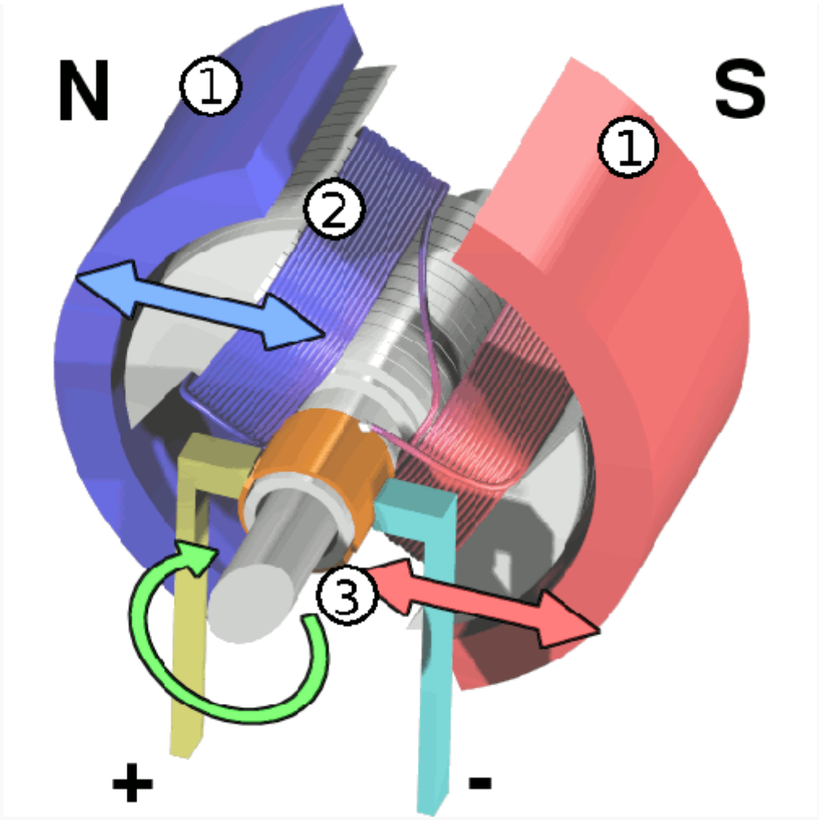
Arduino e Motori DC



Motore DC

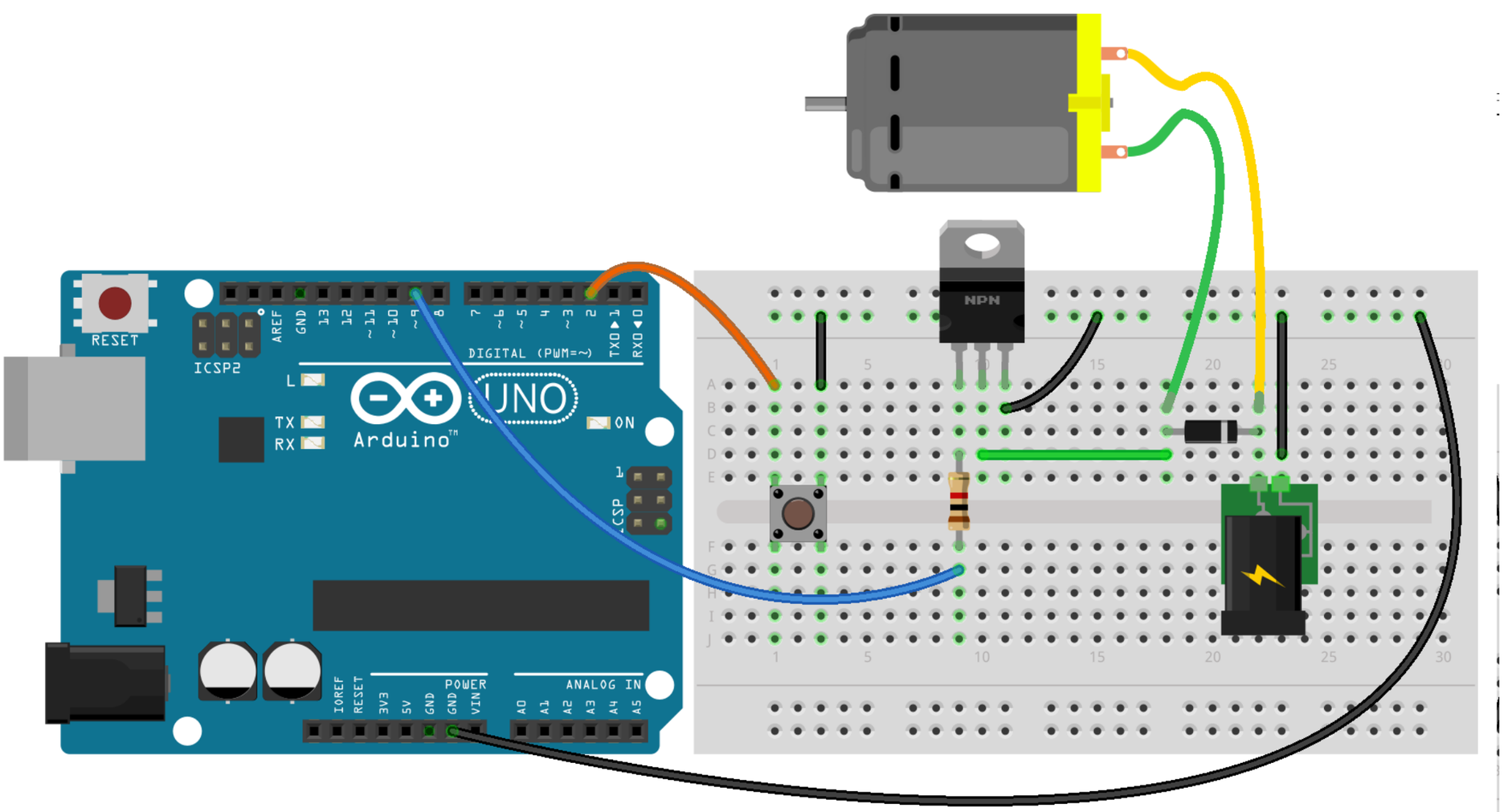
I motori a **corrente continua** (DC dall'inglese *direct current*) hanno soltanto due fili (positivo e negativo). Per farli girare è sufficiente alimentarli con una piccola batteria (in genere 5V sono sufficienti). Al loro interno è presente un magnete permanente, situato sul corpo cilindrico (1), un'elettrocalamita situata sull'asse (2) e dei contatti struscianti (*spazzole*) che alimentano l'elettrocalamita (3). Applicando una tensione l'elettrocalamita si polarizza in maniera opposta al magnete e tende a ruotare per allineare il suo Nord con il Sud del magnete e viceversa. I contatti struscianti sono disposti in modo tale da invertire la polarità dell'elettrocalamita ogni mezzo giro, quando essa si allinea col magnete. Questo fa sì che l'asse continui a ruotare, finché riceve corrente, senza mai stabilizzarsi.

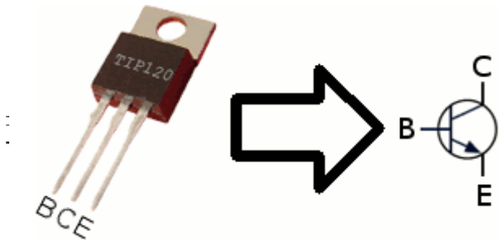
Le spazzole comportano una spesa in energia, infatti i motori dc assorbono una corrente che Arduino non è in grado di erogare dai suoi pin: per questo si usano uno o più componenti di collegamento fra il cervello e l'attuatore: un **transistor** oppure un **ponte H** .

Metodo 1: transistor

Il modo più semplice per far girare un motore DC è usare un transistor, se riesci a procurartene uno, usa un transistor darlington TIP120, altrimenti va bene un generico NPN.

Il transistor funge da interruttore elettronico: applicando la tensione di 5V sulla base tramite un pin digitale di Arduino, emettitore e collettore vengono "connessi", cortocircuitati, permettendo così il passaggio della corrente. Il motivo per cui è necessario un transistor che preleva la tensione da un connettore esterno è che può *sopportare* correnti di centinaia di milliAmpere, a differenza di un pin di Arduino che ne può erogare solamente alcune decine, insufficienti a far ruotare un motore. Questo metodo è *vantaggioso* perché richiede **un solo pin** di Arduino, ma può far girare il motore **in una sola direzione**.
Vediamo un primo circuito con un transistor e un pulsante: il programma manterrà il motore acceso finchè il pulsante è premuto.





