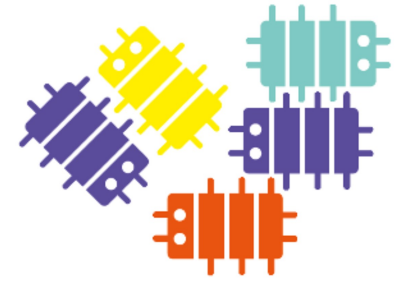
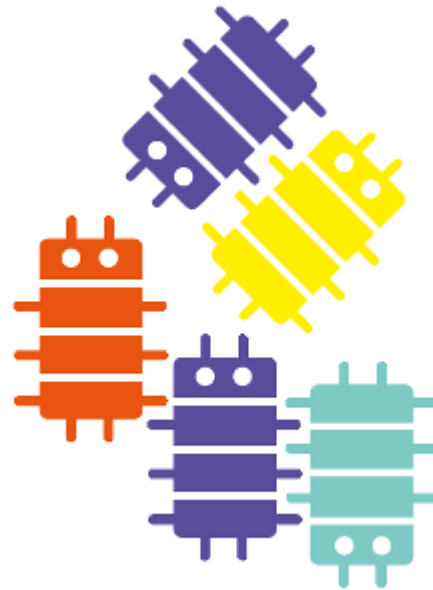


CC BY-SA

Arduino



Laboratorio Arduino Base

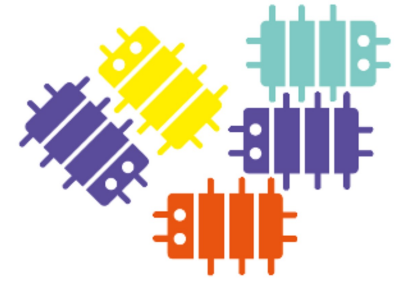


HackLab Terni

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 artisti, 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.

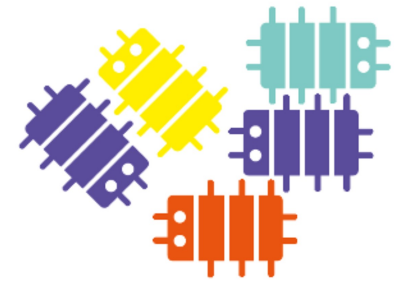
Tutte le informazioni necessarie alla realizzazione di Arduino (schemi elettrici, disegno del circuito stampato, lista dei componenti) ed il software necessario alla programmazione sono disponibili con una licenza open-source e liberamente scaricabili dal sito arduino.cc

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.

CC BY-SA

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

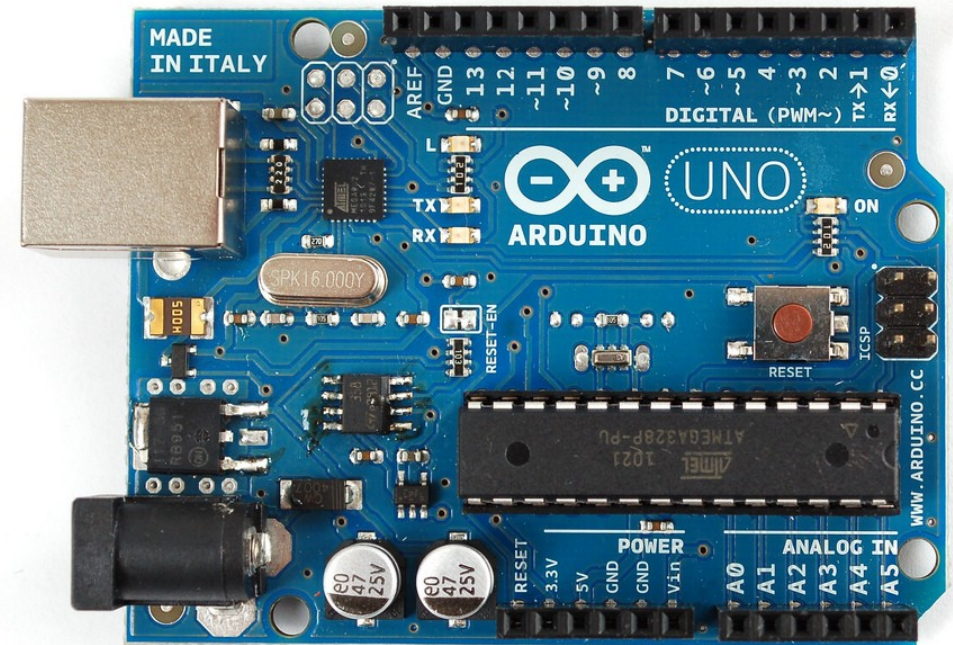
2KB

EEPROM

1KB

Velocità del Clock

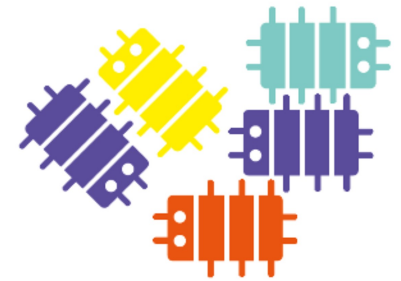
16MHz



Arduino Uno

(fonte: arduino.cc)

