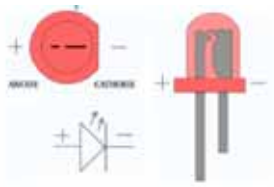
Output digitale - Led interno



Primo esercizio.
Arduino ha un led montato sulla scheda e collegato al pin 13.
Proviamo ad accenderlo e farlo lampeggiare.
I comandi da usare sono:

pinMode(13, OUTPUT); // prepara la porta 13 in uscita
digitalWrite(13, 1); // accende il led
digitalWrite(13, 0); // spegne il led
delay(1000); // pausa di 1 secondo (1000 msec)

; (punto e virgola)
hai notato che tutte i comandi terminano con ; ?
questo indica la fine di un comando,
se ometti questo simbolo il programma non funziona.

{ ... } (parentesi graffe)
con questi simboli indichiamo e raggruppiamo una serie di comandi in un unico blocco.

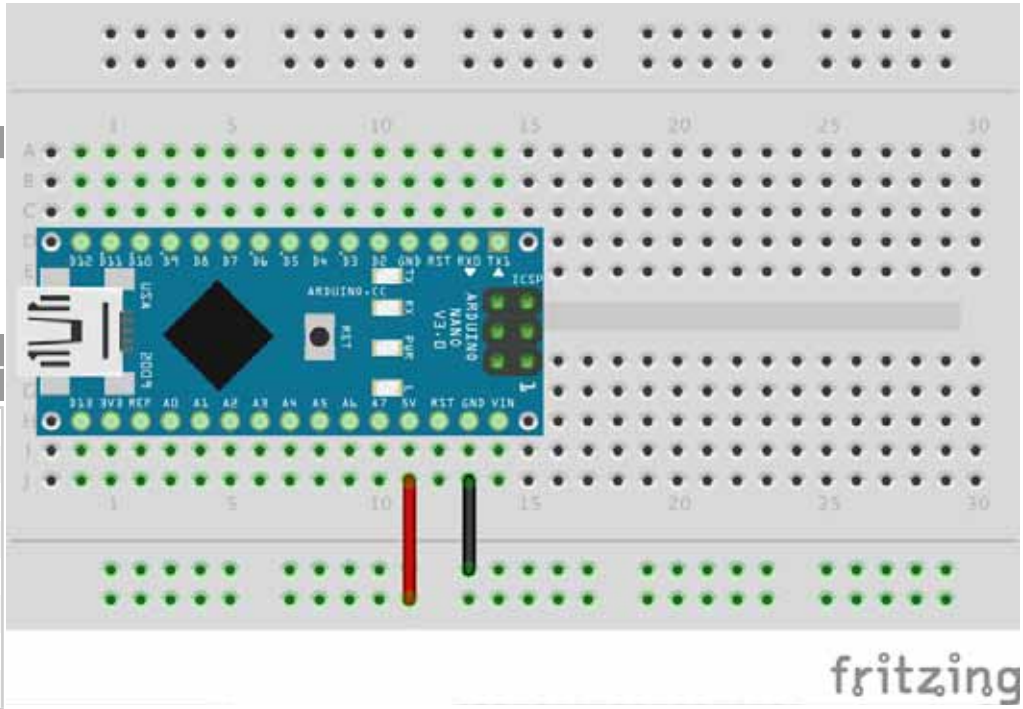
void setup() { }
assegna al blocco il nome setup,
questo all'avvio di Arduino (o al reset) verrà eseguito per primo ed eseguito una sola volta, terminato Arduino passerà il blocco di nome loop.
In questa parte si mettono i comandi di preparazione del microcontrollore.

void loop() { }
assegna al blocco il nome loop, questo verrà eseguito in continuo, loop appunto, cioè Arduino eseguirà i comandi contenuti in continuo.
L'elaborazione terminerà quando Arduino verrà spento o premuto il tasto reset.

librerie

componenti

nome	valore	q.tà
Arduino Nano v3		1
Basetta 400		1



Output digitale - Led interno

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

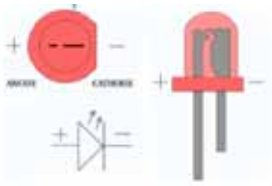
oppure (lampeggio più veloce):

