



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

CONTROLLO DIGITALE
DI VELOCITÀ
DI UN MOTORE BLDC

RELATORE: prof. Simone Buso

LAUREANDO: Enrico De Franceschi

ANNO ACCADEMICO 2021/2022

25 novembre 2022

Sommario

	Introduzione	3
1	Componenti del progetto	4
1.1	Scheda Nucleo-F334R8	4
1.1.1	Microcontrollore STM32F334R8	6
1.2	Scheda di espansione X-Nucleo-IHM07M1	7
1.2.1	L6230 DMOS driver	9
1.3	Motore BLDC	11
1.3.1	Teoria del funzionamento	13
1.3.2	Sensori ad effetto Hall	16
2	Software di sviluppo	17
2.1	STM32CubeMX	17
2.2	IDE Keil μ Vision 5	19
2.2.1	Debugger	21
3	Implementazione fisica del sistema	23
3.1	Controllo di corrente	24
3.2	Acquisizione dati dai sensori ad effetto Hall	28
3.3	Sezione di potenza	29
4	Configurazione iniziale del μC	34
4.1	Panoramica dei collegamenti	34
4.2	Clock	35
4.3	General Purpose IO	37
4.4	Convertitori A/D e D/A	37
4.4.1	ADC1 e ADC2	38
4.4.2	Direct Memory Access	40

4.4.3	DAC	41
4.5	Timer	41
4.5.1	Generatore PWM	41
4.5.2	Timer in modalità Output Compare	43
4.6	Interrupt	45
5	Modellizzazione del sistema in catena chiusa	47
5.1	Analisi del motore nel dominio della frequenza	48
5.2	Periodo di campionamento	51
5.3	Filtri digitali	52
5.4	Progettazione del controllore	55
6	Implementazione del codice	60
6.1	Struttura e funzioni	60
6.1.1	Gruppo Hardware	61
6.1.2	Gruppo Middleware	62
6.1.3	Gruppo Feedback	62
6.1.4	Gruppo User	63
6.2	Algoritmi	64
6.2.1	Calcolo della velocità di rotazione	64
6.2.2	Algoritmo di rotazione <i>6-steps</i>	65
7	Risposta al gradino	67
7.1	Risposta in catena aperta	68
7.2	Risposta in catena chiusa	69
8	Conclusioni	71

Introduzione

Lo scopo di questa tesi sperimentale è quello di riunire le competenze acquisite dal laureando durante i corsi di elettronica industriale e fondamentali di automatica per poi metterle in pratica nello sviluppo di un progetto in laboratorio.

Il lavoro consiste nella progettazione di un sistema di controllo digitale in catena chiusa per controllare la velocità di rotazione di un motore brushless trifase, coinvolgendo lo studente al cento per cento in tutte le diverse fasi che lo compongono: lo studio individuale della teoria su cui verte il funzionamento dei motori a induzione, la scelta di strumenti e dispositivi adeguati per l'implementazione fisica del sistema, la modellizzazione e l'analisi del motore nel dominio della trasformata di Laplace, il dimensionamento dei filtri e del regolatore PID digitali e la scrittura del codice per il microcontrollore.

Le specifiche che il nuovo sistema in catena chiusa dovrà soddisfare sono alquanto basilari: la capacità di inseguire con errore nullo un segnale di riferimento costante mantenendo il più possibile invariati i tempi di accelerazione e decelerazione del motore, in altre parole, senza peggiorare la risposta del sistema in seguito a una variazione a gradino del segnale di riferimento.

Il presente documento vuole riportare in maniera fedele ed esaustiva il percorso fatto per il raggiungimento del risultato finale, suddividendo le informazioni tra i vari capitoli in modo da rispettare un determinato ordine logico e cronologico.

Capitolo 1

Componenti del progetto

Per questo progetto sono stati utilizzati tre hardware fondamentali: un motore BLDC dotato di sensori ad effetto Hall, una scheda Nucleo ospitante il microcontrollore e una scheda “intermedia” di pilotaggio per interfacciare i primi due tra di loro.

Nel seguito saranno illustrate le caratteristiche principali di questi dispositivi.

1.1 Scheda Nucleo-F334R8

Questa scheda prodotta dalla ST Microelectronics è ottimo punto di partenza per lo sviluppo di un progetto basato su microcontrollore. Essa ospita il chip integrato STM32F334R8 (progettato dalla medesima azienda) e lo accompagna a diverse caratteristiche e componenti che ne agevolano l'utilizzo.

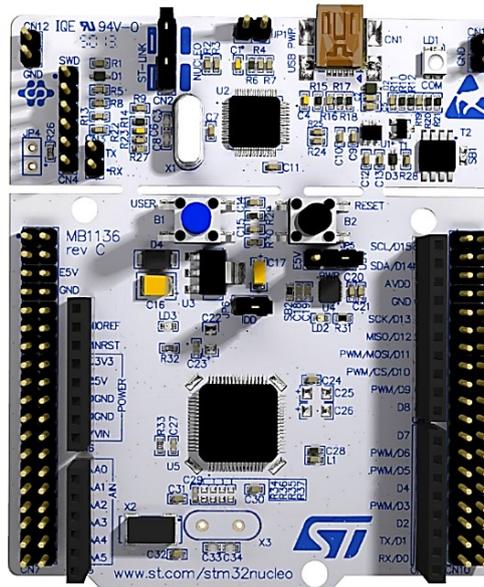


Figura 1.1: scheda Nucleo-F334R8 vista dall'alto

Ecco le caratteristiche principali che rendono la scheda versatile e affidabile:

- Alimentazione flessibile: tramite USB o alimentazione esterna a 3.3 V, 5 V, 7-12 V;
- Un oscillatore a cristallo con frequenza di 8 MHz utilizzabile come sorgente esterna di clock;
- Due pulsanti: uno di reset (nero) e uno configurabile (blu);
- Tre LED: uno che indica la comunicazione via USB (LD1), uno configurabile dall'utente (LD2) e uno indice di presenza dell'alimentazione;
- Connettori ST MORPHO per l'aggiunta di eventuali schede di espansione, e connettori Arduino Uno V3;
- Supporto da parte di diversi ambienti di sviluppo integrati (IDE).

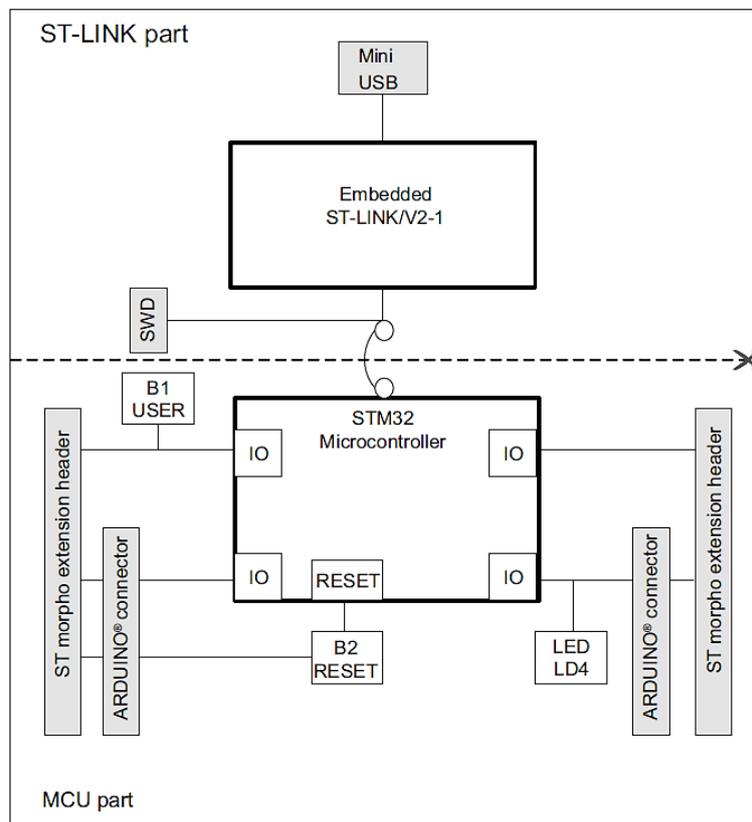


Figura 2: schema a blocchi dell'hardware della scheda Nucleo

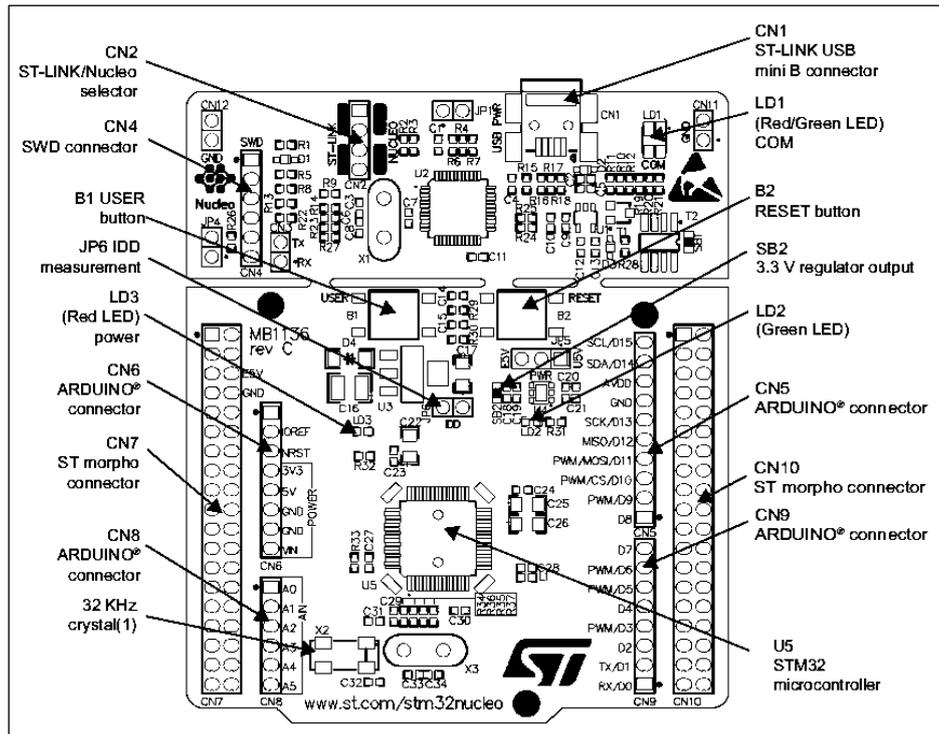


Figura 1.3: layout del lato superiore della scheda Nucleo

1.1.1 Microcontrollore STM32F334R8

Si tratta di un microcontrollore della serie STM32F3xx prodotta da ST Microelectronics ed appartiene alla gamma dei microcontrollori a “medie prestazioni”, ottimi per la maggioranza delle applicazioni basate su questi dispositivi.

Core	ARM® Cortex-M4	
Aritmetica	32	bit
Flash	64	Kbyte
SRAM	12	Kbyte
Max Clock	72	MHz
Voltage In	2.0 ÷ 3.6	V
Package	LQFP64	

Tabella 1.1: caratteristiche principali STM32F334R8

Questo chip è dotato di una vasta gamma di periferiche configurabili e funzionalità, tra cui:

- Due convertitori analogico-digitali, per un massimo di 21 canali utilizzabili in modalità single-ended o differenziale, frequenza di campionamento fino a 72 MHz, risoluzione impostabile tra 6, 8, 10 e 12 bit;
- Fino a 12 timer da 16 bit, di cui uno ad alta risoluzione (217 ps) e uno a 32 bit, configurabili come Input Capture (IC), Output Compare (OC) o generatori di PWM
- Real Time Clock
- DAC a tre canali con risoluzione 12 bit e alimentazione tra 2.4 V e 3.6 V;
- DMA a 7 canali
- Interfacce di comunicazione I²C, CAN, USART;
- Sensore di temperatura;
- Cyclic Redundancy check Calculation unit (CRC).

1.2 Scheda di espansione X-Nucleo-IHM07M1

X-Nucleo-IHM07M1 è una scheda di espansione prodotta da ST Microelectronics per il pilotaggio di motori brushless trifase con alimentazione in corrente continua, basata sul chip L6230 fabbricato dalla medesima compagnia.

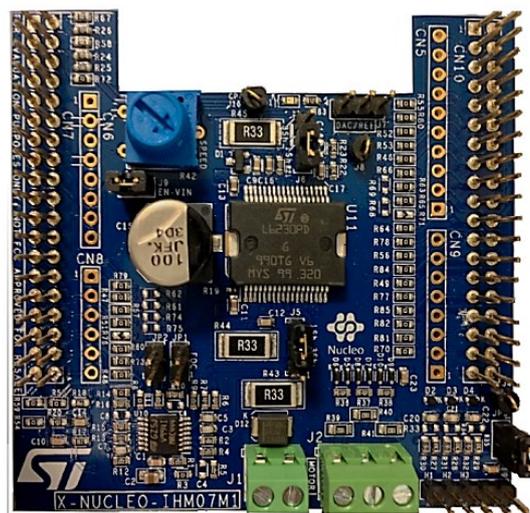


Figura 1.4: scheda X-Nucleo-IHM07M1 vista dall'alto

La scheda è dotata di connettori Morpho, i quali la rendono molto facile da interfacciare alla scheda Nucleo mediante semplice incastro.

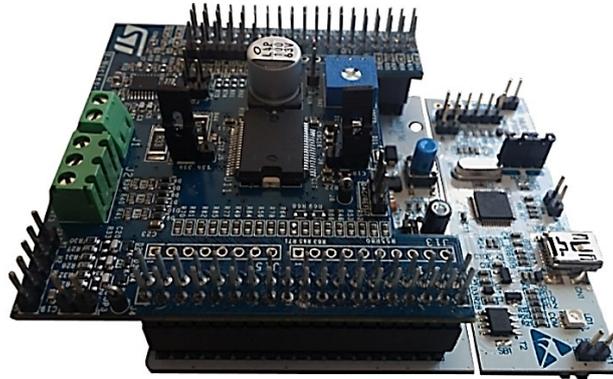


Figura 1.5: scheda di espansione montata sopra la scheda Nucleo

Vengono elencate di seguito alcune caratteristiche principali della scheda:

- Tensione di alimentazione compresa tra 8 V e 48 V;
- Corrente di picco massima in uscita: 2.8 A;
- Corrente media in uscita: 1.4 A;
- Frequenza di switching in uscita fino a 100 kHz;
- Sistema di rilevamento e protezione da sovracorrenti;
- Sensore di temperatura integrato e sistema di protezione da surriscaldamento
- Resistenze di shunt e relativi circuiti di condizionamento del segnale per il sensing di corrente;
- Connettori e circuiti di condizionamento per eventuali segnali di sensori ad effetto Hall o encoder;
- Potenzziometro integrato;
- Led configurabile.

Sulla scheda sono presenti alcuni Jumper, che a seconda della loro condizione (aperto, chiuso) provocano l'apertura o il cortocircuito tra due pin della scheda, permettendo di configurare a piacimento alcune impostazioni di quest'ultima.

Jumper	Configurazione	Posizione
JP ₁	Inserimento pull-up circuito sensing di corrente	Aperto
JP ₂	Incremento guadagno Op-Amp sensing di corrente	Aperto
JP ₃	Inserimento pull-up circuito acquisizione Hall/encoder	Chiuso
J ₉	Alimentazione scheda tramite scheda Nucleo	Aperto
J ₅	Sensing di corrente a singolo o triplo shunt	2-3 chiusi
J ₆	Sensing di corrente a singolo o triplo shunt	2-3 chiusi
J ₇	Connettore di debug, disponibile per collegamento sonda	Aperto

Tabella 1.2: configurazioni disponibili dei jumper

Sono presenti anche due connettori ad avvitamento (J₁ e J₂), uno a 2 terminali per il collegamento della sorgente di alimentazione a tensione costante, l'altro a 3 terminali per il collegamento delle fasi del motore.

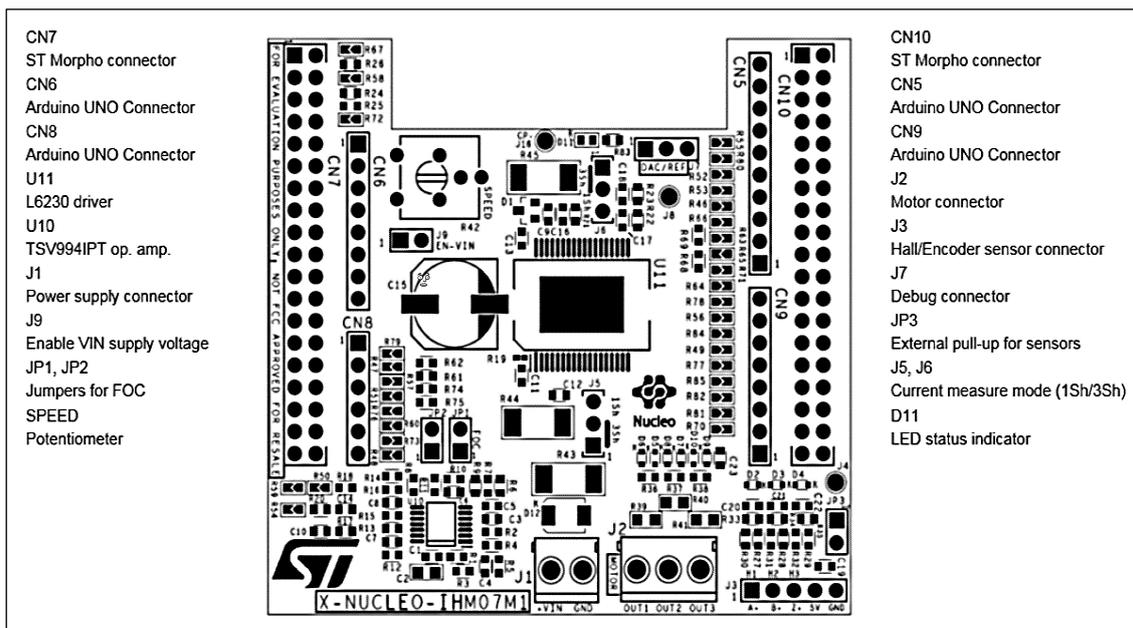


Figura 1.6: layout della parte superiore della scheda di espansione

1.2.1 L6230 DMOS driver

L'intero funzionamento della scheda di espansione X-Nucleo-IHM07M1 è basato sul circuito integrato L6230. Si tratta di un chip pensato appositamente da ST Microelectronics per il pilotaggio di motori BLDC con metodologia a 6 passi o ad orientamento del campo magnetico.

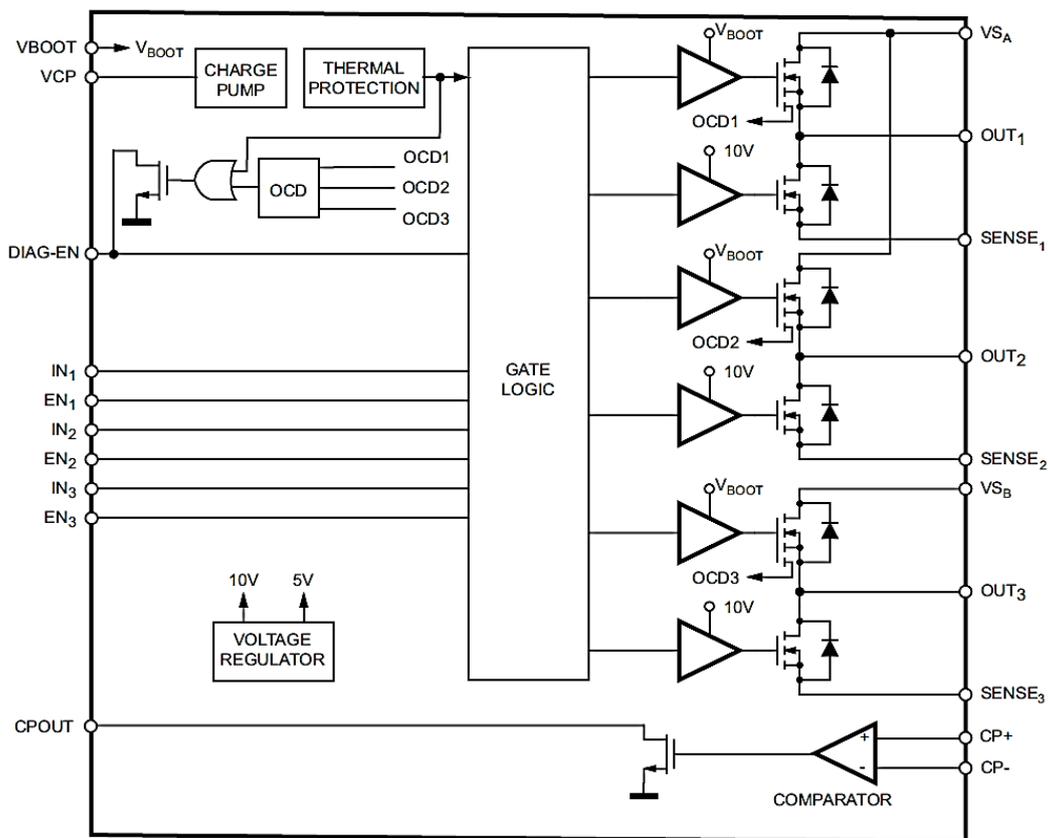


Figura 1.7: schema a blocchi del circuito integrato L6230

Dallo schema a blocchi si nota subito la sezione di potenza (a destra) costituita da tre half-bridge, ognuno dei quali gestisce il flusso di corrente per ciascuna fase del motore. La sezione di potenza e la sezione logica in ingresso sono tenute disaccoppiate mediante dei buffer ad alta impedenza. Ogni Mosfet inoltre è dotato di diodo di freewheeling per permettere il ricircolo di eventuali extracorrenti provocate dall'induttanza dagli avvolgimenti del motore.

Nella tabella 1.3 vengono descritti i pin di ingresso e di uscita più rilevanti per l'applicazione di cui si sta trattando.

I pin di SENSE sono collegati allo zero volt mediante delle apposite resistenze di shunt. Misurando la caduta di tensione su tali resistenze e condizionando opportunamente i segnali, si ricavano le correnti che scorrono all'interno delle fasi del motore.

Pin	Tipo	Funzione
IN ₁	Logic input	Logic input half bridge 1
EN ₁	Logic input	Enable input half bridge 1
IN ₂	Logic input	Logic input half bridge 2
EN ₂	Logic input	Enable input half bridge 2
IN ₃	Logic input	Logic input half bridge 3
EN ₃	Logic input	Enable input half bridge 3
CP ₊	Analog input	Input non invertente del comparatore interno
CP ₋	Analog input	Input invertente del comparatore interno
CP _{out}	Logic output	Output open-drain del comparatore interno
OUT ₁	Power output	Output half bridge 1
OUT ₂	Power output	Output half bridge 2
OUT ₃	Power output	Output half bridge 3
SENSE ₁		Pin di source half bridge 1
SENSE ₂		Pin di source half bridge 2
SENSE ₃		Pin di source half bridge 3

Tabella 1.3: legenda dei pin più significativi del circuito integrato L6230

1.3 Motore BLDC

Il motore utilizzato per questo progetto appartiene alla famiglia di motori BLDC QBL4208 prodotta dall'azienda tedesca Trinamic Motion Control. Si tratta di motori di qualità ad impiego universale, caratterizzati da un periodo di vita lungo grazie ai cuscinetti sferici interni.