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.

```
Blink | Arduino 1.0.3
File Modifica Sketch Strumenti Aiuto

Blink
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

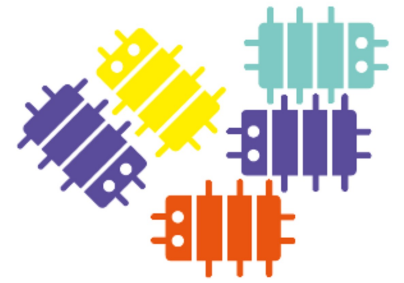
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage
  delay(1000); // wait for a second
  digitalWrite(led, LOW); // turn the LED off by making the voltag
  delay(1000); // wait for a second
}

Compilazione terminata.

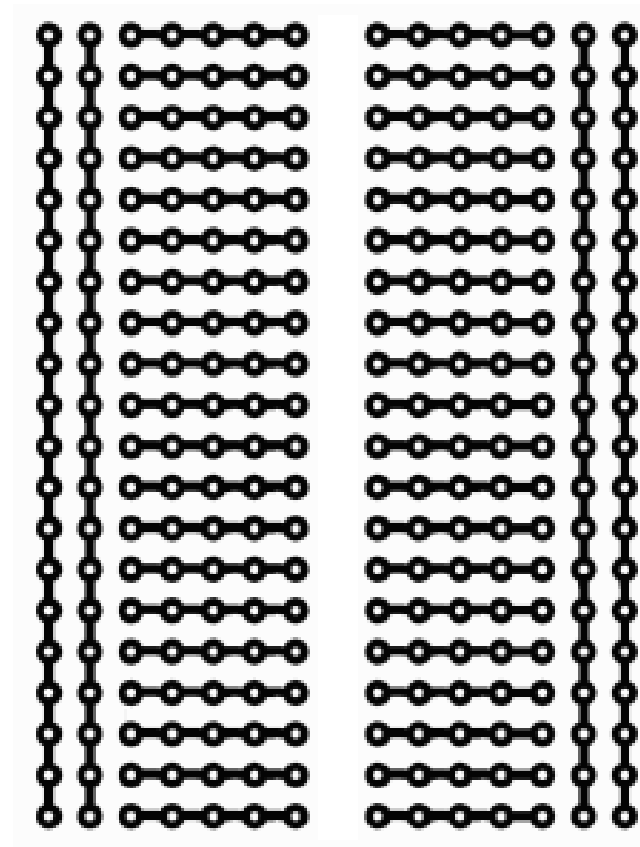
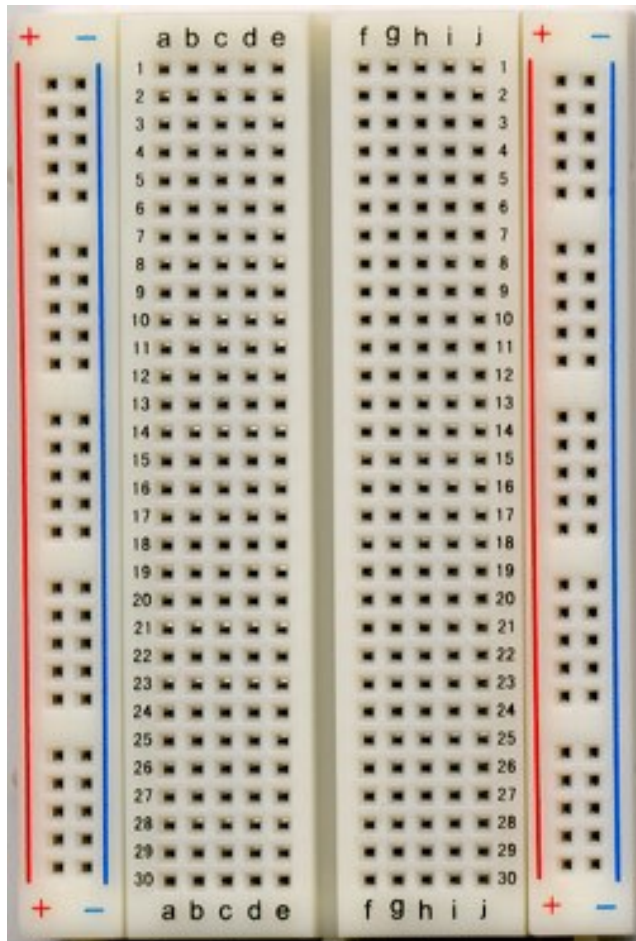
Dimensione del file binario dello sketch: 1.084 bytes (su un
massimo di 32.256 bytes)

1 Arduino Uno on COM1
```

