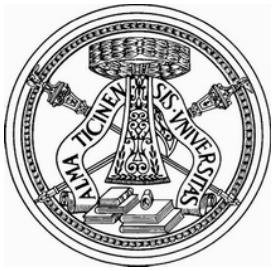
Università degli Studi di Pavia

FACOLTÀ DI INGEGNERIA
Corso di Laurea in Ingegneria Informatica

Progettazione, realizzazione e controllo di un carroponte



Relatore:
Ing. Lalo Magni

Correlatore:
Ing. Gianluca de Felici

Tesi di Laurea di:
Marco FORGIONE

Anno Accademico 2006-2007

Sommario

Questa tesi descrive la realizzazione di un modello in scala di un carroponte e l'implementazione su calcolatore di un relativo controllo. L'attività si è svolta presso il laboratorio di Controllo dei Processi, afferente al Dipartimento di Informatica e Sistemistica dell'Università degli Studi di Pavia.

Indice

1 Introduzione

In questo primo capitolo sono brevemente descritti il funzionamento generale di un carroponte, il dispositivo sperimentale da noi costruito e gli obiettivi generali di controllo che ci poniamo.

1.1 Il carroponte

Il carroponte, detto anche gru a ponte (in inglese gantry crane) è una macchina destinata al sollevamento e allo spostamento di materiali di vario tipo sia in ambienti aperti che al chiuso. Gli usi più comuni sono all'interno delle fabbriche e dei magazzini, ma anche in altri ambiti quali i depositi ferroviari e marittimi. Esistono modelli per portate che variano da centinaia di chili a decine di tonnellate.

In un carroponte il carico è attaccato tramite un cavo a un carrello, il quale può muoversi in un piano orizzontale, ad una certa altezza dal suolo. Il movimento del carrello avviene nel piano lungo due direzioni perpendicolari. Esso può infatti percorrere un binario rettilineo, che a sua volta può traslare rigidamente in direzione perpendicolare alla sua lunghezza, trascinando ovviamente anche il carrello nel suo moto. La lunghezza del cavo è a sua volta variabile: il carico può quindi essere spostato nelle tre dimensioni.

Figura 1.1: Carroponte all'interno di un deposito

1.2 Il dispositivo sperimentale

Il dispositivo sperimentale è stato costruito nel laboratorio di Controllo dei Processi dell'Università di Pavia. Sopra una struttura portante a forma di parallelepipedo è collocato, sulla faccia superiore, il binario. Esso è orientato parallelamente all'asse che d'ora in poi chiameremo x. Il binario stesso si può muovere lungo l'asse y, perpendicolare a x e appartenente allo stesso piano orizzontale. Definiamo inoltre l'asse z, rivolto verso l'alto, in modo da formare una terna destra, la cui origine è posta convenzionalmente al centro della faccia rettangolare superiore. La movimentazione è permessa da tre motori in corrente continua (DC): un motore muove il binario, un altro muove il carrello e un terzo varia la lunghezza del filo. Il sistema è fornito di sensori di posizione angolare (degli encoder) collegati all'albero dei motori e di una telecamera in grado di rilevare i due angoli in figura 1.2: α e β. Questi determinano l'orientamento del cavo che sorregge il carico. Assieme all'informazione sulla lunghezza del filo, ottenibile dal relativo encoder, è così possibile determinare la posizione del carico relativa rispetto al carrello. Il sistema che implementa la rilevazione degli angoli è stato sviluppato sempre nel nostro laboratorio, e una tesi è stata svolta riguardo questo argomento. Il carroponte è fornito infine dell'elettronica opportuna per un pilotaggio automatico o manuale.

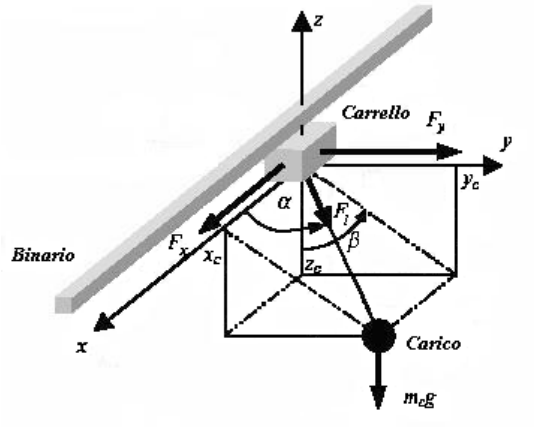


Figura 1.2: Angoli di oscillazione α e β

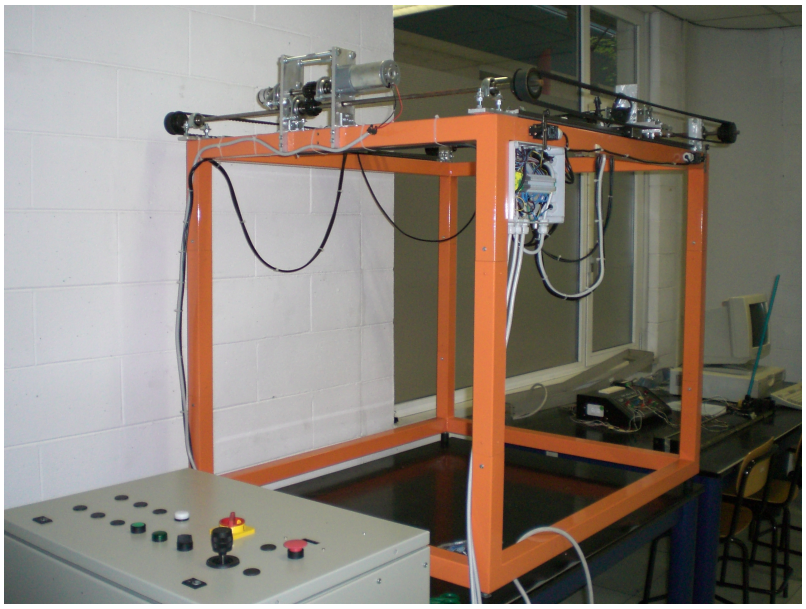


Figura 1.3: Il carroponte all'interno del laboratorio

1.3 Il controllo

Lo scopo primario di controllo è spostare il carico tra due diversi punti nello spazio. Vi è inoltre il vincolo di mantenere abbastanza piccoli gli angoli α e β per tutta la durata del moto, per ragioni di sicurezza. Lo spostamento deve inoltre avvenire in tempi abbastanza rapidi, nel rispetto delle condizioni sugli angoli. Purtroppo non è stato possibile dedicare molto tempo alla realizzazione del controllo, impiegato in gran parte nella realizzazione meccanica ed elettrica. Per questo motivo è stato sviluppato un controllo non particolarmente raffinato, ma comunque efficace a spostare il carico con grande precisione. Non si è fatto in tempo invece a implementare un controllo sugli angoli di oscillazione.

2 Modellizzazione

In questo capitolo è modellizzato il carroponte dal punto di vista della meccanica e dei motori. Come in ogni modellizzazione finalizzata al controllo, è necessario mantentere un giusto compromesso tra semplicità e completezza. Occorre infatti avere un livello di dettaglio sufficiente a garantire un'azione di controllo efficace, senza tuttavia complicare eccessivamente il modello e prendere in considerazione effetti di entità trascurabile.

2.1 Il modello meccanico

Prendiamo dunque in analisi la parte meccanica. A questo livello di analisi, si considerano sono tre corpi: carrello, binario e carico. Su di esso agiscono forze di vario tipo: vincolari, di attrito, gravitazionali e forze fornite dai motori. Il sistema obbedisce ovviamene alle semplici leggi della meccanica newtoniana. Non è tuttavia praticabile la strada dell'ispezione diretta delle forze in gioco, principalmente per la complessa relazione di forze vincolari. Una teoria adatta ad affrontare problemi di questo tipo è la Meccanica Lagrangiana.
E' oltre gli scopi della trattazione discutere questa teoria dal punto di vista teorico, argomento trattato esaustivamente in testi come [3]. Esponiamo dunque brevemente, senza pretese di completezza, i risultati che saranno utilizzati per ricavare le equazioni di moto. Data la complessità dei conti simbolici è stato necesssario l'uso del software Maple 10.

2.1.1 Le Equazioni di Lagrange

Le equazioni di Lagrange consentono di ottenere la descrizione matematica della dinamica di un sistema meccanico costituito da un'insieme di corpi e

sottoposto a un certo numero di vincoli, sotto l'azione di un certo numero di forze attive agenti, che possono essere sia conservative che non conservative. In generale, un sistema di N corpi rigidi, non soggetti a vincoli, ha nello spazio $6N$ coordinate indipendenti o gradi di libertà: $3N$ di traslazione e $3N$ di rotazione.

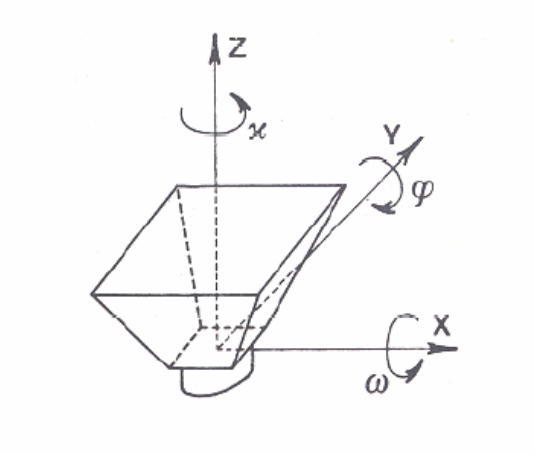


Figura 2.1: Gradi di libertà di un corpo rigido

Per definire la posizione di ogni singolo corpo nello spazio è quindi necessario fissare 6 parametri; nel caso in cui il corpo sia vincolato a non ruotare, risultano sufficienti i 3 parametri traslazionali che possiamo far corrispondere alle coordinate cartesiane. Sotto tale ipotesi, un sistema di N corpi rigidi ha $3N$ gradi di libertà. Se il sistema presenta ulteriori vincoli, il numero di gradi di libertà si riduce ancora. Nell'ipotesi in cui i vincoli siano olonomi, ovvero descrivibili tramite k equazioni indipendenti del tipo:

$$f(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_i, \ldots, \vec{r}_N, t) = 0,$$

si possono individuare $n = 3N - k$ variabili indipendenti $q_1, q_2, \ldots, q_n$ dette coordinate lagrangiane (o coordinate generalizzate), sufficienti per descrivere il sistema. Le coordinate cartesiane $\vec{r}_i$ $(i = 1, 2, \ldots, N)$ descriventi la posizione degli N corpi sono esprimibili, in termini di coordinate lagrangiane, mediante relazioni del tipo:

$$\vec{r}_i = \vec{r}_i(q_1, q_2, \ldots, q_j, \ldots, q_n, t),$$

Indicando con T l'energia cinetica totale del sistema e con V l'energia potenziale totale del sistema, determinata dalle forze attive conservative, si definisce

la Lagrangiana L del sistema:

$$L = T - V \tag{2.1}$$

Nel caso in cui non vi siano forze attive di tipo non conservativo, si possono scrivere n equazioni di Lagrange nella forma:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_j} - \frac{\partial L}{\partial q_j} = 0 \qquad j = 1, 2, \ldots, n \tag{2.2}$$

dove q_j indica la j-esima coordinata lagrangiana. Se invece esistono forze attive non conservative, le equazioni di Lagrange diventano:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_j} - \frac{\partial L}{\partial q_j} = Q_j \qquad j = 1, 2, \ldots, n \tag{2.3}$$

dove i termini Q_j sono le componenti delle forze attive non conservative rispetto alle coordinate q_j e si calcolano secondo la formula:

$$Q_j = \sum_{i=1}^{N} \vec{F}_i \cdot \frac{\partial \vec{r}_i}{\partial q_j} \tag{2.4}$$

in cui $\vec{F}_i$ è l'insieme delle forze attive non conservative agenti sul corpo i-esimo. Tra queste bisogna tenere conto anche gli eventuali attriti, in quanto non sono previsti di per sè dal modello lagrangiano. Si noti che, così come le q_j non hanno necessariamente le dimensioni di una lunghezza, le Q_j non hanno necessariamente le dimensioni di una forza, ma che $Q_j \delta q_j$ ha sempre le dimensioni di un lavoro [3]. Questo è perlaltro indispensabile per la correttezza dimensionale delle equazioni (2.3).

2.1.2 Ipotesi

Ipotizziamo per i tre corpi la presenza di attrito esclusivamente viscoso, ovvero proporzionale alla velocità, e indipendente dalla direzione (isotropo). Per quanto riguarda la dinamica del carico, ipotizzeremo che il filo sia sempre ben teso. Ciò è realistico per moderate oscillazioni.

2.1.3 Nomenclatura

Indichiamo con $\vec{r}_m$ la posizione del carico, con $\vec{r}_c$ la posizione del carrello e con $\vec{r}_b$ la posizione del binario. Si ha dunque:

$$\begin{aligned} \vec{r}_m &= (x_m) \cdot \vec{i} + (y_m) \cdot \vec{j} + (z_m) \cdot \vec{k} \\ \vec{r}_c &= (x_c) \cdot \vec{i} + (y_c) \cdot \vec{j} + (z_c) \cdot \vec{k} \\ \vec{r}_b &= (x_b) \cdot \vec{i} + (y_b) \cdot \vec{j} + (z_b) \cdot \vec{k} \end{aligned} \tag{2.5}$$

dove $\vec{i}$, $\vec{j}$, $\vec{k}$ sono i versori degli assi x, y, z.
Indichiamo con m_m, m_c, m_b rispettivamente le masse del carico, del carrello e del binario.
Indichiamo con l la lunghezza del filo.
Indichiamo con $\vec{F}_1$, $\vec{F}_2$ e $\vec{F}_3$ le forze fornite dai motori rispettivamente al carrello lungo $\vec{i}$, al binario lungo $\vec{j}$ e al carico in direzione parallela al cavo. I moduli di tali sono rispettivamente F_1, F_2, F_3.
Indichiamo con k_1, k_2, k_3 i coefficienti di attrito viscoso relativi al carrello, al binario e al carico.
Gli angoli α e β sono definiti come in figura 1.2.

2.1.4 Vincoli e gradi di libertà

Le equazioni (2.5) sono soggette ai seguenti vincoli:

$$\begin{aligned} x_b &= 0 \\ z_b &= 0 \\ z_c &= 0 \\ y_b &= y_c \end{aligned} \tag{2.6}$$

Il sistema ha complessivamente cinque gradi di libertà. La posizione dei tre corpi è dunque determinata da cinque diverse coordinate generalizzate che esprimiamo in un unico vettore $\vec{q}$:

$$\vec{q} = (x_g, y_g, l, \alpha, \beta) \tag{2.7}$$

Il legame tra coordinate cartesiane e generalizzate, derivante da semplici considerazioni trigonometriche, è dato dalle seguenti equazioni:

$$\begin{aligned} \vec{r}_m &= (x_g + l\cos\alpha) \cdot \vec{i} + (y_g + l\sin\beta\sin\alpha) \cdot \vec{j} - (l\cos\beta\sin\alpha) \cdot \vec{k} \\ \vec{r}_c &= (x_g) \cdot \vec{i} + (y_g) \cdot \vec{j} + (0) \cdot \vec{k} \\ \vec{r}_b &= (0) \cdot \vec{i} + (y_g) \cdot \vec{j} + (0) \cdot \vec{k} \end{aligned} \tag{2.8}$$

2.1.5 Velocità, energia cinetica, energia potenziale

Calcoliamo quindi velocità, energia cinetica e energia potenziale del sistema. Le componenti delle velocità sono:

$$\begin{aligned}
\dot{x}_m &= \dot{x}_g + \dot{l}\cos\alpha - l\dot{\alpha}\sin\alpha \\
\dot{y}_m &= \dot{y}_g + \dot{l}\sin\alpha\sin\beta + l\dot{\alpha}\cos\alpha\sin\beta + l\sin\alpha\dot{\beta}\cos\beta \\
\dot{z}_m &= -\dot{l}\sin\alpha\cos\beta - l\dot{\alpha}\cos\alpha\cos\beta + l\sin\alpha\dot{\beta}\sin\beta \\
\dot{x}_c &= \dot{x}_g \\
\dot{y}_c &= \dot{y}_g \\
\dot{z}_c &= 0 \\
\dot{x}_b &= 0 \\
\dot{y}_b &= \dot{y}_g \\
\dot{z}_b &= 0
\end{aligned} \tag{2.9}$$

L'energia cinetica è:

$$\begin{aligned}
T = {} & \frac{1}{2}m_m((\dot{x}_g + \dot{l}\cos\alpha - l\dot{\alpha}\sin\alpha)^2 + (\dot{y}_g + \dot{l}\sin\alpha\sin\beta + \\
& + l\dot{\alpha}\cos\alpha\sin\beta + l\sin\alpha\dot{\beta}\cos\beta)^2 + (-\dot{l}\sin\alpha\cos\beta + \\
& - l\dot{\alpha}\cos\alpha\cos\beta + l\sin\alpha\dot{\beta}\sin\beta)^2) + \frac{1}{2}m_c(\dot{x}_g^2 + \dot{y}_g^2) + \\
& + \frac{1}{2}m_b(\dot{y}_g^2)
\end{aligned} \tag{2.10}$$

