PWM Tutorial *di Giovanni Giannetti*

Versione 1.0

Introduzione

Uno dei primi scogli che il principiante robottaro deve superare è quello di capire il funzionamento e le modalità di utilizzo dei motori che ha a disposizione. In rete si possono trovare informazioni in grande quantità riguardanti l'argomento ed è facile perdersi tra i tanti termini (spesso acronimi incomprensibili) che si possono incontrare quali PWM, motori DC, ponte H, ponti H integrati, locked anti-phase PWM, sign/magnitude PWM, emulazione del pwm via software, PWM via hardware con i moduli CCP dei PIC etc etc...

Questo breve tutorial guiderà(o almeno tenterà da farlo) l'aspirante robottaro nella comprensione del problema e nella sua soluzione passando dalla teoria, con una breve spiegazione sui motori DC e sul PWM, alla pratica, con l'implementazione di un controllo PWM hardware con un controllore PIC e la realizzazione di una scheda per il controllo di motori DC da collegare alla demoboard per i vostri esperimenti di robotica mobile.

1.Motori DC

Perchè cominciare il tutorial con i motori DC? Perchè i motori DC , collegati ad un'opportuna scheda di controllo, possono essere controllati con un segnale PWM (oggetto del nostro tutorial), inoltre i motori DC di piccola potenza sono economici, facili da reperire e spesso sono corredati di riduttore (gearbox) come nel caso di quelli prodotti dalla Tamiya (ad esempio il twin motor gearbox è costituito da due motorini con riduttori e assi di uscita indipendenti ma coassiali), dei servi RC modificati per la rotazione continua e privati dell'elettronica interna o dei motori di molti dei robot "panettoni" che ultimamente affollanno le edicole (ad esempio, acquistando le prime due uscite del Cybot si ottiene una base completa di motori e motoriduttori a soli 6 euro).

Per una spiegazione dettagliata e adatta ai principianti sui motori DC è consigliabile leggersi il tutorial "Motori DC di piccola potenza" scritto da Vincenzo Villa (disponibile presso il sito www.vincenzov.net , ricco di guide e progetti molto utili), a noi basterà sapere che un motore DC è dotato di due morsetti e che la velocità e il verso di rotazione dell'albero motore dipendono dalla corrente che facciamo scorrere tra loro (cambiando il verso della corrente si cambia il verso di rotazione dell'albero, aumentando o diminuendo il modulo della corrente possiamo variare la coppia fornita dal motore).

1.1 Controllo ON-OFF di motori DC

I motori DC possono essere controllati in vari modi, il pilotaggio più semplice è quello ON-OFF che permette di comandare il motore solo alla massima velocità di rotazione in un verso (interruttore ON) oppure fermarlo (interruttore OFF): questo controllo può essere implementato con un interruttore (es. un mos o un transistor) e con un diodo di ricircolo necessario per evitare danni al resto del circuito (il motore è un carico con una componente induttiva). Lo schema è il seguente:

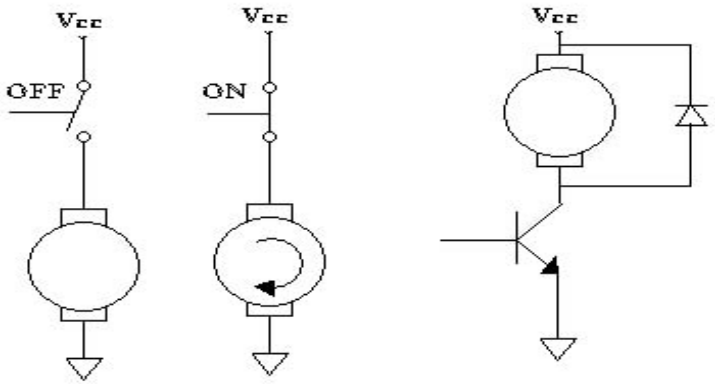


figura2: pilotaggio ON-OFF: quando l'interruttore è aperto il motore è fermo, quando è chiuso gira alla massima velocità. A destra un esempio di implementazione

1.2 Il ponte ad H

Il tipo di controllo appena presentato non permette né la regolazione della velocità del motore né la possibilità di far girare il motore in entrambi i versi di rotazione.

