UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Elettronica per l'Automazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica
(Prof. Riccardo Cassinis)

Sviluppo di un dispositivo hardware e relativo software per la gestione di un marker e di un motore a passo

Elaborato di esame di: **Matteo Manzini**

Consegnato il: **09 giugno 2005**

Sommario

L'elaborato svolto si integra in un più ampio progetto che si propone di sviluppare un sistema di posizionamento a basso costo per il robot ActivMedia Pioneer 3 "Morgul". Al fine di identificare la posizione del robot si è deciso di utilizzare un gruppo di LED ad alta intensità che lampeggiano ad una determinata frequenza scelta dall'utente. Elaborando quindi le immagini riprese da una web cam deve essere possibile riconoscere i lampeggi dei LED e quindi la posizione del robot. Nasce perciò il problema di creare un dispositivo hardware in grado di comandare una serie di LED a frequenze arbitrarie e di posizionarli in modo tale da poter essere rilevati da una web cam, nonché di sviluppare un'interfaccia grafica che permetta all'utente di accedere alle funzionalità dell'apparecchio progettato nel modo più semplice.

1. Introduzione

Questo elaborato si prefigge di creare una sorgente luminosa intermittente le cui tempistiche e la cui posizione possano essere fissate con adeguata precisione. Si compone di una parte hardware il cui scopo è di progettare una scheda atta a gestire le temporizzazioni di tre gruppi di LED e di un motore a passo, e di una parte software che permetta di accedere alle funzionalità offerte dalla scheda tramite calcolatore.

Per una più semplice analisi delle soluzioni adottate il progetto sarà diviso in tre parti:

- Progettazione hardware: in cui verrà illustrato come è stata costruita la scheda e le scelte componentistiche adottate.
- Programmazione firmware: in questo paragrafo si mostrerà come il microcontrollore scelto è stato programmato per svolgere le funzionalità richieste.
- Programmazione interfaccia software: descrive la programmazione dell'interfaccia utente che permette la comunicazione tra calcolatore e microcontrollore.

2. Il problema affrontato

Si è dovuto affrontare il problema di creare un supporto hardware in grado di gestire tre gruppi di LED con tempistiche di lampeggio indipendenti e selezionabili dall'utente, con risoluzione nell'ordine del millisecondo. Inoltre vi era la necessità di realizzare una struttura che permettesse ai LED di essere posizionati nella direzione ottimale per la rilevazione da parte di una web cam.

Per far ciò è stato necessario scegliere un microcontrollore adeguato e costruirgli intorno un circuito adatto ai nostri scopi, che adottasse una connessione seriale per comunicare con il robot per cui è stato progettato. Successivamente si è dovuto programmare il microcontrollore in modo che potesse gestire tutte le sue funzioni richieste, prestando particolare attenzione al rispetto delle tempistiche. In fine si è presentata l'esigenza di sviluppare un'interfaccia software che permettesse di comandare il microcontrollore tramite calcolatore, anche in remoto.

3. La soluzione adottata

3.1. Progettazione hardware

La progettazione hardware ruota intorno al microcontrollore scelto: un PIC 16F876 prodotto dalla Microchip. Questo integrato, abbinato a un quarzo da 4 MHz, risulta infatti abbastanza veloce da gestire la struttura hardware nei limiti imposti dalle specifiche. Anche la memoria è più che sufficiente per l'utilizzo che ne è stato fatto.

La comunicazione del PIC con il calcolatore e con il robot è di tipo seriale. Tale soluzione ha richiesto l'utilizzo di un integrato MAX232 al fine di adattare i livelli di tensione.

Per comandare i tre gruppi di LED si è ricorso a tre MOSFET IRLZ14 che fanno da interruttori su un'alimentazione di 12 Volt.

Al fine di eseguire alcune prove è stata costruita una struttura da collegarsi all'uscita LEDA (vedere schema elettrico in Figura 1) che permette la giusta collimazione di 19 LED ad alta intensità al fine di rendere visibile la luce generata anche in ambiente particolarmente luminoso.

Per il sistema di posizionamento si è ricorso ad un motore a passo, poiché permette una precisione di posizionamento elevata, indispensabile per la progettazione di un sistema di puntamento. A tale scopo si è fatto ricorso a una scheda driver già a disposizione del laboratorio, che sfruttando un integrato Toshiba TA8435, permette di comandare il motore attraverso alcuni semplici ingressi logici.

Dal momento che il motore doveva essere montato su una struttura (che chiameremo torretta rotante) che ne consentisse il posizionamento e che gli impedisse di girare su se stesso più di quanto consentissero i fili dei LED che saranno sistemati sopra, è occorso implementare due finecorsa per non lasciassero ruotare il motore più di un certo angolo e una fotocellula per permettere di spostare il motore in una posizione conosciuta.

In figura 1 viene mostrato lo schema elettrico della scheda logica realizzata (un ingrandimento è presentato in appendice B).

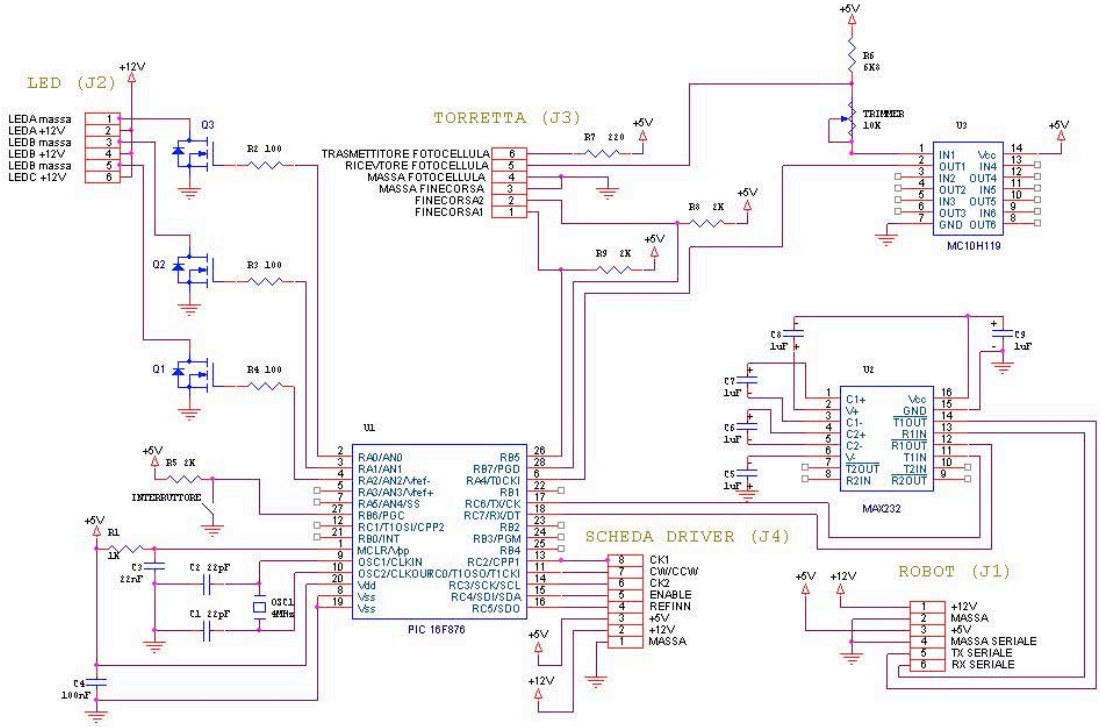

Figura 1: Schema elettrico scheda logica.

Si vede, in altro a sinistra, il connettore J2 destinato al collegamento dei tre gruppi di LED (rispettivamente LEDA, LEDB, LEDC). Appena a destra vi è il connettore J3 per la torretta rotante, con i contatti per la fotocellula e per i due finecorsa. Sotto, vicino al PIC, si trova il connettori J4 per la scheda driver del motore a passo, che comprendono gli ingressi logici per il comando del motore e le alimentazioni per la scheda driver. In basso a destra trova posto il connettore J1 da collegare al robot, esso include l'alimentazione per la scheda e la comunicazione seriale. Si può sfruttare quest'ultima porta per connettere la scheda direttamente ad un calcolatore piuttosto che al robot.

La figura 2 mostra una foto del lato superiore della scheda, mentre nella foto 3 si vede il lato inferiore.

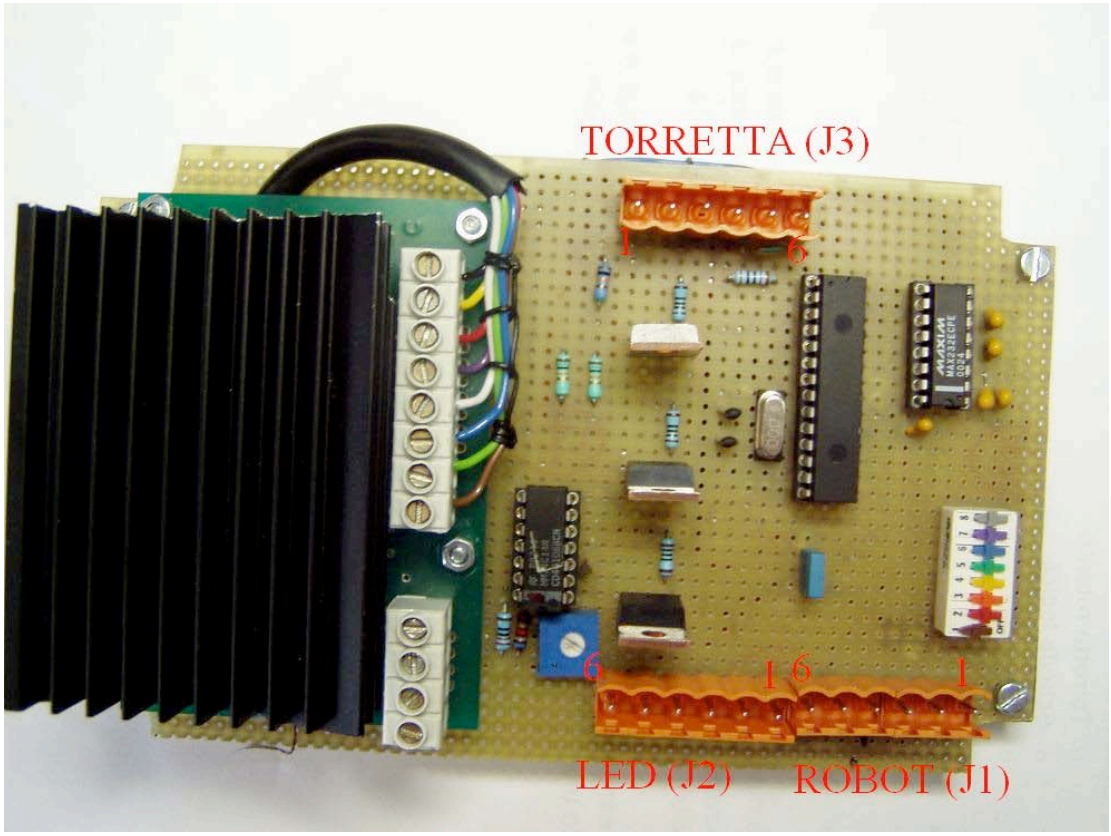

Figura 2: Parte superiore scheda logica.

In questa figura si può notare la scheda driver del motore a passo che occupa quasi tutta la metà sinistra, sormontata da un dissipatore.

