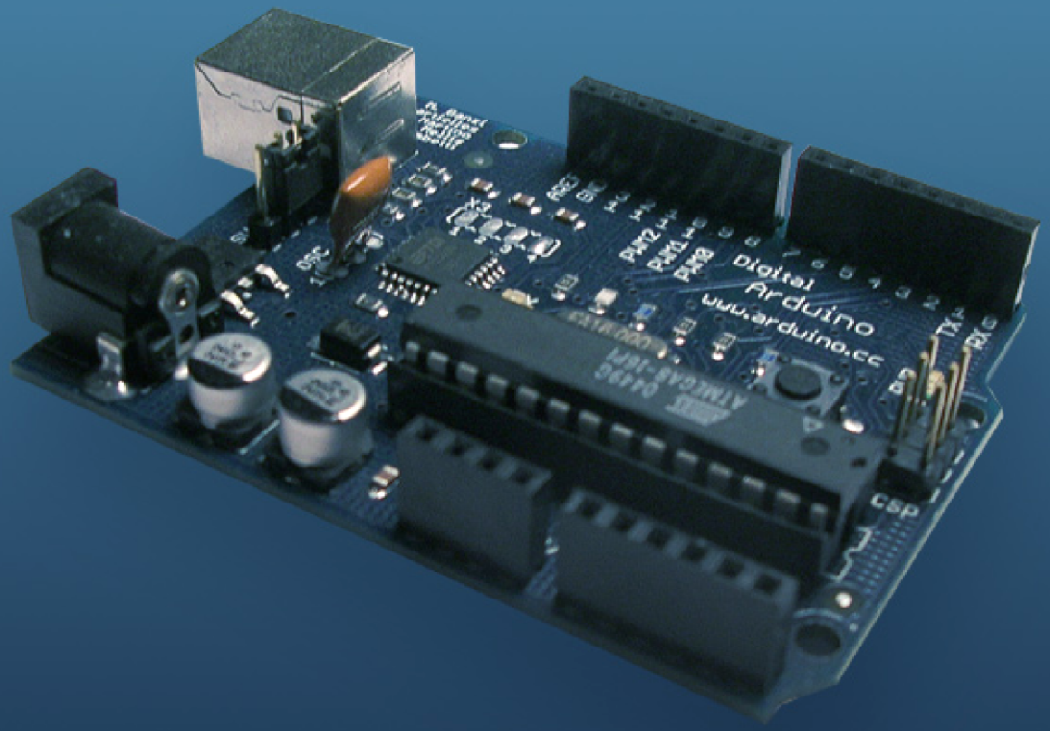
Arduino

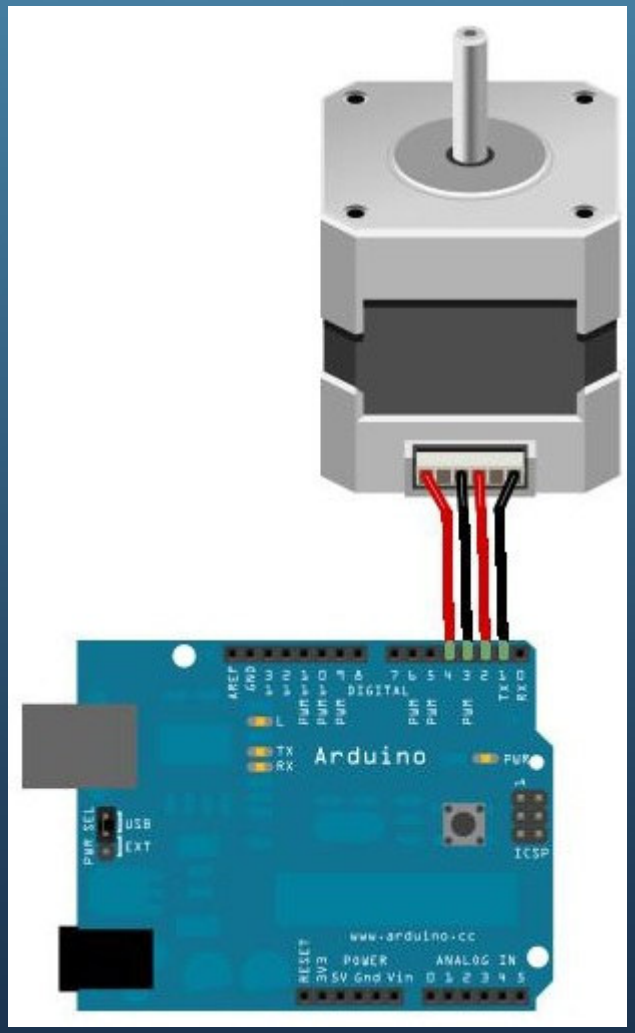
Physical Computing I/O board



8^ parte : Motor shield FE e motore passo-passo bipolare

Author: Ing. Sebastiano Giannitto (ITIS “M.BARTOLO” –PACHINO)

