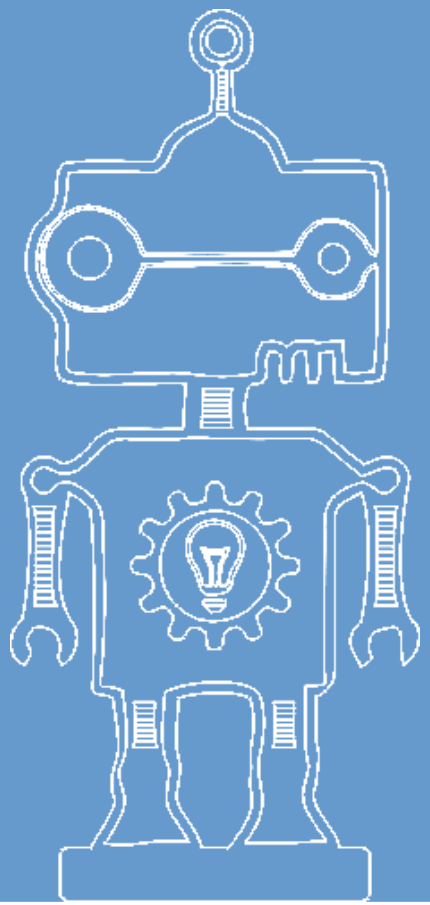


ROBRUTTINO THEREMINO

Workshop Arduino
Febbraio 2016

FAB LAB PALERMO

Viral digital strategies get infected

SENSEJ open source things server

28/02/2016
Palermo

PMO COWORKING

Via Principe di Belmonte, 93

Scopo del workshop

- Apprendere le basi di Arduino
  - Cos'è
  - A cosa serve
  - Come funziona
- Imparare a programmare Arduino
  - Come specificare al microcontrollore che cosa fare
- Costruire il progetto
  - Robruttino Theremino

Modalità di svolgimento

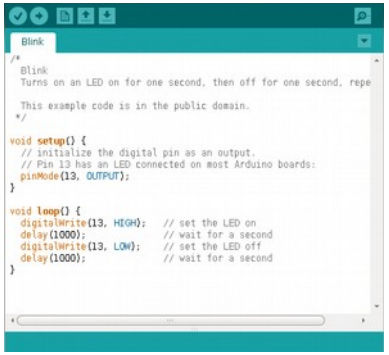
- **Durata 8 ore**
  - 10:00 – 13:00 Teoria
  - 14:00 – 19:00 Project Making
- **Parte teorica**
  - Aprire le slide sul proprio PC
  - Le domande sono benvenute
  - Eventuali approfondimenti vanno richiesti alla fine
- **Pratica**
  - Ogni partecipante autocostruirà il progetto
  - A ogni tavolo sarà presente un Tutor per supporto

Materiali di supporto

- **Slideshow**
  - Questa presentazione
    - Disponibile nella cartella progetto
    - Sul gruppo Facebook
- **Sketch**
  - Il codice sorgente del progetto
    - Sul gruppo Facebook
- **Arduino IDE**
  - http://www.arduino.cc/download

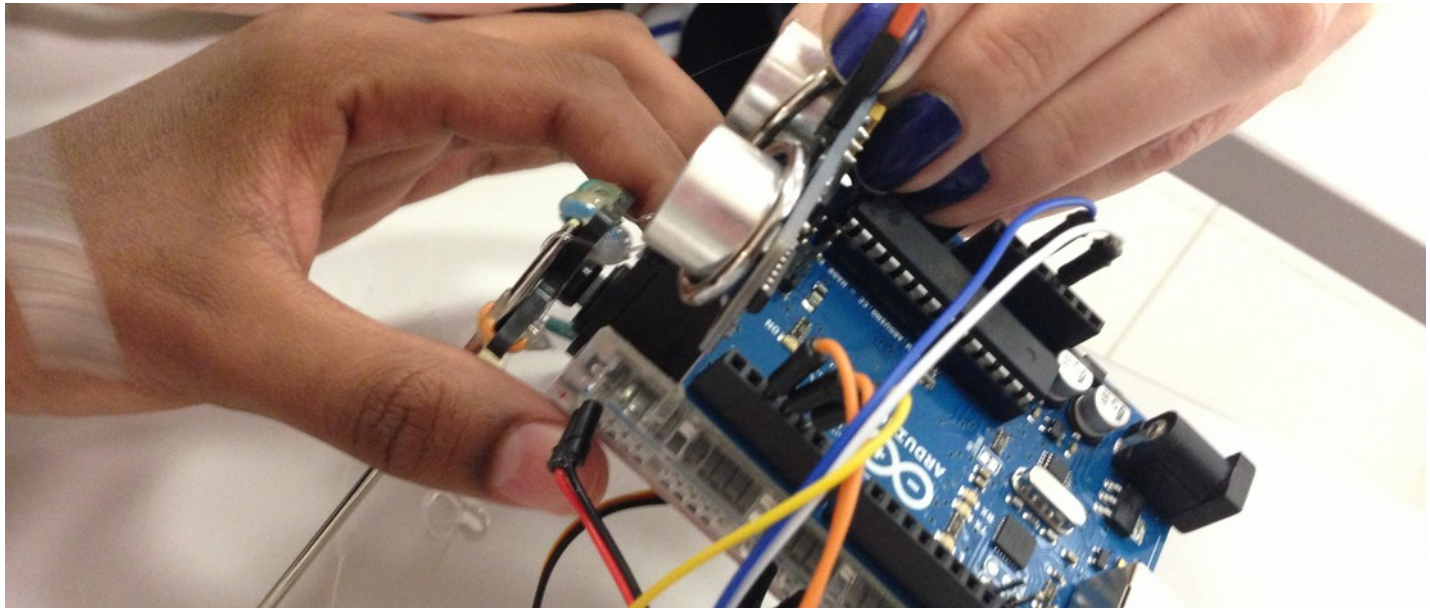
INTRODUZIONE AD ARDUINO

ARDUINO
OPEN-SOURCE
COMMUNITY



Il contesto

- **Physical Computing**
  - **Sistemi fisici interattivi** che, grazie all'uso di hardware e software, **percepiscono e rispondono** al mondo analogico.



Internet delle Cose

- **Internet of Things**
  - L'Internet of Things (IoT) è la **rete di oggetti fisici** (o things) equipaggiati con elettronica, software, sensori e connettività, che consentono a questi oggetti di **collezionare e scambiare dati su Internet**.

Il mondo Maker

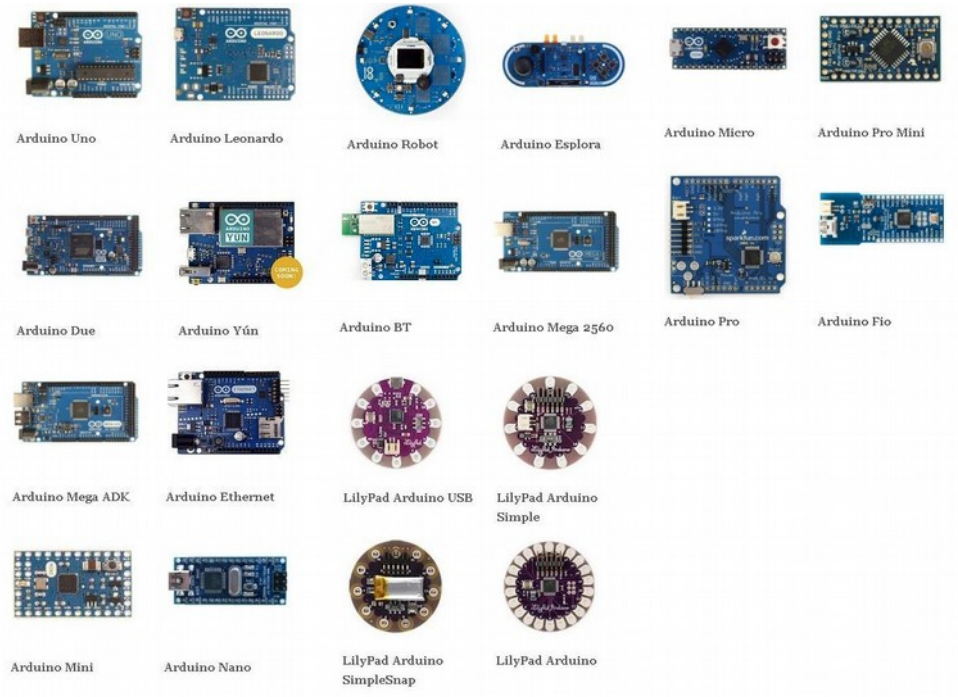
Hobbisti e professionisti con interessi e valori comuni

- Passione di creare innovazione
- Interazione fra digitale e reale
- Condivisione del sapere
- Collaborazione
- Open

Arduino

Arduino è una scheda elettronica
con un **microcontrollore** (Atmel) e circuiteria,
utile per creare rapidamente prototipi
e per scopi hobbistici e didattici.

- **Boards**
  - Uno, Nano, Mega, etc..
- **Gli Sketch**
  - App per il microcontrollore
- **Il Playground**
  - La community
    e la codebase arduino.cc



Arduino Nano

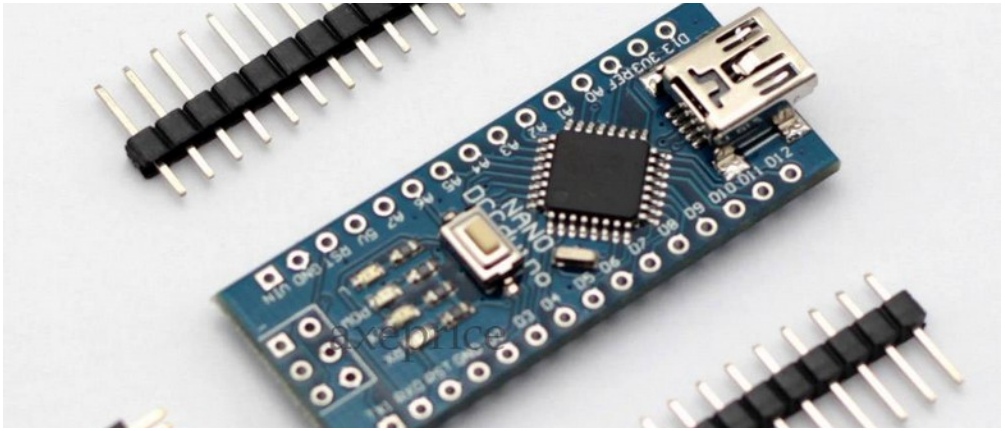
Microcontroller Atmel ATmega328

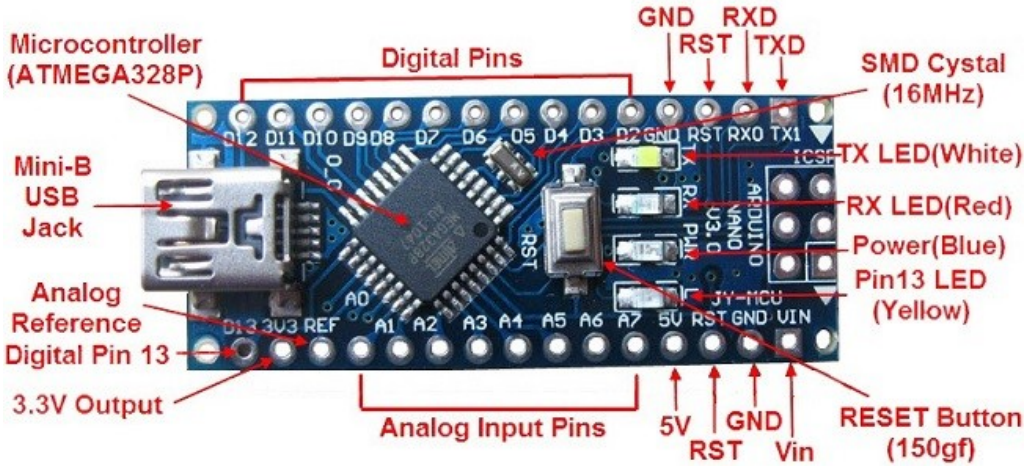
Operating Voltage (logic level) 5 V
Input Voltage (recommended) 7-12 V
Input Voltage (limits) 6-20 V

Digital I/O Pins 14 (of which 6 PWM)
Analog Input Pins 8
DC Current per I/O Pin 40 mA

Flash Memory 32 KB
of which 2 KB used by bootloader
SRAM 2 KB
EEPROM 1 KB
Clock Speed 16 Mhz

Dimensions 0.73" x 1.70"
Length 45 mm
Width 18 mm
Weigth 5 g



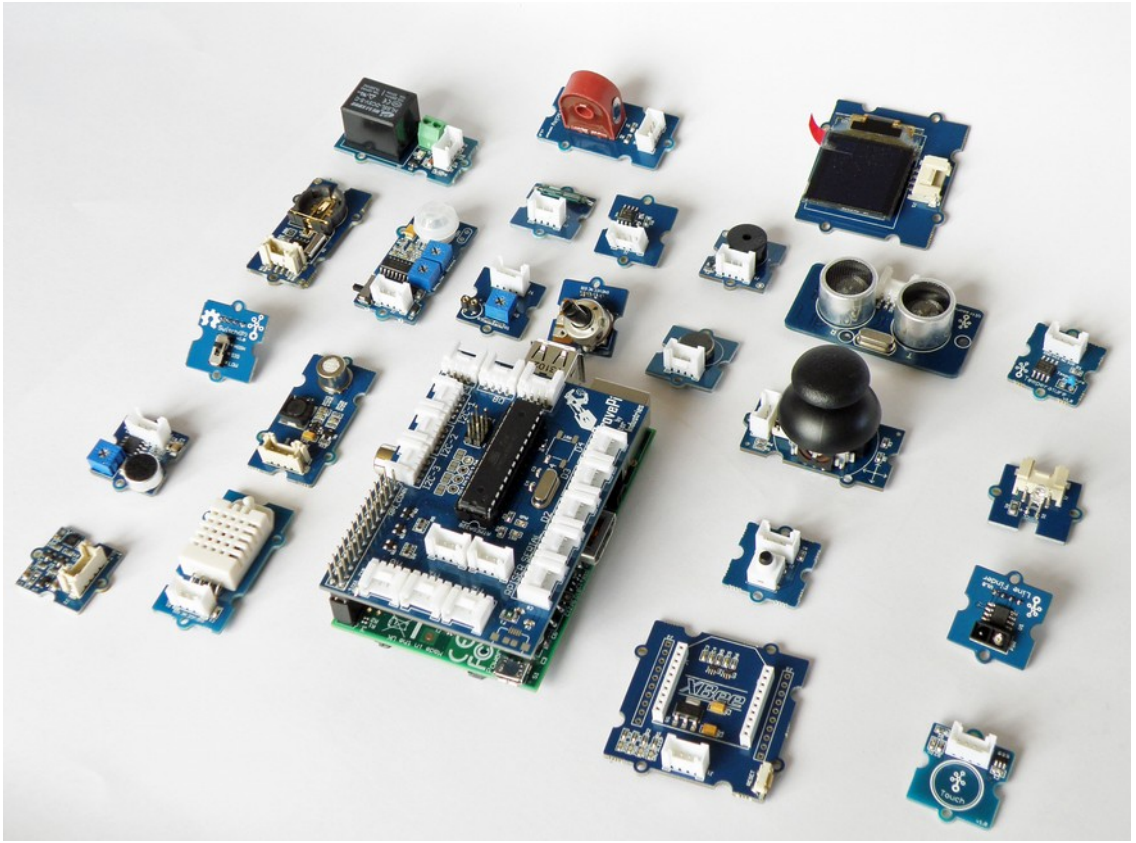


Attuatori da comandare

- **Led**
  - Spie, illuminazione...
- **Speaker**
  - Suoni, allarmi..
- **Motori**
  - Stepper, servo, DC
- **Display**
  - LCD, OLED, etc.
- **Relay**
  - Power on/off

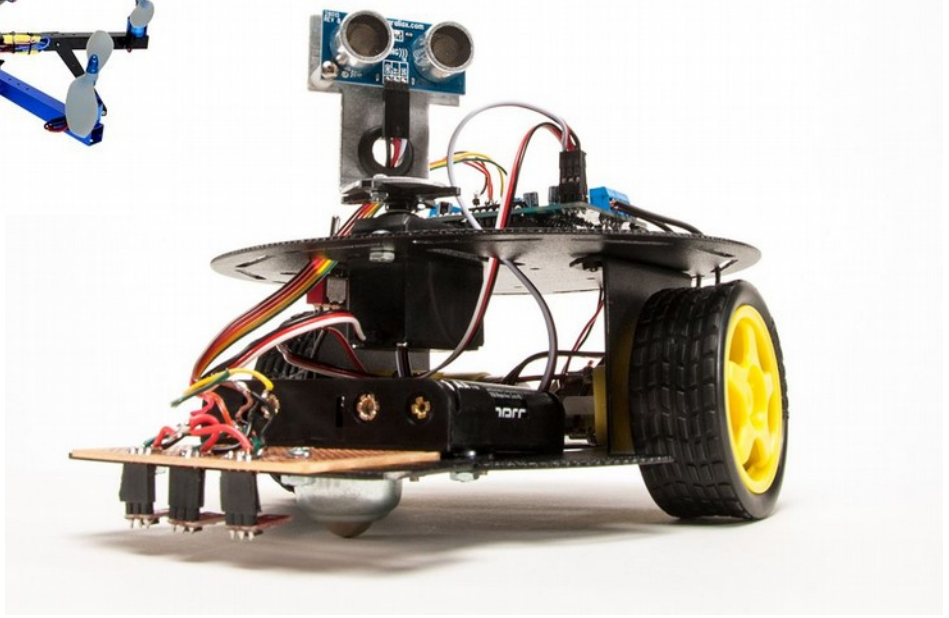
Sensori da leggere

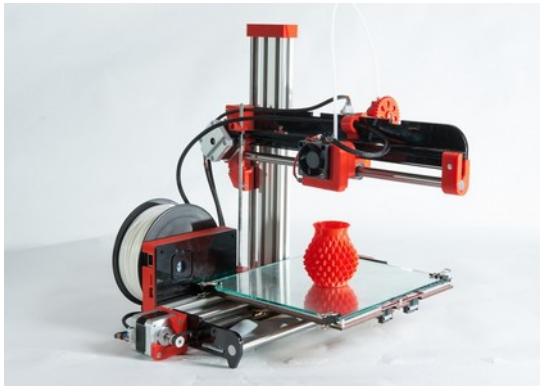
- **Ambientali**
  - Temperatura
  - Umidità
  - Pressione atmosferica
  - etc
- **Prossimità**
  - Infrarossi
  - Ultrasonici
  - Rumore ambientale
- **Biometrici**
  - Battito cardiaco
  - Temperatura



