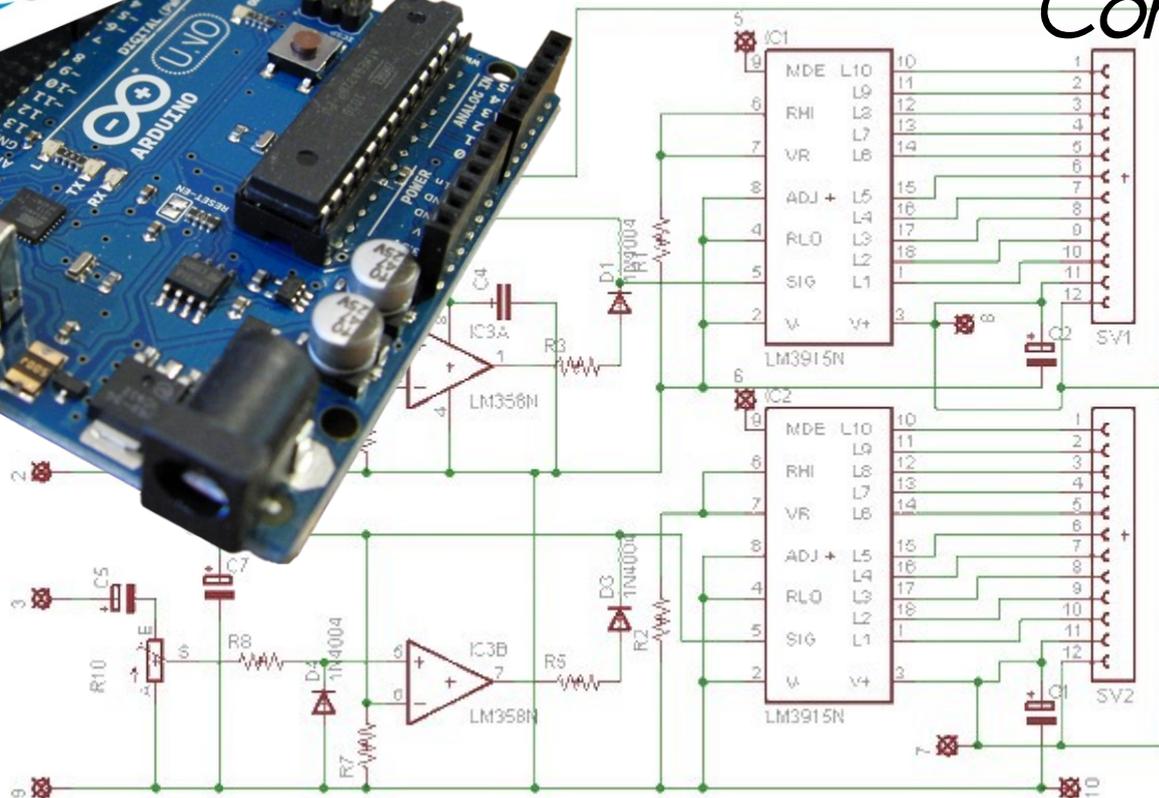
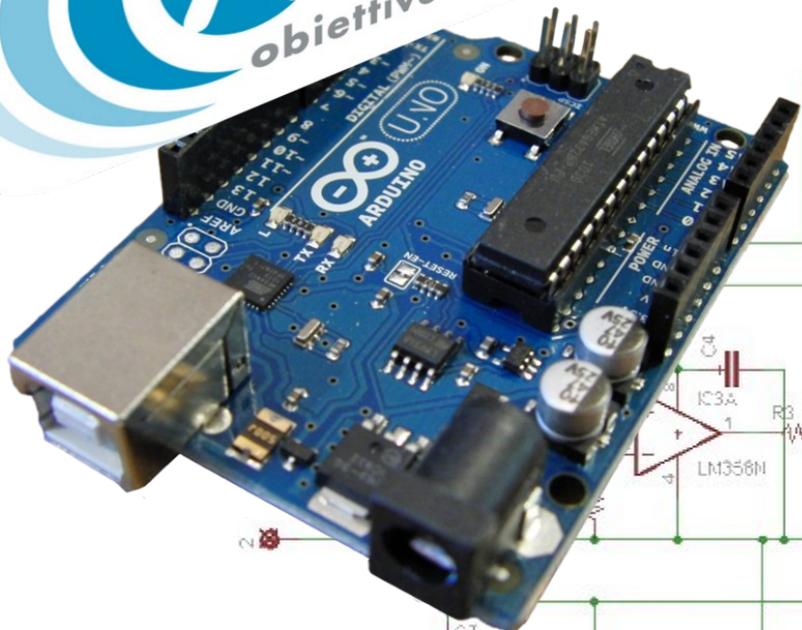


