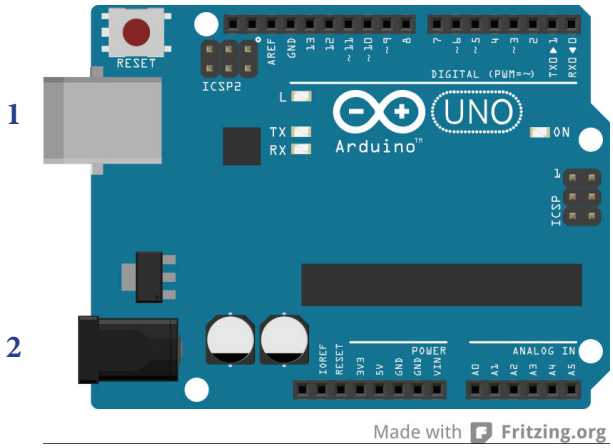
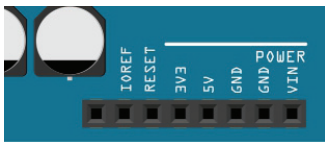
Arduino - Tutorial #2 - Iniziamo a programmare

Prima di parlare di programmazione e di codice vediamo una panoramica della scheda Arduino (per leggere il primo tutorial sulla presentazione e le basi di elettronica della scheda Arduino cliccare qui).

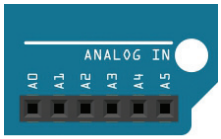


Arduino può essere alimentato inserendo un cavo USB (quello delle stampanti) nell'apposito connettore (n.1 nella figura qui sopra). Lo stesso cavo viene utilizzato per collegare la scheda al computer e permetter la programmazione della scheda stessa. Una volta uplodato il codice che andremo a programmare, Arduino lo memorizzerà nell'apposito microchip finchè non sarà sostituito con un nuovo codice. Questo vuol dire che Arduino è in grado di lavorare anche se separato da un computer. In questo caso dobbiamo alimentarlo con delle batterie (da 9 V oppure batterie stilo), pannelli solari, alimentatori 9V-12V, ecc...
Il connettore per eventuali alimentazioni esterne è il n.2 nella figura qui sopra. Nei tutorial su robot e applicazioni esterne vedremo come realizzare pacchi batteria per far funzionare Arduino in totale autonomia.

Pin Power

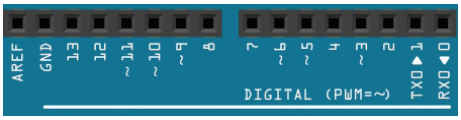


I pin posizionati in basso a sinistra sono chiamati POWER perchè permettono di alimentare un circuito esterno tramite Arduino. Inserendo un cavetto elettrico nel pin 5V e collagando questo alla breadboard, avremo un segnale di 5V per accendere i nostri componenti elettronici come sensori, led, motorini, etc...
I due pin GND servono per chiudere il circuito che stiamo realizzando. Inserendo un cavetto nel pin GND e collegando l'altra estremità alla breadboard siamo pronti per realizzare i nostri circuiti). Per maggiori informazioni clicca qui per vedere il tutorial precedente.



Analog In

I pin analogici sono posizionati in basso a destra e sono composti da 6 ingressi: A0 - A5.
I pin analogici sono solamente di ingresso dati (INPUT) e vengono utilizzati per leggere i dati provenienti da sensori. In questo modo Arduino è in grado di "leggere" quello che succede nel mondo reale e trasformarlo, grazie ai sensori, in numeri che poi andremo ad analizzare.
La particolarità del dato analogico è che può comprendere una serie infinita di numeri, ad esempio un sensore di prossimità rilascia un numero che va da 0 a 900 a seconda di quanto un oggetto è vicino o lontano.
Più avanti, utilizzando dei sensori analogici, spiegherò meglio questo concetto di flusso infinito di numeri.