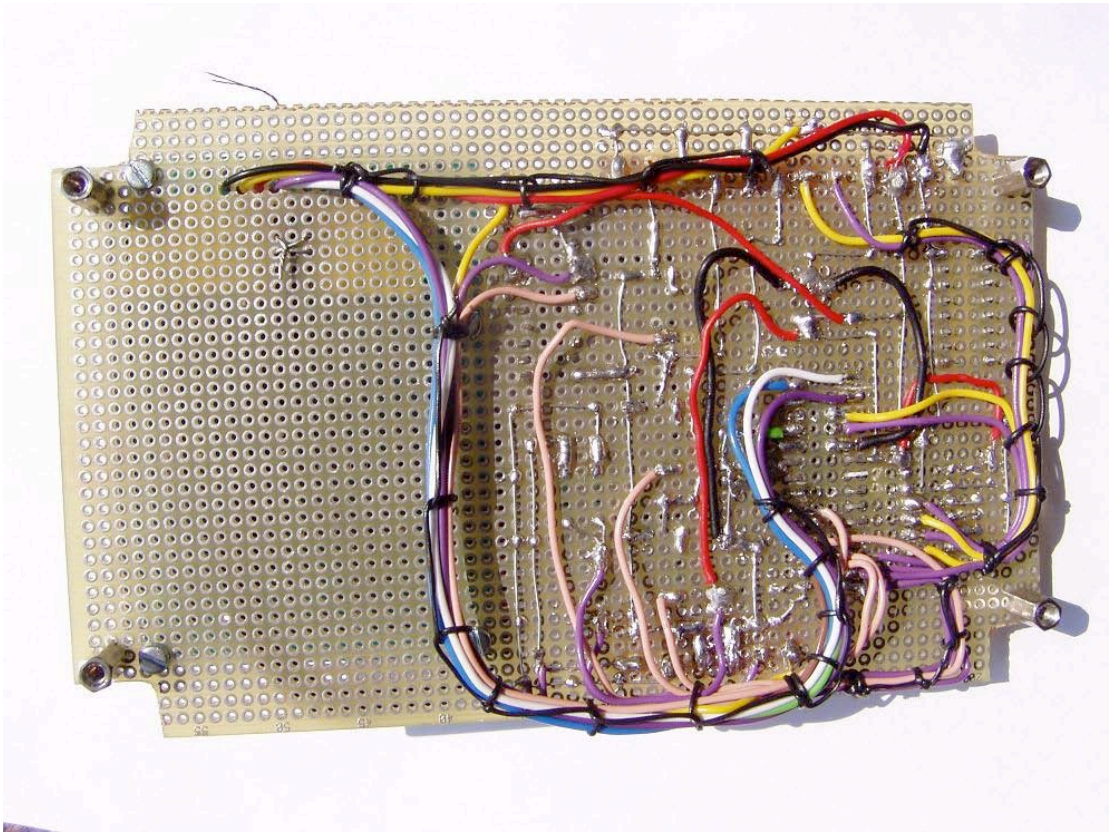

Figura 3: Parte inferiore scheda logica.

Vista la relativa semplicità della scheda si è preferito non fare un circuito stampato e saldare tutto a mano.

3.1.1. PIC 16F876

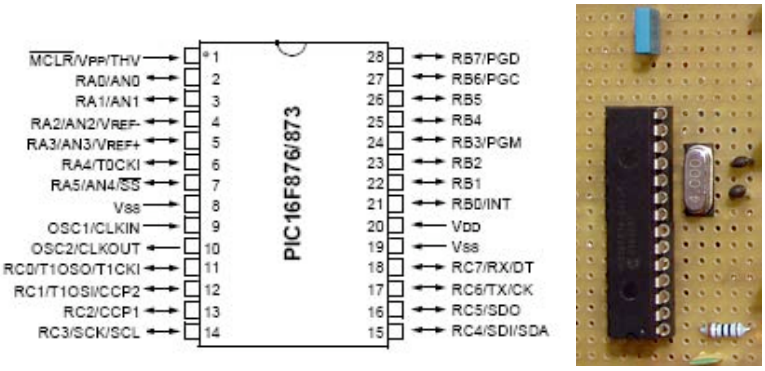

Figura 4: Schema e foto PIC 16F876.

Il microcontrollore da noi utilizzato è un PIC 16F876 prodotto dalla Microchip, abbinato ad un quarzo da 4 MHZ. Dal momento che per un'istruzione normale sono necessari quattro cicli dell'oscillatore risulta che il tempo per un'istruzione è di 1 μs, abbastanza breve per i nostri scopi. Per quanto riguarda la memoria rende disponibile 8K di memoria flash, 386 byte di memoria dati e 256 bytes di memoria EEPROM, più che sufficienti per l'esecuzione del nostro firmware.

Il PIC è dotato di 22 porte di I/O bidirezionabili suddivise in tre gruppi: 6 porte A, 8 porte B e 8 porte C. Sono state utilizzate le seguenti:

- Porta A_0 che comanda il transistor dei LEDB. Su uscita alta accende i LED, su bassa li spegne.
- Porta A_1 che comanda il transistor dei LEDA. Su uscita alta accende i LED, su bassa li spegne.
- Porta A_2 che comanda il transistor dei LEDC. Su uscita alta accende i LED, su bassa li spegne.
- Porta A_4 che riceve l'uscita della fotocellula.
- Porta B_5 che rivela la pressione del FINECORSA 1 del motore.
- Porta B_6 che rivela l'attivazione dell'interruttore per il lampeggio con tempistiche predefinite.
- Porta B_7 che rivela la pressione del FINECORSA 2del motore.
- Porta C_0 che comanda l'ingresso CW della scheda driver del motore a passo.
- Porta C_2 che comanda l'ingresso CK1 della scheda driver del motore a passo.
- Porta C_3 che comanda l'ingresso CK2 della scheda driver del motore a passo.
- Porta C_4 che comanda l'ingresso ENABLE della scheda driver del motore a passo.
- Porta C_5 che comanda l'ingresso REFIN della scheda driver del motore a passo.

Le porte B che vanno dalla 4 alla 7 (che comprendono i due finecorsa e l'interruttore) sono associate ad un interrupt che si attiva quando viene cambiato lo stato di una di queste porte.

Per sincronizzare le varie operazioni eseguite dal firmware si è fatto uso del TIMER1, un timer hardware a 16 bit che si incrementa ogni ciclo istruzione e genera un interrupt quando va in overflow.

3.1.2. MAX 232

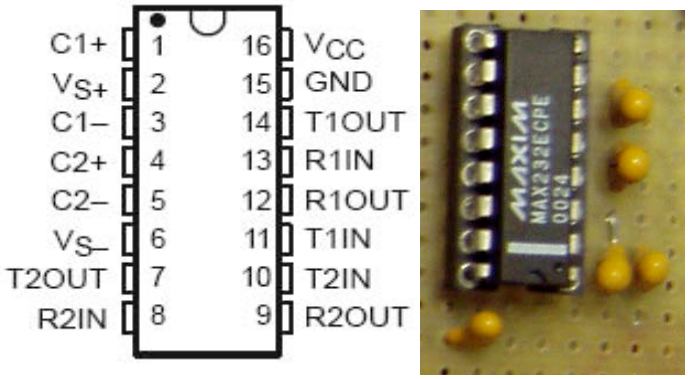

Figura 5: Schema e foto MAX232.

Il pic 16F876 ha una porta seriale integrata che comunica tramite livelli di tensione TTL. Il calcolatore utilizzato per i test e il robot a cui è stato collegato il dispositivo utilizzano invece una porta RS232 che comunica con livelli di tensione differenti. L'integrato MAX232 ha lo scopo di adattare i livelli di tensione da TTL a RS232, permettendo la comunicazione.

3.1.3. Scheda driver del motore a passo

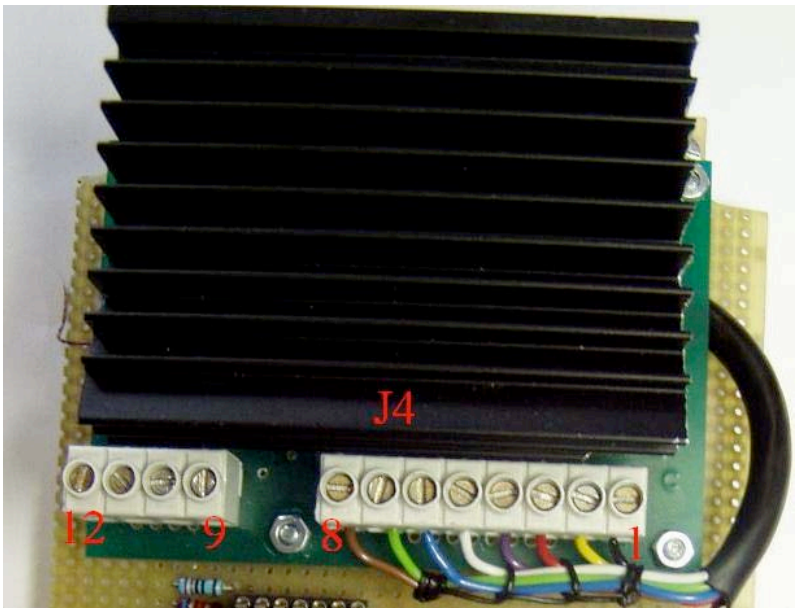

Figura 6: Foto scheda driver.

Questa scheda fornita dal laboratorio, costruita intorno all'integrato Toshiba TA8435, contiene un driver di potenza in grado di alimentare un motore a passo bipolare. Integra la logica di controllo necessaria a generare le corrette forme d'onda di pilotaggio degli avvolgimenti del motore, partendo da segnali d'ingresso logici quali clock e senso di rotazione.

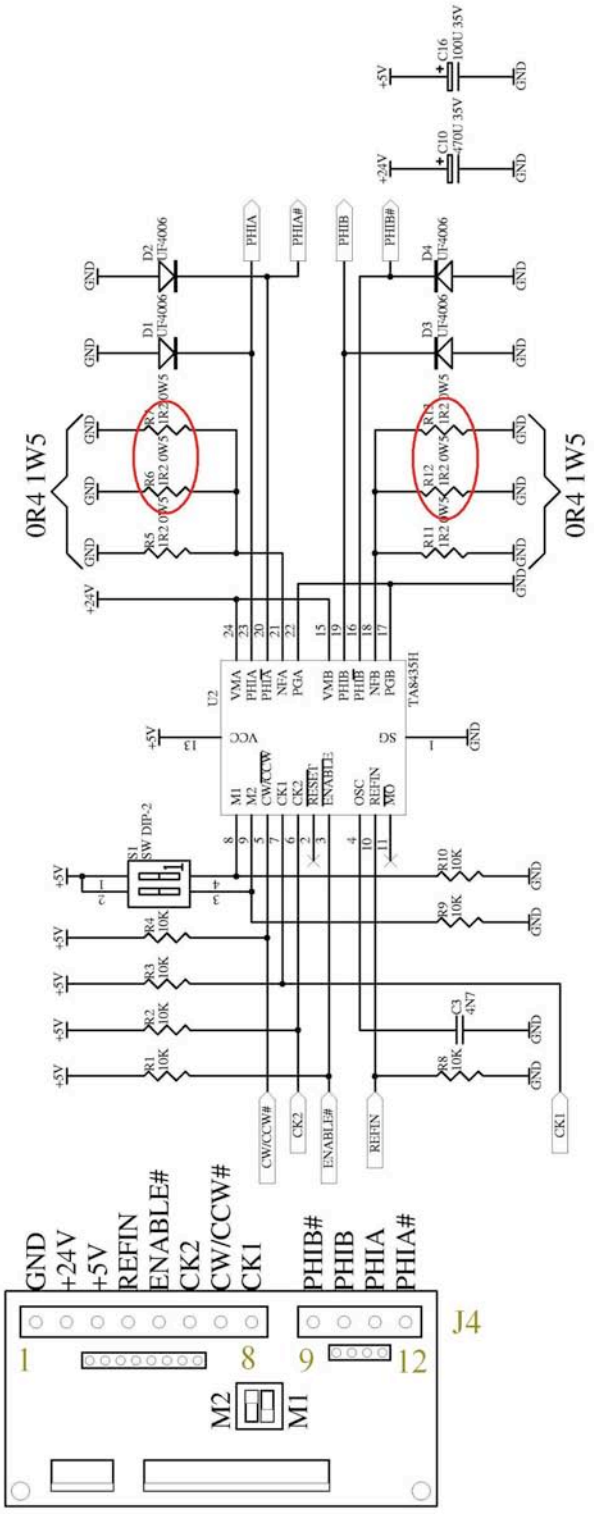

Figura 7 : Schema elettrico scheda driver

È stato necessario modificare il circuito, togliendo due coppie di resistenze (evidenziate in figura 7), per abbassare la corrente di utilizzo e applicare un dissipatore all'integrato.

