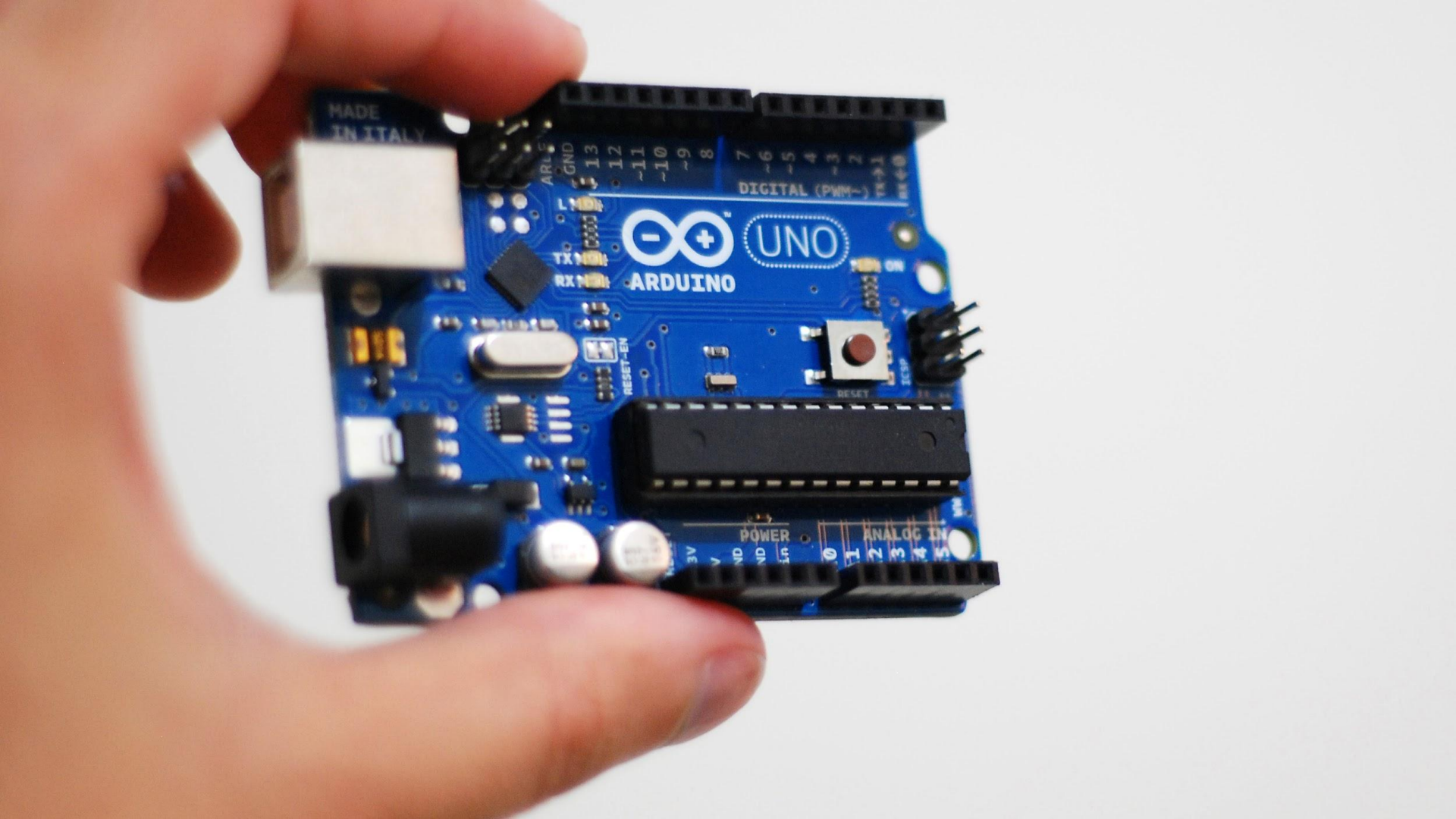
Arduino & co.

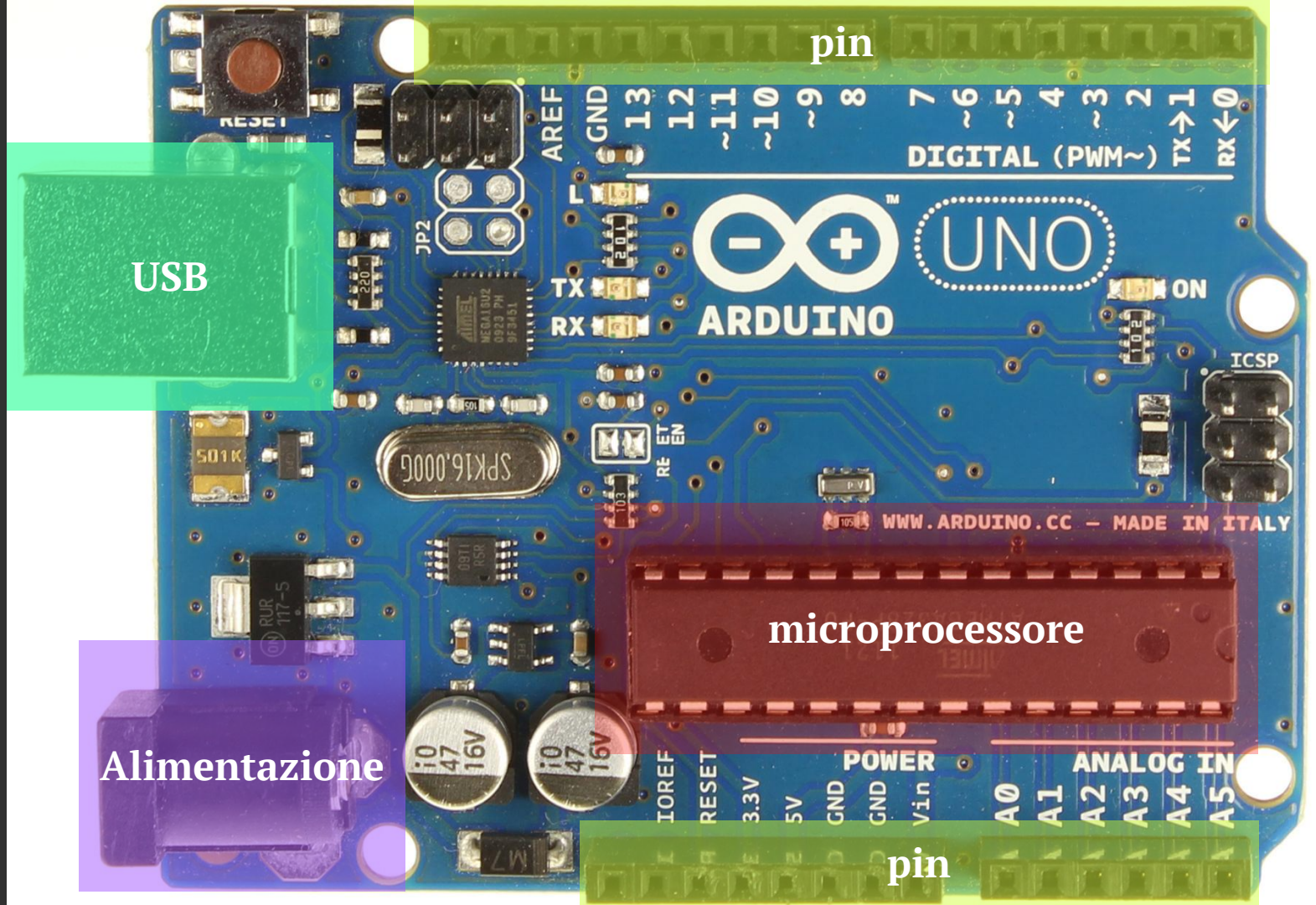
Introduzione alle schede di prototipazione


MADE IN ITALY
DIGITAL (PWM~)
UNO
ARDUINO
TX
RX
ON
RESET
POWER
ANALOG IN

Cos’è una scheda di prototipazione

Una base per realizzare progetti elettronici composta da:

- Un **microprocessore**
- Dei **pin** di entrata e di uscita
- Delle **interfacce** per inserire il proprio codice (USB)
- Presa di alimentazione, oscillatore, etc…


pin
USB
microprocessore
Alimentazione
pin

Arduino è stato il primo kit di prototipazione pensato per un pubblico non esperto, ed è completamente **open source**

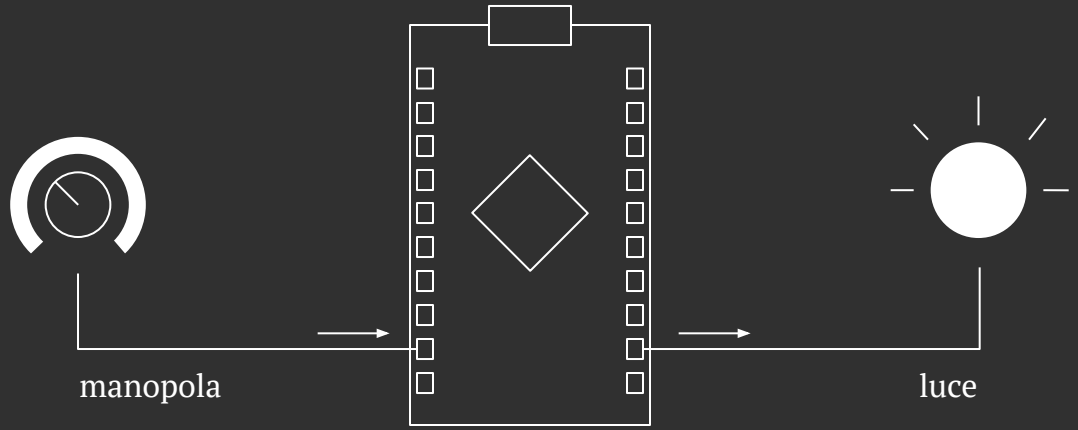
Alimentazione

Arduino può essere alimentato direttamente dalla sua **porta USB**

In alternativa, è possibile collegare i pin **Vin** e **G** (ground) con i fili di un alimentatore a **5 Volt**

Pinout

I pin sono delle porte di **ingresso** o **uscita** per dei segnali elettrici



I pin possono essere di **due tipi**:

Analogico

Il segnale ha un valore variabile da 0 a 1023 (8 bit)

Esempio: un sensore di temperatura trasforma il valore termico in segnale elettrico

Digitale

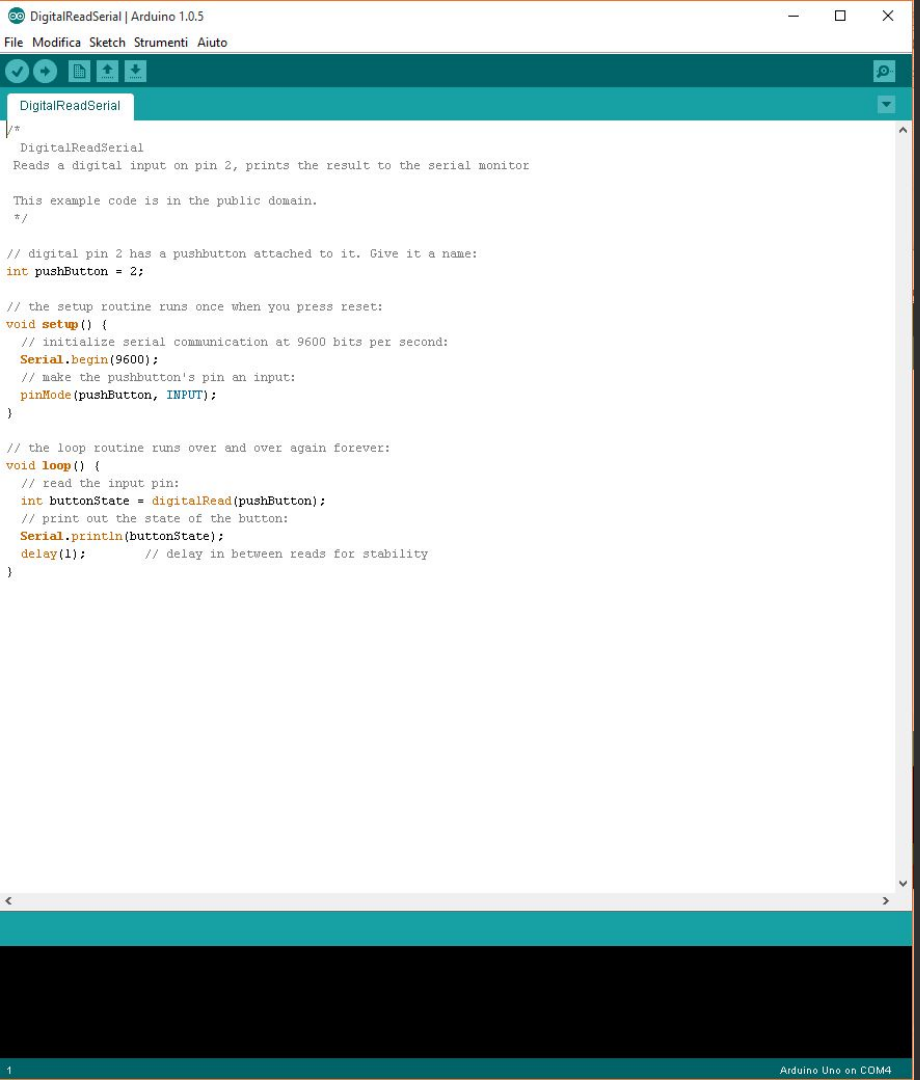
Il segnale ha 2 possibili valori: HIGH (1) o LOW (0)

Esempio: una fotocellula rileva la presenza o l'assenza di un corpo fisico nel suo campo

Come si programma

Per programmare Arduino è necessario scrivere un **programma** in codice, usando un linguaggio di programmazione (C)

Il software in cui viene scritto il codice è chiamato **IDE** (Integrated Development Environment)



Arduino IDE

- Verifica la correttezza del codice
- Carica il codice sulla scheda
- Crea un nuovo sketch
- Carica uno sketch salvato
- Salva lo sketch corrente
- Apre il monitor seriale

← Console per errori e messaggi di sistema

Aiuto, da dove comincio?

Guida ufficiale - https://www.arduino.cc/en/Guide/HomePage

WELCOME TO ARDUINO! BEFORE YOU START CONTROLLING THE WORLD AROUND YOU, YOU'LL NEED TO SET UP THE SOFTWARE TO PROGRAM YOUR BOARD

The Arduino Software (IDE) allows you to write programs and upload them to your board. In the Arduino Software page you will find two options:

1. If you have a reliable Internet connection, you should use the online IDE (Arduino Web Editor). It will allow you to save your sketches in the cloud, having them available from any device and backed up. You will always have the most up-to-date version of the IDE without the need to install updates or community generated libraries.

2. If you would rather work offline, you should use the latest version of the desktop IDE.

Code online on the Arduino Web Editor

To use the online IDE simply follow these instructions. Remember that boards work out-of-the-box on the Web Editor, no need to install anything.

Install the Arduino Desktop IDE

To get step-by-step instructions select one of the following link accordingly to your operating system.

- Windows
- Mac OS X
- Linux
- Portable IDE (Windows and Linux)

Choose your board in the list here on the right to learn how to get started with it and

Instructions for our boards:

- 101
- Due
- Intel Edison
- ISP
- LilyPad, LilyPad Simple and LilyPad SimpleSnap
- LilyPad USB
- MEGA2560
- MKR1000
- MKR WiFi 1010
- MKR FOX 1200
- MKR WAN 1300
- MKR GSM 1400
- MKR Vidor 4000
- MKRZERO
- Nano
- Pro
- Pro Mini
- UNO
- UNO WiFi Rev.2
- Zero
- Yún Rev.2

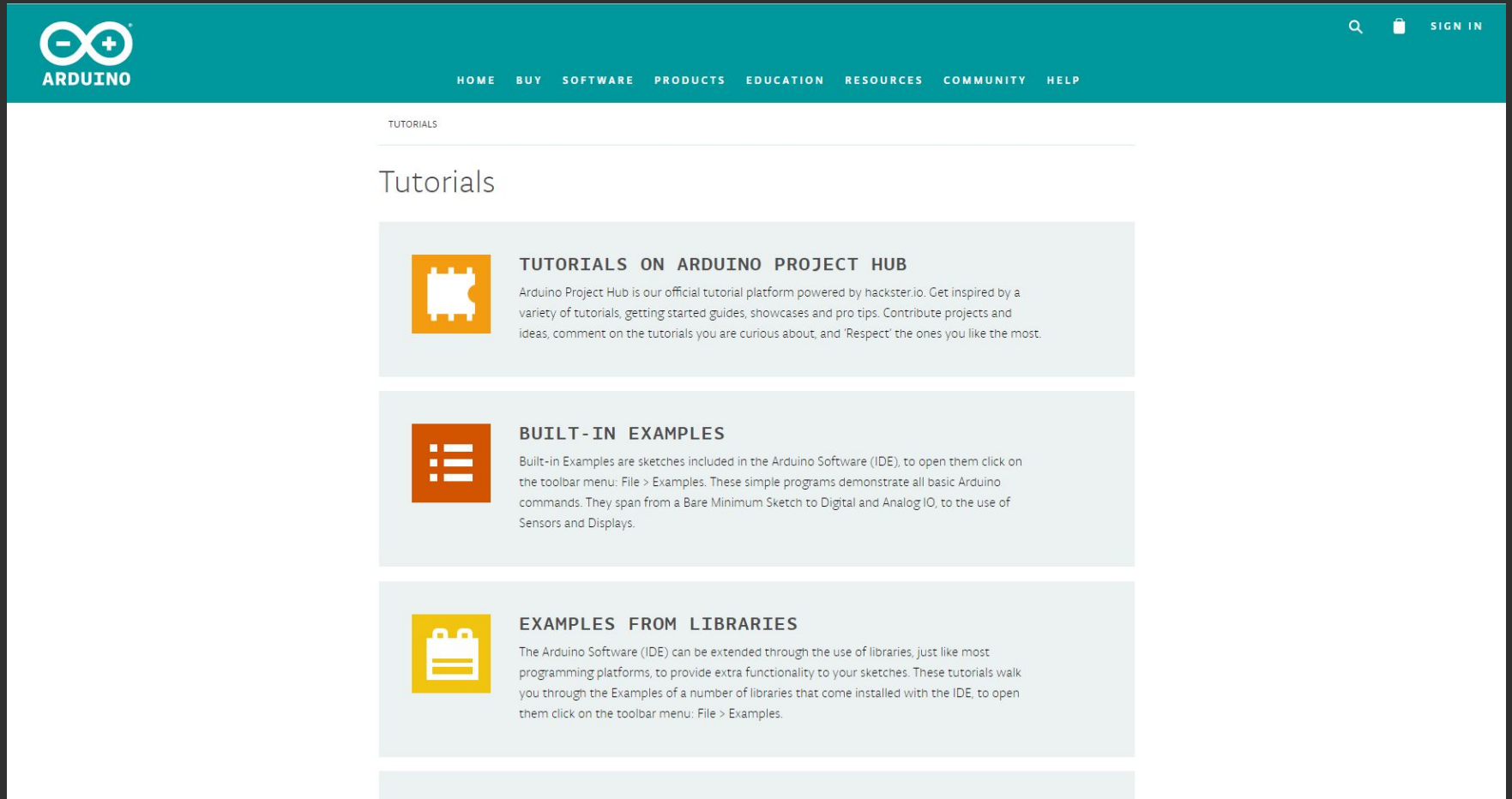
Instructions for shields:

- Arduino MKR 485 Shield
- Arduino MKR CAN Shield
- Arduino MKR Connector Carrier
- Arduino MKR ETH Shield
- Arduino MKR MEM Shield
- WiFi Shield

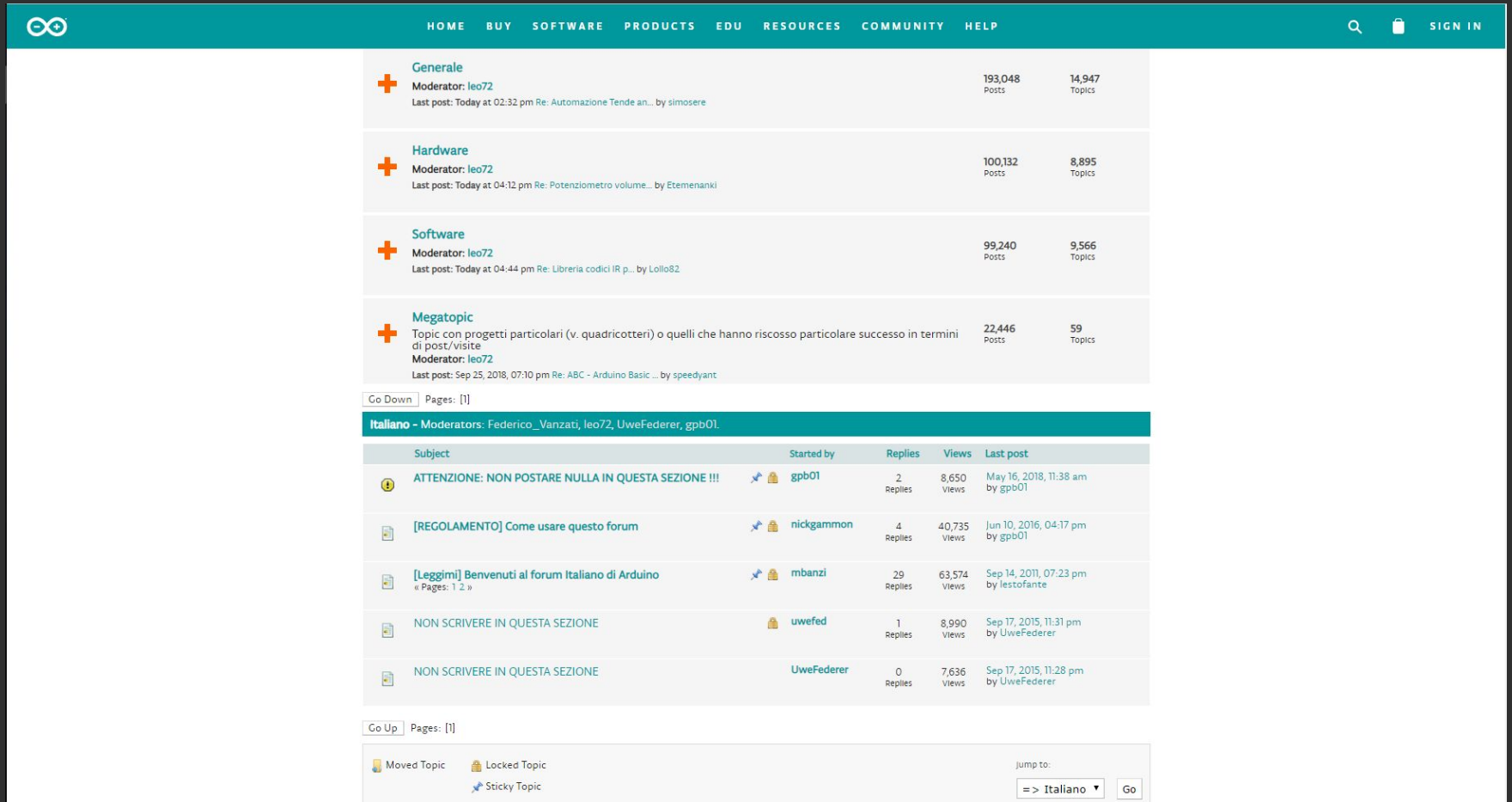
Instructions for retired boards:

- ADK

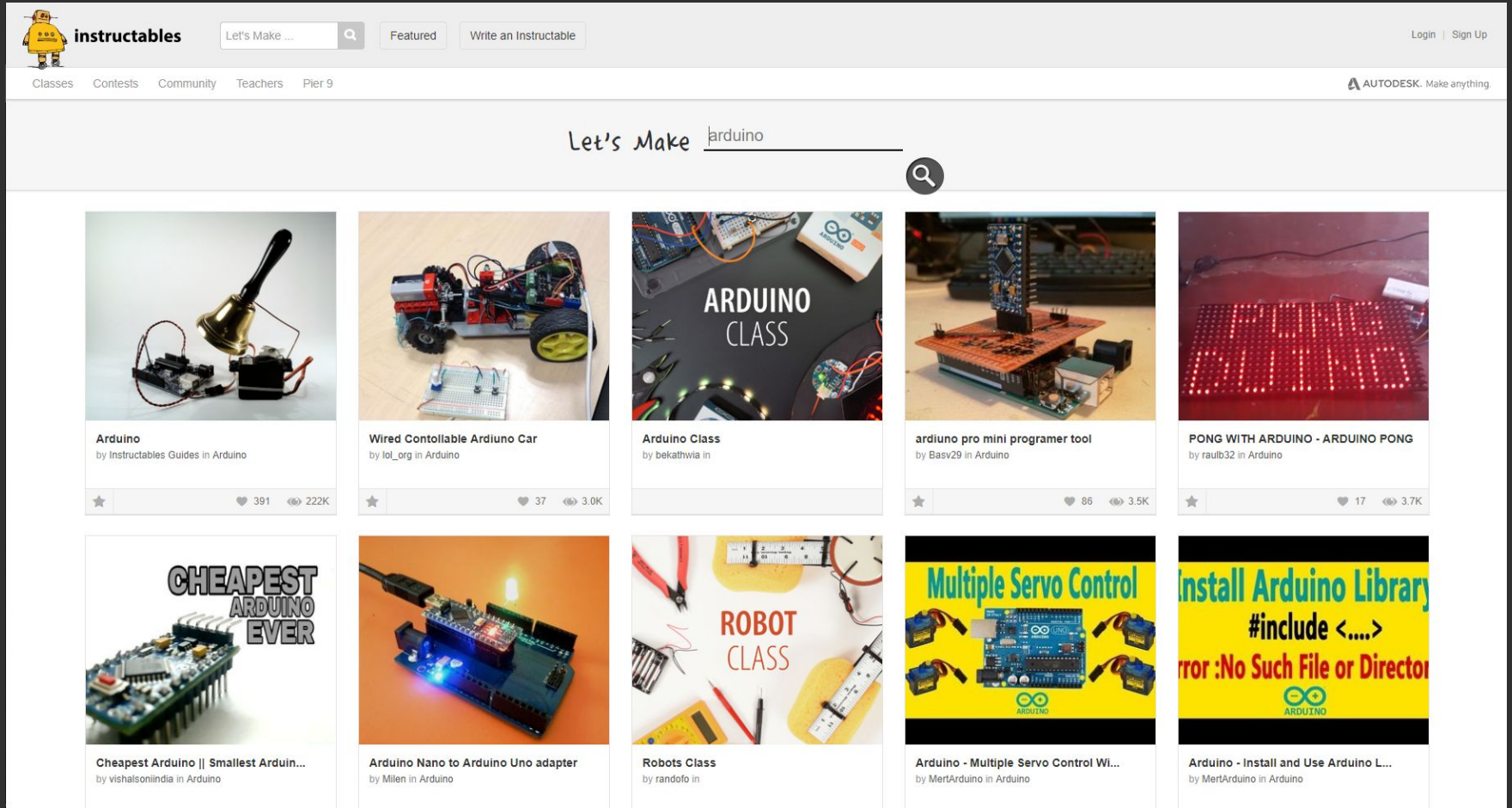
Tutorial - https://www.arduino.cc/en/Tutorial/HomePage



Forum Arduino in italiano - https://forum.arduino.cc/index.php?board=34.0



Instructables - https://www.instructables.com/howto/arduino/



Ma non so nulla di elettronica...

Shield e Breakout

Shield e Breakout

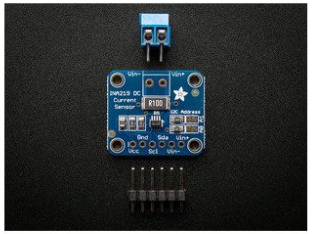
Triple-axis Accelerometer+Magnetometer (Compass) Board - LSM303

PRODUCT ID: 1120

He told you "Go West, young maker!" - but you don't know which way is West! Ah, if only you had this triple-axis accelerometer/magnetometer compass module. Inside are two sensors, one is a classic 3-axis accelerometer, which can tell you which direction is down towards the Earth (by measuring gravity). The other is a magnetometer that can sense where the strongest magnetic force is coming from, generally used to detect magnetic...

ADD TO CART

$14.95
IN STOCK



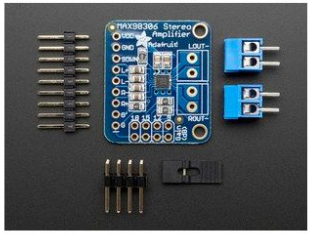
INA219 High Side DC Current Sensor Breakout - 26V ±3.2A Max

PRODUCT ID: 904

This breakout board will solve all your power-monitoring problems. Instead of struggling with two multimeters, you can just use the handy INA219B chip on this breakout to both measure both the high side voltage and DC current draw over I2C with 1% precision. Most current-measuring devices such as our current panel meter are only good for low side measuring. That means that unless you want to get a battery involved, you have to stick the...

ADD TO CART

$9.95
IN STOCK



Stereo 3.7W Class D Audio Amplifier - MAX98306

PRODUCT ID: 987

This incredibly small stereo amplifier is surprisingly powerful - able to deliver 2 x 3.7W channels into 3 ohm impedance speakers. Inside the miniature chip is a class D controller, able to run from 2.7V-5.5VDC. Since the amp is a class D, its incredibly efficient (over 90% efficient when driving an 8Ω speaker at over a Watt) - making it perfect for portable and battery-powered projects. It has built in thermal and over-current protection...

ADD TO CART

$8.95
IN STOCK

Ma non so programmare...

Setup e loop

```
/* In questa parte si devono definire le variabili GLOBALI
*/

// la routine di setup viene eseguita una volta all'avvio:
void setup() {

     /* qui si inizializzano le variabili, si imposta lo stato dei pin,
     si attivano le librerie da usare, si imposta la comunicazione seriale*/

}

// la routine di loop viene eseguita ciclicamente (davvero!?):
void loop() {

     // istruzioni da ripetere in modo ricorsivo;

}
```

Clock

Tutte le operazioni effettuate dal microprocessore della scheda (lettura e scrittura dei pin, svolgimento di calcoli, etc) sono **temporizzate** da un segnale costante nel tempo chiamato **clock**

La velocità del clock su Arduino è di **16 Mhz**

16.000.000 cicli al secondo (!)

Tipi di variabili

Si possono usare vari tipi di variabili per la scrittura di uno sketch, ognuno destinato ad un **tipo di dato** diverso. Ecco i principali:

```
// un int è un numero intero, senza cifre decimali
int intero = 10;

// un float è un numero con cifre decimali
float decimale = 0.5;

// un boolean è una variabile che può essere vera o falsa
boolean acceso = true;

// una string è una stringa di testo
string messaggio = “Ciao mondo!”;
```

Il nostro primo sketch: blink

```
int led = 13;

void setup() {
      pinMode(led, OUTPUT); //inizializza il pin digitale come output
}

void loop() {
      digitalWrite(led, HIGH);   //accendi il LED (ALTO come livello di voltaggio)

      delay(1000);     //aspetta un secondo (1000 millisecondi)

      digitalWrite(led, LOW);     //LED spento portando il voltaggio BASSO

      delay(1000);     //aspetta un secondo
}
```

Comunicazione seriale

Su Arduino è presente un **canale di comunicazione** che viaggia attraverso la porta USB (ma non solo)

La comunicazione seriale consente l'invio e la ricezione di messaggi **durante l'esecuzione** dei programmi. L'interlocutore può essere un computer, un'altra scheda, etc

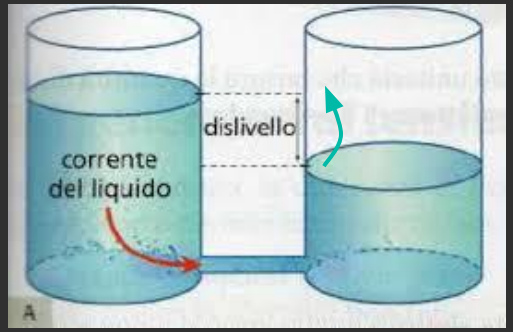
L'uso principale della porta seriale è il **debug** dei programmi. In qualunque momento si può sapere lo stato dell'esecuzione (che valore ha una variabile? Le funzioni fanno quello che volevo?)

Tensione e Corrente

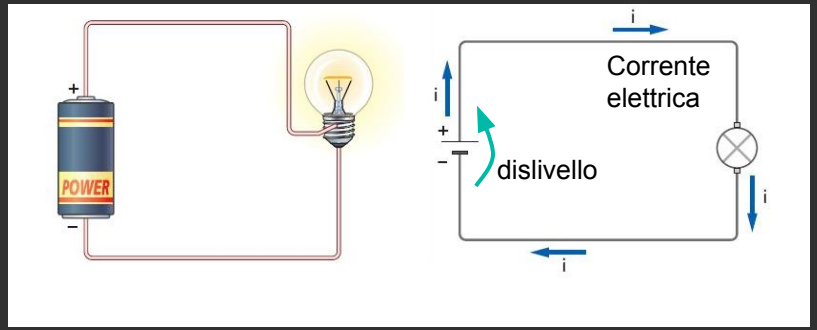
Tensione e Corrente

In un circuito elettrico ci sono due grandezze fondamentali: **tensione** e **corrente**.

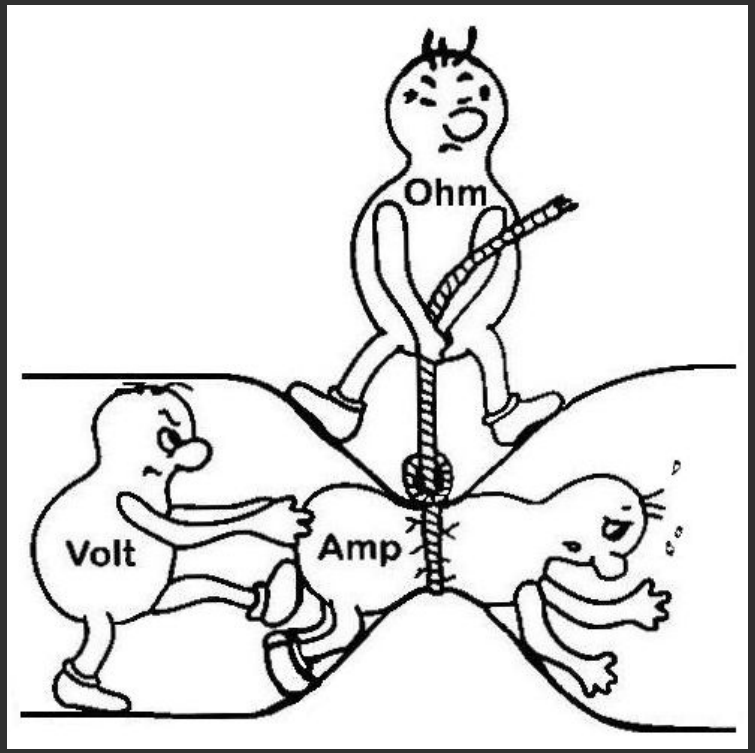
La **corrente** misura quanti **elettroni** attraversano una sezione del circuito in un secondo. Si misura in **Ampère** [A] e si indica con **I** (**intensità** di corrente).

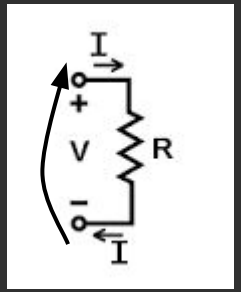


La **tensione** misura la **differenza di energia potenziale elettrica** tra due punti in un circuito, ovvero quanto forte è l'attrazione elettrica (la tensione appunto) tra quei due punti. Si misura in **Volt** [V] e si indica con **V**.



Relazione Tensione e Corrente

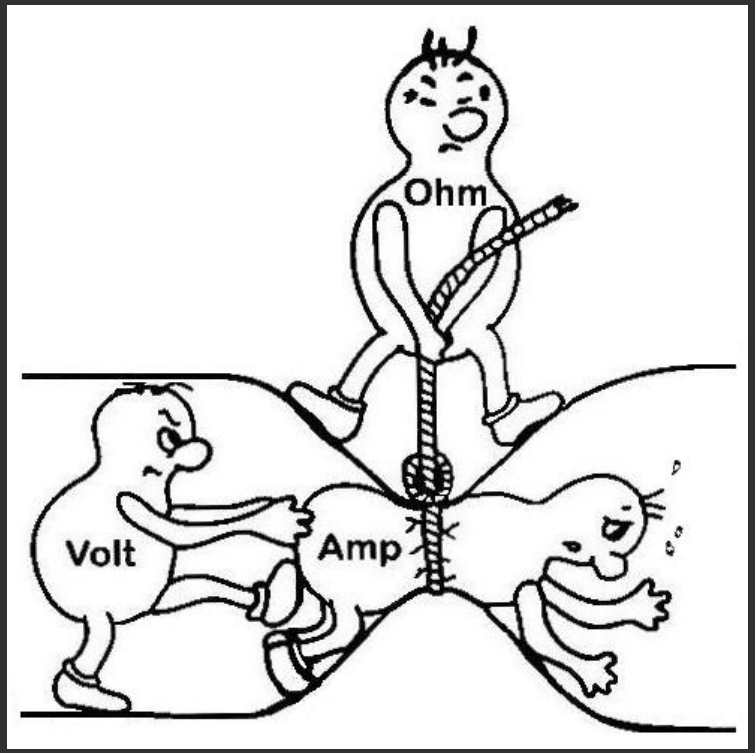


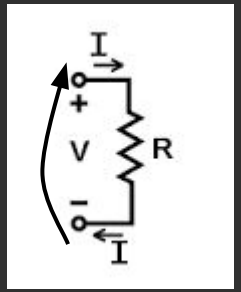


Il **rapporto** tra tensione e corrente si chiama **resistenza** e si misura in Ohm

$$R = \frac{V}{I} \quad [\text{Ohm}]$$

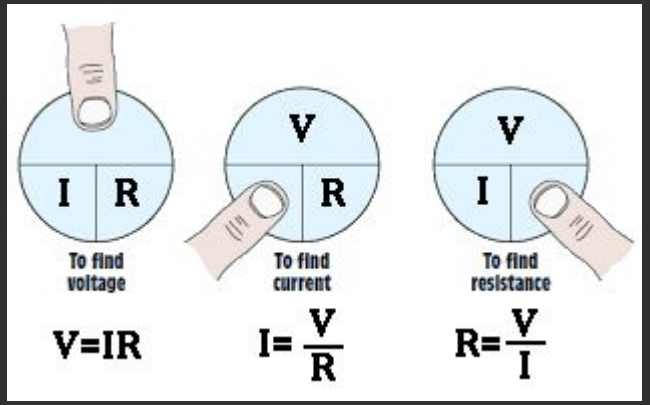
Relazione Tensione e Corrente





Il **rapporto** tra tensione e corrente si chiama **resistenza** e si misura in Ohm

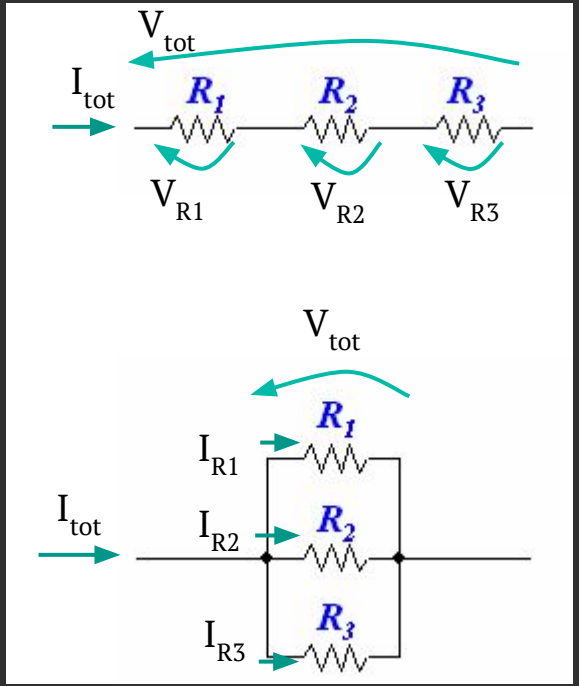
$$R = \frac{V}{I} \quad [\text{Ohm}]$$



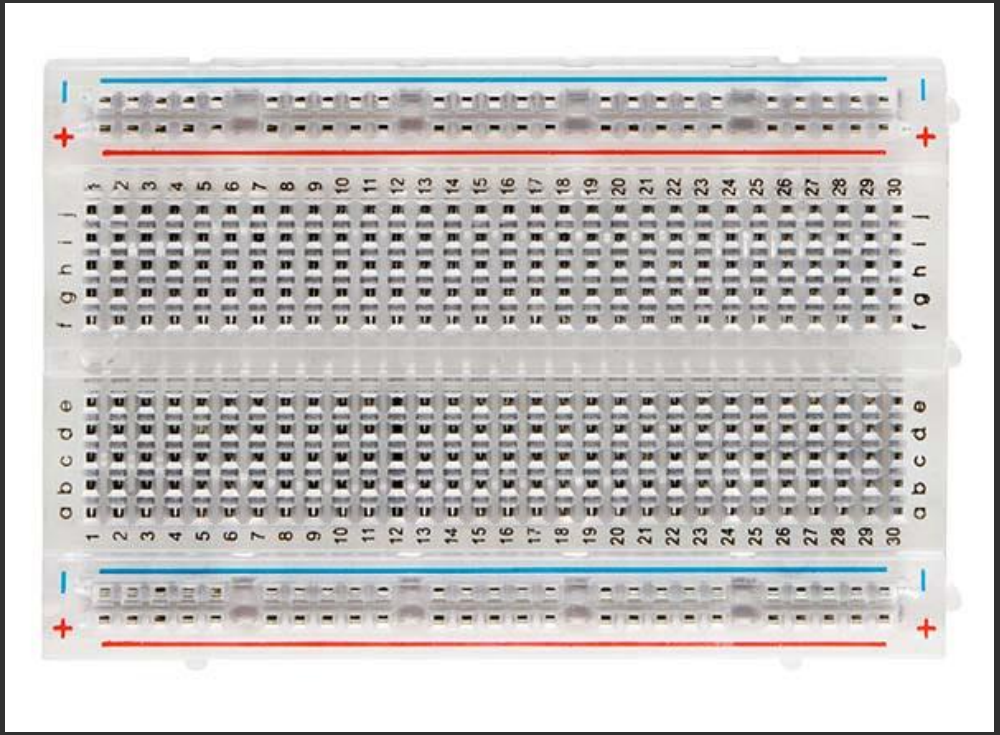
Serie e Parallelo

In un circuito elettrico due o più componenti sono in **serie** quando sono attraversati dalla **stessa corrente**.

In un circuito elettrico due o più componenti sono in **parallelo** quando sono sottoposti alla **stessa tensione**.

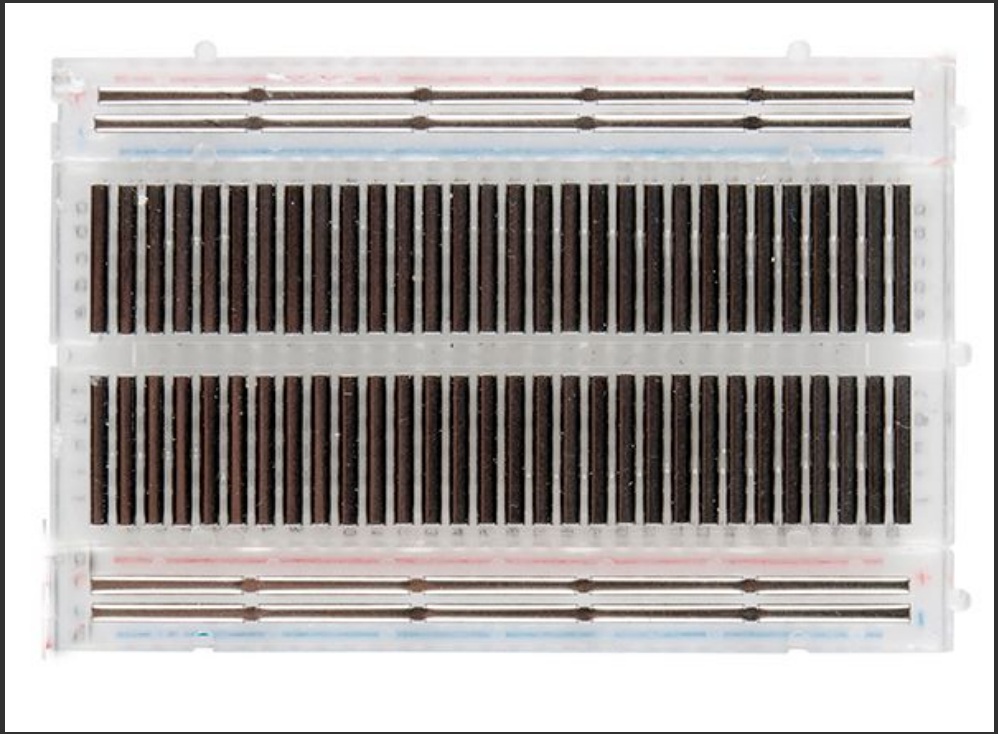


Breadboard



È una **basetta** di prototipizzazione in cui le **connessioni** tra un componente e l'altro sono **già presenti**.
È necessario **conoscerle** per connettere **correttamente** i componenti.

