

Internet of Things (IoT)

Concetti ed esempi con Arduino

Le origini

- Telemetry & Telemetric systems → M2M → IoT
- L'espressione **IoT** fu coniata ed usata per la prima volta da **Peter T. Lewis** nel **Settembre 1985** in una conferenza a Washington DC.
- Nel 1999 **K. Ashton** (MIT) la utilizza come titolo di una presentazione ed inizia a diffondersi.
- Prima di indicarli come IoT, si è spesso parlato degli stessi argomenti o di aspetti strettamente correlati indicandoli con le espressioni:
 - **Pervasive Computing**
 - **Ubiquitous Computing**
 - **Mobilità** del codice e dei dati

Reti di dispositivi: smart connectivity

- La rete Internet fino ad oggi è stata finalizzata alla comunicazione fra persone (mediante PC, smartphone, tablet, server, ecc.).
- In futuro sarà un'infrastruttura anche per entità non umane con comunicazione **M2M**.
- IoT: **connettere in modo intelligente (smart)** una grande varietà di oggetti per farli comunicare.
- Gli oggetti connessi devono essere **univocamente identificabili** (e.g., tramite un indirizzo IP).
- La rete di connessione può essere Internet, ma anche una rete proprietaria/chiusa.
- Molti oggetti connessi e comunicanti implica:
 - enormi quantità di dati: IoT come fonte di **Big Data**,
 - più servizi **automatizzati** ed **intelligenti**,
 - **interazione minima** con le persone.

The "Things" in IoT

- Quali sono i tratti distintivi di una "cosa"?
 1. deve contenere un **ricevitore/trasmittitore wireless** (Wi-Fi, Bluetooth, ZigBee ecc.), ma ciò non esclude la possibilità di connessioni via cavo;
 2. deve possedere un **indirizzo IP univoco**;
 3. deve essere dotata di un **sensore** o di HW in grado di svolgere un compito;
 4. deve avere capacità di tipo **store-and-forward** (memoria interna);
 5. deve essere **low-power** e **low-bandwidth**.
- Le entità connesse non devono essere inanimate: possono anche essere degli animali (cani, gatti ecc.) o persone, dotati di **transponder** o **impianti** con le giuste caratteristiche.
- Quindi il numero di unità connesse è potenzialmente altissimo:
 - Gartner stima che nel 2020 gli oggetti connessi saranno 26 miliardi;
 - Si parla quindi di "**Internet of Everything**".

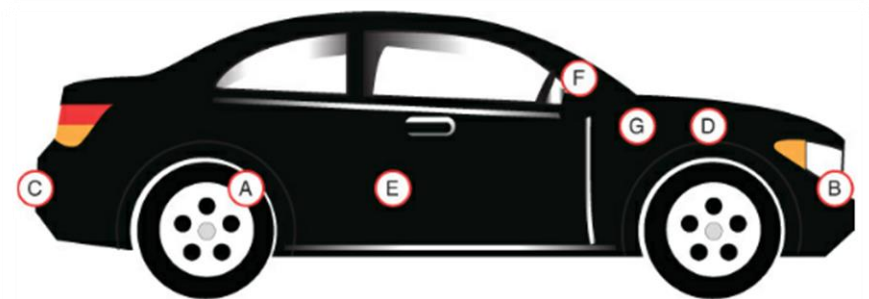
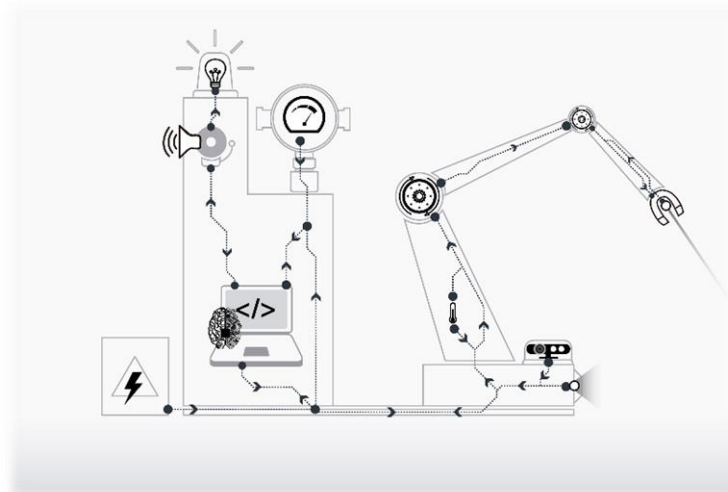
Sensori e raccolta dati

- Grandissima **varietà di sensori** (Sensor Revolution):
 - termometro,
 - rilevatore di luminosità,
 - barometro,
 - bussola,
 - giroscopio,
 - accelerometro,
 - ...
- Tuttavia, una cosa per l' IoT non deve essere necessariamente "fisica" (reale):
 - può anche essere un dato (misurato da un dispositivo fisico);
 - ciò che è veramente importante sono i **dati**.



L'essenza dell'IoT

- Il fatto che rende interessante l'IoT è che
 - **connettendo** un numero sufficiente di sensori, dispositivi, computer si ottiene
 - un **sistema autonomo e coerente** che può agire con una propria **intelligenza artificiale** (ambient intelligence) per **risolvere un problema**, senza bisogno di una costante interazione con gli umani.
- Il risultato finale è qualcosa di più della somma delle sue parti.

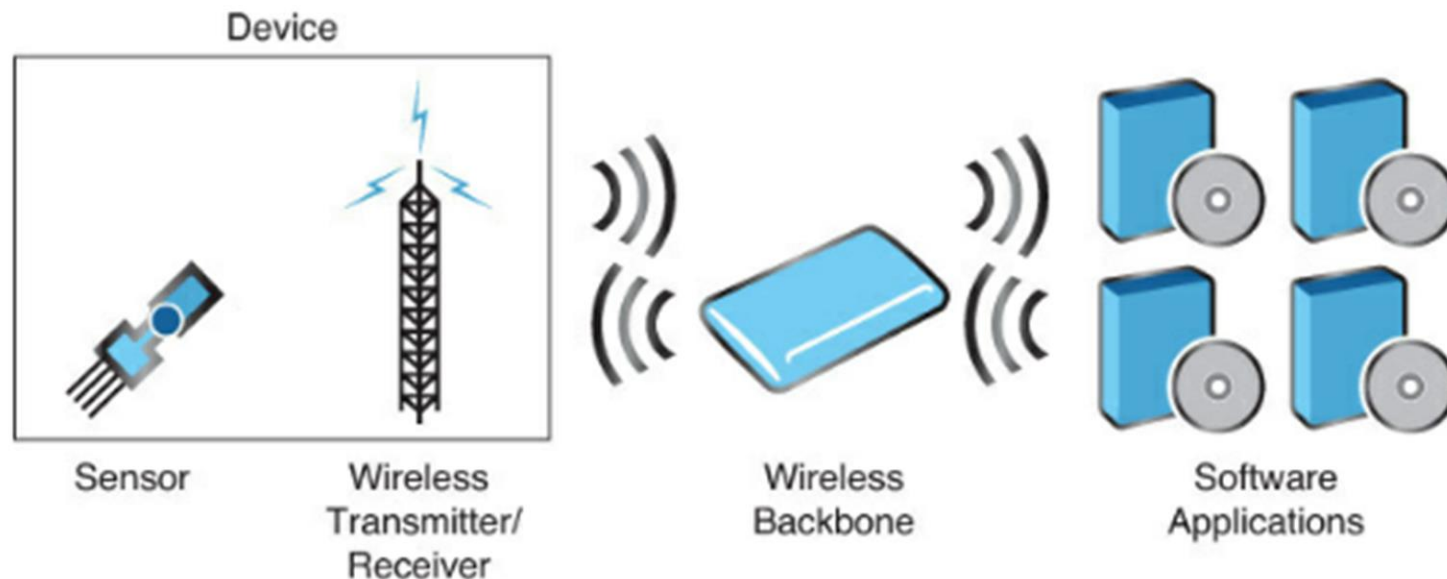


Legend

- | | |
|------------------------------|-----------------------------|
| (A) Brake Sensor | (E) Door Sensors |
| (B) Radar Sensor | (F) Adaptive Cruise Control |
| (C) Backup Sensor and Camera | (G) Master Controller |
| (D) Engine Sensors | |

Lo scenario applicativo di IoT

- **Dispositivi** (con sensori e ricevitori/trasmittitori wireless, raccolgono ed inviano dati)
- **Network backbone** (collega i componenti)
- **Applicazioni software** (elaborano dati, prendono decisioni)



Dati

- Tre sfide:
 - **Raccogliere i dati** (data harvesting/ingestion):
 - connettere i dispositivi.
 - **Memorizzare i dati** (data storage):
 - DBaaS, soluzioni cloud-based.
 - **Analizzare i dati** (data analysis), ovvero,
 - come far fruttare i dati raccolti e memorizzati?
 - figura professionale del Data Analyst
 - Machine Learning

