UNIVERSITÀ DEGLI STUDI DI ROMA
TOR VERGATA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA
DELL'AUTOMAZIONE

A.A. 2009/2010

Tesi di Laurea

PROGETTAZIONE E REALIZZAZZIONE DI UN ROBOT PARALLELO

RELATORE

Martinelli Francesco

CANDIDATO

Pietroni Rocco

Alla mia famiglia ed a Eloisa.

Indice

Ringraziamenti

Innanzitutto vorrei ringraziare il professor Martinelli per l'aiuto e la disponibilità mostrata durante il lavoro che ha portato alla scrittura di questa tesi. Volevo poi ringraziare la mia famiglia, che mi ha sostenuto durante i 'duri' anni dell'università, aiutandomi a far coesistere lavoro e studio, senza farmi pesare le mie assenze nè da una nè dall'altra parte. Un ringraziamento è necessario anche per tutto il corpo docenti della facoltà di automazione, che si è dimostrato molto valido e disponibile. Per concludere volevo ringraziare Eloisa per avermi 'costretto' a impegnarmi fino in fondo anche nei periodi più bui.

Introduzione

La robotica è nata come l'esigenza umana di automatizzare processi ripetitivi e troppo dispendiosi per gli esseri umani. I primi robot sono apparsi nelle fabbriche durante la rivoluzione industriale, con la crescita della richiesta di produzione e l'evoluzione della meccanica sono comparsi i primi automatismi per la produzione in catena di montaggio. In seguito con l'avvento dell'elettrotecnica e successivamente con la nascita dell'elettronica, la robotica da materia puramente meccanica è andata ad abbracciare altri campi dell'ingegneria, fornendo cosi nuove tecniche per la progettazione di automi sempre piu specializzati ed efficenti.

Il progetto LGTP2411 si colloca nel campo dei robot industriali, ancor più nello specifico nei robot pick-and-place, è caratterizzato da una forma a 'ragno' che gli consente di avere un rapporto robustezza-leggerezza estremamente elevato, questo gli permette di aumentare la velocità di lavoro in modo notevole, rispetto ai suoi 'fratelli'. Tali caratteristiche però lo rendono poco adatto a lavorazioni in cui è richiesta una coppia elevata. In particolare LGTP2411 è un delta robot, appartenente alla famiglia dei robot paralleli, ossia quel tipo di automi che lavorano su un piano parallelo alla posizione del telaio. Tale configurazione rappresenza da una parte un vincolo sulla dinamicità delle lavorazioni che si possono intraprendere, da un'altra parte la specializzazzione in una piccola parte di queste.

In queste tesi si analizzeranno le problematiche inerenti alla progettazione mec-

canica ed elettronica, inoltre verrano introdotti due diversi sistemi di controllo. Si inizierà con l'introdurre i problemi di cinematica del meccanismo, verranno poi illustrati i calcoli necessari alla corretta risoluzione. In seguito si procederà con il dimensionamento delle grandezze cinematiche del sistema meccanico, adottando tecniche basate su algoritmi informatici. Nel secondo capitolo verranno introdotte e risolte le problematiche del sistema di controllo, si utilizzeranno tecniche di teoria del controllo a tempo discreto ed a tempo continuo. Verranno infine illustrati gli algoritmi di implementazione dei controllori progettati. Nel terzo capitolo, verrà analizzato il sistema elettronico di potenza e di comando che gestisce l'intero robot. Verranno cosi illustrate le tecnologie utilizzate e i protocolli di comunicazione adottati.

Capitolo 1

Meccanica

1.1 Cinematica

1.1.1 Piattaforma e software di progettazione

La progettazione dei componenti meccanici è stata realizzata su piattaforma Windows 7 mediante alcuni software specifici.

Per il disegno del meccanismo, è stato utilizzato l'ambiente CAD SolidWorks che fornisce nella sua suite anche tool di simulazione cinematici e un esportatore DWG integrato, grazie al quale è stato possibile interfacciare le specifiche metriche dei componenti del meccanismo sviluppate nel programma CAD con QuickCAM. Quest'ultimo è un software atto al controllo di macchine utensili, nel nostro caso una fresa a controllo numerico utilizzata per la lavorazione dei componenti del meccanismo.

Mentre per i calcoli dovuti allo studio della cinematica e a quello della dinamica, è stato utilizzato il software di calcolo Matlab.

1.1.2 La catena cinematica

Il meccanismo in esame è una catena cinematica composta spaziale, quindi siamo in presenza di alcuni membri ternari ed inoltre i punti dei membri mobili non sono paralleli al piano del telaio né concentrici ad esso.

In figura 1.1 è riportato uno schema della catena cinematica del meccanismo, sono inoltre indicate le nomeglature utilizzate nei capitoli successivi.

Di seguito sono elencati riferimenti ai membri del meccanismo:

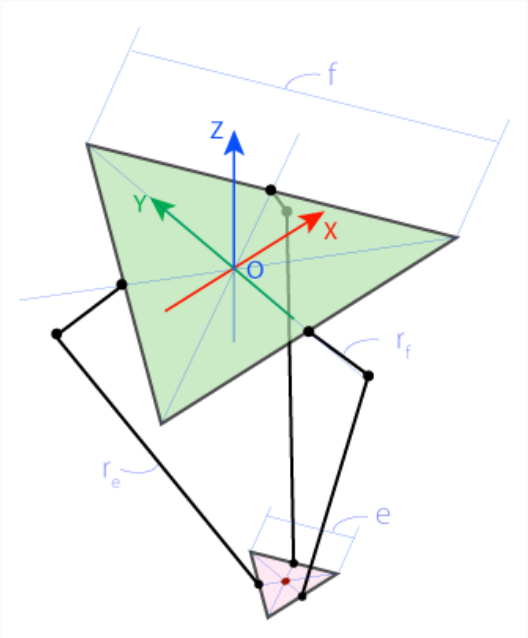


Figura 1.1: Catena cinematica.

Base superiore: Triangolo di lato f. (telaio)

Base Inferiore: Triangolo di lato e. (base dell'effettore)

Manovella: r_f.

Biella: r_e.

Da notare la particolare geometria del sistema, che presenta una base superiore, funzionante da telaio, alla quale sono accoppiati tramite coppia rotoidale tre brac-

ci (manovella) sfasati di 120° ai quali è collegato rispettivamente un secondo membro (biella) a sua volta incernierato sulla base inferiore, rispettando la geometria sopra definita.

1.1.3 Cinematica inversa

La problema della cinematica inversa rappresenta la determinazione di alcuni parametri del sistema meccanico, che se imposti, determinano il corretto raggiungimento di una posizione desiderata. Nel caso del delta robot data la posizione del centro della base inferiore $E(X_o, Y_o, Z_o)$, si vanno a ricavare le posizioni angolari delle tre manovelle $(\theta_1, \theta_2, \theta_3)$. Si può analizzare il problema della cinematica inversa scomponendolo in tre parti, infatti la geometria del robot consente di riferire le equazioni scritte per un unico percorso cinematico composto dai membri base superiore, manovella, biella, base inferiore, agli altri due, tramite un cambiamento di coordinate, definito da una matrice di rotazione.

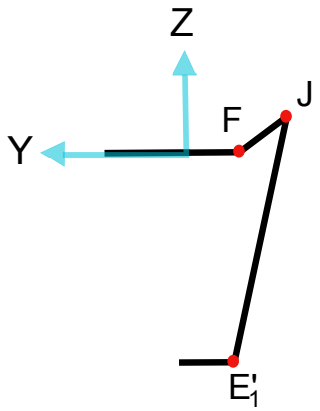


Figura 1.2: Proiezione del meccanismo sull'asse ZY

In figura 1.3 possiamo vedere un'analisi della base inferiore sul piano XY, osservando la geometria e ricordando che e è la lunghezza del lato del triangolo e che l'angolo che un lato forma con la bisettrice adiacente è 30°, possiamo dire che il segmento

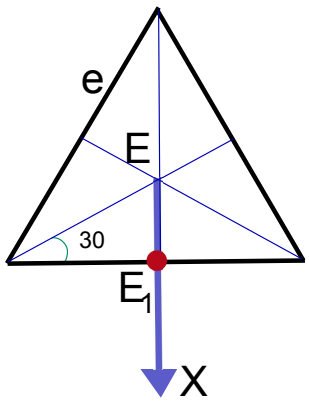


Figura 1.3: Base inferiore piano XY

$$E\bar{E}_1 = \frac{e}{2}\tan 30 \tag{1.1.1}$$

e quindi

$$E_1 = (X_o, Y_o - \frac{e}{2}\tan 30, Z_o) \tag{1.1.2}$$

proiettando nel piano ZY il punto E_1 trovo il punto E_1', formando cosi la geometria in figura 1.2 data la (1.1.2) si possono determinare facilmente le coordinate del punto E_1'.

$$E_1' = (0, Y_o - \frac{e}{2}\tan 30, Z_o) \tag{1.1.3}$$

$$\bar{E_1E_1'} = X_o \tag{1.1.4}$$

Osservando ora la figura 1.2, ponendo che $J(0, y_j, z_j)$, utilizzando il Teorema di Pitagora si deducono le seguenti equazioni:

$$\bar{E_1'J} = \sqrt{\bar{E_1J}^2 - \bar{E_1E_1'}^2} = \sqrt{r_e^2 - X_o^2} \tag{1.1.5}$$

$$\begin{cases} (y_j - y_f)^2 + (z_j - z_f)^2 = r_f^2 \\ (y_j - y_{E_1'})^2 + (z_j - z_{E_1'})^2 = r_e^2 - x_0^2 \end{cases} \tag{1.1.6}$$

Da cui risolvendo il sistema per z_j, y_j si trova l'angolo tra biella e telaio θ_1, dalla formula

$$\theta_1 = arctan\left(\frac{z_j}{y_f - y_j}\right) \tag{1.1.7}$$

Per determinare gli angoli θ_2, θ_3, moltiplico le coordinate $E(X_0, Y_0, Z_0)$ per una matrice di rotazione di 120°, infatti, sia

$$R_{12} = \begin{bmatrix} cos120 & sin120 & 0 \\ -sin120 & cos120 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{1.1.8}$$

la matrice di rotazione sopra descritta, e $\begin{Bmatrix} X_2 \\ Y_2 \\ Z_2 \end{Bmatrix}$ il vettore posizione riferito al secondo circuito cinematico, allora si ha la relazione

$$\begin{Bmatrix} X_2 \\ Y_2 \\ Z_2 \end{Bmatrix} = R_{12} \begin{Bmatrix} X_0 \\ Y_0 \\ Z_0 \end{Bmatrix} \tag{1.1.9}$$

Ora risolvendo le (1.1.6) descritte per le nuove coordinate (1.1.9), si troverà l'angolo θ_2 e con un'ulteriore rotazione delle coordinate, θ_3.

