Istituto Tecnico Industriale e Statale di Faenza

Sezione di Elettronica e Telecomunicazioni

Elettronica e Robotica

BRACCCIO ROBOTICO AUTOMATIZZABILE

ATTRAVERSO L'USO DEL COMPUTER

Tesi di Maturità di :

MARCO DALPRATO

MAXIM GROSU

Relatore :

Prof. Ing. GUIDARINI DANIELE

Correlatori :

Prof. Ing. CURCETTI VINCENZO

Prof. Ing. BARNABĖ BRUNO

Prof. GRAZIANI FRANCESCO

Anno Scolastico 2010 - 11

Titolo BRACCIO ROBOTICO AUTOMATIZZATO ATTRAVERSO L'USO DEL COMPUTER

Candidati MARCO DALPRATO

MAXIM GROSU

Relatore PROF. ING. GUIDARINI DANIELE

Parole chiavi ROBOTICA

ELETTRONICA

ELETTRONICA

AUTOMATISMI INDUSTRIALI

SOFTWARE INFORMATICI

**INDICE**

**CAP. 1 : INTRODUZIONE E SCOPI 4**

**CAP. 2 : CARATTERISTICHE GLOBALI DEL PROGETTO 7**

2.1 Caratteristiche del progetto 7

2.2 Analisi dell'Interfaccia USB 9

2.3 Analisi della scheda di controllo 12

2.4 Analisi del braccio robotico 15

**CAP.3 : CARATTERISTICHE DEL SOFTWARE 19**

3.1 Introduzione al software di controllo 19

3.2 Software di controllo Manuale 23

3.3 Software di controllo Automatico 28

3.4 Accenno ad automazioni di tipo complesse 30

**CAP.4 : REALIZZAZIONE PRATICA DEL PROGETTO 32**

4.1 Realizzazione dell'Interfaccia USB 32

4.2 Realizzazione della scheda di controllo 1

4.3 Sensori del braccio robotico 1

**CAP.4 : COLLAUDO 1**

4.1 Collaudo delle schede elettroniche 1

4.2 Verifica del funzionamento 1

**APPENDICE A : IL SOFTWARE DÌ CONTROLO MANUALE 1**

**APPENDICE B : IL SOFTWARE DÌ CONTROLLO AUTOMATICO 1**

**APPENDICE C : INIZIALIZZAZIONE DEL BRACCIO 1**

**CAP. 1**

**Introduzione**

Lo sviluppo industriale e la nascita di nuove tecnologie ha permesso la creazione di prodotti sempre più complessi e complicati dal punto di vista Tecnico-Scientifico .

Le moderne industrie fanno ampio uso di sistemi automatizzati e di controlli informatici per il monitoraggio della catena di produzione industriale.

Sempre più industrie si sono mano a mano evolute dal punto di vista dell'automazione, basti pensare alle moderne industrie che lavorano in ambito automobilistico.

Rientra in questo campo , l'utilizzo di bracci robotici atti alla semplificazioni di operazioni ripetitive e a volte pericolose e complesse.

L'utilizzo di questi macchinari ha permesso la velocizzazione della produzione e del collaudo di tutti quei prodotti che prima venivano lavorati singolarmente dalla mano umana.

Il Braccio robotico trattato in questo progetto , ovviamente , è di dimensioni e di prestazioni nettamente inferiori a quelli utilizzati in ambito industriale ma rappresenta tuttavia una federe simulazione di ciò che avviene nella pratica.

Il progetto consta di tre schede elettroniche lavoranti in simbiosi e di un braccio robotico di dimensioni ridotte.

Il progetto è costituito da varie fasi di analisi , gestione e controllo che sfruttano la capacita del computer di lavorare simultaneamente su più processi.

I dati vengono inviati dal Computer attraverso la trasmissione seriale all'interfaccia USB , che offre 8 canali di input digitali e 8 canali di output digitali.

L'interfaccia è a sua volta connessa ad una scheda di controllo e gestione dei servomotori posti nei vari giunti del braccio robotico.

I dati sfruttano la logica di tipo TTL ( +5 V ) poiché il microcontrollore presente sull'interfaccia USB e i driver per la gestione dei servomotori sono ti tipo TTL.

Attraverso vari attuatori posti sul braccio , si rilevano gli stati e le posizioni di riposo dei servomotori che vengono rilevati e inviati istantaneamente al software di controllo.

Il sistema è pienamente controllabile attraverso il Computer e i tutti i programmi presenti nel braccio sono stati creati utilizzando il linguaggio di programmazione C++.

Il braccio è comunque programmabile utilizzando anche altri linguaggi di programmazione ma si è ritenuto il C++ come il miglior compromesso in termini di prestazioni , affidabilità e comprensione .

Attraverso il C++ si è quindi creato il software per la gestione manuale e automatizzata del braccio .

Tale software rappresenta il miglior sistema per poter gestire in maniera ottimale , sicura e efficiente il braccio robotico.

Le linee di ingresso sono state utilizzare per controllare lo stato del braccio e per permetterne una più accurata gestione senza incorrere in problemi di natura tecnica quali per esempio , la torsione eccessiva di uno dei vari servomotori presenti.

Il presente lavoro è cosi organizzato :

- Il capitolo 2 analizza il progetto in modo specifico , studiando in maniera analitica e specifica il circuito elettronico di ogni scheda elettronica .

- Il capitolo 3 analizza il progetto dal punto di vista software , focalizzandosi sul linguaggio di programmazione C++ e il modo con cui il progetto interagisce con il computer .

- Il capitolo 4 analizza la realizzazione pratica del progetto , indicando in ogni sua minima parte le tecniche e i modi con cui è stato realizzato il braccio robotico e le relative schede elettroniche .

Sono presenti alcune appendici con le istruzioni per l'utilizzo dei software di gestione del braccio , sia per la versione automatica sia per la versione manuale.

Infine la tesi allega un CD-ROM contenente i sue software sopra citati , il presente testo in versione digitale e il listato dei codici in C++.

Desidero ringraziare , per la preziosa collaborazione il Prof. Ing. Guidarini Daniele e il Prof. Ing. Curcetti Vincenzo , il Prof. Ing. Barnabè Bruno , il Prof. Graziani Francesco e il collega Maxim Grosu .

Al Prof. Ing. Fabbri Gianluca , che mi ha insegnato , oltre l'elettrotecnica , anche la vita .

**CAP. 2**

**Caratteristiche Globali del progetto**

**2.1 Caratteristiche del progetto**

Il progetto è suddiviso in blocchi logici connessi tra di loro mediante BUS cavi che ne permettono una facile integrazione .

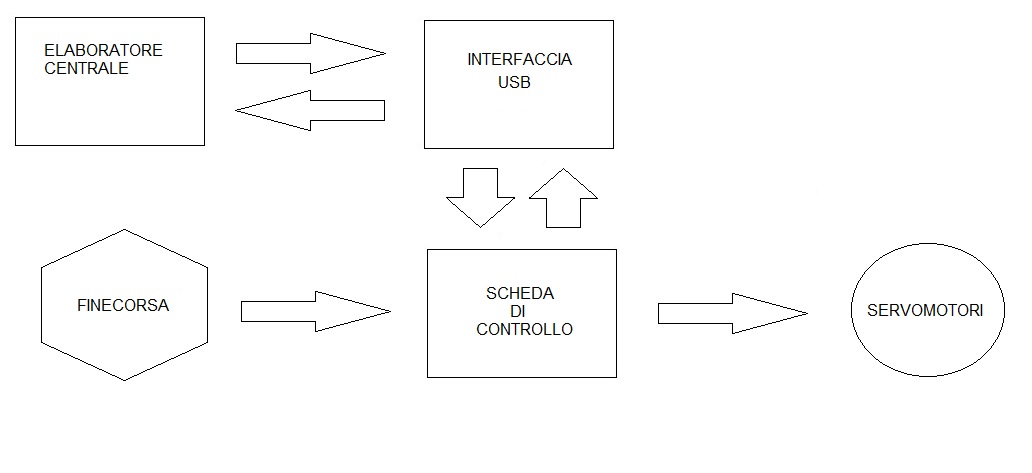
L'Elaboratore centrale , generalmente un personal computer , invia e riceve segnali elettrici attraverso la porta di comunicazione USB.

Tale porta , sfruttando un protocollo di trasmissione a pacchetti asimmetrico permette di inviare e ricevere dati digitali ad una velocità molto elevata .

L'Elaboratore centrale è quindi , connesso attraverso cavo USB all'Interfaccia USB che ne riceve i dati sottoforma di pacchetti e li smista alle 8 uscite presenti sul modulo.

L'Interfaccia è cosi connessa , attraverso un BUS dati di 8 + 8 canali , alla scheda di controllo generale , che rigenera elettricamente i segnali digitali e li elabora per poter controllare e gestire al meglio i servomotori.

Ovviamente sulla scheda di controllo sono presenti anche una serie di 8 connettori che permettono la ricezione di segnali digitali a 5 V .