CORSO ARDUINO

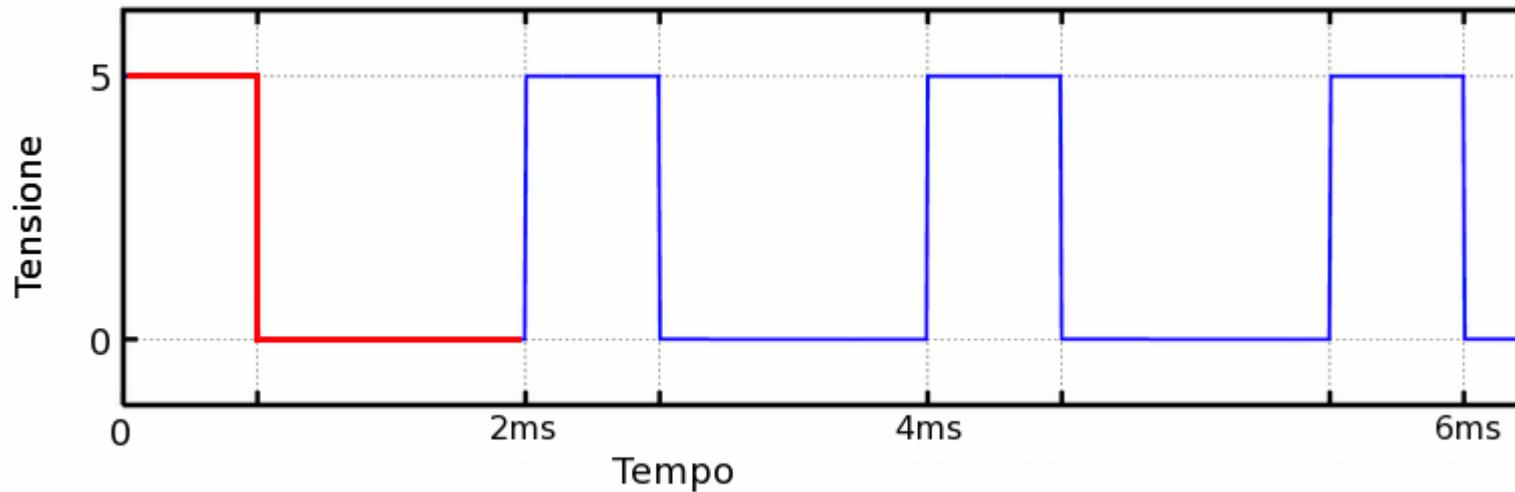


Giulio Fieramosca
Stefano Panichi
Corso ASEV 2015



PWM

modulazione a larghezza d'impulso



Formule e Dati

$$T_{on} = 1/3 \text{ del periodo}$$

$$T_{off} = 2/3 \text{ del periodo}$$

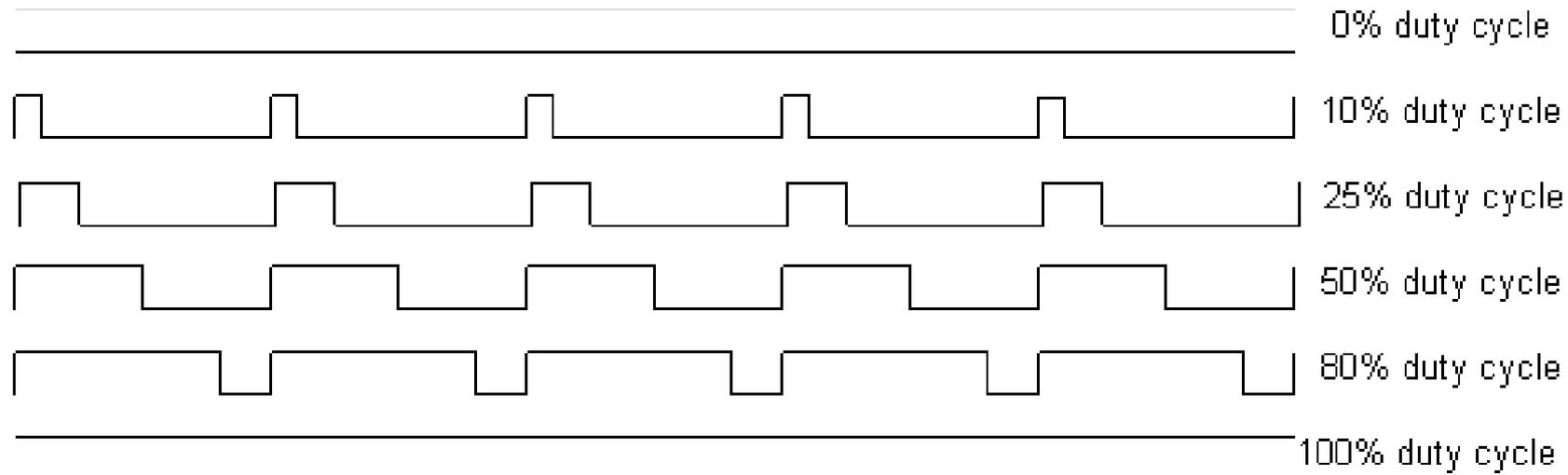
$$\text{Duty Cycle} = T_{on} / \text{periodo}$$

$$\text{Periodo} = T_{on} + T_{off} = 1/\text{Frequenza}$$

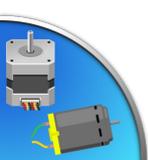
$$V_{media} = 5V * \text{Duty Cycle}$$



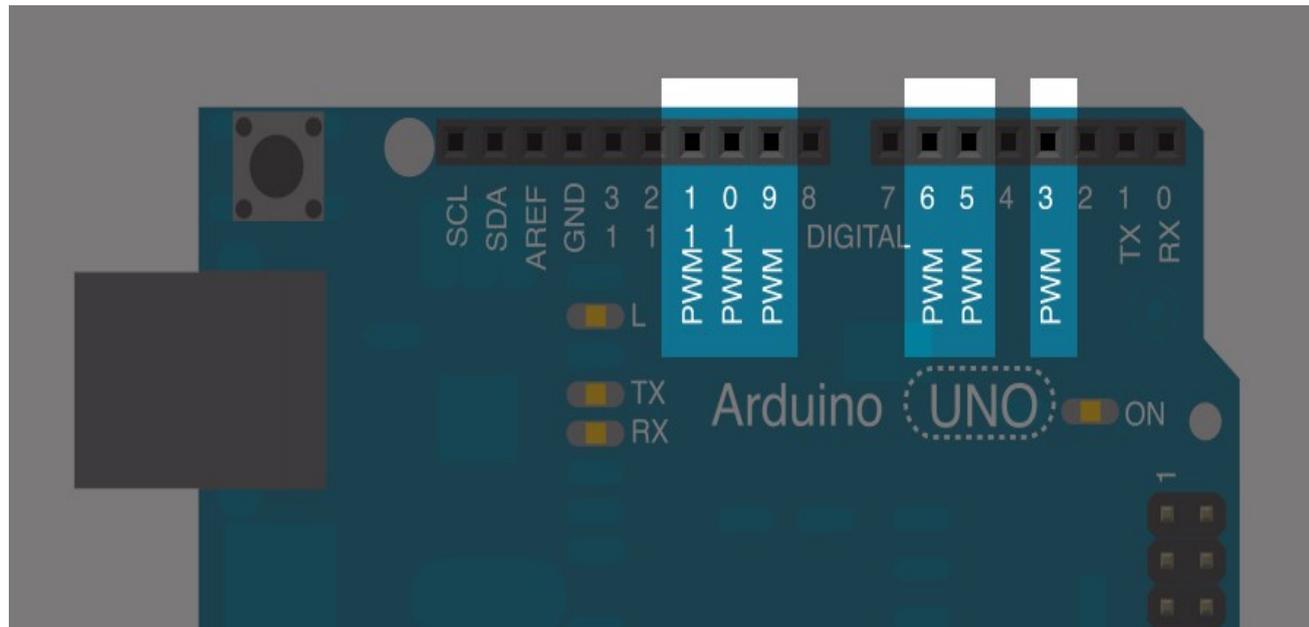
Utilizzi del PWM



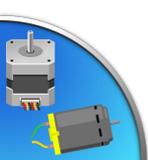
- Regolazione luminosità dei led *“Dimming”*;
- Regolazione velocità motori;
- Pilotaggio di *servomotori* analogici;



Pin di Arduino



- Arduino dispone di 6 piedini digitali dedicati al PWM.
- Ciò consente di gestire i segnali in modo autonomo (*asincrono*) rispetto al programma.