1° modo di gestione di un motore passo-passo bipolare



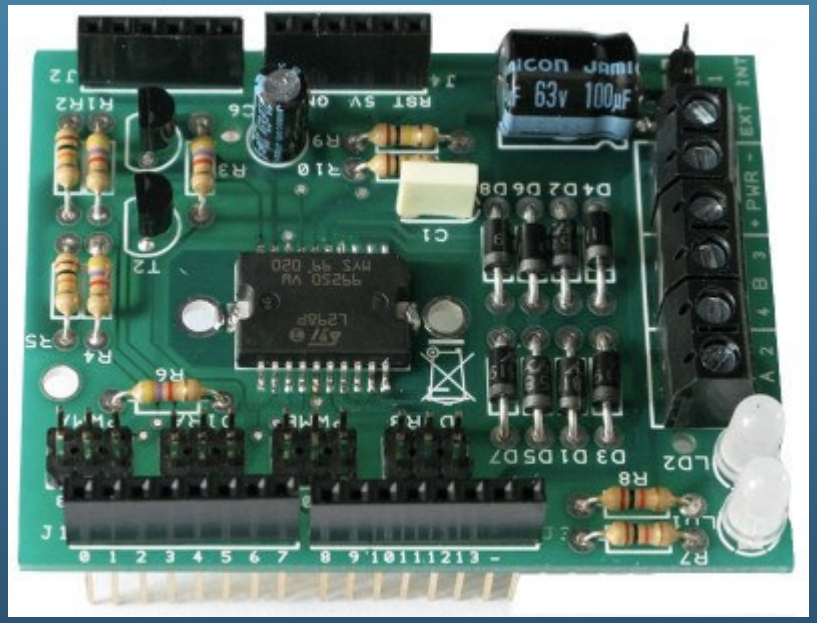

```
// set pin numbers:
const int a1 = 1;
const int a2 =  2;
const int b1 =  3;
const int b2 =  4;

void setup() {
  // initialize pins
  pinMode(a1, OUTPUT);
  pinMode(a2, OUTPUT);
  pinMode(b1, OUTPUT);
  pinMode(b2, OUTPUT);

  digitalWrite(a1, LOW);
  digitalWrite(a2, LOW);
  digitalWrite(b1, LOW);
  digitalWrite(b2, LOW);
}

void loop()
{
  step1();
  delay(10);
  step2();
  delay(10);
  step3();
  delay(10);
  step4();
  delay(10);
 }
```

```
void step1 ()
{
  digitalWrite(a1, HIGH);
  digitalWrite(a2, LOW);
  digitalWrite(b1, LOW);
  digitalWrite(b2, LOW);
}

void step2 ()
{
  digitalWrite(a1, LOW);
  digitalWrite(a2, LOW);
  digitalWrite(b1, HIGH);
  digitalWrite(b2, LOW);
}

void step3 ()
{
  digitalWrite(a1, LOW);
  digitalWrite(a2, HIGH);
  digitalWrite(b1, LOW);
  digitalWrite(b2, LOW);
}

void step4 ()
{
  digitalWrite(a1, LOW);
  digitalWrite(a2, LOW);
  digitalWrite(b1, LOW);
  digitalWrite(b2, HIGH);
}
```

Perché questo modo di gestire un motore passo-passo non va bene.

