Controllo motori con Arduino e la scheda Adafruit Motorshield v2

2014-05-10 00:05:45 Fabio Nelli

Introduzione

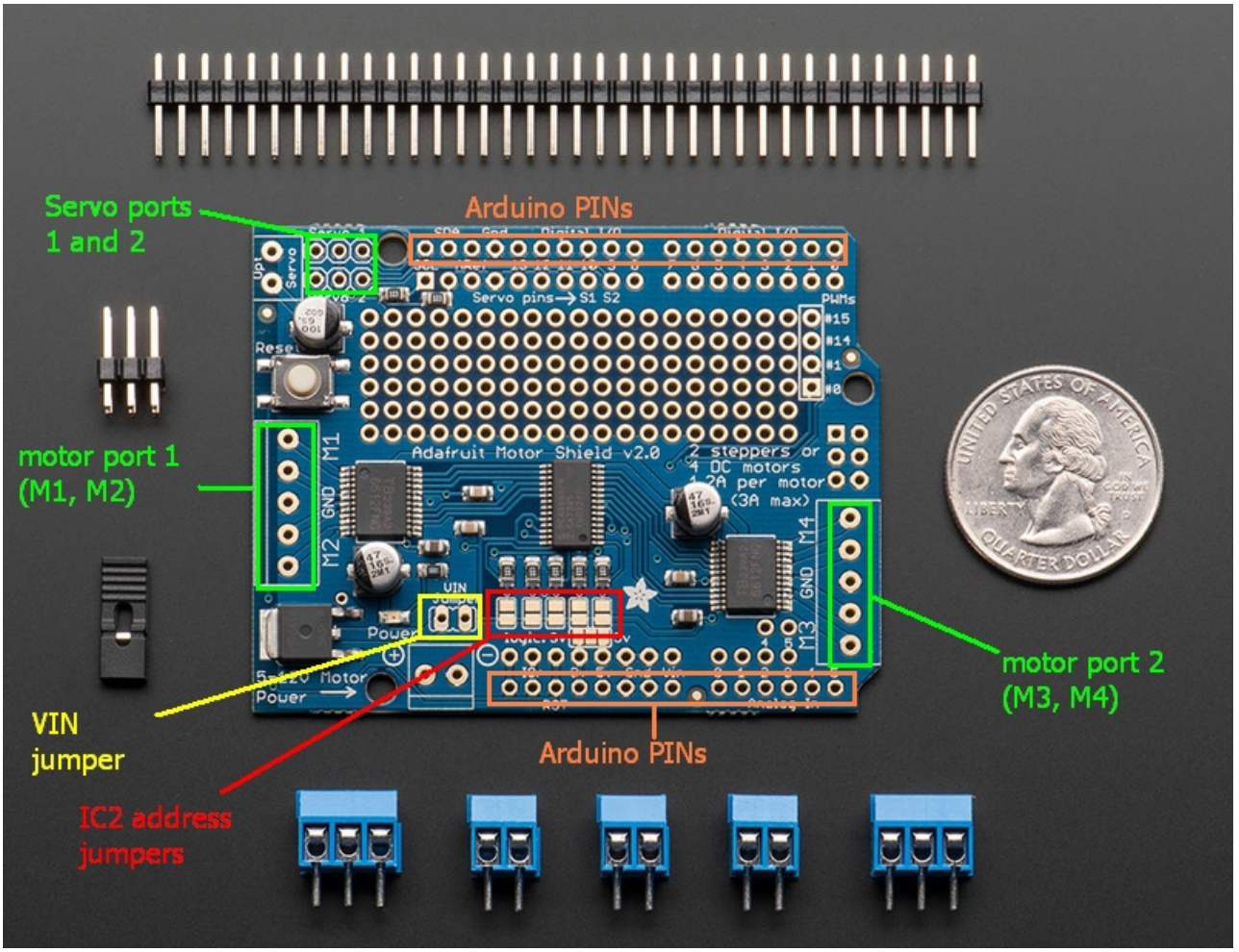
Avendo a disposizione una scheda Arduino UNO, molto probabilmente dopo aver realizzato i tuoi primi circuiti, ti sarà venuta certamente la voglia di utlizzare la tua scheda per poter controllare un motore. Tra l'ampia scelta di motori che si possono utilizzare:

- **motori DC (a corrente continua)**
- **motori passo-passo**
- **servo motori**

ci sono miriadi di tutorial **specifici** sia per tipologia di motore che per il voltaggio.

Ma come posso fare pratica con tutti questi motori cercando di utilizzare meno materiale possibile, cioè mantenendomi in una situazione più generale possibile?