Figura 1.8: motore BLDC QBL4208-41-04-006

Il motore utilizzato è il più piccolo della serie, ovvero il QBL4208-41-04-006. È un motore trifase dotato di un rotore con quattro coppie di poli e sensori ad affetto Hall posizionati a distanza di 30° l'uno dall'altro sullo statore.

Vengono riportate in tabella 1.4 alcune delle specifiche più importanti del motore.

Specifica	Unità di misura	Valore
Tensione di alimentazione	V	24
Corrente a regime	A	1.8
Velocità a regime	RPM	4000
Coppia a regime	Nm	0.0625
Corrente di picco	A	5.4
Resistenza tra 2 fasi	Ω	2
Induttanza tra 2 fasi	mH	2.1
Costante di coppia	Nm/A	0.035

Tabella 1.4: specifiche tecniche del motore

Con “a regime” si intende nella condizione di pieno carico e con la tensione di alimentazione indicata.

Dal motore fuoriescono otto cavi, di cui tre leggermente più spessi corrispondenti alle tre fasi degli avvolgimenti e i rimanenti cinque per l'alimentazione e i segnali dei sensori.

Fasi		
Giallo		Fase A – Canale 1
Rosso		Fase B – Canale 2
Nero		Fase C – Canale 3
Sensori		
Rosso		V _{cc} Hall Sensor (da 5 a 24 V continui)
Nero		GND Hall Sensor Ground
Blu		Sensore H ₁
Verde		Sensore H ₂
Bianco		Sensore H ₃

Tabella 1.5: legenda dei cavi del motore

1.3.1 Teoria del funzionamento

Si cerca ora di dare una spiegazione basilare sul funzionamento di un motore BLDC, spiegando in che modo esso sfrutta l'energia elettrica per ottenere un movimento e cosa sia necessario fare per mantenere il controllo di tale movimento.

Il motore è costituito da due componenti fondamentali: lo statore e il rotore. Come dice il nome, lo statore è la parte che rimane fissa ed è composto da alcuni avvolgimenti di rame opportunamente disposti ed equamente distanziati tra loro.

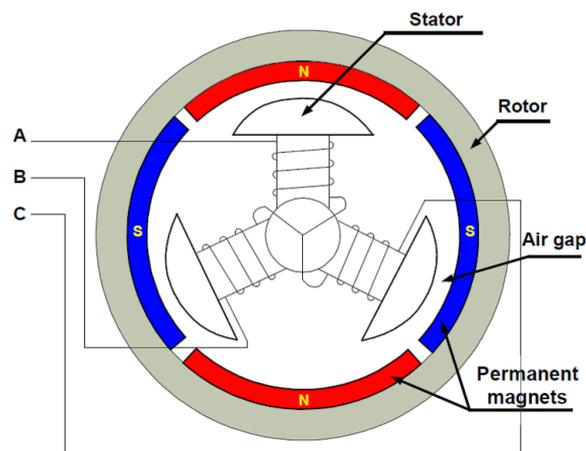


Figura 1.9: raffigurazione semplificata dell'interno di un motore trifase BLDC

Il rotore invece, ovvero la parte che ruota per l'appunto, è caratterizzato dalla presenza di magneti permanenti disposti circolarmente in modo da alternare un polo nord con un polo sud. Un motore con 4 coppie di poli significa che ha 8 magneti nel rotore.

Facendo scorrere una corrente elettrica attraverso un avvolgimento dello statore si crea un vettore di campo magnetico avente direzione parallela all'asse dell'avvolgimento stesso, e intensità proporzionale all'intensità della corrente elettrica. In questa situazione, il nucleo ferromagnetico che costituisce il supporto dall'avvolgimento diventa temporaneamente un magnete, acquisendo la capacità di respingere il medesimo polo o attrarre il polo opposto di un altro magnete.

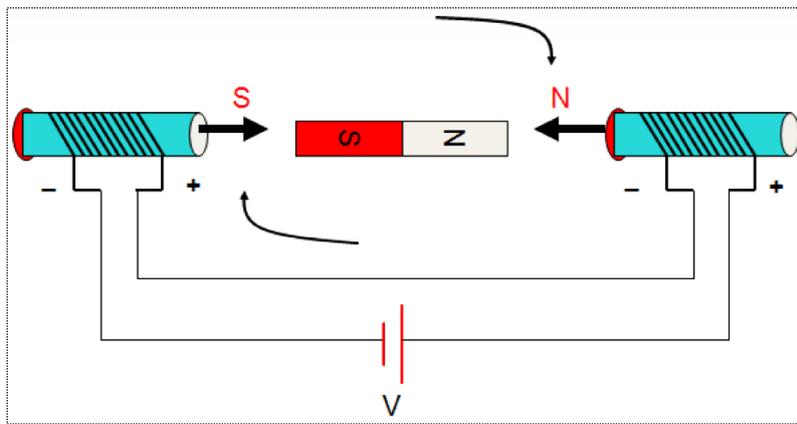
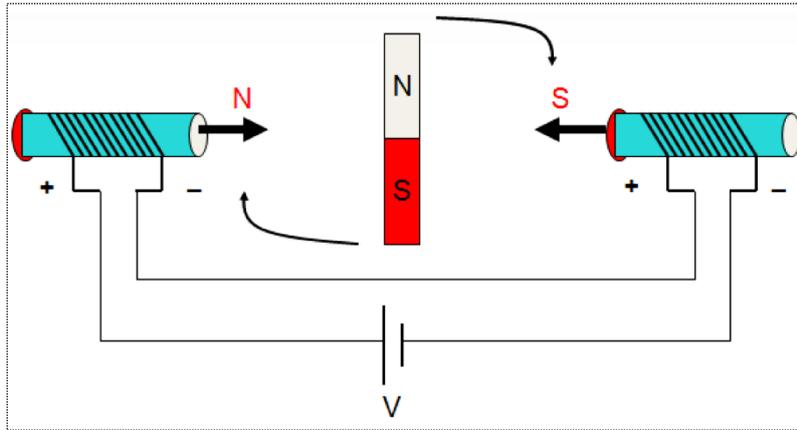


Figura 1.10 e figura 1.11: rappresentazione semplificata del processo di polarizzazione degli statori e movimento del rotore

Polarizzando opportunamente gli statori in successione si ottiene un vettore di campo magnetico detto *rotante* che, grazie alla forza sviluppata nelle interazioni tra i poli di statore e rotore, induce quest'ultimo a muoversi.

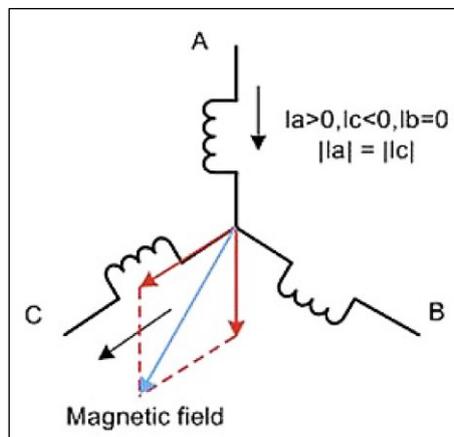


Figura 1.12: raffigurazione del campo magnetico rotante all'interno del motore

La velocità della rotazione dipende dall'intensità del campo magnetico e quindi dalla corrente elettrica che scorre negli avvolgimenti.

Il vettore del campo magnetico rotante si ottiene sommando i vettori dei singoli campi magnetici generati dagli induttori nei quali sta scorrendo la corrente. Applicando una differenza di potenziale, positiva o negativa, ai capi di due fasi e lasciando la terza flottante, si ottengono sei possibili direzioni del campo magnetico che distinguono i sei passi che compie il rotore quando si muove.

Per mezzo dei tre half-bridge presenti nella scheda driver X-Nucleo-IHM07M1 sarà possibile controllare l'intensità e il percorso della corrente nelle tre fasi del motore e di conseguenza decidere arbitrariamente il verso, la direzione e il modulo del campo magnetico rotante.

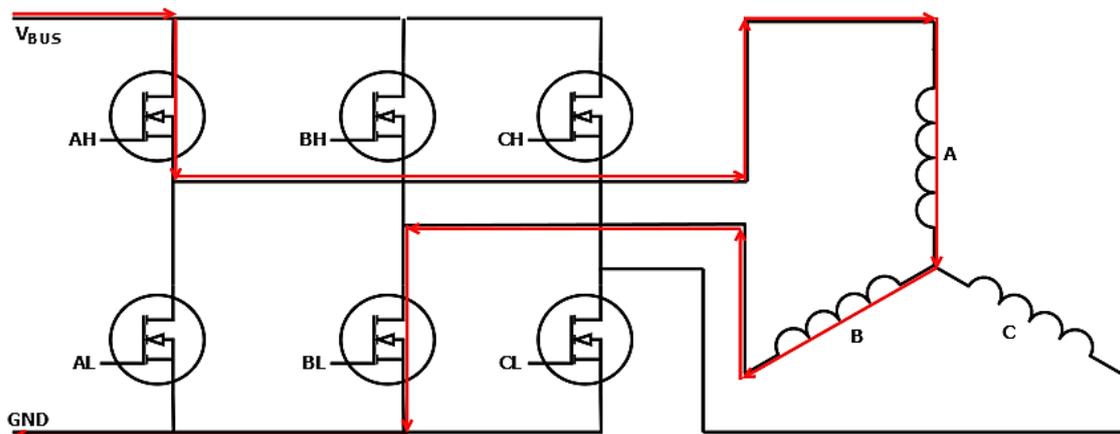


Figura 1.13: interfacciamento del driver con le fasi del motore, in rosso è evidenziato uno dei sei possibili percorsi che può compiere la corrente elettrica

In conclusione, programmando opportunamente il microcontrollore, è possibile impostare la corretta sequenza con cui gli half bridge commutano permettendo al rotore di compiere i sei passi che coprono una rivoluzione elettrica.

Si precisa che una rivoluzione elettrica corrisponde a una rivoluzione meccanica (360°) diviso il numero di coppie di poli del motore. In questo caso il motore ha 4 coppie di poli, dunque, una rivoluzione elettrica equivale a un angolo di 90° .

1.3.2 Sensori ad effetto Hall

Alcuni motori BLDC sono provvisti di sensori ad effetto Hall con lo scopo di conoscere in ogni momento la posizione del rotore. Un sensore ad effetto Hall dà in uscita un valore di tensione alto quando viene attraversato dal campo magnetico dei magneti permanenti del rotore. In particolare, se un polo nord del rotore è posizionato sopra un sensore ad effetto hall, quel sensore risponde con una differenza di potenziale pari a V_{dd} , altrimenti se in prossimità è situato un polo sud, in uscita si leggono zero volt.

Il motore è provvisto di tre sensori, uno per ogni fase. Unendo le risposte di tutti e tre i sensori si ottengono sei differenti parole da 3 bit ciascuna, dove ogni bit corrisponde al valore logico di un sensore. Ciascuna parola quindi ci permette di identificare in quale posizione si trova il rotore rispetto alle fasi.

Passo N^{\wedge}	Sensori $H_1-H_2-H_3$	Polarizzazioni
1	1-0-0	AC
2	1-1-0	AB
3	0-1-0	CB
4	0-1-1	CA
5	0-0-1	BA
6	1-0-1	BC

Tabella 1.6: corrispondenze tra numero di passo, output dei sensori e polarizzazione delle fasi

La tabella 1.6 illustra per ogni passo del rotore le corrispondenti uscite dei sensori e le fasi che devono essere polarizzate per muovere il rotore al passo successivo. Per esempio, la dicitura *AB* indica che si porta la fase *A* alla tensione di alimentazione e la fase *B* allo zero volt, mentre la fase *C* resta flottante.

Naturalmente per ovvi motivi di costruzione le combinazioni **000** e **111** dei sensori non sono contemplate.

Capitolo 2

Software di sviluppo

In questo capitolo verrà fatta una panoramica sui principali software impiegati durante lo sviluppo del progetto, dando una rapida spiegazione di come tali programmi sono stati utilizzati e di come orientarsi attraverso le diverse interfacce grafiche.

2.1 STM32CubeMX

STM32CubeMX è un software sviluppato da ST Microelectronics con lo scopo di aiutare e ridurre il carico di lavoro degli sviluppatori che intendono impiegare un microcontrollore STM32 per lo sviluppo del proprio progetto.

Grazie a questo software, infatti, tramite un'interfaccia grafica intuitiva è possibile nell'immediato settare le impostazioni di base del microcontrollore, scegliere quali periferiche utilizzare ed impostarne i relativi parametri di funzionamento.

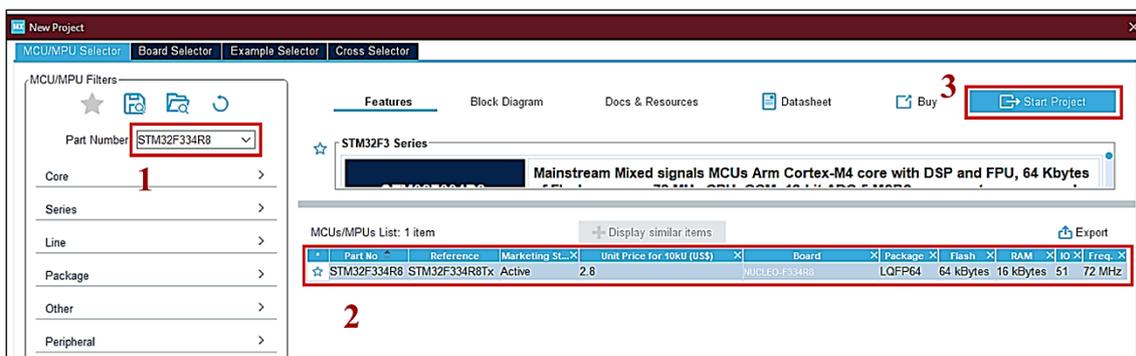


Figura 2.1: finestra "New Project" di STM32CubeMX

Quando si vuole iniziare un nuovo progetto con STM32CubeMX si presenta una finestra dove occorre inserire il nome del microcontrollore (riquadro 1) selezionare il risultato opportuno dalla lista (riquadro 2) e avviare la creazione del progetto (riquadro 3).

La schermata successiva è una overview del progetto, dalla quale è possibile creare la propria configurazione del microcontrollore.

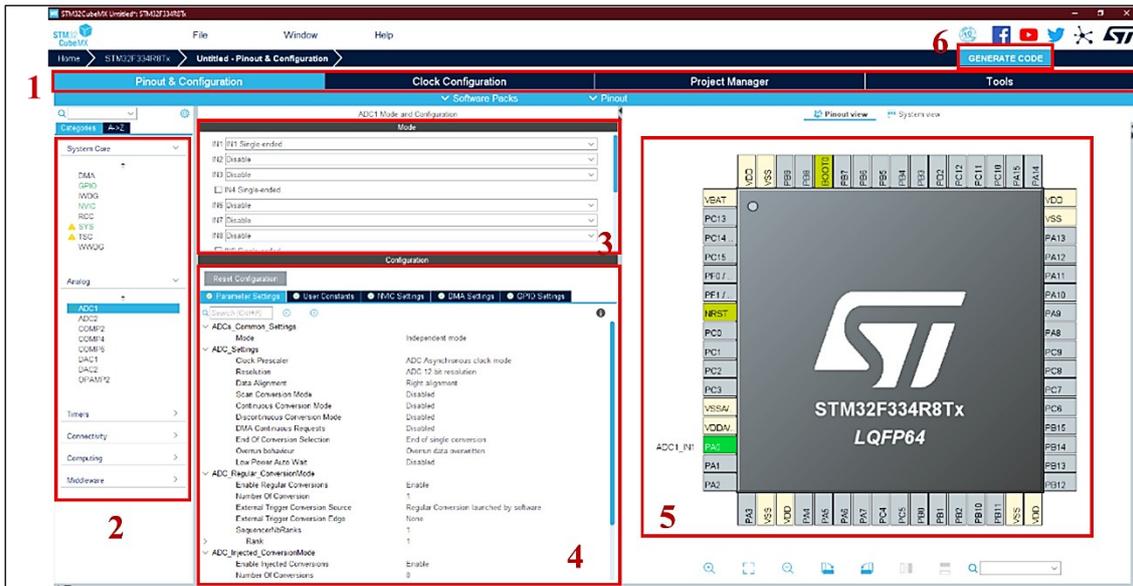


Figura 2.2: schermata principale del progetto

La schermata si presenta suddivisa in molteplici sezioni:

- Riquadro 1: questa è una barra di navigazione con la quale ci si può spostare tra le tre schermate principali di configurazione.
- Riquadro 2: qui sono elencate tutte le periferiche e funzionalità del micro come convertitori A/D e D/A, timers, DMA, comunicazione seriale, gestione degli interrupt, etc.
- Riquadro 3 e 4: cliccando su una periferica del riquadro 2 si aprono queste due sezioni dove è possibile configurare il modo di funzionamento della periferica, gli input e output, e tutti i parametri specifici che ne regolano il funzionamento.
- Riquadro 5: qui viene raffigurato il Pinout del microcontrollore mostrando quali pin sono abilitati, è inoltre possibile cliccare su un pin per visualizzare tutte le funzioni che può effettuare.
- Riquadro 6: una volta terminata la configurazione, cliccando su “Generate code” viene generato il codice.

Oltre alla schermata appena vista ce ne sono altre due fondamentali: la schermata di configurazione del clock il project Manager.

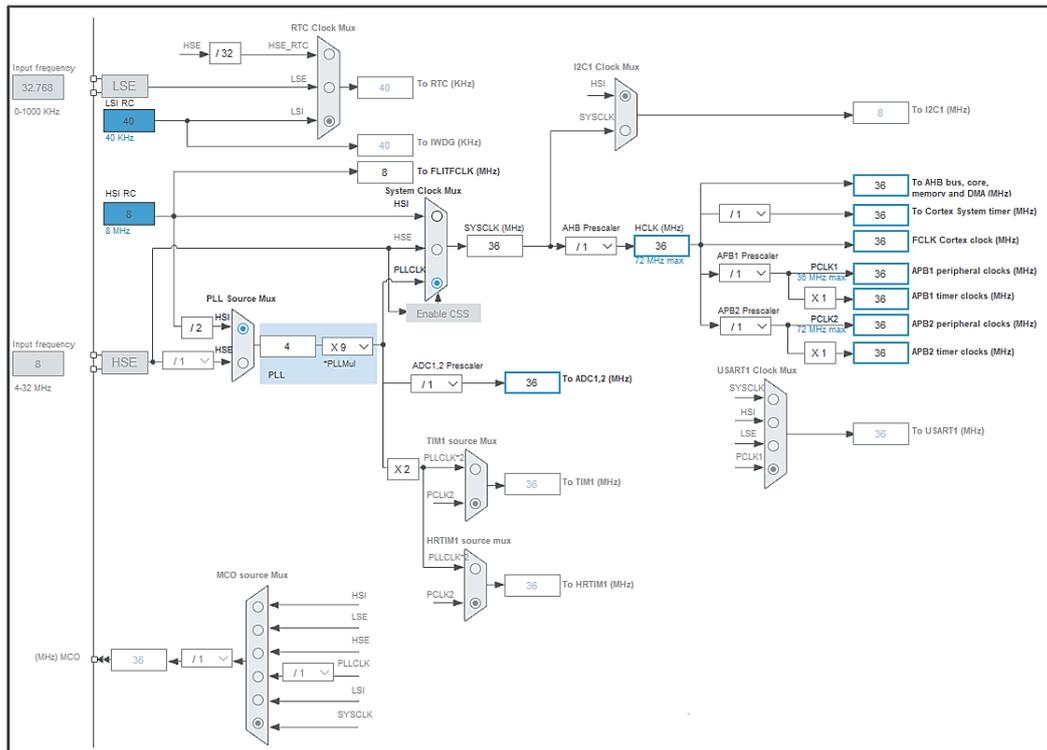


Figura 2.3: schermata di configurazione del clock

In questa schermata è possibile, tramite dei moltiplicatori e divisori di frequenza, andare a settare frequenze di clock differenti tra le varie periferiche. Per esempio, si potrebbe decidere di “rallentare” il funzionamento dell’ADC introducendo un fattore di divisione al suo ingresso rispetto al resto del sistema.

Infine, nella schermata di Project Management si vanno a impostare il nome del progetto, il percorso dove andrà salvato e alcuni parametri relativi alla generazione del codice, ad esempio l’IDE impiegato e la sua versione.

2.2 IDE Keil μ Vision® 5

Keil μ Vision è un software gratuito che offre un pacchetto completo per lo sviluppo di un progetto con un microcontrollore basato su architettura ARM. In un unico ambiente, infatti, è in grado di offrire molteplici funzionalità indispensabili:

- gestione dei progetti;
- editing dei codici sorgente;

- debugging environment.

Dopo aver premuto il pulsante “Generate code” in STM32CubeMX viene creata una cartella nella locazione impostata contenente i file di progetto e un eseguibile per aprire il lavoro in microVision.

All’avvio del programma, l’interfaccia si presenta come mostrato in figura 2.4.

La schermata si compone in varie sezioni e barre degli strumenti: nella figura 2.4 sono stati evidenziati, oltre alle sezioni, i pulsanti più importanti utilizzati frequentemente.

- Riquadro 1: si tratta della finestra di gestione del progetto. Qui i file sorgenti (.c) e i file header (.h) possono essere organizzati in cartelle per mantenere un certo ordine logico-gerarchico e facilitare la navigazione all’interno del progetto stesso.
- Riquadro 2: in questa finestra viene visualizzato il file su cui attualmente si sta lavorando, le linguette in alto permettono di spostarsi velocemente tra i file che sono stati aperti ma che al momento non sono attivi.
- Riquadro 3: questa barra contiene i pulsanti che avviano la compilazione del codice; si può scegliere di ricompilare un singolo file piuttosto che l’intero progetto. Inoltre, il pulsante più a destra permette di caricare il codice macchina nel microcontrollore (che sarà stato precedentemente collegato al computer).
- Riquadro 4: nella finestra di output vengono riportati gli esiti dei processi di compilazione e caricamento dei codici. Se si verificano degli errori durante la compilazione o nel caricamento, vengono stampati dei messaggi con la descrizione dei problemi ed eventualmente la loro posizione all’interno del codice.
- Riquadro 5: questo pulsante apre una finestra nella quale è possibile modificare tutte le impostazioni del programma di debugging.
- Riquadro 6: questo pulsante avvia il debugger. È indispensabile che il computer abbia la possibilità di scambiare informazioni con il microcontrollore per effettuare il debugging del codice.

