

Workshop Introduttivo Arduino + Progetto di Natale



**FAB
LAB**
PALERMO

2017
FabLab Palermo

Scopo del Workshop

- **Apprendere le basi di Arduino**
 - Cos'è
 - A cosa serve
 - Come funziona
- **Imparare a programmare Arduino**
 - Utilizzare le istruzioni di base per i primi programmi
- **Progetto tematico**
 - Studiamo la libreria Neopixel per Led concatenabili
 - Creare un'animazione luminosa con i Led per il Gadget natalizio
 - Assemblare il Gadget natalizio

Modalità di svolgimento

- **Durata 4 ore**

- 15:00 – 17:00 Teoria
- 17:00 – 19:00 Pratica e Progetto Natalizio

- **Parte teorica**

- Proiezione delle slide e approfondimenti teorici del tutor
- Le domande sono benvenute
- Eventuali approfondimenti vanno richiesti alla fine

- **Progetto**

- Utilizzando le schede Arduino e le strisce di Smart Led concatenabili messe a disposizione dei partecipanti, costruiremo un gadget Natalizio luminoso.

Riferimenti e file del corso

- **Slideshow del corso**

- Sul gruppo Facebook del Fablab Palermo
- Chiavetta durante il workshop

- **Sketch degli esempi**

- Sul gruppo Facebook del Fablab Palermo
 - Chiavetta durante il workshop

- **Arduino IDE**

- <http://www.arduino.cc/download>

Introduzione ad Arduino



Cos'è Arduino ?

- **Arduino**

- Arduino è una scheda elettronica con un **microcontrollore** e la circuiteria necessaria per il suo funzionamento, utile per creare rapidamente **prototipi** per scopi hobbistici e didattici.
- Esistono **diverse versioni**: Uno, Nano, Mega, etc.. che si differenziano per le prestazioni, la connettività e il numero di ingressi/uscite.

- **Gli Sketch**

- L'insieme delle istruzioni che scriviamo per creare la nostra applicazione: il **programma**. Il nostro riferimento è l'ambiente di sviluppo integrato meglio conosciuto come **Arduino IDE**

Per chi è pensata Arduino ?

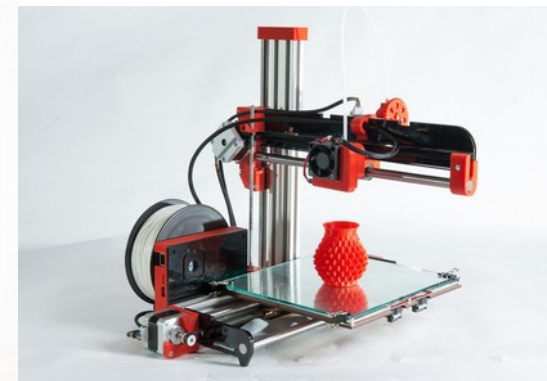
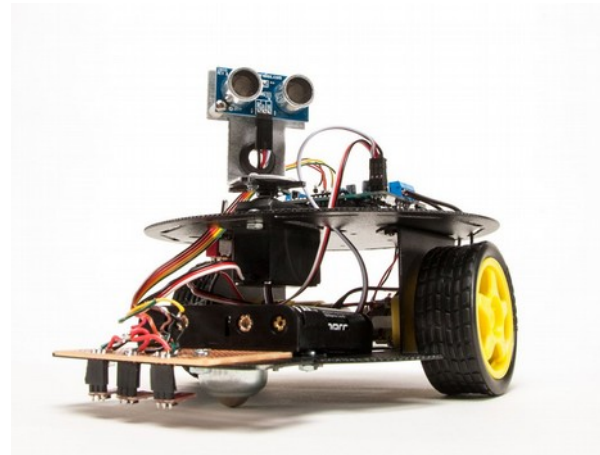
L'hardware Arduino e il software di programmazione costituiscono un semplice e potente sistema di prototipazione elettronica adatto per:

- **Artisti**
- **Designer**
- **Studenti**
- **Maker**

tutti coloro che intendano creare progetti che prevedono il controllo di dispositivi e l'interazione con l'ambiente esterno.

Possibili applicazioni

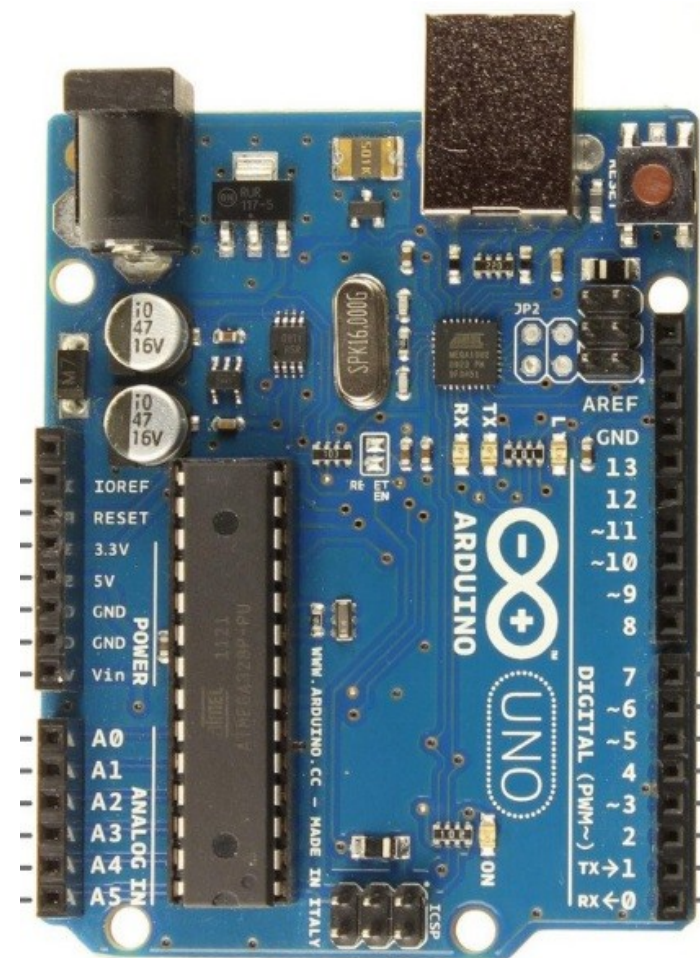
- **Didattica**
- **Prototipazione**
- **Hobbistica**
- **Installazioni artistiche**
- **Domotica**
- **Robotica**
- **Droni**
- **Stampanti 3D**



La scheda elettronica

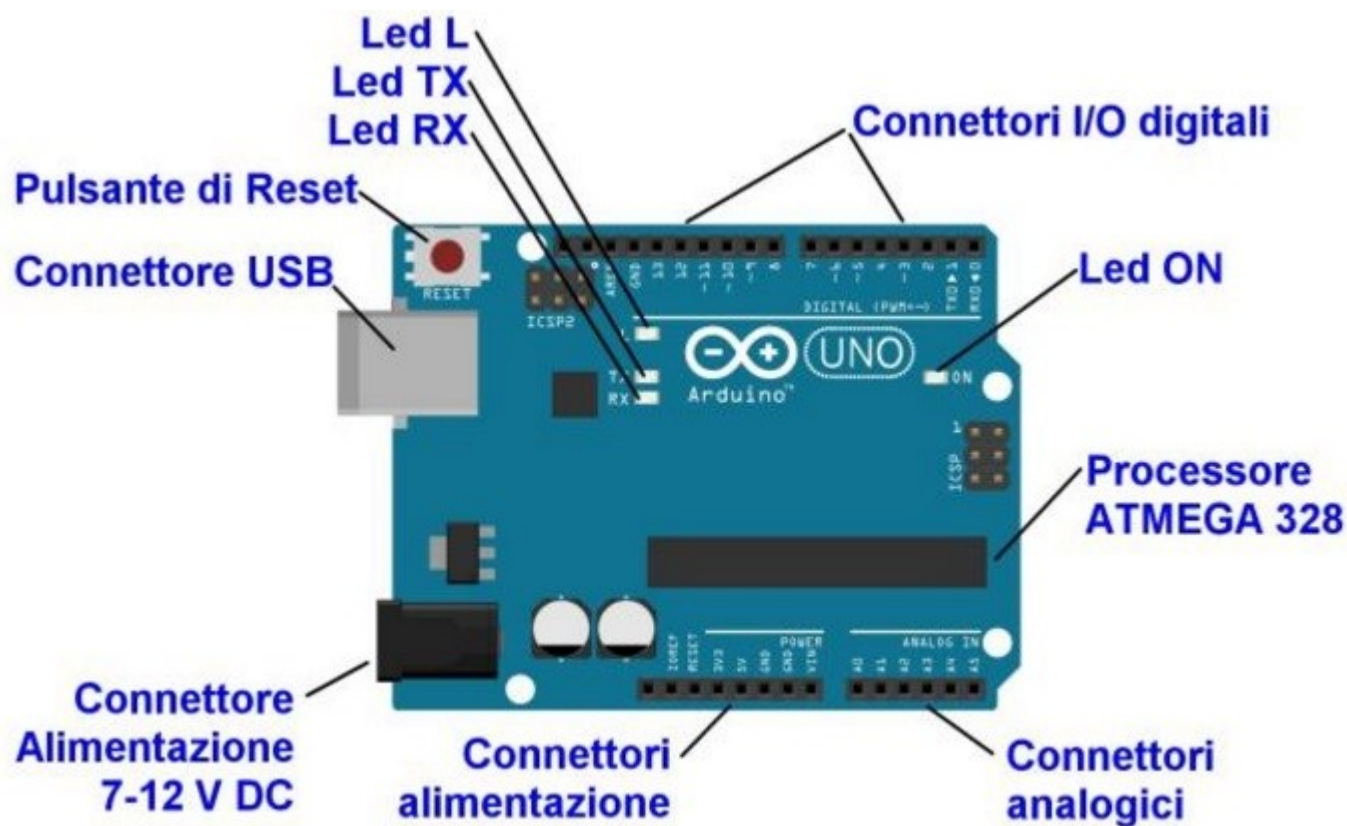
- **Caratteristiche tecniche:**

| | |
|-----------------------------|---|
| Microcontroller | ATmega168/328 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limits) | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 40 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader |
| SRAM | 1 KB (ATmega168) or 2 KB (ATmega328) |
| EEPROM | 512 bytes (ATmega168) or 1 KB (ATmega328) |
| Clock Speed | 16 MHz |



