Barra LED (Knight Rider) [Avanzato]

Obiettivo: Realizzazione della barra LED utilizzata nella serie TV Knight Rider.

Prerequisiti:

Barra LED (Knight Rider)

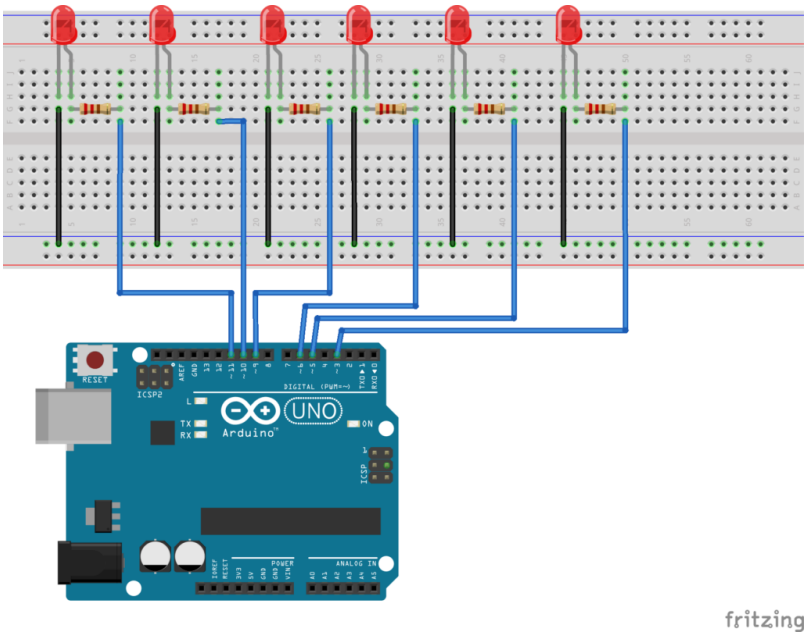
Fading led

Componenti elettronici:

- Arduino UNO
- Breadboard
- 6 Led
- 6 Resistenze (100 Ohm)

Teoria: Al fine di realizzare una barra LED (Light Emitting Diode), 6 diodi ad Emettitore di Luce sono stati utilizzati e collegati a differenti PIN digitali di Arduino. Come nelle lezioni precedenti ad ogni LED è associata una resistenza al fine di limitare il passaggio di corrente.
A differenza della barra led riportata nei prerequisiti, dove un solo led era acceso, in questa attività più led sono accesi in contemporanea. Nello specifico attraverso l'istruzione analogWrite l'intensità luminosa viene modulata in quattro differenti livelli.

Collegamento Circuitale:



Schema Circuitale

Codice:

Personalizzazioni: E’ possibile modificare il comportamento del circuito in questione intervenendo sul valore della variabile *ledTime*. Modificando il suo valore infatti cambia la velocità di lampeggiamento della barra LED.

Barra LED (Knight Rider)

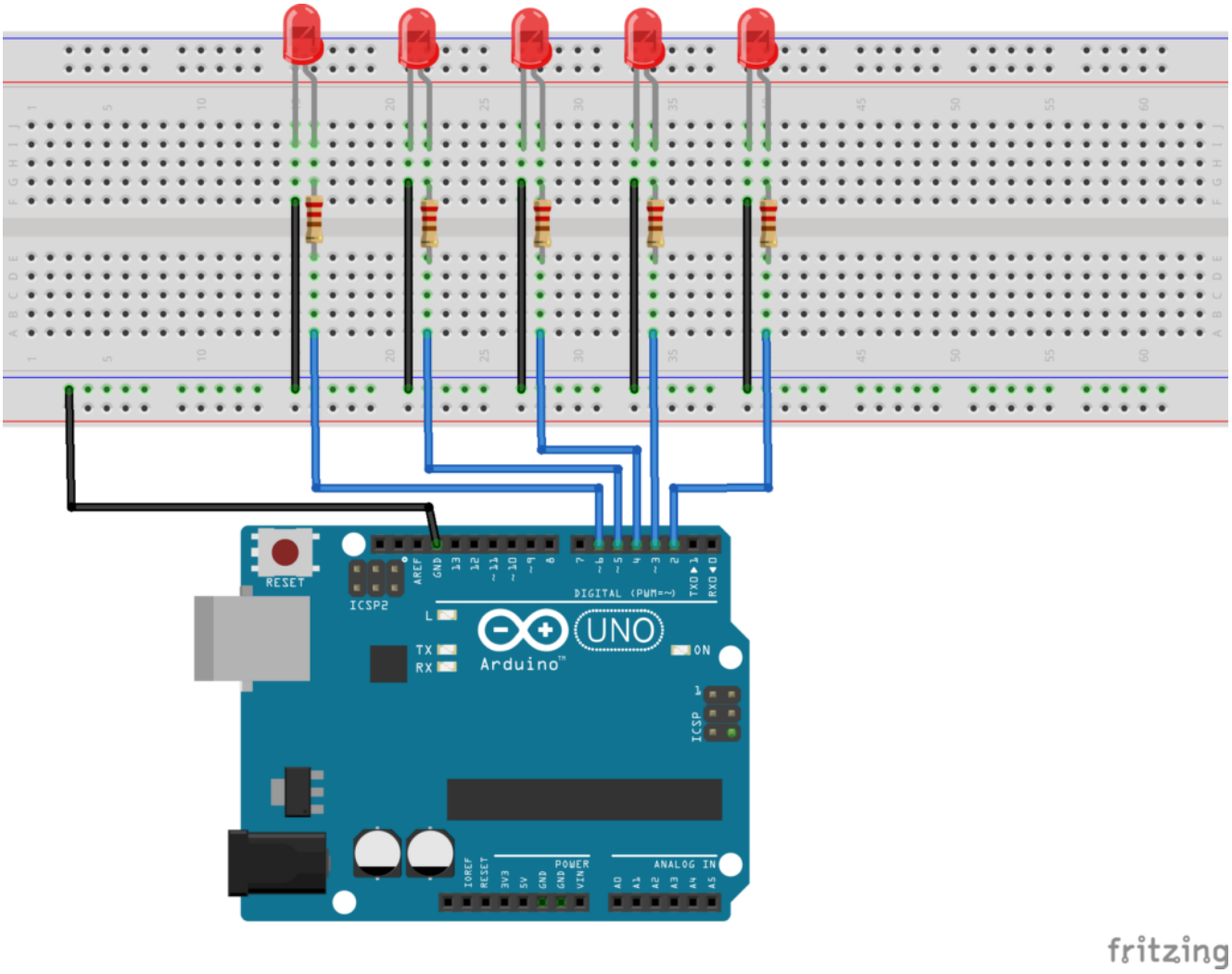
Obiettivo: Realizzazione di una barra LED.

