POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Elettronica

Politecnico
di Torino

Tesi di Laurea Magistrale in collaborazione con *START POWER SRL*

Controllo di un azionamento in corrente continua con Arduino per applicazioni industriali

Relatore accademico:
Prof. Gianmario Pellegrino

Candidato:
Marco Simonetti

Tutor aziendale:
Natalino D'Ambrosio
Doriano Perin

Anno Accademico 2021-2022

Alla mia famiglia

Sommario

Nell'ambito automotive, aerospaziale e industriale, l'utilizzo di motori elettrici è inevitabile. Esistono due tipi di motori, quelli in AC (corrente alternata) e quelli in CC (corrente continua). La prima tipologia sta prendendo sempre più piede in tutti gli ambiti, specialmente in quelli in cui si utilizzano motori da pochi kW poichè quando si ha a che fare con potenze che vanno oltre i 100kW l'uso dei motori in AC risulta molto svantaggioso in termini di costo-beneficio. Per questo, in sistemi in cui si ha bisogno di motori elettrici di maggiore potenza, è più conveniente utilizzare i motori in corrente continua. Inoltre il controllo dei motori CC risulta più semplice in quanto la loro velocità può essere controllata variando la tensione di alimentazione. Questa tipologia di motori è comunque tutt'oggi molto utilizzata anche in applicazioni low cost e in ambiti dove è necessario avere una coppia di spunto elevata. Ne sono un esempio i laminatoi e le trafilerie in cui si ha bisogno di molta potenza per sollevare carichi di diverse tonnellate. Per quanto riguarda l'azionamento dei motori CC, per elevate potenze risulta essere conveniente l'uso di SCR (Silicon Controlled Rectifier) che risulta essere molto robusto permettendo di avere un costo di manutenzione molto ridotto rispetto agli inverter utilizzati per i motori AC in cui i condensatori elettrolitici hanno bisogno di essere sostituiti periodicamente.

Oggigiorno tutti i sistemi automatici hanno bisogno di un dispositivo di controllo. Questo è un particolare sistema per l'elaborazione dell'informazione, destinato al controllo dei processi fisici, il quale deve potersi interfacciare con l'ambiente esterno. Le sue funzionalità generalmente devono essere quelle di controllo a ciclo chiuso classico del sistema (regolazione o asservimento), calcolo dei valori di riferimento (set-point), gestione di eventuali allarmi ed anomalie e infine realizzazione dell'interfaccia di comunicazione con operatore o altri dispositivi. Al giorno d'oggi possiamo trovare un gran numero di tipologie di controllori, ma la più utilizzata è quella basata su controllori PID (Proporzionale-Integrativo-Derivativo). Il successo di questo tipo di controllori è dovuto principalmente alla sua facilità di taratura e al largo impiego che si fa in ambito industriale.

In questo progetto di tesi verrà dunque utilizzato un ponte raddrizzatore trifase SCR, un motore CC e verrà programmato un Arduino per permettere la rotazione unidirezionale del motore. In particolare verrà evidenziato l'azionamento di questo mediante il giusto innesco degli SCR e come è stato progettato il controllore PID. In più verrà spiegata tutta la parte hardware di contorno che è stata progettata per far comunicare questi tre elementi tra di loro.

Ringraziamenti

Prima di tutto vorrei ringraziare la Start Power SRL ed in particolare i miei due tutor, Doriano Perin e Natalino D'Ambrosio che mi hanno seguito e consigliato durante la realizzazione del progetto permettendomi di svolgere questo lavoro di tesi in un ambiente stimolante e dinamico. Grazie anche a tutti i componenti di questa azienda con cui si è venuto a creare un rapporto di amicizia e rispetto reciproco.

Infine un ringraziamento anche al Prof. Gianmario Pellegrino che mi ha seguito durante la stesura di questa tesi dimostrandosi molto disponibile.

Indice

Introduzione al progetto di tesi

Nell'ambito industriale dove bisogna raggiungere potenze elevate utilizzare un azionamento in corrente continua risulta essere un'ottima scelta in termini di efficienza/costo. Inoltre al giorno d'oggi non esiste sistema industriale che non sia automatizzato, soprattutto quando è necessario svolgere un compito in maniera ripetuta o laddove bisogna sollevare carichi di diversi quintali in cui è fondamentale tenere sotto controllo la posizione e la velocità con cui questo carico viene spostato.

Questo progetto di tesi è stato svolto in collaborazione con START POWER SRL e l'obiettivo finale è quello di ottenere un sistema di controllo per azionare un motore in corrente continua regolando e stabilizzando la velocità attraverso un controllore PID digitale implementato su una scheda di valutazione Arduino.

Nel capitolo 1 viene spiegato come funziona un motore CC e vengono analizzati i metodi per regolarne la velocità facendo riferimento al circuito elettrico equivalente.

Nel capitolo 2 vengono date le nozioni base sugli SCR e sul funzionamento di un sistema trifase studiando anche il comportamento di un convertitore AC/DC attraverso le equazioni che lo caratterizzano.

Nel capitolo 3 vengono riportate le caratteristiche della scheda di valutazione utilizzata in sede di progetto, ponendo particolare attenzione ai suoi timer interni.

Il capitolo 4 tratta la teoria del controllore utilizzato in cui viene analizzato come progettarlo, il suo funzionamento e come questo può essere implementato a livello digitale.

Infine, nel capitolo 5 vengono riportati tutti i passaggi effettuati per raggiungere lo scopo finale, spiegando nel dettaglio tutta la parte software e hardware.

Capitolo 1

La macchina a corrente continua

1.1 Introduzione

Le macchine a corrente continua sono state i primi esempi di macchine elettriche rotanti ad entrare in circolazione. Attualmente questi dispositivi sono utilizzati nei settori di trasporti elettrici e industriali.
Lo scopo delle macchine è quello di convertire una certa forma di energia meccanica in energia elettrica. Per questo motivo le macchine sono dette anche convertitori elettromeccanici di energia; nello specifico generatori elettromeccanici se avviene la produzioni di energia elettrica mediante energia meccanica prodotta da turbine o motori a combustione, mentre in caso contrario abbiamo i motori elettrici. La trasformazione di queste due energie avviene mediante l'unica cosa che lega queste due, ovvero il campo magnetico. Dal punto di vista costruttivo le macchine rotanti sono costituite dallo statore , ossia una parte fissa e una mobile detta rotore e tra queste due esiste un piccolo strato di aria chiamato traferro. Sia statore che rotore sono costituiti da materiale ferro magnetico dove, nella parte interna, sono ricavati dei canali detti cave, dove si andrà ad inserire i conduttori per gli avvolgimenti. Nelle macchine a corrente continua la parte che produce flusso normalmente è lo statore, che è denominato induttore o eccitante; il rotore, denominato indotto o armatura, richiama l'induzione delle forze elettromotrici (f_{EM}).

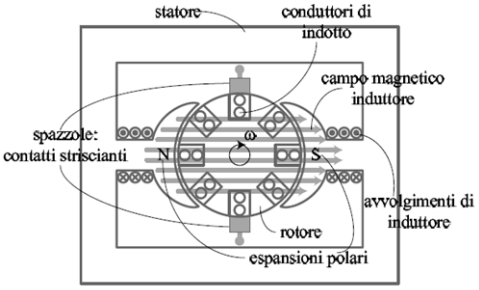


Figura 1.1. Rappresentazione di una macchina a corrente continua

1.2 Motore e dinamo

Una dinamo è una macchina elettrica che trasforma un lavoro meccanico, ricevuto da una fonte di energia meccanica, in energia elettrica sotto forma di corrente continua.
Nella sua forma più semplice, consiste di una spira conduttrice (rotore) immersa in un campo magnetico e messa in rotazione da un albero. Per la legge di Faraday per l'induzione, un conduttore che si muove in un campo magnetico non parallelamente ad esso vede nascere una forza elettromotrice indotta (f_{EM}); chiudendo quindi la spira su un carico elettrico, si ha una corrente scorrere nella spira stessa e nel carico.
Se ci fermassimo qui, la dinamo non darebbe corrente continua. Infatti, la differenza di potenziale nella singola spira varia con legge sinusoidale con l'angolo di rotazione e quindi cambia segno ogni mezzo giro (producendo corrente alternata). È perciò necessario connettere i capi della spira ad un oggetto chiamato "collettore" o "commutatore", calettato sul rotore e solidale ad esso, che, attraverso un contatto strisciante con spazzole (dette carboncini), scambia i capi della spira ogni mezzo giro mantenendo la tensione in uscita dello stesso segno.

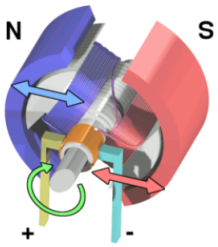


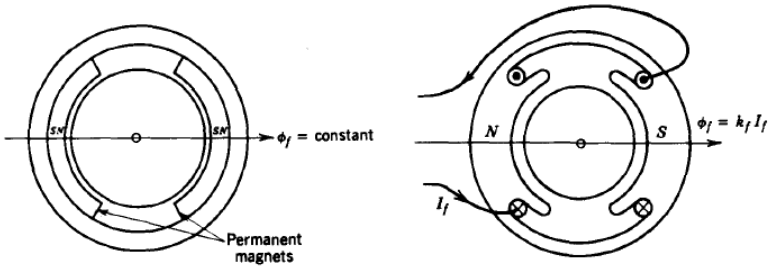
Figura 1.2. Rappresentazione di una dinamo

Un motore in corrente continua (CC) è un tipo di macchina elettrica che converte l'energia elettrica in energia meccanica funzionando in modo opposto al comportamento

della dinamo. Gli avvolgimenti, una volta alimentati da corrente elettrica, creano campi magnetici che mettono in moto i magneti collegati al rotore. Il rotore, infine, converte e trasmette il movimento dei magneti all'albero motore.

1.3 Circuito di armatura del motore a CC

In un motore CC, il campo di flusso ϕ è stabilito dallo statore sia per mezzo di magneti permanenti come mostrato in figura 1.3(a), dove il flusso sarà costante oppure attraverso avvolgimenti di campo come mostrato in figura 1.3(b) in cui il flusso sarà controllato dalla corrente di campo I_f



(a) (b)

Figura 1.3. Motore CC con magnete permanente(a) e con avvolgimento (b)

Possiamo quindi definire

$$\phi_f = k_f I_f \tag{1.1}$$

dove k_f è una costante di campo. In un motore CC la coppia elettromagnetica è prodotta dall'interazione tra ϕ_f e la corrente di armatura i_a:

$$T_{em} = k_t \phi_f i_a \tag{1.2}$$

dove k_t è la costante di coppia del motore. Un altro parametro molto importante che si crea dalla rotazione dell'armatura ad una certa velocità ω_m in presenza di un flusso di campo ϕ_f è la "back-emf", definita come:

$$e_a = k_e \phi_f \omega_m \tag{1.3}$$

dove k_e è la costante di tensione del motore.

Riguardo le costanti k_t e k_e, queste sono numericamente uguali e ciò è possibile dimostrarlo considerando il motore in regime stazionario considerando la potenza elettrica e la potenza meccanica.

Utilizzando la 1.3 la potenza elettrica vale:

$$P_e = e_a i_a = k_e \phi_f \omega_m i_a \tag{1.4}$$

Mentre la potenza meccanica, utilizzando la 1.2, vale:

$$P_m = \omega_m T_{em} = k_t \phi_f \omega_m i_a \tag{1.5}$$

In regime stazionario si avrà che

$$P_m = P_e$$

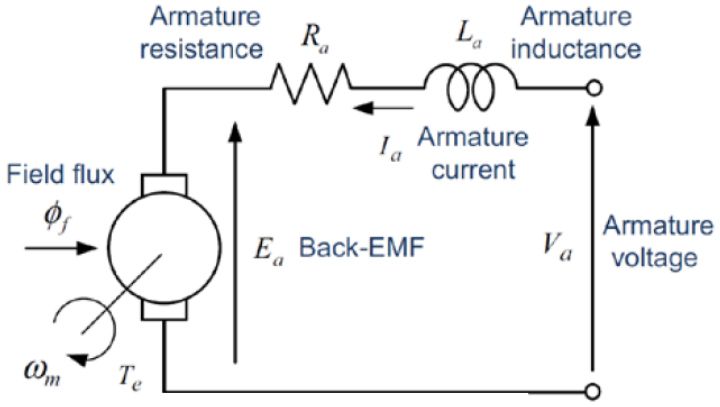


Figura 1.4. Circuito equivalente di un motore CC

In figura 1.4 si può osservare il circuito equivalente di un motore CC e attraverso la legge delle maglie di Kirchhoff si avrà:

$$V_a = e_a + R_a i_a + L_a \frac{di_a}{dt} \tag{1.6}$$

in cui R_a e L_a sono rispettivamente la resistenza e l'induttanza di armatura. Inoltre è possibile nuovamente definire la coppia elettromagnetica come:

$$T_{em} = J \frac{d\omega_m}{dt} + B\omega + T_L(t) \tag{1.7}$$

dove il termine T_L è la coppia di carico, mentre J e B sono l'inerzia equivalente e la frizione.

Regioni operative dei motori CC Considerando il circuito equivalente in figura 1.4 si possono definire le quattro regioni di operazione di un motore CC. In particolare un motore può agire come generatore quando frena riducendo la sua velocità. Per semplicità si considera il flusso ϕ_f costante. Per ridurre la velocità si riduce V_A e quando questa

sarà inferiore alla e_a la corrente avrà direzione opposta. Anche T_{em} cambierà direzione e l'energia cinetica associata sarà convertita in energia elettrica dal motore. Questa energia può essere assorbita dalla sorgente o venire dissipata in una resistenza. Durante la fase di frenata la polarità della back-emf non varia dato che il motore continuerà a girare nella stessa direzione seppur con una riduzione della velocità nel tempo. Nella figura 1.5 vengono riportate le quattro regioni di lavoro di un motore CC.

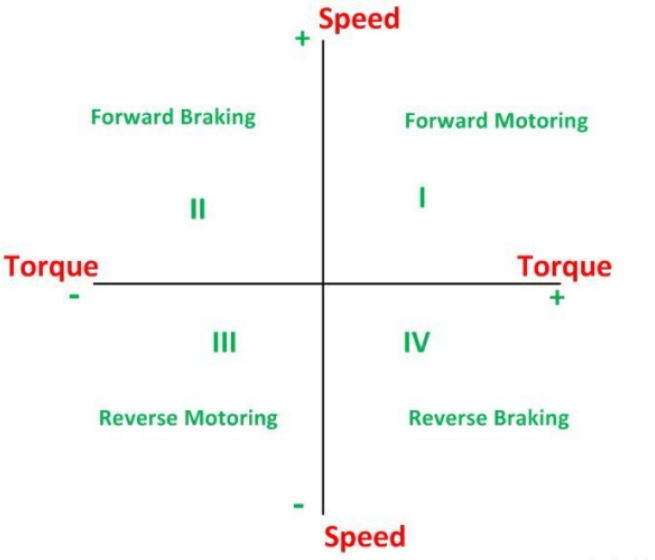


Figura 1.5. I quattro quadranti operativi di un motore CC

- Quadrante I: Motore avanti e $P_e > 0$
- Quadrante II: Freno generatore avanti e $P_e < 0$
- Quadrante III: Motore indietro e $P_e > 0$
- Quadrante IV: Freno generatore indietro e $P_e < 0$

1.4 Regolazione della velocità nei motori CC in regime stazionario

Per spiegare al meglio come è possibile modificare la velocità d un motore CC consideriamo il sistema in regime stazionario. Considerando le equazioni 1.1, 1.3, 1.6 e 1.7, ponendo le derivate rispetto al tempo nulle, trascurando la frizione e impostando la condizione in cui $k = k_t = k_e$ possiamo definire la velocità come:

$$\omega_m = \frac{1}{k\phi_f}\left(V_a - \frac{R_a T_{em}}{k\phi_f}\right) \tag{1.8}$$

Caratteristica velocità-coppia motori CC Attraverso l'equazione 1.8 è possibile ottenere la caratteristica "velocità-coppia" rappresentata in figura 1.6. Ricordando che T_L è la coppia ottenuta con un carico collegato al motore, è interessante notare che per T_L = 0 si ottiene la velocità massima raggiungibile dal motore e chiaramente, più aumenta il carico e più lento sarà.

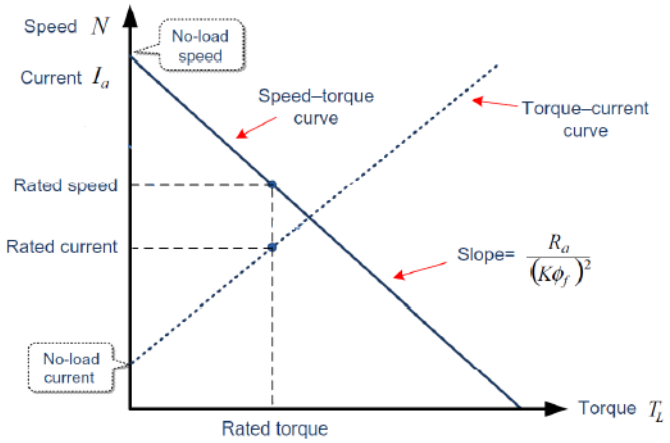


Figura 1.6. Caratteristica velocità-coppia di un motore CC

Controllo attraverso la tensione di armatura Questo tipo di controllo prevede l'aumento di velocità attraverso la modifica della tensione di armatura tenendo il flusso costante. Nella figura 1.7 si noti come la velocità sia lineare alla tensione di armatura. In regime stazionario la corrente di armatura I_a e la coppia non dovrebbero superare il proprio valore nominale (rated value).

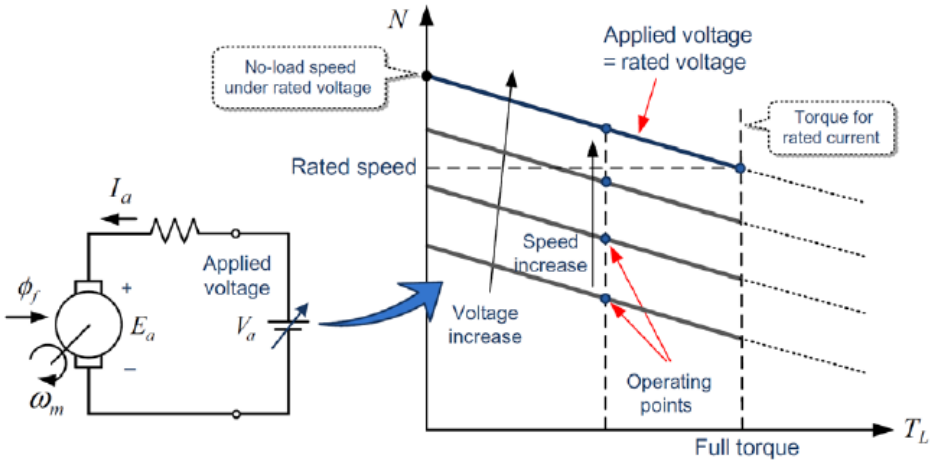


Figura 1.7. Caratteristica velocità-coppia con controllo di tensione di armatura

Questa tipologia di controllo è utilizzata nei motori CC a magnete permanente che, come detto nel sottoparagrafo 1.3, hanno un flusso costante.

Controllo attraverso il flusso di campo magnetico A differenza del precedente tipo di controllo, va innanzitutto detto che questo non può essere utilizzato nei motori a magnete permanente in quanto bisogna agire sul flusso. Considerando l'equazione 1.8 si noti che diminuendo il flusso magnetico, la velocità aumenterà. Bisogna però fare attenzione poiché se si considera invece l'equazione 1.2, ad una diminuzione del flusso corrisponde un decremento della coppia. Perciò quello che si può fare per ottenere una coppia costante è quello di aumentare la corrente i_a cercando comunque un buon compromesso in quanto un aumento di corrente porta ad una diminuzione dell'efficienza.

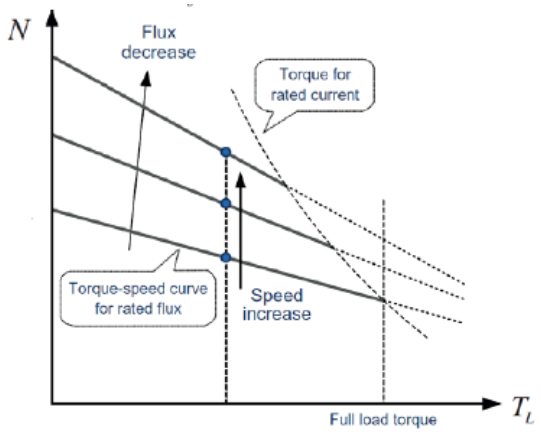


Figura 1.8. Caratteristica velocità-coppia con controllo di flusso di campo magnetico

Modalità operative per il controllo di velocità In definitiva quindi si può dire che i motori CC lavorano essenzialmente in due regioni. Dalla figura 1.9 se si suppone di poter avere una corrente di armatura i_a costante, nella prima zona, considerando l'equazione 1.2 e sapendo che il flusso deve essere costante, avremo una coppia anch'essa costante. Nella seconda regione invece, avremo la tensione massima raggiunta e quindi costante, di conseguenza anche la potenza non varierà. La prima regione dunque è definita "Coppia costante" e la seconda "Potenza costante"

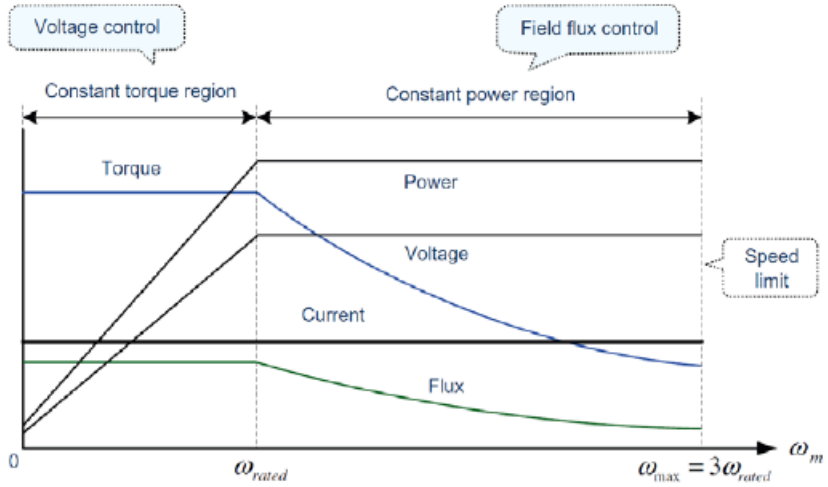


Figura 1.9. Regioni di lavoro per il controllo della velocità di un motore CC

1.5 Eccitazione del motore CC

Nei motori CC lo statore può essere realizzato non con magneti permanenti ma, similmente al rotore, con avvolgimenti su materiale ad alta permeabilità in cui viene fatta passare una corrente: questo circuito è detto di eccitazione. Esistono in letteratura essenzialmente due tipi di eccitazione:

- Eccitazione separata o indipendente
- Eccitazione serie

Per quanto riguarda l'eccitazione separata, questa è quella che è stata considerata nel paragrafo precedente in cui viene spiegato come poter modificare la velocità di un motore CC. In figura 1.10 è possibile osservare lo schema elettrico di un motore con eccitazione separata. In questo tipo di configurazione le tensioni e correnti che alimentano il circuito di armatura e il circuito di eccitazione sono regolabili in modo indipendente una dall'altra. Il motore ad eccitazione separata costituisce, oggi, la tipologia più comune di motore a corrente continua usato in ambito industriale. Nel motore ad eccitazione serie invece, i due

