UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria

Corso di Laurea triennale in Ingegneria
Informatica e dell'automazione

STUDIO E SVILUPPO DI SISTEMI DI CONTROLLO PER MOTORI BRUSHLESS DC

STUDY AND DEVELOPMENT OF CONTROL SYSTEMS FOR DC BRUSHLESS MOTORS

Relatore
Prof. Gianluca Ippoliti

Candidato
Simone Pigliapoco

Correlatore
Prof. Giuseppe Orlando

Anno accademico 2023/2024

"What do you want to be when you grow up?"
"Kind", said the boy.

Charlie Mackesy - The Boy, the Mole, the Fox and the Horse

Ringraziamenti

Giunto al termine del percorso di studi triennale, vorrei spendere alcune parole per ringraziare le persone che hanno contribuito a raggiungere questo primo traguardo.

Innanzitutto vorrei ringraziare il Professor Gianluca Ippoliti, relatore della tesi, per l'aiuto datomi nella realizzazione di questo progetto e del precedente, e per la disponibilità mostrata in questo ultimo anno.

Ai miei amici, quelli di una vita e quelli conosciuti durante il percorso, vanno i miei ringraziamenti per il tempo condiviso assieme, che mi ha permesso di affrontare con leggerezza questa avventura.

Un ringraziamento speciale va ai miei genitori Barbara e Mauro, e a mia sorella Alice, per avermi sempre supportato, non solo in Università ma in tutta la vita, senza aver mai chiesto nulla in cambio se non di fare sempre ciò che mi rendesse felice.

Simone Pigliapoco

Indice

Introduzione

La presente trattazione si propone di studiare e validare sperimentalmente una delle tecniche di controllo più diffuse nel campo dei motori brushless in corrente continua (BLDC), ovvero la six-step commutation. La tesi si divide principalmente in quattro parti.
Nella prima parte viene fornita una panoramica teorica sui motori BLDC con un focus sulla struttura, sulla generazione del moto e sulla componentistica elettronica che consente l'alimentazione delle fasi, approfondendo successivamente il principio di commutazione a sei step con estensione al caso specifico del motore fornito dalla NXP.
Nella seconda ci si sofferma sulla regolazione di velocità del motore BLDC: si presenta l'architettura del sistema di controllo, il modello matematico in forma di funzione di trasferimento, e ci si sofferma successivamente sulla teoria dei controllori PID e, più nello specifico, sui controllori PI che risultano essere ampiamenti utilizzati nel settore dei motori elettrici per la semplicità di implementazione e per la loro robustezza.
Nella terza parte viene affrontata l'applicazione pratica del regolazione della velocità tramite la tecnica di controllo della commutazione a sei step. Si è utilizzato un kit di sviluppo NXP composto di motore BLDC, inverter, sensori a effetto Hall e board con microcontrollore in cui è stato poi caricato il codice C generato a partire da un modello Simulink; tale modello è fornito dal plug-in NXP_MBDToolbox_S32K1xx, e nell'ambito di questo progetto è stato modificato per includere un controllore PI sviluppato tramite script MATLAB, con capacità di gestire saturazione e situazioni di windup integrale.
Nell'ultima parte si sono valutate le prestazioni del sistema attraverso uno strumento di visualizzazione dati in tempo reale, FreeMASTER; tramite la possibilità di salvare in un database i valori campionati della velocità di rotazione, e di calcolare a partire da essi tre indici di prestazione (con l'ausilio di uno script MATLAB), si sono settate le costanti del PI in modo da minimizzare il valore di tali indici. Ovviamente sono forniti anche dei brevi cenni teorici sugli stessi.

L'elaborato è stato sviluppato in ambiente LaTeX.

1 Fondamenti dei motori brushless DC

Nel primo Capitolo di questa tesi, ci si limiterà a presentare la teoria dei motori brushless in corrente continua, ponendo l'attenzione in primis sul funzionamento degli stessi, trattando quindi dei principi fisici che ne inducono il movimento, e successivamente del peculiare sistema elettronico utilizzato per sfruttare al meglio tali principi fisici.

1.1 Introduzione alla famiglia dei motori brushless

Prima di entrare nel dettaglio del motore brushless DC, è necessario presentare una panoramica generale dei motori brushless.
I motori brushless costituiscono una famiglia di motori elettrici la cui peculiarità consiste, come d'altronde suggerisce il nome stesso "brushless", nell'assenza di contatti striscianti; contrariamente ai motori a corrente continua tradizionali che utilizzano spazzole e un commutatore meccanico, i motori brushless utilizzano un commutatore elettronico per gestire la corrente che fluisce nelle bobine dello statore. Il tipico motore brushless è caratterizzato da:

- Uno statore, ossia la componente fissa del motore, sede di una serie di avvolgimenti di filo di rame che generano campi magnetici nel momento in cui vengono attraversate da corrente elettrica. La disposizione di tali avvolgimenti nello statore può variare, ma generalmente vengono disposti in configurazione a stella o a triangolo.
- Un rotore, ossia la componente rotante del motore, costituito da uno o più magneti permanenti (PM rotor). La rotazione del rotore è causata dall'interazione tra i campi magnetici generati dagli avvolgimenti nello statore e i magneti permanenti che compongono il rotore stesso. In base alla posizione reciproca tra rotore e statore, i motori brushless si dividono in:
  - Inrunner, il rotore è situato all'interno dello statore; questo tipo è utilizzato solitamente in applicazioni che richiedono alte velocità di rotazione.
  - Outrunner, lo statore è interno e il rotore ruota intorno ad esso; questo tipo è utilizzato in applicazioni che richiedono una coppia elevata a velocità minori.
- Un controllore elettronico che gestisce la commutazione delle fasi statoriche in base alla posizione del rotore, la quale può essere rilevata tramite sensori di posizioni (sensori a effetto Hall), oppure attraverso algoritmi sensorless.

I motori brushless, rispetto ai motori a collettore, garantiscono prestazioni elevate e durabilità, a discapito però di un maggior costo del dispositivo e di una maggiore complessità del controllo. Volendone elencare i vantaggi, possiamo affermare che questa famiglia di attuatori garantisce: in primis un'elevata efficienza, dal momento che la commutazione elettronica riduce le perdite di energia cui si assiste nel caso di commutazione meccanica, e ciò rende i motori brushless ideali per applicazioni dove il risparmi energetico è cruciale; in secondo

luogo, l'assenza di contatti striscianti elimina i problemi di usura meccanica, garantendo così una maggior durata e affidabilità del dispositivo, nonchè una manutenzione di gran lunga ridotta; inoltre i controlli elettronici avanzati offrono un controllo estremamente preciso di velocità e coppia forniti; infine, l'assenza di contatto tra spazzole e commutatore riduce rumore e vibrazioni del motore.
I motori brushless, quindi, offrono numerosi vantaggi in termini di efficienza, affidabilità, e controllo preciso, rendendoli adatti a una vasta gamma di applicazioni moderne, dalle automobili elettriche alla robotica, dagli elettrodomestici all'industria aerospaziale. Nonostante i costi iniziali e la complessità del controllo, i benefici a lungo termine in termini di prestazioni e manutenzione ridotta rendono questa famiglia di dispositivi una scelta eccellente per molte applicazioni.
La famiglia di motori brushless si divide principalmente in due categorie:

- Brushless DC motors
- Brushless AC motors

La differenza risiede nella forma della forza controelettromotrice (back EMF) indotta dai magneti permanenti di rotore, negli avvolgimenti di statore: nel DC, di cui si parlerà in questa trattazione, l'andamento di tale back EMF è trapezoidale, mentre nell'AC sinusoidale.

1.2 BLDC

Il motore brushless DC, conosciuto anche con il nome abbreviato BLDC o BLDCM (dall'inglese "BrushLess Direct Current Motor"), è un motore elettrico sincrono che opera in corrente continua. Il vantaggio rispetto a un motore della stessa categoria ma in corrente alternata risiede appunto nella alimentazione: l'alimentazione in corrente continua è di gran lunga più agevole da fornire e non necessita obbligatoriamente di circuiti digitali. Tuttavia l'alimentazione in continua non consente di avere coppia costante: si verificano delle cadute di coppia a causa della presenza di fasi ohmico-induttive che vengono collegate e scollegate dal circuito di commutazione; tale fenomeno, invece, non si presenta nel caso di motori brushless AC.

Questa trattazione è volta all'implementazione di un controllore PI per il controllo della velocità di un motore BLDC reale avente tre fasi nello statore e due coppie di poli magnetici nel rotore, pertanto nelle prossime sezioni, in cui verranno affrontati i temi di funzionamento generale e di commutazione in six-step, si farà sempre riferimento a "BLDC two pole-pair three phase motors". Sia il motore reale che il modello che verrà utilizzato in questa tesi sono "inrunner", ma questa specifica non risulta di alcuna utilità. In Figura 1.1 sono chiaramente visibili i tre avvolgimenti statorici, denotati con le lettere A - B - C, e il rotore a magneti permanenti con due coppie di poli Nord - Sud.

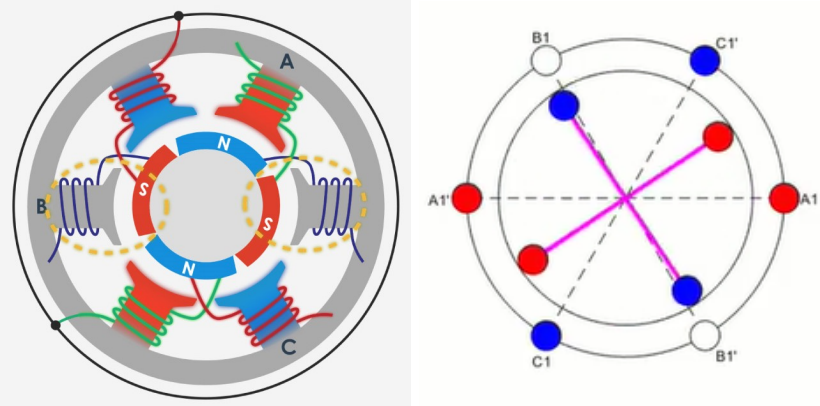


Figure 1.1: *Sezione di un BLDC two pole-pair three phase motor*

1.2.1 Principio di funzionamento

Un motore BLDC segue lo stesso principio di funzionamento di tutti i motori brushless, che risulta facile e intuitivo. Un cavo di rame percorso da corrente elettrica genera un campo magnetico; se poi il cavo di rame è avvolto più volte attorno a un materiale ferromagnetico, il campo magnetico generato dalla bobina polarizza il materiale ferromagnetico rendendolo a sua volta un magnete; questo è ciò che accade nello statore, dove gli avvolgimenti che si trovano attorno alle espansioni statoriche vengono "accese" (fatte attraversare da corrente elettrica) e spente in modo tale da polarizzare e neutralizzare le stesse espansioni. Essendo il rotore composto da magneti permanenti, polarizzare un'espansione statorica significa far sì che il rotore si allinei naturalmente ad essa, generando quindi moto.

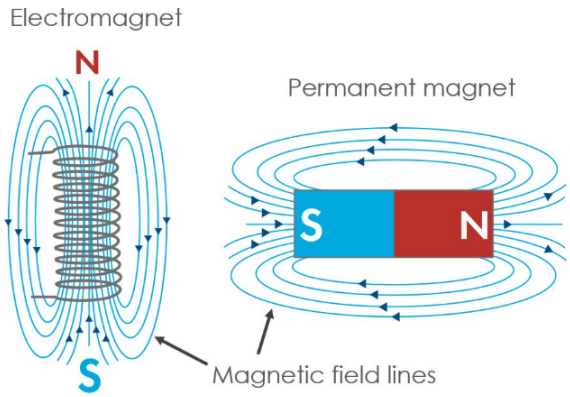


Figure 1.2: *Elettromagneti e magneti permanenti*

In un motore con tre fasi e due coppie di poli magnetici, vengono accese due fasi alla volta, mentre la terza è scollegata: in Figura 1.1 la fase A è accesa e la corrente al suo interno scorre in modo tale che le due espansioni statoriche relative siano polarizzate N, mentre la fase C è attraversata da corrente in modo che le due espansioni siano polarizzate S. La fase B, invece, è scollegata per merito del circuito di commutazione, pertanto le sue espansioni non sono polarizzate. I poli N e S dello statore attirano, rispettivamente, i poli S e N del rotore

a magneti permanenti, mettendo così in rotazione il rotore stesso; accendendo e spegnendo opportunamente le fasi in base alla posizione del rotore si è in grado quindi di generare un moto fluido, con velocità e coppia desiderate. L'operazione di accensione e spegnimento delle fasi viene effettuato dal circuito di commutazione, di cui si parlerà nella Sottosezione 1.2.2.