1.1.4 Cinematica diretta

Il problema della cinematica diretta, rappresenta il voler ricavare, dati gli angoli $\theta_1, \theta_2, \theta_3$ la posizione dell'effettore. Analizzando la figura 1.4a, che descrive la proiezione sul braccio di manovella del punto E_0 e ricordando che valgono le relazioni

$$\bar{E_0E} = \frac{e}{2\sqrt{3}} \tag{1.1.10}$$

$$\bar{OF} = \frac{f}{2\sqrt{3}} \tag{1.1.11}$$

possiamo facilmente ricavare le coordinate del punto J_1', analogamente anche J_2', J_3'

$$J_1' = \left[0; -\frac{f-e}{2\sqrt{3}} - r_f \cos\theta_1; -r_f \sin\theta_1\right] \tag{1.1.12}$$

$$J_2' = \left[[\frac{f-e}{2\sqrt{3}} - r_f \cos\theta_2]\cos 30; [\frac{f-e}{2\sqrt{3}} - r_f \cos\theta_2]\sin 30; -r_f sin\theta_2\right] \tag{1.1.13}$$

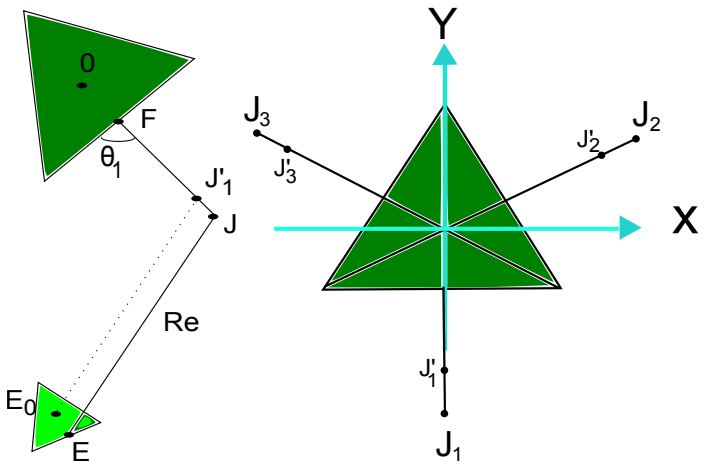
$$J_3' = \left[[\frac{f-e}{2\sqrt{3}} - r_f \cos\theta_3]\cos 30; [\frac{f-e}{2\sqrt{3}} - r_f \cos\theta_3]\sin 30; -r_f sin\theta_3\right] \tag{1.1.14}$$

Costruendo 3 sfere facendo ruotare il segmento $\bar{E_0J_i'}$ intorno a J_i' con $i \in (1,2,3)$, la loro intersezione risulta essere esattamente il punto E_0, quindi imponendo un sistema di tre equazioni rappresentanti tre sfere di raggio r_e come

$$\begin{cases} x^2 + (y-y_1)^2 + (z-z_1)^2 = r_e^2 \\ (x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2 = r_e^2 \\ (x-x_3)^2 + (y-y_3)^2 + (z-z_3)^2 = r_e^2 \end{cases} \tag{1.1.15}$$

dove abbiamo utilizzato le coordinate della sfera i-esima $\begin{Bmatrix} x_i \\ y_i \\ z_i \end{Bmatrix}$, mentre per le coordinate del punto E_0 $\begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$; troviamo che risolvendo i quadrati

$$\begin{cases} x^2 + y^2 + z^2 = r_e^2 + 2y_1y + 2z_1z - y_1^2 - z_1^2 \\ x^2 + y^2 + z^2 = r_e^2 + 2x_2x + 2y_2y + 2z_2z - x_2^2 - y_2^2 - z_2^2 \\ x^2 + y^2 + z^2 = r_e^2 + 2x_3x + 2y_3y + 2z_3z - x_3^2 - y_3^2 - z_3^2 \end{cases} \tag{1.1.16}$$



(a) Proiezione J_1' (b) Proiezione J_i' vista dal piano X-Y

Figura 1.4: Cinematica diretta

posto

$$w_i = x_i^2 + y_i^2 + z_i^2 \tag{1.1.17}$$

$$\begin{cases} xx_2 + y(y_2 - y_1) + z(z_2 - z_1) = \frac{(w_2 - w_1)}{2} \\ x(x_3 - x_1) + y(y_3 - y_1) + z(z_3 - z_1) = \frac{(w_3 - w_1)}{2} \\ xx_3 + y(y_3 - y_2) + z(z_3 - z_2) = \frac{(w_3 - w_2)}{2} \end{cases} \tag{1.1.18}$$

e

$$d = (y_2 - y_1)x_3 - (y_3 - y_1)x_2 \tag{1.1.19}$$

$$a_1 = \frac{1}{d}\left[(z_2 - z_1)(y_3 - y_1) - (z_3 - z_1)(y_2 - y_1)\right] \tag{1.1.20}$$

$$b_1 = -\frac{1}{2d}\left[(w_2 - w_1)(y_3 - y_1) - (w_3 - w_1)(y_2 - y_1)\right] \tag{1.1.21}$$

$$a_2 = -\frac{1}{d}\left[(z_2 - z_1)x_3 - (z_3 - z_1)x_2\right] \tag{1.1.22}$$

$$b_2 = \frac{1}{2d}\left[(w_2 - w_1)x_3 - (w_3 - w_1)x_2\right] \tag{1.1.23}$$

$$\tag{1.1.24}$$

Troviamo che

$$x = a_1 z + b_1 \tag{1.1.25}$$

$$y = a_2 z + b_2 \tag{1.1.26}$$

$$0 = (a_1^2 + a_2^2 + 1)z^2 + 2(a_1 + a_2(b_2 - y_1) - z_1)z + (b_1^2 + (b_2 - y_1)^2 + z_1^2 - r_e^2 \tag{1.1.27}$$

sono le soluzioni del problema di cinematica diretta.

1.1.5 Dimensionamento

Per il dimensionamento delle grandezze dei membri cinematici si è utilizzato un metodo iterativo. Tale metodo ha bisogno di alcuni vincoli matematici per dare una soluzione corretta. I vincoli utilizzati sono:

- $f \geq 2e$
- $r_e \geq \frac{3}{2} r_f$
- $x \in [-25; 25], y \in [-25; 25], z \in [-40, -60]$ in centimetri

L'algoritmo utilizzato, posto il lato dell'effettore e, va a calcolare tutte le altre grandezze. Esso effettua tante chiamate alla funzione cinematica inversa, quanti sono i punti nello spazio di lavoro desiderato, verificando così la corretta raggiungibilità della posizione. Oltre ai vincoli caratteristici dei membri, si necessita anche di conoscere il massimo angolo raggiungibile tra telaio-manovella e i due angoli del giunto sferico tra biella-manovella ed effettore-biella. Tali parametri sono caratteristici dei giunti utilizzati e della reale forma delle basi. Nel caso del progetto LGTP2411 discusso in questa tesi, si utilizzano i seguenti parametri

- $\hat{Tr}_f = \pm 78°$

- $\hat{Er_e}\| = \pm 144°$
- $\hat{Er_e} \perp = \pm 144°$
- $r_e\hat{r}_f\| = \pm 37°$
- $r_e\hat{r}_f \perp = \pm 37°$

dove si è posto T il membro telaio, E l'effettore, r_f la manovella ed r_e la biella. Mentre i simboli $\|, \perp$ rappresentano rispettivamente l'angolo parallelo e perpendicolare ad r_e. I risultati ottenuti sono:

$$e = 10; f = 23; r_f = 51.5; r_e = 30$$

Nella realizzazzione dei componenti è stato fatto un errore in eccesso, dovuto alle meccaniche di collegamento tra i vari membri (distanziali, rondelle, giochi), questo ha causato un incremento dello spazio di lavoro ma anche un abbassamento del piano di lavoro ottimale. Infatti se nei dati teorici il piano di lavoro ottimale si trovava a -50cm, nel reale esso si trova a -51.3 ± 0.5; la tolleranza è dovuta ai giochi.