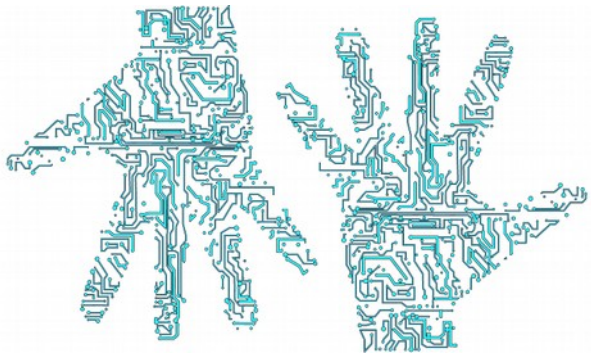
Possibili applicazioni

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



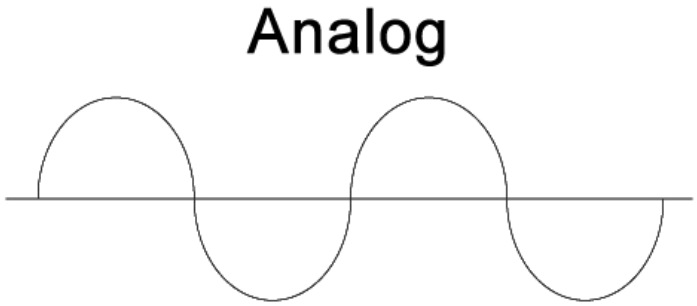


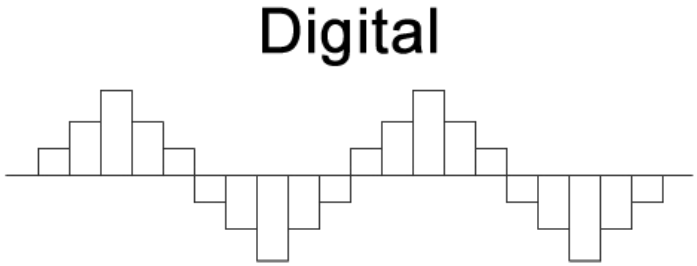
FONDAMENTI DI INFORMATICA

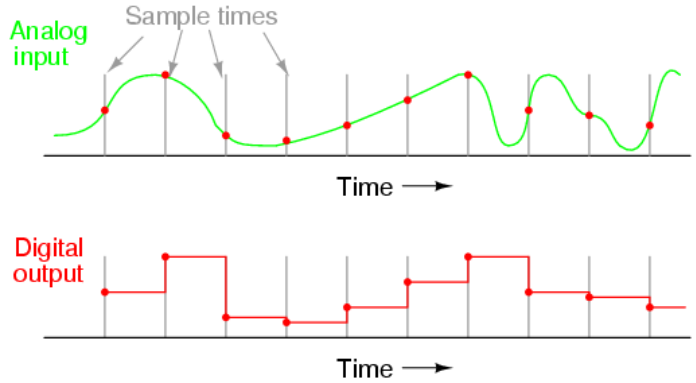


Segnali

- **Segnali analogici**
  - Continui
  - Informazione infinita
- **Segnali digitali**
  - Discreti
  - Informazione quantificabile
- **Campionamento**
  - Digitalizzazione di un segnale





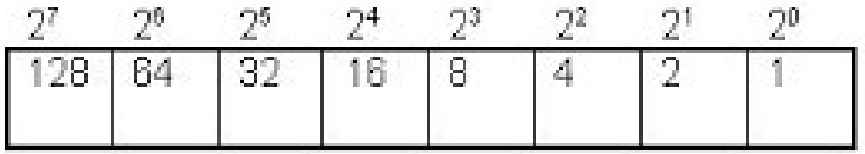


Informazione digitale

- **Sistema binario**
  - Sistema posizionale
  - Base 2
  - Ogni cifra è una potenza di 2
    - 2^0, 2^1, 2^2, 2^3,...
    - 1,2,4,5,16,32,64,128,256,...

- Esempi

- 8 = 1000
- 10 = 1010
- 12 = 1100



So the following binary pattern would be

0	1	1	0	0	0	0	1

(1 × 64) + (1 × 32) + (1 × 1) = 97

Place values
(multiply this number by the 1 or 0 in its place)

128	64	32	16	8	4	2	1
×	×	×	×	×	×	×	×
1	0	1	1	0	1	0	1
=	=	=	=	=	=	=	=

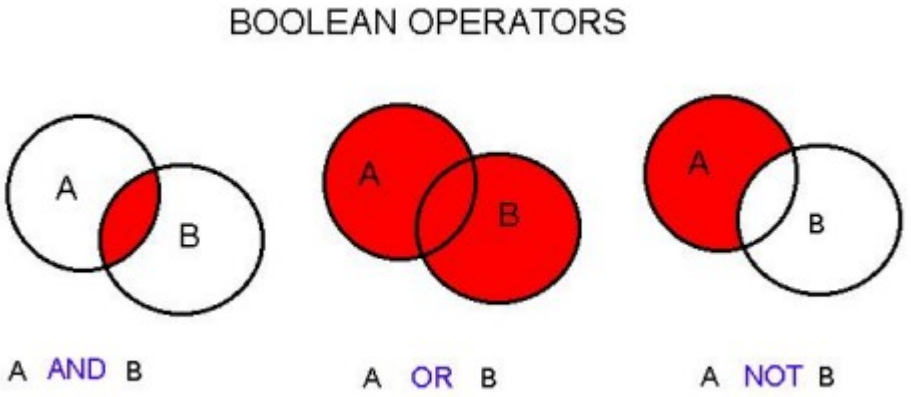
128 + 0 + 32 + 16 + 0 + 4 + 0 + 1

(add all these together to get the decimal number)

= 181

Algebra Booleana

- **Valori booleani**
  - Valori logici opposti
    - Vero / Falso
    - 1 / 0
- **Operatori logici**
  - AND &&&
  - OR | ||
  - NOT ¬ !
  - XOR ^ !=
  - XNOR ≡ ==



A	B	A | B	A & B	A ^ B	~A
0	0	0	0	0	1
1	0	1	0	1	0
0	1	1	0	1	1
1	1	1	1	0	0

Informazione binaria

Zettabyte	1.000.000.000.000.000.000.000 Byte
Exabyte	1.000.000.000.000.000.000 Byte
Petabyte	1.000.000.000.000.000 Byte
Terrabyte	1.000.000.000.000 Byte
Gigabyte	1.000.000.000 Byte
Megabyte	1.000.000 Byte
Kilobyte	1.000 Byte
Byte	1 Byte

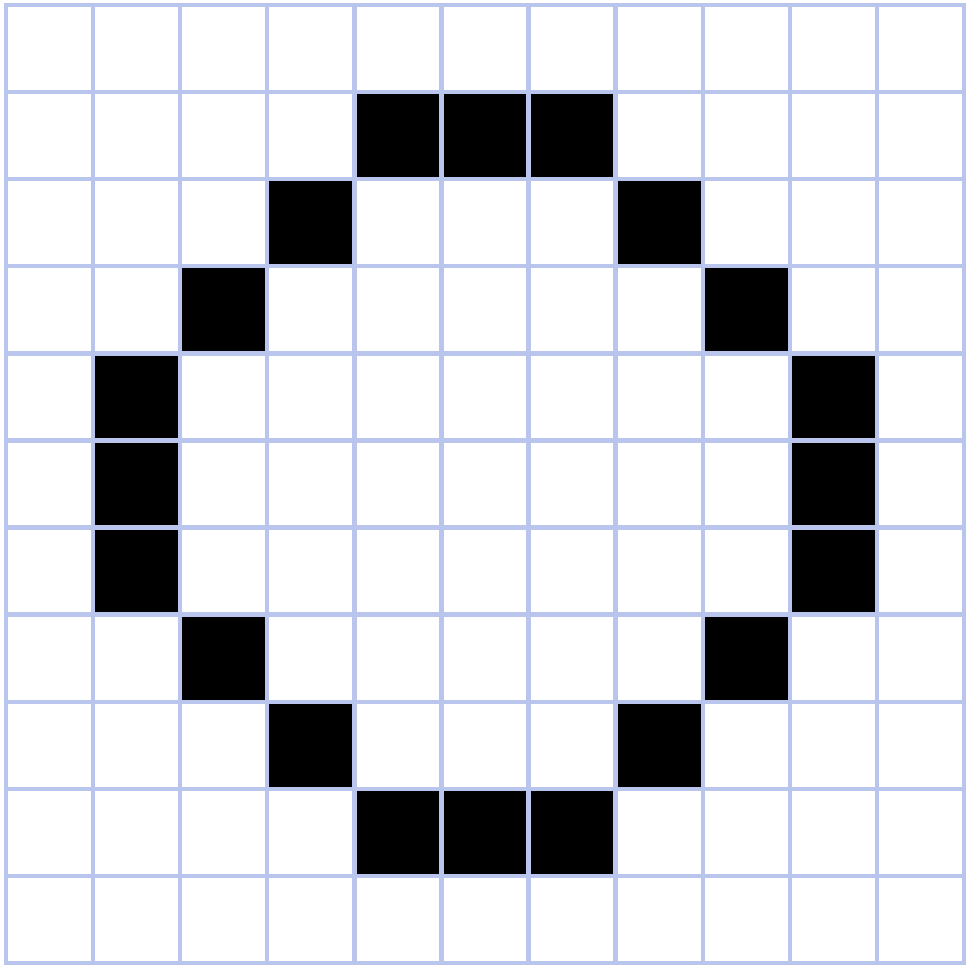
- **bit (b)** = 0-1
- **Byte (B)** = 8 bit = 0-255
- **KiloByte (KB)** = 1024 Byte
- **MegaByte (MB)** = 1024 KiloByte = ~1 Mln Byte
- **GigaByte (GB)** = 1024 MegaByte =~1 Mld Byte

kilobyte (kB)	10^3	2^{10}	kibibyte (KiB)	2^{10}
megabyte (MB)	10^6	2^{20}	mebibyte (MiB)	2^{20}
gigabyte (GB)	10^9	2^{30}	gibibyte (GiB)	2^{30}
terabyte (TB)	10^{12}	2^{40}	tebibyte (TiB)	2^{40}

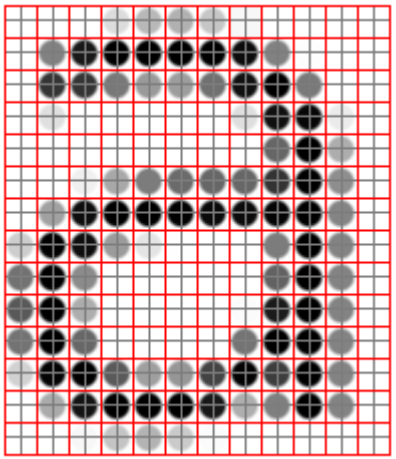
ASCII Table

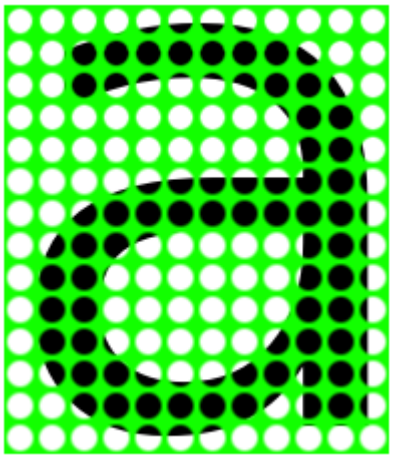
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Bitmap Image Representation

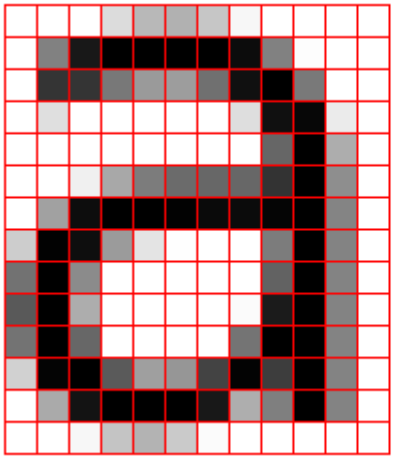


a





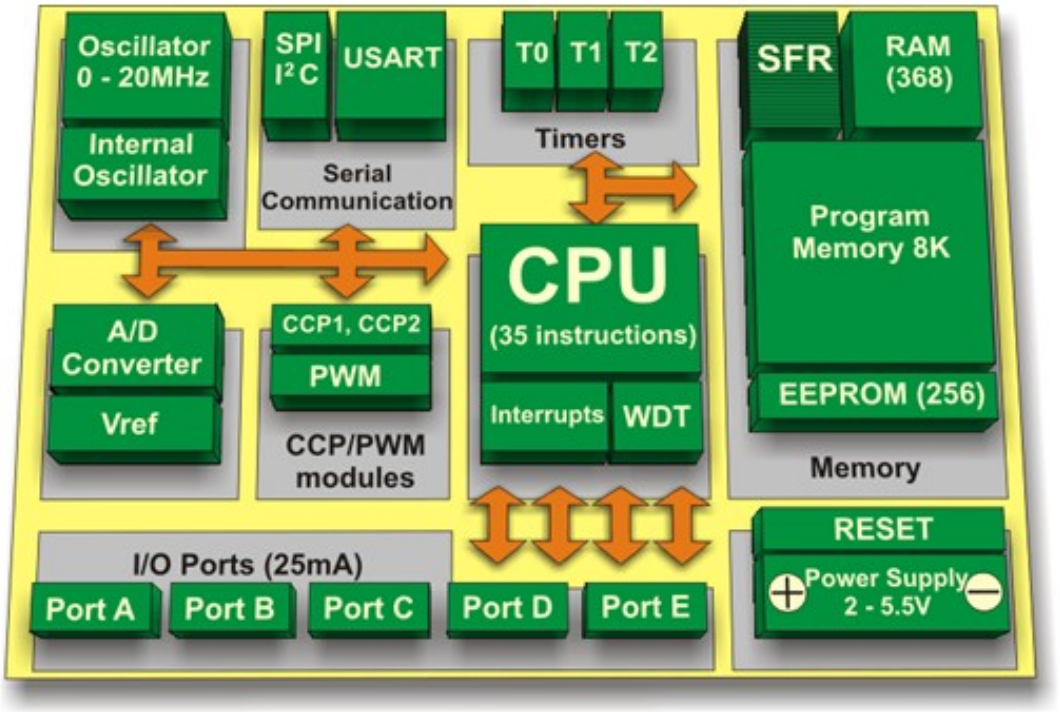
a



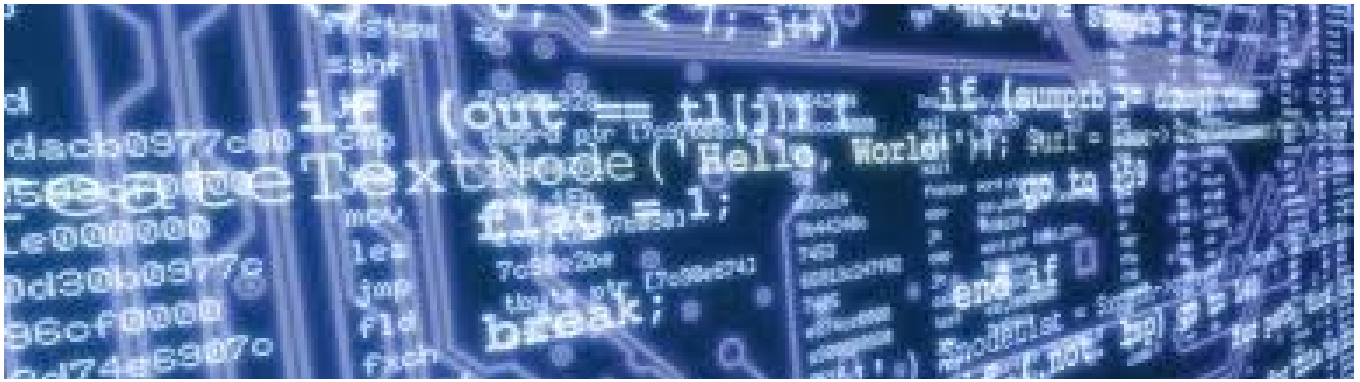
1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0	
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0	
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0	
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0	
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0	
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0	
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0	
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0	
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0	
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0	
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0	
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0	

Architettura di base di un Microcontrollore

- **CPU e MCU**
  - Microprocessore
    - Solo calcolo
    - Richiede periferiche
  - Microcontrollore
    - Sistema integrato
    - Input / output
- **Memorie volatili**
  - RAM
- **Memorie permanenti**
  - EEPROM, Flash,
  - Hard disk, etc...