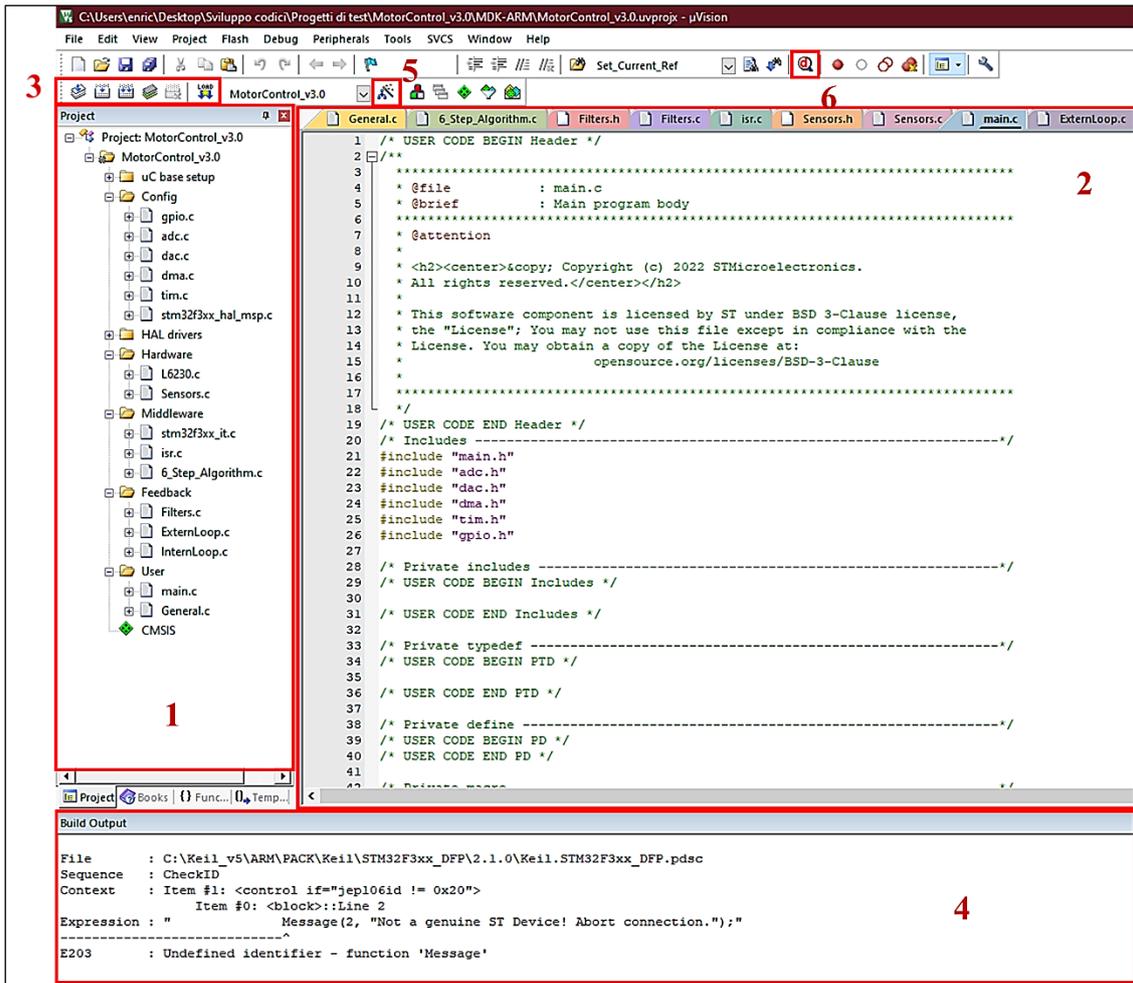


Figura 2.4: schermata principale del progetto in µVision

2.2.1 Debugger

Dopo aver caricato il codice macchina nel microcontrollore, cliccando sull'icona presente nel riquadro 6 di figura 2.4, si entra nel programma di debugging incorporato di µVision.

L'interfaccia del debugger si presenta come mostrato in figura 2.5.

Anche qui la schermata si presenta divisa in molteplici settori, ognuno avente una propria utilità che tra poco andremo a descrivere.

- Riquadro 1: i tre pulsanti fondamentali. Da sinistra verso destra: pulsante *Reset* per resettare il micro, pulsante *Run* per riprendere l'esecuzione del codice e pulsante *Stop* per fermarla.

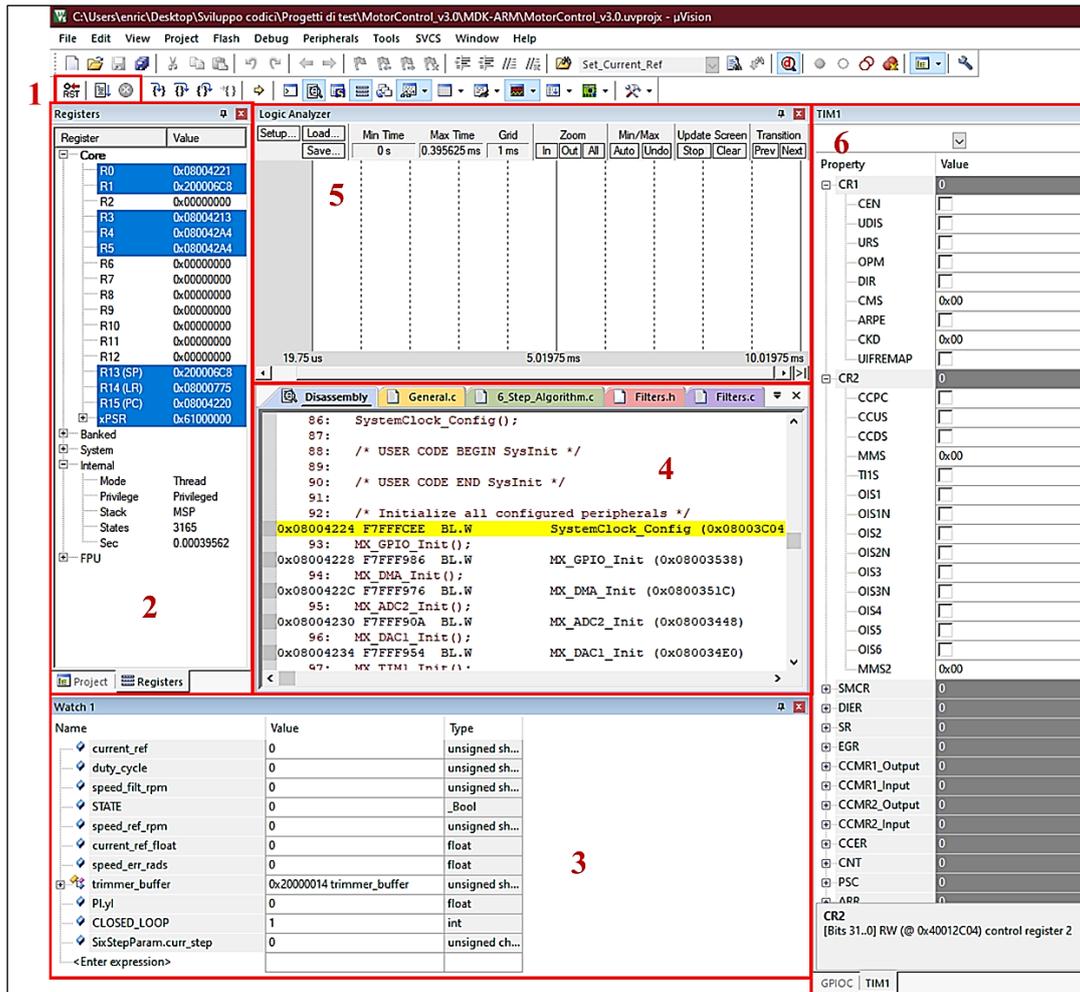


Figura 2.5: schermata di debugging di μ Vision

- Riquadro 2: *Registers*. Qui sono aggiornati i valori dei registri primari del microcontrollore istante per istante durante l'esecuzione.
- Riquadro 3: *Watch Window*. In questa finestra è possibile inserire le variabili e i vettori definiti globalmente che si vuole monitorare durante il debugging, si può anche modificare il valore di una variabile durante l'esecuzione del codice.
- Riquadro 4: *Disassembly Window*. Il segmento di codice in esecuzione viene tradotto in assembly e mostrato in questa finestra.
- Riquadro 5: *Logic Analyzer*. Con il Logic analyzer è possibile visualizzare l'andamento nel tempo di variabili definite globalmente.
- Riquadro 6: finestra delle periferiche. Selezionando la periferica desiderata, vengono mostrati tutti i registri relativi a tale periferica (la frequenza di aggiornamento dei valori non è real-time ma dipende dal bit-rate del canale USB).

Capitolo 3

Implementazione fisica del sistema

Andiamo ora ad analizzare maggiormente nel dettaglio la struttura fisica del progetto.

Come già anticipato, lo scopo di questa applicazione è quello di poter controllare la velocità di rotazione del motore; per farlo si è deciso di adottare un controllo in catena chiusa a doppio anello.

L'anello interno consiste in un controllo della corrente di picco negli avvolgimenti del motore, dove essa viene comparata con un segnale di riferimento. Tale segnale di riferimento di corrente viene generato dall'anello esterno, dove è presente il regolatore.

Il controllo viene implementato digitalmente nel microcontrollore e si tratta di un regolatore PI.

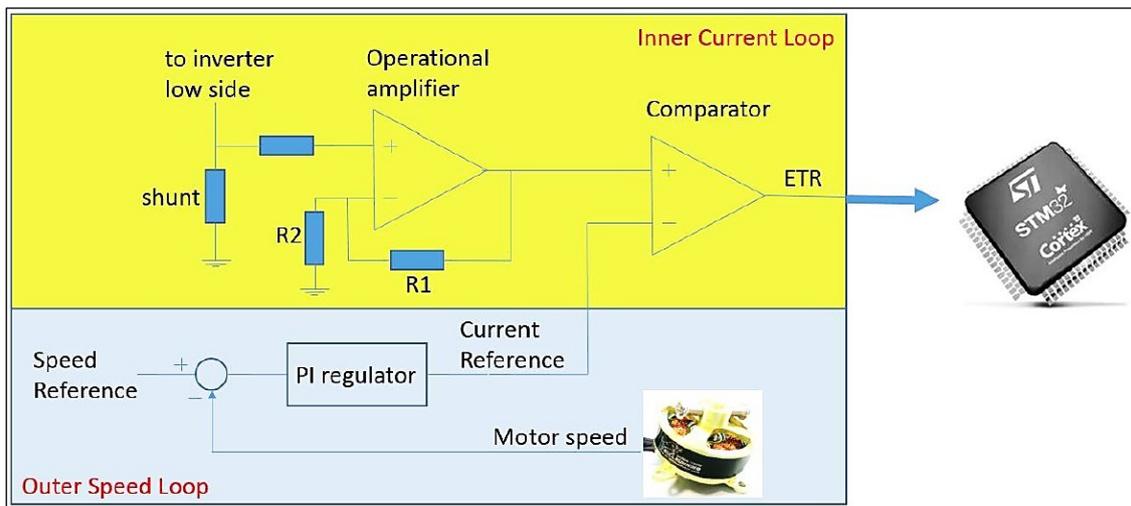


Figura 3.1: schema del controllo in catena chiusa a doppio anello. Controllo di corrente (parte gialla) e controllo di velocità (parte blu)

3.1 Controllo di corrente

Per poter implementare il controllo di corrente è necessario ricavare un segnale che contenga l'informazione di quanta corrente sta scorrendo negli avvolgimenti. Per questo scopo la scheda driver X-Nucleo-IHM07M1 monta delle resistenze di shunt da 0.33 Ω collegate alla parte bassa degli half-bridge presenti nel chip L6230.

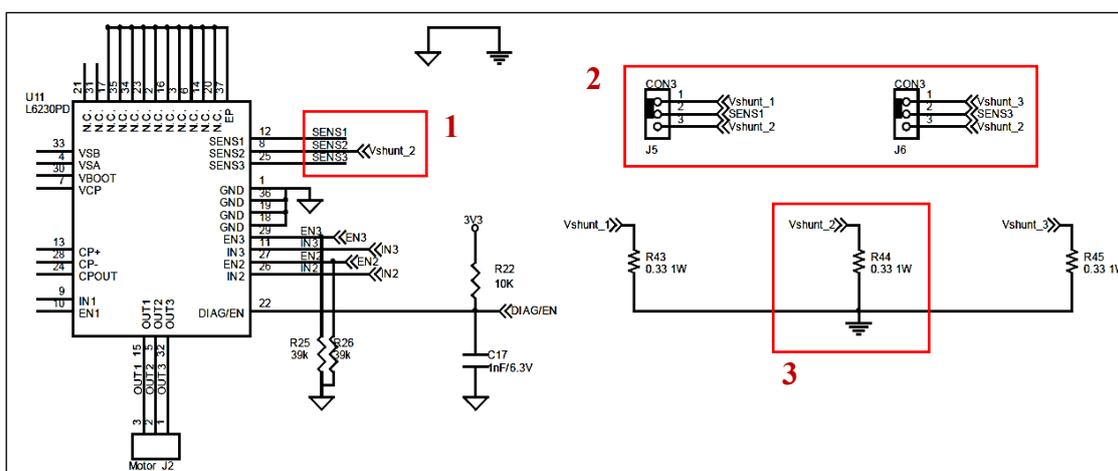


Figura 3.2: schematico riportante le resistenze di shunt, il loro collegamento al circuito integrato L6230 e i jumper di selezione three/single shunt

Le resistenze di shunt sono tre, una per ogni gamba dell'inverter. In base alla posizione in cui si trova il rotore, su una di queste tre resistenze si legge una caduta di tensione non nulla, mentre le altre due restano in uno stato di alta impedenza. La scheda offre la possibilità di cortocircuitare tutti i pin di SENSE in modo tale che tutte le correnti di ciascun half-bridge attraversino sempre il medesimo resistore di shunt; ciò costituisce una semplificazione vantaggiosa nel nostro caso, perché risulta così sufficiente controllare una singola caduta di tensione V_{shunt} invece di tre. Per selezionare la modalità *single shunt* occorre cortocircuitare i pin 2-3 dei jumpers J5 e J6 (riquadro rosso 2 in figura 3.2).

Il segnale V_{shunt} viene poi condizionato dal circuito mostrato in figura 3.3.

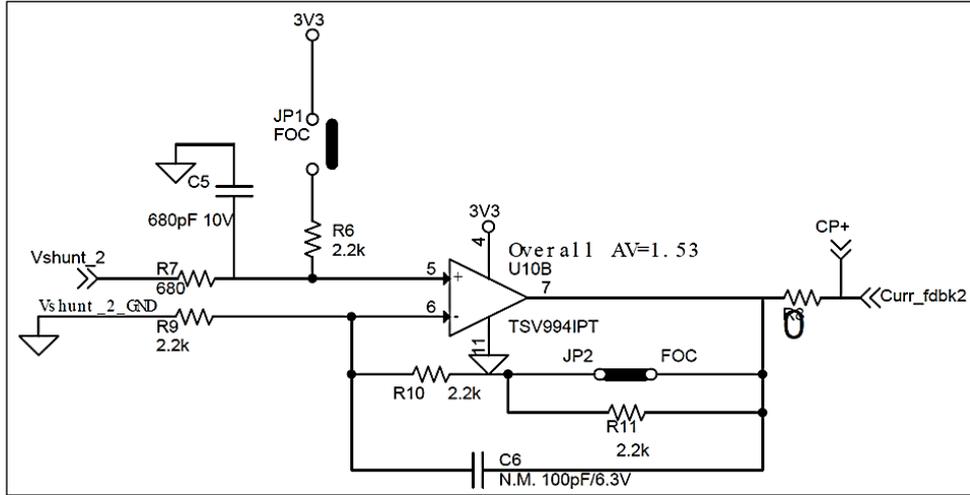


Figura 3.3: circuito per il condizionamento del segnale V_{shunt}

Come si può osservare dalla figura 3.3, tale circuito consiste in un amplificatore operazionale in configurazione non invertente, con un guadagno in tensione pari a 1.53. Il condensatore C_6 limiterebbe la banda dell'amplificatore permettendo di eliminare i disturbi ad alta frequenza, ma non è stato montato sulla scheda dal fabbricante.

I jumper JP_1 e JP_2 permettono rispettivamente di aggiungere un offset al segnale e di incrementare il guadagno dell'amplificatore; nel nostro caso terremo JP_1 aperto e andremo ad aprire JP_2 per raddoppiare il guadagno di tensione. Chiameremo il segnale di tensione in uscita da questo blocco V_{CP+} .

Dunque, abbiamo che

$$\frac{V_{CP+}}{V_{shunt}} = A_v = 3.06 \quad (1)$$

Tale guadagno dell'amplificatore è stato scelto opportunamente per ottenere un rapporto unitario tra V_{shunt} e la corrente $i(t)$ che scorre negli avvolgimenti del motore, infatti:

$$V_{shunt} = R_{shunt} \cdot i(t) \quad (2)$$

Quindi sostituendo la (2) nella (1) otteniamo:

$$\frac{V_{CP+}}{R_{shunt} \cdot i(t)} = A_v \Rightarrow i(t) [A] = \frac{V_{CP+}}{R_{shunt} \cdot A_v} = \frac{V_{CP+}}{0.33[\Omega] \cdot 3.06} \cong \frac{V_{CP+}}{1 \Omega} \quad (3)$$

Ciò significa che un valore di V_{CP+} pari a 0.5 V corrisponde a una corrente di 0.5 A, 1 V corrisponde a 1 A, e così via. Sapendo che la corrente $i(t)$ può variare tra zero e circa 1.5 A, allora sappiamo che V_{CP+} assumerà valori compresi tra zero e 1.5 Volt.

Ora è necessario generare un segnale di riferimento, che chiameremo V_{CP-} , con il quale effettuare il confronto di V_{CP+} . Per questo scopo si utilizza il convertitore digitale-analogico (DAC) contenuto nel microcontrollore.

Infine, per operare il confronto tra V_{CP+} e V_{CP-} sfrutteremo il comparatore analogico presente all'interno dell'integrato L6230. Il segnale d'uscita del comparatore, che chiameremo ETR (External Trigger), è un segnale logico che ci indicherà quando, negli avvolgimenti del motore, la corrente elettrica ha superato il riferimento impostato dal regolatore.

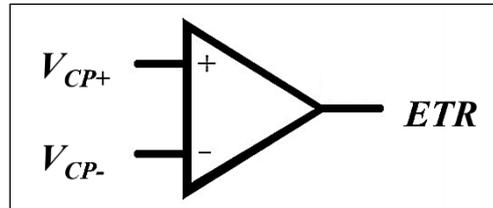


Figura 3.4: segnali di ingresso e di uscita del comparatore

Riassumendo, la relazione tra V_{CP+} , V_{CP-} e ETR è la seguente:

$$ETR = \begin{cases} 0, & \text{se } V_{CP+} < V_{CP-} \\ 1, & \text{altrimenti} \end{cases} \quad (4)$$

Il segnale ETR verrà utilizzato per azzerare i segnali PWM che pilotano gli inverter. In questo modo quando la corrente eccede, verrà a mancare la tensione di alimentazione ai capi degli induttori, i quali inizieranno il processo di scaricamento della corrente.

Osservando la figura 3.5 possiamo notare l'andamento triangolare della corrente, dovuto al fatto che la tensione che viene imposta ai capi degli avvolgimenti è costante a tratti. Infatti, secondo la relazione che lega la corrente i_L e la caduta di tensione v_L di un induttore:

$$i_L(t) = \frac{1}{L} \int_0^t v_L(t') dt' \quad (5)$$

risulta chiaro che nell'intervallo di tempo in cui il segnale PWM è pari a V_S (dove V_S è la

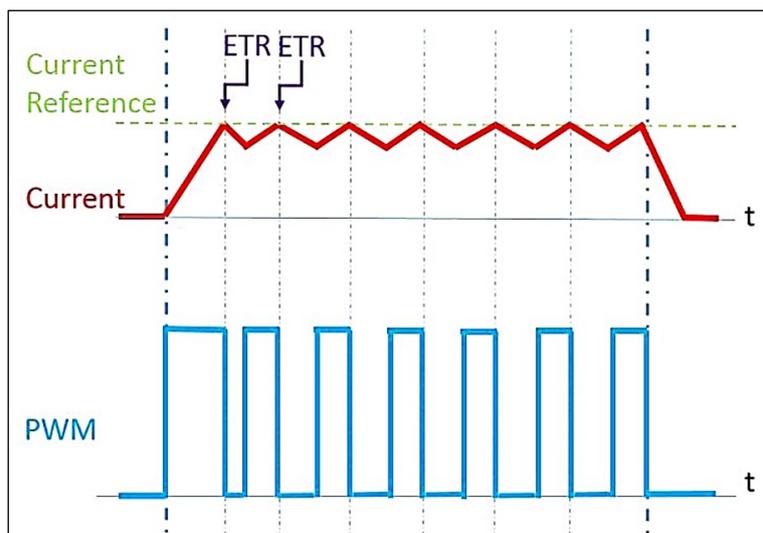


Figura 3.5: esempio grafico dell'andamento della corrente negli avvolgimenti del motore

tensione di alimentazione nominale del generatore) la tensione v_L è vincolata a essere costante e positiva, di conseguenza il suo integrale corrisponde a una rampa lineare crescente, dove il coefficiente angolare di tale rampa è inversamente proporzionale all'induttanza L .

Quando invece la PWM è nella parte di periodo dove è a zero volt, l'induttore vede ai suoi capi una tensione con polarità inversa rispetto al caso precedente, ovvero una tensione negativa: tale fenomeno deriva dalla legge di conservazione dell'energia ed è spiegata dalla legge di Lenz. Integrando dunque questa tensione negativa (costante per un breve periodo iniziale) si ottiene una discesa a rampa lineare della corrente.

In figura 3.6 è riportata una schermata dell'oscilloscopio durante il monitoraggio di alcune grandezze relative al controllo di corrente del motore:

- la traccia verde corrisponde alla corrente elettrica nell'avvolgimento del motore,
- la traccia violetto è il segnale di riferimento i_{ref} generato dal regolatore mediante il DAC,
- infine, la traccia gialla è il segnale ETR che raggiunge i 3.3 Volt quando la corrente supera il riferimento.

