# Biblioduino

Dove partire
per imparare ad
usare Arduino

Biblioduino è nato nel 2014 all'interno di Hubout Makers Lab da un'idea di Andrea Mantelli, Giorgio Rancilio, Andrea Sottocornola.



Versione: 1.4

# COSA È

Biblioduino nasce dall'idea di rendere prestabile in biblioteca un kit comprendente Arduino e alcuni componenti facilmente utilizzabili per permettere a tutti di imparare ad usare questo strumento in maniera gratuita.

Biblioduino, proprio come un libro, può essere prenotato e ritirato in biblioteca da qualsiasi cittadino che sia iscritto ad una delle biblioteche del CSBNO e che possegga una +TECA.

Biblioduino è un progetto pensato per diffondere la cultura digitale, in particolare legata all'elettronica ed ai circuiti, in maniera semplice ed intuitiva.

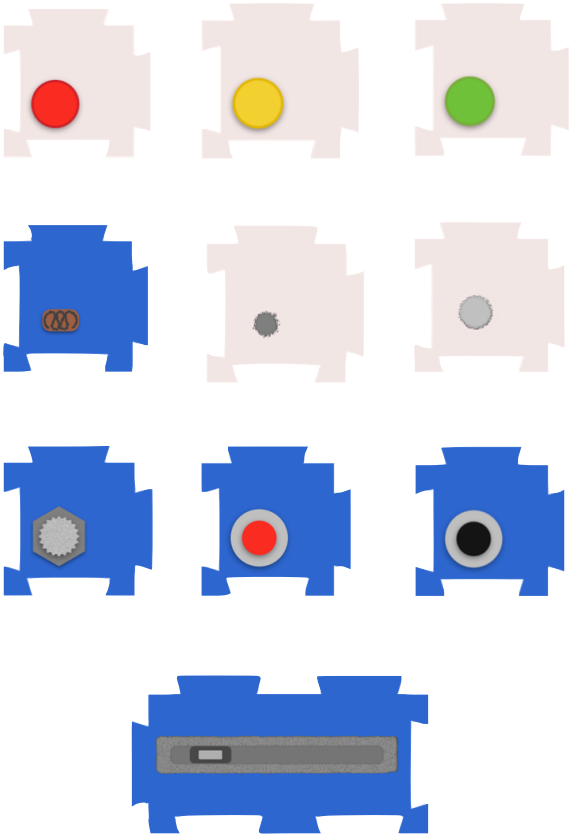
Sono presenti differenti kit che hanno al loro interno diverse tipologie di componenti (base, avanzato e meccanico), in modo da poter scegliere il kit in base al livello di competenza e all'utilizzo.

Ogni kit, oltre ai componenti, pensati affinché possano essere collegati direttamente alla scheda Arduino, contiene questo manuale che in modo semplice ed intuitivo può guidare l'utente nella conoscenza base dell'elettronica e della programmazione in C++.

Biblioduino è anche un esperimento di progettazione partecipata. Il kit è stato pensato e realizzato dai mentors dello spazio creativo Hubout Makers Lab, all'interno del Centro culturale il Pertini di Cinisello Balsamo (MI). Luogo, che vuol essere non solo ideatore del progetto, ma anche collettore di suggerimenti, consigli, dubbi, esperienze rispetto al suo utilizzo.

Vuoi migliorarlo? Vienici a trovare!

# I CUBETTI



***Arduino** è una **piattaforma open source** per il **physical computing** basata su una semplice **scheda input**/**output** a **microprocessore** e un **ambiente di sviluppo** dedicato.*

**Piattaforma**: insieme di componenti hardware (reali, che si possono toccare) e software (relativi alla programmazione, che non si possono toccare).

**Open source**: sono disponibili su internet i "sorgenti" sia dell'hardware e del software, chiunque può costruire il proprio Arduino e modificarlo come vuole.

**Physical computing**: interazione, comunicazione, relazione, contatto fra mondo digitale e mondo reale.

**Scheda**: circuito elettronico costituito da componenti saldati su di una basetta.

**Input**: connessione dal mondo reale verso il mondo analogico.

**Output**: interazione dal mondo digitale verso il mondo analogico.

**Microprocessore**: il cuore di Arduino, semplicemente esegue ciò che gli viene detto di fare.

**Ambiente di sviluppo**: programma che funziona su un computer che permette di dire ad Arduino che cosa fare e quando.

**Microprocessore**: è il cuore di Arduino. Esso è un circuito integrato, un componente elettronico composto da tanti componenti in un'unica parte, detta  chip, capace di fare conti e di eseguire un programma su di esso caricato. Presenta inoltre delle periferiche capaci di parlare con il mondo fisico: ingressi e uscite (Input/Output) analogiche e digitali, contatori, timer, etc.

