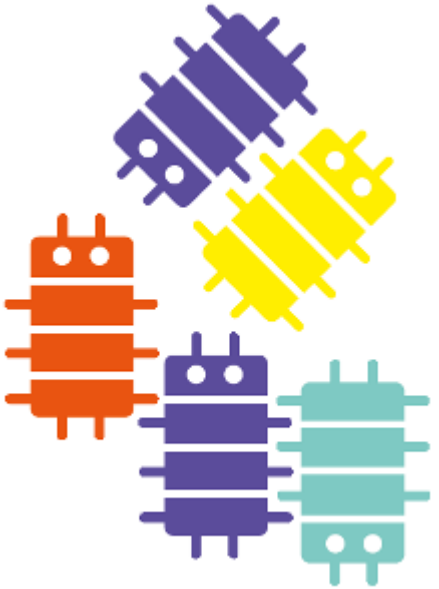
Arduino

Laboratorio Arduino Base I

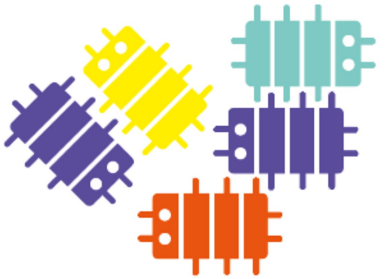


HackLab Terni

Laboratorio aperto a tutti di
elettronica, scienza e arte.

hacklabterni.org

Arduino



Cos'è Arduino?

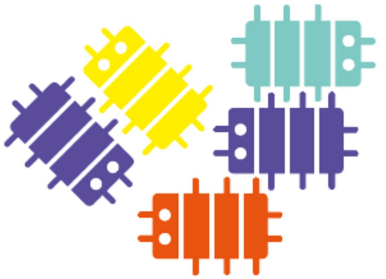
Arduino è una piattaforma di prototipazione elettronica open-source basata su microcontrollore pensata per rendere più accessibile l'uso dell'elettronica nei progetti multidisciplinari.

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

Microcontrollore
- ATMega328

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

Porte di Input Analogico
- 6

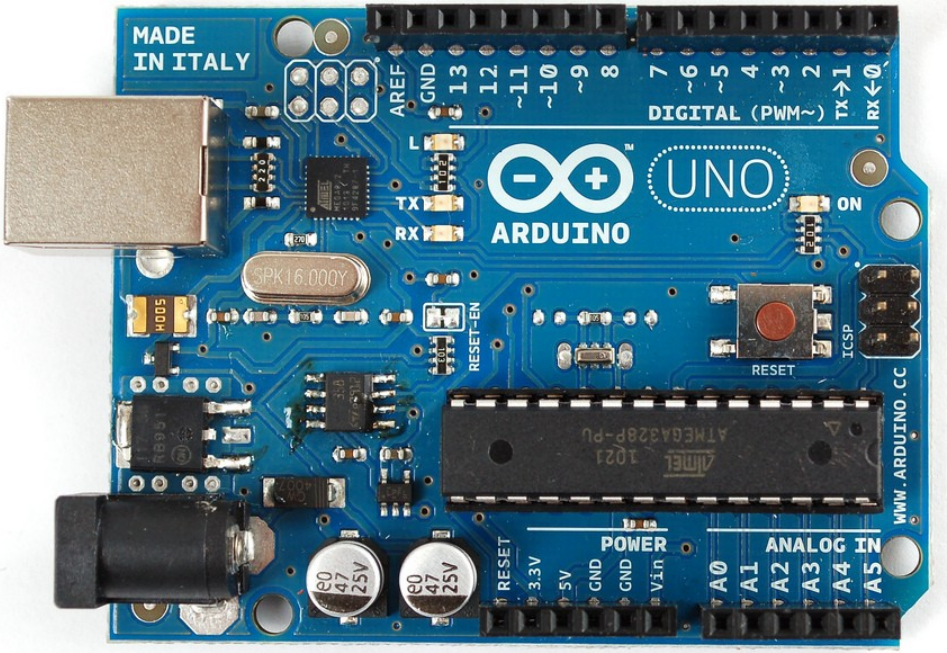
Memoria Flash
- 32KB (0.5KB bootloader)

SRAM
- 2KB

EEPROM
- 1KB

Velocità del Clock
- 16MHz

(fonte: arduino.cc)



Arduino Uno

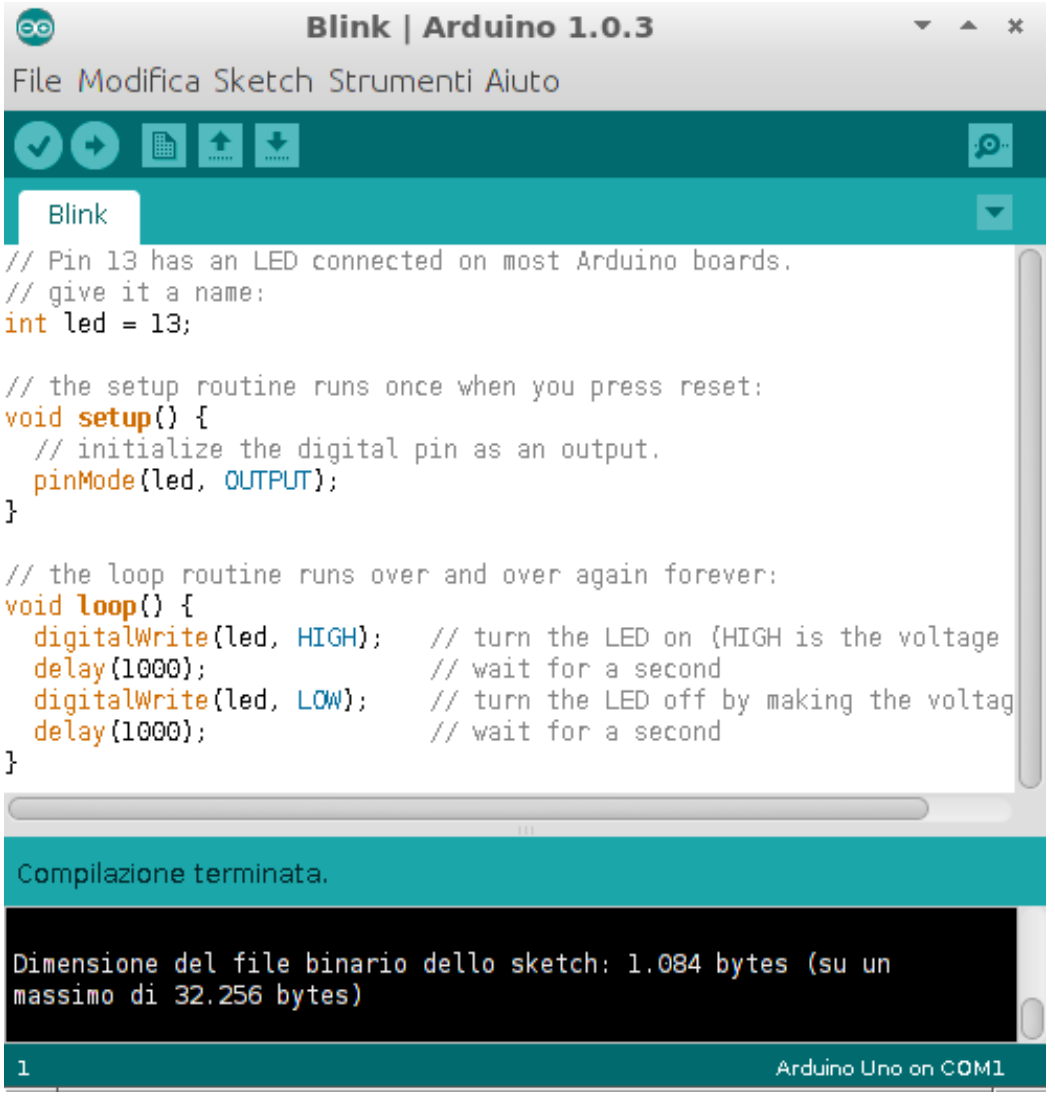
Arduino

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.

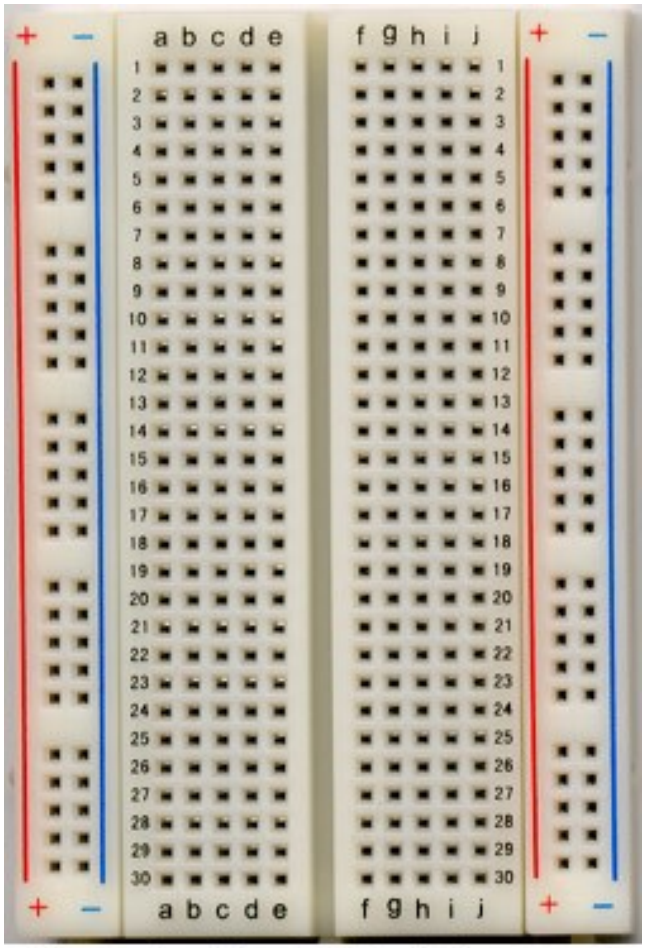
Nella parte in alto si trovano alcuni pulsanti con i quali è possibile compilare, verificare e caricare nel microcontrollore di Arduino il programma scritto.