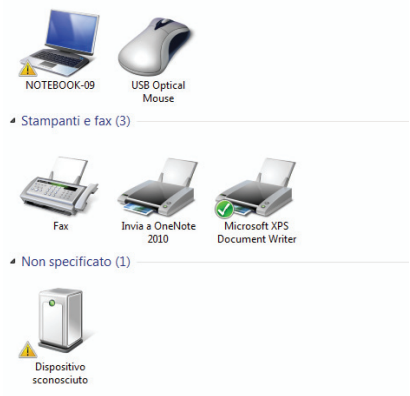
Digital

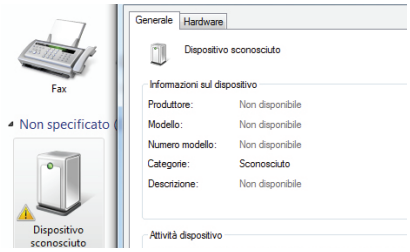
I pin digitali sono 14 (0 - 13) e sono posti nella parte in alto della scheda.
Questi pin al contrario di quelli analogici restituiscono un valore che è uguale a **1** o a **0**. I pin digitali possono essere sia di lettura (INPUT) che di scrittura (OUTPUT). Questo vuol dire che se inseriamo un sensore digitale in un determinato pin possiamo vedere se questo è attivo o spento, oppure in uscita possiamo fare accendere una luce o spegnerla.

N.B: alcuni pin digitali, il 3, 5, 6, 9 ,10 e 11 sono chiamati pin PWM. I pin PWM possono essere sia digitali che analogici e possono essere sia di input che di output. Per il momento non consideriamo questi pin, verranno approfonditi più avanti.

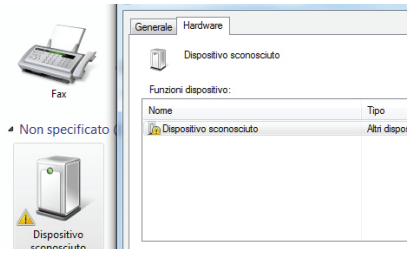
Installiamo Arduino

Dopo aver visto una panoramica della scheda e un pò di elettronica nel precedente tutorial, possiamo installare Arduino.
Scarichiamo dal sito http://arduino.cc/en/Main/Software il software open source per programmare Arduino.
Selezioniamo il nostro sistema operativo (Windows, Apple o Linux) e aspettare la fine del download (per utenti Windows scarichiamo **Windows ZIP file)**. Una volta scaricato scompattiamo la cartella.
Al suo interno c'è una cartella chiamata "libraries" (per utenti Mac bisogna crearla manualmente all'interno della cartella "Arduino"). In seguito utilizzeremo questa cartella per inserire delle librerie che verranno utilizzate da Arduino per gestire alcuni componenti esterni, per il momento lasciamo tutto com'è, e continuiamo con l'installazione dei driver.
Per gli utenti Mac questo passaggio può essere saltato (vai al tutorial qui sotto, pagina 4).
Per utenti Linux potete seguire una delle tante guide su internet, mi raccomando poi tornate qui!
Per utenti Windows continuate a leggere qui sotto...
Colleghiamo Arduino con il cavetto USB al computer. Clicchiamo su **Start/ Pannello di Controllo**. Nell'immagine a sinistra, in basso, il dispositivo sconosciuto.
Doppio Click su dispositivo sconosciuto.

