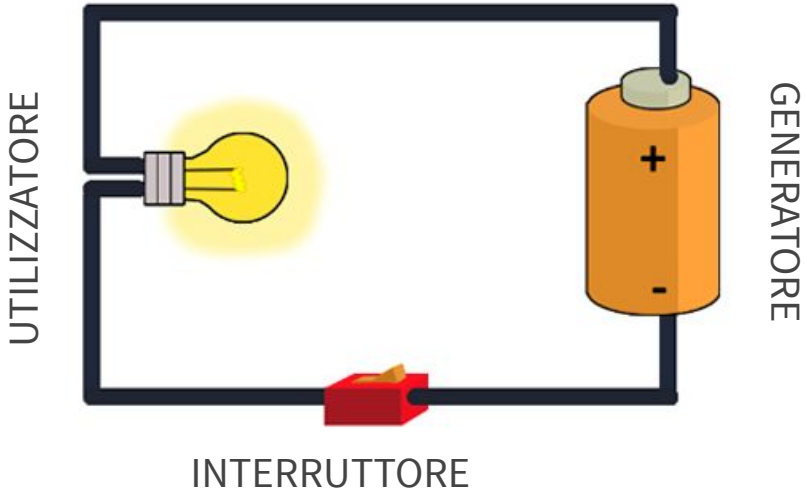
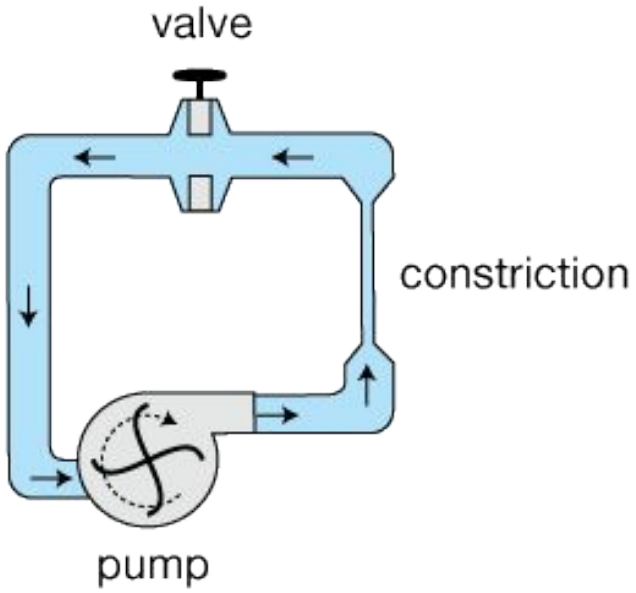
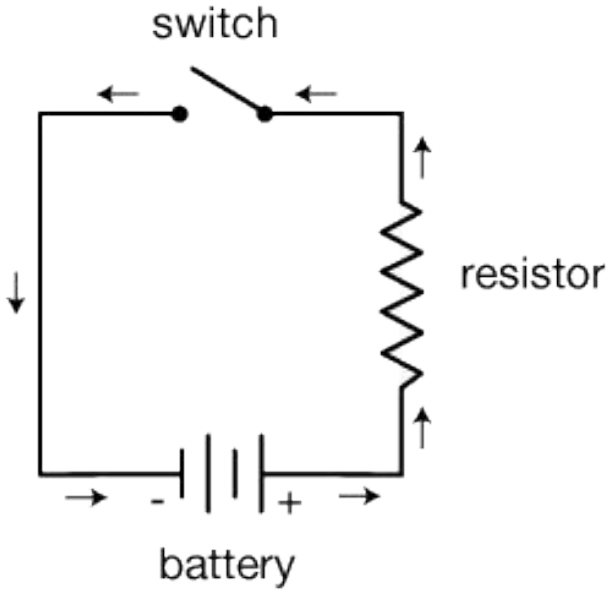
Introduzione (circuiti e legge di Ohm)

Circuiti semplici

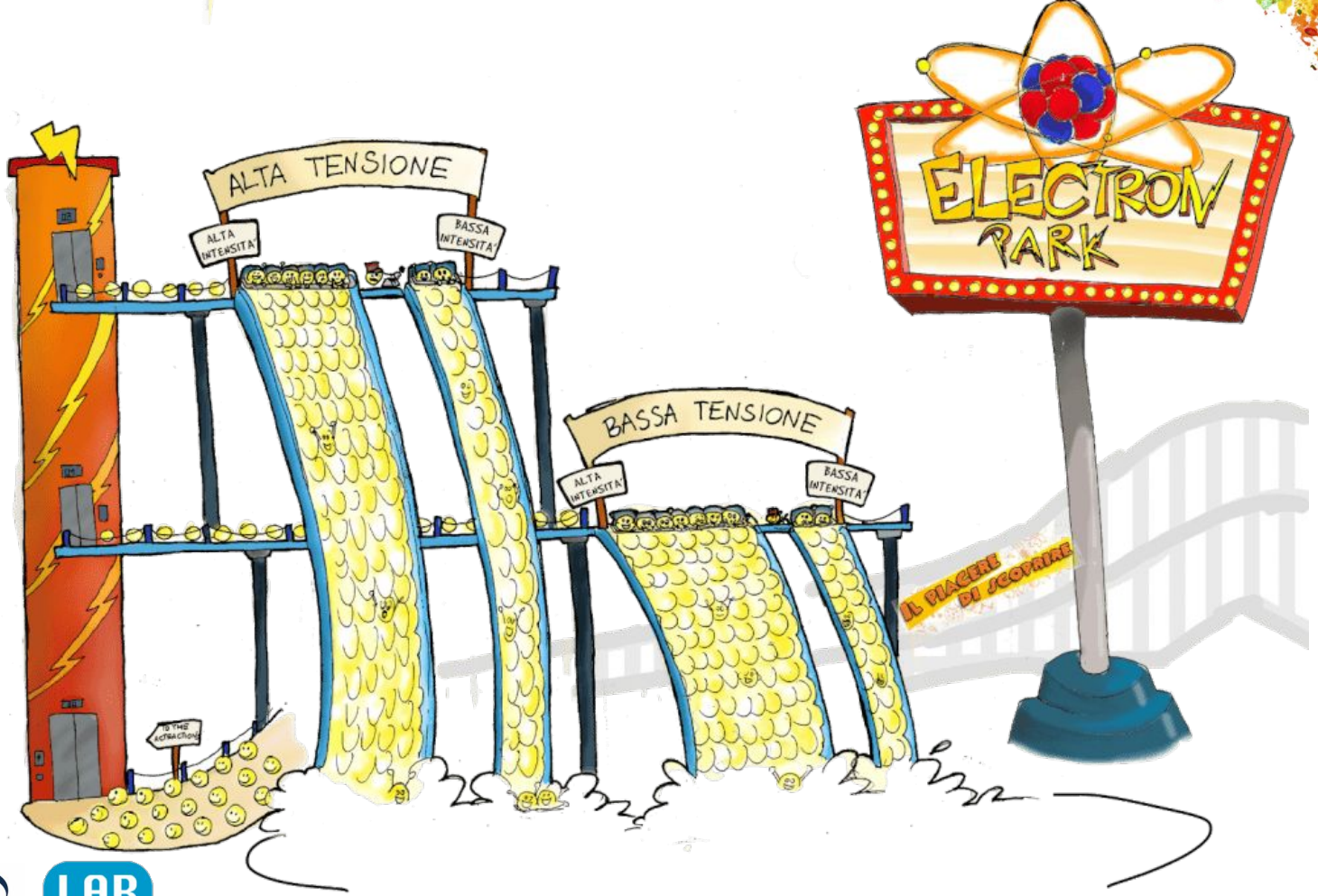


Analogia con i circuiti idraulici





Slides credits: www.nova-aps.it



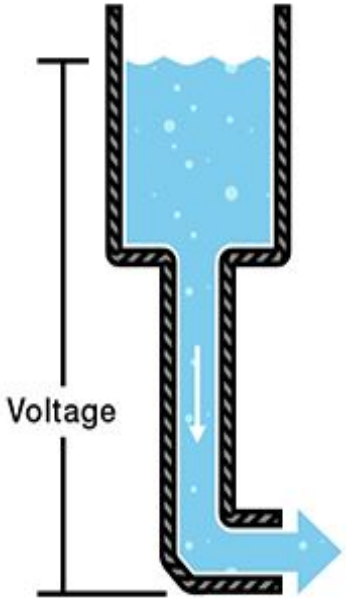
INFN FERRARA

LAB 2GO SCIENZA

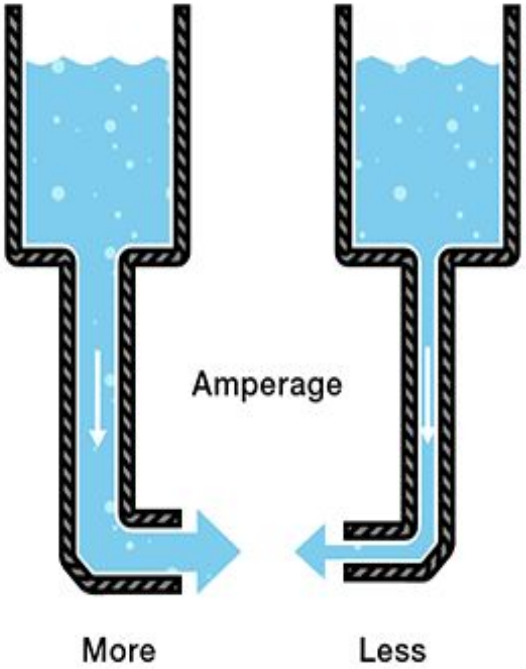
Slides credits: www.nova-aps.it

Analogia con i circuiti idraulici

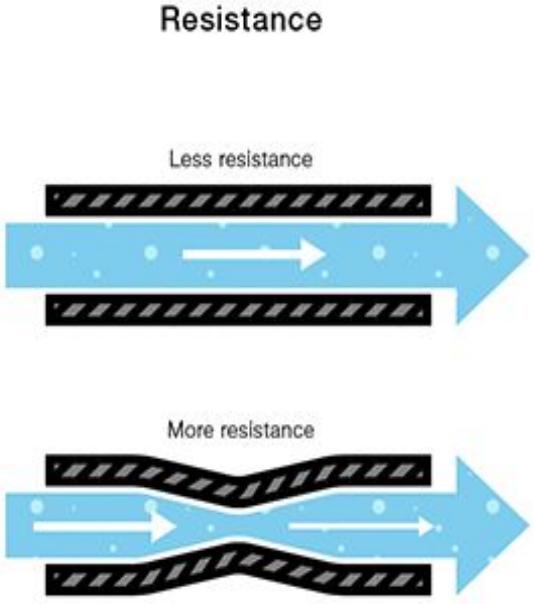
Tensione



Corrente



Resistenza



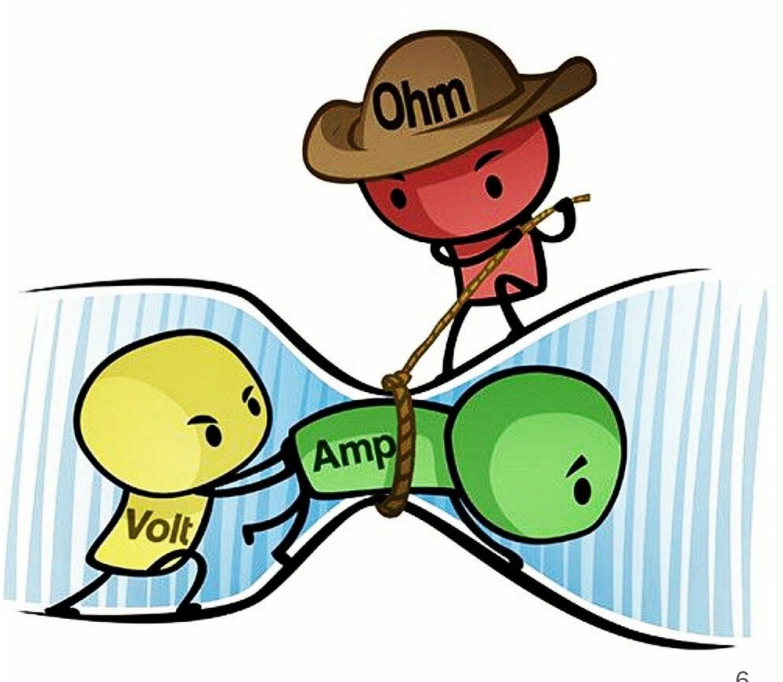
Legge di Ohm

$$\Delta V = R \cdot I$$

V — differenza di potenziale

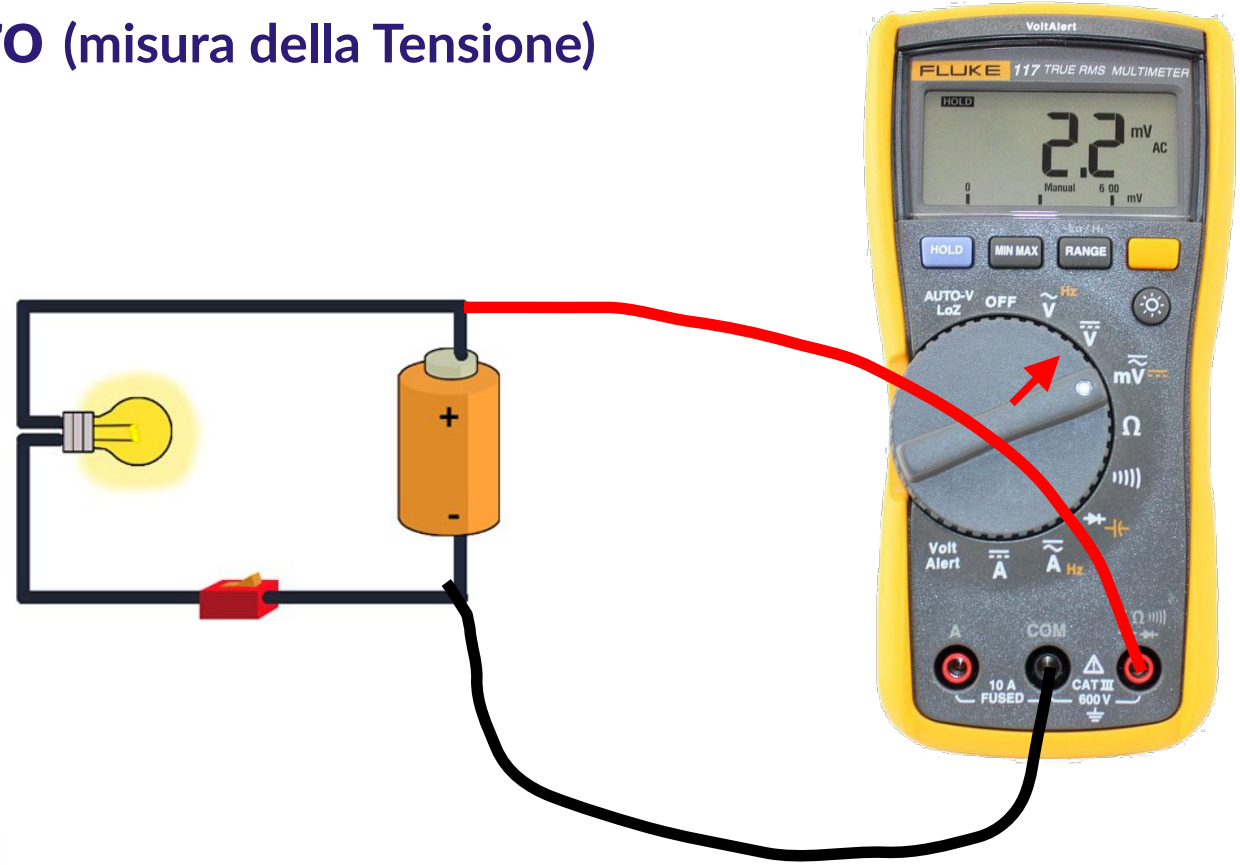
Ω — resistenza

A — corrente

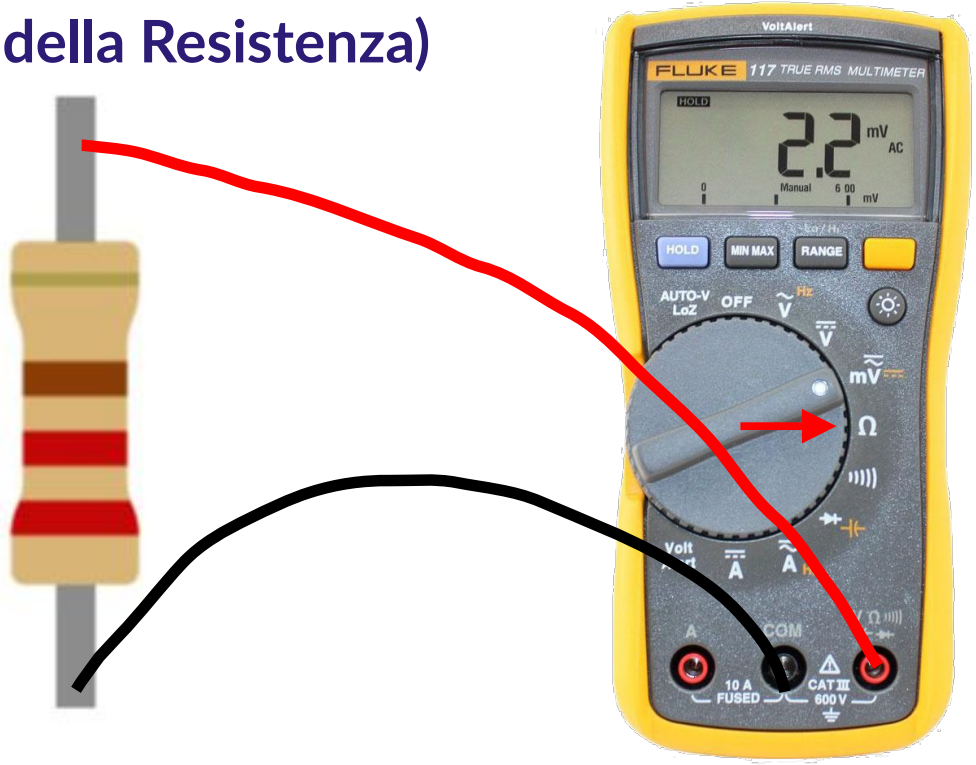


Slides credits: www.nova-aps.it

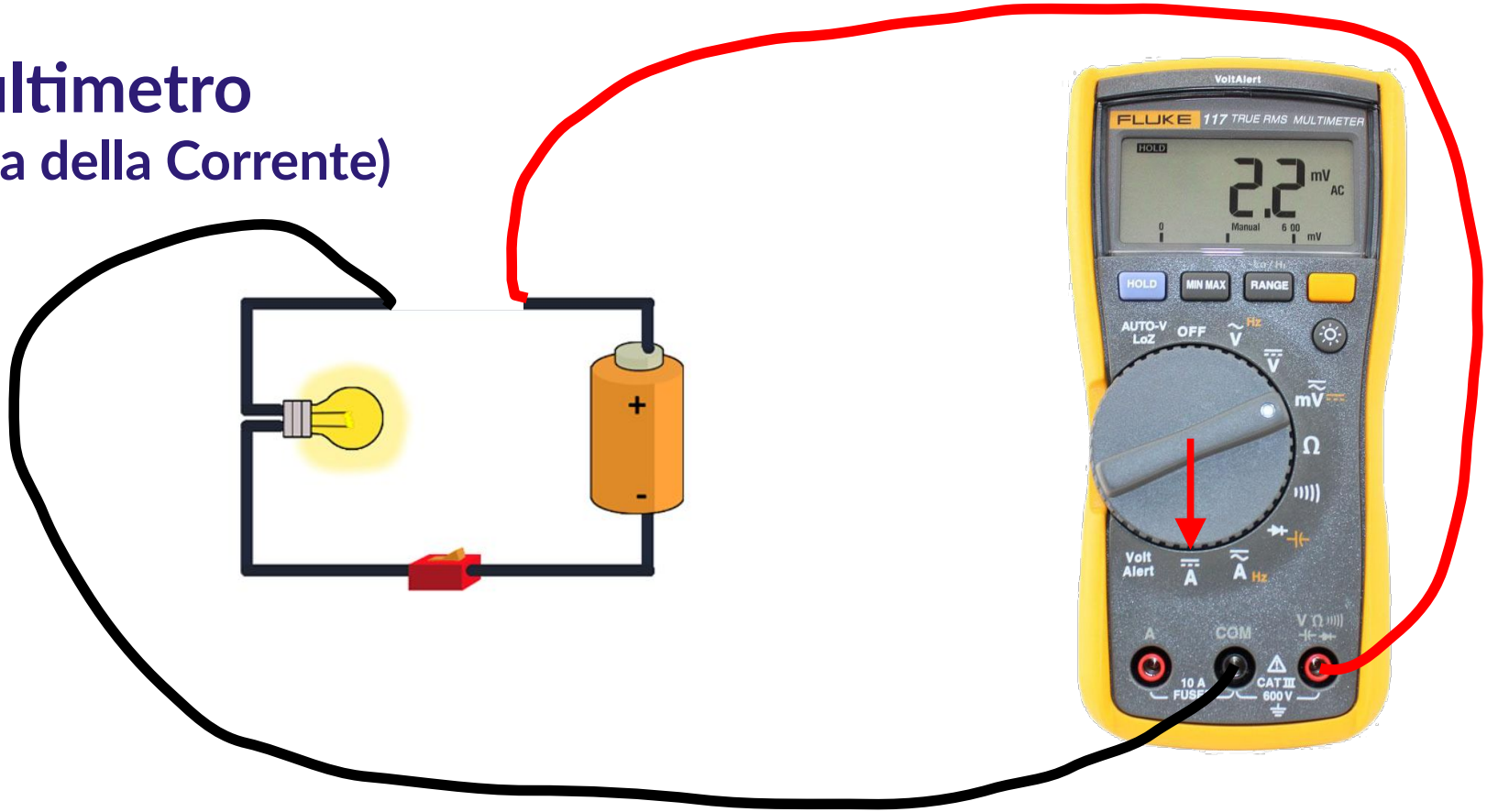
Il Multimetro (misura della Tensione)



