ESPERTO IN UN CLICK

Mattia Valsania

Corso pratico di ARDUINO

MODULO BASE

area51 Publishing

Mattia Valsania

Corso pratico di Arduino

Modulo base

© 2015 Area 51 s.r.l., San Lazzaro di Savena (Bologna)

Prima edizione e-book Area51 Publishing: gennaio 2015

Curatore della collana: Mirco Baragiani

Cover: Valerio Monego

Redazione e sviluppo ebook: Enrico De Benedictis

Se intendi condividere questo ebook con un'altra persona, ti chiediamo cortesemente di scaricare una copia a pagamento per ciascuna delle persone a cui lo vuoi destinare. Se stai leggendo questo ebook e non lo hai acquistato, ti chiediamo, se ti piace, di acquistarne anche una copia a pagamento, al fine di poterci permettere di far crescere il nostro lavoro e di offrirti sempre più titoli e una qualità sempre maggiore. Grazie per il tuo aiuto e per aver rispettato il lavoro dell'autore e dell'editore di questo libro.

Segui **Area51 Publishing** su:

 **Google Plus**

 **Facebook**

 **Twitter**

 **Pinterest**

www.area51editore.com

www.area51editore.com/blog

Introduzione

Arduino è una piccola scheda elettronica open source dotata di un microcontrollore, usata nei prototipi hobbistici e didattici.

Che cosa potrai fare con Arduino?

Con Arduino si possono realizzare in modo rapido piccoli progetti come comandare luci, regolare la velocità dei motori, leggere sensori, comandare attuatori e comunicare con altri dispositivi.

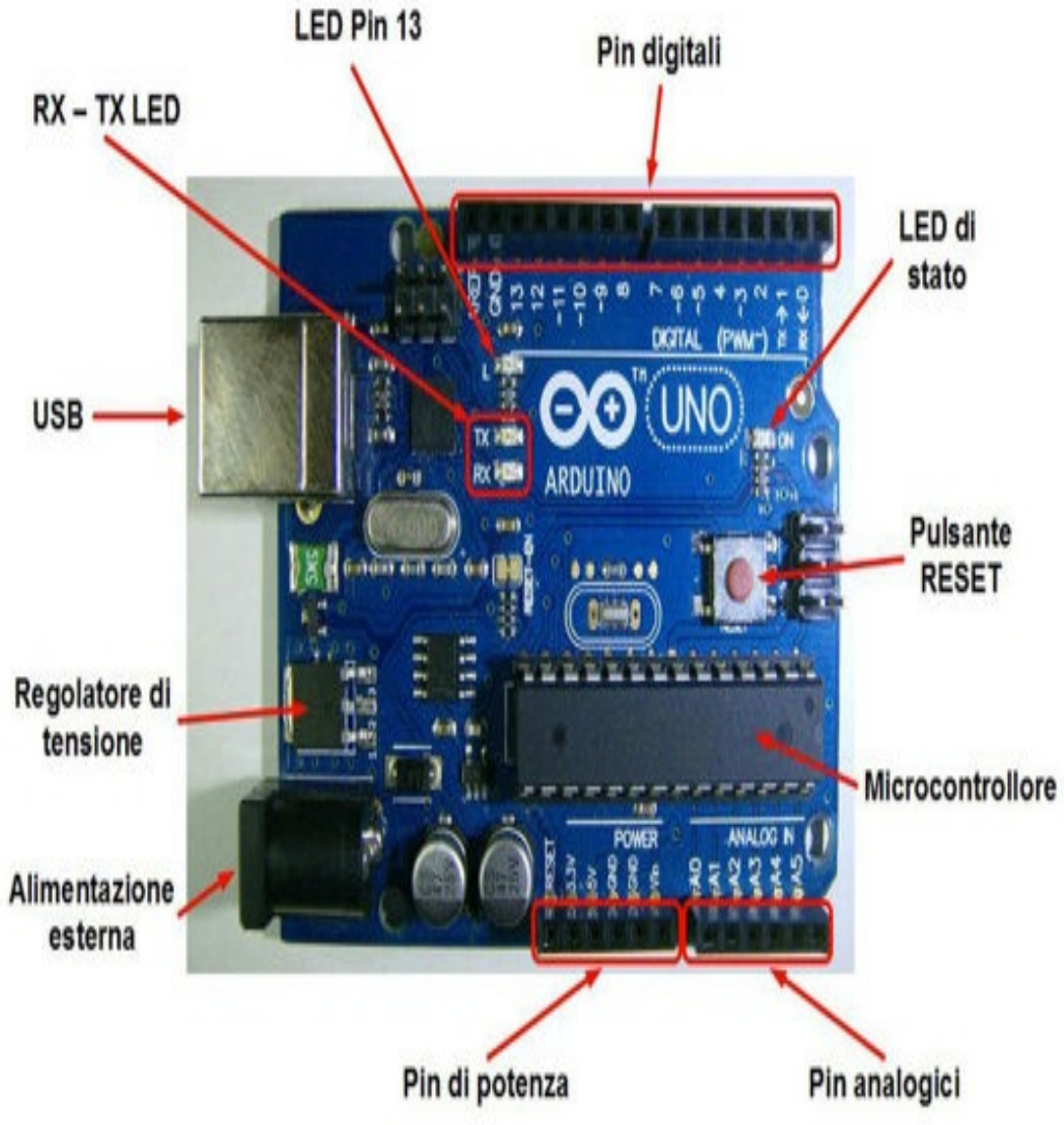
Arduino si divide in due parti, una parte hardware basata sui collegamenti tra i vari componenti elettrici e una parte software utilizzata per la programmazione della scheda.

La realizzazione di semplici sequenze e piccoli progetti con Arduino non richiede nessun requisito da ingegneri, basta munirsi del materiale adatto, acquistabile sui vari store online o su http://store.arduino.cc/, spesso con kit che comprendono la scheda Arduino e diversi componenti elettronici che andrai a conoscere in questa collana.

La scheda Arduino comprende diversi componenti importanti che impareremo a conoscere nel corso dello sviluppo dei progetti.

Esaminiamo meglio questi componenti e la loro funzione.

- **Alimentazione esterna:** Alimentazione da una fonte esterna 5 – 12 V.



- **Regolatore di tensione:** regola la tensione del precedente portandola a 5 V.
- **USB:** porta USB con cui andremo a collegarci al PC per programmare l'Arduino.
- **RX – TX LED:** LED di stato ricezione (RX) e invio (TX) dei dati.
- **LED Pin 13:** LED comunicante con il Pin 13.

- **Pin digitali:** possono avere solo la variabile HAIGHT (5V) o LOW (0V).
- **LED di stato:** LED verde indica che la scheda è alimentata.
- **Pulsante RESET:** interrompe il processo in corso e lo riesegue.
- **Microcontrollore:** elabora i processi e ci fa eseguire le funzioni programmate.
- **Pin analogici:** può assumere qualsiasi valore in un range noto.
- **Pin di potenza:** ci permettono di prendere direttamente la tensione dalla scheda.

La famiglia delle schede Arduino è particolarmente numerosa.

Si parte da Arduino Nano fino alla scheda standard (che autilizzerai nell'area di progetto) fino ad arrivare ad Arduino Mega che comprende maggiore capacità di processi e maggior numero di Pin.

Il software per la programmazione della scheda Arduino, chiamato più comunemente IDE (ambiente di sviluppo integrato), permette la stesura del codice sorgente chiamato "sketch" che verrà poi compilato e caricato sulla memoria del nostro Arduino.

Potrai scaricare il software gratuitamente dal sito ufficiale:

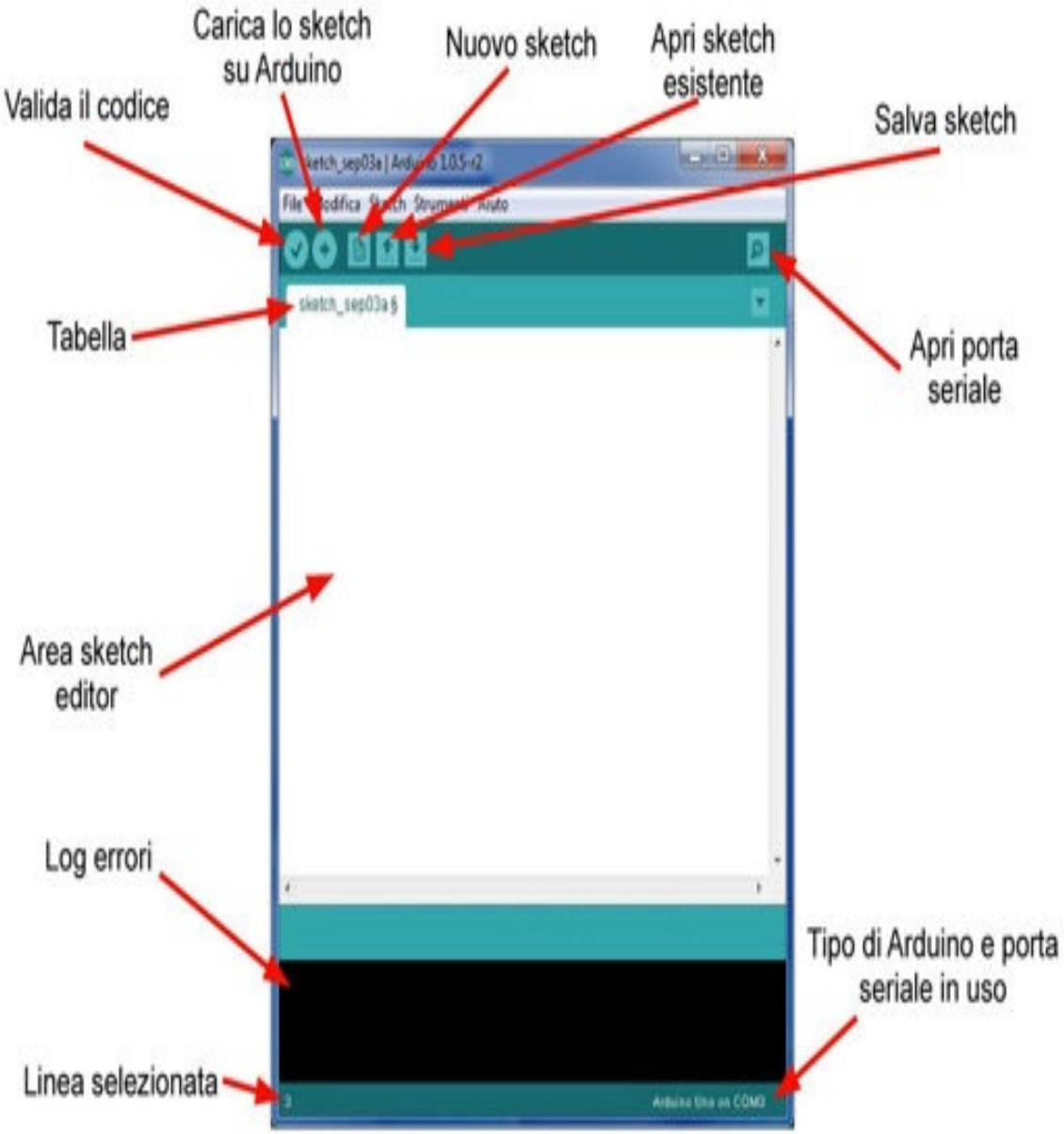
- http://arduino.cc/en/Main/Software

Seleziona il download in base al tuo sistema operativo e avvia l'installazione.

Al termine dell'installazione avvia il programma.

A questo punto si aprirà la schermata dell'IDE, dove andrai a scrivere lo sketch per poi farlo eseguire da Arduino.

Esaminiamo come è strutturata l'interfaccia del programma:



- **Linea selezionata:** rappresenta in quale linea dello sketch hai il cursore del testo.
- **Log errori:** tutti gli errori e dati del software vengono visualizzati qui sotto.

- **Area sketch editor:** è il punto in cui andrai a scrivere il codice da far eseguire ad Arduino.
- **Tabella:** indica quale sketch hai aperto, se apri altri progetti verranno aggiunte nuove tabelle.
- **Valida il codice:** verifica che lo sketch sia corretto, ma non lo carica su Arduino.
- **Carica lo sketch su Arduino:** verifica lo sketch e lo carica su Arduino.
- **Nuovo sketch:** crea un nuovo sketch.
- **Apri sketch esistente:** si apre una tendina che ti permette di aprire gli ultimi progetti oppure aprire un progetto da file.
- **Salva sketch:** salva lo sketch in una cartella a tua scelta.
- **Apri porta seriale:** ti permette di trasmettere dati dal PC ad Arduino e viceversa.
- **Tipo di Arduino e porta seriale in uso:** ti indica il modello di Arduino e la porta seriale su cui è collegato.

Ora comincia con il collegare Arduino al PC tramite cavo USB e attendi che venga riconosciuto dal sistema.

L'IDE è preconfigurato per collegare Arduino UNO alla porta COM3. In caso utilizzassi un Arduino diverso oppure una porta diversa basta andare sull'IDE nella barra dei menù in alto e cliccare **Strumenti > Tipo di Arduino** e selezionare il modello di Arduino.

Per cambiare la porta seriale basta andare su **Strumenti > Porta seriale** e selezionare la porta desiderata.

Per testare se la nostra configurazione iniziale è andata a buon fine ti basta scrivere un piccolo sketch che non farà assolutamente niente, ma con questo modo potrai verificare che il PC e l'Arduino stiano comunicando correttamente.

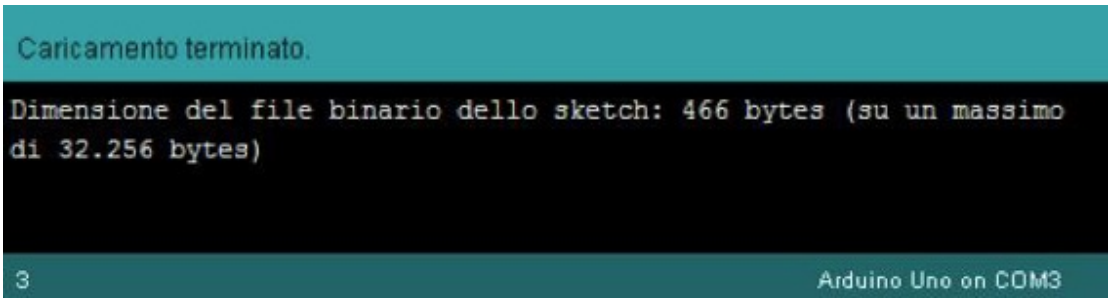
Scriviamo questo brevissimo codice**:**

```
void setup()
{}
void loop()
{}
```

Clicca sul pulsante **Carica lo sketch su Arduino.**

Attendi che l'IDE lo compili e lo invii, noterai che i **LED RX - TX** lampeggeranno per pochi millesimi di secondo, ciò significa che lo sketch è stato caricato correttamente.

Noterai inoltre che nell'area **Log errori** verrà mostrata la dimensione del file che abbiamo caricato.



NOTA

Al termine dei 15 progetti troverai il link per il download di tutti gli sketch.

1. Fai lampeggiare un LED

Per cominciare a familiarizzare con Arduino e la sua programmazione, questo è il primo progetto che andrai a realizzare.

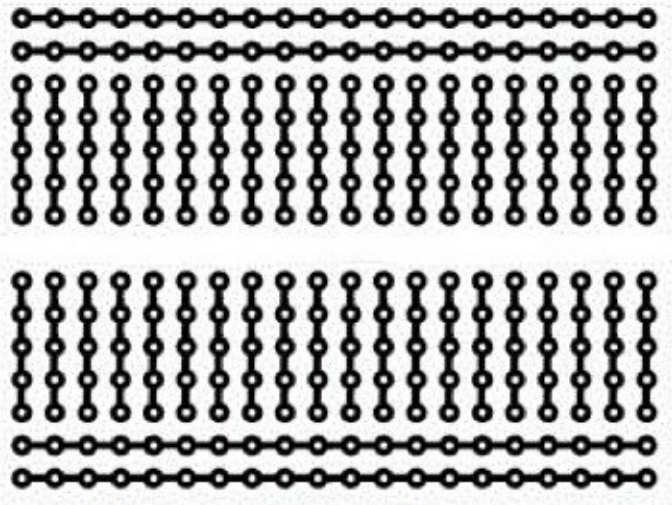
Far lampeggiare un semplice LED e più facile a dirsi che a farsi, perché dovrai inserire delle funzioni in modo da realizzare lo sketch da caricare sull'Arduino.

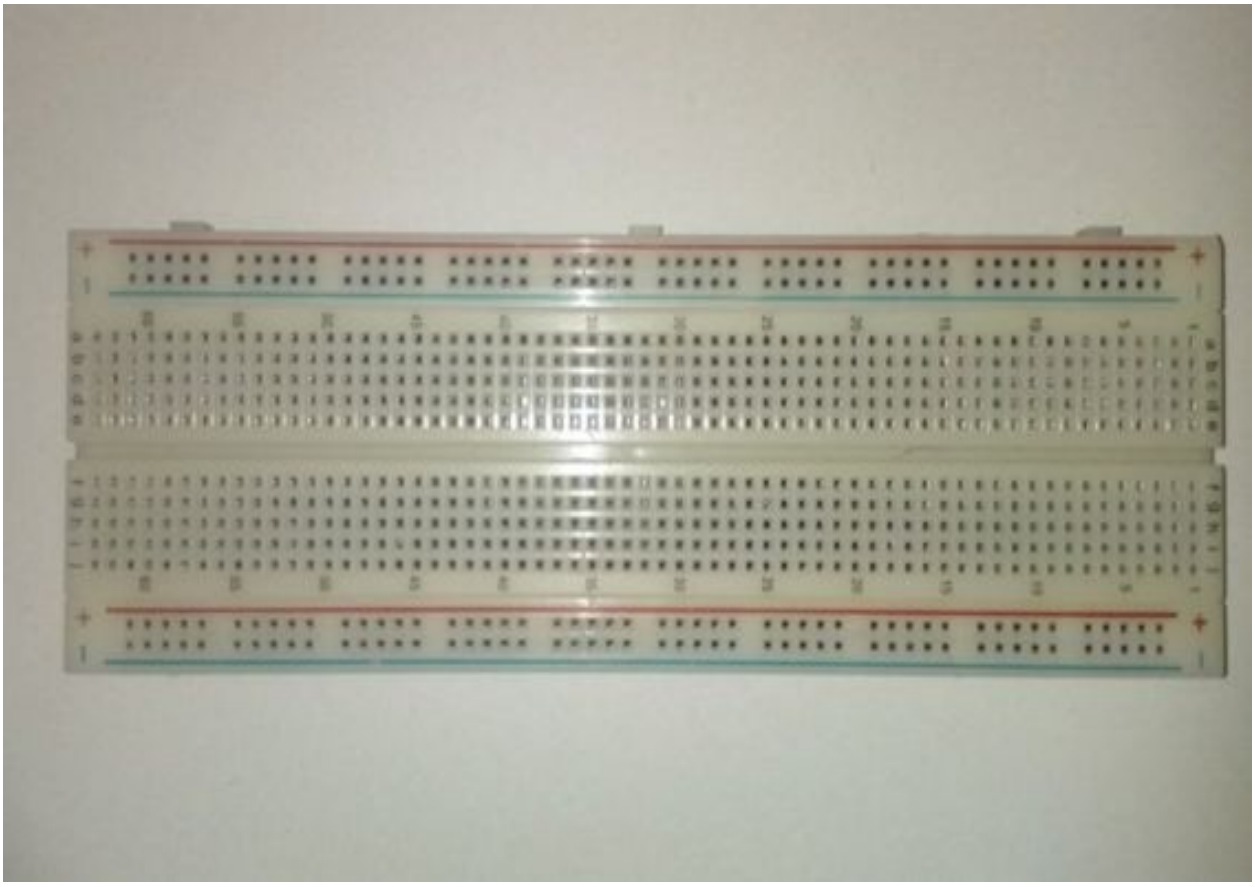
Cosa ti occorre?

