ISTITUTO TECNICO INDUSTRIALE STATALE

"Celestino Rosatelli"

Classe V sez.B ind. Elettronica & Telecomunicazioni

Esami di maturità A.S. 2009-2010

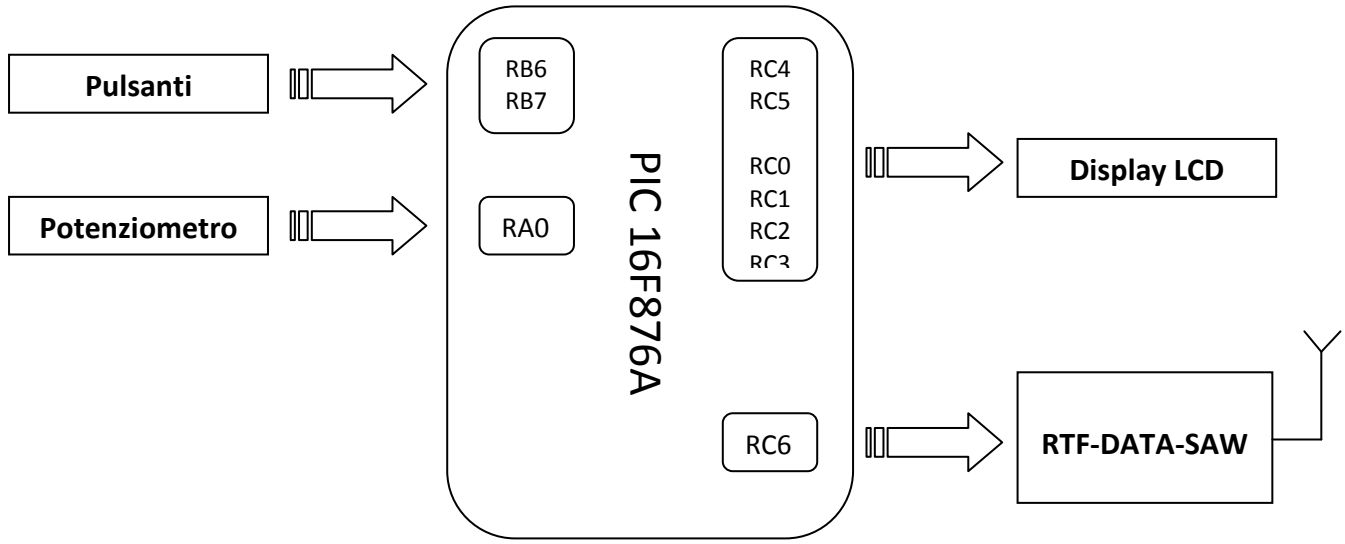
"Robot Radiocomandato"

Matteo Gentileschi

Introduzione

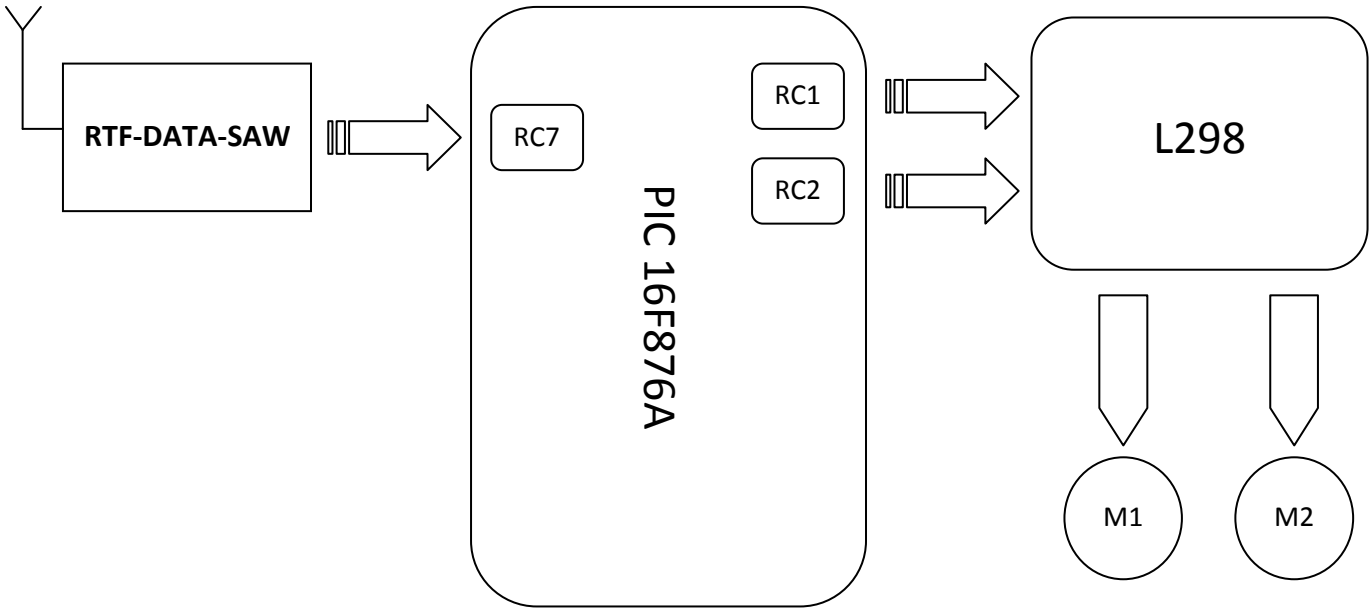
Si vuole realizzare un robot controllato da remoto capace di compiere movimenti elementari quali andare avanti, indietro, destra, sinistra e di variare la velocità di marcia: il classico rover a due ruote motrici e un ball caster. Il movimento del robot è controllato mediante due pulsanti presenti sul radiocomando, che permettono la selezione del verso di marcia (destra o sinistra), mentre la velocità è regolata mediante un cursore, il tutto tramite etere. Sia sul radiocomando che a bordo del robot sono presenti due µC PIC16F876A della microchip che comunicano utilizzando una trasmissione seriale (RS232) unidirezionale tramite due moduli di ricetrasmissione RTF-DATA-SAW della AUREL.

- **Schema a blocchi radiocomando :**



I pulsanti su RB6 e RB7 servono a determinare a quale motore viene applicata un'onda rettangolare il cui duty-cycle, viene impostato da un potenziometro a slitta la cui tensione ai capi è letta sul canale A0. Il µC genera un pacchetto contenente tali informazioni e lo invia usando il protocollo RS232 al modulo RTF-DATA-SAW che lo immette nell'etere. Nel display viene visualizzato il valore del duty-cycle e il motore a cui è applicato il segnale.

- **Schema a blocchi del controllo del robot :**



Il µC attende un interrupt che segnala l'arrivo di un nuovo pacchetto proveniente dal radiocomando, ed estratte le informazioni necessarie a ricavare il duty-cycle del segnale PWM (generato dai moduli CCP) prelevato dai pin RC1 e RC2. Tale segnale applicato appropriatamente ad un L298 consente di regolare direzione e verso al motore come si vedrà in seguito.

Prima di trattare il progetto vero e proprio è opportuno fare una panoramica veloce su tutto gli argomenti trattati per realizzare questo progetto, che com'è possibile intuire racchiude le conoscenze di tutte le materie di indirizzo e del biennio I.T.I.S. .

Protocollo RS232

Come già detto precedentemente, il radiocomando e il robot comunicano via etere.
Per poter comunicare e scambiare dati digitali è necessario che entrambi i dispositivi rispettino un insieme di regole e specifiche che prende il nome di "Protocollo".
Una trasmissione digitale può avvenire in modo parallelo se il dato viaggia in più linee dati contemporaneamente, o in modo seriale se il dato viaggia in una sola linea, come nel caso del RS232, come sequenza di bit.
Lo scambio dei dati tra trasmettitore e ricevitore avviene in modalità differenti che stabiliscono la direzione dei dati: si parla di trasmissione "simplex" quando la trasmissione avviene in un unico senso, da trasmettitore a ricevitore (senza preoccuparsi di un'eventuale risposta del ricevitore ,in questo caso non c'è un vero e proprio "scambio" di dati); "half duplex" se la trasmissione avviene in entrambi i sensi, ma, non contemporaneamente (perciò la stessa linea può essere usata sia per trasmettere che per ricevere) e "full duplex" se la ricetrasmissione avviene in contemporanea.
Le trasmissioni seriali possono avvenire in modo sincrono o in modo asincrono: la trasmissione è sincrona quando ricevitore e trasmettitore condividono lo stesso segnale di clock, ciò succede quando il trasmettitore trasmette i primi bit e il ricevitore tramite un PLL (anello ad aggancio di fase) ricava da essi le caratteristiche del clock che dovrà utilizzare.
La trasmissione asincrona è caratterizzata dal fatto che i clock del ricevitore e del trasmettitore sono indipendenti. A differenza di quella sincrona in cui la trasmissione dei bit è continua, nella trasmissione asincrona i bit di dato sono trasmessi in un momento qualsiasi. Il ricevitore quindi deve essere attivato , insieme al suo clock al momento in cui giunge il carattere e una volta effettuata la sua ricezione può tornare nella condizione di riposo. Normalmente nel trasferire in modo asincrono i dati si associa al livello logico 0 lo stato di attivazione e al livello logico 1 lo stato di riposo. Nella trasmissione asincrona è indispensabile inserire dei bit di controllo che definiscono l'inizio (bit di start) e la fine di un carattere (bit di stop).
Lo standard RS232 è un protocollo per la trasmissione di dati in formato digitale che avviene in maniera asincrona, è può essere simplex, half duplex o full duplex a seconda delle caratteristiche dei dispositivi hardware a disposizione. Possono essere presenti ulteriori linee che permettono l'handshaking ("stretta di mano") ossia permettono al ricevitore e al trasmettitore di scambiare informazioni l'uno con l'altro.
Nella seguente tabella sono riportate le specifiche elettriche richieste dal protocollo RS232:

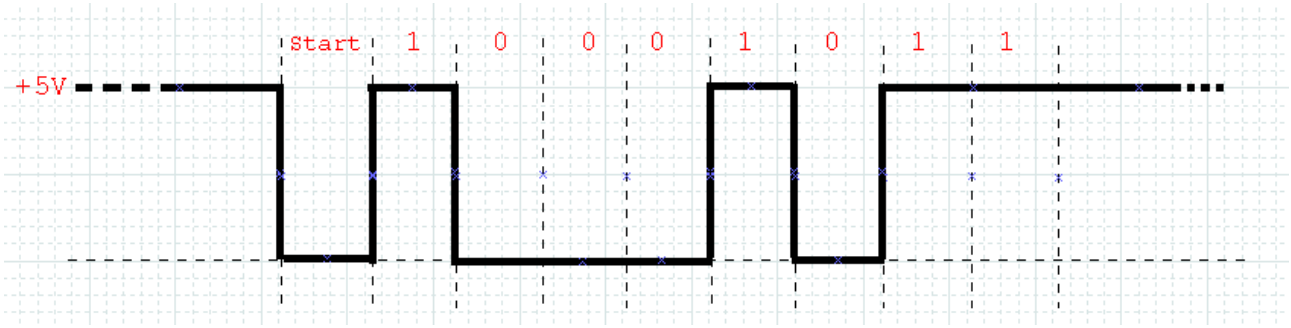
	Livello logico	Trasmettitore	Ricevitore
SPACE	0	+5V ÷ +15V	+3V ÷ +25V
MARK	1	-5V÷ -15V	-3V÷25V
Non definito	-3V ÷ +3V		

In genere si usano livelli di tensione compresi tra +/- 12V,per limitare i disturbi elettromagnetici agli altri cavi che viaggiano in prossimità del cavo seriale(è imposto anche uno slew-rate massimo a $30V/\mu s$) mentre al contrario se si desidera una maggiore immunità ai disturbi elettromagnetici si usano livelli di tensione più alti.
Questi livelli devono spesso essere adattati ai più comuni livelli TTL o CMOS, esistono appositi circuiti integrati che svolgono questa conversione di livello come il MAX232.
Il numero di transizioni che avvengono sulla linea prende il nome di "baud rate" mentre in numero di bit per secondo (bps) rappresenta appunto il massimo numero di bit che possono essere trasmessi per ogni

secondo, in particolare nella trasmissione digitale come l'RS232 il baud rate coincide con il bps,poiché sono solo due livelli.
Nel caso di transizioni a più livelli si possono trasmettere più bit in una sola transizione,ad esempio trasmettendo otto livelli di tensione compresi tra 0V e 7V, ogni livello contiene 3 bit (1V=000;2V=001;...) ,una trasmissione a 1000 Baud equivale a una a 3000bps.
Durante la trasmissione sia essa seriale sia parallela, può accadere che un bit cambi di livello a causa di disturbi esterni,ciò comporta la ricezione di un dato differente da quello inviato.
L'errore di per se non è eliminabile ma aggiungendo della "ridondanza", ovvero degli ulteriori bit che consentano di eseguire un controllo è possibile individuare l'errore, maggiore è l'informazione di ridondanza maggiore è la possibilità di individuare errori. Nell'RS232 viene inserito un "bit di parità" che può essere di cinque tipi:

- NONE: nessun tipo di parità, cioè nessun bit aggiunto;
- PARI (EVEN) : il numero di mark (incluso il bit di parità) è sempre pari;
- DISPARI (ODD): il numero di mark (incluso il bit di parità) è sempre dispari;
- MARK : il bit di parità vale sempre mark;
- SPACE : il bit di parità vale sempre space;

Dal momento che i protocollo RS232 è di tipo asincrono è necessaria qualche strategia per sincronizzare la fase di trasmissione con quella di ricezione. La tecnica adottata è quella di rimanere a livello logico 1 per poi trasmettere un bit di start oltre alla normale sequenza.
Tale sequenza verrà trasmessa a partire dal bit meno significativo ed eventuali bit di parità,saranno posti in coda dopo il bit più significativo (è possibile inviare un dato composto da 6 a 9 bit).
Ad esempio se si vuole trasmettere il byte "11010001" la sequenza di trasmissione sarà la seguente :



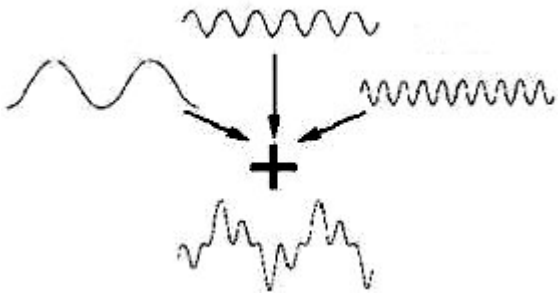
In questo caso non c'è parità,se ci fosse il relativo bit (even) varrebbe 0, in quanto il numero di 1 è pari.

