GUIDA alle Tecniche di Programmazione dei microcontrollori ATMEL

(Versione 4.00 - 23 Agosto 2012)

Realizzata dal Prof. Michele Menniti (27/03/2011)

Introduzione

Cari amici, dopo oltre un anno sono finalmente riuscito a trovare il tempo per aggiornare questa Guida, che è ormai diventata un punto di riferimento per la programmazione ISP, come mi confermano gli innumerevoli attestati di stima ed i ringraziamenti, a motivo di questo lavoro, ed anche svariati suggerimenti, sempre graditissimi. Nel frattempo l'IDE è arrivata alla versione 1.0.1, che finalmente prevede ciò che stiamo facendo da oltre un anno, integrando la programmazione di sketch via ISP.

Tutto ciò ha comportato che alcune informazioni riportate nella Guida non siano più valide, quindi serviva un aggiornamento, richiesto da più parti. Inoltre, avendo continuato a sperimentare e studiare, nel frattempo ho imparato ad usare nuove tecniche che ovviamente trasmetto, con grande piacere, a tutti gli utenti di Arduino.

Avviso gli Utenti che la Guida resta improntata sulla versione IDE 0022, ancora oggi utilizzatissima; per la versione 1.0.1 ho inserito un paio di paragrafi di aggiornamento che, però, da soli non sono sufficiente, essendo integrazioni dei capitoli specifici, in base alla configurazione hardware che si vuole realizzare, e poi integrarli col paragrafo aggiuntivo.

Nasce quindi oggi la quarta versione di questa Guida, alla quale ho dovuto cambiare nome, proprio per le tante novità in essa riportate; infatti ho trattato la programmazione di altri microcontrollori ATMEL e non solo dell'ATmega328P che, naturalmente, resta l'elemento portante dell'intera Guida. Come sempre ogni tecnica è stata verificata con innumerevoli prove e confermata da diversi utenti che hanno fatto sperimentazione in proprio.

Sento il dovere di ringraziare tutti coloro che sul Forum mi hanno fornito indicazioni e suggerimenti. Buona lettura e buona sperimentazione!

Un avviso importante a tutti coloro che useranno questa Guida per le loro prove, sperimentazioni e qualsiasi operazione in generale: come ho scritto più volte ho sperimentato moltissime volte tutto ciò che ho scritto, senza avere problemi (se non quelli indicati e risolti). Tuttavia non posso escludere di aver commesso qualche errore nello spiegare le varie tecniche. Inoltre le foto riportano collegamenti perfetti ma le loro angolazioni potrebbero trarre in inganno, ecco perché è INDISPENSABILE seguire le istruzioni testuali e non rifarsi soltanto alle immagini.

Ho messo diversi avvisi circa il rischio di danneggiare le board Arduino (e compatibili) o i singoli componenti, in caso di errore nelle connessioni.

Comunque sia non posso e non voglio assumermi alcuna responsabilità per eventuali danni, di qualsiasi natura (apparecchiature, attrezzature, personale, ecc),

che i lettori dovessero subire seguendo quanto scritto in questa Guida, che è gratuita ed a disposizione di chiunque, ma che è pur sempre un supporto amatoriale scritto da un hobbysta.

Elenco sintetico degli aggiornamenti della versione 4.0

- Aggiunti collegamenti ISP per le Arduino MEGA
- Sostituita vecchia tecnica anti-autoreset (R-C) con nuova versione (C)
- Aggiornamento per l'uso dell'IDE 1.0.1 per operazioni ISP (caricamento bootloader o sketch)
- Aggiornamento definitivo sul rapporto bootloader/sketch nella tecnica ISP, per le varie versioni di clock
- Cenni sulla programmazione di altri microcontrollori ATMEL con esempio per IDE 0022 e 1.0.1
- Cenni sui fuse bits
- I convertitori USB→Seriale commerciali e autocostruiti

Sono in tutto circa 25 pagine di aggiornamento.