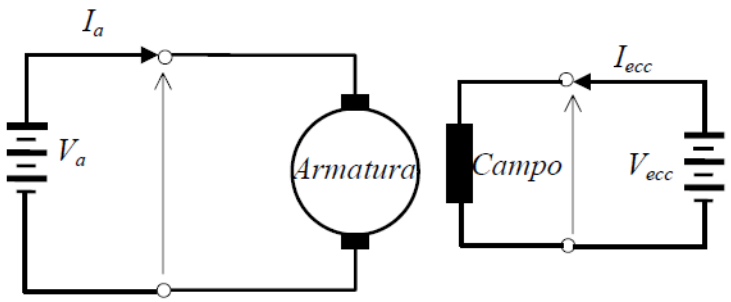


Figura 1.10. Schema di un motore CC con armatura ed eccitazione separata

avvolgimenti (armatura, campo) sono collegati in serie ed alimentati attraverso un'unica sorgente; essi sono quindi percorsi dalla stessa corrente. Questo tipo di collegamento condiziona le dimensioni dei conduttori dell'avvolgimento di eccitazione che dovranno essere adeguate a sopportare l'intera corrente di armatura. Questo tipo di collegamento ha trovato, in passato, particolare fortuna nel campo della trazione elettrica; oggi esso è caduto in disuso.[1]

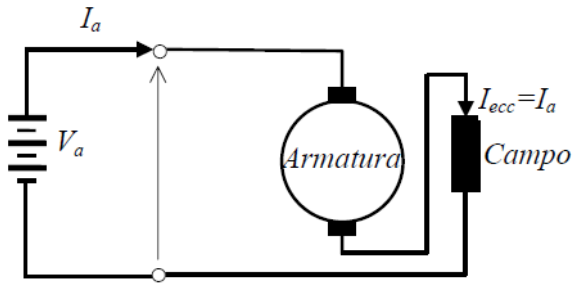


Figura 1.11. Schema di un motore CC con armatura ed eccitazione serie

1.6 Modello dinamico del motore CC

Il regime transitorio dei motori CC è caratterizzato, sia da grandezze elettriche che meccaniche, variabili nel tempo. Il funzionamento transitorio si verifica nel passaggio da una condizione di regime ad un'altra. Per poterla valutare al meglio, si considerino le equazioni 1.6 e 1.7 lavorando però nel dominio di Laplace.

$$V_a(s) = (R_a + sL_a)I_a(s) + E_a(s) = (R_a + sL_a)I_a(s) + k_e\phi_f\omega_m(s) \tag{1.9}$$

$$T_{em}(s) = (Js + B)\omega_m(s) + T_L = k_T\phi_f I_a(s) \tag{1.10}$$

Si ricava quindi la funzione di trasferimento

$$\frac{\omega_m}{V_a(s)} = \frac{\frac{k_e\phi_f}{(L_as+R_a)(Js+B)}}{1 + \left(\frac{k_e\phi_f k_T\phi_f}{(L_as+R_a)(Js+B)}\right)} \tag{1.11}$$

Ponendo successivamente ($k = k_T\phi_f = k_e\phi_f$) e B=0 si ottiene

$$\frac{\omega_m}{V_a(s)} = \frac{\frac{k}{JL_a}}{s^2 + \left(\frac{R_a}{L_a}\right)s + \left(\frac{k^2}{JL_a}\right)} = \frac{\frac{R_ak^2}{kL_aJR_a}}{s^2 + \left(\frac{R_a}{L_a}\right)s + \left(\frac{R_ak^2}{L_aJR_a}\right)} \tag{1.12}$$

Da questa equazione è possibile ottenere la costante di tempo elettromeccanica T_m e la costante di tempo elettrica T_a

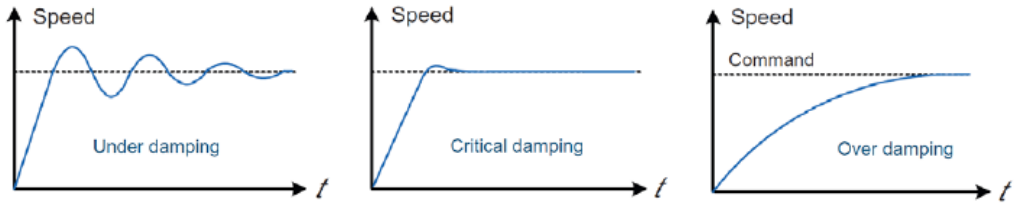
$$T_m = \frac{JR_a}{k^2} \tag{1.13}$$

$$T_a = \frac{L_a}{R_a} \tag{1.14}$$

Andando a sostituire queste due costanti di tempo nell'equazione 1.12 è possibile trovare due radici. In particolare, se $T_m \geq 4T_a$ si hanno due radici reali, contrariamente invece, si avranno due radici complesse. Un fattore molto importante che determina il tipo di transitorio è il "Damping ratio" definito come:

$$\zeta = \frac{1}{2}\sqrt{\frac{T_m}{T_a}} \tag{1.15}$$

Possiamo quindi avere tre particolari situazioni che sono riportate in figura 1.12.



(a) (b) (c)

Figura 1.12. (a)Under damping, (b)Critical damping, (c)Over damping

Nella situazione di Under damping, $\zeta<1$, si avrà un transitorio rapido ma con molte oscillazioni prima della stabilizzazione del segnale. Nella situazione di Critical damping invece $\zeta=1$ e si avrà un transitorio più lento e con una leggera oscillazione iniziale. Infine, nell'Over damping, $\zeta>1$ e si avrà un transitorio molto lento ma con oscillazioni nulle.

Il comportamento del transitorio dipende molto da tipo di sistema che si ha davanti, ma tipicamente nei motori CC si ha una situazione di Critical damping in modo tale da avere il giusto compromesso tra velocità e minime oscillazioni.[2]

All'atto pratico, il controllo della velocità in un motore CC deve essere effettuato attraverso delle retroazioni. In particolare esistono due anelli, il primo è denominato anello di velocità, mentre il secondo, che andrà direttamente ad agire sul controllo del motore, è detto anello di corrente. Risulta perciò molto importante valutare la funzione di trasferimento ad anello chiuso di un motore CC che include un controllore di corrente considerando lo schema a blocchi presente in figura 1.13. Si anticipa inoltre che per un anello di corrente il controllore utilizzato è di tipo PI (Proporzionale-Integrativo), più avanti comunque si entrerà nel dettaglio spiegandone il funzionamento.

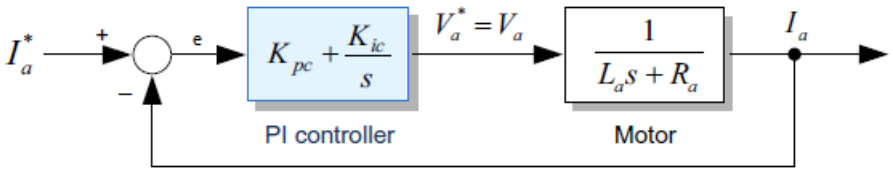


Figura 1.13. Schema a blocchi ad anello chiuso di un controllo in corrente

Sapendo che la f.d.t. dell'impianto equivale a:

$$Y(s) = \frac{I_a(s)}{V_a(s)} = \frac{1}{R_a + sL_a} \tag{1.16}$$

mentre quella del controllore di corrente vale:

$$K(s) = \frac{V_a(s)}{e(s)} = K_{pc} + \frac{K_{ic}}{s} \tag{1.17}$$

Si ottiene infine che la funzione di trasferimento del sistema ad anello chiuso è:

$$\frac{I_a}{I_a^*} = \frac{K(s)Y(s)}{1 + K(s)Y(s)} \tag{1.18}$$

Facendo le opportune sostituzioni e manipolazioni si ottiene:

$$\frac{I_a}{I_a^*} = \frac{K_{pc}s + K_{ic}}{s^2 L_a + s(K_{pc} + R_a) + K_{ic}} \tag{1.19}$$

Capitolo 2

Convertitore AC/DC totalmente controllato

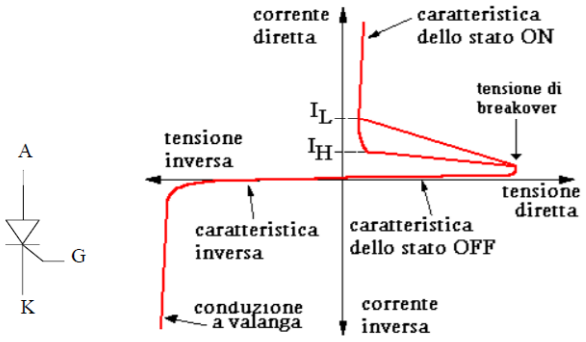
2.1 SCR (Silicon Controlled Rectifier)

L'SCR (Silicon Controlled Rectifier) è un componente elettronico pressoché equivalente al diodo, con la sola differenza che la conduzione diretta avviene solamente in seguito all'applicazione di un opportuno segnale di innesco su un terzo terminale denominato gate.[3]

Il simbolo circuitale di un SCR è mostrato in figura 2.1(a) in cui la corrente principale scorre dall'anodo al catodo.

Nella figura 2.1(b) viene invece mostrata la caratteristica $i - v$. Nel primo quadrante il diodo non conduce finchè non raggiunge una tensione detta di "breakdown" e superato questo valore, la curva è come quella di un classico diodo. La I_L viene detta "corrente di latching" ovvero la corrente necessaria per innescare la conduzione, mentre la I_H è la "corrente di mantenimento" minima per mantenere la conduzione. Nel terzo quadrante infine è presente la caratteristica inversa con la conduzione a valanga una volta che la tensione V_{ak} raggiunge il valore negativo V_z tale per cui si ha l'effetto Zener.[4]

Questo componente permette di avere una tensione continua variabile a seconda di dove viene dato l'impulso e quindi fatto innescare il tiristore. Un'applicazione possibile è mostrata in figura 2.2, dove l'SCR viene fatto condurre in un determinato istante nel semi-periodo positivo. L'angolo α viene chiamato "$firing\ angle$" e non è nient'altro che il ritardo con cui viene dato l'impulso rispetto all'inizio del semi-periodo.



(a) (b)

Figura 2.1. Simbolo SCR e caratteristica $i - v$

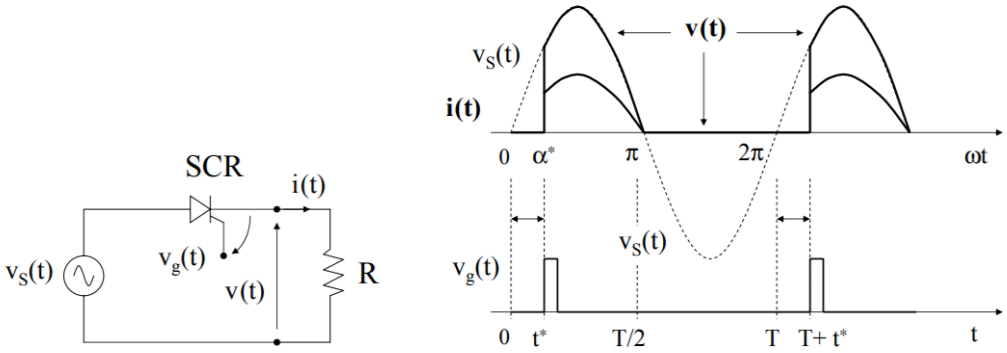


Figura 2.2. Andamento della tensione di uscita con un generico "firing angle α"

2.2 Raddrizzatore trifase totalmente controllato

Nel mondo dei motori CC è ovviamente necessario poter avere una tensione continua variabile. Per questo tipo di applicazione gli SCR risultano essere ottimali poichè come detto nel paragrafo 2.1 è possibile avere differenti valori di tensione in uscita.

In figura 2.3 è possibile osservare lo schema di un *raddrizzatore trifase totalmente controllato*. In ingresso troviamo tre tensioni in alternata alla frequenza di 50-60Hz.

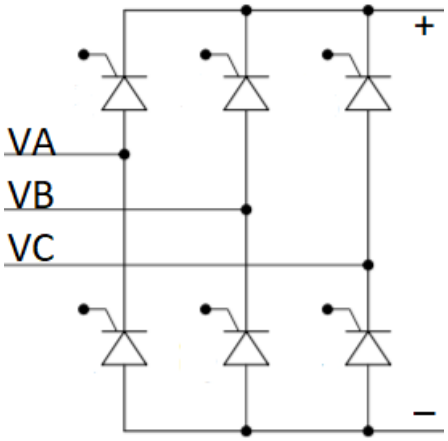


Figura 2.3. Ponte trifase SCR totalmente controllato

2.2.1 Funzionamento sistema trifase

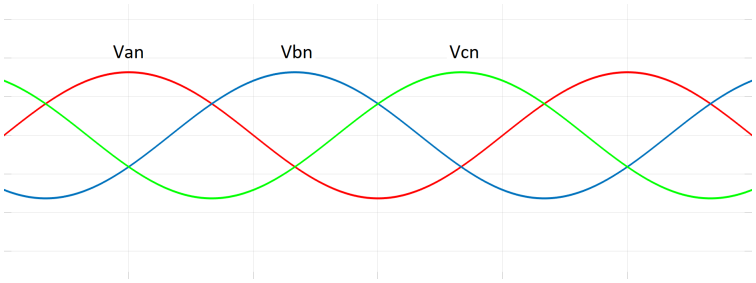
Prima di entrare nel dettaglio del raddrizzatore è bene spiegare come funziona un sistema trifase. Questo è composto da tre tensioni di uguale ampiezza e periodo, (tipicamente 50-60Hz) sfasate tra loro di 120°. Possiamo dunque definire quelle che sono chiamate le tensioni di fase:

$$V_a = V_m sin(\omega t);$$
$$V_b = V_m sin(\omega t - 120°);$$
$$V_c = V_m sin(\omega t - 240°);$$

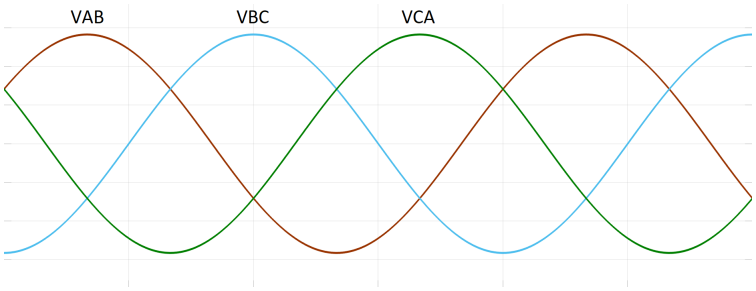
Possiamo inoltre definire le tensioni concatenate:

$$V_a - V_b = \sqrt{3} V_m sin(\omega t + 30°);$$
$$V_b - V_c = \sqrt{3} V_m sin(\omega t - 90°);$$
$$V_c - V_a = \sqrt{3} V_m sin(\omega t - 210°);$$

Le figure 2.4 sono state ottenute con una simulazione facendo uso di *Simulink* e hanno il solo scopo di raffigurare l'andamento delle tensioni appena definite.



(a)



(b)

Figura 2.4. Tensioni di fase (a) e tensioni concatenate (b)

In figura 2.5 sono rappresentate la V_{an} e V_{AB} in cui è possibile notare lo sfasamento di 30° che c'è tra le due tensioni

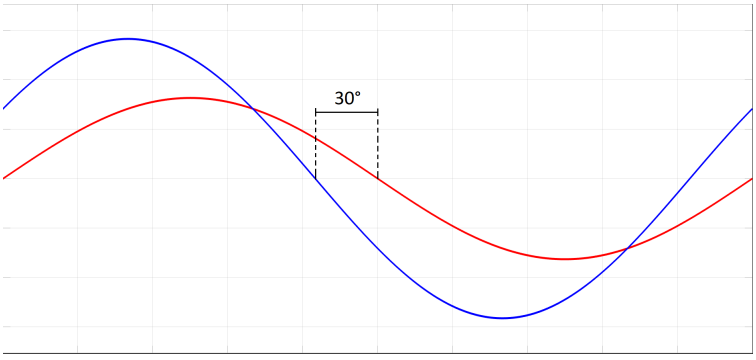


Figura 2.5. Tensione di fase (rossa) e tensione concatenata (blu)

2.2.2 Funzionamento raddrizzatore trifase

Il circuito preso in analisi è quello riportato in figura 2.6 in cui possiamo trovare sul carico una corrente $i_d = I_D(t)$. La numerazione degli SCR è 1,3,5 per la parte positiva e 4,6,2

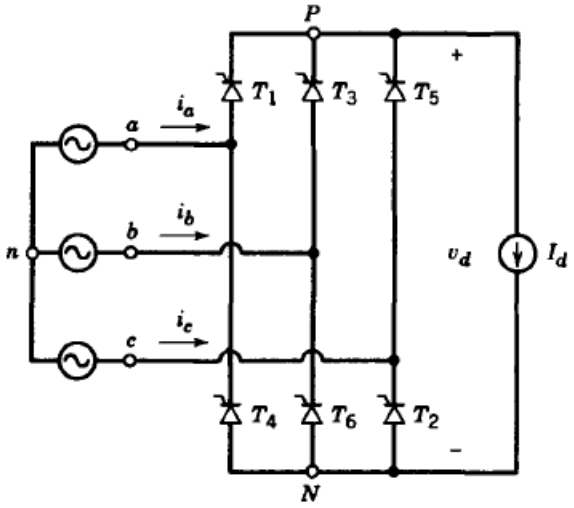


Figura 2.6. Ponte trifase SCR totalmente controllato

per la parte negativa.

Per α=0 tutti gli SCR agiscono come un classico diodo. Nella figura 2.7 è riportata la sequenza degli SCR, inoltre, nella tabella 2.1 sono riportati gli angoli di innesco di ogni tiristore al variare di α.

T1	$30° + \alpha$
T2	$90° + \alpha$
T3	$150° + \alpha$
T4	$210° + \alpha$
T5	$270° + \alpha$
T6	$330° + \alpha$

Tabella 2.1. Angoli di inneschi degli SCR al variare di α rispetto le tensioni di ingresso

Sempre in figura 2.7 è possibile notare che la conduzione di ogni SCR avviene per 120° avendo così in ogni instante una coppia di SCR che conduce permettendo il passaggio di due tensioni di fase. Infatti se prendiamo in esame i tiristori T1 e T6 avremo in uscita la tensione V_{AB} dato che l'SCR T1 è connesso alla tensione V_{an} mentre T6 alla tensione V_{bn}. Inoltre, è possibile notare che si ha una commutazione ogni 60° formando cosi le coppie di SCR T1/T2, T2/T3, T3/T4, T4/T5, T5/T6 e T6/T1.

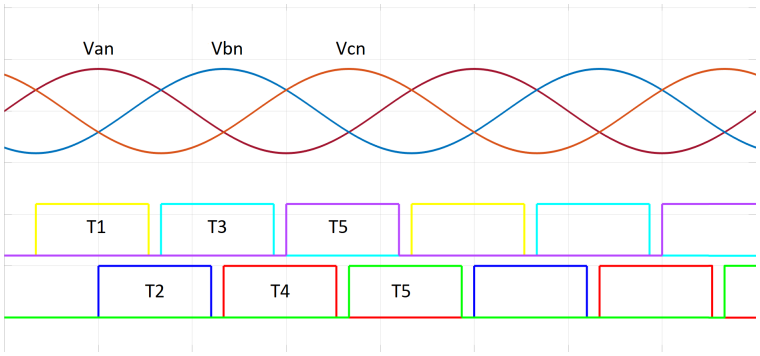


Figura 2.7. Posizione dei segnali di gate con α=0 rispetto alle tensioni di fase

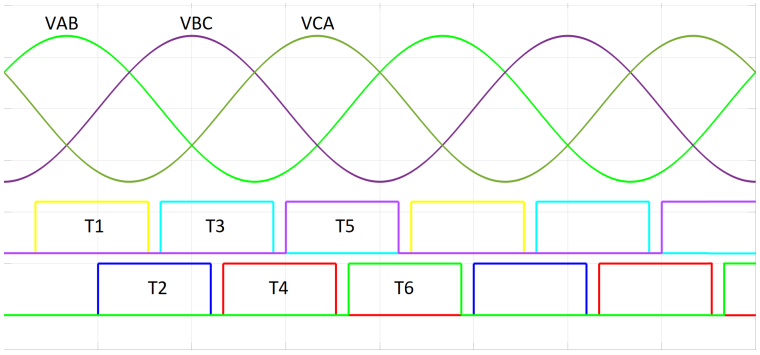


Figura 2.8. Posizione dei segnali di gate con α=0 rispetto alle tensioni concatenate

La posizione degli inneschi, può essere anche studiata rispetto alle tensioni concatenate, infatti, considerando quanto detto nel paragrafo 2.2.1 basterà aggiungere 30° per ogni SCR alla tabella 2.1 dato che le tensioni concatenate sono anticipate di 30° rispetto alle tensioni di fase. Va considerato però che questi angoli hanno come riferimento $t = 0$ se consideriamo l'asse del tempo. Nello specifico invece, ogni SCR viene innescato rispetto alla "propria" semionda.

In particolare:

- T1 rispetto alla semionda positiva di V_{an}
- T2 rispetto alla semionda negativa di V_{cn}

- T3 rispetto alla semionda positiva di V_{bn}
- T4 rispetto alla semionda negativa di V_{an}
- T5 rispetto alla semionda positiva di V_{cn}
- T6 rispetto alla semionda negativa di V_{bn}

Equazioni del ponte raddrizzatore trifase

Ls=0 Ricordando che il *firing angle* è definito come α e sotto la condizione di α=0 è possibile definire la tensione media massima disponibile sul carico del circuito in figura 2.6

$$V_o = \frac{3\sqrt{2}V_{LL}}{\pi} = 1.35V_{LL} \tag{2.1}$$

Dove V_{LL} è il valore rms di una tensione concatenata.

La tensione di uscita dipende fortemente dal "*firing angle*" permettendo così di avere una V_o regolabile. In particolare è possibile dire che la tensione media sul carico vale:

$$V_{d\alpha} = V_o - \frac{A_\alpha}{\pi/3} \tag{2.2}$$

In figura 2.9a sono riportati gli andamenti delle tensioni di fase, mentre in 2.9d è riportato l'andamento della V_o con un α non nullo.