Adafruit ha realizzato recentemente la seconda versione di una scheda **shield** per Arduino: l'**Adafruit Motorshield v2**. In Italia è possibile ordinarla tramite Amazon.

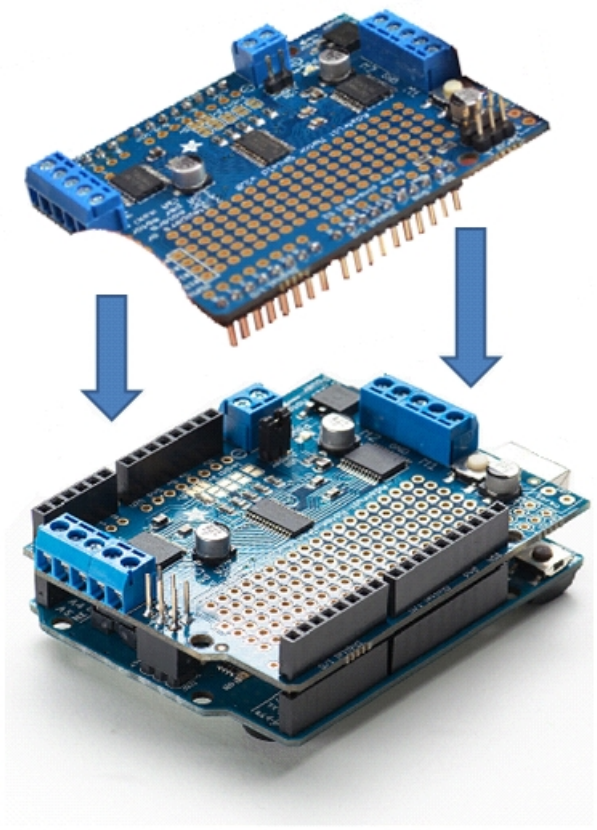


Questa scheda permette di poter pilotare e controllare tutte e tre le tipologie di motore (solo motori a bassa potenza). E con una sola scheda si possono controllare contemporaneamente:

- 4 motori DC

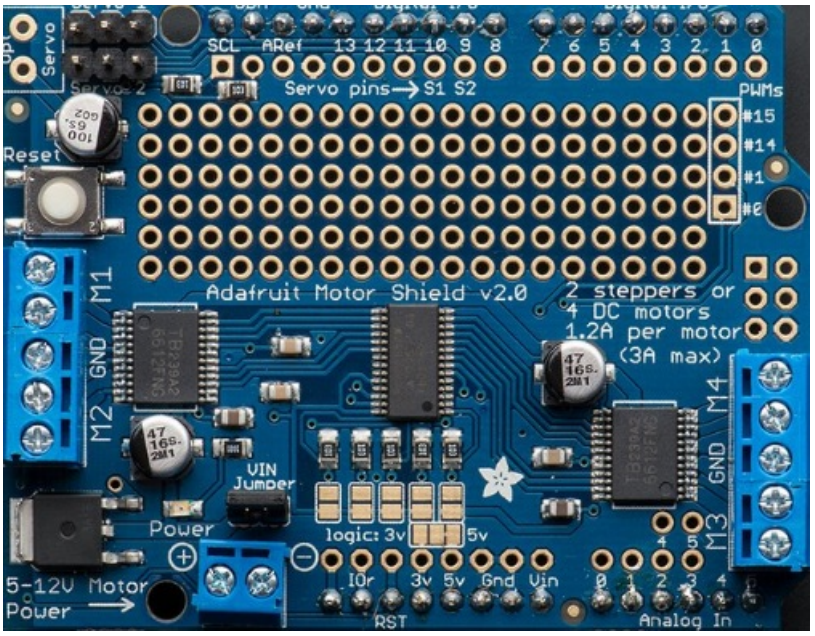
- 2 motori passo-passo
- 2 servo motori

Ma in realtà, questa scheda è uno **stackable shield** che utilizza il protocollo **I2C** per cominicare con Arduino, quindi più schede dello stesso tipo possono essere montate una sopra l'altra, ognuna con un suo indirizzo I2C, e portando così il controllo di un numero illimitato di motori da una sola scheda Arduino.



La scheda Adafruit Motorshield v2

Il cuore di questo shield è il motor driver **Toshiba TB6612FNG**, le specifiche di questo chip di controllo le puoi trovare qui. Nella figura lo potete riconoscere nella parte centrale della scheda.



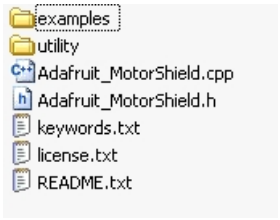
Per programmare questa scheda mediante l'IDE di Arduino è disponibile una libreria specifica per questa versione.