**Tasto di reset**: premendolo permette di far ripartire il programma da capo.

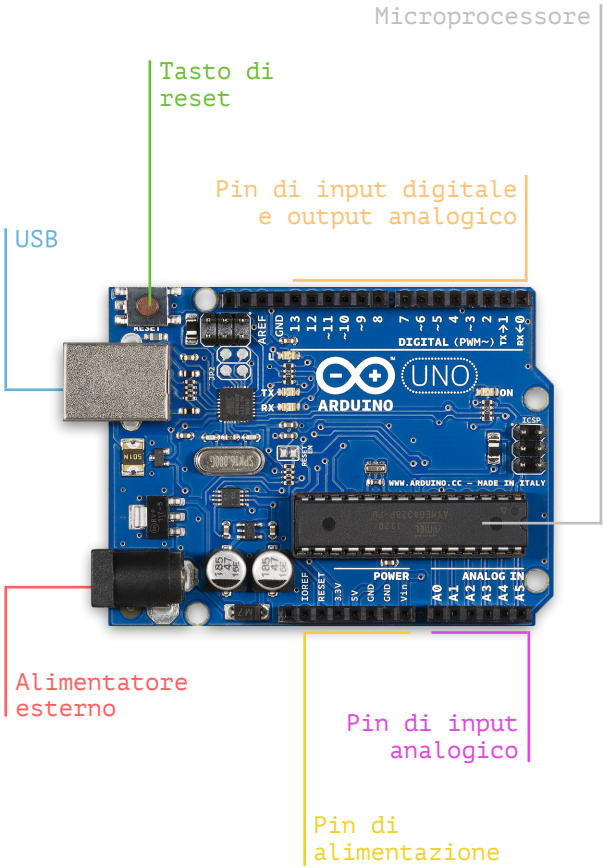
**USB**: porta con il quale Arduino parla con il computer, viene usata per scaricare il programma e permette anche di alimentare la scheda.

**Alimentatore esterno**: serve per alimentare la scheda senza computer collegato si usa questo connettore.

**Pin di input output digitale**: dove si vanno a collegare le periferiche digitali (pin derivante da spillo, a indicare la piccolezza dei piedini usati).

**Pin di input analogico**: dove si collegano, invece, i sensori analogici.

**Pin di Alimentazione**: queste uscite servono ad alimentare dei circuiti esterni, viene usato per tutti i collegamenti con sensori e attuatori.


Microprocessore
Tasto di reset
Pin di input digitale e output analogico
USB
Alimentatore esterno
Pin di input analogico
Pin di alimentazione

# UN PO' DI ELETTRONICA

Arduino è una scheda elettronica, bisogna quindi conoscere i principi su cui essa funziona per saperlo utilizzare. L'elettronica si basa sul movimento degli **elettroni**, particelle fondamentali che compongono la materia. Essi all'interno dei circuiti elettronici vengono messi in moto da un **generatore**, come ad esempio una batteria o un alimentatore. Quando essi passano all'interno dei vari **componenti** (tutto ciò che è presente nel kit Biblioduino, luci, motori, altoparlanti… ) gli danno vita facendoli funzionare.

Per funzionare un qualsiasi circuito deve essere chiuso, in modo tale che gli elettroni possano partire da un lato del generatore, completare il loro giro e finire nell'altro capo del generatore. Gli elettroni devono però scorrere nel verso giusto, per questo i capi dei generatori e dei componenti sono contraddistinti da un lato positivo **+** (rosso) e un lato negativo **-** (nero). Nel momento in cui un circuito viene aperto, con un **interruttore** o con un **tasto**, gli elettroni non possono più passare e i componenti ad esso collegati si addormentano, non funzionando più.

**Alcuni semplici regole per utilizzare questo kit:**

1 - Non connettere tra loro pin di colore diverso

2 - Non mettere oggetti metallici a contatto con Arduino

3 - Avere sempre le mani asciutte quando si utilizza il kit

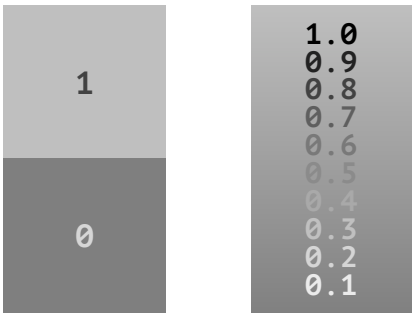
4 - Non usare componenti non presenti nel kit

# ANALOGICO VS DIGITALE

Un concetto fondamentale nel mondo dell'elettronica ma più in generale nella tecnologia è la distinzione tra analogico e digitale.

Qualcosa di **digitale** è rappresentato solamente da due valori: 1 o 0, alto o basso, on o off, accesso o spento. I dati espressi in questo modo sono facilmente utilizzabili all'interno dei computer e dei microprocessori.

Diversamente qualcosa di **analogico** può assumere tanti valori differenti, compresi tra un massimo e un minimo. Questi valori non si possono utilizzare direttamente nei processori e nei computer. Essi devono essere **digitalizzati**, trasformati quindi in una sequenza di valori digitali (composta da 1 e 0), per poter essere elaborati da un microprocessore o da un computer.



# INPUT VS OUTPUT

I pin sono i mezzi che permettono ad Arduino di scambiare informazioni con l'esterno. Essi funzionano in ambo le direzioni: entrata o **input**, uscita o **output**. Ad essi vengono collegati i diversi componenti in base a ciò che si vuole costruire. I componenti sono divisi in due grandi categorie: sensori e attuatori.