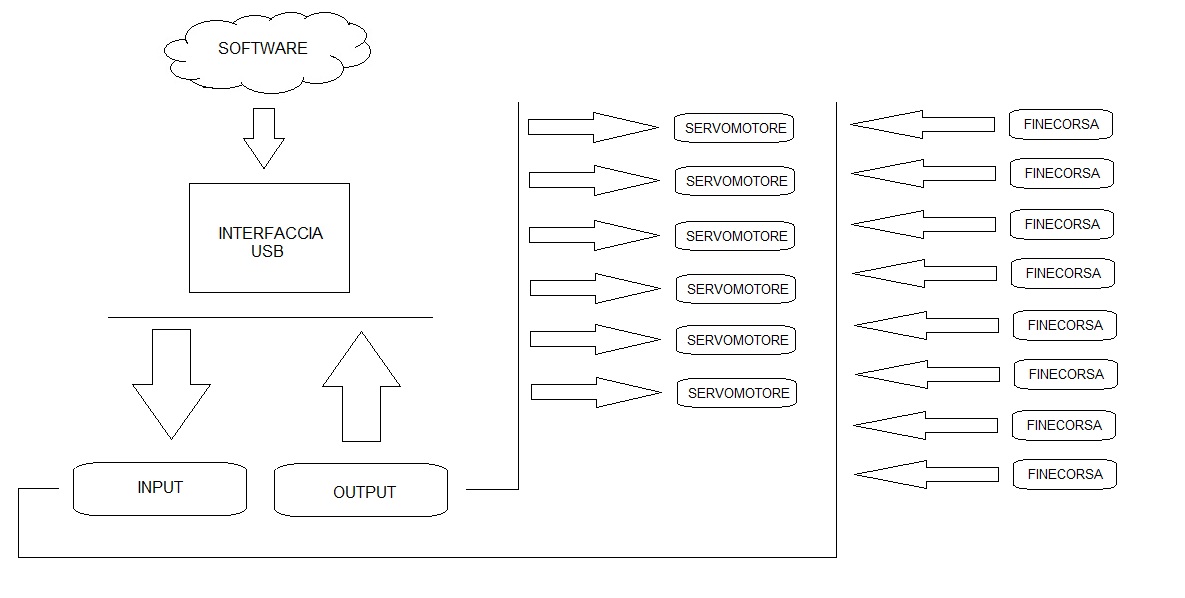
*Fig . 1.0 - Diagramma a blocchi del progetto*

Tali segnali in ingresso , sono instradati dalla scheda di controllo verso l'interfaccia USB , che provvederà ad inviarli all'Elaboratore Centrale.

Nell'Interfaccia USB , le uscite e le entrate lavorano in logica TTL ( +5 V ) e ogni riferimento è sempre rivolto a massa.

La tensione fornita dall'Interfaccia USB è di 5 V con una corrente massima prelevabile di circa 500 mA .

Tale tensione è utilizzata per fornire i segnali logici ai sensori di fine corsa , che provvederanno a fornire un livello logico alto o basso a seconda del loro stato.



*Fig . 1.1 - Diagramma a blocchi dell'Interfaccia USB*

La bassa corrente prelevabile dalla porta USB ha reso necessario l'utilizzo di un alimentatore di tensione continua stabilizzato esterno , per poter fornire un'adeguata potenza ai servo motori.

L'Interfaccia USB sfrutta le potenzialità del PIC 18F4550-I/P sviluppato dalla Microchip per essere utilizzato con la porta di comunicazione seriale USB.

Tale chip poiché non necessita di elevati livelli di corrente è alimentato direttamente dalla porta di comunicazione seriale USB .

La scheda di controllo utilizzata , sfrutta la presenza di tre circuiti integrati siglati L293D che , ricevendo in ingresso 4 bit , commutano le loro uscite attraverso quattro ponti di tipo H .

Questa funzionalità permette di controllare lo stato e il verso di rotazione dei servomotori utilizzati nel braccio robotico.

I finecorsa infine , costituiti da comuni contatti elettrici o interruttori industriali NA , forniscono un segnale di tipo alto nel momento in cui vengono sollecitati dai servomotori.

Il progetto prevede quindi la possibilità di gestire in maniera globale e completa il braccio , stabilendo la posizione che esso dovrà assumere e limitando i movimenti possibili per evitare eventuali danni al braccio stesso.

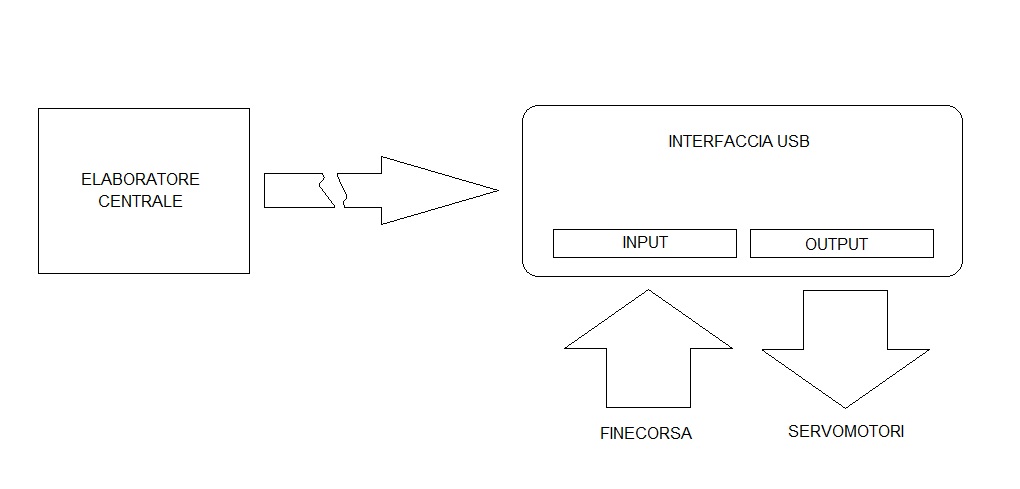
**2.2 Analisi dell'Interfaccia USB**

L'Interfaccia USB è costituita da un Microprocessore centrale , come già accennato precedentemente , del tipo PIC18F4550-I/P sviluppato appunto dalla Microchip.

Tale Chip è stato creato per l'Interfacciamento con la porta Seriale del PC , denominata appunto Universal Serial Bus.

Il PIC in questione , utilizza la porta USB per autoalimentarsi e per comunicare in modo seriale con il PC.

Offre 8 uscite di tipo digitale e 8 ingressi di tipo digitale con riferimento a massa .



*Fig . 2.1 - Schema teorico delll'Interfaccia USB*

Le caratteristiche principali fornite dalla casa costruttrice sono le seguenti :

* Full Speed USB 2.0 (12Mbit/s) interface
* 1K byte Dual Port RAM + 1K byte GP RAM
* Full Speed Transceiver
* 16 Endpoints (IN/OUT)
* Streaming Port
* Internal Pull Up resistors (D+/D-)
* 48 MHz performance (12 MIPS)

Il dato più importante che possiamo rilevare dalle seguenti informazioni è sicuramente la possibilità di disporre di ben 16 canali di Input/Output , che nel nostro caso sono stati separati in 8 di Input e 8 di Output.

Per il controllo della scheda si è scelto di optare per l’uso della suite di programmazione quale C++.

Attraverso un limitato e ristretto set di comandi , è possibile dialogare con l’Interfaccia USB e gestire tutte le linee di ingresso e di uscita .

Sono stati create tre grandi categorie di comandi , tutte quante adibite al settaggio o alla rilevazione dei dati del’Interfaccia USB

Tali comandi si dividono come segue :

* Comandi riguardanti la connessione / gestione dell’Interfaccia
* Comandi riguardanti le uscite
* Comandi riguardanti gli ingressi

Ogni comando fa riferimento ad un “Constructor” che fa a sua volta riferimento a due file , forniti con l’Interfaccia .

Il “Constructor” ha la funzione di collegare tali file al programma e permettere quindi di attribuire ad una determinata parola , i codici per la connessione , rilevazione e settaggio delle uscite / ingressi.

La dichiarazione del “Constructor” rende a sua volta necessario verificare se la Scheda di Interfaccia USB è connessa o meno all’Elaboratore.

Per effettuare questo controllo , si utilizza una particolare stringa che verifica , attraverso l’invio di un pacchetto di bit , l’esistenza o meno dell’Interfaccia USB e se essa è disponibile o meno per la ricezione o invio di dati .

Come è possibile rilevare dalla matrice di codice in figura 2.2 , è possibile verificare o meno la presenza del’Interfaccia USB e di conseguenza eseguire azioni prefissate per tale evento .

Tali azioni possono essere , per esempio , ne caso in cui l’Interfaccia sia connessa al Computer , il reset di tutte le porte di uscita , utile nel caso in cui si siano verificati problemi di comunicazioni nella precedente sessione di programma.

Nel qual caso in cui la scheda non sia connessa al Computer , è possibile far terminare il programma , riportando un messaggio di errore che avvisa l’Utente della mancata connessione di tale dispositivo.

I principali comandi utilizzati in C++ per il controllo delle uscite/ingressi sono :

* **Fotone\* object = new Fotone();** - Constructor , crea l’oggetto Fotone e abbina alla parola “object” tutte le operazioni possibili da eseguire sulla scheda .
* **object->Connect(); -** Prova a connettersi alla scheda USB , e riporta come valore logico “0” se il tentativo è fallito , come valore “1” se il tentativo è andato buon fine.
* **object->InputReadAll();** - Legge tutte gli 8 ingressi digitali e riporta una stringa di 2^8 bit secondo quale entrata è stata connessa a +5 V.
* **object->OutputWriteAll(variabile numerica);** - Setta tutte le 8 uscite a livello logico alto ( +5 V ) se lla variabile numerica è posta ad "1" , se invece la variabile numerica è posta a "0 " setta tutte le 8 uscite a livello logico basso +0 V.
* **object->InputRead(numero ingresso);** - Riporta in forma binaria il valore di un determinato ingresso specificato nel "numero ingresso".
* **object->OutputSet(numero uscita);** - Pone a livello logico altro o basso una delle 8 uscite presenti sull'Interfaccia .
* **object->OutputReset(numero uscita);** - Azione inversa della precedente , setta l'uscita specificata a livello basso.