AREE Principali dell'IoT

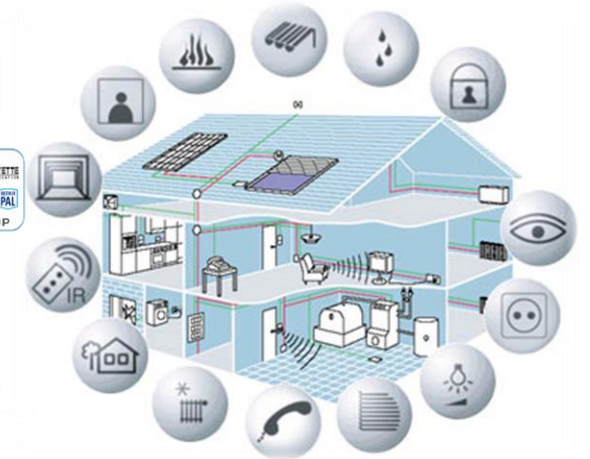
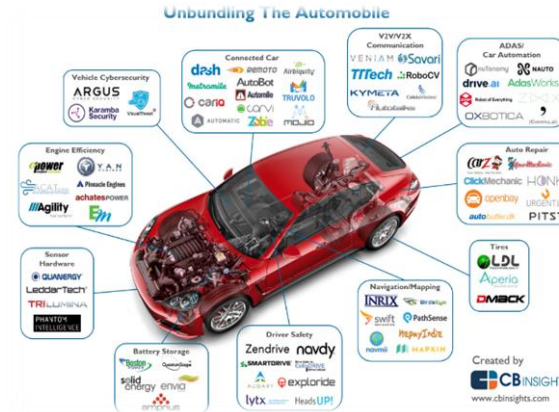
- Home →

- Domotica (Smart Home),
- Smart Clothing,
- Smart Shopping,
- Smart Cars



- Enterprise → Smart office/manufacturing/warehousing/transportation, Industry 4.0

- Government → Smart Medicine, Smart City, Smart World, Smart Warfare



Arduino e Processing

Come interfacciarsi con il mondo reale



Parte I

Arduino

Cos'è Arduino

- Arduino (<https://www.arduino.cc/>) è una piattaforma **open-source** di **prototipazione elettronica** (nato ad opera di italiani ad Ivrea, nel **2005**).
- È stata ideata per rendere facile l'interazione di un sistema di calcolo con l'ambiente circostante utilizzando una grande varietà di sensori, motori ed altri attuatori.
- Il microprocessore sulla scheda si programma con il linguaggio di programmazione **Arduino** (derivato da **Wiring**) e l'ambiente di sviluppo Arduino (basato su Processing).
- I progetti sviluppati con Arduino
 - possono funzionare controllati direttamente dal software sulla scheda (modalità **stand-alone**),
 - oppure possono comunicare con software in esecuzione su un computer (per esempio **Processing**).

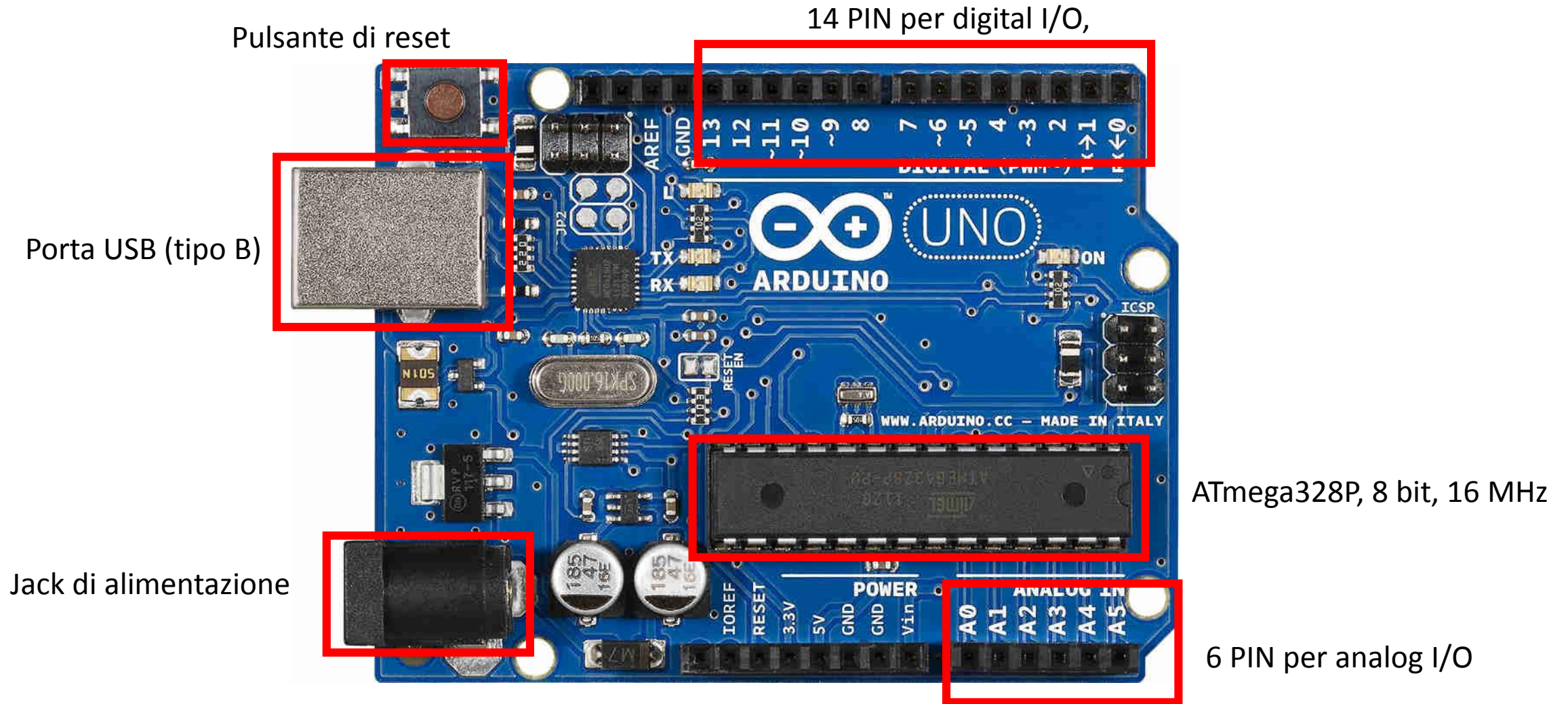
Versioni di Arduino ed applicazioni

- Esistono molte versioni della scheda Arduino e molti prodotti basati su di essa:
 - <https://www.arduino.cc/en/Main/Products>
 - nel 2007, alcuni programmatori rilasciarono un programma Arduino ("ArduCopter") per stabilizzare un modellino di elicottero radiocomandato;
 - nel 2009 venne rilasciato Ardupilot 1.0, ovvero, un software per il controllo di droni, aeromobili, rover ecc. (<http://ardupilot.org/>).

Programmare Arduino

- Arduino si può programmare tramite l'apposito IDE, scaricabile da:
 - <https://www.arduino.cc/en/Main/Software>
- Nei successivi esempi programmeremo la board direttamente da Processing, senza scrivere codice nativo.
- Tuttavia l'IDE va comunque installato perché porta con sé i driver della scheda ed è utile per installare il programma che implementa il protocollo di interfaccia via seriale verso Processing.