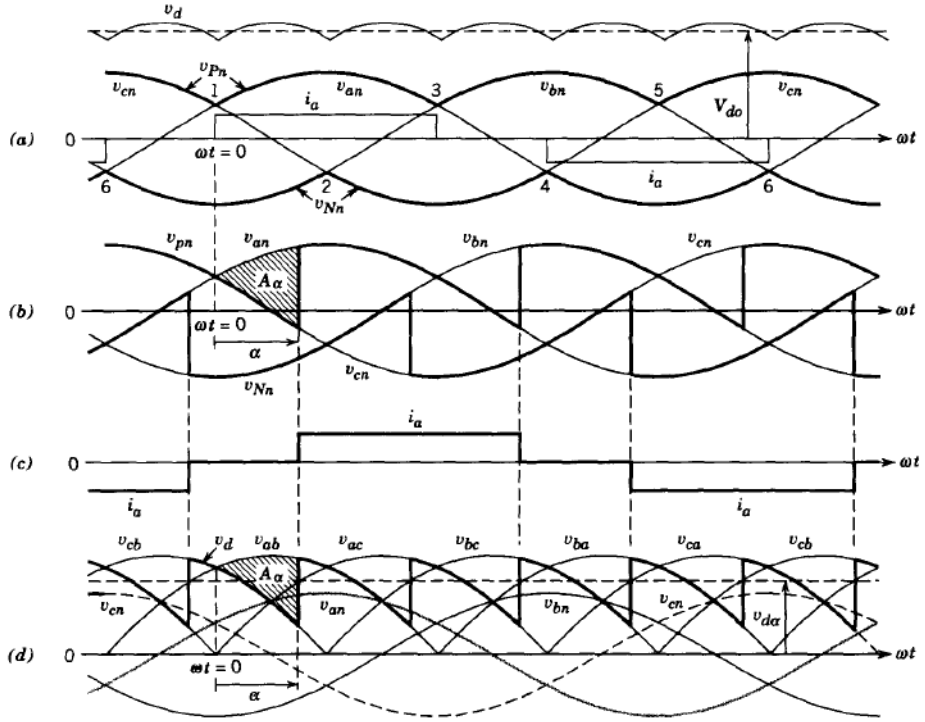


Figura 2.9. Forme d'onda del convertitore

Nell'equazione 2.2 il parametro A_α è definito come "area volt-radianti" (ogni 60°) e osservando la figura 2.9b non è nient'altro che l'integrale della tensione $V_{AC} = V_{an} - V_{cn}$,

$$V_{AC} = \sqrt{2} V_{LL} sin(\omega t) \tag{2.3}$$

Quindi

$$A_\alpha = \int_0^\alpha \sqrt{2} V_{LL} sin(\omega t) d(\omega t) = \sqrt{2} V_{LL} (1 - cos(\alpha)) \tag{2.4}$$

Sostituendo quindi A_α nell'equazione 2.2 e usando la 2.1 si ottiene:

$$V_{d\alpha} = \frac{3\sqrt{2} V_{LL} cos(\alpha)}{\pi} = 1.35 V_{LL} cos(\alpha) = V_o cos(\alpha) \tag{2.5}$$

Prendendo in esame il circuito in figura 2.6 è possibile anche valutare la potenza media:

$$P = V_o I_d = 1.35 V_{LL} I_d cos(\alpha)$$

In figura 2.10 sono riportati gli andamenti della V_o al variare dell'angolo α.

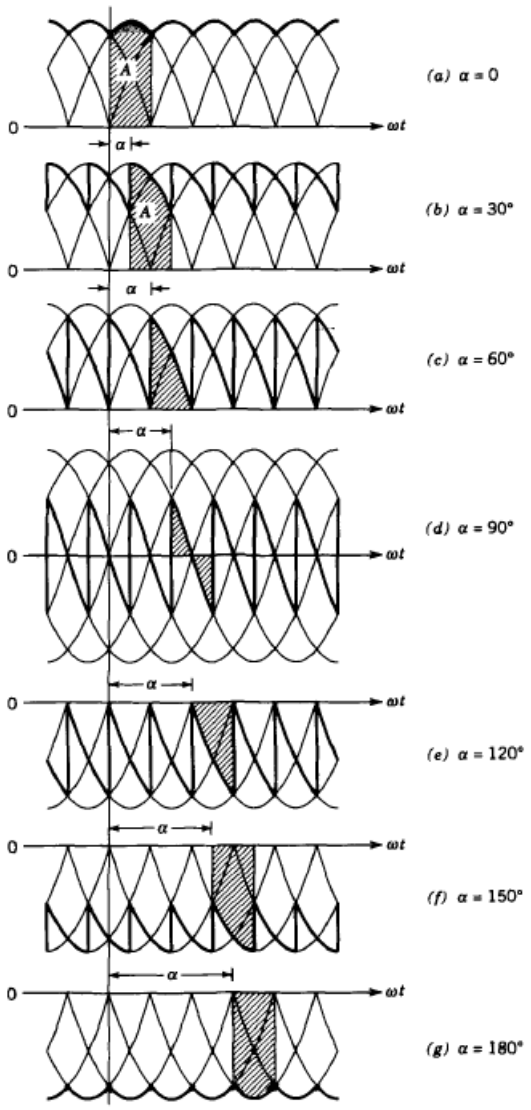


Figura 2.10. Andamento della tensione di uscita per differenti valori di α

Un altro parametro molto importante da valutare è la corrente di fase, come si può notare dalla figura 2.11 viene raffigurata la corrente i_a e anch'essa dipende fortemente dall'angolo α.

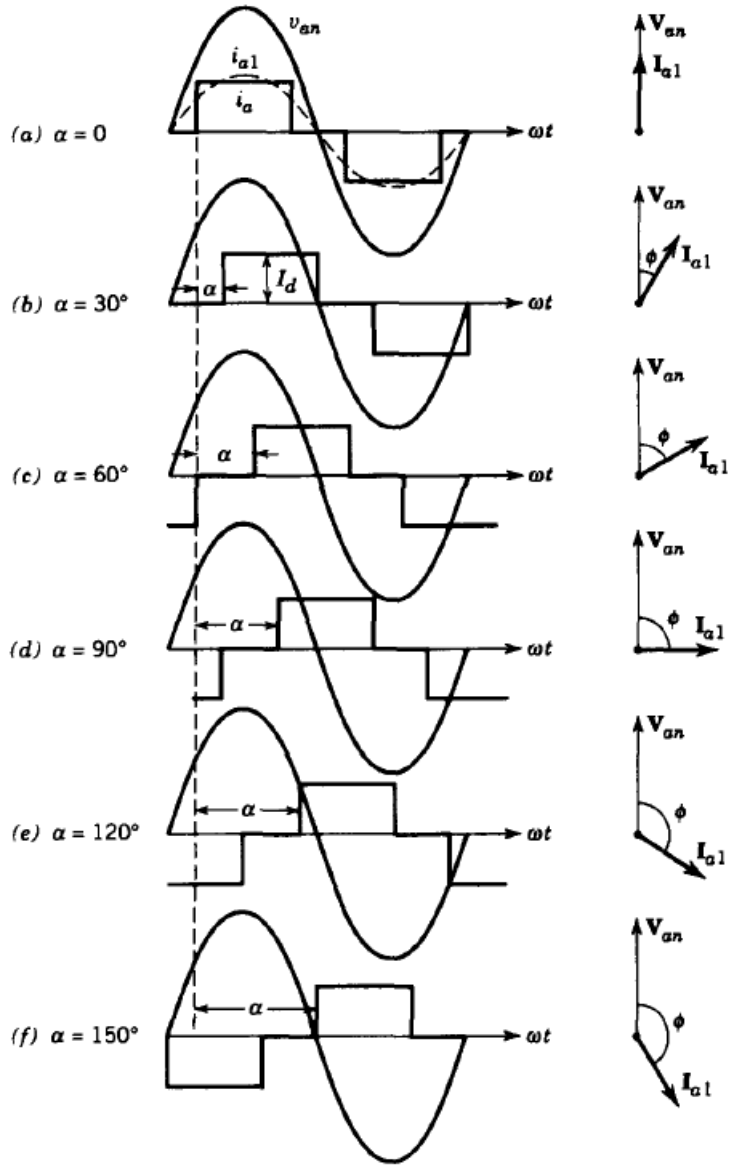


Figura 2.11. Andamento della corrente di fase per differenti valori di α

Il sistema trifase in analisi, come detto ad inizio paragrafo lavora a 50Hz, questo vuol dire che la tensione di uscita avrà una frequenza 6 volte maggiore, ovvero pari a 300Hz. Nella figura 2.10 è possibile notare le tensioni concatenate e l'angolo α rispetto a queste tensione. In particolare è possibile soffermarsi sulla figura 2.10b in cui α=30°. La commutazione avviene esattamente sul valore massimo della tensione concatenata che è a 90°, questo vuol dire che per α nullo, l'innesco del tiristore avviene a 60° rispetto al passaggio per lo zero della semionda positiva di una tensione concatenata.

Ls $\neq$ 0 Il circuito preso in analisi è quello riportato in figura 2.12

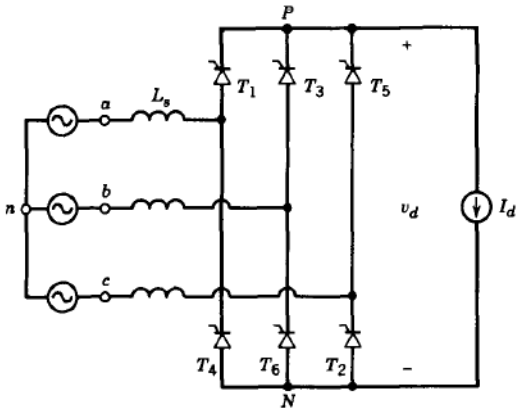


Figura 2.12. Circuito con L_s

Rispetto al caso precedente, in questo circuito troviamo le induttanze di linea, che all'atto pratico danno un contributo che non può essere trascurato.

La commutazione della corrente, ora, impiega un intervallo di tempo u. Se consideriamo infatti la conduzione di T5 e T6, la commutazione che avverrà sarà da T5 a T1. Nell'immagine 2.13 è possibile osservare questo fenomeno in cui sono riportati gli andamenti delle correnti di T1 e T5.

Durante l'intervallo di tempo u, T1 e T5 conducono contemporaneamente e le due tensioni V_{an} e V_{cn} sono cortocircuitate tramite L_s. Come per il caso in cui $L_s = 0$ anche qui è possibile valutare la tensione media continua che si ha sul carico. Prendendo in esame la figura 2.12 si valuta la tensione v_{Pn} sapendo che nell'intervallo $\alpha < \omega t < \alpha + u$

$$v_{Pn} = v_{an} - v_{L_s} \tag{2.6}$$

Dove

$$v_{L_s} = L_s \frac{di_a}{dt} \tag{2.7}$$

Osservando invece la figura 2.13 è possibile identificare A_u che ricordiamo essere "l'area volt-radianti" e vale

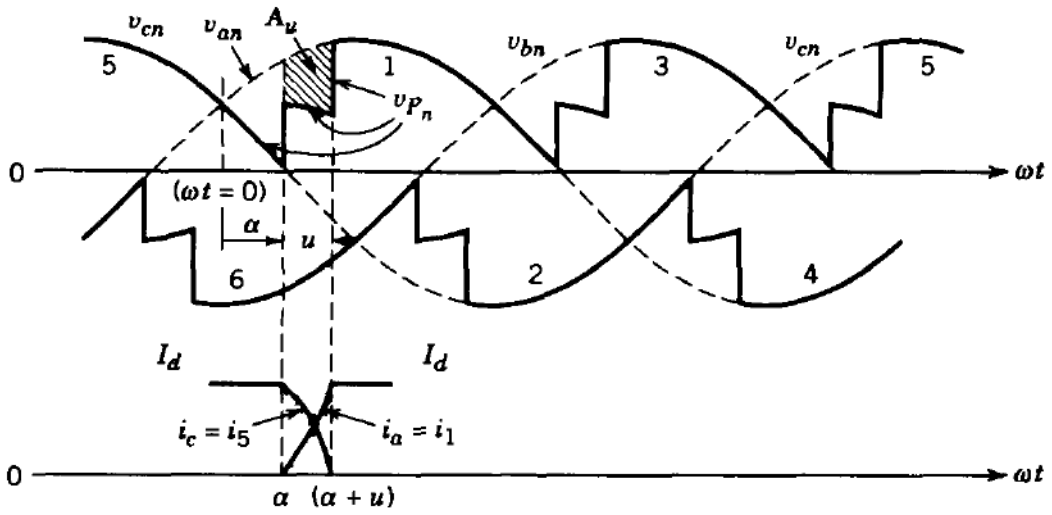


Figura 2.13. Andamento della commutazione in presenza di L_s

$$A_u = \int_{\alpha}^{\alpha+u} v_{L_s} d(\omega t) \tag{2.8}$$

Sostituendo 2.7 in 2.8 e tenendo in considerazione che i_a cambia il suo valore da 0 ad I_d nell'intervallo $\alpha < \omega t < \alpha + u$ si ottiene

$$A_u = \omega L_s \int_0^{I_d} d(i_a) = \omega L_s I_d \tag{2.9}$$

Infine l'equazione finale è ottenuta ricordando l'espressione 2.5 in cui questa viene ridotta di $A_u/(\pi/3)$

$$V_d = \frac{3\sqrt{2}}{\pi} V_{LL} cos(\alpha) - \frac{3\omega L_s I_d}{\pi}$$

Convertitore in pratica: modalità discontinua Il circuito preso in esame è quello in figura 2.14

Il convertitore preso in esame solitamente viene utilizzato per il controllo di motori CC. Osservando la figura 2.14 si noti che come carico viene utilizzato il circuito equivalente di un motore spiegato nel paragrafo 1.3.

La corrente i_d diventa discontinua sotto un certo valore per un α dato. In più va detto che per grandi valori di E_d si avrà una piccola I_d. Nella figura sottostante viene mostrato l'andamento della corrente.[5]

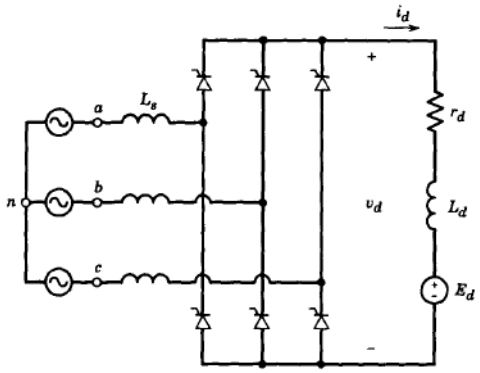


Figura 2.14. Circuito del convertitore in pratica

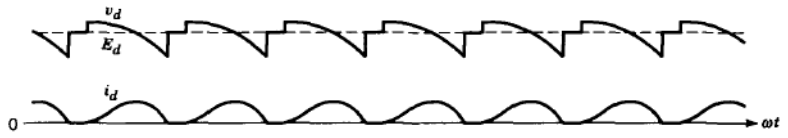


Figura 2.15. Forma d'onda della corrente in modalità discontinua

Capitolo 3

Scheda di controllo

Un microcontrollore (MCU) utilizza tecniche di microelettronica per ridurre in un piccolo pacchetto (o package) vari componenti quali CPU (Central Processing Unit) e memoria. Presenta diversi pin di ingresso e uscita, attraverso i quali è possibile interagire con il mondo esterno. Il microcontrollore così com'è stato realizzato non fa nulla, è necessario programmarlo attraverso un insieme di istruzioni.

Si può pensare a un microcontrollore come un piccolo computer dove è possibile collegare un display, alcuni pulsanti, sensori e motori, il tutto programmato affinché eseguano determinate funzioni. Questo componente ormai è utilizzato ovunque, dagli elettrodomestici al campo aerospaziale per sistemi di controllo digitale complessi.

La selezione di un microcontrollore per un progetto è un compito da non sottovalutare in quanto devono essere non solo considerati i relativi fattori tecnici hardware e software, ma anche i tempi e costi che la realizzazione di un progetto.

3.1 Arduino MEGA 2560 R3

Per la realizzazione di questo progetto è stata scelta l'Arduino MEGA 2560 R3 che è una scheda di valutazione basata sul microcontrollore ATMega2560. La scheda è dotata di 54 input/output digitali di cui 15 possono essere usati come output PWM. Sono presenti anche 16 input analogici e tutte le operazioni sono scandite da un oscillatore al cristallo ad una frequenza di 16MHz. Infine la scheda è dotata di un pin Serial Data (SDA) e uno Serial Clock (SCL) molto utili per la comunicazione $I2C$. La scheda deve essere ovviamente programmata attraverso il software dedicato "Arduino Software (IDE)". In più la scheda è composta da una SRAM di 8 KB ed una EEPROM di 4 KB.[6]

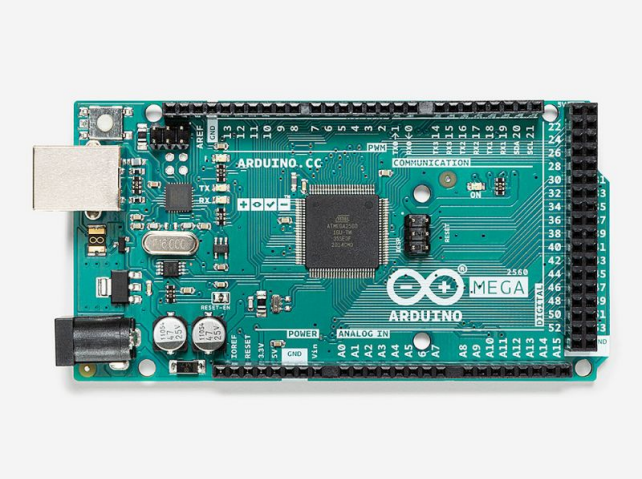


Figura 3.1. Scheda di valutazione Arduino Mega 2560 R3

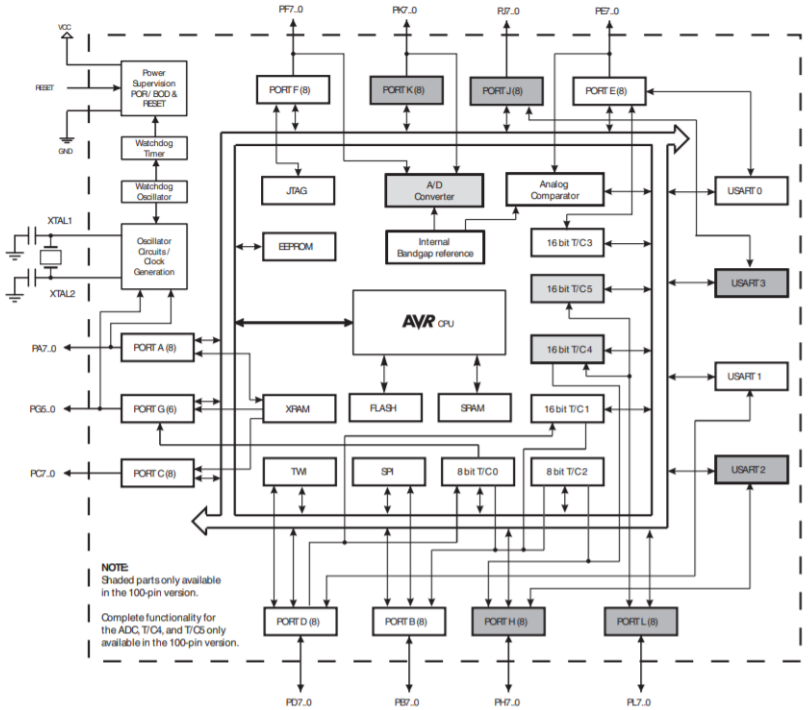


Figura 3.2. Schema a blocchi del microcontrollore ATMega2560

3.2 Timer e Interrupt

Le interrupt sono istruzioni che interrompono il normale ciclo del microcontrollore per essere eseguite. Durante la realizzazione di questo progetto sono stati utilizzati due tipi di interrupt: gli esterni e quelli dovuti dai timer interni della scheda. Riguardo i Timer, nella scheda utilizzata sono a disposizione sei Timer di cui quattro a 16 bit e due a 8 bit. Più avanti vedremo come le interrupt sono state utilizzate per la realizzazione del progetto, nel frattempo è importante conoscere le caratteristiche dei timer.

TIMER3 Il TIMER3 è uno dei timer a 16bit presenti sul microcontrollore che è stato utilizzato (gli altri due sono TIMER1 e TIMER4). Per l'uso dei timer è importante la gestione dei registri interni, in particolare è possibile trovare il registro "OCR3A" in cui scrivere al suo interno un valore. Nell'istante in cui il timer sarà uguale a quel valore verrà generato un interrupt. Nella figura 3.3 è riportato un piccolo diagramma di flusso per comprendere meglio come viene generato l'interrupt.

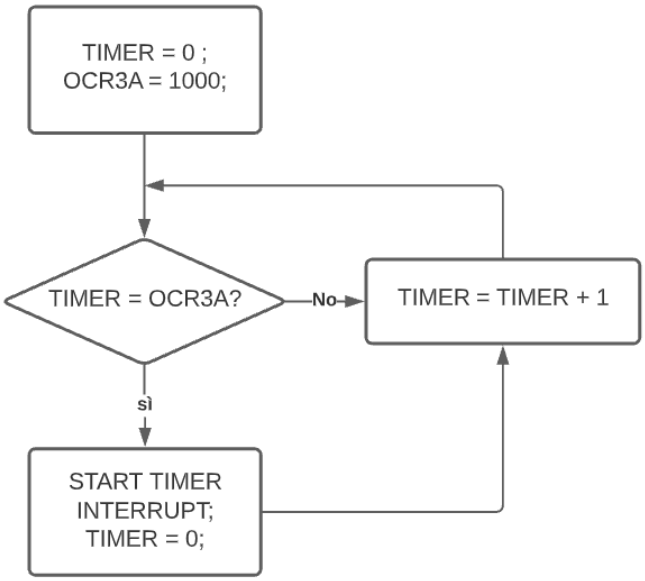


Figura 3.3. Digramma di flusso per generazione timer interrupt

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	PCINT0	Pin Change Interrupt Request 0
11	$0014	PCINT1	Pin Change Interrupt Request 1
12	$0016(3)	PCINT2	Pin Change Interrupt Request 2
13	$0018	WDT	Watchdog Time-out Interrupt
14	$001A	TIMER2 COMPA	Timer/Counter2 Compare Match A
15	$001C	TIMER2 COMPB	Timer/Counter2 Compare Match B
16	$001E	TIMER2 OVF	Timer/Counter2 Overflow
17	$0020	TIMER1 CAPT	Timer/Counter1 Capture Event
18	$0022	TIMER1 COMPA	Timer/Counter1 Compare Match A
19	$0024	TIMER1 COMPB	Timer/Counter1 Compare Match B
20	$0026	TIMER1 COMPC	Timer/Counter1 Compare Match C
21	$0028	TIMER1 OVF	Timer/Counter1 Overflow
22	$002A	TIMER0 COMPA	Timer/Counter0 Compare Match A
23	$002C	TIMER0 COMPB	Timer/Counter0 Compare match B
24	$002E	TIMER0 OVF	Timer/Counter0 Overflow
25	$0030	SPI, STC	SPI Serial Transfer Complete
26	$0032	USART0 RX	USART0 Rx Complete
27	$0034	USART0 UDRE	USART0 Data Register Empty
28	$0036	USART0 TX	USART0 Tx Complete
29	$0038	ANALOG COMP	Analog Comparator
30	$003A	ADC	ADC Conversion Complete

Figura 3.4. Interrupt vector table (parte 1)

Vector No.	Program Address(2)	Source	Interrupt Definition
31	$003C	EE READY	EEPROM Ready
32	$003E	TIMER3 CAPT	Timer/Counter3 Capture Event
33	$0040	TIMER3 COMPA	Timer/Counter3 Compare Match A
34	$0042	TIMER3 COMPB	Timer/Counter3 Compare Match B
35	$0044	TIMER3 COMPC	Timer/Counter3 Compare Match C
36	$0046	TIMER3 OVF	Timer/Counter3 Overflow
37	$0048	USART1 RX	USART1 Rx Complete
38	$004A	USART1 UDRE	USART1 Data Register Empty
39	$004C	USART1 TX	USART1 Tx Complete
40	$004E	TWI	2-wire Serial Interface
41	$0050	SPM READY	Store Program Memory Ready
42	$0052(3)	TIMER4 CAPT	Timer/Counter4 Capture Event
43	$0054	TIMER4 COMPA	Timer/Counter4 Compare Match A
44	$0056	TIMER4 COMPB	Timer/Counter4 Compare Match B
45	$0058	TIMER4 COMPC	Timer/Counter4 Compare Match C
46	$005A	TIMER4 OVF	Timer/Counter4 Overflow
47	$005C(3)	TIMER5 CAPT	Timer/Counter5 Capture Event
48	$005E	TIMER5 COMPA	Timer/Counter5 Compare Match A
49	$0060	TIMER5 COMPB	Timer/Counter5 Compare Match B
50	$0062	TIMER5 COMPC	Timer/Counter5 Compare Match C
51	$0064	TIMER5 OVF	Timer/Counter5 Overflow
52	$0066(3)	USART2 RX	USART2 Rx Complete
53	$0068(3)	USART2 UDRE	USART2 Data Register Empty
54	$006A(3)	USART2 TX	USART2 Tx Complete
55	$006C(3)	USART3 RX	USART3 Rx Complete
56	$006E(3)	USART3 UDRE	USART3 Data Register Empty
57	$0070(3)	USART3 TX	USART3 Tx Complete

Figura 3.5. Interrupt vector table (parte 2)

Nelle immagini appena riportate è possibile osservare quella che è chiamata "Interrupt Vector Table" ed in particolare sono stati evidenziati gli interrupt utilizzati in questo progetto di tesi. Quelli esterni in particolare sono stati utilizzati per trovare il fronte di salita e discesa delle onde quadre, mentre quelli interni, dovuti ai timer, sono stati necessari per generare gli impulsi (vedi cap. 5). Per quanto riguarda la generazione di un interrupt su Arduino, esiste una precisa funziona che permette di generarlo a seconda del fronte che si vuole trovare, in particolare:

- CHANGE: genera un interrupt ogni volta che un fronte è rilevato
- RISING: genera un interrupt ogni volta che un fronte di salita è rilevato
- FALLING: genera un interrupt ogni volta che un fronte di discesa è rilevato

Codice 3.1. Funzione utilizzata per generare interrupt esterno

```
//setup
int x;
attachInterrupt(digitalPinToInterrupt(2), Prova, RISING)
```

```
void Prova()
{
   digitalWrite(4, HIGH);
}
```

Il primo parametro identifica il Pin digitale sul quale verrà mandata l'onda d'ingresso di cui si vuole trovare il fronte di salita in questo caso, mentre "*Prova*" è la ISR che svolge il microcontrollore quando trova il fronte e nell'esempio riportato nel codice 3.1 viene semplicemente impostato il pin digitale 4 a livello logico alto. É importante anche che la ISR sia il più breve possibile poiché quando viene svolta, tutto il resto del codice viene messo in pausa. In più riguardo il primo parametro della funzione, bisogna utilizzare specifici pin che dipendono dal modello della scheda.