L'energia potenziale:

$$V = -m_m g l\sin\alpha\cos\beta \tag{2.11}$$

2.1.6 Un modello unidimensionale

E' qui presentato un modello semplificato del carroponte, in cui il moto avviene in un'unica direzione. Ipotizziamo dunque che il carrello si muova lungo la coordinata x, mentre il binario sia bloccato.
Ipotizziamo inoltre che la lunghezza del filo sia fissa.
Ipotizzando infine inizialmente nullo l'angolo β, esso rimarrà nullo durante tutta la durata del moto. La figura 2.2 mostra questa situazione.
Saranno quindi presenti i seguenti vincoli, oltre a quelli delle equazioni (2.6)

$$\begin{aligned}
y_c &= Y_C \\
l &= L \\
\beta &= 0
\end{aligned}$$

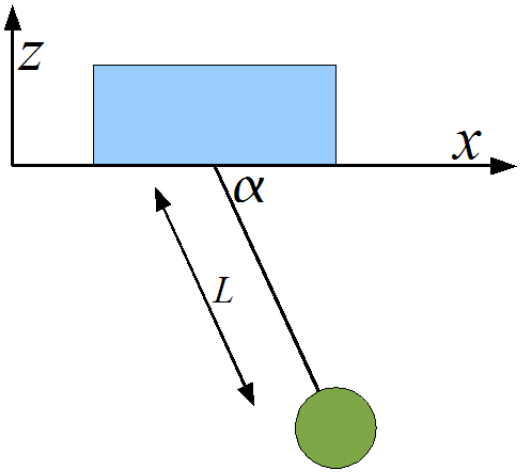


Figura 2.2: Moto del carrello lungo x

con Y_C e L costanti.
Le coordinate generalizzate saranno dunque solo due:

$$\vec{q} = (x_c, \alpha)$$

Le componenti delle velocità si semplificano a:

$$\begin{aligned}
\dot{x}_m &= \dot{x}_g - L\dot{\alpha}\sin\alpha \\
\dot{y}_m &= 0 \\
\dot{z}_m &= -L\dot{\alpha}\cos\alpha \\
\dot{x}_c &= \dot{x}_g \\
\dot{y}_c &= 0 \\
\dot{z}_c &= 0 \\
\dot{x}_b &= 0 \\
\dot{y}_b &= 0 \\
\dot{z}_b &= 0
\end{aligned}$$

L'energia cinetica diviene:

$$T = \frac{1}{2}m_m((\dot{x}_g - L\sin\alpha\dot{\alpha})^2 + L^2\cos\alpha^2\dot{\alpha}^2) + \frac{1}{2}m_c\dot{x}_g^2$$

Le forze attive non conservative totali sono:

$$\begin{aligned}
\vec{F}_c &= \vec{F}_1 - k_1\vec{v}_c \qquad \text{sul carrello} \\
\vec{F}_m &= -k_3\vec{v}_m \qquad \text{sul carico}
\end{aligned}$$

Ricaviamo ora le componenti lagrangiane delle forze non derivabili da potenziale lungo le due coordinate generalizzate, secondo la formula (2.4):

$$\begin{aligned} Q_{x_g} &= \vec{F_c} \cdot \frac{\partial \vec{r}_x}{\partial x_g} + F_m \cdot \frac{\partial \vec{r}_m}{\partial x_g} \\ &= F_1 - k_1 \dot{x}_g - k_3 \dot{x}_g + k_3 l \sin\alpha\dot{\alpha} \\ Q_\alpha &= \vec{F_c} \cdot \frac{\partial \vec{r}_x}{\partial \alpha} + F_m \cdot \frac{\partial \vec{r}_m}{\partial \alpha} \\ &= k_3 L(\sin\alpha \dot{x}_g - L\dot{\alpha}) \end{aligned}$$

Useremo ora le equazioni di Lagrange per determinare le leggi di moto del sistema. Esse sono dunque nel nostro caso:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_{x_g}} - \frac{\partial L}{\partial q_{x_g}} = Q_{x_g} \tag{2.12}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_{\alpha}} - \frac{\partial L}{\partial q_{\alpha}} = Q_{\alpha} \tag{2.13}$$

Sostituendo i valori di L, Q_α, Qx_g precedentemente determinati e risolvendo le equazioni in funzione di $\ddot{x}_g$ e $\ddot{\alpha}$, si trova finalmente:

$$\ddot{x}_g = \frac{F_1 - k_1\dot{x}_g + \sin\alpha m_m g\cos\alpha - k_3\dot{x}_g\cos\alpha^2 + m_m L\cos\alpha\dot{\alpha}^2}{m_m\cos\alpha^2 + m_c} \tag{2.14}$$

e

$$\begin{aligned} \ddot{\alpha} = {} & (m_c k_3 \sin\alpha\dot{x}_g - m_m \sin\alpha k_1 \dot{x}_g + m_m^2 \sin\alpha L\cos\alpha\dot{\alpha}^2 + \\ & + m_m \sin\alpha F_1 + m_m^2 g\cos\alpha + m_c m_m g\cos\alpha - m_c k_3 L\dot{\alpha} + \\ & - m_m k_3 L\dot{\alpha}\cos\alpha^2)/[(m_m\cos\alpha^2 + m_c)m_m L] \end{aligned} \tag{2.15}$$

In figura 2.3 è mostrato l'andamento di α e di $\dot{x}_g$ (la velocità del carrello) nel tempo a fronte di un ingresso F_1 a scalino. Il sistema è inizialmente in equilibrio con $x_g = 0$, $\dot{x}_g = 0$, $\alpha = \pi/2$, $\dot{\alpha} = 0$. I parametri fisici utilizzati non sono quelli reali del sistema ma sono abbastanza plausibili. Come prevedibile dall'esperienza fisica, entrambe le grandezze raggiungono asintoticamente un nuovo valore di equilibrio. Si noti come la dinamica del carrello risenta della presenza del carico. Il grafico della velocità del carrello comprende infatti una componente oscillatoria dovuta proprio alla dinamica del carico.

2.1.7 Il modello generale

E' stato risolto analiticamente anche il caso più generale, con movimentazione lungo i tre assi e quindi a cinque gradi di libertà. Le equazioni e i passaggi

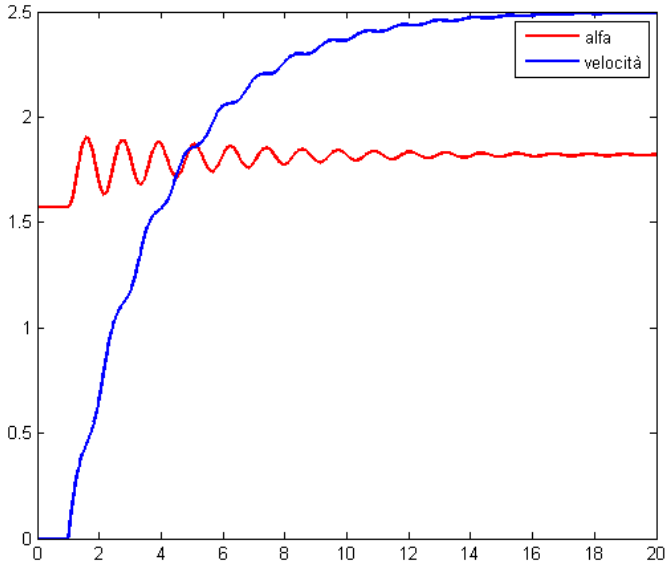


Figura 2.3: Risposta allo scalino del modello unidimensionale

risolutivi sono illustrati nell'appendice A. Come mostreremo, è comodo avere questo modello perchè da esso possono essere ricavati modelli di qualsiasi complessità intermedia, utili nel progetto del controllore.

2.1.8 Considerazioni riguardo l'attrito

Nei modelli precedenti discussi sono state considerate esclusivamente le forze di attrito viscoso. Non si è tenuto conto dell'attrito statico (che agisce a velocità nulla opponendosi ad moto, fino a un valore massimo) e dell'attrito dinamico (che agisce a velocità non nulla, in verso opposto alla velocità e modulo costante). Questa ipotesi è in realtà poco realistica: si è proceduto in questo modo solo per la difficoltà di inserire queste forze nel modello. In fase di controllo sarà peraltro necessario tenere conto di queste forze e compensarle opportunamente.

2.2 I motori

Sono stati impiegati nel progetto tre motori in corrente continua. Più precisamente si tratta di motori a spazzole con magneti permanenti, la tipologia più diffusa per le piccole e medie potenze. Descriveremo un modello sufficien-

temente accurato per i motori che sarà poi integrato al quello meccanico già analizzato. La trattazione sarà particolarmente veloce in quanto non siamo interessati particolarmente alla fisica del motore, ma alle equazioni che ne descrivono, dall'esterno, in funzionamento.

2.2.1 Le equazioni del motore in corrente continua

Un motore in corrente continua è costituito da una carcassa all'interno della quale è presente un campo magnetico generato da magneti permanenti. Il rotore è costituito da una serie di spire solidali con l'albero rotante; il collegamento elettrico con l'alimentazione è costituito da due spazzole striscianti che costituiscono la parte più delicata del motore. Si tratta di cilindri in carbone o contatti in metallo che strisciano sul alcuni contatti elettrici ricavati nell'albero (il collettore). Il circuito equivalente di un motore CC è costituito da una resistenza R_a, da un induttanza L_a e da un generatore di tensione E_g, proporzionale alla velocità di rotazione (fig.2.4). La coppia fornita all'albero, C_m, risulta proporzionale alla corrente che attraversa il circuito, I_a. Le equazioni che regolano il sistema sono le seguenti:

$$V_a = R_a I_a + L_a \dot{I}_a + E_g \tag{2.16}$$

$$E_g = K_t \omega \tag{2.17}$$

$$C_m = K_t I_a \tag{2.18}$$

Dove ω è la velocità angolare di rotazione dell'albero e K_t è la costante di coppia. E' possibile trascurare l'induttanza L_a, di poca rilevanza dal punto di vista delle equazioni del sistema. Fare questa approssimazione significa trascurare nel modello del motore un polo ad alta frequenza.

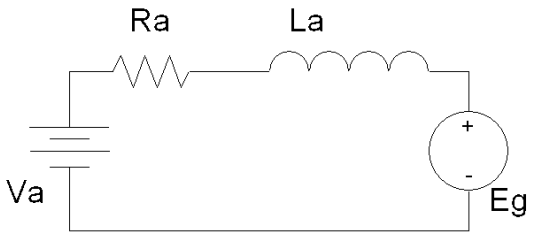


Figura 2.4: Modello elettrico del motore in corrente continua

Essendo la caduta di tensione sulla resistenza pari a $V_a - K_t\omega$ e quindi $(V_a - K_t\omega)/R_a$ la corrente I_a, sostituendo nella (2.18), si ha:

$$C_m = K_t \frac{V_a - K_t\omega}{R_a} \tag{2.19}$$

La potenza meccanica generata, P_m, è pari a $C_m\omega$, e quindi sostituendo:

$$P_m = K_t\omega\frac{V_a - K_t\omega}{R_a} \tag{2.20}$$

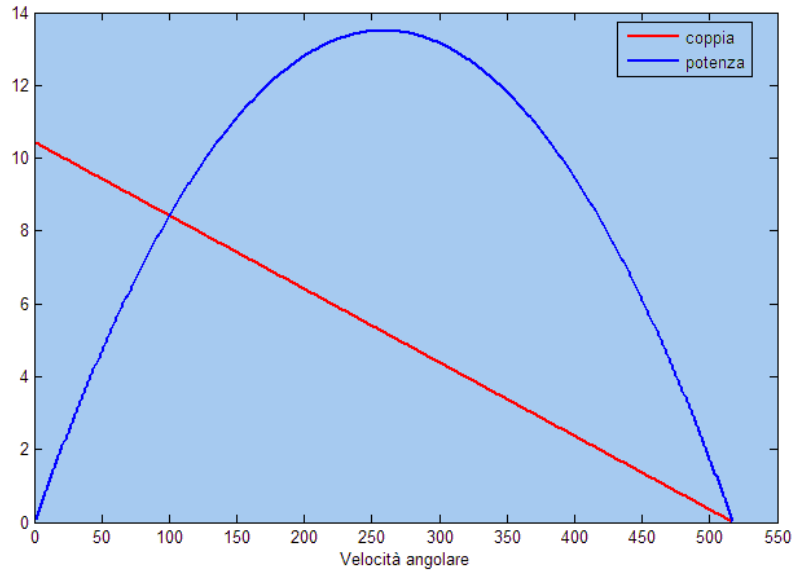


Figura 2.5: Caratteristica del motore in corrente continua

Dall'esame delle equazioni (2.19) e (2.20) si possono fare alcune osservazioni.
La coppia risulta massima a velocità nulla (coppia di stallo) e decresce linearmente con essa, fino alla velocità massima teorica (velocità a vuoto), alla quale è la coppia ad essere nulla.
La disponibilità dei massimi valori di coppia a a bassa velocità angolare rende i motori CC particolarmente adatti a vincere attriti statici di avviamento, problema rilevante in molti casi compreso il nostro.
La potenza meccanica massima si ha al 50% della velocità a vuoto. Nella pratica si fa in modo di operare, a regime, tra il 70% e il 90% della velocità a vuoto, per evitare surriscaldamento. In questa regione è peraltro massima l'efficienza del motore.

E' inoltre opportuno sottolineare che molti motori CC, compresi quelli usati in questa tesi, sono dotati di meccanismi di riduzione: questo perchè in genere essi danno velocità di rotazione troppo elevate e coppie troppo basse rispetto alle esigenze dei carichi. Utilizzando un meccanismo di riduzione

la coppia all'albero risulta moltiplicata per una costante $\eta > 1$ rispetto alla coppia motore mentre la velocità angolare risulta moltiplicata per $1/\eta$.

2.2.2 Schema a blocchi

Il motore in corrente continua fin qui discusso può essere rappresentato con il formalismo degli schemi a blocchi, come in figura 2.6. E' stata aggiunta

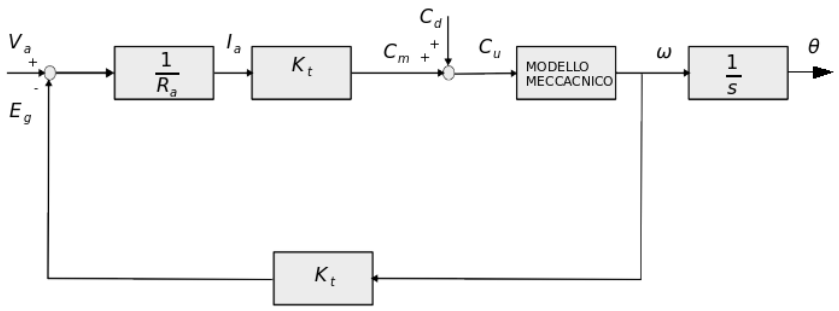


Figura 2.6: Schema a blocchi di un motore in corrente continua

una coppia di disturbo C_d, tramite la quale possiamo modellizzare ad esempio l'attrito dinamico o altri fenomeni esterni. Si noti come il comportamento del motore dipende dal modello meccanico del sistema applicato all'albero. Se ad esempio l'albero risultasse bloccato, si avrebbe corrente I_a a valor costante V_a/Ra, coppia $C_m = K_t V_a/C_m$ e velocità angolare ovviamente nulla.

Un caso di particolare importante è quello di un carico caratterizzato da un momento di inerzia J_t e un attrito viscoso B_t. Questa situazione ha luogo anche se non vi è alcun carico accoppiato all'albero: infatti il rotore stesso ha un certo momento d'inerzia rotore J_m, mentre l'attrito viscoso B_m è dato dalle spazzole. Se invece c'è anche un carico esterno, di momento J_e e attrito B_e, si può dimostrare che vale un modello equivalente con:

$$J_t = J_m + \frac{1}{\eta^2} J_e \tag{2.21}$$

$$B_t = B_m + \frac{1}{\eta^2} B_e \tag{2.22}$$

dove η è il fattore di riduzione del motore. L'equazione che lega coppia e velocità angolare è dunque:

$$C_u = J_t\dot{\omega} + B_t\omega$$

Passando alle trasformate di Laplace:

$$C_u(s) = J_t s\Omega + B_t\Omega$$

La funzione di trasferimento del blocco del modello meccanico in figura 2.6 risulta quindi:

$$\frac{\Omega(s)}{C_u(s)} = \frac{1}{J_t s + B_t} \tag{2.23}$$

Tramite le regole di elaborazione dei blocchi si può ricavare la funzione di trasferimento tra tensione di alimentazione e velocità angolare:

$$G(s) = \frac{\Omega(s)}{V_a(s)} = \frac{K_t/R_a}{J_t s + B_t + K_t^2/R_a} = \frac{K_M}{1 + sT_M} \tag{2.24}$$

Così come la funzione di trasferimento tra coppia di disturbo e velocità angolare:

$$H(s) = \frac{\Omega(s)}{C_d(s)} = \frac{1}{J_t s + B_t + K_t^2/R_a} = \frac{K_C}{1 + sT_M} \tag{2.25}$$

Uno schema a blocchi semplificato per questa situazione è mostrato in figura 2.7. Si noti che se l'ingresso C_d è un'ingresso a scalino (come può essere

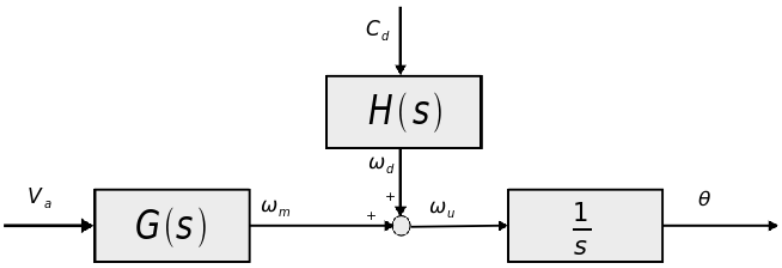


Figura 2.7: Schema a blocchi semplificato di un motore in corrente continua

l'attrito dinamico muovendosi in una direzione), anche ω_d è asintoticamente a scalino e dunque l'effetto su θ è asintoticamente a rampa. Questa considerazione tornerà utile in ambito di controllo per effettuare un'efficace reiezione dei disturbi.

3

Progettazione e Realizzazione

E' qui descritta la realizzazione del carroponte nel laboratorio. Non si è voluto scrivere un capitolo separato per progettazione e realizzazione in quanto le due fasi si sono svolte di pari passo, in tempi non ben distinti. Il capitolo è diviso in due sezioni: la prima riguardante la parte meccanica, la seconda riguardante la parte elettrica.

3.1 Realizzazione meccanica

La struttura portante del carroponte è costituita da un parallelepipedo di materiale metallico. La faccia rettangolare superiore, che definisce il piano xy in cui si mouve il carrello, ha dimensioni $100 \times 120cm$ (fig. 3.1).