Componenti elettronici:

- Arduino UNO
- Breadboard
- 5 Led
- 5 Resistenze (100 Ohm)

Teoria: Al fine di realizzare una barra LED (Light Emitting Diode), 5 diodi ad Emettitore di Luce sono stati utilizzati e collegati a differenti PIN digitali di Arduino. Come nelle lezioni precedenti ad ogni LED è associata una resistenza al fine di limitare il passaggio di corrente.

Collegamento Circuitale:



Collegamento Circuitale

Codice:

[crayon-6632145a2a8f3552921295/]

Tinkercad:

Personalizzazioni: E’ possibile modificare il comportamento del circuito in questione intervenendo sul valore della variabile *ledTime*. Modificando il suo valore infatti cambia la velocità di lampeggiamento della barra LED.
E’ inoltre possibile aggiungere ulteriori LED cambiando rispettivamente l’hardware ed il software presentato.

Il Semaforo

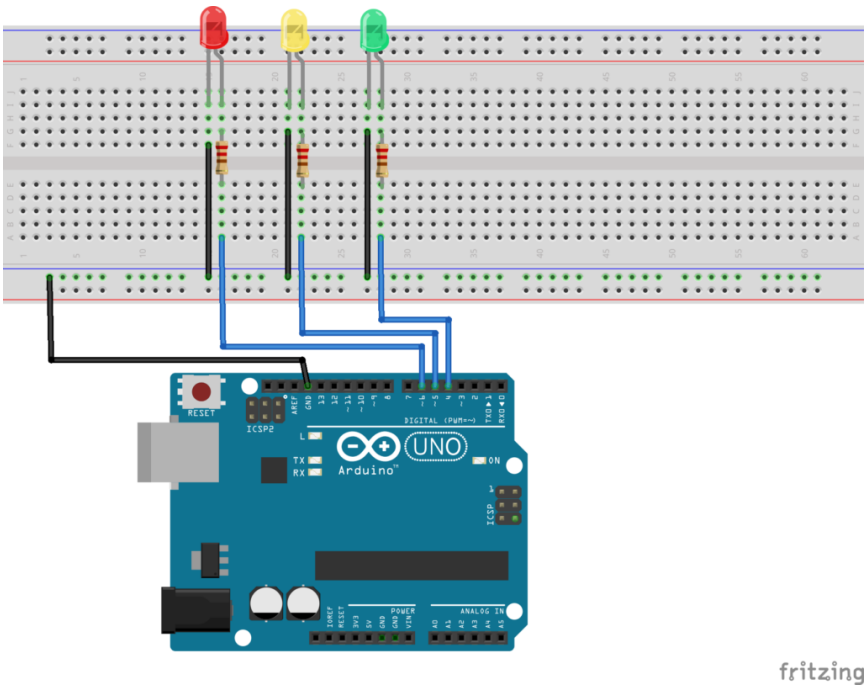
Obiettivo: Realizzazione di un semplice semaforo utilizzando Arduino.

Componenti elettronici:

- Arduino UNO
- Breadboard
- 3 Led (verde giallo e rosso)
- 3 Resistenze (100 Ohm)

Teoria: Al fine di realizzare una semaforo a LED (Light Emitting Diode), 3 diodi ad Emettitore di Luce (1 verde, 1 giallo ed 1 rosso) sono stati utilizzati e collegati a differenti PIN digitali di Arduino. Come nelle lezioni precedenti ad ogni LED è associata una resistenza al fine di limitare il passaggio di corrente.

