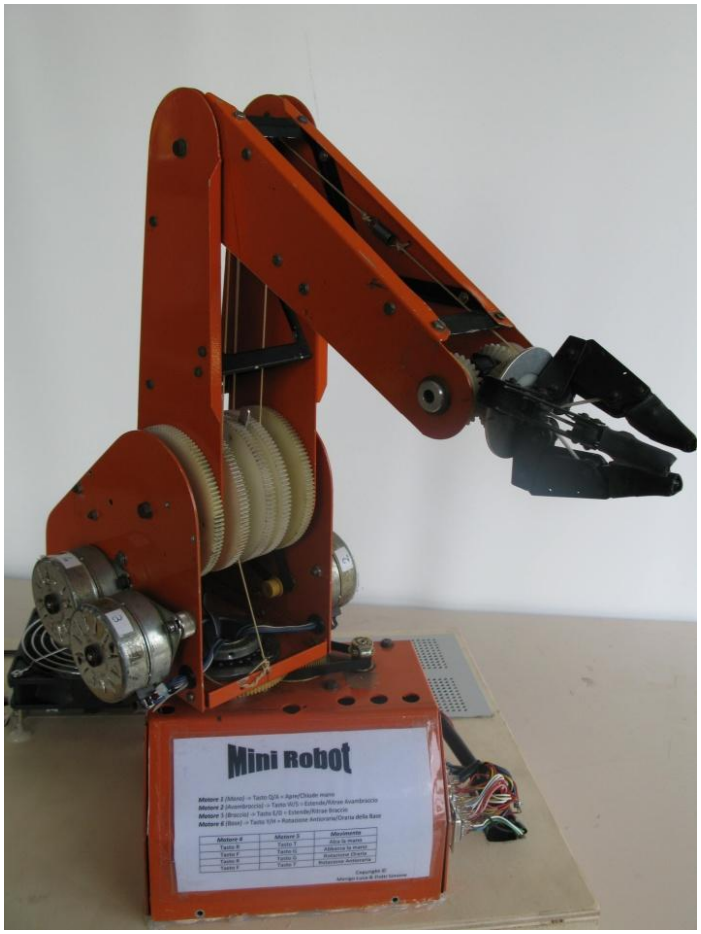
ITIS Castelli Brescia
Istituto Tecnico Industriale Statale
Liceo Scientifico Tecnologico

Mini Robot

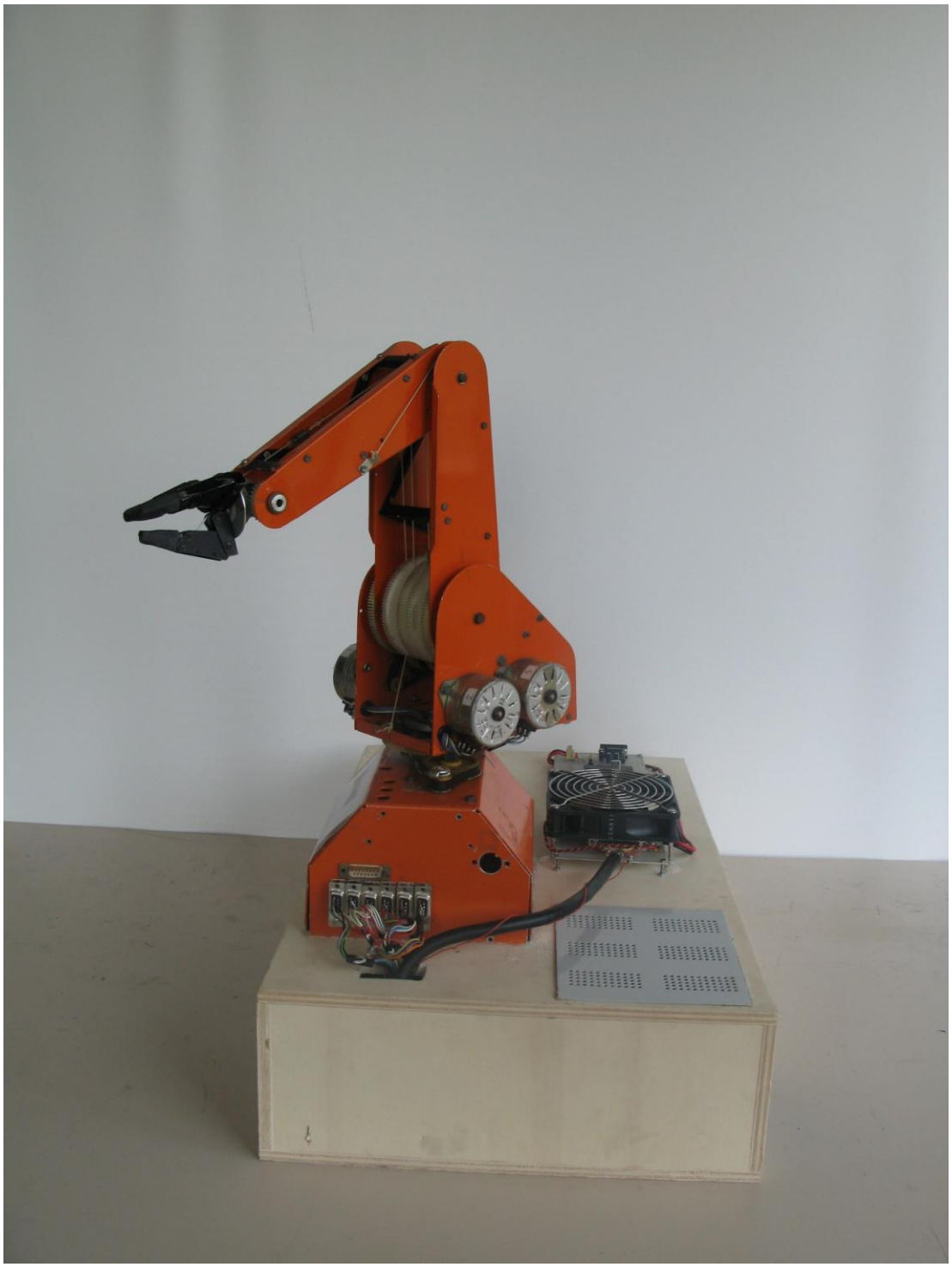
Pilotaggio di un braccio meccanico in miniatura

Merigo Luca e Dotti Simone
5°E
Elettronica e Telecomunicazioni

Anno scolastico 2009-2010



Indice



Parte Prima

Introduzione

1.1) L’automazione industriale

1.1.1) Definizione

Il termine automazione identifica la tecnologia che usa sistemi di controllo per gestire macchine e processi, riducendo la necessità dell'intervento umano. Si realizza per l'esecuzione di operazioni ripetitive o complesse, ma anche ove si richieda sicurezza o certezza dell'azione o semplicemente per maggiore comodità.

1.1.2) Origine del termine

Il termine Automazione fu coniato nell'industria automobilistica nell'immediato dopoguerra per descrivere l'accresciuto uso di dispositivi automatici e di controllo nelle linee di produzione meccanizzate. L'origine della parola è attribuita a D.S. Harder, un dirigente della Ford Motor Company. Il termine è utilizzato estesamente in un contesto manifatturiero, ma è anche utilizzato ogniqualvolta ci sia una significativa sostituzione del lavoro e dell'intelligenza umana con azioni di tipo informatico, elettronico e meccanico.

1.1.3) L’evoluzione dell’automazione

Per automazione si intende l'applicazione di macchine a compiti una volta effettuati dall'uomo o, sempre più in modo crescente, a compiti che altrimenti non potrebbero essere neanche realizzati. Sebbene il termine meccanizzazione sia spesso usato per riferirsi alla semplice sostituzione del lavoro umano da parte delle macchine, l'automazione in genere implica l'integrazione di macchine in un sistema che si autogoverni. L'automazione ha certamente rivoluzionato quelle aree in cui è stata introdotta; ci sono pochi aspetti della vita moderna che non siano stati influenzati da essa. Sebbene le differenze tra automazione e meccanizzazione possano difficilmente essere messe in evidenza in pratica, le differenze tra la società automatizzata (come quella odierna) e quella meccanizzata (come quella alla fine dell'ottocento) si possono più facilmente comprendere; l'introduzione dell'automazione è stata la causa di un nuovo e distinto impulso allo sviluppo della civiltà industriale.
Gli uomini si sono sempre sforzati di trasferire alcuni dei carichi del lavoro a dispositivi meccanici: esempi di questi meccanismi sono le carrucole, gli argani, i sistemi di sollevamento, dei quali sono state trovati dei reperti che datano al terzo millennio AC. Tuttavia, una meccanizzazione estesa ed un incorporamento esteso di macchine in sistemi autogovernati non ebbe luogo fino alla Rivoluzione Industriale nel XVIII secolo. Durante la Rivoluzione Industriale furono sviluppate fabbriche che producevano parti intercambiabili per prodotti diversi, con la conseguente divisione delle diverse lavorazioni ad operai, ad ognuno dei quali veniva riservato un compito specifico che veniva da questo eseguito un numero imprecisato di volte. Da quel momento, fu immediata l'esigenza di sviluppare macchine (originariamente a vapore e successivamente elettriche) che eseguissero questi compiti.
Numerosi e significativi sviluppi si sono succeduti in vari campi durante il XX secolo: i calcolatori digitali, i miglioramenti nelle tecnologie per la registrazione dei dati e nel software per la scrittura di programmi, il progresso nella tecnologia dei sensori, e la definizione di una teoria matematica del controllo. Tutti questi avanzamenti hanno contribuito al progresso nelle tecnologie dell'Automazione.

1.1.4) Componenti di un' automazione

Ogni sistema di automazione è composto principalmente da tre parti:
-Struttura Meccanica
- Schede Elettroniche e sensoristica
-Programma software
Ognuna di queste componenti è strettamente dipendente dalle altre.
Basta pensare ad esempio ad un comune autolavaggio: le spazzole utilizzate per il lavaggio dell'autovettura sono azionate da motori elettrici che a loro volta vengono gestiti da una scheda elettronica programmata.
Il continuo sviluppo di queste tre componenti permette la realizzazione di sistemi sempre più complessi, sofisticati e precisi, con capacità di apprendimento e di diagnostica.

Parte **Seconda**

Introduzione al progetto

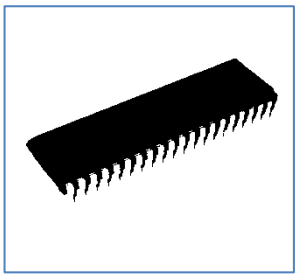
2.1) Presentazione del progetto

Nell'industria moderna una delle automazioni più diffuse è costituita da manipolatori meccanici che presentano articolazioni e movimenti tipici del braccio umano.
Infatti una delle loro caratteristiche salienti è la versatilità: avendo pressoché totale libertà di movimento in tutte le direzioni, essi possono svolgere svariate funzioni quali lo spostamento dei pezzi o la loro lavorazione, fino ad arrivare ad operazioni di verniciatura e saldatura. Uno dei pregi di questo tipo di automi è sicuramente la precisione, che nei modelli più recenti ha raggiunto livelli tali da poter effettuare lavorazioni micrometriche o particolarmente pericolose, come ad esempio lo spostamento di materiali tossici o esplosivi.
L'obiettivo del progetto è appunto la gestione di una riproduzione in scala ridotta di una di queste automazioni, comandata e programmata attraverso un PC. In particolare essa deve permettere la memorizzazione, ottimizzazione e riproduzione di un percorso realizzato dall'utente.
Ovviamente il grado di precisione e accuratezza di questo manipolatore non può essere confrontata con quello degli esempi sopracitati, principalmente per limitazioni dovute alla struttura meccanica e di azionamento.
D'altra parte l'algoritmo e la logica di programmazione utilizzate per questo progetto possono essere reimpiegate per automazioni di dimensioni e qualità maggiore.

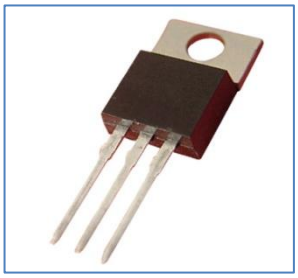
2.2 Schema a Blocchi

Interfaccia a Computer

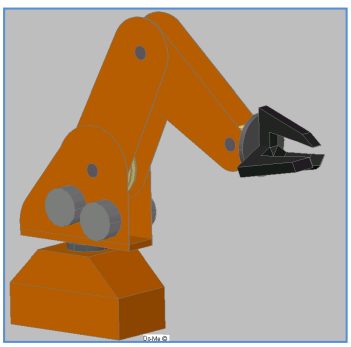
Trasmissione RS232



Microcontrollore



Interfaccia di potenza



Braccio meccanico

Il sistema è composto da un' interfaccia grafica realizzata con il software Delphi su PC che, attraverso l'interfaccia seriale RS-232, comunica con un microcontrollore, il quale pilota, grazie ad un interfaccia di potenza, i vari motori posti sulla parte meccanica.

-Interfaccia a PC

Il programma realizzato con il software Delphi permette all'utente, attraverso un' interfaccia grafica, di comandare il braccio meccanico mediante la tastiera del PC. Inoltre sono presenti funzionalità di memorizzazione e di riproduzione di varie sequenze di azioni attivabili con degli appositi pulsanti "virtuali". È anche possibile salvare i movimenti compiuti su file esterni in modo da poterli ricaricare e riprodurre in un secondo momento. I percorsi memorizzati vengono inoltre ottimizzati dal software in fase di riproduzione: il programma elabora le velocità a cui devono essere pilotati i motori per eseguire il movimento richiesto nel modo più veloce ed efficiente possibile.

-Interfaccia seriale RS-232

Per la realizzazione della comunicazione tra PC e microcontrollore è stato ideato un apposito protocollo di trasmissione utilizzando l'interfaccia RS-232, in modo da minimizzare il numero di caratteri inviati e permettere così migliori prestazioni e maggior chiarezza.
A seconda della situazione in cui il sistema si trova, avvengono le seguenti comunicazioni:
-Movimento libero: il PC invia al microcontrollore una sequenza di sei caratteri che indicano i motori da attivare e il verso di rotazione.
-Memorizzazione percorso: il microcontrollore invia al PC il numero dei passi effettuati dai motori.
-Riproduzione percorso: il PC invia al microcontrollore caratteri che indicano i motori da muovere, il verso, la velocità a cui devono essere pilotati e i passi che ogni motore deve compiere per permettere l'esecuzione ottimizzata del percorso precedentemente memorizzato.
Durante l'utilizzo del braccio il PC provvede inoltre ad inviare un carattere di controllo ogni intervallo di tempo prestabilito in modo da verificare la corretta comunicazione tra i due apparati (handshake).

-Microcontrollore

Il microcontrollore ha il compito di interpretare i dati ricevuti dal PC e pilotare i motori del braccio. Esso deve inoltre, in fase di memorizzazione del percorso, contare ed inviare al PC il numero di passi effettuati dai vari motori. Inoltre segnala con l'accensione di due led se la trasmissione funzione e se il programma è in esecuzione.

-Interfaccia di potenza

L'interfaccia di potenza è costituita da transistor in configurazione Darlington. Essi vengono pilotati dal microcontrollore in modo da convertire segnali di bassa potenza (logica TTL 0-5V) in segnali adatti a pilotare i motori passo-passo presenti sul braccio (12V).

-Struttura meccanica

Il braccio meccanico è mosso da un sistema di carrucole e funicelle comandate da motori passo-passo posti sulla base della struttura meccanica.
Esso è fissato su un supporto di legno su cui sono situate anche le schede elettroniche e l'alimentatore.

Per ulteriori specifiche e approfondimenti si rimandano le spiegazioni ai paragrafi corrispondenti ai determinati argomenti.

2.3) Abstract

-Project presentation

In the modern industry one of the most diffused automation consists of mechanical manipulators, that present articulations and movements similar to the human arm.
One of their main characteristics is their versatility: with almost total freedom of movement in all directions, they can perform various functions such as moving objects, painting and wielding operations.
One of the advantages of this automation is certainly its precision, that allows micrometrical and dangerous processing, such as the movement of toxic and inflammable objects.
The aim of this project is the management of a reproduction in miniature of one of these automations, controlled and programmed trough the PC. In particular it must allow the memorization, optimization and reproduction of a route realized by the user.
Obviously, the grade of precision and accuracy of this manipulator cannot be compared with the examples mentioned above, mainly because of limitations of the mechanical and driver structure.
On the other hand the algorithm and the logic of the program used for this project can be reused for larger and with higher quality automations.

The system consists of a graphic interface, realized through Delphi software that, across the serial interface protocol RS232, communicates with a microcontroller, that pilots, with a power interface, the various motors placed on the mechanical structure.

-PC interface

The program realized with Delphi software allows the user, through a graphic interface, to control the mechanical arm with the pc keyboard. It presents also memorization and reproduction features, than can be activated trough the pressure of virtual buttons. It is also possible to save movements and actions performed by the arm, in order to reproduce them in a second moment.
When a route is reproduced, it is optimized by the software, in order to execute the movements in the more efficient and faster way.

-Serial interface RS232

For the communication between the PC and the microcontroller it was designed a dedicated protocol of transmission using the RS-232 interface, in order to minimize the loss times and allow a better performance and clarity.
Depending on the situation in which the system is, the following communications occur:

-Free movement: the PC sends to the microcontroller a sequence of six characters that indicate the movements and the direction of rotation of each motor;

-Memorization of the path: the microcontroller sends the PC the number of steps taken by each engine.

-Reproduction of the path: the PC sends to the microcontroller characters that indicate the engines to be moved, the direction, the speed required to be driven and the steps that each engine must do to allow the execution of the optimized path previously stored.

During the execution of the program, the PC also sends a check character at each interval of time in order to verify the correct communication between the two devices (handshake).

-Microcontroller

The microcontroller has the task of process the data received from the PC and to drive the motors of the arm. It must also, at the memorization point, count and send to the PC the number of steps taken by the various engines. It also indicates , with two LEDs on the PCB, if the transmission functions and if the program is running.

-Power interface

The power interface consists of transistors in Darlington configuration. They are driven by the microcontroller in order to convert low power signals (0-5V TTL logic) into signals suitable for driving the stepper motors on the arm (12V).

-Mechanical structure

The mechanical arm is moved by a system of ropes and pulleys controlled by stepper motors located on the base of the mechanical structure.
It is mounted on a wooden stand on which the electronic boards and power supply are also located.

Parte Terza

Cenni Teorici

3.1) I microcontrollori

3.1.1) Definizione

Un microcontrollore è un sistema a microprocessore completo, integrato in un solo chip, progettato per ottenere la massima autosufficienza funzionale ed ottimizzare il rapporto prezzo-prestazioni per una specifica applicazione, a differenza, ad esempio, dei microprocessori impiegati nei personal computer, adatti per un uso più generale.

3.1.2) Origini e caratteristiche

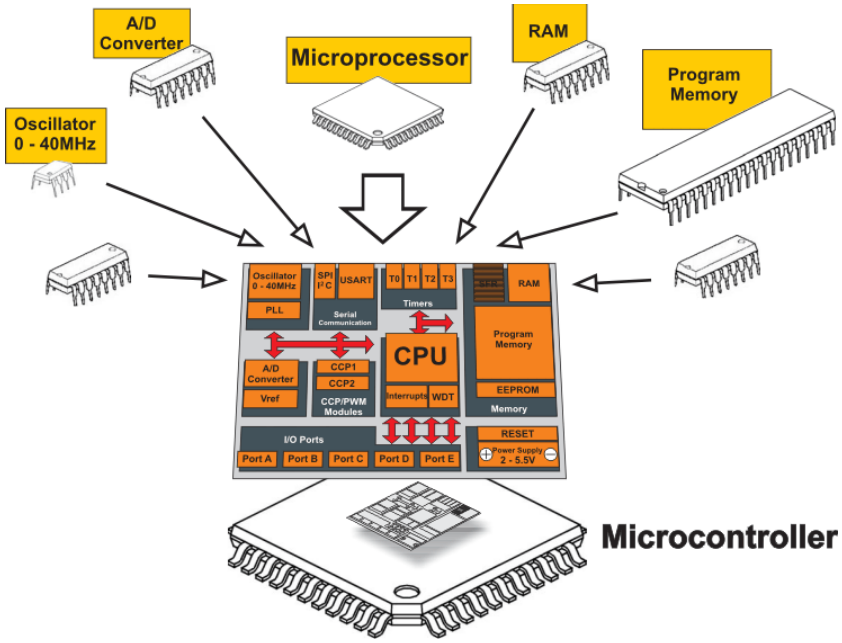
Un microcontrollore (o MCU) è un dispositivo elettronico che opportunamente programmato è in grado di svolgere diverse funzioni in modo autonomo. Essenzialmente gestisce delle linee di input e di output in relazione al programma in esso implementato. Al loro interno trova spazio un vero e proprio microprocessore completo di CPU, RAM, Timer e numerose linee di ingresso/uscita.
A differenza dei microprocessori più evoluti, nei microcontrollori il programma è contenuto all'interno in un'apposita area di memoria (non volatile) e viene eseguito ciclicamente; anche la RAM per i dati volatili è all'interno dello stesso dispositivo ed alcuni dispongono di aree dati non volatili e riscrivibili (EEPROM). Lo stadio di IN/OUT è già implementato all'interno ed alcuni dispongono già di interfacce per segnali analogici, per comparatori o per comunicazioni seriali.
In effetti ciò che distingue un microcontrollore da un microprocessore è che quest' ultimo per poter funzionare ha bisogno di componenti aggiuntivi, come memorie o componenti per la ricezione e spedizione di dati, mentre l'MCU è progettato per contenere quanto necessario e quindi non necessita di alcun componente aggiuntivo.
I microcontrollori sono usati soprattutto laddove la potenza di calcolo non è di fondamentale importanza: controllori embedded e MCU sono usati spesso in robotica dove la potenza e la complessità del calcolo sono distribuite tra i singoli task utilizzando un gran numero di microcontrollori con piccola potenza di calcolo. Inoltre la possibilità di comunicazione tra ogni singolo controllore ed un controllore di maggiore potenza (computer o mainframe) rende possibile l'elaborazione delle informazioni prodotte da ogni singolo MCU.
Le applicazioni degli MCU sono molteplici. Sono usati con successo in applicazioni di monitoring e data recording, dove le caratteristiche vincenti sono il basso consumo e l'auto-riattivazione mediante interrupt, facilmente programmabili. Grosso impiego trovano anche nel campo della Domotica, in cui si crea in genere una rete di microcontrollori per formare un sistema centralizzato o decentralizzato per il controllo dell' intera "Domus". Esempio tipico è il controllo di temperatura: se d'estate la temperatura si alza sopra certi limiti il sistema attua l'accensione dell'impianto di condizionamento d'aria fino a quando essa non scende sotto la soglia che l'utente ha prefissato.
I microcontrollori hanno origine dalla filosofia nata con i primi PLC: integrazione del core di un processore in un ambiente differente rispetto alla macchina general-purpose, per la risoluzione di problemi specialistici (prevalentemente in logica booleana). Molti MCU hanno origine dalle prime versioni di microprocessori come lo Zilog Z80, l'Intel 8088 e il Motorola 6809. Man mano che il processo di miniaturizzazione è avanzato, tutti i componenti necessari per un controllore sono stati inseriti in un unico chip e così è nato il microcontrollore.
La tecnologia costruttiva più diffusa è la CMOS che richiede basse potenze permettendo l'alimentazione

con batterie. I chip CMOS permettono il rallentamento della frequenza di clock (o lo stop) fino a mettere l'MCU in sleep mode ed hanno un'alta immunità alle interferenze elettro-magnetico (fluttuazione dell'alimentazione e picchi di corrente).

Nella foto il primo microcontrollore prodotto: l' 8048 di Intel, rilasciato nel 1975, con RAM e ROM sullo stesso chip. Questo componente è stato utilizzato in più di un miliardo di tastiere per PC e numerose altre applicazioni.

3.1.3) Struttura di un microcontrollore

Un microcontrollore è in genere costituito dai seguenti componenti:
1) Microprocessore a 4, 8, 16, 24 o 32 bit.
2) Memoria RAM di dimensioni ridotte, usata solo per la memorizzazione delle variabili intermedie e delle variabili di ingresso e di uscita.
3) Memoria EPROM per la memorizzazione del programma che il controllore deve eseguire.
4) Dispositivi di I/O per la lettura e la generazione di segnali particolari, verso i quali il controllore è prodotto già specializzato.
5) Memoria aggiuntiva, di tipo EEPROM, che permette all'MCU di memorizzare variabili in modo non volatile, così da preservarle anche in caso di mancanza di alimentazione.
6) Contatori di tempo, o timers.
7) Moduli controllori di interrupt.
8) Altri moduli aggiuntivi, quali convertitori Analogici/Digitali, convertitori Digitali/Analogici, moduli per le comunicazioni seriali su Bus, e così via.

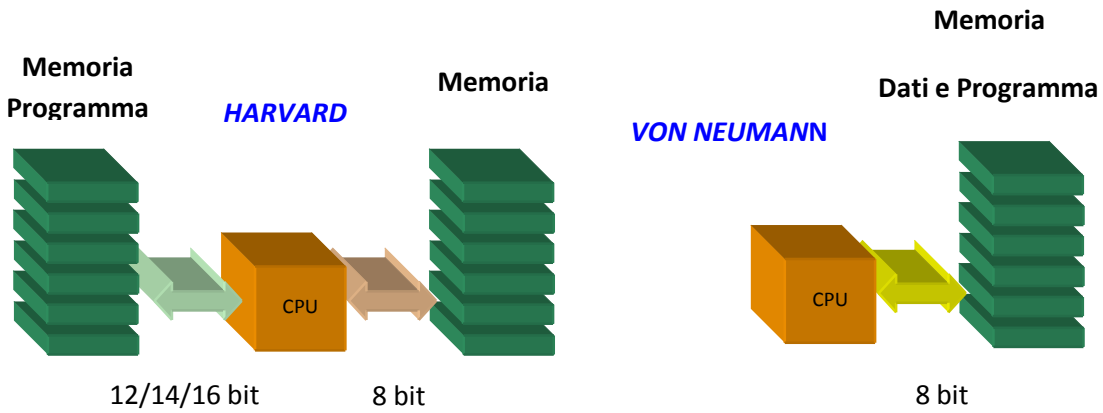


3.1.4) L'architettura di un microcontrollore

Le architetture tipiche di un MCU sono:

1)Harvard:
-Bus separato per i dati e le istruzioni.
-Memoria programma separata da quella dati.

2)Von Neumann:
-Bus singolo condiviso per i dati e per le istruzioni.
-Memoria comune per i dati e le istruzioni.

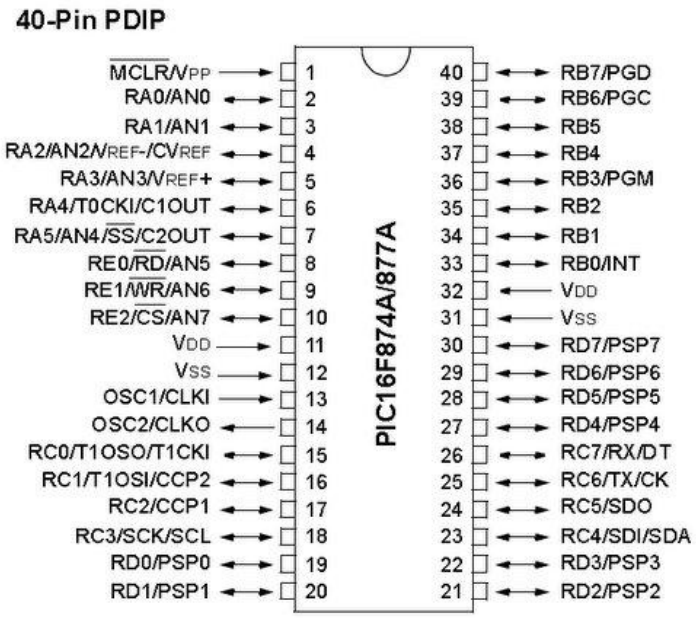


Nell'architettura di Von Neumann per ogni comando c'è la necessità di prelevare prima l'istruzione e quindi i dati, ossia avere due accessi in memoria: ciò può rallentare non poco il dispositivo. Questo limite è superato con un'architettura di tipo Harvard, in cui il processore è in grado di accedere in modo indipendente a dati e istruzioni in quanto sono separati e memorizzati in memorie separate. L'architettura Harvard, quindi, può eseguire più compiti contemporaneamente, effettuando in parallelo le operazioni di lettura e scrittura della memoria. L'aumento di velocità viene compensato dalla presenza di circuiti più complessi all'interno del processore.

3.1.5) Il microcontrollore del MINI ROBOT: PIC 16F877A

Il microcontrollore impiegato in questo progetto è il PIC 16F877A della Microchip.
La scelta è stata dettata da esigenze meccaniche e specifiche di progetto.
Era necessario infatti avere un controllore che lavorasse a velocità elevata, che disponesse di un modulo interno per la trasmissione seriale (per la comunicazione RS-232 con PC) e, in particolare, che avesse un elevato numero di pin utilizzabili come uscite. Quest'ultima specifica si è resa necessaria in quanto il PIC doveva pilotare sei motori passo-passo ognuno dei quali aveva quattro fasi da connettere, attraverso un apposita interfaccia di potenza, al microcontrollore (per un totale di 24 ingressi).
Il PIC 16F877A soddisfa a tutte queste specifiche.
Esso è stato programmato utilizzando MPLAB della Microchip come software per la scrittura del programma sorgente e il CC5X della Knudsen come compilatore.

3.1.6) Piedinatura e descrizione PINOUT



Pin Name	DIP Pin#	SOIC Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	9	9	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	10	10	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, the OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	1	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
					PORTA is a bi-directional I/O port.
RA0/AN0	2	2	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	3	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	4	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	5	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	6	I/O	ST	RA4 can also be the clock input to the Timer0 module. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	7	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	21	21	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	22	22	I/O	TTL	
RB2	23	23	I/O	TTL	
RB3/PGM	24	24	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	25	25	I/O	TTL	Interrupt-on-change pin.
RB5	26	26	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	27	27	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	28	28	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.
					PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	11	11	I/O	ST	RC0 can also be the Timer1 oscillator output or Timer1 clock input.
RC1/T1OSI/CCP2	12	12	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	13	13	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	14	14	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	15	15	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	16	16	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	17	17	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	18	18	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
VSS	8, 19	8, 19	P	—	Ground reference for logic and I/O pins.
VDD	20	20	P	—	Positive supply for logic and I/O pins.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/ counter. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be soft-ware programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

3.1.7) Specifiche del PIC16F887A

Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F877
Operating Frequency	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	8K
Data Memory (bytes)	368
EEPROM Data Memory	256
Interrupts	14
I/O Ports	Ports A,B,C,D,E
Timers	3
Capture/Compare/PWM Modules	2
Serial Communications	MSSP, USART
Parallel Communications	PSP
10-bit Analog-to-Digital Module	8 input channels
Instruction Set	35 instructions

3.2) L’interfaccia seriale RS-232

3.2.1) Definizione

L'interfaccia seriale RS232 è uno standard costituito da una serie di protocolli meccanici, elettrici ed informatici che rendono possibile lo scambio di informazioni a bassa velocità tra dispositivi digitali.