La libreria Adafruit Motorshield v2.

Per usare lo shield su Arduino, sarà necessario installare la libreria **Adafruit Motorshield v2 library** in modo che sia direttamente utilizzabile nell'IDE di Arduino. Puoi scaricarla direttamente da GitHub:

https://github.com/adafruit/Adafruit_Motor_Shield_v2_Library

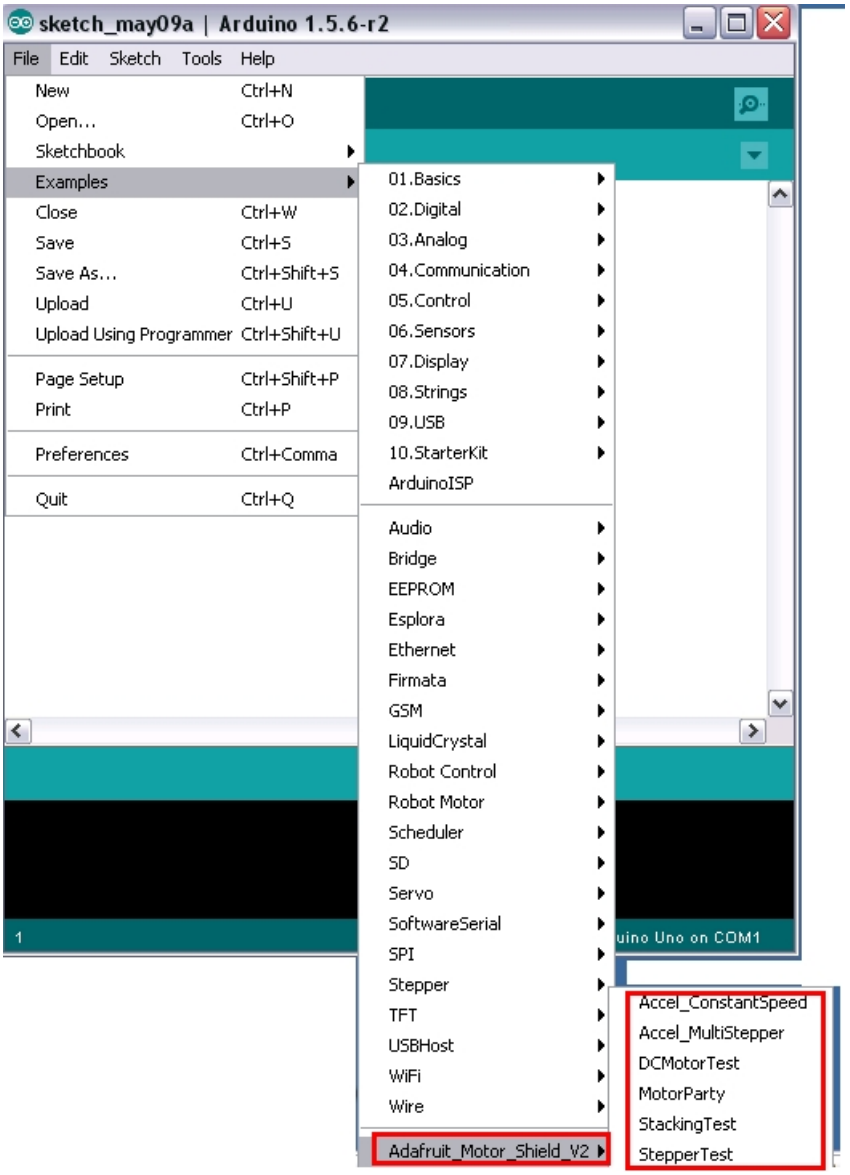
Attenzione: in rete è disponibile anche la libreria per la versione 1 della scheda Motorshield. Bene, questa libreria non è compatibile con la seconda versione. E' riconoscibile perchè spesso viene riferita come AF_Motor. Se nutrite dei dubbi su quello che state utilizzando, puoi facilmente controllare andando a vedere la sua struttura interna che deve corrispondere alla seguente:



Dal sito Github scaricherai un file ZIP contenente la libreria. Una volta terminato estrai il suo contenuto e copialo all'interno della directory Arduino/libraries che troverai all'interno della cartella Documents (qui ci riferiamo ad un ambiente Windows):