Esistono dei dispositivi hardware che si occupano di convertire il flusso dei bit da seriale a parallelo e viceversa,tali dispositivi sono chiamati USART (Universal Synchronous-Asynchronous Receiver & Transmitter) che gestiscono la trasmissione, occupandosi di eventuali ridondanze e sincronismi ed essendo dotati di un buffer di tipo FIFO consentono la ricezione dei dati anche se la CPU è occupata. Tutte le famiglie di microcontrollori contengono al loro interno questi moduli,settabili tramite appositi registri, usati anche in questo progetto.

Per comprendere meglio il principio di funzionamento che fa muovere il robot, è necessario conoscere l'analisi spettrale di un segnale e in particolare analizzare lo spettro di un segnale PWM.

Serie di Fourier

Secondo il teorema di Fourier un segnale periodico, può essere scomposto in una somma di infiniti segnali sinusoidali, aventi ampiezze differenti e frequenze multiple rispetto alla fondamentale che ha la stessa frequenza del segnale stesso, più un termine costante che assume il valore del valor medio del segnale stesso.



Ciò può avvenire solo al verificarsi delle seguenti condizioni, che per quanto riguarda il tipo di segnali da noi trattati sono quasi sempre rispettate:

- La funzione del segnale f(t) deve contenere in un periodo un numero finito di massimi e minimi;
- Se la funzione ha discontinuità,il loro numero deve essere finito in un periodo;
- La funzione del segnale f(t) deve essere assolutamente integrabile in un periodo :

$$\int_0^T f(t)\ dt < \infty$$

Supponiamo di avere un segnale f(t) di periodo T e dunque una fondamentale f=1/T, esso può essere espresso in due forme:

- **Forma trigonometrica:**

$$f(t) = A_0 + \sum_{n=1}^{\infty} A_n \cdot \cos(n \cdot \omega \cdot t) + B_n \cdot sen(n \cdot \omega \cdot t)$$

Dove:

A_0 = valore medio della funzione

$A_n \cdot \cos(n\omega t)$

$B_n \cdot sen(n\omega t)$

} Segnali armonici a pulsazione multipla

$$A_n = \frac{2}{T} \cdot \int_0^T f(t) \cdot \cos(n\omega t)\ dt \qquad B_n = \frac{2}{T} \cdot \int_0^T f(t) \cdot sen(n\omega t)\ dt$$

La funzione f(t) può essere:

- **Pari** se f(t) = f(-t) ,ovvero se la funzione è simmetrica rispetto all'asse delle ordinate.

Poiché la funzione **sen(nωt)** è una funzione dispari anche il prodotto **f(t)sen(nωt)** risulterà dispari,dunque tutti i termini Bn risulteranno nulli in quanto gli integrali calcolati nei due semi periodi sono uguali e di segno opposto.
In questo caso la serie di Fourier del segnale f(t) è la seguente :

$$f(t) = A_0 + \sum_{n=1}^{\infty} A_n \cdot \cos(n \cdot \omega \cdot t)$$

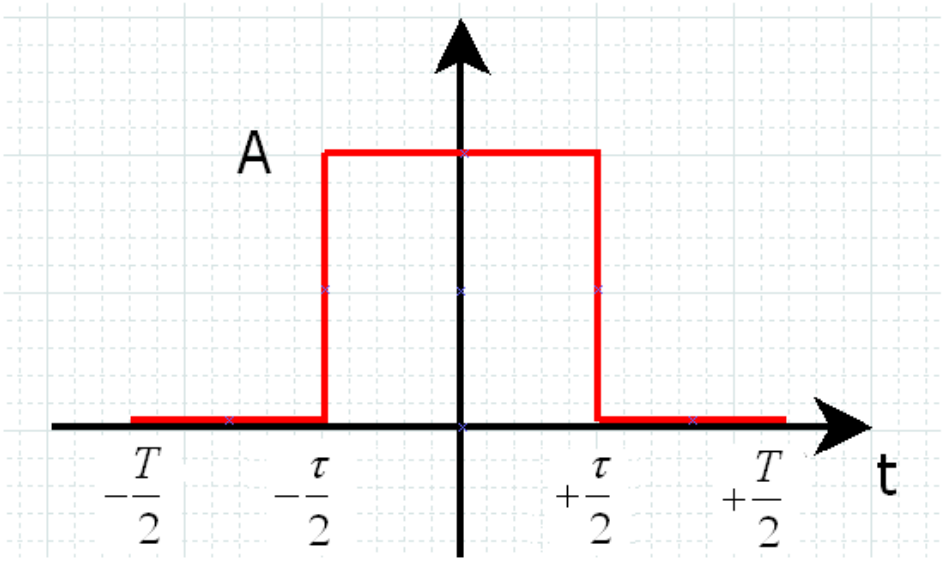
- **Dispari** se f(t) = - f(-t) ,ovvero la funzione è simmetrica rispetto all'origine.
  La funzione **cos(nωt)** è pari,dunque anche **f(t)cos(nωt)** risulterà pari. Dunque,per considerazioni analoghe a quelle precedenti,i termini An saranno nulli e quindi la serie di Fourier del segnale sarà:

$$f(t) = A_0 + \sum_{n=1}^{\infty} B_n \cdot sen(n \cdot \omega \cdot t)$$

- **Forma omplessa:**

$$f(t) = C_0 + \sum_{n=-\infty}^{\infty} C_n \cdot e^{j \cdot \Omega \cdot n \cdot t} \qquad C_n = \frac{1}{T} \cdot \int_T f(t) \cdot e^{-j \cdot n \cdot \Omega \cdot t} dt$$ Dove:

Sviluppo in serie di fourier di un treno d'impulsi



I coefficienti Cn complessi Di Fourier sono calcolato nel seguente modo:

$$C_n = \frac{1}{T} \int_{-\frac{\tau}{2}}^{+\frac{\tau}{2}} A \cdot e^{-j \cdot n \cdot \Omega \cdot t} dt \Rightarrow C_n = \frac{A}{T} \cdot \left[\frac{e^{-j \cdot n \cdot \Omega \cdot t}}{-j \cdot n \cdot \Omega} \right]_{-\frac{\tau}{2}}^{+\frac{\tau}{2}} \Rightarrow$$

Definendo $\delta = \frac{\tau}{T}$ si ha che:

$$C_n = \frac{A}{T} \cdot \left[\frac{e^{-j \cdot n \cdot \pi \cdot \delta} - e^{j \cdot n \cdot \pi \cdot \delta}}{-j \cdot n \cdot 2 \cdot \pi \cdot \frac{1}{T}} \right] \Rightarrow C_n = A \cdot \frac{e^{-j \cdot n \cdot \pi \cdot \delta} - e^{j \cdot n \cdot \pi \cdot \delta}}{-2 \cdot j \cdot \pi \cdot n}$$

Ricordando che secondo Eulero:

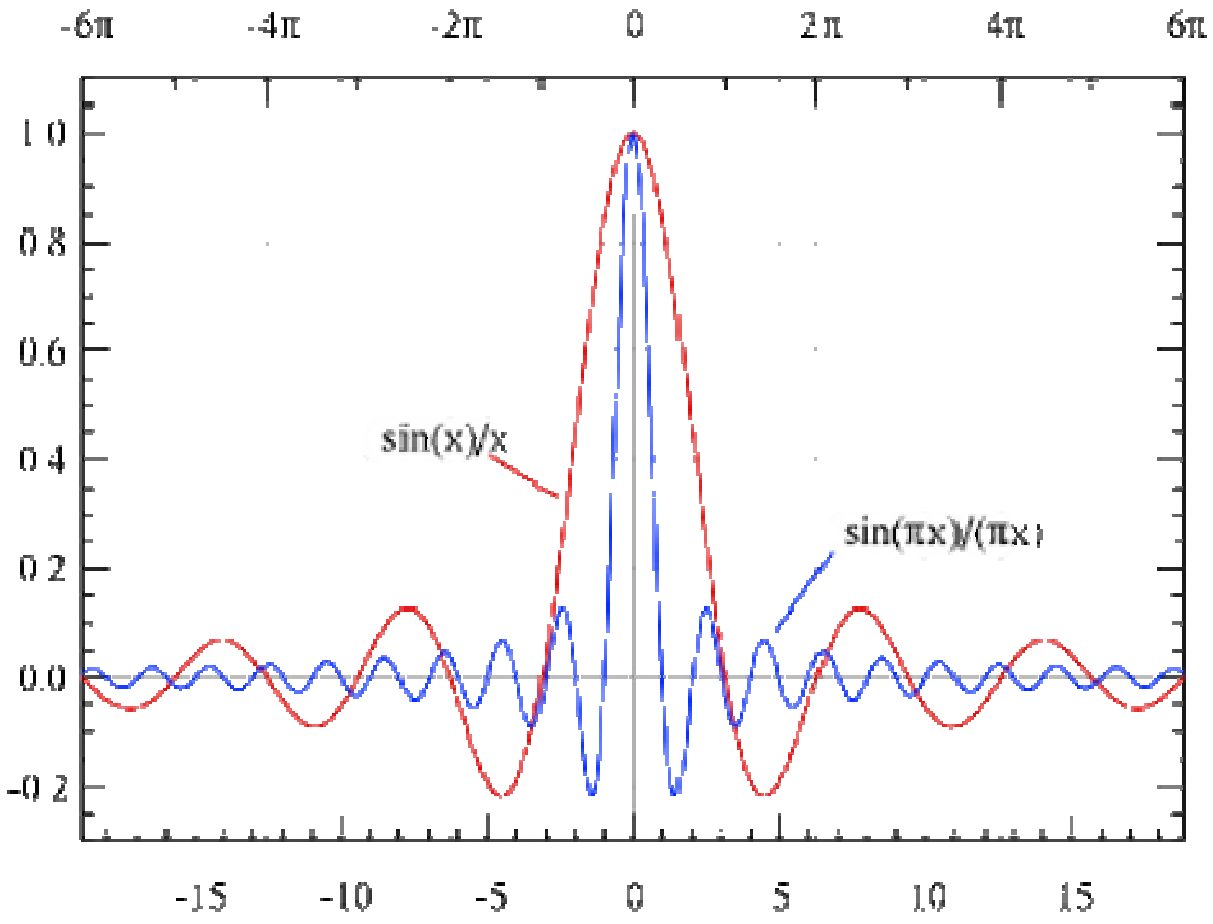
$$\frac{e^{jx} - e^{-jx}}{2j} = sen(x) \Rightarrow \frac{e^{j \cdot n \cdot \pi \cdot \delta} - e^{-j \cdot n \cdot \pi \cdot \delta}}{2 \cdot j} = sen(n \cdot \pi \cdot \delta)$$

Grazie a questa formula è possibile passare dalla forma complessa, alla forma trigonometrica (seno e coseno):

$$C_n = \frac{A}{\pi \cdot n} \cdot sen(n \cdot \pi \cdot \delta)$$

A questo punto è utile introdurre la funzione SINC(X):

$$SINC(x) = \frac{sen(x)}{x}$$



Essa consente di passare dalla rappresentazione complessa a quella trigonometrica dei coefficienti:

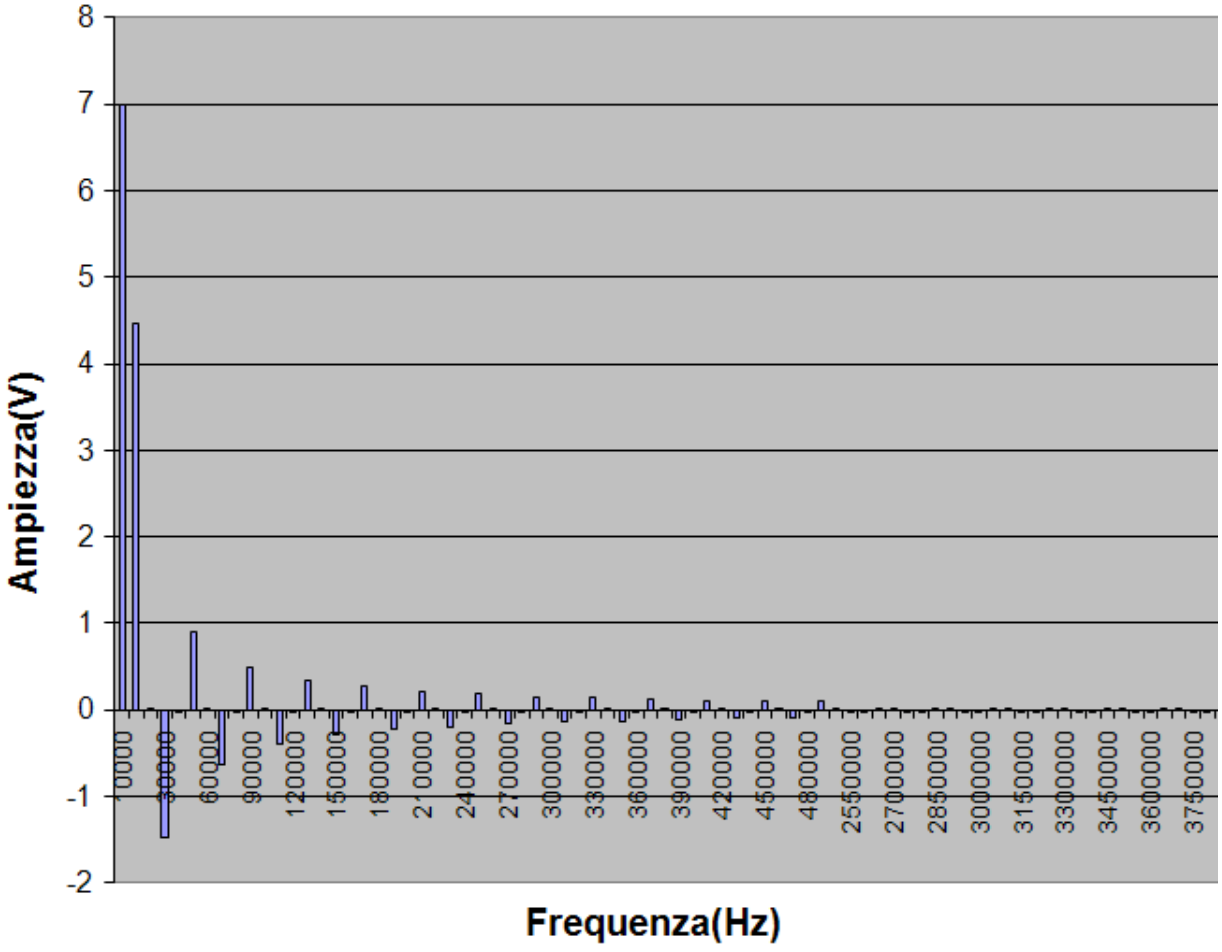
$$C_n = A \cdot \pi \cdot \delta \cdot \frac{sen(n \cdot \pi \cdot \delta)}{\pi \cdot n \cdot \delta} \Rightarrow$$
$$C_n = A \cdot \delta \cdot SINC(n \cdot \pi \cdot \delta)$$

