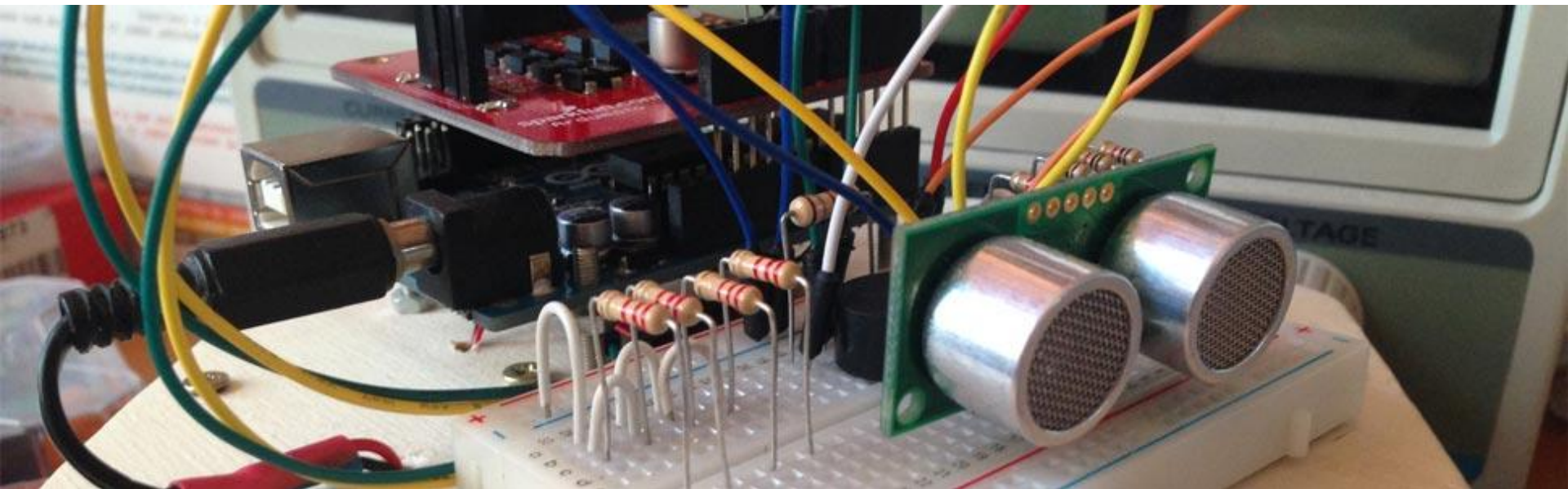


L'alfabeto di Arduino

Introduzione all'uso di Arduino

lezione 3

Prof. Michele Maffucci



Argomenti

- Introduzione
- Input analogici
- Comunicare
- Led RGB
- Da PC ad Arduino
- Musica

Il codice e le slide utilizzate sono suscettibili di variazioni/correzioni che potranno essere fatte in ogni momento.

Introduzione

Il seguente corso intende fornire le **competenze di base** per la realizzazione di lezioni di didattica delle robotica nella scuola secondaria di secondo grado.

Il corso ben si adatta a tutti i maker, studenti ed adulti, che per passione nell'elettronica necessitano di un'introduzione all'uso di Arduino.

Il docente che intendesse sviluppare un percorso didattico in cui si desidera realizzare dispositivi elettronici in grado di interfacciarsi col mondo fisico, potrà utilizzare queste lezioni come base per implementare moduli didattici aggiuntivi, pertanto questo corso è da intendersi come il mio personale tentativo di strutturare un percorso iniziale e modellabile a seconda del tipo di indirizzo della scuola. Chi vorrà potrà effettuare miglioramenti su quanto da me scritto.

Il percorso scelto è un estratto delle lezioni svolte durante i miei corsi di elettronica, sistemi ed impianti elettrici. Nelle slide vi sono cenni teorici di elettrotecnica che non sostituiscono in alcun modo il libro di testo, ma vogliono essere un primo passo per condurre il lettore ad un approfondimento su testi specializzati.

Il corso è basato sulla piattaforma Open Source e Open Hardware **Arduino** e fa uso dell'**Arduino starter kit**. Questa scelta non implica l'adozione di queste slide in corsi che non fanno uso di questo kit, ma è semplicemente una scelta organizzativa per lo svolgimento di questo corso di formazione. Alle proposte incluse nel kit ho aggiunto ulteriori sperimentazioni. Tutti i componenti possono essere acquistati separatamente.

Ulteriori approfondimenti e risorse a questo corso possono essere trovate sul mio sito personale al seguente link:

<http://www.maffucci.it/area-studenti/arduino/>

Nella [sezione dedicata ad Arduino](#), sul mio sito personale, oltre ad ulteriori lezioni, di cui queste slide ne sono una sintesi, è possibile consultare un manuale di programmazione, in cui vengono dettagliate le istruzioni. Per rendere pratico l'utilizzo del manuale ne è stata realizzata anche una versione portable per dispositivi mobili **iOS** e **Android**, maggiori informazioni possono essere trovate seguendo il [link](#).



Esempi utilizzati nel corso.

Tutti i programmi utilizzati nel corso possono essere prelevati al seguente link:

<https://github.com/maffucci/LezioniArduino/tree/master/corso01>

Gli sketch Arduino sono da scompattare nella cartella sketchbook.

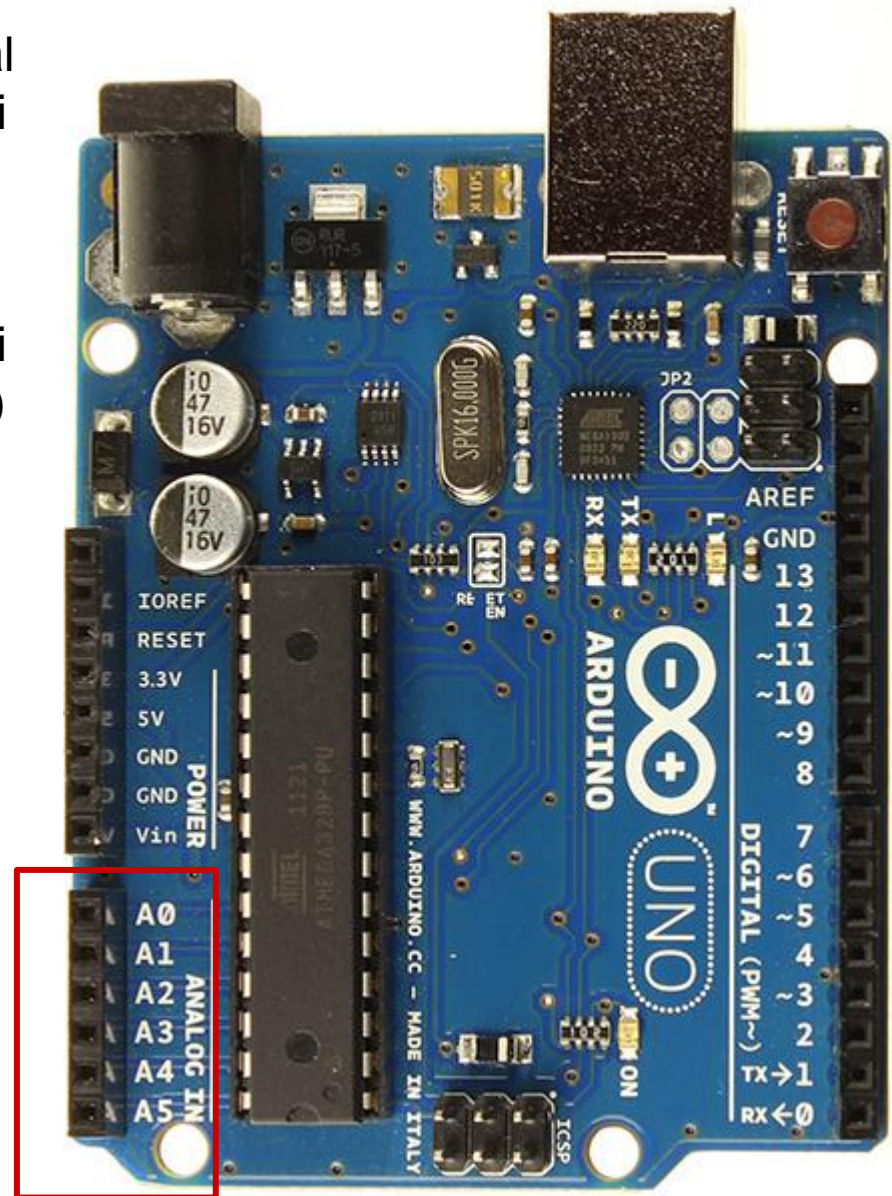
Questo corso è nato in brevissimo tempo (circa 15 giorni) e quindi possibile che siano presenti delle imperfezioni, ringrazio fin d'ora chi vorrà segnalarmi correzioni e miglioramenti.

Per contatti ed ulteriori informazioni rimando alle ultime pagine di queste slide.

Grazie

Input analogici

- Il microcontrollore ATmega328 è dotato di 6 ADC (Analog to Digital Converter - Convertitori Analogici Digitali)
- Gli ingressi analogici leggono valori compresi tra 0 e 5 volt
- La risoluzione dei valori convertiti in digitale è di 10 bit (1024 valori)
- Ogni bit equivale a valori di tensione pari a $5/1024 = 4,8 \text{ mV}$ che è la più piccola misura di tensione che si può rilevare



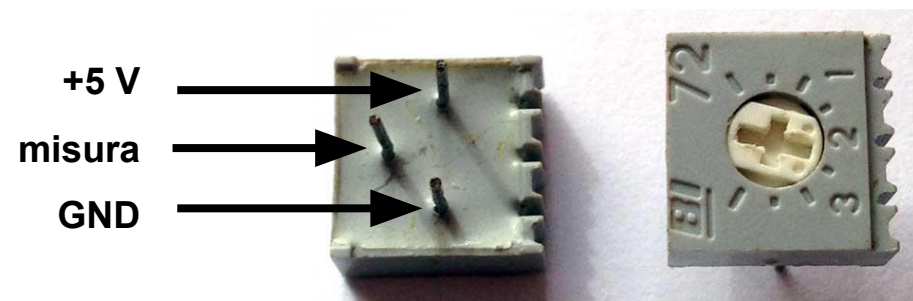
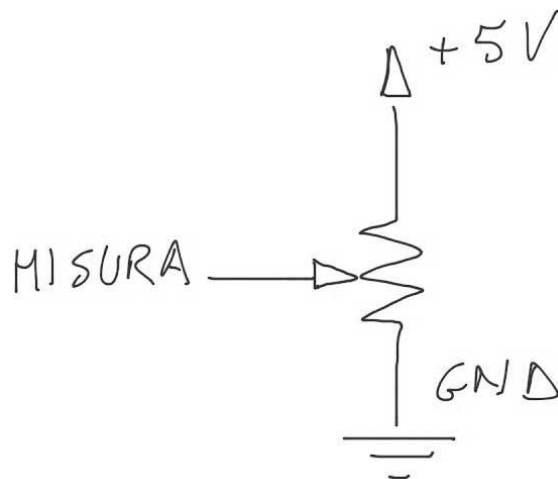
leggere un valore analogico

Le grandezze fisiche del mondo reale sono di tipo analogico ed Arduino dispone di una serie di ingressi adibiti alla lettura di grandezze di tipo analogico, che come dettagliato nella lezione 1, vengono poi convertire dal microcontrollore in grandezze di tipo digitale.

Le variazioni di grandezze di tipo analogico vengono lette da Arduino come variazioni di tensione sugli ingressi analogici.

Un sensore generico quindi, connesso sui pin analogici fornirà una tensione che sarà funzione della grandezza fisica misurata.

Per poter simulare la variazione di tensione e quindi studiare il comportamento di un generico sensore e capire come leggere valori analogici, utilizzeremo in questa fase un potenziometro o un trimmer, componente elettronico che consente di variare la tensione presente ai suoi capi.

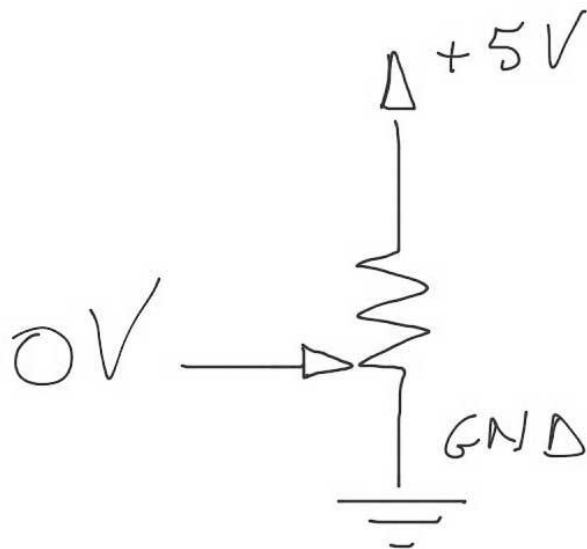


leggere un valore analogico

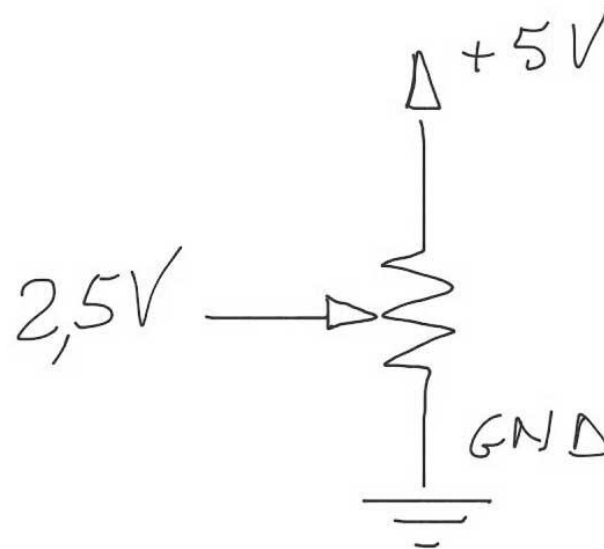
trimmer

1. Ruotando la vite in senso **antiorario** fine a fine corsa la tensione presente sul piedino centrale sarà 0V.
2. Ruotando la vite in senso orario in posizione centrale si avrà una tensione di 2,5 V.
3. Ruotando la vite in senso orario fine a fine corsa si avrà una tensione di 5 V.

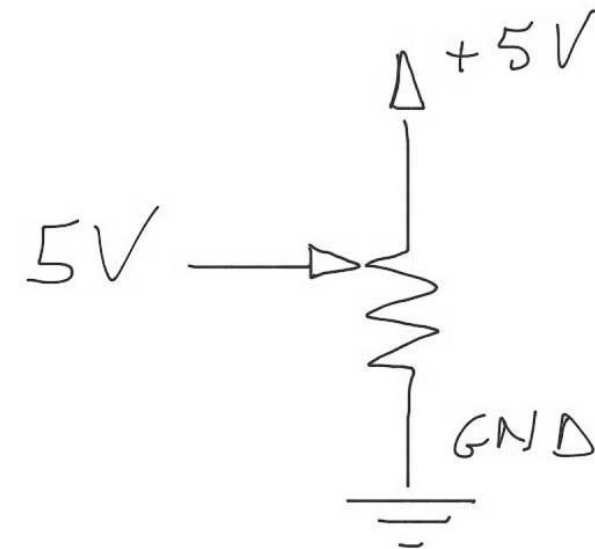
antiorario (fine corsa)



oraria (posizione centrale)

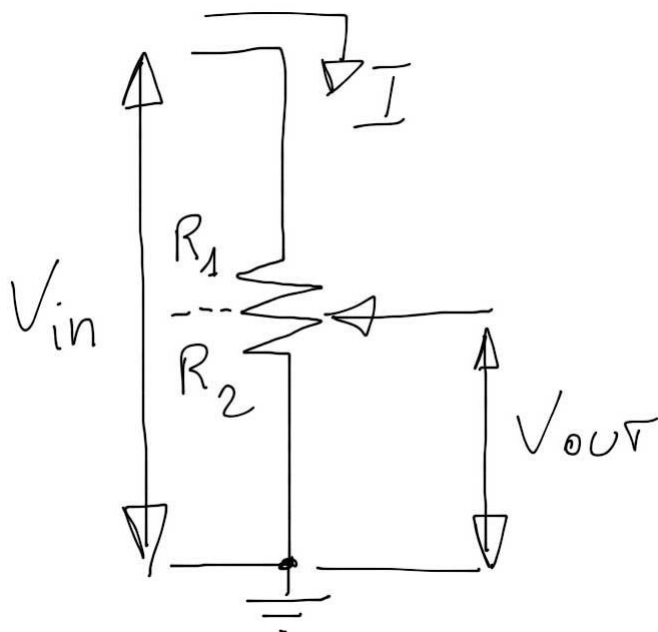


oraria (fine corsa)

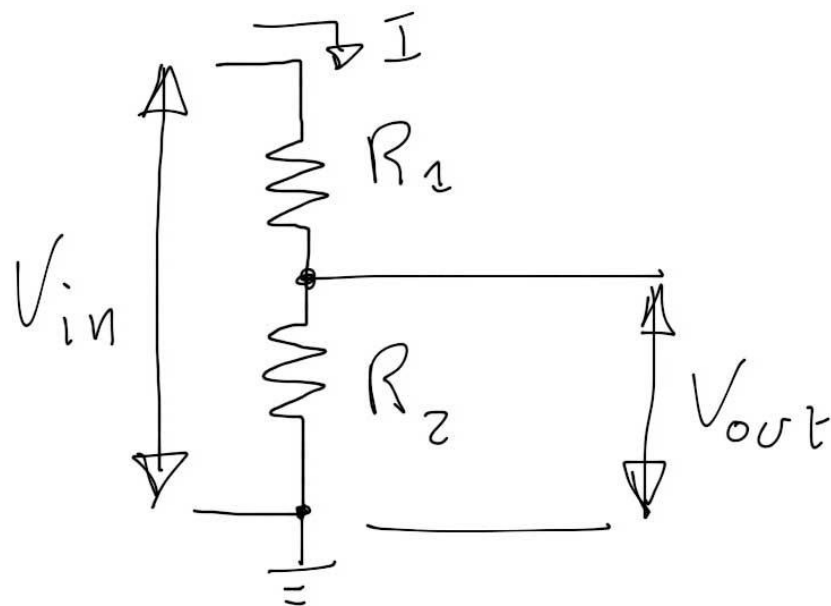


Il cursore del trimmer, rappresentato dalla freccia nel simbolo elettronico, o dalla vite nell'immagine prima esposta, può passare da un estremo all'altro dove è collegata direttamente a massa (0 V) oppure all'alimentazione alla V_{in} nel nostro caso 5 V, passando attraverso tutte le posizioni intermedie, si potrà quindi dosare la tensione sul piedino centrale (V_{out}) come frazione della tensione di alimentazione:

$$V_{in} \geq V_{out} \geq 0$$



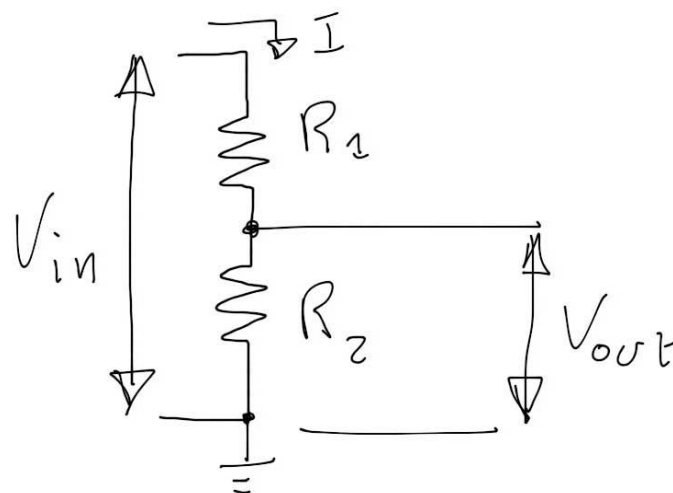
Varinando l'angolo di rotazione varieranno i valori di R_1 ed R_2 . Il valore della tensione di uscita sarà quella del partitore di tensione tra R_1 e R_2 .