Il Multimetro (misura della Resistenza)



Il Multimetro
(misura della Corrente)



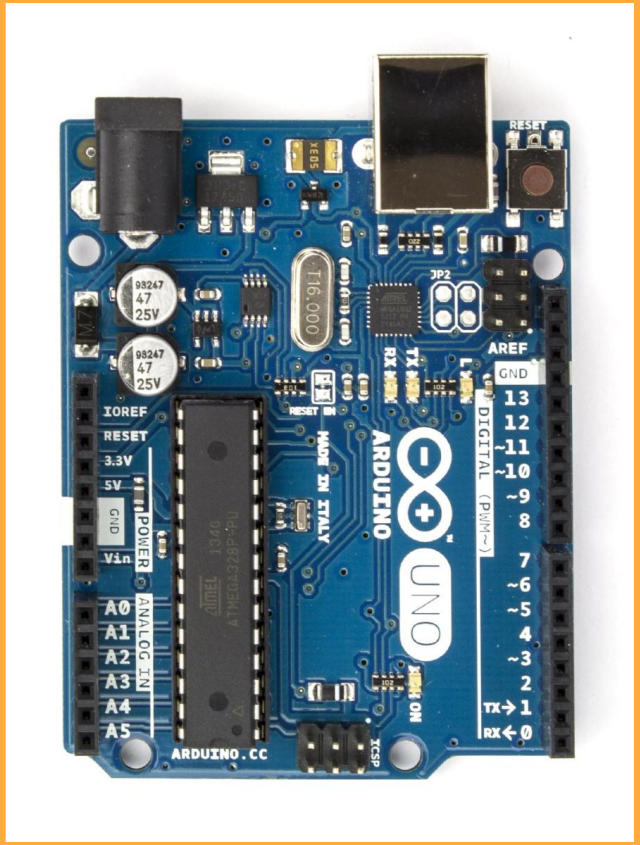
Arduino

INFN FERRARA

LAB 2GO SCIENZA

Cos'è?

www.arduino.cc



Cos'è Arduino?

Arduino è un progetto che comprende un'azienda produttrice di ***open hardware*** e ***software libero***, nonché una community di utilizzatori. L'idea fondamentale del progetto è quella di semplificare la programmazione e l'utilizzo dei *microcontrollori* rendendoli accessibili anche agli utenti non particolarmente esperti di elettronica.

Con il nome Arduino spesso si indicano le diverse schede elettroniche sviluppate all'interno dell'omonimo progetto.

L'idea nasce nel 2003, ed il progetto nel 2005, da alcuni membri dell'Interaction Design Institute di Ivrea come strumento per la **prototipazione rapida**, per scopi hobbistici, **didattici** e professionali.

A cosa serve?

Le schede Arduino sono programmabili, attraverso una una serie di istruzioni (i.e. comandi), scritte nel linguaggio di **programmazione** Arduino (basato su *Wiring*), che vengono poi tradotte in modo da essere "comprensibili" per il *microcontrollore*. Lo strumento di sviluppo software predefinito è la **Arduino IDE**.

Esempio: le schede Arduino sono in grado di **leggere gli input** - luce su un sensore, un dito su un pulsante o un messaggio di Twitter - e **trasformarli in output** - attivare un motore, accendere un LED, pubblicare qualcosa online.

Come lo imparo?

Nel corso degli anni Arduino è stato il "cervello" di migliaia di progetti, dagli oggetti di uso quotidiano a strumenti scientifici complessi.

Una comunità mondiale di produttori - studenti, hobbisti, artisti, programmatori e professionisti - si è riunita attorno a questa piattaforma open source, i loro contributi hanno contribuito a un'incredibile quantità di conoscenza accessibile che può essere di grande aiuto sia per i principianti che per gli esperti.

Forum ufficiale
https://forum.arduino.cc/

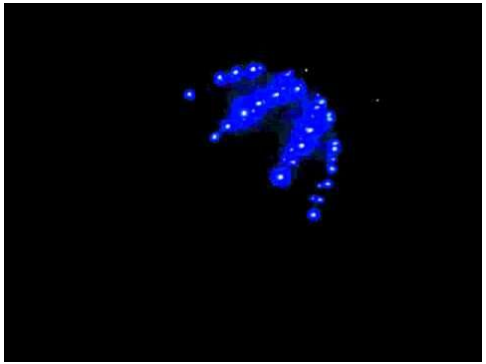
Blog con risorse specifiche (getting started, tutorial, sezione education...)
https://blog.arduino.cc/category/community/

Project HUB - condividere i propri progetti e trarre ispirazione
https://create.arduino.cc/projecthub

Altri siti con idee e progetti:
https://www.hackster.io/arduino

... e molto altro!

IoT pet feeder



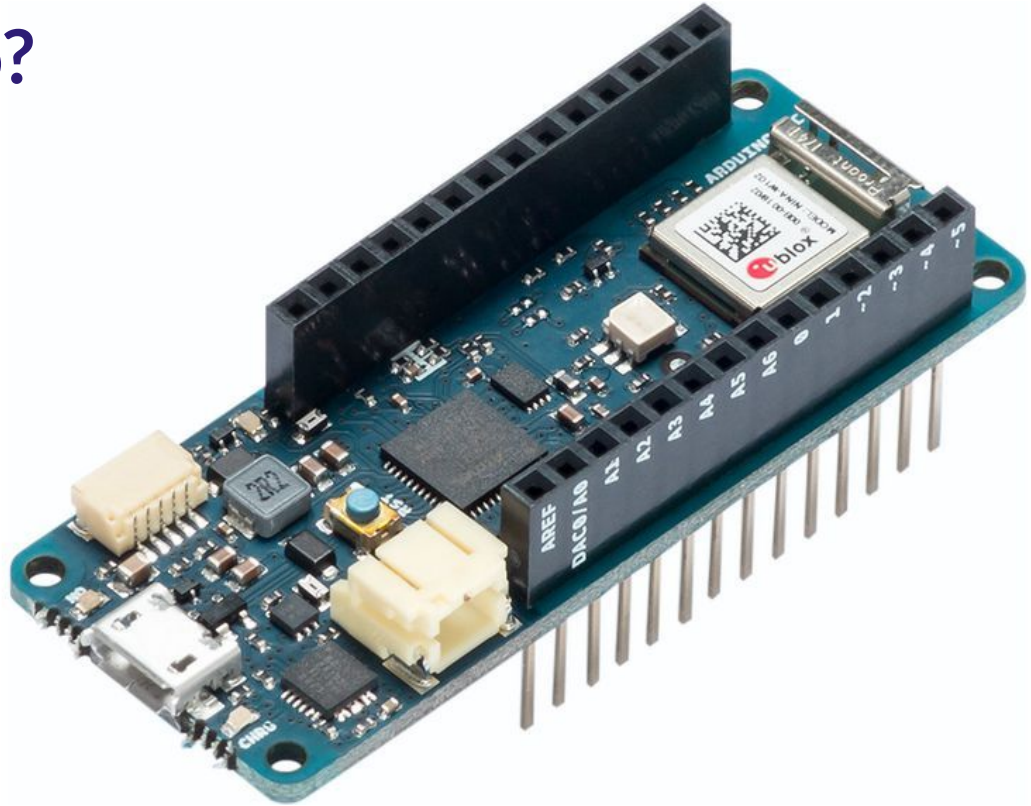
https://www.instructables.com/id/Programmable-LED-Umbrella/

Arduino è nato all'Ivrea Interaction Design Institute come strumento facile per la prototipazione rapida, rivolto a studenti senza esperienza in elettronica e programmazione.

Non appena ha raggiunto una comunità più ampia, la scheda Arduino ha iniziato a cambiare per adattarsi alle nuove esigenze e sfide, differenziando la sua offerta da semplici schede a 8 bit a prodotti per applicazioni **IoT**, **wearables**, **stampa 3D** ed altro ancora.

Tutte le schede Arduino sono completamente open hardware, consentendo agli utenti di costruirle in modo indipendente e adattarle alle loro particolari esigenze. Anche il software è libero e sta crescendo grazie ai contributi degli utenti di tutto il mondo.

Cosa vedo?

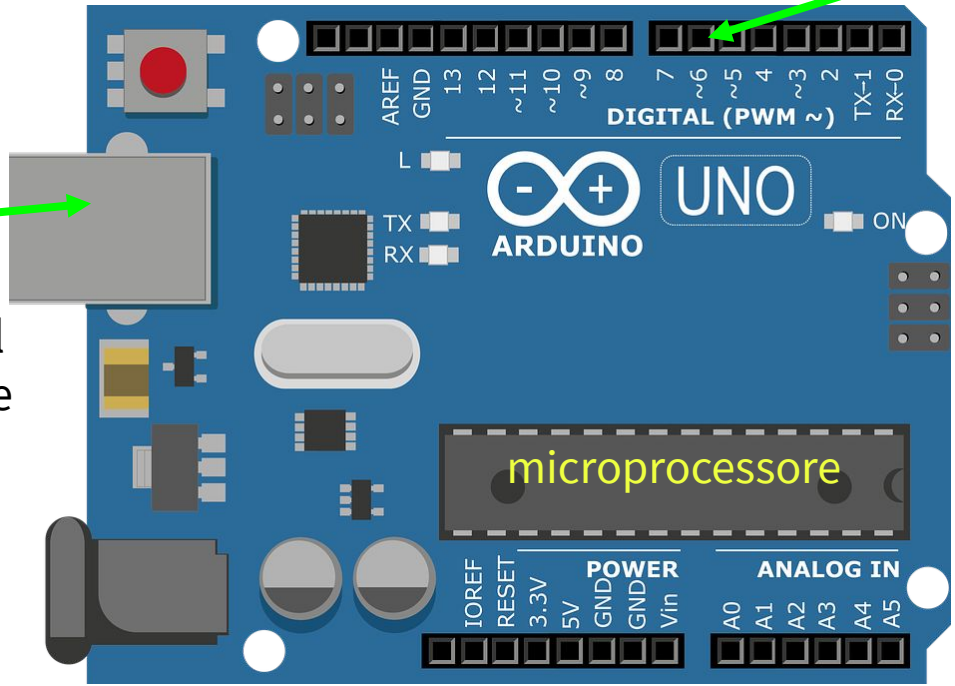


Cosa vedo?


DIGITAL (PWM~)
UNO
ARDUINO
SMDEDITION
ON
TX
RX
WWW.ARDUINO.CC
MADE IN ITALY

introduzione ad Arduino

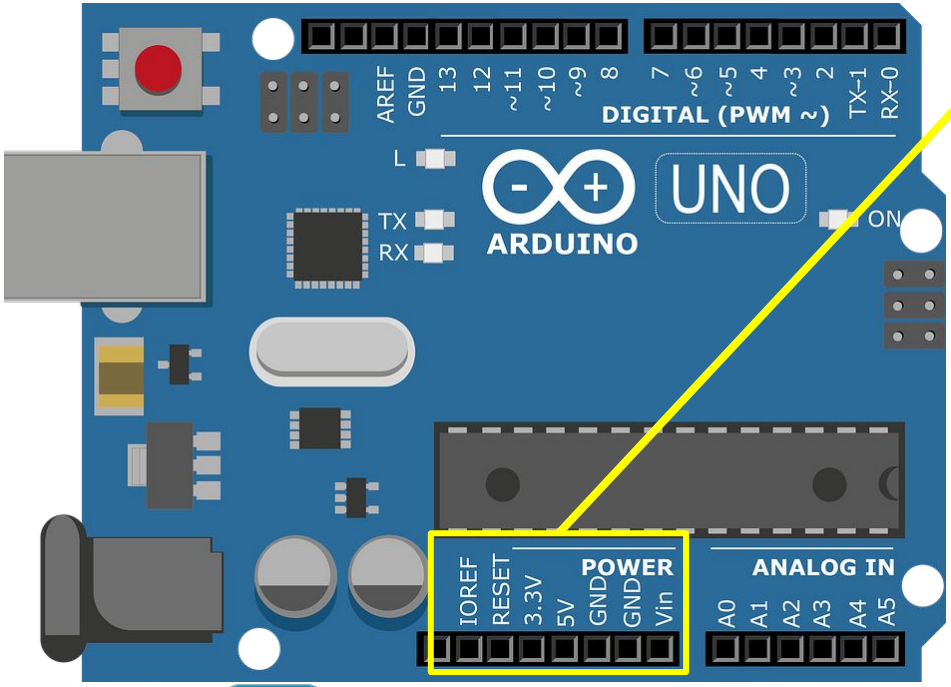
PIN (forellini)
il numero è
importante, perché
serve nella fase di
programmazione

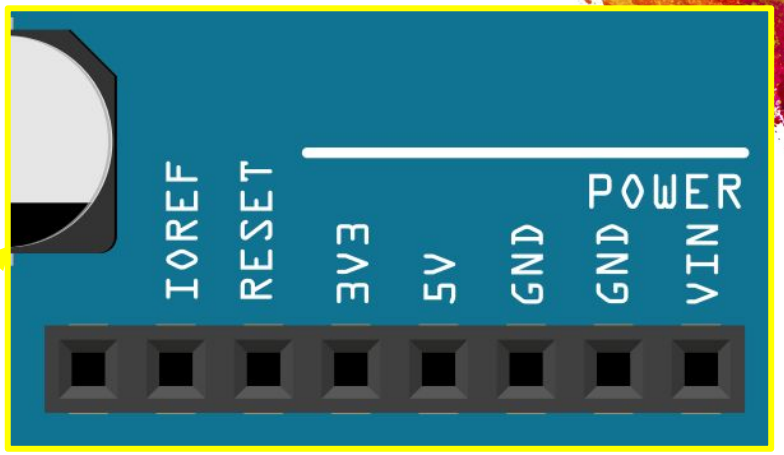


alimentazione
e collegamento al
computer, tramite
cavo USB

Vari PIN possono
lavorare sia in
INPUT (ingresso)
che in OUTPUT
(uscita).

i PIN di Arduino

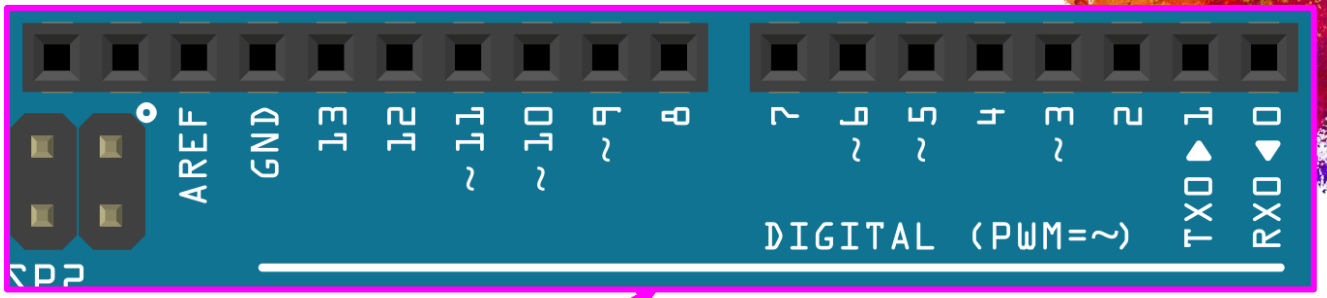




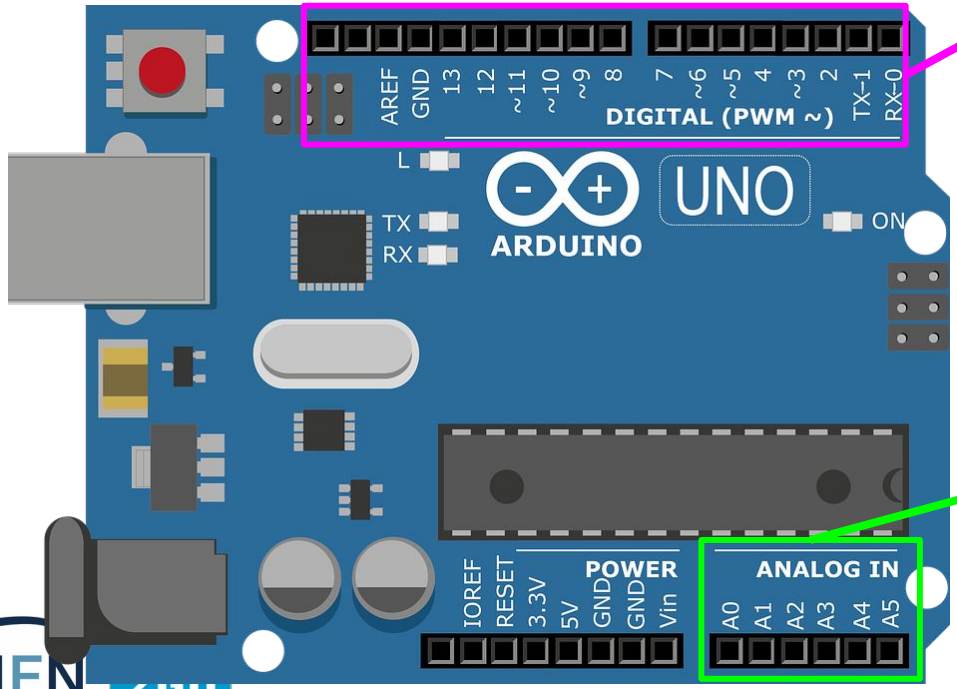
GND = Ground (terra)
3V = alimentazione a 3 volt per altri device
5V = alimentazione a 5 volt per altri device

