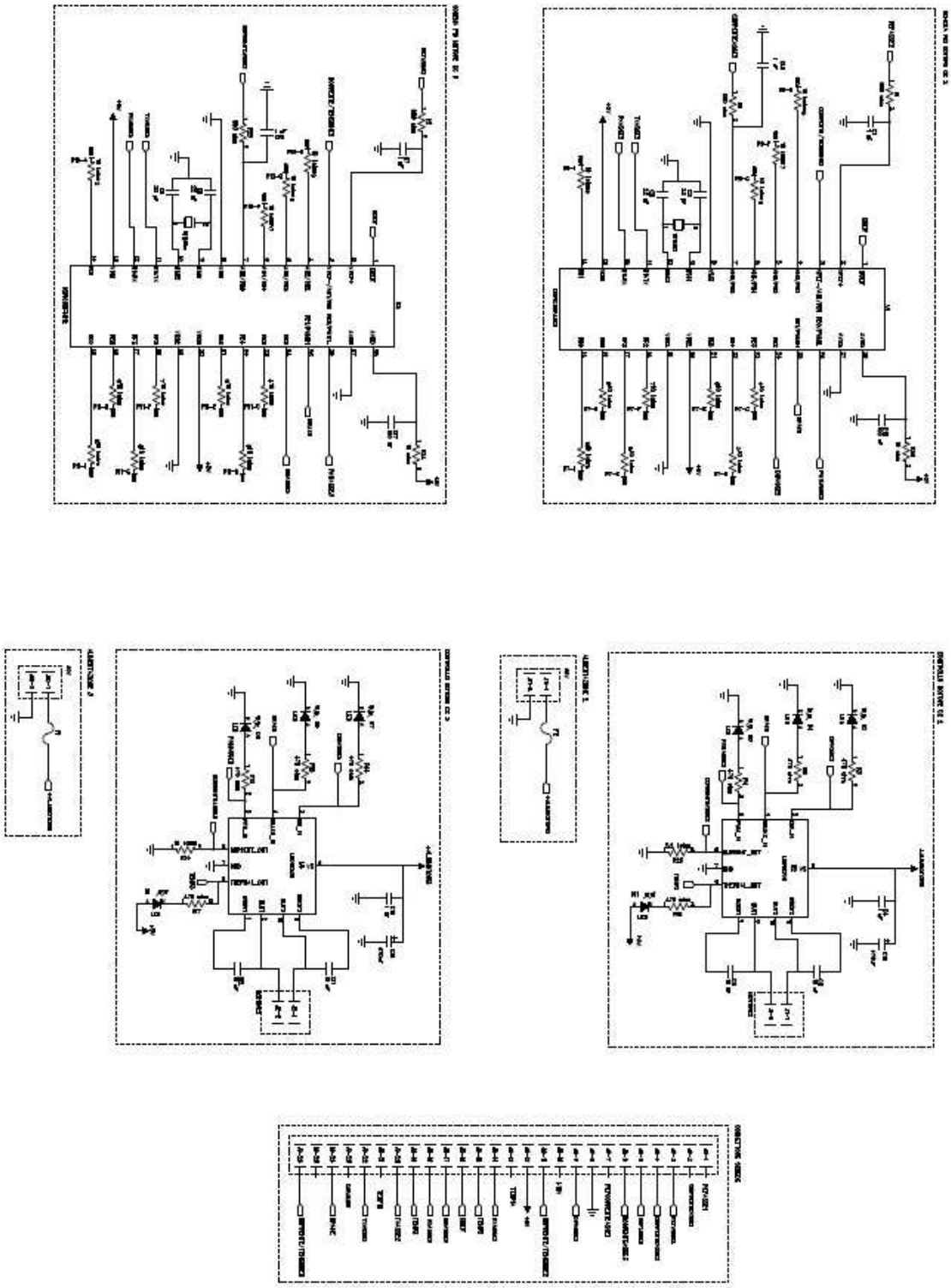


Figura 2.22: Schema logico dei due Driver.

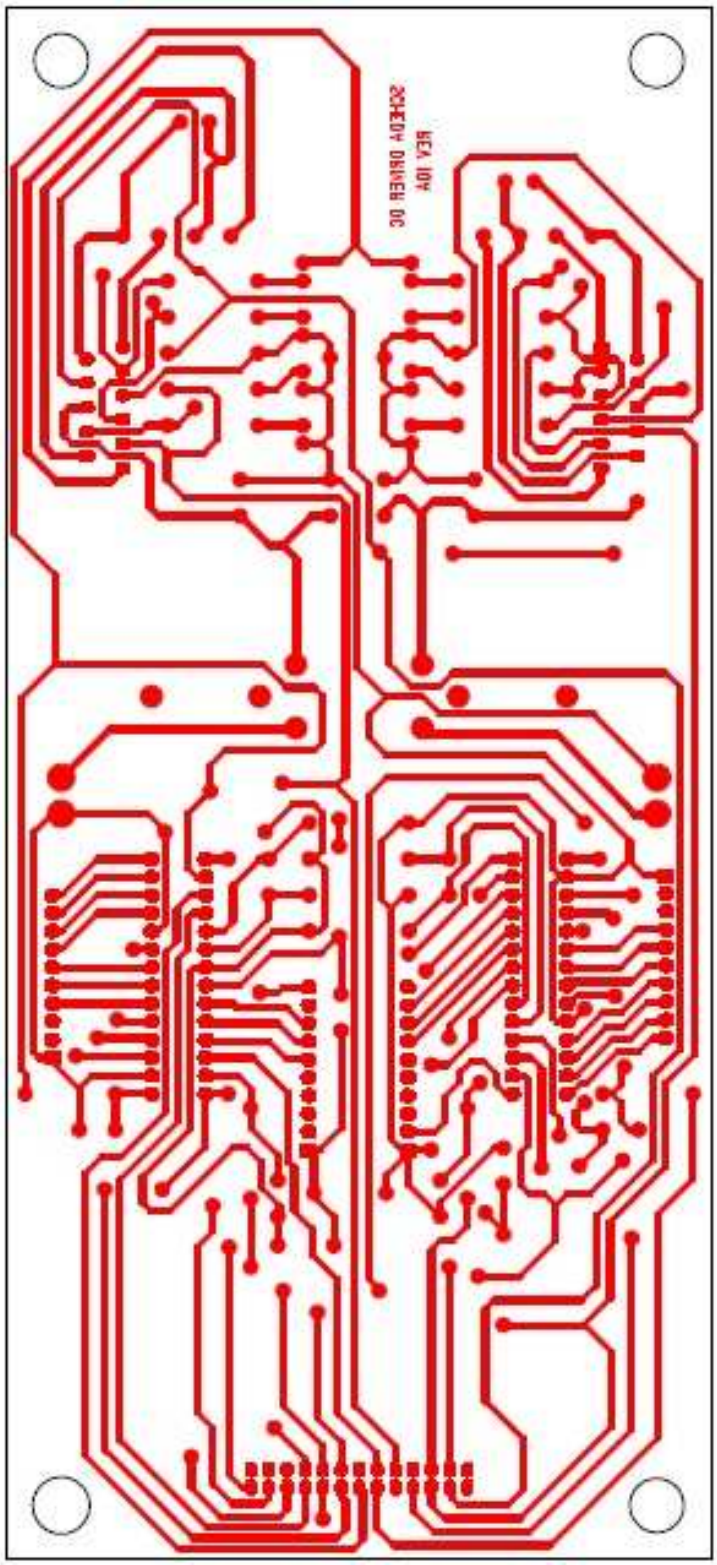


Figura 2.23: Layout PCB - lato bottom.