Alcuni segnali non hanno tutte le armoniche n, infatti come è possibile notare dal grafico della funzione SINCX esistono dei valori di X (o di πX) per i quali la funzione si annulla, causando l'annullamento del coefficiente e di conseguenza dell'armonica stessa.

i coefficienti trigonometrici delle armoniche si calcolano nel modo seguente :

$$A_n = \int_{-\frac{\tau}{2}}^{+\frac{\tau}{2}} \frac{2 \cdot A}{T} \cdot \cos(\omega \cdot n \cdot t)\ dt \Rightarrow A_n = \frac{2 \cdot A}{T} \left[\frac{\sin(\omega \cdot n \cdot t)}{2 \cdot \pi \cdot n} \right]_{-\frac{\tau}{2}}^{+\frac{\tau}{2}} \Rightarrow$$
$$A_n = \frac{2 \cdot A}{\pi \cdot n} \cdot \sin(\pi \cdot n \cdot \frac{\tau}{T}) \Rightarrow A_n = \frac{2 \cdot A \cdot \delta}{\pi \cdot n \cdot \delta} \cdot \sin(\pi \cdot n \cdot \delta) \Rightarrow$$
$$A_n = 2 \cdot A \cdot \delta \cdot SINC(\pi \cdot n \cdot \delta)$$

La proiezione dei segnali armonici sul piano ampiezza-frequenza rappresenta lo "spettro del segnale".
Per esempio,supponiamo di dover esaminare un segnale con frequenza 10KHz,ampiezza 5V e duty-cycle al 70%:

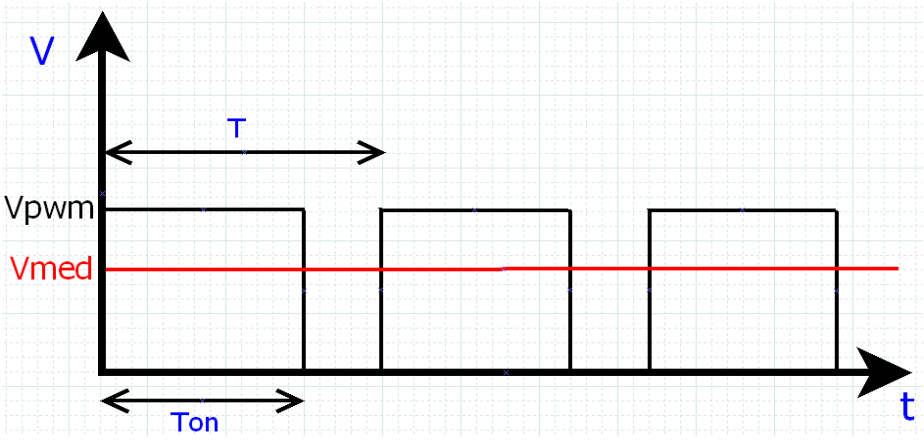


PWM

"PWM" è l'acronimo di "Pulse Width Modulation" ovvero modulazione della larghezza d'impulso:
è un'onda rettangolare con duty-cycle variabile che permette di variare l'assorbimento di un carico elettrico. Il duty-cycle è il rapporto tra il tempo in cui l'onda assume valore alto e il periodo T (T=1/f) ne segue che ad esempio un duty-cycle dell'80% corrisponde ad un'onda rettangolare che assume valore alto per l'80% del tempo e basso per il restante 20%.

Direzione e intensità di rotazione di un motore sono proporzionali al verso e alla densità di corrente che lo attraversano,ma,generare una corrente controllata dell'ordine dell'ampere è piuttosto complicato dunque per pilotare un motore si ricorre al PWM.
Il vantaggio di questa tecnica è di ridurre drasticamente la potenza dissipata dal circuito limitatore rispetto all'impiego di transistor controllati analogicamente. In un semiconduttore infatti, la potenza dissipata è determinata dalla corrente che lo attraversa per la differenza di potenziale presente ai suoi capi. In un circuito PWM il transistor o è in saturazione, riducendo al minimo la caduta ai suoi capi, oppure in interdizione, annullando la corrente, ed in entrambi i casi la potenza dissipata è minima.

Analisi di un segnale PWM:



Il valore medio del segnale è dato dalla seguente formula:

$$Vmed = \frac{1}{T} \times \int_0^T Vpwm\ (t)\ dt \Rightarrow$$

$$Vmed = \frac{1}{T} \times \left(\int_0^{Ton} Vpwm\ dt + \int_{Ton}^{T} Vpwm\ dt \right) \Rightarrow$$

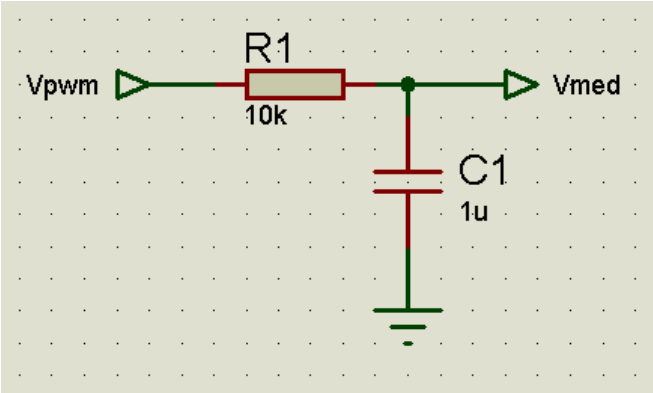
$$Vmed = \frac{1}{T} \times \int_0^{Ton} Vpwm\ dt \Rightarrow$$

$$Vmed = \frac{1}{T} \times Vpwm \times t\Big|_0^{Ton} \Rightarrow Vmed = \frac{Ton}{T} \times Vpwm \Rightarrow$$

$$Vmed = Vpwm \times \delta$$

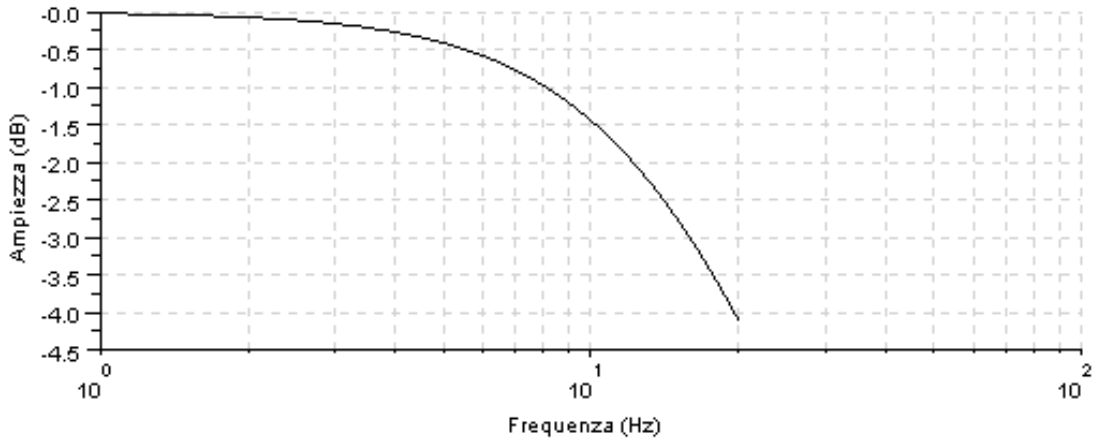
Dalla formula si evince che il valore medio è direttamente proporzionale al duty-cycle. Applicando il segnale PWM a un filtro passa-basso,opportunamente dimensionato è possibile “misurarne” il valore medio.

Filtro passa basso del primo ordine:



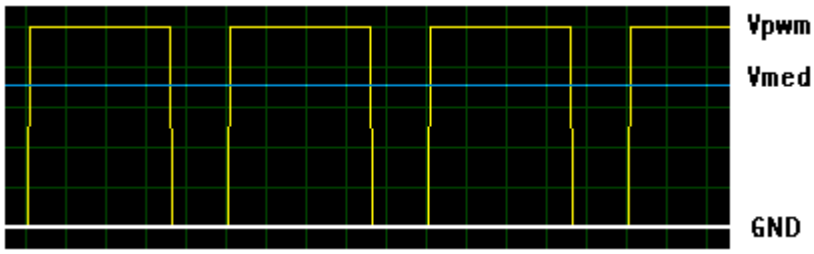
$$G(s) = \frac{100}{S + 100}$$

Diagramma di Bode del filtro

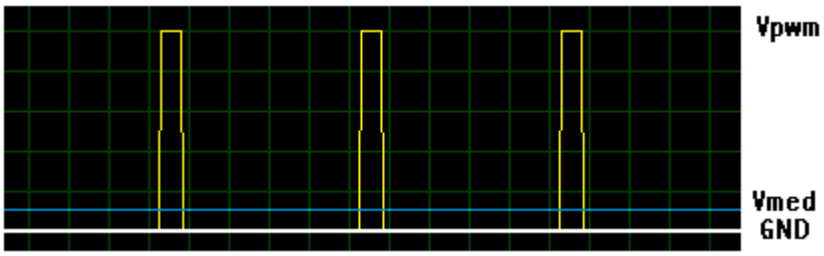


La frequenza di taglio del filtro è ft = 15,92Hz ,dunque poiché funzioni da integratore la frequenza di lavoro deve trovarsi almeno una decade sopra la frequenza di taglio. In questo modo tutte le armoniche del segnale vengono tagliate e passerà solo la componente continua.
Supponiamo dunque di applicare un segnale Vpwm di ampiezza 5V, di frequenza f = 10KHz e duty-cycle variabile:

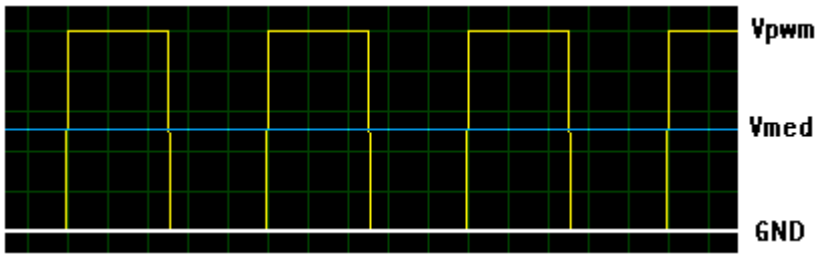
$$Duty - cycle = 70\% \quad Vmed = 5 \times 0{,}7 = 3{,}5V$$



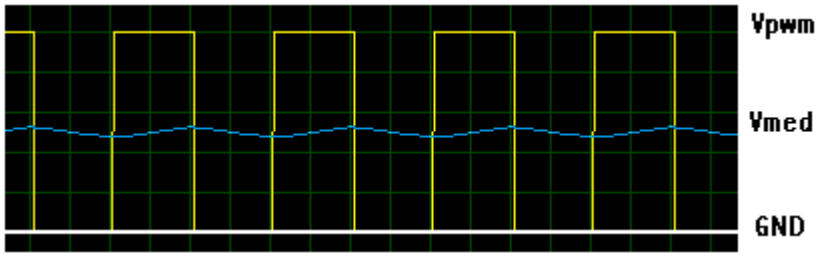
$$Duty-cycle = 10\% \quad Vmed = 5 \times 0{,}1 = 0{,}5V$$



$$Duty-cycle = 50\% \quad Vmed = 5 \times 0{,}5 = 2{,}5V$$

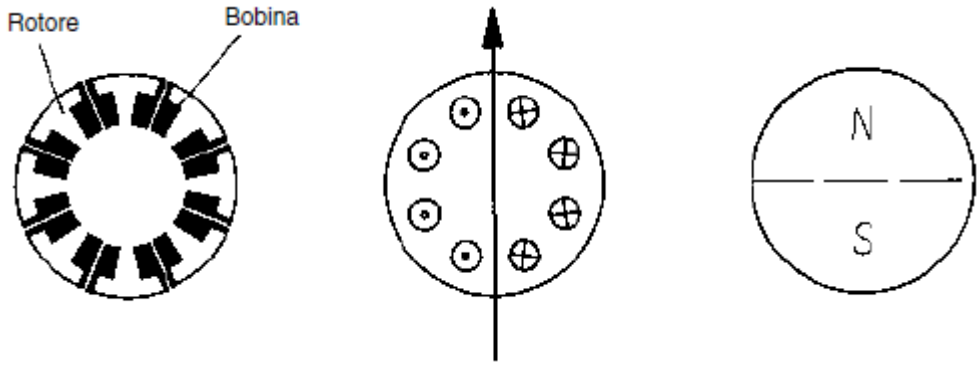


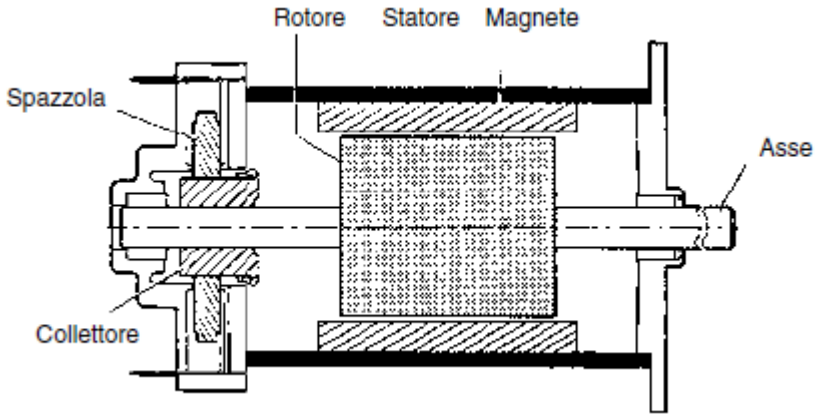
È utile notare che Se si abbassa la frequenza a 500 Hz,mantenendo il duty-cycle al 50%,essendo vicini alla frequenza di taglio, qualche armonica del segnale Vpwm riesce a passare come è possibile notare dalla figura seguente,in quanto Vmed non è continuo.