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

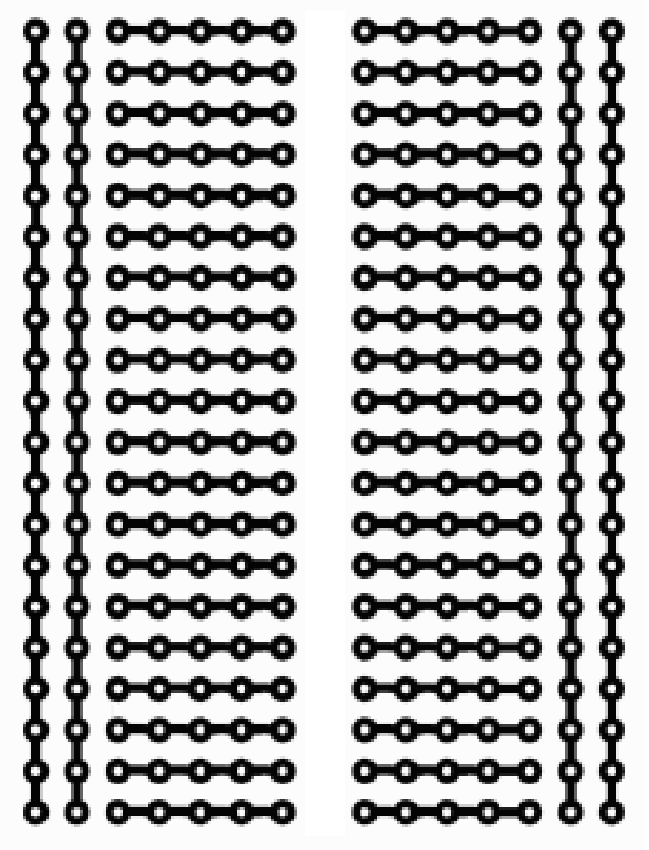


Arduino

La breadboard



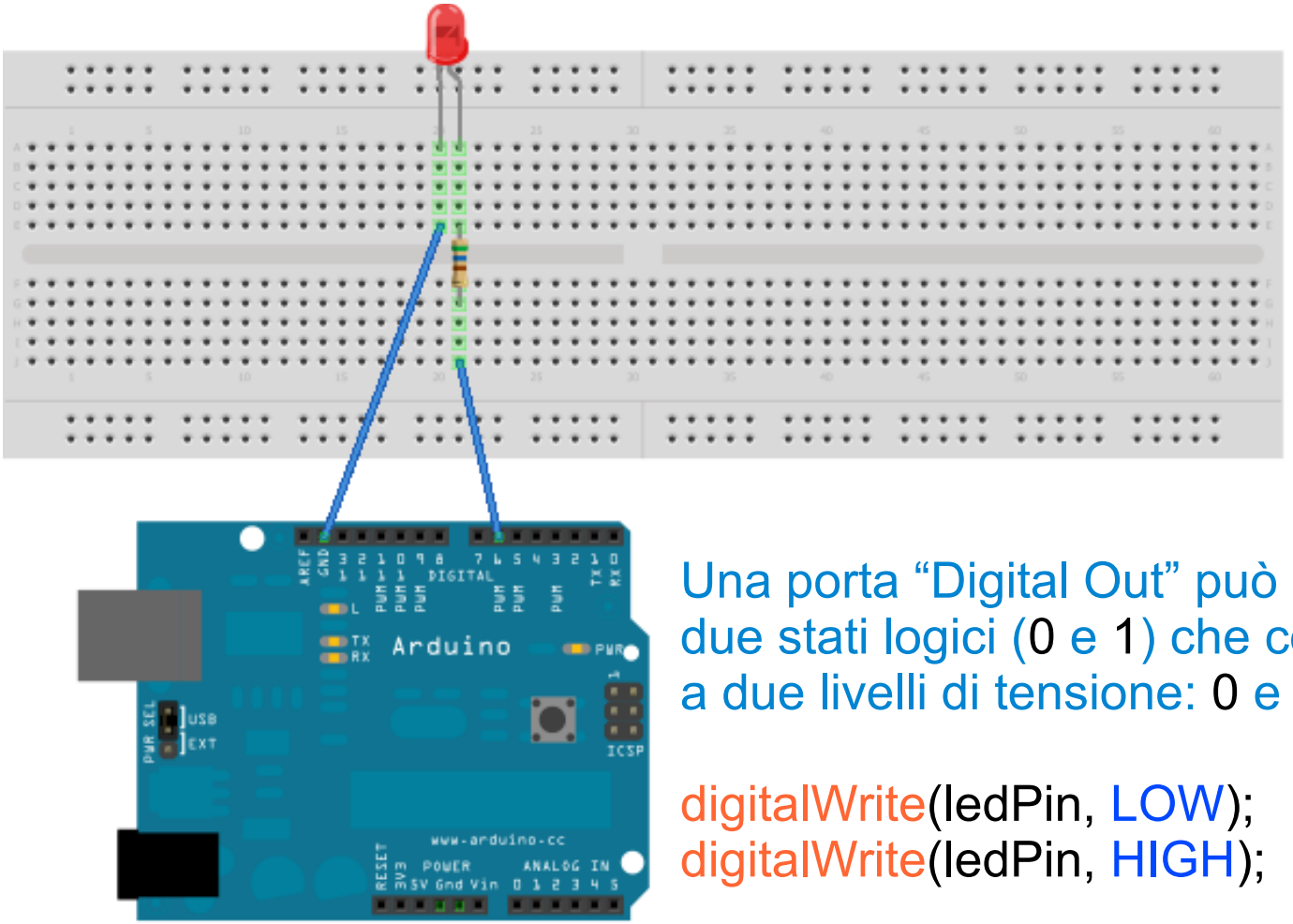
Video and Website © 2004 ClarkZapper.net



Arduino

Blink

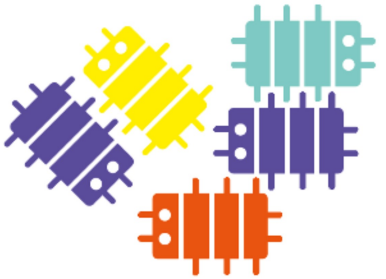
Usa una porta “Digital Out” per far lampeggiare un LED



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



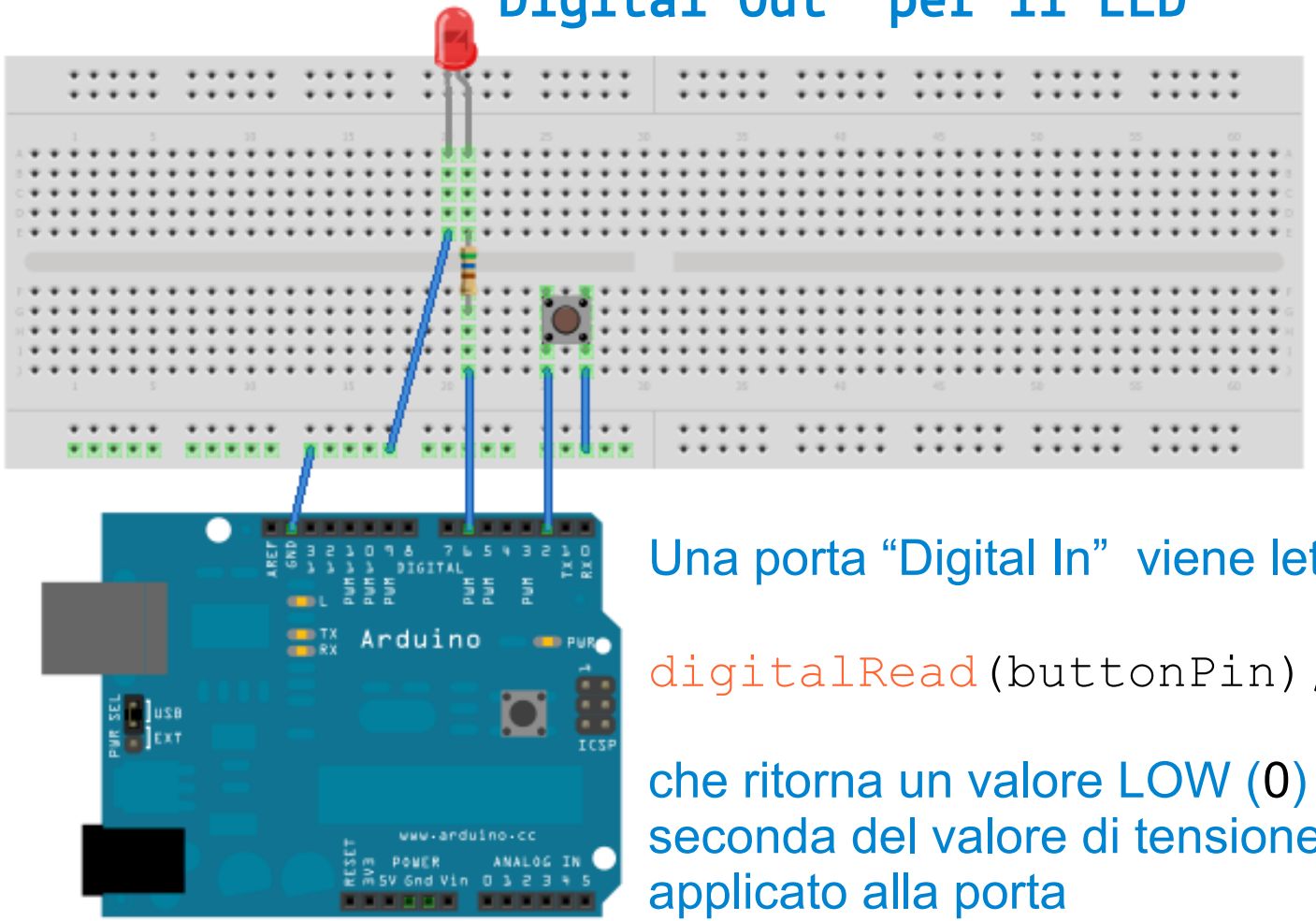
Blink

```
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

Button

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



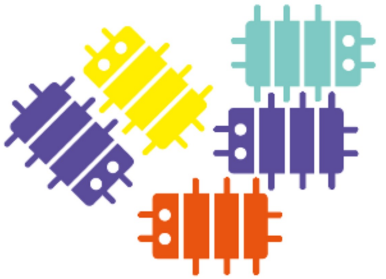
Una porta “Digital In” viene letta dalla funzione:

```
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

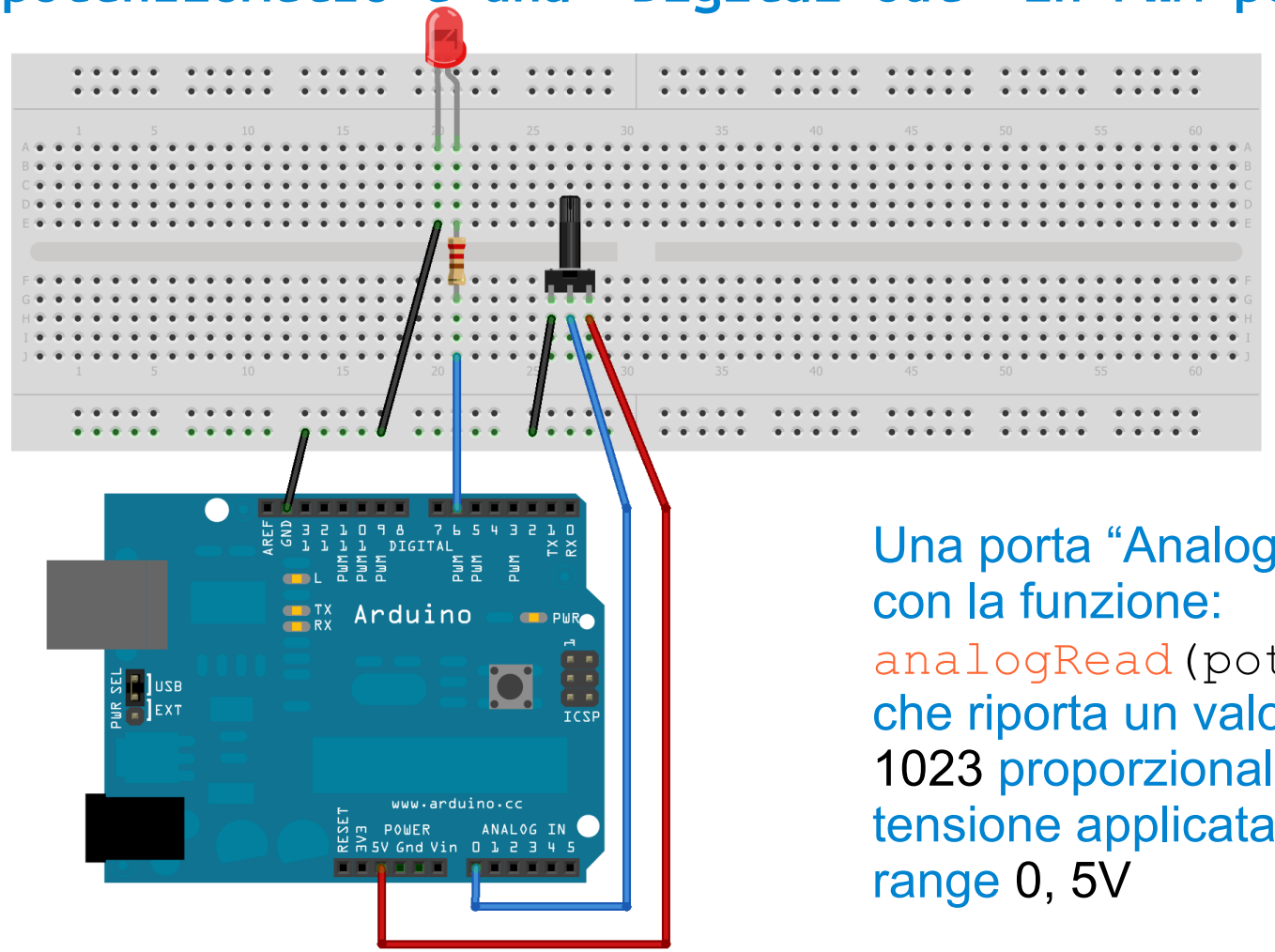


```
int buttonPin = 2;      // the number of the pushbutton pin
int ledPin = 6;         // the number of the LED pin
int buttonState = 0;    // variable for reading the pushbutton status

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



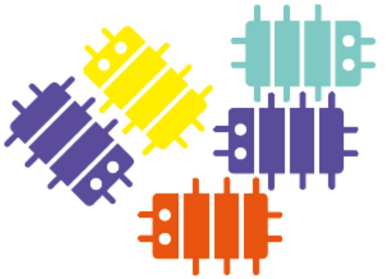
Una porta “Analog In” viene letta con la funzione:

```
analogRead(potPin);
```

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

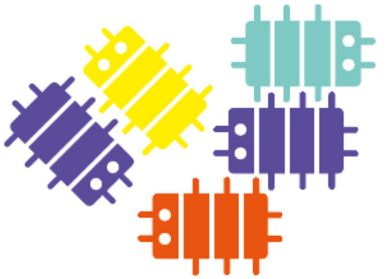
Arduino

FadePot

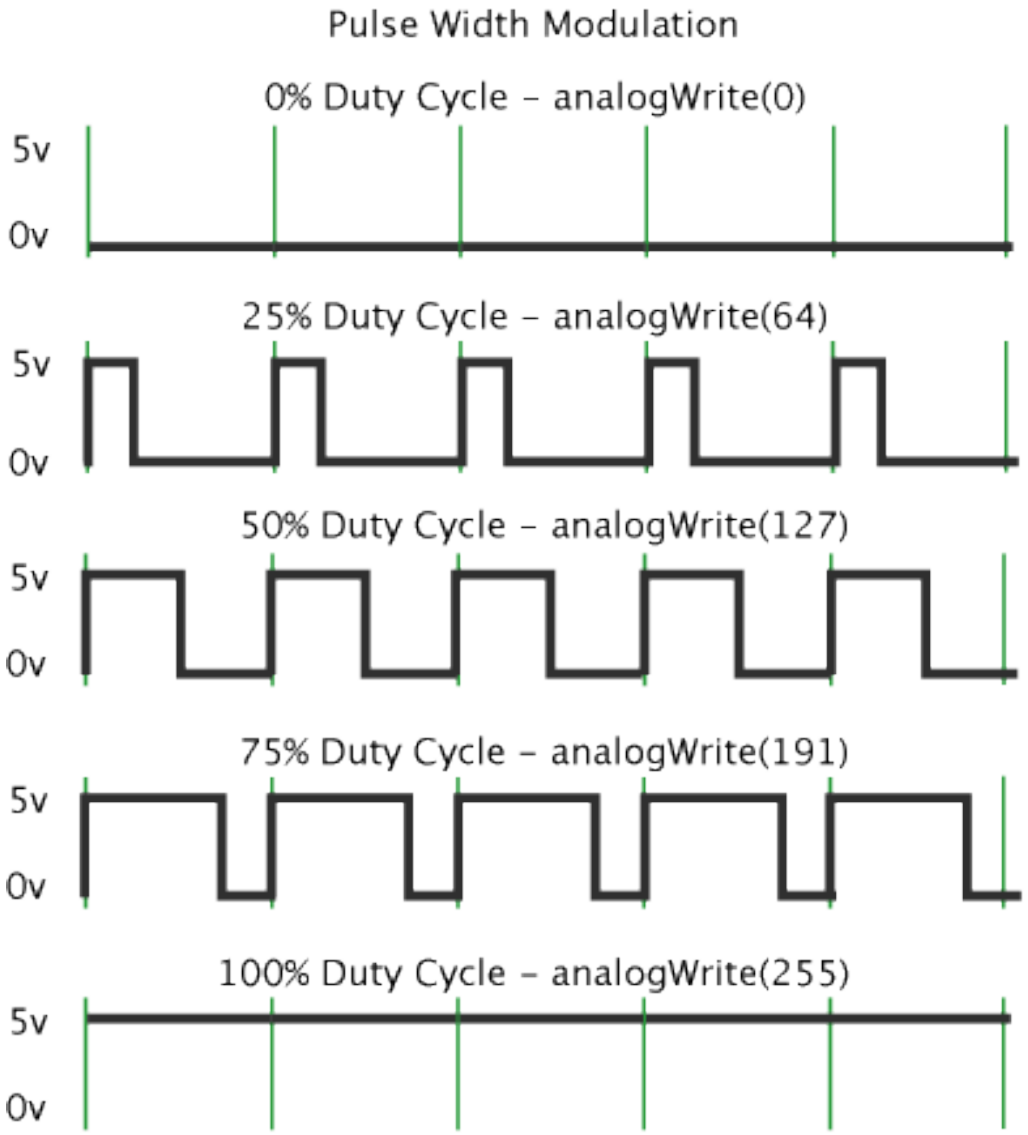


```
int ledPin =  6;       // the number of the LED pin
int potPin =  A0;      // analog input pin

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



PWM (Pulse Width Modulation)