Per far girare il motore nel verso opposto è necessario infatti invertire il segno della corrente che passa all'interno del motore stesso, per far ciò si usa un circuito chiamato ponte H costituito da 4 interruttori comandati e da 4 diodi di ricircolo:

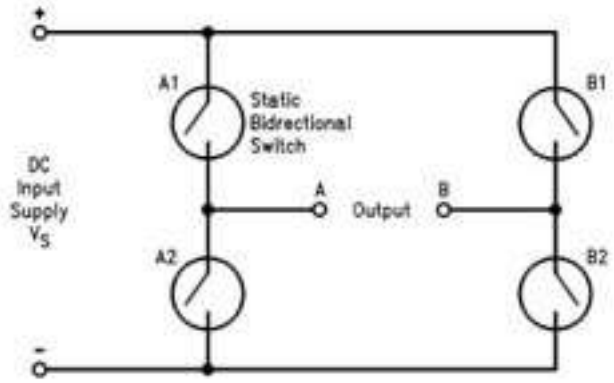


Figura 1: schema semplificato di un ponte H

Attivando i transistor A1 e B2 la corrente scorre nel motore in un verso mentre attivando i transistor B1 e A2 la corrente scorre nel verso opposto. E' da evitare la configurazione in cui sono accesi entrambi i transistor A o entrambi i transistor B infatti la corrente di corto circuito su un lato del ponte potrebbe creare seri danni al

ponte stesso o al circuito di alimentazione: questa eventualità non si presenterà con l'utilizzo della scheda di controllo proposta in seguito.

1.3 Ponti H discreti e Ponti H integrati

*Esistono due tipi di ponti H: i **ponti H discreti**, costituiti da componenti sparsi come transistor e diodi e i **ponti H integrati**, in questo caso tutto il circuito è racchiuso in un package plastico di tipo DIP (dual in-line package) o simile. I ponti H integrati sono molto versatili e, oltre a garantire una bassa occupazione di area nel circuito (in alcuni casi, come il SN745510 e l' L293D, contengono anche i diodi di ricircolo), hanno buone prestazioni(l' L298 può fornire fino a 2A per ponte(di solito ogni integrato ha 4 mezzi ponte H)), possono essere essere montati in parallelo per ottenere alte correnti e riescono a lavorare in un intervallo di tensioni di alimentazione molto ampio (da 6V a 48V circa a seconda del modello).*

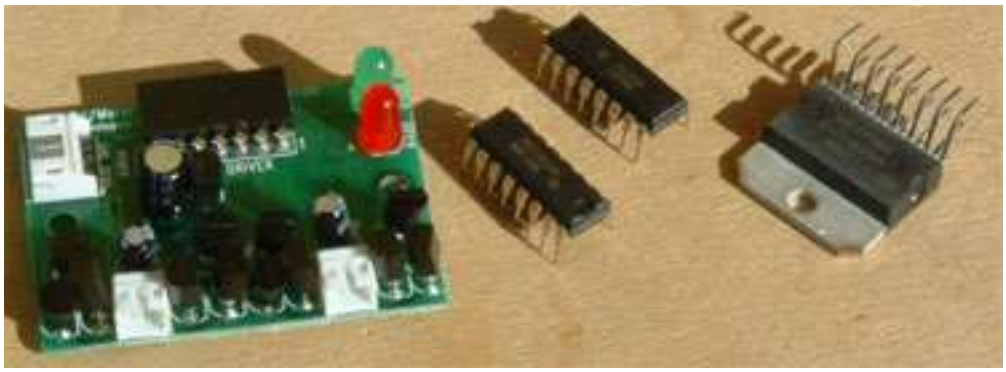
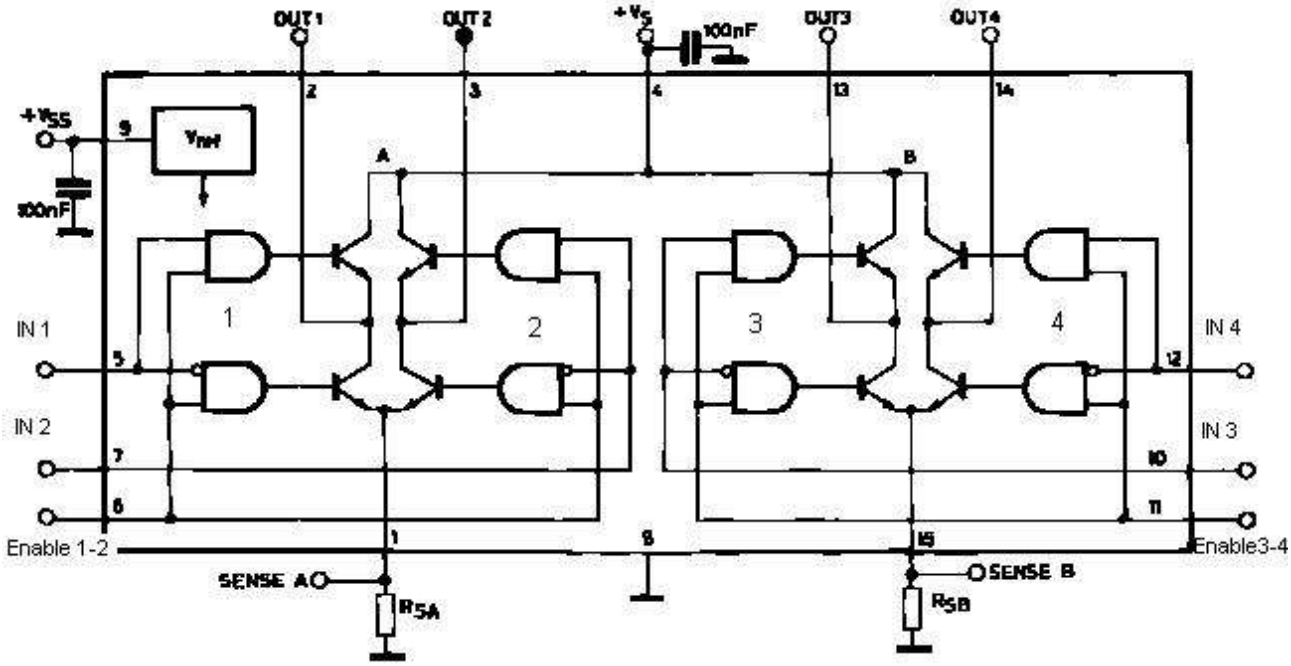