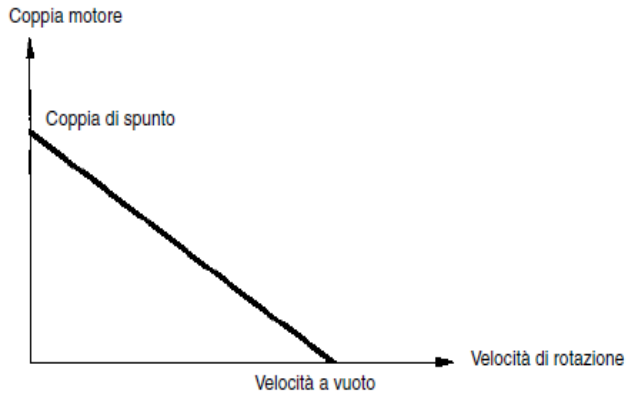
Motoriduttori

I motori in corrente continua (dc) sono costituiti da uno statore formato da una carcassa metallica contenente uno o più magneti che creano un campo all'interno dalla stessa e da un rotore,anche esso costituito da una carcassa metallica , il quale sostiene delle bobine collegate fra loro a livello del collettore che una volta alimentate generano un campo magnetico.
Per effetto dell'attrazione dei poli contrari e della repulsione dei poli di stesso segno, sul rotore si genera una coppia che lo mette in moto.
Non appena il rotore si mette in moto alcune spazzole, operano su lame diverse del collettore,alimentando le bobine in maniera che l'asse dei nuovi poli del rotore sia comunque perpendicolare a quello dello statore, causando quindi la continua rotazione del motore.
La fluttuazione della coppia risultante diminuisce aumentando il numero di lame del collettore.

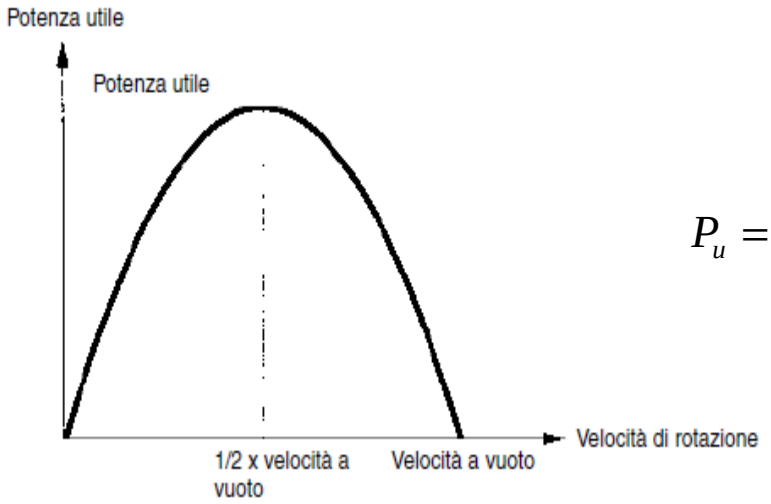




La coppia che fornisce il motore e la sua velocità di rotazione sono dipendenti l'una dall'altra.
Si tratta di una caratteristica essenziale per questo tipo di motore. Tale relazione è lineare e permette di conoscere sia la velocità a vuoto che la coppia di spunto del motore.



Da questo grafico si deduce la curva della potenza utile:

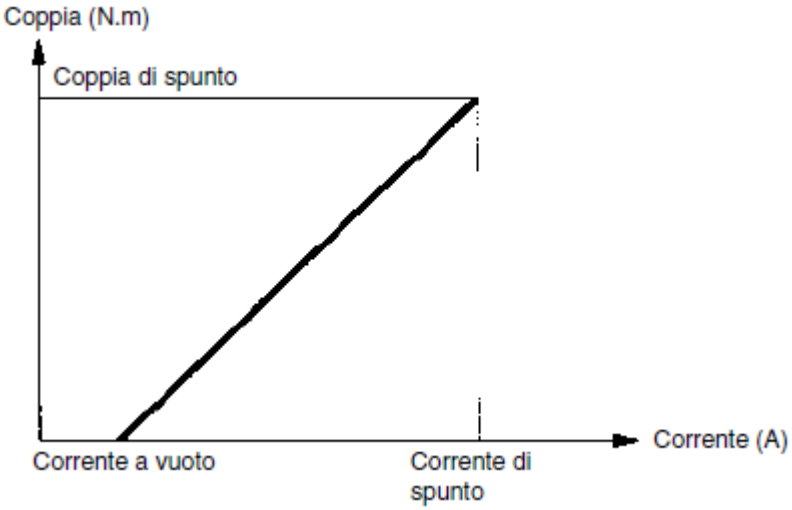


$$P_u = \frac{2\pi}{60} \times C[N \cdot m] \times N[rpm]$$

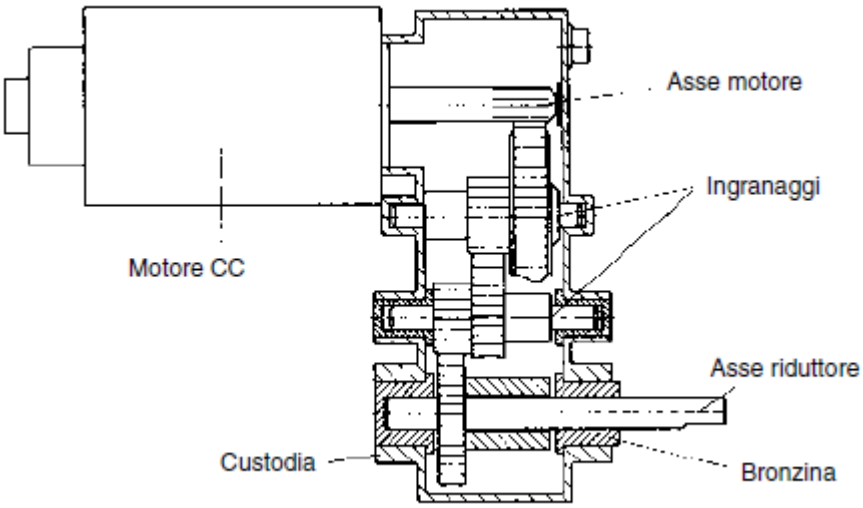
Le curve coppia-velocità e potenza utile dipendono dalla tensione di alimentazione del motore.
La tensione di alimentazione fornita per un certo motore corrisponde a un impiego continuo di tale motore in funzionamento nominale alla temperatura ambiente di 20°C.
E' peraltro possibile alimentare il motore con una tensione diversa, in generale compresa tra -50% e +100% della tensione prevista: ovviamente se lo si sottoalimenta il motore sarà meno potente mentre se lo si sovralimenta sarà più potente ma si riscalderà maggiormente .
Per variazioni della tensione di alimentazione comprese tra -25% e +50%, la nuova curva coppia-velocità sarà parallela a quella nominale. Di conseguenza, sia la coppia di spunto che la velocità a vuoto varieranno dello stesso valore percentuale n%. Per quanto riguarda la potenza utile massima del motore, questa si ottiene :

$$P_{\max} = P_u \times \left[(1+n\%)^2 \right]$$

Un'altra caratteristica importante è data da grafico coppia-corrente,che permette di conoscere la corrente assorbita a vuoto, oppure a motore bloccato



Il rapporto tra la potenza meccanica utile che può essere fornita e la potenza assorbita è detto rendimento.
I motori in corrente continua sono costruiti per funzionare permanentemente a velocità prossime alla propria velocità a vuoto, che essendo nella maggior parte dei casi troppo elevata si applica un motoriduttore.



Dimensionamento

Il dimensionamento del motoriduttore richiede un po' di nozionismi di fisica ma data l'entità dell'applicazione le nozioni da sapere sono abbastanza banali e approssimabili senza problemi. I motori debbono fornire una forza maggiore del prodotto massa per accelerazione, più altre forze che si oppongono al moto come quelle di attrito al rotolamento delle ruote(attrito volvente) .

$$\frac{M}{r} = a \times m + Fr$$

Dove:

- **M** = Coppia del motore
- **r** = raggio della ruota
- **a** = accelerazione
- **m** = massa
- **Fr** = forze resistenti

Supponendo che il robot non debba pesare complessivamente più di 1kg,si voglia un'accelerazione a = 0.5m/s^2 , il raggio delle ruote è 5.1cm e approssimando a zero le forze che si oppongono al moto :

$$M = a \times m \times r \Rightarrow M = 0{,}5\left[\frac{m}{s\text{^}2}\right] \times 1[Kg] \times 5{,}1[Cm] \Rightarrow 2{,}55[N \cdot cm]$$

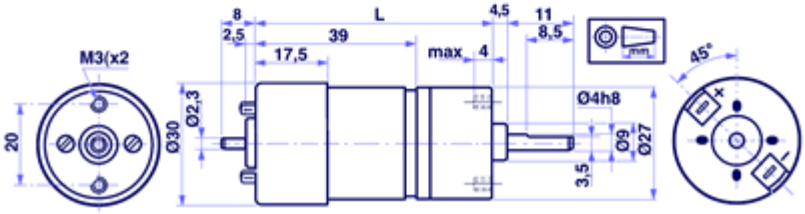
Essendo due I motori che muoveranno il robot, il peso si ripartirà metà su un motore e metà sull'altro (sempre in situazioni ideali) dunque la coppia di ciascun motore dovrà essere :

$$\frac{M}{2} = \frac{2{,}55}{2} = 1{,}275[N \cdot cm] \Rightarrow 1{,}301[Kg \cdot m]$$

Per sicurezza,tenendo anche conto dell'approssimazione a zero delle forze resistenti,si è scelto un motoriduttore con una coppia di 1,84 Kg*m.

La seguente tabella riporta le caratteristiche del motoriduttore in questione:

Caratteristiche Tecniche:		
Codice Articolo	420120	**420121**
Tensione Nominale Vdc	12	12
Consumo a vuoto mA	140	140
Consumo Max mA	580	440
Coppia Max Kg/cm	1.84	2.55
Rapporto Riduzione	43.3:1	90.3:1
Velocita' a vuoto RPM	155	75
Velocita' a coppia Max RPM	115	62
Diametro Asse mm	4	4
Lunghezza mm	67.5	67.5
Peso gr	100	100



A questo punto,conoscendo anche il numero di RPM,è possibile calcolare la velocità di punta moltiplicando il numero di giri al secondo dell'asse per la circonferenza della ruota. Dividendo il numero di RPM (giri al minuto) per 60 si ottengono il numero di giri al secondo.

$$\frac{115}{60} = 2.5\left[\frac{giri}{\sec}\right]$$

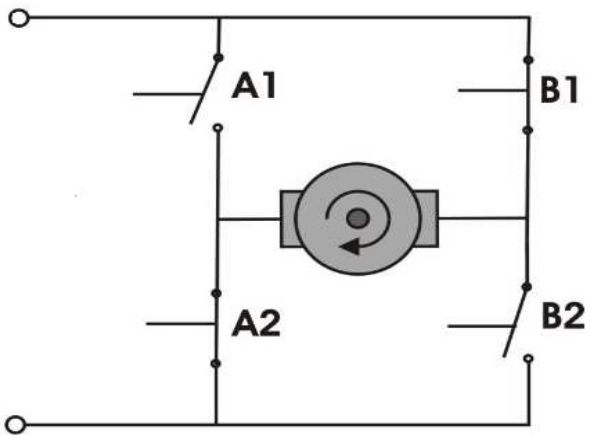
$$Circonferenza\_ruota = 2\pi r = 0{,}32[m]$$

$$Velocità\_\max = \left[\frac{Giri}{\sec}\right] * circonferenza\_ruota \Rightarrow 2{,}5\left[\frac{Giri}{\sec}\right] * 0{,}32[m] = 0{,}8\left[\frac{m}{\sec}\right]$$

Pilotaggio dei motori DC

Il pilotaggio più semplice per un motore è quello ON-OFF che permette di mandare il motore alla massima velocità oppure fermarlo.

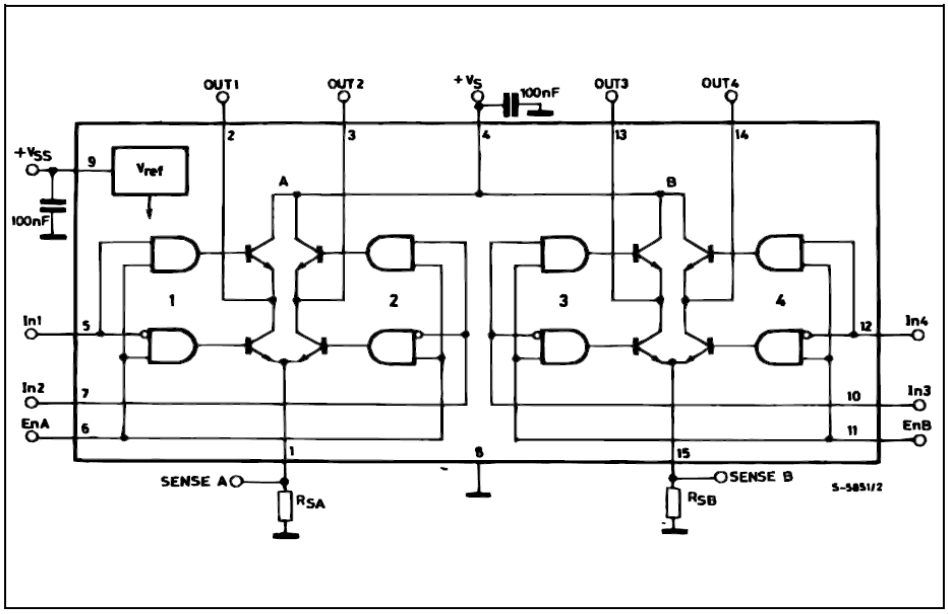
Il circuito risulta abbastanza semplice e consiste in un' interruttore(Transistor,relè ecc) in serie al motore e un diodo di ricircolo sul motorino (si ricorda che il motore in DC è un carico induttivo) per evitare danni sul resto del circuito. Se si vuole invertire anche il verso di rotazione si ricorre ad un "ponte H", che consente di invertire il verso della corrente che attraversa il motore stesso. In seguito è riportato lo schema semplificato di un ponte H:



In questo caso è chiusa la coppia di interruttori A2 e B1 consentendo alla corrente di circolare in un verso,viceversa se fosse chiusa la coppia di interruttori A1 e B2 la corrente circolerebbe in un'altro verso.