Analizziamo i collegamenti: la **base** del transistor è collegata al pin 9 attraverso una resistenza, che nello schema vale 1kΩ; l'**emettitore** a massa; il **collettore** ad un piedino del motore; l'altro terminale del motore va connesso ad un'alimentazione esterna, come una batteria. Ai due terminali del motore è necessario inoltre collegare un diodo, con polarità contraria a quella della corrente che lo alimenta: serve per evitare cortocircuiti mentre il motore rallenta o in caso venga ruotato manualmente. Infine, colleghiamo un piedino del bottone al pin 2 di Arduino e l'altro piedino a massa.

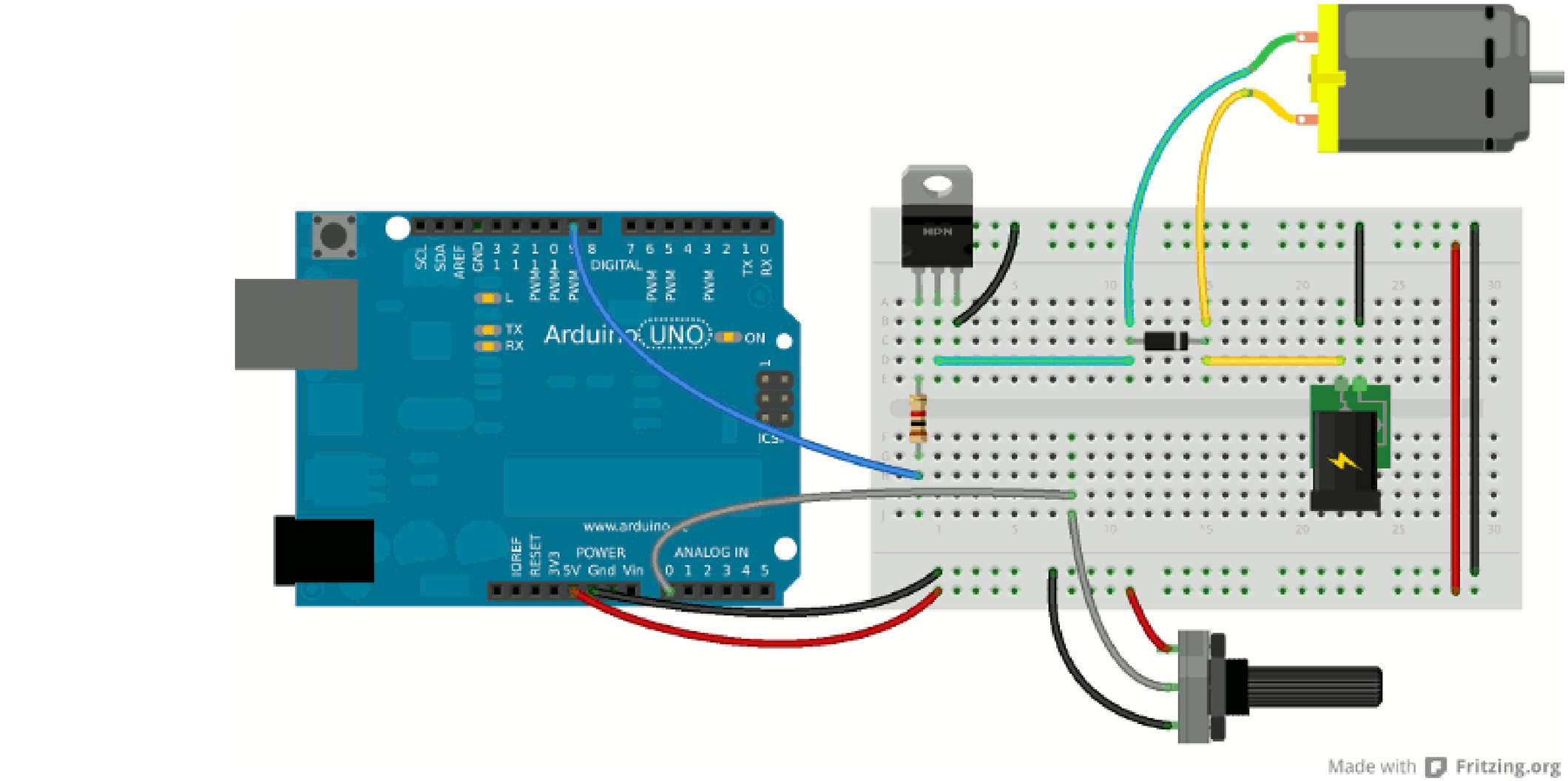
Questo è il listato di Arduino per il circuito sopra.

```
const int bottone = 2;      // pin del bottone
const int motore =  9;      // pin del motore

void setup() {
  // inizializza il bottone come input con pull-up
  // e il motore come output
  pinMode(motore, OUTPUT);
  pinMode(bottone, INPUT_PULLUP);
}

void loop(){
       // Il motore ha lo stesso stato del bottore,
       // ovvero se premi il bottone, il motore gira
       digitalWrite(motore, digitalRead(bottone));
 }
```