Vin = "Voltage Input" pin con cui possiamo alimentare Arduino (7-12V), ad esempio collegando una batteria da 9 Volt.

i PIN di Arduino



PIN digitali

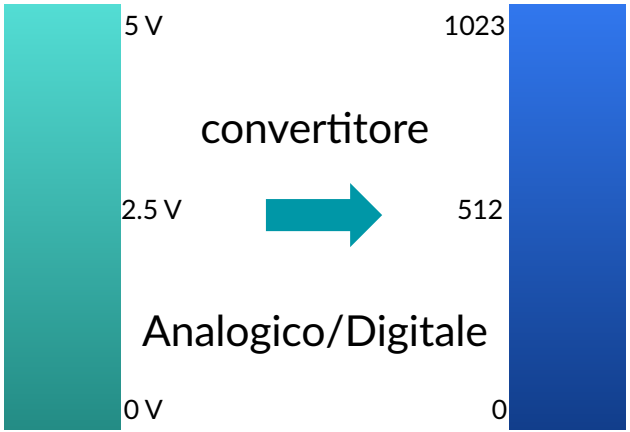


PIN analogici

Acquisizione segnali analogici con Arduino

Microcontrollore:
ATmega328P

- 8 canali → 8 pin
- risoluzione → 10 bit
- 2^{10} → 1024 stati

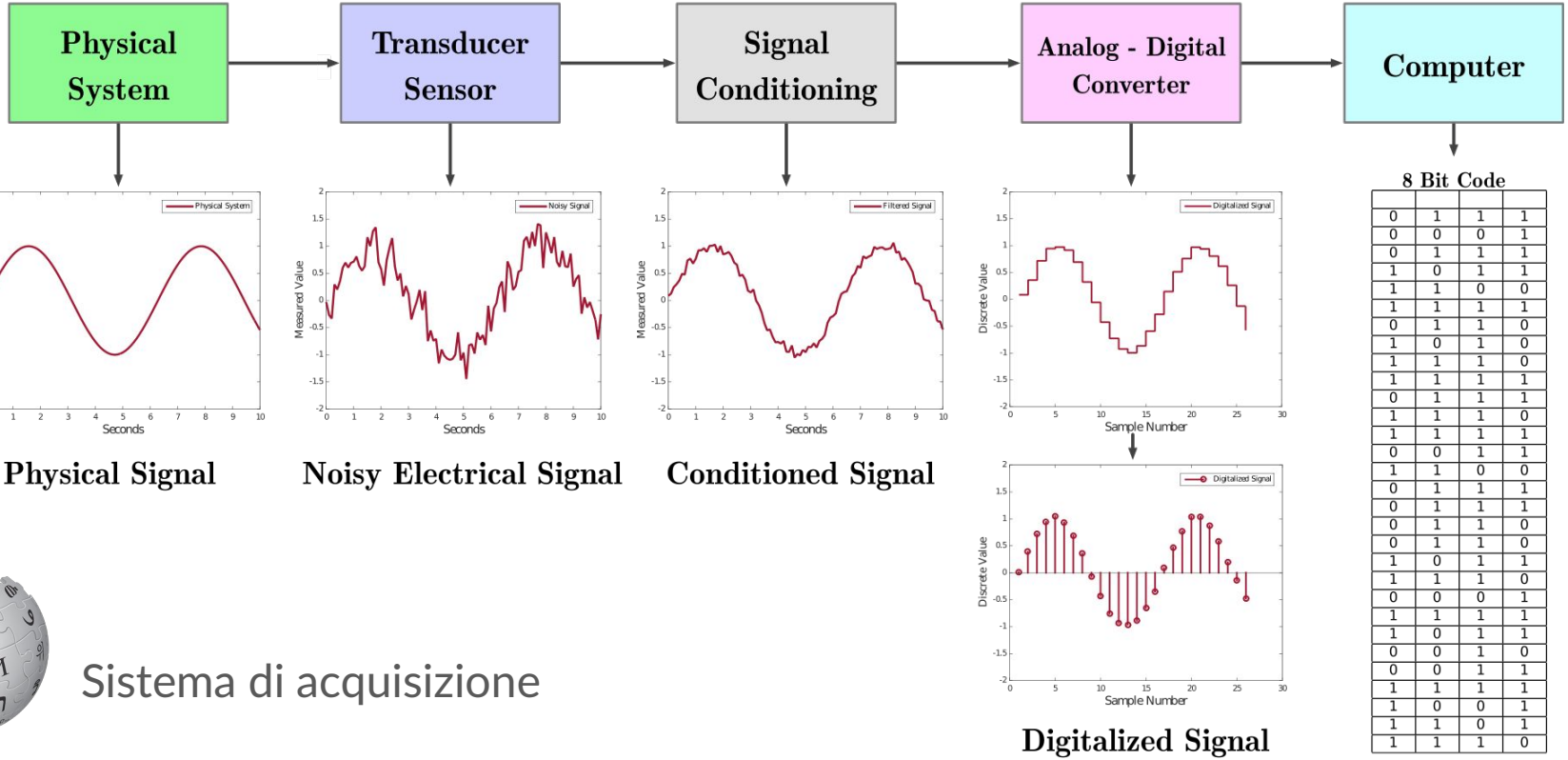


Il microcontrollore ATmega328P usato nell'Arduino UNO mette a disposizione 8 canali per la conversione analogico-digitale (ADC).

Il convertitore ha una risoluzione di 10 bit, restituendo numeri interi da 0 a 1023.

Mentre la funzione principale dei pin analogici è quella di leggere i sensori analogici, essi hanno anche tutte le funzionalità dei pin di input/output per uso generico (lo stesso dei pin digitali 0 - 13).

Digital Data Acquisition System

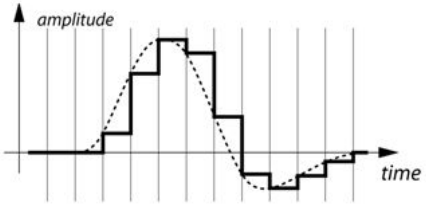


0	1	1	1
0	0	0	1
0	1	1	1
1	0	1	1
1	1	0	0
1	1	1	1
0	1	1	0
1	0	1	0
1	1	1	0
1	1	1	1
0	1	1	1
1	1	1	0
1	1	1	1
0	0	1	1
1	1	0	0
0	1	1	1
0	1	1	1
0	1	1	0
0	1	1	0
1	0	1	1
1	1	1	0
0	0	0	1
1	1	1	1
1	0	1	1
0	0	1	0
0	0	1	1
1	1	1	1
1	0	0	1
1	1	0	1
1	1	1	0

Sistema di acquisizione

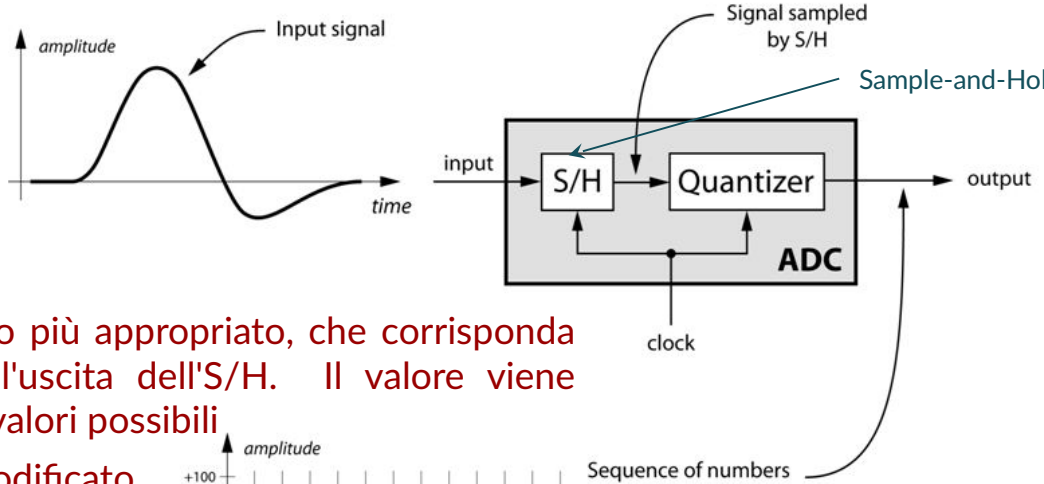
Sampling/Campionamento

Si registra la tensione istantanea di ingresso dell'ADC (sul fronte di salita del segnale di clock) che rimane invariata per l'intero intervallo di campionamento.



Stage 1

Il livello di uscita del S/H viene aggiornato ad ogni fronte di salita dell'ingresso di clock dell'ADC.



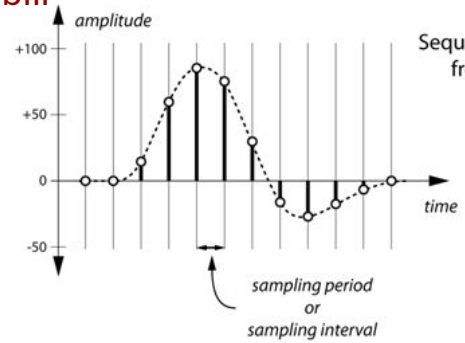
Quantizzazione

Assegna il valore numerico più appropriato, che corrisponda alla tensione presente all'uscita dell'S/H. Il valore viene scelto in un range fisso di valori possibili

Stage 2

Il valore assegnato viene codificato come numero binario.

Per questo motivo, gli ADC vengono spesso definiti ADC N-bit, dove N rappresenta il numero di bit utilizzati dall'ADC per codificare i suoi valori digitalizzati.



⇨ *{…, 0, 0, +15, +60, +85, +72, +28, -17, -30, -21, -8, -1, …}*

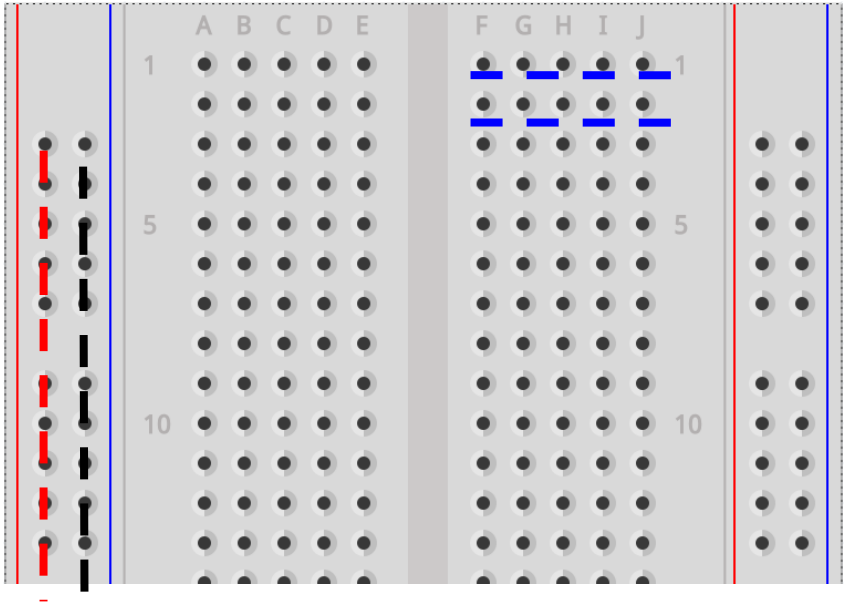
Per convenzione, N-bit è anche usato per indicare la risoluzione dell'ADC, poiché la fase di quantizzazione (la distanza tra i livelli di quantizzazione discreti) è uguale a 1/2N.

Componenti elettronici di base

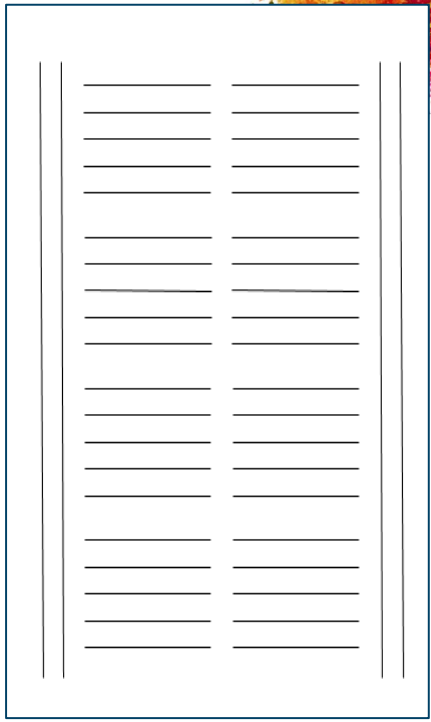
La breadboard

La breadboard facilita i collegamenti: niente saldature o nastro isolante.

I fori che vediamo sono già collegati all'interno della breadboard.

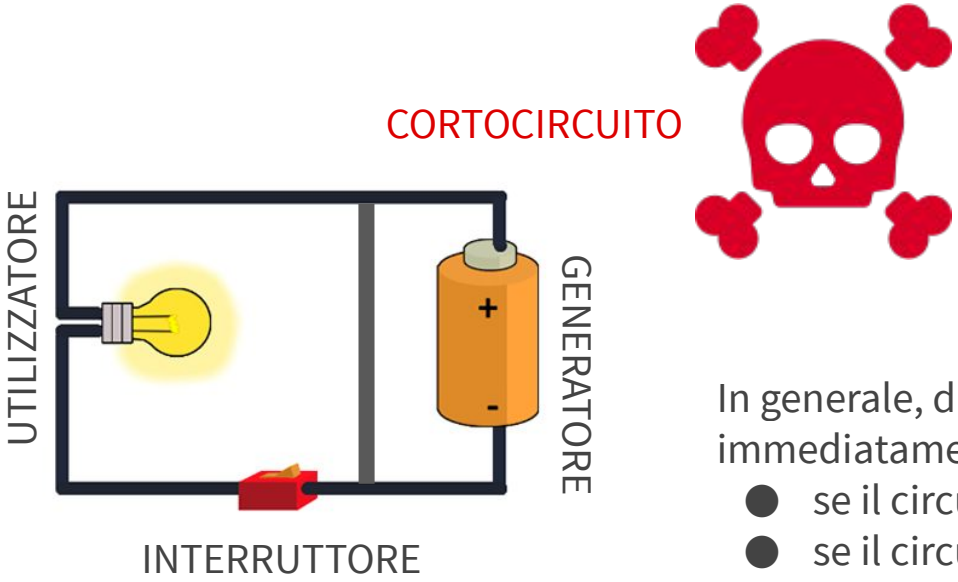


schema →



Le colonne ai lati esterni sono collegate verticalmente. Si usano per l'alimentazione (**+** e **GND**).
Le righe all'interno sono collegate orizzontalmente.Si usano per i collegamenti di cavi e delle componenti elettroniche.

CORTOCIRCUITO



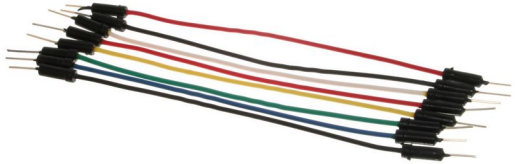
In generale, durante l'utilizzo, staccare immediatamente l'alimentazione:

- se il circuito è caldo
- se il circuito puzza

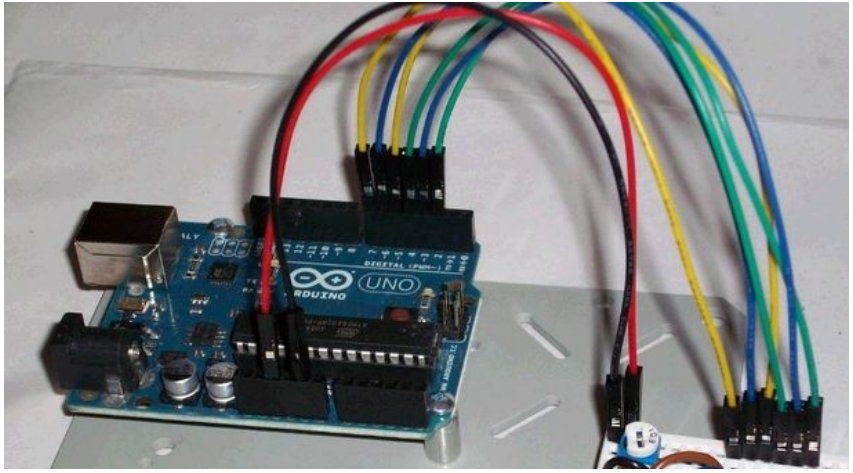
Cavetti

I nomi di questi cavetti sono i più svariati:

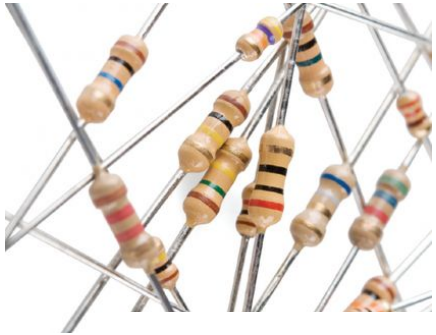
- cavi Dupont
- Jumper
- ponticelli



Le estremità sono progettate per essere infilate nei PIN della scheda o nella breadboard