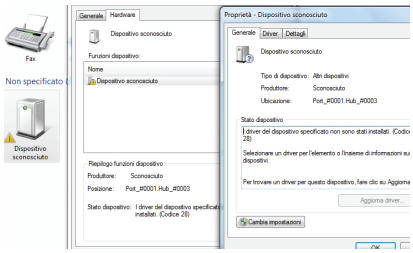




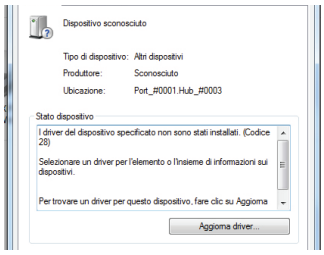
Click su **Hardware**



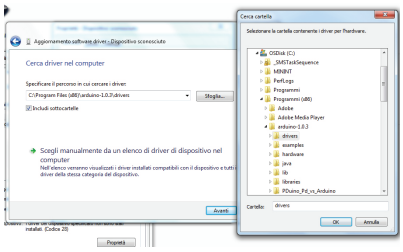
Click su **Dispositivo sconosciuto**



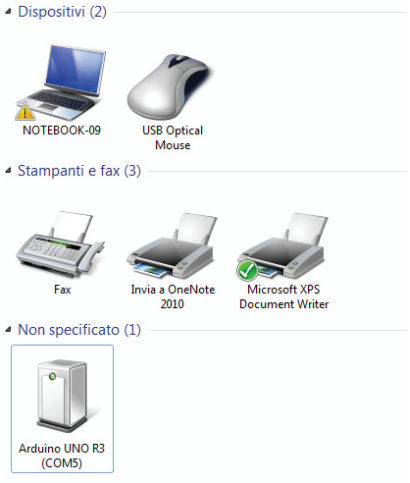
Click su **Cambia impostazioni**



Click su **Aggiorna Driver...**



Click su Sfoglia...
Selezioniamo la cartella **drivers** all'interno della cartella Arduino precedentemente scaricata.

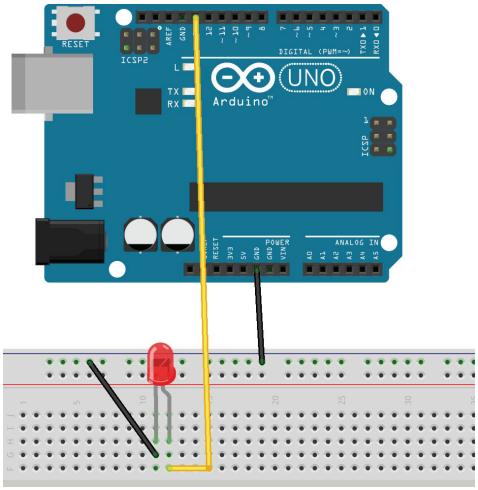


Durante l'installazione dei driver una finestra ci avvisa che il software non ha superato il testing di windows...
clicchiamo su **Continua.**
Se tutto è andato bene nel Pannello di Controllo ora apparirà la scritta Arduino UNO.

Se durante la procedura Windows non riconosce i driver da installare basta scollegare il cavo USB e collegarlo a una nuova porta USB. Ripetere l'installazione sopra descritta e Arduino verrà installato.

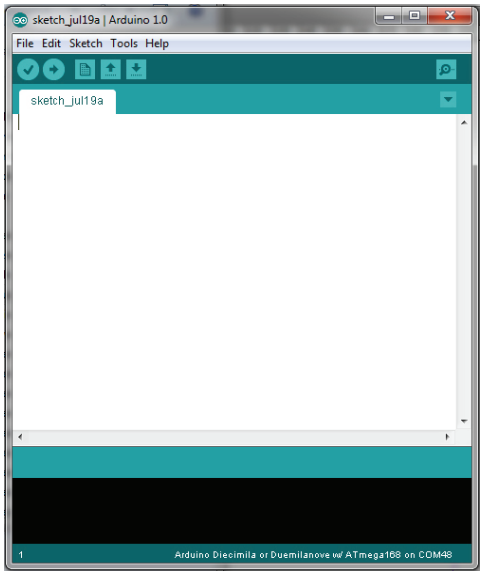
Tutorial Blink

Finalmente possiamo iniziare con il primo e vero tutorial di Arduino, il blink! Per blink si intende l'accensione e lo spegnimento di un led tramite programmazione.
Per prima cosa, ogni volta che iniziamo un progetto con Arduino, dobbiamo realizzare la parte elettronica e poi proseguire con la programmazione.
Per questo primo esercizio abbiamo bisogno di una breadboard, un led e un paio di cavetti.



Colleghiamo il piedino piccolo del Led al pin GND con un cavetto e il piedino lungo al Pin 13 digitale (in alto a sinistra della scheda).
Perchè stiamo collegando il piedino dell'alimentazione del Led al Pin 13? Nel precedente tutorial abbiamo visto come accendere il Led collegandolo direttamente al Pin 3V.
Collegando il pin di alimentazione del Led a un pin di Arduino siamo in grado di comandare l'accensione e lo spegnimento del Led tramite programmazione.