I **sensori** permettono di portare delle grandezze dal mondo reale al mondo digitale. Esempi di questi componenti sono i tasti e gli interruttori, i sensori di temperatura e di luce… I tasti e gli interruttori sono esempi di sensori digitali, invece i sensori di temperatura e luce sono analogici. Tutti i sensori devono essere collegati a un pin di input, in quanto funzionano in entrata rispetto ad Arduino. Nel kit biblioduino tutti i sensori sono scatoline di colore **blu**.

Gli **attuatori** permettono invece la via opposta, portano informazioni dal mondo digitale a quello reale, attuano quindi nella realtà un valore digitale. Esempi di questi sono le luci o i LED, gli altoparlanti, i motori… La maggior parte degli attuatori possono essere utilizzati sia in modo analogico che digitale. Per funzionare devono essere collegati a dei pin di output di Arduino. In questo kit tutti i sensori sono di colore **bianco**.

# QUALCHE RICETTA DI INFORMATICA

L'informatica si potrebbe studiare senza neanche toccare un computer. Questa scienza studia come creare dei procedimenti, chiamati **algoritmi**, in grado di risolvere determinati problemi.
Gli algoritmi sono una serie di comandi, istruzioni messe con ordine una dopo l'altra che permettono di descrivere in modo preciso il procedimento da seguire. Ogni istruzione viene eseguita solamente se prima è stata eseguita quella scritta precedentemente.

Ciò è molto simile a quanto accade per una ricetta di cucina, nella quale si descrive un procedimento preciso per cucinare un piatto. Esso è descritto mediante una serie di passi che devono essere necessariamente eseguiti uno dopo l'altro con ordine, senza possibilità di saltarne alcuni o andare in ordine casuale.

Dire ad Arduino cosa deve fare è come scrivere una ricetta, bisogna però utilizzare un modo di scrivere preciso in modo tale che Arduino possa capirlo. Il linguaggio che Arduino capisce bene è il **C++**, con esso non si possono comunicare le stesse cose che si dicono in Italiano, ma si riescono a descrivere bene dei procedimenti precisi da seguire.

L'informatica di Arduino consiste quindi in una serie di **funzioni**, delle azioni che possono essere eseguite, e di **comandi di controllo**, che determinano l'ordine con cui devono essere eseguite le diverse azioni.

Nell'informatica di Arduino tutti i comandi vengono scritti in quelli che si chiamano programmi o in gergo **sketch**.

La ricetta da scrivere ad Arduino per dirgli che cosa deve fare è divisa in due grandi parti: il **setup** e il **loop**. Tutte le istruzioni che vengono scritte all'interno delle due parentesi graffe { } dopo la dicitura `void setup()` vengono eseguite dopo l'accensione di Arduino, o dopo aver premuto il tasto RESET, una sola volta.

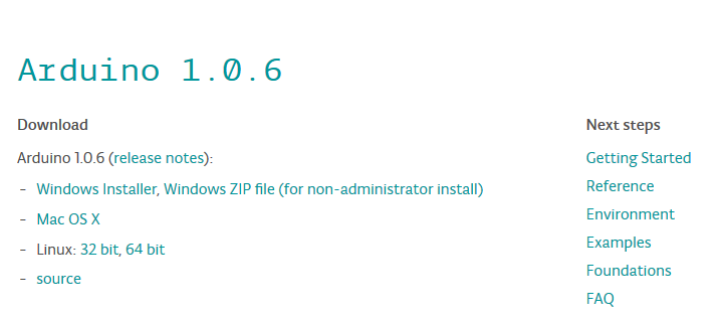
Diversamente le istruzioni all'interno delle parentesi graffe dopo `void loop()` vengono eseguite dopo quelle presente nel setup in modo continuo: quando si arriva in fondo si ricomincia da capo fino a quando Arduino non viene spento staccandogli la spina.
All'interno di setup e loop si possono inserire tutte le funzioni che si vuole, senza limiti.

Tutto ciò che è preceduto da // è un commento, quindi non sarà preso in considerazione da Arduino. I commenti servono, in genere, per annotazioni o spiegazioni riguardanti il codice se ad esempio deve essere letto da un'altra persona.

```
//Commento

void setup()
{

}

void loop()
{

}
```

L'*Integrated Development Environment* (IDE) è un programma, molto simile ad un semplice editor di testo, che mette a disposizione quanto necessario a programmare un microprocessore. E' quindi il modo che abbiamo per scrivere il nostro sketch con descritto cosa Arduino deve fare durante il suo funzionamento.