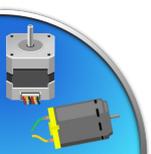
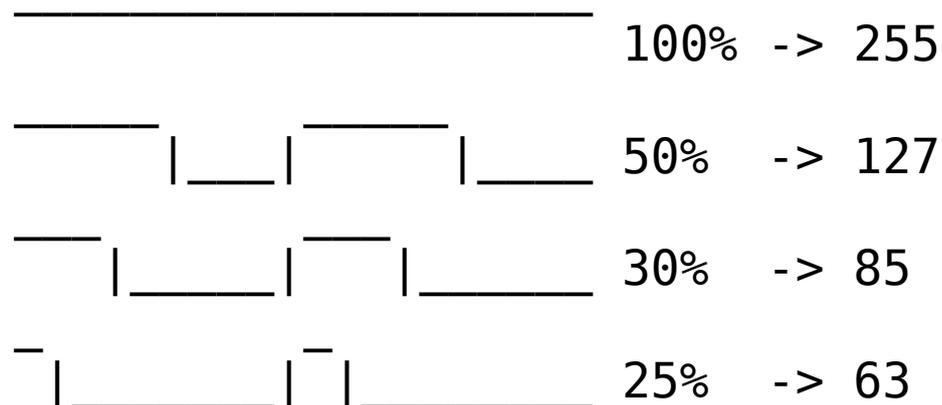


Programmazione del PWM

La funzione per mandare in uscita il PWM con il duty cycle desiderato è

```
analogWrite(pin, duty);
```

Il pwm è scalato a 8 bit, ovvero valori *duty* compresi fra 0 e 255 corrispondono ai duty cycle 0 – 100%

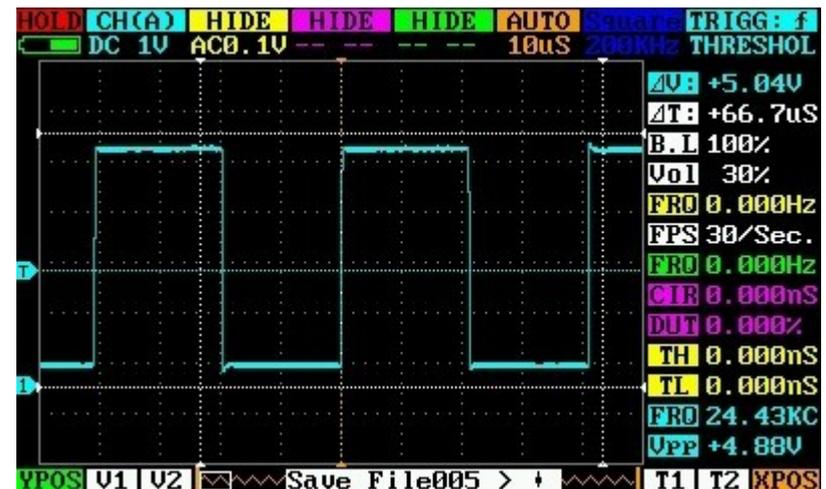


Frequenze di PWM

La frequenza (e quindi il periodo) del PWM è fissata a valori predefiniti:

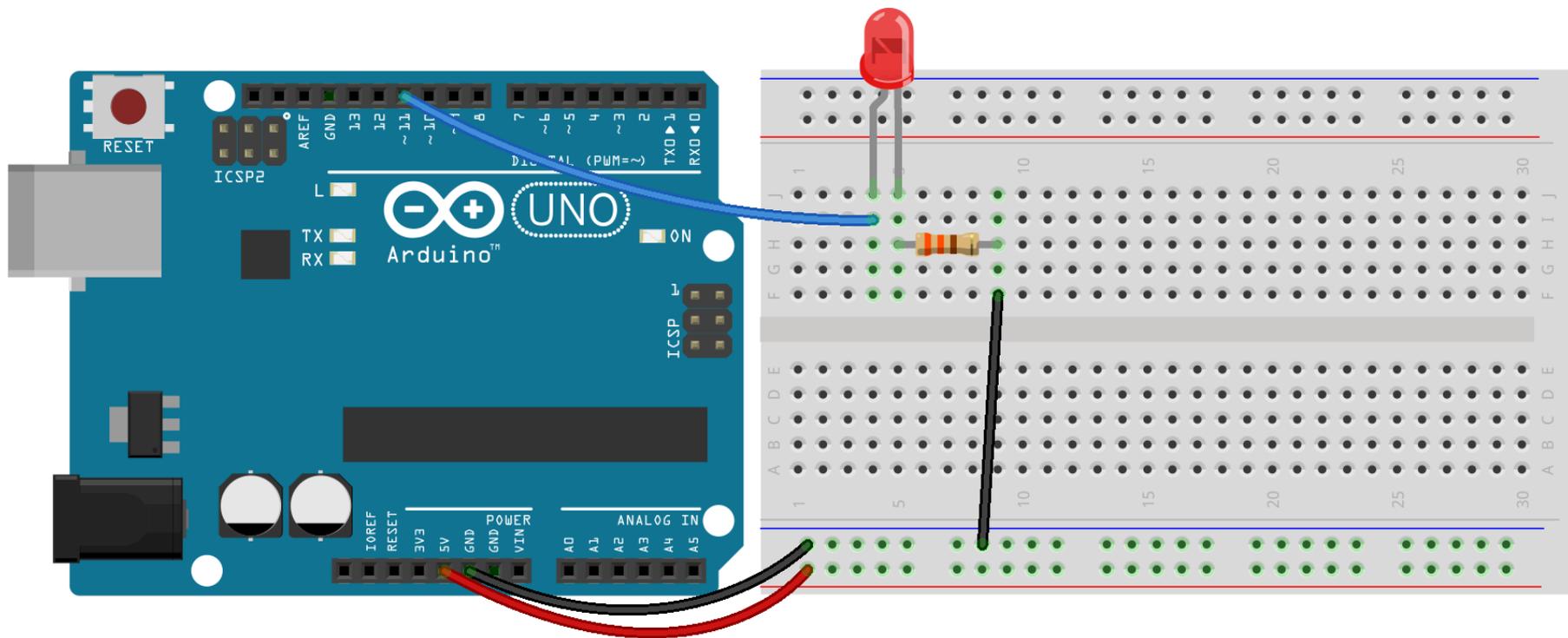
- 490 Hz per i pin 3, 9, 10, 11;
- 980 Hz per i pin 5, 6;

