PWM

Con i microcontrollori

Introduzione

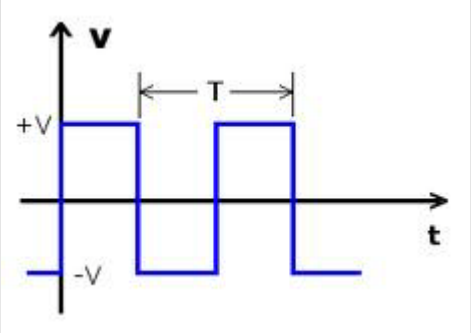
- **PWM= Pulse Width Modulation**. Modulazione digitale che permette di ottenere una tensione media variabile dipendente dal rapporto tra la durata dell'impulso positivo e quello negativo.
- Nella trasmissione tramite modulazione PWM si usa come supporto un segnale a onda quadra, di frequenza e ampiezza fissate, nel quale la larghezza, ovvero la durata degli impulsi, è proporzionale al livello del segnale modulante.
- Si utilizza nei protocolli di comunicazione in cui l'informazione è codificata sottoforma di durata nel tempo di ciascun impulso
- In elettrotecnica viene utilizzata per variare la potenza elettrica inviata ad un carico

PWM in elettronica

- Nei controlli automatici si ha l'esigenza di controllare grandezze analogiche tramite microcontrollori. I micro sono dotati di un convertitore analogico/digitale
- Se si ha l'esigenza di ottenere grandezze analogiche dai microcontrollori, si utilizza la PWM