*//---------------------------------------------------------------------------*

*#include <vcl.h>*

***#include <fotone.h>***

*//---------------------------------------------------------------------------*

*#pragma package(smart\_init)*

*#pragma resource "\*.dfm"*

*THome \*Home;*

*Fotone\* object = new Fotone();*

*//---------------------------------------------------------------------------*

*\_\_fastcall THome::THome(TComponent\* Owner)*

*: TForm(Owner)*

*{*

***if ( object->Connect())****// Controlla se la scheda USB è conessa*

*{*

*// Scheda USB conessa e pronta per ricevere dati*

*object->OutputWriteAll(0); // Resetta tutte le uscite della Scheda USB*

*}*

*else*

*{*

*// Scheda USB non conessa , chiude l’applicazione*

*Close();*

*}*

*}*

*//---------------------------------------------------------------------------*

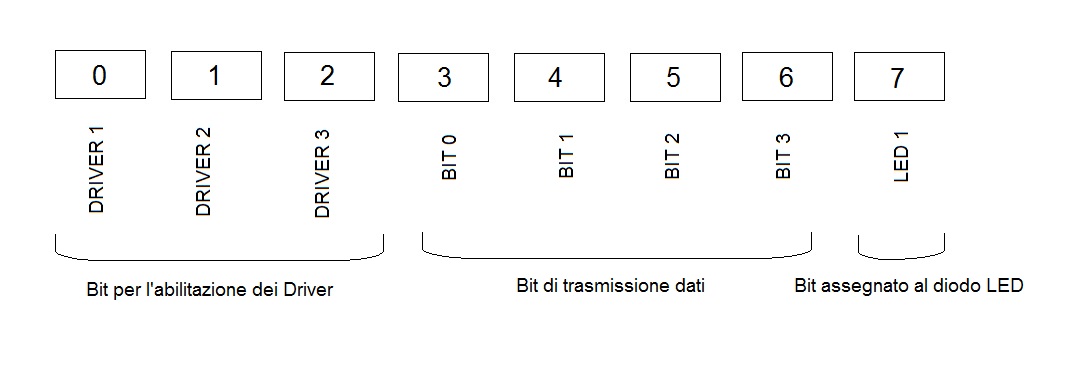
*Fig. 2.2 Esempio di codice per il rilevamento dell’Interfaccia USB*

Ogni comando fa sempre riferimento al Contructor ed è possibilie inserire una sola volta ed un solo Contructor all'interno di un programma , anche se diviso in più Form.

Ogni qual volta viene eseguita un operazione di settaggio a livello altro , si forma un fronte di salita nell'uscita definita nel comando di ampiezza +5 V.

Nel qual caso , invece , venga eseguita un operazione di settaggio a livello basso , si froma un fronte di discesa nell'uscita definita nel comando di ampiezza +0 V.

Le Uscite della scheda sonostate collegare alla scheda di controllo , analizzata nel paragrafo successivo , secondo questo ordine :



*Fig. 2.3 Schema di connessione dei bit di uscita*

I primi tre bit , sono adibiti al controllo dei tre driver presenti nella scheda di controllo , i quattro bit successivi sono adibiti alla gestione del verso di rotazione dei vari servomotori e infine , l'ultimo bit , gestisce direttamente il diodo Led presente nel braccio robotico.

*//---------------------------------------------------------------------------*

*void \_\_fastcall THome::sinistraMouseMove(TObject \*Sender,*

*TShiftState Shift, int X, int Y)*

*{*

*object->OutputSet(2);*

*}*

*//---------------------------------------------------------------------------*

*Fig. 2.2 Esempio di codice per settare un'uscita logica ad un livello alto*

**2.3 Analisi della scheda di controllo**

La scheda di controllo è costituita da tre circuiti integrati , un semplice alimenatore stabilizzato e un'insieme di connettori che permettono la conessione con l'Interfaccia USB.

Attrvaerso il BUS controlli viene inviato un segnale alto per uno dei tre Driver , il relativo Driver si abilita e permette di controllare uno dei due ( o tutti e due ) i servomotori a esso conessi .

Inviando , invece , 4 bit attraverso il BUS dati si settano i versi di rotazione dei due servomotori connessi con il relativo driver.

L'ultimo bit rimasto , come mostra la figura 2.3 , è utilizzato per abilitare o disabilitare il diodo LED presente nella pinza del braccio robotico.

I bit di ingresso , invece , provengono dai vari finecorsa posizionati nelle zone più importanti del braccio e permettono di effettuare movimenti più acucrati e precisi , evitando quindi di superare le torsioni massime di ogni atricolazione .

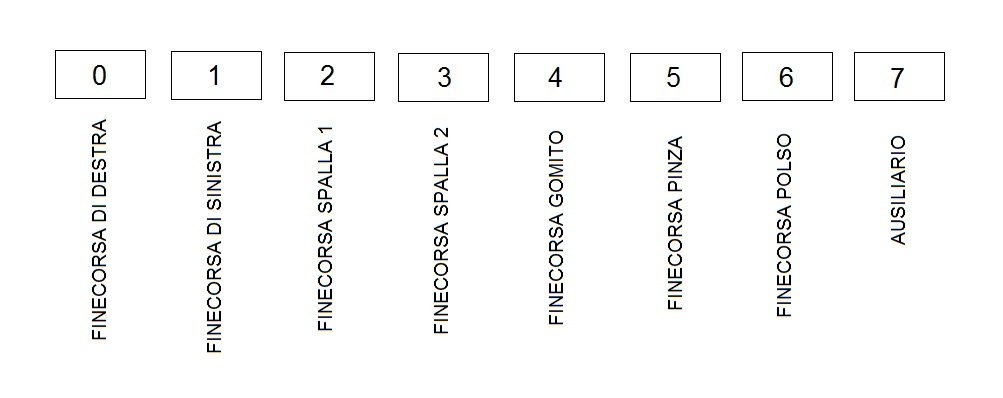
I segnali di ingresso trasportano un segnale logico di tipo alto ( +5 V ) che arriva direttamente dal braccio alla scheda di controllo fino all'Interfaccia USB per poter essere letti dall'Elaboratore.

Ogni Driver funziona con logica TTL a +5 V per la parte di gestione logica invece , la parte dedicata alla gestione dei due servomotori connessi al Driver , la tensione di alimentazione può variare dai +5 V fino ai +12 V D.C.

La corrente massima che può attraversare il circuito integrato durante l'abilitazione di un servomotore è di circa 1000mA.

Per questo motivo , l'alimentatore presente nella scheda di controllo è di tipo stabilizzato con tensione in entrata continua a +12 V e una tensione in uscita che pilota i servomotori variabile .

La tensione ottimale di lavoro per i servomotori è stata fissata a circa +4,25 V , per evitare di stressare troppo gli organi interni contenuti nel braccio .

****

*Fig. 2.4 Schema di connessione dei bit in ingresso*

Come mostrato nella figura 2.4 è possibile osservare come sono stati disposti i vari bit e a quale sensore è stato assegnato il proprio bit.

Infine , il bit numero 7 è stato lasciato libero per eventuali scopi ausiliari di rilevazione dello stato di qualche altro sensore.

L'utilizzo di 3 bit di abilitazione e 4 bit dati ha permesso di gestire il senso di rotazione di ben 6 servomotori , cosa che altrimenti sarebbe stata difficile da effettuare .

Lo schema che segue mostra la logica di funzionamento della scheda di controllo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Servomotore A** | | **Servomotore B** | | **Driver X ( Movimento Base )** | |
| **Bit3** | **Bit4** | **Bit5** | **Bit6** | **Servomotore A** | **Servomotore B** |
| 0 | 0 | 0 | 0 | Fermo | Fermo |
| 0 | 0 | 0 | 1 | Fermo | Muovi a destra |
| 0 | 0 | 1 | 0 | Fermo | Muovi a sinistra |
| 0 | 0 | 1 | 1 | Fermo | Fermo |
| 0 | 1 | 0 | 0 | Muovi a destra | Fermo |
| 0 | 1 | 0 | 1 | Muovi a destra | Muovi a destra |
| 0 | 1 | 1 | 0 | Muovi a destra | Muovi a sinistra |
| 0 | 1 | 1 | 1 | Muovi a destra | Fermo |
| 1 | 0 | 0 | 0 | Muovi a sinistra | Fermo |
| 1 | 0 | 0 | 1 | Muovi a sinistra | Muovi a destra |
| 1 | 0 | 1 | 0 | Muovi a sinistra | Muovi a sinistra |
| 1 | 0 | 1 | 1 | Muovi a sinistra | Fermo |
| 1 | 1 | 0 | 0 | Fermo | Fermo |
| 1 | 1 | 0 | 1 | Fermo | Muovi a destra |
| 1 | 0 | 1 | 0 | Muovi a sinistra | Muovi a sinistra |
| 1 | 0 | 1 | 1 | Muovi a sinistra | Fermo |

*Fig. 2.4 Tabella di verità riguardante i bit di dati*

La tabella di verità rappresenta in modo schematico cosa succede ai servomotori quando viene inviata una stringa di 4 bit al driver di riferimento.

Ovviamente la tabella sovra riportata andrà ripetuta per tutti e tre i driver che interconnessi tra di loro permettono di gestire ogni movimento del braccio robotico.

Oltre ai tre driver per i servomotori , nella scheda è presente anche un alimentatore integrato di bassa potenza , utilizzato per comandare i servomotori garantendo una sufficiente erogazione di corrente .

