

Tutorial:

Incominciamo a programmare l'ATmega8

Conoscenze richieste: Basi di elettronica digitale

Tempo richiesto: 1 - 2 ore circa

Linguaggio tecnico usato: Semplice ed amichevole

Autore: Emanuele Aimone

Contents

INTRODUZIONE.....	3
GLI ELEMENTI USATI IN QUESTO TUTORIAL.....	4
L'hardware.....	4
Il compilatore.....	7
Firmware Uploader.....	7
IL PRIMO FIRMWARE.....	8
Lampeggio di un Led.....	8
Compilare il firmware.....	10
Caricare il firmware	12

INTRODUZIONE

In questo tutorial vedremo come installare i software necessari per programmare un ATmega8. Inoltre, vedremo in pratica come collegare l'ATmega8, come compilare un programma in linguaggio C e come caricarlo tramite il programmatore USBasp.

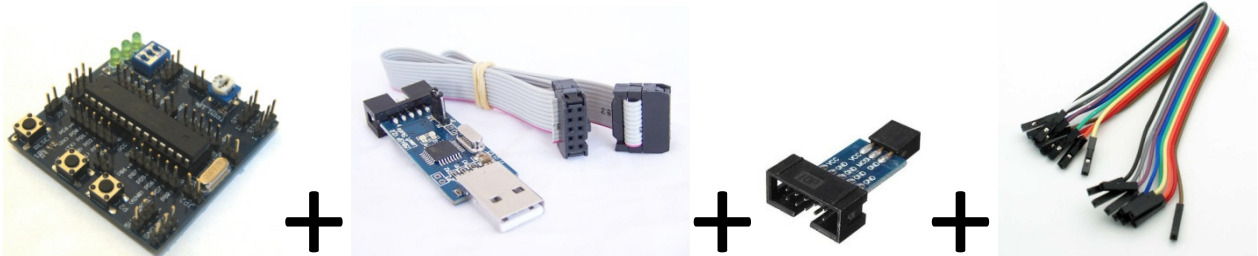
GLI ELEMENTI USATI IN QUESTO TUTORIAL

Per il tutorial abbiamo bisogno di una semplice demoboard, il programmatore USBasp, il compilatore WinAVR 20030913, e il caricatore di firmware (Firmware Uploader) Extreme Burner v1.4.3.

L'hardware

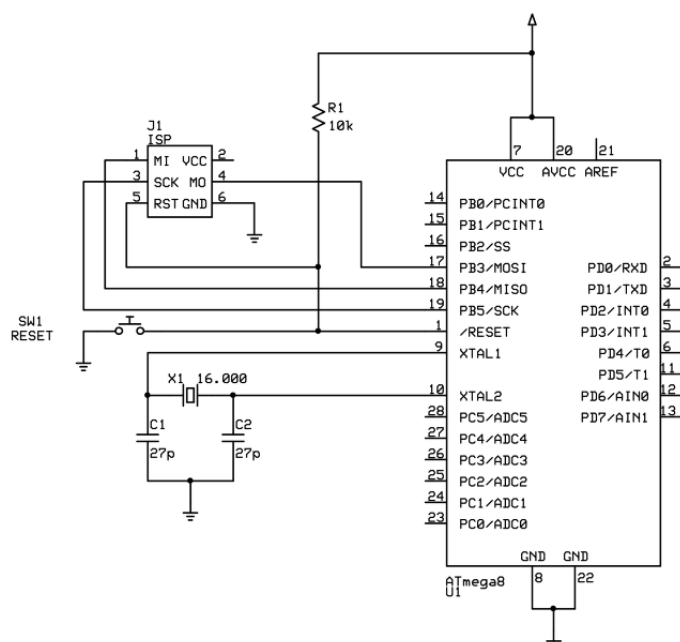
Vediamo subito delle foto dell'hardware usato:

Demoboard ATmega8 + *USBasp* + *Adattatore USBasp* + *Jumpers*



Questa semplice demoboard è costruita per avere i collegamenti minimi per il funzionamento dell'ATmega8. La demoboard attorno all'ATmega8 presenta dell'hardware collegabile tramite jumper adatto a sperimentare.

