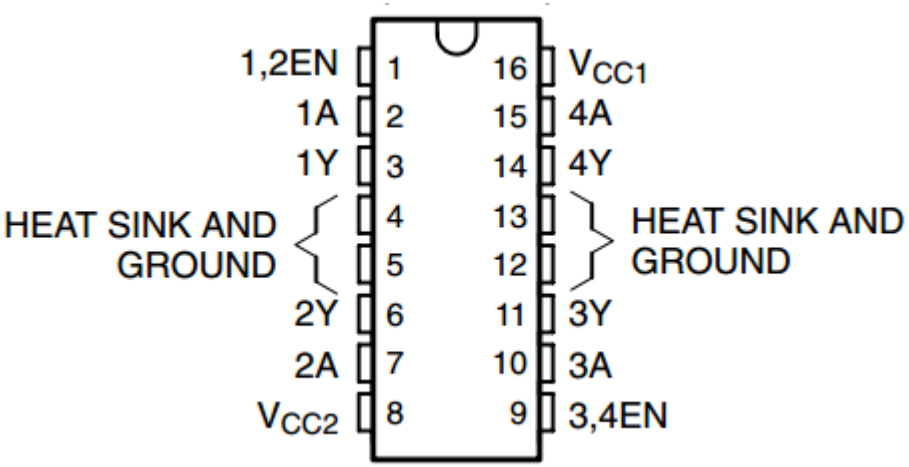
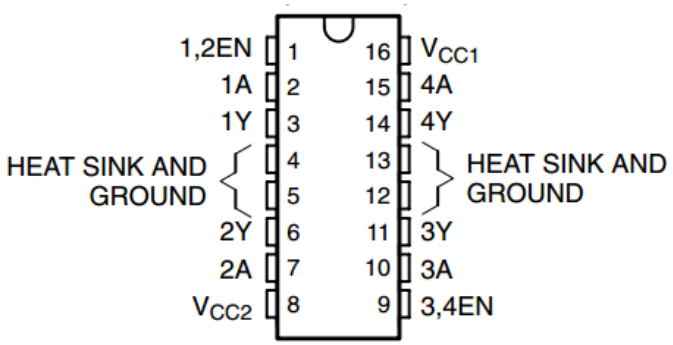
Controllo di velocità e verso di un motore DC con Arduino con L293NE

Per controllare un motore DC (funzionante in corrente continua) con Arduino, devi utilizzare il circuito integrato "Ponte H" (nel mio caso un L293NE della Texas Instruments o L293D o SN754410)
Il ponte H può, invertendo la polarità dell'uscita a cui è collegato il motore, variarne il verso di rotazione.



Controllo di velocità e verso di un motore DC con Arduino con L293NE

Nella tabella sottostante puoi vedere il comportamento del motore a seconda dei settaggi dei pin dell'integrato e lo schema del circuito.