Figura 2: Nell'ordine(da sx a dx): un doppio ponte H discreto(scheda controllo motori Cybot), un SN754410, un L293NE, un L298

La seguente figura rappresenta lo schema di funzionamento di un ponte integrato:



Il circuito integrato in questione è costituito da quattro mezzi ponti H (numerati in figura 1-2-3-4) ognuno dei quali è costituito da due transistor e da una logica che li comanda in modo da accenderne solo uno alla volta: quando il transistor superiore di un mezzo ponte è in conduzione quello inferiore sarà necessariamente spento e viceversa. E' inoltre presente un comando di ENABLE che permette di inibire il funzionamento di una coppia di mezzi ponti.

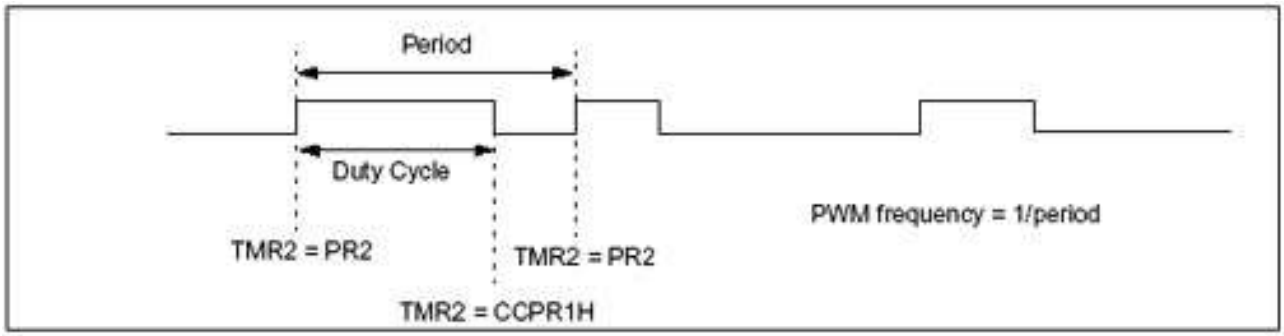
Si può inoltre notare la presenza di due pin dedicati alla connessione di una resistenza di Sense per monitorare la corrente che scorre nel motore (utile per un controllo in retroazione) ma tale caratteristica non è presente in tutti i ponti H integrati tra cui il SN754410 che verrà utilizzato nella scheda controllo motori che verrà presentata in seguito.

Ricapitolando, per ognuno dei due ponti presenti nell'integrato abbiamo a disposizione due ingressi di controllo per permettere il passaggio di corrente in un verso e un ingresso di ENABLE per accendere e spegnere il ponte: come vedremo a breve ci sono più modi di pilotare il ponte H e per comprenderli è necessario aver capito bene a cosa servono questi tre ingressi.