Breadboard



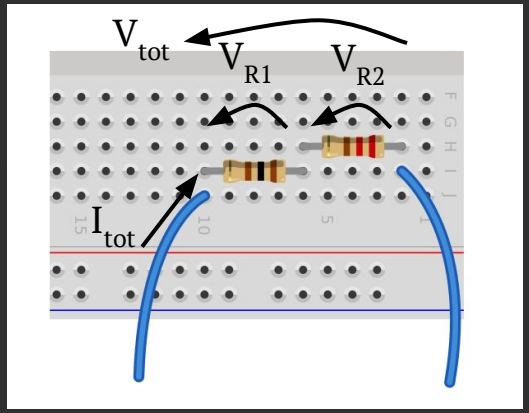
È una **basetta** di prototipizzazione in cui le **connessioni** tra un componente e l'altro sono **già presenti**.
È necessario **conoscerle** per connettere **correttamente** i componenti.

Le linee in alto e in basso sono cortocircuitate **orizzontalmente**

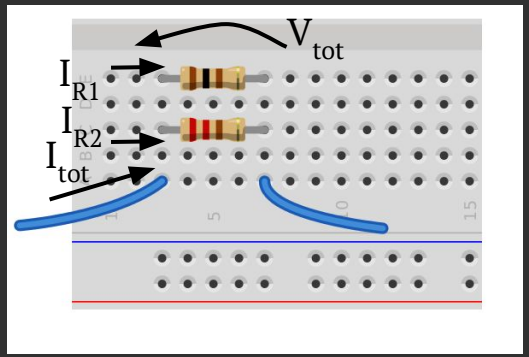
I 5 fori centrali sono cortocircuitati **verticalmente**

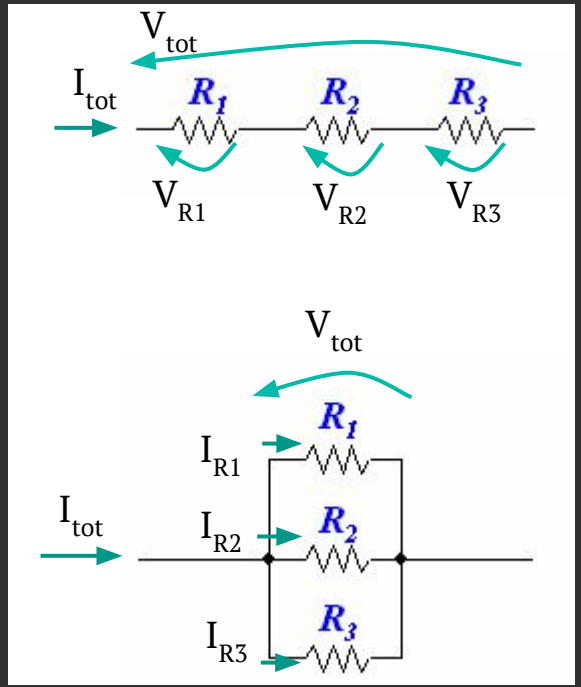
Serie e Parallelo su Breadboard

Serie:
uno **dopo** l'altro



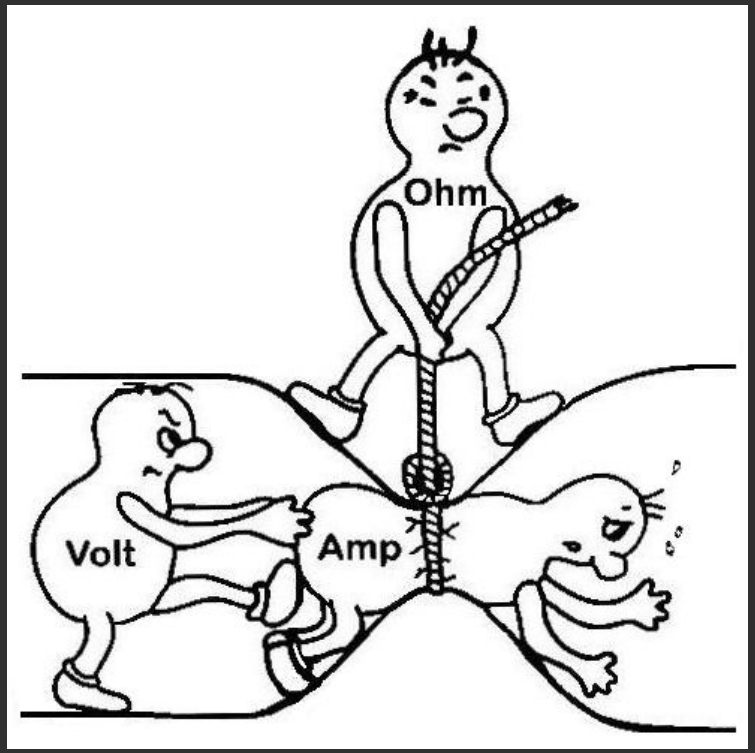
Parallelo:
uno **insieme** all'altro.

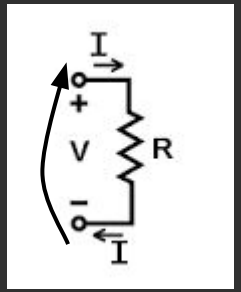




Caratteristica tensione corrente

Resistenza





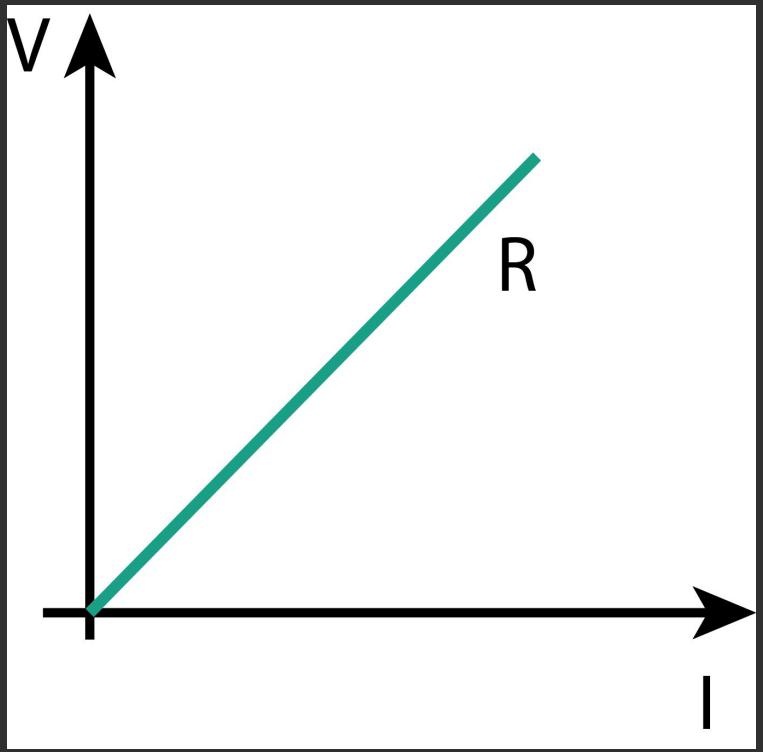
Il **rapporto** tra tensione e corrente si chiama **resistenza** e si misura in Ohm

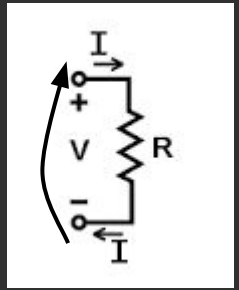
$$R = \frac{V}{I} \quad [\text{Ohm}]$$

Ma anche

$$V = RI$$

Resistenza





Il **rapporto** tra tensione e corrente si chiama **resistenza** e si misura in Ohm

$$R = \frac{V}{I} \quad [\text{Ohm}]$$

Ma anche

$$V = RI$$

A parità di **resistenza**, ad una **corrente maggiore**, corrisponderà una **tensione maggiore**

Resistenze

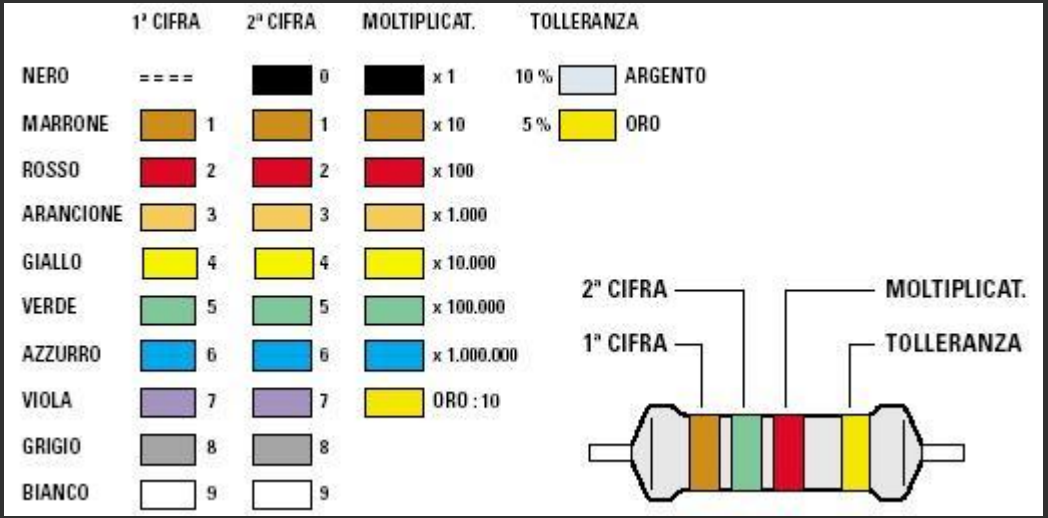
Le resistenze variabili sono **comode**, ma devo regolarle **tutte a mano** e possono essere **accidentalmente modificate** durante il funzionamento causando **errori**

Se i potenziometri non sono indispensabili, è **consigliabile** usare delle resistenze **fisse**, che hanno valori **codificati** e leggibile tramite il **codice colori**

Resistenze

Le resistenze variabili sono **comode**, ma devo regolarle **tutte a mano** e possono essere **accidentalmente modificate** durante il funzionamento causando **errori**

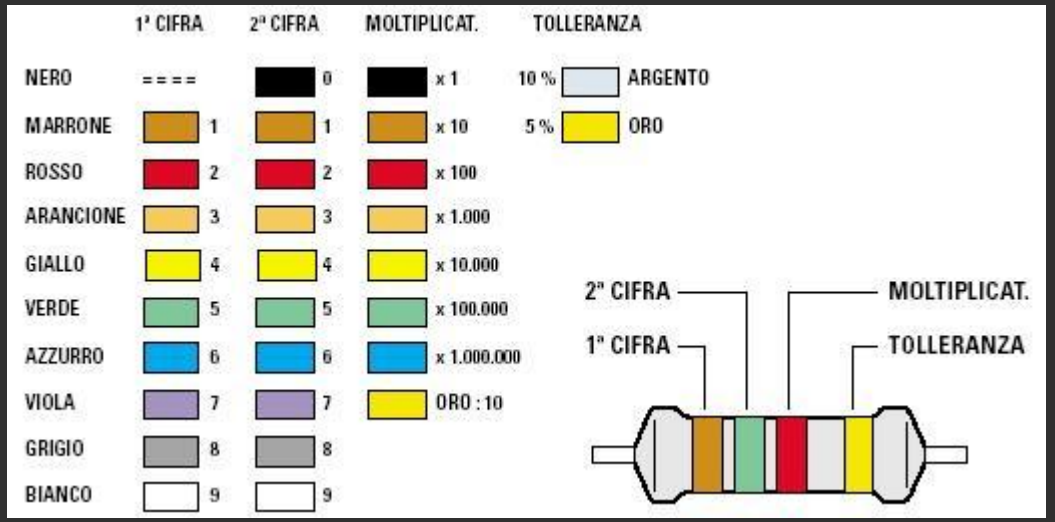
Se i potenziometri non sono indispensabili, è **consigliabile** usare delle resistenze **fisse**, che hanno valori **codificati** e leggibile tramite il **codice colori**

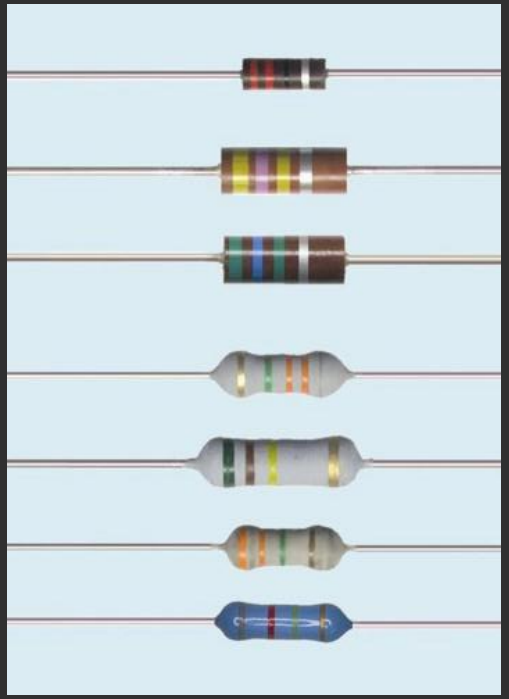


Resistenze

Le resistenze variabili sono **comode**, ma devo regolarle **tutte a mano** e possono essere **accidentalmente modificate** durante il funzionamento causando **errori**

Se i potenziometri non sono indispensabili, è **consigliabile** usare delle resistenze **fisse**, che hanno valori **codificati** e leggibile tramite il **codice colori**





Pulsanti

Funzionamento dei pulsanti

Un **pulsante connette** o **disconnette** due nodi.

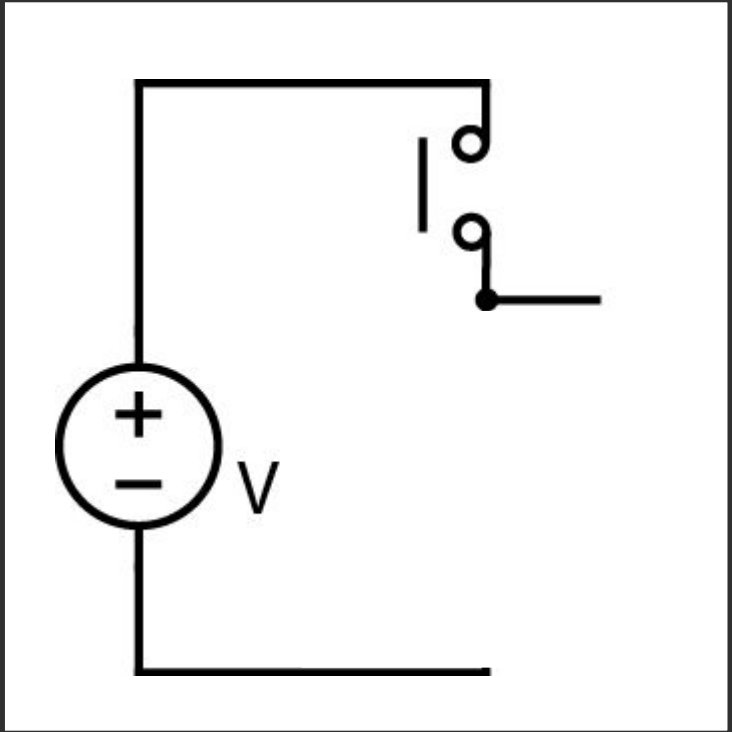
Devo **scegliere** le tensioni dei due nodi affinché il **nodo di interesse** vada

ALTO, HIGH, 1, quando il pulsante è **premuto**

Oppure

BASSO, LOW, 0, quando il pulsante è **rilasciato**

Funzionamento dei pulsanti



Un **pulsante connette** o **disconnette** due nodi.

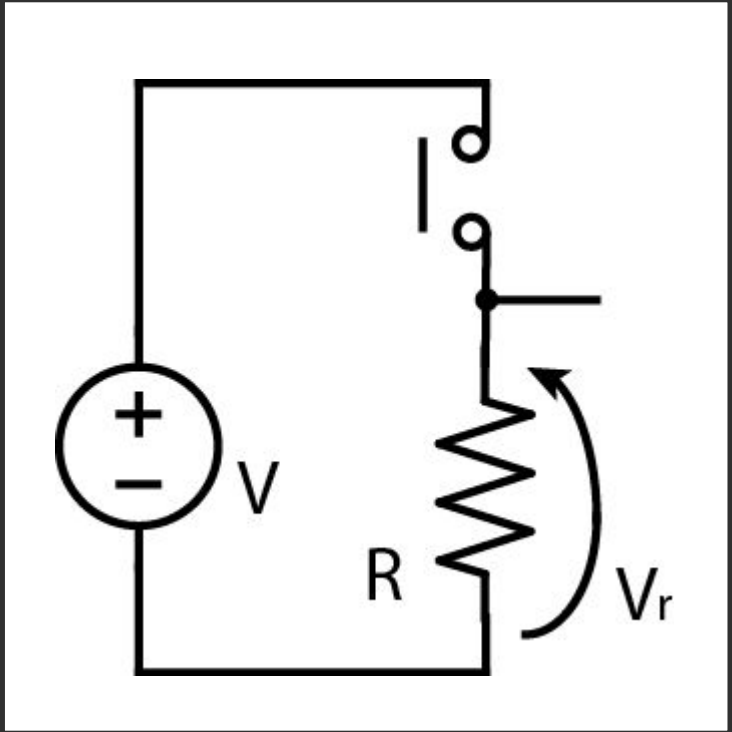
Devo **scegliere** le tensioni dei due nodi affinché il **nodo di interesse** vada

ALTO, HIGH, 1, quando il pulsante è **premuto**

Oppure

BASSO, LOW, 0, quando il pulsante è **rilasciato**

Funzionamento dei pulsanti



Un **pulsante connette** o **disconnette** due nodi.

Devo **scegliere** le tensioni dei due nodi affinché il **nodo di interesse** vada

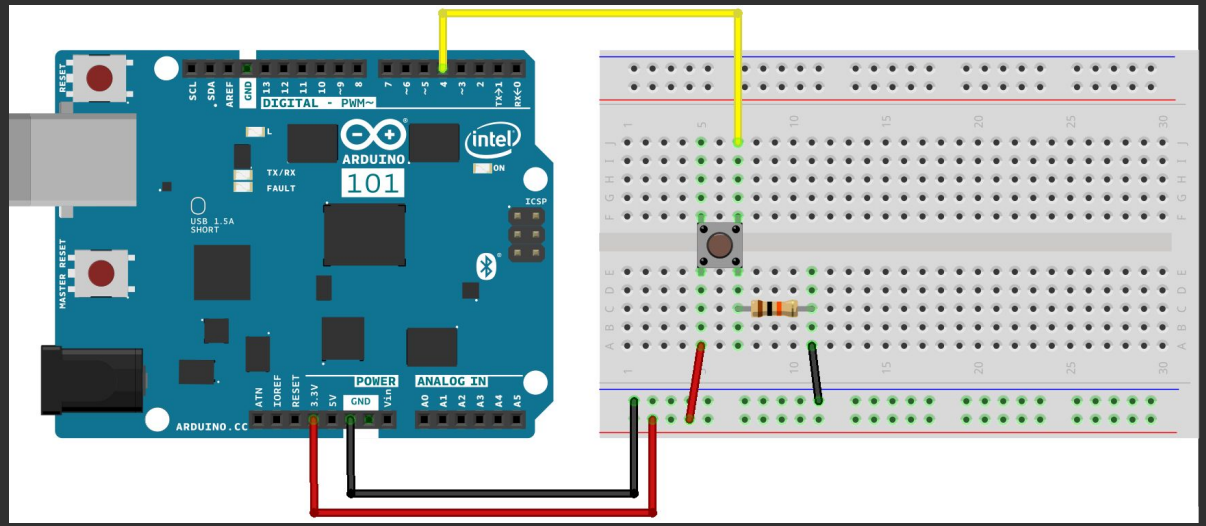
ALTO, HIGH, 1, quando il pulsante è **premuto**

Oppure

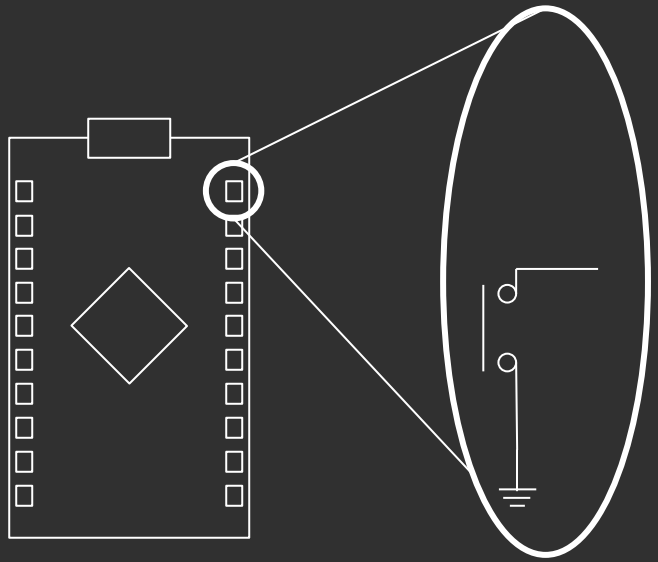
BASSO, LOW, 0, quando il pulsante è **rilasciato**

Pulsante

Un pulsante è come una interruzione su un filo. Quando premi il pulsante permetti alla corrente di scorrere nel circuito.
Un pin di ingresso di Arduino deve avere un resistore che fissi la tensione bassa o alta (LOW o HOGH) altrimenti il valore di ingresso è flottante. Questa configurazione è chiamata pull-down (o pull-up).