È possibile variarle, ma comporterebbe problemi sulle funzioni di temporizzazione (millis, delay, eccetera)

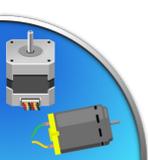


Vista all'oscilloscopio digitale

Collegamenti: Led Dimmer



fritzing



Sketch: Led Dimmer

```
const byte PWMpin = 11;

void setup() {
  pinMode(PWMpin, OUTPUT);
}

void loop() {
  // Accensione
  for (byte dim = 0; dim < 255; dim++) {
    analogWrite(PWMpin, dim);
    delay(10);
  }

  // Spegnimento
  for (byte dim = 255; dim > 0; dim--) {
    analogWrite(PWMpin, dim);
    delay(10);
  }
}
```



Motori

Troviamo sul mercato tre categorie di motori

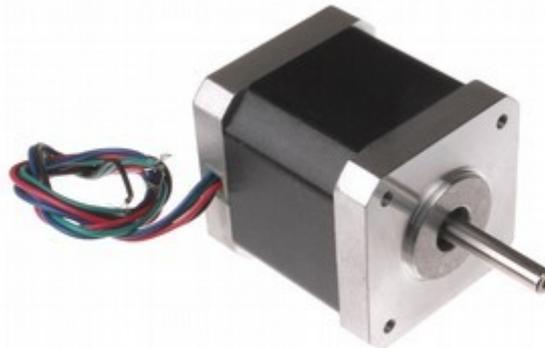
Motori DC:

2 fili



Motori stepper:

4~6 fili

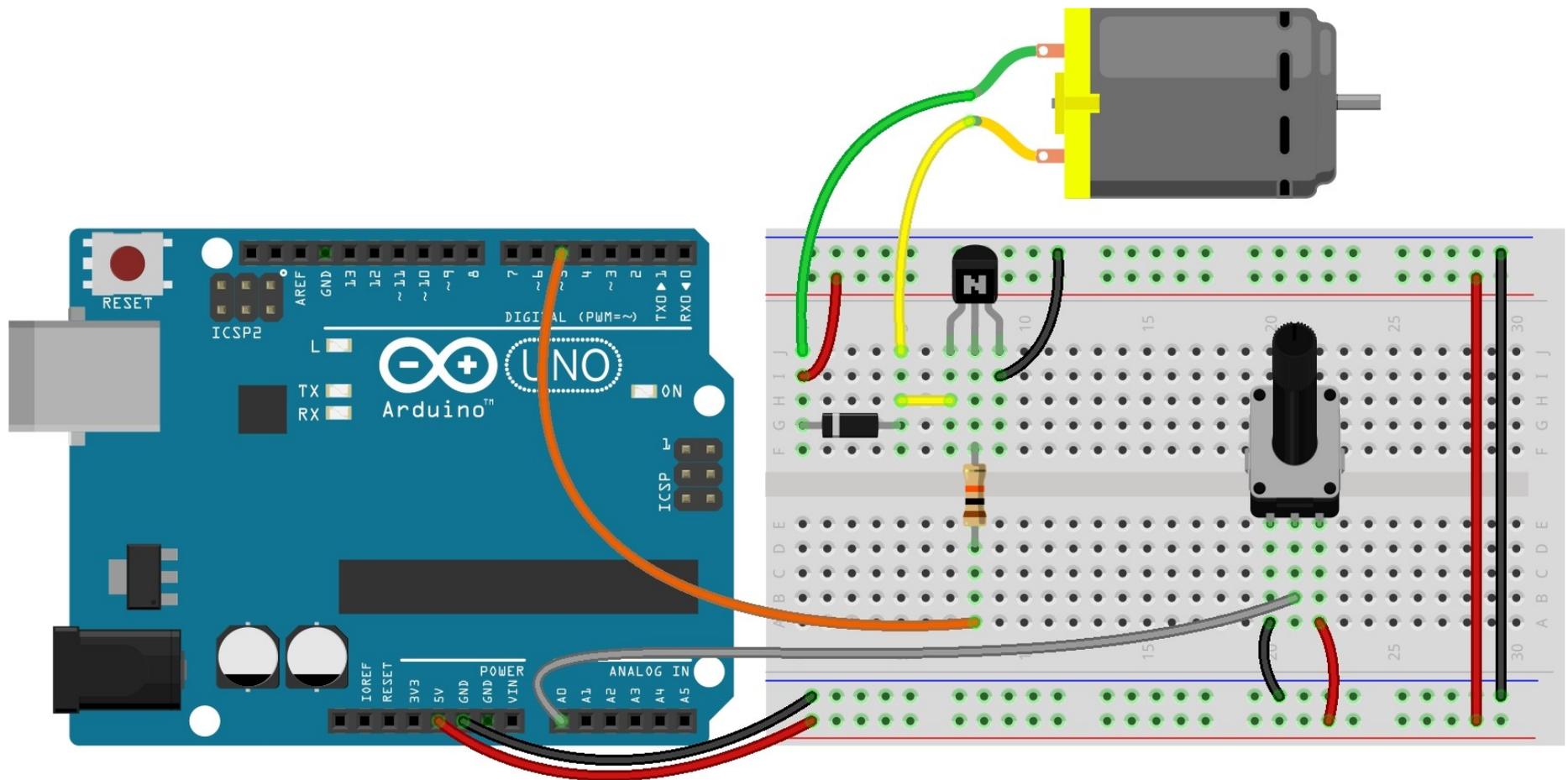


Servomotori:

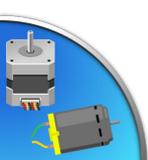
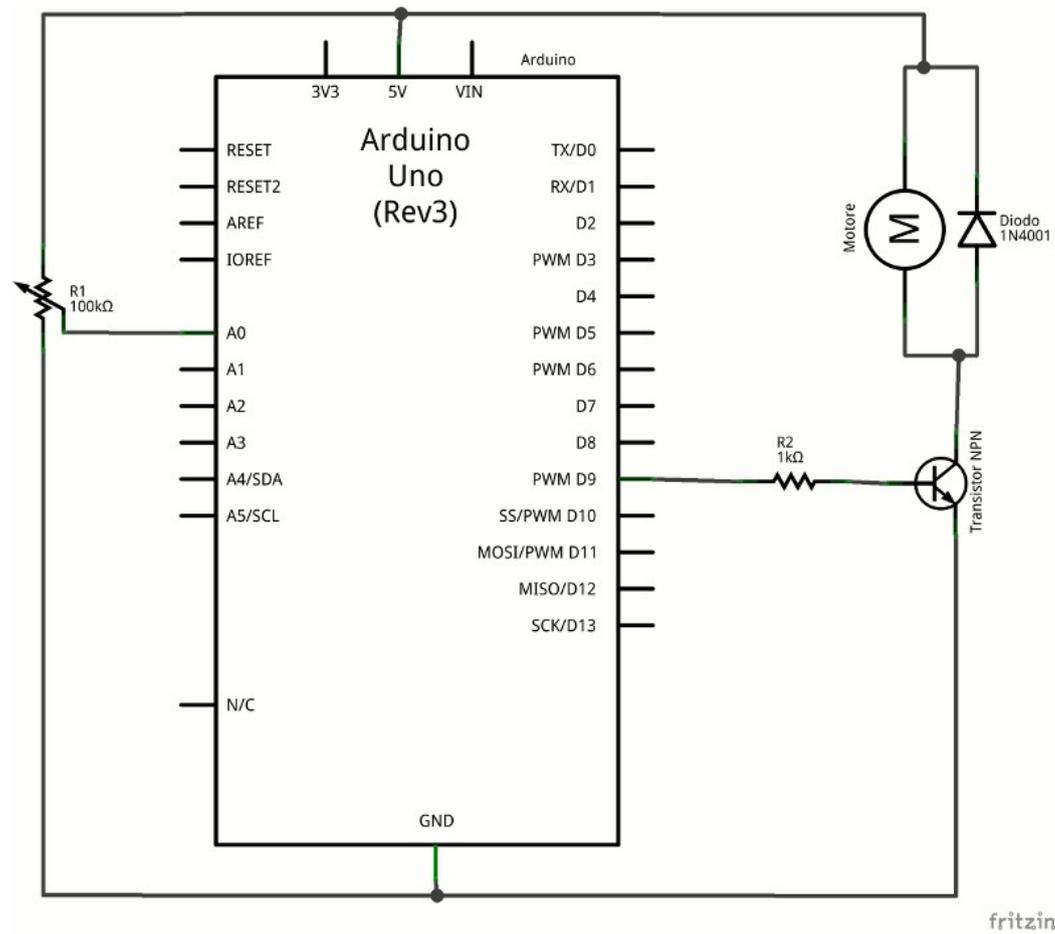
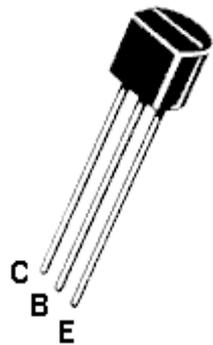
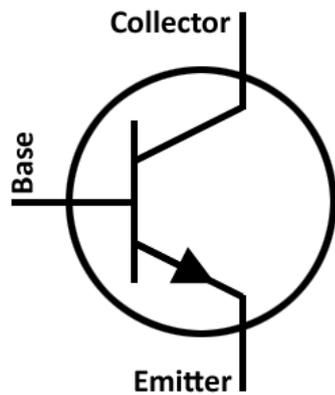
3 fili



Motore DC: metodo "transistor"



Motore DC: metodo "transistor"

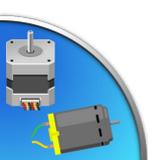


Listato Motore DC: metodo “transistor”

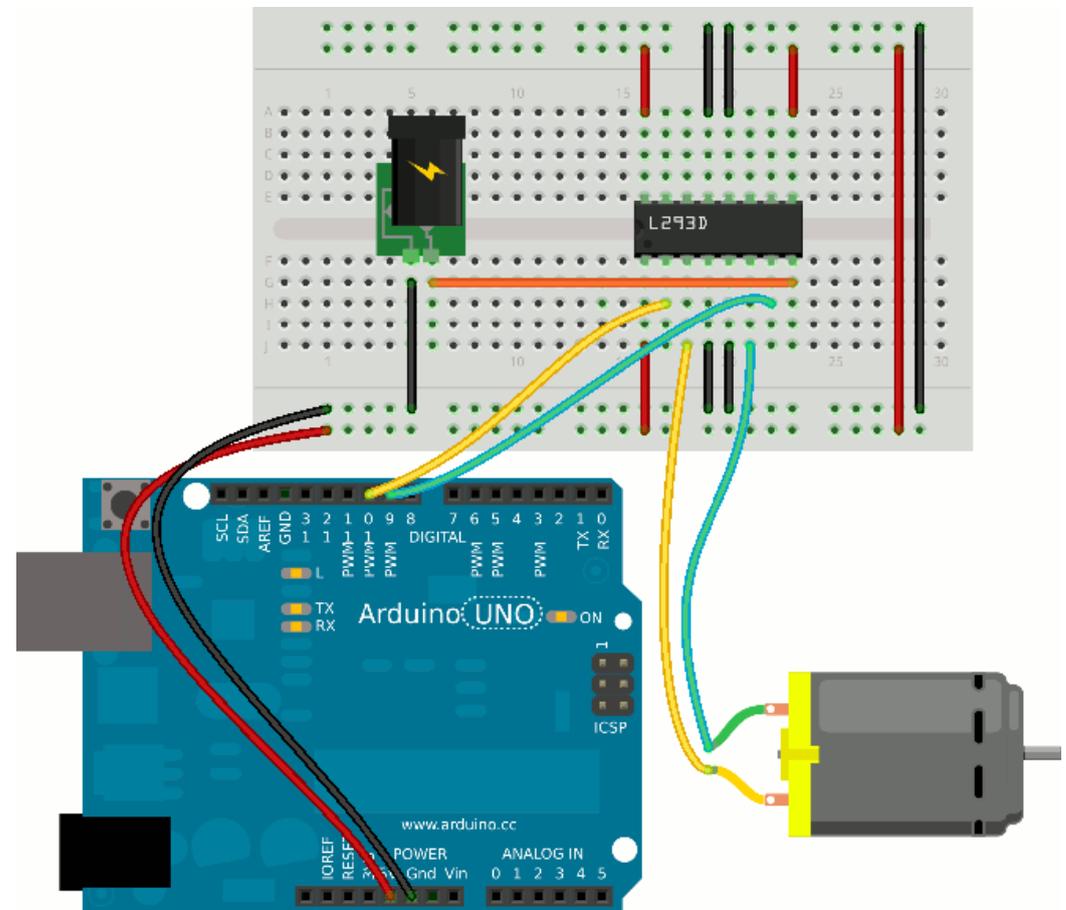
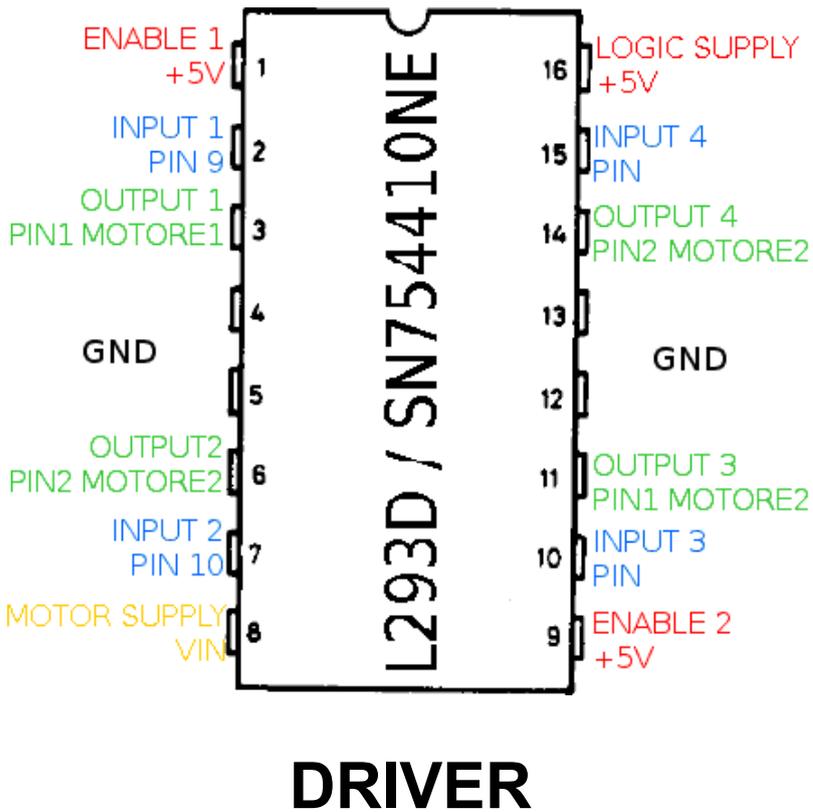
```
const byte POTENZ = A0; // potenziometro
const byte MOTORE = 9; // motore
```

```
void setup() {
    // inizializza il motore come output
    pinMode(MOTORE, OUTPUT);
}
```

```
void loop(){
    byte valore = map(analogRead(POTENZ), 0, 1023, 0, 255);
    // il motore gira con velocità proporzionale alla
    // rotazione del potenziometro
    analogWrite(MOTORE, valore);
    delay(10);
}
```