Tale alimentatore è costituito da un integrato LM317 che utilizza la tensione in ingresso per fornire in uscita una tensione perfettamente stabile e soprattutto regolabile .

La regolazione di tale tensione , viene effettuata mediante un trimmer .

Tale trimmer permette di variare la tensione dei servomotori da un minimo di circa 3 Volt ad un massimo di circa 6 Volt.

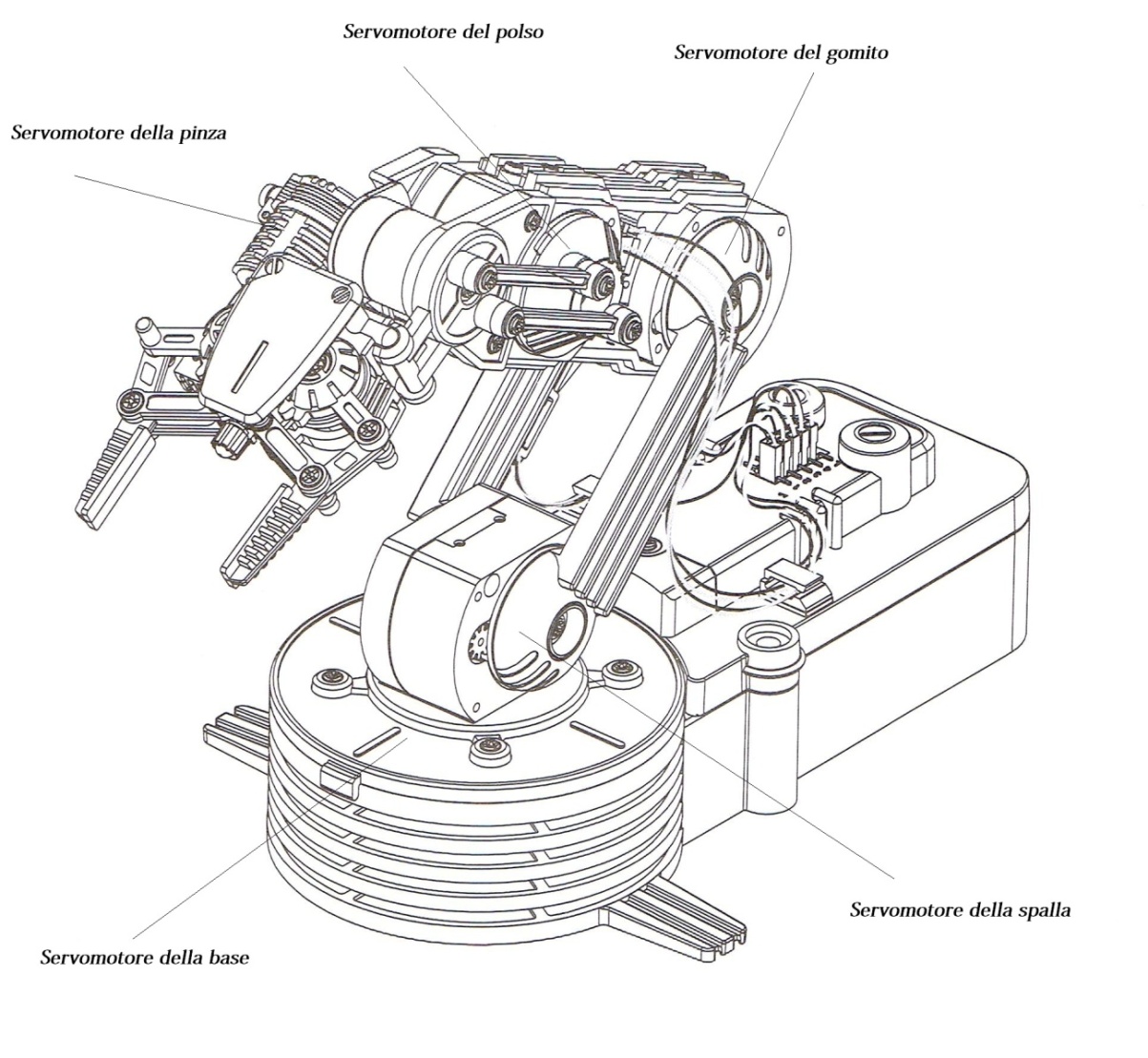
Questa regolazione permette di proteggere i servomotori da brusche variazioni di corrente / tensione e di agire direttamente sulla velocità complessiva delle azioni del braccio.

L’alimentatore provvede direttamente ad alimentare i driver per i servomotori ed è dotato di un circuito integrato per la protezione dei sovraccarichi ( se la caduta massima sull’integrato supera un valore prefissato dalla casa costruttrice , l’integrato smette di funzionare ) e di un circuito di protezione per la temperatura .

Ovviamente l’intera scheda di controllo è connessa al braccio mediante un cavo di tipo flat e all’Interfaccia USB mediante un BUS di cavi che permette di ricevere e inviare i dati da e verso l’Interfaccia.

**2.4 Analisi del Braccio Robotico**

Il braccio robotico utilizzato è costruito da un'azienda americana ed è venduto al pubblico ad una cifra molto bassa.



*Fig. 2.5 Disegno tecnico del braccio robotico utilizzato nel progetto*

Il braccio era munito di telecomando , per poter essere comandato solo attraverso di esso , ma tale telecomando è stato rimosso per poter inserire la scheda di controllo e l'Interfaccia USB.

Il Braccio si compone di 5 articolazioni e 5 relativi servomotori DC a spazzola con motoriduttore fisso con frizione anti - saltellamento .

Ogni articolazione è connessa con la successiva mediante il servomotore che ne trasmette il proprio movimento rotatorio.

Ogni servomotore collegato alla scheda di controllo mediante un connettore femmina .

Nella parte terminale del braccio , ove vi è la pinza , è stato inserito un diodo Led ad alta luminosità utilizzato per lavorazioni notturne .

Sono stati disposti vari sensori disposti lungo il braccio che permettono di controllare il braccio senza rischiare di danneggiare una o più parti del braccio robotico .

Oltre alla funzione di controllo della torsione massima per ogni servomotore , i sensori fungono anche da punto di partenza per ogni movimento del braccio .

I movimenti del braccio sono cinque , che permettono di effettuare tutti i movimenti principali che un braccio robotico dovrebbe realizzare .

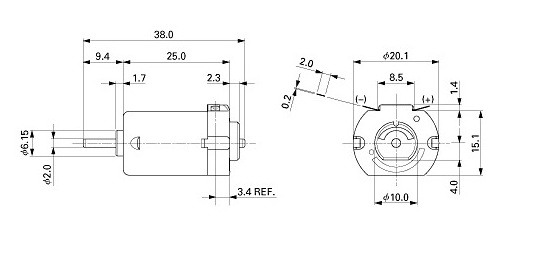
I movimenti principali sono :

* Movimento della base a destra e a sinistra ( autolimitata a destra e a sinistra )
* Movimento della spalla in su e in giù ( autolimitata dai finecorsa )
* Movimento del gomito in su e giù ( autolimitata dai finecorsa )
* Movimento del polso in su e giù ( autolimitata dal finecorsa solo per movimenti verso il basso
* Chiusura e apertura della pinza ( autolimitata solo nella fase di chiusura )

Ogni servomotore è connesso ad una scheda di controllo che smista i dati provenienti dai vari Driver indirizzandoli verso i vari servomotori .

Ogni servomotore , ha un angolo di rotazione limitato sia verso l'altro sia verso il basso .

Questa limitazione ha reso obbligato l'uso di vari sensori di finecorsa per impedire appunto , che i servomotori superino il grado massimo di rotazione.



*Fig. 2.6 Disegno tecnico del motore utilizzato nel progetto*

Per poter dimensionare il circuito dell'alimentatore trattato nel paragrafo precedente è necessario conoscere la potenza massima teorica che il motore può assorbile .

Ipotizzando il motore in funzione nel braccio robotico , quindi avente come carico il solo peso del braccio da spostare , senza ipotizzare nessun peso aggiuntivo , i calcoli per dimensionare l'alimentatore sono i seguenti :

Dai dati tecnici forniti dal produttore sappiamo che la velocità media del motore è di circa 20.000 giri/minuto quindi 20.000 RPM.

Sappiamo inoltre che la coppia che può erogare è di circa 97,34 \* 10 - 6 Kgm .

Per convertire i Kgm in Newton per metro si utilizza la formula :

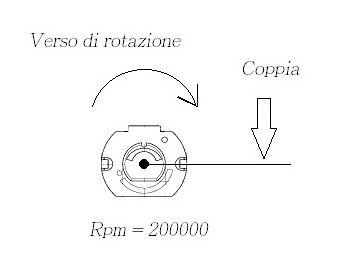
Se ne ricava che :

Quindi 97,34 \* 10 - 6 Kgm corrispondono a circa 954,60 \* 10 - 6 Nm .