Arduino

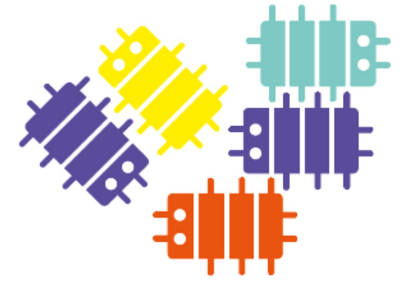


La breadboard



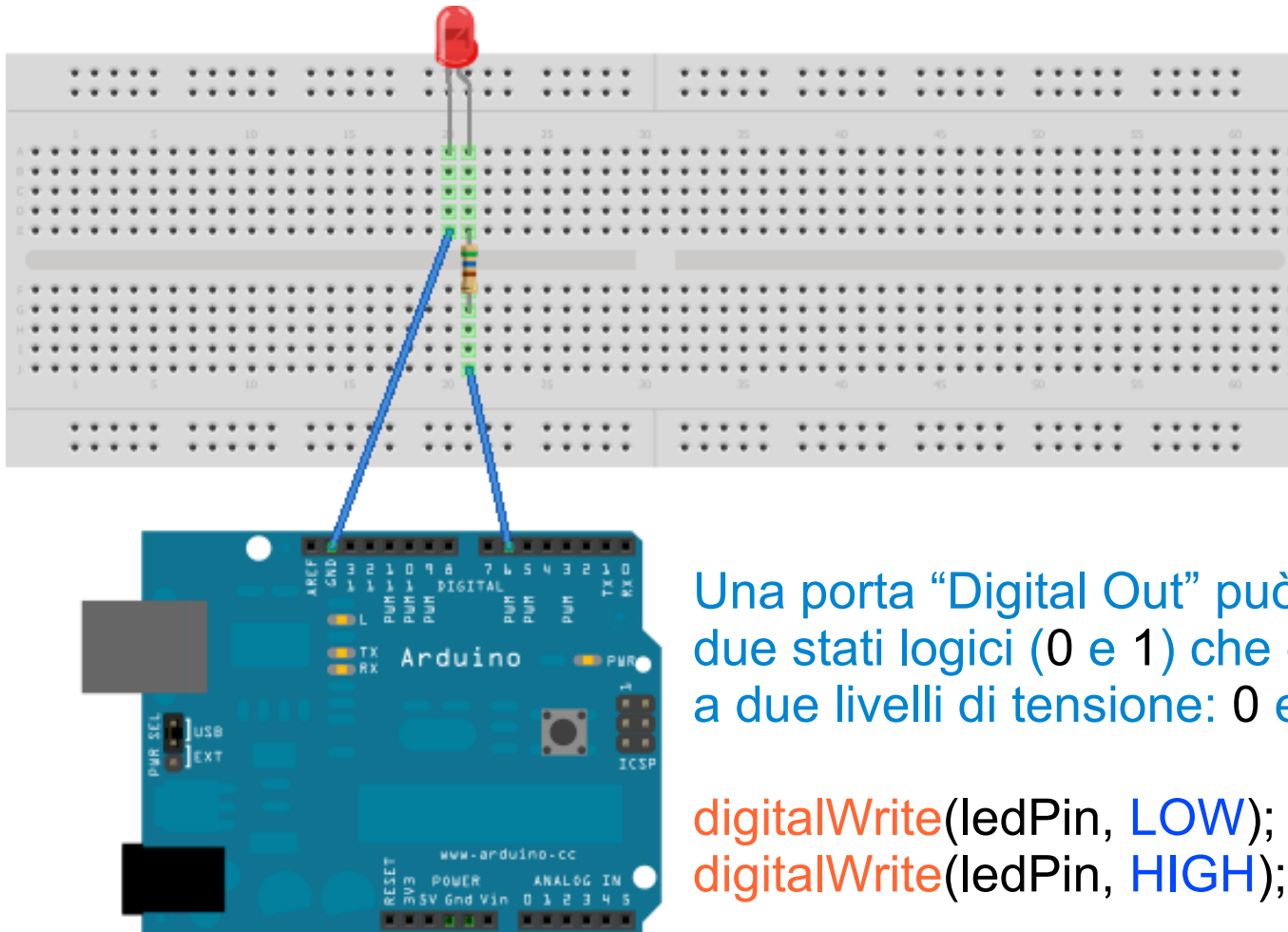
Video and Website © 2004 ClarkZapper.net

Arduino



Blink

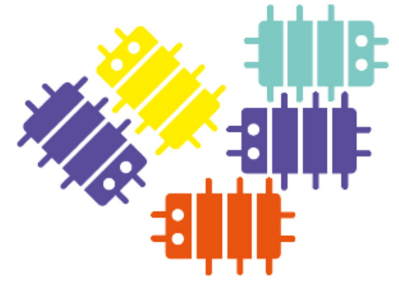
Usa una porta "Digital Out" per far lampeggiare un LED