BOARD	INT.0	INT.1	INT.2	INT.3	INT.4	INT.5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
32u4 based (e.g Leonardo, Micro)	3	2	0	1	7	

Figura 3.6. Pin che possono essere utilizzati per gli interrupt esterni

3.3 Prescaler

Un altro registro molto importante è quello del "Prescaler". Il Prescaler è la possibilità di aumentare la frequenza del timer. Tutti i timer lavorano ovviamente con il clock interno del microcontrollore che ricordiamo essere 16MHz corrispondendo ad un periodo di 62.5ns. Prendendo in considerazione il TIMER3 il valore del prescaler presente sul chip ATmega2560 viene settato sul registro TTCR3B ed in particolare sui bit 0:2. Nella figura 3.7 è possibile osservare i possibili valori che si possono scegliere.

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

Figura 3.7. Tabella per il prescaler

Per chiarire meglio come funziona il prescaler è importante fare un esempio. Si supponga di impostare il prescaler nullo, questo vuol dire che ogni colpo di clock interno corrisponde ad un aumento del timer e perciò anche esso aumenterà di una unità con una frequenza di 16MHz.

Se invece si sceglie un prescaler pari a 8 vuol dire che ogni otto colpi di clock interno si avrà un incremento del timer, che questa volta aumenterà con una frequenza di 16MHz/8 = 2MHz.

Risulta molto importante anche valutare il valore utile da inserire nel registro "OCR3A" che, nel caso generico è possibile calcolare con la seguente formula:

$$valore\ utile = \frac{Periodo\ interrupt}{Prescaler \times \frac{1}{f}} \tag{3.1}$$

Supponendo ora di voler generare un interrupt ogni 10ms:

Prescaler	valore necessario
Prescaler nullo	160000
Prescaler 8	20000

Ricordando che il timer è composto da 16 bit, vuol dire che il valore massimo raggiungibile sarà $2^{16} - 1 = 65535$, perciò la scelta migliore è quella di utilizzare un prescaler pari a 8. Il valore che si ottiene dall'equazione 3.1 è quello che andrà inserito nel registro "OCR3A". E' comunque possibile utilizzare prescaler maggiori, ma vorrebbe dire avere una precisione minore.

3.4 Pin utilizzati nel progetto

La scheda di valutazione Arduino Mega 2650 R3 è provvista di una grande quantità di pin digitali e analogici che si sono rivelati fondamentali per una buona riuscita del progetto. In particolare è stato fatto uso di 12 pin digitali, (Interrupt esterni, generazione impulsi, controlli e gestione allarmi) 3 pin analogici utili per le letture del potenziometro di

riferimento e le due retroazioni di velocità e corrente e i pin SCL e SDA utili per sfruttare un piccolo display lcd 20x4. Per quanto concerne il corretto utilizzo dei pin digitali, questi possono chiaramente essere letti e scritti attraverso delle semplici funzioni presenti sul software dedicato.

Codice 3.2. Funzione per gestire pin digitali

```
//SETUP
pinMode(5, OUTPUT); //Definisco il pin digitale 5 come output

//LOOP
digitalWrite(5, HIGH); // Imposto il pin a 5V
delay(1000);           // Attendo un secondo
digitalWrite(13, LOW); // Imposto il pin a 0V
delay(1000);
//END LOOP

//In questo modo se connetto un piccolo Led al pin questo lampeggerà con
    intervalli di 1 sec
```

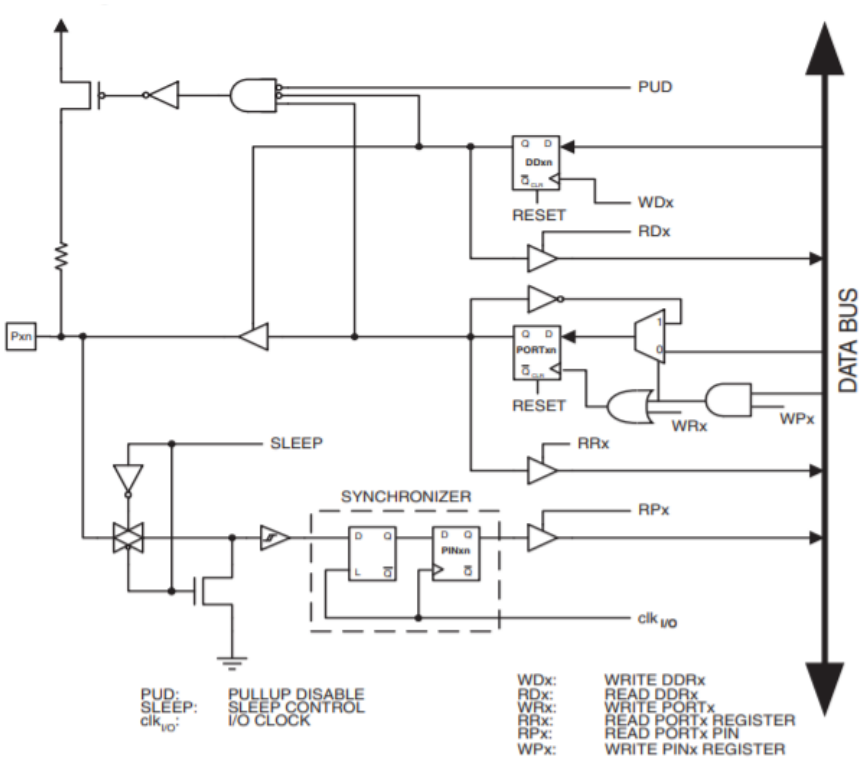


Figura 3.8. Schema generale di I/O digitale

Riguardo invece la lettura dei valori analogici bisogna utilizzare la seguente funzione

Codice 3.3. Funzione per lettura pin analogico

```
//setup
int x;

//LOOP
x = analogRead(A5);
//END LOOP
```

Bisogna prestare attenzione a come viene interpretata dall'utente una lettura analogica. La scheda di valutazione accetta in ingresso tensioni comprese tra 0V e 5V ed essendo provvista di canali ADC da 10 bit restituirà valori compresi fra 0 e 1023.

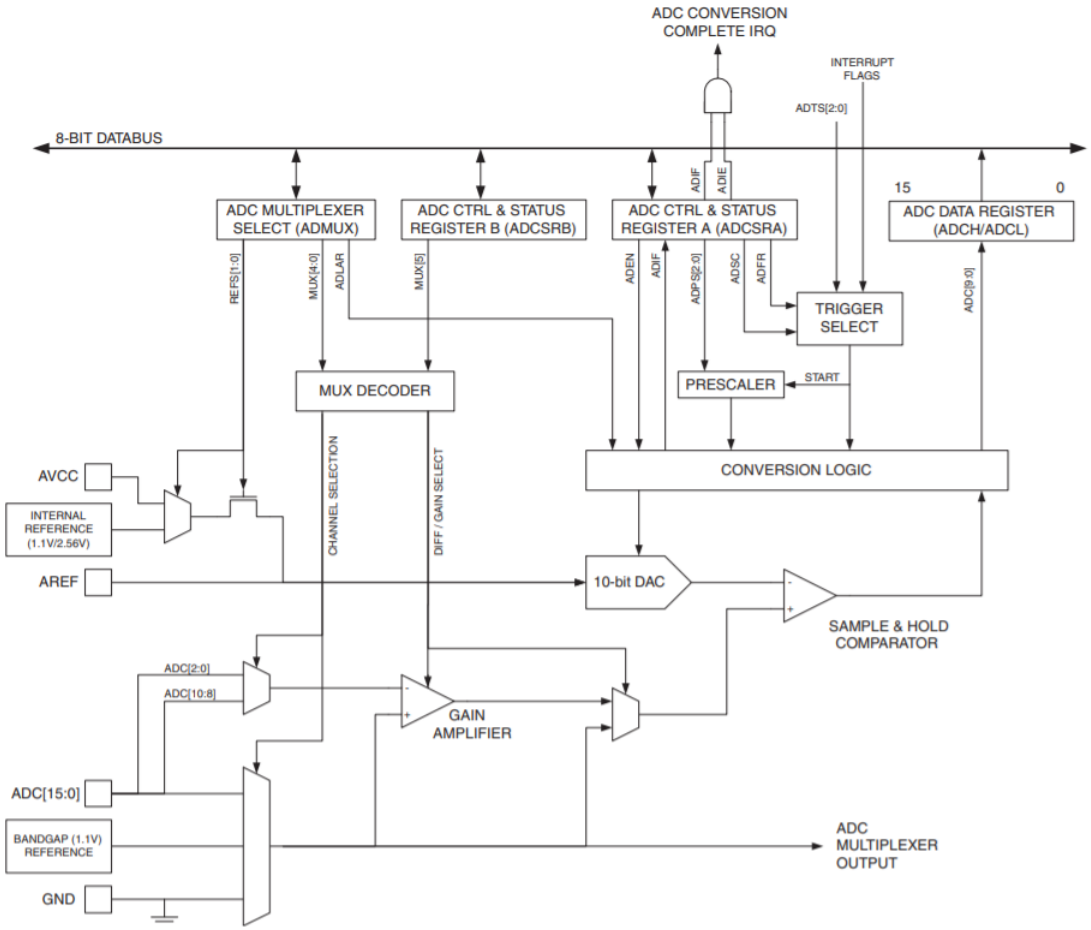


Figura 3.9. Convertitore ADC da 10 bit presente su ATMega2560

Capitolo 4

Teoria controllore PID

In scienza dell'automazione, il controllo automatico di un dato sistema dinamico (ad esempio un motore o un impianto industriale) si prefigge di modificare il comportamento del sistema da controllare attraverso la manipolazione di opportune grandezze d'ingresso. Nell'ambito industriale e non solo, uno dei maggiori controllori utilizzati è quello PID (Proporzionale-Integrativo-Derivativo) che risulta essere molto efficace e semplice da implementare. Un controllore PID all'atto pratico quindi calcola un errore $e(t)$ dato dalla differenza tra un valore di riferimento denominato *Setpoint* (SP) e il valore proveniente da una retroazione chiamato *Process Variable* (PV), applicando successivamente una correzione basata sui termini di proporzione, integrazione e derivazione.

Per comprenderne meglio il comportamento, consideriamo la figura 4.1 e definiamo:

- Setpoint = $r(t)$
- Process variable = $y(t)$
- Errore = $r(t) - y(t)$
- Variabile di controllo = $u(t)$

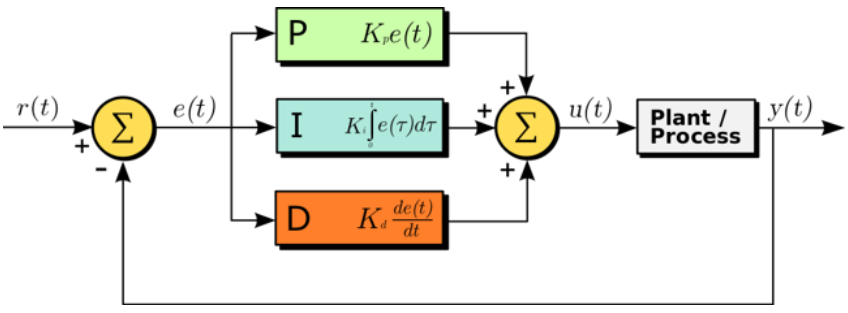


Figura 4.1. Diagramma a blocchi di un controllore PID

Dal punto di vista matematico possiamo quindi valutare la variabile di controllo

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{4.1}$$

dove K_p, K_i e K_d, tutti non negativi, sono i coefficienti per la parte proporzionale, integrativa e derivativa. Tipicamente i termini K_i e K_d sono rispettivamente definiti come K_p/T_i e K_pT_d e sono il tempo di integrazione e il tempo di derivazione. La prima determina in quanto tempo viene eliminato l'errore stazionario, mentre la seconda determina in quanto tempo la caratteristica di uscita si avvicinerà al setpoint.

4.1 Simulazione controllore PID

Di seguito verranno mostrati i risultati di alcune simulazioni per evidenziare come ogni guadagno agisca sul sistema. É importante specificare che i valori dei guadagni dipendono fortemente dal "*plant*" che si analizza. In questo caso è stata analizzata la funzione di trasferimento presente nell'equazione 1.12 e lo schema in figura 4.2 è stato riportato su "$SIMULINK$" in cui come segnale di riferimento è stata utilizzata una funzione "gradino". Per quanto riguarda le specifiche del motore si è fatto riferimento al modello P132NM prodotto dalla SICME MOTORI che riporta le seguenti caratteristiche:

- $R_a = 0.386\ \Omega$
- $L_a = 3.1\text{mH}$
- $\text{J} = 0.085\ \text{Kg}m^2$
- $\text{k} = 1$

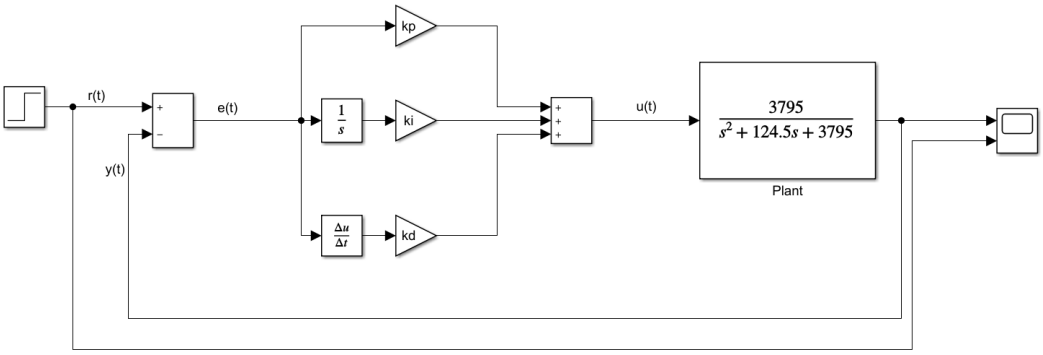


Figura 4.2. Schema utilizzato per simulare un controllore PID

Azione proporzionale P Ogni termine del PID agisce sulla correzione in modo differente. Il termine proporzionale produce un output pari a

$$P_{out} = K_p e(t) \tag{4.2}$$

Perciò si intuisce che avere un K_p elevato produce una correzione elevata. Bisogna comunque fare attenzione al valore K_p che si sceglie poichè avere questo termine troppo alto può portare ad una instabilità del sistema. Al contrario però un guadagno troppo basso può portare ad avere un sistema lento e con un alto errore stazionario. Nei sistemi industriali, l'azione proporzionale è quella che più contribuisce alla correzione dell'errore.

Errore stazionario L'errore stazionario è la differenza che si ha tra il valore desiderato e l'uscita finale. Tipicamente avere un controllore con solo la parte proporzionale può portare ad avere questo tipo di errore. Per eliminare il problema bisogna aggiungere un termine di bias incorporando nel controllore il termine integrale.

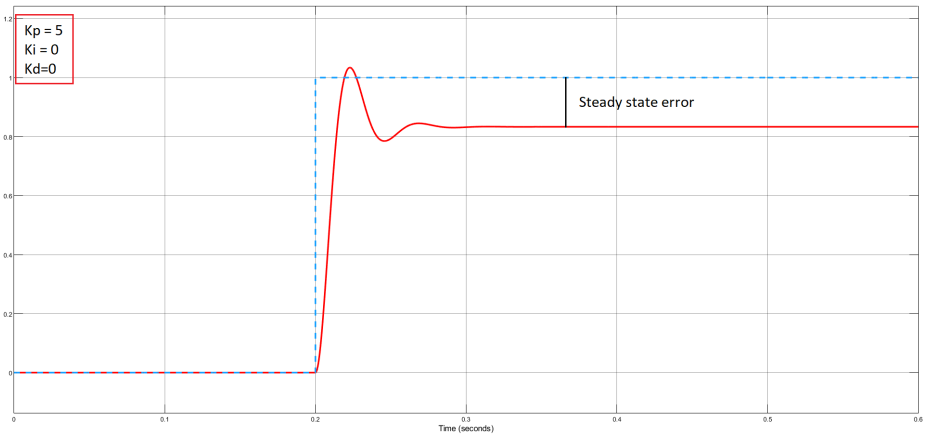


Figura 4.3. Rappresentazione dell'errore stazionario utilizzando solo la parte Proporzionale

Azione integrativa I Per eliminare quindi l'errore stazionario entra in gioco il termine integrale. In questa fase, si tiene conto degli errori precedenti che vengono accumulati nel tempo correggendo l'errore che si viene a creare avendo solo il termine proporzionale. Il termine integrale vale:

$$I_{out} = K_i \int_0^t e(\tau) d\tau \tag{4.3}$$

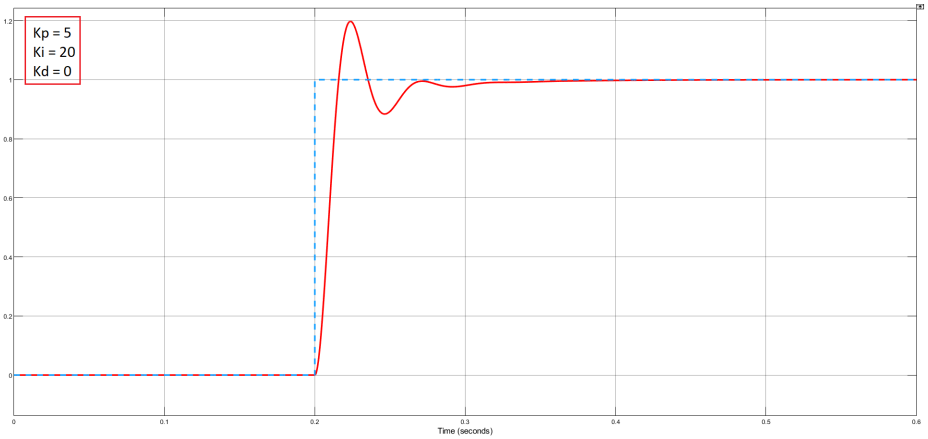


Figura 4.4. Correzione dell'errore stazionario dovuto al termine integrale

Anti wind-up Considerando che il termine integrale accumula tutti gli errori passati, si può incorrere in quel che si chiama "Integral Wind-Up" in cui l'uscita della parte integrale diventa molto grande e può portare il sistema alla saturazione. Per evitare questo problema si fa uso di un limitatore in cui si impone un valore massimo e minimo in cui avere l'uscita dell'integratore. Un'ulteriore tecnica è quella della "Back-calculation" in cui è possibile osservare il funzionamento in figura 4.5, in cui $K_a = 1/K_p$.[2]

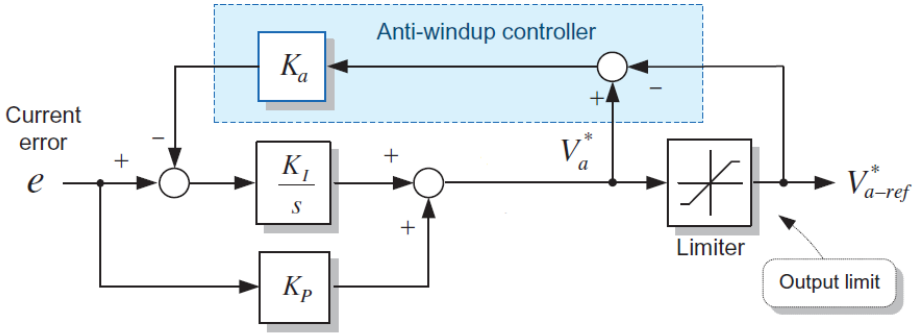


Figura 4.5. Diagramma a blocchi del "Back-calculation"

Azione derivativa D L'ultima azione che viene compiuta sull'uscita del controllore è quella fatta dalla parte derivativa. L'azione derivata prevede il comportamento del sistema e quindi ne migliora il tempo di assestamento e la stabilità. Il termine derivativo vale:

$$D_{out} = K_d \frac{de(t)}{dt} \tag{4.4}$$

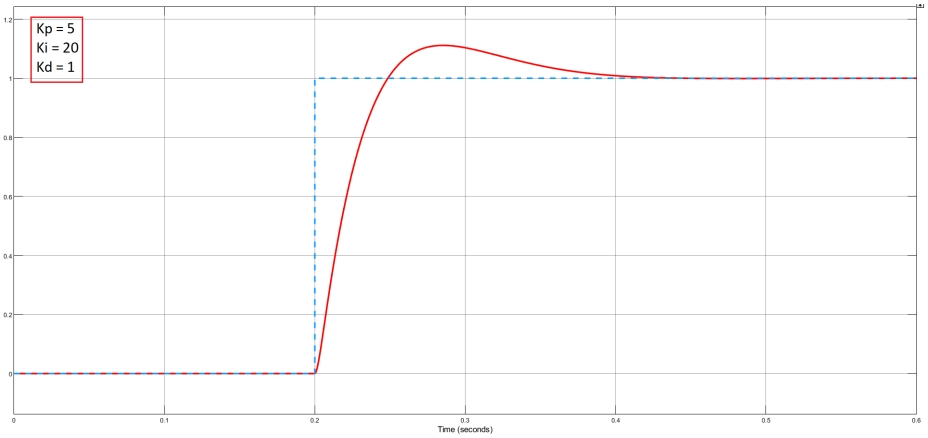


Figura 4.6. Curva di uscita dopo l'azione del termine derivativo

4.2 Tuning di Kp, Ki e Kd

La scelta dei parametri del PID è molto importante per avere un sistema stabile e veloce. In letteratura ci sono diversi metodi di tuning, da quelli manuali a quelli fatti via software. In questa tesi verranno utilizzati e spiegati il metodo manuale e il metodo di Ziegler Nichols.

