Università degli Studi di Padova
Facoltà di Ingegneria

Finito di scrivere il giorno 21 luglio 2010 utilizzando LaTeX 2ε

UNIVERSITÀ DEGLI STUDI DI PADOVA
FACOLTÀ DI INGEGNERIA

—

DIPARTIMENTO DI INNOVAZIONE MECCANICA E GESTIONALE

—

TESI DI LAUREA TRIENNALE IN INGEGNERIA DELL'AUTOMAZIONE

PROGETTAZIONE E REALIZZAZIONE DI DRIVER DI MOTORE PASSO

RELATORE: CH.MO PROF. ING. GIULIO ROSATI

LAUREANDO: CARRON ANDREA

ANNO ACCADEMICO 2009-2010

A mio zio Giuseppe.

“ Yes, there are two paths you can go by ”

Jimmy Page, Robert Plant - 1971

Indice

Sommario

Un manipolatore scara è un robot a tre gradi di libertà in grado di coprire un determinato spazio operativo orizzontale. Negli ultimi anni si è puntato generalmente nel realizzare robot con più gradi di libertà ridondanti in modo da poter evitare punti di lavoro singolari[1].
Il robot realizzato presenta quindi tre gradi di libertà (quattro contando anche l'end effector), grazie ad una base che muove lungo la direzione orizzontale il manipolatore, e tale movimentazione è ottenuta tramite un motore passo collegato ad una vite senza fine.
Molto importante è quindi la realizzazione di un driver per il motore passo che permetta di utilizzare tale motore al massimo delle sue prestazioni.

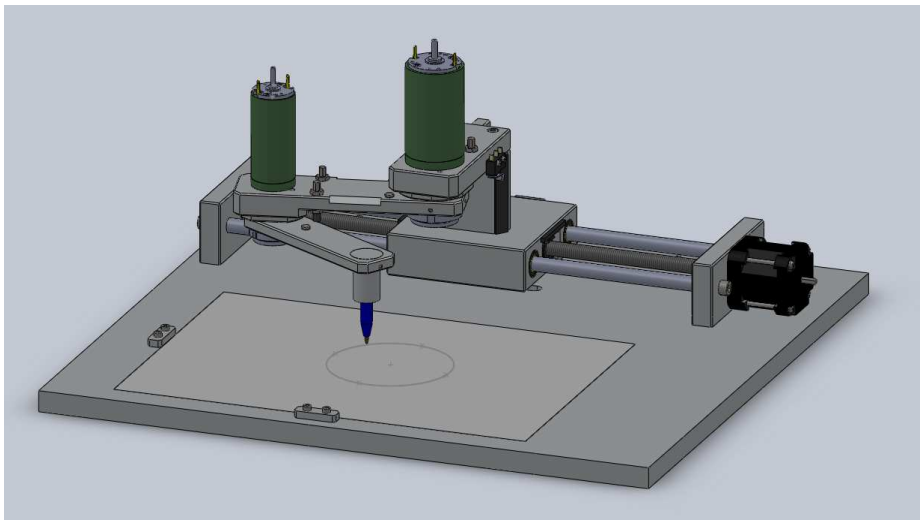

Figura 1: Robot planare.

Introduzione

La tesi che viene descritta nelle seguenti pagine descrive come progettare e realizzare un driver per motore passo e il relativo quadro elettrico.
Questo progetto si inserisce all'interno della realizzazione di un manipolatore piano a 4 gradi di libertà, che è possibile trovare all'interno dell'officina del Dipartimento di Innovazione Meccanica e Gestionale (DIMEG) dell'Università di Padova.

Capitolo 1

Introduzione al progetto

Il manipolatore progettato nasce sull'idea di uno SCARA, acronimo di Selective Compliant Assembly Robot Arm, ossia da un manipolatore a tre gradi di libertà capace di muoversi liberamente sul piano orizzontale e di compiere un movimento sull'asse verticale. Il movimento sull'asse verticale di solito si limita al poter alzare ed abbassare l'end effector per poter interagire con l'ambiente esterno.

A differenza dello SCARA il manipolatore da noi in esame ha una base mobile, ciò introduce un grado di libertà ridondante. Negli ultimi anni si è visto crescere il numero di manipolatori ridondanti perchè anche se non riescono ad eliminare le configurazioni singolari, il grado di libertà aggiuntivo permette di generare configurazioni alternative non singolari.

Il manipolatore da noi in esame ha anche un grado di libertà sull'asse verticale. L'end effector, che nel nostro caso sarà una matita e traccerà su un foglio A4 una parte a piacere della traiettoria da seguire. Lo scopo del progetto è infatti quello di realizzare un robot che segua correttamente le traiettorie pianificate.

1.1 Parte Meccanica

Per quanto riguarda l'implementazione della parte meccanica, i motori sono stati posizionati sopra i relativi link, e quindi a causa del peso di quest'ultimi è stato necessario realizzare i membri in alluminio e non in acrilonitrile-butadiene-stirene (ABS) come si era inizialmente pensato.

Figura 1.1: Manipolatore Scara.

Per quanto riguarda i motori invece i due membri vengono mossi da due motori in continua, quello sul secondo link si è cercato di sceglierlo più leggero possibile per evitare di caricare di troppo peso il manipolatore. Entrambi i motori lavorano in catena chiusa e la loro posizione viene letta da due encoder incrementali, inoltre non è previsto l'uso di riduttori e quindi i motori lavorano in presa diretta.
Per quanto riguarda il movimento della base invece si è optato per un motore passo passo ed una vite senza fine per trasformare il movimento rotatorio del motore in un movimento longitudinale.
Inoltre è stata scartata l'idea della guida lineare da accoppiare al motore passo passo,sostituendola con due sbarre cilindriche sulla quale scorre la chiocciola. La base come si può capire lavora in catena aperta in quanto non è presente nessun

encoder o comunque nessun trasduttore di posizione.
La scelta dei motori è stata effettuata tramite l'ausilio di matlab. E´ stato implementato il modello dinamico del manipolatore per permettere di calcolare in base alla traiettoria da seguire la coppia necessaria per eseguire il movimento, e questo punto è stata cercata la movimentazione che richiedava la maggiore coppia per ogni link. Su questi dati è stata effettuata la scelta dei motori, analizzando soprattutto nei cataloghi i valori della coppia di stallo e della coppia massima continua.

1.2 Pianificazione e Controllo

La pianificazione della traiettoria è indubbiamente una parte fondamentale per un corretto funzionamento del manipolatore, soprattutto avendo un grado di libertà aggiuntivo. La realizzazione di tale parte è stata effettuata tramite Matlab, creando un GUI, cioè un interfaccia grafica con cui l'utente si può interfacciare, ad hoc.
E´ stata prevista l'implementazione di pianificazioni per moti di tipo punto-punto nello spazio dei giunti usando leggi di tipo ad accelerazione costante, trapezoidale e polinomiali di terzo e quinto grado. Sono poi stati previsti anche pianificazioni nello spazio dei giunti di percorsi su n-punti usando polinomi cubici raccordati (spline) e pianificazioni della traiettoria o di n-traiettorie nello spazio operativo definendo dei percorsi e delle leggi orarie. Il tempo imposto per eseguire il movimento è di un secondo.
La pianificazione è stata implementata su un computer con sistema operativo Windows Xp e comunica con il controllo, che viene eseguito su un altro calcolatore, tramite una connessione di rete utilizzando un protocollo UDP. La scelta del protocollo UDP rispetto al più utilizzando TCP è legata a motivi di velocità. Infatti quest'ultimo protocollo è di tipo con connessione con riscontro, e quindi prima di poter utilizzare un pacchetto è necessario riceverlo ed inviare un pacchetto di avvenuta e corretta ricezione. Visto che il supporto fisco di collegamento è un cavo ethernet di lunghezza ridotta, è poco probabile perdere pacchetti o riceverli corrotti per motivi legati al rumore, e quindi risulta conveniente utilizzare

un protocollo senza connessione e senza riscontro come l'UDP. In poche parole una volta ricevuto il pachetto e verificata l'integrità dello stesso questo può essere direttamente utilizzato.
Il controllo è stato quindi implementato su un elaboratore a parte,e per ovviare il problema di lentezza dei sistemi operativi multitasking, che dovendo serivire più task assieme non permettono di lavorare con tempi di campionamento prossimi al ms, è stato scelto di utilizzare un sistema real-time. Per la precisione il sistema usato è RealTime Windows Target di Matlab.
Il controllo è stato implementato su una macchina a stati e appena avviato il sistema si deve occupare di effettuare la calibrazione del manipolatore, perchè utilizzando encoder di tipo incrementale e non di tipo assoluto non è nota a priori la posizione dello stesso. La posizione post-calibrazione scelta è quella con la base esattamente al centro della rotaia e i due link completamente distesi.
La calibrazione prevede una fase iniziale dove viene mosso singolarmente ogni membro fino a raggiungere il relativo finecorsa induttivo, posto prima del finecorsa meccanico (i quali, se azionati, mandano il sistema in emergenza e bloccano i motori), ed essendo in una posizione nota permette facilmente di raggiungere la posizione finale di calibrazione. Per quanto riguarda la base invece sarà necessario raggiungere il finecorsa ed eseguire il numero corretto di passi per portarsi nella posizione centrale. La scelta di utilizzare finecorsa induttivi per la calibrazione è legata anche alla loro precisione, ma sopratutto alla loro ripetibilità, caratteristica fondamentale per un corretto posizionamento del manipolatore. Una volta terminata la fase di calibrazione e determinato dalla pianificazione quella che è la traiettoria da seguire il controllo riceverà il riferimento nello spazio dei giunti o nello spazio operativo(e quindi sarà necessario conoscere la cinematica diretta ed inversa a seconda dell'implementazione). Per i due link il set point verrà sottratto al riferimento di posizione ricevuto dagli encoder e quindi il segnale di errore verrà mandato ad un controllore.
Sono previste l'implentazione di un controllore con schema decentralizzato cioè dove ogni link e la base è un sistema indipendente che però non terrà conto dell'influenza del movimento di un membro sugl'altri, e un controllo di tipo centralizzato che riceve in ingresso il la traiettoria totale e non dei singoli membri e

tramite una dinamica inversa calcola la coppia che è necessario applicare ad ogni link per completare il movimento in modo corretto.
Per concludere il controllo si interfaccia con le schede dei driver e con gli encoder attraverso la scheda di interfaccia della sensoray modello 626. Le principali caratteristiche sono:

- 48 ingressi/uscite digitali
- 16 ingressi analogici differenziali (con risoluzione a 14 bit)
- 4 uscite analogiche (con risoluzione a 13 bit)
- 6 contatori a 24 bit che possono essere usati per encoder incrementali
- Possibilità di selezionare la modalità 1x,2x o 4x per gli encoder.

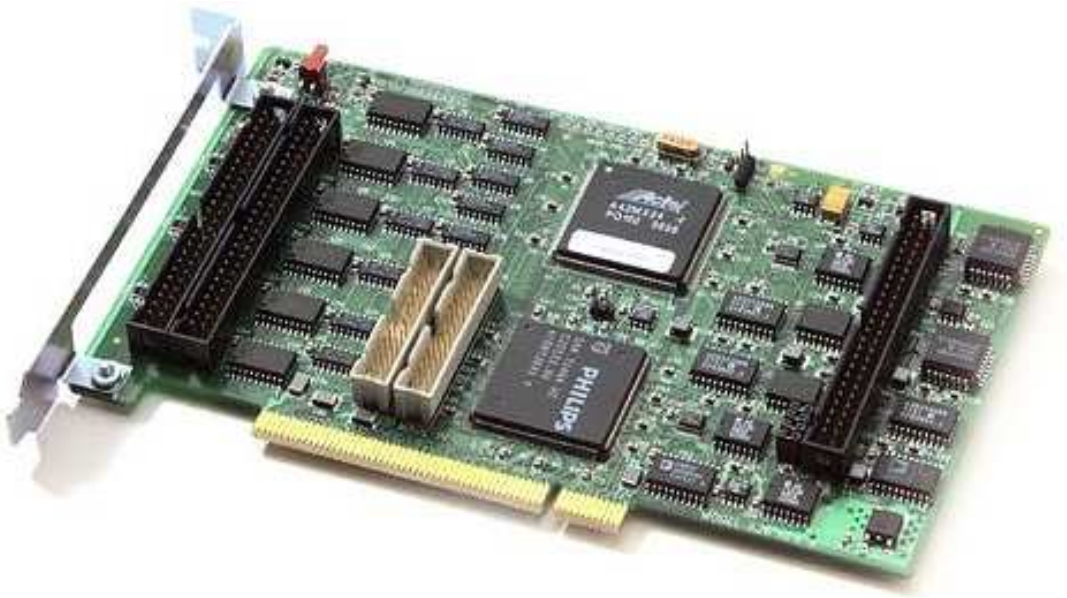

Figura 1.2: Scheda d'interfaccia Sensoray 626.

1.3 Elettronica e Driver

Per quanto riguarda la parte dell'elettronica è stato necessario prevedere un circuito d'emergenza che controllasse lo stato dei finecorsa e la temperatura dei

motori. Qual'ora fosse segnalato uno dei due pericoli il circuito deve interfacciarsi correttamente con lo stadio in potenza e permettere di effettuare una frenata attraverso l'integrato che è direttamente collegato ai motori. Inoltre deve essere possibile frenare il sistema tramite un pulsante di emergenza e dal controllo. Quest'ultimo deve essere chiaramente informato sullo stato del sistema per capire quando viene avviata la frenata elettronica.
Molto spesso il driver viene rappresentato con un semplice guadagno, ma questa è sicuramente una semplificazione che non tiene conto delle molte non linearità che sono presenti. Uno schema generale è dato dalla figura sottostante.

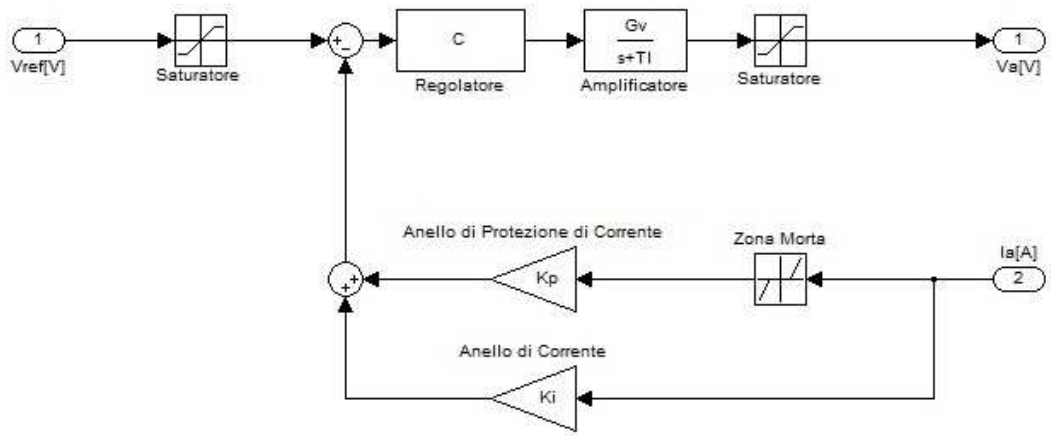

Figura 1.3: Schema logico del driver.

Il funzionamento è molto semplice, la tensione in usicta del controllore arriva al driver, per simulare i limiti di tensione dello stesso è necessario introdurre un saturatore,il segnale quindi passa attraverso un regolatore che non è implementato effettivamente in tutti gli schemi per poi arrivare ad un amplificatore di tensione. Per simulare la saturazione dell'amplificatore è necessario inserire il relativo saturatore con limiti circa pari alla tensione di alimentazione. A questo punto il segnale può essere direttamente applicato al motore. E´ presente una retroazione dalla lettura della corrente, che viene fatta internamente al driver. L'anello di protezione di corrente agisce quando si cerca di superare la corrente massima o la corrente massima continua. Il suo comportamento può essere rappresentato da

una dead-zone con estremi variabili, il driver permetterà di poter applicare la corrente massima solo per tempi ridotti per evitare surriscaldamenti del motore per poi subito modificare i limiti della zona morta a quelli più stringenti. Il secondo anello è l'anello di corrente, anch'esso non è implementato in tutti gli schemi, e permette tramite il coefficiente K_i di confrontare la tensione ai capi del motore e quella desiderata generando quindi un segnale di errore che andrà effettivamente a pilotare l'amplificatore.

I driver possono essere implementati o in controllo di tensione e quindi in questo caso avremo il regolatore C(s)=1 e il coefficiente $K_i = 0$. Supponendo di non essere in saturazione e di non aver attivato l'anello di protezione di corrente si vede come il driver può essere semplificato ad un semplice guadagno. Il driver pilotato in tensione permette ottime prestazioni e ottime reiezioni ai disturbi, questo lo si può vedere collegandolo ad un modello del motore in corrente continua qui presente.

$$M(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K_m}{1 + sT_m} \tag{1.1}$$

Ecco invece il modello dell'accoppiata motore e driver in controllo di tensione.

$$\dot{\Theta}(s) = \frac{\frac{G_v}{K_v}}{1 + sT_m} V_{ref} - \frac{\frac{R_a}{K_t K_v}}{1 + sT_m} C_r \tag{1.2}$$

Prendendo i dati del motore 35NT2R82 della Portescap abbiamo che la costante di tempo $T_m = 12ms$, mentre il numeratore della funzione di trasferimento disturbo-uscita è pari a $\frac{R_a}{K_t K_v} = 1,87 * 10^4$, che all'apparenza può sembrare molto grande, ma ricordando che i disturbi sono nell'ordine di pochi mNm il disturbo in uscita è nell'ordine di poche centinaia di RPM.Il comportamento quindi di questo tipi di driver è ottimo e molto performante.

L'altro schema implementabile quello del driver pilotato in corrente. In questo caso $K_i \neq 0$, questo guadagno converte la tensione che riceve dal controllore in una corrente che poi andrà a pilotare il motore. Anche in questo caso il driver può essere schematizzato con un semplice guadagno se siamo nell'ipotesi di non essere in saturazione.

E´ necessario però utilizzare un modello diverso del motore, perchè a differenza di

prima dobbiamo pilotare il motore in corrente. Il modello del motore è il seguente:

$$\frac{\dot{\Theta}(s)}{C(s)} = \frac{\frac{1}{F_m}}{1 + s\frac{I_m}{F_m}} \tag{1.3}$$

Introducendo quindi in driver in corrente otteniamo:

$$\dot{\Theta}(s) = \frac{\frac{K_t}{K_i F_m}}{1 + s\frac{I_m}{F_m}} V_{ref} - \frac{\frac{1}{F_m}}{1 + s\frac{I_m}{F_m}} C_r \tag{1.4}$$

In questo caso il valore della costante di tempo è pari a 407ms, nettamente più lento rispetto al driver in tensione, e il valore di $F_m = 6,68x10^8$ indica che abbiamo una minore reiezione ai disturbi. Questo tipo di controllo si adatta particolarmente bene quando si sa quanta corrente è necessaria e quando si conoscono molto bene quelle che possono essere le coppie che creano dei disturbi.
E´ necessaria quindi una scheda d'interfaccia che permetta di collegare correttamente l'elaboratore nel quale è implementato il controllo,i tre driver, il circuito d'emergenza e gli encoder.
Lo schema del driver del motore passo passo deve essere comando da un microcontrollore della Microchop, lo stadio in potenza deve essere realizzato tramite un integrato della National Semiconductor, per l'esattezza l'lmd18200 che permette di avere una tensione di alimentazione massima di 55V, una corrente massima continua di 3A e una corrente massima di 6A. Per finire deve essere realizzando un circuito che permetta la corretta gestione dei segnali che arrivano dai finecorsa meccanici. Lo stesso schema viene utilizzato anche per i due motori in continua. Per permettere la comunicazione tra i tre pic e un computer tramite la porta seriale si utilizza un integrato della Maxim, il max232.
Per concludere deve essere realizzato un quadro che permetta di contenere all'interno tutta l'elettronica.

1.4 Obiettivi del progetto

L'obiettivo della tesi è quindi quello di realizzare un driver per pilotare un motore passo, una scheda di interfaccia per permettere la comunicazione con la scheda sensoray 626 e un quadro elettrico dal quale i driver possano essere alimentati e

si possano gestire le varie emergenze.