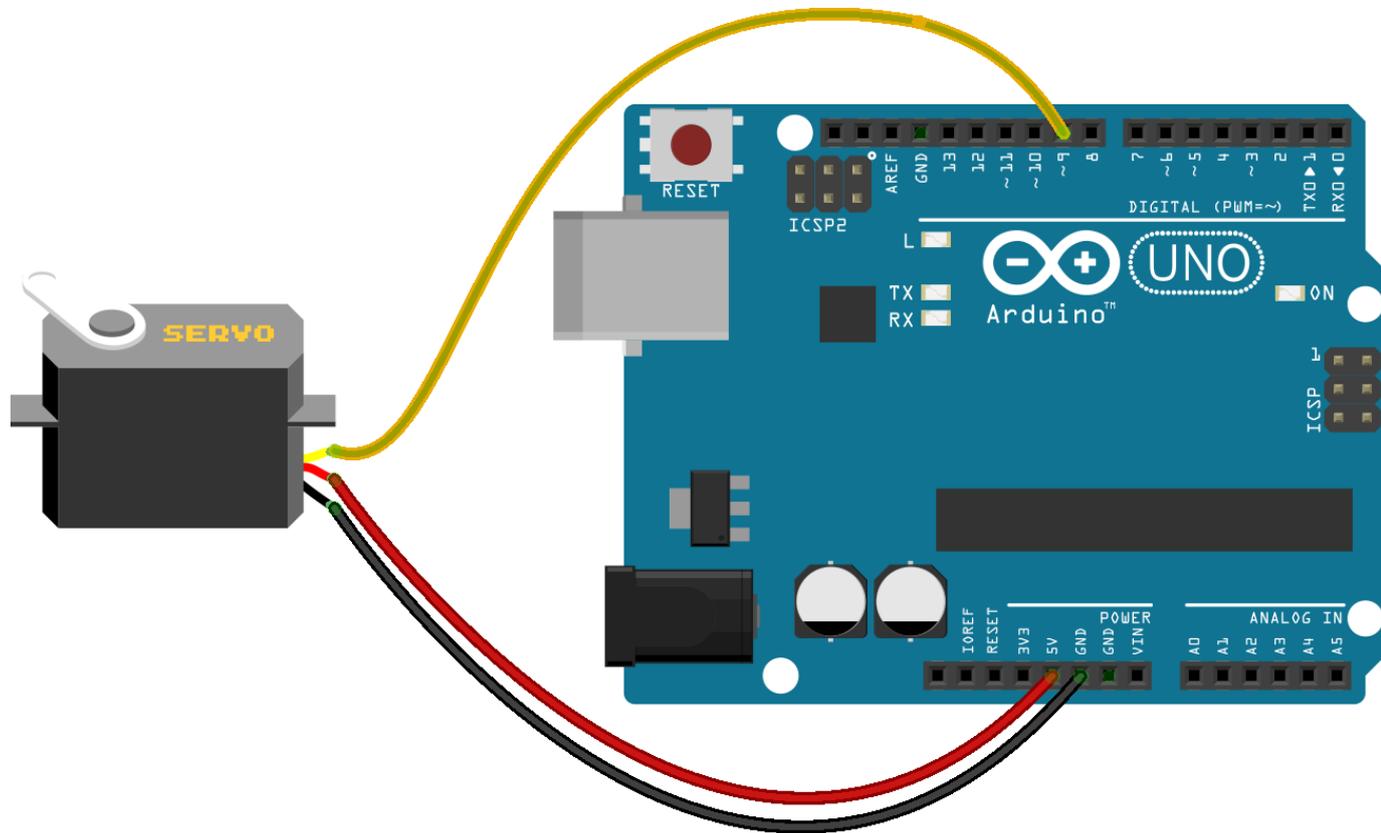


Motore DC: metodo "ponte H"