C://users/myname/My Documents/Arduino/libraries

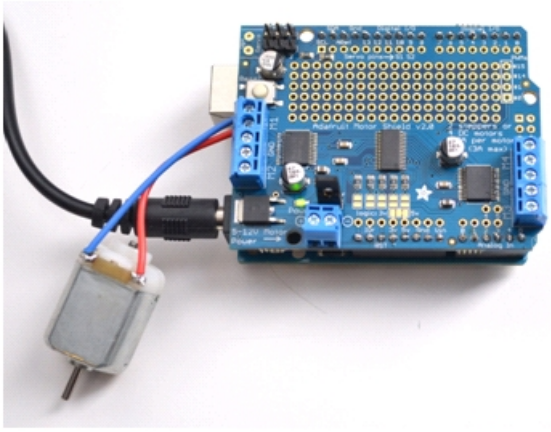
Se adesso lanci l'IDE di Arduino, troverai che si è aggiunta una voce agli esempi "Adafruit Motorshield", cliccandoci sopra troverai una serie di esempi già caricati.



MOTORI DC

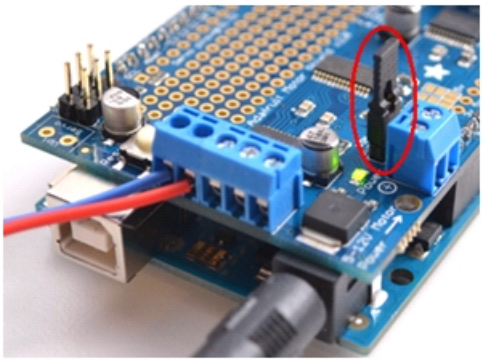
Possiamo cominciare a lavorare con i motori DC. La scheda **Adafruit MotorShield** è in grado di controllare fino a 4 motori di questo tipo contemporaneamente, motori che richiedano un voltaggio compreso tra 5-12V.

Inserisci lo shield sulla scheda Arduino e connetti il motore DC alla **porta motore 1 (M1)**. Per questo tipo di motori non è importante con quale ordine i due fili devono essere inseriti nei morsettti M1. Mi raccomando di non inserire uno dei due cavi nel pin centrale (GND). Nella foto sottostante puoi vedere come il motore DC è stato connesso con lo shield. Assicurati, dopo aver inserito i due cavi nei morsetti, che le viti siano ben avvitate in modo da garantire una buona connessione.



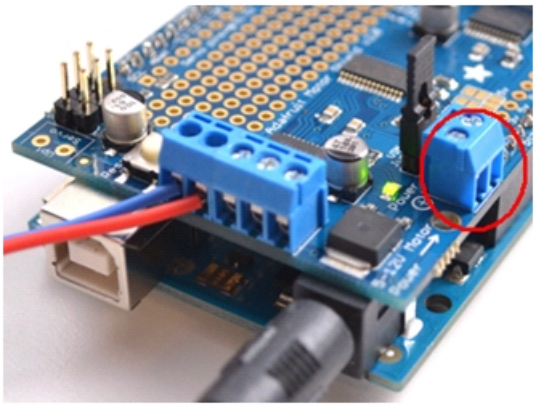
Ci sono due modi diversi per alimentare il motore:

1) Puoi fornire l'alimentazione sia alla scheda Motorshield che al motore DC attraverso la scheda di Arduino che sarà collegata alla corrente attraverso un **DC Barrel Jack** o attraverso la **porta USB.** In questo caso dovrai inserire (se non lo è già) il **jumper VIN** (lo riconosci dalla foto sottostante: è quello vicino al LED verde che indica la corretta alimentazione dello shield)



Sei il LED verde non è brillante, ma ha una illuminazione tenue o ad intervalli, non continuare. Ciò significa che l'alimentazione che può fornire Arduino non è sufficiente ad alimentare anche lo shield ed i motori ad esso collegati. Allora in questi casi, dovrai alimentare la scheda seguendo le indicazioni riportate nel prossimo punto.

2) E' possibile alimentare direttamente la scheda Motorshield, e sarà questa a fornire poi l'alimentazione ai vari motori, attraverso la porta 5-12V DC. Questa porta è rappresentata dal morsetto doppio indicato nella figura sottostante, anch'esso vicino al LED verde di alimentazione. In questo caso, ricordatevi di **rimuovere il jumper VIN**.



Nell'IDE, carica, compila e poi esegui il seguente esempio: *File->Example->Adafruit_MotorShield->DCMotorTest*

Come risultato dovresti vedere e sentire il motore DC ruotare in una direzione e poi nell'altra. Prima partirà accelerando raggiungendo la velocità massima, poi decelerando sempre più fino a fermarsi ed infine partire nella direzione opposta a quella precedente. Il ciclo poi si ripeterà infinitamente. Se non riesci a vedere l'asse girare puoi aggiungere una piccola striscia di adesivo bianco all'asse rotante del motore; in questo modo potrai seguire più chiaramente il movimento del motore.