La scheda

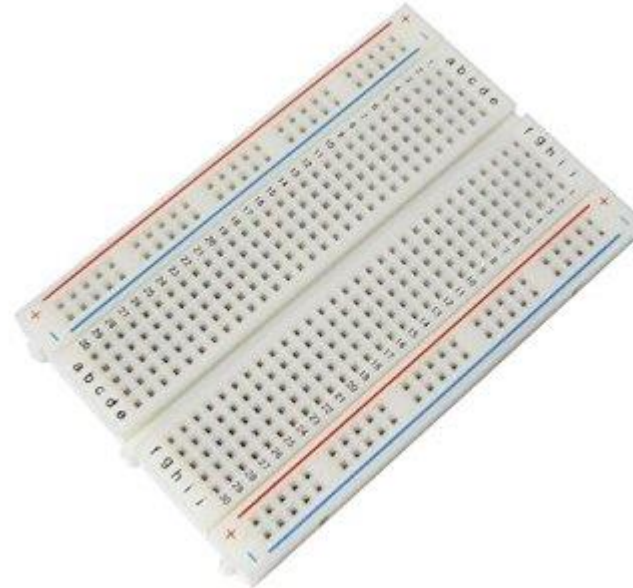


La Breadboard

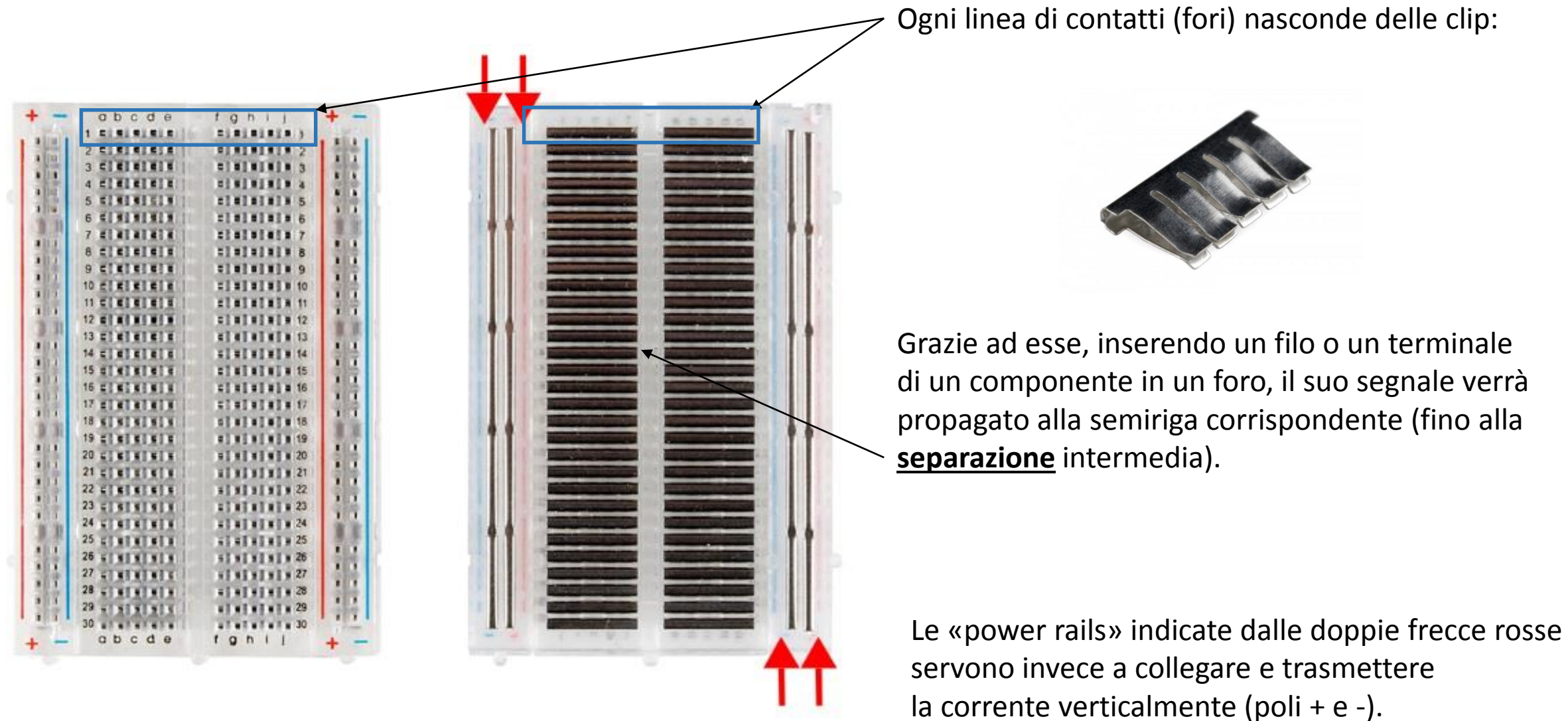
- Origine del nome:



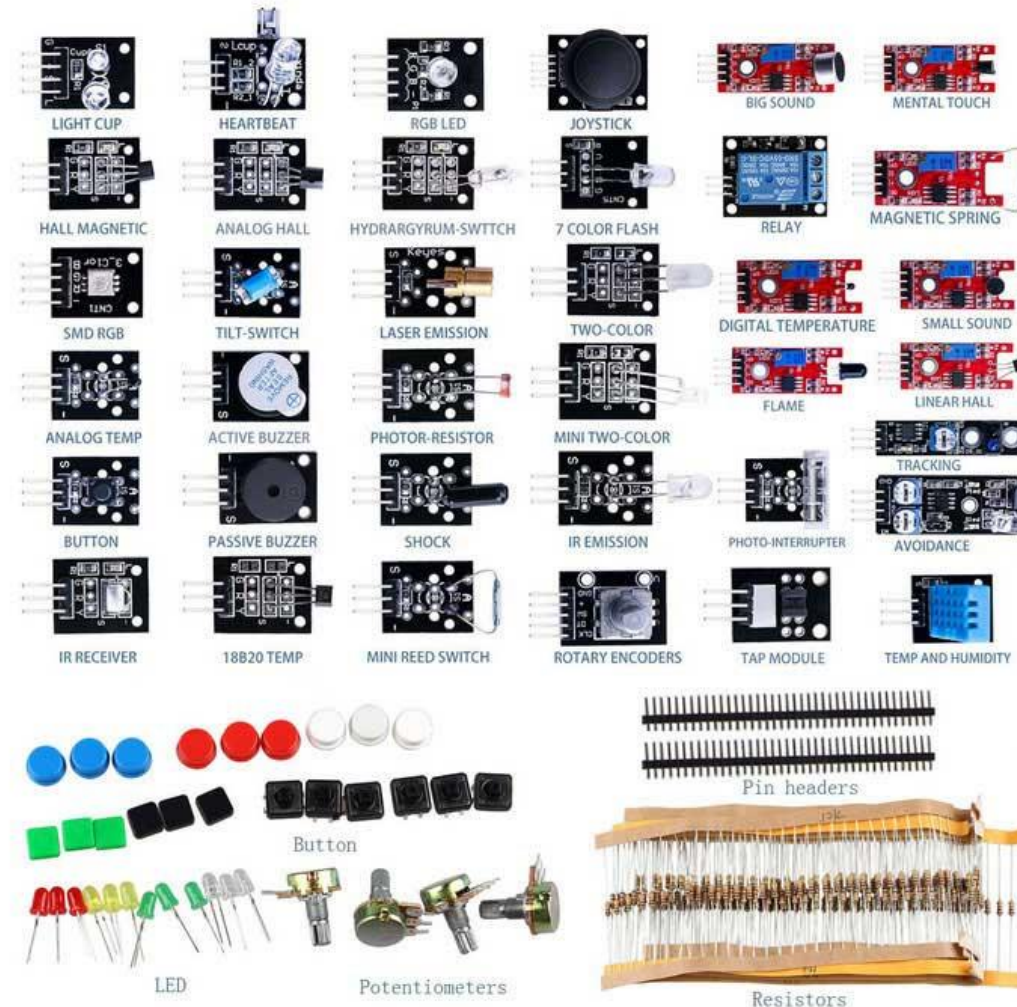
- Una breadboard moderna:



La Breadboard



Esempi di elementi collegabili alla breadboard ed alle porte di Arduino



Parte II

Arduino e Processing

Preparazione di Processing e di Arduino

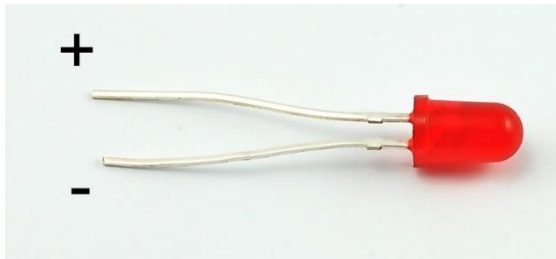
- Installazione della libreria **Arduino (Firmata)** in Processing:
 - Menu *Sketch / Importa Libreria... / Aggiungi Libreria...*
 - Cercare e selezionare **Arduino (Firmata)** di David A. Mellis.
 - Oppure scaricare e installare la libreria manualmente da <https://github.com/firmata/processing/releases/tag/latest>
 - Documentazione:
 - <https://playground.arduino.cc/Interfacing/Processing>
 - Scaricare l'**Arduino IDE** da <https://www.arduino.cc/en/Main/Software>
 - Collegare la board Arduino via USB al PC
 - Lanciare **Arduino IDE** e selezionare
 - dal menu *Strumenti / Scheda* selezionare il tipo di scheda (Arduino/Genuino Uno)
 - dal menu *Strumenti / Porta* selezionare la porta a cui è collegata la scheda
 - dai menu *File / Esempi / Firmata / Standard Firmata*
 - Caricare il programma **Standard Firmata** sulla board Arduino (*Sketch / Carica*, o Ctrl+U o pulsante freccia a destra).