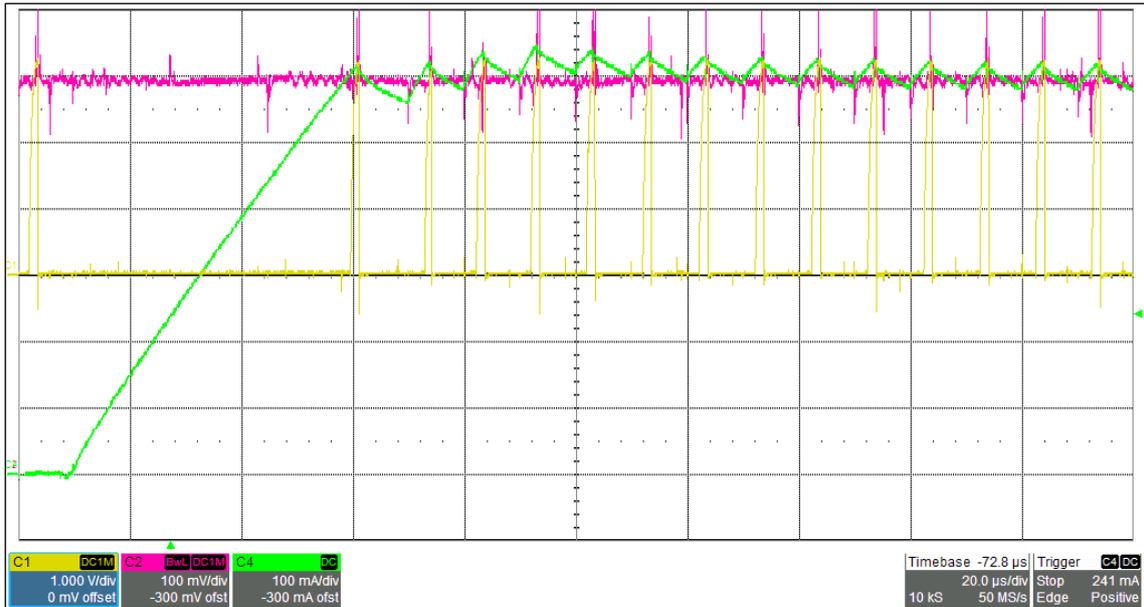


Figura 3.6: schermata dell'oscilloscopio raffigurante l'andamento della corrente in un avvolgimento del motore (in verde)

Si può notare una prima fase di “caricamento” della corrente: quando avviene il cambio di polarizzazione delle fasi è necessario attraversare un transitorio nel quale la corrente elettrica deve raggiungere il valore di regime; in altre parole, occorre del tempo affinché l'induttanza accumuli energia magnetica. La durata del caricamento dipende linearmente dal valore di corrente che si vuole raggiungere: correnti più alte richiedono tempi più lunghi.

3.2 Acquisizione dati dai sensori ad effetto Hall

Come già visto nel capitolo 1, il modello di motore brushless impiegato nel progetto è dotato di sensori ad effetto Hall per conoscere in ogni momento la posizione del rotore rispetto alle fasi dello statore. Nella tabella 1.5 si riporta un totale di cinque cavi relativi alla trasmissione dei dati e all'alimentazione di tali sensori.

I collegamenti devono essere effettuati con la scheda driver X-Nucleo-IHM07M1, la quale ospita un circuito di condizionamento (mostrato in figura 3.7) apposito per i segnali provenienti da sensori ad effetto Hall oppure da encoder.

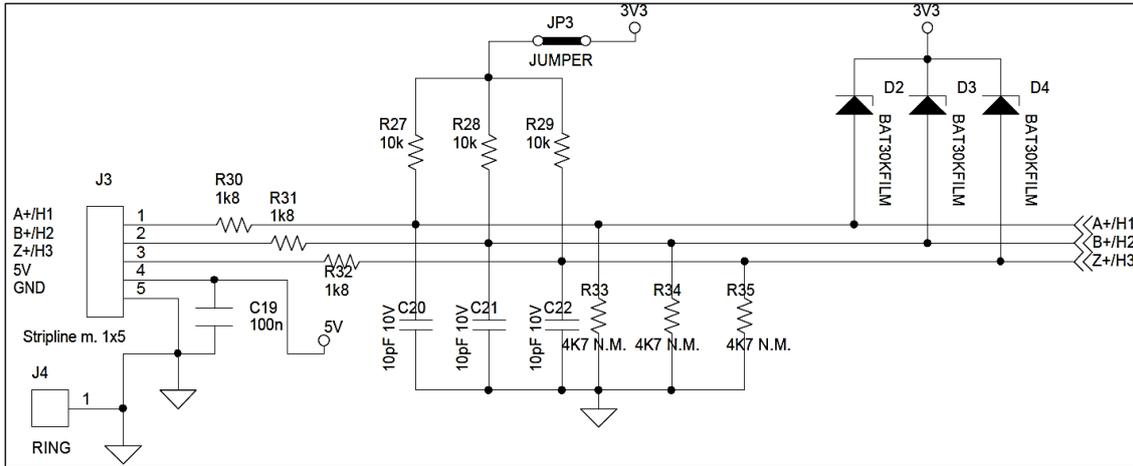


Figura 3.7: circuito di condizionamento per i segnali provenienti dai sensori ad effetto Hall

Il condizionamento consiste nel ridurre il rumore sulle linee con dei filtri passa basso passivi e nell'assicurarsi che i valori di tensione non fuoriescano dal range accettato dagli input del microcontrollore. Inoltre, chiudendo il jumper JP_3 è possibile introdurre un pull-up di tensione sulle linee, in modo da facilitare la distinzione tra "1" logico e "0" logico. Superato il blocco di condizionamento, i tre segnali H₁, H₂ e H₃ vengono campionati dall'ADC e impiegati dall'algoritmo nel microcontrollore.

3.3 Sezione di potenza

Si procede ora con la descrizione della sezione circuitale che comprende la sorgente di alimentazione del motore e la modalità con cui l'energia elettrica viene trasferita nello statore di quest'ultimo.

Come riportato nella scheda tecnica, il motore necessita di una differenza di potenziale ai capi dei terminali pari a 24 V e può assorbire una corrente media non superiore a 1.5 Ampere. Utilizzeremo dunque un alimentatore AC/DC portatile (simile a quelli dei personal computer) con delle caratteristiche compatibili. In figura 3.8 sono illustrate le sue caratteristiche.

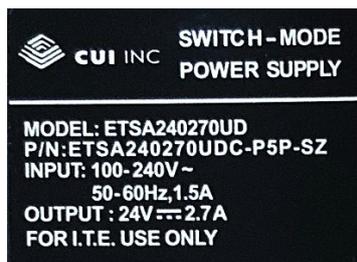


Figura 3.8: specifiche dell'alimentatore impiegato nel progetto

I due poli del generatore di tensione vengono poi collegati alla scheda X-Nucleo-IHM07M1 mediante la morsetteria J_1 (riquadro 1 in figura 3.9).

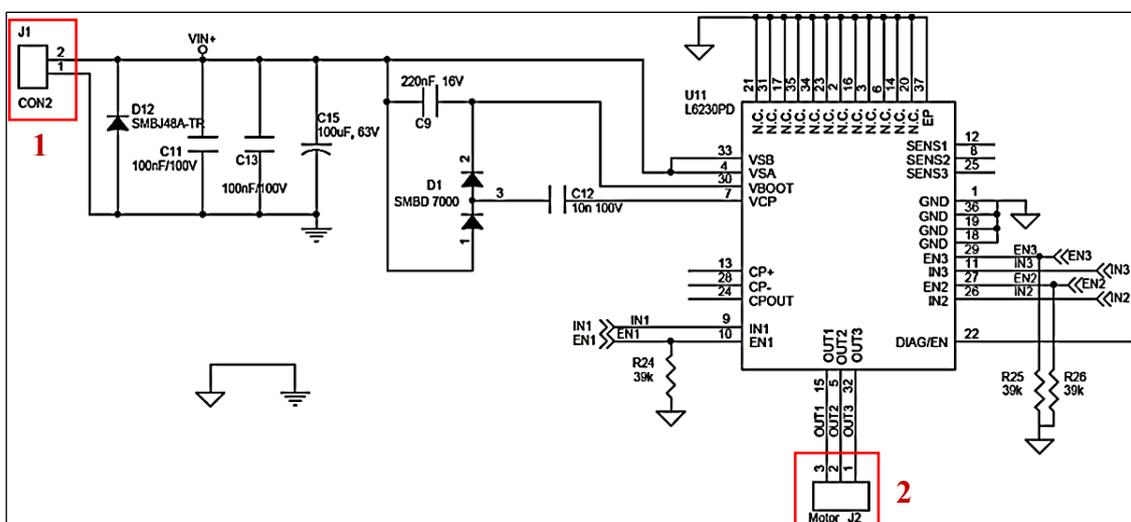


Figura 3.9: sezione di potenza della scheda X-Nucleo-IHM07M

La tensione d'ingresso viene filtrata da eventuali disturbi mediante i condensatori C_{11} , C_{13} , C_{15} ; mentre il diodo D_{12} previene i guasti dovuti a un errato collegamento della sorgente di alimentazione (per esempio l'inversione dei poli positivo e negativo). Le tre fasi del motore si collegano alla scheda driver grazie alla morsetteria J_2 (riquadro 2 in figura 3.9).

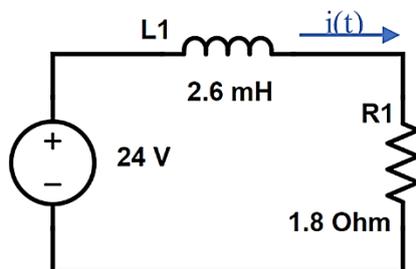
La figura 3.10 è una fotografia dei collegamenti sulle morsettiere della scheda driver. Si può distinguere a sinistra la morsetteria con i poli dell'alimentazione, al centro la morsetteria con le tre fasi del motore (cavi giallo, rosso e nero) e infine a destra i fili provenienti dai sensori ad effetto Hall del motore.



Figura 3.10: foto dei collegamenti nella sezione di potenza

Infine, l'energia elettrica viene distribuita negli avvolgimenti del motore mediante gli inverter dell'integrato L6230, utilizzando la tecnica Pulse-Width Modulation: regolando il duty-cycle dell'onda quadra si modifica in proporzione la corrente media che scorre nel circuito. Si parla di corrente *media* perché in realtà la corrente istantanea $i(t)$ ha l'andamento triangolare illustrato nella sezione 3.1 dedicata al controllo di corrente.

Il ripple di corrente negli avvolgimenti, ovvero la distanza tra il picco massimo e quello minimo di $i(t)$ in condizioni di regime, è dettata dalla frequenza dell'onda rettangolare. Consideriamo il circuito equivalente:



rappresentante la polarizzazione di due fasi del motore durante il periodo di ON del segnale PWM. I valori di L_I e R_I sono ricavati dalla scheda tecnica del motore: L_I rappresenta l'induttanza complessiva della serie di due fasi dello statore, mentre R_I la resistenza. Vogliamo calcolare il ripple di corrente in base alla durata di T_{ON} :

$$i_{ripple} = \frac{1}{L} \int_0^{T_{ON}} v_L(t) dt \quad (6)$$

nella (6) v_L è costante e uguale alla tensione di alimentazione V_S , quindi:

$$i_{ripple} = \frac{1}{L} \int_0^{T_{ON}} V_S dt = \frac{V_S}{L} \cdot T_{ON} \quad (7)$$

Come ci aspettavamo, dalla (7) risulta che tra la corrente e il tempo vige una relazione di linearità. Considerando per semplicità il caso specifico in cui il duty cycle è pari al 50%, e dato il periodo T_{sw} del segnale pwm, la corrente $i(t)$ impiega un tempo $T_{ON} = \frac{T_{sw}}{2}$ per crescere di una quantità pari a i_{ripple} , per poi decrescere della medesima quantità in un secondo periodo di tempo consecutivo avente la medesima durata $T_{OFF} = T_{ON}$.

Dunque, maggiore è la frequenza di switching degli inverter $f_{sw} = \frac{1}{T_{sw}}$, corrispondente alla frequenza del segnale PWM, minore sarà il ripple di corrente negli avvolgimenti del motore. Il ripple di corrente provoca un peggioramento delle prestazioni del sistema, introducendo un disturbo ad alta frequenza sulla grandezza di uscita, ovvero la velocità del motore.

Oltre a ciò, si aggiungono vibrazioni indesiderate delle componenti meccaniche mobili collegate al rotore e maggiori disturbi elettromagnetici emessi nell'ambiente circostante.

Per questa serie di motivi sceglieremo come parametro di progetto una frequenza di switching della PWM il più alta possibile, prendendo come limite superiore il massimo valore suggerito dalle specifiche tecniche dell'integrato L6230. Tale valore corrisponde a una f_{sw} pari a 100 kHz.

Con questo valore del periodo di switching, il ripple di corrente ammonta a (ricordando che $T_{ON} = \frac{T_{sw}}{2} = \frac{1}{2f_{sw}}$):

$$i_{ripple} = \frac{V_S}{2f_{sw}L} = \frac{24}{2 \cdot 10^5 \cdot 2.6 \cdot 10^{-3}} = 46.15 \text{ mA} \quad (8)$$

Considerando che le correnti negli avvolgimenti durante il funzionamento del motore variano in un range compreso tra i 300 e i 1500 mA, il ripple di corrente ammonta nel caso peggiore a circa il 15% della corrente di riferimento (quando il motore è alla minima velocità), il che costituisce un contributo sufficientemente ridotto per lo scopo di questo progetto.

In figura 3.11 è riportata una schermata dell'oscilloscopio nella quale si può osservare dettagliatamente il ripple di corrente in uno degli avvolgimenti.

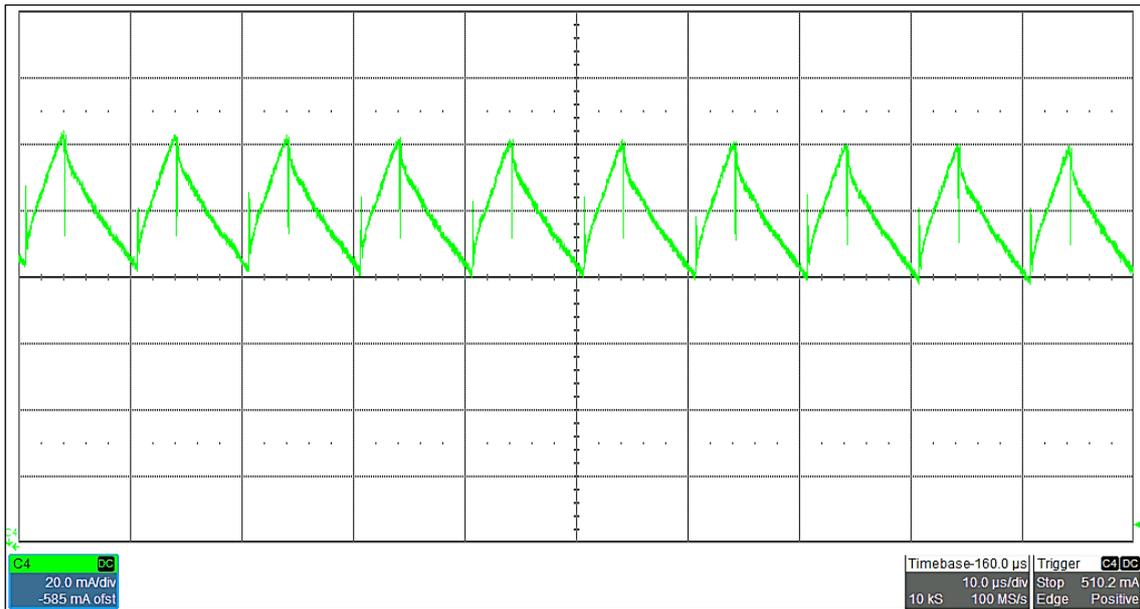


Figura 3.11: schermata dell'oscilloscopio con il ripple di corrente in uno degli avvolgimenti

La larghezza dei triangoli è pari a 10 microsecondi e corrisponde al periodo di switching degli inverter, deciso arbitrariamente per ottenere un ripple non superiore ai 50 mA; l'altezza dei triangoli come anticipato dai calcoli ammonta infatti a circa 40 mA.

Capitolo 4

Configurazione iniziale del μC

In questo capitolo verrà illustrata la fase di configurazione iniziale del microcontrollore con le sue periferiche, effettuata con il software STM32CubeMX.

4.1 Panoramica dei collegamenti

Prima di iniziare con la procedura di configurazione, è opportuno fare un elenco delle porte di input e output della scheda X-Nucleo-IHM07M1 attraverso le quali potremo inviare e ricevere segnali con il microcontrollore. Ricordiamo che la scheda driver è collegata alla scheda Nucleo mediante i connettori ST Morpho; quindi, non siamo liberi di effettuare i collegamenti con il micro a nostro piacimento. I connettori effettuano un collegamento diretto tra i pin del microcontrollore e i vari terminali dei circuiti elettrici insiti nella scheda driver (vedi capitolo 3).

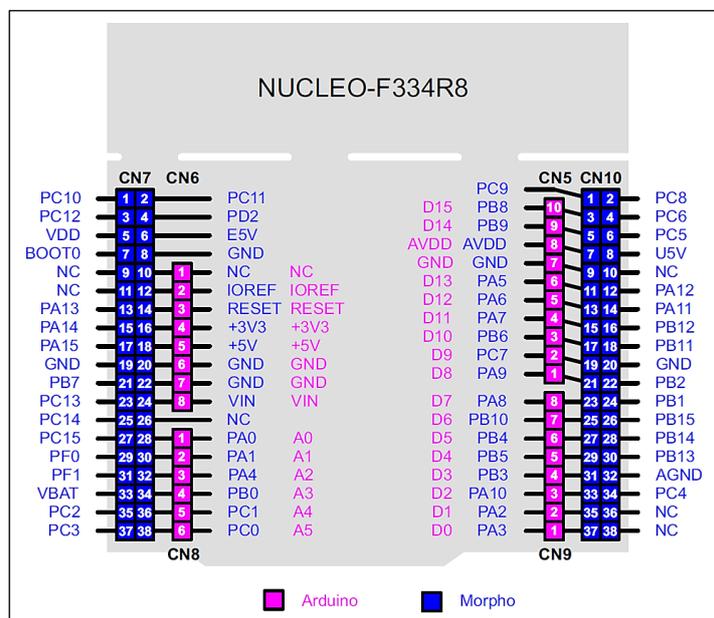


Figura 4.1: pinout dei connettori Morpho (in blu) presenti sulla scheda Nucleo

Nel datasheet di X-Nucleo-IHM07M1 si trova una tabella che illustra tali corrispondenze, riassunte nella tabella 4.1.

Segnale scheda driver	Descrizione	Pin μC
EN_1	Abilitazione canale 1	PC10
EN_2	Abilitazione canale 2	PC11
EN_3	Abilitazione canale 3	PC12
IN_1	PWM canale 1	PA8
IN_2	PWM canale 2	PA9
IN_3	PWM canale 3	PA10
H_1	Sensore Hall H_1	PC4
H_2	Sensore Hall H_2	PC5
H_3	Sensore Hall H_3	PB12
$\text{V}_{\text{CP-}}$	Riferimento di corrente	PA4
ETR	Segnale di overcurrent	PA12
User_button	Start/Stop	PC13
ω_{ref}	Riferimento di velocità	PB1

Tabella 4.1: legenda dei collegamenti tra pin del microcontrollore e segnali della scheda driver X-Nucleo-IHM07M1

Le ultime due righe della tabella 4.1 riportano due segnali non ancora visti: *User_button* è un trigger provocato dalla pressione di un pulsante presente sulla scheda Nucleo e utilizzato per far partire o fermare il funzionamento dell'intero sistema; ω_{ref} invece è un segnale di tensione analogico variabile per mezzo di un potenziometro (situato sulla scheda driver) che verrà impiegato per modificare il riferimento di velocità del motore.

Ora che è chiaro quali dei pin del microcontrollore andremo ad utilizzare, e con quale scopo, si può procedere con la sua configurazione.

4.2 Clock

Nella sezione *RCC*, si imposta come sorgente di clock il cristallo esterno di cui è dotata la scheda Nucleo, risuonante a una frequenza di 8 MHz. Poi, in *Clock Configuration* si imposta il moltiplicatore di frequenza a "x9" ottenendo una frequenza di $8 \times 9 = 72 \text{ MHz}$ nel core, nei bus e nelle periferiche.

Con l'attuale configurazione il convertitore analogico-digitale (ADC) è in grado di campionare segnali con un periodo $T_s = \frac{1}{72 \text{ MHz}} = 13.9 \text{ ns}$; tuttavia si tratta di un tempo inutilmente piccolo; infatti, anche girando alla massima velocità i sensori ad effetto Hall cambiano valore in un tempo circa pari a:

$$t_{step,min} = \frac{60}{6P \cdot \omega_{max} [rpm]} \cong \frac{60}{6 \cdot 4 \cdot 5000} = 500 \mu s, \quad (9)$$

dove ω_{max} è la massima velocità angolare del motore espressa in rpm (5000 giri al minuto è un valore sovrastimato), mentre P è il numero di coppie di poli del motore (4 in questo caso); il risultato $t_{step,min}$ rappresenta il tempo che il rotore impiega per fare un passo quando sta girando alla massima velocità (per fare un giro completo occorrono $6P = 24$ passi).

Dunque risulterebbe necessario verificare lo stato dei sensori ad effetto Hall con una frequenza al più pari a $\frac{1}{t_{step,min}} = \frac{1}{500 \mu s} = 2 \text{ kHz}$. Tuttavia potrebbe risultare rischioso affidarsi ad un solo sample, dunque optiamo per un divisore di frequenza pari a sei, riducendo la frequenza di campionamento dell'ADC a $\frac{72 \text{ MHz}}{6} = 12 \text{ MegaHertz}$. Successivamente durante la configurazione dei convertitori sarà possibile ridurre ulteriormente tale frequenza.

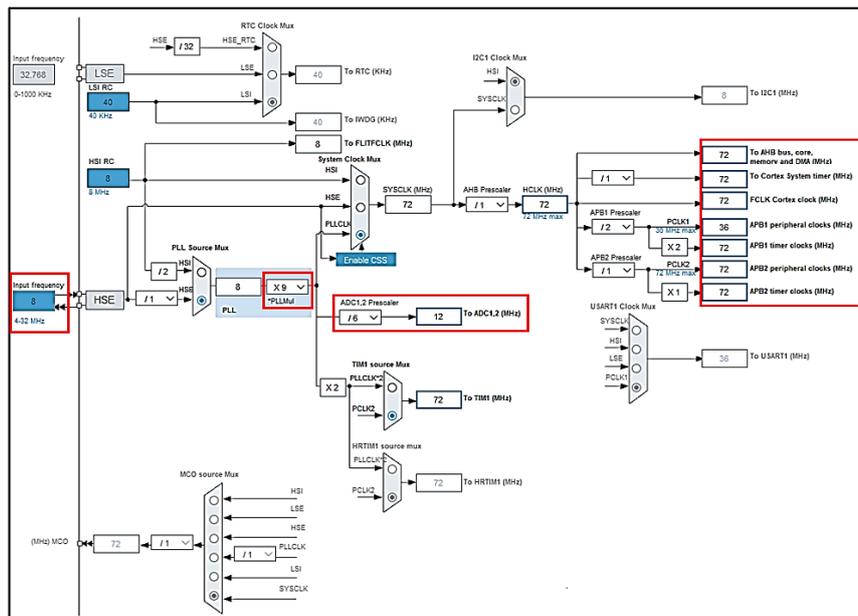


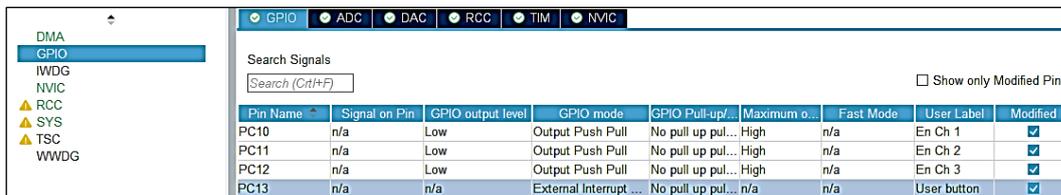
Figura 4.2: schermata di configurazione del clock del μC su STM32CubeMX

4.3 General Purpose IO

I pin impostati come General Purpose Input Output consistono in pin configurabili via software per funzionare come pin di input oppure di output.