Collegamento Circuitale:



Collegamento Circuitale

Codice:

Personalizzazioni: E’ possibile modificare il comportamento del circuito in questione intervenendo sul valore delle variabili *greenTime,.yellowTime, redTime.* Modificando i valori infatti cambia la velocità di funzionamento del semaforo.
E’ inoltre possibile aggiungere ulteriori LED per simulare un semaforo doppio o un incrocio.

Blinking led [Avanzato]

Obiettivo: Realizzazione, mediante breadboard, di un led che

lampeggi ad una frequenza specifica (e.g., 1Hz) su un PIN differente dal 13.

Componenti elettronici:

- Arduino UNO
- Breadboard
- Led
- Resistenza (100 Ohm)

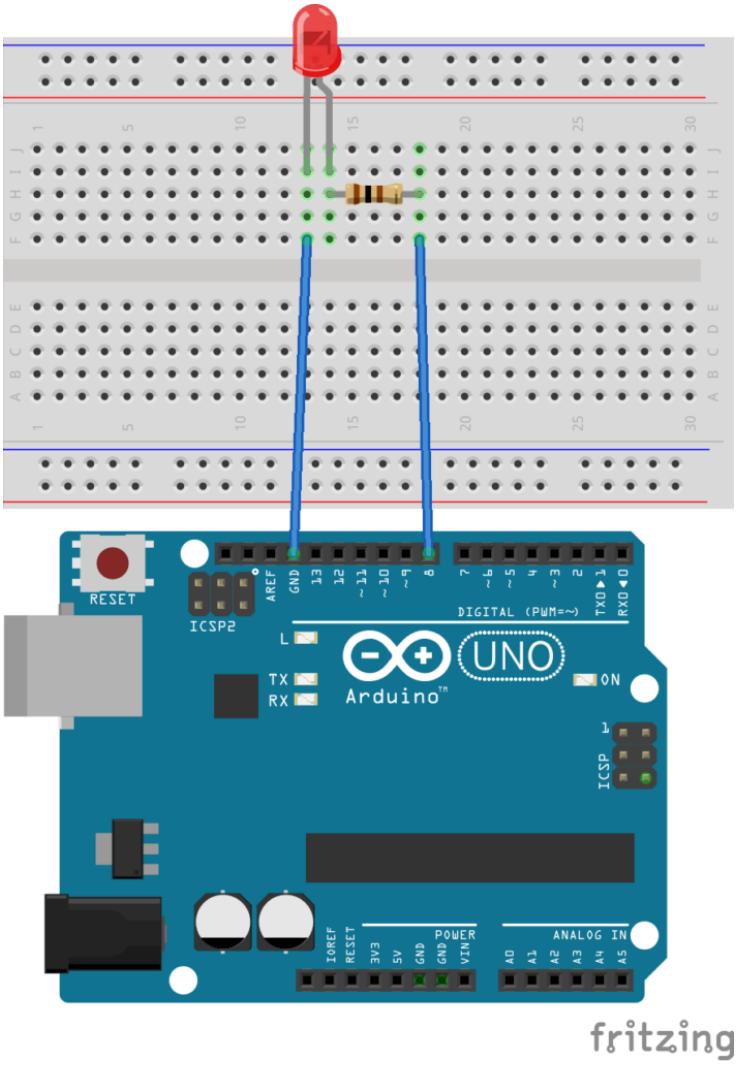
Teoria: Al fine di garantire il corretto funzionamento del LED su un pin differente dal 13, è indispensabile l'utilizzo di una resistenza in serie al dispositivo. La resistenza permette di regolare fissare i corretti valori di tensione e corrente necessari ad alimentare il LED.
Ad esempio, considerando una tensione sul pin di Arduino pari a 5V ed i seguenti parametri caratteristici del LED:

- ILed = 20 mA
- VLed = 1,5 V

Data la **legge di Ohm** V=RI, la resistenza necessaria per garantire un corretto funzionamento del diodo emettitore di luce può essere così calcolata:

R = (5-2)/20*10^(-3)

Collegamento Circuitale:



Collegamento Circuitale

Codice:

[crayon-6632145a2b310064441241/]

Tinkercad:

https://www.tinkercad.com/embed/ckyY3Pamusd

Personalizzazioni: E’ possibile modificare il comportamento del circuito in questione intervenendo sul valore della variabile *ledTime*. Modificando il suo valore infatti cambia la frequenza di lampeggiamento del LED.
E’ inoltre possibile modificare il pin digitale utilizzato per pilotare il LED cambiando rispettivamente hardware e software.

Approfondimento Teorico:

Legge di Ohm: La legge di Ohm mette in relazione le tre grandezze elettriche fondamentali Tensione (Volt), Intensità di Corrente (Ampere) e Resistenza (Ohm) secondo la seguente relazione.

$$V = RI$$

L’utilizzo di una resistenza, in serie al LED, serve appunto per limitare la quantità di corrente presente sul diodo emettitore di luce.