- 1 Arduino Uno
- 1 Breadboard
- 4 Cavetti Jumper per fare i collegamenti
- 1 resistenza da 220 Ω
- 1 LED

Per la realizzazione di questo primo progetto (e per i successivi) dobbiamo avere del materiale elettrico che ci servirà per svolgere vari funzioni:

La Breadboard





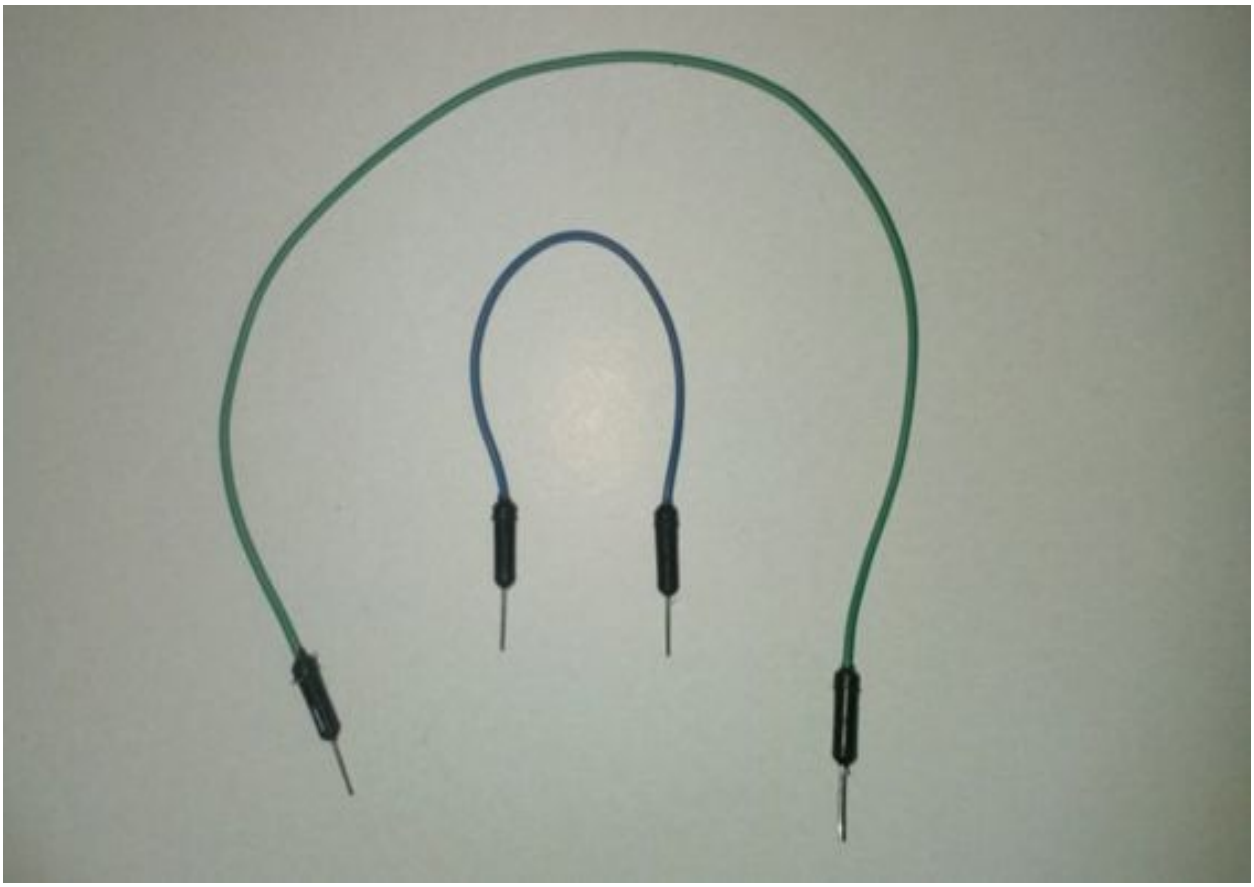
La breadboard (o basetta sperimentale) ti sarà molto utile per moltissimi dei progetti che andrai ad affrontare in questa collana di Arduino.

Viene utilizzata per fare circuiti sperimentali senza la necessità di effetture

saldature in modo da riutilizzare il materiale che hai disposto su di essa.

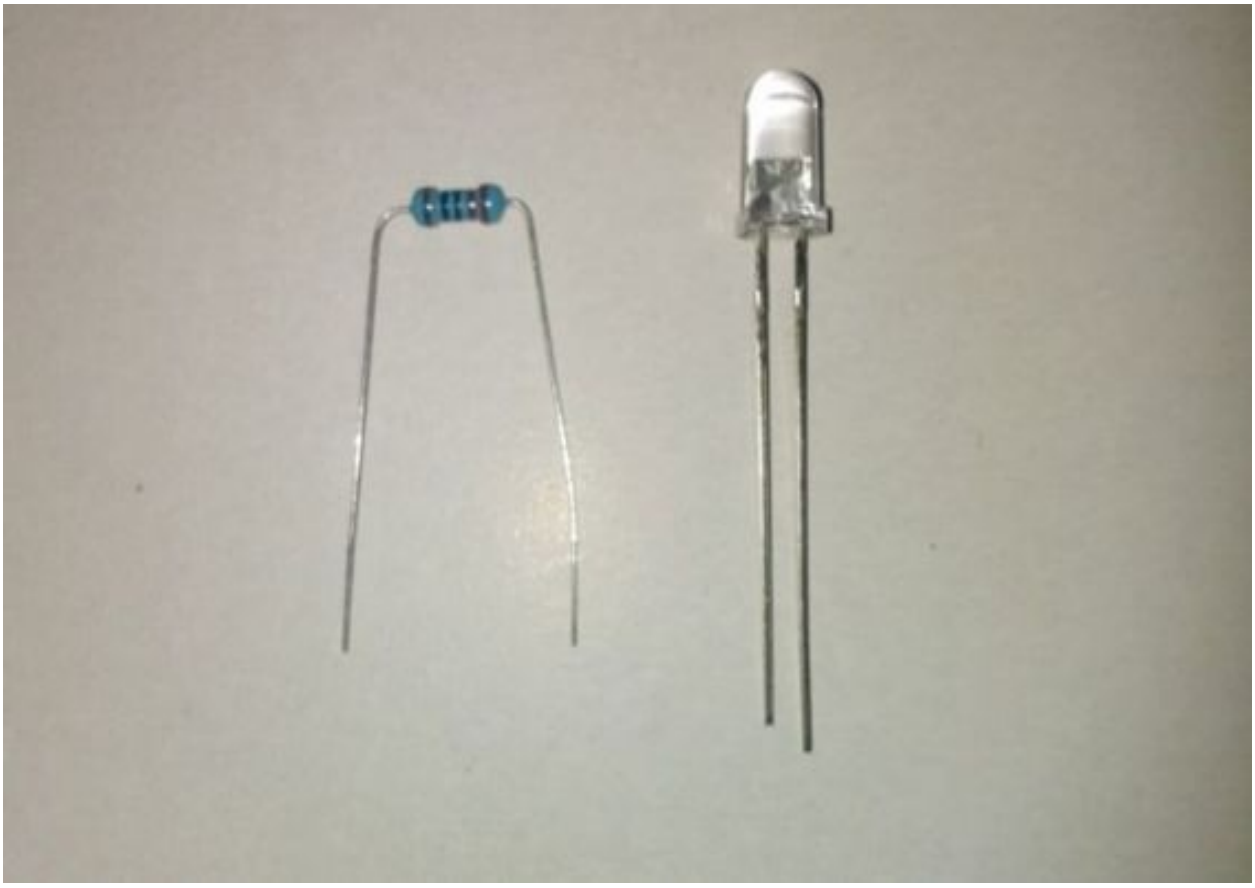
La basetta utilizza dei fori collegati tra di loro mediante uno schema ben preciso come in figura e ciò permette di collegare tutti i tuoi utilizzatori su di essa.

Cavetti Jumper



Sono cavetti di diversa lunghezza e di diverso colore che terminano con un connettore maschio – maschio adatto per connettersi ai Pin della scheda Arduino Uno fino alla Breadboard.

Resistenza e diodo LED



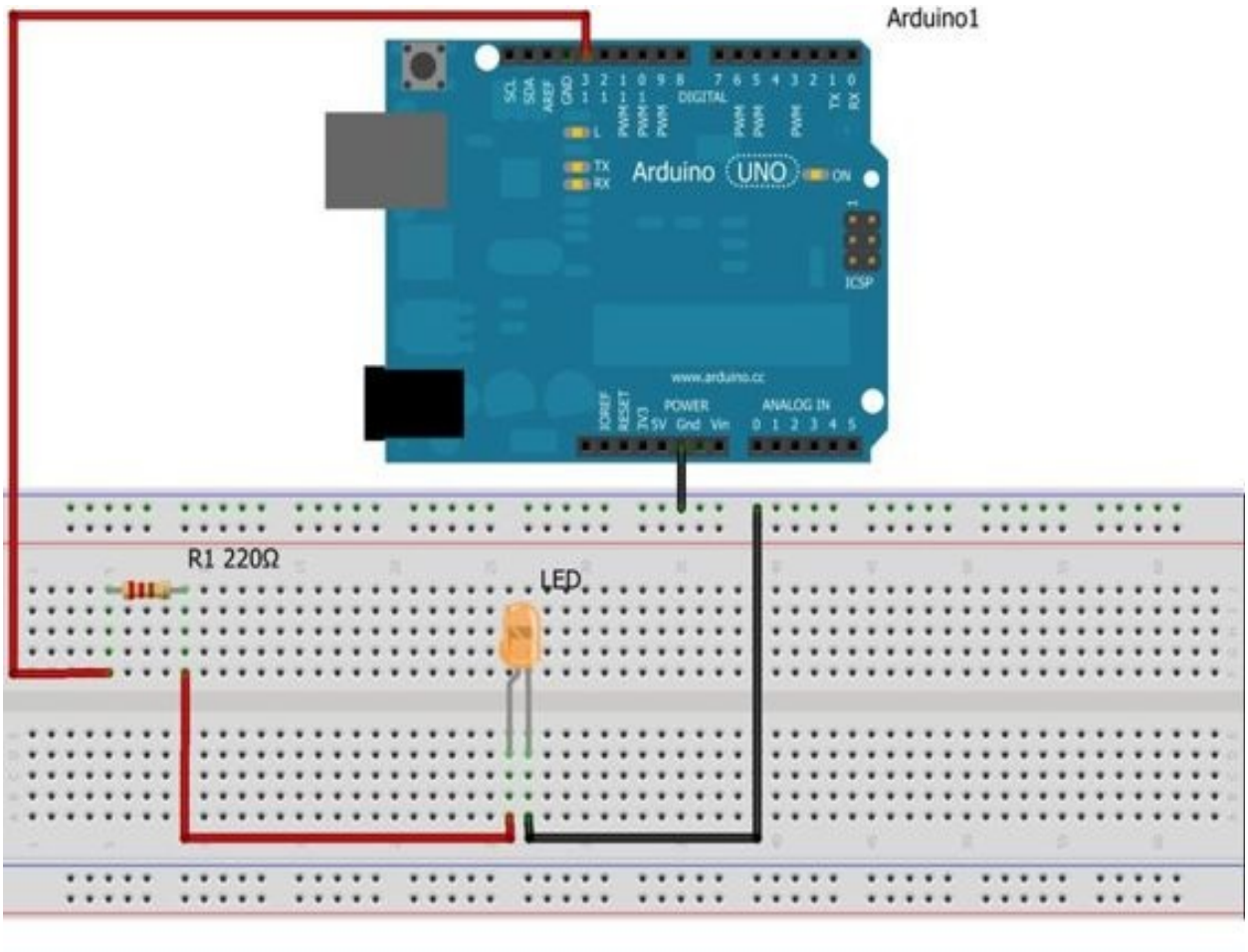
Il diodo LED (*Light Emitting Diode*) è un componente che se attraversato da una corrente molto bassa emette una luce che varia in base al metodo di costruzione.

Il LED ha una caratteristica di un diodo che, quindi, se alimentato dal terminale più lungo (anodo) lascia passare la corrente e si accende, mentre se viene alimentato dal terminale più corto (catodo) non lascia passare la corrente e quindi non emette luce.

La Resistenza è un componente elettronico che impedisce un passaggio eccessivo della corrente, in questo caso impedisce di danneggiare permanentemente il diodo LED.

La loro unità di misura è l'ohm (Ω) ma esistono diversi tipi di resistenze, le più usate sono quelle a bande di colori, che in base ai colori riportati corrispondono a delle cifre del valore della resistenza (vedi tabella nella parte finale dell'ebook).

Ora che hai conosciuto i componenti che andrai a utilizzare, puoi procedere con il collegamento dei componenti come in schema:



Ora non ti resta altro che connettere l'Arduino Uno al PC tramite cavo USB, lanciare l'IDE e trascrivere il codice mostrato di seguito.

```
int LED = 13;

void setup()
{
  pinMode(LED, OUTPUT);
}
void loop()
{
  digitalWrite(LED, HIGH);
  delay(1000);
  digitalWrite(LED, LOW);
  delay(1000);
}
```

Lo sketch e le sue funzioni:

Lo sketch che avete appena scritto sull'IDE contiene tutti i parametri che permettono di dire all'Arduino cosa deve fare.

Noterete che alcune scritte sono in **arancione** mentre altre in **nero** e altre ancora in **blu**: a cosa sono legate e cosa servono?

- I parametri in **arancione** sono le variabili e le funzioni, che servono all'Arduino per capire le tue intenzioni e ciò che vuoi che svolga.
- I parametri in **nero** sono utilizzati per indicare a quale parametro letterale o numerico sono associate variabili o funzioni.
- I parametri in **blu** servono a indicare il valore delle funzioni, se si tratta di un input o un output o lo stato di accessori, come LED e LCD.
- Un ultimo parametro che puoi notare nello sketch è di colore **grigio**; rappresenta solitamente la spiegazione della funzione dello sketch, ma non svolge nessuna funzione essenziale per l'esecuzione.

Analisi dello sketch del primo progetto:

Lo sketch riportato precedentemente svolgerà la funzione di far lampeggiare il LED collegato al Pin 13 di Arduino con l'intervallo di 1 secondo.

Analizziamolo in dettaglio:

```
int LED = 13;

void setup()
{
  pinMode(LED, OUTPUT);
}
void loop()
{
  digitalWrite(LED, HIGH);
  delay(1000);
  digitalWrite(LED, LOW);
  delay(1000);
}
```

- int, è la variabile principale dello sketch, in cui andiamo a dire che la parola **LED** è associata al **Pin 13** di Arduino. Questa funzione ti permette di passare da un valore numero del Pin a un valore letterale, che ti sarà molto utile quando il codice sarà più lungo e complesso.

- void **setup**(), è il punto in cui inserirai il codice di inizializzazione. Esso inizializza tutte le impostazioni e le istruzioni della scheda come gli input e gli output prima che si avvii il ciclo principale del programma.
  Aprendo la parentesi graffa ({) potrai inserire le impostazioni, come pinMode che associa il valore LED (quindi il Pin 13) con la impostazione di OUTPUT cioè di uscita.
  La parentesi graffa (}) termina il parametro.
- void **loop**(), è il contenitore del programma principale. Contiene una serie di istruzioni che verranno eseguire in modalità ciclica, cioè all'infinito fino a quando non andrai a togliere alimentazione alla scheda di Arduino.
  Aprendo la parentesi graffa ({) andrai a scrivere il codice con le azioni che Arduino dovrà compiere.

Con la funzione digitalWrite stabilirai che il valore LED (associato al Pin 13) sarà di valore HIGH **(alto)**, che consente di far uscire dal Pin 13 una tensione pari a 5V.

Per impostare per quanto tempo il LED dovrà essere acceso utilizzerai la

funzione delay, con il valore 1000 (1000 millisecondi = 1 secondo).

Dopo che avrai impostato il tempo, dovrai inserire nuovamente la funzione digitalWrite relativa al valore LED (Pin 13) che sarà di valore LOW **(basso)** e cesserà di erogare i 5V spegnendo il LED.

Successivamente, per completare il ciclo, dovrai inserire nuovamente la variabile delay, con valore 1000 e successivamente chiudere la parentesi graffa (}).

Grazie al void **loop**(), tutto questo processo viene ripetuto in automatico fino a quando forniamo alimentazione alla scheda Arduino.

Ora non ti resta che compilare e caricare il programma sul tuo Arduino e noterai che il LED che hai collegato come in schema incomincerà a lampeggiare a intervalli di 1 secondo.

NB: modificando le variabili delay, potrai modificare i tempi di lampeggiamento del LED.