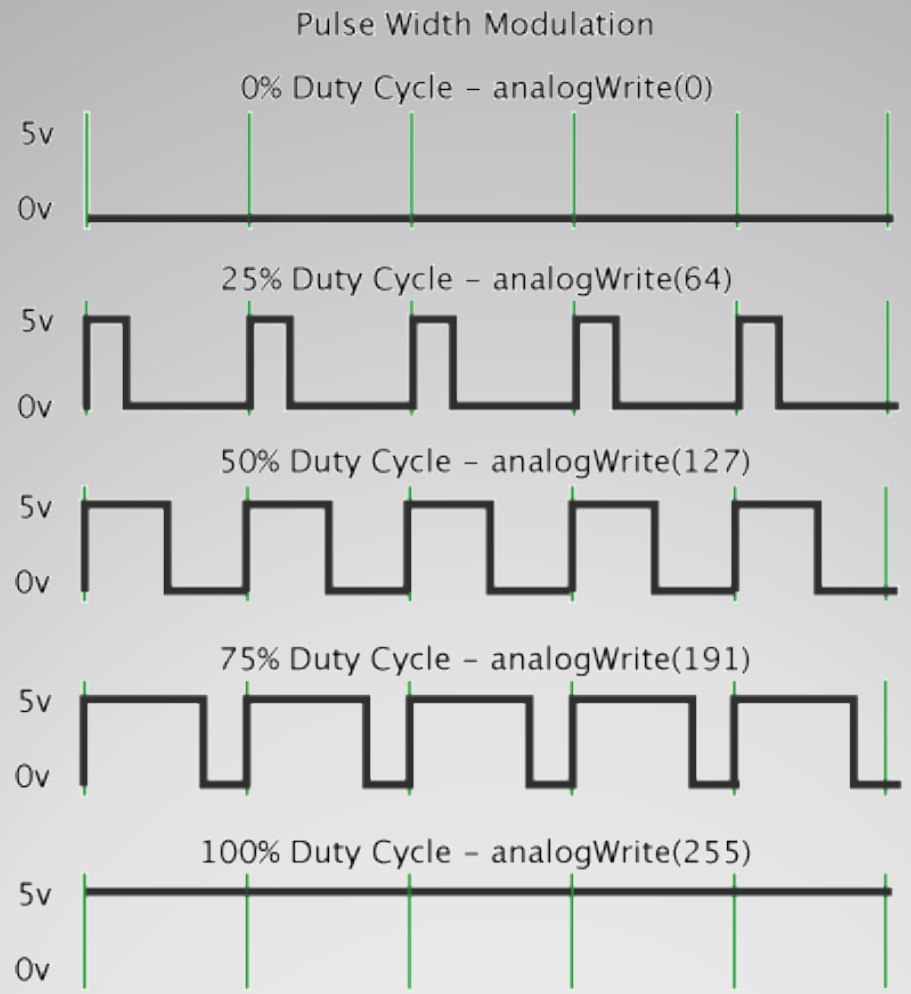
PWM e onda quadra

- Un segnale PWM è un'onda quadra con la larghezza variabile, duty cycle variabile. Questa variazione permette di controllare l'assorbimento di potenza di un carico elettrico.
- Un'onda quadra si presenta nel seguente modo

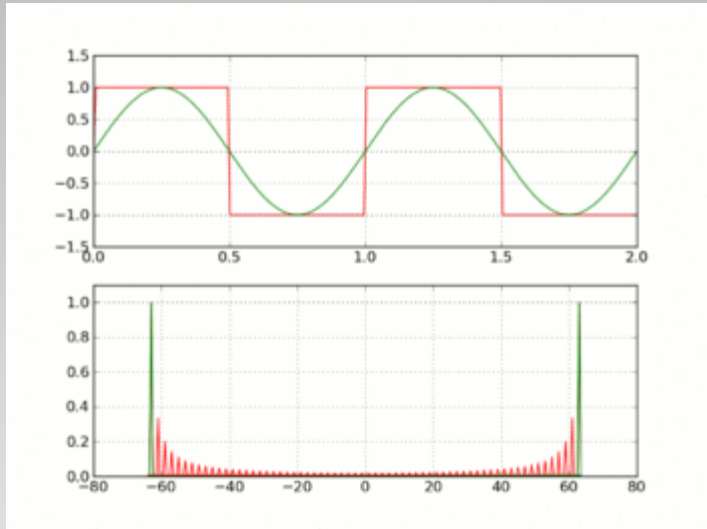


PWM e onda quadra

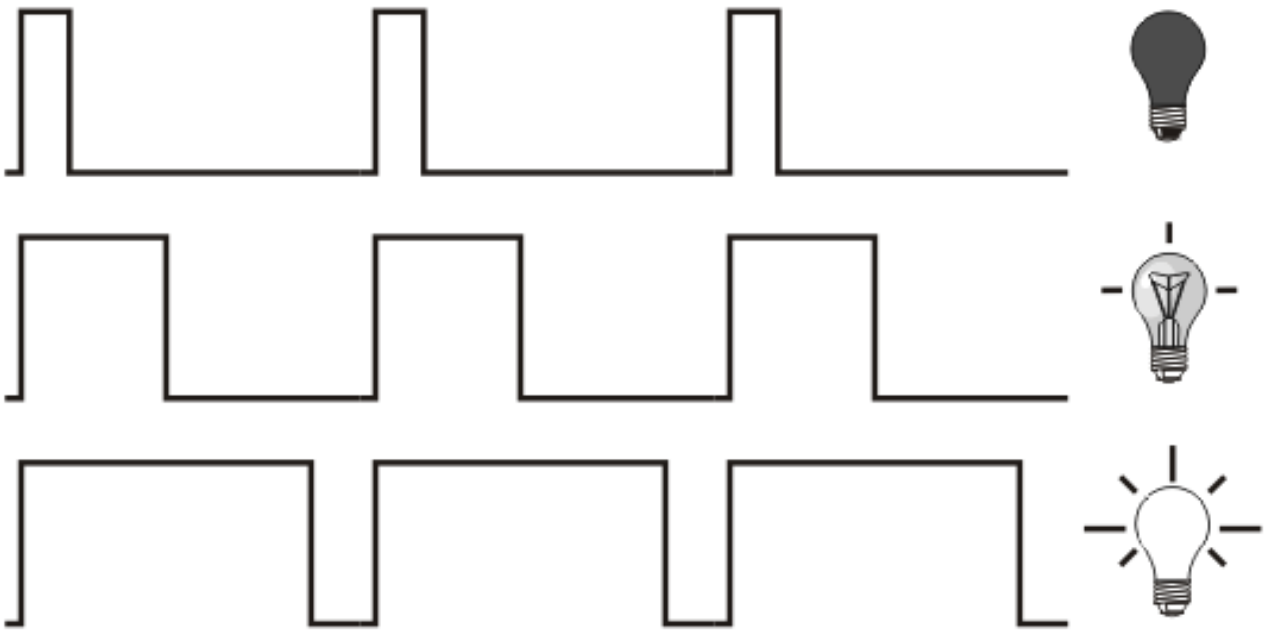
- L'onda quadra è un segnale periodico, cioè un segnale che si ripeto alo stesso modo ad intervalli di tempo uguale detto periodo T.
- Le caratteristiche di un segnale periodico sono: periodo e ampiezza.
- Le caratteristiche di un'onda quadra sono oltre al periodo e all'ampiezza, anche il duty cycle
- Il duty cycle è il rapporto tra il tempo t_h in cui l'onda assume valore alto e l'intero periodo T dell'onda stessa: t_h / T
  - Se il tempo in cui il segnale è alto è uguale a quello in cui il segnale è basso, il duty cycle è del 50% ;
  - se il segnale è sempre alto, il duty cycle è 100%;
  - Se il segnale è sempre basso, il duty cicle è 0%