1.1.6 Scelta dei componenti cinematici

Come materiali costruttivi è stato scelto per quanto riguarda i membri di biella e manovella, dei tondini pieni di alluminio anodizzato, che offrono una rigidità ed una leggerezza molto elevata, al contempo però sono estremamente morbidi, sono quindi stati trattati appositamente. Per quanto riguarda invece il telaio esso è in acciaio temprato, questo perchè offre una migliore resistenza alle sollecitazioni dovute ai motori ad esso collegati. L'effettore è stato realizzato in materiale plastico, inoltre è stato lavorato nella sezione centrale in modo da creare un vano, adibito ad alloggiare strumenti di lavoro (ventose, trapani).

Per quanto riguarda i giunti di collegamento, anch'essi sono in materiale plastico, che offre una notevole resistenza alle forze trasversali all'asse del giunto ed una completa assenza di manutenzione ed oli. Tali caratteristiche offrono la possibilità di utilizzare il robot per operazioni di pick-and-place in ambienti sterili.
Infine i motori sono stati scelti sovradimensionati, per un eventuale sviluppo del progetto. Essi hanno le seguenti caratteristiche

- Velocità di rotazione 175 rpm
- Coppia massima: $10\frac{kg}{cm^2}$

Capitolo 2

Sistema di Controllo

2.1 Introduzione

In questo capitolo si progetterà il sistema di controllo del robot, in particolare si prenderanno in considerazione due diversi tipi di regolazione:

- Controllo in posizione.
- Controllo della velocità.

Nel primo punto lo scopo è quello di raggiungere la posizione desiderata. Per fare ciò si utilizzerà un regolatore non-standard basato su specifiche di sistema in frequenza; mentre nel secondo caso, lo scopo del sistema dinamico è quello di raggiungere una velocità prefissata. Come nel precedente tipo di controllo, si utilizzerà un regolatore non-standard con specifiche nel tempo.
Per progettare un controllore per il controllo di posizione è necessario, quindi, la conoscienza della funzione di trasferimento del processo da controllare.

2.2 Il processo

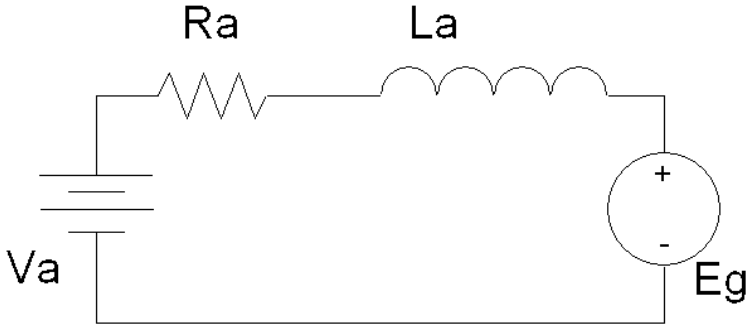
Il processo da controllare, in entrambi i tipi di controllo, è un motoriduttore in corrente continua (CC) con le seguenti caratteristichei:

- Rapporto di riduzione $1 : 50$.
- Tensione di armatura $7.5V$.
- Velocità a vuoto dopo riduzione $175rpm$.
- Coppia massima dopo riduzione $10Kg/cm^2$.

Per analizzare e quindi dedurre le equazioni caratterisctiche di un motore, si scompone il problema in due parti; si analizza dapprima la componente elettrica ed in seguito la parte meccanica, infine si uniscono le due analisi.

2.2.1 Analisi della componente elettrica

Il circuito equivalente di un motore in CC è descritto da questo schema:



dove possiamo scrivere:

$$E_g(t) = K_e w(t) \tag{2.2.1}$$

$$V_a(t) = R_a I(t) + \frac{\delta I(t)}{\delta t} L_a + E_g(t) \tag{2.2.2}$$

$$\tag{2.2.3}$$

Abbiamo poi la relazione

$$C_m(t) = K_t I(t) \tag{2.2.4}$$

che collega la coppia meccanica con la corrente di armatura. Le costanti K_t, K_e sono dette costante coppia del motore e costante di velocità del motore. Sono parametri costruttivi del motore e rese disponibili dal fabbricante, come anche la resistenza di armatura R_a e l'induttanza di armatura L_a.

2.2.2 Analisi della componente meccanica

L'analisi meccanica di un motoriduttore è suddivisa in tre parti distinte, la prima quella del solo motore, la seconda quella del riduttore e la terza è la descrizone della dinamica del carico.

Per quanto riguarda il motore si può scrivere la seguente relazione:

$$C_m = J_m\ddot{\theta_m} + B_m\dot{\theta_m} + d + C_{m_0} \tag{2.2.5}$$

dove d è un disturbo esterno e θ rappresenta la posizione angolare. Anche per il carico la dinamica risulta semplice:

$$C_{m'} = J_o\ddot{\theta} + B_o\dot{\theta} + d' \tag{2.2.6}$$

dove anche in questo caso d' è un disturbo esterno, J_o, J_m coefficenti del momento di inerzia mentre B_o, B_m sono i coefficenti di attrito viscoso.

Per il riduttore supponendo che il rapporto di riduzione sia $\frac{1}{n}$ vale:

$$\theta = \frac{\theta_m}{n} \rightarrow \ddot{\theta} = \frac{\ddot{\theta_m}}{n} \tag{2.2.7}$$

Se uniamo le equazioni (2.2.5), (2.2.6) e (2.2.7) troviamo

$$\xi C_m\dot{\theta_m} = C_{m'}\dot{\theta_m} \rightarrow C_{m_0} = \frac{C_{m'}}{n\xi} \tag{2.2.8}$$

dove ξ è il coefficente di attrito radente. Sosituendo (2.2.8) in (2.2.5) abbiamo

$$C_m = (J_m + \frac{J_o}{n^2\xi})\ddot{\theta} + (B_m + \frac{B_o}{n^2\xi})\dot{\theta} + \frac{d'}{n\xi} + d \tag{2.2.9}$$

$$= J_{eff}\ddot{\theta} + B_{eff}\dot{\theta} + C_d \tag{2.2.10}$$

che rappresenta l'equazione dinamica del motoriduttore.

2.2.3 Funzione di trasferimento

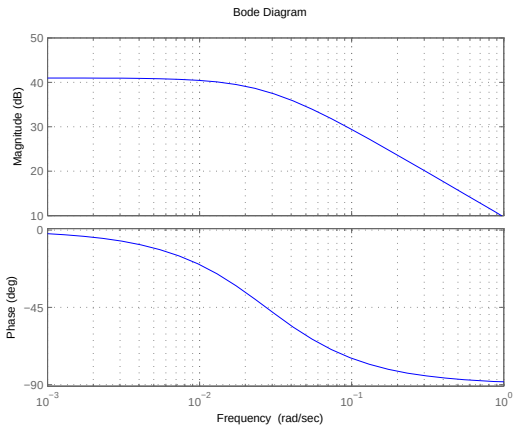
Unendo i risultati ottenuti nei paragrafi precendenti possiamo scrivere la funzione di trasferimento del motoriduttore che collega la velocità angolare dell'albero alla tensione di armatura. Si ha

$$P(s) = \frac{\theta}{V_a} = \frac{K_t n}{s^2(L_a J_{eff}) + s(L_a B_{eff} + R_a J_{eff}) + (R_a B_{eff} + K_e K_t)} \tag{2.2.11}$$

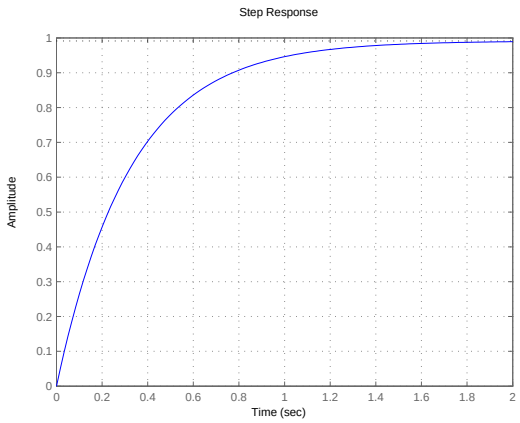
con radici in:

$$s_1 = -0.027 s_2 = -2e^5$$

avendo imposto che i disturbi sul carico e i disturbi sul motore siano 0 ($C_d = 0$).
Sono riportati di seguito i diagrammi di bode e la risposta al gradino del processo.



(a) Diagramma di bode del processo

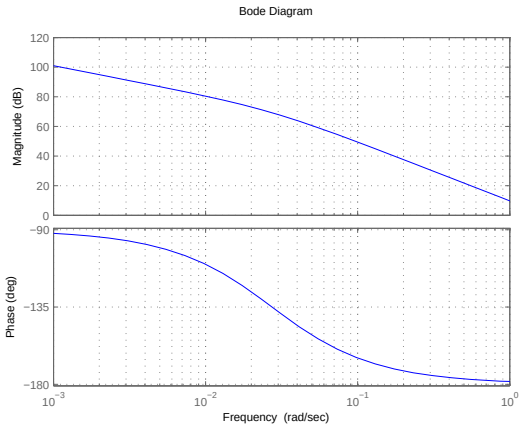


(b) Risposta al gradino

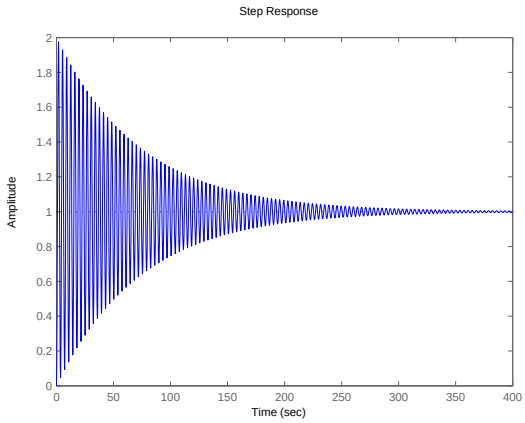
2.3 Controllo in posizione

Per progettare un controllore per il corretto raggiungimento della posizione dobbiamo avere la funzione di trasferimento del processo che colleghi la tensione in ingresso allo

statore alla posizione angolare dell'albero. Per fare ciò si applica un blocco integratore all'uscita del sistema, facendo variare quindi il comportamento del processo. Di seguito sono riportati i diagrammi di Bode e la risposta a gradino del nuovo processo.