In figura 7 si notano in basso due interruttori (M1 e M2) che congiuntamente determinano la modalità di pilotaggio del motore a passo.

M1	M2	TIPO DI PILOTAGGIO
L	L	2 Phase.
H	L	1-2 Phase
L	H	W1-2 Phase
H	H	2W1-2 Phase

Nel disegno sottostante sono schematizzati gli I/O della scheda driver.

$\phi\overline{A}$	ϕA	ϕB	$\phi\overline{B}$	CK1	CW/CCW#	CK2	ENABLE#	REFIN	+5V	+24V	GND

- $\phi\overline{A}$ uscita della scheda driver che da PHIA# per la fase A del motore a passo.
- ϕA uscita della scheda driver che da PHIA per la fase A del motore a passo.
- ϕB uscita della scheda driver ce da PHIB per la fase B del motore a passo.
- $\phi\overline{B}$ uscita della scheda driver che da PHIB# per la fase B del motore a passo.
- CK1 un impulso su questo ingresso fa muovere il motore di una posizione.
- CW/CCW ingresso della scheda che determina la direzione di rotazione del motore.
- CK2 ingresso della scheda che congiuntamente con CK1 da il modo di funzionamento del motore.
- ENABLE se alto disabilita la scheda, se basso la abilita.
- REFIN setta la corrente di utilizzo tra due valori. Se basso utilizza poco corrente, se alto ne utilizza di più.
- +5V alimentazione da 5 V per la parte logica della scheda.
- +24V alimentazione da 24 V necessaria per muovere il motore, noi utilizziamo un'alimentazione a +12 V che si dimostra sufficiente per il nostro motore a passo.

3.1.4. Gestione LED

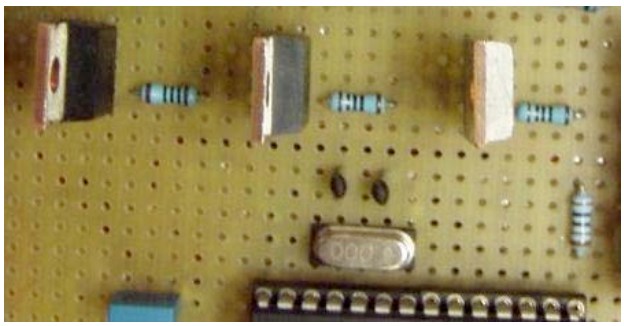

Figura 8: Transistor che comandano i LED.

I tre gruppi di LED sono comandati dal PIC tramite tre MOSFET IRLZ14 (uno per ogni gruppo di LED) che funzionano come interruttori. I transistor sono stati scelti poiché abbastanza veloci da accendere e spegnere i LED senza influenzare negativamente i tempi di esecuzione dei lampeggi.

Sono stati testati diversi tipi di LED: rossi, verdi, infrarossi, ultravioletti. Alla fine la scelta è caduta su LED rossi ad alta intensità. Tali LED presentano una luminosità media di 19000 mcd e una caduta di tensione di 1.9 V con una corrente di 20 mA.

Per rendere visibili i LED anche in situazioni di forte luminosità è stata costruita dal docente una struttura costituita da 19 LED da collegarsi all'uscita LEDA della scheda.

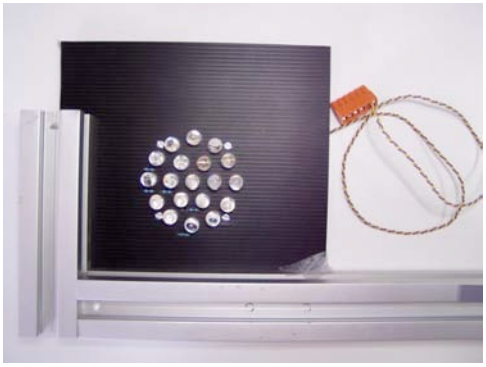

Figura 9: Struttura su cui sono montati i LED.

Nella foto sottostante vediamo l'interruttore che permette ai LED di lampeggiare ad una frequenza fissa senza bisogno di interazione da parte dell'utente. Degli otto interruttori solo quello numerato con 1 è funzionante. Se disattivato (come in figura 10) il PIC funziona normalmente, se attivato inizia il lampeggio.

Figura 10: Interruttore.

3.1.5. Torretta rotante

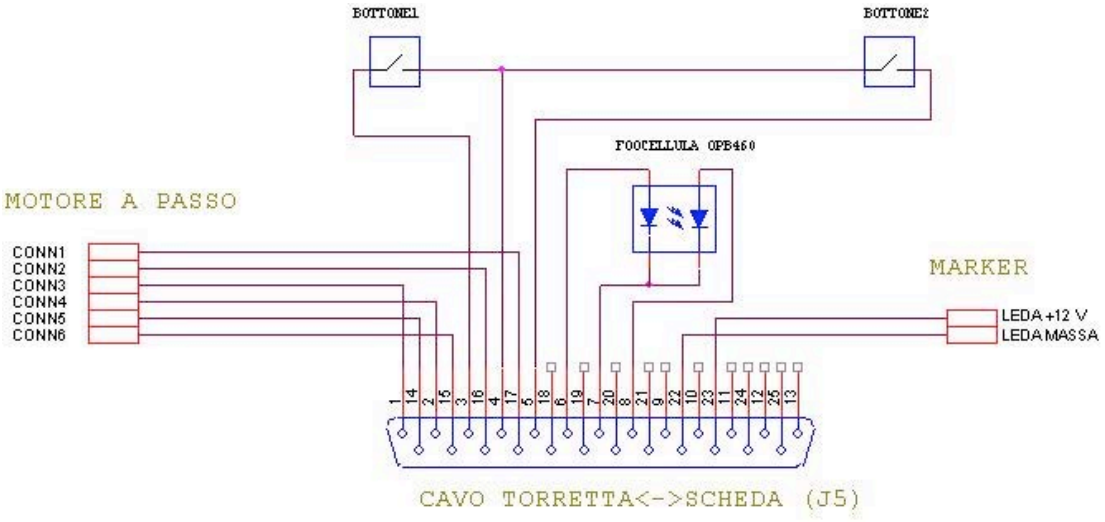

Figura 11: Schema elettrico torretta rotante

Il motore a passo utilizzato alloggia in una struttura in cui sono presenti due finecorsa (FINECORSA1 e FINECORSA2) e una fotocellula per il posizionamento. Il motore è dotato di un'aletta che durante il moto si sposta con il motore e superato un certo angolo va a urtare contro uno dei due finecorsa, a seconda della direzione di rotazione. Questi finecorsa sono collegati al PIC, che rilevando la pressione di uno dei due può agire di conseguenza. La fotocellula, anch'essa collegata al PIC, permette di rilevare il passaggio dell'aletta attaccata al motore e quindi di determinare la posizione dello stesso.

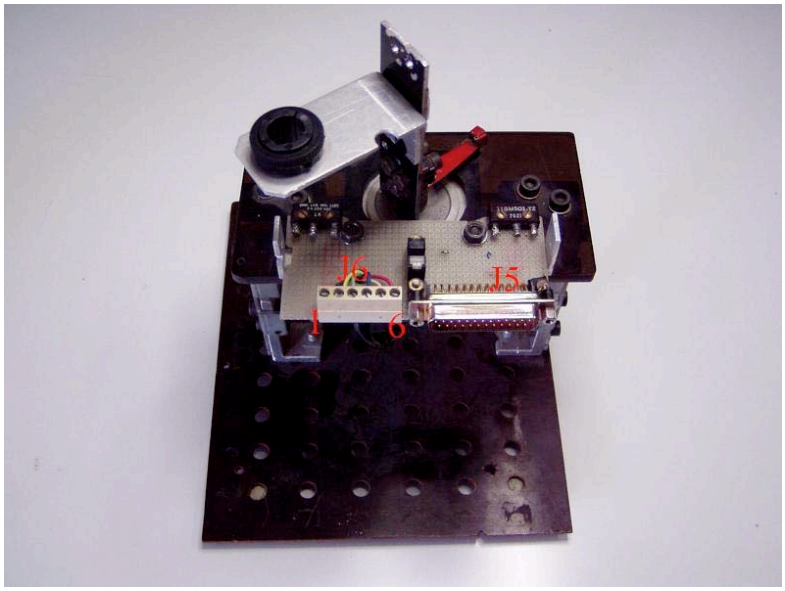

Figura 12: Foto torretta rotante.

Il motore montato sulla torretta conta 200 passi da 1,8 gradi ciascuno. Naturalmente in numero di passi effettivo dipende dal tipo di pilotaggio utilizzato.

Nella foto sottostate si notano i componenti utilizzati per la gestione della fotocellula presenti sulla scheda.

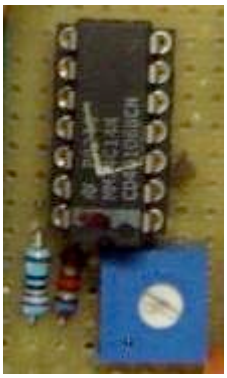

Figura 13 : Foto componenti per gestione fotocellula

Spiccano l'integrato che funge da trigger di Schmitt e il potenziometro per la regolazione della corrente di soglia della fotocellula.

3.1.6. Cablaggi

Per le connessioni fra i vari dispositivi sono stati costruiti degli appositi cavi.

Un cavo consente la connessione tra torretta, scheda logica e scheda driver.

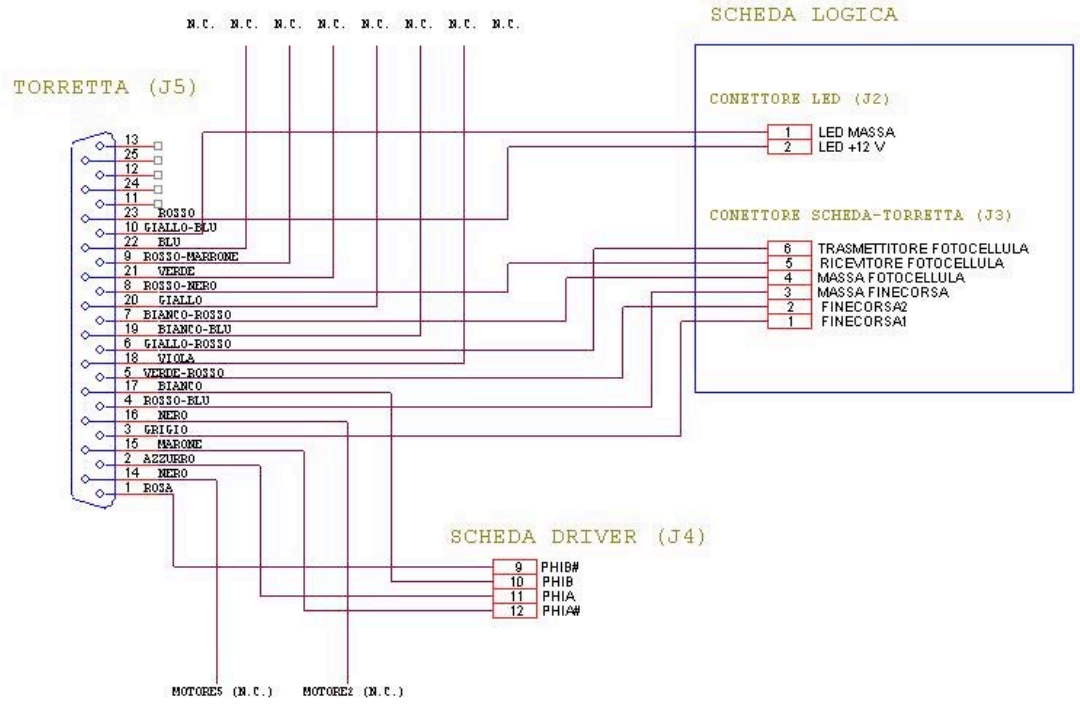

Figura 14: Schema elettrico cavo torretta-scheda.