4.2.1 Tuning manuale

Questa tipologia di metodo è molto semplice da utilizzare ma richiede molta esperienza nel campo da parte dell'utente poiché potrebbe risultare dispendioso in termini di tempo. Per trovare i giusti parametri bisogna innanzitutto impostare K_i e K_d nulli. In seguito si aumenta K_p gradualmente fino all'oscillazione del valore di uscita. A questo punto si imposta il K_p pari a metà del valore tale per cui si ha oscillazione. Successivamente si incrementa K_i in modo tale da eliminare l'errore stazionario, facendo però attenzione a non incrementarlo molto poiché si potrebbe avere instabilità. Infine, si aumenta K_d per smorzare il più possibile le oscillazioni e raggiungere il valore desiderato in un tempo accettabile. Anche in questo caso aumentare troppo il valore di K_d potrebbe portare ad una instabilità del sistema. Nella tabella 4.1 è possibile notare il comportamento di alcuni parametri incrementando in modo indipendente i guadagni di un controllore PID.

Parametro	Tempo di salita	Overshoot	Tempo di assestamento	Errore stazionario	Stabilità
K_p	Diminuisce	Aumenta	Piccola variazione	Diminuisce	Peggiora
K_i	Diminusce	Aumenta	Aumenta	Eliminato	Peggiora
K_d	Piccola variazione	Diminuisce	Diminuisce	Teoricamente nessun effetto	Migliora con K_d piccolo

Tabella 4.1. Andamento di alcuni parametri incrementando k_p, k_i, k_d in modo indipendente

4.2.2 Metodo Ziegler Nichols

Il metodo di tuning Ziegler Nichols è un metodo euristico che prende il nome dai due ingegneri che lo svilupparono, John G. Ziegler e Nathaniel B. Nichols. La ricerca dei parametri del controllore prevede di impostare inizialmente K_i e K_d nulli e di incrementare K_p fino ad ottenere oscillazioni sostenute. Una volte raggiunte queste oscillazioni si valutano il "*Ultime gain*" definito come K_u e il periodo delle oscillazioni sostenute, definito come T_u. Successivamente, in base a questi valori e grazie alla tabella 4.2 è possibile dimensionare un controllore che può essere P, PI, PD o PID.

I parametri definiti nella tabella 4.2 a volte possono richiedere un'ulteriore leggera modifica manuale ma spesso invece possono andar bene sin da subito per il sistema in analisi.[7]

Tipo di controllo	K_p	T_i	T_d	K_i	K_d
P	$0.5K_u$	-	-	-	-
PI	$0.45K_u$	$0.80K_u$	-	$0.54K_u/T_u$	-
PD	$0.8K_u$	-	$0.125T_u$	-	$0.10K_u/T_u$
Classical PID	$0.6K_u$	$0.5T_u$	$0.125T_u$	$1.2K_u/T_u$	$0.075K_u/T_u$

Tabella 4.2. Metodo Ziegler Nichols

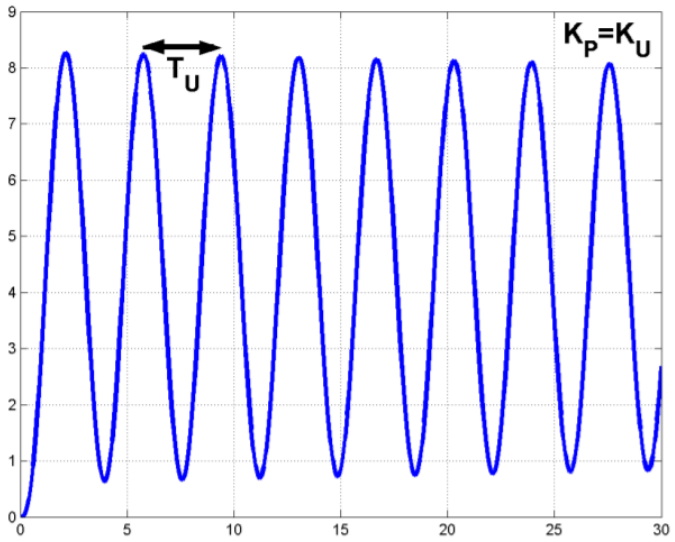


Figura 4.7. Oscillazioni sostenute del valore di uscita

4.3 PID in forma digitale

Nel seguente paragrafo, verrà accennato come è possibile implementare in generale un controllore PID in forma digitale. Vedremo invece nel capitolo 5 come questo verrà implementato su un Arduino.

Innanzitutto è necessario definire e richiamare alcuni parametri ed in particolare:

- $Error$: $Set\ point$ - $Process\ Variable$
- $SampleTime$: Tempo trascorso tra $Error(t)$ ed $Error(t-1)$
- K_p : guadagno proporzionale
- K_i : guadagno integrale
- K_d : guadagno derivata

Di seguito viene riportato uno pseudo-codice utile per l'implementazione del controllore PID in forma digitale.

Codice 4.1. Pseudocodice controllore PID

```
// setup
#define SampleTime;
#define Kp;
#define Ki;
#define Kd;
// loop
TimeChange = (Present - LastTime);
if (TimeChange >= SampleTime)
{
   Error = Setpoint - Input;
   Ierror = Ierror + Error;
   Derror = (Error - LastError);
   Pout = Kp * Error;
   Iout = Ki * Ierror * SampleTime ;

   if(Iout > Upper_limit_Iout)      /*ANTI WINDUP*/
      {
         Iout = Upper_limit_Iout;
      }
   if(Iout < Lower_limit_Iout)
      {
         Iout = Lower_limit_Iout;
      }

   Dout = Kd * Derror / SampleTime ;

   Output = Pout + Iout + Dout;

   if(Output > Upper_limit_Output)
      {
         Output = Upper_limit_Output;
      }
   if(Output < Lower_limit_Output)
      {
         Output = Lower_limit_Output;
      }

   LastTime = Present;
   LastError = Error;
}
```

La variabile "$SampleTime$" è molto importante poiché determina ogni quanto tempo il controllore andrà a correggere l'errore. Questa variabile può dipendere fortemente dal sistema che si analizza, infatti ad esempio per sistemi di controllo temperatura può essere di diversi secondi, mentre per sistemi di controllo motori deve essere dell'ordine dei millisecondi. [8] Si noti anche come è stata implementata l'azione di Anti Windup in cui è bastato semplicemente impostare un range in cui l'uscita dell'integrale può variare.[9]

Capitolo 5

Implementazione

Nel seguente capitolo verranno spiegati tutti i passaggi che sono stati fatti per la realizzazione di questo progetto. Il sistema di azionamento permette il controllo della velocità unidirezionale del motore CC da 38.72kW poiché da specifica può essere alimentato ad un massimo di 440V con una possibile corrente massima assorbita di 88A.

5.1 Sincronismo con tensione di rete

Prima di iniziare con la spiegazione del sincronismo va detto che d'ora in avanti tutti gli impulsi considerati fanno riferimento alle tensioni di fase V_{an}, V_{bn} e V_{cn}, inoltre è importante ricordare la relazione che persiste tra tensioni di fase e tensioni concatenate in cui si ha uno sfasamento di 30°.

Tensioni di fase

$$V_a = V_m sin(\omega t);$$
$$V_b = V_m sin(\omega t - 120°);$$
$$V_c = V_m sin(\omega t - 240°);$$

Tensioni concatenate

$$V_a - V_b = \sqrt{3} V_m sin(\omega t + 30°);$$
$$V_b - V_c = \sqrt{3} V_m sin(\omega t - 90°);$$
$$V_c - V_a = \sqrt{3} V_m sin(\omega t - 210°);$$

Come spiegato nel sottoparagrafo 2.2.2 il ponte raddrizzatore trifase ha bisogno degli impulsi sui gate degli SCR per funzionare. Per permettere ciò è stato quindi necessario generare in uscita ai pin digitali di Arduino, grazie alla generazione di interrupt esterni ed interni, sei impulsi tra di loro indipendenti ed ognuno di loro in fase con la semionda di riferimento.

Generazione delle onde quadre per gli interrupt esterni Per un buon sincronismo è importante sapere quando l'onda sinusoidale di riferimento è positiva o negativa. Per fare ciò bisogna creare un'onda quadra a partire da una sinusoidale. Il circuito utilizzato è mostrato un figura 5.1

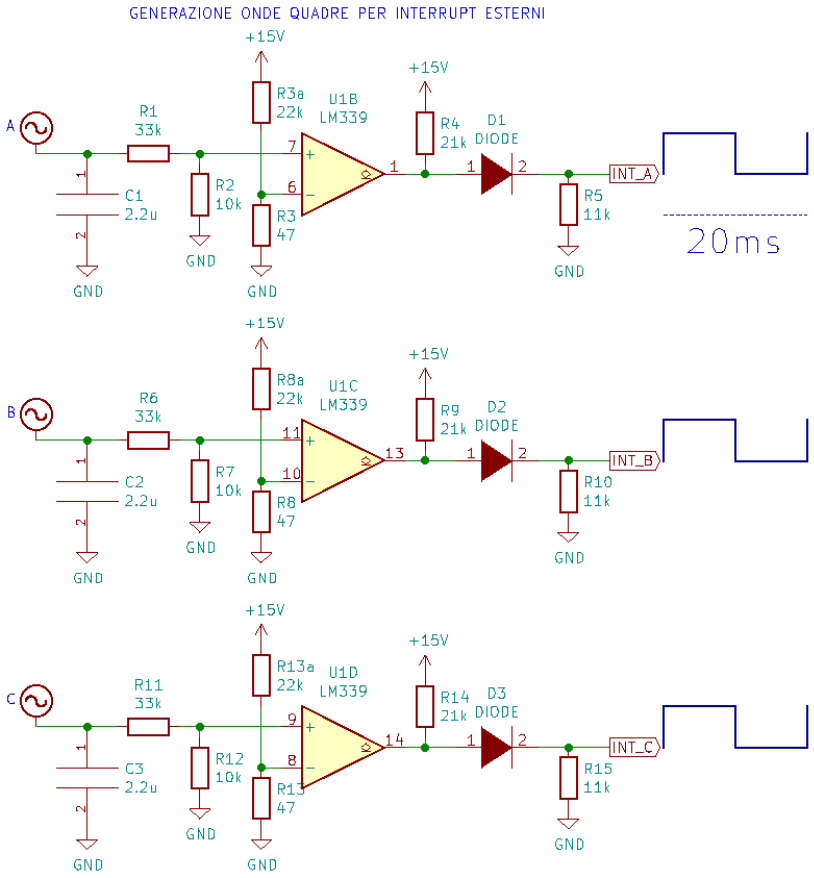


Figura 5.1. Creazione delle onde quadre per generare interrupt esterni

Per semplicità si prenda in considerazione la fase A. Tutte le tensioni di ingresso sono state ottenute attraverso un trasformatore trifase in modo tale da ridurre notevolmente l'ampiezza della sinusoide a $67V_{pp}$. Attraverso la rete formata da $R1$ ed $R2$ si ottiene un'ulteriore riduzione del segnale che viene inserito nel morsetto non-invertente del comparatore LM339. Nell'altro morsetto invece, è presente un partitore che permette di creare un piccolo offset di circa 30mV in modo tale da evitare l'indecisione del comparatore quando l'onda in ingresso è nulla. In uscita dal comparatore è stata dimensionata una rete formata da $R4$, $D1$ e $R5$. In particolare $R4$ è una resistenza di pull-up e $D1$ evita il passaggio della parte negativa dell'onda quadra riuscendo cosi ad avere ai capi di $R5$ una tensione compresa tra 0V e 5V con Duty-Cycle del 50% e periodo di 20ms.

Timer e gestione degli interrupt per generazione singoli impulsi Le onde quadre ottenute saranno degli ingressi per la scheda di valutazione che verrà programmata per generare degli interrupt ogni volta che ci sarà un fronte di salita e discesa. In particolare sono stati sfruttati tre timer interni della scheda Arduino, denominati TIMER1, TIMER3, e TIMER4 in cui ognuno di essi lavora rispettivamente per la fase A, B e C. Di seguito viene riportato il codice utilizzato per generare, attraverso gli interrupt esterni ed interni, i sei impulsi differenti. Per semplicità verrà riportato la parte di una sola fase essendo le altre due del tutto analoghe.

Codice 5.1. Generazione di due impulsi

```
// setup

// Pin digitale definito come ingresso per l'onda quadra
pinMode(X, INPUT);

// Pin digitali definiti come output per generare gli impulsi
pinMode(Y, OUTPUT);
pinMode(Z, OUTPUT);

// Inizializzazione del TIMER1
TCCR1A = 0; // TIMER UTILIZZATO PER LA FASE A (PER SINCRONISMO)
TCCR1B = (0 << CS12) | (1 << CS11) | (0 << CS10); // Prescaler 8
TIMSK1 = (1 << OCIE1C) | (1 << OCIE1B) | (1 << OCIE1A); // Abilitazione
    interrupt
TIFR1 = (0 << OCF1C) | (0 << OCF1B) | (0 << OCF1A); //Flag azzerato
    raggiungimento valore scelto
TCNT1 = 0; // Azzero contatore a 16 bit

//Definisco il pin X per generare interrupt al fronte di salita dell'onda
    quadra.
//Una volta trovato il fronte di salita inizia la funzione "sincronismo_A"
attachInterrupt(digitalPinToInterrupt(X), sincronismo_A, RISING);

//Funzione "sincronismo_A" richiamata
void sincronismo_A()
{
TCNT1 = 0; //Azzeramento del contatore quando trovo un fronte
OCR1A = 18333 - Beta; //primo valore di uguaglianza, quando TCNT1 = OCR1A si
    genera un timer interrupt.
}
// Routine del timer interrupt
ISR(TIMER1_COMPA_vect)
{
iia++;
switch (iia)
{
case 1:
PORTB = (1 << PB7); // Pin 1 di uscita asserito a +5V
```

```
OCR1A += 1000;    //Aggiorno OCR1A
attachInterrupt(digitalPinToInterrupt(X), sincronismo_ST, FALLING); // Imposto
    il pin di ingresso per trovare fronte di discesa
break;

case 2:
PORTB = (0 << PB7); //Pin 1 di uscita asserito a 0V
break;

case 3:
PORTH = (1 << PH5); //Pin 2 di uscita asserito a +5V
OCR1A += 1000; //Aggiorno OCR1A
attachInterrupt(digitalPinToInterrupt(X), sincronismo_ST, RISING); //Imposto
    nuovamente il pin di ingresso per trovare fronte di salita
break;

case 4:
PORTH = (0 << PH5); //Pin 2 asserito a 0V
iib = 0;
break;
}
}
```

La variabile "*Beta*" è quella che muove gli impulsi che, in un sistema ad anello aperto è comandata direttamente da un potenziometro connesso alla scheda, mentre in configurazione ad anello chiuso è controllata dai regolatori PID di velocità e corrente.

Supponendo Beta nullo, si noti che quando la funzione "sincronismo_A" è richiamata si imposta $OCR1A = 18333$ in cui, ricordando quanto detto nel paragrafo 3.3 varrà 9.15ms che equivalgono a 165°. Perciò prendendo in esame un singolo impulso, questo verrà generato rispetto alla tensione concatenata ed in particolare rispetto ad un semiperiodo dopo 165° + 30° = 195°. Questo è molto importante poiché nonostante la conduzione si annulli a 180° all'atto pratico è buona norma posizionare gli impulsi dai 10° ai 30° dopo la fine del semiperiodo evitando così eventuali conduzioni che potrebbero venire a crearsi se gli impulsi fossero posizionati a cavallo del cambio del semiperiodo. Altra variabile molto importante è "*iia*" che viene utilizzata per scandire esattamente i fronti di salita e discesa degli impulsi. Dopo che è stato generato il fronte di salita dell'impulso è necessario definire la larghezza di quest'ultimo. Si noti che nella ISR (Interrupt Service Routine) del Timer è presente la riga di codice "$OCR1A+ = 1000$" in cui praticamente si decide di generare il fronte di discesa dell'impulso esattamente 500µs dopo il fronte di salita. Nelle immagini 5.2 e 5.3 vengono evidenziati rispettivamente, la posizione degli impulsi rispetto alla tensione di fase e concatenata e come viene incrementata "*iia*".

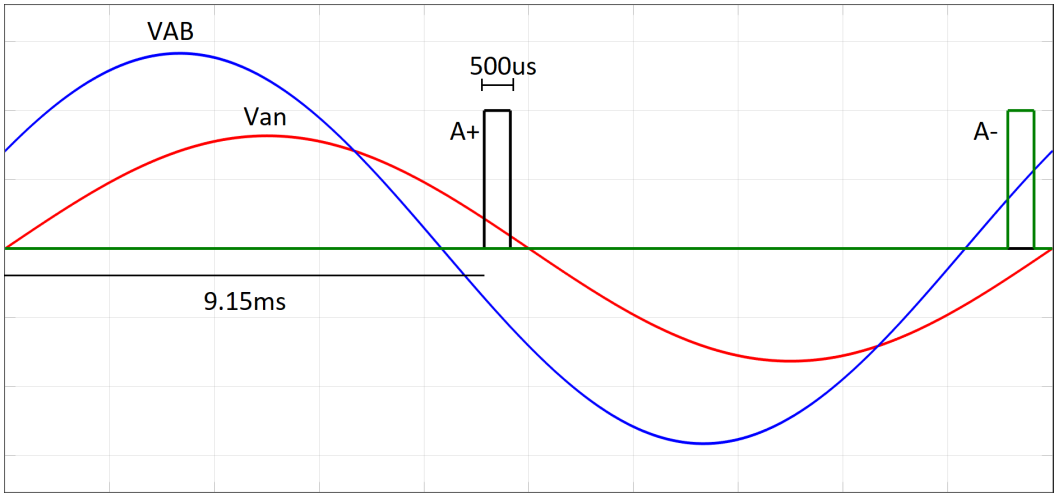


Figura 5.2. Generazione degli impulsi con $Beta = 0$ (conduzione nulla)

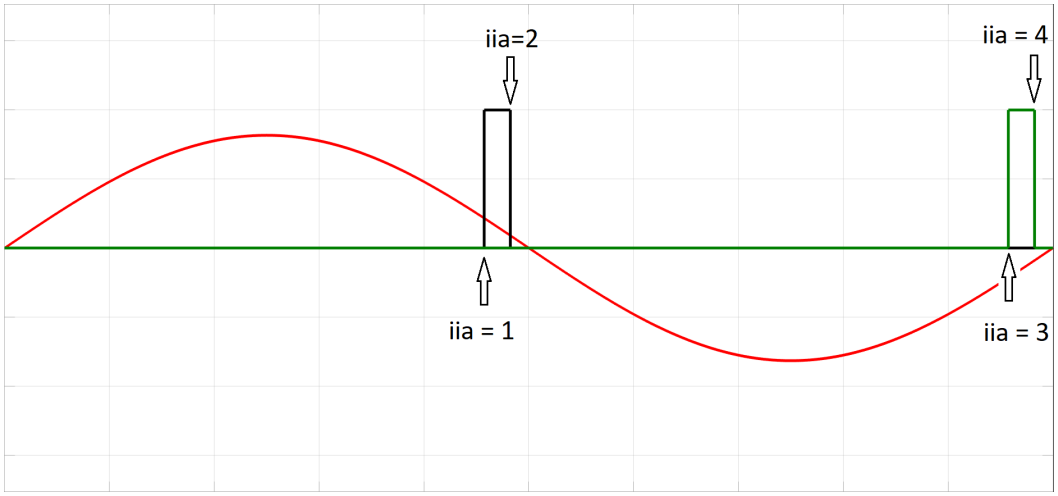


Figura 5.3. Gestione della variabile "*iia*" durante la generazione degli impulsi

Generazione doppi impulsi Nel sottoparagrafo 2.2.2 è stato spiegato che per un corretto funzionamento del ponte trifase è necessario avere coppie di SCR che conducano che ricordiamo essere T1/T2, T2/T3, T3/T4, T4/T5, T5/T6 e T6/T1.

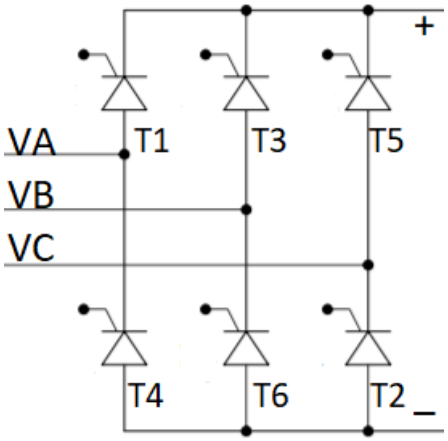


Figura 5.4. Ponte trifase SCR totalmente controllato

All'atto pratico, bisogna essere sicuri che l'SCR continui la sua conduzione dopo essere stato innescato. Perciò è necessario utilizzare per ogni tiristore un doppio impulso in cui il secondo appartiene anche all'altro tiristore facente parte della coppia. Per comprendere meglio, prendiamo in esame la tensione V_{AB} nel semiperiodo negativo, questa per essere raddrizzata ha bisogno della conduzione del tiristore T3 e T4(vedi cap. 2, fig. 2.8) perciò il secondo impulso di T3 sarà il primo di T4. Altro particolare molto importante da dire è lo sfasamento che si viene a creare tra i due impulsi. La frequenza delle onde di ingresso è di 50Hz, ma avendo un sistema trifase avremo in uscita dal ponte una frequenza di 300Hz, questo vuol dire che la distanza tra un impulso e l'altro equivale a 1/300Hz ovvero 3,33ms che convertiti in gradi sono pari a 60°.

Per ottenere quindi dei doppi impulsi è stato utilizzato l'integrato CD4071 che è composto da porte OR in logica CMOS. Lo schema utilizzato e la forma d'onda ottenuta all'uscita dell'integrato CD4071 sono mostrati nelle immagini 5.5 e 5.6

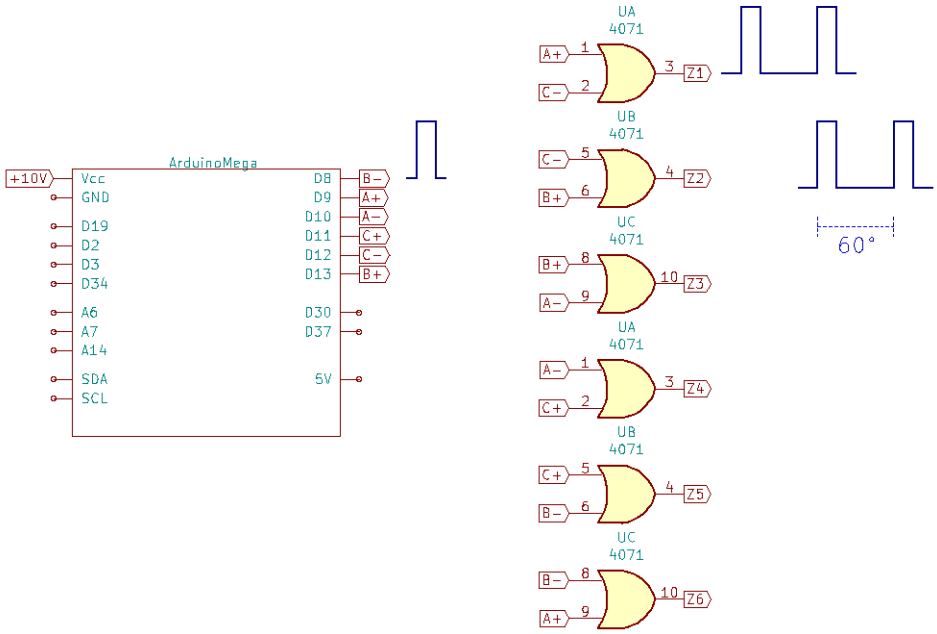


Figura 5.5. Schema per la generazione di doppi impulsi

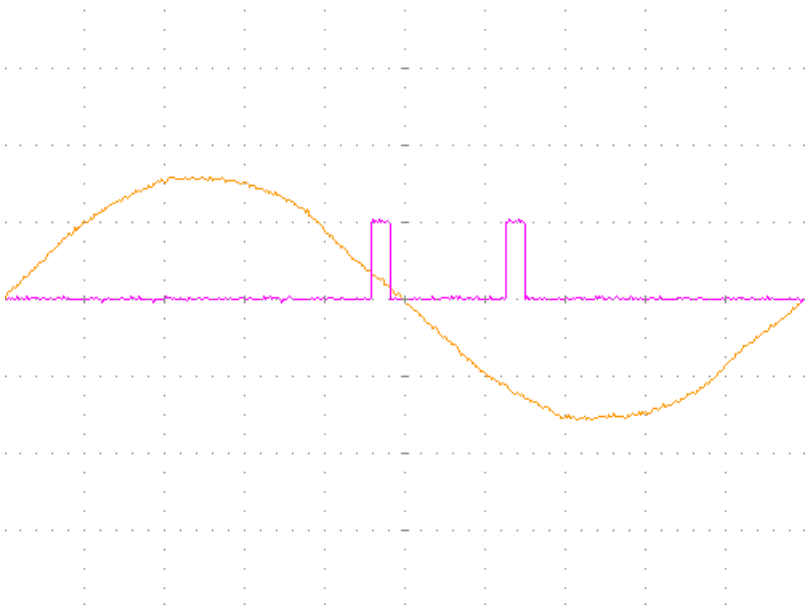


Figura 5.6. Forme d'onda ottenute in uscita dal CD4071

Treno di impulsi e amplificatore di corrente All'atto pratico bisogna assicurarsi che il tiristore entri in conduzione e purtroppo gli impulsi ottenuti precedentemente non sono ancora necessari, inoltre gli SCR vengono attivati se la corrente di innesco è sufficientemente alta. Per l'attivazione dei tiristori quindi è stato utilizzato l'oscillatore NE555, gli integrati M74HC08B1 e ULN2004 e un trasformatore di impulsi.

