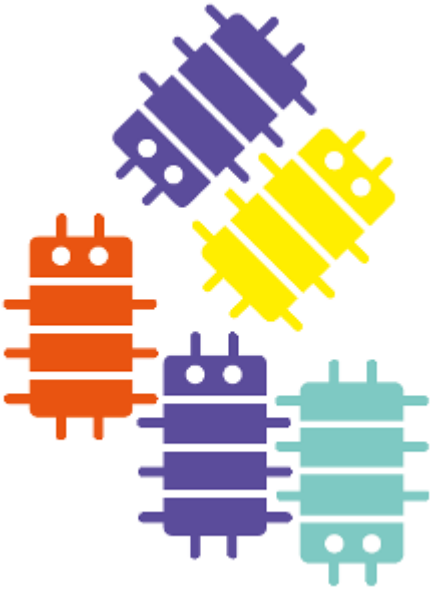
Arduino

Corso base

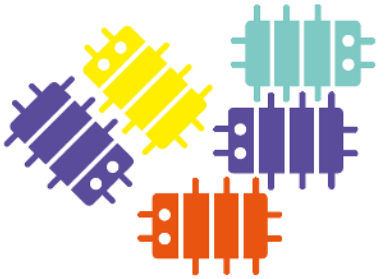


HackLab Terni

Laboratorio aperto a tutti di
elettronica, scienza e arte.

hacklabterni.org

Arduino



Cos'è Arduino?

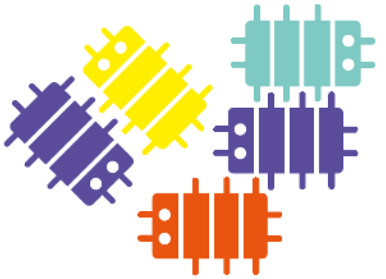
Arduino è una piattaforma di prototipazione elettronica open-source basata su microcontrollore pensata per rendere più accessibile l'uso dell'elettronica nei progetti multidisciplinari.

Permette ad artisiti, designer, hobbysti ed esperti di elettronica di collaborare nella realizzazione di oggetti e ambienti interattivi.

Interagisce con l'ambiente esterno ricevendo informazioni da sensori (interruttori, potenziometri, sensori di temperatura, luce, pressione, ...) attraverso le porte di ingresso e inviando informazioni ad attuatori (motori, luci, display, strumenti musicali, ...) collegati alle porte di uscita.

Gli oggetti realizzati con Arduino possono essere di tipo stand-alone oppure collegati ad un computer che comunica con Arduino attraverso programmi sviluppati usando diversi linguaggi di programmazione (Processing, Python, C++, Pure Data, …).

Arduino



Cos'è Arduino?

Il microcontrollore nella scheda di Arduino viene programmato usando un linguaggio chiamato Wiring dalla sintassi simile al C++ ma molto semplificato ed un ambiente di sviluppo (IDE) basato su Processing anche esso open-source.

Tutte le informazioni necessarie alla realizzazione di Arduino (schemi elettrici, disegno del circuito stampato, lista dei componenti) ed il software necessario alla programmazione sono disponibili con una licenza open-source e liberamente scaricabili dal sito arduino.cc

La licenza usata dal team di Arduino è la “Creative Commons Attribution Share-Alike”, che permette di realizzare lavori derivati sia personali che commerciali purché venga dato credito ad Arduino e i progetti vengano rilasciati con la stessa licenza.

Le schede possono essere acquistate preassemblate, direttamente dal sito arduino.cc o da diversi altri siti in tutto il mondo, o autocostruite comprando i singoli componenti e seguendo le istruzioni liberamente scaricabili.

Arduino

Ripasso elettronica per principianti

Tensione: simbolo V si misura in Volt (V)

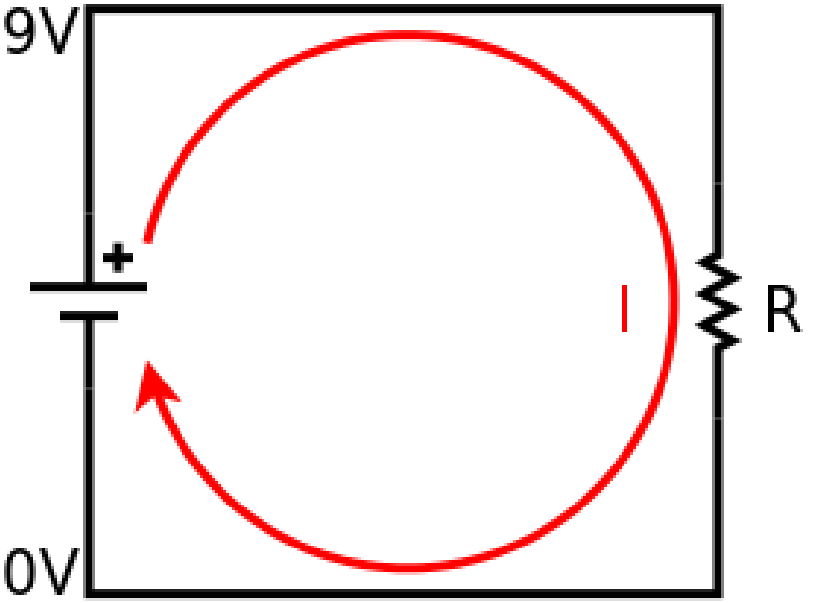
Corrente: simbolo I si misura in Ampere (A)

Resistenza: simbolo R si misura in Ohm (Ω)

La Tensione è una differenza tra due punti, normalmente si considerea la differenza rispetto a un punto di riferimento posto a 0V

La tensione causa lo scorrere della corrente (flusso di cariche) nel circuito

La legge di Ohm mette in relazione queste tre grandezze: V = R x I

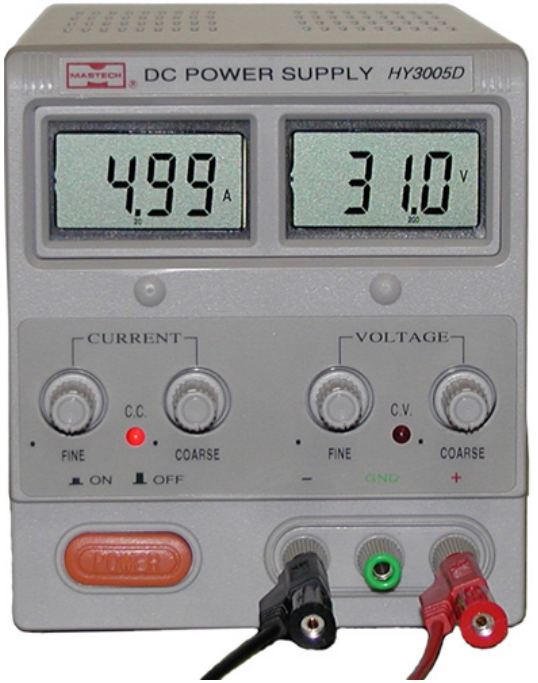


Arduino

Tensioni e correnti alternate e continue (AC - DC)

Esempi di generatori di tensioni continue (DC):

- Pile e batterie
- Alimentatori da laboratorio
- Alimentatori per notebook, smartphone, tablet, ..

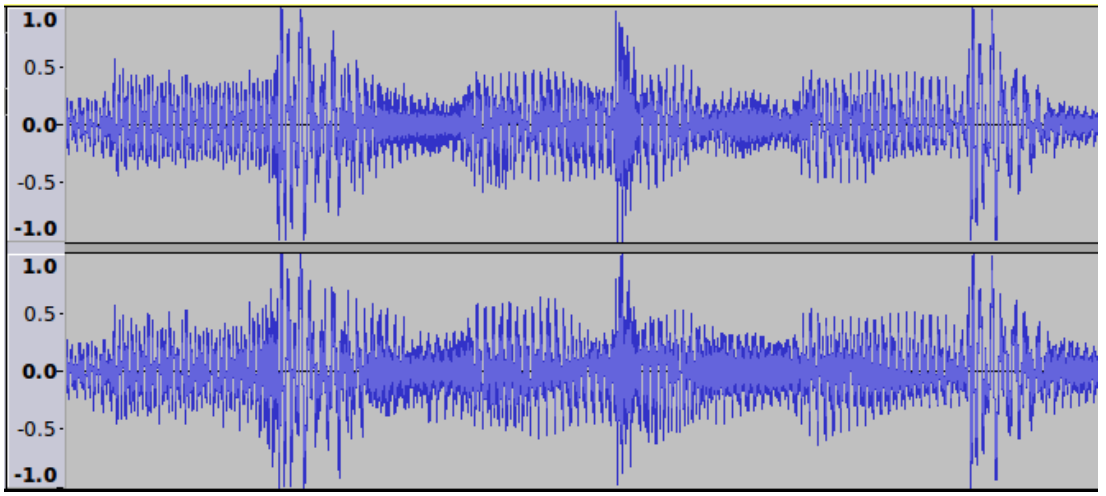


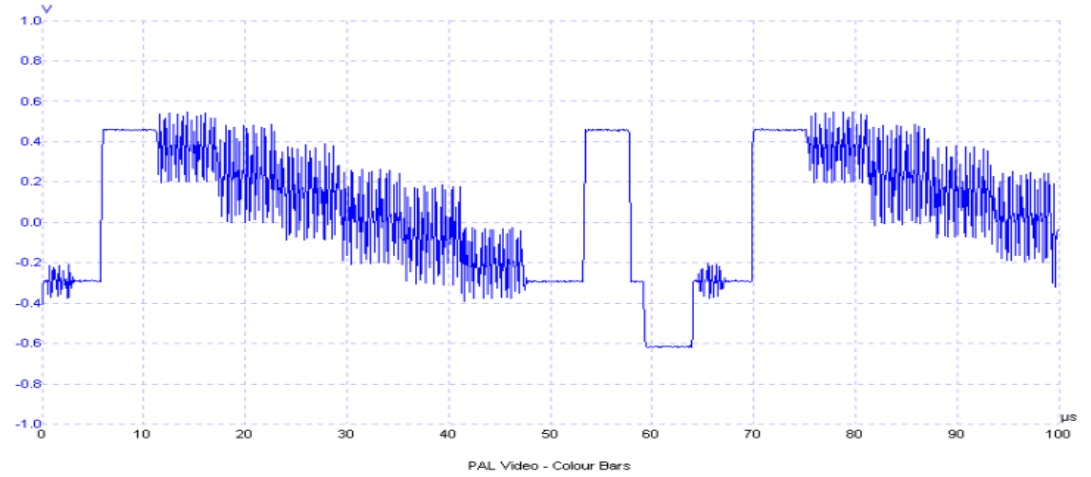
Arduino

Tensioni e correnti alternate e continue (AC - DC)

Esempi di generatori di tensioni alternate (AC):

- Tensione di rete (220V 50Hz)
- Generatori di segnali da laboratorio (HW e SW)
- Segnali audio e video

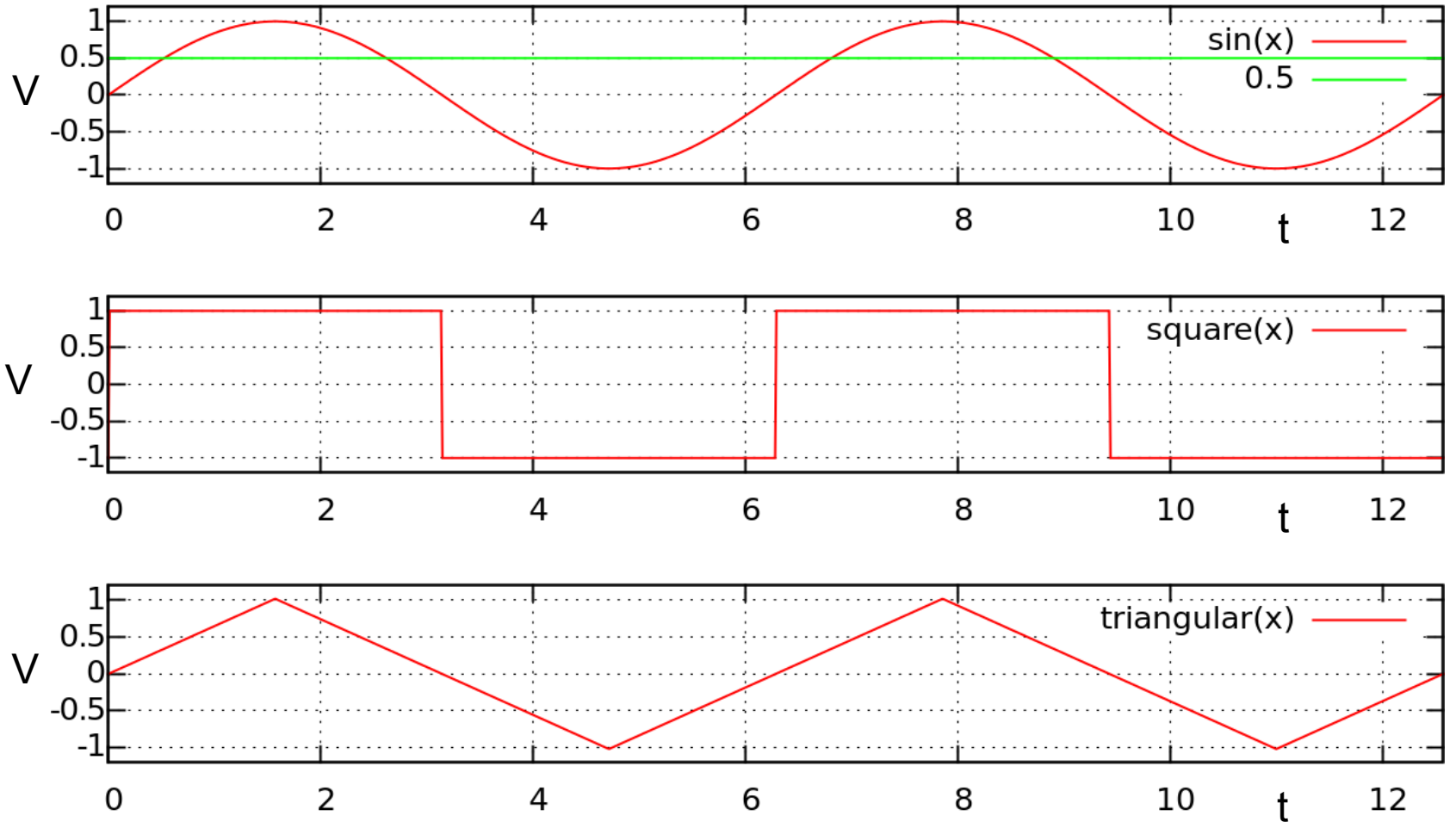




Arduino

Tensioni e correnti alternate e continue (AC - DC)

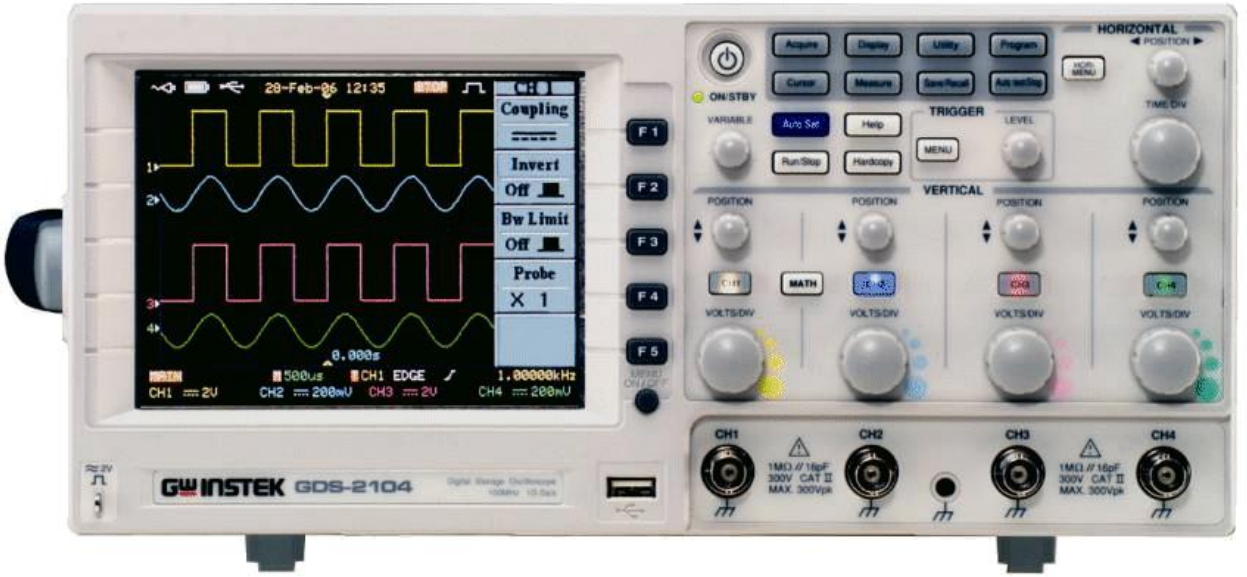
Grafici dell'andamento della tensione rispetto al tempo.
Tensione continua più tre tipiche forme d'onda (sinusoidale, quadrata, triangolare)



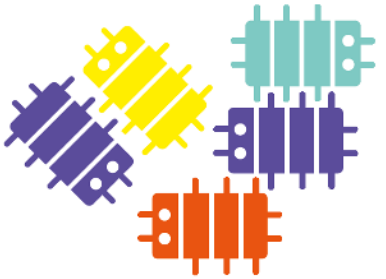
Arduino

Tensioni e correnti alternate e continue (AC - DC)

Il multimetro e l'oscilloscopio sono due importanti strumenti di laboratorio per la misura di tensioni e correnti alternate e continue.



Arduino



Tensioni e correnti alternate e continue (AC - DC)

Un concetto importante:

Distinzione tra tensioni / correnti di alimentazione e tensioni / correnti usate come segnali per veicolare informazione.

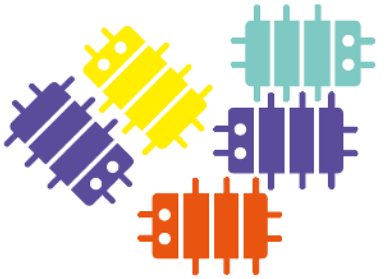
La tensione (e la corrente che scorre nel circuito come effetto) di una batteria o un alimentatore servono a fornire energia al circuito in questione, lo alimentano.

Le tensioni e correnti che entrano o escono da un circuito elettronico e che rappresentano un segnale (audio, video, …) mediante le loro variazioni nel tempo servono a produrre o consumare informazione.

Le seconde sono normalmente una frazione molto piccola delle prime, visto che non è importante il valore assoluto ma le variazioni che, da un istante all'altro, codificano una informazione.

La pila che alimenta Arduino gli fornisce l'energia necessaria a funzionare, le tensioni che entrano o escono dalle porte input / output codificano una informazione letta da un sensore (es. una temperatura) o scritta verso un attuatore (es. La posizione di un servomotore.

