

Workshop Arduino Natale 2015



**FAB
LAB**
PALERMO

19/12/2015
FabLab Palermo

Scopo

- **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**
 - Alberello natalizio con Led e musica

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



Il contesto

- **Physical Computing**

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

- **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**.

- **Movimento Makers**

- Il movimento dei maker unisce persone di diversa formazione che **condividono** l'interesse verso l'apprendimento di capacità tecniche e la loro applicazione creativa al fine di **fabbricare oggetti o inventare soluzioni** innovative

Arduino

- **Arduino**

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

- **Boards**

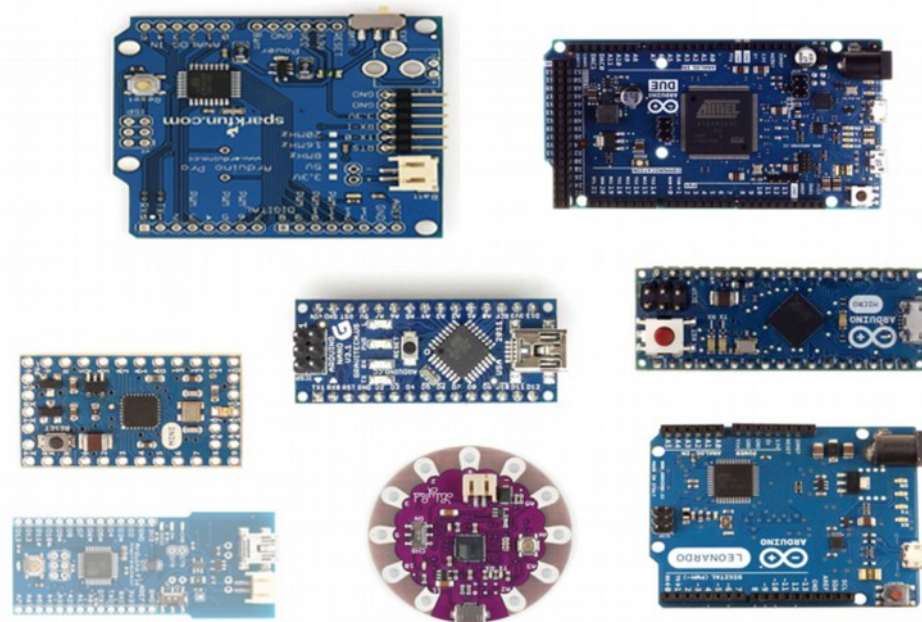
- Uno, Nano, Mega, etc..

- **Gli Sketch**

- Il concetto di applicazione

- **Il Playground**

- La community e la codebase arduino.cc



Il contesto

- **Physical Computing**

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

- **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**.

- **Movimento Makers**

- Il movimento dei maker unisce persone di diversa formazione che **condividono** l'interesse verso l'apprendimento di capacità tecniche e la loro applicazione creativa al fine di **fabbricare oggetti o inventare soluzioni** innovative

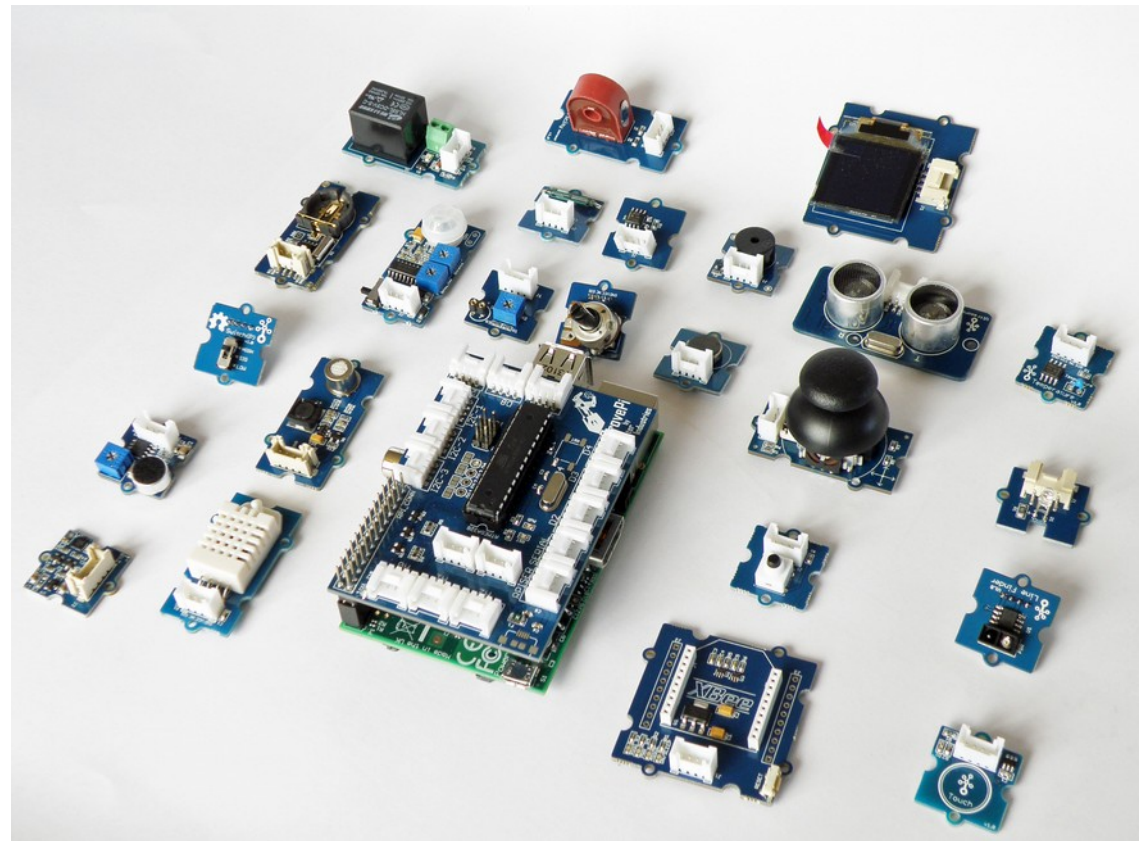
Attuatori

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



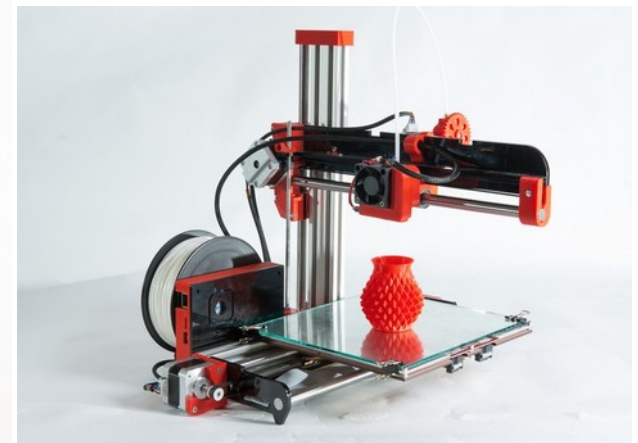
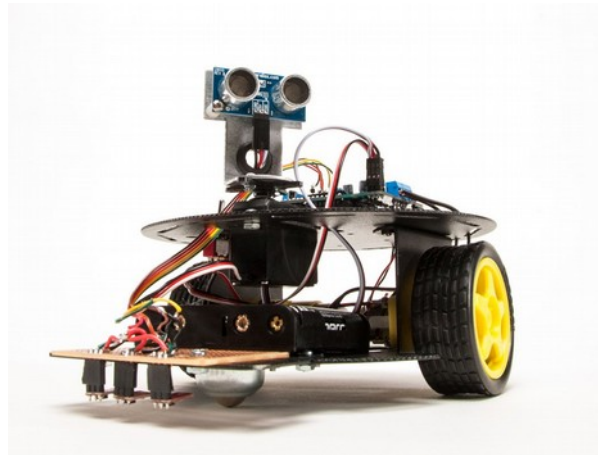
Sensori

- **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**



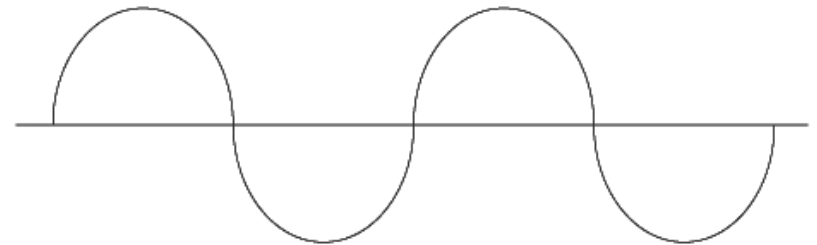
FONDAMENTI DI INFORMATICA



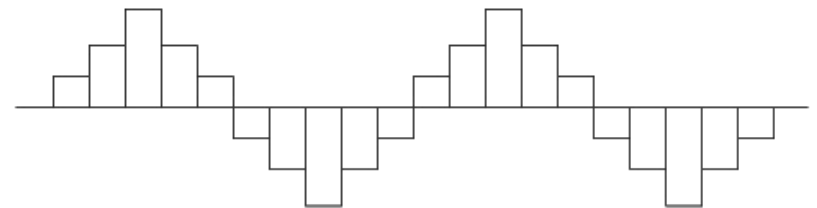
Segnali

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

Analog



Digital



Informazione digitale

- **Sistema binario**

- Sistema posizionale
- Base 2
- Ogni cifra è una potenza di 2
 - $2^0, 2^1, 2^2, 2^3, \dots$
 - 1, 2, 4, 8, 16, 32, 64, 128, 256, ...

- **Esempi**

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

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

So the following binary pattern would be

0	1	1	0	0	0	0	1
---	---	---	---	---	---	---	---

$$(1 \times 64) + (1 \times 32) + (1 \times 1) = 97$$

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

128	64	32	16	8	4	2	1
x	x	x	x	x	x	x	x
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