Blinking led

Obiettivo: Realizzazione di un led che lampeggi ad una frequenza specifica (e.g., 1Hz)

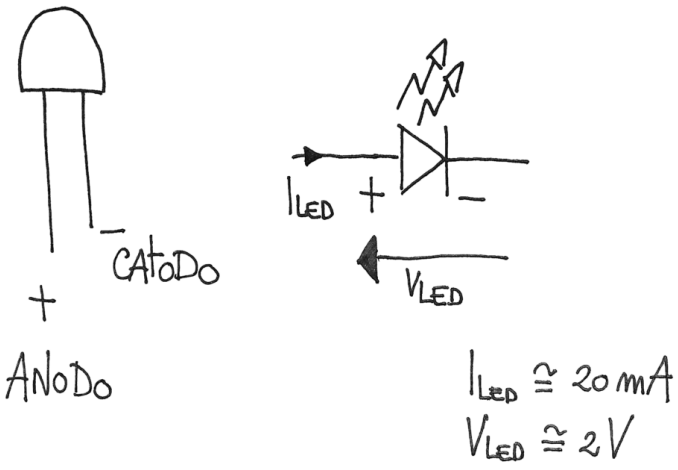
Componenti elettronici:

- Arduino UNO
- Led

Teoria: Il LED (Light Emitting Diode) o Diodo Emettitore di Luce è un dispositivo elettronico che sfrutta le proprietà di alcuni materiali semiconduttori di emettere fotoni (produrre luce).
Questo dispositivo è ampiamente utilizzato in molti applicativi realizzati con Arduino ed è caratterizzato da una propria tensione e corrente di funzionamento. Valori tipici sono:

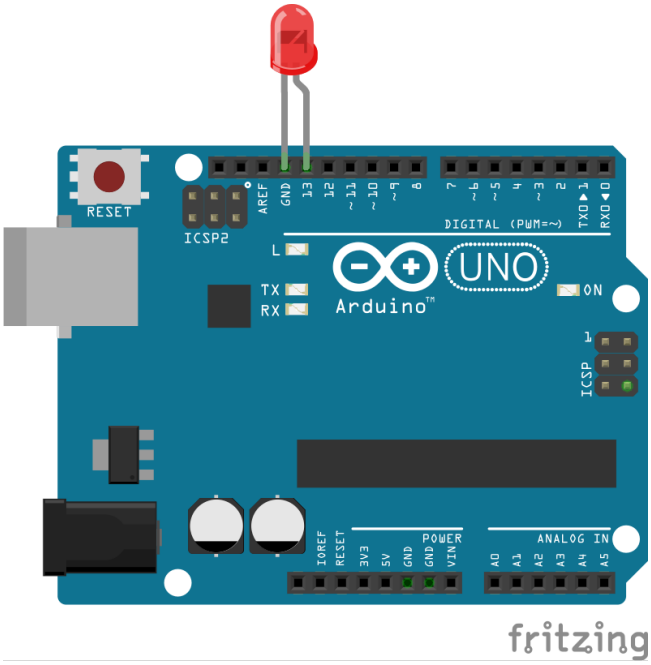
- ILed = 20 mA
- VLed = 2 V

Valori superiori o inferiori possono danneggiare il dispositivo. Per questo motivo si utilizza solitamente una resistenza in serie al LED al fine di limitarne corrente e tensione.
Per un adeguato funzionamento il LED deve anche essere correttamente polarizzato. Nel dettaglio il terminale più lungo di un led rappresenta l'anodo (+) mentre quello più corto il catodo (-).



Rappresentazione grafica di un LED e simbolo circuitale

Collegamento Circuitale:



Collegamento Circuitale

Codice:

L’istruzione **digitalWrite** permette di impostare lo stato logico di un PIN digitale al valore HIGH (5 Volt) o LOW (0 Volt).
Mentre l’istruzione **delay** permette di bloccare Arduino nello stato in considerazione per un certo numero di millisecondi.

Personalizzazioni: E’ possibile modificare il comportamento del circuito in questione intervenendo sul valore della variabile *ledTime*. Modificando il suo valore infatti cambia la frequenza di lampeggiamento del LED.

Approfondimento Teorico:

- Pin13: Nelle prime versioni delle schede Arduino, come riportato nella relativa documentazione: “*there is, however, about 1000 ohms of resistance on pin 13, so you can connect an LED without external resistor*” *il PIN13 presenta un’uscita limitata in correte a causa di una resistenza integrata. Grazie alla presenza di questo elemento non è necessario introdurre una resistenza esterna al fine di limitare la corrente sul LED.*

Caricare il primo programma (Blinking Led)

Obiettivo: Utilizzare il Software Arduino per caricare il primo programma e fare lampeggiare un LED