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(100);
  digitalWrite(13, LOW);
  delay(100);
}
```

oppure (lampeggio uno veloce uno lento):

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(100);
  digitalWrite(13, LOW);
  delay(100);
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Tre leds esterni

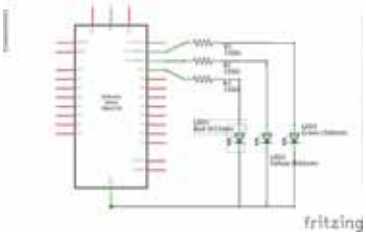


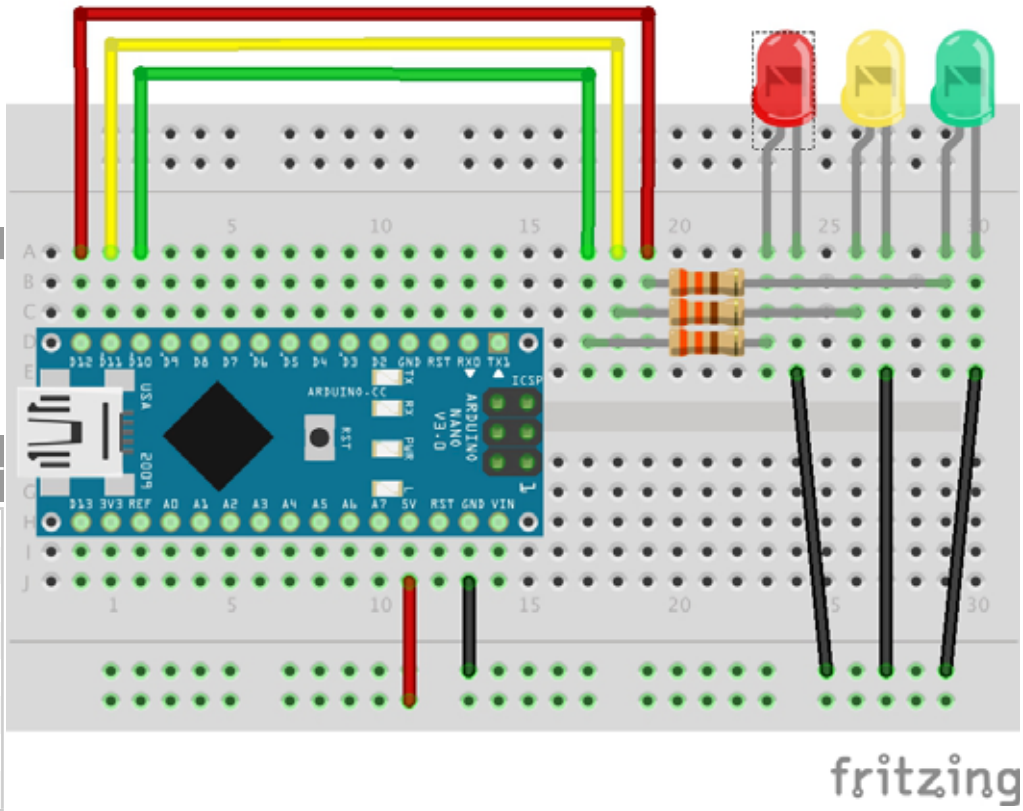
Collega, utilizzando la basetta, un led ed una resistenza ad un piedino digitale.
Ricorda di non usare MAI il pin 0 e pin 1 perché sono impiegati per il collegamento di Arduino al computer via USB.
Un led va alimentato con una tensione di 1 - 1,5 volt, Arduino ne fornisce 5volt: troppi; il led si guasterebbe, (massima corrente 10-20 mA) dobbiamo mettere sempre una resistenza per limitare la corrente che scorre all'interno del led.
La resistenza può avere un valore 270Ω, 330Ω o 390Ω, minore è il valore della resistenza maggiore sarà la luminosità del led.
Modifica il codice dell'esercizio precedente sostituendo il numero 13 (piedino 13) con il numero del piedino che hai impiegato.

Prova a far cambiare la durata dei lampeggi, esempio:

uno breve uno lungo
tre brevi e pausa lunga
ecc.

Aggiungi altri due leds (rosso giallo verde)





librerie

componenti

nome	valore		q.tà
Arduino Nano v3			1
Basetta 400			1
Resistenza	330	Ω	3
Led	?		3

Tre leds esterni

```
void setup() {
  pinMode(12, OUTPUT);
}

void loop() {
  digitalWrite(12, HIGH);
  delay(1000);
  digitalWrite(12, LOW);
  delay(1000);
}
```

********** **oppure:**

```
void setup() {
  pinMode(12, OUTPUT);
}

void loop() {
  digitalWrite(12, HIGH);
  delay(1000);
  digitalWrite(12, LOW);
  delay(500);
  digitalWrite(12, HIGH);
  delay(500);
  digitalWrite(12, LOW);
  delay(1000);

}
```

********** **oppure: il Semaforo**

```
#define ledrosso  12
#define ledgiallo 11
#define ledverde  10

void setup() {
  pinMode(ledrosso, OUTPUT);
  pinMode(ledgiallo, OUTPUT);
  pinMode(ledverde, OUTPUT);
}


void loop() {
  digitalWrite(ledverde, 1);
  delay(10000);
  digitalWrite(ledgiallo, 1);
  delay(2000);
  digitalWrite(ledverde, 0);
  digitalWrite(ledgiallo, 0);
  digitalWrite(ledrosso, 1);
  delay(5000);
  digitalWrite(ledrosso, 0);
}
```

********** **oppure: scorrimento**

```
#define ledrosso  12
#define ledgiallo 11
#define ledverde  10
#define tempo   500

void setup() {
  pinMode(ledrosso, OUTPUT);
  pinMode(ledgiallo, OUTPUT);
  pinMode(ledverde, OUTPUT);
}


void loop() {
  digitalWrite(ledrosso, 1);
  delay(tempo);
  digitalWrite(ledrosso, 0);
  digitalWrite(ledgiallo, 1);
  delay(tempo);
  digitalWrite(ledverde, 1);
```

Tre leds esterni