(a) Diagramma di bode del processo



(b) Risposta al gradino

2.3.1 Analisi in Laplace

Dai diagrammi di bode possiamo ricavare i parametri del sistema:

- Margine di fase: $m_\phi = 0.8949$
- Frequenza di attraversamento: $w_t = 1.7496$
- Margine di guadagno: $1.7863e + 003$

Mentre le specifiche di progetto sono:

- Tempo di salita: $t_s < 0.4$
- Sovraelongazione: $\hat{s} < 20$
- Errore di posizione: $e_p = 0$

E' noto che esistono delle correlazioni tra i parametri temporali e quelli in frequenza, in particolare si ha:

$$B_3 t_s \approx 3 \tag{2.3.1}$$

$$1 + \hat{s} \approx 0.85 M_r \tag{2.3.2}$$

$$\tag{2.3.3}$$

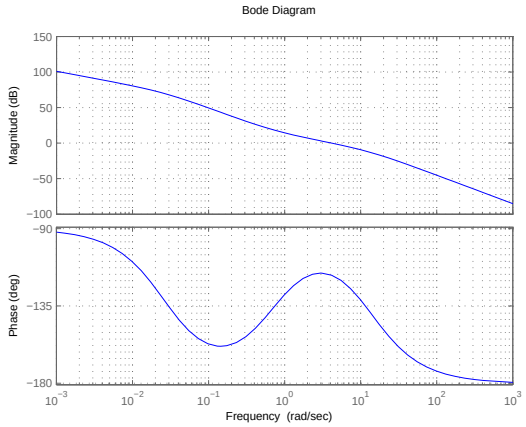
inotre

$$m_\phi < 90^\circ \rightarrow B_3 > w_t \tag{2.3.4}$$

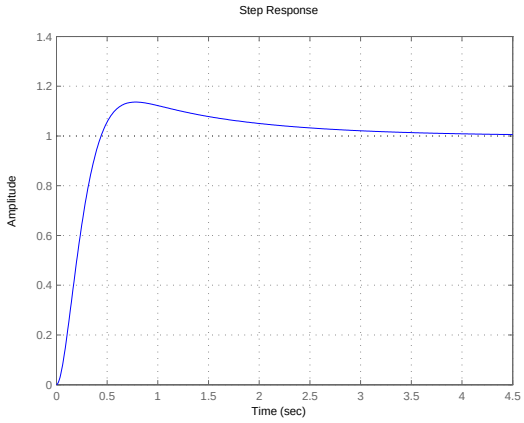
$$m_\phi > 90^\circ \rightarrow B_3 < w_t \tag{2.3.5}$$

$$\tag{2.3.6}$$

Quindi procedendo per tentativi dai diagrammi delle reti correttrici, si sceglie una rete anticipatrice con parametri $m = 15$ $w_t = 4 \rightarrow \tau = 1.4$. Questi forniscono quindi un guadagno di fase di 50° con l'aumento però della frequenza di attraversamento di 6 dB, portandola cosi a $w_t = 7.46$. Per valutare le prestazioni di un regolatore siffatto si analizza il diagramma di bode e la risposta al gradino. in particolare il sistema



(a) Diagramma di bode del processo controllato



(b) Risposta al gradino

presenta un margine di fase di 57°. E' quindi un sistema stabile. Inoltre analizzando la risposta al gradino, vediamo che i vincoli di progetto sono rispettati. Si vuole però una convergenza più rapida al valore di riferimento, ossia un tempo di assestamento meno ampio. Si moltiplica il guadagno statico per una costante $K = 2$.

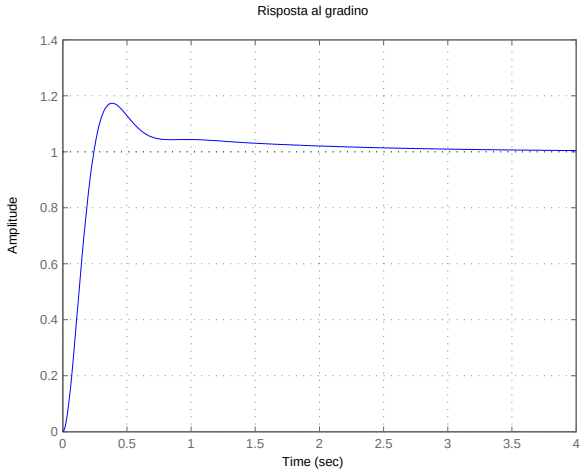


Figura 2.1: Risposta al gradino del processo compensato

Il risultato è quello ottenuto in figura 2.1. Il regolatore ha quindi la forma:

$$R_a = \frac{2.8s + 2}{0.077s + 1} \tag{2.3.7}$$

2.3.2 Discretizzazzione

Visto che il controllore deve essere implementato in un circuito digitale, il regolatore analizzato in precedenza nel dominio di Laplace deve essere discretizzato, ossia esso verrà trasformato in un regolatore digitale descritto da una funzione di trasferimento discreta nel dominio Zeta. Per fare una analisi del sistema dinamico a tempo discreto è necessario portare anche il processo in una forma discreta. Al contrario del regolatore, al quale viene applicata una tecnica di discretizzazzione, alla funzione di trasferimento del motore viene semplicemente applicata la trasformata Z. Nel fare ciò

però si deve tener conto che il feedback che viene prelevato dal motore è elaborato da un convertitore Analogico-Digitale, nel nostro caso un mantenitore di ordine zero (ZOH).

Quindi per quanto riguarda il processo, la sua trasformata zeta comprensiva del contributo dato dal convertitore è:

$$T_s = 0.0084 \tag{2.3.8}$$

$$G(z) = \frac{1.52910^{-4}z^2 + 1.52310^{-4}z + 7.65210^{-11}}{z^3 - 2z^2 - 2.64610^{-23}} \tag{2.3.9}$$

il denominatore è del terzo grado inquanto è stato inserito un blocco integratore in cascata al processo originale, per valutarne la posizione e non la velocità. Mentre per il regolatore, è stata scelta la tecnica della corrispondenza zeri-poli per la discretizzazzione:

$$(2.8s + 2) \rightarrow (z - 0.994) \tag{2.3.10}$$

$$(0.077s + 1) \rightarrow (z - 0.8966) \tag{2.3.11}$$

$$C(z) = 36.36\frac{z - 0.994}{z - 0.8966} \tag{2.3.12}$$

la risposta al gradino del sistema con un regolatore digitale siffatto, con tempo di campionamento pari a 0.0084 è mostrata in figura 2.2. Risulta che le specifiche di progetto sono rispettate ed il sistema risulta stabile. Un'implementazione del sistema in un microcontrollore digitale è mostrata nel listato a fine capitolo. Per determinare l'algoritmo informatico è stato necessario portare il regolatore in una forma a stati

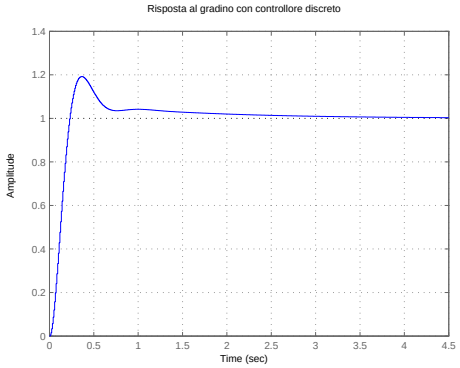


Figura 2.2: Risposta al gradino del processo compensato nel discreto

finiti. Sia r il sgnale d'ingresso al blocco regolatore, e sia g la sua uscita:

$$C(z) = \frac{40.33(1.006 - z^{-1})}{(1.116 - z^{-1})} \tag{2.3.13}$$

$$g(z) = C(z)r(z) \tag{2.3.14}$$

$$1.116g(z) - z^{-1}g(z) = 40.57r(z) - z^{-1}40.33r(z) \tag{2.3.15}$$

$$g(k) = \frac{g(k-1) + 40.57r(k) - 40.33r(k-1)}{1.116} \tag{2.3.16}$$

2.4 Controllo della velocità

La scelta di adottare un controllo della velocità è data dalla necessità di implementare nel robot delle routine di movimento lineare, che inseguono quindi delle traiettorie rettilinee nello spazio. Caratteristiche di questi movimenti sono le costanti rappresentate dai rapporti di velocità dei tre motori che formano il robot. Ne consegue che è assolutamente necessario che ogni singolo motore sia in grado di mantenere una velocità costante per un lasso di tempo determinato.

Discretizzando la P(s) che lega la velocita con la tensione in ingresso data dalla formula (2.2.11) si ha:

$$G(z) = \frac{0.0257z^2 + 1.53e^{-5}}{z^2 - z} \tag{2.4.1}$$

Le specifiche del progetto sono quindi:

- Errore di Velocità: $e_v = 0$.
- Tempo di assestamento: $t_a < 0.1s$.
- Sovraelongazione: $\hat{s} < 20\%$.