Sui due lati a e b, paralleli all'asse y, sono fissati due binari, B_1 e B_2, su cui scorrono due carrelini, C_1 e C_2. Ai due carrelli è fissato un binario B_3, parallelo all'asse x su cui scorre un terzo carrellino: C_3 (fig. 3.2).

Movimento lungo z Il carrellino C_3 trasporta una piattaforma, sulla quale si trovano il motore M_3 relativo all'asse z e la telecamera. Il motore M_3 è accoppiato con un ingranaggio a una piccola ruota metallica, che tramite un filo di nylon può sollevare il carico, una pallina da golf. All'albero di M_3, così come agli altri motori, è accoppiato anche un encoder, il sensore di posizione angolare.

Movimento lungo y Il motore M_1 si trova sul lato c ed è permette il movimento lungo l'asse y. Tramite un ingranaggio il movimento dell'albero motore di M_1 è trasmesso all'albero A_1, parallelo a y e sollevato di qualche centimetro rispetto a c, sostenuto da alcuni cuscinetti. Sorpra lato d si trova

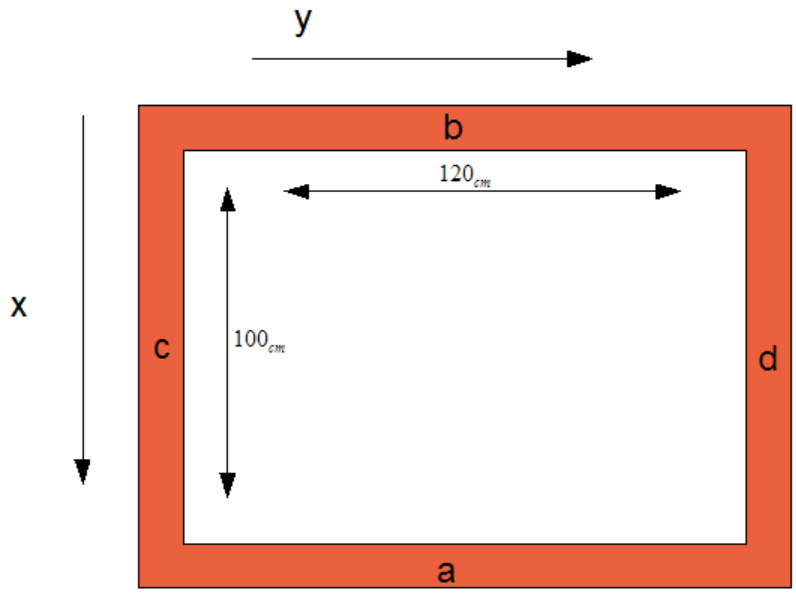


Figura 3.1: Dimensioni del carroponte

l'albero A_2, anch'esso sostenuto da un sistema di cuscinetti alla stessa altezza di A_1. Alle estremità degli alberi vi sono delle pulegge: P_1 e P_2 su A_1, P_3 e $P4$ su A_2. Tra $P1$ e $P3$, così come tra $P2$ e $P4$, sono montate delle cinghie, a cui sono collegati i carrellini C_1 e C_2. Le cinghie trascinano nel loro moto i carrellini, che quindi si muovono alla stessa velocità lungo l'asse y, trasportando nel loro moto il binario B_3.

Movimento lungo x Sul carrellino C_1 è montato il motore M_1, sul cui albero è montata la puleggia P_5. Sul carrellino C_2 è invece montata solo una puleggia, P_6, sostenuta da un'albero. Tra P_5 e P_6 è si trova una cinghia, movimentata dalle pulegge, a cui è attaccato il carrellino C_3, che può dunque muoversi lungo x

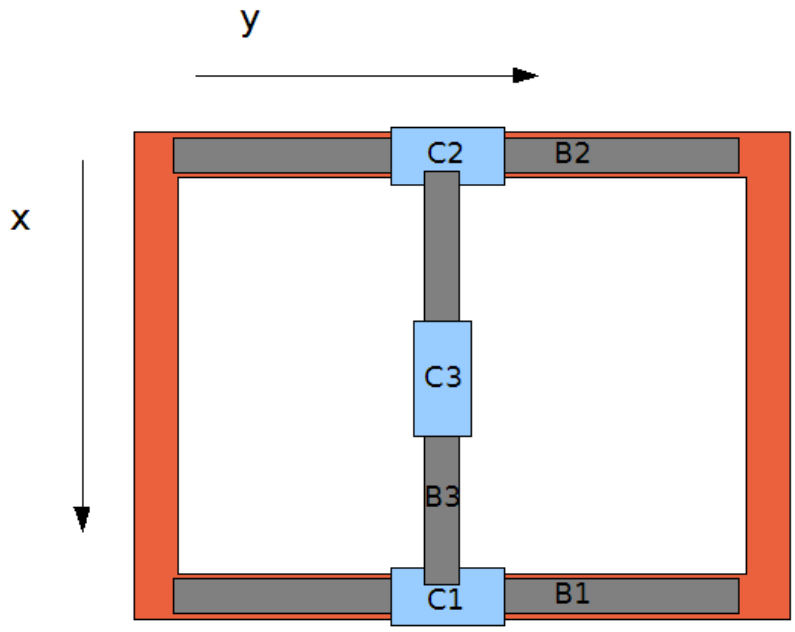


Figura 3.2: La struttura del carroponte

3.2 Realizzazione elettrica

In questa sezione verranno presentati i componenti elettrici ed elettronici che sono stati utilizzati nel progetto e i collegamenti elettrici tra essi.

I vari componenti sono disposti sul PC, sul carroponte e sul quadro elettrico. Sul PC è presente la scheda di RT-DAC4/PCI, che gestisce l'input/output verso i vari dispositivi. Sul carroponte sono collocati gli encoder per i tre assi, i finecorsa e i tre motori elettrici. Sul quadro elettrico sono presenti vari componenti tra cui:

- Un alimentatore da $24V$ destinato al motore M_1
- Un alimentatore da $24V$ destinato ai motori M_2, M_3 e ai componenti ausiliari
- Un alimentatore da $5V$ destinato a vari usi
- Un motor driver per il motore M_1
- Un motor driver per i motori M_2 e M_3
- Un joystick da cui è generato il comando manuale

- Una rete di condizionamento per il segnale manuale proveniente dal joystick.
- Un relè che gestisce il passaggio automatico/manuale, con opportuna elettronica di pilotaggio
- Interruttore generale, contattore di marcia, pulsante di marcia, pulsante automatico/manuale, pulsante di emergenza, lampada di marcia, lampada automatico/manuale.
- Vari fusibili di protezione.

L'alimentazione esterna arriva il ingresso al quadro ed è la comune corrente alternata a $220V$.

Descrizione dei componenti

I motori Il modello dei motori è già stato discusso. In tabella 3.1 sono riportati i dati di targa di M_1, M_2 e M_3.

Motore	M_1	M_2	M_3
Tensione nominale(V)	24	24	24
Corrente nominale(A)	1.9	0.22	0.04
Potenza meccanica nominale(W)	27	3.8	0.3
Velocità nominale(rpm)	1800	8400	4680
Rapporto di riduzione	12.25:1	12.25:1	18:1
Resistenza(Ω)	3.9	-	-
Induttanza(μH)	6.4	-	-

Tabella 3.1: Dati di targa dei motori

I motor driver sono le interfacce di che permettono di comandare in tensione i motori DC. Ad essi viene fornita in ingresso un'alimentazione per la logica ($5V$), un'alimentazione per la potenza (nel nostro caso $24V$) e un segnale di comando in bassa tensione (compreso tra 0 e $5V$), che può essere di vari tipi, a seconda della configurazione di alcuni jumper. In uscita è fornito un segnale di potenza, applicabile direttamente ai motori, che dipende dal segnale di comando. Questo segnale in uscita è in realtà un'onda quadra ad alta frequenza tra 0 e $\pm 24V$ in cui il comando agisce sul segno dell'uscita e la percentuale del tempo di ciclo in cui l'uscita è diversa da 0 (il duty cycle).

Questo tipo di pilotaggio è detto PWM (Pulse Width Modulation). L'uscita è equivalente, per la scomposizione in armoniche di un'onda quadra, a una costante sovrapposta ad alcune sinusoidi ad alta frequenza. Avendo in generale il motore un comportamento da filtro passa-basso, l'effetto rilevante sul motore è solo quello della componente continua. Si preferisce questo tipo di pilotaggio, piuttosto che un pilotaggio di tipo proporzionale, per motivi di efficienza elettrica. Si è utilizzato per il segnale di comando la modalità analogica $0V$-$2.5V$-$5V$, in cui:

- Il segnale di comando è una tensione analogica tra $0V$ e $5V$
- $0V$ corrisponde alla massima tensione negativa (duty cycle 100% a $-24V$)
- $2.5V$ corrisponde a un'uscita nulla (duty cycle 0%)
- $5V$ corrisponde alla massima tensione positiva (duty cycle 100% a $+24V$)

C'è inoltre una piccola banda morta attorno ai $2.5V$, per garantire una stabile posizione di off. I motor driver utilizzati sono stati selezionati in base alle correnti massime in gioco. E' stato scelto il modello MD03 (fig. 3.3) per M_1 e il modello MD22 per M_2 ed M_3.

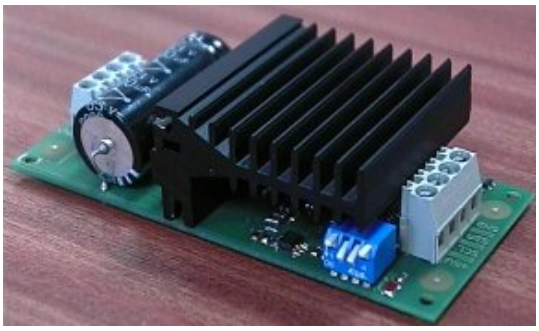


Figura 3.3: Il motor driver MD03

Il joystick è il dispositivo che fornisce il segnale di controllo manuale. Si tratta si un modello a tre gradi di libertà di movimento, ciascuno dei quali genera il seganle per un motore. Per ogni grado di libertà il joystick è sostanzialmente un partitore resistivo del valore complessivo di $5K\Omega$, centrato attorno ai $2.5K\Omega$. La partizione non avviene però lungo tutto il range, ma è limitata all'intervallo $[2K, 3K]$. La tensione in uscita al joystick varia

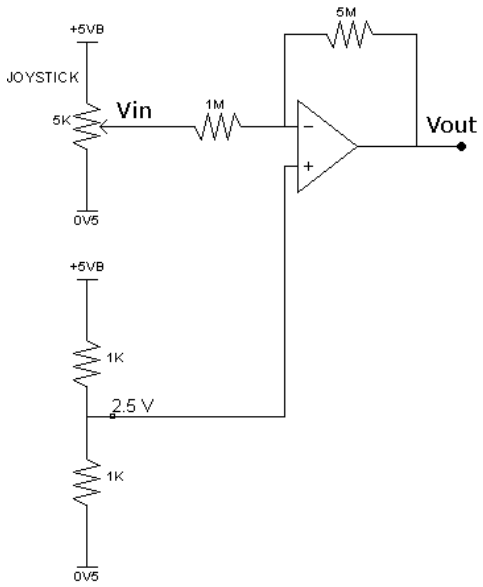


Figura 3.4: Rete di condizionamento per i segnali dal joystick

dunque nell'intervallo $[2V, 3V]$. E' stata quindi progettatta e costruita una rete di condizionamento basata su amplificatori operazionali per i segnali provenienti dal joystick che dilata il segnale all'intero range $[0V, 5V]$ utilizzato dai motor driver. La figura 3.4 mostra lo schema elettrico della rete di condizionamento per uno dei tre assi. Si noti che il circuito realizza la funzione

$$V_{OUT} = 15V - 5V_{IN} \tag{3.1}$$

e l'uscita è invertita rispetto all'ingresso, ovvero $2V$ dal joystick provocano un'uscita di $5V$, mentre $3V$ dal joystick mandano l'uscita a 0.

Gli encoder sono trasduttori di posizione economici, precisi, robusti ed affidabili. E' mostrato in figura 3.5 uno schema che ne evidenzia il funzionamento. L'albero trasmette il moto a un disco coronato da settori circolari opachi e trasparenti. A un lato del disco è situata una sorgente luminosa e a quello opposto i componenti fotosensibili. Con la rotazione del disco vi sono delle ripetute interruzioni del fascio di luce che li percorre. Questa variazione di luminosità genera in uscita un segnale digitale a seconda della posizione dell'albero. Gli encoder utilizzati sono di tipo incrementale. Il sistema di in-

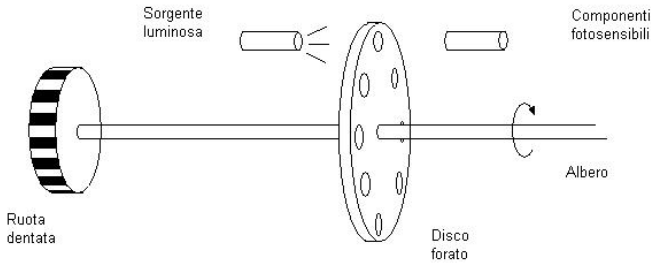


Figura 3.5: Funzionamento di un encoder

tercettazione è costituito da un disco sul quale sono state ricavate due corone concentriche di finestrelle rettangolari.

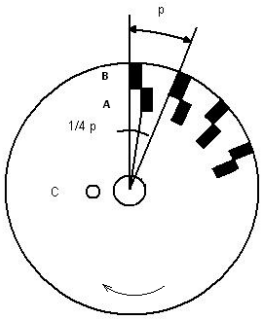


Figura 3.6: Sistema di intercettazione di un encoder incrementale

Tali finestrelle, poste una sopra l'altra, sono sfasate fra di loro di un quarto di passo. Ai lati del sistema di intercettazione sono presenti, da una parte due fotoemettitori e dall'altra due fotorivelatori (un fotoemettitore e un fotorivelatore per ciascuna corona concentrica di finestrelle). Ai fotorilevatori arriva un treno di impulsi il cui numero è pari al numero delle zone trasparenti, alternate alle scure, intercettate dal blocco emettitore-ricevitore. Sono comunemente chiamati canale A e canale B i segnali rilevato dai fotorilevatori.

Il conteggio degli impulsi di uno dei due canali consente di individuare la rotazione compiuta dal disco e quindi, nel nostro caso, lo spostamento lineare in x e y e la lunghezza del filo. La presenza di due canali, come mostrano le figure 3.7 e 3.8, permette inoltre di capire il senso di rotazione del disco perchè cambia l'ordine temporale in cui arrivano i fronti di salita (o di discesa) su A e su B.

Sono stati utilizzati tre identici encoder incrementali di marca Baumer e famiglia BHK, modello $06.05A500 - B6 - 5$ (fig. 3.9). La risoluzione del dispositivo è di 2048 tacche per giro. I cavi in ingresso e uscita dagli encoder sono definiti come in tabella 3.2.

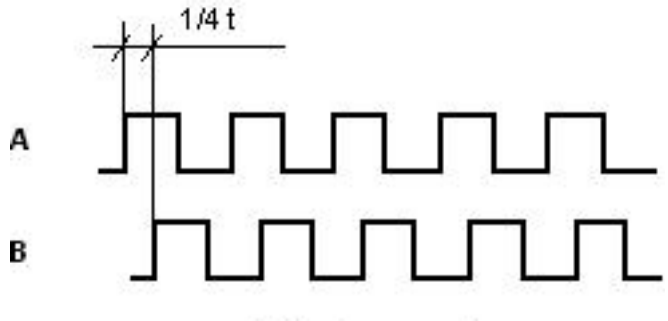


Figura 3.7: Rotazione in senso orario

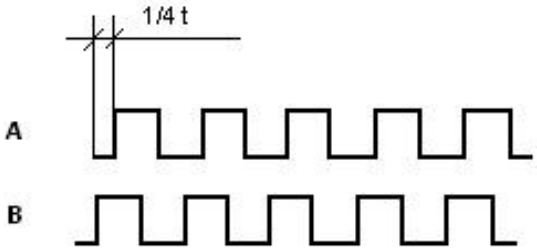


Figura 3.8: Rotazione in senso antiorario

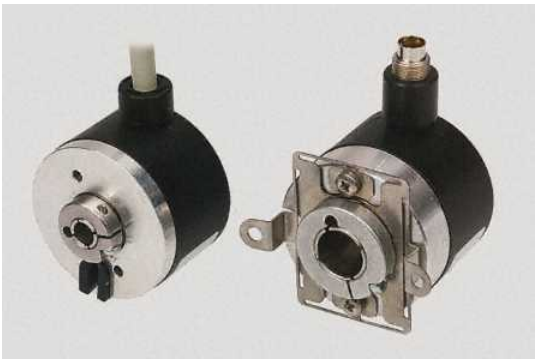


Figura 3.9: L'encoder $BHK06.05A500 - B6 - 5$

Segnale	Colore cavo
$+5V$	marrone
$0V$	bianco
CH A	verde
CH A compl.	rosso
CH B	giallo
CH B compl.	blu
CH N	rosa
CH N compl.	grigio
schermo	a massa

Tabella 3.2: Cavi in ingresso e uscita all'encoder

I finecorsa sono dei semplici interruttori meccanici posti appena prima della fine dei binari. Hanno duplice funzioni: servono per la ricerca del riferimento iniziale e, durante il movimento, garantiscono un blocco di sicurazza al relativo motore. Quando il carrello (o il binario) vengono a toccare il finecorsa, l'interruttore si chiude e il relativo segnale diventa alto ($5V$), mentre normalmente il segnale è basso. Sono stati impiegati 4 finecorsa $D4D - 1121N$ della OMRON ELECTRONICS, denominati nel progetto FCM1A, FCM1I, FCM2A, FCM2I. Essi permettono dunque di rilevare la fine del tratto percorribile x e y in entrambe i sensi di marcia (fig. 3.10).