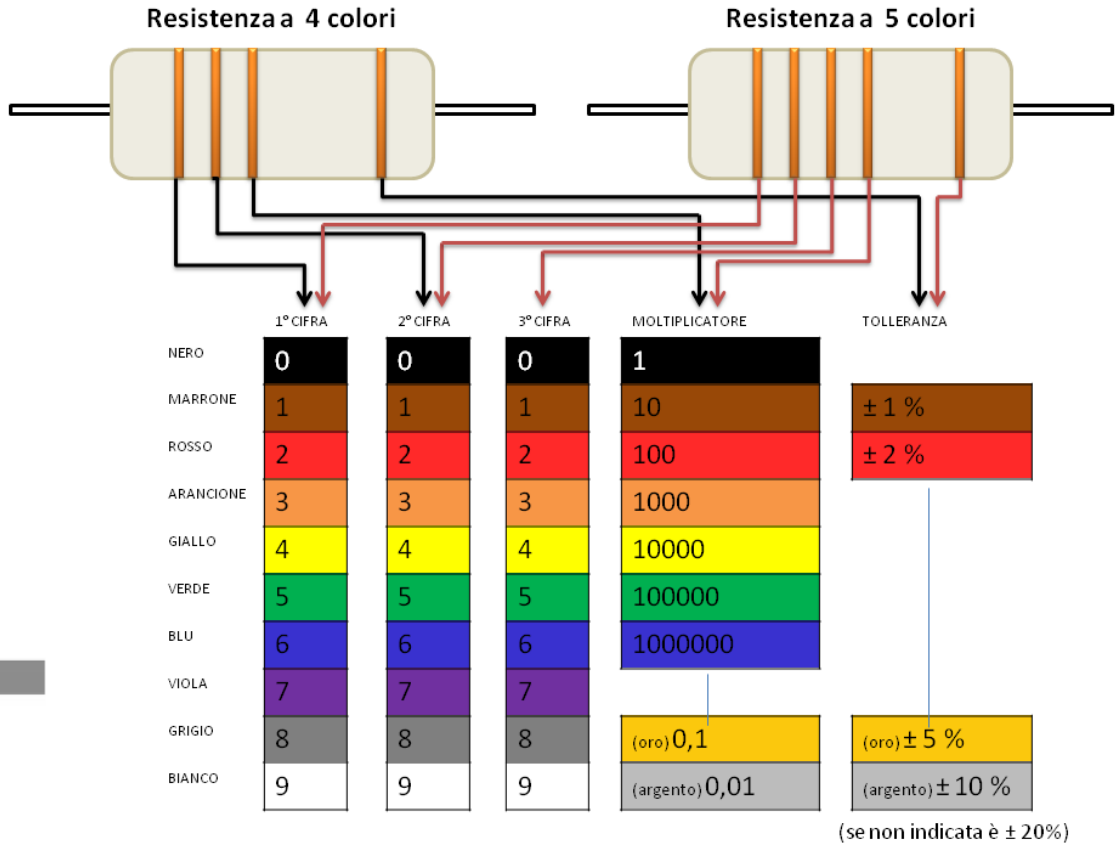


Resistenze

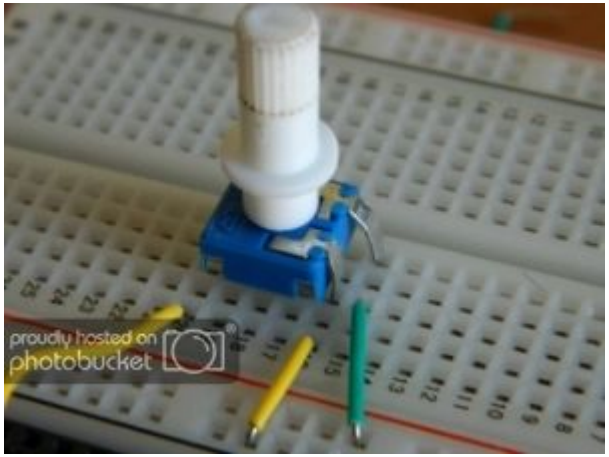


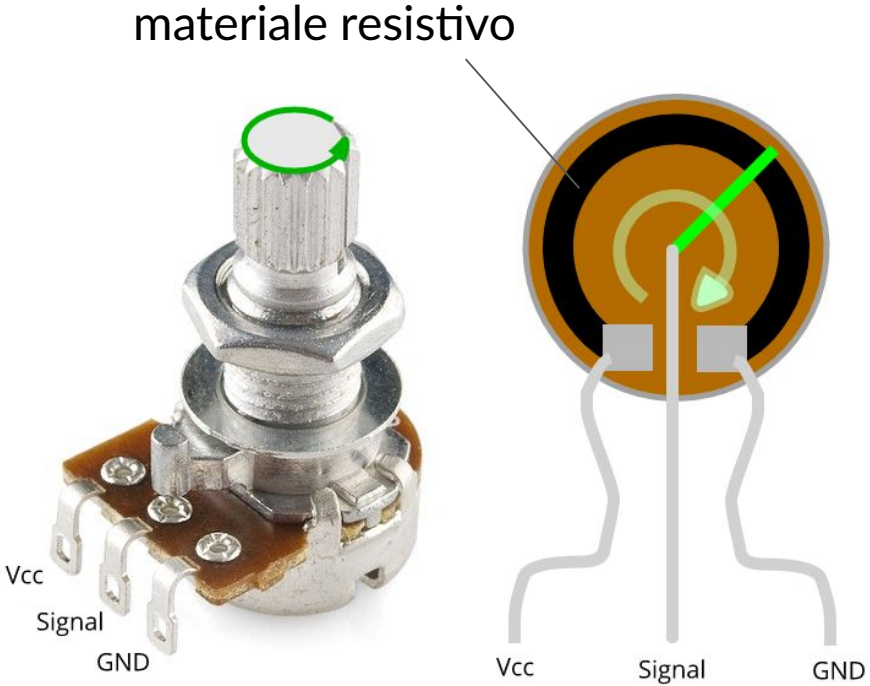
quanto vale questa?

220 ± 11 Ω

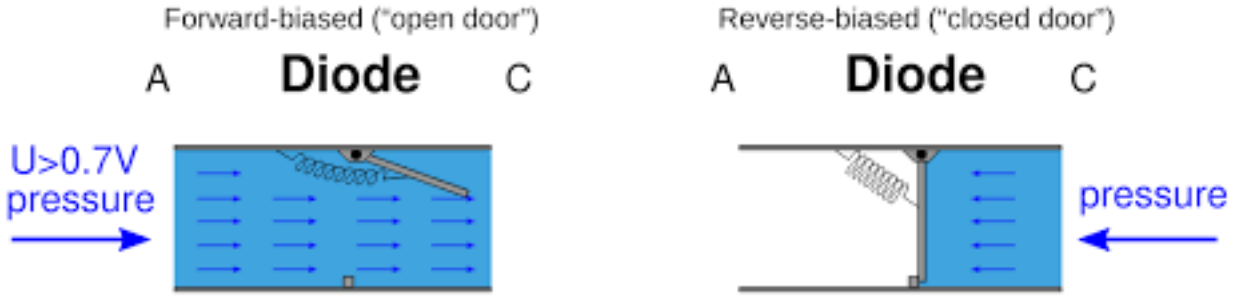


Potenziometro





Il potenziometro è un dispositivo elettrico equivalente ad un partitore di tensione resistivo variabile.
Il potenziometro è costituito da un cilindro isolante su cui è fittamente avvolto un filo metallico con resistività opportuna, le due estremità sono connesse a due morsetti. Longitudinalmente al cilindro e da un'estremità all'altra, scorre un cursore recante un contatto strisciante sul filo, a sua volta collegato ad un morsetto.



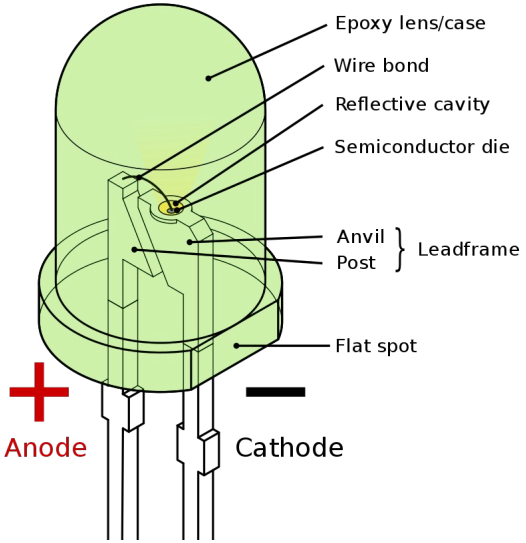
Un diodo è un dispositivo che consente il passaggio di corrente in un'unica direzione. Questi componenti sono spesso usati per isolare l'effetto di un componente da un altro.

Alcuni tipi comuni di diodi:

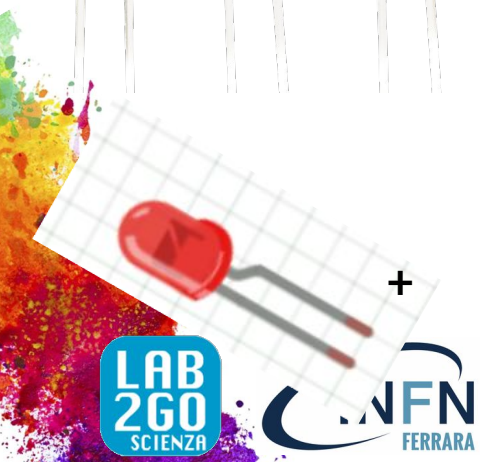
- Diodo ad emissione luminosa (LED)
- Diodo fotografico: rileva la luce
- Diodo laser - emette un raggio di luce coerente (laser)
- Diodo Zener: impedisce il flusso di corrente fino a una soglia di tensione.

LED - Light Emitting Diode

Un LED è un diodo che genera luce monocromatica quando gli viene applicata sufficiente tensione. La luminosità varia al variare della corrente. Ha quindi un polo + ed un polo-.

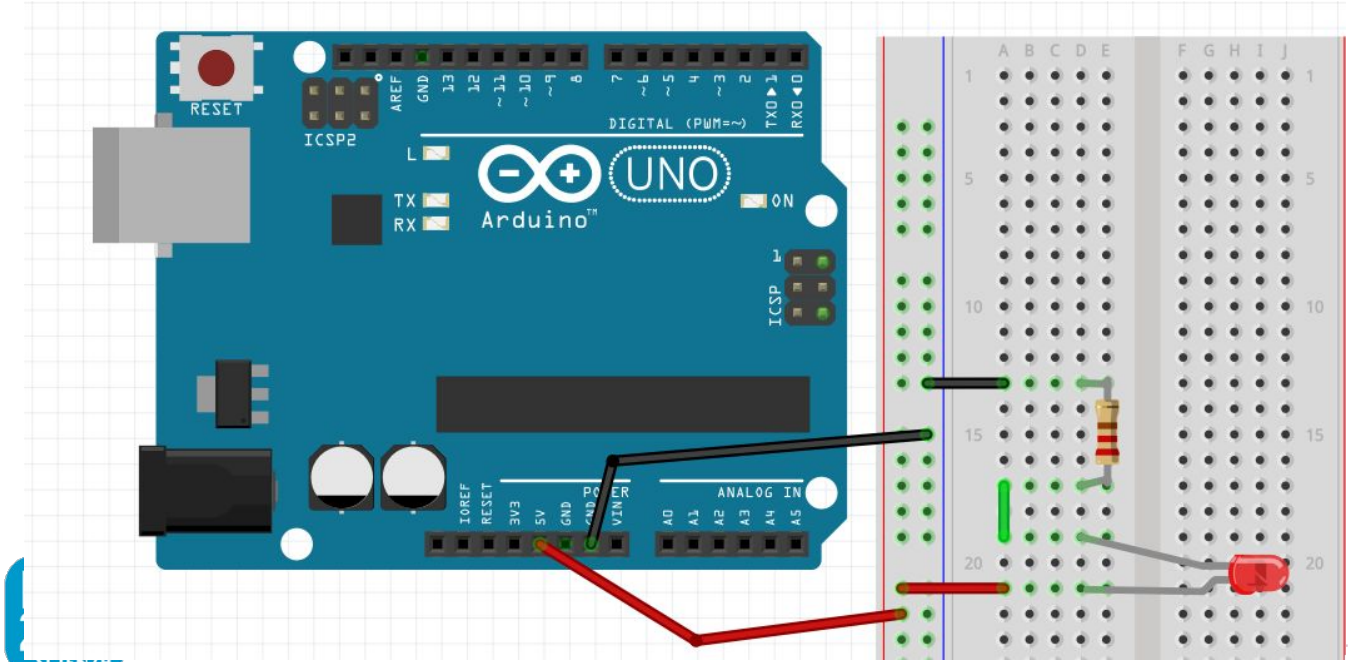


Un LED non ha alcuna limitazione di corrente intrinseca, pertanto l'applicazione di V superiori alla tensione diretta può provocare la bruciatura del LED.
E' sempre consigliabile utilizzare resistenze in serie ai LED.



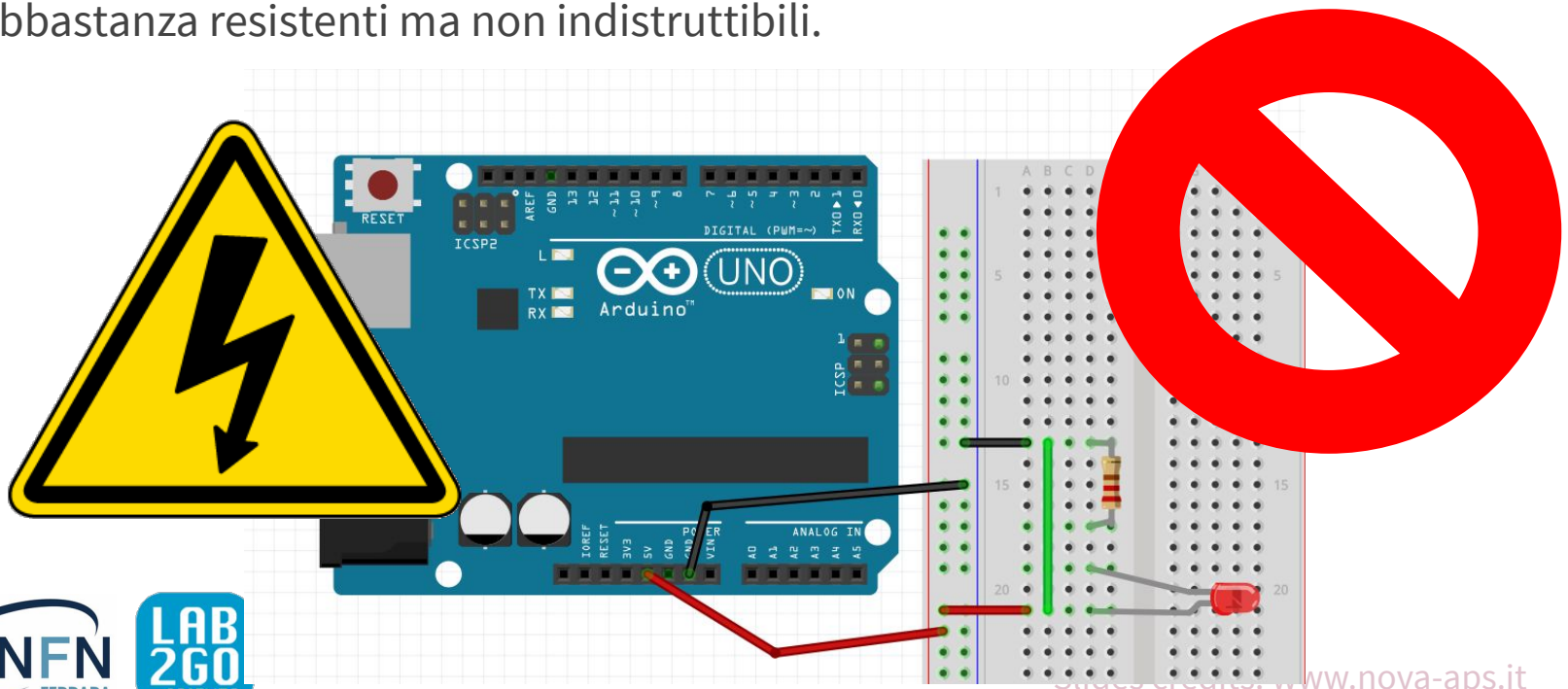
I circuiti...

Questo è un esempio di circuito SEMPLICE, senza programmazione.
Collegando ARDUINO al PC dovrebbe accendersi il LED
NOTA BENE: C'è un utilizzatore tra GND e 5v

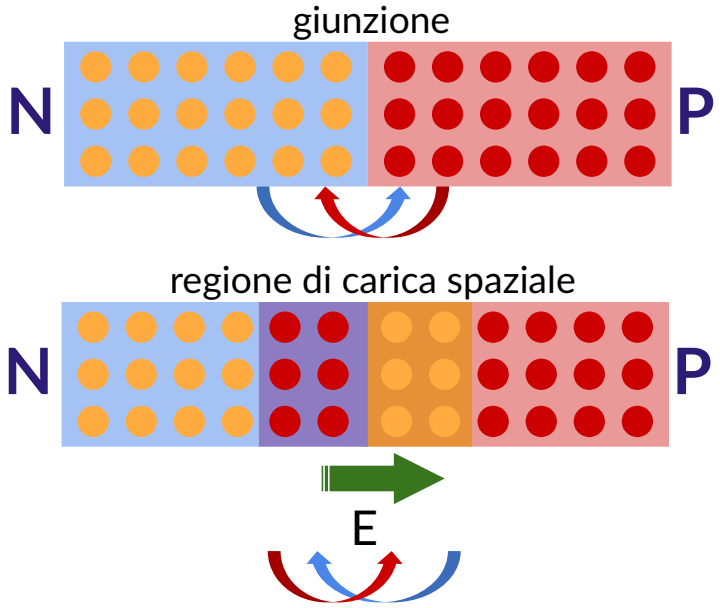


DA NON FARE

Questo è un **cortocircuito**.
E’ molto importante stare attenti ed evitare di fare dei cortocircuiti. Le schedine sono abbastanza resistenti ma non indistruttibili.



Giunzione p-n



il processo continua fino al raggiungimento dell'equilibrio

corrente di diffusione

corrente di trascinamento

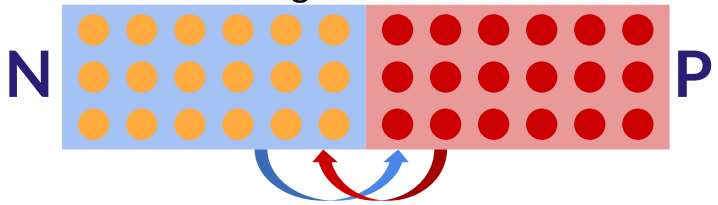
La regione di confine tra i blocchi di tipo P e di tipo N è detta zona/regione di carica spaziale (o di svuotamento).

In questo volume i portatori, rispettivamente del lato p e del lato n, di fronte al forte gradiente dovuto al diverso tipo di drogaggio, diffondono nel semiconduttore adiacente (generando una corrente di diffusione), lasciando non compensati gli atomi ionizzati dei droganti, i quali a loro volta genereranno una differenza di potenziale, un **campo elettrico** e, spostando i portatori, una corrente di trascinamento che si oppone a quella di diffusione.

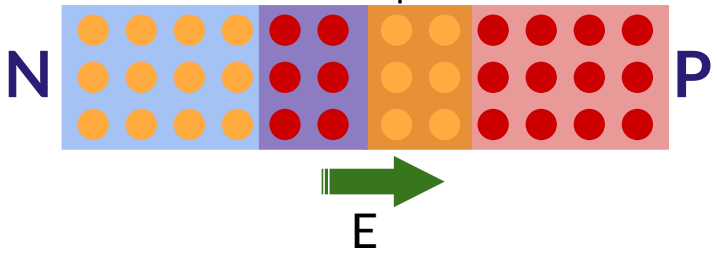
La differenza di potenziale costante generata dagli ioni di materiale drogante è chiamata **tensione di built-in**. La larghezza della zona di carica spaziale dipende dai drogaggi e da ciascun lato è inversamente proporzionale al drogaggio del semiconduttore.