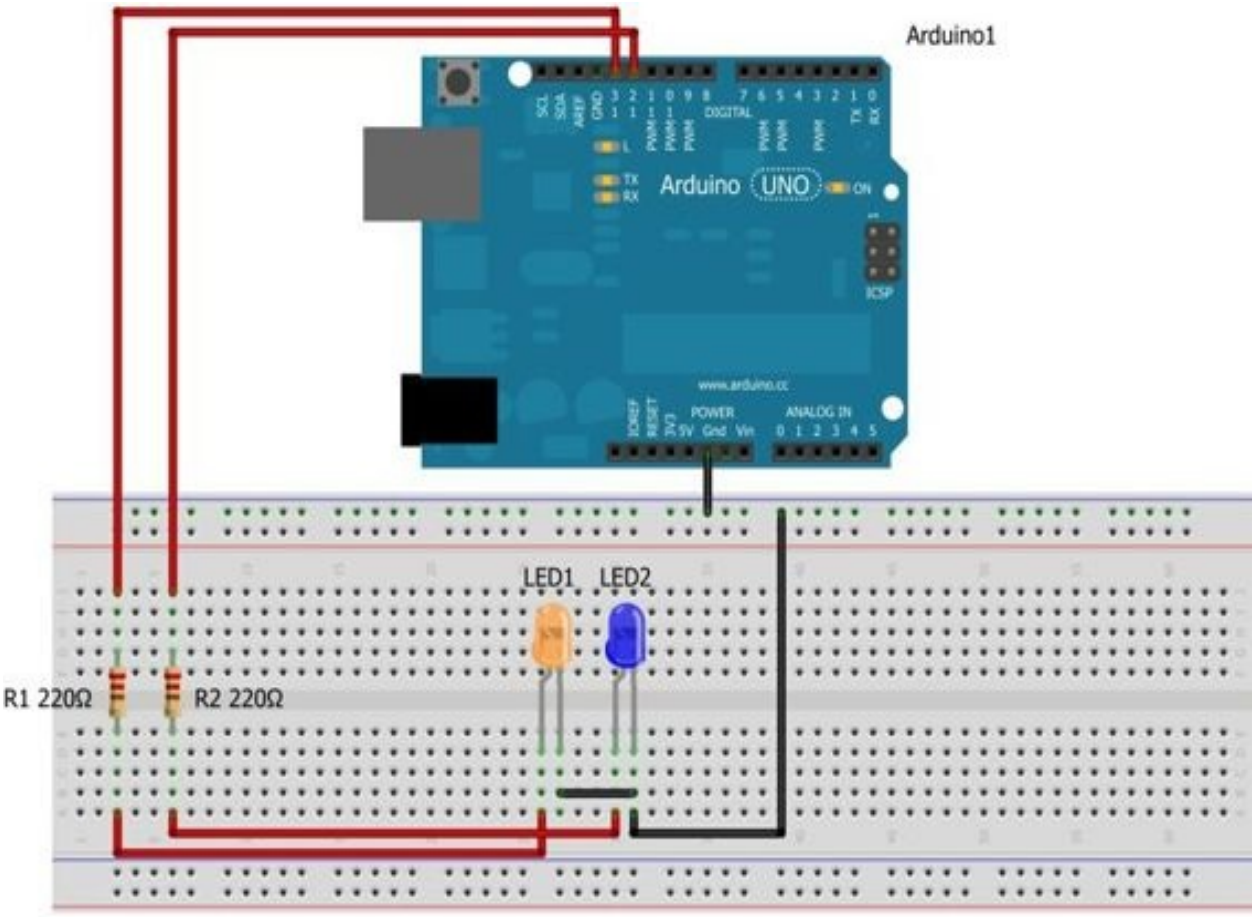
2. FAR LAMPEGGIARE 2 LED

Dopo aver conosciuto le nozioni base di Arduino e come sono strutturati i progetti, puoi passare a questo progetto, che implementa una parte del progetto precedentemente con l'aggiunta di 2 LED che si accenderanno e si spegneranno a intermittenza, con intervallo di 1 secondo per il LED1 e di 2 secondi per il LED2.

Cosa ti occorre?

- 1 Arduino Uno
- 1 Breadboard
- 7 Cavetti Jumper per fare i collegamenti
- 2 resistenze da 220 Ω
- 2 LED

Inizia collegando tutti i componenti come riportato nello schema di collegamento o elettrico, facendo molta attenzione a collegarli in modo esatto, evitando di provocare danni permanenti alla scheda Arduino.



Una volta collegato e verificato correttamente che tutto sia ok, procediamo con aprire l’IDE di Arduino sul PC e trascrivere o scaricare il seguente sketch:

```
int LED1 = 12;                  // Il LED1 è associato al pin 12 di Arduino
int LED2 = 13;                  // Il LED2 è associato al pin 13 di Arduino

void setup()
{
  pinMode(LED1, OUTPUT);        // Il LED1 (Pin 12) è gestito come USCITA
  pinMode(LED2, OUTPUT);        // Il LED2 (Pin 13) è gestito come USCITA
}
void loop()
{
  digitalWrite(LED1, HIGH);     // Il LED1 (Pin 13) avrà un valore ALTO = 5V
  digitalWrite(LED2, LOW);      // Il LED2 (Pin 13) avrà un valore BASSO = 0V
  delay(1000);                  // Funzione tempo della durata 1000 ms = 1 secondo
  digitalWrite(LED1, LOW);      // Il LED1 (Pin 13) avrà un valore BASSO = 0V
  digitalWrite(LED2, HIGH);     // Il LED2 (Pin 13) avrà un valore ALTO = 5V
  delay(2000);                  // Funzione tempo della durata 2000 ms = 2 secondi
}
```

Avendo dato per scontato che hai appreso i concetti spiegati nel primo progetto, questo è molto simile al precedente. Inoltre, per facilitarti le cose, avrai a fianco di ogni funzione la spiegazione e a che cosa si riferisce.

Una volta che hai capito il funzionamento del codice, potrai connettere l'Arduino al PC tramite cavo USB e caricare lo sketch.

Al termine del caricamento lo sketch verrà letto ed eseguito nel seguente modo:

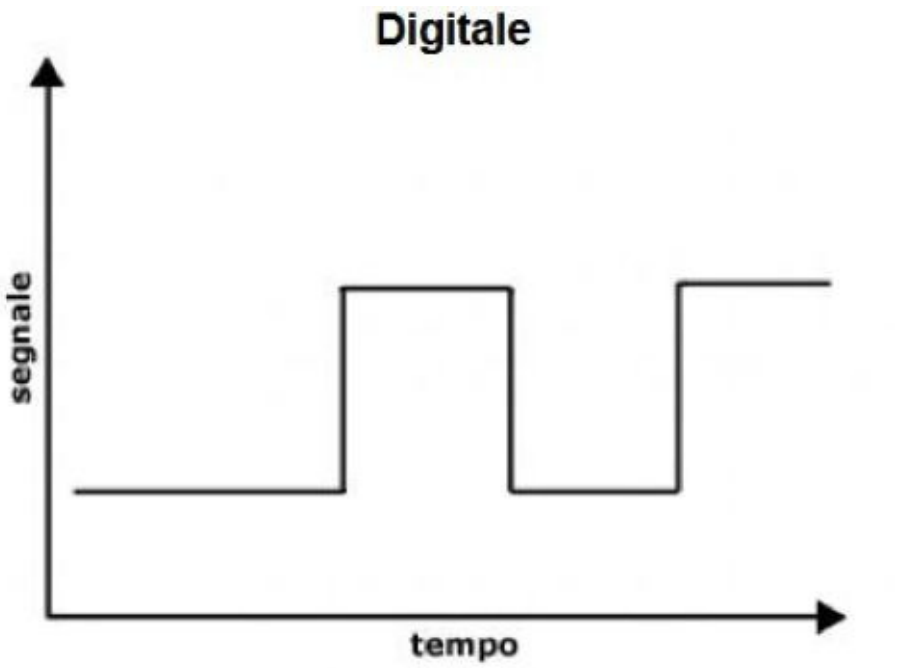
1. Si accende il LED1 e resta spento il LED2 per 1 secondo.
2. Si spegne il LED1 e si accende il LED2 per 2 secondi.
3. Il processo si ripete all'infinito.

3. LEGGERE DATI DIGITALI DA SERIALE

Per implementare componenti come sensori e pulsanti, dovrai avere le conoscenze necessarie a permetterti di leggere dei dati in maniera digitale.

I dati digitali sono chiamati così perché possono avere solamente 2 tipi di valori: 1 e 0.

In questo caso il valore 1 sarà pari a 5V e il valore 0 a 0V.

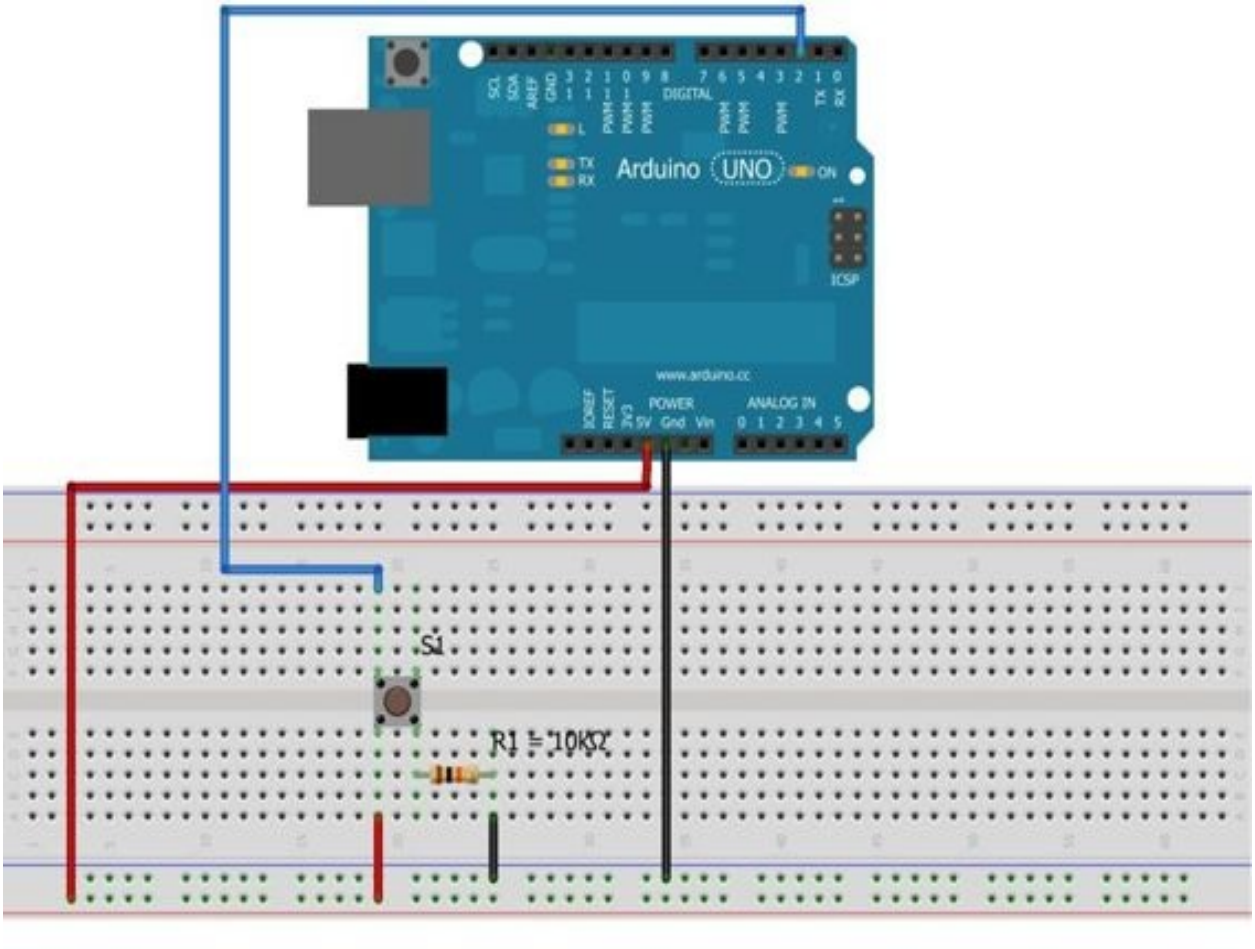


Cosa ti occorre?

- 1 Arduino Uno
- 1 Breadboard
- 3 Cavetti Jumper per fare i collegamenti
- 1 resistenza da 10 kΩ (10000Ω)

- 1 pulsante NO (normalmente aperto)

Procedi con il collegamento di tutti i componenti come mostrato nello schema seguente:

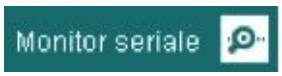


Verificato il corretto montaggio di tutti i componenti, usa il seguente sketch:

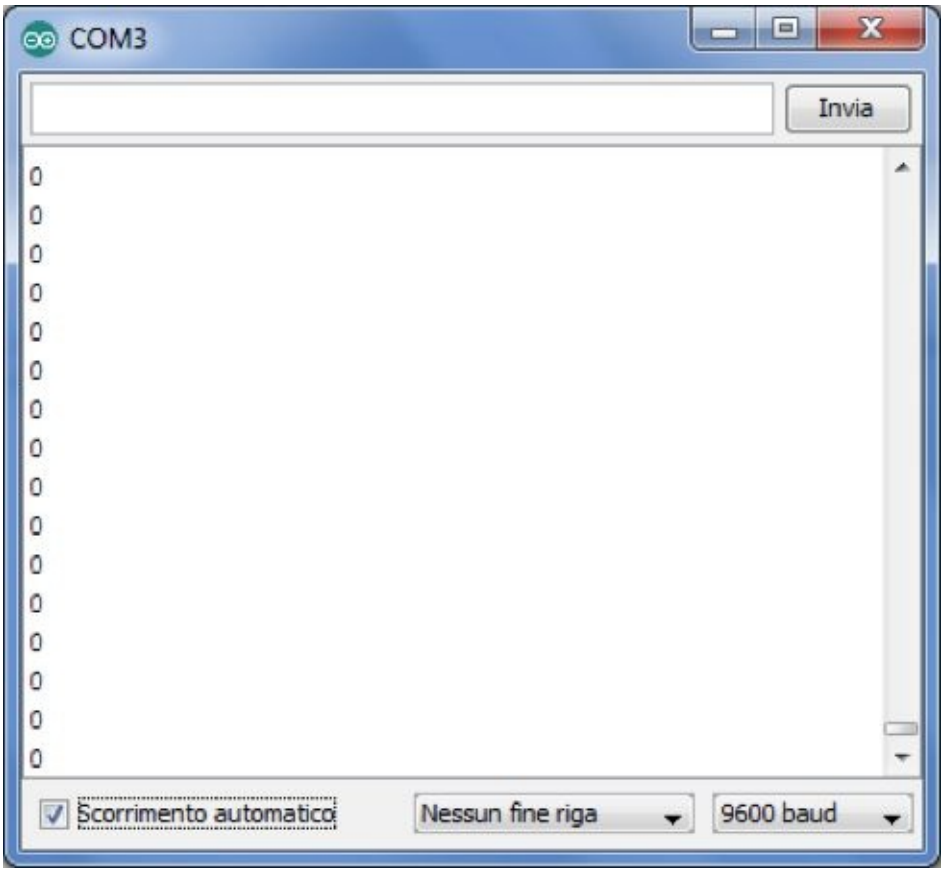
```
int S1 = 2;               // P1 è il pulasnte sul Pin 2

void setup()
{
  Serial.begin(9600);     // E' il valore della porta seriale usata per comunicare da Arduino - PC
  pinMode(S1, INPUT);     // Associalo il P1 (Pin 2) come INPUT
}
void loop()
{
  int S1 = digitalRead(2); // leggo il valore del pin 2 ad ogni ciclo e assegno tale valore alla variabile S1
  Serial.println(S1, DEC); // Stampa su seriale dell'IDE un valore DEC = decimale
}
```

Ora non ti resta che collegare la scheda Arduino al tuo PC e caricare lo sketch.

Al termine del caricamento, come noterai, sembra che non accada nulla. Potrai però osservare che il **LED – TX** di Arduino si accenderà. Questo segnalale ci indica che l'Arduino sta inviando dei dati tramite serial e (**TX**) che potrai visualizzare cliccando sul tasto in alto a destra dell'IDE.

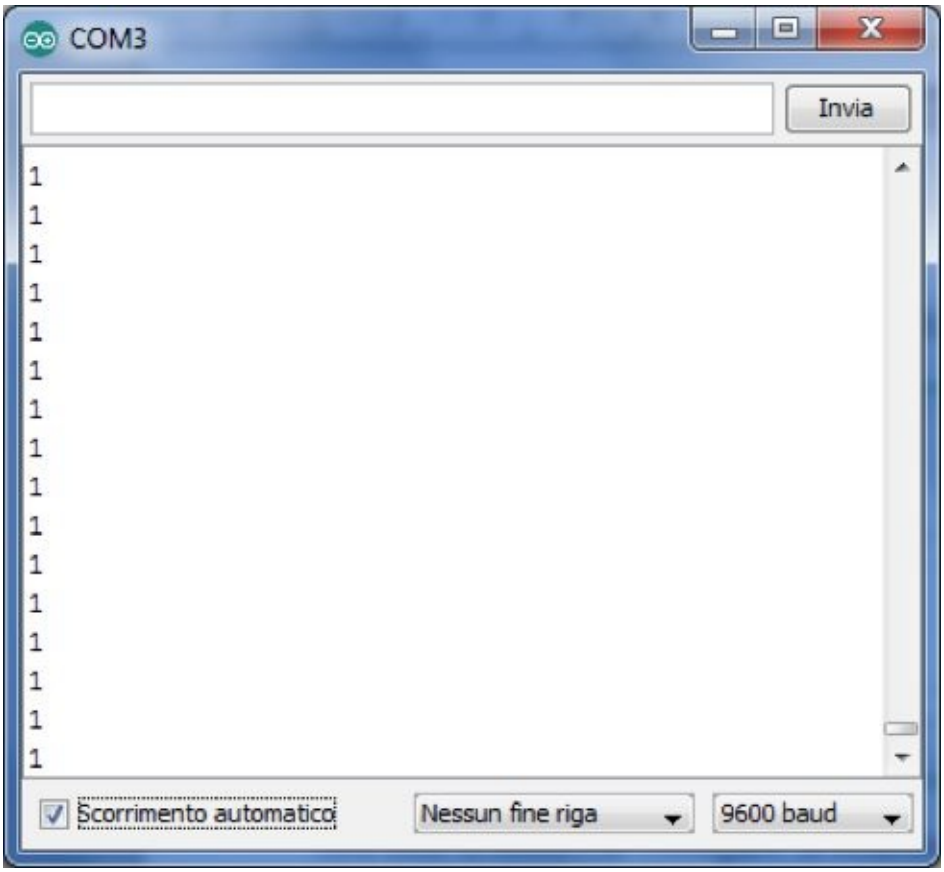


A questo punto si aprirà una finestra che verrà aggiornata automaticamente sui dati che l'Arduino sta trasmettendo via seriale:



Noterai che il valore che riceve è 0 quindi paragonato a 0V, questo perché il pulsante S1 è a riposo, e quindi connesso ai 0V.