Prima cosa da fare è scaricare il programma andando sul sito: http://arduino.cc/en/Main/Software

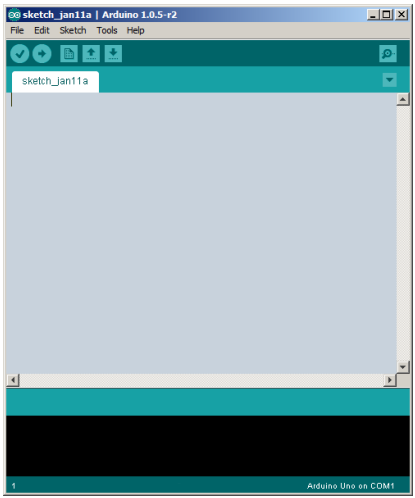



Bisogna quindi selezionare il proprio sistema operativo, scaricare il programma ed installarlo.

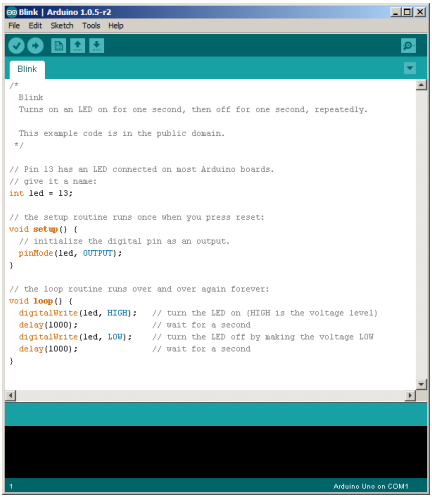
Per problemi relativi all'installazione o alla configurazione si faccia riferimento alla guida ufficiale di Arduino al seguente link: http://arduino.cc/en/Guide/HomePage

# IL PRIMO SKETCH

Una volta installato, aprendo il programma ci si trova davanti ad una finestra simile a questa:




Siamo ora pronti per il primo programma. Colleghiamo la scheda al computer con il cavo USB. Per provare che tutto funzioni bene carichiamo uno sketch di esempio: il blink. Questo si trova nel menu File -> Examples -> 01. Basic -> Blink.



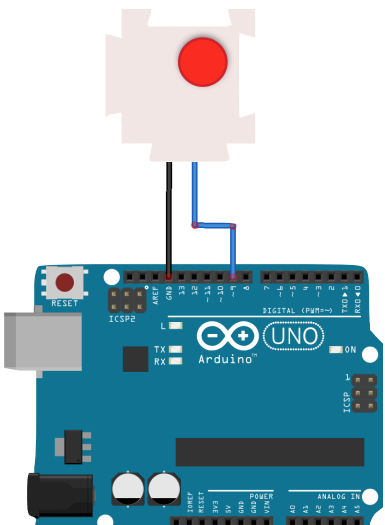

Prima di caricare il nostro programma dobbiamo selezionare la scheda corretta da Tools -> Board selezioniamo Arduino Uno. Selezioniamo poi la porta seriale utilizzata da Tools -> Serial Port, in genere viene indicato di fianco alla porta corretta (Arduino Uno).

Premiamo ora  . Inizierà la compilazione e lo scaricamento dello sketch su Arduino. Se tutto sarà andato a buon fine in basso a sinistra verrà visualizzato "Caricamento terminato" e una luce su Arduino, vicina al pin 13, inizierà a lampeggiare 1 secondo acceso e uno spento.

Dal menu File si possono salvare e caricare gli sketch che vengono scritti con estensione .ino

# ACCENDERE UN LED

Un **led** (Lighting Emitting Diode) è un componente elettronico che funziona come una lampadina: quando gli elettroni passano al suo interno genera luce. Esso è quindi il più semplice attuatore che si possa utilizzare con Arduino. Nel kit vi sono led di diversi colori. Nel primo esempio vogliamo comandare il led in modo tale che esso sia accesso o spento. Dobbiamo quindi prendere il cubetto del led e collegare il filo **blu**, indicante un segnale digitale, ad un pin digitale (ad esempio il pin 9), il filo **nero** al pin GND. Per trovare i pin giusti possono aiutare i colori delle etichette sui diversi pin di Arduino.
Abbiamo quindi creato l'hardware (tutto ciò che prevede il montaggio delle parti elettroniche e fisiche) del nostro primo esempio con Arduino.



E’ ora necessario scrivere il software (lo sketch, la ricetta che dice ad Arduino cosa deve fare).

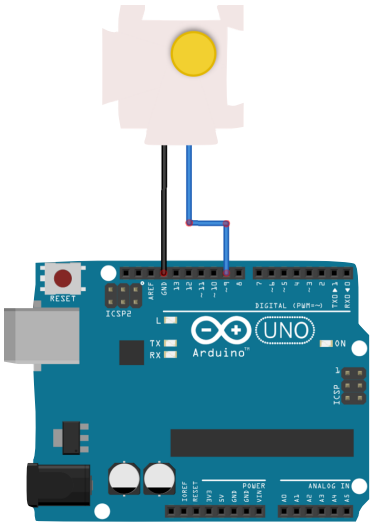
```
void setup()
{
    pinMode(9, OUTPUT);
}

void loop()
{
    digitalWrite(9, HIGH);
}
```

La funzione pinMode(pin, INPUT/OUTPUT), inserita all’interno del setup, dice ad Arduino che utilizzerà il pin scritto dopo (in questo caso il 9) come pin di input o output, quindi ingresso o uscita. Per usare il led, essendo un attuatore, è necessario che il pin a cui esso è collegato sia di output, in modo tale da poterlo comandare.