Adesso analizziamo insieme il codice dell'esempio.

Per prima cosa, sarà necessario includere tutte le librerie necessarie.

```
#include
&lt;Servo.h&gt;
```

```
1 #include &lt;Servo.h&gt;
2 #include &lt;Adafruit_MotorShield.h&gt;
3 #include "utility/Adafruit_PWMServoDriver.h"
4
```

La libreria **Wire** : questa libreria permette ad Arduino di comunicare con dispositivi I2C/TWI. Sulla scheda Arduino, ci sono due pin dedicati proprio al protocollo I2C: **A4** per **SDA** (data line) e **A5** per **SCL** (clock line).

Una volta definite le librerie, la prima istruzione da dichiarare è la creazione dell'oggetto Motorshield:

```
Adafruit_MotorShi
eld AFMS =
1 Adafruit_MotorShield AFMS = Adafruit_MotorShield();
2
```

La classe ***Adafruit_MotorShield*** rappresenta una singola scheda Motorshield e deve essere sempre dichiarata all'inizio se vogliamo utilizzare qualsiasi oggetto o funzione appartenente alla libreria. Se abbiamo montato sopra Arduino più schede Motorshield, utilizzando la configurazione **"stacked"**, dovrai dichiare un oggetto MotorShield per ciascuna scheda impilata.

Il costruttore

Adafruit_MotorShield(uint8_t addr = 0x60);

accetta come argomento (opzionale) l'indirizzo I2C dello shield. L'indirizzo di default è 0x060 che è l'indirizzo assegnato di default a ciascuna scheda Adafruit motorshield se non si apporta alcuna modifica ad esse (vedi più avanti come modificare questo indirizzo). Se hai bisogno di utilizzare più schede Motorshield dovrai definire altrettanti oggetti utilizzando per ciascuno l'indirizzo I2C corrispondente.

```
Adafruit_DCMotor
*myMotor =
1 Adafruit_DCMotor *myMotor = AFMS.getMotor(1);
2
```

La funzione

Adafruit_DCMotor *getMotor(uint8_t n);

restituisce uno dei 4 possibili motori DC che possono essere controllati dallo shield. Il parametro ***n*** specifica proprio il motore a cui riferirsi: 1-4.

```
void setup(){
  AFMS.begin();
1 void setup(){
2   AFMS.begin();
3 }
4
```

La funzione

void begin(uint16_t freq = 1600);

deve essere sempre dichiarata all'interno della funzione *setup()*. La funzione begin() ha lo scopo di inizializzare lo shield. Questa funzione riceve un solo argomento (anche questo opzionale), ***freq*** che è corrisponde alla frequenza PWM. Se non specificato, per default, la frequenza utilizzata sarà di 1.6KHz.

```
void setup(){
  ...
```

```
1 void setup(){
2   ...
3   myMotor-&gt;setSpeed(150);
4   myMotor-&gt;run(FORWARD);
5   myMotor-&gt;run(RELEASE);
6 }
7
```

La funzione

void setSpeed(uint8_t speed);

controlla il livello di potenza fornito dalla scheda al motore. Il parametro ***speed*** assume valori compresi tra 0 e 255.

Nota: la funzione setSpeed() controlla solamente la potenza fornita al motore DC. La velocità effettiva con cui il motore gira dipende da parecchi altri fattori, inclusi:

- il tipo di motore
- l'alimentazione fornita
- il carico

La funzione

void run(uint8_t direction);

imposta la direzione in cui il motore ruota. Il parametro **direction** può assumere solo tre valori:

- FORWARD (rotazione in avanti)
- REVERSE (rotazione indietro)
- RELEASE (blocca la rotazione)

Attenzione che le direzioni specificate dai parametri "forward" e "backward" sono del tutto arbitrarie. Se, una volta montato il sistema, la direzione non dovesse corrispondere con quella desiderata, tutto ciò che dovrai fare sarà invertire i cavi collegati al morsetto M1.