3.2.2) Origini

Lo standard RS232 nacque nei primi anni '60 per opera della Electronic Industries Association (EIA) ed era orientato alla comunicazione tra i mainframe e i terminali attraverso la linea telefonica, utilizzando un modem. Esso includeva le caratteristiche elettriche dei segnali, la struttura e temporizzazioni dei dati seriali, la definizione dei segnali e dei protocolli per il controllo del flusso di dati seriali su un canale telefonico, il connettore e la disposizione dei suoi pin ed infine il tipo e la lunghezza massima dei possibili cavi di collegamento.
Nel corso di questi oltre 40 anni lo standard si è evoluto pur mantenendosi in larga parte invariato. L'evoluzione è riconoscibile dalla sigla, leggendo l'ultima lettera; l'ultima revisione è del 1997 ed è indicata come RS232f. Probabilmente la versione più diffusa è la RS232c, del 1969, corrispondente alle specifiche europee CCITT della raccomandazione V.24/V.28.
Pur essendo un protocollo piuttosto vecchio, attualmente lo standard RS232 è ancora largamente utilizzato per la comunicazione a bassa velocità tra microcontrollori, dispositivi industriali ed altri circuiti relativamente semplici che non necessitano di particolare velocità; è invece praticamente scomparsa in ambito "desktop", nel quale questo standard era nato per la comunicazione tra computer ed modem.

3.2.3) Caratteristiche generali

L'interfaccia RS-232 utilizza una trasmissione seriale asincrona; il segnale elettrico è non bilanciato; il collegamento è di tipo point-to-point. Di seguito si spiegano i significati di questi termini:

-**Seriale** specifica che i bit che costituiscono l’informazione sono trasmessi sequenzialmente su un solo filo. Questo termine è in genere contrapposto a "parallelo", termine che indica una tipologia di trasmissione in cui i dati viaggiano contemporaneamente su più fili, per esempio 8, 16 o 32. Parlando astrattamente si potrebbe pensare che la trasmissione seriale sia intrinsecamente più lenta di quella parallela (su di un filo possono passare meno informazioni che su 16...). In realtà questo non è vero, soprattutto a causa della difficoltà di controllare lo skew, cioè il disallineamento temporale tra i vari segnali, sempre presente nel caso di molti trasmettitori in un bus parallelo; per esempio in una fibra ottica, in un cavo ethernet, USB o FireWire, in un bus PCI-Express (tutti standard seriali) le informazioni transitano ad una velocità spesso superiore a quella di un bus PCI a 32 fili.

- **Asincrono** significa che i dati sono trasmessi senza l’aggiunta di un segnale di clock, cioè il clock del trasmettitore e del ricevitore sono indipendenti; ovviamente sia il trasmettitore che il ricevitore devono comunque essere dotati di un proprio clock locale per poter interpretare correttamente i dati.

-Un segnale **non bilanciato** è caratterizzato dal fatto che la tensione associata al bit trasmesso o ricevuto viene misurata rispetto ad un riferimento comune detto massa. Nel caso dei segnali RS232 questa tensione può essere sia positiva che negativa.

- Una trasmissione è di tipo **point-to-point** (collegamento punto-punto) quando nella comunicazione è presente, per ciascun segnale utilizzato, un solo trasmettitore ed un solo ricevitore; tale termine può essere contrapposto al termine collegamento multipunto, che indica la situazione in cui, a fronte di un trasmettitore, esistono più ricevitori.

3.2.4) La velocità di trasmissione e la velocità di modulazione

Le unità di misura della velocità di modulazione e di rispettivamente trasmissione sono: il *Baud* ed il *bit per secondo* (bps o b/s), spesso trattate erroneamente come sinonimi.

-Il **baud** (o anche baud rate) è l'unità di misura della velocità di modulazione e rappresenta il numero di simboli (raggruppamento di più bit associati ad uno specifico livello di tensione) al secondo che passano sulla linea. Esso è associabile alla banda occupata dal segnale modulato. Il cavo utilizzato per la trasmissione viene dimensionato in funzione di tale valore: secondo il criterio di Nyquist serve una banda passante pari ad almeno alla metà del baud rate; nella pratica si utilizzano cavi con banda pari a

-Il **bps** indica, come dice il nome, quanti bit al secondo sono trasmessi lungo la linea. Questa è la velocità effettiva della trasmissione vista dai dispositivi digitali pari a Vt=1/Tb (dove Tb=durata del singolo bit). In generale il legame tra velocità di trasmissione e di modulazione è regolato dalla seguente relazione:

$$Vt = Vm * log_2 M$$

dove M rappresenta il numero di livelli o elementi dell'alfabeto di trasmissione.

Nel caso di trasmissione binaria (M=2 e A{0,1}) baud rate e bps coincidono numericamente, da cui la parziale equivalenza dei due termini. Nel caso di trasmissioni a più livelli (codifica multilivello M>2), invece, è possibile trasmettere con una sola transizione più bit, ottenendo un baud rate minore a parità di informazioni trasmesse, guadagnando in termini di minore occupazione di banda a spese di una maggiore complicazione circuitale e peggioramento del rapporto segnale/rumore.
Per esempio la codifica multilivello PAM5 (dove il bit rate per ciascun segnale è doppio del baud rate) permette alle reti Gigabit Ethernet di raddoppiare la velocità di trasmissione rispetto alla Fast Ethernet a parità di banda occupata e quindi usando gli stessi cavi.
Lo standard RS232 utilizza due livelli quindi il baud rate coincide numericamente con il bps.

3.2.5) Half-duplex e full-duplex

I due termini fanno riferimento alla situazione in cui due dispositivi si scambiano informazioni tra di loro, comportandosi entrambi sia da sorgente di informazioni (cioè da talker o, in sigla, Tx) sia da ricevitore (listener o Rx).

Half-duplex indica che la trasmissione è bidirezionale ma non contemporanea nei due versi: in un determinato istante uno solo dei due dispositivi emette segnali, l'altro ascolta. Quando è necessario, si scambiano di ruolo.

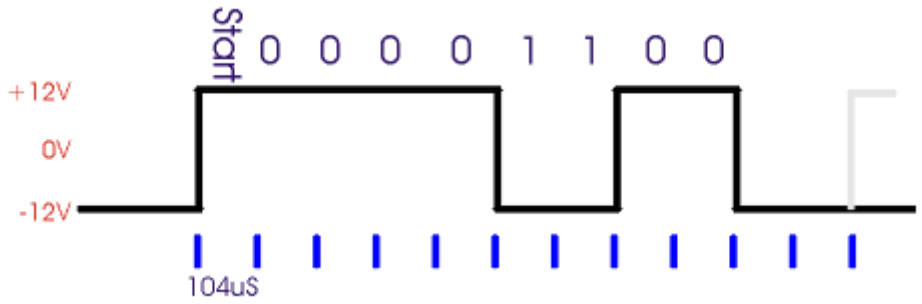
La trasmissione **full-duplex** indica che la trasmissione è bidirezionale e contemporanea. In questo caso sono necessari due fili, uno per ciascun verso di trasmissione. In alcuni sistemi di comunicazione, quali il comune telefono, possono essere adottati meccanismi che permettono la trasmissione full-duplex con un solo filo.

Se la trasmissione è sempre in un solo verso, si parla di **simplex**.

Lo standard RS232 permette tutte e tre queste modalità di funzionamento in quanto è utilizzato un conduttore separato per ciascun verso di trasmissione. In genere nel caso di trasmissione duplex è necessario che i dati in trasmissione e ricezione abbiano lo stesso formato e la stessa velocità. Inoltre ciascuno dei due nodi deve avere sufficiente potenza di calcolo per la gestione del duplice flusso di informazioni, condizione non sempre possibile quando la codifica del segnale è fatta solo utilizzando il software e senza assistenza di hardware dedicato.

3.2.6) Come è fatto un segnale RS-232

La cosa più semplice per descrivere un segnale RS232 è partire con un esempio.
Nell'immagine che segue è rappresentato quanto appare collegando un oscilloscopio ad un filo su cui transita un segnale RS-232 a 9600 bps del tipo 8n2 (più avanti verrà spiegata questa sigla) rappresentante il valore binario 00110000.



L'ampiezza del segnale è caratterizzata da un valore "alto" pari a circa +12V ed un valore "basso" pari a circa –12V. Da notare che, nello standard RS-232 un segnale alto rappresenta lo zero logico ed uno basso un uno, come indicato nel disegno.
A volte un segnale alto (+12V, cioè uno zero logico) è detto anche "space" ed uno basso (-12V, uno logico) è detto mark.
Tutte le transizioni appaiono in corrispondenza di multipli di 104 us, pari ad 1/9600 cioè ciascun bit dura esattamente l'inverso del bit rate.

La linea si trova inizialmente nello stato di riposo, bassa (nessun dato in transito); la prima transizione da basso in alto indica l'inizio della trasmissione (inizia il "bit di start", lungo esattamente 104us). Segue il bit meno significativo (LSB), dopo altri 104 us il secondo bit, e così via, per otto volte, fino al bit più significativo (MSB). Da notare che il byte è trasmesso "al contrario", cioè va letto da destra verso sinistra. Segue infine un periodo di riposo della linea di che può durare 1, 1,5 o 2 bit di stop. Nella figura ci sono 2 bit di stop pari a un periodo di riposo pari a 208us. Quindi (eventualmente) segue un nuovo pacchetto di bit con un nuovo bit di start (in grigio nel disegno).

Le varianti possibili sono le seguenti:
-se la trasmissione è più veloce o più lenta, la distanza tra i fronti varia di conseguenza (p.e. a 1200 bps le

transizioni avvengono a multipli di 0,833 ms, pari a 1/1200);
-invece di trasmettere 8 bit, ne posso trasmettere 6, 7 o anche 9 (ma quest'ultima possibilità non è prevista dalle porte seriali dei normali PC);
-alla fine è possibile aggiungere un bit di parità;
-dopo la trasmissione del pacchetto la linea rimane nello stato di riposo per almeno 1 o 1.5 o 2 bit.

In genere il formato del pacchetto trasmesso è indicato da una sigla composta da numeri e cifre, per esempio 8n1 e 7e2:
-La prima cifra indica quanti bit di dati sono trasmessi (nei due esempi rispettivamente 8 e 7);
-La prima lettera il tipo di parità (rispettivamente nessuna ed even-parity, cioè parità pari);
-La seconda cifra il numero di bit di stop (rispettivamente 1 e 2);
Tenendo conto che esiste sempre un solo bit di start, un singolo blocco di bit è quindi, per i due esempi riportati, costituito rispettivamente da 10 (1+8+0+1) e 11 (1+7+1+2) bit. Da notare che di questi bit solo 8 o, rispettivamente, 7 sono effettivamente utili.

Lo standard originale prevede una velocità fino a 20Kbps. Uno standard successivo (RS-562) ha portato il limite a 64Kbps lasciando gli altri parametri elettrici praticamente invariati e rendendo quindi i due standard compatibili a bassa velocità. Nei normali PC le cosiddette interfacce seriali RS-232 arrivano in genere almeno a 115Kbps, 230Kbps o anche più: pur essendo tali valori formalmente al di fuori di ogni standard ufficiale non si hanno particolari problemi di interconnessione.

Una precisazione: trasmettitore e ricevitore devono accordarsi sul modo di trasmettere prima di iniziare la trasmissione stessa, pena l'impossibilità di instaurare la trasmissione o ricevere bit che appaiono casuali. Questa operazione va fatta configurando opportunamente il software di comunicazione e/o modificando manualmente alcuni dip-switch o altri dispositivi hardware.

E' importante garantire il rigoroso rispetto della durata dei singoli bit: infatti non è presente alcun segnale di clock comune a trasmettitore e ricevitore e l'unico elemento di sincronizzazione è dato dal fronte si salita del bit di start. Come linea guida occorre considerare che il campionamento in ricezione è effettuato di norma al centro di ciascun bit: l'errore massimo ammesso è quindi, teoricamente, pari alla durata di mezzo bit (circa il 5% della frequenza di clock, considerando che anche il decimo bit deve essere correttamente sincronizzato). Naturalmente questo limite non tiene conto della difficoltà di riconoscere con precisione il fronte del bit di start (soprattutto su grandi distanze ed in ambiente rumoroso) e della presenza di interferenze intersimboliche tra bit adiacenti: per questo spesso si consiglia di usare un clock con una precisione migliore dell'1% imponendo, di fatto, l'uso di oscillatori a quarzo.

3.2.7) Il bit di parità

Oltre ai bit dei dati (in numero variabile tra 5 ed 9) viene inserito un bit di parità (opzionale) per verificare la correttezza del dato ricevuto. Esistono diversi tipi di parità:
-None: nessun tipo di parità, cioè nessun bit aggiunto;
-Pari (even): il numero di mark (incluso il bit di parità) è sempre pari;
-Dispari (odd): il numero di mark (incluso il bit di parità) è sempre dispari;

L'idea è quella di predeterminare la quantità di 1 (e di conseguenza di 0) da trasmettere, facendo in modo che il loro numero sia sempre pari (o dispari, a secondo della scelta che si vuole fare): così facendo, se durante la trasmissione dovesse accadere un errore su un singolo bit, il ricevitore sarebbe in grado di

rilevare l'errore, ma non di correggerlo. Si tratta ovviamente di un codice di rivelazione degli errori elementare e di conseguenza in disuso a favore di altri sistemi basati su codici a ridondanza ciclica (CRC) o altri algoritmi più complessi.

Il bit di parità a volte viene mantenuto sempre ad un livello prestabilito, per esempio in alcuni protocolli usati da macchine industriali. Ciò dà origine ad ulteriori due tipologie di parità, peraltro non molto comuni:
-Mark: il bit di parità vale sempre mark;
-Space: il bit di parità vale sempre space;
Tali configurazioni sono a volte usate per identificare la tipologia del byte trasmesso, per esempio potrebbe indicare se si tratta di un dato piuttosto che di un indirizzo.

3.2.8) I parametri elettrici RS-232

La tensione di uscita ad un trasmettitore RS232 deve essere compresa in valore assoluto tra 5V e 25V (quest'ultimo valore ridotto a 13V in alcune revisioni dello standard). A volte le tensioni in uscita sono intenzionalmente diminuite a +/- 6V anziché 12V per permettere minori emissioni EMC, peraltro sempre critiche, e favorire maggiori velocità di trasmissione. Il ricevitore deve funzionare correttamente con tensioni di ingresso comprese, sempre in modulo, tra i 3V ed i 25V. Molti ricevitori commerciali considerano semplicemente una tensione di soglia al valore di +2V (sopra viene riconosciuto un segnale alto, sotto uno basso) anche se ciò non è pienamente aderente alla norme. L'impedenza di uscita del trasmettitore deve in ogni situazione essere maggiore di 300 ohm; l'impedenza di ingresso deve essere compresa tra i 3 ed i 7 kohm, anche a dispositivo spento. La corrente prelevabile in uscita mantenendo i corretti valori logici deve essere di almeno di 1.6 mA (potrebbe però essere maggiore, anche di un ordine di grandezza) e nel caso di corto circuito deve essere minore di 100mA. Infine lo slew-rate (cioè la pendenza del grafico del segnale nel passare da 1 a 0 o viceversa) deve essere minore di 30V/us per evitare eccessive emissioni elettromagnetiche.

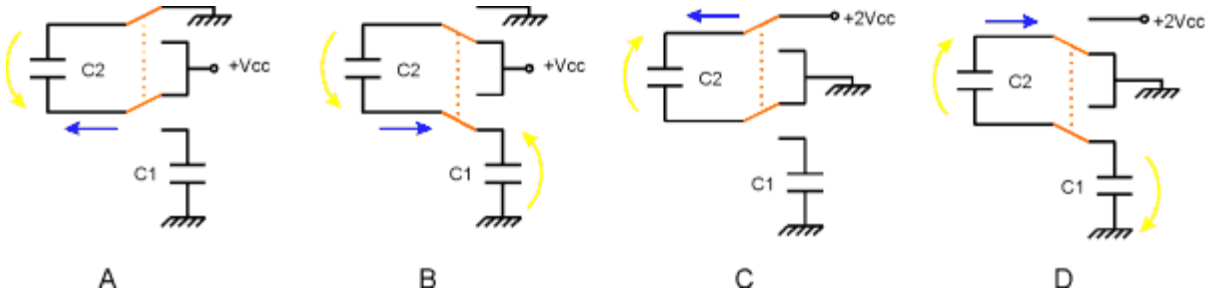
3.2.9) Come collegare una porta TTL o CMOS alla RS232

In genere i segnali utilizzati dai sistemi digitali variano tra 0 e 5V e non sono quindi direttamente compatibili con la standard RS232. In commercio esistono appositi traslatori di livello che hanno il compito di fornire sia in trasmissione che in ricezione gli opportuni livelli pur non modificando la forma del segnale trasmesso. Alcuni integrati (per esempio i classici 1488 e 1489, rispettivamente un trasmettitore ed un ricevitore, ambedue a quattro canali) sono molto usati in sistemi in cui è presente (oltre all'alimentazione logica di 5V o 3.3V) un'alimentazione duale a +/-12V. Questo integrato, come praticamente tutti i circuiti di questo tipo, contiene un inverter per ciascun canale e quindi nel segnale in uscita o in ingresso uno zero logico appare come 0 volt, cioè in quella che a molti sembra essere la rappresentazione ovvia dei segnali digitali. L'uso di questi integrati è semplice ma non è sempre attuabile a causa della necessità di disporre di tre alimentazioni: si pensi per esempio alle apparecchiature alimentate a batteria.
Il MAX232 (ed integrati simili, fatti da un po' tutti i produttori di semiconduttori) è un circuito integrato che permette il collegamento tra logica TTL o CMOS a 5V e le tensioni RS-232, partendo solo da un'alimentazione a 5V. Per ottenere la tensione positiva e negative necessarie per il funzionamento dell'integrato è usata una configurazione a pompa di carica, costituito da circuiti interni all'integrato e quattro condensatori esterni di circa 1 uF. La capacità effettiva dipende dal tipo di integrato e dalla relativa frequenza di commutazione; a volte i condensatori sono presenti all'interno dell'integrato stesso.
Sono disponibili anche integrati che richiedono un'alimentazione di solo 3.3V (p.e. il MAX3232).

La sezione ricevente del MAX232 è costituita da due porte invertenti che accettano in ingresso una tensione di +/- 12V (o altra tensione compatibile allo standard RS232) ed in uscita hanno un segnale TTL compatibile.
La sezione trasmittente ha due driver invertenti con in ingresso TTL compatibile e capaci di erogare a vuoto una tensione di poco meno di +/- 10V, compatibile con lo standard RS232.

3.2.10) Circuito a pompa di carica

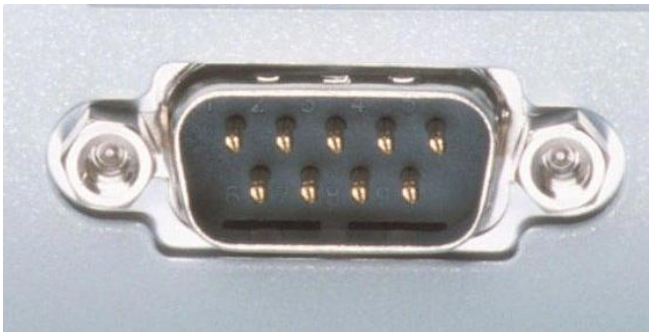
Per ricavare le tensioni positive e negative necessarie per garantire i livelli richiesti dalla RS232 è pratica comune utilizzare un duplicatore ed un invertitore di tensione a pompa di carica.



Le figure A e B mostrano come viene ottenuto il raddoppio della tensione. Una immagine che rende l'idea è quella di un contenitore (C2) che preleva acqua da una fonte e la riversa in un secondo contenitore (C1) posto a maggiore altezza. Più in dettaglio:
-Inizialmente il condensatore C2 viene connesso tra massa e Vcc; quindi la corrente (in blu) carica C2 alla tensione di alimentazione (in giallo). Quindi Vc2 = Vcc;
- C2 viene successivamente connesso tra Vcc ed un secondo condensatore C1; la tensione ai capi di C1 deve essere uguale alla somma di Vcc e Vc2 e quindi C2 si scarica verso C1, che aumenta la propria tensione rispetto a massa;
-Il processo è ripetuto fino a quando la tensione ai capi di C1 è uguale a 2Vcc: in questo caso infatti C2 non si può più scaricare.
Da notare che, nel funzionamento normale, il processo non può mai interrompersi in quanto il carico collegato a C1, non disegnato, assorbe corrente e quindi tende a scaricare C2 stesso.
Analogamente le figure C e D mostra l'inversione di tensione:
-Inizialmente C2 è caricato alla tensione di alimentazione (magari, come nel disegno da 2Vcc, ricavata con il precedente circuito);
-Quindi C2 è connesso tra massa e C1 avendo cura di invertire le polarità. In questo modo C1 si carica a -2Vcc;
Il limite dei circuiti a pompa di carica è la limitata quantità di corrente disponibile: infatti se prelevo corrente da C1 questo tende a scaricarsi, facendo scendere la tensione; la corrente generata da un circuito integrato tipo Max232 è generalmente tutta utilizzata per il solo funzionamento del driver e quindi non è disponibile per altri circuiti.

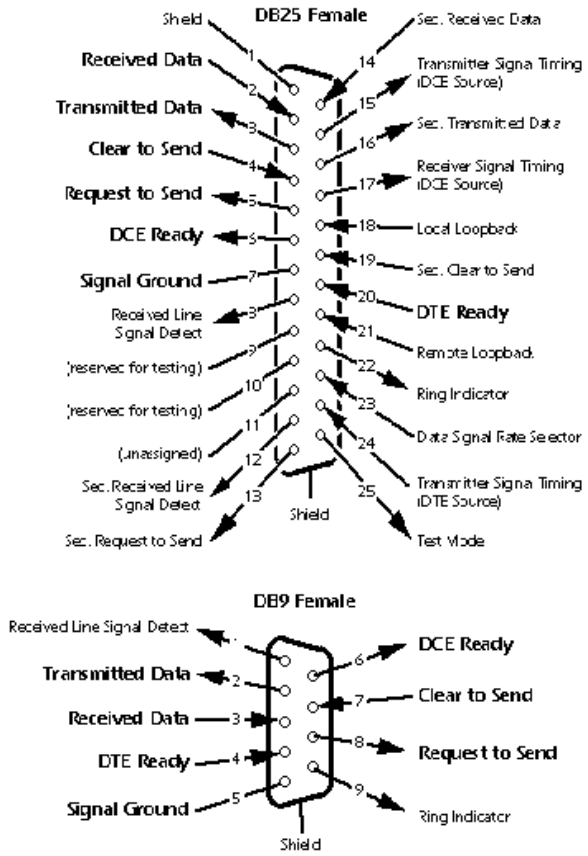
3.2.11) La piedinatura del connettore RS-232

Nei personal computer sono disponibili due tipi di connettori RS-232: DB9 (nove pin) e DB25 (25 pin, il connettore originale e presente solo sui PC più vecchi); ambedue i connettori sono maschi e praticamente identici dal punto di vista funzionale anche se non coincidente con quello proposto dallo standard ufficiale.



Di seguito la tabella con indicati i nomi dei segnali, il numero dei pin e la direzione del segnale;

Sigla	Circuito	DB-25	DB-9	Direzione	Nome	Descrizione
TD	C103	2	3	DTE→DCE	Transmitted Data	Dati seriali trasmessi
RD	C104	3	2	DCE→DTE	Received Data	Dati seriali ricevuti
RTS	C105	4	7	DTE→DCE	Request to Send	Richiesta trasmissione
CTS	C106	5	8	DCE→DTE	Clear to Send	Abilitazione trasmissione
DTR	C108	20	4	DTE→DCE	Data Terminal Ready	DTE pronto
DSR	C107	6	6	DCE→DTE	Data Set Ready	DCE pronto
RI	-	22	9	DCE→DTE	Ring Indicator	Obsoleto
DCD	C109	8	1	DCE→DTE	Data Carrier Detect	Rilevazione portante
TC	-	15	-	DTE→DCE	Trasmitting Clock	Clock per il sincronismo
GND	C102	7	5	-	Signal ground	Massa
-	-	1	-	-	Shield	Schermatura



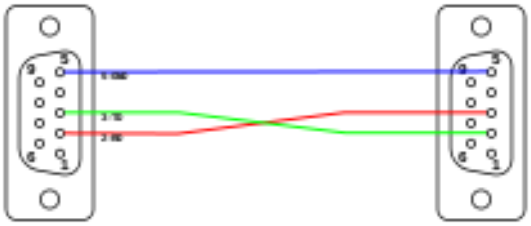
In teoria per ricevere e trasmettere un segnale RS-232 tra due dispositivi senza l'utilizzo di un modem bastano tre fili: ricezione, trasmissione e massa. Spesso lo è anche in pratica.
Gli altri fili (spesso opzionali, ma dipende dall'applicazione) servono per il cosiddetto handshake tra PC e periferica (o tra PC e PC) cioè per sincronizzare in hardware la comunicazione.

Sono presenti due coppie di fili:
-DTR/DSR: Quando il PC è collegato per la prima volta, pone alto DTR. La periferica risponde ponendo alto DSR.
-RTS/CTS: quando il PC inizia la trasmissione pone RTS alto, la periferica risponde quando pronta ponendo CTS alto. Per interrompere la trasmissione la periferica pone CTS basso.

3.2.12) Null modem

Il collegamento “null modem” è un metodo di connessione che permette a due dispositivi DTE di essere collegati direttamente usando un cavo seriale RS-232. Lo standard originale RS-232 prevedeva solo la connessione di un DTE con un DCE (modem). Con un cavo null modem vengono invertite le linee di trasmissione e ricezione di un canale di comunicazione seriale in modo da poter permettere a due dispositivi DTE di dialogare direttamente. Spesso però è necessario invertire fra loro anche i segnali di “handshake” per rendere più sicura la comunicazione.

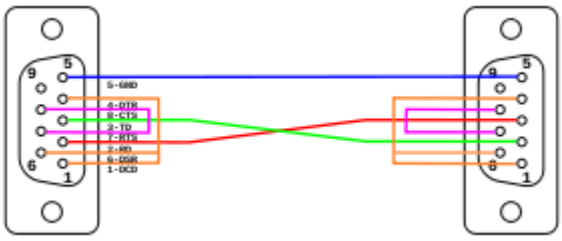
3.2.13) Null modem semplice asincrono



Lo schema più semplice per collegare fra loro due dispositivi DTE è quello di collegare la linea di trasmissione (TD) con quella di ricezione (RD) dati in modo da collegare la linea di trasmissione di un dispositivo DTE con la linea di ricezione dell'altro e vice-versa. Ovviamente la massa (GND, piedino 7 nella seriale DB-25 e 5 in quella DB-9) deve essere connessa direttamente.

Questo schema di collegamento null modem non utilizza nessuna linea di controllo del flusso hardware (CTS,RTS); il controllo di flusso deve essere garantito dal software.

3.2.14) Null modem a 3 fili con handshaking locale asincrono

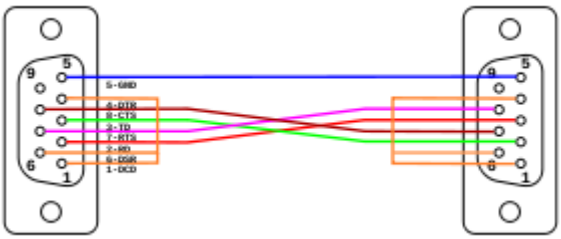


Un null modem senza sincronismi, come quello dello schema precedente, non sempre permette alla comunicazione di avviarsi se uno o entrambi i DTE controllano le linee di sincronismo RTS/CTS, DTR/DSR e DCD.

Ove è necessario utilizzare una comunicazione su soli 3 fili e i DTE richiedano un controllo di flusso hardware si può adottare uno schema che gestisca unhandshake locale. Un handshake locale si ottiene collegando su ciascun connettore, localmente, le linee RTS e CTS fra loro in modo che il DTE, attivando l'uscita RTS, veda il segnale CTS attivo. Allo stesso modo, collegando localmente le linee DTR, DSR e DCD, si ottiene una emulazione locale della gestione di questi segnali. In questo modo quando il DTE è pronto ad inviare i dati e attiva l'uscita DTR si vede attive le linee DSR e DCD e ritiene così che esista la portante (segnale DCD) e il DTE remoto sia operativo (DSR).

Questo tipo di collegamento null modem deve essere utilizzato solo con software in grado di gestire direttamente il controllo di flusso. Il collegamento locale delle linee di controllo di flusso può sì attivare la comunicazione fra due DTE, ma rende anche impossibile il controllo di flusso hardware, pur facendo credere ai DTE che questo sia presente.

3.2.15) Null modem a 5 fili con handshaking parziale asincrono



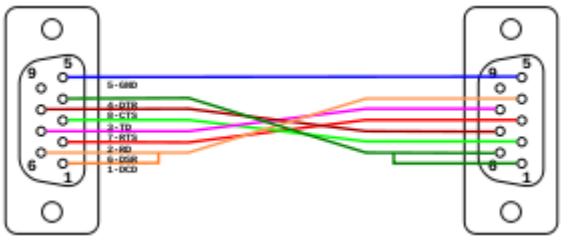
Un controllo di flusso parziale si ottiene su un cavo a 5 fili collegando la linea RTS di un DTE alla linea CTS dell'altro DTE e vice versa. In questo modo, quando il buffer di un DTE si riempie, abbassando la linea RTS si comunica all'altro DTE di sospendere la trasmissione fino a quando l'RTS non sarà nuovamente impostato su alto.

Le linee TD, RD e GND devono essere collegate come nello schema di null modem semplice.

Le linee DTR, DSR e DCD devono essere collegate come nello schema precedente.

Con questo schema di null modem il trasferimento dati è coperto dal controllo di flusso anche se un DTE non sa quando l'altro DTE è operativo.

