



UNIVERSITÀ DI TRENTO
DIPARTIMENTO DI INGEGNERIA INDUSTRIALE
LAUREA TRIENNALE IN INGEGNERIA INDUSTRIALE

~ · ~

ANNO ACCADEMICO 2022–2023

Controllo in coppia di un motore elettrico DC

Relatore
Prof. Matteo SAVERIANO

Studente
Stefano TONINI
218722

Sommario

L'obiettivo di questa tesi è progettare un regolatore di tipo PID per un motore in corrente continua, e di controllarne, tramite microcontrollore, la coppia motrice. Cercando di trovare i parametri ottimali per la parte proporzionale, integrale e derivativa. In particolare il lavoro svolto è stato suddiviso in quattro capitoli, che racchiudono le tematiche studiate:

1. Nel primo capitolo, dopo un'introduzione agli azionamenti elettrici, vengono descritte le operazioni necessarie per progettare il regolatore;
2. Il secondo capitolo illustra il funzionamento di un motore DC, descrivendo le equazioni che lo caratterizzano e le tecniche di pilotaggio;
3. Il terzo capitolo descrive il lato software implementato su microcontrollore e sul computer per la fase di identificazione del sistema. Tramite i dati di ingresso e uscita è infatti possibile identificare, sfruttando opportuni algoritmi, la funzione di trasferimento del motore DC. Inoltre si illustra il funzionamento del controllore, in particolare di un regolatore PID;
4. Nel quarto capitolo vengono descritti i componenti hardware utilizzati per la realizzazione del modello fisico reale e si mostrano i risultati ricavati sperimentalmente per la fase di identificazione della funzione di trasferimento del motore e per la fase di taratura dei parametri del regolatore PID.

Indice

Indice	iii
1 Introduzione	1
1.1 Azionamenti elettrici	1
2 Motori DC e Azionamenti	3
2.1 Motori DC	3
2.1.1 Circuito equivalente elettrico	3
2.1.2 Caratterizzazione del carico meccanico	5
2.1.3 Modello dinamico	6
2.2 Pilotaggio ON-OFF	8
2.3 Pilotaggio PWM	8
2.4 Ponte H	9
3 Identificazione e Controllo	11
3.1 Arduino IDE	11
3.1.1 Linguaggio di programmazione	11
3.1.2 Procedura di azionamento	12
3.2 Identificazione	12
3.2.1 MATLAB	13
3.2.2 System Identification Toolbox	14
3.3 Controlli automatici	14
3.4 Regolatore PID	15
3.4.1 Azioni di controllo PID	16

INDICE

3.5 Taratura regolatore PID	17
3.5.1 Taratura software con Simulink	17
3.5.2 Taratura con metodo di Ziegler-Nichols	18
3.5.3 Taratura manuale	19
4 Implementazione hardware e risultati	21
4.1 Arduino UNO	21
4.2 Pololu Dual VNH5019 Motor Driver Shield	23
4.3 Sensore di corrente INA219	24
4.4 Motore DC	25
4.5 Modello fisico assemblato	26
4.6 Risultati identificazione FDT	26
4.7 Risultati taratura software con pidTuner	29
4.8 Risultati taratura manuale	32
4.9 Risultati	34
5 Conclusioni e sviluppi futuri	35
Appendice A Codice Arduino per controllo coppia	37
Appendice B Codice Arduino per controllo velocità	41
Appendice C Codice MATLAB per comunicazione seriale	45
Elenco delle figure	47
Bibliografia	48
Ringraziamenti	50

Capitolo 1

Introduzione

1.1 Azionamenti elettrici

Un azionamento elettrico è un particolare sistema in grado di convertire energia elettrica in energia meccanica. Ogni azionamento è composto da attuatori elettrici (motori) utilizzati per generare il movimento, da un convertitore elettronico di potenza che adatta la potenza elettrica in ingresso prelevata dalla rete alla potenza elettrica in uscita fornita al motore e da un controllore che regola il convertitore elettronico in maniera tale da fornire la corretta alimentazione al motore per un funzionamento ottimale. Questi azionamenti sono utilizzati nei più disparati settori applicativi che comprendono, tra l'altro, impianti industriali, pompe di calore, elettrodomestici, mobilità elettrica, robot industriali, energie rinnovabili (turbine eoliche). Ciascuna di queste applicazioni ha requisiti differenti in termini di potenza installata e caratteristiche di controllo. Il controllo può avvenire in due modi, a catena aperta oppure a catena chiusa. La prima tipologia, a catena aperta, consiste in una regolazione più grossolana nella quale non viene misurata la variabile da controllare. Nel nostro caso necessitiamo di un controllo in catena chiusa, ossia la variabile da controllare va effettivamente misurata tramite un sensore e questa informazione deve essere passata al controllore attraverso una retroazione. L'obiettivo è perciò misurare precisamente la variabile che deve essere controllata, in questo caso la corrente. Questa variabile verrà confrontata con il segnale di riferimento, ossia il valore desiderato di corrente. La differenza tra queste due grandezze, che nel controllo viene chiamato errore deve essere mandato al controllore. Il controllore avrà il compito di fornire al convertitore di potenza le informazioni necessarie riguardanti l'alimentazione ottimale del motore per andare a ridurre la differenza tra segnale di riferimento e la

CAPITOLO 1. INTRODUZIONE

corrente effettivamente misurata. Per effettuare questa operazione è stato realizzato un modello reale composto da un motore DC connesso ad un microcontrollore Arduino che ha permesso di monitorare le variabili di interesse.

Non avendo a disposizione i parametri fisici caratteristici del motore utilizzato, è necessaria una fase di identificazione e di stima della funzione di trasferimento rappresentativa del comportamento dinamico del motore. Conoscendo il comportamento dinamico del motore è possibile ricreare il modello fisico attraverso software e progettare il regolatore andando a simulare i risultati su questo modello rappresentativo della realtà. Successivamente i risultati del regolatore progettato via software possono essere applicati al sistema reale per verificarne la validità. Se la procedura di identificazione viene eseguita correttamente, ovvero il modello simulato del motore approssima bene il comportamento reale, i risultati ottenuti attraverso software dovrebbero coincidere con le prove sperimentali effettuate sul modello reale. Per effettuare questa operazione saranno necessarie conoscenze teoriche, componenti hardware e componenti software che verranno illustrati nei capitoli successivi.

Capitolo 2

Motori DC e Azionamenti

Il capitolo illustra l'azionamento e il principio di funzionamento di un motore DC.

Inoltre viene descritto il modello matematico e le tecniche di pilotaggio.

2.1 Motori DC

In generale i motori elettrici si possono classificare in due categorie, quelli che funzionano con corrente alternata e quelli che funzionano con corrente continua. La seconda tipologia viene utilizzata per questo progetto. Un motore DC è una macchina elettrica in grado di convertire potenza elettrica in potenza meccanica. Esistono diverse tipologie di motori in corrente continua in base al tipo di applicazione necessaria, ma sono tutti composti da una parte rotante detta rotore ed una parte fissa detta statore. Il principio di funzionamento di un motore elettrico è basato sull'interazione di due campi magnetici che si attraggono e respingono a vicenda. Lo statore crea il campo magnetico principale attraverso magneti permanenti oppure con un avvolgimento in cui scorre corrente. Il rotore è un cilindro di materiale ferromagnetico sulla cui superficie sono ricavate delle cave in cui vengono sistemati i conduttori che costituiscono il circuito d'armatura della macchina, è soggetto al campo principale generato dallo statore. Tra i due esiste un sottile strato (a forma di corona) d'aria detto traferro [1].

2.1.1 Circuito equivalente elettrico

Le equazioni che seguono rappresentano il modello matematico del motore, sotto le seguenti ipotesi semplificative:

1. si assume che il circuito magnetico sia lineare;
2. si assume che l'attrito meccanico sia funzione lineare della velocità del motore.

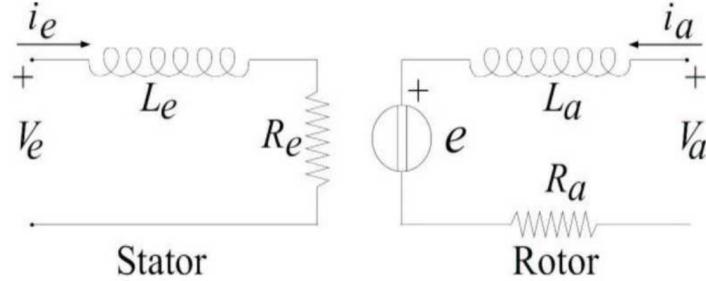


Figura 2.1: Schema elettrico motore DC [2]

In un motore DC, il flusso magnetico ϕ viene generato da avvolgimenti posti sullo statore. Si assumerà che lo statore abbia una sola terminazione polare, caratterizzata da una induttanza L_e associata al relativo avvolgimento e da una resistenza R_e associata alle dispersioni nel conduttore. L'equazione che descrive il circuito elettrico dello statore è la seguente

$$u_e(t) = L_e \frac{di_e}{dt} + R_e i_e \quad (2.1)$$

Come per lo statore, si assume che il rotore abbia una sola terminazione polare caratterizzata da una resistenza di armatura R_a e da un'induttanza di armatura L_a . Inoltre, va considerato nel modello elettrico del rotore l'effetto della forza controelettromotrice $e_a(t)$, che corrisponde ad una differenza di tensione indotta (dallo statore), proporzionale alla velocità di rotazione. L'equazione associata al relativo circuito elettrico è:

$$u_a = R_a i_a + L_a \frac{di_a}{dt} + e_a \quad (2.2)$$

con $u_a(t)$ tensione di armatura e $i_a(t)$ corrente di armatura. Sulla base delle proprietà fisiche del motore e delle relazioni che caratterizzano l'interazione tra le quantità meccaniche e le quantità elettriche del motore, si può dimostrare che le due seguenti relazioni sussistono. Queste relazioni forniscono un'espressione esplicita della forza controelettromotrice e_a e della coppia meccanica T erogata dal motore:

$$e_a = K_e \phi \omega \quad (2.3)$$

$$T = K_t \phi i_a \quad (2.4)$$

con K_e costante elettrica e K_t costante di coppia, due costanti proprie del motore, ed ω velocità angolare.

Basandoci sulle relazioni precedenti è possibile ricavare il legame tra la corrente e la coppia. Dalla equazione 2.4 si evince come la coppia sia linearmente proporzionale alla corrente d'armatura, di conseguenza controllando la corrente i_a , che scorre all'interno del circuito d'armatura, ne stiamo di fatto controllando anche la coppia che è proprio lo scopo di questo elaborato.

2.1.2 Caratterizzazione del carico meccanico

Il comportamento del carico meccanico è praticamente sempre di tipo non lineare, ma solitamente si effettua una implicita o esplicita linearizzazione attorno ad un punto di lavoro in modo da impiegare un modello del carico di tipo lineare.