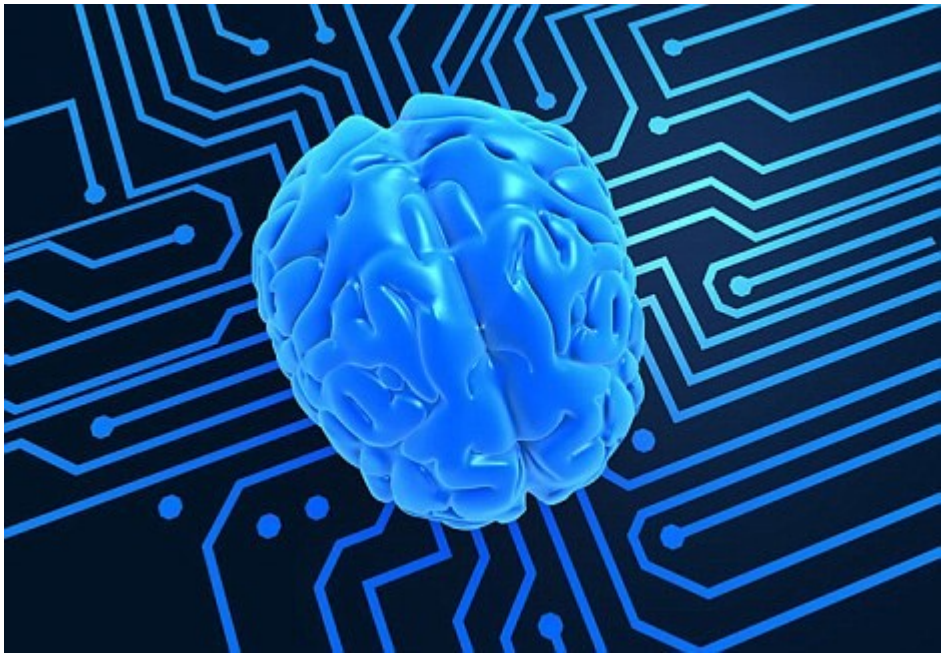


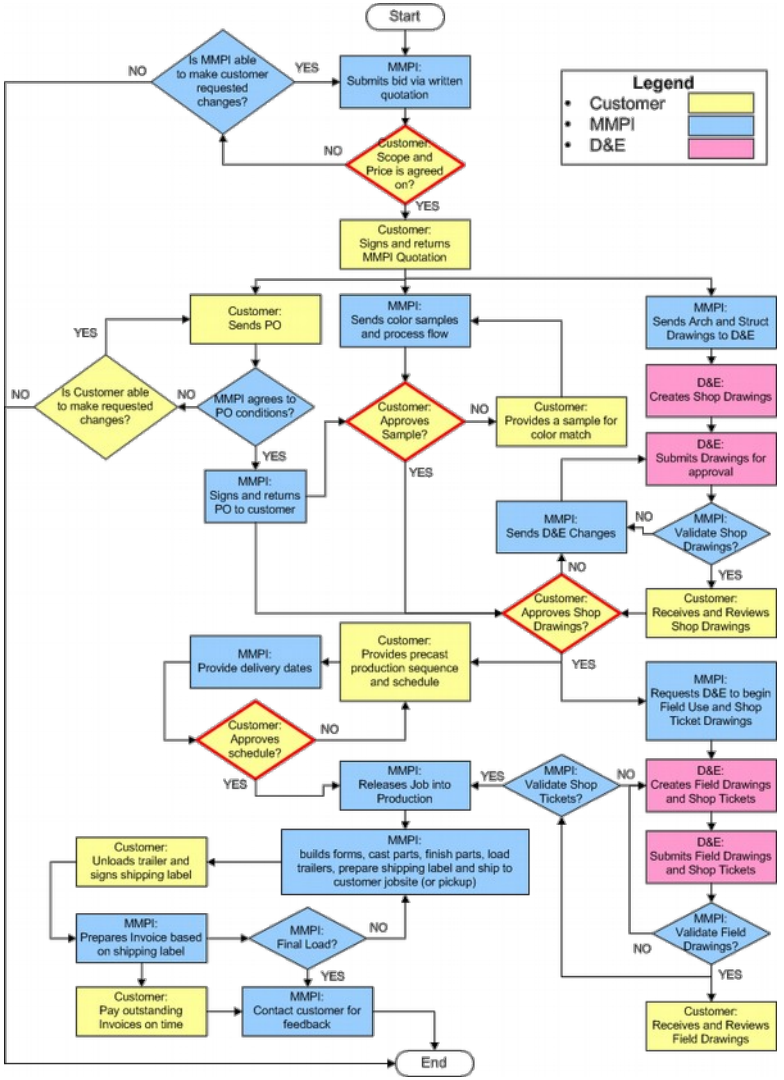
FONDAMENTI DI PROGRAMMAZIONE



Logiche di un calcolatore

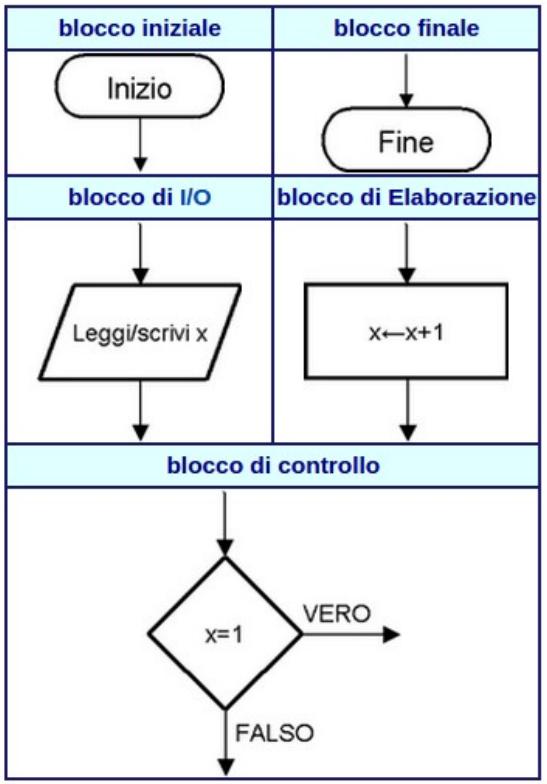
- **Come “ragiona” un microcontrollore**





Algoritmi

- **Il concetto**
  - un **procedimento** che risolve un determinato problema attraverso un numero finito di passi elementari
- **Proprietà fondamentali**
  - **Atomicità**: passi elementari
  - **Non ambiguità**: interpretazione univoca
  - **Finitezza**: passaggi finiti e input finito
  - **Terminazione**: tempo finito
  - **Effettività**: risultato univoco
- **Rappresentazione**
  - Diagrammi di flusso



Linguaggi

- **Linguaggio macchina**
- **Linguaggi di basso livello**
- **Linguaggi di alto livello**

```
1  #include<iostream>
2  using namespace std;
3
4  int main() {
5  int number, reverse = 0;
6   cout<<"Input a Number to Reverse:  ";
7   cin>> number;
8
9     for( ; number!= 0 ; )
10    {
11       reverse = reverse * 10;
12       reverse = reverse + number%10;
13       number = number/10;
14    }
15    cout<<"New Reversed Number is:  "<<reverse;
16
17    return 0;
18 }
```

ADD	00001	RD	RS2	RS1
SUB	00010	RD	RS2	RS1
AND	00011	RD	RS2	RS1
XOR	00100	RD	RS2	RS1
OR	00101	RD	RS2	RS1
NOR	00110	RD	RS2	RS1
MULT	01010	RD	RS2	RS1

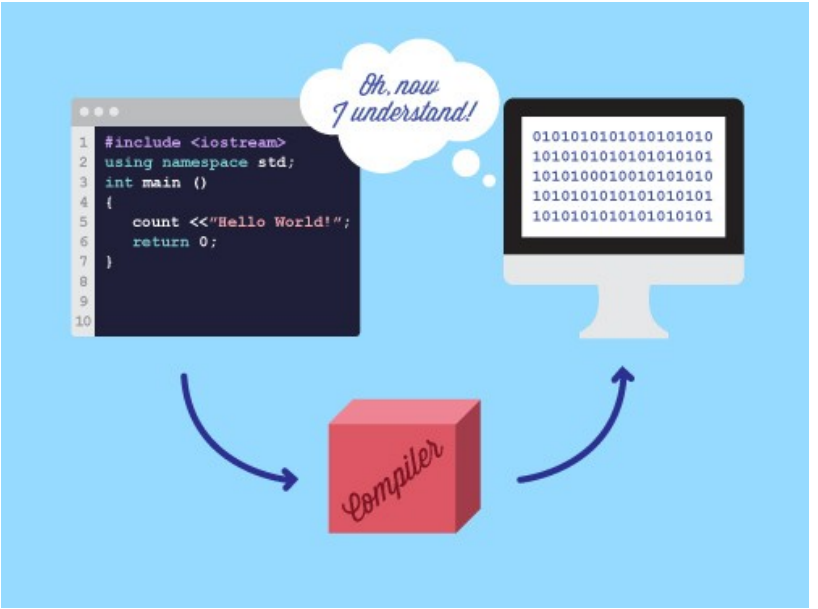
LOADPC	01011	RD	xxxxxxxxxx	
LOADIMM	01001	# data	xx	RD
LOAD	00111	RD	xxxxx	RS1
STORE	01000	xxxxx	RS1	RD

JMPZ	01101	xxxxxxxxxx	RS1
JMPIND	01110	RD	xxxxxxxxxx
JMPINDTR	01111	RD	xxxxxxxxxx

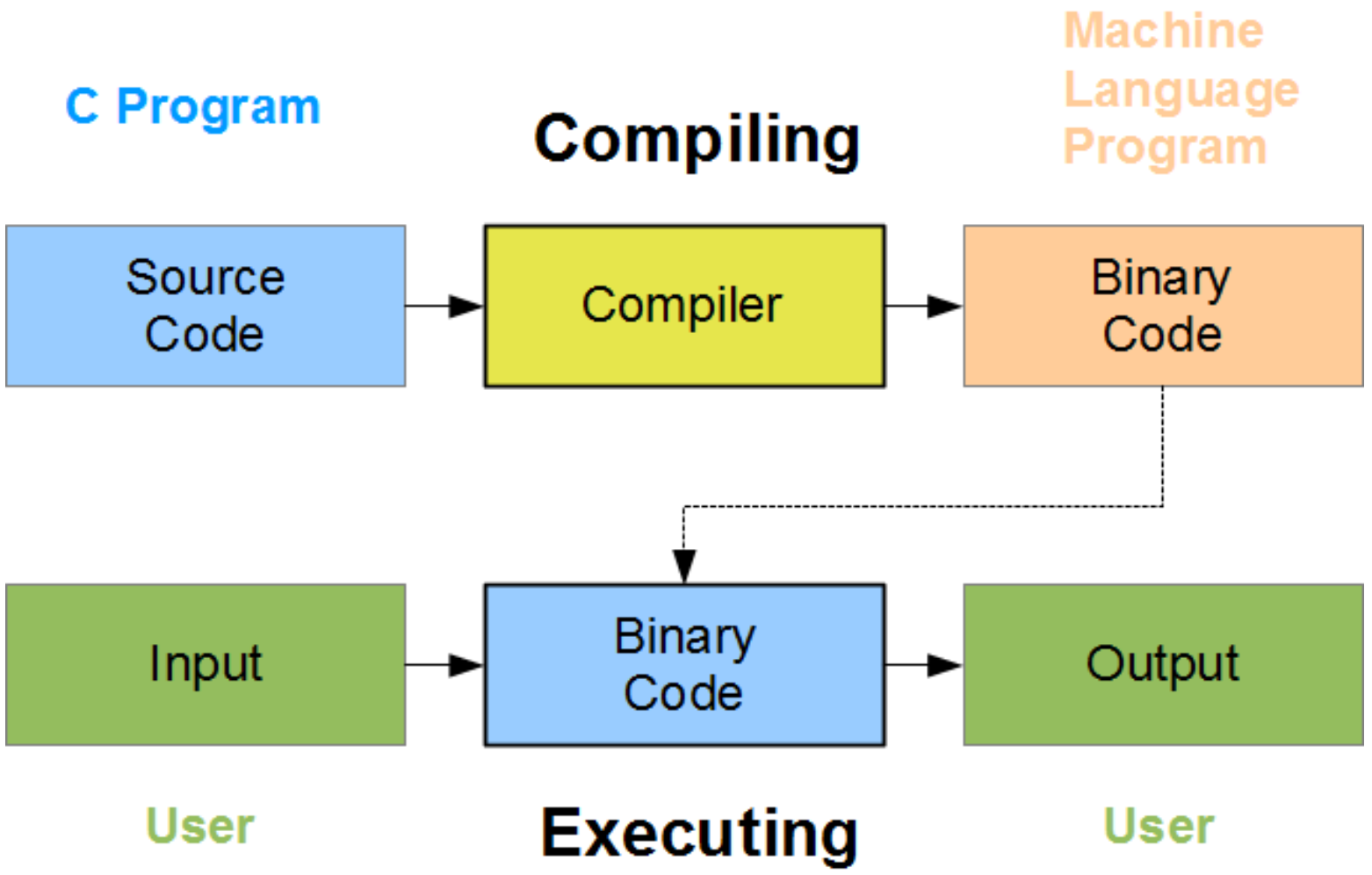
RESET	11110	xxxxxxxxxxxxxxxx
NOP	11111	xxxxxxxxxxxxxxxx

Il processo di traduzione

- **Ambienti compilati**
  - Tutto il sorgente
    - Traduzione completa
  - Si distribuisce il risultato
    - Binario eseguibile
- **Ambienti interpretati**
  - Riga per riga
    - Interpretato sul momento
  - Si distribuisce il sorgente
    - Verrà reinterpretato



Fasi del programma



Strutture di un linguaggio

- **Istruzioni**
  - Un comando che cambia lo stato interno
- **Variabile**
  - Un dato a cui il programma può fare riferimento per nome
- **Espressioni**
  - Una combinazione di variabili, costanti e operatori che verrà valutata in n valore
- **Strutture dati**
  - Organizzazioni complesse di dati
- **Strutture di controllo**
  - Condizioni, cicli, scelte, etc..
- **Sottoprogrammi**
  - Blocchi di codice riutilizzabili: funzioni
- **Funzionalità I/O**
  - Capacità di interagire con i flussi di Input/Output standard
- **Commenti**
  - Testi non interpretati dal compilatore (note, spiegazioni, etc.)

C/C++

- **Il C nasce nel 1972**
- **il C++ nel 1983**
- **Linguaggi alto livello**
  - Ma mantengono il controllo del basso livello
- **Ambiente compilato**
  - Il sorgente viene tradotto in un file binario eseguibile
- **In C/C++ sono scritti molti software di base**
  - Windows, Office, Mac OS X, Chrome, Firefox, Photoshop...

Caratteristiche del linguaggio

- **Case sensitive**
  - C'è differenza fra maiuscole e minuscole
- **Carattere di fine istruzione ;**
  - Ogni “riga” deve terminare con un punto e virgola
- **Identificatori di blocco { }**
  - Ogni “blocco” di codice è raggruppato dentro graffe
- **Fortemente tipizzato**
  - Ogni dato ha uno specifico tipo
    - Int, float, bool, char, ...

Strutture base del linguaggio

- **Macro e Costanti**
  valori che non cambiano
  - #define PI_GRECO 3.14
  - const int LedPin = 3;
- **Variabili**
  valori che cambiano nel tempo
  - int x;
  - bool a;
  - String Nome;
- **Istruzioni**
  - x=3;
  - a=false;
  - Nome=”Mario”;

- **Strutture di controllo:**

If... then... else
Permette di eseguire alcune istruzioni solo quando si verifica una condizione

```
If ( x > 3 )  {
   Serial.print(“X è maggiore di 3”);
} else {
   Serial.print(“X è minore o uguale a 3”);
}
```

Le funzioni

Funzioni

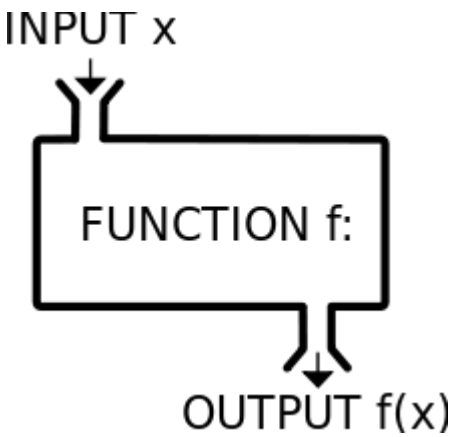
sottoprogrammi che elaborano un risultato su dei parametri passati fra parentesi

Esempio di utilizzo

```
x=somma(2,3);
x=quadrato(5);
```

Esempio di definizione

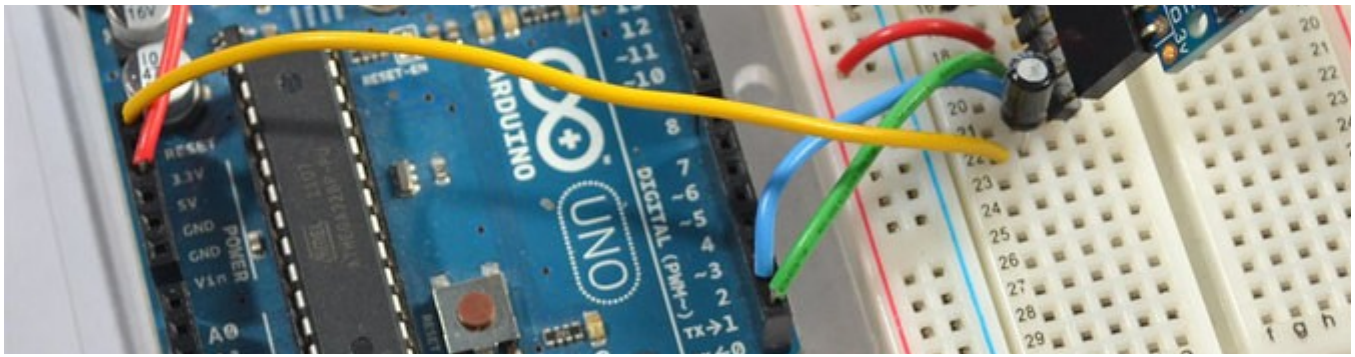
```
int somma(int a, int b){
   return a+b;
}
```