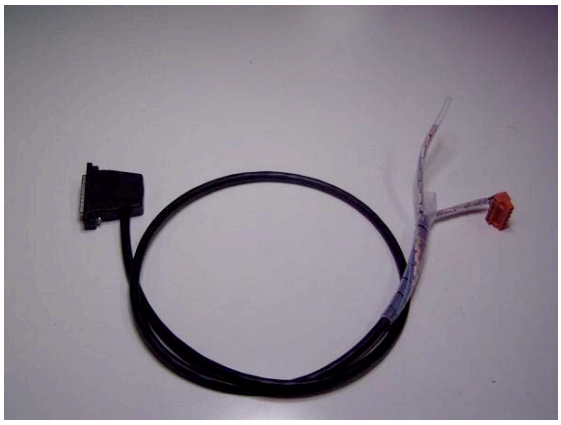

Figura 15: Foto cavo torretta-scheda.

Nella tabella sottostante sono presenti i colori dei fili utilizzati e i relativi contatti sulla torretta e sulla scheda.

CONNET-TORE J5	COLORE FILO	CONNETTORI SCHEDA LOGICA E SCHE-DA DRIVER
1	Rosa	PHIB# (J4-9)
2	Azzurro	PHIA (J4-11)
3	Grigio	Finecorsa 1 (J3-1)
4	Rosso - blu	Finecorsa massa (J3-3)
5	Verde - rosso	Finecorsa 2 (J3-2)
6	Giallo - rosso	Fotocellula trasmettitore (J3-6)
7	Bianco – rosso	Fotocellula massa (J3-4)
8	Rosso- nero	Fotocellula ricevitore (J3-5)
9	Rosso - marrone	N.C.
10	Rosso –marrone	LEDA massa (J2-1)
11	Non usato	Non usato
12	Non usato	Non usato
13	Non usato	Non usato
14	Arancio	Motore 5 (N.C.)
15	Marrone	PHIA# (J4-12)
16	Nero	Motore 2 (N.C.)
17	Bianco	PHIB (J4-10)
18	Viola	N.C.
19	Bianco – blu	N.C.
20	Giallo	N.C.
21	Verde	N.C.
22	Blu	N.C.
23	Rosso	LEDA +12 V (J2-2)

24	Non usato	Non usato
25	Non usato	Non usato

Un altro cavo serve per connettere la scheda logica al robot, fornendo l'alimentazione necessaria e permettendo la comunicazione seriale.

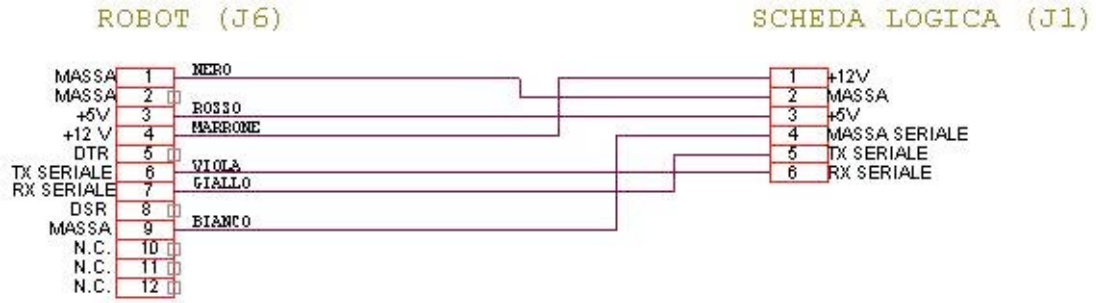

Figura 16: Schema elettrico cavo robot-scheda.

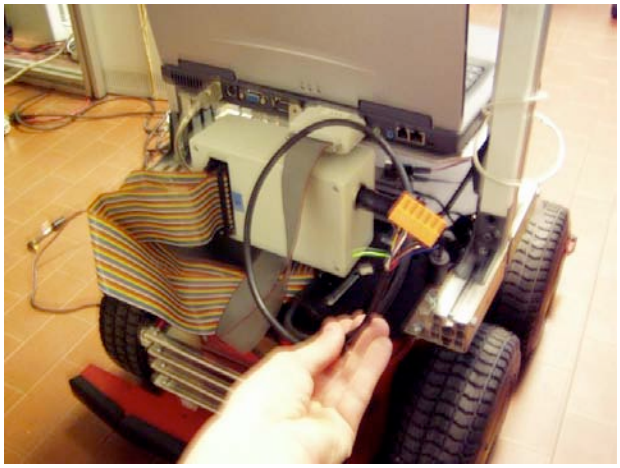

Figura 17: Foto cavo robot-scheda.

Nella tabella che segue sono elencati i contatti e i colori dei file del cavo in questione.

CONNETTORE ROBOT J6	COLORE FILO	CONNETTORE SCHEDA J1
+12 V (J6-4)	Marrone	+ 12 V (J1-1)
Massa (J6-1)	Nero	Massa (J1-2)
+ 5 V (J6-3)	Rosso	+ 5 V (J1-3)
Massa seriale (J6-9)	Bianco	Massa seriale (J1-4)
Ricevitore seriale (J6-7)	Giallo	Trasmettitore seriale (J1-5)
Trasmettitore seriale (J6-6)	Viola	Ricevitore seriale (J1-6)

3.2. Programmazione firmware

Vi sono molti compilatori che permettono di generare software per i PIC più diffusi in commercio, sfruttando un codice sorgente scritto in Assembler piuttosto che C, Basic o altro. Molti di essi sono anche disponibili in versioni gratuite.

In principio si è utilizzato il compilatore PicCLite della HiTech nella sua versione linux (ne esiste anche una per sistemi Windows), poiché utilizzabile gratuitamente se non a scopo di lucro (pur tuttavia con alcune limitazioni).

Sfortunatamente tra le limitazioni presenti nella versione gratuita di questo compilatore figurava l'utilizzo parziale della memoria del PIC, limitazione che ha resa necessaria la migrazione ad un altro compilatore più completo e commerciale. Si è quindi adottato il compilatore C PCWH per Microsoft Windows prodotto dalla Custom Computer Service (Si ringrazia il laboratorio di sensori dell'università di Brescia per averlo reso disponibile).

In fase di sviluppo del firmware si è dovuto riprogrammare il PIC innumerevoli volte. Per semplificare e accelerare tale processo si è fatto ricorso ad un bootloader.

3.2.1. Bootloader

Una volta compilato il codice firmware nasce il problema di mettere il codice generato all'interno del PIC. Per far ciò esistono appositi programmatori hardware (di cui il laboratorio è fornito) che tuttavia hanno l'inconveniente di costringere l'utente a rimuovere il PIC dal suo zoccolo e porlo sul programmatore ogniqualvolta si desideri aggiornare il firmware.

Per ovviare a questo inconveniente si è fatto ricorso ad un bootloader. Questo piccolo programma si occupa della riprogrammazione del PIC tramite interfaccia seriale (naturalmente quest'ultima deve essere stata implementata sul PIC), rendendola possibile senza la rimozione dell'integrato dallo zoccolo e senza il programmatore hardware.

Resta tuttavia la necessità di utilizzare un programmatore hardware per mettere il bootloader sul PIC.

All'avvio del PIC il Bootloader ascolta la porta seriale per 0,2 ms e, nel caso di comunicazione in arrivo, sovrascrive la restante area di memoria di programma del PIC con ciò che viene trasmesso via seriale, quindi esegue il codice appena scritto; se non vi è nessuna comunicazione viene eseguito il codice presente in memoria.

Naturalmente il bootloader riduce lo spazio di memoria disponibile per il firmware, tuttavia le sue dimensioni sono molto ridotte: occupa le prime 4 e le ultime 213 parole della memoria di programma (da 0x0000 a 0x0003, e da 0x1F2A a 0x1FFF).

Per inviare il codice compilato al PIC si fa uso di un programma chiamato downloader che si occupa di verificare la presenza di un bootloader sul PIC e quindi scarica il programma da sovrascrivere.

3.2.2. Downloader

Durante l'utilizzo del compilatore PicCLite della HiTech si è utilizzato il software Downloader per linux disponibile sul sito www.microchip.com. Nel cambiare compilatore e sistema operativo si è passato ad un diverso downloader: il PICbootPlus.

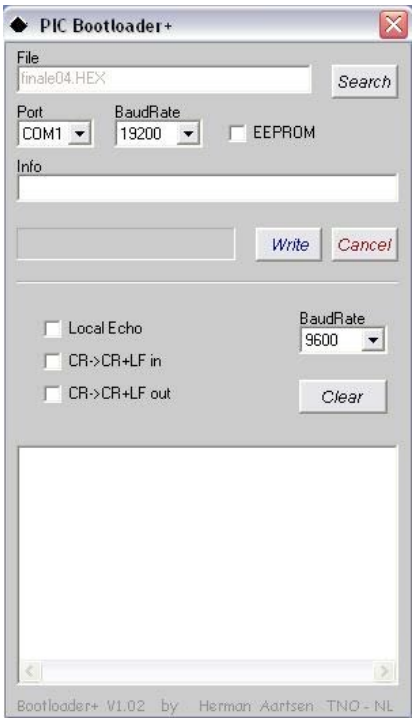

Figura 18: Il downloader PICbootPus

Questo software fruibile gratuitamente si è dimostrato di facile utilizzo. In alto, premendo il pulsante "Search" si può selezionare il file HEX (il file compilato da caricare sul PIC) che si desidera caricare. "Port" permette di scegliere la porta seriale da utilizzare e "BaudRate" la velocità di trasferimento dei dati. Selezionando la voce "EEPROM" si può decidere se scrivere anche la memoria EEPROM. Successivamente è necessario premere il pulsante "Write", e una volta alimentato il PIC il programma viene sovrascritto sul PIC.

Nella parte inferiore è presente una semplice interfaccia di comunicazione seriale che permette di verificare immediatamente le funzionalità del programma caricato.

È da notare come si possa selezionare una velocità di comunicazione seriale diversa dalla velocità utilizzata per programmare il PIC. Infatti la prima è determinata dal programma che viene inserito nella memoria del PIC, mentre la seconda dal bootloader utilizzato.

3.2.3. Discussione listato firmware

Il firmware consta di tre parti principali: un ciclo all'interno della funzione "main" che si occupa di richiedere all'utente l'azione da eseguire, un interrupt dovuto al cambiamento di stato delle porte B dalla 4 alla 7 del PIC e un interrupt di overflow del timer1.

Non appena avviato il programma inizializza le variabili e attiva i due interrupt (INT_RB e INT_TIMER1), successivamente entra in un ciclo infinito che richiede l'operazione desiderata.

L'immissione dei dati è gestita tramite un operatore switch che in base all'azione selezionata richiede tutti i dati necessari, aggiorna le variabili coinvolte e ritorna all'inizio del ciclo richiedendo all'utente una nuova azione.