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

Duty Cycle



Onda quadra come sovrapposizione di armoniche



PWM con i Pic

- Ci sono due metodi per generare la PWM con i microcontrollori:

A. I microcontrollori Pic hanno dei moduli per generare segnali PWM, CCP Capture /Compare/PWM
B. La PWM può essere implementata anche via software facendo variare il duty cicle dell'onda quadra.

es:

L’algoritmo in linguaggio di progetto è:

I=0 inizio ciclo	
Se I=0	Se il contatore è 0
Out=0	spegni
I I-1	Decrementa contatore
Se I=-1	
I=255	Riparte da 255
Se I=N	Se I=N byte di controllo
Out=1	accendi

In ogni ciclo il contatore I viene decrementato del suo valore massimo di 255 . Quando raggiunge N l’uscita si alza e resta alta fino a che il contatore non arriva a 0. il passaggio da 00000000 a 11111111 è automatico

```
// codice in microc


void main() {
    unsigned char cnt, lvl, i;
    TRISC = 0;
    PORTC = 0;
    cnt=0;
    vl=0;
     i=0;
     while(1) {if (cnt>lvl) {
    LATC.f0=0;} else {
     LATC.f0=1; cnt++;
     i++;
     if (i==200) {
    lvl++;
    i=0;}
          }
            }
```

- La variabile cnt è utilizzata come contatore e viene incrementata per ogni iterazione

Esempio PWM con pic

```
void main() {
unsigned char cont, level, i;
TRISB = 0;
PORTB = 0;
cont=0;
level=0;
i=0;
// Loop infinito
while(1) {
if (cont>level) {
PORTB=0;
} else {
PORTB=1;
}
cont++;
i++;
if (i==200) {
level++;
i=0;
}}}
```

Pwm per pic 16f628

Primo metodo:

- Molti pic sono dotati di moduli PWM
- Il Pic 16f887 ha due moduli PWM CCP1 e CCP2
- Il segnale PWM viene generato sul pin CCPx (CCP1 o CCP2). Il duty cycle, il perido e la risoluzione sono determinati dai seguenti registri
  - PR2
  - T2CON
  - CCPRxL (CCPR1L o CCPR2L)
  - CCPxCON (CCP1CON o CCP2CON)

  La risoluzione del segnale generato sul pin CCP può raggiungere al massimo 10 bit

  Il modulo PWM per funzionare ha bisogno di utilizzare come risorsa esclusiva il modulo Timer2.

