P.W.M.
duty cycle
il segnale che gestisce gli attuatori

In questa lezione andremo a conoscere meglio come funzionano i comandi dei sistemi elettronici : il P.W.M. – Duty Cycle e come si leggono i segnali di questi sistemi.
Nei sistemi elettronici, ormai da tempo viene utilizzato il controllo mediante PWM, ma cosa è un comando PWM?, vediamo innanzitutto le definizioni tecniche.
Un segnale PWM, dall'inglese 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 che nel nostro caso potrebbe essere un attuatore o un motore elettrico, variando,("modulando" appunto, il "duty cycle".
"Modulare un segnale" vuol dire cambiare una sua grandezza in funzione di un'altra.
Le grandezze principali di un segnale analogico sinusoidale che possono essere modulate sono: ampiezza, frequenza e fase. Pertanto si può parlare di modulazioni AM (Amplitude Modulation), FM (Frequency Modulation) e PM (Phase Modulation).
Nel nostro caso abbiamo a che fare con "onde quadre": le grandezze che possono essere modulate sono: frequenza, durata, posizione.
Da ciò scaturisce la modulazione PPM (Pulse Position Modulation), PWM (Pulse Width Modulation), PFM (Pulse Frequency Modulation). In questa lezione però andremo ad approfondiremo la PWM,esso è caratterizzato dalla frequenza fissa e dal "duty cycle" variabile.
Il "duty cycle" è il rapporto tra il tempo in cui l'onda quadra assume valore "alto" e il periodo T, dove "T" è l'inverso della frequenza: T=1/f.
Ne segue che (vedi figure A e B):

- un duty cycle del 50% corrisponde a un'onda quadra che assume valore alto per il 50% del tempo e assume nel restante 50% un valore basso;
- un duty cycle del 20% corrisponde a un'onda quadra che assume valore alto per il 20% del tempo e basso per il restante 80%;
- un duty cycle del 100% corrisponde a un segnale sempre alto;
- un duty cycle dello 0% corrisponde a un segnale sempre basso.

Per maggiore chiarezza, se consideriamo gli ultimi due casi citati, un duty cycle pari a 0% indica un impulso di durata nulla, in pratica assenza di segnale e quindi di alimentazione, mentre un valore prossimo al 100% indica, in estrema sintesi, il massimo trasferimento del segnale e, di conseguenza, la piena e costante alimentazione del dispositivo comandato.
Come si può intuire, con un duty cycle pari a zero la potenza trasferita è nulla, mentre al 100% la potenza corrisponde al valore massimo trasferito. Ogni valore intermedio determina quindi una corrispondente "frazione percentuale" di comando.
Ricapitolando, un segnale PWM presenta due stati:

- ON = stato attivo = segnale alto = presenza di tensione
- OFF = stato passivo = segnale basso = assenza di tensione.

I due stati si alternano, come abbiamo visto, entro un periodo di tempo prefissato e costante,frequenza fissa,(vedi grafico figura C).

Ricordiamo che per una corretta diagnosi dei comandi in P.W.M. è indispensabile l'uso dell'oscilloscopio.

Estremamente utile ai fini della diagnosi è l'utilizzo anche di un Generatore di Segnali che consente di "simulare", variando anche frequenza e duty cycle, il comando di uscita in PWM in modo da poter diagnosticare rapidamente e in maniera sicura malfunzionamenti.