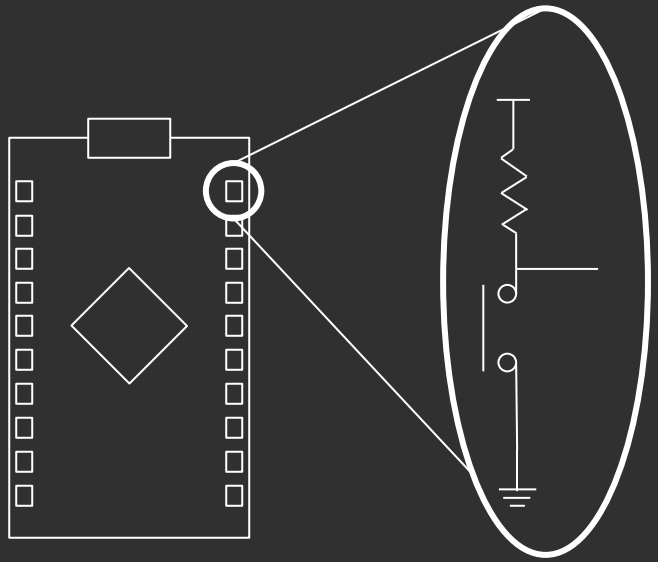
Uscite open collector



Un’uscita **open collector** è come un pulsante collegato **tra 0 V e nient’altro**

Un’uscita open collector può **solo portare bassa** l’uscita

Uscite open collector



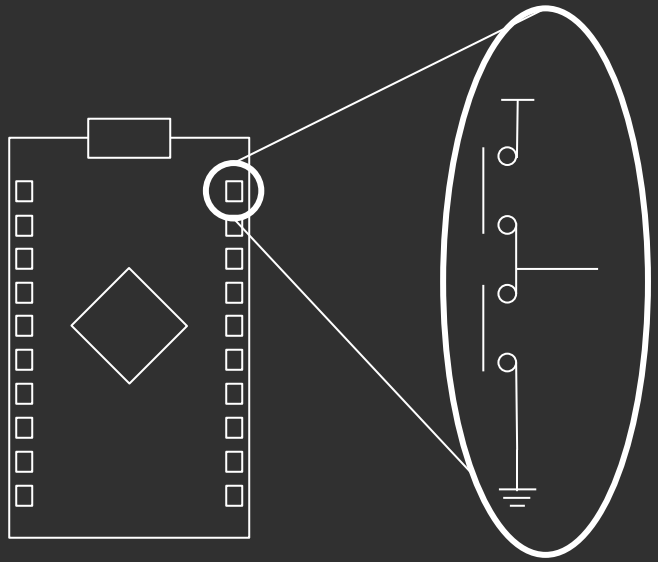
Un’uscita **open collector** è come un pulsante collegato **tra 0 V e nient’altro**

Un’uscita open collector può **solo portare bassa** l’uscita

Un’uscita open collector **necessita** una **resistenza** verso la **tensione alta.** Questa resistenza è chiamata **pull-up**

Quando il pulsante è **aperto**, la resistenza porta **alta** l’uscita

Uscite push-pull



Un’uscita **push-pull** è come due pulsanti in serie, uno collegato **a 0 V** e l’altro alla **tensione alta**

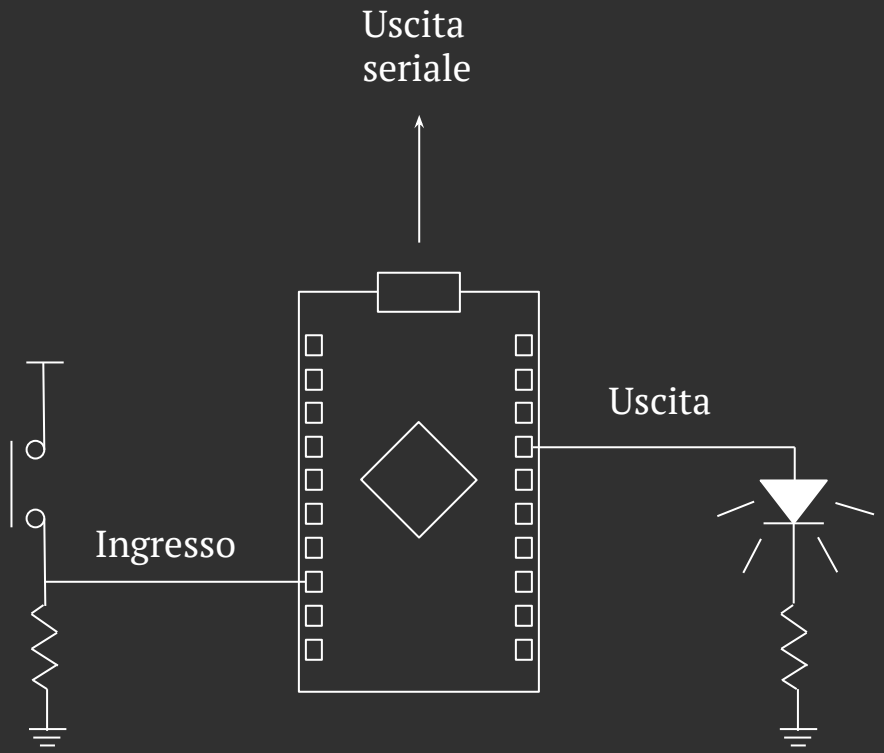
Un’uscita push-pull può **portare** l’uscita **alta** o **bassa**

Un’uscita push-pull **non necessita** di una **resistenza** di pull-up

Esempio: button (digital INPUT)

```
/*Arduino ha un led integrato connesso al pin 13
Dichiariamolo!*/
int butt= 4;

// la routine di setup routine è eseguita una volta allo start up:
void setup() {
      pinMode(butt, INPUT); // inizializziamo il pin digitale come un input
      Serial.begin(9600);   // inizializziamo la comunicazione seriale
}

// dopo il setup, la routine di loop è eseguita continuamente:
void loop() {

      bool buttVal = digitalRead(butt);   // legge il valore del pin e lo scrive in
                                             Una variabile . un boolean è una
                                             variabile che può assumere due valori
                                             1 (HIGH, true) o 0 (LOW, false)
      Serial.println(buttVal);  // scrive sulla seriale
      delay(100);               // aspetta 100 millisecondi
}
```

Esercizio 1: accendere un led con un pulsante



Leggiamo un pulsante

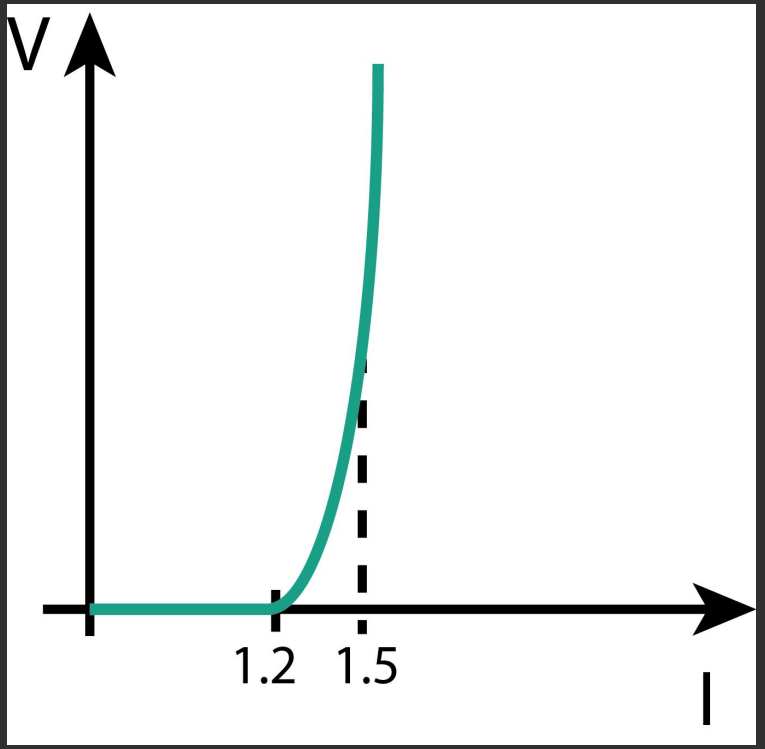
Accendiamo un LED se il pulsante è premuto, altrimenti lo **spegniamo**

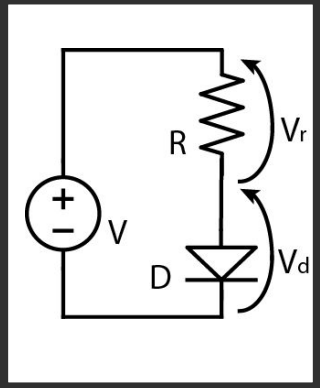
Scriviamo su **seriale** il valore del pulsante e del LED e **monitoriamolo** da terminale.

Exercise 1: light up the LED with a button

```
int butt= 4;
int led = 7;

void setup() {

      pinMode(led, OUTPUT);
      pinMode(butt, INPUT_PULLUP); // Si può usare un resistore interno di Arduino
                                      come pull-up
}

void loop() {

      bool buttVal = digitalRead(butt);
      digitalWrite(led, buttVal);
      delay(100);

}
```

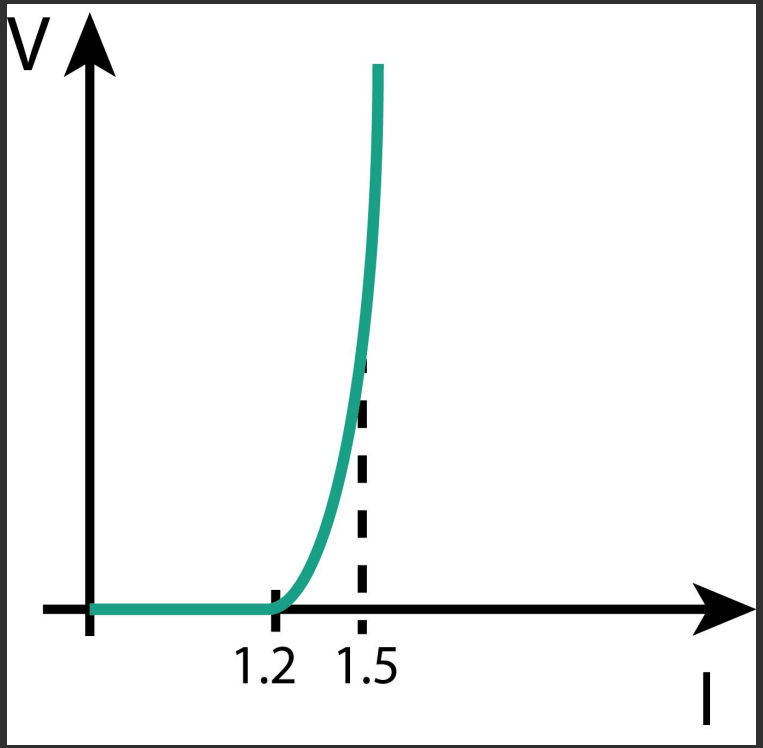
Diodo

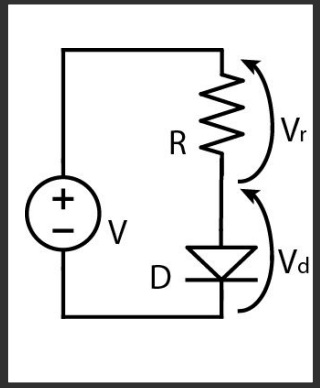




In un **diodo** il rapporto tensione corrente non è lineare.
Esiste una **tensione di soglia** (1.5 V) che discrimina lo **stato del diodo**: **sotto** la tensione di soglia il diodo è **spento**, **alla** tensione di soglia il diodo è **acceso** e può portare tutta la corrente che serve

Diodo



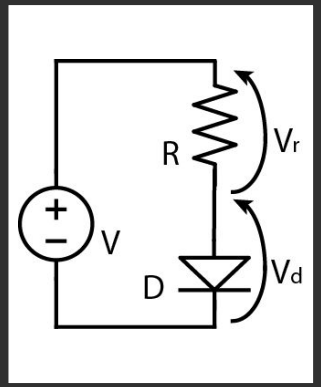


In un **diodo** il rapporto tensione corrente non è lineare.
Esiste una **tensione di soglia** (0.7 V) che discrimina lo **stato del diodo**: **sotto** la tensione di soglia il diodo è **spento**, **alla** tensione di soglia il diodo è **acceso** e può portare tutta la corrente che serve

Questa relazione richiede che un diodo abbia **sempre una resistenza in serie** per evitare che si **bruci!**

Diodo e Resistenze

Come scegliere la **resistenza adatta**?



$$R = \frac{(V - V_d)}{I}$$

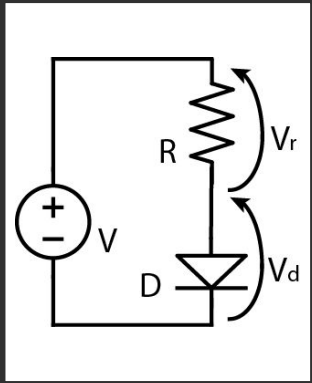
Di solito
Vd = 1.5V
I = tra 5 e 20 mA
V = tra 3.3 e 5 V

$$R_{MAX} = \frac{(5 - 1.5)}{5m} = 700 \text{ Ohm}$$

$$R_{min} = \frac{(3.3 - 1.5)}{20m} = 90 \text{ Ohm}$$

Diodo e Resistenze

L'**intensità luminosa** del diodo **LED** dipende dalla **corrente** che lo attraversa. Per poterla **variare** posso usare una **resistenza variabile**, chiamata **potenziometro**



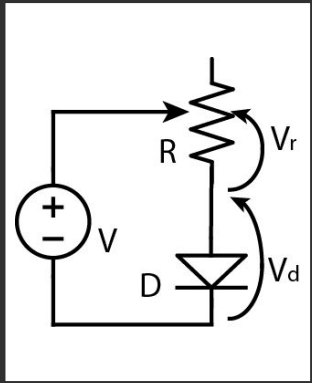
Un **potenziometro** è una resistenza con tre terminali di cui il **centrale** è **regolabile**. Visto che la corrente nel led (ovvero l'intensità) dipende solo dalla resistenza,

se vario la resistenza vario la luce

$$R = \frac{(V - V_d)}{I}$$

Diodo e Resistenze

L'**intensità luminosa** del diodo **LED** dipende dalla **corrente** che lo attraversa. Per poterla **variare** posso usare una **resistenza variabile**, chiamata **potenziometro**



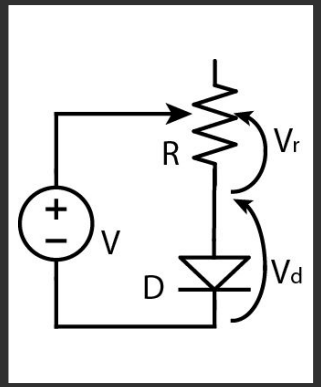
Un **potenziometro** è una resistenza con tre terminali di cui il **centrale** è **regolabile**. Visto che la corrente nel led (ovvero l'intensità) dipende solo dalla resistenza,

se vario la resistenza vario la luce

$$R = \frac{(V - V_d)}{I}$$

Diodo e Resistenze

L'**intensità luminosa** del diodo **LED** dipende dalla **corrente** che lo attraversa. Per poterla **variare** posso usare una **resistenza variabile**, chiamata **potenziometro**



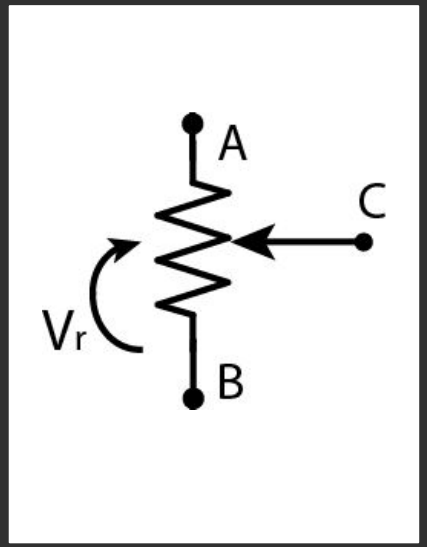
Un **potenziometro** è una resistenza con tre terminali di cui il **centrale** è **regolabile**. Visto che la corrente nel led (ovvero l'intensità) dipende solo dalla resistenza,

se vario la resistenza vario la luce

$$R = \frac{(V - V_d)}{I}$$

Verifichiamolo con un piccolo esempio pratico

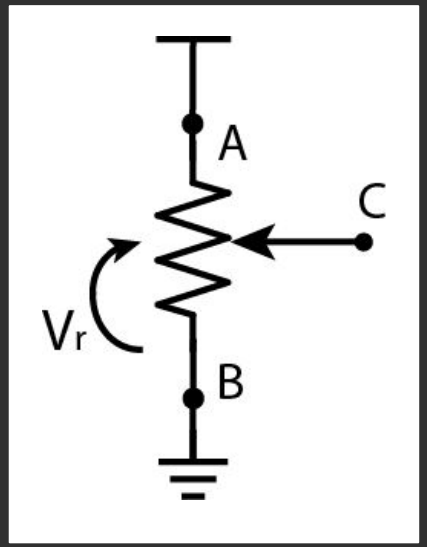
Potenziometro



Uno **slider** è un potenziometro, ossia una **resistenza variabile**

Ha **3 pin**, due fissi e uno mobile

Potenziometro



Uno **slider** è un potenziometro, ossia una **resistenza variabile**

Ha **3 pin**, due fissi e uno mobile

Per leggere la tensione con il Arduino devo:

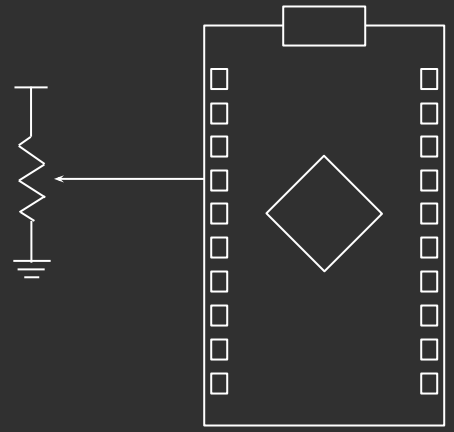
Connettere **A e B** alle tensioni **alta e bassa**

Connettere **C** ad un **pin analogico** del Arduino

Ingressi analogici: ADC

Un segnale **analogico** viene convertito in **digitale** da un **ADC**
Analog to **D**igital **C**onverter

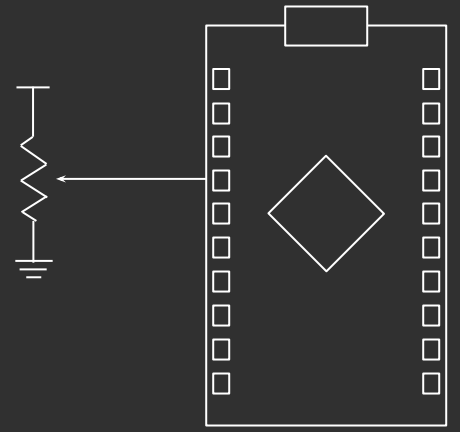
La conversione è a **10 bit**:
il valore della resistenza viene convertita in 2^{10} intervalli (**da 0 a 1023**)



Ingressi analogici: ADC

Un segnale **analogico** viene convertito in **digitale** da un **ADC**
Analog to **D**igital **C**onverter

La conversione è a **10 bit**:
il valore della resistenza viene convertita in 2^{10} intervalli (**da 0 a 1023**)



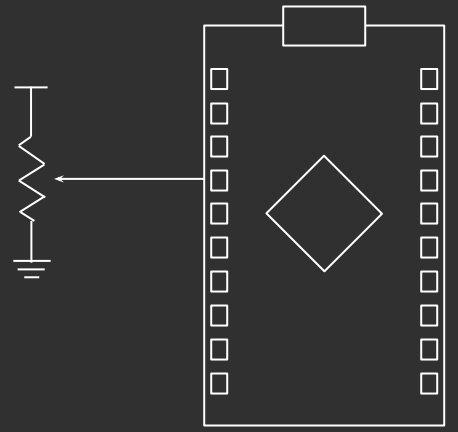
Un byte = **8 bit**: si può adattare con la funzione **map()**

valueOut = **map(**valueIn, fromLow, fromHigh, toLow, toHigh**)**

Ingressi analogici: ADC

Un segnale **analogico** viene convertito in **digitale** da un **ADC**
Analog to **D**igital **C**onverter

La conversione è a **10 bit**:
il valore della resistenza viene convertita in 2^{10} intervalli (**da 0 a 1023**)



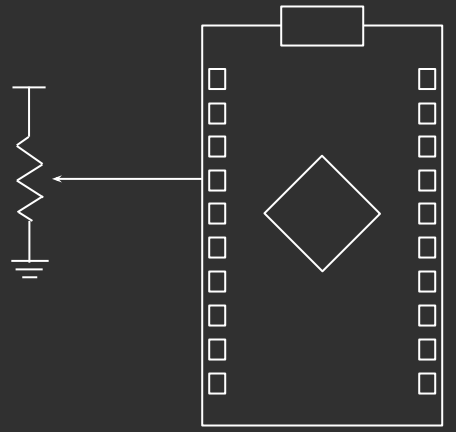
Un byte = **8 bit**: si può adattare con la funzione **map()**

valueOut = **map(**valueIn, fromLow, fromHigh, toLow, toHigh**)**

resByte = **map(**valueIn, 0, 1023, 0, 255**)**

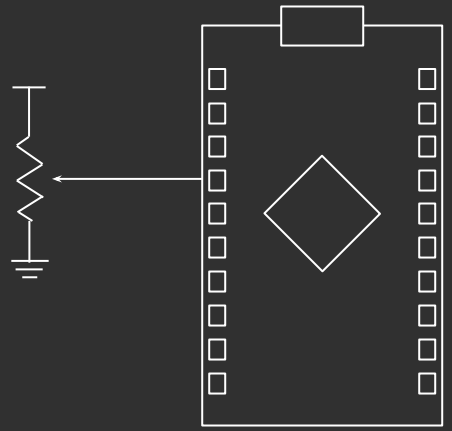
Ingressi analogici: ADC

```
int analogPinIn = A0;
int valueIn, resByte;


void setup() {
      pinMode(analogPinIn, INPUT);
      Serial.begin(9600);
}

void loop() {
      valueIn = analogRead(analogPinIn);
      resByte = map(valueIn, 0, 1023, 0, 255);



      Serial.print("La conversione vale ");
      Serial.println(resByte);

      delay(100);
}
```