1.2.2 Circuito elettronico di commutazione

La commutazione elettronica è il processo di gestione della corrente nelle bobine dello statore per creare un campo magnetico rotante; nei motori BLDC è gestita da un controller elettronico che determina la sequenza di alimentazione degli avvolgimenti statorici. Questo processo di commutazione è sincronizzato con la posizione del rotore, che viene rilevata da sensori di posizione, ad esempio sensori a effetto Hall (come nel caso del dispositivo reale oggetto di questa trattazione) ed encoder di rotazione, oppure attraverso algoritmi sensorless che misurano la "back EMF", ossia la forza controelettromotrice. Il tipico circuito di commutazione utilizzato da motori brushless trifase è un inverter trifase come quello in Figura 1.3, che converte la potenza in continua fornita dall'alimentatore in tre correnti di fase.

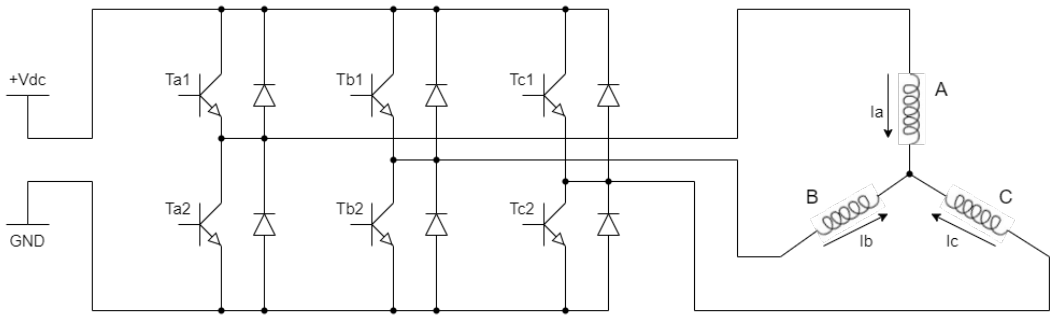


Figure 1.3: *Circuito di commutazione di un BLDC*

Attraverso la lettura o stima della posizione del rotore, il microcontrollore decide in che modo aprire e chiudere i transistor dell'inverter. Intuitivamente, in figura, la fase A è collegata ai rami dei transistor Ta1 e Ta2 e a quello dei rispettivi diodi di flyback, e lo stesso vale per la fase B e la fase C con i rispettivi transistor. Ad ogni commutazione viene attivato un solo transistor del settore superiore (quindi uno solo tra Ta1, Tb1 e Tc1), e uno solo del settore inferiore (Ta2, Tb2 o Tc2); la scelta dei due transistor determina le due fasi su tre che verranno attraversate da corrente, e che direzione avrà la corrente al loro interno.
Se ad esempio si decide di attivare Ta1 e Tc2, il percorso della corrente sarà equivalente a quello in Figura 1.4, partendo da +Vdc e terminando al ground GND. Pertanto, osservando le direzioni delle correnti Ia, Ib, Ic, si avrà:

- Ia positiva, quindi fase A alimentata positivamente
- Ib floating, quindi fase B non alimentata
- Ic negativa, quindi fase C alimentata negativamente

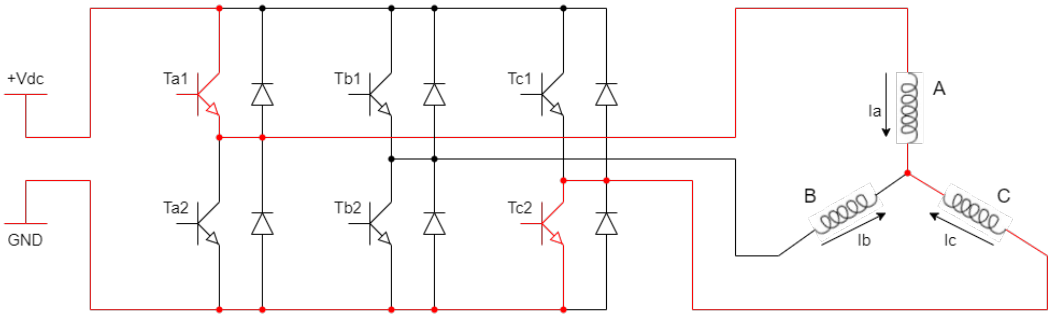


Figure 1.4: *Passaggi di corrente nell'inverter e nelle fasi del motore*

L'effetto prodotto è quello presentato in Figura 1.1: l'alimentazione positiva della fase A costringe le espansioni statoriche A1 e A1' a comportarsi come poli N, mentre l'alimentazione negativa della fase C costringe C1 e C1' a comportarsi come poli S. Questo implica che i poli N del magnete permanente vengano attirati verso le due espansioni di C e respinti da quelle di A, mentre per i poli S del magnete accade l'opposto. Per far sì che il motore ruoti in senso antiorario basterà, una volta allineati i poli N e S del rotore rispettivamente con quelli S e N dello statore, che il microcontrollore spenga Tc2 e accenda al suo posto Tb2.

La commutazione può essere realizzata in vari modi, ma le modalità più conosciute sono la "six-step commutation" (commutazione a sei passi) e la "Field-Oriented Control" (FOC, controllo orientato al campo). Nel merito di questa trattazione ci soffermeremo esclusivamente sulla commutazione a sei passi, di cui si parlerà nel Capitolo 2.

2 Six-step commutation

In questo Capitolo è fornita una dettagliata presentazione per quanto riguarda la commutazione trapezoidale (detta anche six-step) e una spiegazione di come questa viene gestita attraverso un peculiare tipo di sensori, denominati sensori a effetto Hall. Dal momento che, per semplicità, la six-step commutation viene spiegata nel caso di rotore a magnete permanente singolo, l'adattamento al caso di due coppie di poli magnetici viene brevemente spiegato nella Sezione 2.3.

2.1 Fondamenti della six-step commutation

Un motore BLDC, quando è in funzione, genera una forza elettromotrice che si oppone alla variazione di corrente che la induce; tale forza, chiamata "back EMF", può avere una forma trapezoidale o sinusoidale. Per ottimizzare le prestazioni, la corrente che viene passata alle fasi del motore dovrebbe corrispondere alla forma dell'onda della back EMF. Per come sono progettati, la teoria afferma che la famiglia dei motori BLDC produca una back EMF trapezoidale; tuttavia, le induttanze del motore smussano la forma d'onda rendendola leggermente più sinusoidale. Per questo motivo le tecniche di commutazione che possono essere utilizzate su questi dispositivi possono essere:

- Trapezoidali
- Sinusoidali

In questa trattazione ci si soffermerà solo ed esclusivamente sulla tecnica di commutazione trapezoidale.

La commutazione trapezoidale, nota anche come "six-step commutation", è una tecnica di controllo comunemente utilizzata nei motori BLDC: tale tecnica implica la commutazione sequenziale delle fasi del motore per creare un campo magnetico rotante che induce la rotazione del rotore. Il nome "a sei step" nasce dal fatto che l'inverter trifase che controlla il passaggio di corrente negli avvolgimenti può assumere solamente sei configurazioni in base a quali sono i transistor attivi. Facendo riferimento alla Figura 1.3, le configurazioni sono le seguenti.

1. Ta1 e Tb2 ON, gli altri OFF
2. Ta1 e Tc2 ON, gli altri OFF
3. Tb1 e Ta2 ON, gli altri OFF
4. Tb1 e Tc2 ON, gli altri OFF
5. Tc1 e Ta2 ON, gli altri OFF
6. Tc1 e Tb2 ON, gli altri OFF

Come si nota, e come già anticipato sopra, in ogni settore (superiore indicato con il numero 1, inferiore indicato con il numero 2) è attivo un unico transistor: questo consente di attivare solo due fasi nello stesso momento e di guidare la corrente senza situazioni di "imbarazzo" da +Vdc al ground.

La commutazione trapezoidale risulta ampiamente diffusa per la sua semplicità, il basso costo in termine di tempistiche di implementazione e di potenza richiesta dall'algoritmo di controllo, e l'elevata affidabilità. Per questi motivi è di comune utilizzo nelle applicazioni dove è richiesta alta velocità, oppure coppia elevata all'avvio. Tuttavia, questa tecnica soffre di problemi di "ripple di coppia", cioè di piccole cadute di coppia durante la commutazione delle fasi; gli altri svantaggi risiedono nella produzione di rumore elettrico e acustico.

Per spiegare nel modo più semplice e chiaro possibile il funzionamento della six-step commutation, è necessario considerare un motore BLDC a tre fasi con una sola coppia di poli magnetici nel rotore. Questo tipo di motore è schematizzato come di seguito.

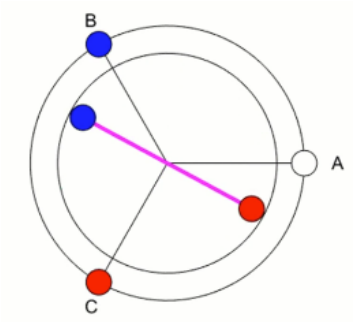


Figure 2.1: *Schema di BLDC trifase con rotore a magnete singolo*

La derivazione al caso di due coppie di poli magnetici verrà effettuata nella sezione 2.3.

La commutazione trapezoidale avviene in sei step: ogni fase del motore viene alimentata da corrente continua e viene elettricamente commutata ogni 60 gradi di rotazione, per un totale di 360 gradi. Il risultato è un'onda a gradini per ogni corrente di fase, con una durata di 60 gradi per ogni gradino. Pur essendo sfasate di 120 gradi l'una dall'altra, le correnti negli avvolgimenti seguono lo stesso principio:

1. Da livello nullo, la corrente viene portata a livello positivo e la si mantiene per 60 gradi.
2. Si mantiene a livello positivo anche per i successivi 60 gradi.
3. Dopo 120 gradi di livello positivo, la fase viene scollegata e la corrente viene lasciata "fluttuare". Per semplicità, si dirà che viene portata a livello nullo e mantenuta tale per i seguenti 60 gradi.
4. Da livello nullo, la corrente viene portata a livello negativo e la si mantiene per 60 gradi.
5. Si mantiene a livello negativo anche per i successivi 60 gradi.

6. Dopo 120 gradi di livello negativo, la corrente viene portata nuovamente a livello nullo e mantenuta per i seguenti 60 gradi.

Questo ciclo si ripete ogni 360 gradi per ogni corrente di fase, e viene mostrato in Figura 2.2. Possono essere effettuate alcune considerazioni.

- Facendo riferimento al nodo tra le fasi statoriche visibile in Figura 1.3, le tre correnti Ia, Ib, Ic rispettano la prima Legge di Kirchhoff; risulta infatti vera l'espressione $Ia + Ib + Ic = 0$ ad ogni istante di tempo.
- Facendo riferimento alla Figura 2.2 si può notare una corrispondenza, per ogni fase, tra grafico della corrente e della rispettiva back EMF.
- In teoria, si cerca di produrre la massima coppia possibile nel motore BLDC energizzando la corretta doppietta di fasi, in modo da ottenere coppia costante. In pratica, questo non è possibile dal momento che non è possibile cambiare istantaneamente la corrente di fase da un valore basso a un valore alto: ci sono periodi di salita e discesa della corrente elettrica che generano ripple nella corrente stessa al momento della commutazione. Essendo la coppia direttamente proporzionale alla corrente, questi ripple si verificano anche nella coppia prodotta.