leggere un valore analogico

trimmer

partitore di tensione



Per la legge di Ohm avremo che la differenza di potenziale ai capi di ogni resistore e la tensione totale V_{in} sarà:

$$V_{R_1} = R_1 \cdot I$$

$$V_{R_2} = R_2 \cdot I$$

$$V_{in} = (R_1 + R_2) \cdot I$$



Si ricava la I dall'equazione alla magli V_{in} e si sostituisce all'interno della V_{R_1} e V_{R_2} :

da cui

$$I = \frac{V_{in}}{(R_1 + R_2)}$$

sostituendolo in V_{R_1} e V_{R_2}



Da cui le due formule del partitore per ottenere la V_{R_1} e V_{R_2} :

$$V_{R_1} = V_{in} \cdot \frac{R_1}{R_1 + R_2}$$

$$V_{R_2} = V_{in} \cdot \frac{R_2}{R_1 + R_2}$$

leggere un valore analogico

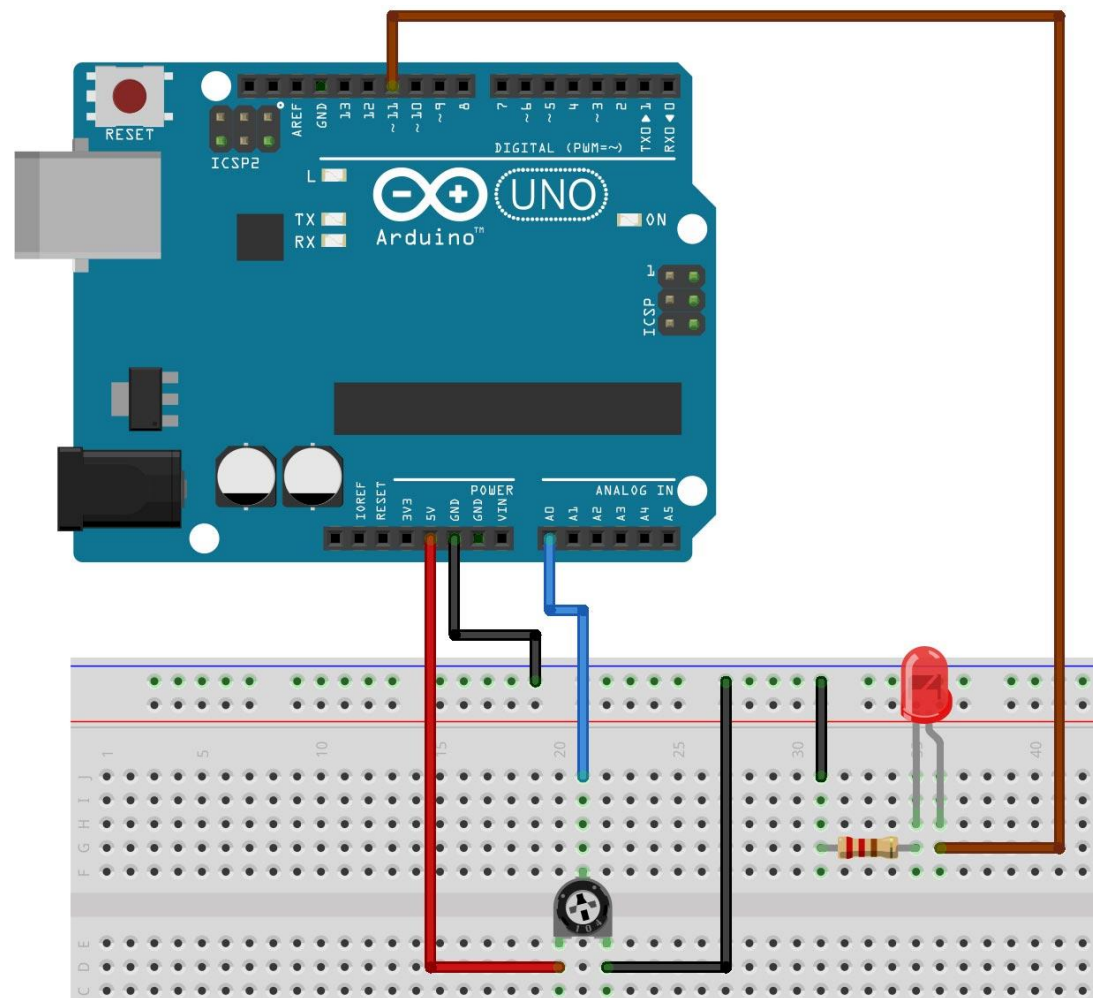
1/3

Si legge il voltaggio di un **pin analogico** usando un un trimmer che fornisce un voltaggio tra 0 e 5 volt.

Si varierà la luminosità del LED in funzione della tensione presente sul pin analogico.

Componenti:

- trimmer da 10KOhm
- R in serie al LED da 220
- LED



Per approfondimenti sulla misura di resistenze su trimmer e potenziometri consultare il [link](#).

leggere un valore analogico

2/3

analogRead()

sketch18

```
/* Prof. Michele Maffucci
15.03.2014

Regolazione luminosità LED mediante
trimmer

Questo codice è di dominio pubblico
*/

// variabile in cui verrà memorizzato il valore presente sul pin A0
int misura = 0;

// pin a cui è connesso il LED
int pinLed = 11;

// variabile in cui conservare il valore inserito su A0 e poi usato per
// impostare il Duty Cycle
int val = 0;

void setup(){
  pinMode(pinLed, OUTPUT); // definizione di ledPin come output
}

void loop(){
  // analogRead leggerà il valore su A0 restituendo un valore tra 0 e 1023
  // per approfondimenti si consultì il link: http://wp.me/p4kwmk-1Qd
  val = analogRead(misura);

  // analogWrite() accetta come secondo parametro (PWM) valori tra 0 e 254
  // pertanto "rimappiamo" i valori letti da analogRead() nell'intervallo
  // tra 0 e 254 dividendo per 4 i valori di val
  val = val/4;

  // accendiamo il LED con un valore del Duty Cycle pari a val
  analogWrite(pinLed,val);
}
```

analogRead(pin)

Legge un valore di tensione applicato al piedino analogico 'pin' con una risoluzione di 10 bit. La funzione restituisce un valore compreso tra 0 e 1023.

I pin analogici a differenza di quelli digitali non hanno bisogno di essere dichiarati come pin di INPUT o OUTPUT.

Per approfondimenti seguire il [link](#).

```

/* Prof. Michele Maffucci
15.03.2014

Regolazione luminosità LED mediante
trimmer
si utilizza la funzione map

Questo codice è di dominio pubblico
*/

// variabile in cui verrà memorizzato il valore presente
int misura = 0;

// pin a cui è connesso il LED
int pinLed = 11;

// variabile in cui conservare il valore inserito su A0
int val = 0;

// variabile in cui memorizzare il Duty Cycle
int inputVal = 0;

void setup(){
  pinMode(pinLed, OUTPUT); // definizione di ledPin come
}

void loop(){
  // analogRead leggerà il valore su A0 restituendo un va
  // per approfondimenti si consulti il link: http://wp.m
  val = analogRead(misura);

  // analogWrite() accetta come secondo parametro (PWM) valori tra 0 e 254
  // pertanto "rimappiamo" i valori letti da analogRead() nell'intervallo
  // tra 0 e 254 usando la funzione map
  // per approfondimenti si consulti il link: http://wp.me/p4kwmk-1Tu
  inputVal = map(val, 0, 1023, 0, 254);

  // accendiamo il LED con un valore del Duty Cycle pari a val
  analogWrite(pinLed,inputVal);
}

```

map(valore, daValMin, daValMax, aValMin, aValMax)

Rimappa un numero da un intervallo ad un altro intervallo

Parametri

valore: valore da rimappare

daValMin: valore minimo dell'intervallo di partenza

daValMax: valore massimo dell'intervallo di partenza

aValMin: valore minimo dell'intervallo di arrivo

aValMax: valore massimo dell'intervallo di arrivo

Risultato

valore rimappato nell'intervallo **aValMin, aValMax**

Per approfondimenti seguire il [link](#).



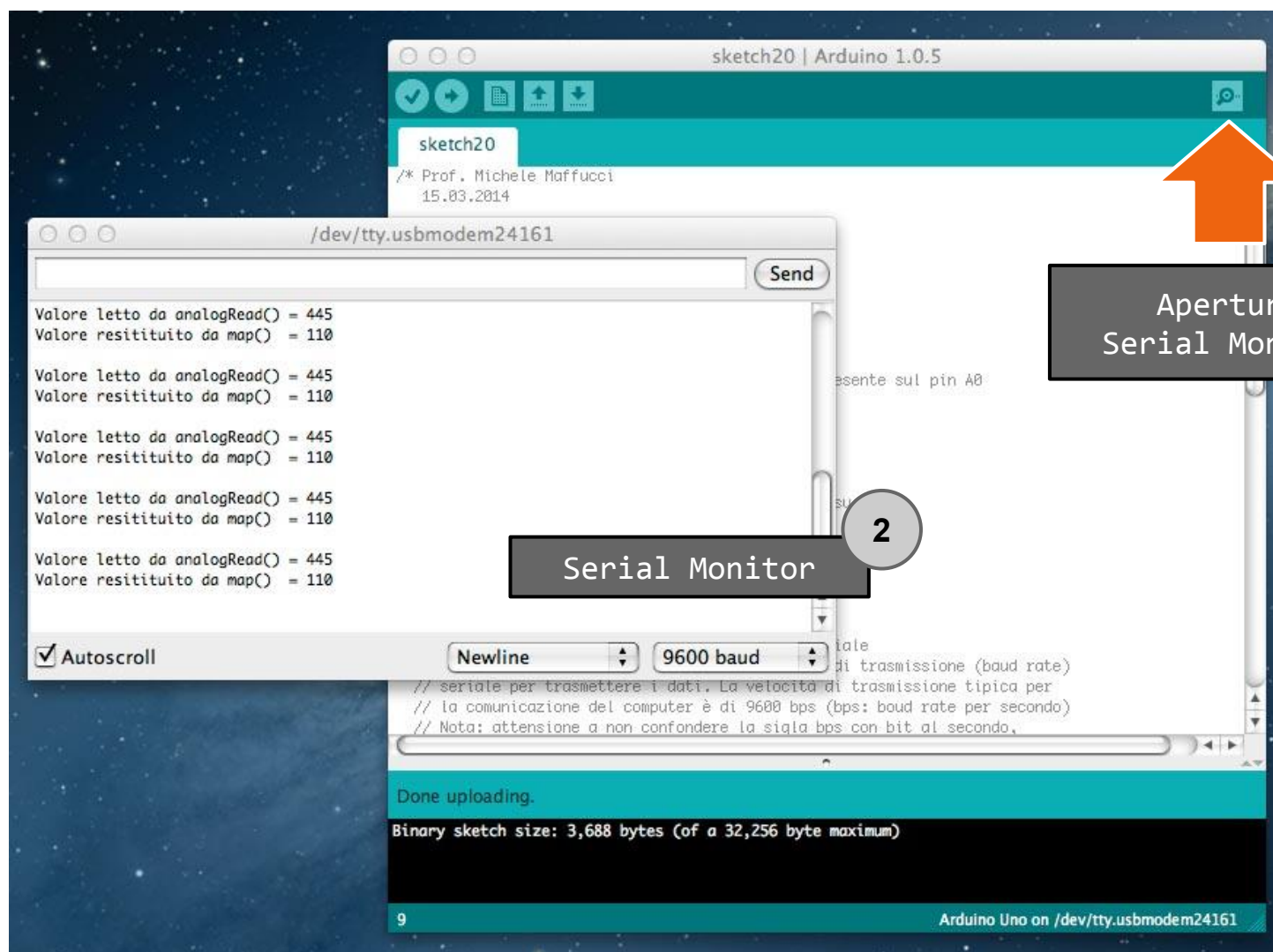
Comunicare

Arduino utilizza il cavo USB per poter essere programmato ma anche per comunicare con il altre periferiche, tra cui anche il computer. Per poter comunicare vengono utilizza i **comandi seriali**:

- **Serial.begin()**
per impostare la comunicazione seriale
- **Serial.print()**
per inviare dati al computer
- **Serial.read()**
per leggere dati dal computer

Serial Monitor

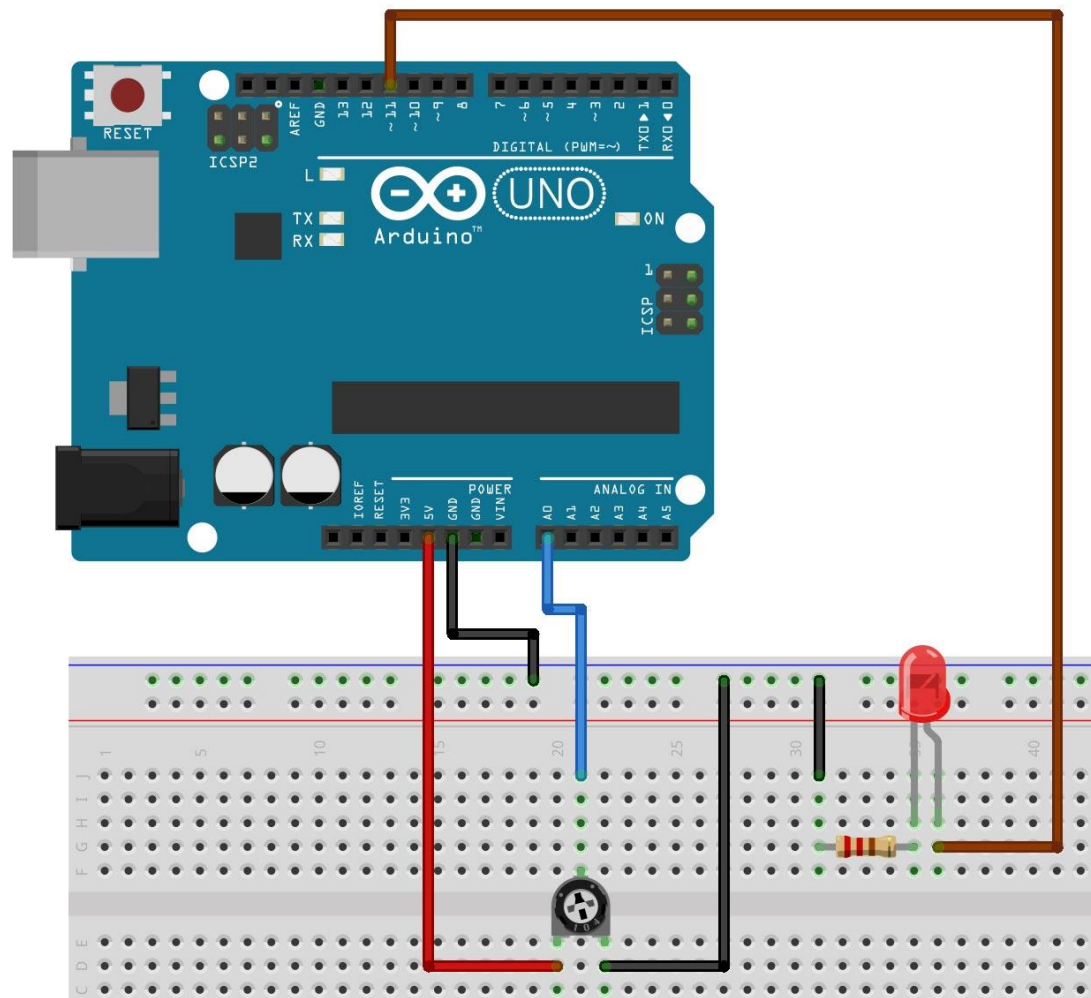
Utilizzando il circuito con trimmer e LED appena realizzato ed utilizziamo i comandi necessari per inviare sulla **Serial Monitor** i valori analogici letti.



Utilizziamo i comandi **SerialPrint()** e **SerialPrintln()** per inviare sulla Serial Monitor i valori analogici letti.
 Si legga il voltaggio di un **pin analogico** usando un un trimmer che fornisce un voltaggio tra 0 e 5 volt.
 Si varierà la luminosità del LED in funzione della tensione presente sul pin analogico.

Componenti:

- trimmer da 10KOhm
- R in serie al LED da 220
- LED



```
/* Prof. Michele Maffucci
15.03.2014

Regolazione luminosità LED mediante
trimmer; si utilizza la funzione map.
Lettura valori sulla Serial Monitor

Questo codice è di dominio pubblico
*/

// variabile in cui verrà memorizzato il valore pres
int misura = 0;

// pin a cui è connesso il LED
int pinLed = 11;

// variabile in cui conservare il valore inserito su
int val = 0;

// variabile in cui memorizzare il Duty Cycle
int inputVal = 0;

void setup(){
  // Serial.begin(rate) inizializzazione della seriale
  // Apre la porta seriale ed imposta la velocità di trasmissione (baud rate)
  // seriale per trasmettere i dati. La velocità di trasmissione tipica per
  // la comunicazione del computer è di 9600 bps (bps: boud rate per secondo)
  // Nota: attenzione a non confondere la sigla bps con bit al secondo,
  // nel nostro caso parliamo di boud, cioè simboli e ad ogni simbolo possono
  // corrispondere più bit.
  // per maggiori informazioni si consulti il link: http://wp.me/p4kwmk-211
  Serial.begin(9600);

  // definizione di ledPin come output
  pinMode(pinLed, OUTPUT);
}
```

Serial.begin(rate)

Serial.begin(rate) inizializzazione della seriale
Apre la porta seriale ed imposta la velocità di trasmissione (baud rate) seriale per trasmettere i dati. La velocità di trasmissione tipica per la comunicazione del computer è di 9600 bps (bps: boud rate per secondo)

Nota: attenzione a non confondere la sigla bps con bit al secondo, nel nostro caso parliamo di boud, cioè simboli e ad ogni simbolo possono corrispondere più bit.