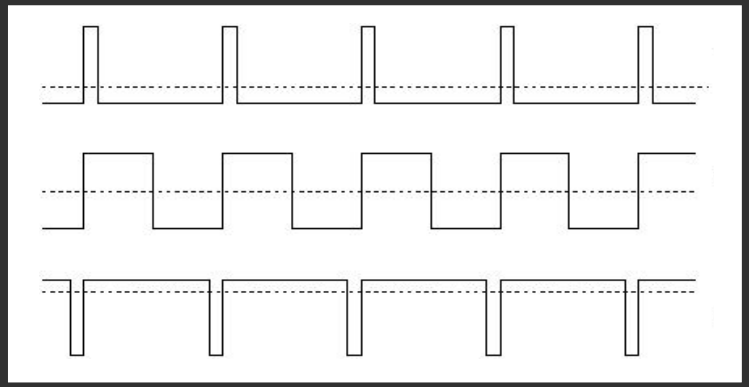
Ingressi analogici: ADC

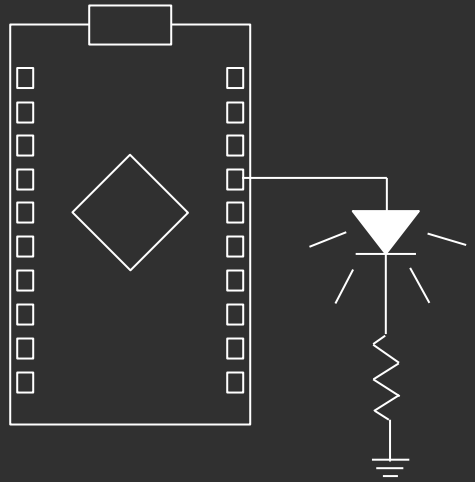
```
#define RMAX 1000;
int analogPinIn = A0;
int valueIn, resByte;
float resFloat;

void setup() {
      pinMode(analogPinIn, INPUT);
      Serial.begin(9600);
}

void loop() {
      valueIn = analogRead(analogPinIn);
      resByte = map(valueIn, 0, 1023, 0, 255);

      resFloat = resByte / 255 * RMAX;

      Serial.print("La resistenza vale ");
      Serial.println(resFloat);

      delay(100);
}
```



Uscita analogica: PWM

Un segnale **digitale** può essere convertito in **analogico** tramite un segnale **PWM**
Pulse **W**idth **M**odulation

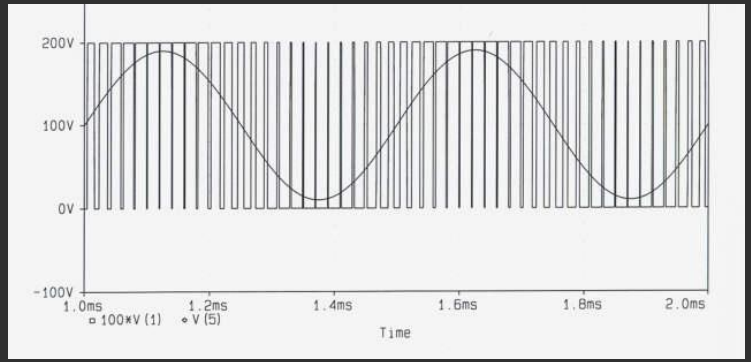


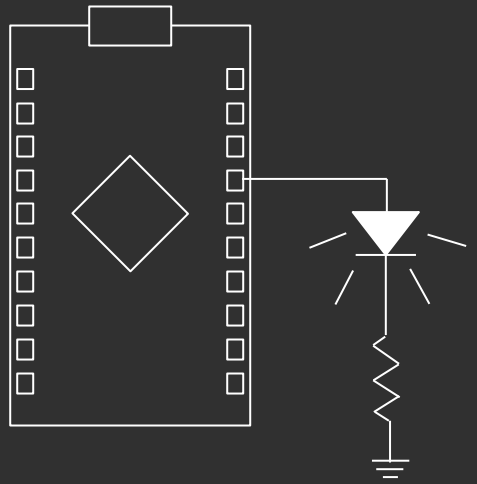


Mediamente il segnale è a livello della **linea tratteggiata**

Uscita analogica: PWM

Un segnale **digitale** può essere convertito in **analogico** tramite un segnale **PWM**
Pulse **W**idth **M**odulation

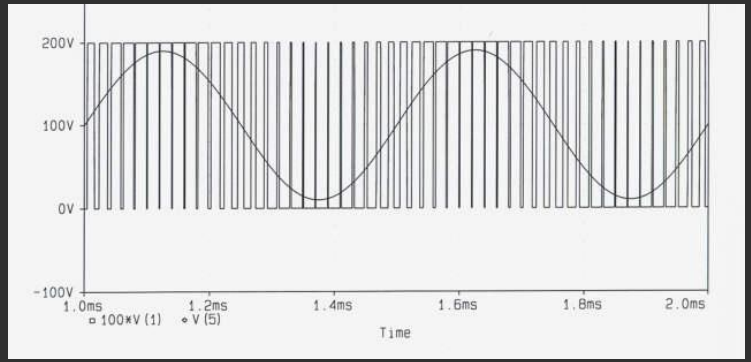


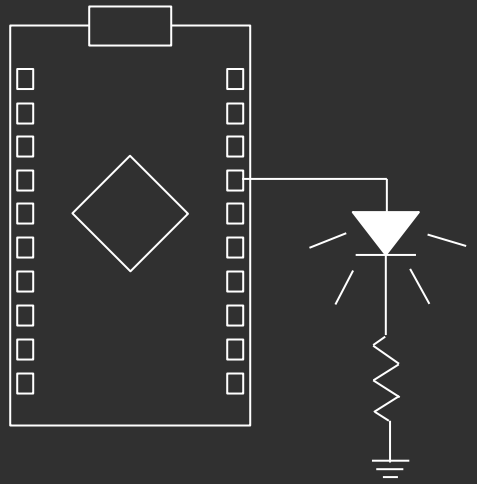


Se gli impulsi **variano** nel tempo ottengo il **segnale desiderato**

Uscita analogica: PWM

Un segnale **digitale** può essere convertito in **analogico** tramite un segnale **PWM**
Pulse **W**idth **M**odulation

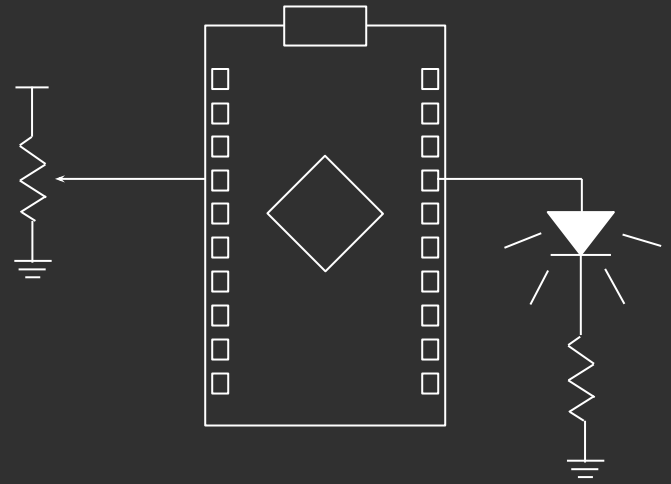




Se gli impulsi **variano** nel tempo ottengo il **segnale desiderato**

analogWrite(pin, value**)** value da **0** a **255**

Esercizio 2: accendere un led con un potenziometro



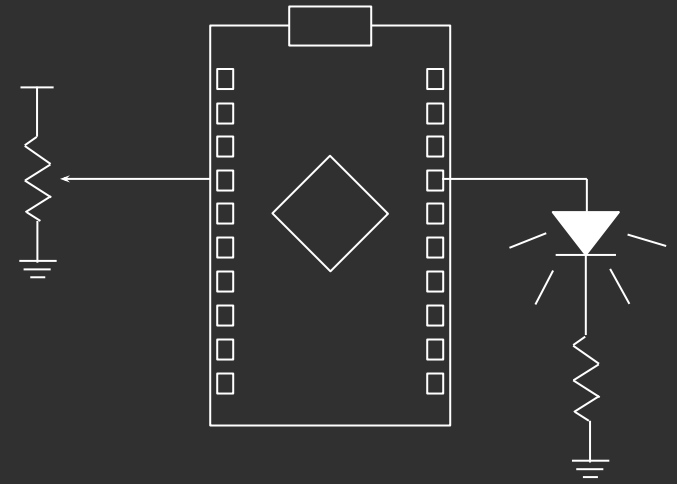
Leggi un potenziometro

Illumina il LED dipendentemente dal valore del potenziometro

Scrivi sulla **seriale** il valore del LED

Esercizio 2: accendere un led con un potenziometro

```
int analogPinIn = A0;
int valueIn, resByte;
int analogPinOut = 11;

void setup() {
      pinMode(analogPinIn, INPUT);
      pinMode(analogPinOut, OUTPUT);
}

void loop() {
      valueIn = analogRead(analogPinIn);
      resByte = map(valueIn, 0, 1023, 0, 255);

      analogWrite(analogPinOut, resByte);

      delay(10);
}
```



Le strutture di controllo

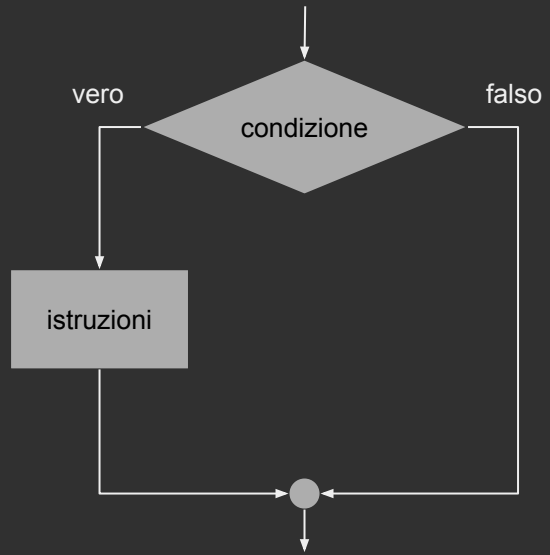
Cosa sono le strutture di controllo

Comandi speciali che servono a specificare **se**, **quando**, **in quale ordine** e **quante volte** devono essere eseguite le istruzioni

Si dividono in strutture **alternative** e strutture **iterative**

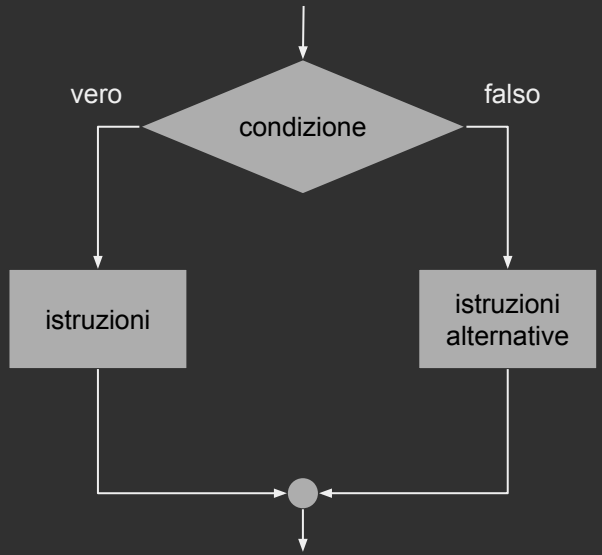
If (alternativo)

```
String messaggio = “ciao mondo!”
boolean stampa = false;

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

void loop() {
     if (stampa) {
          Serial.println(messaggio);
     }
     stampa = !stampa;
}
```



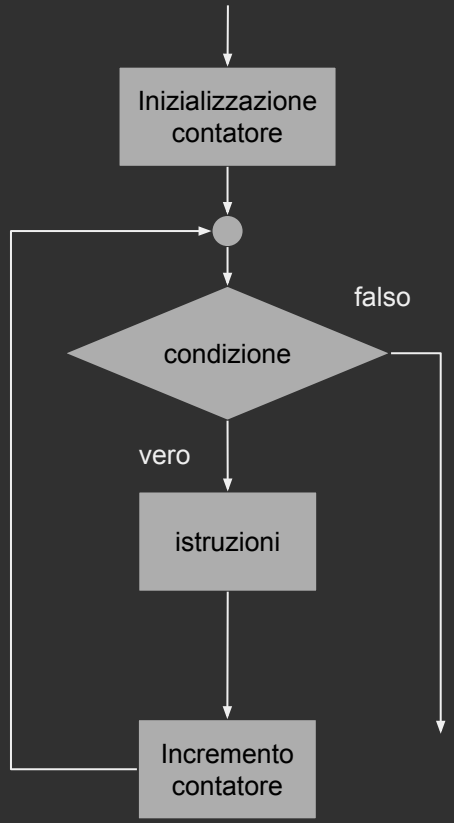
If - else (alternativo)

```
boolean verofalso;
String vf = “”;

void setup() {
     Serial.begin(9600);
     verofalso = true;
}

void loop() {
     if (verofalso) {
          vf= “true”;
     }
     else {
          vf = “false”;
     }
     verofalso = !verofalso;
     Serial.print(“Vero o Falso? “);
     Serial.println(vf);
     delay(150);
}
```



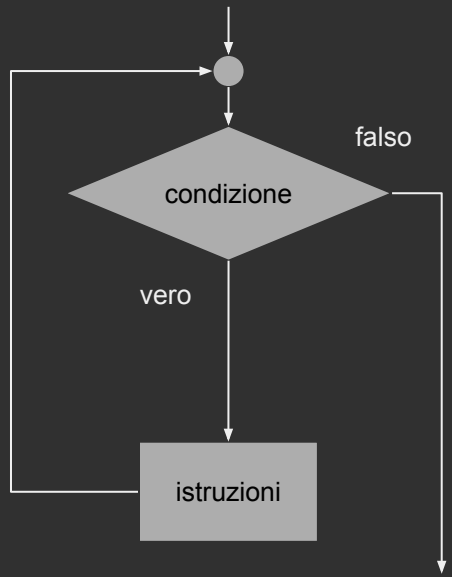
For (iterativo)

```
int counter = 0;

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

void loop() {
    // for(inizializzazione;condizione;incremento)
    for (int i=0; i <= 9; i++) {
        Serial.print(counter);
        Serial.println(i);
    }
    delay(1000);
    counter++;
}
```



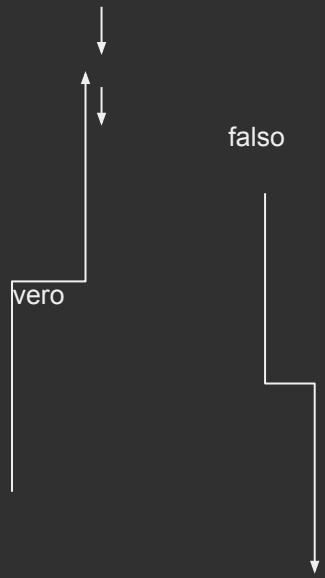
While (iterativo)

```
int counter = 0;
int i;

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

void loop() {
     i = 0;
     while(i <= 9) {
          Serial.print(counter);
          Serial.println(i);
          i++; //necessario per uscire dal while
     }
     delay(1500);
     counter++;
}
```



Switch case

```
#define ZERO 0
#define UNO 1
#defina DUE 2
int stato = 0;

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

void loop() {
     i = 0;
     switch(stato) {
          case ZERO:
               //istruzioni
               stato = 1;
               Break;
          case UNO:
               stato = 2;
               Break;
          case DUE:
               stato = 0;
               break;
```



Differenza tra for e while

Il loop **'for'** è usato quando conosciamo il numero di iterazioni a priori

Il loop **'while'** è usato quando il numero di iterazioni non è conosciuto ed il loop dipende da una condizione.

Le strutture di controllo si possono annidare

Operatori di comparazione

Assumiamo che la variabile **A** valga 10 e la variabile **B** valga 20:

`==`	**uguale**	Controlla se il valore dei due operandi è uguale o meno, si si la condizione è vera (true). *(A == B) non è true*
`!=`	**non uguale**	Controlla se il valore dei due operandi è diverso o meno, si si la condizione è falsa (false). *(A != B) è true*
`<`	**minore**	Controlla se il valore dell'operando di sinistra è minore del valore dell'operando di destra, se si la condizione è vera *(A < B) è true*
`>`	**maggiore**	Controlla se il valore dell'operando di sinistra è maggiore del valore dell'operando di destra, se si la condizione è vera *(A > B) non è true*
`<=`	**minore o uguale**	Controlla se il valore dell'operando di sinistra è minore o uguale del valore dell'operando di destra, se si la condizione è vera *(A <= B) è true*
`>=`	**maggiore o uguale**	Controlla se il valore dell'operando di sinistra è maggiore o uguale del valore dell'operando di destra, se si la condizione è vera *(A >= B) non è true*

Esercizio 3: controlla la variazione di luce di un LED (fade up/down) con un pulsante

Esercizio 3:

```
int buttonPin = 0;
bool buttonValue, prevButtValue = false;
int ledPin = 9;

void setup() {
	pinMode(buttonPin, INPUT);
	pinMode(ledPin, OUTPUT);
}

void loop() {
	buttonValue = digitalRead(buttonPin);
	if (buttonValue != prevButtValue ){
		for (int i=0; i <= 255; i++) {
			analogWrite(ledPin,i);
			delay(50);
			prevButtValue = true;
		}
	} else {
		for (int i=255; i <= 0; i++) {
			analogWrite(ledPin,i);
			delay(50);
			prevButtValue = false;

		}
	}
}
```

Esercizio 3:

```
int buttonPin = 1;
bool buttonValue;
int ledPin = 9;
int ledValue = 0;

void setup() {
	pinMode(buttonPin, INPUT);
	pinMode(ledPin, OUTPUT);
}
void loop() {
	buttonValue = digitalRead(buttonPin);
	if buttonValue {
		while (ledValue<255) {
			analogWrite(ledPin,ledValue);
			delay(100);
			ledValue++;
		}
	} else {
		while (ledValue>0) {
			analogWrite(ledPin,ledValue);
			delay(100);
			ledValue--;
		}
	}
}
```

Librerie

Le librerie sono un **insieme di funzioni predefinite** e predisposte per essere collegate al codice principale

Librerie e schede di prototipazione

Le librerie si basano sul **riuso del codice**, evitando di dover riscrivere ogni volta le stesse funzioni

Per le schede di prototipazione sono presenti molte librerie che facilitano l'uso di **sensori ed attuatori** (motori, display,etc.), di **protocolli di comunicazione** (MIDI, OSC), o di **funzioni complesse** (analisi del'audio, gestione dei dati)

Uso delle librerie

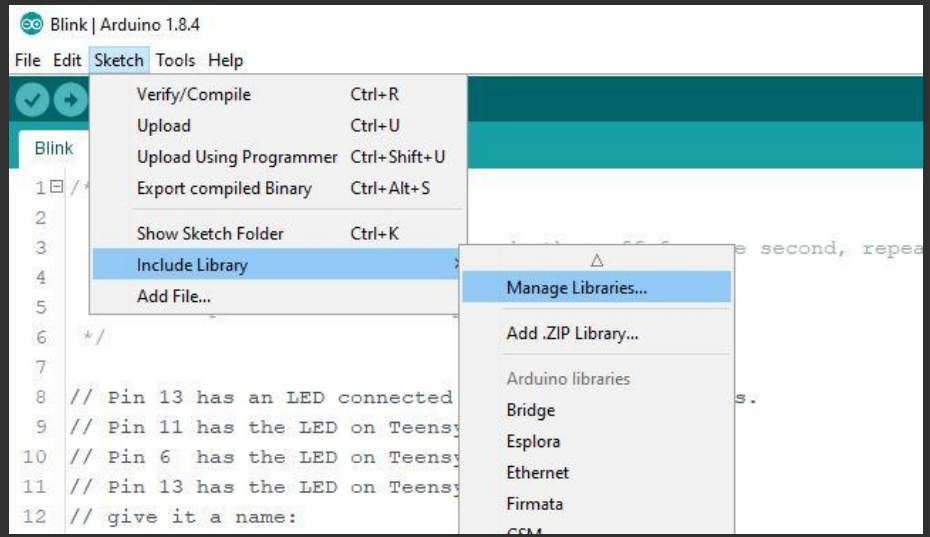
Per usare una libreria si deve innanzitutto dire alla scheda di fare riferimento al suo codice, ovvero si deve **includere** (prima di ogni altra istruzione)

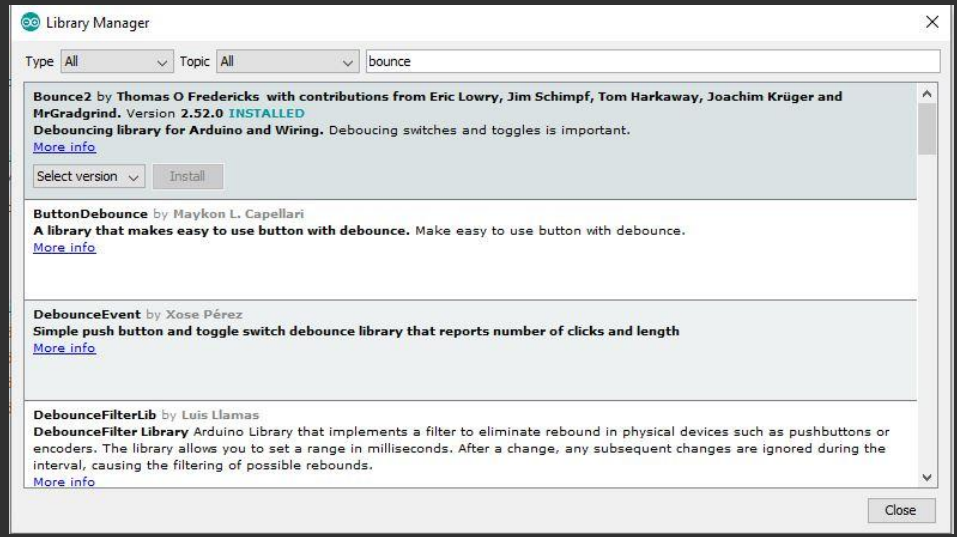
```
#include <NOMEFILE>
```

Ogni libreria è costituita da proprie **funzioni** che possono essere usate passandogli i corretti **argomenti** e ottenendo un **risultato**