Con l'introduzione del Ponte H abbiamo risolto il problema della rotazione del motore in entrambi i versi, ora è necessario risolvere l'altro problema, quello di poter regolare la velocità di rotazione, per far ciò è necessario introdurre il concetto di PWM.

2.PWM

Un segnale PWM (Pulse Width Modulation ovvero modulazione a variazione della larghezza d'impulso) è un' onda quadra di duty cycle variabile che permette di controllare l'assorbimento (la potenza assorbita) di un carico elettrico(nel nostro caso il motore DC), variando (modulando) il duty cycle.

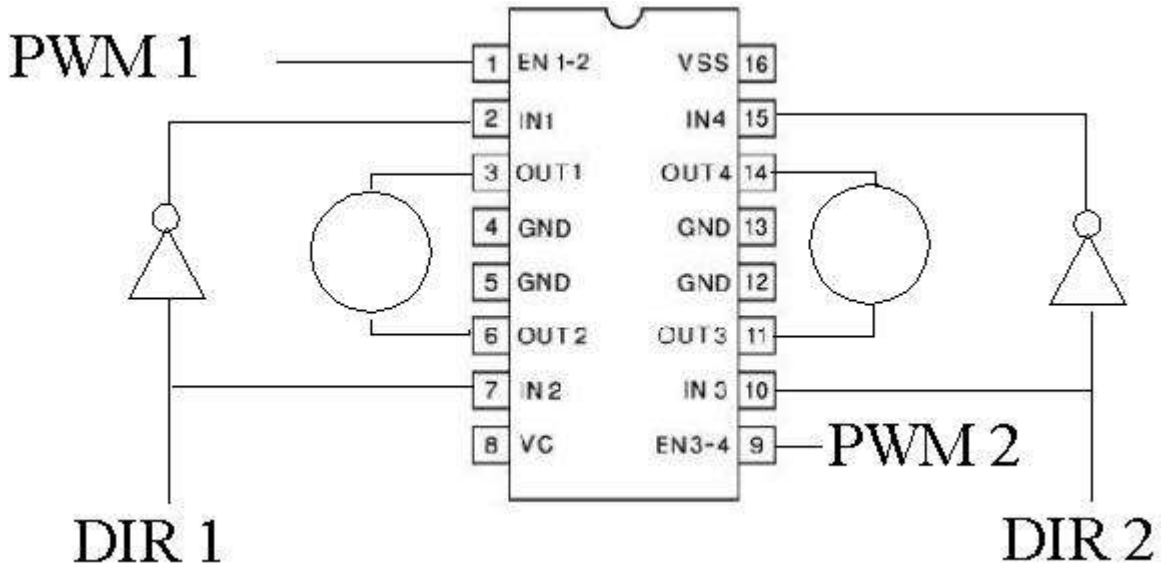


Un segnale PWM è caratterizzato dalla frequenza (fissa) e dal duty cycle (variabile); come si deduce dalla Figura3, il duty cycle è il rapporto tra il tempo in cui l'onda assume valore alto e il periodo T(l'inverso della frequenza: T=1/f) ne segue che un duty cycle del 50% corrisponde ad un'onda quadra che assume valore alto per il 50% del tempo, un duty cycle dell'80% corrisponde ad un'onda quadra che assume valore alto per l'80% del tempo e basso per il restante 20%, un duty cycle del 100% corrisponde ad un segnale sempre alto e un duty cycle dello 0% ad un segnale sempre basso (come vedremo anche questi ultimi due casi non sono del tutto inutili).

Ora è necessario capire come applicare il segnale PWM al ponte H per controllare il motore, esamineremo due modalità: il PWM sign-magnitude e il PWM locked anti-phase.