Prerequisiti

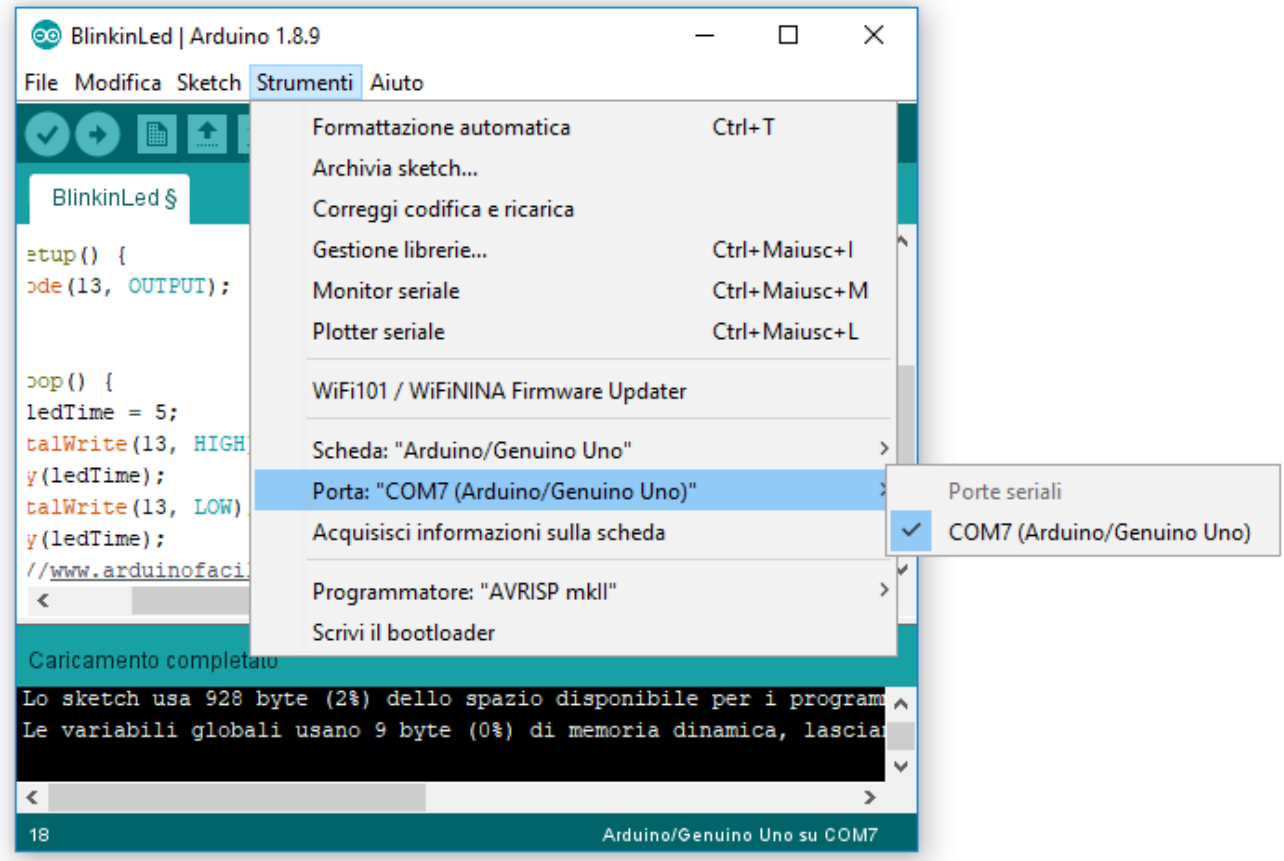
Blinking led [Avanzato]

Come installare l'ambiente Software di Arduino

Teoria:

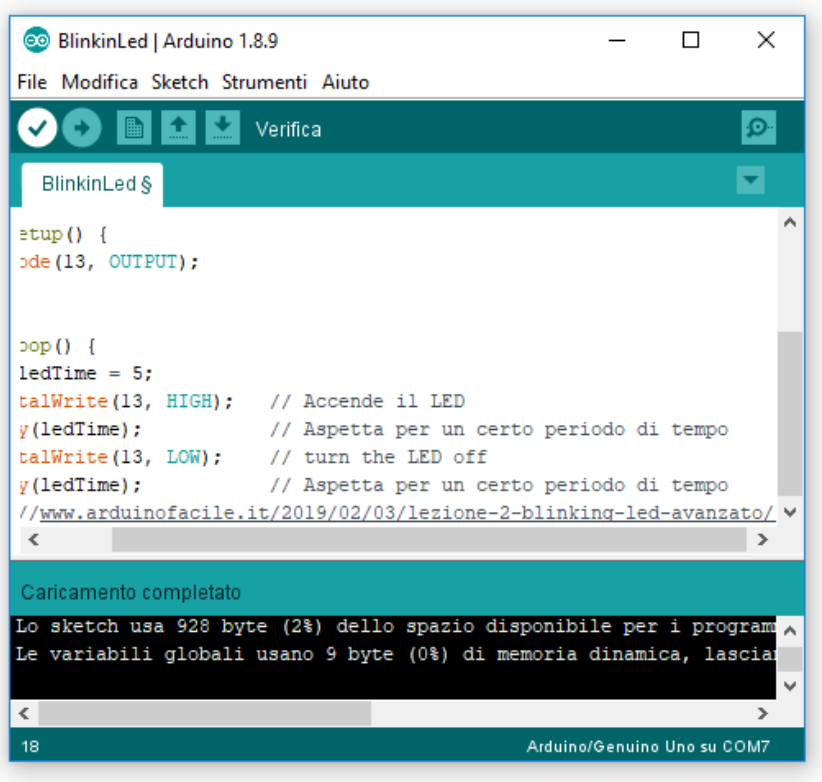
Attraverso il software di programmazione precedentemente scaricato è possibile caricare sulla piattaforma Arduino dei programmi ad hoc che abbiamo il compito di svolgere determinate operazioni utili per trasformare i propri input in output specifici.
A seguire vengono riportate le principali istruzioni utili per caricare un programma prestabilito (nel caso specifico blinking led avanzato)