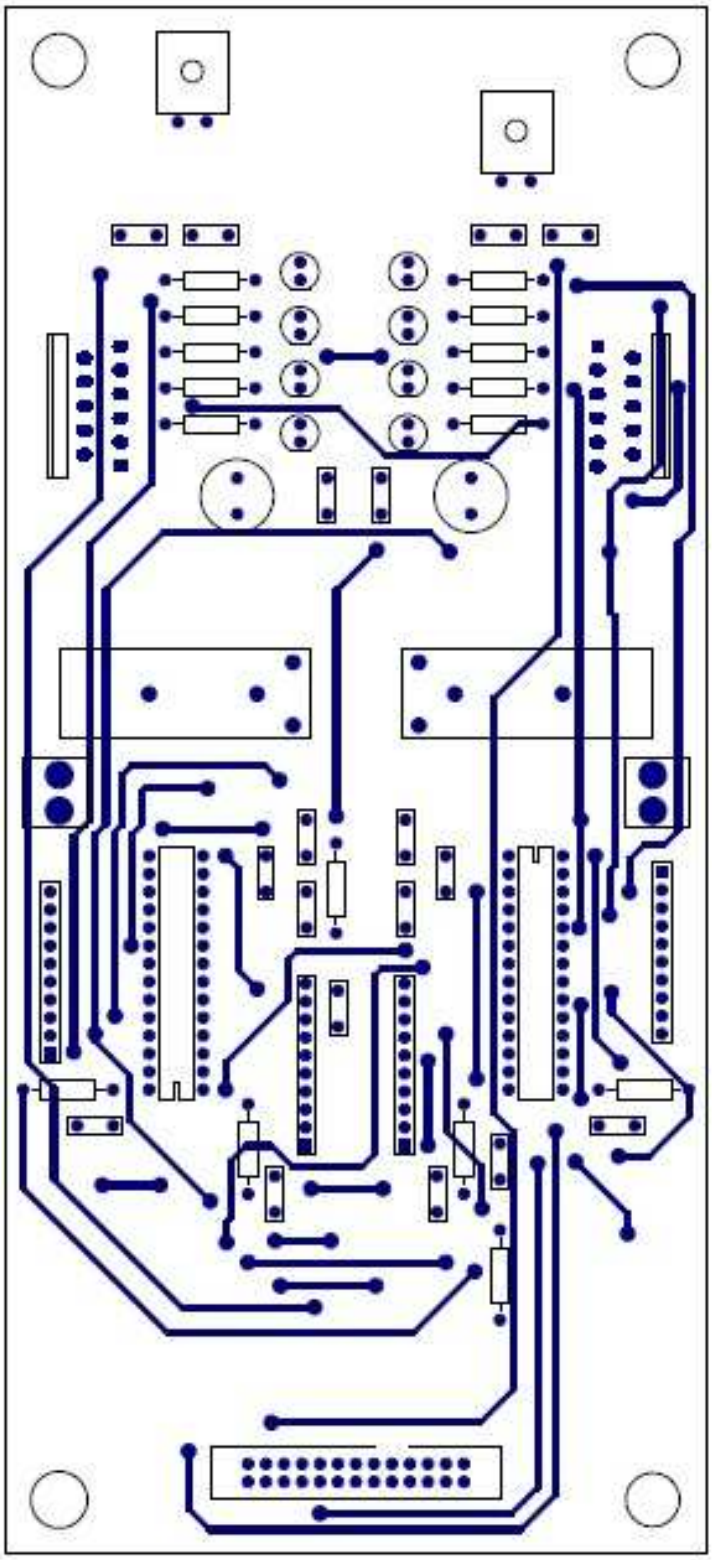


Figura 2.24: Layout PCB - lato top.

La comunicazione tra la scheda d'interfaccia, la scheda contenente i driver dei motori in continua e la scheda contenente il driver del motore passo avviene con un connettore interno a 26 vie (figura 2.22 in basso). Esso inoltre collega tutti gli ingressi e le uscite analogiche/digitali utili a pilotare in modo corretto il motore alla $Sensoray626$.
La scheda dei motori in continua viene quindi alimentata tramite il connettore interno, che la collega alla scheda di interfaccia, alla quale arriva l'alimentazione. Quest'ultima viene garantita e stabilizzata da un integrato apposito, l'LM7805, il quale presenta le seguenti caratteristiche principali:

- Tensione di uscita $5V$ stabile.
- Tensione in ingresso fino da $10V$ a $35V$.
- Corrente in uscita fino a $1A$.
- Protezione sia in caso di cortocircuito che di sovratemperatura.

Lo schema di funzionamento è molto semplice, in quanto consiste solamente nell'utilizzo di due condensatori, posti ai due capi dell'integrato, entrambi del valore di $100\mu F$.

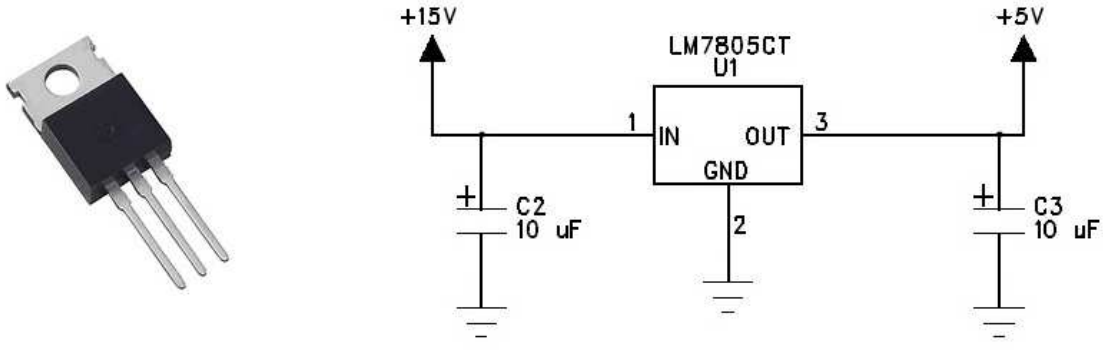


Figura 2.25: LM7805 con schema di funzionamento[10].

Per quanto riguarda l'alimentazione del motore invece, essa arriva a collegarsi direttamente sulla scheda. Per ottenere un'alimentazione consona ai movimenti del robot sono stati acquistati degli alimentatori della Cabur, precisamente i Cabur CSF500D. La tensione nominale in uscita è di $48V$ continui, in realtà regolabile da $30V$ continui sino a $56V$ continui. Grazie a questa opzione è quindi possibile sfruttare al meglio i nostri stadi in potenza, dato che essi supportano una tensione massima di 55V.
Tra i connettori del motore e il piedino di alimentazione è stato inserito un fusibile in modo da preservare il tutto in presenza di correnti troppo elevate.
Il porta-fusibile impiegato supporta fusibili di dimensioni 5x20mm. Sono stati utilizzati fusibili con grado di protezione $IP40$, corrente massima nominale $10A$, tensione massima nominale $250V$.

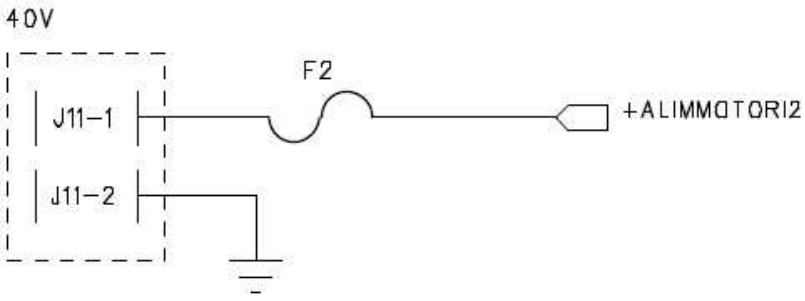


Figura 2.26: Dettaglio schema logico: Alimentazione di uno dei motori in c.c.

Nella progettazione del layout PCB delle tre schede si è il più possibile cercato di mantenere un'identica posizione per il connettore a 26 vie, ossia a lato vicino ad un bordo della scheda. Questa scelta permette di avere una miglior configurazione per quanto riguarda i fili che passano da una scheda all'altra. E' infatti opportuno considerare che una piattina da 26 è un cavo di larghezza non trascurabile, e risulta quindi assai comodo, a castello montato, avere i tre connettori uno sotto l'altro nella medesima posizione.

Nell'osservare il seguente schema logico, è bene tenere presente che ivi sono collegati solo i piedini che fanno riferimento a segnali facenti parte della scheda contenente i driver dei due motori in continua.