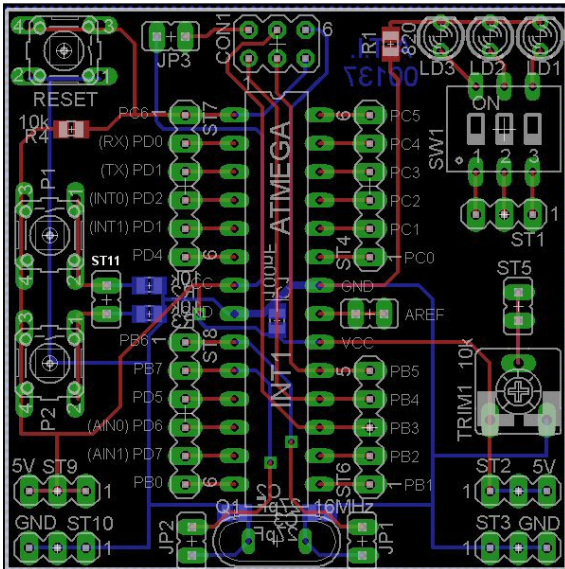
Qui vediamo lo schema di esempio:



Nella demoboard usata, anche il cristallo da 16MHz ha due jumper che permettono di essere scollegato perchè l'ATmega8 può funzionare anche senza.

Ovviamente è sconsigliatissimo collegare tramite dei jumper il quarzo perchè può causare uno spostamento dalla frequenza di lavoro. Il quarzo con i suoi due condensatori deve essere il più possibile vicino al microcontrollore ed eventualmente la sua carcassa collegata a massa.

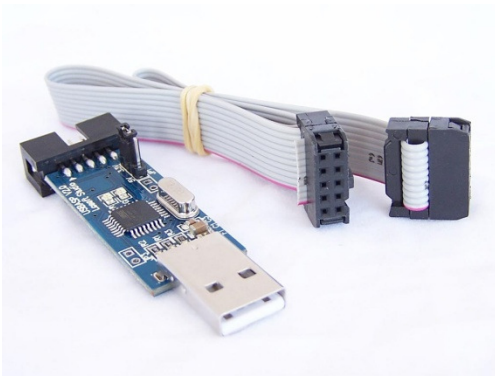
Qui vediamo la demoboard nel suo disegno cad:



Vediamo cosa serve l'hardware attorno:

- ST4, ST6, ST7 e ST8: sono delle strip direttamente collegate agli IO dell'ATmega8.
- RESET: in alto a sinistra premendolo riavvia il firmware caricato sull'ATmega8.
- P1 e P2: due tasti collegabili a piacere con dei jumper attraverso la strip ST11. Questi tasti hanno una resistenza di Pull-Down da 10k.
- TRIM1: è un trimmer da 10k che permette di simulare un'analogica. Il pin centrale del trimmer è collegato alla strip ST5 e gli altri due pin sono collegati a 5V e GND.
- LD1, LD2 e LD3: Sono dei semplici Led collegabili tramite ST1. Il dip switch SW1 permette di abilitarli o disabilitarli.
- JP1 e JP2: sono i jumper che collegano il quarzo da 16MHz (ricordare: sconsigliato collegarlo tramite dei jumper, va bene solo per prove)
- CON1: qui si collega il programmatore USBAsp attraverso un adattatore.
- JP3: collega/scollega i 5V del programmatore USBAsp in caso l'alimentazione della demoboard sia fornita da un'altra fonte.
- ST2, ST3, ST9 e ST10: sono strip per prelevare o fornire l'alimentazione 5V e GND.

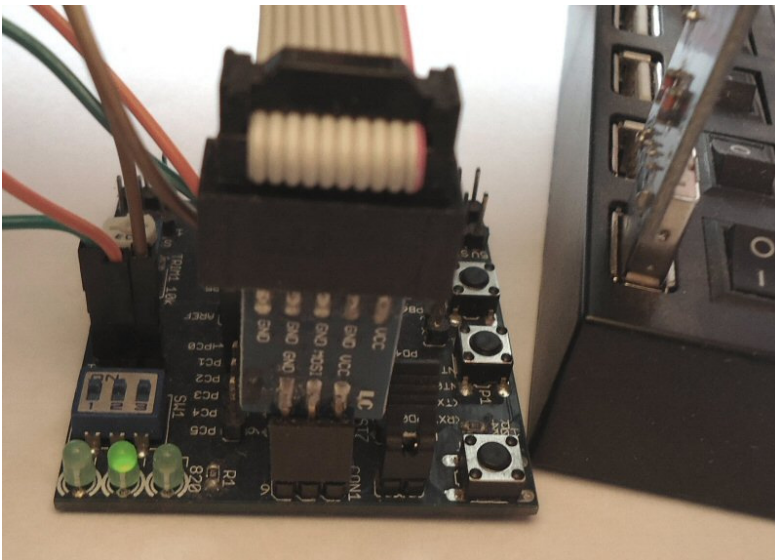
Vediamo Il programmatore usato USBasp:



La demoboard è collegabile attraverso questo adattatore:



Vediamo un collegamento con il PC:



Bene ora siamo pronti per installare i programmi necessari.