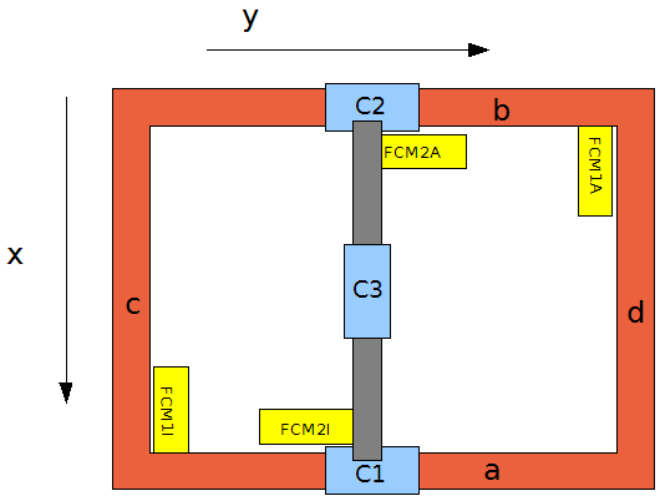


Figura 3.10: Posizione dei finecorsa sul carroponte

La scheda di I/O gestisce l'input/output dei segnali da PC. Il modello utilizzato è la RT-DAC4/PCI di marca Inteco (fig.3.11). Nella configurazione standard da noi utilizzata, essa mette a disposizione:

- 16 ingressi analogici
- 4 uscite analogiche
- 32 ingressi/uscite digitali
- 4 uscite PWM
- 4 ingressi per encoder incrementali (CHA e CHB per ogni ingresso) in configurazione standard

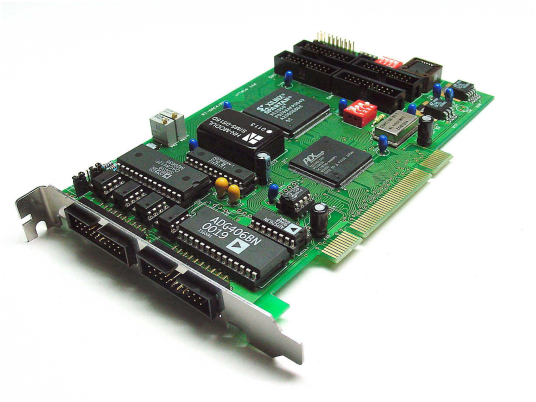


Figura 3.11: La scheda di I/O RT-DAC4/PCI

- 1 contatore a 32 bit, $40Mhz$

La scheda, montata sull'interfaccia PCI del PC, ha una mortsettiera estrerna per il facile accesso ai segnali. La tabella 3.6 mostra i segnali in morsettiera con relativa descrizione. L'elettronica a bordo della scheda di acquisizione gestisce in hardware il segnale proveniente dai canali A e B degli encoder fornendo in uscita un intero con segno corrispondente allo spostamento angolare in tacche di lettura. Il canale N, un contatore di giri interi, non è invece stato utilizzato.

Collegamenti tra i dispositivi

I dispositivi su carro, quadro e PC sono collegati tra loro da cavi elettrici, come mostrato schematicamente in figura 3.12.

Per motivi di compatibilità elettromagnetica i comandi ai motori, in alta frequenza e alta corrente, sono stati fatti passare in cavi separati rispetto ai altri segnali. Nelle tabelle sono riportati i segnali tra quadro e carro, tra carro e scheda e tra scheda e quadro, con il relativo significato e direzione verso la quale il segnale viaggia, dal punto di vista logico. Il nome del segnale è lo stesso effettivamente usato per numerare i fili. In tabella 3.6 sono invece riportati i segnali disponibili alla morsettiere nella scheda, con il relativo nome del pin e il nome con cui il segnale è visto dalla scheda e dal software relativo. E' infine mostrato lo schema elettrico completo del sistema nelle figure 3.13 e 3.14.

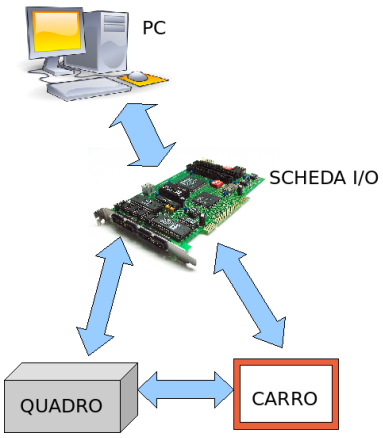


Figura 3.12: Comunicazioni elettriche tra i dispositivi

Segnale	Descrizione	direzione
B1+	Comando + motore M_1	da quadro a carro
B1-	Comando - motore M_1	da quadro a carro
B2+	Comando + motore M_2	da quadro a carro
B2-	Comando - motore M_2	da quadro a carro
B3+	Comando + motore M_3	da quadro a carro
B3-	Comando - motore M_3	da quadro a carro
+5VB	Alimentazione a $5V$	da quadro a carro
0V5	Massa dell'alimentazione a $5V$	da quadro a carro

Tabella 3.3: Segnali tra quadro e carro

Segnale	Descrizione	direzione
81	CHA asse y	da carro a scheda
82	CHB asse y	da carro a scheda
83	CHA asse y	da carro a scheda
84	CHB asse y	da carro a scheda
85	CHA asse y	da carro a scheda
86	CHB asse y	da carro a scheda
FCM1A	Fine corsa asse y avanti	da carro a scheda
FCM1I	Fine corsa asse y indietro	da carro a scheda
FCM2A	Fine corsa asse x avanti	da carro a scheda
FCM2I	Fine corsa asse x indietro	da carro a scheda

Tabella 3.4: Segnali tra carro e scheda

Segnale	Descrizione	direzione
50	CMD all'azionamento del motore M_1	da scheda a quadro
51	CMD all'azionamento del motore M_2	da scheda a quadro
52	CMD all'azionamento del motore M_1	da scheda a quadro
R1	CMD commutazione automatico/manuale	da scheda a quadro
SAM	SEL automatico/manuale	da quadro a scheda
KM	Contattore di marcia	da quadro a scheda
0V5	Massa di riferimento per tutti i segnali	da quadro a scheda

Tabella 3.5: Segnali tra scheda e quadro

Nome Segnale	Descrizione	Pin	Nome software
81	CHA asse y	A1	CHA ENC00
82	CHB asse y	A3	CHB ENC00
83	CHA asse x	A1	CHA ENC01
84	CHB asse x	A3	CHB ENC01
85	CHA asse z	A1	CHA ENC02
86	CHB asse z	A3	CHB ENC02
FCM2A	Fine corsa asse x avanti	B1	ANALOG IN 10
FCM2I	Fine corsa asse x indietro	B3	ANALOG IN 11
FCM1A	Fine corsa asse y avanti	B5	ANALOG IN 12
FCM1I	Fine corsa asse y indietro	B7	ANALOG IN 13
SAM	SEL automatico/manuale	B9	ANALOG IN 14
KM	Contattore di marcia	B11	ANALOG IN 15
50	CMD all'azionamento di M_1	B13	ANALOG OUT 0
51	CMD all'azionamento di M_2	B15	ANALOG OUT 1
52	CMD all'azionamento di M_3	B17	ANALOG OUT 2
R1	CMD automatico/manuale	B19	ANALOG OUT 3

Tabella 3.6: Segnali alla morsettiera della scheda

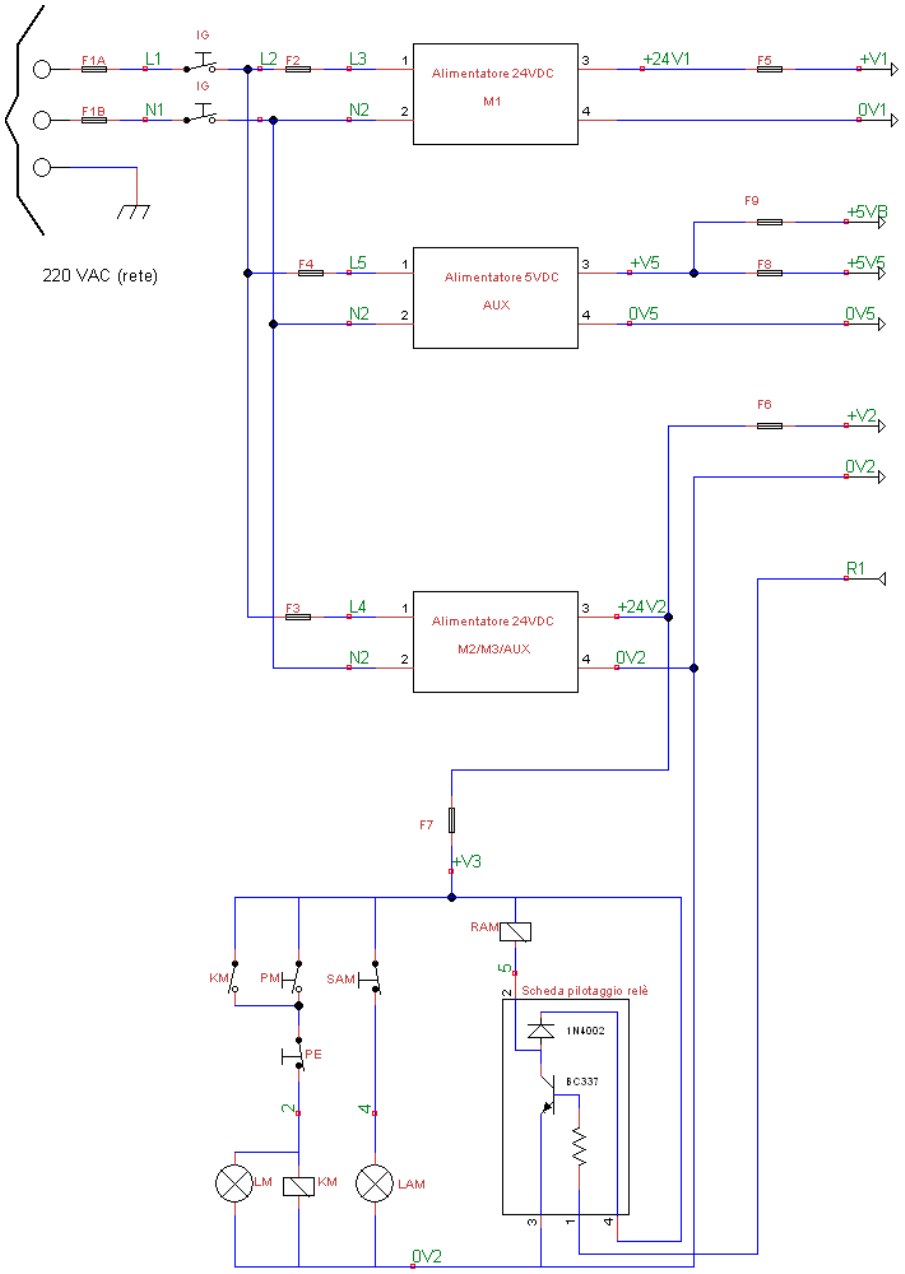


Figura 3.13: Schema elettrico completo - parte 1

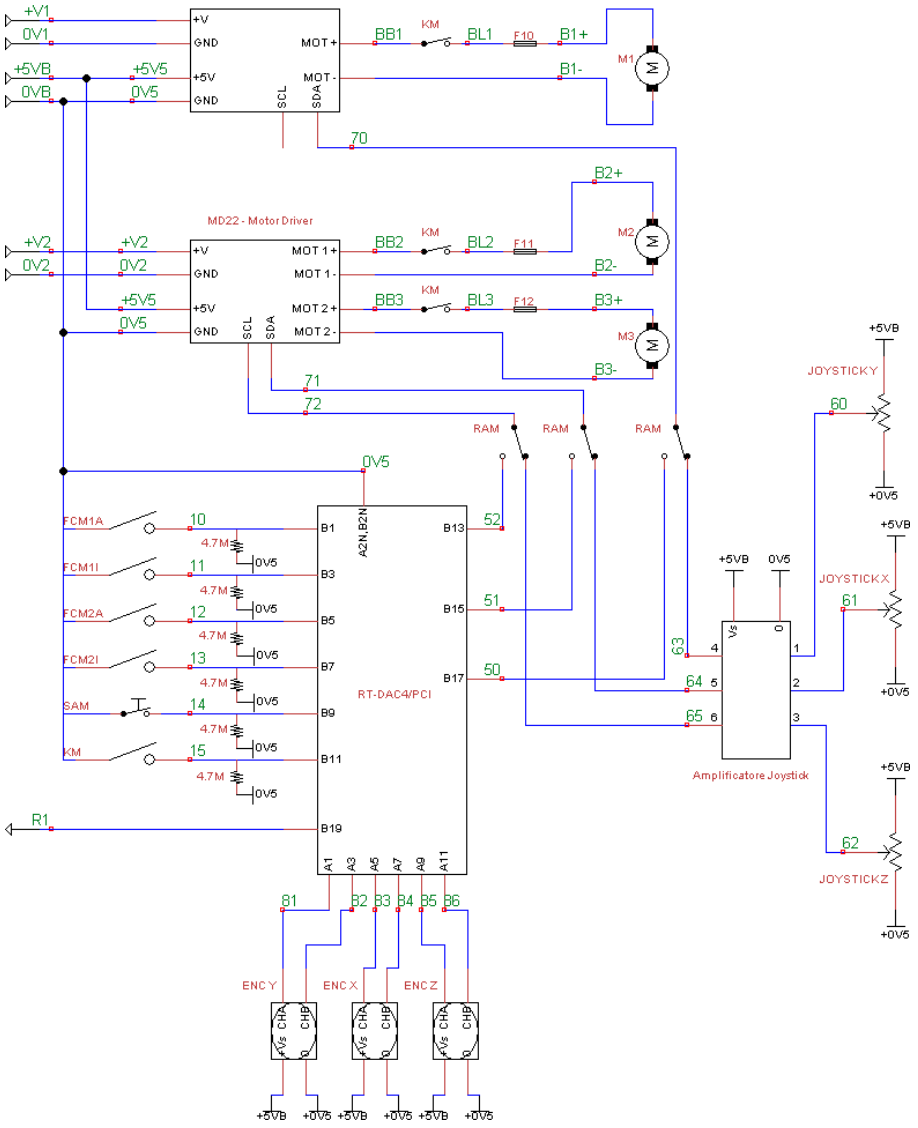


Figura 3.14: Schema elettrico completo - parte 2

Fusibile	Corrente max
F1A	4A
F1B	4A
F2	2A
F3	2A
F4	2A
F5	10A
F6	10A
F7	6A
F8	2A
F9	4A
F10	10A
F11	4A
F12	2A

Tabella 3.7: Valori dei fusibili utilizzati

4 Controllo

E' stato implementato un software di controllo che permette, da PC, di comandare il dispositivo per raggiungere una determinata posizione (set point). Come già accennato, non si è fatto a tempo a integrare il sistema di rilevazione degli angoli α e β tramite telecamera in questa tesi: l'informazione che il software ha del sistema è dunque incompleta. Questo limita a priori le prestazioni del controllore. In primo luogo, ovviamente, non è possibile rispettare la specifica di controllo sugli angoli stessi. In secondo luogo, si noti dall'equazione (2.14) che la stessa accelerazione in x dipende per alcuni termini dall'angolo di oscillazione (lo stesso varrebbe per l'accelerazione lungo y). Non essendo disponibile la misura di questa grandezza siamo costretti a trascurare questa dipendenza: dal punto di vista fisico, ciò equivale a trascurare la dinamica del carico. Useremo quindi per il controllo un modello ulteriormente semplificato, sapendo però che i termini trascurati agiscono come disturbi sul sistema. Non si tratta peraltro un effetto di grande entità avendo il carico massa molto inferiore a binario e carrello. Il sistema generale, MIMO e fortemente non lineare, si riduce con queste semplificazione tre sistemi SISO disaccoppiati, lineari e piuttosto semplici.

4.1 Modelli semplificati in x, y e z

Trascurata da dinamica del carico, le equazioni di moto in x e y si riducono a:

$$\ddot{x}_g = \frac{F_1 - k_1 \dot{x}_g}{m_c} \tag{4.1}$$

$$\ddot{y}_g = \frac{F_2 - (k_1 + k_3)\dot{y}_g}{m_b + m_c} \tag{4.2}$$

Per il moto in x, quindi:

$$F_1 = m_c \ddot{x}_g + k_3 \dot{x}_g$$

Essendo C_u la coppia fornita, R_p il raggio delle puleggie, e ω la velocità angolare del motore M_1, vale che:

$$\begin{aligned} F_1 &= \frac{C_u}{R_p} \\ \dot{x}_g &= \omega R_p \end{aligned}$$

Sostituendo nell'equazione precedente:

$$C_u = m_c R_p^2 \omega + k_3 R_p \dot{\omega}$$

Quest'ultima è formalmente equivalente alla 2.2.2: la funzione di trasferimento del sistema meccanico è dunque del tipo:

$$G(s) = \frac{\Omega(s)}{C_u(s)} = \frac{1}{J_t s + B_t}$$

dove, nel calcolo di J_t e B_t, bisognerebbe sommare anche il contributo dei vari ingranaggi (trasmissione, pulegge, ...) che per brevità non abbiamo considerato. Uguale è il ragionamento per il moto lungo y, per il quale procedendo nei conti si otterrebbe una funzione di trasferimento dello stesso tipo. Vale quindi, in entrambe i casi, lo schema a blocchi semplificato in figura 2.7. Anche il moto in z è modellizzato trascurando le oscillazioni, e vale dunque un modello molto semplificato:

$$\ddot{z} = \frac{F_3 - k_3 \dot{z}}{m_m} - g \tag{4.3}$$

Procedendo nei conti si potrebbe mostrare che vale anche in questo caso lo schema a blocchi di figura 2.7, in cui il disturbo C_d è dovuto al peso del carico.