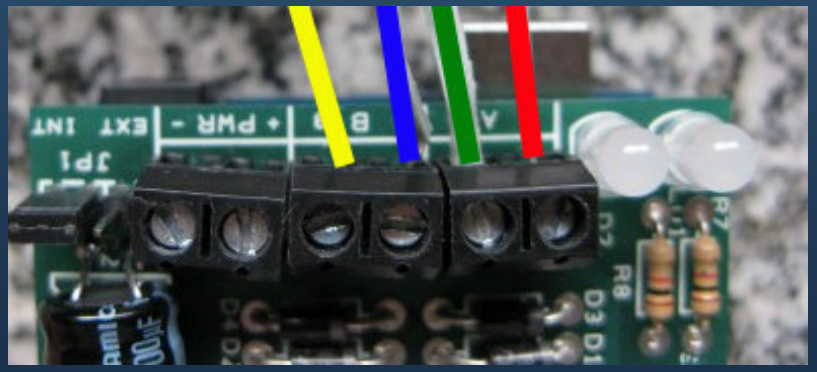
Un pin di Arduino può fornire solo un massimo di 40 mA.
Potrebbe fornire più, ma a rischio di danni permanenti.
20mA è più realistico per la longevità del nostro arduino.
Il motore da noi utilizzato richiede una corrente di 330ma. Pertanto si ricorre a un H-bridge e a una alimentazione esterna per i motori che permetterà di fornire la corrente richiesta dal motore e al nostro arduino di durare più a lungo.
In secondo luogo, un motore causerà alcuni picchi di tensione all'accensione. Senza la corretta protezione, questi picchi saranno pericolosi per la nostra scheda arduino ... di nuovo causando danni.
Alcuni **H-bridge** hanno questa misura di sicurezza tramite l'uso di un opportuno integrato come **L298** o utilizzando l'ultima versione della motor shield_fe che utilizza l'**L298P**