Arduino



Segnali analogici e digitali

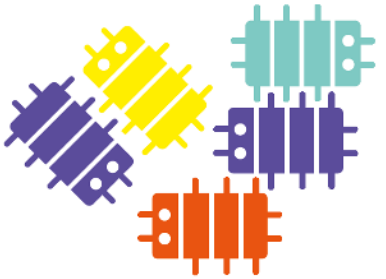
Segnali analogici
Nel mondo reale abbiamo a che fare con grandezze che variano con continuità: temperatura, intensità luminosa, posizione angolare di un servomotore, …

Segnali digitali
All'interno di Arduino (come in un computer) abbiamo a che fare con numeri, qualsiasi tipo di informazione viene codificata e gestita tramite numeri binari (in base 2, sequenze di 1 e 0)

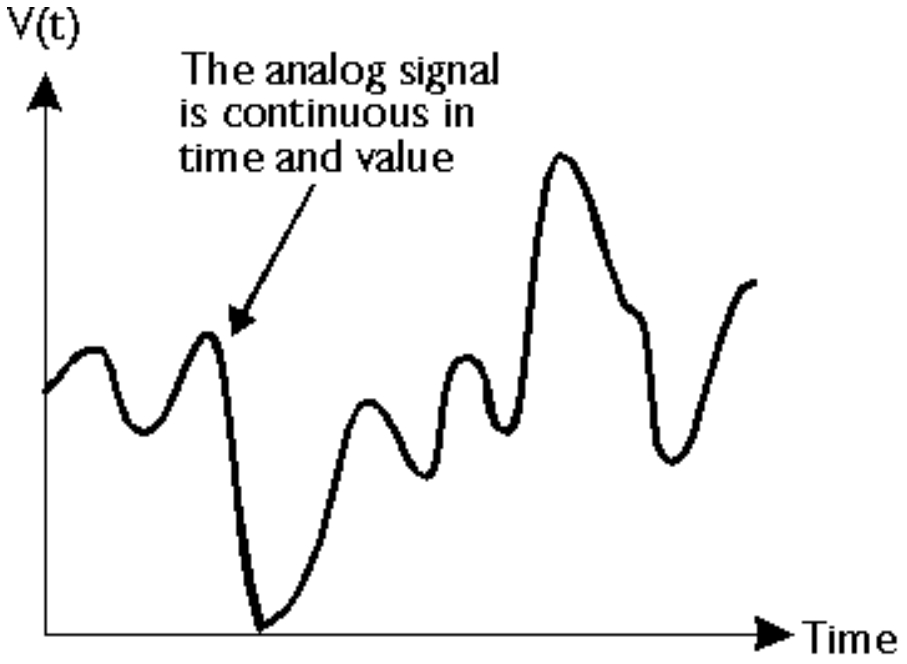
Arduino usato come piattaforma di “physical computing” mette in comunicazione questi due mondi. Le informazioni relative a grandezze fisiche vengono “lette” dal mondo esterno e convertite in numeri da Arduino che, dopo averle elaborate le riconverte in segnali analogici per completare l'interazione.

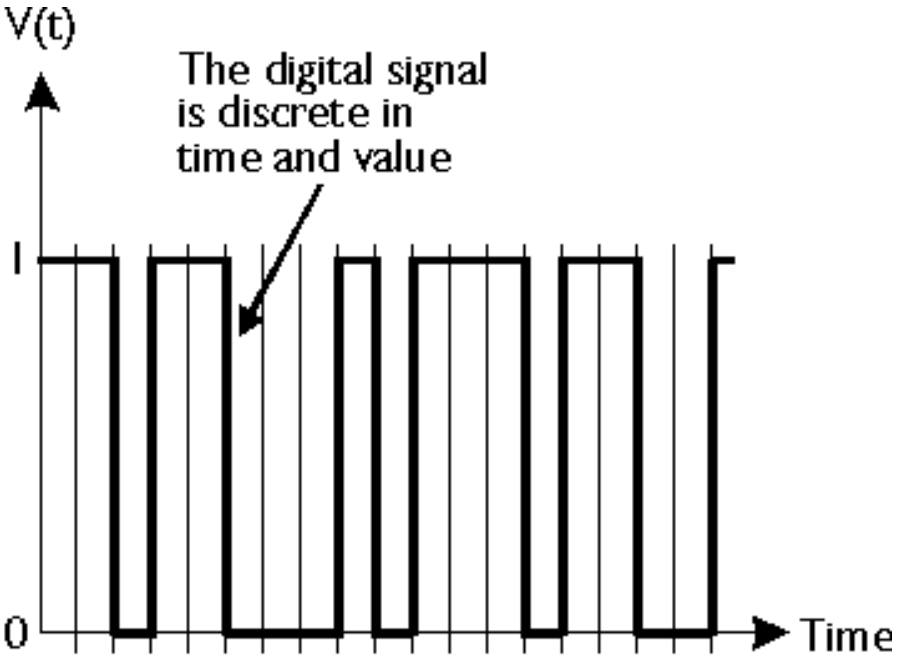
Per I segnali analogici è importante conoscere l'ampiezza in ogni istante di tempo, per I segnali digitali è importante lo stato (0,1) in istanti di tempo discreti (eventi, clock)

Arduino

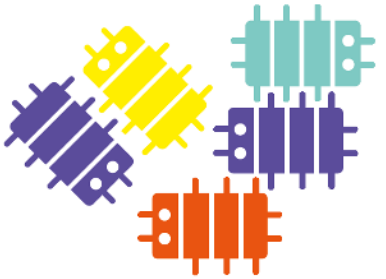


Segnali analogici e digitali



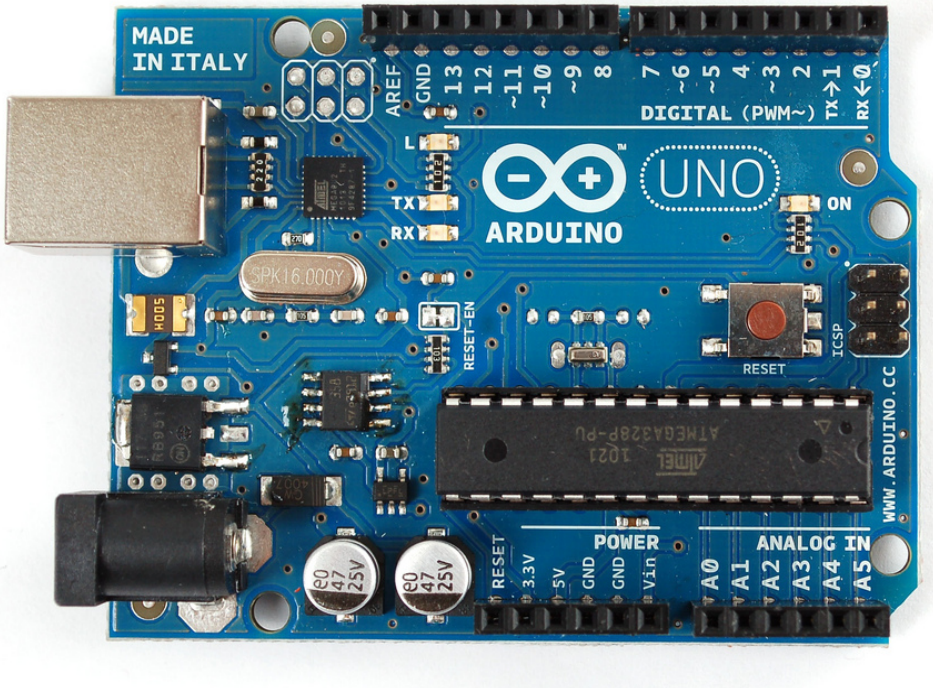


Arduino

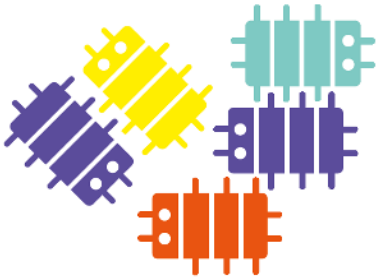


I componenti fondamentali di Arduino

- MCU Micro Controller Unit (funzioni logiche e aritmetiche, memoria, dispositivi input/output)
- Clock (temporizzatore - direttore di orchestra)
- Porta USB (alimentazione, programmazione e comunicazione con PC)
- Porte di input: leggo un valore da un dispositivo esterno (sensore)
- Porte di output: scrivo un valore verso un dispositivo esterno (attuatore)



Arduino



Microcontrollore
- ATMega328

Porte di I/O Digitale
- 14 (delle quali 6 PWM)

Porte di Input Analogico
- 6

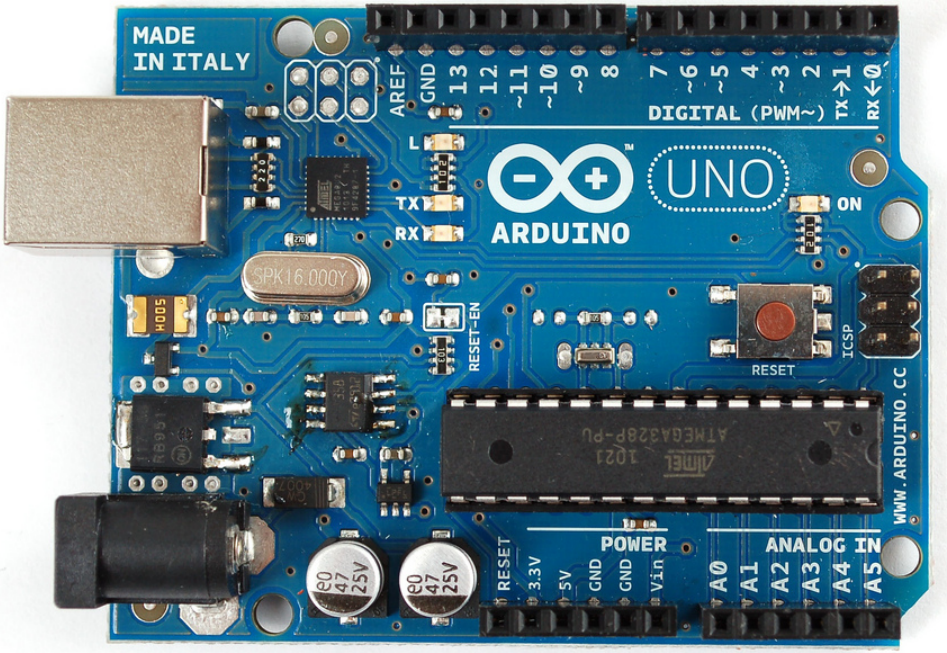
Memoria Flash
- 32KB (0.5KB bootloader)

SRAM
- 2KB

EEPROM
- 1KB

Velocità del Clock
- 16MHz

(fonte: arduino.cc)



Arduino Uno

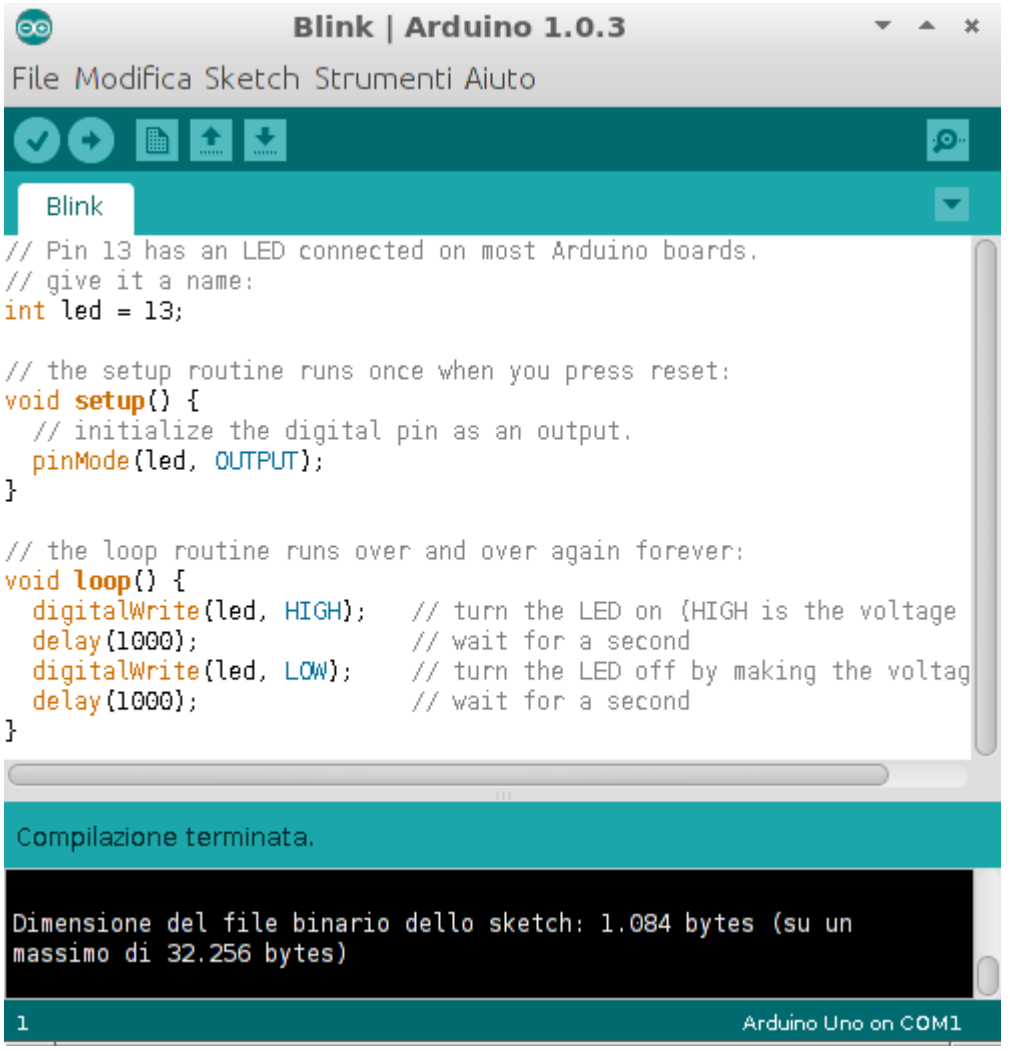
Arduino

L'ambiente di sviluppo (IDE)

La parte principale dell'IDE è costituita dall'editor di testi con il quale si scrivono i programmi per Arduino.

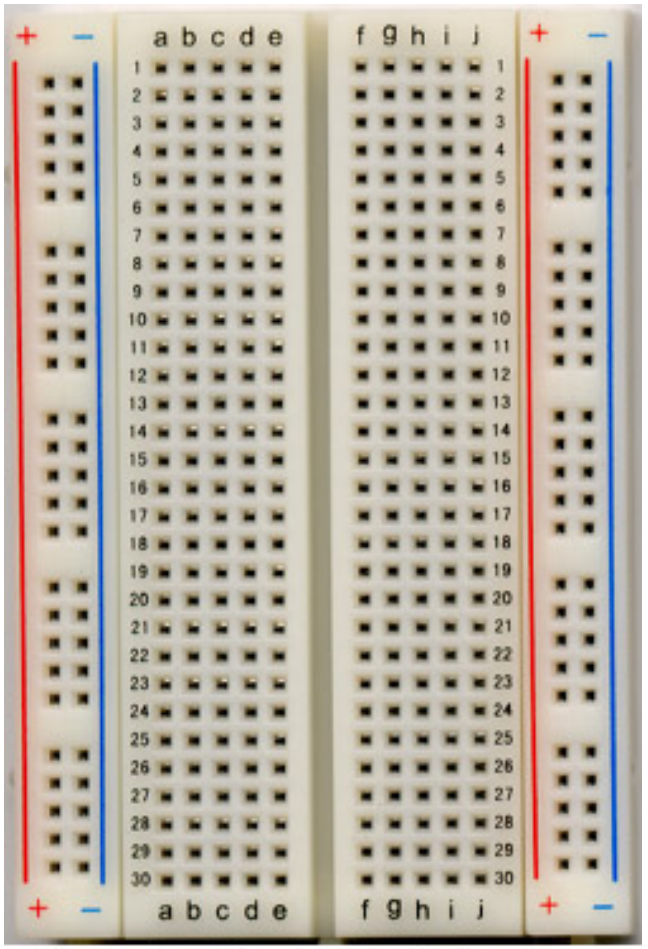
Nella parte in alto si trovano alcuni pulsanti con i quali è possibile compilare, verificare e caricare nel microcontrollore di Arduino il programma scritto.

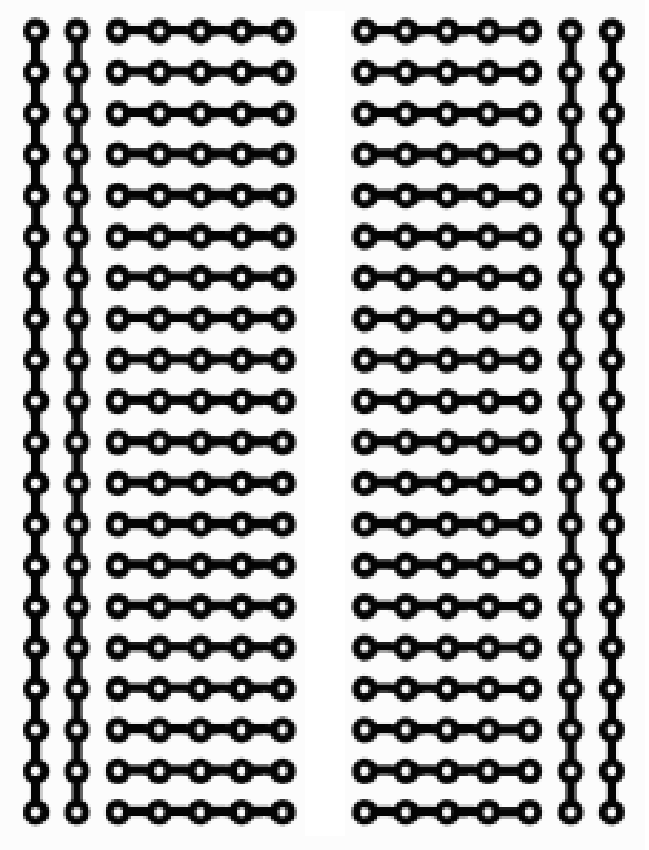
La console nella parte in basso dell'IDE visualizza i messaggi relativi alle azioni effettuate.