Vantaggi

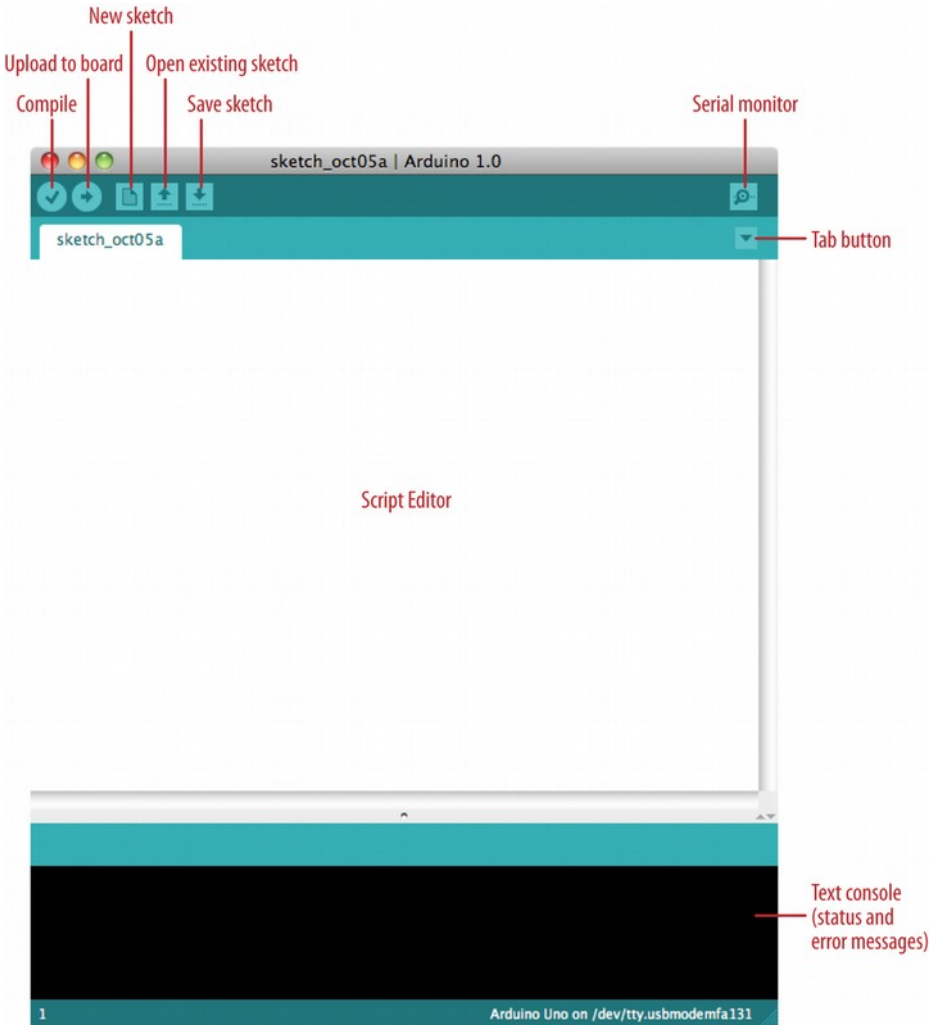
- Incapsulare una logica complessa
- Rendere generico il codice
- Poter lavorare su un pezzo per volta
- Rendere più leggibile il codice
- Rendere più manutenibile il programma

PROGRAMMARE ARDUINO



PROGRAMMARE ARDUINO

- **L'ambiente di sviluppo**
  - Installazione IDE
  - Driver dell'adattatore seriale
  - Impostazione della scheda
  - Impostazione della porta
  - Controlli base



PROGRAMMARE ARDUINO

- **Concetti di base**
  - Gli Sketch: Apps per l'arduino
    - Cartella dello sketch
    - File di progetto .ino
    - File esterni .h e .cpp
  - Cartella degli sketches
    - /home/user/Arduino
    - Documenti\Arduino

Lo sketch base

Struttura base

- **setup()**

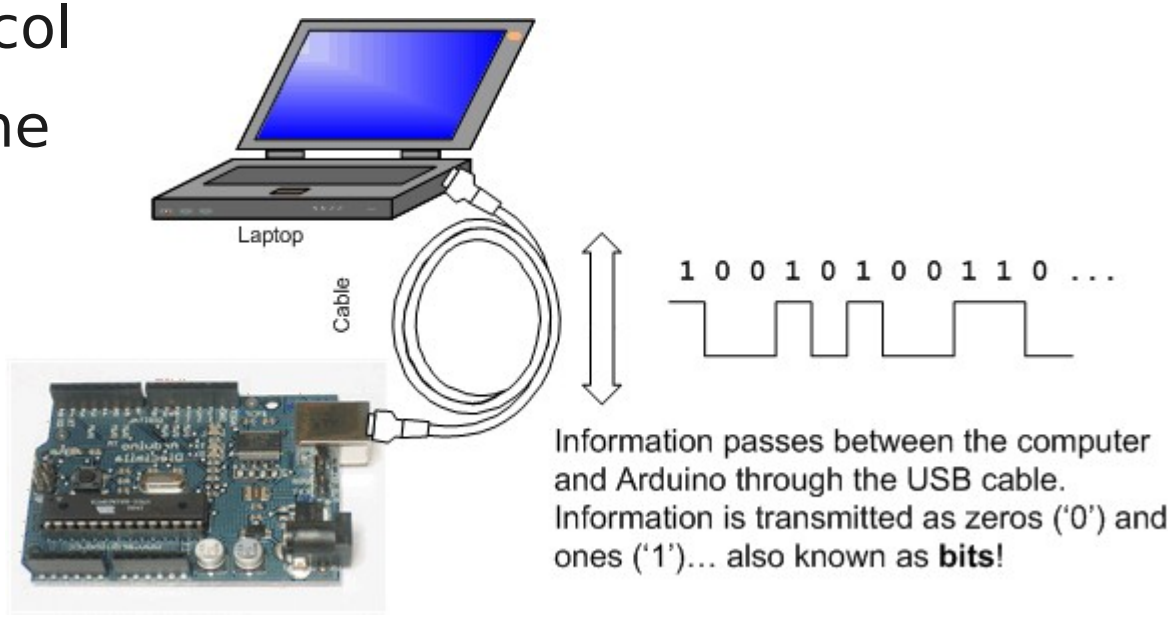
  Viene eseguito una sola volta all'avvio

- **loop()**
  - Viene eseguito di continuo

```
void setup() {

  //write here the setup code

}

void loop() {

  //write here the main code

}
```

Comunicazione Seriale

- **Comunicazione serial per il flash del firmware**
  - Per caricare un nuovo sketch sull'Arduino
- **Comunicazione seriale di debug**
  - Per comunicare col dispositivo in funzione
- **Output seriale**
  - Serial monitor



Oggetto Serial

Gestisce la comunicazione seriale

Serial.begin()

- Normalmente inserita nel setup, serve una sola volta a inizializzare la porta seriale.
- Es: Serial.begin(9600);
- Fra parentesi, la velocità, espressa in baud (bit per secondo)

Serial.print()

- Serve a “stampare” una stringa sulla porta seriale
- Es: Serial.print(“Prova”);
- Fra parentesi una variabile o un testo fisso fra virgolette

Serial.println()

- Come la Serial.print() ma aggiunge un'andata a capo alla fine.

Usare i delay

Spesso si rende necessario inserire delle pause nel codice

A questo scopo si usa la funzione **delay**()

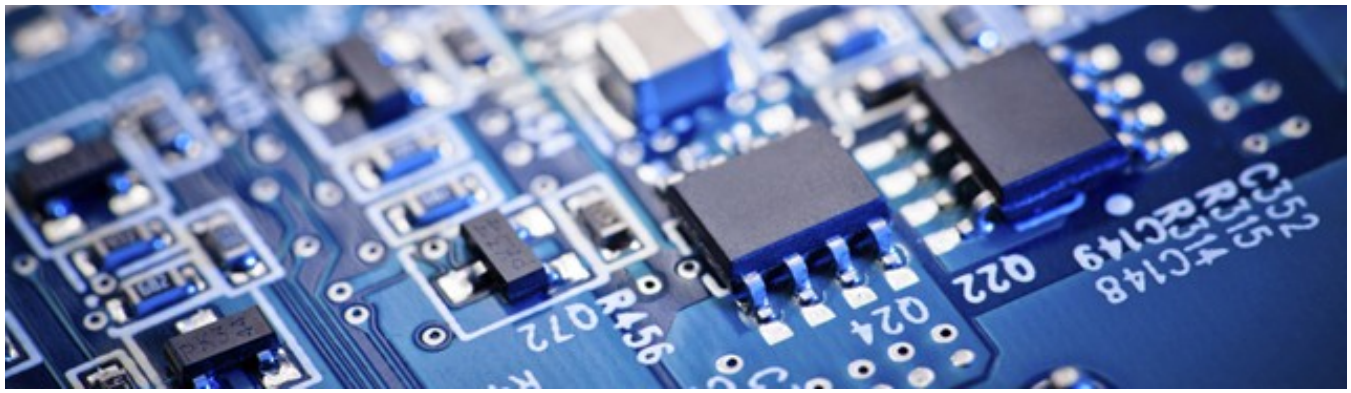
Fra parentesi accetta il numero di millisecondi da attendere

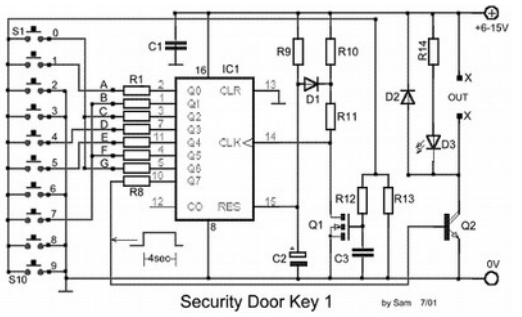
Es: delay(1000);

Test seriale: Hello world

```
void setup()
{
  //initialize serial communications at a 9600 baud rate
  Serial.begin(9600);
}

void loop()
{
  //send 'Hello, world!' over the serial port
  Serial.println('Hello, world!');
  //wait 100 milliseconds so we don't drive ourselves crazy
  delay(100);
}
```

FONDAMENTI DI ELETTRONICA



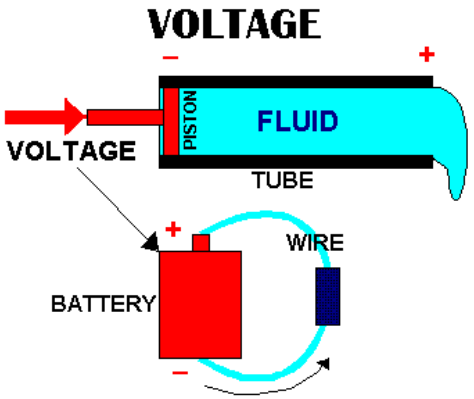


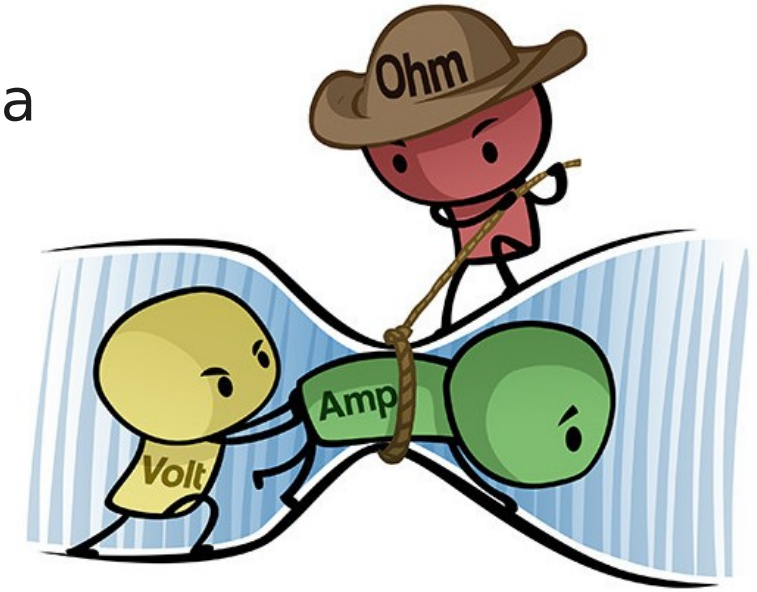
Tensione e corrente

- **Tensione**
  - Quando due corpi hanno una differenza di carica, quindi l'uno ha una quantità di elettroni diversa dall'altro, si ha una differenza di elettroni e quindi una d.d.p. (**differenza di potenziale**) o tensione.
  - Essa si misura in **Volt** con simbolo **V**
- **Corrente**
  - Se colleghiamo un filo conduttore ai poli della batteria, si avrà una corrente che circolerà in esso
  - La corrente è **la quantità di elettroni** (Q - Coulomb) (simbolo C) che passano in una sezione di conduttore **nell'unità di tempo** (t).
  - La corrente è espressa in **Ampère** (simbolo: **A**)

Resitenza

- **Resistenza**
  - Esprime in che quantità un materiale **si fa attraversare da corrente**. Più alta sarà la resistenza minore sarà la corrente che circolerà a parità di tensione. Minore sarà la resistenza maggiore sarà la corrente che circolerà a parità di tensione.

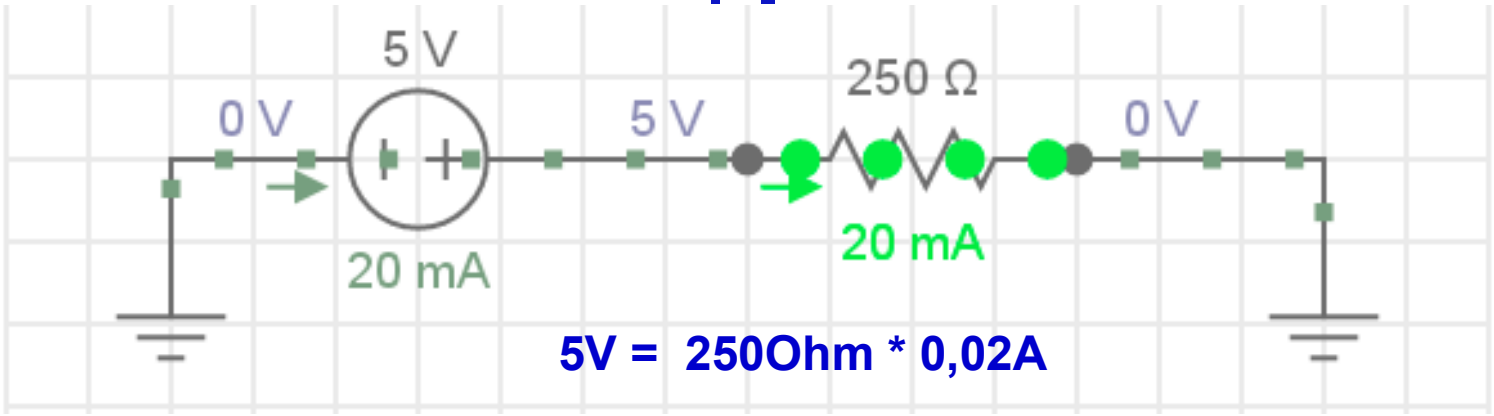




Prima legge di Ohm

- **Enunciato**
  - la **corrente** è direttamente proporzionale alla **tensione** ed inversamente proporzionale alla **resistenza**
- **Formule**

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

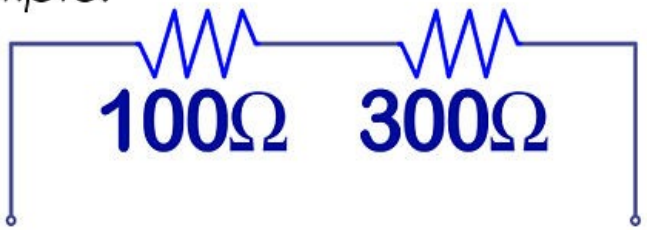


Resistenze in serie

- Due o più **resistenze** messe in **serie** formano una resistenza equivalente alla **somma** dei componenti

$$R_{eq} = R_1 + R_2 + \dots R_n$$

Example:

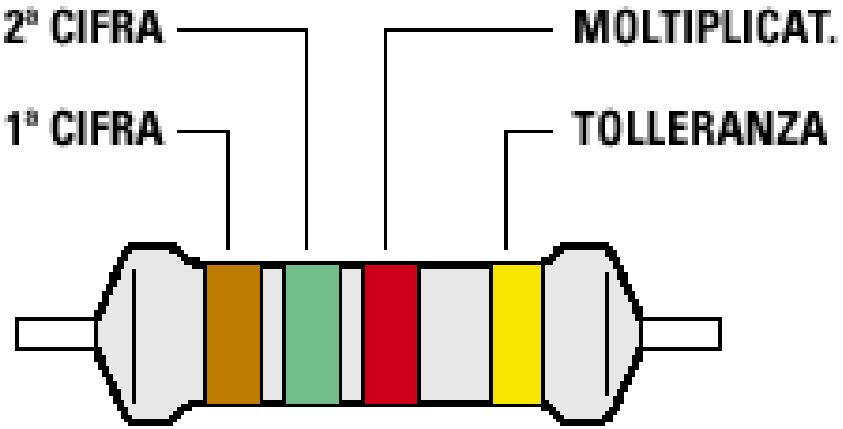


$$= 100\Omega + 300\Omega$$

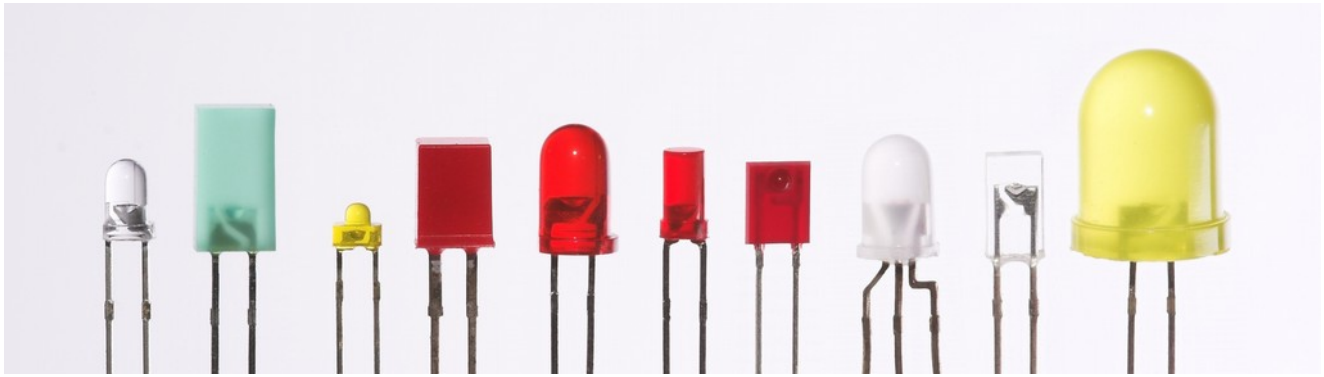
$$= \boxed{400\Omega}$$

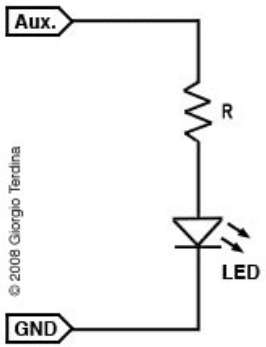
Schema colori resistenze

	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		



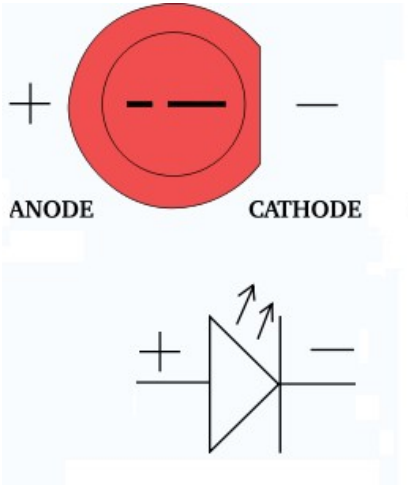
GESTIRE I LED

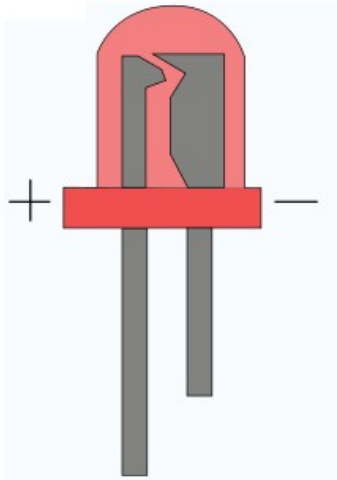




GESTIRE I LED

- **LED e resistenze**
  - Resistenze in serie
  - Ottenere il voltaggio desiderato
- **LED e Arduino**
  - Breadboard e GPIO
  - Pin digitali
  - digitalWrite()
  - Led blink

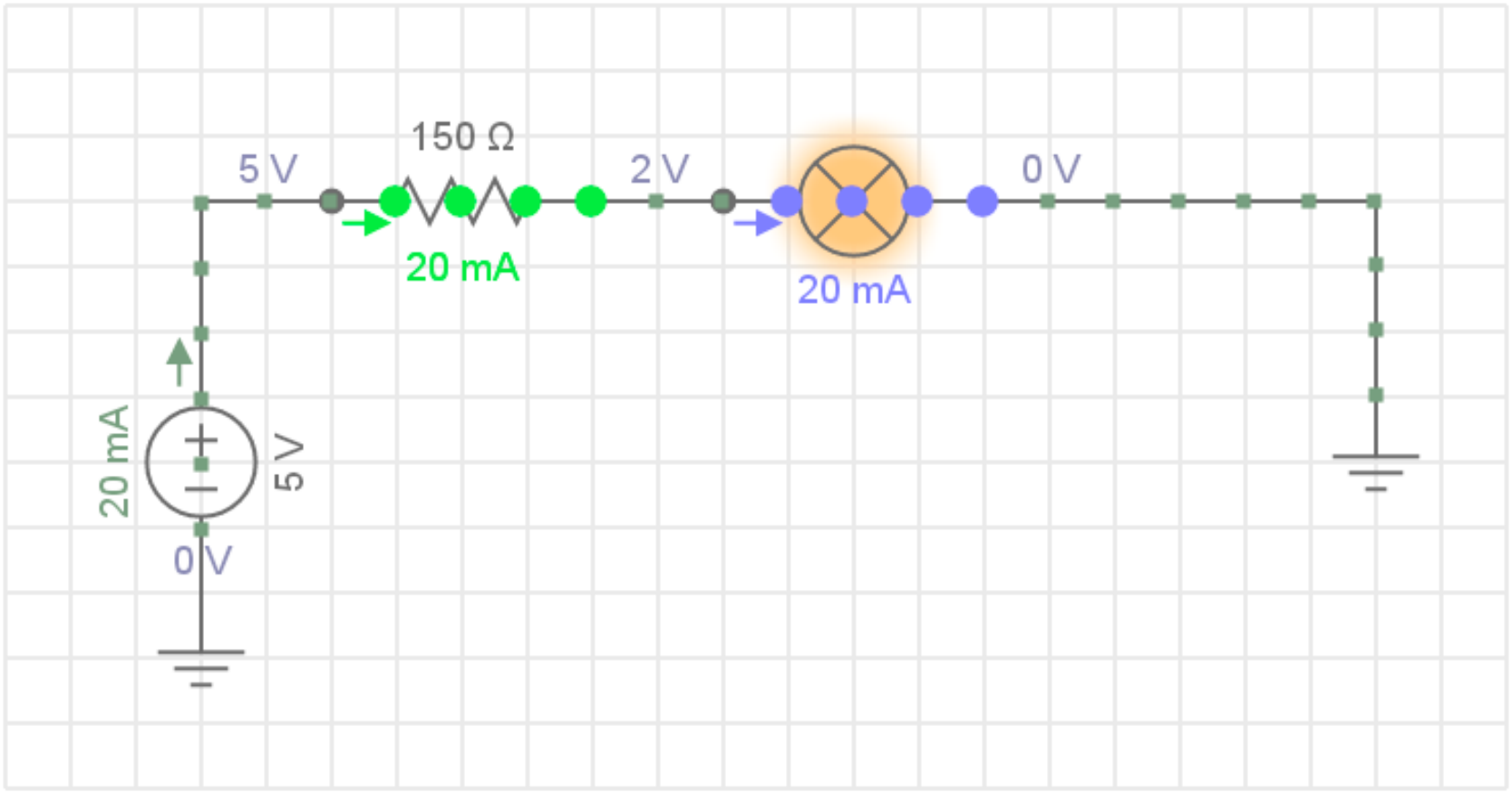




Parametri per i LED

- **Voltaggi operativi medi**
  - colore rosso: 1,8 V
  - colore giallo: 1,9 V
  - colore verde: 2,0 V
  - colore arancio: 2,0 V
  - colore blu: 3,0 V
  - colore bianco: 3,0 V
- **Corrente di esercizio**
  - 20 mA

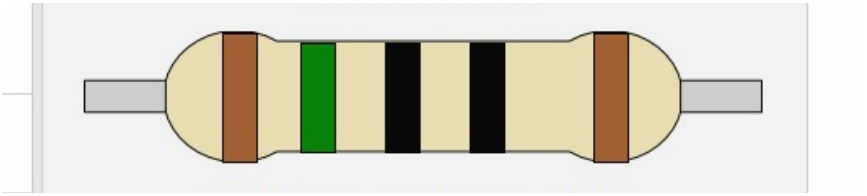
LED e resistenza in serie



Esempio: Con una tensione di 5V e un led che opera a 20mA,
una resistenza da 150 Ohm abbassa la tensione ai capi del led a 2V.

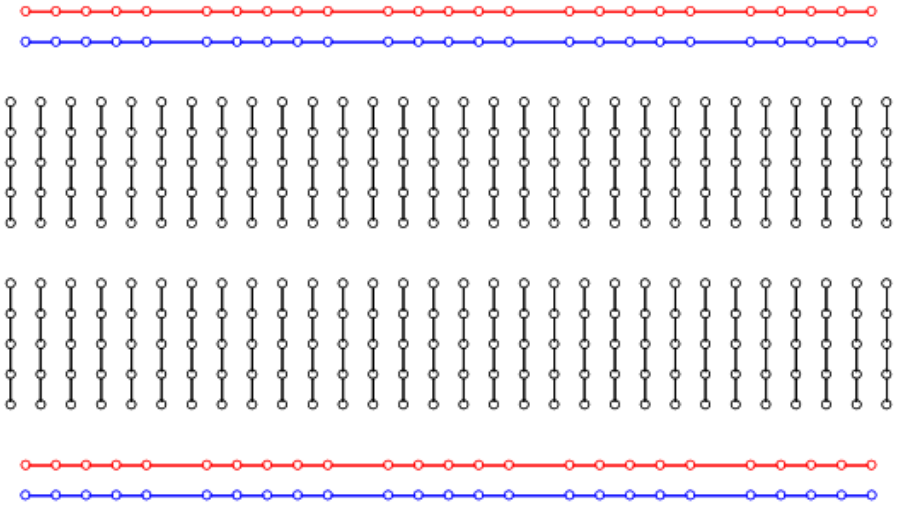
Resistenze consigliate x Kit

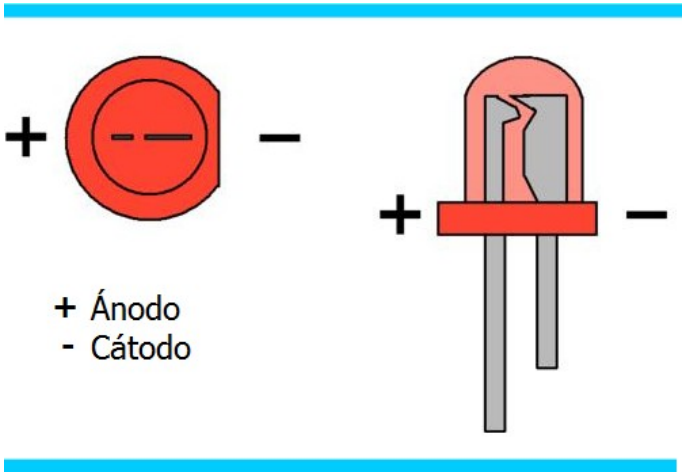
- **Rossi, Gialli**
  - 150 Ohm
  - Ma anche 220, 180...
- **Bianchi, Blu, Verdi**
  - 100 Ohm
  - Ma anche 150, 120 ...



Cablaggio

- **Breadboard e LED**

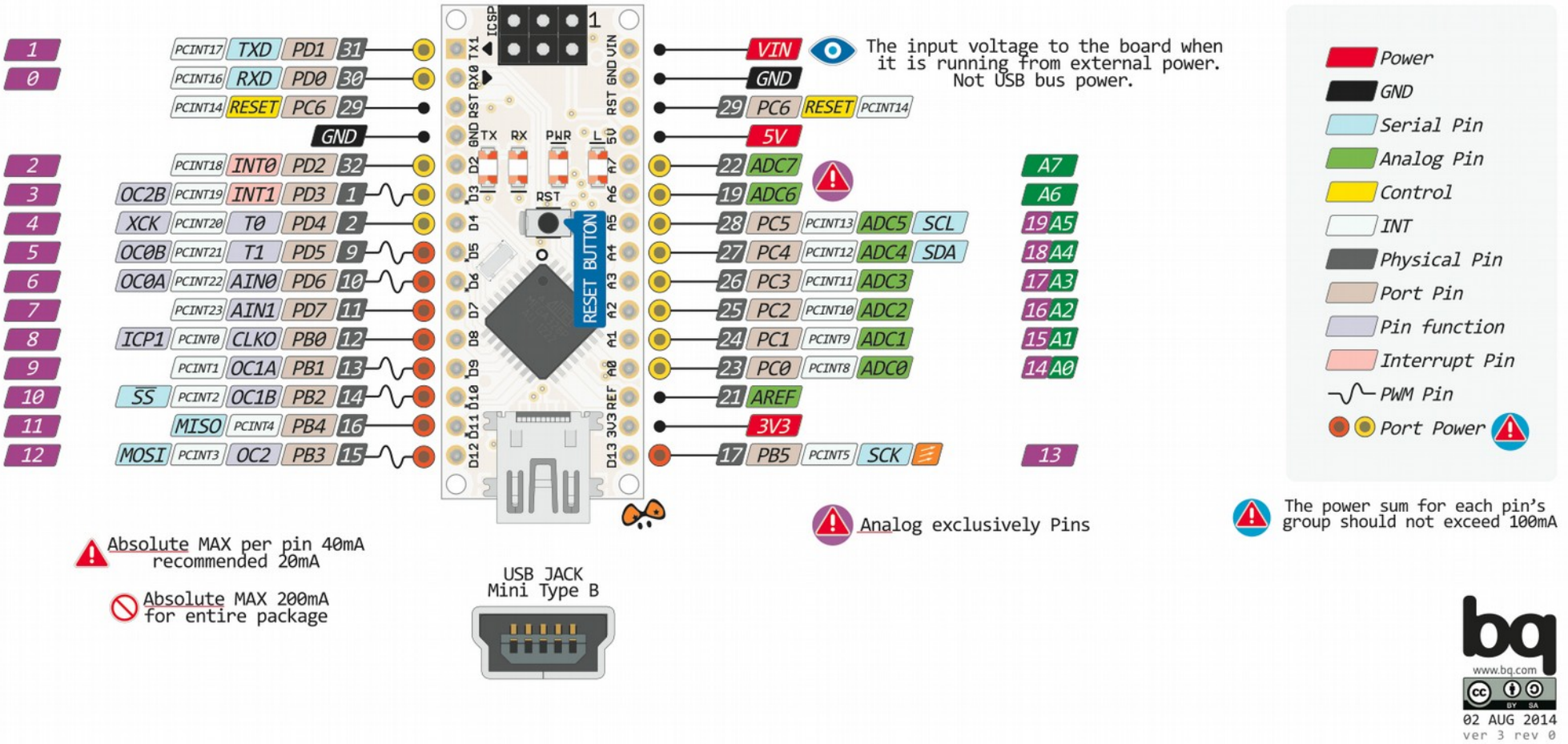




- Colonne numerate
- Colonne unite, come fossero saldate
- Righe rosse e blu unite (per + e -)

- Scegliere una resistenza adeguata
- Controllare Anodo e Catodo
- Collegare in seria alla resistenza