API: un'occhiata all'interfaccia di programmazione

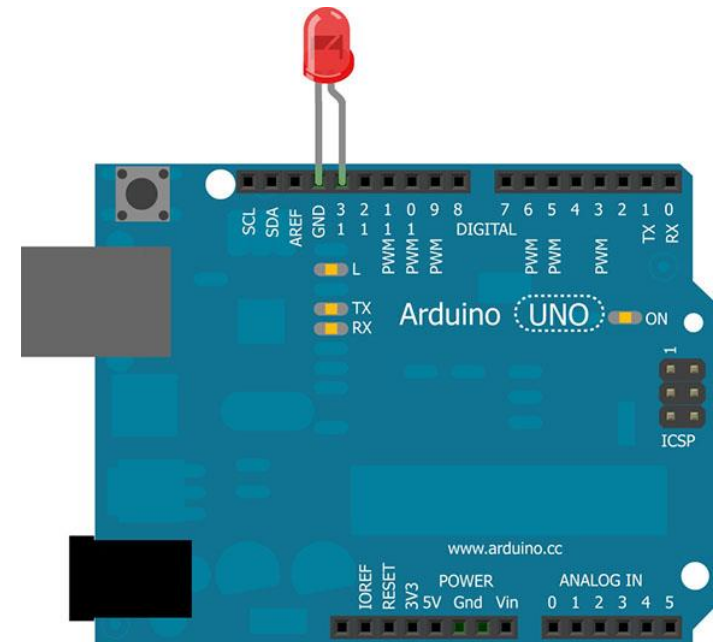
- **Arduino.list():** lista di dispositivi seriali disponibili sul PC (se Arduino è collegato sarà uno di questi).
- **Arduino(parent, name, rate):** crea un oggetto di tipo Arduino (parent assumerà il valore this, name sarà il nome della porta seriale corrispondente e rate sarà la velocità della connessione, e.g., 57600).
- **pinMode(pin, mode):** imposta un digital pin come input, output, o servo mode (Arduino.INPUT, Arduino.OUTPUT o Arduino.SERVO).
- **digitalRead(pin):** restituisce il valore di un digital pin, ovvero, Arduino.LOW o Arduino.HIGH (il pin deve essere impostato come input).
- **digitalWrite(pin, value):** invia Arduino.LOW o Arduino.HIGH ad un digital pin.
- **analogRead(pin):** restituisce il valore di un analog input (da 0 a 1023).
- **analogWrite(pin, value):** invia un valore analogico (PWM wave) ad un digital pin che supporta questa modalità (pin 3, 5, 6, 9, 10 e 11); il valore può variare da 0 (sempre off) a 255 (sempre on).
- **servoWrite(pin, value):** invia un valore ad un servo motore; il valore può variare da 0 a 180.

Sketch 1 – individuare ed inizializzare Arduino

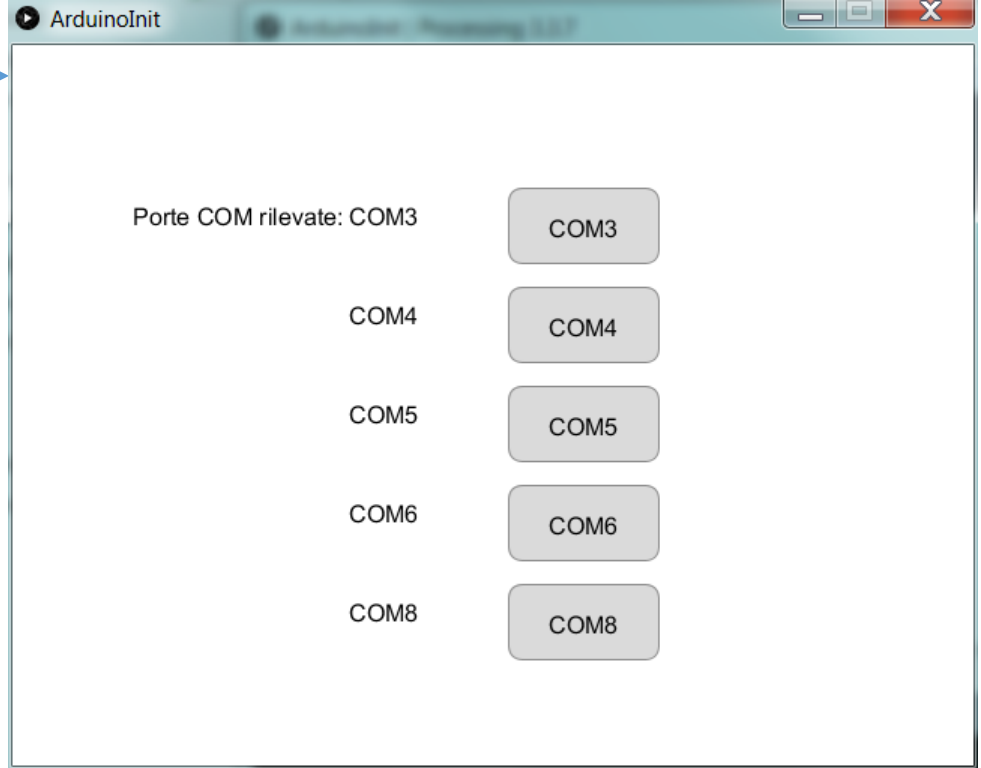
- Utilizzeremo un LED direttamente collegato alla scheda.
- L'estremità più corta del LED rappresenta il catodo (polo negativo), mentre la più lunga rappresenta l'anodo (polo positivo):



- Colleghiamo il catodo al pin GND...
- ... e l'anodo al vicino pin 13



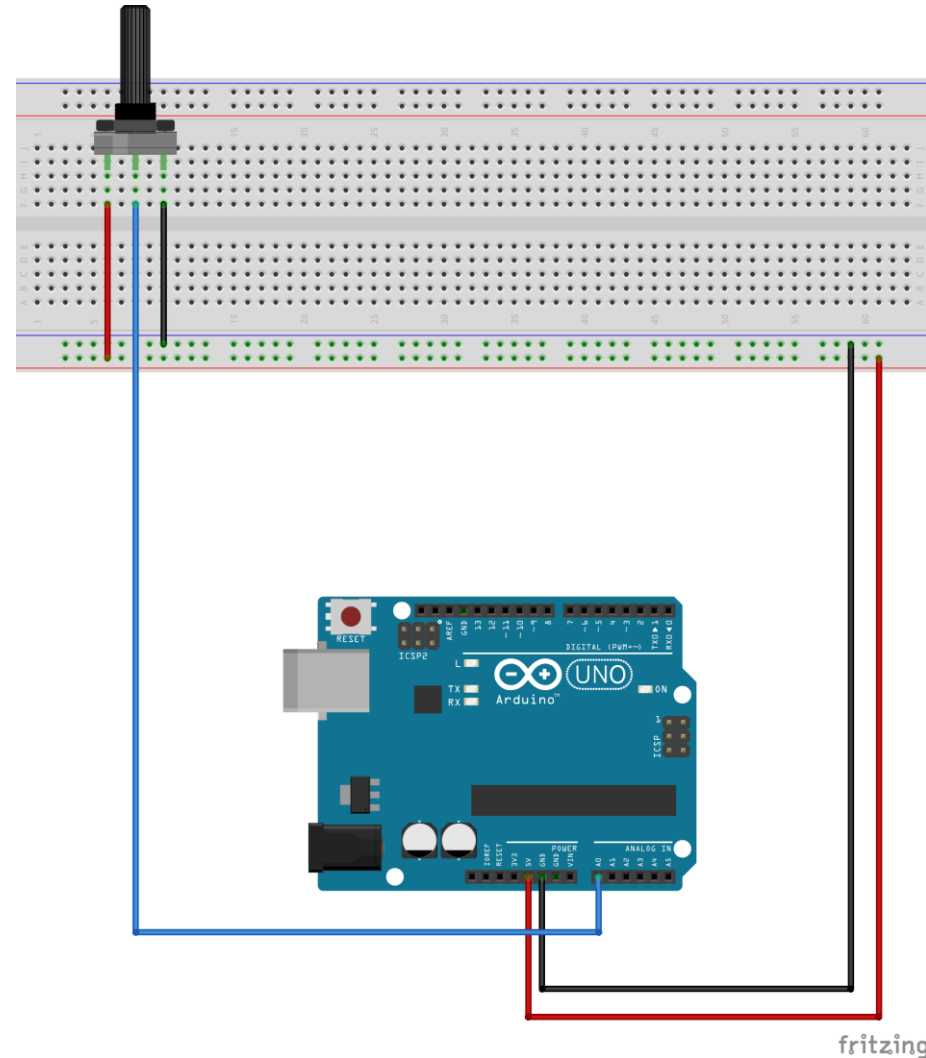
Sketch 1 – individuare ed inizializzare Arduino

- Lo sketch si chiama **ArduinoInit** e, lanciandolo, presenta la seguente interfaccia: 
- Cliccando sui pulsanti il programma tenta di utilizzare la corrispondente porta per accendere il LED collegato ad Arduino per circa 5 secondi:

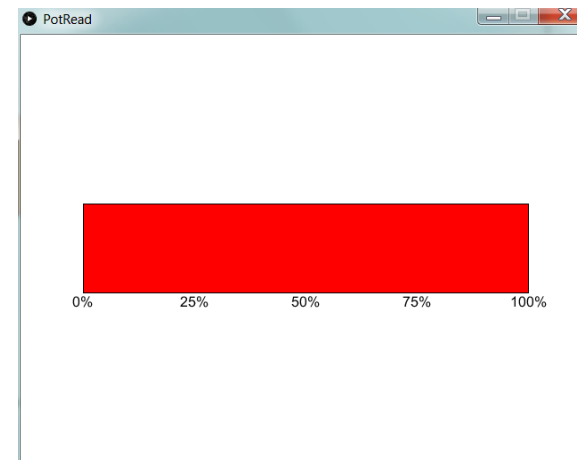
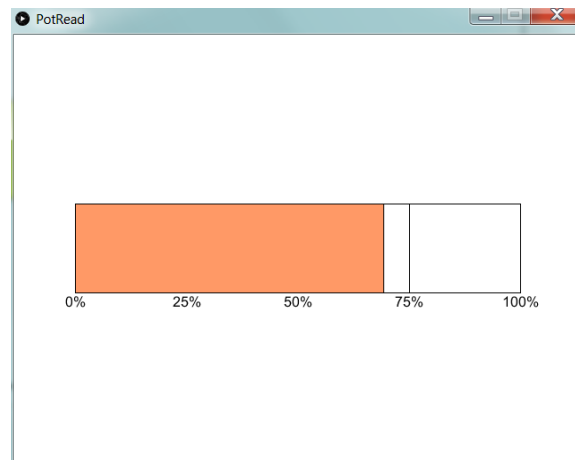
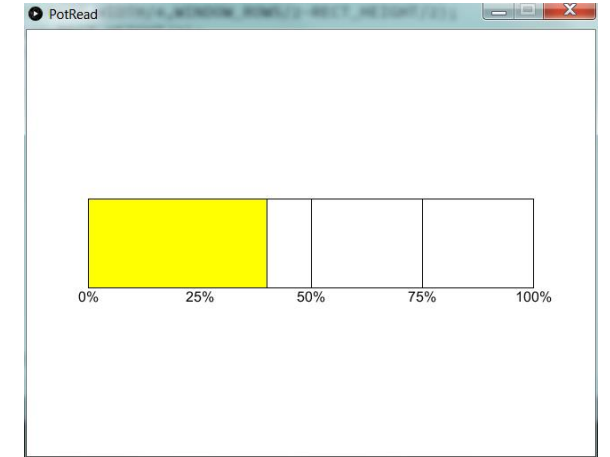
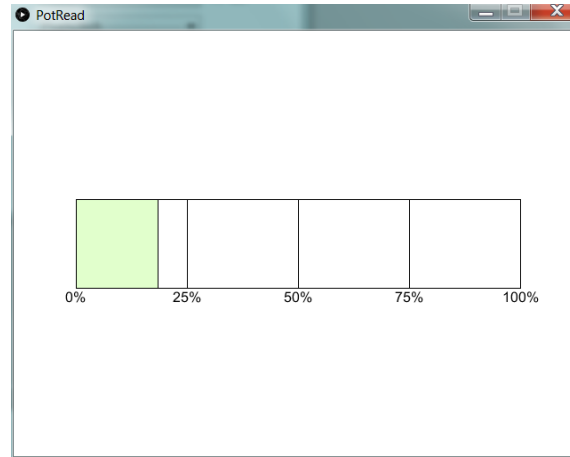
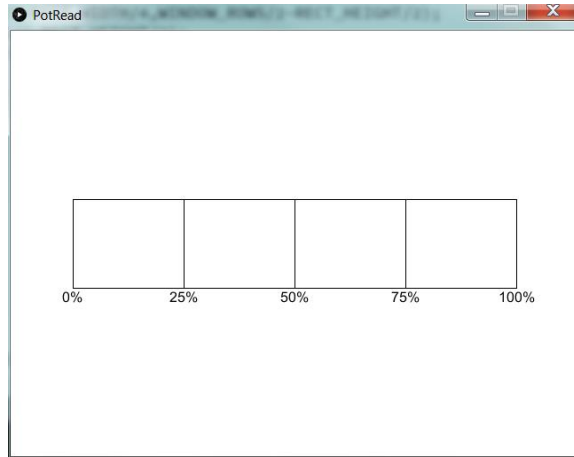
```
void mousePressed() {  
  for(int i=0; i<devices.length; i++) {  
    if(pulsanti.get(i).MouseIsOver()) {  
      println("Pulsante n."+(i+1));  
      try {  
        if(arduino[i]==null)  
          arduino[i] = new Arduino(this, devices[i], 57600);  
        arduino[i].pinMode(13, Arduino.OUTPUT);  
        arduino[i].digitalWrite(13, Arduino.HIGH);  
        delay(5000);  
        arduino[i].digitalWrite(13, Arduino.LOW);  
      }  
      catch(Exception e) {  
        println("Errore: "+e.toString());  
      }  
      break;  
    }  
  }  
}
```


Sketch 2 – monitorare un potenziometro

- Idea: rappresentare il valore assunto da un potenziometro mediante un rettangolo che si riempie di un colore diverso a seconda del livello raggiunto dal segnale.



Sketch 2 – monitorare un potenziometro



Sketch 2 – monitorare un potenziometro

- Parte principale del codice:

```
if(arduinoCheck) {  
    int val=arduino.analogRead(potpin); // Legge il valore del sensore (un intero fra 0 e 1023)  
    int mappedVal=(int)map(val,0,1023,0,RECT_WIDTH); // Mappa il valore sulla lunghezza del rett.  
    println("*****"+mappedVal+"*****");  
  
    if(mappedVal<=RECT_WIDTH/4) // Valore inferiore o uguale al 25%  
        pg.fill(225,255,204); // Verde chiaro  
    if(mappedVal>RECT_WIDTH/4 && mappedVal<=RECT_WIDTH/2) // Valore compreso fra 25% e 50%  
        pg.fill(255,255,0); // Giallo  
    if(mappedVal>=RECT_WIDTH/2 && mappedVal<=RECT_WIDTH/4*3) // Valore fra il 50% ed il 75%  
        pg.fill(255,153,102); // Arancione  
    if(mappedVal>RECT_WIDTH/4*3) // Valore maggiore al 75%  
        pg.fill(255,0,0); // Rosso  
    pg.rect(WINDOW_COLS/2-RECT_WIDTH/2+mappedVal/2,WINDOW_ROWS/2,mappedVal,RECT_HEIGHT);  
}
```

Parte III

Progetti avanzati

Progetto 1: usare un sensore PIR

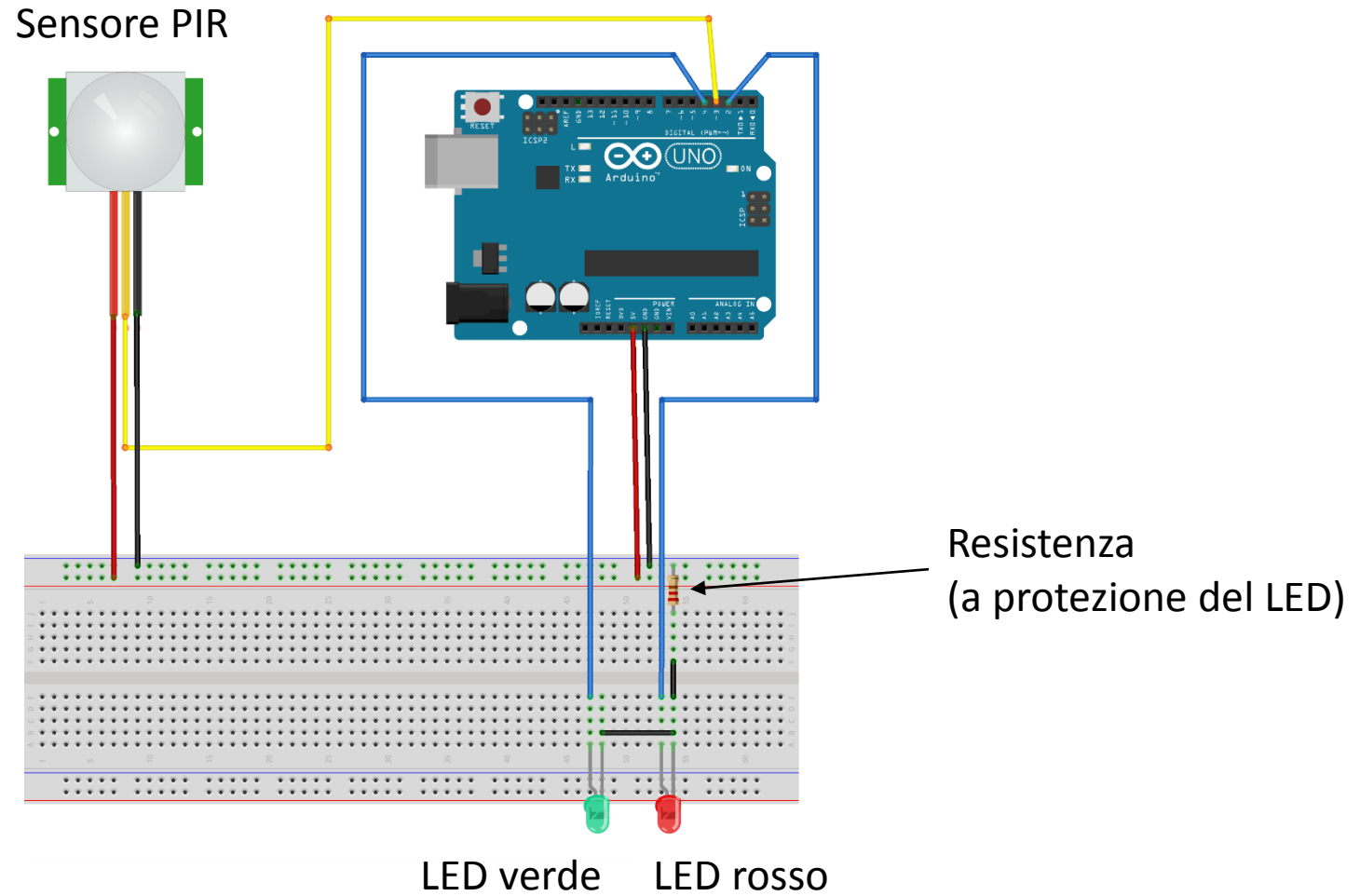
- Usiamo un sensore PIR (Passive InfraRed) per realizzare un rilevatore di movimento.
- Un sensore PIR misura i raggi infrarossi irradiati nel suo campo d'azione.



