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)

CD robot

Elaborato di esame di: **Rizzi Mauro**
Toninelli Paolo

Consegnato il:
??/06/2003

Sommario

Lo scopo del nostro elaborato è di realizzare un robot utilizzando due servomotori per la movimentazione, uno per lo spostamento dei sensori di distanza, un sensore di calore, e dei CD per la struttura meccanica. Il tutto è gestito da un microcontrollore PIC16F876. Abbiamo scritto il programma tramite il linguaggio assembler dedicato che realizza i comportamenti desiderati.

1. Introduzione

Il nostro scopo era quello di realizzare un piccolo robot che si muovesse senza urtare gli ostacoli e reagisse alle fonti di calore attivando un buzzer nel caso un corpo caldo entrasse nel suo raggio d'azione. Il materiale inizialmente a nostra disposizione era il seguente:

- molti CD vecchi da utilizzare per lo scheletro del robot e per le ruote;
- 2 servomotori modificati, della parallax che ruotano a velocità dipendenti dal duty cycle del PWM che gli viene fornito;
- 1 servomotore della MAXX che si posiziona a diverse angolazioni di un anoglo piatto a seconda del duty cycle del PWM del segnale in ingresso;
- 2 sensori di distanza SHARP GP2D02 digitali;
- 1 sensore di calore (human detector).

Figura 1: servo motore, servo motore a rotazione continua e sensore di distanza

2. I problemi affrontati

I problemi affrontati sono stati molteplici, ma andiamo con ordine.

2.1. La struttura meccanica

La nostra idea iniziale era di realizzare il corpo del robot con tre strati di CD sovrapposti a formare un cilindro. Quello in fondo dedicato a contenere le batterie, il centrale dedicato ai motori per le ruote e il terzo per l'elettronica di controllo. Nel centro avremmo posizionato il motore per lo spostamento dei sensori di posizione, in modo che il robot, facendoli ruotare, potesse guardarsi attorno ed accorgersi dell'eventuale presenza di ostacoli. Infine anche le ruote sarebbero state realizzate con dei CD in verticale. Una prima versione della struttura meccanica è visibile nella figura sottostante.

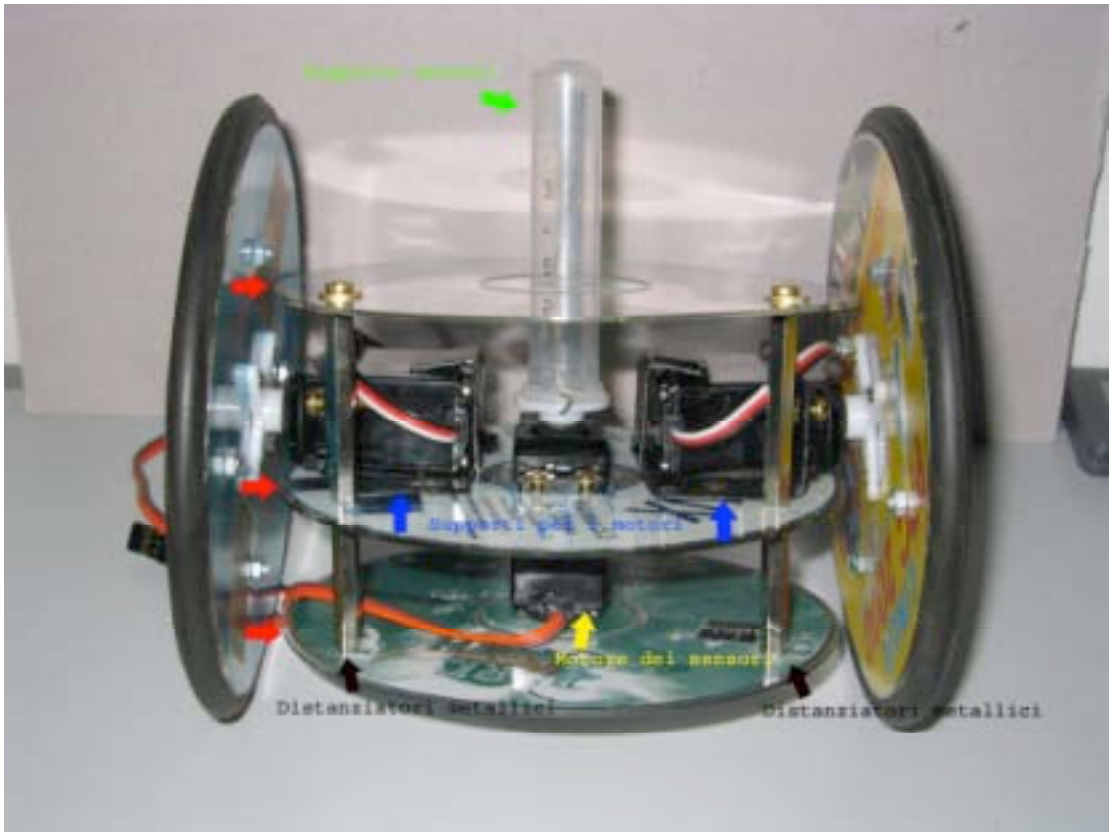

Figura 2: prima versione della struttura meccanica

Le frecce rosse indicano i tre CD che creano il corpo del robot. Le frecce blu i supporti metallici dei motori ricoperti da nastro isolante nero. Infine, la freccia verde indica una siringa usata per prolungare l'albero del motore centrale e permettere ai sensori di distanza di poter guardare al di sopra delle ruote.

Questa struttura presentava numerosi problemi: in primo luogo i supporti dei motori non riuscivano a fornire una rigidità strutturale sufficiente all'asse delle ruote. Le ruote stesse presentavano troppe e antiestetiche viti, e a volte la guarnizione di plastica entrava nello spazio tra i due CD a causa delle viti posizionate troppo verso il centro del disco.

Per sostenere e spostare i sensori abbiano usato un pezzo di metallo forato, lo abbiamo piegato, e ci abbiamo avvitato i sensori. Questo supporto andava poi ancorato all'albero del motore centrale con una vite che andava ad inserirsi all'interno dell'estremità superiore della siringa. Nella foto possiamo notare tutti i particolari.

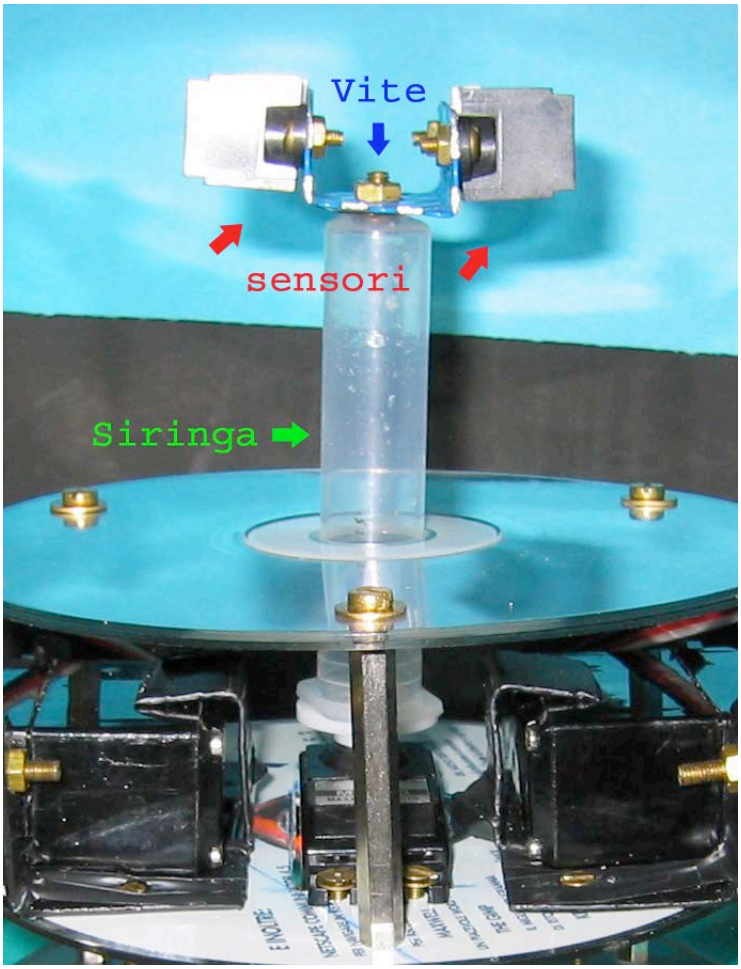

Figura 3: prima versione del supporto per i sensori GP2D02

Anche questo supporto una volta collegati i sensori alla scheda ci ha creato problemi perché la vite non riusciva a far si che i sensori durante il loro movimento trascinassero i cavi di collegamento. Di conseguenza restavano bloccati in una posizione e non si muovevano correttamente.