Motor Shield_FE per pilotare i motori passo passo bipolari

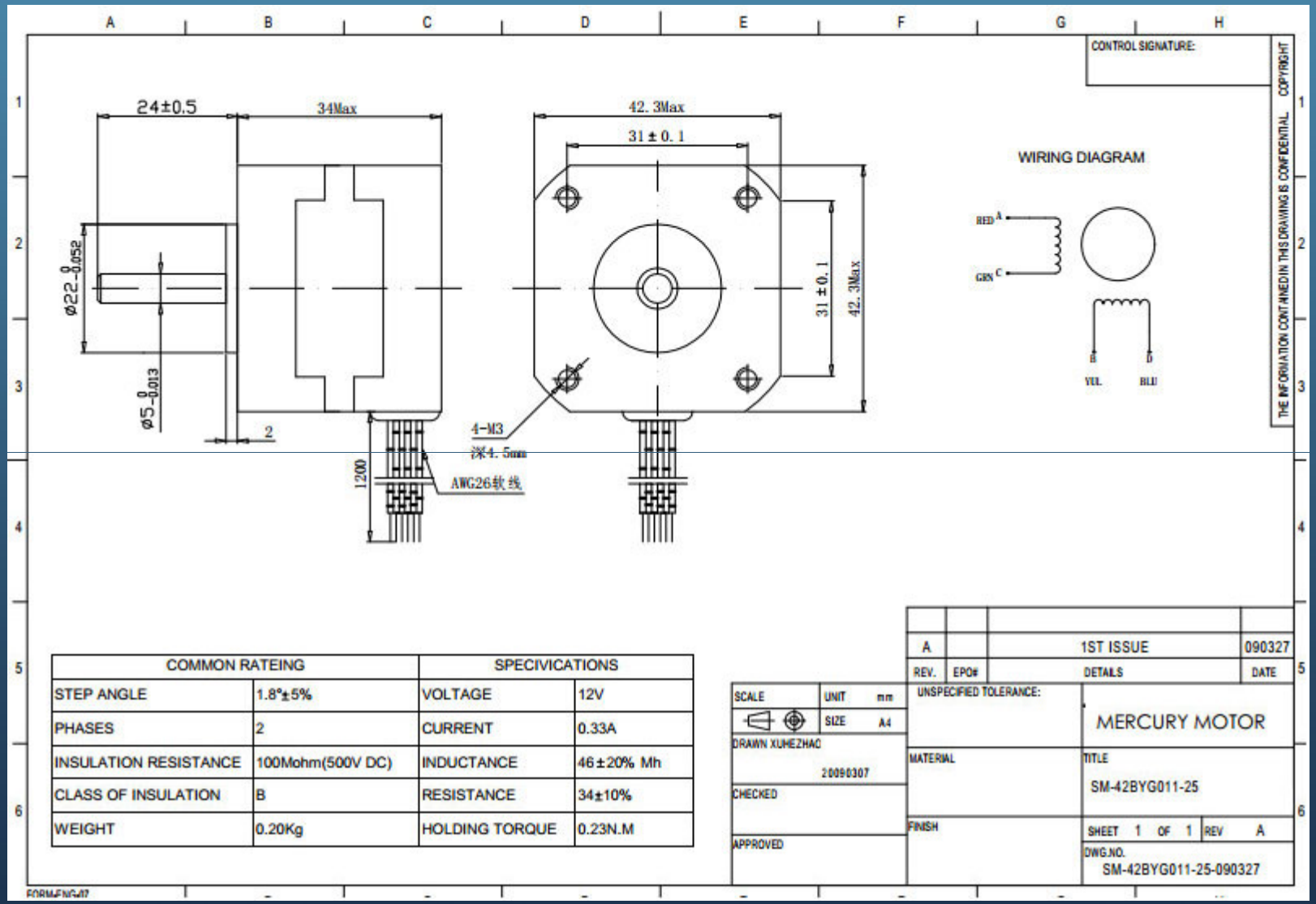


La motor shield FE può essere utilizzata anche per pilotare un motore passo-passo, unipolare o bipolare.
I bipolari sono più difficili da gestire ma anche più semplici da trovare nelle stampanti di recupero.
La motor shield ha 2 morsetti, da due contatti ciascuno, a cui connettere le 4 fasi del motore. Nel bipolare gli avvolgimenti sono pilotati a due a due ciascuno polarizzando correttamente la bobina corrispondente.

Occorre, pertanto, dopo aver individuato le 4 fasi del motore, collegare la prima (A+=rosso) e la terza (A-=verde) ai morsetti del motore A e la seconda (B+=giallo) e quarta (B-=blu) ai morsetti del motore B, come in figura.



Scheda tecnica del bipolare da noi usato.