Arduino

La breadboard

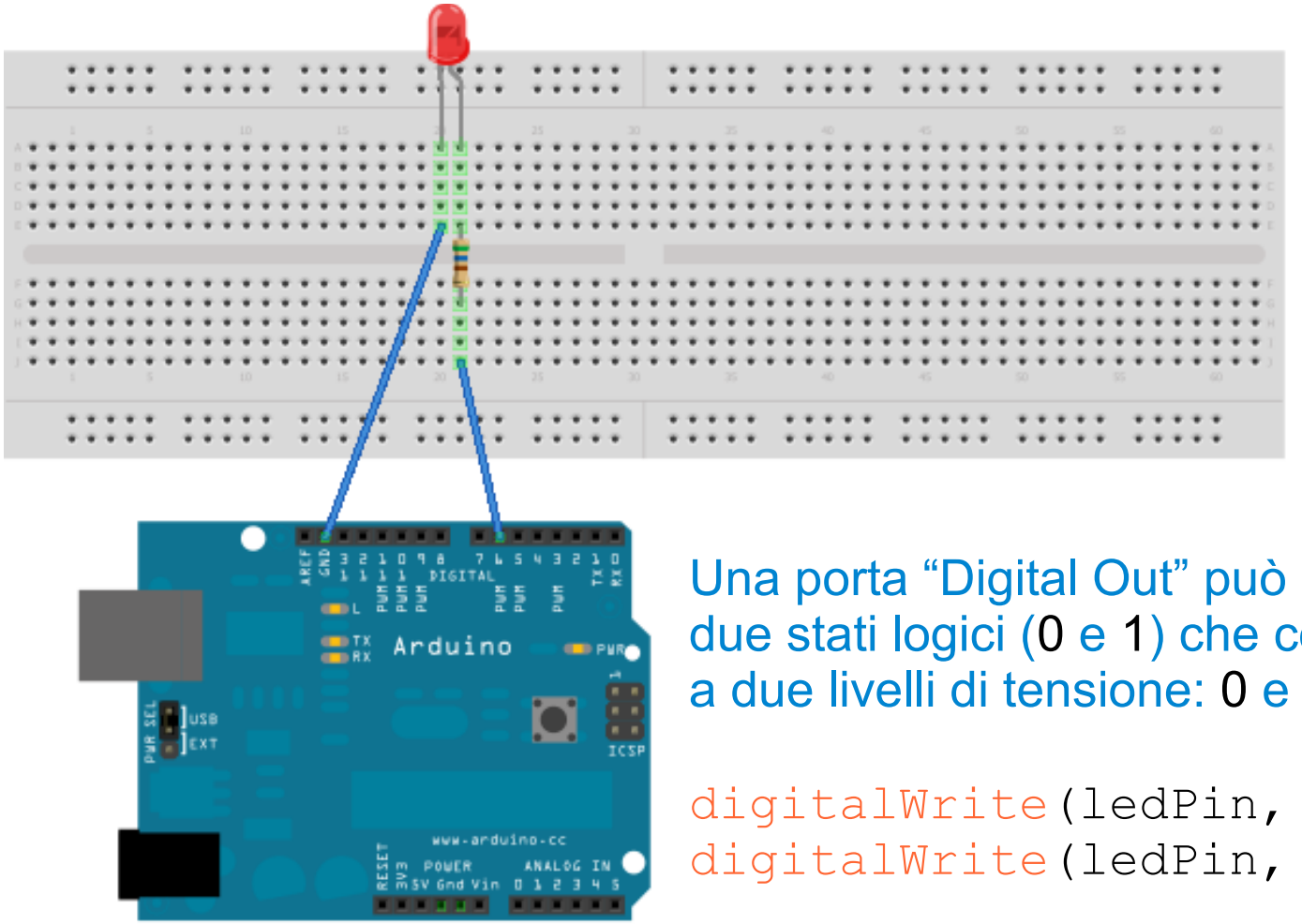




Arduino

Blink

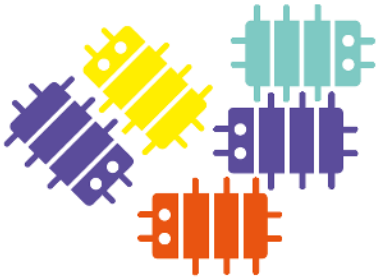
Usa una porta “Digital Out” per far lampeggiare un LED



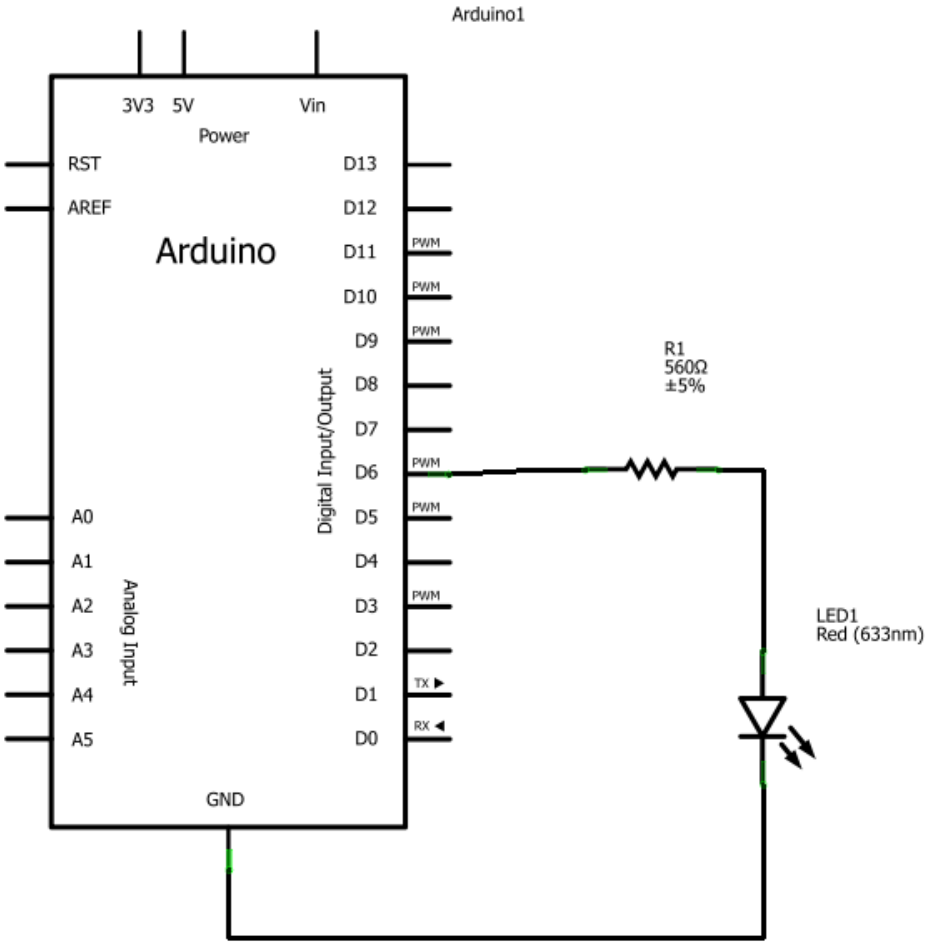
Una porta “Digital Out” può assumere due stati logici (0 e 1) che corrispondono a due livelli di tensione: 0 e 5V

```
digitalWrite(ledPin, LOW);
digitalWrite(ledPin, HIGH);
```

Arduino



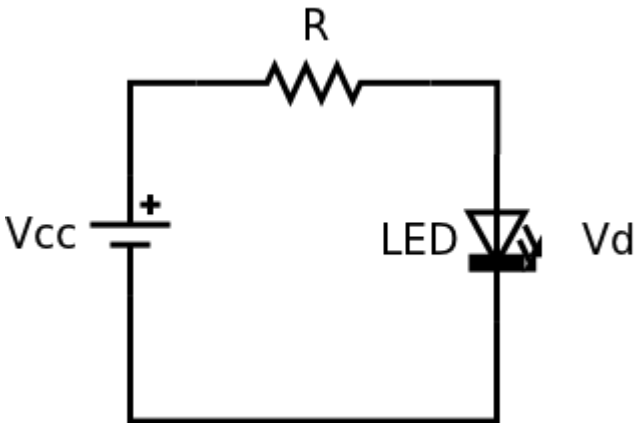
Blink



Arduino

Cenni sui LED

Un LED (Light Emitting Diode) o diodo ad emissione luminosa è una sorgente luminosa a semiconduttore. Quando un LED è alimentato gli elettroni si ricombinano con le lacune rilasciando energia sottoforma di fotoni.

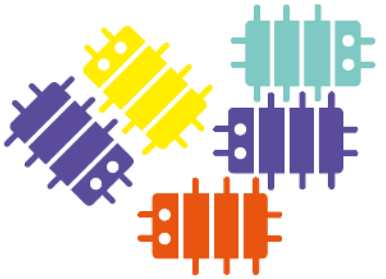


Per fornire al LED la corretta corrente di funzionamento si mette una resistenza in serie. Il valore della resistenza viene calcolato usando la legge di Ohm.

Vcc = 5V
Vd = 2V
I = 10mA

R = (Vcc-Vd) / I = (5 – 2) / 0.01= 300 Ohm

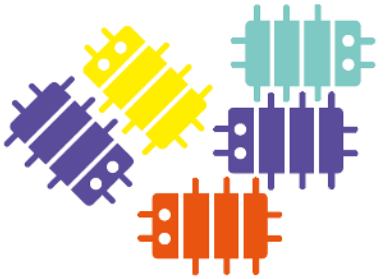
Arduino



Blink

```
int ledPin = 6;          // the number of the LED pin

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
}

void loop(){
  digitalWrite(ledPin, HIGH);   // turn LED on
  delay(1000);                  // wait for a second
  digitalWrite(ledPin, LOW);    // turn LED off
  delay(1000);                  // wait for a second
}
```

Arduino



Programmazione

Il linguaggio di programmazione è basato su Wiring e l'ambiente di sviluppo (IDE) su Processing. Un programma per Arduino viene chiamato "Sketch".

Uno Sketch è una sequenza di istruzioni che dicono ad Arduino cosa deve fare

Tipi di Istruzioni:

- Assegnazione di valori o risultato di espressioni ad una variabile

```
ledPin = 6;
somma = 2 + 2;
somma2 = somma + 2;
```

- Istruzioni condizionali

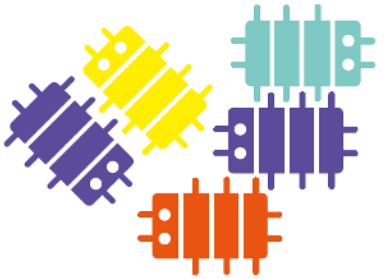
```
if (buttonState == LOW) {
    digitalWrite(ledPin, HIGH);     // turn LED on
} else {
    digitalWrite(ledPin, LOW);      // turn LED off
}
```

- Cicli

```
for (i = 0; i <= 255; i++) {
    analogWrite(ledPin, i);      // fade in LED
    delay(10);
}
```

Arduino

Programmazione



Variabili
Dichiarazione (associo il nome e il tipo e viene riservata la memoria) e inizializzazione (associo un valore iniziale – opzionale)

```
int buttonState = 0;
float temperature = 23.5;
char key = 'A';
```

Funzioni
Servono a raggruppare un insieme di istruzioni che eseguono un particolare compito

```
int somma(int a, int b) {
    int c = a + b;

    return c;
}
```

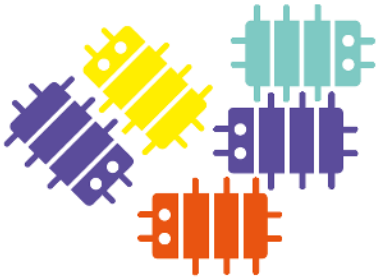
In Arduino ci sono due funzioni speciali che devono essere sempre definite dall'utente

`setup()`: viene chiamata una sola volta all'inizio del programma per l'inizializzazione di variabili e porte

`loop()`: viene chiamata dopo la setup() ripetutamente fino al reset o power-off

Arduino

Programmazione

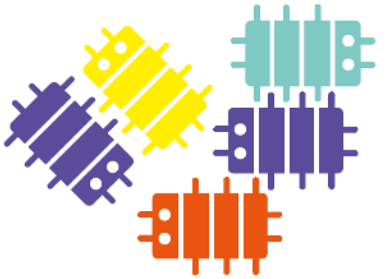


Sketch minimale per Arduino (BareMinimum in File->Examples->01.Basics)

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Se lo compiliamo e lo carichiamo in Arduino non fa nulla però non da errori ne sintattici ne semantici (già è qualcosa)

Arduino



Programmazione

Uso della porta seriale per la comunicazione tra Arduino e il PC per capire cosa succede all'interno di Arduino quando mando in esecuzione un programma.

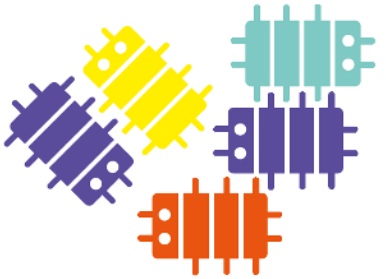
Per usare la porta seriale devo inizializzarla nella funzione setup()

```
void setup() {
    // initialize the Serial Port
    Serial.begin(9600);
    Serial.write("hello ");
    s = somma(3, 5);
    Serial.println(s);
}

void loop() {
}

int somma(int a, int b) {
  int c = a + b;
  return c;
}
```

Se lo compiliamo e lo carichiamo in Arduino scrive “hello 8” nel “Serial Monitor” ogni volta che premiamo il tasto RESET.

Arduino

Programmazione



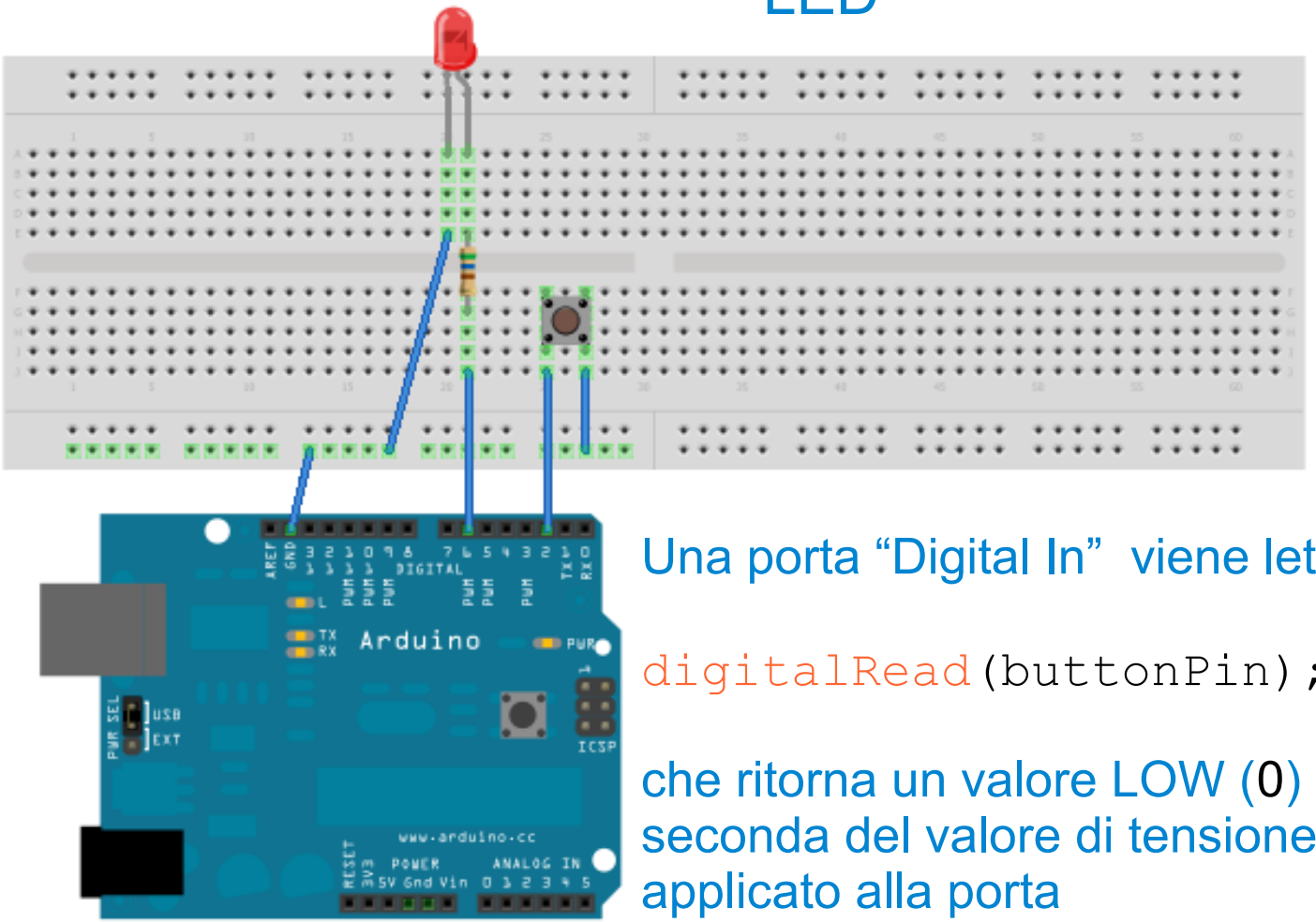
Tornando all'esempio “Blink” vediamo come avere un feedback di quello che succede nel “Serial Monitor”.

```
int ledPin = 6;          // the number of the LED pin

void setup() {
  // initialize the Serial Port
  Serial.begin(9600);
  Serial.write("hello\n");
  pinMode(ledPin, OUTPUT);
}

void loop(){
  digitalWrite(ledPin, HIGH);   // turn LED on
  delay(1000);                  // wait for a second
  Serial.write("LED On\n");
  digitalWrite(ledPin, LOW);    // turn LED off
  delay(1000);                  // wait for a second
  Serial.write("LED Off\n”);
}
```

Arduino

Button

Usa una porta “Digital In” per il pulsante e una “Digital Out” per il LED



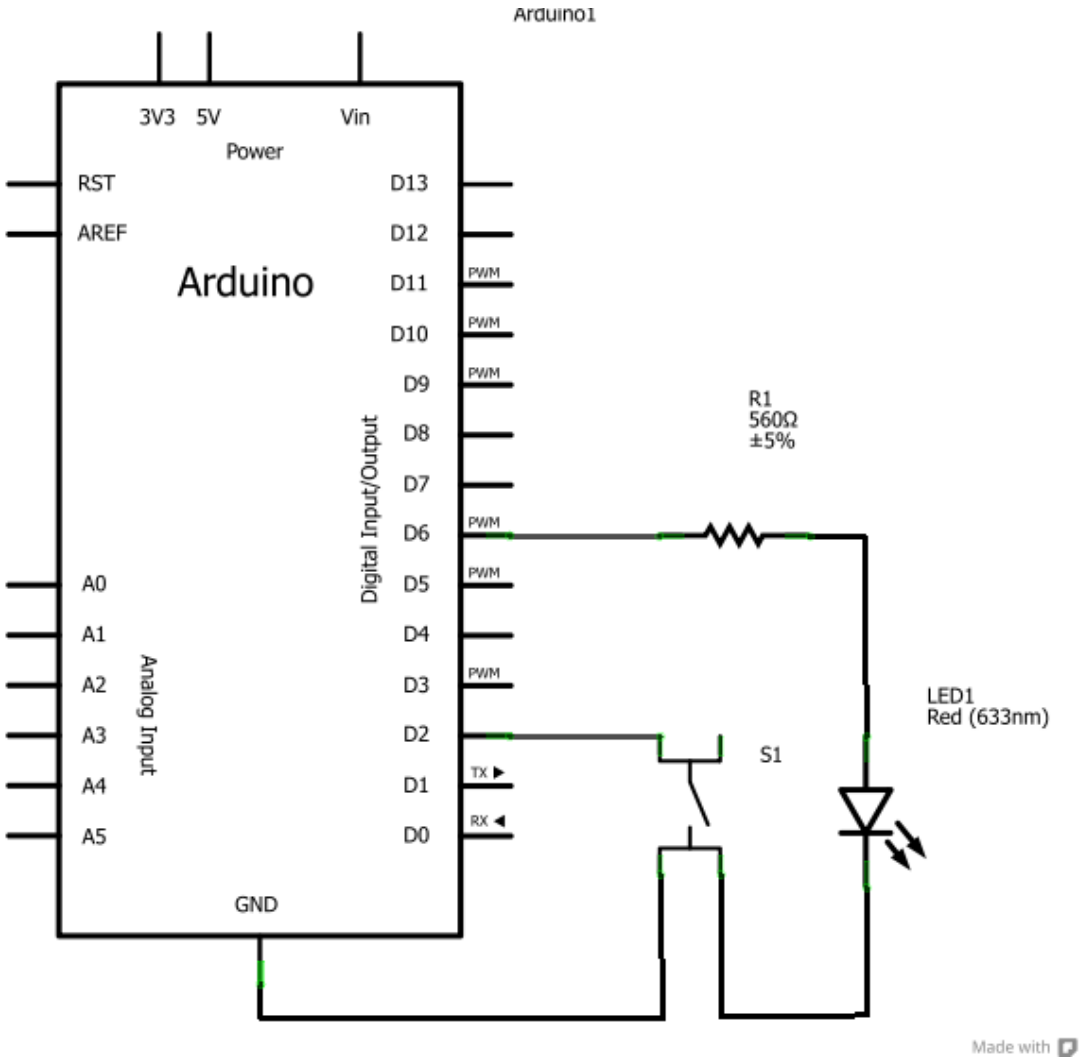
Una porta “Digital In” viene letta dalla funzione:

```
digitalRead(buttonPin);
```

che ritorna un valore LOW (0) o HIGH (1) a seconda del valore di tensione (0 o 5V) applicato alla porta