Esempio: Bounce library

Per includere una libreria bisogna scaricarla nella corretta directory. Uno strumento util è il **Library Manager**





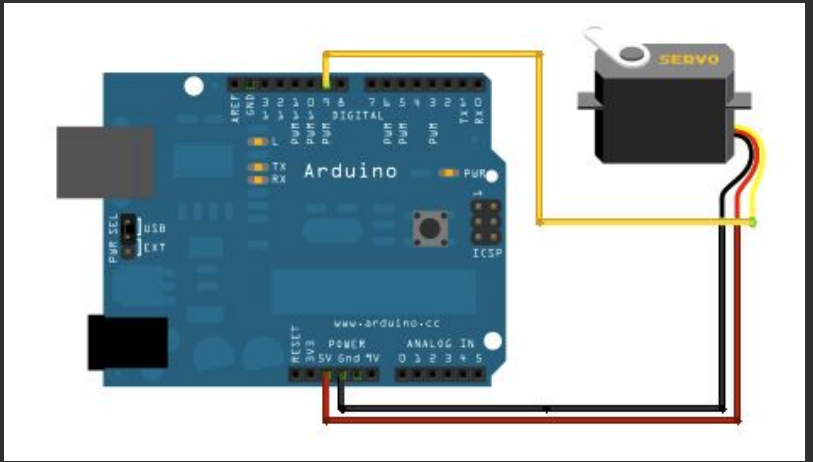
Servo e servo library

Un **servo** un servo è un motore del quale si può controllare precisamente la **posizione** attraverso la PWM

Ci sono molti servo che differiscono per **angolo di rotazione**, **velocità** e **coppia torcente** (quanto peso riesce a muovere) ma tutti possono essere controllati da una libreria, **Servo** appunto…

Servo e servo library

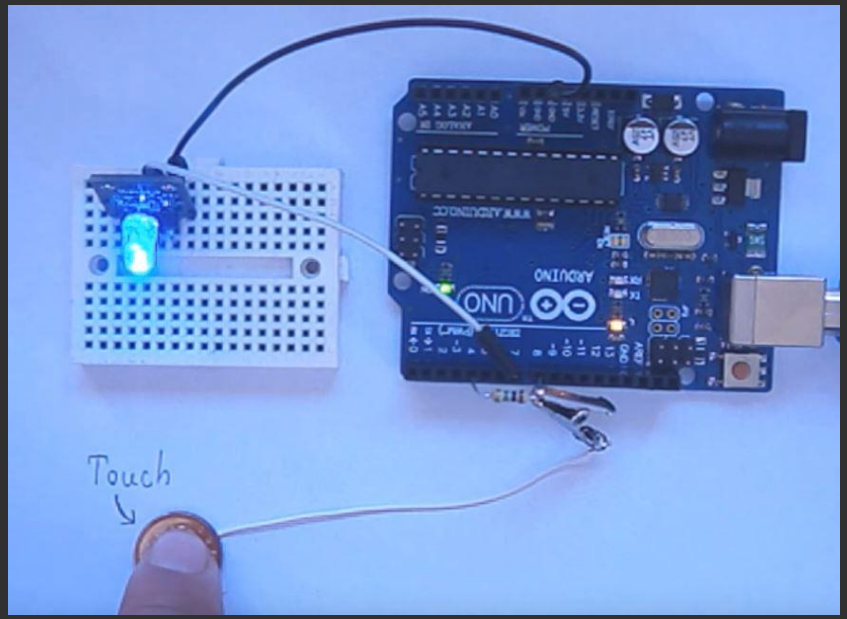
```
#include <Servo.h> // include la libreria
Servo myservo; // inizializza l'oggetto Servo
int pos = 0;
void setup() {
      myservo.attach(9); // connette il servo ad un pin
}
void loop() {
      for(pos = 0;pos < 180;pos +=1) {
            myservo.write(pos); // muove il servo nella
                                      posizione
            delay(15);
      }
      for(pos = 0;pos >= 180;pos -=1) {
            myservo.write(pos);
            delay(15);
      }
      // myservo.detach(); è usato per rilasciare il servo
}
```



Sensore capacitivo

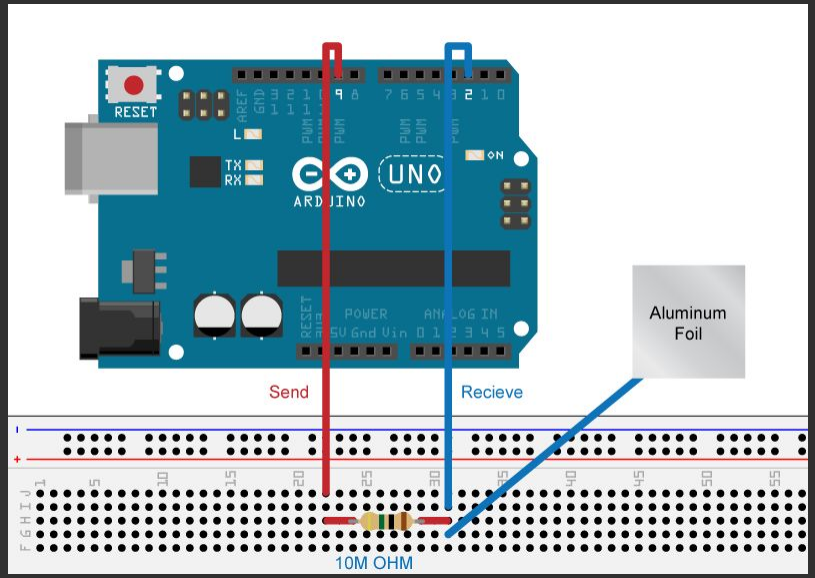
Tutti i materiali conduttivi posso diventare dei sensori con un semplice resistore.
I **sensori capacitivo** funzionano per un principio fisico usando il proprio corpo come un **condensatore**.

Arduino invia un segnale su un pin e legge lo stesso segnale su un altro pin, calcolando la **differenza temporale** tra i due. Mettendo un condensatore tra i due pin (**tu**, trmite un tuo dito), la differenza temporale aumenterà e Arduino la rileverà.



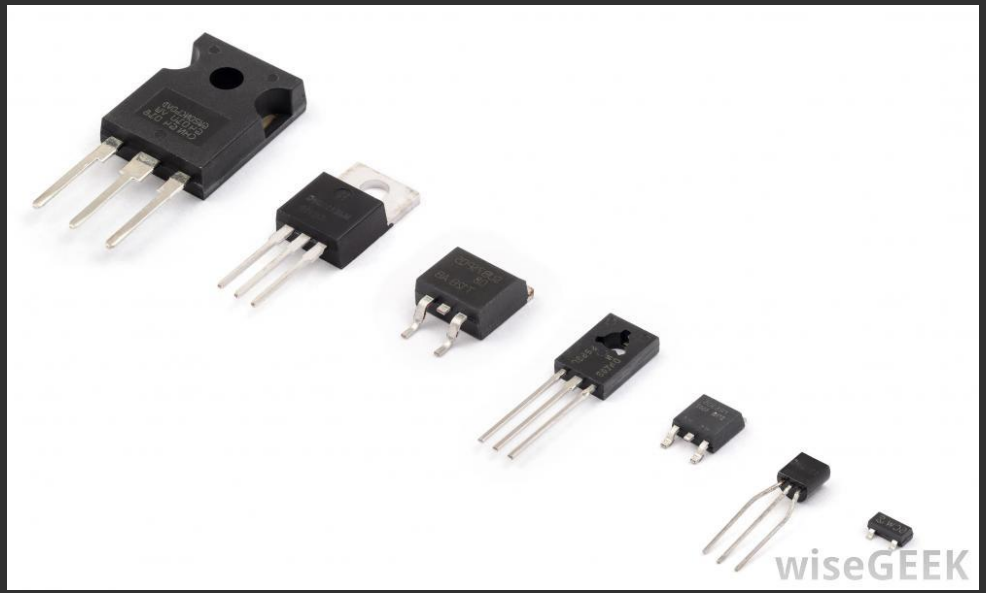
Sensore capacitivo

```
#include <CapacitiveSensor.h>

CapacitiveSensor  touch =  CapacitiveSensor (9,2);
const long touch_Threshold = 1000;
unsigned int ledPin = 13;

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

      pinMode(ledPin,OUTPUT);
      }

void loop() {
      long touch_Value =  touch.capacitiveSensor(30);
      if (touch_Value >= touch_Threshold) {
            digitalWrite(ledPin,1);
      }
      else digitalWrite(ledPin,0);
      Serial.printl(touch_Value );

}
```



Transistor

Transistor

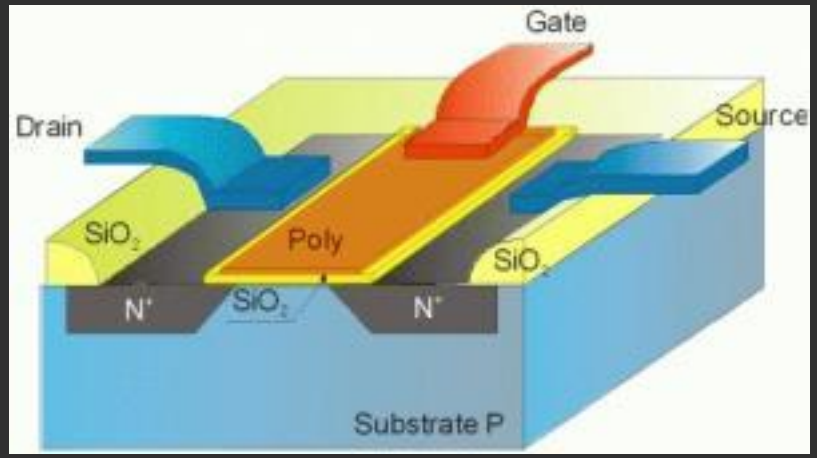


Può fare da **amplificatore** o da **interruttore**

Tre tipi principali:

- BJT
- **MOSFET**
- JFET

Transistor



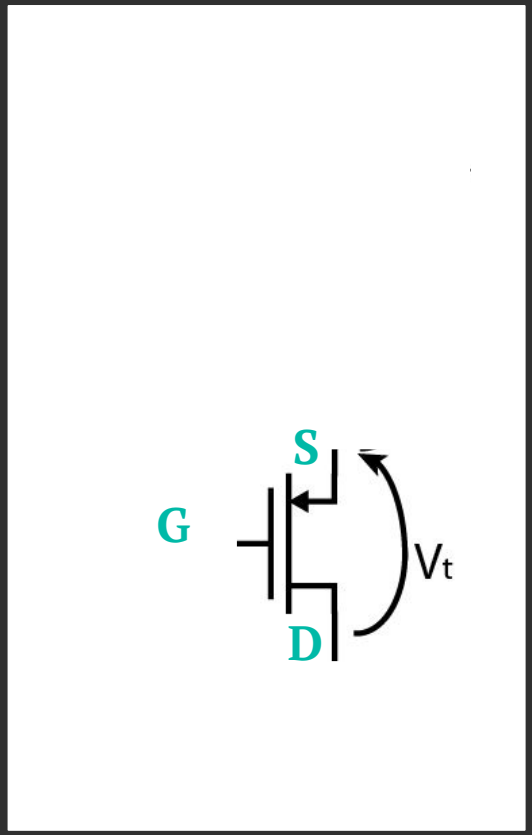
MOSFET

Tre pin:

- Source (**S**): sorgente
- Drain (**D**): pozzo
- Gate (**G**): cancello/controllo

Il **cancello** controlla quanta “**acqua**” (**corrente elettrica**) passa dalla **sorgente** al **pozzo**

Transistor



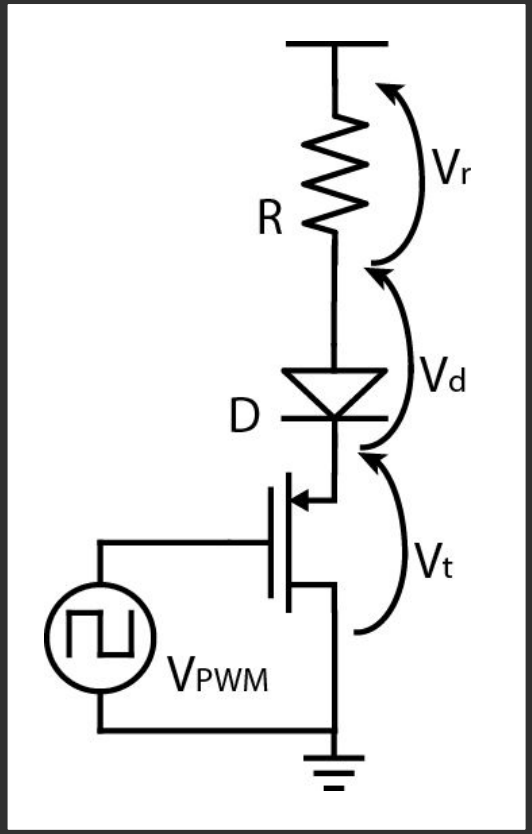
MOSFET

Tre pin:

- Source (**S**): sorgente
- Drain (**D**): pozzo
- Gate (**G**): cancello/controllo

Il **cancello** controlla quanta “**acqua**” (**corrente elettrica**) passa dalla **sorgente** al **pozzo**

Transistor

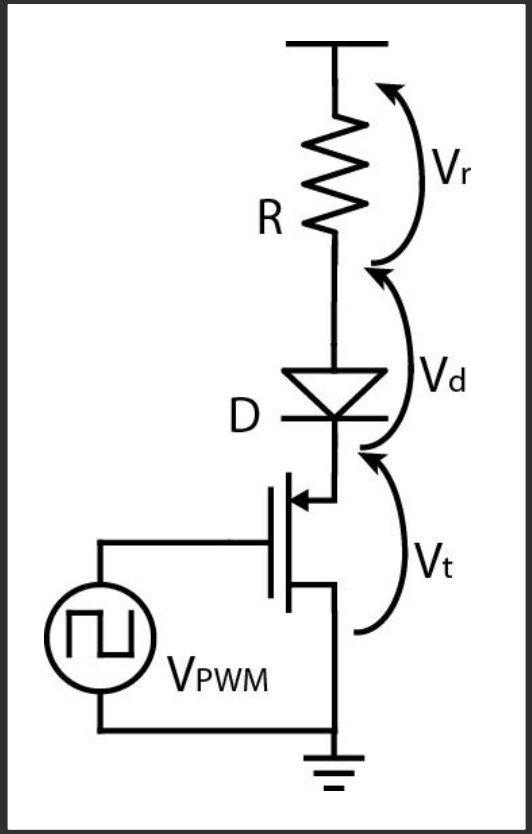


MOSFET come interruttore

Quando il **gate** è **alto** il MOS è spento e **non passa** corrente nel ramo del diodo.

Quando il **gate** è **basso**, il MOS è acceso e **passa** corrente nel ramo del diodo.

Transistor



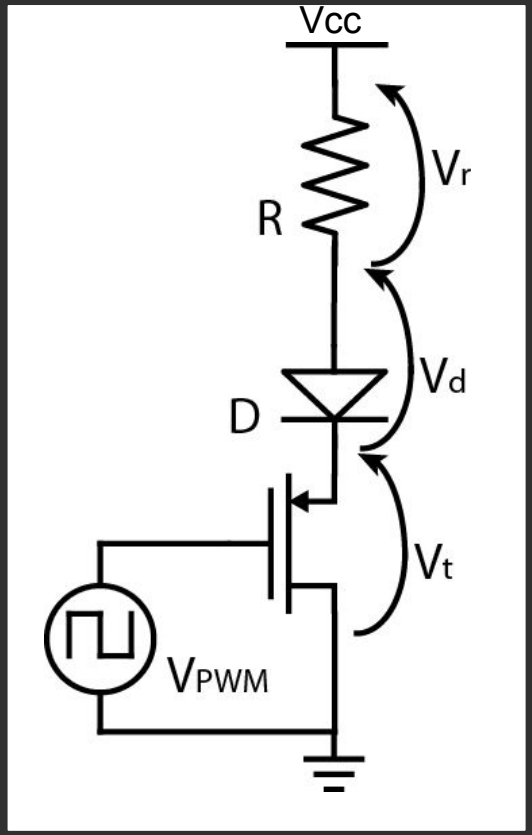
MOSFET come interruttore

Il transistor **trasmette** il **PWM** al ramo di uscita

Il **vantaggio** è che **il diodo prende corrente dall'alimentazione** e non dal generatore PWM (il Photon)

In questo modo può essere più potente, ovvero **può generare molta più luce**

Transistor



Corrente nel MOSFET

Il transistor va dimensionato **correttamente**:

Scelto il diodo (**Id** e **Vd**) e la tensione di alimentazione **Vcc** si ricava la resistenza **R**

$$R = \frac{Vcc - Vd - Vt}{Id}$$

Esempio:

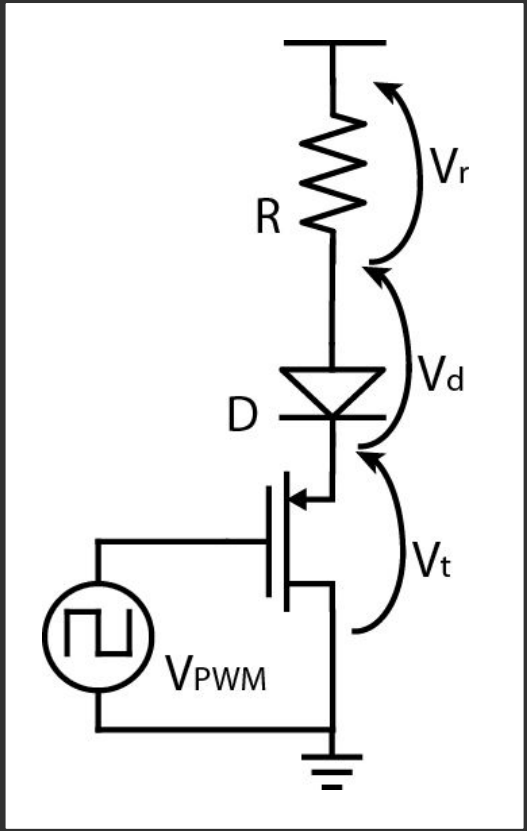
Vcc = 12 V

Id = 100 mA

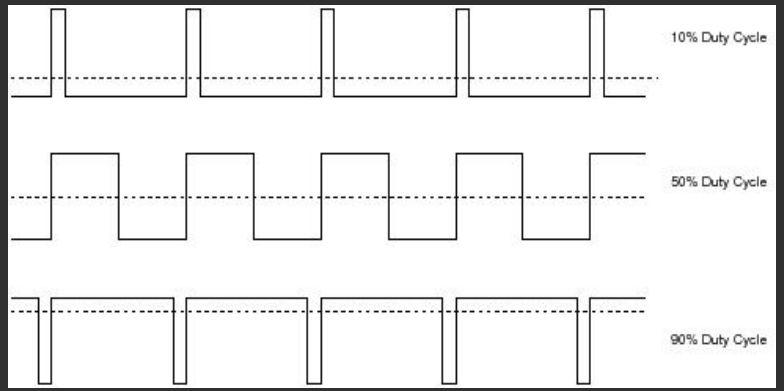
Vd = 3 V

$$R = \frac{12V - 3V - 0V}{100mA} = 90\ Ohm$$

Pulse Width Modulation

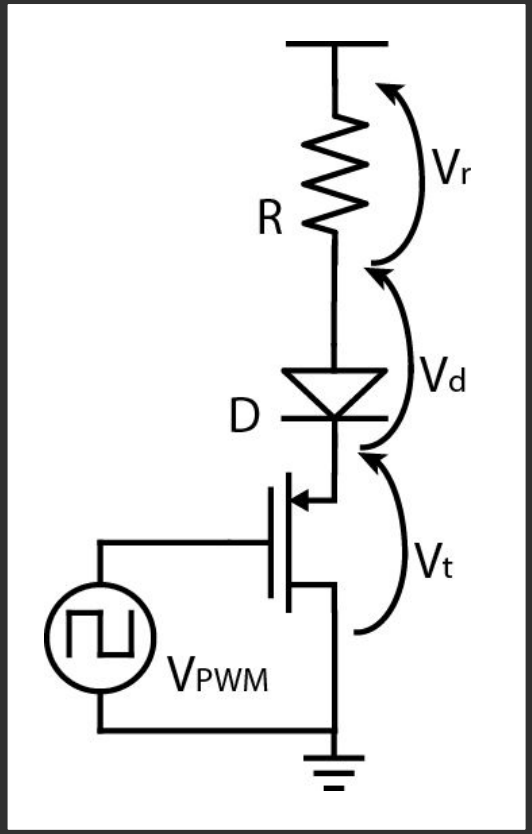


Un segnale **digitale** può essere convertito in **analogico** tramite un segnale **PWM**
Pulse **W**idth **M**odulation