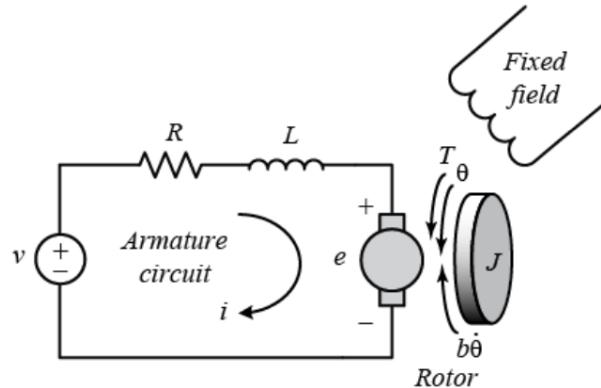


Figura 2.2: Schema meccanico motore DC

Il più generale modello lineare del carico può venire ricavato eguagliando la coppia motrice T alla somma di tre termini:

1. un termine che rappresenta una coppia di carico esercitata sull'asse del motore;
2. un termine proporzionale alla derivata della velocità di rotazione del motore moltiplicata per un momento di inerzia J , pari alla somma del momento di inerzia del motore e di quello del carico, riportato all'asse del motore;

3. un termine proporzionale alla velocità ω di rotazione del motore secondo un coefficiente di attrito B , pari alla somma del coefficiente di attrito del motore e di quello del carico, riportato all'asse del motore.

Si ottiene la seguente equazione per la coppia

$$T = T_L + J \frac{d\omega}{dt} + B\omega \quad (2.5)$$

dove T_L è la coppia esercitata dal carico applicato al motore che nella nostra applicazione non è presente. Tramite queste equazioni è possibile rappresentare il modello dinamico del motore.

2.1.3 Modello dinamico

Utilizzando le equazioni caratteristiche del motore ed uno strumento matematico che prende il nome di trasformata di Laplace è possibile ricreare il modello dinamico del motore[2] con un diagramma a blocchi.

Vengono riepilogate le equazioni descrittive del motore precedentemente discusse

Equazione del circuito elettrico d'armatura

$$u_a = R_a i_a + L_a \frac{di_a}{dt} + e_a \quad (2.6)$$

Equazione della forza controelettromotrice

$$e_a = K_e \phi \omega \quad (2.7)$$

Equazione del circuito di eccitazione

$$u_e = R_e i_e + L_e \frac{di_e}{dt} \quad (2.8)$$

Equazione del carico meccanico

$$T = T_L + J \frac{d\omega}{dt} + B\omega \quad (2.9)$$

Equazione della coppia

$$T = K_t \phi i_a \quad (2.10)$$

Ipotizzando un flusso ϕ costante e una tensione u_a impressa si ottiene la linearità delle equazioni ed è quindi possibile applicare la trasformata di Laplace, grazie a questo strumento le precedenti equazioni integro-differenziali nel dominio del tempo diventano delle relazioni algebriche nella variabile s di Laplace

$$U_a(s) = R_a I_a(s) + L_a s I_a(s) + E_a(s) \quad (2.11)$$

$$E_a(s) = K_e \phi \Omega(s) \quad (2.12)$$

$$T(s) = K_t \phi I_a(s) \quad (2.13)$$

$$T(s) = T_L(s) + J s \Omega(s) + B \Omega(s) \quad (2.14)$$

Utilizzando la teoria degli schemi a blocchi si possono rappresentare le equazioni appena illustrate nel dominio di Laplace attraverso il diagramma a blocchi del motore a corrente continua (Figura 2.3), in cui $K_\omega = K_\tau = K_e \phi$.

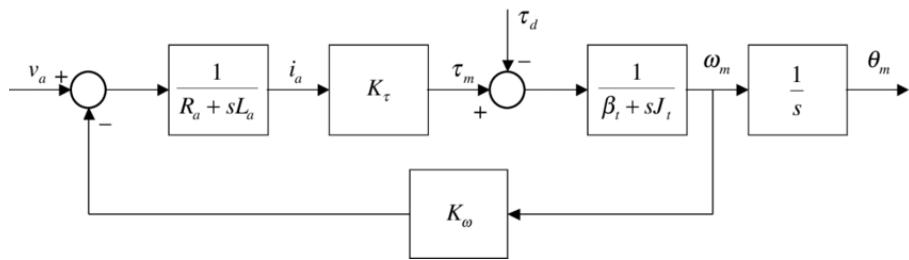


Figura 2.3: Diagramma a blocchi motore DC [3]

Da questo diagramma è possibile ricavare le funzioni di trasferimento scegliendo un ingresso ed un'uscita. Abbiamo bisogno della funzione di trasferimento tra la tensione del circuito d'armatura $U_a(s)$ come ingresso e la corrente d'armatura $I_a(s)$ come uscita. Utilizzando il diagramma a blocchi e l'algebra dei diagrammi a blocchi si ricava

$$\frac{I_a(s)}{U_a(s)} = \frac{\frac{1}{R_a + sL_a}}{1 + \frac{1}{R_a + sL_a} (K_e \phi)^2 \frac{1}{B + sJ}} \quad (2.15)$$

che può essere semplificata in

$$\frac{I_a(s)}{U_a(s)} = \frac{\frac{B+sJ}{(K_e\phi)^2}}{1 + \frac{(R_a+sL_a)(B+sJ)}{(K_e\phi)^2}} \quad (2.16)$$

Questa equazione sarà utile nel capitolo successivo per stimare la funzione di trasferimento del motore, che dovrà essere di questo tipo e di conseguenza avere uno zero e due poli.

Ponendo come ingresso $U_a(s)$ e come uscita la velocità del rotore $\Omega(s)$ ricaviamo la seguente FDT

$$\frac{\Omega(s)}{U_a(s)} = \frac{\frac{1}{R_a+sL_a} K_e\phi \frac{1}{Bs+J}}{1 + \frac{1}{R_a+sL_a} (K_e\phi)^2 \frac{1}{Bs+J}} \quad (2.17)$$

Semplificando si ottiene

$$\frac{\Omega(s)}{U_a(s)} = \frac{\frac{1}{K_e\phi}}{1 + \frac{(R_a+sL_a)(B+sJ)}{(K_e\phi)^2}} \quad (2.18)$$

Queste funzioni di trasferimento sono rappresentative del processo fisico, nel seguente capitolo verrà illustrata la loro importanza al fine di progettare il controllo.

2.2 Pilotaggio ON-OFF

I motori DC possono essere controllati semplicemente tramite un pilotaggio ON-OFF che consente di azionare il motore solo alla massima velocità di rotazione, interruttore ON, oppure fermarlo, interruttore OFF. Questo controllo può essere implementato con un interruttore e con un diodo di ricircolo necessario per evitare danni al resto del circuito, per via della componente induttiva del motore. Il limite di questa tecnica di pilotaggio è evidente, il motore può solamente rimanere fermo oppure girare alla massima velocità. Per questa ragione si utilizza un'altra tecnica, chiamata pilotaggio PWM, per alimentare il motore.

2.3 Pilotaggio PWM

Un segnale Pulse Width Modulation[4], ovvero modulazione di larghezza d'impulso, consiste in un'onda quadra con duty cycle variabile che permette di controllare la potenza assorbita da un carico elettrico (nel nostro caso il motore DC), modulando il duty cycle. Con duty cycle si intende il rapporto tra il tempo in cui l'onda assume valore alto e il periodo T , inverso della frequenza f .

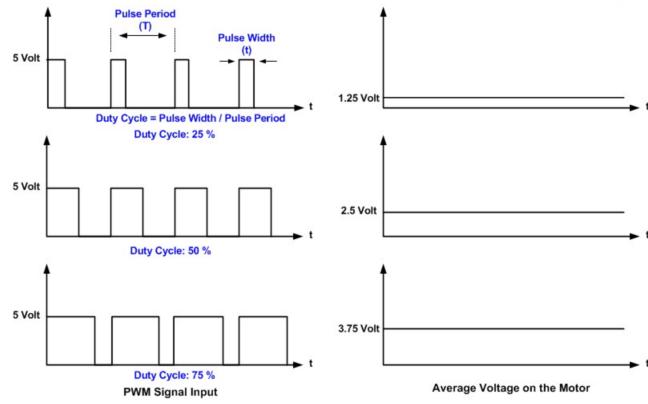


Figura 2.4: Confronto tra segnale PWM e voltaggio medio [5]

Come si nota in Figura 2.4 un duty cycle del 50% corrisponde ad un'onda quadra che assume valore alto per metà del tempo ed un valore basso per la restante metà del tempo. L'onda rettangolare in uscita è il segnale con cui vengono effettivamente pilotati gli switch (MOSFET) del convertitore di potenza. Questa tecnica permette di modificare la velocità di rotazione assicurando un rendimento energetico elevato. Se la frequenza di commutazione del transistor è elevata, la corrente media, a causa del comportamento induttivo del motore, risulta sostanzialmente costante e proporzionale al duty cycle. I principali vantaggi di questo pilotaggio sono le ridotte perdite di potenza e la facilità con cui viene modificato il segnale in uscita, per via della natura digitale del pilotaggio PWM.

2.4 Ponte H

Le tecniche di pilotaggio presentate non permettono la rotazione del motore in entrambi i versi di rotazione. Per consentire al motore di girare nel verso opposto è necessario infatti invertire il segno della corrente che scorre all'interno del motore stesso. Per ottenere ciò si utilizza un circuito che prende il nome di ponte H, costituito da 4 interruttori comandati e da 4 diodi di ricircolo. Il nome deriva dalla somiglianza del circuito alla lettera maiuscola H, dove il motore costituisce il segmento orizzontale ed i quattro transistor i segmenti verticali. Questo tipo di circuito è integrato nella Pololu Dual VNH5019 Motor Driver Shield che verrà illustrata assieme agli altri componenti hardware nel Capitolo 4.

Capitolo 3

Identificazione e Controllo

Nel capitolo si mostrano i programmi utilizzati, fornendo informazioni su di essi, nella procedura per azionare il motore e nel percorso necessario per la taratura dinamica del modello, ossia la determinazione della funzione di trasferimento. Inoltre viene descritto il funzionamento del controllore di tipo lineare PID e le tecniche di taratura

3.1 Arduino IDE

Il primo software utilizzato è un Integrated Development Environment per la piattaforma di open hardware Arduino[6]. Per facilitare la stesura del codice, l'IDE include un editore di testo dotato di alcune particolarità, come il controllo delle parentesi, il syntax highlighting e l'indentazione automatica. L'editor è in grado di compilare e caricare sulla scheda Arduino il programma eseguibile in una sola passata e con un solo click. L'ambiente di sviluppo integrato di Arduino è fornito di una libreria software C/C++, chiamata "Wiring" (dall'omonimo progetto Wiring[7]), la disponibilità della libreria facilita l'implementazione software delle comuni operazioni di input/output. Su questo software sono stati scritti i codici riportati in Appendice A e Appendice B.