L298

L'l298 è un driver per motori DC o stepper,costituito da due ponti H integrati,che supportano un elevato voltaggio (46V) ed elevate correnti (2A per ponte) e che possono essere pilotati con livelli in logica TTL. Ciascun ponte può essere disabilitato o abilitato tramite il relativo piedino di enable.



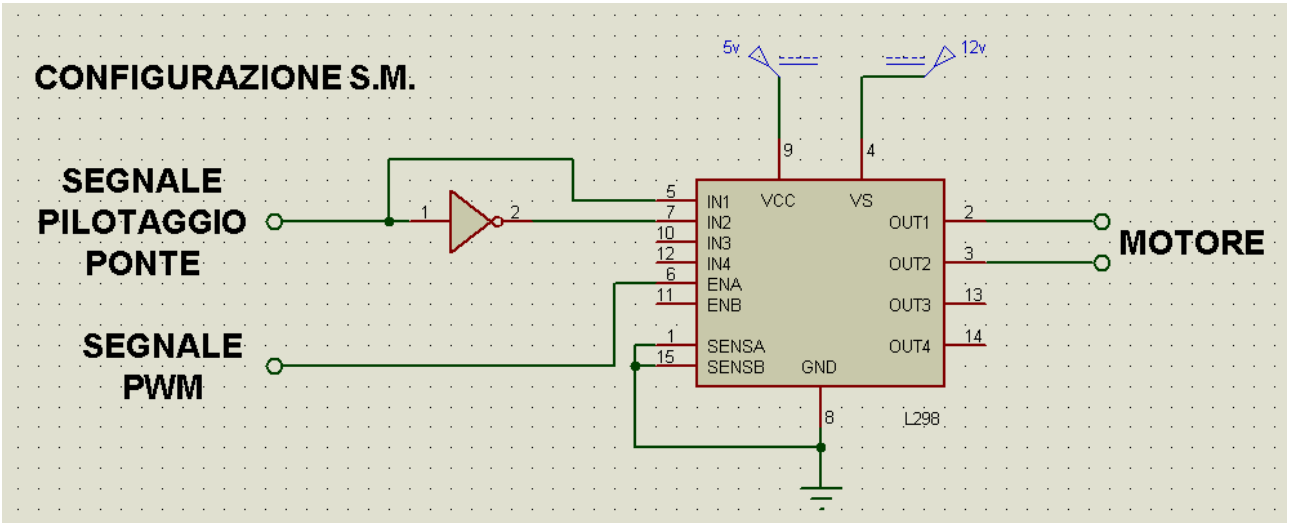
Caratteristiche tecniche dell'L298:

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

L298 in PWM

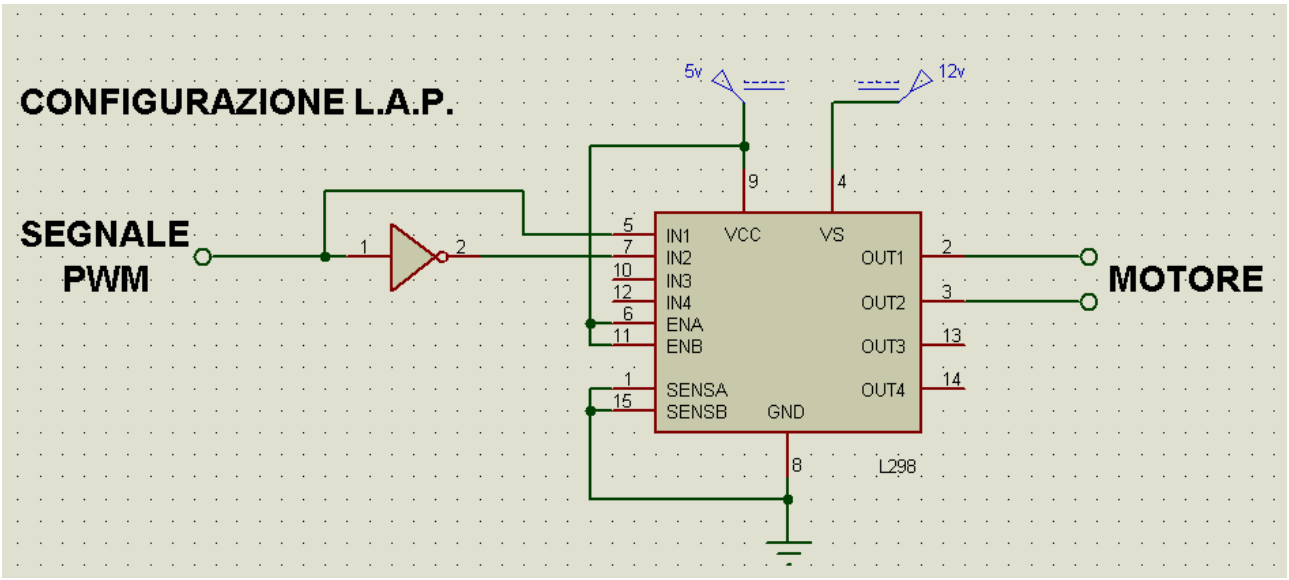
Come detto in precedenza pilotando un ponte H in pwm è possibile variare velocità e direzione di un motore DC. Per fare ciò esistono due configurazioni possibili :

- Configurazione S.M.



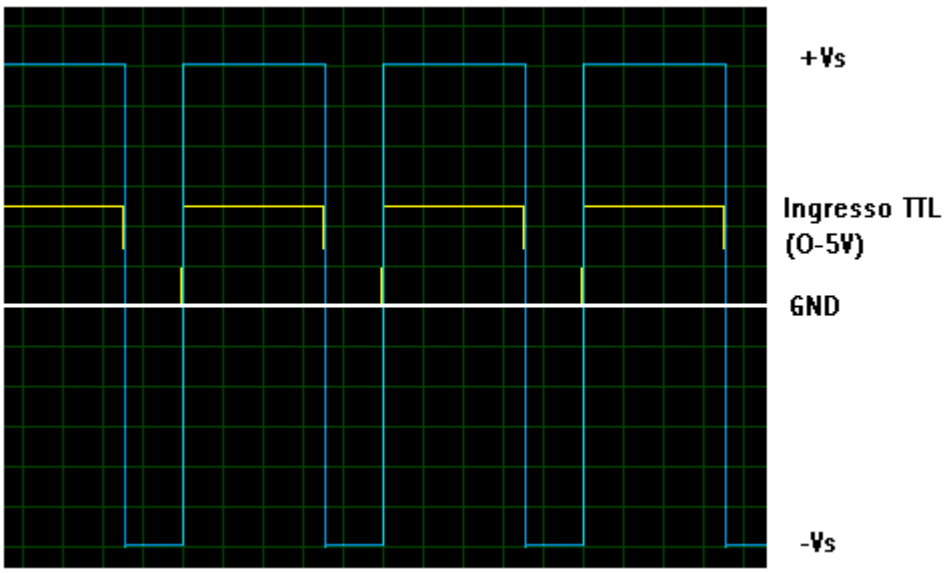
L' SM (sign-magnitude) consiste nell'inviare il segnale pwm all'ingresso di enable per variare la velocità e di pilotare separatamente gli ingressi del ponte per stabilire il verso di rotazione,che dovendo essere comandati con segnali invertiti, utilizzando una not si riduce il numero segnali per ogni motore.

- Configurazione L.A.P.



Nella configurazione LAP (locked anti-phase) invece il segnale PWM viene messo in ingresso all'invertitore in modo da avere ai due lati opposti del ponte due segnali invertiti tra loro e agendo sull'enable è possibile spegnere o accendere il rispettivo ponte (se quest'ultima operazione non è necessaria è possibile fissare alto l'enable), infatti è lo stesso segnale PWM che contiene l'informazione necessaria per determinare verso e velocità del motore.

In questo modo all'uscita dell'L298 si ottiene un'onda rettangolare bipolare con duty-cycle uguale a quello del segnale di ingresso (che è di tipo TTL) e Ampiezza pari alla tensione applicata sul pin VS (nel nostro caso 12v).



In figura : il segnale giallo è l'ingresso TTl, ossia l'uscita del pic,mentre quello blu è l'uscita dell'L298 (in L.A.P.).

Considerando che essendo una commutazione ad alta frequenza,la fondamentale e le armoniche vengono tagliate dal circuito, rimane solo la componente continua, dunque è come se fosse un generatore di tensione continua che va da –vs a +vs, variabile al variare del duty-cycle.

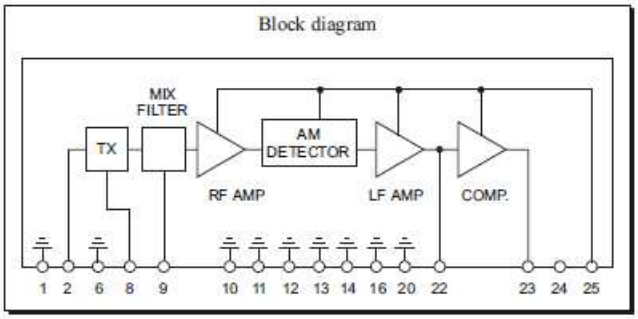
In particolare:

- Con un Duty-Cycle allo 0% la velocità è massima in un verso
- Con un Duty-Cycle al 50% il motore è fermo
- Con un Duty-Cycle al 100% la velocità è massima nell'altro verso

Per questo progetto ho scelto quest'ultima modalità, in quanto bisogna gestire un solo segnale e ciò semplifica di molto le cose, visto che i motori da controllare sono due e il controllo avviene da remoto tramite etere.

Moduli RTF-DATA-SAW Aurel

Sono moduli a basso costo per la ricetrasmissione di dati ,compatibili con l'RS232, per sistemi radio tx-rx alf-duplex con commutazione veloce d'antenna che quindi è unica. La banda passante è adeguata a ricevere fino a 2400 baud con decodifica Manchester.



Pin-out

1) Ground	12) Ground
2) TX data input	13) Ground
0V=Tx Off	14) Ground
5V= Tx continuous On	16) Ground
6) Ground	20) Ground
8) Tx +5V supply	22) RX analog out
9) Antenna	23) RX digital out
10) Ground	24) N.U.
11) Ground	25) RX +5V supply

Caratteristiche Tecniche:

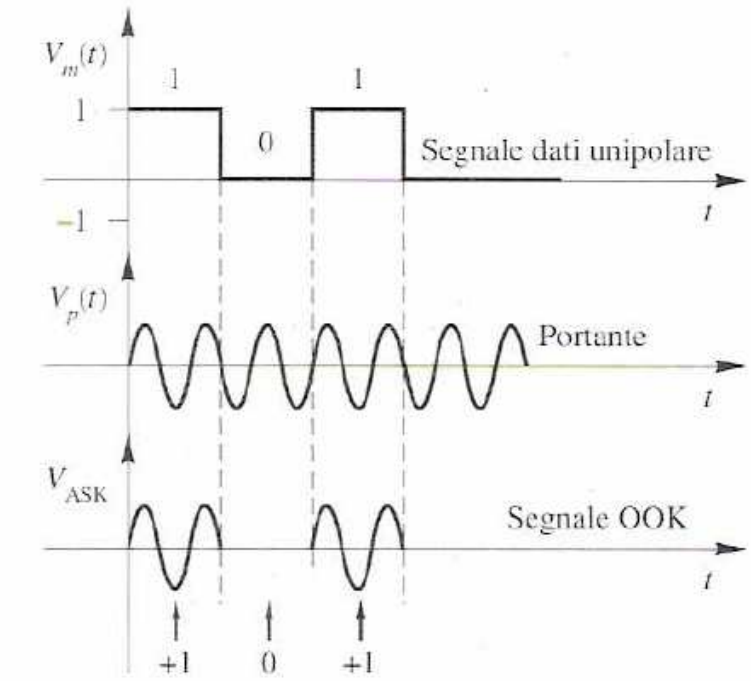
- Realizzazione in circuito ibrido su allumina ad elevata miniaturizzazione ;
- Frequenza disponibile : 433,92 MHz ;
- Potenza TX : 8 mW (9 dBm ± 2 dB) su carico da 50 W ;
- Banda passante BF : onda quadra 5 KHz (Mod. RTF) onda quadra 3 KHz (Mod. RTL) ;
- Tempo di commutazione Tx-Rx : migliore di 100 ms,con Rx sempre On ;
- Sezione Rx di tipo supereattivo ;
- Sezione Tx con risuonatore SAW ;
- Sensibilità RF misurata con segnale On-Off in ingresso :
  migliore di 7 µV (-90 dBm) per il mod. RTF-DATA-SAW,
  migliore di 2,24 µV (-100 dBm) per il mod. RTL-DATA-SAW ;
- Formato "in line" con dimensioni: 63,5 x 17,9 x 5 mm. Pin passo 2,54 mm ;

Assorbimento a +5V :

- Sezione Tx : 4.5 mA con onda quadra in ingresso ;
- Sezione Rx : 2.5 mA ;
- In stand-by : consumo nullo (sia Tx che Rx) ;

Modulazione OOK

Questi moduli eseguono una modulazione di ampiezza di tipo on-off,modulando i bit (che in questo caso sono inviati dal pic) con un segnale portante sinusoidale, ottenendo una modulazione del tipo OOK,che consiste nell'invio del segnale portante in presenza dell'1 logico e nel non inviare nulla in presenza di uno 0 logico:



L’antenna,è costituita da un filo conduttore di lunghezza adeguata : essendo la frequenza portante f=433,92 MHz,si ha una lunghezza d’onda (λ)di circa 69 Cm, per motivi pratici sarà utilizzata un’antenna di 17Cm che corrisponde a circa λ/4.