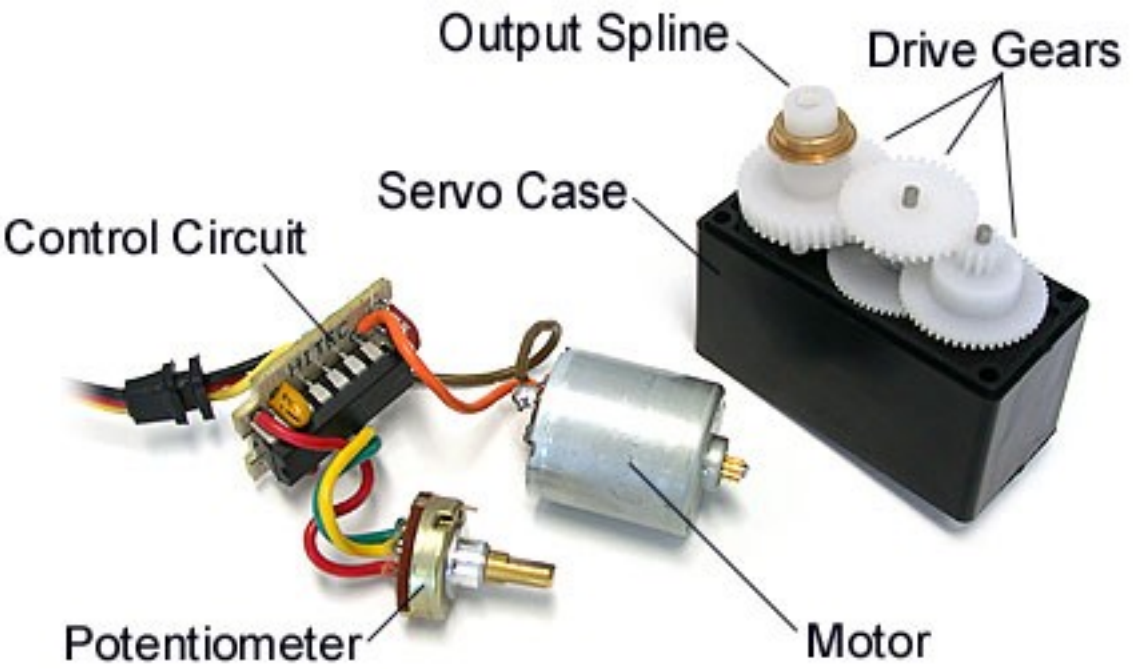
Per scrivere un valore analogico in una porta si usa la funzione:

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

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

Arduino

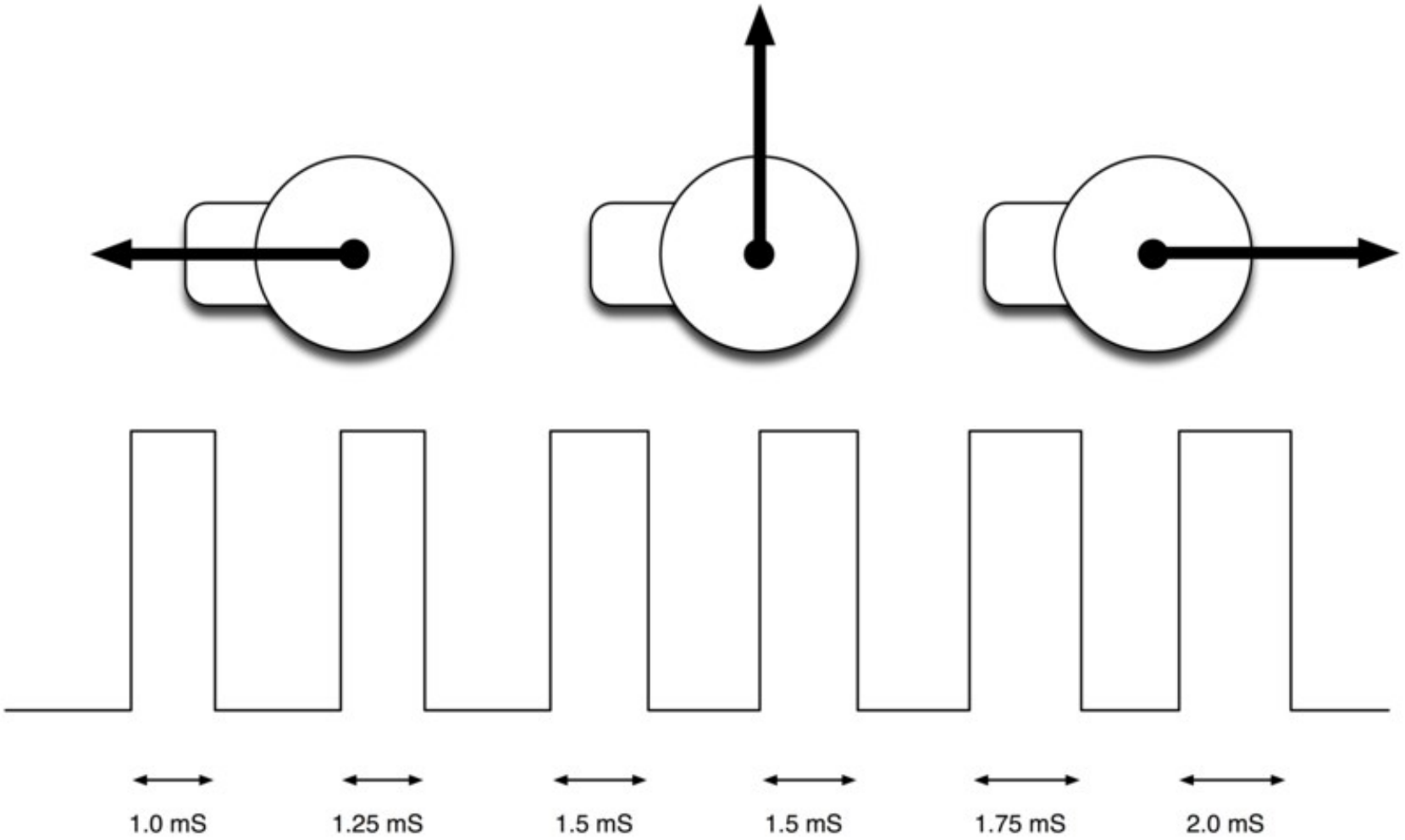
Il servomotore



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

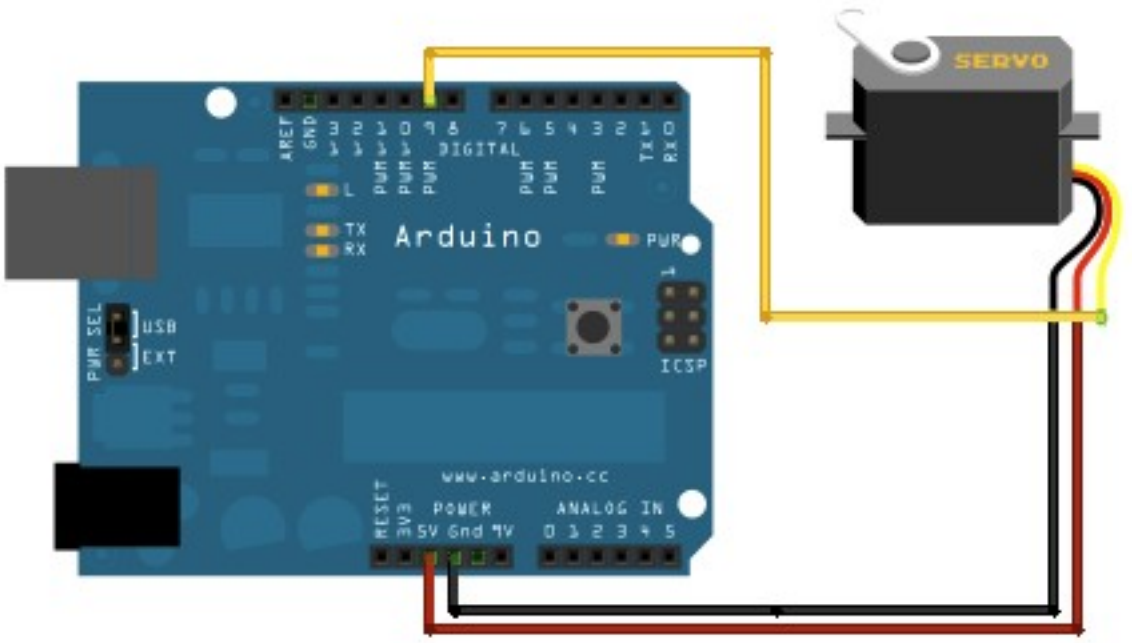


La posizione del servomotore viene impostata dalla lunghezza di un impulso. Il servomotore si aspetta di ricevere un impulso ogni 20ms.