3.1.1 Linguaggio di programmazione

I programmi di Arduino vengono chiamati sketch sono scritti in linguaggio derivato dal C/C++. Per creare un file eseguibile l'utente deve definire solamente due funzioni:

1. void setup() – viene invocata una sola volta all'inizio di un programma, si utilizza per le impostazioni iniziali che rimangono invariate durante le successive esecuzioni;
2. void loop() – viene invocata ciclicamente, la cui esecuzione si interrompe solo quando si toglie l'alimentazione alla scheda.

3.1.2 Procedura di azionamento

In questo paragrafo si illustrano le funzioni principali per azionare il motore, mentre il codice completo è riportato in Appendice A. Inizialmente, nella funzione void setup(), è necessario assegnare ai pin della scheda Arduino la corretta funzione. Dopo aver correttamente connesso il motore alla shield e alimentato quest'ultima è possibile azionare il motore attraverso due comandi da posizionare nel void loop() che vengono perciò eseguiti ripetutamente. Bisogna definire in quale verso fare girare il motore, che significa quali transistor del ponte H andare ad aprire. Una volta fornito il senso di rotazione è necessario scegliere il voltaggio con cui alimentare il motore, per fare ciò si utilizza, come discusso nel capitolo precedente, un segnale PWM. Su Arduino IDE si sceglie un numero tra 0 e 255, che corrispondono rispettivamente ad un duty cycle di 0% e del 100%. Grazie a queste istruzioni è possibile azionare il motore. Il programma inoltre deve registrare i valori di input e di output del nostro sistema i quali sono fondamentali per la fase di identificazione.

3.2 Identificazione

L'operazione di stima della funzione di trasferimento viene chiamata taratura dinamica[8]. Si tratta dell'operazione duale della taratura statica in cui viene stimata la relazione, di solito polinomiale, che lega il misurando con l'uscita dello strumento supposta in condizioni statiche. Nel caso statico tale relazione è una funzione che può essere lineare o non lineare, mentre nel caso dinamico è un'equazione differenziale che, per dare vita ad una funzione di trasferimento, ovvero ad una relazione analitica funzione della frequenza, deve essere lineare. Mediante trasformazione di Fourier l'equazione differenziale diventa una relazione algebrica ed è quindi rappresentabile tramite funzione di trasferimento. Quest'ultima è una funzione complessa della frequenza il cui modulo rappresenta come vengono amplificate o attenuate le varie componenti armoniche del segnale e la cui fase rappresenta come esse vengono sfasate. Come obiettivo si cerca di determinare i parametri caratteristici di questa funzione di trasferimento.

In generale ci si trova di fronte a due casi:

1. non si conosce il modello matematico né i parametri del sistema di misura;
2. il modello matematico è noto ma non i suoi parametri.

Per modello matematico si intende una rappresentazione esemplificativa di un sistema reale, in cui vengono schematizzate le sole caratteristiche fisiche di interesse, tramite una serie di leggi che legano gli ingressi, le uscite ed i parametri interni.

In base alla relazione che intercorre tra queste variabili, esistono tre tipologie di modelli:

1. Modello white box: il sistema è una scatola trasparente di cui si conoscono le componenti interne e il loro funzionamento.
2. Modello black box[9]: il sistema è una scatola nera ovvero non è noto a priori né ciò che contiene né come si comporta. È possibile studiarne il comportamento esclusivamente analizzando le risposte che esso produce a fronte delle sollecitazioni che riceve.
3. Modello grey box: il sistema utilizza un approccio intermedio tra modello white box e modello black box

Per identificare la funzione di trasferimento processo fisico è necessario eccitare il sistema fisico e misurare la risposta del sistema. Il modello matematico utilizzato è quello ricavato nel Capitolo 2 grazie al diagramma a blocchi del motore DC. In questa fase viene utilizzato un altro software chiamato MATLAB.

3.2.1 MATLAB

Questo programma offre un ambiente per il calcolo numerico e l'analisi statistica scritto in C, che comprende anche l'omonimo linguaggio di programmazione creato dalla MathWorks. MATLAB[10] consente di implementare algoritmi, graficare funzioni e dati, manipolare matrici, creare interfacce utente, e interfacciarsi con altri programmi. MATLAB viene utilizzato nell'industria e nelle università per via dei suoi numerosi toolbox sviluppati professionalmente, rigorosamente testati e interamente documentati. Tramite comunicazione seriale il software preleva un campione di dati forniti dal microcontrollore e li salva in una tabella utilizzabile per le operazioni successive.

3.2.2 System Identification Toolbox

Queste software mette a disposizione funzioni aggiuntive di MATLAB e in particolare un'applicazione per l'identificazione di sistemi dinamici [11]. Fornendo al programma la tabella dati di cui si parlava prima è possibile andare a stimare la funzione di trasferimento del motore $P(s)$ e ricavarne tutti i parametri. La tabella utilizzata è composta da due colonne, nella prima si trova l'input fornito al motore mentre la seconda racchiude l'output del motore. Nel prossimo paragrafo si mostra l'importanza della funzione $P(s)$ ai fini del controllo.

3.3 Controlli automatici

La teoria del controllo è una branca dell'ingegneria e della matematica che studia la risposta di sistemi dinamici ad un ingresso e come varia attraverso un feedback. L'obiettivo generale è controllare l'output di un sistema, ovvero fare in modo che quest'ultimo segua un segnale di riferimento che può essere costante oppure variabile. Per questa ragione è necessario progettare un controllore che monitori l'output comparandolo con il segnale di riferimento. Nella figura 3.1 si osserva il diagramma a blocchi rappresentativo del sistema con il controllore.

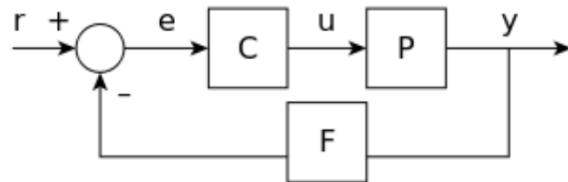


Figura 3.1: Diagramma a blocchi di un sistema single-input e single-output (SISO)

In Figura 3.1 il blocco C rappresenta il controllore, P il processo fisico ed F il feedback misurato con sensori. Usando la trasformata di Laplace è possibile analizzare il sistema ottenendo le seguenti relazioni

$$Y(s) = P(s)U(s) \quad (3.1)$$

$$U(s) = C(s)E(s) \quad (3.2)$$

$$E(s) = R(s) - F(s)Y(s) \quad (3.3)$$

E isolando l'output del sistema $Y(s)$ in termini del segnale di riferimento $R(s)$, si ottiene

$$Y(s) = \frac{P(s)C(s)}{1 + F(s)P(s)C(s)}R(s) = H(s)R(s) \quad (3.4)$$

La funzione $H(s)$ prende il nome di funzione di trasferimento ad anello chiuso. Infatti moltiplicando il segnale di riferimento $R(s)$ per la funzione di trasferimento ad anello chiuso $H(s)$ otteniamo l'output del nostro sistema $Y(s)$, che è proprio la grandezza che si vuole controllare. Nei paragrafi precedenti, nella fase di identificazione, si è visto come determinare la funzione di trasferimento del processo fisico $P(s)$. Dato che spesso il feedback è molto veloce rispetto agli altri blocchi si può approssimare con $F(s) = 1$, perciò rimane solamente da determinare la FDT del controllore $C(s)$. L'obiettivo è progettare il controllore, rappresentato dal blocco C, ossia ricavare l'espressione di questo blocco affinché l'ingresso del sistema ad anello chiuso, segnale di riferimento e l'uscita del sistema, siano il più simili possibile. Per fare ciò è stato implementato un regolatore PID.

3.4 Regolatore PID

Il controllo proporzionale-integrale-derivativo[12], in breve controllo PID, è il sistema di controllo in retroazione più utilizzato nell'industria, in particolare nella versione PI, senza azione derivativa. Grazie a un input che determina il valore attuale, è in grado di reagire a un eventuale errore positivo o negativo. La reazione all'errore può essere regolata e ciò rende questo sistema molto versatile [13].

Il controllore acquisisce in ingresso un valore da un processo e lo confronta con un valore di riferimento. La differenza, il cosiddetto segnale di errore, viene utilizzata per determinare il valore della variabile di uscita del controllore, che è la variabile manipolabile del processo.

Il regolatore PID regola l'uscita in base a:

1. il valore del segnale di errore (azione proporzionale);
2. i valori passati del segnale di errore (azione integrale);
3. quanto velocemente il segnale di errore varia (azione derivativa).

La taratura dei parametri avviene di solito attraverso semplici regole empiriche, come i metodi di Ziegler-Nichols, che risultano in controllori stabilizzanti di buone prestazioni per la maggior parte dei processi. Molto spesso l'azione derivativa viene rimossa, risultando nel comunissimo controllore PI.

3.4.1 Azioni di controllo PID

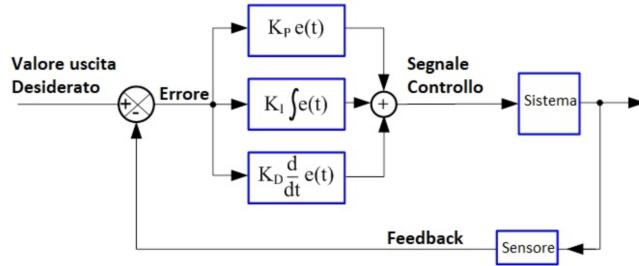


Figura 3.2: Diagramma a blocchi di un controllore PID [14]

Le tre azioni di un PID vengono calcolate separatamente e semplicemente sommate algebricamente:

$$u(t) = u_P(t) + u_I(t) + u_D(t)$$

Azione proporzionale (P)

L'azione proporzionale è ottenuta moltiplicando il segnale d'errore "e" con un'opportuna costante:

$$u_P(t) = K_P e(t)$$

È possibile regolare un processo con un simile controllore, e in alcuni casi semplici riesce anche a stabilizzare processi instabili. Tuttavia, non è possibile garantire che il segnale d'errore "e" converga a zero, visto che un'azione di controllo "u" è possibile solo se "e" è diverso da zero.

Azione integrale (I)

L'azione integrale è proporzionale all'integrale nel tempo del segnale di errore "e", moltiplicato per la costante K_I :

$$u_I(t) = K_I \int_0^t e(\tau) d\tau$$

Questa definizione dell'azione integrale fa sì che il controllore abbia memoria dei valori passati del segnale d'errore; in particolare, il valore dell'azione integrale non è necessariamente nullo se è nullo il segnale d'errore. Questa proprietà dà al PID la capacità di portare il processo esattamente al punto di riferimento richiesto, dove la sola azione proporzionale risulterebbe nulla.