2.2. L'elettronica di controllo

Per quanto riguarda l'elettronica i problemi da risolvere erano i seguenti:

- Pilotare i 3 servomotori generando tre PWM dal duty cycle variabile e con una frequenza di 333 Hz;
- Leggere i sensori digitali di distanza;
- Leggere il sensore di calore (analogico)
- Leggere il livello della batteria e segnalare quando scarica

- Avere una sezione di alimentazione tale da fornire la corrente necessaria al buon funzionamento dei motori e al microcontrollore.
- Programmare il microcontrollore con le funzioni desiderate senza la necessità di rimuoverlo dalla PCB per effettuare gli aggiornamenti.

Abbiamo cercato diversi integrati che ci consentissero di fare agevolmente tutto quello di cui avevamo bisogno. Tra di questi anche un misterioso chip della Mitsubishi (M6642E) di cui abbiamo trovato il datasheet, ma che non siamo riusciti a capire se esistesse veramente e se fosse in vendita. Pensavamo ad una accoppiata PIC16F84 (per ingressi e uscite analogico/digitali) e M6642 (per i PWM). Un'altra soluzione potevano essere i chip della ST (ST52) o quelli della AT (ATmega8L).

Alla fine ci siamo accontentati di un PIC16F876. In primo luogo per la disponibilità a basso costo, in secondo luogo per la semplicità dell'instruction set. Avremmo utilizzato le due uscite PWM dell'integrato per pilotare i motori e avremmo generato la terza via software. Il chip ha inoltre 5 ingressi analogici con convertitore analogico/digitale a 10 bit e altre 16 porte di I/O, 4 delle quali (RB4-RB7) sono interrupt-on-change. Questa è sicuramente una funzione molto potente, ma ci ha creato non pochi problemi in fase di programazzione.

Per permetterci di scrivere tutte le volte che volevamo un nuovo programma all'interno della memoria del microprocessore abbiamo inserito all'interno dello stesso un bootloader che ci permettesse di programmare il PIC dal PC tramite il semplice collegamento seriale, senza bisogno, quindi di togliere il chip dallo zoccolo e inserirlo nel programmatore e viceversa.

A questo punto abbiamo disegnato il circuito posizionando i vari componenti in maniera comoda. Abbiamo chiesto e ottenuto il permesso di farci realizzare la scheda nei laboratori dell'università.

Una prima versione della scheda destinata ad ospitare l'elettronica di controllo è visibile nella foto seguente.

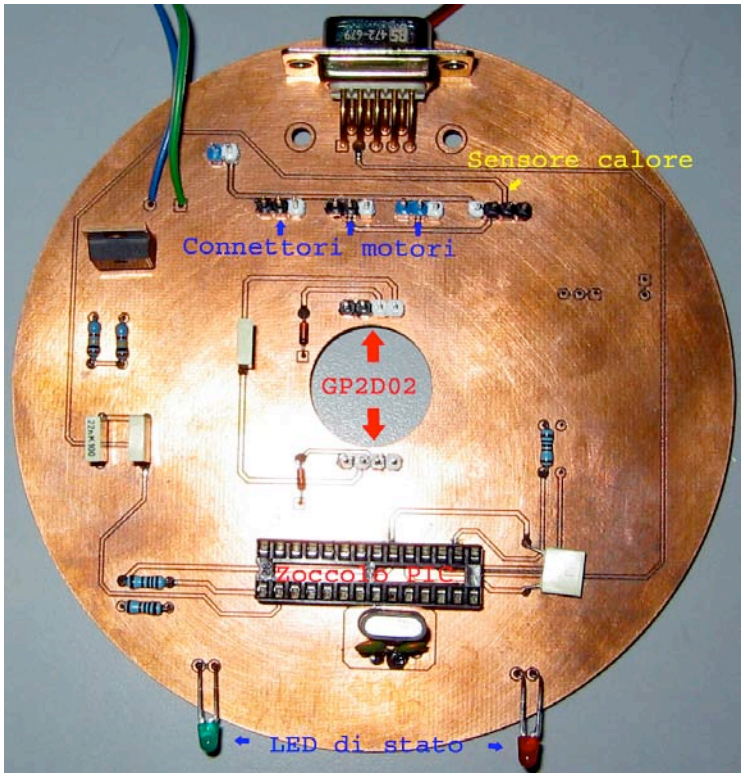

Figura 4: prima versione della scheda di controllo

Questa soluzione ci ha creato diversi problemi. Al primo tentativo sembrava funzionasse tutto, ma dopo un brevissimo periodo di utilizzo abbiamo incontrato difficoltà (disturbi, falsi contatti e interferenze varie) tali da costringerci a rifare tutto il circuito su una mille-fori.

Abbiamo poi misurato i consumi dei motori a varie velocità di rotazione e abbiamo scelto il regolatore di tensione (LM7805) che potesse fornire una corrente superiore a quella richiesta e stabilizzasse la tensione a 5V.

I file con i grafici dell'assorbimento di corrente in funzione del PWM e i commenti sulle prove sono disponibili qui: grafici_assorbimento, commenti_prove.

I

3. Le soluzioni adottate

Per risolvere tutti i problemi che sono sorti durante la realizzazione del robot abbiamo dovuto usare l'immaginazione, l'inventiva e a volte scendere a compromessi fra quello che ci saremmo aspettati e quello che si riusciva a realizzare.

3.1. La struttura meccanica

Per realizzare il corpo del nostro robot abbiano innanzitutto realizzato un buco centrale destinato a contenere il motore per lo spostamento dei sensori

Figura 5: realizzazione del foro centrale

Lo scheletro del robot è composto da tre strati di CD sovrapposti e uniti da separatori metallici a formare un cilindro. Ogni piano è composto da due CD incollati in modo da

irrobustire la struttura, altrimenti troppo fragile del CD. Nel piano centrale è posizionato verticalmente il motore per lo spostamento dei sensori, e orizzontalmente i due motori delle ruote incollati ai CD stessi. Sulla parte frontale abbiamo realizzato un buco rettangolare per consentire l'inserimento del sensore di calore. Il piano superiore ospita l'elettronica di controllo. (vedi figura)