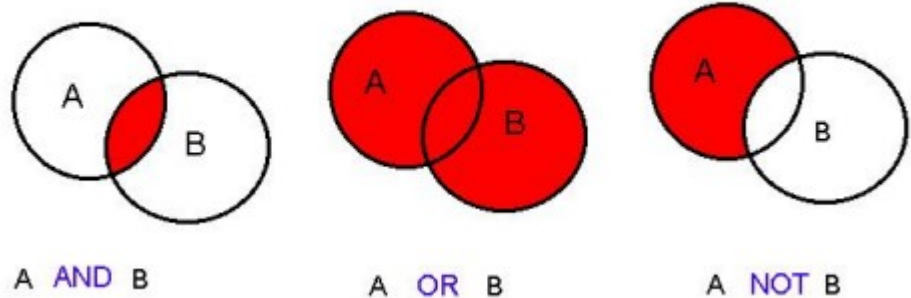
- **Valori booleani**

- Valori logici opposti
 - Vero / Falso
 - 1 / 0

- **Operatori logici**

- AND & &&
- OR | ||
- NOT ¬ !
- XOR ^ !=
- XNOR ≡ ==

BOOLEAN OPERATORS



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

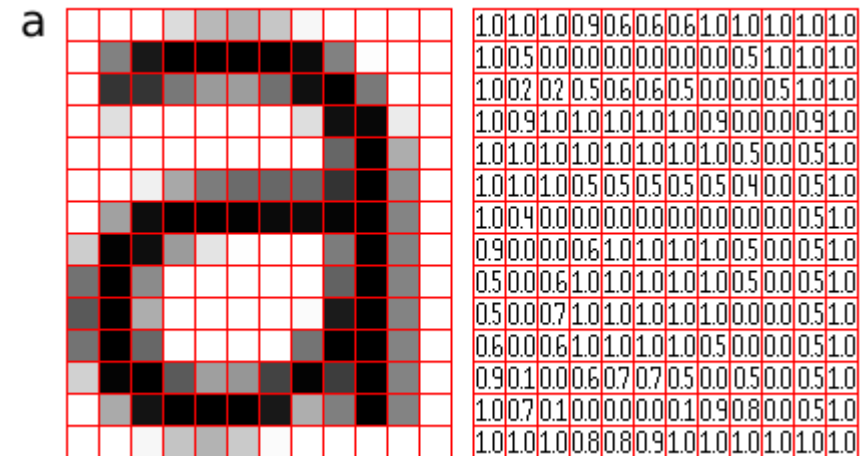
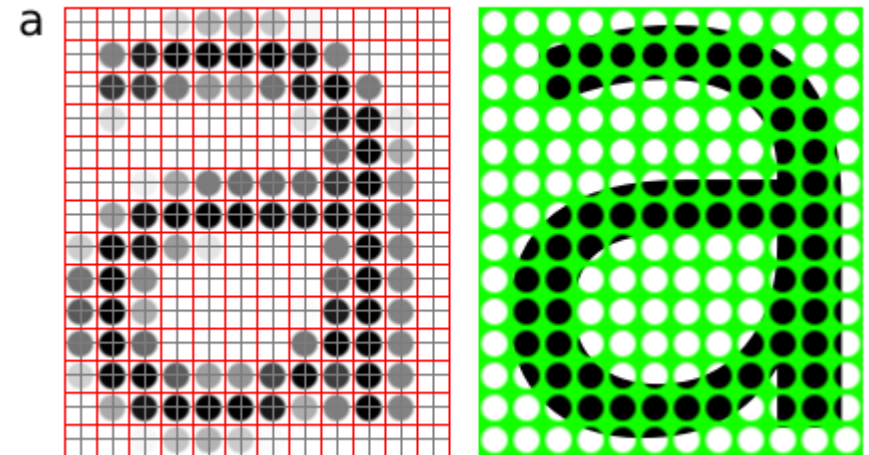
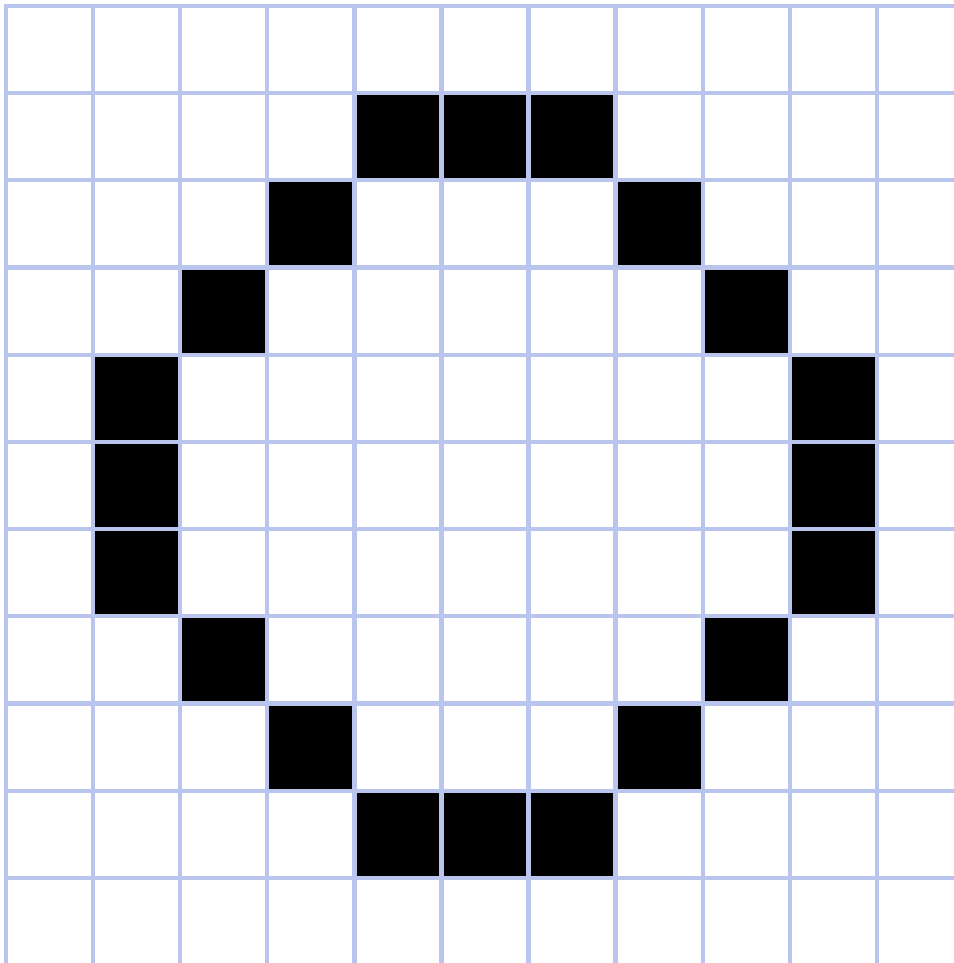
Informazione binaria

- **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
- **TeraByte (TB)**
- **PetaByte (PB)**
- **ExaByte (EB)**
- **YottaByte (YB)**

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



Architettura di base di un Microcontrollore

- **CPU e MCU**

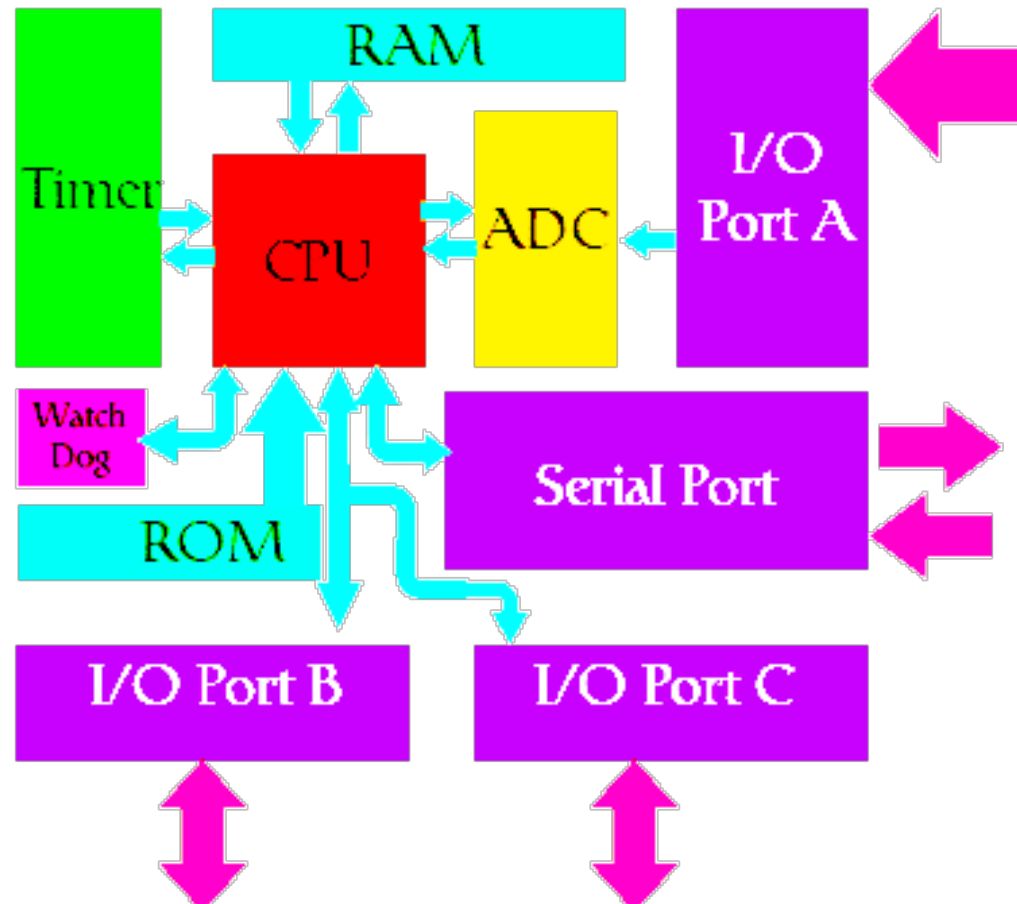
- Microprocessore
 - Solo calcolo
 - Richiede periferiche
- Microcontrollore
 - Sistema integrato
 - Input / output

- **Memorie volatili**

- RAM

- **Memorie permanenti**

- EEPROM, Flash,
- Hard disk, etc...

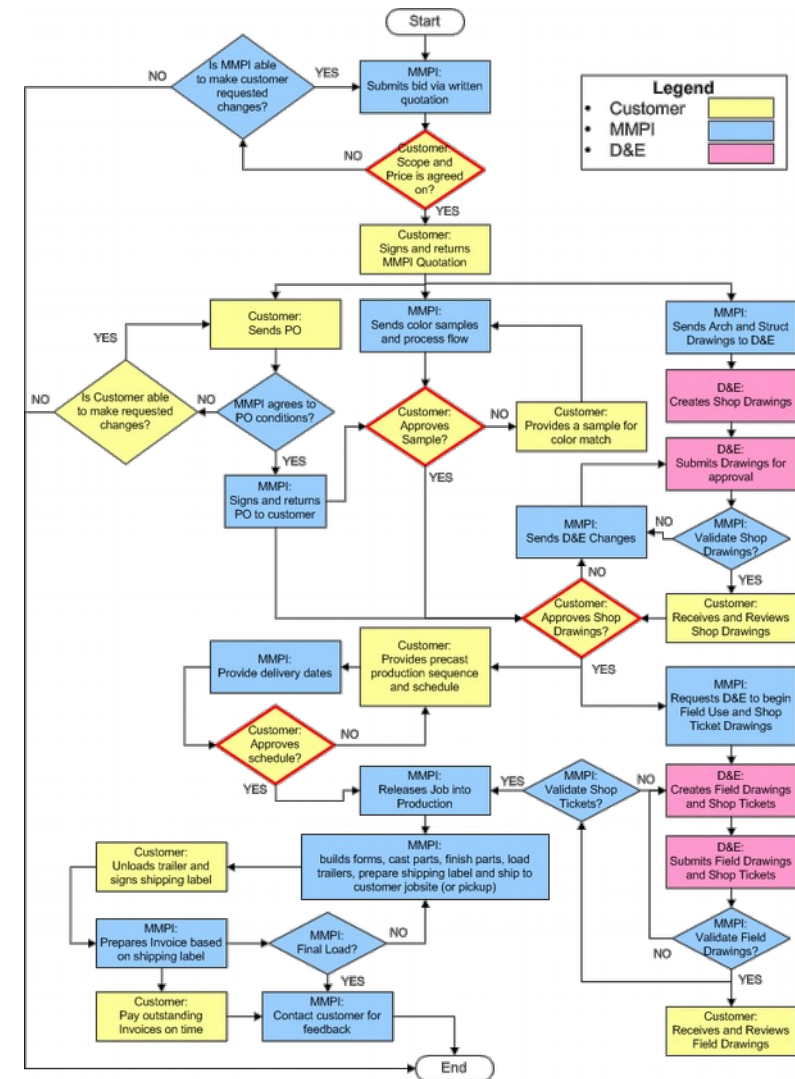


FONDAMENTI DI PROGRAMMAZIONE



Logiche di un calcolatore

- Come “ragiona” un microcontrollore



Algoritmi

- **Il concetto**

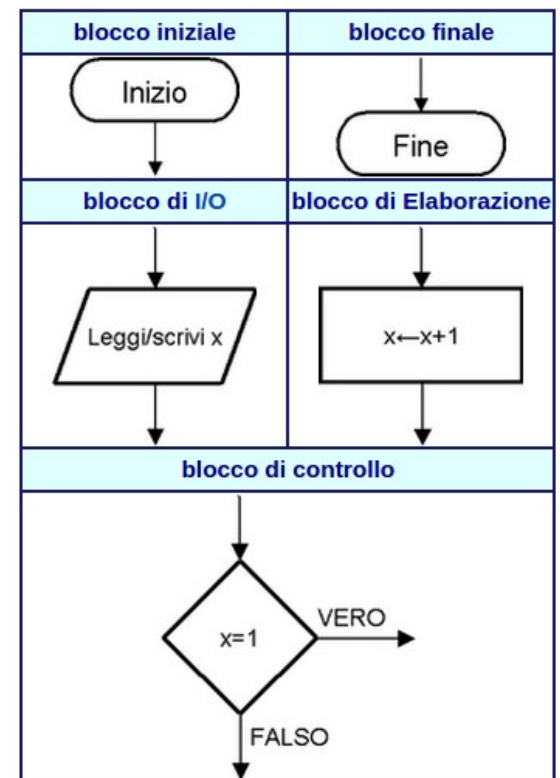
- un procedimento che risolve un determinato problema attraverso un numero finito di passi elementari