Display LCD

La differenza tra cristalli liquidi a differenza dei tubi a raggi catodici che emettono luce,assorbono o riflettono la luce proveniente da un’altra sorgente.
L’informazione viene visualizzata in base allo stato dei pixel,che sono organizzati in righe e colonne e a seconda della loro disposizione il visualizzatore è di tipo “alfanumerico” se hanno una disposizione del tipo 5x8 consentendo di distinguere le righe del carattere stesso, mentre è dun visualizzatore di tipo “grafico” se sono organizzati in modo omogeneo e continuo,non consentendo una distinzione delle righe del carattere e dunque consentendo una maggiore risoluzione.
Il modo al momento più diffuso ed economico per pilotare questi pixel è detto “a matrice passiva” che consiste nell’applicazione di campi elettrici ad un reticolo di elettrodi (riga per colonna) posti ai lati del cristallo liquido,mentre se si vuole ottenere una risoluzione migliore bisogna utilizzare il metodo a “matrice attiva” ossia , ogni pixel è pilotato da un proprio transistor ma,come è possibile intuire, un componente attivo per ogni pixel rende i costi notevolmente più alti. Un circuito di decodifica che contiene in memoria i principali caratteri ASCII (è anche possibile editare o creare caratteri) che vengono caricati in RAM prima di essere visualizzati.

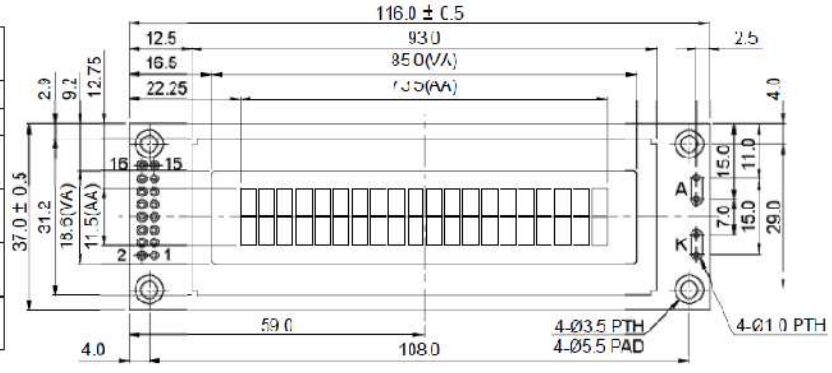
Per questo progetto ho scelto LCD 20x2 (20 colonne x 2 righe) compatibile con lo standard HITACHI 44780, pilotabile facilmente con un microcontrollore grazie anche alle numerose librerie apposite facilmente reperibili.
Per questo progetto non sono richieste grandi prestazioni del display,dunque è stata scelta una particolare configurazione nella quale non viene utilizzato il pin R/W (in quanto il display deve solo ricevere istruzioni e dati e non deve ritrasmettere nulla) che viene forzato a massa , mentre il dato viene inviato solo tramite le linee D4-D7 omettendo le altre linee. Infatti il Byte che contiene il carattere ASCII viene “spaccato” dal micro in due pacchetti chiamati nibble.

Piedinatura del display:

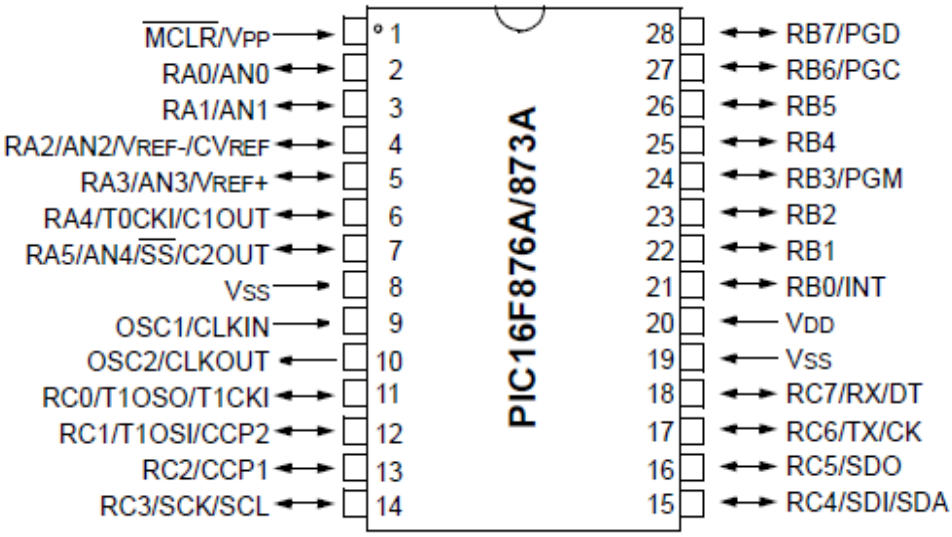
Pin	Descrizione
1	GND
2	VCC
3	Regolazione contrasto
4	R/S (Instruction/Register select)
5	R/W(Read/Write)
6	E(clock)
7	D0(Dato 0)
8	D1(Dato 1)
9	D2(Dato 2)
10	D3(Dato 3)
11	D4(Dato 4)
12	D5(Dato 5)
13	D6(Dato 6)
14	D7(Dato 7)

BC2002A:

Item	Symbol	Condition	Typical Value	Unit
Input Voltage	Vdd	Vdd = +5V	5.0	V
Supply Current	Idd	Vdd = +5V	1.2	mA
LCD Driving Voltage	Vdd-Vo	25°C	4.3	V
LED Forward Voltage	Vf	25°C	4.2	V
LED Forward Current	If	25°C	210	mA
EL Power Voltage	Vel	Vel=110Vac/400Hz		V

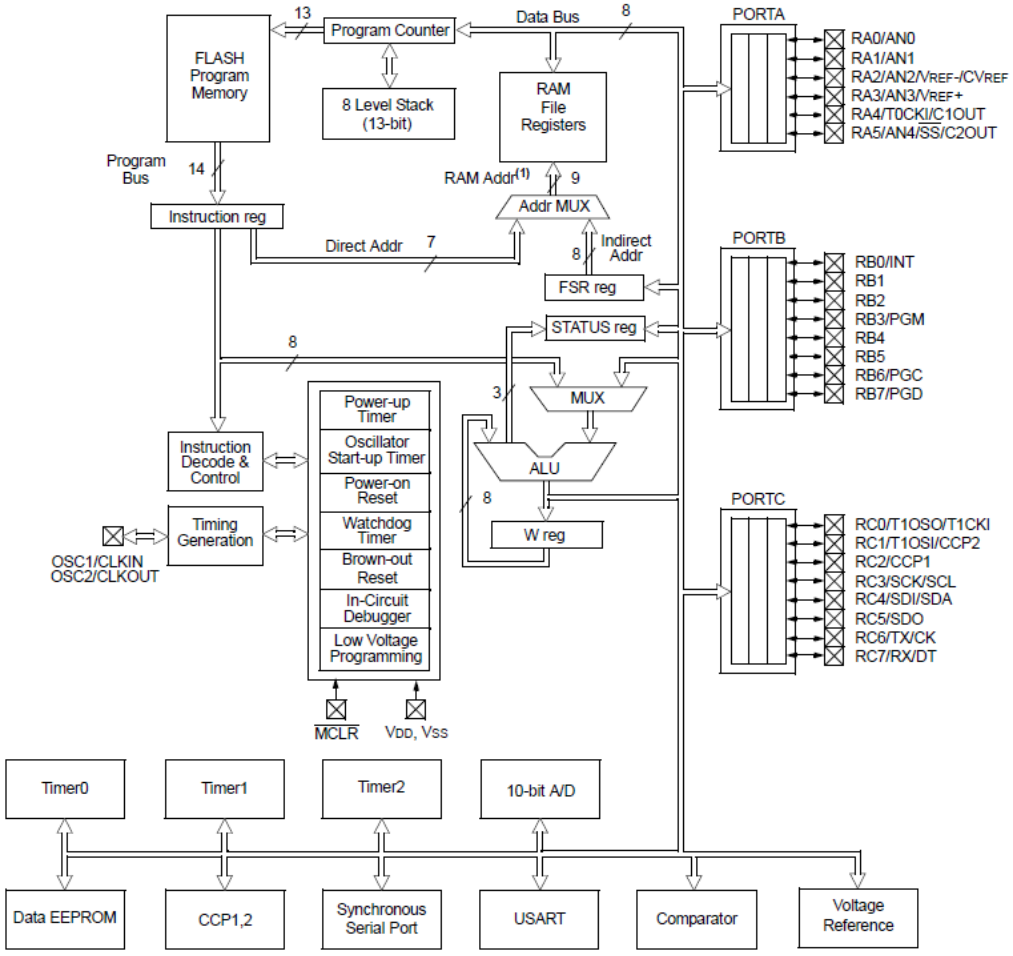


PIC16F876A



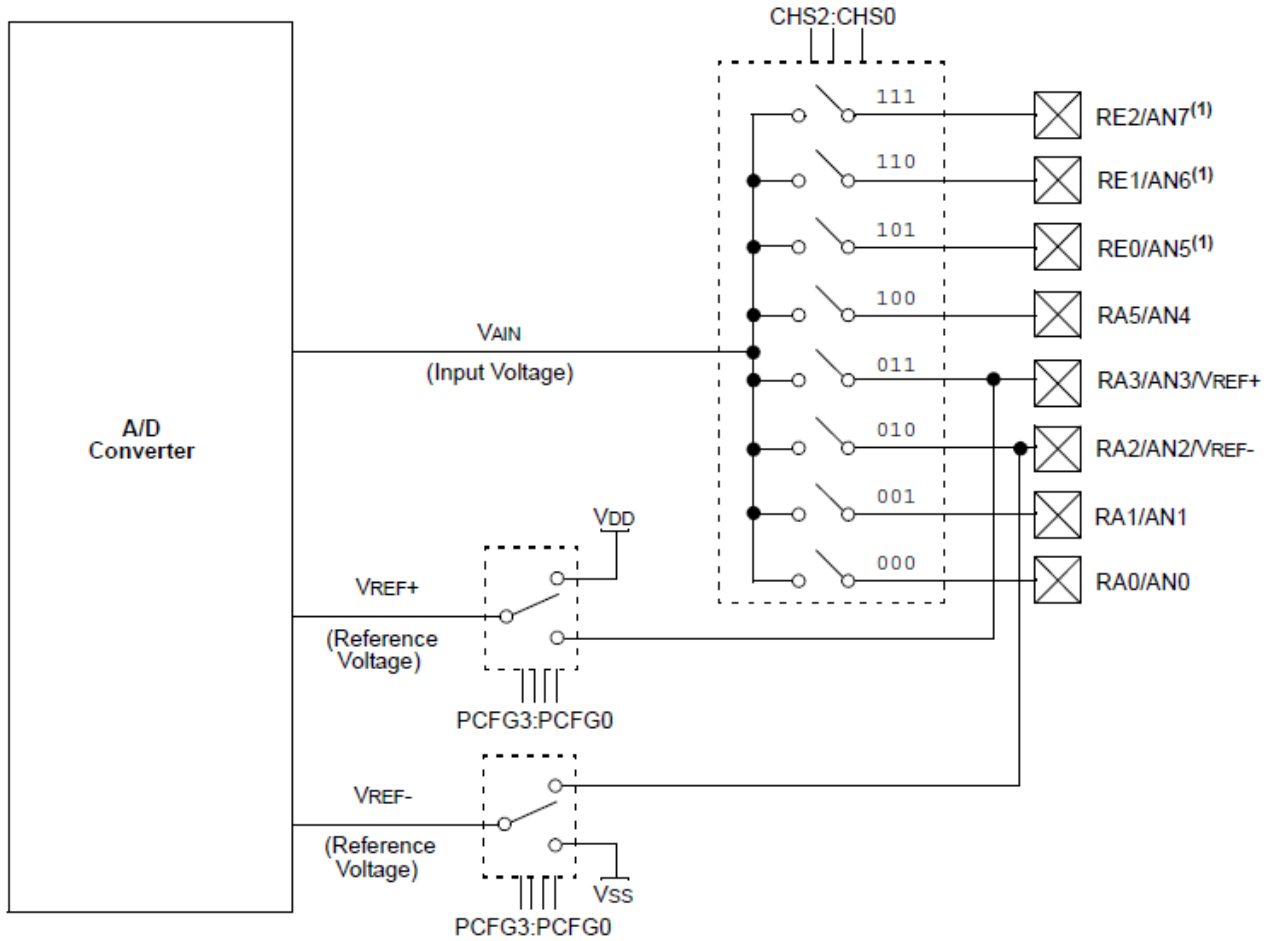
Il PIC16F876A è uno dei pic più completi della famiglia 16FXXX,supporta un clock a 20MHz,22 linee bidirezionali di I/O ,5 linee analogiche a 10 bit e molto altro, ma la scelta di questo pic è dovuta al fatto che possiede ben 2 moduli CCP (PWM) fondamentali per questo progetto.

Architettura interna del pic



Com'è possibile notare il pic possiede 3 porte bidirezionali in cui i diversi pin possono essere settati in modo tale da svolgere particolari funzioni come ad esempio i pin della porta A possono essere configurati come ingressi analogici,RC6 e RC7 possono essere configurati rispettivamente come Tx e Rx per una trasmissione seriale,RC1 e RC2 possono essere configurati in modalità PWM ecc...

Schema a blocchi A/D interno:



I canali analogici possono essere selezionati tramite un multiplexer analogico,mentre le tensioni di riferimento (Vref+ e Vref-) dell'A/D possono essere switchate dall'alimentazione del µC (dunque Vref+= Vdd e Vref- = Vss) alle tensioni presenti su RA3 (Vref+) e RA2(Vref-) a seconda della dinamica d'ingresso desiderata.