Sostituendo i valori conosciuti con la formula indicata precedentemente otteniamo che :

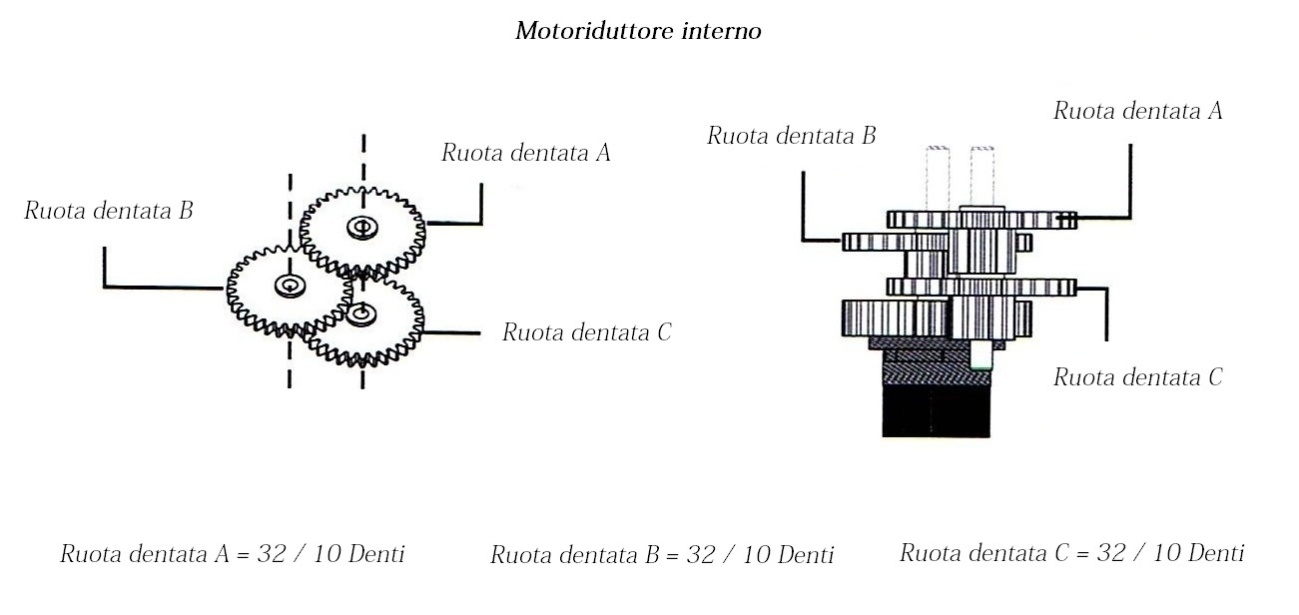
Risolvendo la formula otteniamo che la potenza di un motore in continua a vuoto è di circa 2 W .

Grazie a questa informazione abbiamo potuto realizzare l'alimentatore integrato nella scheda di controllo che quindi dovrà fornire una potenza di qualche Watt superiore a quella calcolata.



*Fig. 2.7 Disegno tecnico del motore indicando Coppia e verso di rotazione*

La coppia erogata dai motori è insufficiente per poter muovere i componenti del braccio , quindi sono stati realizzati dei motoriduttori in cui i motori vengono inglobati per permettere appunto una maggior coppia .



*Fig. 2.8 Disegno tecnico degli ingranaggi presenti nel motoriduttore*

Nel motoriduttore sono presenti tre ruote dentate che prelevano il movimento dal motore e lo trasmettono ad un pignone che provvede a sua volta a trasmetterlo all'articolazione in cui è inserito il motoriduttore.

Quindi , l'utilizzo di questo tipo di motoriduttore permette di aumentare la coppia dei motori e quindi , di aumentare la forza massima che possono imporre sulle varie articolazione del braccio .

In totale il braccio può compire 12 movimenti che simulano il grado di libertà che un braccio umano generalmente ha .

**CAP. 3**

**Caratteristiche del Software**

**3.1 Introduzione al software di controllo**

I software di gestione , come spiegato del Capitolo 1 , sono stati interamente programmati utilizzando il linguaggio di programmazione C++ , sfruttando il programma C++ Builder 4 .

I software presenti in questo progetto sono di due tipi :

- Software per il controllo manuale del braccio

- Software per il controllo automatico del braccio

Il primo software serve per poter comandare i vari movimenti del braccio robotico utilizzando semplicemente il mouse e/o la tastiera .

Ogni movimento è indicato nella videata principale ed è possibile interagire in maniera semplice e rapida su tutti i movimenti del braccio.

Il secondo software , molto più complesso , permette di creare veri e propri programmi visuali utilizzando un linguaggio di programmazione creato specificatamente per il controllo braccio robotico.

Quest'ultimo software ha la capacità di far eseguire complessi e ripetitivi movimenti , utilizzati sopratutto in ambiti quali per esempio l'automazione industriale .

I software sono stati interamente creati utilizzando C++ Builder 4 su ambiente operativo Windows © XP Professional , ma è garantita la perfetta esecuzione anche su ambienti operativi quali Windows © Vista 32 bit e Windows © Seven (7) a 32 bit .

Sono stati riscontrati problemi di compatibilità durante l'installazione dei driver in ambienti operativi a 64 bit.

Necessario per il corretto funzionamento dei software precedentemente descritti è l'installazione del driver per il riconoscimento dell'Interfaccia USB.

Per tale proceduta si riporta quanto descritto nell'Appendice C .

Per comprendere il funzionamento del software di controllo è necessario analizzare in modo dettagliato il linguaggio di programmazione rivolto al braccio e in che modo esso interagisce con il Computer .

Innanzitutto , come analizzato nel Paragrafo 2.2 , il Computer può comunicare attraverso l'Interfaccia USB settando unicamente i bit in uscita e leggendo i bit in ingresso.

Per comandare i cinque servomotori disposti lungo tutto il braccio si deve far uso ci undici combinazioni di codici che permettono l'attivazione o la disattivazione dei vari servomotori .

|  |  |
| --- | --- |
| **Movimento** | **Bit da attivare/disattivare** |
| Rotazione base a destra | 2 ; 6 |
| Rotazione base a sinistra | 2 ; 5 |
| Movimento spalla in giù | 1 ; 6 |
| Movimento spalla in su | 1 ; 5 |
| Movimento gomito in giù | 1 ; 4 |
| Movimento gomito in su | 1 ; 3 |
| Movimento polso in giù | 2 ; 4 |
| Movimento polso in su | 2 ; 3 |
| Chiusura pinza | 0 ; 6 |
| Apertura pinza | 0 ; 5 |
| LED | 7 |

*Fig. 3.0 Tabella rappresentante i bit da abilitare rispettivi al movimento da eseguire*

Per poter far comunicare il Software C++ Builder 4 è comunque necessario , oltre ad aver installato il driver , includere nell'Header il file "fotone.h".

Oltre a includere il file è necessario scrivere anche il "Constructor" e per questa operazione si riporta al Paragrafo 2.2 .

In Figura 3.1 è scritta la parte principale dell'Header .

//---------------------------------------------------------------------------

#include <vcl.h>

#include <fotone.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma link "CSPIN"

#pragma resource "\*.dfm"

TForm1 \*Form1;

Fotone \*object = new Fotone ;

//---------------------------------------------------------------------------

*Fig. 3.1 Header del programma*

Per comprendere meglio il funzionamento di questo sistema , è possibile analizzare un frammento di codice del software per la gestione automatica del braccio robotico.

*if (lista\_azioni->Items->Strings[lista\_azioni->ItemIndex] == "Apri pinza")*

*{*

*object->OutputSet (0);*

*object->OutputSet (5);*

*}*

*Fig. 3.2 Frammento di codice riguardante l'apertura della pinza*

Nel codice riportato in Fig. 3.1 è possibile osservare come sia possibile aprire la pinza attraverso il settaggio dei bit riguardanti il driver e il verso di rotazione del motore.

Il settaggio dei bit permette l'attivazione del servomotore , ma non ne ferma il funzionamento , per poter eseguire quest'ultima operazione è necessario scrivere un'altra stringa di codice che andrà inserita dove dovrà essere fermato il braccio.

La stringa di codice per l'interruzione del braccio conterrà ovviamente il modulo di reset degli stessi bit che precedentemente sono stati settati.

*if (lista\_azioni->Items->Strings[lista\_azioni->ItemIndex] == "Apri pinza")*

*{*

*object->OutputReset (0);*

*object-> OutputReset (5);*

*}*

*Fig. 3.3 Frammento di codice riguardante l'arresto della pinza*

Come mostrato nella Fig. 3.2 il movimento viene fermato dal comando di reset dei bit 0 ; 5 che aprono la pinza .

La combinazione di tutti i movimenti all'interno dei due software permette all'utente di creare e realizzare complesse combinazioni di movimenti .

Molto interessante è la stringa di codice che accende il led posto nella pinza del braccio robotico.

Tale stringa di codice è interessante dal punto di vista informatico poiché fa stretto uso di una variabile pubblica .

La subroutine è la seguente :

*//---------------------------------------------------------------------------*

*void \_\_fastcall THome::led\_0MouseUp(TObject \*Sender, TMouseButton Button,*

*TShiftState Shift, int X, int Y)*

*{*

*led = !led ;*

*if ( led == 1 )*

*{*

*object->OutputSet(7);*

*}*

*else*

*{*

*object->OutputReset(7);*

*}*

*}*

*//---------------------------------------------------------------------------*

*Fig. 3.4 Frammento di codice riguardante l'accensione e lo spegnimento del led*

La subroutine rappresentata in Fig. 3.4 funziona nel seguente modo :

- Viene creata una variabile pubblica ***led*** di tipo Integer a cui viene attribuito il valore numerico intero 0 .

- Quando si richiama la subroutine , il codice ***led = !led***; crea la negazione della variabile led ( quindi led = 1 ) .

- Attraverso il controllo ***if ( led == 1 )***si verifica se la variabile ***led*** è uguale a 1 e si procede con il settaggio del bit in uscita che accende il led .