Il compilatore

Il compilatore è il WinAVR 20030913, ecco il link:

<http://winavr.sourceforge.net/>

Una volta installato è importante conoscere il percorso dove è installato, la versione 20030913 usa questo percorso di default:

C:\WinAVR-20100110

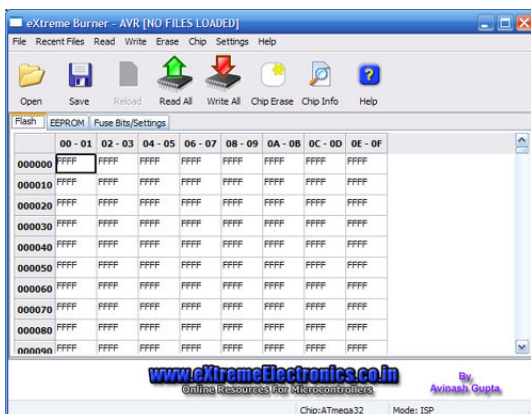
Serve saperlo per creare il Make File che vedremo successivamente.

Firmware Uploader

Il Firmware Uploader usato è il Extreme Burner 1.4.3, ecco un link per scaricarlo:

<http://extremeelectronics.co.in/avr-tutorials/gui-software-for-usbasp-based-usb-avr-programmers/>

Vediamo un' anteprima:



IL PRIMO FIRMWARE

Il primo firmware che scriviamo farà lampeggiare un led sul PIN23 chiamato PC0 del'ATmega8.

Lampeggio di un Led

Vediamo subito l'esempio di programma che dobbiamo scrivere:

```
#include <avr/io.h>
#include <avr/delay.h>

#define DELAY_LED    500 //mS

int main(void){

    //configura PC0 (PIN23) come Output
    //e tutti gli altri PCx vengono configurati come Input
    DDRC = 0x01;

    // loop infinito
    while(1){
        PORTC |= 0x01;  //Led ON
        _delay_ms(DELAY_LED);
        PORTC &= 0xFE;  //Led OFF
        _delay_ms(DELAY_LED);
    }
}
```

Il firmware per prima cosa include le librerie per gli input ed output:

```
#include <avr/io.h>
```

Poi la libreria per creare la pausa tra un cambio di stato e l'altro:

```
#include <avr/delay.h>
```

Quindi definiamo in milli secondi di quanto deve essere la pausa tra un lampeggio e un altro:

```
#define DELAY_LED    500 //mS
```


A questo punto inizia il corpo principale del programma:

```
int main(void){  
.....  
}
```

La prima cosa che si deve fare è inizializzare il microcontrollore. Dato che ci serve solo un pin, inizializziamo il PIN23 (PC0) come output e lasciamo tutti gli altri PIN della porta PC come input:

```
DDRC = 0x01;
```

Poi creiamo un ciclo infinito dal quale non uscirà più per far eseguire le nostre funzioni in modo continuativo:

```
while(1){  
.....  
}
```

Ecco che quindi dentro il ciclo infinito impostiamo a 1 (5V) e a 0 (0V) il PIN23 (PC0):

```
PORTC |= 0x01; //Led ON  
_delay_ms(DELAY_LED);  
PORTC &= 0xFE; //Led OFF  
_delay_ms(DELAY_LED);
```

Ovviamente la funzione:

```
_delay_ms(DELAY_LED);
```

Fa fermare il microcontrollore su tale funzione per il tempo definito precedentemente con il define DELAY_LED.

Una volta scritto il firmware in un file di testo lo salviamo con il nome:

main.c

Ora dobbiamo compilarlo!

Compilare il firmware

Abbiamo bisogno di un file chiamato Make-File, allegata a questa guida abbiamo il file:

make

Il quale non presenta nessuna estensione, ma se noi lo apriamo come file di testo vedremo che ci sono scritte le configurazioni per la compilazione con WinAVR.

In particolare è importante accertarsi che sia configurato il percorso dove è stato installato WinAVR, quindi scorriamo il file fino a quando troviamo la voce:

DIRAVR =

Questa deve contenere il percorso del compilatore WinAVR, che nel nostro caso è:

DIRAVR = C:/WinAVR-20100110

Bene ora vediamo di costruirci un file per permetterci di compilare facilmente il firmware senza dover entrare dentro il prompt dei comandi di window.