Arduino

Button

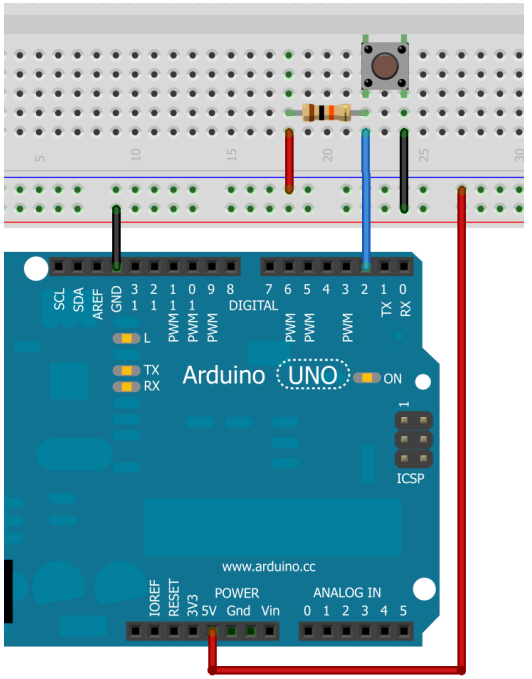


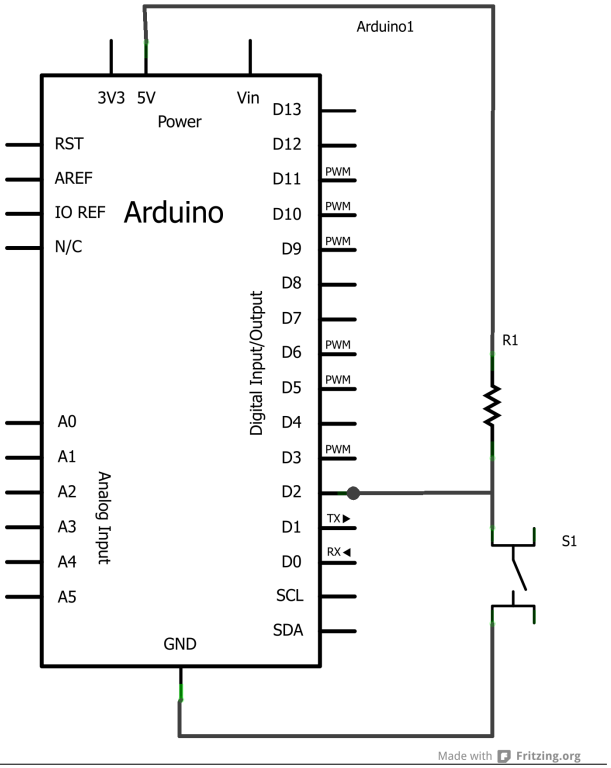
Arduino

Resistenza di pullup

Senza il pullup interno

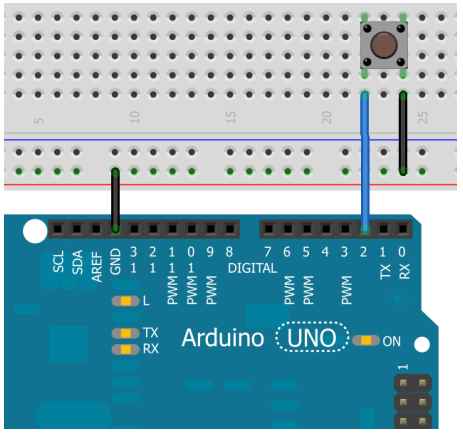
```
int btnPin = 2;
int btnState = 0;

void setup() {
  pinMode(btnPin, INPUT);
}

void loop(){
  btnState =
    digitalRead(buttonPin);

  if (btnState == LOW) {
    // pulsante premuto
  }
}
```

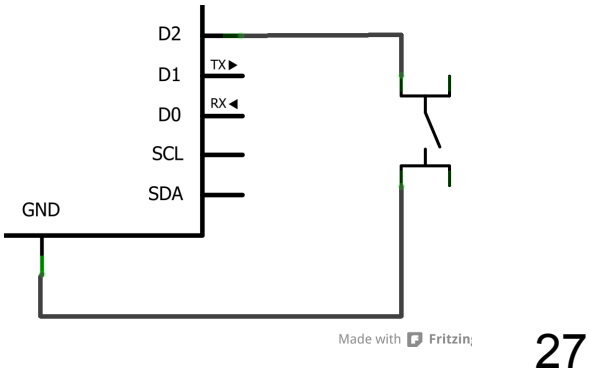




Con il pullup interno

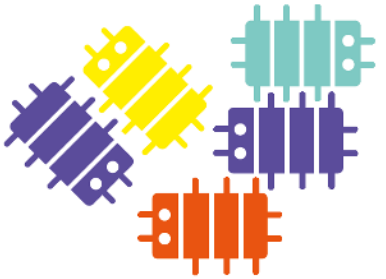
```
int btnPin = 2;
void setup() {
  pinMode(btnPin, INPUT);
  // imposto pullup interno
  digitalWrite(btnPin, HIGH);
}
```





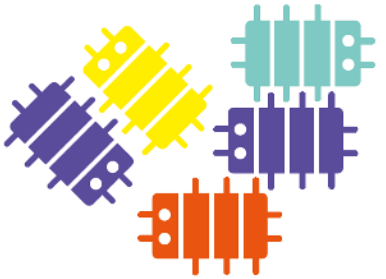
Arduino

Button



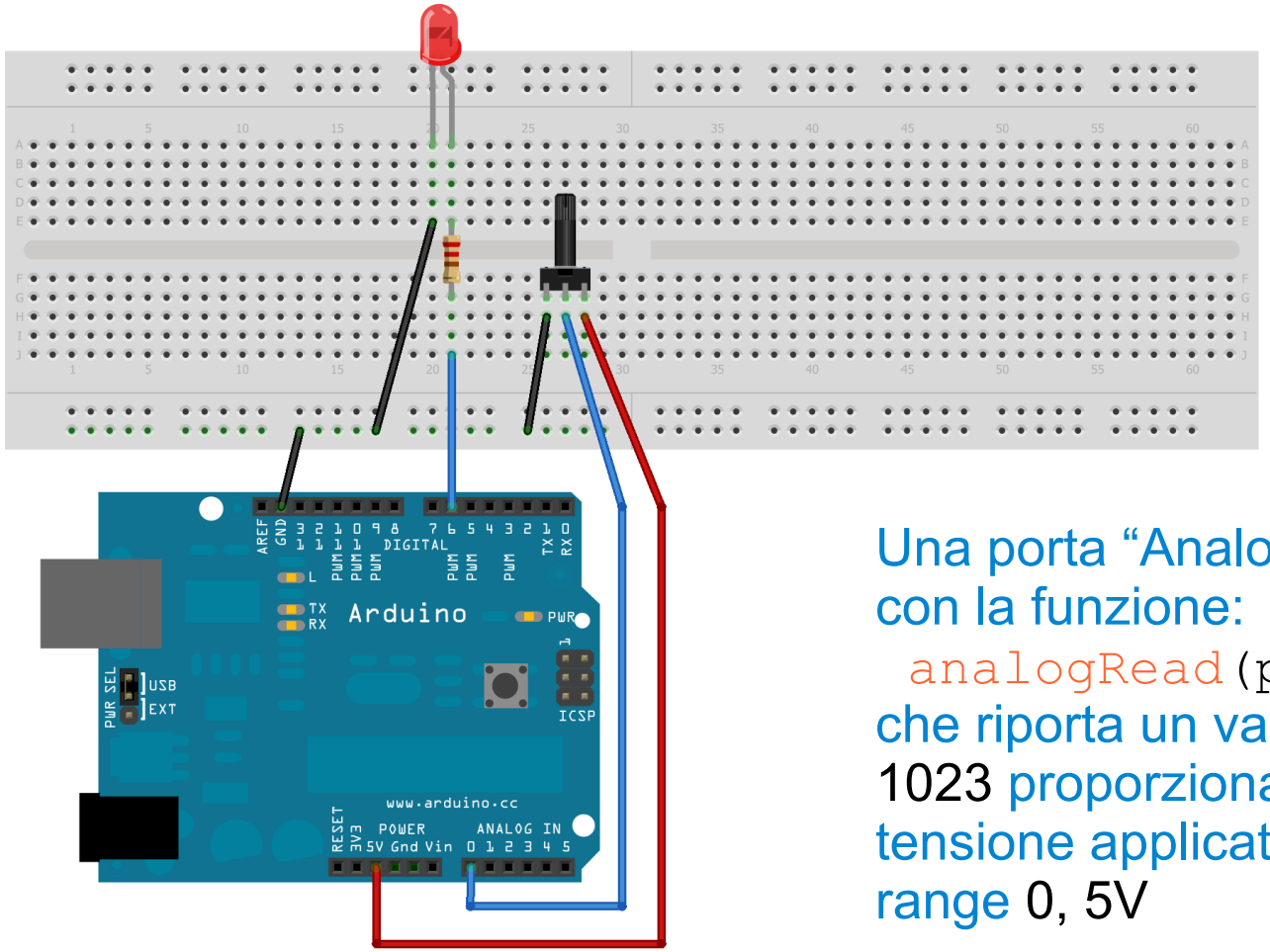
```
int buttonPin = 2;      // the number of the pushbutton pin
int ledPin = 6;         // the number of the LED pin
int buttonState = 0;    // variable for reading the pushbutton status

void setup(){
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  // initialize the pushbutton pin with internal pullup
  digitalWrite(buttonPin, HIGH);
}

void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is LOW:
  if (buttonState == LOW) {
    digitalWrite(ledPin, HIGH);    // turn LED on
  } else {
    digitalWrite(ledPin, LOW);     // turn LED off
  }
}
```

Arduino



FadePot

Usa una porta “Analog In” per leggere la posizione del potenziometro e una “Digital Out” in PWM per il LED



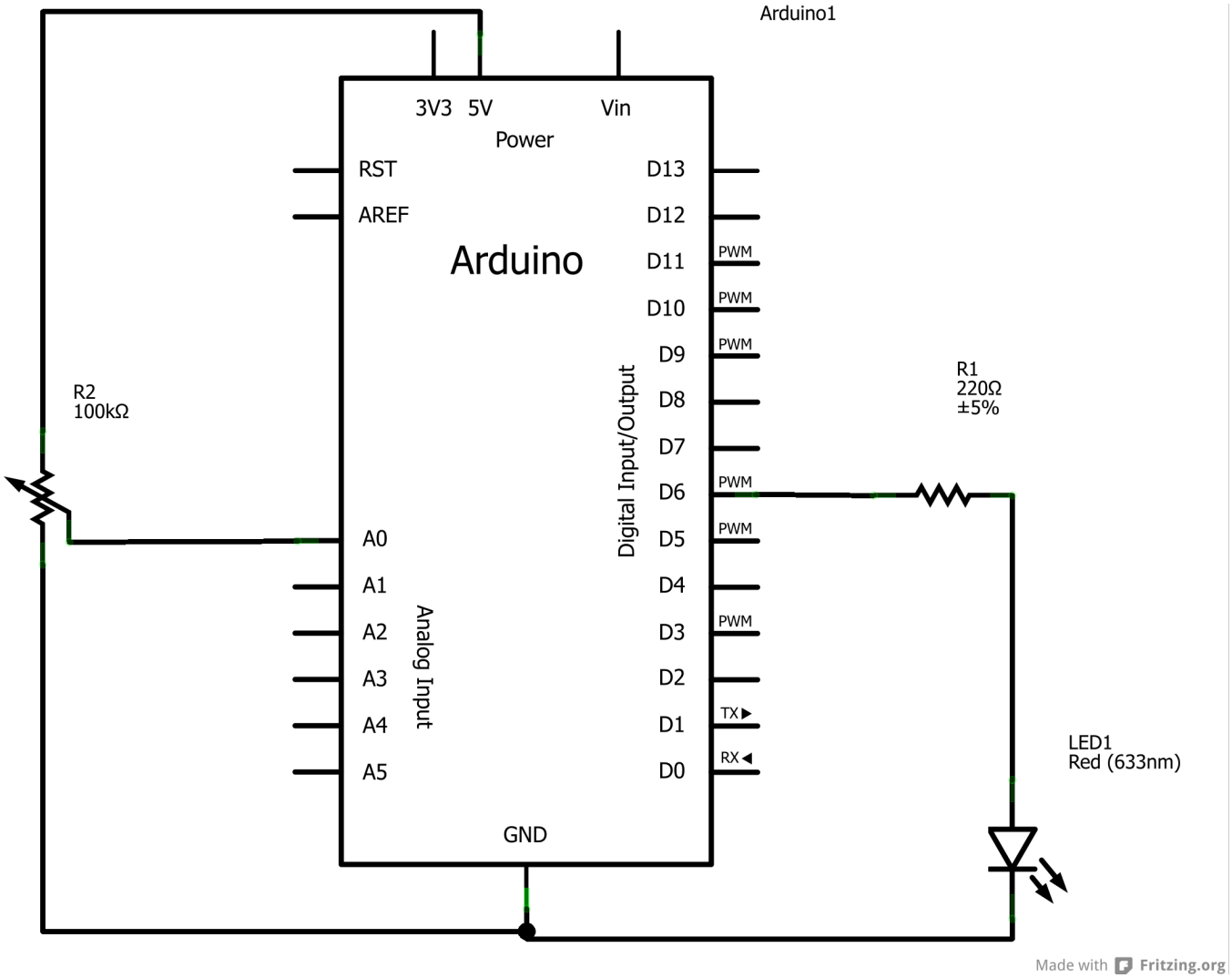
Una porta “Analog In” viene letta con la funzione:

```
analogRead(potPin);
```

che riporta un valore intero tra 0 e 1023 proporzionale al valore della tensione applicata alla porta nel range 0, 5V

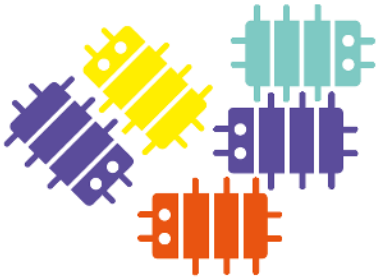
Arduino

FadePot



Arduino

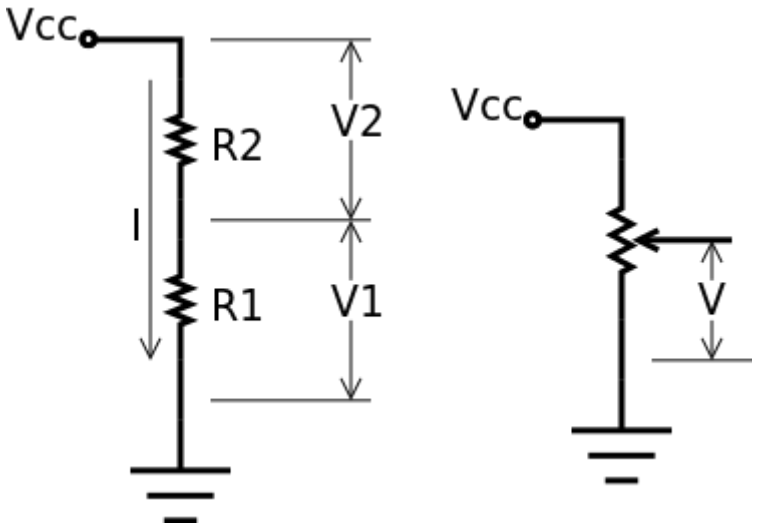
FadePot



```
int ledPin =  6;       // the number of the LED pin
int potPin =  A0;      // analog input pin

int potVal = 0;
int ledVal = 0;

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
}

void loop(){
  potVal = analogRead(potPin);    // 0 - 1023
  ledVal = potVal / 4;
  analogWrite(ledPin, ledVal);    // PWM
  delay(10);
}
```

Arduino

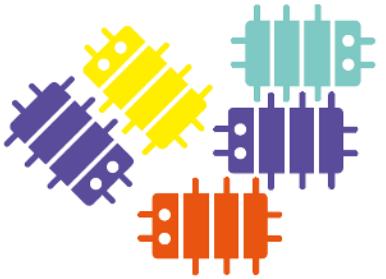
Il partitore resistivo e il potenziometro



