Università degli studi di Padova

Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Elettrica

Corso di Robotica
Docente Prof. Emanuele Menegatti

Progetto Robot Hand V2

Marco Gottardo 347349/IT

Padova, A.A. 2007 - 2008

Indice

Introduzione

Con l'avvento dei moderni microcontrollori, caratterizzati da basso costo e alte performance, è possibile cimentarsi nel campo della robotica non solo a livello professionale ma anche a livello hobbistico.

La tesina si è sviluppata in tre fasi, nella prima ci si è dedicati alla progettazione e realizzazione dell'Hardware composto da una simbiosi di elettronica digitale, analogica e strutture meccaniche, nella seconda muniremo la protesi di apparati sensoriali in grado di fare reagire l'arto al calore e poter tarare la forza di presa sugli oggetti, mentre nella terza si intende fare uno studio di fattibilità, quindi teorico, dell'interfacciamento al sistema nervoso tramite l'acquisizione di segnali provenienti da delle sonde EEG.

Il target è di sviluppare un insieme di protesi e di esoscheletri servoassistiti di basso costo in grado di ridare abilità a pazienti affetti da diversi gradi di menomazioni o tipi di invalidità congenite o acquisite di tipo transitorio o permanete.

Benché nella presente trattazione si presenti una mano meccanica munita di polso, braccio e avambraccio, le tecniche esposte potranno essere prese a modello per lo sviluppo di protesi di diversa natura.

Ogni soluzione prevede una forte integrazione tra argomenti di meccanica, elettronica e informatica.

I pazienti soggetti a studio potranno essere muniti di esoscheletri con asservimenti in corrente continua pilotati con sistemi a microprocessore se la disabilità è solo motoria dovuta a lesioni nervose (l'arto è presente ma è inerte),

Componenti del progetto

La parte principale è costituita da uno chassy in lamierino di ottone, scelto per la facilità con cui si riesce a lavorare anche con scarsi mezzi, in effetti si può tagliare usando una semplice forbice da elettricisti, consente le saldature di piccoli giunti con stagno 40/60 ed è facilmente piegabile e forabile. Una volta ripiegata e munita di opportune nervature fornisce una soddisfacente rigidità e robustezza meccanica, almeno per quanto riguarda la realizzazione di questo prototipo.

La simulazione dei tendini è fatta con cordino di nailon di 1mm di sezione, mentre i principali gradi di libertà DOF (degree of freedom) sono realizzati con dei servomotori di coppia all'asse opportuna a seconda del DOF che dovranno gestire. Per la motorizzazione delle dita, ad esempio, risultano sufficienti dei mini servomotori per modellismo. Altri punti in cui la coppia resistente esterna risulta piuttosto elevata sono stati motorizzati con opportuni motoriduttori trasformati per l'occasione in servomotori di potenza.

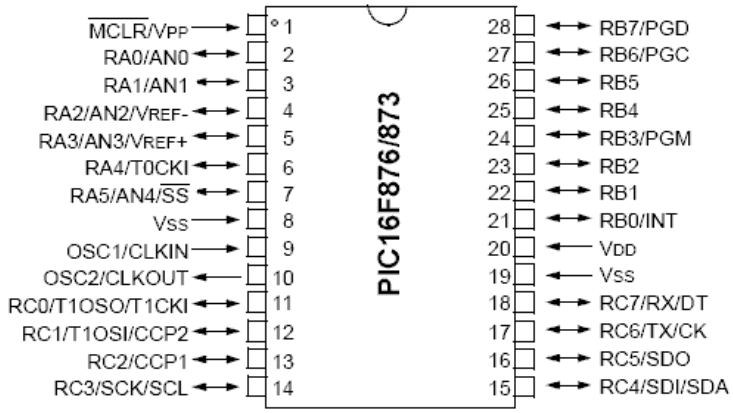
Il microcontrollore utilizzato

Il **PIC16F876** prodotto dalla MicroChip® appartiene alla famiglia dei microcontrollori, ovvero di quei dispositivi che integrano nella CPU un insieme completo e potente di dispositivi dedicati all'I|/O. Il nucleo, denominato **core,** è costituito da una CPU ad alte performance basata sulla tecnologia RISC (***R**educed **I**nstruction **S**et **C**omputing*), la cui programmazione richiede la conoscenza di soli 35 codici mnemonici. Quasi tutte le istruzioni posso essere eseguite in un solo ciclo macchina ad eccezione di quelle di salto "branche" che ne richiedo almeno 2. Il clock di funzionamento può essere impostato a media velocità, 4Mhz, oppure ad alta velocità, 20Mhz a seconda del quarzo che si intende montare e alla velocità di esecuzione richiesta dall'applicazione. Con il clock fissato a 20Mhz la velocità di esecuzione per ogni ciclo macchina scende a 200 ns.

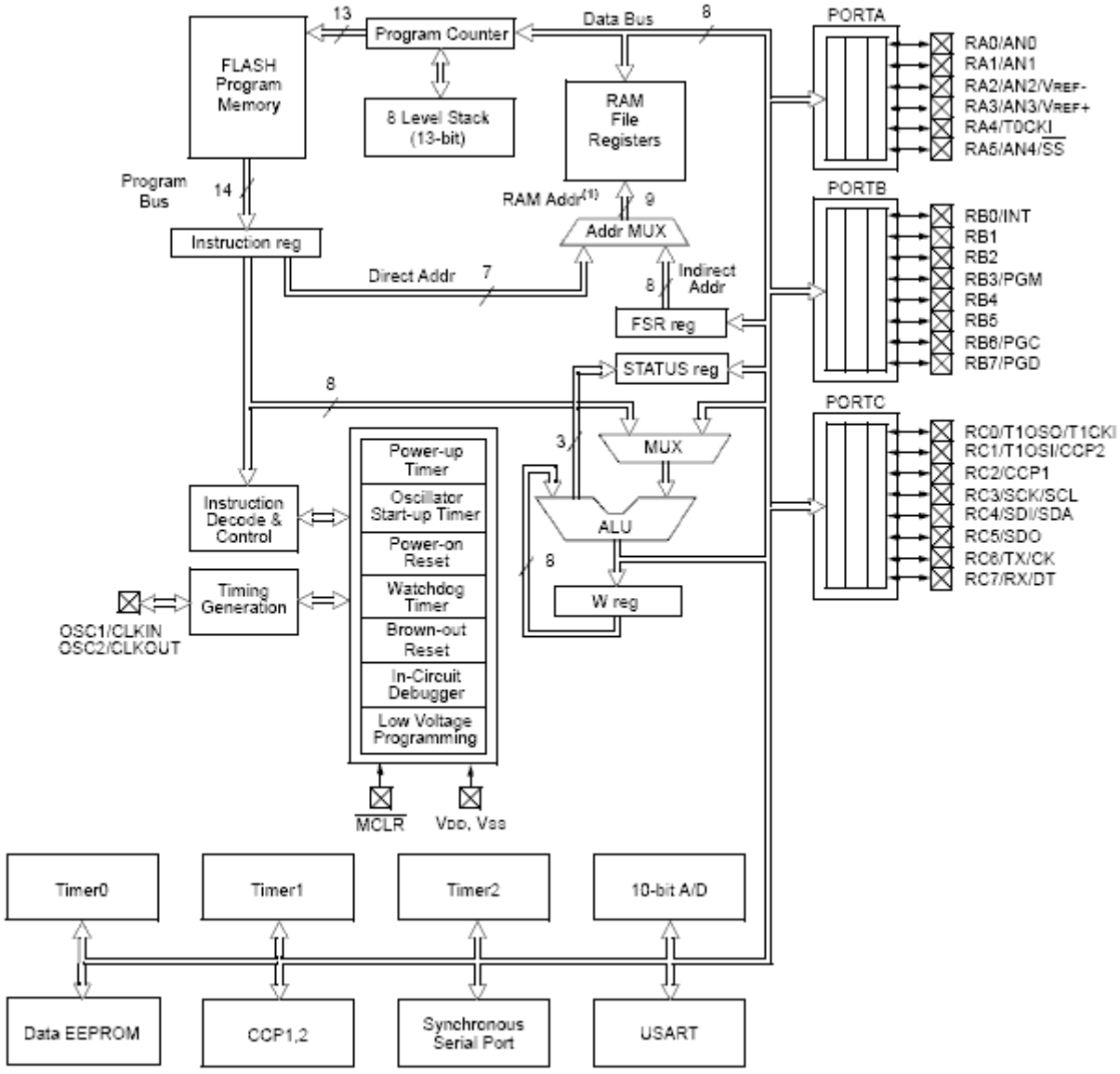
Il dispositivo è munito delle seguenti aree di memoria:

- Memoria FLASH per programma estesa 8K x 14 words.
- Memoria RAM per i dati estesa 368 x 8 bytes
- Memoria EEPROM sempre per i dati estesa 256 x 8 bytes

Il pinout risulta compatibile con i dispositivi PIC16C73B/74B/76/77 appartenenti alla stessa famiglia.
IL programma può rispondere da segnali di interrupt provenienti da oltre 14 fonti vettorizzate, e alcuni di questi segnali di interrupt sono utilizzati nella comunicazione seriale del presente progetto.
In caso di inutilizzo del sistema il processore può essere posto in una sorta di stato di standby chiamato SLEEP MODE che ne riduce notevolmente i consumi.
La tensione di alimentazione risulta valida nel range 2.0V fino a 5.5V, livelli inferiori comportano il reset del processore mentre superiori ne comportano la distruzione.
Importantissimo è conoscere che la massima corrente per punto di I/O, sia come sorgente che come pozzo, è di 25mA.
La potenza dissipata dal dispositivo è molto limitata, con clock di 4Mhz e alimentazione di 3,4 Volt vengono assorbiti meno di 0,6mA, ma riducendo ulteriormente il clock, per quelle applicazioni dove non è richiesta la velocità, ad esempio a 32KHz e alimentazione a 3Volt verranno assorbiti solo 20uA che scendono a meno di 1uA se il dispositivo e posto in standby.



Le periferiche integrate
Il PIC 16F876 mette a disposizione un elevato numero di punti di I/O in grado di soddisfare moltissime applicazioni in ambito di automazione e robotica. Ad una buona parte dei suoi pin è abbinata una doppia o addirittura tripla funzione, si vedano ad esempio i pin 11 e 12.
Sono disponibili 3 timer commutabili a uso contatore con le seguenti caratteristiche tecniche:
• Timer0: 8-bit timer/contatore con prescaler a 8-bit.
• Timer1: 16-bit timer/contatore con prescaler, che può essere incrementato durante la funzione SLEEP usando un clock esterno creato da un apposito cristallo.
• Timer2: 8-bit timer/contatore con registro ad anello a 8-bit, prescaler e postscaler
La comunicazione seriale avviene ai pin 17 (TX) e 18 (RX) tramite un'iterfaccia integrata di tipo USART Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) con rilevamento di indirizzanti a 9 bit.
La porta A, accessibile dal pin 2 al pin 7, ovvero per 5 bit, può essere commutata alla funzionalità di ingresso analogico con risoluzione a 10 bit.
La comunicazione è anche possibile in modalità I^2C con la modalità master/slave, oppure in modalità SSP Synchronous Serial Port con SPI (Master mode).
Nella successiva immagine è riportato lo schema a blocchi dell'architettura interna del PIC 16F876.

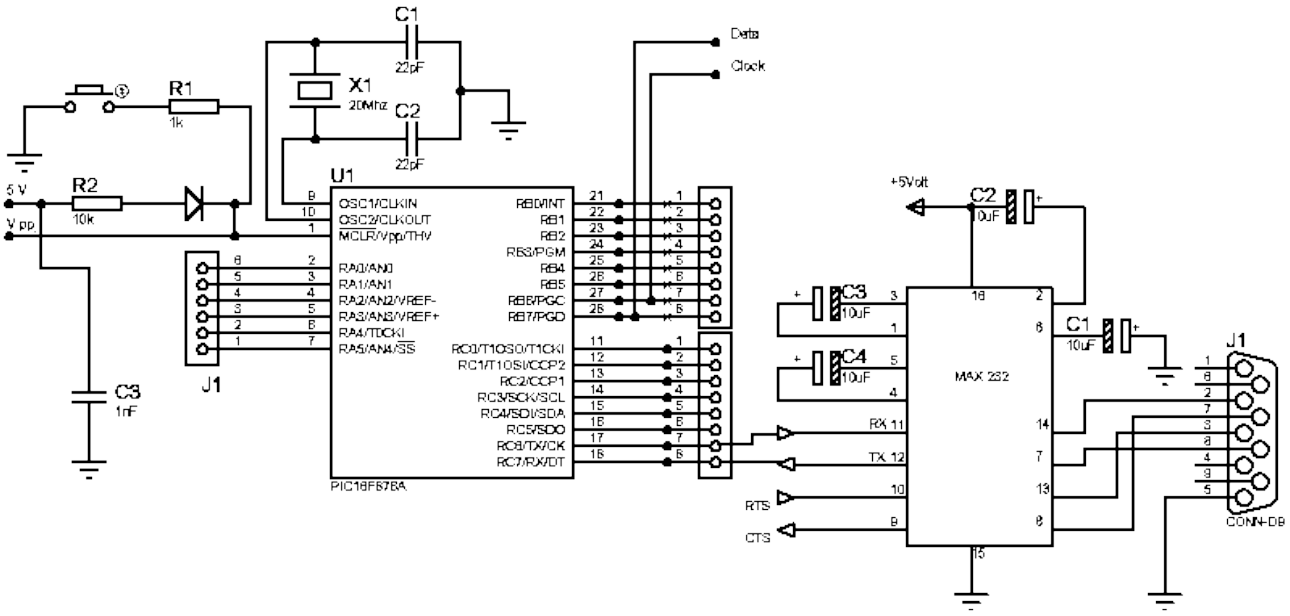


Scheda con microcontrollore

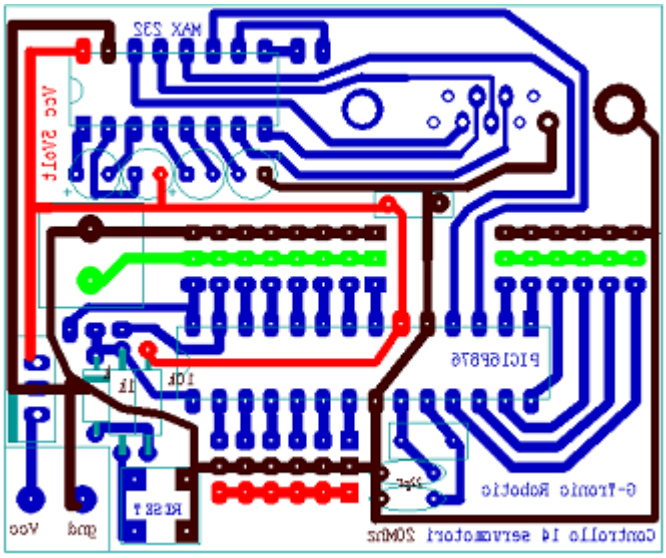
La scheda a microprocessore rappresenta il corpo principale del progetto. Essa è molto compatta ed in sostanza alloggia solo il microprocessore, un chip dedicato alla comunicazione seriale il famoso MAX232, un robusto connettore di tipo Canon DB9 a vaschetta femmina, il quarzo e i relativi condensatori ceramici per realizzare l'oscillatore, il pulsante di reset, e pochi altri componenti

Il diodo 1N4148 connesso con il catodo al pin 1 serve ad impedire dei flussi di corrente inversa nel caso si volesse procedere alla programmazione incircuit.

Per non appesantire lo schema elettrico non sono stati riportati tutti gli streep che sono invece presenti sul PCB.

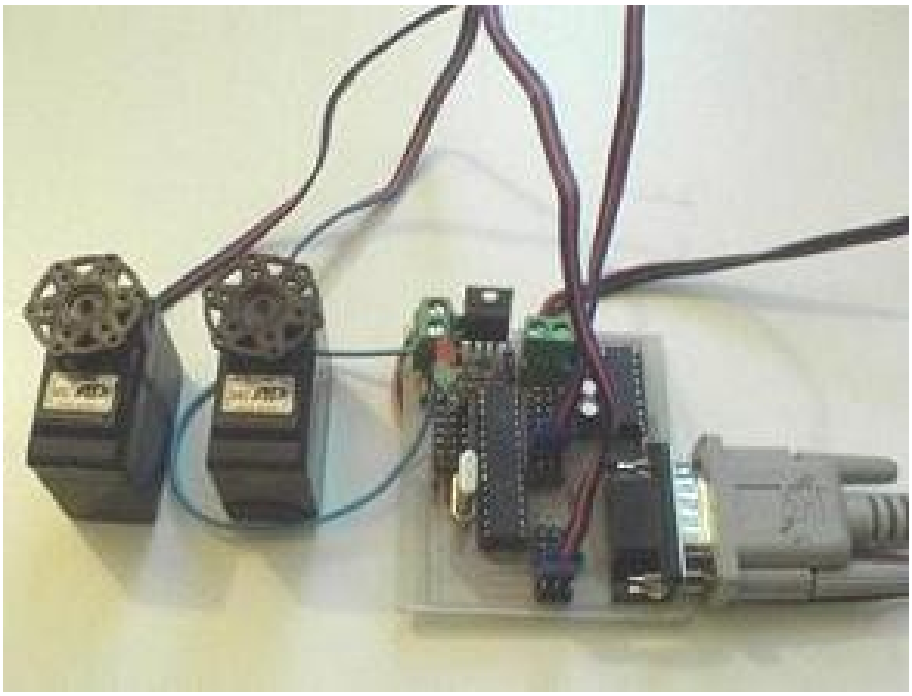


Le alimentazioni della parte logica e della parte di potenza sono tenute separate per impedire che eventuali fluttuazioni o bruschi cali di tensione possano resettare il micro. Nel PCB le piste in rosso portano l'alimentazione +Vcc a 5 volt, stabilizzati grazie alla presenza del regolatore di tensione uA7805, quindi nel morsetto corrispondente potremo portare una tensione massima di 36 volt che verrà portata a +5.



Come si vede dal layout del PCB alcune parti in tensione risultano sezionate, nel caso si volesse usare la scheda di controllo nel pieno delle sue potenzialità sarà necessario eseguire qualche ponticello. Ponticellando la pista verde di sinistra con la pista verde di destra si alimentano tutti e quattordici i servomotori alla tensione presente al morsetto, tipicamente 5 Volt ricavati dalla fonte adibita alla potenza. La tensione presente su questo conduttore non da origine a correnti condivise con il circuito di logica perché direttamente gestite dai circuiti di comando e potenza dei vari servomotori di cui ne costituisce l'effettiva alimentazione.
La pista in rosso visibile sulla parte bassa del PCB è invece l'alimentazione dei sensori analogici, tipicamente +5V e quindi cortocircuitabile con un punto qualsiasi dell'alimentazione a valle del regolatore di tensione. Esistono sensori di campo che vanno alimentati con tensioni diverse dalla

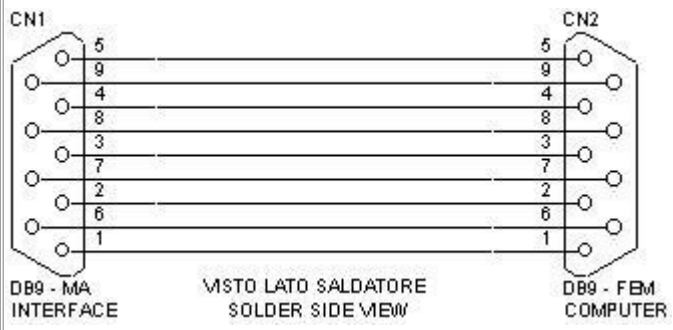
standard TTL a 5V, ad esempio 9 o 12 o addirittura 24 volt, che in ogni caso fornisco in uscita una traduzione analogica a 10 oppure 5 volt, è per la gestione di questi casi che si è preferito tenere separata quella pista dal resto del circuito.
Nel presente progetto verranno collegati 5 potenziometri, che nascosti nel dorso del guanto sinistro daranno in uscita un segnale proporzionale in tensione alle posizione delle dita. Questi segnali saranno usati come mando di posizionamento per le dita della protesi robot della mano destra, ovvero per posizionare i 5 servo motori corrispondente collegati ai tendini delle rispettiva dita.



Nell'immagine sono collegati solo due servomotori dei 14 pilotabili, uno si trova alla port B e uno alla port C del micropic. Anche collegando tutti gli attuatori il cablaggio rimane bene ordinato dato che i cavi si collegano alla scheda tutti dalla medesima parte del microprocessore. L'unica cosa da conoscere è che il motore A si troverà al pin 21 e quindi quello più basso della porta B che nel lato sinistro del PIC risulta essere pressappoco n posizione centrale, non facciamo quindi trarre in inganno dalla posizione "in alto" assunta dal pin 21 nello schema elettrico.
Sempre dalla foto possiamo notare la presenza di un diodo LED rosso, questi è stato installato per eseguire un test di funzionamento runtime. Il programma di controllo dei 14 servomotori contiene alcune righe di codice che commutano inizialmente la porta A come uscita digitale, quindi in questa fase non sono attivi gli ingressi analogici. Contemporaneamente si attiva l'uscita RA0, a cui e connesso il LED e ci si mette in ascolto della porta seriale. Ciò che ci si aspetta è che all'arrivo della stringa **@P123 ret** il LED si spegne. tramite la stringa **@Q123 ret** il LED si riaccende.
Per come inviare la stringa tramite seriale si rimanda alla lettura del capitolo "comunicazione seriale" paragrafo "uso di hyperterminal".
È da sottolineare che solo se il PIC si è avviato regolarmente (e quindi la scheda non ha problemi hardware e il 16F876 sta facendo girare il suo programma correttamente) il LED risulta acceso.

La comunicazione seriale
Il protocollo scelto per la comunicazione tra la scheda di controllo dei servomotori e il PC di comando manuale è il EIA RS232C, ovvero lo standard seriale più vecchio ma nello stesso tempo più collaudato e semplice da usare.

Cavo seriale
Per comunicare con la scheda di controllo dei servomotori si dovrà costruire un **cavo di prolunga seriale** realizzato seguendo lo schema sottoriportato:



9 poli maschio (all'interfaccia PIC)	9 poli femmina (al PC)
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Non si dovranno usare i cavi chiamati "null modem" perchè non funzioneranno. In realtà sono necessari solo i conduttori 2 (TX),3(RX) e 5(massa).

Corrispondenza dei pin connettore 25 poli - 9 poli
Alcuni PC dispongono di un connettore a 25 poli femmina anzichè a 9 poli maschio per la porta seriale. Se non non è disponibile un adattatore idoneo possiamo costruirlo seguendo le corrispondenze riportate nella sottostante tabella.

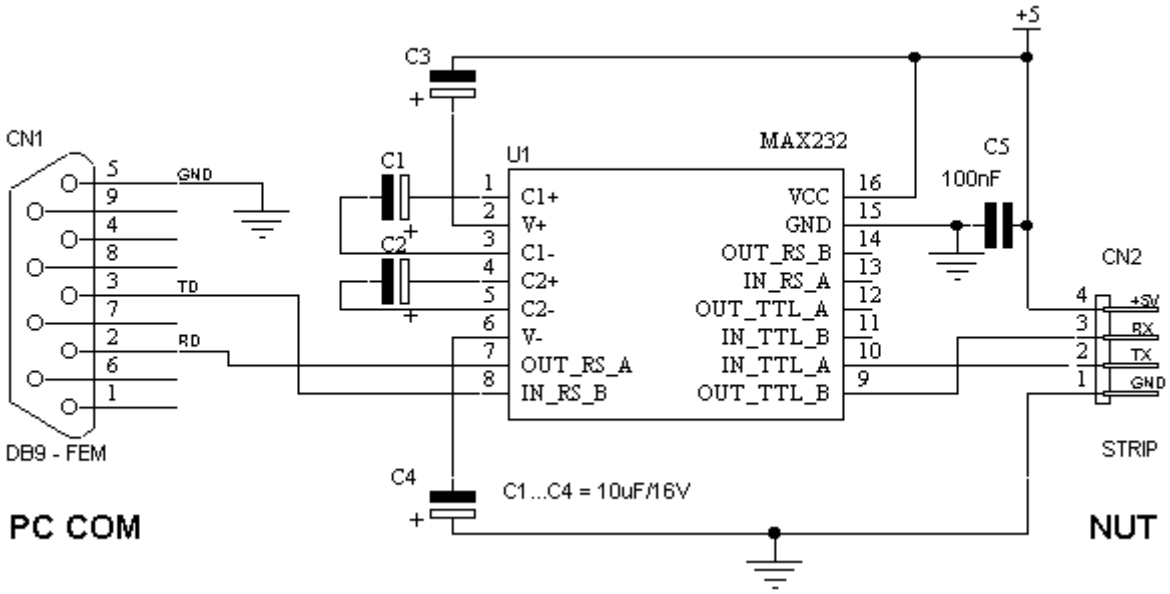
25 poli (femmina sul PC)	9 poli (maschio sul PC)	nome linea RS232
2	3	TD
3	2	RD
4	7	RTS
5	8	CTS
6	6	DSR
7	5	GND
8	1	DCD
20	4	DTR
22	9	RI

Per collegarsi al PC, servno quindi un connettore a vaschetta cannon DB25 poli maschio ed un cannon DB9 poli femmina.

Nella figura è riportata la piedinatura del conettore cannon DB9 femmina, dovremo fare attenzione alla piedinatura del conettore maschio perché risulta essere simmetrica.

Interfaccia adattamento livelli TTL con MAX232

Questa interfaccia utilizza un integrato apposito e fornisce livelli di uscita **standard** adatti per ogni tipo di PC. Trattandosi di un integrato specifico, ha anche la migliore reiezione ai distrurbi e la protezione contro scariche e corto circuiti. Per contro è una soluzione più costosa e ingombrante delle soluzioni che utilizzano i transistor o semplici porte logiche.

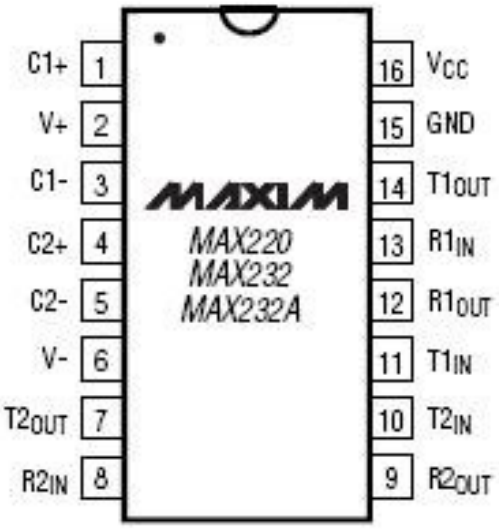


La scheda di controllo dei servomotori integra già questa sezione come è evidente dallo schema elettrico riportato nel capitolo precedente.

Dato che il canale seriale RS-232 deve essere discretamente immune ai disturbi, i livelli di tensione adoperati tra il DTE e il DCE, ovvero lungo il cavo, sono diversi da quelli di tipo TTL o CMOS. Risulta necessario traslare le tensioni per adattarle ai circuiti elettronici. Per maggiori immunità ai disturbi esiste una variante dell'interfaccia RS-232, che prende nome di RS-485, dove affianco ad ogni segnale, viene riportato anche il suo complementare, esso costituisce il protocollo seriale industriale.
Nella tabella si confrontano i livelli di tensione usati dallo standard TTL-CMOS e RS-232.

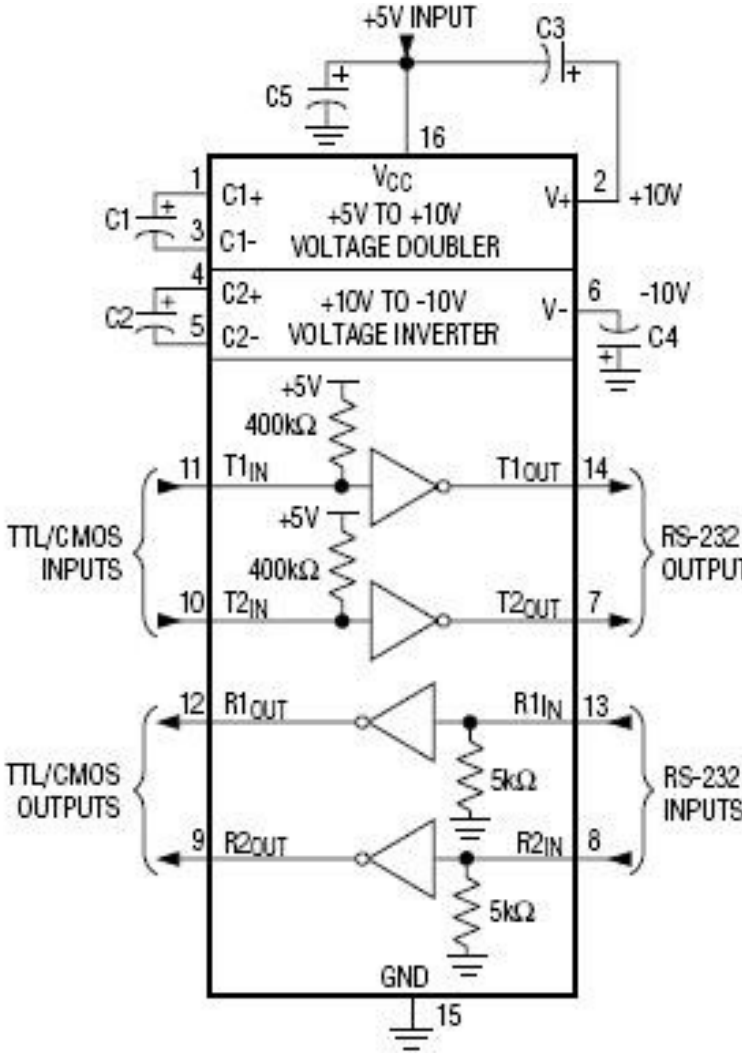
TTL - CMOS	RS-232
livello logico "0"	+12 Volt (+6V...+15V)
livello logico "1"	-12 Volt (-0V...-15V)

La comparazione mette in evidenza che le tensioni sono diverse e ben più alte, al fine di garantire una buon immunità ai disturbi, ed inoltre risultano invertite di polarità.
Il circuito intergrato MAX232 è quindi un traslatore ed invertitore di tensione.
La velocità di comunicazione trasmissione dell'integrato può spingersi fino anche ad 1Mbit/s per una lunghezza del cavo di circa 11 metri.