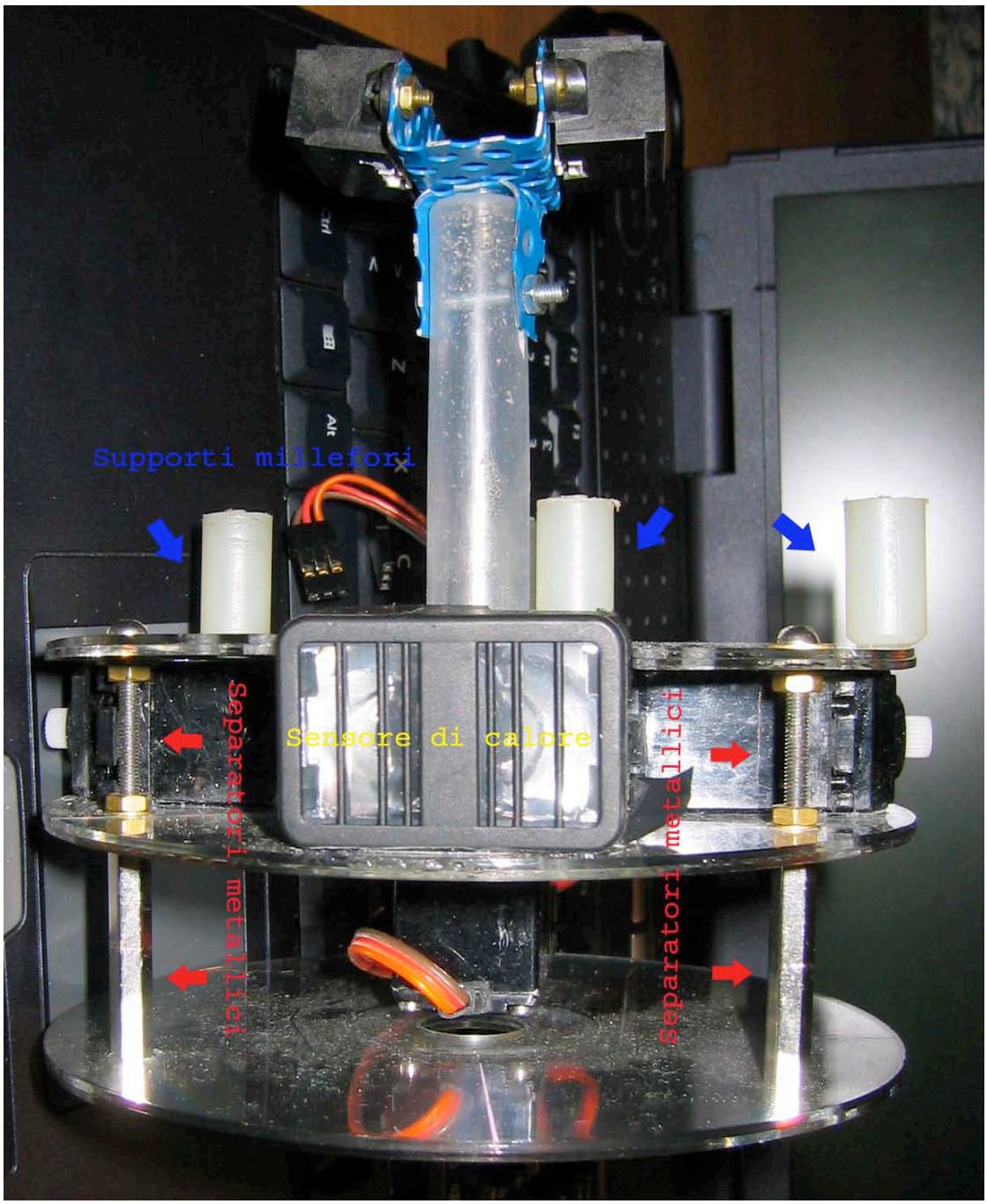

Figura 6: versione finale della struttura meccanica

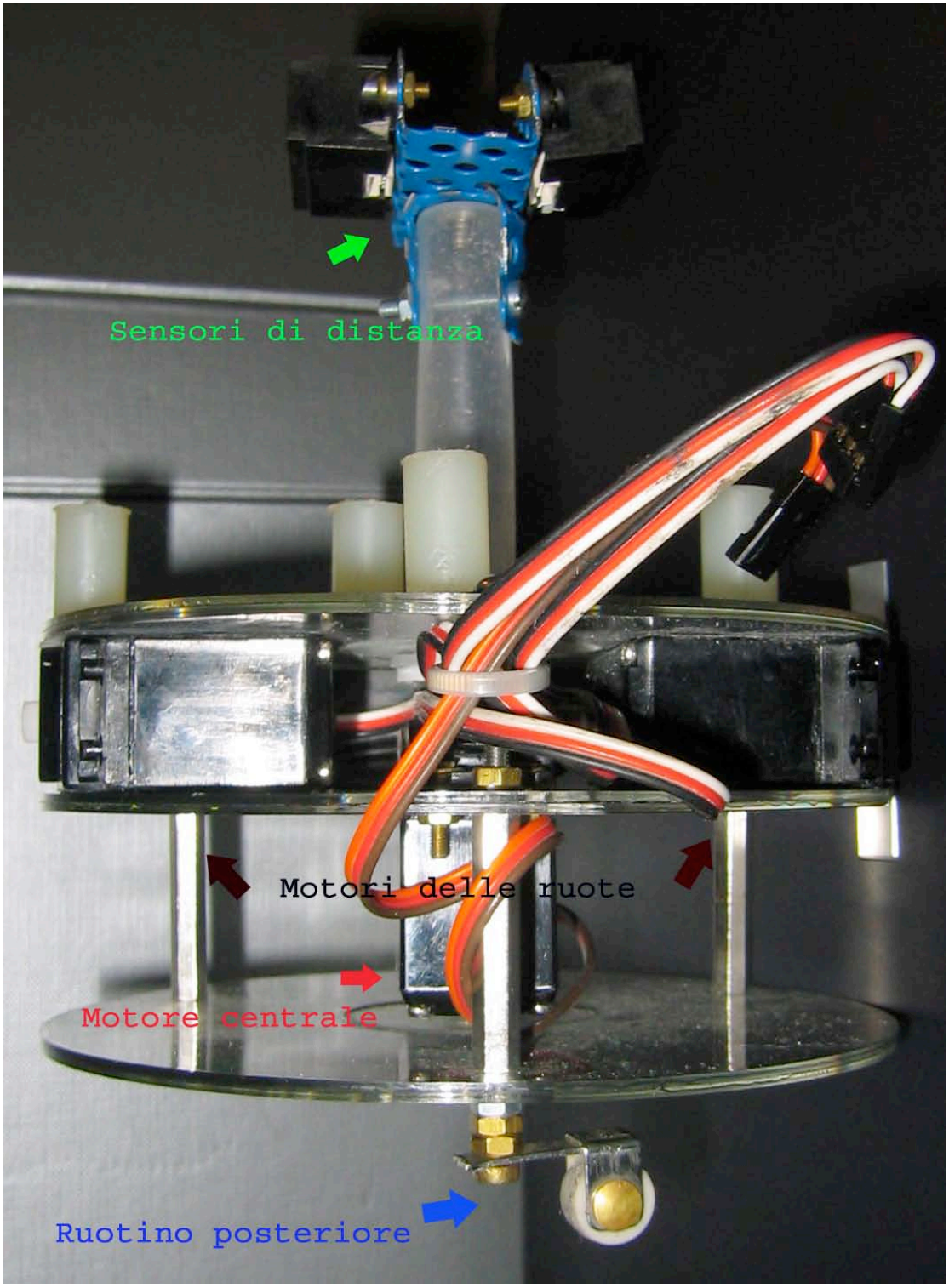

Figura 7: versione finale della struttura meccanica

Le ruote sono realizzate attraverso due CD separati ad uno spessore di 5mm di legno compensato. Il tutto incollato e pressato a dovere.

Figura 8: interno delle ruote

Per aumentare l'aderenza dei CD al suolo abbiamo posizionato una guarnizione di plastica dal diametro di 12 cm sul bordo della ruota, fissata al legno di separazione con del silicone.

Figura 9: le ruote, risultato finale

3.2. L'elettronica

3.2.1. La scheda di controllo

Ecco lo schema elettrico della scheda che controlla il robot.