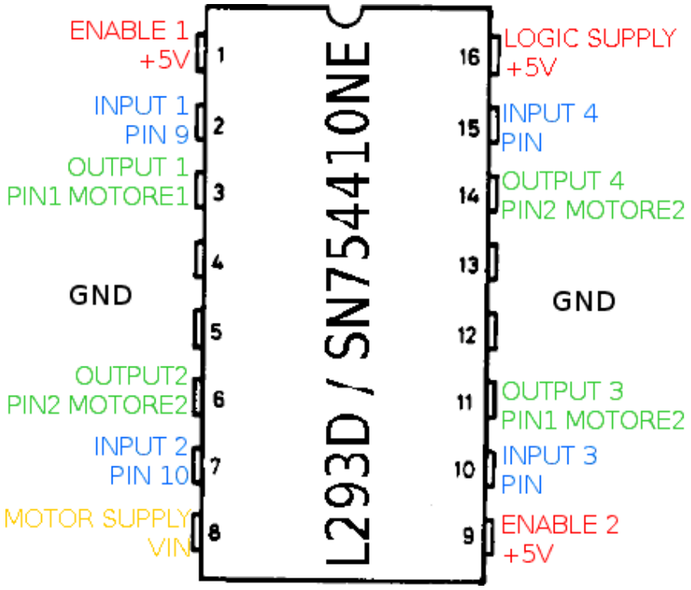
Un secondo esempio, mostra come è possibile modulare la velocità del motore grazie il **PWM** ruotando un potenziometro, collegato al pin analogico A0:

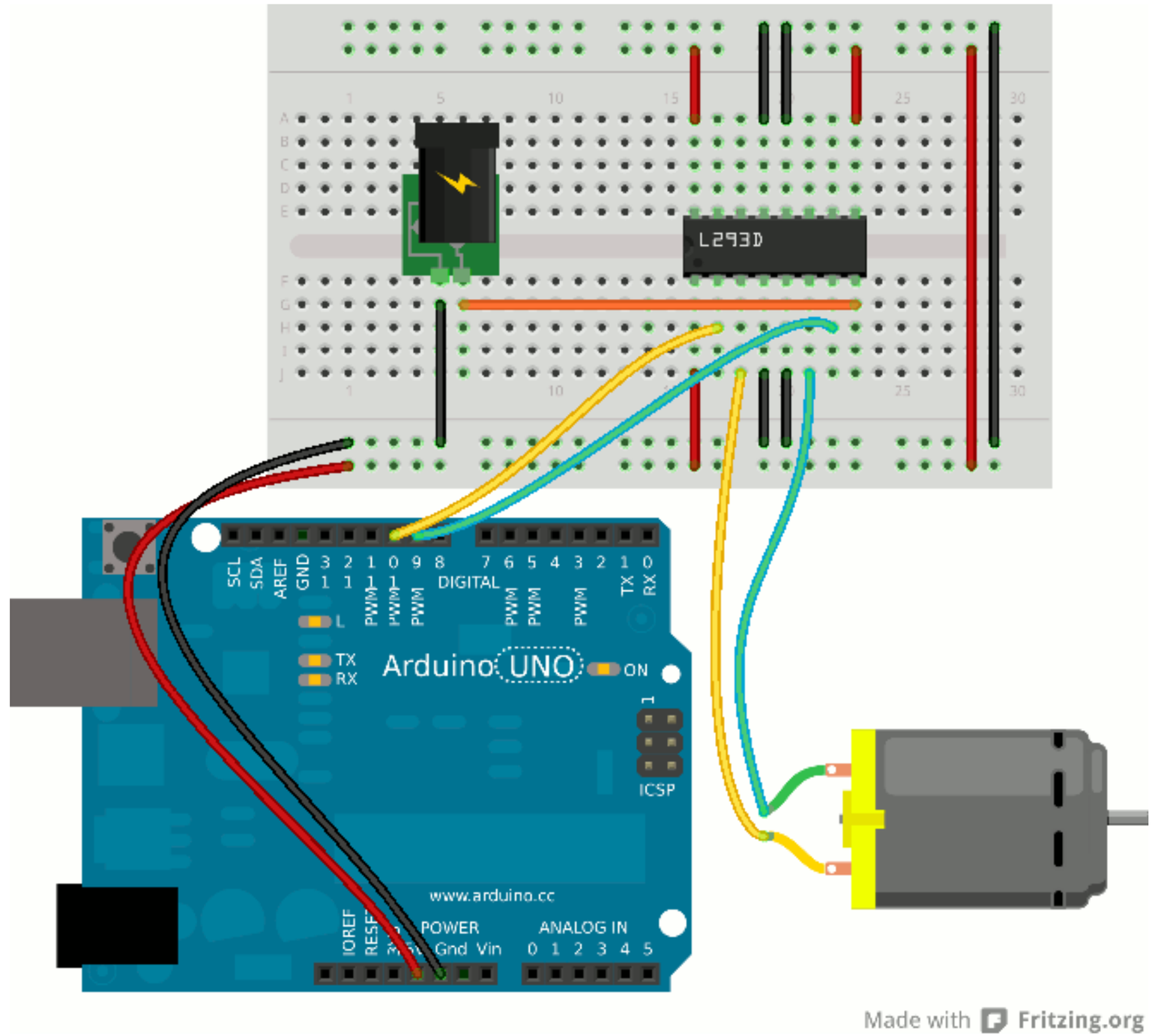



```
1. const int potenziometro = 0;    // pin del potenziometro
2. const int motore = 9;     // pin del motore
3.
4. void setup() {
5.  // inizializza il motore come output
6.  pinMode(motore, OUTPUT);
7. }
8.
9. void loop(){
10.       // Riporta il valore analogico da 0-1023 a 0-255
11.       byte valore = map(analogRead(potenziometro),0,1023,0,255);
12.       // Il motore gira con velocità proporzionale alla rotazione del potenziometro
13.       analogWrite(motore,valore);
14. }
```

Metodo 2: Ponte H

Il ponte H è un chip che contiene un po' di transistor di potenza e diodi di sicurezza, e consente di pilotare due motori DC (in entrambi i versi). In questo esempio si usa un **L293D**, ma il funzionamento è analogo per il SN754410NE, adatto per motori che richiedono più corrente.





Anche se i collegamenti dovrebbero essere chiari, mi soffermo su alcuni dettagli che ritengo importanti: l'alimentazione dei motori (motor supply, pin 8) non dovrebbe essere connessa ai 5V di Arduino perché, come spiegato prima, non eroga corrente a sufficienza. Va invece collegato al pin VIN, oppure ad una batteria esterna;
I pin di enable (1 e 9) servono per attivare i due motori: si presuppone che questi siano sempre in funzione, per cui puoi collegarli ai 5V; Infine, se vuoi usare solo un motore, basta omettere i pin che riguardano il motore2 (9-10-11-14-15);

Gli input controllano lo stato dei relativi output: il motore gira applicando segnali opposti sulla coppia di input, e negli altri casi il motore è fermo.

Input 1	Input 2	Motore
HIGH	LOW	Senso "orario"
LOW	HIGH	Senso "antiorario"
LOW	LOW	Stallo
HIGH	HIGH	Stallo

```
#define motorel 9
#define motorer 10

void setup() {
 // inizializza il bottone come input
 // e il motore come output
 pinMode(motorel, OUTPUT);
 pinMode(motorer, OUTPUT);
 pinMode(bottone, INPUT);
}

void loop(){
      // Tenendo premuto il bottone si inverte il senso di rotazione
      digitalWrite(motorel, digitalRead(bottone));
      digitalWrite(motorer, !digitalRead(bottone));
      delay(50);  // Una piccola pausa per evitare "rimbalzi di segnale"
}
```

Anche con il ponte H puoi utilizzare i piedini PWM per variare la velocità dei motori.