Nel loop c’è invece la funzione digitalWrite(pin, HIGH/LOW) scrive, quindi imposta il valore del pin, un valore digitale high o low, alto o basso, accesso o spento. Scrivendo quindi il valore HIGH decido di mettere alto il pin 9 e quindi di far accendere il led ad esso collegato.

# LED LAMPEGGIANTE

Volendo far lampeggiare il led l'hardware dell'esempio rimane invariato rispetto al precedente.




Bisogna semplicemente modificare il software, inserendo la funzionalità di accensione spegnimento controllato.

```
void setup()
{
    pinMode(9, OUTPUT);
}

void loop()
{
    pinMode(9, HIGH);

    delay(1000);

    digitalWrite(9, LOW);

    delay(1000);

}
```

La funzione delay(tempo) attende un determinato tempo (delay in inglese significa ritardo), espresso in millisecondi (1000 millisecondi fanno 1 secondo) senza fare niente. In questo caso il led viene accesso, si aspetta 1 secondo, si spegne il led, si aspetta un altro secondo e si ricomincia da capo il loop dello sketch. Si noti come l'ultima attesa sia necessaria per far funzionare bene il programma, poiché nel momento in cui si ricomincia il loop il passaggio dalla fine all'inizio del loop è istantaneo. Se non ci fosse il tempo di attesa il led si spegnerebbe e si accenderebbe senza una pausa di mezzo, non si vedrebbe quindi l'effetto in quanto troppo veloce.

# CASSETTI INFORMATICI

In informatica si utilizzano dei particolari cassetti per mettere via i dati: le **variabili**. All'interno di questi oggetti si possono salvare i dati che vengono utilizzati all'interno di un programma.

La prima cosa da fare per utilizzare un variabile è dire ad Arduino che la variabile esiste, creando quindi un cassetto. Per fare ciò è necessario scrivere, al di fuori di setup e loop, la seguente istruzione:

```
int var;
```

All'interno del nostro cassetto, che si chiama var, è possibile metterci dentro qualsiasi numero si voglia, ad esempio:

```
var = 5;
```

All'interno del cassetto var abbiamo quindi messo il valore 5. Posso ora incrementare la variabile var di 10 valori, mettendo all'interno della variabile var la stessa variabile a cui viene sommato il valore 10:

```
var = var + 10;
```

Il valore all'interno delle variabili può essere utilizzato in qualsiasi punto del programma al posto dei numeri. Ad esempio si può usare la funzione delay con all'interno una variabile in cui è salvato un valore invece che direttamente il valore numerico:

```
delay(var);
```

Una volta messo qualcosa in un cassetto spesso bisogna guardare dentro per capire che cosa ci abbiamo messo dentro. In informatica si usa quindi quello che si chiama costrutto if (in italiano *se*) per verificare alcune condizioni su un valore salvato all'interno di una variabile. Ciò può essere utile quando all'interno della variabile è stato messo un valore che arriva dall'esterno, ad esempio da un sensore, e quindi non sappiamo quanto valga. L'if si presenta in questo modo:

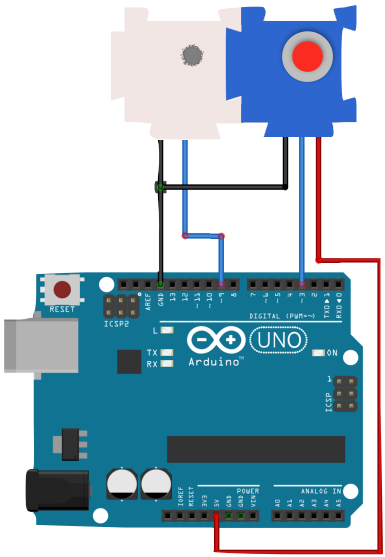
```
if (var == 0)
{

}
else
{

}
```

L'esecuzione di questo pezzo di programma parte con il controllo di ciò che è scritto all'interno delle parentesi tonde. Qui va messa una condizione, se è vera allora vengono eseguite tutte le istruzioni all'interno delle prime parentesi graffe dopo `if` e non vengono considerate le istruzioni all'interno delle seconde parentesi graffe dopo `else` (in italiano *oppure*). Diversamente se la condizione è falsa vengono eseguite le istruzioni all'interno delle seconde graffe, le quali è possibile omettere se si vuole che Arduino non faccia niente in caso di condizione falsa. In questo caso la condizione `var == 0` verifica che la variabile var sia uguale a 0. Al posto del doppio uguale si possono utilizzare tutta una serie di simboli diversi, in gergo operatori, che permettono di fare diversi confronti, tra i quali:

```
== uguale     > maggiore di    >= maggiore o uguale di
!= diverso    < minore di      <= minore o uguale di
```