Si configurano i pin PC10, PC11 e PC12 come pin di output, necessari per abilitare il funzionamento degli half-bridge. Tramite una riga di codice, sarà possibile settare l'uscita del pin a "1" e quindi permettere il funzionamento della rispettiva gamba dell'inverter, oppure resettare l'uscita e spegnere l'half-bridge.

Il quarto pin GPIO è lo User_button. Questo pin (PC13) ha il compito di rilevare se il pulsante è stato premuto: quando ciò accade viene prodotto un gradino di tensione positivo sulla linea, mentre quando viene rilasciato il segnale torna a zero. Scegliendo come opzione "*External interrupt mode with rising edge trigger detection*" si imposta PC13 in modo tale che quando si preme il pulsante viene rilevato solo il fronte di salita di tensione sulla linea.



The screenshot shows the STM32CubeMX interface with the GPIO configuration tab selected. A table lists the configurations for pins PC10, PC11, PC12, and PC13. The table has columns for Pin Name, Signal on Pin, GPIO output level, GPIO mode, GPIO Pull-up/..., Maximum o..., Fast Mode, User Label, and Modified.

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/...	Maximum o...	Fast Mode	User Label	Modified
PC10	n/a	Low	Output Push Pull	No pull up pul...	High	n/a	En Ch 1	<input checked="" type="checkbox"/>
PC11	n/a	Low	Output Push Pull	No pull up pul...	High	n/a	En Ch 2	<input checked="" type="checkbox"/>
PC12	n/a	Low	Output Push Pull	No pull up pul...	High	n/a	En Ch 3	<input checked="" type="checkbox"/>
PC13	n/a	n/a	External Interrupt ...	No pull up pul...	n/a	n/a	User button	<input checked="" type="checkbox"/>

Figura 4.3: impostazioni dei pin GPIO

4.4 Convertitori A/D e D/A

Il microcontrollore STM32F334R8 dispone di due convertitori da analogico a digitale e altri due convertitori da digitale ad analogico. In questo progetto si utilizzerà un convertitore A/D per campionare il segnale analogico ω_{ref} impostato mediante il potenziometro; mentre un secondo ADC servirà per acquisire i segnali d'uscita dei sensori ad effetto Hall. Inoltre, come già anticipato nella sezione 3.1 relativa al controllo di corrente, occorre un convertitore D/A per generare un segnale analogico di tensione che funge da riferimento di corrente.

4.4.1 ADC1 e ADC2

In figura 4.4 sono riportate le impostazioni di configurazione dell'ADC1 per l'acquisizione del segnale ω_{ref} . Il segnale ω_{ref} è presente sul pin PB1 del microcontrollore, corrispondente al canale IN12. Dal momento che il riferimento allo zero volt è il medesimo, si utilizza la modalità *single-ended*.

Si imposta una risoluzione di 10 bit con allineamento a destra; dunque, il valore di tensione analogica viene quantizzato in 1024 valori discreti. Sono abilitati *Continuous Conversion Mode* e *DMA Continuous Requests* e alla voce "External Trigger Conversion Source" si sceglie *Regular Conversion launched by software*: in questo modo l'ADC campiona il segnale in completa autonomia alla frequenza impostata senza generare interrupt, trasferendo i dati direttamente in memoria per mezzo del DMA (vedi impostazioni del DMA alla sottosezione 4.4.2).

ADC1 Mode and Configuration	
Mode	
IN11 Disable	▼
IN12 IN12 Single-ended	▼
Configuration	
Reset Configuration	
NVIC Settings DMA Settings GPIO Settings	
Parameter Settings User Constants	
Search (Ctrl+F)	
ADCs_Common_Settings	Mode: Independent mode
ADC_Settings	Clock Prescaler: ADC Asynchronous clock mode
	Resolution: ADC 10-bit resolution
	Data Alignment: Right alignment
	Scan Conversion Mode: Disabled
	Continuous Conversion Mode: Enabled
	Discontinuous Conversion Mode: Disabled
	DMA Continuous Requests: Enabled
	End Of Conversion Selection: End of single conversion
	Overrun behaviour: Overrun data overwritten
	Low Power Auto Wait: Disabled
ADC_Regular_ConversionMode	Enable Regular Conversions: Enable
	Number Of Conversion: 1
	External Trigger Conversion Source: Regular Conversion launched by software
	External Trigger Conversion Edge: None
	SequencerNbRanks: 1
Rank	1
	Channel: Channel 12
	Sampling Time: 601.5 Cycles
	Offset Number: No offset
	Offset: 0
ADC_Injected_ConversionMode	Enable Injected Conversions: Disable

Figura 4.4: impostazioni ADC1

Infine, si imposta il *sampling time* a *601.5 cycles*, permettendo all'ADC di produrre un campione ogni 601.5 cicli di clock. Ricordiamo che la frequenza di clock degli ADC è di 12 MHz; dunque, la nuova frequenza di campionamento diventa:

$$f_{ADC,1} = \frac{12 \text{ MHz}}{601.5} \cong 20 \text{ kHz} \quad (10)$$

Dal momento che l'algoritmo richiede un campione ogni millisecondo, si tratta di una frequenza adatta allo scopo.

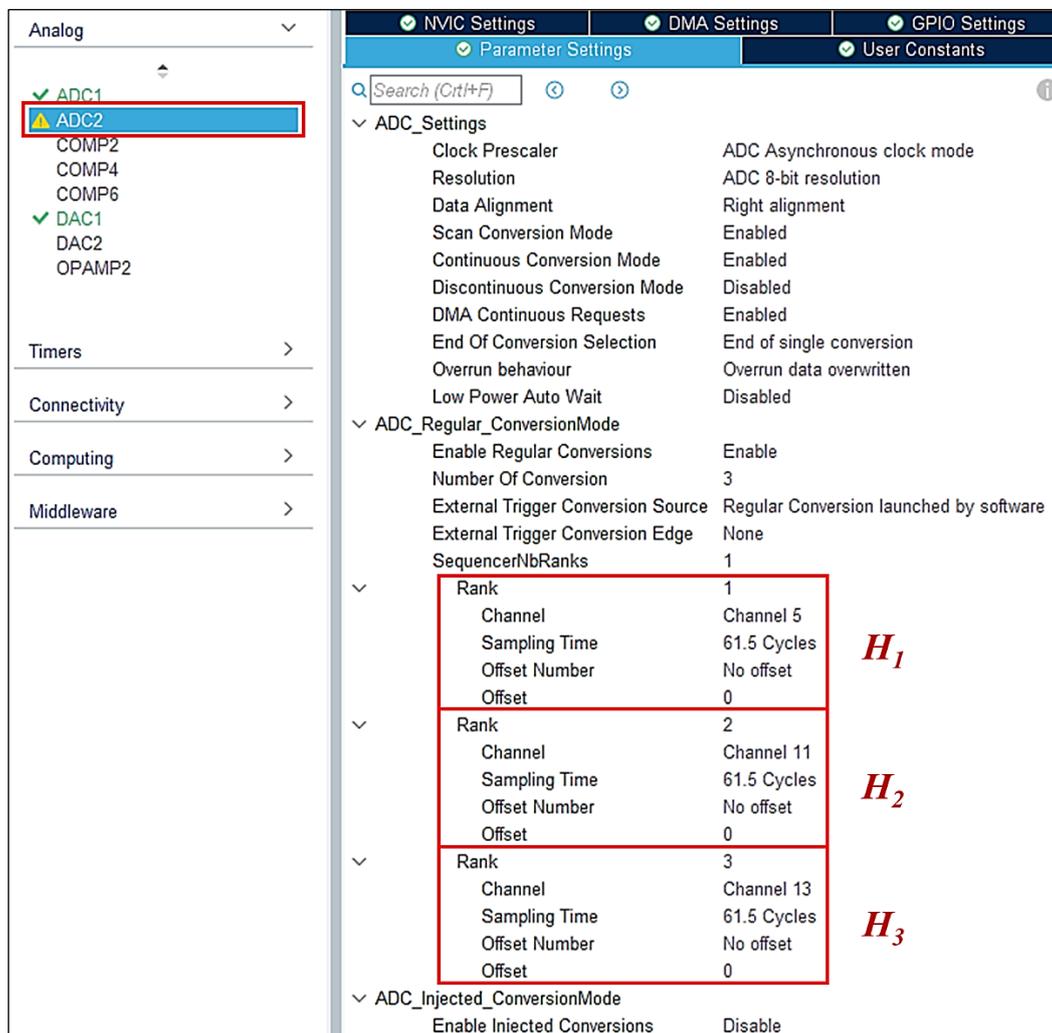


Figura 4.5: impostazioni ADC2

In figura 4.5 sono riportati le impostazioni di configurazione dell'ADC2 per l'acquisizione dei segnali H_1 , H_2 , H_3 . Essi sono presenti rispettivamente sui pin PC4, PC5 e PB12 corrispondenti ai canali 5, 11 e 13 dell'ADC2, impostati in modalità single-ended.

Impostiamo la risoluzione minima possibile, ovvero 8 bit, dal momento che si deve solo distinguere il livello logico “basso” da quello “alto”. Come per l’ADC1 si abilitano *Continuous Conversion Mode* e *DMA Continuous Requests* e si seleziona *Regular Conversion launched by software* come sorgente di trigger per la conversione; la *Scan Conversion Mode* si abilita in automatico quando sono utilizzati più canali in contemporanea nel medesimo convertitore.

Infine, si riduce la frequenza di campionamento impostando un sampling time pari a 61.5 cicli di clock, ciò significa che la nuova frequenza è pari a:

$$f_{ADC,2} = \frac{12 \text{ MHz}}{61.5} \cong 200 \text{ kHz} \quad (11)$$

ciò significa che si ottiene un nuovo campione di H_1 , H_2 e H_3 ogni 5 microsecondi.

4.4.2 Direct Memory Access

Il Direct Memory Access (DMA) permette il salvataggio dei dati prodotti dagli ADC direttamente in memoria senza necessità di intervento da parte della CPU.

Dal momento che ADC2 produce parole lunghe 8 bit, si imposta nelle sue *DMA Request Settings* la larghezza dei dati pari a un byte; al contrario per l’ADC1 che produce parole di 10 bit sono necessari due byte per contenere i dati, dunque, si imposta la larghezza di una half-word (16 bit).

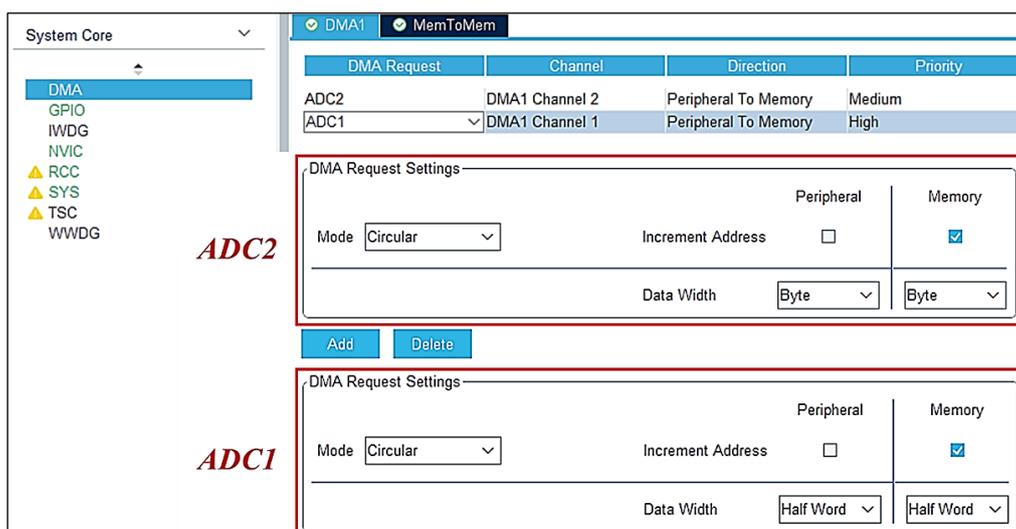


Figura 4.6: impostazioni del DMA per ADC1 e ADC2

È importante anche ricordarsi di settare il *Mode* su *Circular*, in modo che una volta riempito il vettore in memoria con tutte le parole acquisite in un istante di campionamento, al prossimo istante il puntatore ritorni all'inizio del vettore, sovrascrivendo i dati vecchi.

4.4.3 DAC

In figura 4.7 sono riportate le impostazioni per il DAC1.

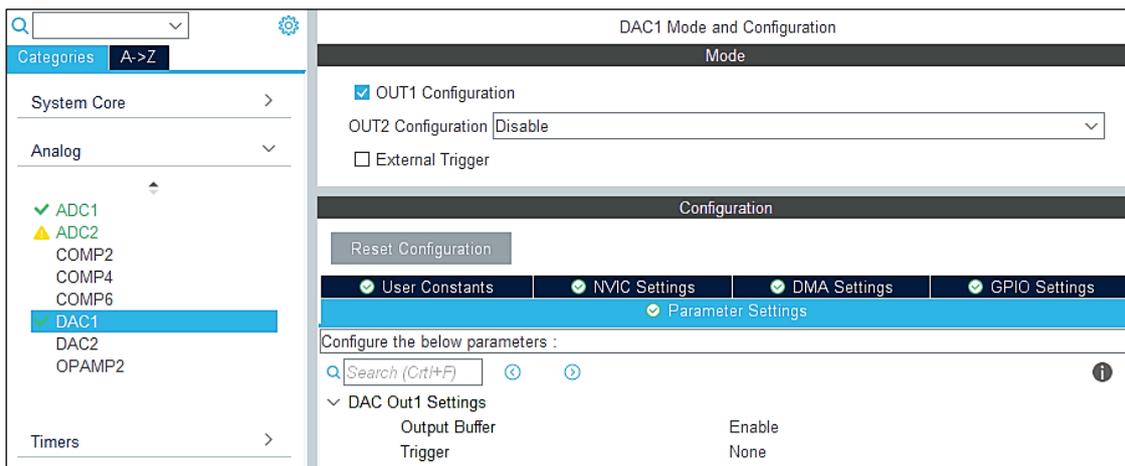


Figura 4.7: impostazioni del DAC1

Si abilita l'output numero 1 e il buffer in uscita; non è necessario alcun tipo di trigger per la conversione.

4.5 Timer

Per lo sviluppo dell'applicazione verranno utilizzati tre timer indipendenti tra loro: un timer si occupa di generare i segnali PWM su tre differenti canali per pilotare le gambe dell'inverter, gli altri due invece lavorano in modalità *Output Compare* per generare degli interrupt a cadenza regolare. Questi interrupt sono fondamentali perché fungono da base dei tempi per gli algoritmi implementati nel microcontrollore.

4.5.1 Generatore PWM

Per generare i segnali PWM si utilizzerà il TIM1. In figura 4.8 e 4.9 sono visibili i parametri di configurazione.

Come *Trigger source* si sceglie *Internal Clock*, in questo modo viene utilizzato il clock del microcontrollore come riferimento dello scorrere del tempo; si impostano i primi tre canali come sorgenti dei segnali pwm IN_1 , IN_2 , IN_3 corrispondenti rispettivamente ai pin PA8, PA9 e PA10.

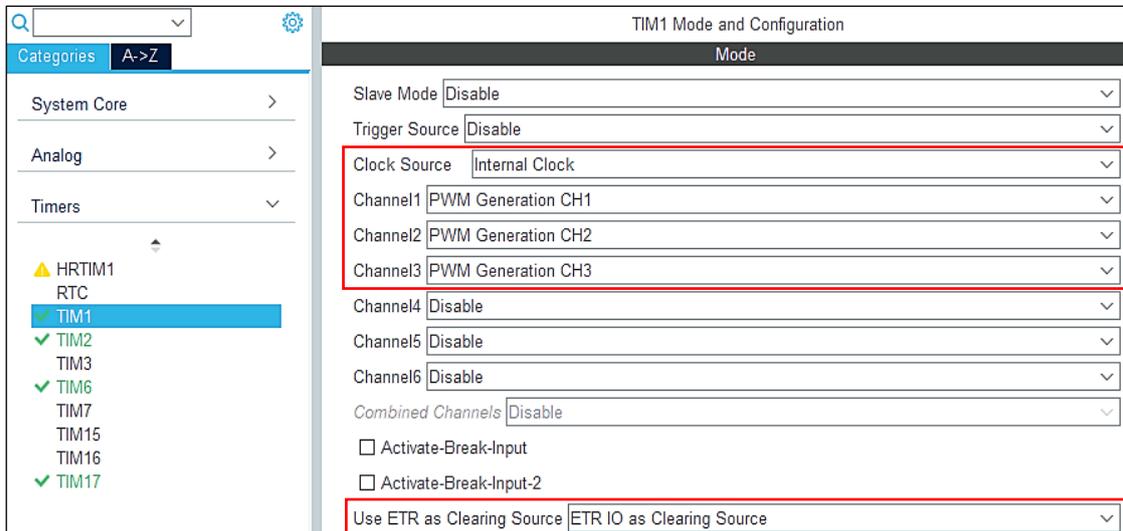


Figura 4.8: configurazione TIM1, sezione “Mode”

Successivamente occorre impostare il trigger esterno affinché ponga a zero le uscite quando si verifica l’evento di sovracorrente, ovvero quando il segnale ETR assume il livello logico alto; per fare ciò si seleziona l’opzione “*ETR IO as Clearing Source*” in corrispondenza della voce “*Use ETR as Clearing Source*”, il pin PA12 viene settato automaticamente come sorgente di trigger esterno per TIM1.

Per impostare la frequenza delle PWM pari a 100 kHz occorre calcolare quanti incrementi del counter occorrono per ottenere un tempo pari a 10 μ s, sapendo che la frequenza del counter è di 72 MHz (non si inserisce alcun prescaler).

$$N = 72 \text{ MHz} \times 10 \mu\text{s} = 720 \quad (12)$$

720 è quindi il valore da inserire nel campo “*Counter Period*”.

Il timer 1 non genera alcun interrupt. Si abilita il *Clear Input Source* per tutti e tre i canali.

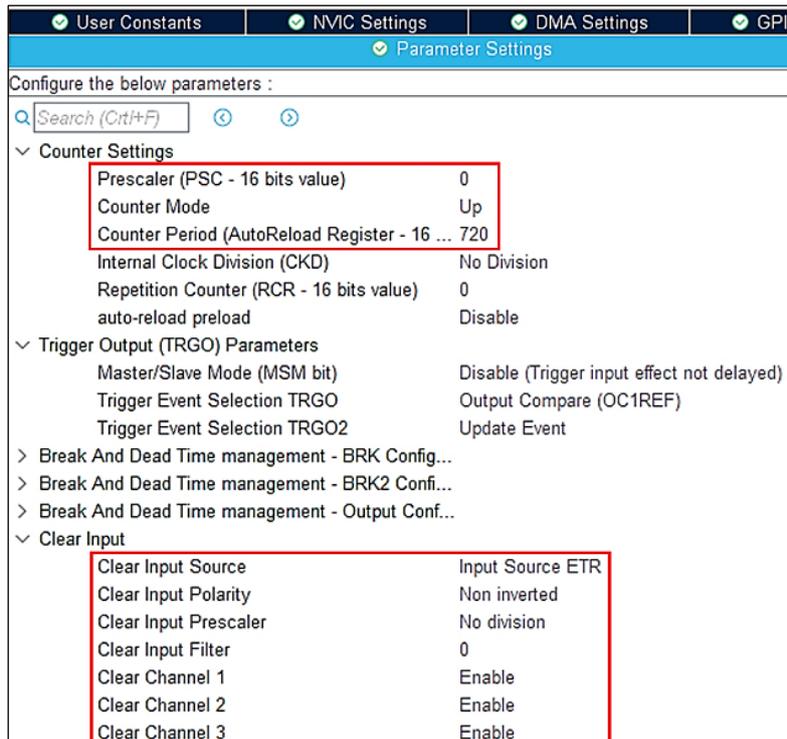


Figura 4.9: impostazioni TIM1, sezione “Configuration”

4.5.2 Timer in modalità Output Compare

Affinché l’algoritmo eseguito dal microcontrollore possa funzionare, è necessario che si eseguano determinate azioni con cadenza regolare. A questo proposito sono stati configurati due timers appositamente per misurare il trascorrere del tempo e generare un interrupt quando si verifica la condizione impostata.

I timer utilizzati sono il TIM2 e il TIM6.

Il primo ha il compito di generare regolarmente un interrupt con un periodo di 50 μ s. In figura 4.10 sono riportate le sue impostazioni di iniziali.

La modalità di funzionamento scelta è *Output Compare No Output*, ovvero quando il registro *Counter* raggiunge il valore contenuto nell’*AutoReload Register* viene generato un interrupt. Nessun segnale è presente in uscita dal microcontrollore, tutto viene gestito via software.

Si imposta il prescaler a 71, in questo modo essendo la frequenza di partenza 72 MHz si ottiene un incremento del registro *Counter* ogni 1 μ s. A questo punto, per arrivare a 50 μ s è sufficiente inserire “50” nel campo *Counter Period*.

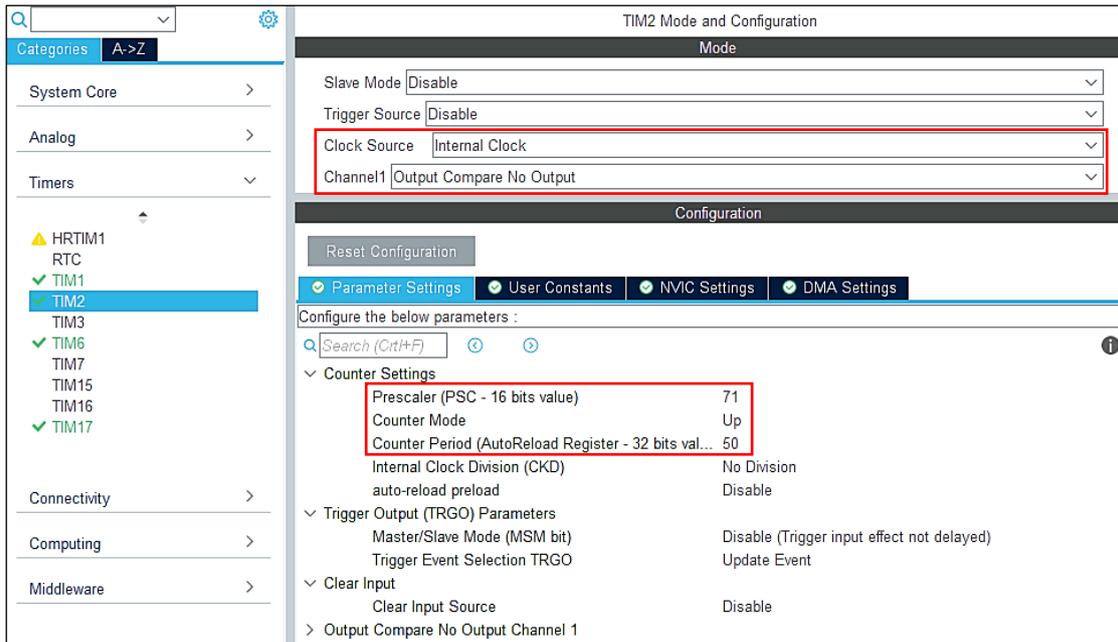


Figura 4.10: impostazioni TIM2

Il TIM6 funziona allo stesso modo, ma la frequenza della sua routine è molto più lenta: 1 millisecondo.