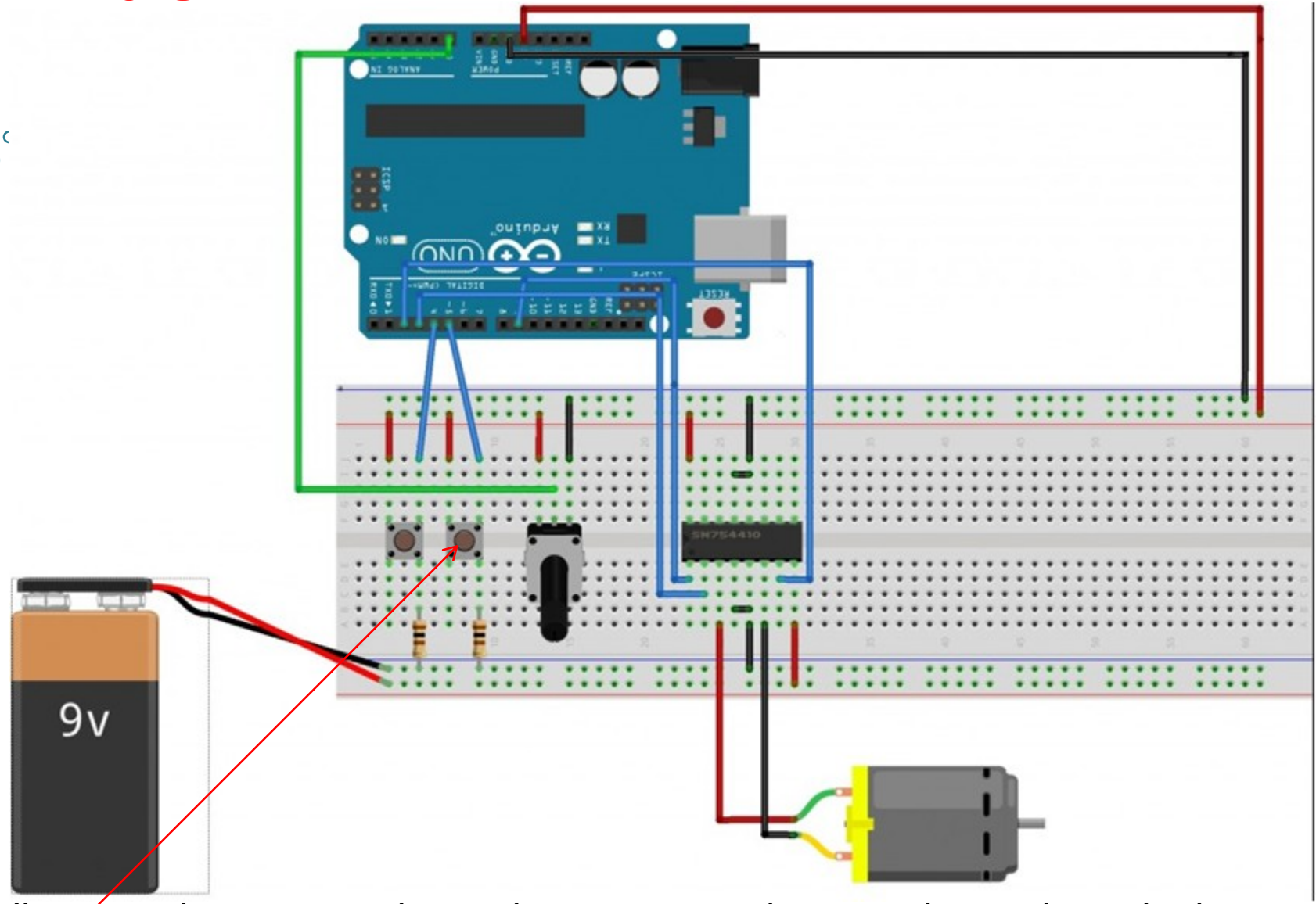


EN	1A	2A	FUNCTION
H	L	H	Turn right
H	H	L	Turn left
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Fast motor stop

L = low, H = high, X = don't care

Controllando l'EN pin del chip con un pin digitale di Arduino: settandolo HIGH o LOW accenderemo o spegneremo il motore.
Anche i pin logici del chip saranno connessi a dei pin digitali di Arduino, in modo da andare in una direzione con HIGH e LOW, nella direzione opposta con LOW e HIGH.
L'alimentazione del motore é comunemente esterna. Sei l motore che usate può andare a 5V e a meno di 500 mA potete usare i 5V di Arduino.
Questo é molto raro.
Se notate che il vostro microcontrollore va in reset quando il motore si accende, aggiungete in condensatore tra alimentazione e massa vicino al motore.
Il condensatore spianerà lo sbalzo elettrico che avviene quando il motore si accende.

I° SCHEMA



Il primo pulsante serve ad accendere o spegnere il motore, il secondo cambia la direzione di rotazione, mentre il potenziometro aumenta o diminuisce la velocità di rotazione.

SKETCH 1/4

Definiamo le costanti e le variabili poi recuperate nel loop().

```
const int controlPin1 = 2;                  //costanti di riferimento
const int controlPin2 = 3;                  //ai pin
const int accendiMotore = 9;
const int pulsanteDirezione = 4;
const int pulsanteOnOff = 5;
const int potenziometro = A0;

int statoPulsanteOnOff;                     //variabili di stato
int statoPulsanteOnOffPrecedente;
int statoPulsanteDirezione;
int statoPulsanteDirezionePrecedente;

int motoreAttivo;
int velocitaMotore;
int direzioneMotore = 1;
```

SKETCH 2/4

Nel setup() inizializziamo i pin dei pulsanti in INPUT e quelli collegati al ponte H in OUTPUT.

```
void setup() {
  pinMode(pulsanteDirezione, INPUT);          //inizializzazione della modalità dei
pin
  pinMode(pulsanteOnOff, INPUT);
  pinMode(controlPin1, OUTPUT);
  pinMode(controlPin2, OUTPUT);
  pinMode(pulsanteOnOff, OUTPUT);
                                              //imposto accendiMotore su LOW
  digitalWrite(accendiMotore, LOW);           //inizialmente il motore sarà spento

}
```