Arduino

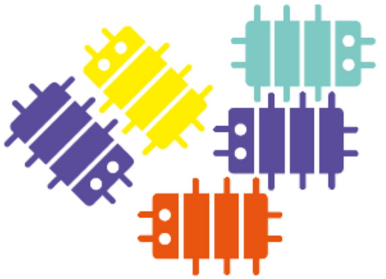
Sweep

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



Arduino

Sweep

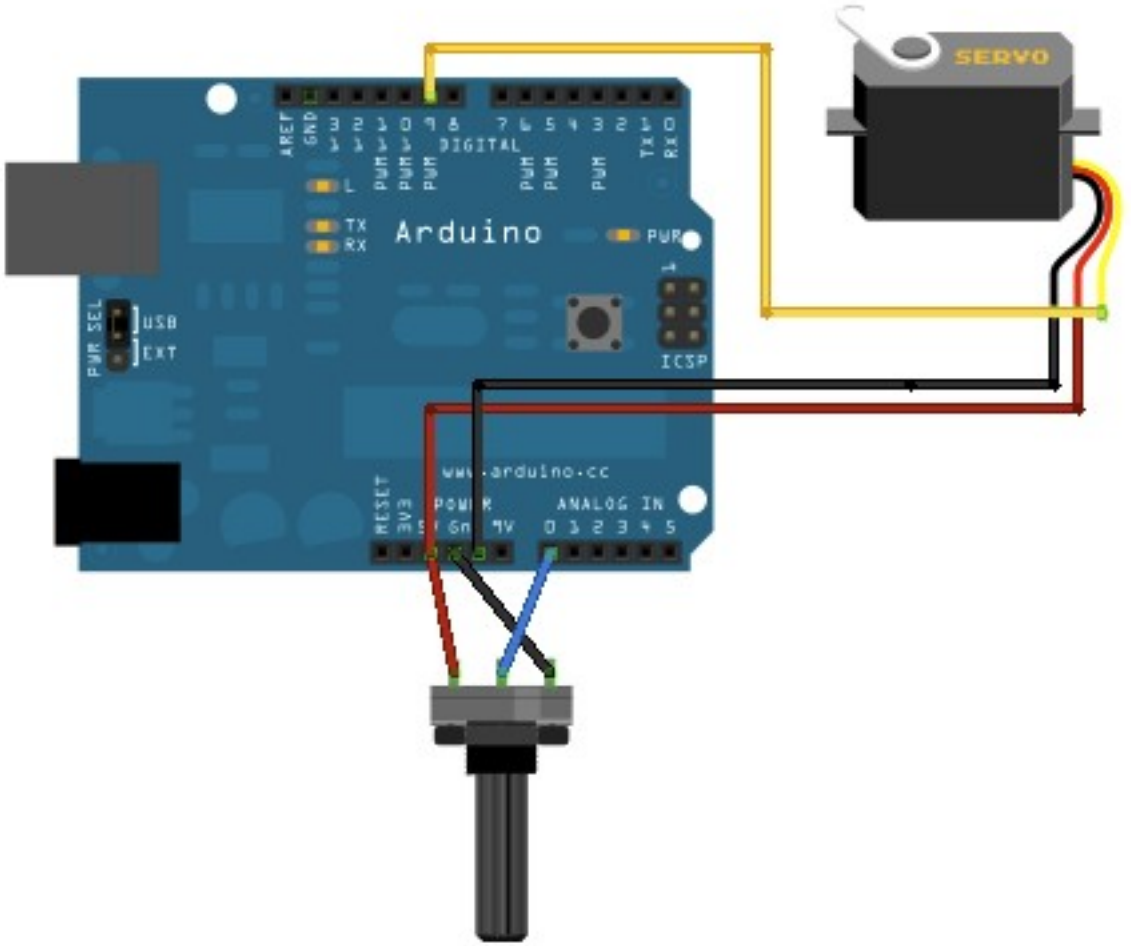


```
#include <Servo.h>

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

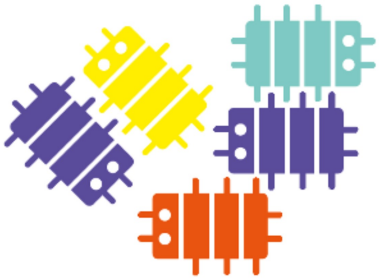
Knob

Controlla la posizione del servomotore con un potenziometro



Arduino

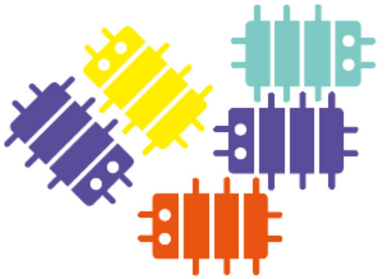
Knob



```
// 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

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/

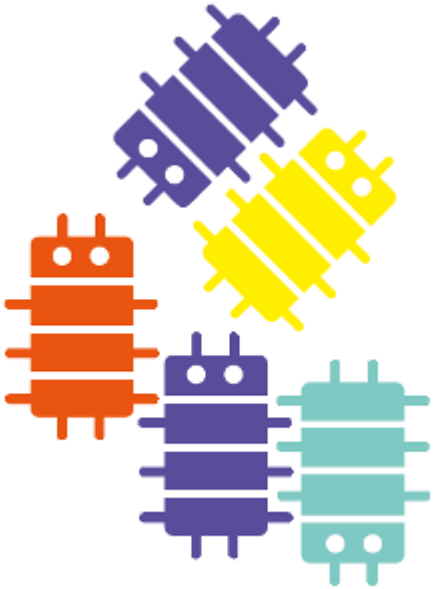
http://www.adafruit.com/blog/

==========

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

Arduino

Laboratorio Arduino Base II

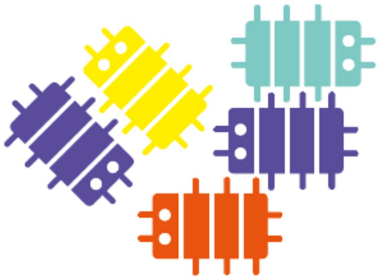


HackLab Terni

Laboratorio aperto a tutti di
elettronica, scienza e arte.

hacklabterni.org

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.

Elementi del linguaggio:

- Variabili: dichiarazione (associo il nome e il tipo e viene riservata la memoria) e inizializzazione (associo un valore iniziale - opzionale)

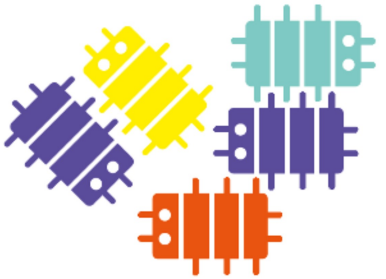
```
int buttonState = 0;
float temperature = 23.5;
char key = 'A';
```

- Assegnazione di valori o risultato di espressioni ad una variabile

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

Arduino

Programmazione



- **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);
}
```

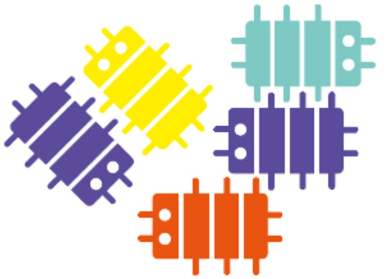
In Arduino ci sono due funzioni speciali che devono essere sempre definite dall'utente

`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



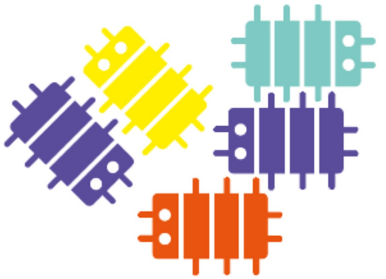
Sketch minimale per Arduino (BareMinimum in File->Examples->01.Basics)

```
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.

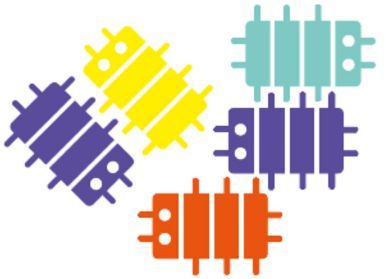
Per usare la porta seriale devo inizializzarla nella funzione `setup()`

```
int var = 8;

void setup() {
    // initialize the Serial Port
    Serial.begin(9600);
    Serial.write("hello ");

    Serial.println(var);
}

void loop() {

}
```

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

Arduino

Programmazione

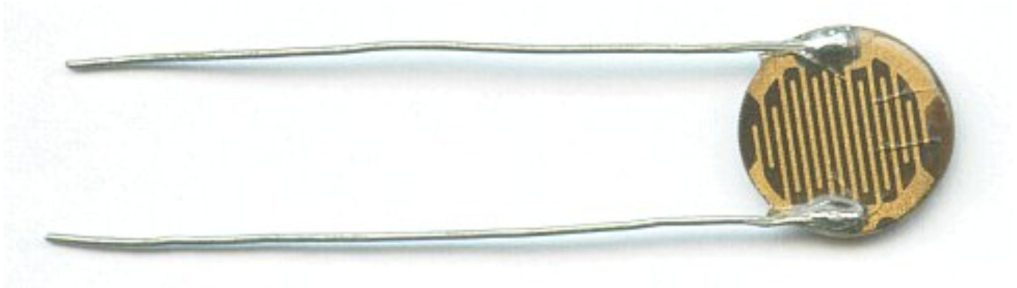


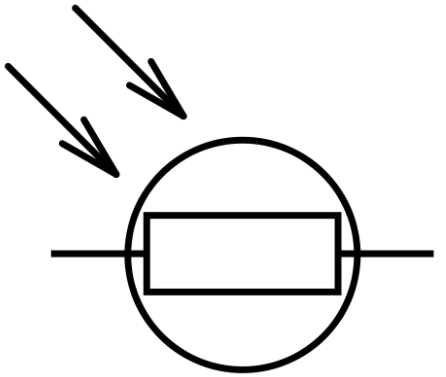
Tornando all'esempio "Blink" vediamo come avere un feedback di quello che succede nel "Serial Monitor".

```
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

Il fotoresistore o LDR (Light Dependent Resistor)



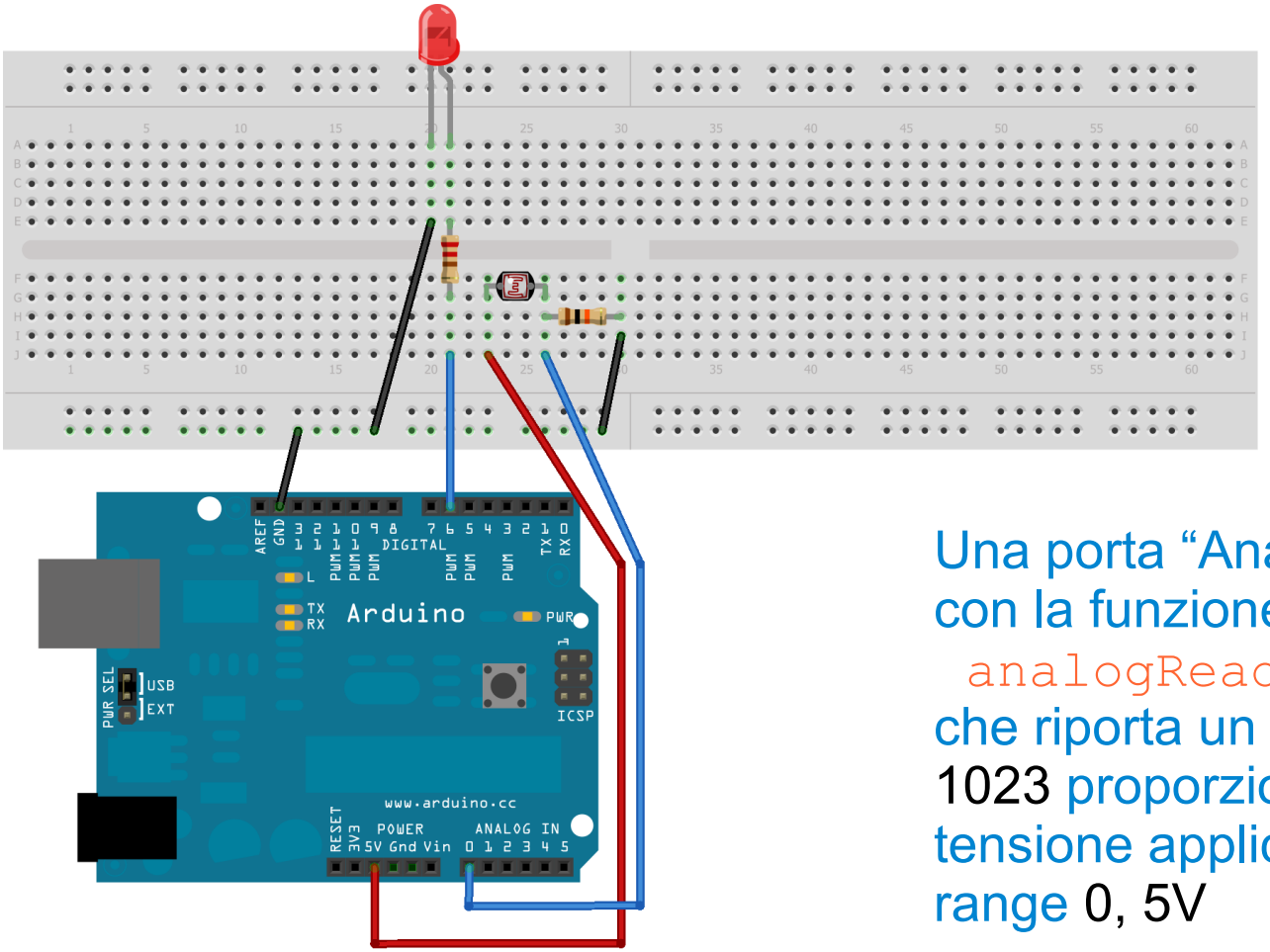


Il fotoresistore o LDR è un componente elettrico la cui resistenza diminuisce all'aumentare della intensità luminosa incidente.

Arduino

FadeLight

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



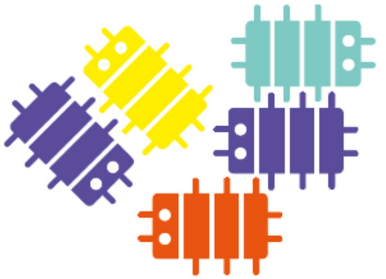
Una porta “Analog In” viene letta con la funzione:

```
analogRead(potPin);
```

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

Made with Fritzing.org

Arduino



FadeLight

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

int sensorMin = 0;
int sensorMax = 600;

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  int analogVal = analogRead(lightPin);    // 0 - 1023
  int ledVal = map(analogVal, sensorMin, sensorMax, 0, 255);

  Serial.println(analogVal);

  analogWrite(ledPin, ledVal);    // PWM
  delay(10);
}
```

Arduino

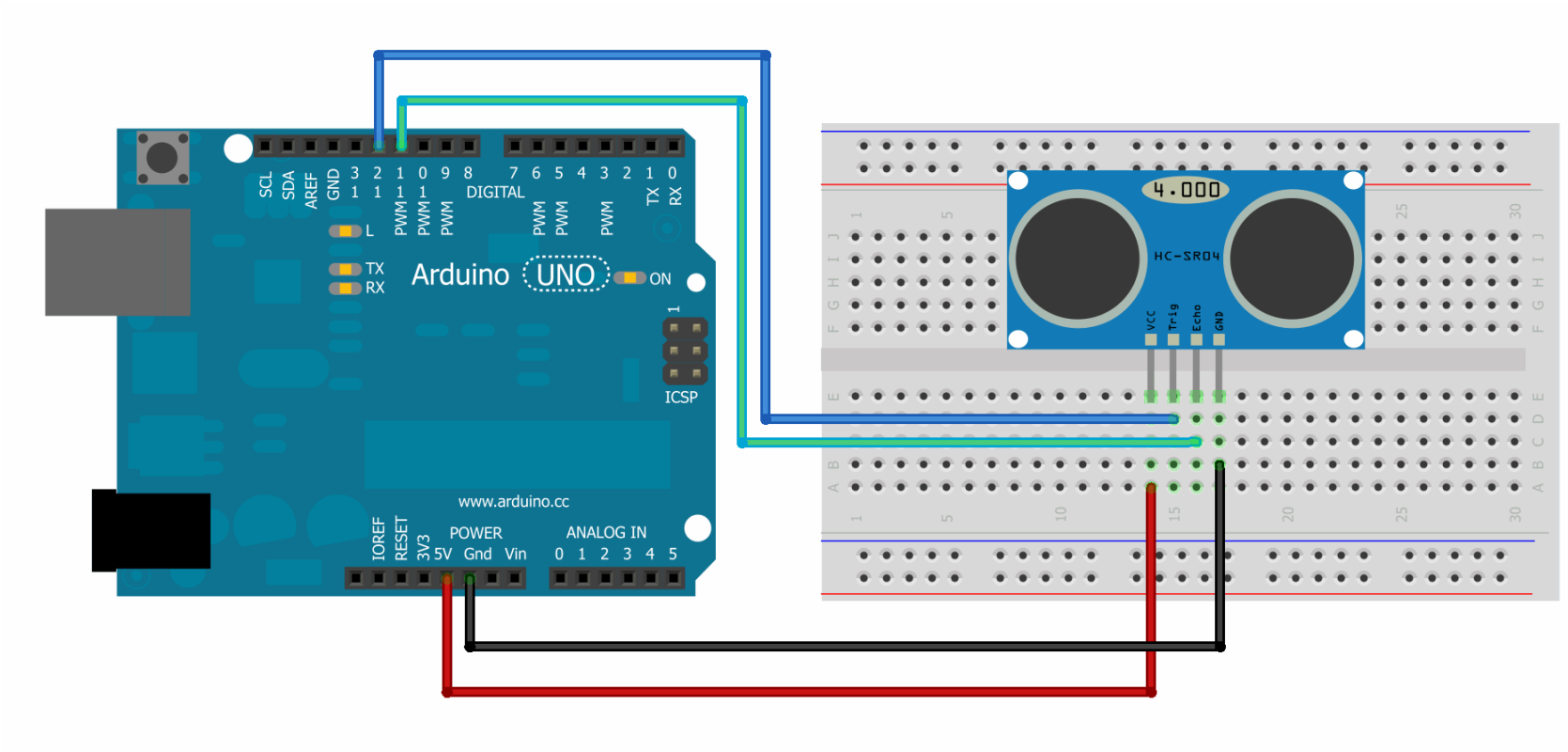
Sensore Ping

Il Ping è un sensore di distanza a ultrasuoni. Misura la distanza dell'oggetto più vicino davanti al sensore. Funziona inviando un impulso a ultrasuoni e misurando l'eco dovuto al rimbalzo dell'impulso sull'oggetto.

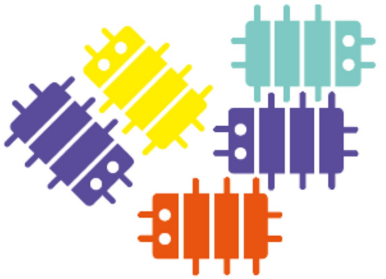
Arduino

Sensore Ping

In questo esempio useremo la libreria “NewPing” per ottenere la distanza misurata dal sensore



Arduino



Sensore Ping

```
#include <NewPing.h>

#define TRIGGER_PIN  12  // Arduino pin tied to trigger pin
#define ECHO_PIN     11  // Arduino pin tied to echo pin
#define MAX_DISTANCE 200 // Maximum distance we want to ping for (in
         centimeters). Maximum sensor distance is rated at 400-500cm

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins
and maximum distance.

void setup() {
  Serial.begin(115200);
}

void loop() {
  Delay(50);                     // Wait 50ms between pings (about 20 pings/sec)
  unsigned int uS = sonar.ping(); // Send ping, get ping time in microseconds
  Serial.print("Ping: ");
  Serial.print(uS / US_ROUNDTRIP_CM); // Convert ping time to distance in cm
                                 and print result (0 = outside set distance range)
  Serial.println("cm");
}
```

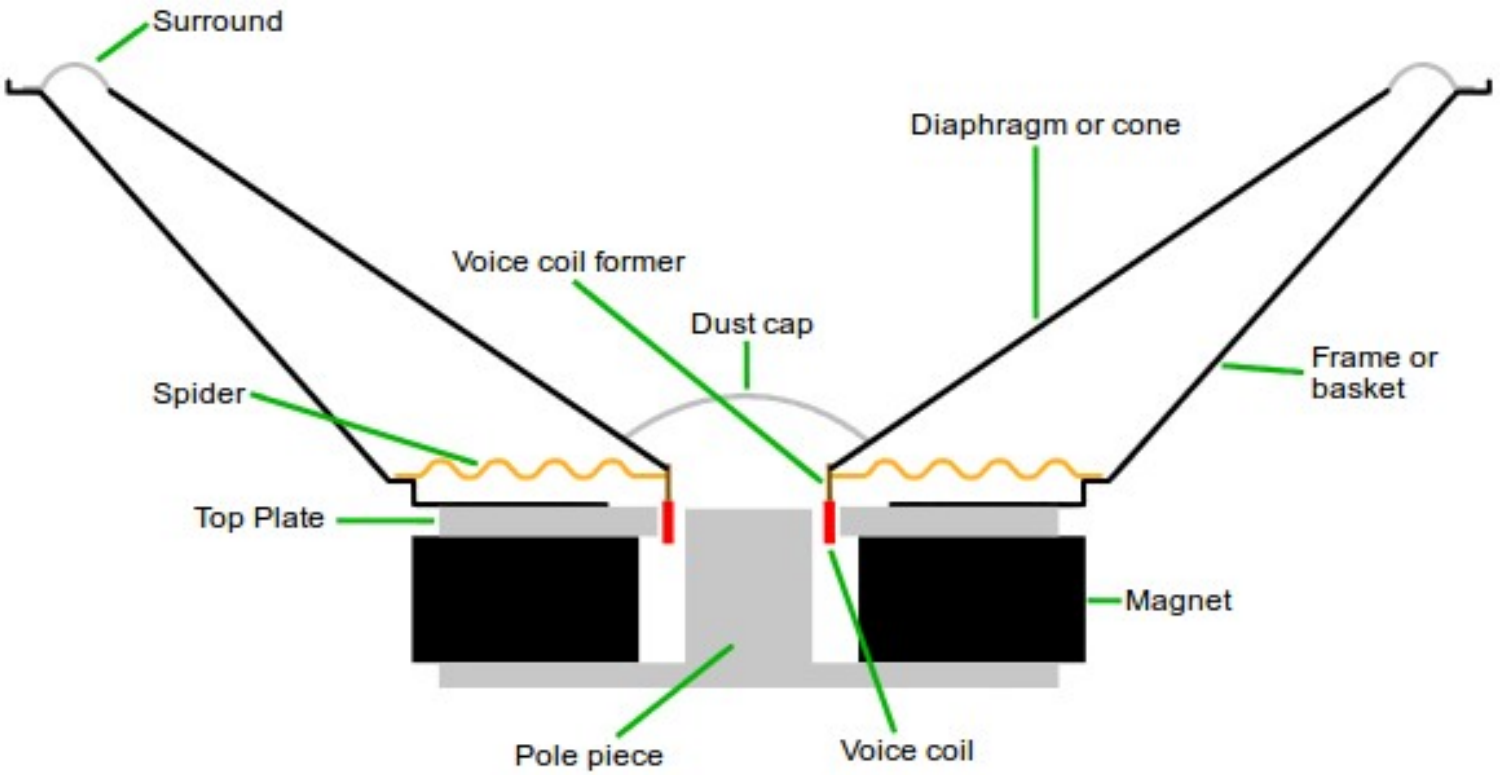
Arduino

Altoparlante

L'altoparlante è un attuatore che converte un segnale elettrico in onde sonore. Si può quindi definire un trasduttore elettroacustico. Il suono in sostanza è generato da una serie di compressioni e rarefazioni dell'aria, compito dell'altoparlante è generare tali compressioni e rarefazioni nell'ambiente d'ascolto. (Wikipedia)