Arduino Nano



bq
www.bq.com

02 AUG 2014
ver 3 rev 0

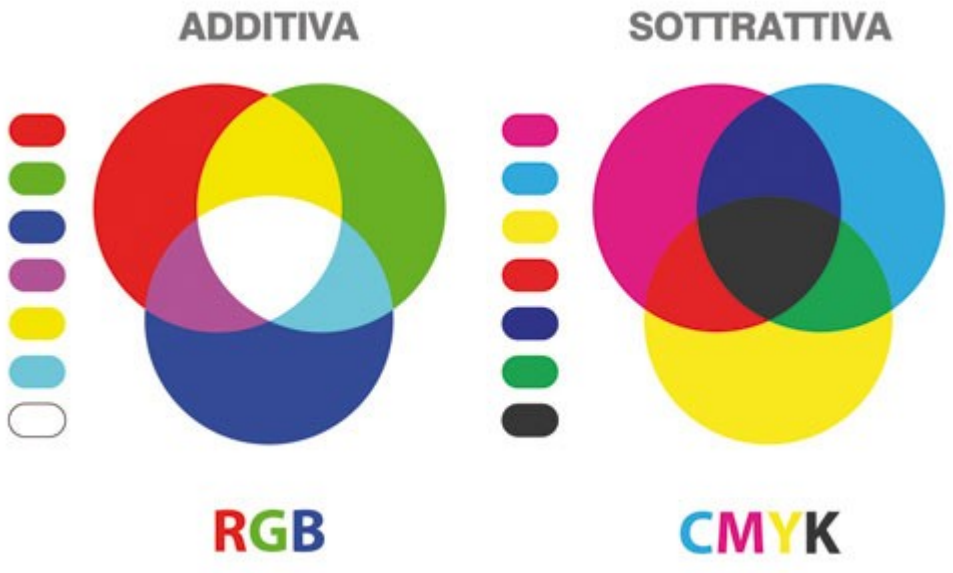
Led Blink

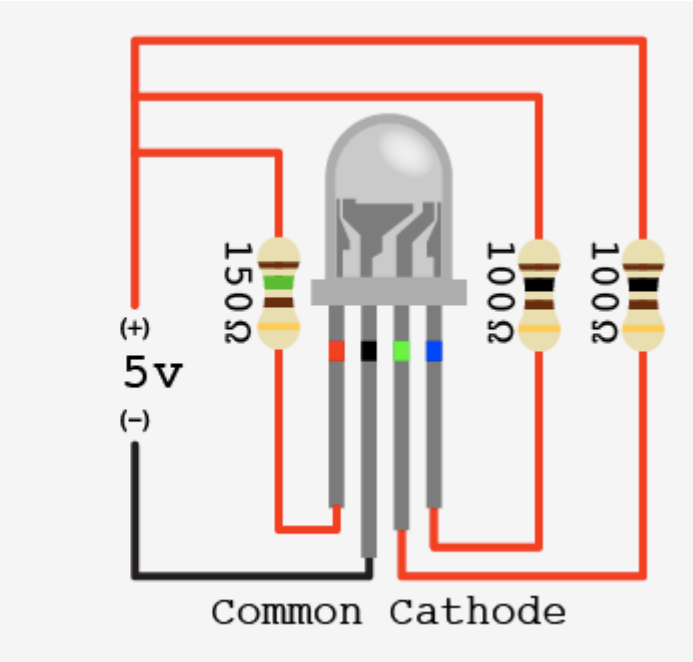
Blink §

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

GESTIRE LED RGB

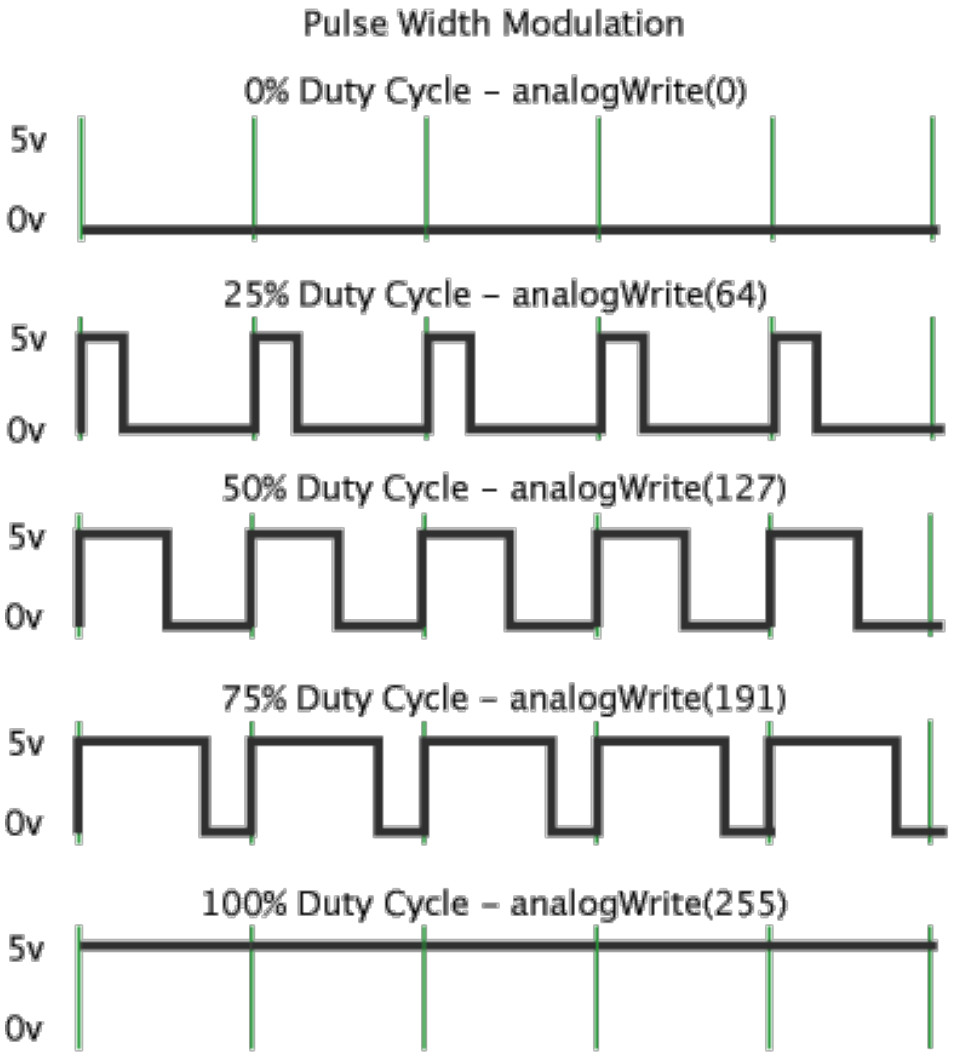
- **Composizione del colore RGB**
  - Sintesi additiva e sottrattiva
- **LED RGB e Arduino**





Ouput analogico: PWM

- **analogWrite()**
  - Modula un segnale
  - Da 0 a 255
- **Pin PWM**
  - Solo alcuni PIN
  - Es: Nano
    - Digital PWM
      - D3
      - D5, D6
      - D9, D10, D11



RGB Led

Blink §

```
// the setup function runs once when you press reset or power the board
#define RED 9
#define GREEN 10
#define BLUE 11

void setup() {
  // initialize digital pin 13 as an output.
  pinMode(RED, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
  analogWrite(RED, 100);
  analogWrite(GREEN, 200);
  analogWrite(BLUE, 50);
  delay(1000);
  analogWrite(RED, 0);
  analogWrite(GREEN, 0);
  analogWrite(BLUE, 0);
  delay(1000);
}
```

Valori casuali

random()

La funzione random genera numeri pseudo-casuali

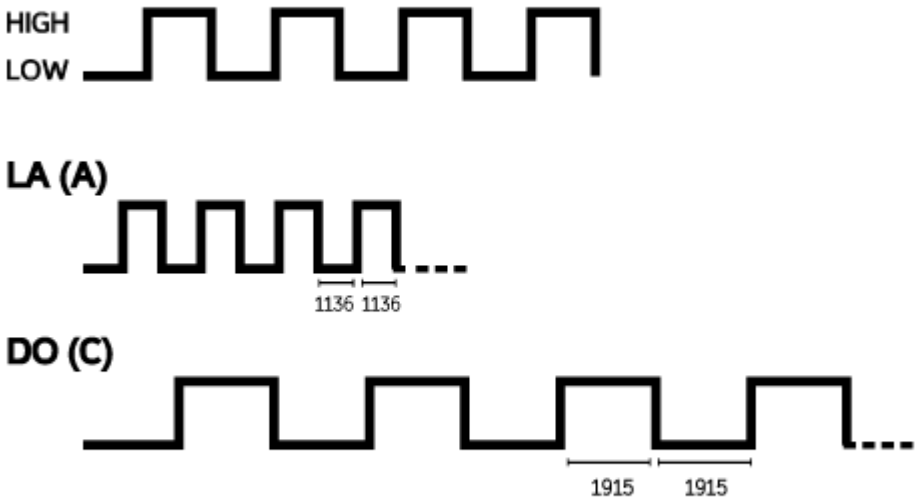
Sintassi

random(max)

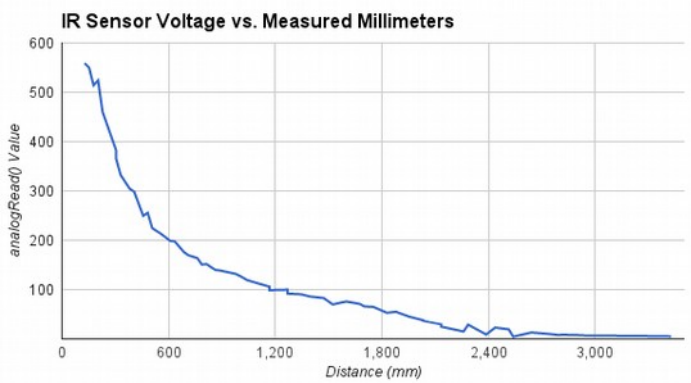
random(min, max)

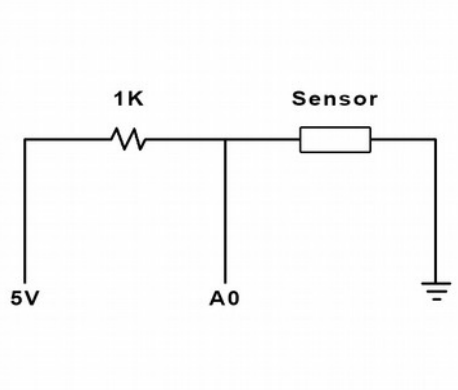
GESTIRE UN BUZZER

- **modulare suoni con Arduino**
  - PIN
    - Qualsiasi PIN digitale
    - ***WARNING***: usa interrupts
      - Potrebbe influire su PWM e librerie che usano timer
  - tone(pin, freq, duration)
- **mappatura delle note**
  - Frequenza in Hz
- **produrre una melodia**
  - tone e delay



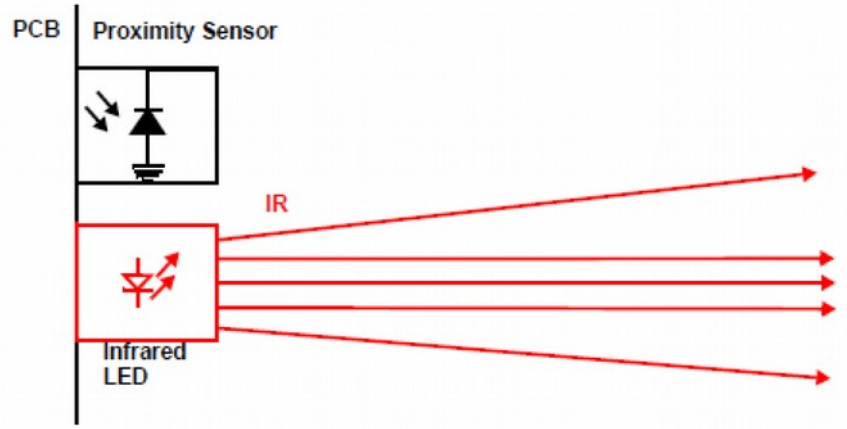
USARE I SENSORI

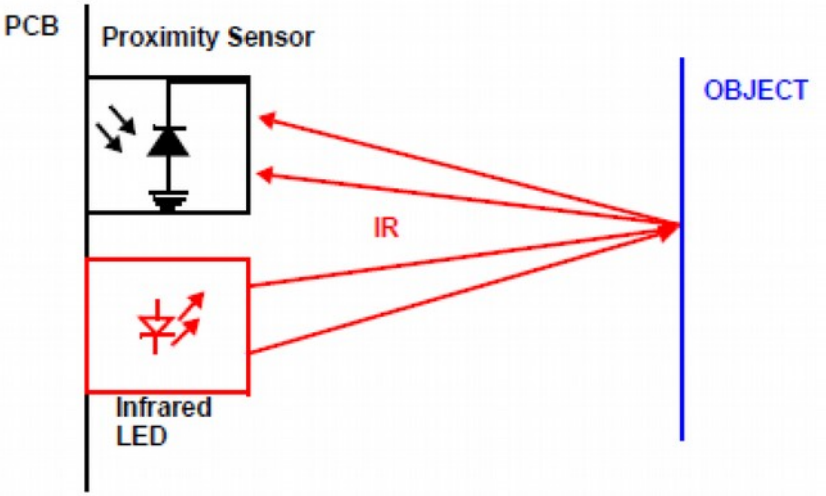




IR PROXIMITY SENSOR

- **Struttura del sensore**
  - Emettitore IR
  - Ricevitore IR
  - Principio di funzionamento





digitalRead

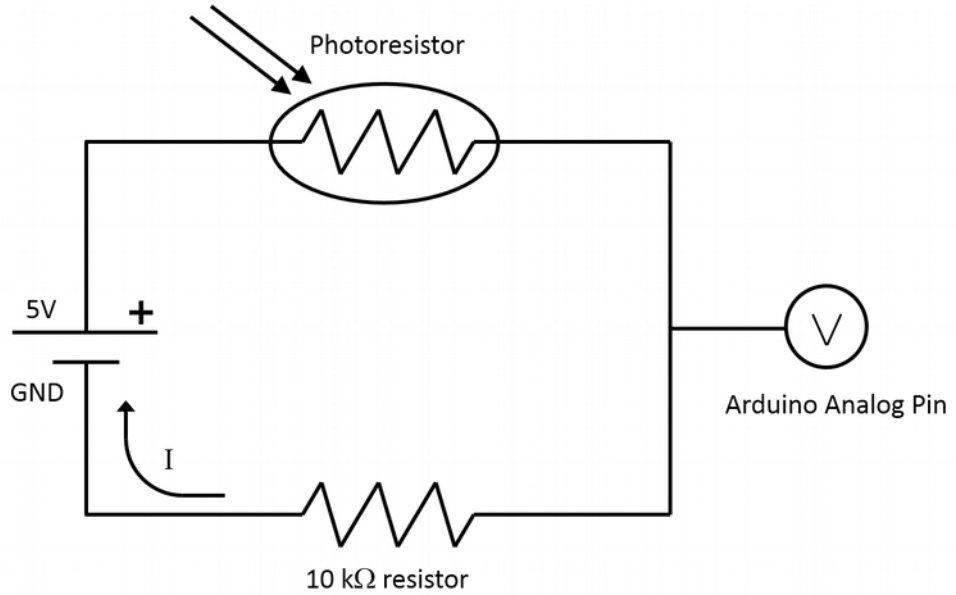
- **Gestire un sensore IR da Arduino**
  - Sensori digitali con DO
    - Digital Output
    - Segnale digitale (binario) in base a soglia
- **Leggere input digitale**
  - digitalRead()
  - una funzione che restituisce il valore 1/0 letto da un sensore
  - Come parametro vuole il PIN a cui è collegato il sensore
  - ES:

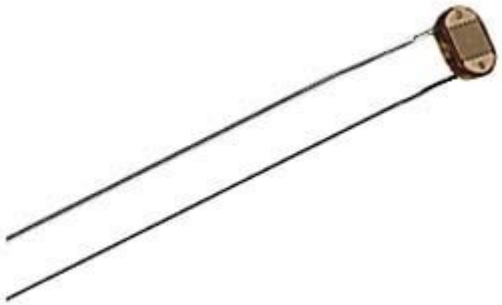
```
bool input=digitalRead(9);
```

LIGHT SENSOR

Fotoresistenza

- È una resistenza che varia in base alla luce che la investe





analogRead

- **Leggere input analogico**
  - anlogRead()
  - una funzione che restituisce il valore 0~1023 letto da un sensore
  - Come parametro vuole il PIN a cui è collegato il sensore
  - ES:

```
int val=analogRead(A1);
```

Riscalare valori

- **La funzione map**

map(value, fromLow, fromHigh, toLow, toHigh)

- **Scalare valori da un intervallo a un altro**
- **Esempio:**

```
int val = analogRead(A1);
val = map(val, 0, 1023, 0, 255);
analogWrite(9, val);
```

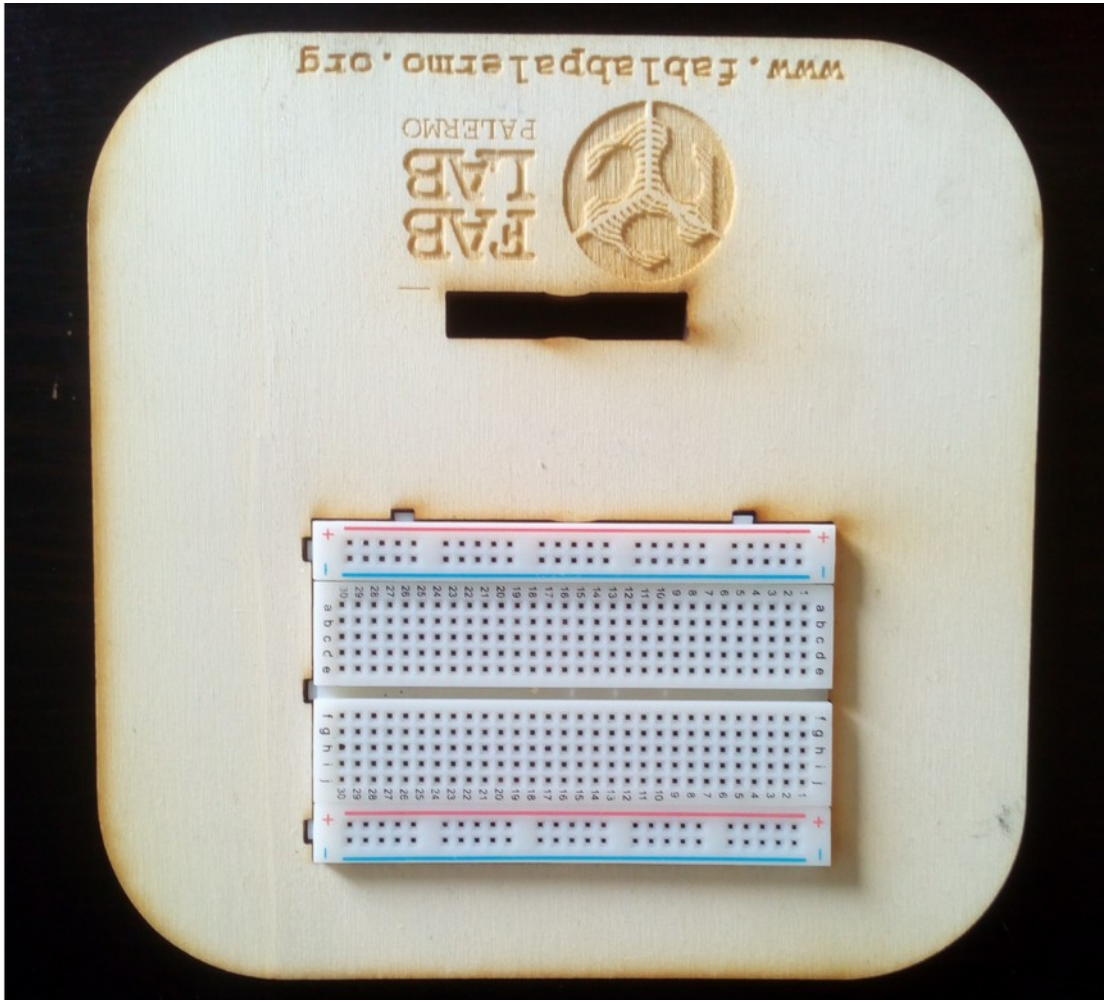
COSTRUZIONE DEL PROGETTO

Fasi del progetto

- **Preparazione**
  - Base, Nano e Breadboard
  - Resistenze e ponticelli
- **Cablaggio e montaggio**
  - LEDs
  - Sensore LIGHT
  - Sensore IR
  - Speaker