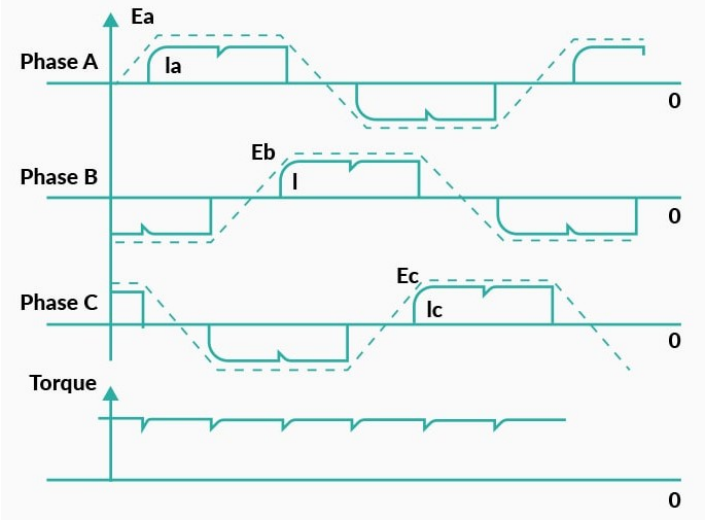


Figure 2.2: *Correnti di fase e coppia generata al variare della posizione angolare del motore*

Energizzando due fasi alla volta seguendo la routine presentata sopra, il campo magnetico generato dallo statore commuta costringendo il rotore ad allinearsi con esso. Avendo sei configurazioni di attivazione delle fasi, sei saranno anche le posizioni a cui il rotore viene "costretto" ad allinearsi. In Figura 2.3 l'angolo del rotore è misurato rispetto all'asse orizzontale, e sono visibili i sei allineamenti, ognuno spostato di 60 gradi rispetto al precedente.

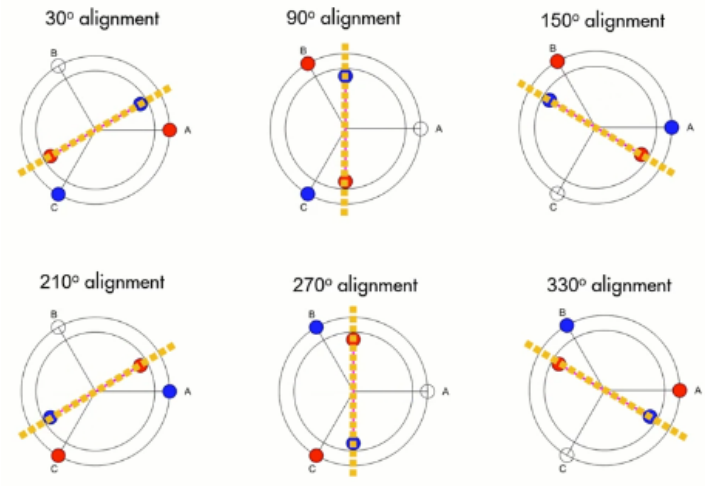


Figure 2.3: *I sei allineamenti generati dalle sei differenti alimentazioni delle fasi*

I transistor del circuito di commutazione vengono attivati e disattivati in sequenza per effettuare la commutazione. Come visto in precedenza, ogni settore del circuito di commutazione (superiore e inferiore) ha un solo transistor che conduce alla volta, con i cambi effettuati con un certo ritardo per evitare danni al sistema. Attraverso i transistor attivati, la corrente viene inviata alle fasi del motore in relazione alla posizione del rotore: risulta quindi di vitale importanza conoscere la posizione angolare del motore in ogni istante per poter implementare efficientemente lo schema di controllo. Questa operazione può essere effettuata tramite sensori (sensori a effetto Hall oppure encoders) oppure tramite algoritmi sensorless di stima della back EMF. Dal momento che in questa trattazione si parlerà di uno specifico motore che utilizza sensori Hall, ci concentreremo solo ed esclusivamente sulla tecnica di rilevazione della posizione tramite questo tipo di dispositivi.

2.2 Rilevazione della posizione tramite sensori a effetto Hall

I sensori a effetto Hall misurano la polarità di campo magnetico, pertantro nel contesto dei motori BLDC sono in grado di riconoscere la posizione del magnete permanente del rotore; l'uscita del sensore è un segnale che varia tra valore alto e valore basso in relazione a quale polo il sensore "vede" davanti a sè. Per conoscere con certezza la posizione del motore, è necessario posizionare i sensori Hall in punti specifici: un singolo sensore può dividere l'intero ciclo elettrico in due parti (una in cui restituisce un segnale di una certa polarità, e una in cui restituisce un segnale di polarità opposta), quindi, per un motore trifase, posizionare tre sensori Hall a distanza di 120 gradi l'uno dall'altro garantisce il riconoscimento di tutte le sei posizioni del ciclo.

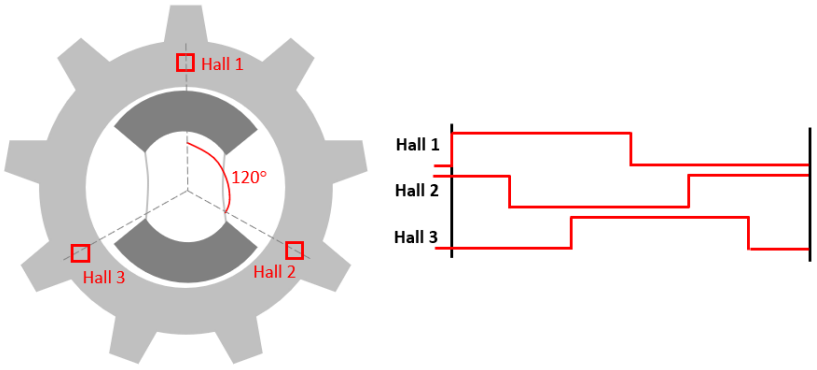
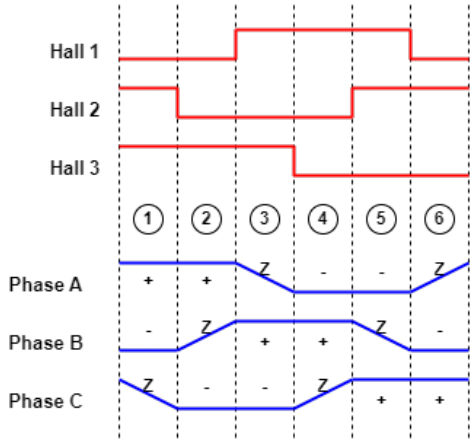


Figure 2.4: *Posizionamento sensori Hall e loro output*

I tre sensori, eseguendo contemporaneamente l'operazione di rilevazione del campo magnetico e di generazione di un segnale avente una certa polarità, producono sei stati, ognuno dei quali corrisponde a uno dei sei settori della commutazione. A ogni combinazione dei segnali output dei tre sensori a effetto Hall corrisponde una modalità di alimentazione delle fasi, come mostrato in Figura 2.5. Lo stato "Z" corrisponde all'assenza di alimentazione (quando contrassegnata da "Z" la fase in questione è quella che non viene alimentata).



Hall 1	Hall 2	Hall 3	Phase A	Phase B	Phase C
0	0	0	Invalid		
0	0	1	+	Z	-
0	1	0	Z	-	+
0	1	1	+	-	Z
1	0	0	-	+	Z
1	0	1	Z	+	-
1	1	0	-	Z	+
1	1	1	Invalid		

Figure 2.5: *Relazione tra l'output dei sensori Hall e l'alimentazione delle fasi*

La tabella di verità mostra lo stato di ogni sensore in relazione alla posizione del rotore: in base al valore di ogni sensore per ogni step, il controllore sa quali fasi devono essere attivate: il risultato è una commutazione continua e fluida. Grazie ai sensori a effetto Hall, la posizione del motore è definita chiaramente anche quando il motore gira molto lentamente o è fermo.

2.3 Derivazione della six-step commutation nel caso di rotore con due coppie di poli

Le macchine BLDC possono avere rotore con più di una coppia di poli magnetici, come nel caso del motore reale di cui si parlerà in questa trattazione, e ciò richiede che la commutazione

venga effettuata più spesso. Supponendo un motore brushless DC trifase con due coppie di poli magnetici, notiamo come nell'ambito di una sola rotazione meccanica avvengano ben due rotazioni elettriche, al contrario di quanto accade nel caso di un singolo magnete permanente nel rotore; ciò si traduce in una commutazione che avviene non più ogni 60 gradi meccanici, bensì ogni 30. Si assiste quindi a due cicli identici di commutazione delle fasi per ogni rotazione completa del motore.

3 Controllo di velocità per motori BLDC che sfruttano la six-step commutation

In questo Capitolo è presentata tutta la teoria necessaria a comprendere al meglio lo schema di controllo per motori BLDC che verrà utilizzato nell'applicazione reale di cui si parlerà nei prossimi capitoli: vengono quindi presentati il diagramma del controllo, la modellazione matematica di un generico motore BLDC, e infine dei cenni sui controllori PID e, più in particolare, sui controllori PI che risultano i più utilizzati nella applicazioni per motori elettrici.

3.1 Schema di controllo six-step per BLDC

Esistono vari schemi di controllo cui far riferimento per l'implementazione di algoritmi che controllino la velocità di un motore BLDC: alcuni hanno un controllore della velocità che restituisce come segnale di controllo il riferimento di corrente, e poi in cascata hanno un altro controllore di corrente che fornisce il duty cycle; altri hanno un unico controllore di velocità che fornisce come output la corrente. Il più inusuale è quello che verrà sfruttato poi per il controllo del motore reale utilizzato in questo progetto, ossia uno schema di controllo dotato di un unico controllore (che come si vedrà successivamente è un controllore PI) della velocità che restituisce, come sforzo di controllo, un segnale di voltaggio che tramite alcuni semplici calcoli viene trasformato nel duty cycle da inviare al modulatore PWM. In Figura 3.1 è presentato il diagramma del controllo.

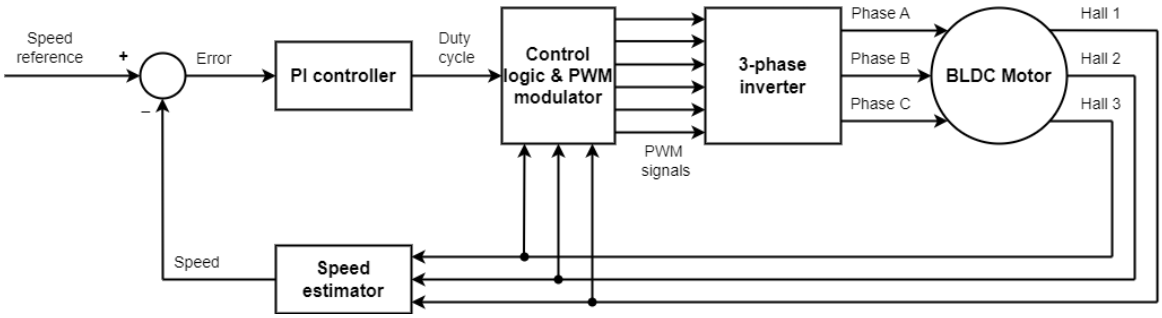


Figure 3.1: *Diagramma a blocchi del controllo*

Al controllore PI, il più utilizzato nei motori elettrici, viene passato in ingresso la differenza tra la velocità di riferimento, ossia quella che si vuole che il motore mantenga, e quella misurata, per poi restituire in uscita come sforzo di controllo un duty cycle. Tale duty cycle costituisce uno dei segnali di ingresso del modulatore PWM che, sfruttando le informazioni provenienti dai sensori Hall (opportunamente tradotte dalla logica di controllo) è in grado di generare 6 segnali PWM, uno per ogni transistor dell'inverter trifase, il cui scopo è quello di aprire e chiudere gli stessi con le tempistiche adeguate. L'apertura e chiusura sequenziale dei transistor dell'inverter fa sì che le fasi siano alimentate correttamente e che quindi il

motore possa girare; successivamente la posizione del motore è letta nuovamente dai sensori a effetto Hall i cui segnali andranno al blocco che implementa la logica di controllo (come già anticipato) e a uno stimatore che, attraverso essi, fornirà la misura reale della velocità tenuta.

3.2 Modellazione matematica del BLDC

Una macchina brushless in corrente continua si modella seguendo lo stesso principio di tutte le macchine in corrente continua. Il circuito equivalente del tipico motore DC è mostrato in Figura 3.2: esso si compone di un circuito elettrico su cui viene adattato un sistema meccanico.