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



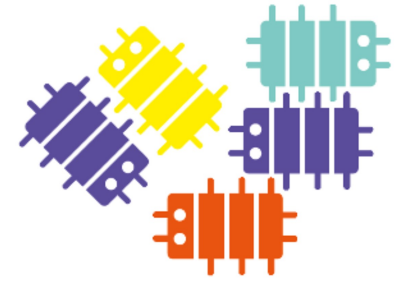
Blink

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

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

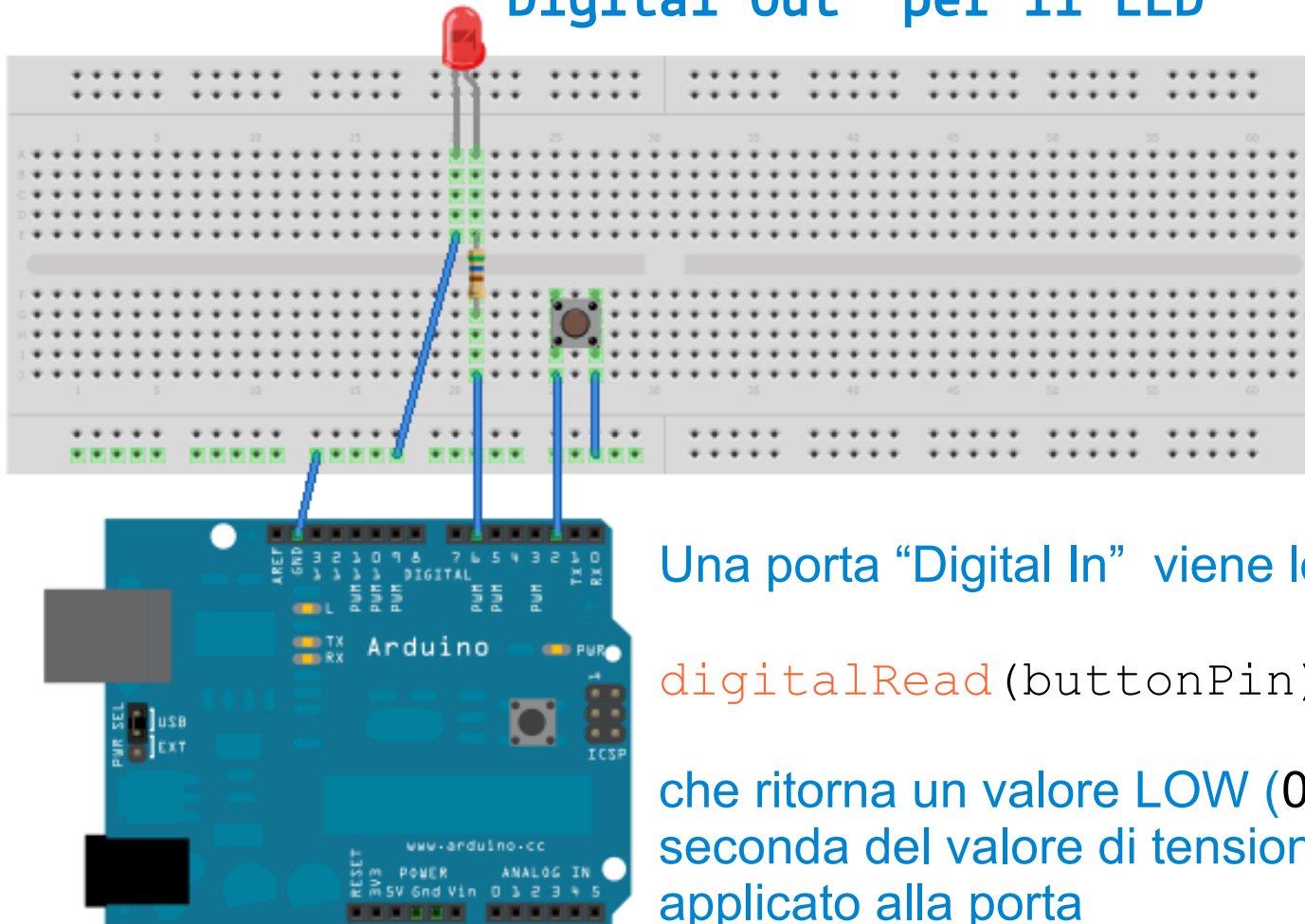
void loop() {
  digitalWrite(ledPin, HIGH); // turn LED on
  delay(1000);                // wait for a second
  digitalWrite(ledPin, LOW);  // turn LED off
  delay(1000);                // wait for a second
}
```


Arduino



Button

Usa una porta "Digital In" per il pulsante e una "Digital Out" per il LED



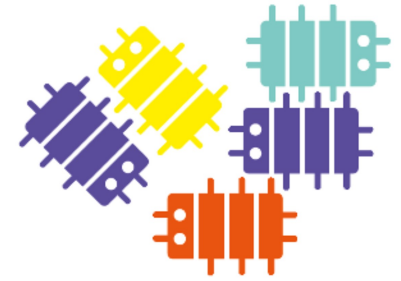