- Accenderemo un led quando il sensore rileverà un oggetto.

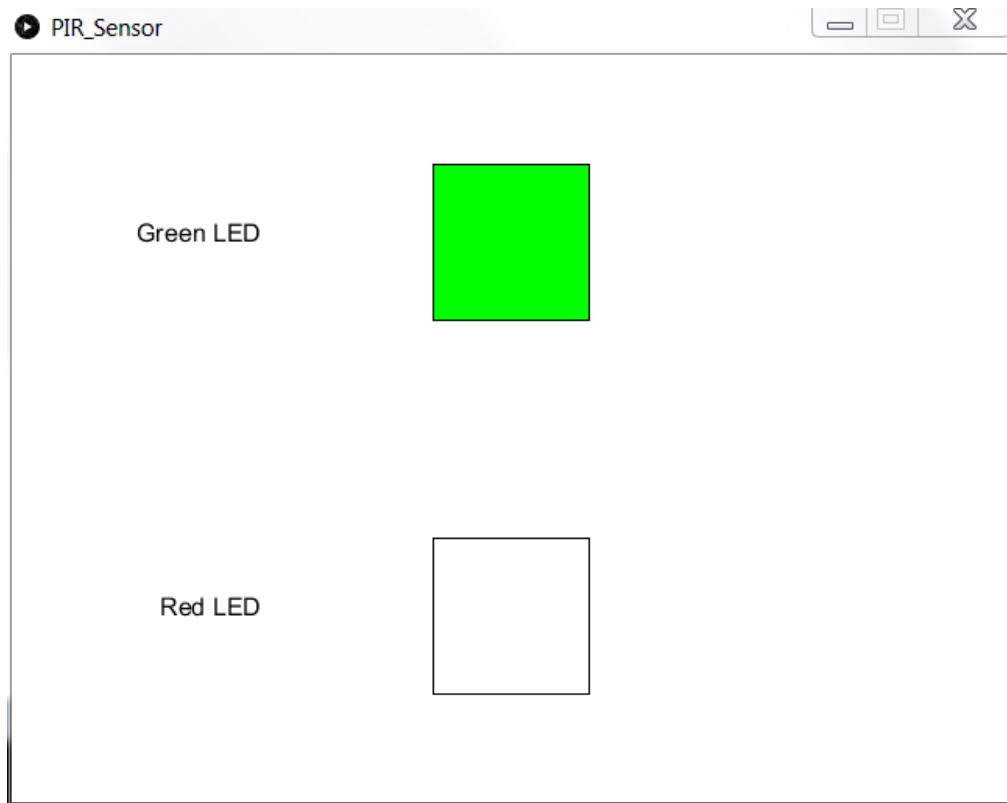
Schema del progetto

Colore cavi:
Rosso: tensione (+5V)
Nero: terra (ground)
Blu/Giallo: segnale

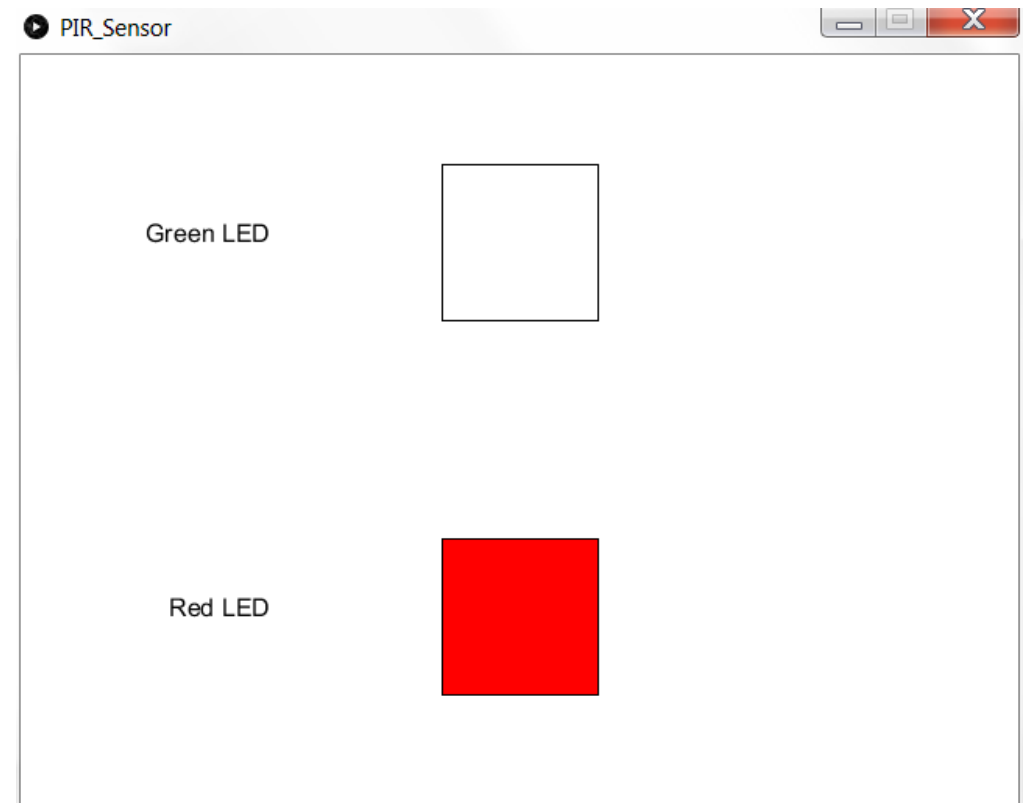


Funzionamento

Fase 1: LED verde acceso, nessun movimento rilevato



Fase 2: LED rosso acceso, movimento rilevato



Uno sguardo al codice (I)

```
import processing.serial.*; ← Importiamo le librerie necessarie
import cc.arduino.*;
...
int pirPin = 3; // PIN collegato al sensore PIR
int greenLedPin = 4; // PIN collegato al LED verde
int redLedPin = 2; // PIN collegato al LED rosso
Arduino arduino; // oggetto di tipo Arduino: serve a
dialogare con la board
boolean arduinoCheck = false; // booleano: true ->
Arduino collegato, false -> Arduino assente
```


Uno sguardo al codice (II)

```
void setup() {  
  ...  
  try {  
    println((Object[])Arduino.list());  
    arduino = new Arduino(this, Arduino.list()[2], 57600); // inizializza l'oggetto arduino  
    arduino.pinMode(greenLedPin, Arduino.OUTPUT); // imposta il LED verde come output PIN  
    arduino.digitalWrite(greenLedPin, Arduino.LOW); // spegne il LED verde  
    arduino.pinMode(redLedPin, Arduino.OUTPUT); // imposta il LED rosso come output PIN  
    arduino.digitalWrite(redLedPin, Arduino.LOW); // spegne il LED rosso  
    arduino.pinMode(pirPin, Arduino.INPUT); // imposta il sensore PIR come input PIN  
    arduinoCheck=true;  
  }  
  catch(Exception e) {  
    println("Error: "+e.toString());  
    arduinoCheck=false;  
  }  
  
  delay(5000); // aspettiamo 5 secondi in modo che tutto venga inizializzato correttamente  
}
```