Giunzione p-n

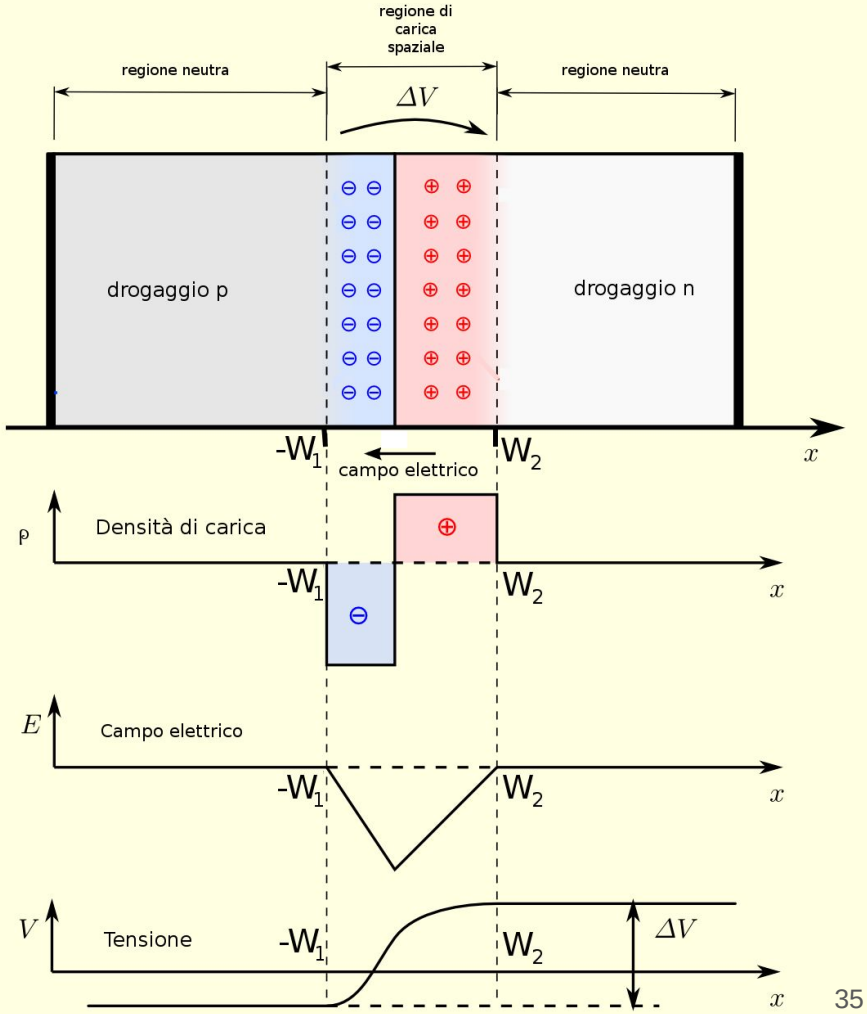
giunzione



zona di deplezione

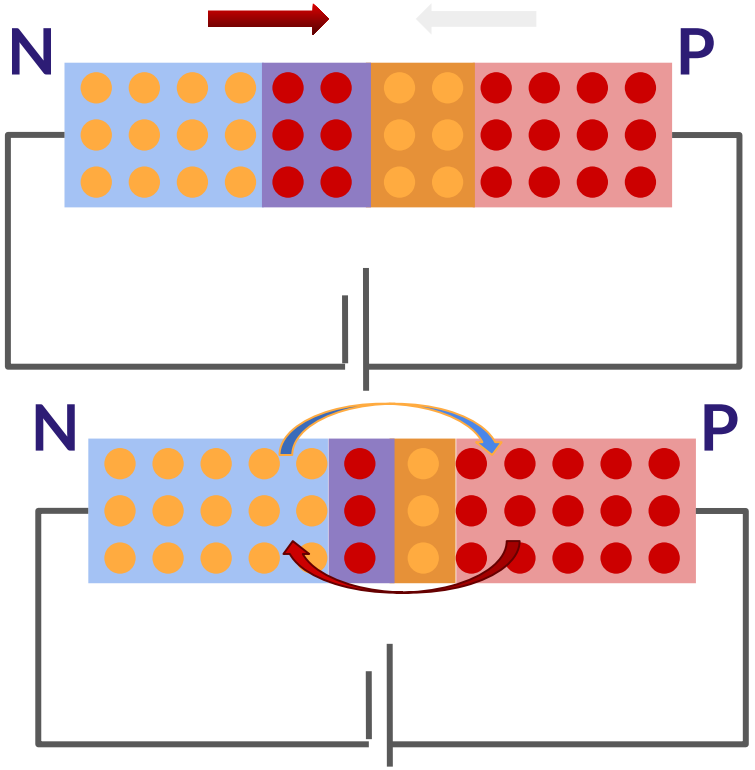


il processo continua fino al raggiungimento dell'equilibrio



Forward Bias

1. (+) repelle le lacune e (-) gli elettroni
2. la zona di deplezione si restringe
3. Se la V è al di sopra di un determinato valore, gli elettroni nella regione N migrano nella regione P (e le lacune nella regione P migrano nella regione N)
4. La corrente attraversa il circuito. Prende il nome di drift current

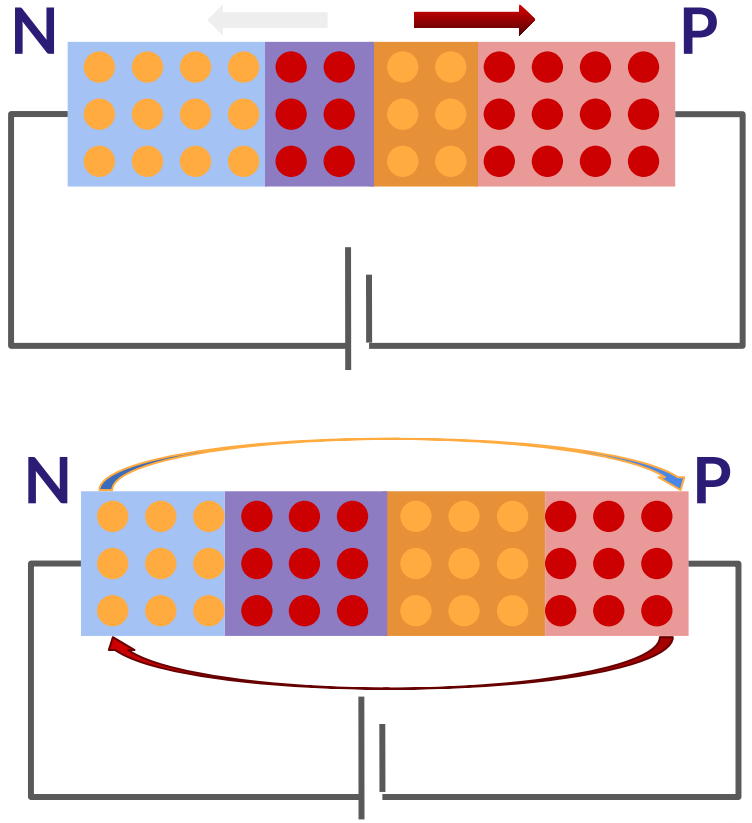


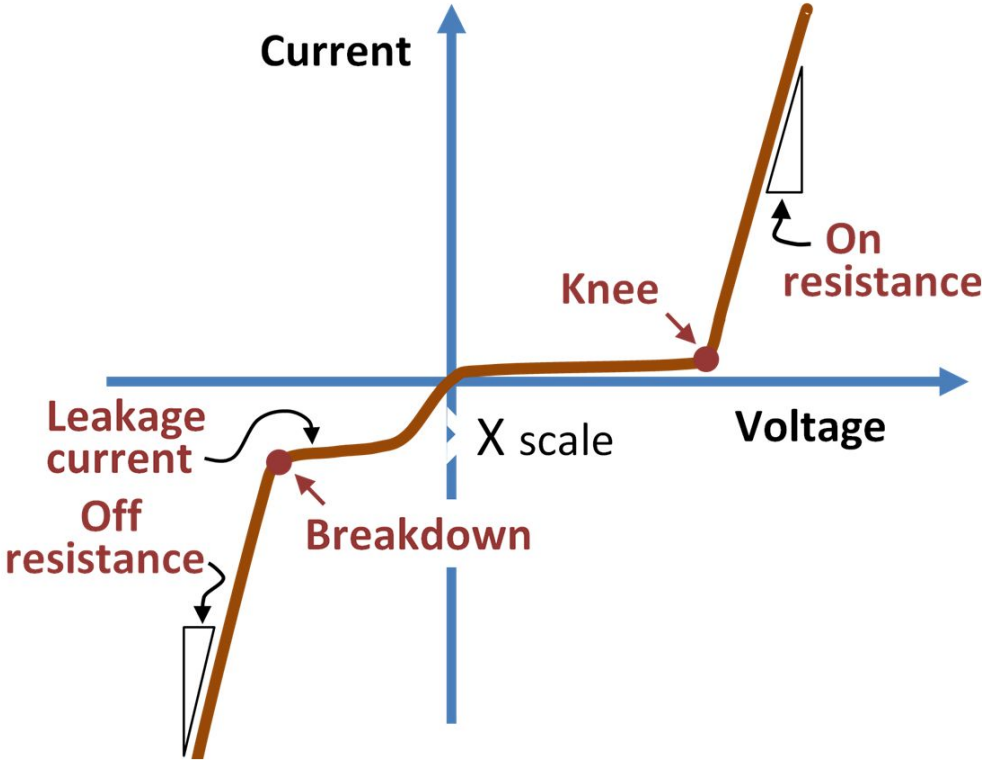
Reverse Bias

1. (+) attraegli elettroni e (-) le lacune
2. la zona di deplezione si allarga
3. La giunzione PN agisce come un isolante.

Se V cresce al di sopra di un particolare limite (reverse bias breakdown voltage) la corrente riprende a fluire.

Avalanche Breakdown: V è molto alta → la giunzione subisce danni permanenti a causa del surriscaldamento (la corrente è più alta di quanto possa sostenere).




Current
Knee
On resistance
Leakage current
X scale
Voltage
Off resistance
Breakdown

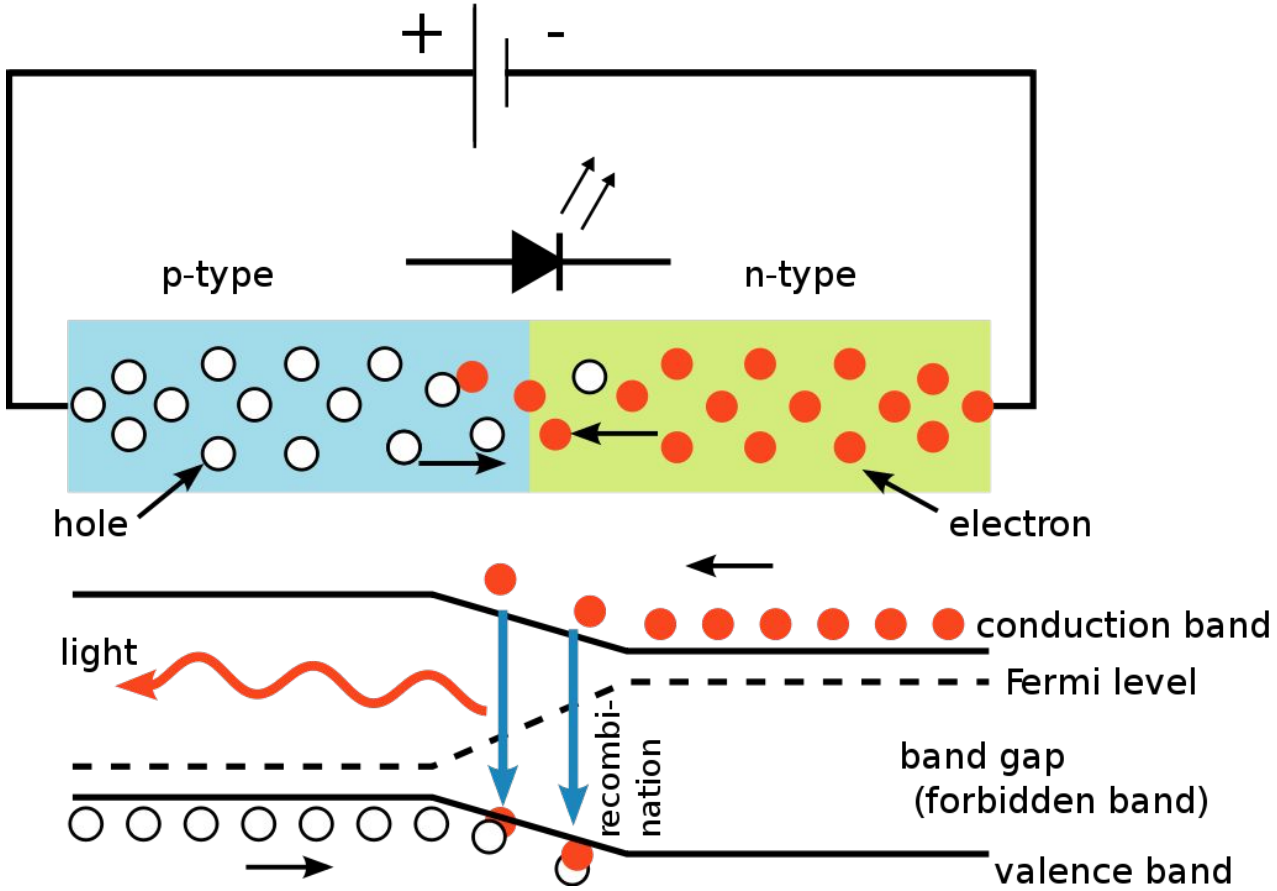
Elettroluminescenza

La ricombinazione genera:

- radiazione infrarossa per diodi al silicio e germanio,
- radiazione visibile per semiconduttori di arseniuro di gallio (GaAsP) e fosfuro di gallio (GaP),

Se il semiconduttore è traslucido, la giunzione diventa la fonte di luce, diventando così un diodo ad emissione luminosa.

La lunghezza d'onda della luce emessa, e quindi il suo colore, dipende del gap di banda di energia dei materiali che formano la giunzione p-n.

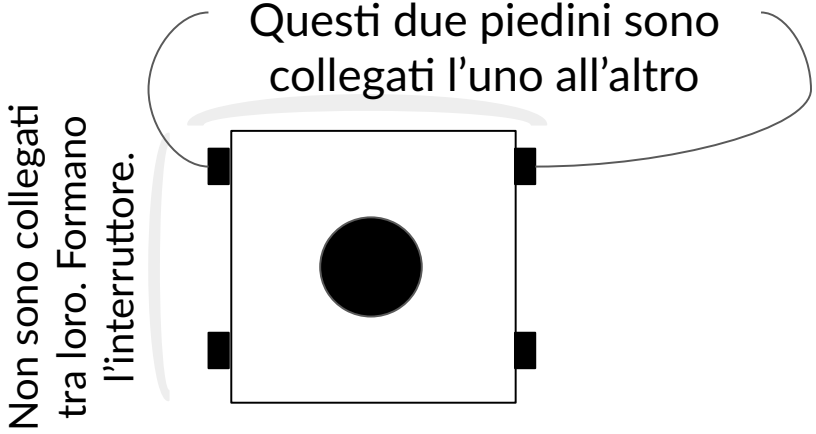


RGB LED

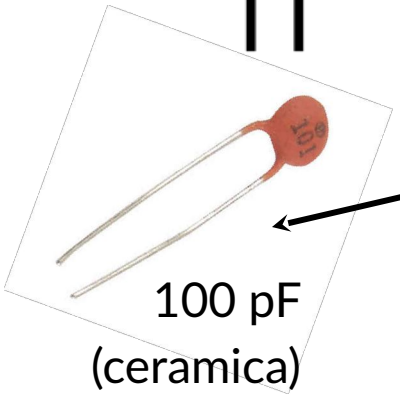
Poiché i LED hanno schemi di emissione leggermente diversi, il bilanciamento del colore può variare in base all'angolo di visualizzazione, anche se le sorgenti RGB si trovano in un unico emettitore, pertanto i diodi RGB vengono utilizzati raramente per produrre l'illuminazione bianca.

Provate ad aggiungere un bottone

pulsante/bottone interruttore



Condensatori

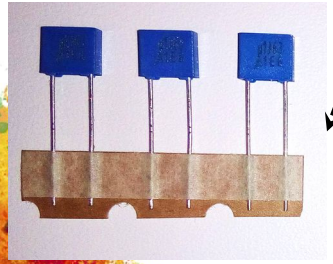


100 pF
(ceramica)

a dielettrico solido:
(non hanno una polarità)

100 µF

Elettrolitici:
hanno una polarità → esplodono se collegati al contrario



0.1 µF
(poliestere)

Nei condensatori elettrolitici non è presente un materiale dielettrico, ma l'isolamento è dovuto alla formazione e mantenimento di un sottilissimo strato di ossido metallico sulla superficie di una armatura a contatto con una soluzione chimica umida. Vista la esiguità fisica del dielettrico, non possono sopportare tensioni molto alte.

A differenza dei condensatori comuni, la sottigliezza dello strato di ossido consente di ottenere, a parità di dimensioni, capacità molto più elevate. Per contro, occorre adottare particolari accorgimenti per conservare l'ossido stesso.

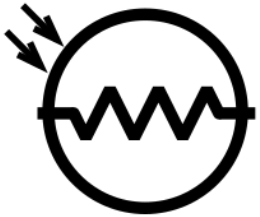
Sensori e Attuatori

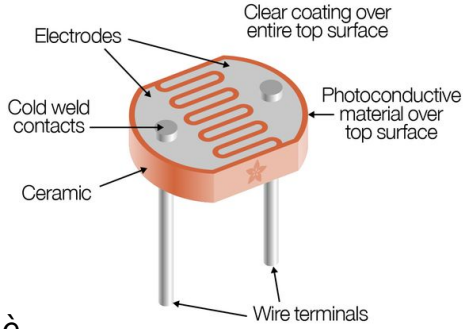
Sensori e Attuatori

I **sensori** sono componenti che misurano una forma di energia (come la luce, il calore o l'energia meccanica) e la converte in energia elettrica. Utilizziamo i **sensori** per rilevare un **INPUT.**

Gli **attuatori** servono per produrre un risultato pratico. Utilizziamo gli **attuatori** per produrre un risultato, quindi per produrre un **OUTPUT**.
Ogni sensore/attuatore può essere analogico o digitale.

Fotoresistenza





La fotoresistenza (o fotoresistore) è un componente elettronico la cui **resistenza** è **inversamente proporzionale** all'intensità della luce che lo colpisce.

Questo componente è utilizzato anche per la realizzazione di crepuscolari (circuiti che permettono di accendere una o più luci al calare del sole).

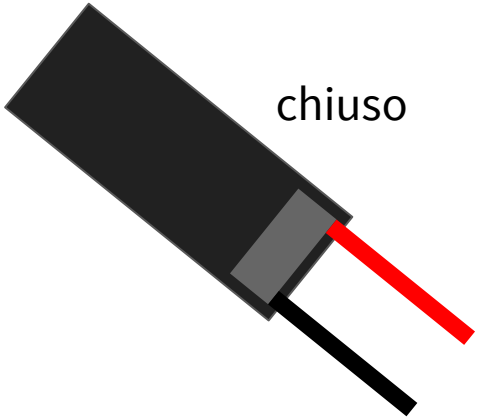
E' composta da materiale semiconduttore. L'energia radiante fornita a un semiconduttore provoca la produzione di coppie elettrone-lacuna in eccesso rispetto a quelle generate termicamente, che causa una diminuzione della resistenza elettrica del materiale (effetto fotoconduttivo). Quando la radiazione incidente viene interrotta i portatori di carica in eccesso si ricombinano riportando la conducibilità del semiconduttore al suo valore iniziale in condizioni di oscurità.

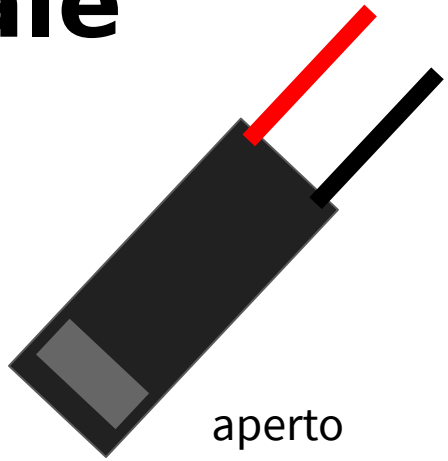
analogico

Sensore di inclinazione (TILT)

Il **sensore di inclinazione** è in realtà un interruttore: chiuso per una inclinazione, aperto per un'altra.

digitale





attenzione a non confonderlo con i transistor

Sensore di temperatura

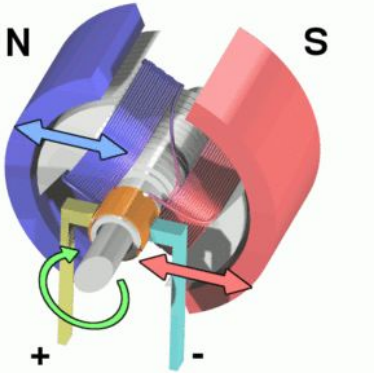
datasheet

Termistori e termoresistenze sono sensori di temperatura che sfruttano la variazione della resistività di alcuni materiali al variare della temperatura.