Per approfondimenti seguire il [link](#).

continua...

```
void loop(){
  // analogRead leggerà il valore su A0
  // per approfondimenti si consulti il
  val = analogRead(misura);
```

```
// analogWrite() accetta come secondo parametro (PWM) valori tra 0 e 254
```

```
// pertanto "rimappiamo" i valori letti
// tra 0 e 254 usando la funzione map
// per approfondimenti si consulti il
inputVal = map(val, 0, 1023, 0, 254);
```

```
// la Serial.println(data) stampa i dati
// (invia i dati alla seriale e li visualizza)
// ritorno a capo automatico ed un avanzamento linea.
```

```
Serial.print("Valore letto da analogRead() = "); // stampa ciò che è incluso tra ""
Serial.println(val); // stampa val
Serial.print("Valore resituito da map() = "); // stampa ciò che è incluso tra ""
Serial.println(inputVal); // stampa inputVal
Serial.println(""); // va a capo
delay(1000); // attesa di 1 secondo per consentire la lettura
```

```
// accendiamo il LED con un valore del Duty Cycle pari a val
analogWrite(pinLed,inputVal);
```

}

Serial.print(data)

Stampa i dati sulla porta seriale (invia i dati alla seriale e li visualizza)

Per approfondimenti seguire il [link](#).

Serial.println(data)

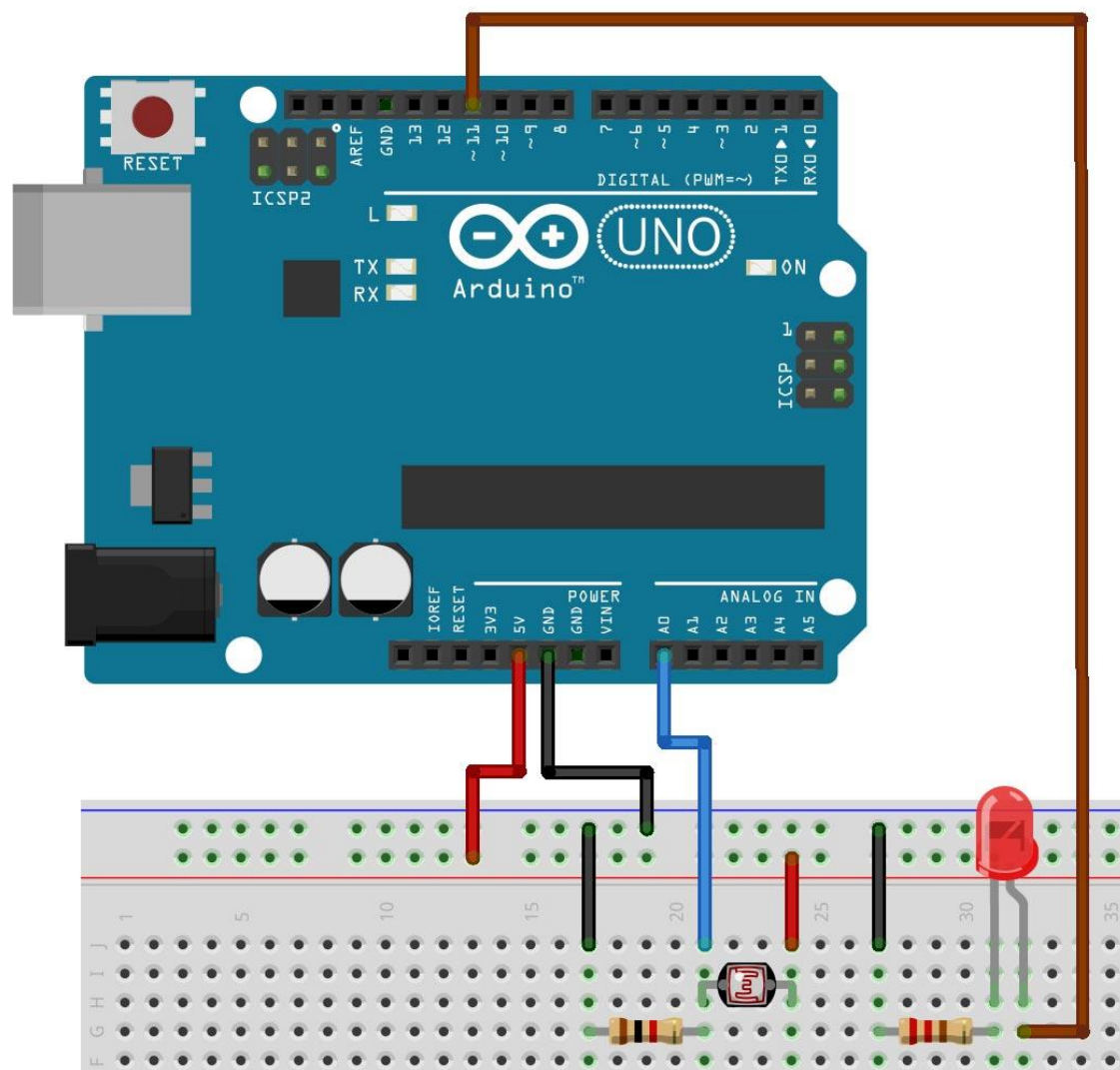
Stampa i dati sulla porta seriale (invia i dati alla seriale e li visualizza) seguito da un ritorno a capo automatico ed un avanzamento linea.

Per approfondimenti seguire il [link](#).

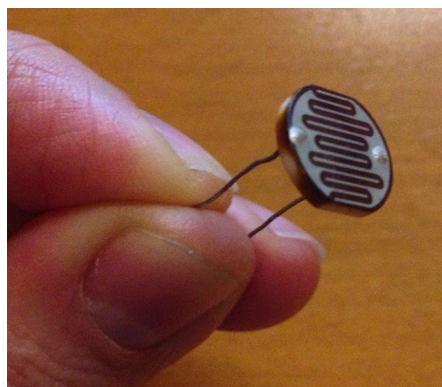
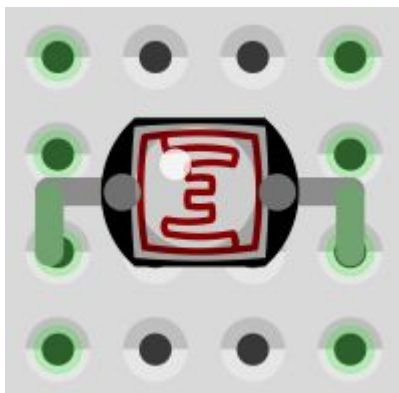
Utilizzare una fotoresistenza (LDR) per far variare la luminosità del LED.

Componenti:

- LDR
- R da 1 KOhm da mettere in serie all'LDR
- R in serie al LED da 220
- LED



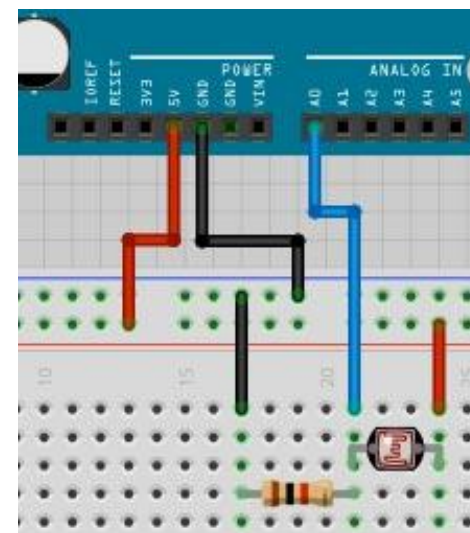
Fotoresistenza (LDR: Light Dependent Resistor)



LDR

- resistenza elevata se non colpita dalla luce;
- resistenza bassa se illuminata.

Grazie a questa sua caratteristica possiamo far variare la differenza di potenziale ai suoi capi.

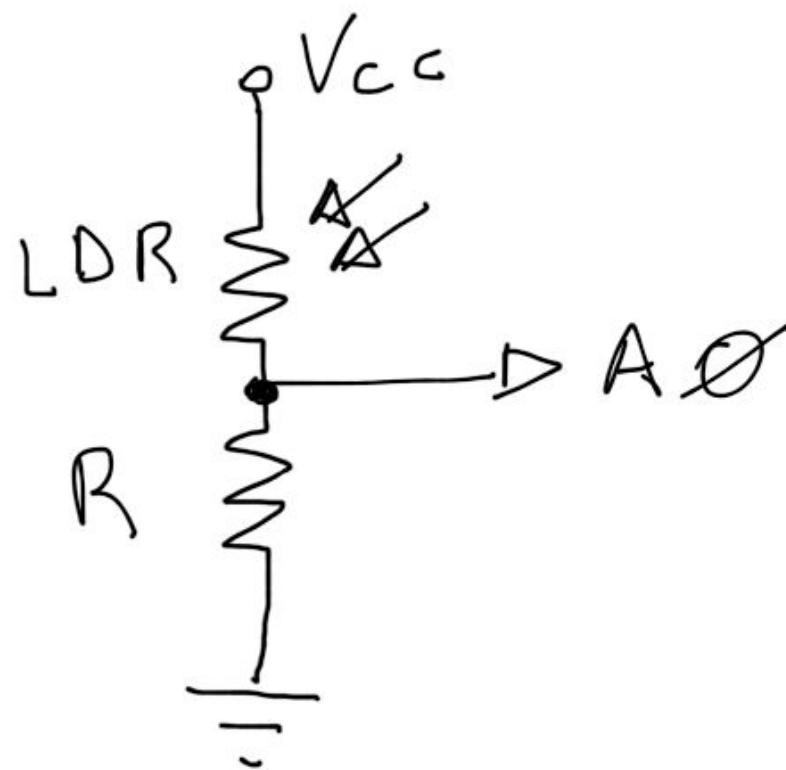
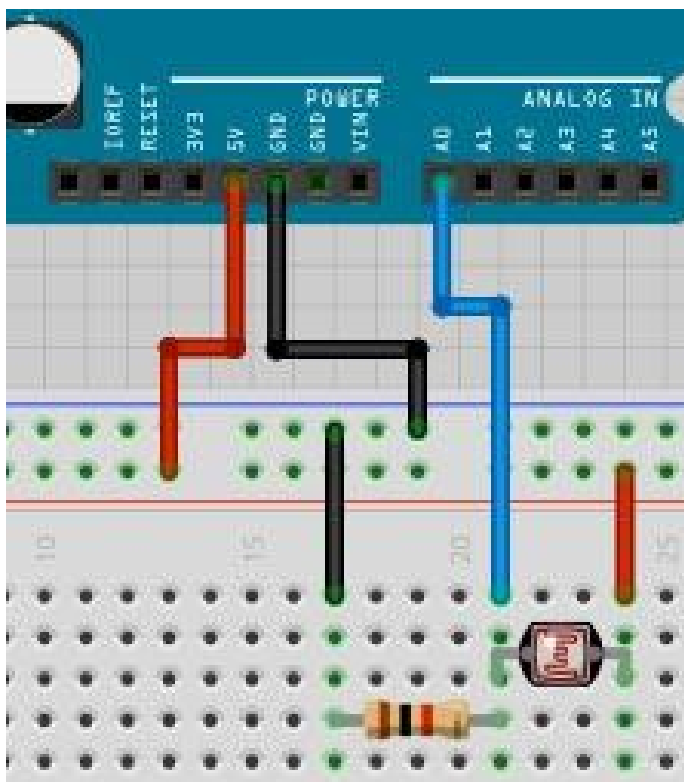


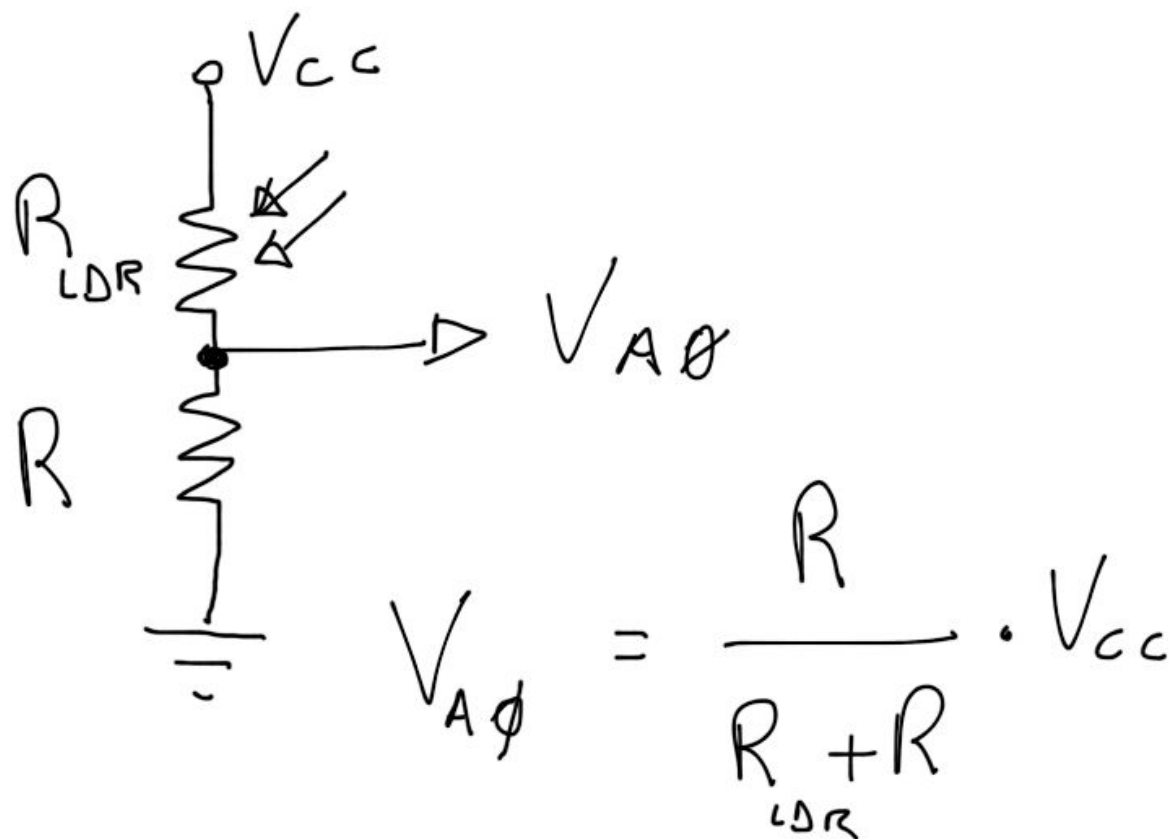
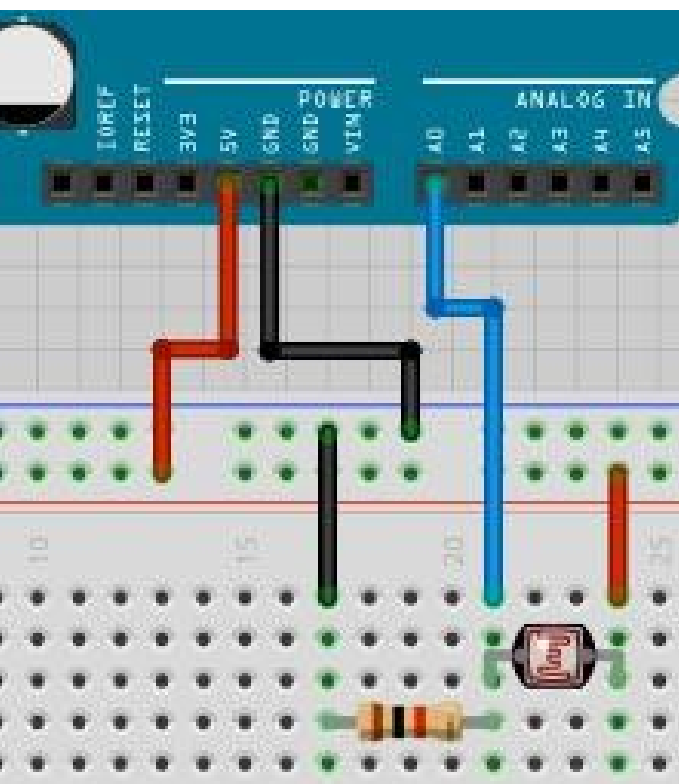
Serial Monitor

3/8

LDR

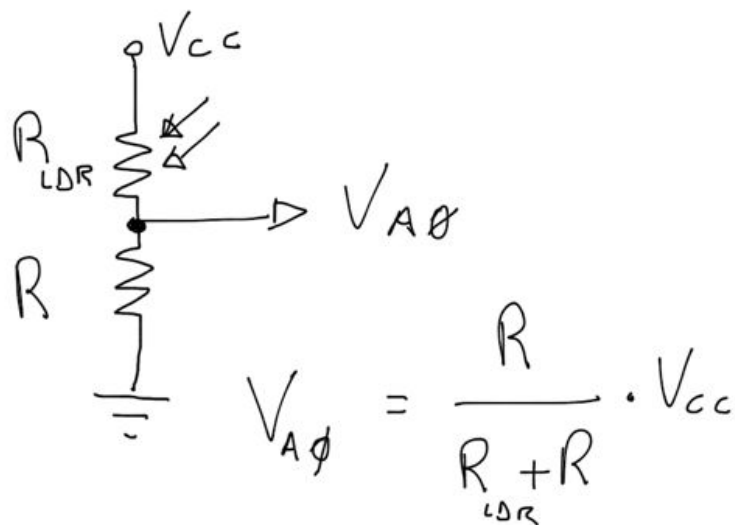
- L'LDR é connessa in serie ad una resistenza da 1K Ω . La tensione che verrà letta da Arduino sul pin A0 sarà quella esistente al nodo di collegamento tra LDR ed R.
- Utilizziamo le formule (viste nelle precedenti slide) sul **PARTITORE DI TENSIONE**.