Stampa in console qualcosa come:
COM3 COM4 COM5 COM6 COM8

Se Arduino corrisponde a COM5
nell'elenco precedente

Uno sguardo al codice (III)

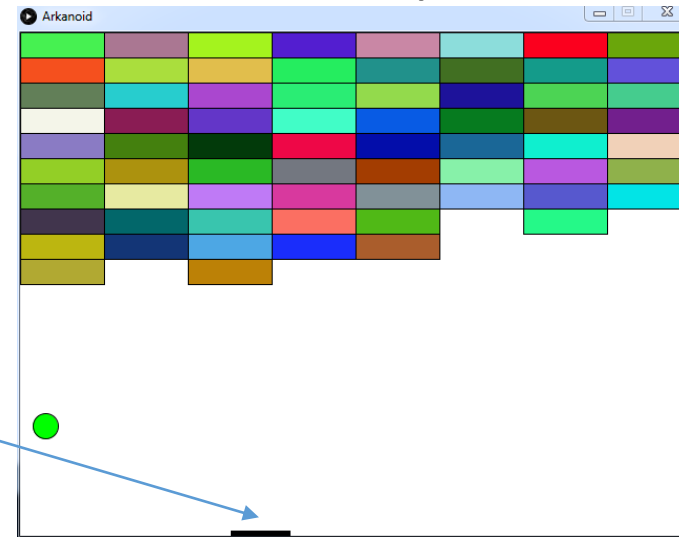
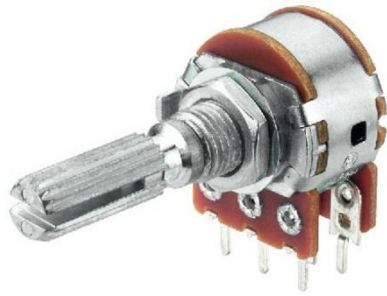
```
void draw() {  
  ...  
  if(arduinoCheck) {  
    int pirSignal=arduino.digitalRead(pirPin);  
  
    if(pirSignal==Arduino.HIGH) {  
      arduino.digitalWrite(redLedPin, Arduino.HIGH);  
      arduino.digitalWrite(greenLedPin, Arduino.LOW);  
      ...  
    }  
    else {  
      arduino.digitalWrite(greenLedPin, Arduino.HIGH);  
      arduino.digitalWrite(redLedPin, Arduino.LOW);  
      ...  
    }  
  }  
  ...  
}
```

Se rilevo un movimento,
accendo il led rosso
e spengo il led verde...

...altrimenti, faccio il viceversa.

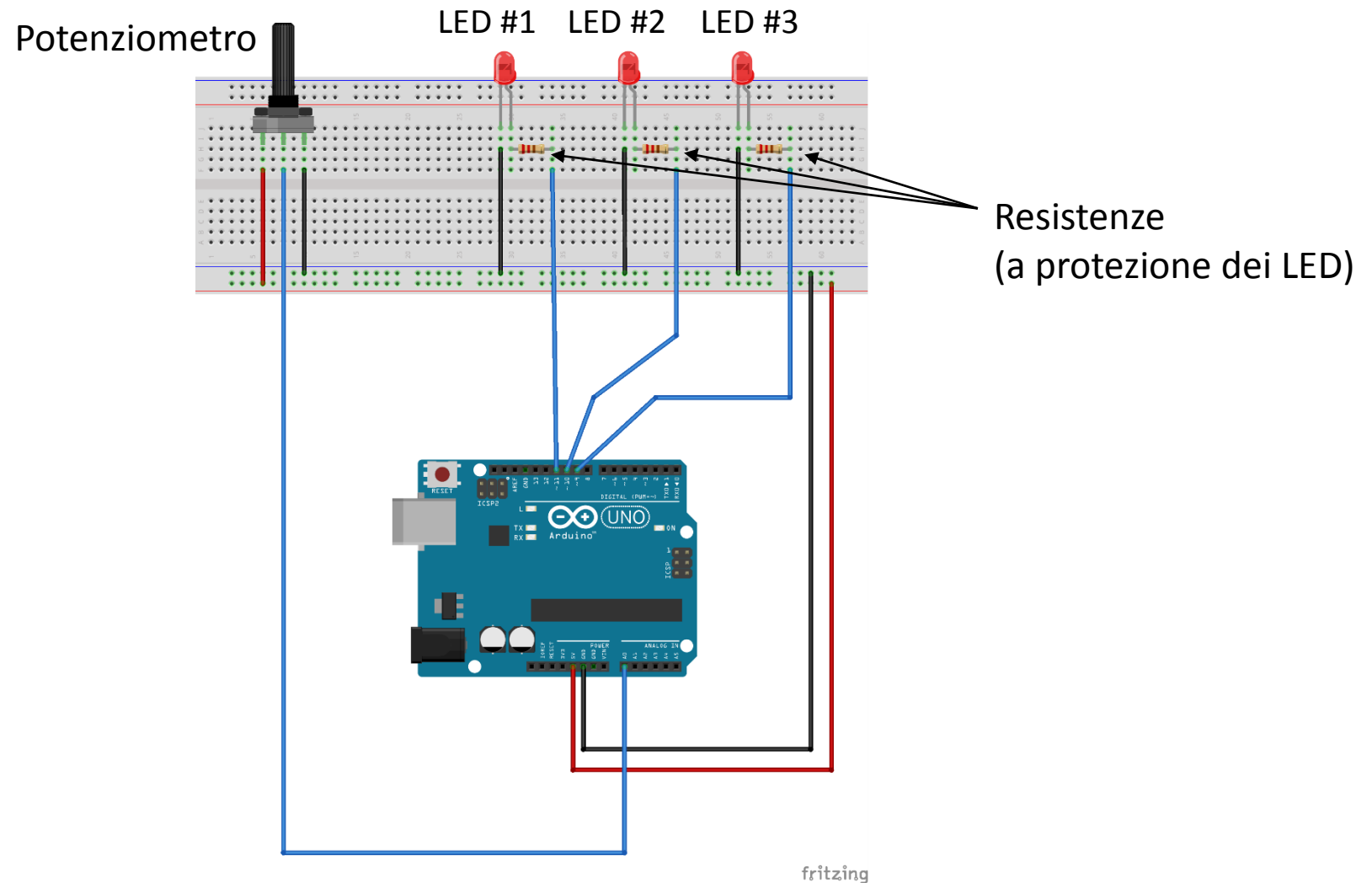
Progetto 2: usare un potenziometro come "gamepad" per Arkanoid

- Un potenziometro è un partitore di tensione regolabile.
- Può essere utilizzato come un reostato, ovvero, permette di ottenere una resistenza variabile.
- L'idea è di tradurre il valore in uscita (ottenuto ruotando la manopola) in uno spostamento per la paletta del gioco Arkanoid.



- Useremo anche tre LED per rappresentare le vite rimanenti al giocatore (LED spento=vita disponibile, LED acceso=vita persa).

Schema del progetto



Uno sguardo al codice (I)

```
import processing.serial.*;
import cc.arduino.*;

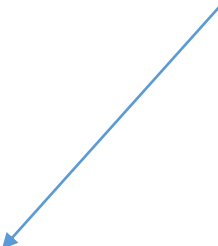
...

int potpin = 0; // potenziometro
int firstled = 9, secondled = 10, thirdled = 11; //
i tre led che rappresentano le vite del giocatore
Arduino arduino; // oggetto di tipo Arduino: serve a
dialogare con la board

boolean arduinoCheck = false; // booleano: true ->
Arduino collegato, false -> Arduino assente
```

Uno sguardo al codice (II)

Sostituire con il numero di porta corretto sul proprio PC.



```
void setup() {  
  ...  
  try {  
    println((Object[])Arduino.list());  
    arduino = new Arduino(this, Arduino.list()[4], 57600); // inizializza l'oggetto arduino  
    arduino.pinMode(firstled, Arduino.OUTPUT); // imposta il primo LED come output PIN  
    arduino.digitalWrite(firstled, Arduino.LOW); // spegne il primo LED  
    arduino.pinMode(secondled, Arduino.OUTPUT); // imposta il secondo LED come output PIN  
    arduino.digitalWrite(secondled, Arduino.LOW); // spegne il secondo LED  
    arduino.pinMode(thirdled, Arduino.OUTPUT); // imposta il terzo LED come output PIN  
    arduino.digitalWrite(thirdled, Arduino.LOW); // spegne il terzo LED  
    arduinoCheck = true;  
  }  
  catch(Exception e) {  
    println("Error: "+e.toString());  
    arduinoCheck = false;  
  }  
  ...  
}
```

Uno sguardo al codice (III): righe 255—274 (funzione moveBall())

```
...
if(ballY+ballRadius>=WINDOW_ROWS) { // rimbalzo sul fondo -> vita persa (accendo un LED diverso
per ognuna delle tre vite perse)
    life--;
    print("Vite rimanenti: "+life+"\n");
    if(arduinoCheck) {
        switch(life) {
            case 0: // 0 vite, game over -> tutti i LED accesi
                // Esercizio: scrivere il codice per accendere i LED
                break;
            case 1: // 1 vita rimasta -> primi due LED accesi, terzo LED spento
                // Esercizio: scrivere il codice per accendere/spegnere i LED
                break;
            case 2: // 2 vite rimaste -> primo LED acceso, secondo e terzo LED spenti
                // Esercizio: scrivere il codice per accendere/spegnere i LED
                break;
        }
    }
}
...