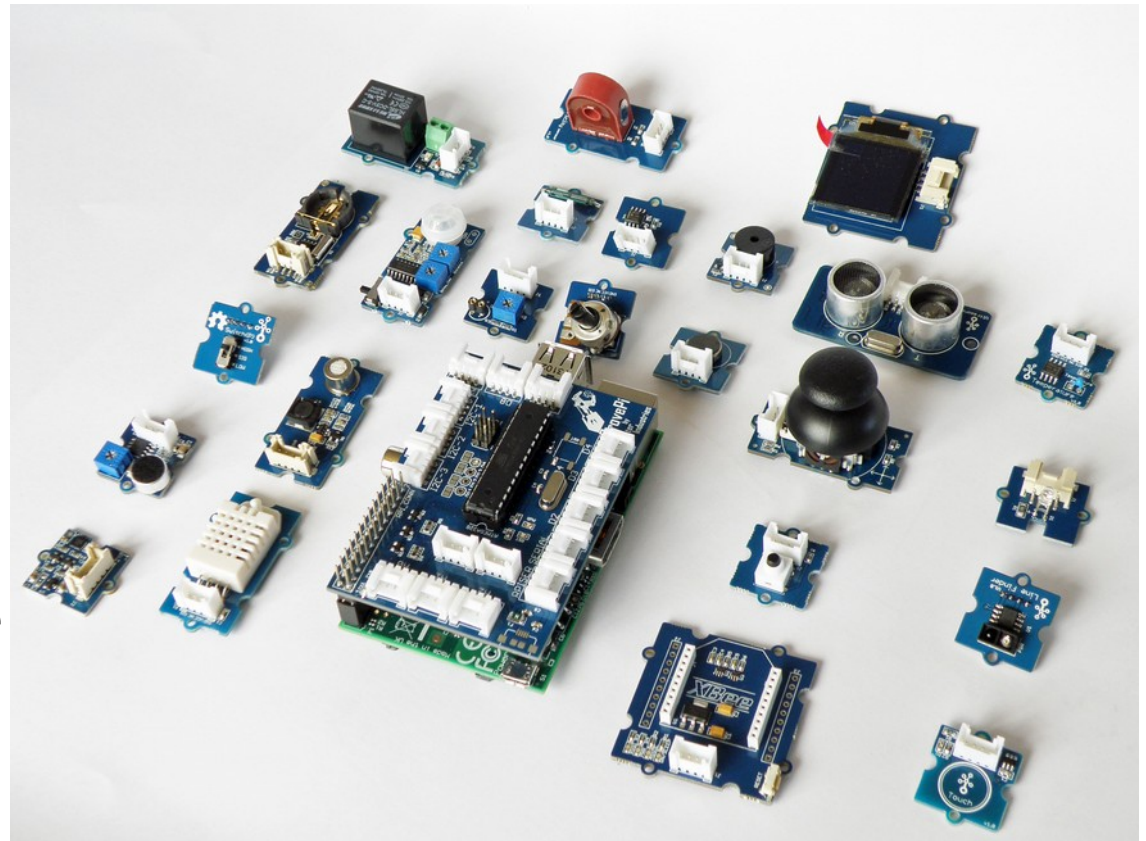
La scheda elettronica

- **Scopriamo la scheda:**



La scheda elettronica

- **Posso connettere ad Arduino diversi sensori:**
- **Ambientali**
 - Temperatura
 - Umidità
 - Pressione atmosferica
 - etc
- **Prossimità**
 - Infrarossi
 - Ultrasonici
 - Rumore ambientale
- **Biometrici**
 - Battito cardiaco
 - Temperatura



La scheda elettronica

- Posso **connettere ad Arduino diversi attuatori:**
- **Led**
 - Spie, illuminazione...
- **Speaker**
 - Suoni, allarmi..
- **Motori**
 - Stepper, servo, DC
- **Display**
 - LCD, OLED, etc.
- **Relay**
 - Power on/off



L'ambiente di sviluppo IDE

- **Dove scaricare il file:**

Arduino - Home

<https://www.arduino.cc/> ▼ Traduci questa pagina

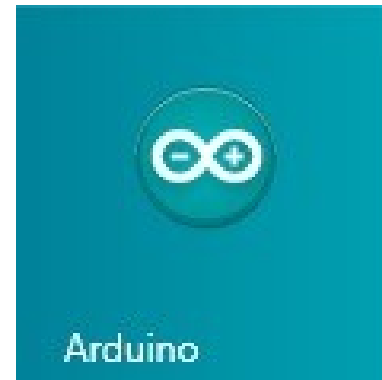
Open-source electronic prototyping platform enabling users to create interactive electronic objects.



The screenshot shows the Arduino website's download page. At the top is a teal navigation bar with the Arduino logo, links for Buy, Download, Products, Learning, Forum, Support, and Blog, and buttons for LOG IN and SIGN UP. Below the navigation bar is the heading "Download the Arduino Software". The main content area features the Arduino logo on the left and text on the right stating "ARDUINO 1.6.12". The text describes the IDE as open-source, written in Java, and based on Processing. It mentions compatibility with Windows, Mac OS X, and Linux, and refers to the "Getting Started" page for installation instructions. On the right side of the page, there are several download options: "Windows Installer", "Windows ZIP file for non admin Install", "Windows app (Microsoft Store)" with a "Get" button, "Mac OS X 10.7 Lion or newer", "Linux 32 bits", "Linux 64 bits", and "Linux ARM (experimental)".