3.2.16) Null modem a 7 fili con handshaking completo asincrono



Un controllo di flusso completo può essere ottenuto con un cavo a 7 fili in cui le linee TD, RD, GND, RTS e CTS sono collegate come nello schema di null modem a 5 fili mentre la linea DTR di un DTE è collegata alla linea DSR e alla linea DCD dell'altro DTE.

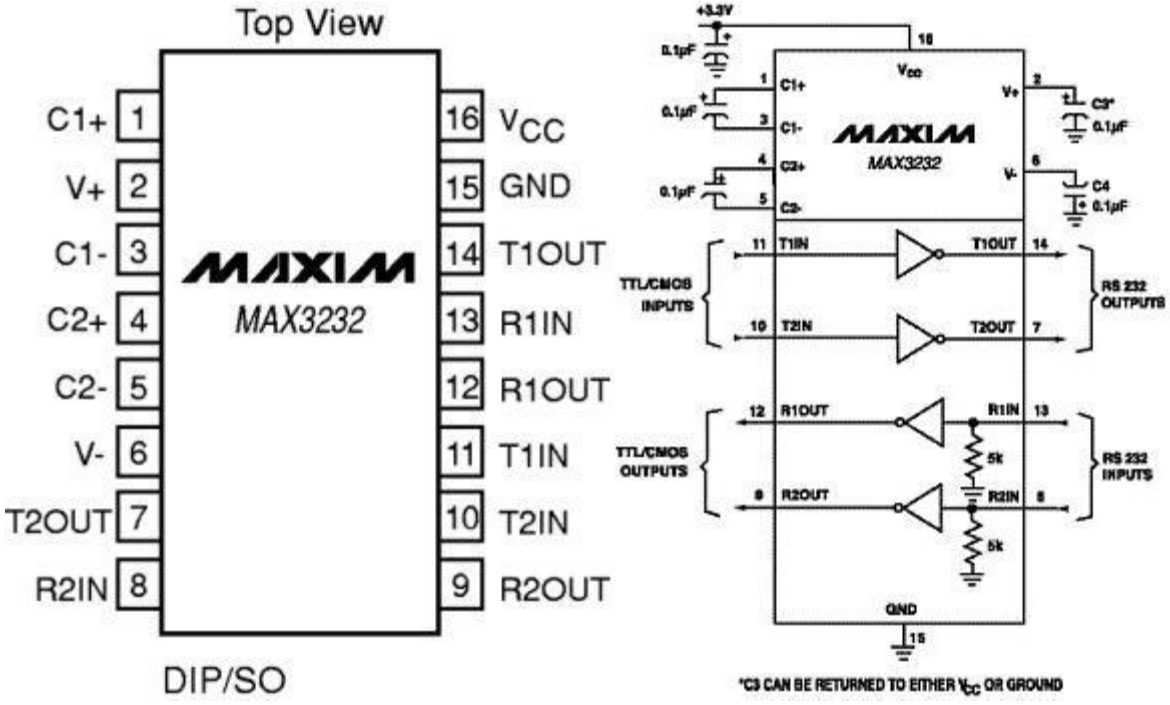
Il controllo di flusso durante la trasmissione avviene utilizzando le linee RTS e CTS come nello schema precedente, ma un DTE può in questo modo comunicare all'altro DTE quando ci sono dei dati da trasferire alzando il segnale DTR. Il segnale DTR su un lato deve essere collegato sia al DSR del DTE remoto sia al DCD dello stesso DTE remoto. Il collegamento DTR/DSR fa si che un DTE comunichi all'altro quando ha dei dati da trasmettere mentre il collegamento DTR/DCD emula il segnale di presenza della portante in modo che il DTE ritenga che il modem remoto (emulato dal null modem) sia in linea.

Con un handshaking completo si possono ottenere anche velocità di trasmissioni più alte.

La linea RI (Ring Indicator - piedino 22 sul connettore DB-25, piedino 9 su DB-9) non è normalmente utilizzata su un collegamento null modem.

3.2.17) La trasmissione seriale nel progetto

Nel progetto è stata utilizzata la trasmissione seriale utilizzando lo standard RS-232.
È stato realizzato un collegamento null modem semplice asincrono per permettere la comunicazione tra PC e microcontrollore. La comunicazione avviene in modalità half-duplex e l'handshake viene gestito dal software del PC che, ad ogni intervallo di tempo, invia un carattere per verificare il corretto funzionamento della trasmissione. La configurazione della trasmissione seriale utilizzata nel progetto è 9600 8n1.
Per la conversione dei segnali dallo standard RS-232 a livelli logici TTL e viceversa, è stato utilizzato il MAX232 della Maxim.



3.3) I motori elettrici

3.3.1) Introduzione ai motori elettrici

I motori elettrici si possono suddividere in varie categorie, che si differenziano a seconda della tensione di alimentazione e del tipo di pilotaggio: vi sono infatti i motori in continua (a spazzole e brushless), quelli in alternata (sincroni e asincroni), quelli detti "universali" (possono funzionare sia in continua che in alternata) e i motori passo-passo. Ciascun tipo di motore presenta caratteristiche proprie, risponde a ben determinate esigenze applicative e richiede tecniche di pilotaggio e circuiti di controllo specifici. Fra i vari tipi di motori elettrici, quelli di tipo "passo-passo" occupano un ruolo del tutto particolare, sia per il modo di pilotaggio che per il tipo di impiego.

Vengono infatti denominati "a passo" o "passo-passo" i motori che possono essere fatti avanzare a singoli passi e bloccati in una posizione qualunque, in modo da consentire facilmente posizionamenti di grande precisione, utili in una vasta serie di applicazioni quali ad esempio i servomeccanismi nell'automazione industriale, nella robotica, nelle stampanti, negli hard-disk, ecc. Per poter essere pilotati, i motori passo-passo richiedono però sequenze di impulsi particolari, che debbono essere generati da opportuni circuiti elettronici.

3.3.2) I Motori Passo-Passo

I motori passo-passo sono degli attuatori elettrici in corrente continua che compiono una rotazione comandata da una sequenza di impulsi elettrici. Come per tutti i motori elettrici, un motore passo passo è composto da una parte fissa (statore) e da una parte mobile (rotore), le quali differiscono per caratteristiche e dimensioni a seconda del tipo di motore a cui appartengono.

I motori passo-passo sono infatti raggruppabili in tre grandi famiglie, a seconda della loro struttura:

-Motori a riluttanza variabile;
-Motori ibridi;
-Motori a magnete permanente.

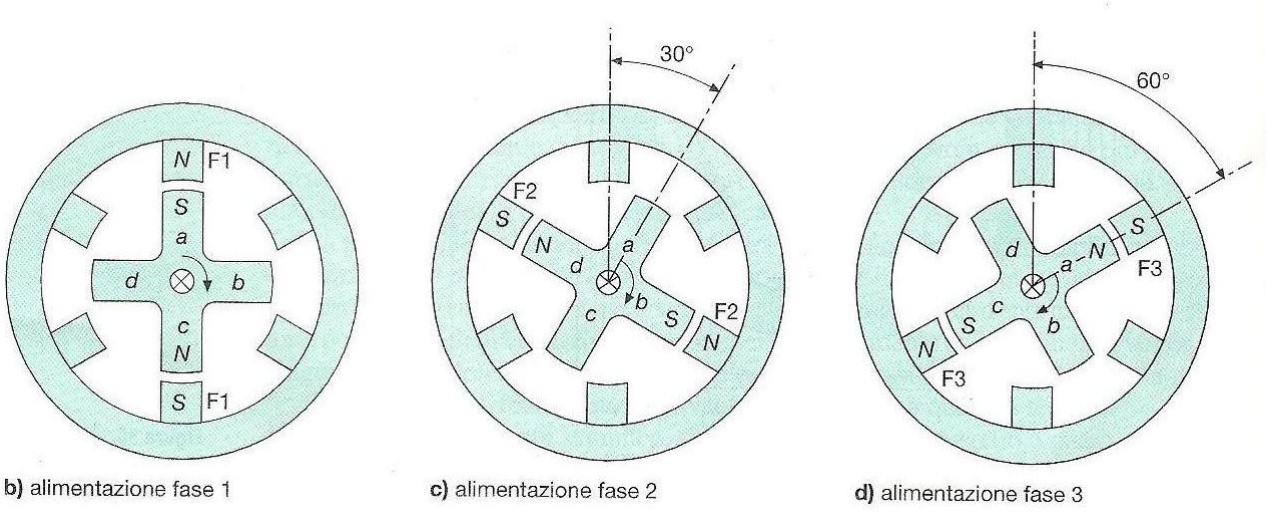
-Motori a riluttanza variabile

Come suggerisce il nome, il funzionamento di questi motori si basa sul principio della riluttanza magnetica, ossia l'opposizione di un materiale al transito di un flusso magnetico. Infatti durante il suo funzionamento il rotore tenderà sempre ad assumere una posizione rispetto allo statore tale da rendere minima la riluttanza del circuito magnetico in cui si sviluppa il flusso.

Questi motori hanno allo statore un certo numero di coppie polari; il rotore è costituito di materiale ad alta permeabilità con un certo numero di avvolgimenti o fasi, destinati a magnetizzarsi per induzione, mentre lo statore è costituito da un numero di fasi statoriche avvolte su poli diametralmente opposti, in numero maggiore di quelle presenti sul rotore. Alimentando in sequenza le fasi , come illustrato nella figura sottostante, la polarità ruota e il rotore segue lo spostamento, allineandosi con due suoi denti ai poli statorici sede del flusso; in questa situazione si rende infatti minima la riluttanza del tra ferro, per realizzare la qual situazione nasce la coppia motrice che fa spostare il rotore. Come si può notare nel passaggio dalla figura b alla figura c, alimentando due fasi adiacenti è possibile rilevare l'angolo di passo del motore semplicemente misurando lo spostamento angolare effettuato dal rotore, che risulta essere in questo caso di 30°, oppure calcolandone il valore attraverso la seguente relazione:

$$\theta_s = \frac{360}{mN_{dr}}$$

Dove m rappresenta il numero delle fasi statoriche, mentre Ndr rappresenta il numero dei denti rotorici. Nel caso in esempio, infatti, le fasi staoriche sono 3, mentre il numero di denti rotorici è pari a 4, e infatti l'angolo di passo risulta essere di 30°.

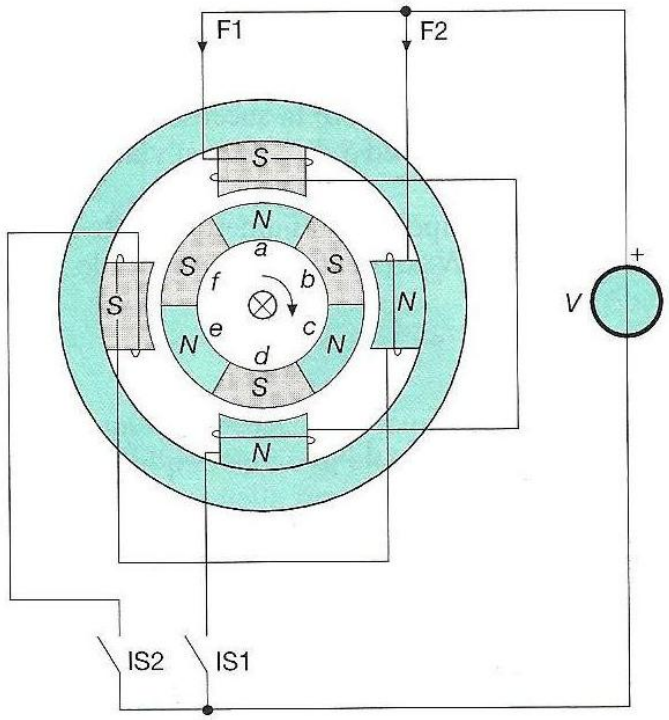


b) alimentazione fase 1 c) alimentazione fase 2 d) alimentazione fase 3

Il motore a riluttanza variabile ha il vantaggio di essere robusto e di semplice costruzione, ma presenta alcuni inconvenienti di funzionamento, tra cui una certa rumorosità, la tendenza a oscillare prima di raggiungere le posizioni di equilibrio, e soprattutto l'assenza della coppia di bloccaggio negli intervalli di tempo in cui non è alimentato, che permette al carico di ruotare liberamente, a causa dell'assenza di poli magnetici statorici e rotorici, che vengono a crearsi solo in conseguenza della magnetizzazione indotta dallo statore.

-Motori a magneti permanenti

In questo particolar tipo di motore, lo statore è simile a quello presente nei motori a riluttanza variabile, con polarità dipendenti dal verso delle correnti di eccitazione, mentre il rotore è costituito da un magnete permanente, con poli che occupano ognuno un settore rotorico, come in figura.



Facendo riferimento alla figura, eccitando la fase F1 si creano i poli statorici che esercitano delle forze di attrazione sul polo rotorico opposto e di repulsione su quello omonimo, per cui il rotore raggiunge la posizione di equilibrio rappresentata. Una volta alimentata la fase F2 i poli statorici precedenti si smagnetizzano e vengono a crearsi delle polarità N-S spostate di 90°, in quanto il settore rotorico b di polarità S viene attirato dal polo N del rotore, mentre il polo S opposto sullo statore attira il settore rotorico e, di polarità opposta anch'esso. In questo modo avviene una rotazione del rotore di 30°. Per realizzare un ulteriore rotazione di altri 30° del rotore sarà necessario eccitare nuovamente F1 e F2, ma invertendone la corrente, in modo da ottenere polarità opposte.
Il valore dell'angolo di passo di questo motore è ricavabile dalla relazione precedentemente utilizzata con il motore a riluttanza variabile.
Questo tipo di motore a passo è molto economico e ha il vantaggio di presentare un'ottima coppia di bloccaggio in mancanza di eccitazione, in quanto i poli rotorici, essendo permanenti, in assenza di corrente di eccitazione magnetizzano per induzione i poli statorici contrapposti, creando una coppia resistente che si oppone alle oscillazioni libere del carico. Gli svantaggi di questo motore sono relativi alle limitate prestazioni di coppia e velocità ed alla difficoltà nell'ottenere piccoli valori dell'angolo di passo.

-Motori Ibridi

Questi motori combinano i principi dei motori a riluttanza variabile e a magneti permanenti.
Sono formati da un rotore diviso assialmente in due settori dentati, montati su un magnete permanente cilindrico con polarità S-N alle due estremità. I denti rotorici dei due settori sono sfasati tra di loro di mezzo passo. Lo statore è simile a quello dei motori a riluttanza e presenta due o più fasi.

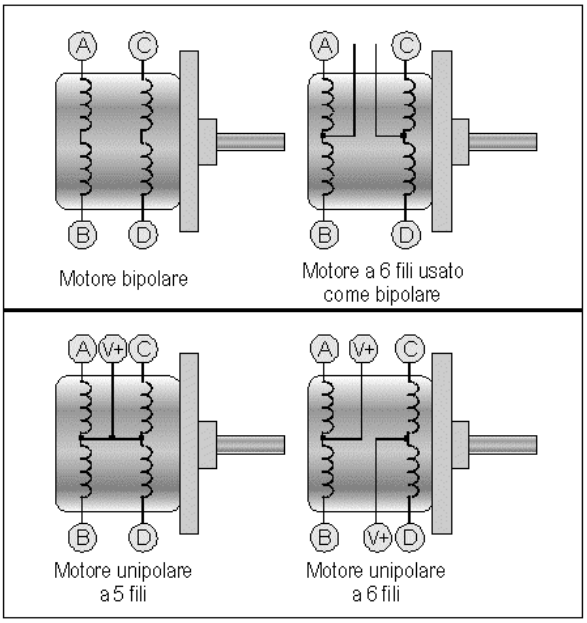
Il funzionamento avviene alimentando le due fasi in successione, con l'inversione della corrente della stessa fase ogni due scatti. Si ottiene cosi un movimento a passi dovuto all'azione combinata della coppia prodotta dall'eccitazione statorica, di quella di riluttanza e della coppia prodotta dal magnete rotorico, con un angolo di passo pari a:

$$\theta_s = \frac{90^\circ}{N_{dr}}$$

Dove Ndr è il numero di denti rotorici. Da questa relazione si evince come, essendo in questo tipo di motore il numero di denti rotorici molto più elevato rispetto ai casi precedenti, sia possibile ottenere un angolo di passo molto più ridotto. Oltretutto, la presenza del magnete permanente assicura una buona coppia di bloccaggio.
Si dividono in bipolari e unipolari.

3.3.3) Motori Bipolari e unipolari

Osservando la figura a lato, è possibile notare come le differenze tra motori bipolari e unipolari siano insite solamente nella connessione presente tra le varie fasi statoriche del motore.
Nei motori bipolari , a 4 o 6 fili, è possibile invertire la polarità della corrente, facendola fluire in entrambe le direzioni, mentre nei motori unipolari viene applicato un potenziale ad un determinato punto e la corrente fluisce solo in una determinata direzione. I vantaggi di quest'ultima soluzione sono una semplificazione del circuito di pilotaggio, il che permette di ridurne sensibilmente il costo.

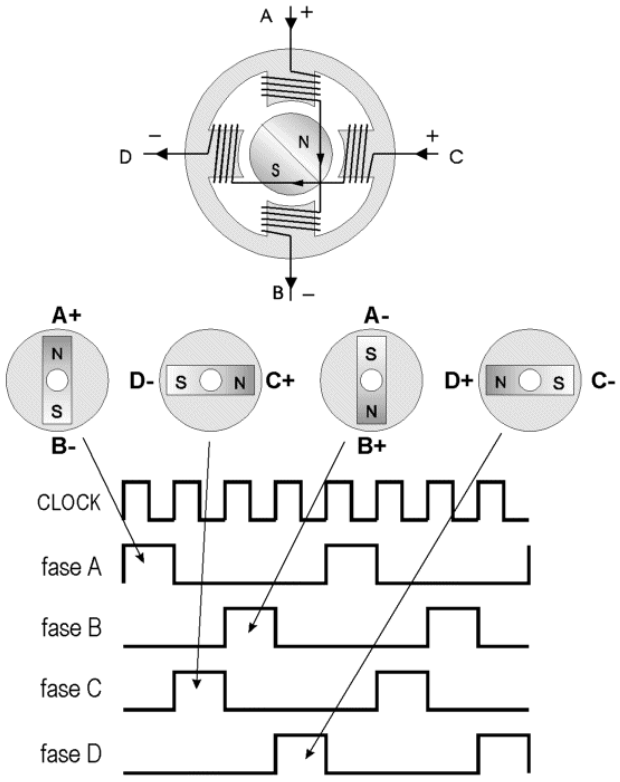


3.3.4) Modalità di pilotaggio dei motori Passo-Passo

Come già detto, i motori passo-passo vengono pilotati da opportuni circuiti digitali che realizzano una sequenza di impulsi tale da alimentare e de alimentare le varie fasi, in modo da muovere il motore. Questa sequenza di impulsi non è univoca, in quanto esistono vari metodi, ognuno coi propri pregi e difetti, di pilotare un motore passo-passo. Sostanzialmente le modalità di pilotaggio più diffuse ed efficaci sono tre:

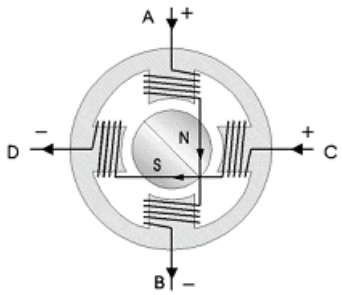
-Pilotaggio a Singola fase

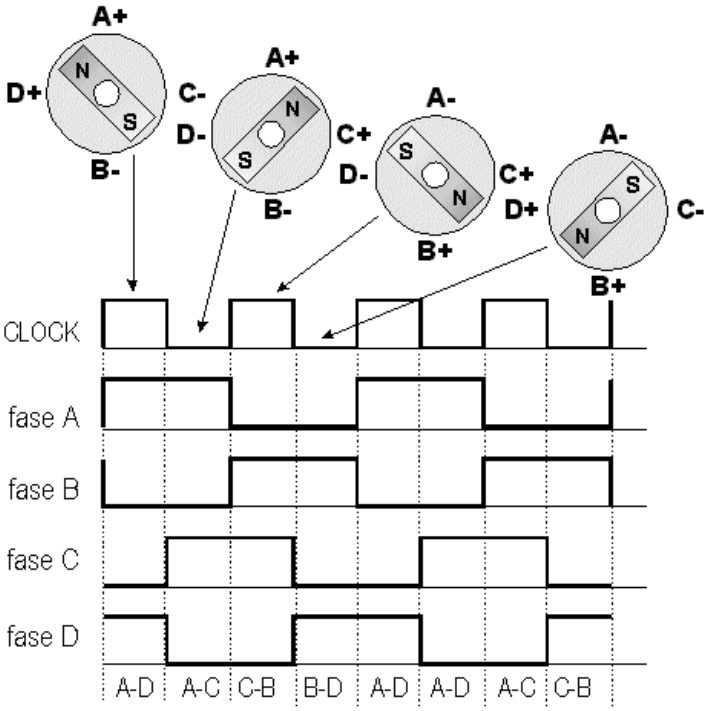
Preso in esame il motore illustrato in figura, si immagini di fornire tensione al conduttore A, collegando B a massa e lasciando scollegate le fasi C e D: a causa della magnetizzazione delle espansioni polari connesse alle fasi A e B il magnete permanente del rotore ruoterà, orientandosi in modo da allineare le proprie espansioni polari Nord e Sud nella direzione A-B, come mostrato in figura. Se successivamente si toglie tensione alla fase A e la si commuta alla fase C in modo da alimentare il percorso di corrente da C a D, il rotore ruoterà in senso orario di un quarto di giro allineandosi lungo la direzione C-D. Per provocare un ulteriore avanzamento si fornirà tensione alla fase B, poi alla D e così via, provocando una continua rotazione dell'asse del motore. La sequenza degli impulsi elettrici da fornire è quella evidenziata (corrispondente a due giri del motore), dove le varie sequenze sono sincronizzate da un clock. Come si può dedurre dal tipo di pilotaggio, le quattro fasi A, B, C e D non sono fra di loro equivalenti, bensì richiedono una sequenza di pilotaggio ben precisa. Per questo motivo, un erroneo collegamento anche di una sola delle fasi determina l'avanzamento irregolare o addirittura il blocco del motore.



-Pilotaggio a doppia fase

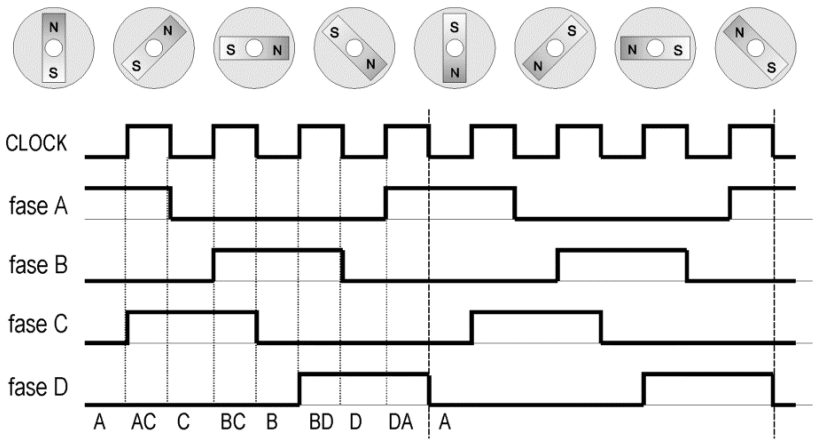
Un altro modo di pilotare il motore è quello di alimentare contemporaneamente due fasi, ad esempio A e C: in tal modo il rotore si allinea in una direzione intermedia fra le due espansioni polari dello statore. Lo scopo di questa doppia alimentazione è quello di ottenere una forza di attrazione magnetica più intensa fra statore e rotore, e quindi una maggior coppia meccanica di rotazione per il motore. Con questa tecnica, per fare avanzare il motore in senso orario è necessario fornire sequenze di impulsi del tipo A-C, poi C-B, in seguito B-D e infine D-A, in modo da provocare la completa rotazione dell'asse, ma sempre con il polo Nord del rotore allineato in una posizione intermedia fra due fasi. In figura è mostrata la sequenza degli impulsi richiesti con le rispettive posizioni assunte dal rotore.





-Pilotaggio a mezzo passo

Si può osservare che è possibile altresì alternare le sequenze delle due modalità descritte in precedenza: ad esempio si può alimentare prima A, poi la coppia A-C, poi C, poi la coppia B-C, poi B, e così via. Semplicemente, il rotore si allinea dapprima verso A, poi in posizione intermedia fra A e C, poi verso C, poi in posizione intermedia fra C e B, poi verso B, e così via. Lo scopo è quello di far compiere al rotore movimenti "a mezzo passo", e quindi attuare posizionamenti più precisi, anche se con una complicazione della sequenza degli impulsi ,come illustrato nella figura sottostante.



3.3.5) Vantaggi e difetti dei motori passo passo

Vantaggi:

-Quelli costruiti con tecnologia comune hanno un costo non elevato, relativamente ad altri tipi di motore con analoghe prestazioni.
-È possibile realizzare azionamenti di precisione controllati da computer in catena aperta, cioè senza utilizzare sensori di posizione o di velocità. Sono quindi utilizzabili con relativa semplicità e senza richiedere particolare potenza di calcolo.
-Hanno un'elevata robustezza meccanica ed elettrica: infatti non esistono contatti elettrici striscianti e, se necessario, possono essere realizzati anche in ambiente completamente stagno.
-È facile far compiere all'albero piccole rotazioni angolari arbitrarie in ambedue i versi e bloccarlo in una determinata posizione.
-La velocità di rotazione può essere molto bassa anche senza l'uso di riduttori meccanici.
-Hanno molto spesso momento d'inerzia piuttosto basso.

Difetti:

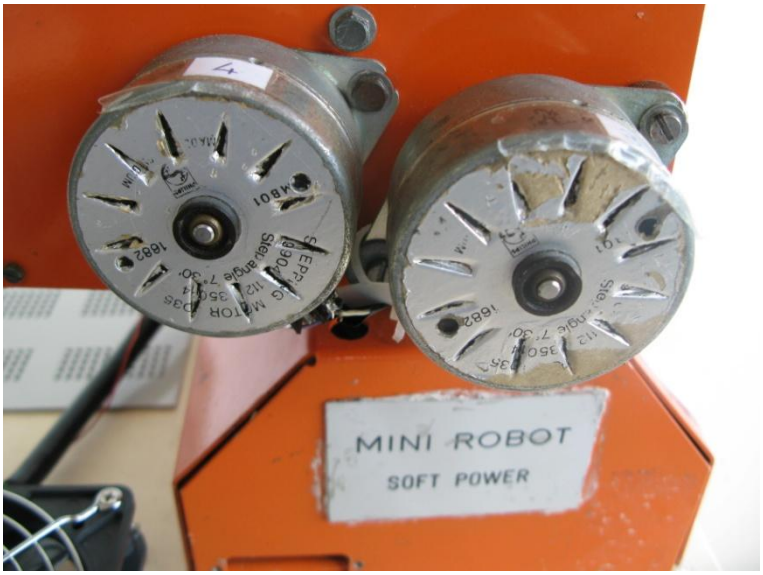
-Richiedono sempre circuiti elettronici per il pilotaggio, in genere di tipo digitale.
-Hanno un funzionamento a scatti e producono vibrazioni, soprattutto ai bassi regimi e se si adottano le tecniche di pilotaggio più semplici.
-Il loro rendimento energetico dipende dalla tecnologia costruttiva adottata, la potenza meccanica espressa come coppia e misurata in Nm (Newton per metro), a parità di assorbimento in corrente, dipende spesso dal tipo di pilotaggio elettrico/elettronico adottato.
-Permettono una velocità di rotazione massima intorno a 1000-1500 rpm. Esistono tuttavia motori che raggiungono i 4000-5000 rpm tramite sistemi di retroazione ad anello chiuso. La loro caratteristica di coppia tuttavia scende quasi esponenzialmente al crescere della velocità.
-Producono molto calore anche dopo pochi minuti.

3.3.6) Motori Utilizzati nel progetto Mini Robot

I motori passo-passo utilizzati per il progetto sono stati quelli originariamente montati sulla struttura meccanica. Essendo la suddetta struttura particolarmente datata (risale ai primi anni '80) anche i motori ovviamente lo sono. Infatti si tratta, come è possibile leggere dai dati di targa, di Stepping motor D35 modello MB01 prodotti dalla Philips, e dopo esaustive ricerche non sono state trovate informazioni utili per classificarlo adeguatamente, e l'unica informazione in nostro possesso è quella relativa al passo di ogni motore, ossia 7° 30'.
Quest'ultima informazione però, attraverso un semplice ragionamento, permette di ricavare la tipologia del motore. Infatti angoli di passo così ridotti sono realizzabili solo con motori ibridi, come è possibile verificare anche solo dalla relazione matematica che lega l'angolo di passo col numero di denti rotorici, riportata nelle pagine precedenti.
Per semplificare le modalità di pilotaggio i motori sono stati utilizzati in modalità unipolare utilizzando 5 dei 6 fili presenti su ogni motore (i due fili dell'alimentazione sono stai cortocircuitati).

Ovviamente ogni motore presente nel braccio meccanico, in base alla sua funzione, è stato utilizzato a determinate velocità, ma senza utilizzare una rampa di accelerazione, per semplificare ulteriormente il pilotaggio dei motori.
Per quanto concerne l'alimentazione, i motori sono stati alimentati a 12 V, valore desunto da tesine precedenti e relazioni. Successivamente è stata misurata la resistenza di ogni fase (50 ohm) ed è stato possibile ricavare la corrente che scorre in ogni avvolgimento, ossia 240mA.
Infine sono stati svolti vari test che hanno coinvolto tutti i motori presenti sul braccio meccanico, in modo da rilevarne le velocità e le modalità di pilotaggio ottimali.