Il circuito è finito. Se non abbiamo dei cavetti e la breadboard possiamo inserire il Led direttamente nella scheda Arduino. Il piedino lungo dentro il Pin 13 e il piedino corto del Led dentro il Pin GND (vicino al Pin13)

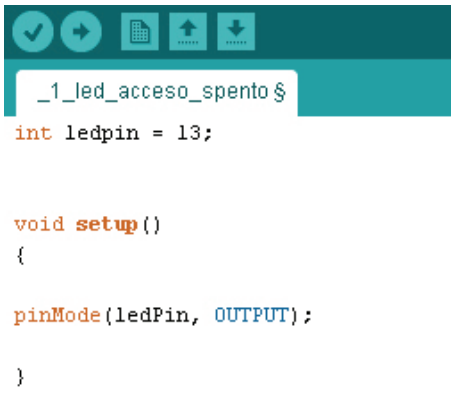


Passiamo ora alla programmazione.
Apriamo il software di Arduino, doppio click sul file arduino.exe all'interno della cartella precedentemente scaricata.
Per chi non ha mai programmato è importante sapere che tutti i linguaggi di programmazione (c++, java, etc..) hanno una loro sintassi, ovvero un vero e proprio linguaggio e bisogna rispettarne le regole di scrittura.
Per il momento non cercate di capire tutto il codice che andremo a scrivere. L'importante è capire la logica della programmazione. Molto presto impareremo a realizzare i nostri sketch di Arduino semplicemente con dei copia-incolla. Per arrivare a questo è però necessario sapere quale parte di codice copiare e soprattutto incollare senza creare errori.
Il codice per programmare Arduino è molto semplice e si suddivide in tre parti fondamentali: la dichiarazione delle variabili e dei pin utilizzati, la funzione setup e la funzione loop.
Iniziamo a scrivere un pò di codice.

1° fase: dichiarazione delle variabili e dei pin utilizzati:
In questo primo esercizio accendiamo e spegniamo un Led posizionato nel Pin13 di Arduino.
Scriviamo all'interno del programma, nella schermata che si è aperta, il seguente codice: int ledpin = 13;
ledpin è il nome che stiamo dando al pin di Arduino.
13 è in numero del pin utilizzato.
int vuol dire che stiamo utlizzando un numero intero (senza virgole).
Da ora in poi per gestire le azioni sul Pin13 di Arduino utilizzeremo il nome **ledpin.**

N.B: il nome **ledpin** è un nome inventato. Possiamo utilizzare qualsiasi nome ad esempio, ledcomandato, ledarduino, etc... la cosa importante è il numero che gli assegniamo che deve corrispondere al pin utilizzato nel circuito.

La prima fase della programmazione è terminata, proseguiamo con la seconda fase, ovvero la funzione **setup**.



La funzione setup deve essere scritta seguendo queste regole:

```
void setup() {

}
```

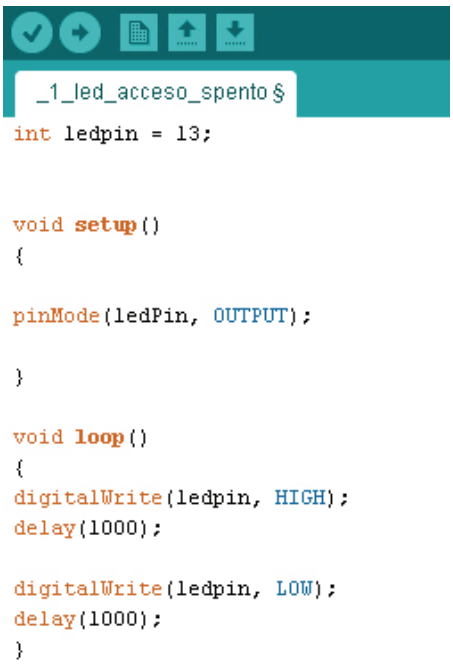
tutto quello che scriviamo tra le due parentesi graffe fa parte della funzione setup. A cosa serve questa funzione? Con void setup dichiariamo se i pin che stiamo utilizzando su Arduino sono di Input o di Output (se dobbiamo leggere i dati provenienti da un sensore il pin deve essere di Input, se dobbiamo inviare segnali elettrici al di fuori della scheda allora il pin deve essere di Output). Più avanti vedremo altre impostazioni da inserire all'interno della funzione. Nel nostro esempio, dato che dobbiamo accendere un Led tramite il pin13, dobbiamo dichiararlo di Output:

```
pinMode(ledpin,OUTPUT);
```

La seconda fase della programmazione di Arduino è terminata, passiamo subito alla terza fase ovvero la funzione loop.
Anche questa funziona deve essere scritta tra le parentesi graffe:

```
void loop() {

}
```



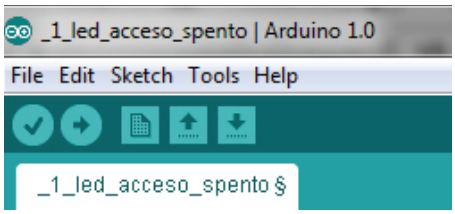
N.B.: i valori "0" e "1" possono essere scritti nel programma anche "HIGH" e "LOW" come nella figura sopra.