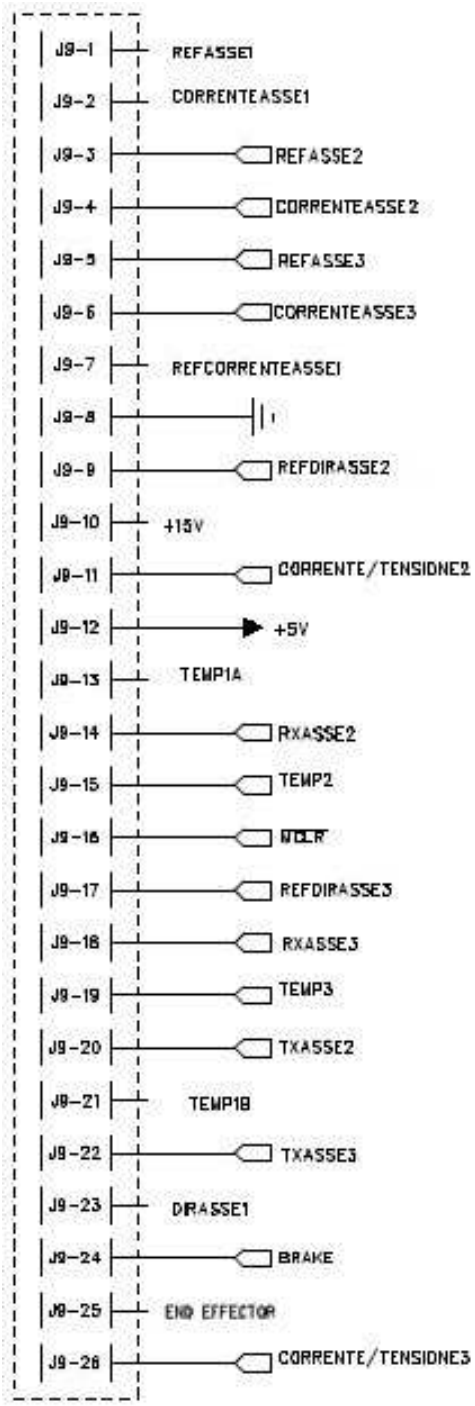


Figura 2.27: Dettaglio schema logico: connettore a 26 vie.

I collegamenti utilizzati in questa scheda sono i seguenti:

- 3-REFASSE2: canale a cui viene inviata la tensione di riferimento utile per il movimento del primo membro del robot.
- 4-CORRENTEASSE2: canale contenente il valore del monitor di corrente riferito al movimento del primo membro del robot.
- 5-REFASSE3: canale a cui viene inviata la tensione di riferimento utile per il movimento del secondo membro del robot.
- 6-CORRENTEASSE3: canale contenente il valore del monitor di corrente riferito al movimento del secondo membro del robot.
- 8-GND: canale di massa per l'intero circuito dei driver.
- 9-REFDIRASSE2: canale contenente il segnale di riferimento digitale per la direzione di rotazione del motore del primo membro.
- 11-CORRENTE/TENSIONE2: canale contenente il segnale di riferimento digitale che comanda il tipo di controllo da effettuare ('1' = controllo in tensione, '0' = controllo in corrente) per il driver del primo membro.
- 12-+5V: canale conenente i +5V comune all'intero circuito dei driver.
- 14-RXASSE2: canale di ricezione del dsPic che gestisce il primo membro del robot.
- 15-TEMP2: canale contenente il segnale di riferimento della temperatura dello stadio in potenza del primo membro del robot.
- 16-MCLR: canale di Master Clear, comune a tutti e tre i dsPic. Se esso cambia livello logico resetta tutti e tre i dsPic.
- 17-REFDIRASSE3: canale contenente il segnale di riferimento digitale per la direzione di rotazione del motore del secondo membro.
- 18-RXASSE3: canale di ricezione del dsPic che gestisce il secondo membro del robot.

- 19-TEMP3: canale contenente il segnale di riferimento della temperatura dello stadio in potenza del secondo membro del robot.
- 20-TXASSE2: canale di trasmissione del dsPic che gestisce il primo membro del robot.
- 22-TXASSE3: canale di trasmissione del dsPic che gestisce il secondo membro del robot.
- 24-BRAKE: Canale di emergenza, che se settato ad un livello logico alto frena elettricamente i motori e pone l'intero sistema in uno stato di emergenza.
- 26-CORRENTE/TENSIONE3: canale contenente il segnale di riferimento digitale che comanda il tipo di controllo da effettuare ('1' = controllo in tensione, '0' = controllo in corrente) per il driver del secondo membro.

2.7 Procedimento di realizzazione delle schede

Dopo aver progettato lo schema logico (con *PowerLogic*©) e il layout delle piste (con *PowerPCB*©) per realizzare fisicamente le schede abbiamo effettuato il procedimento di fotoincisione. I punti principali sono, in ordine cronologico:

1. Stampare ogni lato delle scheda (top e bottom) su un foglio A4 di carta lucida, tramite una stampante laser (il lato superiore deve essere specchiato). Affinché la stampa sia di buona qualità, non utilizzare 'stampa veloce'.
2. Ripetere la stampa quattro o cinque volte sullo stesso foglio di carta lucida, affinché lo strato di inchiostro diventi sufficientemente spesso da non far passare la luce. Durante questa operazione è necessario stare molto attenti a non disallineare il foglio nelle varie stampe, poiché se le piste non sono perfettamente combacianti bisogna buttare via il foglio e ricominciare da capo.
3. Procurarsi una basetta della dimensione desiderata per ogni scheda, quindi appoggiare (fissando con dello scotch) le stampe effettuate sui lati corrispon-

denti della basetta, facendo in modo che quest'ultima risulti perfettamente aderente al foglio di carta lucido.

4. Inserire la basetta in uno scanner apposito (bromografo) per il tempo necessario a impressionare la scheda, che varia a seconda della potenza della lampade UV. Questa operazione permette di eliminare il photoresist nei punti della scheda non coperti dell'inchiostro del lucido.

5. Ripetere tale operazione per ogni lato della scheda. Si suggerisce di ruotare la scheda di 45 gradi quando è trascorso metà tempo di esposizione, in modo da uniformare il più possibile il photoresist.

6. Conclusa la fase di esposizioone, lavare con un panno bagnato d'acqua la scheda. In questa fase è già possibile osservare il formarsi pian piano delle piste sulla scheda.

7. La fase successiva consiste nello scaldare prima un contenitore di cloruro ferrico, quindi immergerci dentro ogni singola scheda per un tempo limitato. Questo procedimento permette di eliminare il rame non coperto da photoresist, in modo da delineare più precisamente e correttamente le piste. E' importante non lasciare troppo tempo le schede nel cloruro ferrico, poiché altrimenti si rischia di eliminare anche il rame sulle piste in cui è necessario. E' bene tenere presente che il cloruro ferrico è un sale di ferro dell'acido cloridrico, ed è un composto tossico e irritante. Per questo è assolutamente necessario prestare la massima attenzione durante l'intero procedimento.

8. Infine, trascorso il tempo necessario (circa 2-3 giorni), quando tutte le piste hanno un colore uniforme è possibile togliere dal bagno ogni scheda e, dopo averla debitamente sciaquata, si può considerare pronta per essere utilizzata.

A questo punto bisogna praticare i fori sulla scheda per fare in modo che tutti i componenti possano essere posizionati correttamente. Per far ciò nel nostro caso si è utilizzato un trapano a colonna, con punte di varie dimensioni. E' buona pratica prima misurare con un calibro i reofori dei vari componenti, quindi praticare il foro con una punta di dimensione maggiore, ma che si discosti il meno possibile

dal valore visualizzato sul calibro. Nel caso infatti si andasse a forare con una punta di diametro troppo grande, è possibile danneggiare il rame del pad creando dei punti di non-conducibilità sulle piste.

Una volta terminato di praticare tutti i fori, si possono inserire i componenti nel lato *top* della scheda.
Comincia ora un'operazione delicata e, se non prese le dovurte precauzioni, rischiosa: la saldatura dei componenti. Gli strumenti utilizzati sono un rotolo di filo di stagno, un saldatore, una stazione saldante e gli occhiali protettivi da indossare per evitare che dello stagno fuso possa danneggiare gli occhi. Per realizzare una saldatura efficiente è buona norma seguire il procedimento qui indicato:

1. Dopo aver inserito tutti i componenti, piegare leggermente i reofori dei componenti in modo che essi non cadano o non si spostino nel momento di capovolgere la scheda.