SKETCH 3/4

Nel loop() associo dei valori a delle variabili, con digitalRead().
Alla variabile velocitaMotore viene associata la lettura analogica di potenziometro, che genera un numero che va da 0 a 1023.
Siccome la funzione analogWrite() che verrà utilizzata più avanti (riga 55) vuole un numero compreso tra 0 e 255, il valore restituito da analogRead() è diviso per 4.

```
void loop() {
  statoPulsanteOnOff = digitalRead(pulsanteOnOff);
  delay(1);
  statoPulsanteDirezione = digitalRead(pulsanteDirezione);
  velocitaMotore = analogRead(potenziometro)/4;

  if (statoPulsanteOnOff != statoPulsanteOnOffPrecedente) {
    if (statoPulsanteOnOff == HIGH) {
      motoreAttivo =! motoreAttivo;
    }
  }
  if (statoPulsanteDirezione != statoPulsanteDirezionePrecedente) {
    if (statoPulsanteDirezione == HIGH) {
      direzioneMotore =! direzioneMotore;
    }
  }
```

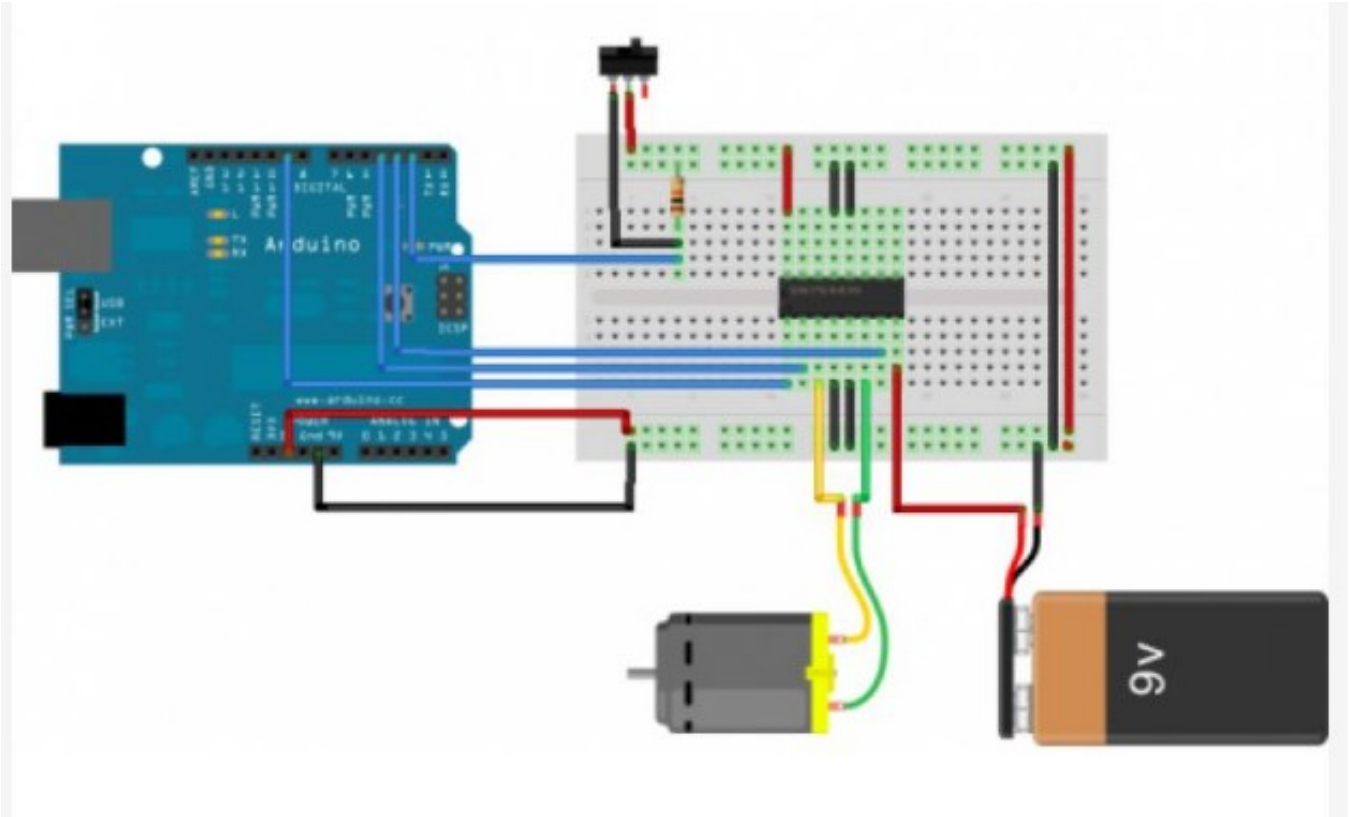
SKETCH 4/4

```
if (direzioneMotore == 1) {
    digitalWrite(controlPin1, HIGH);
    digitalWrite(controlPin2, LOW);
  }
  else {
    digitalWrite(controlPin1, LOW);
    digitalWrite(controlPin2, HIGH);
  }

  if (motoreAttivo == 1) {
    analogWrite(accendiMotore, velocitaMotore);
  }
  else {
    analogWrite(accendiMotore, 0);
  }

  statoPulsanteDirezionePrecedente = statoPulsanteDirezione;
  statoPulsanteOnOffPrecedente = statoPulsanteOnOff;
}
}
```