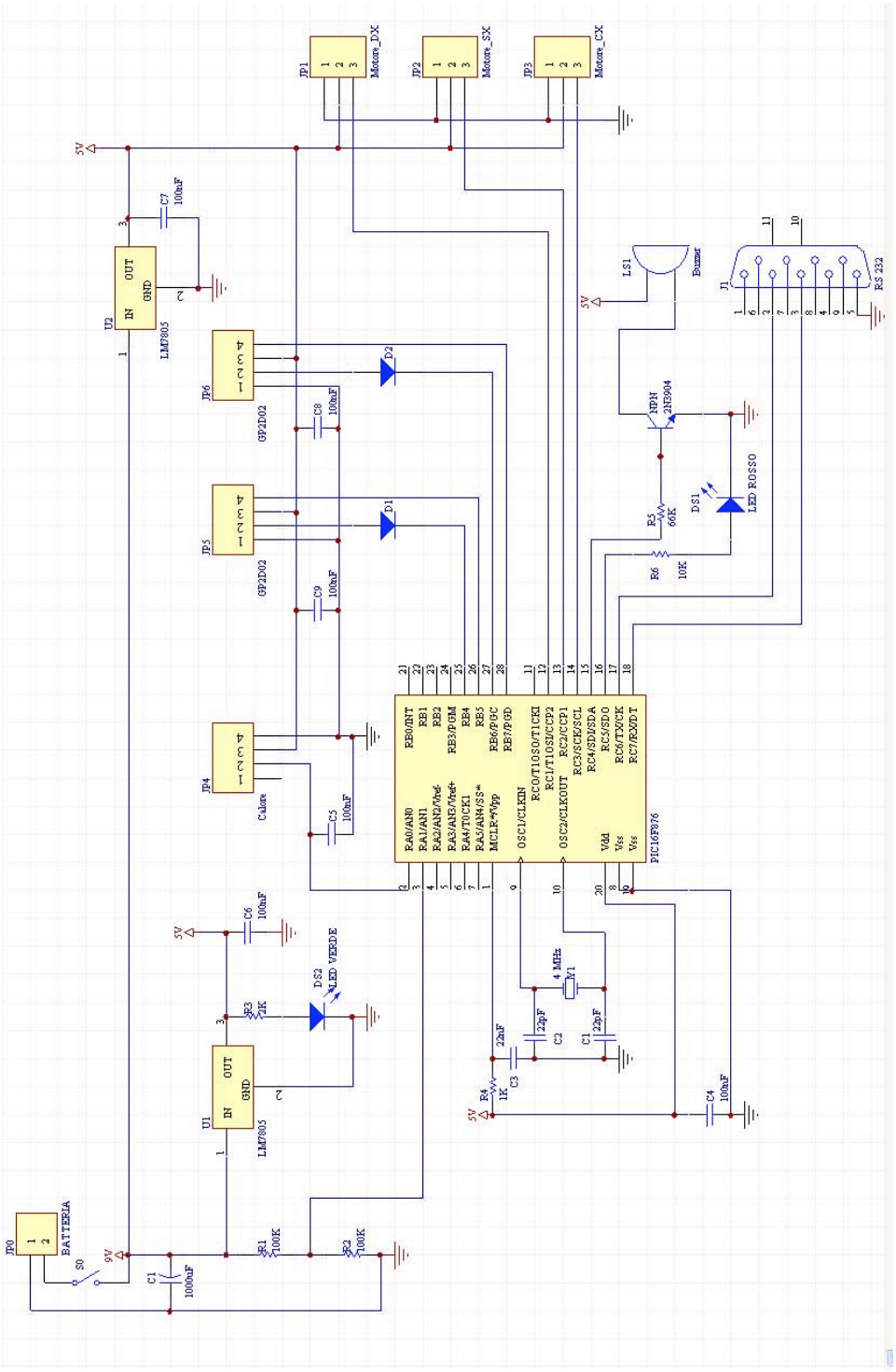

Figura 10: schema elettrico della scheda di controllo

Abbiamo rifatto il circuito da zero su una millefori apportando alcune modifiche. Poiché la sezione di alimentazione è particolarmente critica per il funzionamento corretto del PIC, abbiamo utilizzato due circuiti separati con due regolatori di tensione (LM7805). Uno di questi dedicato esclusivamente ad alimentare i motori e l'altro per l'alimentazione del microcontrollore, di tutti i sensori e del buzzer. Abbiamo inoltre inserito un condensatore elettrolitico da 1000μF per stabilizzare i picchi di cali di tensione che ancora si verificavano durante il funzionamento dei motori e per evitare che il PIC si resettasse a causa della perdita di tensione.

Abbiamo cercato di realizzare i collegamenti, la comunicazione seriale, la lettura dei sensori e i segnali del PWM, che vanno dai vari componenti al PIC il più brevi possibile, in modo che fossero meno soggetti a disturbi elettromagnetici. Inoltre dove possibile abbiamo inserito dei condensatori per cercare di limitare queste interferenze.

Per eliminare i piccoli disturbi in uscita ai regolatori di tensione abbiamo inserito due condensatori da 100 nF. Per cercare di contenere le fluttuazioni del segnale in uscita dal sensore di calore abbiamo inserito, anche qui, un condensatore da 100 nF. Gli altri condensatori presenti sul circuito servono per il corretto funzionamento dell'alimentazione del microcontrollore e per ottimizzare la funzionalità del quarzo.

Il led verde, posto in uscita del regolatore di tensione che alimenta il PIC, serve per indicare l'operatività dello stesso. Il led rosso, collegato al PIC serve per indicare quando il voltaggio della batteria è sceso sotto il livello ottimale per il corretto funzionamento dei regolatori di tensione (cioè sotto i 7.2 V). Tramite due resistenze che fanno da partitore, la tensione viene costantemente monitorata dal microcontrollore e se questa scende al di sotto della soglia prestabilita viene acceso il led rosso.

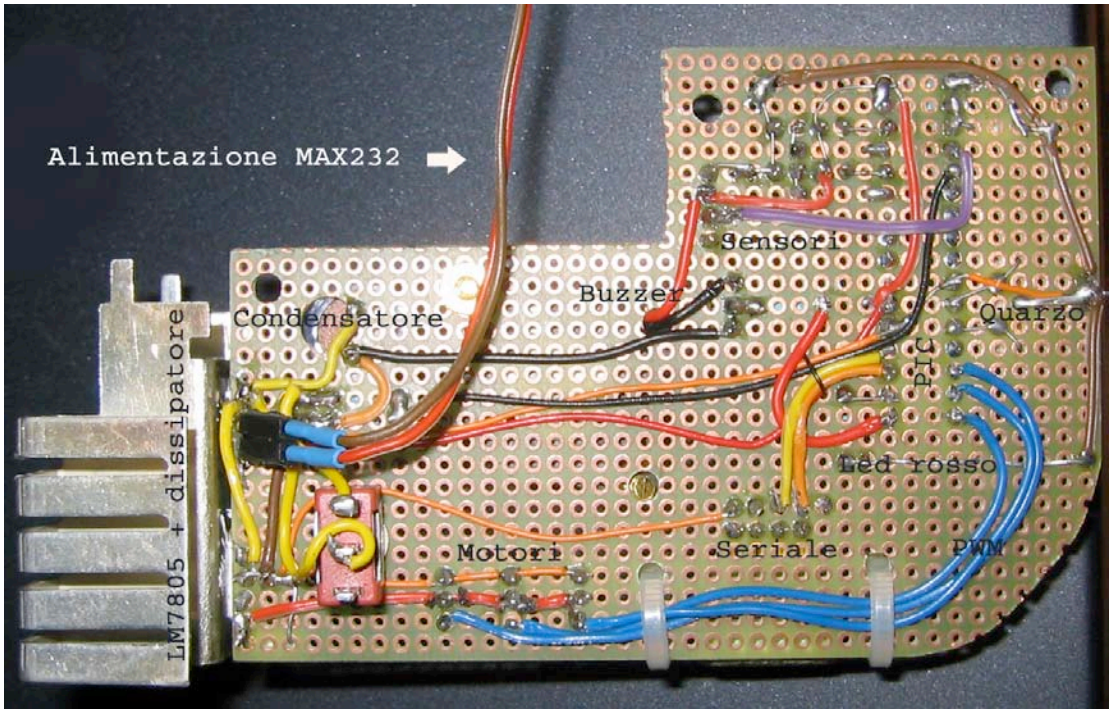

Figura 11: elettronica di controllo

Il buzzer è collegato ad un transistor NPN, ma poteva essere attaccato anche direttamente al PIC, visto che dalle specifiche tecniche questo fornisce in uscita sulle sue

porte di I/O una corrente massima di 200 mA. Il buzzer richiede solamente 1.2 mA se alimentato a 5 V, per sicurezza lo abbiamo comunque collegato tramite un transistor.