2. Fissare bene la scheda da saldare alla stazione saldante dopo averla capovolta, in modo da avere il lato *bottom* verso di noi.

3. Scaldare il saldatore fino alla temperatura di $450°C$. Grazie a questa elevata temperatura è possibile fondere lo stagno.

4. Comincia la fase di saldatura vera e propria. Essa consiste in tre azioni da fare rapidamente. In questa fase è assolutamente necessario indossare gli occhiali protettivi.

    a) Appoggiare la punta rovente al pad da saldare per pochi attimi, in modo che poi lo stagno faccia meglio presa. E' importante non scaldare troppo, altrimenti si rischia di danneggiare perennemente il componente. Proprio per questo, per gli integrati più delicati (come il microcontrollore) è buona norma utilizzare uno zoccolo.

    b) Posizionare il filo di stagno tra la punta del saldatore, il pad e il reoforo del componente per pochi istanti. Si osserverà lo stagno fondersi.

    c) Una volta che lo stagno ha coperto in modo uniforme tutto il pad di rame, sollevare la punta del saldatore seguendo il reoforo, modo da creare con lo stagno una superficie concava.

Se la saldatura è stata effettuata in modo corretto, essa dovrà risultare lucida. Se invece si nota lo stagno di un colore opaco, significa che la saldatura è fredda e, se conduce al momento, probabilmente dopo qualche giorno essa non garantirà più la conducibilità dovuta. E' meglio quindi dissaldare le saldature fredde e ripetere la procedura.

Terminata la fase di saldatura su tutti i componenti, una volta verificate le connessioni delle piste con il tester la scheda risulta pronta per essere testata. Nel nostro caso la scheda contenente i driver dei motori in continua è la seguente:

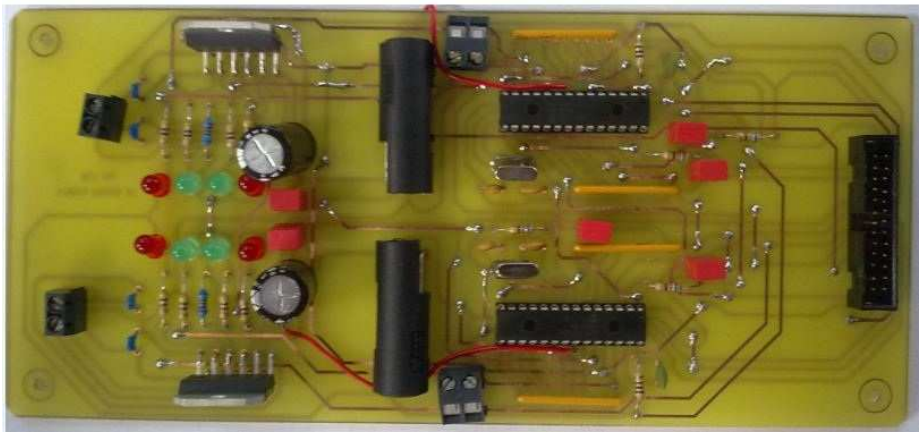


Figura 2.28: Scheda contenente i due driver in continua - lato *top*

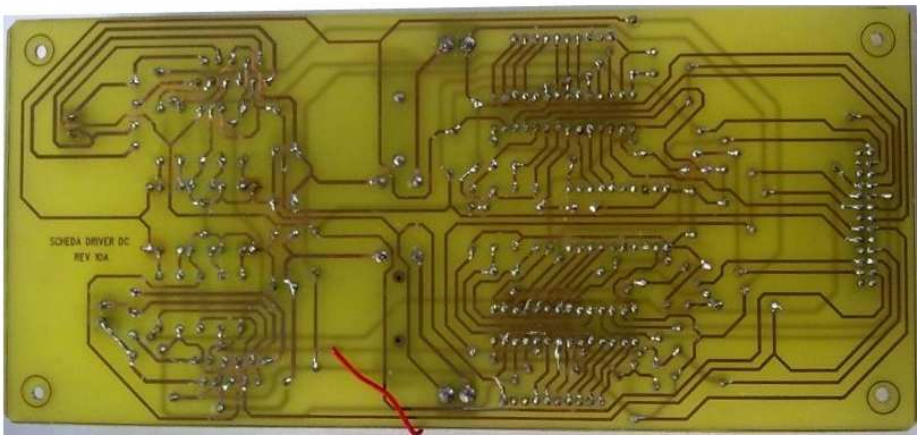


Figura 2.29: Scheda contenente i due driver in continua - lato *bottom*

Capitolo 3

Test sperimentali

3.1 Test sulla scheda di interfaccia

Per poter testare la scheda contenente i driver dei motori in corrente continua è necessario prima verificare il corretto funzionamento della scheda di interfaccia. Sono stati quindi applicati i $15V$ di alimentazione in ingresso e si è andata a misurare, tramite il tester, la tensione di uscita di ogni integrato LM7805, descritto in precedenza.
Una volta verificato che la tensione di uscita (e quindi di alimentazione per le due schede contenenti i driver) fosse proprio di $5V$, si è collegata (tramite le corrispondenti piattine da 50 vie) la scheda d'interfaccia alla *Sensoray*626, in modo da verificare che gli ingressi e le uscite sia analogiche che digitali corrispondessero e funzionassero in modo corretto.
A questo punto, superato anche questo test, è stato possibile testare approfonditamente sia la scheda contenente il driver per il motore passo-passo che quella contenente i driver per i motori in corrente continua. In particolare in quest'ultima sono stati effettuati dei test sia prima che dopo il montaggio del primo link.

3.2 Controllo del motore

Prima di montare i link del robot, stati effettuati i seguenti test sul sistema motore-giunto elastico:

1. Prima di tutto nel microcontrollore è stato implementato un software che fornisse una semplice tensione costante al motore. Il test non ha presentato problemi, in quanto il motore ha cominciato a girare, seguendo il verso di percorrenza desiderato, a velocità costante.

2. Il secondo test prevedeva invece di testare anche il cambio del verso di rotazione del motore. Per questo nel firmware el microcontrollore è stato implementato un programma nel quale, grazie ad un ciclo, fosse possibile verificare tutti i valori di PWM, da 0 (velocità massima in senso antiorario) a 2000 (velocità massima in senso orario). In questo programma il valore di PWM aumenta di 1 ogni millisecondo, dimodoché dopo un secondo il valore sia a 1000 (e quindi il motore si trovi in stato di quiete). Dal comportamento del motore e dal lampeggiamento dei led relativi alle uscite dello stadio in potenza è stato possibile confermare il corretto funzionamento del sistema e quindi la positività del test.

3. Dopo aver implementato un controllo di posizione in Simulink©, comunicante in *Real-time* con il motore, è stato possibile tarare un regolatore PD che permettesse una risposta ad un gradino di 60° molto performante. Con i guadagni $P = 1.77 \cdot 10^{-3}$ e $D = 1.7 \cdot 10^{-5}$ si è arrrivati ad ottenere un tempo di salita di $12ms$, con errore a regime nullo e overshoot assente. Osservando il *datasheet* del motore (si osservi l'Appendice) si può affermare che le prestazioni sono in linea con la costante di tempo meccanica del motore ($4.39ms$). Bisogna infatti considerare che era presente un carico, ossia il giunto elastico.

3.3 Controllo del sistema motore-link

Dopo aver effettuato i test sopraelencati, si è montato il primo link del manipolatore. Con questa nuova configurazione si è rivelato necessario ritarare completamente tutti i guadagni del regolatore, dato che, oltre ad avere un carico differente applicato al motore, siamo in presenza di un sistema con un'inerzia decisamente maggiore.