Arduino

Altoparlante

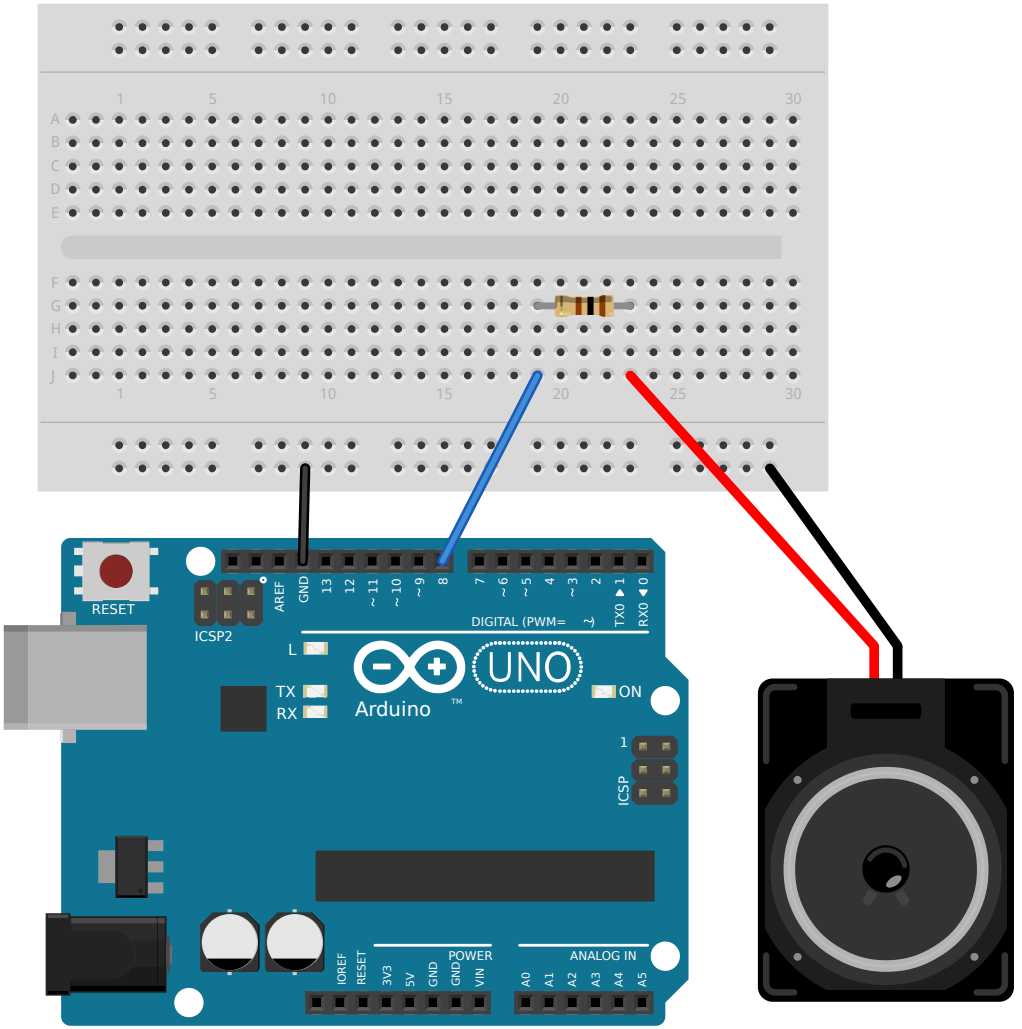


Un magnete permanente genera un campo magnetico nel quale è immersa una bobina mobile, direttamente collegata alla membrana dell'altoparlante; ad essa viene applicato un segnale elettrico, opportunamente amplificato, il quale, grazie alla Forza di Lorentz la fa muovere permettendo alla membrana, la cui forma più diffusa è conformata a cono, di comprimere l'aria circostante e quindi di produrre un'onda sonora. (Wikipedia)

Arduino

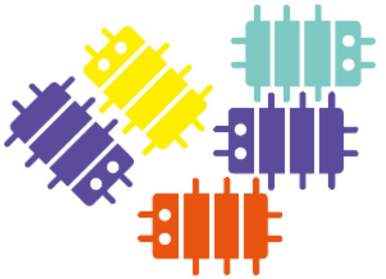
Altoparlante

Usa la funzione `tone(pin, frequency, duration)` **per generare un'onda quadra della frequenza specificata**



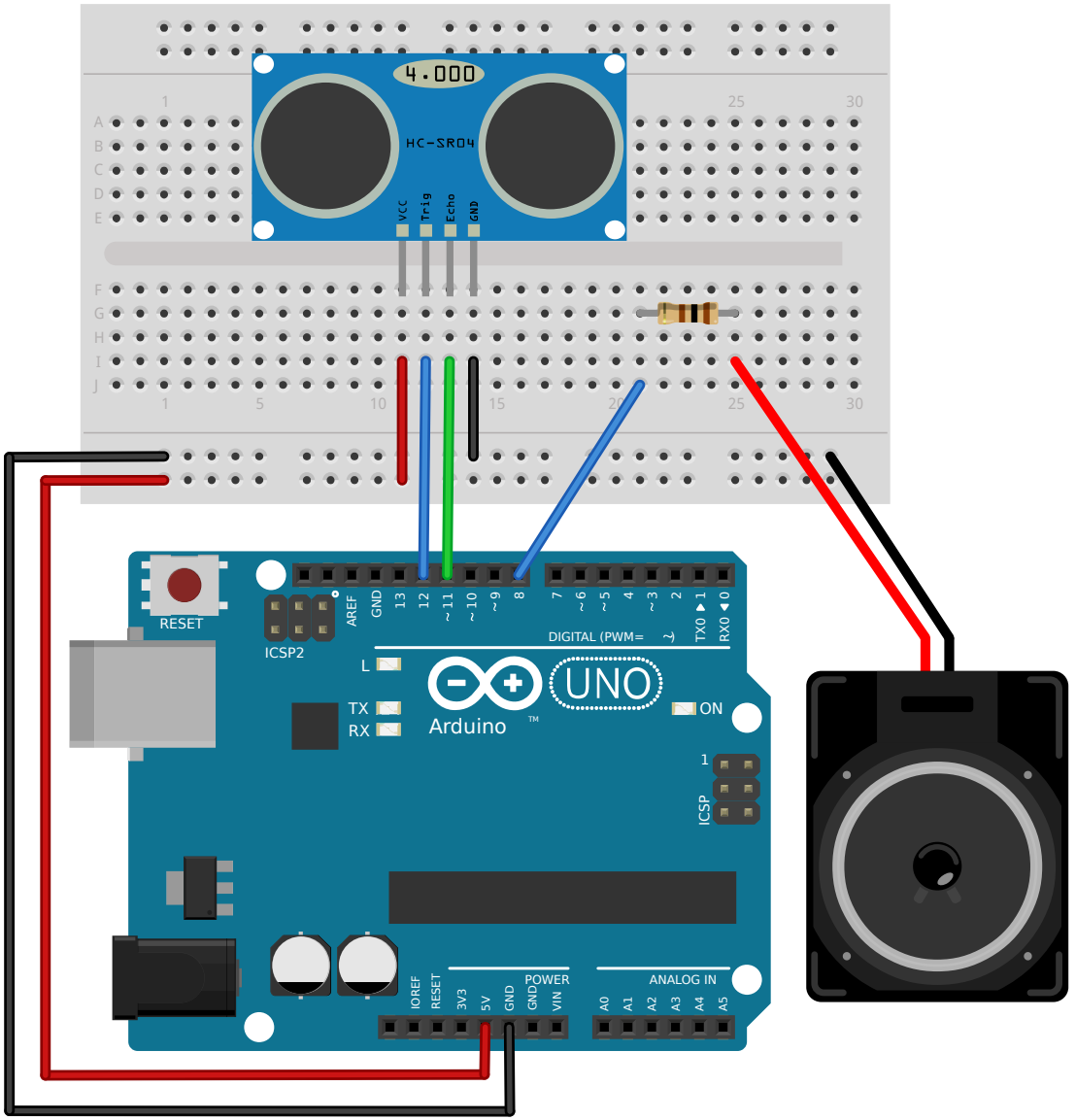
Arduino

Altoparlante



```
/* Esempio incluso nell'IDE:
       File->Examples->02.Digital->toneMelody */

#include "pitches.h"

int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3,0, NOTE_B3, NOTE_C4};

int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4 };

void setup() {
  for (int thisNote = 0; thisNote < 8; thisNote++) {
    int noteDuration = 1000/noteDurations[thisNote];
    tone(8, melody[thisNote],noteDuration);

    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);

    noTone(8);
  }
}

void loop() {
}
```