- Aprire il software Arduino
- Collegare la scheda Arduino al PC mediante cavo USB
- Impostare la porta di comunicazione corretta (**IMPORTANTE: se la porta non viene impostata non è possibile caricare il programma**)



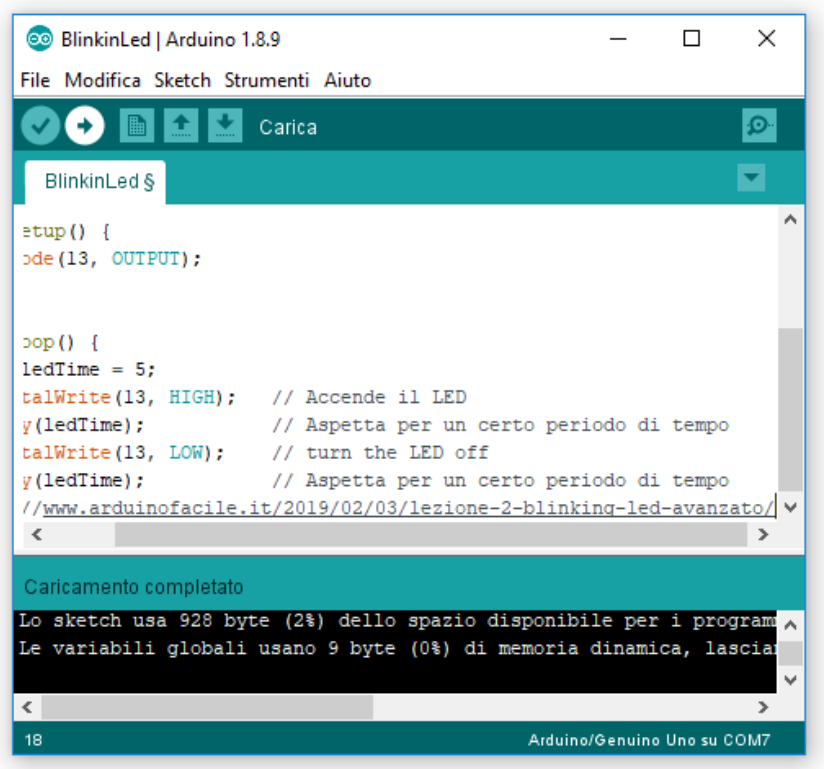
Configurare Porta Seriale Arduino

- r ePCompilare il codice: questa operazione serve per vedere se il codice scritto presenta degli errori. La compilazione si esegue premendo la spunta evidenziata in figura (in alto a sinistra).



Compilare il codice

- Caricare il codice sulla piattaforma Arduino (questa operazione deve essere eseguita solo se il codice e' stato opportunamente compilato).
  Il caricamento si esegue premendo la freccia a destra evidenziata in figura (in alto a sinistra).



Variabili Locali e Globali

Obiettivo: Comprendere cosa è una variabile e quali sono le differenze tra variabili locali e globali.

Teoria: Le variabili sono gli strumenti utilizzati in tutti i linguaggi di programmazione per la memorizzazione dei dati. Da un punto di vista pratico, le variabili possono essere considerate come contenitori di dati situati in una determinata area della memoria. Le variabili sono caratterizzate da tre differenti elementi: il tipo di dato, il nome, ed il valore contenuto.

- **Tipo di dato**: A seconda del dato che una variabile deve memorizzare il programmatore deve scegliere il corretto tipo di riferimento. Nella seguente tabella vengono riportati i principali tipi di dati utilizzati in Arduino

Tipo di Dato	Dimensione	Descrizione
char	1 Byte	Contenitore per caratteri e valori alfanumerici (e.g., ‘a’, ‘b’, ‘1’, etc)
bool	1 Byte	Contenitore per valori booleani (e.g., true, false)
int	2 Byte	Contenitore per numeri interi nel range numerico, da -32768 a 32767.
unsigned int	2 Byte	Contenitore per numeri interi senza segno nel range numerico, da 0 a 65535.
long	4 Byte	Contenitore per numeri interi nel range numerico, da -2147483648 a 2147483647
unsigned long	4 Byte	Contenitore per numeri interi nel range numerico, da 0 to 4,294,967,295 (2^32 – 1)
float	4 Byte	Contenitore per numeri con la virgola nel range numerico da 3.4028235E+38 a -3.4028235E+38
double	4 Byte	Contenitore per numeri con la virgola nel range numerico da 3.4028235E+38 a -3.4028235E+38
String		Contenitore per testo (e.g., “Resistenza”)

- **<u>Identificativo (Nome della variabile):</u>** La variabile è caratterizzata da un nome attraverso il quale si fa riferimento al contenitore stesso dei dati. E' importante considerare che il nome della variabile non deve iniziare con un numero e soprattutto non può essere uguale ad una keyword del linguaggio di programmazione. Ad esempio il nome di una variabile non può essere loop.

- **Valore:** Le variabili sono utilizzate per contenere dei dati pertanto ad ogni variabile è associato un valore (nel caso in cui l'utente non assegni nessun valore molto probabilmente è il compilatore che assegna un valore di default).