2.1 Sign-Magnitude PWM

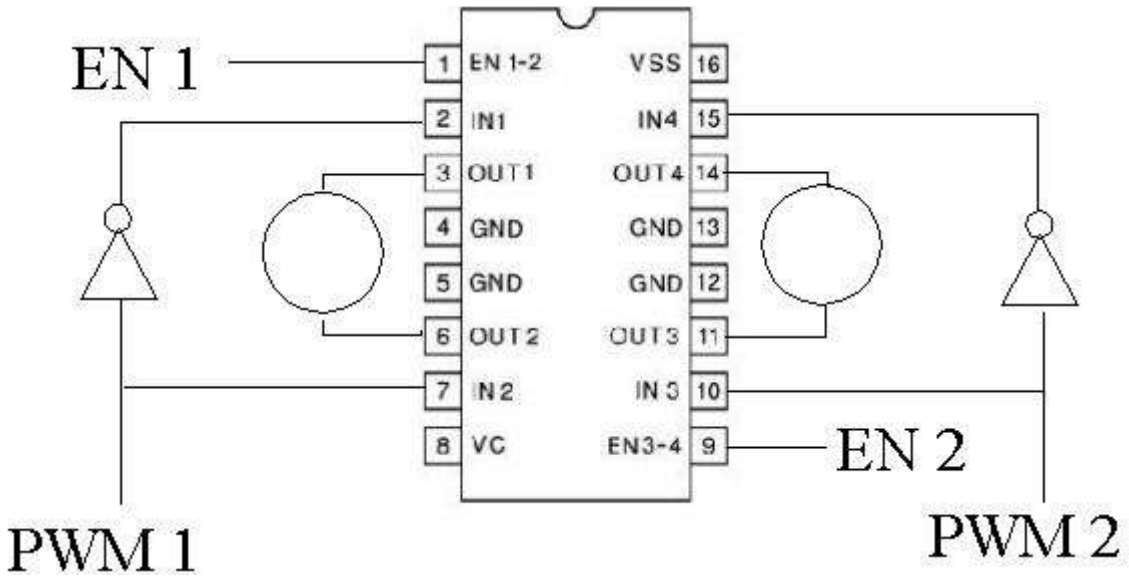
Come è possibile vedere dallo schema che segue, il pilotaggio SM (sign-magnitude) consiste nell'inviare il segnale PWM all'ingresso di enable del ponte e di comandare la direzione di rotazione del motore tramite i due ingressi di controllo del ponte. Tali due ingressi devono essere comandati da segnali invertiti, in questo modo si riduce anche il numero di pin necessari per il controllo.



Per il controllo SM sono necessari quindi due segnali: il primo è un'onda quadra di duty cycle variabile tra 0 e 100% che stabilisce la velocità di rotazione, il secondo è un segnale costante che determina il verso di rotazione(segnale basso rotazione in un verso, segnale alto rotazione nell'altro verso).

2.2Locked Anti-phase PWM

Il controllo LAP (locked anti-phase) si basa sulla stessa configurazione circuitale del controllo SM tuttavia i segnali di comandi sono applicati in modo diverso, come è possibile vedere nella seguente figura:



In questo caso il segnale PWM viene messo in ingresso all'invertitore in modo da avere ai due lati opposti del ponte due segnali invertiti tra loro; agendo sull'enable è possibile spegnere il rispettivo ponte.

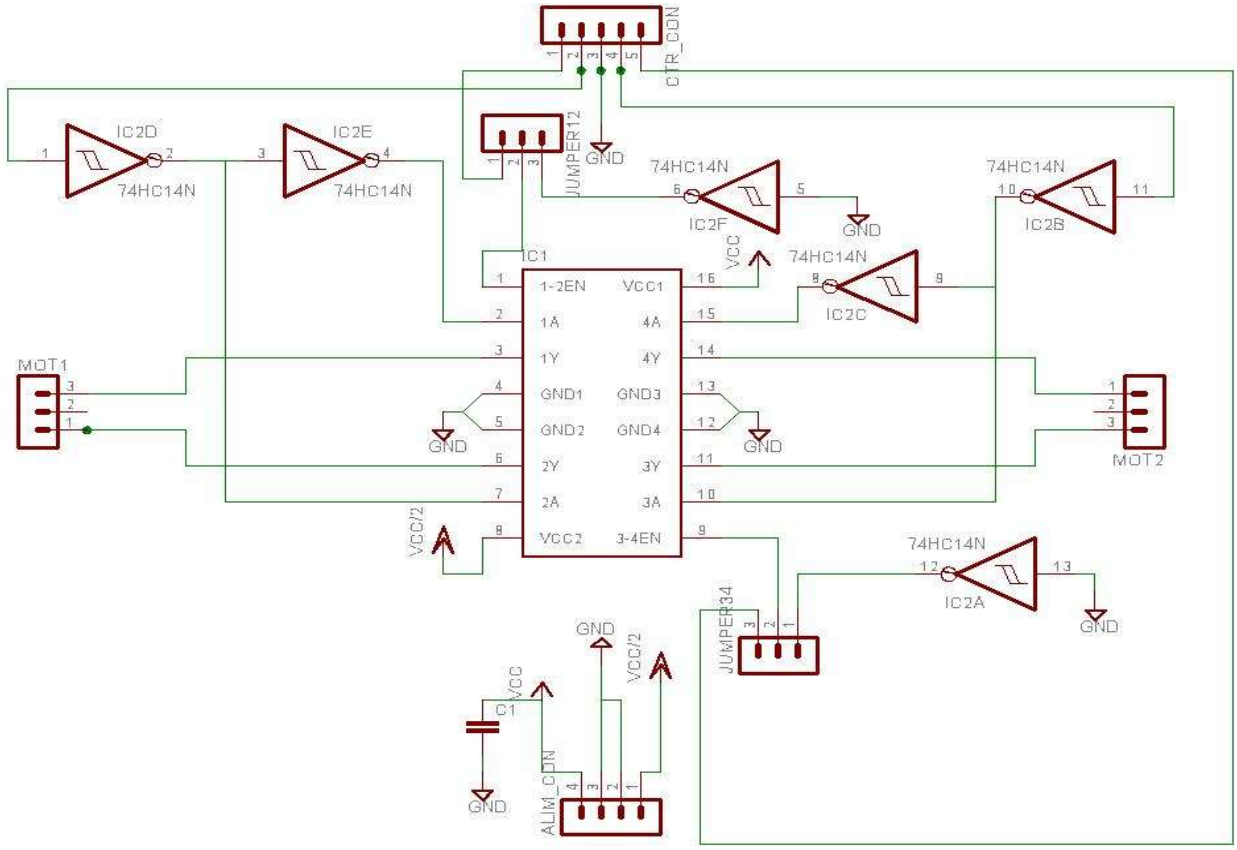
Per il controllo LAP può bastare anche solo un segnale di comando(l'enable può essere fissato alto se non necessario) infatti l'onda quadra stabilisce sia la velocità che il verso di rotazione nel seguente modo:

- *Duty cycle a 0 : rotazione alla massima velocità in un verso*
- *Duty cycle al 50%: motore fermo*
- *Duty cycle al 100%: rotazione alla massima velocità nell'altro verso*

*3.*Scheda di controllo motori DC

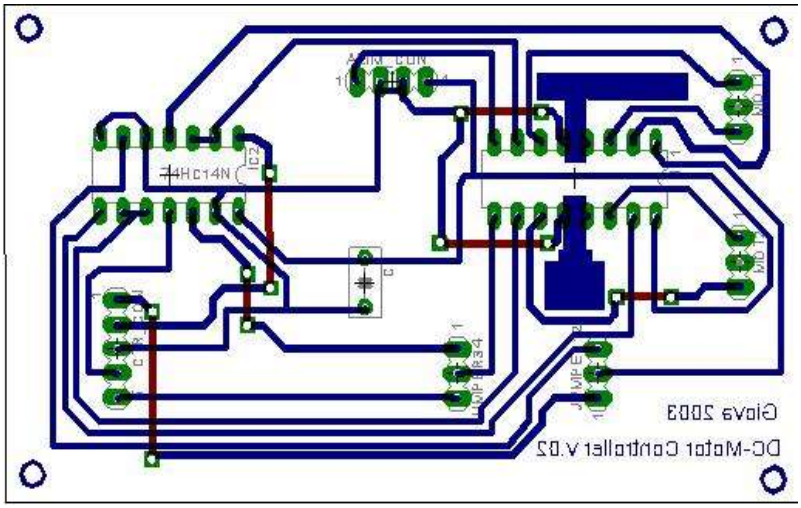
E' giunto il momento di passare alla pratica (tah-dah!), armatevi di fogli lucidi o pnp blue, spolverate il vostro bromografo o il ferro da stiro e preparatevi all'incisione di questo circuito stampato (ricordatevi le necessarie precauzioni quando maneggiate soda caustica, percloruro ferrico e acqua ossigenata!). Quella che segue è una semplice scheda di controllo motori DC in PWM che può essere usata sia in modalità LAP-PWM che in modalità SM-PWM(la descrizione che segue tiene conto del LAP, per SM basta applicare quanto detto prima) basata sull'integrato SN754410 visto in precedenza e su un sestuplo inverter 74HC14.