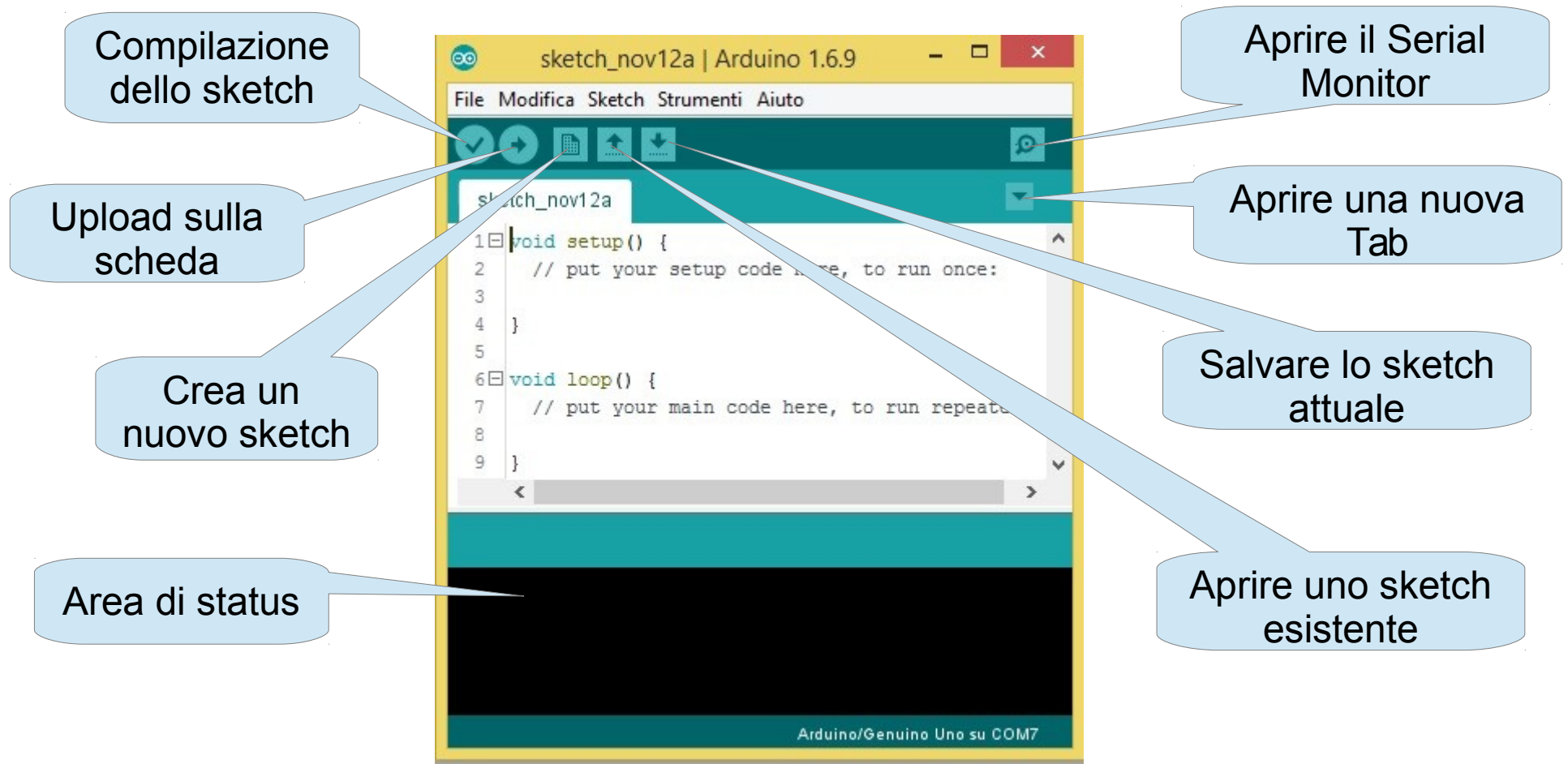
L'ambiente di sviluppo IDE

- Connettere Arduino al Pc tramite il cavo USB
- Installare Arduino IDE come un normale programma
- Installare il driver se non presente sul PC
- Avviare l'IDE di programmazione facendo doppio click sull'icona Arduino



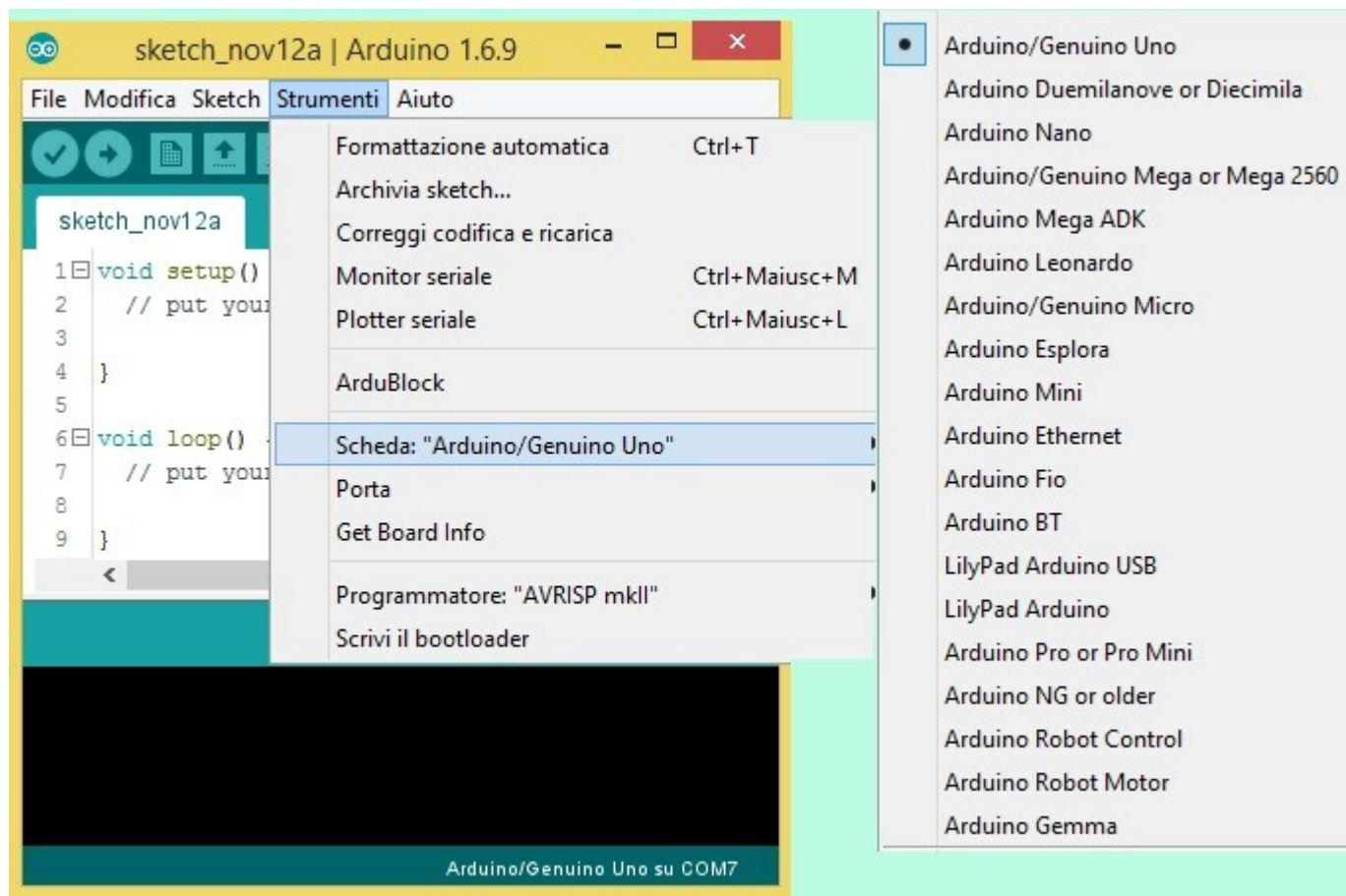
L'ambiente di sviluppo IDE

- **L'interfaccia dell'IDE Arduino si presenta così:**



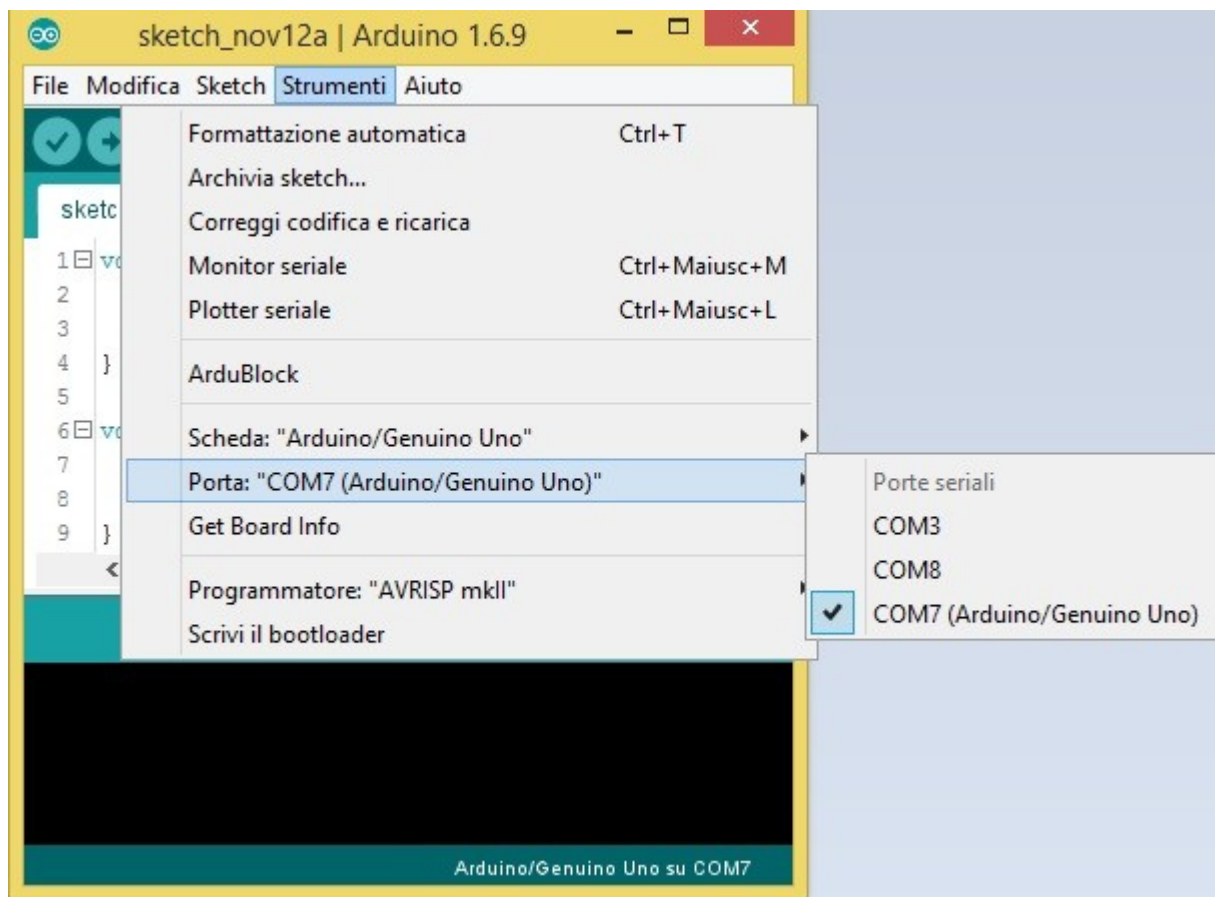
L'ambiente di sviluppo IDE

- **Selezionare il tipo di board Arduino:**



L'ambiente di sviluppo IDE

- **Selezionare la porta del Pc assegnata ad Arduino:**



Basi di elettronica per Arduino



Segnali d'ingresso e d'uscita

- **Segnali d'ingresso:**

- Per acquisire informazioni dall'ambiente esterno.
- Arduino riceve segnali in ingresso da sensori, pulsanti e altri dispositivi