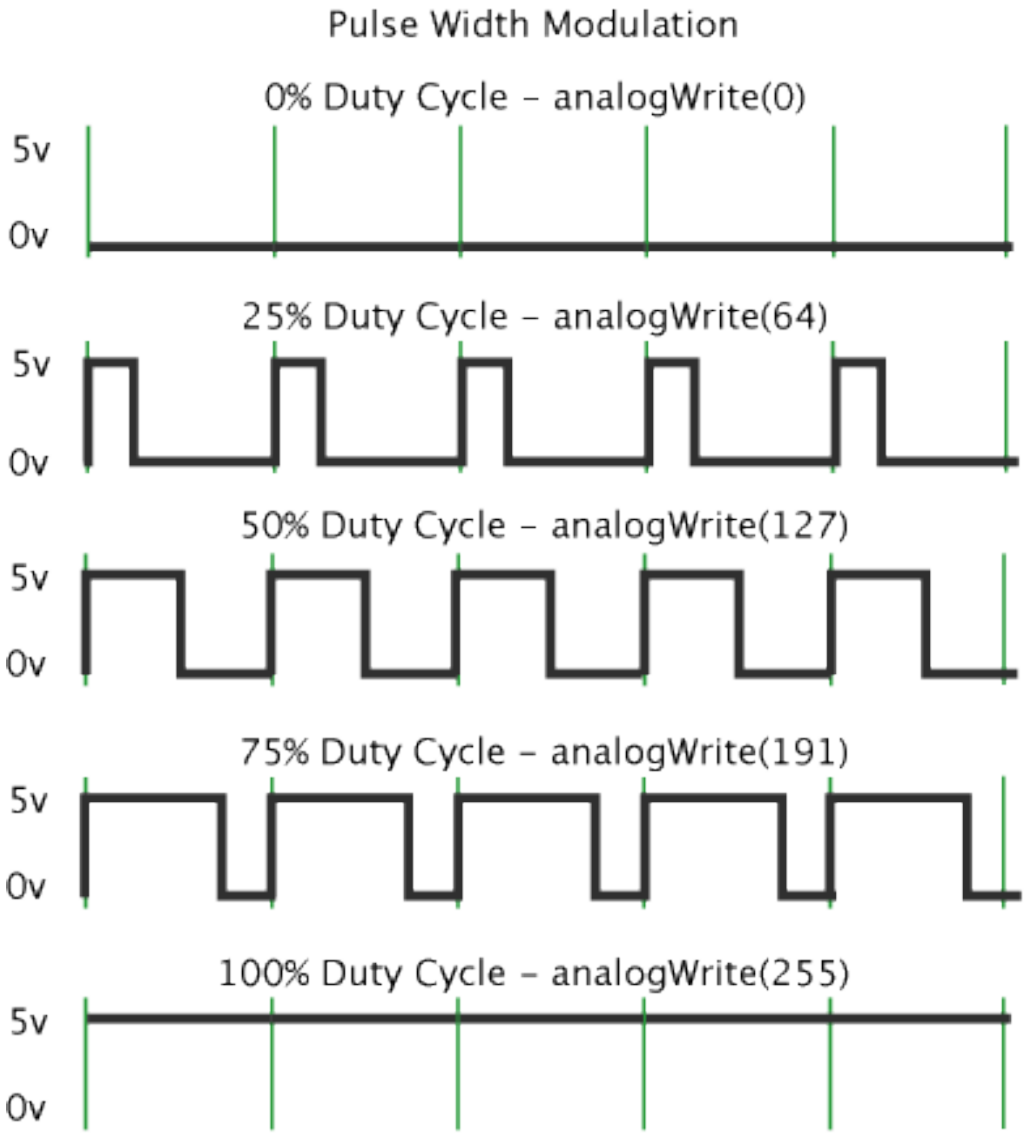
$Vcc = V1 + V2 \quad V1 = I\,R1 \quad V2 = I\,R2$

$I = Vcc / (R1 + R2) \quad V1 = Vcc\,R1 / (R1 + R2) \quad V2 = Vcc\,R2 / (R1 + R2)$

Arduino



PWM (Pulse Width Modulation)



Per scrivere un valore analogico in una porta si usa la funzione:

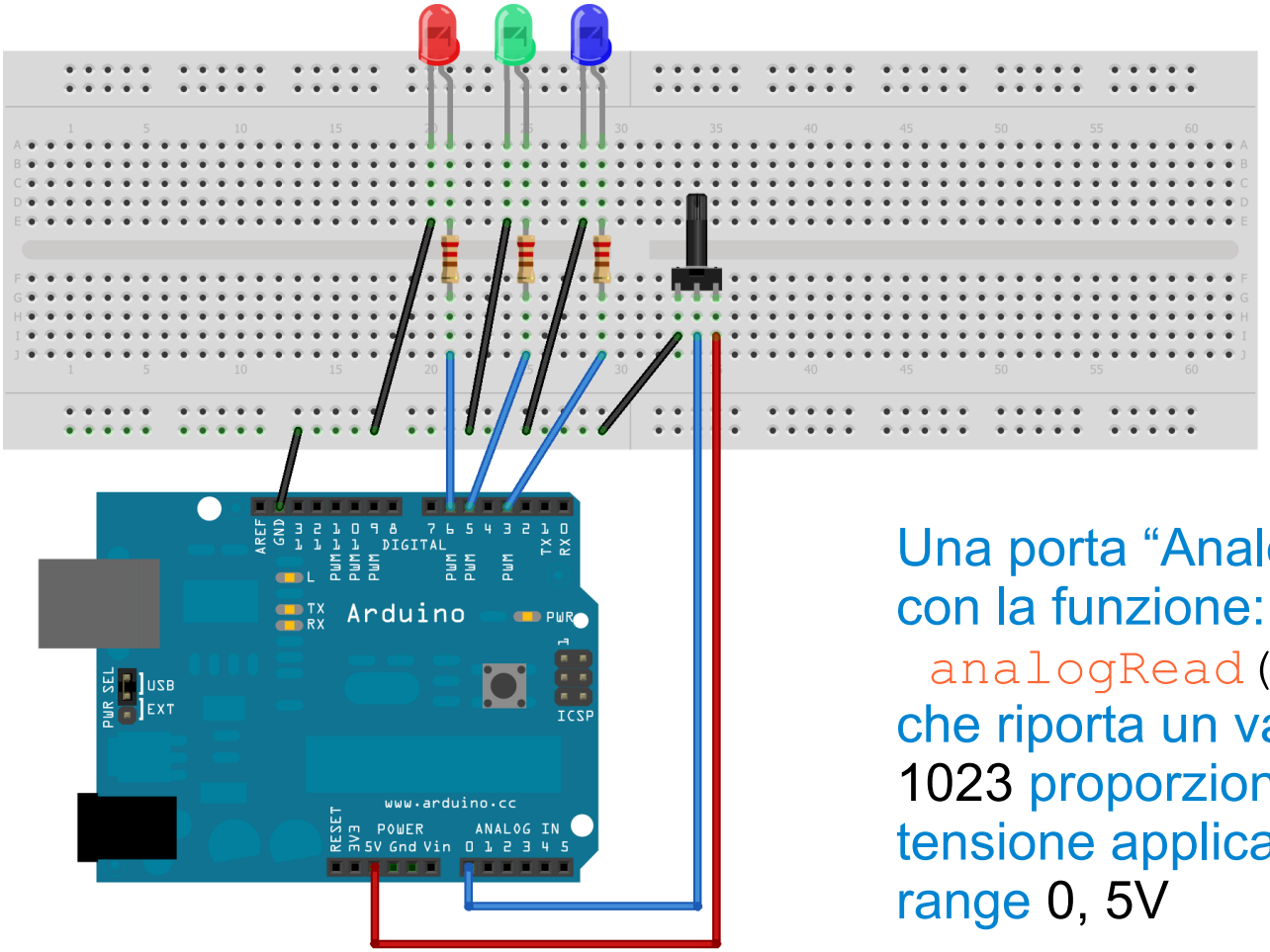
```
analogWrite(ledPin, val);
```

dove val è un intero che può assumere valori tra 0 e 255

Arduino

FadePot RGB

Usa una porta “Analog In” per leggere la posizione del potenziometro e tre “Digital Out” in PWM per i LED



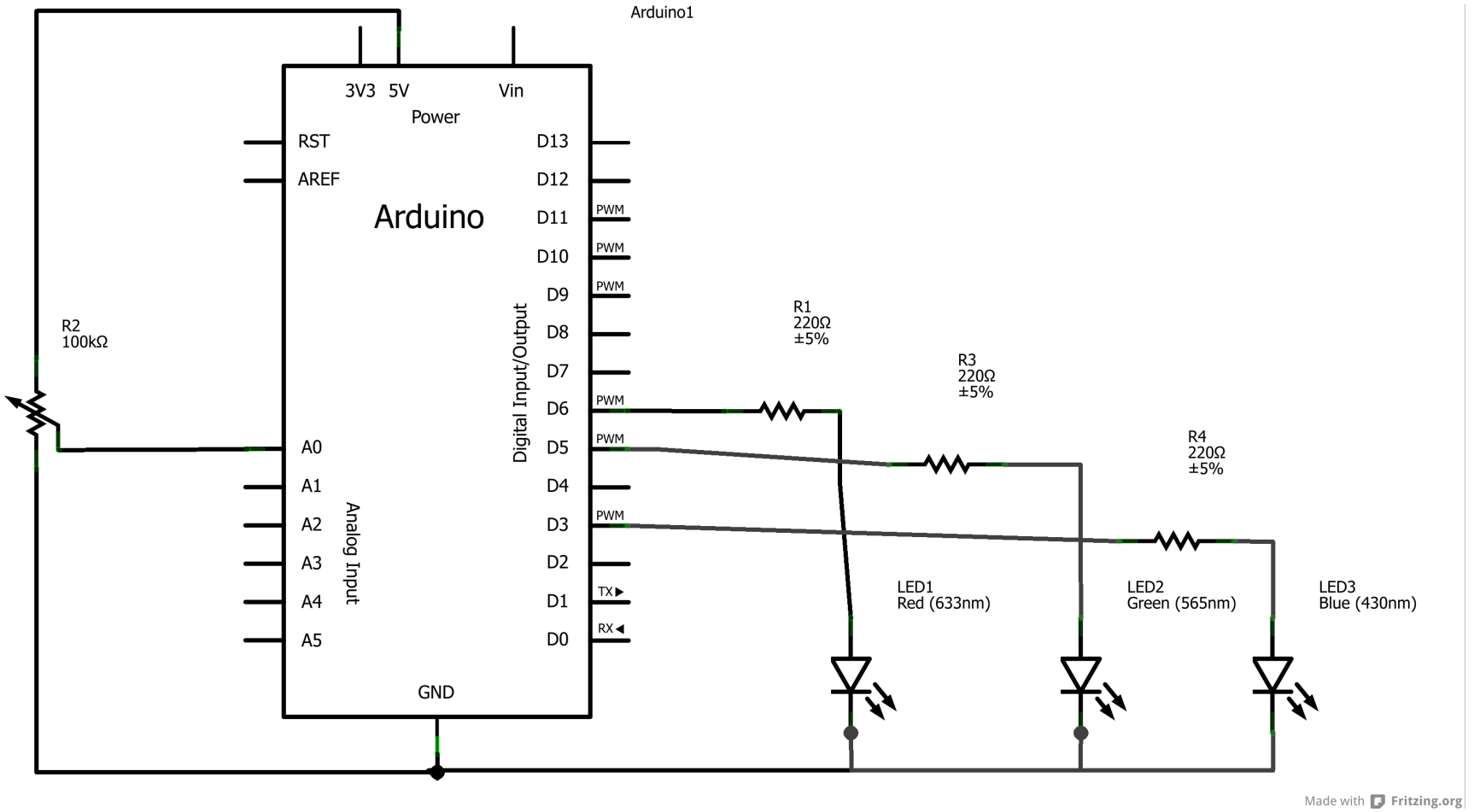
Una porta “Analog In” viene letta con la funzione:

```
analogRead(potPin);
```

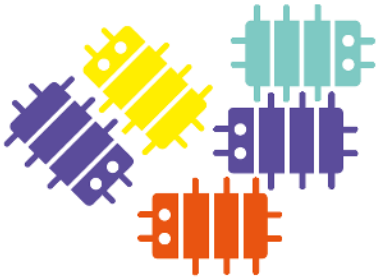
che riporta un valore intero tra 0 e 1023 proporzionale al valore della tensione applicata alla porta nel range 0, 5V

Arduino

FadePot RGB



Arduino

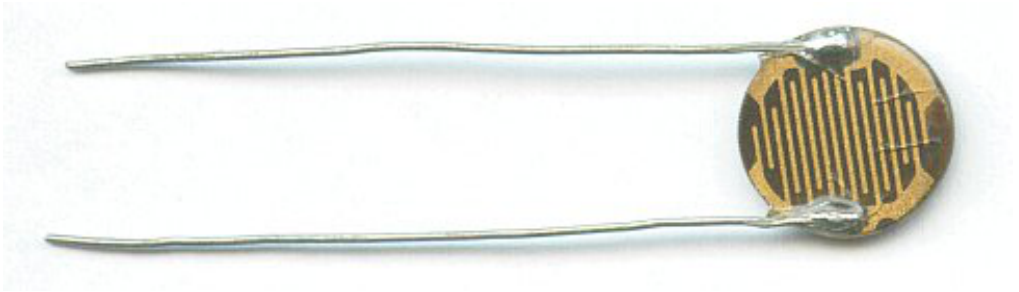


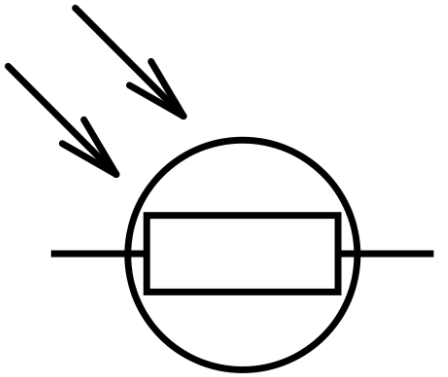
FadePot RGB

```
int ledPinR =  6;       // the number of the Red LED pin
int ledPinG =  5;       // the number of the Green LED pin
int ledPinB =  3;       // the number of the Blue LED pin
int potPin =  A0;       // Analog input pin

int potVal = 0;
int ledVal = 0;

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPinR, OUTPUT);
  pinMode(ledPinG, OUTPUT);
  pinMode(ledPinB, OUTPUT);
}

void loop(){
  potVal = analogRead(potPin);    // 0 - 1023
  ledVal = potVal / 4;
  analogWrite(ledPinR, ledVal);    // PWM
  analogWrite(ledPinG, ledVal);    // PWM
  analogWrite(ledPinB, ledVal);    // PWM
  delay(10);
}
```

Arduino

Il fotoresistore o LDR (Light Dependent Resistor)



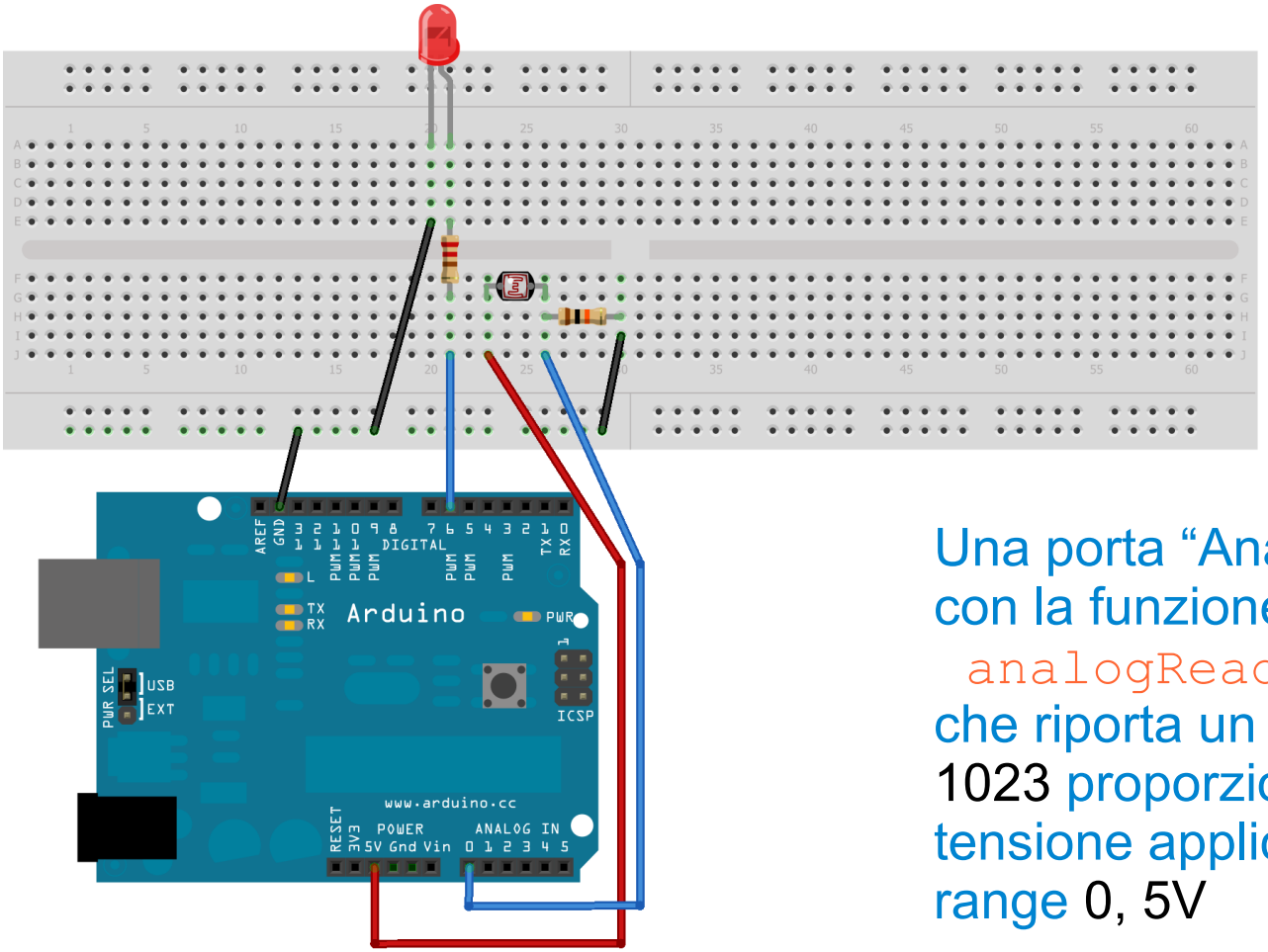


Il fotoresistore o LDR è un componente elettrico la cui resistenza diminuisce all'aumentare della intensità luminosa incidente.