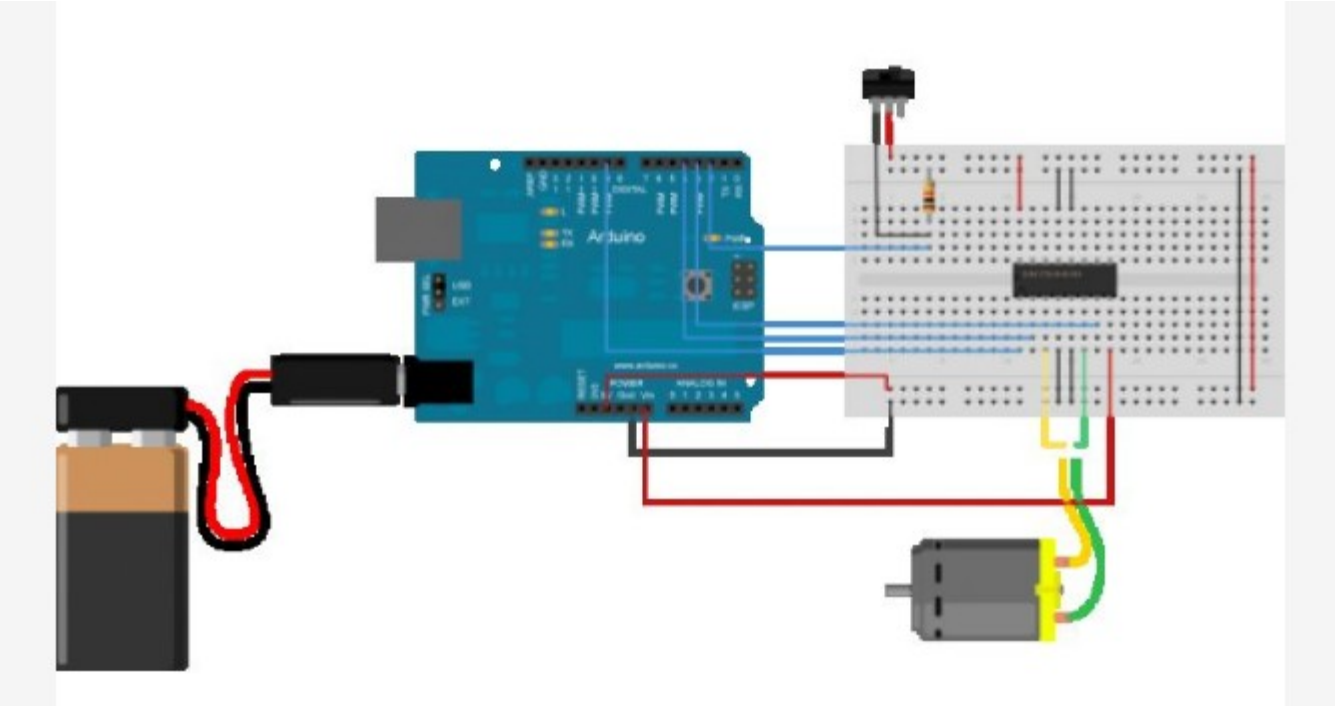
Sono poi presenti tre quattro if:
Se lo stato del pulsante di accensione del motore è diverso da quello precedente ed ha stato logico alto, lo stato logico del motore cambia (se prima era acceso ora è spento e viceversa)
Stesso ragionamento per il pulsante di cambio direzione
Se direzioneMotore corrisponde a 1, il primo pin di controllo ha stato logico alto, mentre il secondo ha stato logico basso.
Altrimenti viceversa
Se il motore è attivo viene eseguita la funzione analogWrite() sul pin accendiMotore (9) ed associato il valore di velocitaMotore. In questo modo, grazie ad una modulazione in PWM, la velocità cambia
Infine associamo alle variabili in "-Precedente" le corrispondenti dell'"attuale" loop(). In questo modo nel successivo loop() possono essere richiamate.