L'interrupt relativo al timer1 è programmato in maniera tale da generare un overflow ogni millisecondo e costringendo quindi l'esecuzione del suo codice. Questo codice si occupa di controllare uno per volta i tre gruppi di LED per determinare quali debbano essere accesi o spenti e di inviare, se necessario, alla scheda driver l'impulso per far muovere il motore di un passo. Queste decisioni sono prese in base ai dati immessi dall'utente durante il ciclo di immissione dei dati.

Si è scelto di utilizzare un interrupt temporizzato per gestire le funzionalità della scheda poiché esso permette l'immissione di dati anche durante l'esecuzione di altri comandi antecedentemente impartiti,

consentendo quindi un'elevata capacità di interazione. Al fine di garantire la maggior precisione temporale possibile il tempo impiegato dall'esecuzione di questo interrupt è stato reso indipendente delle varie situazioni che possono presentarsi tramite l'inserimento di opportuni ritardi in corrispondenza di ogni istruzione condizionale.

Il secondo interrupt, che si attiva sul cambiamento di stato delle porte B dalla 4 alla 7 del PIC, viene utilizzato dai finecorsa, permettendo di fermare il motore non appena uno di questi viene premuto, indipendentemente dai dati inseriti dall'utente. Questo interrupt viene anche usato da un interruttore posto sulla scheda che una volta attivato blocca l'esecuzione di qualsiasi altra azione e fa lampeggiare i LEDA a una frequenza fissa determinata in fase di compilazione (al momento della stesura della relazione il lampeggio consiste in 33 millisecondi di accensione e 42 millisecondi di spegnimento).

Vi è inoltre la funzione "LeggiNumero" che richiede all'utente l'inserimento di un numero di tre cifre. Tale funzione trova ampio utilizzo durante l'immissione dei dati. È importante notare che nell'immissione di numeri che possono variare da 1 a 999 è sempre necessario utilizzare tre cifre (quindi, per esempio, 10 diventa 010).

La porta seriale viene impostata alla velocità di 9600 baud con 8 bit di dati, senza parità e con un bit di stop. I pin adibiti alla comunicazione sono porta C_6 per la trasmissione e porta C_7 per la ricezione.

Particolare attenzione è stata prestata alla gestione della corrente che circola nel motore a passo. Tramite l'ingresso logico REFIN della scheda driver del motore a passo si abbassa la corrente nel motore mentre quest'ultimo sta fermo, invece, durante la rotazione, quando è richiesta una coppia maggiore, la corrente nel motore viene alzata.

L'attivazione della scheda driver del motore a passo, gestita dall'ingresso ENABLE della stessa, avviene in due modi: o utilizzando l'apposito comando o dando al motore a passo l'ordine di muoversi. A questo punto la scheda rimane attiva fintanto che non si invia il comando di disattivazione, permettendo un minor consumo quando non è più necessario il suo utilizzo.

Prima di passare alla discussione delle funzionalità offerte dalla scheda è bene specificare le differenti modalità di lampeggio: è possibile infatti definire l'alternarsi di diversi tipi di lampeggi diversi per ogni gruppo di LED (per esempio si può programmare un gruppo di LED in maniera che lampeggi alternando due tempistiche diverse: un primo lampeggio con una certa tempistica, il secondo con una tempistica diversa, il terzo nuovamente con la tempistica del primo lampeggio, il quarto con la tempistica del secondo, e così via). Il massimo numero di tipi di lampeggi diversi è quattro per ogni gruppo di LED.

Il firmware è programmato in maniera tale da accettare determinati input, sotto forma di caratteri (anche i numeri sono inviati sotto forma di caratteri), tramite la connessione seriale:

COMANDO	DATI AGGIUNTIVI	AZIONE
a	txxxyyyzzz t = tipi di lampeggi diversi (da 1 a 4) xxx = numero di lampeggi da eseguire (da 001 a 999 a 999 è stato associato un numero di lampeggi infinito). yyy = tempo di accensione di un lampeggio (da 1 a 999). zzz = tempo di spegnimento di un lampeggio (da 1 a 999).	Aziona i LEDA.
b	txxxyyyzzz t = tipi di lampeggi diversi (da 1 a 4) xxx = numero di lampeggi da eseguire (da 001 a 999 a 999 è stato associato un numero di lampeggi infinito). yyy = tempo di accensione di un lampeggio (da 1 a 999). zzz = tempo di spegnimento di un lampeggio (da 1 a 999).	Aziona i LEDB.
c	txxxyyyzzz	Aziona i LEDC.

	t = tipi di lampeggi diversi (da 1 a 4) xxx = numero di lampeggi da eseguire (da 001 a 999 a 999 è stato associato un numero di lampeggi infinito). yyy = tempo di accensione di un lampeggio (da 1 a 999). zzz = tempo di spegnimento di un lampeggio (da 1 a 999).	
d		Accende i LED del gruppo LEDA e ne ferma i lampeggi se in esecuzione
D		Spegne i LED del gruppo LEDA e ne ferma i lampeggi se in esecuzione.
e		Accende i LED del gruppo LEDB e ne ferma i lampeggi se in esecuzione.
E		Spegne i LED del gruppo LEDB e ne ferma i lampeggi se in esecuzione.
f		Accende i LED del gruppo LEDC e ne ferma i lampeggi se in esecuzione.
F		Spegne i LED del gruppo LEDC e ne ferma i lampeggi se in esecuzione.
g		Restituisce la posizione del motore a passo. Se non è stato eseguito il posizionamento restituisce -1.
l		Ferma il motore a passo se è in movimento.
m	xxxyzzz xxx = numero di millisecondi ogni cui viene eseguito un passo (da 1 a 999). y = direzione di rotazione (f/r. f per senso orario, r per senso antiorario). zzz = numero di passi (da 1 a 999. A 999 corrispondono un numero infinito di passi).	Aziona il motore a passo.
M	xxxyzzzzzz xxx = numero di millisecondi ogni cui viene eseguito un passo (da 1 a 999). y = direzione di rotazione (f/r. f per senso orario, r per senso antiorario).	Sposta il motore nella posizione indicata.

	zzzzzz = passo di arrivo (da 0 al numero di passi per giro).	
n		Accende la scheda driver del motore a passo.
N		Spegne la scheda driver del motore a passo.
o	xxxxxx = numero di passi per giro.	Permette di modificare il numero di passi per giro. Utile nel caso si cambi il metodo i pilotaggio del motore.
p		Posiziona il motore a passo in corrispondenza della fotocellula.
s		Spegne i LED di tutti e tre i gruppi, fermandone i lampeggi se in esecuzione. Ferma anche il motore a passo se in movimento.
v		Attiva la modalità verbose.
V		Disattiva la modalità verbose.

Inoltre vengono inviati via seriale alcuni segnali (sempre sotto forma di caratteri) in corrispondeza di determinati eventi:

a	Terminato lampeggio LED del gruppo LEDA.
A	Ricezione del comando di attivazione dei LEDA.
b	Terminato lampeggio LED del gruppo LEDB.
B	Ricezione del comando di attivazione dei LEDC.
c	Terminato lampeggio LED del gruppo LEDB.
C	Ricezione del comando di attivazione dei LEDC.
m	Arresto del motore a passo.
M	Ricezione del comando di attivazione del motore a passo.
s	Spegnimento della scheda driver del motore a passo.
S	Ricezione comando di attivazione della scheda driver del motore a passo.

Un controllo sui dati impedisce l'inserimento di comandi e valori inesistenti o fuori scala. L'introduzione di un valore errato annulla l'inserimento corrente e richiede un nuovo comando.

L'invio di un comando può avvenire in qualsiasi momento, senza che ciò influisca su eventuali operazioni in corso (quindi se sta lampeggiando il gruppo di LEDA posso inviare comandi agli altri gruppi di LED o al motore senza che ciò si rifletta sull'azione corrente). Se viene dato un comando a un apparato che sta già eseguendo un'altra azione l'operazione corrente viene annullata e viene eseguito il nuovo comando.

Gli unici momenti in cui la scheda non accetta comandi sono durante l'esecuzione del comando di posizionamento del motore a passo e quando viene attivato l'interruttore presente sulla scheda che attiva i lampeggi con una tempistica predeterminata (attualmente 33 millisecondi acceso e 42 spento).

Se la modalità “verbose” (vedere comando v/V) non è attivata è presente un timeout di 200 milisecondi sull’inserimento dei dati. Ovvero, se è richiesto l’invio di dati, e essi non giungono entro il tempo stabilito, il programma annulla il comando che si stava inserendo e ritorna a richiedere chiederne uno nuovo.

ESEMPI

“a1100200100” : fa lampeggiare i LEDA 100 volte con tempo di accensione 200 millisencodi e tempo di spegnimento 100 millisecondi.

“c2999010020030040” : fa lampeggiare i LEBC un numero infinito di volte alternando due tempistiche diverse. La prima acceso 10 millisecondi e spento 20 millisecondi, la seconda acceso 30 millisecondi e spento 40 millisecondi.

“m100f050” : ordina al motore a passo di eseguire 50 passi in senso orario ad una velocità di un passo ogni 100 millisecondi.

“o” : spegne la scheda driver del motore a passo, se è accesa.

3.3. Programmazione interfaccia software

Lo sviluppo del software per il controllo dalla scheda attraverso il calcolatore non è stato ancora completato e presenta alcune lacune.

Sono stati sviluppati due programmi, uno utilizzando le librerie Aria, l’altro facendo uso dell’ambiente di sviluppo Saphira. In questo capitolo verrà trattata la prima interfaccia utente, per approfondimenti sulla seconda si rimanda il lettore all’appendice A.

3.3.1. Aria

Aria, acronimo di ActivMedia Robotics Interface for Application, è un software destinato alla programmazione di applicazioni per il controllo dei Robot prodotti dalla ActiveMedia. Presenta numerose librerie che permettono un semplice ed efficace accesso alle funzionalità del robot. Il linguaggio di programmazione utilizzato è il C++.

Il programma per la gestione della scheda è stato suddiviso in due parti: una parte server che deve girare sul calcolatore connesso al robot tramite seriale e una parte client che può girare su un qualsiasi calcolatore connesso in rete con il calcolatore su cui è presente il server. Questo permette di accedere alle funzionalità della scheda anche in remoto.

Entrambe le parti sono state scritte e compilate in ambiente windows, a causa di alcuni problemi di compilazione sotto linux, sicuramente risolvibili configurando in maniera adeguata il compilatore.

3.3.2. Server

Questa parte del programma è stata pensata per fungere da tramite tra il client e la scheda, il suo compito è quindi quello di inviare alla scheda i dati provenienti dal client e di spedire al client i dati ricevuti dalla scheda.

Il programma server deve girare sul computer collegato tramite seriale al robot. Si occupa di stabilire una connessione seriale con il robot e di accettare la connessione TCP del client. Quindi, tramite il robot, trasmette alla scheda ciò che arriva dal client. Tuttavia non è ancora stata sviluppata la parte del programma che si occupa di rinviare al client i dati provenienti dalla scheda.

3.3.3. Client

Il client tenta di stabilire una connessione TCP con il server. Una volta stabilita tale connessione invia al server tutto ciò che viene premuto sulla tastiera del calcolatore. Inoltre visualizza a schermo tutti i dati che arrivano dal server (che tuttavia non invia niente dal momento che non è in grado di ricevere dai dalla scheda). La pressione del tasto ‘q’ permette di chiudere il client.

3.3.4. Consigli per il completamento del software

Il programma server si basa sull'esempio ServerHelloWorld.cpp presente nella cartella "examples" nella directory di istallazione di Aria. Per la connessione al robot si è preso spunto dall'esempio GetAuxExample.cpp, che spiega anche come riuscire a prelevare i dati della porta AUX (cui è collegata la scheda). Un'attenta analisi e rielaborazione di questi due esempi dovrebbe permettere di completare il programma server senza troppe difficoltà.