Appena premi il pulsante S1 tali valori cambieranno, assumendo il valore 1:

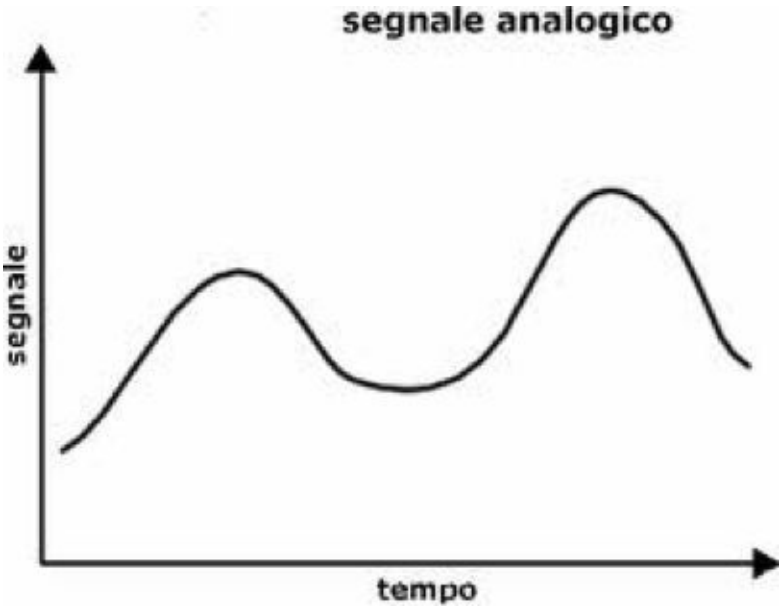


Questo accade perché premendo S1 fai commutare i 5V, quindi con valore decimale 1.

4. LEGGERE DATI ANALOGICI DA SERIALE

Sembra simile al progetto 3 nella lettura dei dati digitali da seriale, con la differenza che questi dati analogici possono avere diverse forme d'onda e sono racchiusi in un valore che varia da 0 a 1024.

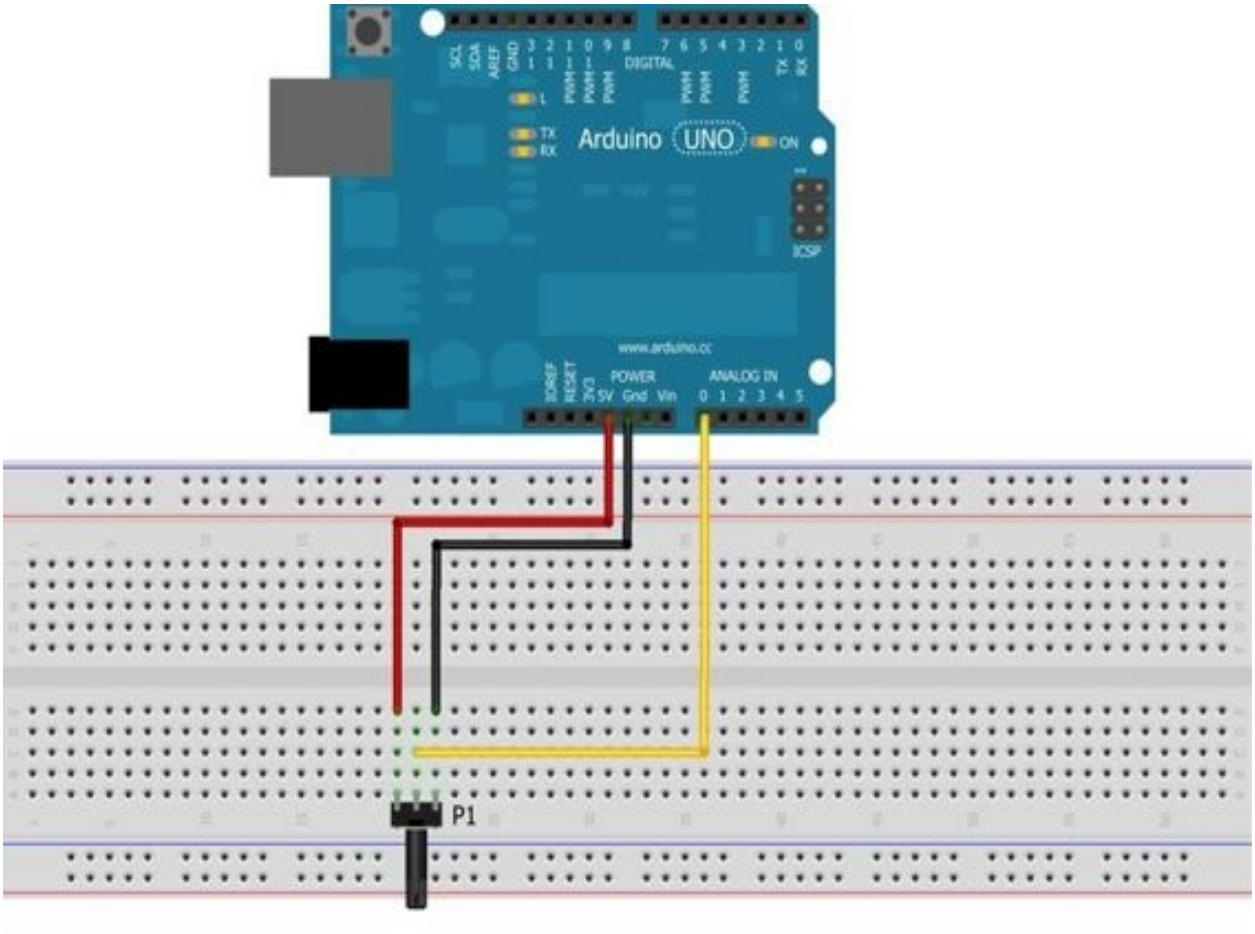
Questo strumento è molto utile per la lettura di valori di potenziometri, sensori di temperatura e luminosità.



Cosa ti occorre?

- 1 Arduino Uno
- 1 Breadboard
- Cavetti Jumper per fare i collegamenti
- 1 potenziometro da 10 kΩ (10000Ω)

Procedi con il collegamento come da schema:

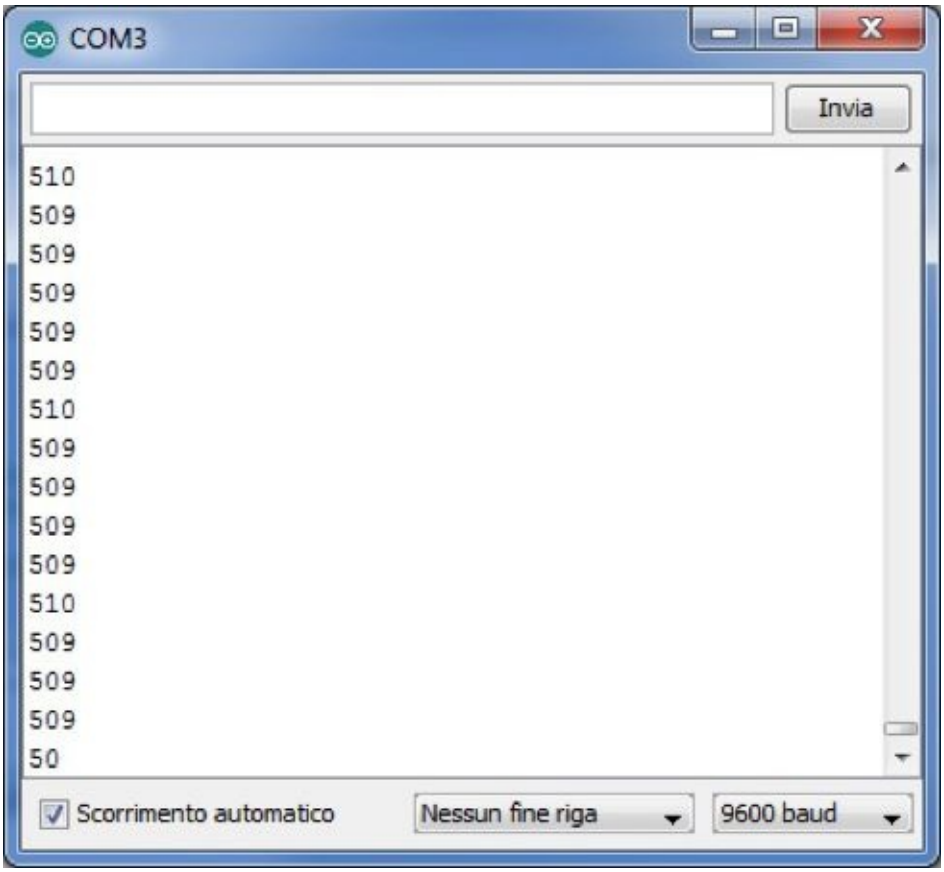


A collegamenti effettuati trascrivi e carica il seguente sketch:

```
void setup()
{
  Serial.begin(9600);               // porta seriale di comunicazione arduino - PC
}
 void loop()
{
  int sensorValue = analogRead(A0); // lettura sensore di tipo analogico sul pin A0
  Serial.println(sensorValue, DEC); // stampa su seriale un valore DEC = decimale del sensore
}
```

Carica lo sketch sull'Arduino e, come nel progetto precedente, noterai che il **LED – TX** di Arduino si accenderà. Ora clicca sul tasto in alto a destra dell'IDE.

Si aprirà la finestra dei dati seriali, che aggiornandosi mostrerà dei valori numerici che variano da 0 a 1024.



Questi valori sono riferiti al potenziometro che abbiamo collegato all'Arduino.

Il valore 0 è uguale ai 0V e il valore 1024 a 5V, ma a differenza del precedente progetto, questo può acquisire anche valori intermedi.

Nello screen riportato sopra, ad esempio, il valore 509 è circa a 2,5 V.

Ruotando il potenziometro noterai che verranno visualizzati diversi valori intermedi.

Questo progetto può sembrare banale e inutile, ma ciò ti permetterà di utilizzare tutti i sensori che andrai a scoprire tramite questa collana di ebook.

5. Accendere LED tramite pulsante

Nei progetti precedenti hai avuto un'infarinatura sulle funzionalità base di Arduino. In questo progetto proverai a fondere alcuni progetti precedenti per dar luogo a un progetto più complicato.

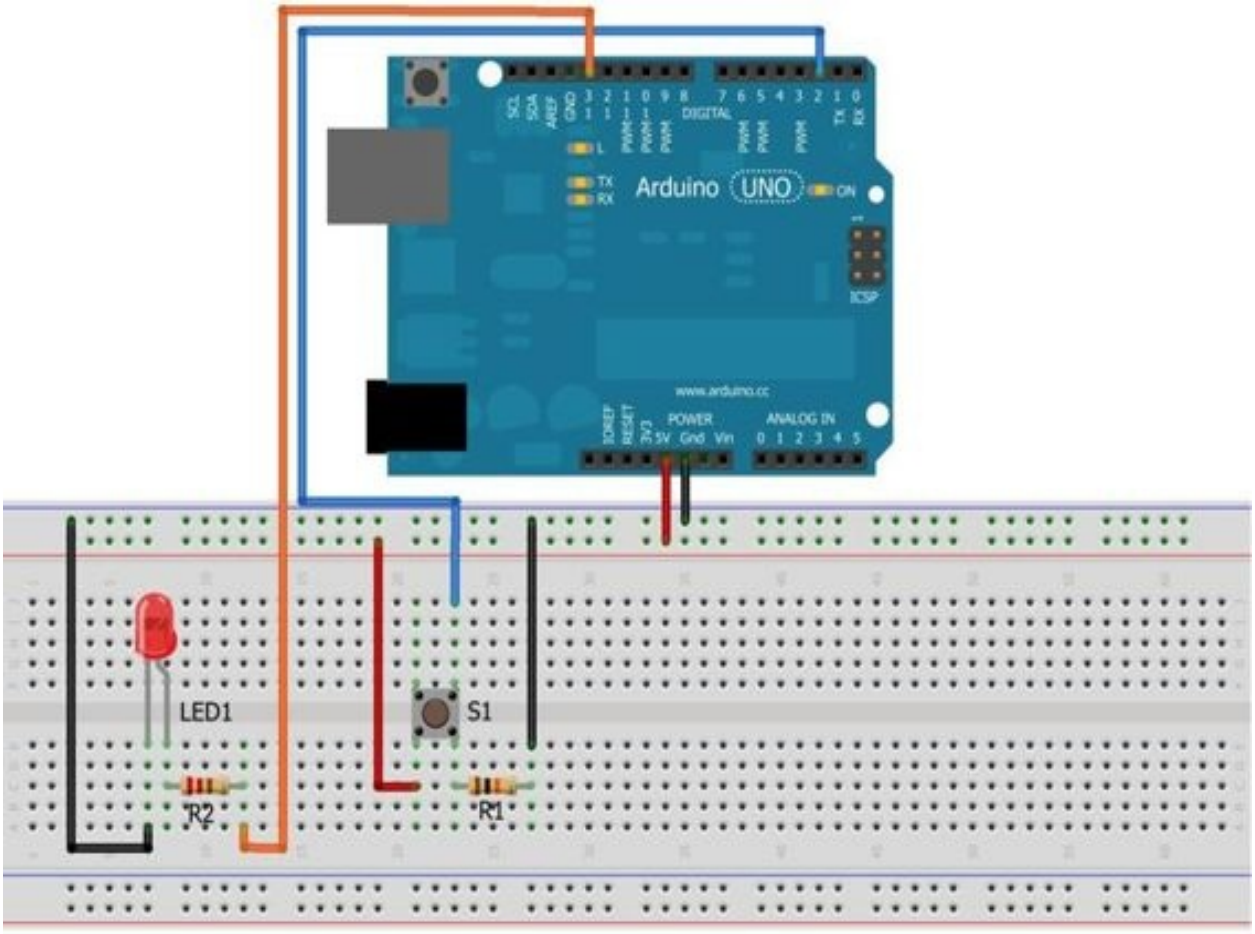
In questo progetto andrai ad accendere un LED utilizzando un pulsante, con modalità digitale e con alcune variabili.

Il funzionamento è questo: se premi il pulsante accendi il LED e rimane acceso anche se smetti di premere il pulsante, per spegnerlo dovrai premere nuovamente il pulsante.

Cosa ti occorre?

- 1 Arduino Uno
- 1 Breadboard
- Cavetti Jumper per fare i collegamenti
- 1 LED
- 1 resistenza da 220Ω
- 1 resistenza da 10 kΩ
- 1 pulsante del tipo NO

Procedi con il collegamento come da schema:


Arduino UNO
LED1
S1
R2
R1

```
int LED = 13;                    // LED = pin 13
int S1 = 2;                      // S1 = pin 2
int fs1LED = LOW;                // fase 1, o fase iniziale dello stato del LED = LOW (spento)

void setup()
{
  pinMode(LED, OUTPUT);          // LED = pin 13 impostato come OUTPUT (uscita)
  pinMode(S1, INPUT);            // S1 = pin 2 impostato come INPUT (entrata)
}

void loop()
{
  int leggiS1=digitalRead(S1);   // leggiS1 con variabile in modo digitale di S1

  if (leggiS1 == 1)              // Variabile if = se leggiS1 premuto = valore 1
  {
     if (fs1LED == LOW)          // Verfica se lo stato del led è LOW (basso)
     {
       fs1LED = HIGH;            // egegue comando stato LED HIGH (alto)
       digitalWrite(LED, fs1LED); // usa la funzione il valore LED e stato LED su HIGH (alto)
       delay(1000);              // conta intervallo di tempo 1 secondo = 1000 ms
     }
     else                        // variabile se diverso da quello letto da if, esegue comando sottostante
     {
       fs1LED = LOW;             // egegue comando stato LED LOW (basso)
       digitalWrite(LED, fs1LED); // usa la funzione il valore LED e stato LED su LOW (basso)
       delay(1000);              // conta intervallo di tempo 1 secondo = 1000 ms
     }
  }
}
```

Nello sketch riportato sopra, come hai notato, ci sono delle variabili nuove: **if** e **else**. Ma cosa sono?

Queste variabili si applicano a una condizione che può essere di due stati: **SI** oppure **NO**

Nello sketch sfruttiamo questa condizione per sapere se il LED è acceso oppure no, in modo da poterlo spegnere se la variabile è su SI, oppure spegnere il LED se la variabile è su NO.

Ora non ti resta che caricare lo sketch sull’Arduino per verificare il corretto funzionamento del progetto.

Questo progetto sarà molto utile per i sensori e per verificare lo stato di attuatori che andrai a scoprire più avanti in questa collana.

6. LEGGERE SEGNALI DA FOTORESISTENZA

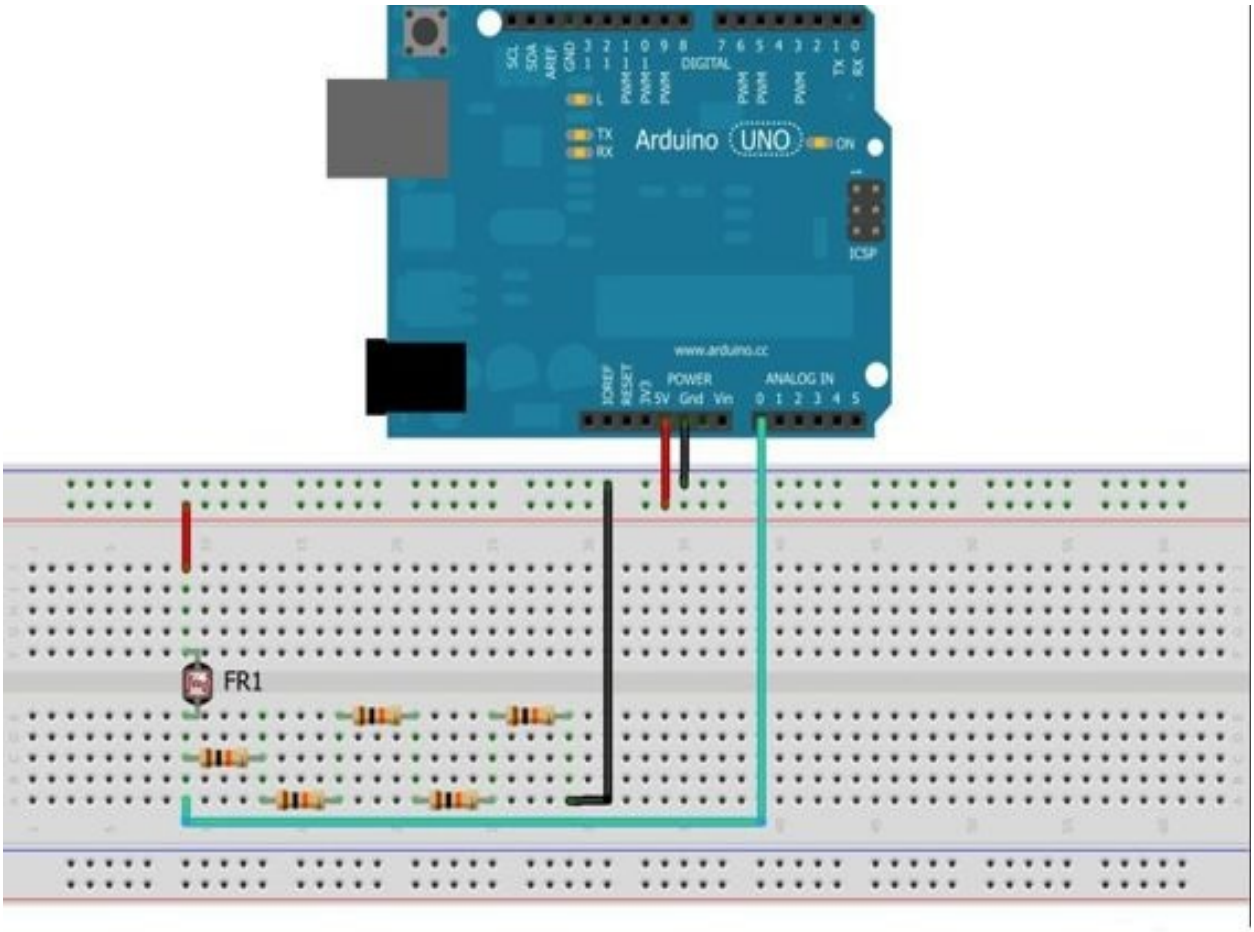
Questo progetto ti servirà per misurare il livello di luce tramite una fotoresistenza mediante seriale, utilizzando Arduino Uno.