DIP/SO

CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1



Il MAX232 dispone di 2 canali per la comunicazione RS-232 bidirezionale, come è possibile vedere osservando la piedinatura. Inoltre ha il pregio di richiedere solo i +5Vcc anche per supportare, in trasmissione, lo standard RS-232. Ciò è possibile, grazie a due stadi convertitori DC-DC, ovvero un elevatore di tensione a capacità, da +5Vcc a +12Vcc; cui segue uno stadio invertitore di polarità, sempre a capacità, da +12Vcc a -12Vcc. Queste tensioni, poi sono pure rese disponili per altri impieghi, ai seguenti piedini: pin 2: +12Vcc, pin 6: -12Vcc. E i condensatori presenti collegati al circuito integrato permettono il regolare funzionamento degli stadi convertitori DC-DC, come è possibile vedere osservando lo schema applicativo, si nota ad esempio che l'elettrolitico al piedino 6 ha il positivo a massa dovendo gestire una tensione negativa internamente prodotta.

Piedinatura del MAX232.
Può essere utile spiegare brevemente il significato della piedinatura del chip.

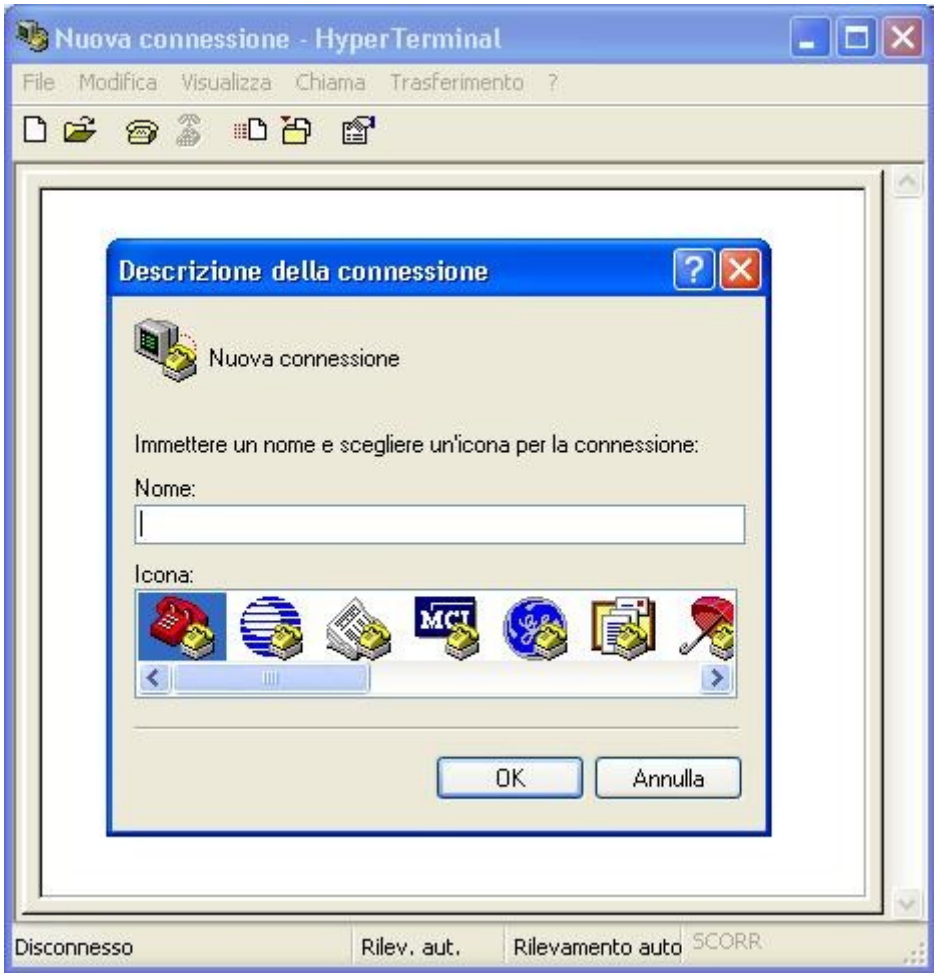
PIN	DESCRIZIONE
1	Polo positivo del condensatore per il convertitore a pompa di carica per generare la tensione positiva
2	Uscita tensione positiva dello stadio di duplicazione della tensione di alimentazione
3	Polo negativo del condensatore per il convertitore a pompa di carica per generare la tensione positiva
4	Polo positivo del condensatore per il convertitore a pompa di carica per generare la tensione negativa
5	Polo negativo del condensatore per il convertitore a pompa di carica per generare la tensione negativa
6	Uscita tensione negativa dello stadio dell'inversione della tensione di alimentazione
7	Uscita RS232 canale 2
8	Ingresso RS232 canale 2
9	Uscita RS232 canale 2
10	Ingresso TTL/CMOS canale 2
11	Ingresso TTL/CMOS canale 1
12	Uscita TTL/CMOS canale 1
13	Ingresso RS232 canale 1
14	Segnale di massa
15	Uscita RS232 canale 1
16	Positivo dell'alimentazione

Uso di HyperTerminal
Hyperterminal è una console di "echo" di ciò che avviene nelle porte di comunicazione del PC.
Principalmente è un software di comunicazione utilizzato per connettersi ad altri computer ad esempio tramite modem seriale, oppure con collegamenti RS-232 con i quali effettuare il telnet. Al fine di utilizzare HyperTerminal, l'utente dovrà conoscere i dettagli circa il computer a cui desidera connettersi, ad esempio il numero telefonico di chiamata se via banda fonica o l'indirizzo IP se via canale dati.

Ad HyperTerminal di windows si può accedere nella seguente maniera:

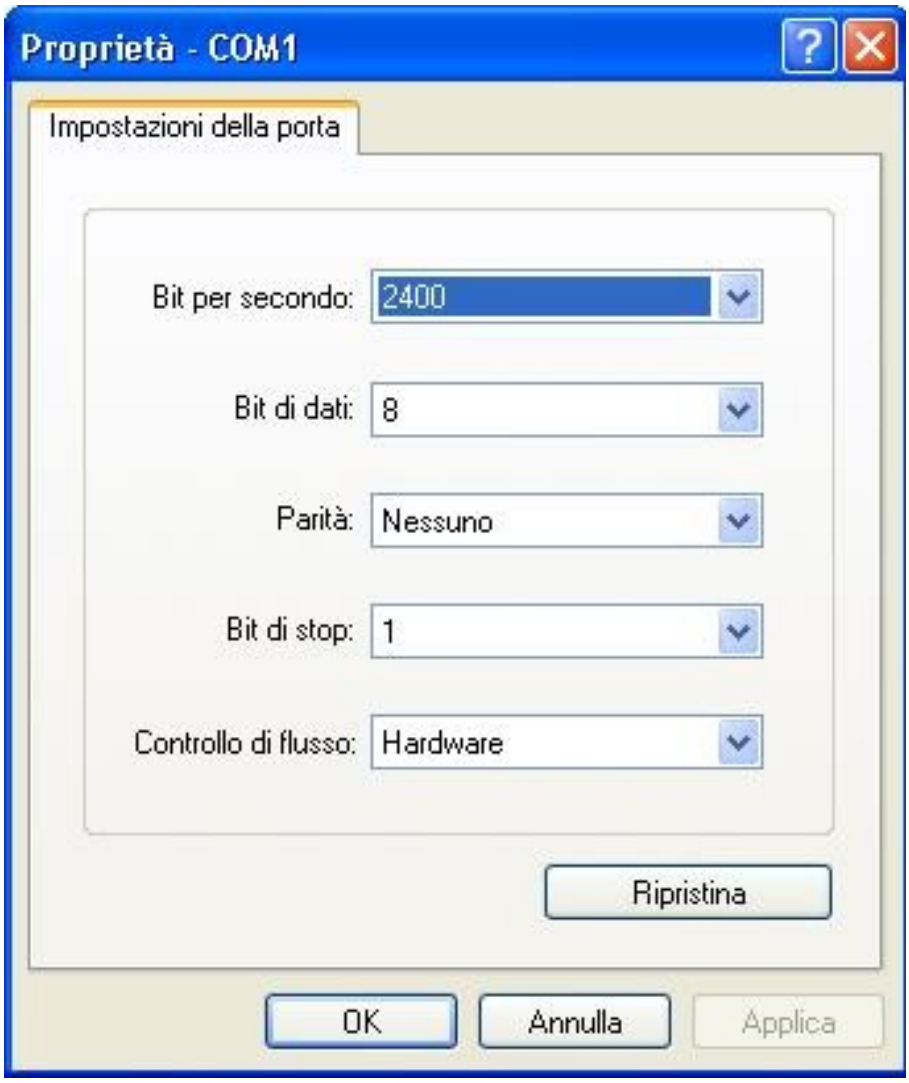
- Clic su **Start** della barra delle applicazioni del desktop
- Accedere al menù **tutti i programmi**
- Ciccare su **Accessori**
- Ciccare su **comunicazioni**
- **HyperTerminal** clic su Avanti per avviare il programma

Al primo avvio dell'applicazione verrà chiesto di configurare la chiamata identificando l'utente sia locale "informazioni del tipo dove siamo" che remoto, ovvero il numero a cui ci vogliamo allacciare. Dovremo fornire informazioni del tipo nazione da chiamare, prefisso, il modo in cui si desidera collegare (modem / TCP), inoltre dovremo indicare se il telefono da cui chiamiamo utilizza la selezione a toni o genera il numero ad impulsi, ovvero nella vecchia maniera. Quindi fare clic su OK.
Giunti alla schermata "Nuova connessione" si apre una finestra, che permette di scegliere un nome e un'icona per la nuova sessione di HyperTerminal. Effettuati questi passaggi il sistema di comunicazione è configurato per utilizzare le linee del telefono, difatti agendo sulla icona con quella forma, sulla barra degli strumenti HyperTerminal si potrà collegare o scollegare una chiamata a un altro computer.
HyperTerminal può essere utilizzato per monitorare lo stato del modem collegato al PC anche se alla data odierna è una tecnologia obsoleta. I dettagli di ogni connessione effettuata utilizzando HyperTerminal sono registrati in un file di log. La lettura di questo file può essere un modo per risolvere questioni di connessione modem.



Se la comunicazione è impostata solo er avere un eco del buffer della porta seriale è importante selezionare il giusto formato della stringa RS232 inviata sulle linee.
Per i dispositivi presentati in questa tesina impostare un boude rate di 9600, 8 bit di dati, nessuna parità e nessun controllo del flusso.
Impostazioni diverse potrebbero comportare che la comunicazione con la scheda demoboard non avviene.
Eventuali impostazioni diverse dovranno essere impostate nella routine in C da inserire nel PIC.

Nulla vieta di impostare una velocità di 19200 bit per secondo ma ovviamente deve esserci concordanza tra la micro programmazione del processore che imposta la porta a livello hardware e il protocollo software che qui vediamo esemplificato con HyperTerminal.
La figura sottostante mostra la configurazione a velocità minima con controllo di flusso maggiore.
Questa è incompatibile con il programma inserito nel PIC ed è a puro titolo di esempio.
Vedi le impostazioni corrette in neretto e nella pagina precedente.



Per istaurare un coretto flusso di comunicazione è ovviamente fondamentale aprire la porta coretta, ad esempio la COM 1 come in figura o altra a cui è connesso il DTE con cui comunicare.
I test della comunicazione potranno avvenire ad esempio inviando gli elementi della tabella ascii.
Per comodità riportiamo nelle pagine successive le tabelle del codiceASCII, standard ed esteso.

DEC	HEX	OCT	HTML	CHR
0	0	000	-	NUL
1	1	001	-	SOH
2	2	002	-	STX
3	3	003	-	ETX
4	4	004	-	EOT
5	5	005	-	ENQ
6	6	006	-	ACK
7	7	007	-	BEL
8	8	010	-	BS
9	9	011	-	TAB
10	A	012	-	LF
11	B	013	-	VT
12	C	014	-	FF
13	D	015	-	CR
14	E	016	-	SO
15	F	017	-	SI
16	10	020	-	DLE
17	11	021	-	DC1
18	12	022	-	DC2
19	13	023	-	DC3
20	14	024	-	DC4
21	15	025	-	NAK
22	16	026	-	SYN
23	17	027	-	ETB
24	18	030	-	CAN
25	19	031	-	EM
26	1A	032	-	SUB
27	1B	033	-	ESC
28	1C	034	-	FS
29	1D	035	-	GS
30	1E	036	-	RS
31	1F	037	-	US
32	20	040		Space

DEC	HEX	OCT	HTML	CHR
33	21	041	&#33;	!
34	22	042	&#34;	"
35	23	043	&#35;	#
36	24	044	&#36;	$
37	25	045	&#37;	%
38	26	046	&#38;	&
39	27	047	&#39;	'
40	28	050	&#40;	(
41	29	051	&#41;	)
42	2A	052	&#42;	*
43	2B	053	&#43;	+
44	2C	054	&#44;	,
45	2D	055	&#45;	-
46	2E	056	&#46;	.
47	2F	057	&#47;	/
48	30	060	&#48;	0
49	31	061	&#49;	1
50	32	062	&#50;	2
51	33	063	&#51;	3
52	34	064	&#52;	4
53	35	065	&#53;	5
54	36	066	&#54;	6
55	37	067	&#55;	7
56	38	070	&#56;	8
57	39	071	&#57;	9
58	3A	072	&#58;	:
59	3B	073	&#59;	;
60	3C	074	&#60;	<
61	3D	075	&#61;	=
62	3E	076	&#62;	>
63	3F	077	&#63;	?
95	5F	137	&#95;	_
96	60	140	&#96;	`

DEC	HEX	OCT	HTML	CHR
97	61	141	&#97;	a
98	62	142	&#98;	b
99	63	143	&#99;	c
100	64	144	&#100;	d
101	65	145	&#101;	e
102	66	146	&#102;	f
103	67	147	&#103;	g
104	68	150	&#104;	h
105	69	151	&#105;	i
106	6A	152	&#106;	j
107	6B	153	&#107;	k
108	6C	154	&#108;	l
109	6D	155	&#109;	m
110	6E	156	&#110;	n
111	6F	157	&#111;	o
112	70	160	&#112;	p
113	71	161	&#113;	q
114	72	162	&#114;	r
115	73	163	&#115;	s
116	74	164	&#116;	t
117	75	165	&#117;	u
118	76	166	&#118;	v
119	77	167	&#119;	w
120	78	170	&#120;	x
121	79	171	&#121;	y
122	7A	172	&#122;	z
123	7B	173	&#123;	{
124	7C	174	&#124;	|
125	7D	175	&#125;	}
126	7E	176	&#126;	~
127	7F	177	&#127;	DEL

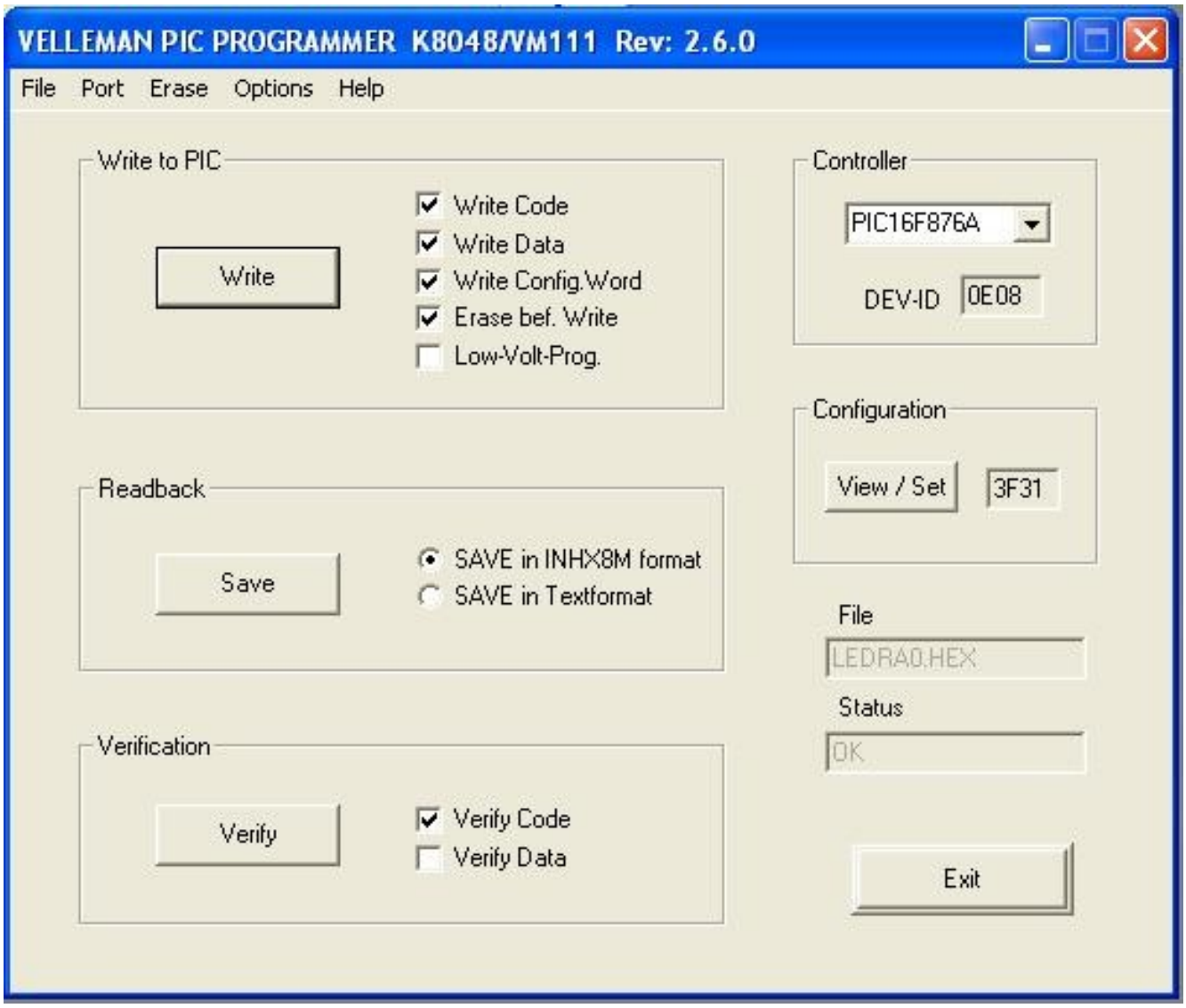
Nell'immagine successiva riportiamo il codice ASCII esteso

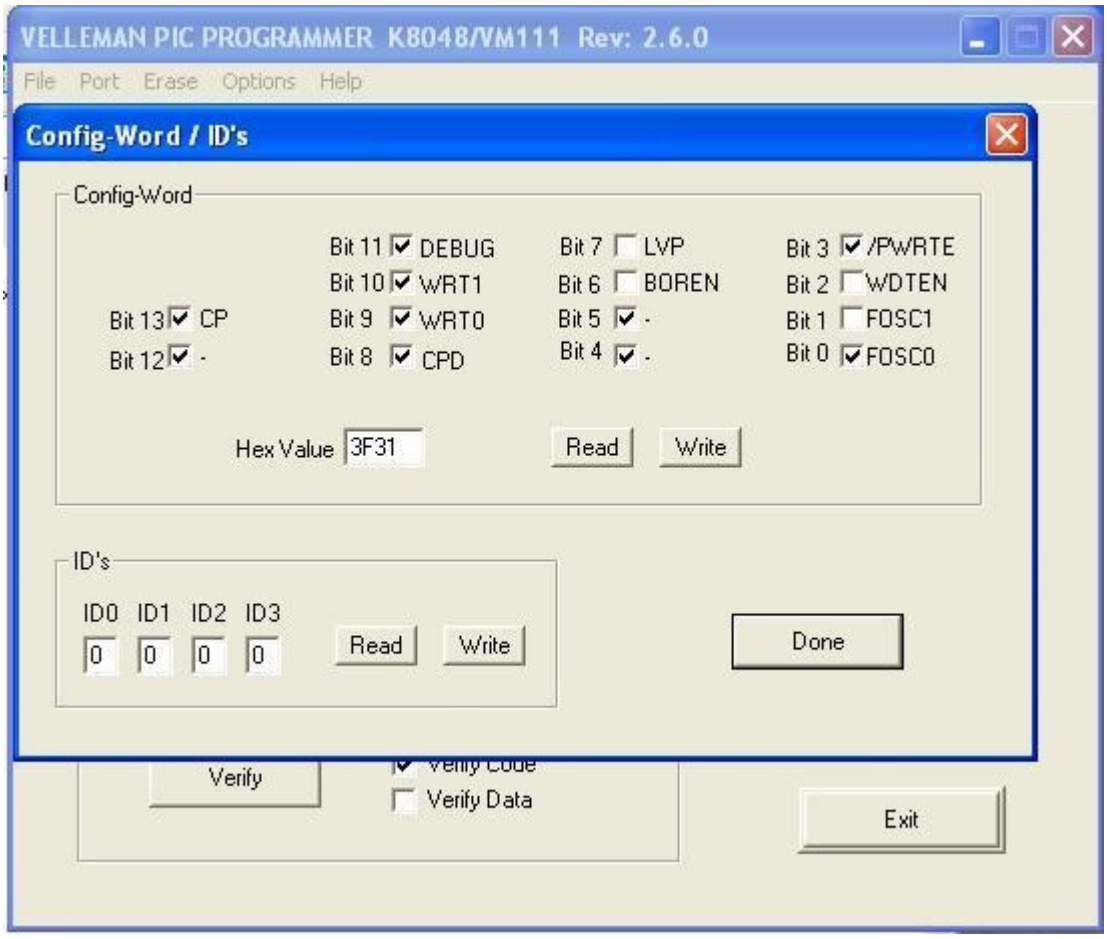
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	⌂

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	⌂

Download del programma nella memoria del PIC

Ogni programmer disponibile in commercio è fornito con un software in grado di comunicare con il PIC al fine di riversarci all'interno il programma .HEX che ne costituisce il microcodice.
Un esempio è quello prodotto dalla Velleman.
Fondamentale è l'impostzione dei Flag, detti fuse che predispongo l'abilitazione o meno di alcune funzioni specialicome il timer di watch dog, la programmazione in circuit, la frequenza del clock ecc.
I Flag sono raccolti in un'unica word che per i programmi presenti in questa tesina va impostata come in figura al valore esadecimale **3F31**
In bibliografia i flag in questione sono spesso denominati **FUSES.**





Impostazione standard dei fuses per i programmi contenuti in questa tesina.

La programmazione ICSP

In questo paragrafo sarà necessario distinguere tra bootloader, ovvero un codice .hex da inserire nel PIC, e downloader, ovvero un programma da lanciare sul PC che consente il caricamento di un programma esecutivo .hex nel Pic una volta che in esso sia preesistente ovvero precaricato il bootloader.

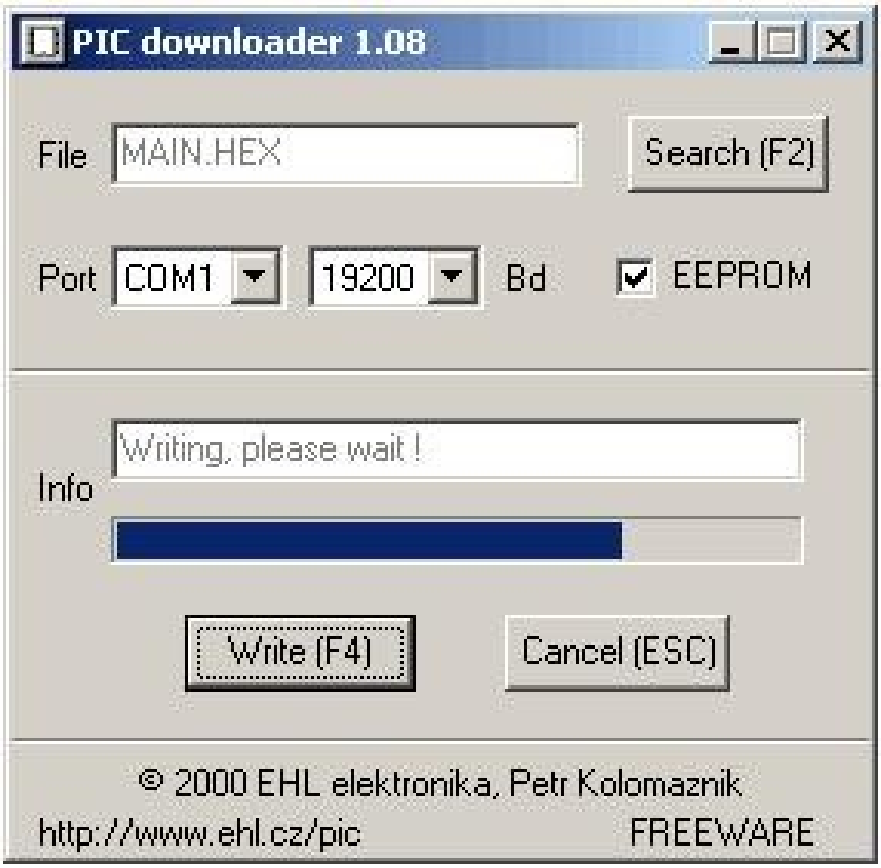
ICSP significa In-Circuit Serial Programming e può essere eseguita usando solo due pins.

Alcuni PIC posso essere programmati in maniera ICSP tramite una tensione singola a 5V, altri invece richiedono che questa sia diversa da quella di alimentazione della logica e valga 13,8 V.

Il bootloader è un programma usato per scaricare rapidamente un nuovo programma all'interno del micro PIC. Una volta impostato il bootloader il trasferimento del fil .HEX avviene in pochi secondi con la pressione di un singolo tasto.

La manovra detta Bootloading viene eseguita facilmente e direttamente in-circuit, ovvero senza rimuovere il micro dalla scheda finale.

Per l'uso basilare del bootloader non è necessario neppure modificare la scheda elettronica preesistente.



Come usare il bootloader

L'itilizzo è immediato e comporta solo pochissime preparazioni preliminari. La scheda di controllo servomotori progettata per questa applicazione dispone già dei terminali di allacciamento al cavo ICSP. Ovviamente il Pic micro 16F876 oppure il 16F876A dovrà essere presente sullo zoccolo e normalmente alimentato.
Inseriamo il cavo RS232 nella porta del PC e dall'altro lato colleghiamo i corrispondenti 4 file di cui due dedicati all'alimentazione di programmazione e riferimento di massa.
IL programma da inserire nel PIC dovrà riservare la memoria alta, ovvero i primi 255 byte per l'allocazione del bootloader. Per riservare questa memoria basta aggiungere una riga di codice al file sorgente.
La manovra di download inizia nel momento in cui clicchiamo Write sulla maschera del downloader seguita da un reset hardware della scheda target.
Dopo il reset il bootloader rimane attivo per circa 0,2 secondi in modo da consentire l'uploading del nuovo codice, durante questo tempo la porta com risulterà impegnata. Essa sarà di nuovo disponibile dopo 0,2 sec dal rilascio del pulsante di reset oppure dopo il trasferimento completo del nuovo codice.
Il bootloader deve però essere caricato nella memoria del micro Pic usando un programmer, sfortunatamente questo passo è obbligato, dato che ovviamente il bootloader non può caricare se stesso, quindi se non si possiede un programmatore bisognerà rivolgersi a qualcuno che lo ha disponibile. Una volta eseguita questa operazione si diventa indipendenti perché il programma può essere variato all'interno del target usando il downloader.

Caratteristiche del bootloader

l bootloader qui proposto è compatibile con tutta la serie di PIC16F87x. In modo particolare è stato testato con I PIC 16F870/1/3/4/6/7 e il 16F876, 16F876A, 16F877, 16F877A. Riporto di seguito un elenco sintetico delle caratteristiche tecniche:

- Lunghezza 255 istruzioni allocate nella memoria alta.
- Necessità di soli due fili per la comunicazione - TX e RX. Questo lo migliora rispetto ad altri bootloaders che invece necessitano di 5 conduttori.
- Funziona correttamente per quarzi a 3.6864, 4,16 e 20Mhz.
- Downloads fino a 19200bps. Quindi un programma completamente è trasferito in pochi secondi.
- Può scrivere direttamente sulla EEPROM.
- Supporta ogni tipo di Hex file, inclusi quelli provenienti da standard assemblers, compilatori Basic, C e Pascal.
- Compatibile con ogni tipo di circuito – non è necessario il pin di trigger o settare jumpers per attivare il bootloader.
- Viene fornito il codice sorgente scritto da Petr Kolomaznik. vedere bootloader page.

Collegamento fisico del bootloader

IL collegamento necessita di un adattatore che esegua la connessione della porta COM del computer con il Micro PIC. Un adattatore seriale ha il compito di convertire lo standard RS232 nei livelli TTL, -13V in +5V e +13V in 0V. il PIC si interfaccia correttamente con questi livelli direttamente nella sua porta seriale hardware integrata.

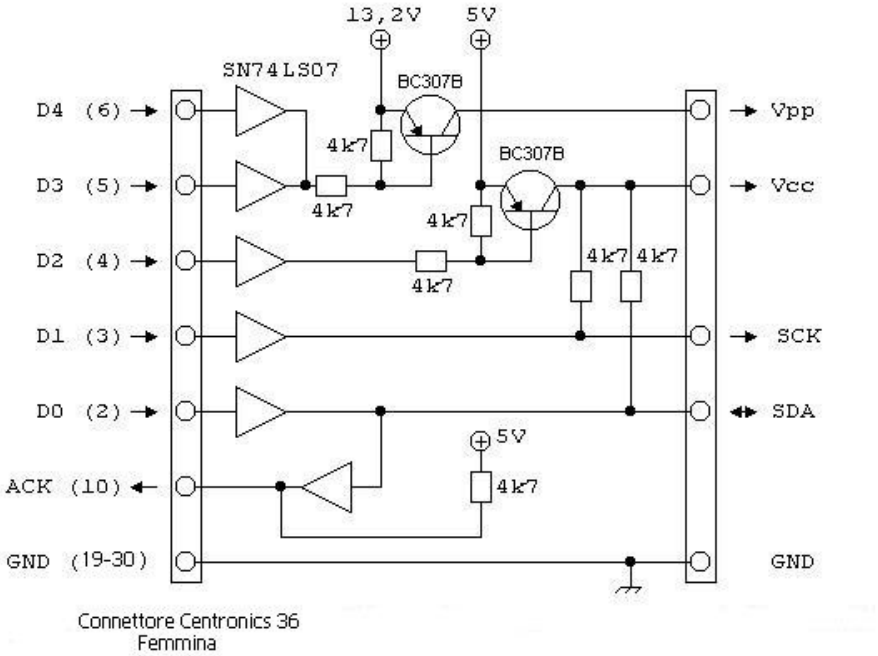
La costruzione dell'adattatore è delegata all'utente.