# TASTO E BEEP

In questo esempio utilizziamo il primo sensore: un tasto. Alla pressione del tasto verrà attivato un **beeper**, un componente che inizierà a suonare quando alimentato. Il filo **rosso** del beeper va collegato al pin 5V.



```
int val;

void setup()
{
    pinMode(9, OUTPUT);
    pinMode(3, INPUT);
}

void loop()
{
    val = digitalRead(3);

    if(val == LOW)
    {
        digitalWrite(9, HIGH);
    }
    else
    {
        digitalWrite(9, LOW);
    }
}
```

Nello sketch si dichiara la variabile *var* al di fuori di setup e loop. Per utilizzare un sensore bisogna impostare il pin a cui esso è connesso come pin di INPUT con la funzione pinMode. La funzione digitalRead(pin) legge un valore digitale da un determinato pin impostato come input; riporta quindi un valore HIGH o LOW e lo mette nella variabile val. Quando il tasto non è chiuso il valore è HIGH, mentre quando è premuto diventa LOW. Si utilizza poi l'if per verificare il contenuto della variabile, se è LOW mette HIGH il pin a cui è collegato il beeper accendendolo, diversamente lo mette LOW spegnendolo.

# CICLO FOR

Un altro costrutto importante in informatica è il ciclo for (in italiano *per*), che permette di eseguire tante volte delle istruzioni tenendo il conto del numero di volte che le si esegue e mettendo questo numero in una variabile di solito chiamata `i`.

```
int i;

for(i=0; i<100; i++)
{
    analogWrite(9, i);
    delay(5);
}
```

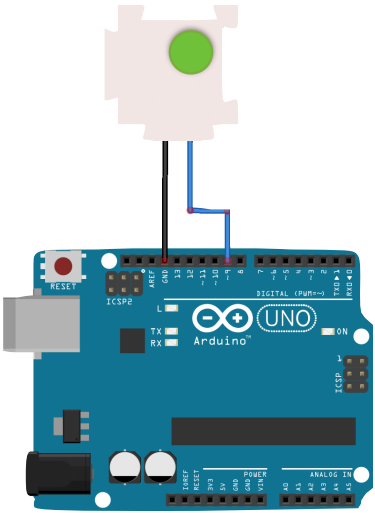
Per usare il for bisogna dapprima dichiarare la variabile i. All'interno dello sketch, dopo la parola `for`, bisogna, in ordine e tra le parentesi separate da punto e virgola: dare un **valore iniziale** alla variabile i, in questo caso `i=0`; fornire una **condizione di uscita** dal ciclo, qui `i<100`; fornire un **incremento** che viene eseguito alla fine di ogni ciclo, sopra `i++` che equivale a `i = i + 1`. Il ciclo incomincia dando il valore iniziale alla variabile, i parte quindi dal valore `0`. Viene poi controllata la condizione di uscita, se vera si procede nell'eseguire le istruzioni presenti dentro le parentesi graffe. Nel caso del primo ciclo `0<100` e quindi la condizione è vera. Una volta completate le istruzioni interne al ciclo si esegue l'incremento, in questo caso si somma 1 alla variabile `i`, che assume quindi il valore 1.

Il ciclo ricomincia quindi da capo senza ridare il valore iniziale, ma partendo subito controllando la condizione di uscita. Essendo `i` uguale a 1 la condizione è ancora vera, vengono quindi eseguite tutte le istruzioni interne ed alla fine viene sommato ad `i` un'altra volta 1, il valore della variabile diventa quindi 2. Tutto questo continua in modo ciclico fino a quando la variabile `i` non assume il valore `100`, momento in cui la condizione di uscita è falsa e quindi si esce dal ciclo proseguendo nel programma. Dall'inizio al momento in cui si esce dal ciclo le istruzioni contenute nel suo interno vengono eseguite esattamente 100 volte.

La funzione principale di questo costrutto non è tanto quella di ripetere molte volte delle funzioni ma quanto quella di poter ripeterle utilizzando una variabile che incrementa man mano che si eseguono le stesse istruzioni. Nel prossimo esempio si vedrà un'applicazione di ciò.

# FADE LED

In questo altro esempio vediamo come accendere un led lentamente, comandandolo con un valore analogico.




Arduino può utilizzare alcuni pin digitali come output analogici. Ciò sembra un controsenso ma si può ingannare l'elettronica comandando un pin digitale in modo molto veloce, riuscendo quindi ad avere un output analogico da un pin digitale. Solamente i pin digitali con affianco il simbolo **~**.

La funzione `digitalWrite(pin, valore)` permette di impostare il valore analogico del pin, che deve essere compreso tra `0` e 255.