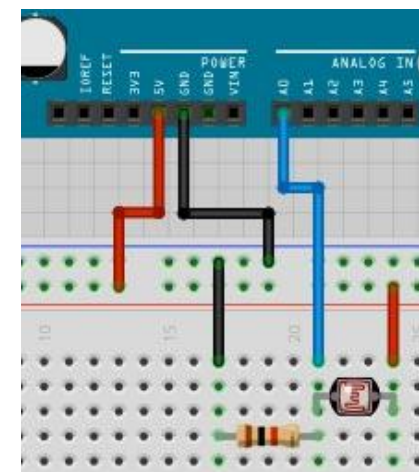


L'inserimento della resistenza da 1K Ω consente di evitare un collegamento diretto tra +Vcc e GND quando la resistenza dell'LDR è prossima a 0 (cortocircuito)

Condizione	Resistenza
LDR coperta con un dito	8 KOhm
Luce nella stanza	1 KOhm
Luce che colpisce direttamente LDR	150 Ohm



V_{cc}	R	R_{LDR}	V_{A0}
5V	1 KOhm	8 KOhm	0,56 V
5V	1 KOhm	1 KOhm	2,5 V
5V	1 KOhm	150 Ohm	4,35 V



```
/* Prof. Michele Maffucci
16.03.2014

Utilizzo di un LDR per controllare
la luminosità di un LED
Lettura valori sulla Serial Monitor

Questo codice è di dominio pubblico
*/

// variabile in cui verrà memorizzato il valore
const int misura = A0;

// pin a cui è connesso il LED
int val = 0;

// variabile in cui memorizzare il valore letto dal sensore
int inputVal = 0;

// pin a cui è connesso il LED
int pinLed = 11;

void setup(){
  // Serial.begin(rate) inizializzazione della seriale
  // Per maggiori informazioni si consulti il link: http://wp.me/p4kwmk-211
  Serial.begin(9600);

  // definizione di ledPin come output
  pinMode(pinLed, OUTPUT);
}
```

const

Consente di impostare una variabile in modo costante, il cui valore non potrà essere modificato in nessuna parte del programma.

A0

Costante per il pin analogico 1. E' un modo diverso per istanziare il primo pin analogico

continua...

```
void loop(){
  // analogRead leggerà il valore su A0 restituendo un valore tra 0 e 1023
  // per approfondimenti si consulti il link: http://wp.me/p4kwmk-1Qd
  val = analogRead(misura);

  // analogWrite() accetta come secondo parametro (PWM) valori tra 0 e 254
  // pertanto "rimappiamo" i valori letti da analogRead() nell'intervallo
  // tra 0 e 254 usando la funzione map
  // per approfondimenti si consulti il link: http://wp.me/p4kwmk-1Tu
  inputVal = map(val, 0, 1023, 0, 254);

  // la Serial.println(data) stampa i dati sulla porta seriale
  // (invia i dati alla seriale e li visualizza) seguito da un
  // ritorno a capo automatico ed un avanzamento linea.

  Serial.print("Valore letto dal sensore = "); // stampa ciò che è incluso tra ""
  Serial.println(val);                        // stampa val
  delay(2);                                   // attesa di 2 millisecondi per consentire la lettura
}
```

Provare a cambiare la quantità di luce che colpisce l'LDR e notare la variazione dei valori letti.

sketch21 | Arduino 1.0.5

```
sketch21 $  
/* Prof. Michele Maffucci  
15.03.2014  
  
Utilizzo di un LDR per controllare  
la luminosità di un LED  
Lettura valori sulla Serial Monitor
```

pin A0

Valore letto dal sensore = 163
Valore letto dal sensore = 170
Valore letto dal sensore = 170
Valore letto dal sensore = 168
Valore letto dal sensore = 167
Valore letto dal sensore = 169
Valore letto dal sensore = 171
Valore letto dal sensore = 171
Valore letto dal sensore = 169
Valore letto dal sensore = 67
Valore letto dal sensore = 50
Valore letto dal sensore = 47
Valore letto dal sensore = 46
Valore letto dal sensore = 46
v

Autoscroll Newline 9600 baud

```
void loop(){  
  // analogRead leggerà il valore su A0 restituendo un valore tra 0 e 1023  
  // per approfondimenti si consultì il link: http://wp.me/p4kwmk-10d  
  val = analogRead(misura);  
  
  // analogWrite() accetta come secondo parametro (PWM) valori tra 0 e 254
```

Done uploading.
Binary sketch size: 3,572 bytes (of a 32,256 byte maximum)

15 Arduino Uno on /dev/tty.usbmodem24161

1

Apertura Serial Monitor

2

Serial Monitor

Costante	Numero del pin
A0	Input analogico 0 (digitale 14)
A1	Input analogico 1 (digitale 15)
A2	Input analogico 2 (digitale 16)
A3	Input analogico 3 (digitale 17)
A4	Input analogico 4 (digitale 18)
A5	Input analogico 5 (digitale 19)
LED_BUILTIN	LED on-board (digitale 13)
SDA	I2C Data (digitale 18)
SCL	I2C Clock (digitale 19)
SS	SPI Select (digitale 10)
MOSI	SPI Input (digitale 11)
MISO	SPI Output (digitale 12)
SCL	SPI Clock (digitale 13)

Dalla versione dell'IDE di Arduino sono state introdotti nomi logici per la maggior parte dei pin.

Le costanti in tabella possono essere usate in tutte le funzioni che si attendono un numero di pin.

La tabella si riferisce alla scheda Arduino UNO R3.

La scheda Mega è dotata di molti più pin analogici e digitali.

Esercizio 1

Realizziamo un crepuscolare, ovvero un sistema che attiva l'accensione di un LED quando l'illuminazione scende al di sotto di un determinato valore.

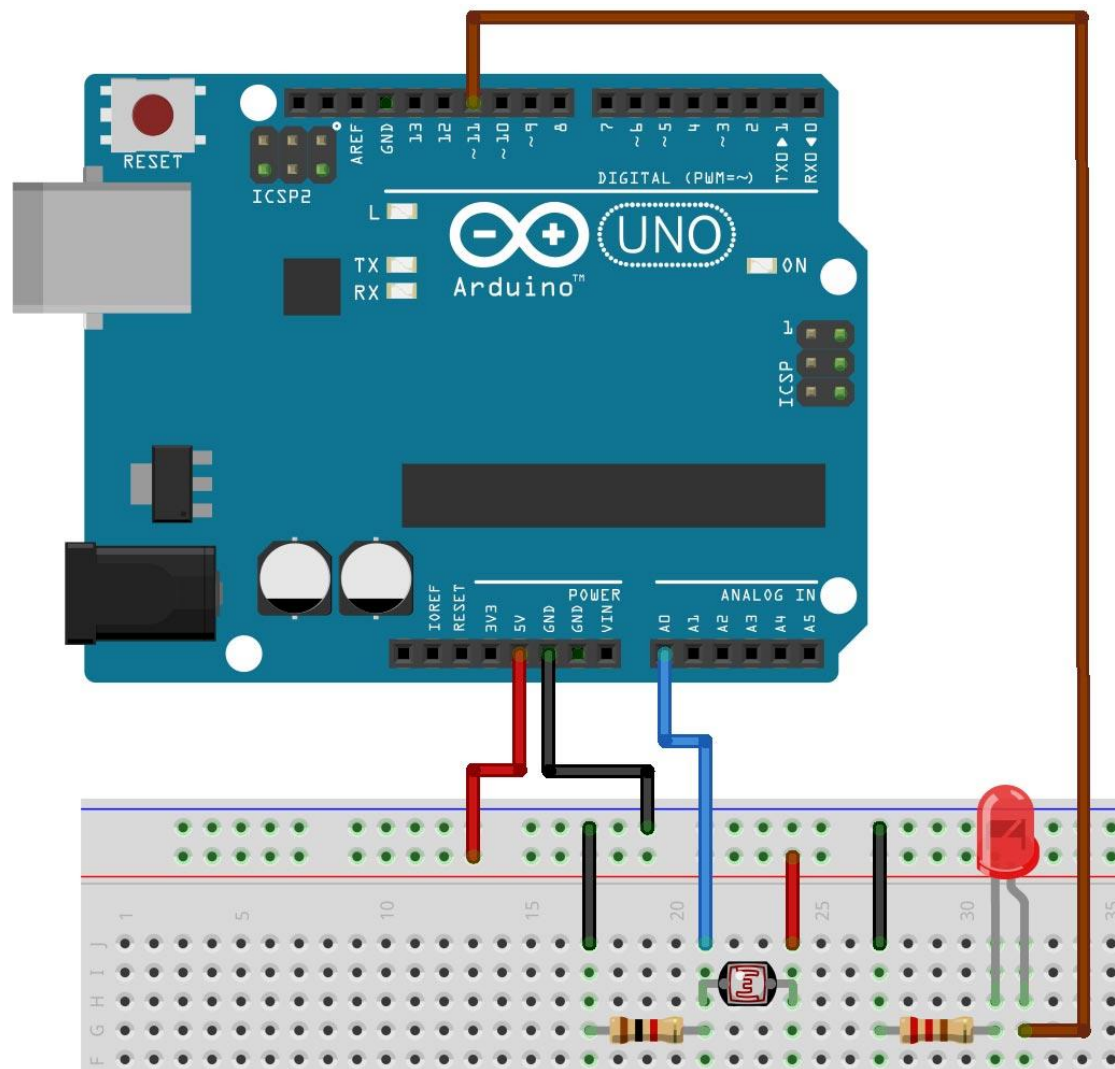
Si visualizzi sulla Serial Monitor i valori letti dal sensore (LDR) e lo stato del LED (acceso, spento).

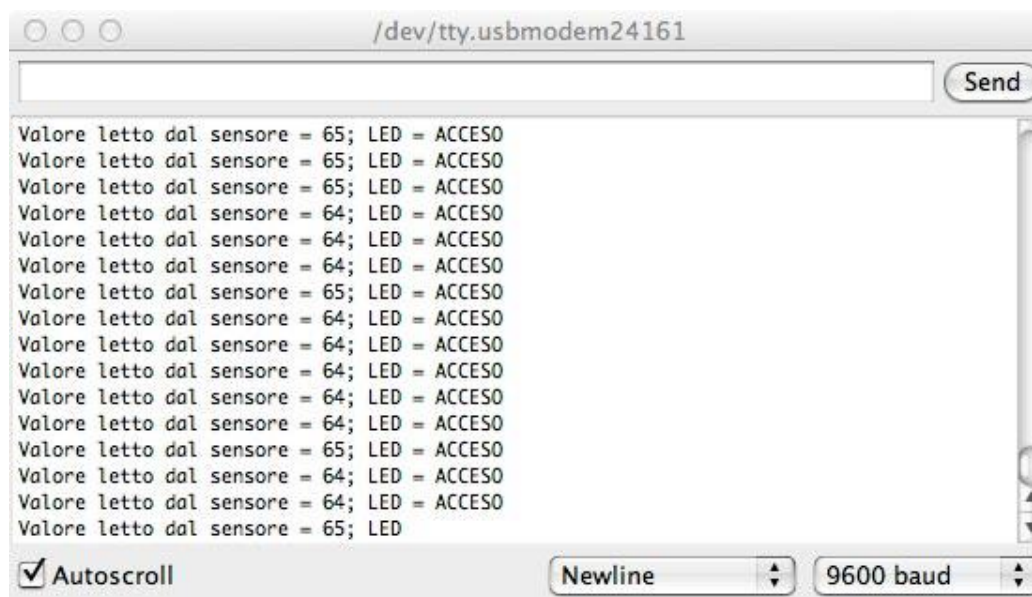
- Per valori letti del sensore **inferiori** a 250 il LED si **accende**.
- Per valori **maggiore** di 250 il LED si **spegne**.

Componenti:

- LDR
- R da 1 KOhm da mettere in serie all'LDR
- R in serie al LED da 220 Ohm
- LED

Il circuito è il medesimo dell'esempio precedente

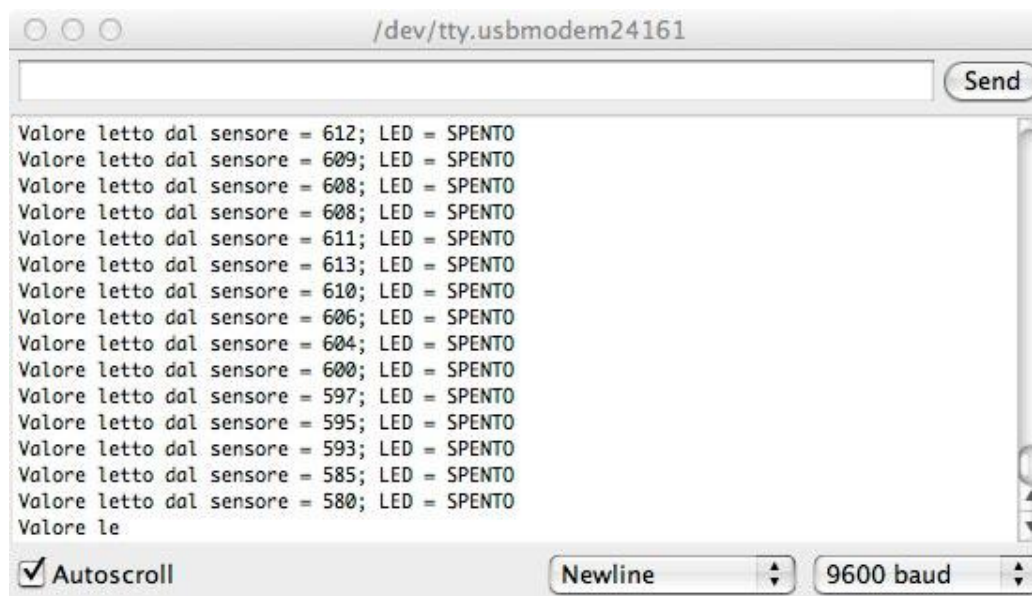




A terminal window titled "/dev/tty.usbmodem24161" showing a series of sensor readings. Each line displays a sensor value followed by "LED = ACCESO". The values are: 65, 65, 65, 64, 64, 64, 65, 64, 64, 64, 64, 65, 64, 64, 64, 65, 64, 64, 65. The "Autoscroll" checkbox is checked, and the baud rate is set to 9600.

```
/dev/tty.usbmodem24161  
Valore letto dal sensore = 65; LED = ACCESO  
Valore letto dal sensore = 65; LED = ACCESO  
Valore letto dal sensore = 65; LED = ACCESO  
Valore letto dal sensore = 64; LED = ACCESO  
Valore letto dal sensore = 64; LED = ACCESO  
Valore letto dal sensore = 64; LED = ACCESO  
Valore letto dal sensore = 65; LED = ACCESO  
Valore letto dal sensore = 64; LED = ACCESO  
Valore letto dal sensore = 64; LED = ACCESO  
Valore letto dal sensore = 64; LED = ACCESO  
Valore letto dal sensore = 64; LED = ACCESO  
Valore letto dal sensore = 65; LED = ACCESO  
Valore letto dal sensore = 64; LED = ACCESO  
Valore letto dal sensore = 64; LED = ACCESO  
Valore letto dal sensore = 65; LED  
Autoscroll Newline 9600 baud
```

Per valori letti del sensore **inferiori** a 250 il LED si **accende**.



A terminal window titled "/dev/tty.usbmodem24161" showing a series of sensor readings. Each line displays a sensor value followed by "LED = SPENTO". The values are: 612, 609, 608, 608, 611, 613, 610, 606, 604, 600, 597, 595, 593, 585, 580. The "Autoscroll" checkbox is checked, and the baud rate is set to 9600.

```
/dev/tty.usbmodem24161  
Valore letto dal sensore = 612; LED = SPENTO  
Valore letto dal sensore = 609; LED = SPENTO  
Valore letto dal sensore = 608; LED = SPENTO  
Valore letto dal sensore = 608; LED = SPENTO  
Valore letto dal sensore = 611; LED = SPENTO  
Valore letto dal sensore = 613; LED = SPENTO  
Valore letto dal sensore = 610; LED = SPENTO  
Valore letto dal sensore = 606; LED = SPENTO  
Valore letto dal sensore = 604; LED = SPENTO  
Valore letto dal sensore = 600; LED = SPENTO  
Valore letto dal sensore = 597; LED = SPENTO  
Valore letto dal sensore = 595; LED = SPENTO  
Valore letto dal sensore = 593; LED = SPENTO  
Valore letto dal sensore = 585; LED = SPENTO  
Valore letto dal sensore = 580; LED = SPENTO  
Valore le  
Autoscroll Newline 9600 baud
```

Per valori **maggiori** di 250 il LED si **spegne**.

Esercizio 2

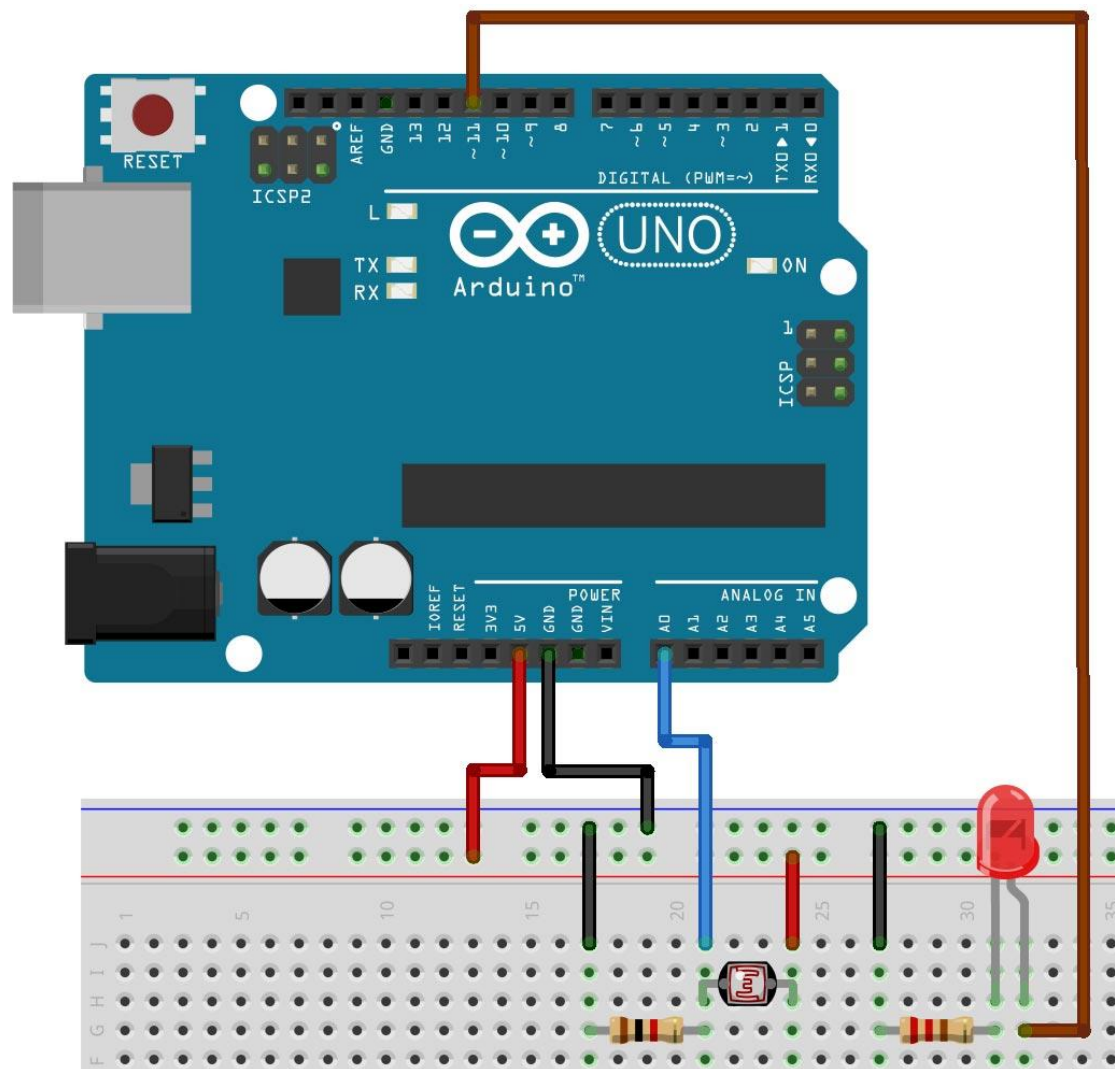
Realizziamo un crepuscolare ad accensione graduale, ovvero un sistema che attiva l'accensione graduale di un LED quando l'illuminazione scende al di sotto di un determinato valore.