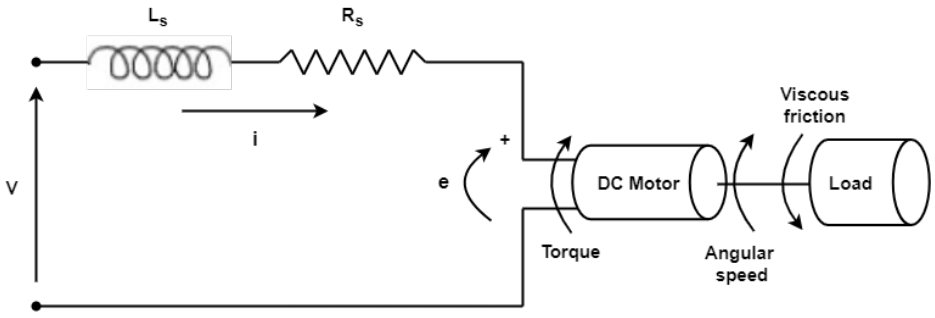


Figure 3.2: *Un tipico sistema elettromeccanico del motore DC*

Esso può essere descrito tramite due equazioni differenziali, una per il sistema elettrico e uno per il sistema meccanico, utilizzando le seguenti variabili:

- V = tensione
- i = corrente
- R_s = resistenza statorica per fase
- L_s = induttanza statorica
- e = back EMF
- J = inerzia
- B = coefficiente di frizione
- w_m = velocità angolare meccanica
- T_e = coppia elettrica
- T_l = coppia di carico
- k_e = costante di back EMF
- k_t = costante di coppia

Utilizzando la Legge di Kirchhoff alle tensioni, si ottiene l'equazione della parte elettrica:

$$V = L_s i + R_s \frac{di}{dt} + e \tag{3.1}$$

Mentre l'equazione meccanica risulta essere:

$$J\frac{dw_m}{dt} = T_e - Bw_m - T_l \Rightarrow J\frac{dw_m}{dt} + Bw_m = T_e - T_l \tag{3.2}$$

Portando le equazioni 3.1 e 3.2 nel dominio di Laplace, si ottengono rispettivamente le equazioni 3.3 e 3.4, di sotto riportate.

$$V = (R_s + sL_s)I + E \tag{3.3}$$

$$(B + sJ)\Omega_m = T_e - T_l \tag{3.4}$$

La somiglianza tra la generica macchina elettrica in DC e il motore BLDC vale:

- Quando tutte le fasi sono elettricamente simmetriche ed equilibrate.
- Durante il periodo in cui il flusso magnetico concatenato alle due fasi che si trovano in fase di conduzione risulta costante.

Fatte queste premesse, notiamo come in un motore BLDC funzionane in six-step commutation, l'impedenza delle fasi del motore, vista dall'inverter, risulta:

$$Z = 2[R_s + s(L_s - M_s)] = 2R_s + 2s(L_s - M_s) = R_a + sL_a \tag{3.5}$$

Dove:

- M_s è la mutua induttanza statorica
- $R_a = 2R_s$ è il doppio della resistenza statorica per fase
- $L_a = 2(L_s - M_s)$

Pertanto durante la conduzione a due fasi, il voltaggio di output dell'inverter avrà la forma:

$$V = (R_a + sL_a)I + E_a - E_c \tag{3.6}$$

Dove E_a è la back EMF indotta sulla fase A, e E_c quella indotta su C. Sapendo che:

$$E_a = -E_c = \frac{k_e}{2}\Omega_m \tag{3.7}$$

Possiamo riscrivere l'equazione 3.6 come:

$$V = (R_a + sL_a)I + 2E_a = (R_a + sL_a)I + k_e\Omega_m \tag{3.8}$$

Abbiamo quindi riscritto l'equazione elettrica nel caso di motore BLDC. L'equazione meccanica non subisce variazioni, ma può essere semplificata supponendo che non vi sia carico attaccato al motore ($T_l = 0$), e che la coppia elettrica possa essere scritta come:

$$T_e = k_t I \tag{3.9}$$

Per cui, dalla 3.4 si ottiene:

$$(B + sJ)\Omega_m = k_t I \tag{3.10}$$

Ricavando la I dall'equazione 3.8 si ottiene:

$$I = \frac{V - k_e\Omega_m}{R_a + sL_a} \tag{3.11}$$

E la si può sostituire nella 3.10 ottendo così:

$$(B + sJ)\Omega_m = k_t\frac{V - k_e\Omega_m}{R_a + sL_a} \tag{3.12}$$

Isolando il termine Ω_m a sinistra e V a destra, si ha:

$$[(B + sJ)(R_a + sL_a) + k_e k_t]\Omega_m = k_t V \tag{3.13}$$

Volendo ricavare la funzione di trasferimento tra la velocità del motore Ω_m e la tensione V in uscita dall'inverter:

$$\frac{\Omega_m}{V} = \frac{k_t}{(L_aJ)s^2 + (R_aJ + L_aB)s + (R_aB + k_e k_t)} \tag{3.14}$$

Assumendo che il coefficiente di frizione abbia un valore molto piccolo, tendente allo zero (stiamo supponendo di far girare il motore senza carico e in aria), si possono fare le seguenti considerazioni:

- $R_aJ >> L_aB$
- $k_e k_t >> R_aB$

Pertanto si può riscrivere la funzione di trasferimento 3.14 come:

$$\frac{\Omega_m}{V} = \frac{k_t}{(L_aJ)s^2 + (R_aJ)s + k_e k_t} \tag{3.15}$$

3.3 Cenni di teoria sul regolatore PID

Il controllore PID (Proporzionale-Integrale-Derivativo) è uno degli algoritmi di controllo più utilizzati nell'ingegneria dei sistemi e nel controllo automatico. La sua popolarità deriva dalla sua semplicità di implementazione e dalla sua efficacia nel migliorare le prestazioni dei sistemi di controllo. Esso si compone di tre parti che interagiscono assieme per mantenere il controllo di un sistema; le tre componenti fondamentali sono presentate di seguito.

- **Componente proporzionale (P)**
  Il termine proporzionale è responsabile della risposta immediata del controllore. Esso è direttamente proporzionale all'errore attuale $e(t)$ tra il valore desiderato (setpoint) e il valore misurato della risposta del sistema. La legge di controllo proporzionale può essere scritta come:
  $$P = K_P \cdot e(t) \tag{3.16}$$
  Un valore alto di K_P aumenta la reattività del sistema, ma può anche portare a instabilità e oscillazioni se non ben calibrato.

- **Componente integrale (I)**
  Il termine integrale si occupa dell'accumulazione degli errori passati: questa funzione risulta particolarmente utile per eliminare l'errore stazionario, ossia l'errore residuo che rimane dopo che il termine proporzionale ha fatto il suo lavoro. La legge di controllo integrale può essere espressa come:

$$I = K_I \cdot \int_0^t e(\tau)d\tau \tag{3.17}$$

  Un guadagno integrale troppo elevato può causare un effetto noto come "integral windup" (che verrà spiegato nella Sezione 3.3.2), dove l'accumulazione dell'errore porta a una risposta eccessiva e instabile.

- **Componente derivativo (D)**
  Il termine derivativo si basa sulla velocità di variazione dell'errore e fornisce un'azione di smorzamento. Questo componente aiuta a prevedere l'andamento futuro dell'errore e a correggerlo in anticipo, riducendo le oscillazioni. La legge di controllo derivativo è:

$$D = K_D \cdot \frac{d}{dt}[e(t)] \tag{3.18}$$

  Un valore elevato di K_D può migliorare la stabilità, ma rende anche il sistema sensibile al rumore, poiché il termine derivativo amplifica le variazioni rapide dell'errore.

L'unione delle tre componenti risulta nella seguente forma dello sforzo di controllo applicato dal regolatore:

$$u(t) = P + I + D = K_P \cdot e(t) + K_I \cdot \int_0^t e(\tau)d\tau + K_D \cdot \frac{d}{dt}[e(t)] \tag{3.19}$$

La sintonizzazione del controllore PID, ossia la scelta dei parametri K_P, K_I e K_D, è un processo critico ma che, se effettuato correttamente, risulta in una risposta del sistema di controllo reattiva, stabile e precisa. In Figura 3.3 è riportato come l'incremento di ogni guadagno influisca su alcune caratteristiche della risposta del sistema: la freccia in alto significa che aumentare il guadagno comporta l'incremento della caratteristica; la freccia in basso significa che ne comporta il decremento.

Parameter	Rise Time	Overshoot	Settling time	Steady-state error
Kp	↓	↑	small change	↓
Ki	↓	↑	↑	eliminate
Kd	small change	↓	↓	small change

Figure 3.3: *Come l'incremento dei guadagni di un controllore PID influiscono sulla risposta del sistema*

Chiaramente ogni applicazione richiede determinate caratteristiche: conoscere la teoria dei PID permette di creare un controllore adatto alle nostre esigenze, e con risposta che soddisfi le specifiche.

3.3.1 Controllore Proporzionale-Integrale

Nell'ambito dei motori elettrici si tende a non utilizzare controllori PID completi, e quindi composto di tutti i fattori proporzionale, integrale e derivativo, bensì si preferiscono regolatori PI, che possiedono quindi solamente le componenti proporzionale e integrale. I motivi di questa scelta sono molteplici.

- Il primo motivo risiede nella dinamica relativamente semplice e prevedibile dei motori elettrici; in molti casi il comportamento degli stessi è di primo o secondo ordine, per cui la risposa del sistema non richiede il termine derivativo nel controllore perché si ottenga una buona prestazione.
- Il termine derivativo in un controllore PID amplifica la risposta alle variazioni rapide dell'errore, incluse quelle dovute al rumore presente nei segnali di feedback (ad esempio, nelle letture dei sensori). Nei motori elettrici, il rumore nei segnali di feedback può essere significativo, pertanto il termine derivativo potrebbe introdurre instabilità o oscillazioni indesiderate. Il controllore PI, essendone privo, è meno sensibile al rumore e quindi garantisce risposte più stabili.
- Il controllore PI è più semplice da settare dal momento che ha un parametro in meno (il guadagno derivativo Kd: nei sistemi di controllo per motori elettrici, dove le condizioni operative possono variare e la sintonizzazione può richiedere adattamenti frequenti, questo rappresentare un vantaggio significativo in termini di tempi di sviluppo e manutenzione.
- Uno degli obiettivi principali nel controllo dei motori elettrici è eliminare l'errore stazionario, cioè l'errore residuo che può verificarsi quando il motore raggiunge la velocità desiderata. Il termine integrale del controllore PI è particolarmente efficace in questo compito, poiché accumula l'errore nel tempo e forza il sistema a correggere qualsiasi deviazione residua: in molti casi questo è sufficiente a garantire le prestazioni desiderate senza bisogno di un controllo derivativo.

Lo sforzo di controllo $u(t)$ prodotto da un controllore PI si presenta nella seguente forma:

$$u(t) = K_P \cdot e(t) + K_I \cdot \int_0^t e(\tau) d\tau \tag{3.20}$$

Notare che l'equazione precedente è uguale a quella del segnale di controllo del controllore PI, ponendo $K_D = 0$.

3.3.2 Saturazione e Anti-Windup

Gli attuatori reali, come ad esempio i motori elettrici, possiedono limiti fisici o operativi che definiscono il massimo e minimo valori dell'output. La saturazione di un attuatore si riferisce alla condizione in cui l'attuatore ha raggiunto il suo limite massimo o minimo di output e non è in grado quindi di fornire una risposta maggiore oppure minore, indipendentemente dal segnale di controllo inviato dal controllore. Tali limiti possono essere:

- Fisici: un motore elettrico può avere un limite di velocità o di coppia massimo oltre il quale non può operare.

- Di alimentazione: i motori elettrici sfruttano circuiti di commutazione che possono avere limiti di corrente e di tensione che, se superati, potrebbero rovinare irrimediabilmente le componenti elettroniche.

La saturazione di un attuatore risulta essere particolarmente pericolosa per un sistema, dal momento che può comportare una diminuzione delle prestazioni e una generale instabilità. Nel caso in cui il sistema di controllo sia dotato di un termine integrale, come nel caso dei controllori PI o PID, la saturazione risulta in un problema chiamato "windup integrale".

Il windup integrale si verifica principalmente in situazioni in cui il controllore continua ad accumulare errore integrale anche quando l'attuatore è già alla sua capacità massima o minima, ossia è saturo. Come spiegato in precedenza, un attuatore ha limiti fisici o operativi che, una volta raggiunti, ne impediscono l'aumento dell'output, anche se il controllore lo richiede: in questa situazione, l'errore tra output del sistema e setpoint continua a persistere, e di conseguenza il termine integrale continua ad accumulare l'errore. Nel momento in cui l'errore si riduce (ad esempio tramite un cambiamento del setpoint) e l'attuatore è nuovamente in grado di operare, il termine integrale è divenuto ormai troppo grande e ciò si traduce in una correzione eccessiva. Il windup integrale ha come effetti più comuni:

- Sovraelongazione, dal momento che la correzione eccessiva porta l'output del sistema a superare di molto il valore desiderato.
- Oscillazioni, che non necessariamente scompaiono nel tempo.
- Tempo di assestamento molto lungo.