In seguito sono riportati i tempi di acquisizione:

TACQ	=	Amplifier Settling Time +
		Hold Capacitor Charging Time +
		Temperature Coefficient
	=	TAMP + TC + TCOFF
	=	2µs + TC + [(Temperature -25°C)(0.05µs/°C)]
TC	=	CHOLD (RIC + RSS + RS) In(1/2047)
	=	- 120pF (1kΩ + 7kΩ + 10kΩ) In(0.0004885)
	=	16.47µs
TACQ	=	2µs + 16.47µs + [(50°C -25°C)(0.05µs/°C)
	=	19.72µs

Generazione PWM con i pic

Per generare un segnale PWM con i pic si può scrivere un codice che generi un'onda rettangolare di opportuna frequenza e duty-cycle su un certo piedino d'uscita, ma, risulta evidente che questa tecnica toglie cicli di clock al resto dell'applicazione,perciò si preferisce utilizzare delle apposite periferiche hardware dei pic , i CCP(Capture Compare Pwm) che una volta settati continuano a generare l'onda da soli,consentendo al pic di svolgere altre operazioni. Tali moduli si appoggiano al timer 2 e quindi hanno la stessa frequenza mentre il duty-cycle viene variato scrivendo su appositi registri. Il periodo di pwm è dato dalla seguente formula:

Tpwm = (PR2+1) * 4*Tosc *[Tmr2 prescaler]

In cui PR2 è il valore inserito nel registro PR2 eTosc è il periodo dell'oscillatore esterno. Per impostare questi parametri il CCS Pic compiler fornisce tre istruzioni che agevolano il lavoro :

- **setup_timer_2 (prescaler,** period**,** postscale**)**

period => PR2 value
postscaler => numero di overflow prima di generare l'interrupt

- **setup_ccp1(mode)**

mode = per scegliere la modalità di funzionamento del modulo,nel nostro caso CCP_PWM.

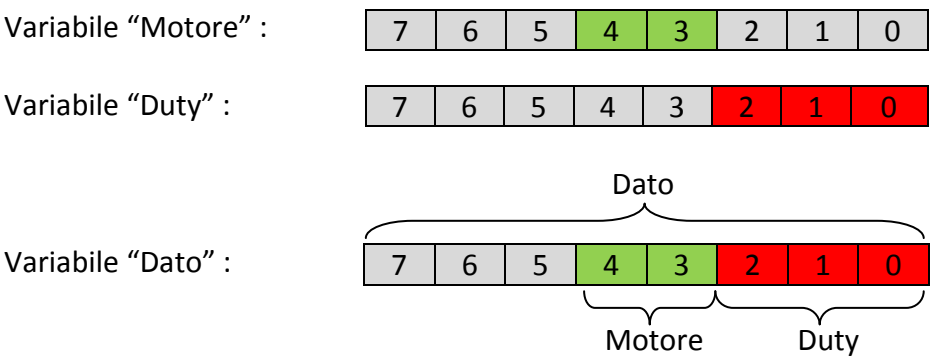
- **set_pwm1_Duty (**value**)**

value è compreso tra 0 e il valore di PR2 e rappresenta il valore del duty-cycle del segnale:
se ad esempio in PR2 c'è 200,se value = 20,il pwm è al 10%,se value = 50% il pwm è al 25% ecc.

Codifica/Decodifica stringhe

Il dato che viene trasmesso è codificato in modo tale da contenere in soli 8 bit velocità e verso del prototipo. La stringa "dato" che viene creata e trasmessa dal µC è formata da otto bit :
i primi tre bit meno significativi contengono il valore del duty-cycle (quindi la velocità) e il 3° e 4° bit servono a determinare su quale dei due motori applicare il valore del duty (direzione).
La variabile "Dato" viene generata da un operazione di or tra due variabili,una contenente appunto la direzione ("motore") e una contenente la velocità ("duty"),strutturate nel seguente modo :

```
dato =  motore | pwm;  // Dato = motore or duty
```



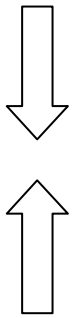
Una volta arrivato a destinazione,il pacchetto dovrà ovviamente essere scompattato per estrarre le informazioni.
Per estrarre "Duty" dalla stringa "dato" basta eseguire un'operazione di and con 7(111) di quest'ultima,mentre per estrarre "motore" occorre eseguire un'and con 24(11000) e shiftare di 3 bit verso destra:

```
duty = dato;
duty&=7;  //duty and 7
```

```
motore = dato & 24;  //dato and 24
motore = motore >>3;  //shift verso destra
```

Le variabili “duty” e “motore” vengono generate nel seguente modo :
“Duty”:
Come visto in precedenza il valore del duty-cycle impostabile va da 0 al valore caricato su PR2 che è un registro a 8 bit,ma, in questo caso, risulta del tutto inutile variare il duty-cycle dell’1% occupando tutti gli 8 bit della trasmissione RS232 solo per la velocità, dunque ho scelto una scala di 8 valori che vengono impostati in base al range di tensione presente sul canale RA0,variabile mediante un potenziometro:

“Duty”	Duty-cycle	Tensione letta sul canale RAØ
0 0 0	0%	0,6V
0 0 1	10%	1,2 V
0 1 0	20%	1,8 V
0 1 1	30%	2,4 V
1 0 0	70%	3V
1 0 1	80%	3,6 V
1 1 0	90%	4,2 V
1 1 1	100%	4,8 V



“MOTORE”:
Una volta impostata la velocità e il verso ,per determinare la direzione occorre selezionare il motore a cui applicare questi parametri,selezione che avviene In base allo stato logico presente su RB7 E RB6 :

RB7	RB6	MOTORE
0	0	50% (robot fermo)
0	1	DX
1	0	SX
1	1	SX/DX (marcia)

- Si noti che i tre bit rimanenti di “dato” possono essere utilizzati per altre istruzioni o dati.

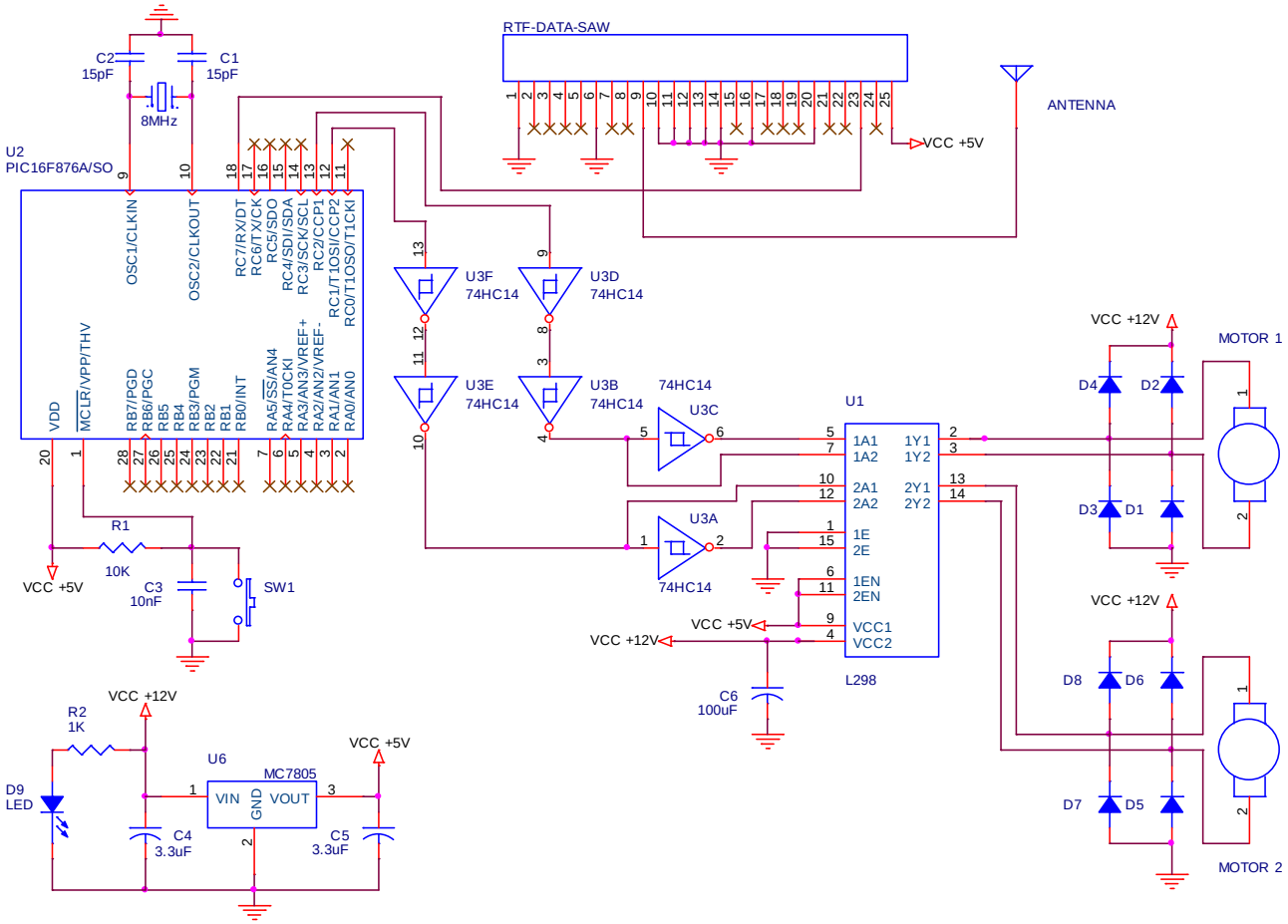
Ambienti di programmazione e simulazione

I µC possono essere programmati in assembler tramite una serie di istruzioni mnemoniche specifiche per ogni controllore. Questo è un linguaggio a basso livello che consente al programmatore di “avvicinarsi” molto alla macchina,consentendogli una maggiore accuratezza per quanto riguarda tempi, eventuali ritardi, una migliore gestione delle memorie ecc. È anche possibile programmare un µC con un linguaggio ad alto livello come ad esempio il C che facilitano la programmazione in quanto hanno un set di istruzioni più “vicine a noi”.
In particolare,il compilatore usato per questo progetto è il CCS Pic C compiler.

Ovviamente,non conviene provare il programma appena scritto direttamente sul circuito vero e proprio,ma, è più conveniente e pratico simularlo con appositi programmi. Per la simulazione ho usato ISIS Proteus che permette una volta disegnato il circuito,di simularlo consentendo l’utilizzo di strumenti come oscilloscopi,multimetri,generatori di funzioni ecc. In particolare Proteus offre la possibilita di caricare il programma già compilato sui microcontrollori.

La maggior parte degli schemi e dei grafici presenti in questa relazione sono stati realizzati con proteus (gli schemi elettrici sono stati realizzati con Capture).

Schema elettrico controllo del robot



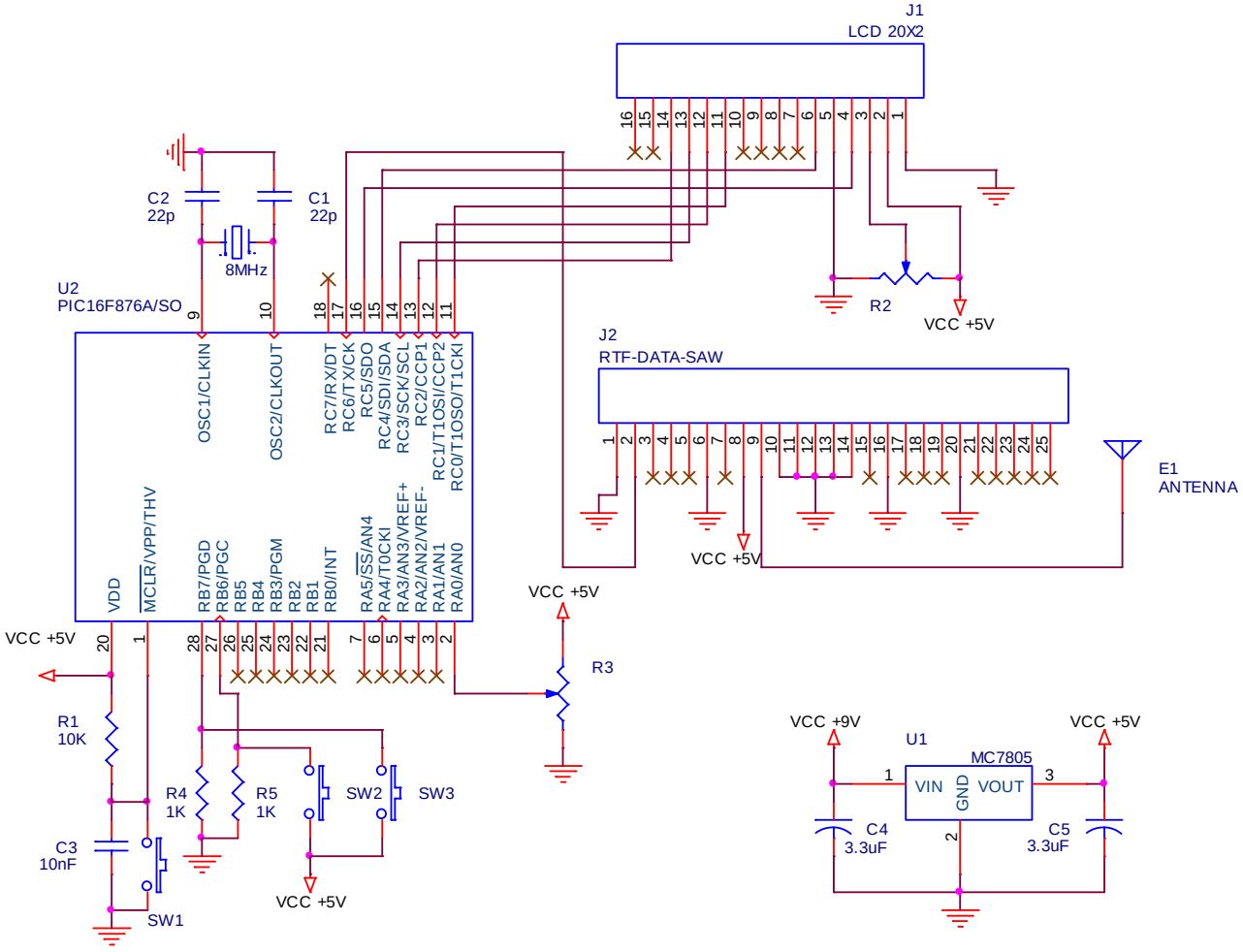
Elenco componenti:

C1 = C2 = 15pF
C3 = 10nF
C4 =C5 = 3.3µF
C6 = 100µF
R1 = 10KΩ
R2 = 1KΩ
D1÷D8 = 1N4007
D9 = diodo led
X1 = quarzo 8MHz

U1 =L298
U2 = PIC16F876A
U3 = 74HC14
RTF-DATA-SAW = modulo ricetrasmissione (on/off) AUREL
MOTOR 1-2 = Motoriduttori

Da notare il blocco di not U3F-U3E e U3D-U3B : a prima vista la doppia negazione è inutile,ma,ha la funzione di buffer,ossia di separare ulteriormente l'L298 dal µC,in quanto L298 può provocare problemi alla logica di controllo. Il ritardo generato dalla doppia negazione non è critico in quanto e dell'ordine dei nano secondi e dunque non sono di alcun disturbo alle basse frequenze con cui lavorano i moduli ccp del pic.Va anche dato uno sguardo alla configurazione delle porte not :esse infatti non sono state disposte casualmente come potrebbe sembrare, ma in questo modo i due ingressi sono da una parte dell'integrato e le uscite dall'altra. I diodi D1÷D8 servono per il ricircolo delle correnti,in quanto essendo il motore un carico induttivo, al variare della corrente che lo attraversa,genera f.e.m. auto indotte che in mancanza dei diodi verterebbero direttamente sull'L298.