E' noto che tali specifiche possono essere espresse in forma matematica esplicita per sistemi di secondo grado. Infatti dato un sistema con poli in catena chiusa definiti da:

$$s^2 + 2\omega_n \xi s + \omega_n^2 \tag{2.4.2}$$

si possono anche definire le seguenti grandezze

$$\hat{s} = 100e^{-\frac{\xi\pi}{\sqrt{1-\xi^2}}} \tag{2.4.3}$$

$$t_a = \frac{3}{\xi\omega_n} \tag{2.4.4}$$

$$\tag{2.4.5}$$

Nel caso si abbia un sistema di grado superiore a due, si possono utilizzare le specifiche date dalle (2.4.3), (2.4.4), come una buona approssimazione di quelle cercate, imponendo che i poli dominanti del sistema a circuito chiuso siano la soluzione dell'equazione (2.4.2). Quindi nel nostro caso si ha che

$$20 \geq 100e^{-\frac{\xi\pi}{\sqrt{1-\xi^2}}} \rightarrow \xi = 0.6 \tag{2.4.6}$$

$$0.1 \geq \frac{3}{\xi\omega_n} \rightarrow \omega_n = 50 \tag{2.4.7}$$

porta ad una coppia di poli complessi e coniugati

$$s = -30 \pm i40$$

Trasformandoli attraverso la trasformazione $z = e^{sT}$ si ha

$$z = 0.7338 \pm i0.2563 \tag{2.4.8}$$

Ne consegue che il regolatore da progettare, deve introdurre nel sistema a catena chiusa, i poli sopra descritti. Ricordando che il denominatore del sistema in catena chiusa è dato da

$$D_w(z) = D_{C(z)}D_{G(z)} + N_{C(z)}N_{G(z)} \tag{2.4.9}$$

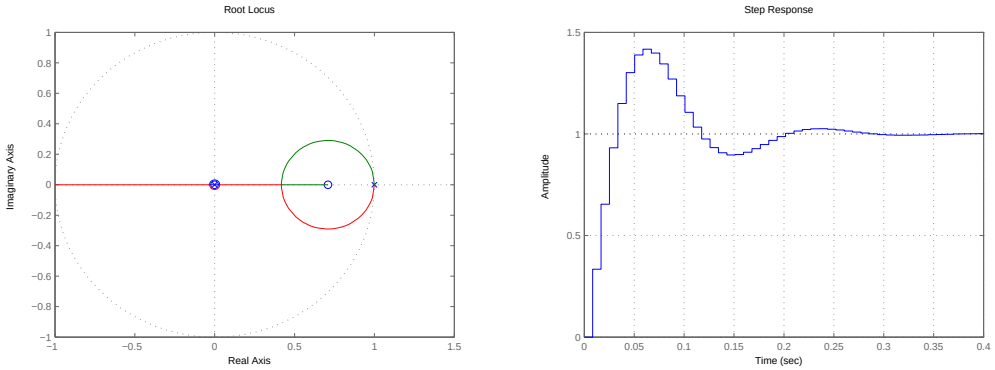
e che per avere un errore in velocità pari a zero è necessario avere due poli in $(z-1)$ in catena aperta, allora il regolatore al primo tentativo è dato da

$$C(z) = 20\frac{z - 0.75}{z - 1} \tag{2.4.10}$$

che introduce una coppia di poli dominanti nel sistema pari a

$$z = 0.7428 - 0.2500i$$

Il luogo delle radici e la risposta a gradino del sistema composto da C(z) e G(z) è:



(a) Luogo delle radici (b) Risposta al gradino

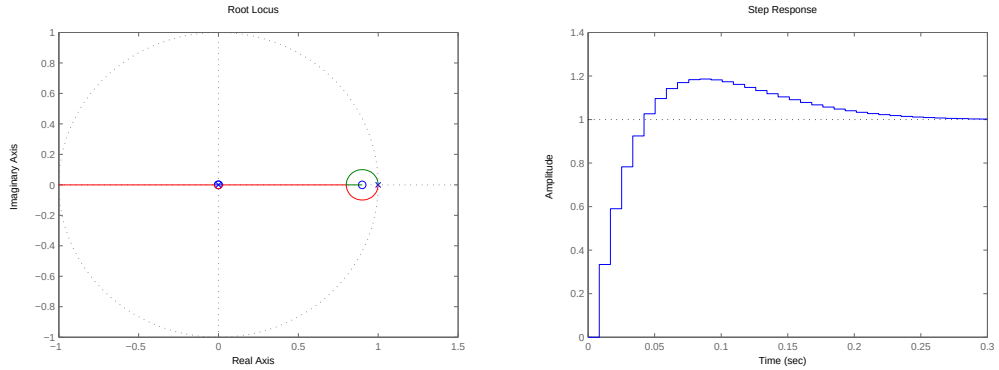
Figura 2.3: Compensatore 1

Analizzando il luogo delle radici in figura 2.3a , e ricordando che la zona a sinistra

dell'origine del piano s, è trasformata attraverso la trasformazione $z = e^{st}$, in una circonferenza di raggio 1 intorno all'origine nel piano z, possiamo dire che il sistema risulta stabile. Ma mentre le specifiche a regime sono verificate, questo non vale per quelle nel transitorio. Infatti risulta una sovraelongazione pari a 40% ed un tempo di assestamento 0.2 s. Per migliorare le prestazioni, come secondo tentativo, si è portato lo zero in $z - 0.9$ con la volontà di spostare il punto singolare più a destra e quindi ridurre i rami complessi del luogo delle radici, diminuendo di fatto la sovraelongazione. Con un regolatore pari a

$$C(z) = 20\frac{z - 0.9}{z - 1} \tag{2.4.11}$$

si ha un luogo delle radici ed una risposta al gradino come mostrati in figura 2.4.



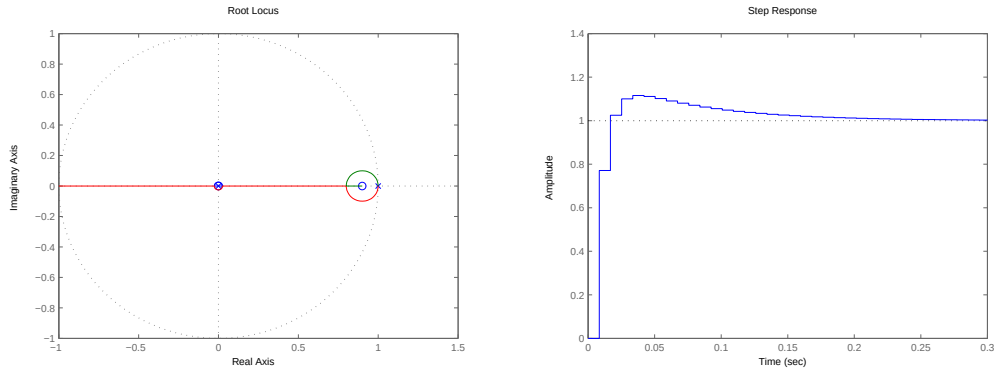
(a) Luogo delle radici (b) Risposta al gradino

Figura 2.4: Compensatore 2

Si nota che mentre la sovraelongazione risulta essere nei limiti del 20%, il tempo di assestamento è invece aumentato. Per modificare tale paramentro aumentiamo il guadagno del regolatore, sapendo che diminuirà ancora la sovraelongazione. Quindi

$$C(z) = 30\frac{z - 0.9}{z - 1} \tag{2.4.12}$$

Analizzando i nuovi grafici del luogo delle radici e della risposta al gradino mostrati in figura 2.5



(a) Luogo delle radici (b) Risposta al gradino

Figura 2.5: Compensatore 3

risuta essere $s < 20\%, t_a < 0.1s$. Si è così progettato un compensatore che renda il sistema stabile e verifica tutte le specifiche di progetto. Un'implementazione come equazione a stati finiti del compensatore è

$$C(z) = \frac{30(z - 0.9)}{z - 1} \tag{2.4.13}$$

$$g(z) = C(z)r(z) \tag{2.4.14}$$

$$zg(z) - g(z) = 30zr(z) - 27r(z) \tag{2.4.15}$$

$$g(k) = g(k - 1) + 30r(k) - 27r(k - 1) \tag{2.4.16}$$

2.5 Implementazione

Due semplici implementazioni in linguaggio C30 di Microchip, per un dsPic a 16bit, dei controllori sopra progettati.

Listing 2.1: Implementazione controllo della velocità

```
unisgned int cp_FLAG;
fractional rz[1];
```

```
fractional gz[1];
fractional v_value;
fractional a;
fractional b;
fractional c;
fractional 1td;
...
a = Q15(0.30);
b = Q15(-0.27)
...
if(cv_FLAG) calculate();
setDCPWM1(1, gz[0]>>4 , 0);
...
velocity=(ENC[1]-ENC[0])*1td
...
void calculate(void){
        rz=v_value-velocity;

        gz[0]=gz[1] + a * rz[0] - b rz[1];

        gz[1]=gz[0];
        rz[1]=rz[0];
        cp_FLAG=0;
}
...
void __attribute__((_interrupt_, no_psv)) T1Interrupt(){
        cv_FLAG=1;
}
```

Listing 2.2: Implementazione controllo della posizione

```
unisgned int cp_FLAG;
fractional rz[1];
fractional gz[1];
fractional s_value;
fractional a;
fractional b;
fractional c;
...
a = Q15(0.4057);
b = Q15(0.4033)
c = Q15(0.008961);
```

```
...
if(cp_FLAG) calculate();
setDCPWM1(1, gz[0]>>4 , 0);
...
void calculate(void){
        rz=s_value - ENC[1];

        gz[0]=(gz[1] + a * rz[0] - b rz[1])*c;

        gz[1]=gz[0];
        rz[1]=rz[0];
        cp_FLAG=0;
}
...
void __attribute__((_interrupt_, no_psv)) T1Interrupt(){
        cp_FLAG=1;
}
```

2.6 Interfaccia di controllo

2.6.1 Interfaccia di basso livello

L'interfaccia a basso livello risiede nell'elettronica di comando, dove sono implementati i sistemi di controllo. Si tratta di una vera e propria API che serve per la comunicazione tra l'interfaccia di alto livello e i sistemi di controllo. La particolarità di tali chiamate di sistema è che si basano su una tecnica di polling, infatti il sistema operativo dei microcontrollori è uno pseudo-realtime. E' quindi necessario disabilitare quasiasi tipo di interrupt che interromperebbe le routine di controllo. Di seguito sono riportati i comandi delle API dibasso livello.