- **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
SUB
AND
OR
JOR
AUI.T

00001	RD	RS2	RS1
00010	RD	RS2	RS1
00011	RD	RS2	RS1
00100	RD	RS2	RS1
00101	RD	RS2	RS1
00110	RD	RS2	RS1
01010	RD	RS2	RS1

OADPC
OADIMM
OAO
TORE

01011	RD	x x x x x x x x	
01001	# dato	x x	RD
00111	RD	x x x x x	RS1
01000	x x x x x	RS1	RD

JMPZ
JMPIND
JMPINDYR

01101	x x x x x x x x x x	RS1
01110	RD	x x x x x x x x x x
01111	RD	x x x x x x x x x x

ESET
OP

11110	x x x x x x x x x x x x x x
11111	x x x x x x x x x x x x x x

Il processo di traduzione

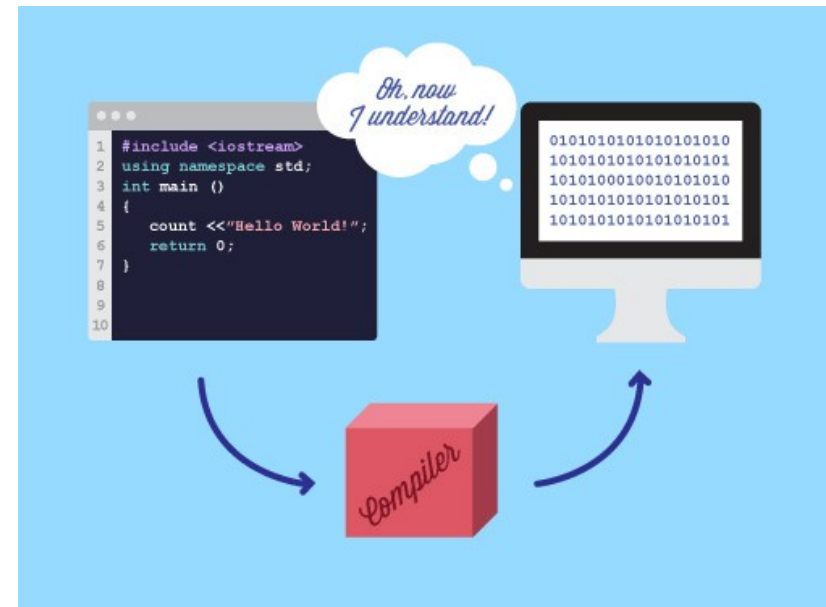
- **Ambienti compilati**

- Tutto il sorgente
 - Traduzione completa
- Si distribuisce il risultato
 - Binario eseguibile

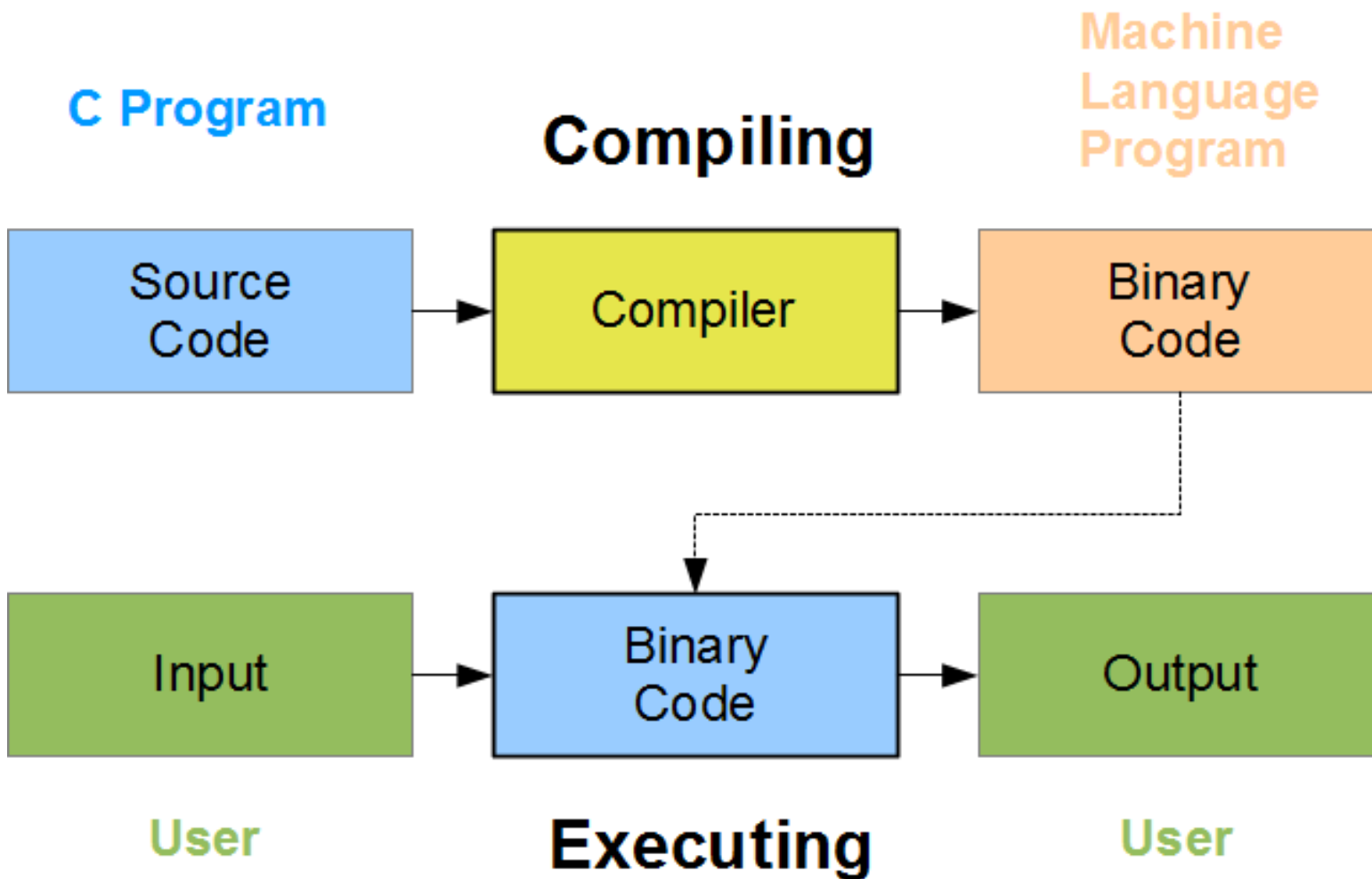


- **Ambienti interpretati**

- Riga per riga
 - Interpretato sul momento
- Si distribuisce il sorgente
 - Verrà reinterpretrato



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 del linguaggio

- **Macro e Costanti**

- `#define PI_GRECO 3.14`
- `const int LedPin =3;`

- **Variabili**

- `int x;`

- **Istruzioni**

- `x=3;`

- **Funzioni**

- `x=somma(2,3);`

- **Strutture di controllo: if**

- `If (x>3) { }`

PROGRAMMARE ARDUINO



PROGRAMMARE ARDUINO

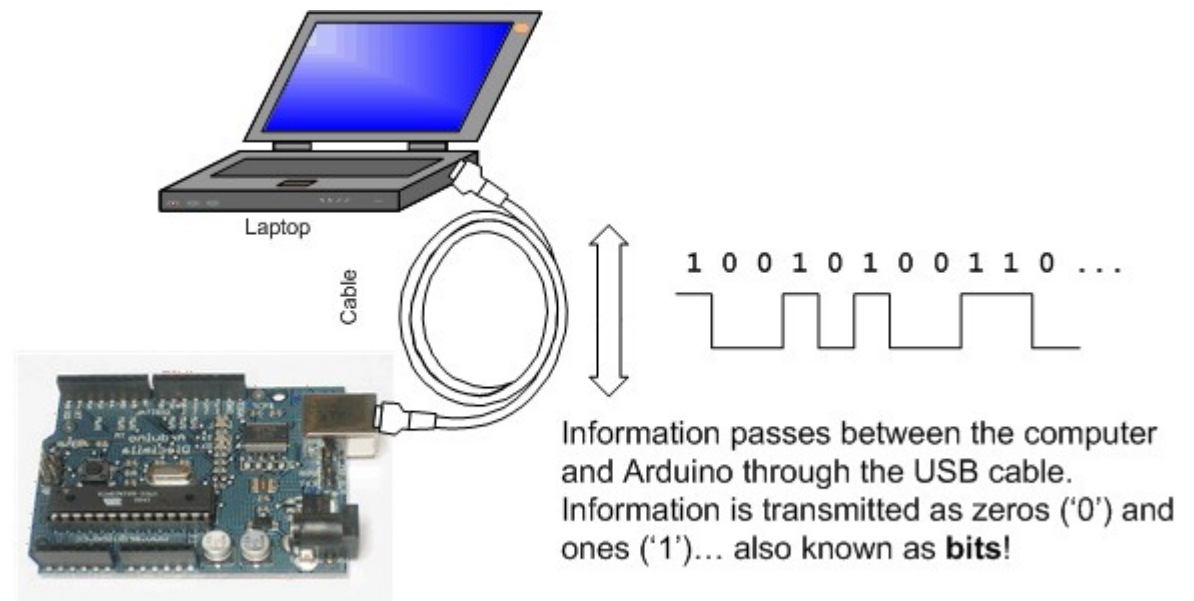
- **L'ambiente di sviluppo**
 - Installazione e setup
- **Struttura di base di uno sketch**
 - Cartella degli sketch
 - Struttura dello sketch
 - setup
 - loop
 - File di progetto .ino
 - File esterni .h e .cpp
- **Caricare uno sketch**
 - Comunicazione seriale

Comunicazione Seriale

- **Comunicazione serial per il flash del firmware**
- **Comunicazione seriale di debug**
- **Output seriale**

- **Oggetto Serial**

- `Serial.begin()`
- `Serial.print()`
- `Serial.println()`



Serial 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**

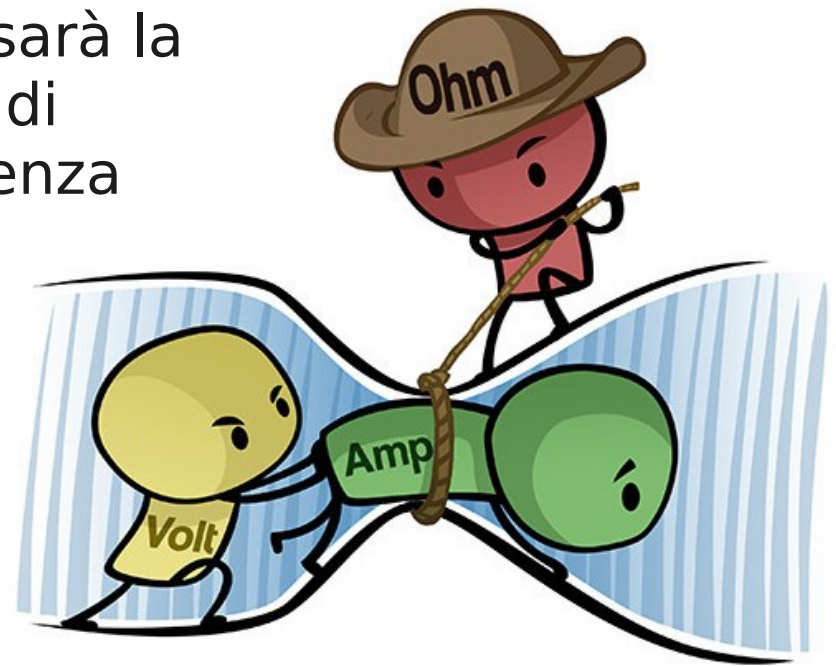
- **Corrente**

- 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**)

Resistenza

- **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.