- Le termoresistenze sono composte da materiali conduttori metallici (per esempio platino)
- I termistori sono composti da materiali semiconduttori e si differenziano per drogaggio e comportamento:
  - NTC, la resistenza diminuisce all'aumentare della temperatura. Comunemente usato come sensore di temperatura o in serie con un circuito come limitatore di corrente di spunto.
  - PTC, la resistenza aumenta all'aumentare della temperatura. Comunemente installati in serie con un circuito per proteggere dalle condizioni di sovracorrente, come i fusibili ripristinabili.

analogico

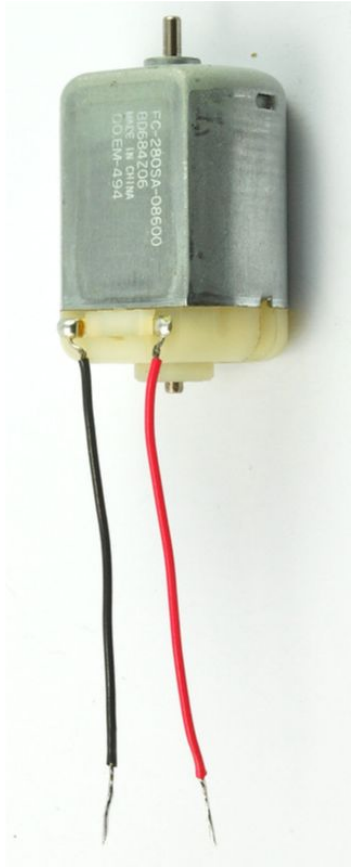
Motore CC



Quando la corrente scorre negli avvolgimenti, si genera un campo magnetico intorno al rotore.

Una parte del rotore è respinta dal magnete ed attratta dall'altra. La coppia genera la rotazione. Quando le armature si allineano orizzontalmente, il commutatore inverte la direzione di corrente attraverso gli avvolgimenti, modificando anche il campo magnetico.

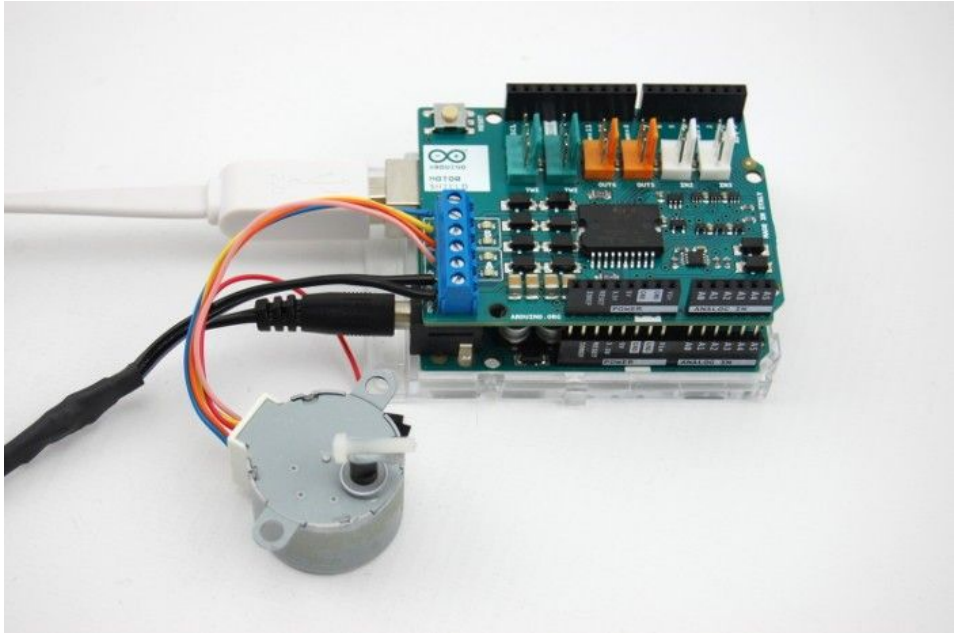
Il processo ritorna quindi allo stato di partenza e il ciclo si ripete.



Motore passo-passo

Il motore passo-passo spesso chiamato anche step o stepper è un motore elettrico sincrono in corrente continua pulsata con gestione elettronica senza spazzole (brushless) che può suddividere la propria rotazione in un grande numero di passi (step). La posizione del motore può essere controllata accuratamente se la taglia ed il tipo di motore sono scelti in modo adeguato all'applicazione.

È considerato la scelta ideale per tutte quelle applicazioni che richiedono precisione nello spostamento angolare e nella velocità di rotazione, quali la robotica, le montature dei telescopi ed i servomeccanismi in generale.



Servomotore

Un servomotore è un meccanismo a circuito chiuso che utilizza il feedback di posizione per controllarne il movimento e la posizione finale. Il segnale di controllo (analogico o digitale) rappresenta la posizione comandata per l'albero di uscita.

Il motore è accoppiato con un tipo di encoder per fornire feedback di posizione e velocità. Nel caso più semplice, viene misurata solo la posizione. La posizione misurata dell'uscita viene confrontata con la posizione di comando, l'ingresso esterno sul controller.

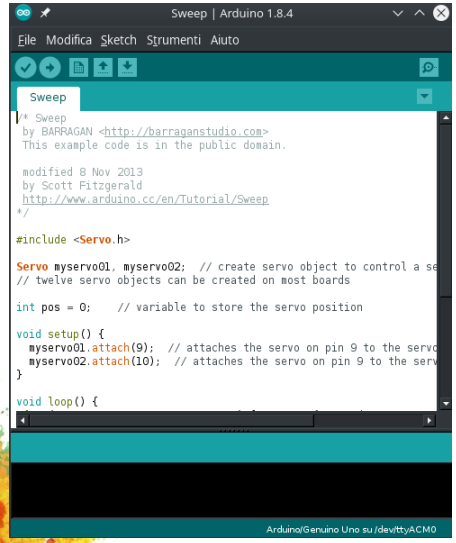
Servomotore

Se la posizione di uscita è diversa da quella richiesta, viene generato un segnale di errore che provoca la rotazione del motore in entrambe le direzioni, in base alle necessità per portare l'albero di uscita nella posizione appropriata.

All'avvicinarsi delle posizioni, il segnale di errore si riduce a zero e il motore si arresta.

La IDE di Arduino

IDE



Un **ambiente di sviluppo integrato** (Integrated Development Environment è un software che, in fase di programmazione, aiuta i programmatori nello sviluppo del codice sorgente di un programma.

Spesso l'IDE aiuta lo sviluppatore segnalando errori di sintassi del codice direttamente in fase di scrittura, oltre a tutta una serie di strumenti e funzionalità di supporto alla fase di sviluppo e debugging.

Editor + compilatore + debugger + linker + … = IDE

Sketch

Ogni sketch è composto da 2 parti diverse:

- **Setup**: inizializza il programma ed esegue alcune funzioni una volta sola all'accensione
- **Loop**: è la parte del programma che viene continuamente ripetuta dalla prima all'ultima istruzione, dalla prima all'ultima e così via...

```
/*
  Blink - Turns an LED on for one second, then off for one
second, repeatedly.  [...]
  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power
the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

La programmazione

Arduino ha un linguaggio di programmazione rigoroso. Alcune regole non possono essere violate, altrimenti il programma restituisce un errore: keywords, regole di sintassi, differenza tra maiuscole e minuscole, punteggiatura...

Altre sono convenzioni dei programmatori, utili per lo più a migliorare la leggibilità.

I commenti sono parti di testo che non vengono eseguiti, servono solo come promemoria per noi.

```
/*
  Blink - Turns an LED on for one second, then off for one
second, repeatedly.  [...]
  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power
the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

La programmazione

commenti:

- // ignora tutto il testo che trova dopo, fino a fine linea;
- /* e */ indicano inizio e fine di un commento, potendo racchiudere commenti di più linee

parentesi: ogni parentesi aperta deve anche essere chiusa, anche se non contengono nulla, (es. setup())

- le tonde racchiudono gli argomenti che vengono passati ad una funzione
- le graffe racchiudono un blocco di istruzioni

Ogni comando deve terminare con punto e virgola ;

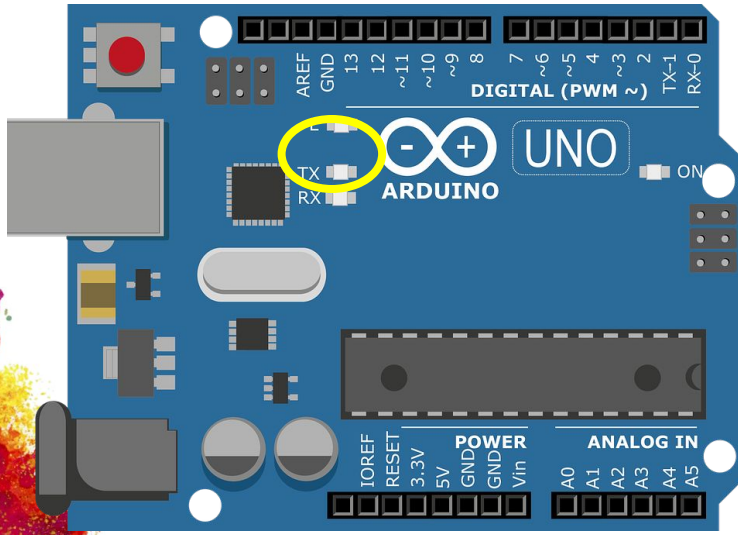
Il linguaggio è case-sensitive (differenzia tra maiuscole e minuscole) e la prassi è camelCase.

```
/*
  Blink - Turns an LED on for one second, then off for one
second, repeatedly.  [...]
  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power
the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

1st try - Blink!

Proviamo a realizzare un semplice programma che fa lampeggiare il LED integrato sulla schedina.

E' sufficiente collegare Arduino al PC con il cavo USB, non è necessaria la breadboard.




```
/*
  Blink - Turns an LED on for one second, then off for one
second, repeatedly.  [...]
  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power
the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

1st try - Blink!

Setup di utilizzo: eseguiamo questo set di comandi una sola volta, all'avvio del programma.

La funzione pinMode imposta la modalità del pin che desideriamo utilizzare. Richiede 2 argomenti: il numero di pin e la modalità.

Ad esempio, se vogliamo utilizzare il pin 13 come output scriveremo: `pinMode(13, OUTPUT);`

La costante LED_BUILTIN è preimpostata al pin che corrisponde al LED integrato nella scheda (che corrisponde proprio al 13).

```
/*
  Blink - Turns an LED on for one second, then off for one second, repeatedly.  [...]
  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

1st try - Blink!

Le istruzioni che scriviamo dentro la

Ad esempio, se vogliamo utilizzare il pin 13 come output scriveremo: `pinMode(13, OUTPUT);`

La costante LED_BUILTIN è preimpostata al pin che corrisponde al LED integrato nella scheda (che corrisponde proprio al 13).

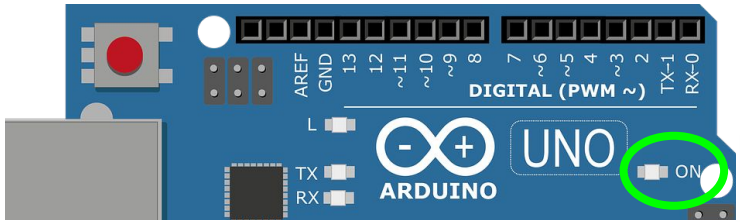
La funzione digitalWrite ci permette di impostare il pin 13 (LED_BUILTIN) allo stato HIGH,ogni pin digitale infatti può assumere solo 2 stati: HIGH o LOW.

La funzione delay ci permette di bloccare l'esecuzione del programma per un numero definito di millisecondi (nell'esempio, 1000).

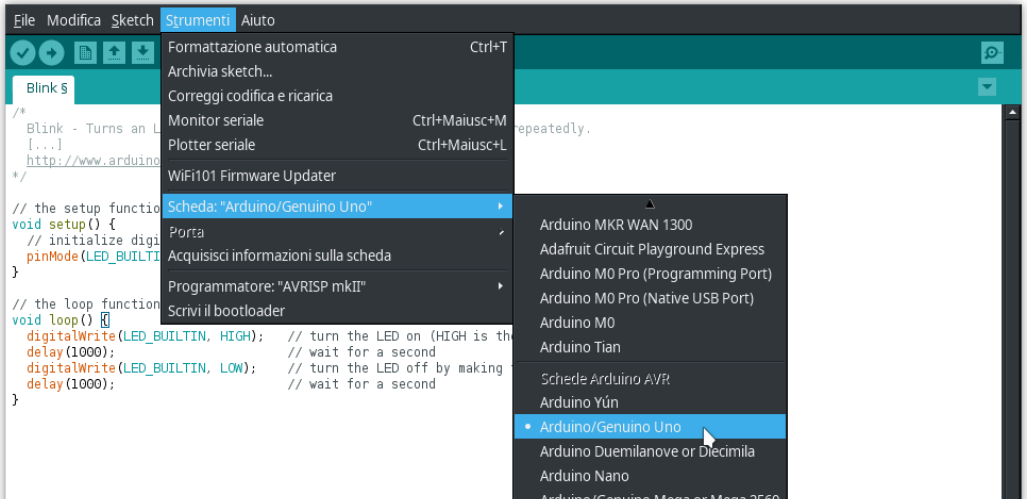
```
/*
  Blink - Turns an LED on for one second, then off for one
second, repeatedly.  [...]
  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power
the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);
// turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);
// turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Trasferiamo il firmware



1. assicurarsi che Arduino sia collegato al PC tramite USB
2. assicurarsi che la scheda sia correttamente riconosciuta, dal menù strumenti > scheda > Arduino/Genuino Uno

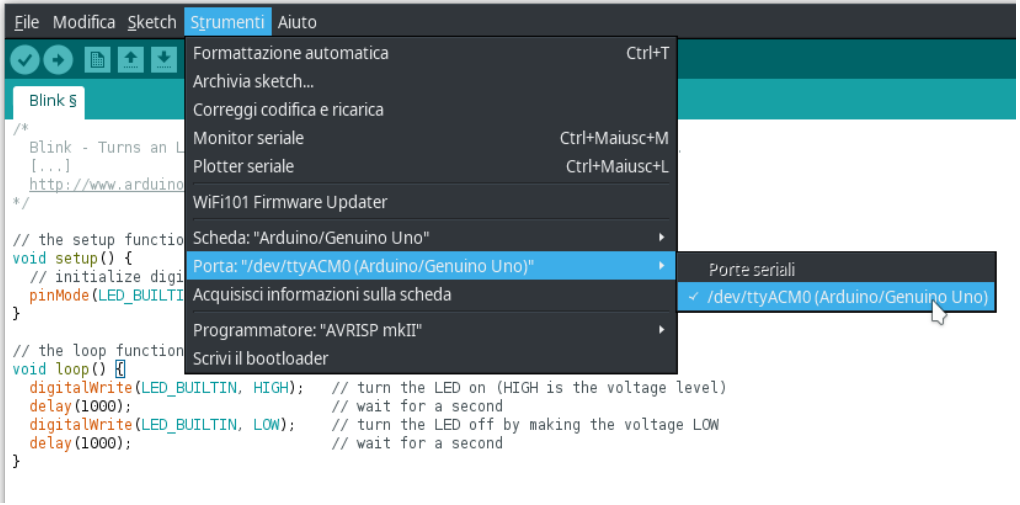


Trasferiamo il firmware

3. Ci assicuriamo che la porta USB sia correttamente impostata, dal menù Strumenti > Porta

   di solito in Windows si chiamano COM1, COM2 …

   in Linux si chiamano ttyACM0, ttyACM1, …

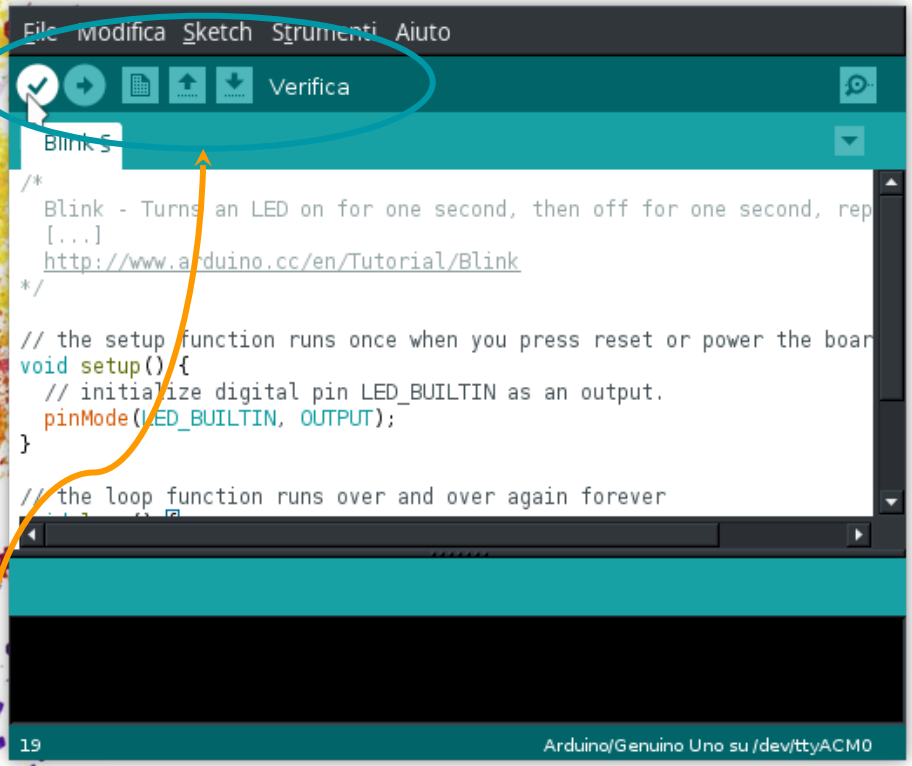


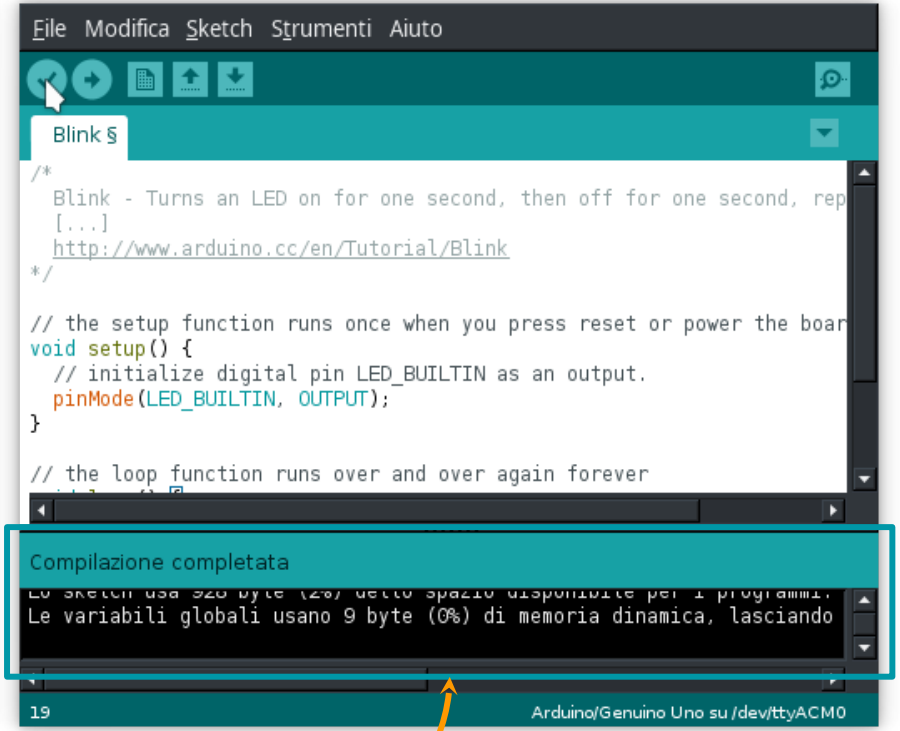
Compilazione e trasferimento

Arduino IDE :