Uno schema generale del progetto è quindi il seguente:

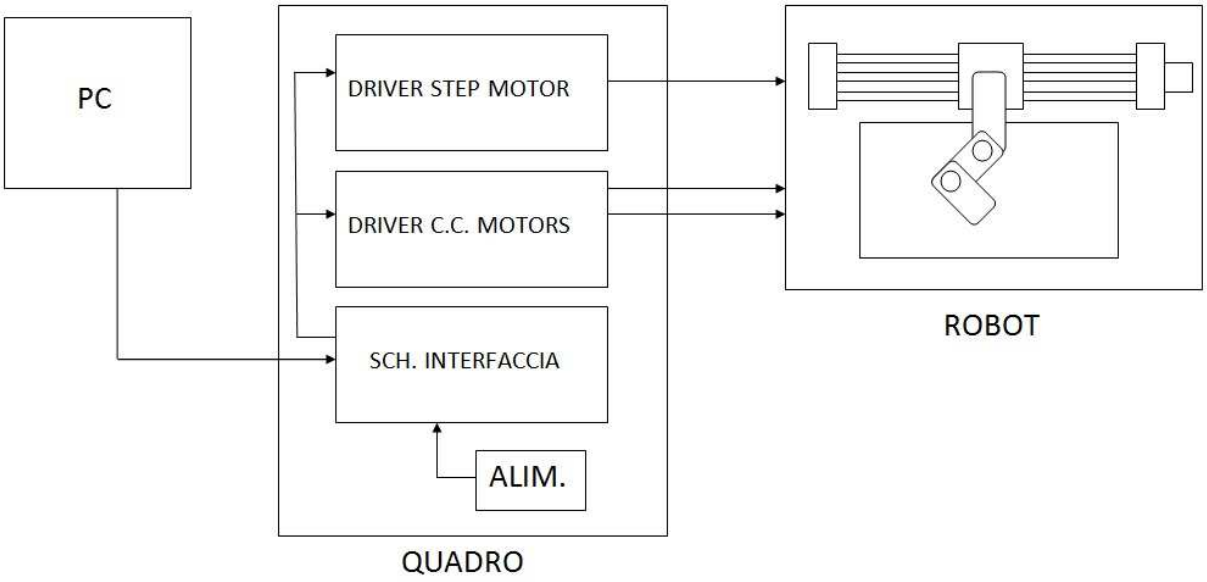

Figura 1.4: Schema generale del progetto.

Capitolo 2

Driver motore passo

Il driver passo passo parte da delle specifiche molto chiare, in quanto deve ricevere in ingresso il numero di passi che deve effettuare in un millisecondo, ossia la frequenza dei passi, a questo punto questo dato deve essere tradotto nel numero di passi che deve essere eseguito in un ciclo di clock e per evitare un comportamento scattoso ed evitare impuntamenti, il numero di passi eseguiti deve essere il più possibile uniformemente distribuito lungo tale periodo. Per finire sulla stessa scheda stata implementato anche un circuito che permette di far comunicare i microcontrollori con un elaboratore tramite la porta seriale. Per garantire l'espandibilità di tale driver è stato inserito un connettore che permette di collegarsi ad alcuni ingressi/uscite digitali del microcontrollore (tra i quali anche quello del bus I2C) ed un connettore che riunisce i piedini di ingresso/uscita usati per la comunicazione seriale per poter prevedere in un futuro la comunicazione tra più microcontrollori. Lo schema generale è il seguente:

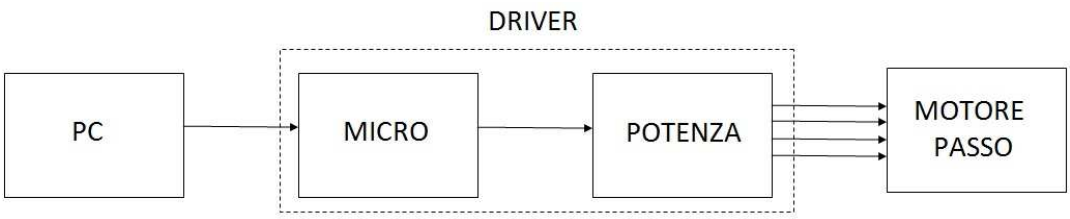

Figura 2.1: Schema generale del driver.

2.1 Scelta delle componenti

La scelta delle componenti riguarda essenzialmente le tre parti principali della scheda del driver:

- La scelta del microcontrollore.
- La scelta dello stadio di potenza.
- La scelta dell'integrato per la comunicazione seriale.

2.1.1 Microchip dsPic30F4011

Per quanto riguarda il microcontrollore la scelta è caduta su un dsPic della Microchip.

Caratteristiche	Quantità
Canali Digitali	30
Canali Analogici	9
Canali Pwm	6
Canali Uart	3
Timer	5
Moduli I2C	1
Moduli CAN	1
Moduli SPI	1
Moduli QEI	1

Figura 2.2: Microchip dsPic30f4011.

Per l'esattezza è stato scelto il dsPic30F4011 con un architettura simile a quella Harvard e quindi il microcontrollore può accedere contemporaneamente alla memoria volatile, quella non volatile, all'ALU oltre che alle porte di I/O perchè comunica con essi tramite quattro bus separati. L'architettura oltretutto è di tipo RISC a 16 bit, ossia è presente un set di istruzioni ridotto, per cui la controller

unit è di tipo cablato in modo da permettere maggiori prestazioni.Inoltre l'istruction set è composto da 83 istruzioni.
I programmi si possono scrivere in assembly, ma viene fornito dalla stessa microChip un compilatore C ottimizzato che permette una programmazione più rapida e semplice.Sono presenti 48KByte di Flash Program Space, 2Kbyte di Ram e una EEPROM da 1Kbyte.
La frequenza di lavoro massima di 120Mhz ottenibile utilizzando un oscillatore esterno da 4 a 10Mhz e sfruttando lo stadio PLL interno che permette di ottenere un segnale di clcok 4,8 e 16 volte più veloce, che permettono portare il Dsp alla velocit di 30 MIPS. Altrimenti utilizzando sempre un clock esterno senza PLL si pu arrivare fino a 40 Mhz. Per finire c' anche la possibilità di usare un meno performante clock interno che pu arrivare fino a 7,37 Mhz. Questo integrato è disponibile in vari package, per comodità d'uso è stato scelto il package PDIP con 40 pin. Sono dedicati alle operazioni di I/O 30 pin. Le periferiche di I/O hanno le seguenti caratteristiche:

- 30 canali digitali.
- 9 canali analogici
- 6 canali PWM per Motor Control.
- 3 canali Uart.
- 5 timer a 16 bit o 2 a 32 bit.
- 1 Modulo I2C.
- 1 Modulo SPI.
- 1 Modulo CAN.
- 1 QEI (Quadrature Encoder Interface).

I canali digitali hanno uno schema di funzionamento come quello indicato in figura. Ad ogni porta sono associati tre registri.
Il Data Direction Register (TRISx) determina se la porta in questione è utilizzata come ingresso o come uscita, più precisamente un livello logico alto sta ad

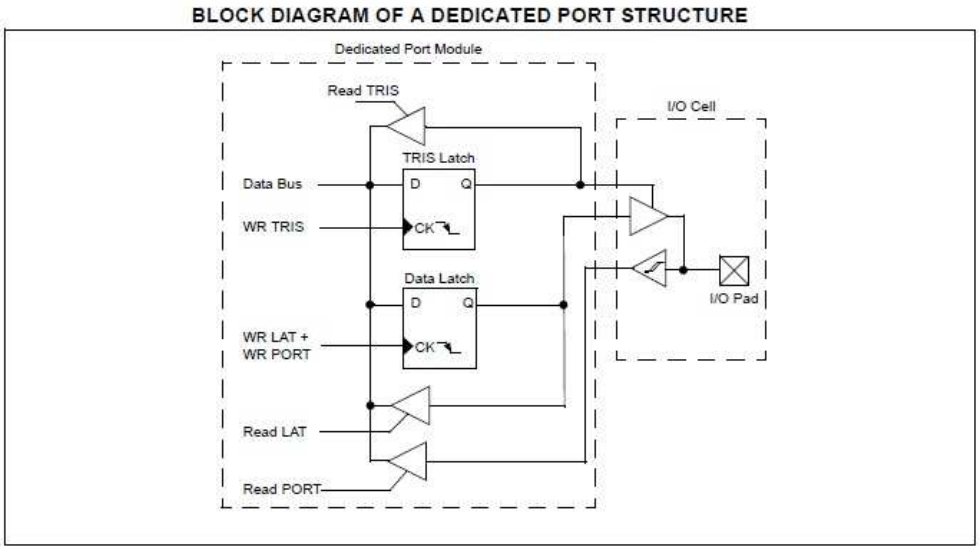

Figura 2.3: Diagramma a blocchi dei pin I/O digitale[2].

indicare input(1 viene infatti identificato come I di input), mentre 0 setta il pin come uscita(come prima 0 indica Output).Dopo un azione di reset tutte le porte sono vengono portate ad un livello logico alto.
Il LAT Register (LATx) viene utilizzato per imporre un livello logico su un pin.
Il PORT Register (PORTx) viene utilizzato per leggere il vero stato logico del pin. Per aumentare l'immunità ai rumori vengono utilizzati solo trigger di Schmitt.

Per quanto riguarda i 9 canali analogici il pic mette a disposizione un ADC a 10 bit con 4 Sample and Hold. Lo schema di funzionamento è quello mostrato in figura. Per un corretto funzionamento è necessario impostare sei registri. Sono presenti tre registri ADCONx che tra le tante funzionalità permettono di configurare il formato dei dati (che pu essere interno con segno, intero senza segno, frazionario con segno o frazionario senza segno), quali dei 4 canali utilizzare, quali sono i riferimenti di tensione che devono essere utilizzati per la conversione, che clock deve essere utilizato e a quanto devono essere impostati il prescaler e il postscaler per ridurre la frequenza del segnale di clock.
I registri ADCHS (ADC CHannel Seclect Register) e ADPCFG (ADC Port Con-

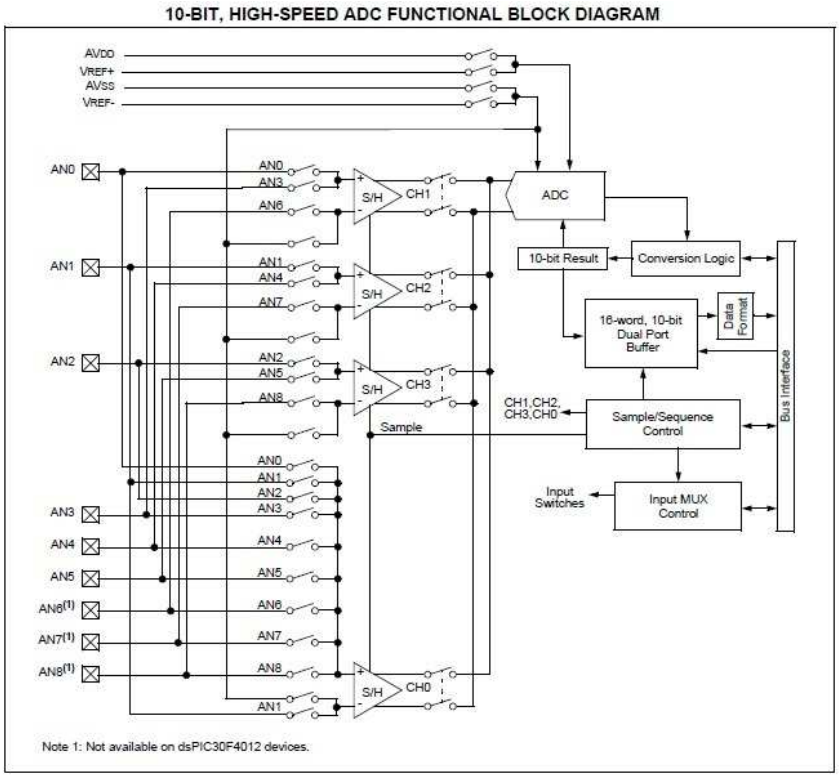

Figura 2.4: Diagramma a blocchi dell'adc[2].

FiGuration Register) permettono di determinare quali canali sono utilizzati come ingressi analogici e non digitali e permettono di scegliere su quale canale di lettura devono essere rediretti.
Per finire il registro ADCSSL (ADC Scan SeLect Register) permette di impostare un lettura sequenziale degli ingressi analogici scegliendo ovviamente quelli che sono gli ingressi da leggere.
I passi fondamentali per una corretta configurazione e lettura dei dati sono i seguenti:

1. Configurare il modulo ADC:
    - Scegliere quali sono i pin con funzionamento analogico e i riferimenti di tensione.
    - Selezionare la frequenza del clock.
    - Attivare il modulo ADC.

2. Configurare gli interrupt (se necessario).

3. Iniziare il campionamento.

4. Attendere il tempo di acquisizione.

5. Iniziare le conversione.

6. Quando la conversione è completata attendere che venga settato ad uno il bit DONE del registro ADCON1

7. Leggere il dato convertito nel buffer.

I 6 canali PWM possono essere utilizzati a due a due visto che sono presenti solo 3 generatori di duty cycle. I canali collegati allo stesso generatore di duty cycle al più potranno essere complementari. Tra le tante caratteristiche è possibile cambiare la frequenza del PWM durante l'esecuzione di nostri programmi, sono disponibili le modalità Center o Edge-Align, è inoltre presente la modalità Singole Pulse Generator.

I 3 canali UART permettono una comunicazione Full-Duplex ad 8 o 9 bit, ossia è possibile contemporaneamente ricevere e trasmettere dati sul mezzo trasmissivo. Questo tipo di cotroller è di tipo NRZ(Non Return Zero) ed utilizza un bit per indicare l'inizio del pacchetto, 8 o 9 bit dati ed uno o due bit per indicare la fine del pacchetto. Inoltre è possibile impostare un bit di parità scegliendo se questo deve essere pari o dispari. La configurazione più utilizzata è la (8,N,1) dove si utilizzano 8 bit di dati, non è usato il bit di parità (no parity) ed è presente un solo bit di Stop, è comunque possibile modificare tale configurazione agendo sul registro UxMODE. L'abilitazione e la disabilitazione avviene configurando il bit UARTEN nel registro UxMODE.
Il funzionamento può essere descritto dal seguente schema a blocchi.

Il cuore del trasmettitore è il registro UxTSR (Uart Transmit Shift Register), questo registro riceve i dati da trasmettere da un buffer di tipo FIFO (per la precisione il registro UxTXREG nella quale i dati vengono scritti via software).

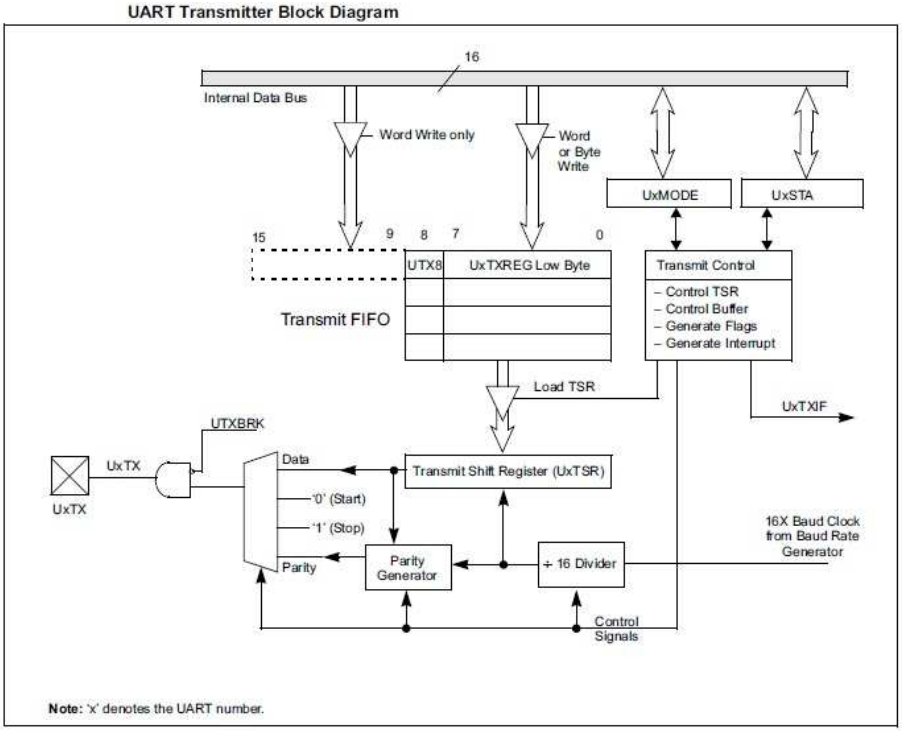

Figura 2.5: Diagramma a blocchi trasmettiore uart[2].

Il registro UxTSR non viene letto finchè non viene spedito l'ultimo bit di Stop, dopo di questo se è presente un dato da spedire viene correttamente caricato e trasmesso.
Viene definito in modo del tutto similare lo schema del ricevitore.

In modo del tutto speculare il cuore del ricevitore è il registro UxRSR(Recive Serial Shift Register), i dati vengono ricevuti dal relativo pin e inviati a tale registro. Una volta ricevuto anche il bit di Stop i dati vengono spostati nel registro FIFO dal quale è possible leggere byte o intere parole.

Per concludere la descrizione di questo microcontrollore parliamo velocemente delle altre interfacce che sono messe a disposizione. Il modulo I2C è un interfaccia seriale molto utilizzata per la comunicazione tra microcontrollori,memorie di tipo EEPROM, display e molti altri dispositivi ancora. Quest'interfaccia prevede un sistema di comunicazione bifilare di tipo master-slave, e la tipica realizzazione

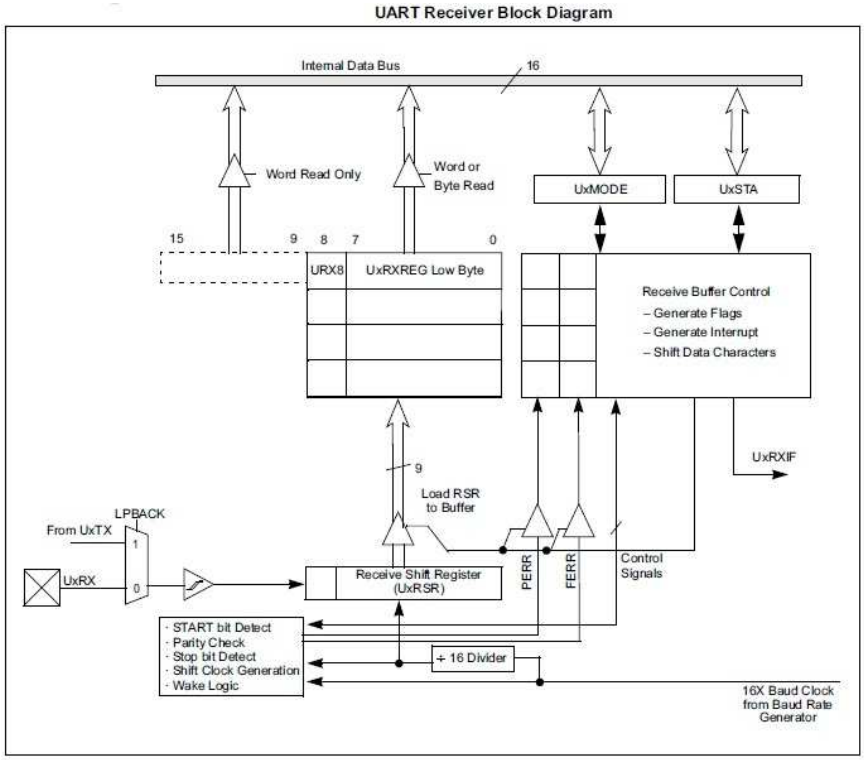

Figura 2.6: Diagramma a blocchi ricevitore uart[2].

prevede architetture con un singolo master e più slave(anche se non sono proibite architetture multimaster). I due canali sono rispettivamente chiamati SDA,SCL dove nel primo vengono effettivamente scambiati i dati mentre il secondo, visto che I2C sfrutta un bus sincrono,è il segnale di clock.

Il modulo SPI sempre un modulo per la comunicazione seriale, ma a differenza di I2C è di tipi full-dulpex anzichè half-duplex.La linea quindi non è bifilare, ma è composta da 4 collegamenti: SCLK è il segnale di clock emesso dal master,SDI(Serial Data Input) è l'ingresso per il master e l'uscita per lo slave,SDO(Serial Data Output)è l'uscita per il master e l'ingresso per lo slave e per finire CS(Chip Select)è il segnale emesso dal master per sceglie con quale slave comunicare.