Il primo passo è stato quello di configurare l'oscillatore NE555 in modalità astabile così da poter generare un treno di impulsi ad una frequenza di circa 10kHz utilizzando le seguenti formule:

$$f = \frac{1.44}{(R_{16} + 2R_{17})C_5}$$

$$t_H = 0.693(R_{16} + R_{17})C_5$$

$$t_L = 0.693(R_{17})C_5$$

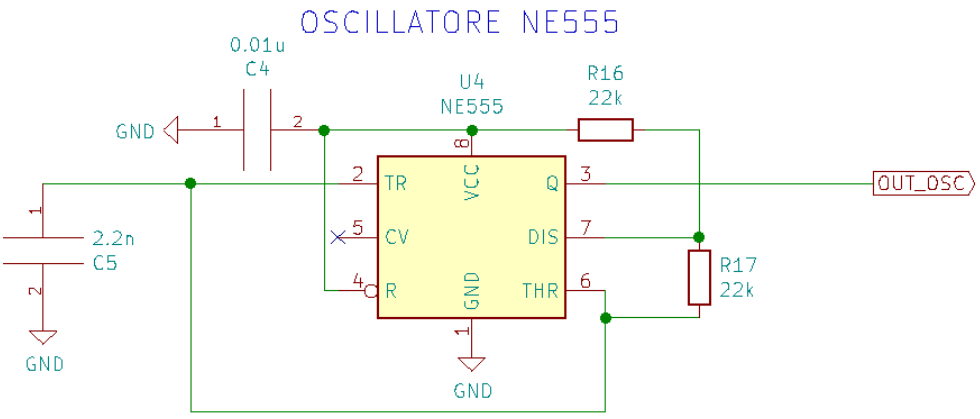


Figura 5.7. Configurazione dell'oscillatore NE555 in modalità astabile

Successivamente è stato utilizzato l'integrato M74HC08B1 che è composto da porte AND. Gli ingressi di questo integrato sono rispettivamente le uscite del CD4071 e quella dell'oscillatore NE555. In questo modo è possibile ottenere un treno di impulsi alla frequenza di 10kHz all'interno delle coppie di impulsi generati in precedenza. Nelle immagini 5.8 e 5.9 è possibile osservare il circuito che è stato utilizzato e le forme d'onda ottenute.

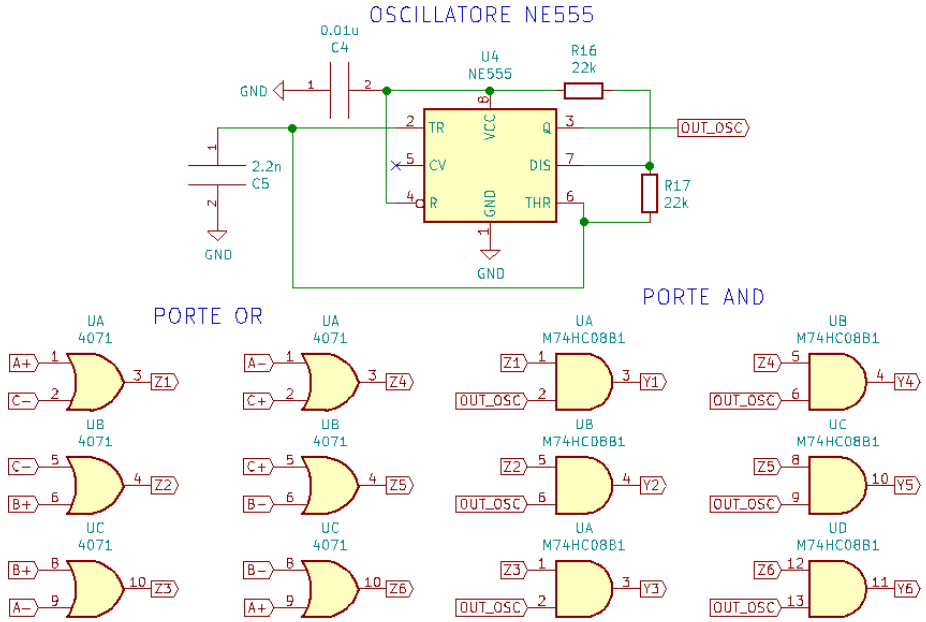


Figura 5.8. Circuito finale per generazione treno all'interno degli impulsi

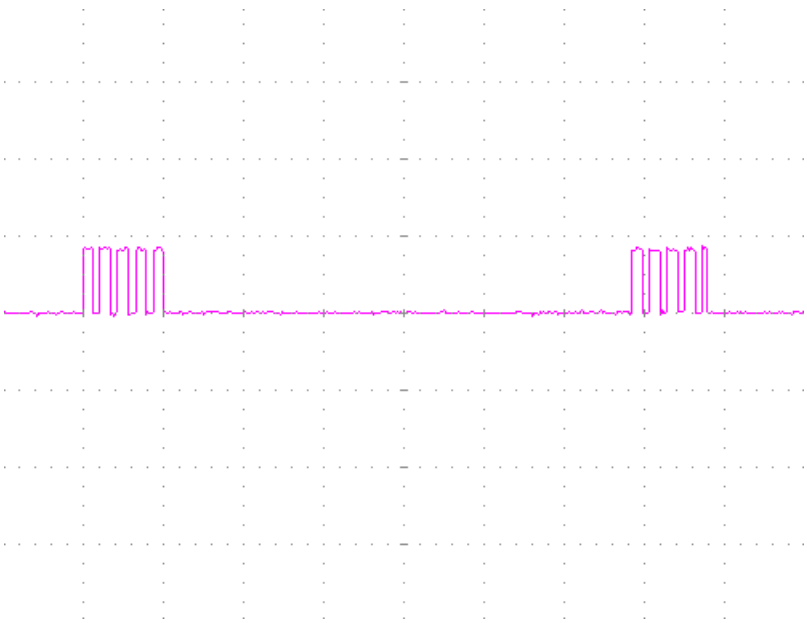


Figura 5.9. Forma d'onda in uscita dall'integrato M74HC08B1

Il ponte raddrizzatore trifase utilizzato in sede di progetto è formato da SCR modello VSKT250-PBF che, secondo quanto riportato sul datasheet, ha bisogno al massimo di 200mA a temperatura ambiente per poter funzionare nel migliore dei modi.[10] Per avere la corrente necessaria all'attivazione quindi è stato utilizzato l'ULN2004 che al suo interno è composto da un array di Darlington capace di erogare un massimo di 500mA ciascuno. In figura 5.10 è riportato il circuito utilizzato.

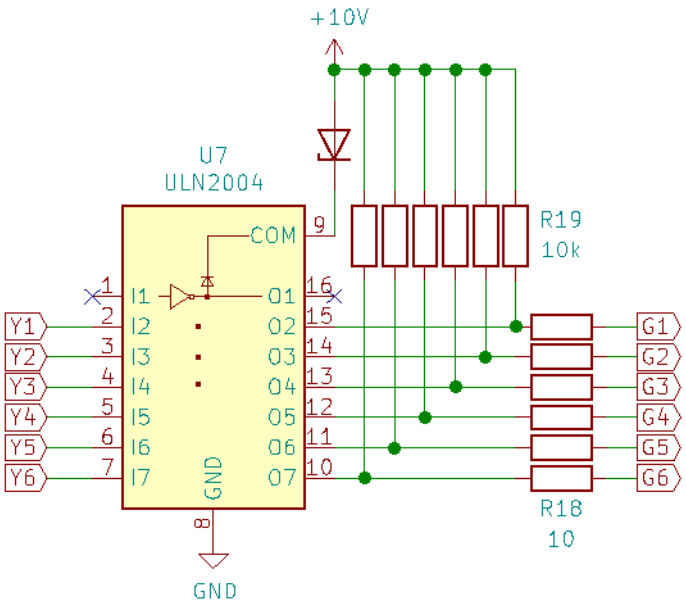


Figura 5.10. Circuito integrato ULN2004 per l'amplificazione della corrente

Gli ingressi di questo integrato, non sono altro che le uscite del M74HC08B1. Per quanto riguarda gli output invece, è stata utilizzata una resistenza di pull-up ed una resistenza serie che entra direttamente nel trasformatore di impulsi. Particolare attenzione va posta al piedino n°9 dell'integrato, questo viene connesso a 10V attraverso un diodo Zener che viene utilizzato come stabilizzatore di tensione per alimentare l'integrato.

Trasformatori di impulsi Alla fine della catena, per un corretto azionamento degli SCR, ci sono i trasformatori di impulsi in cui è possibile osservare in figura 5.11 lo schema circuitale. Questi, in uscita, posseggono principalmente due pin denominati con "T" e "K" che in seguito verranno applicati rispettivamente al gate e al catodo del tiristore.

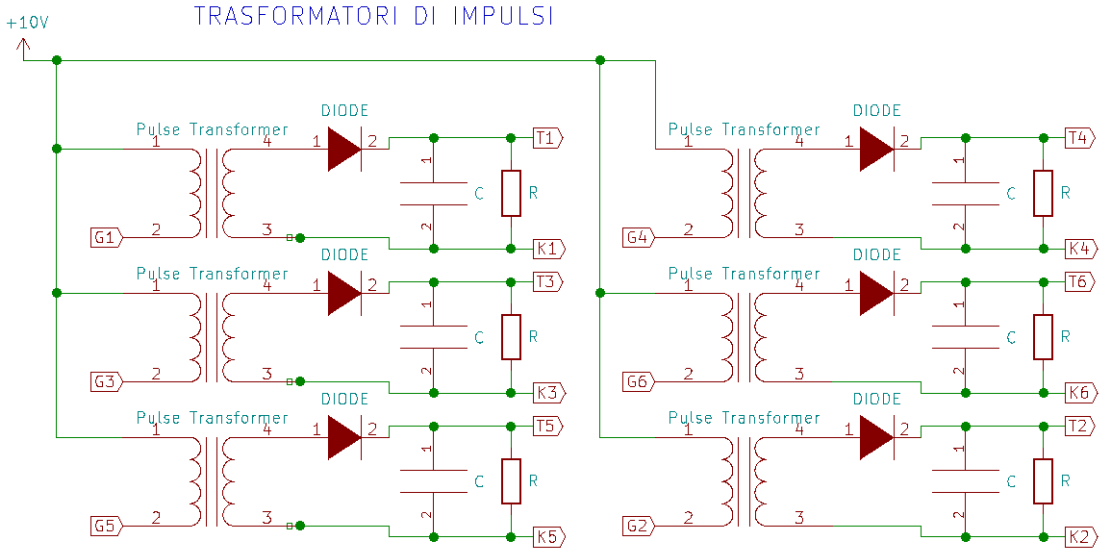


Figura 5.11. Schema circuitale dei trasformatori di impulsi

Tensione di uscita del ponte variabile: spostamento degli impulsi Nel codice 5.1 è presente la variabile "*Beta*". Questa permette di muovere gli impulsi per poter avere in uscita dal ponte raddrizzatore una tensione continua variabile. Il movimento degli impulsi però può avvenire attraverso un anello chiuso grazie ai controllori PID o ad anello aperto grazie ad un controllo diretto da parte di un potenziometro. Inoltre nel codice 5.1 con "*Beta*"=0 si ha conduzione nulla e l'impulso si trova a 165° dopo rispetto l'inizio della tensione di fase di riferimento. Per poter raggiungere la massima conduzione è necessario che l'impulso venga posizionato 30° dopo, ciò vuol dire che rispetto alla tensione concatenata si ritroverà 60° dopo. Quindi se per conduzione nulla abbiamo 165° che tradotto vuol dire avere "$OCRxA$" = 18333 e per conduzione massima invece abbiamo 30° che equivale ad avere "$OCRxA$" = 3333, vuol dire che il potenziometro viaggerà in un range compreso tra 0 e 15000. Va inoltre detto che gli ingressi analogici di Arduino sono composti da 10 bit, ciò vuol dire che bisogna trasformare il range da 0 a 1023 in uno da 0 a 15000. Di seguito è possibile osservare come vengono ottenuti i giusti limiti della variabile "*Beta*" e il circuito completo per la gestione degli impulsi in un sistema ad anello aperto.

Codice 5.2. Funzione "map" di Arduino

```
Potenziometro = analogRead(y); // y: numero del pin analogico utilizzato
Beta = map(Potenziometro, 0 ,1023, 0, 15000);
```

In più in figura 5.12 è riportato lo schema a blocchi dell'intero sistema fin'ora analizzato utile per il controllo ad anello aperto.

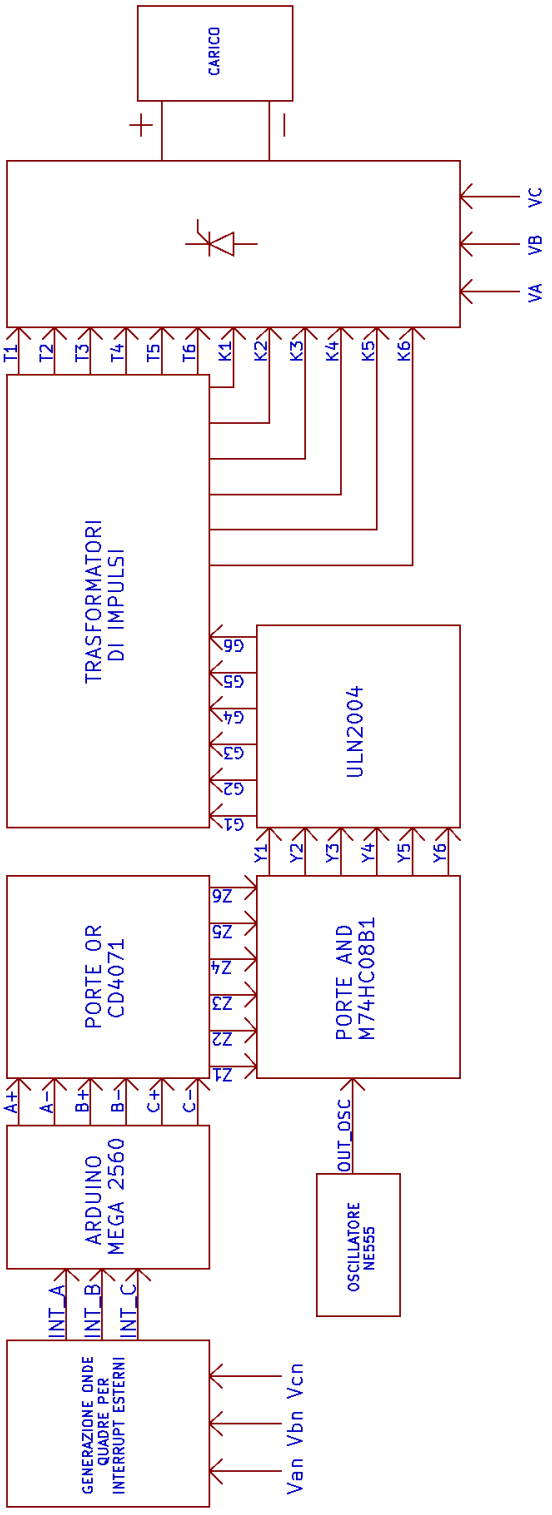


Figura 5.12. Schema a blocchi per la generazione e lo spostamento degli impulsi con sistema ad anello aperto

5.2 Controllo di corrente e velocità

Dopo aver generato in maniera corretta gli impulsi è necessario rendere il sistema stabile attraverso un controllore. Nel progetto svolto sono stati utilizzati due controllori ed in particolare uno per la velocità di tipo PID ed uno per la corrente di tipo PI.

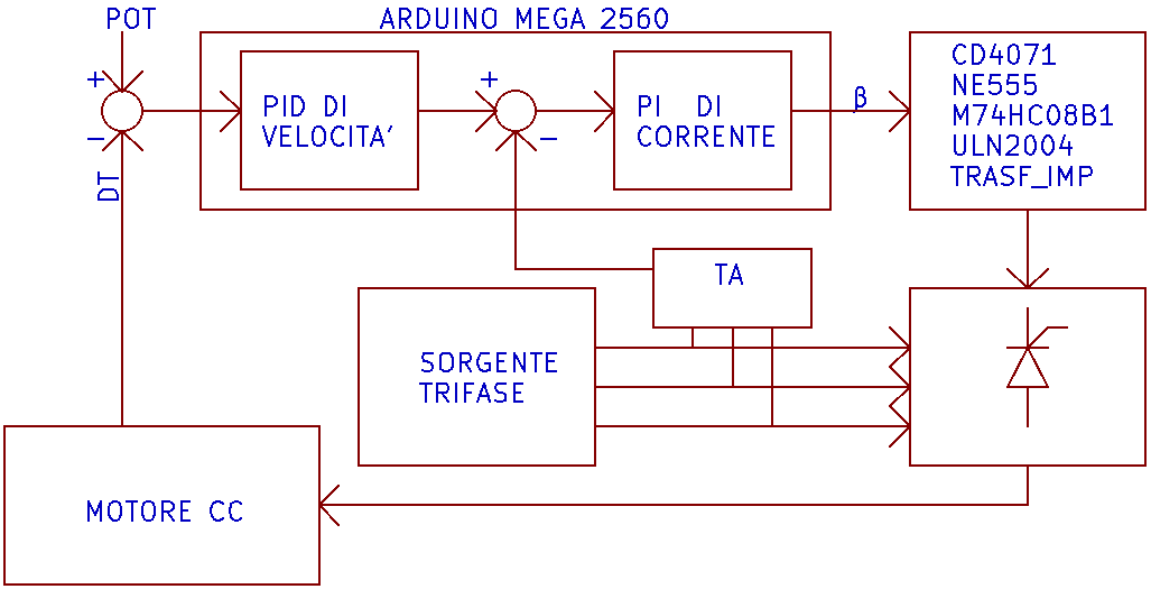


Figura 5.13. Schema a blocchi PID di velocità e PI di corrente

PID di velocità Il PID di velocità, come si può vedere dalla figura 5.13 è quello che forma l'anello più esterno del sistema. Gli ingressi del controllore sono rispettivamente il potenziometro di riferimento e il segnale proveniente dalla Dinamo Tachimetrica (DT) attaccata al motore che, ricordando quanto detto nel capitolo 1, trasforma energia meccanica in energia elettrica producendo corrente continua. Prima di parlare dell'implementazione software che è stata effettuata è bene spiegare come è stato prelevato il segnale della dinamo. Questo componente produce una tensione proporzionale alla velocità del motore. In particolare, considerando che la velocità è misurata in [RPM] (Rotazioni Per Minuto), la dinamo avrà ai suoi capi una tensione pari a:

$$V_{DT} = Velocità\_motore \times 0.06 \frac{V}{RPM} \tag{5.1}$$

Il motore utilizzato in sede di progetto può raggiungere una velocità massima di 2500 RPM, tenendo conto perciò della relazione in 5.1 si ottiene che la tensione massima generata dalla dinamo è pari a 150V. Naturalmente una tensione così elevata non può entrare direttamente nella scheda di valutazione, è stata perciò dimensionata una rete formata da un partitore ed un filtro riuscendo ad avere una tensione massima di 4.5V in ingresso al microcontrollore. Il potenziometro di riferimento viene alimentato a 5V ma anche in

questo caso è stata utilizzata la stessa tipologia di rete per ridurre la tensione a 4.5V in modo tale da poterla comparare con il segnale della DT.

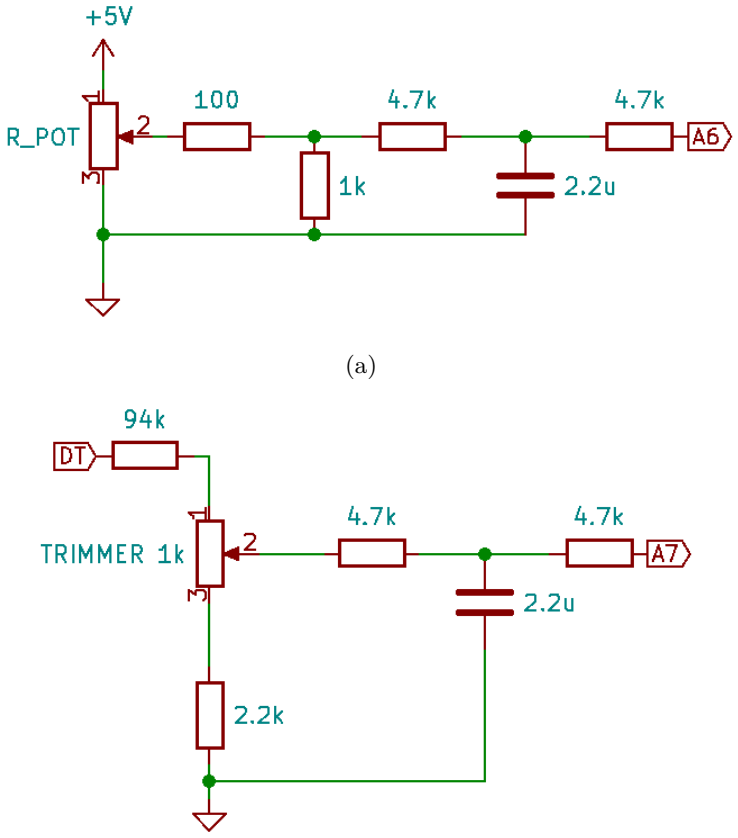


Figura 5.14. Reti dimensionate per Potenziometro (a) e Dinamo Tachimetrica (b)

Particolare attenzione va posta alla rete dimensionata per la dinamo. Per quanto riguarda il lato del partitore è stato utilizzato un trimmer da 1kΩ in modo tale da poter regolare la tensione massima in ingresso ad Arduino. In più, la resistenza da 2.2kΩ evita che la tensione raggiunga gli 0V quando il trimmer è posizionato al minimo.

Dopo aver dimensionato la parte hardware dell'anello di velocità è possibile occuparsi della parte software. Di seguito viene riportato il codice utilizzato.