Il codice seguente può essere utilizzato per far muovere il motore con una accelerazione costante partendo da fermo fino a raggiungere la velocità massima (255).

```
void loop(){
  ...
```

```
1 void loop(){
2   ...
3   uint8_t i;
4   myMotor-&gt;run(FORWARD);
5   for(i=0; i&lt;255; i++){ myMotor-&gt;setSpeed(i);
6     delay(10);
7   }
8 }
9
```

Invece se desideriamo decellerare la rotazione del motore dalla massima velocità fino all'arresto della rotazione, allora sarà necessario utilizzare il codice seguente.

```
void loop(){
  ...
```

```
1 void loop(){
2   ...
3   for(i=255; i!=0; i--){
4     myMotor-&gt;setSpeed(i);
5     delay(10);
6   }
7 }
8
```

SERVO MOTORI

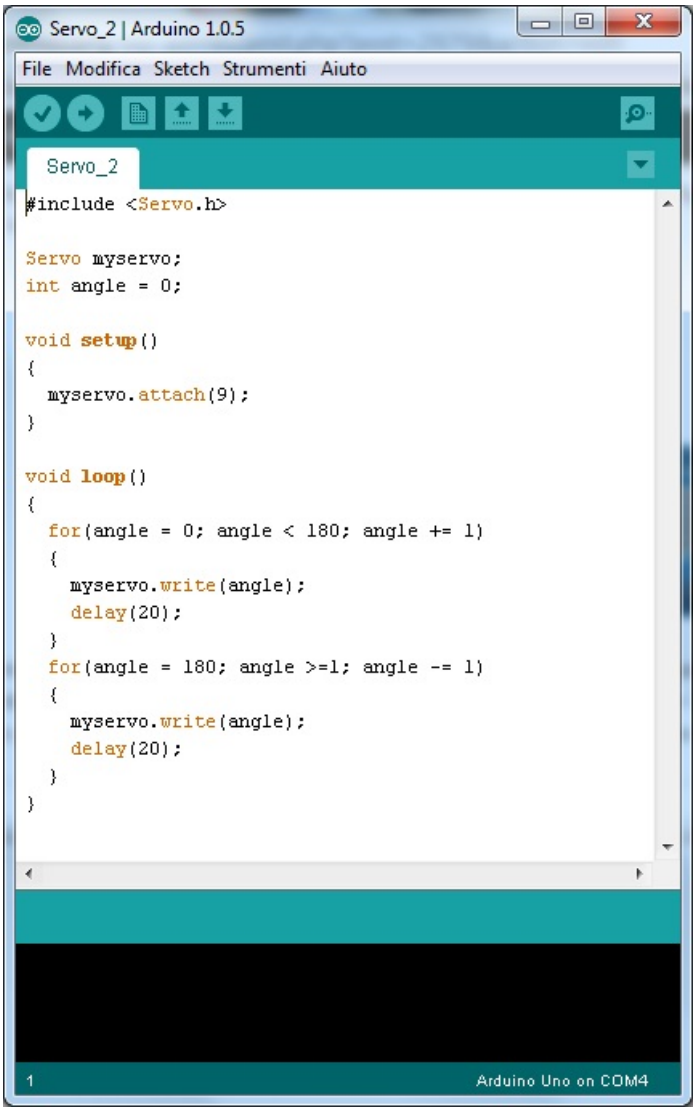
I servo motori da hobby sono un buon punto di partenza per prendere dimestichezza con il **motion control**(branca dell'automazione). Questo tipo di motori ha un cavo a 3-pin (+5V,ground, signal). La scheda Adafruit motorshield, in questo caso, agisce solo come un'intermediario, trasmettendo il segnale PWM a 16bit da Arduino al servo motore. Quindi quando si ha a che fare con i servo motori non c'è alcun bisogno di utilizzare la libreria **Adafruit Motorshield**, ma si utilizzerà invece la libreria **Servos**, già inclusa nell'IDE di Arduino.

```
#include
&lt;Servo.h&gt;
```

```
1 #include &lt;Servo.h&gt;
2
```

Per connettere il servo motore nel modo corretto, bisogna rispettare la giusta collocazione dei cavi a seconda dei colori, come mostrato in figura. Lo shield può controllare fino a due servo motori. Se c'è bisogno di controllare più motori servo, è meglio utilizzare altre soluzioni invece che utilizzare più schede Motorshield in configurazione stacked (presto lo spiegherò in un articolo).

Nella libreria non c'è un esempio specifico per il controllo dei servo motori. Comunque ecco qui un esempio che possa ben delucidare le stesse funzionalità viste per i motori DC e che vedremo anche per i motori passo-passo.