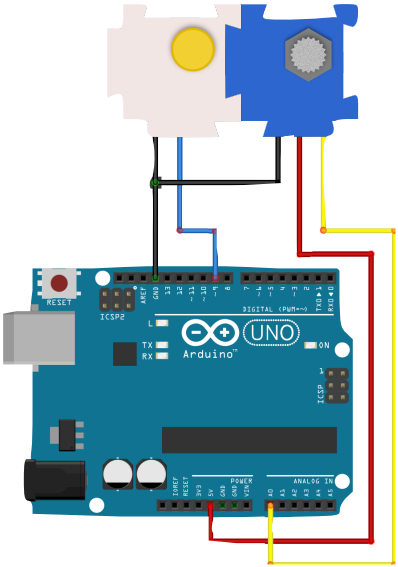
```
int i;

void setup()
{
    pinMode(9, OUTPUT);
}

void loop()
{
    for(i=0; i<255; i++)
    {
        analogWrite(9, i);
        delay(5);
    }
}
```

Nell'esempio si utilizza un ciclo `for` che parte da `i=0` ed esegue 255 volte le stesse istruzioni. Ad ogni giro si imposta il nuovo valore analogico sul pin con la funzione `digitalWrite`. In particolare si scrive il valore della variabile `i` che ad ogni giro del ciclo aumenterà il suo valore di 1. Inoltre prima di finire un giro si aspettano 5 millisecondi. Il risultato di ciò è un led che si accende piano piano, all'aumentare del valore della variabile `i`. Aumentando il tempo di attesa tra un giro e l'altro del for si può aumentare il tempo con cui il led si accende.

# LED COMANDATO

Per completare le funzionalità di Arduino manca vedere un esempio in cui si utilizza un input analogico, utilizzando quindi i pin A0, A1, etc. e i fili colorati di **giallo**. In questo caso utilizziamo un **potenziometro**: un sensore che permette di regolare quanti elettroni far passare nel circuito a cui è collegato, permettendo quindi ad Arduino di acquisire un valore analogico. Nel kit sono compresi due tipi di potenziometri: quello circolare e quello lineare. Entrambi funzionano in modo uguale e si possono utilizzare in questo esempio.

```
int pot;
int tempo;

void setup()
{
    pinMode(9, OUTPUT);
    pinMode(A0, INPUT);
}

void loop()
{
    pot = analogRead(A0);

    tempo = (pot*100)/1000 + 1;

    digitalWrite(9, HIGH);
    delay(tempo);
    pinMode(9, LOW);
    delay(tempo);
}
```




In questo esempio sono necessarie due variabili pot e tempo che vengono dichiarate prima di setup e loop. Nel setup si imposta il pin A0 come INPUT sempre con la solita funzione pinMode. Nel loop invece vediamo subito la funzione analogRead(pin) che permette di leggere un valore analogico da un pin e metterlo nella variabile che viene messa prima, in questo caso dentro pot. Arduino è capace di leggere dei valori analogici che vanno da 0 a 1023. La strana istruzione tempo = (pot*100)/1000 + 1 non è che una formula matematica molto semplice, una proporzione tra il valore di pot (che va da 0 a 1024) e il valore di tempo. Questo calcolo permette quindi di adattare il valore analogico che viene acquisito a come questo valore viene utilizzato dentro lo sketch, in questo caso come valore del tempo. Provando a cambiare questi numeri ci si rende conto di come si può modificare la "scala" dei valori. Dopo di ciò si accende e si fa lampeggiare il led comandato per un intervallo in millisecondi pari al valore della variabile tempo.

# DI COSA È FATTA LA MUSICA

Un qualsiasi suono è una vibrazione dell'aria ad una certa velocità che viene chiamata **frequenza** e si misura in *Hertz* [Hz]. A frequenze differenti corrispondo suoni differenti. Anche le note musicali sono dei suoni e ad ognuna di esse corrisponde una determinata frequenza. Si riportano qui i valori di frequenza delle principali note musicali:

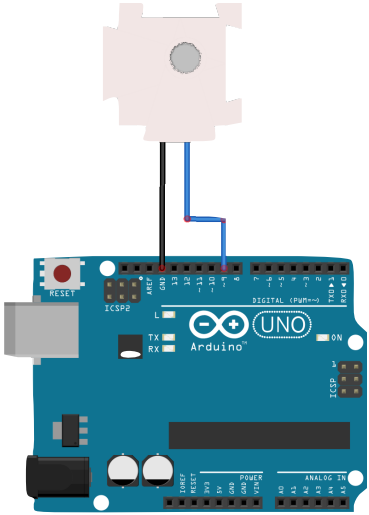
| Nota | Frequenza | Nota | Frequenza | Nota | Frequenza |
|---|---|---|---|---|---|
| DO0 | 15 Hz | DO1 | 33 Hz | DO2 | 65 Hz |
| DO#0 | 17 Hz | DO#1 | 35 Hz | DO#2 | 69 Hz |
| RE0 | 18 Hz | RE1 | 37 Hz | RE2 | 73 Hz |
| RE#0 | 19 Hz | RE#1 | 39 Hz | RE#2 | 77 Hz |
| MI0 | 21 Hz | MI1 | 41 Hz | MI2 | 82 Hz |
| FA0 | 22 Hz | FA1 | 44 Hz | FA2 | 87 Hz |
| FA#0 | 23 Hz | FA#1 | 46 Hz | FA#2 | 92 Hz |
| SOL0 | 25 Hz | SOL1 | 49 Hz | SOL2 | 97 Hz |
| SOL#0 | 26 Hz | SOL#1 | 52 Hz | SOL#2 | 103 Hz |
| LA0 | 28 Hz | LA1 | 55 Hz | LA2 | 110 Hz |
| LA#0 | 29 Hz | LA#1 | 58 Hz | LA#2 | 116 Hz |
| SI0 | 31 Hz | SI1 | 62 Hz | SI2 | 123 Hz |