Made with Fritzing.org

Motore Servo

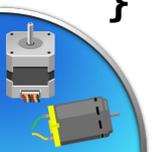


Motore Servo: listato

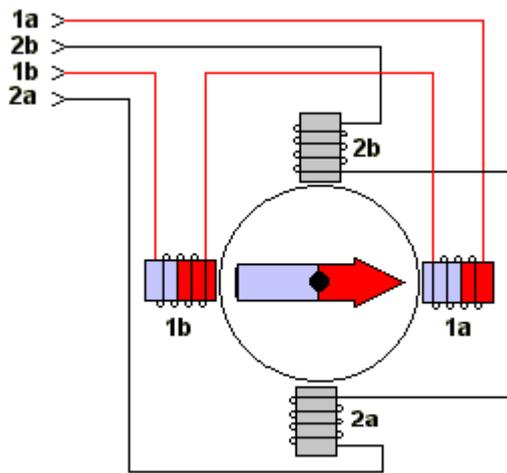
```
#include <Servo.h>    // Libreria per i servo
Servo myservo;       // crea un oggetto Servo (myservo)

void setup()
{
  myservo.attach(9); // setta il pin 9 al servo
}

void loop()
{
  for (byte passi = 0; passi < 180; passi++) {
    myservo.write(passi);           // Muove il servo a zero
    delay(10);
  }
  delay(1000); // Pausa
  for (byte passi = 180; passi > 0; passi--) {
    myservo.write(passi);           // Muove il servo a zero
    delay(10);
  }
  delay(1000); // Pausa
}
```

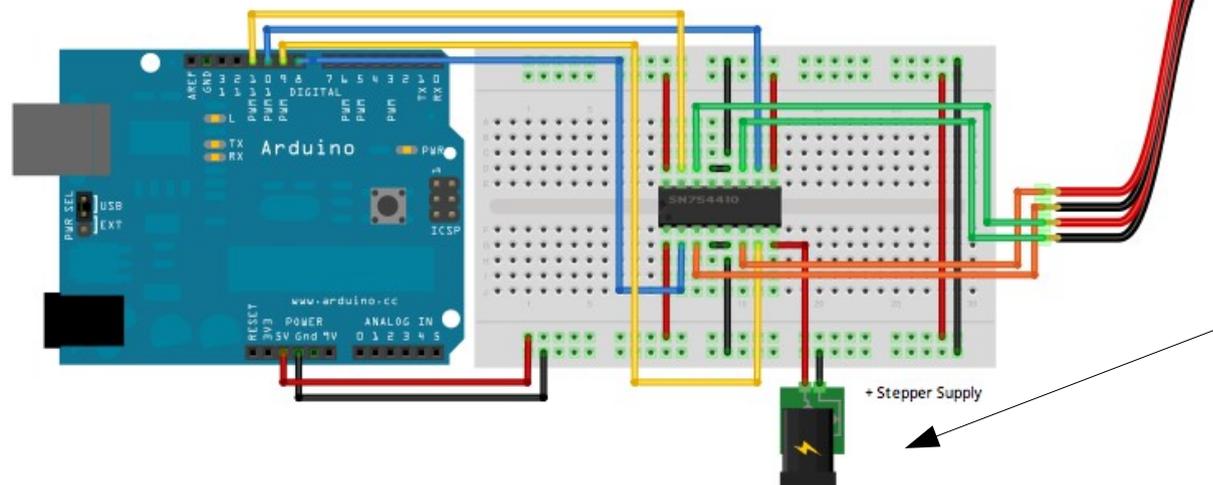
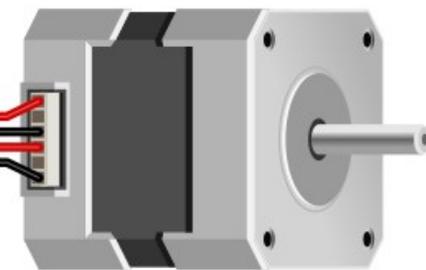
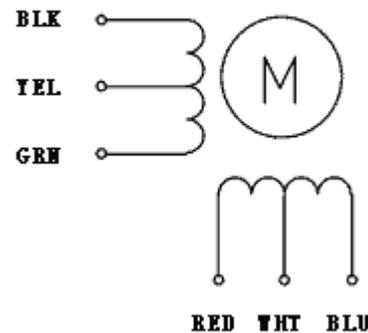


Pilotare uno Stepper bipolare



Conceptual Model of Bipolar Stepper Motor

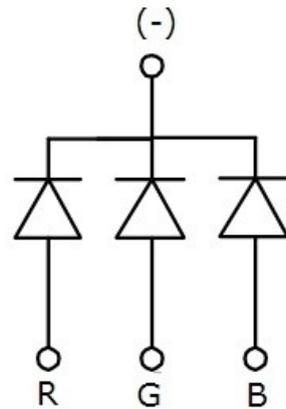
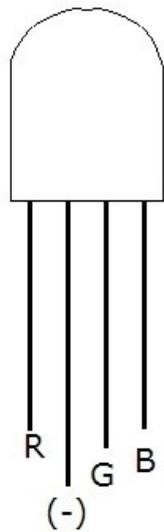
6 LEADS



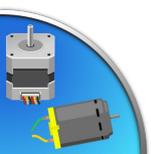
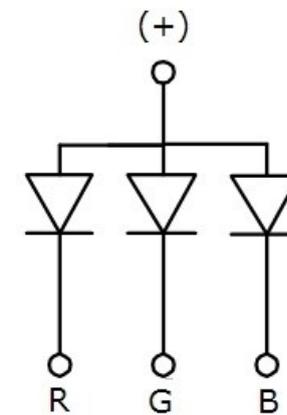
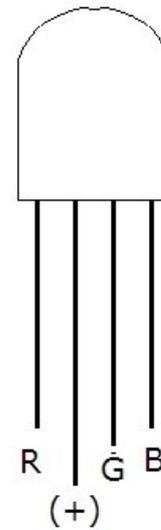
Alimentazione esterna:
Gli stepper funzionano perlopiù a 12~24 V

Approfondimento: LED RGB

Common Cathode (-)