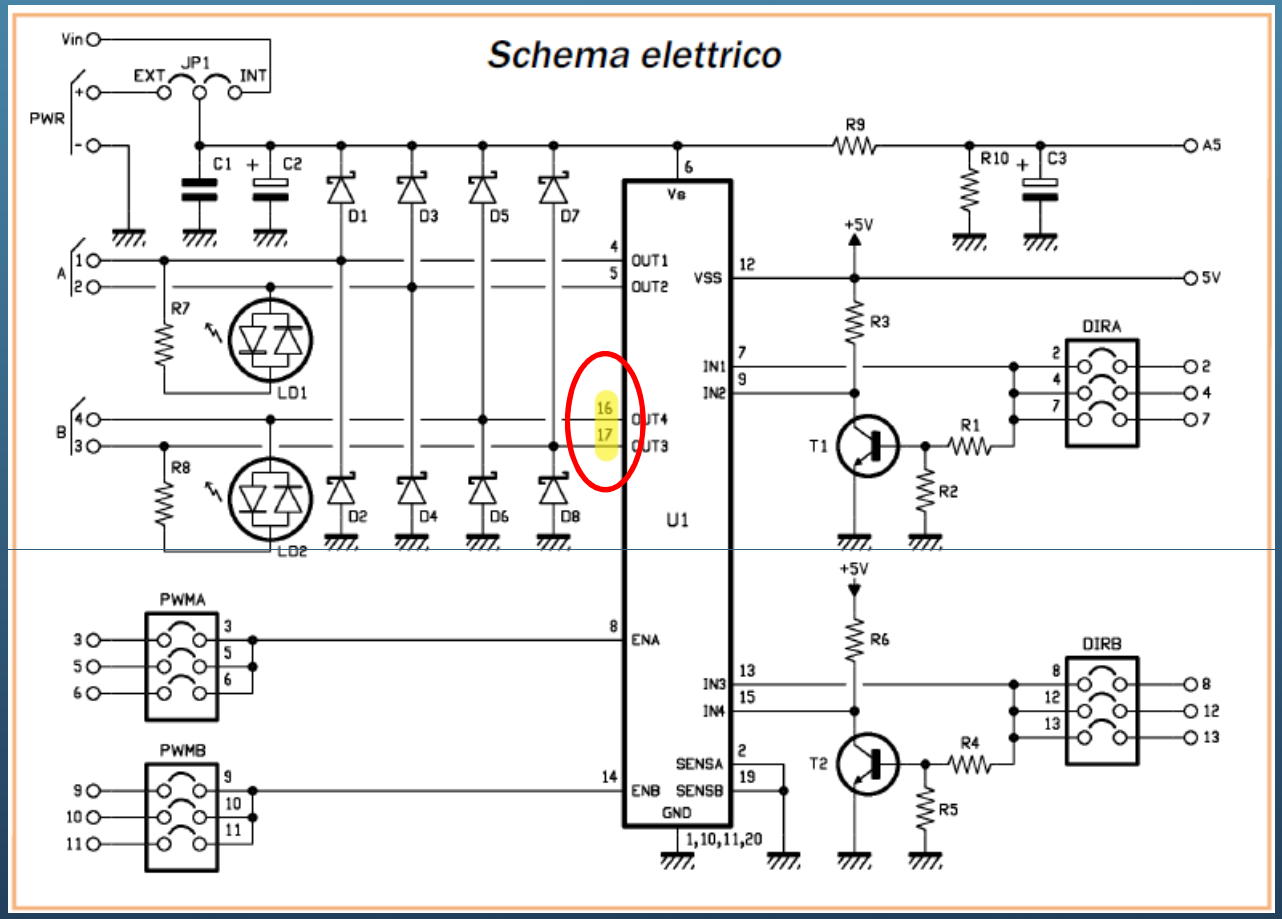


COMMON RATEING		SPECIVICATIONS	
STEP ANGLE	1.8°±5%	VOLTAGE	12V
PHASES	2	CURRENT	0.33A
INSULATION RESISTANCE	100Mohm(500V DC)	INDUCTANCE	46±20% Mh
CLASS OF INSULATION	B	RESISTANCE	34±10%
WEIGHT	0.20Kg	HOLDING TORQUE	0.23N.M

REV.	EPO#	DETAILS	DATE
A		1ST ISSUE	090327

SCALE | UNIT mm | SIZE A4
DRAWN XUMEZHAO 20090307
CHECKED
APPROVED
UNSPECIFIED TOLERANCE:
MATERIAL
FINISH
MERCURY MOTOR
TITLE SM-42BYG011-25
SHEET 1 OF 1 REV A
DWG.NO. SM-42BYG011-25-090327

La motor shield_fe si basa sul driver a ponte H L298, in particolare questo driver accetta 4 segnali di input: IN1, IN2, IN3, IN4 e 2 segnali di enable EN1 ed EN2.



Nella motor shield_fe i pin ENx sono ciascuno l'abilitazione della coppia di INx corrispondente, in pratica perchè IN1 e IN2 accettino un segnale e lo commutino su OUT 1 e OUT2 (uscite corrispondenti) è necessario che EN1 sia ad un livello logico alto (+5v); uguale situazione è valida per IN3 e IN4 con EN2.

Dobbiamo considerare anche che IN1 e IN2 sono pilotati da segnali invertiti mediante l'utilizzo di transistor rispetto ad un unico pin che li collega ad Arduino, analogamente avviene per IN3 e IN4.
Questo comporta che se IN1 è a livello logico alto (+5v) IN2 è certamente a livello logico basso (0v).

Lo sketch di esempio

```
int motorPinDirA = 2;
int motorPinDirB = 8;
int motorPinPwmA = 3;
int motorPinPwmB = 9;
int delayTime = 100;

void setup() {
  pinMode(motorPinDirA, OUTPUT);
  pinMode(motorPinDirB, OUTPUT);
  pinMode(motorPinPwmA, OUTPUT);
  pinMode(motorPinPwmB, OUTPUT);
}
```

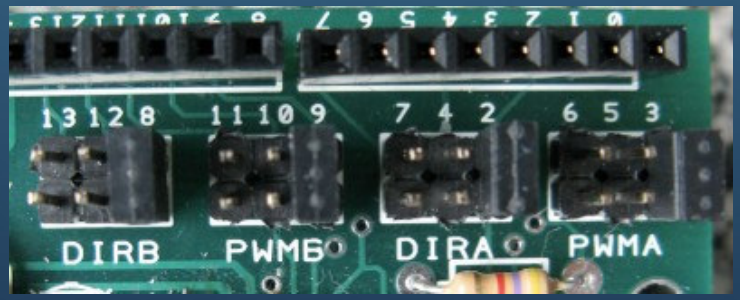
```
void loop()
{
  digitalWrite(motorPinDirA, HIGH);
  digitalWrite(motorPinDirB, LOW);
  digitalWrite(motorPinPwmA, HIGH);
  digitalWrite(motorPinPwmB, LOW);
  delay(delayTime);

  digitalWrite(motorPinDirA, LOW);
  digitalWrite(motorPinDirB, HIGH);
  digitalWrite(motorPinPwmA, LOW);
  digitalWrite(motorPinPwmB, HIGH);
  delay(delayTime);

  digitalWrite(motorPinDirA, LOW);
  digitalWrite(motorPinDirB, LOW);
  digitalWrite(motorPinPwmA, HIGH);
  digitalWrite(motorPinPwmB, LOW);
  delay(delayTime);

  digitalWrite(motorPinDirA, LOW);
  digitalWrite(motorPinDirB, LOW);
  digitalWrite(motorPinPwmA, LOW);
  digitalWrite(motorPinPwmB, HIGH);
  delay(delayTime);
 }
```

```
pinMode(motorPinDirA, OUTPUT);
pinMode(motorPinDirB, OUTPUT);
```