PWM e Pic

Registro CCP1CON

CCP1CON							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0

CCP1X e CCP1Y sono utilizzati solo in PWM

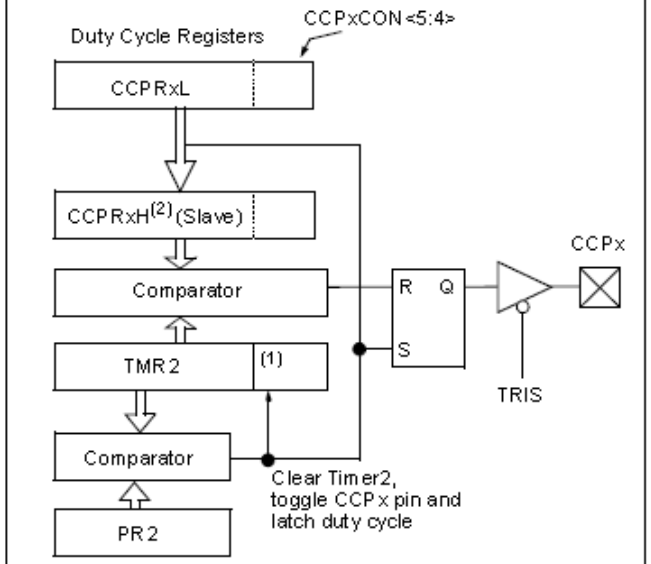
Per l'uso di un modulo CCP per la PWM bisogna eseguire i seguenti passl:

1. Definire il periodo del segnale PWM generato andando a scrivere un opportuno valore nel registro PR2 del PIC
2. Definire il modulo CCPx come modulo PWM
3. Abilitare il timer Tmr2 con opportuno prescaler
4. Definire il Duty cycle del segnale generato

CCP1M3	CCP1M2	CCP1M1	CCP1M0	Modalità
0	0	0	0	Disabilita tutti i modi
0	1	0	0	Capture:ogni fronte di discesa su RC2/CCP1
0	1	0	1	Capture:ogni fronte di salita su RC2/CCP1
0	1	1	0	Capture:ogni 4 fronti di salita su RC2/CCP1
0	1	1	1	Capture:ogni 16 fronti di salita su RC2/CCP1
1	0	0	0	Compare: porta a livello alto RC2/CCP1
1	0	0	1	Compare:porta a livello basso RC2/CCP1
1	0	1	0	Compare: RC2/CCP1 non varia, viene generato un interrupt
1	0	1	1	Compare: si generano eventi speciali
1	1	x	x	PWM

Settaggio bit di CCP1CON

FIGURE 11-3: SIMPLIFIED PWM BLOCK DIAGRAM

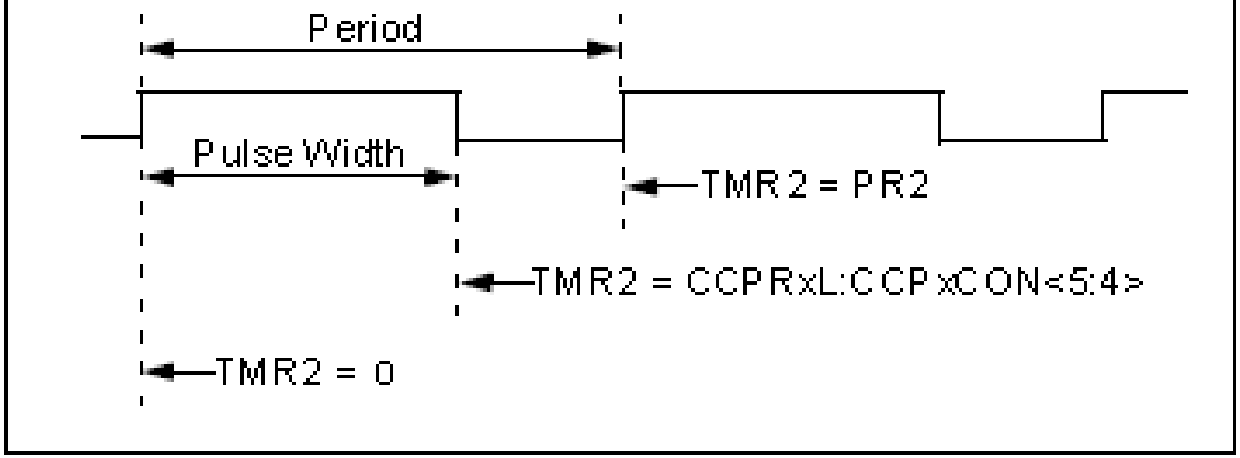


Note 1: The 8-bit timer TMR2 register is concatenated with the 2-bit internal system clock (Fosc), or 2 bits of the prescaler, to create the 10-bit time base.