Per quanto riguarda il client si è utilizzato l'esempio ClientHelloWorld come linea guida. Per semplificare la costruzione di un'interfaccia grafica si consiglia di guardare il programma sviluppato sotto Saphira discusso nell'appendice A. Tale interfaccia grafica è stata sviluppata facendo uso delle librerie grafiche FLTK. Una volta configurato adeguatamente il compilatore linux per l'uso delle stesse non dovrebbe presentare troppe difficoltà l'adattamento dell'interfaccia grafica già costruita.

4. Modalità operative

In questo paragrafo viene mostrato come il marker, torreta e scheda logica siano state montate sul robot, nonché come connettere la scheda al robot per cui è stata progettata o come collegarla direttamente ad un calcolatore tramite porta seriale.

4.1. Assemblaggio componenti

Al fine di rendere operativo l'elaborato svolto si è studiato il modo migliore di montare il marker e la scheda sul robot.

In figura 19 è mostrata la soluzione adottata.

Figura 19 : Posizionamento componenti sul robot.

Si nota che la torretta e il marker hanno subito alcune modifiche.

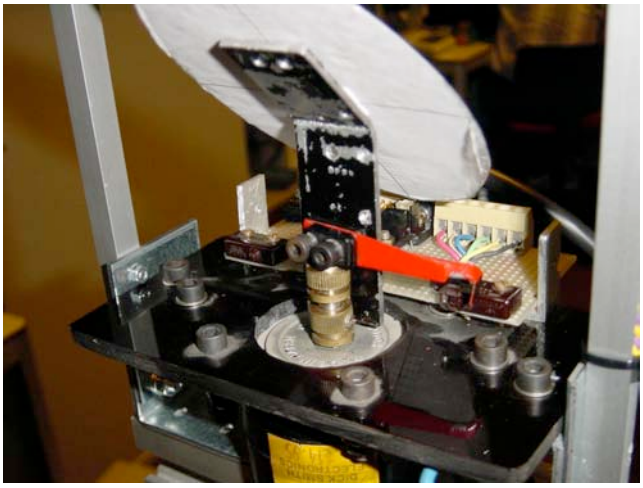

Figura 20 : Parte posteriore torretta.

In figura 20 si vede che l'aletta destinata a urtare i finecorsa, e quindi a impedire al motore di girare oltre un certo angolo, è stata tolta. In questo modo il motore può ruotare indefinitamente su se stesso. Tale scelta è giustificata dal fatte che il marker è stato montato in posizione fissa e la luce viene direzionata da una superficie riflettente sistemata sul motore a passo (si veda figura 21), quindi non vi è più la limitazione alla rotazione imposta dai fili.

Figura 21 : Parte frontale torretta.

4.2. Collegamento della scheda al robot "Morgul"

Partendo dall'immagine sottostante che mostra la scheda senza connessioni esterne vediamo come connetterla al robot.

All'inizio la scheda si presenta come in figura 22.

Figura 22: Scheda inserita nella sua scatola.

Il robot è dotato di un cavo creato apposta per la connessione alla scheda.

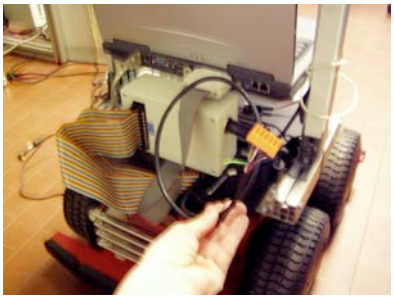

Figura 23: Cavo robot-scheda.

Tale cavo va connesso all'ingresso J1 in alto a destra in figura 22, e fornisce l'alimentazione alla scheda e la connessione seriale.

Per connettere la torretta alla scheda si utilizza l'apposito cavo.

Figura 24 : Cavo scheda-torretta.

Da un'estremità è presente una porta a 25 poli che va connessa all'ingresso J5 della torretta. L'altra terminazione presenta diversi connettori. Uno di questi (l'unico connettore in cui tutti e sei i pin sono utilizzati) va collegato all'ingresso J3 della scheda logica e ha il compito di fornire i collegamenti per i finecorsa e la fotocellula. Un secondo (il connettore a sei pin di cui solo due sono utilizzati) viene collegato alla porta J2 della scheda logica e si occupa dell'alimentazione dei LED. Rimangono quattro fili che vanno connessi all'ingresso J4 della scheda driver e servono a far muovere il motore a passo.

La scheda risulterà quindi come mostrata nelle foto sottostante.

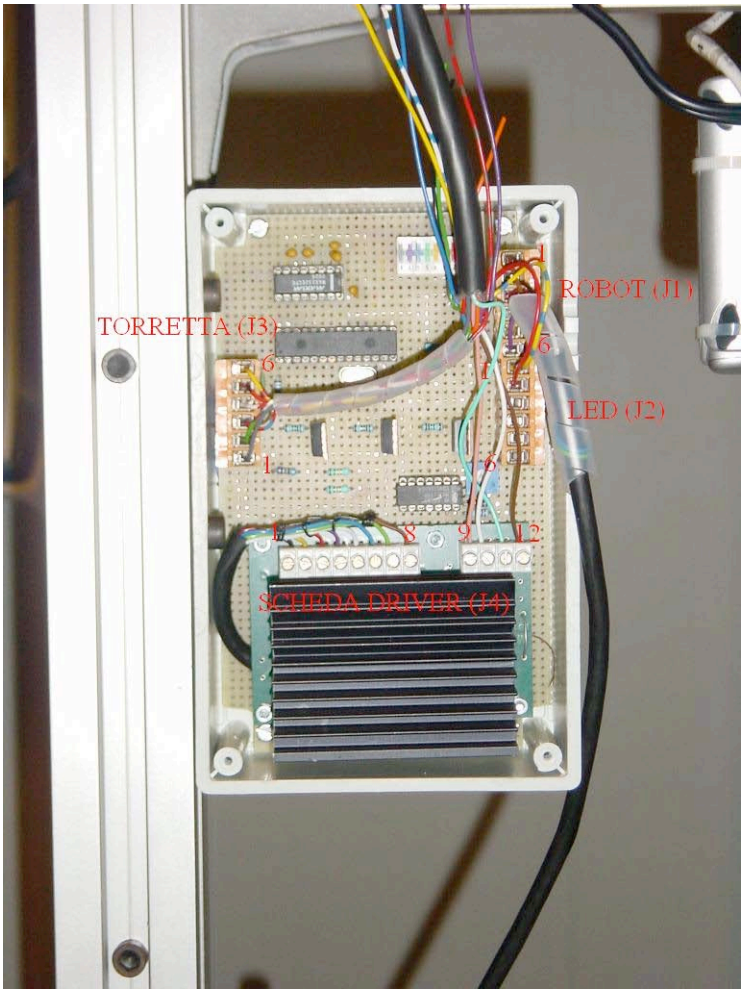

Figura 25 : Scheda connessa alle periferiche.

Nel caso si sostituisca il motore bisogna stare particolarmente attenti a come si connette il motore a passo alla scheda driver, poiché se vengono invertiti i contatti il motore non funziona o si muove nella direzione opposta a quella voluta. Per verificare che la connessione sia corretta accertarsi che quando viene dato il comando di movimento del motore a passo (quindi il comando "m") alla scheda con direzione "f" il motore si muova in senso orario. (per esempio si può inviare alla scheda la stringa "m100f010" e verificare che il motore a passo compia 10 passi in senso orario).

Una volta eseguite tutte le connessioni si può comandare la scheda tramite l'apposito software discusso nel capitolo precedente.

4.3. Collegamento della scheda ad un calcolatore

Per connettere la scheda ad un calcolatore la procedura è identica alla precedente per quanto riguarda la connessione alla torretta e al marker.

All'ingressi J1 (in alto a destra in figura 22) bisogna quindi connettere nei primi tre pin le alimentazioni. A tale scopo si può utilizzare il cavo costruito sfruttando l'integrato 7805. Il connettore arancione di tale cavo va connesso ai primi tre pin dell'ingresso J1 della scheda, mentre i fili dall'altra estremità devono essere collegati ad un'alimentazione da 12V (una batteria del robot, per esempio). Inoltre è presente un interruttore per accendere e spegnere la scheda (molto utile in fase di programmazione del PIC).

Figura 26: Cavo di alimentazione.

I successivi tre pin della porta J1 della scheda (quelli dedicati alla comunicazione seriale) vanno connessi alla porta seriale del calcolatore. Si può sfruttare il cavo già costruito il cui connettore arancione va collegato agli ultimi tre pin dell'ingresso J1 della scheda, mentre la presa gialla deve essere connessa alla porta seriale del calcolatore.

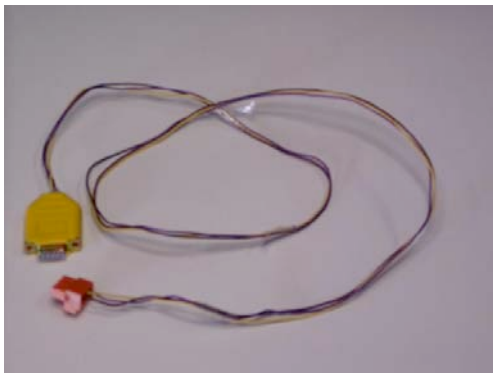

Figura 27: Cavo seriale scheda-calcolatore.

A questo punto si può accedere alle funzionalità della scheda utilizzando un qualsiasi programma di comunicazione seriale (per esempio HyperTerminal per Windows o Minicom per linux) settando la velocità della porta seriale a 9600 baud, con 8 bit di dati, senza parità e con un bit di stop.

4.4. Avvertenze

Nel settare la velocità del motore, si consiglia di non scendere sotto i 3 millisecondi per passo poiché si rischia che il motore nel girare perda alcuni passi o si blocchi.

Un'altra importante considerazione riguarda i due finecorsa. Tali finecorsa sono normalmente chiusi, quindi se non si connette la scheda alla torretta il PIC "pensa" che siano stati premuti e attiva il relativo interrupt, impedendo alla scheda di funzionare correttamente. Alternativamente si possono connettere le uscite presenti sulla scheda relative ai finecorsa a massa, in modo da chiudere il circuito. A tale scopo è stato costruito un semplice connettore da inserirsi nei pin che vanno da 1 a 3 della porta J3 della scheda.

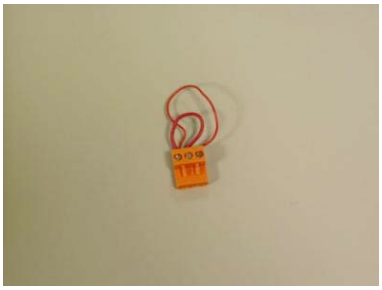

Figura 28 : Connettore per il cortocircuitamento dei finecorsa.

Un discorso analogo vale per l'interruttore montato sulla scheda per eseguire lampeggi ad una frequenza prefissata. Se al momento dell'accensione della scheda tale interruttore è attivato risulta impossibile l'invio di comandi.

Infine, nello sviluppo di un software per la gestione delle funzionalità della scheda, è opportuno tener presente che la scheda non è in grado di accettare un flusso di dati troppo elevato. Ciò è dovuto al fatto che durante l'esecuzione dell'interrupt del timer1 (vedere il paragrafo 3.2.3) i dati in arrivo non possono essere acquisiti e vengono quindi inseriti nel buffer del PIC. Tuttavia tale buffer è piuttosto piccolo e si riempie in fretta, con conseguente perdita di dati. Per ovviare a questo problema è sufficiente inserire un breve ritardo dopo l'invio di ogni dato e non abilitare l'opzione verbose che invia numerose stringhe lungo la porta seriale.

5. Conclusioni e sviluppi futuri

L'elaborato può considerarsi concluso per quanto riguarda gli aspetti hardware e firmware. Invece, per la parte software, risulta essenziale, nel breve periodo, lo sviluppo di un'interfaccia grafica completa e funzionale che sfrutti le librerie Aria.