Azione derivativa (D)

Per migliorare le prestazioni del controllore si può aggiungere l’azione derivativa:

$$u_D(t) = K_D \frac{de(t)}{dt}$$

L’azione derivativa cerca di compensare l’errore in base alla sua velocità di cambiamento, senza aspettare che l’errore diventi significativo (azione proporzionale) o che persista per un certo tempo (azione integrale). L’azione derivativa è spesso tralasciata nelle implementazioni dei PID perché li rende troppo sensibili: un PID con azione derivativa, per esempio, subirebbe una brusca variazione nel momento in cui il riferimento venisse cambiato quasi istantaneamente da un valore a un altro, risultando in una derivata di ”e” tendente a infinito, o comunque molto elevata. Se ben tarata e se il processo è abbastanza ”tollerante”, comunque, l’azione derivativa può dare un contributo determinante alle prestazioni del controllore.

3.5 Taratura regolatore PID

L’ultima fase al fine del progetto è proprio la taratura del regolatore, che consiste nel trovare i tre parametri K_P , K_I e K_D che rendano il sistema rapido e robusto nel seguire il segnale di riferimento desiderato. Solitamente al progettista vengono poste delle specifiche asintotiche da rispettare, valori prestabiliti per l’errore a regime e la risposta del sistema a regime e specifiche dinamiche, ovvero valori limite per il tempo di assestamento e la sovraelongazione. Si traducono in limiti da porre sulla frequenza di attraversamento e sul margine di fase. Vengono illustrate tre tecniche di taratura con il comune obiettivo di trovare i parametri ottimali per il regolatore.

3.5.1 Taratura software con Simulink

Grazie alla stima del comportamento dinamico del motore ed alla relativa funzione di trasferimento $P(s)$ è possibile utilizzare un’applicazione che prende il nome di ”PID Tuner[15]” che attraverso algoritmi riesce a stimare con precisione i parametri ottimali del controllore per ottenere prontezza e robustezza. Questa tecnica è di gran lunga la più comoda e precisa ma bisogna prestare attenzione, se la FDT determinata non è accurata, ovvero non descrive precisamente il comportamento dinamico, i parametri

del regolatore stimati dal software non saranno precisi quando verranno applicati al modello reale. Nella seguente figura si può vedere l'interfaccia grafica di questa applicazione

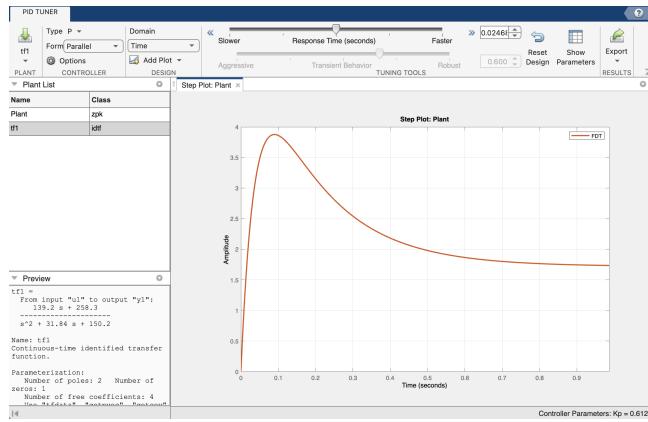


Figura 3.3: Interfaccia grafica PID Tuner

In aggiunta all'interfaccia grafica, in figura 3.3 viene mostrata anche la risposta al gradino, ovvero come il modello stimato risponde quando in ingresso arriva un gradino.

Selezionando la tipologia di controllore desiderato il programma trova in automatico i parametri del regolatore e permette di graficare la risposta al gradino del sistema con annesso il regolatore. Per risposta al gradino si intende l'output del sistema quando viene eccitato con un input che inizialmente è pari a zero e passa istantaneamente ad un altro valore, solitamente unitario. Il software calcola anche alcuni parametri importanti per verificare la bontà del regolatore progettato, tra cui la sovraelongazione, il tempo di assestamento ed il margine di fase. Inoltre studia anche la stabilità del sistema ad anello chiuso.

3.5.2 Taratura con metodo di Ziegler-Nichols

Il metodo di Ziegler-Nichols[16], risalente al 1942, viene apprezzato per la sua semplicità, per le prestazioni che riesce a produrre e per il fatto di non richiedere un modello matematico del processo. In particolare grazie a questa tecnica non è necessaria la fase di identificazione del sistema, visto che i parametri K_P , K_I e K_D del regolatore vengono tarati grazie ad un algoritmo che richiede solamente di misurare l'uscita del sistema, variando l'ingresso, senza essere a conoscenza della funzione di trasferimento del processo fisico $P(s)$. Si tratta di un algoritmo per trovare il cosiddetto "guadagno critico", dal quale si deriveranno gli altri parametri del PID.

1. Il processo viene fatto controllare da un controllore esclusivamente proporzionale, mentre K_I e K_D vengono impostati a zero;
2. Si aumenta gradualmente il guadagno K_P del controllore proporzionale;
3. Il guadagno critico K_u è il valore del guadagno per cui la variabile controllata presenta oscillazioni sostenute, cioè che non spariscono dopo un transitorio: questa è una misura dell'effetto dei ritardi e della dinamica del processo;
4. Si registra il periodo critico P_u delle oscillazioni sostenute;
5. Secondo la seguente tabella, si determinano le costanti per il controllore P, PI o PID, con $\tau_i = \frac{K_P}{K_I}$ e $\tau_d = \frac{K_D}{K_P}$

Tipo	K_p	τ_i	τ_d
P	$0,50K_u$	-	-
PI	$0,45K_u$	$P_u/1,2$	-
PID	$0,60K_u$	$P_u/2$	$P_u/8$

3.5.3 Taratura manuale

L'ultima tecnica utilizzata è manuale e consiste nel variare questi tre parametri nel software in Appendice A e verificare volta per volta la risposta motore e quindi la prontezza nel seguire il setpoint. Per osservare la bontà del regolatore implementato è stato utilizzato un setpoint formato da un gradino, ovvero un istantaneo cambiamento da un valore ad un altro differente ed un segnale sinusoidale. In aggiunta al controllo di coppia è stato progettato anche un controllo di velocità per il motore, il codice completo si trova in Appendice B.

Nel capitolo successivo, dopo una descrizione hardware dei componenti utilizzati, verranno mostrati i risultati del processo di identificazione della funzione di trasferimento $P(s)$ e della fase di taratura con i relativi parametri K_P , K_I e K_D del regolatore.

Capitolo 4

Implementazione hardware e risultati

Nel seguente capitolo si forniscono informazioni hardware del modello analizzato, prendendo in considerazione gli strumenti utilizzati. Inoltre vengono mostrati i risultati ottenuti sperimentalmente.

4.1 Arduino UNO

Il primo componente utilizzato si basa su un circuito stampato che integra un microcontrollore con dei pin connessi alle porte I/O, un regolatore di tensione e un’interfaccia USB che permette la comunicazione con il computer utilizzato per programmare. A questo hardware viene affiancato l’ambiente di sviluppo integrato (IDE) descritto nel capitolo precedente ed utilizzato per la stesura del codice.

Con Arduino si possono realizzare in maniera relativamente rapida e semplice piccoli dispositivi come controllori di luci, automatismi per il controllo della temperatura e dell’umidità e molti altri progetti che utilizzano sensori, attuatori e comunicazione con altri dispositivi. È progettato per interagire direttamente con il mondo esterno tramite un programma residente nella propria memoria interna e mediante l’uso di pin specializzati o configurabili dal programmatore. Gli schemi circuituali sono disponibili come hardware libero e per questo motivo è molto utilizzato nella didattica educativa.

I microcontrollori offrono la possibilità di configurare la maggior parte dei propri pin come input o output mediante specifici registri. Quando un pin è configurato come input, il microcontrollore legge il livello di tensione e restituisce al programma il valore logico alto o basso (HIGH o LOW). Se il pin è configurato come output, il programma in esecuzione può portare il suo livello a 0V (LOW) o a

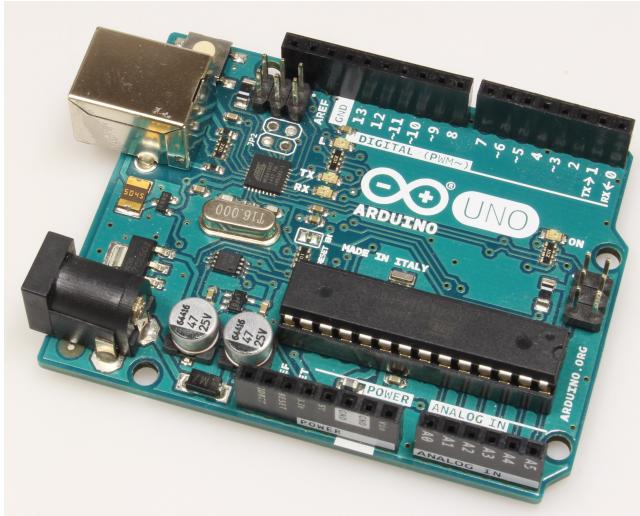


Figura 4.1: Arduino UNO R3

5V o 3.3V (HIGH). La corrente gestita da ciascun pin è limitata e ammonta a 20mA. Se un pin non viene configurato correttamente può avere un comportamento imprevedibile. Tutti i pin di I/O sono collocati sulla parte superiore della scheda mediante connettori femmina, come si vede in Figura 4.1.

La Arduino Uno[17] offre 14 porte per l'I/O digitale, numerati da 0 a 13. La direzione di funzionamento delle porte utilizzata, input o output, deve essere stabilita mediante apposite istruzioni da inserire nello sketch programmato sull'IDE. Per queste elaborato due pin vengono impostati come output e servono per regolare il verso di rotazione del motore, grazie al ponte H. In altri due pin vengono connessi i due canali dell'encoder integrato al motore e vengono impostati come input. Sei dei quattordici canali I/O possono generare segnali Pulse-width modulation (PWM), uno di questi è stato utilizzato per il pilotaggio PWM del motore, descritto nel Capitolo 2.

Sono presenti altri 6 connettori specificamente dedicati a ingressi di segnali analogici che ricevono valori di tensione letti da sensori esterni, fino a un massimo di 5 volt, che sono convertiti in 1024 livelli discreti (da 0 a 1023). In questi connettori viene inserito il sensore di corrente INA219.

Inoltre sono disponibili commercialmente molte schede applicative plug-in, note come "shield", una di queste specifica per il controllo dei motori elettrici è proprio la Pololu Dual VNH5019 Motor Driver Shield utilizzata in questa applicazione.

4.2 Pololu Dual VNH5019 Motor Driver Shield

La Pololu Dual VNH5019 Motor Driver Shield[18] è una scheda di controllo dei motori avanzata progettata per gestire con precisione e affidabilità il movimento di motori DC. La sua progettazione compatta e versatile la rende ideale per una vasta gamma di scenari di utilizzo.