Ecco lo schema elettrico:



L'interfaccia della scheda comprende quattro connettori e due jumper: dei quattro connettori due(MOT1 e MOT2) servono per la connessione dei motori DC, uno (ALIM_CON)per l'alimentazione della logica e dei motori(il connettore è a 4 pin, il primo per l'alimentazione della logica, il secondo e il terzo per la massa e il quarto per l'alimentazione dei motori) e uno(CTR_CON) per il controllo vero e proprio. I cinque pin del connettore di controllo sono così usati: il primo e il secondo servono rispettivamente per l'ENABLE e per il controllo PWM del primo motore, il terzo per la massa, il quarto e il quinto rspettivamente per il controllo e l'ENABLE del secondo motore. Nel caso in cui non sia necessario gestire l'ENABLE di un motore o di entrambi tramite il uC basterà cambiare la posizione del jumper relativo al motore, così facendo l'ENABLE sarà tenuto alto da uno degli inverter del 74HC14 e potremo controllare i nostri motori con soli due fili limitando al minimo l'impiego di porte del microcontrollore.

Ecco il risultato dello sbroglio del circuito:



Le due zone blu sotto l' SN754410 sono previste dal datasheet e servono per creare un'area di rame sulla pcb adibita a dissipare il calore prodotto dall'integrato. In rosso sono riportati i ponticelli.

Per avere il file BMP 300 dpi della basetta mandatemi una email con oggetto "PWM tutorial"

4.PWM e PIC

Vediamo ora come generare dei segnali PWM con i microcontrollori PIC. Esistono due modi per generare tali segnali: il primo (emulazione PWM software) consiste nello scrivere del codice che generi un'onda quadra di frequenza e duty cycle assegnata su un certo piedino di uscita, il secondo (PWM hardware) consiste nell'utilizzare delle periferiche apposite dei PIC , i moduli CCP (capture compare pwm), per generare l'onda. Risulta ovvio che il secondo metodo è il migliore(con il primo la generazione dell'onda quadra può essere difficile e comunque toglie preziosi cicli di clock al resto dell'applicazione) tuttavia non tutti i microcontrollori PIC ne sono dotati: nel nostro caso utilizzeremo dei uC PIC16F877 e PIC16F876 che mettono a disposizione ben due moduli CCP per il PWM: tali moduli si appoggiano al timer 2 e sono quindi vincolati ad avere la stessa frequenza mentre il duty cycle può essere variato indipendentemente per i moduli andando a scrivere in appositi registri.

Per poter utilizzare i moduli CCP è necessaria una fase di setup che consiste nell'inserimento di opportuni valori nei registri che regolano il timer2 (T2CON e PR2), i due moduli CCP (CCP1CON , CCP2CON,CCPRxL) e la porta C (TRISC).

Cominciamo dai registri del Timer2 dai quali dipende la frequenza di PWM:

***T2CON** è un registro a 8 bit così suddiviso:*

7	6	5	4	3	2	1	0
-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

- *Il bit 7 non viene utilizzato*
- *I bit 6-3 servono per il valore del postscaler (nel nostro caso vanno messi a 0)*
- *Il bit 2 serve per attivare(valore 1)-disattivare(valore 0) il Timer2*
- *I bit 1-0 servono per impostare il valore del prescaler (come vedremo è uno dei parametri che permettono di impostare la frequenza di PWM)*

un esempio di contenuto del registro può essere 0b00000100 (timer2 ON e valori di postscaler e prescaler unitari).

*Il valore del periodo di PWM viene stabilito scrivendo opportunamente sul registro a 8 bit **PR2.***

La seguente equazione permette di ricavare il Periodo di PWM:

periodo PWM =[(PR2)+1]*4*Tosc*(TMR2 prescale value)

dove:

- *PR2 è il valore inserito nel registro PR2*
- *Tosc è l'inverso della frequenza del quarzo esterno che fornisce il clock al PIC*
- *TMR2 prescale value è il valore dei due bit meno significativi del registro T2CON*

Ecco alcuni esempi di frequenze di PWM che si possono ottenere con diverse impostazioni dei registri e quarzo esterno a 20MHz:

TABLE 8-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 20 MHz

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12kHz	156.3 kHz	208.3 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0xFFh	0xFFh	0xFFh	0x3Fh	0x1Fh	0x17h
Maximum Resolution (bits)	10	10	10	8	7	5.5

*Passiamo ora alla configurazione dei due moduli CCP1 e CCP2, i registri **CCPxCON** (sostituire la x con 1 e 2) svolgono una duplice funzione: contengono la configurazione del modulo e parte del valore binario con cui è espresso il duty cycle. Sono così suddivisi:*