- **Segnali d'uscita:**

- Per controllare o modificare l'ambiente esterno.
- Arduino può inviare segnali in uscita ad attuatori, relè, led, motori

Segnali Analogici e Digitali

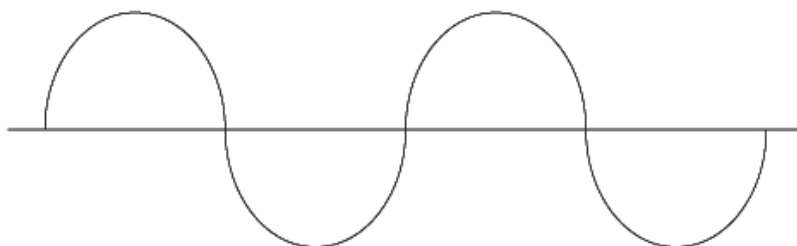
- **Segnali Analogici:**

- quando i valori che può assumere sono continui (infiniti valori intermedi)

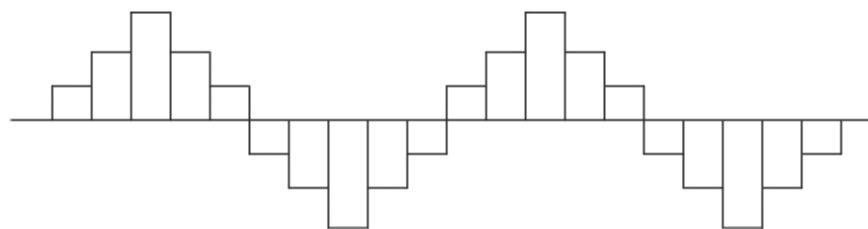
- **Segnali digitali:**

- quando i valori che può assumere sono discreti (finiti valori intermedi)

Analog



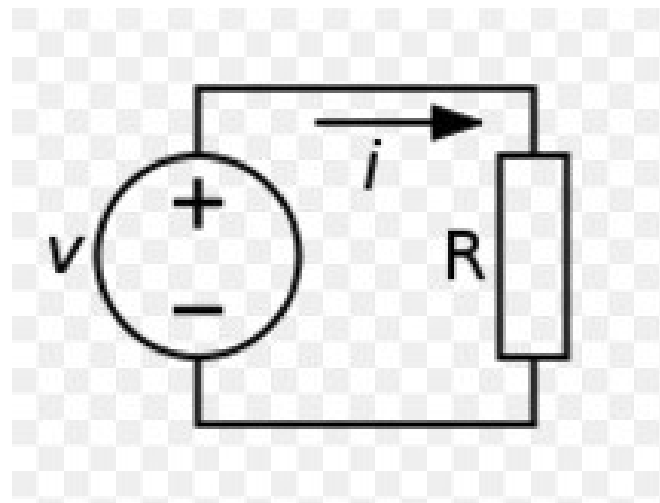
Digital



Maglia elettrica

- **Maglia elettrica:**

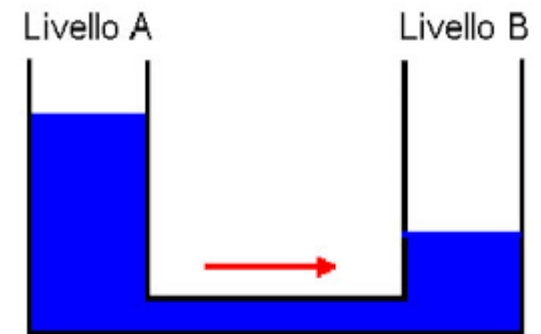
- All'interno di un circuito elettrico, indica un percorso chiuso costituito da più rami concatenati e utilizzati ciascuno una volta.



Tensione elettrica

Tensione elettrica:

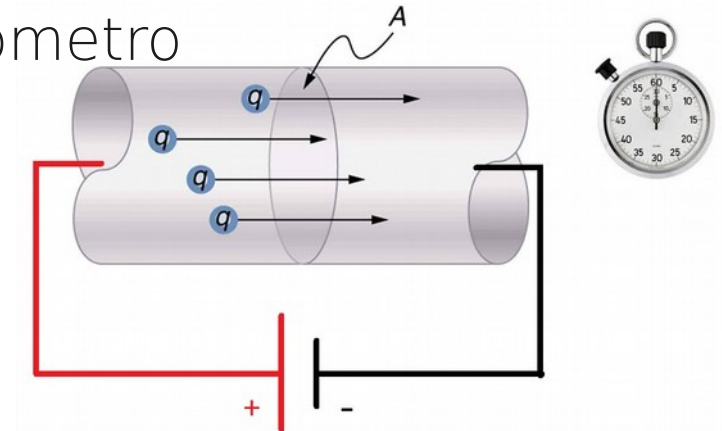
- Indica la **differenza di potenziale elettrico** (d.d.p.) tra due punti di un circuito elettrico.
- La carica elettrica (ad esempio quella dell'elettrone) si sposta dal punto a potenziale maggiore verso quello a potenziale minore.
- Più tensione significa quindi una maggiore capacità di spostare la carica elettrica (elettroni) tra i due punti di un circuito.
- La tensione si misura in **Volt [V]**
- Il suo strumento di misura è il Voltmetro



Corrente elettrica

Corrente elettrica:

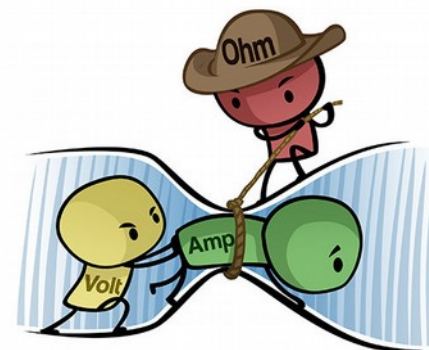
- La corrente è **la quantità di elettroni** (Coulomb [C]) che passano in una sezione di conduttore **nell'unità di tempo** (t).
- Collegando un filo conduttore tra due punti di un circuito a diverso potenziale elettrico (tensione), si otterrà in esso una circolazione di corrente.
- La corrente si misura in **Ampère [A]**
- Il suo strumento di misura è l'Amperometro



Resistenza elettrica

Resistenza elettrica:

- La resistenza elettrica è una grandezza fisica che misura la tendenza di un corpo ad opporsi al passaggio di una corrente elettrica (elettroni), quando sottoposto ad una tensione elettrica.
- Questa opposizione dipende dal materiale con cui è realizzato, dalle sue dimensioni e dalla sua temperatura.
- Uno degli effetti del passaggio di corrente in un conduttore è il suo riscaldamento (effetto Joule).
- La resistenza si misura in **Ohm** [Ω]
- Il suo strumento di misura è l'Ohmmetro



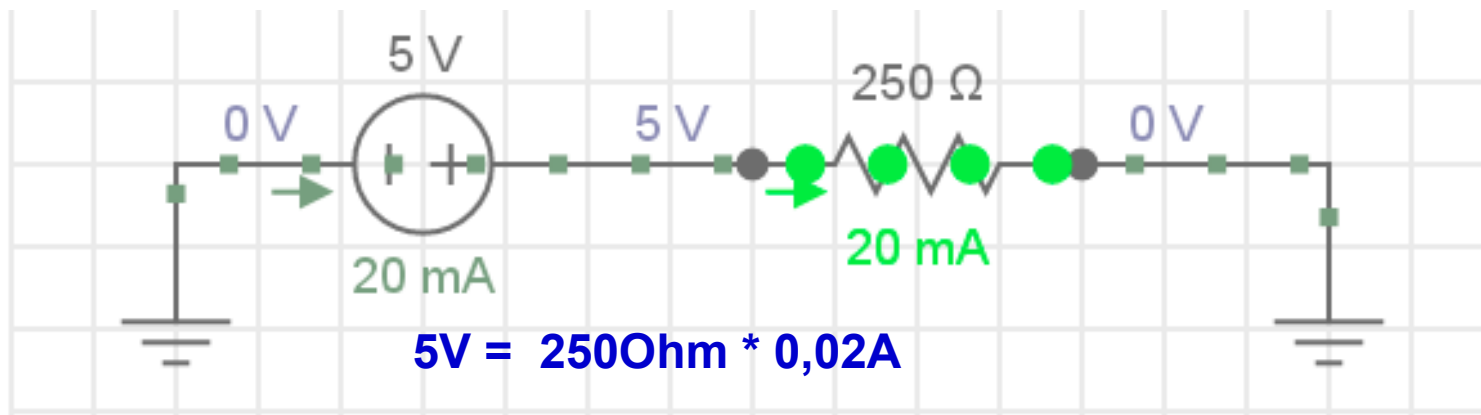
Prima legge di Ohm

- **Enunciato**

- La corrente che circola in un conduttore è direttamente proporzionale alla tensione applicata ed inversamente proporzionale alla resistenza

- **Formule**

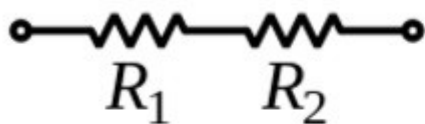
$$V = IR \quad \text{or} \quad I = \frac{V}{R} \quad \text{or} \quad R = \frac{V}{I}$$