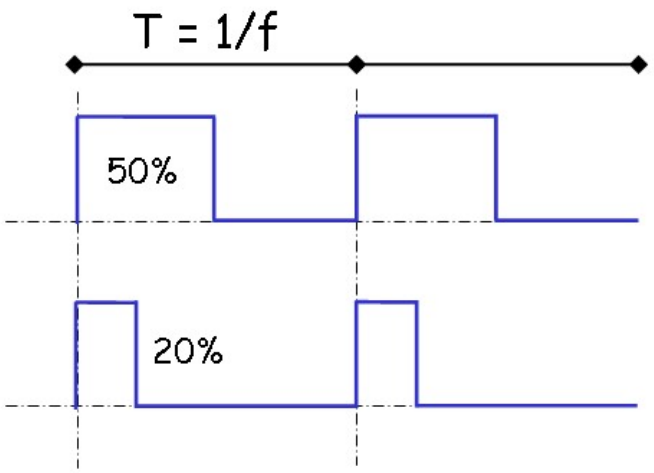


Figura A

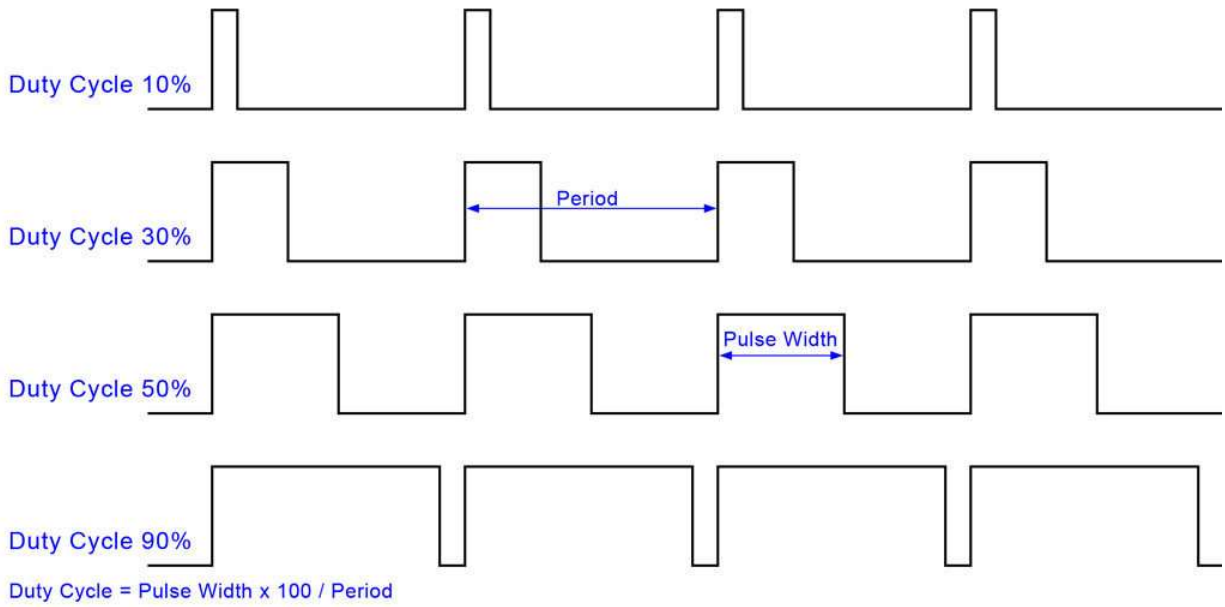


Figura B

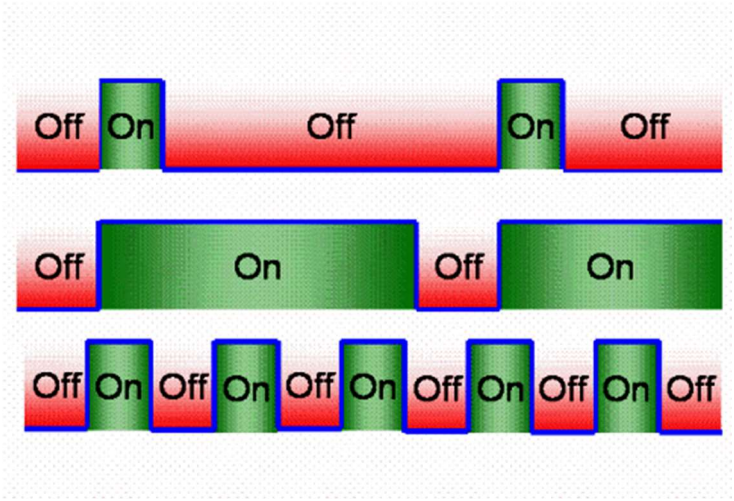
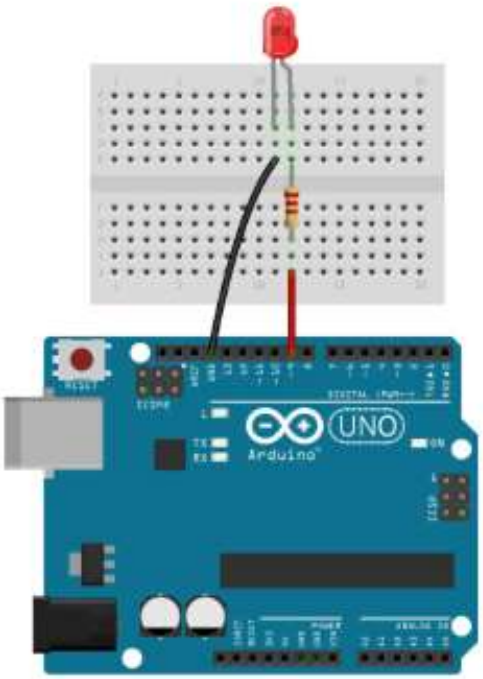


Figura C

REALIZZARE DI UN PWM CON ARDUINO

Per realizzare un PWM con Arduino andremo a modulare un uscita per far lampeggiare un led o una lampadina, cioè a cambia dal valore basso, 0V, a quello alto, 5V così velocemente che l'occhio umano non è in grado di vedere questo lampeggio velocissimo, ma si noterà una differenza di luminosità. Più tempo l'uscita sarà a 5V e più alta sarà la luminosità della lampadina, viceversa se l'uscita sarà alta solo per poche frazioni di secondo, la luminosità sarà bassissima.



Codice

```
1.      int i = 0;
2.
3.      void setup()  {
4.        pinMode(9, OUTPUT);
5.      }
6.
7.      void loop()  {
8.        for (i = 0; i < 255; i++ ) {
9.          analogWrite(9, i);
10.         delay(10);
11.       }
12.       for (i = 255; i > 0; i-- ) {
13.         analogWrite(9, i);
14.         delay(10);
15.       }
16.     }
```

Alla riga 8 è stato usato un semplice ciclo "for" che permette di incrementare la variabile "i" dal valore "0" al valore "255" incrementando di 1 ad ogni ciclo.