Per evitare o limitare il windup integrale, esistono diverse tecniche e strategie, tra cui le più famose sono il "clamping" dell'integrale e la procedura di "anti-windup". Dal momento che nell'ambito dell'implementazione del controllore PI per il motore BLDC reale si è deciso di utilizzare la procedura di anti-windup (come si vedrà poi nella Sezione 4.3.2), ci si limiterà a spiegare solo questa tecnica. Quando l'attuatore è in saturazione, il termine integrale può essere "bloccato" o ridotto in modo da non accumulare ulteriormente errore. In particolare, una procedura di anti-windup consiste nell'incrementare il termine integrale solo quando il segnale di controllo non risulti già saturato. Nel caso in cui esso sia saturato, bisogna effettuare una valutazione: nel caso in cui l'errore sia in direzione di saturare ancor più il segnale, allora il termine integrale non deve essere incrementato; nel caso in cui invece l'errore sia tale per cui, aggiungendolo al termine integrale, si abbia un avvicinamento al limite di saturazione (e quindi un "miglioramento" dello sforzo di controllo, che riduce la sua saturazione), allora è necessario sommare l'errore al termine integrale.
Inoltre è buona norma introdurre un controllo che, nel caso in cui lo sforzo di controllo superi i limiti di saturazione, corregga lo sforzo di controllo al limite stesso, per preservare le componenti meccaniche ed elettroniche dell'attuatore.

4 Applicazione sul motore reale

In questo Capitolo verranno presentati gli strumenti hardware e software utilizzati per lo sviluppo di sistemi di controllo per un motore BLDC reale. Si avrà una breve presentazione del kit di sviluppo MCSPTEK144, si parlerà dei software MATLAB e Simulink nonchè di FreeMASTER, e infine si presenteranno le modifiche apportate al modello Simulink per la generazione del codice in linguaggio C da fornire al microcontrollore presente nel kit.

4.1 Hardware

Il MCSPTE1AK144 è un kit di sviluppo prodotto dalla NXP per motori trifase, siano essi brushless in corrente continua (BLDC) oppure sincroni a magneti permanenti (PMSM). Basato su microcontrollore Arm Cortex-M S32K1 a 32-bit e su Pre-Driver GD3000, questo kit permette un rapido prototipaggio di applicazioni di controllo per BLDC (Six-Step Commutation) e PMSM (Field-Oriented Control). I suoi applicativi software sfruttano la "Automotive Math and Control Library Set" (AMMCLib) e forniscono una vasta gamma di possibilità per l'implementazione di sistemi di controllo per questo tipo di motori.

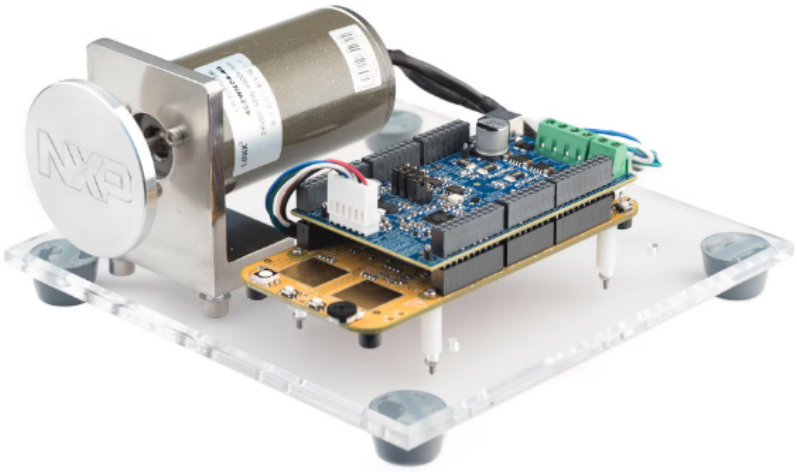


Figure 4.1: *Kit di sviluppo S32K144*

Il kit di sviluppo è composto da:

- **S32K144EVB**
  "Evaluation and development board" a basso costo per applicativi industriali e di automotive. Si basa, come anticipato in precedenza, su un MCU Arm Cortex-M4F S32k14 a 32 bit.
- **DEVKIT-MOTORGD**
  "Shield board" alimentata a 12 V e 5 A, compatibile con la S32K144EVB. Tale board

si basa sul pre-driver SMARTMOS GD3000 e possiede un'interfaccia per sensori Hall ed encoder, pertanto supporta sia applicativi di controllo sensorless, sia applicativi che sfruttano la sensoristica.

- **LINIX 45ZWN24-40**
  Motore BLDC trifase con due coppie di poli magnetici nel rotore, con valori nominali 24 V, 40 W, 4000 rpm, 2.3 A.

- Power Supply 1250APL05/S3 a 12 V e 5 A

4.1.1 Hardware setup

Il setup hardware risulta estremamente semplice. I passi da seguire sono i seguenti:

- Nella S32K144EVB:
  - assicurarsi che il jumper J107 sia in posizione 2-3; questa opzione fa sì che la board S32k144 sia alimentata attraverso cavo micro USB.
  - collegare la board al PC via cavo attraverso la porta micro USB.
- Nella DEVKIT-MOTORGD:
  - assicurarsi che il jumper J8 sia aperto.
  - assicurarsi che i jumpers J9, J10, J11 siano in posizione 2-3; se in questa posizione, la shield board viene utilizzata per controllo six-step (trapezoidale) per motori BLDC.
  - connettere i cavi del motore alle fasi del connettore J13 (fase A con cavo bianco, fase B con cavo blu, fase C con cavo verde).
  - connettere i sensori a effetto Hall al connettore JP1.
  - ruotare il trimmer R72 leggermente a sinistra rispetto alla posizione intermedia.
  - alimentare la board a 12 V attraverso il connnettore J7.

In Figura 4.2 viene proposto lo schema dei collegamenti da effettuare.

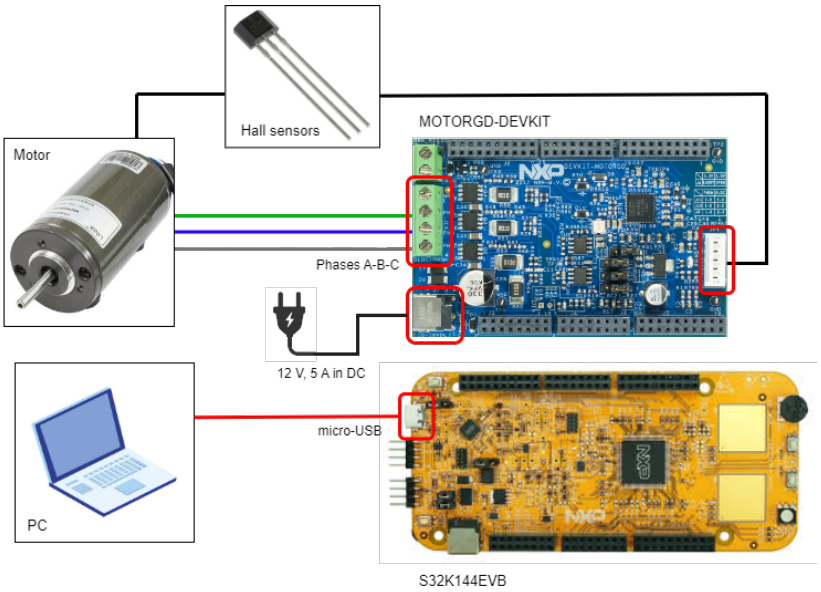


Figure 4.2: *Collegamenti hardware. Le due board MOTORGD-DEVKIT e S32K144EVB vanno sovrapposte l'una all'altra; i collegamenti dei pin sono banali e pertanto sono stati tralasciati in questo schema.*

4.2 Software

In questa Sezione vengono presentati gli strumenti software utili al fine di questa trattazione. I software utilizzati sono:

- MATLAB R2024a
- FreeMASTER 3.2
- AMMCLib

All'interno di MATLAB, in particolare, sono stati utilizzati vari Toolbox:

- Simulink
- Control System Toolbox
- Motor Control Blockset
- MATLAB Coder
- Simulink Coder
- Embedded Coder
- NXP_Support_Package_S32K1xx
- NXP_MBDToolbox_S32K1xx

- Stateflow

Spieghiamo più nel dettaglio alcuni dei sopra citati strumenti.

- **MATLAB R2024a**
  MATLAB (abbreviazione di MATrix LABoratory) è un ambiente creato dall'azienda Mathworks per il calcolo numerico e l'analisi statistica scritto in linguaggio C: costituisce un potentissimo strumento ingegneristico per via dei suoi numerosi strumenti a supporto dei più disparati campi di studio. Ai fini di questo progetto è stata utilizzata la versione R2024a, che comprende i toolbox precedentemente nominati aggiornati alle versioni più recenti (almeno nel momento in cui viene scritta questa tesi).

- **FreeMASTER 3.2**
  FreeMASTER è un software sviluppato da NXP Semiconductors che viene utilizzato per il monitoraggio e il controllo in tempo reale dei sistemi embedded. È particolarmente utile nel contesto dello sviluppo di applicazioni automotive, industriali e di automazione, dove è essenziale avere un feedback immediato dal sistema per garantire prestazioni ottimali e affidabilità. Ai fini di questo progetto, tale software verrà utilizzato con lo scopo di:

  – Inviare il riferimento di velocità al motore
  – Controllare le grandezze del sistema
  – Registrazione di dati per la verifica delle prestazioni

- **AMMCLib**
  La AMMCLib (Automotive Math and Motor Control Library) è una libreria software sviluppata da NXP Semiconductors, progettata per supportare lo sviluppo di applicazioni di controllo motore ad alte prestazioni e per fornire funzionalità matematiche avanzate nei sistemi embedded. Questa libreria è particolarmente utile nei settori automotive e industriale, dove la precisione e l'efficienza del controllo motore sono essenziali.

4.2.1 Software setup

Per effettuare il corretto setup dell'ambiente software, è necessario seguire la guida sotto riportata.

- In primis su MATLAB cliccare su Add-Ons > Install Add-Ons e installare:

  NXP_Support_Package_S32K1xx

  Successivamente cliccare su Manage Add-Ons e, nel menù a lato del toolbox stesso selezionare Open Folder. A questo punto sarà necessario, nella Current Folder di MATLAB, cliccare sul file con estensione .mltbx; questa operazione permette l'installazione dell'NXP_MBDToolbox_S32K1xx all'interno del quale si trova il modello Simulink del BLDC.

- Nella barra di ricerca del browser, navigare su www.nxp.com/AutoMCDevKits e selezionare il kit MCSPTE1AK144. Nella sezione Software cliccare "download options" di FreeMASTER Run-Time Debugging Tool e, successivamente, selezionare FreeMASTER tool 3.2 (includes Lite 1.3) – Windows installer. Una volta scaricato l'installer sarà necessario aprirlo ed effettuare i seguenti passaggi:
  - Accettare il License Agreement
  - Su "Choose additional" features basta selezionare solo la prima opzione, ossia FreeMASTER 3.2
  - Selezionare la cartella in cui effettuare l'installazione
  - Su "Choose Shortcut Folder" selezionare la prima opzione e spuntare "Create Icons for All Users"
  - Attendere che l'installazione sia terminata
- Nella barra di ricerca del browser, navigare su www.nxp.com/AutoMCDevKits e selezionare il kit MCSPTE1AK144. Nella sezione Software cliccare "download" su MCSPTE1AK144 Development Kit Application Software e, una volta scaricato l'installer, aprirlo e seguire la procedura standard. Al termine della procedura sarà richiesto di installare anche la AMMCLib, pertanto sarà necessario cliccare su "Download AMMCLIB" e, nella pagina che si aprirà, scaricare la versione più recente dell'installer (essendo in questo momento la versione 1.1.37 la più recente, l'installer avrà nome: S32K14X_AMMCLIB_RTM_1_1_37_BIN.exe). Una volta terminata l'installazione, aprire l'applicativo e seguirne i passaggi. Terminata l'installazione della AMMCLib, il software del kit MCSPTE1AK144 ne verificherà l'installazione.

Le operazioni da effettuare nel sito della NXP non possono essere eseguite senza prima essersi registrati.

4.3 Generazione e utilizzo dell'applicativo per il kit

Di seguito sono presentati il modello Simulink che verrà utilizzato per la generazione del codice C con le modifiche apportate, e si elencheranno le procedure da seguire per poter utilizzare l'applicativo per motore BLDC.