3.2.2. L'interfaccia seriale

La comunicazione con il PC avviene via seriale in maniera asincrona a 19200 baud

Il bootoloader si occupa della sincronizzazione della comunicazione. Nella foto seguente possiamo vedere lo schema elettrico del cavo di collegamento.

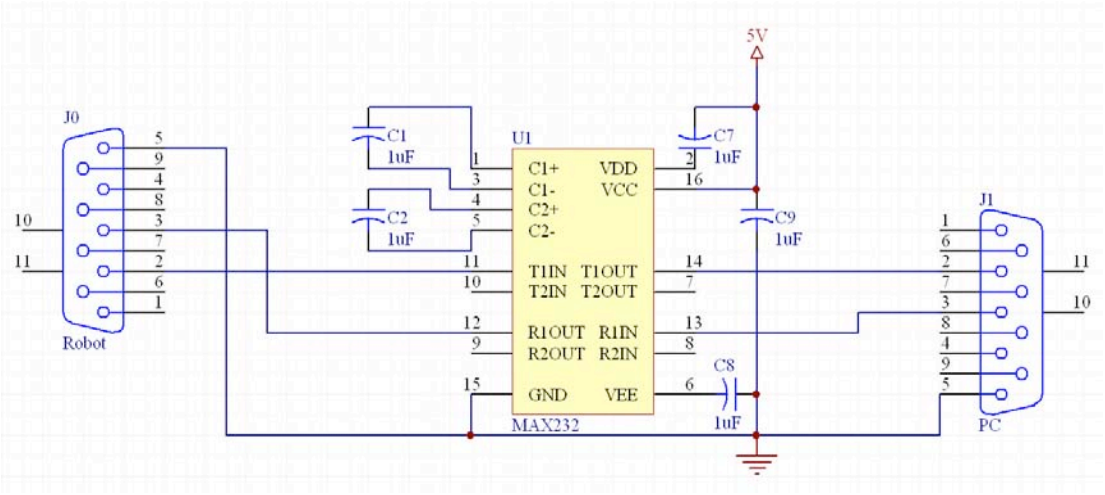

Figura 12: schema elettrico del cavo di collegamento seriale

Ecco come si presenta quello realizzato da noi.

Figura 13: cavo seriale e interfaccia

Per alleggerire la struttura del robot e consentirne ulteriori utilizzi futuri abbiamo evitato di mettere sulla stessa scheda l'integrato che si occupa della traslazione dei segnali da TTL a seriale, il MAX232.

La realizzazione dell'interfaccia è abbastanza semplice, e comporta solo il collegamento dei 5 condensatori elettrolitici da 1 μF necessari al corretto funzionamento del chip. Noi abbiamo inserito anche un led verde (e relativa resistenza) che indicasse la corretta alimentazione dell'integrato.

Per evitare che il circuito venisse accidentalmente a contatto con parti metalliche lo abbiamo inserito in una confezione di plastica e fissato con del silicone. Anche le due porte seriali sono state inserite in due apposite shell in modo da proteggerne i cavi da movimenti bruschi e strappi.

La scheda ha bisogno dell'alimentazione a 5 V DC. Noi l'abbiamo prelevata direttamente da uno dei regolatori di tensione sulla scheda del robot. Nel collegamento basta fare attenzione che il terminale segnato di bianco venga messo a massa e sull'altro la tensione positiva.

3.2.3. L'alimentazione

Per l'alimentazione del circuito sfruttiamo 7 batterie al NiMh da 940 mAh a 1.2 V collegate in serie in modo da ottenere 8.4 V e far funzionare correttamente i regolatori di tensione. Per stabilizzare la tensione in ingresso utilizziamo un condensatore elettrolitico da 47 μF. Quando il livello della tensione scende sotto la soglia, le letture dei sensori di distanza non sono più affidabili e il robot non "vede" correttamente.

4. Programma del PIC

4.1. Il bootloader

Per scrivere il software da noi realizzato all'interno della memoria del PIC ci siamo serviti di un piccola utility che ci ha permesso di effettuare la programmazione senza rimuovere il chip dallo zoccolo e tramite una semplice interfaccia seriale verso il PC, il bootloader. Questo ci ha permesso di semplificare le operazioni di preparazione, debug e installazione del software di controllo sul PIC utilizzato.

Abbiamo scaricato dal sito www.microchipc.com il file contenente sia il codice sorgente che quello da programmare sul PIC a seconda del modello del microcontrollore. Dopo aver scelto il file giusto in base alla frequenza del quarzo e alla velocità del collegamento, lo abbiamo scritto sul chip tramite un normale programmatore attraverso il software ICPROG. A questo punto abbiamo tolto il PIC dal programmatore per inserirlo nello zoccolo della scheda, dopo di che abbiamo caricato, grazie al bootloader, un piccolo programma di test per verificare l'effettiva funzionalità del PIC. Si tratta di un programma che invia in una finestra di terminale (Hyper Terminal) il messaggio [alive] tramite collegamento seriale e restituisce il codice ASCII incrementato di una unità del tasto che si preme sul PC.

Il bootloader è semplicemente un piccolo programma che occupa le ultime (da 0x1F2A a 0x1FFF) 213 parole della memoria di programma e le prime: da 0x0000 a 0x0003. Per utilizzarlo correttamente basta scrivere il programma normalmente in linguaggio assembler e ponendo come prima riga di codice l'istruzione PAGESEL <label> in modo che il programma faccia un salto lungo e non vada a sovrascrivere le prime istruzioni del bootloader.

Il bootloader funziona nel seguente modo: dopo l'accensione del PIC aspetta 0.2 ms controllando se sulla linea seriale vi sia una comunicazione in corso. Se rileva la comunicazione passa a scrivere all'interno della memoria di programma quello che arriva dalla seriale, altrimenti passa all'esecuzione del programma memorizzato precedentemente.

Sul PC bisogna utilizzare il software (PICdowloader) che si occupa della sincronizzazione e del download del programma desiderato sul PIC tramite collegamento seriale.

Se da una parte questo ha reso più agevole la verifica del corretto funzionamento delle varie parti del programma (che poteva essere modificato e reinstallato in modo semplice e veloce) dall'altra ha condizionato alcune scelte di programmazione, a causa della necessità di rispettare alcuni vincoli imposti dal bootloader. Per questo motivo, ad esempio, abbiamo evitato l'uso del WatchDog Timer. Nonostante sarebbe stato utile per motivi di affidabilità, infatti, il suo utilizzo avrebbe causato continui salti all'interno del bootloader con conseguenze per lo più imprevedibili sul comportamento del Robot (ad esempio abbiamo constatato che in seguito ad un reset ed in presenza di disturbi sui piedini del collegamento seriale il bootloader reagiva "pensando" di dover riprogrammare il pic e distruggendo, quindi, il software precedentemente installato).

Il bootloader esiste per diversi sistemi operativi (LINUX, windows, dos). Maggiori informazioni possono essere reperite sui file della documentazione associata e sul sito Internet

4.2. Il programma

4.2.1. Introduzione

Il programma di controllo, inoltre, sebbene realizzato direttamente in assembler, è stato fortemente influenzato dall'abitudine alla programmazione ad alto livello e dal tentativo di renderlo il più possibile modulare e facilmente estendibile. Questo lo ha reso in alcune parti ridondante, ma a nostro parere ne ha aumentato la leggibilità, nonostante l'utilizzo di un linguaggio di basso livello. In particolare abbiamo evidenziato, come si può notare anche dai commenti introdotti nel file sorgente, cinque moduli con compiti diversi:

- La gestione delle interruzioni
- Un insieme di piccole routine di utilità (ad esempio i delay)
- Un blocco decisionale che analizza lo stato del robot per scegliere un'azione
- Un blocco di esecuzione dell'azione scelta
- Un modulo di aggiornamento dello stato del robot

Il main del programma, invece, si riduce alla semplice chiamata dei diversi moduli (vedi figura 1), oltre che a poche righe che abbiamo dovuto aggiungere per temporizzare la lettura dei canali analogici.