```
  digitalWrite(ledgiallo, 0);
  delay(tempo);
  digitalWrite(ledverde, 0);
}
```

8 leds - Array resistivo

Quando vi ha la necessità di aggiungere altri leds si occupano le porte libere a ciascun led si deve collegare la solita resistenza.
Vi è un componente che integra una serie di resistenze uguali, queste sono tutte collegate ad un piedino che collegheremo alla massa. **Array Resistivo** con questo semplifichiamo il montaggio.

librerie

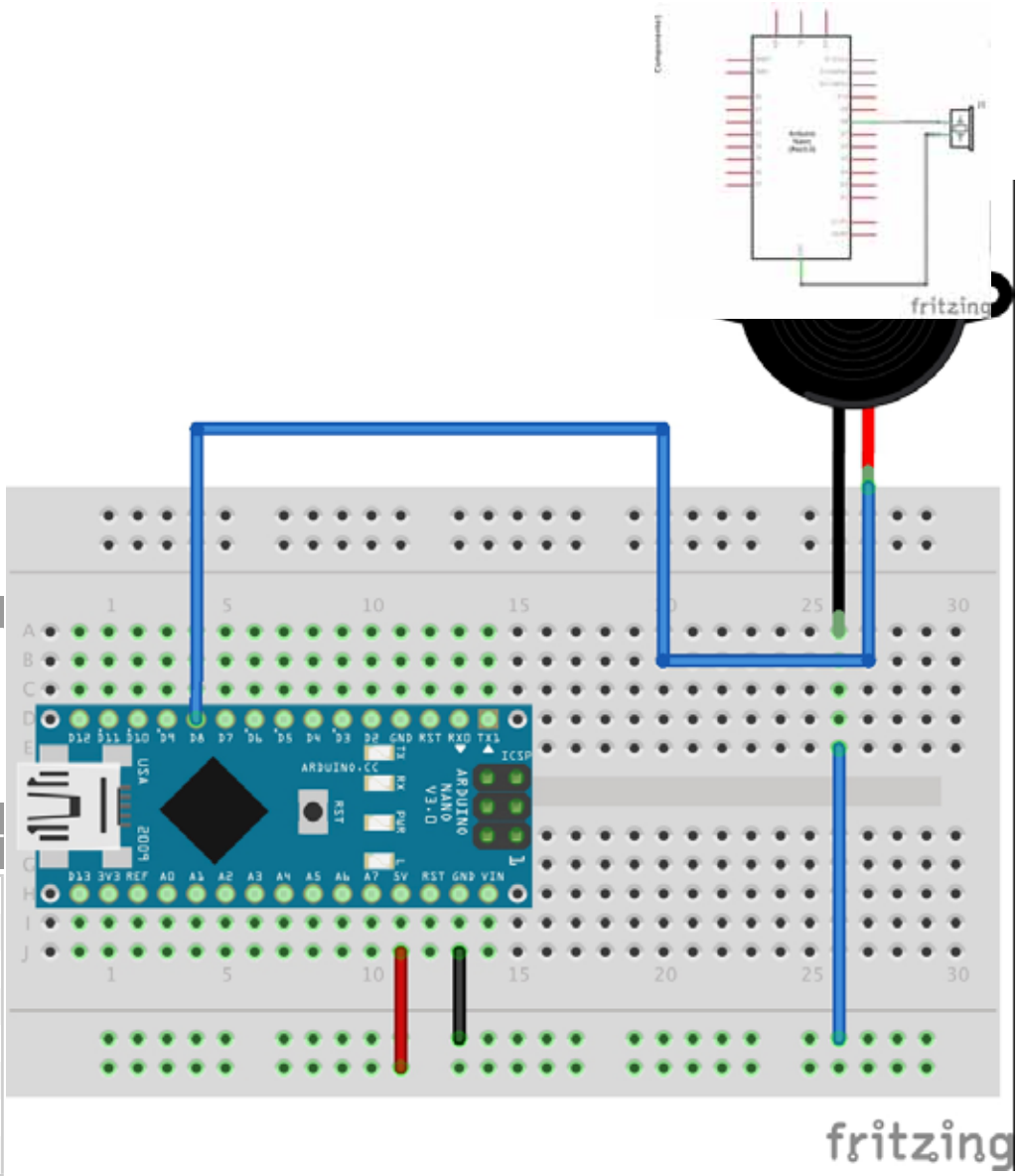
componenti		
nome	**valore**	**q.tà**
Arduino Nano v3		1
Basetta 400		1

8 leds - Array resistivo

```
**********************  primo esercizio:
8 leds collegati ai piedini da 2 a 10

int led,i;   // variabile per indicare su quale led si lavora

void setup() {
  for (i=2; i<11; i++) pinMode(i, OUTPUT);
  // tutte le porte in output
  led=10;
}

void loop() {
  digitalWrite(led, 0);
  led = led+1;
  if (led>10) led=2;
  digitalWrite(led, 1);
  delay(250);
}