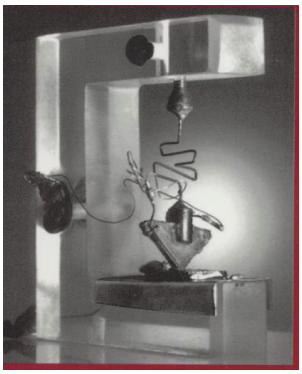
Numero motore	Movimento	Modalità di pilotaggio	Velocità
1	Apertura mano	doppia fase	100passi/s
2	Avambraccio	doppia fase	100passi/s
3	Braccio	doppia fase	62.5passi/s
4	Mano	doppia fase	100passi/s
5	Mano	doppia fase	100passi/s
6	Base	mezzo passo	125passi/s

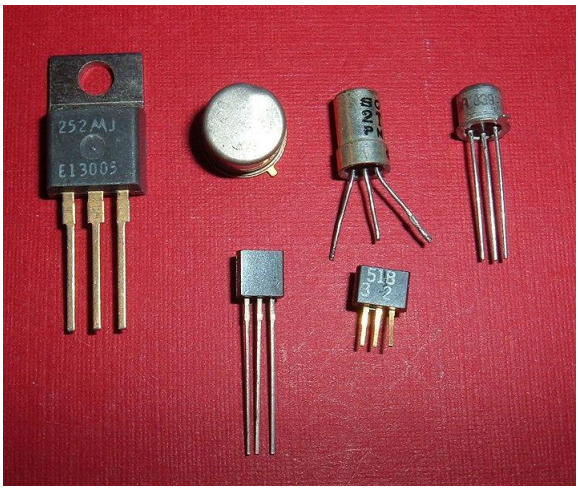


3.4) Transistor in configurazione Darlington

3.4.1) Introduzione: il transistor

Il transistor, detto anche transistore, abbreviazione del termine inglese transfer-resistor, è un dispositivo a semiconduttore largamente usato sia nell'elettronica analogica che nell'elettronica digitale.
Il transistor è costituito da un materiale semiconduttore al quale sono applicati tre terminali che lo collegano al circuito esterno. L'applicazione di una tensione elettrica o di una corrente a due terminali permette di regolare il flusso di corrente che attraversa il dispositivo, e questo permette di amplificare il segnale in ingresso.
Il funzionamento del transistor è basato sulla giunzione p-n, scoperta casualmente da Russell Ohl il 23 febbraio 1939.
Esistono principalmente due diverse tipologie di transistor, il transistor a giunzione bipolare ed il transistor ad effetto di campo, ed è possibile miniaturizzare i dispositivi di entrambe le categorie all'interno di circuiti integrati, il che li rende un componente fondamentale nell'ambito della microelettronica.
I transistor vengono impiegati principalmente come amplificatori di segnali elettrici o come interruttori elettronici comandati, ed hanno in larga parte sostituito i tubi termoionici.





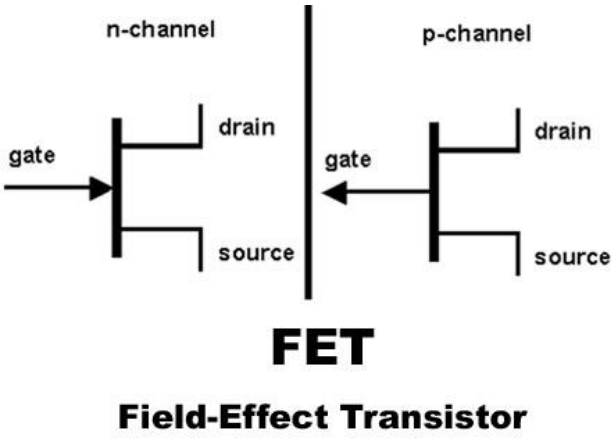
3.4.2) Tipologie

Le due principali tipologie di transistor sono il transistor a giunzione bipolare ed il transistor ad effetto di campo, descritti di seguito:

-Transistor ad effetto di campo (FET)

l transistor ad effetto di campo, anche chiamato con l'acronimo FET, è una tipologia di transistor largamente usata nel campo dell'elettronica digitale e diffusa, in maniera minore, anche nell'elettronica analogica.

Si tratta di un substrato di materiale semiconduttore drogato, solitamente il silicio, al quale sono applicati quattro terminali: gate (porta), source (sorgente), drain (pozzo) e bulk (substrato); quest'ultimo, se presente, è generalmente connesso al source. Il principio di funzionamento del transistor a effetto di campo si fonda sulla possibilità di controllare la conduttività elettrica del dispositivo, e quindi la corrente elettrica che lo attraversa, mediante la formazione di un campo elettrico al suo interno. Il processo di conduzione coinvolge solo i portatori di carica maggioritari, pertanto questo tipo di transistore è detto unipolare.
Insieme al transistor a giunzione bipolare, il FET è il transistor più diffuso in elettronica: a differenza del BJT esso presenta il vantaggio di avere il terminale gate di controllo isolato, nel quale non passa alcuna corrente; mentre ha lo svantaggio di non essere in grado di offrire molta corrente in uscita. In genere i circuiti con transistor FET hanno infatti una alta impedenza di uscita, erogando quindi correnti molto deboli.



-Transistor a giunzione bipolare (BJT)

l transistor a giunzione bipolare, anche chiamato con l'acronimo BJT, è una tipologia di transistor largamente usata nel campo dell'elettronica analogica, ed è usato principalmente come amplificatore ed interruttore.
E' composto da tre strati di materiale semiconduttore drogato, solitamente il silicio, in cui lo strato centrale ha drogaggio opposto agli altri due, in modo da formare una doppia giunzione p-n. Ad ogni strato è associato un terminale: quello centrale prende il nome di base, quelli esterni sono detti collettore ed emettitore. Il principio di funzionamento del BJT si fonda sulla possibilità di controllare la conduttività elettrica del dispositivo, e quindi la corrente elettrica che lo attraversa, mediante l'applicazione di una tensione tra i suoi terminali. Tale dispositivo coinvolge sia i portatori di carica maggioritari che quelli minoritari, e pertanto questo tipo di transistore è detto bipolare. Insieme

al transistor ad effetto di campo, il BJT è il transistor più diffuso in elettronica. Il dispositivo è in grado di offrire una maggiore corrente in uscita rispetto al FET, mentre ha lo svantaggio di non avere il terminale di controllo isolato.
Dal momento che il tipo di transistor utilizzato in questa sede è il BJT, ci soffermeremo ad analizzarli in modo più approfondito.

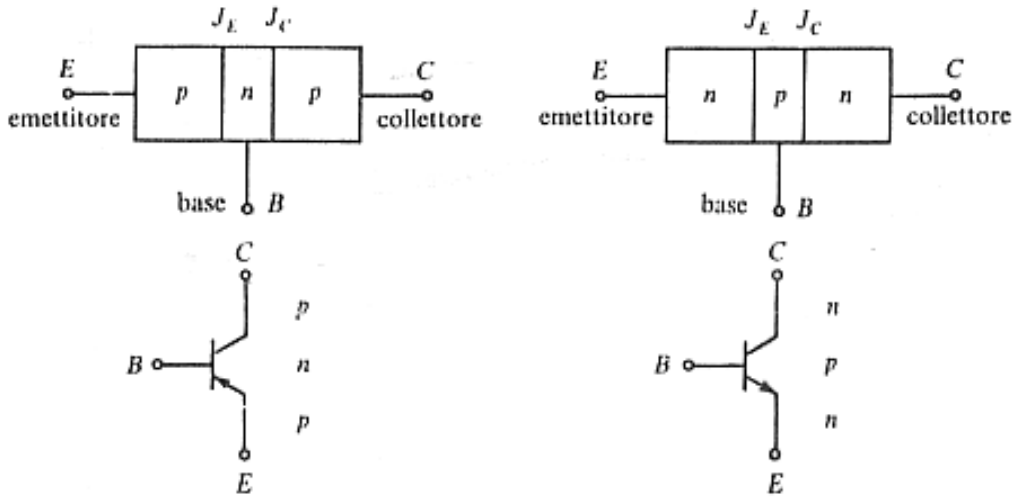
3.4.3) Struttura e principio di funzionamento del BJT

Tre regioni adiacenti di semiconduttore drogate alternativamente di tipo p e di tipo n costitui-scono, sotto certe condizioni, un transistor BJT. Questa struttura è rappresentata in forma schematica in fig. 1 nelle due possibili versioni: pnp e npn, con i simboli grafici corrispondenti. La parte centrale viene chiamata base e le due zone laterali emettitore e collettore. Il dispositivo presenta dunque due giunzioni, base-emettitore e base-collettore, che indicheremo in seguito rispettivamente con JE e JC . La simmetria dei modelli di fig. 1 è convenzionale; in realtà le giunzioni JE e JC hanno aree di-verse, come risulta pure diversa l'intensità del drogaggio dell'emettitore e del collettore. Ne consegue che i terminali E e C non sono intercambiabili e vengono espressamente indicati dal costruttore. Il ver-so della freccia nel simbolo è quello della corrente di JE nel caso in cui la giunzione sia polarizzata di-rettamente.

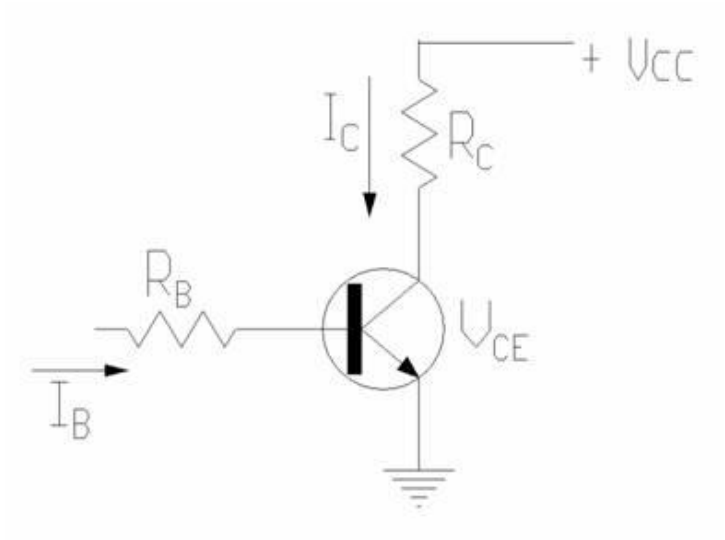
Due particolarità costruttive sono veramente essenziali per il funzionamento del BJT:

• la regione di base deve essere molto sottile (pochi µm);

• la stessa regione deve essere poco drogata rispetto a quella di emettitore.



Facendo riferimento allo schema generale riportato nella figura sottostante, è possibile individuare due equazioni, relative alla maglia di uscita ed alla maglia di ingresso:



$$Vcc = R_c \cdot I + Vce + R_e \cdot I_e$$

$$Vbb = R_b \cdot I_b + Vbe + R_e \cdot I_e$$

Inoltre è possibile ricavare il valore delle correnti di collettore, base ed emettitore, una volta noto il valore di hfe (amplificazione statica):

$$I_c = h_{FE} \cdot I_b$$

$$I_e = I_c + I_b$$

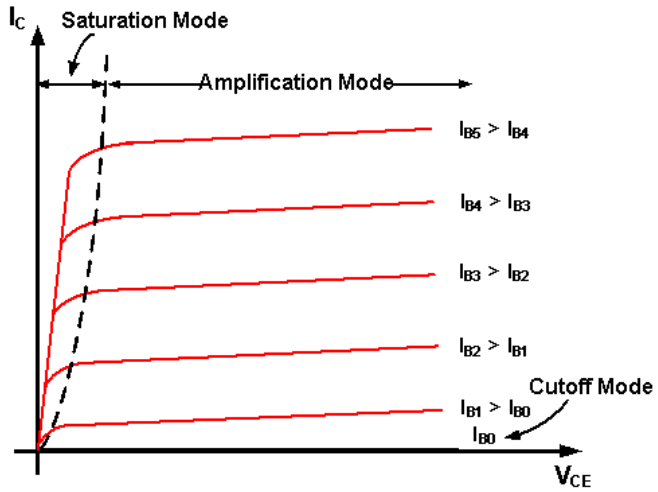
3.4.4) Zone di funzionamento del BJT

Esaminando le caratteristiche di uscita del BJT, nella figura sottostante, è possibile distinguere tre zone di funzionamento:

-Una zona attiva o lineare, che occupa la maggior parte del piano delle caratteristiche;

-Una zona di saturazione, caratterizzata da bassi valori di Vce;

-Una zona di interdizione, che coincide con l'asse delle ascisse, in cui sia Ib che Ic assumono valori trascurabili.



Il BJT viene utilizzato in zona lineare quando deve funzionare come amplificatore di segnale, mentre viene fatto lavorare in commutazione (switching mode) tra la zona di saturazione e quella di interdizione quando viene utilizzato come interruttore.

-Zona attiva

Il BJT viene utilizzato in zona attiva o lineare quando deve essere utilizzato come amplificatore di segnale, e la giunzione base-emettitore viene polarizzata direttamente, mentre la giunzione base-collettore è polarizzata inversamente. In questa configurazione le variazioni sinusoidali della corrente di ingresso Ib producono variazioni sinusoidali delle grandezze di uscita Ic e Vce.

-Zona di saturazione

In questa zona il BJT si trova in piena conduzione, e la sua tensione Vce è molto bassa, in quanto sia la giunzione base-emettitore che quella base-collettore sono polarizzate direttamente. In questo caso il transistor presenta sui terminali di uscita (collettore ed emettitore) una tensione molto bassa, sicchè può essere considerato in prima approssimazione come un interruttore chiuso.

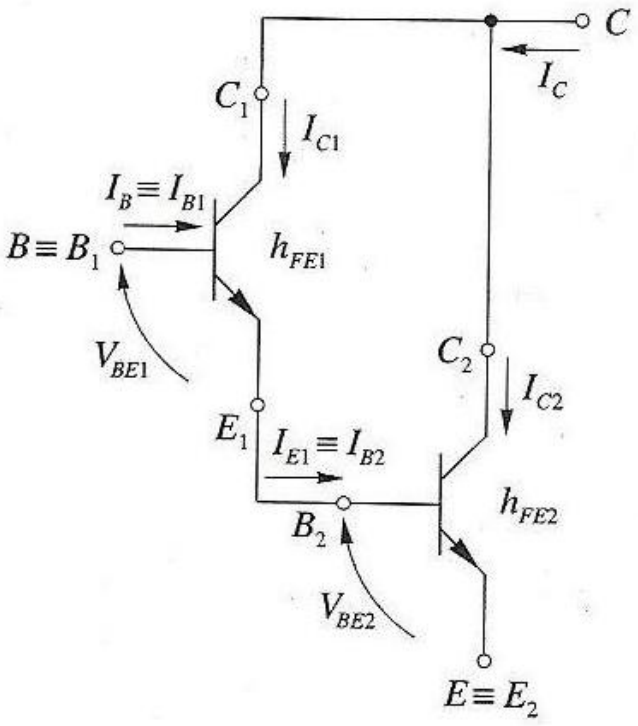
-Zona di interdizione

In questa zona le correnti nel BJT sono praticamente nulle, e le giunzioni base-emettitore e base-collettore sono polarizzate inversamente. Il transistor è quindi analogo ad un interruttore aperto, che non permette alla corrente applicata sul collettore di scorrere attraverso la base e verso l'emettitore.

3.4.5) Configurazione Darlington

Dal momento che i motori utilizzati nel progetto necessitano di correnti elevate per il loro corretto funzionamento, i transistor utilizzati sono di tipo BJT.
In particolare è stata utilizzata una struttura che permette di amplificare ulteriormente la corrente erogata, e ha come vantaggio secondario una maggiore impedenza di ingresso.

In questa particolare configurazione, detta Darlington, i collettori di due transistor BJT sono collegati tra di loro, e l'emettitore del primo transistor è connesso alla base del secondo.



Il guadagno di corrente hfe è pari a:

$$h_{FE} = \frac{I_C}{I_B} = \frac{I_{C1} + I_{C2}}{I_{B1}} = \frac{h_{FE1} \cdot I_{B1} + h_{FE2} \cdot I_{B2}}{I_{B1}} = \frac{h_{FE1} \cdot I_{B1} + h_{FE2}(h_{FE1} + 1) \cdot I_{B1}}{I_{B1}}$$

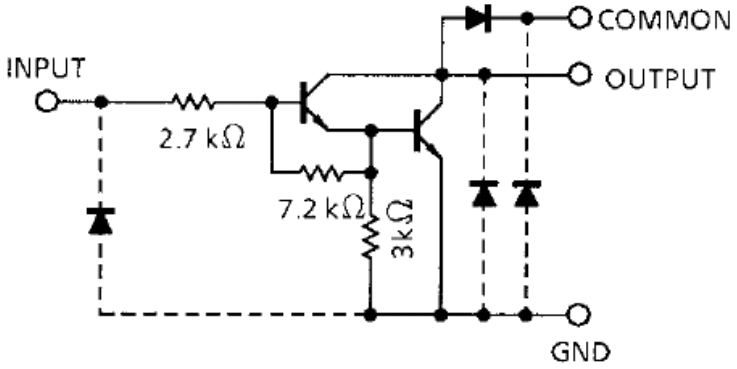
Ovvero:

$$h_{FE} = h_{FE1} + h_{FE2} + h_{FE1} \cdot h_{FE2} \cong h_{FE1} \cdot h_{FE2}$$

Risulta quindi evidente come il guadagno di corrente della configurazione corrisponda al prodotto dei guadagni dei singoli transistor. Questo la rende particolarmente adatta per applicazioni di potenza, sia di tipo lineare (stadi finali di amplificatori audio) sia di tipo switching, dove l'alto valore di Hfe permette di controllare carichi elevati con correnti di base relativamente modeste, che possono essere fornite anche da un semplice circuito integrato.

3.4.6) Il transistor utilizzato nel progetto del Mini Robot: ULN2803

Il circuito integrato utilizzato per l'azionamento in potenza dei motori utilizzati dal progetto è l'ULN2083 prodotto dalla Toshiba. Esso è composto da 8 drivers Darlington, che presentano i vantaggi esposti nel paragrafo precedente. In particolare la massima corrente erogabile da ogni driver è pari a 500mA, il che li rende ideali per il pilotaggio dei motori del braccio meccanico, visto che necessitano di circa 240mA per funzionare correttamente**.**



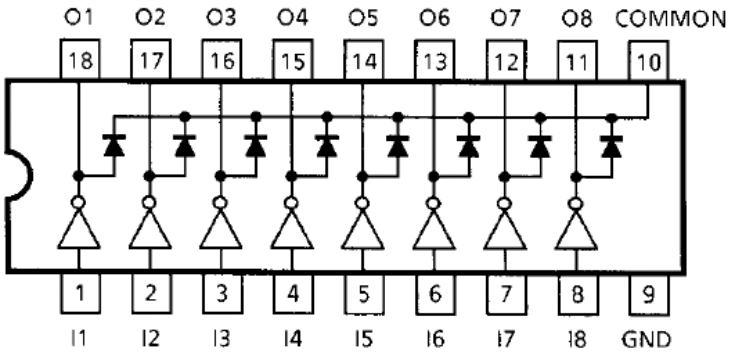
Oltre alla configurazione Darlington illustrata precedentemente, il circuito integrato presenta vari accorgimenti atti a renderlo adatto al pilotaggio in potenza.
In particolare la presenza di un diodo, chiamato diodo di libera circolazione, tra il morsetto di uscita e il comune (dove viene applicata l'alimentazione) risulta necessario per la protezione dei transistor quando essi pilotano dei carichi induttivi (come i motori). Il problema si presenta quando il transistor (in questo caso darlington) passa dallo stato di saturazione a quello di interdizione: l'induttore, avendo capacità inerziale, farebbe circolare corrente sul transistor danneggiandolo. Inserendo nel circuito il diodo di libera circolazione la corrente scorre nel diodo e in seguito di nuovo sull'induttore, fino all'esaurimento per dissipazione.

Gli ulteriori due diodi presenti nello stadio di uscita servono per ovviare alle giunzioni parassite che vengono a crearsi utilizzando due transistor in configurazione Darlington.

Infine, come è possibile apprezzare dal disegno, è presente una moderata resistenza di carico con cui è possibile pilotare i transistor con ingressi TTL , oltre a due resistenze tra base e emettitore e tra emettitore e massa che hanno lo scopo di stabilizzare i guadagni dei due transistor, con il piccolo inconveniente di diminuirli leggermente.

Di seguito è riportata la piedinatura dell'ULN2803:

PIN CONNECTION (TOP VIEW)



3.4.7) Caratteristiche elettriche ULN2803

MAXIMUM RATINGS (Ta = 25°C)

CHARACTERISTIC		SYMBOL	RATING	UNIT
Output Sustaining Voltage		$V_{CE\ (SUS)}$	−0.5~50	V
Output Current		I_{OUT}	500	mA / ch
Input Voltage		V_{IN}	−0.5~30	V
Clamp Diode Reverse Voltage		V_R	50	V
Clamp Diode Forward Current		I_F	500	mA
Power Dissipation	AP	P_D	1.47	W
	AFW		0.92 / 1.31 (Note)	
Operating Temperature		T_{opr}	−40~85	°C
Storage Temperature		T_{stg}	−55~150	°C

Parte Quarta

Specifiche di progetto

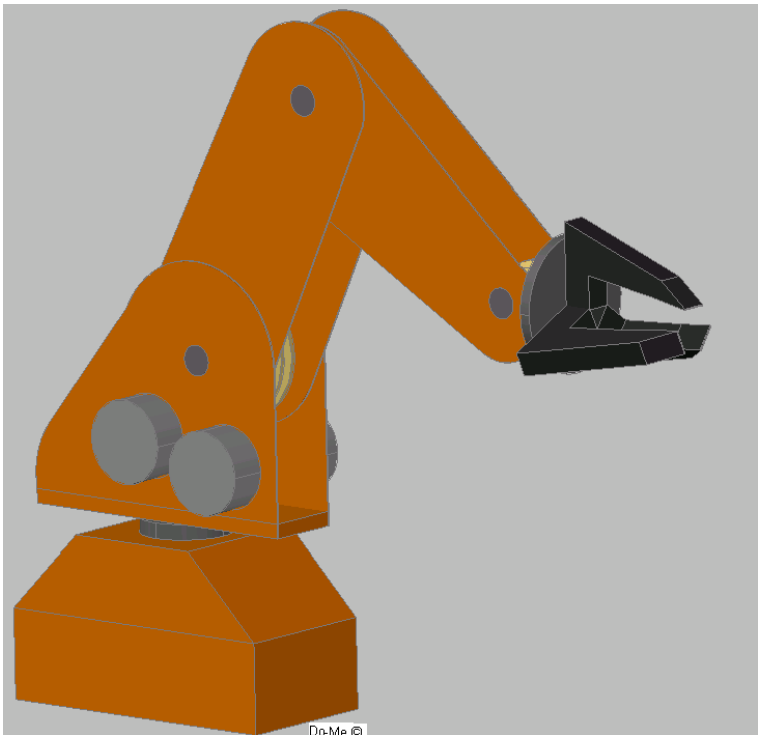
4.1) La struttura meccanica

4.1.1) Introduzione

La struttura meccanica intorno alla quale è stato sviluppato il progetto è molto datata. Infatti il robot risale all'inizio degli anni '80 e da allora non ha subito interventi di riparazione e di manutenzione degni di nota. Ciò è stato anche possibile dedurlo dallo stato delle cordicelle su cui si basa il movimento dell'intera struttura meccanica. Infatti esse si trovavano in una condizione fatiscente, in quanto erano sfilacciate e tendevano a spezzarsi se sottoposte ad uno sforzo eccessivo. Il primo intervento effettuato è stato quindi quello di andare a sostituirle con fili in nylon utilizzati normalmente in ambito modellistico, in grado di sostenere forze anche molto elevate (40 Kg). In seguito si è passati all'analisi dei sistemi di trasmissione del moto del robot, formati da carrucole e ingranaggi, in modo da ottimizzare, per quanto possibile, i vari movimenti eseguibili dal braccio meccanico.

4.1.2) Descrizione generale

Il braccio meccanico si presenta come una riproduzione sommaria del braccio umano. Infatti, come è possibile osservare nella figura sottostante, è composto da varie articolazioni che gli permettono, se opportunamente comandato, di realizzare movimenti e azioni particolarmente complesse.
Analizzandolo è possibile individuare 4 componenti principali dell'articolazione:
- La base, su cui poggia il braccio e che ne permette la rotazione intorno al proprio asse;
- La parte iniziale del braccio, che permette di alzare ed abbassare la mano, oltre a permetterne l'estensione;
- L'avambraccio, che permette di eseguire le stesse operazioni del braccio;
- la mano, che oltre a potersi aprire e chiudere può ruotare sul suo asse o alzarsi ed abbassarsi.

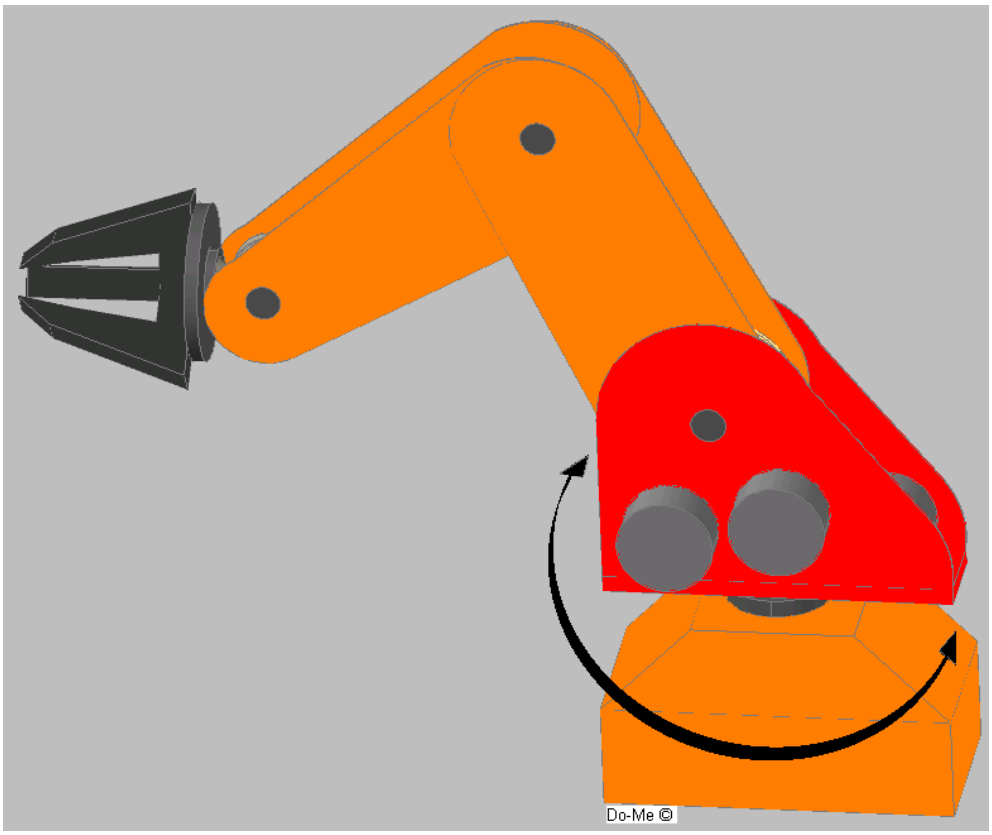


Numero motore	Movimento
1	Apertura mano
2	Avambraccio
3	Braccio
4	Polso
5	Polso
6	Base

I motori che permettono di muovere le varie parti del robot sono solidali alla struttura meccanica, e più precisamente sono posti sopra alla base. Per permettere la trasmissione del moto dei suddetti motori (che sono di tipo passo-passo) è presente un sistema di funicelle, ingranaggi e carrucole, che collega l'albero di ogni motore alla corrispondente articolazione.
In seguito verranno analizzate le varie componenti del braccio, con particolare riferimento ai vari movimenti eseguibili e al percorso realizzato dalle funicelle.

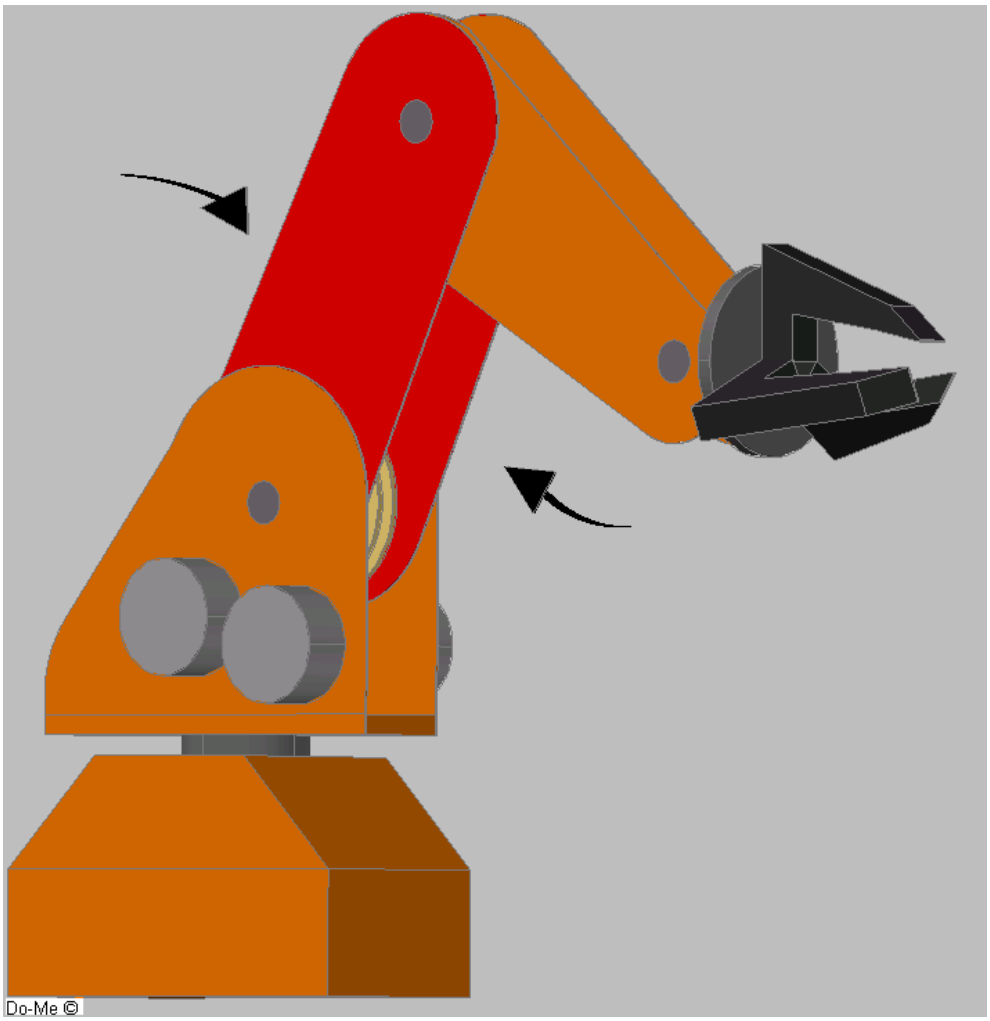
-La Base

La base, insieme alla parte iniziale del braccio, è l'unica componente del robot che non possiede un sistema di carrucole e funicelle atte a trasmettere il moto prodotto dai motori. Infatti il motore che aziona la base (il numero 6) è nascosto alla vista, in quanto è posto direttamente all'interno del supporto su cui poggia il robot, ed è orientato con l'albero verso l'alto, sporgendo all'esterno della struttura. L'albero è infine collegato alla struttura circolare mobile della base tramite una cinghia dentata.
Azionando il motore numero 6 è possibile far compiere alla base i movimenti illustrati nella figura sottostante, che consistono quindi in una rotazione di 360° intorno al proprio asse. Infine, è presente un fermo meccanico che, una volta raggiunti i 360° di rotazione, blocca il movimento della base.