Figura 14: il main loop

4.2.2. La gestione delle interruzioni

L'utilizzo delle interruzioni è stata una scelta “forzata” a causa della necessità di pilotare uno dei motori del robot (quello che provvede al posizionamento dei sensori di distanza GP2D02) tramite software. Poiché i sensori devono essere riposizionati di continuo (ogni volta che hanno effettuato una lettura) era impensabile utilizzare dei delay per generare il segnale. Questo infatti avrebbe reso impossibile svolgere altre azioni contemporaneamente allo spostamento dei sensori. Quello che volevamo, al contrario, era che il pic potesse pilotare continuamente questo motore e contemporaneamente eseguire altre parti del programma (considerando che il segnale pilota di tale motore ha un duty cycle compreso tra i 500 e i 2500 μs e che per spostarlo in una certa posizione è necessario attendere per un certo numero di cicli di PWM, è ragionevole fare in modo che durante questo tempo il pic faccia qualcosa di più utile che stare ad aspettare).

In seguito l'utilizzo delle interruzioni è risultato comodo allo scopo di risolvere altri problemi legati a questioni di temporizzazione e competizione tra azioni diverse. Complessivamente, quindi, abbiamo deciso di attivare e gestire 3 tipi di interruzione:

1. Overflow del Timer 1
2. Terminazione della conversione A/D
3. Match tra il Timer 2 e il registro PR2

La 1. viene utilizzata per generare il PWM che pilota il motore di posizionamento dei sensori; la 2. per gestire in modo semplice la lettura dei canali analogici (viene avviata la conversione e quando scatta l'interruzione andiamo a leggere il risultato); la 3. per contare i cicli di PWM (ogni match corrisponde ad 1 periodo) utilizzati per pilotare le ruote del robot e poterne stimare la posizione allorché si debba provvedere a far ruotare il robot in vista di un ostacolo.

4.2.2.1 Gestione del Timer1

L'interruzione segnala che è scaduto il periodo di tempo per cui il segnale sul piedino di controllo del motore che sposta i GP2D02 doveva rimanere ALTO(BASSO). Quindi si provvede a portare la tensione su tale piedino a livello BASSO(ALTO) e a reimpostare il timer.

4.2.2.2 Gestione dell'A/D

L'interruzione segnala che è terminata una conversione A/D. Quindi si provvede alla lettura, al salvataggio in un apposito registro e all'impostazione del prossimo canale analogico da leggere.

4.2.2.3 Gestione del match tra Timer2 e registro PR2

Indica che è scaduto 1 periodo di PWM (nel nostro caso 3000 μs). Contando i match e conoscendo la velocità dei motori, è possibile stimare di quanto si sia spostato il robot. In particolare questo si è reso necessario per controllare la rotazione del robot in vista di un ostacolo.

La routine di gestione delle interruzioni, inoltre, prevede il salvataggio dello stato dei registri del microprocessore in ingresso e il suo ripristino in uscita. In questo modo il

resto del programma risulta virtualmente indipendente da essa. Abbiamo poi cercato di ottimizzare tale indipendenza ordinando le operazioni di tale routine in modo che sprechi meno tempo possibile nella fase di identificazione dell'interruzione e che termini immediatamente nel caso di interruzione non prevista.

4.2.3. Routine di utilità

Questo blocco di codice comprende per lo più piccole routine utilizzate in varie parti del programma. In particolare vi sono due routine di delay (attesa di 20, e di 200 μs), una moltiplicazione per 25 (utilizzata nel calcolo dei parametri del PWM che pilota la posizione dei sensori di distanza) e alcune altre brevi routine di impostazione di diversi moduli del pic (ad esempio per la lettura dei GP2D02 e l'inizializzazione delle conversioni A/D).

4.2.4. Blocco decisionale

Questo modulo costituisce il cuore di tutto il programma e del comportamento del robot. Fondamentalmente si tratta di una serie di salti condizionati che portano alla decisione, tramite il controllo dei vari bit di stato del robot, dell'azione da intraprendere. Inoltre un controllo finale verifica che l'azione scelta non sia già in atto (nel qual caso essa viene scartata e si procede ad una nuova decisione). La scelta dell'azione da effettuare segue lo schema seguente (figura 2).

Questa routine si disinteressa dei dettagli dell'azione da eseguire, di cosa significhi e di quali passi preveda. Al contrario si limita a salvare il risultato del proprio lavoro nel registro new_azione e a ritornare il controllo al main. Sarà poi la routine di esecuzione a provvedere a quanto necessario ad intraprendere questa azione. In questo modo risulta molto semplice modificare lo schema decisionale, non essendo necessario conoscere i dettagli di ogni singola azione: a questo livello un'azione è semplicemente un nome (ad esempio “sposta_GP” , “leggi_GP”, “avanti”, “gira_Dx”, ecc...) e tutto ciò che bisogna fare è decidere in quali condizioni la si voglia intraprendere.