Allegato troviamo un file chiamato:

build.bat

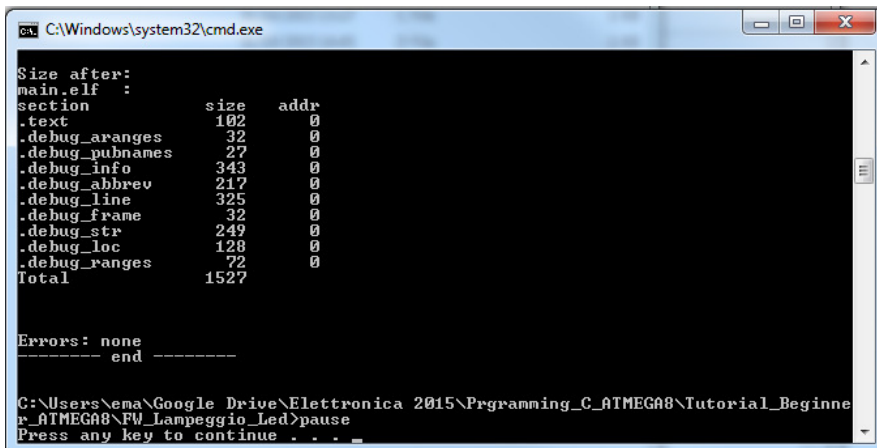
questo file apre il prompt di window, ed esegue il file **make**.

Per eseguire il file **make** (che sta nello stesso percorso del vostro file **main.c**) dobbiamo fornirgli il percorso, quindi apriamo il file **build.bat** con un editor di testo e assicuriamoci che la prima riga contenga il percorso dove risiedono i files **make** e **main.c**, esempio di file **build.bat**:

```
cd C:\mia_cartella_del_firmware_lampeggio_led
make
pause
```

Ora possiamo fare doppio click su **build.bat** e vedere se il compilatore restituisce dei messaggi di errori.

Esempio di schermata per il firmware senza errori:



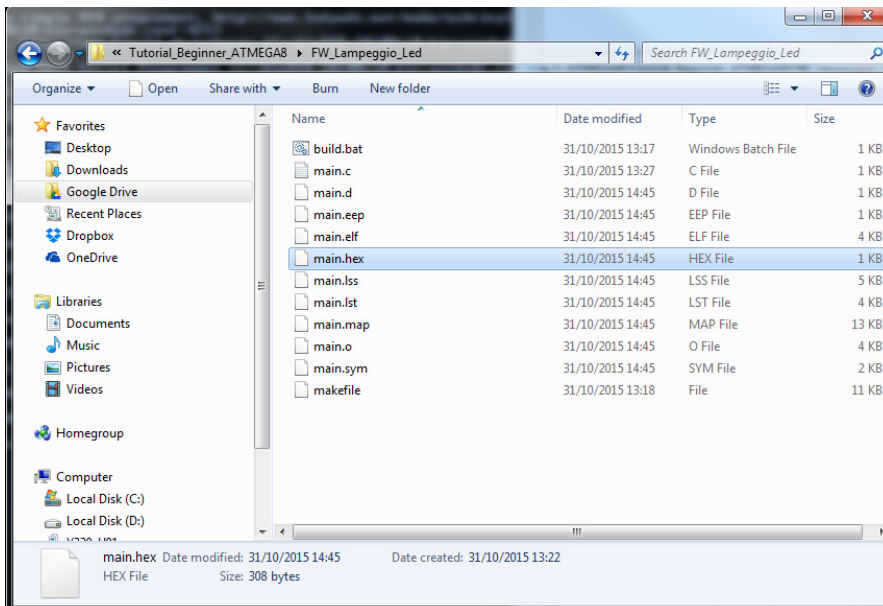
```
C:\Windows\system32\cmd.exe

Size after:
main.elf :
section      size      addr
.text        102      0
.debug_aranges  32      0
.debug_pubnames 27      0
.debug_info   343      0
.debug_abbrev  217      0
.debug_line   325      0
.debug_frame   32      0
.debug_str    249      0
.debug_loc    128      0
.debug_ranges  72      0
Total        1527

Errors: none
-----
end

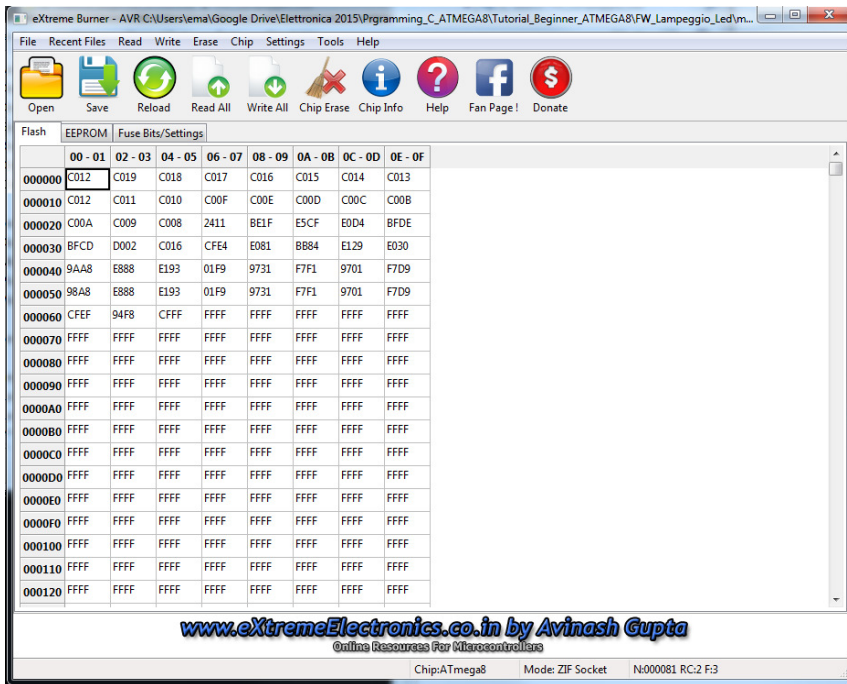
C:\Users\ema\Google Drive\Elettronica 2015\Prgramming_C_ATMEGA8\Tutorial_Beginner_ATMEGA8\FW_Lampeggio_Led>pause
Press any key to continue . . .
```