- Nel caso in cui venga rieseguita la subroutine , la variabile ***led*** si nega di nuovo e da 1 si riporta a 0

- Viene rieseguito il controllo ***if ( led == 1 )***e in questo caso , poiché ***led*** è uguale a 0 viene resettato il bit 7 e quindi spento il led .

La struttura precedentemente descritta è molto utilizzata nel software di gestione manuale , perché premette di porre a livello logico alto un'uscita rispettivamente a ciò che succede ad una variabile X collegata con l'uscita in questione.

**3.2 Software di controllo manuale**

Il software di controllo manuale permette di eseguire varie funzioni , tutte in real-time e senza automazioni di alcun genere .

Le funzioni principali di questo software sono :

- Completa gestione del braccio robotico attraverso l'uso del mouse

- Parziale gestione del braccio robotico attraverso l'uso della tastiera

- Lettura degli ingressi

- Diagnostica generale dell'Interfaccia USB

- Diagnostica specifica dell'Interfaccia USB

- Completa gestione della Mouse Wheel e possibilità di attribuire tutte le articolazioni del braccio a essa

- Lettura in real-time degli ingressi e rispettiva analisi della posizione di ogni articolazione

- Possibilità di escludere gli ingressi

- Completa gestione del LED di illuminazione

- Finestra per la gestione dei comandi rapidi attraverso la Mouse Wheel

- Finestra per il controllo da tastiera

- Controllo simultaneo da tastiera e da Mouse Wheel

Funzioni aggiuntive sono meglio indicate e spiegate in maniera esauriente nell'Appendice A .

Il software di controllo Manuale è composto da una serie di finestre attive che permettono una facile gestione del braccio.

Un primo controllo di diagnostica è effettuato all'avio del braccio , dove viene rilevata la presenza del'Interfaccia USB e nel caso in cui quest'ultima non venga rilevata , si provvede a lanciare un messaggio di errore all'Utente e ad interrompere ogni tentativo di comunicazione con l'Interfaccia USB.

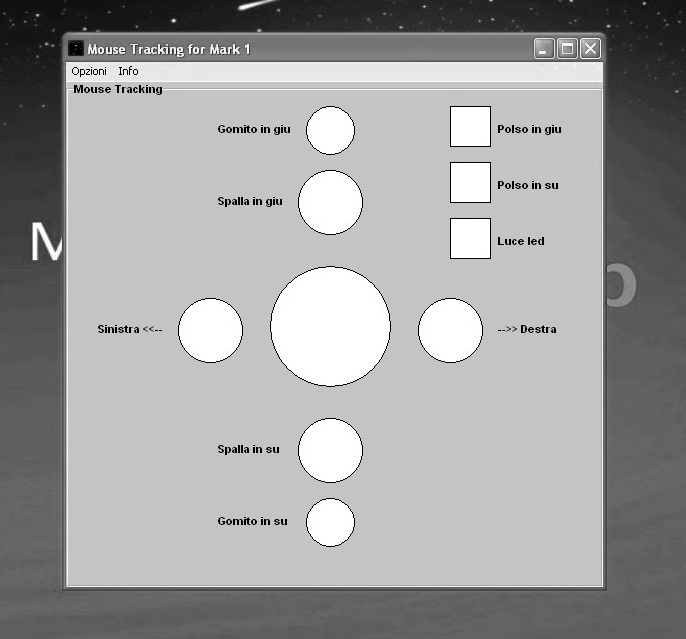
Un'altra procedura di diagnostica è effettuata sulle porte di ingresso , permettendo cosi di poter controllare il braccio senza rischiare di danneggiarne i componenti principali.

La schermata principale del software di controllo manuale rappresenta in maniera schematica la disposizione dei vari servomotori e i vari movimenti che il braccio robotico può fare .

Passando con il mouse su uno nei sette cerchi ( al cerchio centrale non è affidata alcuna operazione ma indica lo stato di stallo ) si mette in movimento il relativo servomotore .

Per esempio , se si posiziona il mouse sul cerchio recante la scritta " Spalla in giù " , il braccio eseguirà il movimento indicato.

In Figura 3.5 è rappresentata la finestra principale del programma di controllo manuale per il braccio robotico.



*Fig. 3.5 Finestra principale del software di controllo*

Per il controllo di ogni evento è stato attribuito una stringa di codice che permette di muovere il relativo servomotore.

Per esempio , il codice per l'avvio del servomotore che muoverà la spalla in giù sarà il seguente :

void \_\_fastcall THome::spalla\_giuMouseMove(TObject \*Sender,

TShiftState Shift, int X, int Y)

{

// \*\*\*\* muovi spalla in giu \*\*\*\*

spalla\_giu->Brush->Color = clBlue ;

object->OutputSet(1);

object->OutputSet(6);

}

//---------------------------------------------------------------------------

*Fig. 3.6 Codice per il movimento della spalla in giù*

Nell'istante in cui l'utente sposta il mouse dalla figura che permette il movimento prestabilito , il braccio si arresta e attende un nuovo comando.

La codifica di tale operazione è osservabile nel riquadro di Figura 3.7 .

void \_\_fastcall THome::GroupBox1MouseMove(TObject \*Sender,

TShiftState Shift, int X, int Y)

{

// \*\*\*\* Resetto colori \*\*\*\*\*\*

centro->Brush->Color = clWhite ;

sinistra->Brush->Color = clWhite ;

destra->Brush->Color = clWhite ;

spalla\_su->Brush->Color = clWhite ;

spalla\_giu->Brush->Color = clWhite ;

gomito\_su->Brush->Color = clWhite ;

gomito\_giu->Brush->Color = clWhite ;

polso\_su->Brush->Color = clWhite ;

polso\_giu->Brush->Color = clWhite ;

// \*\*\*\*\* Rsett uscite \*\*\*\*\*\*\*\*

object->OutputReset(0);

object->OutputReset(1);

object->OutputReset(2);

object->OutputReset(3);

object->OutputReset(4);

object->OutputReset(5);

object->OutputReset(6);

}

//---------------------------------------------------------------------------

*Fig. 3.6 Codice per l'arresto del braccio*

Ripetendo l'operazione di settaggio delle uscite per ogni movimento , si ottiene il programma che gestisce tutti gli undici movimenti possibili.

La Figura 3.7 mostra , invece , lo stato di tutte e otto le linee di ingresso indicando attraverso una spunta se la l'ingresso è attivo .

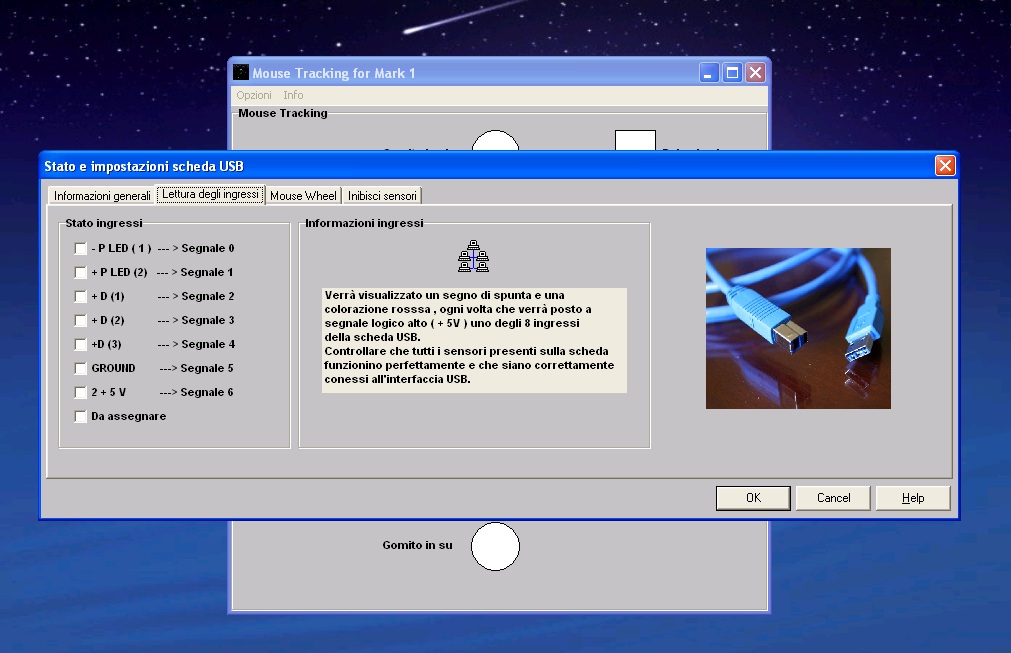
Nel menu riguardante lo Stato e le Impostazioni della Scheda USB si possono settare anche le principali caratteristiche del software di controllo manuale , come per esempio quale articolazione attribuire alla rotellina del mouse o la durata dei tale movimento .

Ogni qual volta un finecorsa viene premuto , il segnale logico viene trasportato dalla scheda di controllo fino ad arrivare all'Interfaccia USB che a sua volta trasferirà il segnale attraverso il cavo di connessione USB per poter essere letto e manipolato via software dal programmatore o dall'Utente.

La lettura degli ingressi , come specificato meglio nei precedenti paragrafi ha permesso di controllare al meglio il braccio robotico , impedendogli effettuare manovre o torsioni superiori al massimo consentito.

La possibilità li leggere i dati in ingresso permette anche , come verrà analizzato meglio nel capitolo successivo , di creare movimenti automatici basandosi su punti di riferimento prestabiliti.

Tali punti fungono appunto da " basi di partenza " per tutti quei movimenti automatici.