Adesso analizziamo insieme il codice.

```
Servo myServo;
int angle = 0;
```

```
1 Servo myServo;
2 int angle = 0;
3 void setup(){
4    myServo.attach(9);
5 }
6
```

Devi dichiarare ciascun servo motore che vuoi utilizzare, usando la classe ***Servo***. Con la funzione ***attach()*** puoi assegnare un pin a ciascun specificandolo come argomento della funzione.

attach(10) -> assegni il pin out Servo1 sulla scheda al motore.

attach(9) -> assegni il pin out Servo2 sulla scheda al motore;

Con il codice seguente, si fa ruotare il servo motore in entrambe le direzioni coprendo tutto l'intervallo degli angoli possibili [0-180°] e viceversa.

```
void loop(){
  for(angle=0;
```

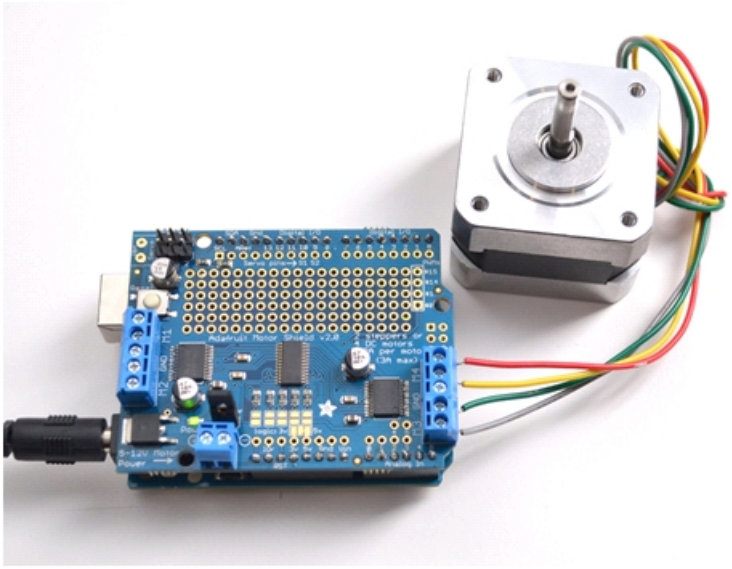
```
  void loop(){
1   for(angle=0; angle &lt; 180; angle += 1)
2 {                  myServo.write(angle);                  delay(20);
3 }               for(angle=180; angle &gt;= 1; angle -= 1){
4     myServo.write(angle);
5     delay(20);
6   }
7 }
```

I servo motori sono spesso utilizzati proprio per il loro basso costo (pochi euro). Inoltre esistono servo motori che possono ruotare anche di 360° effettuando un giro completo (a costo maggiore), vedi qui.
.

MOTORI PASSO-PASSO

La scheda **Adafruit Motorshield** può controllare sia motori passo-passo unipolari (a 5-cavi e 6-cavi) e bipolari (4-cavi). Non è in grado di controllare motori passo-passo con un diverso numero di cavi! Il codice che utilizzeremo è lo stesso sia per motori unipolari e bipolari.

In questo caso, differentemente dai motori DC, l'ordine dei cavi da inserire nei morsetti è importate. Così per prima cosa, è necessario conoscere la funzione di ciascun cavo. E' possibile connettere due motori passo-passo contemporaneamente sulla shield la quale fornisce due porte motori, ciascuna composta da morsetti a 5 pin.



I cavi della bobina 1 (coil) del motore deve essere connessa ai due pin terminali in alto per ciascuna porta motore (M1 or M3) mentre i cavi della bobina 2 invece devono essere connessi ai due pin terminali più in basso (M2 or M4). Quindi a seconda del tipo di motore passo-passo, dovrai fare coneessioni diverse.

- Se hai un **motore bipolare**, riconoscibile dai **4 cavi** in uscita, connetti le due bobine alle porte motori come indicato sopra, lasciando vuoto il pin ventrale GND.
- Se hai un **motore unipolare a 5-cavi**, connetti oltre i 4 cavi indicati prima, anche il cavo in comune alla massa GND.
- Se hai un motore unipolare a 6-cavi, puoi connettere entrambe i due cavi centrali di bobina direttamente alla massa GND.

E' necessario fornire un ulteriore alimentazione da 5-12V DC allo shield se si ha intenzione di ottenere dalla scheda buoni risultati (per 1 solo motore senza carico l'alimentazione fornita dalla scheda Arduino è più che sufficiente). Controlla comunque sempre che il LED verdi sia sempre acceso e in modo brillante.

Una volta che hai finito di fare tutte le connessioni necessarie, sei pronto per compilare ed eseguire il codice.

C'è un esempio fornito dalla libreria pronto per essere utilizzato su questo tipo di motori.

Nell'IDE carica, compila ed esegui il seguente esempio: *File->Example->Adafruit_MotorShield->StepperTest*

Eseguendo l'esempio vedrai il motore passo-passo far ruotare il suo asse avanti e indietro in maniera molto simile a quella che avevamo visto per i motori DC.