**********************  secondo esercizio:


int led;   // variabile per indicare su quale led si lavora
int i;

void setup() {
  for (i=2; i<11; i++) pinMode(i, OUTPUT);
  // tutte le porte in output da 2 a 10
}

void loop() {
 led = random(2, 11);   // numero casuale da 2 a 10
 for (i=2; i<11; i++) {
   if (i < led) digitalWrite(i, 1);  // accendi
   else         digitalWrite(i, 0);  // spegni
  }
  delay(250);
}
```

Buzzer attivo

I buzzer o cicalini sono dei componenti che emettono un suono (segnale acustico).
Vi sono due tipi di componenti: attivi o passivi.
il buzzer attivo si comporta come un led, è polarizzato (un piedino va collegato al + l'altro alla massa) ed appena si alimenta emette una nota.
Il tono della nota è fissa e stabilita dal costruttore.
Sulla testa a volte troviamo un adesivo di carta, questo serve per protteggerlo durante il montaggio, durante il funzionamento va tolta.



librerie

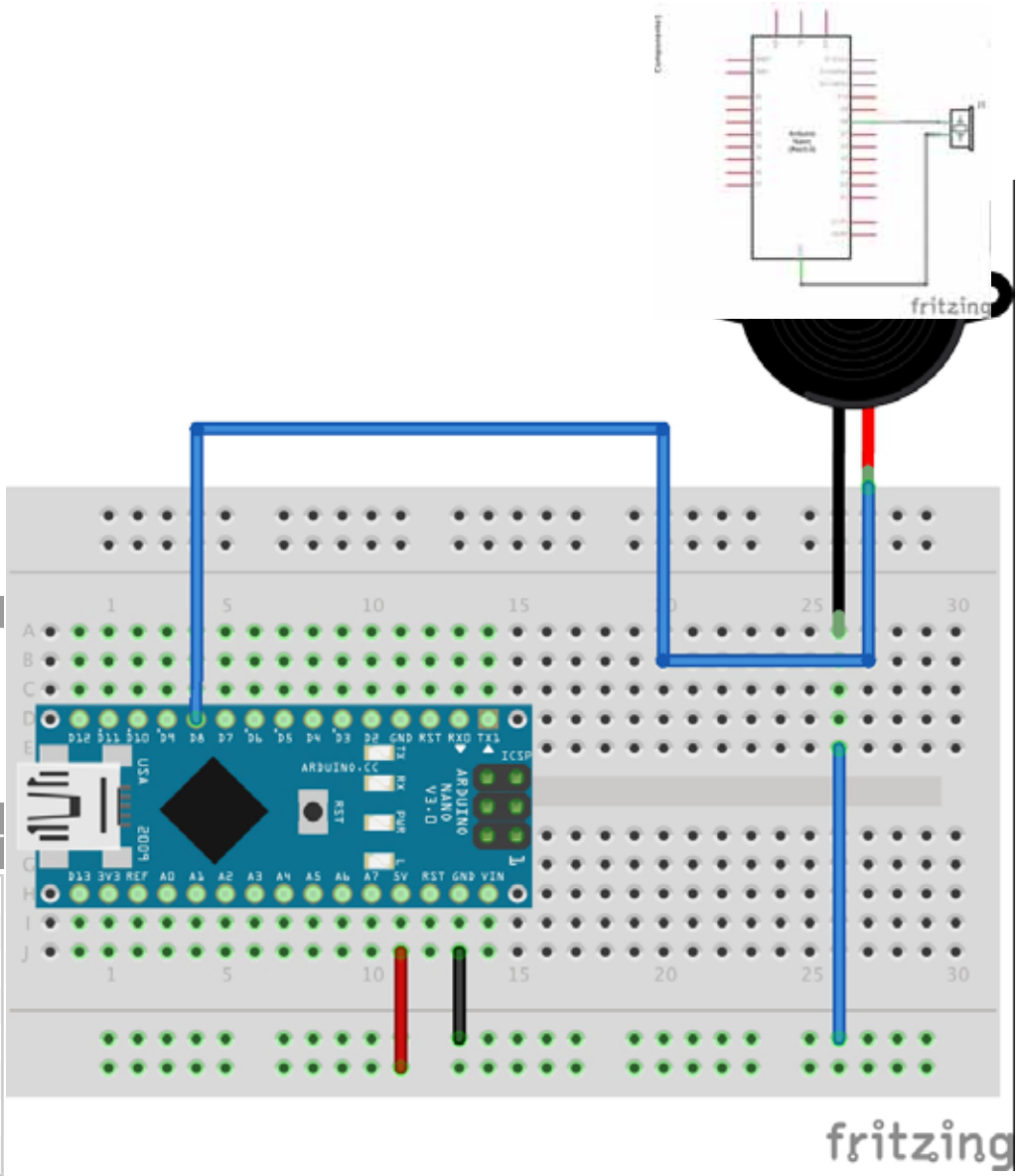
componenti

nome	valore	q.tà
Arduino Nano v3		1
Basetta 400		1
Buzzer Attivo		1

Buzzer attivo