COMANDO	DESCRIZIONE
G,xxxx,yyyy,zzzz	Impone ai motori di raggiungere gli angoli definiti da xxxx,yyyy,zzzz
V,xxx,1/2/3	Impone ai motori 1 o 2 o 3 una velocità xxx in rpm
R,x,1/0	Attiva o disattive il relay x
S,x,1/2/3	Impone una legge di controllo al motore 1,2,3
T,xx	Setta il tempo di campionamento in ms
E	Ferma i motori, interrompe anche le routine PID, comando di emergenza
L,xx,yy,zz,tttt	Movimento lineare alle coordinate cartesiane xx,yy,zz, in tempo ttt in secondi
P,x/s	Aspetta la fine del movimento o aspetta x secondi

2.6.2 Interfaccia di alto livello

L'interfaccia di alto livello è anche detta HMI, Human Machine Interface, si tratta di un programma risiedente sul server a cui è connesso il robot. Il software è sviluppato in HTML5 - PHP - Ajax e permette di:

- Inviare comandi delle api a basso livello.
- Editor di programma.
- Settaggio di tutti i parametri del robot.
- Framework per la costruzione di nuovi Tool.
- Controllo di ogni parametro del movimento.

Il layout si presenta come un Window Manager, dove ogni finestra rappresenta un Tool. Inoltre il programma server si occupa di risolvere i problemi di cinematica inversa e diretta.

Capitolo 3

Elettronica

3.1 Hardware

3.1.1 Schema generale

In figura 3.1 è riportato lo schema logico dell'elettronica di potenza e di comando. Il funzionamento è semplice, il SERVER risolve i problemi legati alla cinematica

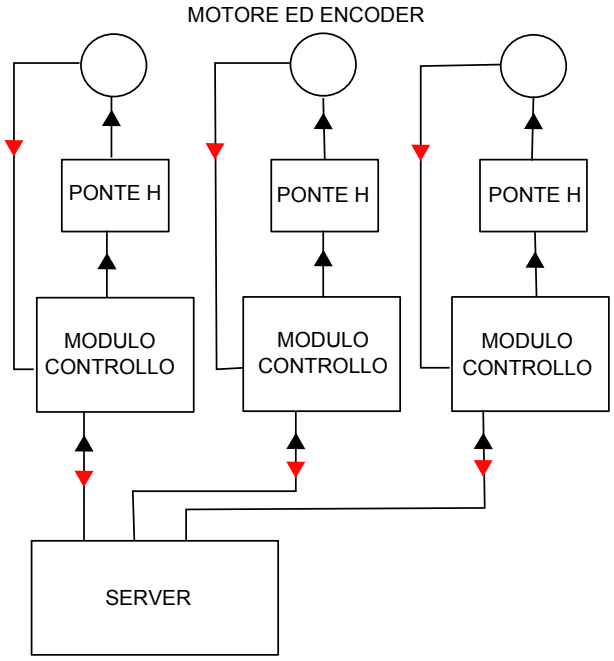


Figura 3.1: Schema Logico

inversa e diretta, ed invia i dati ai moduli di controllo; inoltre riceve dagli stessi le

informazioni per HMI. Infatti questa risiede proprio nel server. Nei moduli di comando è implementato il regolatore studiato nel precedente capitolo, inoltre vi è una routine parser che implementa l'api di basso livello. Ogni modulo di comando comunica con un Ponte H, o modulo di potenza, tramite un'uscita analogia (pwm in lap-mode), riceve inoltre il feedback dall'encoder collegato al motore. Infine il modulo di potenza o PONTE H, trasforma il segnale in ingresso dal modulo di controllo in una tensione costante che va ad applicare ai capi del motore.

3.1.2 Modulo controllo

Il modulo di controllo si basa su una scheda prodotta da Droids SAS, che implementa un microcontrollore di Microchip dsPIC33FJ128MC802. Inoltre possiede un pinout ottimale per il controllo dei motori. Le caratteristiche del microcontrollore sono eccellenti, si tratta infatti di uno dei più potenti in commercio, ha un'architettura 16bit con un clock fino a 80Mhz con 40MIPS due periferiche CAN e due moduli QEI (Quadrature Encoder Interface), 2 moduli PWM da 8 canali. Inoltre possiede un core DSP, che permette di effettuare moltiplicazioni in maniera diretta. Il dsPic si programma in Assembler Microchip o C30, un fork di GCC che rispetta le ANSI C, inoltre possiede una nutrita collezione di librerie standard per la gestione delle periferiche.
L'implementazione del sistema di controllo è stata fatta rispettando le specifiche per l'utilizzo del core DSP. Infatti è necessario utilizzare una notazione numerica a 15Bit + 1 di segno in fixed-point (Q15); anche se poco agevole, tali accorgimenti sono risultati estremamente utili, tanto da poter dimezzare il tempo di campionamento del sistema, in quanto la velocità di escuzione della routine di controllo è estremamente breve (3+10 cicli di clock per il PID, crica 20 per il controllo di posizione). Il sistema operativo creato nel dsPIC è uno pseudo-RTOS: esso infatti disabilita tutti gli inter-

rupt tranne quelli necessari al sistema stesso. Inoltre le comunicazioni avvengono con una tecnica di polling, ed il feedback dell'encoder, viene gestito da un modulo DMA, quindi eseguito parallelamente al processore. I regolatori agiscono su un registro del modulo PWM, chiamato Duty Cycle. Tale registro va a specificare l'ampiezza dell'onda quadra in uscita da dsPIC. Tale onda va in ingresso al modulo di potenza. Si vedrà in seguito che tale ampiezza è proporzionale alla tensione applicata ai motori.

3.1.3 Modulo potenza

Il modulo di potenza, è in pratica un Doppio Ponte H. Il suo schema logico si può schematizzare come in figura 3.2. Prendendo in ingresso un segnale ad onde quadre di ampiezza d, esso porta in uscita un valore di tensione proporzionale a d stesso. Le caratteristiche di cui tener conto nella scelta di un Ponte H sono quelle di portata della corrente. In questo caso abbiamo scelto un Doppio Ponte H con 4A di corrente massima complessiva, basato sull'integrato L298, uno tra i più diffusi e collaudati sul mercato.

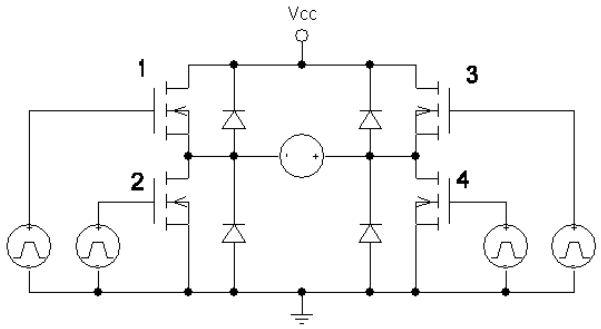


Figura 3.2: Schema logico di un PonteH

3.1.4 Server

Il server è costituito da un vero e proprio calcolatore. Esso si occupa di implementare l'interfaccia HMI ed i problemi di cinematica inversa e diretta. Per quanto riguarda

l'interfaccia HMI, si tratta di un vero e proprio sito internet sviluppato in PHP ed Ajax, che permette l'interfacciamento tra l'uomo e il sistema di controllo. Il web server installato nel server è Apache 2.2 e grazie alla classe PHP-Serial esso può essere messo in comunicazione con le porte seriali. Sul server sono attivi sempre alcuni demoni, che gestiscono per lo più le routine che potrebbero danneggiare i moduli di comando, per esempio le routine di emergenza (Stop, Spegnimento), e alcuni demoni che invece gestiscono le comunicazioni.

3.1.5 Comunicazione

Si possono distinguere due tipi di comunicazioni. La prima è quella tra i moduli di comando ed il server, ed avviene per via seriale. La seconda è quella tra i demoni di comunicazione seriale e l'interfaccia HMI, ed avviene via database.

La comunicazione tra i moduli ed il demone di comunicazione, come già detto, avviene per via seriale, ogni modulo di controllo possiede due periferiche seriali, una di esse è stata collegata ad una porta seriale del calcolatore attraverso un convertitore RS232-USB basato su un FTD232R. I comandi inviati e ricevuti seguono una sintassi ben precisa. I demoni, come anche i moduli, non hanno la possibilità di ritardare l'invio o la ricezione dei comandi, altrimenti si andrebbe a violare la specifica di Real-Time, tale capacità però può essere attuata dall'interfaccia HMI. Infatti questa dà la possibilità di processare un serie di comani da inviare a distanze temporali diverse.