Prima di potere essere utilizzate all'interno del codice le variabili devono essere dichiarate e se possibile inizializzate con un valore specifico. La sintassi utilizzata per la dichiarazione di una variabile è la seguente.

```
int lato = 10;
float peso = 43.5;
bool condizione = true;
char lettera = 'A';
String nome = "Andrea"
```

Le variabili possono essere definite sia all'interno dei blocchi loop e setup sia al loro esterno. In base a dove vengono definite, queste prendono il nome di **variabili globali** o **locali**.

- **Variabili Globali:** Sono variabili che hanno visibilità in tutto il programma. Le variabili globali sono

definite fuori dal corpo (fuori dalle parentesi graffe) delle funzioni loop e setup.

- **Variabili Locali:** Sono variabili che hanno visibilità solo all'interno di una funzione. Le variabili locali sono definite dentro il corpo (dentro le parentesi graffe) delle funzioni loop e setup.

Codice: Viene in seguito riportato l'esempio di un codice per la gestione di un un pulsate dove sono utilizzate sia variabili locali che variabili globali.

Le Funzioni Setup() e Loop()

Obiettivo: Conoscere le caratteristiche principali delle funzioni Setup() e Loop()

Teoria: Le funzioni **Setup()** e **Loop()** rappresentano il core principale di ogni programma sviluppato utilizzando il controllore Arduino. Ogni programma parte infatti dal codice definito all'interno di queste funzioni. Nello specifico le due funzioni sono così caratterizzate:

- **Setup():** In questa funzione è definita la configurazione

iniziale di Arduino. Attraverso l'istruzione **pinMode** è possibile stabilire quali pin di Arduino sono utilizzati per gestire ingressi e quali per gestire uscite. Vengono inoltre inizializzati i vari oggetti impiegati nel loop (e.g., **Serial.begin(9600)**). E' importante considerare che le istruzioni presenti nel setup vengono eseguite solamente una volta all'avvio del controllore Arduino.

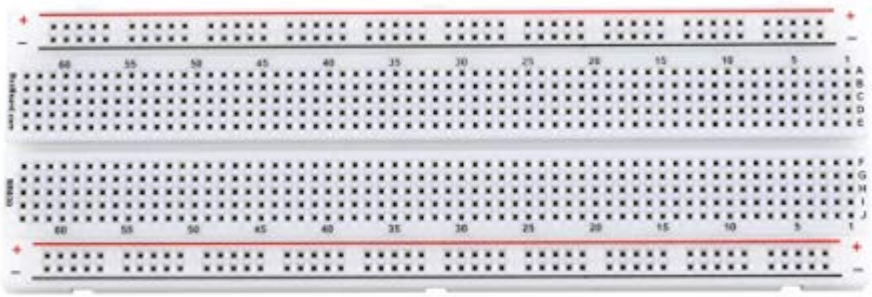
- **Loop():** In questa funzione è definito il processing ovvero le operazioni che Arduino in modo ciclico. E' importante considerare che, a differenza dalla funzione di setup, le istruzioni presenti nel loop vengono eseguite ciclicamente: **terminata l'ultima istruzione si ricomincia con la prima** (per sempre). Le principali istruzioni presenti all'interno del corpo della funzione sono quelle utilizzate per gestire gli ingressi e le uscite (e.g., digitalRead, digitalWrite, analogRead, analogRead).

E' importante considerare che il corpo di una funzione è definito mediante le parentesi graffe. Nello specifico, la parentesi graffa aperta { rappresenta l'inizio della funzione, mentre la parentesi graffa chiusa } indica la sua fine.

Codice: Vengono in seguito riportati due esempi di codice. Il primo riguarda un progetto vuoto, mentre il secondo è relativo al codice di implementazione utilizzato per fare lampeggiare un led.

La Breadboard

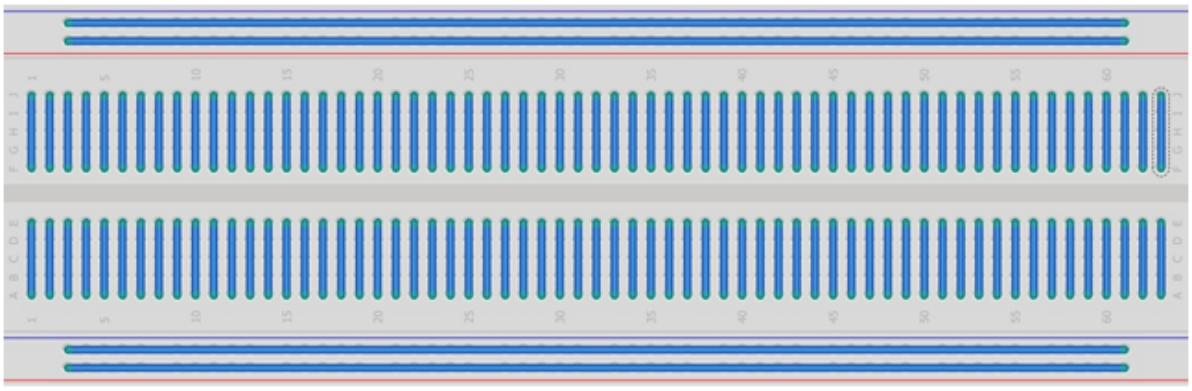
Obiettivo: Conoscere le caratteristiche e il funzionamento di una breadboard



Teoria:

La breadboard è una basetta che serve a creare prototipi di circuiti elettrici. Ha dei fori su cui vengono inseriti i cavi. Non necessita di saldature e può essere riutilizzata

Lo schema funzionale della breadboard è il seguente:

- 

Le righe continue rappresentano il collegamento tra i fori.