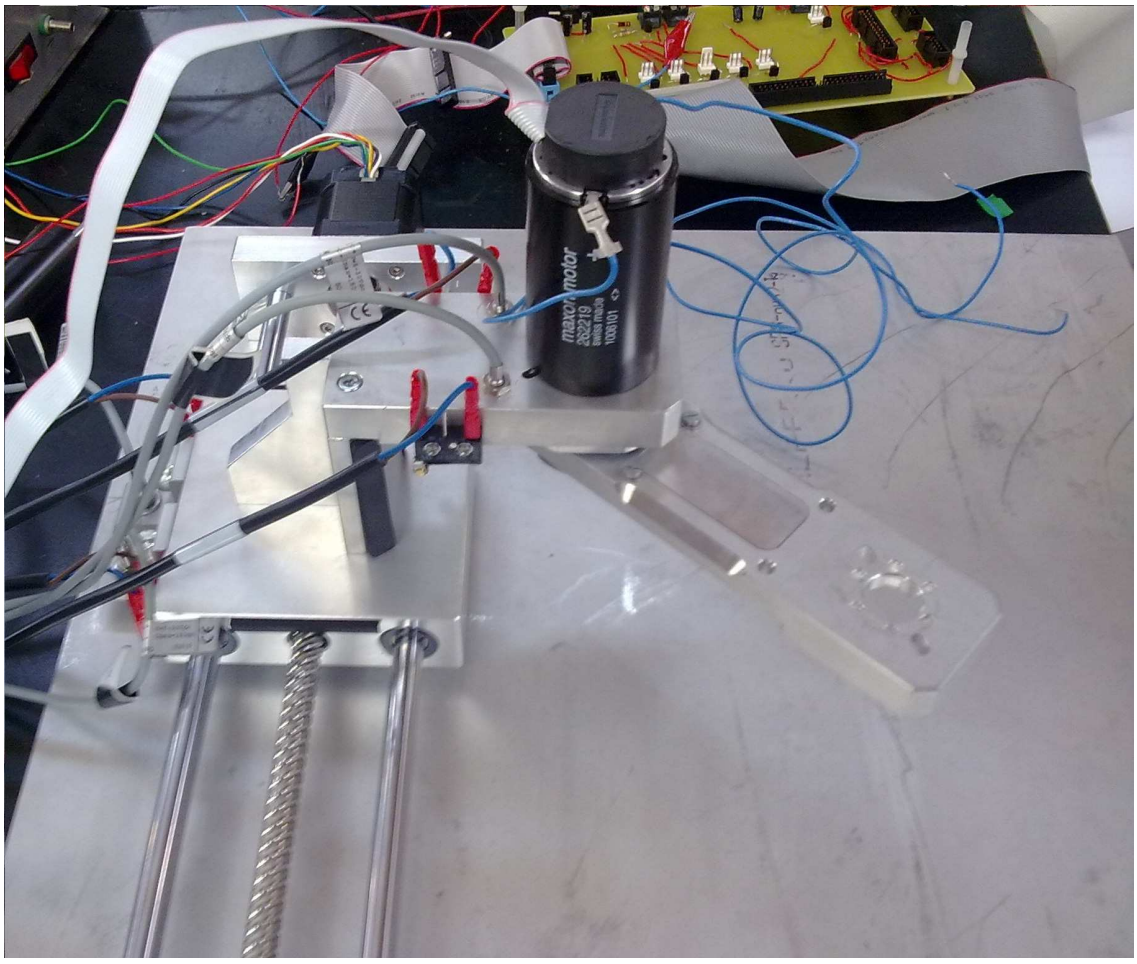


Figura 3.1: Robot con primo link montato.

Dopo il montaggio del primo link sono stati effettuati i seguenti test:

(a) Il primo riferimento di posizione sperimentato è stato uno spostamento da 0 a 60 gradi da effettuare in 3 secondi, secondo una legge a velo-

cità trapezoidale, con tratti di accelerazione (iniziale) e decelerazione (finale) della durata di 0.6666 secondi:

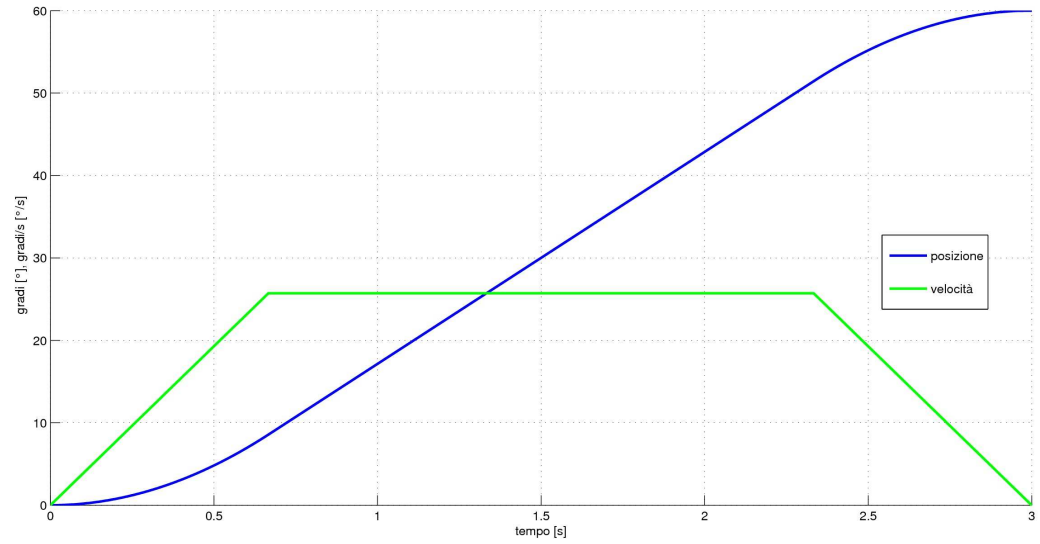


Figura 3.2: Grafico del riferimento - posizione e velocità.

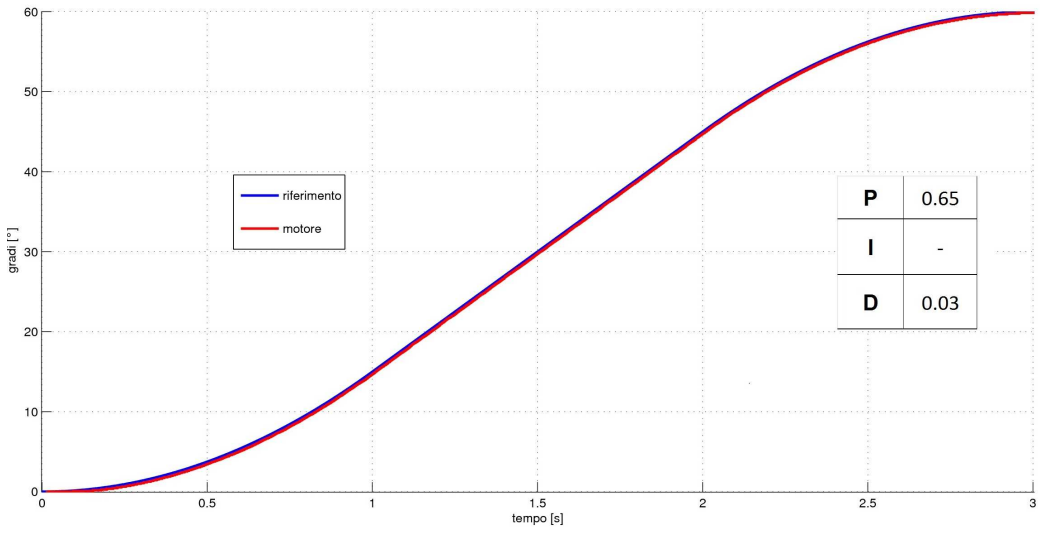


Figura 3.3: Inseguimento di un riferimento a legge trapezoidale di velocità.

Come si può notare dalla figura, l'inseguimento risulta molto buono. E' presente solo un piccolo ritardo al momento del primo distacco, dovuto all'azione di *stick-slip*. Il regolatore utilizzato in questo caso è un PD, nel quale sono stati adeguatamente tarati i guadagni.