Cosa ti occorre?

- 1 Arduino Uno
- 1 Breadboard
- Cavetti Jumper per fare i collegamenti
- 1 fotoresistenza da 50Ω
- 5 resistenze da 10 kΩ

Come avrai notato, avrai bisogno di un nuovo componente: la fotoresistenza o fotoresistore, che varia la sua resistenza in base alle radiazioni luminose sia solari che di luce artificiale.

Procedi con il collegamento come da schema:



Per il progetto sarebbe richiesta una resistenza pari a 50 kΩ che non si trovano in vendita. Dovrai collegare 5 resistenze da 10 kΩ in serie come nella figura soprastante.

```
int FR1 = A0;                    // FR1 (fotoresistenza) = Pin (analogico) A0
int FR1valore = 0;               // variabile in cui memorizzare il valore della fotoresistenza

void setup()
{
  Serial.begin(9600);            // Comunicazione porta seriale 9600
  pinMode( FR1, INPUT);          // Indichiamo ad Arduino che il pin A0 (FR1) sarà di INPUT
}

void loop()
{
  FR1valore = analogRead(FR1);   // Legge il segnale dul pin A0 e lo assena alla variabile FR1valore = 0

  Serial.print("sensore = " );    // Scrive sul monitor seriale la frase "sensore ="
  Serial.println(FR1valore);     // Scrive il valore letto dal Pin A0 e manda a capo il cursore

  delay(1000);                   // Attendi 1000 ms = 1 secondo e riesegue il processo
}
```

Ora carica lo sketch sull'Arduino.

Non ti resta che aprire il monitor seriale di Arduino: noterai che ogni secondo riceverai dei dati che possono variare da 0 a 1024.

Coprendo la fotoresistenza questi valori si abbasseranno fino a raggiungere circa il valore di 512.

Se lasci la fotoresistenza per diversi minuti al buio, il valore che leggerai si abbasserà sempre di più intorno al valore 100. Questo è dovuto al fatto che la fotoresistenza assume un valore che si avvicina ai 10 MΩ, risultando praticamente un circuito interrotto.

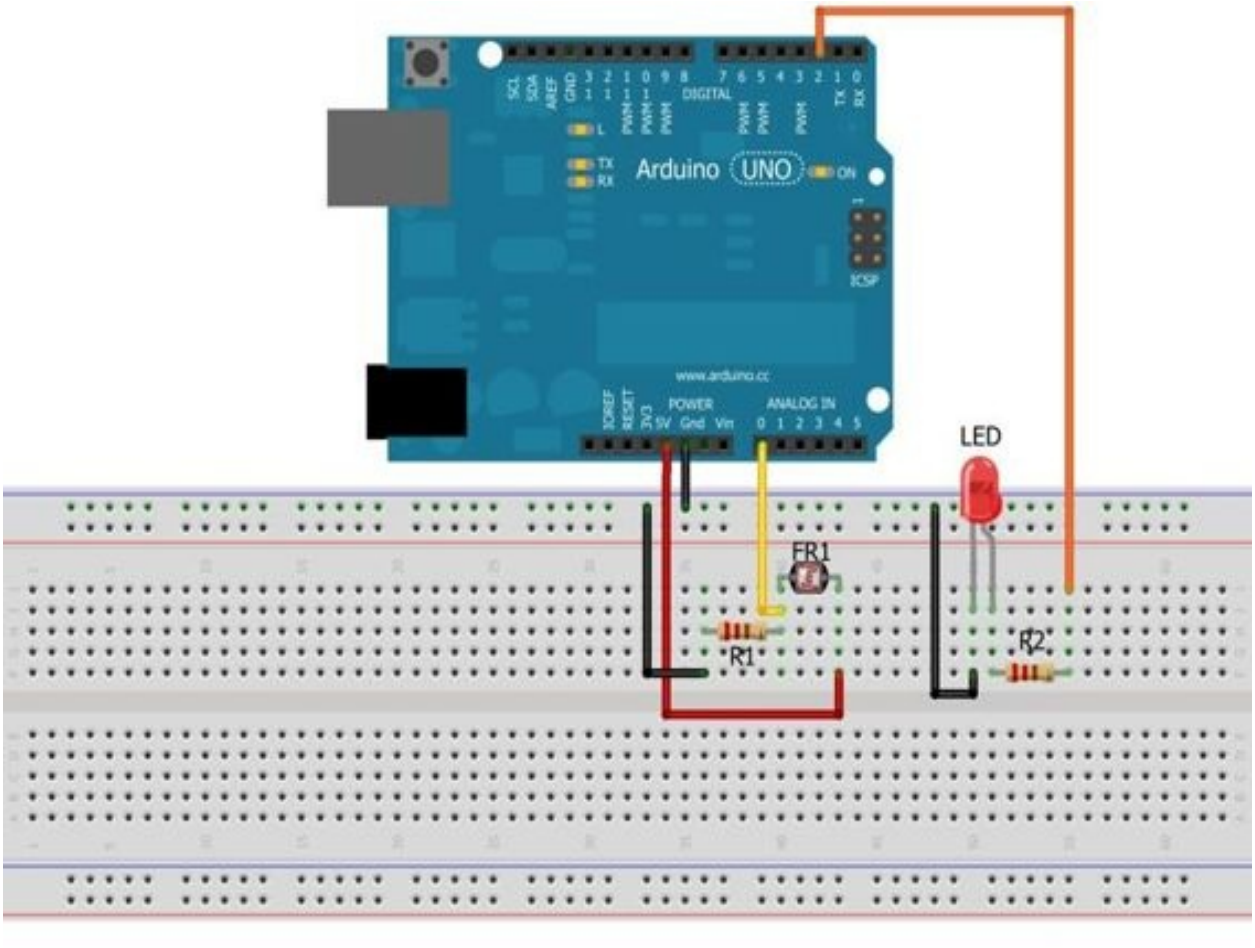
7. ACCENDERE LED TRAMITE FOTORESISTENZA

In questo progetto andrai a comandare un LED mediante l'uso di una fotoresistenza, questa funzione è chiamato anche crepuscolare cioè, se la fotoresistenza rileva la luce il LED rimane spento, mentre man mano che la luce diminuisce fino a raggiungere un certo valore andrai ad impostare, accenderà il LED.

Cosa ti occorre?

- 1 Arduino Uno
- 1 Breadboard
- Cavetti Jumper per fare i collegamenti
- 1 fotresistenza
- 2 resistenze da 220 Ω

Procedi con il collegamento come da schema:


Arduino UNO
www.arduino.cc
DIGITAL
POWER
ANALOG IN
ICSP
LED
FR1
R1
R2

```
int FR1 = A0;
int LED = 2;

void setup()

{

 pinMode(FR1, INPUT);
 pinMode(LED, OUTPUT);
 Serial.begin(9600);        // Inizializzo la comunicazione seriale

}

void loop()

{

 int val = analogRead(FR1); // Salve il valore fotoresistenza dentro alla variabile val
 Serial.println(val, DEC);  // Scrive il valore della fotoresistenza, espresso in numeri decimali
 if(val<800)                // Se il valore letto dalla fotoresistenza (luminosità) è basso, accendo il led
 digitalWrite(LED, HIGH);
 else
 digitalWrite(LED, LOW);    // altrimenti lo spengo (o lo lascio spento)

}
```

Associamo **FR1** (fotoresistenza) al **Pin A0** e il valore **LED** al **Pin 2** di Arduino.

Ora carica lo sketch sull'Arduino.

Una volta caricato lo sketch su Arduino noterai che (in base alle condizioni di luce) il LED sarà acceso oppure spento.

Ora dovrai provare se tutta la configurazione è ok.

Se la fotoresistenza è esposta alla luce il LED dovrebbe essere spento, mentre coprendola il LED si dovrebbe accendere.

Il valore che è riportato nello sketch nella variabile **if(val<800)** è stato calcolato approssimativamente per i diversi tipi di fotoresistenze, per questo se il LED non si accende o non funziona correttamente, potrai modificare questo valore alzandolo e abbassandolo da 0 fino a 1024.

Inoltre, per vedere il valore che riceve la fotorsistenza, potrai cliccare sul monitor seriale nell'IDE di Arduino.

Questa funzione che hai realizzato con Arduino viene chiamata comunemente *crepuscolare* ed è utilizzata principalmente per illuminazioni esterne pubbliche e private.

8. LED RGB CON REGOLAZIONE MANUALE

Con questo progetto andrai a pilotare manualmente mediante 3 potenziometri e un LED RGB (deriva da RED – GREEN – BLUE, ovvero i colori che riesce a emettere).

Cosa ti occorre?

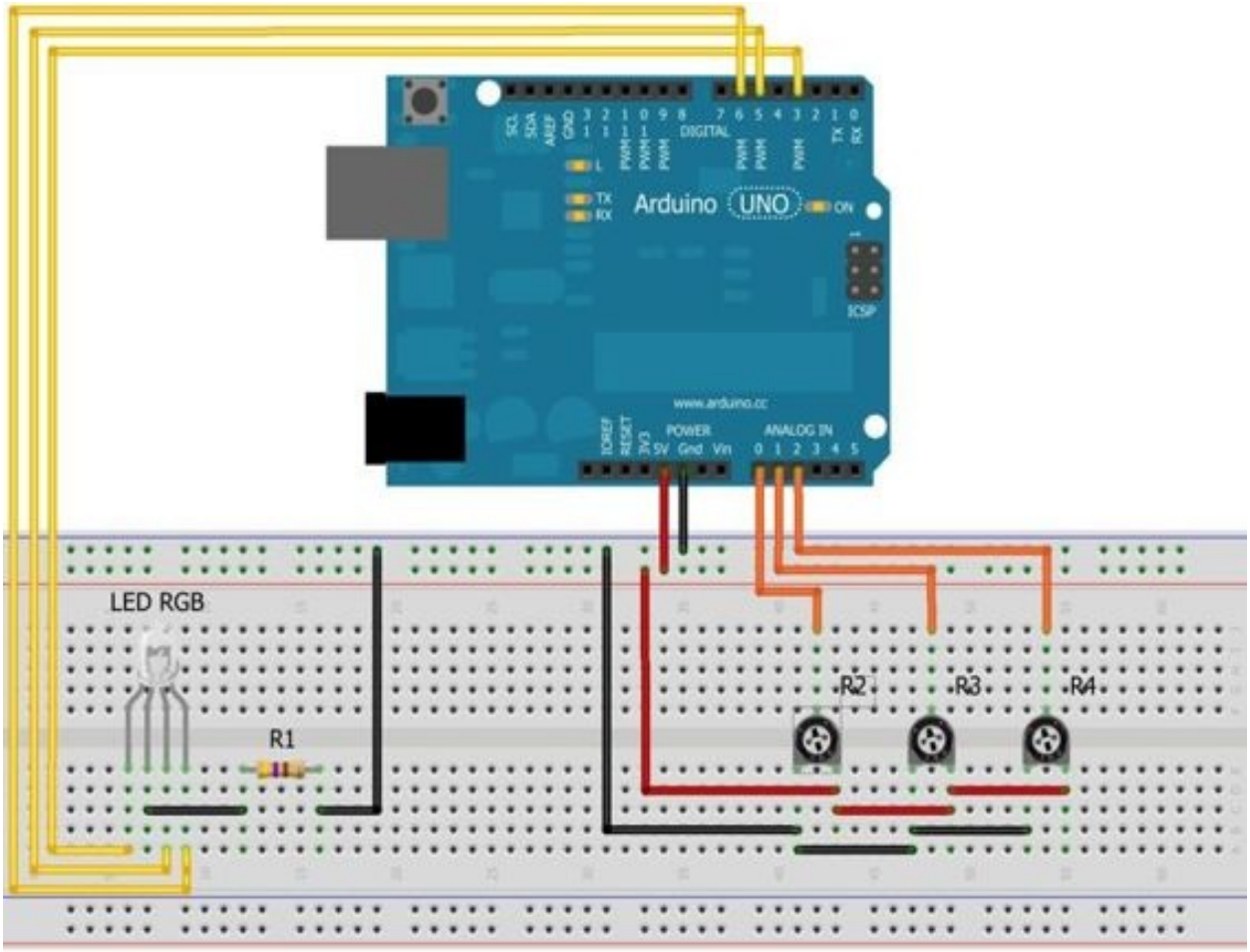
- 1 Arduino Uno
- 1 Breadboard
- Cavetti Jumper per fare i collegamenti
- 1 LED RGB
- 3 potenziometri da 10 KΩ
- 1 resistenza da 470 Ω

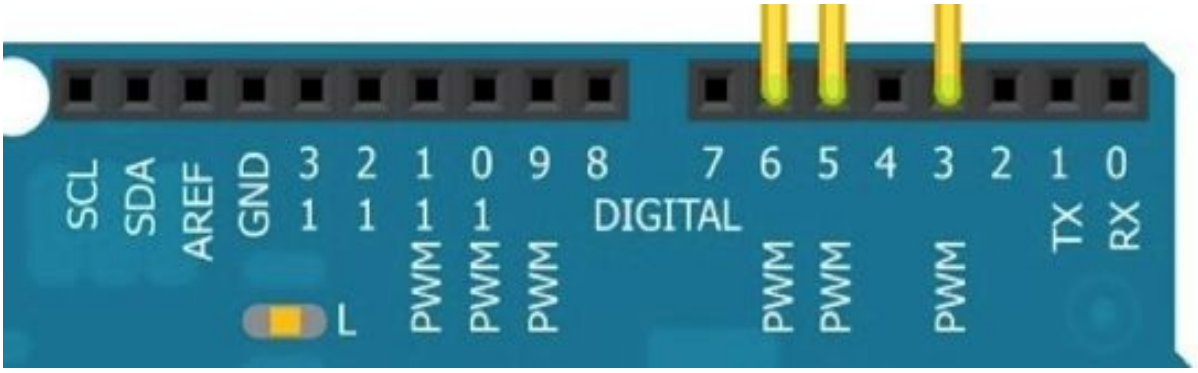
Il LED RGB contiene al suo interno 3 LED normali di colore rosso, verde e blu.

Questi 3 colori, se “dimmerati” (regolando la loro luminosità), posso dare a luogo a giochi di luci molto belli come andrai a realizzare.

Procedi con il collegamento come da schema.

Noterai che nel collegamento utilizzerai dei Pin specifici di Arduino, che sono contrassegnati dalla sigla PWM.





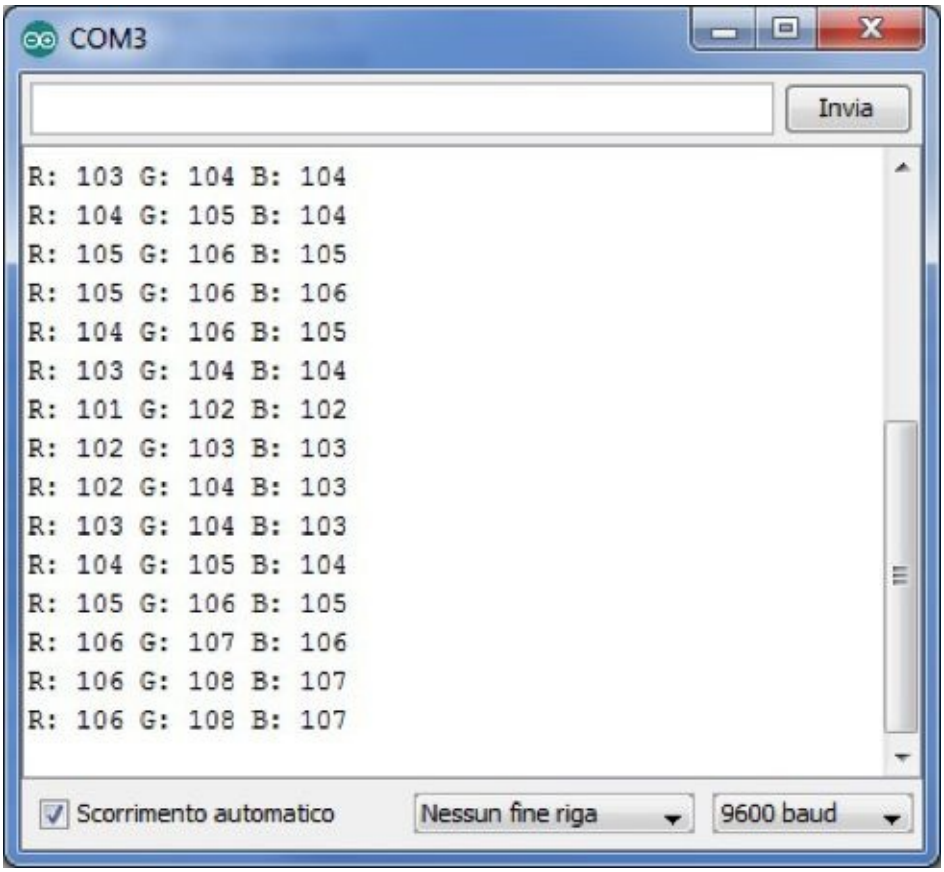
Questi Pin hanno la possibilità di fornire valori variabili e non fissi (come 0V – 5V) permettendoti così di regolare la luminosità di un LED.