Il modulo CAN è sempre un modulo di comunicazione seriale che viene molto utilizzato in ambienti rumorosi. L'implementazione su dsPic necessita di un modulo di Transceiver per poter leggere e scriver sul bus bifilare.

Per finire è implementato anche un modulo per la lettura di encoder in quadra-

tura, sono presenti i due ingressi per la lettura dei due segnali dell'encoder ed un segnale di index nel caso si desiderasse implementarlo. Il conteggio viene memorizzato su un contatore a 16bit oltretutto sono supportati anche le modalit di conteggio 2x e 4x.

2.1.2 National Semiconductor LMD18200

Per pilotare dei motori è fondamentale utilizzare un ponte H, così chiamato per la sua forma che ricorda la relativa lettera dell'alfabeto. Nel caso di un carico induttivo (tipico di un motore in continua) è possibile controllare agevolmente la potenza e la velocità del motore. A seconda del segno di tensione e corrente possiamo ottenere un accelerazione positiva o negativa del motore. Lo schema di funzionamento prevede l'impiego di quattro transistor che fungono da interruttori elettronici (lavrando in zona di saturazione o interdizione) che se correttamente azionati permettono di far fluire la corrente nei due versi, per evitare di azionare in modo scorretto i transistor sono inseriti degli inverter che non permettono di cortocircuitare i due mezzi ponti, evitandone quindi la rottura. Ricordando che un induttore tende a mantenere costante la corrente che lo attraversa, facendola variare con un tempo proporzionale alla sua costante di tempo di carica e scarica, e che un motore può essere rappresentato da un induttore in serie ad un resistore, si possono creare dei problemi. Quando il transistor si chiude la corrente che lo attraversa inizialmente è nulla ed impiegherà un certo lasso di tempo (sempre legato alla costante di tempo) per arrivare a regime. I problemi sorgono quando il transistor si apre, infatti la corrente non tende a calare in modo istantaneo, ma varierà solo in modo istantaneo la resistenza del transistor (che passa da uno stato di saturazione ad uno di interdizione). Avremo quindi sul collettore di tale componente un'altissima tensione che potrebbe provocarne la rottura. Per evitare questa tensione detta di fly-back si utilizza un diodo posto in parallelo all'avvolgimento del motore che crea così una via di fuga per la corrente ed evita le sovratensioni sul collettore del transistor. Tali diodi devono essere molto veloci nel passare dallo stato di interdizione a quello di conduzione e viceversa e devono essere in grado di sopportare grandi correnti (si può parlare anche di molti ampere).

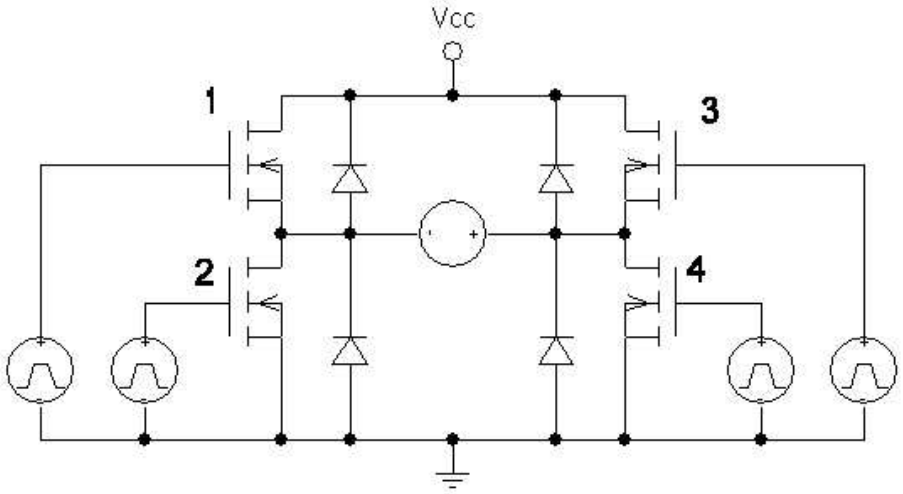

Figura 2.7: Schema di un ponte H.

Lo stadio in potenza da noi scelto è l'integrato LMD18200 della National Semiconductor. Tale integrato permette di raggiungere una corrente massima continua di 3A e una corrente massima di 6A, con una tensione di alimentazione massima di 55V. Sono presenti le funzionalità di thermal shutdown nel caso la temperatura superasse i 170°C, e di thermal flag portando a livello logico alto il relativo piedino quando la temperatura supera i 140°C, oltre ad un uscita per la lettura della corrente. Gli ingressi sono tre, quello denominato PWM accetta all'ingresso l'analogo segnale per pilotare il motore, il piedino DIR (Direction) determina il verso in cui fluisce la corrente e il piedino BRAKE invece permette di frenare il motore elettronicamente.
Lo schema logico il seguente:

Il pilotaggio del driver avviene con un segnale PWM, esistono però due tecniche per pilotare il ponte:

- PWM SEGNO E AMPIEZZA: Al piedino PWM viene applicato il segnale modulato mentre al piedino DIR viene applicato un segnale che permette di determinare la direzione della corrente negli avvolgimenti. La variazione del

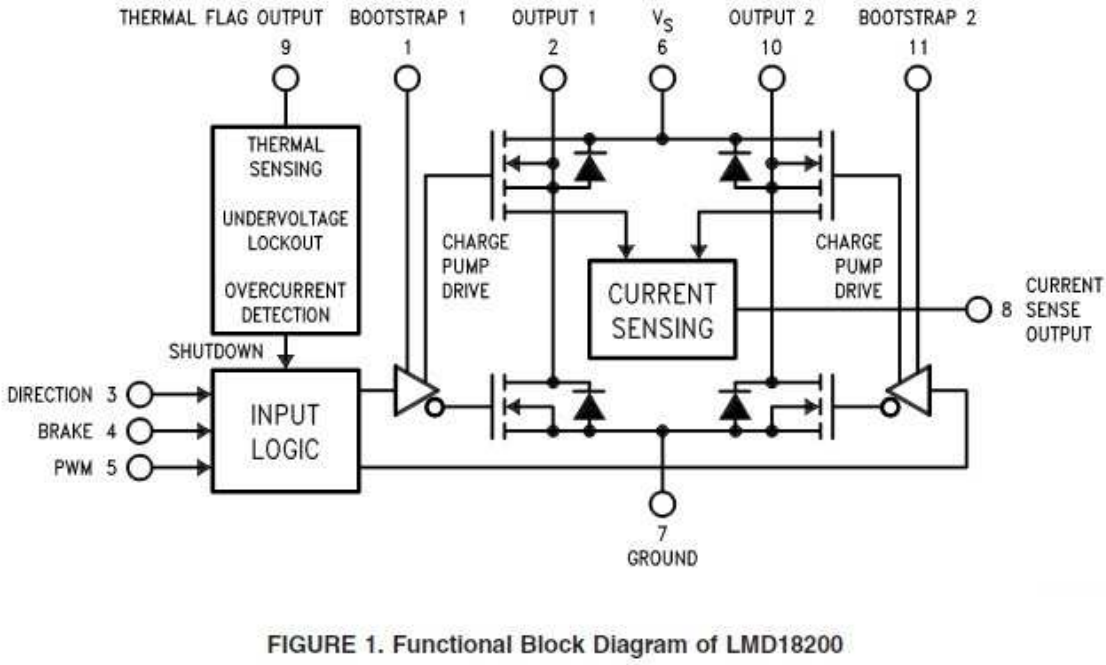

Figura 2.8: Schema logico dell'LMD18200.[3]

duty cycle permette quindi di variare la tensione media applicata al motore e di conseguenze la sua velocità.

- LOCKED ANTI-PHASE PWM: Il segnale modulato viene mandato al piedino DIR, mentre al piedino PWM viene mandato un segnale di enable. Quando il duty cycle è del 50% il motore è fermo, mentre un duty cycle del 100% e 0% permettono rispettivamente di far ruotare il motore al massimo delle sue potenzialità in un verso o nell'altro.

L'integrato scelto è disponibile in due package differente, noi abbiamo scelto il TO-220 ad 11 pin.

2.1.3 Maxim MAX 232

Quest'ultimo integrato permette di convertire un segnale TTL in un segnale basato sullo standard EIA RS-232 (utilizzato per la comunicazione seriale a bassa velocità tra due dispositivi digitali). Il segnale EIA RS-232 lavora con una logica

inversa rispetto a quella TTL, le tensioni di che possono essere utilizzate vanno dai $\pm 5V$ ai $\pm 15V$, anche se il riferimento maggiormente utilizzato di $\pm 12V$. In figura vediamo un esempio di tale segnale:

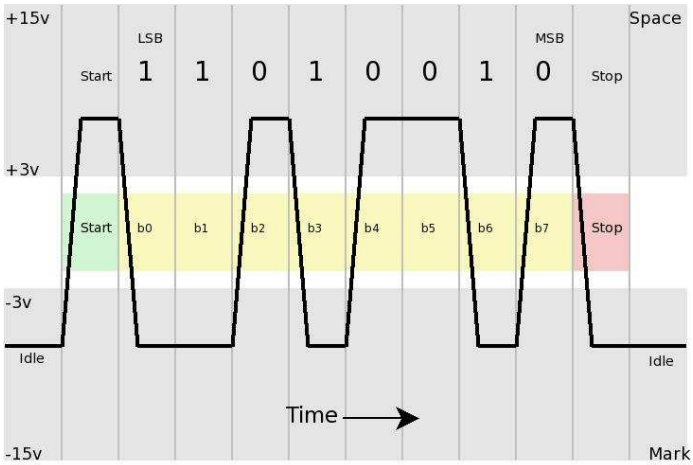

Figura 2.9: Logica di funzionamento del codice EIA RS-232.

L'integrato scelto disponibile in un package PDIP con 18 pin.

2.2 Progettazione dello schema del microcontrollore

Lo schema del microcontrollore qui sotto raffigurato è quello utilizzato per il nostro driver. Analizziamo nel dettaglio per capirne al meglio il funzionamento.
Il piedino uno, chiamato MCLR (Master CLear Reset) è un ingresso attivo basso che permette il reset del pic portandolo a massa. Lo schema iniziale prevedeva solamente un pulsante in grado di mettere a massa il piedino ed un resistore di pull-up per mantenere ad un livello logico alto il pin quando il pulsante non è premuto. Questa soluzione per può essere ampiamente migliorata prevedendo un circuito di anti-rimbalzo come quello presente in figura, dove l'accoppiamento RC permette di eliminare quelle che sono le grandi oscillazioni dovute all'intermittenza nella pressione tasto. Inoltre spesso è introdotto un diodo in parallelo al condensatore con il catodo posto a Vdd, che permette al condensatore di scaricarsi sul diodo una volta che la tensione di alimentazione è stata tolta. Il diodo da

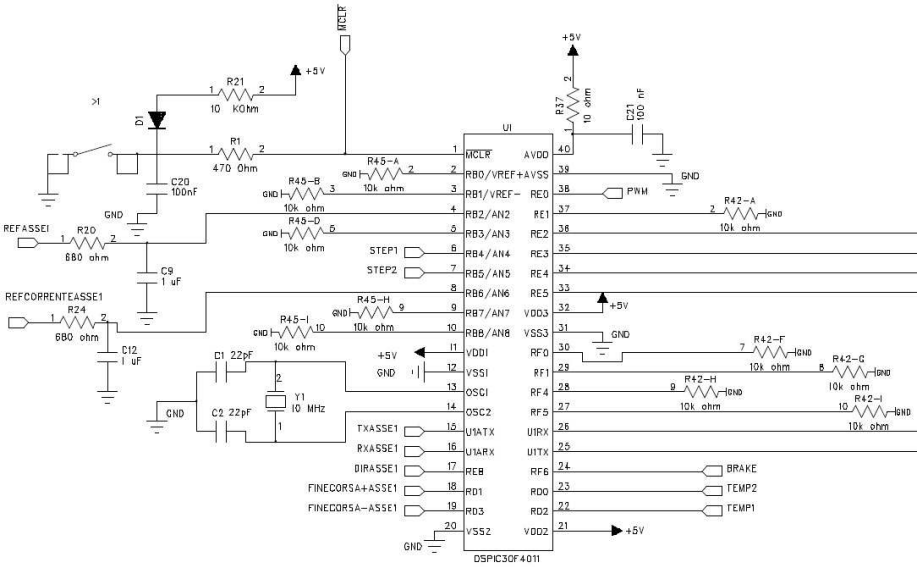

Figura 2.10: Schema microcontrollore.

noi inserito invece permette di proteggere l'alimentazione dalla tensione di 13.5V che viene applicata al piedino MCLR in fase di programmazione. Questo segnale di reset viene utilizzato anche dall'altra scheda driver, che si occupa di pilotare i motori in continua, sempre per resettare i due microcontrollori presenti.
Ai piedini 13 e 14 inveceè collegato un circuito che permette di generare il clock, nel datasheet del dsPic[2] questa configurazione è denominata come la seconda versione, ossia si utilizza come oscillatore un cristallo al quarzo con frequenza compresa dai 4 ai 10Mhz, la quale poi può essere moltiplicata per un fattore 4,8 o 16 attraverso uno stadio PLL. Appena il sistema viene alimentato il circuito risonante inizia ad oscillare come nella curva in figura.
Il tempo di start-up dipende da moltissimi fattori tra i quali ci sono:

- La qualità del quarzo utilizzato.
- La frequenza del quarzo.
- Il valore dei due condensatori C1 e C2.
- Il tempo di salita della tensione di alimentazione.
- il rumore.

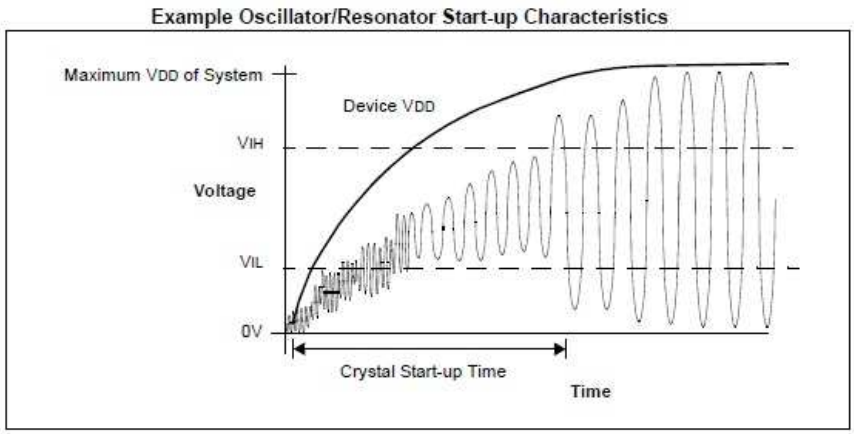

Figura 2.11: Caratteristica di avvio di un oscillatore[2].

Rimane ora da capire come scegliere il valore di C1 e C2. Il datasheet consiglia valori compresi tra 15pF e 33pF, maggiore è il valore di tali condensatori maggiora sarà il settiling time, anche se d'altra parte avremo un sistema più stabile, viceversa invece il sistema sarà più instabile, ma chiaramente più veloce. Abbiamo optato per un valore centrale pari a 22 pF perchè permette di avere il miglior compromesso tra velocità e stabilità.
Le altre configurazioni disponibili sono la LP, HS e la RC. LP significa Low Power ed in questa configurazione avremo il minor consumo in assoluto di corrente, d'altro canto però non avremo mai prestazioni molto elevate. La configurazione HS invece permette di arrivare ad altissime frequenza a discapito dei consumi, e l'acronimo infatti significa High Speed. Per finire RC utilizza un condensatore e una resistenza esterni per generare il clock. Questi microcontrollori mettono a disposizione anche degli oscillatori interni che però sono molto lenti e poco precisi, ma sono utili per applicazioni miniaturizzate ed a bassissimi consumi.
Ai piedini 39 e 40 sono collegati i riferimenti di tensione per il convertitore analogico digitale. I +5V di alimentazione sono accoppiati all'ingresso con un filtro passa basso che taglia alla frequenza di circa 16 Khz, utilizzato per ridurre il rumore.
Ai piedini 11,12,20,21,31 e 33 invece applicata l'alimentazione che viene prelevata dal circuito di alimentazione presente sulla scheda d'interfaccia, e che arriva

su tale scheda tramite un connettore di comunicazione che prenderemo in esame successivamente.
Ai piedini 4 ed 8, che sono rispettivamente AN2 ed AN6, sono applicati due segnali analogici. Il primo riceve in ingresso il numero di passi da far compiere al motore passo, mentre il secondo invece è stato introdotto nel caso fosse necessario utilizzare un altro canale analogico che potrebbe servire, per esempio, a leggere un monitor di corrente. Per avere una maggiore pulizia del segnale anche questi ingressi sono stati accoppiati con un filtro passa basso che taglia alla frequenza di 234 Hz.
Gli ingressi e uscite digitali invece hanno il seguenti scopi:

- Pin 6,RB4: Uscita utilizzate per far effettuare uno step al motore passo.
- Pin 7,RB5: Uscita utilizzate per far effettuare uno step al motore passo.
- Pin 17,RE8: Questo ingresso indica la direzione del movimento. Questa soluzione è stata implementata per evitare di dover usare un alimentazione duale per avere delle tensioni negative.
- Pin 22,RD2: Ingresso utilizzato del pic per leggere lo stato della temperatura di uno dei due avvolgimenti del motore.
- Pin 23,RD0: Ingresso utilizzato del pic per leggere lo stato della temperatura di uno dei due avvolgimenti del motore.
- Pin 24,RF6: Ingresso/Uscita utilizzato del pic per accedere al piedino di brake dell'lmd e per poterne leggere lo stato.
- Pin 38,PWM1H: Uscita utilizzata per generare l'onda quadra del PWM.

Per quanto riguarda gli ingressi/uscite U1ATX e U1ARX, sono collegati al circuito per la comunicazione tramite porta seriale. Alcuni ingressi sono stati collegati ad un connettore che permettere la futura espandibilità della scheda, lo schema di tale connettore è il seguente.
Sono stati previsti dei jumper che permettono di collegare i pin ai resistori di pull down in parallelo ad un uscita, oppure di collegare direttamente il pin ad una periferica esterna, settando il jumper nell'altra configurazione, nel caso il resistore

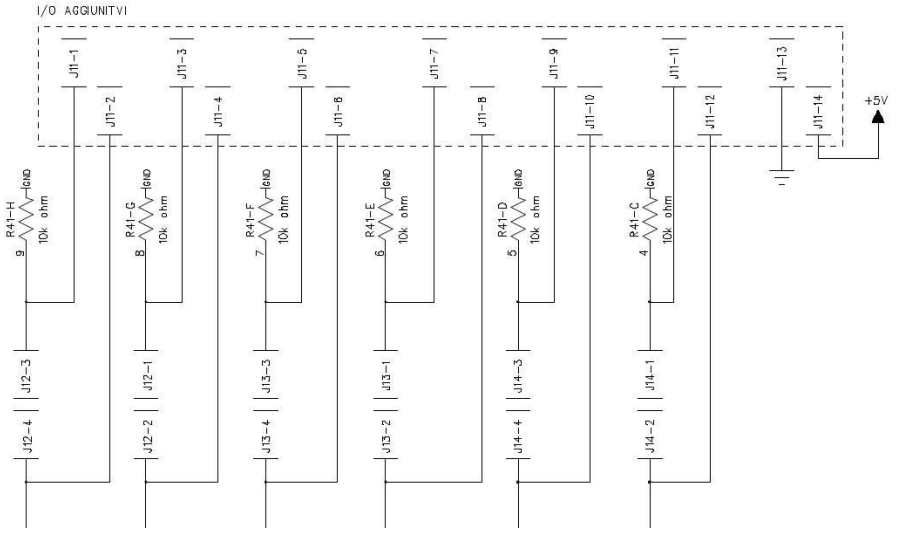

Figura 2.12: Ingressi e uscite aggiuntive per il pic.

di pull down potesse creare qualche problema in uscita. Per facilitare l'espandibilità della scheda in questo connettore è stata inserita l'alimentazione che quindi può essere direttamente prelevata. Per finire i rimanenti pin sono collegati a dei resistori di pull down.

2.3 Progettazione dello Stadio in Potenza

Lo stadio in potenza come già detto si basa su un integrato della National Semiconductor, l'LMD18200. Tale integrato permette di applicare una tensione al motore da 12V a 55V, con una corrente di picco di 6A ed una corrente massima continua di 3A.