4.2 Identificazione del modello

Date le semplificazioni effettuate, è valido in x, y e z un modello tra tensione di alimentazione del motore e velocità angolare dell'albero del tipo di (2.24), qui sotto riportata:

$$G(s) = \frac{K_M}{1 + sT_M}$$

La risposta tra tensione di alimentazione e spostamento è quindi:

$$P(s) = \frac{\Theta(s)}{V_a(s)} = \frac{K_M}{s(1 + sT_M)} \tag{4.4}$$

Data la difficoltà di stima dei parametri è sembrato più rapido, semplice e opportuno effettuare un'identificazione del sistema dinamico tramite esperimenti sul sistema reale. Il comando OPENLOOPY del software di controllo esegue un test raccogliendo i dati in tempo reale. Il test consiste in una risposta del sistema a uno scalino di tensione sul motore relativo all'asse y. Per rendere trascurabile l'effetto dell'attrito lo scalino di tensione di prova ha il valore massimo, ovvero $V_h = 24V$. Infatti, facendo riferimento alle equazioni 2.24 e 2.25, si ha che:

$$\Omega(s) = \frac{K_M}{1 + sT_M} V(s) + \frac{K_C}{1 + sT_M} C_d(s)$$

L'ingresso $C_d(s)$ è anch'esso uno scalino di modulo costante C_h e segno negativo:

$$\Omega(s) = \frac{K_M}{1 + sT_M} \frac{V_h}{s} - \frac{K_C}{1 + sT_M} \frac{C_h}{s}$$

o anche:

$$\Omega(s) = V_h(K_M - K_C \frac{C_h}{V_h}) \frac{1}{1 + sT_M}$$

Non considerando l'attrito nell'identificazione sottostimiamo quindi K_M della quantità:

$$K_C \frac{C_h}{V_h}$$

Il grafico 4.1 mostra la risposta allo scalino per l'asse y. L'unità di misura per lo spostamento angolare sono le tacche di lettura dell'encoder. Sono presenti sia il valore di posizione che di velocità angolare. Il primo valore è letto direttamente dal sensore, il secondo è approssimato con la semplice approssimazione:

$$(p(t) - p(t-1))/T_c$$

con T_c tempo di campionamento e $p(t)$ posizione all'istante t. Il grafico di velocità appare più rumoroso di quello di posizione. In effetti l'operazione di derivazione della posizione comporta l'amplificazione delle alte frequenze, in cui il rumore è collocato.

I parametri K_M e T_M possono essere stimati per via grafica guardando l'andamento della velocità angolare e considerando che corrisponde a una funzione di trasferimento del tipo di (2.24). Essendo un semplice sistema

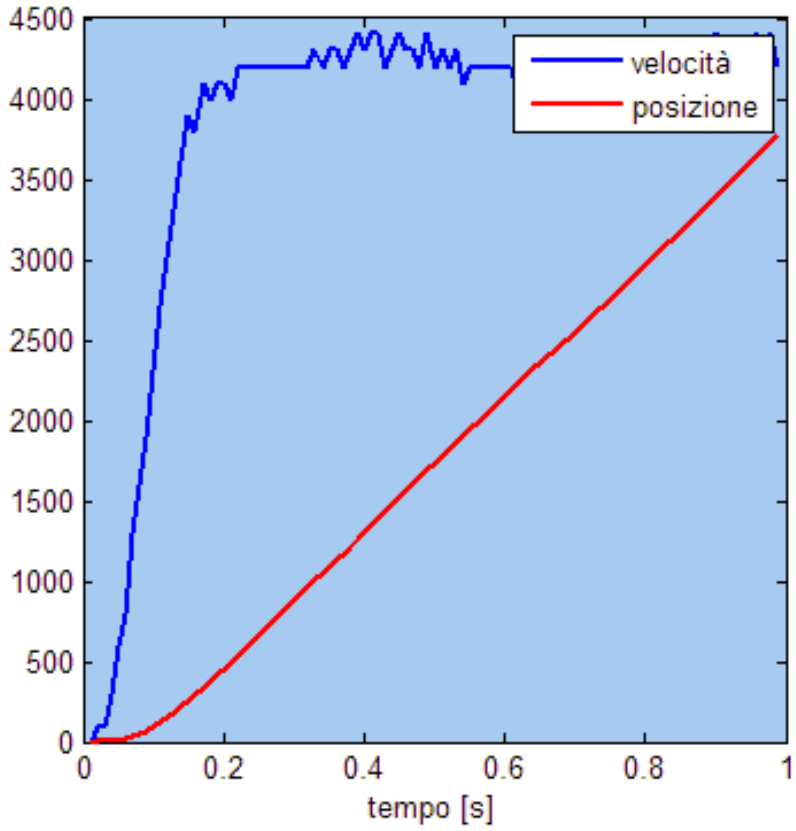


Figura 4.1: fig: Risposta allo scalino per l'asse y

a un solo polo reale, l'identificazione del modello dalla risposta allo scalino può essere fatta cercando il valore di equilibrio e la durata del transitorio. Per avere senso dal punto di vista numerico, fissiamo le unità di misura di ingresso, uscita e tempo della funzione di trasferimento. Per la tensione l'unità è definita dai livelli utilizzati per pilotare la scheda di I/0: in questa scala i $24V$ al motore equivalgono al valore numerico 1023. Per la posizione angolare l'unità di misura è la tacca di lettura dell'encoder. Per il tempo l'unità di misura sono i secondi. Essendo il valore di equilibrio della velocità angolare ≈ 4300,

$$K_M \approx 4300/1023 = 4.2$$

Il valore di equilibrio è raggiunto dopo circa 0.3, quindi $5T_M = 0.3$, da cui:

$$T_M \approx 0.06$$

Un'identificazione più accurata è stata effettuata tramite il System Identification ToolBox di Matlab:

```
m = idproc('P1I');   % il modello ideale della risposta tensione->posizione
                     % con parametri da calcolare
INPUT = 1023*ones(size(POSIZIONE)); % lo scalino di tensione in ingresso
DATA = iddata(POSIZIONE, INPUT, 0.01); % i dati in ingresso e uscita
me = pem(DATA, m) % il modello con parametri stimati
```

da cui otteniamo i valori $K_M = 3.9731$ e $T_M = 0.058001$

Un'analogo test è stato effeffuato per la risposta dell'asse x, per il quale troviamo che $K_M = 3$ e $T_M = 0.06$. Per l'asse z, invece, $K_M = 3$ e $T_M = 0.03$.

4.3 Implementazione

Il software di controllo è stato realizzato in linguaggio C sotto Windows XP, essendo le librerie della scheda di acquisizione disponibili per tale architettura. E' stato utilizzato l'ambiente di sviluppo Visual Studio 2005. I sorgenti del programma sono riportati nell'appendice B.

4.4 Controllore PI

Sono stati progettati dei semplici schemi di controllo con retroazione dell'uscita (fig.4.2). E' stata utilizzata la tecnica di sintesi in tempo continuo per tentativi in frequenza. La funzione di trasferimento continua del regolatore definita in fase di progetto è stata poi implementata su calcolatore mediante un'approssimazione discreta. In un primo tempo sono stati provati dei regolatori di tipo PI.

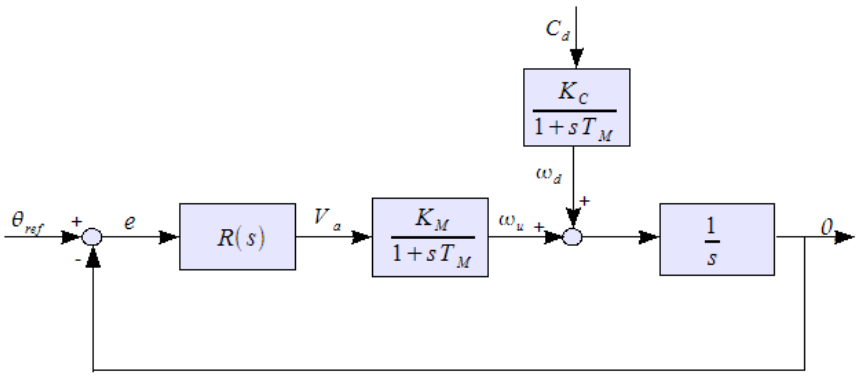


Figura 4.2: Schema di controllo per un asse

Il requisito primario è garantire un posizionamento preciso, ovvero errore a regime nullo. Il controllore PI è il più semplice che possa garantire questa specifica in quanto il disturbo sull'uscita (la posizione angolare dell'albero) è asintotico a una rampa. Infatti, facendo riferimento alla figura 4.2, si ha che C_d comprende un'ingresso a scalino, che deriva dall'attrito. Il disturbo ω_d è quindi a sua volta asintotico a uno scalino, e il disturbo sull'uscita è l'integrale di ω_d. Per garantire errore a regime nullo occorrono dunque due integratori nella funzione di anello di cui uno è già nel processo: un secondo deve essere messo nel regolatore. Un controllo puramente integrale non potrebbe però garantire la stabilità del sistema di controllo in quanto avrebbe un margine di fase negativo. Il diagramma di Bode della fase della funzione di anello partirebbe infatti da -180 gradi a pulsazione nulla e avrebbe un andamento strettamente decrescente. Occorre dunque avere almeno uno zero nel regolatore, che deve essere posizionato in modo da alzare appportunamente il diagramma della fase in corrispondenza della pulsazione di taglio. Il controllore PI, la cui funzione di trasferimento è:

$$R_{PI}(s) = K_P + \frac{K_I}{s} = K_r \frac{1 + sTr}{s} \tag{4.5}$$

ha esattamente un integratore e uno zero, e può dunque rispettare le specifiche di progetto.

4.4.1 Controllo dell'asse y

La funzione di trasferimento tra tensione di comando al motore e posizione per l'asse y è stata approssimata a:

$$P(s) = \frac{4}{s(1 + 0.06s)}$$

il cui diagramma di Bode è in figura 4.3.

E' stato sintetizzato il regolatore

$$R_{PI}(s) = \frac{V(s)}{E(s)} = K_r \frac{1 + sT_r}{s} = 0.025 \frac{1 + s/0.1}{s} \tag{4.6}$$

In modo da avere la pulsazione di taglio in $\omega_0 \approx 1$ rad/sec. Il diagramma di Bode della funzione d'anello:

$$L_{PI}(s) = R_{PI}(s)P(s)$$

è in figura 4.4. Lo zero del regolatore in 0.1 rad/sec alza opportunamente il diagramma della fase in corrispondenza della pulsazione di taglio e il margine

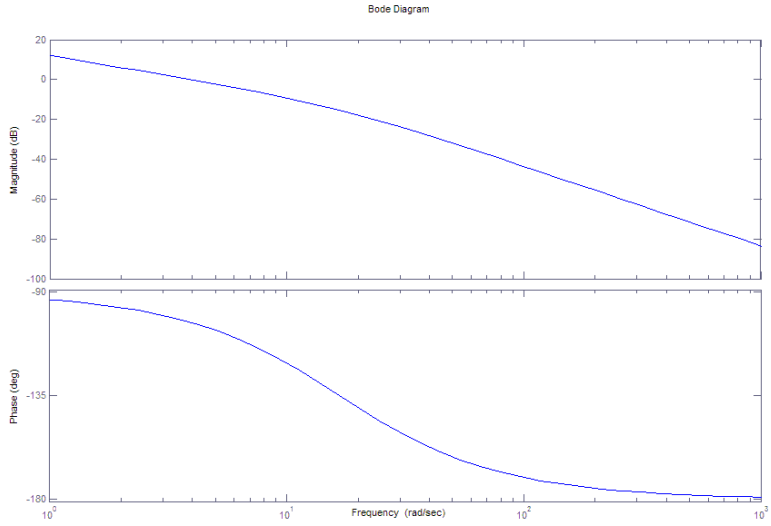


Figura 4.3: Diagramma di Bode di $P(s)$

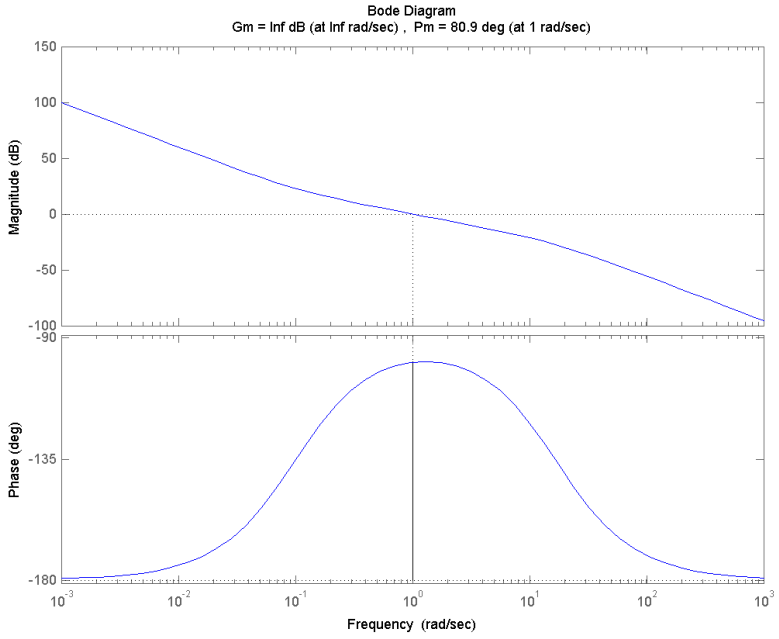


Figura 4.4: Diagramma di Bode di $L_{PI}(j\omega)$

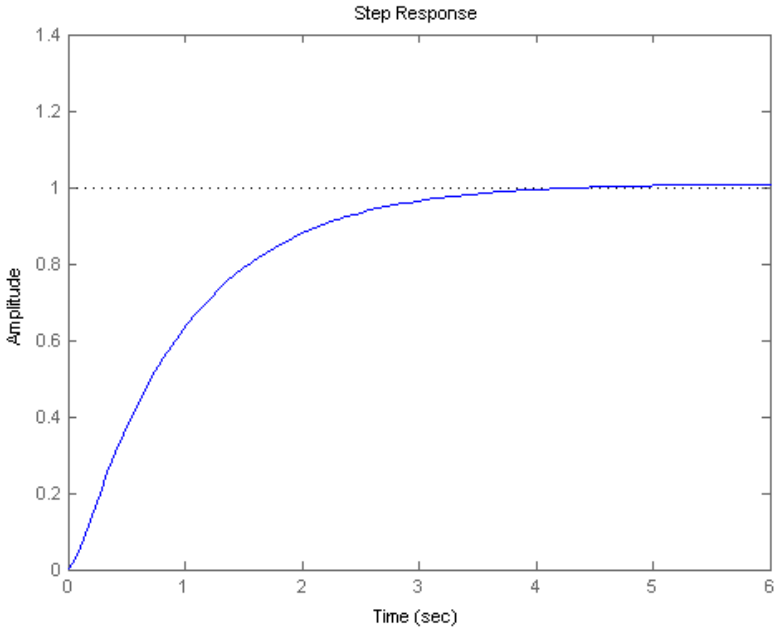


Figura 4.5: Risposta allo scalino di F_{PI}(s)

di fase risulta essere di 80.9 gradi. La funzione di trasferimento in anello chiuso è:

$$F_{PI}(s) = \frac{L_{PI}(s)}{1 + L_{PI}(s)} = \frac{1 + s/0.1}{(1 + s/0.0101)(1 + s/1.058)(1 + s/15.6)}$$

Il polo in 0.0101 risulta quasi perfettamente cancellato dallo zero e quindi il polo dominante è quello in 1.058. Come previsto dalla teoria il sistema in anello chiuso è stabile e non oscillante in virtù del margine di fase elevato, ha banda passante circa uguale alla pulsazione di taglio della funzione di anello e guadagno precisamente unitario, per la presenza di integratori nella funzione di anello. Non è possibile con un PI allargare di molto la banda passante rispetto a questo progetto in quanto si inizierebbe a sentire l'influenza del polo del motore e il sistema in anello chiuso diventerebbe oscillante o anche instabile.

Il regolatore è stato discretizzato con il metodo di Tustin. La funzione di trasferimento del regolatore discreto si trova nel dominio della trasformata Zeta facendo la sostituzione:

$$s = \frac{2}{T_c}\frac{z - 1}{z + 1} \tag{4.7}$$

dove T_c è il tempo di campionamento. La versione discreta del regolatore ha quindi funzione di trasferimento:

$$\frac{K_r}{2}\frac{z(T_c+2T_r)+(T_c-2T_r)}{z-1}$$

la cui legge di controllo può essere espressa nel dominio del tempo discreto in modo algoritmico:

$$v^*(k) = v^*(k-1) + K_r(T_c+2T_r)e^*(k) + K_r(T_c-2T_r)e^*(k-1)$$

Il tempo di campionamento T_c è stato fissato a $10ms$ e il margine di fase perso per la discretizzazione è approssimativamente

$$\phi_0 = \frac{\omega_0}{\omega_c}180 \approx 0.3 \text{ gradi}$$

dove $\omega_c = 2\pi/T_c$. E' stato inoltre implementato un sistema l'anti-windup per impedire all'azione integrale di saturare oltre i limiti dell'attuatore.

4.5 Controllore PID

Le prestazioni del controllore PI garantiscono la precisione statica ma sono risultate poco soddisfacenti dal punto di vista dinamico. Il problema rilevante osservato è l'eccessiva lentezza con cui avviene la reiezione del disturbo in prossimità del raggiungimento del set point. Aggiungendo una componente derivativa è stato possibile migliorare significativamente le prestazioni del sistema. Un controllore PID in forma ideale ha funzione di trasferimento:

$$R^i_{PID}(s) = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s} \tag{4.8}$$

Si hanno quindi a disposizione due zeri nella funzione di trasferimento, uno in più rispetto al PI. L'idea immediata è di cancellare con uno dei due zeri il polo del motore in modo da poter allargare maggiormente la banda passante senza avere un indesiderato decremento del matgine di fase.