Resistenze: serie e parallelo

Collegamento serie:

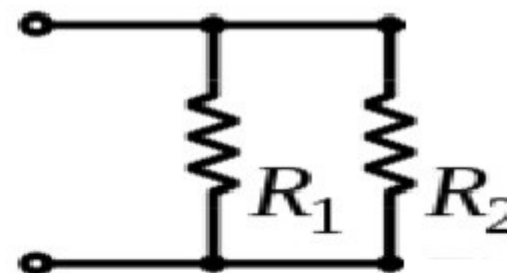
- La resistenza equivalente è la somma delle singole resistenze



$$R_{\text{equivalente}} = R_1 + R_2$$




























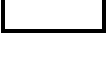
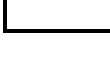
Collegamento parallelo:

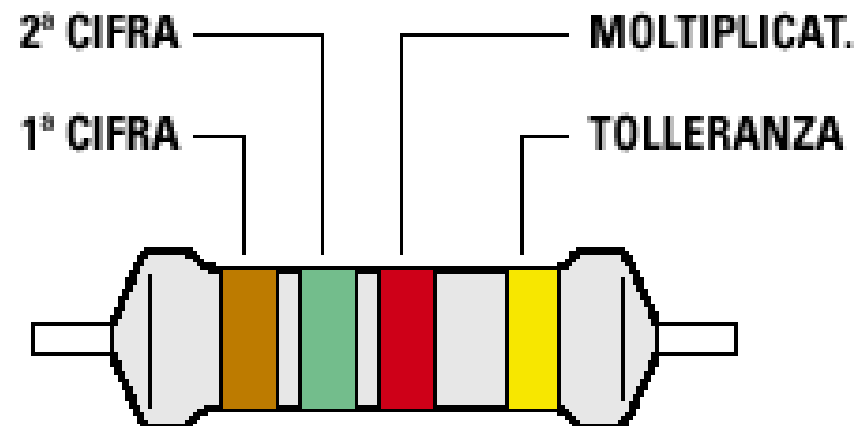
- La resistenza equivalente è la somma dei reciproci delle singole resistenze



$$R_{\text{equivalente}} = \frac{1}{R_1} + \frac{1}{R_2} = \frac{R_1 * R_2}{R_1 + R_2}$$

I colori delle resistenze: schema

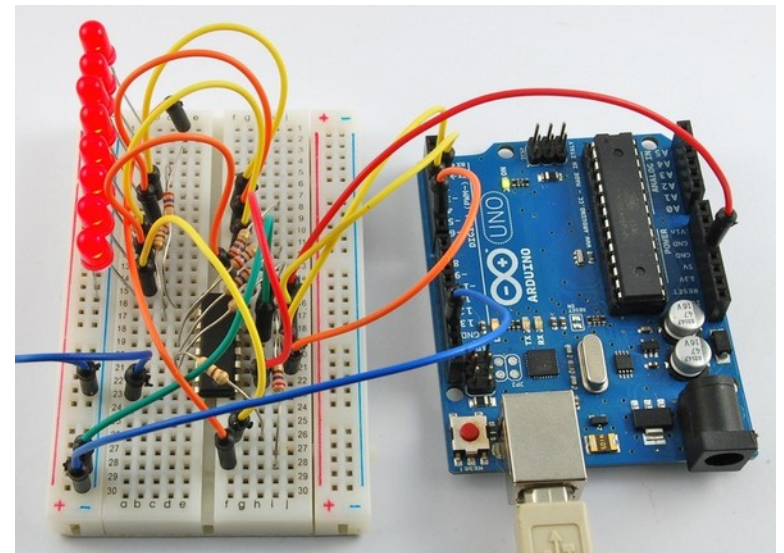
| | 1ª CIFRA | 2ª CIFRA | MOLTIPLICAT. | TOLLERANZA |
|-----------|---|---|---|--|
| NERO | ==== |  0 |  x 1 | 10 %  ARGENTO |
| MARRONE |  1 |  1 |  x 10 | 5 %  ORO |
| ROSSO |  2 |  2 |  x 100 | |
| ARANCIONE |  3 |  3 |  x 1.000 | |
| GIALLO |  4 |  4 |  x 10.000 | |
| VERDE |  5 |  5 |  x 100.000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



Breadboard

- **Basetta sperimentale:**

- E' uno strumento molto comodo e potente per realizzare montaggi di circuiti elettronici senza saldature
- I fori sono distanziati di 2,54 mm (1/10 di pollice) misura tipica della distanza dei pin dei circuiti integrati.
- Indispensabili e comodissimi i connettori Dupont m/m m/f f/f



Breadboard

- **Righe laterali:**

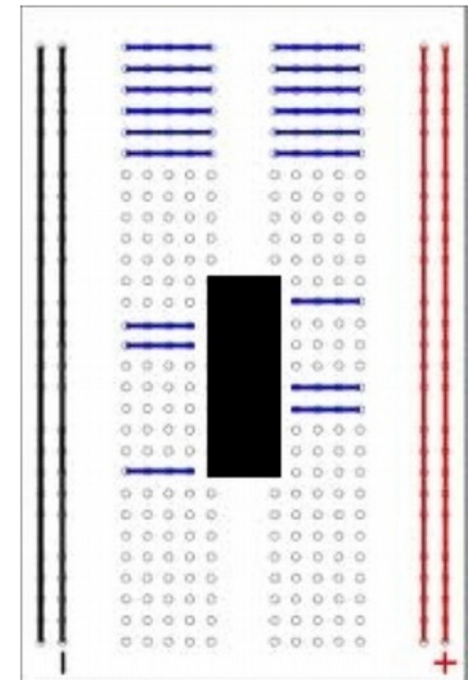
- I fori delle righe laterali sono tutti connessi tra di loro. Normalmente vengono utilizzate per +/- dell'alimentazione. E' distribuita su tutta la basetta

- **Colonne centrali:**

- I fori delle colonne centrali sono tutti connessi tra di loro. Equivalgono ai nodi del circuito elettronico

- **Simmetria:**

- La scanalatura centrale consente di inserire i circuiti integrati a cavallo di essa, in questo modo rimangono a disposizione di ogni pin altri fori per collegare ulteriori componenti



Basi di programmazione per Arduino



Come ragiona un microcontrollore

- **Algoritmo:**

- Un algoritmo è un procedimento logico pensato dall'uomo per risolvere un determinato problema attraverso un numero finito di passi elementari.
- Interpretazione e risultato univoci, passi, tempo, e input finiti

- **Programma:**

- E' la traduzione o la codifica dell'algoritmo, scritta in un certo linguaggio adatto per essere eseguito da un microcontrollore.

- **Esecuzione:**

- Il microcontrollore legge i passi codificati del programma uno per volta e li esegue secondo la logica stessa del programma

Linguaggio di programmazione di Arduino

- **Wiring... un C semplificato:**

- I programmi Wiring sono chiamati **sketch** e sono scritti in **C/C++** e necessitano soltanto di due funzioni per poter essere eseguiti:
- **setup()** – funzione che viene eseguita una sola volta, all'avvio del programma, che può essere utilizzata per definire delle impostazioni del programma che non verranno più cambiate nel corso della sua esecuzione;
- **loop()** – funzione che viene richiamata continuamente, fino allo spegnimento del dispositivo. Al suo interno troviamo il centro operativo del programma. E' da qui che partono le chiamate alle altre funzioni.

Strutture di un linguaggio

- **Istruzioni:**

- Un comando che cambia lo stato interno del microcontrollore

`a = b + 100;`

- **Variabile:**

- Un dato suscettibile di modifica a cui il programma può fare riferimento per nome.

- **Costante:**

`int pressione = 975;`

- Un dato non modificabile a cui il programma può fare riferimento per nome.

- **Operatore:**

`const int anno_di_nascita = 1982;`

- un simbolo che specifica quale legge applicare a uno o più operandi, per generare un risultato.

Aritmetici, Assegnamento, Booleani, Confronto

`A < 10;` `tempo == 0;` `pulsante != 0;`

Strutture di un linguaggio

- **Espressioni:** `int posizione = (offset + 975) / 2;`
 - Una combinazione di variabili, costanti e operatori che verrà valutata per produrre un valore utile a risolvere il “problema”
- **Strutture di controllo:**
 - Permettono di gestire il flusso di esecuzione del programma in base alla valutazione di espressioni.
 - **Iterativi** come `for`, `while`, `do`
 - **Condizionali** come `if`, `switch-case`
- **Funzioni:**
 - Blocchi di codice specializzato, riutilizzabile e richiamabile secondo le necessità. Una libreria è una collezione di funzioni.
- **Commenti:**
 - Testi non interpretati dal compilatore: note o spiegazioni utili alla comprensione del programma
 - `// una riga` ; `/* più righe */`

Struttura del linguaggio

- **Sintassi di base:**

- Linguaggio è CASE SENSITIVE, maiuscole/minuscole fanno differenza
- **(;)** serve per terminare un' istruzione
- (**{**) Un gruppo di istruzioni va racchiuso tra graffe (**}**)
- return,

- **Parole chiave:**

- Sono parole predefinite con valori speciali
- **HIGH** / **LOW** si usano, quando si vuole accendere o spegnere un pin di Arduino.
- **INPUT** / **OUTPUT** si usano per impostare un determinato pin come ingresso o uscita.
- **true** / **false** indica il fatto che una condizione o un'espressione è vera o falsa.