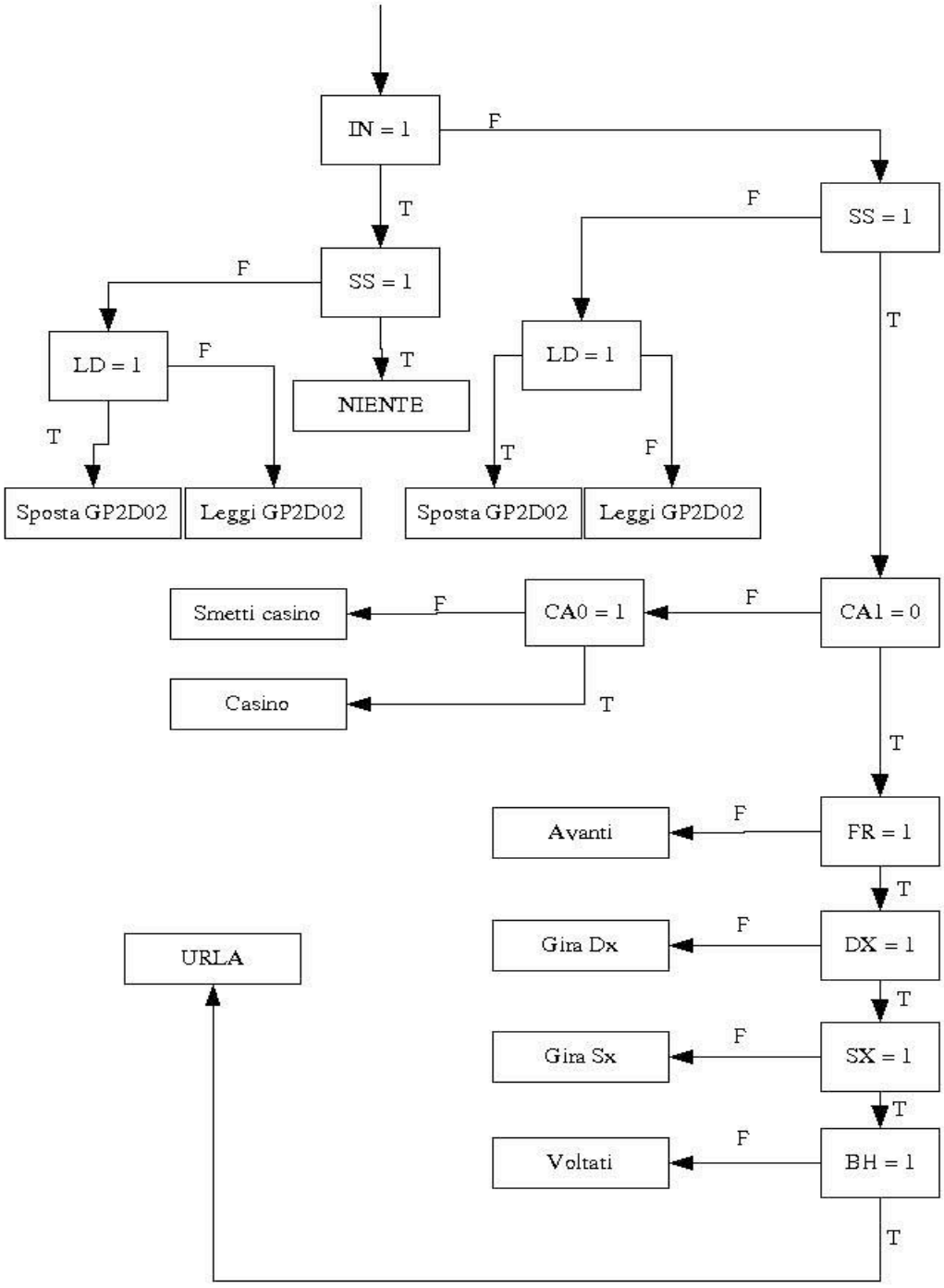

Figura 15: schema decisionale

4.2.5. Esecuzione delle azioni

Si occupa dell'esecuzione vera e propria dell'azione contenuta nel registro "new_azione". Lo schema è stato realizzato in modo da rendere il più possibile ogni azione indipendente, come se ognuna fosse un modulo a sé da poter togliere o inserire a piacimento.

A questo scopo ogni azione và a costituire un blocco (sempre identico) di codice che esegue questi passi

- Controlla se questa è l'azione da eseguire: se no passa al blocco dell'azione successiva,
- se si: Chiama la routine di esecuzione approppriata (ad esempio spostare i sensori ho scelto l'azione "sposta_GP" e chiamo la routine "ex_sposta_GP")
- Se il flag EXACT è settato (1) termino la routine e torno al main

Il bit EXACT (che nelle nostre intenzioni starebbe per Executed ACTion) viene settato al termine di ogni routine di esecuzione (identificate dal prefisso "Ex_"). In questo modo nel momento in cui un'azione viene identificata ed eseguita è possibile tornare immediatamente al main senza spendere inutilmente altro tempo per cercare di identificare l'azione da eseguire con quelle successive. Questo, unitamente al fatto di aver ordinato le azioni da quelle presumibilmente più frequenti a quelle meno frequenti, dovrebbe minimizzare il tempo di esecuzione di questa parte del codice, nonostante la sua ridondanza. Quest'ultima, invece, introdotta allo scopo di rendere il codice molto modulare, permette di aggiungere facilmente nuove azioni, semplicemente "copiando e incollando" la parte riguardante l'identificazione (facendo solo attenzione alla numerazione delle azioni eventualmente modificabile) e implementando una routine "Ex_<action>" opportuna (ricordandosi di settare il flag EXACT come ultima istruzione).

Di seguito riportiamo l'elenco delle azioni previste.

- Niente : Non esegue nessuna operazione. Questa azione viene scelta se l'azione che si dovrebbe intraprendere è già in esecuzione (perchè era stata scelta precedentemente e non è ancora terminata)
- Sposta_GP : Imposta i parametri del PWM da generare per spostare i sensori di distanza, in base alla posizione a cui li si vuole portare; attiva il Timer1; attiva l'interruzione "Timer1 overflow"; porta a livello alto la tensione sul piedino che pilota l motore dei sensori.
- Leggi_GP : Esegue la lettura dei sensori di distanza; confronta il risultato della lettura con una soglia e se l'ostacolo risulta essere sotto-soglia setta a 1 il bit corrispondente alla posizione dell'ostacolo rilevato nei registri DAVANTI e DIETRO, altrimenti porta a 0 lo stesso bit.
- Avanti : Attiva i motori delle ruote impostando i due PWM in modo da far avanzare il robot
- Gira_Dx : Attiva i motori delle ruote in modo da far ruotare il robot verso destra; attiva l'interruzione "Timer2 match PR2" e imposta il numero di cicli di PWM in modo che il robot si volti di circa 90°
- Gira_Sx : Esegue le stesse operazioni di Gira_Dx ma invertendo i PWM

- Voltati : Esegue le stesse operazioni di Gira_Dx ma impostando il numero di cicli di PWM in modo da far ruotare il robot di circa 180°
- Fa_casino : attiva il buzzer
- No_casino : spegne il buzzer
- Urla :

4.2.6. Aggiornamento dello stato

Abbiamo previsto un registro di stato a 16 bit (diviso nei registri stato_MSB e stato_LSB), in cui alcuni bit rappresentano informazioni sullo stato del robot e di ciò che lo circonda (ostacoli rilevati), altri sono stati utilizzati come flag per segnalare eventi particolari e per gestire l'esecuzione delle varie routine.

Il registro di stato completo prevede questi bit :

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
stato_MSB	Passo_1	PRC3	EXACT	VGP	ANC	LD	CA1	CA0
stato_LSB	DX	SX	FR	BH	SS	RDX	RSX	IN

Che hanno il seguente significato :

- IN = Sono in fase di inizializzazione. Questo bit viene settato per forzare il robot a non sceglier azioni di movimento mentre si sta girando (in modo da terminare un'azione prima di intraprendere la successiva).
- RSX = motore sinistro acceso. Viene settato a 1 dalle azioni di movimentazione del robot e resettato quando l'azione termina.
- RDX = motore destro acceso. Viene settato a 1 dalle azioni di movimentazione del robot e resettato al termine dell'esecuzione dell'azione.
- SS = motore sensori acceso. Rimane a 1 per tutto il tempo necessario a riposizionare i sensori, poi viene portato a 0 per permettere la lettura degli stessi.
- BH = presenza di un ostacolo dietro la robot. Viene settato o resettato in base ai bit del registro DIETRO (descritto in seguito)
- FR = ostacolo di fronte al robot. Viene settato o resettato in base ai bit del registro DAVANTI (descritto in seguito)
- SX = ostacolo a sinistra. Viene settato o resettato in base alla combinazione dei registri DAVANTI e DIETRO.
- DX = ostacolo a destra. Viene settato o resettato in base alla combinazione dei registri DAVANTI e DIETRO.
- CA0 = variazione di calore : 0 se in allontanamento, 1 se in avvicinamento
- CA1 = variazione di calore : 0 nessuna variazione, 1 rilevata una variazione

- LD = 1 se la lettura dei sensori di distanza è terminata
- ANC = canale analogico corrente (abbiamo usato un bit perchè gestiamo solo due canali, 0 e 1, per leggere il livello di tensione della batteria e il sensore di presenza)
- VGP = verso di rotazione dei sensori di distanza : 0 orario, 1 antiorario
- EXACT = 1 se ho identificato ed eseguito un'azione
- PRC3 = mirror del piedino RC3 (PORTC, 3) che pilota il motore dei sensori. Questo bit permette alla routine di gestione interruzioni di conoscere il livello di tale piedino (non può essere letto essendo impostato come output) e decidere il da farsi per generare correttamente il PWM che pilota il motore.
- Passo_1 = 1 se sto eseguendo il primo passo (fase iniziale) nel posizionamento dei sensori. In questo caso mantengo il PWM per un tempo più lungo in quanto non conosco la posizione di partenza e voglio assicurarmi che i sensori si portino in posizione iniziale.