Una porta "Digital In" viene letta dalla funzione:

```
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



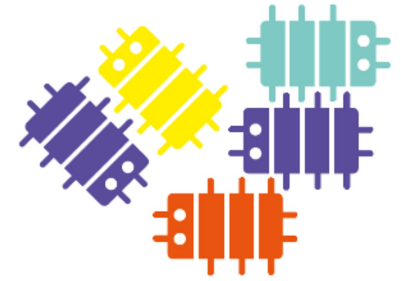
```
int buttonPin = 2;    // the number of the pushbutton pin
int ledPin = 6;      // the number of the LED pin
int buttonState = 0; // variable for reading the pushbutton status

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);

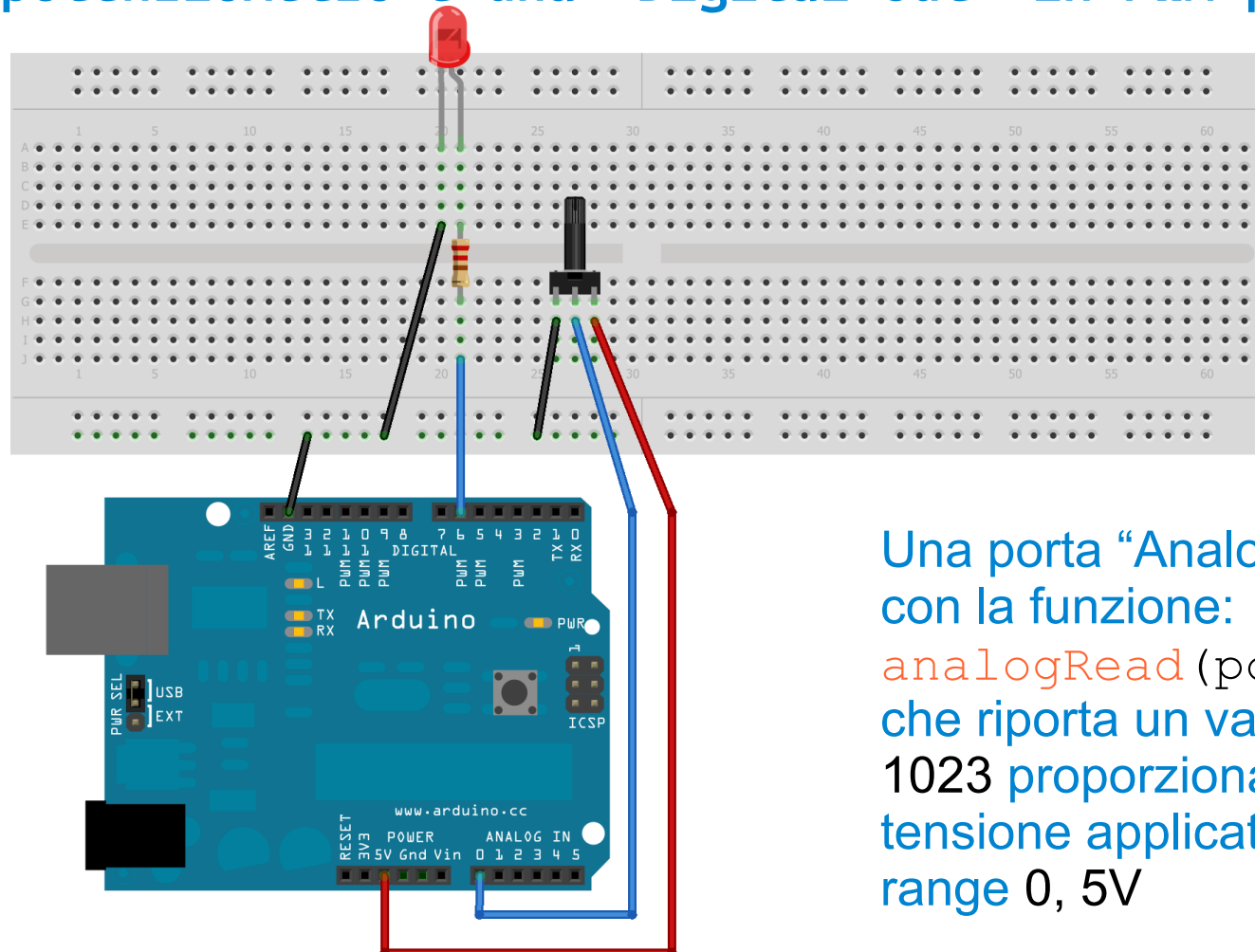
  // check if the pushbutton is pressed.
  // if it is, the buttonState is LOW:
  if (buttonState == LOW) {
    digitalWrite(ledPin, HIGH); // turn LED on
  } else {
    digitalWrite(ledPin, LOW);  // turn LED off
  }
}
```