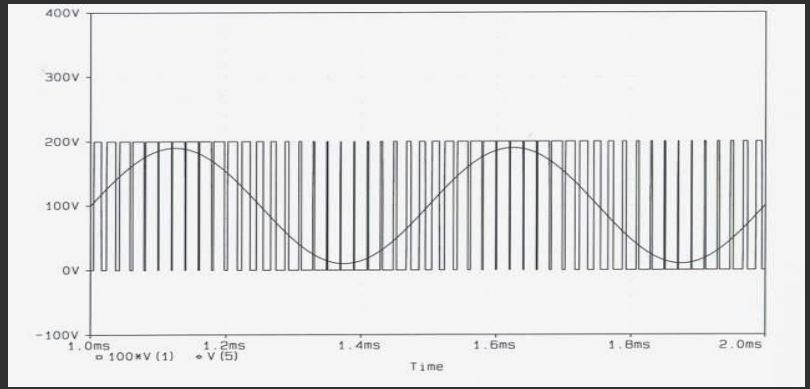


Mediamente il segnale è a livello della **linea tratteggiata**

Pulse Width Modulation

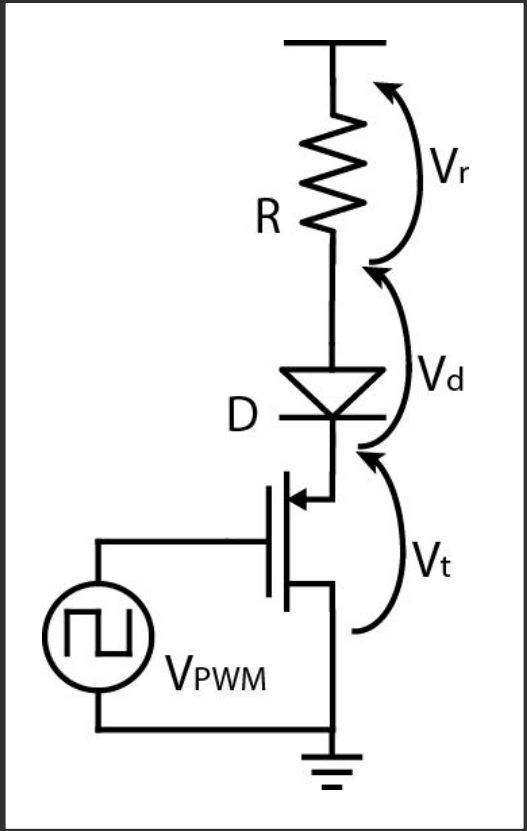


Un segnale **digitale** può essere convertito in **analogico** tramite un segnale **PWM**
Pulse **W**idth **M**odulation

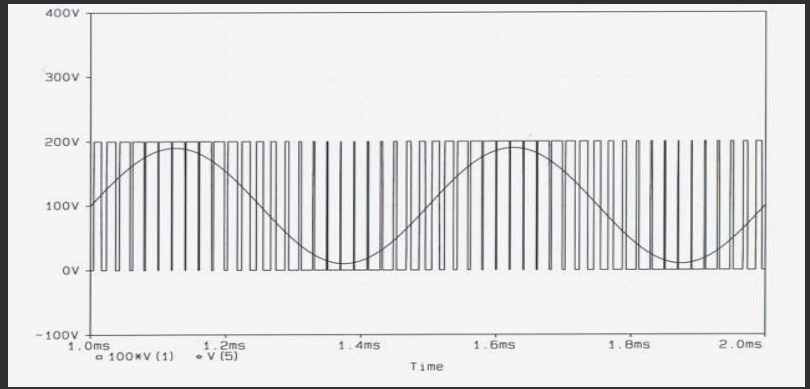


Se gli impulsi **variano** nel tempo ottengo il **segnale desiderato**

Pulse Width Modulation



Un segnale **digitale** può essere convertito in **analogico** tramite un segnale **PWM**
Pulse **W**idth **M**odulation



Se gli impulsi **variano** nel tempo ottengo il **segnale desiderato**

analogWrite(pin, value**)** value da **0** a **255**

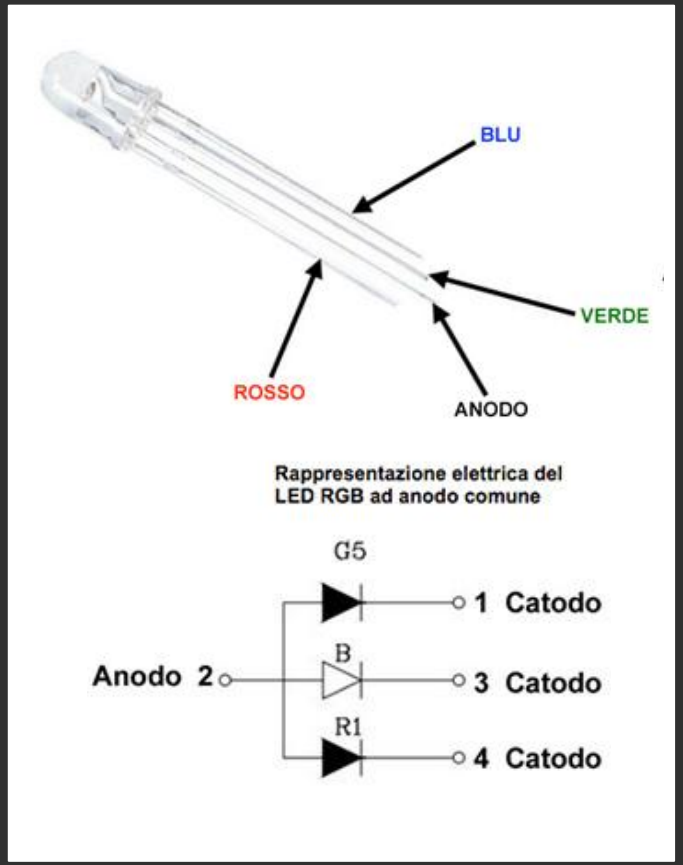
LED RGB

LED

I LED esistono di diversi colori: rosso, verde, blu, arancione, bianco

Esistono LED che integrano più colori al loro interno: i LED RGB

LED RGB

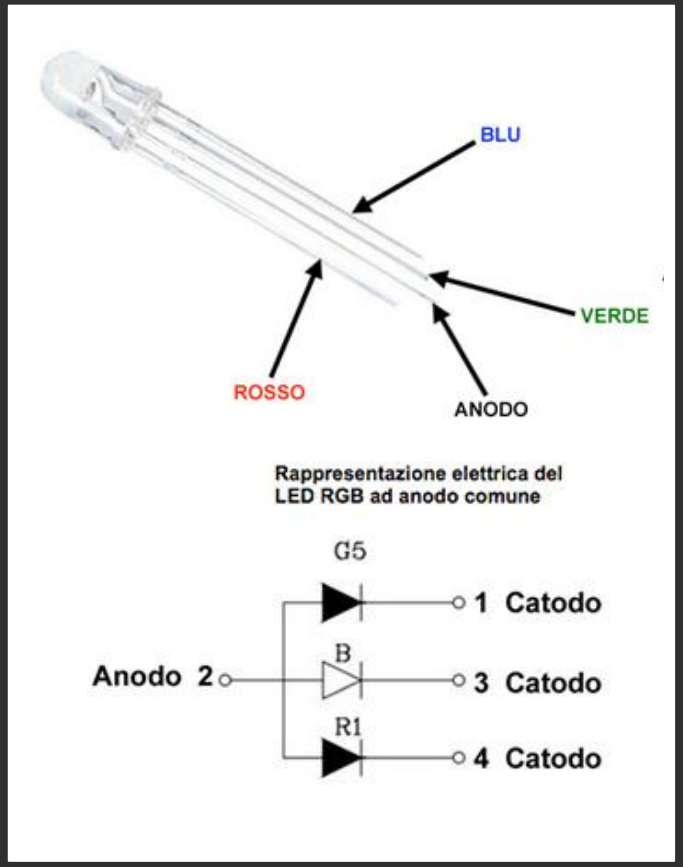


LED

I LED esistono di diversi colori: rosso, verde, blu, arancione, bianco

Esistono LED che integrano più colori al loro interno: i LED RGB

LED RGB



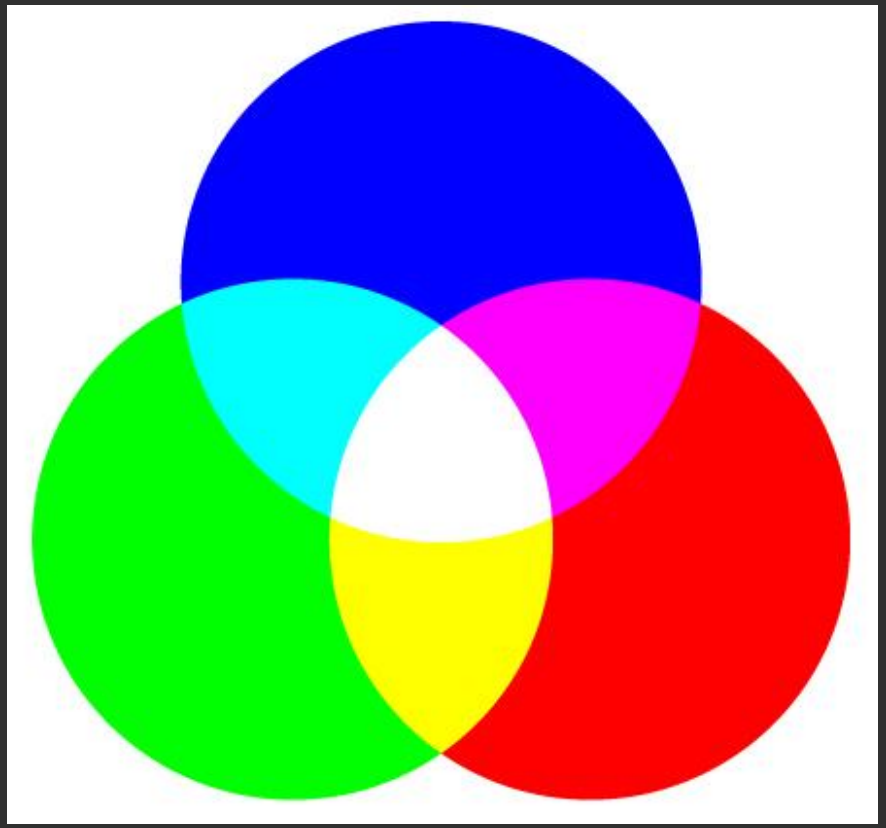
LED

I **LED** esistono di diversi **colori**: **rosso**, **verde**, **blu**, **arancione**, **bianco**

Esistono LED che integrano **più colori** al loro interno: i **LED RGB**

I tre colori sono **pilotati separatamente** e la loro **luce** viene **sommata** insieme generando il **colore desiderato** (codifca RGB)

LED RGB



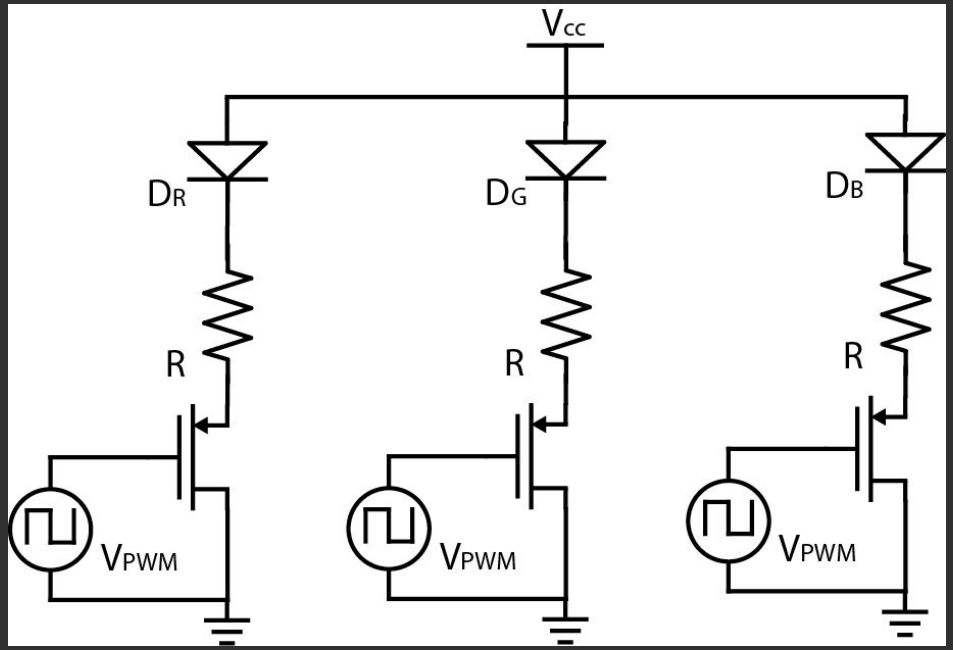
LED

I **LED** esistono di diversi **colori**: **rosso**, **verde**, **blu**, **arancione**, **bianco**

Esistono LED che integrano **più colori** al loro interno: i **LED RGB**

I tre colori sono **pilotati separatamente** e la loro **luce** viene **sommata** insieme generando il **colore desiderato** (codifca RGB)

LED RGB



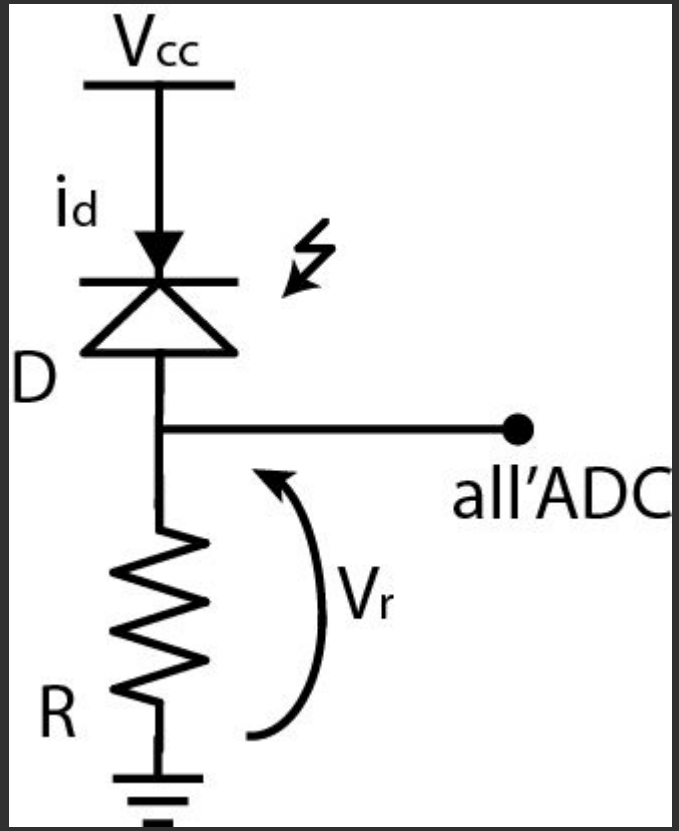
LED RGB

I **LED RGB** si pilotano ciascuno come un **LED discreto**

Ciascun colore ha il suo ramo con **transistor** e **resistenza**. Il **dimensionamento** segue la formula di **prima**

$$R = \frac{Vcc - Vd - Vt}{Id}$$

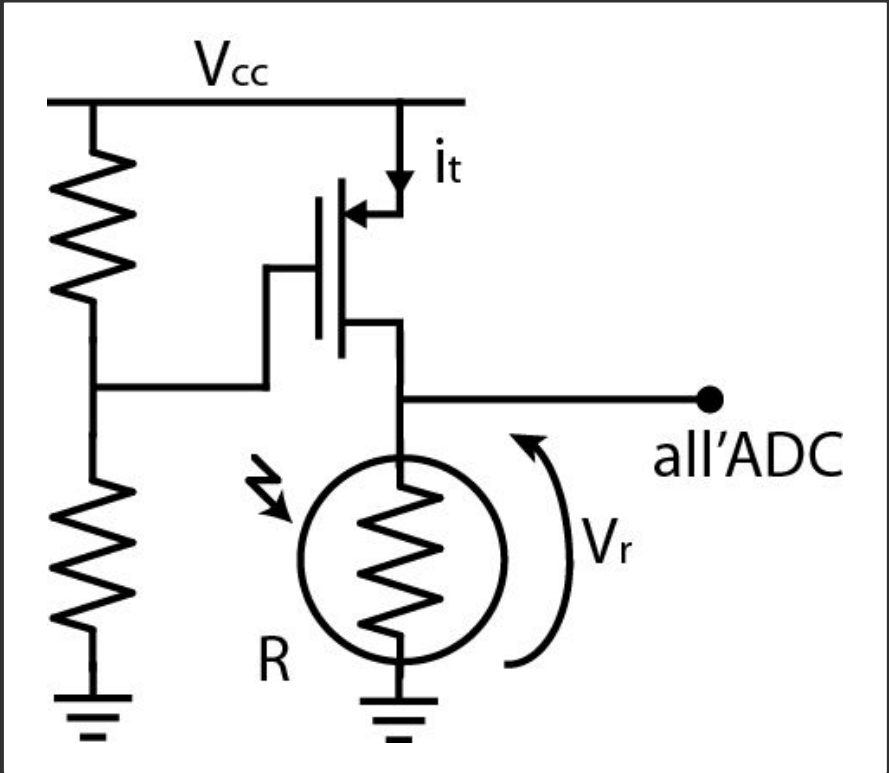
Fotodiodo



Un **fotodiodo** converte un segnale **luminoso** in un segnale in **corrente** che può essere **letto** in **analogico** tramite una resistenza

```
valueIn = analogRead(analogPinIn);
```

Fotoresistenza



Una **fotoresistenza abbassa** la sua **resistenza** all'**aumentare** della **luce** che la colpisce.

Come nel fotodiodo si legge la **tensione analogica** ai capi della fotoresistenza.

Il **transistor** genera una **corrente di riferimento** in modo che:

$$V_r = R \cdot I_t$$

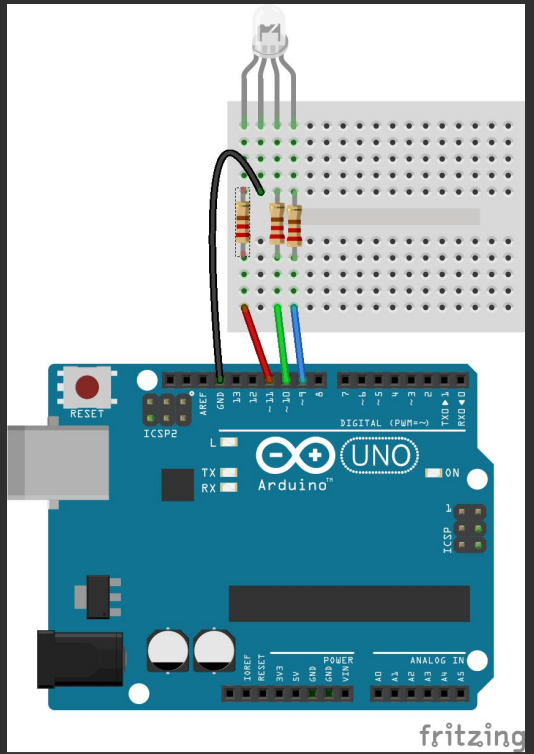
Photoresistor

A **photoresistor** device has the property to increase and decrease his resistance due to **variation of light** on his surface

You can use it simply as an analog sensor, like a **potentiometer** adding the right resistance

RGB LED

```
int redPin= 11, greenPin = 9, bluePin = 10;

void setup() {
      pinMode(redPin, OUTPUT);
      pinMode(greenPin, OUTPUT);
      pinMode(bluePin, OUTPUT);
}

void loop() {
      setRGB(255,0,0); delay(500);
      setRGB(255,255,0); delay(500);
      setRGB(0,255,0); delay(500);
      setRGB(0,255,255); delay(500);
      setRGB(0,0,255); delay(500);
      setRGB(255,0,255); delay(500);
}
void setRGB(int red, int green, int blue) {
      analogWrite(redPin,red);
      analogWrite(greenPin,green);
      analogWrite(bluePin,blue);
}
```



Esercizio 4: transizione tra tutti i colori di un RGB

Esercizio 4: transizione tra tutti i colori di un RGB

```
int redValue= 255, greenValue = 0, blueValue = 0;

// other variables declaration and setup() as before

void loop() {
      for (int i = 0;i < 255; i++) {
            redValue--;
            greenValue++;
            setRGB(redValue,greenValue,blueValue); delay(15);
      }
      for (int i = 0;i < 255; i++) {
            greenValue--;
            blueValue++;
            setRGB(redValue,greenValue,blueValue); delay(15);
      }
      for (int i = 0;i < 255; i++) {
            blueValue--;
            redValue++;
            setRGB(redValue,greenValue,blueValue); delay(15);
      }
}
```

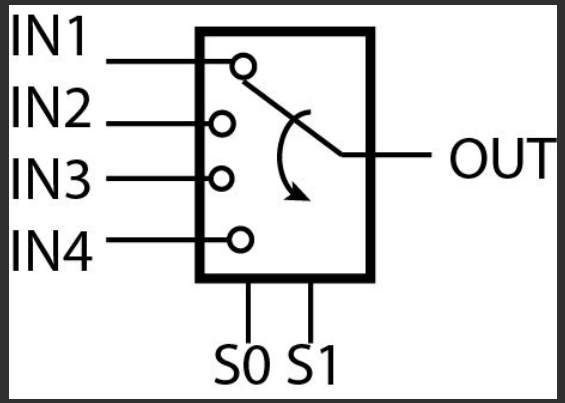
Esercizio 4: transizione tra tutti i colori di un RGB

```
int rgbValues[] = {255, 0, 0};
int speed = 10;
// other variables declaration and setup() as before

void loop() {
      int y = 0;
      while (y < 3) {
            for (int i = 0;i < 255; i+=speed) { // i+=speed is equal to i=i+speed
                  rgbValues[y] = rgbValues[y] - speed;
                  rgbValues[(y+1)%3] = rgbValues[(y+1)%3] + speed; // % module operand is the is
                                                                   the remainder of the euclidean division
                  setRGB(rgbValues[0] ,rgbValues[1] ,rgbValues[2]);
                  delay(15);
            }
      y++;
      }
}

// setRGB as before
```

https://electronics.stackexchange.com/questions/43498/how-can-i-use-a-12-v-input-on-a-digital-arduino-pin