1. compila il codice, cioè controlla che non ci siano errori di sintassi e lo converte in un binario eseguibile
2. se tutto è sintatticamente corretto, trasferisce il programma sulla scheda

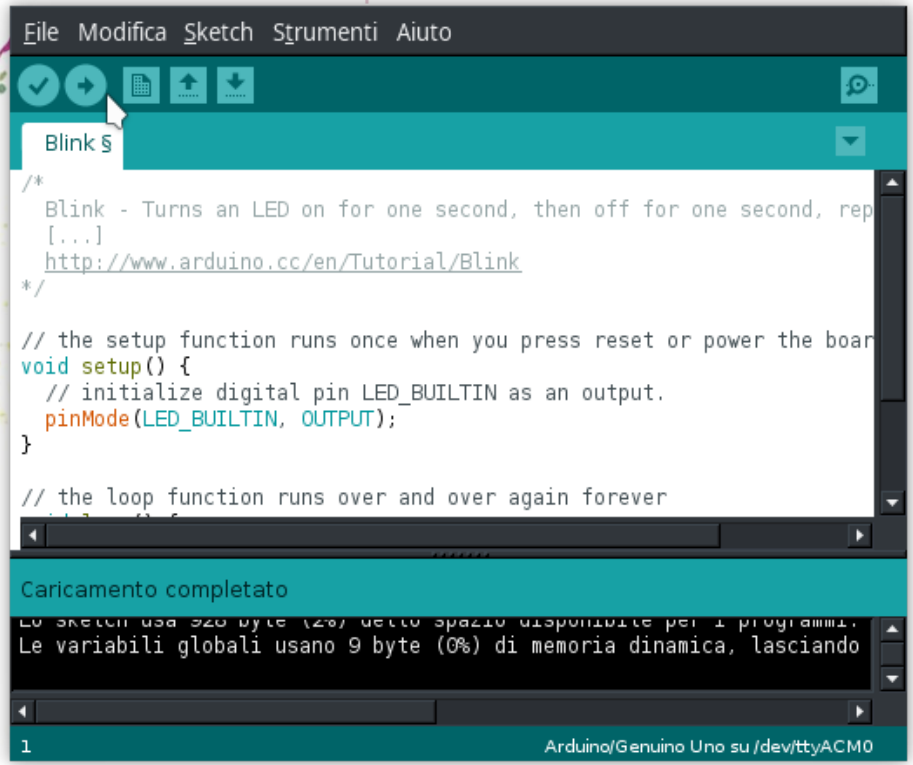
Se ci sono errori, la parte in basso diventa arancione indicando gli errori trovati.

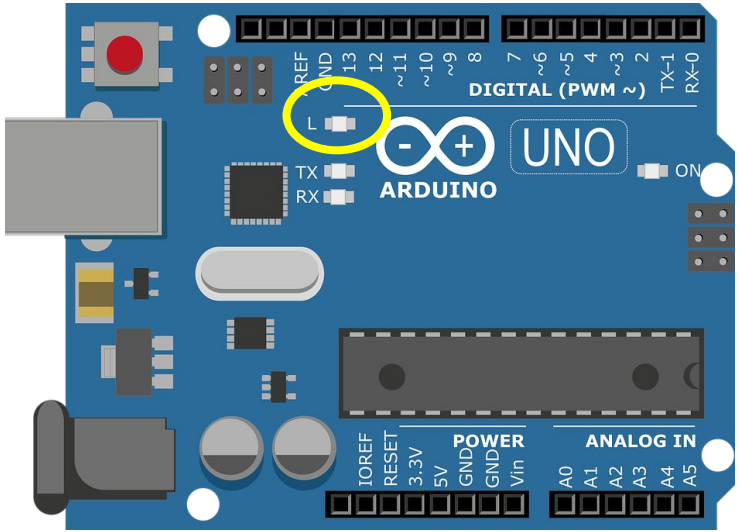




Compilo e controllo che non ci siano errori

Slides credits: www.nova-aps.it

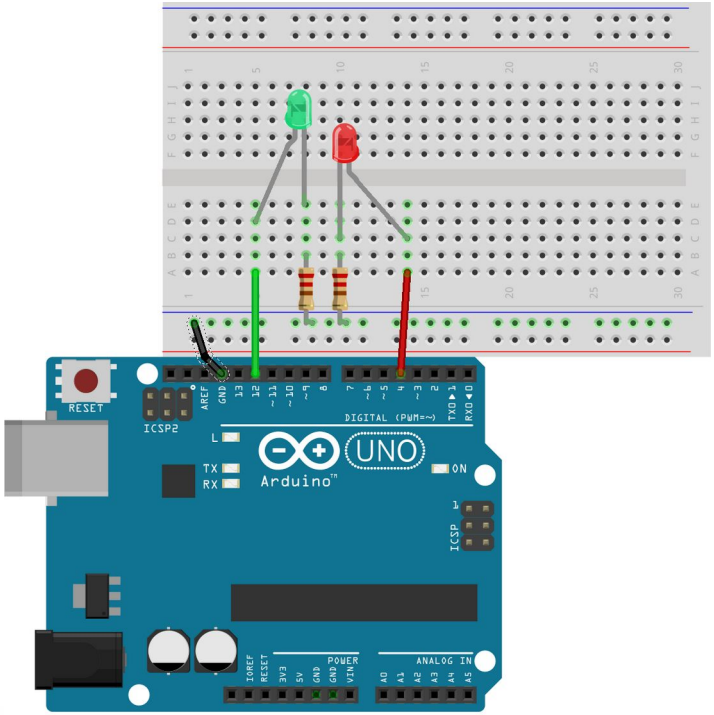




Clicchiamo “carica” e dovremmo vedere il LED integrato che si accende e spegne ad intervalli di un secondo

Step 2 - doppio blink

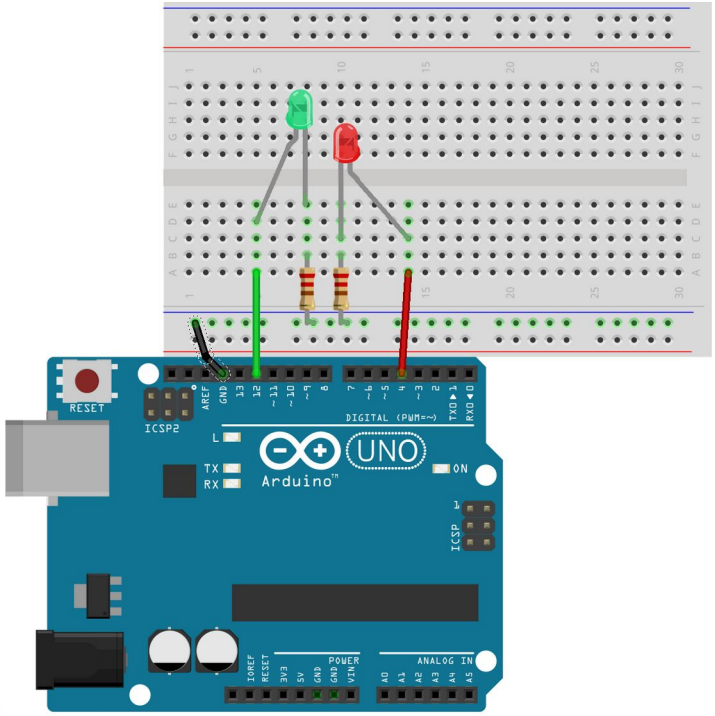
con breadoard



fritzing

Step 2 - doppio blink

con breadoard



fritzing

```
int const  LED_ROSSO=12;
int const  LED_VERDE=4;

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_ROSSO, OUTPUT);
  pinMode(LED_VERDE, OUTPUT);
}

void loop() {
  digitalWrite(LED_ROSSO, HIGH);
  digitalWrite(LED_VERDE, HIGH);
  delay(500);
  digitalWrite(LED_VERDE, LOW);
  delay(500);
  digitalWrite(LED_VERDE, HIGH);
  digitalWrite(LED_ROSSO, LOW);
  delay(500);
  digitalWrite(LED_VERDE, LOW);
  delay(500);
}
```

Arduino Programming Cheat Sheet

Primary source: Arduino Language Reference
https://arduino.cc/en/Reference/

Arduino Cheat Sheet on GitHub

This is a poster-sized cheat sheet for Arduino programmers. It draws primarily from the Arduino Language Reference, including most of the common, basic syntax and a variety of the built-in functions. It is based on a cheat sheet by Gavin Smith and an SVG adaptation by Frederic Dufourg. Additionally, the Arduino Uno board drawing is adapted from an Arduino board drawing in Fritzing (under a CC-BY-SA license).

The latest version of the PDF can be downloaded directly from GitHub here.

Structure & Flow

Basic Program Structure

```
void setup() {
  // Runs once when sketch starts
}
void loop() {
  // Runs repeatedly
}
```

Control Structures

```
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
for (int i = 0; i < 10; i++) { ... }
break;     // Exit a loop immediately
continue;  // Go to next iteration
switch (var) {
  case 1:
    ...
    break;
  case 2:
    ...
    break;
  default:
    ...
}
return x;  // x must match return type
return;    // For void return type
```

Function Definitions

```
<ret. type> <name>(<params>) { ... }
e.g. int double(int x) {return x*2;}
```

Operators

General Operators

=	assignment		
+	add	-	subtract
*	multiply	/	divide
%	modulo		
==	equal to	!=	not equal to
<	less than	>	greater than
<=	less than or equal to		
>=	greater than or equal to		
&&	and	||	or
!	not		

Compound Operators

++	increment
--	decrement
+=	compound addition
-=	compound subtraction
*=	compound multiplication
/=	compound division
&=	compound bitwise and
|=	compound bitwise or

Bitwise Operators

&	bitwise and	|	bitwise or
^	bitwise xor	~	bitwise not
<<	shift left	>>	shift right

Pointer Access

&	reference: get a pointer
*	dereference: follow a pointer

Built-in Functions

Pin Input/Output

Digital I/O - pins 0-13 A0-A5

```
pinMode(pin,
  [INPUT, OUTPUT, INPUT_PULLUP])
int digitalRead(pin)
digitalWrite(pin, [HIGH, LOW])
```

Analog In - pins A0-A5

```
int analogRead(pin)
analogReference(
  [DEFAULT, INTERNAL, EXTERNAL])
```

PWM Out - pins 3 5 6 9 10 11

```
analogWrite(pin, value)
```

Advanced I/O

```
tone(pin, freq_Hz)
tone(pin, freq_Hz, duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin,
  [MSBFIRST, LSBFIRST], value)
unsigned long pulseIn(pin,
  [HIGH, LOW])
```

Time

```
unsigned long millis()
  // Overflows at 50 days
unsigned long micros()
  // Overflows at 70 minutes
delay(msec)
delayMicroseconds(usec)
```

Math

```
min(x, y)   max(x, y)   abs(x)
sin(rad)    cos(rad)    tan(rad)
sqrt(x)     pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

Random Numbers

```
randomSeed(seed) // long or int
long random(max) // 0 to max-1
long random(min, max)
```

Bits and Bytes

```
lowByte(x)    highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn)  // bitn: 0=LSB 7=MSB
```

Type Conversions

```
char(val)     byte(val)
int(val)      word(val)
long(val)     float(val)
```

External Interrupts

```
attachInterrupt(interrupt, func,
  [LOW, CHANGE, RISING, FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```

Libraries

Serial - comm. with PC or via RX/TX

```
begin(long speed)  // Up to 115200
end()
int available()  // #bytes availabl
int read()   // -1 if none availabl
int peek()   // Read w/o removing
flush()
print(data)     println(data)
write(byte)     write(char * string)
write(byte * data, size)
SerialEvent()  // Called if data r
```

SoftwareSerial.h - comm. on any pin

```
SoftwareSerial(rxPin, txPin)
begin(long speed)  // Up to 115200
listen()           // Only 1 can listen
isListening()  // at a time.
read, peek, print, println, write
  // Equivalent to Serial library
```

EEPROM.h - access non-volatile memor

```
byte read(addr)
write(addr, byte)
EEPROM[index]  // Access as array
```

Servo.h - control servo motors

```
attach(pin, [min_uS, max_uS])
write(angle)  // 0 to 180
writeMicroseconds(uS)
   // 1000-2000; 1500 is midpoint
int read()    // 0 to 180
bool attached()
detach()
```

Wire.h - I²C communication

```
begin()      // Join a master
begin(addr) // Join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(byte)              // Step 2
send(char * string)
send(byte * data, size)
endTransmission()       // Step 3
int available() // #bytes available
byte receive()  // Get next byte
onReceive(handler)
onRequest(handler)
```

Variables, Arrays, and Data

Data Types

boolean	true | false	
char	-128 - 127, 'a' '$' etc.	
unsigned char	0 - 255	
byte	0 - 255	
int	-32768 - 32767	
unsigned int	0 - 65535	
word	0 - 65535	
long	-2147483648 - 2147483647	
unsigned long	0 - 4294967295	
float	-3.4028e+38 - 3.4028e+38	
double	currently same as float	
void	i.e., no return value	

Strings

```
char str1[8] =
  {'A','r','d','u','i','n','o','\0'};
  // Includes \0 null termination
char str2[8] =
  {'A','r','d','u','i','n','o'};
  // Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numeric Constants

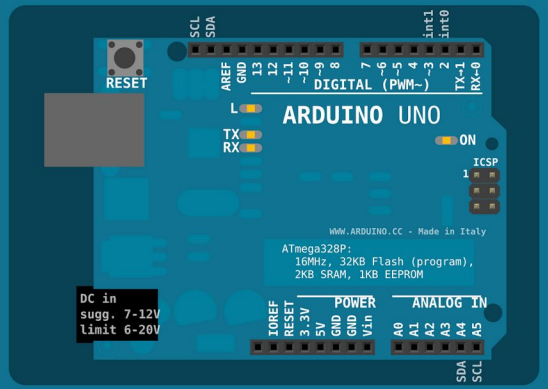
123	decimal
0b01111011	binary
0173	octal - base 8
0x7B	hexadecimal - base 16
123U	force unsigned
123L	force long
123UL	force unsigned long
123.0	force floating point
1.23e6	1.23*10^6 = 1230000

Qualifiers

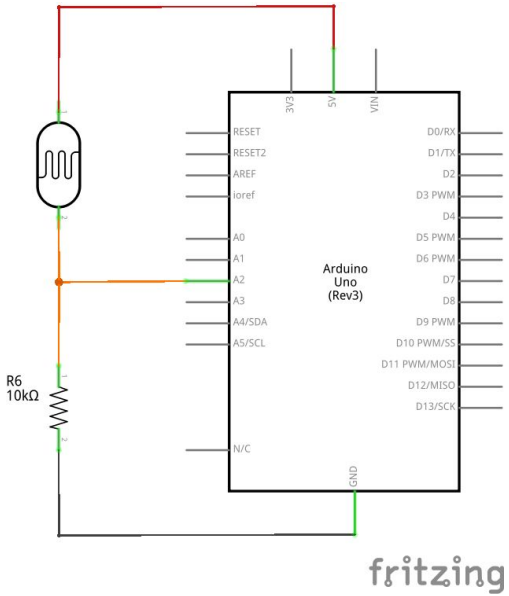
static	persists between calls
volatile	in RAM (nice for ISR)
const	read-only
PROGMEM	in flash

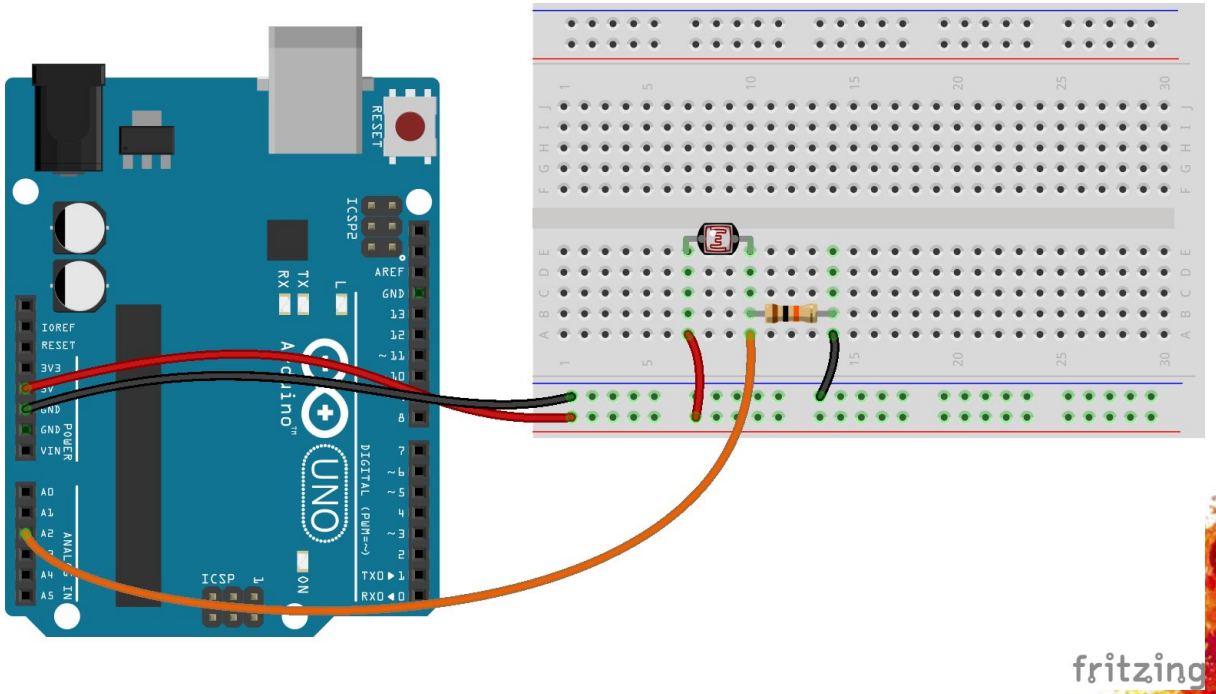
Arrays

```
int myPins[] = {2, 4, 8, 3, 6};
int myInts[6];   // Array of 6 ints
myInts[0] = 42;  // Assigning first
                 // index of myInts
myInts[6] = 12;  // ERROR! Indexes
                 // are 0 though 5
```



Misuriamo il valore di una fotoresistenza





Slides credits: www.nova-aps.it

Semplice codice per misure analogiche.

Questo codice non esegue calcoli, stampa solo ciò che rileva come intensità luminosa, in modo qualitativo.

Questa volta vogliamo utilizzare il serial monitor

E vogliamo stampare a schermo delle parole a seconda dell'intensità luminosa

```
int photocellPin = 0;
// the cell and 10K pulldown are connected to a0
int photocellReading;
// the analog reading from the analog resistor divider

void setup(void) {
  // We'll send debugging information via the Serial monitor
  Serial.begin(9600);
}

void loop(void) {
  photocellReading = analogRead(photocellPin);