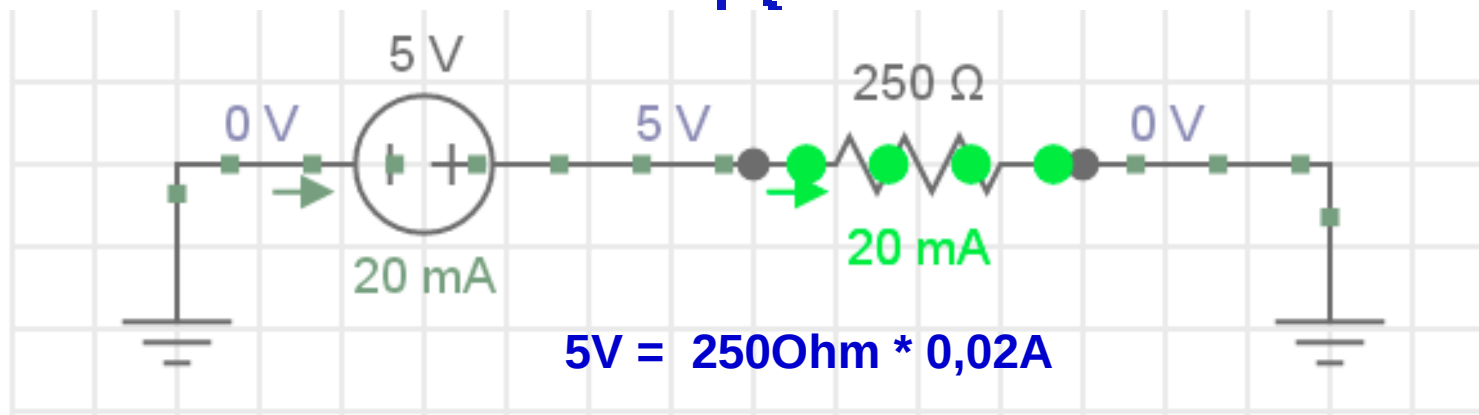
Prima legge di Ohm

- **Enunciato**

- la corrente è direttamente proporzionale alla tensione ed inversamente proporzionale alla resistenza

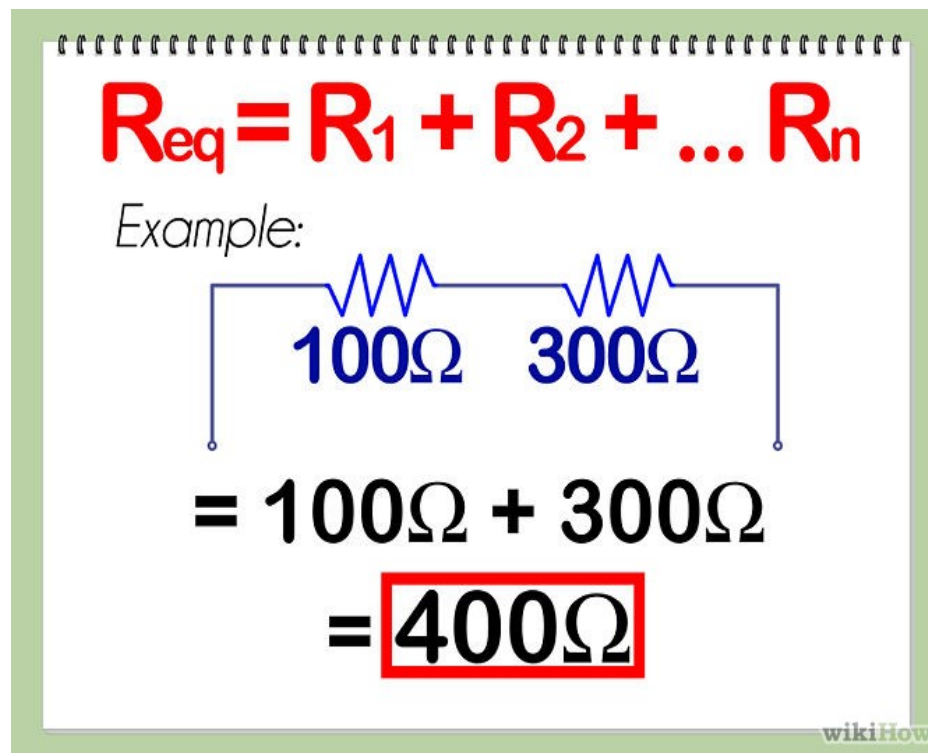
- **Formule**

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




























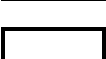
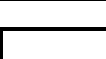


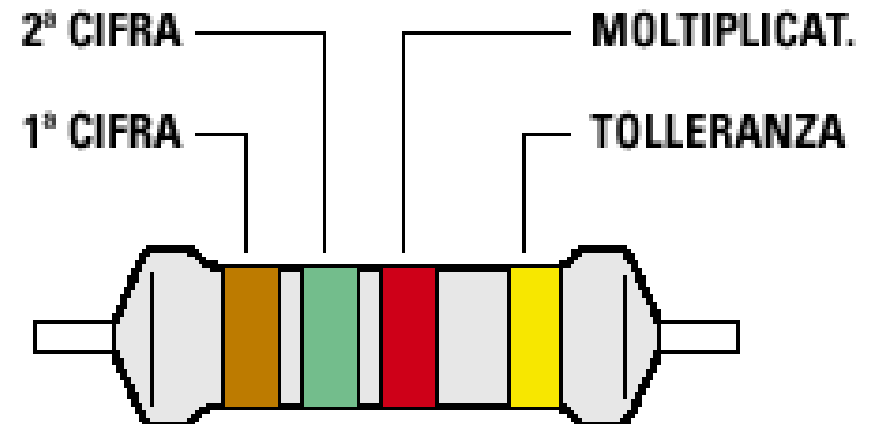
Resistenze in serie

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



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()
- **Fare lampeggiare un LED**
 - complessità

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

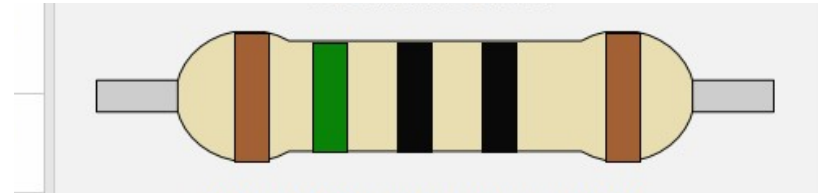
- **Corrente di esercizio**

- 20 mA

Resistenze consigliate x Kit

- **Rossi, Gialli**

- 150 Ohm
- Ma anche 220, 180...

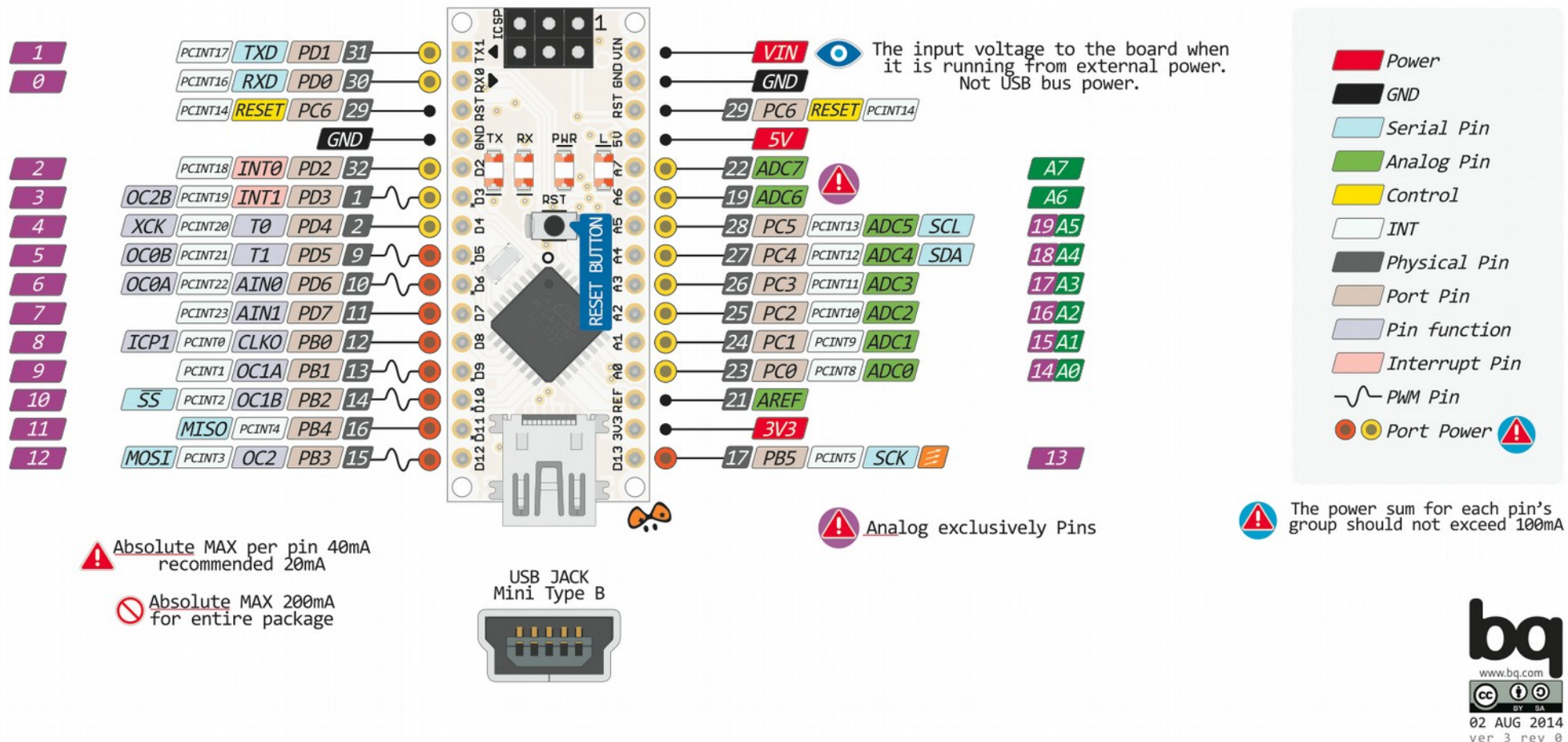


- **Bianchi, Blu, Verdi**

- 100 Ohm
- Ma anche 150, 120 ...