Abbiamo preferito utilizzare il metodo sign and magnitude per pilotare il motore, dove bisogna applicare un onda quadra modulandone il duty cycle per determinare la tensione da applicare al motore, mentre la direzione della corrente viene decisa tramite un segnale digitale aggiuntivo. Questa tecnica permette di avere una risoluzione maggiore rispetto al metodo locked anti-phase e quindi ci dà un maggiore controllo sul motore stesso. Il driver che andiamo a progettare è di tipo

bipolare, per cui potranno essere utilizzati motori stepper bipolari oppure motori unipolari a cinque o sei fili, non utilizzando i fili centrali dei due avvolgimenti. Essendo quindi presenti due avvolgimenti separtati sono necessari due integrati per pilotare un motore passo, ognuno dei quali dovrà essere correttamente pilotato per ottenere la corretta sequenza di passi. Per rendere operativo tale stadio in potenza sono necessari pochissimi componenti esterni. Nel nostro caso bastano soltanto due condensatori utilizzati come charge-pump che vengono utilizzata per pilotare il gate dei dmos presenti nell'integrato. Il valore consigliato è di 10nF. Per rendere più pulito il segnale di alimentazione è possibile inserire un condensatore a monte del piedino V_s per filtrare così il rumore. Le due uscite devono essere collegate ad un avvolgimento del motore. Per finire rimangono cinque pin che permettono la gestione della logica dell'lmd18200 che sono:

- PWM: Tramite questo piedino possiamo applicare un onda quadra modulata nel duty cycle, che determina quindi la tensione applicata al motore.

- DIRECTION: Questo piedino determina come viene applicata la polarità agli avvolgimenti. Cambiando quindi tale valore si inverte il flusso della corrente che scorre nell'avvolgimento del motore.

- BRAKE: Settando tale pin ad un livello logico alto si può ottenere una frenata del motore con due modalità diverse a seconda del valore del pin PWM. Se quest'ultimo è ad un livello logico alto gli avvolgimenti del motore vengono internamente cortocircuitati permettendo quindi una migliore frenata sfruttando la corrente presente. Nel caso invece fosse presente al piedino PWM un livello logico basso allora l'avvolgimento viene aperto e il motore continuerà la sua corsa per inerzia.

- CURRENT SENSING: Tale piedino permette di leggere il valore della corrente in circolo nell'avvolgimento. Una ampere circolante corrisponde a 377μA in uscita dal monitor di corrente. Quindi tramite una resistenza è possibile leggere una tensione proporzionale alla corrente stessa.

- TEMPERATURE ALARM: Anche questo piedino dà informazioni in sola lettura e permette di capire quando l'integrato supera la temperatura critica

di 140°C. La sua logica di funzionamento è inversa e di conseguenza si rileva l'allarme quando è presente un livello logico basso. C'è comunque da dire che quando l'integrato supera i 170°C questo si spegne in modo del tutto automatico per preservare la vita di tutte le componenti collegate.

Per facilitare le operazioni di test del circuito è risultato molto comodo inserire dei diodi led collegati ad ognuna delle porte digitale dell'lmd18200. Come detto precedentemente è necessario stare attenti alla logica inversa del temperature alarm, che avendo un logica inversa, richiede di collegare il diodo in modo diverso dagli altri. Sono stati usati in totale quattro diodi led verdi per segnalare le variazioni dei segnali di direction e pwm, mentre sono stati usati quattro diodi led rossi per i segnali di brake e temperature alarm.
Il circuito implementato è il seguente:

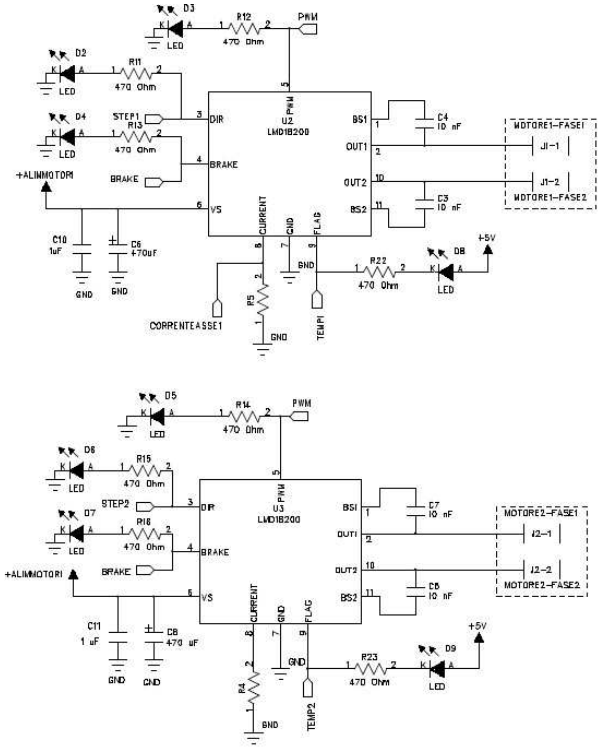

Figura 2.13: Schema utilizzato per lo stadio in potenza.

Il piedino di current sensing come precedentemente accennato necessita di una resistenza per leggere in modo corretto il valore del monitor di corrente. Dal datasheet il comportamento risulta essere lineare se in uscita si ottengono al massimo

5V. Tale valore, tra l'altro, risulta essere congeniale visto che l'adc del pic ha come tensioni di riferimento 0-5V. L'uscita di tale piedino inoltre è sempre positiva, quindi dal monitor di corrente otterremo il modulo del valore della corrente. Il calcolo della resistenza è quanto mai semplice. Sapendo che $V = Ri$ e sapendo che la corrente massima che potremmo leggere dal monitor è di $6A * 377\frac{\mu A}{A}$ è semplice ricavarsi R.

$$R = \frac{V}{I} = \frac{5}{377 * 6 * 10^{-}6} = 2210\Omega \tag{2.1}$$

A questo punto è necessario utilizzare una resistenza di precisione per evitare letture poco precise.
Per concludere sono stati usati dei connettori a morsetto per collegarsi al motore. Come si può vedere dallo schema verrà applicato la stessa tensione ad entrambi gli avvolgimenti, mentre il flusso della corrente è regolabile distintamente tramite i comandi di step che devono essere gestiti in modo opportuno dal pic.
L'alimentazione del motore viene direttamente collegata sulla scheda, ed è previsto l'inserimento di un fusibile tra i connettore e il pin di alimentazione dell'integrato. Il portafusibile utilizzato supporta fusibili di dimensione 5x20mm, con corrente massima nominale di 10A, tensione massima nominale di 250V c.a. e con grado di protezione IP40.

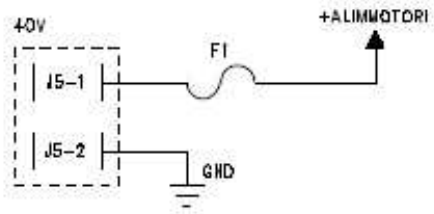

Figura 2.14: Alimentazione del motore.

2.4 Firmware basato su ciclo while

Per far funzionare correttamente il driver è necessario realizzare l'opportuno programma che andrà ad eseguire il microcontrollore. Tale programma può essere scritto essenzialmente in tre modi diversi:

- Tramite codice assembly, dove per necessario conoscere molto bene l'architettura del microcontrollore, inoltre il codice è poco portabile.
- Tramite codice C, utilizzando i vari compilatori a disposizione. Il codice così è molto più facile da scrivere. Di contro il codice assembly è sicuramente più efficiente a livello prestazionale.
- Tramite Simulink è possibile programmare tutti i dsPic della Microchip con dei semplici blocchi. Questo metodo è sicuramente il più semplice da utilizzare, ma d'altra parte nonè assicurata una buona efficienza.

Per il driver del motore passo ho optato per la seconda scelta.E´ stato per necessaria la scelta di un compilatore tra i tanti a disposizione:

- Ambiente di sviluppo MPLAB con compilatore Microchip. Il pacchetto è realizzato dalla stessa Mircochip ed è gratutito.
- Compilatore Hitech C PRO della htsoft. Software house che realizza compilatori per la Mircochip a pagamento. Era a disposizione una versione di prova funzionante per due mesi.
- Compilatore MircoC PRO for dsPic realizzato dalla MikroElektronika, azienda produttrice di scheda di prova per vari tipi di microcontrollori.

La scelta è caduta sul compilatore della MikroElektronika, che oltretutto al suo interno ha una vasta libreria di funzioni C per utilizzare in modo veramente molto semplice ogni funzionalità messa a disposizione dal dsPic.

Prima di tutto è interessante mostrare un semplice flow chart che mostra il funzionamento del programma.
Come si può vedere il programma dopo aver eseguito uno stadio di inizializzazione del generatore di pwm, dell'adc e dei port di ingresso ed uscita digitali entra all'interno di un ciclo infinito dove prima vengono letti il numero di passi da eseguire e successivamente vengono eseguiti con la relativa temporizzazione. Dopo di che il ciclo ricomincia.

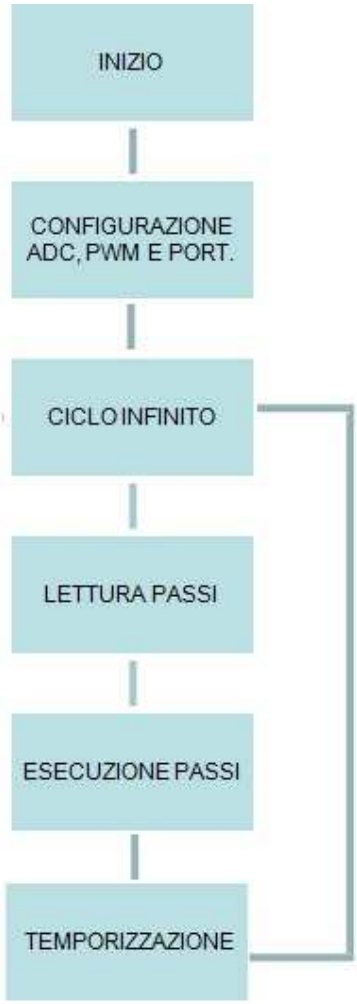

Figura 2.15: Flow chart firmware basato su ciclo while.

```
int step = 0; //Accumulatore degli step

//FUNZIONE PER ESEGUIRE UN STEP IN AVANTI
void avanti(){
   step = (step+1)\%8;
   switch (step)
  {
    //0 -
    case 0:
    Pwm_Mc_Set_Duty(2000,3);
    LATB = 0b0000000;
    Pwm_Mc_Set_Duty(1000,1);
    break;

    // 0 0
    case 1:
    Pwm_Mc_Set_Duty(1000,3);
    break;

    // - 0
    case 2:
    Pwm_Mc_Set_Duty(0,1);
    LATB = 0b0000000;
    Pwm_Mc_Set_Duty(1000,3);
    break;
    //1 0
    case 3:
    LATB = 0b0010000;
    Pwm_Mc_Set_Duty(1000,1);
    break;
```

```
    // 1 -
    case 4:
    Pwm_Mc_Set_Duty(2000,3);
    LATB = 0b0010000;
    Pwm_Mc_Set_Duty(1000,1);
    break;
    //1 1
    case 5:
    LATB = 0b0110000;
    Pwm_Mc_Set_Duty(1000,3);
    break;

    // - 1
    case 6:
    Pwm_Mc_Set_Duty(0,1);
    LATB = 0b0100000;
    Pwm_Mc_Set_Duty(1000,3);
    break;

    // 0 1
    case 7:
    Pwm_Mc_Set_Duty(1000,1);
    break;
  }
}
```

```
//FUNZIONE PER ESEGUIRE UN STEP INDIETRO
void indietro(){
   step = step-1;
   if(step == -1)
     step = 7;
   switch (step)
  {
    case 0:
    Pwm_Mc_Set_Duty(1000,1);
    Pwm_Mc_Set_Duty(2000,3);
    LATB = 0b0000000;
    break;

    // 0 0
    case 1:
    Pwm_Mc_Set_Duty(1000,1);
    Pwm_Mc_Set_Duty(1000,3);
    LATB = 0b0000000;
    break;

    // - 0
    case 2:
    Pwm_Mc_Set_Duty(0,1);
    Pwm_Mc_Set_Duty(1000,3);
    LATB = 0b0000000;
    break;

    //1 0
    case 3:
    Pwm_Mc_Set_Duty(1000,1);
    Pwm_Mc_Set_Duty(1000,3);
    LATB = 0b0010000;
    break;
```

```
1
    // 1 -
3   case 4:
    Pwm_Mc_Set_Duty(1000,1);
5   Pwm_Mc_Set_Duty(2000,3);
    LATB = 0b0010000;
7   break;

9   //1 1
    case 5:
11  Pwm_Mc_Set_Duty(1000,1);
    Pwm_Mc_Set_Duty(1000,3);
13  LATB = 0b0110000;
    break;
15
    // - 1
17  case 6:
    Pwm_Mc_Set_Duty(0,1);
19  Pwm_Mc_Set_Duty(1000,3);
    LATB = 0b0100000;
21  break;

23  // 0 1
    case 7:
25  Pwm_Mc_Set_Duty(1000,1);
    Pwm_Mc_Set_Duty(1000,3);
27  LATB = 0b0100000;
    break;
29  }
}
31 void main()
{
33
  int frequenza = 0;
35 int tempo;
  int conta = 0;
```

```
int resto;

//CONFIGURAZIONE PWM
Pwm_mc_Init(20000,1,0x05,0); //Perido pwm = 20Khz
Pwm_mc_Set_Duty(2000,1);
Pwm_mc_Set_Duty(0,3);
Pwm_mc_Start();

//CONFIGURAZIONE USCITE DIREZIONE
TRISBbits.TRISB4 = 0;
TRISBbits.TRISB5 = 0;
ADPCFGbits.PCFG4 = 1; // USCITA DIGITALE
ADPCFGbits.PCFG5 = 1; // USCITA DIGITALE

//CONFIGURAZIONE INGRESSI DIGITALI
TRISEbits.TRISE8 = 1; //INGRESSO DIREZIONE
TRISFbits.TRISF6 = 1; //INGRESSO BRAKE

//CONFIGURAZIONE INGRESSO ADC
TRISBbits.TRISB2 = 1;
ADPCFGbits.PCFG2 = 0;

//CICLO PRINCIPALE
while(1)
{
  //LETTURA DA ADC
   do{
      frequenza =(int)(Adc_Read(2)*10/1023);
   }while(frequenza == 0);
```

```
    //CALCOLO TEMPORIZZAZIONE
2   tempo = 1/frequenza;
    resto = 1,220 \% frequenza;
4
    //ESECUZIONE DEI PASSI
6   while(conta<frequenza)
    {
8     if(PORTEbits.RE8)
      {
10          avanti();
      }
12    else
      {
14        indietro();
      }
16    delay_cyc(tempo,resto);//TEMPORIZZAZIONE
      conta++;
18  }
    conta = 0;
20 }
}
```

Analizziamo ora il programma nelle sue parti pi importanti.
La struttura TRISXbits, dove al posto della X inseriremo il nome della porta d'interesse, permette di configurare se un pin è attivo come un ingresso od un uscita digitale. L'uscita PWM può essere facilmente configurata tramite le tre funzioni C, presenti nella libreria della MikroElektronika. La prima funzione permette di impostare la frequenza di lavoro, nel nostro caso è di 20KHz,su quale porta si deve attivare il PWM (nel nostro caso 0x01 significa che è attiva solo la porta RE0) e se la frequenza del PWM deve essere modificata tramite un prescaler (nel nostro caso non necessario ed è stato messo il valore 0). L'intero che ritorna la funzione è il valore da impostare per avere un dutycycle del 50%, questo valore può essere utilizzato subito nella funzione successiva che per l'appunto permette di impostare il duty cycle. Per finire l'istruzione di start avvia il PWM sulla porta desiderata. La lettura da ADC avviene tramite al funzione Adc_Read(numero del canale da leggere) e ovviamente ritorna un valore tra 0 e 1023, essendo l'ADC a 10bit. La prima conversione è:
Numero di passi = Valore letto da ADC $*\frac{10}{1023}$

Sapendo che il massimo numero di passi che si possono effettuare in ciclo di clock sono 10. Ora prima di spiegare il funzionamento della successiva parte del programma necessario capire che segnali applicare allo stadio in potenza per far ruotare il motore passo. Dal datasheet dell'lmd ricaviamo delle informazioni molto utili, infatti è scritto che per effettuare una rotazione di un passo è necessario cambiare in modo opportuno il segnale di direction. Per capire meglio che segnale utilizzare analizziamo come esempio un semplice motore step da 4 passi giro:

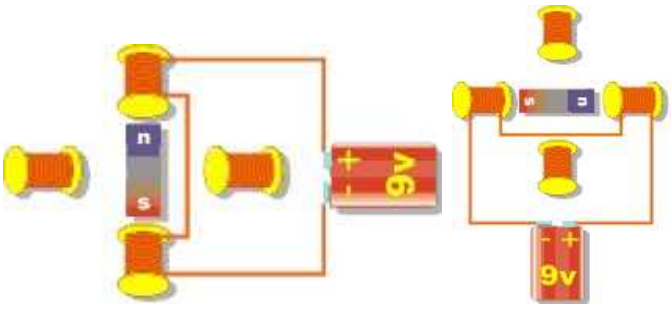

Figura 2.16: Primo e secondo step di un motore passo[4].

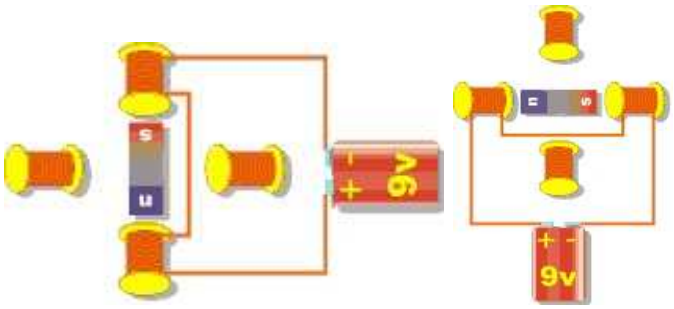

Figura 2.17: Terzo e quarto step di un motore passo[4].

Nella prima figura vengono alimentati i due elettromagneti verticali. Il solenoide in alto ha una polarizzazione SUD, mentre il solenoide in basso ha una polarizzazione NORD, di conseguenza la calamita al centro(che è il nostro rotore) si allinea agli elettromagneti mantenendo in alto il NORD ed in basso il SUD. Per ottenere lo spostamento di un passo basta a questo punto alimentare correttamente i due elettromagneti orizzontali, polarizzando quello a destra come SUD e quello a sinistra come NORD. Il motore compierà una rotazione di 90° in senso orario. Per ottenere un ulteriore passo dovremmo polarizzare di nuovo gli elettromagneti verticali, ma con una polarizzazione inversa rispetto a quella iniziale. Il passo successivo si ottiene similarmente polarizzando i magneti orizzontali in modo inverso rispetto al secondo step e per finire polarizzando come all'inizio quelli verticali otterremo una rotazione completa.

Per ottenere tale rotazione è quindi necessario applicare l'opportuno codice binario ai comandi direction degli lmd,nel nostro caso è proprio un codice Gray a due bit. Per invertire il verso di rotazione basta semplicemente invertire il verso del conteggio.

Il codice C infatti una volta ricevuto il numero di passi da effettuare entra in ciclo while che termina solo quando i passi sono esauriti, tramite l'ingresso applicato al pic che determina la direzione di rotazione si può determinare se il conteggio deve essere di tipo up o down.

Per evitare possibili impuntamenti del carrello su cui poggia la base del manipolatore è fondamentale distribuire in modo uniforme il numero di passi da eseguire lungo un ciclo di clock del controllo, cioè in 1 ms. Questo problema si può risolvere introducendo un ritardo inversamente proporzionale al numero di passi

che si vogliono effettuare, la funzione delay_cyc svolge proprio questo compito. In ingresso tale funzione necessita del numero di cicli di clock durante il quale il microcontrollore deve rimanere in attesa, visto che il valore letto è un intero a 16 bit,la funzione vuole in ingresso il numero di cicli diviso 16384 per aumentare le potenzialità del delay. Per considerare il possibile resto è presente una secondo parametro sempre intero.