Figura 4.2: Pololu Dual VNH5019 Motor Driver Shield[18]

Al cuore della scheda si trova il driver di motori VNH5019 della STMicroelectronics. Questo driver di potenza dual-channel è progettato per affrontare correnti elevate, supportando fino a 12 A per canale. La presenza di protezioni termiche e di sovraccorrente contribuisce a garantire la sicurezza e la durata del sistema.

La connettività è un aspetto fondamentale della scheda. Con i connettori dedicati per i motori, è possibile collegare facilmente i motori DC o passo-passo desiderati. Questi connettori forniscono anche punti di accesso per l'alimentazione dei motori, la selezione della direzione di rotazione e la regolazione della velocità. L'integrazione di LED di stato può offrire un feedback visivo immediato sul funzionamento dei motori e segnalare eventuali problemi.

Non solo la scheda gestisce l'alimentazione per i motori, ma potrebbe anche includere regolatori di tensione per fornire le tensioni necessarie ai circuiti di controllo o ai dispositivi di livello logico.

In definitiva, la Pololu Dual VNH5019 Motor Driver Shield rappresenta una soluzione compatta e potente per il controllo dei motori in progetti di robotica e automazione. La sua combinazione di funzionalità di controllo preciso, capacità di gestione delle correnti elevate e interfaccia flessibile la

rendono uno strumento prezioso per gli ingegneri e gli hobbisti che cercano di realizzare sistemi di movimento sofisticati e affidabili.

4.3 Sensore di corrente INA219

Questo sensore è un circuito integrato[19] sviluppato da Texas Instruments, progettato per misurare con precisione la corrente e la tensione all'interno di un sistema elettronico. Questo processo avviene mediante il rilevamento della tensione ai capi di una resistenza di shunt posta in serie al carico. La caduta di tensione su questa resistenza è direttamente proporzionale alla corrente che la attraversa. Il sensore amplifica questa tensione e la converte in un segnale digitale, fornendo una misurazione precisa della corrente.



Figura 4.3: Sensore di corrente INA219

Oltre alla misurazione della corrente, il dispositivo è in grado di misurare anche la tensione ai capi del carico. Questo consente di monitorare sia la corrente che la tensione, consentendo calcoli di potenza e altre analisi importanti.

La gamma di misurazione è determinata dalla capacità dell'amplificatore di gestire una differenza di tensione massima di $\pm 320\text{mV}$. Poiché la resistenza di shunt è da 0,1 ohm e ha una tolleranza dell'1%,

la corrente massima che può essere misurata è $\pm 3,2$ Ampere ($V = I * R$, dove V è la tensione, I è la corrente e R è la resistenza).

Per quanto riguarda la risoluzione, l'INA219 integra un convertitore analogico-digitale (ADC) interno a 12 bit per digitalizzare la tensione misurata. Nell'intervallo di $\pm 3,2$ A, la risoluzione dell'ADC è di 0,8mA.

Sono incluse funzionalità di protezione da sovraccorrente e sovratensione, contribuendo così a prevenire danni al sensore e al sistema in situazioni anomale.

La corrente d'armatura I_a , variabile da controllare, scorre nel sensore e grazie a questo viene misurata.

4.4 Motore DC

Nel progetto è stato utilizzato un motore a corrente continua con un encoder integrato. Solamente alcuni dati di targa erano disponibili tra cui la tensione nominale 6V, la velocità nominale pari a 185 Rpm $\pm 1\%$ e inoltre il rapporto di riduzione di 1:46. Il rapporto di riduzione di 1:46 significa



Figura 4.4: Motore DC

che il motore compie 46 giri completi mentre l'uscita dopo la riduzione compie un solo giro. Questo rapporto di riduzione aumenta notevolmente la coppia dell'uscita, rendendolo idoneo per applicazioni in cui è richiesta una forza di torsione superiore. Il motore è compatto e portatile infatti ha un diametro di 25 mm ed lungo 67 mm, queste misure lo rendono utile per applicazioni in cui lo spazio è ridotto. Il motore inoltre dispone di un encoder integrato fondamentale per il controllo di velocità, questo

strumento di misura fornisce 12 impulsi per ogni rotazione dell'albero del motore. Registrando quindi il numero di impulsi fornito dall'encoder è possibile calcolare la velocità del motore in giri al minuto.

4.5 Modello fisico assemblato

In Figura 4.5 è possibile vedere tutti i componenti hardware appena descritti e collegati fra loro.

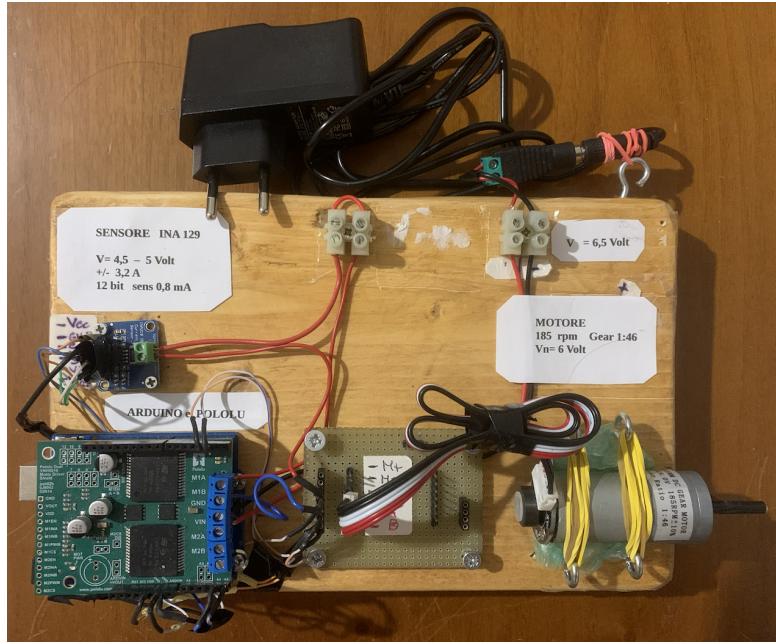


Figura 4.5: Modello reale assemblato

Grazie a questo sistema assemblato ed ai software sono state realizzate le operazioni necessarie al progetto del regolatore discusse nel capitolo precedente. Nei paragrafi successivi vengono mostrati i risultati ricavati.

4.6 Risultati identificazione FDT

In questo paragrafo vengono presentati i risultati della taratura dinamica del motore DC. Lo scopo di questa procedura è ricavare la funzione di trasferimento che permette di ricreare un modello del comportamento dinamico del motore. La funzione di trasferimento è stata determinata attraverso due tipi di modelli:

1. il primo utilizza come input un'onda quadra e come output la velocità del motore in giri al minuto misurata attraverso l'encoder connesso al motore;
2. il secondo utilizza come input un'onda quadra e come output la corrente nel motore in milliampercere misurata grazie al sensore di corrente INA219.

Nella seguente figura si mostra l'ingresso e l'uscita del primo modello utilizzato

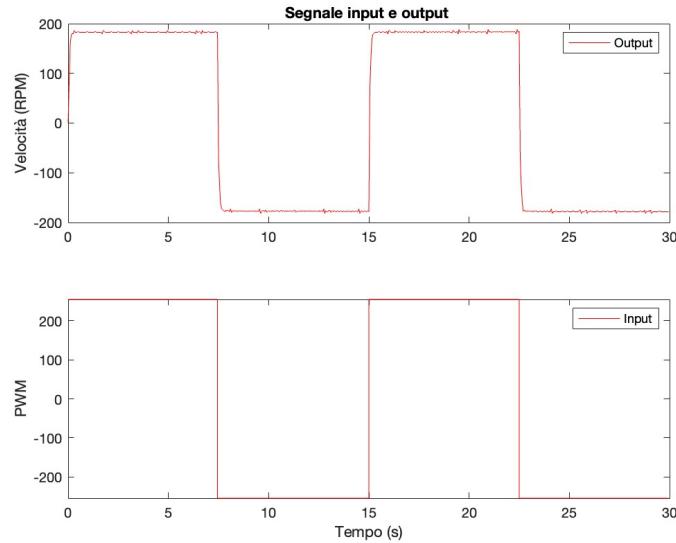


Figura 4.6: Grafico con input un'onda quadra e output la velocità in Rpm

Importando i dati nel System Identification Toolbox è possibile identificare la funzione di trasferimento. L'espressione della funzione di trasferimento ricavata con il primo modello nel dominio di Laplace è la seguente

$$P(s) = \frac{1.238e05}{s^2 + 8086s + 1.752e05} \quad (4.1)$$

Nella Figura 4.7 è possibile osservare la bontà del modello ricavato, ovvero quanto la funzione di trasferimento ricavata riesce ad approssimare l'output misurato.

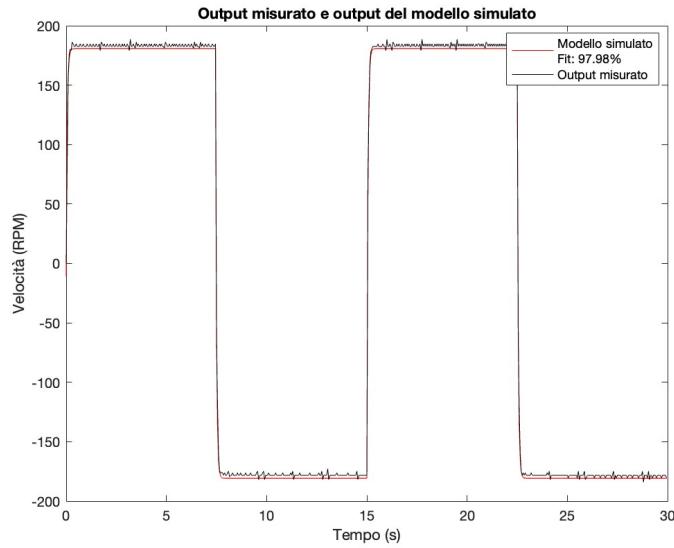


Figura 4.7: Grafico con confronto tra i dati misurati e i dati simulati con il modello trovato in velocità

Nella seguente figura si mostra l'ingresso e l'uscita del secondo modello utilizzato

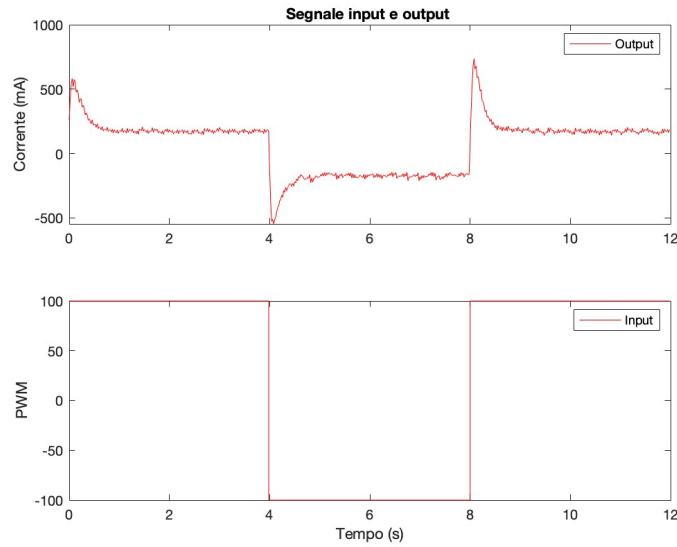


Figura 4.8: Grafico con input un'onda quadra e output la corrente in mA

La funzione di trasferimento ricavata con il secondo modello è composta come segue

$$P(s) = \frac{139.2s + 258.3}{s^2 + 31.84s + 150.2} \quad (4.2)$$

Nella Figura 4.9 è possibile osservare la bontà del modello ricavato, ovvero quanto la funzione di trasferimento ricavata riesce a stimare l'output misurato.