La sincronizzazzione tra HMI e moduli di comando avviene per via indiretta, infatti questi ultimi, se non diversamente specificato, alla fine di ogni comando eseguito, inviano ai demoni di comunicazione un comando speciale che abilita l'invio di un nuovo comando. A sua volta i demoni inoltrano tale richiesta all'interfaccia HMI che procede con il nuovo reinvio. Vista la differenza di velocità di esecuzione tra moduli

e l'interfaccia, i metodi di comunicazione tra quest'ultima e i demoni implementano uno stack. Infatti oltre ai comandi di movimento e i relativi riscontri, nel database di appoggio passano anche le informazioni relative alla posizione attuale dei motori, la frequenza degli arrivi,quindi, può essere maggiore rispetto la velocità di smaltimento. Di conseguenza per non perdere informazioni altrimenti utili (si potrebbe perdere un riscontro di fine movimento), il database funziona come uno stack.

Per quando riguarda la comunicazione tra HMI e demoni di comunicazione essa avviene via database. Questa scelta è nata dal fatto che mentre i demoni sono una forma di programmazione lato server, che implementa codice PHP ed SQL, l'interfaccia viene gestita via browser, quindi con tecnologia javascript ed Ajax, ossia via client.

Una scelta di questo tipo di interfaccia è stata quasi obbligata, infatti il progetto LGTP2411 va a collocarsi all'interno di una linea produttiva che è gestita informaticamente da applicazione web-based. E' possibile quindi utilizzare il robot per via remota collegandosi appunto all'interfaccia tramite rete wifi o ethernet.

Capitolo 4

Applicazione

Il progetto LGTP2411 è parte integrante di un sistema di produzione basato su software web. Infatti il robot descritto dal progetto si colloca nel mezzo di una linea di produzione di pulsanti e kit per ascensori. L'automa progettato ha il compito di leggere attraverso un web-software le commesse in entrata al reparto lavorativo, e prelevare i componenti semilavorati da un'apposita struttura già presente in industria, per posizionarli all'interno di fustelle in PVC, che andranno, dopo un opportuno controllo da parte dell'operatore, direttamente in spedizione al cliente. Un altro impiego del robot è quello dell'assemblaggio di kit di viteria. Infatti tale lavorazione è caratterizzata dalle posizioni fisse dei componenti che la compongono, è quindi possibile automatizzare attraverso pick and place tale operazione. E' altresì in via di sviluppo l'implementazione nel progetto LGTP2411 di un sistema di visione artificiale che consente di eliminare i vincoli di preposizionamento dei componenti da prelevare e rilasciare.

Capitolo 5

Conclusioni e sviluppi futuri

La tesi ha raggiunto gli obbiettivi previsti, progettando e realizzando un robot delta ed il suo sistema di controllo. Producendo una precisione di posizionamento inferiore al millimetro. E' necessario però, per una corretta implementazione del progetto preposto per la tesi, nell'ambito della produzione industriale, un sistema elettronico più robusto e potente; a tal proposito si sono individuati i seguenti miglioramenti da eseguire su progetto:

- Sostituzione del calcolatore server, con una Fox Board G20 basata su ARM9.
- Sostituzione dei moduli di comando con un a scheda di National Istrument basata su ARM Cortex M3.
- Implementazione di un sistema di visione artificiale per la determinazione delle posizioni da raggiungere.
- Nuovo sistema di comunicazione basato su CAN ed Ethernet industriale.

Tali miglioramenti oltre ad irrobustire il sistema elettronico per portarlo ad avere requisiti fondamentali per l'inserimento nella prduzione industriale, offre la possibilità di interfacciare il sistema server con altre apparecchiature.

Appendice A

Alcune tracce del sorgente utilizzato nell'elettronica di comando.

Listing 5.1: main.c

```
int main(void) {

        Clock();
        Remap();
        Setting_Uart2_DMA();
        Setting_PWM();
        Setting_QEI();
        Setting_IC();
        Settings_ISR();
        init_pid1();
        init_Pos();
        Rx2Count=CmdCount2=ENC[0]=ENC[1]=0;
        theta[0]=0;
        theta_enc[0]=0;
        STEP = 2.387324147;
        StopFLAG=0;
        WaitFLAG=0;
        cnt=1;
        Ts=0.00084;
        DMA0_Tx_Send("--LGTP2411--\r\n");
        while(StopFLAG==0) {
                _RA4=1;
                if(Rx2FLAG) Rx2();

                if(ParserFLAG) Parser();
                if(PID1_FLAG) pid_p();
                if(POS_FLAG) Pos();
                if(time[0]>100){
                        debug();
                        time[0]=0;
```

```
                }
        }
        return 0;
}

void Rx2(void){
                        Rx2Buff[Rx2Count]=U2RXREG;
                        if(Rx2Buff[Rx2Count]=='@'){
                                CmdOffset2=Rx2Count;
                        }
                        if(Rx2Buff[Rx2Count]=='#'){
                                CmdCount2=Rx2Count;
                                Rx2Count=0;
                                ParserFLAG=1;
                        if(WaitFLAG && MotFLAG[0])
                                DMA0_Tx_Send("@P,1#");
                        }
                        Rx2Count++;
                        Rx2FLAG=0;
}

void Parser() {
        switch(Rx2Buff[CmdOffset2+1]) {
                case 'G':
                                        tmp[0]=Rx2Buff[CmdOffset2+3];
                                        tmp[1]=Rx2Buff[CmdOffset2+4];
                                        tmp[2]=Rx2Buff[CmdOffset2+5];
                                        tmp[3]=Rx2Buff[CmdOffset2+6];
                                        tmp[4]=Rx2Buff[CmdOffset2+7];
                                        theta[0]=atoi(tmp);
                                        theta_enc[0] = (STEP * theta[0])
                                        break;

                case 'P':       WaitFLAG=1;
                                        break;
                case 'I':       tmp[0]=Rx2Buff[CmdOffset2+3];
                                        tmp[1]=Rx2Buff[CmdOffset2+4];
                                        tmp[2]=Rx2Buff[CmdOffset2+5];
                                        tmp[3]=Rx2Buff[CmdOffset2+6];
                                        Ki=atof(tmp);
                                        kCoeffs[1]=Ki*327.68;
                                        PIDCoeffCalc(&kCoeffs[0], &PID1);
```

```
                                        break;
                case 'D':       tmp[0]=Rx2Buff[CmdOffset2+3];
                                        tmp[1]=Rx2Buff[CmdOffset2+4];
                                        tmp[2]=Rx2Buff[CmdOffset2+5];
                                        tmp[3]=Rx2Buff[CmdOffset2+6];
                                        Kd=atof(tmp);
                                        kCoeffs[2]=Kd*3276.8;
                                        PIDCoeffCalc(&kCoeffs[0], &PID1);
                                        break;
                case 'M':
                                        tmp[0]=Rx2Buff[CmdOffset2+3];
                                        tmp[1]=Rx2Buff[CmdOffset2+4];
                                        tmp[2]=Rx2Buff[CmdOffset2+5];
                                        tmp[3]=Rx2Buff[CmdOffset2+6];
                                        Kp=atof(tmp);
                                        kCoeffs[0]=Kp*32768;
                                        PIDCoeffCalc(&kCoeffs[0], &PID1);
                                        break;
                case 'E':       StopFLAG=1;
                                        MotFLAG[0]=0;
                                        break;
                case 'V':       tmp[0]=Rx2Buff[CmdOffset2+3];
                                        tmp[1]=Rx2Buff[CmdOffset2+4];
                                        tmp[2]=Rx2Buff[CmdOffset2+5];
                                        tmp[3]=Rx2Buff[CmdOffset2+6];
                                        duty[0]=atoi(tmp);
                                        SetDCMCPWM1(1,duty[0],0);
                                        break;
                default:        break;
                }

        ParserFLAG=0;
}

void init_pid1(void) {
        PID1.abcCoefficients = &abcCoefficient[0];      /*Set up pointer to
    PID1.controlHistory = &controlHistory[0];       /*Set up pointer to co
    PIDInit(&PID1);
        /*Kp, Ki and Kd vale is in float*fractional value, so for
        Kp: 0.99*2^15 = 32440
        Ki: 0.0025*2^15 = 81
        Kd: 0.007*2^15 = 229
```

```
        the numerator of PID fdt is (kp+ki+kd)-(z^-1)(kp+2kd)+(z^-2)kd fo
                a=kp+ki+kd < |1|
                b=kp+2kd  < |1|
                c=kd < |1|
        cause the 1*2^15 exceeds the 15bit space.
        */
        kCoeffs[0] = Q15(0.99);
        kCoeffs[1] = Q15(0.012);
        kCoeffs[2] = Q15(0.008);
        PIDCoeffCalc(&kCoeffs[0], &PID1);                /*Derive the a,b, &
        ctrOut[1]=0;

}


void pid_p(){
                PID1_FLAG=0;
                PID1.controlReference = (theta_enc[0]);
                PID1.measuredOutput = ENC[0];
        PID(&PID1);
                /* Anti - WindUP system */
                if(PID1.controlOutput>MAXERROR)
                PID1.controlOutput=MAXERROR;
        if(PID1.controlOutput<MINERROR)
                PID1.controlOutput=MINERROR;
                duty[0] = PID1.controlOutput + 2048;
                if(duty[0] < 0) duty[0]=1;
                if(duty[0] > 4096) duty[0]=4095;
                SetDCMCPWM1(1, duty[0], 0);
}

void init_Pos(){
a=40.57;
b=-40.17;
c=0.896;
yf[0]=0;
}


void Pos(void){
        e[0]=theta_enc[0]-ENC[0];
        y[0]=(y[1]+a*e[0]+b*e[1])*c;
```

```
/*FINE REGULATION-------------------------------*/
        if(e[0] < 500 && e[0] > 5){
                if(y[0]<0) yf[0]+=(-y[0]/e[0])*MotFLAG[0];
                else yf[0]+=(y[0]/e[0])*MotFLAG[0];
                duty[0]=y[0] + yf[0] + 2048;
        } else if(e[0] > -500 && e[0] < -5){
                if(y[0]<0) yf[0]+=(-y[0]/e[0])*MotFLAG[0];
                else yf[0]+=(y[0]/e[0]*MotFLAG[0]);
                duty[0]=y[0] + yf[0] + 2048;
        }
/*-----------------------------------------------*/
        else{
                if(y[0] > 20000) duty[0] = 4095;
                else if (y[0] < -20000) duty[0] = 1;
                else duty[0]= y[0] + 2048;
        }
        if(duty[0] < 0) duty[0]=1;
        if(duty[0] > 4096) duty[0]=4095;
        SetDCMCPWM1(1,duty[0],0);

        if((e[0] <= 5) && (e[0]==e[1]) && (e[0] >= 0))
                MotFLAG[0]=0;
        if((e[0] >= -5) && (e[0]==e[1]) && (e[0] <= 0))
                MotFLAG[0]=0;

        e[1]=e[0];
        y[1]=y[0];
        POS_FLAG=0;
}
```