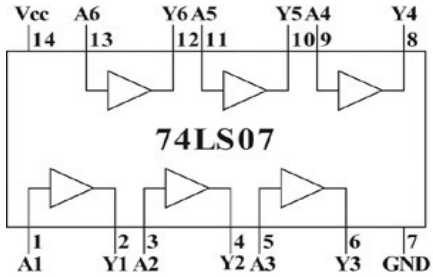
Lo schema per PIC >> RS232 comms è pubblicata in PICLIST.

Attenzione, quello di cui si necessita è di una conversione seriale compatibile con la periferica hardware UART presente nel PIC, la quale richiede una forma di adattamento dei livelli di tensione e inversione degli stessi. Non si può fare con delle semplici resistenze ma si ottiene facilmente usando un circuito basato sul circuito integrato MAX232. vedere PICLIST RS232 conversion circuits. Lo schema in questione è già integrato nella scheda di controllo servomotori presentato nella seguente tesina, quindi la connessione potrà essere diretta tramite la porta seriale munita di connettore DB9 femmina onboard.

Programmer per PIC 16F876A

Per caricare su un PIC nuovo il Bootloader, è necessario un dispositivo hardware chiamato programmatore.
In commercio è possibile reperire numerosi programmatori adatti al PIC16F876A e 877A, con costi spesso significativi.
In questa sezione verrà illustrato come realizzarne una versione di tipo ProPic2.
Lo schema elettrico è il seguente :



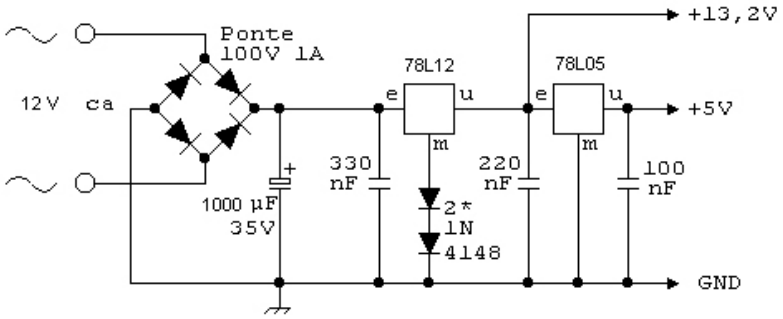


Il micro PIC16F876 ha le seguenti corrispondenze pin -> segnale:

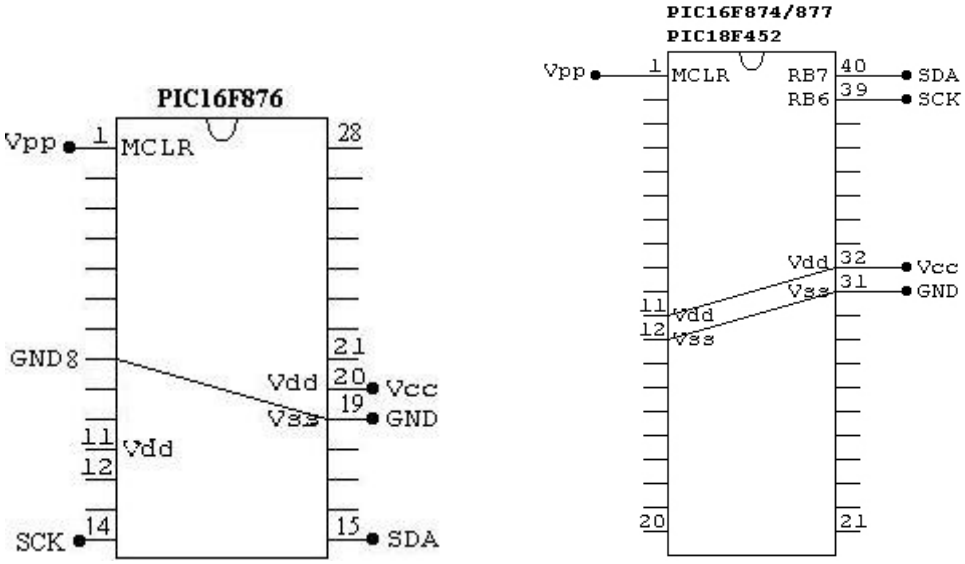
piedinatura micro PIC16F876	
SDA	15
SCK	14
Vcc	20
GND	19-8
Vpp	1

Il circuito, utilizza una porta parallela, i pin indicati per il collegamento alla porta fanno riferimento ad un connettore di tipo Centronics (femmina) che permette l'utilizzo di un normale cavo parallelo da stampanti.

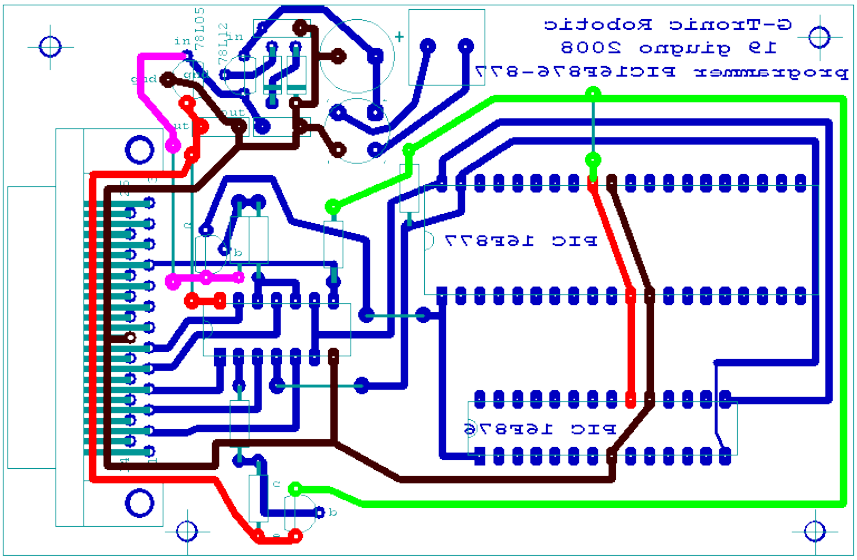
Questo semplice alimentatore in grado di fornire i 13.2V ed i 5V viene integrato nel PCB :



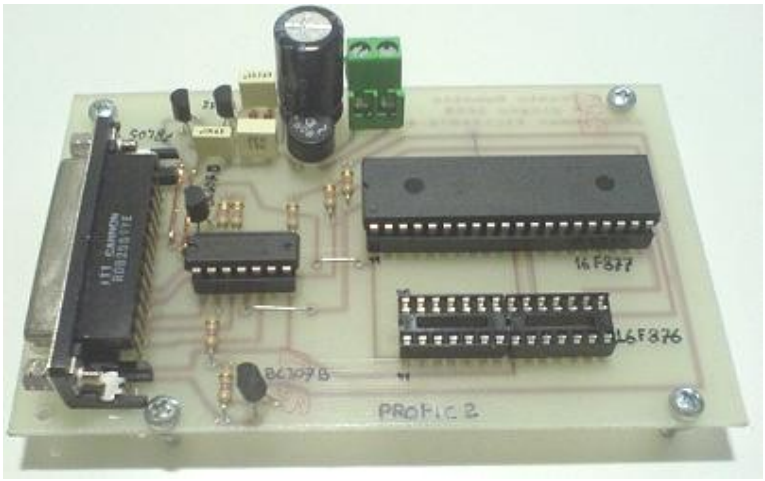
Il circuito permette di programmare tutti i PIC 16F8xx . I collegamenti per il PIC 16F877A sono i seguenti :



Il prototipo del circuito eseguito su una basetta fotosensibile è visibile nell'immagine qui sotto, gli zoccoli per il PIC 16F877A a 40 pin e per il PIC16F876A a 28 pin sono ben accessibili per facilitare le numerose manovre di inserimento estrazione dei processori da programmare.



Le dimensioni del PCB sono 110x70 mm, vi è un solo morsetto a cui collegare indifferentemente una tensione continua o alternata, grazie alla presenza del ponte di diodi. L'unico vincolo sulle alimentazioni è che risulti superiore alla Vpp di almeno un paio di volt, si consiglia di quindi tenerla in un range compreso tra i 15 e circa 20 Volt. Non ci saranno polarità da rispettare al morsetto delle alimentazioni perché queste verranno eventualmente girate dal ponte.
L'eventuale trasformatore da collegare al morsetto può avere dimensioni piuttosto ridotte, 3VA risulta già sufficiente.

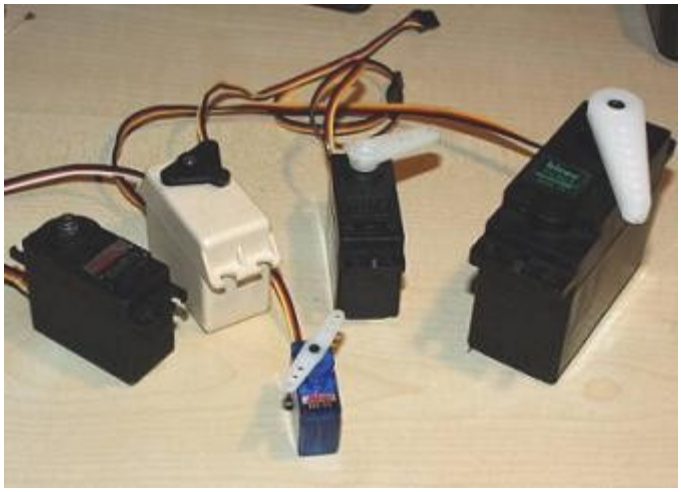


Nella foto vediamo un esemplare del programmer per PIC16F876/877, con un pennarello indelebile sono stati riportati affianco agli zoccoli i nomi dei PIC da inserire per la programmazione. Si nota in primo piano la scritta "**propic2**" che corrisponde alla voce da selezionare nel software che verrà usato per inserire il programma nel PIC.
ll PCB alloggia due zoccoli, uno da 28 pin e uno da 40, rispettivamente per il pic 16F876 e 876A e per il PIC 16F877. Il circuito è molto semplice e non ha alcuna protezione quindi *evitare assolutamente di inserire il PIC in presenza di alimentazione o peggio mentre è stabilita una comunicazione.* trascurare questa avvertenza potrebbe causare la rottura del microPIC. evitare anche di inserire contemporaneamente i due PIC nei due zoccoli perchè il dispositivo non è in grado di distinguere con chi sta comunicando.

In questa seconda foto è messo in evidenza il connettore cannon femmina a 25 poli da collegare alla porta parallela del PC e configurato secondo lo standard usato dai cavi centronics. Sul lato sinistro sono ben visibili i regolatori di tensione 78L12 e 78L05 utili per stabilizzare la tensione di alimentazione e programmazione dei PIC

I servomotori

Un Servomotore è un piccolo attuatore il cui albero è controllato in maniera angolare. Questo può essere portato a posizioni specifiche inviando dei segnali codificati.



Finchè esiste un segnale codificato nella linea d'ingresso, il servo conserverà la posizione angolare dell'albero. Quando il segnale codificato cambia, la posizione angolare dell'albero varia. Nella pratica si usano dei servo per posizionare superfici di controllo quali piccoli sollevatori, flap ,timoni di aeromodelli, ecc .

Inoltre i servo vengono usati in giocattoli radio-controllati, e logicamente in robots. I servo sono molto utili in robotica miniatutizzata,dato che i motori sono piccoli, come vediamo nella foto sovrastante. Essi integrano un circuito di controllo costituito da un anello chiuso che ne rileva l'angolo dell'asse, risultano molto potenti per i loro piccole dimensioni. Un esempio di servo è il HS-300 di Hitec che ha una coppia di 3 kg/cm , per le ridotte dimensione è abbastanza potente.

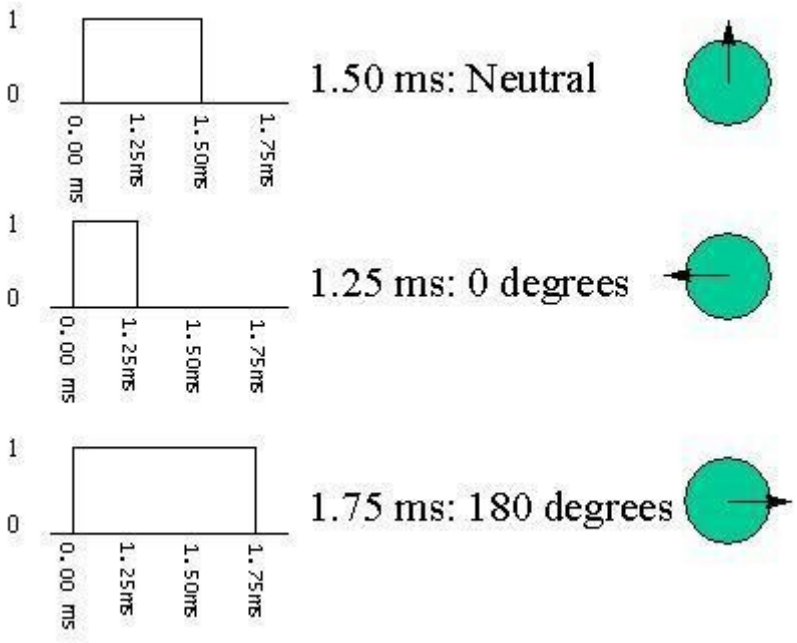
Questo modello offre una potenza proporzionale ai carichi meccanici applicati. Un servo per tanto, dissipa poca potenza.

Nella figura di sotto si vede la composizione interna di un motore di questo genere, sono ben visibili i circuiti di controllo, il motore, l'albero, gli ingranaggi e la scatola.

Ben in evidenza sono i tre fili di connessione esterna. Uno è per l'alimentazione Vcc (+5volts) di colore rosso, altro per la massa GND solitamente di colore nero,e il filo bianco è quello di controllo. Il filo di controllo può essere di colore blu o giallo a seconda della casa costruttrice.

Uno servo smontato.

Modalità di funzionamento del servomotore.
Il sistema è costituito da un anello di retroazione che comprende parti meccaniche e parti elettriche/elettroniche, tipicamente questi sono, una scheda elettronica di controllo con un ingresso analogico, dei comparatori, e un ponte ad H per l'inversione di marcia del motore DC di cui quasi sempre è dotato il servo (molto raramente di un motore A.C. specie per piccole applicazioni), una catena cinematica ovvero un insieme di ingranaggi per l'aumento della coppia all'asse e la riduzione dei giri, e un potenziometro lineare per rilevare la posizione dello stesso.
Nella fotografia vediamo il potenziometro installato alla destra del motore DC e l'insieme degli ingranaggi (deassemblati) che dovrebbero chiudere la catena di controllo.
Il potenziometro permette al circuito di controllo di supervisionare l'angolo attuale del servo motore. Se l'asse si trova nell'angolo giusto, allora il motore sarà spento , se il circuito vede che l'angolo non è giusto l'asse ruoterà fino ad arrivare all'angolo imposto dal comando, questo sia ruotando in verso che nel complementare. Il motore quindi non sarà mai realmente spente perché effettuerà delle oscillazioni, per quanto strette, nell'intorno del punto di posizionamento. La rapidità del controllo e la larghezza della banda in cui il motore può oscillare attorno al punto di posizionamento (death zone) costituiscono la bontà del servomotore. L'asse del servo potrà ruotare fino a 180 o 270 gradi, quota che dipende dal modello.



Nelle normali applicazioni un servo si usa per controllare un posizionamento angolare fra 0 e 180 gradi. Ovviamente un servo, non è capace di andare alla posizione comandata se è soggetto a una coppia resistente applicata all'asse maggiore delle specifiche del fabbricante. Il voltaggio è applicato per un tempo proporzionale alla distanza che l'albero deve percorrere. Così se l'asse deve percorrere una distanza grande, il motore andrà ad altissima velocità, ma se già si trova in prossimità del setpoint il motore si muoverà più lentamente. Questa tecnica è nota come controllo proporzionale.

Fornire al servomotore l'angolo di posizionamento.
Il terzo filo si usa per passare l'angolo al motore. Quest'angolo viene determinato tramite la durata di un impulso monostabile che si applica al filo di controllo. Questa tecnica viene chiamata Modulazione codificata degli impulsi (PCM). Il servo si aspetta ricevere un impulso di 20 milisecondi (0.2 sec). La lunghezza temporale del latch monostabile dell'impulso determina i giri del motore. Un impulso di 1.5 ms, per esempio, farà si che l'albero si posizioni alla quota di 90

gradi (chiamata posizione neutra). Se l' impulso è minore di 1.5ms, allora il motore si avvicinerà ai 0 gradi. Se l' impulso è maggiore di 1.5ms, l'albero si avvicinerà ai 180 gradi.

Come si vede nella figura, la durata del latch indica l'angolo dell'albero (disegnato come un circolo verde con una freccia). Chiariamo che le ilustrazioni e i tempi reali dipendono del fabbricante del servo motore. Il principio, invece, è lo stesso per tutti.

Per i servo Hitec: 0.5 ms= 0 gradi, 1.5 ms = 90 gradi e 2.5 ms = 180 gradi.

0 GRADI (tutto a sinistra)
90 GRADI NEUTRALE (centro)
180 GRADI (tutto a destra)

Modalità di controllo
Per controllare un servo, si invia un certo angolo al terzo conduttore, misurato a partire di zero gradi. Per inviare l'angolo si lanciano una serie di impulsi attraverso il cavo di controllo. In un tempo ON di impulso si impone l'angolo a cui posizionarsi, 1ms = 0°, 2ms= max gradi (circa 120°) e un valore fra questi estremi si traduce in un angolo di uscita proporzionale. Generalmente si considera che in 1.5ms corrisponde alla posizione "centrale". I fabbricanti consigliano di usare il range 1 ~ 2ms, ma si può usare un impulso di 2ms o perfino maggiore per avere in uscita un angolo anche superiore a 180°. Il fattore limitante è il valore massimo del potenziometro nonchè i limiti meccanici sia dell'albero che degli ingranaggi. Un rumore un pò acuto indica che si sta forzando troppo il servo, al di là delle sue capacità di coppia e di angolo. Il tempo di OFF nel servo non è critico, si può stabilire in 20ms. Noi abbiamo usato fra 10ms e 30ms. Questo tempo non è un vincolo che si deve rispettare strettamente, esso può cambiare da un impulso a un altro.
I latch alti che si verificano frequentemente nel tempo di OFF possono interferire con la sincronizzazione interna del servo e potrebbero causare un suono o vibrazione dell'albero. Se la lunghezza dell' impulso è maggiore ai 50ms (dipende sempre del fabbricante), allora il servo potrebbe essere in modalità SLEEP fra i due impulsi contigui. In queste condizioni il servo inizia a funzionare con piccoli movimenti e il rendimento non sarebbe ottimo. Nell'immagine c'è un esempio del segnale che dovrebbe ricevere un servo:

Sopra il tempo di OFF cambia con l'evolversi temporale. Questo non ha effetti negativi se l'impulso si trova fra 10 ~ 30ms. Il tempo di ON determina la posizione del albero. L'utente del servo deve essere attento dato che esistono servomotori con polarità di impulso invertita (cioè dove il tempo di OFF è determinate per la posizione e non quello di ON), attualmente questo tipo di motore è poco reperibile in commercio. Esistono inoltre motori in cui il centro ha una posizione diversa e range temporali diversi. In questi casi non ci resta che cambiare la lunghezza latch o la polarità.

Come muovere il servo ad esempio di 30°
Per muovere a 30 gradi si calcola l'ampiezza dell'impulso:

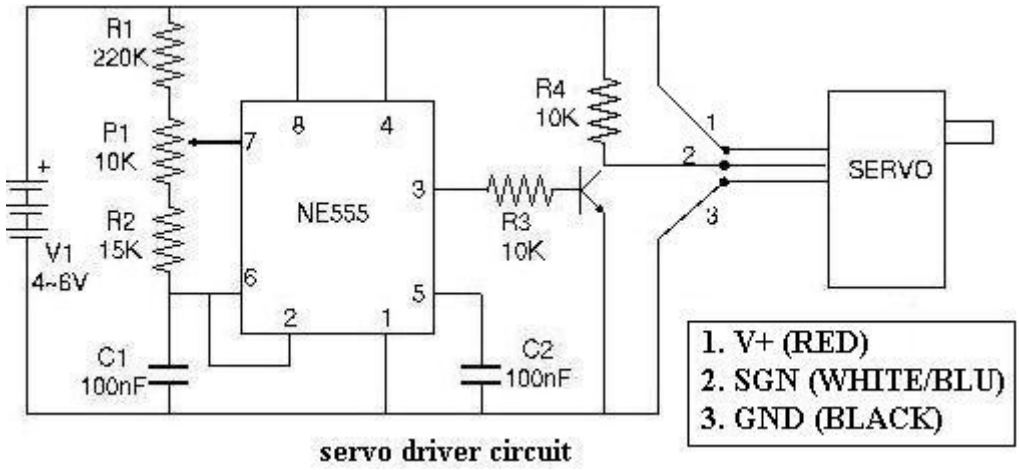
In 0 ° =1ms, 120 ° = 2ms => 30 ° =1.16ms. con relazione lineare.

Così, se si continua a inviare un impulso di 1.16ms, l'albero passa a 30 gradi della posizione attuale.

Se agisce una forza esterna che cerca di bloccarlo, il servo fa resistenza attiva (se l'albero si muove dall'esterno il circuito muove il motore per correggere l'errore riportando l'asse alla quota imposta secondo un concetto di retroazione negativa). Una volta raggiunta la quota desiderata si può anche smettere di inviare gli impulsi. Dopo circa 50ms (dipende sempre del servo) di latenza di trasmissione al cavo di comando il servo si spegne e rimane in posizione solo grazie all'attrito e alla resistenza posta dalla catena cinematica.

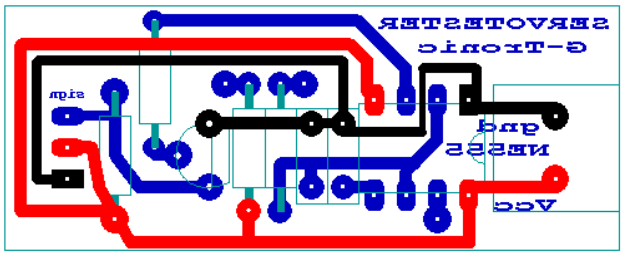
Circuito Driver del servo

Per cominciare a comprendere il funzionamento e il pilotaggio dei servomotori possiamo usare questo circuito. Si tratta in realtà di un circuito da usarsi per "giocare" con i servomotori, ma utili per verifiche di funzionamento o per testare un servo già collegato in un Robot. La prima cosa per usare questo Driver è visualizzare l'impulso inviato al del servo con un oscilloscopio per poi usarlo come modello per riprodurlo tramite un software caricato in un microcontrollore.



servo driver circuit

Usando il FidoCad si è ottenuto il seguente supporto PCB. Come di consueto il colore rosso indica l'alimentazione Vcc a più 5 volt, il nero la massa, il blu le normali piste sul lato rame. Lo streep a tre pin visibile sul lato sinistro rispetta la polarità standard della maggior parte dei servomotori per eseguire il test quindi sarà sufficiente innestare lo spinotto femmina a tre terminali del servo direttamente su questi pin facendo attenzione a tenere il conduttore di segnale (solitamente blu o giallo) nel lato indicato dalla dicitura "sign".



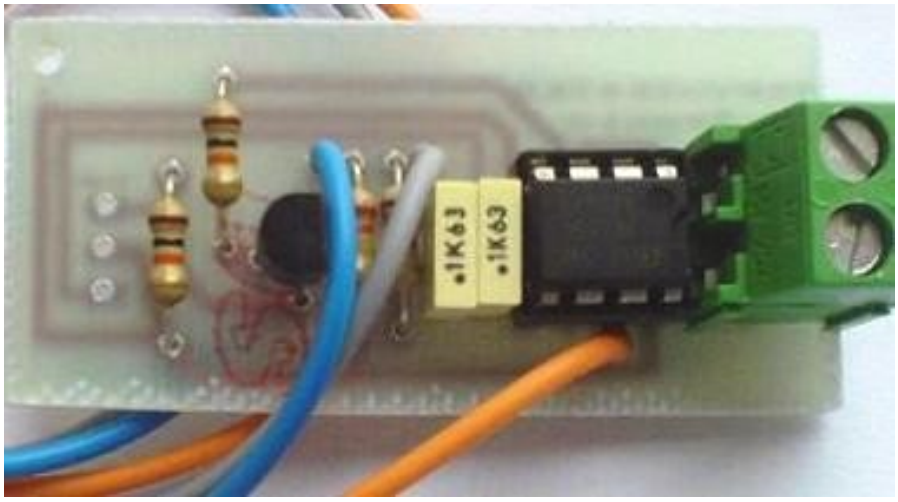
Il potenziometro per la regolazione avrà i due terminali laterali connessi alle piazzole direttamente collegati alle resistenze, il cursore ovvero il terminale centrale sarà saldato alla piazzola presente al pin 7 dell'integrato.

Questo circuito usa un integrato 'Timer' NE555. fabbricato praticamente da tutti i costruttori di componenti. Lo schema interno si trova nei manuali di ST, National, Motorola e altri, con i valori dei resistori/condensatori calcolati con le formule del caso. Il potenziometro il cui asse è controllato dall'utente regolerà i tempi del latch monostabile a cui corrisponde la posizione del servo. Il segnale di uscita del IC (pin3) risulta invertito nei livelli logici quindi per invertirla è stato inserto il transistor. Il BJT collegato in modalità 'collettore comune' e tenuto nella zona di saturazione (opera

quindi in modalità ACCESSO o SPENTO), invertendo i livelli logici del pin 3. Può essere usato qualsiasi transistore npn (Nel nostro esempio usiamo un C1959Y).
Qui sotto viene riportata la lista dei componenti :
R1: 220K
R2: 15K
R3: 10K
R4: 10K
P1: 10K
C1: 100nF
C2: 100nF
V1: 4~6V

Batterie 4 AA o usare alimentatore a 5 Volt e usare qualsiasi transistore npn per piccoli segnali.
Il circuito risulta essere molto compatto e potrà essere inserito in un piccolo contenitore di plastica a cui aggiungere una batteria da 9 volt e il relativo regolatore uA7805, un mini interruttore, un led per la segnalazione dell'accensione ed ovviamente il potenziometro miniaturizzato con la relativa manopola. Dal contenitore dovrà sporgere il connettore a 3 pin in cui innestare i terminali del servomotore da testare.

Come evidenziato nella foto in una basetta di dimensioni standard 100 x 160 mm, grazie alle contenute dimensioni ci stanno ben 14 esemplari del servotester.



Altro utilizzo del circuito di driver del servo.
Finora abbiamo sempre fatto cenno ai servomotori miniaturizzati per modellismo dato che possono essere impiegati in molti ambiti della robotica hobbistica, in realtà la limitazione di coppia disponibile all'asse potrebbe impedirne il corretto uso in quei casi in cui si debba generare "potenza muscolare simulata" ad esempio in un umanoide a grandezza naturale. Benché il circuito elettronico di controllo del servo rimanga invariato si dovrà adeguare la sezione di potenza per l'inversione del motore D.C. appropriato alla coppia che sarà necessario sviluppare. Un ponte ad H in grado di sostenere fino a 8 ampere realizzato con 4 darlinghton di potenza è presentato nei capitoli successivi. Il problema di una motoriduzione adeguata al controllo di un grado di libertà di questa entità è l'assoluta impossibilità di riposizionamento manuale, difatti l'asse esterno risulterà molto duro da muovere. Forzare, ad esempio, il braccio dell'umanoide per metterlo in posizione di parcheggio in assenza di alimentazione potrebbe significare il danneggiamento della struttura meccanica dell'articolazione. La manovra diviene semplice se si scollega provvisoriamente il circuito "driver" che posizionerà il motore alla quota desiderata agendo sul potenziometro. Naturalmente la batteria da 9 volt integrata nella scatolina del circuito potrebbe non bastare ma in questo caso si potrà collegare un opportuno generatore al morsetto a vite ovviamente rispettando la corretta polarità

Colori standard dei cavi del servomotore
I cavi del servo hanno dei colori secondo l'uso, come nella leggenda dello schema precedente. Anche i servo Hitec, Futaba di uso hobbistico usano la stessa notazione. JR e Graupner hanno il filo di controllo arancio. Altri come Sanwa (Airtronics) hanno la linea GND di colore blu. Altri motori Sanwa hanno tutti i fili neri, con una riga rossa a fianco. Il filo con la riga è il Vcc, quello dopo è GND e l'ultimo e il filo di controllo (questa notazione e diversa dai motori Futaba)

Gli Hitec, Futaba o Hobbico hanno la seguente notazione :

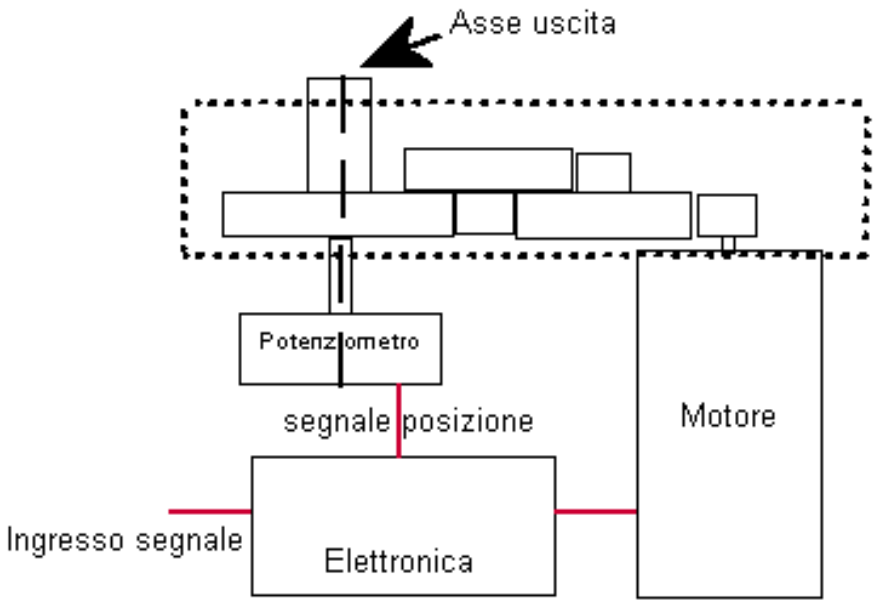
Segnale di controllo	(Giallo o Bianco)
Vcc	(Rosso)
GND	(Nero).

I numeri e le posizioni dei cavi nello schema sono arbitrarie, ma si deve verificare il proprio servo prima di collegarlo.
Invertire il segnale di alimentazione può danneggiare il servomotore.