6. Appendice A

Questa appendice prende brevemente in esame l'interfaccia grafica programmata sotto linux con Saphira, un ambiente di sviluppo per la programmazione di robot ActivMedia che si appoggia sul già menzionato Aria (capitolo 3.3.1).

Il motivo per cui si tratta l'argomento in appendice è che tale sistema di sviluppo è stato abbandonato dai sui produttori, rendendo più significativo la creazione di un programma in Aria.

Il programma si poggia sulle librerie grafiche FLTK (Fast Light Tool Kit), viene compilato sotto forma di libreria e necessita la presenza di Saphira installato sul calcolatore.

Una volta compilato il programma, sfruttando il "Makefile" incluso, bisogna far partire Saphira e digitare "loadlib marker" nella finestra di comando.

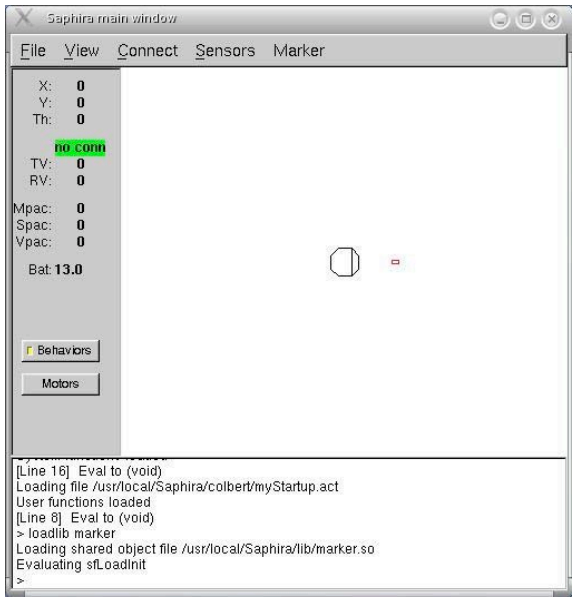

Figura 29 : Saphira.

Apparirà quindi un nuovo menù in alto a destra denominato “Marker”. Una volta selezionata l’unica voce presente si aprirà l’interfaccia grafica per il controllo della scheda progettata.

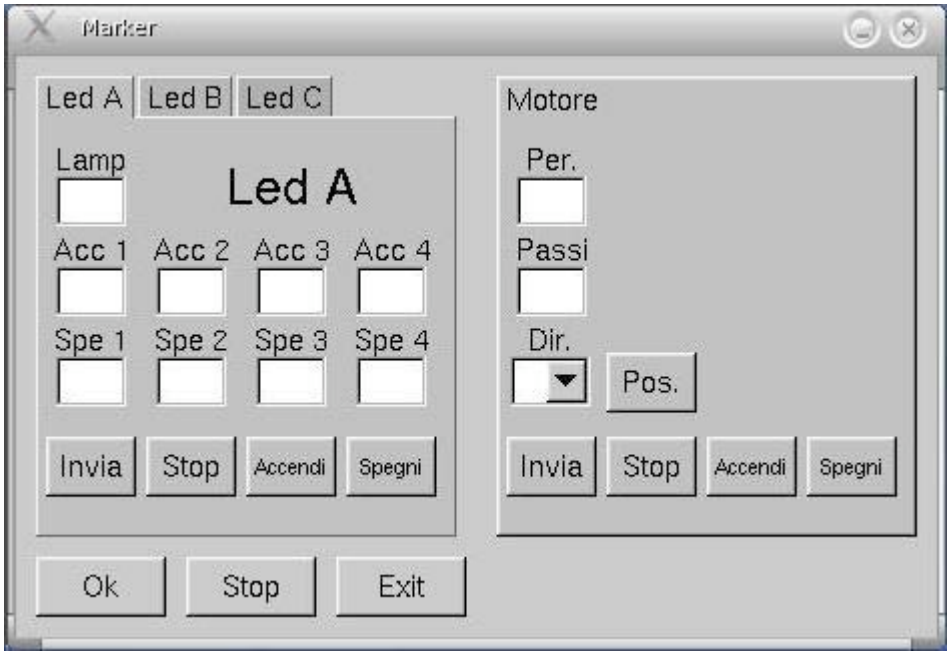

Figura 30 : L’interfaccia per il controllo della scheda.

L’interfaccia è divisa in due parti, quella sinistra dedicata ai LED e quella destra dedicata al motore.

Partendo dalla sezione relativa alla gestione dei LED notiamo in alto la possibilità di scegliere per quale LED immettere i dati (LedA LedB, LedC). La voce “Lamp” richiede il numero di lampeggi da eseguire, appena sotto si inseriscono i tempi di accensione e di spegnimento. Si notano quattro colonne per l’inserimento di tali tempi. L’immissione di dati in qualsiasi di esse determina il numero di tipi di lampeggi da effettuare (se si utilizza solo “Acc 1” e “Spe 1” si avrà un solo tipo di lampeggio, se invece si riempiono anche i campi relativi a “Acc 2” e “Spe 2” avremo l’alternarsi di due tipi di lampeggi diversi). I pulsanti appena sotto servono, nell’ordine, per inviare alla scheda i dati inseriti riguardanti il

LED selezionato, per fermare i lampeggi del LED selezionato e per accendere o spegnere il LED selezionato.

Nella sezione riguardante il motore è presente una casella “Per.” in cui inserire i passi per millisecondo, una casella “Passi” per il numero di passi da compiere e un menù “Dir.” In cui scegliere la direzione di rotazione. Il pulsante “Invia” spedisce alla scheda i dati relativi al motore, “Stop” ferma il motore se è in movimento, “Accendi” e “Spegni” servono rispettivamente per accendere e spegnere la scheda driver e “Pos.” attiva l’operazione di posizionamento del motore a passo in corrispondenza della fotocellula.

Vi sono tre pulsanti all’estremità inferiore dell’interfaccia: “Ok” invia alla scheda i dati relativi a tutti e tre i LED e al motore, “Stop” ferma tutti i LED e il motore e “Exit” chiude l’interfaccia grafica.

Questo programma non può tuttavia considerarsi completo in quanto non è in grado di ricevere dati dalla scheda.

7. Appendice B

Di seguito sono mostrati gli schemi elettrici (spesso ingranditi) del progetto. Vi è inoltre un elenco dei connettori presenti.

CONNETTORE	POSIZIONE	UTILIZZO
J1	Scheda logica	Connettore da collegarsi al robot. Fornisce le alimentazioni alla scheda logica e la connessione seriale tra scheda e robot.
J2	Scheda logica	Connettore da collegarsi al marker. Fornisce l’uscita dei LED.
J3	Scheda logica	Connettore da collegarsi alla torretta rotante. Contiene le connessioni per la gestione dei due finecorsa e della fotocellula montati sulla torretta.
J4	Scheda driver	I pin che vanno da 1 a 8 vanno collegati alla scheda logica e forniscono l’alimentazione per la scheda driver e i segnali logici per il movimento del motore a passo. I pin dal 9 al 12 sono invece le uscite da connettersi direttamente alle due fasi del motore a passo.
J5	Torretta rotante	Connettore a 25 poli che contiene i collegamenti per motore a passo, marker, finecorsa e fotocellula.
J6	Robot	Connettore presente nella parte posteriore del robot. Fornisce le alimentazioni alla scheda e permette la comunicazione seriale.

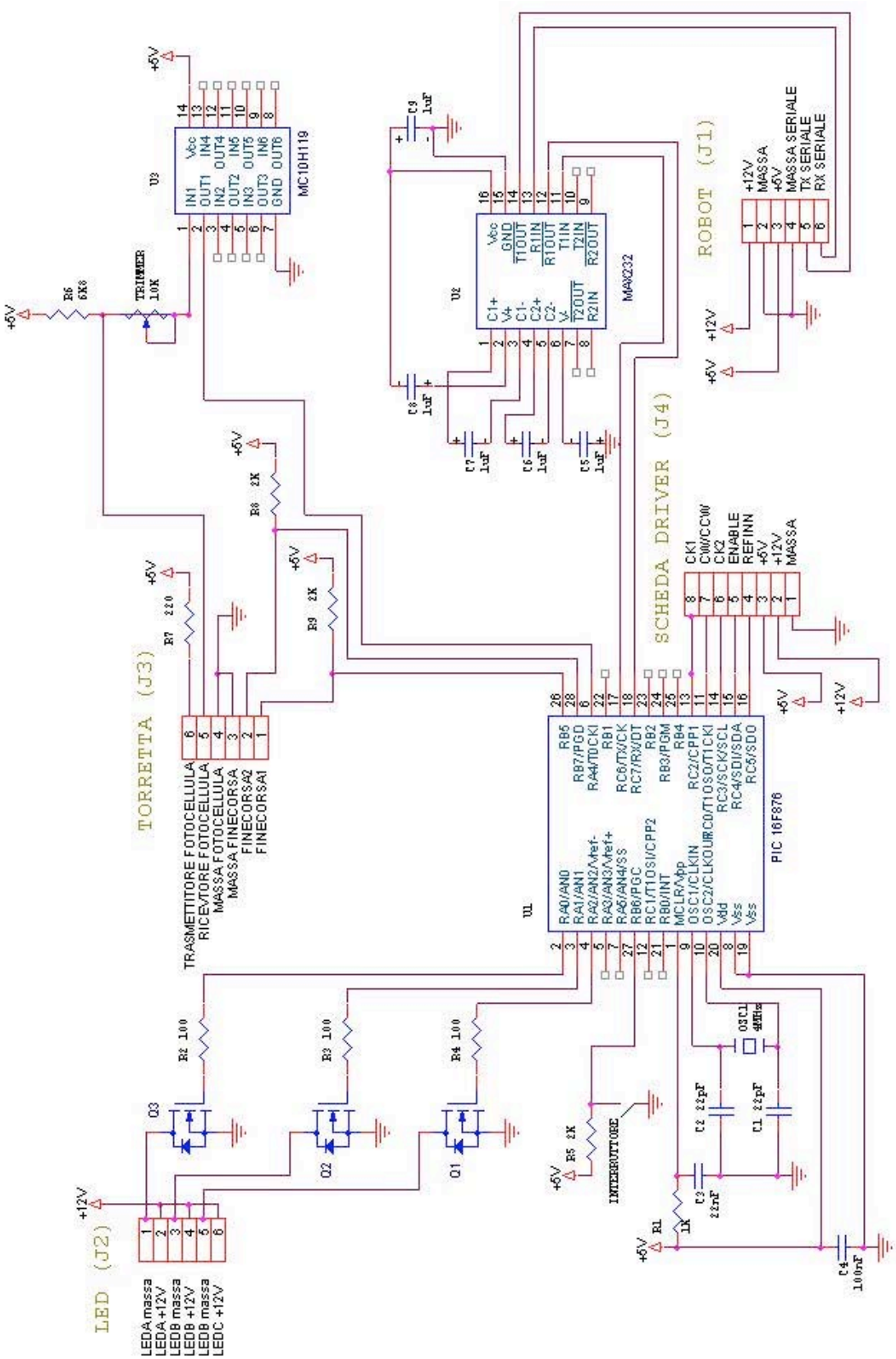

Figura 31 : Schema elettrico scheda logica.