| Nota | Frequenza | Nota | Frequenza | Nota | Frequenza |
|---|---|---|---|---|---|
| DO3 | 131 Hz | DO4 | 262 Hz | DO5 | 523 Hz |
| DO#3 | 139 Hz | DO#4 | 277 Hz | DO#5 | 554 Hz |
| RE3 | 147 Hz | RE4 | 294 Hz | RE5 | 587 Hz |
| RE#3 | 156 Hz | RE#4 | 311 Hz | RE#5 | 622 Hz |
| MI3 | 165 Hz | MI4 | 330 Hz | MI5 | 659 Hz |
| FA3 | 175 Hz | FA4 | 349 Hz | FA5 | 698 Hz |
| FA#3 | 185 Hz | FA#4 | 370 Hz | FA#5 | 740 Hz |
| SOL3 | 196 Hz | SOL4 | 392 Hz | SOL5 | 784 Hz |
| SOL#3 | 208 Hz | SOL#4 | 415 Hz | SOL#5 | 831 Hz |
| LA3 | 220 Hz | LA4 | 440 Hz | LA5 | 880 Hz |
| LA#3 | 233 Hz | LA#4 | 466 Hz | LA#5 | 932 Hz |
| SI3 | 247 Hz | SI4 | 494 Hz | SI5 | 988 Hz |

| Nota | Frequenza | Nota | Frequenza | Nota | Frequenza |
|---|---|---|---|---|---|
| DO6 | 1046 Hz | DO7 | 2093 Hz | DO8 | 4186 Hz |
| DO#6 | 1109 Hz | DO#7 | 2217 Hz | DO#8 | 4435 Hz |
| RE6 | 1175 Hz | RE7 | 2349 Hz | RE8 | 4699 Hz |
| RE#6 | 1245 Hz | RE#7 | 2489 Hz | RE#8 | 4978 Hz |
| MI6 | 1319 Hz | MI7 | 2637 Hz | MI8 | 5274 Hz |
| FA6 | 1397 Hz | FA7 | 2794 Hz | FA8 | 5587 Hz |
| FA#6 | 1480 Hz | FA#7 | 2960 Hz | FA#8 | 5920 Hz |
| SOL6 | 1568 Hz | SOL7 | 3136 Hz | SOL8 | 6272 Hz |
| SOL#6 | 1661 Hz | SOL#7 | 3322 Hz | SOL#8 | 6645 Hz |
| LA6 | 1760 Hz | LA7 | 3520 Hz | LA8 | 7040 Hz |
| LA#6 | 1865 Hz | LA#7 | 3729 Hz | LA#8 | 7459 Hz |
| SI6 | 1976 Hz | SI7 | 3951 Hz | SI8 | 7902 Hz |

# PICCOLO ARDUINO MUSICISTA

Per far suonare Arduino si può collegare il cubetto **speaker** ed usare la funzione `tone(pin, frequenza, durata)`. Per far suonare le diverse note si possono utilizzare le frequenze riportate nelle pagine precedenti. Per fare delle pause si possono sempre utilizzare delle funzioni delay.



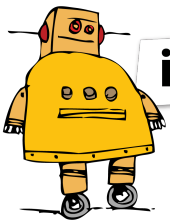
```
void setup()
{
    pinMode(9, OUTPUT);
}

void loop()
{
    tone(9, 523, 10);
    delay(10);

    tone(9, 659, 10);
    delay(10);

    tone(9, 784, 10);
    delay(10);
}
```

www.hubout.it/makerslab/biblioduino

La pagina ufficiale del progetto Biblioduino, dove si potranno trovare informazioni, ulteriori esempi e molte altre risorse per approfondire.

www.arduino.cc

Il sito di Arduino riporta tutta la documentazione del linguaggio di programmazione, tutorial per spiegarne il funzionamento, un forum frequentatissimo e moltissime altre informazioni.



instructables

www.instructables.com

Instructables è un sito su cui si possono trovare progetti che vanno dalle ricette di cucina ai più complicati e grandi progetti meccanici. Tra questi vi sono moltissimi progetti realizzati con Arduino e molto ben commentati, da cui si può prendere spunto per imparare e crearne di nuovi.

Scrivici a makers@hubout.it

Cercaci su www.hubout.it/makerslab
su Facebook e Twitter

Vieni a trovarci ad Hubout Makers Lab
ogni Giovedì sera dalle 18.00 alle 22.00
al piano 2 del Centro culturale Il Pertini
di Cinisello Balsamo (MI).

Ti aspettiamo!

COMUNE DI
CiniselloBalsamo

ILPERTINI
La piazza dei saperi

MAKERS LAB

MARSE

COOPERATIVA SOCIALE ONLUS
IL TORPEDONE

CSBNO
Consorzio Sistema Bibliotecario Nord - Ovest