Si visualizzi sulla Serial Monitor i valori letti dal sensore (LDR) e i valori convertiti usati come duty cycle per l'accensione graduale del LED

- Parametrizzare il sistema di rilevamento in modo che si possa fissare il valore massimo di luce dell'ambiente.
- Lo spegnimento del LED deve essere totale quando si è in massima condizione di luce ambientale (eliminare ogni oscillazione di accensione)

Componenti:

- LDR
- R da 1 KOhm da mettere in serie all'LDR
- R in serie al LED da 220 Ohm
- LED

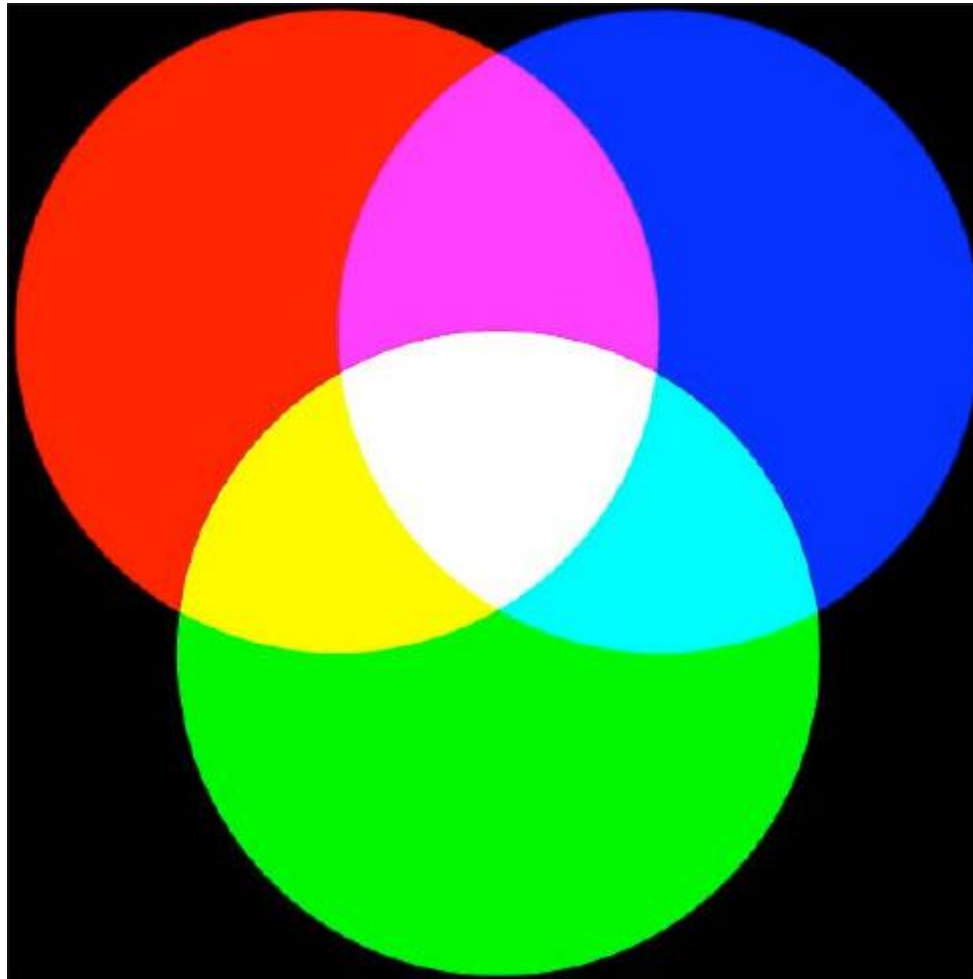


Il circuito è il medesimo dell'esercizio precedente

LED RGB

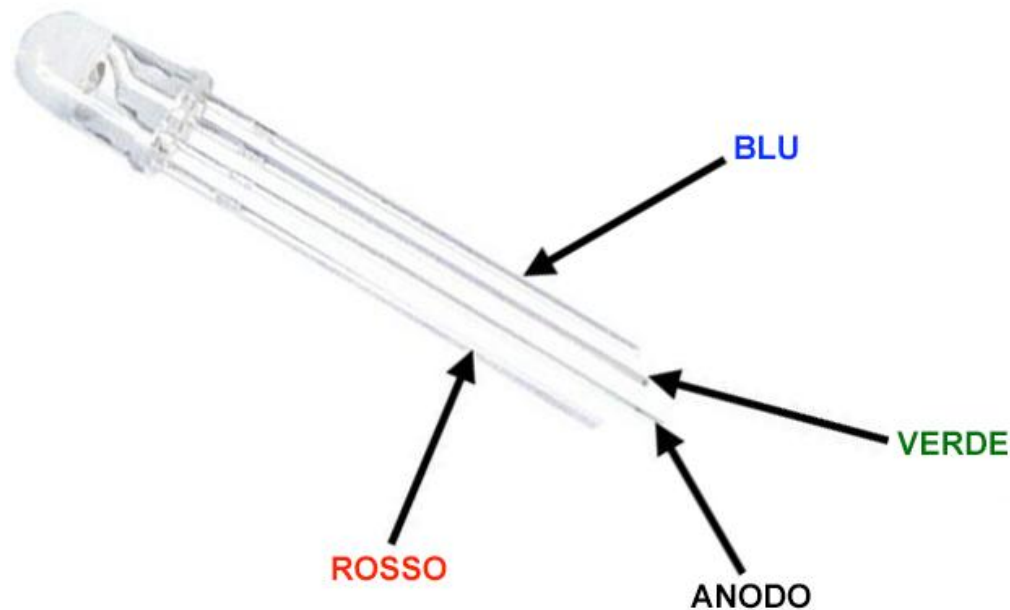
Funzionamento del LED RGB

La mescolanza dei tre colori dà luogo ad una luce di un determinato colore che dipende dall'intensità di ciascuno dei tre colori originari (si veda la composizione RGB)

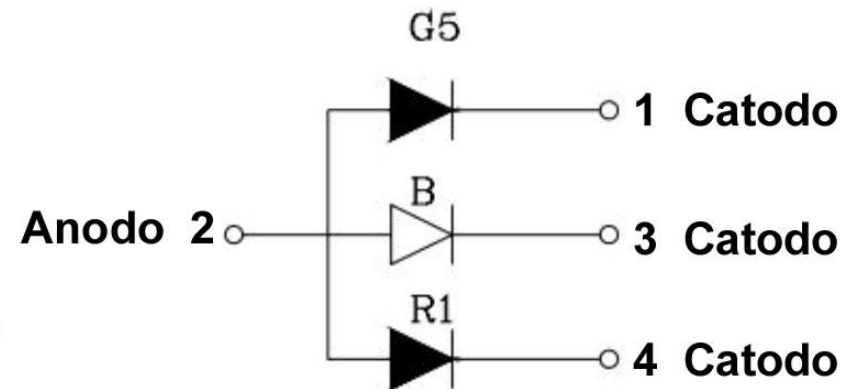


Funzionamento del LED RGB

Negli esempi che seguiranno sono stati utilizzati dei diodi ad **anodo comune**.



Rappresentazione elettrica del LED RGB ad anodo comune

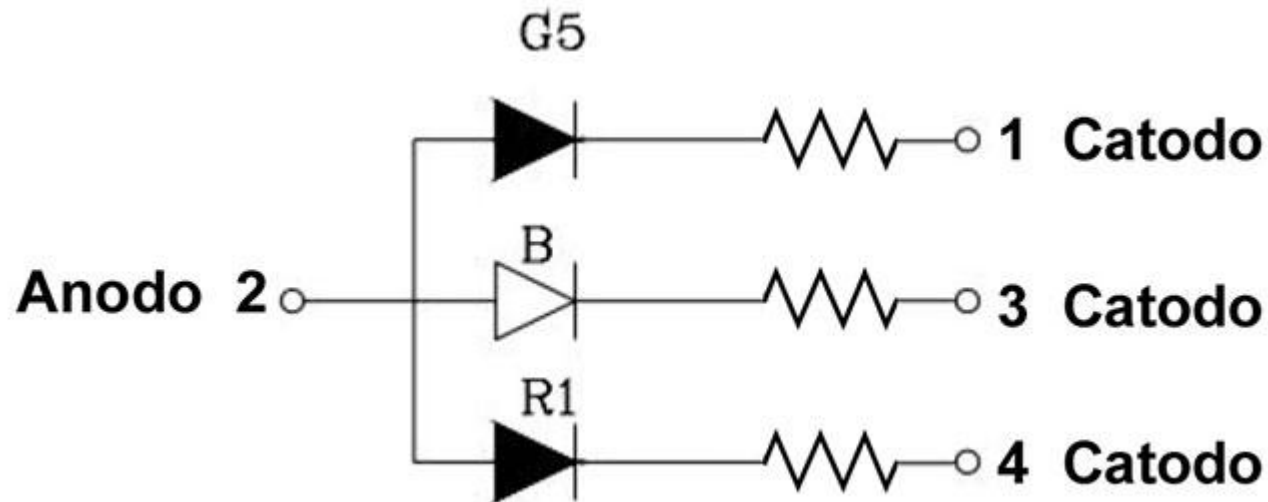


NOTA PER LA REALIZZAZIONE DEGLI ESPERIMENTI

Il LED RGB in dotazione con l'Arduino Starter Kit è a catodo comune, fare attenzione alle modalità diverse di connessione

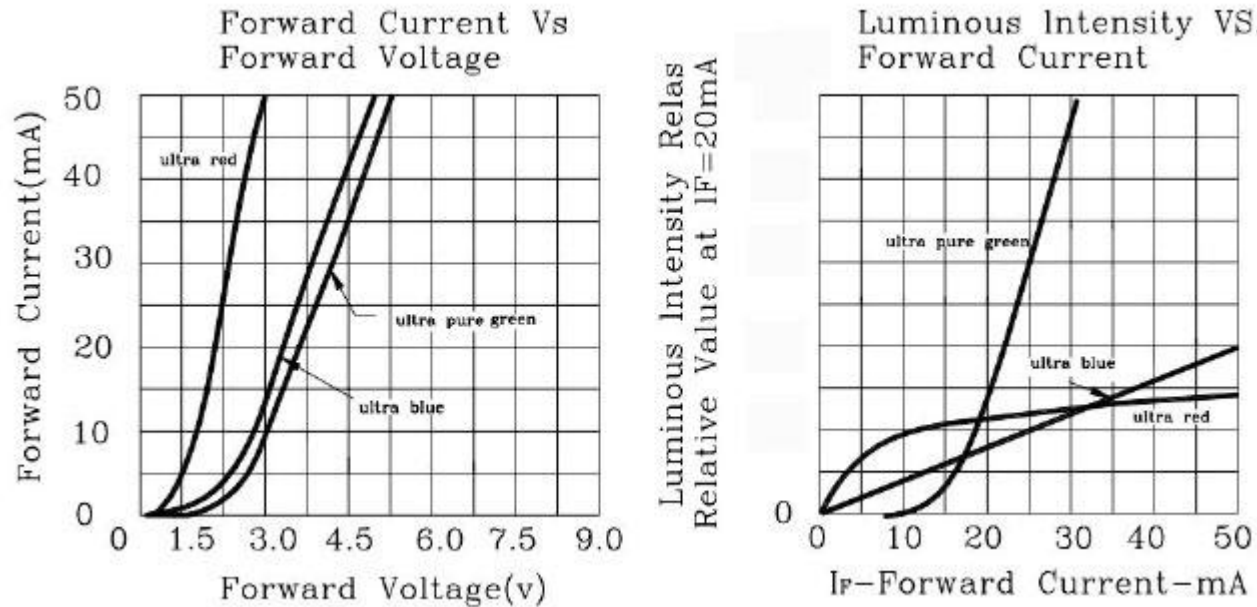
Funzionamento del LED RGB

In serie ad ogni LED sarà inserita una resistenza che consentirà di regolare la corrente circolante nel diodo.



Funzionamento del LED RGB

Dai datasheet si può notare come la caduta di tensione V_f , a parità di corrente nominale I_f sia diversa per ogni LED e la variabilità di V_f è piuttosto ampia, per questo motivo per effettuare i calcoli delle resistenze da porre in serie ai rispettivi LED bisognerà considerare un valore medio di V_f .

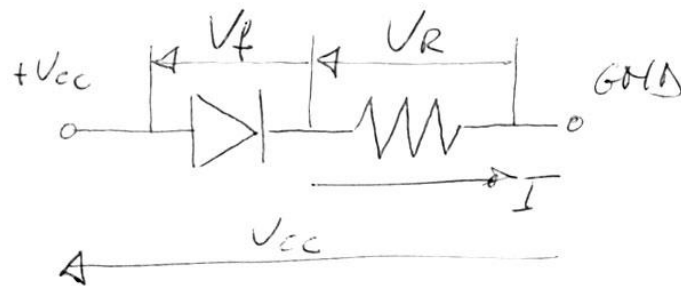


SYMBOL	PARAMETER	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
V _F	Forward Voltage	I _F =20mA	Ultra Red	2.0	2.6	V
			Ultra Pure Green	3.5	4.0	V
			Ultra Blue	3.5	4.0	V
I _R	Reverse Current	V _R =5V	Ultra Red		100	μA
			Ultra Pure Green		100	μA
			Ultra Blue		100	μA
λ _D	Dominant Wavelength	I _F =20mA	Ultra Red	625		nm
			Ultra Pure Green	525		nm
			Ultra Blue	460		nm

Funzionamento del LED RGB

La tensione di funzionamento dei diodi da considerare può essere letta in colonna TYP questi valori fanno riferimento ad una corrente diretta di 20 mA, usando questi valori di tensione siamo ora in grado di calcolare il valore della resistenza da porre in serie ai singoli diodi led.

Sapendo che la tensione di alimentazione sarà di 5V (tensione in uscita sul piedino digitale della scheda Arduino) e che su ogni singolo diodo led è da considerare una tensione tipica come da colonna TYP, sia avrà:



$$V_R = V_{cc} - V_f$$

$$R = \frac{V_{cc} - V_f}{I}$$

$$R_{ROSSO} = \frac{5 - 2}{0,02} = 150 \Omega$$

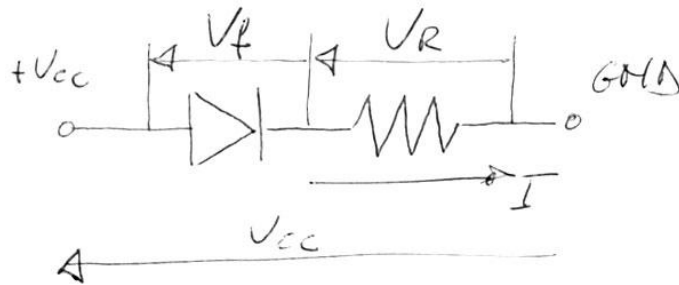
$$R_{VERDE} = R_{BLU} = \frac{5 - 3,5}{0,02} = 75 \Omega$$

Si sceglieranno dei valori commerciali di resistenza prossime a quelle calcolate. Poiché nella dotazione disponibile si hanno resistenze minime da 220 Ohm sceglieremo queste per gli esempi che seguiranno

Funzionamento del LED RGB

La tensione di funzionamento dei diodi da considerare può essere letta in colonna TYP questi valori fanno riferimento ad una corrente diretta di 20 mA, usando questi valori di tensione siamo ora in grado di calcolare il valore della resistenza da porre in serie ai singoli diodi led.

Sapendo che la tensione di alimentazione sarà di 5V (tensione in uscita sul piedino digitale della scheda Arduino) e che su ogni singolo diodo led è da considerare una tensione tipica come da colonna TYP, sia avrà:



$$V_R = V_{cc} - V_f$$

$$R = \frac{V_{cc} - V_f}{I}$$

$$R_{ROSSO} = \frac{5 - 2}{0,02} = 150 \Omega$$

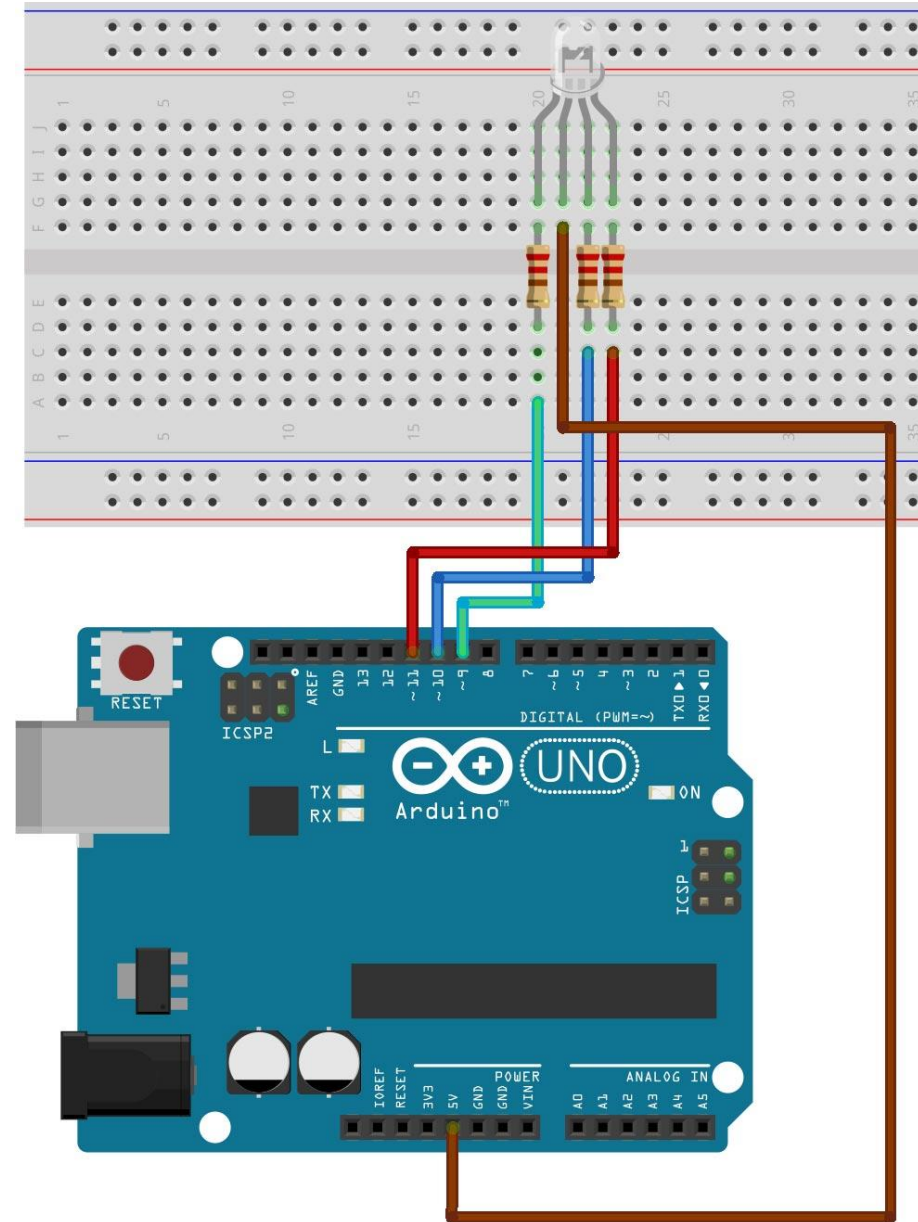
$$R_{VERDE} = R_{BLU} = \frac{5 - 3,5}{0,02} = 75 \Omega$$

Si sceglieranno dei valori commerciali di resistenza prossime a quelle calcolate. Poiché nella dotazione disponibile si hanno resistenze minime da 220 Ohm sceglieremo queste per gli esempi che seguiranno

Realizzare il circuito rappresentato in figura e implementare uno sketch in cui il **led verde si accende e si spegne gradualmente** mentre gli altri diodi restano spenti.