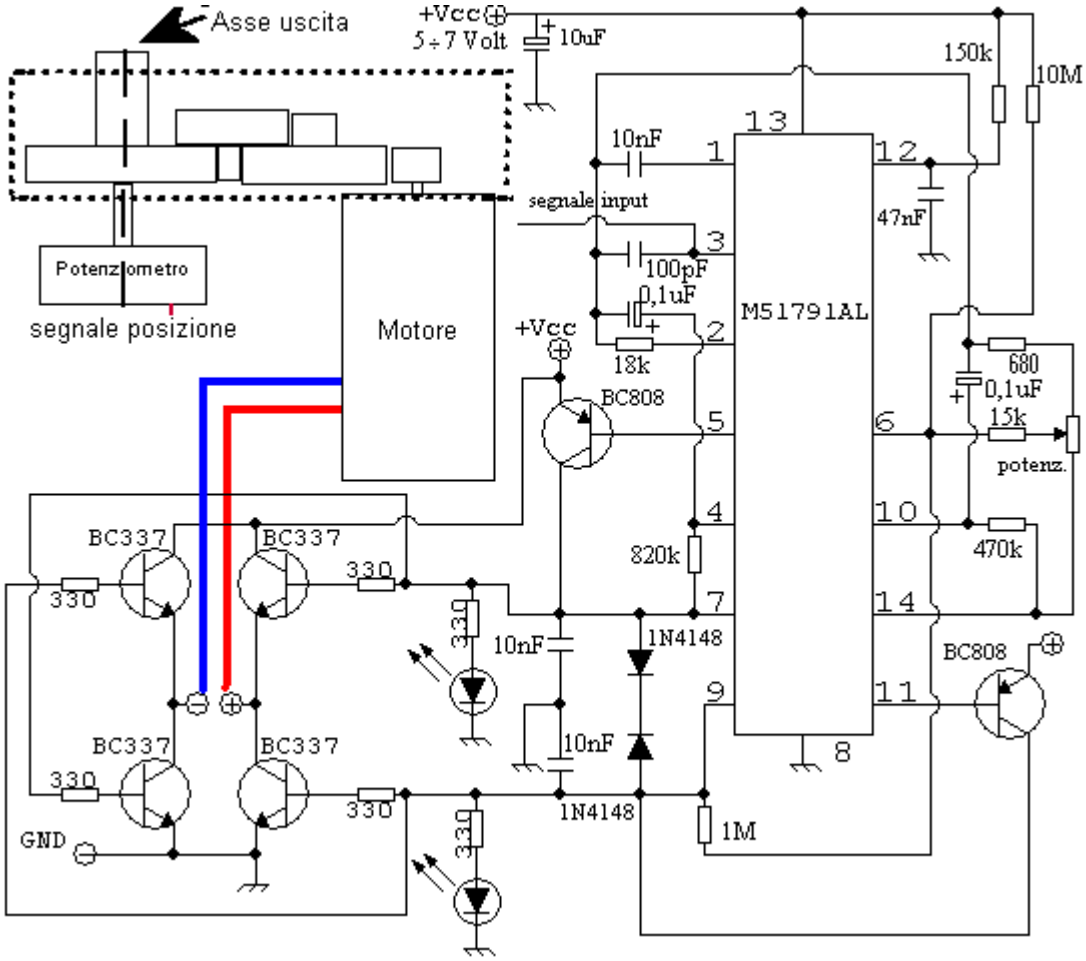
Alimentazione
I volt richiesti sono quelli che fornisce un pacco di batterie di 4x1.2V di NiCd ; cioè 4.8V. Nella pratica questo livello può variare molto. Alcune aziende di servo producono pacchetti di 5 unità di NiCd, con un voltaggio di 6V, ma forniscono 6.5 ~ 7V appena caricate. Futaba offre le specifiche del servo (velocità/coppia) per 6V e consideriamo 7V come un massimo 'sicuro'. Inoltre supponiamo che i servo lavorino con un pacco di batterie NiCd di 4 unità, a 4.4V, questo comporterebbe che la risposta sia un pò lenta. Il range di lavoro è coretto quindi fra 4.4 e 7 V, a scelta dell'utente da farsi in base alla necessità di velocità in risposta o di durata del servizio. Dai test eseguiti si consiglia di alimentare i servomotori a 5V. Si può usare un alimentatore di 5V, o usare circuiti regolatori, come il 7805. La corrente richiesta dipende delle dimensioni del servo e dall'attuatore DC che esso contiene. Normalmente il fabbricante dice qual è la corrente richesta dal

proprio prodotto. Ovviamente questa informazione è molto variabile quando si muovono diversi servo allo stesso tempo. In ogni caso la corrente dipende principalmente della coppia usata dal motore e può superare 1A se il servo è incastrato riferendoci ovviamente al prodotto commerciale per applicazioni hobbistiche.
In quei casi in cui si intende usare un servomotore costruito ad hoc per una applicazione professionale conviene misurare le specifiche eseguendo le opportune verifiche volt amperometriche.

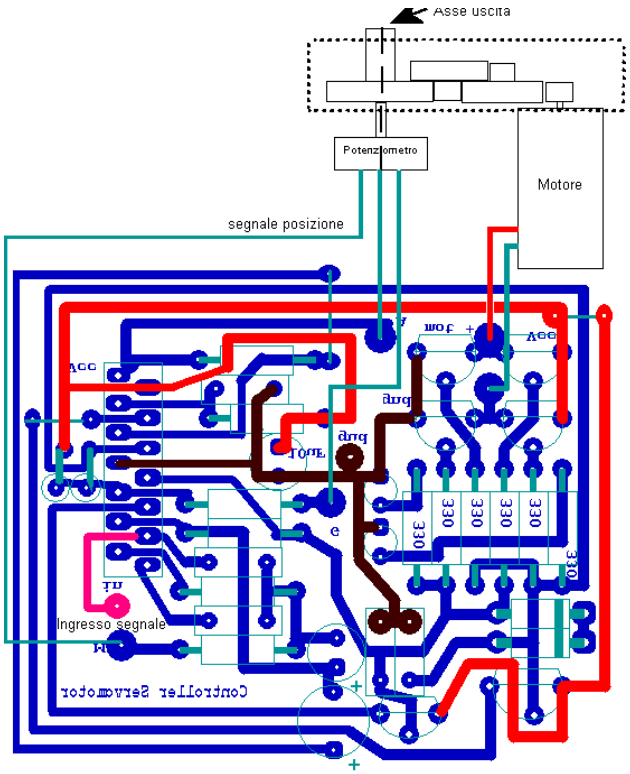
Versione dedicata del servomotore.



Tutti i servomotori rispettano lo schema a blocchi soprariportato, quindi formano un anello chiuso da una catena cinematica formata da una riduzione ad ingranaggi generalmente in nailon o metallo. Le parti strettamente elettriche ed elettroniche sono l'attuatore D.C., l'elettronica di controllo, il sensore di posizione costituito generalmente da un potenziometro che trasduce un angolo in un valore analogico di tensione. Nulla vieta che il trasduttore sia un encoder in grado di dare direttamente un'informazione digitale. Benché nei prodotti commerciali il potenziometro si trovi quasi sempre all'interno della scatola che costituisce il corpo del servomotore si può senza problemi tenerlo esterno senza compromettere il funzionamento del dispositivo. Ovviamente in questo caso si perde l'impermeabilità, caratteristica essenziale di alcune applicazioni modellistiche.



Esistono in commercio diversi circuiti integrati che realizzano il circuito di controllo del servomotore con altri pochi componenti di contorno, nella versione dedicata del servomotore si è deciso di usare l' M51791AL perché risulta essere impiego con grande successo da una delle principali case costruttrici di servomotori.
Come si vede dallo schema il segnale del trasduttore entra nel pin 6 dell'integrato dove viene convertito e comparto con il segnale del falso PWM, anch'esso convertito, che arriva dalla scheda a micro PIC. Il risultato delle comparazioni sono due segnali di comando avanti e indietro dell'attuatore e un terzo stato logico che corrisponde ad assenza di comando in entrambe le uscite. Quando viene generato il terzo stato il motore risulta spento, ma nei casi reali in cui vi sia una coppia esterna applicata all'asse è una condizione rara, difatti il motore oscillerà in una banda più o meno stretta nell'intorno del setpoint generando il tipico ronzio che accompagna sempre il funzionamento dei servomotori. Cercando si muovere con le mani l'asse di un servo alimentato e portato in quota si avrà l'impressione che esista una coppia resitente continua sia in un verso che nell'altro, ovvero a meno dei una piccola vibrazione che il motore sia bloccato.
Il filo che normalmente è blu e che costituisce il segnale di comando del servo motore entra nel pin 3, qui è presente il falso PWM che come è noto avrà il tempo di latch alto (il duty cycle se fosse un effettivo PWM) proporzionale all'angolo a cui si deve posizionare l'asse.



Lo schema elettrico del servomotore dedicato è stato elaborato con il FidoCad per ottenere il sovrastante PCB. Sul lato sinistro è visibile il circuito integrato M51791AL nella versione con housing SIP14. Esiste anche la versione dual in line (D.I.L.) ma in questo caso si dovrà ridisegnare il layout. In alto a destra vediamo il ponte ad H integrato che realizzato con 4 BJT di tipo NPN modello BC337 può sopportare oltre 1 A con correnti di saturazione dell'ordine dei 2 milli ampere. Nel caso fosse necessario un ponte ancora più robusto si può interfacciare il ponte a darlinghton TIP122 presentato più avanti. L'interfacciamento con questo circuito avverrà semplicemente non montando i BC337 e prelevando i segnali di marcia avanti e marcia indietro direttamente a monte delle resistenze da 330 ohm attualmente connesse alla base. È opportuno bypassare queste resistenze perché nel circuito del ponte rinforzato sono già presenti quelle corrette. Se non si vuole caricare troppo l'uscita del micropic a cui il circuito del servo risulta collegato potremmo scollegare anche i due led rosso e verde di segnalazione del senso di marcia evitando di buttare a massa attraverso di essi ben 1 milli ampere, valore che corrisponde a quasi il 50% del valore minimo da portare alla base di un TIP122 per farlo saturare.

Nella foto vediamo il prototipo del servomotore montato e collaudato con un motoriduttore della casa micromotors.

Questo attuatore ha una tensione nominale di indotto paria a 12 volt continui e una corrente di regime di circa 50 milli ampere. La velocità di rotazione è di 66 rpm. Il piccolo attrattore visibile alla sua sinistra è quello normalmente impiegato nei servomotori ad uso modellistica, la differenza di misura può darci una stima indicativa della molto più elevata coppia motrice prodotta.

Motoriduttore MG-S-3736-03-90
- Tensione 12V
- Velocità 66 rpm
- Coppia Nominale 7 Kg cm
- Dimensioni: diametro 37 mm, lunghezza 72 mm

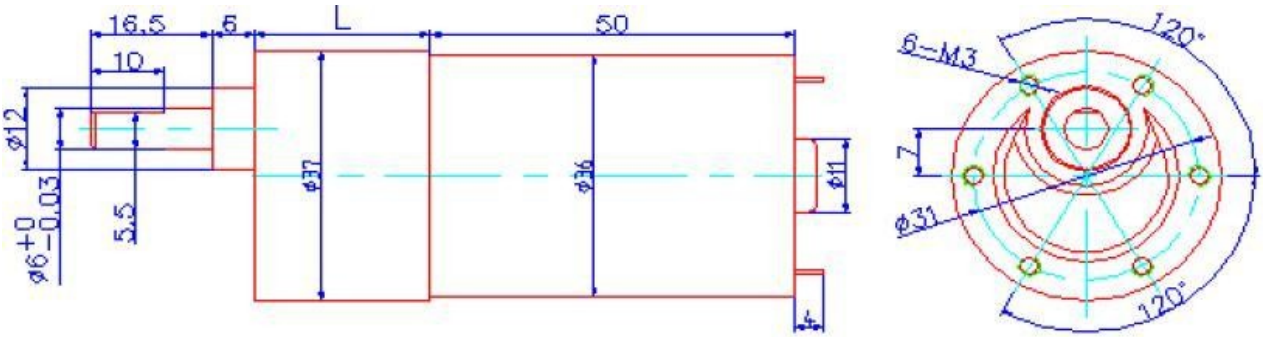


Tabella delle riduzioni

La casa costruttrice del motoriduttore mette a disposizione la tabella delle riduzioni che permette al cliente finale di configurare la catena cinematica nella maniera più adeguata alle proprie esigenze. Le unità di misura di riferimento sono la coppia in chilogrammi per centimetro [kgcm] e la velocità in rivoluzioni complete al minuto [rpm]. Di non secondaria importanza sono anche le informazioni relative alla alimentazione dell'indotto in Volt continui [V DC] e l'assorbimento in milliampere ad asse scarico o alla coppia nominale resistente applicata all'asse [mA].

Il rapporto di riduzione, o reduction ratio, è il parametro di riferimento da fornire all'atto dell'ordinazione

MODELLO	rapporto	1/6	1/10	1/17.5	1/30	1/50	1/90	1/160	1/270	1/470	1/810	Voltage	Non-load Current	Loaded Current
	coppia [Kgcm]	0,8	1	1,5	2,5	4	7	12	20	20	20	(V DC)	(mA)	(mA)
MG-S 3736-01	Velocità [rpm]	500	300	170	100	60	33	18	11	6	3,7	12	=200	=700
MG-S 3736-02	velocità [rpm]	750	450	250	150	90	50	28	16	9	5,5	12	=250	=1000
MG-S 3736-03	velocità [rpm]	1000	600	340	200	120	66	36	22	12	7,4	12	=350	=1300
MG-S 3736-04	velocità [rpm]	500	300	170	100	60	33	18	11	6	3,7	24	=120	=350
MG-S 3736-05	velocità [rpm]	750	450	250	150	90	50	28	16	9	5,5	24	=150	=500
MG-S 3736-06	velocità [rpm]	1000	600	340	200	120	66	36	22	12	7,4	24	=180	=700

Reduction Ratio	L (mm)
1/6~1/30	19.05
1/30~1/100	22.05
1/120~1/300	26
1/360~1/900	29

Ponte ad H rinforzato

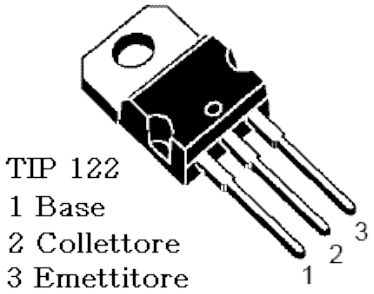
Se si istallano motori DC più robusti è necessario rinforzare il ponte ad H, a questo scopo ben si presta il semplice circuito costituito da 4 transistors darlighton TIP122. è un configurazione piuttosto classica che permette l'inversione di marcia dei motori a corrente continua con eccitazione a magnete permanete.

I TIP122 sono transistor molto robusti che portano al collettore una corrente di oltre 8 ampere in grado di gestire qualsiasi applicazione di media potenza.

I segnali in tensione che pilotano i versi di marcia (avanti o indietro) vengono adattati con opportune resistenze di base le quali lasceranno passare in base i 10 mA necessari per la saturazione del TIP122.

La velocità di commutazione tipica dei transistors Darlinghton consentono inoltre il pilotaggio con segnali di Tipo PWM quindi il circuito ben si presta anche alla regolazione di velocità del motore ad esso collegato teoricamente senza perdita di coppia motrice.

Due spie L.E.D. segnalano il senso di marcia, Verde (motore in marcia avanti) Rosso (motore in marcia indietro).

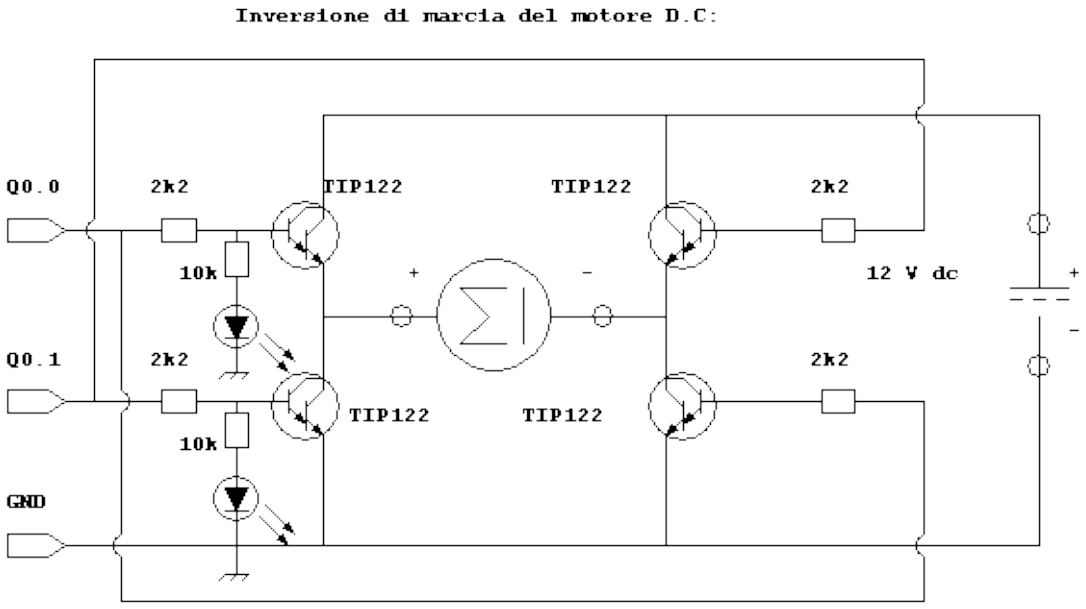


TO-220

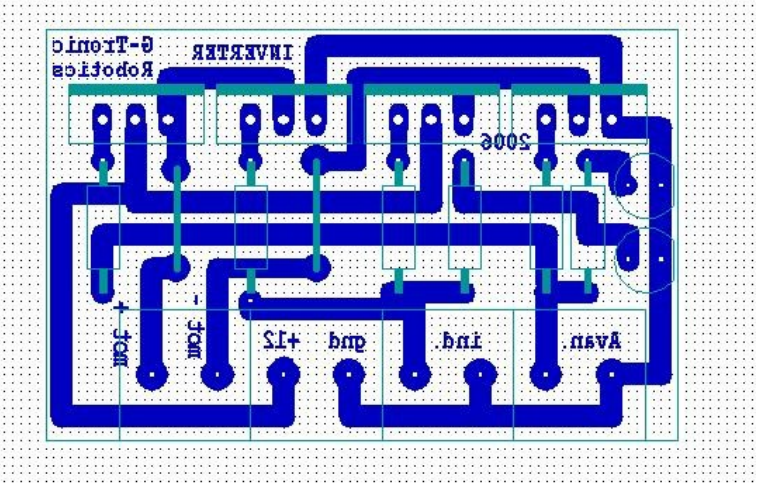
Nella tabella sono riportati i valori più opportuni delle resistenze di Base per il collegamento a una scheda a microprocessore, a una scheda generica a 12 Volt, oppure a un controllore industriale (P.L.C. a + 24 Volt D.C.).

tensioni di comando più classiche.	
5 Volt	330 Ohm
12 Volt	1 K Ohm
24 Volt (comando da PLC)	2,2 KOhm

Lo schema elettrico è una tipica configurazione a ponte spesso chiamato nei testi di elettronica ponte ad H.



Il circuito stampato dell'interfaccia per l'inversione di marcia risulta molto compatto come possiamo vedere in figura.



Nella foto è rappresentato un esemplare montato e collaudato del circuito. In questo caso non bisogna assolutamente cortocircuitare con un dissipatore i corpi metallici dei TIP122 perchè essi sono collegati internamente al collettore del BJT, come si vede dallo schema elettrico comporterebbe un sicuro e distruttivo malfunzionamento.

L'ingombro di questo circuito è di soli 55 mm x 33 mm.

Apparati sensoriali

Affinché l'arto risulti il più possibile "human like" è necessario che esso sia munito almeno dei principali sensi che permettano al sistema di controllo di distinguere se l'oggetto afferrato è caldo o freddo, o se la presa è sicura o labile.

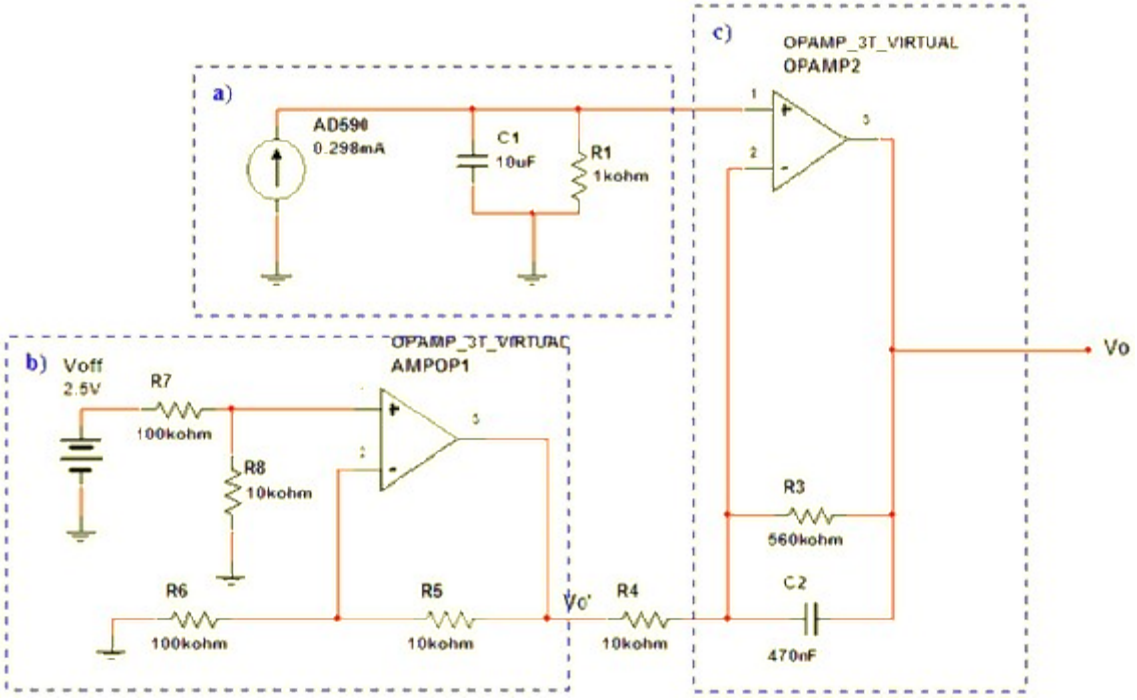
Sonda AD590

Le sonde usate per rilevare la temperatura sono le AD590, esse vanno collegate con cavetto schermato a due conduttori interni (un cavetto per ogni sonda se si intende dislocarne più di una nelle varie posizioni della protesi). si può arrivare anche a10 m di lunghezza, sono controllate in corrente, per cui la lunghezza del cavo non dovrebbe influire. La sonda AD590 genera una corrente di 1µ A / ° K pertanto su una resistenza da 1Kohm si avrà una caduta di tensione parti ad 1mV/ ° K.

Consiglio di calibrare le sonde per confronto con un termometro a mercurio abbastanza preciso, rilevando diverse temperatura (almeno 6 valori) abbracciando tutta la possibile escursione che si prevede di dover registrare.
Le letture vanno fatte registrando i valori dei codici binari in uscita dal AD converter ottenibili facendo girare le routine readtemp1 (sonda asciutta) readtemp2 (sonda bagnata). Quest' ultima deve essere asciutta quando viene calibrata. Si consiglia di leggere i valori -5°C, 0°C, 7°C, 15°C, 25°C, 35°C. Il range si può aggiustare in base alle proprie esigenze, o si possono aggiungere altre acquisizioni di riferimento.

Traduzione e condizionamento

L'acquisizione del valore della temperatura ambientale, la trasduzione in corrente elettrica e la manipolazione del segnale acquisito, verranno effettuate da un unico circuito , di cui sotto si riporta lo schema :

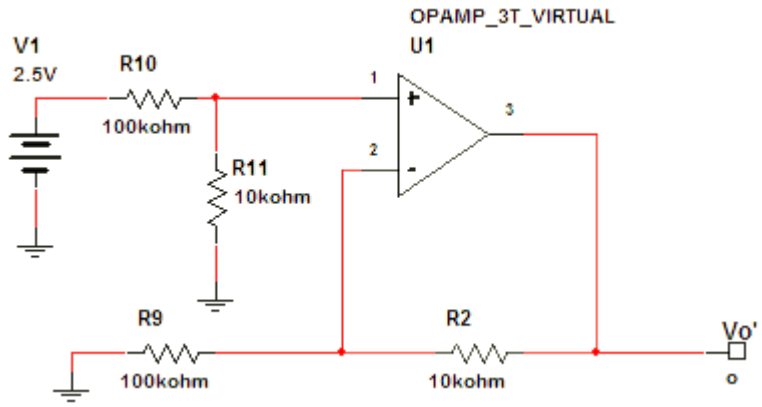


L'intero circuito lo si può vedere suddiviso in tre sezioni principali :

- a) trasduttore AD590 con circuito di condizionamento

- b) stadio amplificatore in configurazione non invertente con funzione di offset
- c) stadio amplificatore in configurazione differenziale

Iniziamo con l'analizzare la sezione 'b)' del circuito, con il fine di ricavarne una relazione generale che ci fornisca informazioni circa i valori assunti da Vo' ed il tipo di relazione che la lega alla Voff (tensione di offset).



$$V_0' = \left(1 + \frac{R5}{R6}\right) * \left(\frac{R8}{R8 + R7}\right) * Voff$$

sostituendo i valori :

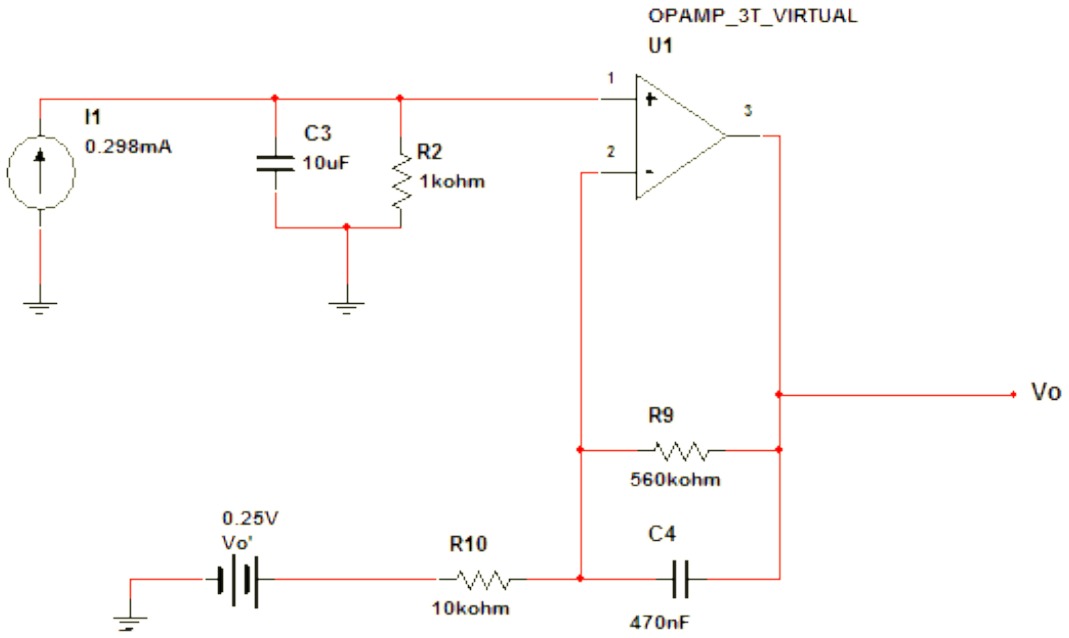
$$V_0' = \left(1 + \frac{10K\Omega}{100K\Omega}\right) * \left(\frac{10K\Omega}{10K\Omega * 100K\Omega}\right) * 2{,}5V$$

$$V_0' = \left(1 + \frac{1}{10}\right) * \left(\frac{1}{11}\right) * 2{,}5V$$

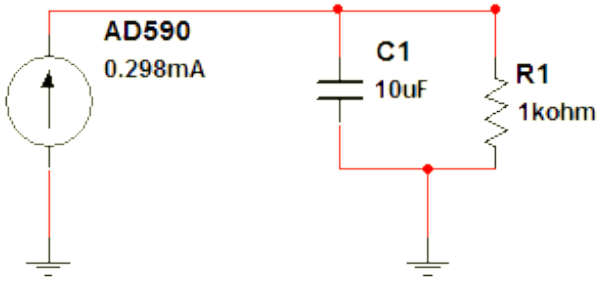
$$V_0' = \left(\frac{1}{10}\right) * 2{,}5V$$

$$V_0' = 0{,}25V$$

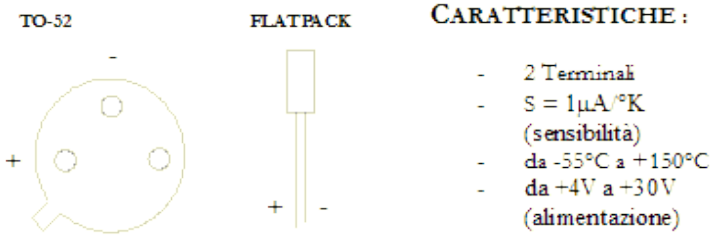
In uscita dalla sezione b) del circuito si ha dunque una tensione Vo = 0,25 Volt . Possiamo quindi semplificare l'intero circuito nel seguente modo :



Per quanto riguarda la sezione a) del circuito , essa si compone di un trasduttore AD590, un condensatore da 10(micro-Farad) con in parallelo una resistenza da 1(kilo-Omh).