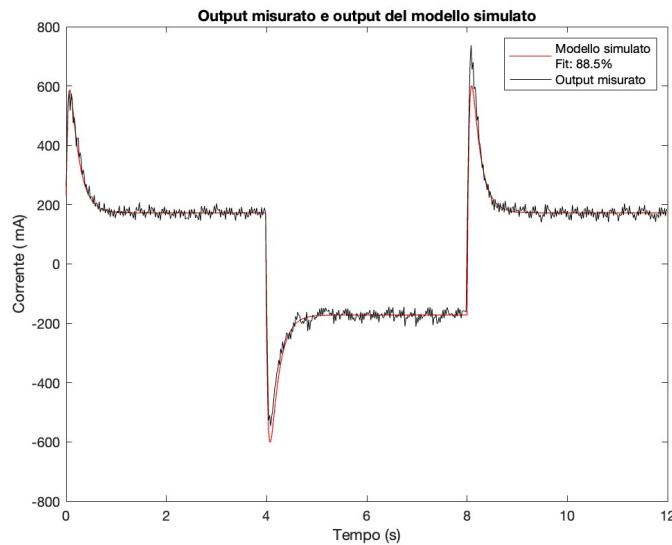


Figura 4.9: Grafico con confronto tra i dati misurati e i dati simulati con il modello trovato in corrente

Delle due funzioni di trasferimento è possibile studiarne la stabilità attraverso il criterio di Nyquist. Queste funzioni di trasferimento identificate stimano il comportamento dinamico del motore e vengono utilizzate per la taratura software dei parametri del regolatore, oggetto del prossimo paragrafo.

4.7 Risultati taratura software con pidTuner

Grazie alla fase di identificazione è possibile utilizzare le funzioni di trasferimento ricavate e procedere con la taratura software utilizzando il "PID Tuner".

Selezionando la tipologia di controllore desiderato il programma trova i parametri ottimali del regolatore in automatico e permette di graficare la risposta al gradino del sistema sia senza regolatore, sia con il regolatore implementato.

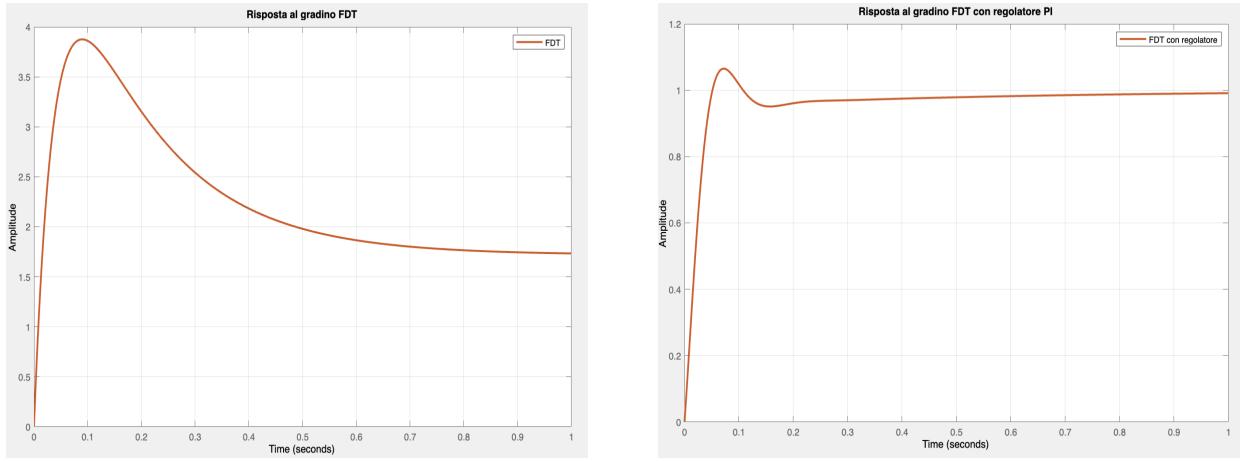


Figura 4.10: Risposta al gradino con modello in corrente

Oltre a fornire i parametri del regolatore il software fornisce anche il tempo di assestamento, ossia dopo quanti secondi si annulla l'errore tra segnale di riferimento e output misurato e la sovraelongazione.

Controller Parameters	
	Tuned
Kp	0.16999
Ki	13.5575
Kd	n/a
Tf	n/a

Performance and Robustness	
	Tuned
Rise time	0.037 seconds
Settling time	0.535 seconds
Overshoot	6.51 %
Peak	1.07
Gain margin	Inf dB @ NaN rad/s
Phase margin	64.8 deg @ 42.5 rad/s
Closed-loop stability	Stable

Figura 4.11: Parametri regolatore tarati con pidTuner

In Figura 4.10 si nota come la sovraelongazione senza il regolatore è di circa 400% mentre con l'implementazione del regolatore viene ridotta al 6.51%. I parametri del regolatore ricavati grazie a questa tecnica di taratura possono essere applicati al modello reale per valutare la risposta effettiva del motore. Il segnale di riferimento utilizzato è composto da un onda quadra e da una sinusoida.

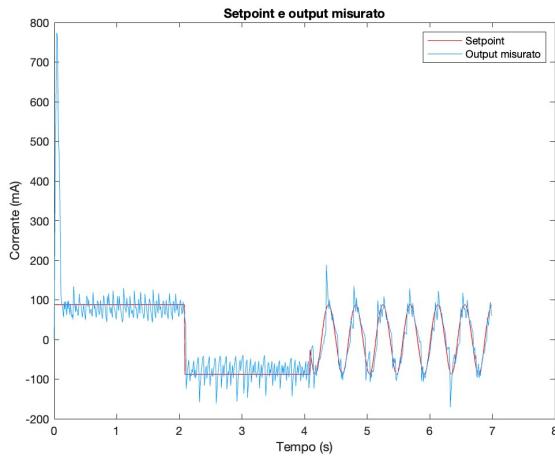


Figura 4.12: Verifica sperimentale controllo corrente con taratura software

In Figura 4.12 viene mostrato come la corrente, misurata grazie al sensore, segue il segnale di riferimento fornito al motore.

Avendo stimato la funzione di trasferimento con il modello per la velocità (Eq. 4.1) è possibile importare anche questa funzione di trasferimento all'interno del pidTuner per ricavare i parametri del PID al fine di controllare la velocità del motore. Come per il controllo della coppia, si riportano i grafici della risposta al gradino del sistema senza regolatore e del sistema con il regolatore.

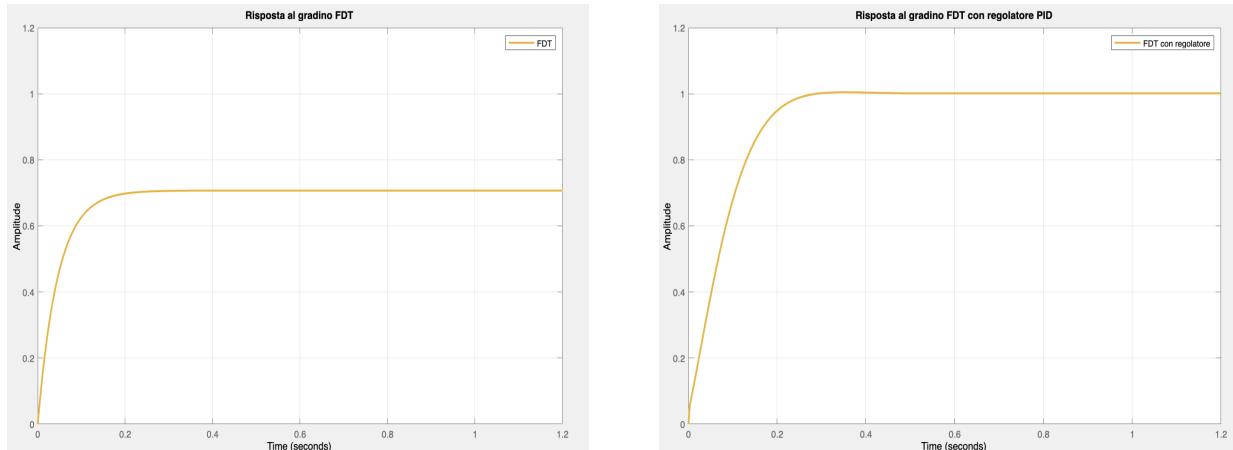


Figura 4.13: Risposta al gradino con pidTuner

In Figura 4.14 si vedono i parametri del regolatore, il tempo di assestamento e la sovraelongazione.

Controller Parameters	
Kp	Tuned 0.49961
Ki	17.9158
Kd	0.00259
Tf	n/a

Performance and Robustness	
Rise time	Tuned 0.159 seconds
Settling time	0.235 seconds
Overshoot	0.451 %
Peak	1
Gain margin	Inf dB @ NaN rad/s
Phase margin	80.1 deg @ 11.5 rad/s
Closed-loop stability	Stable

Figura 4.14: Parametri regolatore tarati con pidTuner

I parametri del regolatore stimati dal PID Tuner vengono inseriti nel codice su Arduino IDE e testati sul modello reale.

4.8 Risultati taratura manuale

La seconda tecnica utilizzata è manuale e consiste nel variare i tre parametri K_P , K_I e K_D nel software in appendice A e verificare volta per volta la risposta motore misurando effettivamente i valori. Si è scelto di trascurare l'azione derivativa per il controllo della corrente visto che rendeva il sistema troppo sensibile. In seguito ad una serie di prove sperimentali i parametri trovati sono i seguenti

$$K_P = 0.46007$$

$$K_I = 0.0001$$

$$K_D = 0.00$$

Per osservare la bontà del regolatore implementato è stato utilizzato il medesimo setpoint implementato in Figura 4.12, formato da un gradino e da un segnale sinusoidale.