4.5.1 Controllo dell'asse y

E' stato progettato per l'asse y un controllore PID in modo da avere una banda passante attorno i 10 rad/sec. Uno dei due zeri è stato sfruttato per cancellare il polo del motore in $1/T_M \approx 16.67$ rad/sec mentre l'altro è posto in 1 rad/sec, in modo da alzare il diagramma della fase in corrispondenza

della pulsazione di taglio. La funzione di trasferimento ideale del controllore è:

$$R^i_{PID}(s) = \frac{2.5(1+s)(1+s/16.67)}{s} = \frac{0.15s^2 + 2.65s + 2.5}{s}$$

e si ha dunque $K_D = 0.15$, $K_P = 2.65$ e $K_I = 2.5$. Per rendere realizzabile la funzione di trasferimento è stato aggiunto un polo in alta frequenza:

$$R_{PID}(s) = 2.65 + \frac{2.5}{s} + \frac{0.15s}{1+s/200}$$

Quest'ultima può essere discretizzata e implementata algoritmicamente, come già visto nel caso del PI. Definiamo anche nel caso del PID la funzione di anello:

$$L_{PID}(s) = R_{PID}(s)P(s)$$

La funzione di trasferimento in anello chiuso è:

$$F_{PID}(s) = \frac{L_{PID}(s)}{1 + L_{PID}(s)} = \frac{(1+s)}{(1+s/189.5)(1.s/9.372)(1+s/1.126)}$$

Il cui polo dominante è in 9.372 rad/sec. Come si vede dalla figura 4.6, il margine di fase per il sistema di controllo continuo è di 81.4 gradi. L'implementazione discreta del regolatore comporta, campionando a $10ms$, una perdita di margine di fase di circa 3 gradi.

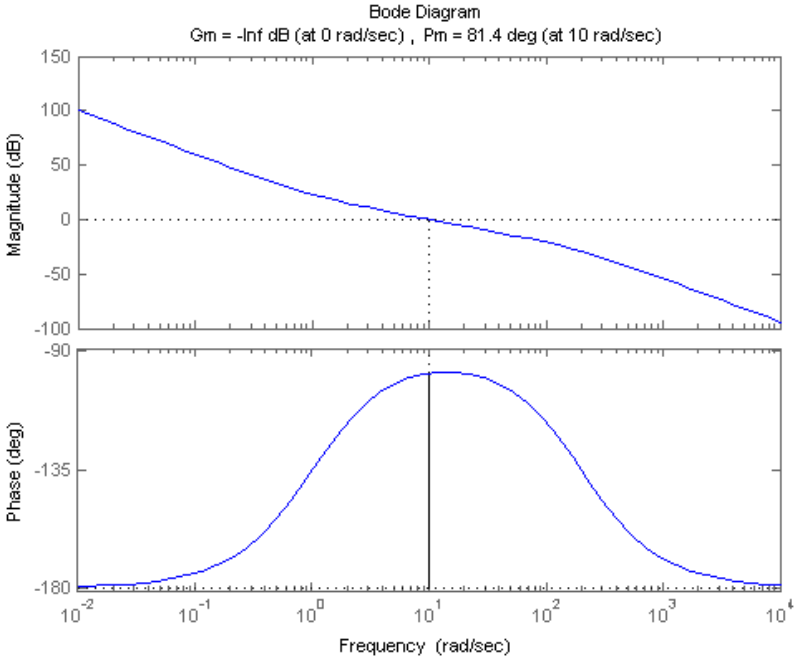


Figura 4.6: Diagramma di Bode di $L_{PID}(j\omega)$

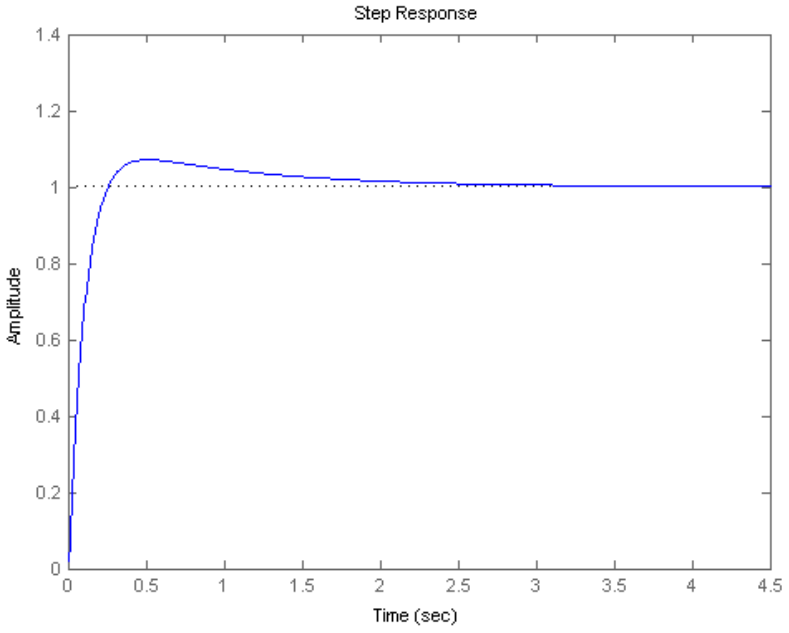


Figura 4.7: Risposta allo scalino di $F_{PID}(s)$

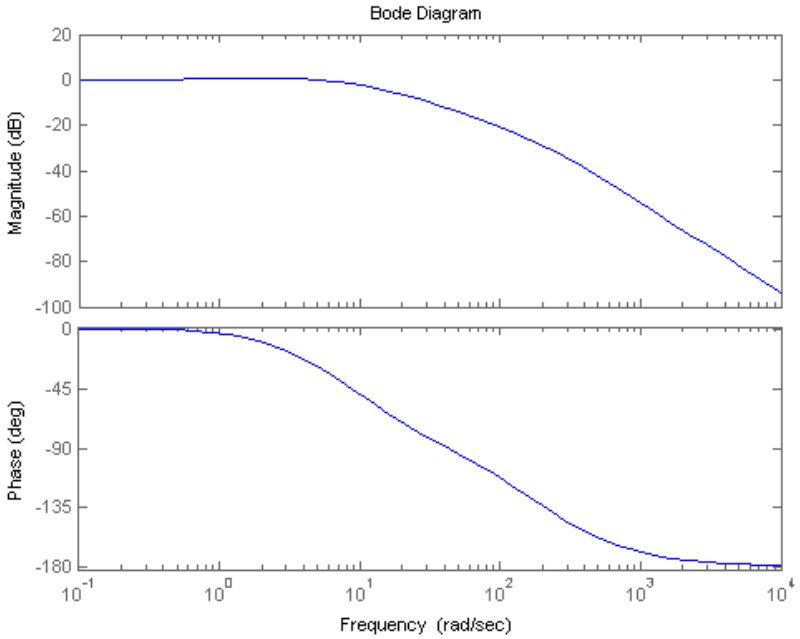


Figura 4.8: Diagramma di Bode di $F_{PID}(s)$

4.5.2 Controllo degli assi x e z

Un analogo controllore PID è stato implementato per gli assi x e z. Il due regolatori garantiscono prestazioni simili al precendente, con banda passante in anello chiuso di circa 10 rad/sec.
I parametri relativi al regolatore dell'asse x sono $K_d = 0.2$, $K_p = 3.433$, $K_i = 3.333$.
I parametri relativi al regolatore dell'asse z sono $K_d = 0.1$, $K_p = 3.433$, $K_i = 3.333$.
Il tempo di campionamento è sempre di $10ms$, e il polo in alta frequenza per la componente derivativa è in 200 rad/sec per entrambe i regolatori.

4.5.3 Prestazioni reali

E' qui analizzato il comportamento del sistema di controllo reale. Un primo esperimento riguarda il raggiungimento di un set point costante, ovvero di una coordinata spaziale $(x_{ref}, y_{ref}, zref)$. Le figure 4.9, 4.10 e 4.11 mostrano le risposte degli assi y, x e z. E' messo in grafico l'andamento del tempo della posizione, del comando ai motori e il set point. Le unità di misura sono sempre le tacche dell'encoder per la posizione e i livelli di tensione della

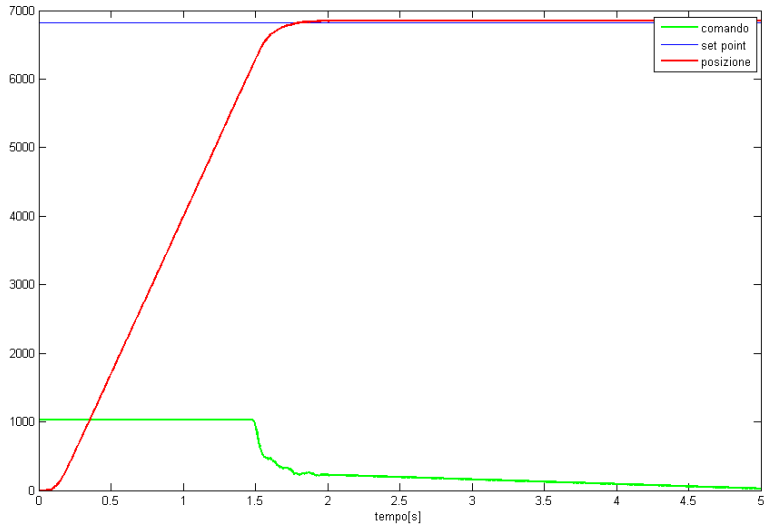


Figura 4.9: Risposta dell'asse y a un set point costante.

scheda di I/O per la variabile di comando.s Il grafico di posizione ha un lieve overshot in tutti e tre i casi, appena percettibile nelle figure. Il comando ai motori è in saturazione per la maggior parte del tempo di moto e diminuisce fino a tendere a zero in prossimità del set point impostato.

Un secondo esperimento è consistito nel dare come set point un riferimento sinusoidale agli assi del binario e del carrello. L'obiettivo è tracciare nel piano xy una traiettoria circolare. I riferimenti sono dunque due sinusoidi sfasate di $\pi/2$, con identica pulsazione ω e ampiezza r, sommati a una costante che rappresenta le coordinate del centro:

$$\begin{aligned} y_{ref}(t) &= y_c + r\cos(\omega t) \\ x_{ref}(t) &= x_c + r\sin(\omega t) \end{aligned}$$

Perchè l'obiettivo sia effettivamente raggiungibile occorre che la pulsazione ω sia sensibilmente inferiore alla banda del sistema in anello chiuso. In questa regione, infatti, la funzione di trasferimento in anello chiuso $F(j\omega)$ ha modulo circa unitario e fase nulla (si guardi ad esempio la figura 4.8 per l'asse y). La stessa sinusoide in ingresso si trova quindi a regime sull'uscita senza distorsioni evidenti. Occorre inoltre che la variabile di comando ai motori non saturi durante il moto. Dei valori che si sono rilevati ammissibili sono $\omega = 1.5$ rad/sec e $r = 150mm$. I grafici 4.12 e 4.13 mostrano l'andamento nel tempo del set point e la posizione effettiva per gli assi y e x. Il grafico 4.14

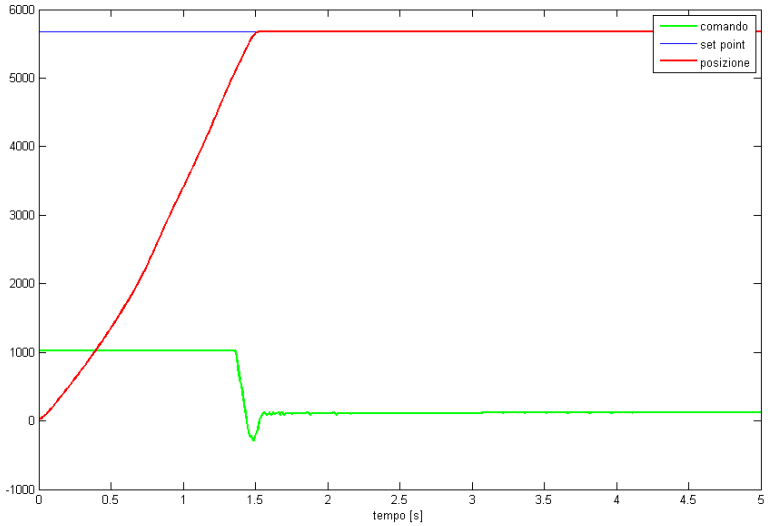


Figura 4.10: Risposta dell'asse x a un set point costante.

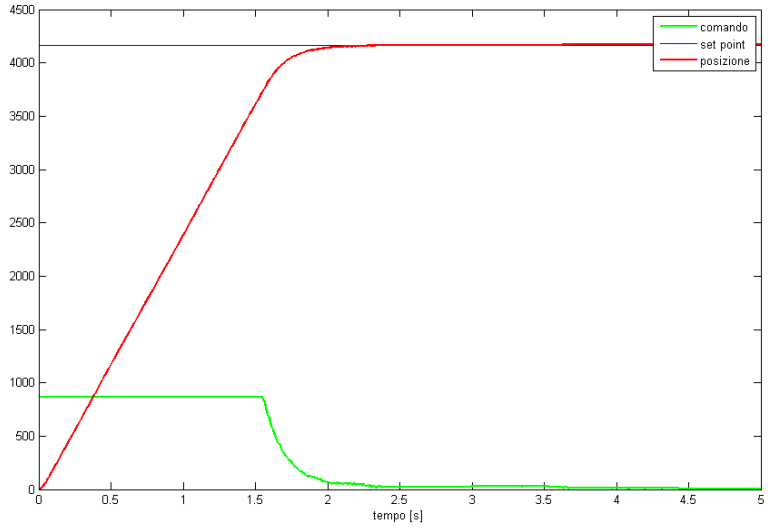


Figura 4.11: Risposta dell'asse z a un set point costante.

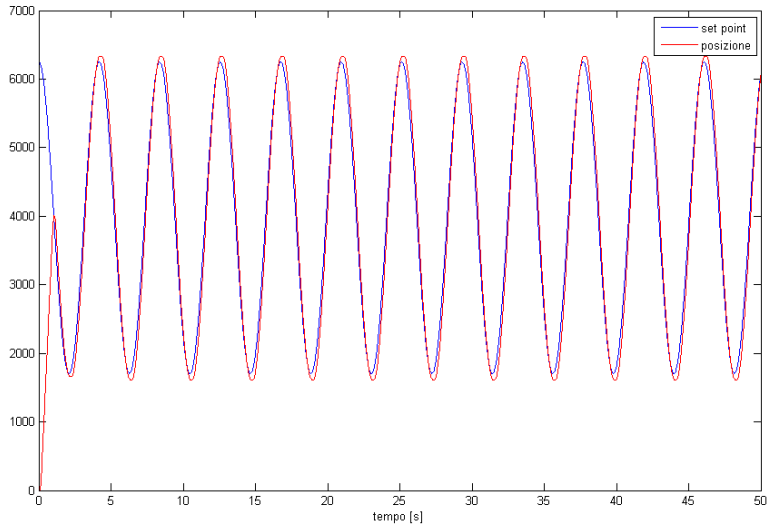


Figura 4.12: Risposta dell'asse y a un set point sinusoidale

mostra invece la traiettoria tracciata nel piano xy dal set point (traiettora ideale) e dalla posizione effettiva del carrello (traiettoria reale).

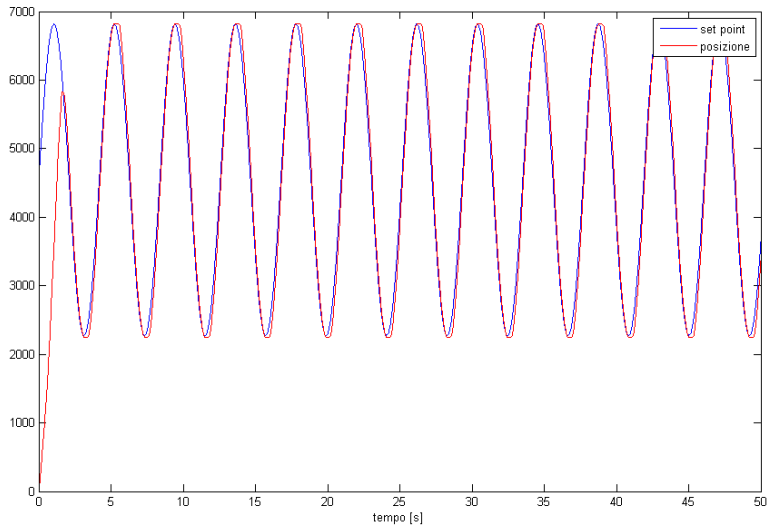


Figura 4.13: Risposta dell'asse x a un set point sinusoidale

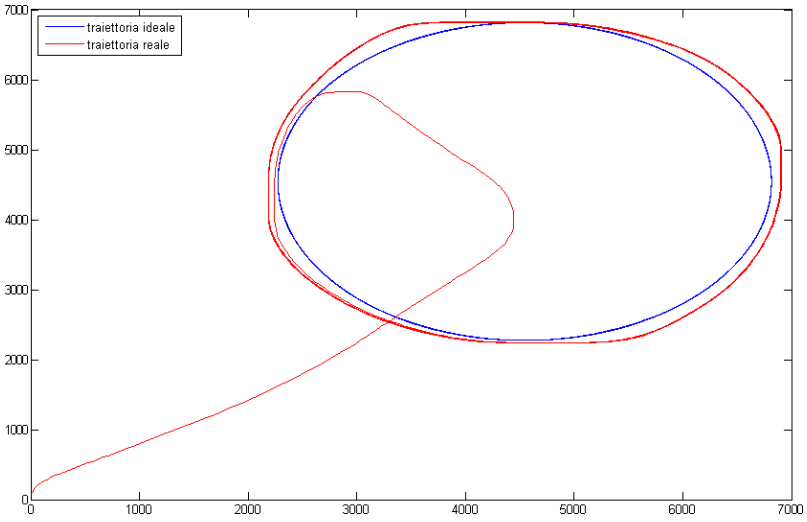


Figura 4.14: Traiettoria seguita del carrello e dal set point

5 Conclusioni

In questa tesi è stato realizzato un sistema di controllo per un'apparato meccanico abbastanza complesso, completo però solo sotto alcuni aspetti. Il lavoro svolto si presta a ulteriori sviluppi in varie direzioni. Sono qui analizzati alcuni dei punti che possono essere approfonditi o migliorati.

Dal punto di vista meccanico, il motore dell'asse x risulta leggermente sottodimensionato e andrebbe sostituito con uno più potente.