Arduino

FadeLight

Usa una porta “Analog In” per leggere l'intensità luminosa incidente sul LDR e una “Digital Out” in PWM per il LED



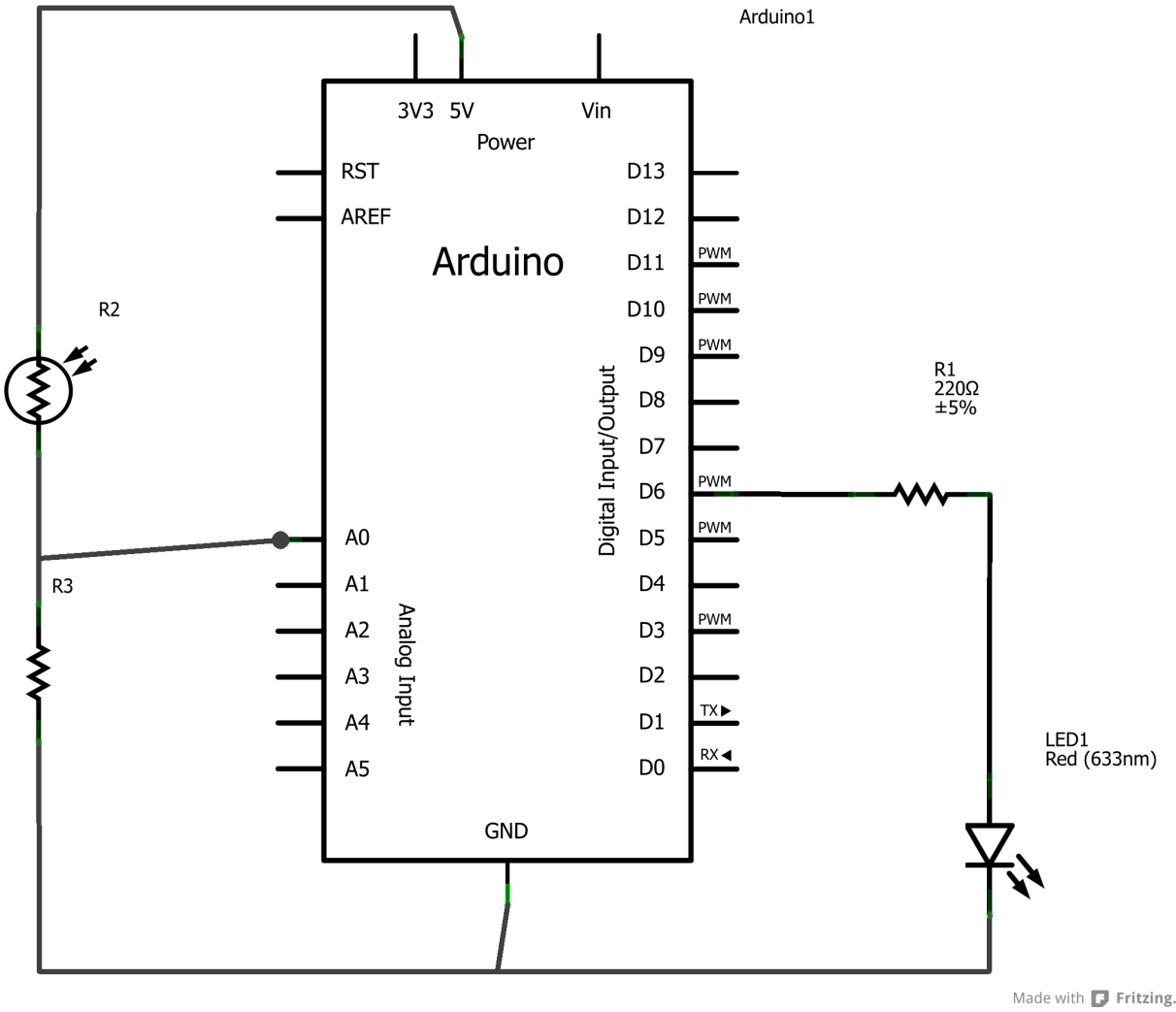
Una porta “Analog In” viene letta con la funzione:

```
analogRead(potPin);
```

che riporta un valore intero tra 0 e 1023 proporzionale al valore della tensione applicata alla porta nel range 0, 5V

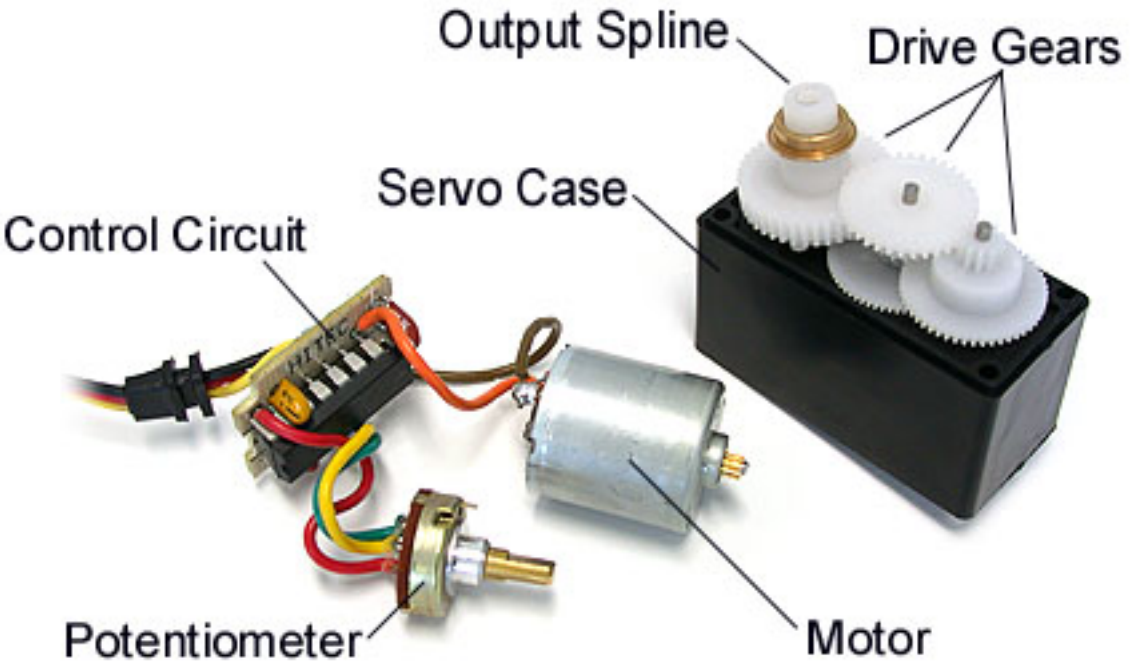
Arduino

FadeLight



Arduino

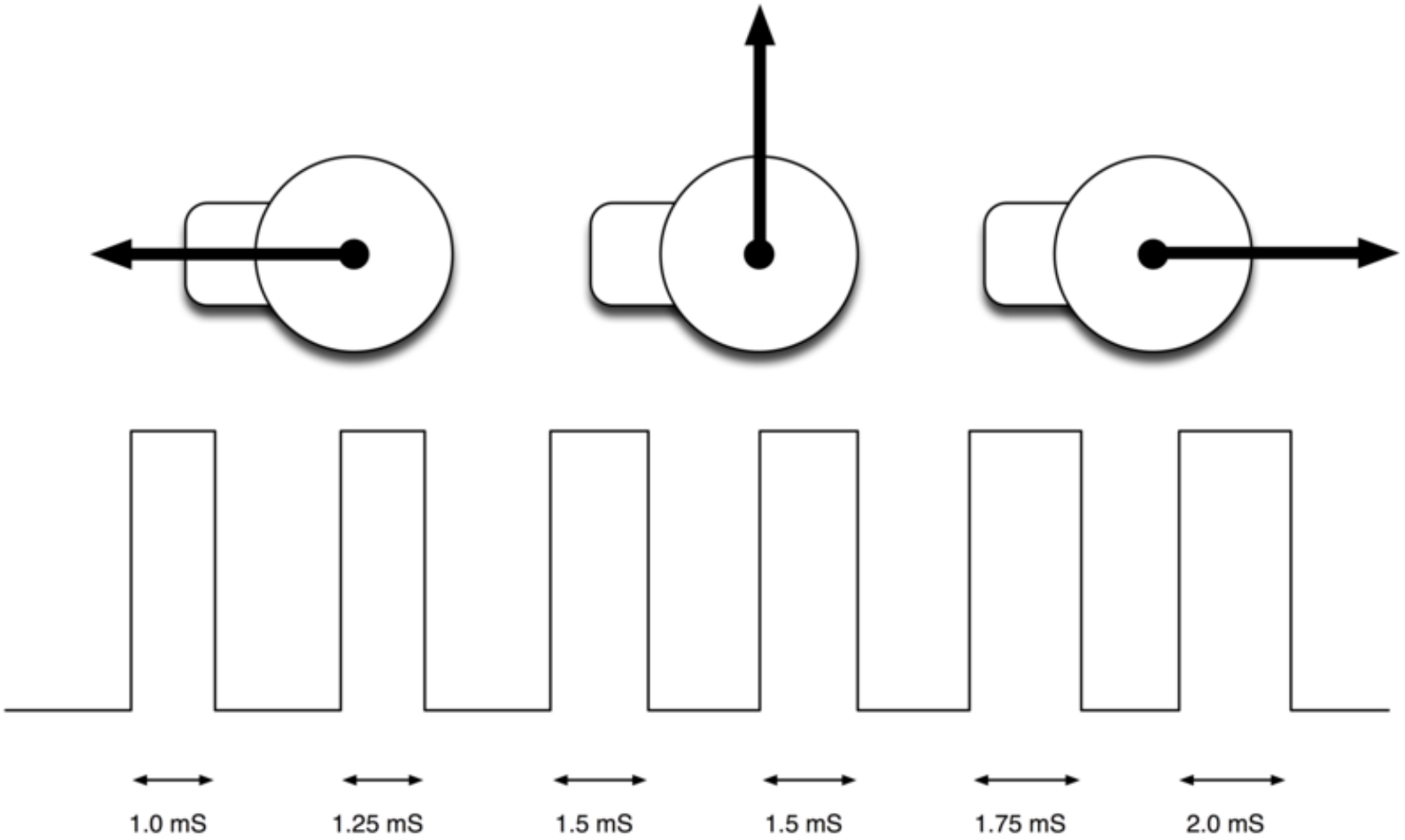
Il servomotore



All'interno di un servomotore troviamo un motore elettrico DC, gli ingranaggi per la demoltiplica, un potenziometro per misurare la posizione e l'elettronica per il controllo della posizione.

Arduino

Il servomotore

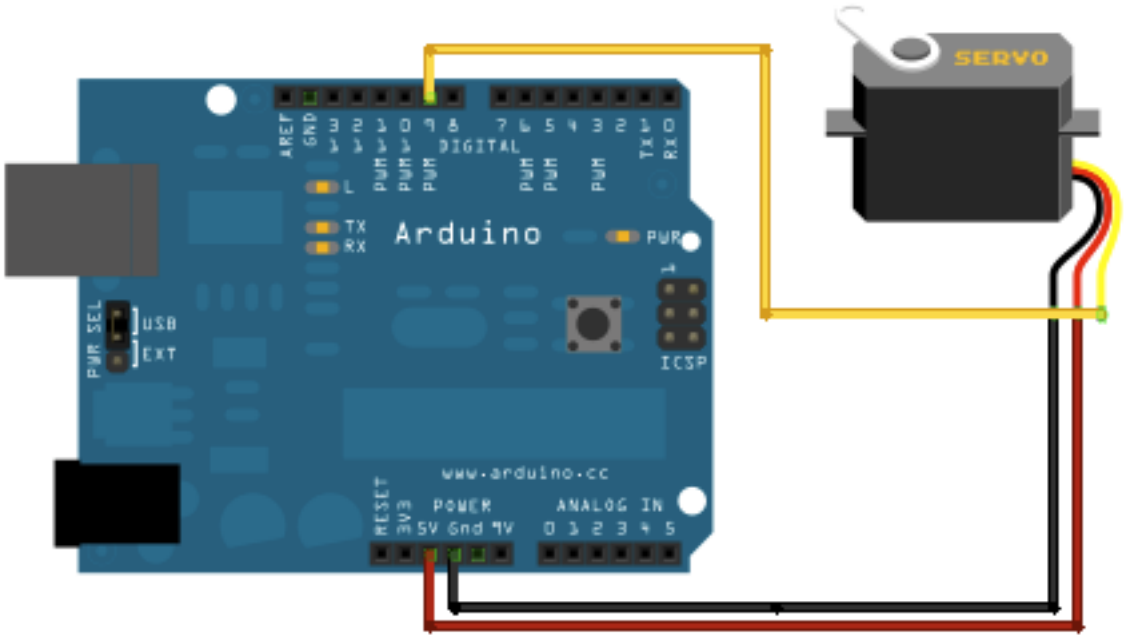


La posizione del servomotore viene impostata dalla lunghezza di un impulso. Il servomotore si aspetta di ricevere un impulso ogni 20ms.