A differenza della funzione setup(), la funzione loop() permette di ripetere all'infinito i comandi che scriveremo al suo interno. Questo vuol dire che le operazioni verranno ripetute in modo continuo (partendo dall'alto verso il basso) fino allo spegnimento della scheda.
Quali sono i comandi che possiamo scrivere nella funzione loop()?
I principali comandi sono fondamentalmente due: **digitalWite** e **analogRead** (più avanti vedremo anche i controlli, i cicli e altri comandi importanti, ma per il momento studieremo solamente questi due).
Con analogRead possiamo "leggere i valori analogici" provenienti da un sensore (prossimi tutorial).
Con digitalWrite possiamo scrivere e inviare un segnale, su un pin di Arduino. I segnali digitali possono essere solamente 0 e 1. Inviando uno "0" il nostro led si spegnerà, inviando "1" si accenderà.
Il segnale digitale **0** viene interpretato da Arduino come 0V mentre il segnale digitale **1** diventa nel mondo reale 5V e quindi è in grado di accendere un Led.
Scriviamo subito dopo il void setup() questo codice:

```
void loop()
{
digitalWrite(ledpin, 1);
delay(1000);
digitalWrite(ledpin, 0);
delay(1000);
}
```

La funzione loop() è la terza e ultima parte della programmazione di Arduino. Questa funzione viene letta dall'alto verso il basso e tutte le operazioni in essa presenti vengono ripetute all'infinito.
In questo caso il primo comando che incontra è: digitalWrite(ledpin, 1); - il Led posizionato nel Pin13 riceverà un segnale "alto" ovvero di 5V che farà accendere il Led.
Subito dopo riceverà un delay(1000); - che rappresenta una pausa di 1 secondo. Il Led rimane acceso per un secondo.
Con digitalWrite(ledpin, LOW); e delay(1000); - il Led riceverà un segnale di 0V e quindi si spegnerà per un secondo. Subito dopo Arduino ricomincerà a leggerà i comandi dall'alto, dalla funzione loop(), e quindi accenderà il Led e lo spegnerà all'infinito.
Per testare se lo sketch è stato scritto correttamente clicchiamo sull'icona in alto a forma di "V".

Se tutto è andato bene, sotto apparirà la scritta **Done Compling**.
In caso di errore il programma ci segnala la riga dove è possibile che sia stato scritto un comando sbagliato. Fate molta attenzione alla sintassi di programmazione, ovvero punteggiatura, lettere minuscole o maiuscole, parentesi graffe, etc...
E' arrivato il momento di caricare il programma nella scheda Arduino (uplodare).
Colleghiamo Arduino con il cavetto USB al computer.
Prima di uplodare dobbiamo però segnalare al programma quale scheda Arduino stiamo utilizzando.



Click su **Tools** e poi su **Board**, selezioniamo la scheda utilizzata.
Sempre su **Tools** selezioniamo **Serial Port** e clicchiamo sulla porta USB dove il nostro Arduino è collegato.
(Queste due operazioni, se utilizziamo sempre la stessa scheda e la colleghiamo alla stessa porta USB, non devono essere effettuarle in futuro).
Per uplodare clicchiamo sull'icona con la freccia verso destra. Se tutto è andato bene apparirà in basso la scritta **Done Uploding**.
Il Led posizionato nella breadboard inizierà ad accendersi e spegnersi ad intervalli di un secondo.

Per completare il tutorial salviamo il nostro primo sketch (quanto abbiamo scritto nel programma).
Click su File - Salva. Scegliamo un percorso dove salveremo tutti i nostri esempi in modo da avere una collezione di sketch sempre pronti all'utlizzo e alle varie modifiche. Salviamo il file con il nome: **1_led_acceso_spento**