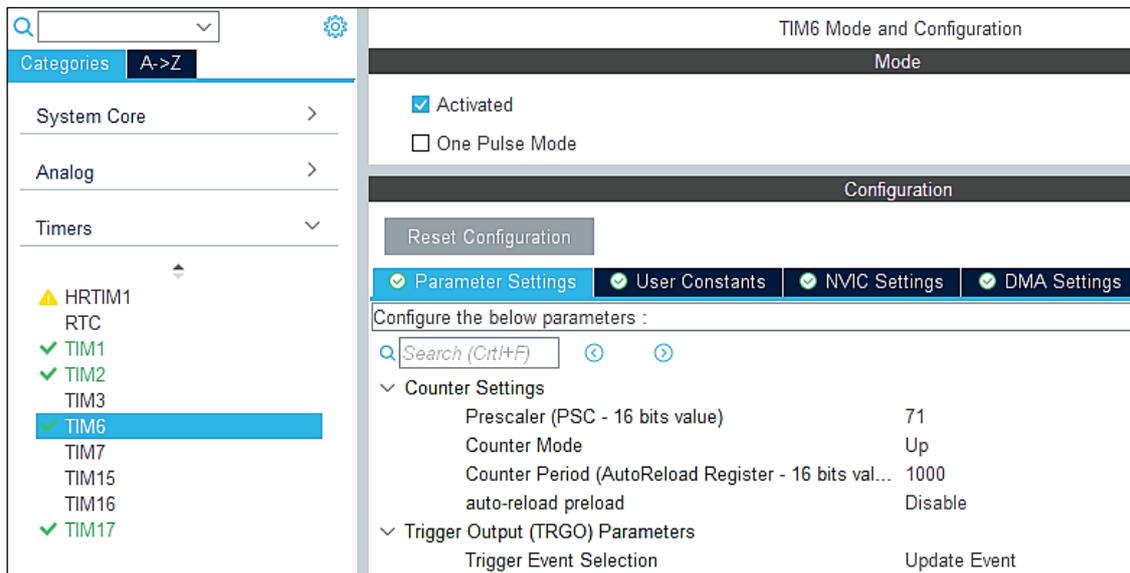


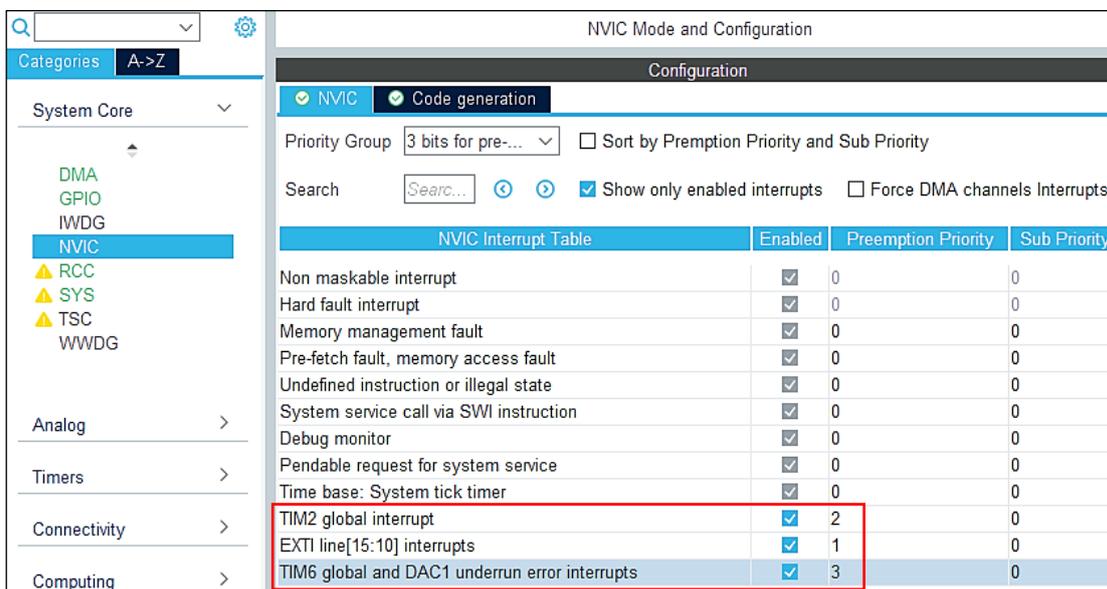
Figura 4.11: impostazioni TIM6

4.6 Interrupt

Nella sezione NVIC (Nested Vectored Interrupt Control) è possibile abilitare o disabilitare le Interrupt Service Routine (ISR) e dar loro una priorità. Se durante l'esecuzione di una ISR si verifica un interrupt con priorità maggiore, il microcontrollore è in grado di sospendere ciò che sta facendo ed eseguire la routine con più importanza, ed una volta che essa è terminata riprendere l'esecuzione di quella precedentemente interrotta.

Le ISR abilitate sono le seguenti:

- Priorità **zero** (massima priorità): si tratta delle ISR di sistema per la gestione di eventuali errori o situazioni indesiderate. Si consiglia di non disattivarle per evitare di compromettere il corretto funzionamento del dispositivo.
- Priorità **uno**: ISR relativa al pulsante di start/stop, si tratta di una routine non eseguita periodicamente.
- Priorità **due**: ISR relativa all'interrupt di TIM2.
- Priorità **tre**: ISR relativa all'interrupt di TIM6.



NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
TIM2 global interrupt	<input checked="" type="checkbox"/>	2	0
EXTI line[15:10] interrupts	<input checked="" type="checkbox"/>	1	0
TIM6 global and DAC1 underrun error interrupts	<input checked="" type="checkbox"/>	3	0

Figura 4.12: sezione di gestione degli interrupt in STM32CubeMX

Gli interrupt di TIM2 e TIM6 si verificano periodicamente, dunque è necessario calcolare la percentuale di tempo che la CPU spende per servire le interruzioni e se sono soddisfatte le condizioni di intervallo per ciascuna di esse.

Con l'ausilio di un pin di flag del microcontrollore e dell'oscilloscopio si misura la durata di ciascuna ISR; chiamiamo $T_{p,1}$ e $T_{d,1}$ rispettivamente il periodo e la durata della ISR di TIM2 (che ha priorità maggiore) e $T_{p,2}$, $T_{d,2}$ periodo e durata della ISR di TIM6.

Dalle misure risulta che:

- $T_{p,1} = 50 \mu s$, $T_{d,1} = 6 \mu s$;
- $T_{p,2} = 1000 \mu s$, $T_{d,2} = 70 \mu s$;

La percentuale di impiego della CPU si calcola nel seguente modo:

$$\frac{T_{d,1}}{T_{p,1}} + \frac{T_{d,2}}{T_{p,2}} = \frac{6}{50} + \frac{70}{1000} = 0.19 \rightarrow 19\% \quad (13)$$

Dalla (12) risulta quindi che circa un quinto del tempo totale di esecuzione viene speso dal micro nell'elaborazione delle richieste di interruzione. È importante che la percentuale non cresca mai oltre il 100%.

Verifichiamo ora le condizioni di intervallo, ricordando che il microcontrollore supporta il *nesting* delle interruzioni. Per la ISR di TIM2, avente priorità più alta, si deve avere che

$$T_{d,1} < T_{p,1} \rightarrow 6 \mu s < 50 \mu s \quad (14)$$

La condizione di TIM2 è verificata. Verifichiamo anche la condizione di TIM6, la quale ha priorità inferiore rispetto a TIM2

$$T_{d,2} + T_{d,1} \cdot \frac{T_{p,2}}{T_{p,1}} < T_{p,2} \rightarrow 70 \mu s + 6 \cdot \frac{1000}{50} \mu s = 190 \mu s < 1000 \mu s \quad (15)$$

Anche la condizione in (15) è verificata.

Effettuando questi controlli, si è accertato che il sistema è in grado di elaborare tutte le richieste di interruzione senza il rischio di saltarne alcuna.

Capitolo 5

Modellizzazione del sistema in catena chiusa

In questo capitolo si vuole dare uno sguardo d'insieme all'intero sistema e spiegare come sono stati modellizzati e progettati i principali blocchi che compongono lo schema di figura 5.1.

Lo studio prevede un'analisi del sistema nel dominio della frequenza, adottando le tecniche studiate nei corsi di studio di Controlli Automatici ed Elettronica Industriale per risolvere il *problema di controllo*.

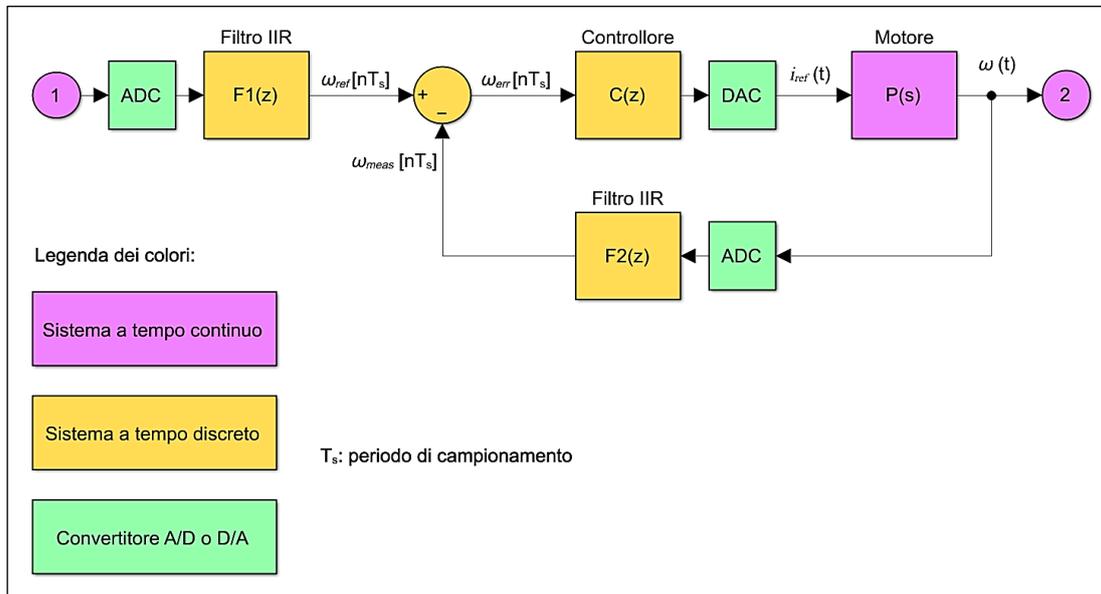


Figura 5.1: schema a blocchi del sistema in catena chiusa

Nello schema a blocchi in figura 5.1 si possono distinguere quattro blocchi fondamentali:

- il blocco C , corrispondente al regolatore digitale;
- il blocco P , che rappresenta una modellizzazione basilare del motore, ovvero il processo che si vuole controllare;
- due filtri digitali F_1 e F_2 , necessari per rimuovere il rumore indesiderato dal segnale di riferimento ω_{ref} e dalla velocità misurata ω_{meas} .

Si può notare che i blocchi in figura 5.1 hanno differenti colori:

- i blocchi viola indicano sistemi a tempo continuo: per descriverne il modello nel dominio della frequenza si deve utilizzare la trasformata di Laplace. Di fatto l'unico sistema rilevante di questo tipo è il motore.
- I blocchi arancioni indicano sistemi a tempo discreto, dunque saranno espressi nel dominio della trasformata *Zeta*. Tali sistemi non sono fisici bensì virtuali, trattandosi di algoritmi implementati nel microcontrollore.
- Infine, i blocchi verdi si occupano di convertire i segnali analogici in segnali discreti (ADC) o viceversa (DAC), in modo da poter interfacciare tra di loro sistemi di tipologie differenti.

Il periodo di campionamento T_s relativo ai sistemi a tempo discreto (filtri e controllore), verrà trattato specificatamente nella sezione 5.3.

5.1 Analisi del motore nel dominio della frequenza

Il motore viene modellizzato come un sistema che varia la propria velocità a seconda di come varia la corrente nei suoi avvolgimenti. Dunque, esso ha come segnale di ingresso una corrente elettrica i_{ref} e come segnale d'uscita una velocità ω le cui unità di misura sono rispettivamente Ampere [A] e radianti al secondo [rad/s]. Il segnale i_{ref} coincide con il riferimento generato dal DAC per il controllo di corrente interno.

Per semplicità si è deciso di non considerare il controllo di corrente nel modello del sistema di figura 5.1 ma di considerarlo "implicito" nel motore; di conseguenza si presuppone che la corrente elettrica negli avvolgimenti ricalchi perfettamente il riferimento i_{ref} impostato, sebbene come si è visto nella sezione 3.1, non sia esattamente così. Dal momento che il ripple di corrente è basso (meno di 50 mA) rispetto a i_{ref} , si tratta di un'approssimazione accettabile per lo scopo del progetto.

L'equazione differenziale che descrive il motore e mette in relazione la velocità ω con la corrente elettrica i_{ref} è la seguente:

$$J \frac{d\omega(t)}{dt} + B\omega(t) = T \tag{16}$$

dove:

- J è il momento d'inerzia del rotore sommata a quella di un eventuale carico collegato;
- B è il coefficiente di attrito viscoso;
- T è la coppia generata dal dispositivo. on si tratta di un valore costante.

In questa equazione non compare la corrente, occorre infatti introdurre un'altra relazione:

$$T = k_t \cdot i_{ref} \quad (17)$$

dove k_t è la costante di coppia del motore. Ora, inserendo l'equazione (17) nella (16) si ottiene:

$$J \frac{d\omega(t)}{dt} + B\omega(t) = k_t i_{ref}(t) \quad (18)$$

Applichiamo la trasformata di Laplace all'equazione (18):

$$Js\Omega(s) + B\Omega(s) = k_t I_{ref}(s) \quad (19)$$

dove:

$$\Omega(s) = \mathcal{L}[\omega(t)] \quad (20)$$

$$I_{ref}(s) = \mathcal{L}[i_{ref}(t)] \quad (21)$$

Isoliamo $\Omega(s)$:

$$\Omega(s) = \frac{k_t}{Js + B} I_{ref}(s) \quad (22)$$

La (22) ci indica la relazione esistente tra ingresso e uscita nel dominio di Laplace. Dividendo l'uscita per l'ingresso si ottiene infine la funzione di trasferimento del motore:

$$P(s) = \frac{\Omega(s)}{I_{ref}(s)} = \frac{k_t}{Js + B} \quad (23)$$

I valori dei parametri sono elencati in tabella 5.1.

k_t	0.035	Nm/A
J	$46 \cdot 10^{-6}$	kg·m ²
B	$2 \cdot 10^{-4}$	Nm·s

Tabella 5.1: parametri del motore

I valori di k_t e J sono riportati nella scheda tecnica del motore, mentre il valore di B è stato determinato sperimentalmente misurando la corrente media negli avvolgimenti a differenti velocità. Moltiplicando la corrente misurata per la costante di coppia k_t si ottiene la coppia sviluppata dal motore, successivamente dividendo tale valore di coppia per la velocità misurata si ha il coefficiente di attrito viscoso.

$$T = k_t i = B \omega_{meas} \quad (24)$$

$$B = \frac{T}{\omega_{meas}} \quad (25)$$

Effettuando alcuni test a velocità differenti si sono ottenuti dei valori di B dell'ordine di 10^{-4} Nm·s, volendo prendere la mediana si è scelto il valore $2 \cdot 10^{-4}$ Nm·s.

Analizzando meglio la funzione di trasferimento $P(s)$ e portandola nella forma di Bode:

$$P(s) = \frac{k_t}{B} \cdot \frac{1}{1 + \frac{J}{B}s} = \frac{P_0}{1 + p_1 s} \quad (26)$$

abbiamo che

$$P_0 = |P(j\omega = 0)| = \frac{k_t}{B} = \frac{0.035}{0.0002} = 175 \text{ A}^{-1}\text{s}^{-1} \quad (27)$$

$$P_{0,dB} = 20 \log_{10} P_0 = 44.9 \text{ dB} \quad (28)$$

è il guadagno di $P(s)$ in banda passante. Inoltre, il denominatore tende a zero per s che tende a $-\frac{B}{J}$, dunque esiste un polo p_1 tale che

$$p_1 = -\frac{B}{J} = -\frac{2 \cdot 10^{-4}}{46 \cdot 10^{-6}} = -4.35 \text{ rad/s} \quad (29)$$

ovvero un polo reale negativo, quindi un polo stabile.

Riassumendo, $P(s)$ ha un comportamento di tipo passa basso con un guadagno in banda passante P_0 e pulsazione di taglio $\omega_t = |p_1| = 4.35$ rad/s.

Dai grafici di Bode in figura 5.2 possiamo stimare con facilità la pulsazione di attraversamento (o di crossover) e il margine di fase φ_P del sistema $P(s)$.

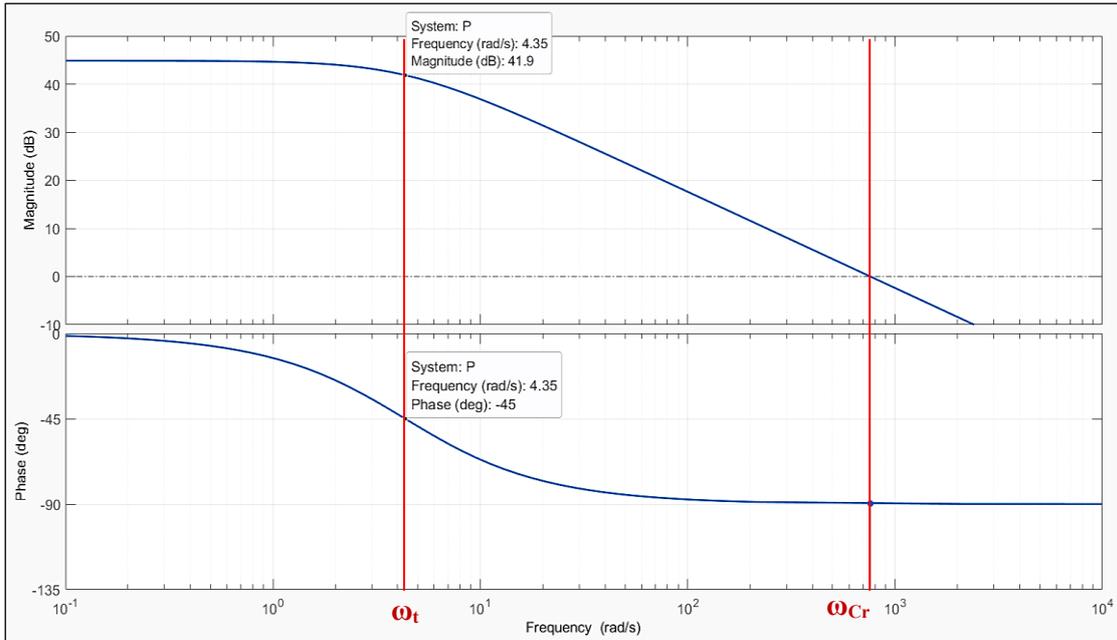


Figura 5.2: diagrammi di bode di modulo e fase relativi a $P(s)$, sono evidenziate la pulsazione di taglio ω_t e di crossover ω_{Cr}

Approssimativamente risulta:

$$\omega_{Cr} \cong 800 \text{ rad/s}$$

e

$$\varphi_P \cong -90^\circ$$

La pulsazione di attraversamento indica la velocità angolare alla quale il sistema ha guadagno unitario, ovvero per frequenze superiori esso perde la sua capacità di amplificare il segnale d'ingresso. Nel diagramma di Bode del modulo, essendo le ordinate espresse in dB, ciò si traduce nell'attraversamento dell'asse delle ascisse in corrispondenza degli zero dB. Il margine di fase è la fase del sistema in ω_{Cr} .

5.2 Periodo di campionamento

Quando viene definito un sistema a tempo discreto è indispensabile specificare anche un periodo di campionamento ad esso associato, ovvero la distanza temporale che intercorre tra due campioni adiacenti $x[k]$ e $x[k+1]$ del segnale d'ingresso x del sistema.

In termini pratici, dal momento che i sistemi discreti sono implementati tramite degli algoritmi eseguiti dal microcontrollore, il periodo di campionamento T_s corrisponde al

periodo di tempo che intercorre tra due esecuzioni successive dell'algoritmo corrispondente al sistema.

Nel progetto, gli algoritmi dei sistemi a tempo discreto F_1 , F_2 , C vengono eseguiti dalla routine di interrupt del timer 6, la quale viene eseguita periodicamente alla frequenza di 1 kHz.

Si può concludere dunque che la frequenza di campionamento vale $T_s = \frac{1}{f_s} = 1 \text{ ms}$.

5.3 Filtri digitali

Come anticipato, all'interno del sistema sono stati inseriti due filtri digitali con lo scopo di eliminare il rumore dal segnale di velocità $\omega(t)$ e dal segnale di riferimento ottenuto per mezzo del potenziometro. In questo modo si riducono le fluttuazioni del segnale di errore ω_{err} in ingresso al controllore e di conseguenza la presenza di disturbi ad alta frequenza in uscita, che possono causare vibrazioni indesiderate del dispositivo.

F_1 e F_2 consistono in due filtri passa-basso *Infinite Impulse Response* del primo ordine, caratterizzati da un'equazione alle differenze del tipo:

$$y[n] = ax[n] + (1 - a)y[n - 1] \quad (30)$$

dove $y[n]$ è l'uscita e $x[n]$ è l'ingresso del sistema all'istante n . Affinché sia verificata la BIBO stabilità del sistema è condizione necessaria e sufficiente che il parametro a (anche chiamato *smoothing factor*) sia minore o uguale all'unità.

Ricaviamo la funzione di trasferimento del filtro $F(z)$ partendo dall'equazione (30). Effettuiamo la trasformata Zeta su ambo i membri:

$$y[n] \xrightarrow{z} Y(z) = aX(z) + (1 - a)Y(z)z^{-1} \quad (31)$$

dalla (2) si isola $Y(z)$ e si divide per $X(z)$:

$$\frac{Y(z)}{X(z)} = F(z) = \frac{a}{1 - (1 - a)z^{-1}} \quad (32)$$

Dalla funzione di trasferimento in z ora vogliamo passare alla sua equivalente nel dominio di Laplace, attraverso il processo di discretizzazione (inverso) *Backward Euler*, il quale impone la seguente relazione:

$$z^{-1} = 1 - sT_s \quad (33)$$

Il motivo per quale si effettua questa operazione è trovare il polo p_1 della funzione di trasferimento e con esso la frequenza di taglio f_t del filtro F .