Arduino

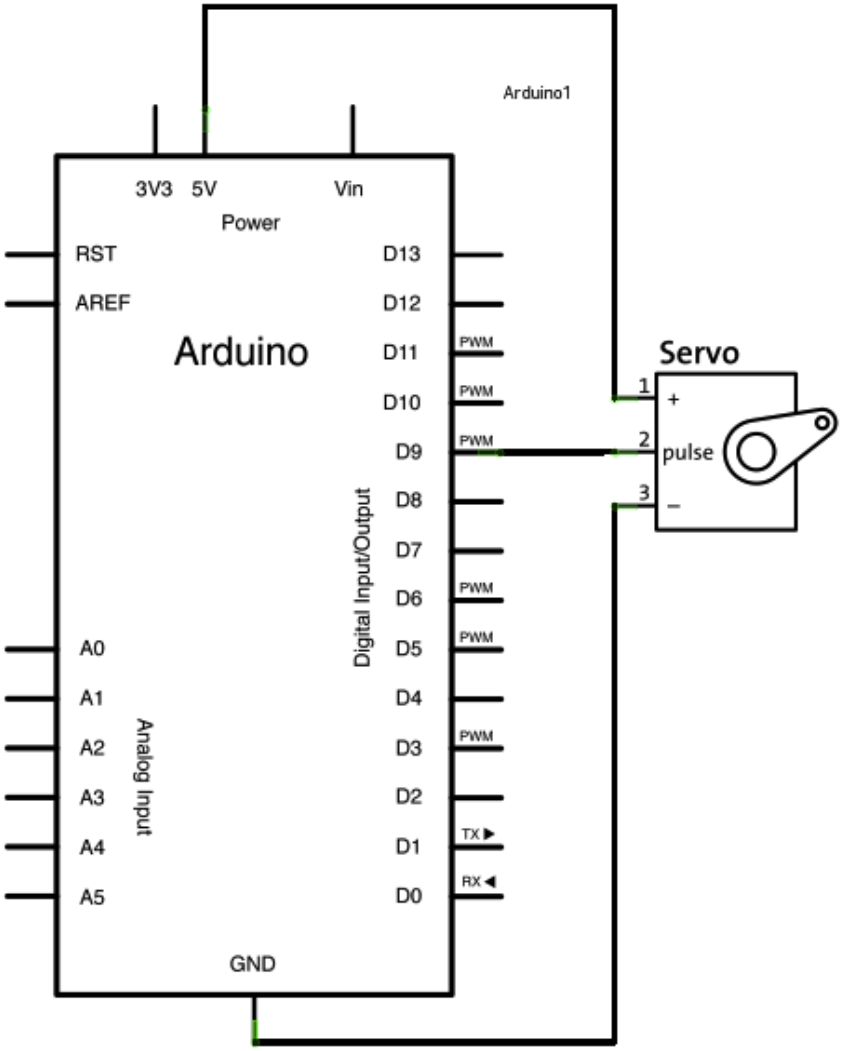
Sweep

Gira l'albero del servomotore avanti e indietro di 180°

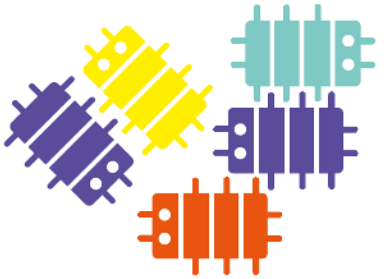


Arduino

Sweep



Arduino
Sweep

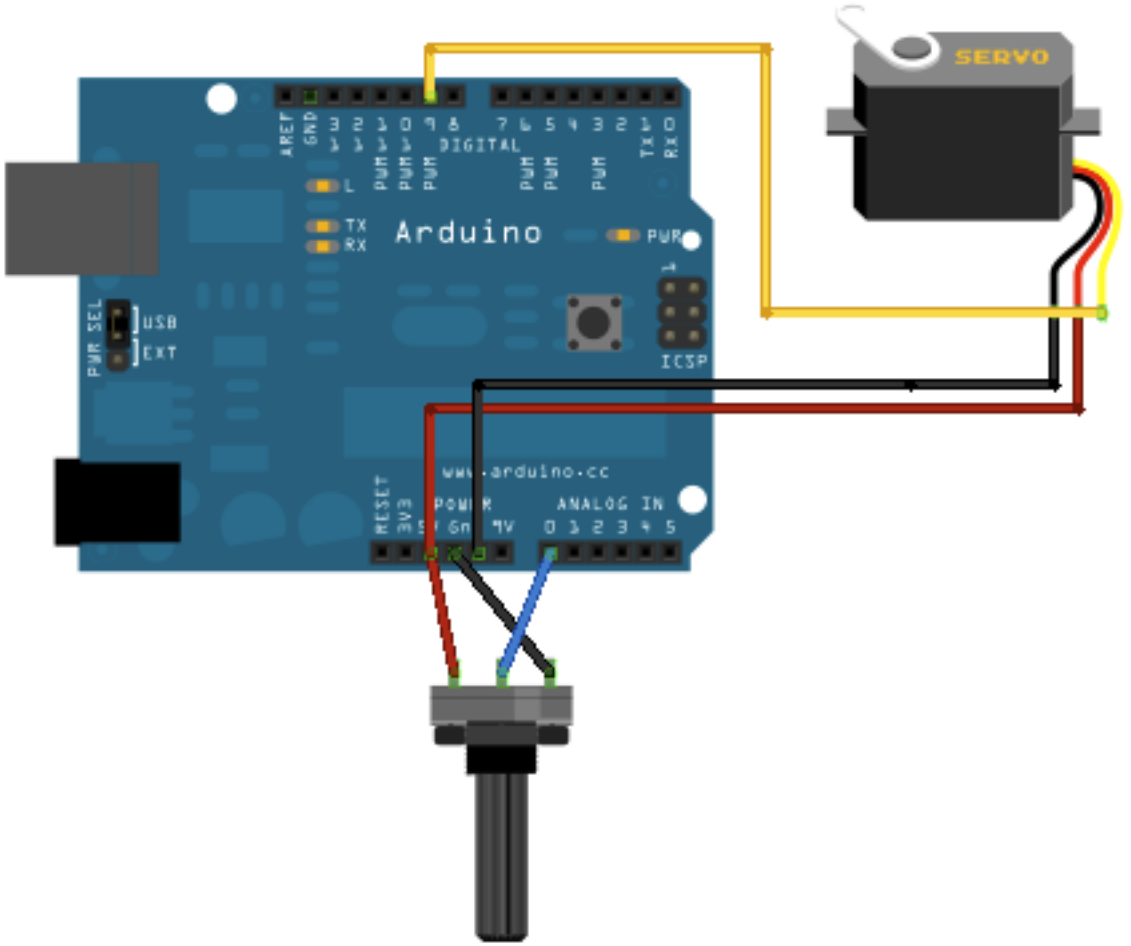


```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
                // a maximum of eight servo objects can be created

int pos = 0;    // variable to store the servo position

void setup()
{
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop()
{
  for(pos = 0; pos < 180; pos += 1)  // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for(pos = 180; pos>=1; pos-=1)     // goes from 180 degrees to 0 degrees
  {
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

Arduino

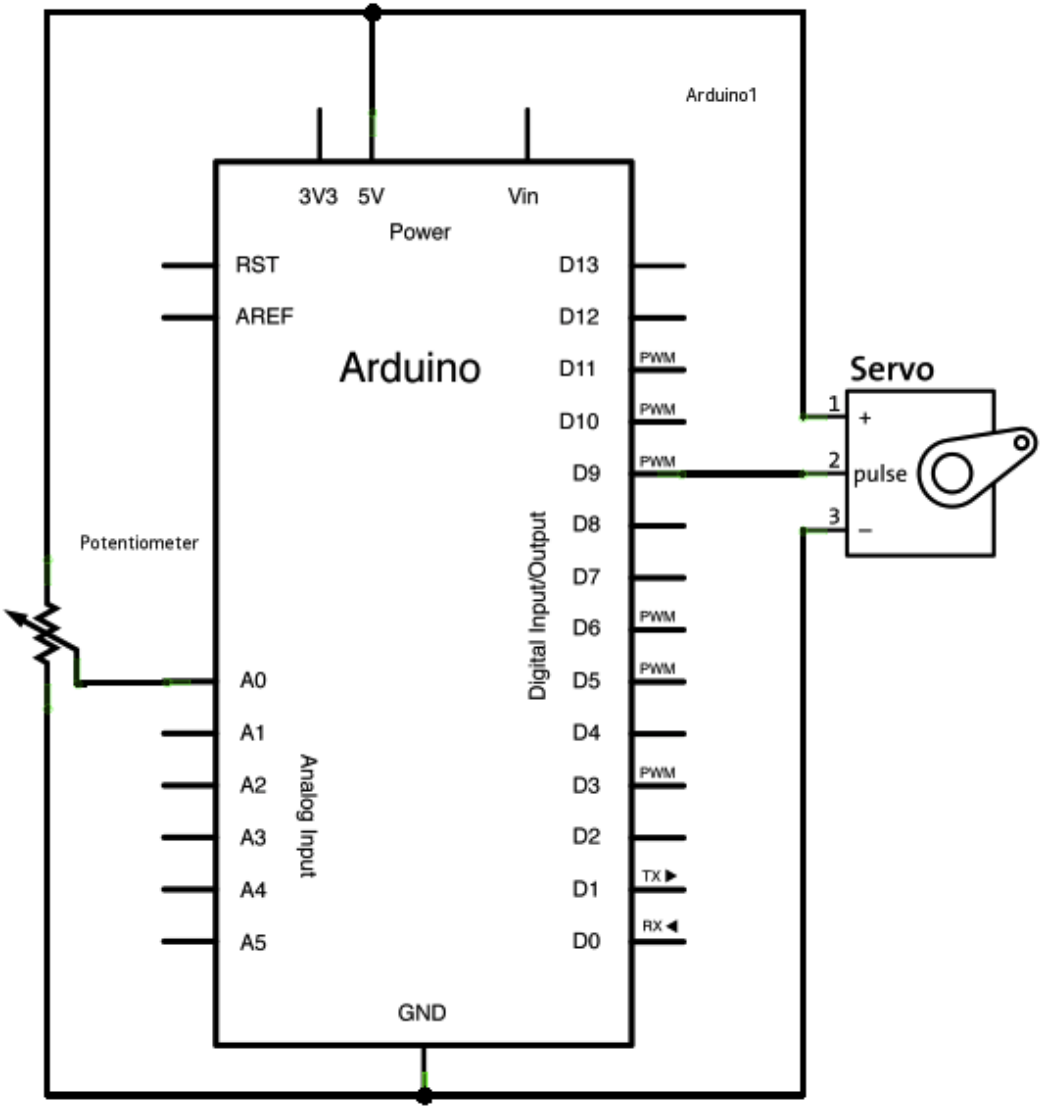
Knob

Controlla la posizione del servomotore con un potenziometro

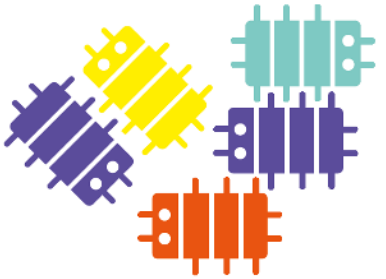


Arduino

Knob



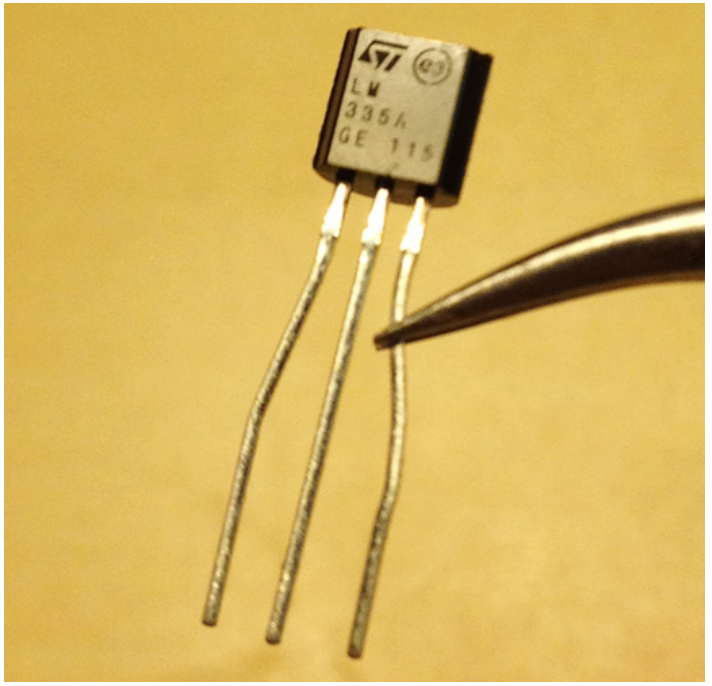
Arduino
Knob

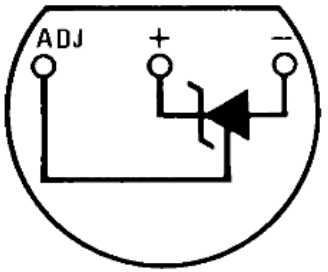


```
// Controlling a servo position using a potentiometer (variable resistor)
// by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>

#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup()
{
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop()
{
  val = analogRead(potpin);            // reads the value of the potentiometer
  val = map(val, 0, 1023, 0, 179); // scale it to use it with the servo
  myservo.write(val);  // sets the servo position according to the scaled value
  delay(15);                           // waits for the servo to get there
}
```

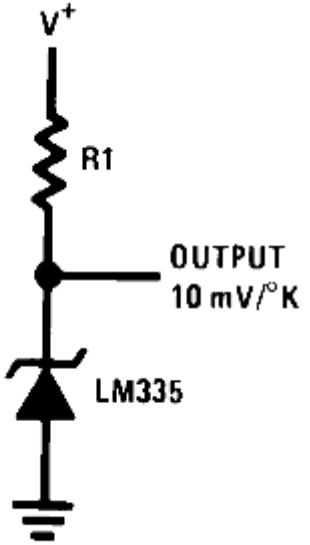
Arduino

Il sensore di temperatura LM335





TO-92 package
Bottom view

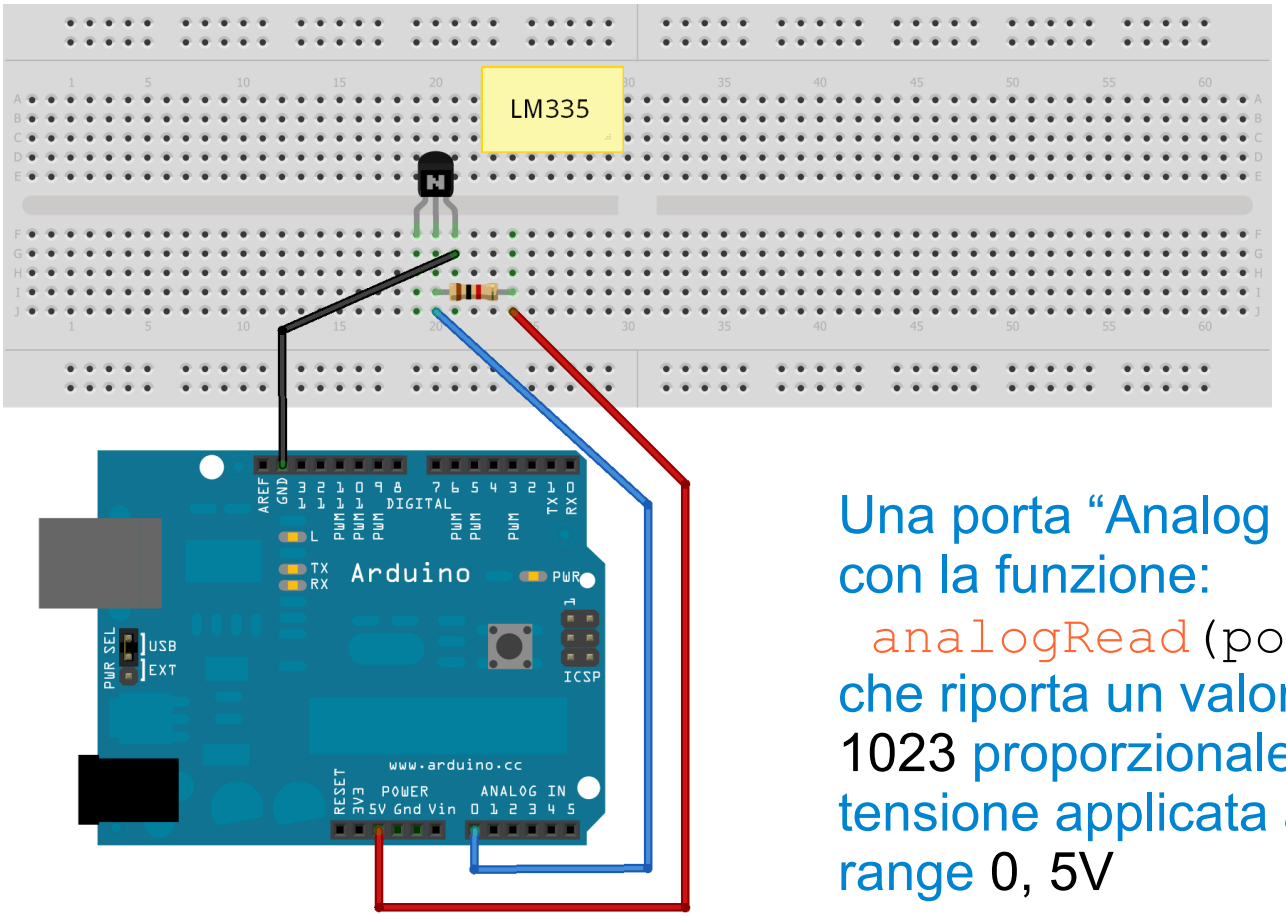


Il componente LM335 è un sensore di temperatura economico con una precisione di circa +/- 3°C. Essenzialmente è un diodo zener con una tensione di polarizzazione inversa proporzionale alla temperatura.

Arduino

Temperature

Usa una porta “Analog In” per leggere un valore in tensione proporzionale alla temperatura



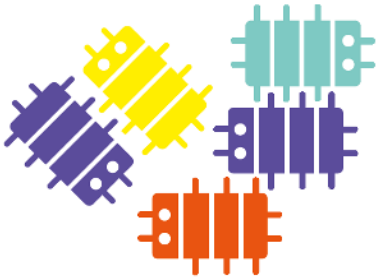
Una porta “Analog In” viene letta con la funzione:

```
analogRead(potPin);
```

che riporta un valore intero tra 0 e 1023 proporzionale al valore della tensione applicata alla porta nel range 0, 5V

Arduino

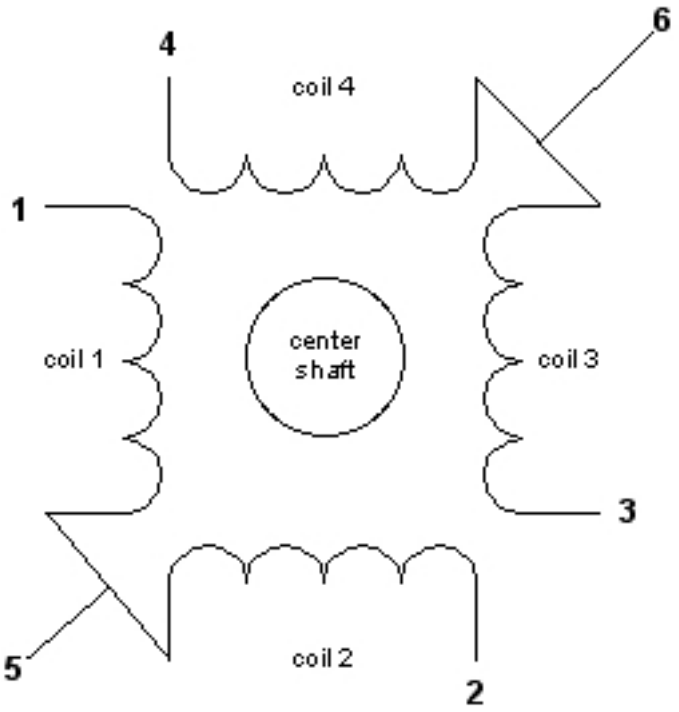
Temperature

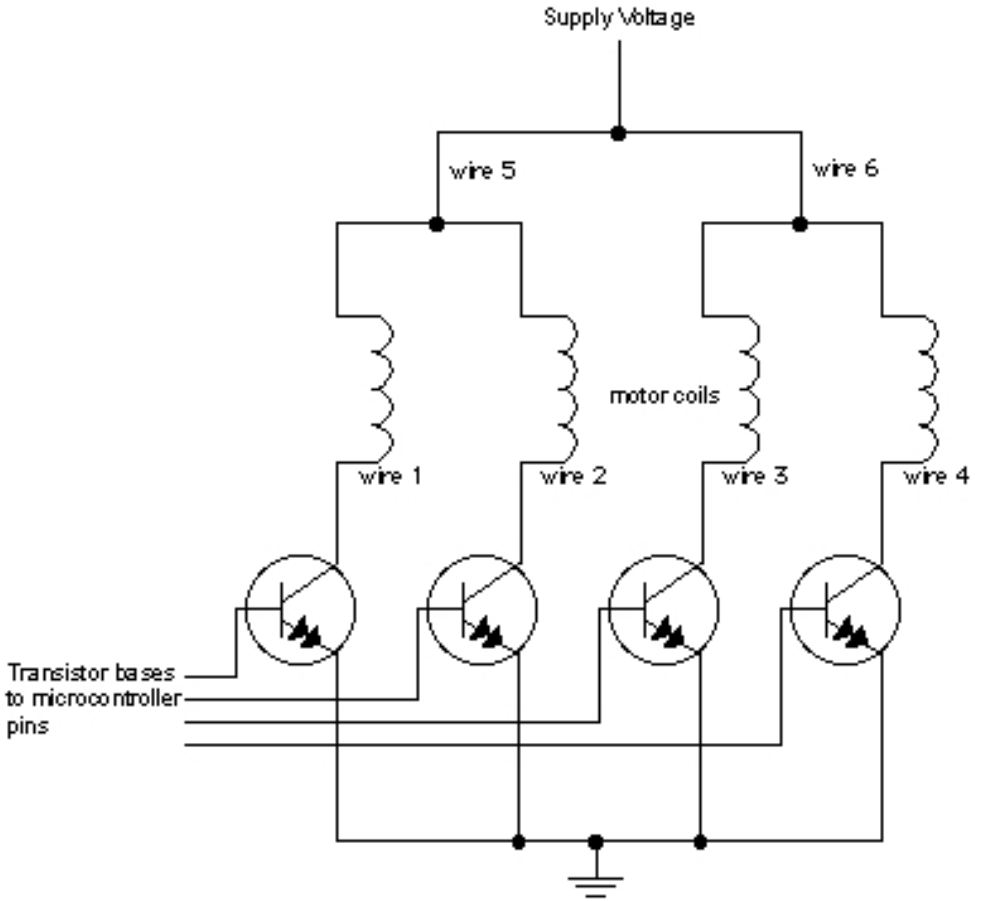


```
/*
  LMM335 temperature sensor connected to pin 0, prints the result to the serial
  monitor
*/