In questo caso andrai a variare la luminosità dei 3 colori principali del LED RGB.

```
void setup()
{
  Serial.begin( 9600 );                          // Comunicazione monitor seriale

  pinMode( A0, INPUT );                          // Pin A0 come entrata
  pinMode( A1, INPUT );                          // Pin A1 come entrata
  pinMode( A2, INPUT );                          // Pin A2 come entrata
  pinMode( 3, OUTPUT );                          // Pin 3 PWM come uscita
  pinMode( 5, OUTPUT );                          // Pin 5 PWM come uscita
  pinMode( 6, OUTPUT );                          // PIN 6 PWM come uscita
}
 void loop()
{
  int R = map( analogRead( A0 ),0,1023,0,255 );  // Leggo valori di INPUT Pin A0 e lo converto da 0 - 1023 a 0 - 255
  int G = map( analogRead( A1 ),0,1023,0,255 );  // Leggo valori di INPUT Pin A1 e lo converto da 0 - 1023 a 0 - 255
  int B = map( analogRead( A2 ),0,1023,0,255 );  // Leggo valori di INPUT Pin A2 e lo converto da 0 - 1023 a 0 - 255

  Serial.print( "R: " );                          // Scrivo su seriale "R: "
  Serial.print( R );                             // Scrivo valore R
  Serial.print( " G: " );                        // Scrivo su seriale "G: "
  Serial.print( G );                             // Scrivo valore G
  Serial.print( " B: " );                        // Scrivo su seriale "B: "
  Serial.println( B );                           // Scrivo valore B e mando a capo

  analogWrite( 3, R );                           // Invia lo stesso valore letto da seriale sul Pin 3 PWM
  analogWrite( 5, G );                           // Invia lo stesso valore letto da seriale sul Pin 5 PWM
  analogWrite( 6, B );                           // Invia lo stesso valore letto da seriale sul Pin 6 PWM

  delay( 500 );                                  // Attendi 1/2 secondo prima di procedere con il Loop
}
```

Una volta che carichi lo sketch su Arduino, il processo si avvia e puoi notare che il LED – TX comincerà a lampeggiare al ritmo di ½ secondo, questo perchè sta trasmettendo i dati che potrai visualizzare in seriale premendo il relativo tasto Monitor Seriale presente sull'IDE.

Noterai che verranno scritti tutti i relativi valori dei diversi colori RGB come segue:



Questi valori si riferiscono alla regolazione dei potenziometri. Ruotandoli osserverai che i valori presenti sul monitor seriale cambieranno, proprio come la luminosità del LED RGB.

9. COMANDARE UN SERVOMOTORE

In questi progetto andrai a comandare un servomotore con Arduino.

I servomotori possono essere molto utili per effettuare automatismi e sono utilizzati nei modellini telecomandati.

Cosa ti occorre?

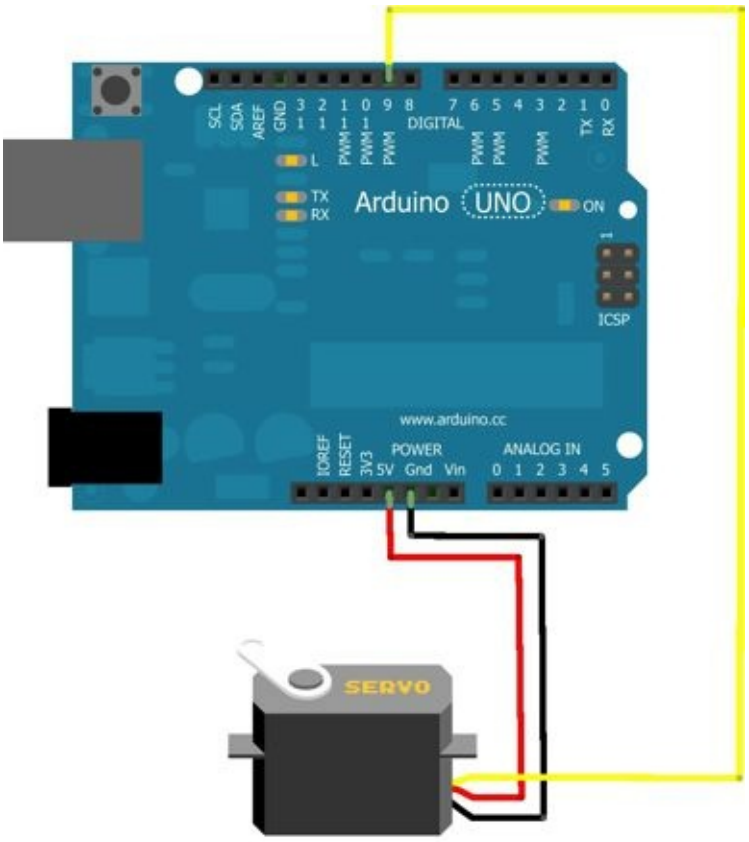
- 1 Arduino Uno
- 1 Servomotore

I servomotori sono composti da un motore che consente di effettuare il movimento e un sensore di posizione che indica esattamente in che punto si trova.

I servomotori hanno la caratteristica di ruotare normalmente per un massimo di 180° (alcuni possono ruotare fino a 360°).

Noterai che ogni servomotore ha 3 fili, solitamente **l'arancione (segnale)**, il **rosso (5V)** e il **nero (0V)** che individuano i relativi cavetti che andranno all'Arduino.

Procedi con il collegamento come da schema:



Il collegamento è molto semplice e la sequenza verrà impostata direttamente tramite PC.

```
#include <Servo.h>   // Includo la funzione servo (servomotore)

Servo SERVO;         // Associo la funzione Servo e la chiamo SERVO

void setup()

{
  SERVO.attach(9);   // Comunicazione SERVO sul Pin 9
}

void loop()

{
  SERVO.write(0);    // Porto il SERVO alla posizione 0°
  delay(1000);       // Attendo 1 secondo

  SERVO.write(90);   // Porto il SERVO alla posizione 90°
  delay(1000);       // Attendo 1 secondo

  SERVO.write(180);  // Porto il SERVO alla posizione 180°
  delay(1000);       // Attendo 1 secondo prima del loop
}
```

Ora non ti resta che caricare lo sketch, da subito il servomotore si imposterà alla sua posizione 0°. Successivamente, dopo 1 secondo, si regolerà a 90° e dopo un secondo a 180° e sempre dopo 1 secondo ritornerà a 0° e rieseguirà lo sketch a loop.

Potrai provare a modificare i valori 0° – 90° - 180° impostandoli a tuo piacimento come anche il tempo di attesa tra un processo e l'altro.

NB: ricordati che la rotazione espressa in gradi (°) deve essere compatibile con la rotazione del servomotore in uso.

10. COMANDARE UN SERVO MEDIANTE POTENZIOMETRO

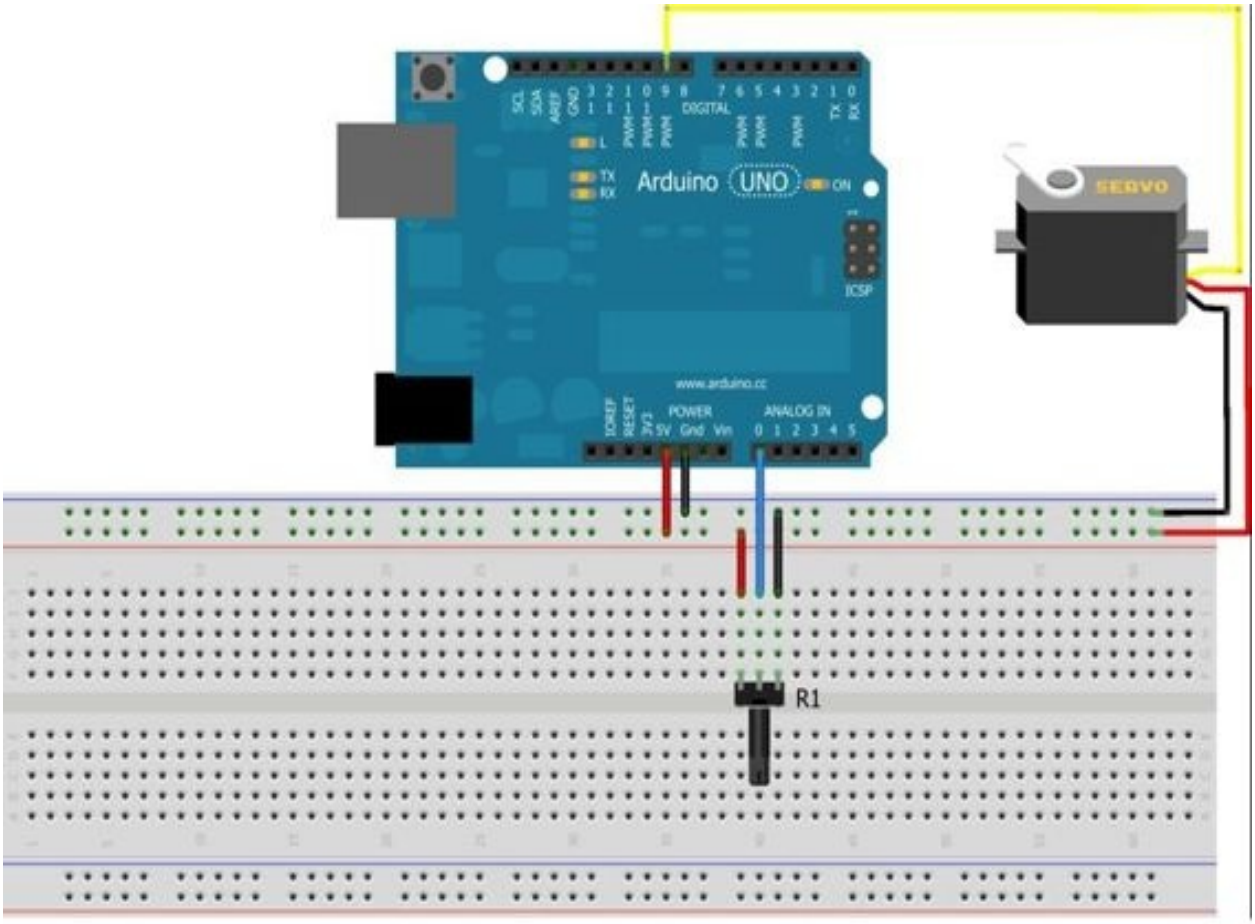
Questo progetto si interfaccia con il precedente, con la differenza che la regolazione della posizione del servomotore sarà manuale grazie a un potenziometro.

La scheda Arduino ci servirà in questo caso per leggere un valore di una resistenza variabile (potenziometro) e trasformarlo in un segnale da inviare al cavetto giallo del servomotore.

Cosa ti occorre?

- 1 Arduino Uno
- 1 Breadboard
- Cavetti Jumper per fare i collegamenti
- 1 Potenziometro da 10 KΩ
- 1 Servomotore

Procedi con il collegamento come da schema:



Il collegamento è simile a quello del progetto precedente, con l’aggiunta di una resistenza variabile (potenziometro) che chiameremo **R1**.

```
#include <Servo.h>                    // incudiamo i diversi dispositivi:

Servo SERVO;                          // Includiamo nella librearia il servomotore e lo chiamiamo SERVO

int R1 = A0;                          // Associa R1 (potenziometro) al Pin A0
int R1val;                            // Variabile di lettura del potenziometro (R1)

void setup()
{
  SERVO.attach(9);                    // Associa al Pin 9 il segnale del servomotore
}

void loop()
{
  R1val = analogRead(R1);             // Legge il valore analogico del potenziometro
  R1val = map(R1val, 0, 1023, 0, 179);// Mappo il valore analogico letto (0 - 1023) e lo porto a (0 - 179)
  SERVO.write(R1val);                 // Eseguo al servo il valore mappato
  delay(15);                          // Attendo 15 ms ed eseguo il loop
}
```

In questo sketch dobbiamo modificare il valore analogico letto dal potenziometro che varia da 0 a 1023, portandolo a valori compresi fra 0 e 179, ovvero la rotazione in gradi (°) che può eseguire il servomotore come mostrato di seguito.

```
R1val = map(R1val, 0, 1023, 0, 179);
```

Ora potrai caricare lo sketch dall'IDE sull'Arduino e, ruotando il potenziometro in senso orario e antiorario, osserverai contemporaneamente il movimento del servomotore alla tua stessa velocità.

Questa funzione è molto utilizzata nell'ambito del radicontrollo di macchine telecomandate e modellini vari.

11. COMANDARE UN SERVO MEDIANTE DUE PULSANTI

In questo progetto utilizzerai due pulsanti per comandare un servomotore.

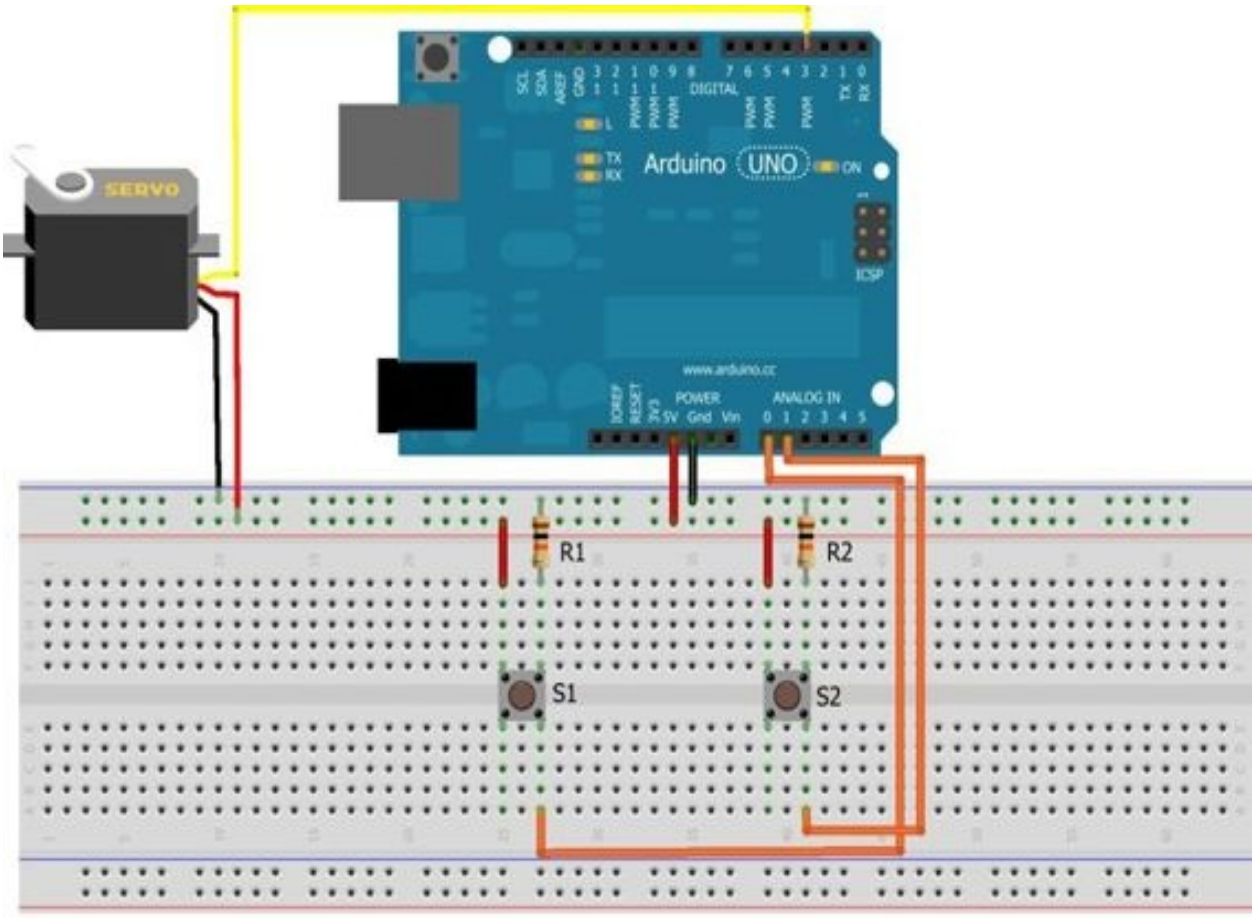
Questo sistema ti permette di far eseguire al servomotore tutte le posizioni possibili tenendo premuto uno dei due pulsanti alla volta.

Un pulsante ti permetterà di far girare in senso orario il servomotore e l'altro in senso antiorario.

Cosa ti occorre?

- 1 Arduino Uno
- 1 Breadboard
- Cavetti Jumper per fare i collegamenti
- 2 Resistenze da 10 KΩ
- 2 pulsanti del tipo NO
- 1 Servomotore

Procedi con il collegamento:



I pulsanti **S1** e **S2** serviranno a comandare la direzione del servomotore.

```
#include <Servo.h>

Servo SERVO;                                    // Servo oggetto

int GRADSERVO = 0;                              // Grado di partenza del servo = 0°
int MINSERVO = 0;                               // Grado minimo posizione servo = 0°
int MAXSERVO = 180;                             // Grado massimo posizione servo = 180°
int SERVOPIN = 3;                               // Pin di collegamento SERVO (cavo giallo)
int PREV = A1;                                  // Pin alalogico A1 (pulsante S1) indietro
int NEXT = A0;                                  // Pin alalogico A2 (pulsante S2) avanti

void setup()
{
  SERVO.attach( SERVOPIN );                     // Servo con riferimento al Pin 3
  SERVO.write( GRADSERVO );                     // Servo va in posizione grado di partenza = 0°

  pinMode( PREV, INPUT );                       // Imposta PREV (A1) come INPUT
  pinMode( NEXT, INPUT );                       // Imposta NEXT (A2) come INPUT
}
 void loop()
{
  if ( analogRead( PREV ) > 1000 ) { GRADSERVO--; }
  if ( analogRead( NEXT ) > 1000 ) { GRADSERVO++; }

  if ( GRADSERVO > MAXSERVO ) { GRADSERVO = MAXSERVO; }
  if ( GRADSERVO < MINSERVO ) { GRADSERVO = MINSERVO; }

  SERVO.write( GRADSERVO );
  delay(15);                                    // Attende 15 ms
}
```

Analizziamo la funzione loop:

```
 void loop()
{
  if ( analogRead( PREV ) > 1000 ) { GRADSERVO--; }
  if ( analogRead( NEXT ) > 1000 ) { GRADSERVO++; }

  if ( GRADSERVO > MAXSERVO ) { GRADSERVO = MAXSERVO; }
  if ( GRADSERVO < MINSERVO ) { GRADSERVO = MINSERVO; }

  SERVO.write( GRADSERVO );
  delay(15);
}
```

If è la funzione di controllo che se riceve un segnale analogico dal parametro

PREV (pulsante S1) maggiore (>) di 1000 (il massimo valore è di 1023) diminuisce GRADSERVO. Se riceve un segnale analogico dal parametro NEXT (pulsante S2) maggiore (>) di 1000 aumenta GRADSERVO.

```
if ( analogRead( PREV ) > 1000 ) { GRADSERVO--; }
if ( analogRead( NEXT ) > 1000 ) { GRADSERVO++; }
```

Con l'aggiunta della seconda funzione if verificherai che se GRADSERVO è maggiore (>) di MAXSERVO (180°) imposti GRADSERVO = MAXSERVO = 180°

La stessa funzione viene applicata se il valore è minore di MINSERVO, come riportato sotto.

```
if ( GRADSERVO > MAXSERVO ) { GRADSERVO = MAXSERVO; }
if ( GRADSERVO < MINSERVO ) { GRADSERVO = MINSERVO; }
```

Anche questo progetto viene utilizzato diffusamente nel campo del modellismo e radiocomando di apparecchiature.