*Fig. 3.7 Menu informativo sulle linee di ingresso*

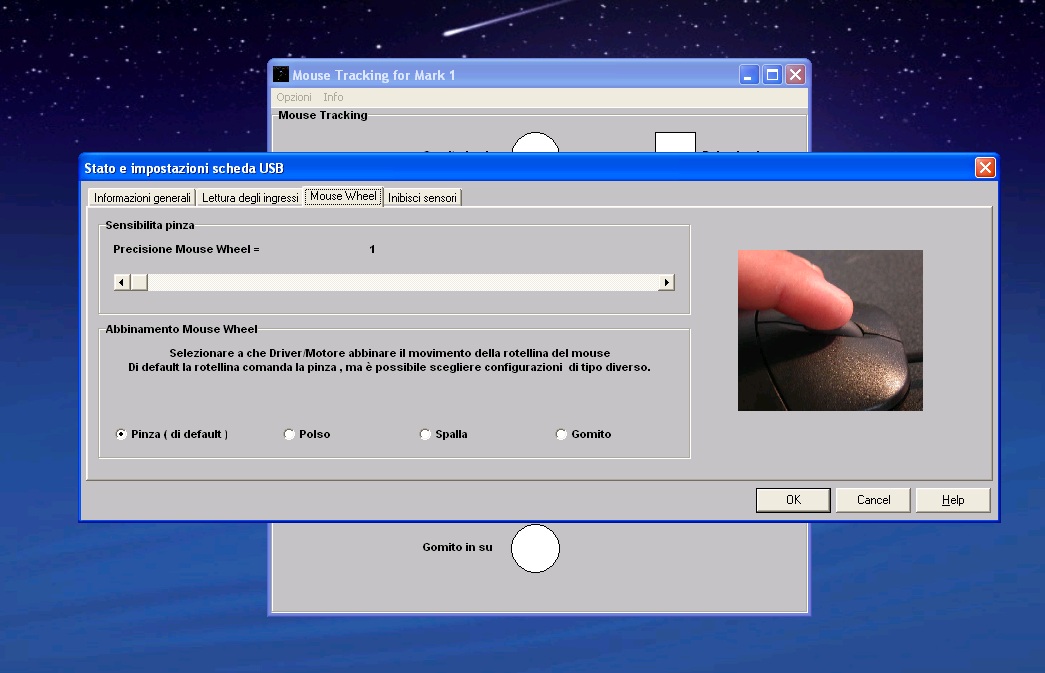
Un'altra interessante funzione del software di controllo manuale è quella di poter controllare una delle cinque articolazioni attraverso l'uso della rotella del mouse in contemporanea con i comandi della tastiera .

In questo modo è possibile , per esempio , con la mano destra controllare il polso e con la sinistra muovere a destra o a sinistra il braccio.

Come mostrato in Figura3.8 , all'interno del software di controllo manuale è presente una finestra che permette si impostare la durata dell'impulso che andrà a comandare l'articolazione del braccio e il tipo di articolazione alla quale si vuole mandare tale impulso.

Per esempio è possibile , con uno singolo movimento in giù della rotella del mouse , far scendere la spalla del braccio robotico per 10 secondi .

Di default , la rotella del mouse è sempre abbinata alla pinza del braccio , in questo modo è possibile raggiungere un oggetto e raccoglierlo.



*Fig. 3.8 Menu di settaggio per il controllo dei movimenti attraverso la mouse Wheel*

La Figura 3.8 mostra la finestra di dialogo in cui è possibile settare il tempo di durata dell'impulso ed è possibile inoltre abbinare una delle articolazione ad uno dei vari servomotori.

Impostando la durata dell'impulso al minimo possibile , ovvero circa 500 µS si ottengono movimenti precisi al millimetro che permettono l'esecuzione di operazione particolari e molto complesse.

Un'altra interessante finestra è quella relativa alle informazioni di diagnostica riguardanti l'Interfaccia USB.

Tale finestra riporta tutte le principali informazioni che possono interessare l'utente quali per esempio , il numero di ingressi e/o uscite disponibili , la velocità di comunicazione e la velocità di re flash con cui vengono aggiornate costantemente le informazioni in ingresso alla scheda .

Altra funzione importante di questo software è la possibilità di comandare la spalla e la base rotante attraverso i quattro tasti della tastiera quali : W,A,S,D .

Questa funzione permette quindi l'interazione simultanea del braccio utilizzando la tastiera e il mouse.

**3.2 Software di controllo automatico**

Il secondo software fornito con il braccio robotico è utilizzato per realizzare complessi automatismi che simulano il funzionamento dei bracci robotici nelle industrie .

Tale software permette di realizzare sequenze di movimenti prestabiliti che rispondo alle esigenze specifiche per l'utente , come per esempio l'automazione di un impianto industriale o di una catena di produzione .

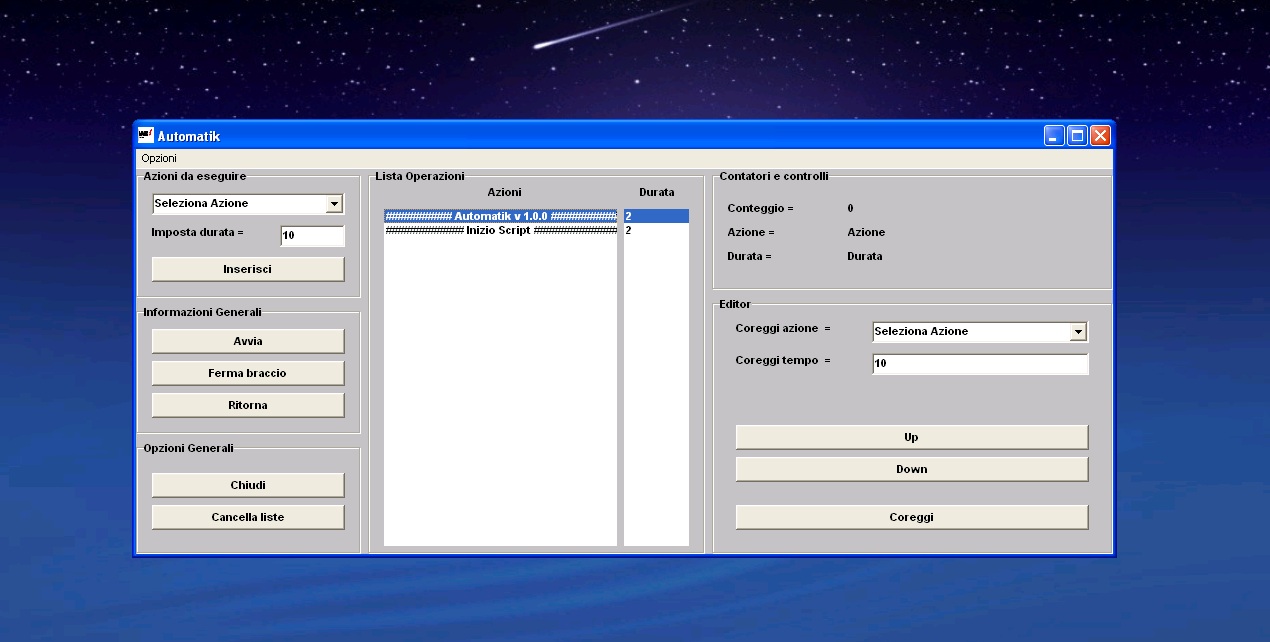
Il software si presenta con una singola finestra principale contendente due liste specifiche , una riguardante la lista delle azioni e un'altra , sincronizzata con la precedente , contenente la lista dei tempi .

Ogni azione , viene letta per una durata di tempo prestabilita dalla lista dei tempi e terminata tale azione , il software passa automaticamente alla lettura ed esecuzione dell'azione successiva .

Come mostra la figura 3.9 , il software , oltre ad offrire le liste di caricamento , permette anche di effettuare movimenti di andata e di ritorno precisi al millimetro.

In oltre , è possibile effettuare movimenti che terminano di essere eseguiti nel momento in cui il finecorsa relativo al servomotore che si sta muovendo.

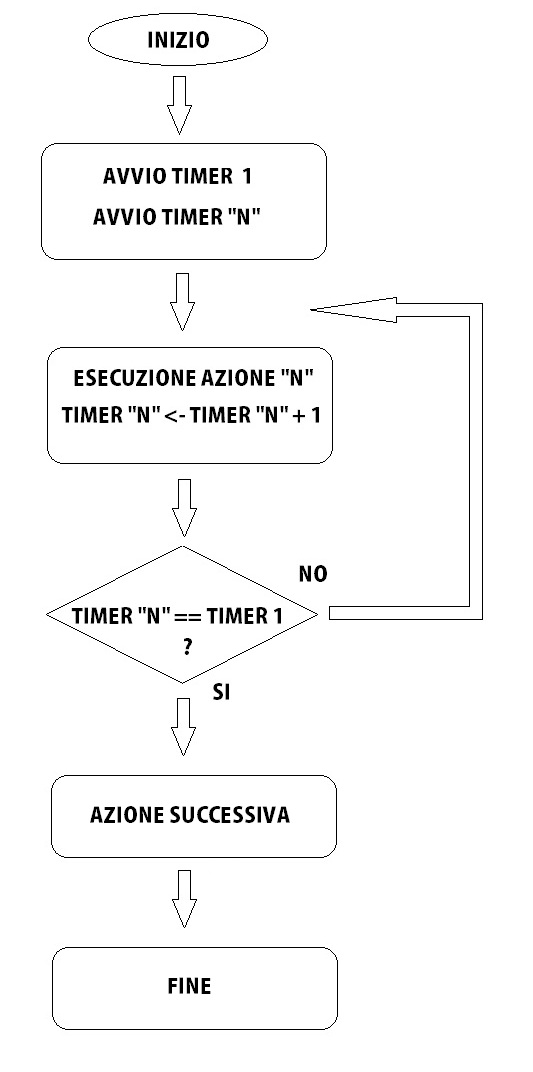
Sono presenti anche vari programmi preinstallati che permettono una gestione più semplificata del braccio , come ad esempio programmi che permettono di riposizionare il braccio o di spostare un oggetto da un luogo predefinito ad un'altro.