Elettronica varia

Multiplexer



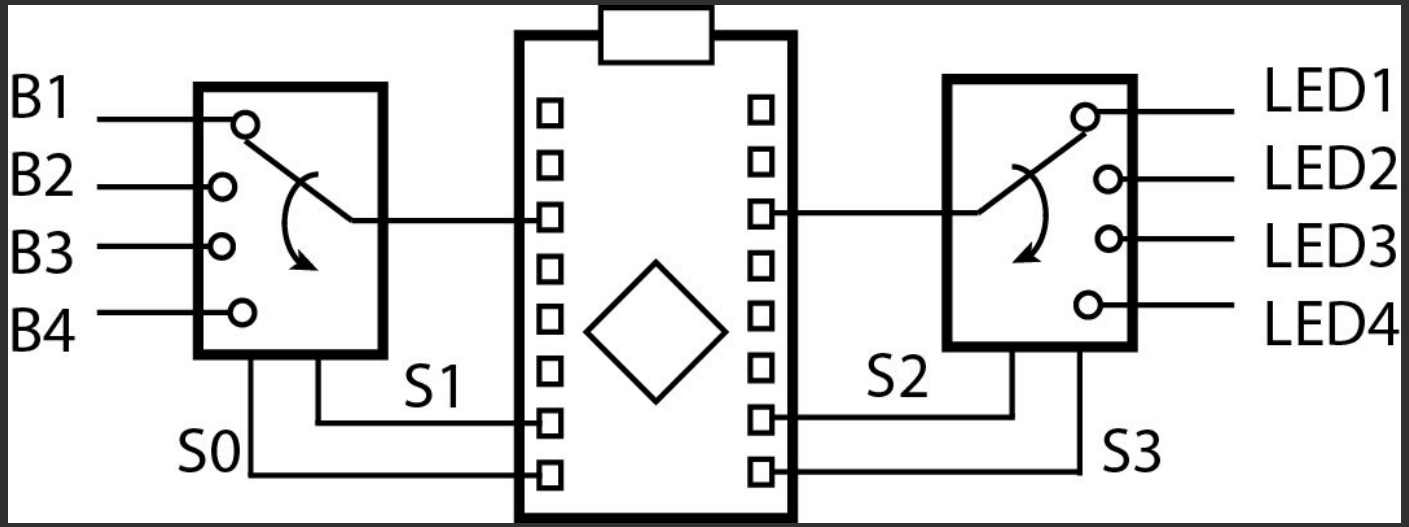
Un **multiplexer** condensa **più segnali** (IN1-4) in **uno unico** (OUT)

I segnali **S selezionano** quale **IN** è **connesso** ad **OUT**

Si **risparmiano pin** del Arduino

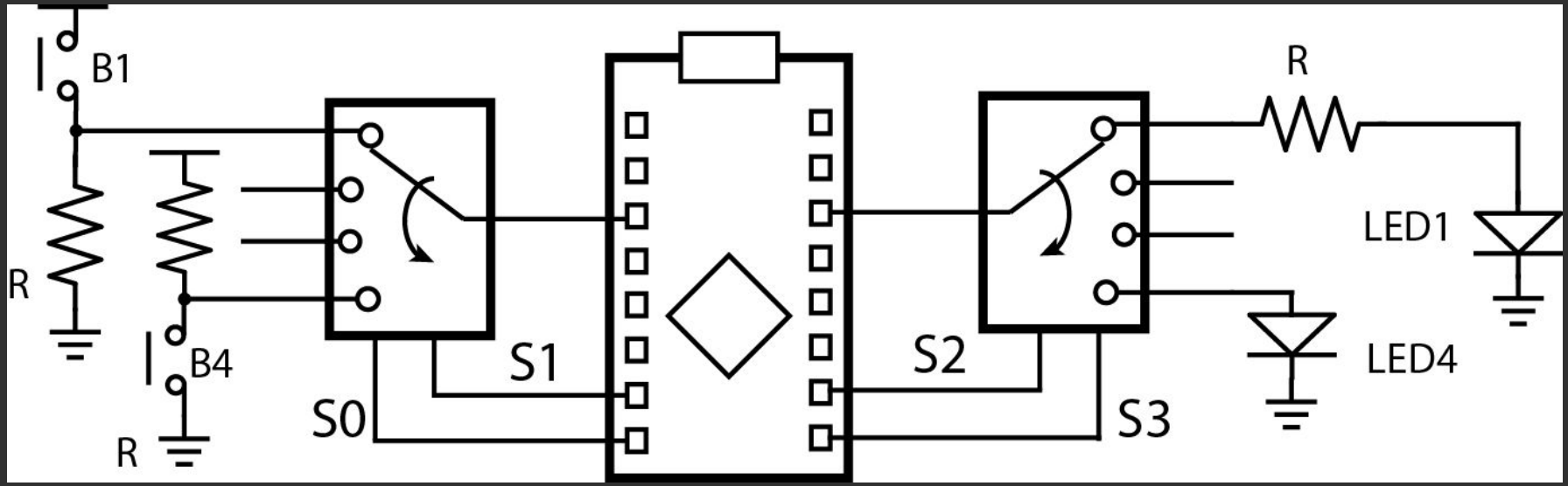
S1	*S0*	*OUT*
0	0	IN1
0	1	IN2
1	0	IN3
1	1	IN4

Multiplexer



Gestire 4 led con **4 pulsanti** interponendo dei **multiplexer**

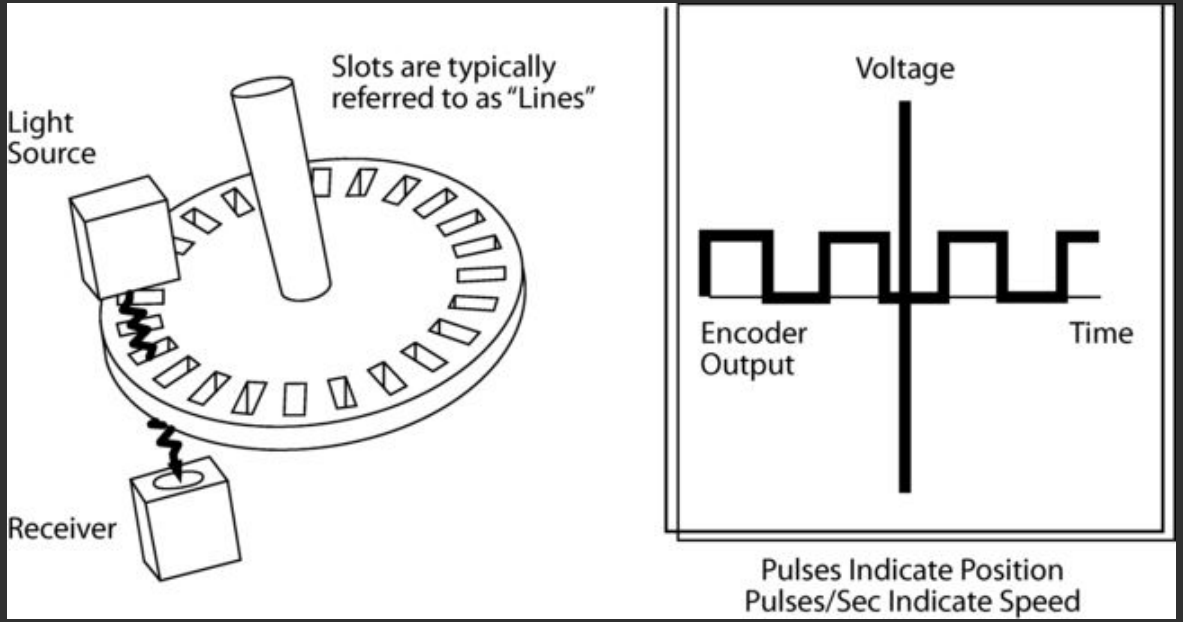
Multiplexer



Metto il **pull-up** o il **pull-down** sui **pulsanti**?

Che **resistenza** metto in **serie** ai **led**?

Rotary Encoder

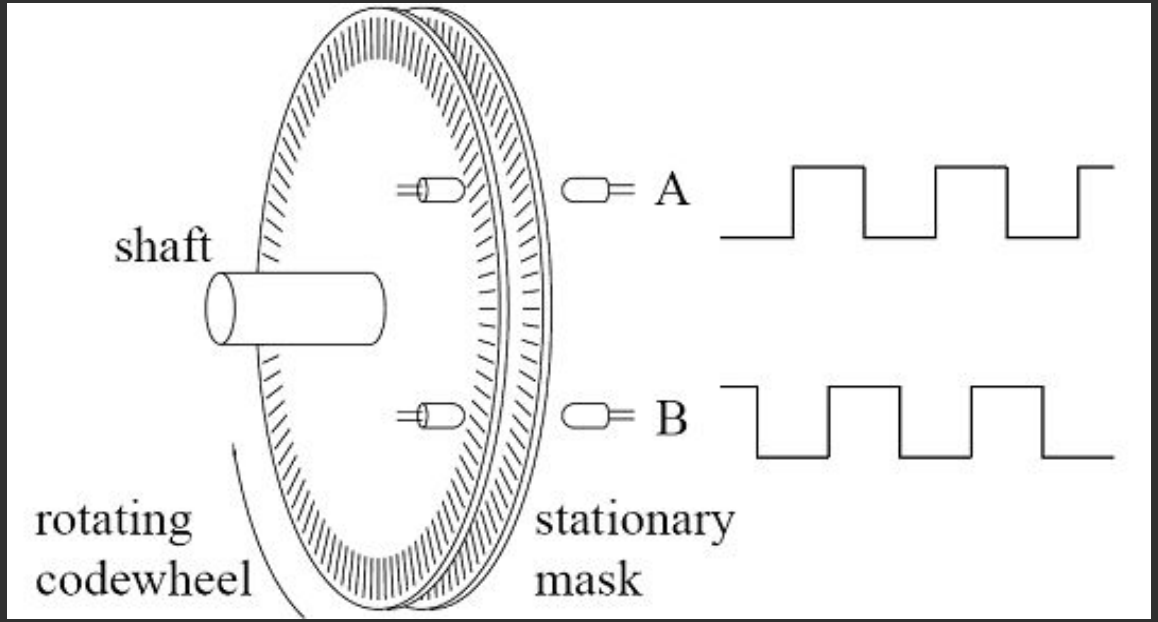


Un **encoder rotativo** possiede un **anello dentellato**

Contare **1, 0, 1, 0** significa spostarsi di **4 posizioni**

Non ho informazioni sul **senso** della rotazione

Rotary Encoder



Usando **2 ruote non allineate** ottengo informazione anche sul **senso** di rotazione

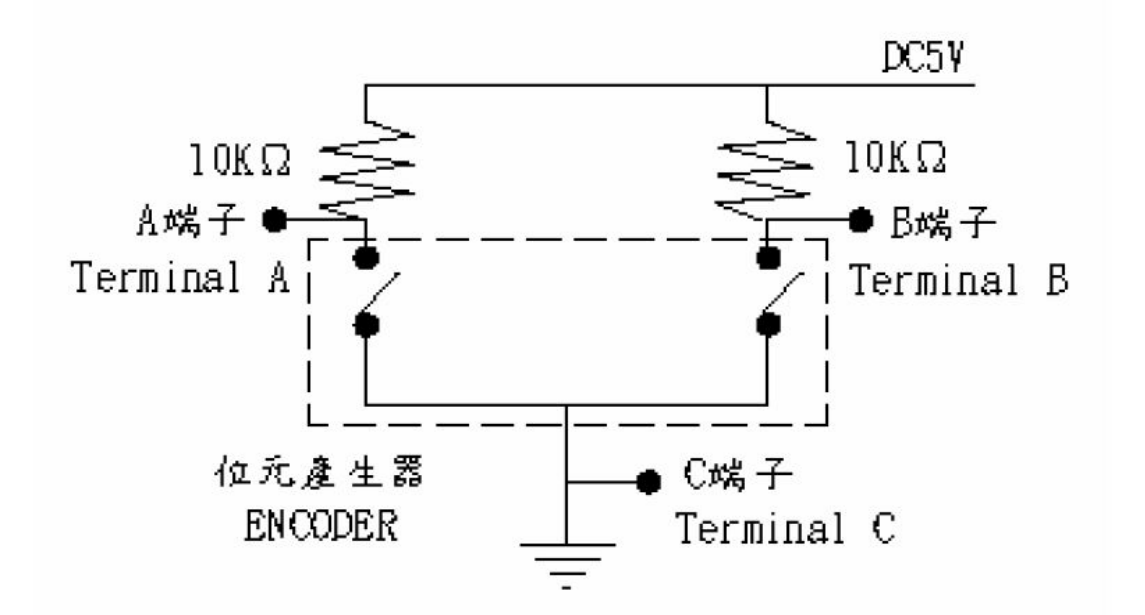
Orario

A	*B*
0	1
0	0
1	0
1	1

Antiorario

A	*B*
1	1
1	0
0	0
0	1

Rotary Encoder



Il **datasheet** del componente riporta le **informazioni utili** al suo utilizzo

In questo caso chiede:

- **Terminale C** a 0 V

- l'uso di 2 resistenze di **pull-up da 10kOhm** sui terminali A e B

- pull-up connessi a **5 V** in continua (**DC**)

Rotary Encoder

```
int encA = 14;
int encB = 15;

void setup() {
      pinMode(encA, INPUT);
      pinMode(encB, INPUT);
}

void loop() {
      encANew = digitalRead(encA);
      encBNew = digitalRead(encB);









}
```

Orario

A	B
0	1
0	0
1	0
1	1

Antiorario

A	B
1	1
1	0
0	0
0	1

Rotary Encoder

```
int encA = 14;
int encB = 15;

void setup() {
      pinMode(encA, INPUT);
      pinMode(encB, INPUT);
}

void loop() {
      encANew = digitalRead(encA);
      encBNew = digitalRead(encB);









}
```

EX_OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Orario

A	B
0	1
0	0
1	0
1	1

Antiorario

A	B
1	1
1	0
0	0
0	1

Rotary Encoder

```
int encA = 14;
int encB = 15;

void setup() {
      pinMode(encA, INPUT);
      pinMode(encB, INPUT);
}

void loop() {
      encANew = digitalRead(encA);
      encBNew = digitalRead(encB);

      if(encBOld^encANew) // funzione EX-OR
            counter++;
      else
            counter--;

      encAOld = encANew;
      encBOld = encBNew;
}
```

EX_OR

A	B	Y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Orario

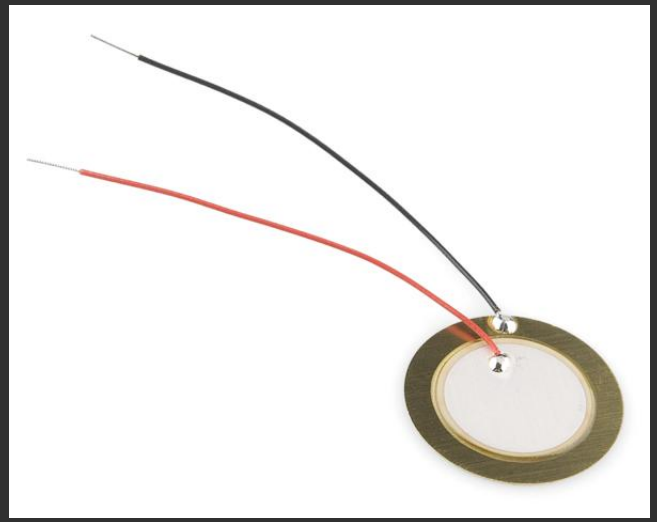
A	B
0	1
0	0
1	0
1	1

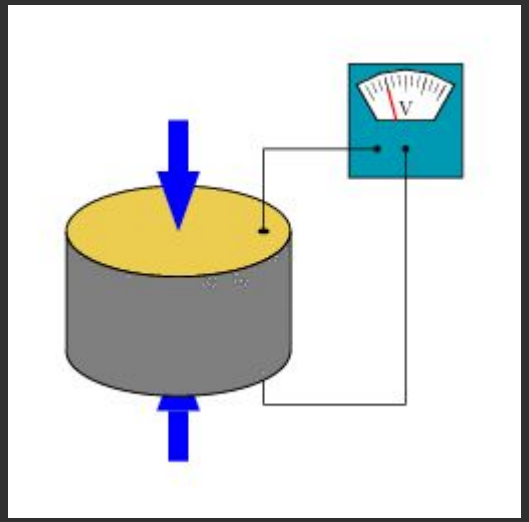
Antiorario

A	B
1	1
1	0
0	0
0	1

Piezoelettrico

Un dispositivo **piezoelettrico** è in grado di convertire una **vibrazione meccanica** in un **segnale elettrico** e **viceversa**

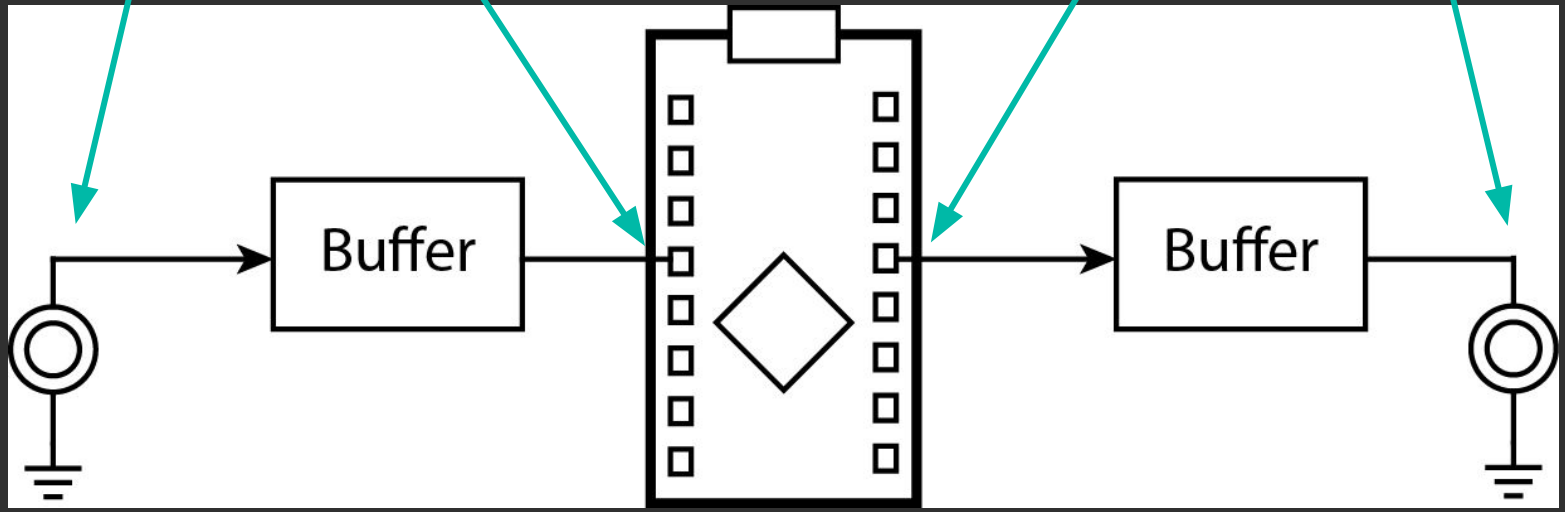




Piezoelettrico

Ingresso tramite pin **analogico**

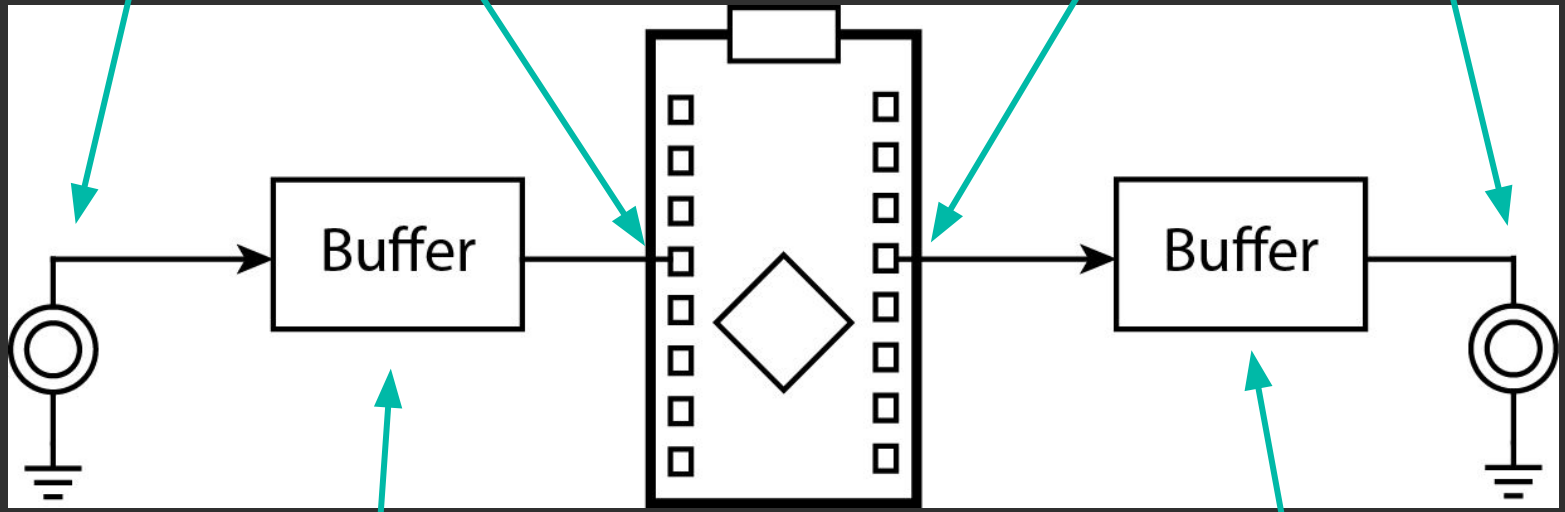
Pin **PWM** che genera l'**uscita**



Piezoelettrico

Ingresso tramite pin **analogico**

Pin **PWM** che genera l'**uscita**



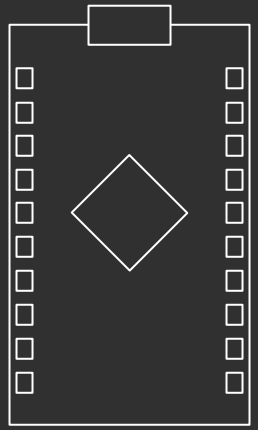
I **buffer** adattano

il segnale del piezoelettrico a Arduino

il segnale da Arduino al piezoelettrico

Piezoelettrico

Ingresso: qualunque cosa che **genera suono** o **vibra**



Uscita: mini **speaker** o **buzzer**

Secret door-knock lock

```
int analogPinIn = 14;
int valueIn, index = 0, elapsedTime = 0;
int sequence = [10, 10, 10]; // tempo di attesa

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

void loop() {
      while (index <= 3) {
            valueIn = analogRead(analogPinIn);
            while (valueIn > 50) {      // qualcosa sta vibrando
                  delay(300);
                  valueIn = analogRead(analogPinIn);
                  index++;
            }
            delay(1);
            elapsedTime[index]++;
      }
      ... continuate voi ...
}
```

Bussando, genero delle **vibrazioni** sulla porta che possono essere lette tramite un **piezoelettrico**. Se la **sequenza** è quella **impostata**, la serratura **si sblocca**