12. LCD con Arduino

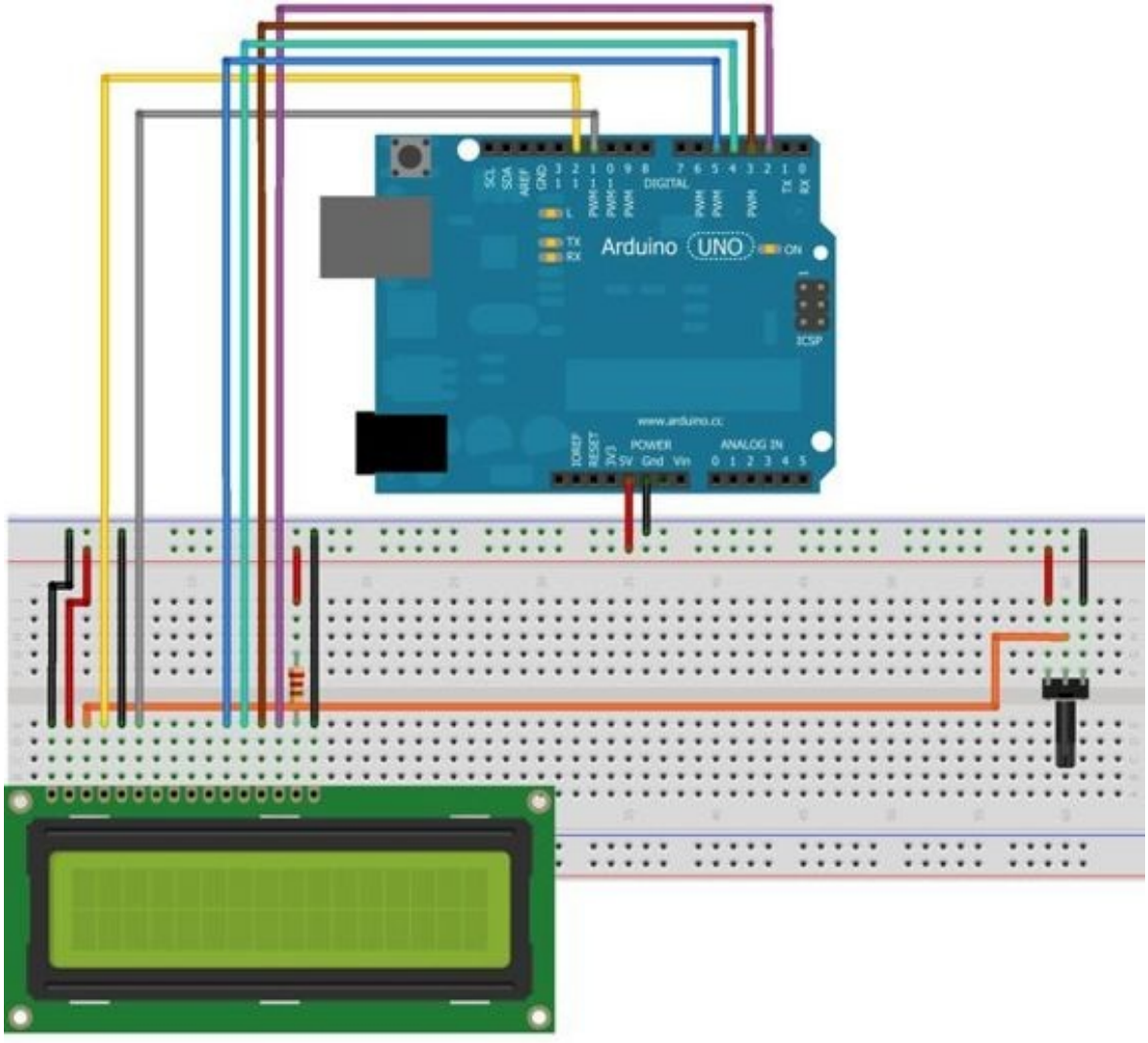
Arduino è utilizzato spesso nella gestione di uno schermo LCD (display a cristalli liquidi). In questo caso utilizzerai un LCD molto comune, il Hitachi HD44780.

Questo LCD ha la caratteristica di 16 colonne e 2 righe su cui poter scrivere e visualizzare lettere e numeri, che andrai a impostare nello sketch.

Cosa ti occorre?

- 1 Arduino Uno
- 1 Breadboard
- Cavetti Jumper per fare i collegamenti
- 1 Potenziometro da 10 KΩ
- 1 schermo LCD Hitachi HD44780

Procedi con il collegamento, facendo molta attenzione a non commettere errori.



Fai molta attenzione, uno sbaglio potrebbe causare danni permanenti all'LCD e ad Arduino.

Il potenziometro ti servirà per regolare il contrasto dell'LCD in modo da visualizzare in modo ottimale le scritte che compariranno.

```
#include <LiquidCrystal.h>              // Incudiamo nella libreraria LiquidCrystal (LCD)

LiquidCrystal LCD(12, 11, 5, 4, 3, 2); // Chiama LiquidCrystal = LCD e lo associa ai Pin BUS

void setup()
{
  LCD.begin(16, 2);                    // Specifica il tipo di LCD che è formato da 16 colonne e 2 righe
  LCD.print("Ciao mondo!");            // Scrive sull'LCD la scritta indicata
}
void loop()
{
  LCD.setCursor(0, 1);                 // Imposta sul cursore 0, 1 la scritta
}
```

Carica lo sketch e da subito dovrebbe essere visibile la scritta sull'LCD.

Se non riesci a visualizzarla in maniera ottimale, potrai ruotare in senso orario o antiorario il potenziometro, in modo da aumentare e diminuire il contrasto.

È possibile anche modificare il testo sull'LCD modificando il valore in blu "Ciao mondo".

Ricordati di lasciare le (") prima e dopo il testo che vuoi visualizzare.

13. LCD E SENSORE DI TEMPERATURA

In questo progetto aggiungerai il sensore di temperatura TMP36, in modo da poter visualizzare sull'LCD la temperatura misurata.

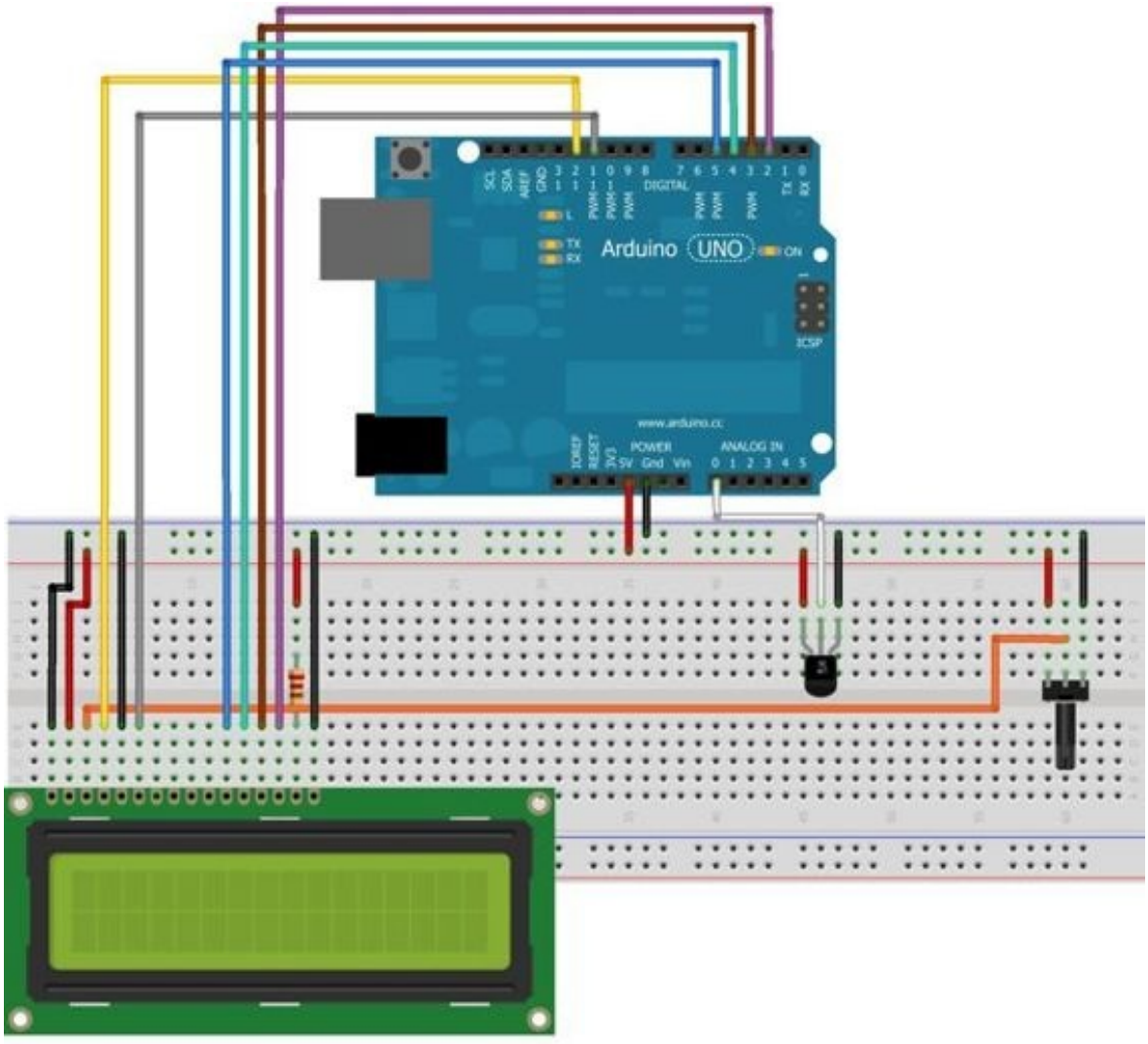
Questo tipo di sensore è uno dei più precisi e ha la possibilità di lavorare a tensioni bassissime 2,2V – 5V.

Il sensore è in grado di misurare valori da -40 C° fino a un massimo di 125 C°.

Cosa ti occorre?

- 1 Arduino Uno
- 1 Breadboard
- Cavetti Jumper per fare i collegamenti
- 1 sensore di temperatura TMP36
- 1 potenziometro da 10 KΩ
- 1 LCD 16 X 2

Procedi con il collegamento, facendo molta attenzione a non sbagliarti.



Fai attenzione nel collegare correttamente il sensore di temperatura, rispettando la piedinatura.

Un collegamento errato potrebbe far surriscaldare il sensore e danneggiarlo permanentemente.

```
#include <LiquidCrystal.h>

LiquidCrystal LCD(12, 11, 5, 4, 3, 2);  // LCD Pin BUS

void setup()
{
  LCD.begin(16, 2);                     // Colonne e righe LCD
}
void loop()
{
  int TEMP = analogRead(A0);            // Lettura analogica TEMP (sensore di temperatura) Pin A0
  float volt = (TEMP/1020.0) * 4.9;     // Lettura volt
  float tempC = (volt -0.5) * 100;      // Centigradi conversione
  LCD.setCursor(0, 0);                  // Prima righa cursore
  LCD.print("Temperatura:");            // Scrive ("Temperatura:")
  LCD.setCursor(0, 1);                  // Seconda righa cursore
  LCD.print(tempC);                     // Scrive valore tempertura letto
  LCD.print(" C");                      // Scrive ("C")
  delay(3000);                          // Attende 3 secondi ed esegue loop
}
```

Carica lo sketch e visualizzerai la temperatura in °C letta dal sensore sullo schermo LCD.

Nello sketch è presente la funzione float, che contiene le formule di conversione da segnale elettrico a valore numerico in C°.

Questi valori sono standard per questo tipo di sensore di temperatura.

14. COMANDARE UN MOTORE PASSO-PASSO

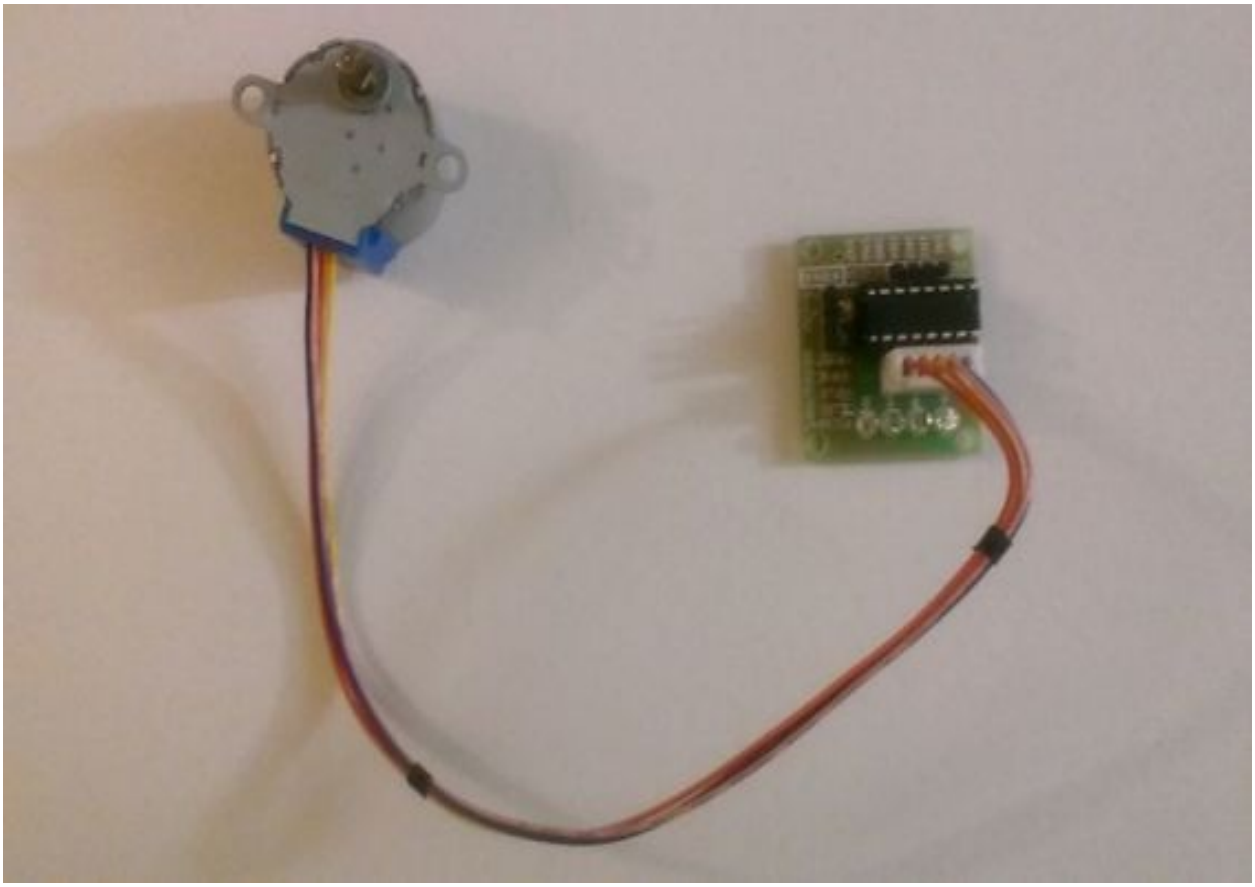
Il progetto seguente ti consentirà di comandare un motore passo-passo.

Questo tipo di motore (a differenza di un motore standard) ha la possibilità di regolare la velocità e specificare il punto in cui si deve fermare con precisione.

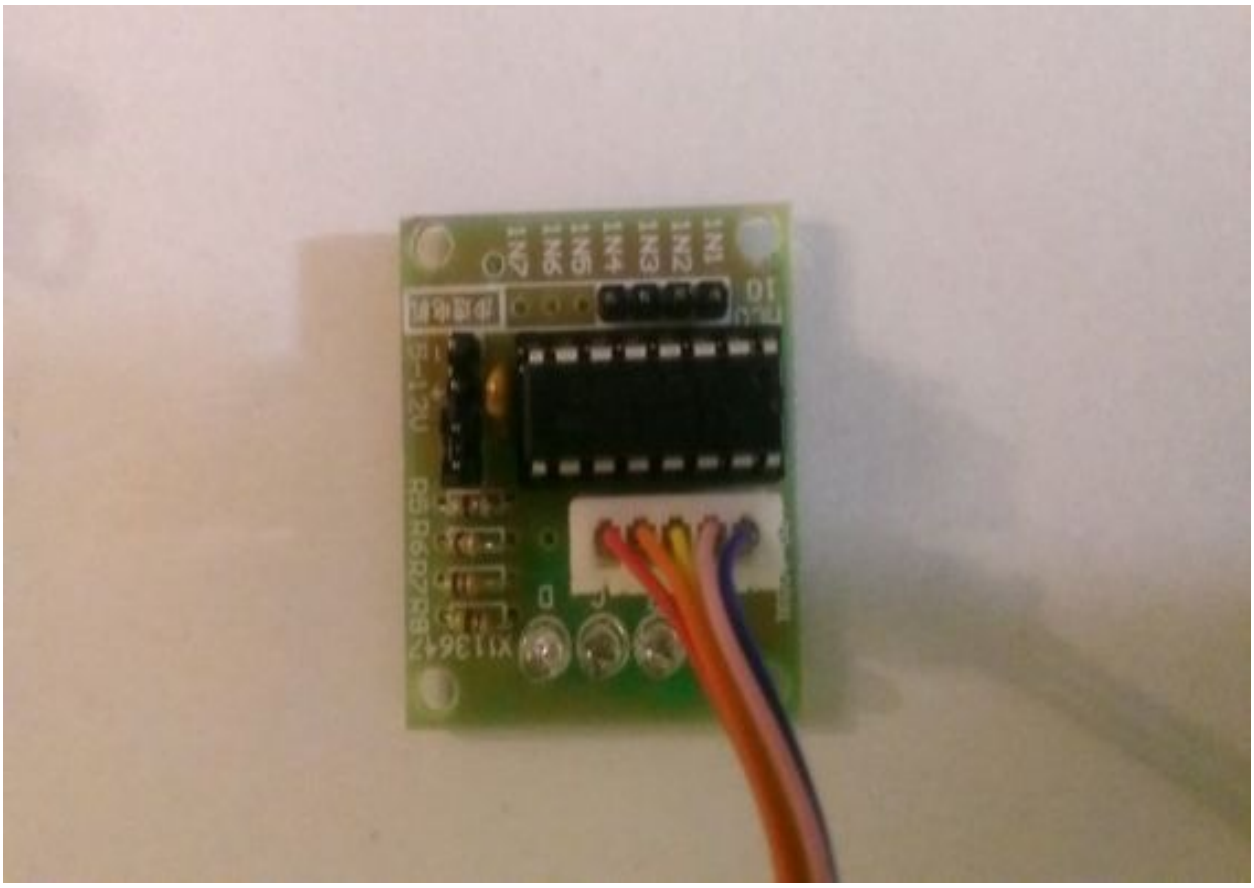
Il motore passo-passo viene anche chiamato **step** o **stepper**, questo perchè può suddividere la sua rotazione in diversi passi (step).