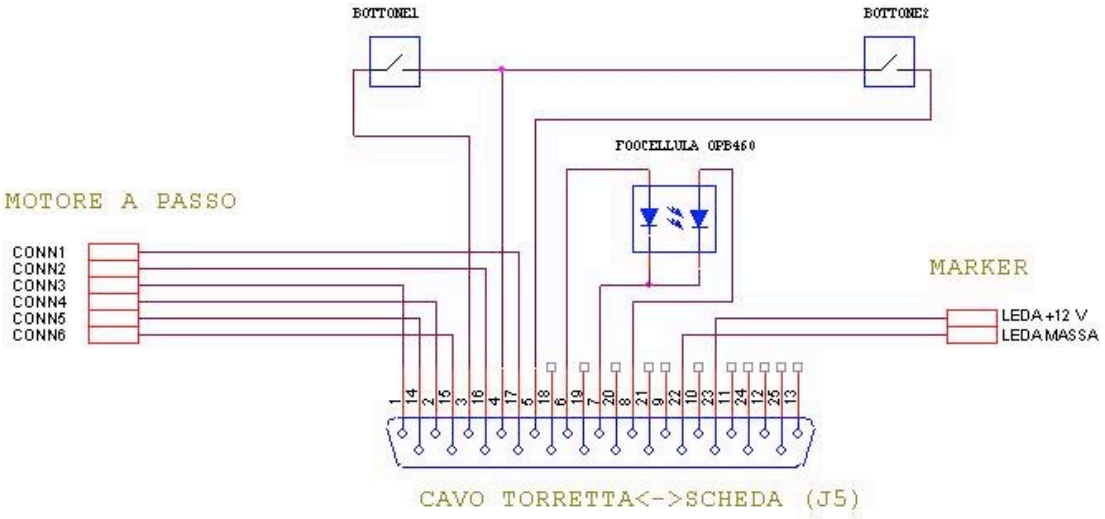

Figura 32 : Schema elettrico torretta rotante.

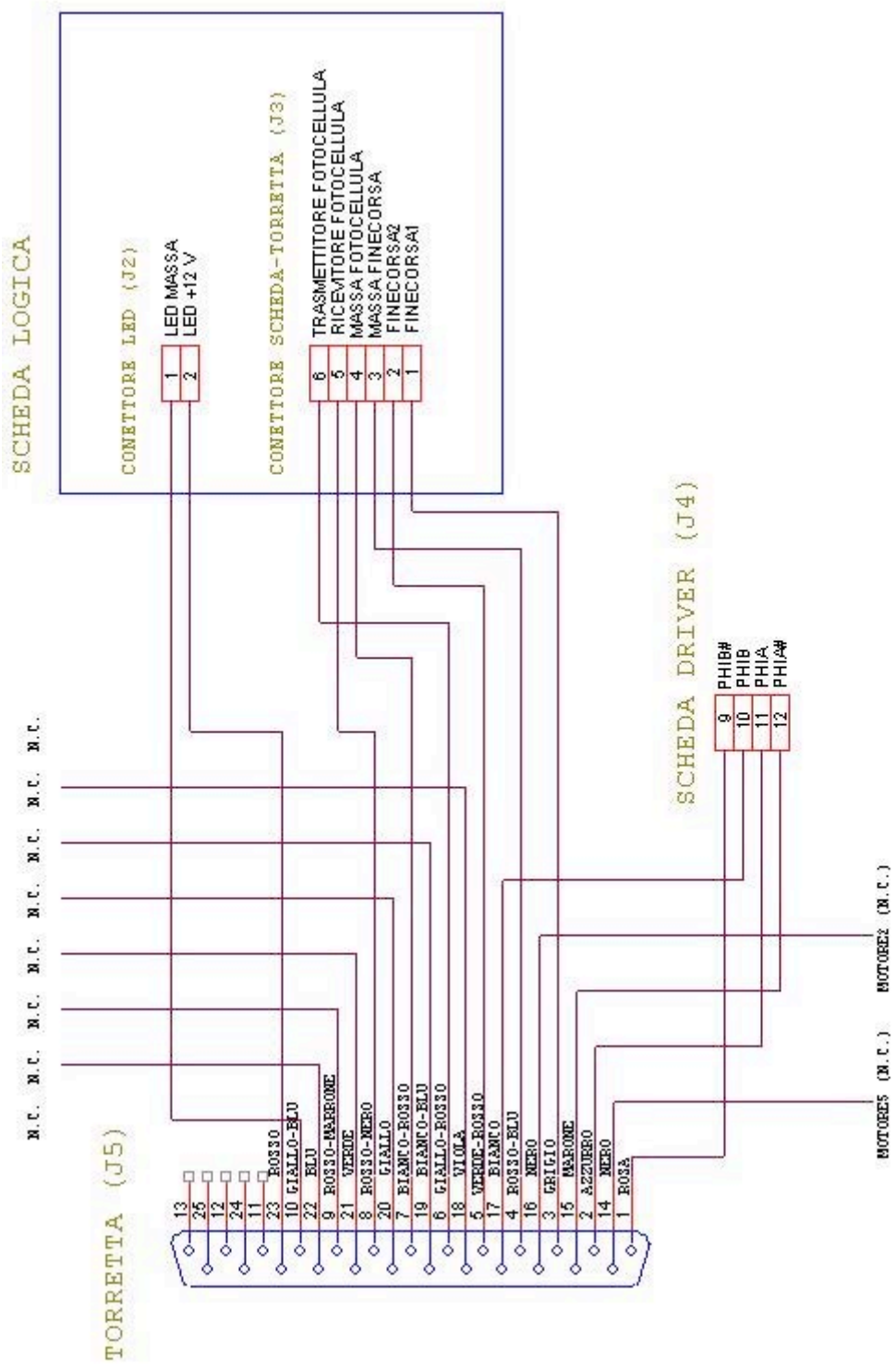

Figura 33 : Schema elettrico cavo Torretta-scheda.

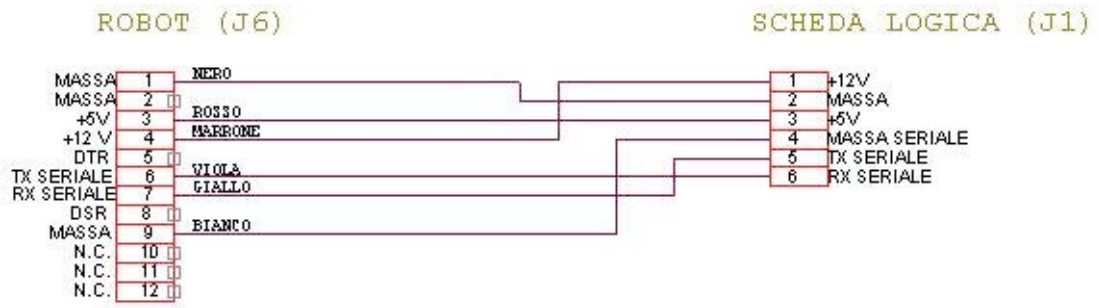

Figura 34 : Schema elettrico cavo robot-scheda.

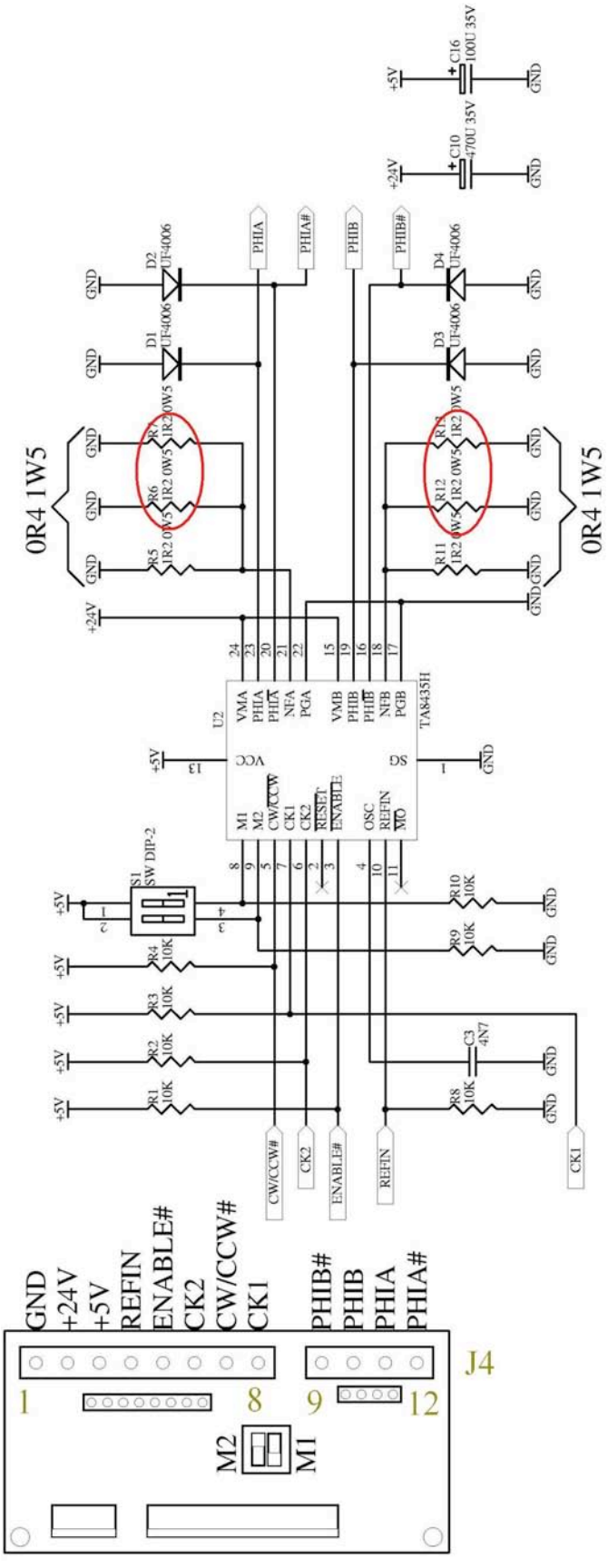
0R4 1W5
0R4 1W5
TA8435H
U2
VMA
PHIA
PHIA#
NFA
PGA
VMB
PHIB
PHIB#
NFB
PGB
VCC
SG
M1
M2
CW/CCW
CK1
CK2
RESET
ENABLE
OSC
REFIN
MO
S1
SW DIP-2
C16
100U 35V
C10
470U 35V
C3
4N7
D1
D2
D3
D4
UF4006
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
10K
IR2 0W5
+5V
+24V
GND
CW/CCW#
CK2
ENABLE#
REFIN
CK1
PHIB#
PHIB
PHIA
PHIA#
J4
1
8
9
12
M2
M1

Figura 35 : Schema elettrico scheda driver

Bibliografia

Buona parte della documentazione utilizzata è stata fornita dal docente sotto forma di fotocopie.

[1] PIC16F87X Datasheet ,Microchip.

[2] PICmicro Mid-Range MCU Family Reference Manual, Microchip.

[3] Max232 Datasheet, Texas Instrument Incorporated.

[4] MM74C14 Hex Schmitt Trigger Datasheet, Fairchild.

[5] TA8435H pwm chopper type bipolar stepping motor driver, Toshiba.

[6] Photologic Slotted Optical Switches (OPB460) Datasheet, Optek Technology, Inc.

[7] Pioneer 3 OperationsManual, Activemedia.

[8] FTLK Manual, Bill Spitzak.

[9] Aria Reference Manual, Activemedia.

[10] Saphira Software Manual, Kurt G. Konolige SRI International.

[11] C Compiler Reference Manual, Custom Computer Services Incorporated

Sono inoltre state eseguite numerose ricerche su internet. Presetiamo di seguito alcuni sito di particolare interesse.

[12] www.microchip.com sito della casa produttrice dei PIC.

[13] www.rs-components.it sito del rivenditore utilizzato dal laboratorio

[14] robots.activmedia.com sito della casa produttrice del robot. È presente anche un utile forum di discussione riguardate Aria.

[15] Documentazione sito che contiene varia documenazione riguardante il laboratorio d robotica.

[16] Fiser Tek sito di robotica amatoriale che contiene utili consigli per imparare a programmare un PIC

[17] www.ccsinfo.com Sito del produttore del compilatore utilizzato.

Indice