4.3.1 Modello Simulink

Con l'installazione di NXP_MBDToolbox_S32K1xx verranno creati una serie di modelli utili per le applicazioni NXP, come quello che verrà utilizzata in questa trattazione; i passi da seguire per accedervi sono elencati di seguito.

1. Sulla home di MATLAB cliccare su Add-Ons e poi su Manage Add-Ons
2. Nel menù alla destra di NXP_MBDToolbox_S32K1xx, selezionare Open Folder
3. Nella finestra Current Folder di MATLAB, seguire il path:

   S32_Examples\s32k14x\mc\BLDC_Demo\BLDC_ClosedLoop

Il modello Simulink che si ha a questo path, ossia il BLDC_ClosedLoop_s32k144.mdl, è un modello di controllo a ciclo chiuso della velocità di rotazione del motore che utilizza come metodo di commutazione la six-step commutation, progettato per interfacciarsi con la S32K144EVB.

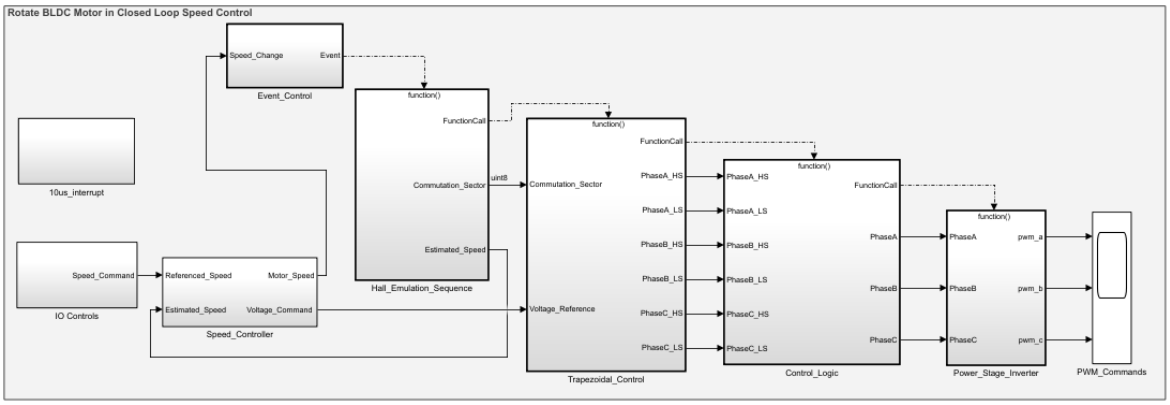


Figure 4.3: *Schema di controllo proposto dal modello Simulink*

Di seguito viene proposta una breve spiegazione dei blocchi principali; il fulcro del progetto però risiede nello Speed_Controller di cui si parlerà nella Sezione 4.3.2.

- Speed_Controller: riceve in ingresso la velocità desiderata del motore e la velocità reale dello stesso e, in base alla loro differenza, genera un segnale di voltaggio il cui obiettivo è quello di annullare tale differenza.

- Hall_Emulation_Sequence: è il blocco che elabora i segnali generati dai sensori a effetto Hall. Attraverso di essi è in grado: in primis di misurare la velocità del motore, e invia tale informazione allo Speed_Controller; in secondo luogo di determinare qual è il settore di commutazione.

- Trapezoidal_Control: riceve in ingresso il settore di commutazione e il comando di voltaggio. Al suo interno tali informazioni vengono tradotte in segnali da inviare all'inverter: sono definiti quindi i transistor da attivare e con velocità farlo (ricavato dal comando di voltaggio).

- Power_Stage_Inverter: riceve le informazioni su quali fasi attivare ad ogni istante e genera il segnale PWM da inviare alle stesse.

In generale quindi, data una velocità di riferimento in RPM (Revolutions Per Minute) e la velocità reale del motore calcolata attraverso i segnali prodotti dai tre sensori a effetto Hall, viene prodotto come comando di controllo un segnale di voltaggio. Quest'ultimo viene poi convertito in Duty Cycle che determina la velocità di accensione e spegnimento dei transistor propri del settore di commutazione individuato dai sensori Hall. Una volta immagazzinate tali informazioni, si è in grado di alimentare correttamente le fasi A - B - C attraverso tre segnali PWM, in modo da annullare la differenza tra velocità di riferimento e velocità misurata del motore.

4.3.2 Speed Controller & MATLAB function PI

Il fulcro della trattazione risiede nel blocco Speed_Controller. Al suo interno la differenza tra i segnali Reference_Speed e Motor_Speed, classificata come Speed_Error, viene utilizzata da un controllore PI per il calcolo del segnale di controllo. Tale controllore PI, che prende in ingresso anche i guadagni proporzionale (K_P), integrale (K_I), e i limiti di velocità superiore e inferiore del motore, è stato implementato scrivendo il codice direttamente all'interno di un blocco MATLAB Function, reperibile da Library Browser posto in alto a sinistra nella schermata di Simulink, alla voce Simulink > User-Defined Functions > MATLAB Function.

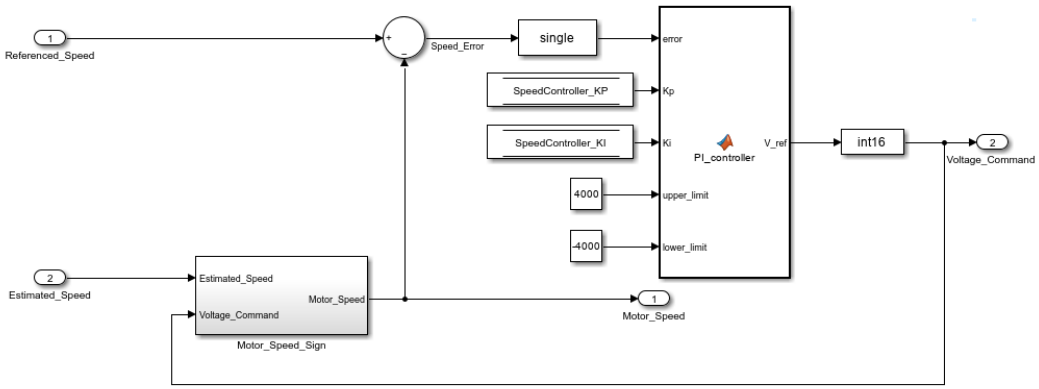


Figure 4.4: *Controllore della velocità dotato di PI Controller*

Avendo già presentato la teoria del controllore PI alla Sezione 3.3.1 di questa tesi, ci si limita alla spiegazione del codice all'interno del blocco PI_controller, riportato in Figura 4.5.

Dopo essersi assicurati che tutte le variabili siano di tipo `single`, e che la variabile persistente (che quindi non si annullerà ogni volta che verrà effettuata la procedura di calcolo del segnale di controllo, ma conserverà il suo valore) integral alla prima esecuzione venga inizializzata al valore nullo, si calcola il termine proporzionale come `P = Kp * error`. Il passo seguente corrisponde all'implementazione della tecnica anti-windup, per cui il termine integrale viene aggiornato con il valore al campionamento corrente solo ed esclusivamente in due casi:

- Il segnale di controllo non sia già saturato.
- Il segnale di controllo sia già saturato ma l'errore sia tale per cui, aggiungendolo al termine integrale, si abbia un avvicinamento al limite di saturazione (quindi un "miglioramento" del segnale saturato).

Questa procedura si traduce in un semplice `if` avente come condizioni le due casistiche presentate nel precedente elenco, e come codice eseguibile l'aggiornamento del termine integrale con l'errore all'istante di campionamento corrente (`integral = integral + error`). Successivamente si calcola il segnale di controllo non saturato come `V_ref_unsat = P + Ki * integral` secondo la teoria dei PID. L'ultima riga di codice genera il vero segnale di controllo, che deve rispettare i limiti di saturazione: nel caso in cui `V_ref_unsat` ecceda i limiti di saturazione (quindi sia minore del limite inferiore o maggiore del limite superiore) allora il

segnale di controllo sarà il limite che è stato infranto; nel caso in cui invece esso sia nel range consentito, allora varrà `V_ref = V_ref_unsat`.

```
function V_ref = PI_controller(error, Kp, Ki, upper_limit, lower_limit)

    % Persistent variable to memorize precedent values
    persistent integral

    % Convertion of every variable to single type
    error = single(error);
    Kp = single(Kp);
    Ki = single(Ki);
    u_limit = single(upper_limit);
    l_limit = single(lower_limit);

    % Persistent variable initialization
    if isempty(integral)
        integral = single(0);
    end

    % Proportional term calculation
    P = Kp * error;

    % Integral update must happen only if the output is not saturated
    % or if the error is reducing the saturation (Anti-windup procedure)
    if (P + Ki * integral < u_limit && P + Ki * integral > l_limit) || ...
       (P + Ki * integral >= u_limit && error < 0) || ...
       (P + Ki * integral <= l_limit && error > 0)
        integral = integral + error;
    end

    % Reference voltage calculation
    V_ref_unsat = P + Ki * integral;

    % Saturation application to reference voltage
    V_ref = min(max(V_ref_unsat, l_limit), u_limit);

end
```

Figure 4.5: *Codice MATLAB che implementa il PI controller con controllo di saturazione e anti-windup*

4.3.3 Generazione e caricamento del codice C

Simulink è in grado, a partire da un modello, di generare codice in linguaggio C che soddisfi le specifiche del modello, e di caricarlo all'interno di uno specifico hardware. Per effettuare tale operazione è necessario seguire la seguente procedura.

1. Collegare tramite cavo micro USB la board S32K144EVB del kit di sviluppo al PC che si utilizza per la generazione del codice a partire dal modello Simulink.

2. Assicurarsi che la Current Folder selezionata nel Workspace di MATLAB contenga il modello Simulink di cui vogliamo generare i codici: nel caso in cui modello e Workspace

non sabbiano stesso path, la generazione del codice non andrà a buon fine poichè il compilatore non sarà in grado di trovare i file contenenti le direttive di costruzione.

3. Nella prospettiva principale del modello BLDC_ClosedLoop_s32k144, cliccare sul riquadro MBD_S32K1xx_Config_Information > Target Connection > OpenSDA Drive Name nel riquadro OpenSDA: qui selezionare la board che si sta utilizzando.

4. Cliccare sul tasto a comparsa "Show Perspectives views" nell'angolo in basso a destra del riquadro in cui viene visualizzato il modello, e di conseguenza su "Enter perspective Code"; alla barra di navigazione di Simulink si aggiungerà la sezione "C CODE", nella quale sarà necessario cliccare il tasto "Build".

5. Attendere pochi minuti che l'operazione sia completata. Notare che quando il codice sarà completamente caricato all'interno del microcontrollore montato sulla board, il led verde di quest'ultima si spegnerà per qualche secondo ad indicare che sta avvenendo l'operazione di reboot; una volta riacceso il led, sarà possibile utilizzare il motore.

4.3.4 Applicativo FreeMASTER

Il modello Simulink presentato precedentemente non prevede alcun tipo di simulazione, viene utilizzato solo per la generazione del codice e il caricamento dello stesso sul MCU della board S23K144EVB. Pertanto, per poter utilizzare il motore con il codice prodotto è necessario utilizzare un applicativo FreeMASTER, denotato con lo stesso nome del modello ma con estensione .pmp, che si trova allo stesso path del modello: da MATLAB, nel riquadro Current Folder, fare tasto destro sul file BLDC_ClosedLoop_s32k144.pmp e selezionare l'opzione "Open Outside MATLAB".
Una volta aperto l'applicativo, sarà necessario effettuare il collegamento con l'hardware target, ossia il kit di sviluppo MCSPTE1AK144 della NXP. Ricordando di effettuare tutti i collegamenti (S32K144EVB collegata al PC tramite la porta micro USB, e la shield board MOTORGD collegata all'alimentatore), si procede cliccando, nella barra in alto, su Tools > Connection Wizard; all'interno di quest'ultimo cliccare il tasto "Avanti" finchè non si giunge alla scheda "Probing USB Ports", nella quale bisognerà selezionare la porta COM del PC cui è connessa la board S32K144EVB, e assicurarsi che l'opzione 115200 bit/s tra i Baud-rates sia selezionata. Una volta terminata l'operazione, il software FreeMASTER si collegherà alla board del kit permettendo l'utilizzo del motore.