Schema Trasmettitore



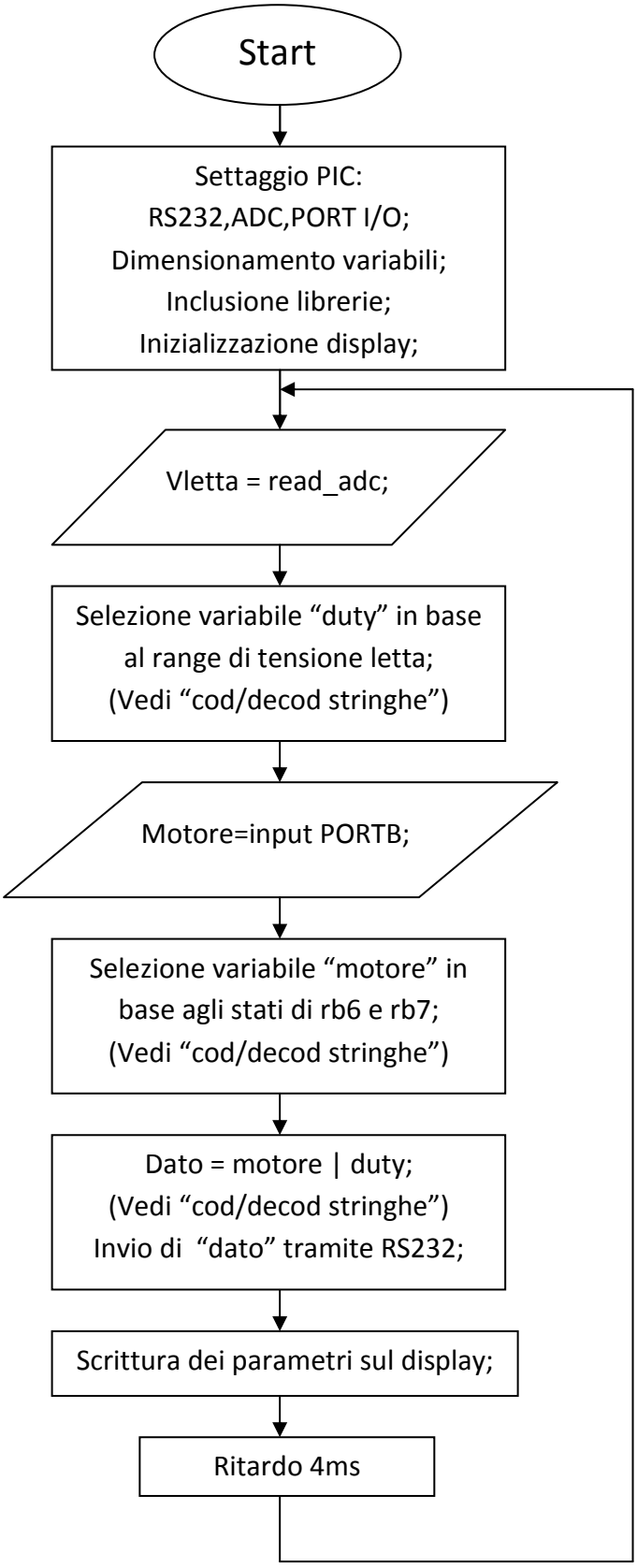
Elenco componenti:

C1 = C2 = 22pF
C3 = 10nF
C4 =C5 = 3.3µF
R1 = 10KΩ
R2 = potenziometro 4,7KΩ
R3 = potenziometro slider 4,7KΩ
R4 = 1KΩ
R5 = 1KΩ
X1 = quarzo 8MHz

U1 = 7805
U2 = PIC16F876A
RTF-DATA-SAW = modulo ricetrasmissione (on/off) AUREL
DISPLAY LCD 20X2

Software Pic

Diagramma di flusso del Trasmettitore:



Sorgente trasmettitore:

"TX.C":

```
#include "Tx.h"
#include "lcd20x2.c"  //Libreria per la gestione dell'LCD
void main()
{
setup_adc_ports(AN0); //settaggio RA0 come canale analogico
setup_adc(ADC_CLOCK_INTERNAL);
setup_spi(SPI_SS_DISABLED);
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
setup_vref(FALSE);
delay_ms(1);
set_adc_channel(0); //settaggio del canale analogico A0
delay_ms(1);
lcd_init(); //inizzializzazione display
lcd_putc("\f"); //pulitura dislay
lcd_gotoxy(1,1); //sposto il cursore alla prima colonna della prima riga
lcd_putc("Gentileschi Matteo");
lcd_gotoxy(1,0);
lcd_putc("I.T.I.S. C.Rosatelli");
delay_ms(2500);
lcd_putc("\f"); //pulitura dislay

while(1){
vletta=read_adc(); //Lettura tensione sul canale analogico
vd= (float)(vletta * 0.019);  //proporzione
//=============================================================================
//selezione del pwm in base al range di tensione letta:
if (vd <= 06){
duty=0;  //duty 0% stringa = 000
strcpy(duty_cycle,"Duty-cycle:0% ");
}
else if ((vd >0.6) && (vd <= 1.2)){
duty=1; //duty 10% stringa =001
strcpy(duty_cycle,"Duty-cycle:10% ");
}
else if ((vd >1.2) && (vd <= 1.8)) {
duty=2; //duty 20% stringa =010
strcpy(duty_cycle,"Duty-cycle:20% ");
}
else if ((vd >1.8) && (vd <= 2.4)) {
duty=3; //duty 30% stringa =011
strcpy(duty_cycle,"Duty-cycle:30% ");
}
else if ((vd >2.4) && (vd <= 3)) {
duty=4; //duty 70%  stringa =100
strcpy(duty_cycle,"Duty-cycle:70% ");
}
else if ((vd >3) && (vd <= 3.6)) {
duty=5; //duty 80%  stringa =101
strcpy(duty_cycle,"Duty-cycle:80% ");
}
else if ((vd > 3.6) && (vd <= 4.2)){
```

```
duty=6; //duty 90% stringa =110
strcpy(duty_cycle,"Duty-cycle:90% ");
}
else {
duty=7; //duty 100% stringa = 111
strcpy(duty_cycle,"Duty-cycle:100%");
}
//=============================================================================
/*
Selezione del motore a cui applicare il pwm :
00 = entrambe i motori fermi (duty al 50%)
01 = pwm al motore di destra
10 = pwm al motore di sinistra
11 = pwm a entrambe i motori
*/
motore =input_b(); //legge il dato sulla porta B
motore&= 192; //And con 192 = 11000000
switch(motore){
case 0 :   strcpy(motores,"Fermo         "); //nessuno
break;
case 192 : strcpy(motores,"Marcia        "); //entrambi
break;
case 128 : strcpy(motores,"Motore sinistro"); //sinistra
break;
case 64:   strcpy(motores,"Motore destro  "); //destra
break;
}
motore = motore >>3; //shift verso destra di 3 posti
//=============================================================================
dato = duty | motore; //concatenamento pwm e motore (OR)
putc(dato);//invio "dato" tramite RS232
lcd_gotoxy(1,1);
printf(lcd_putc,duty_cycle);//scrivo sul display la stringa duty_cycle
lcd_gotoxy(1,0);
printf(lcd_putc,motores);//scrivo sul display la stringa motores
delay_ms(4);
}
}
//Matteo Gentileschi
```

“TX.H”:

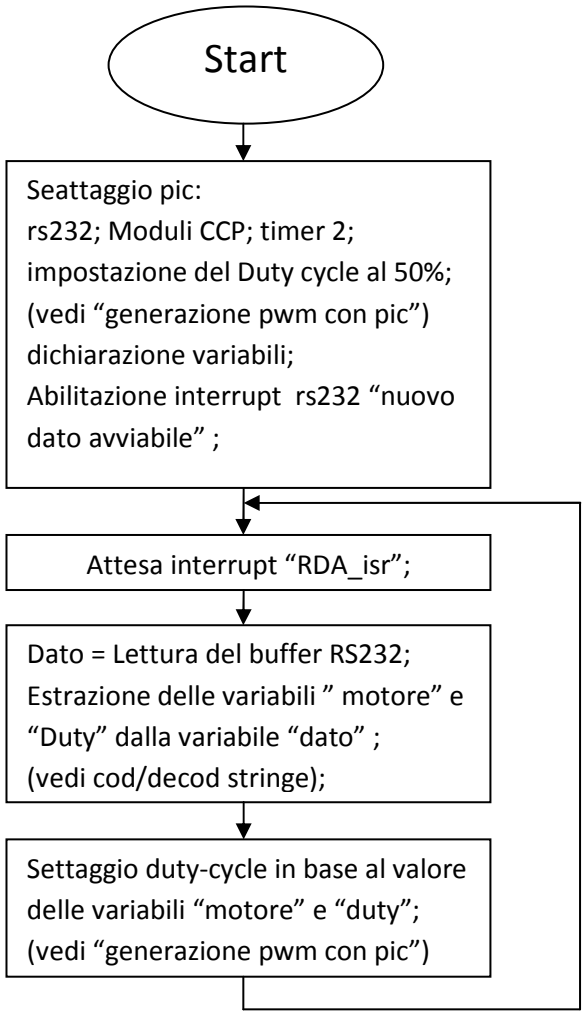
```
#include <16F876A.h>
#device adc=8

#FUSES NOWDT                 //No Watch Dog Timer
#FUSES HS                    //Crystal osc > 4mhz
#FUSES NOPUT                 //No Power Up Timer
#FUSES NOPROTECT             //Code not protected from reading
#FUSES NODEBUG               //No Debug mode for ICD
#FUSES NOBROWNOUT            //No brownout reset
#FUSES NOLVP                 //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD                 //No EE protection
```

```
#FUSES NOWRT

#use delay(clock=8000000)
#use rs232(baud=1200,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
int8 duty;
int8 motore;
int8 dato;
int8 vletta;
float vd;
char duty_cycle[20];
char motores[20];
```

diagramma di flusso del ricevitore:



Sorgente ricevitore:

"RX.C":

```
#include "Rx.h"
#int_RDA  //Sub dell'interrupt di nuovo dato presente sul buffer;
void  RDA_isr(void)
{
dato =getc(); //legge il contenuto del buffer
j = 1;
}
void main()
{
  setup_adc_ports(AN0);
  setup_adc(ADC_OFF);
  setup_spi(SPI_SS_DISABLED);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_1(T1_DISABLED);
  //Settaggio frequenza segnale PWM
  setup_timer_2(T2_DIV_BY_1,200,1);   //Tpwm =[pr2+1] * 4*Tosc * [Tmr2 prescaler];
  //Settagio modo di funzionamento dei moduli ;
  setup_ccp1(CCP_PWM);
  setup_ccp2(CCP_PWM);
  //Settaggio valore duty-cycle al 50% [pr2 value/2];
  set_pwm1_duty(100);
  set_pwm2_duty(100);
  enable_interrupts(INT_RDA);
  enable_interrupts(GLOBAL);
while(1){
if (j =1){
duty = dato;
duty&=7; //duty and '00000111'
motore = dato;
motore&= 24; //motore and '00011000'
motore = motore >>3;
switch (duty) {
case 0:
duty_1 = 0; //0%
break;
case 1:
duty_1 = 20; //10%
break;
case 2:
duty_1 = 40; //20%
break;
case 3:
duty_1 = 60; //30%
break;
case 4:
duty_1 = 140; //70%
break;
case 5:
duty_1 = 160; //80%
break;
case 6:
```

```
duty_1 = 180; //90%
break;
case 7:
duty_1 = 200; //100%
break;
}
switch (motore) {
case 0: //robot fermo
set_pwm1_duty(100); //50%
set_pwm2_duty(100); //50%
break;
case 1: //motore destro
set_pwm1_duty(100);
set_pwm2_duty(duty_1); //50%
break;
case 2: //motore destro
set_pwm1_duty(duty_1);//50%
set_pwm2_duty(100);
break;
case 3: //Entrambe i motori
set_pwm1_duty(duty_1);
set_pwm2_duty(duty_1);
break;
}
}
j = 0;
}
}
//Matteo Gentileschi
```

"RX.H":

```
 #include <16F876A.h>
#device adc=8
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                       //High speed Osc (> 4mhz)
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD                    //No EE protection
#FUSES NOWRT

#use delay(clock=8000000)
#use rs232(baud=1200,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
int1 j;
int8 dato;
int8 duty;
int8 duty_1;
int8 motore;
```

Ringraziamenti

“ Questo progetto segna la conclusione di un’esperienza durata cinque anni.
Vorrei ringraziare tutte le persone che mi sono state vicino e che mi hanno supportato (e sopportato…) nella realizzazione di questo prototipo.
Vorrei ringraziare tutti i miei compagni di classe, i ragazzi e le ragazze della gita a Barcellona e tutte le persone che ho conosciuto in questa scuola.
Vorrei ringraziare la mia scuola in generale, poichè è quì dentro che ho provato i primi affetti e le prime grandi delusioni ed è quì dentro che sono nate alcune delle mie amicizie migliori.
Un ringraziamento speciale va a tutte le persone che hanno creduto in me… ”

Grazie di cuore Matteo

Poggio Bustone 20/06/2010