Componenti:

- led RGB
- 3 resistenze da da 220 Ohm da porre in serie ai catodi



Funzionamento del LED RGB

2/3

sketch24

```

/* Prof. Michele Maffucci
16.03.2014

Uso di un led RGB
Spegnimento graduale del LED Verde

Questo codice è di dominio pubblico
*/

// pin a cui collegare i piedini del LED RGB
const int VERDE = 9;
const int BLU = 10;
const int ROSSO = 11;

// tempo di transizione colore
const int delayTime = 20;

void setup() {

// imposta il pin digitale come output
pinMode(VERDE, OUTPUT);
pinMode(BLU, OUTPUT);
pinMode(ROSSO, OUTPUT);

// si impostano ad HIGH i pin VERDE, BLU, ROSSO
// inizialmente il led RGB sarà spento
digitalWrite(VERDE, HIGH);
digitalWrite(BLU, HIGH);
digitalWrite(ROSSO, HIGH);
}

// definizione di variabili globali
int ValVerde;

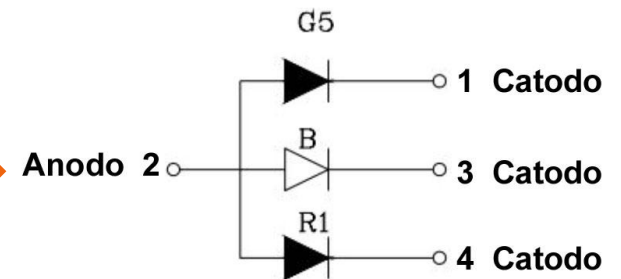
```

← Uso dei pin PWM 9, 10, 11

← led RGB spento →

continua..

Se i catodi, connessi ai pin 9, 10, 11, si trovano ad HIGH, poiché l'anodo si trova a +Vcc si avrà che i tre non vanno in conduzione, quindi non si accendono



```
void loop() {  
  // spegnimento graduale del verde  
  
  // coordinate RGB del rosso: 0, 255, 0  
  
  ValVerde = 255;  
  
  for( int i = 0 ; i < 255 ; i += 1 ){  
    ValVerde -= 1;  
  
    /* ad ogni ciclo la differenza  
    255 - ValVerde AUMENTA  
    provocando un graduale spegnimento del verde  
    */  
  
    analogWrite( VERDE, 255 - ValVerde );  
  
    // attesa di 20 ms per percepire il colore  
    delay( delayTime );  
  }  
}
```

Vengono eseguite 255 iterazioni e ad ogni ciclo l'indice `i` viene incrementato di 1. Ad ogni iterazione **ValVerde** viene decrementato di 1. Il led VERDE sarà al primo ciclo del for spento:

```
analogWrite(VERDE, 0)
```

nell'ultimo ciclo del for il led VERDE sarà acceso:

```
analogWrite(VERDE, 255)
```

Esercizio 3

Realizzare il circuito rappresentato in figura e implementare due sketch:

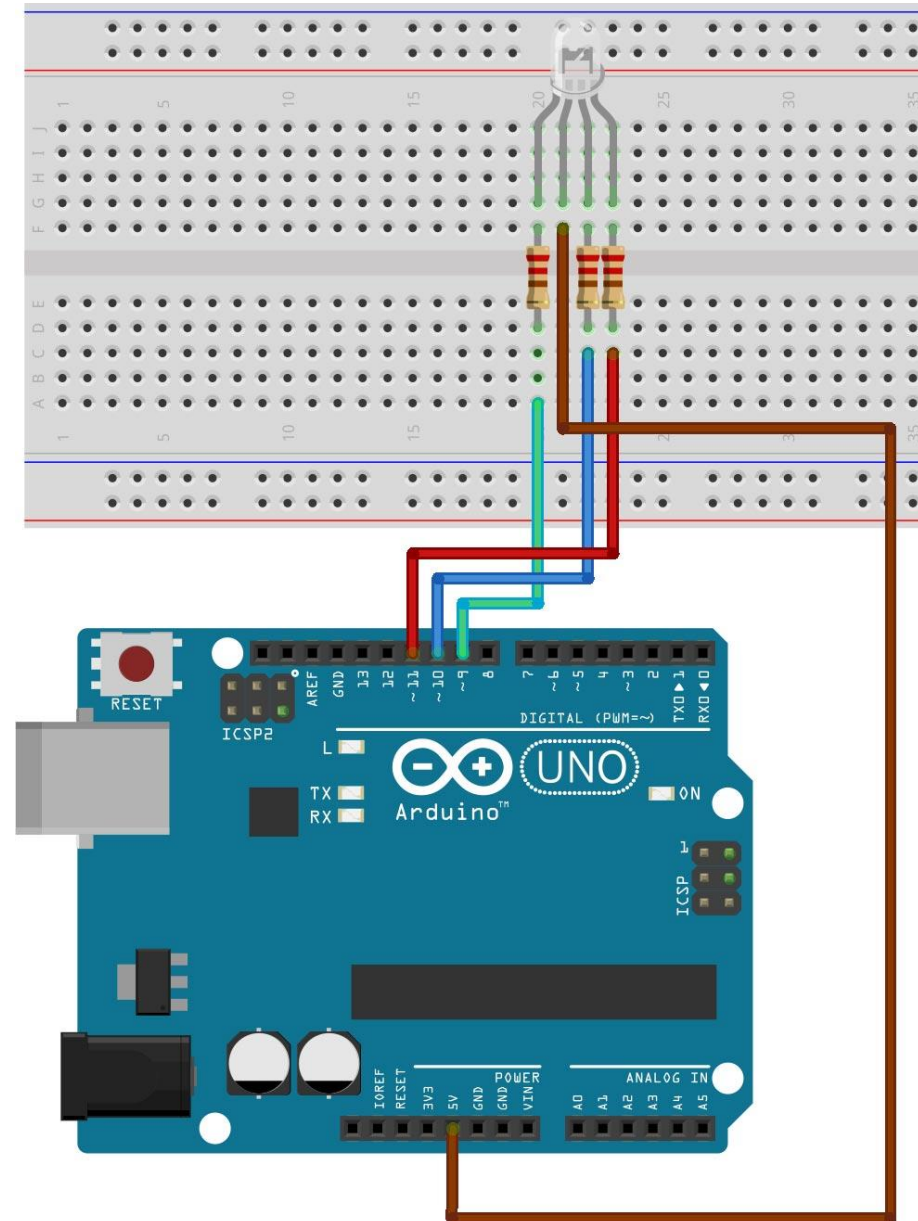
- led **rosso** si accende e si spegne gradualmente
- led **blu** si accende e si spegne gradualmente
- realizzare una sequenza di spegnimenti: verde, rosso, blu

Ricordare che

- le coordinate RGB del rosso sono: 255,0,0
- le coordinate RGB del blu sono: 0,0,255

Componenti:

- led RGB
- 3 resistenze da da 220 Ohm da porre in serie ai catodi



Realizzare il circuito rappresentato in figura e implementiamo lo sketch che consente di realizzare una variazione continua e graduale:

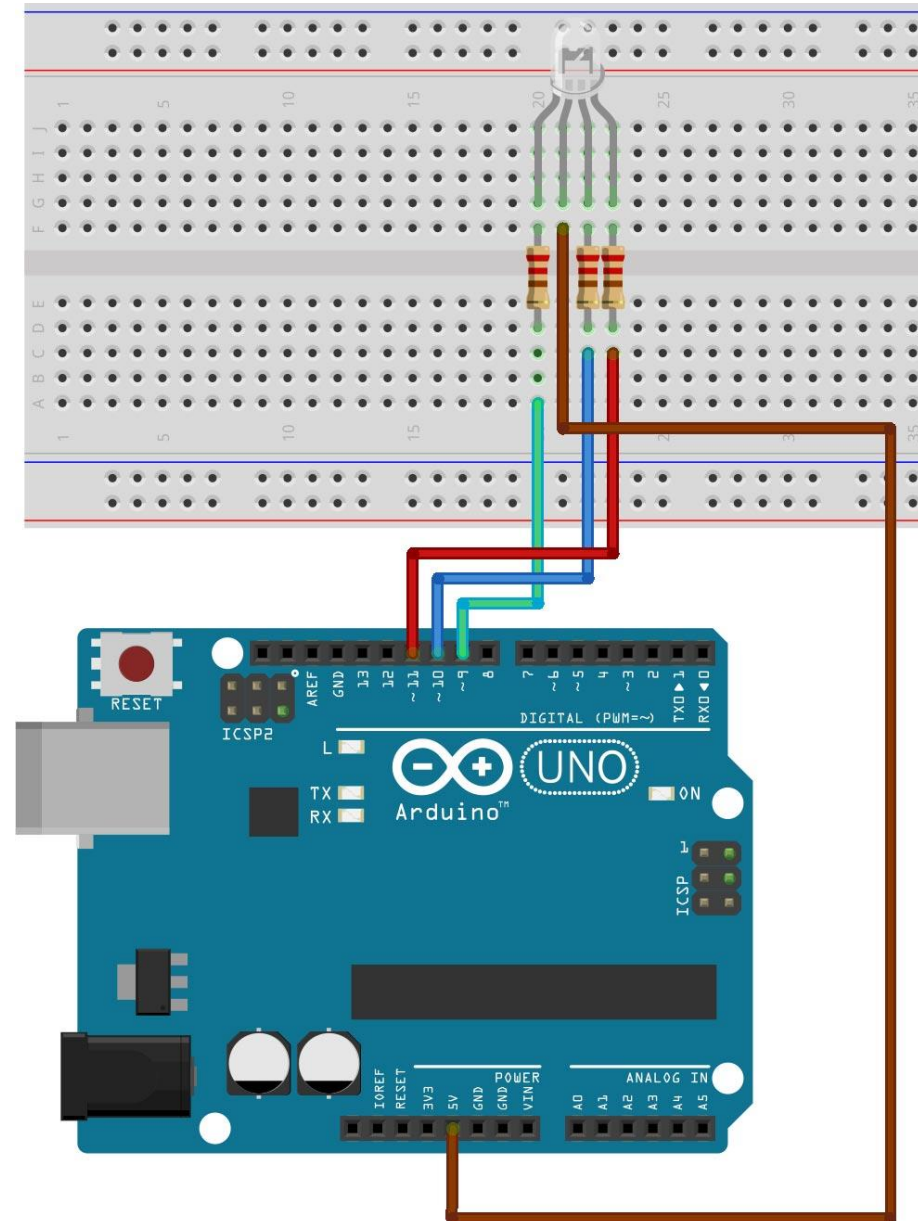
1. da **verde a rosso**
2. da **blu a verde**
3. da **rosso a blu**

Ricordare che:

- le coordinate RGB del rosso sono: 255,0,0
- le coordinate RGB del verde sono: 0,255,0
- le coordinate RGB del blu sono: 0,0,255

Componenti:

- led RGB
- 3 resistenze da da 220 Ohm da porre in serie ai catodi




```

/* Prof. Michele Maffucci
16.03.2014

Variazione colore continuo di un
LED RGB ad anodo comune

passaggi:
verde-rosso
blu-verde
rosso-blu

Questo codice è di dominio pubblico
*/

// pin a cui collegare i piedini del LED RGB
const int VERDE = 9;
const int BLU = 10;
const int ROSSO = 11;

// tempo di transizione colore
const int delayTime = 20;

void setup() {

  // imposta il pin digitale come output
  pinMode(VERDE, OUTPUT);
  pinMode(BLU, OUTPUT);
  pinMode(ROSSO, OUTPUT);

  // si impostano ad HIGH i pin VERDE, BLU, ROSSO
  // inizialmente il led RGB sarà spento
  digitalWrite(VERDE, HIGH);
  digitalWrite(BLU, HIGH);
  digitalWrite(ROSSO, HIGH);
}

// definizione di variabili globali
int ValRosso;
int ValBlu;
int ValVerde;

```

```

void loop() {

  // variazione da verde a rosso

  int ValRosso = 255;
  int ValBlu = 0;
  int ValVerde = 0;

  for( int i = 0 ; i < 255 ; i += 1 ){

    ValVerde += 1;
    ValRosso -= 1;

    /* ad ogni ciclo la differenza
    255 - ValVerde DIMINUISCE
    255 - ValRosso AUMENTA
    provocando un graduale passaggio
    dal verde al rosso
    */

    analogWrite( VERDE, 255 - ValVerde );
    analogWrite( ROSSO, 255 - ValRosso );

    // attesa di 20 ms per percepire il colore
    delay( delayTime );
  }
}

```

continua...

```
// variazione da blu al verde

ValRosso = 0;
ValBlu = 0;
ValVerde = 255;

ValRosso = 0;
ValBlu = 0;
ValVerde = 255;

for( int i = 0 ; i < 255 ; i += 1 ){

    ValBlu += 1;
    ValVerde -= 1;

    /* ad ogni ciclo la differenza
    255 - ValBlu DIMINUISCE
    255 - ValVerde AUMENTA
    provocando un graduale passaggio
    dal blu al verde
    */

    analogWrite( BLU, 255 - ValBlu );
    analogWrite( VERDE, 255 - ValVerde );

    // attesa di 20 ms per percepire il colore
    delay( delayTime );
}
}
```

```
// variazione da rosso al blu

ValRosso = 0;
ValBlu = 255;
ValVerde = 0;

for( int i = 0 ; i < 255 ; i += 1 ){

    ValRosso += 1;
    ValBlu -= 1;

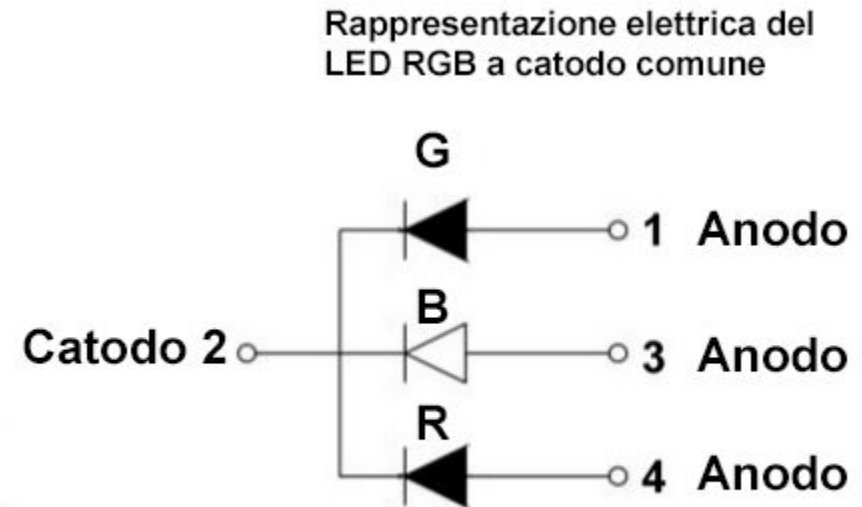
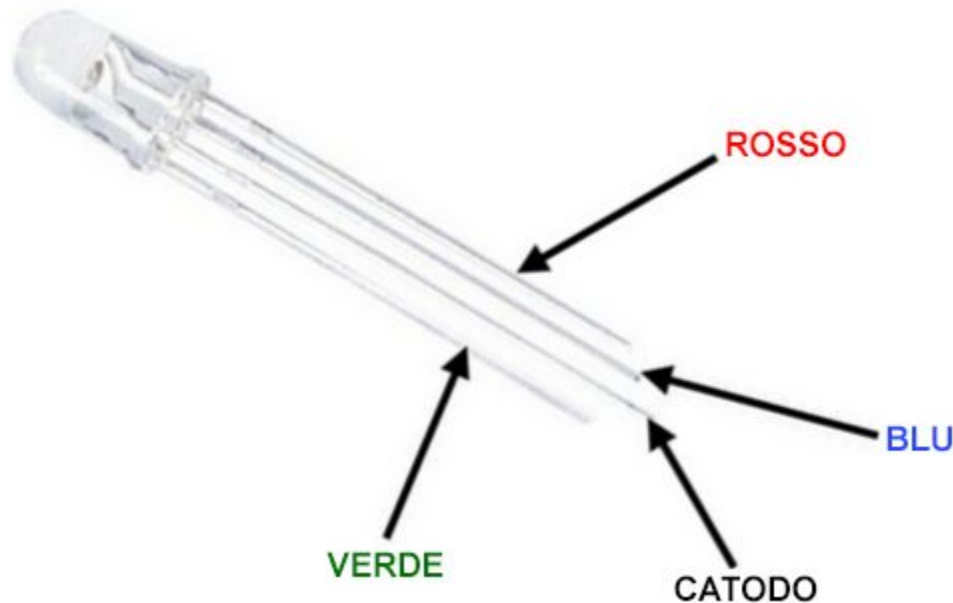
    /* ad ogni ciclo la differenza
    255 - ValRosso DIMINUISCE
    255 - ValBlu AUMENTA
    provocando un graduale passaggio
    dal rosso al blu
    */

    analogWrite( ROSSO, 255 - ValRosso );
    analogWrite( BLU, 255 - ValBlu );

    // attesa di 20 ms per percepire il colore
    delay( delayTime );
}
}
```

Funzionamento del LED RGB catodo comune

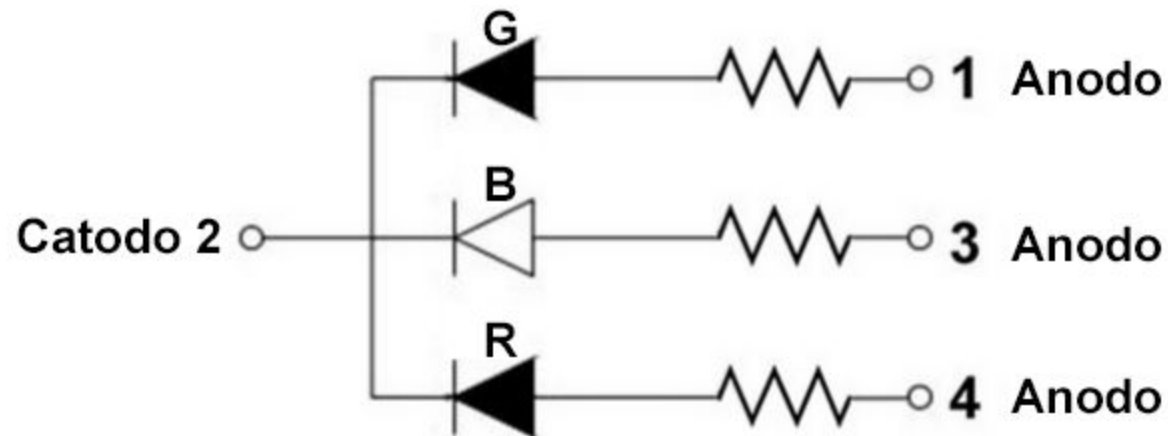
Per completezza si aggiungono gli schemi e gli sketch per l'utilizzo di un diodo RGB a **catodo comune**.