Se non ci sono stati errori, allora nella stessa cartella compariranno vari file tra cui il file **main.hex** che sarà quello che caricheremo sull'ATmega8 con il Firmware Uploader.

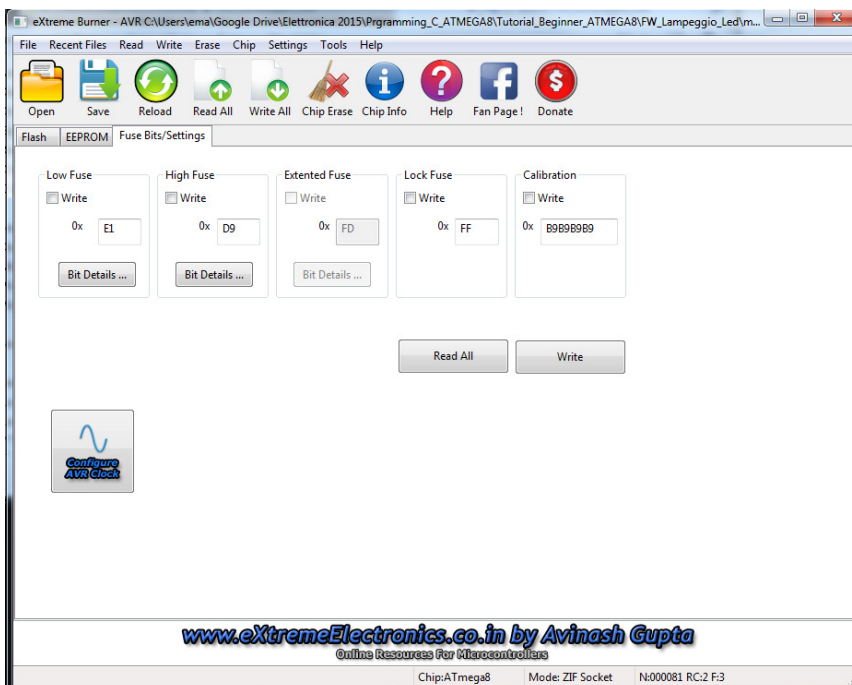


Caricare il firmware

Apriamo Extreme Burner v1.4.3 e clicchiamo sul tasto Open per andare a caricare il file main.hex:



Ora è molto importante impostare i Fuse, andiamo sul tab chiamato "Fuse Bits/Settings" e impostiamo i Fuse esattamente come l'immagine che segue:



Quindi:

Low Fuse = 0xE1

High Fuse = 0xD9

Lock Fuse = 0xFF

Calibration = 0xB9B9B9B9

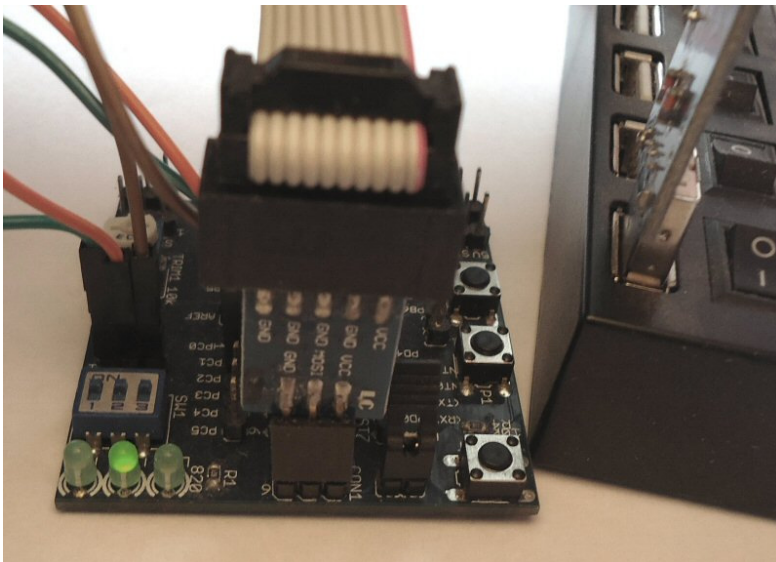
I fuse servono per impostare la configurazione con cui deve partire l'ATmega8. Per esempio, i fuse impostati come l'immagine sopra fanno partire l'ATmega8 senza bisogno del quarzo, cioè impostano l'oscillatore interno.

Non mi dilungherò su come configurare i fuse, su internet si trovano dei calcolatori come questo:

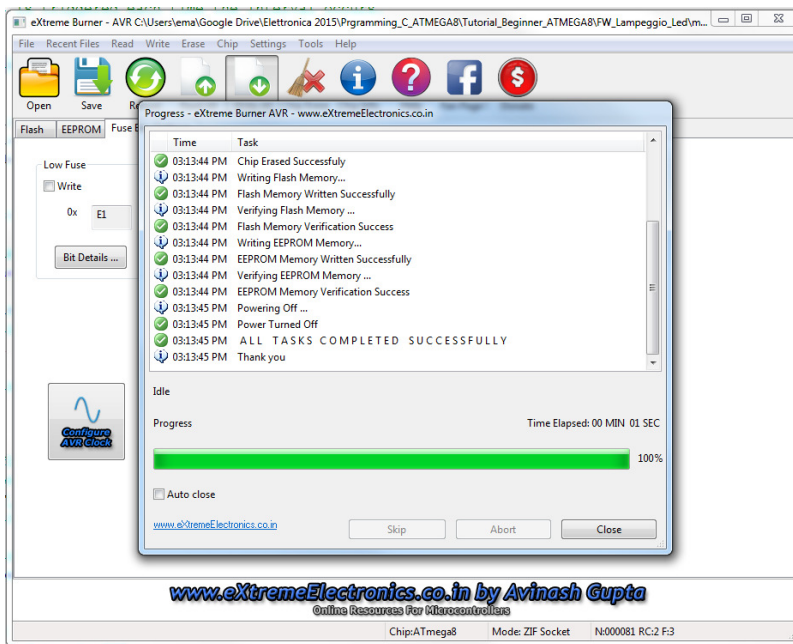
<http://www.engbedded.com/fusecalc/>

E' importante ricordare che se sbagliate un fuse potreste impostare l'ATmega8 per non essere più programmato, oltre a quelli di protezione, bisogna fare attenzione per esempio al fuse che imposta il pin di reset come un IO, però una volta impostato avete bisogno di un programmatore come l'STK500 per riprogrammare il vostro ATmega8 e non vi basta più l'USBasp.

Ora possiamo finalmente caricare il programma, assicuriamoci dei collegamenti come da foto:



Clicchiamo su Write All, se il caricamento è andato a buon fine allora avremo un schermata tipo questa:



Se avete collegato con un jumper il PIN23 al led centrale lo vedremo immediatamente lampeggiare!

Ora che avete appreso la procedura per compilare e caricare i vostri firmware non vi rimane che mettere in gioco la vostra creatività!