Le breadboard sono un elemento fondamentale nella costruzione di circuiti con Arduino. Grazie a questo prezioso strumento, è possibile eseguire velocemente i collegamenti tra i vari componenti e moltiplicare le prese di alimentazione e di contatto degli ingressi e uscite della scheda.

Le breadboard sono modulari e possono essere assemblate posizionandole una accanto all’altra.

Come installare l’ambiente Software di Arduino

Obiettivo: Installare il Software Arduino

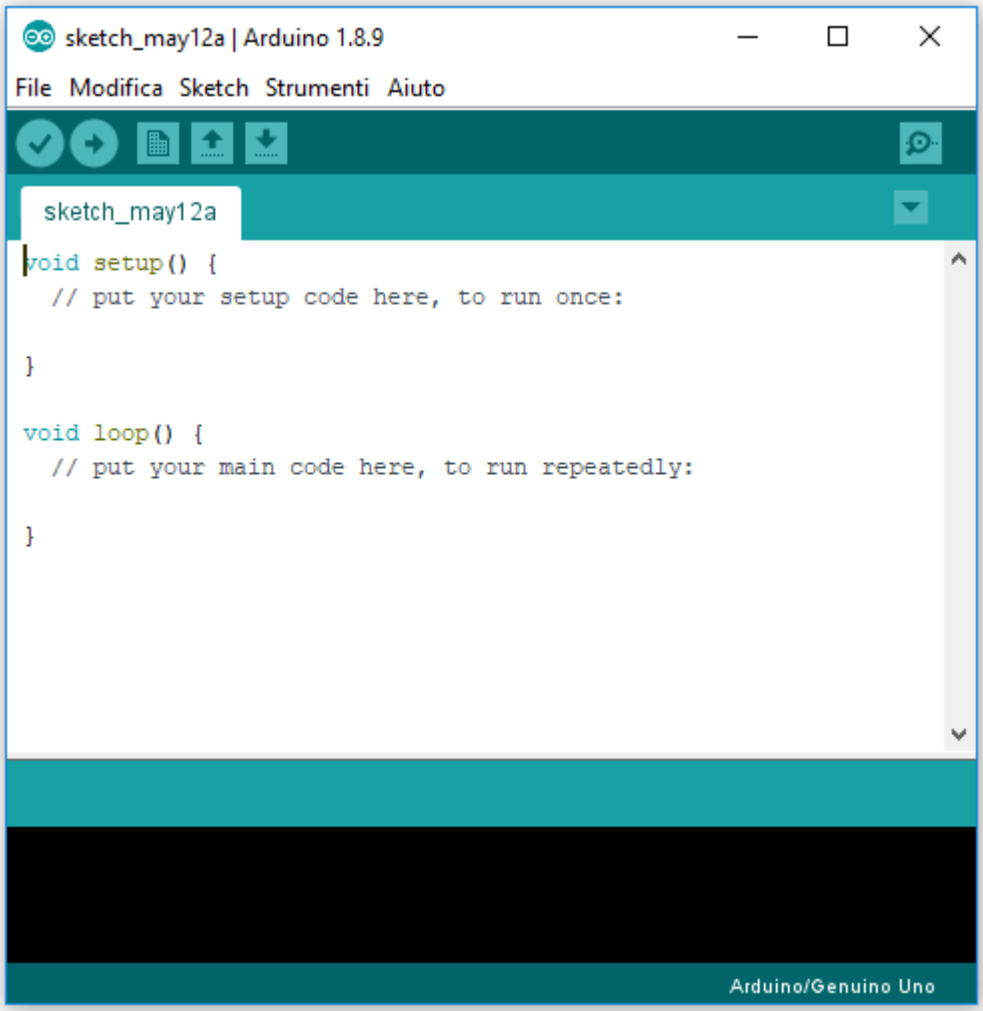
Teoria:

Qualunque dispositivo elettronico programmabile presente in commercio (ad esempio, una lampada intelligente, un cancello automatico, una robot esploratore) è costituito da un **Hardware** dedicato e da un **Software** specifico. Nel caso di Arduino (strumento attraverso il quale è possibile creare una moltitudine di dispositivi elettronici) la programmazione avviene attraverso l'impiego di un software scaricabile dal sito ufficiale. Mediante questo ambiente di sviluppo è possibile scrivere un programma specifico (istruzioni necessarie per comandare l'hardware) e caricarlo sul microcontrollore.

A seguire sono riportate le istruzioni per scaricare l'ambiente di sviluppo di Arduino:

- Aprire il browser
- Collegarsi al sito www.arduino.cc
- Nella sezione software scaricare il software specifico per il proprio sistema operativo (software windows, software mac)
- Selezionato il file da scaricare è possibile effettuare una donazione oppure semplicemente scaricare il software gratuitamente.

Una volta installato, il programma appare così all'utilizzatore:



Software Arduino