Nota: slide e codice aggiunto in fase successiva rispetto alla prima implementazione di questa presentazione al fine di permettere la sperimentazione con il componente fornito nell'Arduino Starter Kit.

Funzionamento del LED RGB catodo comune

In serie ad ogni LED sarà inserita una resistenza che consentirà di regolare la corrente circolante nel diodo.



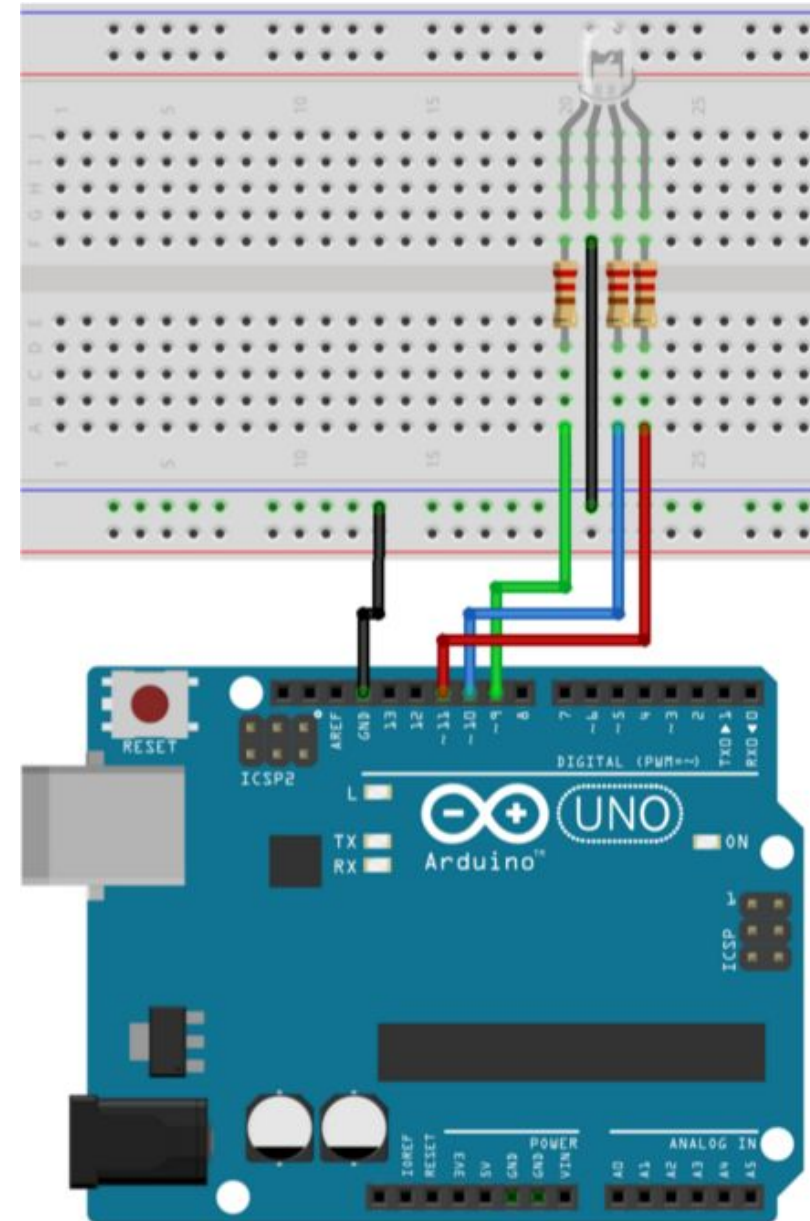
Variante che utilizza il LED RGB a catodo comune così come fornito nell'Arduino Starter Kit.

Componenti:

- led RGB catodo comune
- 3 resistenze da da 220 Ohm da porre in serie ai catodi

Si faccia attenzione che in questo caso il catodo deve essere collegato a GND, mentre gli anodi del LED RGB vanno connessi ai rispettivi pin di Arduino utilizzando gli stessi pin degli esempi precedenti.

Per gli sketch per la versione del LED RGB a catodo comune, si faccia riferimento al repository su [GitHub](#).



Da PC ad Arduino

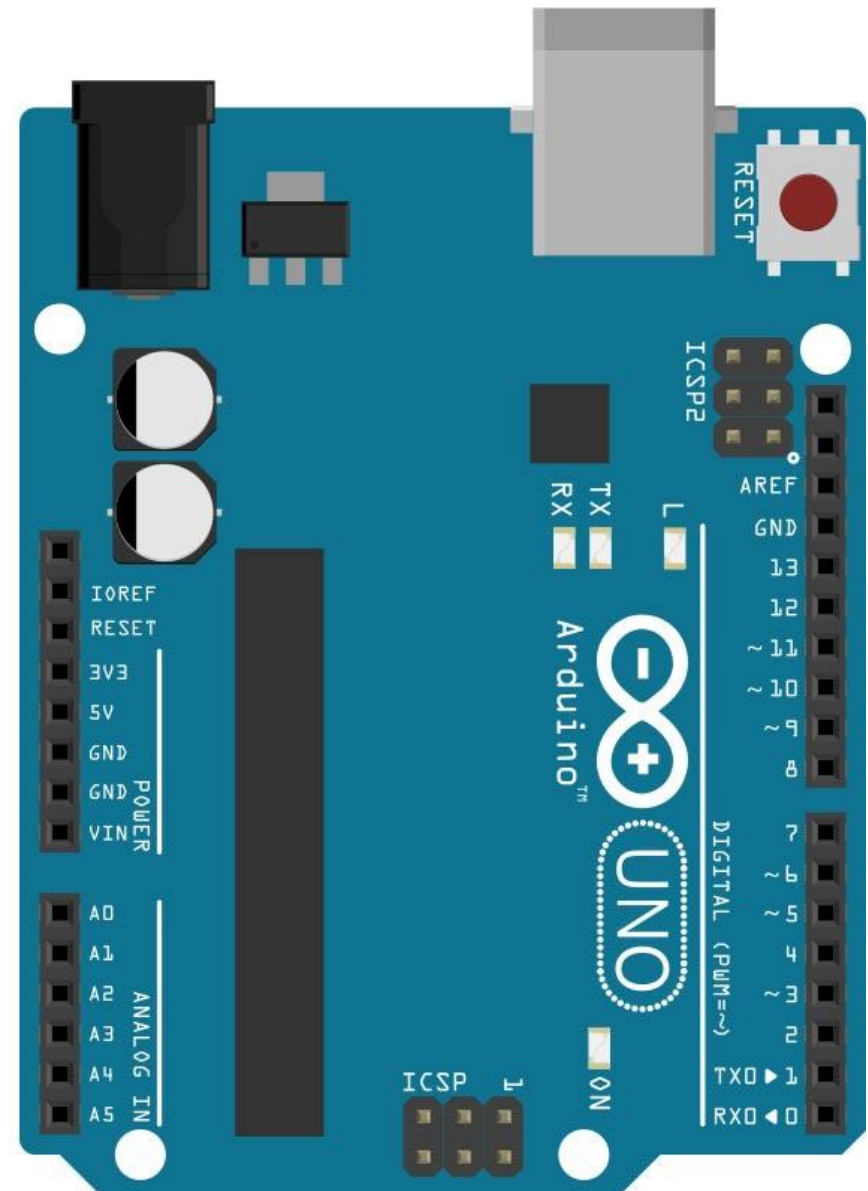
Si vogliono ricevere sulla scheda Arduino i dati provenienti dal computer o da altro dispositivo seriale.

Nello sketch si **riceve una cifra formata da un solo carattere** compreso tra 0 e 9, che farà lampeggiare il LED connesso al pin 13 ad una velocità proporzionale al valore della cifra ricevuta da Arduino.

Sulla Serial Monitor viene stampato il delay con cui lampeggia il LED.

Componenti

- Arduino




```

/* Prof. Michele Maffucci
16.03.2014

Uso della Serial.read() per inviare,
tramite la Serial Monitor, comandi ad Arduino.

Facciamo lampeggiare il led connesso al pin 13 ad una
velocità proporzionale al numero (da 0 a 9) inviato ad Arduino.
Viene stampato sulla Serial Monitor il delay con cui lampeggia
il LED

Questo codice è di dominio pubblico
*/

const int pinLed = 13; // il pin a cui è collegato il LED
int indiceBlink=0; // velocità con cui lampeggia il LED

void setup()
{
  Serial.begin(9600); // inizializzazione della porta seriale
  pinMode(pinLed, OUTPUT); // imposta il pin come output
}

void loop()
{
  if ( Serial.available() ) // Viene controllato se è disponibile un carattere
  {
    char ch = Serial.read(); // definizione di una variabile di tipo char in cui memorizzare
                             // il carattere inviato ad Arduino mediante la finestra Serial Monitor
    if( isDigit(ch) ) // si verifica se il carattere ASCII è un numero compreso tra 0 e 9
    {
      indiceBlink = (ch - '0'); // Il valore ASCII viene convertito in valore numerico
      indiceBlink = indiceBlink * 100; // il valore numerico viene moltiplicato per 100 millisecondi
    }
  }
  blink();
}

```

isDigit(ch)

Verifica se il carattere inserito è un numero compreso tra 0 e 9

indiceBlink = (ch - '0');

I caratteri ASCII da '0' a '9' hanno un valore compreso tra 48 e 57, quindi la conversione del carattere '1' nel valore numerico 1 viene fatta sottraendo il codice del carattere '1' corrispondente al codice ASCII 49 al codice del carattere '0' corrispondente al numero ASCII 48, ottenendo 49-48=1.

L'espressione (ch - '0') corrisponde all'espressione (ch - 48)

continua...

```
void loop()
{
  if ( Serial.available()           // Viene controllato se è disponibile un carattere
  {
    char ch = Serial.read();        // definizione di una variabile di tipo char in cui memorizzare
                                     // il carattere inviato ad Arduino mediante la finestra Serial Monitor
    if( isDigit(ch) )               // si verifica se il carattere ASCII è un numero compreso tra 0 e 9
    {
      indiceBlink = (ch - '0');      // Il valore ASCII viene convertito in valore numerico
      indiceBlink = indiceBlink * 100; // il valore numerico viene moltiplicato per 100 millisecondi
    }
  }
  blink();
}

// il led lampeggia con una tempo di accensione e spegnimento determinato da indiceBlink
void blink()
{
  Serial.print("Sto lampeggiando con un delay di: "); // stampa il testo compreso tra ""
  Serial.println(indiceBlink);                       // stampa indiceBlink e va a capo
  digitalWrite(pinLed,HIGH);
  delay(indiceBlink);                                // ritardo che dipende da indiceBlink
  digitalWrite(pinLed,LOW);
  delay(indiceBlink);
}
```

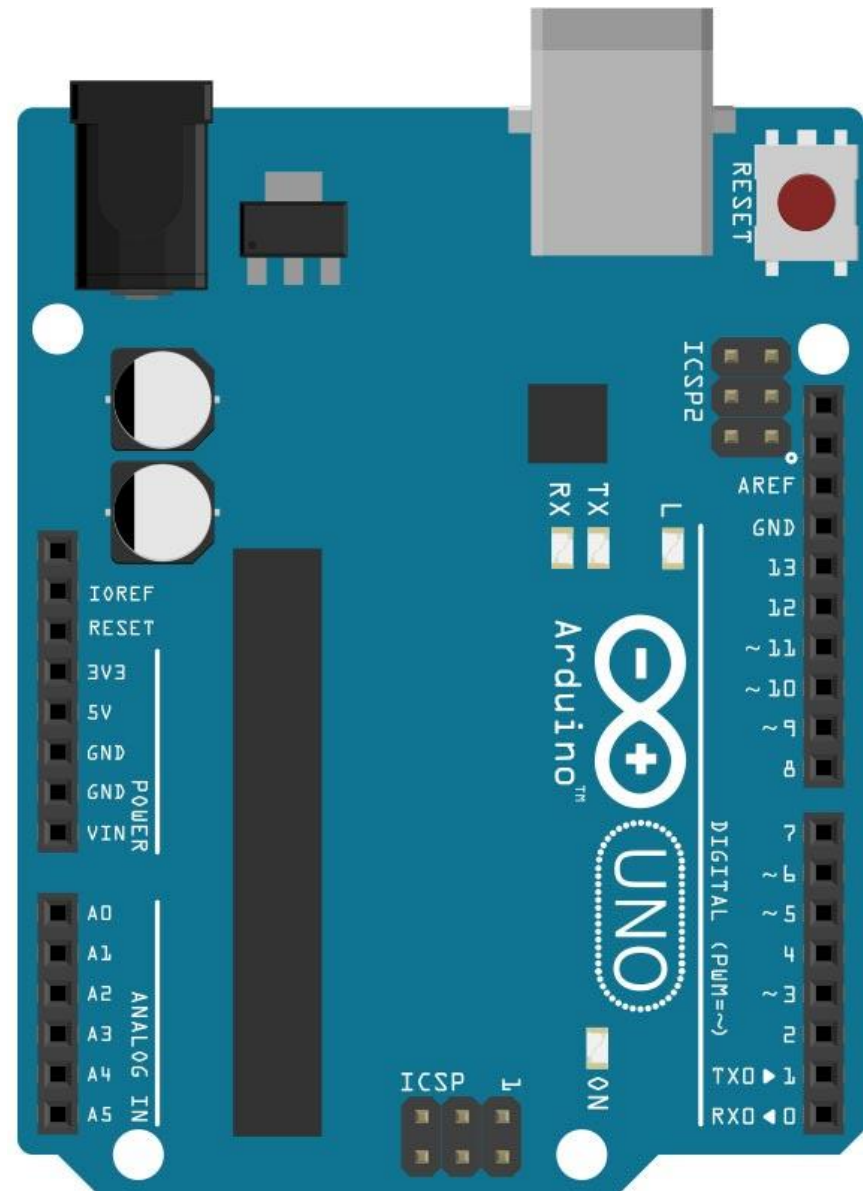
Si vogliono ricevere sulla scheda Arduino i dati provenienti dal computer o da altro dispositivo seriale.

Nello sketch si riceve **una cifra composta da un numero qualsiasi di caratteri**, che farà lampeggiare il LED connesso al pin 13 ad una velocità proporzionale al valore della cifra ricevuta da Arduino.

Sulla Serial Monitor viene stampato il delay con cui lampeggia il LED.

Componenti

- Arduino



uso della Serial.read()

2/3

sketch27

```
/* Prof. Michele Maffucci  
16.03.2014
```

```
Usa della Serial.read() per inviare,  
tramite la Serial Monitor, comandi ad Arduino.
```

```
Facciamo lampeggiare il led connesso al pin 13 ad una  
velocità proporzionale al numero inviato ad Arduino.  
Viene stampato sulla Serial Monitor il delay con cui lampeggia  
il LED
```

```
Questo codice è di dominio pubblico
```

```
*/
```

```
const int pinLed = 13;    // il pin a cui è collegato il LED  
int indiceBlink=0;       // velocità con cui lampeggia il LED  
int valore = 0;  
  
void setup()  
{  
  Serial.begin(9600);    // inizializzazione della porta seriale  
  pinMode(pinLed, OUTPUT); // imposta il pin come output  
}
```

[continua...](#)

```

void loop()
{
  if ( Serial.available() ) // Viene controllato se c'è qualcosa da leggere
  {
    char ch = Serial.read(); // definizione di un carattere
                              // il carattere viene letto e memorizzato in ch
    if( isDigit(ch) ) // si verifica se il carattere è un numero
    {
      valore = (valore*10) + (ch - '0'); // calcolo per accumulare il valore che si sta digitando
                                          // es. inserendo 276
                                          // valore1 = 0*10+50-48=2
                                          // valore2 = 2*10+55-48=27
                                          // valore3 = 27*10+54-48=276
    }
    // l'invio da tastiera corrisponde codice ASCII 10
    // vuol dire che abbiamo terminato la scrittura del numero ed abbiamo premuto sull'invio
    else if (ch == 10) // il valore del carattere è uguale a 10
    {
      indiceBlink = valore; // il valore del numero digitato
      Serial.println(indiceBlink); // stampa sulla porta seriale
      valore=0; // si reimposta il valore a 0
    }
  }
  blink();
}

// il led lampeggia con una tempo di accensione e spegnimento determinato da indiceBlink
void blink()
{
  Serial.print("Sto lampeggiando con un delay di: "); // stampa il testo compreso tra ""
  Serial.println(indiceBlink); // stampa indiceBlink e va a capo
  digitalWrite(pinLed,HIGH); // accende il led
  delay(indiceBlink); // ritardo che dipende da indiceBlink
  digitalWrite(pinLed,LOW); // spegne il led
  delay(indiceBlink); // ritardo che dipende da indiceBlink
}

```

valore = (valore*10) + (ch - '0');

calcolo per accumulare il valore che si sta digitando

es. inserendo 276

valore1 = 0*10+50-48=2

valore2 = 2*10+55-48=27

valore3 = 27*10+54-48=276

ch == 10

l'invio da tastiera (return) corrisponde codice ASCII 10

vuol dire che abbiamo terminato la scrittura del numero ed abbiamo premuto sull'invio.

continua...