Struttura del linguaggio

- **Tipi di dato:**

| <i>Tipi di dichiarazione</i> | <i>Rappresentazione</i> | <i>N. di byte</i> | <i>Intervallo</i> |
|------------------------------|-------------------------|-------------------|--|
| Boolean | | | True – False / On –Off / High - Low |
| Char | Carattere | 1 (8 bit) | - 127 +127 |
| Byte | Carattere | 1 (8 bit) | 0 +255 |
| Int | Numero intero | 2 (16 bit) | -32.768 + 32.767 |
| Unsigned int | Numero intero | 2 (16 bit) | 0 + 65.535 |
| Short | Numero intero "corto" | 2 (16 bit) | |
| Long | Numero intero "lungo" | 4 (32 bit) | -2.147.483.648 +2.147.483.647 |
| Float | Numero reale | 4 (32 bit) | -3.4028235E+38 a + 3.4028235E+38 |
| Double | Numero reale "lungo" | 8 (64 bit) | 1.7976931348623157 x 10 ³⁰⁸ |

Gestire la libreria Serial

E' una libreria integrata di Arduino:

- Contiene diverse funzioni richiamabili nel nostro programma relative alla comunicazione seriale (nativa su Arduino).
- Può essere utilizzata per comunicare dati o individuare i bug dei nostri programmi
- Dall' IDE è possibile aprire un monitor sulla comunicazione seriale per ispezionarne il flusso dati.



```
sketch_nov15c $  
1 void setup() {  
2   Serial.begin(9600); // inizializza la comunicazione seriale a 9600 baud rate  
3 }  
4  
5 void loop() {  
6   Serial.println("Hello , world"); //manda in porta seriale "Hello, world"  
7   delay(1000); // attende per un secondo  
8 }
```

Monitor
seriale

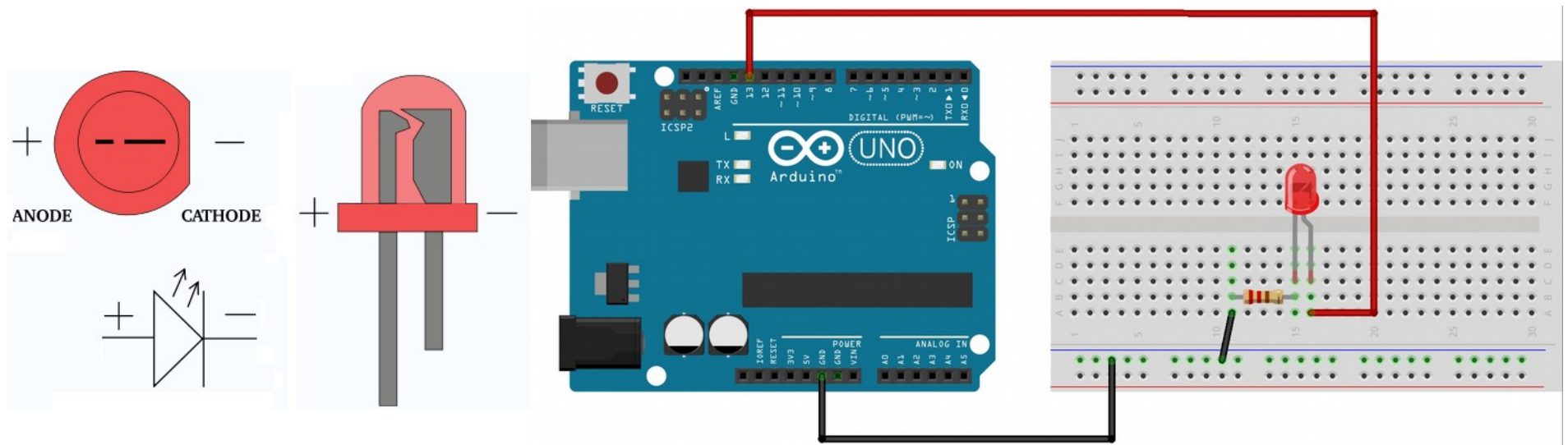
Blink del Led

- **pinMode(pin, mode)**

- Utilizzato in void setup (), serve per configurare un determinato pin e stabilire se deve essere un INPUT o OUTPUT.

- **digitalWrite(pin, valore)**

- Attiva o disattiva un pin digitale, ponendolo a livello logico HIGH o LOW.



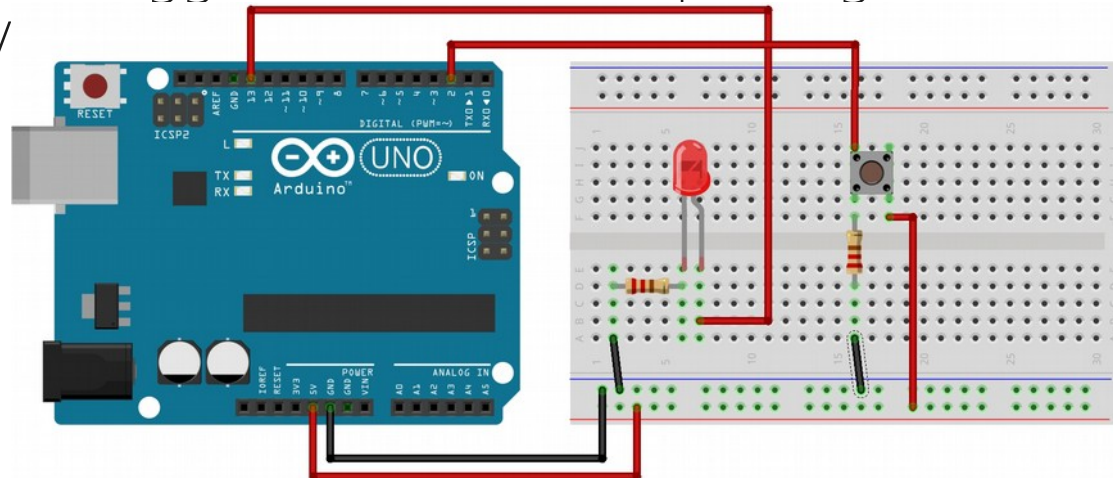
fritzing

Blink del Led

```
blink_del_led
1 void setup() {
2   pinMode(13, OUTPUT);           //imposta il pin digitale 13 come uscita
3 }
4
5 void loop() {
6   digitalWrite(13,HIGH);         // fornisce 5V al pin 13 se c'è un led si accenderà
7   delay(1000);                   //attende per un secondo
8   digitalWrite(13,LOW);         //mette a 0V il pin 13 se il Led è acceso si spegnerà
9   delay(1000);                   //attende per un secondo
10 }
```

Utilizzare un pulsante

- **If (condizione)**
{istruzioni se la condizione è vera;}
else {istruzioni se la condizione è falsa;}
 - gestisce il flusso di esecuzione del codice tramite condizioni
- **digitalRead(pin);**
 - Serve per leggere lo stato del pin digitale: HIGH 5v || LOW 0v



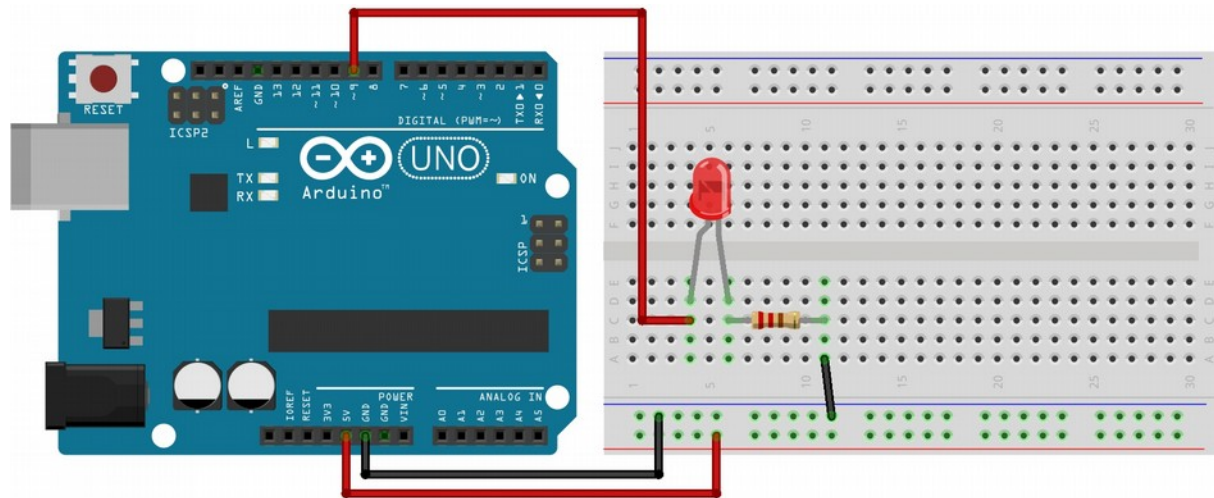
Utilizzare un pulsante

Led_con_pulsante\$

```
1  const int buttonPin = 2;           //il numero del pin collegato al pulsante
2  const int ledPin = 13;             //il numero del pin collegato al Led
3  int buttonState = 0;               //variabile che assume lo stato del Led
4
5  void setup() {
6    pinMode(ledPin, OUTPUT);
7    pinMode(buttonPin, INPUT);
8  }
9
10 void loop() {
11   buttonState = digitalRead(buttonPin); //legge lo stato del pulsante e lo assegna alla variabile
12
13   if (buttonState == HIGH) {         //valutazione della condizione nell'if
14     digitalWrite(ledPin, HIGH);      //se la condizione è verificata accende il led (5V sul pin)
15   }
16   else {
17     digitalWrite(ledPin, LOW);       //se la condizione non è verificata spegne il led (0V sul pin)
18   }
19 }
```