(b) Nel successivo test si è tarato il regolatore PD in base alla risposta del motore ad un gradino di 60 gradi. La soluzione con un'azione proporzionale di 0.65 e un'azione derivativa di 0.03 si è rivelata anche in questo caso la più performante, e si è riusciti ad ottenere una siffatta risposta:

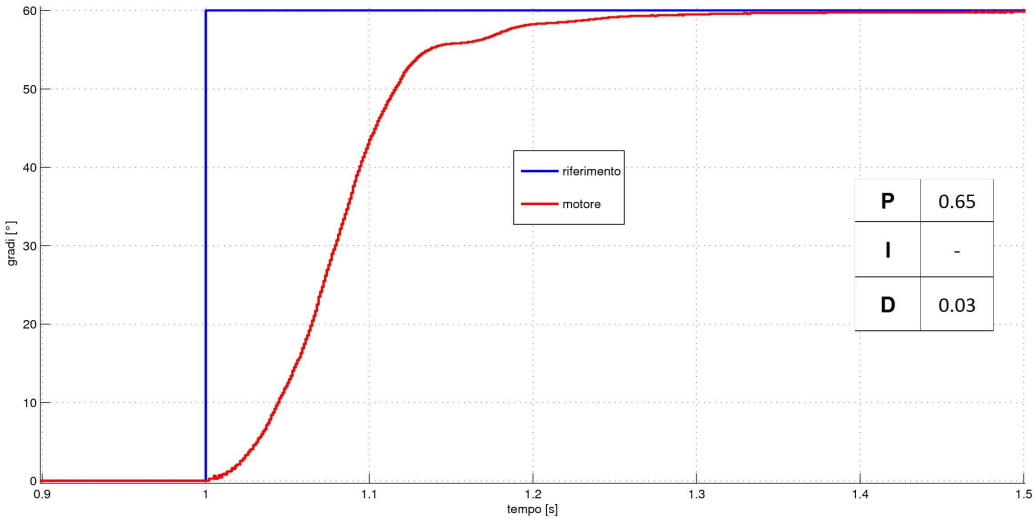


Figura 3.4: Inseguimento di un gradino con regolatore PD.

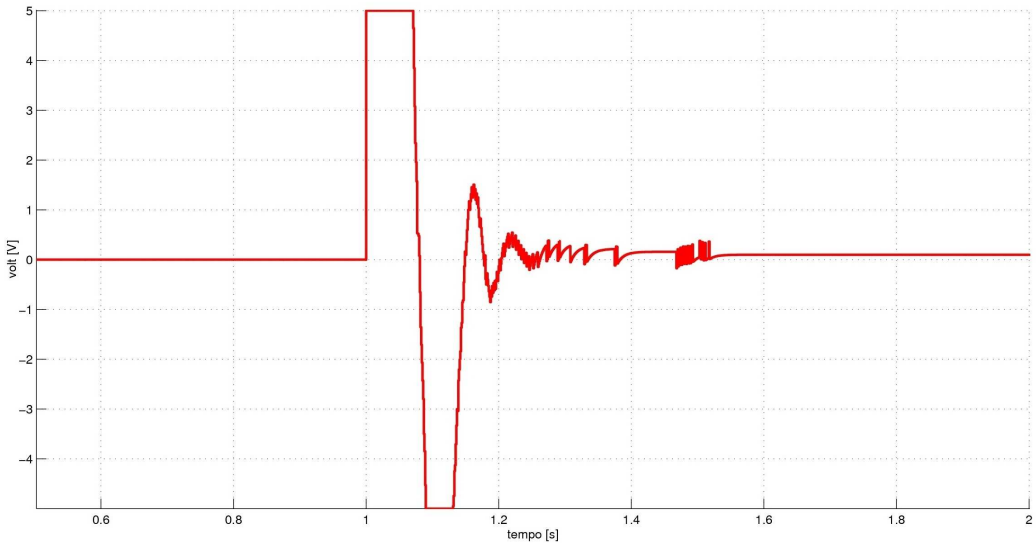


Figura 3.5: Segnale di comando della risposta al gradino.

Come si può notare dal grafico in fig. 3.4, il tempo di salita è di circa $100ms$. Nella taratura dei guadagni si è ovviamente dovuto scartare quasiasi soluzione comprendente un *overshoot*, poiché altrimenti nel nostro caso, se spinto ai limiti della posizione angolare, il robot andrebbe ad azionare i finecorsa induttivi o addirittura quelli meccanici.

E' anche interessante prestare attenzione al grafico dell'attuazione, per osservare se intervenga o meno la saturazione. Come si può osservare dal grafico di fig. 3.5, il segnale di comando satura nell'istante in cui viene inviato il gradino e successivamente a causa della 'frenata'.
Era prevedibile un comportamento del genere, dato che un gradino è il segnale di riferimento meno indicato per l'inseguimento di posizione.

(c) Nel test $n°3$ si è scelto di dare come posizione di riferimento ancora una legge a velocità trapezoidale, ma questa volta lo spostamento angolare di 60° dev'essere sostenuto in soli 0.3 secondi. Il riferimento trapezoidale si avvicina quindi ad un gradino, poiché, anche se non istantaneo, è molto rapido. In più in questo test si comanda al motore di ritornare alla posizione di partenza al secondo 3, avendo come riferimento un gradino, quindi, di $-60°$.

Questo tipo di legge andata/ritorno è molto interessante poiché permette di confrontare direttamente il comportamento del motore ad un riferimento a legge trapezoidale molto spinta con uno a gradino, in modo da notare immediatamente le sostanziali differenze (fig. 3.6).

E' inoltre interessante, anche in questo caso, osservare con la dovuta attenzione il grafico riguardante il segnale di comando: come si può notare in figura 3.7, esso non entra mai in saturazione nell'inseguimento della trapezoidale (se non all'istante 0), mentre, come nel caso precedente, nel caso del riferimento a gradino discendente presenta una doppia zona di saturazione, sia all'istante del gradino che in 'frenata'.

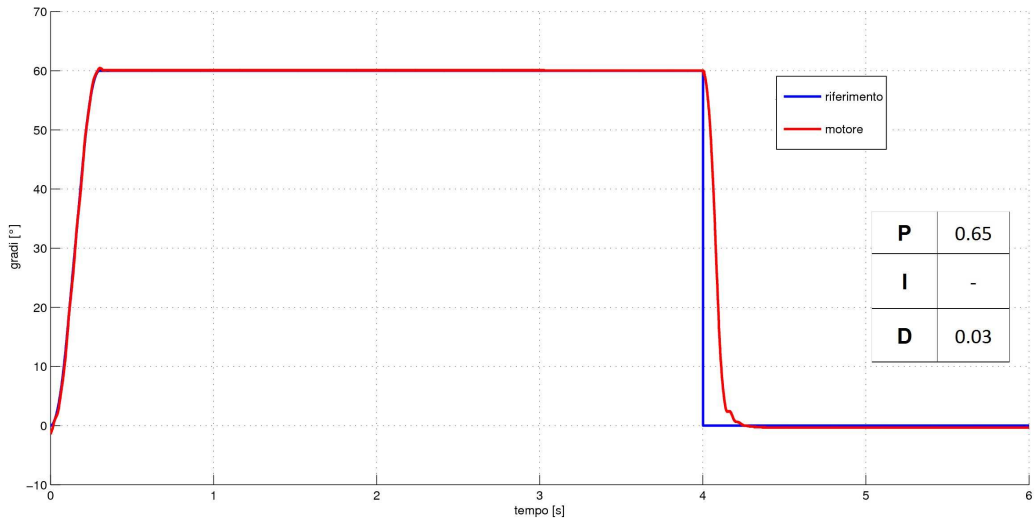


Figura 3.6: Inseguimento andata/ritorno di una trapezoidale + gradino.