Pinot dell'AD590 (trasduttore di temperatura a due terminali) :



Il circuito integrato AD590 è un trasduttore di temperatura a **2 terminali**, esso fornisce in uscita una corrente direttamente proporzionale alla temperatura assoluta, con una sensibilità di 1(micro-Ampere)/°K. Ai fini della taratura è importante sottolineare che ad una temperatura di 25°C (298.2°K) l'IC (Circuito Integrato) fornisce una corrente di output di 298.2 A.
L'AD590 può essere alimentato con tensioni che vanno dai 4V ai 30V e consente di rilevare temperature che vanno da **-55°C (218°K) a +150°C (423°K).**

Il trasduttore fornendo in uscita una corrente, è insensibile ai disturbi di tensione che si potrebbero verificare lungo le linee di collegamento dell'AD590 al circuito e per questo è particolarmente indicato per le misurazioni di temperature remote.
Al fine di semplificarne l'analisi è possibile non tener conto del condensatore, il cui scopo è quello di attenuare i disturbi nel segnale, riducendo il circuito di trasduzione al solo AD590 con in parallelo il resistore da 1(Kilo-Ohm); lo si può dunque paragonare ad un generatore reale di corrente, che fornisce una corrente proporzionale alla temperatura secondo la seguente relazione :

$$I1 = T * S$$

dove: T = temperatura in gradi Kelvin

S = sensibilità del trasduttore (1μA/°K)

I1 = corrente fornita dal trasduttore

sostituendo si ottiene :

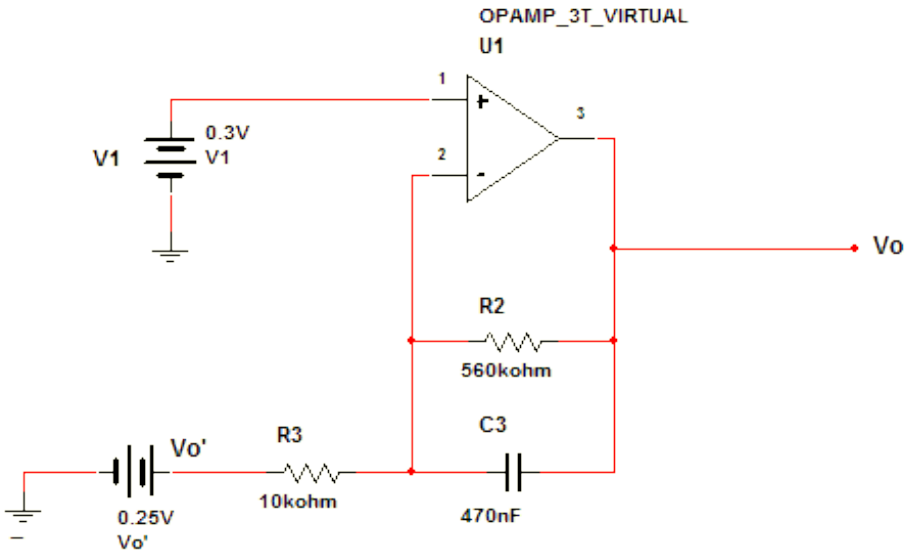
$$I1 = T * 1\frac{\mu A}{^{\circ}K}$$

E' possibile inoltre sostituire al raggruppamento trasduttore - resistore , un generatore di tensione che fornisce una d.d.p. pari alla caduta di tensione ai capi della resistenza R1, in tale caso la V1(tensione ai capi di R1) si otterrebbe dalla seguente formula :

$$V1 = R1 * I1$$

$$V1 = T * 1\frac{\mu A}{^{\circ}K} * R1$$

Il circuito generale può dunque subire una ulteriore semplificazione, riducendosi ad un solo stadio amplificatore in configurazione differenziale , corrispondente alla sezione c) del circuito :



Anche in questo caso, in analisi può essere trascurata l'influenza del condensatore, quindi la relazione finale ingresso-uscita sarà la seguente :

$$Vo'' = -\frac{R2}{R3} * Vo' + \left(1 + \frac{R2}{R3}\right) * V1$$

sostituendo i valori :

$$Vo'' = -\frac{560K\Omega}{10K\Omega} * 0,25V + \left(1 + \frac{560K\Omega}{10K\Omega}\right) * V1$$

$$Vo'' = -56 * 0,25V + (1 + 56) * V1$$

$$Vo'' = -14V + 57 * V1$$

secondo la relazione finale, la V1 viene amplificata di 57 volte e gli viene sommato un offset negativo di 14V.
Analizziamo ora il range di variabilità della V1, al fine di comprendere di conseguenza i valori entro i quali può oscillare la Vo" :

$$V1_{MAX} = R1 * I1_{MAX}$$

$$I1_{MAX} = T_{MAX} * S \Rightarrow I1_{MAX} = T_{MAX} * 1\frac{\mu A}{^\circ K}$$

$$T_{MAX} = 423^\circ K$$

riassumendo:

$$V1_{MAX} = 1K\Omega * 423^\circ K * 1\frac{\mu A}{^\circ K}$$

$$V1_{MAX} = 0,423V$$

allo stesso modo calcoliamo il valore minimo assumibile da V1 :

$$V1_{MIN} = R1 * I1_{MIN}$$

$$I1_{MIN} = T_{MIN} * S \Rightarrow I1_{MIN} = T_{MIN} * 1\frac{\mu A}{^\circ K}$$

$$T_{MIN} = 218^\circ K$$

riassumendo :

$$V1_{MIN} = 1K\Omega * 218^\circ K * 1\frac{\mu A}{^\circ K}$$

$$V1_{MIN} = 0,218V$$

quindi :

$$0,218V \le V1 \le 0,423V$$

determiniamo dunque i valori entro cui è compresa la Vo" (tensione d'uscita) :

$$-14V + 57 * 0{,}218V \leq Vo'' \leq -14V + 57 * 0{,}423V$$

$$-1{,}57V \leq Vo'' \leq +10{,}11V$$

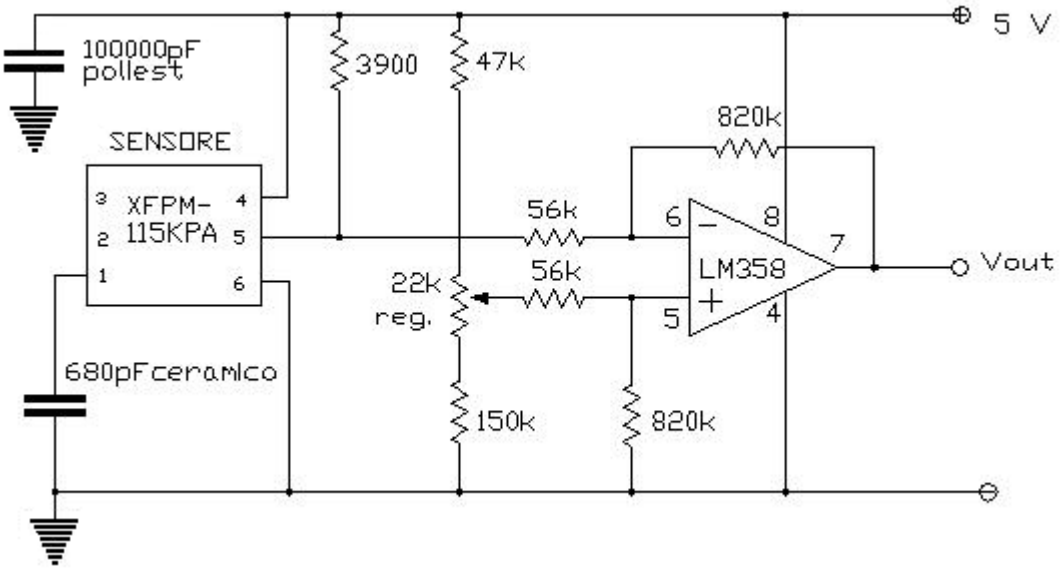
al fine di rendere TTL compatibile l'uscita del circuito , è possibile introdurre tra l'uscita e massa un resistore con in parallelo un Diodo Zener di Vd=4,7V in modo da limitare la Vo" a valori compresi tra 0V e 5V; riducendo tuttavia in tal modo l'escursione dei valori rilevabili, a temperature comprese tra i -28°C (245°K) ed i 60°C (333°K).

Valori Teorici delle trasduzioni :

Temperatura (°C)	Temperatura (°K)	I1 (mA)	V1(V)	Vo"(V)
-5°C	268°K	268	0,268	1,27
2°C	275°K	275	0,275	1,40
26°C	299°K	299	0,299	3,04
60°C	333°K	329	0,320	4,70

sensore di pressione XFPM 115KPa

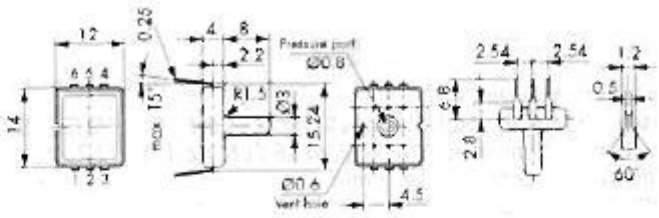
Il sensore di pressione XFPM 115KPa della Fujikura, fornisce una tensione proporzionale alla pressione applicata a partire da 4.083 V a 1013 hPa con variazioni 3.82 mV per hPa. Viene acquisito tramite un semplice amplificatore differenziale che provvede ad amplificare queste variazioni di circa 12 volte (rapporto 820k/56K). Il potenziometro da 22K serve a centrare le variazioni all'interno dei 5V di alimentazione.
Tutte le resistenze sono al 5% di tolleranza.



Sensori di pressione

Fujikura

■ XFPM PDF



- *Adatto per gas non corrosivi*
- *Sensori di pressione relativa e di depressione -1...+12 bar*
- *Compensato in temperatura*
- *Adatto per gas non corrosivi*
- *Alta precisione*
- *Montaggio su circuito stampato di 6-Pin DIL package*

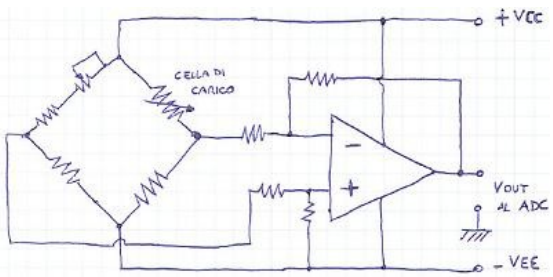
Tensione di ingresso	5 VDC ±0,25 VDC
Segnale di uscita	0,2...4,7 VDC
Precisione	±2,5% @ 0...85 °C
Temperatura di esercizio compensata	0...+85 °C
Temperatura di esercizio max.	-40...+125 °C

Art.No	Tipo	Portata[bar	Tipo di misura	Prezzo per 1 unità 1+	2+	5+
24 10 57	XFPM-025KPGR	0...0,250	Pressione relativa	18.60	17.88	16.90
24 10 58	XFPM-050KPGR	0...0,500	Pressione relativa	18.60	17.88	16.90
24 10 59	XFPM-100KPGR	0...1	Pressione relativa	18.60	17.88	16.90
24 10 60	XFPM-200KPGR	0...2	Pressione relativa	18.60	17.88	16.90
24 10 61	XFPM-700KPGR	0...7	Pressione relativa	23.34	22.47	21.23
24 10 62	XFPM-001MPGR	0...10	Pressione relativa	18.60	17.88	16.90
24 10 63	XFPM-001,2MPGR	0...12	Pressione relativa	17.22	16.31	15.50
24 10 64	XFPM-100KPGVR	0...-1	Vuoto	22.07	20.82	19.71

Celle di carico miniaturizzate

Le celle di carico sono delle resistenze soggette a una variazione del loro valore ohmico proporzionale con buona linearità allo sforzo meccanico ad esse applicate. Ne esistono di svariate forme e dimensioni con ampi range di variazione dei parametri, in ogni caso esse possono essere acquisite tramite un ponte resistivo che ne consente la traduzione bipolare e con alta sensibilità. Tale ponte è bene costituisca lo stadio di ingresso di uno stadio differenziale ad operazionali per strumentazione.

Segue l'immagine di uno schizzo a mano dell'idea di base.



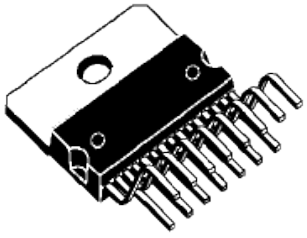
Alimentazioni per la logica e per la potenza

Prima di presentare lo schema elettrico del progetto è doveroso almeno un rapido confronto tra le caratteristiche di un alimentatore lineare e uno switching. Questo ultimo risulta più efficiente del tipo lineare così che a parità di potenza erogata si può optare per un trasformatore e un dissipatore più piccolo e quindi più economico. Ne ha ovviamente vantaggio anche l'ingombro e il peso del prodotto finito.

Lineare
Per una regolazione ottimale la caduta minima deve valere almeno **5Volt su 4A.** La tensione nominale del trasformatore dovrà valere almeno **14 Vnom** per ottenere in qualsiasi condizione di stress una tenuta dela fornitura di energia senza cali di tensione. La potenza dissipata dalla cascata di componenti dalla sorgente fino all'uscita vale **Pd=(Vnom-Vo)Io = 36W** il dissipatore necessario a smaltire il calore avra una resistenza termica **Rth = 0.8 °C/W** La potenza fornita dal trasformatore deve vale: **Pdiss = 14 x 4 = 56 W** Questi dovrà quindi essere dimensionato per: **Pd= 56/0,9 = 62 VA**

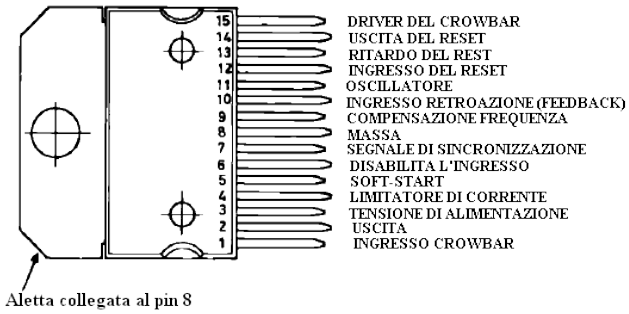
Switching
Assunto lo stesso voltaggio nominale di **14 Vnom**, il data sheet del circuito integrato **L296** indica che la potenza dissipata in questo caso è di soli **7W.** Tale potenza è dissipata in soli due elementi, Il circuito integrato L296 e nel diodo di ricircolo (nello schema **BPW80**). Ne consegue che il trasformatore può essere dimensionato attorno a **30 VA** è il dissipatore di calore deve avere una resistenza termica nell'intorno di **11°C/W.** **questa comparazione mette in evidenza che il regolatore switching L296 consente un risparmio all'incirca del 50% nel costo del trasformatore (in denaro e in spazio) ed un impressionate vantaggio del 80 - 90% nel costo del dissipatore** Va considerato inoltre che il circuito integrato mette a disposizioni un numero piuttosto elevato di opzioni che se implementate con componenti discreti richiederebbero una ulteriore spesa. Per finire va segnalato che L296 grazie alla sua bassa dissipazione di potenza è il componente ideale per lavorare in chassy di ridotte dimensioni.

il circuito integrato L296 è alloggiato in un housing plastico di tipo MULTIWATT® a 15 pin, che ne include anche la sezione di potenza e un regolatore p.w.m. utilizzablie anche per pilotare motori in continua di piccola e media potenza. Questa sezione pwm è controllata da un oscillatore implementato internamente in grado di generare un segnale fino a 200 Khz. La sezione di potenza può fornire fino a 4 ampère al carico e la tensione risulta regolabile da 5.1 Volt fino a 40 volt, mentre la tensione massima con cui alimentarlo non deve superare i 46 Volt. con un accorgimento circuitale è possibile far regolare la tensione fino a livello 0 volt o addirittura fargli generare tensioni negative, è quindi possibile costruire alimentatori duali o di uscite multilivello. La capacità di dissipazione termica del dispositivo è molto alta infatti la resistenza termica è molto bassa, appena 3°C/W tra la giunzione e il corpo metallico a contatto con il dissipatore termico. Tutto il percorso termico tra la giunzione e l'ambiente attraverso il punto di contatto tra corpo metallico del componete e il dissipatore si mantiene sotto i 35°C/w.



Multiwatt15 V

La figura sottostante mostra il pinout (connessione interna dei reofori) dell'integrato L296, in modo particolare viene evidenziato che l'aletta di raffreddamento risulta essere elettricamente collegata al pin 8, quindi il necessario dissipatore esterno risulterà connesso ed equipotenziale a massa.

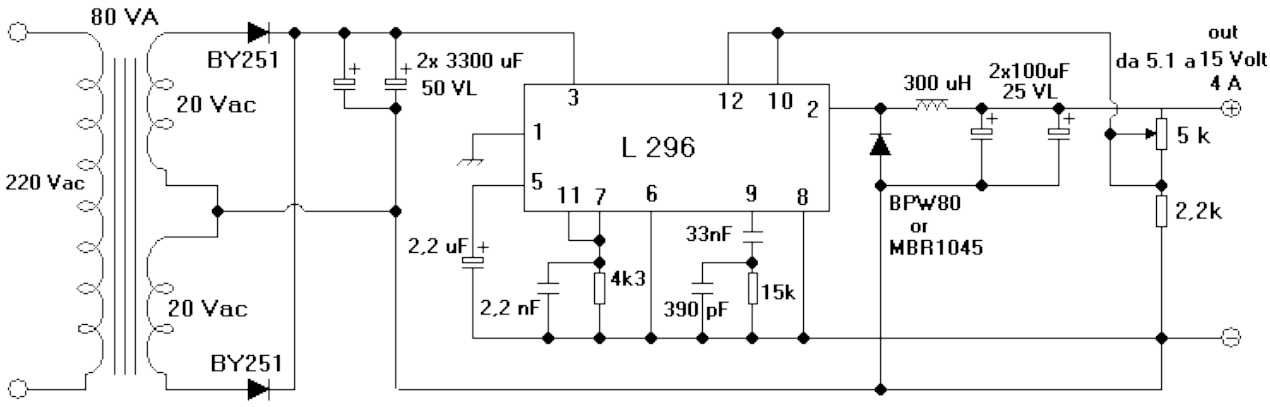


La casa costruttrice SGS-Thompson, mette a disposizione la seguente tabella che chiarisce la funzionalità di ogni pin del circuito integrato.

PIN FUNCTIONS

N°	Name	Function
1	CROWBAR INPUT	Voltage Sense Input for Crowbar Overvoltage Protection. Normally connected to the feedback input thus triggering the SCR when V_{out} exceeds nominal by 20 %. May also monitor the input and a voltage divider can be added to increase the threshold. Connected to ground when SCR not used.
2	OUTPUT	Regulator Output
3	SUPPLY VOLTAGE	Unrergulated Voltage Input. An internal Regulator Powers the L296s Internal Logic.
4	CURRENT LIMIT	A resistor connected between this terminal and ground sets the current limiter threshold. If this terminal is left unconnected the threshold is internally set (see electrical characteristics).
5	SOFT START	Soft Start Time Constant. A capacitor is connected between this terminal and ground to define the soft start time constant. This capacitor also determines the average short circuit output current.
6	INHIBIT INPUT	TTL – Level Remote Inhibit. A logic high level on this input disables the device.
7	SYNC INPUT	Multiple L296s are synchronized by connecting the pin 7 inputs together and omitting the oscillator RC network on all but one device.
8	GROUND	Common Ground Terminal
9	FREQUENCY COMPENSATION	A series RC network connected between this terminal and ground determines the regulation loop gain characteristics.
10	FEEDBACK INPUT	The Feedback Terminal on the Regulation Loop. The output is connected directly to this terminal for 5.1V operation ; it is connected via a divider for higher voltages.
11	OSCILLATOR	A parallel RC networki connected to this terminal determines the switching frequency. This pin must be connected to pin 7 input when the internal oscillator is used.
12	RESET INPUT	Input of the Reset Circuit. The threshold is roughly 5 V. It may be connected to the feedback point or via a divider to the input.
13	RESET DELAY	A capacitor connected between this terminal and ground determines the reset signal delay time.
14	RESET OUTPUT	Open collector reset signal output. This output is high when the supply is safe.
15	CROWBAR OUTPUT	SCR gate drive output of the crowbar circuit.

Attenzione: una caratteristica degli alimentatori switching è quella di non essere operativi in assenza di carico applicato all'uscita, in questo caso l'integrato L296 si accende solo se la corrente minima estratta dall'uscita vale 100 mA.



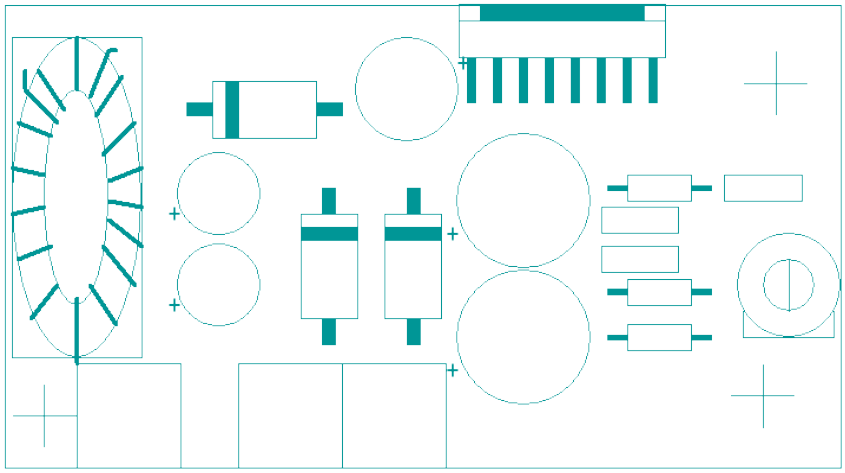
Il trasformatore a presa centrale può essere sostituito da un trasformatore a singolo avvolgimento con tensione al secondario non inferiore a 15 Vac, ma in questo caso si dovranno sostituire i due diodi BY251 con un ponte preferibilmente costituito con quattro essi.
Nello schema elettrico vi sono due punti nei quali si deve procedere a un dimensionamento, il punto di regolazione e il punto reazione induttiva costituito dall'induttanza da 300 uH. Per la regolazione da 5,1 fino a 15 d'uscita la serie del potenziometro da 5K con la resistenza da 2,2k è corretta, ma nel caso si volesse realizzare una versione a tensione fissa può sostituire il gruppo con una serie dimensionata come in tabella.

Valore delle resistenze del partitore per Vo fisse		
Vo	R a massa	R a +Vcc
12 Volt	4,7 KΩ	6,2 KΩ
15 Volt	4,7 KΩ	9,1 KΩ
18 Volt	4,7 KΩ	12 KΩ
24 Volt	4,7 KΩ	18 KΩ

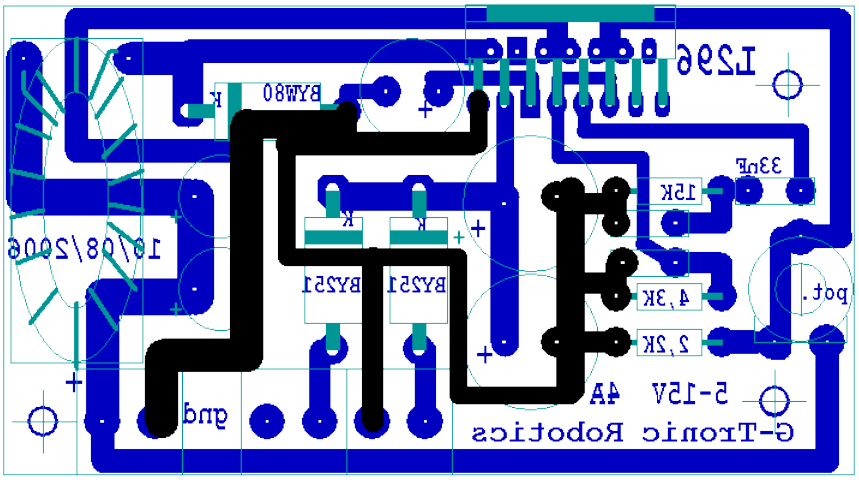
Il dimensionamento dell'induttanza posta in serie all'uscita procede secondo la tabella riportata che utilizza sezioni di filo di rame standard e core toroidali metallici di serie. seguendo questi suggerimenti otterrete il valore di 300 uH richiesti dal progetto.

Tipo di core	**n° avvolgimenti**	**sezione filo**	**spazio tra i fili**
Magnetics 58930 - A2MPP	43	1,0 mm	-
Thomson GUP 20x16x7	65	0.8 mm	1 mm
Siemens EC35/17/10 (b6633&-G0500-X127	40	2 x 0,8 mm	-
VOGT 250 uH bobina Toroidale, numero d'ordine 5730501800 (standard)			

Il circuito stampato che alloggia il circuito è di dimensioni abbastanza ridotte, circa 80mmX50mm, nell'immagine sottostante è rappresentato il layout dei componenti.



come si può vedere il circuito integrato L296 è come di consueto posizionato in posizione tale da poter essere fissato al dissipatore di calore. Sul lato sinistro è rappresentato l'induttanza L1, avvolta su un nucleo toroidale con un numero di spire e sezione del filo indicato nella tabella precedentemente riportata. Sul lato destro si nota la presenza di un trimmer che in realtà non verrà montato perchè sostituito dal potenziometro di regolazione della tensione d'uscita che sarà fissato sul frontalino del contenitore.

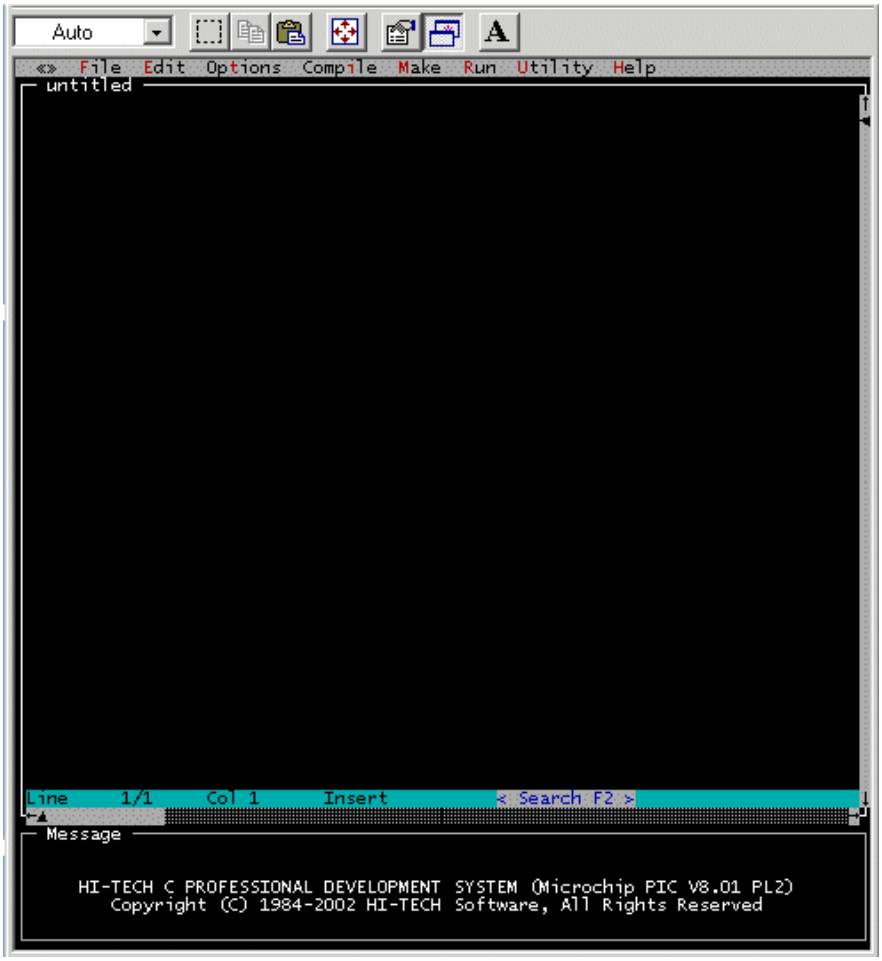


il disegno sovrastante mostra lo sbroglio del circuito eseguito con il FidoCad, in nero sono state evidenziate le piste che costituiscono la massa. Il primo morsetto a vite sulla sinistra è l'uscita positiva regolata. I primi tre morsetti da destra sono l'ingresso ed a essi andranno collegati gli avvolgimenti del secondario del trasformatore con l'avvertenza di collegare al centro la presa centrale di esso.
Viste le ridotte dimensioni in una basetta fotosensibile di misura standard otteniamo ben quattro esemplari dell'alimentatore.

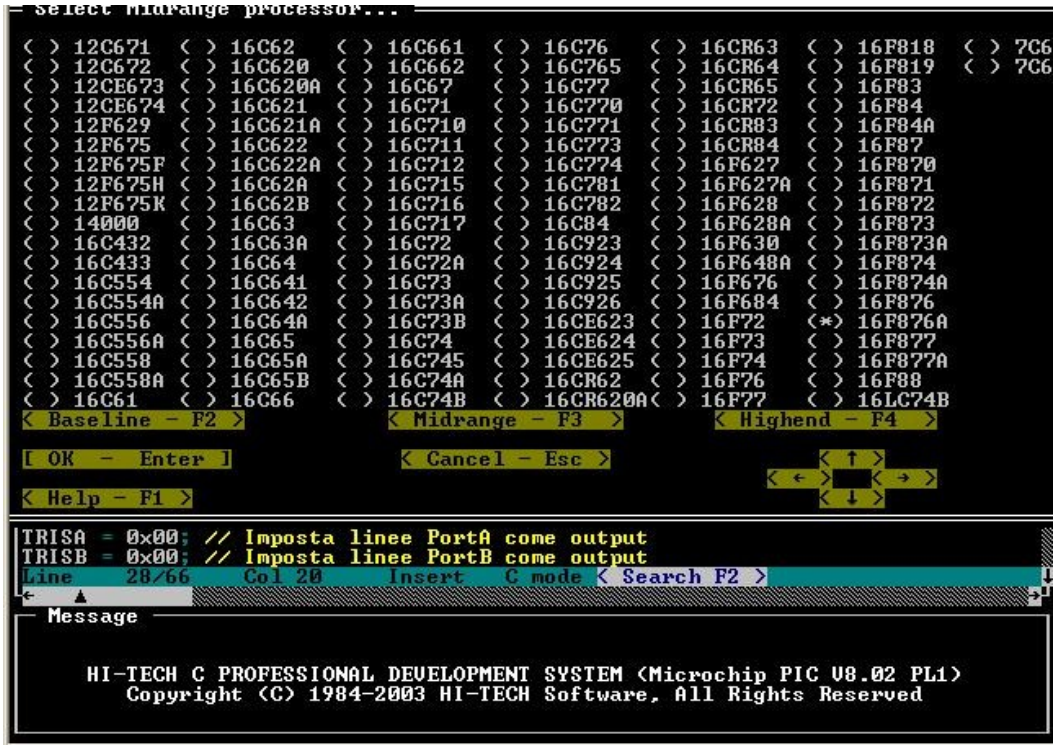
Nella foto un esemplare dell'alimentatore di potenza, sono messe in evidenza le dimensioni consigliate del dissipatore, il diodo di potenza e il nucleo torroidale. Si nota la ridotta dimensione dei condensatori elettrolitici grazie alla tecnica costruttiva switching.