Fade del Led

- **for (inizializzazione; controllo; incremento)**
{istruzioni; }
 - Serve a ripetere un blocco di codice n volte
- **analogWrite(pin, valore);**
 - Cambia la percentuale della modulazione PWM ricostruendo un segnale peseudo-analogico 0/5 v con 256 livelli

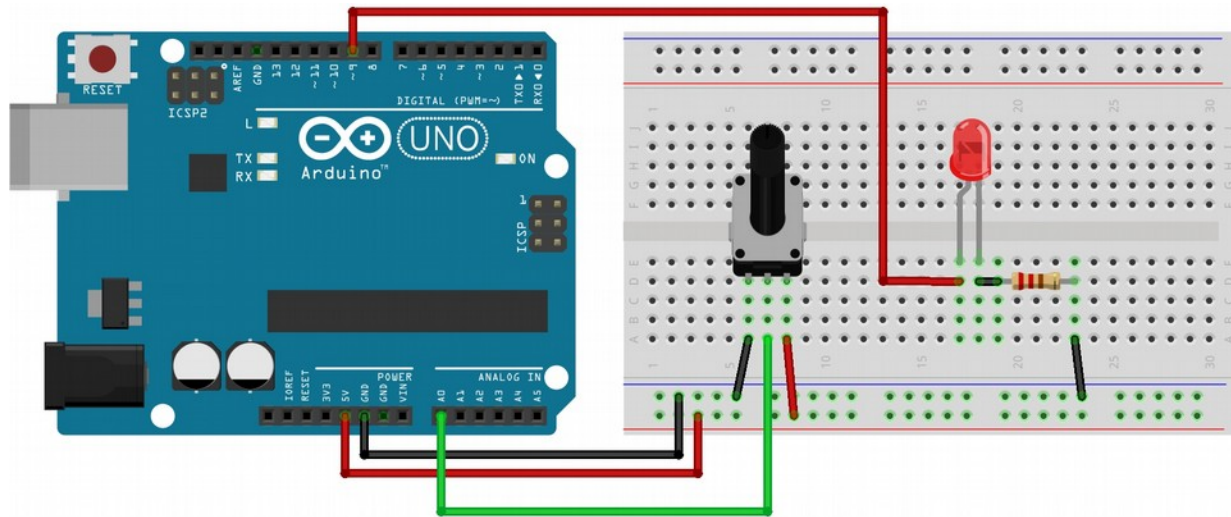


Fade del Led

```
1 int led = 9;
2
3 void setup() {
4     pinMode(led, OUTPUT);
5 }
6
7 void loop() {
8
9     for(int i=0;i<255;i++){
10         analogWrite(led,i);
11         delay(10);
12     }
13
14 }
```

Sensore analogico: potenziometro

- **analogRead(pin);**
 - Legge il valore di tensione applicato ad un pin di input analogico con una risoluzione pari a 10 bit, restituendo un numero intero compreso tra 0 e 1023.
- **map(value, fromLow, fromHigh, toLow, toHigh);**
 - Mappa valori con un range d'ingresso in valori con un range d'uscita



Sensore analogico: potenziometro

```
1 int sensorPin = 0;           //il pin A0 collegato al potenziometro
2 int ledPin = 9;              //il pin 9 collegato al Led
3 int sensorValue = 0;         //variabile per il valore analogico letto
4
5 void setup() {
6   pinMode(ledPin, OUTPUT);    //imposta il pin digitale 9 come uscita
7 }
8
9 void loop() {
10  sensorValue = analogRead(sensorPin); //acquisisce da A0 assegnando un valore 0-1023
11  sensorValue = map(sensorValue, 0, 1023, 0, 255); //mappa i valori da 0-1023 in 0-255
12  analogWrite(ledPin, sensorValue);    //ricostruzione uscita analogica sul pin 9
13 }
```

Progetto

**Costruiamo un Gadget Natalizio
libreria NeoPixel + Smart Led**



Smart Led

Led concatenabili modello WS2812:

- Led innovativi che possiedono un circuito di controllo integrato
- Vengono collegati in serie uno dietro l'altro più o meno spazati
- Il pilotaggio avviene tramite un solo pin indipendentemente dal numero di led: si riduce la complessità del cablaggio.
- Si trovano pre-assemblati in numerose forme: strip, anelli, matrici



Libreria NeoPixel

- **Creata da Adafruit per Arduino:**

- E' pensata per utilizzare i Led concatenabili del tipo: WS2812
- Contiene diverse funzioni per gestire facilmente il pilotaggio
- Permette di dedicarsi fin da subito all'animazione anche complessa
- Esiste anche **NeoMatrix** libreria dedicata alle matrici

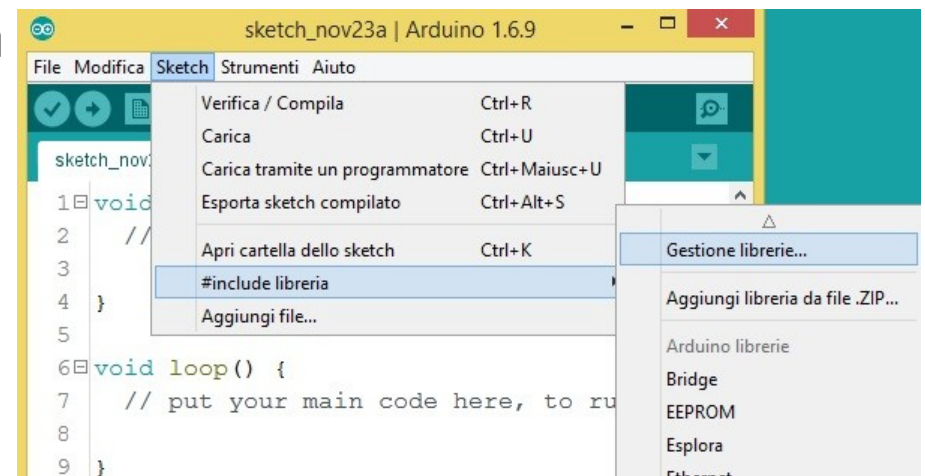
- **Installazione della libreria**

- Dall'IDE Arduino seguire:

Sketch

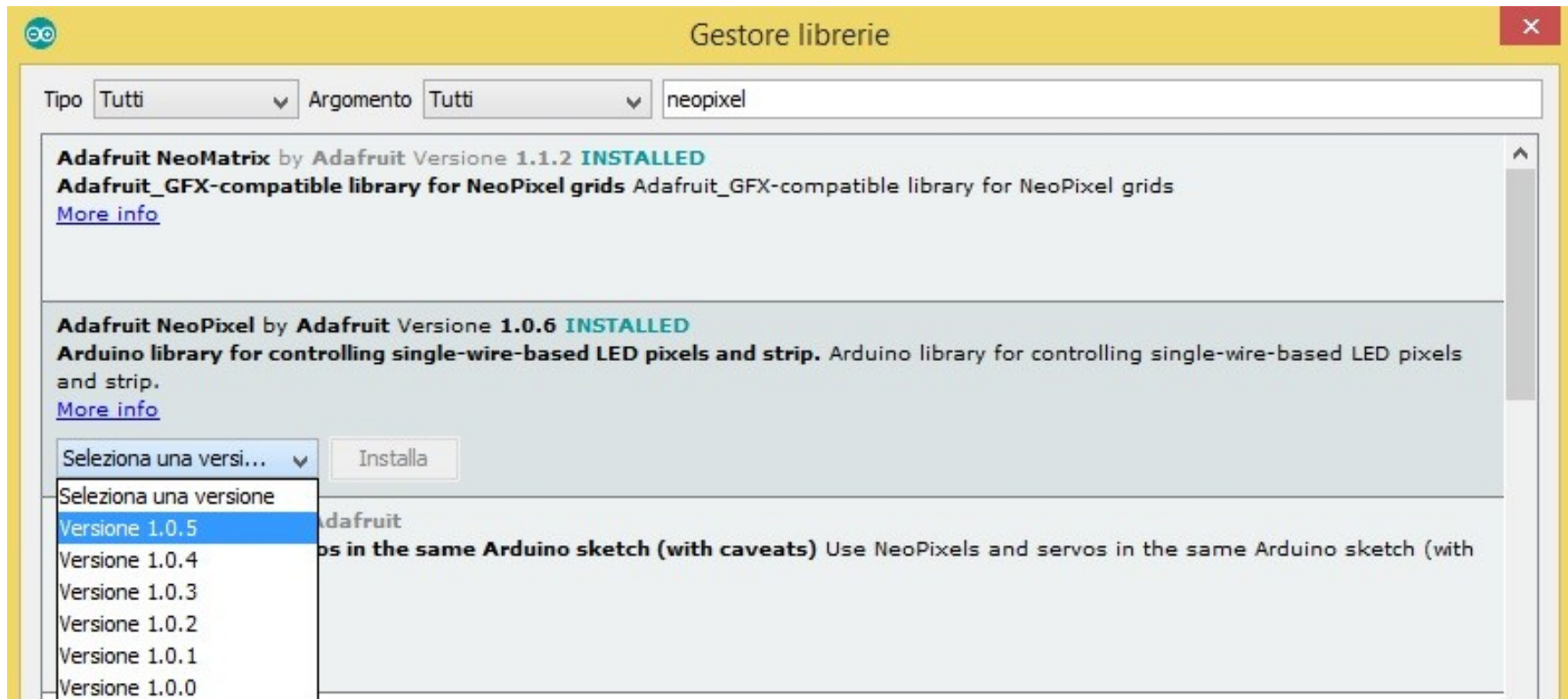
#include libreria

Gestione librerie



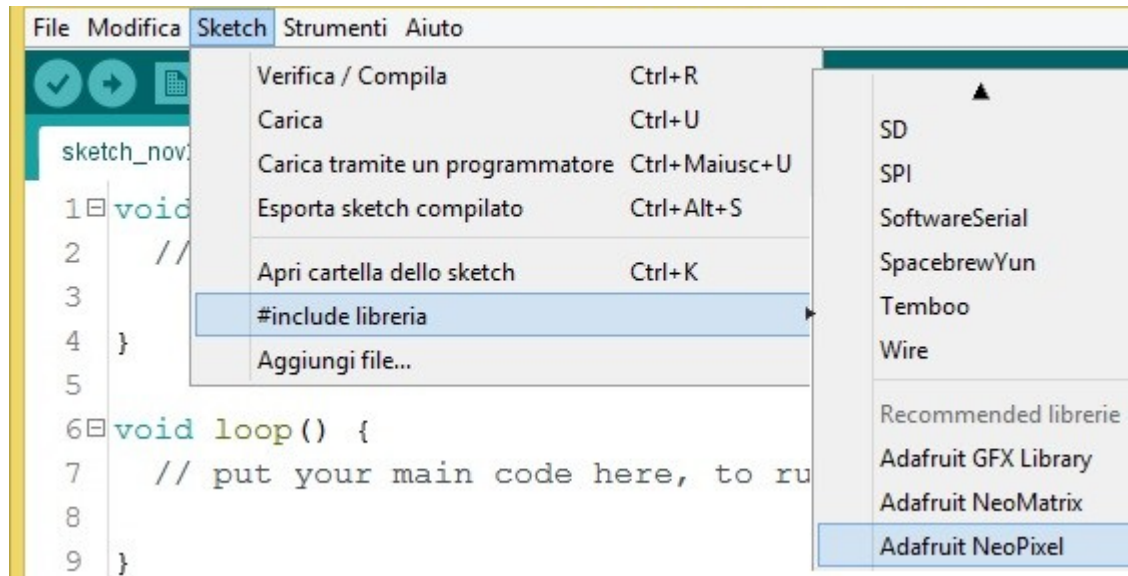
Libreria NeoPixel

- Scegliere la versione e avviare l'installazione
- Non bisogna fare altro il procedimento è automatico, semplicemente riavviare l'IDE



Libreria NeoPixel

Inclusione della libreria nello sketch



Effetto nello sketch

```
1 #include <Adafruit_NeoPixel.h>
2
3 void setup() {
4   // put your setup code here, to run once:
5
6 }
```

NeoPixel: funzioni principali

- **Adafruit_NeoPixel NOME = Adafruit_NeoPixel (N_Led , PIN , NEO_GRB + NEO_KHZ800);**
 - Definisce un "Oggetto" Adafruit_NeoPixel personalizzato
 - NOME specifica il nome scelto per l'oggetto
 - N_Led specifica il numero di Led concatenati
 - PIN specifica con quale Pin di Arduino gestirà il pilotaggio
 - NEO_GRB o NEO_RGB specifica la mappatura del Led WS2812
 - NEO_KHZ400 o NEO_KHZ800 specifica la frequenza del Led WS2812
- **NOME.begin();**
 - Inizializza la comunicazione dati sul Pin scelto
 - Va richiamata all'interno di setup()

NeoPixel: funzioni principali

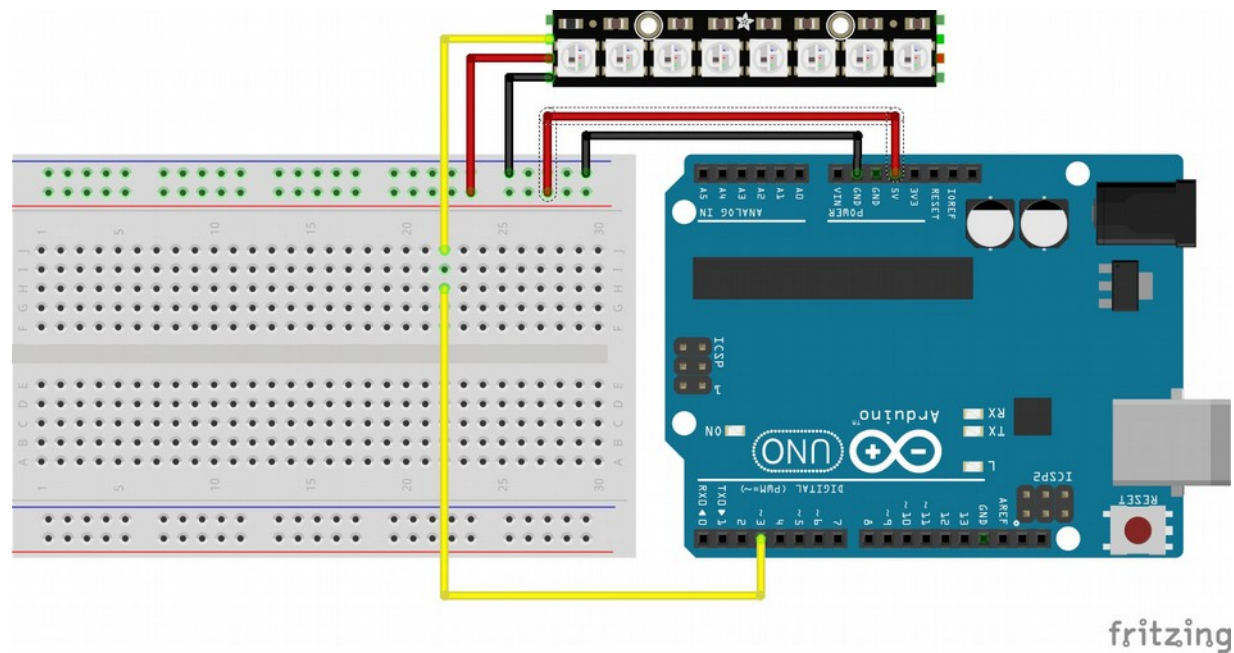
- **NOME.setPixelColor(N, Red, Green, Blue);**
 - E' utilizzata per settare il colore di un singolo Led
 - N indica la posizione dell'ennesimo Led concatenato (il primo è 0)
 - Red, Green, Blue possono variare tra 0-255 ed esprimono l'intensità luminosa per ogni canale del Led
- **uint32_t nomecolore = NOME.Color(Red, Green, Blue);**
 - Permette di memorizzare in una variabile a 32 bit la composizione cromatica dei tre canali R G B in maniera compatta e riconoscibile
 - E' possibile passare direttamente la variabile all'argomento della funzione: **NOME.setPixelColor(N,nomecolore)**

NeoPixel: funzioni principali

- **NOME.setBrightness(Val);**
 - E' utilizzata per settare la luminosità di tutti i Led ad un valore unico
 - Val indica il valore di luminosità che si vuole ottenere da 0 a 255.
- **NOME.show();**
 - Manda ai Led le modifiche impostate con le altre funzioni al fine di renderle visibili.

NeoPixel:

Circuito di collegamento con una Strip da 8 Smart Led



Accendere tutti i Pixel

Usiamo un ciclo for per accendere tutti i Pixel in sequenza

```
#include <Adafruit_NeoPixel.h>

#define PIN 6
#define PIXELS 44

Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXELS, PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  strip.begin();
}

void loop() {
  for(int i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0, 0, 255);
    strip.show();
    delay(500);
  }
}
```

Generare colori casuali

Usiamo la funzione random per creare un colore casuale

```
#include <Adafruit_NeoPixel.h>

#define PIN 6
#define PIXELS 44

Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXELS, PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  strip.begin();
}

void loop() {
  int r = random(0,255);
  int g = random(0,255);
  int b = random(0,255);
  uint32_t color = strip.Color(r, g, b);

  for(int i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, color);
    strip.show();
    delay(500);
  }
}
```


Creiamo una funzione per il RandomColor

Incapsuliamo la creazione di un colore casuale in una funzione

```
uint32_t randomColor(){  
    int r = random(0,255);  
    int g = random(0,255);  
    int b = random(0,255);  
    uint32_t c = strip.Color(r, g, b);  
    return c;  
}
```

per utilizzarlo possiamo fare

```
uint32_t c = randomColor();  
strip.setPixelColor(i, c);
```

**Grazie e
arrivederci**