void setup() {
  Serial.begin(9600);
}

void loop() {
  long sensorValue = analogRead(A0);
  long tempKelvin = sensorValue*500/1023;
  long tempCelsius = tempKelvin-273;
  Serial.print("Valore Sensore: ");
  Serial.print(sensorValue, DEC);
  Serial.print("   Teperatura Kelvin: ");
  Serial.print(tempKelvin, DEC);
  Serial.print("   Temperatura Celsius: ");
  Serial.println(tempCelsius,DEC);
  delay(1000);
}
```

Arduino

Il motore passo-passo

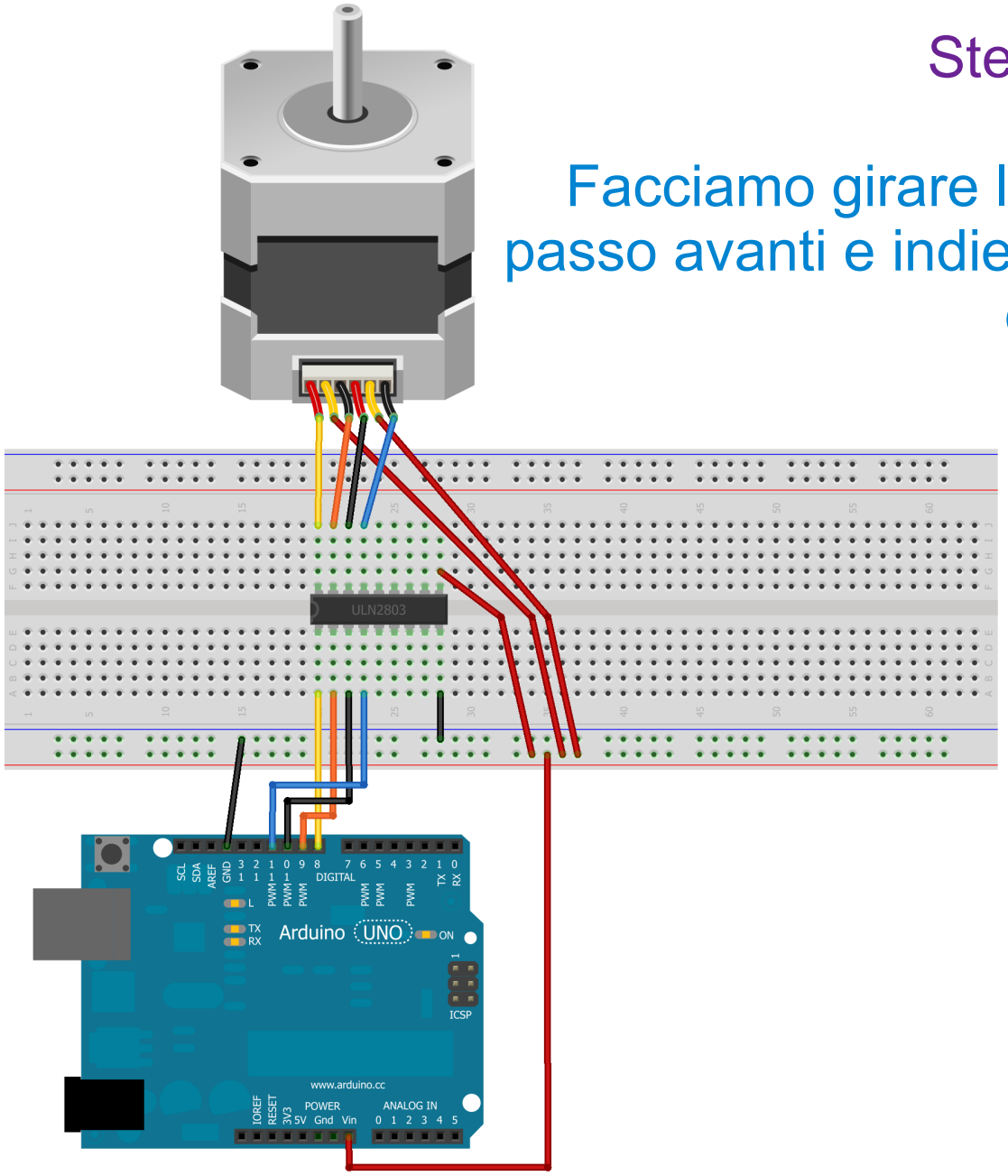




Il motore passo-passo è un motore controllato da una serie di elettromagneti. Sull'albero centrale ci sono una serie di magneti, le bobine che circondano l'albero sono percorse da corrente (una si una no) con la conseguente creazione di campi magnetici che attraggono o respingono i magneti sull'albero provocando la rotazione del motore.

Arduino

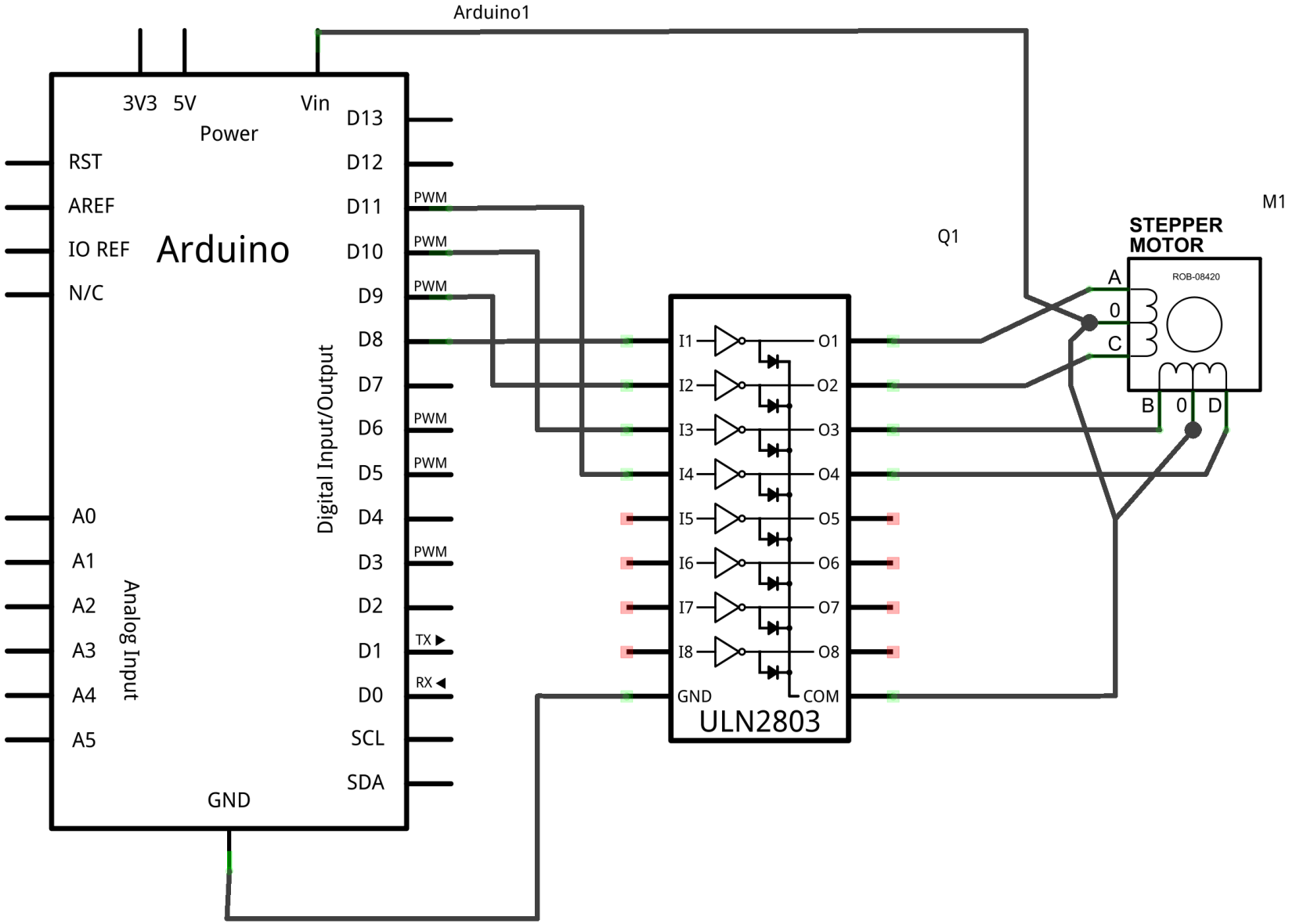
StepperMotor



Facciamo girare l'albero del motore passo passo avanti e indietro di un numero prefissato di passi

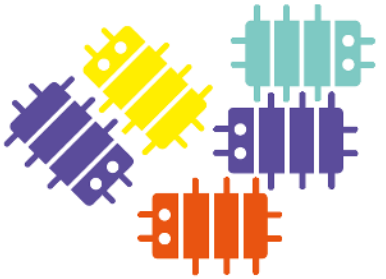
Arduino

StepperMotor



Arduino

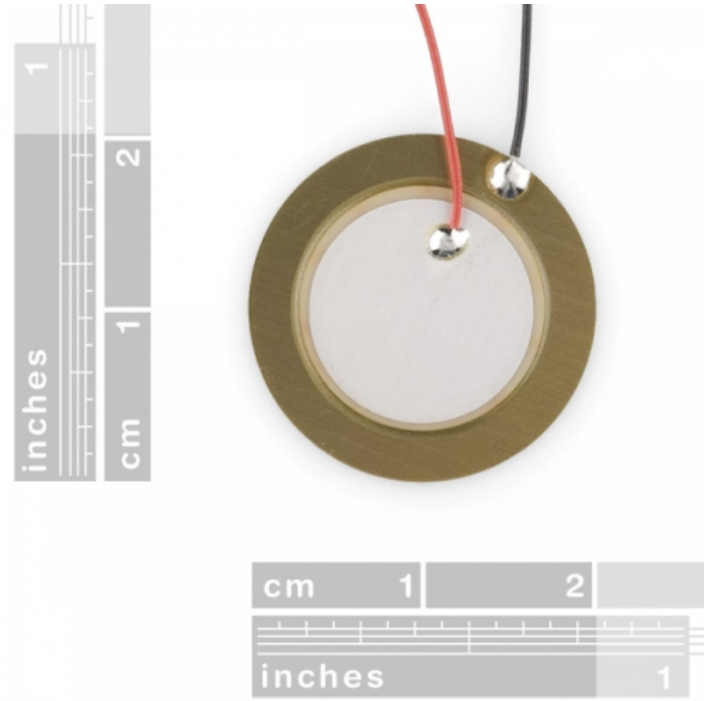
StepperMotor



```
#include <Stepper.h>

// create an instance of the stepper class, specifying
// the number of steps of the motor and the pins it's
// attached to
Stepper stepper(100, 8, 9, 10, 11);

void setup()
{
  // set the speed of the motor to 30 RPMs
  stepper.setSpeed(30);
}

void loop()
{
  stepper.step(100);
  delay(500);
  stepper.step(-100);
  delay(500);
}
```

Arduino

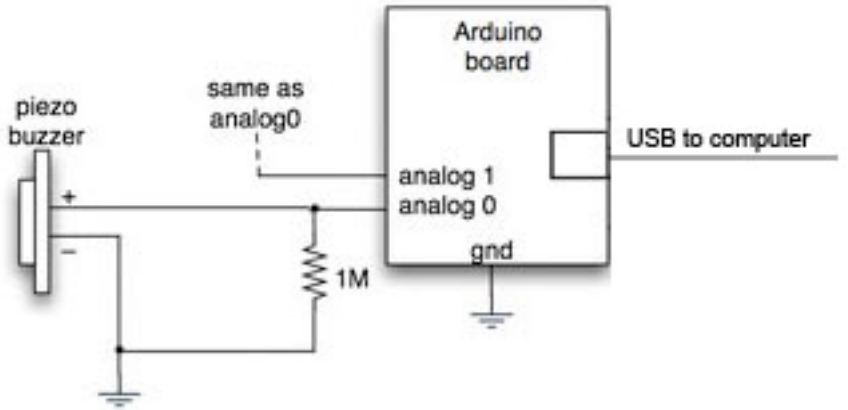
Il trasduttore piezoelettrico



È realizzato con materiali che sfruttano l'effetto piezoelettrico: quando viene applicata una pressione esterna cariche di segno opposto si posizionano sulle facce opposte e se collegate ad un circuito viene generata una corrente. Al contrario se viene applicata una differenza di potenziale sulle due facce il materiale di espande e si contrae.

Arduino

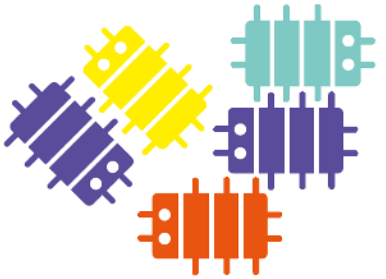
Ardrumo



Per i dettagli vedere: http://inivent.com/ardrumo/

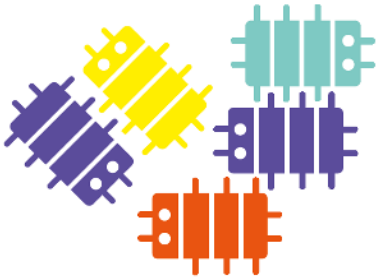
Arduino

Ardrumo



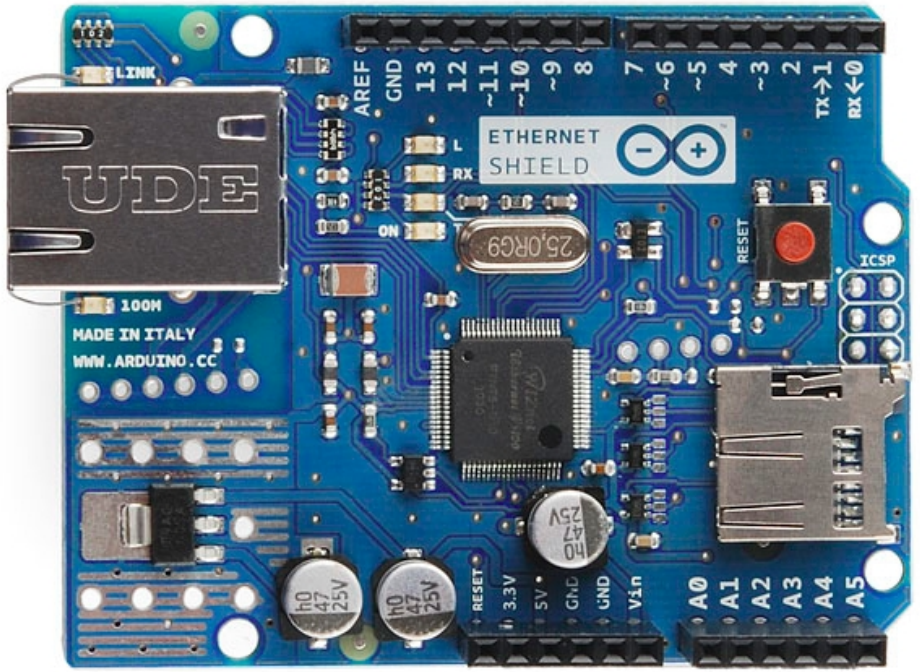
```
#define LEDPIN      13      // status LED pin
#define PIEZOTHRESHOLD 5    // analog threshold for piezo sensing
#define PADNUM 6            // number of pads

int val;

void setup() {
  pinMode(LEDPIN, OUTPUT);
  Serial.begin(57600);    // set serial output rate
}

void loop() {
  for(int i = 0; i < PADNUM; i++) {
    val = analogRead(i);
    if( val >= PIEZOTHRESHOLD ) {
      digitalWrite(LEDPIN,HIGH);  // indicate we're sending MIDI data
      Serial.print(i);
      Serial.print(",");
      Serial.print(val);
      Serial.println();
      digitalWrite(LEDPIN,LOW);
    }
  }
}
```

Arduino



Oltre alle interfacce e porte I/O disponibili in Arduino è possibile aggiungere delle schede che servono ad espandere le funzionalità e le possibilità di connessione. Queste schede, che devono rispettare un formato ben preciso, vengono chiamate “Shields”.

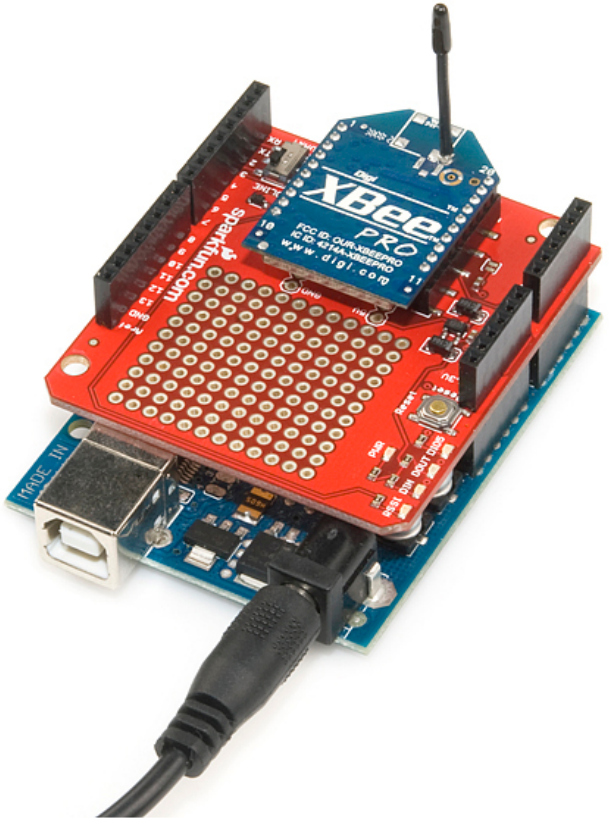
Arduino



Ethernet Shield (fonte: arduino.cc)

L'Ethernet Shield permette di connettere Arduino a reti LAN e ad Internet. É realizzato usando il chip Wiznet W5100 che oltre alla parte fisica implementa in hardware il protocollo TCP/IP.

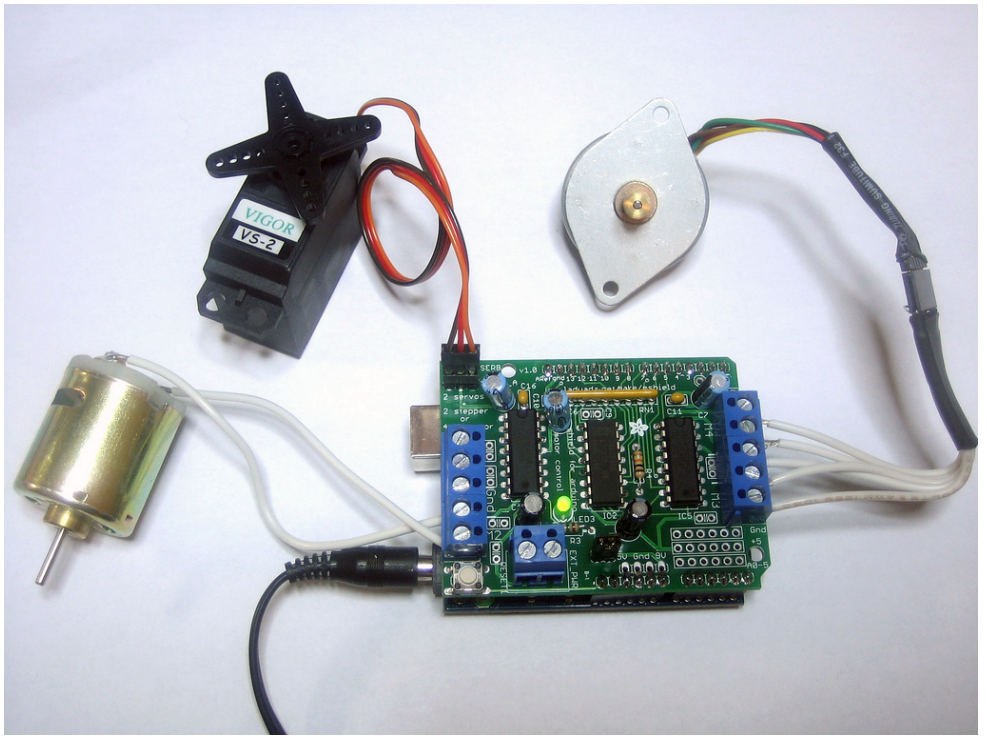
Arduino



Xbee Shield (fonte: sparkfun.com)

Lo Shield Xbee permette ad Arduino di comunicare wireless usando il protocollo Zigbee.

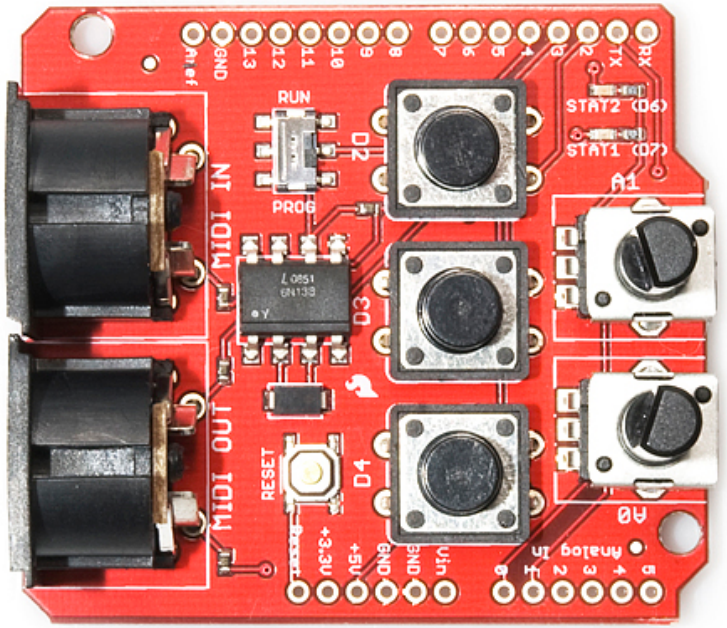
Arduino



Motor/Stepper/Servo Shield (fonte: adafruit.com)

Con questo shield è possibile collegare ad Arduino fino a 2 servo motori da 5V, 4 motori DC bidirezionali e 2 motori passo passo (unipolari o bipolari)

Arduino

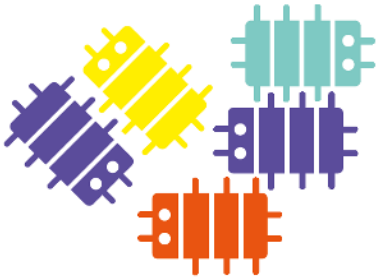


MIDI Shield (fonte: sparkfun.com)

Il MIDI shield permette ad Arduino di controllare sintetizzatori, sequencer e altri strumenti musicali e di ricevere ed elaborare messaggi nel protocollo MIDI.

Arduino

Link utili



Sito principale di Arduino:
http://arduino.cc/

Importante la sezione “Reference”
http://arduino.cc/en/Reference/HomePage

==========

“Adafruit” – sito e-commerce per tutto quello che riguarda Open Hardware, Arduino e dintorni:
http://adafruit.com/

Molto interessanti le sezioni “TUTORIALS” e “BLOG”:
http://learn.adafruit.com/

http://www.adafruit.com/blog/

==========

I disegni delle breadbord e degli schemi che abbiamo visto sono stati realizzati con “Fritzing”:
http://fritzing.org/