Programmazione in C del MicroPic

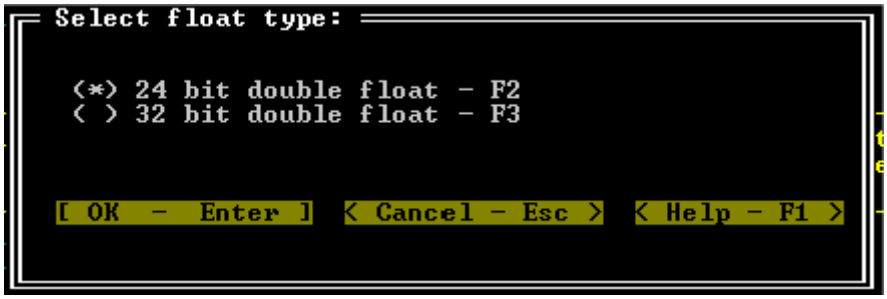
Nell'immagine vediamo l'editor del PICLITE. Funziona sotto DOS ma funziona anche in finestra in ambiente windows. I percorsi di include non dovranno essere più lunghi di 64 caratteri quindi è bene creare una cartella "Cfiles" per i sorgenti direttamente sulla radice del disco.



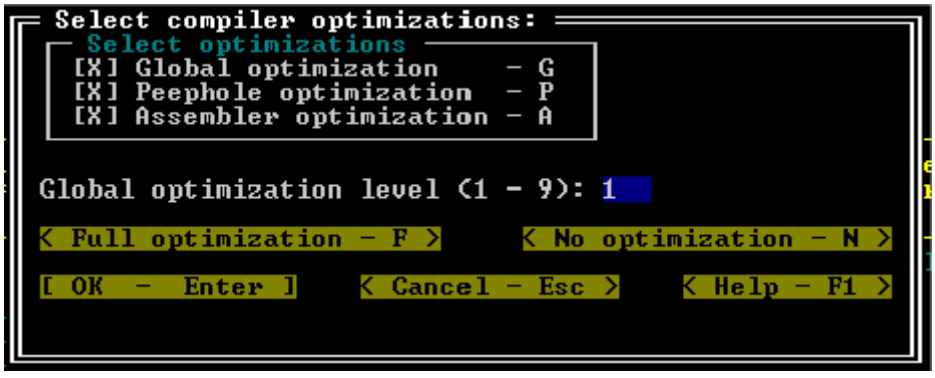
Nel passaggio successivo viene chiesto di selezionare la famiglia del PIC e successivamente lo specifico dispositivo. Il PIC16F876 usato nel progetto RobotHand V2 appartiene alla feglia medium range.



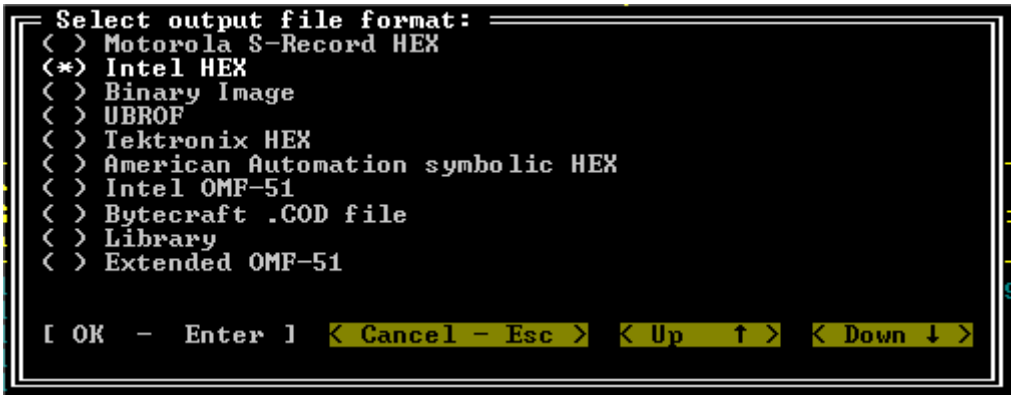
Nel progetto non vengono eseguiti calcoli in virgola mobile quindi disabilitiamo pure il calcolo a 32 bit.



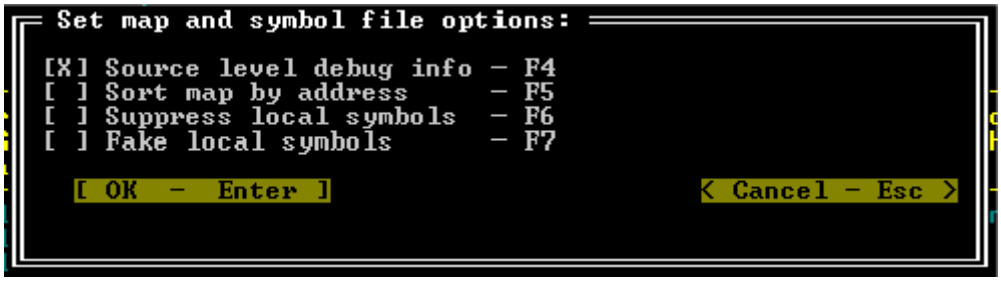
Impostiamo la modalità in cui il compilatore ottimizzerà il codice, lasciamo pure quella proposta di default.



Produciamo un file .hex inseribile nel pic di tipo standard intel Hex, scieremo sempre questo per tutti i PIC, dovremmo comportarci diversamente se usiamo processori di diverse case costruttrici come ad esempio la Motorola.



Il livello e la modalità di debug sono le ultime informazioni che ci verranno richieste.
Conviene lasciare anche qui l'impostazione proposta di default.



A questo punto siamo pronti per assemblare il codice C e ottenere il codice HEX che ovviamente non sarà più comprensibile se aperto con qualsiasi editor.
Troveremo il file nella cartella da cui è partita la compilazione, salvo diverse specificazioni del programmatore.
Possiamo ora aprire questo file con il downloader se il nostro pic contiene già il bootloader, oppure con un software abbinato ad un programmer e riversarlo nella memoria del PIC.

Codice sorgente del programma di controllo dei 14 servomotori.

```
/*
 *     RobotHand V 2
 *  IAS-LAB Giugno 2008
*
*/

#include "pic.h"
#include "delay.c"

static bit Servo1                       @ ((unsigned)&PORTB*8+0);       // R/W
static bit Servo2                       @ ((unsigned)&PORTB*8+1);       // R/W
static bit Servo3                       @ ((unsigned)&PORTB*8+2);       // R/W
static bit Servo4                       @ ((unsigned)&PORTB*8+3);       // R/W
static bit Servo5                       @ ((unsigned)&PORTB*8+4);       // R/W
static bit Servo6                       @ ((unsigned)&PORTB*8+5);       // R/W
static bit Servo7                       @ ((unsigned)&PORTB*8+6);       // R/W
static bit Servo8                       @ ((unsigned)&PORTB*8+7);       // R/W

static bit Servo9                       @ ((unsigned)&PORTC*8+0);       // R/W
static bit Servo10                      @ ((unsigned)&PORTC*8+1);       // R/W
static bit Servo11                      @ ((unsigned)&PORTC*8+2);       // R/W
static bit Servo12                      @ ((unsigned)&PORTC*8+3);       // R/W
static bit Servo13                      @ ((unsigned)&PORTC*8+4);       // R/W
static bit Servo14                      @ ((unsigned)&PORTC*8+5);       // R/W


extern void Var_Init(void);  //prototipi
extern void Agg_Pos(void);
extern void Par_ser(void);

unsigned char S1H,S2H,S3H,S4H,S5H,S6H,S7H,S8H,S9H,S10H,S11H,S12H,S13H,S14H;  //variabili
unsigned char S1L,S2L,S3L,S4L,S5L,S6L,S7L,S8L,S9L,S10L,S11L,S12L,S13L,S14L;
unsigned char C1,C2,C3,C4,RxPos,Flags;
int Pos1,Pos2,Pos3,Pos4,Pos5,Pos6,Pos7,Pos8,Pos9,Pos10,Pos11,Pos12,Pos13,Pos14;

main(void)
{
PORTA = 0x00;                           // azzera le porte
PORTB = 0x00;
PORTC = 0x00;

TRISA = 0x00;                           // tutte uscite per evitare interferenze
TRISB = 0x00;                           // uscite per servi
TRISC = 0xf8;                           // bit 6-7 Seriale

Var_Init();                             // init variabili posizione
ADCON1= 0x87;                           // PORTA all digital mode

//Inizializzazione uart
TXSTA= 0x20;                            // TX abilitato
RCSTA= 0X90;                            // Registro RX USART
BRGH = 1;                               // USART High velocity
SPBRG= 25;                              // Baud Rate Generator (25=9600 baud)

//Inizializzazione TIMER1
T1CON   = 0;                            // TIMER 1 resettato
T1CKPS1 = 0,T1CKPS0=0;                  // TIMER1 prescaler = 1
T1OSCEN = 0;                            // TIMER1 external oscillator disable
TMR1CS  = 0;                            // TIMER1 internal clock (fosc/4)
```

```
TMR1ON  = 0;                            // TIMER1 in STOP

//Inizializzazione interrupt
INTCON = 0;                             // Interrupt disabilitato
PIE1   = 0;                             // Interrupt di periferica disabilitati
GIE    = 1;                             // Global Interrupt Enable
PEIE   = 1;                             // Periferal Interrupt Enable
RCIE   = 1;                             // USART Receive Interrupt Enable

for(;;)
 {
 Servo1 = 1;                            // Servo out High
 TMR1H=S1H;                             // Durata impulso
 TMR1L=S1L;
 TMR1ON = 1;                            // TIMER1 ON
  while (TMR1IF==0){}                   // Controllo flag interrupt timer 0
   Servo1 = 0;                          // Servo out Low
   TMR1IF = 0;                          // Ripristino flag Interrupt
   TMR1ON = 0;                          // TIMER 1 STOP

 Servo2 = 1;
 TMR1H=S2H;
 TMR1L=S2L;
 TMR1ON = 1;
  while (TMR1IF==0){}
   Servo2 = 0;
   TMR1IF = 0;
   TMR1ON = 0;

 Servo3 = 1;
 TMR1H=S3H;
 TMR1L=S3L;
 TMR1ON = 1;
  while (TMR1IF==0){}
   Servo3 = 0;
   TMR1IF = 0;
   TMR1ON = 0;

 Servo4 = 1;
 TMR1H=S4H;
 TMR1L=S4L;
 TMR1ON = 1;
  while (TMR1IF==0){}
   Servo4 = 0;
   TMR1IF = 0;
   TMR1ON = 0;

 Servo5 = 1;
 TMR1H=S5H;
 TMR1L=S5L;
 TMR1ON = 1;
  while (TMR1IF==0){}
   Servo5 = 0;
   TMR1IF = 0;
   TMR1ON = 0;

 Servo6 = 1;
 TMR1H=S6H;
 TMR1L=S6L;
 TMR1ON = 1;
  while (TMR1IF==0){}
```

```
   Servo6 = 0;
   TMR1IF = 0;
   TMR1ON = 0;

 Servo7 = 1;
 TMR1H=S7H;
 TMR1L=S7L;
 TMR1ON = 1;
  while (TMR1IF==0){}
   Servo7 = 0;
   TMR1IF = 0;
   TMR1ON = 0;

 Servo8 = 1;
 TMR1H=S8H;
 TMR1L=S8L;
 TMR1ON = 1;
  while (TMR1IF==0){}
   Servo8 = 0;
   TMR1IF = 0;
   TMR1ON = 0;

Servo9 = 1;
 TMR1H=S9H;
 TMR1L=S9L;
 TMR1ON = 1;
  while (TMR1IF==0){}
   Servo9 = 0;
   TMR1IF = 0;
   TMR1ON = 0;

Servo10 = 1;
 TMR1H=S10H;
 TMR1L=S10L;
 TMR1ON = 1;
  while (TMR1IF==0){}
   Servo10 = 0;
   TMR1IF = 0;
   TMR1ON = 0;

Servo11 = 1;
 TMR1H=S11H;
 TMR1L=S11L;
 TMR1ON = 1;
  while (TMR1IF==0){}
   Servo11 = 0;
   TMR1IF = 0;
   TMR1ON = 0;

Servo12 = 1;
 TMR1H=S12H;
 TMR1L=S12L;
 TMR1ON = 1;
  while (TMR1IF==0){}
   Servo12 = 0;
   TMR1IF = 0;
   TMR1ON = 0;


Servo13 = 1;
 TMR1H=S13H;
```

```
 TMR1L=S13L;
 TMR1ON = 1;
  while (TMR1IF==0){}
   Servo13 = 0;
   TMR1IF = 0;
   TMR1ON = 0;


Servo14 = 1;
 TMR1H=S14H;
 TMR1L=S14L;
 TMR1ON = 1;
  while (TMR1IF==0){}
   Servo14 = 0;
   TMR1IF = 0;
   TMR1ON = 0;


 if (Flags==1) Par_ser();

 Agg_Pos();                                  // Sposta Servomotori
 DelayMs(2);                                 // Delay aggiuntivo
 }
}

void
Var_Init(void)
{
S1H= 0xFF;                                   // Servo neutral (1.5ms)
S1L= 0X00;
S2H= 0xFA;
S2L= 0X23;
S3H= 0xFA;
S3L= 0X23;
S4H= 0xFA;
S4L= 0X23;
S5H= 0xFA;
S5L= 0X23;
S6H= 0xFA;
S6L= 0X23;
S7H= 0xFA;
S7L= 0X23;
S8H= 0xFA;
S8L= 0X23;
S9H= 0xFA;
S9L= 0X23;
S10H= 0xFA;
S10L= 0X23;
S11H= 0xFA;
S11L= 0X23;
S12H= 0xFA;
S12L= 0X23;
S13H= 0xFA;
S13L= 0X23;
S14H= 0xFA;
S14L= 0X23;

Pos1 = 500;                                  // Nuova posizione (1.5 ms)
Pos2 = 500;
Pos3 = 500;
Pos4 = 500;
```

```
Pos5 = 500;
Pos6 = 500;
Pos7 = 500;
Pos8 = 500;
Pos9 = 500;
Pos10 = 500;
Pos11 = 500;
Pos12 = 500;
Pos13 = 500;
Pos14 = 500;

RxPos = 0;
Flags  = 0;
}

void
Agg_Pos(void)
{
S1H = (64535-Pos1)/256;
S1L = (64535-Pos1)-S1H*256;

S2H = (64535-Pos2)/256;
S2L = (64535-Pos2)-S2H*256;

S3H = (64535-Pos3)/256;
S3L = (64535-Pos3)-S3H*256;

S4H = (64535-Pos4)/256;
S4L = (64535-Pos4)-S4H*256;

S5H = (64535-Pos5)/256;
S5L = (64535-Pos5)-S5H*256;

S6H = (64535-Pos6)/256;
S6L = (64535-Pos6)-S6H*256;

S7H = (64535-Pos7)/256;
S7L = (64535-Pos7)-S7H*256;

S8H = (64535-Pos8)/256;
S8L = (64535-Pos8)-S8H*256;

S9H = (64535-Pos3)/256;
S9L = (64535-Pos3)-S3H*256;

S10H = (64535-Pos4)/256;
S10L = (64535-Pos4)-S4H*256;

S11H = (64535-Pos5)/256;
S11L = (64535-Pos5)-S5H*256;

S12H = (64535-Pos6)/256;
S12L = (64535-Pos6)-S6H*256;

S13H = (64535-Pos7)/256;
S13L = (64535-Pos7)-S7H*256;

S14H = (64535-Pos8)/256;
S14L = (64535-Pos8)-S8H*256;
}
```

```
void interrupt SerChar(void)    //prototipo ricezione
{
 if (RCREG==0x40 | RxPos > 6)  //se arriva la chiocciola dimensiona la stringa
  {
  C1=0,C2=0,C3=0,C4=0;                              //azzero la stringa
  RxPos=0,Flags=0;
  }
 if (RxPos==1) C1=RCREG;                          //ricezione della stringa comandi
 if (RxPos==2) C2=RCREG;
 if (RxPos==3) C3=RCREG;
 if (RxPos==4) C4=RCREG;
 if (RxPos==5 & RCREG==0x0d) Flags=1; // <CR> terminator
 RxPos++;
}

void
Par_ser(void)                                              // Analisi comandi
{
 Flags=0;
 switch (C1)
  {
 case 0x41:    //carattere A
  Pos1 = C2*4;                                    //Pos1 assume il valore C2 trasmesso con A
  break;
 case 0x42:                                       //carattere B
  Pos2 = C2*4;
  break;
 case 0x43:                                       //carattere C
  Pos3 = C2*4;
  break;
 case 0x44:                                       //carattere D
  Pos4 = C2*4;
  break;
 case 0x45:                                       //carattere E
  Pos5 = C2*4;
  break;
 case 0x46:                                       //carattere F
  Pos6 = C2*4;
  break;
 case 0x47:                                       //carattere G
  Pos7 = C2*4;
  break;
 case 0x48:                                       //carattere H
  Pos8 = C2*4;
  break;
 case 0x49:                                       //carattere I
  Pos9 = C2*4;
  break;
 case 0x4A:                                       //carattere J
  Pos10 = C2*4;
  break;
 case 0x4B:                                       //carattere K
  Pos11 = C2*4;
  break;
 case 0x4C:                                       //carattere L
  Pos12 = C2*4;
  break;
 case 0x4D:                                       //carattere M
  Pos13 = C2*4;
  break;
 case 0x4E:                                       //carattere N
```

```
  Pos14 = C2*4;
  break;
 }
}
```

Codice sorgente del programma di controllo servo da analogici.

Il prossimo programma abilita la porta A in modalità analogica, acquisisce il segnale e posiziona 5 servomotori di conseguenza.
Il programma serve per la modalità di controllo **mirror**, ovvero dei potenziometri di tipo spider nascosti nel dorso della buona danno il comando di movimento alla protesi.

```
/*
 * controllo di 5 servo tramite 5 potenziometri
 * I movimenti delle dita della mano sinistra vengono copiati
 * sulla protesi della mano destra.
 * I potenziometri sono nascosti nel dorso di un guanto
 * nella mano buona.
 */

#include                    <pic.h>
#include                    "delay.h"
#include                    "delay.c"

#define                     ON     1
#define                     OFF    0
#define                     SERVO1  RB5       //indice
#define                     SERVO2  RB4       //medio
#define                     SERVO3  RB3       //anulare
#define                     SERVO4  RB2       //mignolo
#define                     SERVO5  RB1       //pollice

//Routine principale...

int leggi_ad(char canale);

main()
{
                            unsigned intvalore1;
                            unsigned intvalore2;
                            unsigned intvalore3;
                            unsigned intvalore4;
                            unsigned intvalore5;
                            unsigned intx;

                            TRISB=0x00;    //PORTB tutte uscite
                            TRISA=0xFF;    //PORTA tutti ingressi per gli A/D

                            /*
                                ADCON1
                                1 - ADFM Risultato giustificato a destra (0 SX)
                                0 - non usato
                                0 - non usato
                                0 - non usato
                                0 - PCFG3 Tutti ingressi analogici
                                0 - PCGG2 Vref
                                0 - PCFG1 -Vdd
                                0 - PCFG0 -Vss
                            */
```

```
/*
    ADCON0
    0 - ADCS1 Frequenza oscillatore
    1 - ADCS0 Fosc/8 (1/4Mhz=0.23uS .25*8=2uS minimo 1.6uS)
    0 - CHS2
    0 - CHS1 Seleziona l'ingresso dell'ADC
    0 - CHS0
    0 - ADGO Mettere a 1 per inizio conversione 0 -Fine convers.
    0 - non usato
    0 - ADON ON/OFF ADC (1 ON) Quando ON dissipa potenza
*/

ADCON1 = 0b10000000;

for(;;) {
                        valore1= leggi_ad(0);
                        valore2= leggi_ad(1);
                        valore3= leggi_ad(2);
                        valore4= leggi_ad(3);
                        valore5= leggi_ad(4);

            // Crea l'impulso per il servo 1
        SERVO1=ON;
        DelayUs(150);
                        valore1=valore1/5;
        for(x=0;x<=valore1;x++)
              {
              }
        SERVO1=OFF;

            // Crea l'impulso per il servo 2
        SERVO2=ON;
        DelayUs(150);
                        valore2=valore2/5;
        for(x=0;x<=valore2;x++)
                                {
              }
        SERVO2=OFF;

            // Crea l'impulso per il servo 3
        SERVO3=ON;
        DelayUs(150);
                        valore3=valore3/5;
        for(x=0;x<=valore3;x++)
                                {
              }
        SERVO3=OFF;

            // Crea l'impulso per il servo 4
        SERVO4=ON;
        DelayUs(150);
                        valore4=valore4/5;
        for(x=0;x<=valore4;x++)
                                {
              }
        SERVO4=OFF;

            // Crea l'impulso per il servo 5
        SERVO5=ON;
        DelayUs(150);
                        valore5=valore5/5;
```

```
                                        for(x=0;x<=valore5;x++)
                                                        {
                                        }
                                        SERVO5=OFF;

                                        DelayMs(10);
                                                }
}

int leggi_ad(char canale)
{
int valore;

ADCON0 = (canale << 3) + 0xC1;          // enable ADC, RC osc.

DelayUs(10);                            //Ritardo per dare modo all'A/D di stabilizzarsi

ADGO = 1;                               //Fa partire la conversione

while(ADGO)
continue;                               //Attende che la conversione sia completa

valore=ADRESL;                          //Parte bassa del risultato
valore= valore + (ADRESH<<8);           //Parte alta del risultato

return(valore);
}
```

Programma di test della comunicazione seriale

Il prossimo file sorgente una volta compilato e riversato nel pic abiliterà la porta seriale e inizializza una comunicazione. Sul lato del pc con sistema operativo windows, verrà lanciato "hyperterminal" impostato per com 1, 9600 baud , nessun controllo di flusso.
Dopo aver lanciato hyperterminal e avviata la sessione, avverà l'accesso alla demoboard grazie alla presenza del chip MAX232 in essa presente. La connessione deve essere effettuata tramite una prolunga seriale, ovvero un cavo non crossato nei pin 2 (TX) e 3 (RX), distinguibile da un nullmodem perché i due DB9 sono femmina nel lato PC e maschio dal lato della scheda controllo. Dopo 2 o 3 secondi dall'accensione, all'interno della finestra dell' hyperterminal apparirà la scritta "Test seriale v0.2". Da questo momento quello che viene digitato sulla tastiera apparirà nella finestra di hyperterminal. Se ne deduce che il programma inserito
nel pic non fa altro che ritrasmettere quello che riceve da PC.

```
// ------------------------------------------------------------------
// File: TUart.c -
//     Com Test
 //IAS-LAB Giugno 2008
// Descrizione: Esempio di comunicazione seriale con il PC -
// Verifiche: Prove eseguite con 16F876 20Mhz -
// Note: Vedere Uart.c per le routine di gestione seriale -
// ------------------------------------------------------------------

#include <pic.h>
#include "delay.c"
#include "macro.h"
#include "ascii.c"
#include "uart.c"

main(void) {
char c;
unsigned char clock;
unsigned long baudrate;
char string[5];
unsigned char rc;

ADCON0 = 0;
ADCON1 = 7;
PORTB = 0x00; // Azzera linee PortA
PORTB = 0x00; // Azzera linee PortB
PORTC = 0x00; // Azzera linea PortC
TRISA = 0x00; // Imposta linee PortA come output
TRISB = 0x00; // Imposta linee PortB come output
TRISC = 0x80; // Imposta linee PortC come output
              // tranne RC7/Rx come input

DelayMs(250);
DelayMs(250);
DelayMs(250);
DelayMs(250);

// Inizializza la seriale
clock = 20; //inserire qui il valore del quarzo
baudrate = 9600;
rc = UartInit(clock, baudrate, UART_CFG_BITSTOP_1);
int2dec(rc, string);
DelayMs(250);
UartPutch('-');
UartPutch('-');
UartPuts(string);
UartPutch('-');
UartPutch('-');
UartPuts("Test seriale v0.2");
while (TRUE)
{
c = UartGetch();
UartPutch(c);
}
}
```

Programmazione in Visual Basic dell'interfaccia controllo manuale da PC

La programmazione dell'interfaccia per PC Windows in visual Basic 6, una volta compilata, da origine a un file eseguibile di circa 56KByte visibile nella prima riga dell'elenco nell'immagine sottostante. Questo file eseguibile costituisce la nostra applicazione finale e potrà essere fornito in maniera svincolata dagli altri files che compaiono nella medesima directory di programmazione. Visual Basic gestisce i files che compongono il programma tramite un elenco software denominato "progetto", fisicamente visibile nella terza riga. Quando si esegue la compilazione e la conseguente creazione dell'applicazione eseguibile verrà generato un output che porterà per default lo stesso nome del file progetto. Se nel PC è installato VB6 basterà eseguire un doppio click sul file progetto affinché il sistema si configuri correttamente per proseguire nella programmazione o modifica.

MotorControl	56 KB	Applicazione	02/07/2008 23.17
MotorControl	28 KB	Visual Basic Form File	02/07/2008 23.16
MotorControl	1 KB	Visual Basic Project	02/07/2008 23.17
MotorControl	1 KB	Visual Basic Project ...	02/07/2008 23.24
PortComunication	6 KB	Visual Basic Form File	15/06/2008 17.09

La configurazione della porta di comunicazione avviene tramite il Form "PortCommunication", visibile nella lista dell'immagine in ultima posizione. È un modulo standard piuttosto piccolo, solo 6Kbyte, a cui si accede semplicemente eseguendo un drag and drop nel form principale. La presenza della routine di comunicazione seriale è rilevata dall'icona a forma di telefono.

Ispezionandone il codice troviamo quanto segue, si nota che per default la porta seriale viene impostata ad un boudrate di 9600 Kbit per secondo, nessuna parità, 8 bit di dati, 1 bit di stop.

```
Private Sub Command1_Click()
On Error Resume Next
 If ProcessON = False Then
    NumPorta = Combo1.Text
    ProcessON = True
    Command1.BackColor = &HC000&
    Command1.Caption = "STOP"
    MSComm1.CommPort = NumPorta
    MSComm1.Settings = "9600,n,8,1" ' Le impostazioni della seriale
    MSComm1.PortOpen = True ' Apriamo la porta.
 Else
    ProcessON = False
    ProcessStepON = False
    Command1.BackColor = &H8000000F
    Command2.BackColor = &H8000000F
    Command1.Caption = "START"
    MSComm1.PortOpen = False ' Apriamo la porta.
  End If
End Sub
```

La classe che segue invia alla seriale la stringa di comando per l'accensione e lo spegnimento del LED di controllo operatività connesso al pin RA0 del pic. Si veda il paragrafo "**Scheda con microcontrollore"** per capirne meglio lo scopo.

```
Private Sub Form_Initialize()
On Error Resume Next
   CharStart = Chr(64)
   CharReturn = Chr(13)
   CharLedON = Chr(81)
   CharLedOFF = Chr(80)
   CharLedPosition = "123"
End Sub
```

L'assegnazione CharLedOFF = Chr(80) imposta la variabile da inviare alla seriale con il carattere 'P' utilizzato per lo spegnimento del led di test, mentre CharLedON = Chr(81) assegna alla medesima variabile il carattere aschii 'Q' per la riaccensione del medesimo. La variabile CharLedPosition viene inizializzata al valore testuale "123", dove per testuale si intende ovviamente char 1, char 2, char 3. L'insieme dei caratteri, con l'aggiunta del "ret" completa la stringa a 6 elementi char come si aspetta di ricevere nel buffer dell'UART il programma .hex che sta girando nel PIC.