Arduino



FadePot

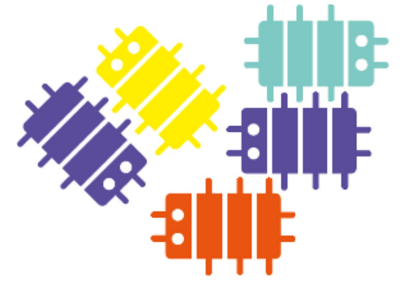
Usa una porta "Analog In" per leggere la posizione del potenziometro e una "Digital Out" in PWM per il LED



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

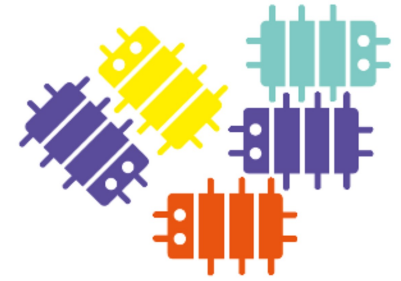


```
int ledPin = 6;           // the number of the LED pin
int potPin = A0;         // analog input pin

int potVal = 0;
int ledVal = 0;

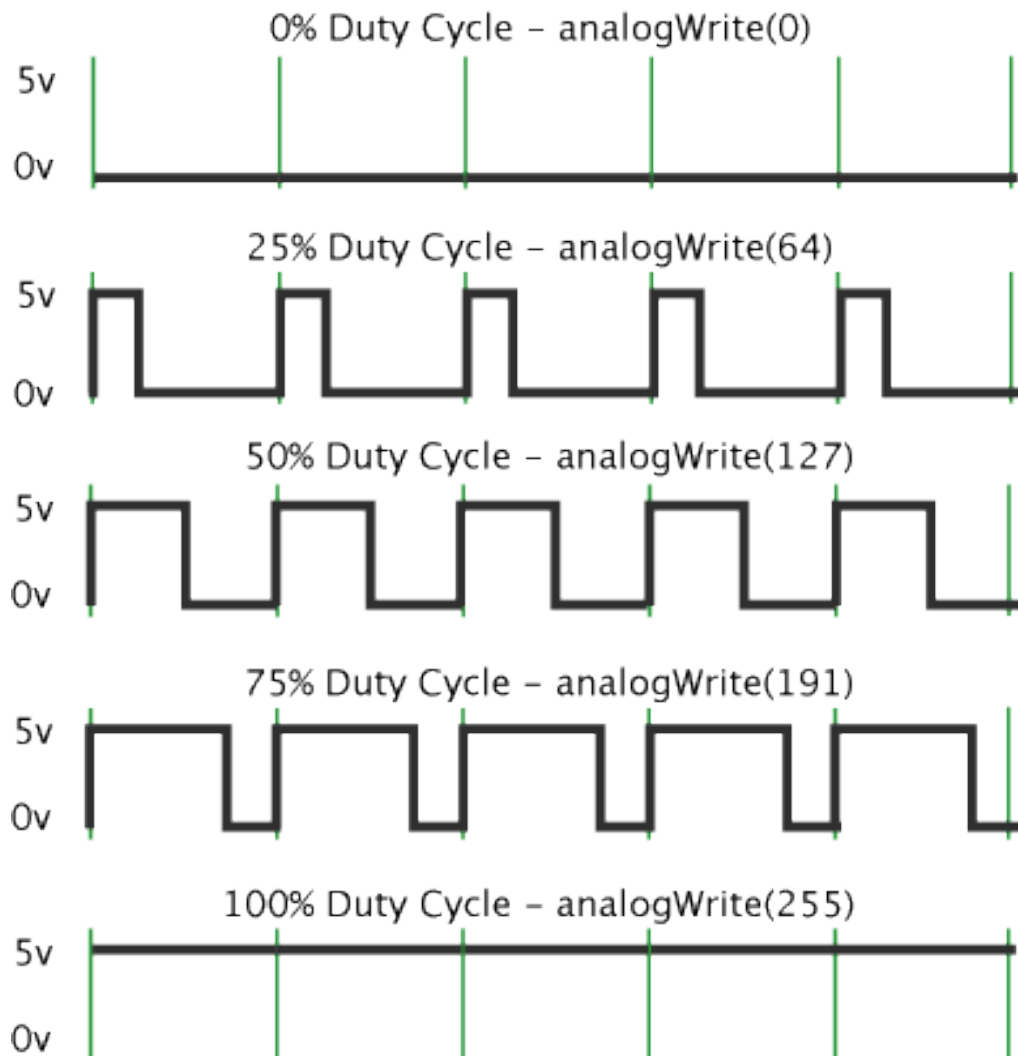
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
}

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



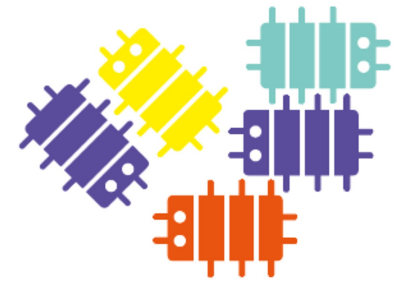
PWM (Pulse Width Modulation)

Pulse Width Modulation

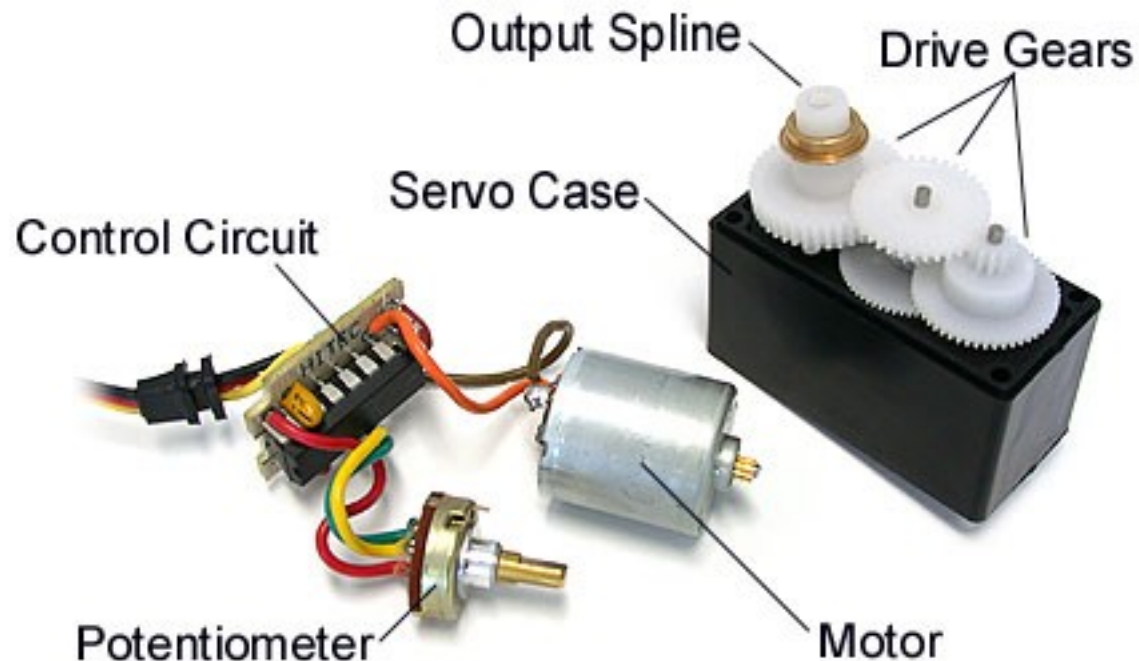