2.5 Firmware basato su ISR

Il software precedentemente sviluppato non garantisce la lettura esattamente allo scadere del ciclo di clock, ossia ogni millisecondo, per ovviare il rischio di perdere la lettura di alcuni passi è necessario cambiare tipo di approccio per la realizzazione del programma.
Per essere sicuri di leggere l'ADC dopo un tempo prestabilito è necessario utilizzare gli interrupt. Il dsPic permette di gestire in modo molto semplice ed efficace gli interrupt, sono impostabili fino a 7 livelli di proprità dell'interruzione, inoltre Interrupt Vector Table (IVT) può contenere fino a 62 vettori d'interruzione.
Nel nostro caso l'interruzione verrà sollevata da un contatore che finito il suo conteggio bloccherà l'esecuzione del programma principale e permetterà di eseguire il codice d'interesse. I passi da seguire sono i seguenti:

- Impostare tutti i registri al valore di corretto funzionamento.
- Impostare la priorità massima al contatore scelte per il conteggio.
- Abilitare l'interruzione per il timer.
- Inizializzare il timer al valore di countdown.
- Avviare il timer.

Fatte queste operazioni si può entrare nel ciclo principale del programma che verrà bloccato ad intervalli di tempo regolari. Una volta all'interno della Interrupt Service Routine (ISR) è fondamentale eseguire le seguenti operazioni:

- Azzerare il flag che indica che si è sollevata un eccezione.

- Se necessario far ripartire il conteggio.

Molto importante è non realizzare un ISR con del codice computazionalmente molto pesante da eseguire altrimenti si potrebbe correre il rischio di generare un altra interruzione legata sempre allo stesso contatore e di non aver ancora finito di eseguire la routine per l'interruzione precedente.
Analizziamo ora il codice del programma per il motore passo.
La prima parte come per il precedente programma è solo una semplice inizializzazione dei registri, dove si decide quale ingresso deve essere impostato come ingresso e quale come uscita.
Il pwm viene inizializzato ad una frequenza di 20Khz con un duty cycle iniziale del 50%.
Una volta configurato il timer e le interruzioni è fondamentale sincronizzarsi con il controllo. Infatti il contatore verrà fatto partire solamente quando riceverà in ingresso una frequenza di passi diversa da zero, successivamente verrà eseguito il primo passo e dopo di che il software entra in ciclo infinito che rappresenta il nostro programma principale. All'interno di tale ciclo è possibile inserire tutte quelle operazioni che non hanno un importanza fondamentale per la gestione del sistema, come per esempio la trasmissione di dati a scopo puramente informativo tramite la porta seriale, in quanto l'esecuzione verrà interrotta più volte.
Scrivere l'ISR è molto semplice in quanto non necessita di prototipo, la firma della funzione è quella di una funzione normale dove deve essere aggiunta la direttiva *org indirizzo_IVT*. Nel nostro caso l'indirizzo all'interno dall Interrupt Vector Table è 0x1A.
Il codice all'interno della funzione viene eseguito ogni $10\mu s$, quindi è possibile eseguire fino a 100 passi in un ciclo di clock senza dover alterare il valore del contatore. In base alla frequenza di passi da eseguire la funzione calcola quando è necessario eseguire una commutazione delle uscite STEP 1 e 2, mentre quando ciò non è necessario viene semplicemente incrementata una variabile interna che permette di sapere quanto tempo è trascorso dalla reinizializzazione del contatore. Alla centesima esecuzione della ISR viene riletto il valore presente sull'ADC e aggiornate tutte le variabili atte al conteggio, inoltre viene anche letto il valore del duty cycle che è necessario impostare al PWM.

```
int conta=1; //Variabile globale per il conteggio del "tempo"
double frequenza=0; //Lettura della frequenza di passi dall'adc
int delay; //Delay tra un passo e l'altro in us
int delaytot; //Delay totale ottenuto sommando il valore di delay
int duty_cycle; //Duty cycle pwm

void main(){
  ////////////////////////////////////////////////////////////////////
  //CONFIGURAZIONE VARIABILI E REGISTRI PASSI E DIREZIONE
  ////////////////////////////////////////////////////////////////////
  int x; //Variabile di scambio
  TRISEbits.TRISE8 = 1; //Ingresso direzione
  TRISBbits.TRISB4 = 0; //Uscita step 1
  TRISBbits.TRISB5 = 0; //Uscita step 2
  ////////////////////////////////////////////////////////////////////
  //CONFIGURAZIONE PWM
  ////////////////////////////////////////////////////////////////////
  duty_cycle = Pwm_Mc_Init(20000,1,0x01,0); //Perido pwm = 20Khz
  Pwm_Mc_Set_Duty(duty_cycle,1);
  Pwm_Mc_Start();
  ////////////////////////////////////////////////////////////////////
  //CONFIGURAZIONE ADC
  ////////////////////////////////////////////////////////////////////
  LATBbits.LATB2 = 1; //Ingresso analogico per la frequenza di passi
  LATBbits.LATB6 = 1; //Ingresso analogico corrente per decidere duty cycle pwm
  ////////////////////////////////////////////////////////////////////
  //CONFIGURAZIONE TEMP CONTROL E BRAKE
  ////////////////////////////////////////////////////////////////////
  TRISDbits.TRISD0 = 1; //Ingresso tempo control 2
  TRISDbits.TRISD2 = 1; //Ingresso tempo control 1
  TRISFbits.TRISF6 = 1; //Ingresso comando brake
  ////////////////////////////////////////////////////////////////////
  //CONFIGURAZIONE TIMER E INTERRUZIONI
  ////////////////////////////////////////////////////////////////////
  IPC0 = IPC0 | 0x1000; // Priorità interruzione massima = 1
  IEC0 = IEC0 | 0x0008; // Abilitata interruzione timer
  PR1 = 200; // Valore da inserire nel contatore
```

```
1  ////////////////////////////////////////////////////////////////////
   //SINCRONIZZAZIONE CON IL CONTROLLO
3  ////////////////////////////////////////////////////////////////////
   while(frequenza==0)
5  {
       frequenza=Adc_Read(2)*5000/1024 ; //Ipotesi max 5000 passi/sec
7  }
   ////////////////////////////////////////////////////////////////////
9  //AVVIO TIMER - AVVIO PRIMO STEP
   ////////////////////////////////////////////////////////////////////
11 T1CON = 0x8000; // Abilitazione timer (prescaler a 1)
   delay =(int)(1/frequenza)*10000*10; // Calcolo del ritardo
13 delaytot = delay ; // Calcolo del tempo totale di attesa
   if(PORTEbits.RE8) //Scelta direzione avanti-indietro
15 {
       x = LATBbits.LATB4; //Eseguo un passo avanti
17     LATBbits.LATB4 = LATBbits.LATB5;
       LATBbits.LATB5 = ~x;
19 }
   else
21 {
       x = LATBbits.LATB5; //Eseguo un passo indietro
23     LATBbits.LATB5 = LATBbits.LATB4;
       LATBbits.LATB4 = ~x;
25 }

27 while(1); // Loop infinito
}
29

void Timer1Int() org 0x1A // ISR per Timer1
31 {
   int x; //variabile di scambio
33 IFS0 = IFS0 & 0xFFF7; // Reset del flag dell'interruzione
   conta++; //Incremento il conteggio
```

```
if(conta == 99) //se sono arrivato a cento conteggi leggo l'adc
{
  duty_cycle = ((int) Adc_Read(6))*(4000/1024); //Leggo il duty cycle
  Pwm_Mc_Set_Duty(duty_cycle,1); //Setto il duty cycle
  frequenza = Adc_Read(2)*5000/1024 ; //Ipotesi max 5000 passi/sec
  if(frequenza != 0) //se devo muovermi eseguo un psso
  {
      delay =(int)(1/frequenza)*10000*10; // Calcolo del ritardo
      delaytot =delay ; // Calcolo del tempo totale di attesa
      if(PORTEbits.RE8) //scelgo la direzione
      {
          x = LATBbits.LATB4; //Eseguo un passo avanti
          LATBbits.LATB4 = LATBbits.LATB5;
          LATBbits.LATB5 = ~x;
      }
      else
      {
          x = LATBbits.LATB5; //Eseguo un passo indietro
          LATBbits.LATB5 = LATBbits.LATB4;
          LATBbits.LATB4 = ~x;
      }
  }
  else
  {
      delaytot = 1000; //valore che sicuramente non sarà mai raggiunto
                       //e non effettueremo mai un passo entrando nell'if
                       //successivo.
  }
  conta=0;
}
if(conta >= delaytot)
{
  delaytot=delaytot+delay; // Calcolo del ritardo
```

```
    if(PORTEbits.RE8) // Calcolo del tempo totale di attesa
    {
        x = LATBbits.LATB4; //Eseguo un passo avanti
        LATBbits.LATB4 = LATBbits.LATB5;
        LATBbits.LATB5 = ~x;
    }
    else
    {
        x = LATBbits.LATB5; //Eseguo un passo indietro
        LATBbits.LATB5 = LATBbits.LATB4;
        LATBbits.LATB4 = ~x;
    }
  }
}
```

2.6 Progettazione dello schema per la comunicazione seriale

Per quanto riguarda la comunicazione seriale in questa scheda viene gestita la comunicazione tramite uart di tutti e tre i pic presenti nelle schede di controllo dei motori. L'idea è quella di poter selezionare quale pic far comunicare con un pc oppure con un altra scheda tramite un selettore a slitta e successivamente far avviare la comunicazione.

Visto che in futuro potrebbe essere necessario permettere la comunicazione tra i tre driver allora è previsto un connettore al quale arrivano tutti i segnali di uart dai vari pic. Per gestire correttamente la comunicazione sarebbe necessario implementare un semplice protocollo di comunicazione come quello che andremo adesso a descrivere.

La catena di comunicazione potrebbe essere effettuata in questo modo, dove il picC è il microcontrollore che controlla il motore passo, il picA e il picB invece

gestiscono i due motori in continua.

PicC 4011 tx → picA 2011 rx → picB 2011 rx
PicC 4011 rx → picA 2011 tx → picB 2011 tx

Con questa modalità il picC è definito come pic master e sarà esso stesso a comandare la comunicazione. In questo modo è possibile gestire una comunicazione interna oppure una comunicazione esterna con una qualsiasi altra periferica. Per realizzare tale tipologia di comunicazione è necessario però implementare via software il protocollo in modo corretto per determinare a chi spetta un determinato pacchetto ed evitare collisioni nel caso di scrittura contemporanea nel bus.
L'integrato scelto ed usato per gestire la comunicazione è il max232. Il suo funzionamento si basa semplicemente su delle pompe di carica per riuscire ad ottenere una tensione di 10V partendo da una tensione di 5V. I valori dei cosiddetti flying capacitor è determinato nel datasheet del componente. Nel nostro caso sono necessari 4 condensatori da 330nF ed uno da 47nF[5]. Per quanto riguarda la scelta della slitta è stato utilizzato un d3pt. Lo schema e il componente scelto sono quelli presenti nella seguente figura:

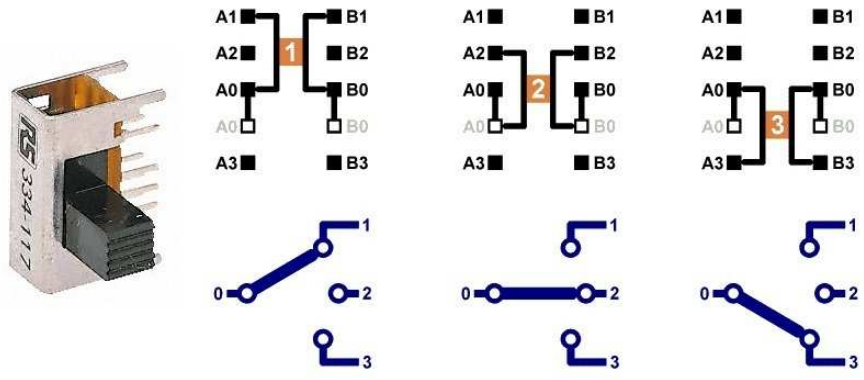

Figura 2.18: Schema di funzionamento del selettore.

Spostando la slitta si può ottenere quindi il doppio collegamento voluto del ricevitore e del trasmettitore.
Lo schema finale implementato è quindi il seguente:

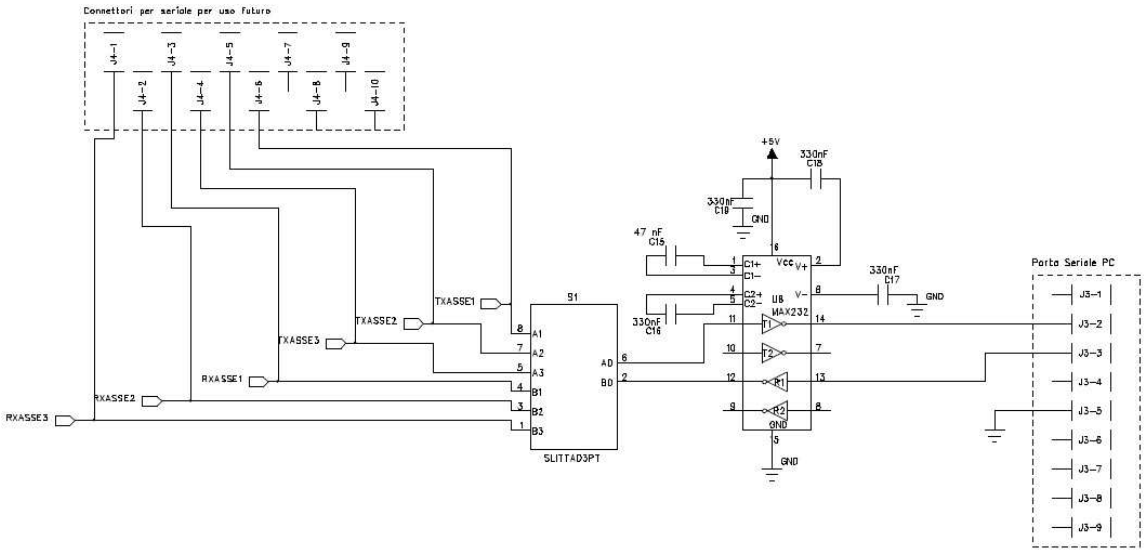

Figura 2.19: Schema circuitale del collegamento seriale.

Come si vede dalla figura le capacità per la pompa di carica sono collegate al max232 come consigliato dal datasheet della maxim[5], con i valori indicati nella documentazione relativa. Si può vedere come è predisposto il connettore per usare in futuro in modo diverso i piedini uart messi a disposizione. Per finire sul connettore seriale ci si collega ai piedini 2 e 3 per quanto riguarda la trasmissione e la ricezione dei dati, mentre al piedino 5 per quanto riguarda la massa.

2.7 End-Effector

Anche se non previsto nel progetto iniziale è necessario prevedere la possibilità di pilotare un end-effector tramite un elettrocalamita o similare. Il circuito elettronico che provvederà a pilotarlo è quello in figura.

Il circuito è basato su un semplice transistor npn che viene pilotato direttamente dalla base. Per quanto riguarda l'alimentazione si è preferito utilizzare i 15V forniti dall'alimentatore e non i 5V per avere minori limiti di tensione e per non avere limiti di corrente.

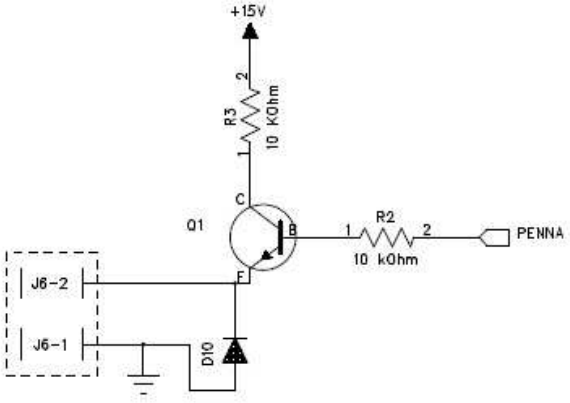

Figura 2.20: Schema circuitale per pilotare l'end-effector.

2.8 Connettore per la comunicazione

I segnali di comando che devono arrivare alla schede e quelli di informazione che partono dalla stessa transitano su un bus interno a 26 vie. Il connettore usato con i relativi collegamenti è quello in figura.

I collegamenti utilizzati sono i seguenti:

- REF ASSE1: Su questo canale viene spedita la frequenza di passi che deve eseguire il motore passo.
- CORRENTE ASSE1: Il valore della corrente letto dal monitor di corrente viene mandato al controllo sul secondo canale.
- REF CORRENTE ASSE1: Questo canale può essere utilizzato per implementare la lettura da parte del pic del monitor di corrente o può essere usata per determinare il duty cycle del pwm.
- GND: Massa dell'intero circuito.
- +15V: Tensione fornita direttamente dall'alimentatore.
- +5V: Tensione stabilizzata dai 15V tramite l'integrato LM7805.
- TEMP1A: Riferimento per il controllo della temperatura dello stadio in potenza.

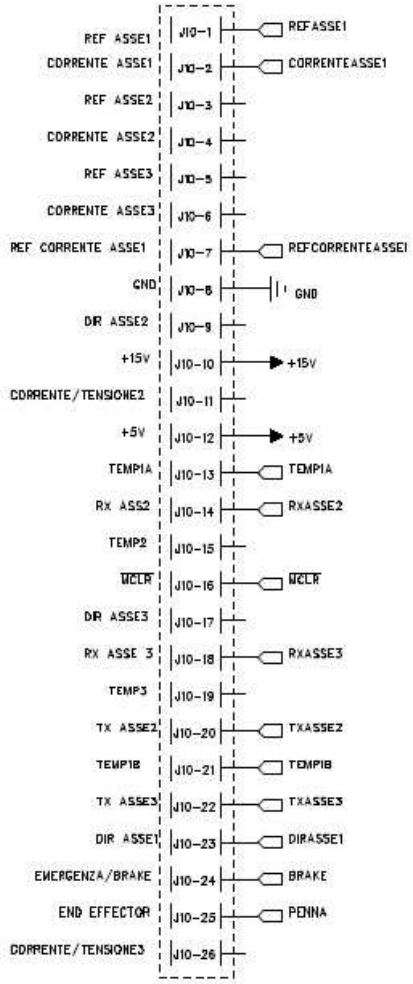

Figura 2.21: Schema del connettore.

- RXASSE2: Canale di ricezione del pic che gestisce il primo membro.
- MCLR: Master Clear dei pic. Per semplicità solamente sul pic che controlla il motore passo è stato implementato. Tale canale se cambia di livello procura il reset di tutti e tre i pic.
- RXASSE3: Canale di ricezione del pic che gestisce il secondo membro.
- TXASSE2: Canale di trasmissione del pic che gestisce il primo membro.
- TEMP1B: Riferimento per il controllo della temperatura dello stadio in potenza.

- TXASSE3: Canale di trasmissione del pic che gestisce il secondo membro.
- DIRASSE1: Riferimento che determina il verso di rotazione del motore.
- BRAKE: Canale di emergenza. Se alto permette mette in frenata i motori e manda il sistema in emergenza.
- END-EFFECTOR: Azione di comando dell'end effector.

2.9 Schemi completi del driver

Sono qui presenti gli schemi sia della parte logica che del layout del pcb.

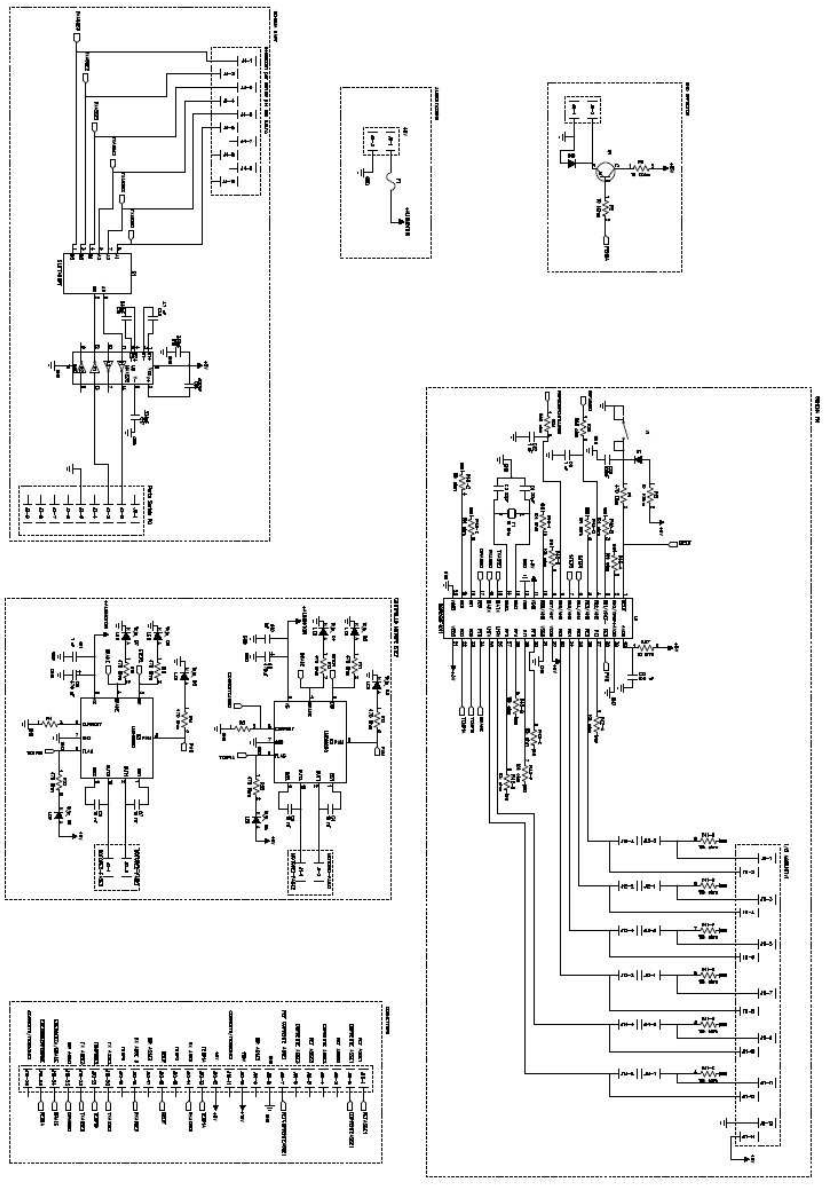

Figura 2.22: Schema del Driver.