Questi motori possono essere comandati con estrema precisione, e non richiedono l'uso di sensori per identificare la loro posizione; essi vengono utilizzati frequentemente nell'ambito della robotica e nei servomeccanismi in genere.

Il motore passo-passo che andrai a utilizzare è il **28BYJ-48** che è il più comune usato con Arduino.



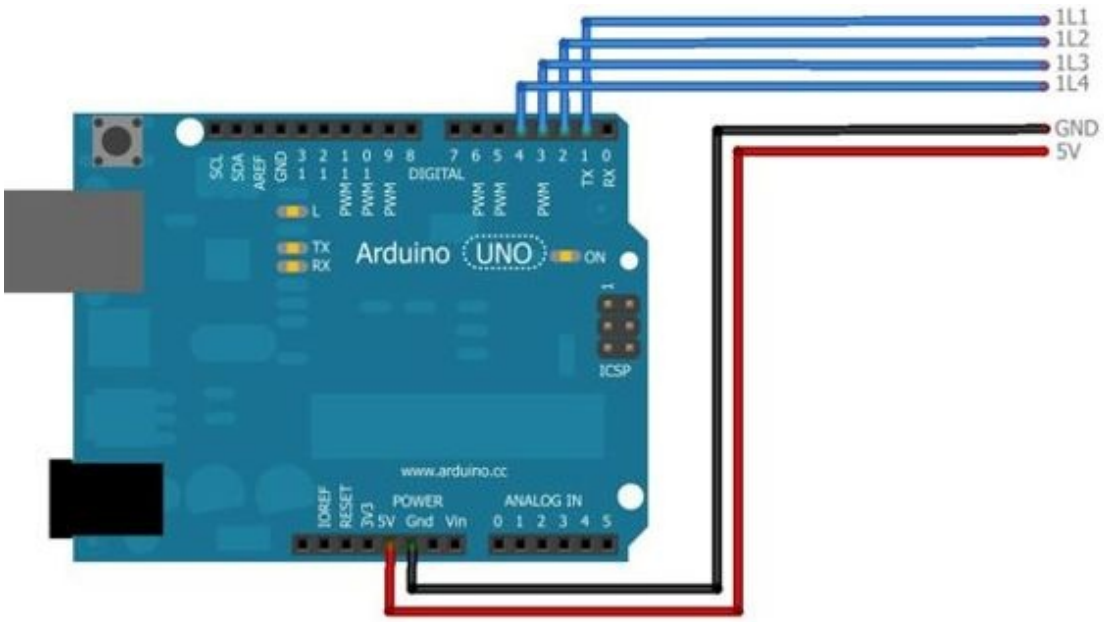
Questo tipo di motore passo-passo contiene una scheda driver, che svolge la funzione di compilazione dei segnali che serviranno al motore.



Cosa ti occorre?

- 1 Arduino Uno
- Cavetti Jumper per fare i collegamenti
- 1 motore passo-passo 28BYJ-48

Procedi con il collegamento, attento a rispettare la piedinautura sulla scheda driver del motore all’Arduino.



Collega i cavetti Jumper, dall'Arduino alla scheda driver prestando attenzione alla piedinatura scritta sopra.

```
#include   <Stepper.h>          // Include libreria Stepper

Stepper STEPPER(2048,2,4,3,5); // Chiamiamo il motore = STEPPER formato da 2048 step e lo assegniamo ai relativi Pin

void setup()
{
  STEPPER.setSpeed(10);        // Imposta la velocità dello stepper = 10
}
void loop()
{
  STEPPER.step(2048);          // Avvia il motore avanti a fare 1 giro = 2048 step
  delay(1000);                 // Ferma il motore per 1 secondo
  STEPPER.step(-2048);         // Avvia il motore indietro a fare 1 giro = 2048 step
  delay(1000);                 // Ferma il motore per 1 secondo ed esegue loop
}
```

Carica lo sketch e si avvierà la sequenza del motore.

NB: se la sequenza non dovesse funzionare, potrebbe essere presente una modifica all'hardware della scheda drive che dovrai sostituire con il codice:

```
Stepper STEPPER(2048,2,3,4,5);
```

15. GENERARE UNA MELODIA

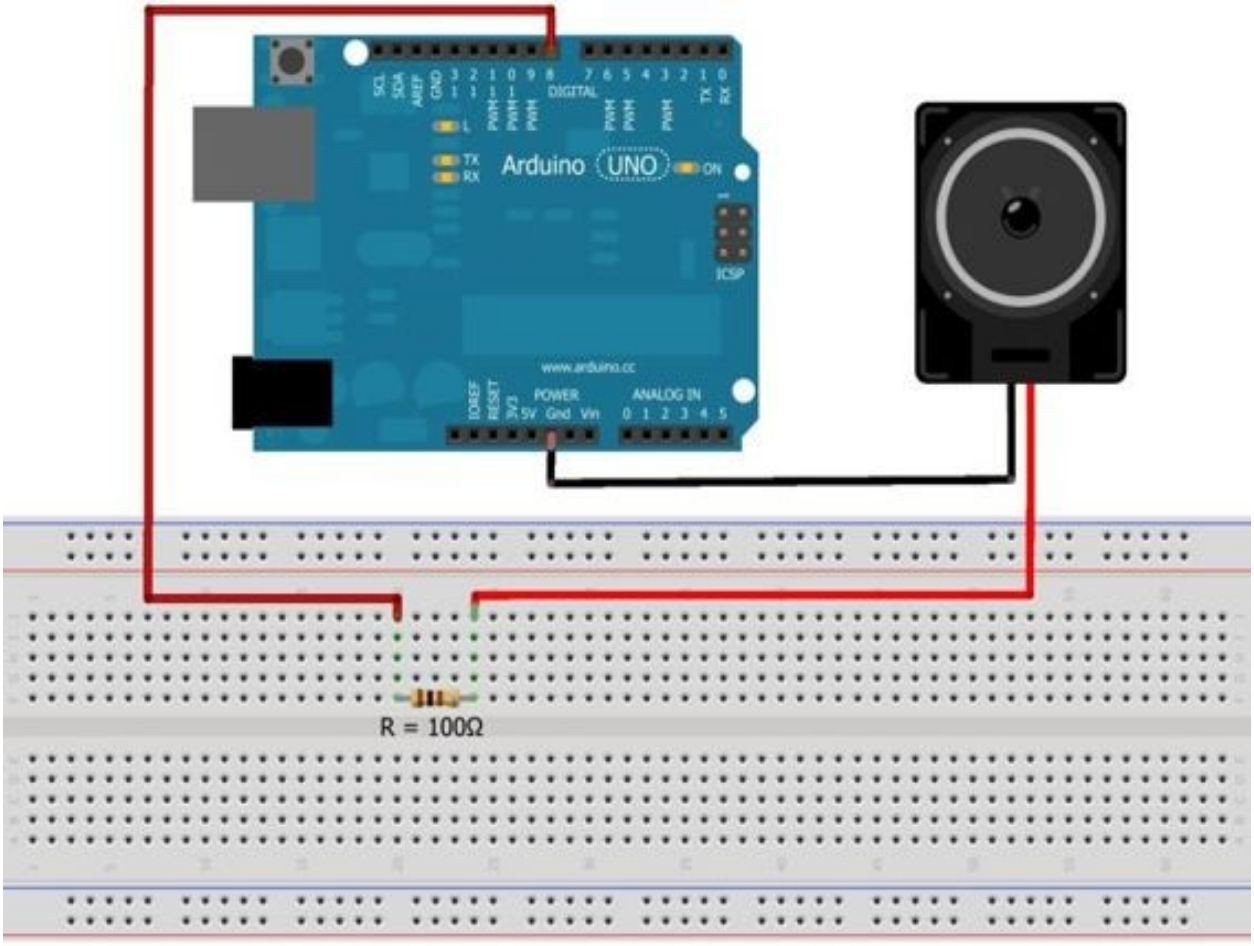
Una delle capacità di Arduino è quella di generare frequenze che se utilizzate con un diffusore audio potranno essere ascoltate.

Arduino non è in grado di fornire un onda sinusoidale, come quelle fornite dai normali stereo, ma solo onde quadre, quindi l'unica differenza è che sentiremo un suono del tipo "robotico-meccanico" tipico dei giocattoli dei bambini.

Cosa ti occorre?

- 1 Arduino Uno
- Cavetti Jumper per fare i collegamenti
- 1 Breadboard
- 1 diffusore audio da 8Ω
- 1 resistenza da 100 Ω

Il collegamento è molto semplice, basta collegare il Pin 8 di uscita del segnale con la resistenza che a sua volta porterà il segnale attenuato al diffusore audio.


Arduino UNO
www.arduino.cc
DIGITAL
POWER
ANALOG IN
ICSP
R = 100Ω

```
int speakerPin = 8;                                          // Uscita Pin 8 cassa

int length = 15;                                             // Numero di note
char notes[] = "ccggaagffeeddc ";                            // Note numeriche e lo spazio rappresnta la pausa
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };  // Durata delle note
int tempo = 300;                                             // Durata di una nota da 1 = 300 ms

void playTone(int tone, int duration) {                      // Parametro di riproduzione musicale
  for (long i = 0; i < duration * 1000L; i += tone * 2) {    // Codice per decifrare le note e riprodurle
    digitalWrite(speakerPin, HIGH);                          // Settaggio valore CASSA = ALTO
    delayMicroseconds(tone);                                 // Attendi microsecondi per il riprodurre tono
    digitalWrite(speakerPin, LOW);                           // Settaggio valore CASSA = BASSO
    delayMicroseconds(tone);                                 // Attendi microsecondi per il riprodurre tono
  }
}

void playNote(char note, int duration) {                     // Parametro di riproduzione musicale
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };  // Valore letterale note
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 }; // Frequenza delle note indicate precedentemente

// CODICE DI RIPRODUZIONE DELLE NOTE INDICATE PRECEDENTEMENTE
  for (int i = 0; i < 8; i++) {
    if (names[i] == note) {
      playTone(tones[i], duration);
    }
  }
}



void setup() {
  pinMode(speakerPin, OUTPUT);                               // Pin 8 come OUTPUT
}

void loop() {
  for (int i = 0; i < length; i++) {
    if (notes[i] == ' ') {
      delay(beats[i] * tempo);
    } else {
      playNote(notes[i], beats[i] * tempo);
    }

    delay(tempo / 2);                                        // Pausa tra le note riprodotte = 150 ms
  }
}
```

Questo sketch racchiude tutte le funzioni per la generazione delle frequenze di uscita dal Pin 8.

Ogni frequenza diversa corrisponde a una nota diversa, combinate insieme danno origine a una melodia.

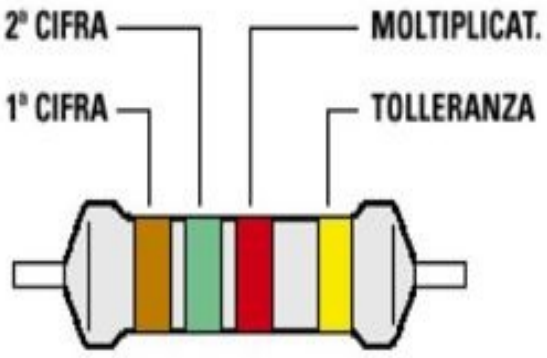
Download

Per il download degli sketch clicca o fai touch sul seguente link:
http://bit.ly/1zMZ45Q

VALORI RESISTENZE

In questa tabella sono mostrati tutti i valori delle resistenze che utilizzerai.

	1ª CIFRA	2ª CIFRA	MOLTIPLICAT.	TOLLERANZA
NERO	====	0	x 1	10 % ARGENTO
MARRONE	1	1	x 10	5 % ORO
ROSSO	2	2	x 100	
ARANCIONE	3	3	x 1.000	
GIALLO	4	4	x 10.000	
VERDE	5	5	x 100.000	
AZZURRO	6	6	x 1.000.000	
VIOLA	7	7	ORO : 10	
GRIGIO	8	8		
BIANCO	9	9		



Le resistenze sono del tipo a 4 bande come mostrato in figura.

NOTA BIOGRAFICA

Mattia Valsania nasce a Torino il 27 aprile 1996 e vive a San Damiano d'Asti.

Fin da piccolo è attratto dal funzionamento degli apparecchi elettronici, dalla musica e dalla realizzazione di siti web.

Frequenta perito elettrotecnico un I.T.I.S. ad Asti e studia su internet, documentandosi sulla realizzazione di progetti e circuiti elettronici; da lì arriva la scoperta della programmazione, della domotica e di Arduino.

Nel 2012 nasce Valsania Network che si occupa della realizzazione di siti web e servizi.

Nel 2014, dopo aver frequentato 4 anni dell'I.T.I.S, deluso dalla formazione, si iscrive a un corso per la formazione professionale.

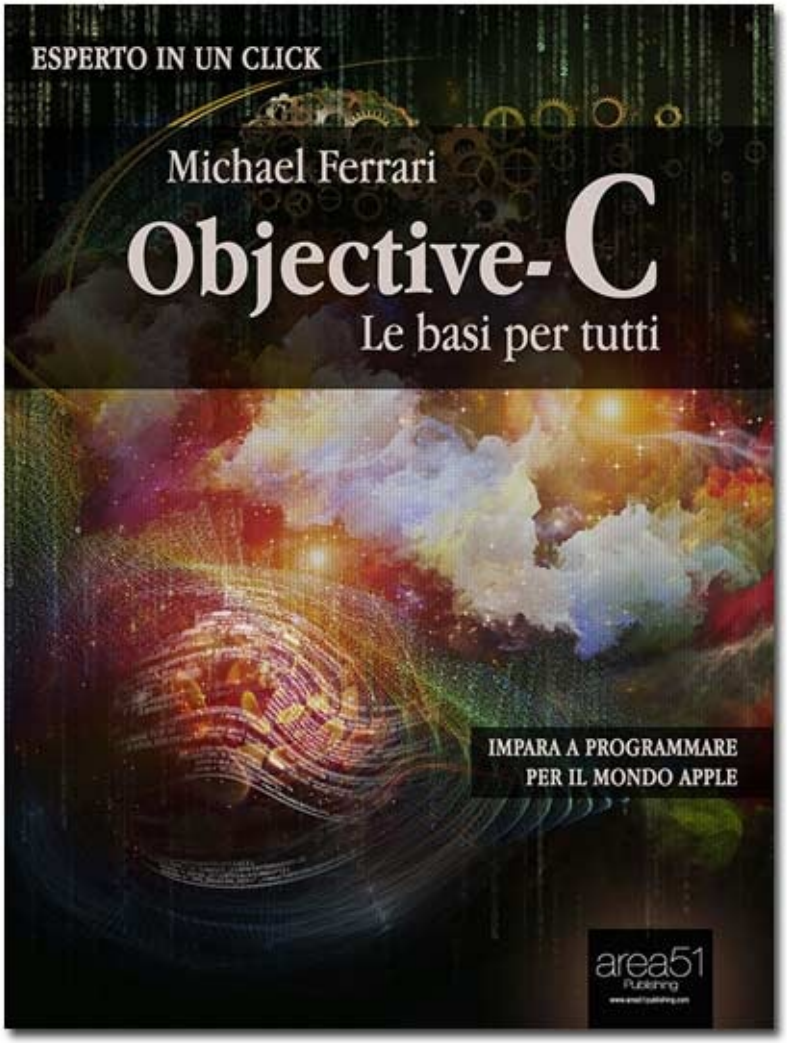
Collabora con la casa editrice Area51 Publishing e studia presso il conservatorio di Alessandria.

Maggiori informazioni e curiosità sulle sue attività sono reperibili su www.mattiavalsania.com

CLICCA O FAI TOUCH QUI

Esperto in un click


ESPERTO IN UN CLICK
Mirco Baragiani
Programmazione C
Le basi per tutti
I FONDAMENTI PER CHI
VUOLE DIVENTARE PROGRAMMATORE
area51
Publishing


ESPERTO IN UN CLICK
Michael Ferrari
Objective-C
Le basi per tutti
IMPARA A PROGRAMMARE
PER IL MONDO APPLE
area51
Publishing


ESPERTO IN UN CLICK
Michael Ferrari
C#
LE BASI PER TUTTI
IMPARA A PROGRAMMARE
PER IL MONDO .NET
area51


Esperto in un click
Michael Ferrari
Java
Le basi per tutti
area51
Publishing

ALBERTO MISURACA

HTML

PRATICO PER WEB

area51
Publishing


ALBERTO MISURACA
HTML
PRATICO
PER WEB APP
orientamento e design
App Web - native
API PhoneGap
jQuery
jQuery
area51 Publishing


CORONA SDK
+ iOS


ESPERTO IN UN CLICK
Mirco Baragiani
Corona SDK
VIDEOCORSO
MODULO BASE
1 2 3
60 MIN. DI
VIDEOTUTORIAL
area51


ESPERTO IN UN CLICK
Mirco Baragiani
Corona SDK
VIDEOCORSO
MODULO BASE
1 2 3
60 MIN. DI
VIDEOTUTORIAL
area51


ESPERTO IN UN CLICK
Mirco Baragiani
Corona SDK
VIDEOCORSO
MODULO BASE
1 2 3
60 MIN. DI
VIDEOTUTORIAL
area51

Indice