```
void setup() {
  pinMode(8, OUTPUT);
}

void loop() {
  digitalWrite(8, HIGH);
  delay(1000);
  digitalWrite(8, LOW);
  delay(1000);
}
```

Buzzer Attivo - alfabeto morse



librerie

componenti

nome	valore	q.tà
Arduino Nano v3		1
Basetta 400		1
Buzzer Attivo		1

Buzzer Attivo - alfabeto morse

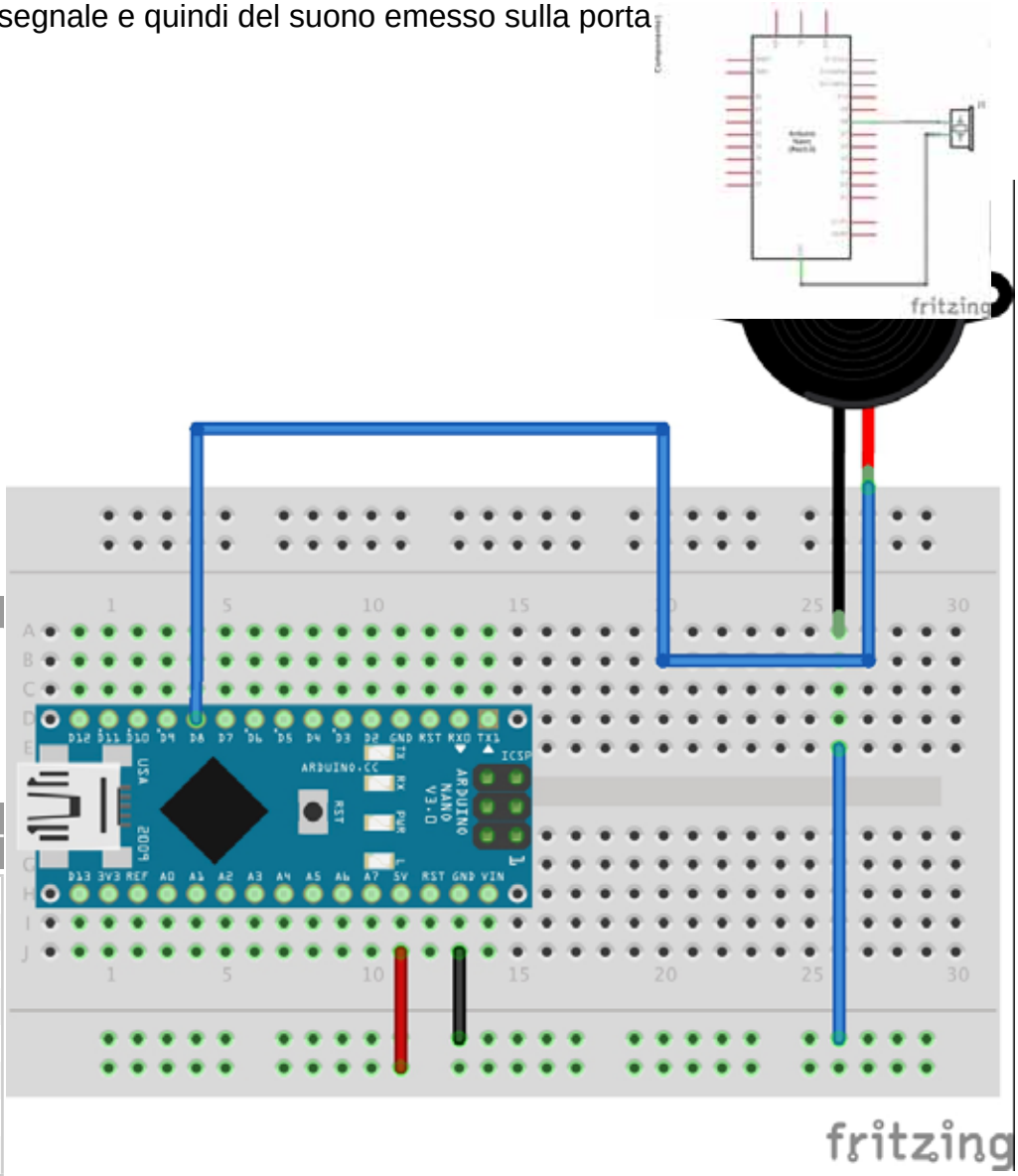
Buzzer passivo

Il Buzzer passivo si comporta come un altoparlante, emette un suono se riceve un segnale alternato. La nota che emette è proporzionale alla frequenza del segnale elettrico.
Lo si collega ad un piedino digitale, non occorre la resistenza.
Lo si distingue da quello attivo perchè il cilindretto generalmente è più corto e non ha un segno sulla circonferenza.

i comandi a disposizione sono:
tone(pin, tono);
genera sulla porta definita una frequenza espressa in Hz.
Questa nota continuerà dopo terminata l'istruzione.

tone(pin, tono, durata);
genera sulla porta definita una frequenza espressa in Hz.
Questa nota continuerà una durata specifica (in millisecondi).

noTone(pin);
interrompe l'emissione del segnale e quindi del suono emesso sulla porta



librerie

componenti

nome	valore	q.tà

Buzzer passivo

Input digitale - Pulsante

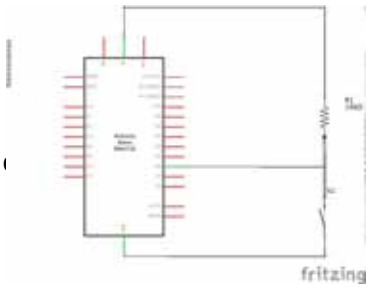
Possiamo comunicare con Arduino tramite pulsanti ed interruttori.