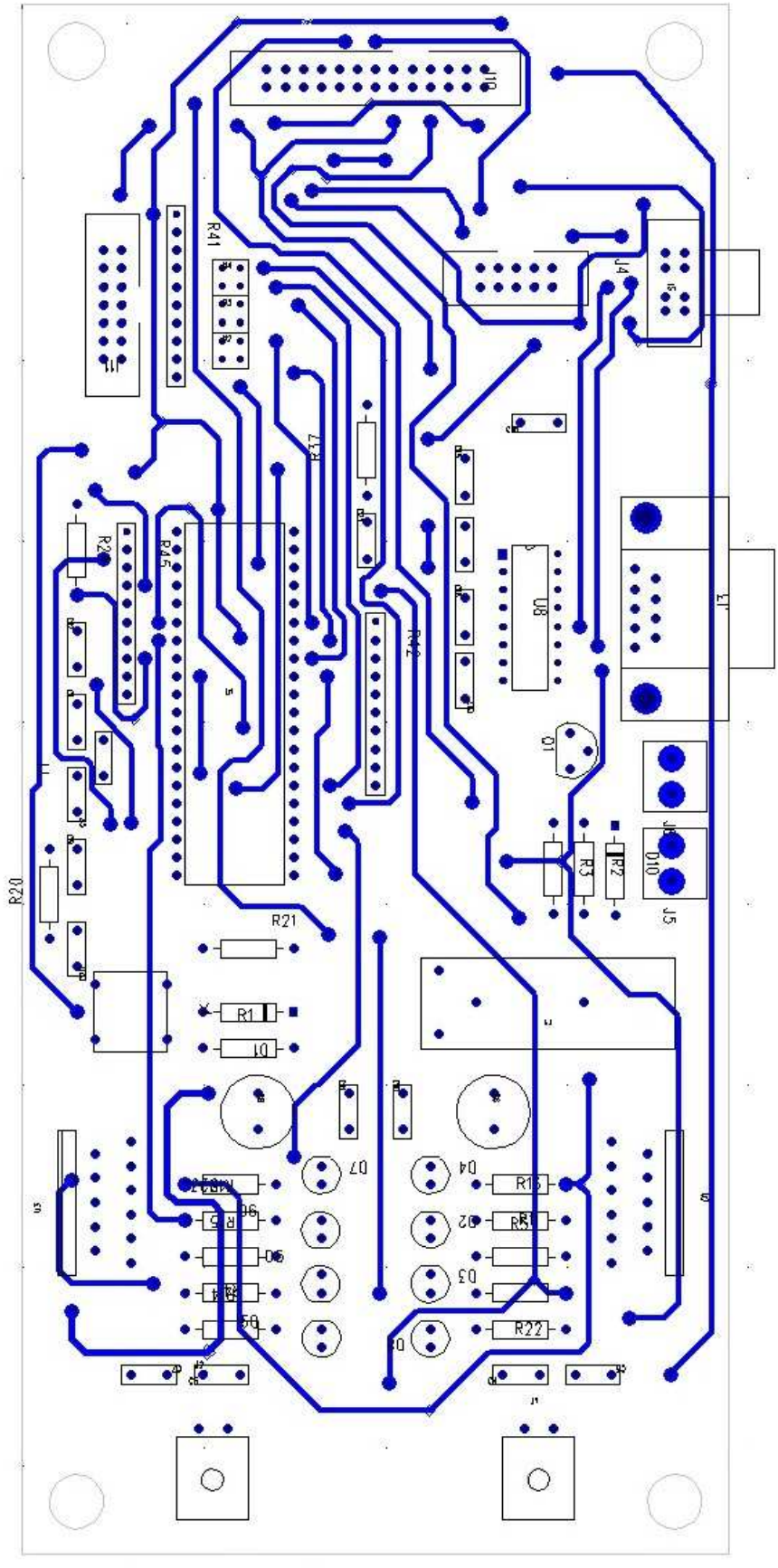

Figura 2.23: Schema pcb.Lato top.

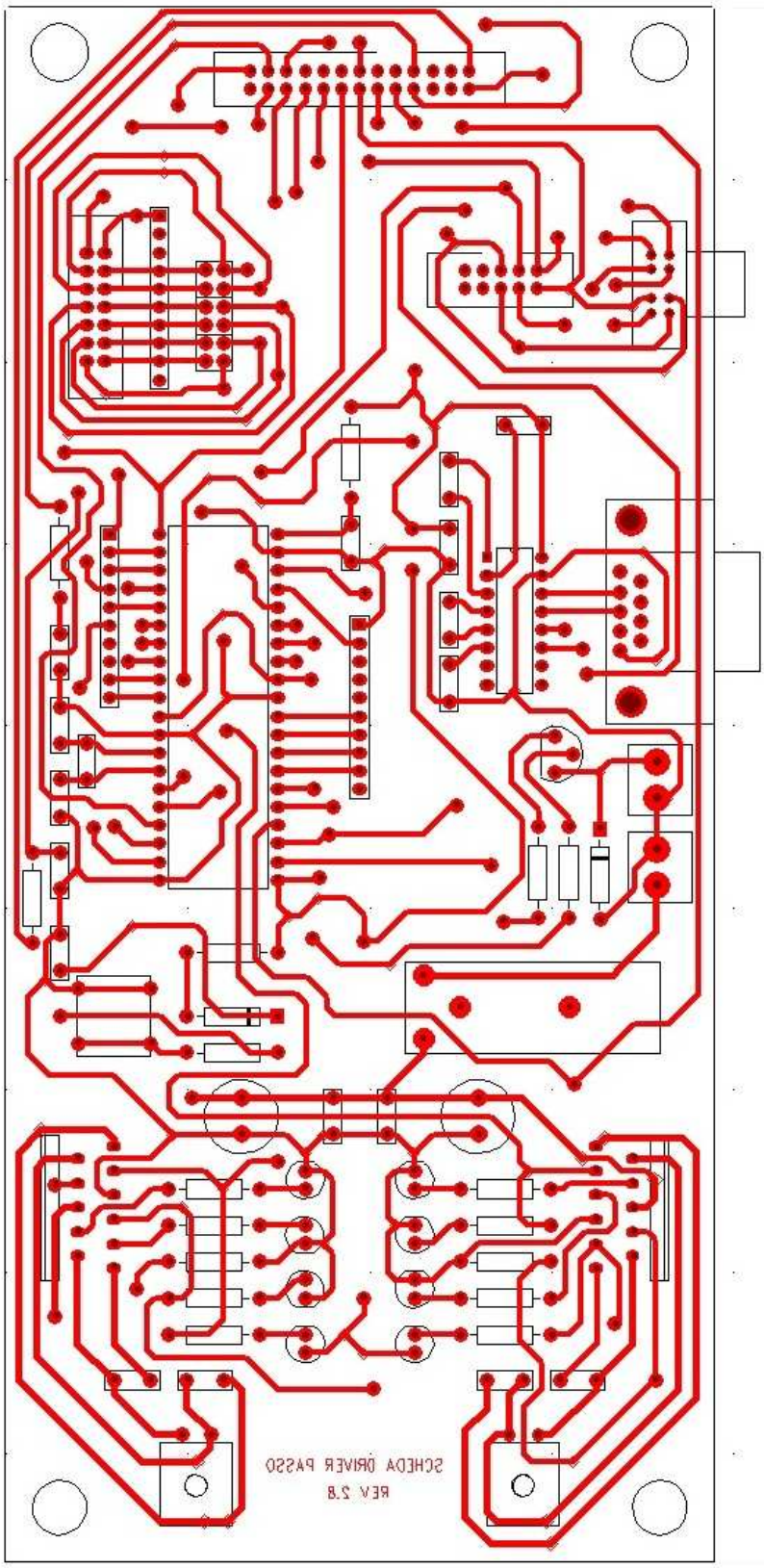

Figura 2.24: Schema pcb.Lato bottom.

Capitolo 3

Scheda d'interfaccia

Per realizzare un corretto interfacciamento dei driver con la scheda d'acquisizione sensoray 626 e gestire correttamente emergenze ad un livello superiore rispetto a quello dei driver è stato necessario introdurre una scheda d'interfaccia. I suoi compiti sono molto semplici, in quanto deve permettere il collegamento con la sensoray 626 sia tramite porte analogiche e digitali, che tramite gli encoder. Inoltre su questa scheda è stato implementato il circuito di alimentazione e stabilizzazione della tensione che arriva dagli alimentatori, oltre ai circuiti di gestione dei finecorsa induttivi e dell'emergenza.

Figura 3.1: Schema scheda d'interfaccia.

Prima di scegliere i componenti però è stato necessario capire com'è costruita l'uscita della scheda sensoray. Leggendo il datasheet di tale schede si può trovare

quest'immagine per quanto riguarda le uscite digitali:

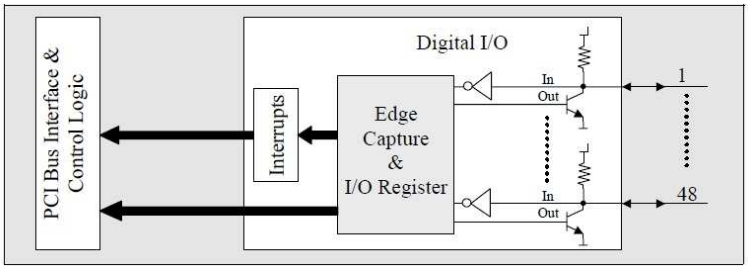

Figura 3.2: Diagramma a blocchi delle uscite digitali della sensoray 626.

Come si può vedere se le porte digitali vengono utilizzate come uscite allora siamo direttamente pilotati da un transistor, e preleviamo la tensione proprio dal collettore. Nel caso invece fossero impostate come ingressi allora è molto importante ricordarsi chè è presente una resistenza di pull-up, dove il valore della resistenza è di 10kΩ e la tensione di alimentazione è di 5V.

3.1 Scelta della componenti

In questa scheda abbiamo scelto essenzialmente due componenti:

- Scelta dell'integrato per l'alimentazione.
- Scelta del transistor per pilotare i finecorsa.

3.1.1 Fairchild Semiconductor LM7805

Questo integrato è un regolatore di tensione positivo ossia è un dispositivo che permette di avere in uscita una tensione di 5V stabile se è presente una sufficiente tensione in ingresso. Le principali caratteristiche sono le seguenti:

- Tensione di uscita 5V.
- Tensione massima in ingresso 35V.
- Tensione minima in ingrsso 10V

- Corrente massima in uscita 1A
- Protezione da sovratemperatura.
- Protezione da cortocircuito.

3.1.2 Fairchild Semiconductor BC337

Il transistor in questione è un NPN che ha come principali caratteristiche le seguenti caratteristiche:

- Tensione massima collettore emettitore di 50V.
- Tensione massima base emettitore di 5V.
- Corrente di collettore continua 800mA.
- Massimo guadagno in continua di 630.

3.2 Alimentazione delle schede

Le componenti che andranno alimentate a +5V sono i microcontrollori, il max232, i transistor e gli encoder. Per evitare che sia troppo riduttivo avere un solo ampere in uscita sono stati inseriti due lm7805. Uno di questi andrà ad alimentare i soli encoder per essere sicuri che non venga mai meno la loro alimentazione e che quindi il controllo non perda mai i lori passi. L'altro invece va a pilotare le rimanenti parti dell'elettronica che non dovrebbero consumare molto.
Lo schema di utilizzo prevede l'utilizzo di due capacità, una posta a monte dell'integrato ed una posta a valle. La seconda ha un valore raccomandato di $100\mu F$, e anche la prima è stata posta dello stesso valore.Il package scelto è il classico T0-220. Lo schema utilizzato per l'lm7805[**?**] è il seguente:

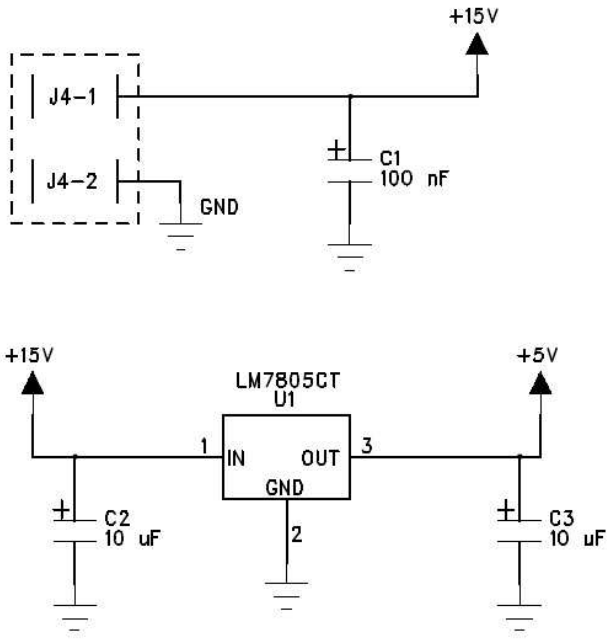

Figura 3.3: Schema di funzionamento dell'lm7805.

3.3 Finecorsa Induttivi

I sensori induttivi da noi utilizzati sono gli Aeco SI5-C0.8 NPN NC. Tali finecorsa necessitano di un alimentazione compresa tra 6 e 30V, hanno una logica di uscita di tipo npn e sono normalmente chiusi. Cosa fondamentale è l'assorbimento, pari a 10mA a 24V, veramente ridotto. Nel nostro caso sarà minore visto che verranno alimentati a 15V. Altro parametro fondamentale è la ripetibilità che secondo il costrutto si assesta a meno del 3%, importantissimo visto che la fase di calibrazione utilizza questa tipologia di sensori visto che sono molto più precisi rispetto a quelli di tipo meccanico.
Per fare in modo che la scheda sensoray potesse leggere correttamente il valore dei finecorsa è stato necessario prevedere un semplice circuito a transistor.
Lo schema utilizzato per le simulazioni è il seguente:

Il generatore di tensione V2 è il nostro finecorsa che può assumere valori compresi tra 0.65 e 0.7V se siamo arrivati a finecorsa, oppure una tensione superiore a 6V se siamo in funzionamento normale. Tali valori sono stati ricavati sperimental-

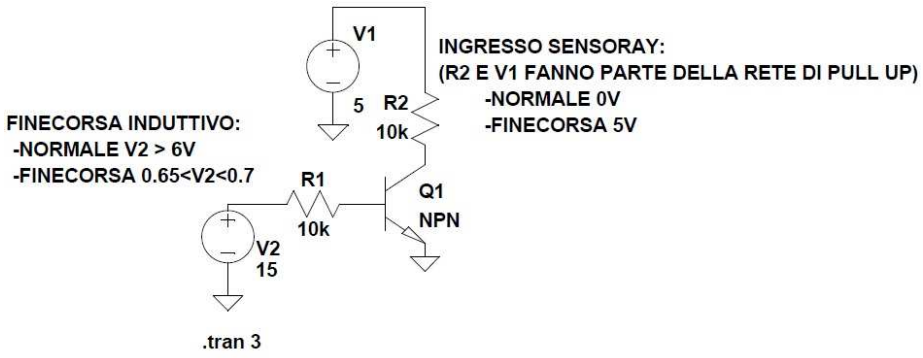

Figura 3.4: Schema simulazione dei finecorsa.

mente.

Il generatore V1 e la resistenza R2, come descritto nella figura, fanno parte della rete di pull-up all'interno della sensoray, quindi la lettura verrà effettuata esattamente sul collettore del transistor.

In questo caso se la tensione in uscita dal transistor è di circa 0V allora significa che il funzionamento è normale, mentre se è di circa 5V allora siamo in finecorsa. Nella seguente tabella possiamo trovare la risposta che riceve la sensoray in base allo stato logico del finecorsa. I valori presenti sono stati ricavati dalle simulazioni con LTSpice e poi verificati nella realtà.

Uscita Induttivo [V]	Ingresso Sensoray [V]
6	0
0.65	5

3.4 Circuito d'emergenza

Prevedere un circuito d'emergenza che possa comandare entrambi i driver è fondamentale per la realizzazione di un qualsiasi robot. Vista la sua importanza e l'azione che deve prendere è stato implementato proprio su questa scheda per poter agire in modo totale sull'elettronica.

Prima di tutto è importante capire dove e come si vuole che agisca l'emergenza. Questa deve pilotare direttamente gli stadi in potenza portando alto il valore del piedino brake dell'lmd18200. In questo modo se i microcontrollori, leggendo il

livello logico alto del brake, portassero ad un livello logico alto il pwm, otterremo la massima frenata ottenibile elettronicamente, infatti tutti gli avvolgimenti dei motori vengono chiusi su una resistenza interna e sfruttano così la corrente circolante per frenare. Nel caso comunque il piedino pwm fosse ad un livello logico basso l'integrato non farebbe altro che aprire gli avvolgimenti dei motori, il che comporterebbe un avanzamento per inerzia.
Vogliamo poi che l'emergenza possa essere azionata direttamente dal quadro, nel caso l'utente si accorgesse visivamente di un problema che richieda una frenatura immediata, oppure dal controllo nel caso, per esempio, ci si accorgesse di una velocità troppo elevata dei membri. In questo caso il controllo deve uscire con un abilitazione software sul quadro, e nel caso questa venisse a mancare è necessario fermare il tutto.
La prima parte è stata risolta semplicemente collegando assieme tutti i piedini brake dei quattro lmd18200 presenti sui driver e collegandoli ad una delle 26 vie del bus che collega le schede, in questo modo l'interfaccia, e quindi sia dal quadro, che dal controllo, può agire su tali pin.
La seconda parte invece richede un semplice circuito con dei transistor e delle resistenze, il problema principle è legato al fatto che dal quadro si arriva direttamente sulla scheda con una tensione di 15V e lo stadio in potenza avendo una logica di tipo TTL non può ricevere in ingresso più di 5V.
La soluzione è quanto mai semplice perchè con due resistenze in serie, una da 20kΩ ed una da 10kΩ si riescono ad ottenere i 5V desiderati che possono andare a pilotare direttamente gli lmd18200. Per quanto riguarda la sensoray a causa delle resistenze di pull-up presenti in ingresso è necessario introdurre un circuito a transistor uguale a quello visto per i finecorsa induttivi. Il controllo poi agirà direttamente su un relè sul quadro per realizzare l'abilitazione software.
Il circuito simulato ed implementato è il seguente:

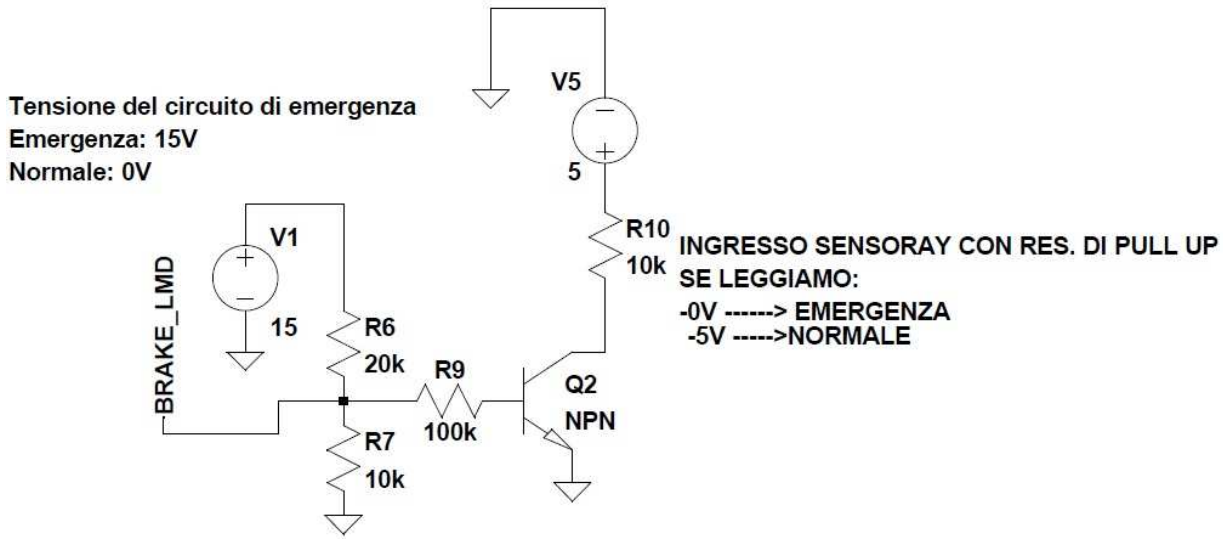

Figura 3.5: Schema circuito d'emergenza.