7	6	5	4	3	2	1	0
-	-	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0

- *I bit 7-6 non vengono utilizzati*
- *I bit 5-4 contengono i bit meno significativi del valore del duty cycle*
- *I bit 3-0 contengono la configurazione del modulo CCP corrispondente, per settare il modulo per il funzionamento in PWM i bit 3 e2 devono essere posti a 1 mentre il valore dei bit 1 e 0 è ininfluente(noi lo metteremo a 0).*

*Il registro **TRISC** deve essere configurato in modo che i pin 1 e 2 (quelli su cui avremo l'uscita PWM) siano pin di uscita, quindi TRISC1 e TRISC2 dovranno essere posti a 0.*

*I registri **CCPR1L** e **CCPR2L** contengono invece gli otto bit più significativi del valore del duty cycle. Il duty cycle può quindi essere espresso con una risoluzione di 10 bit (quindi valori da 0 a 1023!).*

Ricapitolando ecco i settaggi da eseguire per inizializzare i moduli PWM:

1. *Scrivere nel registro PR2 il periodo del PWM*
2. *Scrivere il duty cycle nei registri CCPRxL e nei bit 5:4 del registro CCPxCON*
3. *impostare i pin 1 e 2 della porta C come uscite azzerando gli opportuni bit del registro TRISC*
4. *inizializzare il timer2 scrivendo sul registro T2CON*
5. *inizializzare i moduli CCP per le operazioni PWM scrivendo sui registri CCPxCON*

Tali settaggi sono necessari solo all'inizio del programma che verrà inserito nel microcontrollore, successivamente, quando sarà necessario cambiare il duty cycle dell'onda quadra, sarà sufficiente modificare i valori presenti nei registri CCPRxL e nei bit 5:4 dei registri CCPxCON (questi ultimi due valori spesso non vengono modificati per comodità: così facendo si perdono due bit di risoluzione ma per usi standard può andar bene).

*Di seguito sono riportate quattro funzioni in linguaggio C che possono essere molto utili, la prima, **PWMSetup()**, si occupa di tutta la fase di inizializzazione, basterà quindi richiamarla una sola volta all'inizio del nostro main.*

```
void PWMSetup(void) {
 CCP1CON=0; //resetto il modulo CCP1
 CCP2CON=0; //resetto il modulo CCP2
 PR2=0xFF; //valore massimo
 PWMDuty1(512); //DC 50%
 PWMDuty2(512); //DC 50%
 TRISC2=0; //setto come out il pin
```

```
 TRISC1=0;
 T2CON=0b00000100; //senza prescaler e post scaler  e attivo Timer2
 CCP1M3=1; CCP1M2=1; CCP1M1=0; CCP1M0=0; //modalit... PWM
 CCP2M3=1; CCP2M2=1; CCP2M1=0; CCP2M0=0; //modalit... PWM
}
```

*Le altre tre funzioni permettono di variare il duty cycle del primo modulo CCP (**PWMDuty1**),del secondo modulo CCP (**PWMDuty2**) e di entrambi(**PWMDuty**).*

```
void PWMDuty1(unsigned int duty1) {
 CCP1Y= 0b00000001&duty1;
 CCP1X= 0b00000001&(duty1>>1);
 CCPR1L=0b11111111&(duty1>>2);
 }
void PWMDuty2(unsigned int duty2) {
 CCP2Y= 0b00000001&duty2;
 CCP2X= 0b00000001&(duty2>>1);
 CCPR2L=0b11111111&(duty2>>2);
 }
void PWMDuty(unsigned int duty1,unsigned int duty2){
 PWMDuty1(duty1);
 PWMDuty2(duty2);
}
```

Fonti

- *Faq di roboteck(un buon inizio)*
- *Tutorial "Motori DC di piccola potenza" di Vincenzo Villa (www.vincenzov.net sezione tutorial)*
- *PIC16F87XA datasheet disponibile sul sito www.microchip.com (sempre leggere i datasheet!)*
- *Application Note AN594 disponibile sul sito www.microchip.com*
- *Application Note AN694 della National Semiconductors*
- *Application Note SGS-Thomson: "Applications of monolithic bridge drivers"*
- *Datasheet degli integrati L293, L298, SN745510(disponibili sui siti www.ti.com e www.st.com)*

Contatti

Sono ben accetti commenti-critiche sul tutorial, richieste di spiegazione di parti del testo o di termini non chiari, suggerimenti per migliorare-espandere il tutorial, etc.

Potete contattarmi all'indirizzo giovagiann@tiscali.it, (con oggetto "PWM TUTORIAL")

Versioni del documento:

v1.0 : inserite le immagini su LAP e SM e prima pubblicazione del manuale