2: In PWM mode, CCPRxH is a read-only register.

Schema della microchip

FIGURE 11-4: CCP PWM OUTPUT



OUTPUT

PWM e Pic

- Il duty cycle viene caricato in CCPR1RL (CCPR2L) e sul bit 5 e 4 di CCP1CON
- La lunghezza del DC è 10 bit
- Il valore caricato su PR2 viene confrontato con quello di TMR2 che è incrementato con la frequenza $f_{osc}/4$
- Quando i due valori sono uguali, viene posto alto il pin RC2/CCP1 TMR2 viene azzerato
- TMR2 viene confrontato con CCPR1L e i bit 4 e 5 del registro CCP1CON
- Quando c'è uguaglianza, viene posto a zero R2C/CCP1

Operazioni da compiere per utilizzare i moduli CCP

- Configurare il pin RC2/CCP1 come uscita
- Caricare in PR2 il valore del periodo della PWM
- Caricare nel registro CCPR1L e nei bit 5 e 4 di CCP1CON il valore del dc
- Configurare il prescaler in T2CON
- Configurare CCP1CON in modalità PWM

PWM e Pic

Settare la PWM

- PWM period = [(PR2) + 1] • 4 • TOSC • (TMR2 prescaler value)
- Per una frequenza di 2 kHz: *PWM period = 1/ Frequency (1/2000 = .0005)*
  0.0005 = [PR2 + 1] • [1 / 8000000] • 16
- PR2 + 1 = [.0005 • 8000000] / 16
- PR2 + 1 = 250
- PR2 = 249
- PR2 = 0xF9 (249 in hex)
- PR2 è un registro che contiene il periodo dell'onda quadra secondo la relazione sopra esposta
- Duty cycle=(CCPRIL*CCPICON<5:4>)*(TMR2)/Fosc

Assembly e mikroc

- Se si lavora in assembly è necessario conoscere come settare i registri della PWM
- Lavorando in mikroc, non è necessario conoscere tali registri perché esistono delle apposite librerie

```
int i=0;

void blink_up(){
  i++;
  PWM1_Set_Duty(i);
  Delay_ms(10);
}

void blink_down(){
  i--;
 PWM1_Set_Duty(i);
 Delay_ms(10);
}
```

```
void main() {
  trisc=0;
 portc=0;

 PWM1_Init(5000); //initilize PWM 1 at
    5kHz
 PWM1_Start();
 PWM1_Set_Duty(i);

 while(1){
  while(i != 1023){
  blink_up();
   }

  while(i!=0) {
 blink_down();
  }

} }
```

PWM con i microcontrollori pic esempio di programma in mikroc utilizzando i moduli CCP

Applicazioni

- Se su RC2 del pic 16f887 si connette un transistor di media potenza, si può pilotare la velocità di un motorino. La potenza che arriva al motore è pari al valore medio della PWM

- Convertitore digitale analogico a 10 bit. Se in uscita a RC2/CCP1 si connette un semplice filtro passa basso che tagli ad una frequenza che sia almeno la meta' di quella dell'onda Pwm, meglio se ancor piu' bassa, cioe' meno di 1 Hz, viene estratta la componente continua del segnale. In questo modo abbiamo ottenuto un convertitore digitale analogico a 10 bit, su 5 volt di escursione del segnale. Con 2 tasti UP e DOWN, connessi a due ingressi del PIC, andiamo a variare la parola di controllo del modulatore PWM. Per ogni pressione del tasto UP o del tasto DOWN abbiamo un incrmento/decremento di soli 5/1024 = 4.883 mV.