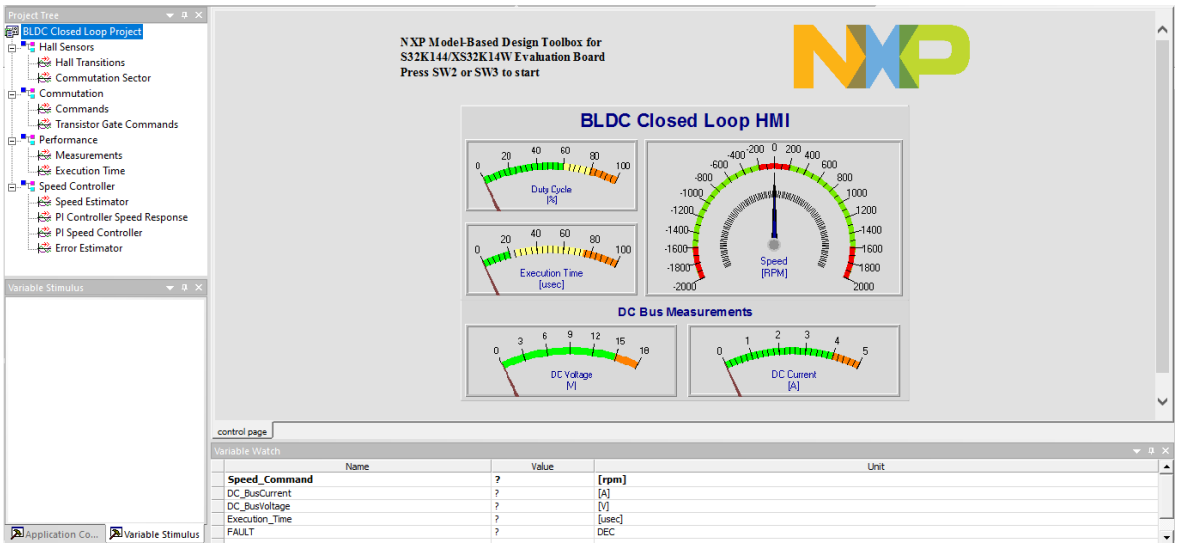


Figure 4.6: *Pannello di controllo FreeMASTER*

- A sinistra si ha il Project Tree, che può essere navigato e offre i grafici in tempo reale di alcuni grandezze fisiche del sistema, come la velocità e i segnali provenienti dai sensori Hall per fare un esempio.
- Nel riquadro centrale si può selezionare se mostrare a schermo la pagina di controllo, oppure uno dei grafici in tempo reale.
- In basso si ha il Variable Watch, che contiene i valori in tempo reale di alcune variabili fondamentali per il grafico che si è selezionato nel Project Tree. Se ad esempio si seleziona Speed Estimator alla voce Speed Controller, potrà essere monitorata la Motor_Speed e, allo stesso tempo, si potranno modificare i valori di Speed_Command (quindi il valore di velocità da raggiungere) e i valori dei guadagni K_P e K_I, senza dover rigenerare il codice a ogni modifica degli stessi.

Per avviare il motore per la prima volta dopo l'accensione dell'applicativo FreeMASTER è necessario, come scritto sul modello Simulink e sulla pagina di controllo, cliccare uno dei due pulsanti SW2 o SW3 sulla board: SW2 per inviare il primo comando di velocità in senso orario, SW3 per inviarlo in senso antiorario.

5 Analisi delle prestazioni

In questo Capitolo si presenterà la teoria degli indici di prestazioni integrali, e come questi sono stati calcolati, attraverso uno Script MATLAB, e applicati per settare al meglio i guadagni del controllore PI che gestisce la velocità del motore BLDC reale.

5.1 Indici di prestazione integrali

Per valutare la performance del sistema controllato tramite un controllore PID si utilizzano degli indici di prestazione, che si distinguono tra:

- Stazionari
- Dinamici, ossia basati sulla risposta dinamica del sistema controllato

Il discorso sugli indici di prestazione stazionari è presto liquidato. Esiste, infatti, un unico criterio basato sulla condizione raggiunta dal sistema controllato a transitorio esaurito (e quindi in condizione di regime): si chiede che l'errore e sia nullo in condizioni stazionarie. Per quanto riguarda gli indici di prestazione dinamici si ha un'ulteriore classificazione:

- Criteri puntuali, ossia si basano solo su alcuni punti della risposta
- Criteri integrali, ossia si basano sulla risposta completa del sistema nel corso del transitorio

Riducendo la trattazione agli indici di prestazione integrali, questi, questi risultano di fondamentale importanza per determinare quanto bene un controllore PID riduce l'errore tra il valore desiderato (setpoint) e il valore effettivo (output) di un sistema ($e(t) = y_{sp}(t) - y(t)$). Gli indici di prestazione integrali principali sono tre, e verranno presentati nelle successive Sezioni; per ora ci si limita a elencarli.

- Integral of Absolute Error (IAE)
- Integral of Squared Error (ISE)
- Integral of Time-weighted Absolute Error (ITAE)

La sintonizzazione dei parametri del controllore in retroazione si riduce quindi a un problema di ottimo consistente nella minimizzazione di uno o più indici di prestazione, in base al comportamento che si vuol fare adottare al sistema. La scelta degli indici da minimizzare dipende quindi dalle specifiche esigenze del sistema di controllo. Un sistema che richiede rapidità nel raggiungimento del setpoint beneficerà dell'uso dell'ITAE, mentre un sistema dove è cruciale minimizzare errori significativi potrebbe richiedere l'ISE. L'IAE rappresenta un buon compromesso per sistemi dove è richiesto un buon comportamento generale senza particolari enfasi su errori di diversa durata o ampiezza.

- **Integral of Absolute Error (IAE)**
  L'IAE misura l'integrale del valore assoluto dell'errore $e(t)$ nel tempo; costituisce un indicatore diretto di quanto l'errore si accumula nel tempo, senza dare particolare importanza alla durata temporale in cui esso si verifica. L'obiettivo del controllo è minimizzare questo indice per ridurre l'errore complessivo accumulato.

$$IAE = \int_0^\infty |e(t)|dt \tag{5.1}$$

  I suoi vantaggi consistono nella semplicità del calcolo e dell'interpretazione; inoltre non enfatizza errori a lungo termine rispetto agli errori a breve termine. Tuttavia non è in grado di distinguere tra errori grandi ma di breve durata ed errori piccoli ma persistenti, e può risultare in un comportamento oscillatorio del sistema, se non gestito correttamente.

- **Integral of Squared Error (ISE)**
  L'ISE considera il quadrato dell'errore nel tempo, dando maggiore importanza agli errori più grandi: minimizzare questo indice riduce quindi l'impatto dei grandi errori.

$$ISE = \int_0^\infty e(t)^2 dt \tag{5.2}$$

  Poiché l'errore viene elevato al quadrato, l'ISE penalizza maggiormente gli errori significativi, rendendo questo indice utile per sistemi dove tale tipo di errori è meno tollerabile. Tuttavia può trascurare errori piccoli ma persistenti, portando alla sottovalutazione della prestazione del controllore in certi contesti.

- **Integral of Time-weighted Absolute Error (ITAE)**
  L'ITAE moltiplica l'errore assoluto per il tempo t, il che significa che gli errori che si verificano più tardi nel tempo hanno un peso maggiore. Questo indice favorisce i sistemi che risolvono rapidamente gli errori iniziali, per cui la sua minimizzazione permette di ridurre gli errori a lungo termine.

$$ITAE = \int_0^\infty t|e(t)|dt \tag{5.3}$$

  I vantaggi consistono nel fatto che promuove un rapido assestamento del sistema, minimizzando gli errori nel lungo periodo, e risulta particolarmente utile per sistemi dove è cruciale ridurre l'errore il più presto possibile. Sfortunatamente risulta più difficile da calcolare rispetto agli altri indici, e potrebbe non essere adatto per tutti i tipi di sistemi, specialmente quelli dove gli errori iniziali sono meno critici.

5.1.1 Indici a tempo discreto

In un sistema di controllo a tempo discreto, questi indici di prestazione sono utilizzati per valutare l'efficacia del controllore PID. I dati dell'errore $e(k)$ sono raccolti ad ogni passo temporale k, e gli indici sono calcolati sommando i contributi di ciascun errore secondo le

loro formule discretizzate. Supponendo quindi $e(k)$ l'errore al passo temporale k, e N il numero totale di campioni considerati, i tre indici a tempo discreto risultano:

$$IAE = \sum_{k=0}^{N} |e(k)| \tag{5.4}$$

$$ISE = \sum_{k=0}^{N} e(k)^2 \tag{5.5}$$

$$ITAE = \sum_{k=0}^{N} k|e(k)| \tag{5.6}$$

Lo scopo di questa trattazione sarà quello di settare i guadagni K_P e K_I del controllore PI della velocità affinchè i tre indici a tempo discreto risultino nella "miglior" risposta (ulteriori considerazioni in merito verranno fatte in seguito).

5.2 Script MATLAB per il calcolo degli indici di prestazione

Il modello Simulink utilizzato in questa trattazione non supporta alcuna modalità di simulazione, pertanto tutti i test vanno effettuati sul sistema reale dopo che si è caricato il codice sul micro della board S32K144EVB. Il software FreeMASTER permette di salvare i valori misurati in tempo reale di alcune variabili di sistema all'interno di un database: nel nostro caso, esso consiste in un semplice file di testo composto di due colonne, in cui quella di sinistra contiene tutti gli istanti di tempo in cui il software FreeMASTER ha effettuato il campionamento della variabile, mentre quella di destra tutti i valori della variabile campionata. Il database si presenta come di seguito:

```
# Oscilloscope Data

# Seconds       Speed_Error

57.243726       0
57.245987       0
57.248575       0
57.251079       0
57.254690       0
57.258255       0
57.260600       1000
57.262874       1000
57.265564       1000
57.267787       1000
```

Figure 5.1: *Database dell'errore di velocità misurato*

Per registrare i valori assunti da una variabile durante l'esecuzione è necessario, nel "Variable Watch" di FreeMASTER, fare click con il tasto destro sulla variabile desiderata (che sarà Speed_Error) e selezionare "Create New Recorder"; nel pop-up che si apre, spostarsi nella sezione "Data Capture" e selezionare le opzioni "Capture when view is active" oppure "Capture data on background", poi la modalità di registrazione desiderata (si consiglia di

iniziare la registrazione sempre su un nuovo file, per semplicità), e infine "Text" come tipo di file. Al termine di ogni esecuzione, il file di testo verrà salvato direttamente nella cartella selezionata. Per elaborare i dati e calcolare gli indici di prestazione, si è implementato il seguente Script MATLAB:

```
clear; clc;

% Upload file data
data = readtable('file.txt', 'Delimiter', '\t', 'HeaderLines', 1);

% Divide the two columns of interest
time = data{:, 1};
speed_e = data{:, 2};

% Find the first index where speed_error is non-zero
first_index = find(speed_e ~= 0, 1);

% If non-zero values are found, stop the execution
if isempty(first_index)
    error('Non ci sono valori non zero nella colonna Speed_Error.');
end

% Extract data starting from the first non-zero value
time = time(first_index:end);
speed_e = speed_e(first_index:end);

% Find the index where time exceeds the first istant of three seconds
end_time = time(1) + 3;
end_index = find(time > end_time, 1);

% If there is not enough data for three seconds, use all available data
if isempty(end_index)
    end_index = length(time);
end

% Considers only data within three seconds of the first non-zero value
time_limited = time(1:end_index);
speed_e_limited = speed_e(1:end_index);

% Calculate the time differences between successive samples
dt = diff(time_limited);

% Calculate IAE
iae = sum(abs(speed_e_limited(1:end-1)) .* dt);

% Calculate ISE
ise = sum((speed_e_limited(1:end-1).^2) .* dt);

% Calculate ITAE
itae = sum((time_limited(1:end-1)-time(1)) ...
    .* abs(speed_e_limited(1:end-1)) .* dt);

% Show results
fprintf('IAE (3 secondi): %.2f\n', iae);
fprintf('ISE (3 secondi): %.2f\n', ise);
fprintf('ITAE (3 secondi): %.2f\n', itae);
```

Figure 5.2: *Script MATLAB che da file di testo calcola gli indici di prestazione integrali*

Si fanno alcune osservazioni sul codice:

- I valori presenti nelle due colonne vengono salvati in due vettori differenti che avranno, in ogni posizione, i valori di istante di campionamento e di variabile campionata in quell'istante.
- Nell'ambito del progetto, da motore fermo si è sempre immesso come segnale di riferimento un gradino di ampiezza 1000 rpm. Si è quindi voluto iniziare il calcolo degli indici a partire dal primo valore non nullo, e si è voluto terminare 3 secondi dopo, supponendo che nell'arco di quei 3 secondi l'errore si sia stabilizzato attorno allo 0 e che quindi sia già terminata la fase transitoria.
- Essendo che in FreeMASTER la distanza tra due campionamenti successivi non è costante, ma varia nell'ordine dei $10^{-4}s$, si è comunque preferito pesare ogni errore per il tempo che passa tra l'istante in cui esso è campionato e l'istante successivo. Da qui la necessità di utilizzare la variabile `dt = diff(time_limited)`.