Alla riga 9 c'è il comando "analogWrite", che è la funzione per scrivere un segnale PWM in uscita. Tale funzione scrive un valore analogico (PWM) su un pin.può essere utilizzata non solo per far illuminare un led variandone l'intensità luminosa, ma anche per comandare un motore e variarne la velocità. Dopo la chiamata del metodo analogWrite(), sul pin verrà generata un'onda quadra regolare con un duty-cycle specifico fino alla prossima chiamata di analogWrite() o di digitalWrite(), o di digitalRead() sullo stesso pin. La frequenza del segnale PWM è all'incirca 490Hz (sulla maggior parte delle schede di Arduino questa funzione lavora sui pin 3, 5, 6, 9, 10 e 11; sulle altre schede potrebbero esserci delle varianti). I pin analogici non necessitano di essere impostati prima della chiamata di analogWrite() con pinMode nel setup.
Si ricorda che per sintassi "analogWrite"(pin,valore),il parametro pin rappresenta il pin su cui scrivere; il parametro valore rappresenta il valore del duty-cycle tra 0 (sempre OFF) e 255 (sempre ON).
Grazie al ciclo "for" verrà scritto il valore di "i" incrementato di 1 e quindi ad ogni ciclo del loop il led si accenderà un po' di più, fino a raggiungere la massima luminosità.
Alla riga 12 viene ripetuto il ciclo "for" ma al contrario, quindi il led ridurrà piano piano la sua luminosità.