Per scrivere un valore analogico in una porta si usa la funzione: `analogWrite(ledPin, val);` dove `val` è un intero che può assumere valori tra 0 e 255

Arduino



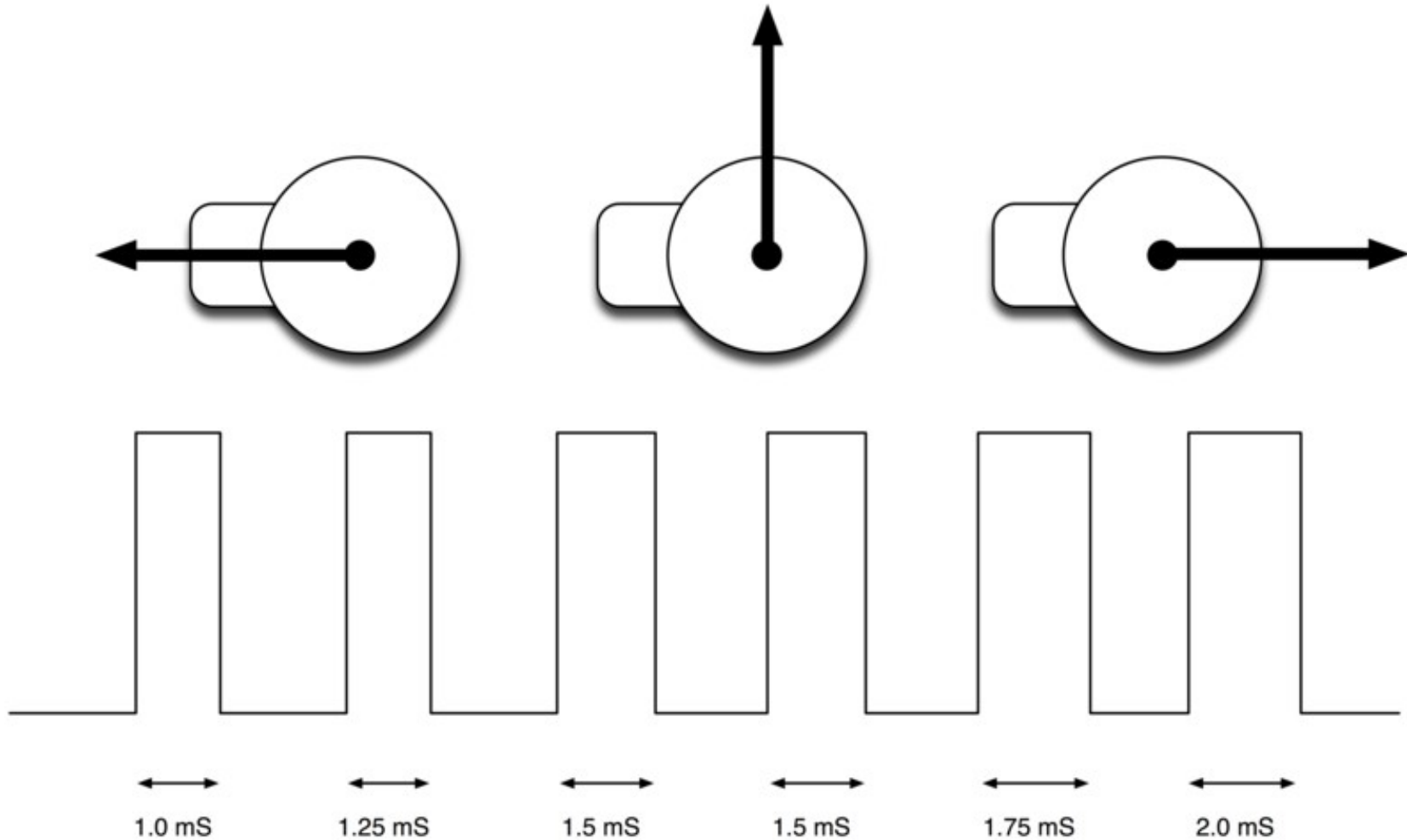
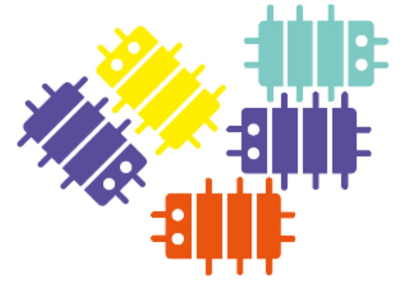
Il servomotore



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.

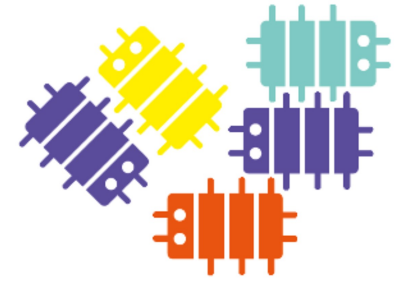
Arduino

Il servomotore



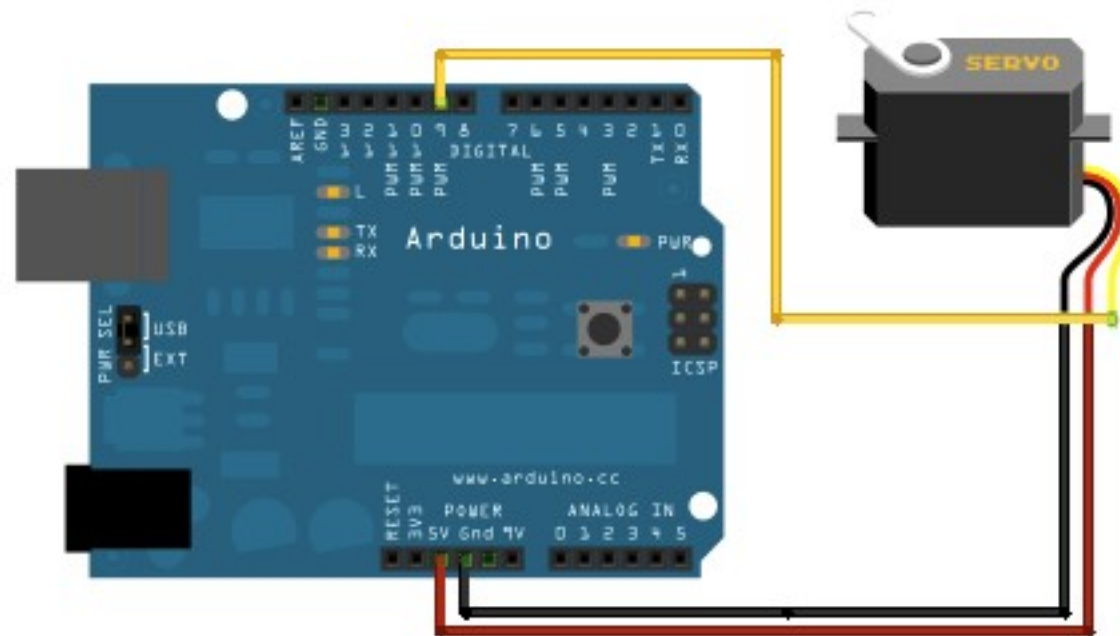
La posizione del servomotore viene impostata dalla lunghezza di un impulso. Il servomotore si aspetta di ricevere un impulso ogni 20ms.

Arduino



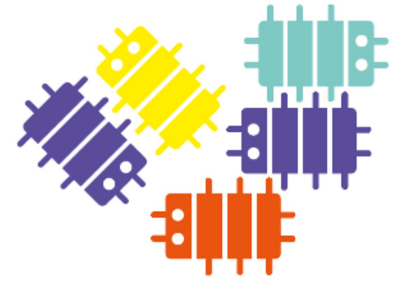
Sweep

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



Arduino

Sweep