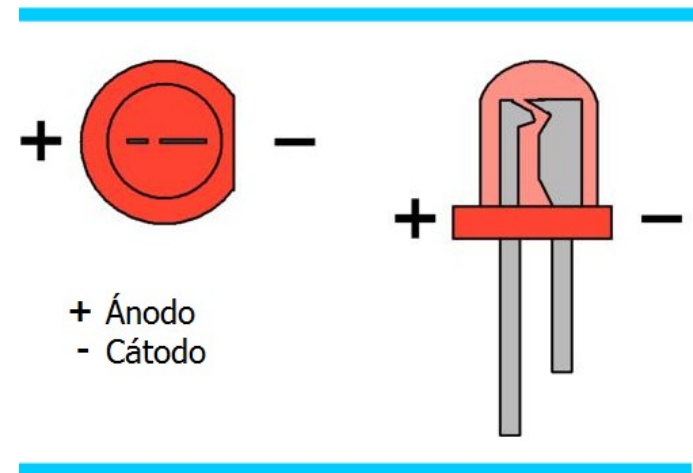
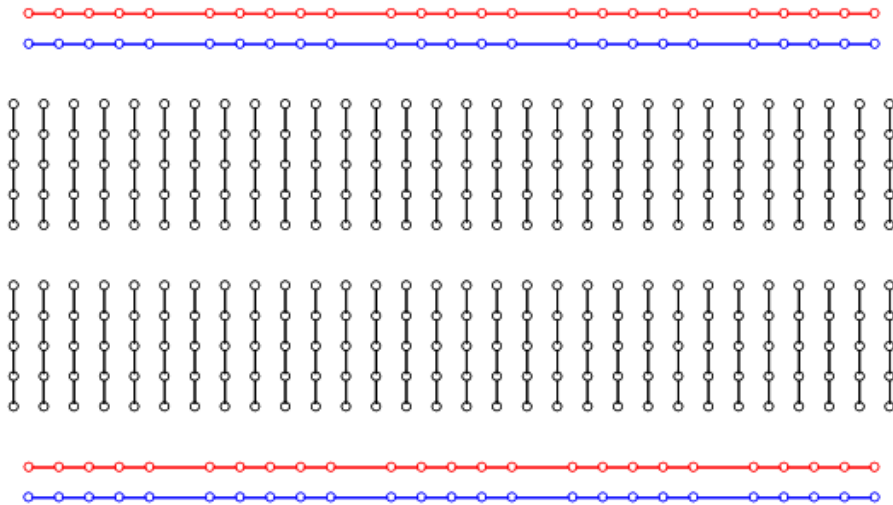


Arduino Nano

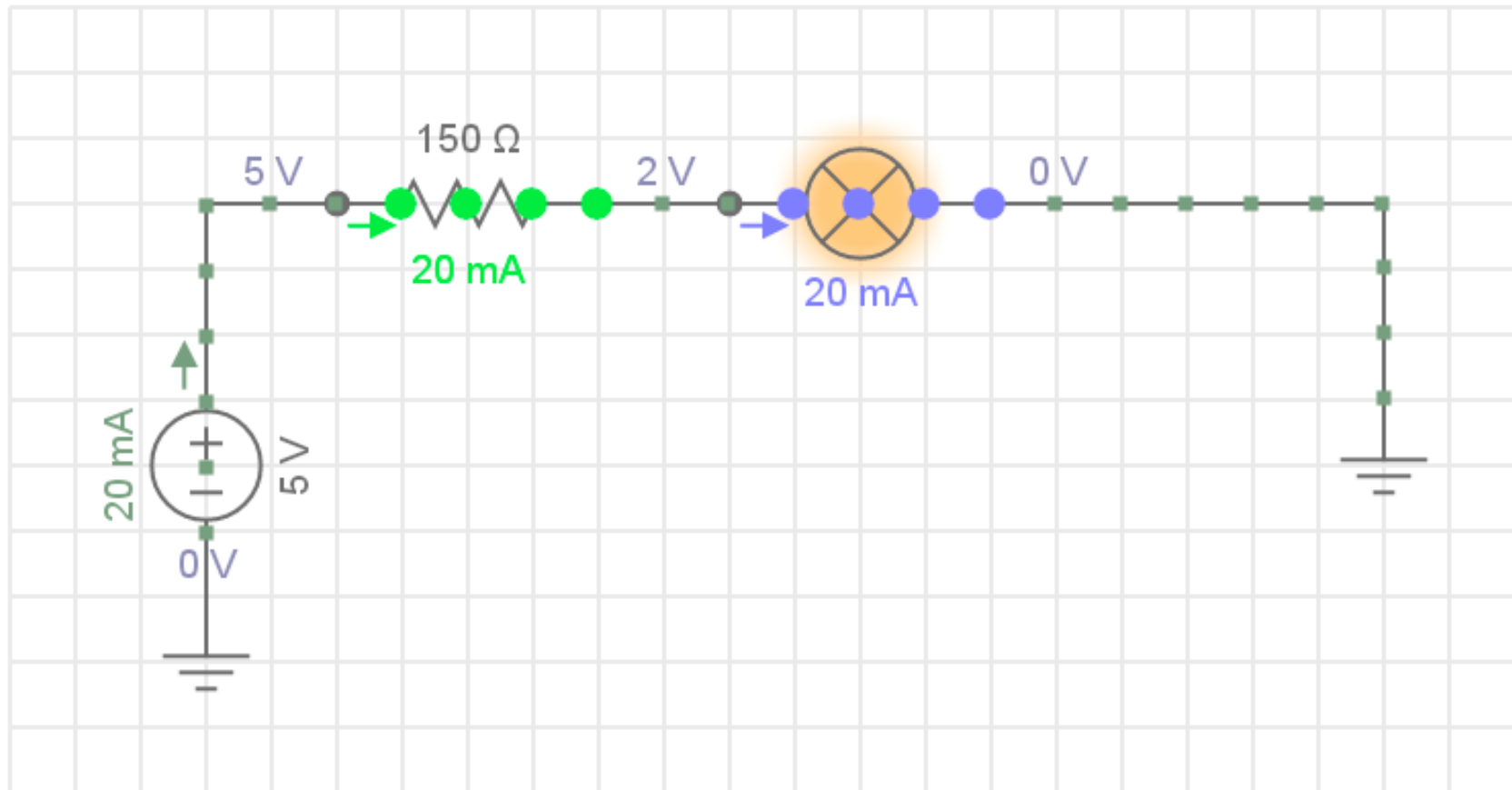


Cablaggio

- **Breadboard e LED**



LED e resistenza in serie



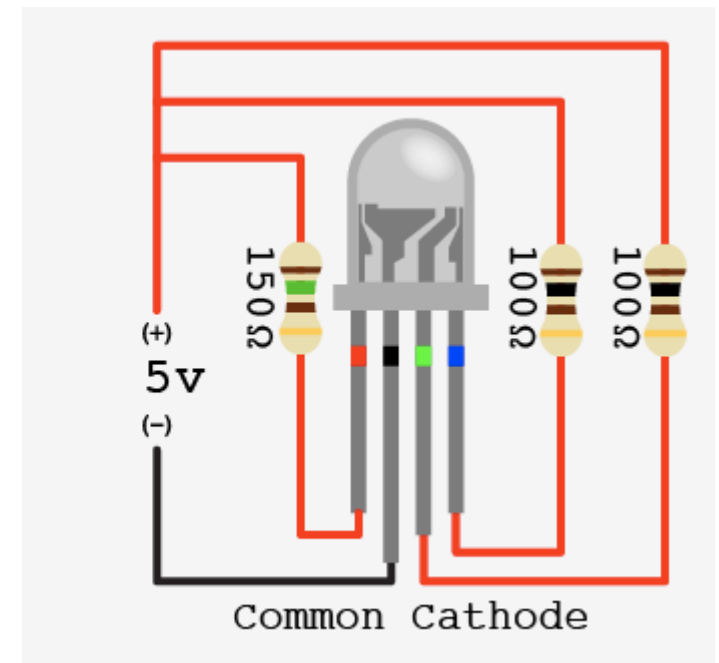
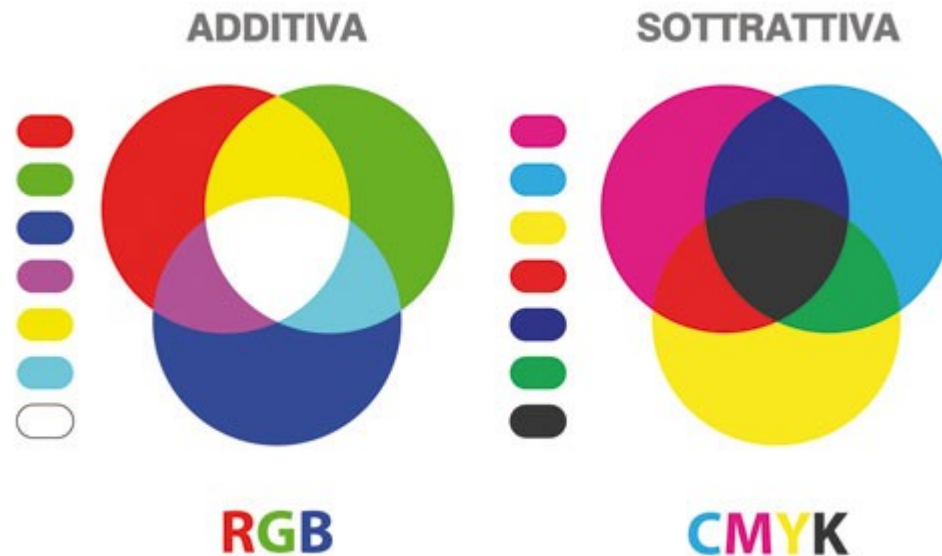
Led Blink

Blink §

```
1 // the setup function runs once when you press reset or power the board
2 void setup() {
3   // initialize digital pin 13 as an output.
4   pinMode(13, OUTPUT);
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
9   digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
10  delay(1000);             // wait for a second
11  digitalWrite(13, LOW);   // turn the LED off by making the voltage LOW
12  delay(1000);             // wait for a second
13 }
```