```

Uno sguardo al codice (IV): righe 291—299 (funzione draw())

```
if(arduinoCheck) {  
    // lettura del valore del potenziometro  
    int val=...; // Completare il codice  
    // mappatura del valore in termini di ascisse della finestra  
    int mappedVal=(int)map(val,0,1023,...); // Completare  
    if(DEBUG) println("*****"+mappedVal+"*****");  
  
    if(matchStatus==1) { // se la partita è in corso  
        paddleX=...; // Completare: spostare la paletta  
    }  
}
```

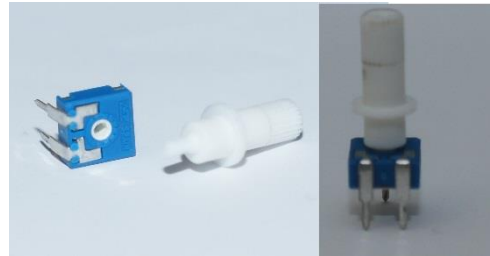

Uno sguardo al codice (V): righe 323—337 (funzione keyPressed())

```
...
if(keyCode==KeyEvent.VK_F1) { // nuova partita: l'utente ha premuto F1
    ...
    life=3;
    points=0;
    ...
    if(arduinoCheck) { // spengo i 3 LED: le vite vengono ripristinate.
        // Completare il codice
    }
}
...

```

Progetto 3: gestire un motore elettrico

- Useremo Arduino e Processing per accendere/spegnere e gestire i giri di un motore elettrico per droni.
- Per regolare i giri del motore, useremo ancora una volta un potenziometro.



- I rimanenti materiali necessari al progetto sono:
 - un motore brushless (senza spazzole) a corrente continua,
 - un ESC (Electronic Speed Controller) per pilotare il motore,
 - una batteria LiPo (ai polimeri di Litio), per alimentare ESC e motore.
 - un LED che accenderemo ogni volta che il motore girerà ai regimi più alti (da metà in su).

Schema del progetto

Colore cavi:
Rosso: tensione (+5V)
Nero: terra (ground)
Blu/giallo: segnale

Potenziometro

Resistenza
(a protezione del LED)

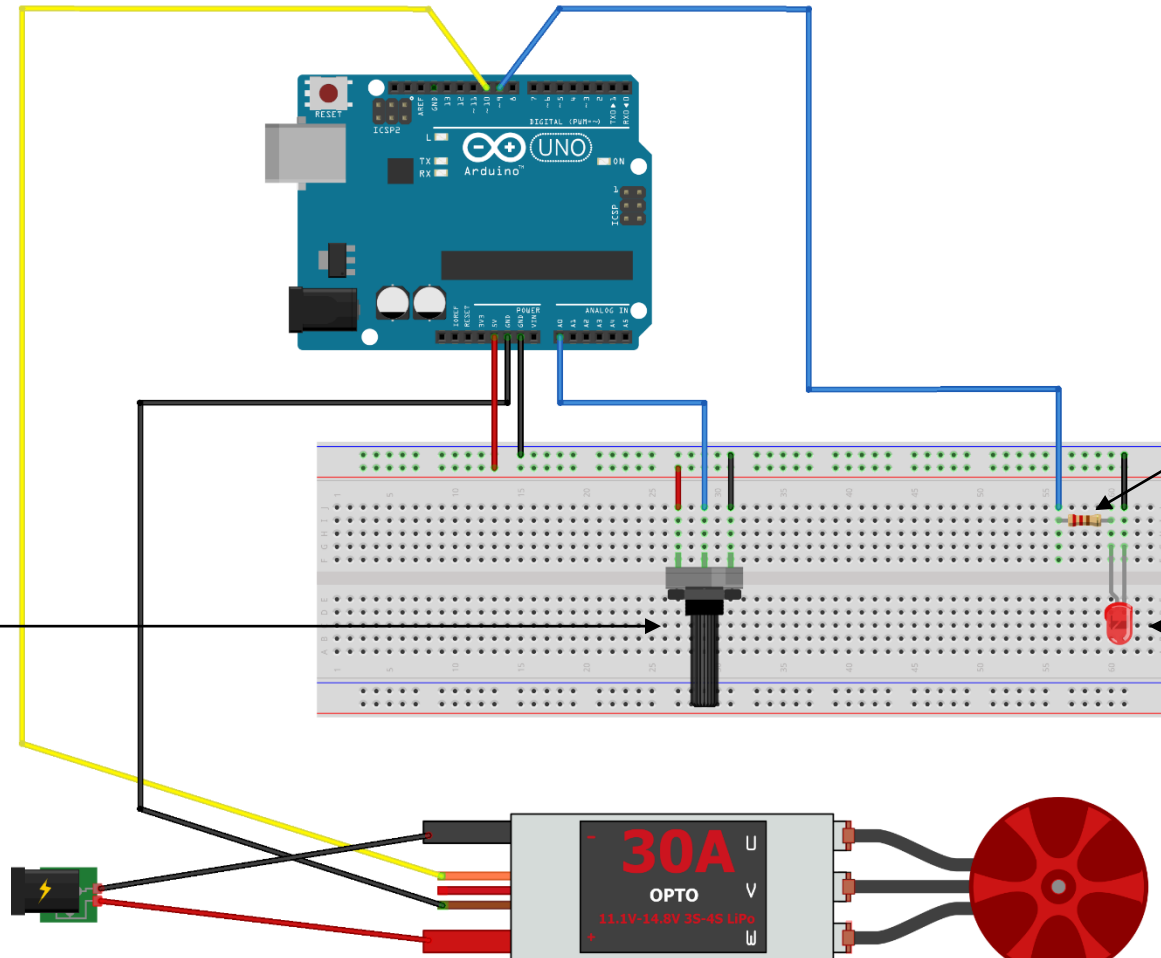
LED

Batteria LiPo

ESC

Motore brushless

fritzing



Uno sguardo al codice (I)

```
import processing.serial.*;
import cc.arduino.*;

...

static final int OFF_THRESHOLD=54; // soglia minima di
RPM


static final int MIDDLE_THRESHOLD=117; // metà potenza

static final float RPM=920*12.6; // Rotazioni per
minuto del motore a 12.6 V

static final float RPS=RPM/60.0; // Rotazioni per
secondo del motore a 12.6 V
```

Uno sguardo al codice (II)

Sostituire con il numero di porta corretto sul proprio PC.



```
void setup() {
  ...
  try {
    println((Object[])Arduino.list());
    arduino = new Arduino(this, Arduino.list()[0], 57600);
    arduino.pinMode(ledPin, Arduino.OUTPUT);           // LED -> output PIN
    arduino.digitalWrite(ledPin, Arduino.LOW);         // spengo il LED
    arduino.pinMode(potPin, Arduino.INPUT);           // potenziometro -> input PIN
    arduino.pinMode(servoPin, Arduino.SERVO);         // motore -> servo
    arduinoCheck=true;
  }
  catch(Exception e) {
    println("Error: "+e.toString());
    arduinoCheck=false;
  }

  delay(10000); // Attendiamo 10 secondi: il motore e l'ESC devono essere inizializzati
}
```

Uno sguardo al codice (III)

```
void draw() {
  int val=0;

  if(arduinoCheck) {
    val=arduino.analogRead(potPin); // leggo il potenziometro
    val = (int)map(val, 0, 1023, 0, 179); // mappo il valore letto nella scala del servo-motore
    println(val);
    arduino.servoWrite(servoPin,val); // invio il valore mappato al servo-motore
    if(val>=MIDDLE_THRESHOLD) // raggiunta metà potenza, accendo il LED
      arduino.digitalWrite(ledPin,Arduino.HIGH);
    else // altrimenti il LED è spento
      arduino.digitalWrite(ledPin,Arduino.LOW);
  }

  ...

  if(val>OFF_THRESHOLD) // se la soglia minima è stata superata...
    rotAngle+=2*PI*RPS*val/179; // ...imposta l'angolo di rotazione dell'elica su schermo
}
```