- Trimmer digitale. Esso puo' essere usato laddove si voglia variare una tensione continua senza utilizzare un classico trimmer a vite. Un applicazione classica puo' essere la variazione di contrasto di un lcd.

- Giocando opportunamente con dei cicli for e con la frequenza di taglio del filtro e' possibile generare onde triangolari, a dente di sega e, se si mappano i valori giusti in memoria, anche delle sinusoidi. Attraverso il modulo CCP2 del PIC16F877 è possibile realizzare un convertitore di tipo digitale analogico a 10 bit sfruttando il modulatore PWM presente all%u2019interno del suddetto modulo.
  Tramite il CCP2 è infatti possibile generare un segnali PWM a 1024 step di variazione del duty cicle.
  In questo modo otteniamo un segnale Pwm il cui duty cicle è proporzionale alla parola digitale, a 10 bit, di controllo del CCP2.

Modo compare

- Nella modalità compare il contenuto del registro CPR1 viene confrontato con quello di TMR1. Se c'è uguaglianza si verifica sul pin RC2/CCP1 uno dei seguenti eventi:
- RC2/CCP1 viene portato a livello alto
- RC2/CCP1 viene portato a livello basso
- RC2/CCP1 non subisce variazioni ma genera un interrupt
- Si genera un trigger in uno dei seguenti modi:
  - Viene posto a livello alto il bit CCP1IF e resettato TMR1
  - Viene posto a livello alto il bit CCP2IF e resettato TMR1; viene avviata una conversione A/D se il convertitore è abilitato

Modo capture

- Quando si manifesta un evento su R2C/CCP1, il registro a 16 bit CCPR1 (CCPR1H:CCPR1L) catturano il valore attuale contenuto su TMR1 che lavora in modalità timer
- L'evento che può attivare la cattura può essere:
  - Ogni fronte di salita di un segnale posto su RC2/CCP1
  - Ogni fronte di discesa di un segnale posto su RC2/CCP1
  - Ogni 4 fronti di salita di un segnale posto su RC2/CCP1
  - Ogni 16 fronti di salita di un segnale posto su RC2/CCP1
- La porta C2 deve essere impostata come input
- Quando si verifica una cattura, il bit 2 del registro PIR1 viene posto ad 1 se è stato precedentemente abilitato (posto a zero i bit 2 di PIE1).

PWM con Arduino

- void setup() { pinMode(13, OUTPUT); } void loop() { digitalWrite(13, HIGH); delayMicroseconds(100);

// circa il 10% del duty cycle con f= 1KHz
digitalWrite(13, LOW); delayMicroseconds(900); }

T=(100+900)microsecondi=1000 microsecondi=1000*10^{-6}=10^{-3} secondi=1 ms

f=1/T=10^{3} Hz=1kHz

dc=(100 microsecondi)/(1000microsecondi)*100=10%

- void setup() { pinMode(13, OUTPUT); }
  void loop() { digitalWrite(13, HIGH);
  delayMicroseconds(800);