Nel successivo codice vengono definiti gli 59pider di posizionamento dei motori.

```
Private Sub Slider1_Change(Index As Integer)
    If ProcessON = True And ProcessStepON = False Then
 Select Case Index
   Case Is = 0
      ChangedValue1 = True
   Case Is = 1
      ChangedValue2 = True
   Case Is = 2
      ChangedValue3 = True
   Case Is = 3
      ChangedValue4 = True
   Case Is = 4
      ChangedValue5 = True
   Case Is = 5
      ChangedValue6 = True
   Case Is = 6
      ChangedValue7 = True
   Case Is = 7
      ChangedValue8 = True
   Case Is = 8
      ChangedValue9 = True
   Case Is = 9
      ChangedValue10 = True
   Case Is = 10
      ChangedValue11 = True
   Case Is = 11
      ChangedValue12 = True
   Case Is = 12
      ChangedValue12 = True
   Case Is = 13
      ChangedValue14 = True
   End Select
End If
```

```
End Sub
```

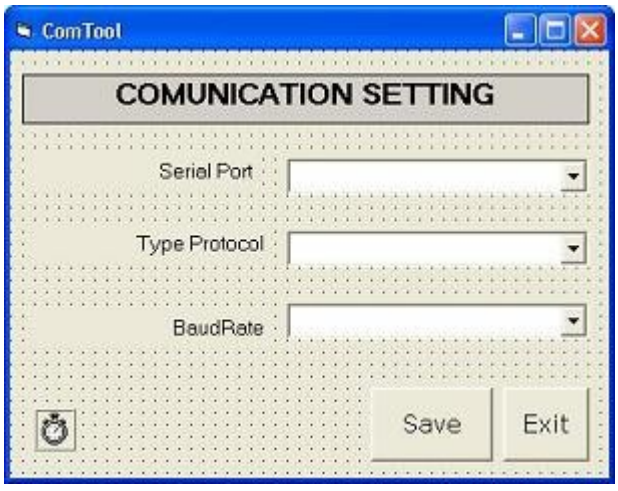
Ad ogni cursore viene assegnata una variabile intera di index che sarà poi convertita in un elemento della tabella aschii, per questo motivo è importante che il cursore non assegni mai una variabile maggiore o uguale a 256 perché il tentativo di conversione in un carattere di questo valore intero comporterebbe il crasch del sistema e l'uscita immediata dell'applicativo.
Prima di definire il codice che cura l'effettivo invio del comando di posizionamento abbinato ad ogni cursore e al corrispettivo motore, viene definito un timer con lo scopo di eseguire una scansione ciclica di tutti i motori presenti "polling".

```
Private Sub Timer1_Timer()
'On Error Resume Next
   Dim Value As Long
   Dim StringCommand As String

    If ProcessON = True And ProcessStepON = False Then
```

Al controllo del timer segue il codice di definizione e comunicazione dei parametri a ogni singolo motore. Questo verrà ripetuto per ciascuno dei 14 motori.

```
'********** Motor1 ******************
If ChangedValue1 = True Then
      Value = Slider1(0).Value
      CharIndex = Chr(65)
      CharPosition = Chr(Value)
      StringCommand = CharStart & CharIndex & CharPosition & "0" & "0" & CharReturn
      MSComm1.Output = StringCommand
      StatusBar1.Panels(1).Text = CharIndex
      StatusBar1.Panels(2).Text = CharPosition
      StatusBar1.Panels(3).Text = StringCommand
      ChangedValue1 = False
      Exit Sub
End If
```

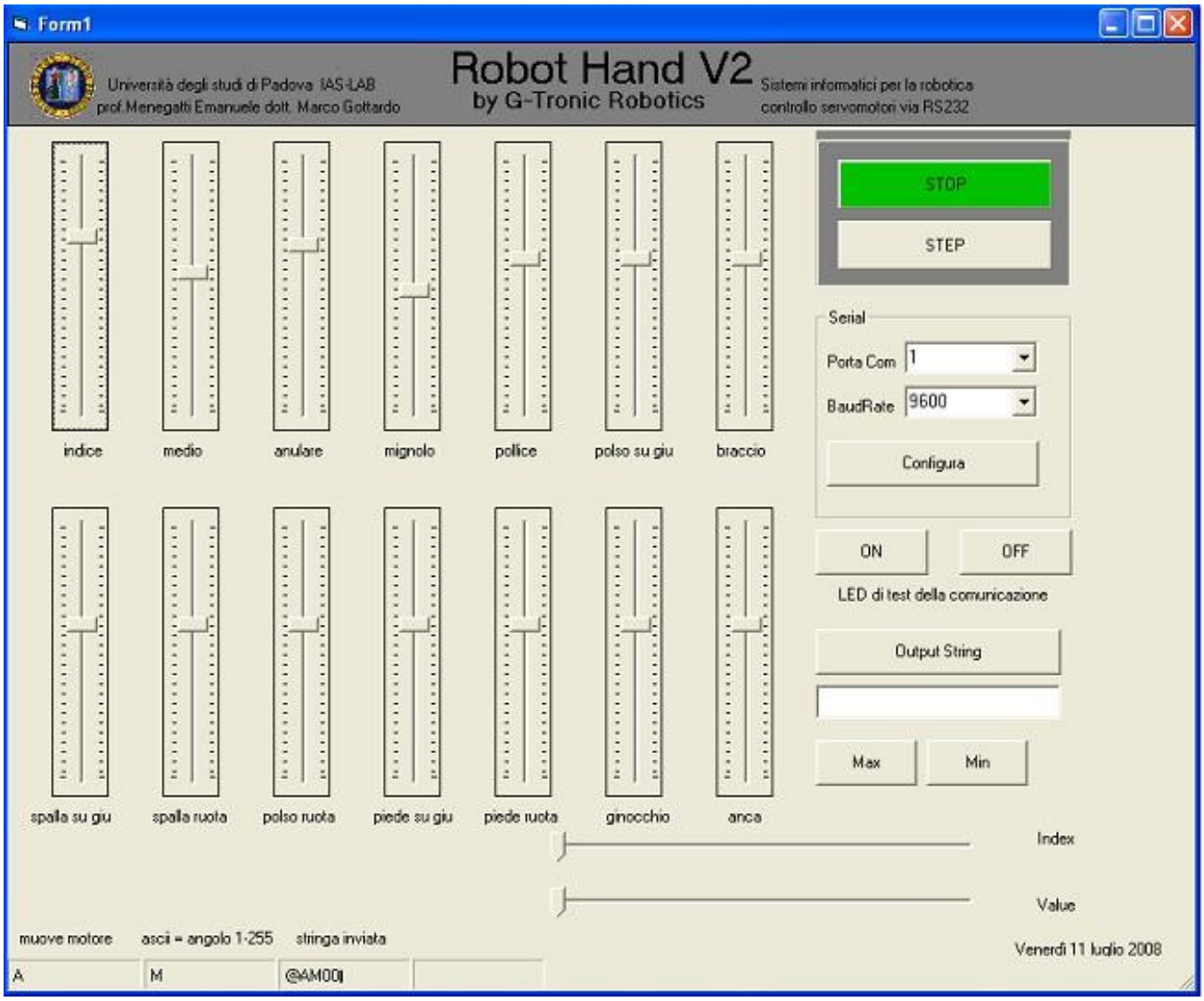
Le impostazione per la comunicazione sono fornite nel modulo "portcomunication" che da origine al seguente form.



Benché questo "form" esista e sia ben presente all'atto della programmazione non comparirà mai durante l'esecuzione dell'applicativo.

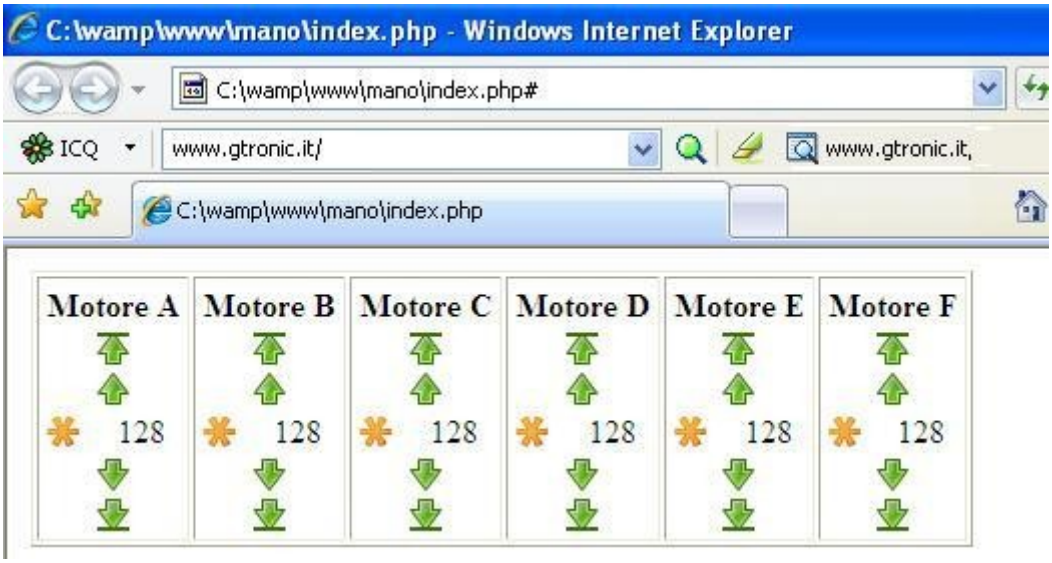
ATTENZIONE:

Non lanciare due istanze del programma, oppure un altro programma di comunicazione che usa la porta seriale



Interfaccia Web di telecontrollo

Nel caso si rendesse necessario un intervento esterno da parte di un operatore abilitato è possibile telecontrollare l'arto via web tramite un'interfaccia sviluppata con una variante evoluta del PHP, il symfony. I casi in cui potrebbe essere necessario un intervento di telecontrollo vanno dal malessere grave della persona in cui è installata protesi, al soddisfare una semplice richiesta di aiuto nella fase di traning per imparare ad utilizzare l'arto.
L'immagine è uno screenshot di una versione beta del software.



Nell'interfaccia è anche possibile visualizzare l'immagine di una webcam che mostra la posizione dell'arto durante il movimento telecontrollato.

Il codice PHP per realizzare l'interfaccia è suddiviso in tre files principali:

- index.php
- muovi.php
- php_serial.class.php

come per tutte le applicazioni web il punto di accesso è il file index, qui riportato con estensione php perché utilizza le primitive di questo linguaggio. Per poter essere lanciato bisogna che la macchina host sia impostata allo scopo. E' necessario che sia installato principalmente un server WEB che nel nostro caso è costituito da WAMP, scaricabile gratuitamente.

Codice sorgente del file index.php

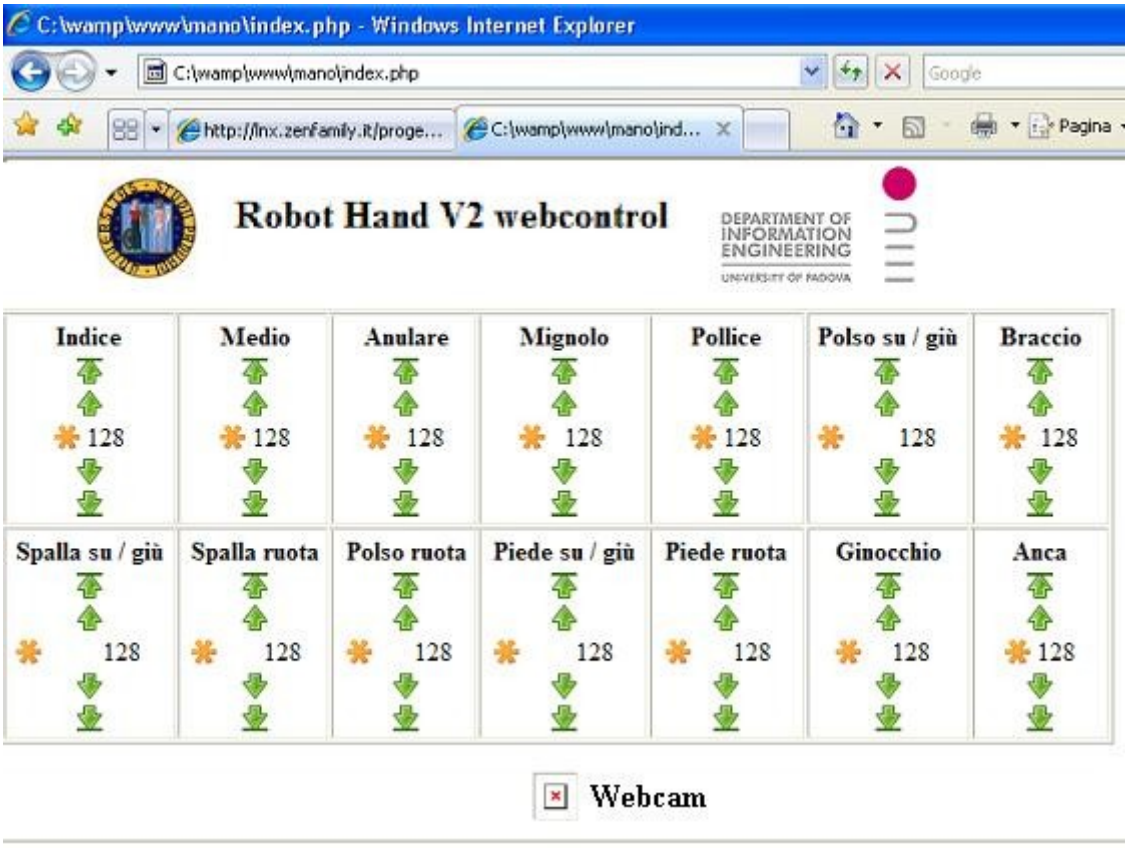
```
<script src="prototype.js"></script>
<script>
		function muovi(motore,direzione)
		{
			var pars = 'motore=' + motore + '&direzione=' + direzione + '&stato=' + $('motore'+motore).innerHTML;
			var myAjax = new Ajax.Request( 'muovi.php', { method: 'get', parameters: pars, onComplete: function(transport){
   var response = transport.responseText || "no response text";
   var mot= 'motore'  + motore;
   $(mot).innerHTML =response;
} });
  }
</script>


<table align=center border=1>
<tr>
<td>



<table>
<tr><td colspan=2 align=center><strong>Motore A</strong></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('A','top')"><img src='go-top.png' border=0></a></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('A','dx')"><img src='go-up.png' border=0></a></td></tr>
<tr><td><a href='#' onClick="javascript:muovi('A','zero')"><img src='asterisk_orange.png' border=0></a></td><td><div id='motoreA'
align=center>128</div></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('A','sx')"><img src='go-down.png' border=0></a></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('A','bottom')"><img src='go-bottom.png' border=0></a></td></tr>
</table>

</td>
<td>

<table>
<tr><td colspan=2 align=center><strong>Motore B</strong></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('B','top')"><img src='go-top.png' border=0></a></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('B','dx')"><img src='go-up.png' border=0></a></td></tr>
<tr><td><a href='#' onClick="javascript:muovi('B','zero')"><img src='asterisk_orange.png' border=0></a></td><td><div id='motoreB'
align=center>128</div></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('B','sx')"><img src='go-down.png' border=0></a></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('B','bottom')"><img src='go-bottom.png' border=0></a></td></tr>
</table>

</td>
<td>


<table>
<tr><td colspan=2 align=center><strong>Motore C</strong></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('C','top')"><img src='go-top.png' border=0></a></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('C','dx')"><img src='go-up.png' border=0></a></td></tr>
<tr><td><a href='#' onClick="javascript:muovi('C','zero')"><img src='asterisk_orange.png' border=0></a></td><td><div id='motoreC'
align=center>128</div></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('C','sx')"><img src='go-down.png' border=0></a></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('C','bottom')"><img src='go-bottom.png' border=0></a></td></tr>
</table>

</td>
<td>


<table>
<tr><td colspan=2 align=center><strong>Motore D</strong></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('D','top')"><img src='go-top.png' border=0></a></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('D','dx')"><img src='go-up.png' border=0></a></td></tr>
<tr><td><a href='#' onClick="javascript:muovi('D','zero')"><img src='asterisk_orange.png' border=0></a></td><td><div id='motoreD'
align=center>128</div></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('D','sx')"><img src='go-down.png' border=0></a></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('D','bottom')"><img src='go-bottom.png' border=0></a></td></tr>
</table>

</td>
<td>
```

```
<table>
<tr><td colspan=2 align=center><strong>Motore E</strong></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('E','top')"><img src='go-top.png' border=0></a></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('E','dx')"><img src='go-up.png' border=0></a></td></tr>
<tr><td><a href='#' onClick="javascript:muovi('E','zero')"><img src='asterisk_orange.png' border=0></a></td><td><div id='motoreE'
align=center>128</div></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('E','sx')"><img src='go-down.png' border=0></a></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('E','bottom')"><img src='go-bottom.png' border=0></a></td></tr>
</table>

</td>
<td>


<table>
<tr><td colspan=2 align=center><strong>Motore F</strong></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('F','top')"><img src='go-top.png' border=0></a></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('F','dx')"><img src='go-up.png' border=0></a></td></tr>
<tr><td><a href='#' onClick="javascript:muovi('F','zero')"><img src='asterisk_orange.png' border=0></a></td><td><div id='motoreF'
align=center>128</div></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('F','sx')"><img src='go-down.png' border=0></a></td></tr>
<tr><td colspan=2 align=center><a href='#' onClick="javascript:muovi('F','bottom')"><img src='go-bottom.png' border=0></a></td></tr>
</table>

</td>
</tr>
</table>

<center>
<p> 
<img src='video.jpg'>
```

Codice sorgente del modulo muovi.php

```
<?php

$max=255;
$min=0;

$utilizzaSeriale=true;
$motore=$_REQUEST['motore'];
$direzione=$_REQUEST['direzione'];
$stato=$_REQUEST['stato'];
if ($direzione=='top') {
  $stato=$max;
  // muovi motore alla posizione $max;

}
else {
if ($direzione=='bottom') {
  $stato=$min;
  // muovi motore alla posizione $min;
} else {
if ($direzione=='zero') {
  $stato=round(($max-$min)/2);
  // muovi motore alla posizione 0;
} else {
$stato=($direzione=='dx')?($stato+1):($stato-1);
if ($stato>$max) {
  $stato=$max;
  }
if ($stato<$min) {
  $stato=$min;
  }

  // muovi motore alla posizione $stato

}

}

}

if ($utilizzaSeriale) muoviMotore("@".$motore.chr($stato)."00\r");
echo $stato;
```

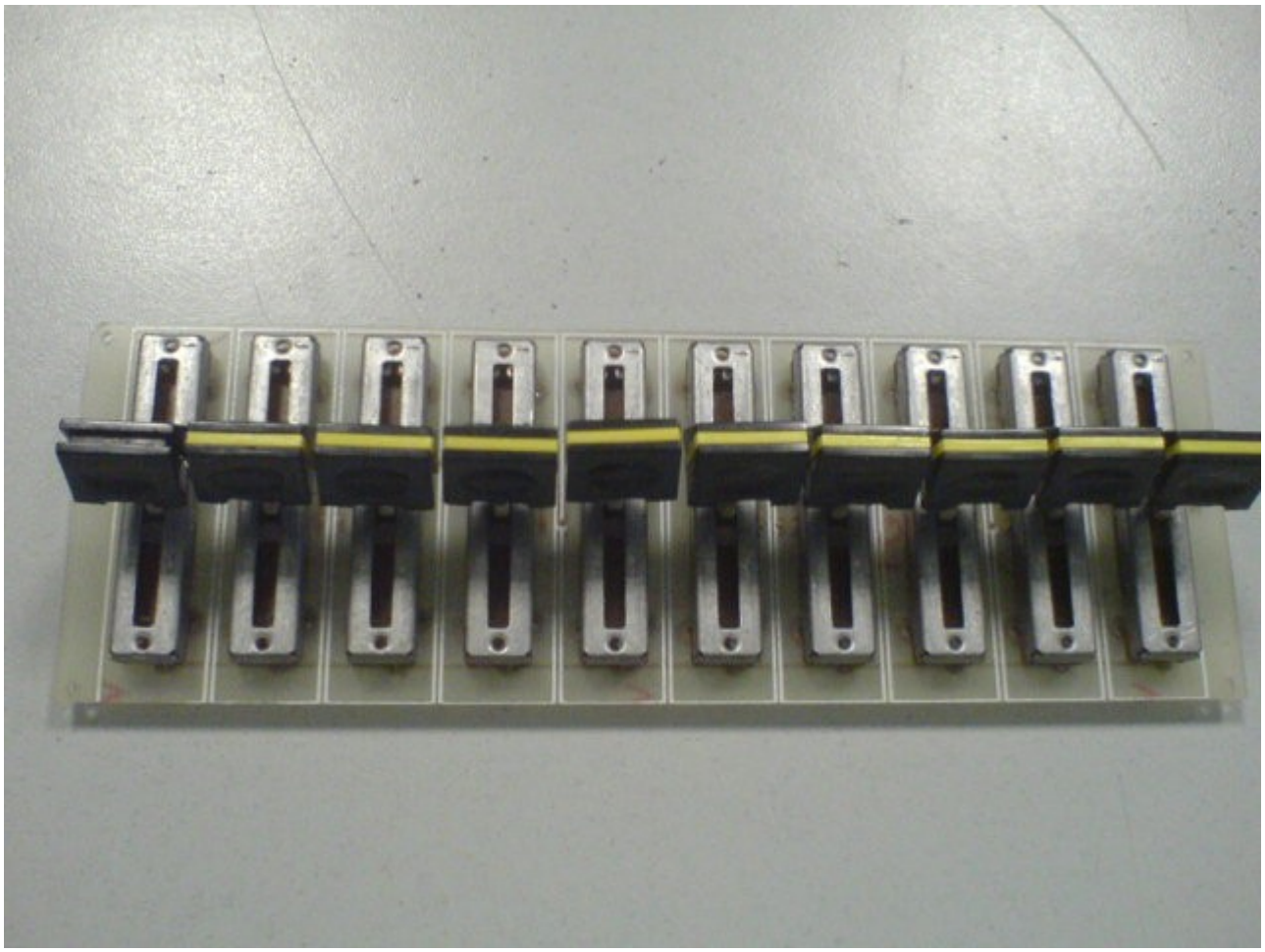
```
function muoviMotore() {
   require("php_serial.class.php");
   $serial = new phpSerial();
   $serial->deviceSet("COM1");
   $serial->confBaudRate(9600); //Baud rate: 9600
   $serial->confParity("none");  //Parity (this is the "N" in "8-N-1")
   $serial->confCharacterLength(8); //Character length
   $serial->confStopBits(1);  //Stop bits (this is the "1" in "8-N-1")
   $serial->confFlowControl("none");
   $serial->deviceOpen();
   $serial->sendMessage("N1\r");
   $serial->deviceClose();
}


?>
```



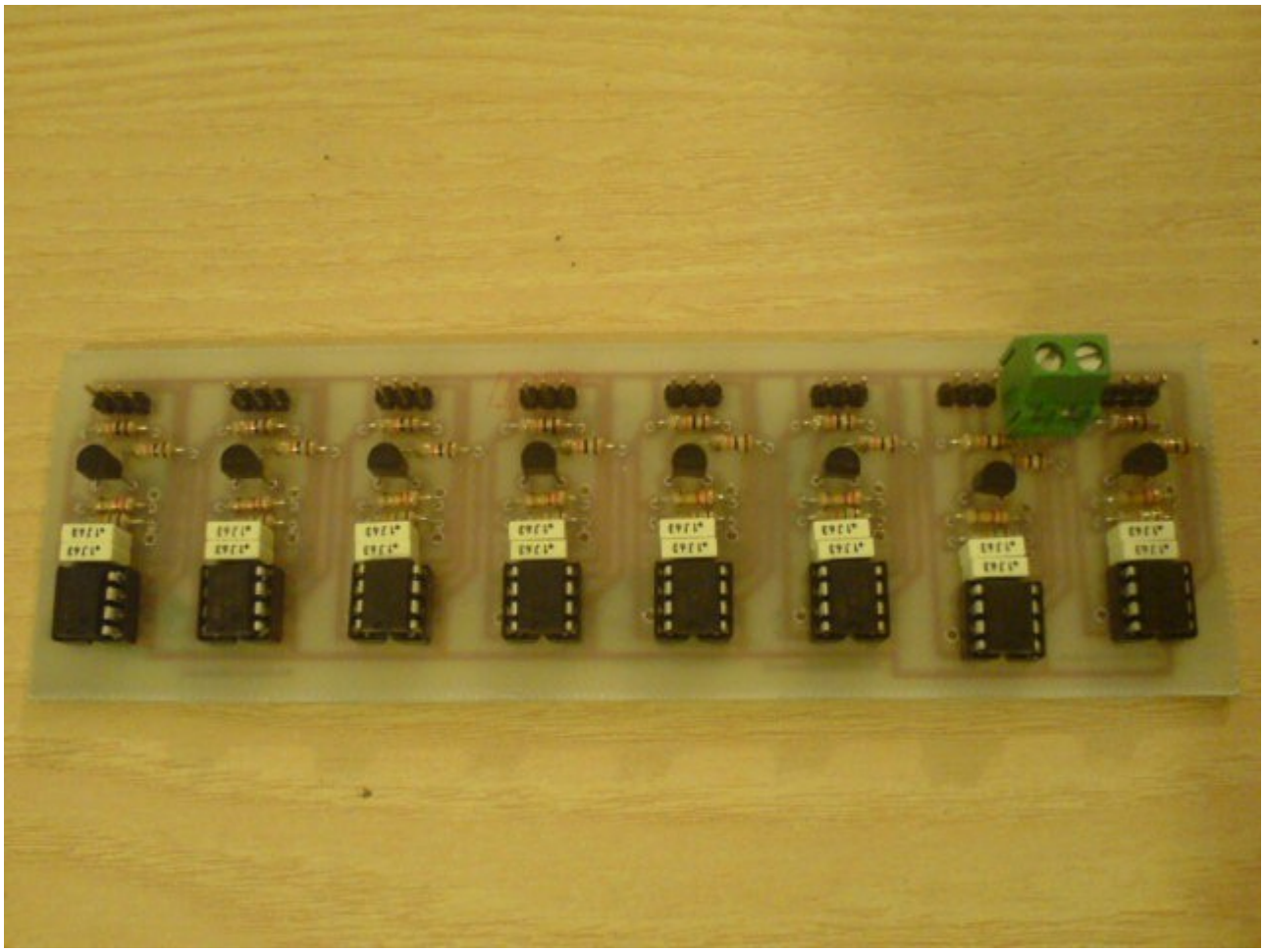
Una volta completato il codice del file index.php l'interfaccia appare come come nell'immagine sopra. Di fondamentale importanza risulta essere la presenza della webcam (nell'immagine è spenta) che mostra via web la posizione attuale dell'arto controllato.
La webcam nella versione attuale risulta essere temporizzata con step di un secondo, tempo in cui avviene il refresch del caricamento dell'immagine associato a quell'area, ma facendo affidamento su una trasmissione web rapida e sicura si potrà optare per l'acqusizione continua dell'immagine.

Console di comando Hardware

La medesima forma dell'interfaccia VisualBasic, è riprodotta in una console concepita puramente hardware. Essa conta ben 10 cursori lineari abbinati al gruppo RC formante la costante di tempo dello stadio di comparazione dei Timer NE555 che figurano nello schemo del servotester.
La funzionalità è semplice ed intuitiva, non richiede la presenza di un PC, è stabile e molto rapida in risposta.



.



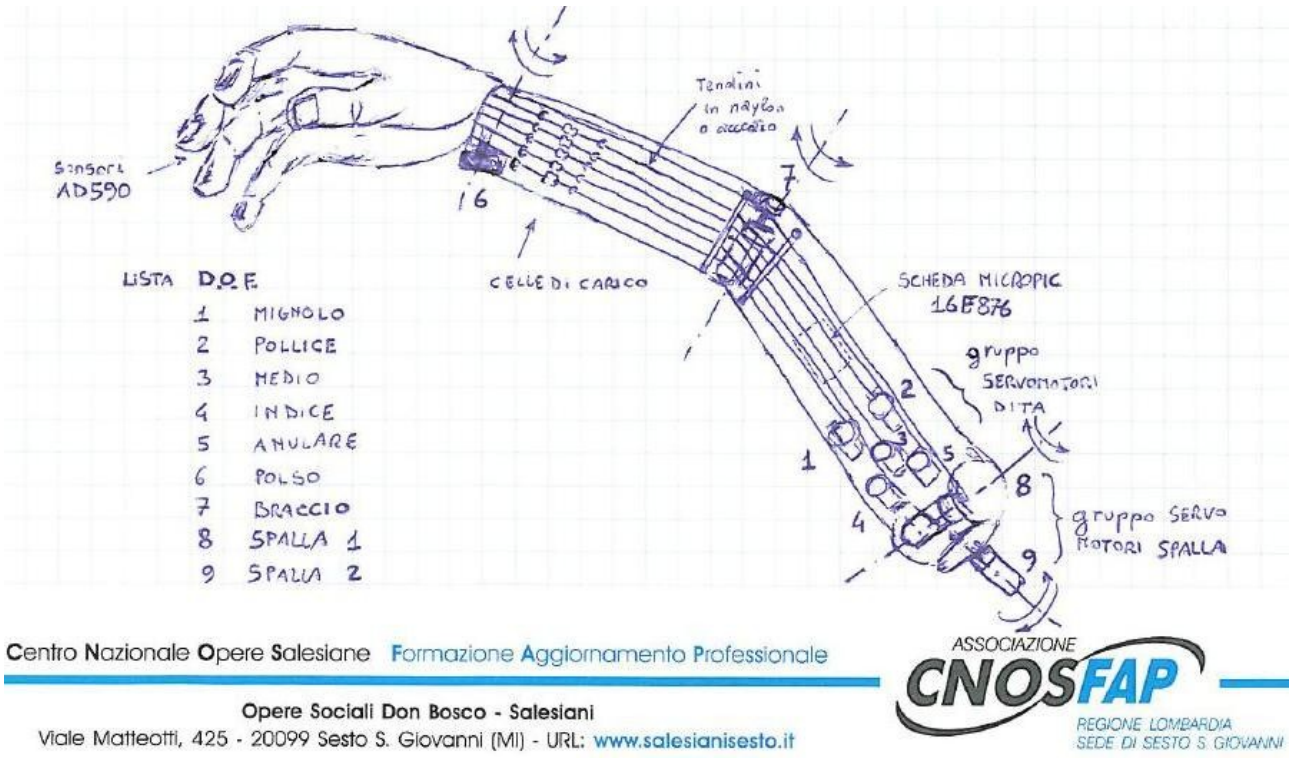
Ogni canale dell'interfaccia Hardware è costituito da un esemplare del servotester.
Esiste un unico morsetto per l'alimentazione che va posta a +5 Volt.
Il sistema è ovviamente modulare e quindi espandibile in funzione delle esigenze.
Per pilotare servomotori di potenza direttamente bisogna sostituire il BC337 ben visibile nella foto con un più robusto TIP122, ovvero un darlighton di potenza in grado di sopportare ben 8 ampere.
Possiamo comunque mantenere il BC337 ed eseguire l'interfacciamento con il servo di potenza tramite tramite un'ulteriore interfaccia esterna.

Assemblaggio dell'hardware del progetto RobotHand V2

Il prototipo della protesi, avente puro scopo dimostrativo è stato realizzato in lamierino di ottone dello spessore di 20 decimi di millimetro. La scelta di tale materiale è dovuta alla facilità di lavorazione, quale piegatura e foratura, nonché possibilità di eseguire piccole saldature con un normale saldatore per stagno 40/60 per normali lavorazioni elettroniche.
Una volta eseguite le ripiegature il materiale risulta essere di buona solidità e particolarmente adatto per fare da supporto non solo agi attuattori (servomotori) ma anche ai sensori (AD590, celle di carico, ecc).
La prima versione del braccio robotizzato presenta delle evidenti carenze strutturali, dovute senz'altro a una maggiore concentrazione in fase di sviluppo nei controlli software e alla componentistica elettronica, piuttosto che a una eccellente funzionalità meccanica.

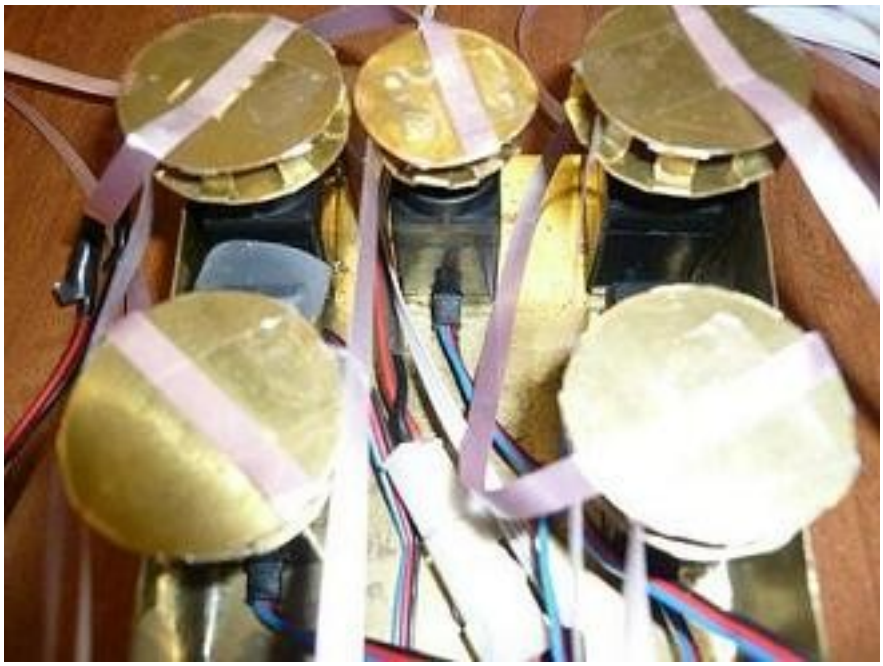
I tendini, nella prima versione, sono stati realizzati in nastrino piatto normalmente adoperato per confezionare i regali, e la scelta si è rilevata molta funzionale. Nella seconda versione si sono sostituiti con del filo di nailon usato per la pesca perché più indicati nel vincolare le celle di carico miniaturizzate.