  Serial.print("Analog reading = ");
  Serial.print(photocellReading);     // the raw analog reading

  if (photocellReading < 10) {
    Serial.println(" - Dark");
  } else if (photocellReading < 200) {
    Serial.println(" - Dim");
  } else if (photocellReading < 500) {
    Serial.println(" - Light");
  } else if (photocellReading < 800) {
    Serial.println(" - Bright");
  } else {
    Serial.println(" - Very bright");
  }
  delay(1000);
}
```

Strumenti > Serial monitor

File Modifica Sketch Strumenti Aiuto

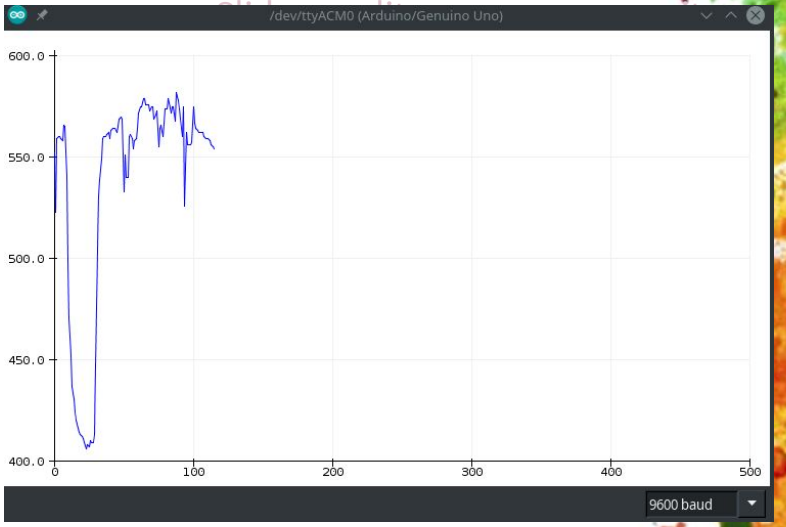
Blink_2led_serialmon

```
int photocellPin = 0;
// the cell and 10K pulldown are connected to a0
int photocellReading;
// the analog reading from the analog resistor divider

void setup(void) {
  // We'll send debugging information via the Serial monitor
  Serial.begin(9600);
}

void loop(void) {
  photocellReading = analogRead(photocellPin);

  Serial.print("Analog reading = ");
  Serial.print(photocellReading);     // the raw analog reading

  if (photocellReading < 10) {
    Serial.println(" - Dark");
  } else if (photocellReading < 200) {
    Serial.println(" - Dim");
  } else if (photocellReading < 500) {
    Serial.println(" - Light");
  } else if (photocellReading < 800) {
    Serial.println(" - Bright");
  } else {
    Serial.println(" - Very bright");
  }
  delay(1000);
}
```

/dev/ttyACM0 (Arduino/Genui

```
Analog reading = 566 - Bright
Analog reading = 567 - Bright
Analog reading = 574 - Bright
Analog reading = 570 - Bright
Analog reading = 569 - Bright
Analog reading = 572 - Bright
Analog reading = 571 - Bright
Analog reading = 555 - Bright
Analog reading = 555 - Bright
Analog reading = 515 - Bright
Analog reading = 487 - Light
Analog reading = 462 - Light
Analog reading = 447 - Light
Analog reading = 436 - Light
Analog reading = 426 - Light
Analog reading = 422 - Light
Analog reading = 414 - Light
Analog reading = 414 - Light
Analog reading = 413 - Light
Analog reading = 410 - Light
Analog reading = 412 - Light
Analog reading = 417 - Light
Analog reading = 415 - Light
Analog reading = 424 - Light
Analog reading = 424 - Light
Analog reading = 424 - Light
```

☑ Scorrimento automatico Nessun fine ri



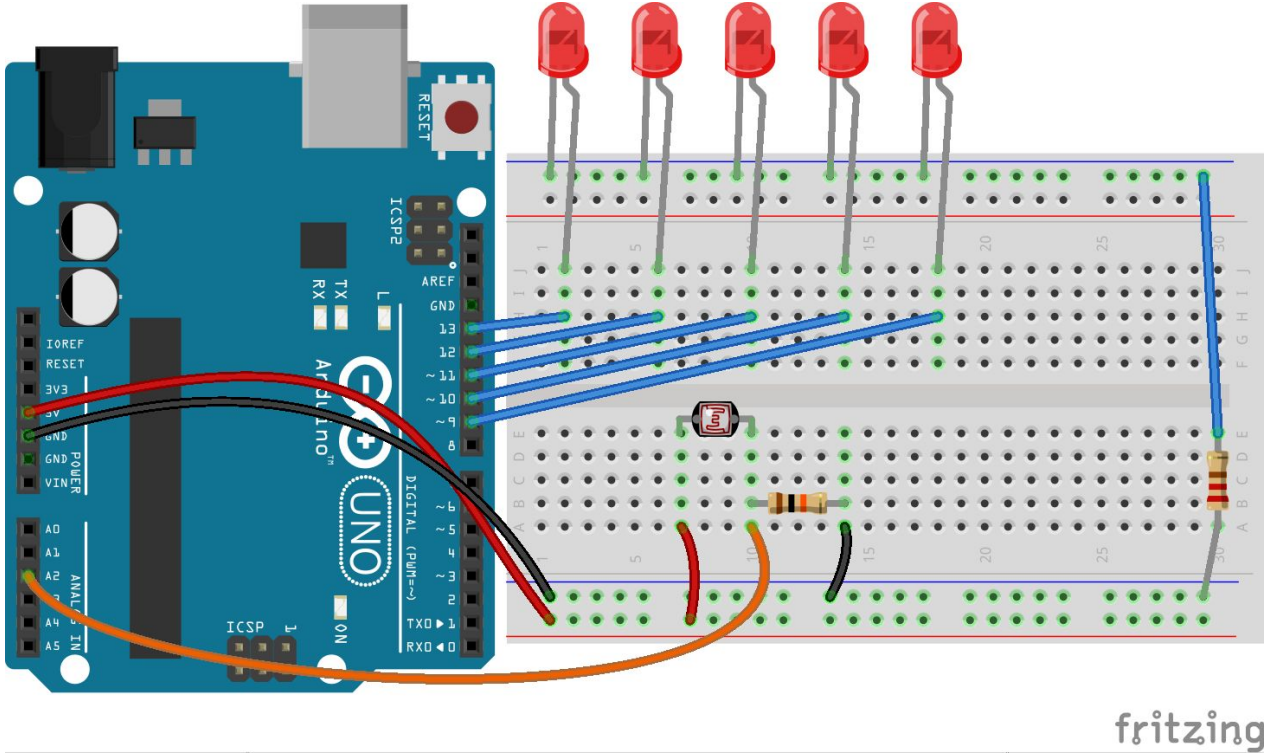
Strumenti > Serial plotter

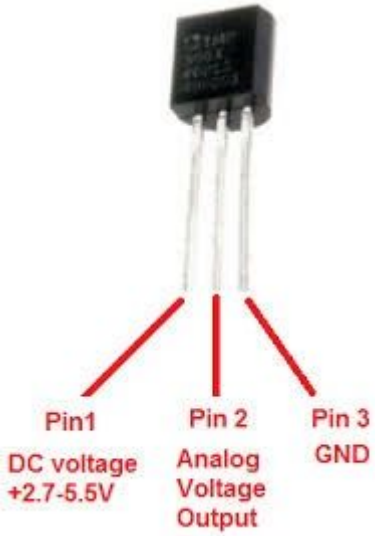
INFN FERRARA

LAB 2GO SCIENZA

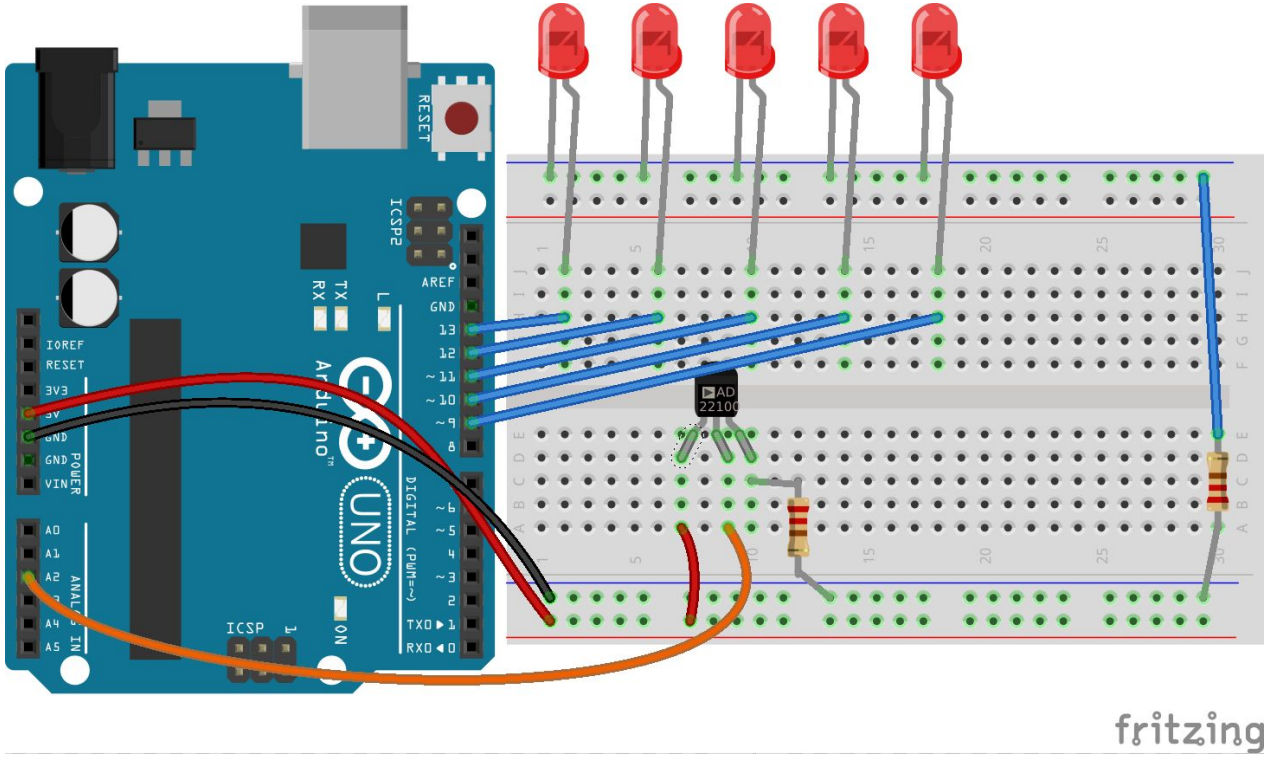
Esercizio 1

Riesci a creare una scala di 5 LED che si accendano in base alla luminosità percepita dalla fotoresistenza?





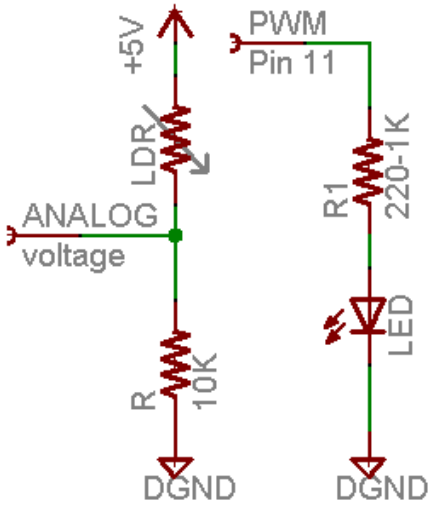
Cosa cambia se sostituiamo il sensore di temperatura alla fotoresistenza?

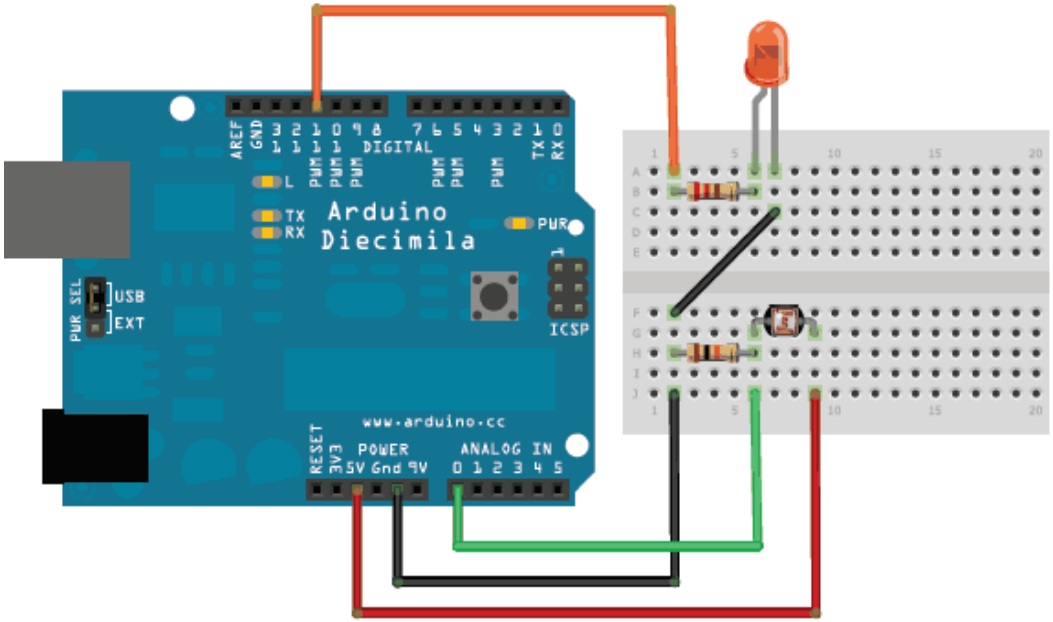


FADE e rimappare i valori

Torniamo alla fotoresistenza.

Vogliamo che l'intensità del LED vari in maniera inversamente proporzionale all'intensità della luce misurata





la funzione map()

Mappa un numero da un intervallo all'altro.

Sintassi: `map (value, fromLow, fromHigh, toLow, toHigh)`

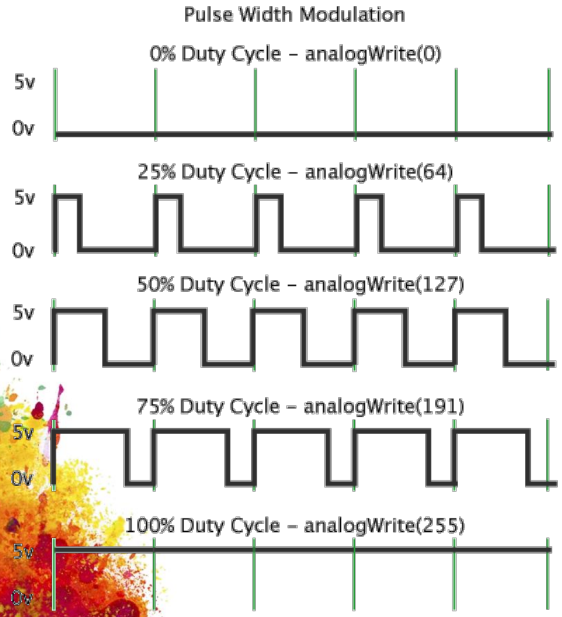
Non vincola i valori all'interno dell'intervallo, perché i valori fuori intervallo a volte sono intesi e utili. La funzione `constrain ()` può essere utilizzata prima o dopo questa funzione, se si desiderano limiti agli intervalli.

Si noti che i "limiti inferiori" di entrambi gli intervalli possono essere più grandi o più piccoli dei "limiti superiori", quindi la funzione map () può essere utilizzata per invertire un intervallo di numeri. Funziona correttamente anche con numeri negativi.

La funzione map () gestisce operazioni tra valori interi: i resti frazionari vengono troncati e non arrotondati o mediati.

```
long map(long x, long in_min, long in_max, long out_min, long out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

Un pin digitale che si finge analogico: PMW



Pulse Width Modulation, o PWM, è una tecnica per ottenere risultati analogici con mezzi digitali.

Il controllo digitale viene utilizzato per creare un'onda quadra che può simulare tensioni tra full on (5 Volt) e off (0 Volt) cambiando l'intervallo di tempo in cui il segnale è full on. La durata di "on time" è definita larghezza dell'impulso. Per ottenere valori analogici variabili, si modifica o si modula la larghezza dell'impulso.

Il segnale ricevuto è quindi "indistinguibile" da una tensione costante tra 0 e 5v che controlla la luminosità del LED.

Il range è 0-255.

```
1.  int led = 9;
2.  int brightness = 0;
3.  int fadeAmount = 5;
4.  int photoresistor =A0;
5.  int light_in;

6.  void setup() {
7.    pinMode(led, OUTPUT);
8.    pinMode(photoresistor, INPUT);
9.    Serial.begin(9600);
10. }

11. void loop() {
12.   light_in=analogRead(photoresistor);

13.   brightness=200-map(light_in, 80, 1000, 0, 255);
14.   Serial.print("light in=");
15.   Serial.print(light_in);
16.   Serial.print("\tbrightness=");
17.   Serial.println(brightness);
18.   analogWrite(led, brightness);
19.   delay(30);
20. }
```

i valori che scegliamo per la funzione map sono calibrati sulla luminosità della stanza

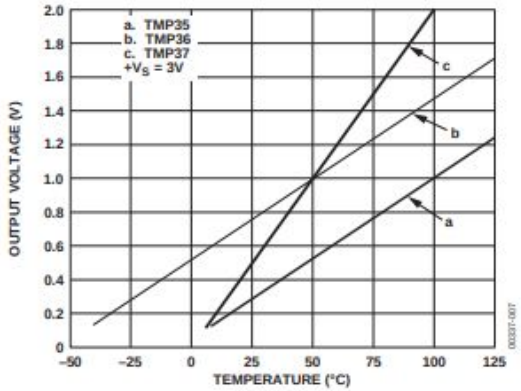


Figure 6. Output Voltage vs. Temperature

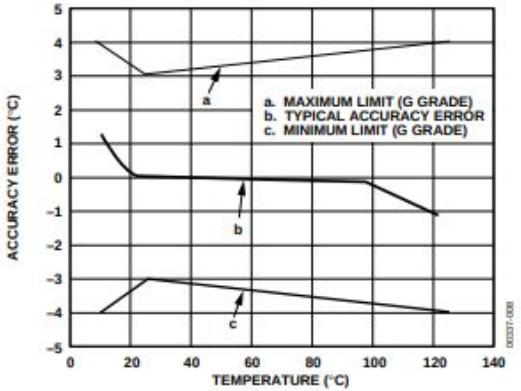


Figure 7. Accuracy Error vs. Temperature

La risposta in tensione dei vari sensori dipende dalla loro curva caratteristica.

Per avere ulteriori indicazioni sull'accuratezza e la linearità della risposta del sensore, è necessario scaricare le schede tecniche dai siti dei produttori.

Bibliografia e sitografia

Links Utili:

- Sito ufficiale Arduino: https://www.arduino.cc
- https://en.wikipedia.org/wiki/Data_acquisition
- https://www.nutaq.com/blog/analog-digital-%E2%80%93-part-2-conversion-process
- https://www.debiagi.cloud/coding-robotica/
- https://learn.adafruit.com

Credits:

- Associazione NOVA APS: https://www.nova-aps.it
- Progetto PON a cura di Valentina Rolando presso il Liceo Carducci (2021)
- Laboratorio didattico a cura di Alessandro Bertagnon, festival How I Met Science! 4
- Lezioni di Marcello Bonfè, Elena Mainardi per il progetto UniFe How I Met Science! 3