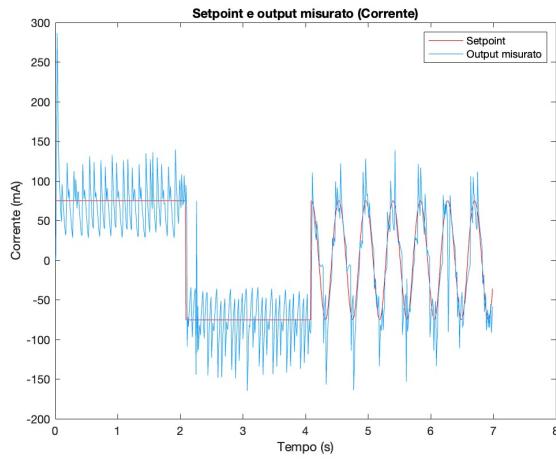


Figura 4.15: Verifica sperimentale controllo corrente con taratura manuale

Inoltre è stato progettato anche un controllo di velocità per il motore, il codice completo si trova in Appendice B. I parametri trovati sono i seguenti

$$K_P = 0.04$$

$$K_I = 0.0008$$

$$K_D = 0.1$$

Come setpoint per verificare il regolatore è stata utilizzata un onda quadra ed una rampa

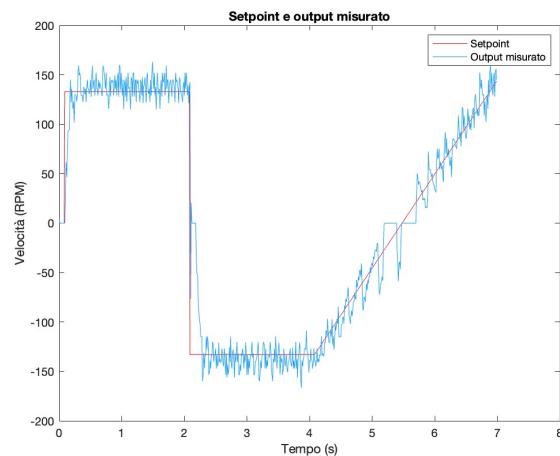


Figura 4.16: Verifica sperimentale controllo velocità con taratura manuale

In Figura 4.16 si nota come il motore segua molto bene il setpoint di velocità desiderato.

4.9 Risultati

Nella seguente tabella vengono riassunti i parametri trovati grazie alle due tecniche di taratura implementate

Parametri regolatore				
	Controllo di coppia		Controllo di velocità	
Tecnica Taratura	Manuale	Software	Manuale	Software
K_P	0.46007	0.16999	0.04	0.49961
K_I	0.0001	13.5575	0.0008	17.9158
K_D	0.00	0.00	0.1	0.00259

Come già preannunciato se la funzione di trasferimento del processo fisico $P(s)$ identificata non è rappresentativa del sistema reale la taratura software attraverso PID tuner è imprecisa. Il programma infatti progetterà il miglior controllore per quella funzione di trasferimento $P(s)$ che non è esattamente quella del sistema fisico. Inoltre la taratura manuale è una tecnica più grossolana che si basa su un approssimazione per tentativi. Per queste ragioni i parametri del regolatore tarato via software si discostano da quelli tarati manualmente sul sistema reale. Confrontando i grafici in Figura 4.12 e in Figura 4.15 che rappresentano le prove effettuate sul modello reale dei regolatori progettati con le due tecniche di taratura si nota come l'ampiezza delle oscillazioni del regolatore tarato via software sia minore del regolatore tarato manualmente, ma entrambi i regolatori seguono abbastanza bene il segnale di riferimento. Questo mostra come il regolatore ottimale non sia unico ma esistano più possibili combinazioni dei parametri che soddisfano una specifica operazione, è compito del progettista, in base alle specifiche poste trovare una terna di parametri che le rispettino.

Capitolo 5

Conclusioni e sviluppi futuri

Lo scopo di questo capitolo è fornire al lettore le informazioni apprese durante questo progetto e discutere i possibili sviluppi futuri. L'elaborato realizzato ha richiesto la conoscenza di diversi ambiti dell'ingegneria tra cui gli azionamenti elettrici, elettronica, informatica e l'automatica. Riguardo l'identificazione del sistema e della funzione di trasferimento si è notata la difficoltà che cresce esponenzialmente all'aumentare dei parametri del modello e richiede strumenti di misura accurati per misurare con precisione l'output e la necessità di fornire input consoni su un ampio range di frequenze. Inoltre, lo studio dettagliato della teoria dei regolatori e l'applicazione delle nozioni acquisite per tarare i parametri del regolatore, è stato necessario per ottenere i risultati trovati. Per i possibili sviluppi futuri tutto l'esperimento è replicabile avendo a disposizione i componenti hardware utilizzati e copiando i codici in appendice è possibile identificare la funzione di trasferimento del motore utilizzato e progettare un regolatore PID sia per controllare la coppia, sia per controllare la velocità. Si può utilizzare anche un motore elettrico in corrente continua con caratteristiche differenti per modellare e progettare il controllo, infatti la trattazione seguita nell'elaborato è del tutto generale ed adattabile.

Appendice A

Codice Arduino per controllo coppia

```
#define ENCA 3
#define ENCB 5
#define PWM 9
#define IN2 4
#define IN1 2
#include <Wire.h>
#include <Adafruit_INA219.h>

Adafruit_INA219 ina219;

float current_setpoint = 80;
float current_error = 0;
float current_integral = 0;
float current_derivative = 0;
float last_current_error = 0;
float current = 0;
float Previouscurrent = 0;
float alpha = 0.80;
long prevT = 0;
int i = 0;

void misure(float x) {
    // Filtro media mobile

    if (i==0) {
        current = ina219.getCurrent_mA();
    }
    i++;
    Previouscurrent = ina219.getCurrent_mA();
    current = x * current + (1.0 - x) * Previouscurrent;
    delay(1);
}

int iteratore = 0;

const float PWMMAX = 255; // massimo valore PWM
```

APPENDICE A. CODICE ARDUINO PER CONTROLLO COPPIA

```
const float KP = 0.46007;      // costante proporzionale del regolatore
const float KI = 13.1257;     // costante integrale del regolatore
const float KD = 0.00;        // costante derivativa del regolatore

void setup() {
    pinMode(ENCA, INPUT);
    pinMode(ENCB, INPUT);
    pinMode(PWM, OUTPUT);
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    if (!ina219.begin()) {
        Serial.println("Failed to find INA219 chip");
        while (1) { delay(10); }
    }
    ina219.setCalibration_32V_1A();
    delay(1000);
    Serial.begin(9600);
}

void loop() {

    // current_setpoint = 0.1 * sin(prevT/1e6);
    if (iteratore++ > 199) {
        iteratore = 0;
    }
    if (iteratore == 199) {
        current_setpoint = - current_setpoint;
    }

    // time difference
    long currT = micros();
    float deltaT = ((float)(currT - prevT)) / (1.0e6);
    prevT = currT;

    // Leggi la corrente dal sensore
    // int adc = analogRead(A0);
    // float Voltage = adc*5/1023.0;
    // current = (Voltage - 2.5)/ 0.185; // sottrae la corrente di offset

    misure(alpha);
    // Calcola l'errore di corrente
    current_error = current_setpoint - current;

    // Calcola l'integrale dell'errore di corrente
    current_integral = current_integral + current_error * deltaT;

    // Limita l'integrale dell'errore di corrente
    if (current_integral > PWMLMAX / KI) {
        current_integral = PWMLMAX / KI;
    } else if (current_integral < -PWMLMAX / KI) {
        current_integral = -PWMLMAX / KI;
    }

    // Calcola la derivata dell'errore di corrente
    current_derivative = (current_error - last_current_error) / deltaT;

    // Calcola il valore di controllo del motore
    float control_value = KP * current_error + KI * current_integral + KD * current_derivative;
```

```
// Limita il valore di controllo del motore
if (control_value > PWMMAX) {
    control_value = PWMMAX;
} else if (control_value < -PWMMAX) {
    control_value = -PWMMAX;
}

// Imposta il segnale PWM
if (control_value >= 0) {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    analogWrite(PWM, control_value);
} else {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(PWM, -control_value);
}

if (current_integral > PWMLMAX / KI) {
    current_integral = PWMLMAX / KI;
} else if (current_integral < -PWMLMAX / KI) {
    current_integral = -PWMLMAX / KI;
}

// Memorizza l'errore di corrente attuale
last_current_error = current_error;

// Stampa i valori sulla seriale
// Serial.print("Current:");
Serial.print(current);
Serial.print(",");
// Serial.print("Control.value:");
Serial.print(control_value);
Serial.print(",");
// Serial.print("Current.error:");
Serial.print(current_error);
Serial.print(",");
// Serial.print("Current.setpoint:");
Serial.println(current_setpoint);
// Serial.print(",");
}
```

APPENDICE A. CODICE ARDUINO PER CONTROLLO COPPIA

Appendice B

Codice Arduino per controllo velocità

```
float kp = 0.04;      // costante proporzionale del regolatore
float ki = 0.0008;    // costante integrale del regolatore
float kd = 0.1;       // costante derivativa del regolatore

TIME VARIABLES
unsigned long t_now;
unsigned long t_prev = 0;
int iteratore = 0;
int pwm;

//*****INTERRUPTS PINS*****
const byte interruptPinA = 2;
const byte interruptPinB = 3;

//*****COUNTER VARIABLE*****
volatile long EncoderCount = 0;

const byte DirPin1 = 2;
const byte DirPin2 = 4;
const byte PWMPin = 9;
int PWMval = 0;

//*****ENCODER PULSES*****
volatile unsigned long count = 0;
unsigned long count_prev = 0;

//*****RPM*****
float Theta_now;
float Theta_prev = 0;
```