Al termine dell'esecuzione, all'interno della Command Window di MATLAB verranno mostrati i tre indici. Cambiando i valori dei guadagni proporzionale e integrale, e salvando il risultato dell'esecuzione in un file diverso, è possibile confrontare con facilità i valore degli indici ottenuti per ogni combinazioni di guadagni.

5.3 Performance del controllo

Un sistema di controllo progettato correttamente tende a minimizzare gli indici di prestazione integrali. Si nota infatti che un basso valore dell'indice IAE risulta in una risposta complessivamente buona in ognuna delle sue fasi; un basso valore di ISE è indice che il sistema è in grado di gestire efficacemente le fasi transitorie e le oscillazioni; un basso valore di ITAE evidenzia la capacità del sistema in esame di stabilizzarsi in breve periodo e di non soffrire di oscillazioni a regime.
I test effettuati sul motore in funzione hanno evidenziato che le prestazioni con guadagno proporzionale $K_P = 0.2$ sono di gran lunga migliori rispetto a quelle generate da un qualsivoglia differente valore dello stesso, per cui ci si è successivamente concentrati nel settaggio del parametro K_I tenendo fisso K_P. Le verifiche effettuate sono state numerose, ma di seguito vengono proposti solo i tre test più significativi, con il calcolo degli indici dai valori della risposta a un gradino di 1000 rpm nei primi tre secondi: è infatti visibile nelle figure successive come dopo tre secondi si sia già raggiunta la velocità desiderata senza oscillazioni significative.

- **$K_P = 0.2$ & $K_I = 0.6$**
  Settare il guadagno $K_I = 0.6$ comporta una salita "lenta" ma sovraelongazione e oscillazioni praticamente nulli: può essere un valido settaggio per applicazioni che richiedono molta precisione anche a discapito della velocità di assestamento del sistema.

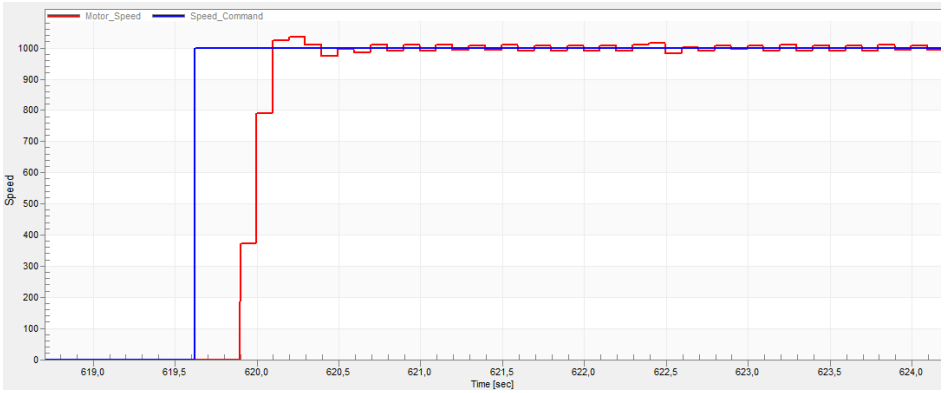


Figure 5.3: *Inseguimento del profilo di velocità con* $K_I = 0.6$

Gli indici di prestazione calcolati nei primi tre secondi assumono i seguenti valori:

- IAE = 325.61
- ISE = 250870.05
- ITAE = 92.36

- **$K_P = 0.2$ & $K_I = 0.7$**
  Settare il guadagno $K_I = 0.7$ comporta una salita rapida ma una sovraelongazione di quasi 200 rpm e una brevissima fase oscillatoria prima di assestarsi al valore desiderato: la risposta è complessivamente buona nonostante un picco iniziale della risposta che può risultare problematica in applicazioni ad alta precisione.

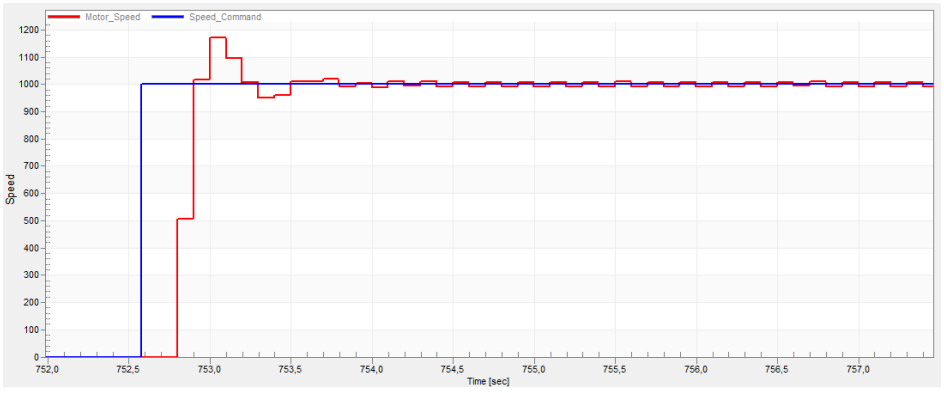


Figure 5.4: *Inseguimento del profilo di velocità con* $K_I = 0.7$

Gli indici di prestazione calcolati nei primi tre secondi assumono i seguenti valori:

- IAE = 312.91
- ISE = 234361.99
- ITAE = 88.43

- **$K_P = 0.2$ & $K_I = 0.8$**
  Settare il guadagno $K_I = 0.8$ comporta una salita rapida quanto quella ottenuta con $K_I = 0.7$, probabilmente a causa dei limiti fisici del sistema, ma risulta in una sovraelongazione ben più ampia (parliamo di circa 300 rpm) e in un maggior tempo di assestamento attorno al setpoint: tale settaggio risulta forse essere troppo responsivo e instabile rispetto agli altri due.

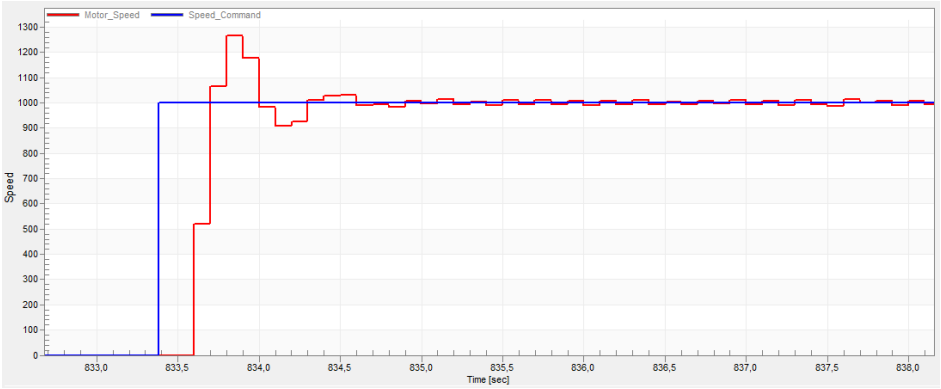


Figure 5.5: *Inseguimento del profilo di velocità con* $K_I = 0.8$

Gli indici di prestazione calcolati nei primi tre secondi assumono i seguenti valori:

- IAE = 342.99
- ISE = 233101.98
- ITAE = 110.36

Come possiamo notare il secondo settaggio è quello che minimizza sia l'indice IAE che l'indice ITAE; per quanto riguarda l'indice ISE, considerando l'incertezza dei sistemi di misura, il rumore e la particolare realizzazione dell'esperimento, possiamo considerare minimizzato anche quello. Settare quindi $K_P = 0.2$ e $K_I = 0.7$ permette di massimizzare le prestazioni del motore. Ovviamente in base al tipo di applicazione del motore, potrà essere più conveniente utilizzare un settaggio differente.

6 Conclusioni e sviluppi futuri

L'obiettivo principale di questa trattazione è il controllo della velocità di rotazione di un motore elettrico montato su di un kit di sviluppo fornito da NXP. Tale obiettivo è stato raggiunto attraverso l'implementazione di regolatori PI (proporzionali-integrali) in un modello di controllo nel software Simulink, utilizzando blocchi MATLAB Function che hanno permesso di sovrascrivere il codice esistente e di sviluppare così il controllore desiderato. Successivamente si sono poi valutati i valori ottimali dei guadagni del controllore attraverso il calcolo e la minimizzazione di alcuni indici di prestazione, IAE, ISE e ITAE.

Come specificato in precedenza, il modello Simulink utilizzato non prevede alcuna modalità di simulazione, pertanto tutti i test sono stati effettuati sul motore reale. L'ambiente reale comporta, a differenza di quello simulato, la presenza di rumore, di non linearità, di malfunzionamenti randomici: perciò è risultato di vitale importanza effettuare numerosissimi test, anche utilizzando sempre gli stessi valori dei guadagni, in modo tale da individuare un comportamento "medio" che dipendesse il minimo possibile dalle singole realizzazioni dell'esperimento. Un interessante sviluppo del progetto potrebbe essere appunto quello di creare un ambiente di simulazione ad hoc sfruttando Simulink; ciò permetterebbe di velocizzare molte delle operazioni che richiedono tempo e di poter stressare il sistema senza rischiare di fare danni.

Un altro possibile sviluppo del progetto potrebbe essere quello di ricavare in maniera sperimentale i parametri caratterstici del motore, consentendo quindi non solo di settare l'ambiente di simulazione, ma anche di sfruttare tecniche più avanzate per la sintesi dei regolatori: si parla ad esempio della tecnica di assegnamento dei poli, in parte utilizzata nell'ambito di questo progetto con l'ausilio di parametri stimati a partire da quelli relativi a un diverso frame, ossia il frame d-q utilizzato nell'ambito del controllo Field Oriented per motori PMSM. Non si è parlato di questa parte nella trattazione dal momento che i parametri stimati sono incerti e sono stati ricavati effettuando approssimazioni e semplificazioni; tuttavia tale operazione ha permesso di ricavare dei valori di partenza dei guadagni proporzionali e integrali, e quindi di poter agire in un intorno degli stessi per la ricerca dei migliori valori di K_P e K_I.

Si potrebbe inoltre sviluppare un sistema di controllo sensorless per lo switch sequenziale dei transistor che regolano il passaggio della corrente nelle fasi del motore, attualmente effettuato tramite la lettura del campo magnetico da parte dei sensori a effetto Hall. Aumentando la complessità computazionale dell'algoritmo di controllo, si potrebbe però ovviare a tutti i problemi figli dell'approccio sensoristico, primo tra tutti il rumore nelle misurazioni.

Bibliografia e sitografia

1. R. Krishnan. *Permanent Magnet Synchronous and Brushless DC Motor Drives*. Taylor & Francis Group, 2010

2. Liuping Wang, Shan Chai, Dae Yoo, Lu Gan, Ki Ng. *PID and Predictive Control of Electrical Drives and Power Converters using MATLAB/Simulink*. John Wiley & Sons, 2015

3. Krishna, Gowri, Babu, Akshita, Agarwal, Netha. *Modelling and Development of Controller for BLDC Motor*. International Journal of Electrical Engineering and Technology (IJEET), Volume 12, Issue 6, June 2021

4. Mathworks. *BLDC motor control*. https://it.mathworks.com/discovery/bldc-motor-control.html

5. NXP Semiconductors. *3-Phase Sensorless BLDC Motor Control Kit with S32K144 - Application Note*. Rev. 1, 06/2020

6. NXP Semiconductors. *Getting started with the s32k144 bldc/pmsm development kit*. 2022

7. Toshiba Electronic Devices & Storage Corporation. *120° Square-Wave Commutation for Brushless DC Motors - Application Note*. 2018

8. NXP Semiconductors. *S32k1 microcontrollers for automotive general purpose*. 2024.

9. H. K. Khalil. *Performance criteria for control systems: Ise, iae, and itae*. Journal of Dynamic Systems, Measurement, and Control, 2005