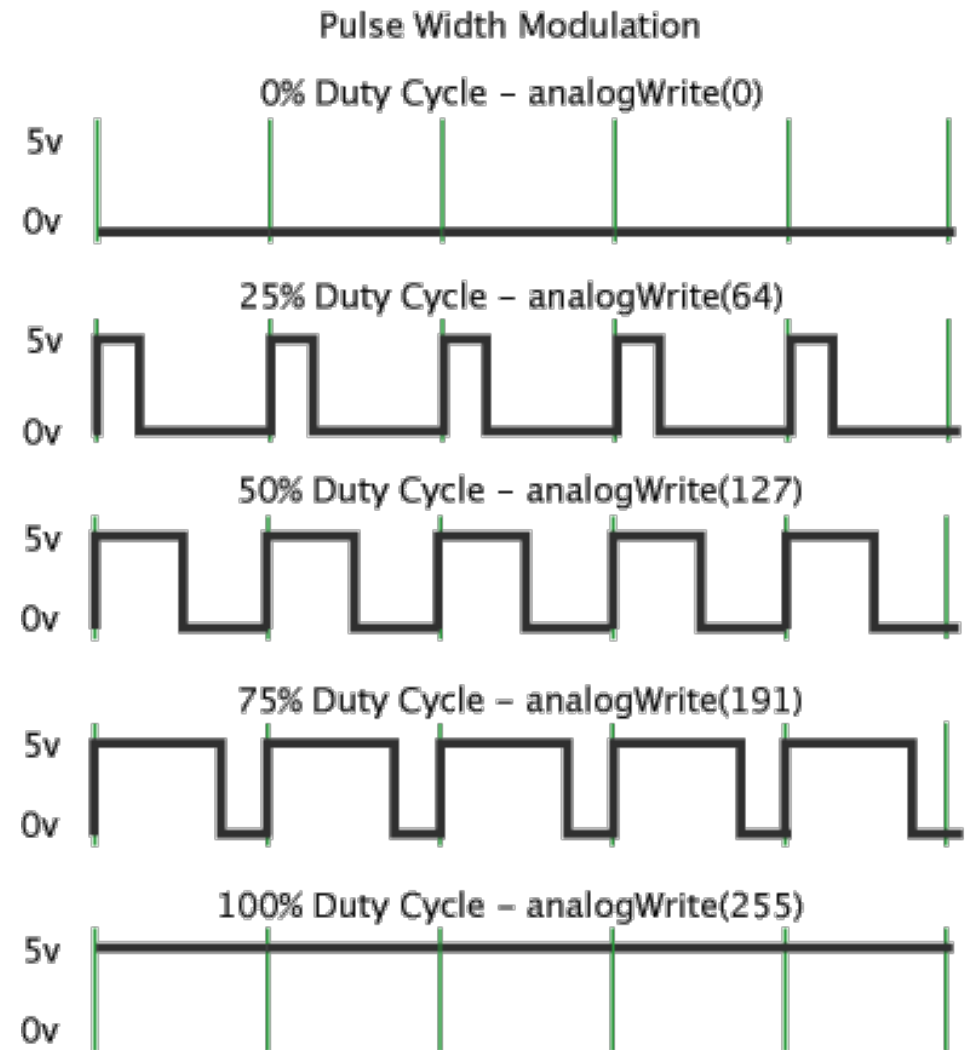
GESTIRE LED RGB

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



Output 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 5

```
1 // the setup function runs once when you press reset or power the board
2 #define RED 9
3 #define GREEN 10
4 #define BLUE 11
5
6 void setup() {
7     // initialize digital pin 13 as an output.
8     pinMode(RED, OUTPUT);
9     pinMode(GREEN, OUTPUT);
10    pinMode(BLUE, OUTPUT);
11 }
12 // the loop function runs over and over again forever
13 void loop() {
14     analogWrite(RED, 100);
15     analogWrite(GREEN, 200);
16     analogWrite(BLUE, 50);
17     delay(1000);
18     analogWrite(RED, 0);
19     analogWrite(GREEN, 0);
20     analogWrite(BLUE, 0);
21     delay(1000);
22 }
```

GESTIRE UN BUZZER

- **modulare suoni con Arduino**

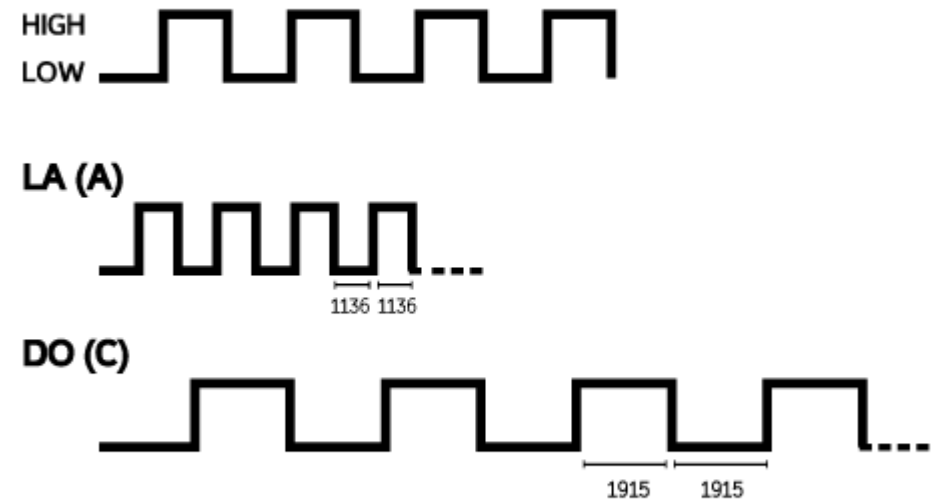
- PIN
 - Qualsiasi PIN digitale
 - WARNING: usa interrupts
 - Potrebbe influire su PWM e librerie che usano timer
- tone()

- **mappatura delle note**

- Frequenza in Hz

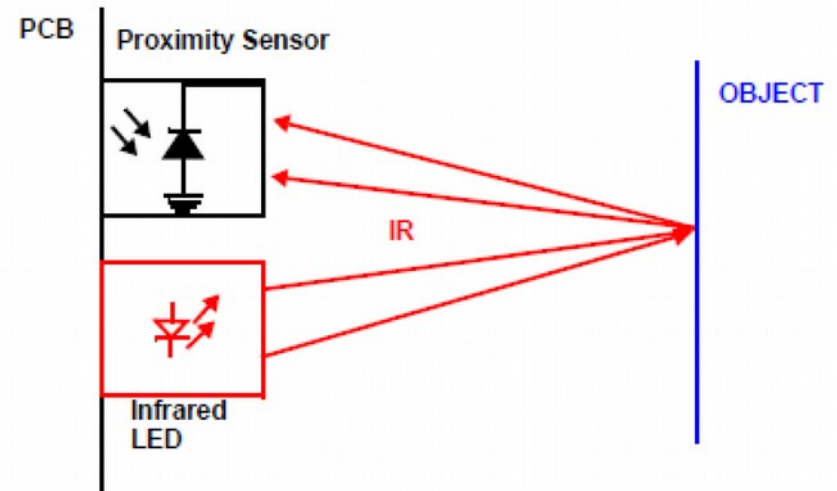
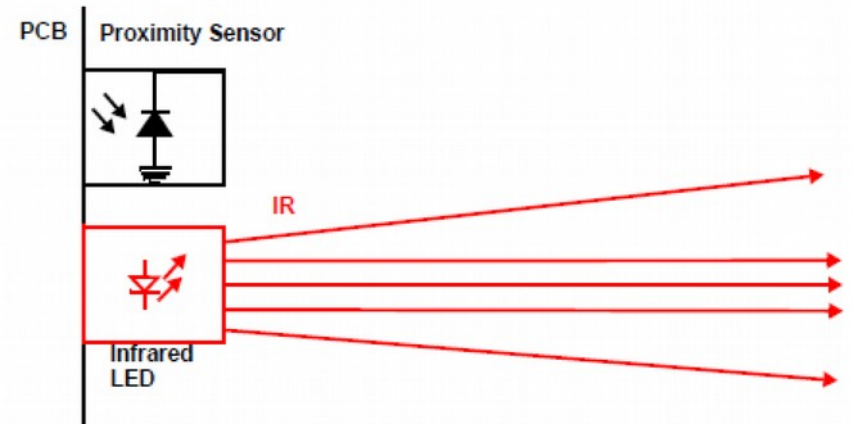
- **produrre una melodia**

- tone e delay



IR PROXIMITY SENSOR

- **Struttura del sensore**
 - Emettitore IR
 - Ricevitore IR
 - Principio di funzionamento
- **Gestire un sensore IR da Arduino**
 - Sensori digitali con DO
 - Digital Output
- **Leggere input digitale**
 - `digitalRead()`