Analizziamo il codice. Per prima cosa includiamo anche qui tutte le librerie necessarie.

```
#include
&lt;Servo.h&gt;
```

```
1 #include &lt;Servo.h&gt;
2 #include &lt;Adafruit_MotorShield.h&gt;
3 #include "utility/Adafruit_PWMServoDriver.h"
4
```

Poi, avremo bisogno di dicharare un oggetto Adafruit_MotorShield:

```
Adafruit_MotorShi
eld AFMS =
```

```
1 Adafruit_MotorShield AFMS = Adafruit_MotorShield();
2
```

Adesso, è necessario dichiarare un oggetto motore per ciascun motore che vogliamo collegare alla scheda.

```
Adafruit_StepperM
otor *myMotor =
```

```
1 Adafruit_StepperMotor *myMotor = AFMS.getStepper(200,2);
2
```

La funzione

Adafruit_StepperMotor *getStepper(uint16_t steps, uint8_t n);

restituisce uno dei due possibili motori passo-passo utilizzabili dallo shield che sarà specificato dal parametro ***n*** passato come secondo argomento alla funzione getStepper. Il parametro ***steps*** specifica il numero di passi per ciascuna rivoluzione. Se non si conosce questo valore, è necessario calcolarlo. Questo valore varia da motore a motore. Per esempio un motore con 7.5 gradi/passo ha 360/7.5 = 48 passi per rivoluzione. Invece, per quanto riguarda il parametro **n**, se connetti il motore alle porte M1 e M2, allora dovrai specificare n come 1, altrimenti se connetti il motore alle porte M3 e M4 allora dovrai specificare n come 2.

Una volta definiti gli oggetti motore, è il momento di definire la funzione **setup()**.

```
void setup(){
  AFMS.begin();
```

```
1 void setup(){
2   AFMS.begin();
3   myMotor-&gt;setSpeed(10);
4 }
5
```

A differenze dei motori DC, il parametro che viene passato alla funzione **setSpeed()** è la velocità reale espressa in rpm.

Nella gunzione **loop()** definiremo tutti i comandi che caratterizzano i movimenti del motore. Ogni volta che si desidera che il motori ruoti di un certo angolo, si scriverà la funzione:

void step(uint16_t steps, uint8_t dir, uint8_t style = SINGLE);

Il parametro ***steps*** specifica di quanti step il motore si deve muovere, il parametro ***dir*** specifica la direzione (FORWARD o BACKWARD) e il parametrio **style** specifica lo stile dello stepping (SINGLE, DOUBLE, INTERLEAVED or MICROSTEP).

La funzione **step()** è sincrona e non restituisce il controllo finchè tutti gli step non siano stati completati. Quando saranno completati il motore rimarra comunque sotto potenza in modo da mantenere la posizione.

```
void loop(){
  myMotor-
```

```
1 void loop(){
2   myMotor-&gt;step(100, FORWARD, SINGLE);
3   myMotor-&gt;step(100, BACKWARD, SINGLE);
4 }
5
```

Vediamo in dettaglio i quattro stili di rotazione.

- **SINGLE** significa l'attivazione a bobina-singola
- **DOUBLE** significa che le due bobine siano attivate insieme (per una torsione maggiore)
- **INTERLEAVE** significa che si avrà alternanza tra bobine singole e doppie in modo da ottenere coì una risoluzione doppia (ma ovviamente a metà della velocità)
- **MICROSTEP** è un metodo dove le bobine sono soggette a PWM in modo da creare un moto fluido attraverso i vari passi del motore.

Per default, il motore manterrà la posizione dopo che ciascun passo verrà eseguito. Per si desidera il rilascio da parte di tutte le bobine, sarà necessario chiamare la funzione:

void release();

Questa funzione rimuove l'alimentazione dal motore. Se non desideri mantenere la coppia di forze attiva durante altre operazioni, è buona prassi chiamare questa funzione in modo da ridurre i consumi di potenza che un motore richiede per mantenere la posizione.

Impostare un indirizzo I2C allo shield

Se desideri mettere varie schede in configurazione stack, è necessario impostare un indirizzo I2C diverso per ciascuna scheda. Se si lascia invariata una scheda così come l'abbiamo acquisatat, allora l'indirizzo di default sarà **0x60**. Ma è possibile assegnare qualsiasi indirizzo compreso tra i valori 0x60 e 0x80, per un totale di 32 indirizzi I2C univoci.

Si possono assegnare tali indirizzi, utilizzando gli **address jumpers** presenti sul bordo inferiore della scheda. Per impostare l'indirizzo, è sufficiente saldare un ponte di stagno al corrispondete address jumper, ciascuno corrispondente ad un bit dell'indirizzo in forma binaria. Il jumper più a destra è il bit 0, poi seguito a sinistra dal bit 1, e così via fino ad arrivare al bit 5.