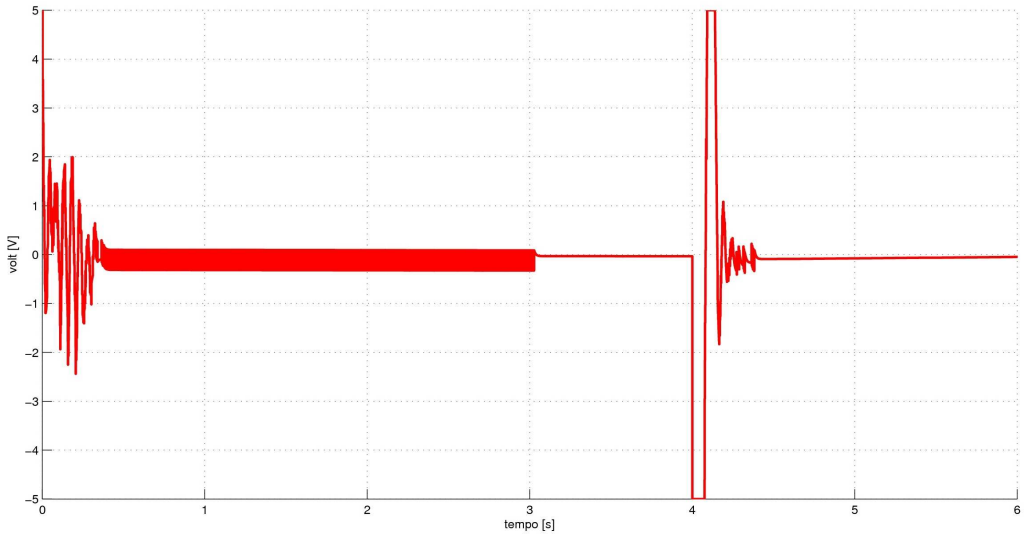


Figura 3.7: Segnale di comando della risposta alla traiettoria sopradescritta.

(d) Infine, nell'ultimo test sperimentale si è scelto di inviare anche in questo caso una traiettoria completa, ossia 'andata e ritorno', ma questa volta utilizzando come riferimento prima una trapezoidale in avanti e successivamente una trapezoidale all'indietro. Il tempo stabilito è di 3 secondi per l'andata e 3 secondi per il ritorno, senza alcuna fase di stallo tra le due azioni.

Questo tipo di riferimento è certamente più adatto all'inseguimento, e come si nota dal grafico in figura 3.8 il robot, trascurando il piccolo ritardo sempre presente, insegue perfettamente la traiettoria. Bisogna tenere presente che stiamo tarando i guadagni con un regolatore di tipo PD, quindi in assenza di azione integrale, la quale sicuramente è un aiuto per il controllo dato che fa in modo di avere errore a regime nullo.

Per far notare la precisione dell'inseguimento, in fig. 3.9 è presente un grafico che illustra l'andamento dell'errore. Come si nota, esso non supera mai i ± 0.5 gradi, errore dovuto al ritardo causato dall'azione dello *stick-slip*.

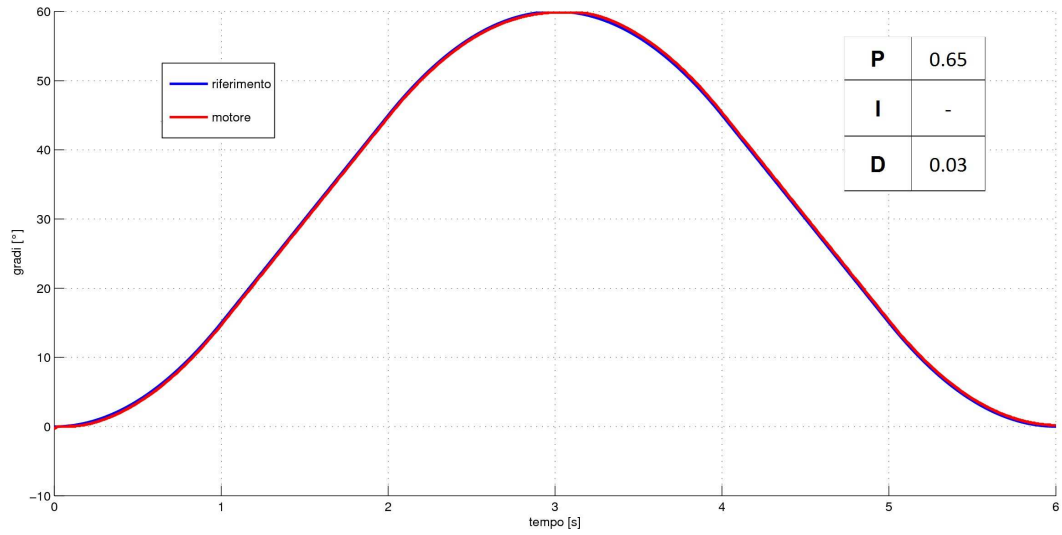


Figura 3.8: Inseguimento di una traiettoria a legge trapezoidale.

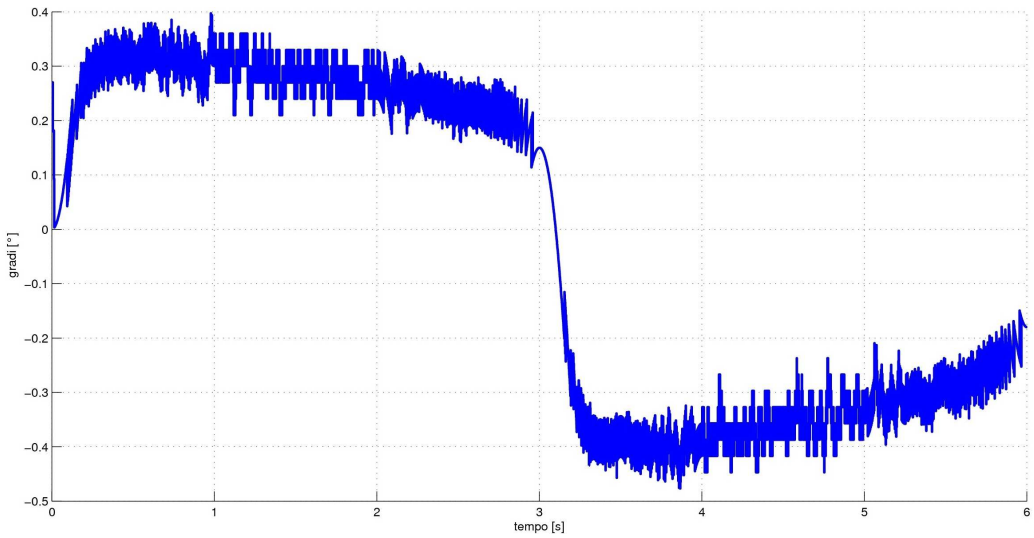


Figura 3.9: Andamento dell'errore nell'inseguimento della traiettoria.

Come ultimo test è stato tarato un regolatore PID sulla risposta al gradino. Grazie all'inserimento dell'azione integrativa si è potuto arrivare ad avere un tempo di salita di $80ms$, come si può osservare in fig. 3.10:

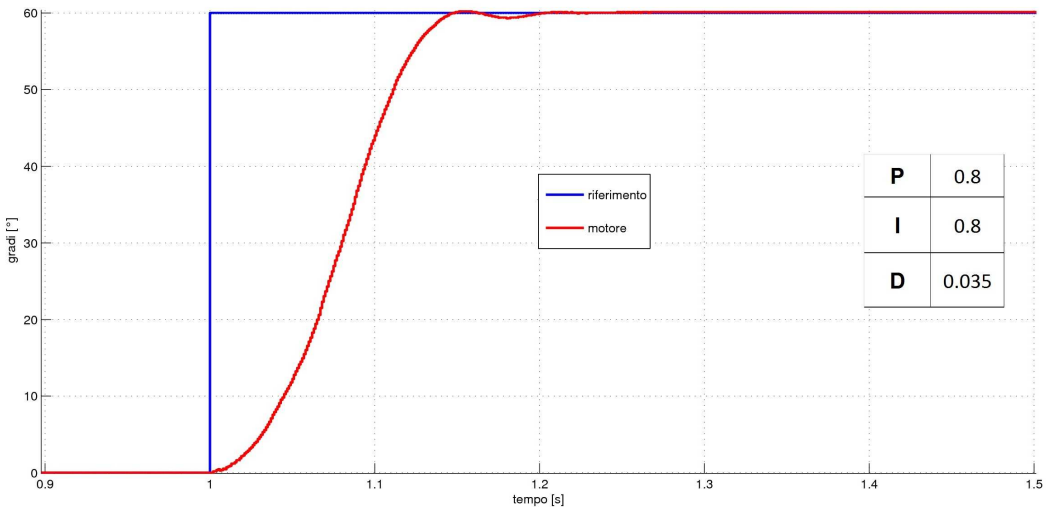


Figura 3.10: Risposta al gradino con regolatore PID.