Dal punto di vista modellistico, sarebbe utile ricavare i valori numerici di tutti i parametri fisici considerati. Per alcuni di questi la soluzione è immediata. Le masse di carrello, binario e carico, ad esempio, si possono direttamente misurare. Altri sono un poco più laboriosi da stimare: ad esempio i momenti di inerzia dei vari componenti (pulegge, trasmissioni, alberi...) si possono calcolare avendo le misure di massa e dimensioni. Altri ancora sono più complessi, in particolare quelli riguardanti i fenomeni di attrito. I coefficienti di attrito viscoso k_1, k_2 e k_3 non sono di facile stima, così come i valori dell'attrito dinamico e dell'attrito statico massimo (di distacco). I parametri caratteristici dei motori sono riportati dalle case costruttrici, ma quelli che sono riuscito a reperire sono incompleti. Per motivi di tempo non sono riuscito ad affrontare il problema della stima dei parametri fisici e ho seguito un'altra via. Ho ricavato per via teorica delle relazioni ingresso/uscita complessive, peraltro significativamente semplificate. Tramite degli esperimenti ho poi ricavato i valori numerici di un numero ristretto di parametri relativi alle equazioni semplificate (non i parametri fisici del sistema). Se si volesse utilizzare schemi di controllo più complessi, potrebbe essere necessari modelli più accurati. Avere a di sposizione i parametri fisici effettivi permetterebbe l'utilizzo di un qualsiasi modello.

Dal punto di vista del controllo molte cose possono essere migliorate: prima di tutto è auspicabile l'integrazione con il sistema di rilevazione degli

angoli α e β tramite telecamera per poter rispettare il vincolo di moderazione delle osclillazioni, fin qui completamente trascurato.

A

Il modello lagrangiano generale

In questa appendice sono riportati i calcoli per ottenere le equazioni del modello generale, ovvero il modello con movimento in tre dimensioni. Come nel caso unidimensionale, è stato necessario avvalersi per i conti del software del software Maple 10.

La lagrangiana del sistema è:

$$\begin{aligned} L &= \frac{1}{2}m_m((\dot{x}_g + \dot{l}\cos\alpha - l\dot{\alpha}\sin\alpha)^2 + (\dot{y}_g + \dot{l}\sin\alpha\sin\beta + \\ &+ l\dot{\alpha}\cos\alpha\sin\beta + l\sin\alpha\dot{\beta}\cos\beta)^2 + (-\dot{l}\sin\alpha\cos\beta + \\ &- l\dot{\alpha}\cos\alpha\cos\beta + l\sin\alpha\dot{\beta}\sin\beta)^2) + \frac{1}{2}m_c(\dot{x}_g^2 + \dot{y}_g^2) + \\ &+ \frac{1}{2}m_b(\dot{y}_b^2) - m_m g l\sin\alpha \end{aligned}$$

Ricaviamo quindi le componenti delle forze non derivabili da potenziali lungo le coordinate lagrangiane, secondo la formula (2.4):

$$\begin{aligned} Q_{x_g} &= F_1 - k_1\dot{x_g} - k_3\dot{x}_g - k_3\dot{l}\cos\alpha + k_3 l\sin\alpha\dot{\alpha} \\ Q_{y_g} &= F_2 - k_2\dot{y_g} - k_1\dot{y_g} - k_3\dot{y_g} - k_3\dot{l}\sin\beta\sin\alpha \\ &\quad -k_3 l\cos\beta\dot{\beta}\sin\alpha - k_3 l\sin\beta\cos\alpha\dot{\alpha} \\ Q_l &= -F_3 - k_3\dot{l} - k_3\dot{x}_g\cos\alpha - k_3\dot{y}_g\sin\alpha\sin\beta \\ Q_\alpha &= -k_3 l(-\dot{x}_g\sin\alpha + l\dot{\alpha} + \dot{y}_g\sin\beta\cos\alpha) \\ Q_\beta &= -k_3 l\sin\alpha(\dot{y}_g cos\beta + l\dot{\beta}\sin\alpha) \end{aligned} \tag{A.1}$$

Si possono quindi scrivere cinque equazioni di Lagrange (tante quante i gradi di libertà), usando la formula (2.3). La prima equazione, in forma simbolica, è:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x_g}} - \frac{\partial L}{\partial x_g} = Q_{x_g} \tag{A.2}$$

e quindi:

$$\begin{aligned} F_1 \quad & -k_1\dot{x}_g - k_3\dot{x}_g - k_3\dot{l}\cos\alpha + k_3 l\sin\alpha\dot{\alpha} = -2m_m\dot{l}\sin\alpha\dot{\alpha} + \\ - \quad & m_m l\cos\alpha\dot{\alpha}^2 - m_m l\sin\alpha\ddot{\alpha} + m_m\ddot{l}\cos\alpha + m_c\ddot{x}_g + m_m\ddot{x}_g \end{aligned}$$

La seconda equazione, in forma simbolica, è:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{y_g}} - \frac{\partial L}{\partial y_g} = Q_{y_g} \tag{A.3}$$

e quindi:

$$\begin{aligned} F_2 \quad & -k_2\dot{y_g} - k_1\dot{y_g} - k_3\dot{y_g} - k_3\dot{l}\sin\beta\sin\alpha - k_3 l\cos\beta\dot{\beta}\sin\alpha \\ - \quad & k_3 l\sin\beta\cos\alpha\dot{\alpha} = m_m\ddot{l}\sin\beta\sin\alpha + 2m_m\dot{l}\cos\beta\dot{\beta}\sin\alpha \\ + \quad & 2m_m\dot{l}\sin\beta\cos\alpha\dot{\alpha} - m_m l\sin\beta\dot{\beta}^2\sin\alpha + m_m l\cos\beta\ddot{\beta}\sin\alpha \\ + \quad & 2m_m l\cos\beta\dot{\beta}\cos\alpha\dot{\alpha} - m_m l\sin\beta\sin\alpha\dot{\alpha}^2 \\ + \quad & m_m l\sin\beta cos\alpha\ddot{\alpha} + m_b\ddot{y}_g + m_c\ddot{y}_g + m_m\ddot{y}_g \end{aligned}$$

La terza equazione, in forma simbolica, è:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{l}} - \frac{\partial L}{\partial l} = Q_l \tag{A.4}$$

e quindi:

$$\begin{aligned} - \quad & F_3 - k_3\dot{l} - k_3\dot{x}_g\cos\alpha - k_3\dot{y}_g\sin\alpha\sin\beta = \\ - \quad & m_m l\dot{\alpha}^2 + m_m l\dot{\beta}^2\cos\alpha^2 - m_m g\cos\beta\sin\alpha \\ + \quad & m_m\ddot{y}_g\sin\beta\sin\alpha + m_m\ddot{x}_g\cos\alpha + m_m\ddot{l} \\ - \quad & m_m l\dot{\beta}^2 \end{aligned}$$

La quarta equazione, in forma simbolica, è:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\alpha}} - \frac{\partial L}{\partial \alpha} = Q_\alpha \tag{A.5}$$

e quindi:

$$\begin{aligned} - \quad & k_3 l(-\dot{x}_g\sin\alpha + l\dot{\alpha} + \dot{y}_g\sin\beta\cos\alpha) = \\ - \quad & m_m l^2\dot{\beta}^2\cos\alpha\sin\alpha - m_m g l\cos\beta\cos\alpha \\ - \quad & m_m\ddot{x}_g l\sin\alpha + m_m\ddot{y}_g l\sin\beta\cos\alpha + 2m_m l\dot{\alpha}\dot{l} + m_m l^2\ddot{\alpha} \end{aligned}$$

La quinta equazione, in forma simbolica, è:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\beta}} - \frac{\partial L}{\partial \beta} = Q_\beta \tag{A.6}$$

e quindi:

$$\begin{aligned} & -k_3 l \sin\alpha(\dot{y}_g cos\beta + l\dot{\beta}\sin\alpha) = \\ & m_m g l \sin\beta\sin\alpha + 2m_m l\dot{\beta}\dot{l} + m_m l^2\ddot{\beta} \\ + \quad & m_m \ddot{y}_g l\cos\beta\sin\alpha - 2m_m l\dot{\beta}\cos\alpha^2\dot{l} \\ - \quad & m_m l^2\ddot{\beta}\cos\alpha^2 + 2m_m l^2\dot{\beta}\cos\alpha\sin\alpha\dot{\alpha} \end{aligned}$$

Si noti che le cinque equazioni così trovate sono lineari rispetto a $\ddot{x}_g$, $\ddot{y}_g$, $\ddot{l}$, $\ddot{\alpha}$, $\ddot{\beta}$. Con procedimenti algebrici concettualmente semplice è quindi possibile risolvele esplicitamente rispetto a tali variabili. Tralasciamo di riportare questo passaggio per l'eccessiva lunghezza dei conti risultanti. Come già accennato, avere le equazioni del modello generale è comodo per ricavare modelli semplificati, utili nella successiva fase di controlo. Ad esempio, se fossimo interessati a un modello con lunghezza del cavo fisso a un valore L, ci basta imporre dalle equazioni sopra trovate le condizioni $l = L$, $\dot{l} = 0$ e $\ddot{l} = 0$ e risolvere in funzione di $\ddot{x}_g$, $\ddot{y}_g$, $\ddot{\alpha}$, $\ddot{\beta}$ e F_3, che diventa un parametro. Anche questo passaggio può essere svolto con un semplice comando Maple. Imponendo invece $\ddot{y}_g = 0$, $\dot{y}_g$, $l = L$, $\dot{l} = 0$, $\ddot{l} = 0$, $\beta = 0$, $\dot{\beta} = 0$, $\ddot{\beta} = 0$ ritroviamo, a riconferma dei risultati trovati, le equazioni (2.14) e (2.15).

B

Sorgenti del software di controllo

Sono mostrati in questa appendice i sorgenti del software di controllo, Caronte. I file sono:

- `costants.h` - contiene alcuni parametri di funzionamento (tempo di campionamento, fattore di conversione tra tacche e millimetri, etc)
- `pins.h` - contiene il numero dei pin dei vari segnali
- `rtdac4functions.h` - header file di `rtdac4functions.cpp`
- `controllers.h` - header file di `controllers.cpp`
- `stdafx.h` - header file per inclusioni di file di libreria standard
- `stdafx.cpp` - include `stdafx.h`
- `rtdac4functions.cpp` - contiene le funzioni relative alla scheda di I/O
- `controller.cpp` - contiene l'implementazione degli algoritmi di controllo
- `main.cpp` - il programma principale

costants.h

```
#define RELE_ON 2000 // circa 5V, per pilotare il transistor che accende il rele
#define RELE_OFF 0 // 0V
#define AUTO 1
#define MAN 0

// Costanti relative al posizionamento iniziale (vedi initialPosition )
#define ZERO_SPEED 1023 //2.5V (Stop per i motori)
#define SAFE_SPEED_X_PLUS 500
#define SAFE_SPEED_X_MIN 1500
#define SAFE_SPEED_Y_PLUS 200
#define SAFE_SPEED_Y_MIN 1800


#define MM_TAC 0.088 // conversione da "tacche" encoder a millimetri
#define TAC_MM 11.3636 // conversione da millimetri a tacche

#define MM_TAC_Z 0.12
#define TAC_MM_Z 8.3333

#define TCAMP 0.01f // tempo di campionamento usato
```

pins.h

```
// Configurazione relativa alle uscite analogiche
#define MOTORE_X 0
#define MOTORE_Y 1
#define MOTORE_Z 2
#define RAM 3

// Configurazione relativa agli ingressi analogici
#define FC_ASSEY_AVANTI 10
#define FC_ASSEY_INDIETRO 11
#define FC_ASSEX_AVANTI 12
#define FC_ASSEX_INDIETRO 13
#define SAM 14 // selettore automatico/manuale
#define KM 15 // contattore di marcia

//Configurazione relativa agli encoder
#define ENCODER_X 0
#define ENCODER_Y 1
#define ENCODER_Z 2
```

rtdac4functions.h

```
typedef unsigned char   UI8,  *PUI8;
typedef unsigned short int UI16, *PUI16;
typedef unsigned long int UI32, *PUI32;

typedef UI32 (*T_RtdacPCI_DA)(UI16, UI8, UI16);
typedef UI32 (*T_RtdacPCI_AD)(UI16, UI8, UI8);
typedef UI32 (*T_RtdacPCI_ResetEncoder)(UI16,UI8,UI8);
typedef UI32 (*T_RtdacPCI_ReadEncoder)(UI16,UI8);
typedef int   (*T_NoOfDetectedBoards)();
typedef int   (*T_BoardLocation)(int, int*, int*, int*,  int*,  int*);
```

stdafx.h

```
#pragma once
```

```
#define WIN32_LEAN_AND_MEAN
#include <stdio.h>
#include <tchar.h>
#include <string.h>
#include <windows.h>
#include <stdlib.h>
#include <iostream>
#include <math.h>
```

stdafx.c

```
#include "stdafx.h"
```

rtdac4functions.cpp

```
/** Questo sorgente serve a estrarre le funzioni di libreria dalla dll Rtdacapi.dll
    non avendo a disposizione il file .lib occorre estrarle dinamicamente */
#include "stdafx.h"
#include "rtdac4functions.h"


UI16 BaseAddress;
T_RtdacPCI_DA RtdacPCI_DA;
T_RtdacPCI_AD RtdacPCI_AD;
T_RtdacPCI_ResetEncoder RtdacPCI_ResetEncoder;
T_RtdacPCI_ReadEncoder RtdacPCI_ReadEncoder;
T_NoOfDetectedBoards NoOfDetectedBoards;
T_BoardLocation BoardLocation;

static HINSTANCE rtLib;

int getDllFunctions(){
    rtLib = LoadLibrary(TEXT("Rtdacapi"));
    if (!rtLib) {
        return -1;
    }
    RtdacPCI_DA = (T_RtdacPCI_DA) GetProcAddress(rtLib, "RtdacPCI_DA");
    RtdacPCI_AD = (T_RtdacPCI_AD) GetProcAddress(rtLib, "RtdacPCI_AD");
    RtdacPCI_ResetEncoder = (T_RtdacPCI_ResetEncoder)
                GetProcAddress(rtLib, "RtdacPCI_ResetEncoder");
    RtdacPCI_ReadEncoder = (T_RtdacPCI_ReadEncoder)
                GetProcAddress(rtLib, "RtdacPCI_ReadEncoder");
    NoOfDetectedBoards = (T_NoOfDetectedBoards)
                GetProcAddress(rtLib, "NoOfDetectedBoards");
    BoardLocation = (T_BoardLocation) GetProcAddress(rtLib, "BoardLocation");
    return 0;
}

int setBaseAddress() {
    if(NoOfDetectedBoards() < 1) return -1; // Non c' la scheda RTDAC4/PCI
    int a,b,c,d,e;
    BoardLocation(1, &a, &b, &c, &d, &e);
    BaseAddress = e;
    return 0;
}
```

controller.cpp

```
#include "stdafx.h"
#include "costants.h"

/* Controllore PI per l'asse x (non pi  utilizzato )*/
int xAxisPi(int set, int val){
    float Kr = 0.025f;
    float Tr = 10.0f;
    float Tc = TCAMP; // tempo di campionamento
    float uk, ek;
    static float uk_old, ek_old;
    ek = (float)(set - val);
    uk = (Kr/2)*(Tc + 2*Tr)*ek + (Kr/2)*(Tc - 2*Tr)*ek_old + uk_old;
    if(uk > 1023) uk = 1023;
    if(uk < -1023) uk = -1023;
    uk_old = uk;
    ek_old = ek;
    return (int)uk;
}

/* Controllore PID per l'asse x */
int xAxisPid(int set, int val){
 float Kd = 0.15f;
 float Kp = 2.650f;
 float Ki = 2.50f;
 float Tc = TCAMP;
 float Th = 0.005f;

 float ik, pk, dk, ek, uk;
 ek = (float)(set - val);
 static float ek_old, ik_old, dk_old;

 pk = Kp*ek;

 dk = ((2*Kd)/(Tc + 2*Th))*(ek - ek_old) - ((Tc - 2*Th)/(Tc + 2*Th))*dk_old;
 dk_old = dk;
 ik = ik_old + (Ki*Tc/2)*ek + (Ki*Tc/2)*ek_old;
 if(ik + pk + dk > 1023 || ik + pk + dk < -1023) ik = ik_old;
 ik_old = ik;

 ek_old = ek;
 uk = (ik + pk + dk);
 if(uk > 1023) uk = 1023;
 if(uk < -1023) uk = -1023;
 return (int)(1*(uk));
}

/* Controllore PID per l'asse y */
int yAxisPid(int set, int val){
 float Kd = 0.2f;
 float Kp = 3.433f;
 float Ki = 3.333f;
 float Tc = TCAMP;
 float Th = 0.005f;

 float ik, pk, dk, ek, uk;
 ek = (float)(set - val);
 static float ek_old, ik_old, dk_old;

 pk = Kp*ek;

 dk = ((2*Kd)/(Tc + 2*Th))*(ek - ek_old) - ((Tc - 2*Th)/(Tc + 2*Th))*dk_old;
 dk_old = dk;
```

```
 ik = ik_old + (Ki*Tc/2)*ek + (Ki*Tc/2)*ek_old;
 if(ik + pk + dk > 1023 || ik + pk + dk < -1023) ik = ik_old;
 ik_old = ik;

 ek_old = ek;
 uk = (ik + pk + dk);
 if(uk > 1023) uk = 1023;
 if(uk < -1023) uk = -1023;
 return (int)((uk));
}

/* Controllore PID per l'asse Z */
int zAxisPid(int set, int val){
 float Kd = 0.1f;
 float Kp = 3.433f;
 float Ki = 3.333f;
 float Tc = 0.01f;
 float Th = 0.005f;

 float ik, pk, dk, ek, uk;
 ek = (float)(set - val);
 static float ek_old, ik_old, dk_old;

 pk = Kp*ek;

 dk = ((2*Kd)/(Tc + 2*Th))*(ek - ek_old) - ((Tc - 2*Th)/(Tc + 2*Th))*dk_old;
 dk_old = dk;
 ik = ik_old + (Ki*Tc/2)*ek + (Ki*Tc/2)*ek_old;
 if(ik + pk + dk > 1023 || ik + pk + dk < -1023) ik = ik_old;
 ik_old = ik;

 ek_old = ek;
 uk = (ik + pk + dk);
 if(uk > 1023) uk = 1023;
 if(uk < -1023) uk = -1023;
 return (int)(0.85*(uk));
}
```