Codice 5.3. Controllore PID di velocità

```
float Kp_vel = 5;  //Definizione dei parametri del PID
float Ki_vel = 2.5;
float Kd_vel = 1.2;

float error_vel = 0; //errore dato dalla differenza tra DT e Pot. di
    riferimento
float Total_error_vel = 0; //Valore di uscita del PID di vel.
float lasterror_vel = 0; //errore al controllo i-1

float Error_vel_P = 0; //inizializzazione a 0 dei termini del PID
float Error_vel_I = 0;
float Error_vel_D = 0;

unsigned long SampleTimeVel = 30; //millisecondi trascorsi tra il controllo i
    e i-1
unsigned long PastTimeVel = 0; //variabile utilizzata per controllare se sono
    trascorsi 30ms

// Start loop
void loop()
{
   DT = analogRead(A7); //Lettura continua del valore della dinamo
   Potenziometro = analogRead(A6); //Lettura continua del valore del
       potenziometro;
   Curr_ref = MyPIDV(Potenziometro, DT, Kp_vel, Ki_vel, Kd_vel); //Uscita del
       PID, diventa il riferimento del PI di corrente
}

//Funzione utilizzata per il PID
float MyPIDV( int Potenziometro, int DT, float Kp_vel, float Ki_vel, float
    Kd_vel)
{
unsigned long ActualTimeVel = millis(); //Salvo il tempo attuale

int Delta_T_vel = ActualTimeVel - PastTimeVel;
if (Delta_T_vel >= SampleTimeVel) //Controllo il tempo passato (>= 30ms)
{
error_vel = Potenziometro - DT;
Error_vel_P = error_vel * Kp_vel;
Error_vel_I = (Error_vel_I + error_vel ) * (Delta_T_vel / 1e3) * Ki_vel;
if (Error_vel_I >= 255) //Anti Wind-up
{
   Error_vel_I = 255;
}
else if (Error_vel_I <= -255)
```

```
{
   Error_vel_I = -255;
}

Error_vel_D = (error_vel - lasterror_vel) * Kd_vel / (Delta_T_vel / 1e3);


Total_error_vel = Error_vel_P + Error_vel_I + Error_vel_D;

if (Total_error_vel >= 4096)
{
Total_error_vel = 4096;
}
else if (Total_error_vel <= -4096)
{
Total_error_vel = -4096;
}

lasterror_vel = error_vel; //Aggiornamento paramentri per il controllo
    successivo
PastTimeVel = ActualTimeVel;
}  //end condizione if
return Total_error_vel;
}
```

Va considerato che in sistemi riguardati il controllo di motori il tempo di campionamento di un controllore PID è dell'ordine dei ms [8], infatti in questo progetto si è scelto un tempo pari a 30ms. Per il controllo del tempo è stata utilizzata la funzione di Arduino "$millis()$" che tiene conto dei millisecondi trascorsi dalla partenza del programma e con un semplice controllo attraverso una "$if\ condition$" è stato possibile definire quando il controllore PID avrebbe calcolato l'errore. Inoltre si noti anche come è stato gestito "$L'Anti\ Wind-up$" dove per evitare la saturazione della parte integrale è bastato limitarne l'uscita.[9]

PI di corrente L'anello più interno del sistema è formato dal controllore PI di corrente. In questo caso si è utilizzata questa tipologia di controllore poiché nella maggior parte dei sistemi attuali la parte derivativa diventa superflua.

Per quanto concerne la parte hardware, si noti in figura 5.13 un blocco denominato "TA" che identifica la presenza di Trasformatori Amperometrici. Questi dispositivi sono caratterizzati da un foro centrale in cui far passare il cavo di cui si vuole misurare la corrente. All'atto pratico quando la corrente attraversa il cavo in questione, il campo magnetico che si genera intercetta la bobina del lato primario e sul lato secondario quindi si verrà a creare una corrente proporzionale di diversi ordini più piccola. Per il progetto in questione sono stati utilizzati dei TA modello 151008 della Sirio caratterizzati da un rapporto spire 1:1000[11], ciò vuol dire che la corrente in uscita avrà un valore 1000 volte più piccola rispetto a quella in ingresso. La corrente valutata non è quella continua del motore ma bensì la corrente che viene erogata dalla sorgente trifase. Per il dimensionamento del circuito utile si è fatto riferimento alla seguente relazione [5]:

$$I_s = \sqrt{\frac{2}{3}} I_d = 0.816 I_d \tag{5.2}$$

dove I_s è la corrente rms della linea mentre I_d è la corrente continua che arriva al motore. Considerando che quest'ultimo assorbe un massimo di 88A, vuol dire che la I_{smax} è pari a 71.8A. In sede di progetto, sono stati effettuati due passaggi attraverso il foro del TA ottenendo così un rapporto pari a 1:500, inoltre è stato scelto come limite di corrente massima 75A e riutilizzando l'equazione 5.2 si è ottenuta la corrente all'uscita del ponte diodi trifase pari a 183mA utilizzando quindi un carico di 27Ω si è ottenuta una tensione massima di circa 5V in modo tale da poterla interfacciare con la Scheda di valutazione Arduino.

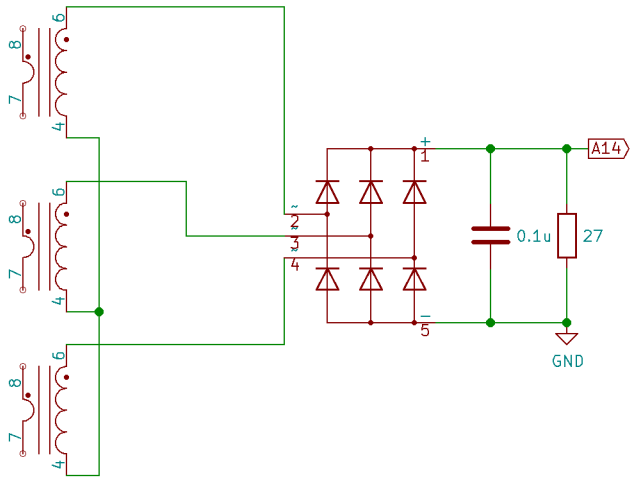


Figura 5.15. Schema elettrico utilizzato per PID di corrente

Figura 5.16. Trasformatore amperometrico utilizzato in sede di progetto

Come per il PID di tensione, anche in questo caso, dopo essersi occupati della parte hardware è bene implementare per via software il controllore PI. Di seguito viene riportato il codice, compreso della parte del PID, utilizzato in sede di progetto.

Codice 5.4. Controllore PID di velocità e PI di corrente

```
float Kp_vel = 5;  //Definizione dei parametri del PID
float Ki_vel = 2.5;
float Kd_vel = 1.2;

float Kp_curr = 0.07;  //Definizione dei parametri del PI
float Ki_curr = 1;

float error_vel = 0; //errore dato dalla differenza tra DT e Pot. di
    riferimento
float Total_error_vel = 0; //Valore di uscita del PID di vel.
float lasterror_vel = 0; //errore al controllo i-1 (PID)
float error_curr = 0; //errore dato dalla differenza tra uscita del PID di
    vel. e A14
float Total_error_curr = 0; //Valore di uscita del PI di corrente
float lasterror_curr = 0; //errore al controllo i-1 (PI)

float Error_vel_P = 0; //inizializzazione a 0 dei termini del PID
float Error_vel_I = 0;
float Error_vel_D = 0;
float Error_curr_P = 0; //inizializzazione a 0 dei termini del PI
float Error_curr_I = 0;
float Error_curr_D = 0;

unsigned long SampleTimeVel = 30; //millisecondi trascorsi tra il controllo i
    e i-1
unsigned long PastTimeVel = 0; //variabile utilizzata per controllare se sono
    trascorsi 30ms
unsigned long SampleTimeCurr = 10; //millisecondi trascorsi tra il controllo i
    e i-1
```

```
unsigned long PastTimeCurr = 0; //variabile utilizzata per controllare se sono
     trascorsi 10ms

int value_curr[30];
float curr_feed = 0;
int y = 0;

// Start loop
void loop()
{
DT = analogRead(A7); //Lettura continua del valore della dinamo
Feedbackcurr =Letture_curr();
Potenziometro = analogRead(A6); //Lettura continua del valore del
    potenziometro;
Curr_ref = MyPIDV(Potenziometro, DT, Kp_vel, Ki_vel, Kd_vel); //Uscita del
    PID, diventa il riferimento del PI di corrente
Beta = MyPIDC(Curr_ref, FeedbackCurr, Kp_curr, Ki_curr);
}

//Funzione utilizzata per il PID di velocità
float MyPIDV( int Potenziometro, int DT, float Kp_vel, float Ki_vel, float
    Kd_vel)
{
unsigned long ActualTimeVel = millis(); //Salvo il tempo attuale

int Delta_T_vel = ActualTimeVel - PastTimeVel;
if (Delta_T_vel >= SampleTimeVel) //Controllo il tempo passato (>= 30ms)
{
error_vel = Potenziometro - DT;
Error_vel_P = error_vel * Kp_vel;
Error_vel_I = (Error_vel_I + error_vel ) * (Delta_T_vel / 1e3) * Ki_vel;
if (Error_vel_I >= 255) //Anti Wind-up
{
Error_vel_I = 255;
}
else if (Error_vel_I <= -255)
{
Error_vel_I = -255;
}

Error_vel_D = (error_vel - lasterror_vel) * Kd_vel / (Delta_T_vel / 1e3);


Total_error_vel = Error_vel_P + Error_vel_I + Error_vel_D;

if (Total_error_vel >= 4096)
{
Total_error_vel = 4096;
}
```

```
else if (Total_error_vel <= -4096)
{
Total_error_vel = -4096;
}

lasterror_vel = error_vel; //Aggiornamento paramentri per il controllo
    successivo
PastTimeVel = ActualTimeVel;
}  //end condizione if
return Total_error_vel;
}


int MyPIDC(float Curr_ref, int FeedbackCurr, float Kp_curr, float Ki_curr)
{
unsigned long ActualTimeCurr = millis();
int Delta_T_curr = ActualTimeCurr - PastTimeCurr;
if (Delta_T_curr >= SampleTimeCurr)
{
error_curr = Curr_ref - FeedbackCurr;
Error_curr_P = error_curr * Kp_curr;
Error_curr_I = (Error_curr_I + error_c) * (Delta_T_curr / 1e3) * Ki_curr;

if (Error_curr_I >= 1024) //Anti Wind-up
{
Error_curr_I = 1024;
}
else if (Error_curr_I <= -1024)
{
Error_curr_I = -1024;
}

Total_error_curr += Error_curr_P + Error_curr_I;

if (Total_error_curr >= 15000)
{
Total_error_curr = 15000;
}
else if (Total_error_curr <= 0)
{
Total_error_curr = 0;
}
lasterror_curr = error_curr;
PastTimeCurr = ActualTimeCurr;
}
return Total_error_curr;
}
```

```
int Letture_curr() //funzione utilizzata per mediare letture
{
if(y < 30)
{
value_curr[i] = analogRead(A14);
for(int j = 0; j<30; j++)
{
curr_feed += value_curr[j];
}
curr_feed = curr_feed/30;
y++;
if(y == 30)
{
y=0;
}
}
return curr_feed;
}
```

Chiaramente l'implementazione è analoga a quella del PID di velocità. Particolare attenzione va posta invece sul tempo di campionamento poiché il PI di corrente deve essere più veloce dell'anello esterno ed in questo caso è stato scelto un tempo pari a 10ms. Per quanto riguarda la variabile di uscita del PI di corrente, questa andrà direttamente a modificare la posizione degli impulsi. Nel paragrafo 5.1 è stato infatti spiegato qual è il limite massimo e minimo per avere rispettivamente conduzione completa e nulla. Va studiato anche l'andamento della corrente della sorgente trifase. Essendo la frequenza di uscita pari a 300Hz, ci si aspetta che gli impulsi di corrente abbiano la stessa frequenza, infatti come è possibile osservare in figura 5.17 tra un impulso e l'altro c'è una distanza di 3.33ms. Essendo quindi la corrente analizzata non continua è stata utilizzata la funzione

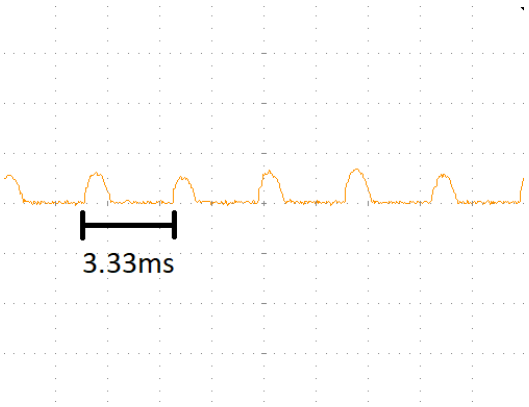


Figura 5.17. Andamento della corrente in uscita dal ponte diodi trifase

"*Letture_curr*" in modo tale da poter effettuare una media aritmetica sulle ultime 30 letture effettuate dalla scheda di valutazione.

5.3 Abilitazione controllori, allarmi e rampa di accelerazione

In tutti i sistemi moderni bisogna avere la possibilità di azionare o interrompere il lavoro del microcontrollore senza togliere l'alimentazione e questo è possibile attraverso un interruttore o un pulsante. Inoltre un buon sistema, oltre ad avere il controllo di una grandezza, deve essere capace di rilevare eventuali anomalie. Due tipologie di allarmi che non possono sicuramente mancare in un azionamento sono quelli dovuti alla mancanza di fase o alla rottura della dinamo tachimetrica in più date le caratteristiche meccaniche del motore è stata implementata anche una rampa di accelerazione.

Azionamento e interruzione del controllore L'implementazione di un interruttore è stata ovviamente eseguita sia a livello hardware che software. In particolare per quanto concerne la prima parte è stato utilizzato un elementare switch intercambiabile tra GND e 10V ed è stato posto in un ingresso digitale alla scheda di valutazione tramite un partitore. La funzione dello switch è molto semplice, quando questo è direzionato verso il GND

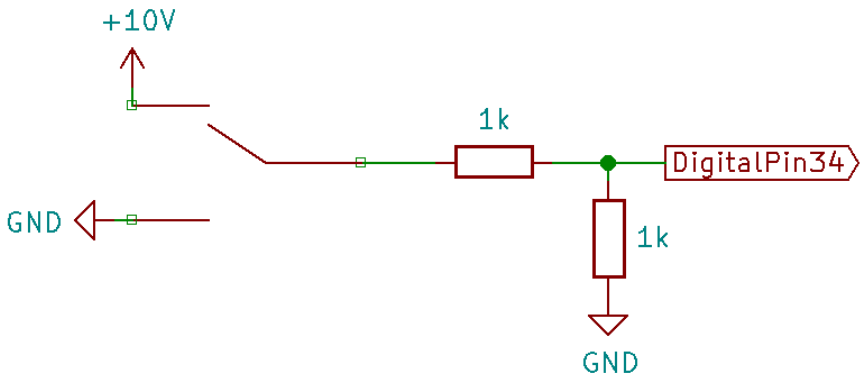


Figura 5.18. Interfaccia tra switch e Arduino

posiziona gli impulsi in condizione di conduzione nulla (Beta = 0) ed interrompe l'azione del PID e del PI, in questo caso anche se il potenziometro dovesse variare, nulla verrà messo in moto. In più, lo switch è anche collegato all'alimentazione dell'integrato "$ULN2004$" che blocca il passaggio degli impulsi impedendone l'arrivo al ponte. Nell'immagine 5.19 viene mostrato per intero il collegamento dello switch.

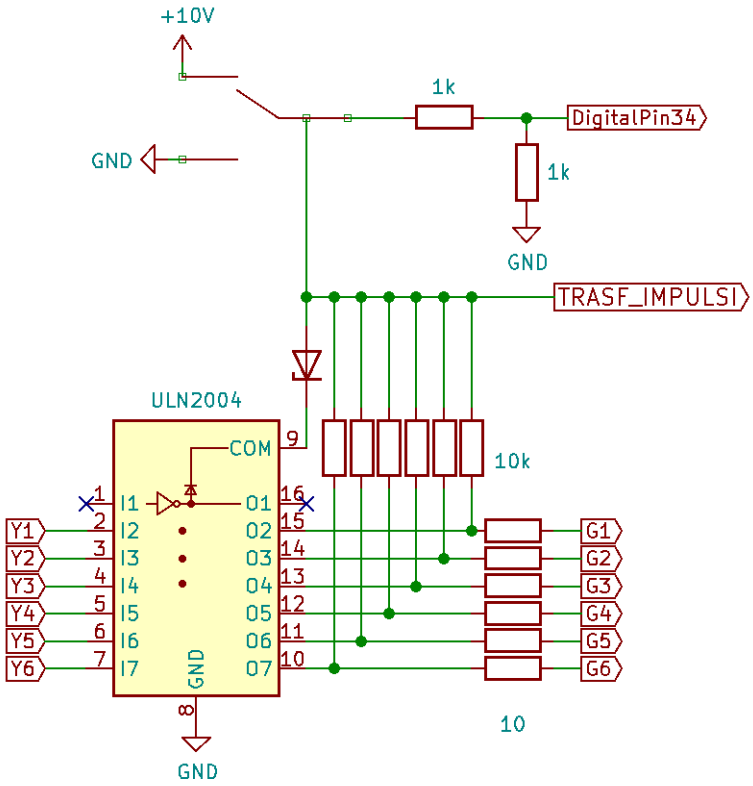
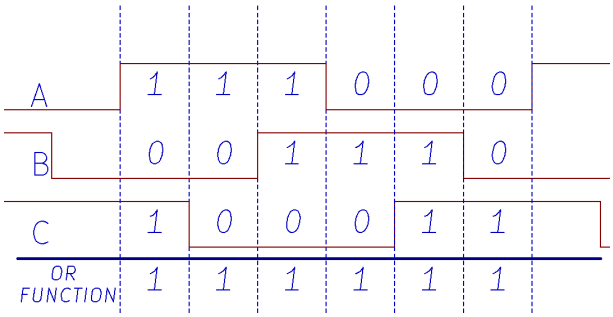
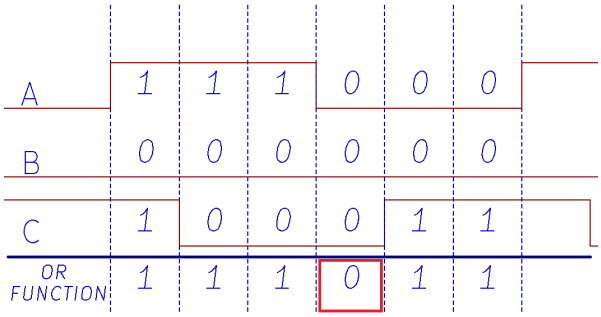


Figura 5.19. Collegamento dello switch per Ardino e ULN2004

Rilevazione mancanza fase Oltre a disabilitare i controllori, lo switch viene utilizzato per effettuare un controllo riguardo la presenza delle fasi o meno. Per rendere possibile ciò sono state analizzate le onde quadre che entrano nel microcontrollore (vedi 5.1).



(a)



(b)

Figura 5.20. Rilevazione presenza fasi attraverso logica OR

Come è possibile osservare in figura 5.20, durante il normale funzionamento se si esegue la funzione logica OR tra le onde, si avrà sempre un livello logico alto, mentre quando una fase viene a mancare si ha un periodo di 3.33ms in cui il livello logico diventa basso, generando quindi un segnale di allarme. Per rendere visibile ciò che accade è stato utilizzato un piccolo diodo LED che lampeggia con un periodo di 500ms;

Codice 5.5. Gestione del controllo per presenza di fase

```
void Mancanza_fase()
{
A = digitalRead(19); //lettura delle onde quadre
B = digitalRead(2);
C = digitalRead(3);
Fase_on = digitalRead(34); //lettura del valore dello switch
Orlogic = A | B | C;
if (Orlogic == false && Fase_on == true) //Situazione di allarme
{
```

```
   TCCR1B = (0 << CS12) | (0 << CS11) | (0 << CS10); //Blocco dei Timer
   TCCR3B = (0 << CS12) | (0 << CS31) | (0 << CS30);
   TCCR4B = (0 << CS42) | (0 << CS41) | (0 << CS40);
   PORTB = (0 << PB7) | (0 << PB6) | (0 << PB5) | (0 << PB4); //Tutte le
       uscite sono disattivate
   PORTH = (0 << PH6) | (0 << PH5);
   Allarme_fase(); //Richiamo della funzione che genera allarme
}
else if (Orlogic == false && Fase_on == false) //Potenza e switch disattivati
{
      Beta = 0;
      if( x == false )
      {
         lcd.clear();
         x = true;
      }
   lcd.setCursor(2, 0);
   lcd.print("Attivare potenza");
   lcd.setCursor(9, 1);
   lcd.print("e");
   lcd.setCursor(3, 2);
   lcd.print("Attivare PID");
   Fase_on = digitalRead(34);
   TCCR1B = (0 << CS12) | (0 << CS11) | (0 << CS10);
   TCCR3B = (0 << CS12) | (0 << CS31) | (0 << CS30);
   TCCR4B = (0 << CS42) | (0 << CS41) | (0 << CS40);
}
else if (Orlogic == true && Fase_on == false) //Attiva solo la parte di potenza
{
      if( x == true)
      {
         lcd.clear();
         x = false;
      }
   lcd.setCursor(4, 1);
   lcd.print("Attivare PID");
   Fase_on = digitalRead(34);
   TCCR1B = (0 << CS12) | (1 << CS11) | (0 << CS10); //Attivazione dei Timer
   TCCR3B = (0 << CS12) | (1 << CS31) | (0 << CS30);
   TCCR4B = (0 << CS42) | (1 << CS41) | (0 << CS40);
}
else if( Orlogic == true && Fase_on == true) //Potenza e switch attivi
{ //Do nothing
}
}


void Allarme_fase() //Funzione responsabile dell'allarme
{
```

```
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Allarme Fase!");
for (;;)
{
Allarme_FASE = digitalRead(37);
unsigned long ActualTimeAllFase = millis();
if (ActualTimeAllFase - PastTimeFas >= IntervalloAllFase) {
PastTimeFas = ActualTimeAllFase;


if (Allarme_FASE == false) {
Allarme_FASE = true;
} else {
Allarme_FASE = false;
}

digitalWrite(37, Allarme_FASE); //Output digitale collegato al LED
}
}
}
```

Nel codice appena riportato è possibile notare la presenza della funzione "*lcd*" utilizzata per la gestione di un piccolo display LCD 20x4 che sfrutta la comunicazione seriale attraverso i pin SDA e SCL di Arduino. In più la variabile "x" permette la pulizia di ciò che viene stampato sul display quando si passa da una condizione all'altra.

Allarme mancanza Dinamo tachimetrica Questo è il secondo allarme fondamentale in un sistema sicuro. Può capitare che la Dinamo Tachimetrica possa rompersi, in tal caso nonostante il motore giri, il segnale generato dalla DT sarà nullo, questo comporta una situazione chiamata "$Motore\ in\ fuga$" in cui il motore tenderà ad arrivare al massimo ma senza alcun controllo. Questo accade quando il segnale del potenziometro è diverso da zero e il segnale della DT è nullo, si avrà quindi sempre un errore che i controllori proveranno a compensare senza mai riuscirci incrementando in continuazione la variabile che sposta gli impulsi. Anche in questo caso è stato fatto uso di un piccolo LED per rendere visibile la situazione di allarme.

```
void Mancanza_dynamo(float Curr_ref, int DT)
{
   if (Beta > 7500 && DT <= 50 && Fase_on == true) //7500 metà dinamica
   {

      Allarme_DT();
   }
   else
   {
      \\Do nothing
```

```
   }

}


void Allarme_DT()
{

lcd.clear();
lcd.setCursor(0,0);
lcd.print("Allarme dinamo!");
TCCR1B = (0 << CS12) | (0 << CS11) | (0 << CS10); //Disabilitazione Timer
TCCR3B = (0 << CS12) | (0 << CS31) | (0 << CS30);
TCCR4B = (0 << CS42) | (0 << CS41) | (0 << CS40);
PORTB = (0 << PB7) | (0 << PB6) | (0 << PB5) | (0 << PB4);
PORTH = (0 << PH6) | (0 << PH5);
for (;;)
{
Allarme_dinamo = digitalRead(30);
unsigned long ActualTimeAlldinamo = millis();
if (ActualTimeAlldinamo - PastTimedinamo >= IntervalloAlldinamo)
{
PastTimedinamo = ActualTimeAlldinamo;

if (Allarme_dinamo == false)
{
   Allarme_dinamo = true;
}
else
{
   Allarme_dinamo = false;
}
digitalWrite(30, Allarme_dinamo);
}
}
}
```

In questo caso l'implementazione è molto semplice, è sufficiente fare un controllo sulla variabile che sposta gli impulsi (Beta) e se questa supera un certo limite quando il segnale della DT è prossimo alla zero, allora viene generato l'allarme in questione.