Sostituiamo la (33) nella (32):

$$F(s) = \frac{a}{1 - (1 - a)(1 - sT_s)} = \frac{a}{a + (1 - a)sT_s} = \frac{1}{1 + \frac{(1 - a)T_s}{a} \cdot s} \quad (34)$$

Dalla (34) è facile dedurre che la funzione di trasferimento $F(s)$ ha un polo p_1 reale che vale:

$$p_1 = -\frac{a}{(1 - a)T_s} \quad (35)$$

Il polo p_1 è negativo, quindi stabile, fintantoché $a \leq 1$.

Troviamo la frequenza di taglio del filtro:

$$f_t = \frac{|p_1|}{2\pi} = \frac{a}{(1 - a)2\pi T_s} \quad (36)$$

Dalla (36) si nota che diminuendo a diminuisce anche f_t e con essa la banda del sistema, se la banda diminuisce il tempo di salita della risposta al gradino aumenta. Dunque, lo *smoothing factor* “ a ” è un parametro che regola la velocità di risposta del filtro.

Arrivati a questo punto, dalla (36) otteniamo la formula per calcolare a in funzione della frequenza di taglio desiderata.

$$a = \frac{1}{1 + \frac{1}{2\pi T_s f_t}} \quad (37)$$

Occorre ricordare che per diminuire al minimo la distorsione del segnale d’ingresso, dovuta al processo di discretizzazione del sistema, la frequenza di taglio del filtro deve essere molto minore della frequenza di Nyquist:

$$f_t \ll f_{nyq} = \frac{1}{2T_s} = \frac{1}{2 \cdot 10^{-3}} = 500 \text{ Hz} \quad (38)$$

Si decide di assegnare ad entrambi i filtri F_1 e F_2 una frequenza di taglio nell'intorno di $\frac{f_{nyq}}{20}$ ovvero 25 Hz. A questa, corrisponde un valore del parametro a :

$$a = \frac{1}{1 + \frac{1}{2\pi \cdot 10^{-3} \cdot 25}} = 0.1358 \rightarrow 0.14 \quad (39)$$

Una volta trovato lo *smoothing factor* desiderato, l'equazione alle differenze dei filtri è completamente definita ed implementabile come algoritmo nel microcontrollore.

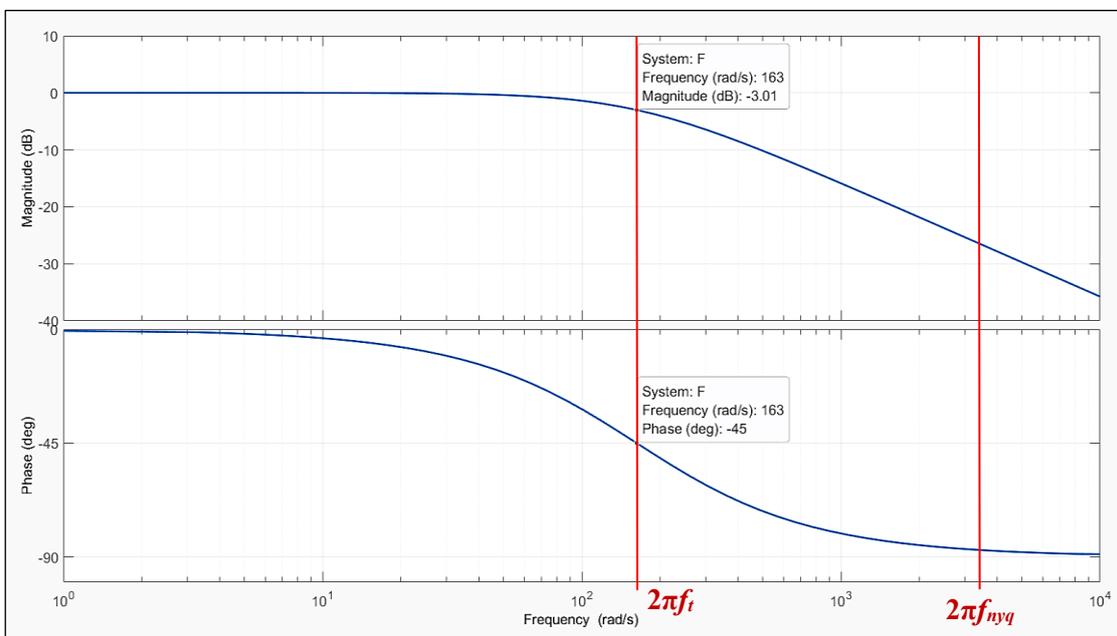


Figura 5.3: diagrammi di Bode della risposta in frequenza dei filtri F_1 e F_2 ; sono state evidenziate la pulsazione di taglio del filtro e la pulsazione di Nyquist.

5.4 Progettazione del controllore

Il blocco di controllo C riceve in ingresso il segnale errore di velocità ω_{err} , che quantifica il divario tra l'effettiva velocità del motore ω_{meas} e il riferimento impostato ω_{ref} , producendo in uscita un segnale di riferimento di corrente elettrica i_{ref} per il blocco di processo P . Esso consiste in un regolatore proporzionale-integrativo (abbreviato PI) e trattandosi di un sistema a tempo discreto anch'esso è implementato come algoritmo nel μC .

Si è deciso di escludere il contributo derivativo dal momento che il processo da controllare $P(s)$ non necessita di un incremento del suo margine di fase, essendo quest'ultimo già prossimo a 90° ; aggiungendo anche la parte derivativa si otterrebbe unicamente un'amplificazione del rumore presente sul segnale d'ingresso.

Per la progettazione del controllore si parte dalla sua funzione di trasferimento nel dominio delle trasformate di Laplace:

$$C(s) = k_p + \frac{k_i}{s} = \frac{k_i}{s} \left(1 + \frac{k_p}{k_i} s \right) \quad (40)$$

dove k_p e k_i sono denominati rispettivamente *guadagno proporzionale* e *guadagno integrale* del PI. La funzione di trasferimento è dunque composta da un guadagno di Bode pari a k_i , un polo nell'origine e uno zero reale stabile (k_p e k_i hanno valori reali positivi).

Rinominiamo:

$$G(s) = C(s)P(s) = \frac{k_i k_t}{B} \cdot \frac{1}{s} \cdot \frac{1 + \frac{k_p}{k_i} s}{1 + \frac{J}{B} s} = \frac{G_0}{s} \cdot G_1(s) \quad (41)$$

la serie dei due sistemi $C(s)$ e $P(s)$, ovvero la funzione di trasferimento del sistema in catena aperta.

Dalla teoria dei sistemi sappiamo che, se il sistema in catena aperta $G(s)$ ha un polo nell'origine con molteplicità uno, allora il corrispondente sistema in catena chiusa

$$W(s) = \frac{G(s)}{1 + G(s)} \quad (42)$$

ottenuto mediante retroazione negativa unitaria, è in grado di inseguire un segnale di riferimento a rampa lineare con errore costante e non nullo. Tale errore si può calcolare nel seguente modo:

$$e_{rp} = \frac{1}{G_0} = \frac{B}{k_t k_i} \quad (43)$$

dove G_0 è il guadagno in banda passante del sistema in catena aperta $G(s)$. Fissiamo un valore di e_{rp} molto piccolo, in modo da ridurre il più possibile l'errore; per esempio: $e_{rp} = 10^{-3}$. Con l'equazione (43) ci si può ricavare il valore di k_i :

$$k_i = \frac{B}{k_t e_{rp}} = \frac{2 \cdot 10^{-4}}{0.035 \cdot 10^{-3}} \cong 5.7 \quad (44)$$

Ora occorre fissare la posizione dello zero reale di $C(s)$. Il criterio con il quale si effettua questa scelta riguarda la pulsazione di attraversamento del sistema in catena aperta $G(s)$. Come specifica di progetto si decide di fissare tale pulsazione ω_{cr} pari a un ventesimo della pulsazione di Nyquist, ricordando che:

- frequenza di campionamento: $f_s = 1000$ Hz
- frequenza di Nyquist: $f_{nyq} = \frac{f_s}{2} = 500$ Hz

quindi la pulsazione di attraversamento, analogamente a quanto visto per i filtri digitali nella sezione 5.3, vale:

$$\omega_{cr} = \frac{2\pi f_s}{20} \cong 160 \text{ rad/s} \quad (45)$$

Consideriamo il sottosistema

$$G'(s) = \frac{k_i}{s} \cdot P(s) \quad (46)$$

equivalente al sistema in catena aperta $G(s)$ ma senza il contributo dello zero dato dal controllore.

In corrispondenza della pulsazione di attraversamento, il modulo in dB di $G'(j\omega)$ vale

$$|G'(j\omega_{cr})|_{dB} = -15 \text{ dB} \quad (47)$$

dunque, si deve sfruttare lo zero per far sì che il modulo di $G(s)$ valga circa zero dB in corrispondenza di ω_{Cr} . Dal momento che uno zero stabile contribuisce sul modulo della funzione di trasferimento con una pendenza di +20 dB/decade per ogni pulsazione superiore a quella in cui esso è posizionato, ne consegue che per ottenere un incremento di +15 dB in ω_{Cr} occorre posizionare lo zero a tre quarti di decade prima, ovvero alla pulsazione

$$\omega_z = 10^{-\frac{3}{4}} \cdot \omega_{Cr} = 28 \text{ rad/s} \quad (48)$$

Ora che conosciamo la posizione dello zero possiamo ricavare il guadagno proporzionale:

$$\omega_z = \frac{k_i}{k_p} \implies k_p = \frac{k_i}{\omega_z} = 0.2 \quad (49)$$

A questo punto la funzione di trasferimento del controllo $C(s)$ è completamente definita.

In figura 5.4 si possono osservare le risposte in frequenza del processo senza controllo $P(s)$, del sistema in catena aperta $G(s)$ e del sistema in catena chiusa $W(s)$; rispettivamente in blu, rosso e giallo. È stata evidenziata la pulsazione di attraversamento ω_{Cr} dove il modulo di $G(s)$ ha valore unitario.

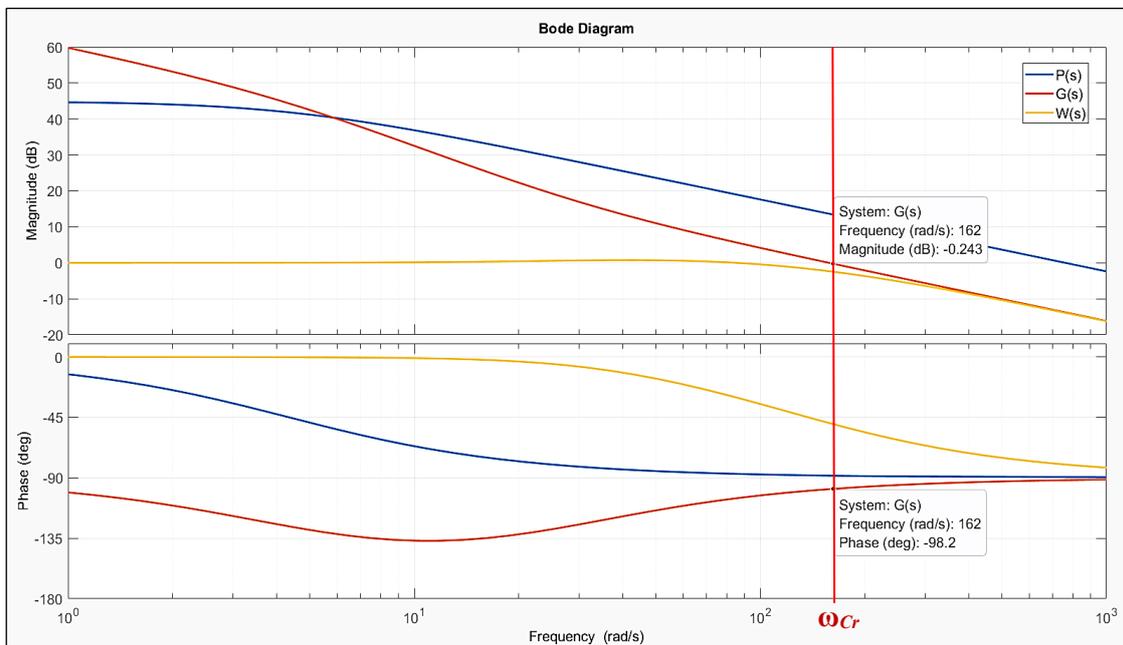


Figura 5.4: risposta in frequenza di $P(s)$, $G(s)$, $W(s)$. Vedi legenda in alto a destra

Il margine di fase del sistema in catena aperta, in corrispondenza di ω_{cr} , dopo l'aggiunta del regolatore PI, è diventato:

$$\varphi_G = \angle[G(s)] + 180^\circ = -98.5^\circ + 180^\circ = 81.5^\circ \quad (50)$$

Procediamo con la discretizzazione del sistema, alla quale ne consegue una ridefinizione dei parametri k_p e k_i . Utilizziamo il metodo *Backward Euler* per ottenere la funzione di trasferimento nel dominio delle *Zeta*, partendo da $C(s)$.

$$s = \frac{1 - z^{-1}}{T_s} \quad (51)$$

$$C(s) = k_p + \frac{k_i}{s} \quad (52)$$

Sostituendo la (8) nella (9) si ottiene

$$C(z) = k_p + \frac{k_i \cdot T_s}{1 - z^{-1}} = k'_p + \frac{k'_i}{1 - z^{-1}} \quad (53)$$

I nuovi guadagni del controllore a tempo discreto $C(z)$ sono:

$$\begin{cases} k'_p = k_p = 0.2 \\ k'_i = k_i \cdot T_s = 0.0057 \cong 0.006 \end{cases} \quad (54)$$

Ora, dalla (8) cerchiamo di ricavare l'equazione alle differenze collegata:

$$C(z) = \frac{U(z)}{E(z)} = \frac{k'_p(1 - z^{-1}) + k'_i}{1 - z^{-1}} \quad (55)$$

dove $E(z)$ e $U(z)$ sono le trasformate Zeta rispettivamente dei segnali d'ingresso e d'uscita del sistema $C(z)$. Proseguendo con i calcoli si arriva a:

$$U(z) = U(z)z^{-1} + k'_p(1 - z^{-1})X(z) + k'_iX(z) \quad (56)$$

e anti-trasformando ambo i membri

$$U(z) \xrightarrow{z^{-1}} u[k] = u[k - 1] + k'_p\{e[k] - e[k - 1]\} + k'_ie[k] \quad (57)$$

si ottiene l'equazione alle differenze che permette di calcolare l'uscita del regolatore all'istante k . Si noti che nell'equazione compare l'uscita nell'istante passato $k-1$, il che indica la presenza di una retroazione interna nel regolatore.

Con la (57) il regolatore PI è completamente definito e implementabile come algoritmo nel μC .

Capitolo 6

Implementazione del codice

Nel seguente capitolo si vuole mostrare come è stato organizzato il codice nell'IDE e quali sono le funzioni principali che sono utilizzate per controllare, mediante il microcontrollore, le varie parti del sistema. Verranno successivamente illustrati due algoritmi fondamentali per il funzionamento del progetto.

6.1 Struttura e funzioni

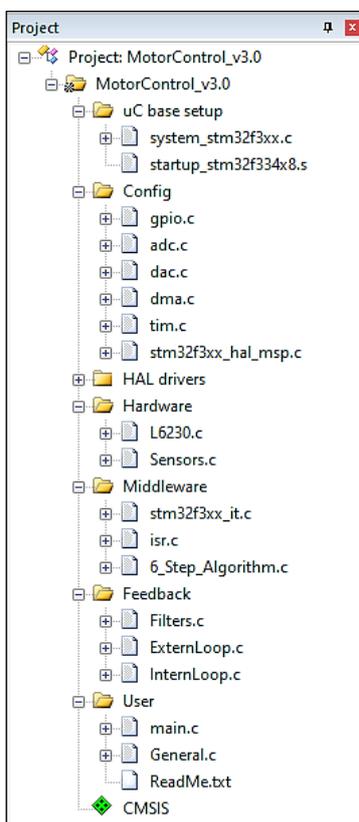


Figura 6.1: finestra project di µVision

All'interno dell'IDE è possibile organizzare i file ".c" e i file header ".h" in diverse cartelle; la struttura ad albero che ne risulta è visualizzata nella finestra "project" di µVision (figura 6.1).

Il criterio di suddivisione dei file è basato sulla finalità che hanno le funzioni scritte al loro interno. Le categorie sono:

- *µC base setup*: file per l'inizializzazione del microcontrollore;
- *Config*: file di configurazione delle periferiche;
- *HAL drivers*: file di driver delle periferiche mediante Hardware Abstraction Layer;
- *Hardware*: funzioni per il controllo dei sensori e degli inverter nel chip LM6230;
- *Middleware*: gestione delle interruzioni e implementazione dell'algoritmo di rotazione;
- *Feedback*: codici per l'implementazione dei filtri digitali, del regolatore PI e dei segnali di riferimento;
- *User*: funzioni di setup, avvio e arresto del sistema.

6.1.1 Categoria Hardware

Le funzioni nel file *L6230.c* gestiscono gli inverter che forniscono energia elettrica alle fasi del motore.

```
void EnableInput_CH1_E_CH2_E_CH3_D(void);  
void EnableInput_CH1_E_CH2_D_CH3_E(void);  
void EnableInput_CH1_D_CH2_E_CH3_E(void);  
void DisableInput_CH1_D_CH2_D_CH3_D(void);
```

Permettono di accendere una coppia di fasi e tenere la terza disattivata o se necessario spegnerle tutte e tre;

```
void Start_PWM_Generation(void);  
void Stop_PWM_Generation(void);
```

avviano e arrestano la generazione dei segnali PWM;

```
void Set_DC_PWM_Ch1(uint16_t pulse_value);  
void Set_DC_PWM_Ch2(uint16_t pulse_value);  
void Set_DC_PWM_Ch3(uint16_t pulse_value);
```

impostano un nuovo duty cycle (proporzionale a *pulse_value*) ai segnali PWM per ciascuna fase.

Le funzioni nel file *Sensors.c*, invece gestiscono i convertitori A/D, i segnali provenienti dai sensori ad effetto Hall e dal potenziometro.

```
void Start_ADC_Reading(void);  
void Stop_ADC_Reading(void);
```

Avviano e arrestano il campionamento dei segnali da parte dei convertitori A/D;

```
void Get_Hall_States(void);
```

aggiorna il buffer contenente gli stati dei sensori ad effetto hall;

```
uint16_t Get_Trimmer_Value(void);
```

restituisce il riferimento di velocità (non ancora filtrato).

6.1.2 Categoria Middleware

Il file *isr.c* contiene le routine di interrupt:

```
void TIM6_DAC1_IRQHandler(void);  
void TIM2_IRQHandler(void);  
void EXTI15_10_IRQHandler(void);
```

le prime due relative rispettivamente all'interrupt generato dal timer 6 e dal timer 2, mentre la terza viene eseguita quando viene premuto il pulsante *User*.

Nel file *6_Step_Algorithm.c* si trovano le funzioni necessarie per implementare l'algoritmo di rotazione:

```
void Setup_6Step_Alg(void);
```

inizializza le variabili necessarie,

```
void Get_Curr_Step(void);  
void Get_Next_Step(void);  
void Check_Step_Change(void);
```

rispettivamente calcolano: lo *step* (ovvero la posizione) attuale del rotore, calcola la posizione successiva da raggiungere, controlla se il rotore l'ha raggiunta;

```
void Move_To_Next_Step(uint16_t pulse_value);  
void Move_To_Step_1(uint16_t pulse_value);  
void Move_To_Step_2(uint16_t pulse_value);  
void Move_To_Step_3(uint16_t pulse_value);  
void Move_To_Step_4(uint16_t pulse_value);  
void Move_To_Step_5(uint16_t pulse_value);  
void Move_To_Step_6(uint16_t pulse_value);
```

impostano opportunamente le uscite per permettere al rotore di spostarsi allo step successivo in base alla posizione attuale e modifica il duty cycle della pwm con un nuovo valore proporzionale a *pulse_value*.

6.1.3 Categoria Feedback

Nel file *Filters.c* viene implementato il filtro digitale IIR passa-basso, mediante la funzione:

```
void IIR_LowPass_Filter(uint16_t *addr, float fact);
```

occorre fornire l'indirizzo del buffer contenente i campioni del segnale da filtrare e lo smoothing factor, l'output viene salvato automaticamente nel vettore indicato.

Nel file *InternLoop.c* si trova:

```
void Start_Current_Ref(void);  
void Stop_Current_Ref(void);
```

avviano e arrestano il convertitore D/A per la generazione del riferimento di corrente;

```
void Set_Current_Ref(void);
```

imposta un nuovo valore per tale riferimento controllando che non venga superata la massima corrente consentita.

Infine, nel file *ExternLoop.c* abbiamo

```
void Get_Rotor_Speed(void);  
void Get_Filtered_Speed(void);
```

la prima funzione calcola la velocità di rotazione del rotore, la seconda funzione prende i dati prodotti dalla prima e li fa passare tramite il filtro digitale;

```
void Get_Speed_Reference(void);
```

questa funzione prende il segnale di riferimento "grezzo" prodotto dalla funzione *Get_Trimmer_Value* e lo fa passare attraverso il filtro digitale. Viene inoltre controllato che il valore sia dentro un determinato range di valori.

```
void PI_Regulator(void);
```

questa funzione computa l'uscita del controllo digitale PI applicando l'algoritmo descritto nella sezione 5.4.

6.1.4 Categoria User

Il file *main.c* è quasi del tutto vuoto, prima di cadere nel `while(1)`, il microcontrollore esegue alcune funzioni di inizializzazione.

Nel file *General.c* sono contenute le seguenti funzioni:

```
void Setup(void);
```

effettua l'inizializzazione di alcune variabili all'accensione del microcontrollore;

```
void Start_System(void);  
void Stop_System(void);
```

la prima funzione porta il sistema dallo stato “fermo” allo stato “operativo” quando viene premuto il pulsante *User*, la terza effettua il procedimento inverso, ovvero ferma il sistema, se il medesimo pulsante viene premuto una seconda volta.

6.2 Algoritmi

Prima della scrittura del codice per il microcontrollore è stato necessario sviluppare due algoritmi indispensabili per ottenere il funzionamento del sistema.

Il primo algoritmo ha come scopo il calcolo della velocità del motore sfruttando le informazioni dei sensori ad effetto Hall; il secondo algoritmo invece, si prefigge l’obiettivo di avviare e mantenere il motore in funzione polarizzando opportunamente le fasi dello statore a seconda della posizione del rotore.

6.2.1 Calcolo della velocità di rotazione

Per calcolare la velocità del motore si utilizzano due variabili `speed_cont` e `step_cont` come dei contatori e i sensori ad effetto Hall per sapere quando il rotore effettua un cambio di passo. Il contatore `speed_cont` viene incrementato di un’unità ogni volta che viene eseguita la ISR del timer 2, ovvero ogni 50 microsecondi; il contatore `step_count`, invece, si incrementa ogni volta che viene rilevato un cambio di posizione del rotore. Quando il conteggio di `step_cont` raggiunge il numero “24” significa che il rotore ha effettuato una rivoluzione completa; in questa situazione il valore `speed_cont` indica quanti intervalli da 50 μ s sono necessari per effettuare una rivoluzione a quella determinata velocità.