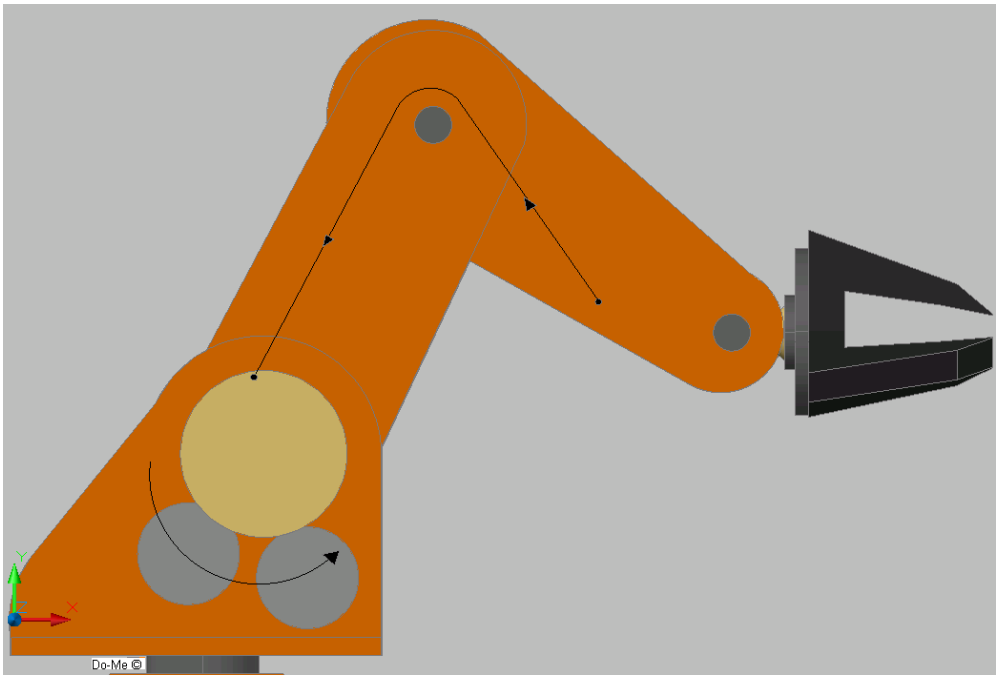
-Il Braccio

Il braccio, come già precedentemente ribadito, non presenta cordicelle o carrucole atte a trasmettere il moto del motore ad esso preposto, in quanto il suddetto motore (il numero 3) si trova nelle immediate vicinanze della parte iniziale del braccio. L'albero del motore è quindi collegato, tramite una coppia di ingranaggi, direttamente al braccio del robot.
Azionando il motore passo-passo numero 3 è possibile compiere i movimenti raffigurati nell'immagine sottostante, che consistono quindi in un estensione o ritrazione del braccio e con essa delle altre componenti dell'articolazione, vale a dire l'avambraccio e la mano.



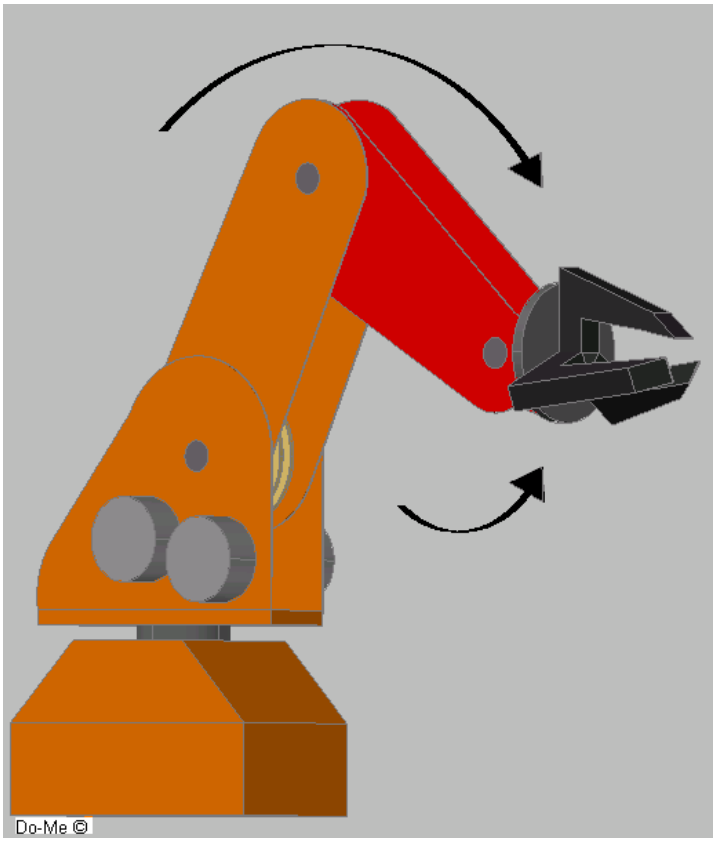
-L’ Avambraccio

L’avambraccio rappresenta uno dei movimenti più difettosi e imprecisi dell’intera struttura meccanica, e ciò è dovuto unicamente alla modalità con cui viene trasmesso il moto dal motore (il numero 2) all’avambraccio vero e proprio.
Infatti, come è possibile verificare dall’immagine sottostante che rappresenta il percorso effettuato dalla funicella collegata alla carrucola solidale all’albero del motore numero 2, sono presenti 2 punti di fissaggio, uno sulla carrucola presente sull’albero del motore ed uno sull’avambraccio.



La rotazione dell’albero del motore comporta un attrito molto elevato che agisce sulla carrucola presente sul “gomito” del robot e che rende quindi il movimento discontinuo con alcuni strappi. Da un punto di vista pratico questa discontinuità comporta vari problemi, in quanto far avanzare il motore di un determinato numero di passi può non coincidere con un movimento vero e proprio, in quanto solo al superamento di una certa soglia l’avambraccio si muoverà nella direzione voluta. Non è stato possibile risolvere in alcun modo questa problematica, in quanto per rendere il movimento fluido e proporzionale al numero di passi effettuati dal motore sarebbe stato necessario posizionare il motore proprio sul “ gomito” del robot, oppure utilizzare una molla particolare da posizionare sempre nel gomito, analoga a quella utilizzata in ambiti sportivi e di fitness (figura sottostante), in grado di produrre una forza resistente che si opponesse al movimento, e che quindi lo avrebbe reso decisamente più fluido.

Queste ultime soluzioni, ovvero lo spostamento del motore e la molla, non sono state implementate rispettivamente per motivi di complessità e di tempo, in quanto spostare il motore e posizionarlo sul gomito avrebbe comportato modifiche strutturali al robot e soprattutto un maggior carico per il motore numero 3 (dedicato allo spostamento del braccio) , mentre la molla avrebbe dovuto essere costruita appositamente per il braccio, in quanto in commercio i modelli esistenti sono caratterizzati da un eccessiva forza resistente e quindi avrebbero letteralmente bloccato il motore dedicato al movimento dell'avambraccio.
Consapevoli di queste limitazioni, l'unica strada percorribile è stata quella di utilizzare il meno possibile questo movimento, che comunque per determinati intervalli risulta avere una precisione accettabile.

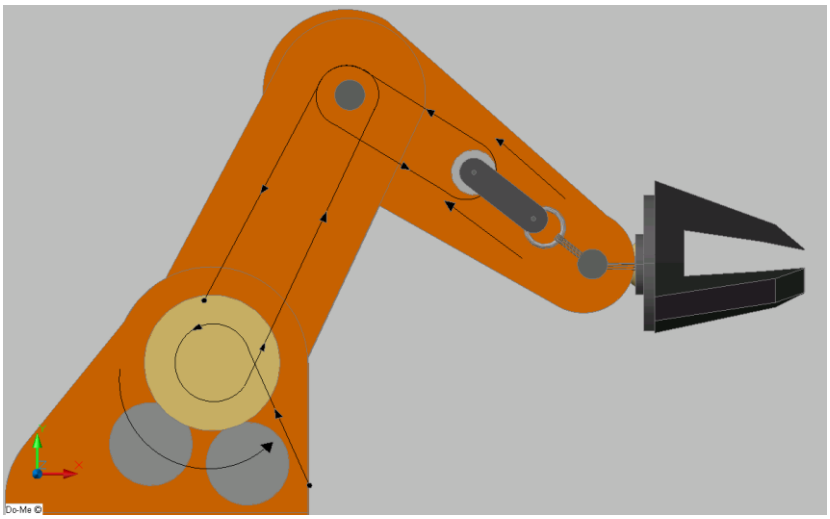


-La Mano

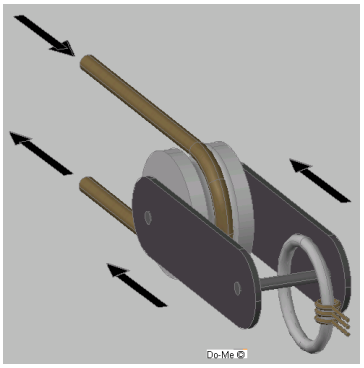
I vari movimenti realizzabili dalla mano sono comandati da ben 3 motori e più precisamente:
- Il motore numero 1 agisce sull'apertura e sulla chiusura delle 3 "dita" della mano;
- I motori 4 e 5 agiscono invece sulla rotazione della piastra circolare su cui poggia la mano e, in base al verso con cui vengono azionati, comportano una rotazione o un movimento verso l'alto o verso il basso della mano.

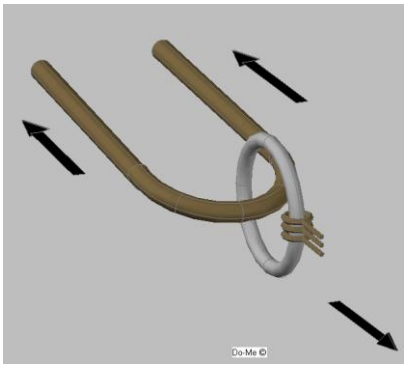
-Apertura e chiusura della mano

L'apertura e la chiusura delle dita che compongono la mano del robot è comandata dal motore numero 1. Come si può osservare nell'immagine sottostante, si tratta di un sistema di funicelle e carrucole piuttosto complesso, che presenta due punti di fissaggio: uno sulla carrucola inferiore, solidale all'albero motore, ed uno sulla struttura in metallo del robot. E' presente inoltre una carrucola nel "gomito" del braccio, ma l'elemento più importante di questo movimento è rappresentato dalla piccola struttura in metallo che si trova tra l'avambraccio e la mano.



Infatti questo piccolo particolare permette alla mano, nel momento in cui viene mosso il braccio e/o l'avambraccio, di mantenere la stessa pressione e lo stesso grado di chiusura, indipendentemente dalla posizione delle altre due componenti del robot. Questo accorgimento è stato adottato in seguito appunto alla problematica appena esposta, che comportava, una volta afferrato un oggetto, la sua caduta in seguito ai movimenti del braccio.
Nelle due figure sottostanti sono messi a confronto due sistemi, rispettivamente quello utilizzato attualmente e quello presente in origine sul robot.

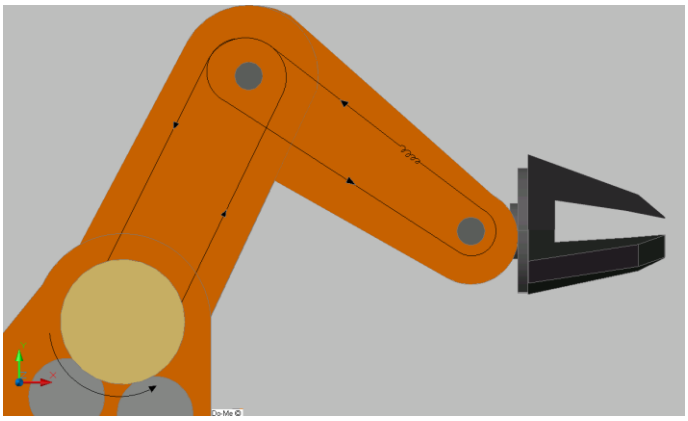


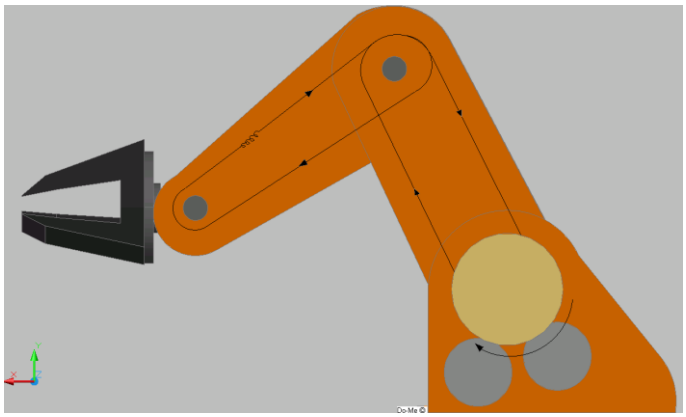


La prima struttura si rende necessaria in quanto, a seconda dei movimenti di braccio ed avambraccio, la corda proveniente dal motore 1 si smolla o si tende. Se nella prima struttura la corda è libera di scorrere nella carrucola e quindi non influenza lo stato della mano, nel secondo tipo di struttura ciò non avviene, in quanto la corda non scorre a causa dell'elevato attrito. Ciò comporta quindi una variazione dello stato di chiusura della mano e nel caso siano presenti oggetti nella stessa una conseguente perdita della presa. La struttura utilizzata attualmente permette invece di ovviare a questo problema. Probabilmente la soluzione che si è trovata presente sul robot è a sua volta una riparazione effettuata negli anni scorsi, forse senza il necessario approfondimento teorico.

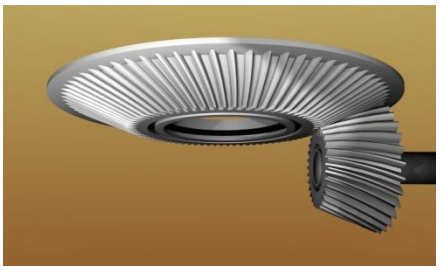
-Altri movimenti della mano

I movimenti verso l'alto e verso il basso ed i movimenti rotatori del polso sono comandati dai motori numero 4 e 5. Entrambi agiscono sulle rispettive carrucole, solidali ai propri alberi motore, in modo da far ruotare altre due carrucole provviste di ingranaggi posizionate su di un perno nella parte finale dell'avambraccio.

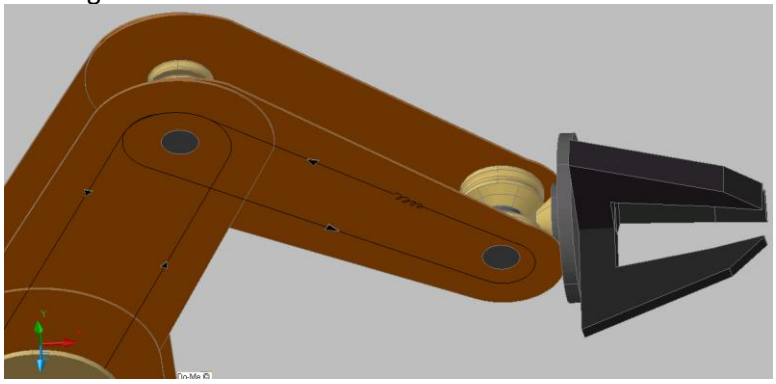




Come si può osservare dalle immagini, in questo caso è presente una molla nei due segmenti che vanno dal gomito alla mano, allo scopo di mantenere in tensione le due funicelle. Le corde in questo caso non sono fissate in un punto esterno alle carrucole, come nei casi precedenti, ma sono altresì fissate in un foro passante nella carrucola presente sull'albero di ogni motore e in quelle presenti nella parte finale dell'avambraccio. In questo modo le funicelle si avvolgono sulle due carrucole presenti ai due estremi dell'articolazione in modo opposto: ossia quando su una carrucola la fune si avvolge, sull'altra si svolge e viceversa.
Per meglio capire come avviene la trasmissione del moto tra la carrucola presente nella terminazione dell'avambraccio e la base della mano, fare riferimento alla figura sottostante.



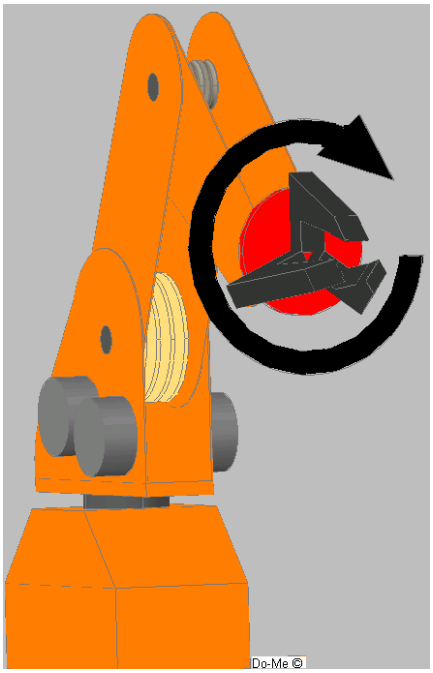
La struttura realizzata è chiamata pignone-corona (dove per corona si intende l'ingranaggio posto orizzontalmente, mentre per pignone quello posto verticalmente). Rispetto alla figura analizzata, nel robot sono presenti due corone(una comandata dal motore 4 ed una comandata dal motore 5), poste parallelamente l'una rispetto all'altra, posizionate ai due estremi del diametro esterno del pignone, come nella figura sottostante.

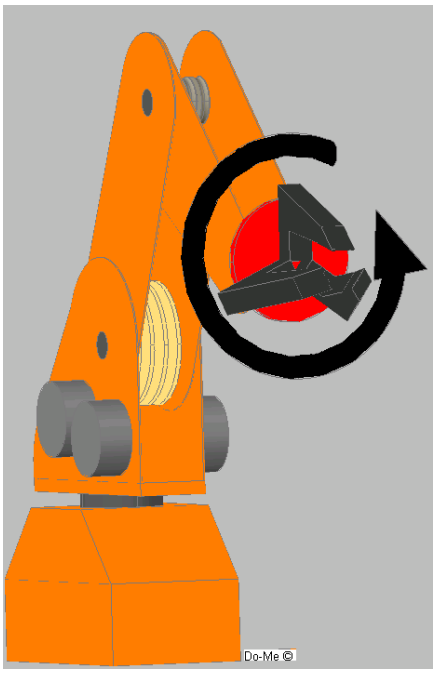


La struttura pignone-corona ha quindi il compito di traslare il movimento rotatorio assunto da una o entrambe le corone su un asse diverso da quello su cui viene originariamente esercitato il movimento stesso.
Inoltre, nel caso in cui le due corone si muovano nello stesso verso, il pignone si trova bloccato e impossibilitato a girare, quindi la forza esercitata dai denti delle due corone sui denti del pignone provoca uno spostamento verticale della mano, consentendole quindi di inclinarsi verso l'alto o verso il basso a seconda del verso di rotazione delle carrucole su cui sono poste le corone.

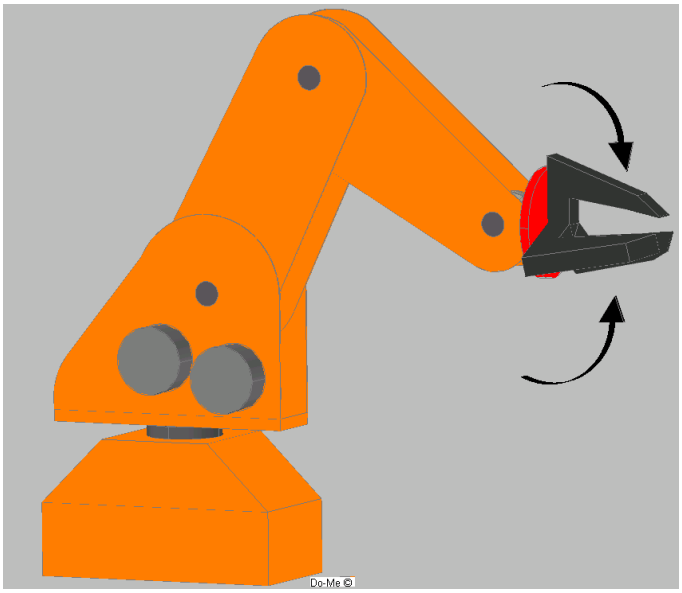
Ricapitolando, a seconda del verso di rotazione delle due corone, e quindi dell'azionamento dei motori 4 e 5, è possibile far eseguire alla mano diversi movimenti:

- Azionando entrambi i motori con verso opposto l'uno all'altro si realizza una rotazione in senso orario o antiorario della mano;





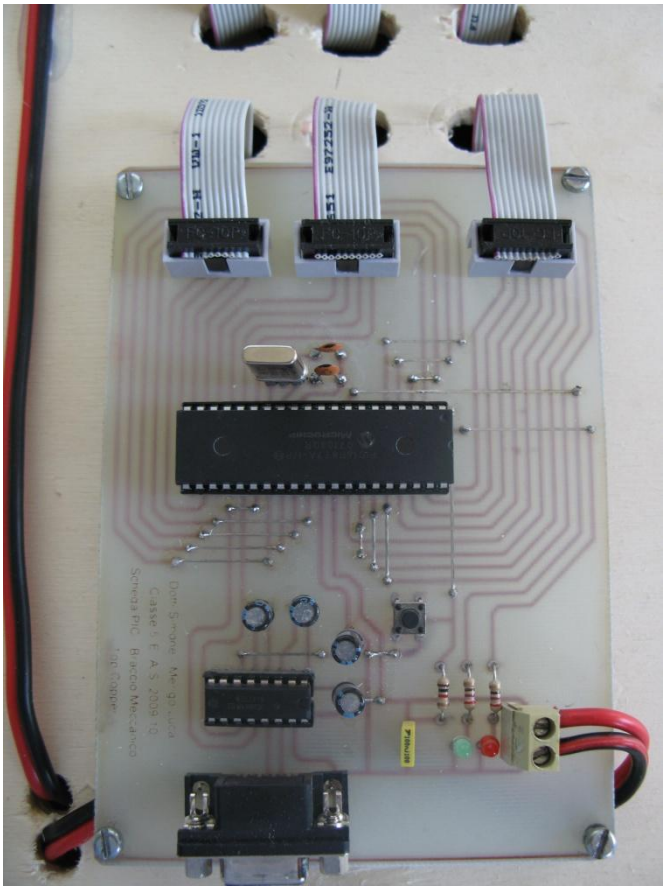
- Azionando entrambi i motori nello stesso verso è possibile inclinare la mano verso il basso o verso l'alto;



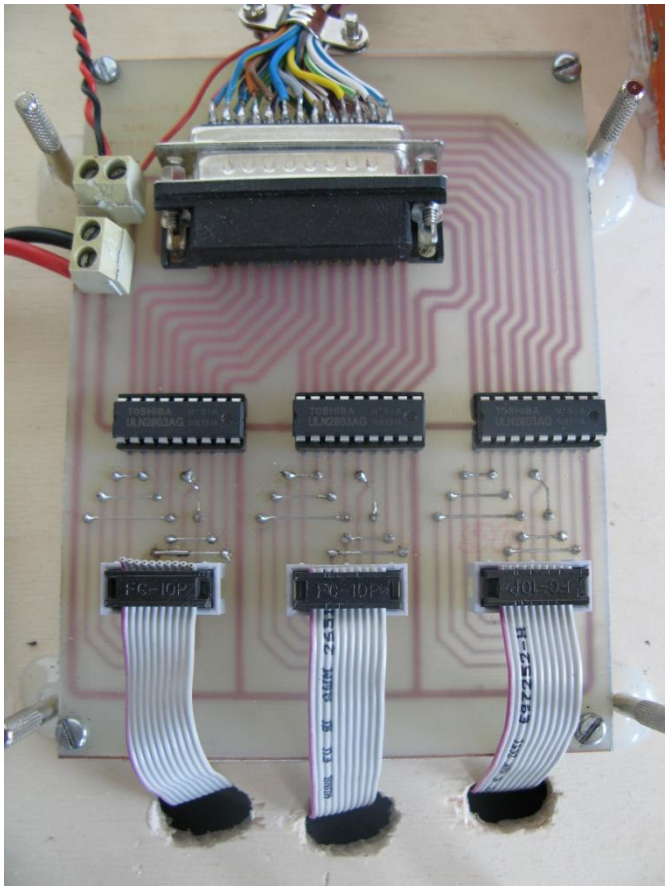
4.2) Le schede elettroniche

Per il progetto sono state realizzate due schede elettroniche: una dedicata alla gestione dei motori tramite microcontrollore (scheda 1), connesso al PC tramite un collegamento RS-232, ed una di potenza per il corretto azionamento dei motori (scheda 2).
Gli schematici e i successivi circuiti stampati sono stati elaborati con il software Eagle 4.13.
Per l'alimentazione delle schede è stato utilizzato un alimentatore per PC: esso infatti permette la stabilizzazione a 5 e 12 V della tensione di rete, necessarie rispettivamente per la scheda 1 e 2.
Esso inoltre eroga una corrente elevata, sufficiente per il pilotaggio dei motori del braccio.

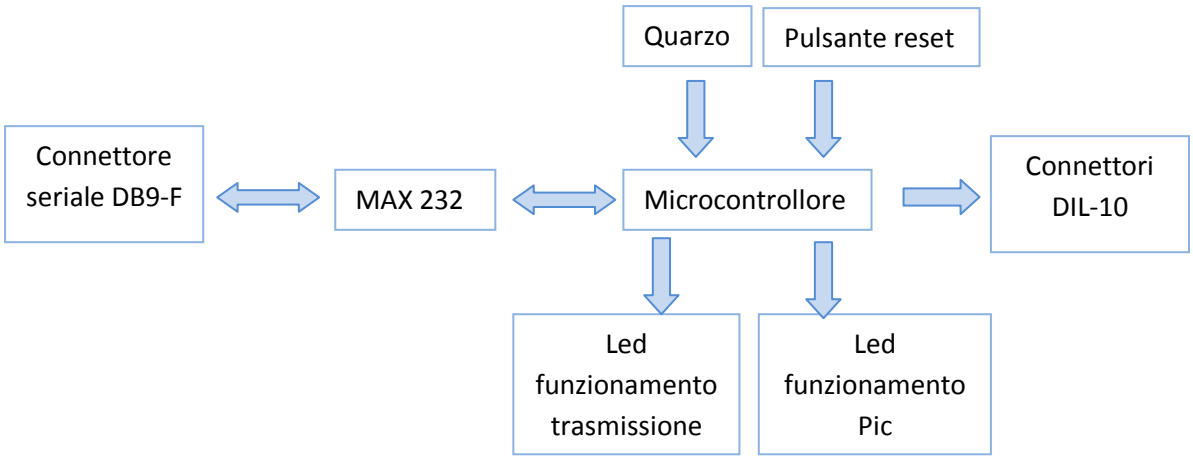
Scheda 1



Scheda 2



4.2.1) Scheda 1

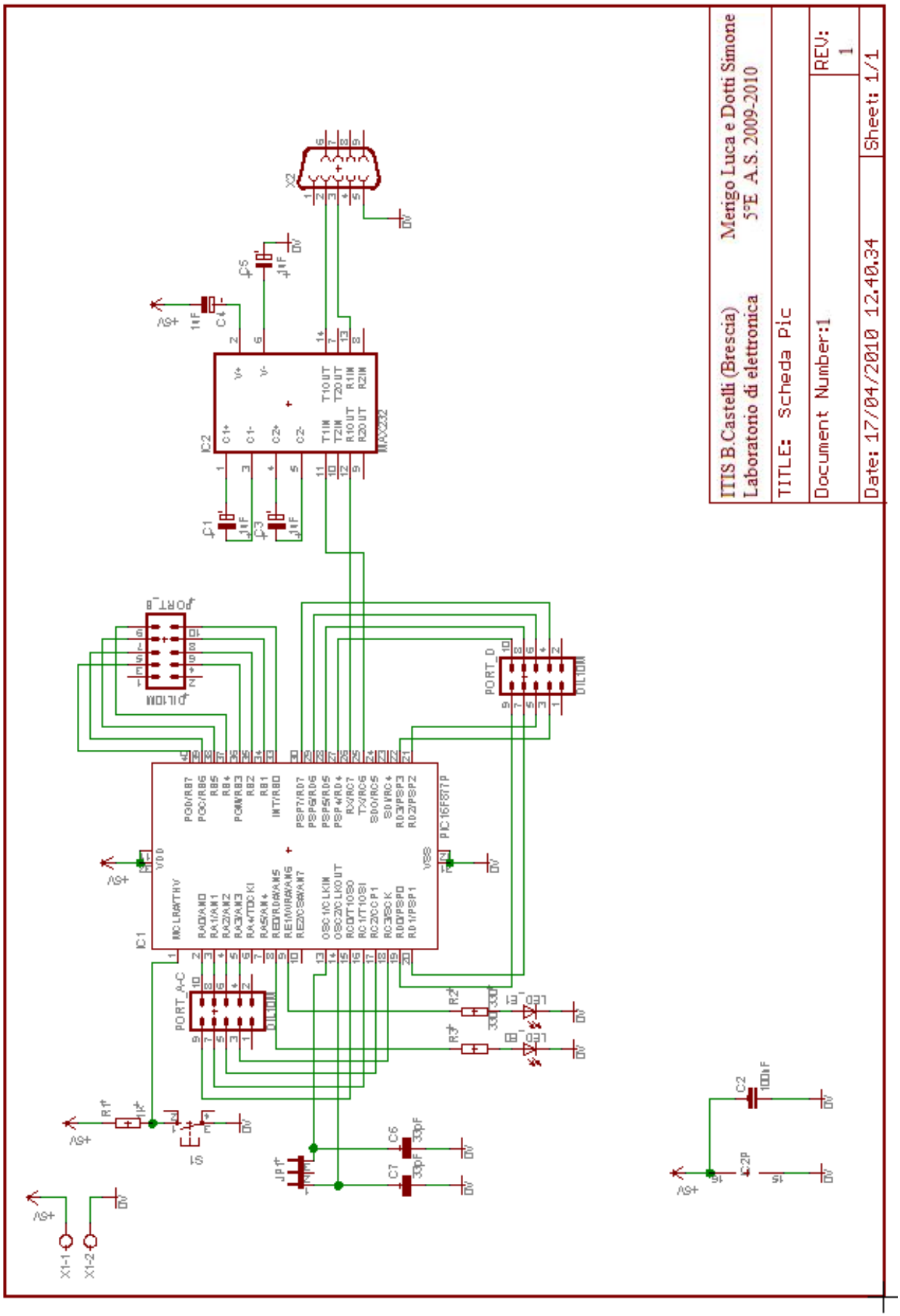


La scheda 1 comprende:
-Connettore DB9-F per la connessione con la porta seriale del PC;
-MAX232 per la conversione dei segnali TTL/standard RS-232 e viceversa;
-Microcontrollore per l'interpretazione dei dati ricevuti dal PC e per la gestione dei motori;
-Due led che segnalano il corretto funzionamento del PIC e della trasmissione;
-Pulsante reset per riavviare il microcontrollore;
-Quarzo per il funzionamento del microcontrollore;
-Connettori DIL-10 per la connessione del PIC all'interfaccia di potenza presente sulla scheda 2.