Arduino può leggere la condizione del pulsante (premuto o rilasciato) semplicemente leggendo il valore elettrico della porta (pin).
La porta impiegata deve essere configurata in modalità **INPUT** (lettura) ricorda che il pin accetta tensioni di
0 Volt quando è collegato a massa segnale basso **LOW**
5 Volt quando è collegato all'alimentazione: alto **HIGH**
non di deve MAI superare la tensione di alimentazione altrimenti di danneggia il processore.
Questo circuito impiega una resistenza (pull up) per mantenere il pin sempre allo stato alto. Se il pulsante viene premuto il pin va a livello basso.
Se non si collega la resistenza quando il pulsane non viene premuto il pin non è collegato a nulla e quindi il processore legge valori errati.

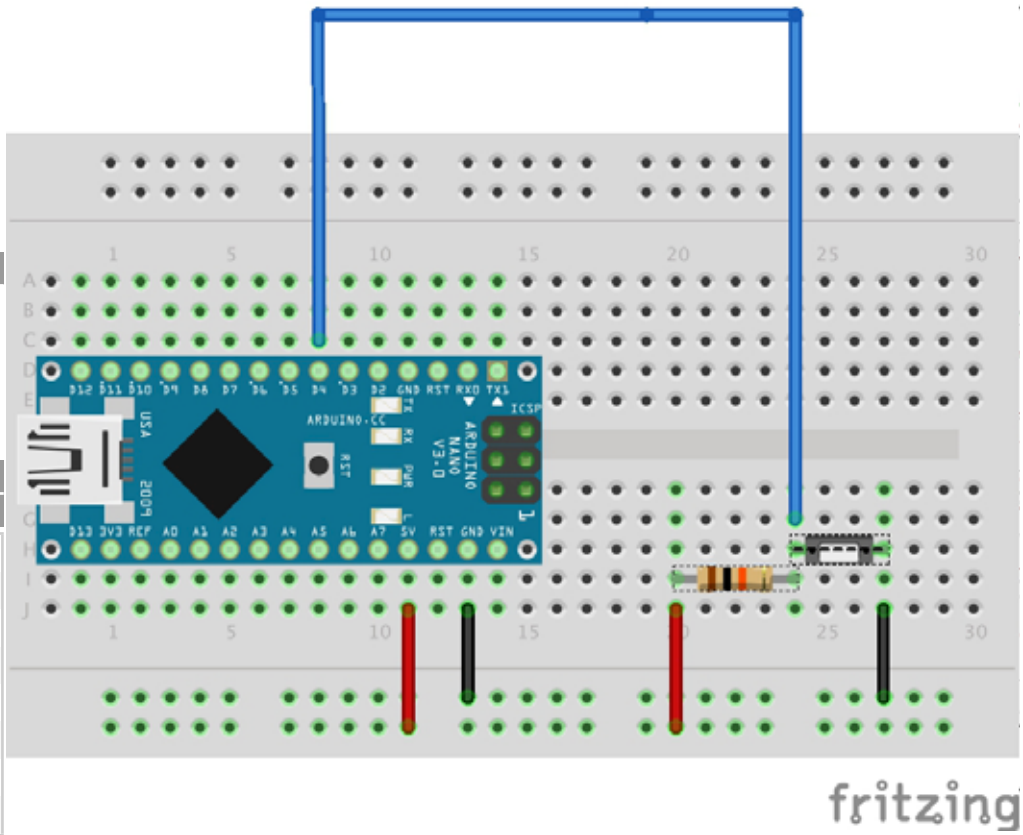
pinMode(10, INPUT);
configuriamo il piedino 10 in modalità input

pinMode(10, INPUT_PULLUP);
configuriamo il piedino 10 in modalità input ed attiviamo la resistenza di pull_up

a=digitalRead(10);
leggiamo lo stato del piedino 10 e mettiamo il risultato nella variabile "a"

if (a==1) { }
comando condizionale, le istruzioni contenute tra {} vengono eseguite solo se è vera la (tonde





librerie

componenti

nome	valore	q.tà
Basetta 400		1
Resistenza	10 KΩ	1
Pulsante		1
		1
		1

Input digitale - Pulsante

********** **primo esercizio:**

accendi il led interno (pin 13) quando è premuto il pulsante