Si vuole realizzare un voltmetro per misure di tensioni non superiori a 5V.

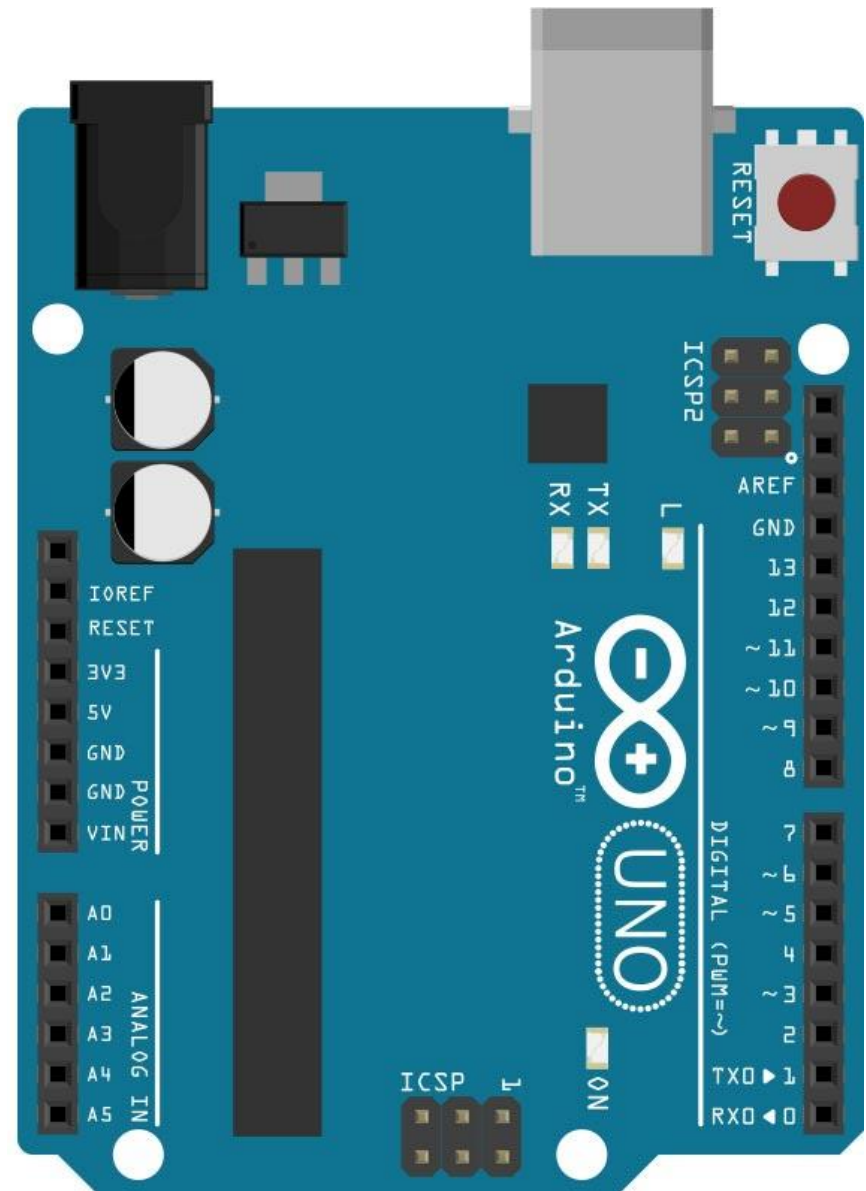
Usare un carattere di controllo per avviare la lettura ed un altro per interrompere la lettura

Quando la misurazione è attiva accendere il LED connesso al pin 13, quando si disattiva la misurazione il LED deve spegnersi.

Sulla Serial Monitor viene stampato il valore della tensione misurata.

Componenti

- Arduino



uso della Serial.read()

2/3

sketch28

```
/* Prof. Michele Maffucci  
18.03.2014
```

```
Realizziamo un voltmetro  
per tensioni non superiori a 5V
```

```
Utilizzare il numero 0 per spegnere il voltmetro  
ed il LED connesso al pin 13
```

```
Utilizzare il numero 1 per visualizzare la tensione  
ed accendere il LED connesso al pin 13
```

```
ATTENZIONE non collegare tensioni superiori  
a 5 V sui pin di Arduino
```

```
Questo codice è di dominio pubblico
```

```
*/
```

```
// tensione di riferimento  
const float voltRiferimento = 5.0;
```

```
// pin a cui è connessa la batteria  
const int pinBatteria = A0;
```

```
// il pin a cui è collegato il LED  
const int pinLed = 13;
```

continua...


```

void setup()
{
  Serial.begin(9600); // inizializzazione della porta seriale
  pinMode(pinLed, OUTPUT); // imposta il pin come output
}

void loop()
{
  if ( Serial.available() ) // Viene controllato se è disponibile un carattere
  {
    char ch = Serial.read(); // definizione di una variabile di tipo char in cui memorizzare
    // il carattere inviato ad Arduino mediante la finestra Serial Monitor
    if( isDigit(ch) ) // si verifica se il carattere ASCII è un numero compreso tra 0 e 9
    {
      if (ch=='0')
      {
        Serial.println("Voltmetro spento");
        digitalWrite(pinLed, LOW);
      }
      if ( ch=='1' )
      {
        digitalWrite(pinLed, HIGH);
        int val = analogRead(pinBatteria);

        // calcola la proporzione
        // il valore restituito da analogRead() è un
        // 0 e 1024 pertanto ciascuna unità vale 5/1024 = 4,88 mV
        // da cui valore misurato (volt) = val * 4,88 mV
        float volt = val * (voltRiferimento/1024.0);
        Serial.print(volt); // stampa il valore
        Serial.println(" V"); // stampa il valore
      }
    }
  }
}

```

$$\text{volt} = \text{val} * (\text{voltRiferimento}/1024.0);$$

Calcola la proporzione

Il valore restituito da analogRead() è un numero compreso tra 0 e 1023 pertanto ciascuna unità vale $5/1023 = 4,88$ mV da cui valore misurato (volt) = val * 4,88 mV

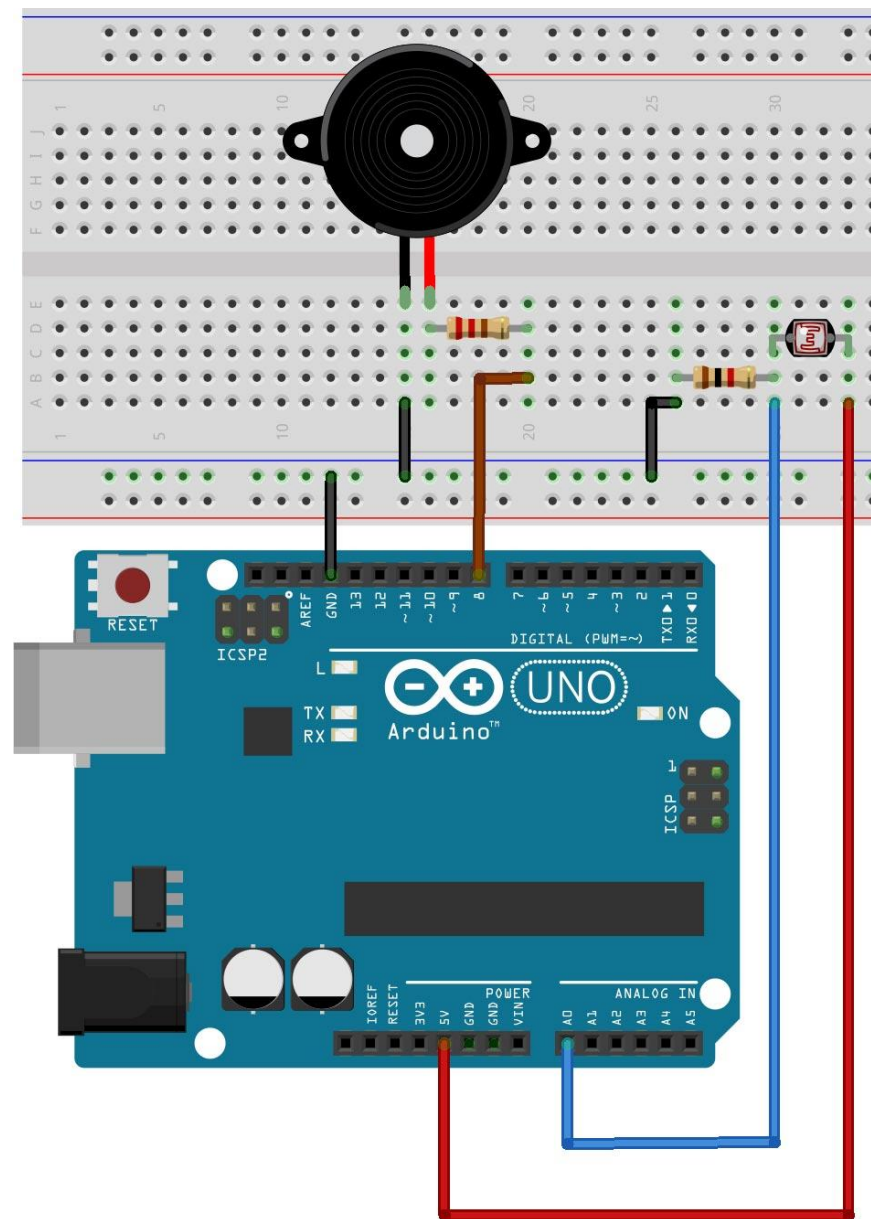


Musica

Si vuole realizzare Theremin comandato dalla luce, ovvero uno strumento in grado di generare un tono in funzione della quantità di luce che colpisce un LDR.

Componenti

- LDR
- piezo
- R da 1KOhm da porre in serie all' LDR
- R da 220 Ohm da porre in serie al piezo



```
// variabile usata per calibrare il valore minimo
int valoreBasso = 1023;

// variabile usata per calibrare il valore massimo
int valoreAlto = 0;

// il pin a cui è collegato il LED
const int pinLed = 13;

void setup()
{
  pinMode(pinLed, OUTPUT); // imposta il pin come output

  // viene segnalato che incomincia la fase di calibrazione
  digitalWrite(pinLed, HIGH);
  // calibrazione per i primi cinque secondi dopo l'avvio del programma
  // millis() Restituisce il numero di millisecondi da quando la scheda
  // Arduino ha incominciato l'esecuzione del programma corrente.
  // Il tipo di dato è un unsigned long.
  // Nota: questo valore va in overflow (supera il limite
  // per cui ricomincia da zero dopo circa 9 ore.

  // per calibrare muovere la mano sopra il sensore

  while (millis() < 5000) {
    // registra il massimo valore rilevato
    valoreSensore = analogRead(A0);
    if (valoreSensore > valoreAlto) {
      valoreAlto = valoreSensore;
    }
    // registra il valore minimo rilevato
    if (valoreSensore < valoreBasso) {
      valoreBasso = valoreSensore;
    }
  }

  // spegne il LED collegato al pin 13, in questo modo
  // si segnala che è terminata la fase di calibrazione
  digitalWrite(pinLed, LOW);
}
```

while (millis() < 5000) ...

la funzione millis() restituisce il numero di millisecondi da quando la scheda Arduino è stata messa in funzione. Il corpo del while verrà eseguito fino a quando non si raggiungono i 5 secondi di funzionamento di Arduino.

Nel corpo del while vengono stabiliti i valori massimi e minimi di quantità di luce rilevati dal sensore.

Per approfondimenti seguire il [link](#).

continua...

uso di tone()


3/3

sketch29

```
void loop() {  
  // legge il valore da A0 e lo memorizza nella variabile  
  valoreSensore = analogRead(A0);  
  
  // mappa i valori letti dal sensore nell'intervallo 50, 4000  
  int tono = map(valoreSensore, valoreBasso, valoreAlto, 50, 4000);  
  
  // suona un tono per 20 millisecondi sul pin 8  
  // la funzione tone(pin, frequenza, durata) ha tre argomenti:  
  // pin: il piedino su cui inviare il tono  
  // frequenza: frequenza del tono emesso  
  // durata: durata in millisecondi del tono emesso  
  tone(8, tono, 20);  
  
  // attesa di 10 millisecondi  
  delay(10);  
}
```

map(...)

valori rimappati nell'intervallo 50,4000 campo di frequenza udibile.



tone(pin, frequenza, durata)



la funzione tone(pin, frequenza, durata) ha tre argomenti:

pin: il piedino su cui inviare il tono

frequenza: frequenza del tono emesso

durata: durata in millisecondi del tono emesso

uso di `tone()`

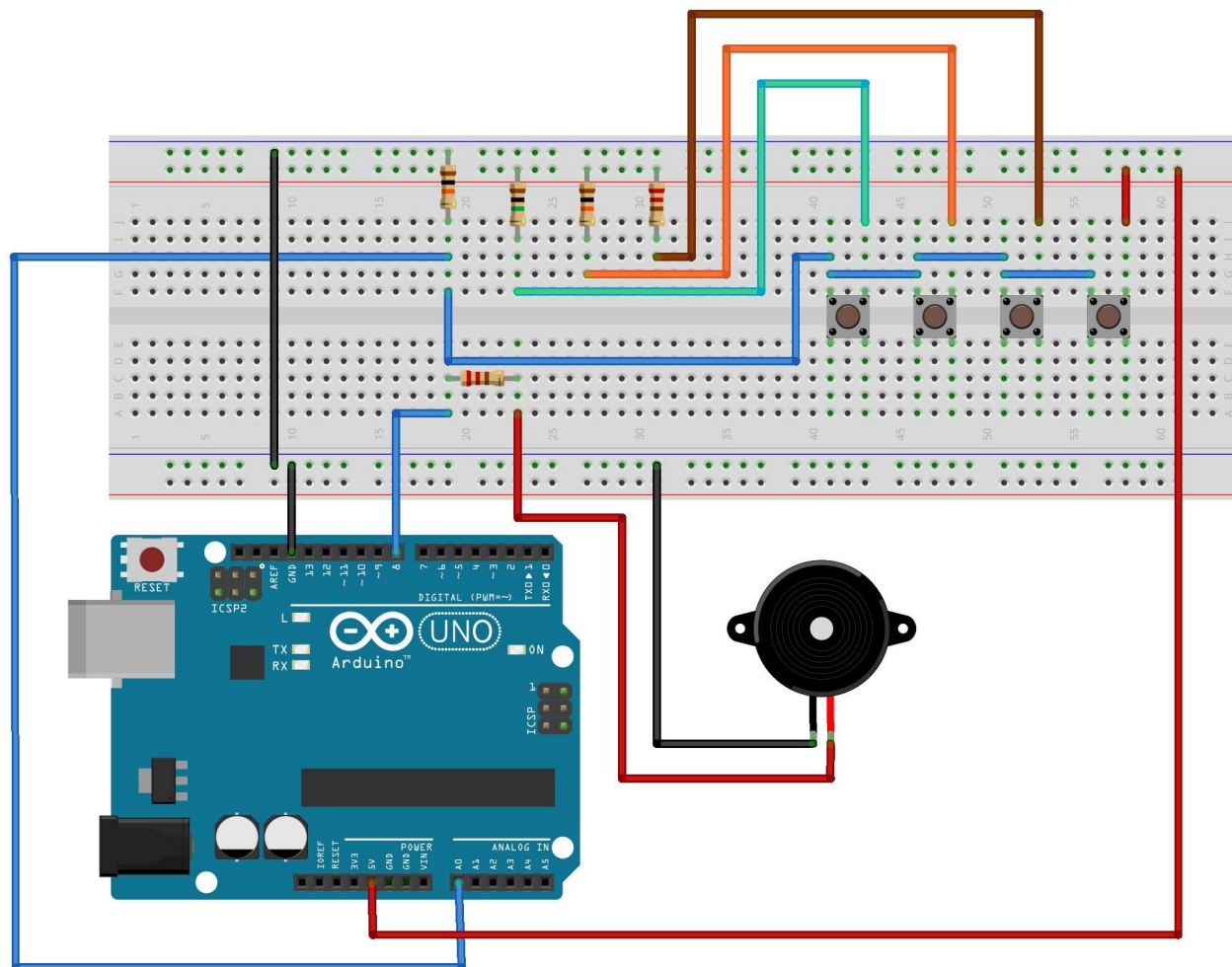
1/2

sketch30

Tastiera musicale a tre tasti. In funzione del partitore di tensione selezionate viene emessa una nota musicale.

Componenti

- n. 2 R da 220 Ohm
- n. 1 R da 1 M Ohm
- n. 2 R da 10 KOhm
- n. 1 piezo



```
/* Prof. Michele Maffucci
18.03.2014

Realizzazione Theremin comandato
dalla luce.

Tratto dall'esempio: Examples > 10.StarterKit > p07_Keyboard

Questo codice è di dominio pubblico
*/

// array delle frequenze delle note
// c, d, e, f
int notes[] = {262, 294, 330, 349};

int pinPiezzo = 9;

void setup() {
  // inizializzazione seriale
  Serial.begin(9600);
  pinMode(pinPiezzo, OUTPUT);
}
```

```
void loop() {
  // variabile locale in cui memorizzare i valori sul pin A0
  int keyVal = analogRead(A0);
  // stampa i valori di A0 sulla Serial Monitor
  Serial.println(keyVal);

  // suona la nota corrispondente a ciascun valore di A0
  if(keyVal == 1023){
    // emette la prima nota sul pin 8 presente nell'array
    tone(pinPiezzo, notes[0]);
  }
  else if(keyVal >= 990 && keyVal <= 1010){
    // emette la seconda nota sul pin 8 presente nell'array
    tone(pinPiezzo, notes[1]);
  }
  else if(keyVal >= 505 && keyVal <= 515){
    // emette la terza nota sul pin 8 presente nell'array
    tone(pinPiezzo, notes[2]);
  }
  else if(keyVal >= 5 && keyVal <= 10){
    // emette la quarta nota sul pin 8 presente nell'array
    tone(pinPiezzo, notes[3]);
  }
  else{
    // se i valori sono fuori dall'intervallo non emette suono
    noTone(pinPiezzo);
  }
}
```


Grazie

Prof. Michele Maffucci

www.maffucci.it

michele@maffucci.it

www.twitter.com/maffucci/

www.facebook.com/maffucci.it/

plus.google.com/+MicheleMaffucci/

it.linkedin.com/in/maffucci

Licenza presentazione:

