

Blinking Led

Un semplice circuito costruito e programmato con Arduino, NodeJS e Johnny-five.

Introduzione

Se siete in possesso di una copia di questo tutorial avete probabilmente già visto il repository git del Coderdojo di Firenze contenente documentazione e codice per questo semplice ma divertente esercizio di tinkering e programmazione:

- <https://github.com/coderdojofirenze/j5-blinkingled.git>

Nel realizzare quanto descritto in questo tutorial potete utilizzare come base di partenza questo repository o, ancora meglio, partire da zero e realizzare tutto per conto vostro. Questo tutorial fa riferimento al secondo caso, tutto viene spiegato nel dettaglio in modo che possiate realizzare il tutto senza dover utilizzare git in nessun modo.

Installazione software

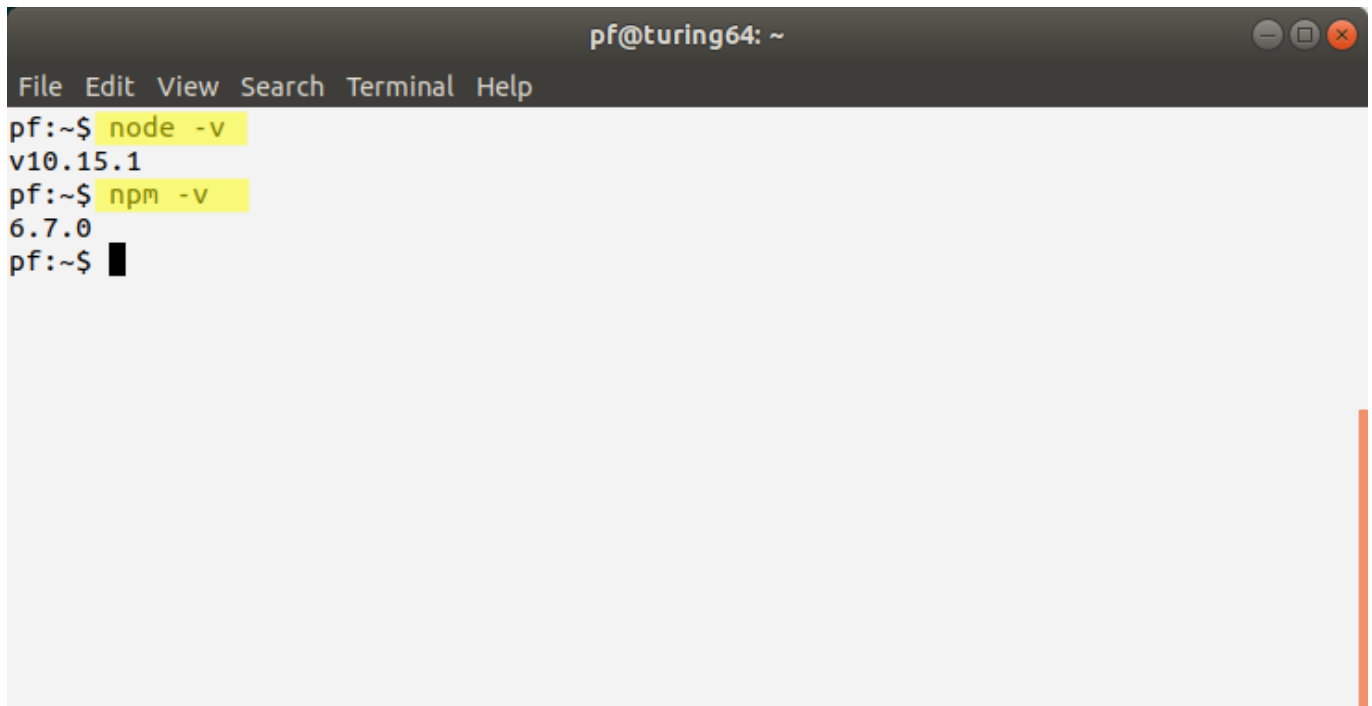
Questo tutorial è stato provato con la scheda **Arduino Uno**, per cui nel resto della descrizione si farà riferimento ad una scheda di questo tipo, anche se è naturalmente possibile che il tutto funzioni anche con altre tipologie di schede.

È necessario installare sul vostro PC:

- [La IDE di Arduino](#)
- [NodeJS](#)

L'installazione di questi software dipende dal sistema operativo che avete sul vostro PC per cui non entreremo qui in troppi dettagli. Fate riferimento ai siti indicati e, se siete ad una sessione di Coderdojo, chiedete aiuto ai mentor.

Alla fine dell'installazione per verificare che NodeJS sia stato installato correttamente aprire un terminale, dare i comandi `node -v` e `npm -v` e verificare che rispondano con il numero di versione:

A terminal window titled 'pf@turing64: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output: 'node -v' returns 'v10.15.1', and 'npm -v' returns '6.7.0'. The prompt 'pf:~\$' is shown at the end of the last line.

```
pf@turing64: ~  
File Edit View Search Terminal Help  
pf:~$ node -v  
v10.15.1  
pf:~$ npm -v  
6.7.0  
pf:~$
```

Inizializzazione progetto NodeJS

A questo punto occorre inizializzare un nuovo progetto NodeJS. Il modo più semplice per farlo consiste nell'utilizzare il comando `npm init` dalla command line. Per fare questo utilizziamo il terminale che abbiamo aperto in precedenza, creiamo una directory dove mettere il nostro progetto, entriamoci e diamo il comando suddetto. In tutto dare i seguenti comandi:

```
mkdir j5-blinkingled  
cd j5-blinkingled  
npm init
```

Rispondere alle domande che appaiono sul terminale. In molti casi si può premere invio accettando la risposta proposta, in alcuni casi può essere utile scrivere qualcosa di diverso. Per esempio alla domanda `description:` si può rispondere **"Un led lampeggiante con Arduino, NodeJS e Johnny-five"** e alla domanda `author:` si può rispondere con il proprio nome (e volendo con il proprio indirizzo email).

Importante: Alla domanda `entry point:` si deve rispondere con **"blink.js"**.

Alla fine, sul terminale dovrebbe apparire qualcosa di simile a quanto raffigurato nell'immagine che segue.

```
pf@turing64: ~/j5-blinkingled
File Edit View Search Terminal Help

pf:j5-blinkingled$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (j5-blinkingled) <invio>
version: (1.0.0) <invio>
description: un led lampeggiante con Arduino, Nodejs e Johnny-five
entry point: (index.js) blink.js
test command: <invio>
git repository: <invio>
keywords: <invio>
author: Francesco Piantini
license: (ISC) <invio>
About to write to /home/pf/j5-blinkingled/package.json:

{
  "name": "j5-blinkingled",
  "version": "1.0.0",
  "description": "un led lampeggiante con Arduino, Nodejs e Johnny-five",
  "main": "blink.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Francesco Piantini",
  "license": "ISC"
}

Is this OK? (yes) yes
pf:j5-blinkingled$
```

Alla fine di questa operazione all'interno della nostra directory è stato creato un file testo di nome `package.json` che contiene tutte le informazioni del nostro progetto così come gliele abbiamo date. Se necessario è possibile modificarlo in qualsiasi momento.

Installazione di johnny-five

N.B: per completare questo passo è necessario disporre della connettività a internet.

I progetti NodeJS utilizzano spesso delle librerie che aggiungono funzionalità all'ambiente di base. Queste si installano tramite il comando `npm install`. Nel nostro caso l'unica libreria necessaria è **Johnny-Five** (più tutti i componenti da cui dipende). Il tutto si installa tramite il comando:

```
npm install johnny-five --save
```

Alla fine di questa operazione, all'interno della directory di progetto è stata creata una directory `node_modules` che contiene tutte le librerie scaricate. Cosa più importante, nel file `package.json` è stata salvata l'informazione che il progetto dipende dalla libreria che abbiamo appena installato. Controllando si può infatti vedere che in questo file è stato aggiunto un blocco `"dependencies"` che contiene la riga `johnny-five`:

```
"dependencies": {  
  "johnny-five": "^1.0.0"  
}
```

Tramite questa informazione, se per qualche motivo la directory `node_modules` venisse a mancare o risultasse corrotta è possibile scaricare nuovamente tutto il necessario tramite il semplice comando:

```
npm install
```

Installazione Firmware su scheda Arduino

Collegare la scheda Arduino al PC utilizzando il cavo USB. Controllare che si accenda e che venga correttamente riconosciuta dalla IDE di Arduino: su quest'ultima controllare che **Board** e **Port** siano inizializzate correttamente e che la scheda risponda regolarmente al comando **Tools > Get Board info**.

A questo punto dobbiamo installare sulla scheda il firmware "Firmata", necessario per permettere a Johnny-Five di comunicare con la scheda stessa.

- Aprire il progetto "Standard Firmata" seguendo il menù: **File > Examples > Firmata > StandardFirmata**.
- Cliccare sul pulsante di **Upload**
- Aspettare finché sulla IDE non appare il messaggio **Done Uploading**

A questo punto la IDE può essere chiusa. Infatti non ci servirà più per tutta la durata del tutorial.

Un primo led lampeggiante

Per ottenere un primo risultato e capire come funziona Johnny-Five scriviamo un primo programma che per funzionare non ha bisogno di realizzare nessun circuito. Infatti le schede Arduino dispongono di un led collegato direttamente sulla scheda che si controlla tramite la linea led numero **13**. Scriviamo quindi il seguente codice nel file di nome `blink.js`:

```
// Un LED Lampeggiante

// Carichiamo la libreria 'johnny-five'
var five = require('johnny-five');

// Dichiariamo una scheda di tipo Arduino
var board = new five.Board();

// Dichiariamo una funzione che fa lampeggiare un LED
// per sempre con un periodo di mezzo secondo.
board.on('ready', function() {
  var led = new five.Led(13);
  led.blink(500)
});
```

Per mandare in esecuzione il programma, dal terminale dare il seguente comando:

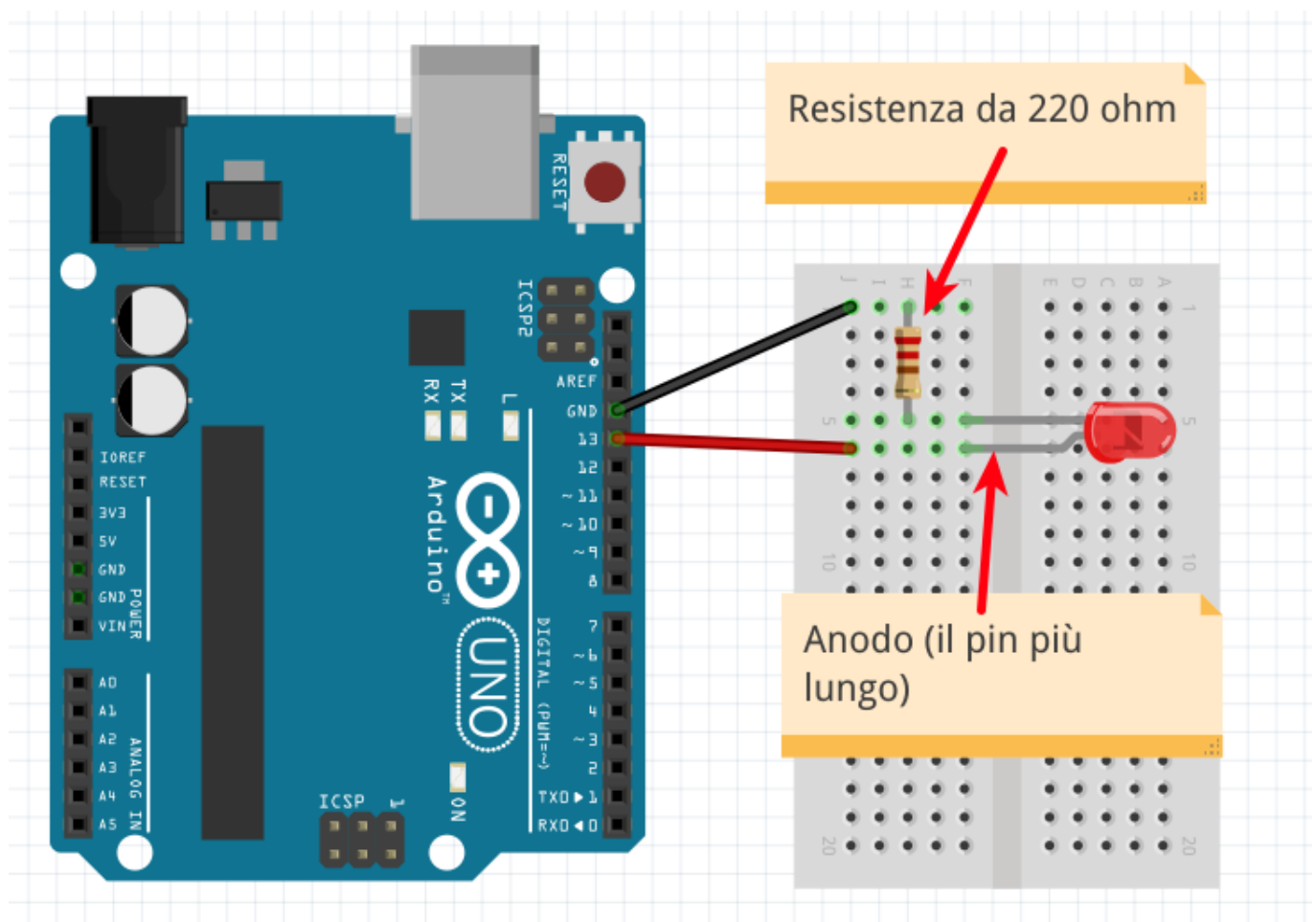
```
node blink.js
```

Dopo una fase iniziale di setup si può notare come il piccolo led vicino al connettore laterale più lungo della scheda Arduino (quello con accanto l'etichetta "L"), comincia a lampeggiare. Il nostro primo codice JavaScript è adesso in esecuzione su una scheda a microcontrollore!

Per interrompere l'esecuzione del programma digitare i tasti "Ctrl-C" due volte.

Un "vero" circuito

Per rendere più interessante il nostro esercizio possiamo realizzare un piccolo circuito collegando un "vero" led alla scheda. Realizziamo quindi il seguente circuito:



Il circuito è molto semplice. L'unico particolare a cui fare attenzione è la polarità del diodo LED. Il LED infatti si accenderà solo se è collegato nel giusto verso. In particolare l'**anodo** (riconoscibile perché è il pin più lungo) va collegato dal lato del pin 13 (opposto alla massa indicata come "GND" ossia ground). L'altro pin del LED si chiama **catodo**, va collegato verso la massa ed è riconoscibile anche per un piccolo taglio che c'è sul corpo del LED.


Adesso eseguiamo nuovamente il programma tramite il comando `node blink.js`. Adesso oltre al LED sulla scheda lampeggerà anche il LED che abbiamo montato noi.

Appendice 1: Il codice dei colori delle resistenze

La resistenza, un componente fondamentale dei circuiti elettronici si misura in Ohm (Ω), dal nome dello scienziato tedesco [Georg Simon Alfred Ohm](#) lo scopritore della famosa [legge di Ohm](#) che esprime la legge di proporzionalità tra differenza di potenziale e corrente.

Quando si costruisce un circuito elettrico, le resistenze si riconoscono facilmente: sono dei piccoli cilindri di materiale che sembra plastica molto dura dal quale sbucano dei fili metallici che poi sono quelli che servono per collegare la resistenza al circuito. Il valore di resistenza viene indicato tramite delle strisce colorate presenti sul corpo della resistenza. Ogni colore rappresenta un numero. Anche se possono esistere versioni diverse, le resistenze più comuni sono quelle con il codice a quattro strisce. Le prime tre strisce rappresentano il valore in ohm e la quarta la tolleranza.

Delle tre strisce che indicano il valore le prime due sono le prime due cifre del numero e la terza il moltiplicatore. Fare riferimento alla seguente immagine per capire come calcolare il valore.



Valori numerici		Moltiplicatore		Tolleranza	
Nero	0	Argento	0,01	Argento	$\pm 10\%$
Marrone	1	Oro	0,1	Oro	$\pm 5\%$
Rosso	2	Nero	1	Marrone	$\pm 1\%$
Arancio	3	Marrone	10	Rosso	$\pm 2\%$
Giallo	4	Rosso	100	Verde	$\pm 0,5\%$
Verde	5	Arancio	1K	Blu	$\pm 0,25\%$
Blu	6	Giallo	10K	Viola	$\pm 0,1\%$
Viola	7	Verde	100K		
Grigio	8	Blu	1M		
Bianco	9	Viola	10M		

Per esempio per capire i colori presenti su una resistenza con il valore di $220\ \Omega$ bisogna pensare che $220 = 22 \times 10$, quindi su queste resistenze ci saranno due strisce rosse (che corrispondono al 2) e una striscia marrone (che corrisponde al moltiplicatore 10).

Appendice 2: Codice Nativo "Arduino" vs. Johnny-Five

Torniamo al nostro circuito. Abbiamo visto che per ottenere il semplice risultato di far lampeggiare un LED abbiamo dovuto scomodare una quantità ingente di codice. Solo in termini di moduli NodeJS (tra cui Johnny-Five) abbiamo dovuto scaricare quasi 10 Mbytes di codice! Per non parlare del complesso firmware Firmata che abbiamo dovuto installare su Arduino.

Inoltre per far girare il programma dobbiamo tenere costantemente collegato alla scheda Arduino un PC con in esecuzione `node`. Come stacciamo il PC o fermiamo NodeJS, il programma smette di funzionare.

L'alternativa "classica" per ottenere lo stesso risultato con il codice "nativo" Arduino consiste molto più semplicemente nell'aprire la IDE, digitare il seguente codice e scaricarlo sulla scheda tramite il pulsante "Upload":

```
// La funzione setup() viene eseguita una sola volta tutte le volte
// che si accende la scheda o quando si preme il pulsante Reset
void setup() {
  // Inizializza come output la linea digitale dove è collegato il LED
  incorporato.
  // La costante LED_BUILTIN è equivalente al numero 13
  pinMode(LED_BUILTIN, OUTPUT);
}

// La funzione LOOP viene eseguita continuamente
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // Accende il LED (HIGH = tensione di
  alimentazione)
  delay(500);                      // attende 500 ms = 1/2 s
  digitalWrite(LED_BUILTIN, LOW);  // Spenge il LED togliendo la tensione
  di alimentazione
  delay(500);                      // attende 500 ms = 1/2 s
}
```

Tutto molto più semplice, no? Inoltre il programma rimane in esecuzione sulla scheda anche quando la stacciamo dal PC. Infatti questo codice è stato *compilato* e quindi scaricato fisicamente sulla EEPROM della scheda. Tutte le volte che la alimenteremo il codice inesorabilmente andrà in esecuzione senza bisogno di alcun supporto dall'esterno.

Invece il codice NodeJS che abbiamo realizzato in precedenza, viene eseguito sul PC collegato alla scheda e i "comandi" come l'accensione del LED vengono inviati in tempo reale tramite il dialogo tra la libreria Johnny-Five sul PC e il firmware Firmata installato sulla scheda.

Quale è quindi il vantaggio di utilizzare NodeJS e Johnny-Five rispetto a lavorare direttamente con la IDE di Arduino? Per avere una risposta, basta pensare al mondo a cui ci permette di accedere la programmazione JavaScript... Maggiori dettagli nei prossimi tutorial!