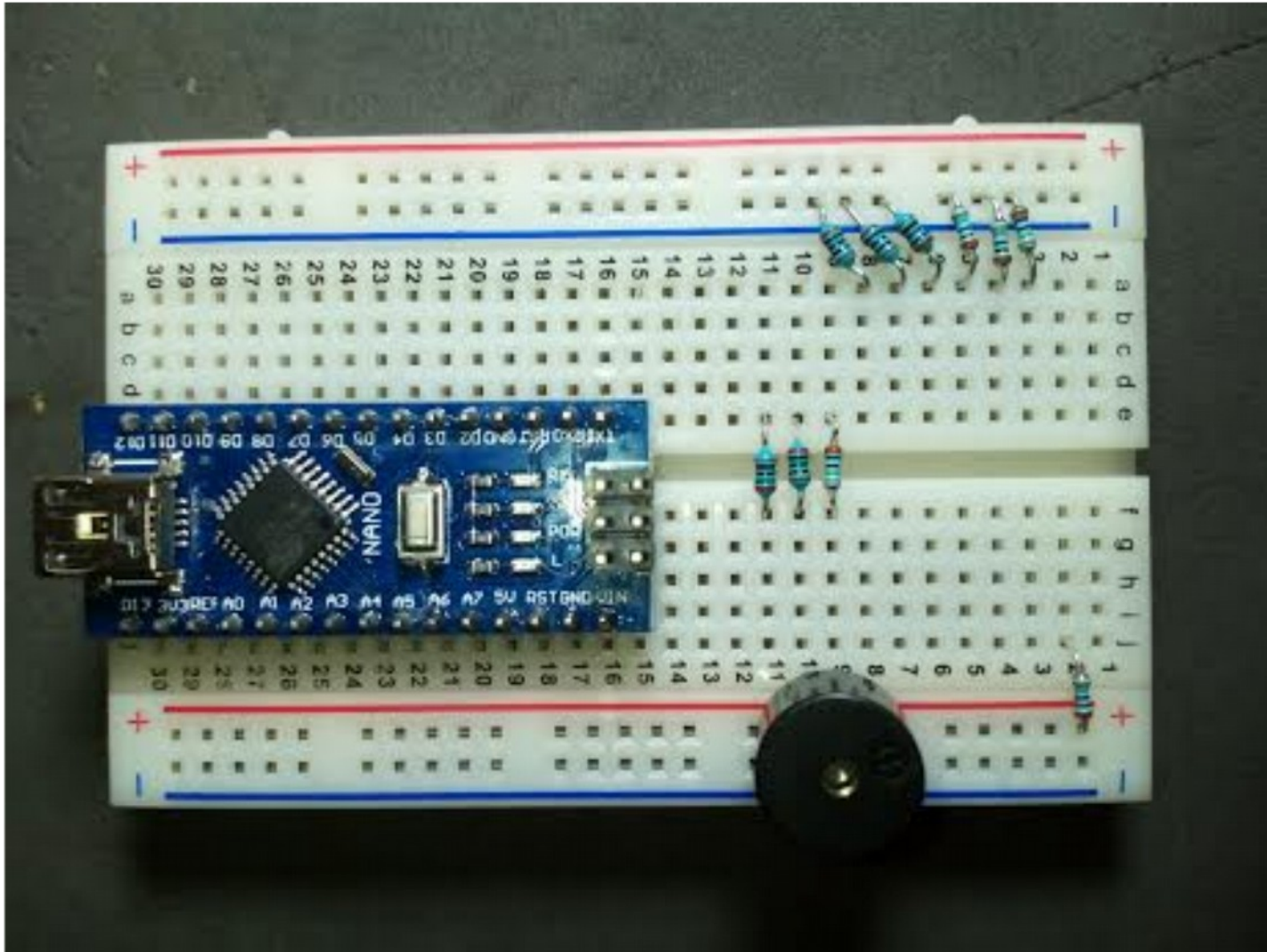
COSTRUZIONE DEL PROGETTO



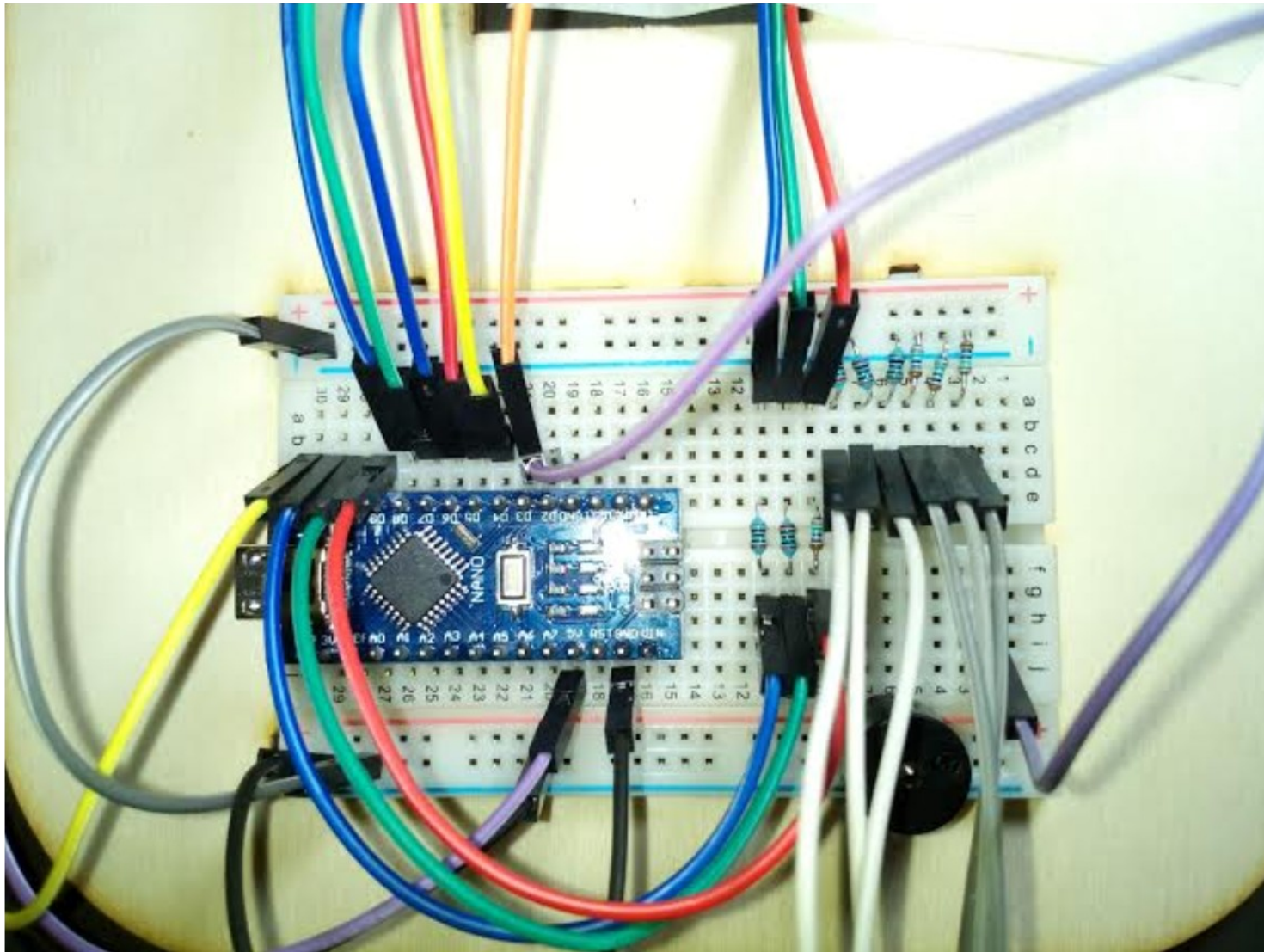
Fasi del progetto

- **Preparazione**
 - Breadboard
 - Resistenze
- **Cablaggio e montaggio**
 - LED RGB
 - LEDs
 - Sensore IR
 - Speaker
- **Sketch**
 - setup
 - loop