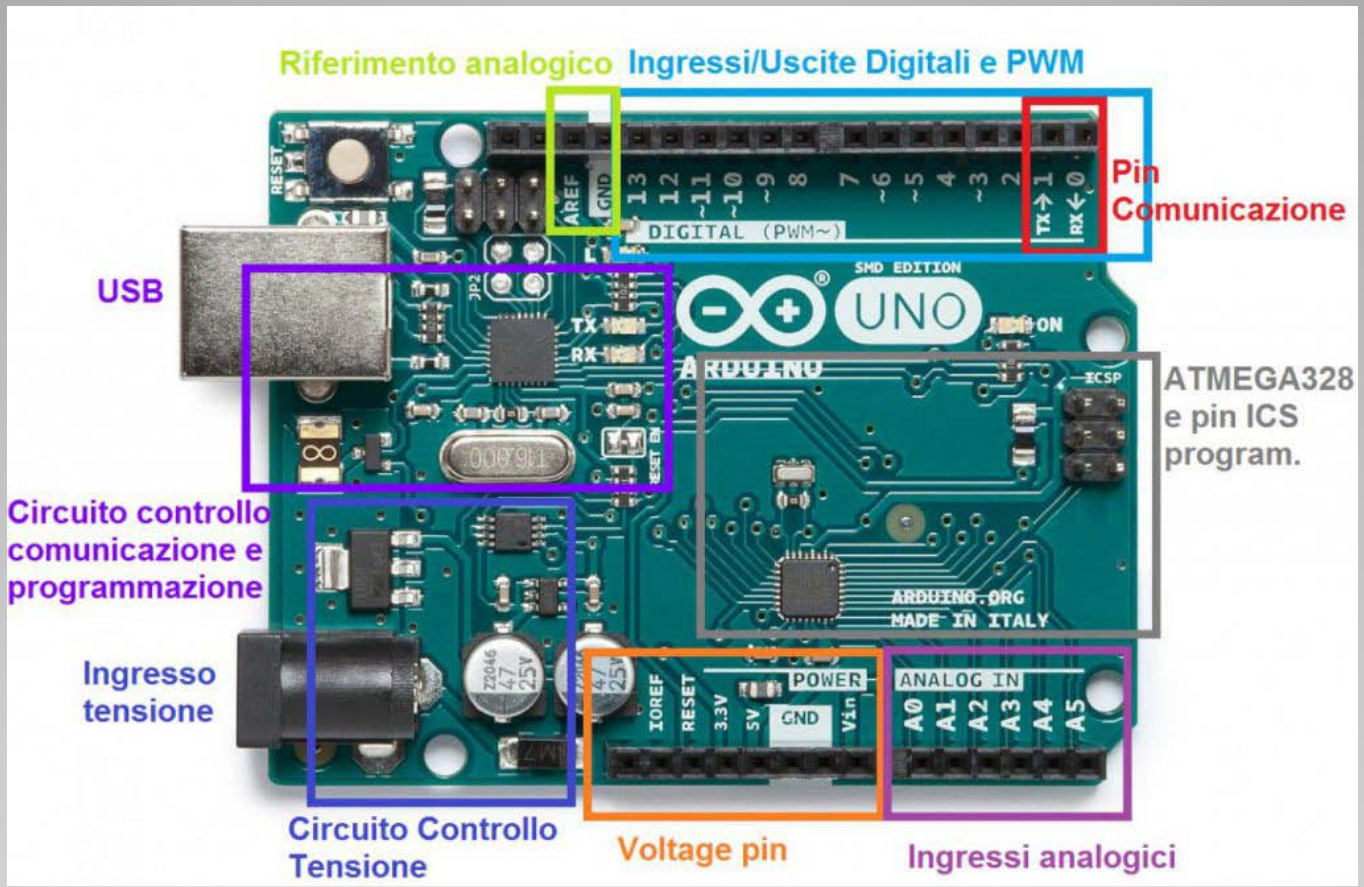
// circa il 80% del duty cycle con f= 1KHz
  digitalWrite(13, LOW);
  delayMicroseconds(200); }

T=(800+200)microsecondi=1000
  microcsecondi

dc=(800/1000)*100=80%

PWM Arduino

- Per arduino uno solo alcuni pin sono dedicati alla PWM:3,5, 6, 9,10,11 e sono anche a frequenza differenti
- La differenza di frequenza è dovuta al fatto che i pin sono collegati su memorie di 8 bit e di 16 bit
- I pin 3, 9,10 e 11 sono a 16 bit, gli altri a 8 bit
- Il valore di prescaler è quello con divisore 8
- Sui pin 5 e 6 si trova una PWM a frequenza pari a 976 Hz mentre sui pin 3,9,10,11 la frequenza è 490 Hz
- Sulla scheda Arduino i pin con la PWM sono indicati con il simbolo ~


Riferimento analogico
Ingressi/Uscite Digitali e PWM
Pin Comunicazione
USB
ATMEGA328 e pin ICS program.
Circuito controllo comunicazione e programmazione
Ingresso tensione
Circuito Controllo Tensione
Voltage pin
Ingressi analogici
AREF
GND
DIGITAL (PWM~)
TX→1
RX←0
SMD EDITION
UNO
ARDUINO
ICSP
ARDUINO.ORG
MADE IN ITALY
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
Vin
A0
A1
A2
A3
A4
A5