*Fig. 3.9 Finestra principale del software di controllo automatico.*

Oltre ai controlli per l'esecuzione delle operazioni automatiche , il software permette anche il caricamento o il salvataggio della lista delle varie operazioni , in modo da poter salvare i propri programmi e ricaricarli ogni volta che necessita .

Il funzionamento del programma è analizzato nella schema di Figura 4.0 , in cui , attraverso uno schema a blocchi viene mostrato in che modo funziona la parte principale del software di gestione automatica.



*Fig. 4.0 Schema a blocchi semplificato del programma di gestione automatica*

Il software permette inoltre di gestire basilari processi di scrittura attraverso l'uso combinato di diverse azioni che permettono di tracciare in maniera predefinita una parola composta da un'insieme di lettere .

Un'altra importante funzione del software di controllo automatico è la possibilità di caricare programmi scritti su un semplice Editor di testo quale per esempio Blocco Note di Windows ©.

Quindi è possibile scrivere , leggere e editare il programma anche su un Personal Computer in cui non è installato il software di controllo .

Inoltre , i programmi scritti in questo modo hanno la capacità di essere molto leggere e di essere letti su qualsiasi sistema operativo come per esempio Macintosh © .

**3.4 Accenno ad automazioni di tipo complesse**

Le automazioni di tipo complesse sono automazioni utilizzate specificatamente all'interno delle grandi industrie .

Tali automazioni permettono la produzione sequenziale , rapida e precisa di una quantità infinita di oggetti .

Attraverso l'uso del software di controllo automatico è possibile eseguire infinite operazioni sequenziali connesse l'una con l'altra attraverso eventi prestabiliti dall'utente o che variano nel tempo .

Il software automatico permette di eseguire ben 19 operazioni differenti , suddivide in tre categorie : Generiche , Automatiche o di routine.

|  |  |  |
| --- | --- | --- |
| **Generiche** | **Automatiche** | **Routine** |
| Muovi a destra  Muovi a sinistra  Spalla in su  Spalla in giu  Gomito in su  Gomito in giu  Polso in su  Polso in giu  Apri pinza  Chiudi pinza  Accendi Led  Spegni Led | Muovi a sinistra - auto  Muovi a destra - auto  Spalla in su - auto  Spalla in giu - auto  Gomito in giu - auto  Polso in giu - auto | Skip  Open Viewer  Close Viewer  End Terminal  Return |

*Fig. 4.1 Tabella azioni Generiche , Automatiche o di Routine*

L'azione "End terminal" è classificata come azione di Routine anche se quest'ultima è un'azione che si presenta una sola volta durante tutto il programma , precisamente alla fine di esso.

Ogni programma come in qualsiasi software di programmazione contiene un Header senza il quale il software non riesce a riconoscere alcun azione scritta successivamente .

L'Header e un semplice programma è rappresentato nella Figura 4.2 , in cui è possibile osservare la stressa correlazione esistente tra la lista delle azioni e la lista dei tempi .

2

2

200

93

50

50

30

2

2

########### Automatik v 1.0.0 ############

############# Inizio Script ##############

Muovi a destra - auto

Muovi a sinistra - auto

Spalla in su - auto

Gomito in giu - auto

Polso in giu - auto

End Terminal

########### END TERMINAL ############

*Fig. 4.2 Codice di un programma*

Il codice riportato rappresenta un semplice programma che permette di posizionare il braccio al centro della base e di portare in posizione "iniziale" la spalla e il gomito.

Sviluppando e creando programmi più complessi è possibile rendere più sofisticate e precise le operazioni di gestione del braccio.

Un'altra valida applicazione dimostrativa potrebbe riguardare lo spostamento di un determinato oggetto ( da una posizione prestabilita ) ad un'altra posizione , attraverso l'uso dei sensori di finecorsa o di sensori aggiuntivi installabili in un secondo momento.

Ovviamente la lista delle azioni deve essere sempre affiancata alla relativa lista dei tempi , altrimenti si corre il rischio di annullare la stretta relazione coesistente tra le due liste .

Inoltre è fondamentale controllare sempre lo stato del braccio robotico per evitare che vi siano oggetti che possono ostruire il passaggio del braccio durante i suoi movimenti.

I movimenti complessi che può far eseguire il software di controllo al braccio sono infiniti e permettono di realizzare una varietà di combinazione di automazioni .

Ovviamente , il limitato numero di sensori e la prevalente funzione dimostrativa di questo progetto ne limitano fortemente le espandibilità , ma l’utilità di questo progetto è principalmente quella di mostrare la perfetta integrazione che si può realizzare da un computer ed una scheda elettronica esterna interfacciata con la porta di comunicazione standard USB.

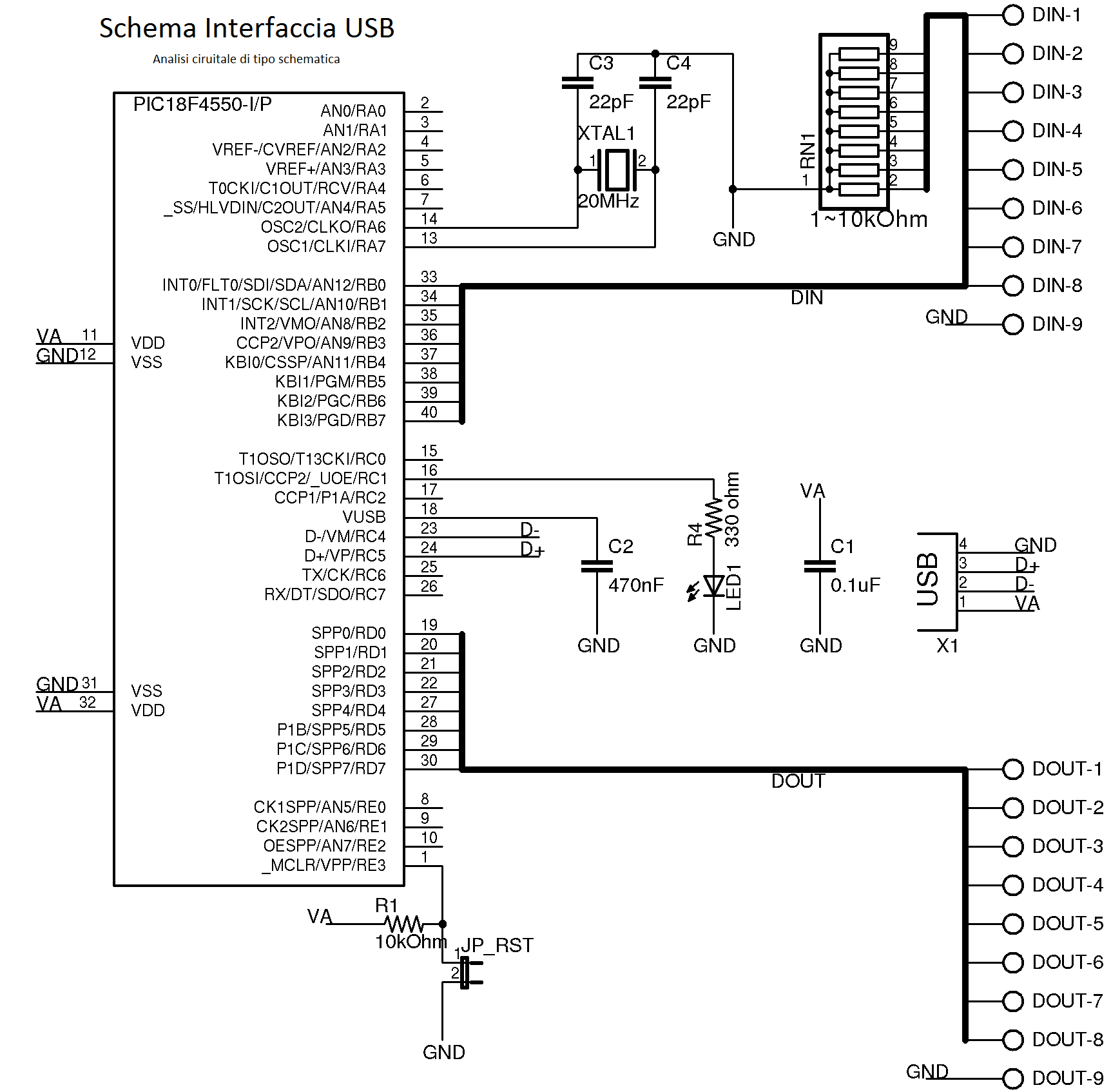
**CAP. 4**

**Realizzazione pratica del progetto**

**4.1 Realizzazione dell'Interfaccia USB**

L’interfaccia USB è stata realizzata seguendo i più alti standard di precisione e correttezza circuitale possibile .

Ogni componente e ogni collegamento elettrico è stato realizzato per evitare qualsiasi eventuale problema durante la realizzazione pratica della scheda .

****

*Fig. 4.3 Schema Interfaccia USB*

Attraverso l’analisi del circuito in figura 4.3 è stato possibile ridisegnare in maniera corretta e precisa il file PCB e il disegno tecnico della scheda.