APPENDICE B. CODICE ARDUINO PER CONTROLLO VELOCITÀ

```
//RPM_input is the user input RPM (set value)
//RPM_output is the motor output RPM as measured by the encoder
float RPM_output, RPM_input;
int dt; // Period of time used to calculate RPM
float RPM_max = 190; // Setting up a safe maximum RPM for the Motor

//*****MISC VARIABLES*****
#define pi 3.1416
//Maximum motor voltage in clockwise rotation
float Vmax = 6; // Check Motor Datasheet
//Minimum motor voltage
float Vmin = -6;
//Initialize motor input voltage value to zero
float V = 0; // set initial voltage to zero
// Error signals and PID (proportional Integral Derivative) terms
float error_now, error_prev = 0, integ_now, integ_prev = 0;

//*****ISR FUNCTIONS*****
//*****ENCODER A*****
//This is the pinA function, Interrupt Service Routine for pin A
// below is the logic for pinA
void ISR_EncoderA() {
    bool PinB = digitalRead(interruptPinB);
    bool PinA = digitalRead(interruptPinA);
    // detecting if motor rotating clockwise
    // if A is high while B is low, then direction of rotation is clockwise
    if (PinB == LOW) {
        if (PinA == HIGH) {
            EncoderCount++;
        }
    } else {
        if (PinA == LOW) {
            EncoderCount++;
        }
    }
}

//*****ISR FUNCTIONS*****
//*****ENCODER B*****

void ISR_EncoderB() {
    bool PinB = digitalRead(interruptPinA);
    bool PinA = digitalRead(interruptPinB);

    if (PinA == LOW) {
        if (PinB == LOW) {
            EncoderCount--;
        }
    } else {
        if (PinB == HIGH) {
            EncoderCount++;
        }
    }
}
//Funzione per azionare il motore
```

```

void WriteDriverVoltage(float V, float Vmax) {
    int PWMval = int((255 * V) / Vmax);
    if (PWMval > 255) {
        PWMval = 255;
    }
    if (PWMval < -255) {
        PWMval = -255;
    }
    if (PWMval > 0) {
        //setting motor direction
        digitalWrite(DirPin1, HIGH);
        digitalWrite(DirPin2, LOW);
        analogWrite(PWMPin, PWMval);
    } else {
        digitalWrite(DirPin1, LOW);
        digitalWrite(DirPin2, HIGH);
        analogWrite(PWMPin, -PWMval);
    }
}
//*****INTERRUPT SERVICE ROUTINE*****
ISR(TIMER1_COMPA_vect) {
    count++;
}

void setup() {
    //General setup

    Serial.begin(9600);
    pinMode(interruptPinA, INPUT_PULLUP);
    pinMode(interruptPinB, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(interruptPinA), ISR_EncoderA, CHANGE);
    attachInterrupt(digitalPinToInterrupt(interruptPinB), ISR_EncoderB, CHANGE);
    pinMode(DirPin1, OUTPUT);
    pinMode(DirPin2, OUTPUT);

    //*****INTERRUPT SETUP*****
    cli();
    TCCR1A = 0;
    TCCR1B = 0;
    TCNT1 = 0;
    OCR1A = 12499; //Prescaler = 64
    TCCR1B |= (1 << WGM12);
    TCCR1B |= (1 << CS11 | 1 << CS10);
    TIMSK1 |= (1 << OCIE1A);
    sei();
}

void loop() {

    if (count > count_prev) {
        t_now = millis();
        Theta_now = EncoderCount / 552.0;
        dt = (t_now - t_prev);

        if (iteratore++ > 699) {
            iteratore = 0;
        }
        if (iteratore < 10) {
            RPM_input = 0;
        } else if (iteratore < 210) {

```

APPENDICE B. CODICE ARDUINO PER CONTROLLO VELOCITÀ

```
RPM_input = 140;
} else if (iteratore < 410) {
    RPM_input = -140;
}
else {
    RPM_input++;
} // This is the user input RPM

//*****CONDITION OF OPERATION*****
//*****SWITCH MOTOR OFF AFTER A PERIOD OF NO INPUT*****
if (t_now / 1000.0 > 100) {
    RPM_input = 0;
}

//*****ERROR AND PWM CALCULATIONS*****
//*****PID MOTOR TERMS CALCULATIONS*****
RPM_output = (Theta_now - Theta_prev) / (dt / 1000.0) * 60;
error_now = RPM_input - RPM_output;
integ_now = integ_prev + (dt * (error_now + error_prev) / 2);

// Now calculating the Motor voltage or the PID controller output
V = kp * error_now + ki * integ_now + (kd * (error_now - error_prev) / dt);

if (V > Vmax) {
    V = Vmax;
    integ_now = integ_prev;
}
if (V < Vmin) {
    V = Vmin;
    integ_now = integ_prev;
}

WriteDriverVoltage(V, Vmax);
Serial.print(RPM_input);
Serial.print(",");
Serial.print(RPM_output);
Serial.println();
Theta_prev = Theta_now;
count_prev = count;
t_prev = t_now;
integ_prev = integ_now;
error_prev = error_now;
}
}
```

Appendice C

Codice MATLAB per comunicazione seriale

```
close all
clear
clc
%%

% Crea il collegamento con la porta connessa ad Arduino

arduino = serialport('/dev/cu.usbmodem1201', 9600);

num_samples = 700; % Numero di campioni da prelevare
data = zeros(num_samples, 2);

disp('Starting data acquisition ...');

i = 1;
while i <= num_samples
    if arduino.NumBytesAvailable > 0
        data_str = readline(arduino);
        data_str = rtrim(data_str); % Remove leading/trailing whitespace

        if ~isempty(data_str)
            % disp(['Received data string:', data_str]); % Debug print

            data_parts = str2double(strsplit(data_str, ','));
            % disp(['Parsed data parts:', num2str(data_parts)]); % Debug print
            data(i, :) = data_parts;
            i = i + 1;
        end
    end
end

disp('Data acquisition complete.');

clear arduino;
```

APPENDICE C. CODICE MATLAB PER COMUNICAZIONE SERIALE

```
disp('Serial_port_closed.');
%%
figure;

subplot(2, 1, 1);
plot(data(:, 1));
xlabel('Time(s)');
ylabel('Input');
title('Arduino_Data_Analysis');

subplot(2, 1, 2);
plot(data(:, 2));
xlabel('Time(s)');
ylabel('Current(mA)');

%%
u = data(:, 1);
y = data(:, 2);
Ts = 0.01;
data = iddata(y,u,Ts);
% Istruzioni per aprire i tool utilizzati
systemIdentification
pidTuner
```

Elenco delle figure

2.1	Schema elettrico motore DC [2]	4
2.2	Schema meccanico motore DC	5
2.3	Diagramma a blocchi motore DC [3]	7
2.4	Confronto tra segnale PWM e voltaggio medio [5]	9
3.1	Diagramma a blocchi di un sistema single-input e single-output (SISO)	14
3.2	Diagramma a blocchi di un controllore PID [14]	16
3.3	Interfaccia grafica PID Tuner	18
4.1	Arduino UNO R3	22
4.2	Pololu Dual VNH5019 Motor Driver Shield[18]	23
4.3	Sensore di corrente INA219	24
4.4	Motore DC	25
4.5	Modello reale assemblato	26
4.6	Grafico con input un'onda quadra e output la velocità in Rpm	27
4.7	Grafico con confronto tra i dati misurati e i dati simulati con il modello trovato in velocità	28
4.8	Grafico con input un'onda quadra e output la corrente in mA	28
4.9	Grafico con confronto tra i dati misurati e i dati simulati con il modello trovato in corrente	29
4.10	Risposta al gradino con modello in corrente	30
4.11	Parametri regolatore tarati con pidTuner	30
4.12	Verifica sperimentale controllo corrente con taratura software	31
4.13	Risposta al gradino con pidTuner	31
4.14	Parametri regolatore tarati con pidTuner	32

ELENCO DELLE FIGURE

4.15 Verifica sperimentale controllo corrente con taratura manuale	33
4.16 Verifica sperimentale controllo velocità con taratura manuale	33

Bibliografia

- [1] Ned Mohan, Tore M. Undeland e William P. Robbins. *Power Electronics. Converters, Applications and Design*. third. John Wiley e Sons, Inc, 2003.
- [2] Luca Zaccarian. *DC motors: dynamic model and control techniques*. URL: <http://control disp.uniroma2.it/~zack/LabRob/DCmotors.pdf>.
- [3] Basilio Bona. «Metodi di Controllo per Manipolatori Industriali». In: (set. 2023).
- [4] Wikipedia. *Modulazione di larghezza d'impulso*. URL: https://it.wikipedia.org/wiki/Modulazione_di_larghezza_d%27impulso.
- [5] Ronald Willem B. *H-Bridge Microchip PIC Microcontroller PWM Motor Controller*. URL: <http://www.ermicro.com/blog/?p=706>.
- [6] Donald Knuth. *Arduino IDE*. URL: <https://www.arduino.cc/en/software>.
- [7] Wikipedia. *Wiring*. URL: <https://it.wikipedia.org/wiki/Wiring>.
- [8] Mariolino De Cecco. *Taratura Dinamica*. URL: <https://www.miro.ing.unitn.it/1b-riduzione-effetti-disturbo/>.
- [9] Wikipedia. *Modello black box*. URL: https://it.wikipedia.org/wiki/Modello_black_box.
- [10] MathWorks. *MATLAB*. URL: <https://it.mathworks.com/products/matlab.html>.
- [11] MathWorks. *System Identification Toolbox*. URL: <https://it.mathworks.com/products/sysid.html>.
- [12] Wikipedia. *Controllo PID*. URL: https://it.wikipedia.org/wiki/Controllo_PID.
- [13] K.J. Åström e R.M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2010. ISBN: 9781400828739. URL: <https://books.google.it/books?id=cdG9fNqTDS8C>.

BIBLIOGRAFIA

- [14] Elettronica per tutti. *Schema a blocchi controllo PID*. URL: <https://www.ne555.it/controllo-pid-pi-microcontrollore/sistema-pid-schema-a-blocchi/>.
- [15] MathWorks. *PIDTuner*. URL: <https://it.mathworks.com/help/control/ref/pidtuner-app.html>.
- [16] Wikipedia. *Metodo di Ziegler-Nichols*. URL: https://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols_method.
- [17] Arduino. *Datasheet Arduino UNO R3*. URL: <https://docs.arduino.cc/hardware/uno-rev3>.
- [18] Pololu. *Documentazione Pololu*. URL: <https://www.pololu.com/docs/0J49/all>.
- [19] Texas Instruments. *Datasheet INA219*. URL: https://www.ti.com/lit/ds/symlink/ina219.pdf?HQS=dis-mous-null-mousermode-dsf-pf-null-wwe&ts=1694099099618&ref_url=https%253A%252F%252Fwww.mouser.it%252F.

Ringraziamenti

Desidero ringraziare in primis la mia famiglia che mi ha sostenuto psicologicamente ed economicamente durante tutto il mio percorso di studi, contribuendo alla mia carriera scolastica. Ovviamente ringrazio il professor Matteo Saveriano per l'opportunità concessa con questa tesi e per la continua disponibilità. Voglio ringraziare i miei amici Vittorio Pruner, Gianluca Menapace, Lorenzo Santoni, Leonardo Carnessali e Lorenzo Troncon per tutte le esperienze condivise durante gli anni di studio e per il supporto morale ed il legame creatosi durante gli anni. Ritengo necessario menzionare i miei compagni di corso e colleghi Jacopo Dallafior, Mattia Bonvecchio, Alessio Sarcletti e Dimitri Bazza nella con cui ho condiviso tutti i momenti di questo percorso. Grazie ai parenti che hanno saputo interessarsi alla mia carriera dimostrando disponibilità per qualsiasi evenienza. Infine ringrazio tutti i professori del dipartimento di Ingegneria Industriale di Trento per aver condiviso il loro bagaglio culturale e le loro conoscenze apprese con noi studenti.