4.2.7. Registri utilizzati

Le principali variabili che abbiamo utilizzato (escludendo quelle di appoggio create al solo scopo di fare da contatori e da registri temporanei a cui assegnare i risultati parziali delle varie operazioni) sono :

- new_azione : memorizza l'azione scelta dal blocco decisionale
- old_azione : memorizza l'azione in esecuzione in modo da poterla confrontare con la nuova scelta e decidere se effettuarla o scartarla.
- Lettura_CA : risultato della lettura del sensore di calore
- batteria : livello di tensione della batteria
- pos_GP_n : posizione dei sensori di distanza, viene utilizzato per il calcolo del duty ccle del PWM da generare.
- pos_GP_b : posizione dei sensori di distanza -> ogni bit corrisponde ad una posizione. Questo registro viene utilizzato come maschera per modificare il bit corretto dei registri DAVANTI e DIETRO in presenza/assenza di un ostacolo in una data posizione.
- pwm_duty : il duty cycle del PWM che pilota il motore dei sensori
- ora_AD : timer per temporizzare le letture dei canali analogici
- DAVANTI : ogni bit rappresenta lo stato dell'eventuale ostacolo (1=presente 0=assente) in una data posizione di fronte al robot.
- DIETRO : ogni bit rappresenta lo stato dell'eventuale ostacolo in una data posizione dietro al robot.

4.2.8. Rilevamento degli ostacoli (registri DAVANTI e DIETRO)

Il robot è dotato di due sensori di distanza che si occupano di rilevare rispettivamente gli ostacoli di fronte e dietro al robot. Ognuno dei sensori può coprire un angolo di circa 160° in 8 passi (vedi figura).

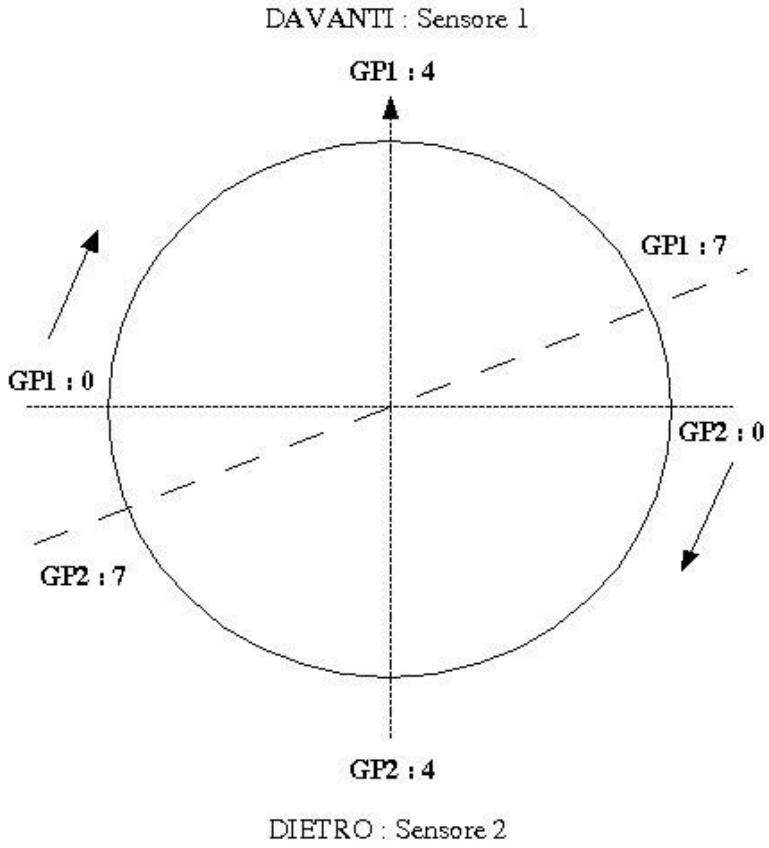

Figura 16: posizioni dei sensori GP2D02

In questo modo le letture dei due sensori sono disgiunte (se avessero coperto 180° la prima lettura di un sensore avrebbe coinciso con l'ultima lettura del sensore opposto). Il fatto di avere utilizzato 8 passi, inoltre, permette di assegnare ad ogni posizione un bit nei registri rispettivamente DAVANTI (per il primo sensore) e DIETRO (per il secondo sensore). In seguito ad ogni lettura viene effettuato un confronto con una distanza di soglia ed il bit corrispondente alla posizione dei sensori n questi due registri viene settato o resettato in base al risultato di tale confronto (un 1 indicherà la presenza di un ostacolo, uno 0 la sua assenza).

In particolare il registro DAVANTI riassume i risultati delle letture effettuate nelle posizioni 0 – 7 del primo sensore, dove la posizione 0 rappresenta l'estrema sinistra (allineata con l'asse della ruota sinistra) e la 7 si trova a circa 160° in senso orario dalla 0.

Lo stesso discorso vale per il registro DIETRO, per il quale la posizione 0 corrisponde all'estrema destra (allineata con l'asse della ruota destra).

In questo modo è possibile settare/resettare il bit corretto dei registri DAVANTI e DIETRO utilizzando per entrambi la stessa maschera, costituita dal registro pos_GP_b

che in ogni istante ha un solo bit a 1 (quello che rappresenta la posizione dei sensori, da 0 a 7).

Una volta eseguite le letture i bit di questi due registri devono essere combinati per capre se e dove si trovino degli ostacoli. In particolare vogliamo riassumere queste informazioni nei 4 bit di stato SX, DX, FR e BH. Per far questo consideriamo per ogni direzione di interesse (destra, sinistra, davanti e dietro) un gruppo di 5 bit rilevanti presi dai due registri e controlliamo se almeno uno di essi risulta settato (presenza di un ostacolo). I bit utilizzati per riassumere lo stato degli ostacoli sono i seguenti :

	DAVANTI								DIETRO							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
FR	✘	✓	✓	✓	✓	✓	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
BH	✘	✘	✘	✘	✘	✘	✘	✘	✘	✓	✓	✓	✓	✓	✘	✘
DX	✓	✓	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✓	✓	✓
SX	✘	✘	✘	✘	✘	✓	✓	✓	✓	✓	✘	✘	✘	✘	✘	✘

4.3. Listato del programma

Il listato del programma si può trovare sul CD nel file “listato.txt” e si può consultare direttamente tramite il link seguente: listato.

5. Componenti necessari

La realizzazione della scheda comporta l’acquisto di pochi componenti, piuttosto comuni e quindi reperibili in qualunque negozio di elettronica di consumo. Sono necessari: un microcontrollore della Microchip PIC16F876, un multiplexer analogico Maxim DG408DJ, un 7805, due transistor NPN (2n2222) e due PNP (2n2907) (questi BJT sono sovradimensionati ma erano disponibili in laboratorio), un quarzo da 4 MHz, led, resistori e condensatori vari (vedi schema elettrico).

Tutti i componenti sono reperibili da RS componenti, tranne i sensori di distanza GP2Y0A02YK acquistati sul sito www.acroname.com.

Ecco lo schema della scheda:

6. Conclusioni e sviluppi futuri

La fase di preparazione del software, ci ha portato allo sviluppo di diverse versioni del programma, man mano che cercavamo di risolvere i problemi incontrati e di migliorare il comportamento del robot. Per questo motivo abbiamo cercato di strutturare il listato in modo da renderlo facilmente estensibile e/o modificabile. Allo stato attuale si possono facilmente aggiungere azioni allo scopo di migliorare le reazioni del robot in presenza di ostacoli o di fonti di calore.

7. Bibliografia

Tutta la documentazione necessaria al progetto è reperibile sui datasheet dei vari componenti che si possono trovare sui siti internet sotto elencati (reperibili anche sul CD allegato):

Microchip: “PIC16F87X Data Sheet 28/40-Pin 8-Bit CMOS FLASH Microcontrollers”, www.microchip.com .

Microchip: “PICmicro™ Mid-Range MCU Family Reference Manual”, www.microchip.com .

Maxim: MAX232 Datasheet, www.maxim-ic.com

Sharp: ”GP2Y0A02YK Long Distance Measuring Sensor”, www.sharpsma.com

Indice