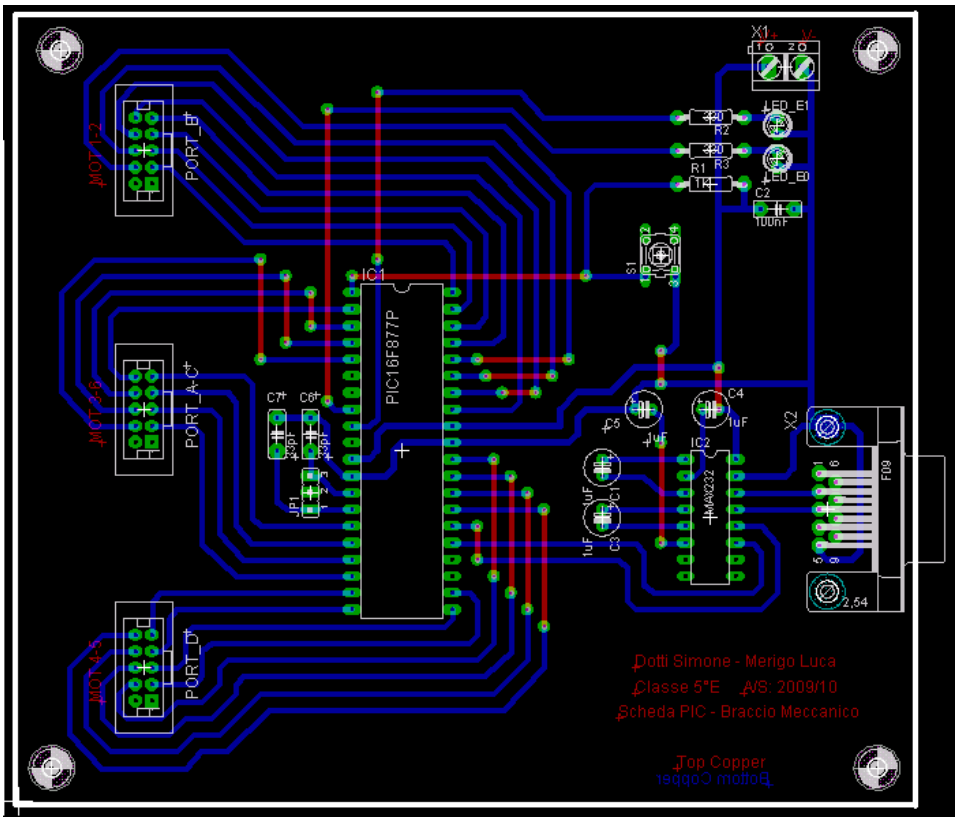
-Elenco dei componenti

Part	Value	Device	Package	Description
C1	1uF	CPOL-EUE2.5-6	E2,5-6	POLARIZED CAPACITOR
C2	100nF	C-EU050-025X075	C050-025X075	CAPACITOR
C3	1uF	CPOL-EUE2.5-6	E2,5-6	POLARIZED CAPACITOR
C4	1uF	CPOL-EUE2.5-6	E2,5-6	POLARIZED CAPACITOR
C5	1uF	CPOL-EUE2.5-6	E2,5-6	POLARIZED CAPACITOR
C6	33pF	C-EU050-025X075	C050-025X075	CAPACITOR
C7	33pF	C-EU050-025X075	C050-025X075	CAPACITOR
IC1	PIC16F877A	PIC16F877A	DIL40	MICROCONTROLLER
IC2	MAX232	MAX232	DIL16	RS232 TRANSEIVER
JP1		JP2E	JP2	JUMPER
LED_E0		LED3MM	LED3MM	LED
LED_E1		LED3MM	LED3MM	LED
PORT_A-C	DIL10M	DIL10M	DIL10PM	
PORT_B	DIL10M	DIL10M	DIL10PM	
PORT_D	DIL10M	DIL10M	DIL10PM	
R1	1K Ω	RCL_R-EU_0207/10	RCL_0207/10	RESISTOR
R2	330Ω	RCL_R-EU_0207/10	RCL_0207/10	RESISTOR
R3	330Ω	RCL_R-EU_0207/10	RCL_0207/10	RESISTOR
S1		10-XX	B3F-10XX	OMRON SWITCH
X1		AK500/2	AK500/2	CONNECTOR
X2		CON-SUBD_F09H	CON-SUBD_F09H	SUB-D

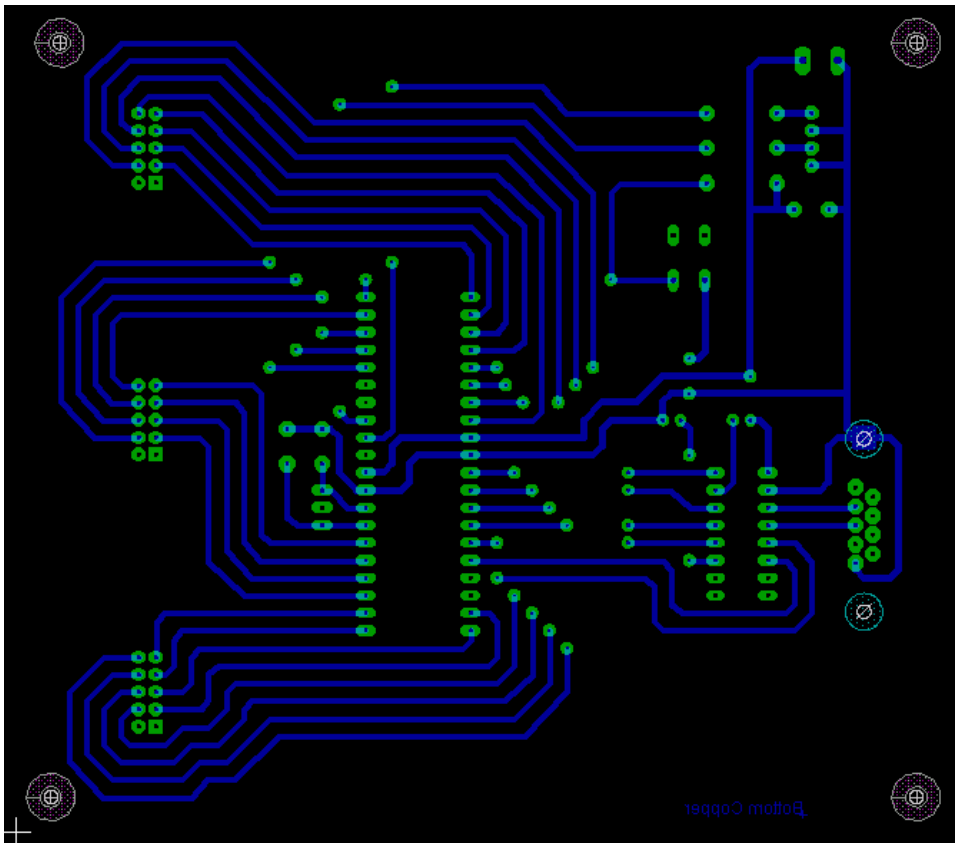
-Schematico


ITIS B.Castelli (Brescia)
Laboratorio di elettronica
Merigo Luca e Dotti Simone
5°E A.S. 2009-2010
TITLE: Scheda Pic
Document Number:1
REV: 1
Date: 17/04/2010 12.40.34
Sheet: 1/1
IC1
PIC16F877P
IC2
MAX232
X2
PORT_A-C
PORT_B
PORT_D
LED1
LED2
S1
JP1
X1-1
X1-2
C1
C2
C3
C4
C5
C6
C7
R1
R2
R3
+5V
0V

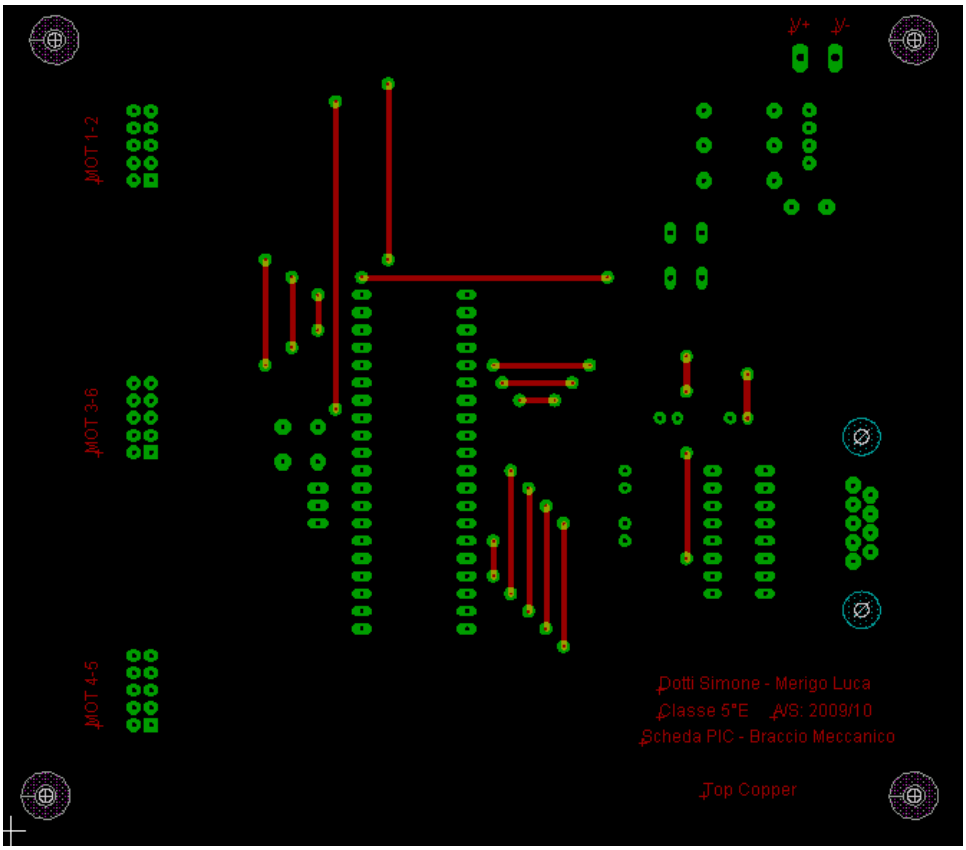
-Circuito stampato



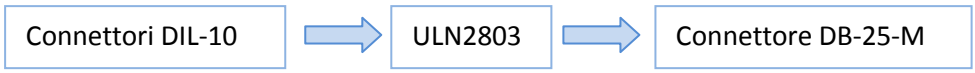
Bottom



Top


V+
V-
MOT 1-2
MOT 3-6
MOT 4-5
Dotti Simone - Merigo Luca
Classe 5°E A/S: 2009/10
Scheda PIC - Braccio Meccanico
Top Copper

4.2.2) Scheda 2

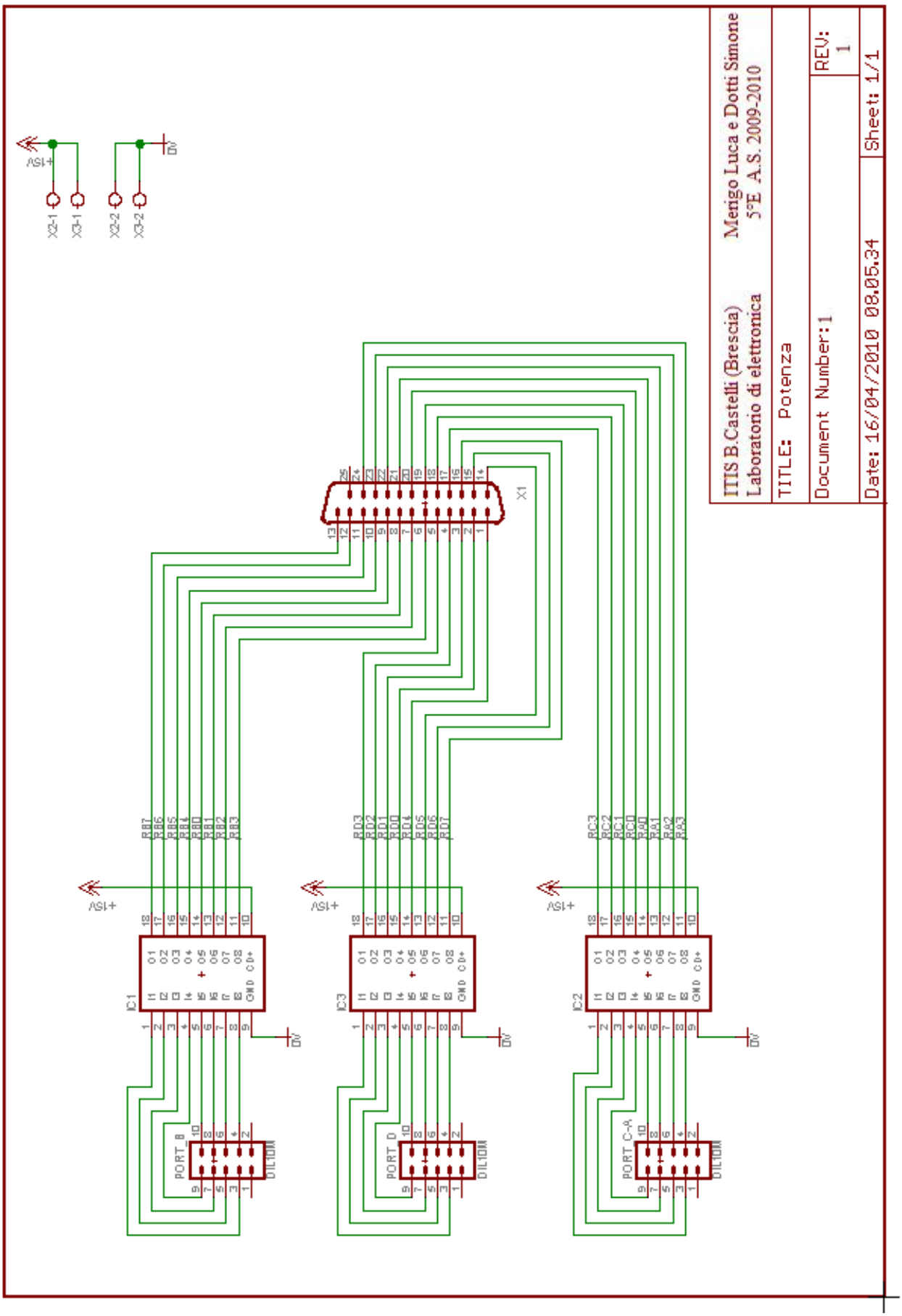


La scheda 2 comprende:
-Connettori DIL-10 per la connessione col microcontrollore presente sulla scheda 1;
-ULN2803 per l'interfaccia di potenza necessaria per l'azionamento dei motori;
-Connettore DB-25-M per la connessione degli ULN2803 ai motori del braccio meccanico.

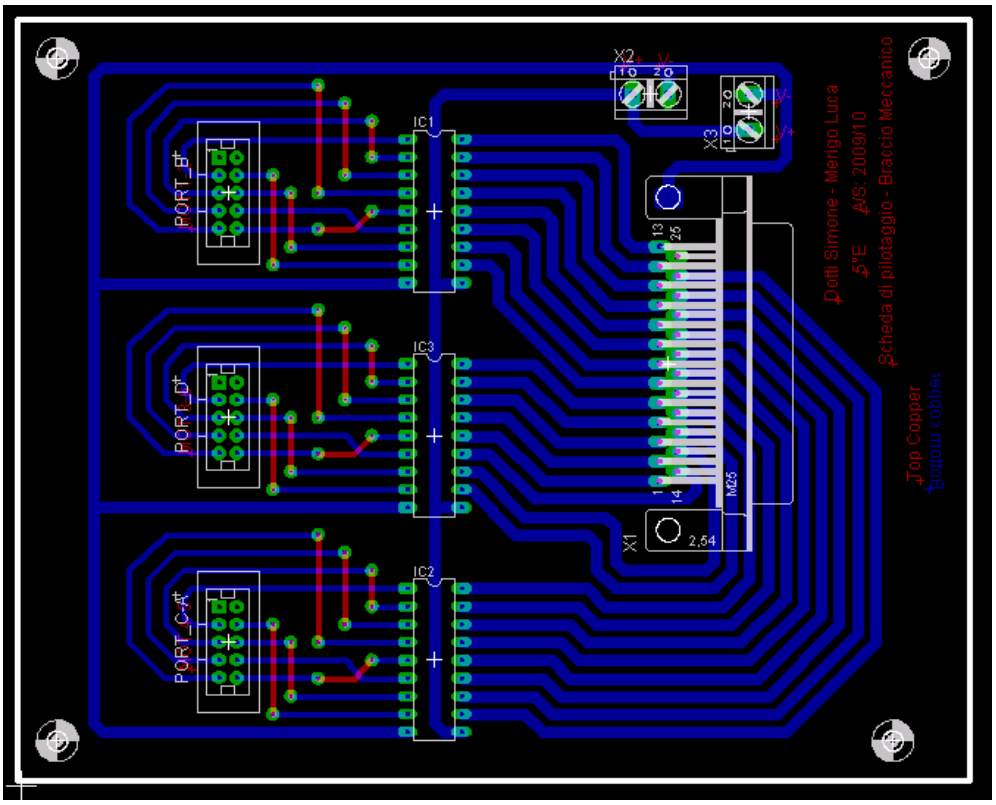
-Elenco dei componenti

Part	Value	Device	Package	Description
IC1		ULN2803	DIL18	DRIVER ARRAY
IC2		ULN2803	DIL18	DRIVER ARRAY
IC3		ULN2803	DIL18	DRIVER ARRAY
PORT_B	DIL10M	DIL10M	DIL10PM	
PORT_C-A	DIL10M	DIL10M	DIL10PM	
PORT_D	DIL10M	DIL10M	DIL10PM	
X1		CON-SUBD_M25H	CON-SUBD_M25H	SUB-D
X2		AK500/2	AK500/2	CONNECTOR
X3		AK500/2	AK500/2	CONNECTOR

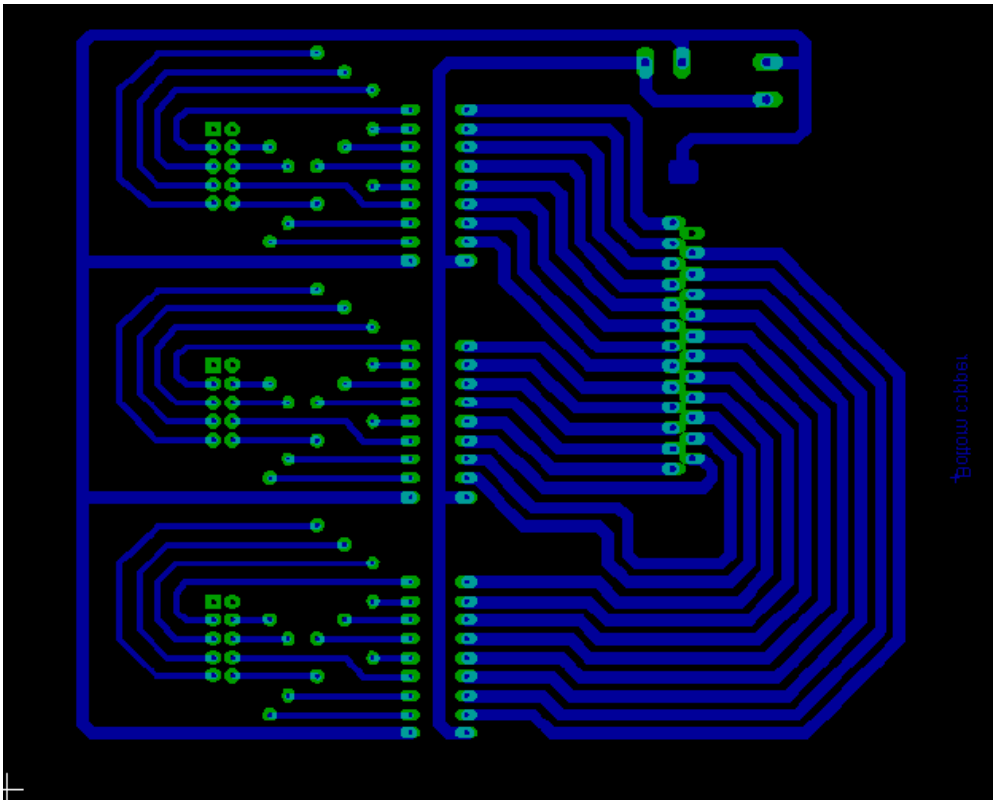
-Schematico


ITIS B.Castelli (Brescia)
Laboratorio di elettronica
Merigo Luca e Dotti Simone
5°E A.S. 2009-2010
TITLE: Potenza
Document Number: 1
REV: 1
Date: 16/04/2010 08.05.34
Sheet: 1/1
+15V
0V
X2-1
X3-1
X2-2
X3-2
X1
IC1
IC2
IC3
PORT_B
PORT_D
PORT_C-A
DIL10DM

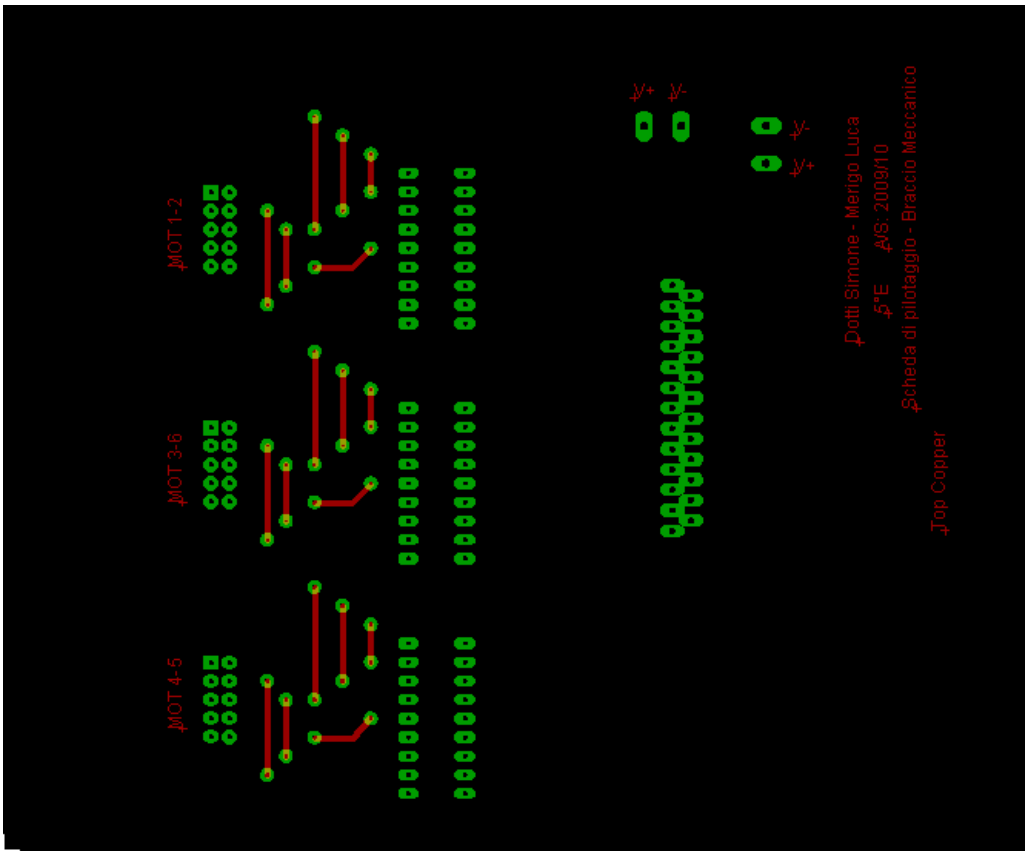
-Circuito stampato



Bottom



Top


MOT 1-2
MOT 3-6
MOT 4-5
V+ V-
V-
V+
Dotti Simone - Merigo Luca
5°E A/S: 2009/10
Scheda di pilotaggio - Braccio Meccanico
Top Copper

4.3) Il software

4.3.1) Introduzione

Per il progetto del Mini Robot sono stati realizzati due software: uno di interfaccia a PC in Delphi e uno di gestione dei motori sul microcontrollore.
La connessione RS-232 dei due apparati rende possibile la comunicazione tra PC e microcontrollore, i quali si scambiano dati per permettere il corretto funzionamento del sistema.
In particolare il software di interfaccia a PC gestisce:
-la tastiera del PC, con la quale l'utente comanda i movimenti del braccio;
-la memorizzazione, l'ottimizzazione e la riproduzione di un percorso;
-il controllo del corretto funzionamento della trasmissione (handshake).
Il microcontrollore si occupa invece del pilotaggio dei motori passo-passo; in particolare:
-trasmette al PC il numero dei passi effettuati dai motori in fase di memorizzazione di un percorso;
-pilota i motori in fase di riproduzione ad una certa velocità e facendo compiere ad essi un certo numero di passi comunicati dal PC.
Entrambi i software sono stati realizzati con la tecnica della macchina a stati: i vari compiti che devono essere svolti dai due apparati, sono stati suddivisi in vari automi per semplificare e gestire in maniera ottimale l'intero sistema.
Lo stato dei i vari automi è determinato dalla particolare situazione in cui può trovarsi il sistema:
-Movimento libero: l'utente pilota il braccio con l'utilizzo della tastiera;
-Memorizzazione percorso: l'utente, oltre a pilotare il braccio con la tasiera, fa memorizzare al PC dei punti che determineranno il percorso;
-Riproduzione percorso: l'utente non gestisce il braccio ma il PC invia al microcontrollore il numero dei passi e le velocità per i vari motori in modo da eseguire un percorso precedentemente memorizzato; l'utente può fermare la riproduzione momentaneamente o definitivamente;
-Errore: la trasmissione non funzione; è in questo caso necessario che l'utente riavvii il sistema.

4.3.2) Interfaccia Delphi

Il software in Delphi è costituito da tre automi:
-automa 1: avvia la comunicazione tra PC e microcontrollore su richiesta dell'utente e si occupa della modalità movimento libero;
-automa 2: gestisce la memorizzazione, l'ottimizzazione, la riproduzione e il salvataggio su file esterno del percorso;
-automa 3: gestisce l'handshake per verificare il corretto funzionamento della trasmissione.

-Automa 1

Di seguito vengono riportati i vari stati in cui si può trovare l'automa 1:

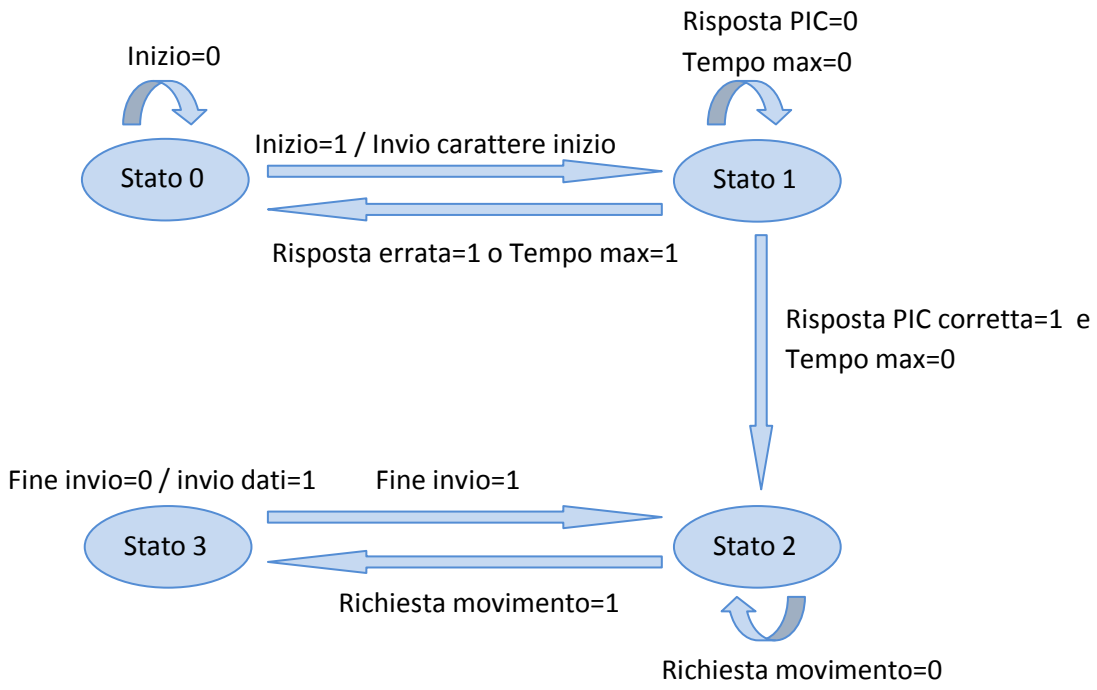
-Stato 0: il software attende che l'utente segnali l'avvio del sistema (con un apposito pulsante); dopodiché viene inviato un carattere che segnala l'inizio della comunicazione;

-Stato 1: si controlla se il microcontrollore ha risposto entro un certo tempo prestabilito: se questo non avviene, viene segnalato all'utente un problema sulla trasmissione.

-Stato 2: viene gestita la modalità di movimento libero: in particolare si controlla se l'utente, utilizzando la tastiera, ha richiesto un movimento del braccio.

-Stato 3: vengono inviati al microcontrollore dei dati che segnalano quali motori dovrà pilotare e in che verso devono ruotare.

-Stato 4: è uno stato temporaneo dell'automa utilizzato durante la riproduzione del percorso: in questo stato l'automa è fermo in quanto i motori non vengono pilotati dall'utente.