Conclusioni

In questo lavoro di tesi è stata sviluppata la progettazione e la realizzazione del driver per due motori in continua.

Dopo vari test sperimentali è possibile affermare che entrambi i driver hanno un funzionamento corretto, in quanto il manipolatore risponde in giusta maniera ai riferimenti passàtigli.

E' interessante notare tutte le possibile vie di sviluppo che offre un tale robot: innanzitutto, per quanto riguarda la movimentazione dell'*end-effector*, sarebbe molto interessante implementare un controllo che possa permettere al robot di alzare/abbassare la matita, in modo da interagire con l'ambiente circostante (il piano di lavoro) solo nei momenti desiderati. Per far ciò una buona soluzione potrebbe essere l'impiego di un'elettrocalamita.
Per concludere un'ultima via di sviluppo potrebbe riguardare il controllo della base. Sarebbe interessante implementare, come per i motori in continua, un controllo di corrente anche per il motore passo, che tramite un regolatore PI permetta il fluire di una desiderata corrente definita dal controllo.

In conclusione possiamo considerarci soddisfatti per il lavoro svolto, il quale risulta offrire anche numerosi sbocchi che permettono di ampliare e completare la movimentazione e il controllo del manipolatore a quattro gradi di libertà.

Appendice A

Datasheet

Nell'appendice è riportato il *datasheet* del motore adibito alla movimentazione del primo link del manipolatore.

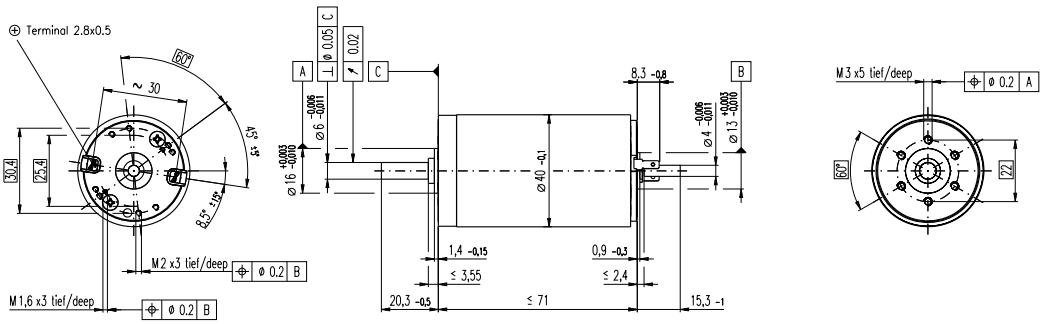
Nel *datasheet* sono presenti vari modelli di motori: quello da noi utilizzato è il modello 148877, ossia la terza colonna da sinistra.

Come accennato precedentemente, la costante di tempo meccanica del motore è di $4.39ms$.

Il *datasheet* relativo al motore adibito alla movimentazione del secondo link non è stato riportato poiché non è stato possibile effettuare i test necessari con i link del manipolatore entrambi montati.

maxon DC motor

RE 40 ∅40 mm, Graphite Brushes, 150 Watt



M 1:2

Stock program
Standard program
Special program (on request)

Order Number

		148866	148867	148877	218008	218009	218010	218011	218012	218013	218014
Motor Data											
	Values at nominal voltage										
1	Nominal voltage	V	12.0	24.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0
2	No load speed	rpm	6920	7580	7580	6420	5560	3330	2690	2130	1710	1420
3	No load current	mA	241	137	68.6	53.7	43.7	21.9	16.7	12.5	9.67	7.77
4	Nominal speed	rpm	6370	6930	7000	5810	4920	2700	2050	1500	1080	774
5	Nominal torque (max. continuous torque)	mNm	94.9	170	184	183	177	187	187	189	189	188
6	Nominal current (max. continuous current)	A	6.00	5.77	3.12	2.62	2.20	1.38	1.12	0.898	0.721	0.593
7	Stall torque	mNm	1680	2280	2500	1990	1580	995	796	641	512	415
8	Starting current	A	102	75.7	41.4	28.0	19.2	7.26	4.68	3.00	1.92	1.29
9	Max. efficiency	%	88	91	92	91	91	89	88	87	86	85
	Characteristics											
10	Terminal resistance	Ω	0.117	0.317	1.16	1.72	2.50	6.61	10.2	16.0	24.9	37.1
11	Terminal inductance	mH	0.0245	0.0823	0.329	0.460	0.612	1.70	2.62	4.14	6.40	9.31
12	Torque constant	mNm / A	16.4	30.2	60.3	71.3	82.2	137	170	214	266	321
13	Speed constant	rpm / V	581	317	158	134	116	69.7	56.2	44.7	35.9	29.8
14	Speed / torque gradient	rpm / mNm	4.15	3.33	3.04	3.23	3.53	3.36	3.39	3.35	3.37	3.44
15	Mechanical time constant	ms	6.03	4.81	4.39	4.36	4.35	4.31	4.31	4.31	4.31	4.32
16	Rotor inertia	gcm²	139	138	138	129	118	123	121	123	122	120

Specifications

Thermal data

17	Thermal resistance housing-ambient	4.65 K / W
18	Thermal resistance winding-housing	1.93 K / W
19	Thermal time constant winding	41.6 s
20	Thermal time constant motor	1120 s
21	Ambient temperature	-30 ... +100°C
22	Max. permissible winding temperature	+155°C

Mechanical data (ball bearings)

23	Max. permissible speed	12000 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	5.6 N
27	Max. force for press fits (static)	110 N
	(static, shaft supported)	1200 N
28	Max. radial loading, 5 mm from flange	28 N

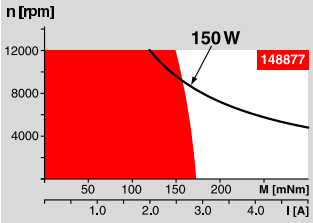
Other specifications

29	Number of pole pairs	1
30	Number of commutator segments	13
31	Weight of motor	480 g

Values listed in the table are nominal.
Explanation of the figures on page 49.

Option
Preloaded ball bearings

Operating Range



Comments

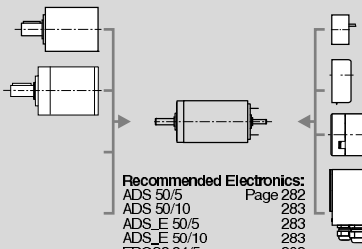
Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System

Overview on page 16 - 21



Planetary Gearhead
∅42 mm
3 - 15 Nm
Page 240

Planetary Gearhead
∅52 mm
4 - 30 Nm
Page 243

Encoder MR
256 - 1024 CPT,
3 channels
Page 265

Encoder HED_ 5540
500 CPT,
3 channels
Page 268 / 270

Brake AB 28
∅45 mm
24 VDC, 0.4 Nm
Page 316

Industrial Version
Encoder HEDL 9140
Page 273
Brake AB 28
Page 317

Recommended Electronics:

ADS 50/5	Page 282
ADS 50/10	283
ADS_E 50/5	283
ADS_E 50/10	283
EPOS2 24/5	303
EPOS2 50/5	303
EPOS 70/10	303
EPOS P 24/5	306
Notes	**18**

May 2009 edition / subject to change

Figura A.1: Datasheet del motore relativo al primo link.

Bibliografia

[1] http://www.adept.com

[2] http://www.sensoray.com/downloads/626%20Instruction%20Manual_F.pdf

[3] http://www.sensoray.com/assets/images/626photo.jpg

[4] http://www.microchip.com

[5] *dsPIC*30*F*4011/4012 *Data Sheet*, Microchip Technology Inc., 2008

[6] http://www.vincenzov.net/tutorial/motoridc/driver.htm

[7] *LMD*18200 *Data Sheet*, National Semiconductor Corporation, 2005

[8] Tim Regan, *Application Note* 694, National Semiconductor, 1999

[9] http://www.kerhuel.eu/

[10] *LM*78*XX*/*LM*78*XXA Data Sheet*, Farchild Semiconductor, 2010