Guarda il video https://www.youtube.com/watch?v=iYXwSQWGLH0

2° SCHEMA



Con Alimentazione esterna



Codice

```
const int switchPin = 2;     // switch input
const int motor1Pin = 3;     // H-bridge leg 1 (pin 2, 1A)
const int motor2Pin = 4;     // H-bridge leg 2 (pin 7, 2A)
const int enablePin = 9;     // H-bridge enable pin
const int ledPin = 13;       // LED

void setup() {
  // set the switch as an input:
  pinMode(switchPin, INPUT);

  // set all the other pins you're using as outputs:
  pinMode(motor1Pin, OUTPUT);
  pinMode(motor2Pin, OUTPUT);
  pinMode(enablePin, OUTPUT);
  pinMode(ledPin, OUTPUT);

  // set enablePin high so that motor can turn on:
  digitalWrite(enablePin, HIGH);

  // blink the LED 3 times. This should happen only once.
  // if you see the LED blink three times, it means that the module
  // reset itself,. probably because the motor caused a brownout
  // or a short.
```

Codice

```
  blink(ledPin, 3, 100);
}
 void loop() {
  // if the switch is high, motor will turn on one direction:
  if (digitalRead(switchPin) == HIGH) {
    digitalWrite(motor1Pin, LOW);   // set leg 1 of the H-bridge low
    digitalWrite(motor2Pin, HIGH);  // set leg 2 of the H-bridge high
  }
  // if the switch is low, motor will turn in the other direction:
  else {
    digitalWrite(motor1Pin, HIGH);  // set leg 1 of the H-bridge high
    digitalWrite(motor2Pin, LOW);   // set leg 2 of the H-bridge low
          }
}

/*  blinks an LED   */
void blink(int whatPin, int howManyTimes, int milliSecs) {
  int i = 0;
  for ( i = 0; i < howManyTimes; i++) {
    digitalWrite(whatPin, HIGH);
    delay(milliSecs/2);
    digitalWrite(whatPin, LOW);
    delay(milliSecs/2);
  }
}
```

Codice test motori via seriale

```
// testing motors
const int motor1Pin = 3;    // H-bridge leg 1 (pin 2, 1A)
const int motor2Pin = 4;    // H-bridge leg 2 (pin 7, 2A)
const int enablePin = 9;    // H-bridge enable pin
const int ledPin = 13;      // LED

int incomingByte;     // a variable to read incoming serial data into

void setup()
 {
  Serial.begin(9600); // open serial port to receive data
  // set all the other pins you're using as outputs:
  pinMode(motor1Pin, OUTPUT);
  pinMode(motor2Pin, OUTPUT);
  pinMode(enablePin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  // set enablePin high so that motor can turn on:
  digitalWrite(enablePin, LOW);
 }
```

Codice test motori via seriale

```
void loop()
{
 // see if there's incoming serial data:
 if (Serial.available() > 0) {
   // read the oldest byte in the serial buffer:
   incomingByte = Serial.read();
   // if it's a capital H (ASCII 72), turn on the LED:
   if (incomingByte == 'H') {
     digitalWrite(enablePin, HIGH);   // accende  il motore
     digitalWrite(motor1Pin, LOW);   // set leg 1 of the H-bridge low
     digitalWrite(motor2Pin, HIGH);   // set leg 2 of the H-bridge high
   }
   else if (incomingByte == 'J') {
     digitalWrite(enablePin, HIGH);    // accende il motore
     digitalWrite(motor1Pin, HIGH);   // set leg 1 of the H-bridge low
     digitalWrite(motor2Pin, LOW);    // set leg 2 of the H-bridge high
   }
   else if (incomingByte == 'L') {
     digitalWrite(enablePin, LOW);     // spegne il motore
   }
  }
}
```