-Automa 2

Di seguito vengono riportati i vari stati in cui si può trovare l'automa 2:

-Stato 0: il software controlla se l'utente ha richiesto uno dei seguenti comandi: memorizzazione di un punto, fine della memorizzazione, inizio della riproduzione di un percorso. In base alla richiesta l'automa passerà ad allo stato corrispondente;

-Stato 1: in caso l'utente abbia richiesto la memorizzazione di un punto si attende la ricezione dei dati indicanti quanti passi hanno compiuto i vari motori; dopodiché si provvede ad interpretare i dati e salvare i valori ricevuti in una tabella;

-Stato 2: in caso l'utente abbia segnalato la fine della memorizzazione, si provvede all'invio di un carattere al PIC: esso indica al microcontrollore che da questo momento sarà il PC a indicare le velocità e il numero di passi per ogni motore fino a che il braccio non sarà ritornato nella posizione iniziale;

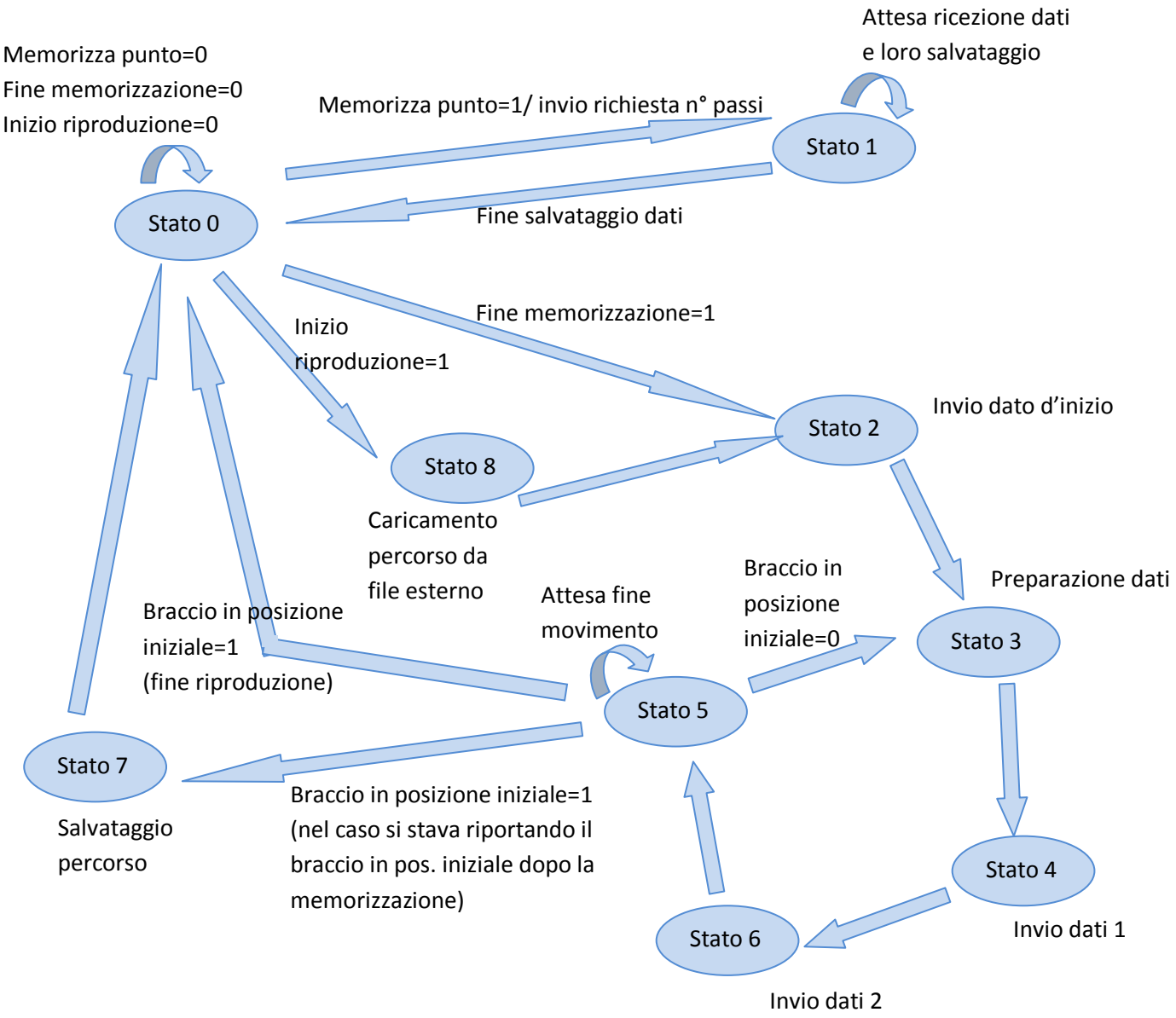
-Stato 3: vengono preparati i dati da inviare al PIC (passi + velocità) per spostare il braccio da un punto ad un altro in modo ottimale;

-Stato 4 e stato 6: invio dei dati al PIC;

-Stato 5: dopo aver inviato i dati, si attende la fine del movimento: durante questa attesa si controlla però l'intenzione dell'utente di fermare momentaneamente o definitivamente il movimento;

-Stato 7: Salvataggio del percorso memorizzato in un file esterno;

-Stato 8: Un percorso precedentemente salvato in un file esterno viene ricaricato nel programma.


Attesa ricezione dati
e loro salvataggio
Memorizza punto=0
Fine memorizzazione=0
Inizio riproduzione=0
Memorizza punto=1/ invio richiesta n° passi
Stato 1
Stato 0
Fine salvataggio dati
Fine memorizzazione=1
Inizio
riproduzione=1
Invio dato d'inizio
Stato 2
Stato 8
Caricamento
percorso da
file esterno
Braccio in
posizione
iniziale=0
Preparazione dati
Attesa fine
movimento
Braccio in posizione
iniziale=1
(fine riproduzione)
Stato 3
Stato 5
Stato 7
Salvataggio
percorso
Braccio in posizione iniziale=1
(nel caso si stava riportando il
braccio in pos. iniziale dopo la
memorizzazione)
Stato 4
Invio dati 1
Stato 6
Invio dati 2

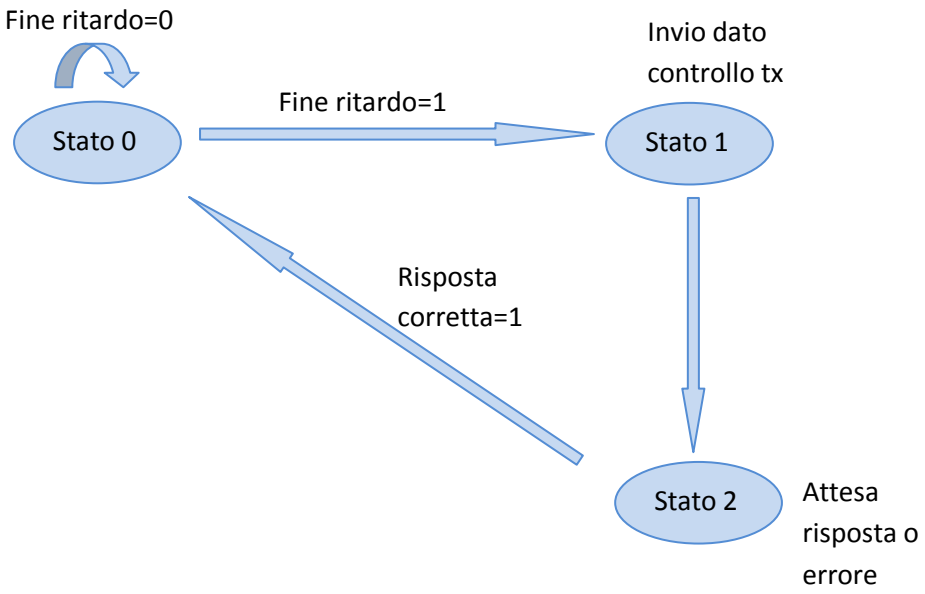
-Automa 3

Di seguito vengono riportati i vari stati in cui si può trovare l'automa 3:

-Stato 0: Viene realizzato un ritardo tra l'invio di un carattere di controllo ed il successivo (tempo di ritardo di 3 secondi);

-Stato 1: Viene inviato il carattere di controllo del corretto funzionamento della trasmissione;

-Stato 2: Si attende la risposta dal PIC; viene inoltre verificato che la risposta sia corretta e sia stata ricevuta entro un determinato intervallo di tempo: in caso questo non si verifichi si segnala all'utente il problema.



4.3.3) Programma del microcontrollore

Il software del PIC è costituito da 3 automi:
-automa 1: gestisce l'handshake col PC: in caso la trasmissione non funzioni segnala il problema all'utente con l'accensione di un led e blocca il programma;
-automa 2: gestisce la trasmissione col PC, interpreta i dati ricevuti e inizializza dei parametri utilizzati dal secondo automa (passi da effettuare, velocità e motori da pilotare);
-automa 3: gestisce il pilotaggio dei motori nelle tre modalità di funzionamento (movimento libero, memorizzazione e riproduzione).

-Automa 1

Di seguito vengono riportati i vari stati in cui si può trovare l'automa 1:

-Stato 0: attende il carattere di controllo del corretto funzionamento della trasmissione: in caso esso non arrivi entro un determinato intervallo di tempo, segnala all'utente il problema con l'accensione di un led e blocca il programma;

-Automa 2

Di seguito vengono riportati i vari stati in cui si può trovare l'automa 2:

-Stato 0: attende la ricezione del carattere di inizio comunicazione PC-PIC inviato dal PC in seguito al comando dell'utente;

-Stato 1: gestisce tutti i possibili dati ricevuti dal PC:
-riavvio del programma;
-dati su quali motori pilotare in modalità movimento libero;
-posizione iniziale del braccio;
-memorizza punto;
-fine memorizzazione del percorso;

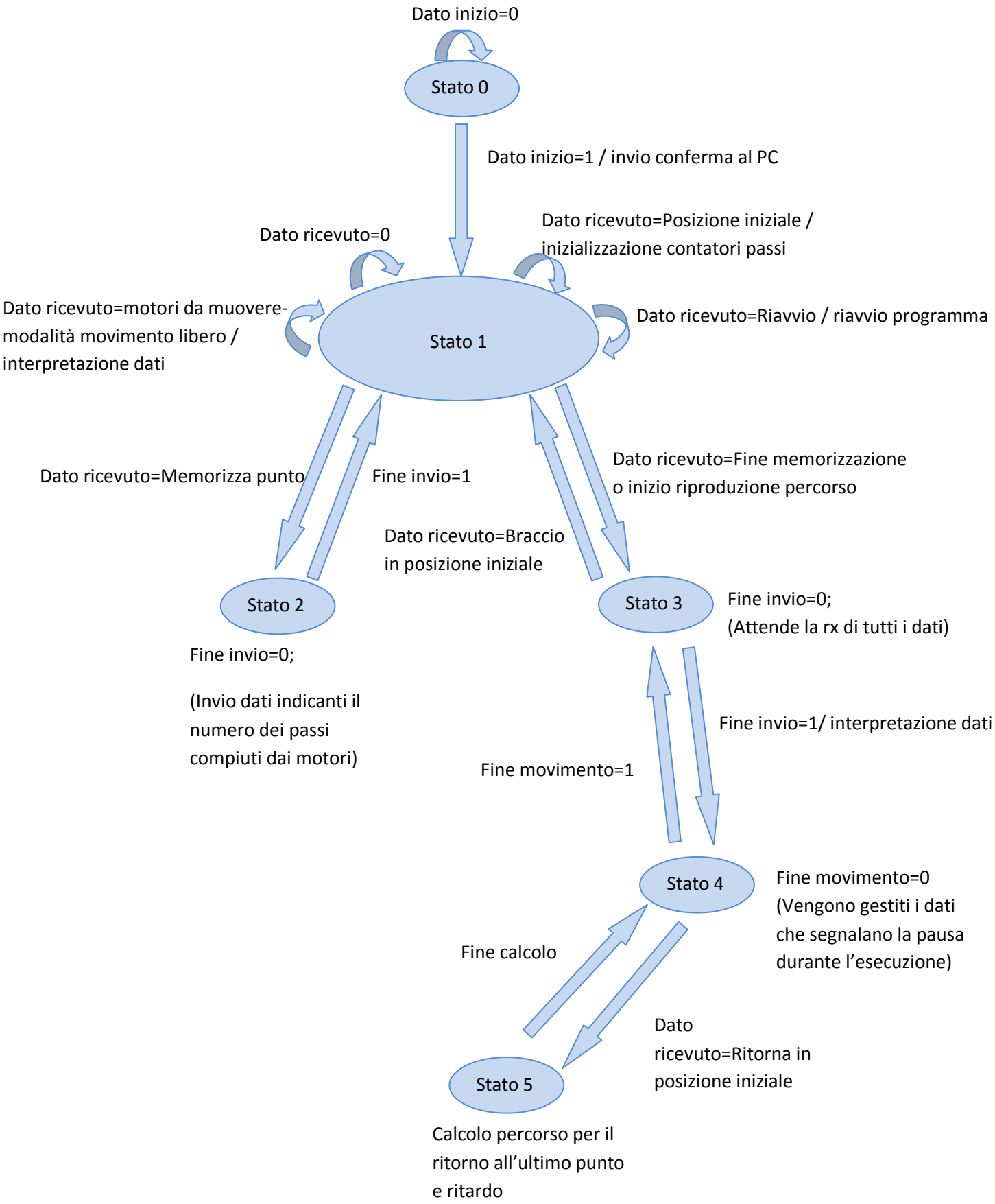
In base al dato ricevuto, l'automa si sposterà in uno stato specifico. Per ulteriori informazioni sui dati ricevuti vedere il capitolo successivo sul protocollo di trasmissione.

-Stato 2: gestisce l'invio del numero di passi effettuati dai singoli motori in modalità di memorizzazione di un percorso;

-Stato 3: gestisce i dati ricevuti dal PC indicanti la velocità e i numero dei passi per ogni motore nella modalità riproduzione di un percorso;

-Stato 4: mentre si esegue il percorso in modalità riproduzione controlla l'eventuale ricezione di dati (per la pausa o per il ritorno alla posizione iniziale);

-Stato 5: in caso l'utente decida di far ritornare il braccio in posizione iniziale durante la riproduzione, l'automa si porta in questo stato temporaneo. Esso serve per far tornare il braccio all'ultimo punto dal quale è partito per il movimento che si stava eseguendo quando l'utente ha premuto il pulsante di ritorno in posizione iniziale.


Dato inizio=0
Stato 0
Dato inizio=1 / invio conferma al PC
Dato ricevuto=0
Dato ricevuto=Posizione iniziale / inizializzazione contatori passi
Dato ricevuto=motori da muovere-modalità movimento libero / interpretazione dati
Stato 1
Dato ricevuto=Riavvio / riavvio programma
Dato ricevuto=Memorizza punto
Fine invio=1
Dato ricevuto=Fine memorizzazione o inizio riproduzione percorso
Dato ricevuto=Braccio in posizione iniziale
Stato 2
Stato 3
Fine invio=0;
(Attende la rx di tutti i dati)
Fine invio=0;
(Invio dati indicanti il numero dei passi compiuti dai motori)
Fine invio=1/ interpretazione dati
Fine movimento=1
Stato 4
Fine movimento=0
(Vengono gestiti i dati che segnalano la pausa durante l'esecuzione)
Fine calcolo
Dato ricevuto=Ritorna in posizione iniziale
Stato 5
Calcolo percorso per il ritorno all'ultimo punto e ritardo

-Automa 3

Di seguito vengono riportati i vari stati in cui si può trovare l'automa 3:

-Stato 0: gestisce i motori in modalità movimento libero e memorizzazione del percorso;

-Stato 1: gestisce i motori in modalità riproduzione del percorso;

Lo stato di questo automa viene determinato dai dati ricevuti e interpretati dall'automa 1.

4.3.4) Il protocollo di trasmissione

PC e PIC sono connessi tra loro con interfaccia RS-232 per permettere lo scambio dei dati. In particolare vengono inviati due tipi di dato:
-caratteri di controllo e di indicazione: indicano al PIC la particolare situazione in cui si trova il sistema, eventuali richieste o semplicemente il controllo della trasmissione;
-dati indicanti il numero dei passi effettuati o da effettuare.
Di seguito verranno trattate in dettaglio queste due tipologie di dato.

-Caratteri di controllo e di indicazione

Sono costituiti da semplici caratteri ASCII e svolgono ognuno una particolare funzione; in particolare:

Carattere	Funzione	Inviato da:
o	-Inizio comunicazione tra PC e PIC e riavvio sistema	PC
n	-Conferma della ricezione del carattere di inizio/riavvio	PIC
m	-Controllo del corretto funzionamento della trasmissione	PC
l	-Conferma della ricezione del carattere di controllo tx	PIC
y	-Indica che il braccio è stato posizionato in posizione iniziale	PC
x	-Indica la funzione memorizzazione punto	PC
z	-Fine memorizzazione del percorso; -Braccio ritornato in posizione iniziale; -Fine riproduzione percorso;	PC
p	-Pausa durante la riproduzione del percorso	PC
r	-Riprendi dopo la pausa durante la riproduzione del percorso	PC
s	-Ritorno del braccio in posizione iniziale (il braccio stava dirigendosi verso l'ultimo punto memorizzato)	PC
I	-Ritorno del braccio in posizione iniziale (il braccio stava dirigendosi verso la posizione iniziale)	PC
u	-Fine movimento in modalità riproduzione percorso	PIC

-Caratteri dato

In modalità memorizzazione e riproduzione di un percorso, PC e PIC si devono scambiare i passi effettuati o che devono essere effettuati dai motori; si tratta quindi di numeri che vengono però inviati in un modo particolare. Per semplicità viene proposto un esempio: un motore ha compiuto 234 passi in avanti. Il PIC invierà quindi i seguenti caratteri: “0”-“2”-“3”-“4”. Il numero viene quindi diviso in quattro caratteri che vengono inviati uno di seguito all’altro al posto dell’invio del numero intero. Questo potrebbe sembrare a prima vista uno svantaggio in quanto rallenta la trasmissione e necessita di funzioni software più complesse per ricostruire il numero. In verità questo viene fatto per agevolare il controllo della trasmissione in caso che il sistema non funzioni o abbia problemi legati alla comunicazione seriale. L’invio di caratteri ASCII ben definiti e non numeri e l’utilizzo di Hyper Terminal rende semplice questa verifica in quanto vengono visualizzate a PC delle lettere e non simboli legati ai numeri. I ritardi dovuti all’uso di questo tipo di comunicazione nel progetto del MINI ROBOT sono comunque limitati e trascurabili in quanto l’intero sistema è lento: vi sono infatti dei ritardi tra un movimento e l’altro per non danneggiare i motori che permettono la trasmissione dei dati, che sono comunque limitati. Nel paragrafo successivo sarà possibile consultare come il software gestisce, crea e interpreta questo tipo di dati.

In dettaglio i caratteri dato inviati sono in seguenti:

Verso trasmissione	Numero caratteri	Descrizione
Da PIC a PC	24	Indicano il numero dei passi compiuti dai motori in fase di memorizzazione di un percorso; per ogni motore vengono inviati 4 caratteri, in quanto il numero massimo di passi che ogni motore non può superare (per limiti meccanici) è 9999 passi. Il segno del numero ricostruito indicherà il verso di rotazione del motore: -negativo=indietro; -positivo=avanti.
Da PC a PIC	6	Indicano i motori da muovere e il verso in fase di memorizzazione di un percorso o in modalità movimento libero. In particolare per ogni motore viene inviato: -0=motore fermo; -1=motore avanti; -2=motore indietro;
Da PC a PIC	48	Indicano i passi e le velocità per ciascun motore in modalità riproduzione percorso. Anche in questo caso il segno del numero ricostruito indicherà il verso di rotazione del motore.

4.3.5) Programma del PIC

-Note iniziali

-Il microcontrollore lavora con un quarzo esterno da 20MHz e, utilizzando il modulo interno TMR0, viene generato il clock per le macchine a stati di 2ms.
-Il programma contiene salvati 3 array di caratteri che contengono le sequenze che devono essere utilizzate per pilotare i motori nelle varie modalità (mezzo passo, doppia fase, singola fase).
-I motori e i led controllati dal PIC sono così collegati:

Motore 1	Da RB4 a RB7
Motore 2	Da RB0 a RB3
Motore 3	Da RD0 a RD3
Motore 4	Da RC0 a RC3
Motore 5	Da RA0 a RA3
Motore 6	Da RD4 a RD7
Led funz. Trasmissione	RE1
Led funz. PIC	RE0

-I dati ricevuti dalla trasmissione seriale vengono salvati in un array: questo viene realizzato nella routine di interrupt.
-I dati da inviare, invece, non vengono gestiti in modo asincrono all'automa: è infatti la macchina a stati che, in particolari stati, provvede ad inviare un dato ad ogni ciclo di clock. L'insieme dei caratteri da inviare sono contenuti in un array.

Di seguito verranno riportati i flow chart di maggior importanza o di maggior complessità del programma del microcontrollore. Nella sezione allegati vi sarà poi l'intero programma.

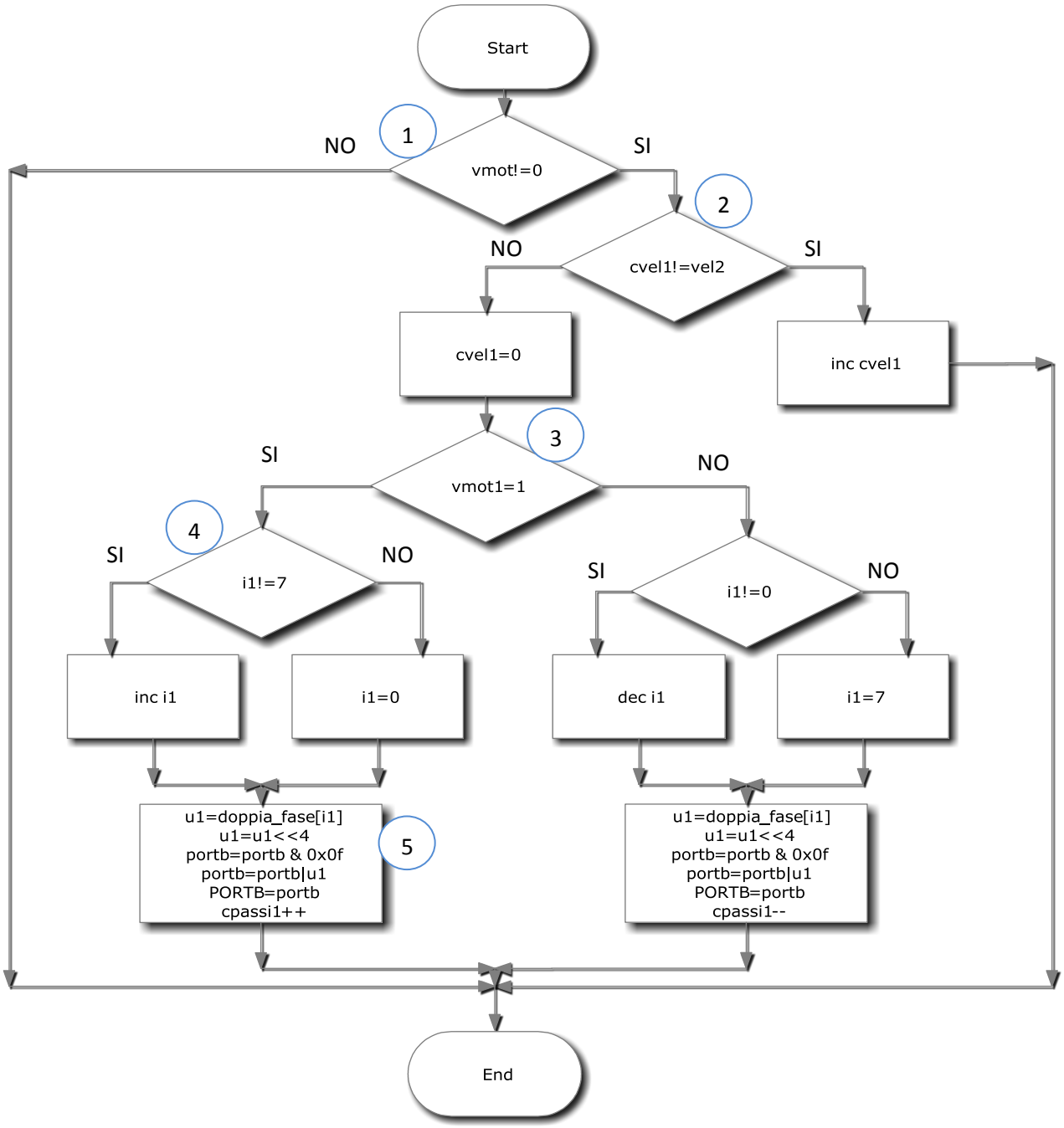
-Pilotaggio motori in modalità libera

Viene presa come esempio la funzione che pilota il motore 1 (automa 3-stato 0). Le procedure corrispondenti agli altri motori sono analoghe a questa.

Software:

```
        if (vmot1!=0){
                if (cvel1!=vel2)
                 cvel1++;
                else {
                        cvel1=0;
                        if (vmot1==1) {
                         if (i1!=7)
                         i1++;
                         else i1=0;
                         u1=doppia_fase[i1];
                         u1=u1<<4;
                         portb=portb & 0x0f;
                         portb=portb|u1;
                         PORTB=portb;
                         cpassi1++;
                        }
                        else {
                         if (i1!=0)
                          i1--;
                         else i1=7;
                         u1=doppia_fase[i1];
                         u1=u1<<4;
                         portb=portb & 0x0f;
                         portb=portb|u1;
                         PORTB=portb;
                         cpassi1--;
                        }

                }
}
```

Flow chart:



1) La variabile “vmot1” viene inizializzata dall’automa 1; quest’ultimo interpreta i 6 dati ricevuti in questa modalità di funzionamento che indicano quali motori pilotare e in che verso e assegna alla variabile uno dei seguenti valori:

0	Motore 1 fermo
1	Motore 1 attivo-verso: aventi
2	Motore 1 attivo-verso: indietro

Quindi se vmot1=0 questa procedura non viene eseguita.

2) La variabile “cvel1” è un contatore necessario ad effettuare un ritardo tra un passo e l’altro del motore per pilotare quest’ultimo alla velocità prefissata (indicata dalla costante vel2). Per

effettuare il ritardo richiesto vengono contati i cicli d'automa da 2ms ciascuno. Se cvel1 non ha ancora raggiunto il valore di vel2 essa viene incrementata, altrimenti viene posta a 0.

3) Viene effettuato un test su vmot1: il motore verrà pilotato nel verso indicato dalla variabile (vedi anche punto 1).
4) La variabile "i1" è un contatore che viene utilizzato come puntatore all'array che contiene la sequenza che deve essere utilizzata per pilotare i motori nelle modalità prefissata (in questo caso doppia fase).
5) In questa sequenza di istruzioni viene posto sul PORT d'uscita la codifica necessaria per far avanzare di un passo il motore 1 (contenuta nel posto "i1" dell'array "doppia_fase"). Viene inoltre incrementata la variabile "cpassi1": essa è un contatore utilizzato per memorizzare i passi effettuati dal motore. Viene fatto l'analogo in caso vmot1=2: in questo caso il motore farà un passo all'indietro e la variabile "cpassi1" verrà decrementata.

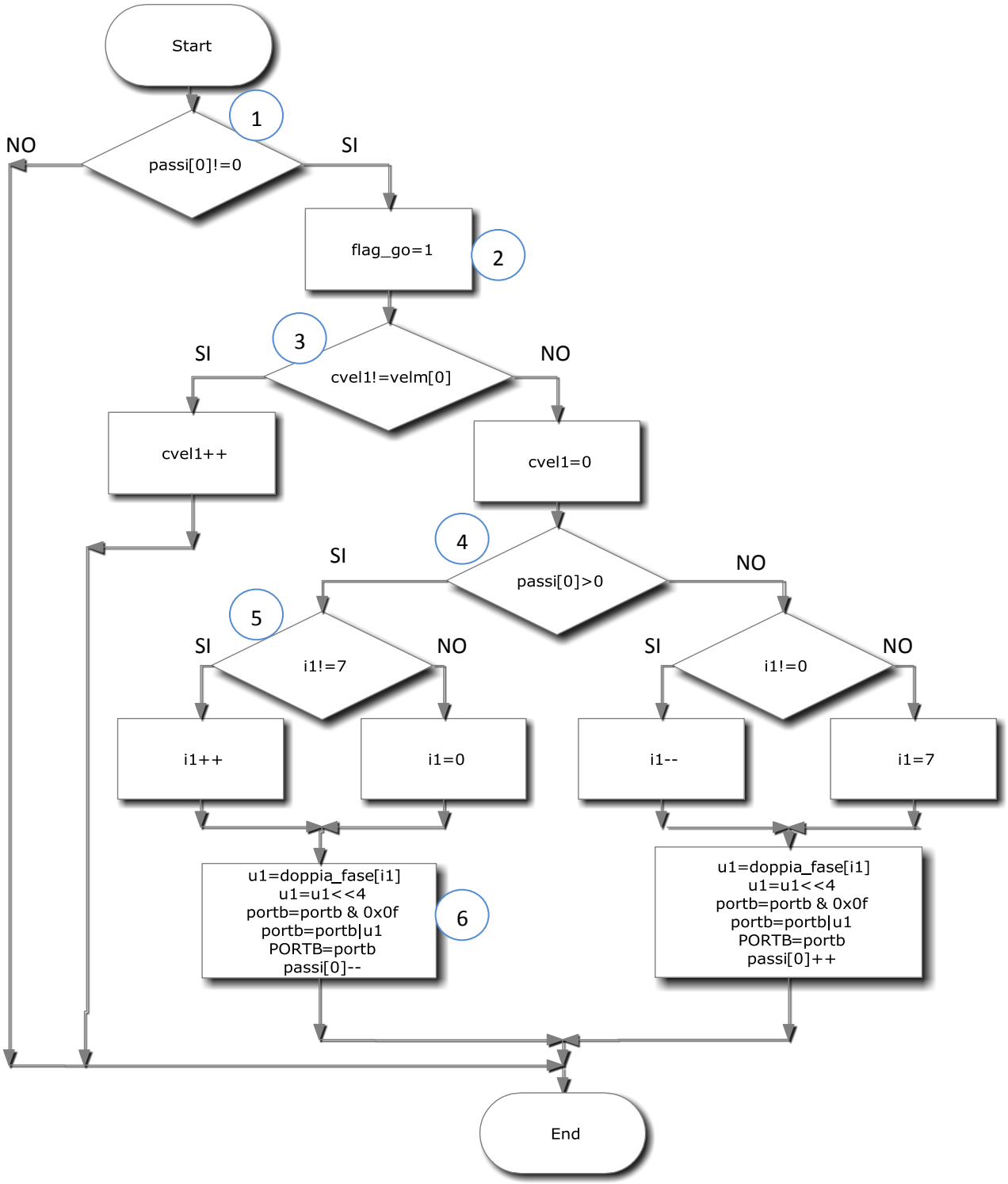
-Pilotaggio motori in modalità riproduzione percorso

Viene presa come esempio la funzione che pilota il motore 1 (automa 3-stato 1). Le procedure corrispondenti agli altri motori sono analoghe a questa.