Dunque, si può calcolare la velocità angolare utilizzando la formula (58), sintetizzata nel flowchart dalla funzione $f(\dots)$.

$$\text{speed} = \frac{60 \cdot 0.105}{\text{speed_cont} \cdot 50 \cdot 10^{-6}} \quad (58)$$

La costante “0.105” converte il valore di velocità dall’unità di misura [rpm] (rivoluzioni per minuto) a [rad/s].

La figura 6.1 rappresenta graficamente il diagramma di flusso dell'algorithm appena descritto.

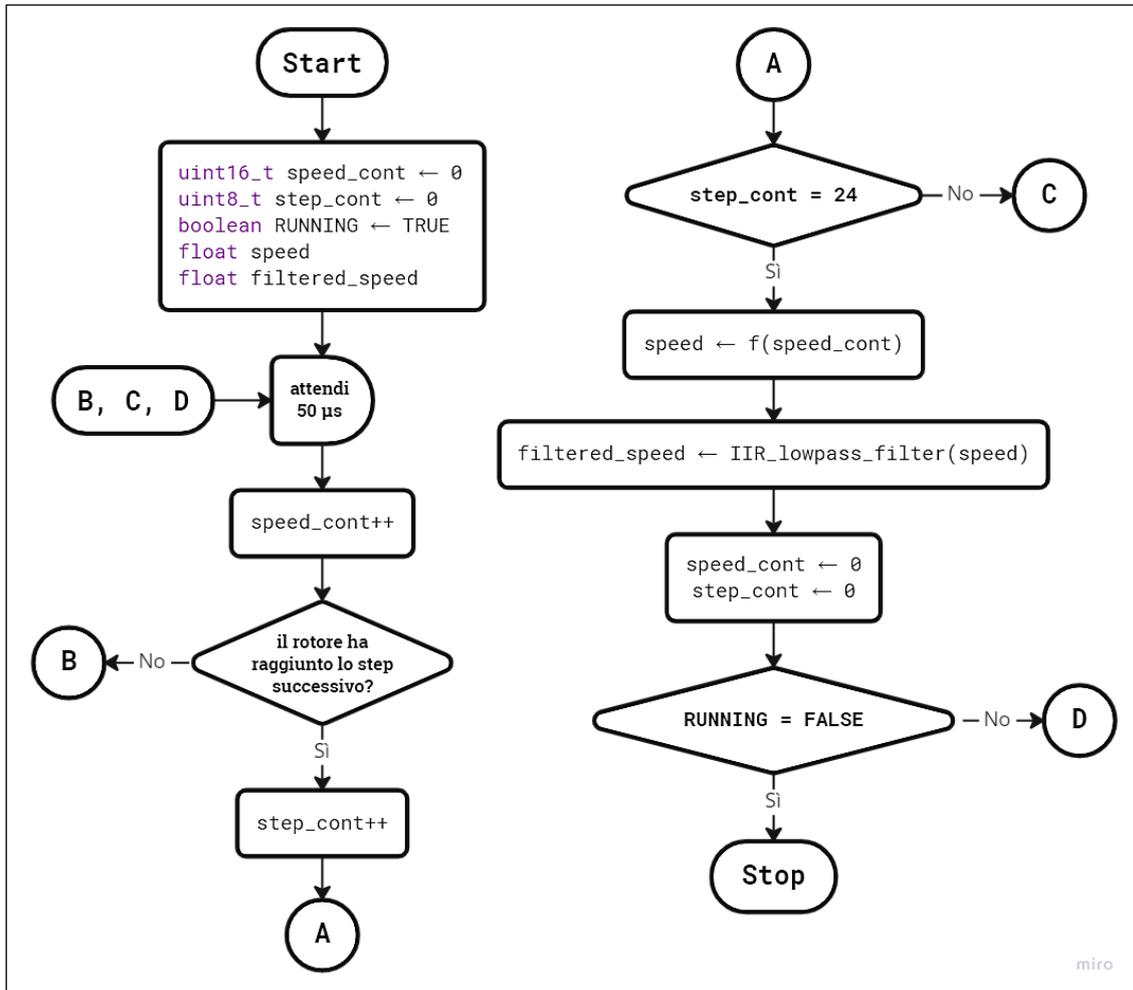


Figura 6.1: flow-chart dell'algorithm per il calcolo della velocità angolare

6.2.2 Algoritmo di rotazione 6-steps

Come già spiegato nei capitoli precedenti, affinché il motore giri e continui a girare occorre polarizzare opportunamente le tre fasi dello statore a seconda della posizione del rotore. Per mezzo dei sensori ad effetto Hall si possono distinguere sei differenti posizioni (anche detti *step*) del rotore, da qui il nome *6-steps* che indica questa metodologia di pilotaggio del dispositivo. Occorre dunque un algoritmo a “basso livello” (con un maggiore grado di vicinanza all’hardware del sistema) che si occupi di leggere la posizione del rotore e generi l’output corretto, risultando del tutto trasparente ai segmenti di codice ad “alto livello” ovvero che implementano il controllo esterno di velocità.

In figura 6.2 si riporta il flow-chart che descrive il principio di funzionamento dell'algoritmo.

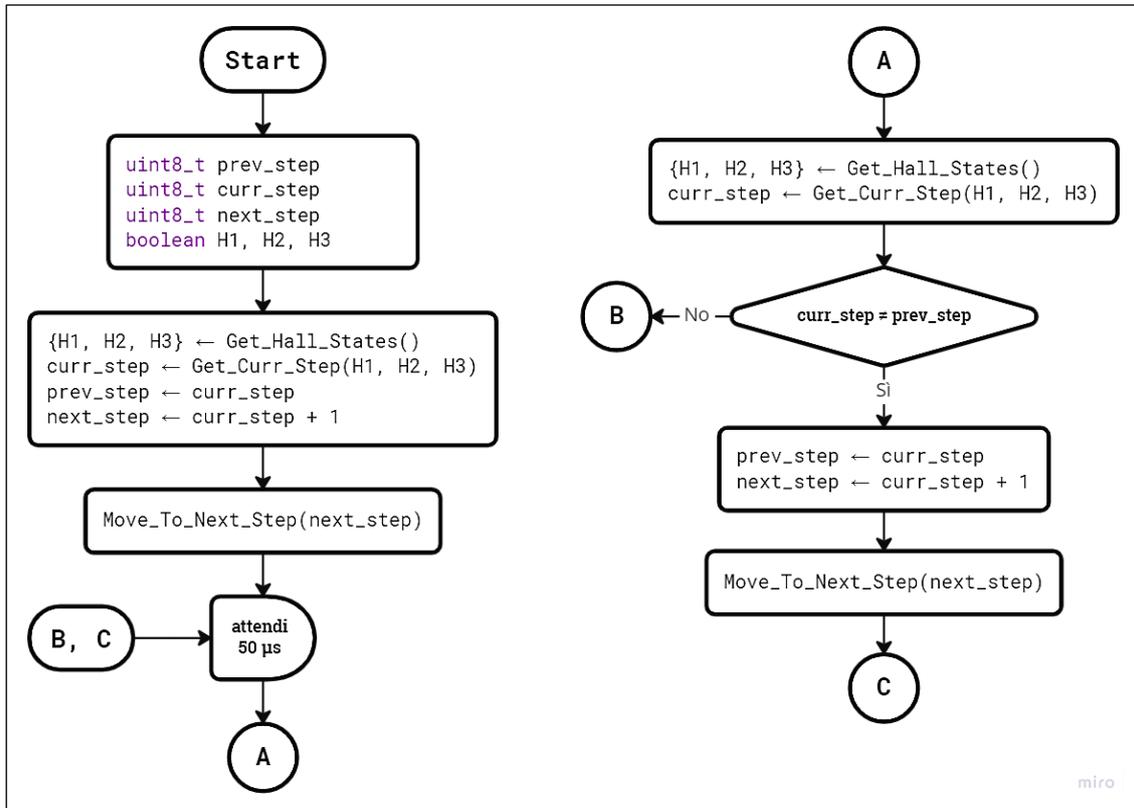


Figura 6.2: flow-chart dell'algoritmo di rotazione 6-steps

Capitolo 7

Risposta al gradino

In seguito a una miglior taratura dei parametri del controllore dettata dai test pratici effettuati in laboratorio, si vuole mostrare la risposta del sistema a un gradino del segnale di riferimento e le differenze riportate in tale risposta utilizzando il sistema prima in catena aperta (senza controllo della velocità), e successivamente in catena chiusa.

Il segnale della velocità misurata viene convertito in un segnale di tensione per mezzo di un convertitore digitale-analogico presente nel microcontrollore e visualizzato tramite l'oscilloscopio. A causa di questa conversione il vero segnale viene sommato con dei disturbi ad alta frequenza (nell'ordine dei megaHertz) che comportano la visione all'oscilloscopio di una traccia leggermente "sporca".

Il segnale di riferimento viene modificato manualmente nell'interfaccia del debugger di μ Vision, andando a modificare repentinamente il valore della variabile che lo rappresenta. Per effettuare i test si è scelto di utilizzare i seguenti valori di riferimento:

- 400 RPM come livello basso di velocità;
- 2400 RPM come livello alto.

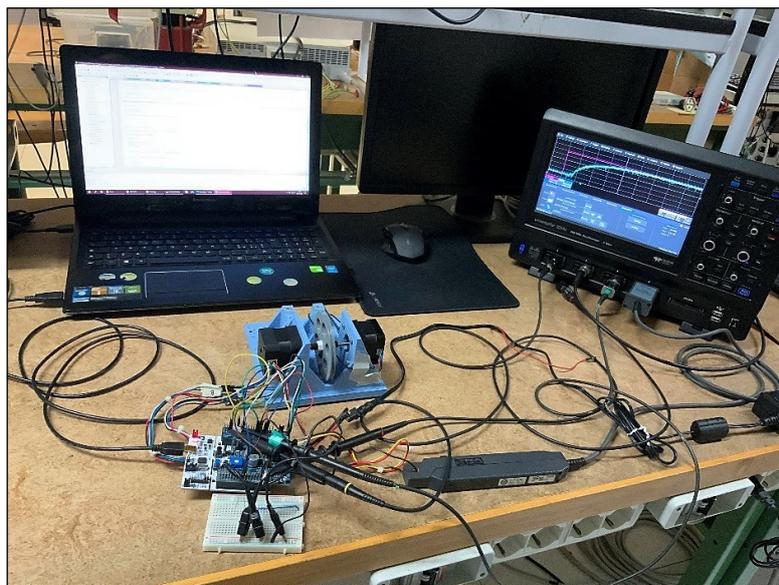


Figura 7.1: foto della postazione di collaudo in laboratorio

Per motivi funzionali non è possibile partire con il rotore completamente fermo (zero RMP).

7.1 Risposta in catena aperta

In figura 7.2 si osserva l'evoluzione dell'uscita del sistema (in blu) in funzione del segnale di riferimento (in violetto). Essendo in catena aperta, il sistema non possiede il polo nell'origine introdotto dal regolatore PI e di conseguenza si tratta di un sistema di tipo zero. Alla luce di ciò l'uscita insegue i riferimenti costanti con un errore costante e non nullo, come si può notare dai grafici.

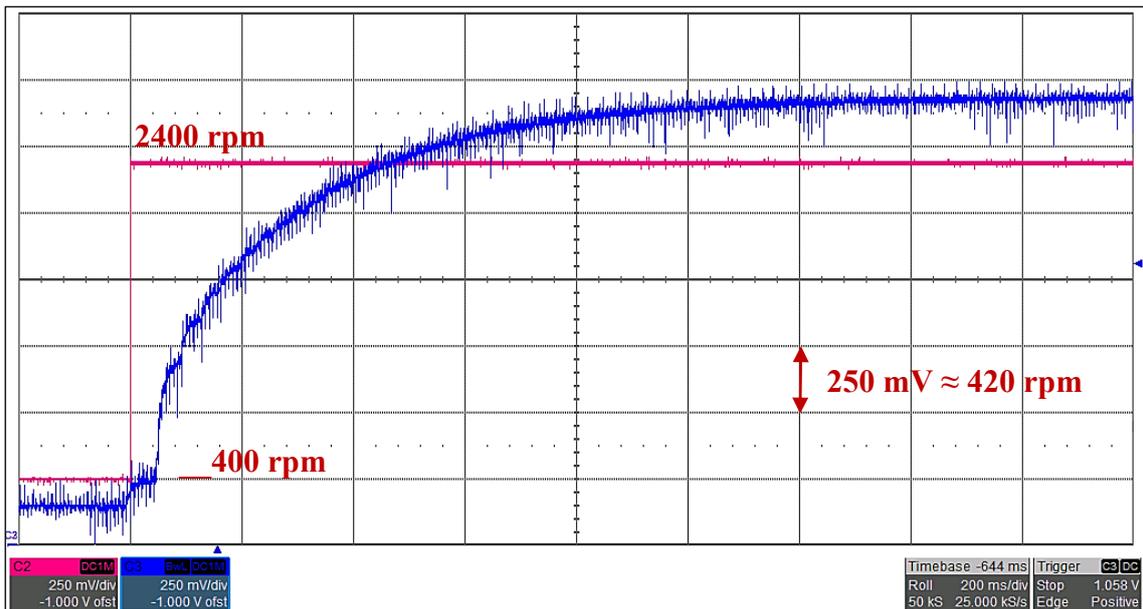


Figura 7.2: risposta del sistema in catena aperta al gradino positivo

Si stima un tempo di salita, nel caso di una variazione positiva del riferimento, in catena aperta, pari a circa un secondo.

$$t_{LH,open} \cong 1 \text{ s}$$

Nel caso invece di una variazione negativa, osservabile in figura 7.3, il tempo di discesa cala leggermente rispetto al precedente e ammonta a poco più di mezzo secondo.

$$t_{HL,open} \cong 0.7 \text{ s}$$

Possiamo notare che in fase di decelerazione del motore la forma d'onda dell'uscita non mantiene il caratteristico andamento esponenziale visto durante la fase di accelerazione in figura 7.2; ciò significa che in questa situazione entrano in gioco dei fenomeni dei quali non si è tenuto conto durante la modellizzazione del motore.

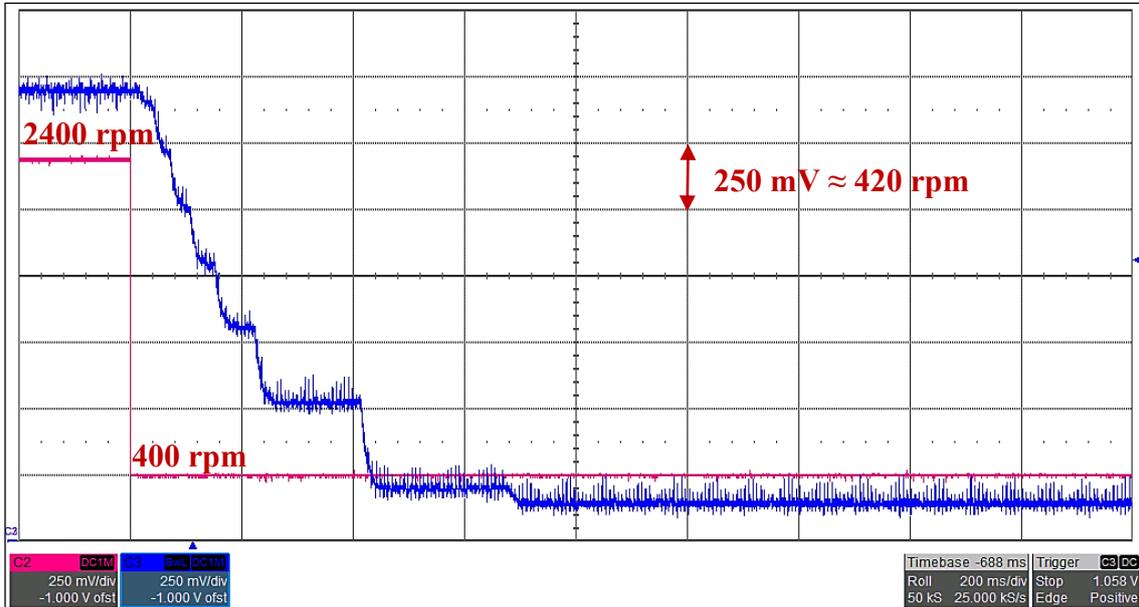


Figura 7.3: risposta del sistema in catena aperta al gradino negativo

7.2 Risposta in catena chiusa

Consideriamo ora il sistema retroazionato con il regolatore PI e osserviamo come la sua uscita reagisce a fronte di una variazione a gradino del segnale di riferimento e con quali differenze rispetto al suo corrispettivo in catena aperta.

La prima cosa che salta all'occhio osservando la figura 7.4 è che ora l'uscita del sistema è in grado di inseguire un valore costante del riferimento con errore costante e nullo. Il tempo di salita è leggermente aumentato rispetto al caso in catena aperta, questo perché la banda passante del sistema retroazionato è stata ridotta con l'introduzione del regolatore; tuttavia si tratta di incremento molto ridotto.

$$t_{LH,closed} \cong 1.2 \text{ s}$$

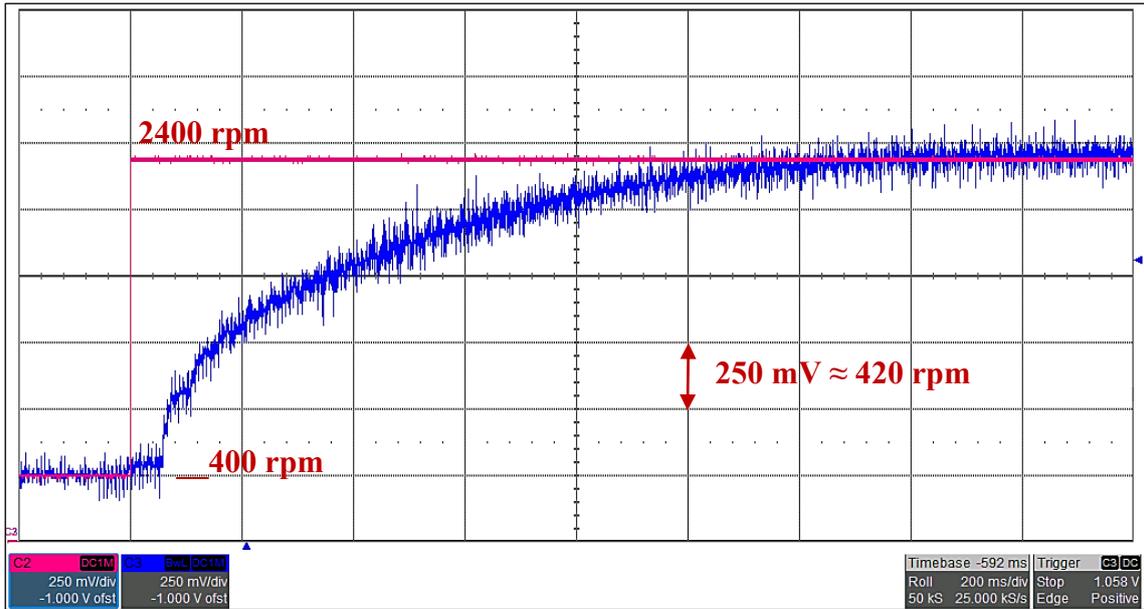


Figura 7.4: risposta del sistema in catena chiusa al gradino positivo

Con l'introduzione del regolatore si è potuto correggere l'andamento della velocità in fase di decelerazione (vedi figura 7.5) riportandola al tipico comportamento di esponenziale decrescente; a scapito, tuttavia, del tempo di discesa che si vede più che raddoppiato.

$$t_{HL,closed} \cong 1.6 s$$

Dunque il motore decelera in maniera più "morbida" ma in un tempo maggiore.

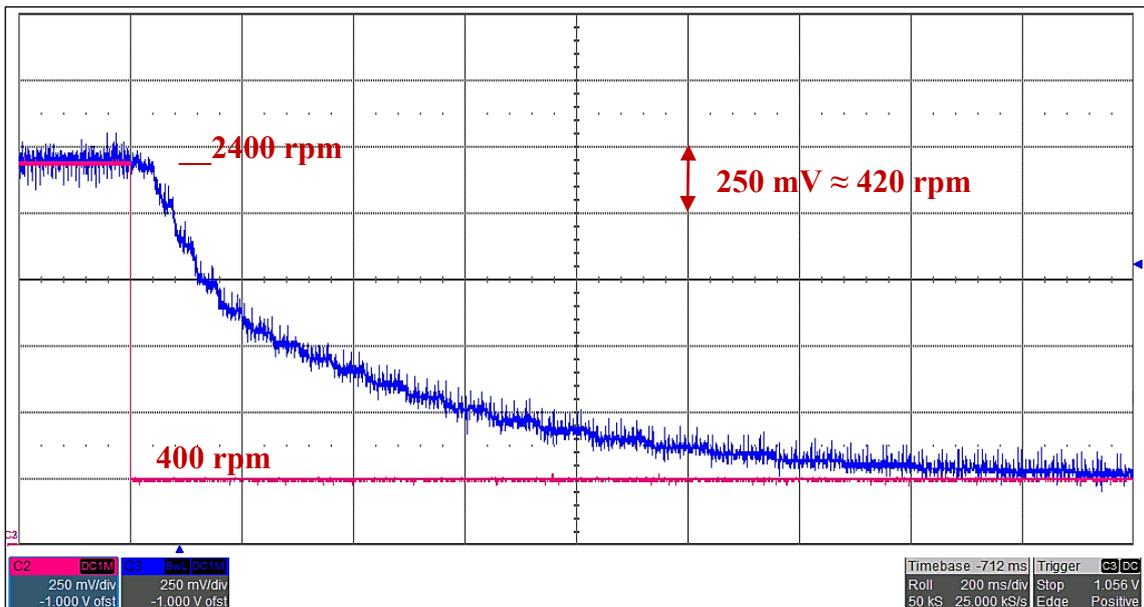


Figura 7.5: risposta del sistema in catena chiusa al gradino negativo

Conclusioni

I test effettuati dimostrano che il nuovo sistema progettato per controllare la velocità del motore funziona e soddisfa le specifiche esposte all'inizio di questo documento; senza dubbio è possibile migliorare ulteriormente il progetto, per esempio sviluppando un controllo in grado di far fronte anche ad eventuali carichi meccanici variabili in uscita.

Lo svolgimento di questo progetto di tesi ha indubbiamente permesso allo studente laureando di acquisire nuove conoscenze circa il funzionamento e il pilotaggio dei motori elettrici BLDC e di consolidare le proprie competenze riguardo i microcontrollori di ST Microelectronics e la loro programmazione.

L'esperienza fatta in laboratorio ha inoltre messo alla prova le capacità dello studente nel far fronte a un problema di carattere ingegneristico, senza l'ausilio di importanti aiuti da terze persone, ma facendo affidamento unicamente sulle proprie abilità di risoluzione dei problemi e sulle conoscenze acquisite nei corsi di laurea frequentati e da fonti esterne quali libri e siti internet.