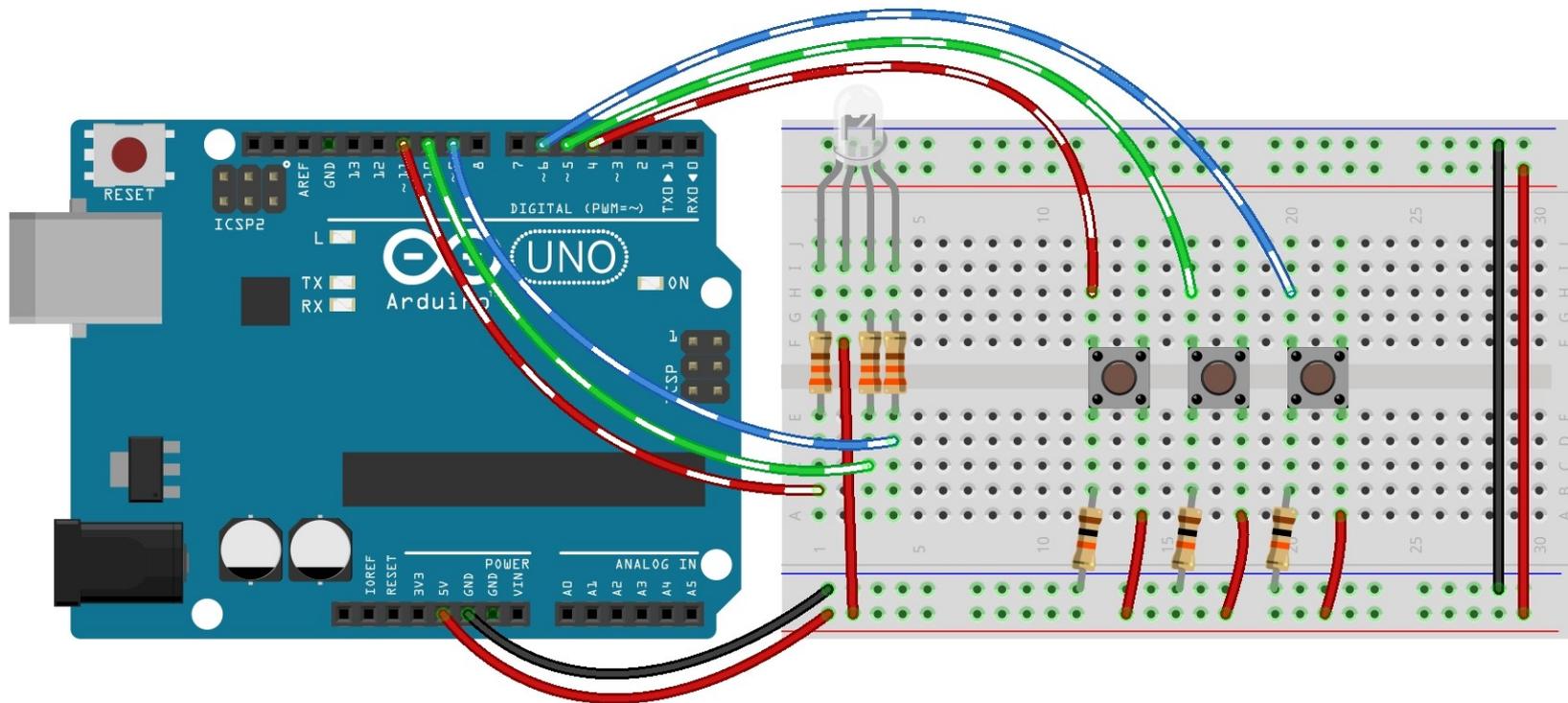


Common Anode (+)



Approfondimento: LED RGB

Setting Up



fritzing

Approfondimento: LED RGB

Sketch – variabili globali

```
const byte redLED = 11;  
const byte greenLED = 10;  
const byte blueLED = 9;
```

```
const byte redBUTTON = 6;  
const byte greenBUTTON = 5;  
const byte blueBUTTON = 4;
```

```
byte redColor = 0;  
byte greenColor = 0;  
byte blueColor = 0;
```

Approfondimento: LED RGB

Sketch – setup ()

```
void setup () {  
  pinMode (redLED , OUTPUT) ;  
  pinMode (greenLED, OUTPUT) ;  
  pinMode (blueLED, OUTPUT) ;  
  
  pinMode (redBUTTON, INPUT) ;  
  pinMode (greenBUTTON, INPUT) ;  
  pinMode (blueBUTTON , INPUT) ;  
  
  analogWrite (redLED, redColor) ;  
  analogWrite (greenLED, greenColor) ;  
  analogWrite (blueLED, blueColor) ;  
}
```

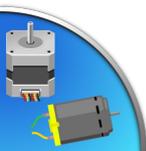
Approfondimento: LED RGB

Sketch – loop ()

```
void loop() {  
  if (digitalRead(redBUTTON) == HIGH) {  
    if (redColor < 255)  
      redColor++;  
    else  
      redColor = 0;  
  
    analogWrite(redLED, redColor);  
  }  
  
  if (digitalRead(greenBUTTON) == HIGH) {  
    if (greenColor < 255)  
      greenColor++;  
    else  
      greenColor = 0;  
  
    analogWrite(greenLED, greenColor);  
  }  
  
  if (digitalRead(blueBUTTON) == HIGH) {  
    if (blueColor < 255)  
      blueColor++;  
    else  
      blueColor = 0;  
  
    analogWrite(blueLED, blueColor);  
  }  
  
  delay(10);  
}
```

Compiti per casa - I

- **Crepuscolare proporzionale:** riprendi l'esercizio del crepuscolare della lezione precedente, e accendi il led con una luminosità inversamente proporzionale alla luce letta;
- **Arcobaleno:** realizza un programma che replichi i colori della scala cromatica con un LED RGB, in modo continuo;
- **Lampada colorata regolabile:** crea una piccola lampada, usando un led RGB, un pulsante e un potenziometro. Con il pulsante selezioni uno dei colori fondamentali (rosso, verde o blu), mentre con il potenziometro regoli l'intensità di quello selezionato;



Compiti per casa - 2

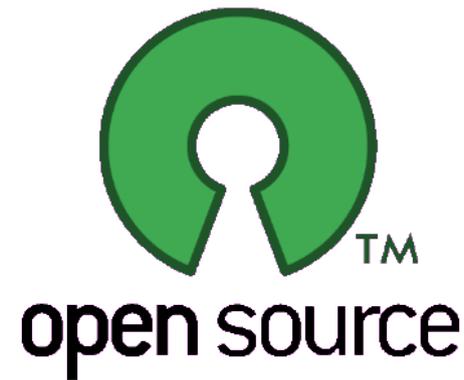
- **Ventilatore:** Realizza una piccola elica in cartoncino da apporre sul motorino, e accendilo con una velocità proporzionale alla misurazione dal sensore di temperatura;
- **Trappola rumorosa:** fai vibrare il motorino se rilevi un'ombra sulla fotoresistenza;



Compiti per casa - 3

- **Servo comando:** muovi il servo nella posizione letta da un potenziometro, stabilizzando le letture per evitare oscillazioni;
- **Termometro a lancetta:** utilizza il servomotore e una termoresistenza per simulare un vecchio termometro a lancetta. Puoi usare del cartoncino per realizzare una freccia ed una scala graduata;





Presentazione realizzata con software open source
(LibreOffice Impress, Gimp, Arduino, Fritzing)

Quest'opera è distribuita con Licenza **CC-BY-SA**
e realizzata da *Stefano Panichi e Giulio Fieramosca*