Il generatore di tensione V1 simula il comportamento del quadro, se non è in emergenza abbiamo 0V, se è in emergenza sono presenti 15V. La serie di resistenze permette di diminure la tensione e di ridurre la corrente. Infatti è vero che:

$$I_{emergenza} = \frac{V_{alim}}{R} = \frac{15}{30k\Omega} = 0.5mA \tag{3.1}$$

Grazie alle due resistenze riusciamo a limitare la corrente e quindi evitare che una sovracorrente possa danneggiare l'lmd18200. Quest'ultimo non influenza la serie delle due resistenze perchè la resistenza interna alla sua porta è molto elevata e può essere considerata come un circuito aperto.

La rimanente parte del circuito invece è del tutto simile a quella vista per i finecorsa induttivi, il transistor infatti viene pilotato dalla base e l'alimentazione viene direttamente presa dalla sensoray grazie alla rete di pull-up presente all'interno di essa.

In uscita sul collettore del transistor avremo 5V in caso di funzionamento normale e 0V nel caso fossimo in emergenza. Andiamo ora ad analizzare i grafici delle simulazioni per verificare il funzionamento del circuito.

Ingresso Emergenza [V]	Ingresso Sensoray [V]
5	0
0	4.8

3.5 Connettori e bus

In questa sezione verrà semplicemente elencati quali sono i collegamenti presenti tra la scheda sensoray e la scheda d'interfaccia.
Sono presenti quattro connettori principali, il primo è il bus di collegamento tra le tre schede del manipolatore, il secondo sono gli ingressi/uscite analogiche della sensoray, il terzo sono sempre ingressi/uscite della sensoray, ma questa volta digitali, l'ultimo invece è il connettore per tre encoder.
In questo progetto ne verrano implementati solamente due, ma è presente il collegamento anche per un terzo encoder in modo chiudere la catena anche per la base.
Per quanto riguarda il connettore tra le schede abbiamo:

1. REF ASSE 1: Frequenza di passi che deve effettuare il motore passo.
2. CORRENTE ASSE 1: Monitor di corrente dell'asse 1.
3. REF ASSE 2: Riferimento di tensione per il link 2.
4. CORRENTE ASSE 2: Monitor di corrente per l'asse 2.
5. REF ASSE 3: Riferimento di tensione per il link 3.
6. CORRENTE ASSE 3: Monitor di corrente per l'asse 3.
7. REF CORRENTE ASSE 1: Riferimento di corrente per l'asse 1.
8. GND: Riferimento di massa.
9. DIR ASSE 2: Riferimento di direzione dell'asse 2.
10. +15V: Tensione di alimentazione.
11. CORRENTE/TENSIONE 2: Abilitazione driver tensione/corrente asse 2.

12. +5V: Tensione di alimentazione.
13. TEMP 1 A: Flag di temperatura stadio in potenza link1.
14. RX ASSE 2: Ricezione uart asse 2.
15. TEMP 2: Flag di temperatura stadio in potenza link2.
16. MCLR: Master Clear, reset per i PIC.
17. DIR ASSE 3: Riferimento di direzione dell'asse 3.
18. RX ASSE 3: Ricezione uart asse 3.
19. TEMP 3: Flag di temperatura stadio in potenza link3.
20. TX ASSE 2: Trasmissione uart asse 2.
21. TEMP 1 B: Flag di temperatura stadio in potenza link1.
22. TX ASSE 3: Trasmissione uart asse 3.
23. DIR ASSE 1: Riferimento di direzione dell'asse 1.
24. EMERGENZA: Segnale esterno di emergenza.
25. END EFFECTOR: Alza/Abbassa end-effector.
26. CORRENTE/TENSIONE 3: Abilitazione driver tensione/corrente asse 3.

Per quanto riguarda gli ingressi e le uscite analogiche abbiamo che:

- 1-42 REF ASSE 1: Frequenza di passi che deve effettuare il motore passo.
- 2-44 REF ASSE 2: Riferimento di tensione per il link 2.
- 3-46 REF ASSE 3: Riferimento di tensione per il link 3.
- 4-48 REF CORRENTE ASSE 1: Riferimento di corrente per l'asse 1.
- 1-42 CORRENTE ASSE 1: Monitor di corrente dell'asse 1.
- 2-44 CORRENTE ASSE 2: Monitor di corrente per l'asse 2.

- 3-46 CORRENTE ASSE 3: Monitor di corrente per l'asse 3.

In quest'elenco il primo valore identifica il canale della sensoray utilizzato il secondo invece identifica il pin nella piattina presente nella scheda d'interfaccia. Usando la stessa notazione, consideriamo ora le uscite digitali:

- 1-47 FI: Finecorsa induttivo
- 2-45 FI: Finecorsa induttivo
- 3-43 FI: Finecorsa induttivo
- 4-41 FI: Finecorsa induttivo
- 5-39 FI: Finecorsa induttivo
- 6-37 FI: Finecorsa induttivo
- 7-35 END EFFECTOR: Alza/Abbassa end-effector.
- 9-31 DIR ASSE 2: Riferimento di direzione dell'asse 2.
- 10-29 CORRENTE/TENSIONE 2: Abilitazione driver tensione/corrente asse 2.
- 11-27 TEMP 1 A: Flag di temperatura stadio in potenza link1.
- 12-25 TEMP 2: Flag di temperatura stadio in potenza link2.
- 13-23 DIR ASSE 3: Riferimento di direzione dell'asse 3.
- 14-21 TEMP 3: Flag di temperatura stadio in potenza link3.
- 15-19 TEMP 1 B: Flag di temperatura stadio in potenza link1.
- 16-17 DIR ASSE 1: Riferimento di direzione dell'asse 1.
- 18-13 CORRENTE/TENSIONE 3: Abilitazione driver tensione/corrente asse 3.
- 23-3 ABILITAZIONE SOFTWARE: Segnale per abilitare il quadro e sbloccare i motori.

- 24-1 EMERGENZA: Segnale esterno di emergenza.

Porta Analogica Simulink	Pin scheda	Descrizione
1	42	REF ASSE 1
2	44	REF ASSE 2
3	46	REF ASSE 3
4	48	REF CORRENTE ASSE 1
1	42	CORRENTE ASSE 1
2	44	CORRENTE ASSE 2
3	46	CORRENTE ASSE 3

Porta Digitale Simulink	Pin scheda	Descrizione
1	47	FI
2	45	FI
3	43	FI
4	41	FI
5	39	FI
6	37	FI
7	35	END EFFECTOR
9	31	DIR ASSE 2
10	29	CORRENTE/TENSIONE 2
11	27	TEMP 1 A
12	25	TEMP 2
13	23	DIR ASSE 3
14	21	TEMP 3
15	19	TEMP 1 B
16	17	DIR ASSE 1
18	13	CORRENTE/TENSIONE 3
23	3	ABILITAZIONE SOFTWARE
24	1	EMERGENZA

3.6 Schemi completi dell'interfaccia

Ecco qui presenti gli schemi sia della parte logica che del layout del pcb.

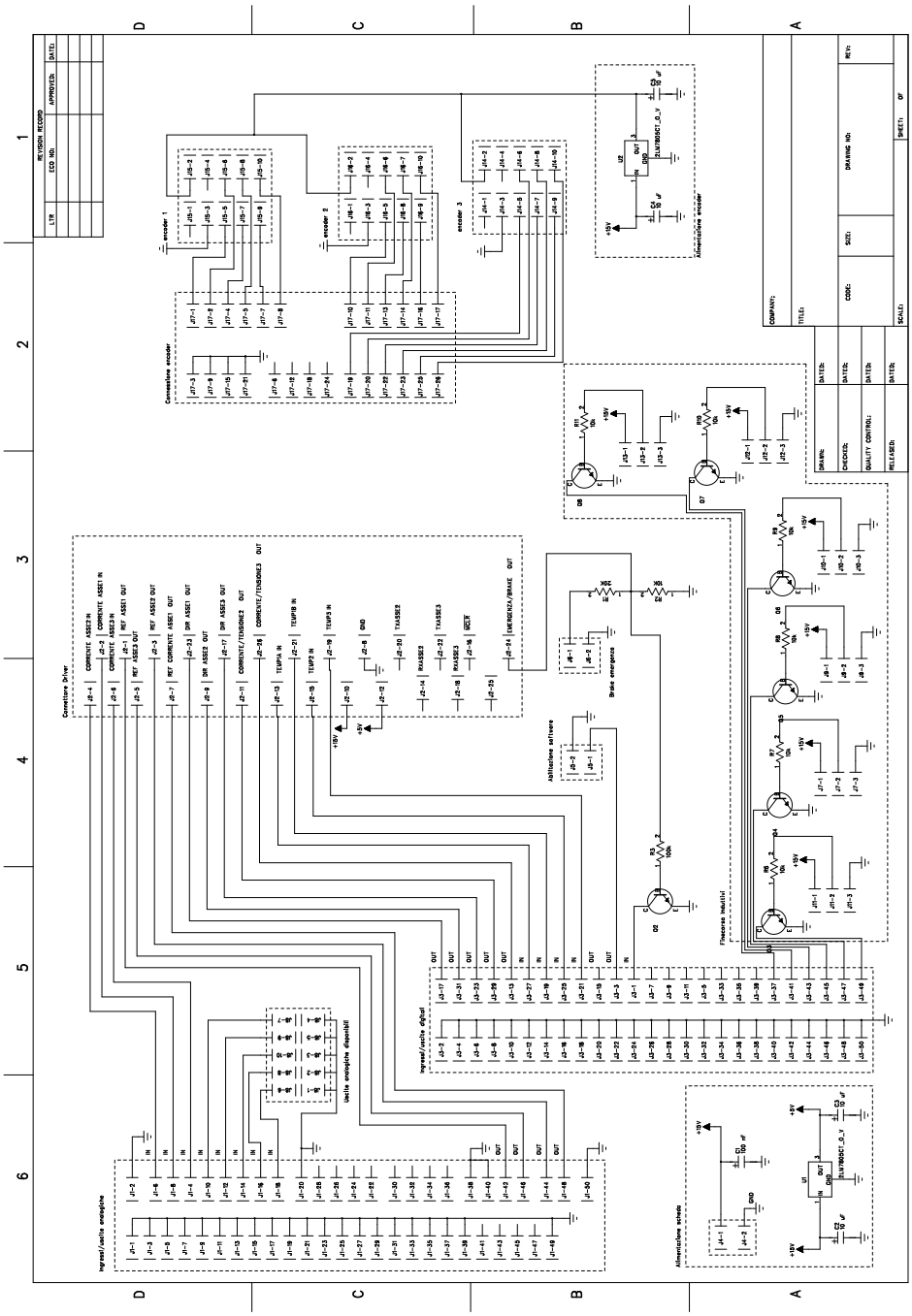

Figura 3.6: Schema logico della scheda d'interfaccia.

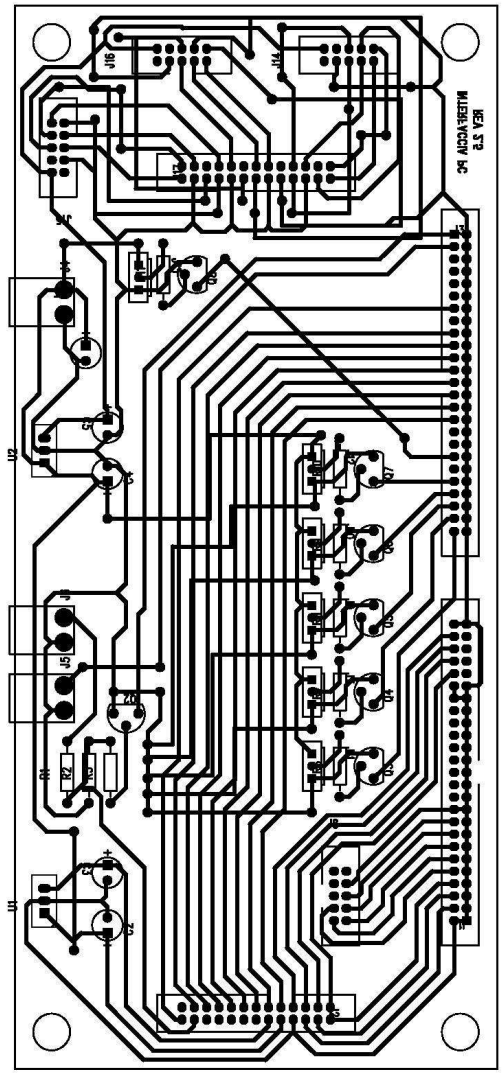

Figura 3.7: Schema pcb della scheda d'interfaccia.

Capitolo 4

Quadro elettrico

Il quadro elettrico ha il compito di contenere all'interno i driver e tutta l'elettronica necessaria al funzionamento del robot. Parte fondamentale sono i gli alimentatori, oltre che al circuito utilizzato per gestire le emergenze. Analizzeremo ora le componenti che verranno utilizzate per la realizzazione del quadro elettrico.

4.1 Alimentatore dei driver - Cabur CSF120B

L'alimentatore scelto per i driver è un cabur CSF120B. La tensione di uscita può essere regolata da 12 a 15V ed è in grado di erogare fino a 7A, la tensione in ingresso nominale è compresa tra 90-230Vac. Tale alimentatore è di tipo switching stabilizzato. Il montaggio è previsto su guida DIN all'interno del quadro. Tra le tante caratteristiche sono è da sottolineare la protezione da cortocircuito, sovraccarico, sovratemperatura e sovratensioni in ingresso e in uscita[6].

4.2 Alimentatori motori - Cabur CSF500D

Gli alimentatori dei motori sono sempre della Cabur[6]. La tensione di ingresso nominale anche in questo caso si adatta alla linee monofase casalinghe ed è pari a 85 Vac - 264 Vac, ma può essere alimentato anche in continua 90 Vdc - 350 Vdc. La tensione nominale in uscita è pari a 48Vdc, ma è regolabile da 30Vdc fino a 56Vdc e questo ci permette di sfruttare al massimo gli stadi in potenza che ope-

rano al massimo a 55V. La corrente di uscita si assesta a 10A, ma si può arrivare fino a 13A permanenti grazie alla tecnologia POWER BOOST. Sono previsti due di questi alimentatori e saranno entrambi montati su guida DIN all'interno del quadro. Le protezioni presenti sono da cortocircuito, sovraccarico, sovratemperatura e sovratensioni in ingresso e in uscita. Saranno necessari due alimentatori in quanto uno andrà ad alimentare il motore passo ed uno i due motori in continua. Visto che l'alimentatore può erograre fino a 15A i motori potranno raggiungere le massime performance visto che la corrente di picco erogabile dagli stadi in potenza è di 6A.

4.3 Schema del quadro

Lo schema di funzionamento del quadro è molto semplice. A monte di tutto è prevista una protezione magnetotermica-differenziale per proteggere la circuiteria da un qualsiasi malfunzionamento o guasto. A valle del magnetotermico-differenziale sono collegati i tre alimentatori. I due alimentatori da 48V sono direttamente collegati alle schede tramite i relativi morsetti. Per l'alimentatore da 15V invece è presente un circuito di marcia ed arresto, ed in serie sono collegati tutti gli switch ed i relè che possono far scattare un emergenza. Lo schema utilizzato è quello di un relè autoalimentato o con autoritenuta. Infatti l'alimentatore è direttamente collegato alla bobina di un relè, la quale verrà chiusa solo se non sono presenti emergenze, è premuto il pulsante di marcia ed è presente l'abilitazione software. Quando tutte queste condizioni sono avverate i secondari del relè permettono la chiusura del circuito di autoalimentazione e disattivano l'emergenza all'ingresso della scheda di interfaccia, in questo modo il robot è pronto per essere utilizzato. Nel caso invece si premesse il pulsante di arresto o scattasse un emergenza il sistema entrerà in emergenza e i motori verrano frenati tramite il loro piedino di brake.
Lo schema del quadro è quello presente in figura:

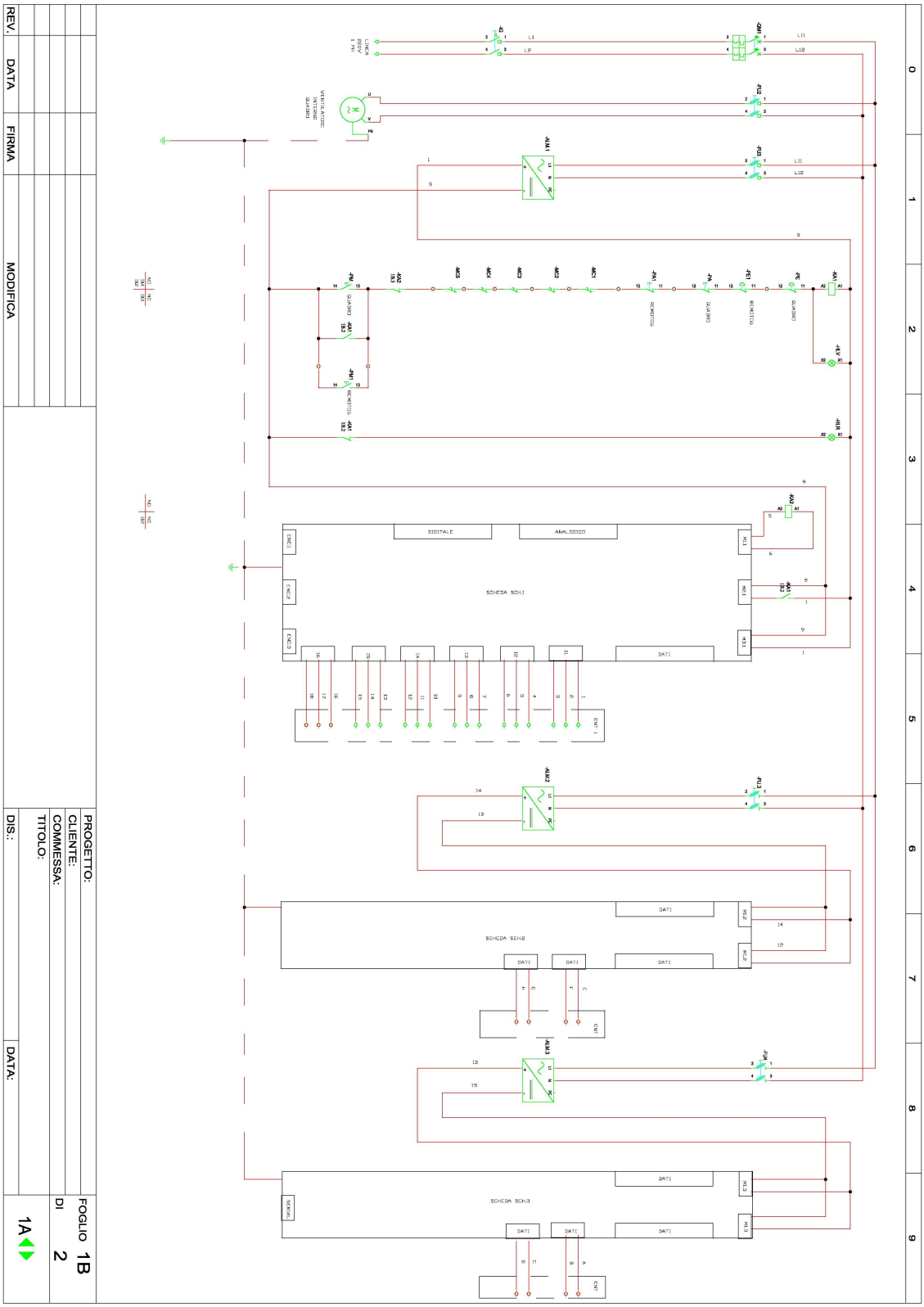

Figura 4.1: Schema del quadro.