Rampa di accelerazione A causa delle caratteristiche meccaniche del motore è fortmemente consigliabile l'implementazione di una rampa di accelerazione per evitare che questo entri in limite di corrente. In questo caso il riferimento del PID di velocità non sarà più direttamente il segnale del potenziometro ma una variabile interna ad Arduino che raggiungerà il valore del potenziometro in un tempo prestabilito. In figura 5.21 viene riportato l'andamento del segnale interno ad Arduino con un tempo pari a 3s.

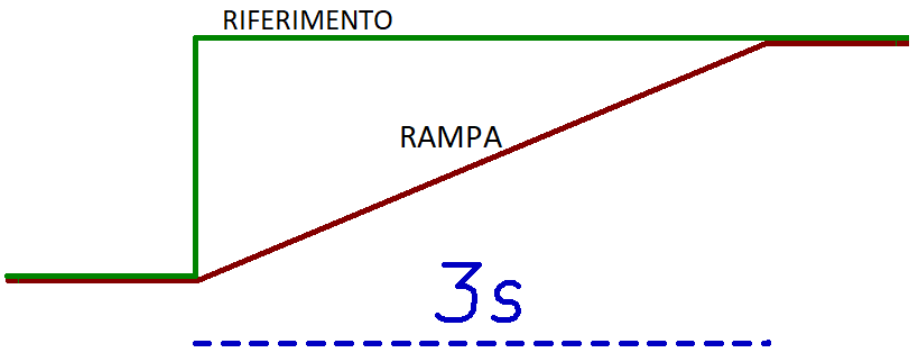


Figura 5.21. Rampa di accelerazione

Codice 5.6. Implementazione Rampa di accelerazione

```
#define slope_time
unsigned long SampleTimeSlope = 1; //1 millisecondo;
float Var_slope = 920 / ((slope_time * 1000) / SampleTimeSlope) ;

void loop()
{
   Potenzio = analogRead(A6);
   Potenziometro = Slope(Potenzio, Var_slope);
}

float Slope(float Potenzio, float Var_slope)
{
   unsigned long ActualTimeSlope = millis();

   int Delta_T_Slope = ActualTimeSlope - PastTimeSlope;
   if (Delta_T_Slope >= SampleTimeSlope)
      {
      if (Potenzio > Potenziometro)
      {
         Potenziometro += Var_slope;
      }
   else if (Potenzio < Potenziometro)
      {
         Potenziometro -= Var_slope;
      }
   else
      {
      Potenziometro = Potenzio;
      }
PastTimeSlope = ActualTimeSlope;
```

```
}
return Potenziometro;
}
```

Particolare attenzione va posta alla variabile "Var_slope" che andrà a creare passo-passo la rampa aggiungendo o togliendo alla variabile "$Potenziometro$" una determinata quantità.

$$Var\_slope = \frac{920}{\frac{1000 \times slope\_time}{SampleTimeSlope}} \tag{5.3}$$

Dove 920 è la quantità massima raggiungibile dal valore del potenziometro in ingresso ad Arduino (vedi fig. 5.14) tenendo conto che 1023 equivale a 5V. $Slope\_time$ invece è il tempo prestabilito che deve impiegare la rampa per raggiungere il valore del potenziometro. Infine "$SampleTimeSlope$" è la frequenza con cui viene aggiunta o tolta la quantità dalla variabile "$Potenziometro$" che in fase di progetto accade ogni millisecondo. La componente "$SampleTimeSlope$" è molto importante poichè definisce anche la precisione con cui la rampa raggiunge il valore di arrivo.

5.4 Risultati sperimentali finali

Nel seguente paragrafo verranno mostrate le forme d'onda ottenute, la caratteristica dinamica del sistema, una panoramica generale del banco prova e il motore utilizzato. In particolare verranno utilizzati diversi valori per la rampa di accelerazione e verrà mostrata la posizione degli impulsi in condizione di minima e massima conduzione evidenziando dopo quanto tempo questi vengono generati rispetto alla semionda di riferimento

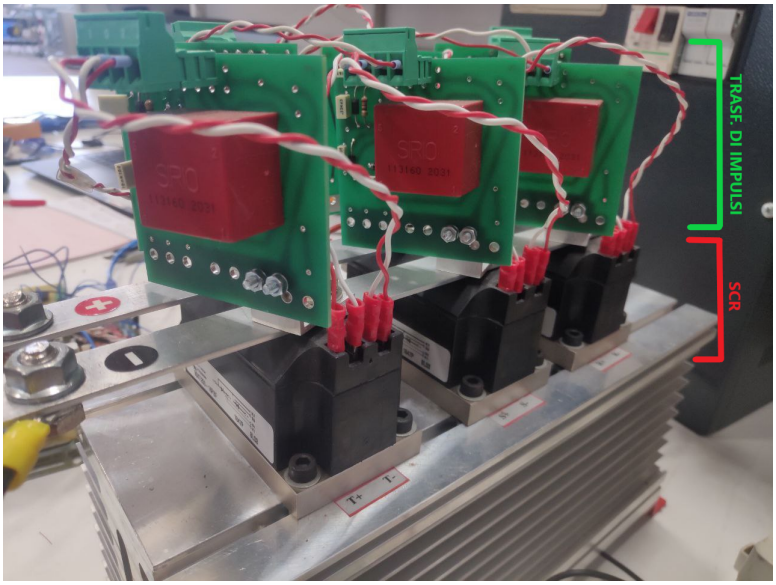


Figura 5.22. Ponte raddrizzatore trifase SCR utilizzato in fase di progetto

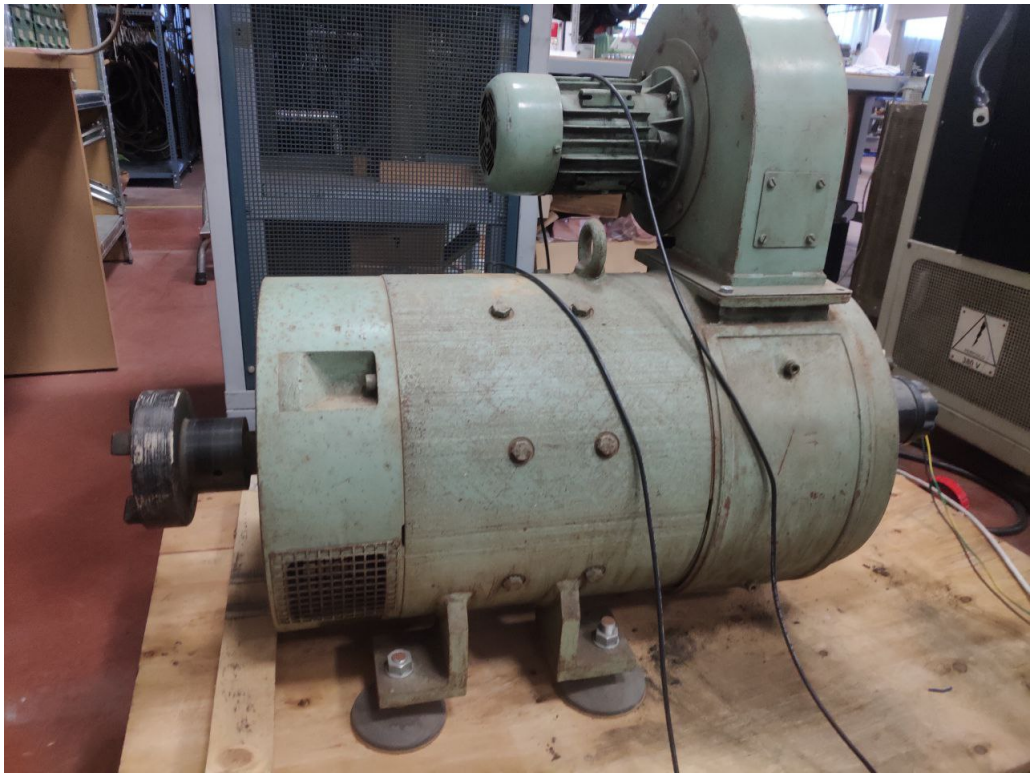


Figura 5.23. Motore CC da 38.72kW utilizzato in fase di progetto

In figura 5.24 sono riportati tutti i circuiti utili al raggiungimento dell'obiettivo, in particolare:

- Rosso: Circuito composto dai comparatori LM339, NE555, M74HC08B1 e ULN2004
- Azzurro: Circuiti integrati CD4071
- Verde: Potenziometro di riferimento per l'anello di velocità
- Marrone: Trasformatore utilizzato per ridurre sorgente trifase
- Nero: Circuito composto dai TA per l'anello di corrente
- Arancione: Display lcd 20x4
- Blu: Regolatori di tensione 10V, 15V e 5V

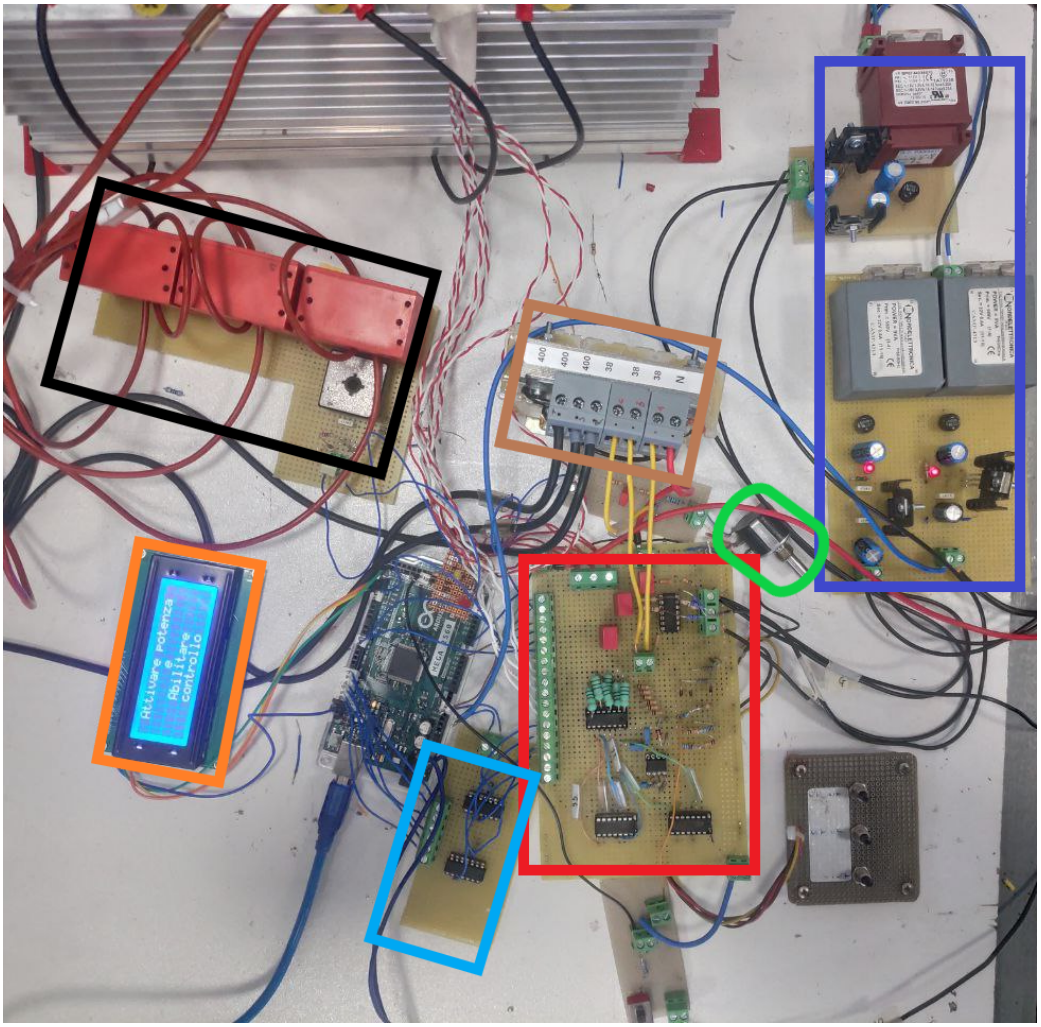


Figura 5.24. Banco di prova utilizzato in fase di progetto

Forme d'onda Di seguito sono riportate le forme d'onda del sistema.

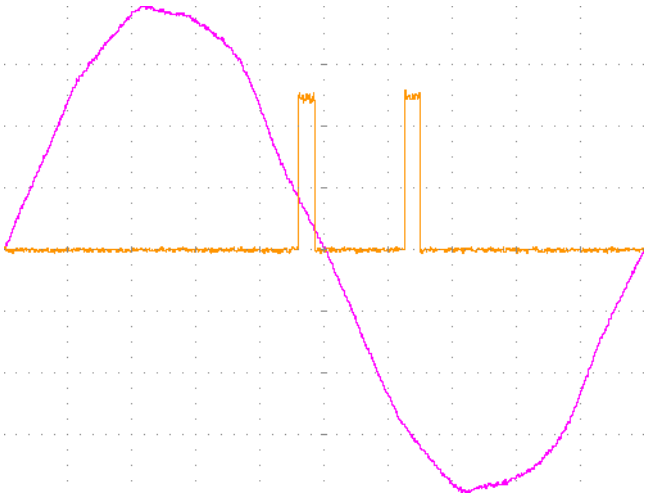


Figura 5.25. Posizione della coppia di impulsi in condizione di conduzione nulla

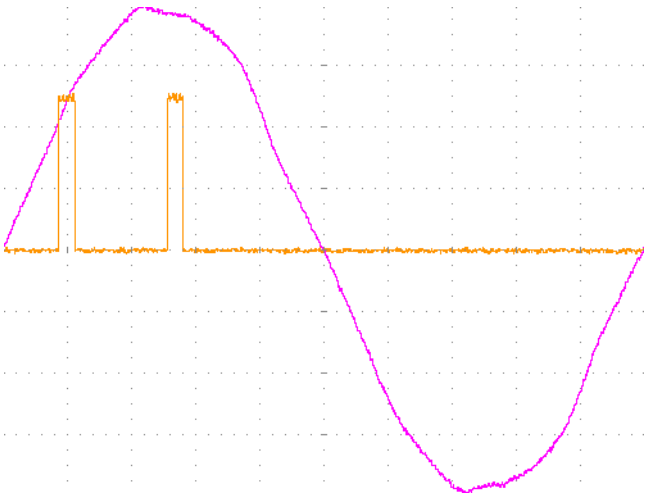


Figura 5.26. Posizione della coppia di impulsi in condizione di conduzione massima

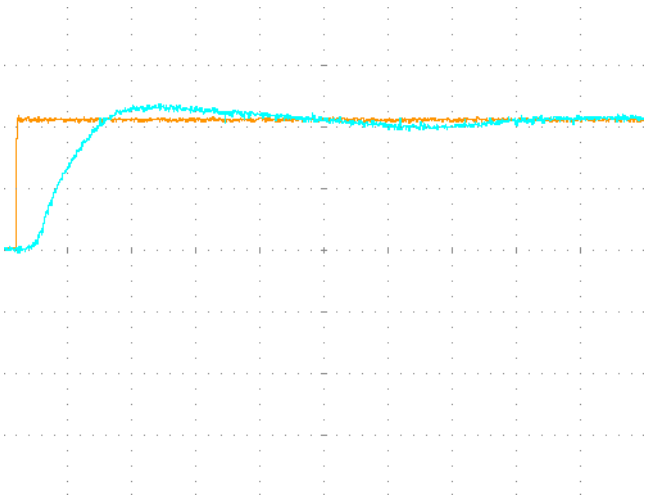


Figura 5.27. Modello dinamico con rampa nulla e V_a=220V (500ms/div)

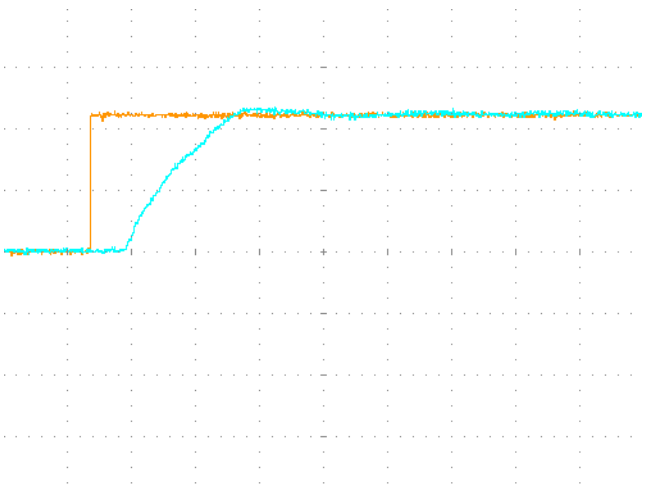


Figura 5.28. Modello dinamico con rampa 2s e V_a=440V (1s/div)

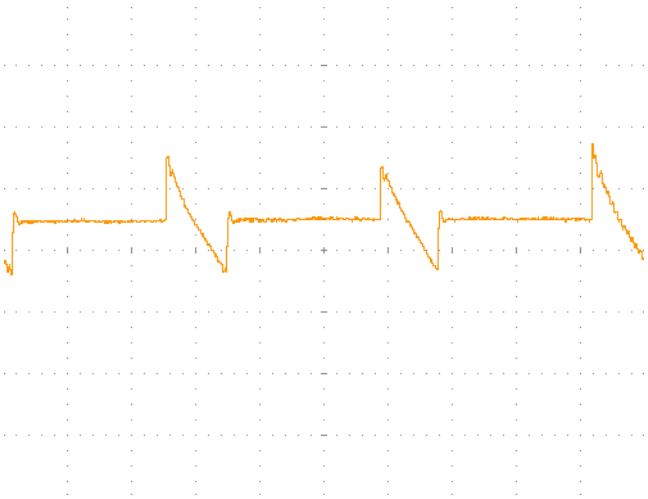


Figura 5.29. Tensione di armatura V_a in arrivo dal ponte trifase SCR

É necessario spendere due parole riguardo la tensione di armatura V_a, in particolare è possibile notare una parte superiore ed una inferiore. Queste rappresentano rispettivamente l'esatto momento in cui l'SCR sta conducendo e la forza contro elettromotrice che si viene a creare dovuta alla presenza del motore sul carico che è composto da una parte resistiva e una induttiva.

Conclusioni

In questo lavoro di tesi l'obiettivo è stato progettare un sistema di azionamento in corrente continua preoccupandosi sia della parte software che della parte hardware rendendo il tutto stabile grazie all'implementazione di un PID e un PI. Riguardo la parte hardware, la parte più complicata è risultata essere quella dedicata alla generazione delle onde quadre utili per generare gli interrupt e avere il giusto sincronismo. In particolare una prima soluzione è stata quella di fare uso di optoisolatori ma è risultato un sistema impreciso poichè questi componenti devono avere in ingresso una tensione sopra una certa soglia per operare e ciò comportava avere in uscita delle onde quadre con un Duty Cycle minore del 50%, specifica impensabile in un sistema che lavora a 50Hz per un periodo di 20ms. Per quanto concerne la parte software invece, due sono stati gli ostacoli da superare, il primo rigurda la generazione degli impulsi, in particolare come rendere possibile la ricerca del fronte di salita e di discesa dell'onda quadra per generarli al momento esatto e senza avere pendolamenti. Il secondo problema è sorto durante l'implementazione del PID e del PI. In letteratura e in rete sono tanti gli esempi riportati perciò è stato speso del tempo per trovare il più adatto.

Anche la ricerca dei parametri dei controllori ha richiesto del tempo, in particolare è importante dire che prima di utilizzare un motore CC è stato fatto uso di un carico resistivo. Per quest'ultimo è stato possibile utilizzare sia per PID che PI il metodo di Ziegler-Nichols essendoci una relazione lineare tra corrente e carico. Nel motore CC invece, in prima analisi si è utilizzato esclusivamente un controllo in tensione e anche in questo caso il metodo ZN è risultato efficace trovando parametri iniziali molto vicini a quelli utilizzati nel finale. Quando successivamente si è decisi di utilizzare anche il PI di corrente è risultato più efficace fare uso del metodo manuale a causa della relazione non lineare che si ha tra corrente e carico essendo il motore CC composto da una parte induttiva.

In defintiva entrambi sono sicuramente metodi versatili per la ricerca del K_p, K_i e K_d, il metodo ZN permette di risparmiare del tempo e nonstante ciò a volte è possibile che i parametri trovati non siano del tutto corretti. Tuttavia non sempre è possibile ottenere oscillazioni sostenute ed è quindi necessario utilizzare il metodo manuale che richiede più esperienza da parte del progettista e un tempo maggiore.

L'impostazione dei parametri dei controllori è stata effettuata prima che il motore iniziasse a girare. Generalmente, in sistemi come questo, la ricerca dei guadagni viene effettuta "*online*", ovvero durante il funzionamento del motore in modo tale da poter osservare in tempo reale come risponde il carico. Sicuramente questo sarà uno dei lavori

futuri rendendolo possibile attraverso l'uso di un tastierino e un display, entrambi interfacciati con la scheda di valutazione. Riguardo la parte hardware tutto è stato saldato su schede millefori dopo essere stato testato su breadboard, ciò però risulta poco ordinato e comprensibile per utenti esterni. Per questo motivo un prossimo step è quello di trasferire tutto su PCB utilizzando tecniche di sbroglio.

Bibliografia

[1] Prof. Andrea Cavagnino. *Macchine Elettriche*. Appunti delle lezioni, 2005.

[2] Sang hun Kim. *Electric motor control : DC, AC, and BLCD motors*. Amsterdam, Netherlands: Elsevier, 2017.

[3] Wikipedia. Tiristore — wikipedia l'enciclopedia libera. `https://it.wikipedia.org/wiki/Tiristore`, 2021.

[4] `https://moodle.calvino.ge.it/pluginfile.php/5932/mod_resource/content/1/23.%20Thyristors%20e%20Controllo%20di%20Potenza.pdf`.

[5] Ned Mohan, Tore M. Undeland, and William P. Robbins. *Power Electronics. Converters, Applications and Design*. John Wiley and Sons, Inc, third edition, 2003.

[6] `https://store.arduino.cc/products/arduino-mega-2560-rev3?selectedStore=eu`.

[7] Wikipedia contributors. Pid controller — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=PID_controller&oldid=1066921735`, 2022.

[8] Rick Anderson and Dan Cervo. *Pro Arduino*. TECHNOLOGY IN ACTION.

[9] `http://brettbeauregard.com/blog/2011/04/improving-the-beginner%e2%80%99s-pid-reset-windup/`.

[10] `https://docs.rs-online.com/fb17/0900766b80fbcdd8.pdf`.

[11] `http://www.sirio-ic.com/images/sirio/pdf/TA_151008.pdf`.