Software:

```
if (passi[0]!=0){
        flag_go=1;
        if (cvel1!=velm[0])
         cvel1++;
        else {
                cvel1=0;
                if (passi[0]>0){
                 if (i1!=7)
                        i1++;
                 else i1=0;
                 u1=doppia_fase[i1];
                 u1=u1<<4;
                 portb=portb & 0x0f;
                 portb=portb|u1;
                 PORTB=portb;
            passi[0]--;
                }
                else{
                 if (i1!=0)
              i1--;
             else i1=7;
            u1=doppia_fase[i1];
            u1=u1<<4;
            portb=portb & 0x0f;
            portb=portb|u1;
            PORTB=portb;
            passi[0]++;
                }
        }
}
```

Flow chart:



1) Gli array passi[] e velm[] sono inizializzati dall'automa 1; quest'ultimo interpreta i 48 dati ricevuti in questa modalità di funzionamento che indicano il numero dei passi e la velocità di rotazione per ogni motore. I dati ricevuti e interpretati vengono salvati negli array passi[] e velm[]. In questo test iniziale si controlla se passi[0], corrispondente al numero dei passi che deve eseguire il motore 1, è diverso da 0: in caso contrario significa che il motore non deve essere pilotato e la procedura non viene eseguita.

2) “flag_go” è un flag che viene settato in caso passi[0] sia diverso da 0. Questo serve per indicare all’automa 1 che almeno un motore non ha ancora terminato il percorso. Quando flag_go sarà a livello logico basso significherà che i motori hanno terminato il movimento e l’automa 1 provvederà ad inviare al PC il carattere che indica la fine del movimento.
3) La variabile “cvel1” è un contatore necessario ad effettuare un ritardo tra un passo e l’altro del motore per pilotare quest’ultimo alla velocità prefissata (indicata nell’array velm[]). Per effettuare il ritardo richiesto vengono contati i cicli d’automa da 2ms ciascuno. Se cvel1 non ha ancora raggiunto il valore di velm[0] essa viene incrementata, altrimenti viene posta a 0.
4) Viene effettuato un test sul contenuto di passi[0]: a seconda del segno del valore presente nell’array verrà deciso il verso di rotazione (vedi anche punto 1).
5) La variabile “i1” è un contatore che viene utilizzato come puntatore all’array che contiene la sequenza che deve essere utilizzata per pilotare i motori nelle modalità prefissata (in questo caso doppia fase).
6) In questa sequenza di istruzioni viene posto sul PORT d’uscita la codifica necessaria per far avanzare di un passo il motore 1 (contenuta nel posto “i1” dell’array “doppia_fase”). Viene inoltre decrementato il contenuto dell’array passi[0]: essa infatti indica quanti passi deve ancora compiere il motore per completare il movimento. Viene fatto l’analogo in caso passi[0] sia negativo: in questo caso il motore farà un passo all’indietro e il contenuto dell’array passi[0] verrà incrementato.

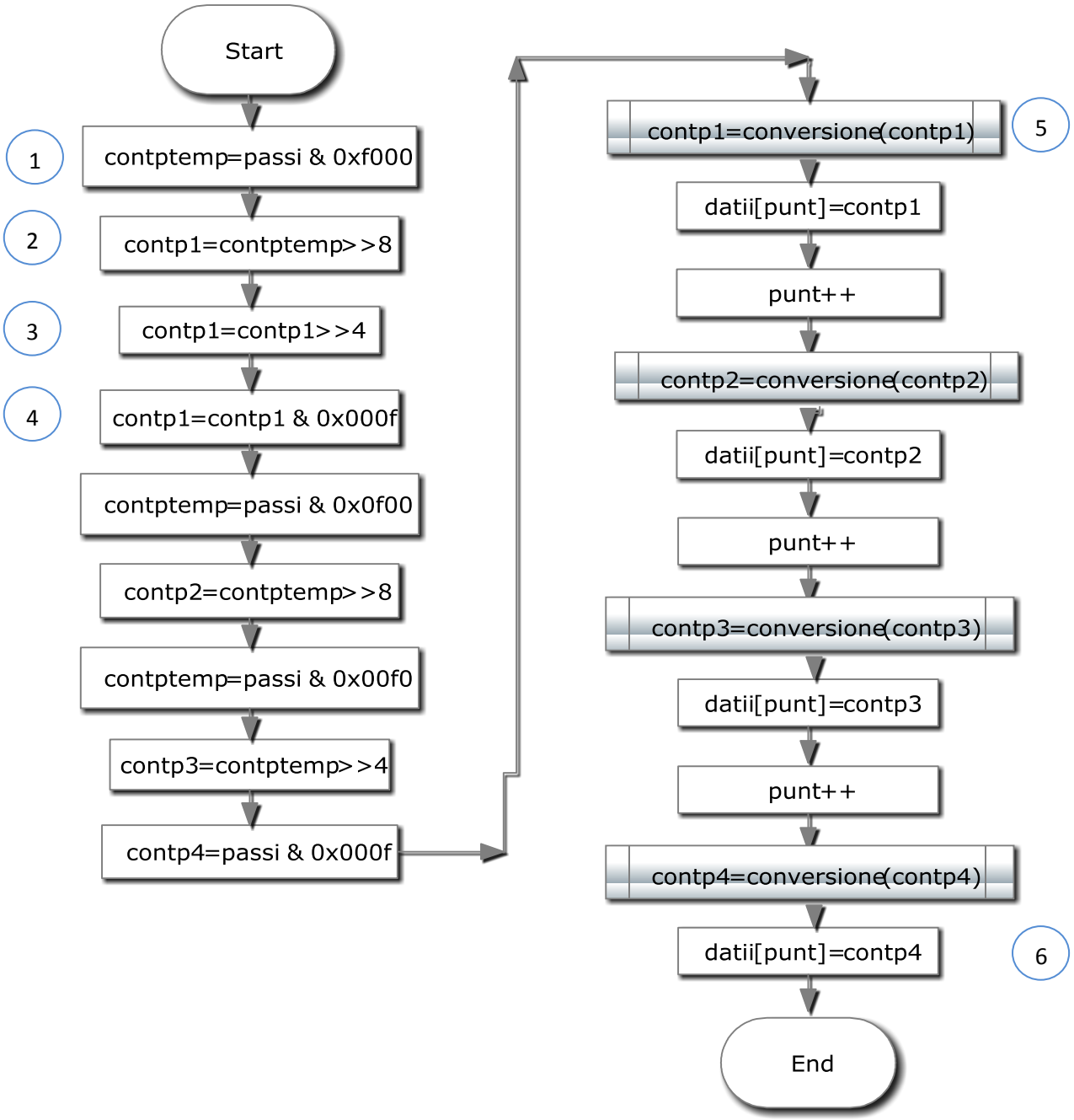
-Elaborazione dei dati da inviare

Questa procedura viene eseguita per elaborare i dati che devono essere inviati al PC, in particolare quelli indicanti i passi eseguiti dai motori in modalità memorizzazione percorso. Essi vengono salvati in un array chiamato “datii” e verranno in seguito trasmessi al PC dall’automa 1. Questa procedura è necessaria per codificare i caratteri nel modo indicato dal paragrafo 4.3.4. I passaggi eseguiti da questa funzione sono analoghi a quelli effettuati dal programma in Delphi per spedire dati al microcontrollore. A questa funzione vengono passati due parametri: “passi” che indica il numero dei passi eseguiti dal motore e “punt” che è un puntatore all’array “datii”.

Software:

```
{
        contptemp=passi & 0xf000;
        contp1=contptemp>>8;
        contp1=contp1>>4;
        contp1=contp1 & 0x000f;

        contptemp=passi & 0x0f00;
        contp2=contptemp>>8;

        contptemp=passi & 0x00f0;
        contp3=contptemp>>4;

        contp4=passi & 0x000f;

        contp1=conversione(contp1);
        datii[punt]=contp1;
        punt++;
        contp2=conversione(contp2);
        datii[punt]=contp2;
        punt++;
        contp3=conversione(contp3);
        datii[punt]=contp3;
        punt++;
        contp4=conversione(contp4);
        datii[punt]=contp4;
}
```

Flow chart:



Per capire il funzionamento di questa funzione viene proposto un esempio. Si suppone che il motore 1 abbia compiuto 5000 passi (1388 in esadecimale) e il puntatore all'array (punt) sia 0. Quindi:

1) conttemp = passi & 0xf000 = 0x1388 & 0xf000 = 0x1000;
And logico tra il numero di passi e 0x000f;

2)contp1 = contptemp>>8 = 0x1000>>8 = 0x0010;
shift a destra di 8;

3)contp1 = contp1>>4 = 0x0010>>4 = 0x0001;

4)contp1 = contp1 & 0x000f = 0x0001 & 0x000f =0x0001; //per sicurezza
Si ha quindi in contp1 la prima cifra del numero indicante il numero dei passi. La stessa cosa viene fatta per avere in contp2,3 e 4 le altre cifre. Quello che varia sono le operazioni di AND logico.

5) In questo punto della funzione si avrà:
-contp1=0x0001;
-contp2=0x0003;
-contp3=0x0008;
-contp4=0x0008.
Si richiama ora un'altra funzione che svolge la conversione dal numero al carattere ASCII corrispondente. Al punto 6 si avrà quindi:
-datii[0]='1';
-datii[1]='3';
-datii[2]='8';
-datii[3]='8';

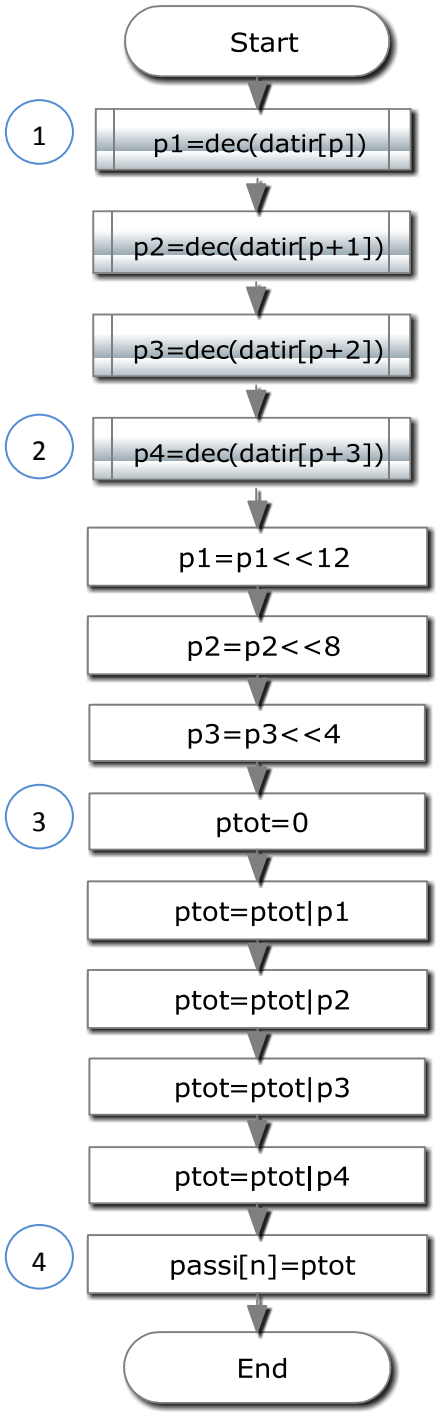
-Elaborazione dei dati ricevuti

Questa procedura viene eseguita per elaborare i dati vengono inviati dal PC, in particolare quelli indicanti i passi da eseguire dai motori in modalità riproduzione percorso. Essi vengono salvati in un array chiamato "datir"dall'interrupt della ricezione seriale. Questa procedura è necessaria per convertire i caratteri ASCII in numeri. I passaggi eseguiti da questa funzione sono analoghi a quelli effettuati dal programma in Delphi per interpretare i dati inviati dal microcontrollore. A questa funzione vengono passati il parametro "p" che è un puntatore all'array "datir".

Software:

```
{
        p1=dec(datir[p]);
        p2=dec(datir[p+1]);
        p3=dec(datir[p+2]);
        p4=dec(datir[p+3]);

        p1=p1<<12;
        p2=p2<<8;
        p3=p3<<4;

        ptot=0;
        ptot=ptot|p1;
        ptot=ptot|p2;
        ptot=ptot|p3;
        ptot=ptot|p4;

        passi[n]=ptot;
}
```

Flow chart:



Per capire il funzionamento di questa funzione viene proposto un esempio. Si suppone di aver "p"=0 e di aver ricevuto il numero dei passi che deve compiere il motore 1. Essi saranno stati ricevuti e salvati nell'array nel seguente modo:
-datir[0]='1';
-datir[2]='3';
-datir[2]='8';
-datir[3]='8'.

1)Viene richiamata una procedura che converte il carattere ASCII nel numero corrispondente. Al punto 2 si avrà quindi:
-p1=0x0001;
-p2=0x0003;
-p3=0x0008;
-p4=0x0008;

3)In seguito agli shift le variabili avranno i seguenti valori:
-p1=0x1000;
-p2=0x0300;
-p3=0x0080;
-p4=0x0008;

4)Alla fine della funzione si avrà, in seguito alle operazioni di OR logico, ptot=0x1388. Esso verrà poi salvato nell'array "passi[]".

4.3.6) Programma in Delphi

-Note iniziali

-In Delphi sono stati realizzati tre automi con un clock di 10ms.
-Come nel microcontrollore, la ricezione dei dati è asincrona al programma mentre l'invio viene gestito da particolari stati degli automi.
-Nel software è presente una matrice da 200 righe e 6 colonne necessaria per il salvataggio dei vari punti che compongono un percorso.

-Ottimizzazione del percorso

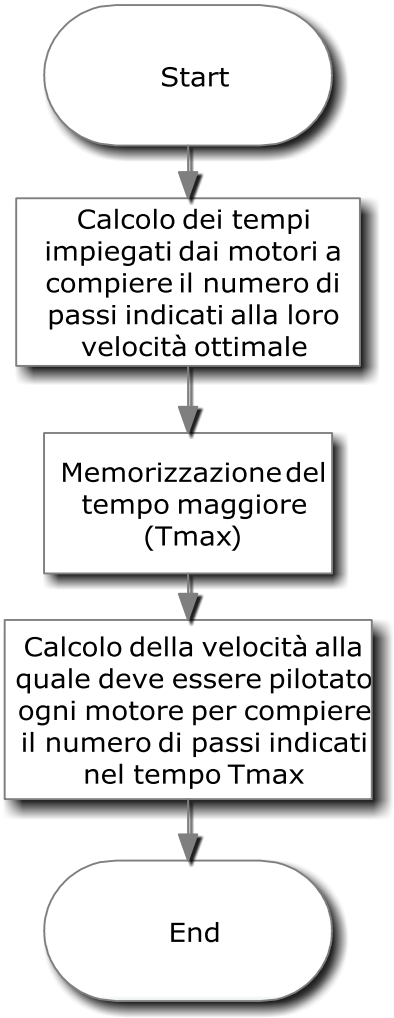
La maggior difficoltà riscontrata nella realizzazione del programma in Delphi riguarda l'ottimizzazione del percorso. Infatti, in seguito alla memorizzazione, si ha un percorso salvato nella matrice sopra citata nel seguente modo:
-Ogni riga corrisponde ad un punto del percorso;
-In ogni "casella" di una riga sono contenuti i passi che hanno compiuto i motori per spostare il braccio dal punto precedente.
Nella memorizzazione del percorso ogni motore era pilotato con la sua velocità ottimale di funzionamento. Se per riprodurre il percorso si utilizzasse la suddetta matrice, pilotando i motori alle loro velocità ottimale, non si avrebbe un ottimizzazione del percorso, in quanto quest'ultimo risulterebbe discontinuo a causa della mancanza di calcoli necessari per la regolazione della velocità. Il risultato che si vuole ottenere dall'ottimizzazione è di portare il braccio da un punto ad un altro muovendo contemporaneamente i motori a velocità diverse e pre-calcolate: questo dovrebbe permettere ai motori di compiere i passi necessari allo spostamento nello stesso intervallo di tempo, rendendo lineare il movimento.

Di seguito è spiegata in modo approfondito come viene realizzata questa ottimizzazione, facendo riferimento alla procedura che la esegue.

Per capire il funzionamento della funzione che realizza l'ottimizzazione del percorso viene proposto un esempio; supponiamo che in una certa riga della tabella ci siano i seguenti dati, indicanti il percorso da far compiere al braccio per portarlo al punto successivo:

N° Motore	1	2	3	4	5	6
N° passi	200	-50	400	-37	-350	125

Il software deve spedire al PIC i passi da effettuare e le velocità.
Per il calcolo delle velocità a cui deve essere pilotato ogni motore viene eseguito il seguente algoritmo:



Nel caso preso in considerazione:

	Motore 1	Motore 2	Motore 3	Motore 4	Motore 5	Motore 6
N° Passi	200	-50	400	-37	-350	125
Vmax[passi/s]	165	100	165	100	100	125
Tempo (N° passi/Vmax)[s]	1,21	0,5	2,42	0,37	3,5	2
Velocità ottimale (N° passi/Tmax)[passi/s]	57,14	14,29	114,29	10,57	100	35,71

Ovviamente le velocità ottenute sono ideali: in seguito ad approssimazioni il percorso non sarà ottimizzato al 100%. In seguito il programma provvederà a codificare le velocità con il metodo indicato nel paragrafo sul protocollo di trasmissione.

Parte Quinta

Conclusioni

5.1) Conclusioni

Il Mini Robot era stato iniziato da studenti dello scorso anno scolastico, ma non era stato portato a compimento per varie problematiche emerse durante il progetto. Pertanto è stato richiesto un lasso di tempo relativamente lungo per operazioni di riparazione e sistemazione della struttura meccanica del robot a partire dal mese di novembre.
In particolare, le operazioni di sostituzione e di successivo riposizionamento delle funicelle utilizzate per trasmettere il moto prodotto dai motori sono state particolarmente complicate, per via delle complesse strutture che le funicelle stesse formavano e per i determinati livelli di tensione a cui dovevano essere sottoposte.
Successivamente sono stati realizzati dei programmi di prova, in modo da testare i motori passo-passo del braccio meccanico e rilevarne le velocità di rotazione ideali. Questi programmi erano relativamente semplici, in quanto non prevedevano operazioni di memorizzazione e riproduzione, ma unicamente modalità di pilotaggio libero. Successivamente è stato realizzato il programma finale, ed ovviamente per quanto riguarda la modalità di pilotaggio libero la struttura utilizzata è stata analoga a quella dei programmi precedenti.
Nel complesso, gli obiettivi prefissati, ossia la possibilità di pilotare i vari motori del robot e di memorizzare e successivamente riprodurre ottimizzando particolari sequenze di azioni direttamente da PC, sono stati conseguiti con buoni risultati.
Analizzando il risultato finale, alcuni problemi non sono stati risolti per motivi di tempo o di eccessiva complessità. Ad esempio, il movimento dell'avambraccio del robot (come evidenziato nel paragrafo apposito) può essere significativamente migliorato, in quanto è irregolare e assolutamente non lineare.
A tal proposito sarebbe possibile posizionare una molla sul gomito del braccio, caratterizzata da un'adeguata forza resistente, in grado di linearizzare il movimento dell'avambraccio.
Un altro difetto è relativo alla pressione esercitata dalla mano del robot. Infatti, una volta afferrato un oggetto, un'eccessiva pressione comandata dall'utente tramite il programma su PC comporta un ulteriore movimento del motore preposto all'azionamento della mano, facendogli perdere i passi. Il problema viene evidenziato particolarmente durante il ritorno in posizione iniziale del robot, in quanto la mano presenta un grado di chiusura diverso da quello desiderato. Una delle possibili soluzioni a questa problematica può essere il posizionamento di un sensore di pressione (ad esempio un estensimetro) sulle dita della mano, in modo da permettere all'utente di rilevare il livello di pressione esercitata e quindi di evitare di applicarne una eccessiva.
Inoltre il software può essere ulteriormente ottimizzato, in modo da ridurne le dimensioni.
Possibili sviluppi del progetto potrebbero riguardare il montaggio sulla mano del robot di una videocamera, in grado di rilevare le sagome di vari oggetti e quindi permettere lo sviluppo di programmi più complessi oppure la comunicazione tra PC e microcontrollore tramite USB (Universal Serial Bus), ovviamente utilizzando un opportuno microcontrollore e un adeguato protocollo di trasmissione.

5.2) Ringraziamenti

Un doveroso ringraziamento va a tutte quelle persone che ci hanno offerto il loro supporto affinché il progetto funzionasse correttamente:

-I professori Belloni Marco, Stagnoli Gianfranco e Nastasio Maurizio, docenti di sistemi e Tdp, che hanno supervisionato e offerto i propri consigli e suggerimenti durante la realizzazione del progetto;
- La professoressa Costantini Barbara, docente di Telecomunicazioni, che ha supervisionato e corretto la presente relazione;
- La professoressa Civello Mariagrazia, che ha corretto la sezione in inglese della presente relazione;
-Tutti gli altri membri del consiglio di classe, che ci hanno messo a disposizione numerose ore per ultimare il progetto e la presente relazione;
- Gli assistenti dei laboratori del dipartimento di elettronica e meccanica;
Un ringraziamento anche alle nostre famiglie, che ci hanno supportato e sostenuto durante tutto l'anno scolastico.

Parte Sesta

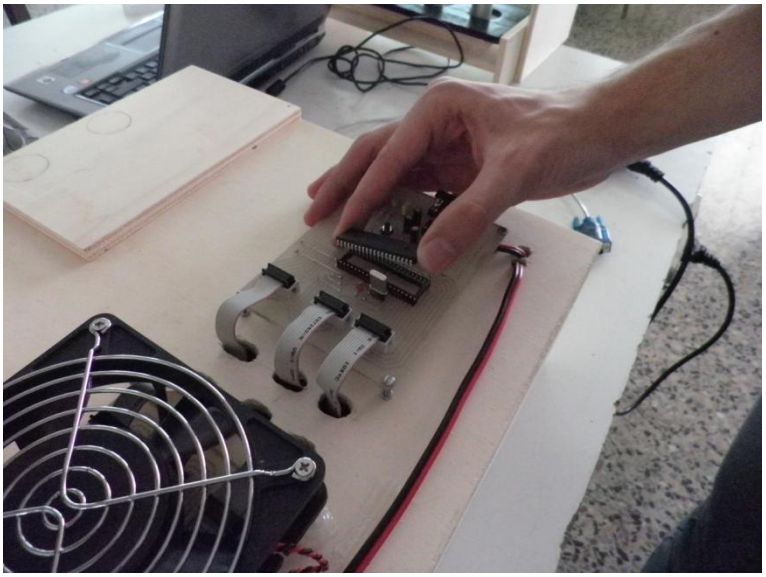
Allegati

6.1) Guida all'utilizzo del MINI ROBOT

-Operazioni iniziali

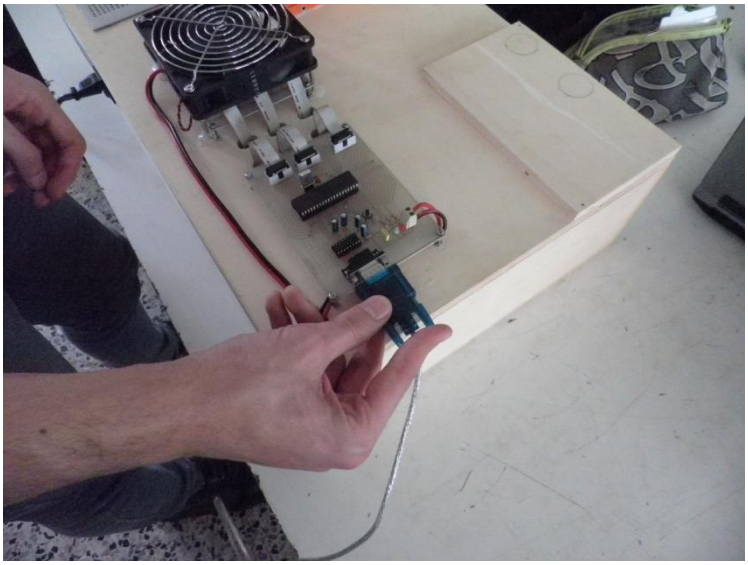
1) Inserire il PIC nell'apposito alloggiamento sulla scheda elettronica, avendo cura di inserirlo nel verso corretto.

2) Controllare che l'oscillatore al quarzo sia inserito correttamente.



3) Connettere l'alimentazione alla rete elettrica.

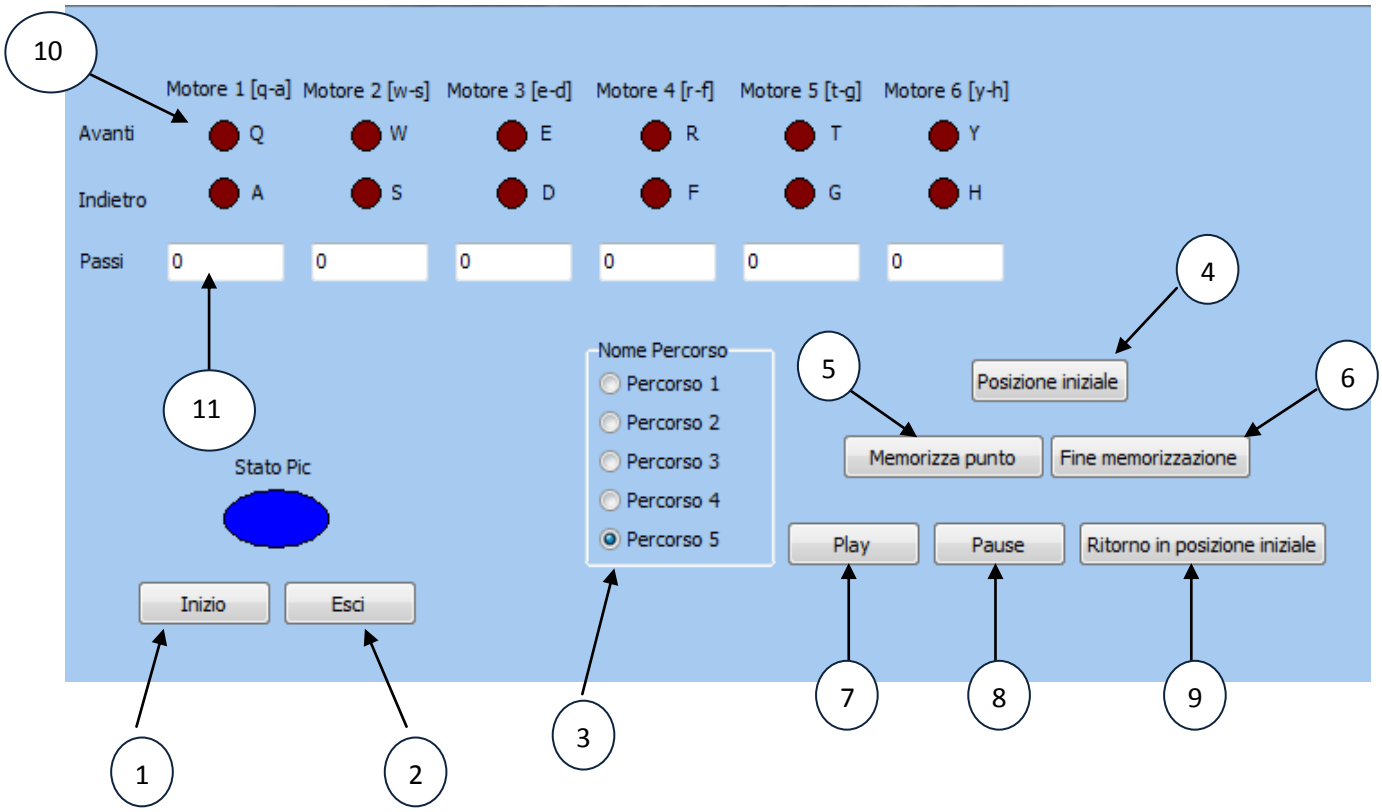
4) Connettere il cavo seriale dall'apposito connettore DB9 presente sulla scheda al PC.



5) Alimentare il MINI ROBOT agendo sull'interruttore posto sull'alimentatore.

6) Aprire l'interfaccia Delphi su PC e seguire le successive istruzioni.

-Utilizzo dell'interfaccia Delphi



1) Il pulsante "Inizio" permette l'avvio del programma e della comunicazione tra PC e microcontrollore;
2) Il pulsante "Esci" permette di uscire dal programma e di chiudere la comunicazione tre PC e microcontrollore;
3) La selezione del nome del percorso è necessaria per:
   -il nome del percorso con cui esso verrà salvato in modalità memorizzazione;
   -indicare quale percorso salvato si vuole riprodurre.
4) Per memorizzare un percorso è necessario portare il braccio in una posizione iniziale: se questo non viene fatto non è possibile accedere alla modalità di memorizzazione. La posizione iniziale è fondamentale per avere un punto di riferimento da cui far partire la riproduzione del percorso e per far tornare il braccio in una posizione nota al termine di essa.
5) Questo pulsante serve per memorizzare i vari punti del percorso: quando l'utente lo preme, vengono salvati il numero dei passi effettuati dai motori.
6) Il pulsante "Fine memorizzazione" serve a segnalare al sistema la fine della memorizzazione del percorso: quando esso viene premuto il braccio si riporta alla posizione iniziale.

7-8-9) Questi pulsanti sono necessari per la modalità riproduzione del percorso: in particolare "play" permette l'inizio della riproduzione del percorso selezionato (pnt 3), "pausa" permette la sospensione temporanea dell'esecuzione e "ritorno in posizione iniziale" consente la fine della riproduzione con il ritorno in posizione iniziale del braccio.

10) I led rossi segnalano quali motori sono attivati e in che verso nella modalità movimento libero.

11) Vengono stampati negli appositi indicatori i passi effettuati dai vari motori nella modalità memorizzazione del percorso.