Il disegno a mano libera sovrastante rappresenta lo schema di principio per la rilevazione della trazione dei tendini. Questa informazione si traduce direttamente in quanto forte la mano stringe un oggetto afferrato. La parte sinistra, il vicolo, è il polpastrello del dito a cui è collegato il servomotore a cui rilevare la coppia. Il servomotore è collegato al lato destro del sistema vincolo, cella di carico, puleggia di trazione, e la sua coppia è rappresentata dal vettore **F**.
La cella di carico varia il suo valore ohmico in funzione della trazione ad essa applicata ed è normalmente impiegata per la costruzione dei bilance elettroniche di precisione.
Esistono molti tipi di celle di carico, con svariate forme, ma quelle più adatte a questo scopo sono a forma di piccolo anello come quella rappresentata nel disegno.
L'idea di base, su come si sarebbe potuto sviluppare il progetto robot hand V2 è arrivata durante un viaggio in treno tra Milano e Venezia, ed è stata fissata a mano libera sul foglio di carta intestata di cui si riporta la scansione.

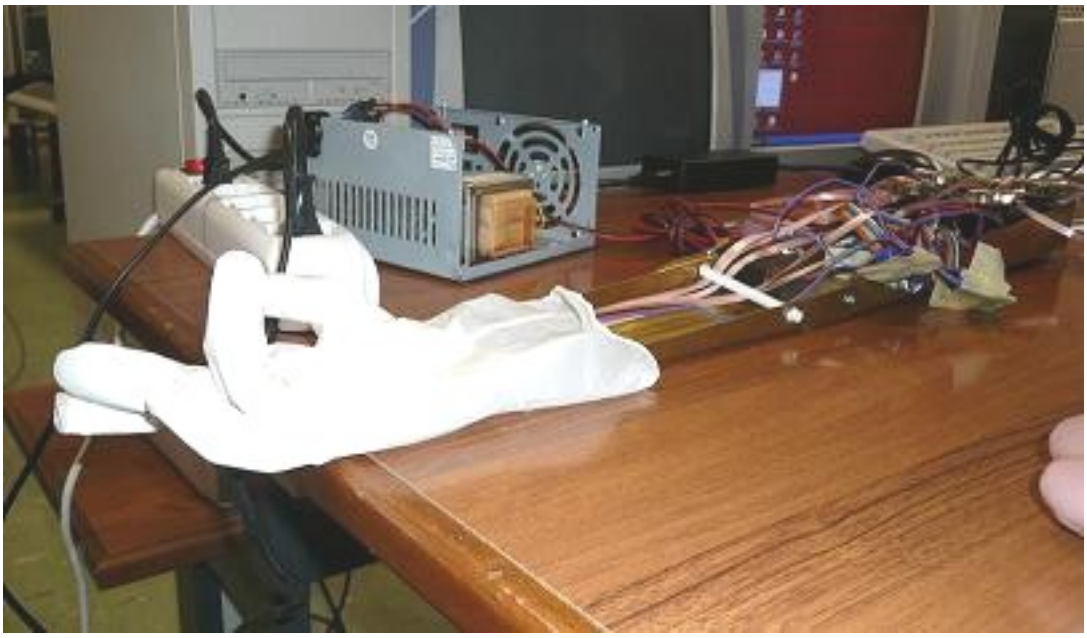


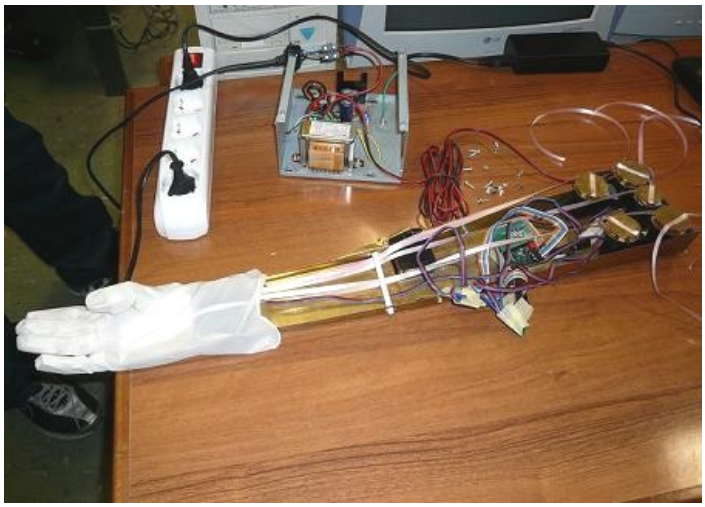
Nel disegno sono indicati i D.O.F. (degree of freedom o gradi di libertà), ben 9 corrispondenti grossomodo ad ogni asse di rotazione governato da uno specifico servomotore.

I servomotori da 1 a 5 sono dei normali microservo da modellismo, visto che dovranno governare solo la trazione dei tendini a cui sono vincolate le singole dita.
I servo 7, 8, 9 sono invece di tipo dedicato, si veda il paragrafo **Versione dedicata del servomotore,** in cui per ragioni di coppia e di forza da sviluppare, si pensi anche al solo peso dell'arto scarico, è necessario installare un motore DC di opportuna potenza.
Il servomotore numero 6 è invece dedicato al polso. Nel prototipo è stato impiegato un normale servo per modellismo, ma è a scopo dimostrativo. Se volessimo sovraccaricare l'arto sarebbe necessario sostituirlo con un attrattore di coppia maggiore.
Il gruppo principale di servomotori è alloggiato appena sotto la spalla, ovvero nella parte più spaziosa dello chassy di ottone.
Gli assi dei servomotori sono stati dotati di pulegge autocostruite, sempre in lamierino di ottone, avente lo scopo di dare la giusta corsa ad ogni parte mobile, ad esempio il dito medio necessita di una corsa maggiore del pollice, scopo che si ottiene calcolando opportunamente il raggio delle pulegge.



L'immagine riporta una prima prova di collegamento dei tendini, nastrini della prima versione, con le pulegge. Si nota che la dimensione della puleggia centrale è notevolmente ridotta rispetto alle altre, infatti questa è il pollice che ha notoriamente una corsa inferiore.
Pochi centimetri sopra il gruppo servomotori principale è installata la scheda di controllo a microPIC. I tendini passano proprio sopra la scheda quindi non devono risultare un ingombro per il movimento.
Nella versione definitiva si dovranno alloggiare anche le schede dell'alimentazione di potenza.



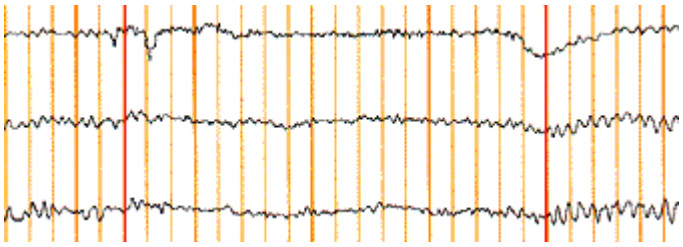


Tecnica di comando “mirror mode”
La tecnica di pilotaggio mirror mode consiste nel copiare i movimenti della mano sana sulla protesi. A tal scopo vengono nascosti sul dorso del guanto 5 potenziometri lineari a slider con ritorno a molla. Questi saranno vincolati tramite cordini di acciaio alle punta delle dita del guanto.
La curvatura delle falangi comporterà la trazione del cavetto di acciaio e il movimento del cursore che ovviamente viene acquisito da una variante della scheda usata per fare la console di pilotaggi hardware.
I cavetti di alimentazione dei motori scorrono lungo il braccio, nascosti sotto i vestiti, e finiscono nei servomotori installati nella protesi.
La scheda di controllo è vestita come un bracciale sul braccio della mano buona poco più in alto del guanto che fa da supporto ai cursori.

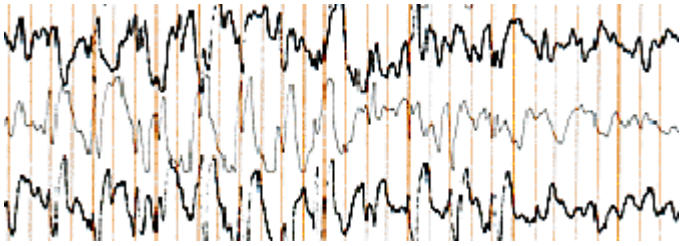
Segnali EEG.

Segnale elettro encefalografico di un soggetto adulto normodotato acquisito a livello del cuoio capelluto tramite sensori esteri. L’acquisizione è avvenuta a soggetto sveglio ad occhi aperti. Il segnale risulta omogeneo e ripetitivo con ampiezze dell’ordine di 50 micro Volt. All’altezza della

seconda riga verticale rossa è presente un picco negativo corrispondente al battito delle ciglia. Tale picco negativo risulta presente ad ogni battito di ciglia volontario o involtario pertanto potrebbe essere sfruttato, con opportune elaborazioni e emissioni in sequenze volontarie, come segnale di pilotaggio del sistema di controllo dell'arto artificiale.

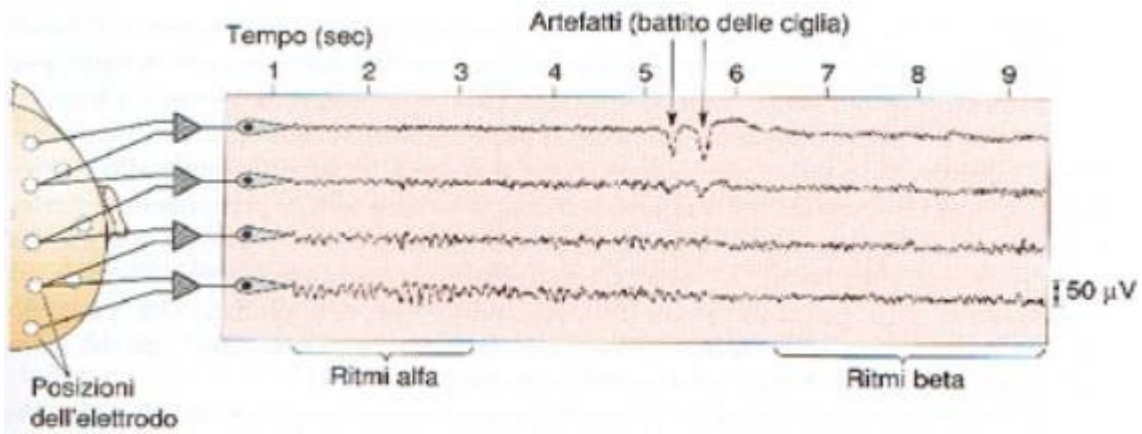


Il primo problema che incontriamo nell'applicazione del progetto è l'esistenza di tracciati EEG anomali sviluppati dai soggetti epilettici. Ne riportiamo un esempio nell'immagine sottostante.



Il tracciato non presenta segnali filtrabili corrispondenti a azioni volontarie/involontarie come ad esempio il battito delle ciglia.

I sensori utilizzati dai normali strumenti elettromedicali per EEG hanno l'aspetto di piccole placche applicabili in maniera non invasiva in zone standard della calotta cranica. La mappatura encefalica dei segnali risulta poco diversa da soggetto a soggetto.

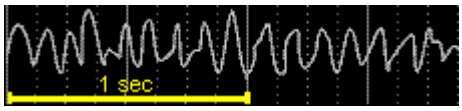


Con la tecnica dell'averaging, che consiste nel sovrapporre porzioni di segnali allineandole rispetto ad un determinato evento temporale, è possibile ottenere dall'EEG i cosiddetti ERP, event related potentials.

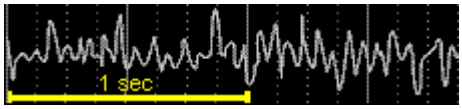
Il potenziale evocato, ERP, consiste in una variazione specifica dell'EEG e della MEG conseguente ad una stimolazione di una via sensoriale o ad un evento motorio.

Esso è costituito da oscillazioni del potenziale elettrico o del flusso magnetico (secondo la tecnica sperimentale utilizzata) e da forme d'onda caratterizzate da una serie di deflessioni positive o negative. Queste deflessioni vengono normalmente definite componenti. La polarità delle componenti ERP dipende dalla posizione dell'elettrodo EEG o sensore MEG all'interno della distribuzione del campo elettrico e magnetico superficiale. A sua volta la distribuzione dei campi superficiali dipende dall'area corticale attivata e dal suo orientamento rispetto al cuoio capelluto.

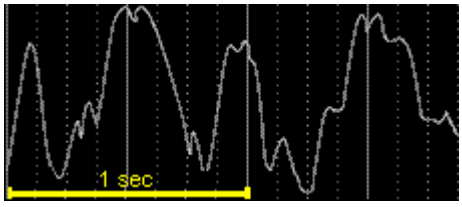
Classificazione dei segnali celebrali.
Le onde prodotte dal cervello sotto forma di emissione elettrica sono suddivise in 4 principali categorie, onde alfa, beta, delta, theta. Esse si distinguono per ampiezza, frequenza, e stato del paziente in esame. La prima distinzione riguarda la differenza di tracciato tra stato onirico e stato di veglia a sua volta suddiviso soggetto a occhi chiusi e soggetto ad occhi aperti.
I tracciati che seguono rappresentano le registrazioni oscillografiche alfa,beta, delta, theta di un paziente in stato di veglia ad occhi aperti.



onde ALFA (da 8 a 13 Hz -cicli al secondo) appaiono durante il riposo o la meditazione.
Compaiono quasi sempre appena chiudiamo gli occhi.
Indicano in genere una situazione di benessere psicofisico.
Con il rilassamento il ritmo della nostra attività cerebrale rallenta, le onde cerebrali diventano più distanti, ampie, lente e meglio formate. A questo livello la coscienza è vigile, ma l'attività fisica e la percezione degli stimoli estemi è ridotta al minimo, il corpo è rilassato. E' lo stato che proviamo nel dormiveglia e al risveglio, possono apparire delle immagini legate a situazioni precedenti o future. In queste situazioni ci capita di "fantasticare" ricevendo comunicazioni e intuizioni.

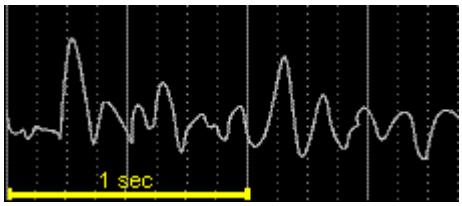


onde BETA (da 14 a 35 Hz -cicli al secondo) stato di veglia sono presenti quando puntiamo l'attenzione o ci impegniamo nella soluzione di un problema.
A questo livello percepiamo attraverso i sensi fisici. Le onde cerebrali prevalenti sono piccole, brevi, ravvicinate, mal sincronizzate.
E' uno stato ricettivo alle stimolazioni provenienti dal mondo fisico, le seleziona, cataloga, elabora e applica. Leggendo queste righe il vostro cervello sta producendo onde beta

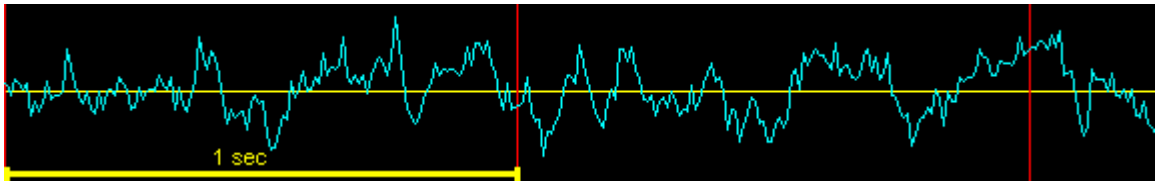


onde DELTA (da 0,5 a 4 Hz -cicli al secondo) Inconscio Sono le onde più lente. Compaiono soprattutto durante lo stato di sonno non REM (senza sogni). Sono in correlazione con il nostro inconscio più antico e profondo. Sono collegate anche a stati di tensione muscolare. l'inconscio mantiene le attività vitali dell'organismo al minimo e il subconscio si è aperto totalmente. Questo stato di coscienza viene anche chiamato "estasi". Le onde delta possono essere raggiunte

artificialmente con le droghe che però, poi, portano il cervello a perdere neuroni sino alla precoce ed irreversibile degenerazione totale. Mentre con le tecniche di meditazione si possono raggiungere, egualmente, questi livelli di "estasi" volontariamente e senza danni cerebrali.



onde THETA (da 5 a 7 Hz -cicli al secondo) Si rilevano soprattutto durante lo stato di sonno REM (sogni).
Sono in correlazione con i nostri cambiamenti di umore, con le emozioni e i sentimenti, con le nostre immagine profonde.
Le onde cerebrali sono più lente e ampie. domina i processi biologici, il subconscio sta aprendo la porta di comunicazione con il conscio. E' il livello del sonno profondo e del sogno. Siamo in un'altra dimensione, dove è attiva una coscienza diversa che usa organi di percezione diversi senza limiti spazio-temporali è lo stato dell' "ispirazione". e delle guarigioni cosiddette "miracolose" è usato in molte terapie di solistiche



Un tracciato EEG completo delle armoniche alfa, beta , delta e theta ha quindi, in un soggetto normodotato e sveglio ad occhi aperti, l'aspetto riportato nella figura sottostante.
Nel tracciato non è evidenziato alcun movimento, spasmo involontario o segnale diverso dalle normali funzioni vitali nello stato di riposo.

L'esperimento di Martingale.

Verso la fine degli anni '70 uno psicologo americano, dell'Università del Maine, di nome Martindale, usò l'elettroencefalogramma per testare le onde cerebrali di alcune persone che inventavano una storia. L'esperimento rivelò che la creatività ha due fasi, l'ispirazione e l'elaborazione. Durante l'ispirazione il cervello è molto tranquillo perché lavora in onde ALFA che caratterizzano le sedute di meditazione e di yoga. Questo spiegare anche quello che accade a molte persone che pensano ad un problema prima di addormentarsi perché sanno che poi svegliandosi avranno la soluzione a portata di mano. Infatti, durante il sonno e il rilassamento le onde ALFA aiutino il cervello ad essere creativo e a trovare soluzioni impensabili in stato di veglia. Martindale osservò, però, che quando veniva chiesto di elaborare la storia, le onde passavano da ALFA a BETA e l'attività del cervello diventava più frenetica. Le persone che presentavano una maggior differenza di frequenza fra la fase creativa e la verbalizzazione della storia stessa producevano racconti più fantasiosi.

Cosa sono le onde cerebrali e come funzionano
Qui di seguito abbiamo riprodotto un grafico (elettroencefalogramma) tratto del EEG ovvero dalla macchina chiamata elettroencefalografo inventato attorno al 1920 dallo svizzero H. Berger che aiutò moltissimo nelle diagnosi dell'epilessia e delle lesioni cerebrali. segue poi uno schema delle

varie onde cerebrali i cui nomi abbiamo scritto nei colori che corrispondono alle loro vibrazioni. E' da sottolineare soltanto che non tutti i testi scientifici sono in accordo sull'esatta frequenza delle onde cerebrali calcolate in "cicli al secondo" ovvero in Hertz (Hz). Noi abbiamo riportato una media accettata dalla maggioranza dei testi consultati.
Per gli appassionati di internet, fra i molti siti interessanti da segnalare a proposito citiamo:

http://www.intelligenzartificiale.com/onde_cerebrali.htm

http://www.guruji.it/ondealfa.htm

dove, in quest'ultimo, si possono ascoltare interessanti brani che portano il cervello in ALFA

Future applicazioni della robotica umanoide

Algoritmi genetici
Un **algoritmo genetico** è un metodo di ricerca globale e stocastico basato sulla metafora dell'evoluzione biologica. Gli **Algoritmi Genetici (GA)** operano su una popolazione di potenziali soluzioni applicando il principio della sopravvivenza del migliore, evolvendo verso una soluzione che si spera si avvicini quanto più possibile alla reale soluzione del problema.
Ad ogni generazione, un nuovo insieme di soluzioni è creato dal processo di selezione che, basandosi sul livello di adeguatezza (**Fitness**), seleziona i migliori membri della popolazione e li fa evolvere utilizzando una serie di **operatori genetici** mutuati dalla genetica naturale. Questo processo porta ad una evoluzione robusta verso individui che meglio si adattano all'ambiente, in altre parole, all'insieme di soluzioni che meglio rispondono al problema posto in principio.
Ogni individuo della popolazione, (o ogni soluzione che dir si voglia) è codificato sotto forma di stringa, detta **Cromosoma**. Il sistema di codifica più utilizzato è il sistema **binario**. Ogni soluzione è quindi rappresentata (codificata) come una stringa di 0 ed 1. Ad esempio, una soluzione di un problema a 2 variabili può essere rappresentata come:

0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 1 0 1 0 1

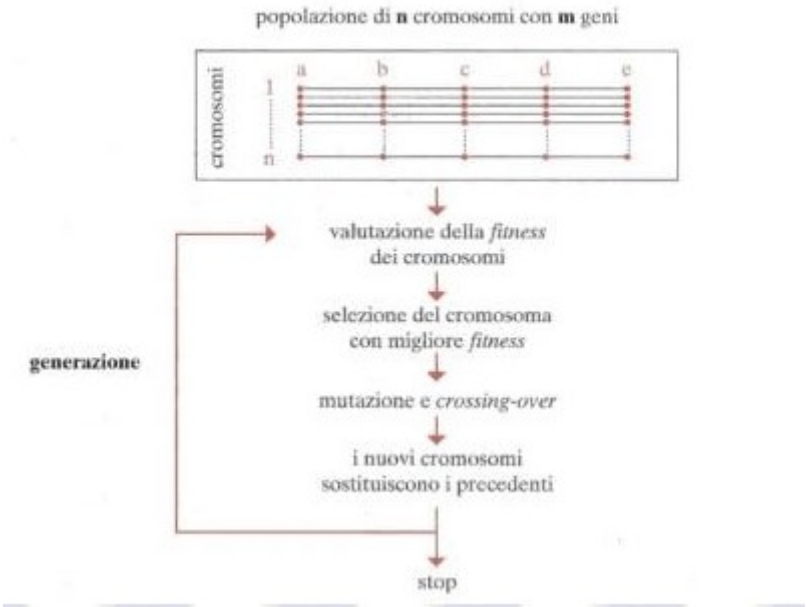
Le prime dieci cifre codificano la prima variabile y, le seconde 10 codificano la seconde variabile. Ogni soluzione è caratterizzata da un indice di Fitness che esprima quanto si adatta al problema data in input. Come accade in natura solamente gli individui con maggiore capacità di adattamento all'ambiente sono capaci di sopravvivere e di riprodursi, anche l'algoritmo genetico selezione le soluzione con maggiore indice di fiteness e gli abbina una maggiore probabailità di soppravivenza consentndogli di trasmettere i suoi geni come input della generazione futura. Gli individui con fitnees elevato (a confronto con la fitness media della popolazione) con molta probabilità saranno selezionati come genitori della generazione di soluzioni future. Data una selezione composta da un certo numero di individui, l'algoritmo genetico simula la riproduzione sessuata che si verifica in natura creando una biodiversità ovvero una variabilità genetica tramite un operatore genetico di **Cross Over** e di **Mutazione puntuale**.

La nuova generazione di soluzioni prende il posto della generazione precedente, dalla quale è nata per ri-combinazione.
Il processo viene reiterato per un numero x di volte fino a quando o si raggiunge una approssimazione accettabile della soluzione al problema o si raggiunge il numero massimo di iterazioni prefissato.

Elementi costitutivi di un algoritmo genetico
Schematizzando, gli elementi costitutivi di un algoritmo genetico sono:

- **Popolazione**
  Costituita da un numero n di individui. Ogni individuo rappresenta una possibile soluzione al problema.
- **Funzione Fitness**
  La funzione di Fitness è una funzione in grado di valutare quanto una soluzione è adatta a risolvere il problema dato. Ad ogni soluzione corrisponde quindi un valore di fitness.
- **Principio di selezione**
  Il principio di selezione ha il compito di selezionare gli individui della popolazione (le soluzioni) che meglio rispondono al problema da risolvere. La selezione si basa sulla fitness degli individui; le soluzioni con fitness maggiore (rispetto alla media della popolazione) avranno maggiori possibilità di partecipare alla riproduzione e quindi di trasmettere alle future generazioni i propri geni.
- **Operatori genetici**
  Prendendo spunto dalla biologia, gli operatori genetici combinano i geni delle diverse soluzioni al fine di esplorare nuove soluzioni. Una volta che un gruppo di soluzioni viene individuato come idoneo alla riproduzione, l'operatore genetico di cross over, emulando la riproduzione sessuata degli esseri viventi, combina i geni dei genitori e formula una nuova generazione di soluzioni. Un altro operatore genetico largamente utilizzato è la Mutazione puntuale. La mutazione puntuale agisce direttamente sui figli, andando a modificare un gene a caso.

Ecco uno schema che spiega in sintesi il funzionamento di un algoritmo genetico:



Di seguito, un esempio di un algoritmo genetico base implementato in Matlab. Il problema è la minimizzazione della funzione di De Jong:

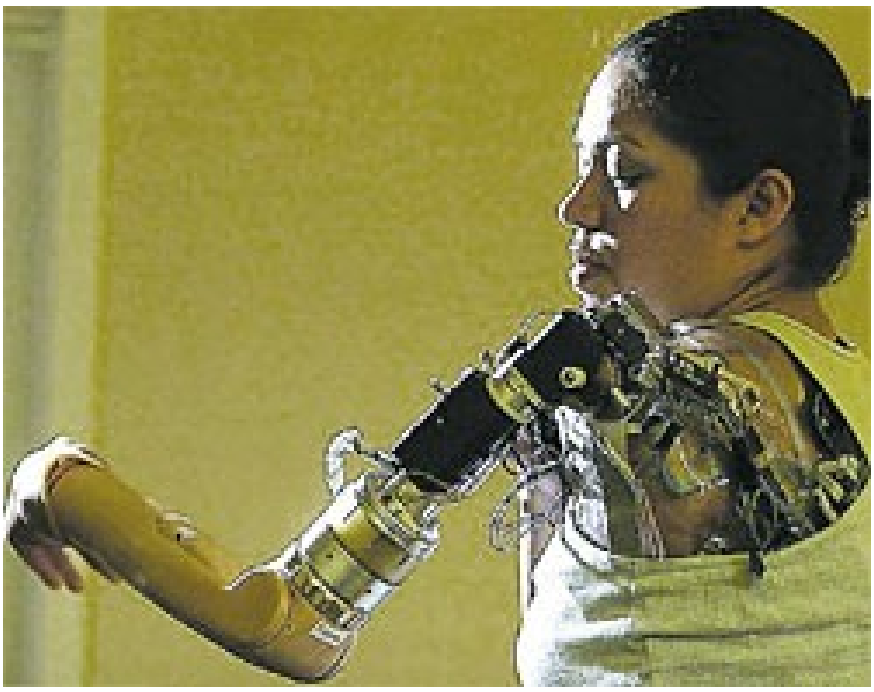
$$f_1(x) = \sum_{i=1}^{n} x_i^2, \qquad -512 \le x_i \le 512$$

Il minimo di questa funzione si ottiene per x=0.
Ecco il listato del programma per Matlab:

```
NIND = 10;
% Numero di individui

MAXGEN = 1000;
% Numero di generazioni (iterazioni)

NVAR = 20;
% numero di variabili

PRECI = 20;
% Bit utilizzati er decrivere ciascuna variabile, PRECISIONE

GGAP = 0.9;
% Quanti individui vengono sostituiti da nuovi
% individui ad ogni iterazione

FieldD = [rep([PRECI],[1,NVAR]);…
rep([-512;512],[1,NVAR]); rep([1;0;1;1],[1,NVAR])];
% Field Descriptor, ovvero indichiamo che il nostro cromosoma
% è costituito da 20 variabili, rappresentate
% da 20 bit, che possono assumere valori da
% -512 a 512 codificate
% in Gray code.

Chrom = crtbp(NIND, NVAR*PRECI);
% Costruzione del cromosoma, ovvero
% una matrice di righe = NIND (Numero INDividui)
% e colonne = NVAR * PRECI (Numero VARiabili * PRECIsione)

gen = 1;
% Contatore generazioni

ObjV = objfun1(bs2rv(Chrom,FieldD));
% Codifico il Fenotipo corrispondente al Genotipo, sulla base del
% descrittore FieldD
% Dopodichè lo invio alla Funzione oggettiva,
% che nel nostro caso è rappresentata
% dalla fuznione De Jong. Restituisce un vettore colonna
% con il valore obbiettivo
% corrispondente ad ogni individuo

while gen < MAXGEN,

FitnV = ranking(ObjV);
% Calcola il valore di fitness sulla base dei valori obbiettivi.
% Restituisce un vettore colonna contenente i
% valori di fitness di ogni individuo.
% La fitness assume valori da 0 a 2.
% La funzione ranking assume
% che la funzione oggettva sia da MINIMIZZARE

minobjv(gen) = min(ObjV);
totobjv(gen) = mean(ObjV);
% Dati per statistiche finali
```

```
SelCh = select('sus', Chrom, FitnV, GGAP);
% Questa funzione seleziona con metodo
% Stochastic Universal Sampling,
% dalla matrice
% Chrom, basandosi sulla Fitness, una quantità di individui
% pari a GGAP*NIND

% La matrice SelCh contiene ora i cromosomi
% dei genitori, selezionati in base alla fitness,
% per la riproduzione.

SelCh = recombin('XOVSP',SelCh,0.7);
% I figli vengono creati ricombinando i geni
% dei cromosomi contenuti in SelCh
% Per la riproduzione viene usato Crossover
% Single Point con possibilità del 70%

SelCh = MUT(SelCh);
% Applica la mutazione ai genitori contenuti
% nella matrice SelCh. Se non specificato diversamente,
% la probabilità d mutazione è pari a
% 0.7/Lunghezza del cromosoma.
% Nel nostro caso, 0.7/20*20

ObjVSel = objfun1(bs2rv(SelCh,FieldD));
% Calcolo il valore della funzione oggettiva relativa ai genitori

[Chrom ObjV]=REINS(Chrom,SelCh,1,1,ObjV,ObjVSel);
% Sostituisco i figli ai genitori.
% La sostituzione avviene all'interno
% della matrice originale Chrom e
% sulla base della Fitness (il quarto
% parametro "1" indica reinserimento in base alla fitness)

gen = gen+1
% Incremento il contatore generazione di 1 e
% continuo fino alla generazione massima indicata

end

figure ('name','Andamento minimo funzione obbiettivo')
plot (minobjv)
figure ('name','Andamento del totale della funzione obbiettivo')
plot (totobjv)
min (ObjV
```

protesi cibernetiche




Rehabilitation Institute of Chicago
www.ric.org