```
#include <Servo.h>

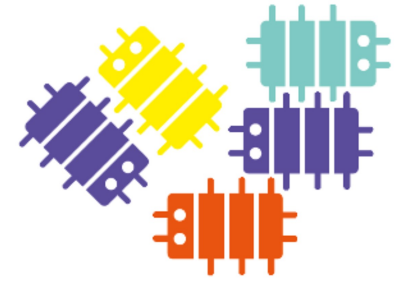
Servo myservo; // create servo object to control a servo
                // a maximum of eight servo objects can be created

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

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

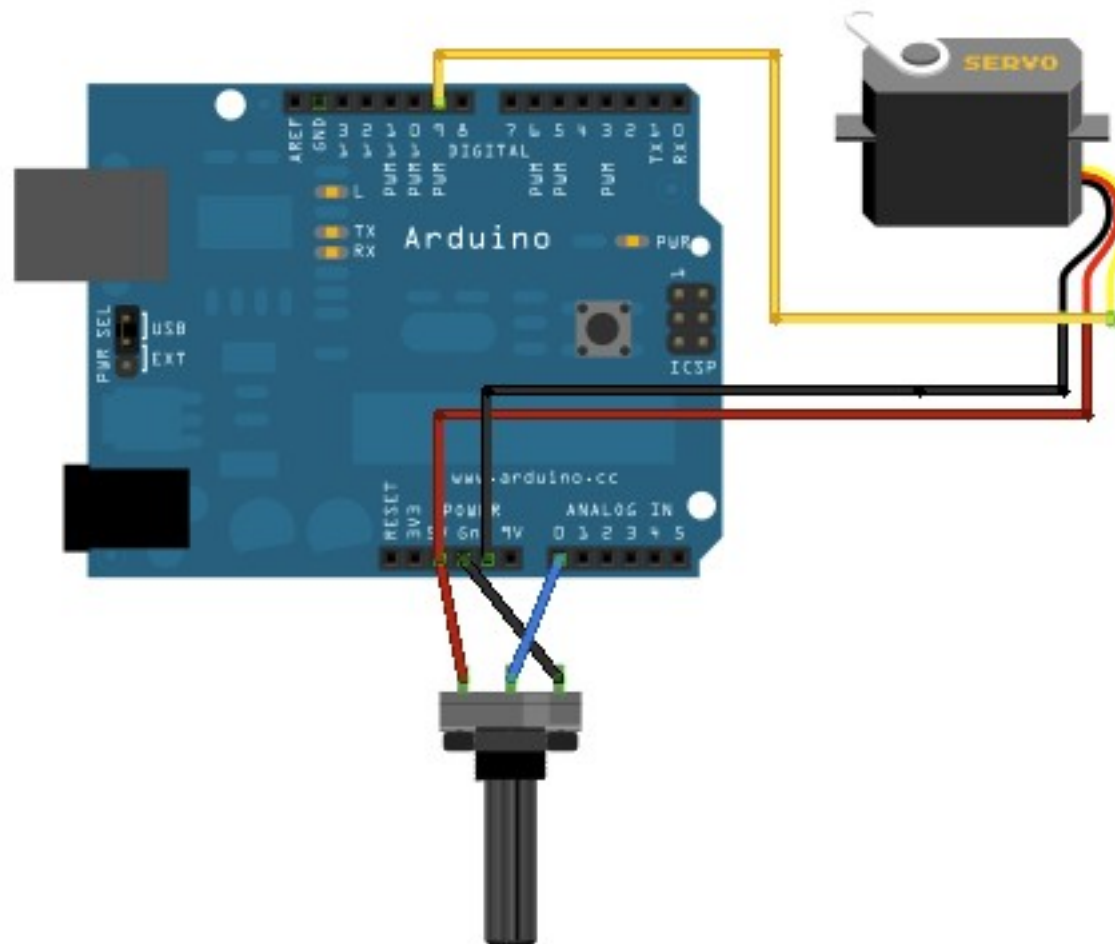
void loop()
{
  for(pos = 0; pos < 180; pos += 1) // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    myservo.write(pos);             // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to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



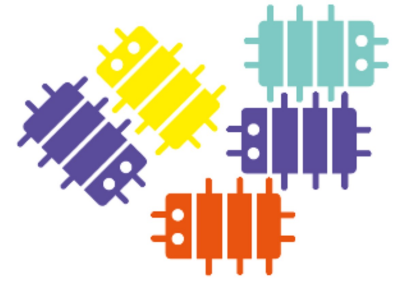
Knob

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

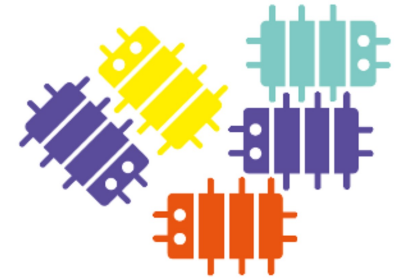
int potpin = 0;  // analog pin used to connect the potentiometer
int val;        // variable to read the value from the analog pin

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 breadboard e degli schemi che abbiamo visto sono stati realizzati con "Fritzing":

<http://fritzing.org/>