Capitolo 5

Prove Sperimentali

Dopo la progettazione delle schede siamo passati alla loro realizzazione. In questo caso è stata usata la procedura di fotoincisione per ottenere gli stampati. Questa procedura consiste nelle seguente operazioni:

1. Stampare su carta lucida, sul lato ruvido il disegno dello stampato.

2. Riptetere tale stampa per almeno quattro cinque volte in modo da avere uno strato di inchiostro sufficientemente grosso ed uniforme.

3. Incollare sul lato o suoi lati da fotoincidere il lucido stampato e fare in modo che esso sia molto aderente al basetta presensibilizzata.

4. Inserire la basetta all'interno del bromografo per un tempo sufficiente ad impressionare la scheda. Il tempo di esposizione varia a seconda della potenza della lampade UV. In questo modo elimineremo il photoresist nei punti non coperti dell'inchiostro del lucido.

5. Ripetere tale operazione per entrambi i lati e girando la scheda di 45 gradi di volta in volta.

6. Finita la precedente operazione togliere i lucidi dalla scheda e lavarla bene con acqua. Si potranno notare già le piste che vanno formandosi nella basetta.

7. A questo punto è necessario scaldare del cloruro ferrico (è un sale di ferro dell'acido cloridrico). Bisogna ora fare molta attenzione in quanto è un composto nocivo ed irritante. Scaldandolo aumenteremo le sue capacità corrosive, in questo modo inserendo la basetta all'interno del composto andremo ad eliminare il rame non coperto del photoresist. Non bisogna lasciare la schede per troppo tempo all'interno di questo composto, altrimenti c'è il rischio di eliminare anche le piste coperte dal photoresist.

8. Quando visivamente la scheda è pronta, ossia quando le piste si vedono distintamente e con un colore uniforme su tutta la basetta, possiamo toglierla dal composto e sciacquarla con acqua.

A questo punto siamo pronti per forare la scheda utilizzando un fresino o ancora meglio un fresino alloggiato su un semplice trapano a colonna. Bisogna prestare attenzione effettuando i fori perchè nel caso venisse utilizzata una punta troppo larga oppure si andasse a forare su un pad molto stretto c'è il rischio di alzare il pad o rimuoverlo del tutto.
Una volta realizzati tutti i fori e verificato che ogni componente si riesce ad inserire in essi si può procedere all'operazione di saldatura, ricordando che è necessario utilizzare gli occhiali protettivi. Per effettuare una buona saldatura le operazioni da seguire sono le seguenti:

1. Inserire tutti i componenti da montare sulla scheda.

2. Piegare i reofori se necessario per fare in modo che non cadano una volta che la scheda viene rovesciata.

3. Fissare la scheda rovescia ad una stazione saldante.

4. Preparare il saldatore ad una temperatura di circa 400 gradi. Se il saldatore ha una punta molto usurata è meglio portare la temperatura a 450 gradi.

5. Scaldare con la punta del saldatore il pad per pochi attimi in modo da evitare i scaldare troppo il componente, l'eccessivo calore potrebbe danneggiarlo. Per le componenti più delicate è più indicato utilizzare degli zoccoli.

6. Appoggiare il filo di stagno tra la punta del saldatore, il pad e il reoforo.

7. Aspettare che lo stagno copra in modo uniforme il pad circolando tutt'attorno.

8. Alzare verso l'alto la punta del saldatore seguento il reoforo.

9. Se la saldatura è avvenuta correttamente sarà lucida e non opaca. Inoltre non deve avere la tipica forma della pallina, ma deve seguire la forma del cono, in modo da far bene contatto con il pad.

Le schede a questo punto si possono ritenere completate e pronte per essere testate. Il risultato ottenuto è il seguente:

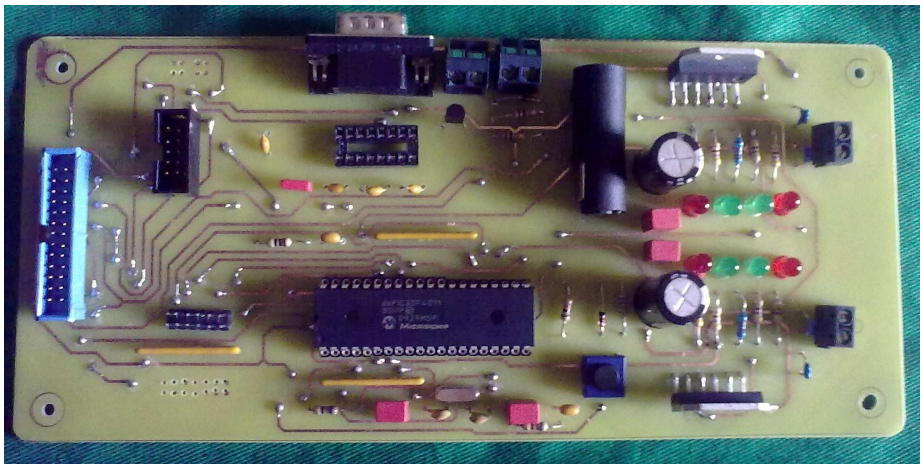

Figura 5.1: Scheda del driver motore passo.

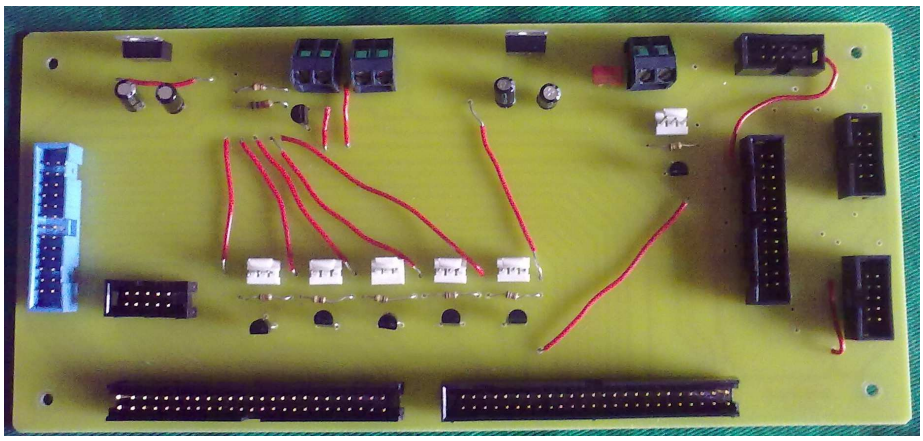

Figura 5.2: Scheda d'Intefaccia.

5.1 Test scheda interfaccia

Prima di tutto è stata testata la scheda di interfaccia. Per verificarne il corretto funzionamento sono stati applicati 15V all'ingresso di alimentazione e sono state verificate con il tester le tensioni all'uscita degli LM7805.
L'esito è stato subito positivo e quindi non è stata necessaria alcuna modifica della scheda.
Successivamente è stata collegata, tramite le relative piattine, alla sensoray per verificare che che ogni ingresso/uscita predisposto corrispondesse effettivamente a quello desiderato. Anche in questo caso il test è stato superato con successo.

5.2 Test driver motore passo

Il driver del motore passo è stato testato prima di tutto in modalità stand-alone, collegato alla scheda d'interfaccia per ottenere l'alimentazione. Per effettuare il test è stato cambiato il firmware del microcontrollore. In questa fase infatti il software non deve leggere nessun dato dall'adc, ma deve semplicemente eseguire un numero di passi costante.

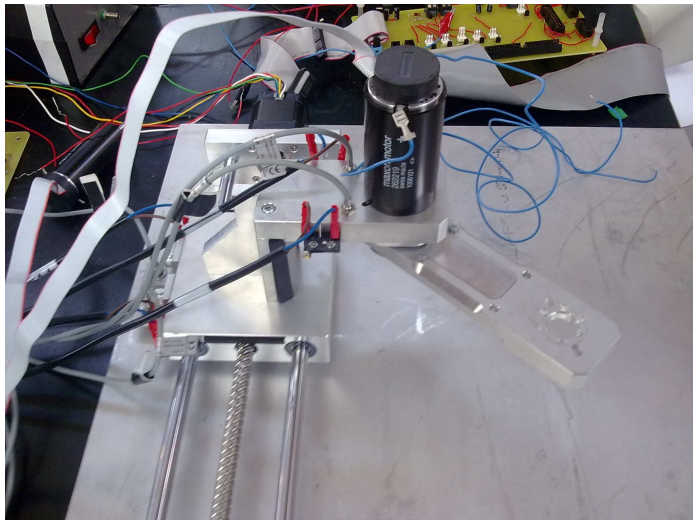

Figura 5.3: Robot in fase di test.

La prima prova ha subito rivelato alcuni problemi legati al pilotaggio del motore, infatti questo vibrava veramente molto. Come soluzione è stato cambiata la

modalità di driving del motore, passando dal full step drive prima al wave step drive e poi all'half step drive.

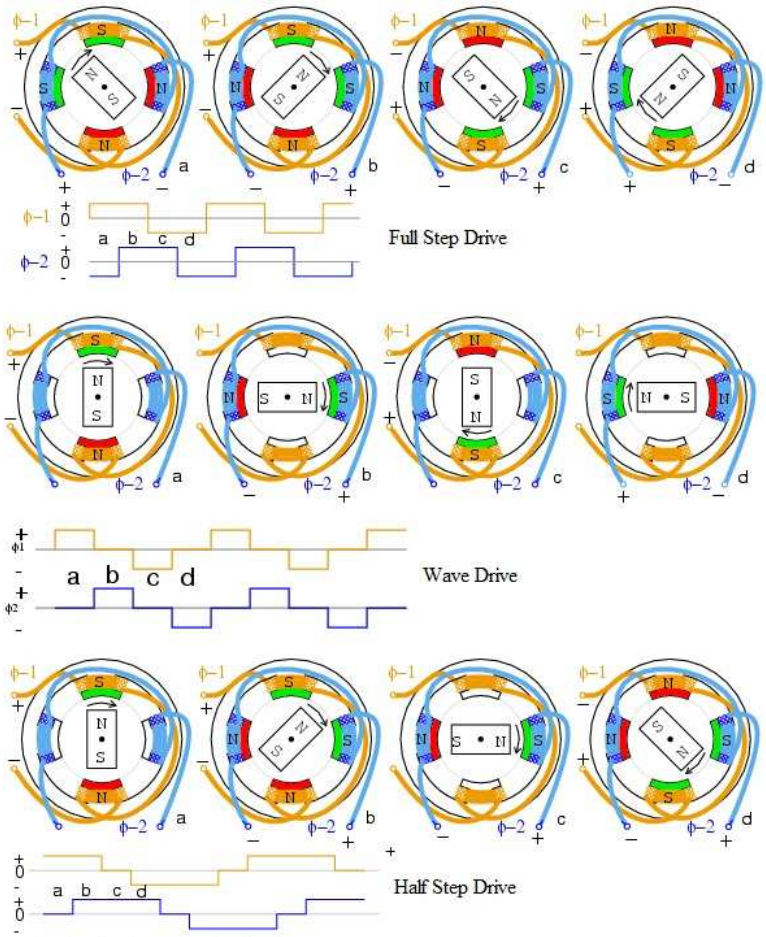

Figura 5.4: Metodi di drive.

La seconda modalità prevede di polarizzare un solo avvolgimento lasciando spento l'altro, in questo modo la curva coppia velocità dovrebbe essere meno irregolare e presentare meno picchi di risonanza. I risultati sono stati buoni, ma non del tutto soddisfacenti in quanto il motore continuava a presentare ancora una leggere vibrazione.

Passando all'half step drive, che non è altro che una combinazione dei due metodi di drive precedenti, il motore non ha più presentato alcun tipo di vibrazione.

Nel seguente grafico si possono vedere appunto le tre modalità di drive con i relativi esempi di come pilotare gli stadi in potenza per ottenere tale tipologia di

pilotaggio.

A questo punto la scheda d'interfaccia è stata collegata alla sensoray ed tramite un semplice programma simulink è stato verificato il corretto funzionamento dell'adc.

Superato anche questo test è stato preparato un semplice file matlab in cui è stato implementato una profilo trapezoidale di velocità. La velocità lineare successivamente è stata convertita in passi al secondo e successivamente è stata implementata una semplice funzione per inviare un numero di passi intero ad ogni ciclo evitando così di perdere passi.

A questo punto il motore si è mosso correttamente e in questi grafici possiamo trovare nel seguente ordine il profilo trapezoidale di velocità applicato con la relativa posizione, il numero di passi eseguiti nel tempo ed il monitor di corrente.

Il primo grafico fa capire come sia necessario un programma all'interno del controllo in grado di recuperare tutte le frazioni di passo, raggruppandole finchè non diventano un passo intero.

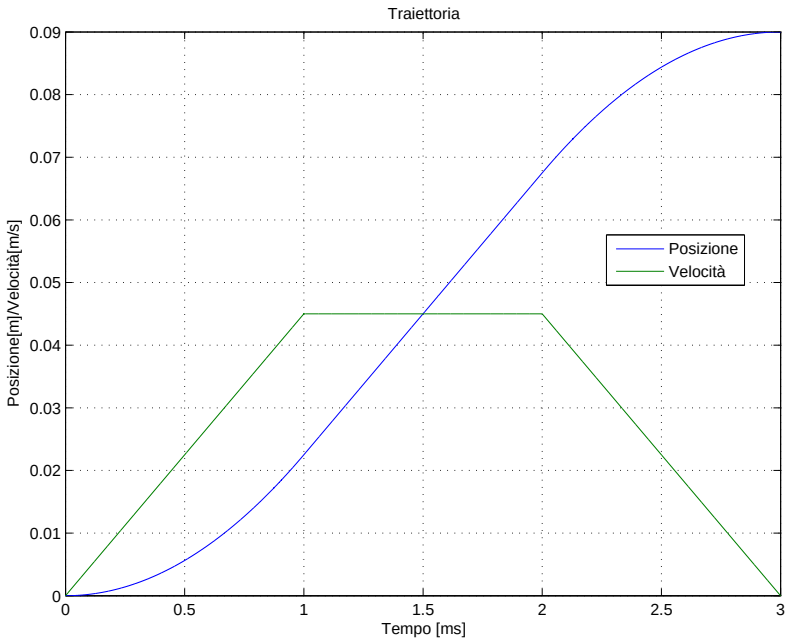

Figura 5.5: Traiettoria pianificata.

Il secondo grafico mostra invece come in totale debbano essere eseguiti 900 passi

e quindi, visto che un passo permette uno spostamento di 0.1mm, lo spostamento totale è di 9cm in 3 secondi.

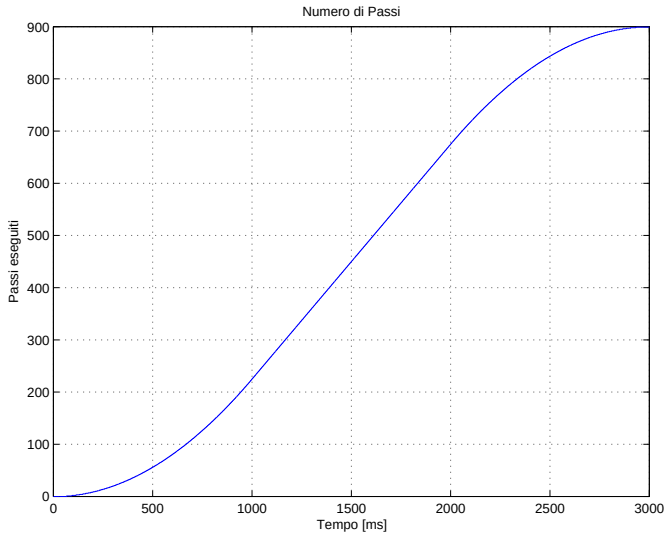

Figura 5.6: Passi da eseguire nel tempo.

Il monitor di corrente invece mostra come il sistema ha una corrente maggiore nella fase di accelerazione e decelerazione, è da ricordare che il monitor di corrente è sempre positivo e che quindi la corrente nella fase di decelerazione avrebbe segno opposto.

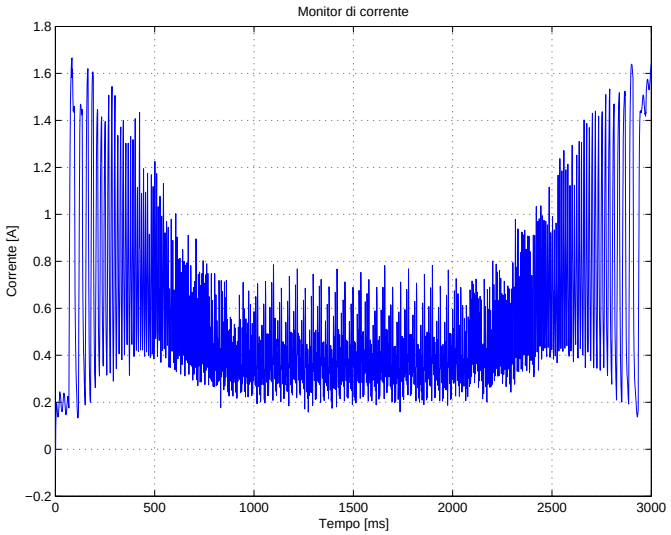

Figura 5.7: Monitor di corrente.

Conclusioni

In questo lavoro di tesi è stato progettato e realizzato il driver per pilotare un motore passo. Possiamo ritenerci pienamente soddisfatti del lavoro svolto in quanto il driver risponde come desiderato. Sono presenti però delle possibili vie di sviluppo molto interessanti.
La prima è la realizzazione del cosiddetto chopper driver che prevede di mantenere una corrente costante all'interno degli avvolgimenti e non una tensione costante ai capi degli stessi. Sarebbe quindi necessario modificare di poco il driver esistente facendo leggere al microcontrollore entrambi i monitor di corrente e realizzando all'interno di microcontrollore un regolatore PI in grado di mantenere la corrente costante ad un riferimento dato.
La seconda via di sviluppo riguarda la possibilità di introdurre un ulteriore anello di retroazione anche per il motore passo, in modo da rendere a catena chiusa anche tale motore. Le possibili implementazioni sono due: una potrebbe utilizzare un encoder ottico, ricordando che sulla scheda d'interfaccia è già prevista l'implementazione di un terzo encoder, oppure si potrebbe usare una potenziometro collegato al carro in modo da leggere la posizione lineare della base anzichè quella angolare del motore. I due approcci sono completamente differenti in quanto il primo misura la posizione relativa del motore, mentre il secondo misura la posizione assoluta del carro.

Bibliografia

[1] http://www.federmacchine.it/.

[2] MicroChip, *dsPic30F4011 Datasheet*. MicroChip, 2008.

[3] N. Semiconductor, *LMD18200 H-Bridge*. National SemiConductor, 2005.

[4] http://www.vincenzov.net/tutorial/passopasso/stepper.htm.

[5] Maxim, *Max 220-232*. Maxim, 2009.

[6] www.cabur.it.

Ringraziamenti

Vorrei ringraziare la mia famiglia e in particolare i miei genitori per il supporto che mi hanno dato in tutti questi anni di studi, sopportando l'incredibile confusione che portavo in tutte le stanze dove solamente pensavo di studiare e soprattutto per avermi sempre lasciato la libertà di scegliere quella che era la strada che ritenevo migliore.
Ringrazio anche la mia fidanzata Elisa per aver avuto pazienza in questi ultimi mesi in cui il lavoro per la tesi si è fatto molto serrato, per tutte le attese fuori dalle aule e la pazienza nelle settimane post-esami.
Altri ringraziamenti ad Alberto,Daniele,Diane e Diego miei compagni di studi in questi bellissimi anni e compagni anche per questo incredibile progetto che abbiamo sviluppato assieme.
Per finire ringrazio tutti gli amici per essermi stati sempre vicini, tra i tanti una ringraziamento in particolare va a Marco per il supporto tecnico.