```
int a;

void setup() {
  pinMode (13, OUTPUT);
  pinMode (4, INPUT_PULLUP);
}

void loop() {
  a= digitalRead(4);
  if (a==1) digitalWrite(13, 1);
  if (a==0) digitalWrite(13, 0);
  delay(500);
}
```

********** **secondo esercizio:**

```
int a;

void setup() {
  pinMode (13, OUTPUT);
  pinMode (4, INPUT_PULLUP);
}

void loop() {
  a= digitalRead(4);
  if (a==1) digitalWrite(13, 1);
  else      digitalWrite(13, 0);  // altrimenti fai....
  delay(500);
}
```

Input analogico - Potenziometro

Abbiamo lavorato con i segnali elettrici digitali: un pulsante aperto o chiuso, un led acceso o spento ecc. Questi segnali hanno due stati: 0 volt e 5 volt ma non hanno valori intermedi.
Vi è un'altra categoria di segnali che variano la loro tensione nel tempo: **i segnali analogici.**

Arduino può leggere questi segnali e li trasforma in un numero proporzionale alla tensione. Essi vengono trasformati in digitale da un convertitore chiamato "DAC"; quello di Arduino lavora a 12 bit cioè converte il segnale in un numero che va da 0 a 1023.

Il potenziometro è una resistenza variabile, ruotando il suo pomello si modifica la resistenza (da 0 ohm al massimo del potenziometro).
Esso ha tre piedini: i due laterali vanno collegati rispettivamente alla massa 0 volt e all'alimentazione 5 volt, quello centrale è l'uscita e avremo un segnale variabile tra 0 e 5.

Leggiamo i valori del potenziometro ed visualizziamo sul terminale i valori letti.
prova ad impiegare il **"plotter grafico"**.

Nel secondo esempio trasformiamo il valore letto da una scala 0..1023 ad una scala 0..10.
la funzione **map** ci semplifica il calcolo.

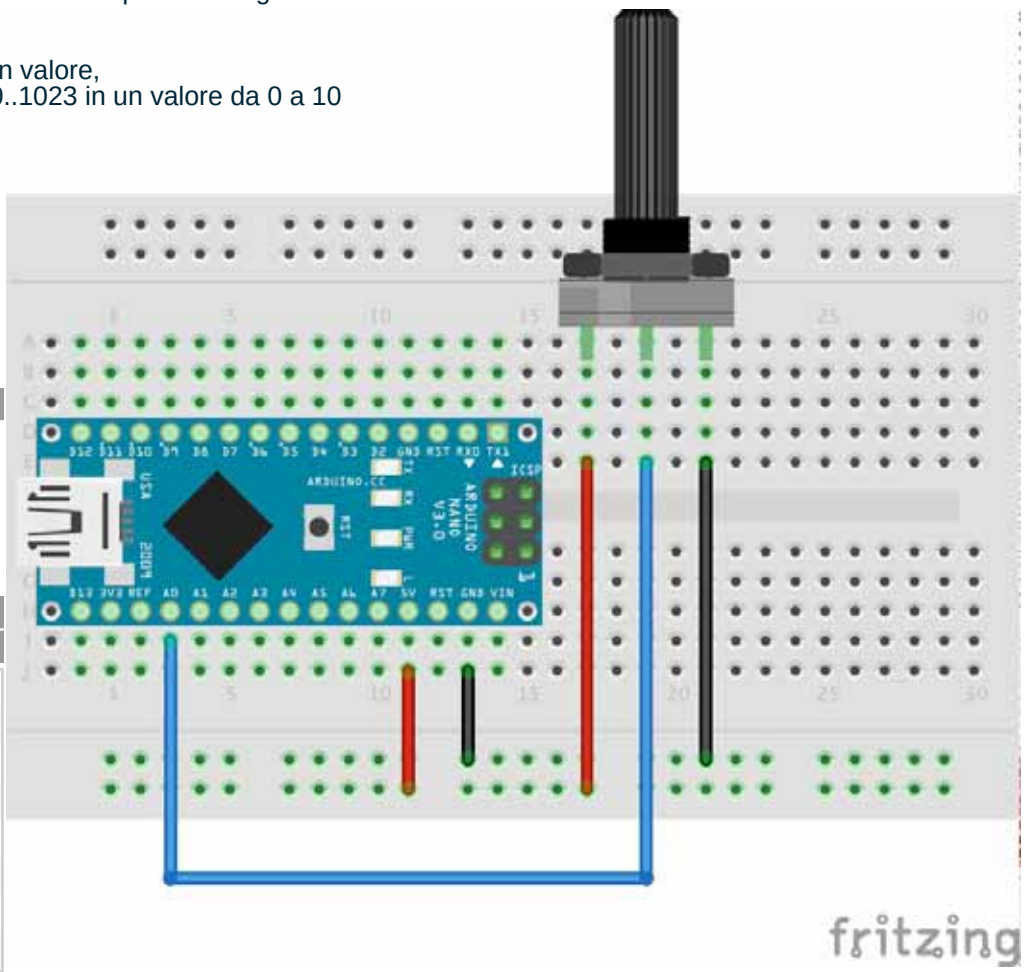
analogRead(A0)
funzione che restituisce il valore letto sulla porta analogica

map(valore, 0, 1023, 0, 10)
funzione che adatta la scala di un valore,
trasforma il "valore" dalla scala 0..1023 in un valore da 0 a 10

librerie

componenti

nome	valore		q.tà
Arduino Nano v3			1
Basetta 400			1
Potenziometro	10	KΩ	1



Input analogico - Potenziometro

********** **primo esercizio:**

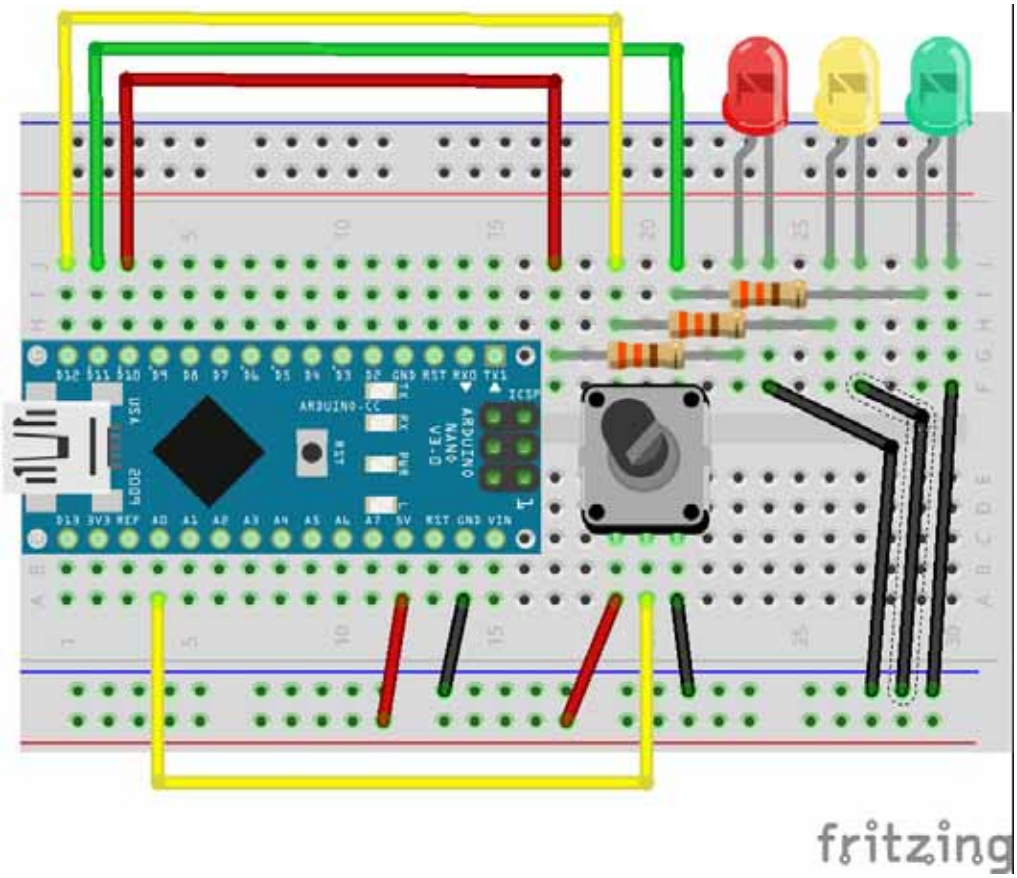
```
#define pot A0