main.cpp

```
#include "stdafx.h"
#include "rtdac4functions.h"
#include "pins.h"
#include "costants.h"

using namespace std;

/* Queste funzioni sono importate da rtdacfunctions.cpp */
extern UI16 BaseAddress;
extern T_RtdacPCI_DA RtdacPCI_DA;
extern T_RtdacPCI_AD RtdacPCI_AD;
extern T_RtdacPCI_ResetEncoder RtdacPCI_ResetEncoder;
extern T_RtdacPCI_ReadEncoder RtdacPCI_ReadEncoder;
extern T_NoOfDetectedBoards NoOfDetectedBoards;
extern T_BoardLocation BoardLocation;
extern int getDllFunctions();
extern int setBaseAddress();

/* Gli algoritmi di controllo importati da controllers.cpp*/

extern int xAxisPid(int set, int val);
```

```
extern int yAxisPid(int set, int val);
extern int zAxisPid(int set, int val);
extern int xAxisPi(int set, int val);

/* I comandi riconosciuti*/
typedef enum command {
    UNK = 0,
    EXIT,
    INIT,
    MOVE,
    ENCTEST,
    OPENLOOPX,
    OPENLOOPY,
    OPENLOOPZ,
    HELP,
    CIRCLE,
    GENTEST
} command;


void boardInit();
void setRele(int status);
void initialPosition ();
void safeValues();
void resetEncoders();
void moveTo(int setX, int setY, int setZ, FILE *fp);
int isUp(int signal);
command getCommand(char *comm);
void moveCircle(int centX, int centY, int radius, float w, FILE *fp);


int _tmain(int argc, _TCHAR* argv[])
{
    char line[100]; // la riga acquisita
    char comm[100]; // comando
    char param[100]; // parametro
    int x, y, z; // set point
    int X,Y,Z; // valore raggiunto
    int centX, centY, radius; // centro e raggio per il moto circolare
    float w; // pulsazione per il moto circolare
    FILE *fp; // un file dove salvare i risultati .
    boardInit();
    safeValues();
    while(1){
        setRele(MAN);
        printf("Caronte: ");
        param[0] = 0;
        comm[0] = 0;
        do{
         fgets(line, 100, stdin);
        } while(!strcmp(line, "\n"));
        sscanf_s(line, "%s %s", comm, 100, param, 100);
        switch(getCommand(comm)){
            case EXIT:
                printf("Esecuzione di Caronte terminata correttamente\n");
                return 0;
                break;
            case MOVE:
                setRele(AUTO);
                if(!strcmp(param, "")){
                    fp = stdout;
                }
```

```
        else{
            fopen_s(&fp, param, "w");
        }
        puts("Inserire le coordinate X,Y e Z in millimetri (xxx yyy zzz):");
        cin >> x >> y >> z;
        initialPosition();
        printf("Posizionamento iniziale terminato\n");
        resetEncoders();
        Sleep(1000);
        moveTo(x,y,z, fp);
        X = RtdacPCI_ReadEncoder(BaseAddress, 0);
        Y = RtdacPCI_ReadEncoder(BaseAddress, 1);
        Z = RtdacPCI_ReadEncoder(BaseAddress, 2);
        printf("Posizione raggiunta: %.2f %.2f %.2f\n", X*MM_TAC, Y*MM_TAC, Z*MM_TAC_Z);
        break;
    case INIT:
        setRele(AUTO);
        printf("Sto andando in posizione di partenza\n");
        initialPosition();
        printf("Posizionamento terminato\n");
        break;
    case CIRCLE:
        setRele(AUTO);
        if(!strcmp(param, "")){
            fp = stdout;
        }
        else{
            fopen_s(&fp, param, "w");
        }
        initialPosition();
        printf("Posizionamento iniziale terminato\n");
        resetEncoders();
        puts("Inserire le coordinate di centro, raggio e la pulsazione (xc, yc, r, w)");
        cin >> centX >> centY >> radius >> w;
        moveCircle((int)(centX*TAC_MM), (int)(centY*TAC_MM),
        (int)(radius*TAC_MM), w, fp);
        break;
    case ENCTEST:
        RtdacPCI_ResetEncoder(BaseAddress, 0, 1);
        RtdacPCI_ResetEncoder(BaseAddress, 0, 0);
        RtdacPCI_ResetEncoder(BaseAddress, 1, 1);
        RtdacPCI_ResetEncoder(BaseAddress, 1, 0);
        RtdacPCI_ResetEncoder(BaseAddress, 2, 1);
        RtdacPCI_ResetEncoder(BaseAddress, 2, 0);
        Sleep(1000);
        for(int i = 0; i < 200; i++){
            int a = RtdacPCI_ReadEncoder(BaseAddress, 0);
            int b = RtdacPCI_ReadEncoder(BaseAddress, 1);
            int c = RtdacPCI_ReadEncoder(BaseAddress, 2);
            printf("%d %d %d\n", a,b,c);
            Sleep(500);
        }
        break;
    case OPENLOOPX:
        setRele(AUTO);
        printf(param);
        if(!strcmp(param, "")){
            fp = stdout;
        }
        else{
            fopen_s(&fp,param, "w");
        }
```

```
        printf("Sto andando in posizione di partenza\n");
        initialPosition();
        printf("Posizionamento terminato\n");
        fprintf(fp, "istante, posizione\n");
        resetEncoders();
        RtdacPCI_DA(BaseAddress, MOTORE_X, 2047);
        for(int i = 0; i < 200; i++) {
            X = RtdacPCI_ReadEncoder(BaseAddress, 0);
            fprintf(fp, "%d, %d\n", i, X);
            Sleep(10);
        }
        safeValues();
        fflush(fp);
        break;
    case OPENLOOPY:
        setRele(AUTO);
        printf(param);
        if(!strcmp(param, "")){
            fp = stdout;
        }
        else{
            fopen_s(&fp, param, "w");
        }
        printf("Sto andando in posizione di partenza\n");
        initialPosition();
        printf("Posizionamento terminato\n");
        fprintf(fp, "istante, posizione\n");
        resetEncoders();
        RtdacPCI_DA(BaseAddress, MOTORE_Y, 2047);
        for(int i = 0; i < 100; i++) {
            X = RtdacPCI_ReadEncoder(BaseAddress, 1);
            fprintf(fp, "%d, %d\n", i, X);
            Sleep(10);
        }
        safeValues();
        fflush(fp);
        break;
    case OPENLOOPZ:
        setRele(AUTO);
        printf(param);
        if(!strcmp(param, "")){
            fp = stdout;
        }
        else{
            fopen_s(&fp, param, "w");
        }
        fprintf(fp, "istante, posizione\n");
        resetEncoders();
        RtdacPCI_DA(BaseAddress, MOTORE_Z, 0);
        for(int i = 0; i < 1000; i++) {
            X = RtdacPCI_ReadEncoder(BaseAddress, 2);
            fprintf(fp, "%d, %d\n", i, X);
            Sleep(10);
        }
        safeValues();
        fflush(fp);
        break;

    case HELP:
        if(!strcmp(param, "")){
            printf("Elenco dei comandi utili\n\n");
            printf("INIT - Porta il carroponte in posizione di partenza\n");
```

```
                printf("MOVE - Muove il carico in una posizione x y z\n");
                printf("CIRCLE - Muove il carrello secondo una traiettoria circolare\n");
                printf("OPENLOOPX - Test in anello aperto per l'asse x\n");
                printf("OPENLOOPY - Test in anello aperto per l'asse y\n");
                printf("OPENLOOPZ - Test in anello aperto per l'asse z\n");
                printf("HELP - Questa guida\n");
                printf("EXIT - Esce dal programma\n\n");
                printf("Digita HELP [nomecomando] per maggiori dettagli\n\n");
            }
            switch(getCommand(param)) {
                case MOVE:
                    printf("\nMOVE [file]\n\n");
                    printf("Muove il carico in x y z.\n"
                        "Il parametro opzionale e'"
                        " un file in cui sono salvati i risultati dell'esperimento"
                        " in tempo reale. Se non e' presente sono stampati a video\n\n");
                    break;
                case CIRCLE:
                    printf("\nCIRCLE [file]\n\n");
                    printf("Muove il carello secondo una traietoria circolare.\n"
                        "Il parametro opzionale e'"
                        " un file in cui sono salvati i risultati dell'esperimento"
                        " in tempo reale. Se non e' presente sono stampati a video.\n"
                        "Sono richiesti all'utente x del centro, y del centro,"
                        "raggio e velocita' angolare. \n\n");
                    break;
                case OPENLOOPX:
                case OPENLOOPY:
                case OPENLOOPZ:
                    printf("\nOPENLOOP<X | Y | Z> [file]\n\n");
                    printf("Effettua una simulazione in anello aperto sull'asse"
                        " relativo.\nIl parametro opzionale un file in cui sono"
                        " salvati i risultati dell'esperimento. Se non presente sono"
                        " stampati a video\n\n");
                    break;
                case INIT:
                    printf("\nINIT\n\n");
                    printf("Porta il carrello alla coordinata iniziale in x e y\n\n");
                    break;
                default:
                    printf("\nComando inesistente o non documentato\n\n");

            }
        break;
        case GENTEST:
            setRele(AUTO);
            while(1){
                RtdacPCI_DA(BaseAddress, MOTORE_X, 2047);
                Sleep(12000);
                RtdacPCI_DA(BaseAddress, MOTORE_X, 0);
                Sleep(12000);
            }
            break;
        case UNK:
            printf("Comando non trovato. Digita HELP per avere l'elenco dei comandi\n");
            break;
        default:
            break;
    }
  }
  fclose(fp);
}
```

```
/* Inizializzazione della scheda di I/O */
void boardInit()
{
    if(getDllFunctions() == -1) fprintf(stderr, "Errore: file Rtdacapi.dll non trovato");
    if(setBaseAddress() == -1) fprintf(stderr, "Errore: non stata rilevata la scheda Rtad4/PCI");
}

/* Comando automatico/manuale */
void setRele(int status)
{
    UI16 val = (status == AUTO) ? RELE_ON : RELE_OFF;
    RtdacPCI_DA(BaseAddress, RAM, val);
}

/* Porta il carrello nell'origine */
void initialPosition (){
    int fcMuro, fcSx;

    // Mi avvicino all'inizio dell'asse X fino a toccare il finecorsa
    while(1){
        Sleep(20);
        fcSx = RtdacPCI_AD(BaseAddress, FC_ASSEX_INDIETRO, 1);
        if(isUp(fcSx)) {
            RtdacPCI_DA(BaseAddress, MOTORE_X, ZERO_SPEED);
            break;
        }
        RtdacPCI_DA(BaseAddress, MOTORE_X, SAFE_SPEED_X_PLUS);
    }
    Sleep(500);
    // Mi stacco dal finecorsa asse X
    while(1){
        Sleep(20);
        fcSx = RtdacPCI_AD(BaseAddress, FC_ASSEX_INDIETRO, 1);
        if(!isUp(fcSx)) {
            RtdacPCI_DA(BaseAddress, MOTORE_X, ZERO_SPEED);
            break;
        }
        RtdacPCI_DA(BaseAddress, MOTORE_X, SAFE_SPEED_X_MIN);
    }
    Sleep(500);
    // Mi avvicino all'inizio dell'asse Y fino a toccare il finecorsa
    while(1){
        Sleep(20);
        fcMuro = RtdacPCI_AD(BaseAddress, FC_ASSEY_AVANTI,1);
        if(isUp(fcMuro)) {
            RtdacPCI_DA(BaseAddress, MOTORE_Y, ZERO_SPEED);
            break;
        }
        RtdacPCI_DA(BaseAddress,MOTORE_Y, SAFE_SPEED_Y_PLUS);
    }
    Sleep(500);
    // Mi stacco dal finecorsa
    while(1){
        Sleep(20);
        fcMuro = RtdacPCI_AD(BaseAddress, FC_ASSEY_AVANTI,1);
        if(!isUp(fcMuro)) {
            RtdacPCI_DA(BaseAddress, MOTORE_Y, ZERO_SPEED);
            break;
        }
```

```
        RtdacPCI_DA(BaseAddress,MOTORE_Y, SAFE_SPEED_Y_MIN);
    }
}

/* Mette dei valori "sicuri" sulle uscite */
void safeValues(){
    setRele(MAN);
    RtdacPCI_DA(BaseAddress, MOTORE_X, ZERO_SPEED);
    RtdacPCI_DA(BaseAddress, MOTORE_Y, ZERO_SPEED);
    RtdacPCI_DA(BaseAddress, MOTORE_Z, ZERO_SPEED);
}

/* Il segnale dei finecorsa molto disturbato,
   ma se in questa banda alto */
int isUp(int signal){
    if(signal < 1023 + 100 && signal > 1023 - 100) {
        return 1;
    }
    else {
        return 0;
    }
}

command getCommand(char *comm){
    if(!strcmp(comm, "EXIT")) return EXIT;
    if(!strcmp(comm, "INIT")) return INIT;
    if(!strcmp(comm, "MOVE")) return MOVE;
    if(!strcmp(comm, "ENCTEST")) return ENCTEST;
    if(!strcmp(comm, "OPENLOOPX")) return OPENLOOPX;
    if(!strcmp(comm, "OPENLOOPY")) return OPENLOOPY;
    if(!strcmp(comm, "OPENLOOPZ")) return OPENLOOPZ;
    if(!strcmp(comm, "HELP")) return HELP;
    if(!strcmp(comm, "CIRCLE")) return CIRCLE;
    if(!strcmp(comm, "GENTEST")) return GENTEST;
    return UNK;
}


void moveTo(int setX, int setY, int setZ, FILE* fp) {
    int fcXa, fcXi, fcYa, fcYi; // Segnale di fine corsa
    int X, cX, uX;
    int Y, cY, uY;
    int Z, cZ, uZ;
    fprintf(fp, "t, setX, posizioneX, comandoX, setY, posizioneY, comandoY,"
            "setZ, posizioneZ, comandoZ\n");

    for(int i = 0; i < 1000; i++) {

        fcXa = RtdacPCI_AD(BaseAddress, FC_ASSEX_AVANTI,1);
        fcXi = RtdacPCI_AD(BaseAddress, FC_ASSEX_INDIETRO,1);
        fcYa = RtdacPCI_AD(BaseAddress, FC_ASSEY_AVANTI,1);
        fcYi = RtdacPCI_AD(BaseAddress, FC_ASSEY_INDIETRO,1);
        if(isUp(fcXa) || isUp(fcXi) || isUp(fcYa) || isUp(fcYi)){
            printf("Fine corsa incontrato lungo il percorso!\n");
            safeValues();
            break;
        }

        X = RtdacPCI_ReadEncoder(BaseAddress, 0);
        Y = RtdacPCI_ReadEncoder(BaseAddress, 1);
        Z = RtdacPCI_ReadEncoder(BaseAddress, 2);
```

```
        cX = xAxisPid((int)(setX*TAC_MM), X);
        uX = (1023 + cX);

        cY = yAxisPid((int)(setY*TAC_MM), Y);
        uY = (1023 + yAxisPid((int)(setY*TAC_MM), Y));

        cZ = zAxisPid((int)(setZ*TAC_MM_Z), Z);
        uZ = (1023 - cZ);

        RtdacPCI_DA(BaseAddress, MOTORE_X, uX);
        RtdacPCI_DA(BaseAddress, MOTORE_Y, uY);
        RtdacPCI_DA(BaseAddress, MOTORE_Z, uZ);

        fprintf( fp, "%d, %d, %d, %d, %d, %d, %d, %d, %d, %d\n",
        i,
        (int)(setX*TAC_MM), X, cX,
        (int)(setY*TAC_MM), Y, cY,
        (int)(setZ*TAC_MM_Z), Z, cZ);
        Sleep(10);
    }
}

/* Movimento lungo una traiettoria circolare */

void moveCircle(int centX, int centY, int radius, float w, FILE *fp){
    int setX, X, cX, uX;
    int setY, Y, cY, uY;
    fprintf(fp, "t, setX, posizioneX, comandoX, setY, posizioneY, comandoY\n");
    for(int i = 0; i < 5000; i++){
        setX = centX + (int)(radius*cos(w*i*TCAMP));
        setY = centY + (int)(radius*sin(w*i*TCAMP));
        X = RtdacPCI_ReadEncoder(BaseAddress, 0);
        Y = RtdacPCI_ReadEncoder(BaseAddress, 1);
        cX = xAxisPid(setX, X);
        cY = yAxisPid(setY, Y);
        uX = cX + 1023;
        uY = cY + 1023;
        RtdacPCI_DA(BaseAddress, MOTORE_X, uX);
        RtdacPCI_DA(BaseAddress, MOTORE_Y, uY);
        fprintf( fp, "%d, %d, %d, %d, %d, %d, %d\n",
        i,
        setX, X, cX,
        setY, Y, cY);
        Sleep(10);
    }
}

/* Startup per gli encoder */
void resetEncoders(){
    RtdacPCI_ResetEncoder(BaseAddress, 0, 1);
    RtdacPCI_ResetEncoder(BaseAddress, 0, 0);
    RtdacPCI_ResetEncoder(BaseAddress, 1, 1);
    RtdacPCI_ResetEncoder(BaseAddress, 1, 0);
    RtdacPCI_ResetEncoder(BaseAddress, 2, 1);
    RtdacPCI_ResetEncoder(BaseAddress, 2, 0);
}
```

Bibliografia

[1] P Bolzern, R Scattolini, and N Schiavoni. *Fondamenti di controlli automatici*. McGraw-Hill, second edition, 2002.

[2] D Chiasson. *Modeling and high-performance control of electric machines*. Wiley-Interscience, 2005.

[3] H Goldstein. *Classical Mechanics*. Addison Wesley, third edition, 2002.

[4] D Raimondo. *Modellizzazione e controllo di un carroponte in tre dimensioni*. 2003.