Fattore di prescaler

T_{ck} = 62.5 ns f_{ck} = 16 MHz

k	Ts	fs
1	16 µs	62,5 kHz
8	128 µs	7,81 kHz
32	512 µs	1,95 kHz
64	1,024 ms	976,5 Hz
128	2.048 ms	488,8 Hz
256	4,096 ms	244,1 Hz
1024	16,38 4ms	61 Hz

k fattore di prescaler
Ts periodo del segnale PWM
fs frequenza del segnale PWM (1/Ts)

Table 10-2. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 5.0V
0	0	0	0	2K (2048) cycles	16 ms
0	0	0	1	4K (4096) cycles	32 ms
0	0	1	0	8K (8192) cycles	64 ms
0	0	1	1	16K (16384) cycles	0.125 s
0	1	0	0	32K (32768) cycles	0.25 s
0	1	0	1	64K (65536) cycles	0.5 s
0	1	1	0	128K (131072) cycles	1.0 s
0	1	1	1	256K (262144) cycles	2.0 s
1	0	0	0	512K (524288) cycles	4.0 s
1	0	0	1	1024K (1048576) cycles	8.0 s
1	0	1	0	Reserved	
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

Fattori di prescaler

```
int led =3;
 void setup(){
 pinMode(led,OUTPUT);}
 void loop(){
 analogWrite(led, 125);//dc al 50%
delay(1000);
analogWrite(led, 255); //dc al 100%
 delay(2000);
 analogWrite(led, 90);dc al 35%
 delay(3000);
}
```

Per far variare il valore di duty cycle, si scrive *analogWrite(nomepin, valore);*
Il valore varia da 0 255 perché stiamo lavorando a 8 bit

Esempio PWM Arduino

```
#define LED 11
 int valoreFade = 0;
void setup()  {
  pinMode(LED, OUTPUT);     }
void loop()
   {
for (valoreFade = 0 ; valoreFade < 255; valoreFade++
   ) {
 analogWrite(LED, valoreFade);
 delay(10);                                  }
for(valoreFade = 255 ; valoreFade > 0; valoreFade--
   ) {      analogWrite(LED, valoreFade);
 delay(10);                    }   }
```

Un terzo programma PWM per Arduino

PWM e frequenza

Timer	Frequenza Base	Output Compare		Pin Arduino	Frequenza Default
TCCR0	62500Hz	OC0	OC0A	6	976Hz
	62500Hz		OC0B	5	976Hz
TCCR1	31250Hz	OC1	OC1A	9	488Hz
	31250Hz		OC0B	10	488Hz
TCCR2	31250Hz	OC2	OC1A	11	488Hz
	31250Hz		OC2B	3	488Hz

```
void setPwmFrequency(int pin, int divisor) {
  byte mode;
  if(pin == 5 || pin == 6 || pin == 9 || pin == 10) {
    switch(divisor) {
      case 1: mode = 0x01; break;
      case 8: mode = 0x02; break;
      case 64: mode = 0x03; break;
      case 256: mode = 0x04; break;
      case 1024: mode = 0x05; break;
      default: return;
    }
    if(pin == 5 || pin == 6) {
      TCCR0B = TCCR0B & 0b11111000 | mode;
    } else {
      TCCR1B = TCCR1B & 0b11111000 | mode;
    }
  } else if(pin == 3 || pin == 11) {
    switch(divisor) {
      case 1: mode = 0x01; break;
      case 8: mode = 0x02; break;
      case 32: mode = 0x03; break;
      case 64: mode = 0x04; break;
      case 128: mode = 0x05; break;
      case 256: mode = 0x06; break;
      case 1024: mode = 0x7; break;
      default: return;
    }
    TCCR2B = TCCR2B & 0b11111000 | mode;
  }
}
```

Cambiare la frequenza di PWM