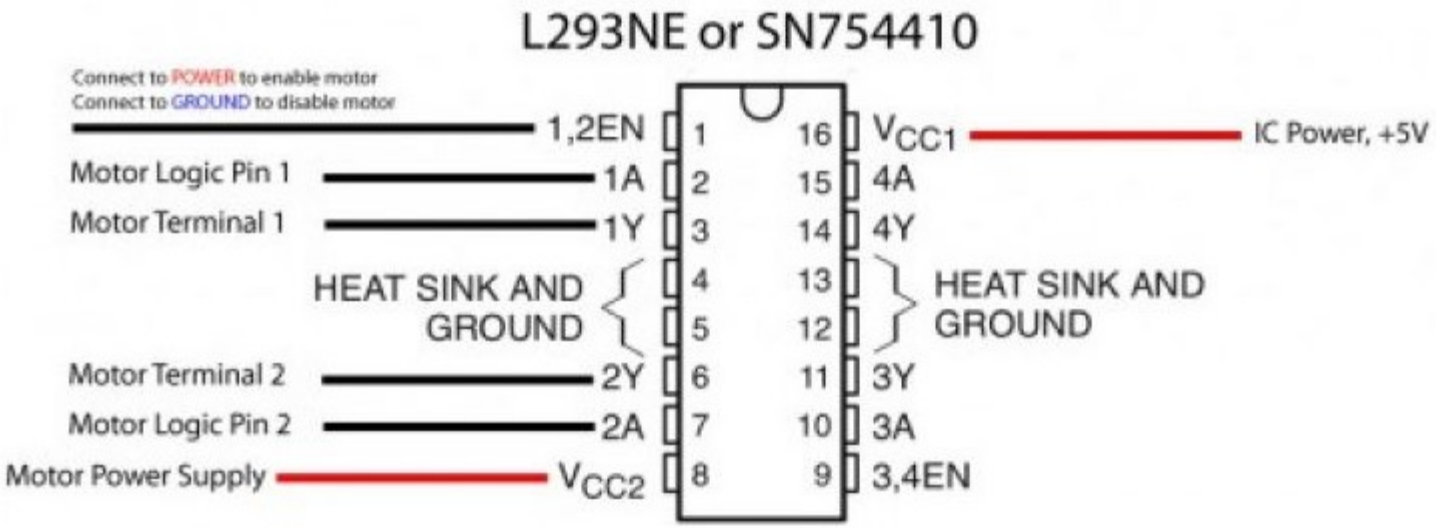
COME FUNZIONA UN PONTE H

L‘ L293NE/SN754410 é un ponte-H molto semplice. E' costituito da due "ponti", uno sul lato sinistro del chip ed uno sul lato destro, e può controllare 2 motori. Può controllare fino ad 1 ampere di corrente, e funziona tra i 4.5 V e i 36 V.
I pin del ponte H:

Pin 1 (1,2EN) attiva/disattiva il nostro motore a seconda che sia HIGH o LOW
Pin 2 (1A) é un pin logico per il nostro motore (ppuò essere HIGH o LOW)
Pin 3 (1Y) é uno dei due poin a cui metteremo il motore (motor terminals)
Pin 4-5 sono per la massa (GND)
Pin 6 (2Y) é per l'altro motor terminal
Pin 7 (2A)é un pin logico per il nostro motore (ppuò essere HIGH o LOW)
Pin 8 (VCC2) é l'alimentazione per il nostor motore, a cui dovremmo dare il voltaggio giusto per il nostro motore
Pin 9-11 non sono utilizzati (stiamo controllando solo un motore in questa lezione)
Pin 12-13sono per la massa (GND)
Pin 14-15 non sono connessi
Pin 16 (VCC1) é connesso a 5V

COME FUNZIONA UN PONTE H

Qui sotto c'é un diagramma sui pin che useremo e su come li useremo nel nostro esempio. Nella tabella sottostante potete vedere il comportamento del motore seconda si come vengono settati i pin dell'integrato (controllati da Arduino).



EN	1A	2A	FUNCTION
H	L	H	Turn right
H	H	L	Turn left
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Fast motor stop

L = low, H = high, X = don't care