Listing 5.2: setting.c

```
unsigned int BufferA[120] __attribute__((space(dma)));
unsigned int BufferB[120] __attribute__((space(dma)));

#define FCY 40000000
#define BAUDRATE 115200
#define BRGVAL ((FCY/BAUDRATE)/16) - 1

void Remap(void){
    TRISA = 0x0000;         //All PortA pins set to output
        AD1PCFGL = 0xFFFF;        //Set all ADC ports to digital I/O mode
        LATA  = 0x0000;   //Set PortA latch values to zero
        LATB = 0x0000;
```

```
    PORTA = 0x0000;        //Set PortA latch values to zero *Write to port
    TRISB = 0b0000000000000000;

        _TRISB3=1; //U2RX

        _TRISB10=1; //QEA1
        _TRISB11=1; //QEB1

        PPSUnLock;
        PPSInput(PPS_U2RX, PPS_RP3);
        PPSOutput(PPS_U2TX, PPS_RP2);
        PPSInput(PPS_IC1,PPS_RP10);
        PPSInput(PPS_QEA1,PPS_RP10);
        PPSInput(PPS_QEB1,PPS_RP11);
        PPSLock;
}

void Clock(void){
PLLFBD=30;                                                    // M=32
CLKDIVbits.PLLPOST=0;              // N1=2
CLKDIVbits.PLLPRE=0;               // N2=2
// Disable Watch Dog Timer
RCONbits.SWDTEN=0;

// Clock switching to incorporate PLL
__builtin_write_OSCCONH(0x03);            // Initiate Clock Switch to Prim
                                          // Oscillator with PLL (NOSC=0b0
__builtin_write_OSCCONL(0x01);            // Start clock switching

#ifndef SIM      // [21]
        while (OSCCONbits.COSC != 0b011);        // Wait for Clock switch

        while(OSCCONbits.LOCK!=1) {};            // Wait for PLL to lock
#endif

#define TMR1_VALUE 40000
        OpenTimer1(T1_ON &
                        T1_GATE_OFF &
                        T1_PS_1_1 &
                        T1_SYNC_EXT_OFF &
                        T1_SOURCE_INT,
                        TMR1_VALUE);
```

```
#define TMR2_VALUE 0xFFFF
T2CONbits.TON = 0;                  // Disable Timer
T2CONbits.TCS = 0;                  // Select internal instruction cycle cloc
T2CONbits.TGATE = 0;      // Disable Gated Timer mode
T2CONbits.TCKPS = 0b00; // Select 1:1 Prescaler
TMR2 = 0x00;                        // Clear timer register
PR2 = TMR2_VALUE;                   // Load the period value
}

void Setting_Uart2_DMA(void){
U2MODEbits.UARTEN = 0;
U2MODEbits.STSEL = 0; // 1 Stop bit
U2MODEbits.PDSEL = 0; // No Parity, 8 data bits
U2MODEbits.ABAUD = 0; // Auto-Baud Disabled
U2BRG = BRGVAL;// BAUD Rate Setting for 115200
U2STAbits.UTXISEL0 = 0; // Interrupt after one TX character is transmitte
U2STAbits.UTXISEL1 = 0;
U2STAbits.URXISEL = 0; // Interrupt after one RX character is received
IPC7 = 0x4400;
IFS1bits.U2RXIF = 0;      // Clear the Recieve Interrupt Flag
IEC1bits.U2RXIE = 1;      // Enable Recieve Interrupts
_U2EIF = 0;                         // Clear UART2 error interrupt Flag
_U2EIE = 1;                         // Enable UART2 error interrupt
U2MODEbits.UARTEN = 1;   // Enable UART
U2STAbits.UTXEN = 1;      // Enable UART TX

//Set Up DMA Channel 0 to Transmit in One-Shot, Single-Buffer Mode:
DMA0CON = 0x2001; // One-Shot, Post-Increment, RAM-to-Peripheral
DMA0CNT = 7; // 8 DMA requests
DMA0REQ = 0x001F; // Select UART2 Transmitter
DMA0PAD = (volatile unsigned int) &U2TXREG;
DMA0STA = __builtin_dmaoffset(BufferA);
IFS0bits.DMA0IF = 0; // Clear DMA Interrupt Flag
IEC0bits.DMA0IE = 1; // Enable DMA Interrupt
}

void Setting_PWM(void) {

unsigned int period;

unsigned int sptime;
```

```
unsigned int config1;
unsigned int config2;
unsigned int config3;

period = 2048;
sptime = 0x0;
config1 =       PWM1_EN & PWM1_IDLE_CON & PWM1_OP_SCALE1 & PWM1_IPCLK_SCA
                PWM1_MOD_FREE;

config2 =       PWM1_MOD1_COMP & PWM1_PEN1L & PWM1_PEN1H & PWM1_PDIS3H &

config3 =       PWM1_SEVOPS1 & PWM1_OSYNC_PWM & PWM1_UEN;
OpenMCPWM1(period, sptime, config1, config2, config3);

P1DTCON2bits.DTS1A = 0;
P1DTCON2bits.DTS1I = 0;
P1DTCON2bits.DTS2A = 0;
P1DTCON2bits.DTS2I = 0;

SetMCPWM1DeadTimeGeneration(PWM1_DTA4 & PWM1_DTAPS1);
SetDCMCPWM1(1, 2048, 0);
PWM2CON1bits.PEN1L = 0;
PWM2CON1bits.PEN1H = 0;

}


void Setting_QEI(void){

QEI1CONbits.QEIM        = 7;    //      QEI_MODE_x4_MATCH
QEI1CONbits.SWPAB       = 0;    //      QEI_INPUTS_SWAP
QEI1CONbits.QEISIDL     = 1;    //      QEI_IDLE_STOP
QEI1CONbits.POSRES      = 0;    //      QEI_INDEX_RESET_DISABLE
QEI1CONbits.PCDOUT      = 0;    //      QEI_NORMAL_IO
QEI1CONbits.POSRES      = 0;    //      POS_CNT_ERR_INT_DISABLE

DFLT1CONbits.QECK       = 6;    //      QEI_QE_CLK_DIVIDE_1_128
DFLT1CONbits.QEOUT      = 1;    //      QEI_QE_OUT_ENABLE

MAX1CNT = 0xFFFF;
POS1CNT = 0;
}
```

```
void Setting_IC(void){

IC1CONbits.ICSIDL = 1;   //      Stop in idle
IC1CONbits.ICTMR = 1;    //      Timer 2
IC1CONbits.ICI = 0;              //      Interrupt on every capture event
IC1CONbits.ICM = 3;              //      Capture mode every rising edge

}


void Settings_ISR(void)
{
ConfigIntMCPWM1(PWM1_INT_DIS);
ConfigIntCapture1(IC_INT_ON & IC_INT_PRIOR_4);
ConfigIntTimer1(T1_INT_PRIOR_4 & T1_INT_ON);
PR1 = TMR1_VALUE;
}
```

Elenco delle figure

Bibliografia

[1] Prof. Paul Zsombor-Murray, *Descriptive Geometric Kinematic Analysis of Clavel's 'Delta' Robot"*, McGill University, 2004.

[2] C. Bonivento - C. Melchiorri - R. Zanasi, *Sistemi di controllo digitale"*, Progetto Leonardo, 1995.

[3] O.M.Grasselli - L.Menini - S.Galeani, *Sistemi dinamici"*, Hoepli, 2009.

[4] N.P. Belfiore - A. Di Benedetto - E. Pennestri', *Fondamenti di Meccanica Teorica e Applicata"*, Ambrosiana,

[5] A. Di Benedetto - E. Pennestri', *Introduzione alla Cinematica dei Meccanismi"*, Ambrosiana,

[6] E. Pennestri', *Dinamica Tecnica e Computazionale"*, Ambrosiana,

[7] Microchip', *dsPIC33FJ128MC802 datasheet"*, www.microchip.com,

[8] G.Ottaviani', *dsNavCon Project"*, http://code.google.com/p/dspid33/,