Arduino

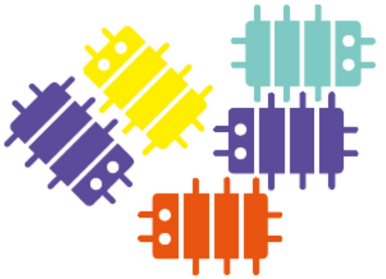


PingMelody

Usiamo il sensore Ping per generare suoni a frequenze diverse

Arduino

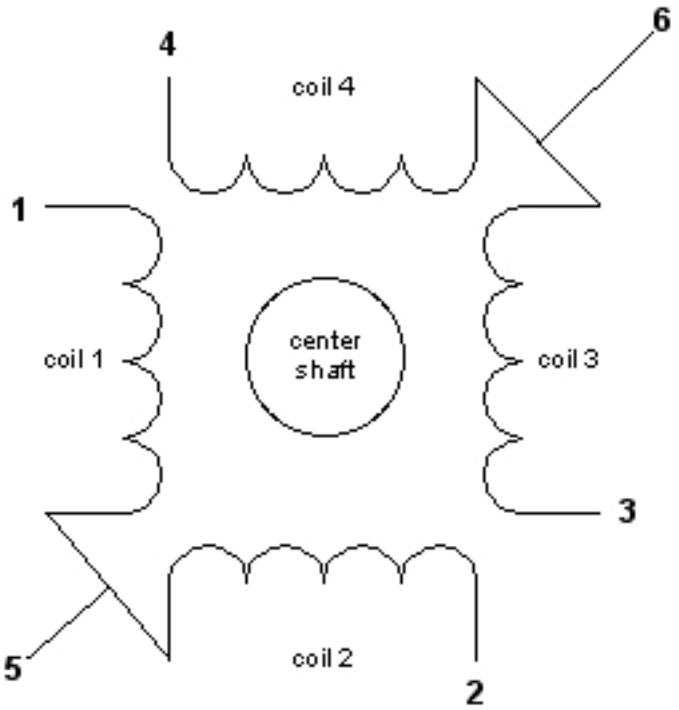
PingMelody

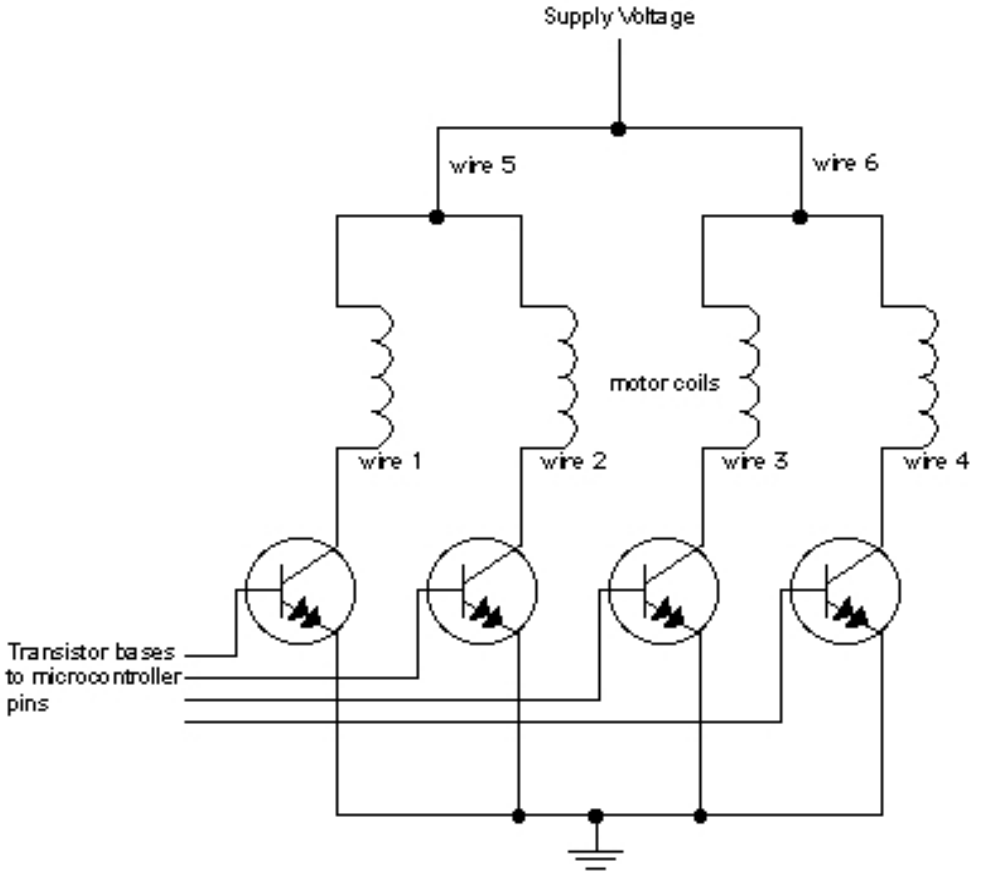


```
#include <NewPing.h>

#define TRIGGER_PIN  12  // Arduino pin tied to trigger pin
#define ECHO_PIN     11  // Arduino pin tied to echo pin
#define MAX_DISTANCE 200 // Maximum distance we want to ping (cm)

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup() {
  Serial.begin(115200);
}

void loop() {
  delay(100);                     // Wait 100ms between pings
  unsigned int uS = sonar.ping(); // Send ping, get ping time in microseconds
  Serial.print("Ping: ");
  Serial.print(uS / US_ROUNDTRIP_CM); // Convert ping time to distance in cm
  Serial.println("cm");

  int freq = (uS / US_ROUNDTRIP_CM)*10;
  tone(8, freq);
}
```

Arduino

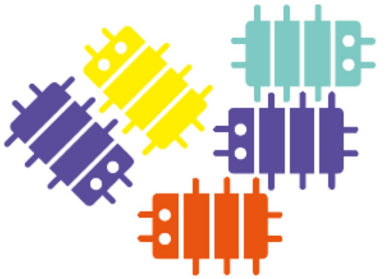
Il motore passo-passo





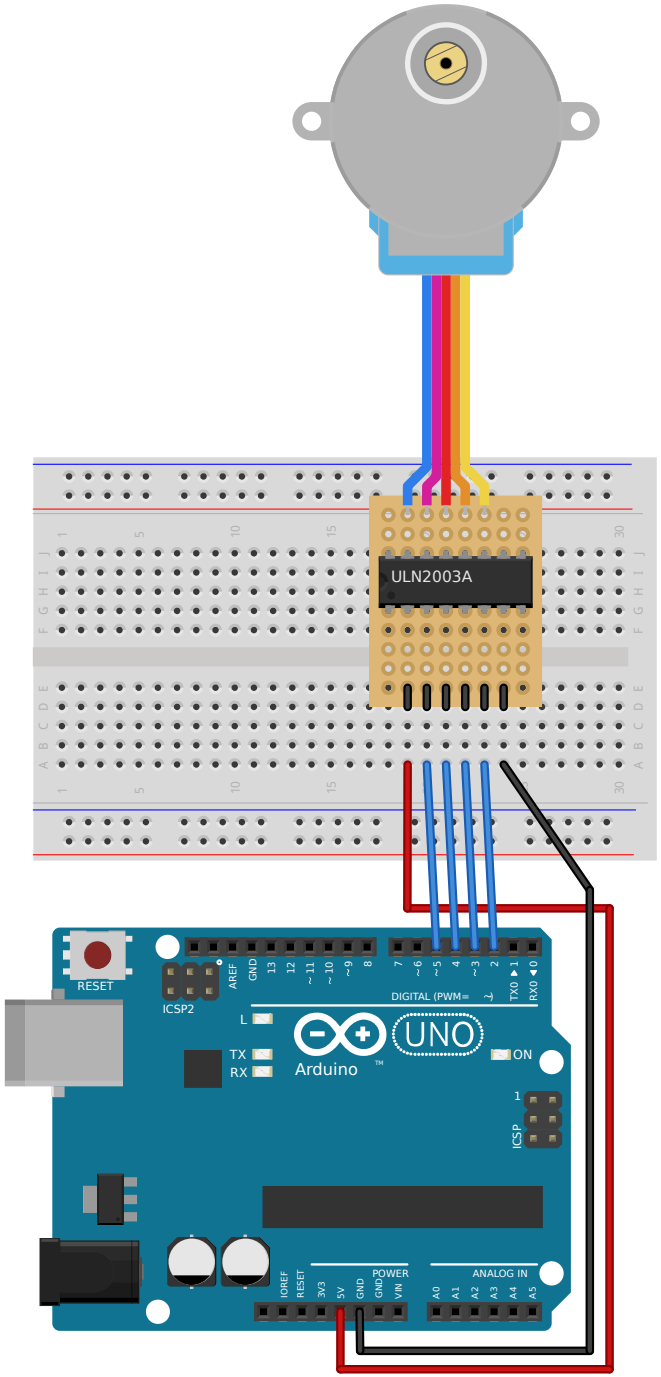
Il motore passo-passo è un motore controllato da una serie di elettromagneti. Sull'albero centrale ci sono dei magneti, le bobine che circondano l'albero sono percorse da corrente (una si una no), si creano campi magnetici che attraggono o respingono i magneti sull'albero provocando la rotazione del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

Arduino



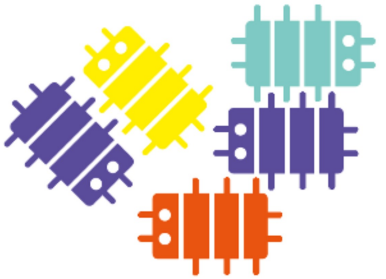
StepperMotor

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



Arduino

StepperMotor



```
#include <Stepper.h>

const int stepRev = 2038;   // 63.68395 * 64.0 / 2.0

// initialize the stepper library on pins 2 through 5:
Stepper myStepper(stepRev, 2, 4, 5, 3);

void setup() {
  // set the speed at 10 rpm:
  myStepper.setSpeed(10);
}

void loop() {
  // step one revolution  in one direction:
  myStepper.step(stepRev);
  delay(500);
  // step one revolution in the other direction:
  myStepper.step(-stepRev);
  delay(500);
}
```