Breadboard

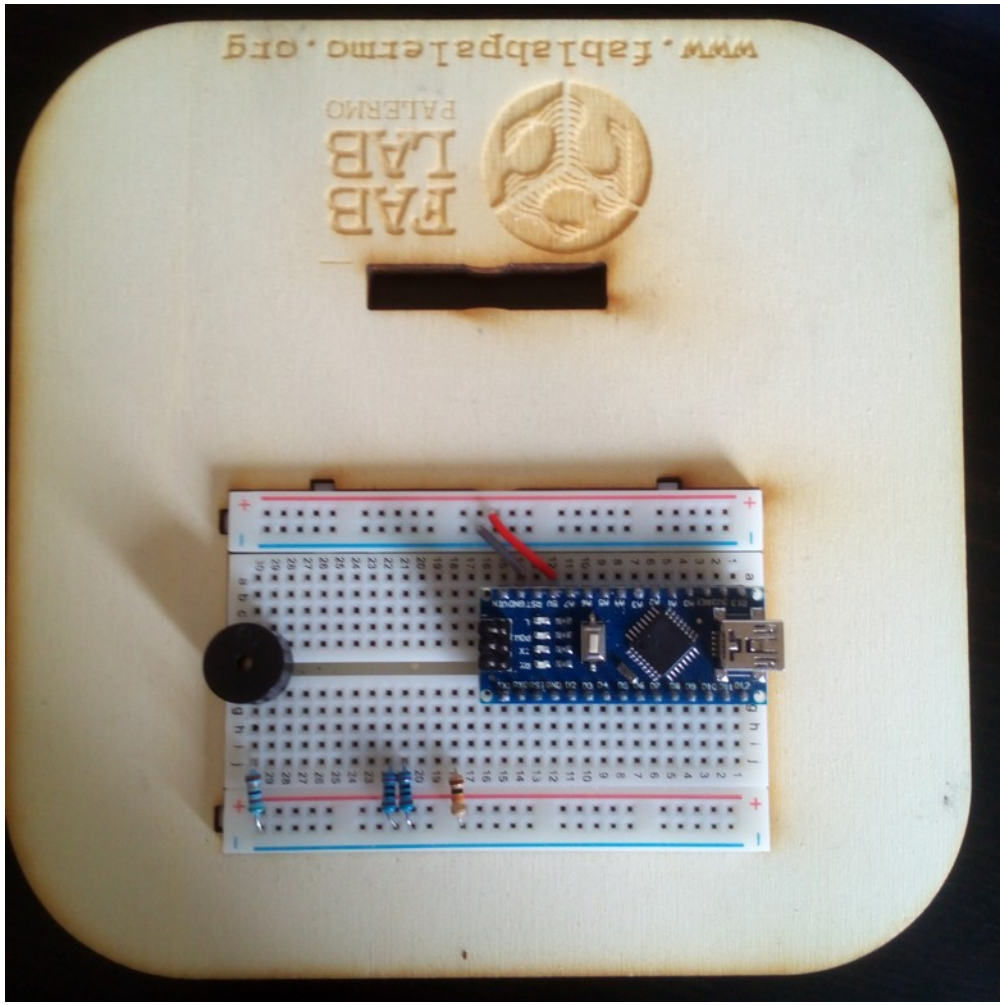
- **Base in legno**
- **Breadboard**



Nano

- **Nano**
- **Resistenze**
  - 51ohm → GND
  - 120ohm → GND
  - 120ohm → GND
  - 10Kohm → GND
- **Ponticelli**
  - VCC → +
  - GND → -
- **Speaker**

  → 51ohm

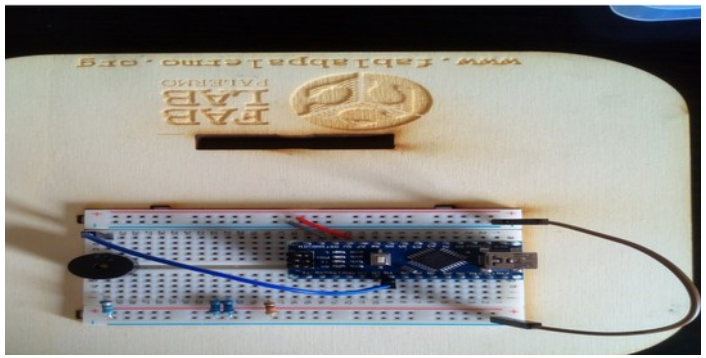


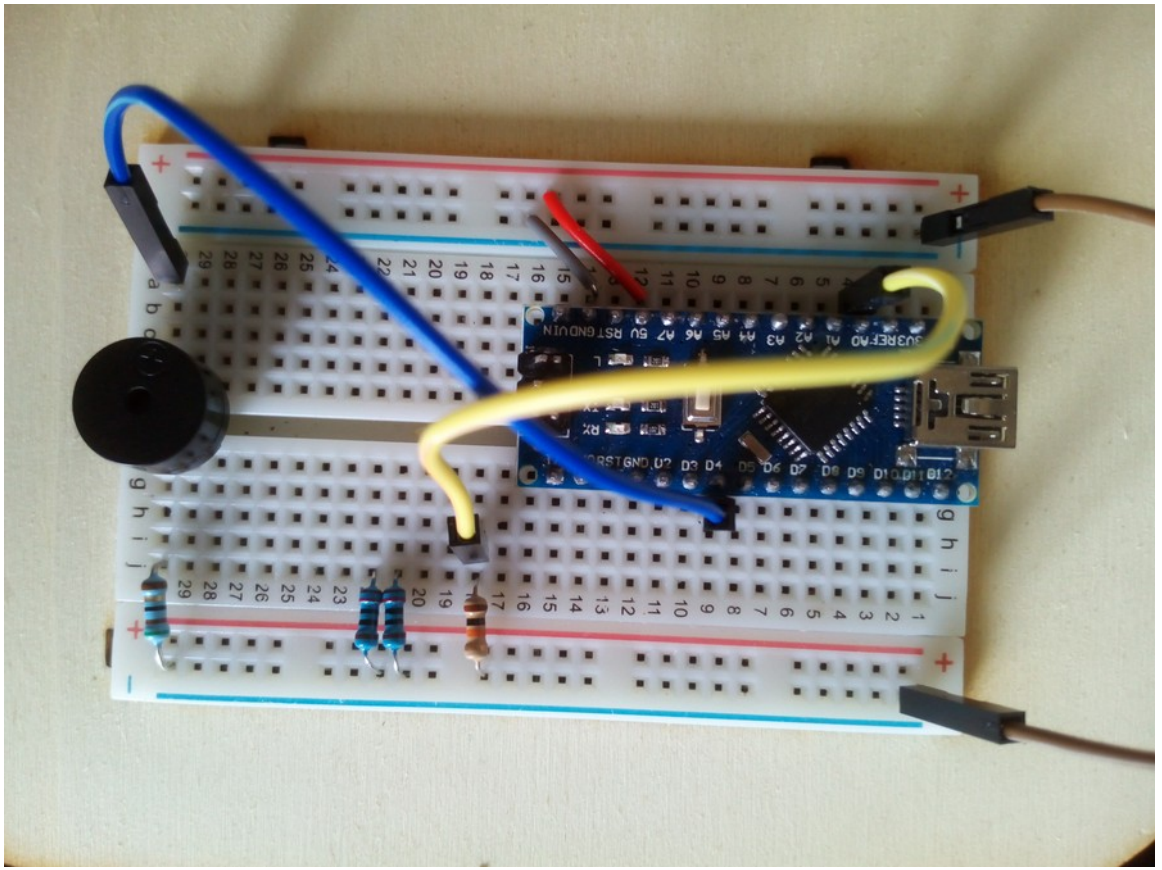
Primi componenti

- - Ponte Ground
- - Speaker

  Pin D4 51ohm
- - Light

  Pin A0 10Kohm

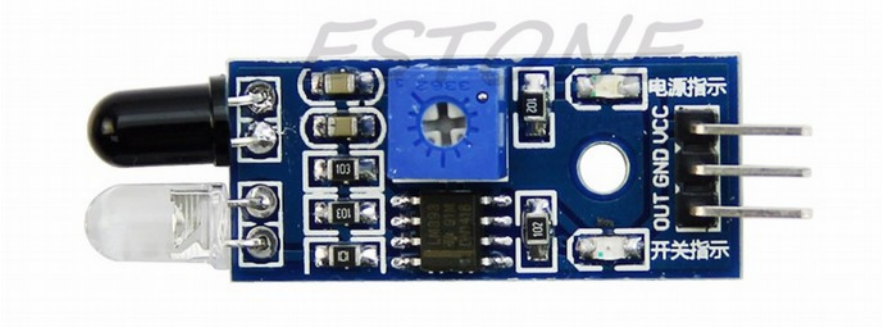


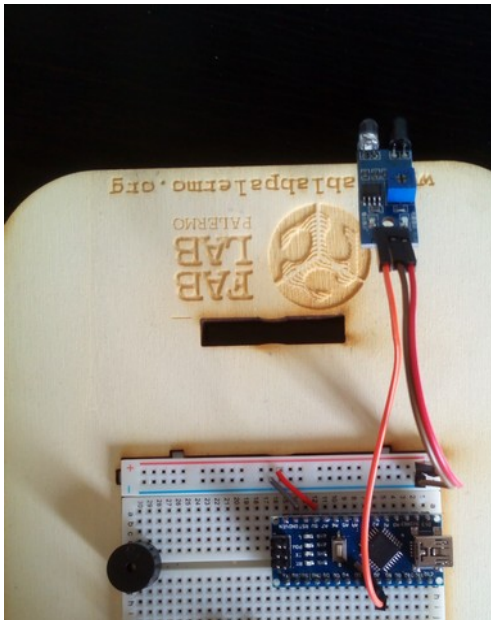


Sensore infrarosso

Modulo IR

- **- Vcc (5V)**
- **- GND**
- **- OUT (al PIN)**

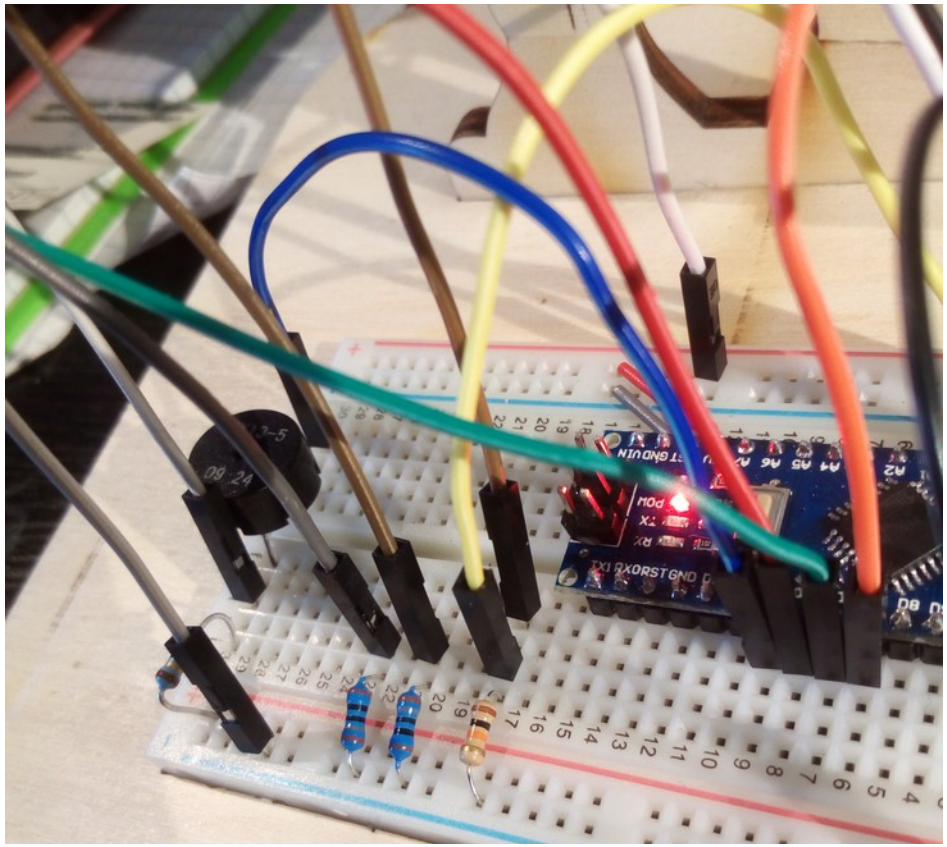


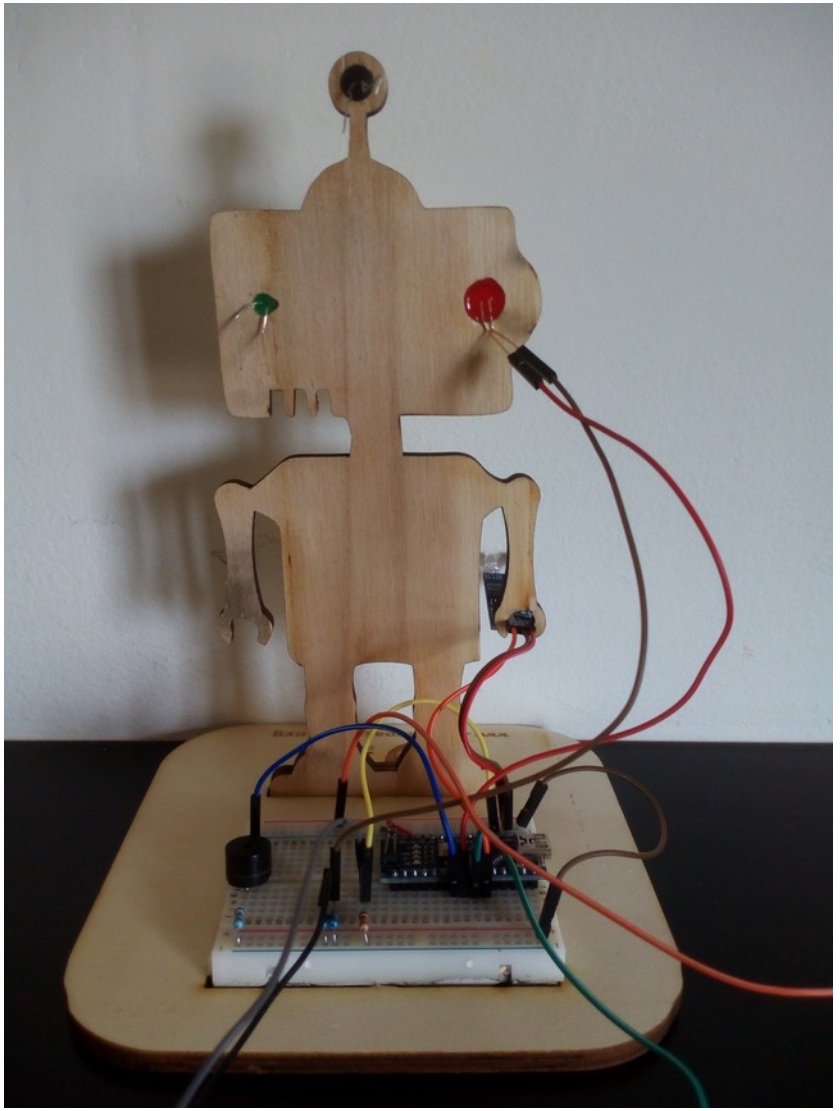


BigLed

Led grosso:

Pin D5 (+) 120ohm (-)

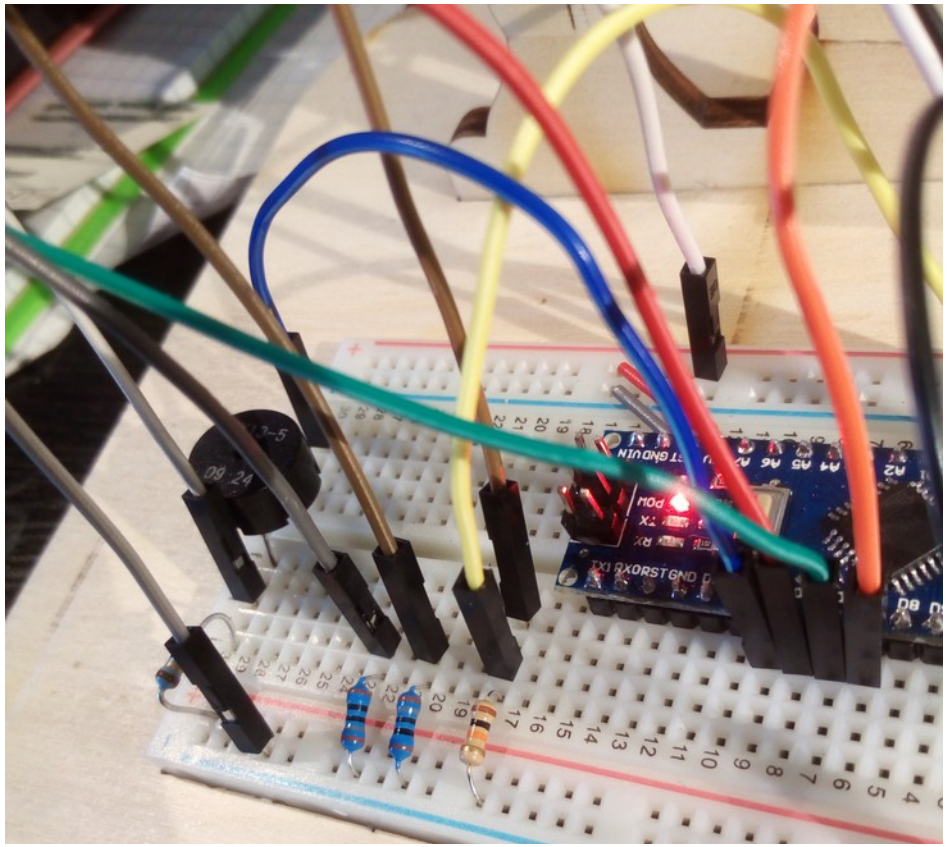


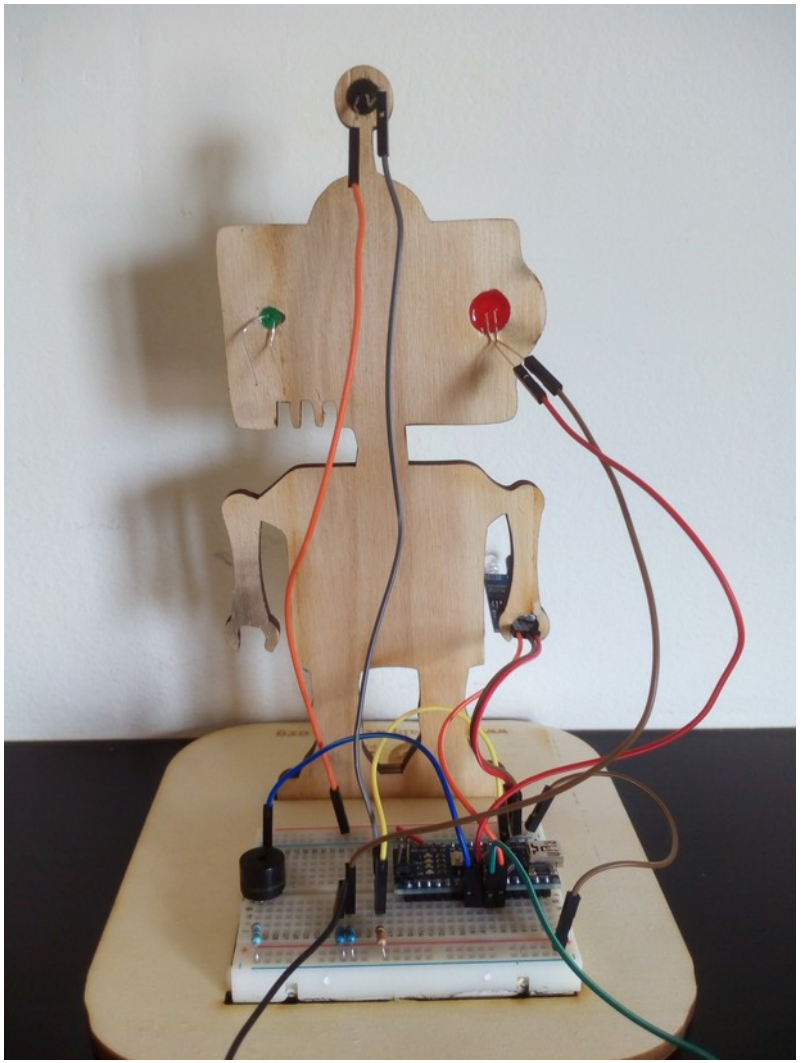


SmallLed

Led piccolo:

Pin D6 (+) 120ohm (-)

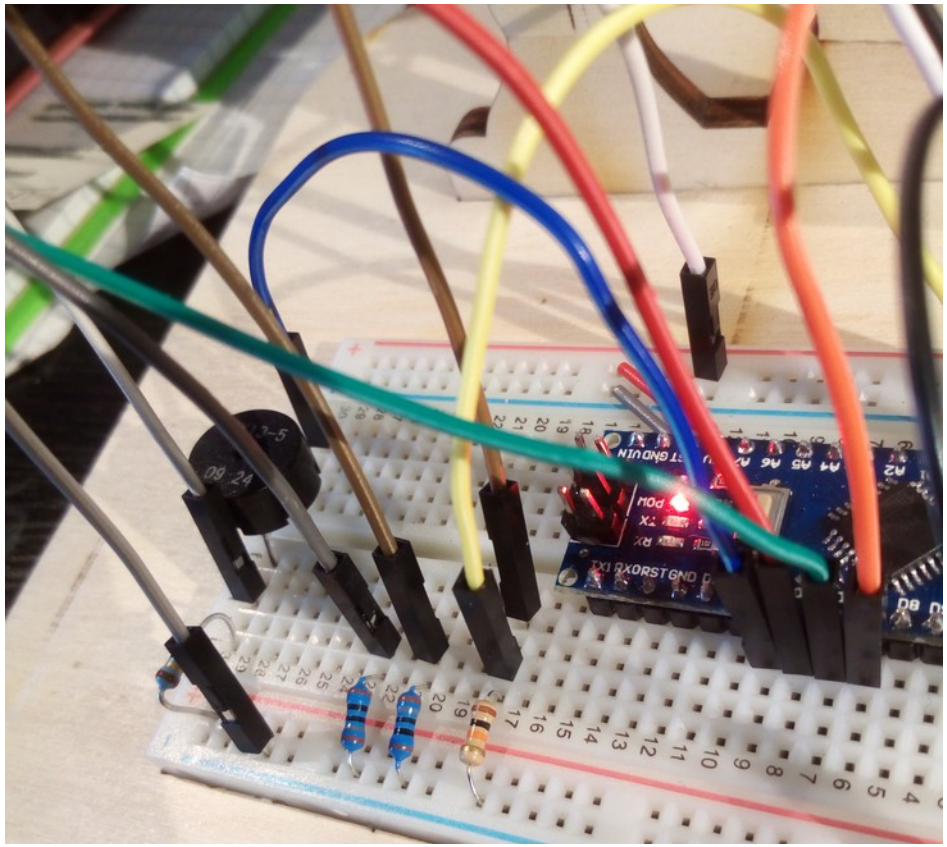


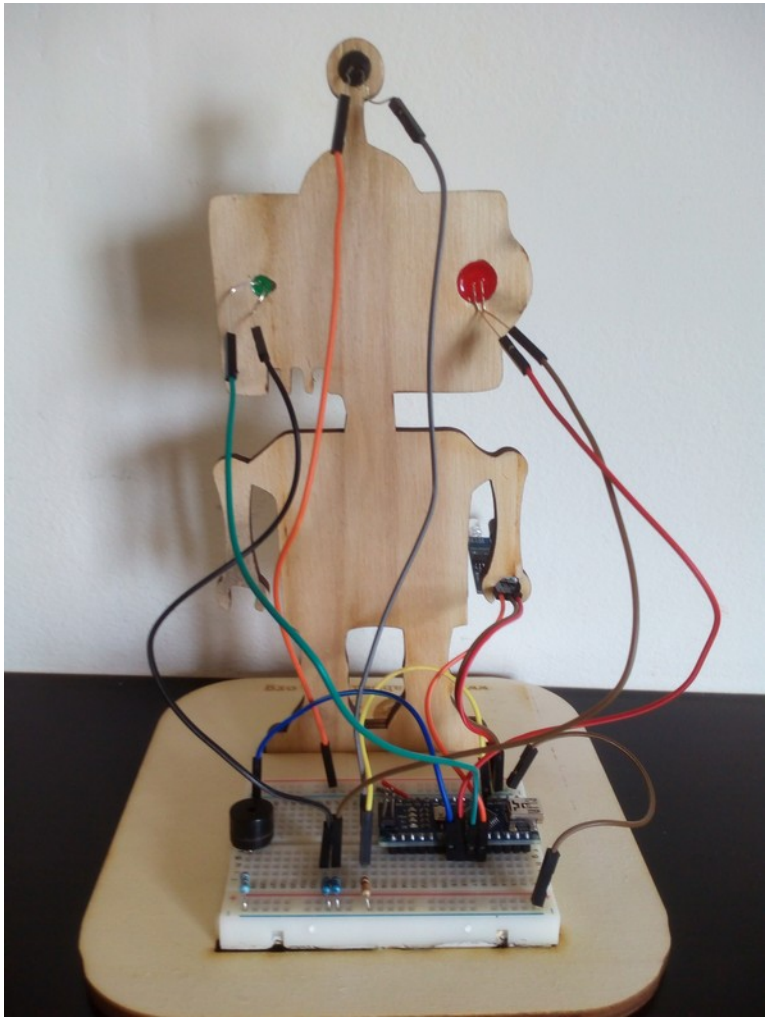


Light

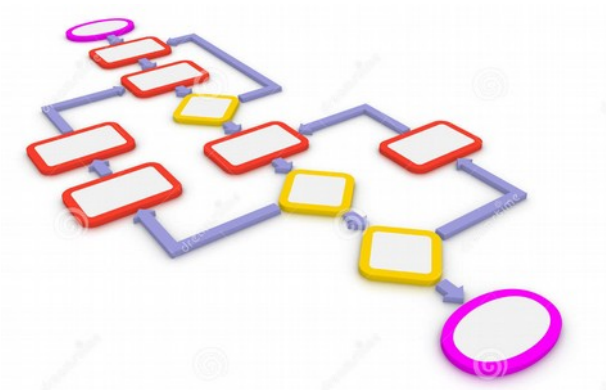
Fotoresistenza:

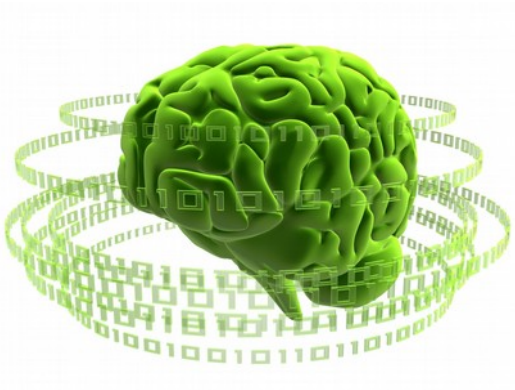
- VCC (~)
- ponte 10Kohm (~)

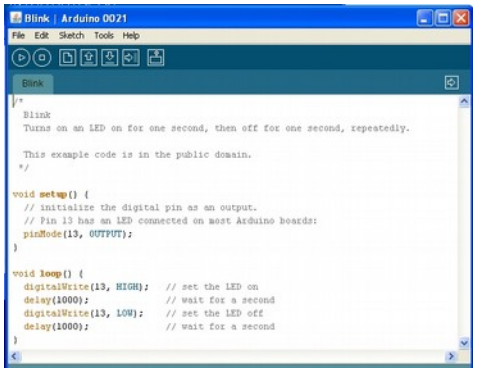




LO SKETCH






Blink | Arduino 0021
File Edit Sketch Tools Help
Blink
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
This example code is in the public domain.
*/
void setup() {
// initialize the digital pin as an output.
// Pin 13 has an LED connected on most Arduino boards:
pinMode(13, OUTPUT);
}
void loop() {
digitalWrite(13, HIGH); // set the LED on
delay(1000); // wait for a second
digitalWrite(13, LOW); // set the LED off
delay(1000); // wait for a second
}

Il comportamento

- **Comportamento parametrico** in base a caratteristiche personalizzabili
  - ENERGY
  - VIGILANCY
  - CRAZYNESS
  - SENSITIVITY
  - VOICETONE

Il comportamento

- **Funzionalità Theremino**

  Quando in allerta, Robruttino funziona da Theremino

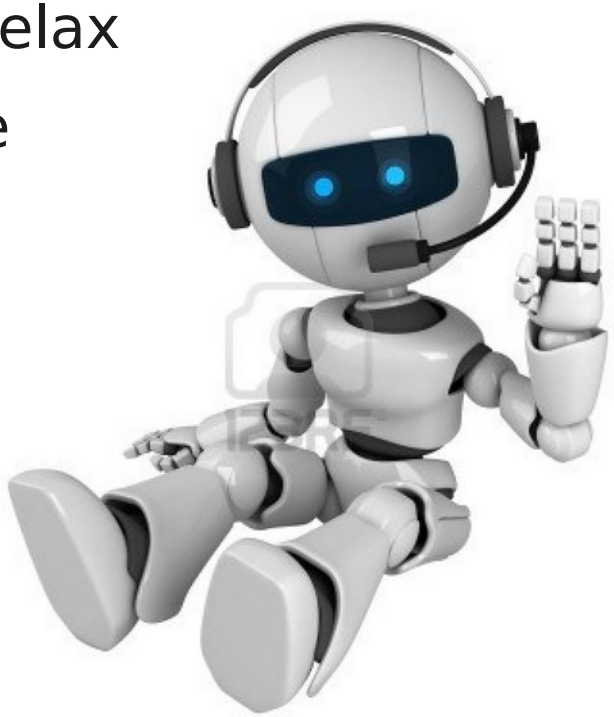
  - Il Light modula il tono
  - L'IR attiva e disattiva il suono

Il comportamento

- **Modalità Relax**

  Quando non disturbato per un po'
  Robruttino entra in modalità Relax

  - Reagisce alle variazioni di luce modulando un suono
  - Se disturbato rientra in modalità vigile, attivandosi solo con l'infrarosso



Iniziamo lo sketch

Configurazioni

i Pin utilizzati e le caratteristiche

```
/* Robruttino Theremino
 * --------------------
 * FabLab Palermo 2016
 * --------------------
 * Firmware version 0.1
 */

// Configurazione PIN
#define LIGHT  A0     // resistenza 10K
#define SPK    4      // resistenza 51ohm
#define BIGLED 5      // resistenza 120ohm
#define LED    6      // resistenza 120ohm
#define IR     7

// Caratteristiche
#define ENERGY      10  // Controlla la luminosità dei led e le pause fra i toni
#define VIGILANCY   2  // Controlla il tempo di relax e l'attività
#define CRAZYNESS   5  // Controlla la velocità dei toni e i comportamenti casuali
#define SENSITIVITY 5  // Controlla la sensibilità alle variazioni di luce
#define VOICETONE   5  // Controlla il tono (bassa-alta frequenza) dei buzz

```

Variabili globali

Variabili di lavoro

Dai parametri iniziali ricaviamo delle variabili su cui lavorare utilizzando map per scalarli

```
22 // Parametri derivati
23 int maxLedPower = map(ENERGY, 0, 10, 0, 100);
24 int relaxTime = map(VIGILANCY, 0, 10, 0, 200);
25 int interval = map(CRAZYNESS, 10, 0, 0, 100);
26 int toneDuration = ENERGY * interval / 10;
27 int tollerance = map(SENSITIVITY, 10, 0, 0, 100);
28 int maxTone = map(VOICETONE, 0, 10, 0, 5000);
29
30 // Valori variabili
31 int prevLight = 0;
32 int relax = 0;
```

Il setup: Robruttino si avvia

Setup: inizializziamo i componenti dichiariamo i pin, accendiamo i led, emettiamo due toni dipendenti dalle caratteristiche

```
34 void setup() {
35
36   pinMode(SPK, OUTPUT);
37   pinMode(BIGLED, OUTPUT);
38   pinMode(LED, OUTPUT);
39   pinMode(LIGHT, INPUT);
40   pinMode(IR, INPUT);
41
42   analogWrite(BIGLED, maxLedPower );
43   analogWrite(LED, maxLedPower );
44
45   tone(SPK, maxTone/CRAZYNESS, interval*ENERGY);
46   delay(interval*ENERGY);
47   tone(SPK, maxTone/SENSITIVITY, interval*VIGILANCY);
48   delay(interval*VIGILANCY);
49
50 }
```

Leggiamo i dati e reagiamo all'IR

Loop: variabili di stato e gestione dell'IR

```
52 void loop() {
53
54   int obstacle = !digitalRead(IR);
55   int currentLight = analogRead(LIGHT);
56   int ledPower = map(currentLight, 0, 1023, 0, maxLedPower);
57   int buzz = map(currentLight,0,1023, 0, maxTone);
58   int diff = abs(prevLight - currentLight);
59
60   // sotto una certa soglia di luce il led si spegne
61   if(currentLight <= SENSITIVITY * 10)
62     ledPower = 0;
63
64   // Il sensore IR viene chiuso
65   if (obstacle) {
66     tone(SPK,buzz, toneDuration);
67     analogWrite(BIGLED, ledPower);
68     relax=0;
69   } else {
70     relax++;
71     relax=constrain(relax,0,1000);
72   }
```

Un comportamento in standby

RELAX MODE: led random, blink e tono

```
73
74  // RELAX MODE: il relaxTime è trascorso senza disturbi
75  if(relax > relaxTime){
76
77    // Il led piccolo lampeggia in funzione di CRAZYNESS
78    int smallEye = random(0,ledPower);
79    if(smallEye % CRAZYNESS){
80        smallEye = 0;
81    }
82    analogWrite(LED, smallEye);
83
84    // Il led grande si accende in funzione di VIGILANCY
85    int bigEye = 0;
86    if(random(0,100) < VIGILANCY){
87      bigEye = ledPower;
88      tone(SPK,50, toneDuration);
89    }
90    analogWrite(BIGLED, bigEye);
91
92  }
```

Reazione alla luce

Reazione alle variazioni di luminosità

```
94    // Il sensore LIGHT rileva una differenza sensibile
95    if (diff > tollerance){
96     analogWrite(BIGLED, ledPower);
97     analogWrite(LED, ledPower);
98     prevLight = currentLight;
99     relax -= VIGILANCY;
100    if(relax>relaxTime){
101     tone(SPK,buzz, toneDuration);
102    }
103   }
104
105   delay(interval);
106 }
```

FINE

APPROFONDIMENTI

- **Spazio per**
  - Discussioni
  - Domande e risposte
  - Possibili evoluzioni del progetto
  - Ulteriori applicazioni
- **Fonti di approfondimento**
  - FabLab
    - Gruppo
    - Altri workshop
  - Playground
  - Tutorials, Howto, Instructables..

Stay in touch

- **www.fablabpalermo.org**
  - Sito ufficiale
- **Pagina facebook: FabLab Palermo**
  - Pubblica
  - News su attività ed eventi
- **Gruppo facebook: FabLab Palermo**
  - Attività riservate ai soci del FabLab

Customizzare lo sketch

- **HACK IT: Migliorate lo sketch**
  - Provate a cambiare le caratteristiche (CRAZYNESS,...)
  - Aggiungete funzionalità e componenti (RGB, DHT, etc...)
  - Personalizzate suoni e giochi di LED
- **SHARE IT: Postatelo sul gruppo**
  - Condividetelo con gli altri Makers
- **TRY IT: Provate gli sketch degli altri makers**
  - Caricate sul vostro Robruttino gli sketch dei vostri amici Makers

Grazie e arrivederci

FabLab Palermo