Preparazione Resistenze sulla Breadboard



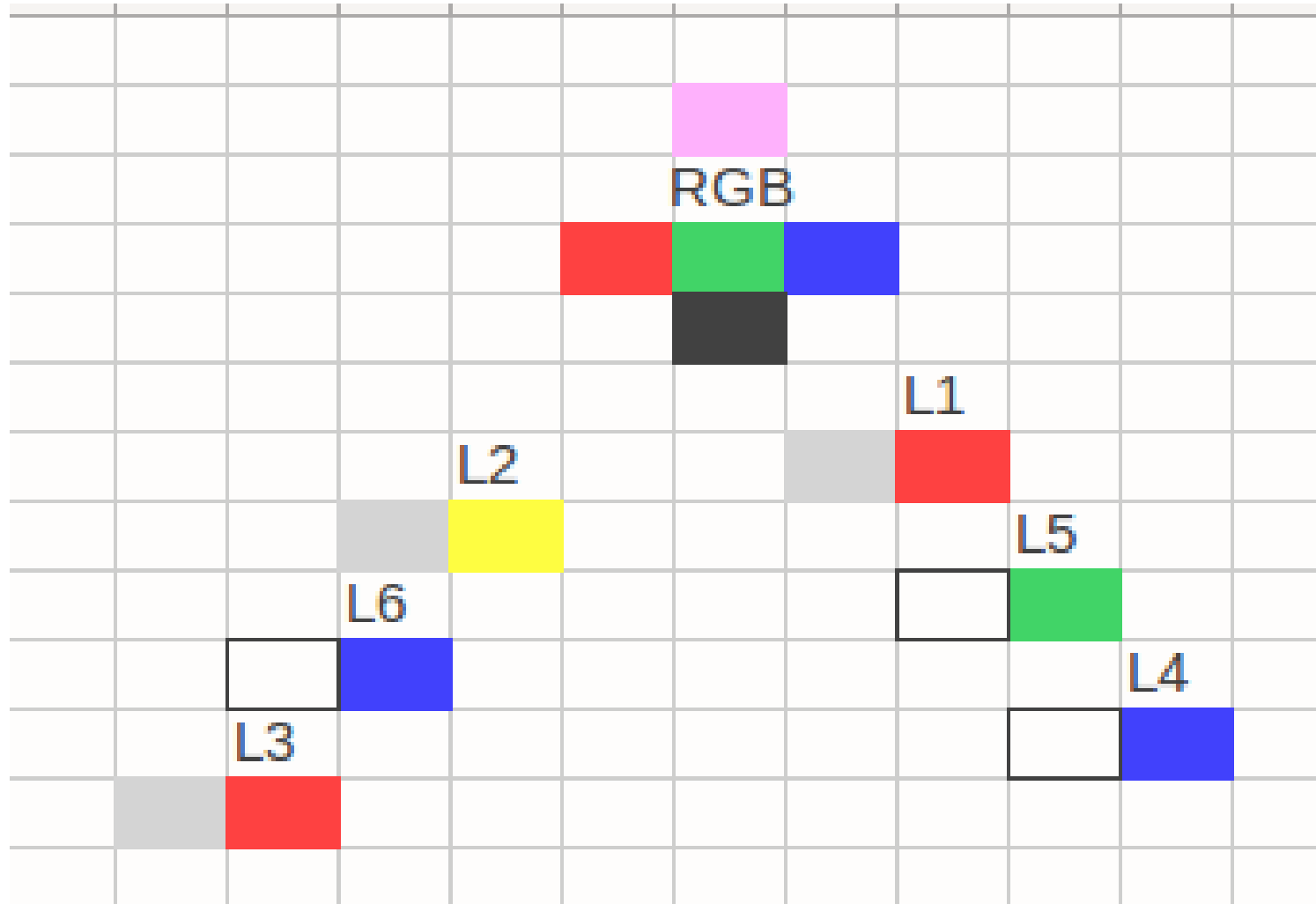
Cablaggio

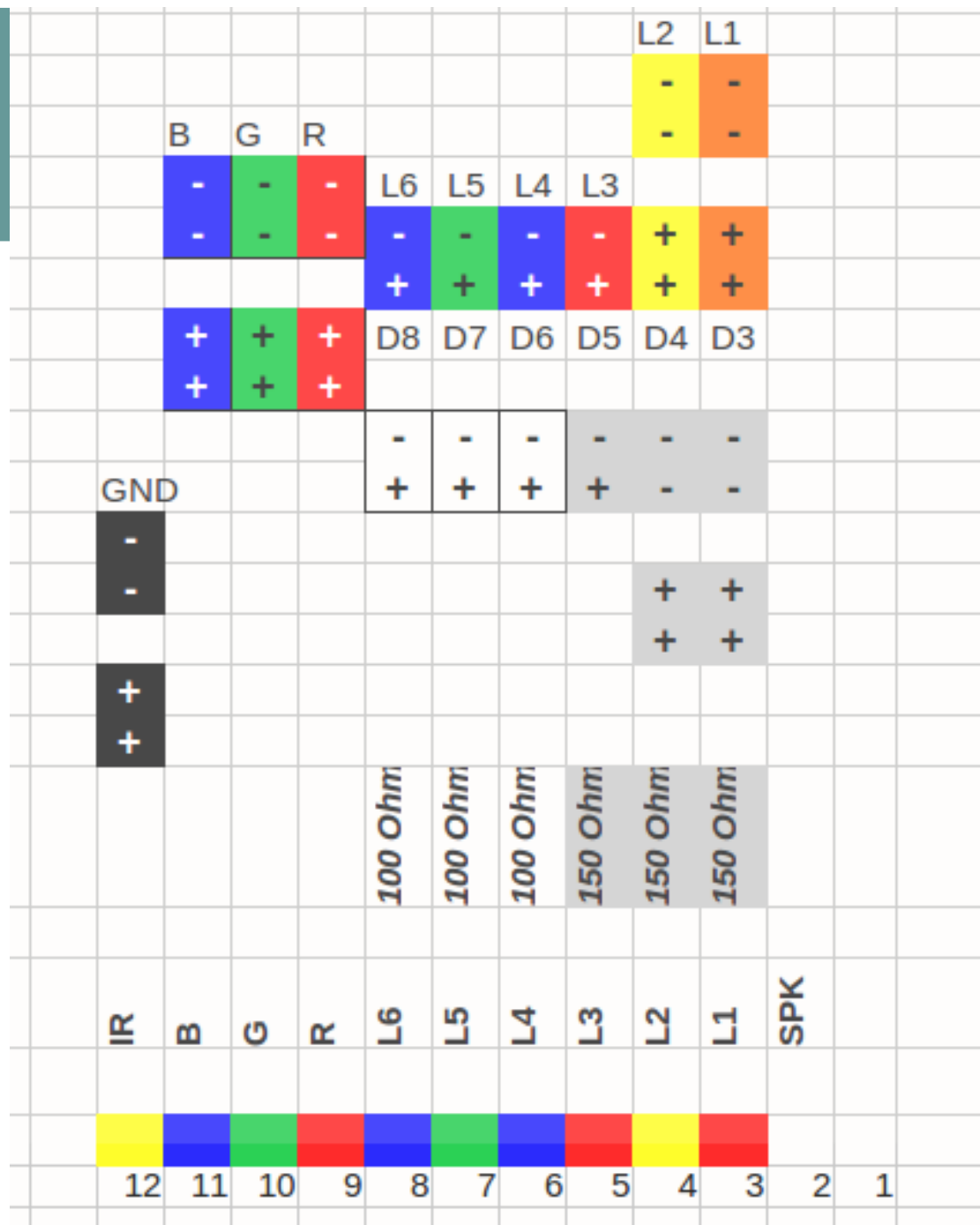


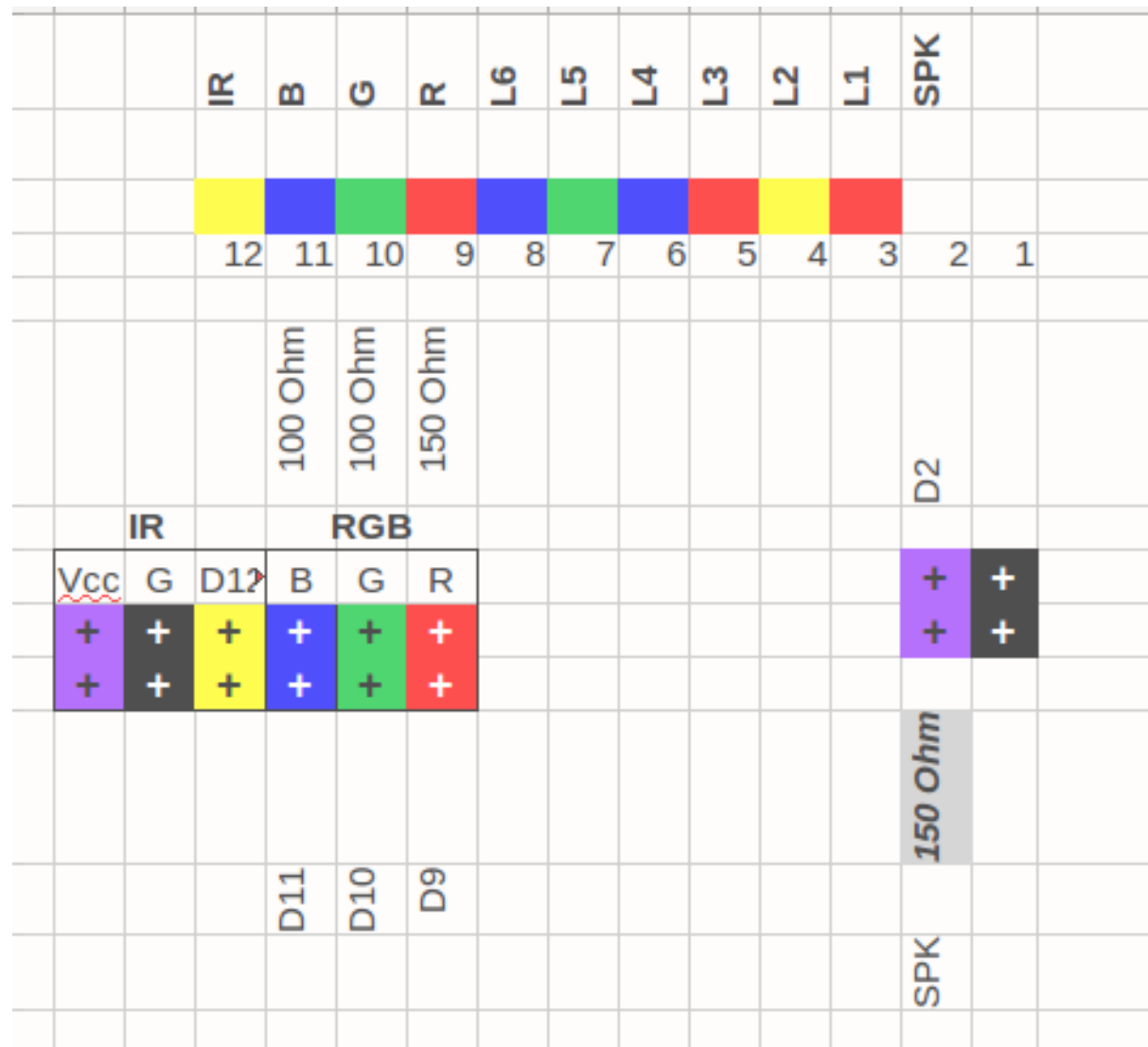
LED




Schema LED








Sketch: definizioni



WORKSHOP_XMAS15_def

```
1 // Notes definition
2 #define D0 523
3 #define RE 587
4 #define MI 659
5 #define FA 698
6 #define SOL 784
7 #define LA 880
8 #define SI 988
9
10 // PINS definitions
11 #define SPKR_PIN 2
12 #define L1_PIN 3
13 #define L2_PIN 4
14 #define L3_PIN 5
15 #define L4_PIN 6
16 #define L5_PIN 7
17 #define L6_PIN 8
18 #define R_PIN 9
19 #define G_PIN 10
20 #define B_PIN 11
21 #define IR_PIN 12
22
```

Sketch: Setup



```
WORKSHOP_XMAS15_def
23 void setup() {
24   // Hello
25   Serial.begin(9600);
26   Serial.println("Merry Christmas");
27   // Pins setup
28   pinMode(L1_PIN, OUTPUT);
29   pinMode(L2_PIN, OUTPUT);
30   pinMode(L3_PIN, OUTPUT);
31   pinMode(L4_PIN, OUTPUT);
32   pinMode(L5_PIN, OUTPUT);
33   pinMode(L6_PIN, OUTPUT);
34   pinMode(R_PIN, OUTPUT);
35   pinMode(G_PIN, OUTPUT);
36   pinMode(B_PIN, OUTPUT);
37   pinMode(SPKR_PIN, OUTPUT);
38   pinMode(IR_PIN, INPUT);
39   // Initial state
40   digitalWrite(L1_PIN, OUTPUT);
41   digitalWrite(L2_PIN, OUTPUT);
42   digitalWrite(L3_PIN, OUTPUT);
43   digitalWrite(L4_PIN, OUTPUT);
44   digitalWrite(L5_PIN, OUTPUT);
45   digitalWrite(L6_PIN, OUTPUT);
46   analogWrite(R_PIN, 50);
47   analogWrite(G_PIN, 50);
48   analogWrite(B_PIN, 50);
49 }
50
```

Sketch: loop 1/3



```
51 void loop() {
52   // proximity check
53   if(digitalRead(12)==1) return;
54
55   // Jingle Bells
56
57   tone(SPKR_PIN,MI,250);
58   delay(500);
59   analogWrite(R_PIN,0);
60
61   tone(SPKR_PIN,MI,250);
62   delay(500);
63   analogWrite(G_PIN,0);
64
65   tone(SPKR_PIN,MI,500);
66   delay(1000);
67   analogWrite(B_PIN,0);
68
69   // -----
70
71   digitalWrite(L1_PIN,LOW);
72   tone(SPKR_PIN,MI,250);
73   delay(500);
74
75   digitalWrite(L2_PIN,LOW);
76   tone(SPKR_PIN,MI,250);
77   delay(500);
78
79   digitalWrite(L3_PIN,LOW);
80   tone(SPKR_PIN,MI,500);
81   delay(1000);
```

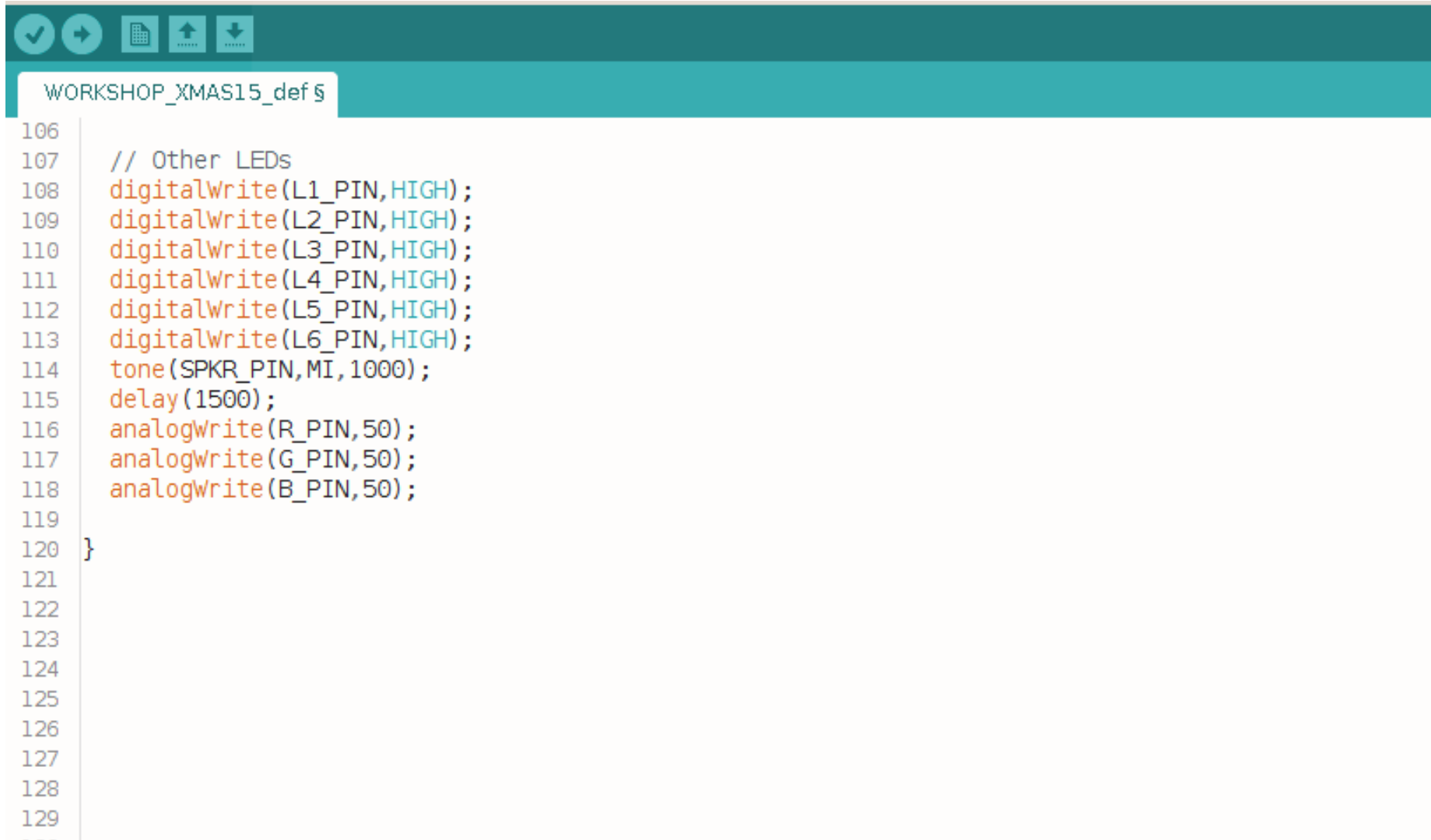
Sketch: loop 2/3



WORKSHOP_XMAS15_def \$

```
82
83 // -----
84
85 digitalWrite(L4_PIN, LOW);
86 tone(SPKR_PIN, MI, 250);
87 delay(500);
88
89 digitalWrite(L5_PIN, LOW);
90 tone(SPKR_PIN, SOL, 250);
91 delay(500);
92
93 digitalWrite(L6_PIN, LOW);
94 tone(SPKR_PIN, D0, 250);
95 delay(500);
96
97 // RGB LED
98 analogWrite(R_PIN, 100);
99 analogWrite(G_PIN, 120);
100 analogWrite(B_PIN, 80);
101 tone(SPKR_PIN, RE, 250);
102 delay(500);
103 analogWrite(R_PIN, 0);
104 analogWrite(G_PIN, 0);
105 analogWrite(B_PIN, 0);
106
```


Sketch: loop 3/3



```
106
107 // Other LEDs
108 digitalWrite(L1_PIN,HIGH);
109 digitalWrite(L2_PIN,HIGH);
110 digitalWrite(L3_PIN,HIGH);
111 digitalWrite(L4_PIN,HIGH);
112 digitalWrite(L5_PIN,HIGH);
113 digitalWrite(L6_PIN,HIGH);
114 tone(SPKR_PIN,MI,1000);
115 delay(1500);
116 analogWrite(R_PIN,50);
117 analogWrite(G_PIN,50);
118 analogWrite(B_PIN,50);
119
120 }
121
122
123
124
125
126
127
128
129
---
```

LETS TRY



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**
 - Aggiungete funzionalità
 - 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 Christmas Tree gli sketch dei vostri amici Makers

**Grazie e
arrivederci**