void setup() {
  Serial.begin(57600);
}

void loop() {
  Serial.println(analogRead(pot));
  delay(250);
}

// fine programma
```

********** **secondo esercizio:**

```
#define pot A0
int valore;

void setup() {
  Serial.begin(57600);
}

void loop() {
  valore = analogRead(pot);
  Serial.println(map(valore, 0, 1023, 0, 10));
  delay(250);
}

// END
```

Potenziometro - tre led

Accendiamo un led in base alla posizione del potenziometro.
Rosso se inferiore a 200
Verde se superiore a 800
Giallo i valori intermedi.



librerie

componenti

nome	valore	q.tà
Arduino Nano v3		1
Basetta 400		1
Potenziometro		1
Led		3
Resistenza	330 Ω	3

Potenziometro - tre led

```
//     progetto: corso base - componenti
//  descrizione: accendere i leds in base alla posizione del potenziometro
//      shields: potenziometro
//         data: v1.1 02-02-2020


#define ledrosso  12
#define ledgiallo 11
#define ledverde  10
#define potenziometro   A0
int valore;

void setup() {
  pinMode(ledrosso,  OUTPUT);
  pinMode(ledgiallo, OUTPUT);
  pinMode(ledverde,  OUTPUT);
}

void loop() {
  valore = analogRead(potenziometro);
  if (valore < 200) {
    digitalWrite(ledrosso,  1);
    digitalWrite(ledgiallo, 0);
    digitalWrite(ledverde,  0);
  } else if (valore > 800) {
    digitalWrite(ledrosso,  1);
    digitalWrite(ledgiallo, 0);
    digitalWrite(ledverde,  1);
  } else {
    digitalWrite(ledrosso,  0);
    digitalWrite(ledgiallo, 1);
    digitalWrite(ledverde,  0);
  }
  delay(100);
}

// END
```