Si definisco le due variabili di tipo *integer* a cui sono collegati i pin di controllo della direzione A e B, ossi a IN1 e IN3, e di conseguenza sono invertiti IN2 e IN4, questi pin possimao modificarli sulla motor shield spostando i ponticelli illustrati nella figura sotto;

```
pinMode(motorPinPwmA, OUTPUT);
pinMode(motorPinPwmB, OUTPUT);
```

Si definisco i pin collegati ai segnali PWM, in questo esempio non li utilizzeremo come tali per cui sono solo i pin di enable EN1 e 2, anche questi li possiamo modificare grazie ai ponticelli di selezione presenti sulla scheda e rappresentati in figura.



In figura i pin DIRA è connesso al pin 2 di arduino, DIRB al pin 8, PWMA al pin 3 e PWMB al pin 9 come nello sketch, possiamo spostare tutti i ponticelli come preferisci, ricordando di cambiare le impostazioni dei pin nello sketch;

```
int delayTime = 100;
```

Definiamo un tempo di attesa, nell'esempio di 100 millisecondi, di attesa tra una fase e la successiva;

```
pinMode(motorPinDirA, OUTPUT);
pinMode(motorPinDirB, OUTPUT);
pinMode(motorPinPwmA, OUTPUT);
pinMode(motorPinPwmB, OUTPUT);
```

Impostiamo tutti i pin definiti alle linee precedenti come pin di OUTPUT mediante il comando ***pinMode*** di arduino.

```
digitalWrite(motorPinDirA, HIGH);
digitalWrite(motorPinDirB, LOW);
digitalWrite(motorPinPwmA, HIGH);
digitalWrite(motorPinPwmB, LOW);
```

E' la prima fase del motore, abilita la conduzione degli input IN1 e IN2 portando a livello logico alto il PWM corrispondente, a sua volta connesso all'EN1, quindi PWMA impostato a livello logico alto (+5v), e IN1 a livello logico alto (PinDirA), di conseguenza IN2 andrà da solo a livello logico basso portando l'uscita OUT2 a 0v mentre OUT1 a livello alto;

```
delay(delayTime);
```

Imposta il tempo di attesa *delayTime*

```
digitalWrite(motorPinDirA, LOW);
digitalWrite(motorPinDirB, HIGH);
digitalWrite(motorPinPwmA, LOW);
digitalWrite(motorPinPwmB, HIGH);
```

E' la seconda fase, ad essere interssati in questo caso sono i pin DirB e PwmB che vanno a livello logico alto per consentire a IN3 e EN2 di andare a livello logico alto e di conseguenza all'OUT3;

```
digitalWrite(motorPinDirA, LOW);
digitalWrite(motorPinDirB, LOW);
digitalWrite(motorPinPwmA, HIGH);
digitalWrite(motorPinPwmB, LOW);
```

E' la fase 3 e il segnale su EN1 deve tornare ad essere alto, per portare in conduzione IN1 e IN2 secondo il segnale impostato su DirA, quindi imposta DirA a LOW (0v) per ottenere che IN1 vada a livello logico basso e IN2 alto;

```
digitalWrite(motorPinDirA, LOW);
digitalWrite(motorPinDirB, LOW);
digitalWrite(motorPinPwmA, LOW);
digitalWrite(motorPinPwmB, HIGH);
```

E' l'ultima fase del motore, ad essere coinvolto è tutto il canale B, per cui PwmB va a livello logico alto e DirB a livello logico basso, tradotto sul tuo L298 significa IN3 a 0v e IN4 a +5v e OUT4 a +5V;

A questo punto il ciclo ricomincia facendo compiere all'albero motore altri 4 passi.

E' importante che sia chiaro che ogni fase corrisponde da un passo del motore ma che allo stesso tempo 4 passi non è detto che facciano compiere un giro completo al motore, quando il motore è un 48 passi, ad esempio, sono necessari 12 cicli *loop()* di arduino per compiere un giro completo (48 / 4 = 12).