Cap. 1 Materiale necessario

Cosa serve per portare a buon fine le prossime indicazioni? Potete scegliere, in base alla vostra "dotazione", uno dei seguenti gruppi di materiale:

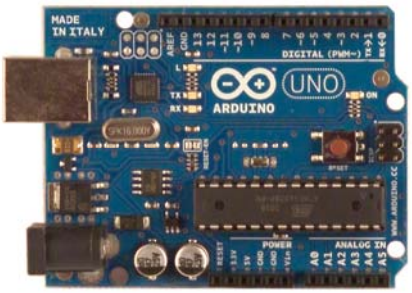
- **Per chi dispone di una sola board Arduino**:
    - Una **board** Arduino UNO, oppure 2009 o Mega (qualsiasi modello), oppure una board compatibile.
    - Una **breadboard** di buona qualità (quelle scadenti creano falsi contatti che potrebbero vanificare tutto).
    - Un **ATmega328P**[1] (vergine o con bootloader precaricato), un quarzo da 16MHz, due condensatori ceramici da 22pF, una resistenza da 10 kohm.
    - Opzionali, ma potrebbero diventare indispensabili/utili: un condensatore (al tantalio o elettrolitico) da 10µF/16V, una resistenza da 1 kohm, , un led (3 o 5mm, di qualsiasi colore).
    - Opzionali (servono solo se volete costruire il cavo ISP Arduino-Breadboard), un connettore femmina 6 pin (2x3), un connettore maschio da cs 7-8 pin (7-8x1), un connettore femmina 7-8 pin (7-8 pin), un pin maschio singolo, 7 fili di lunghezza sufficiente (almeno 15 cm).
- **Gruppo con due Arduino**:
    - **Due board Arduino** (UNO, 2009, Mega, compatibile) in qualsiasi combinazione, uguali o diverse, non ha alcuna importanza.
    - **Un ATmega328P** (vergine o con bootloader precaricato).
    - Un condensatore (al tantalio o elettrolitico) da 10µF/16V.
    - Opzionali (servono solo se volete costruire il cavo ISP Arduino-Arduino), due connettori femmina 6 pin (2x3) ed un pin maschio singolo, 6 fili di lunghezza sufficiente (almeno 15 cm).

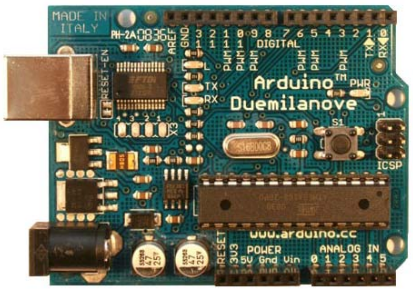
Nella pagina seguente troverete una carrellata di immagini dei materiali descritti in precedenza, dedicata a coloro che non hanno ancora la pur minima dimestichezza con tali

[1] E' importante che il micro sia un ATmega328P (con la P finale) diversamente occorre far riferimento al capitolo specifico per la versione "non P". Variando opportunamente le board virtuali è possibile lavorare senza problemi con micro ATmega88/168, identici al 328P, che fondamentalmente cambiando per la quantità di memoria flash disponibile (rispettivamente 8K e 16K invece dei 32K del 328). Esiste anche l'ATmega8, che appartiene ad altra famiglia ma comunque sempre al mondo Arduino, in questo caso bisogna trattarlo selezionando come board di "destinazione" (e quindi lo specifico bootloader) la voce "Arduino NG o older w/ ATmega8".

componenti. Potrebbe essere utile stamparla e portarla in un negozio di elettronica, sapranno certamente darvi il materiale necessario, sulla base di quanto scritto e delle foto.

Da sinistra a destra: Arduino UNO, Arduino 2009, Luigino 328 (Arduino compatibile)





Resistenze (da sinistra a destra):

10 KOhm (marrone-nero-arancio-oro)

1 KOhm (marrone-nero-rosso-oro)

120 Ohm (marrone-rosso-marrone-oro)



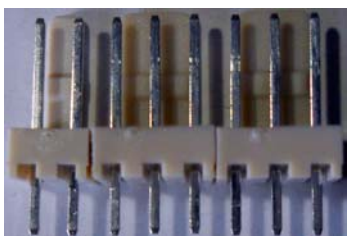
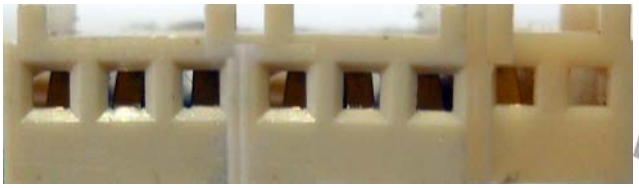
Condensatori e led (da sinistra a destra):

Cond. Elettrolitico 10µF, Cond Tantalio 10µF, Cond. Ceramico 22pF, Led 3mm rosso

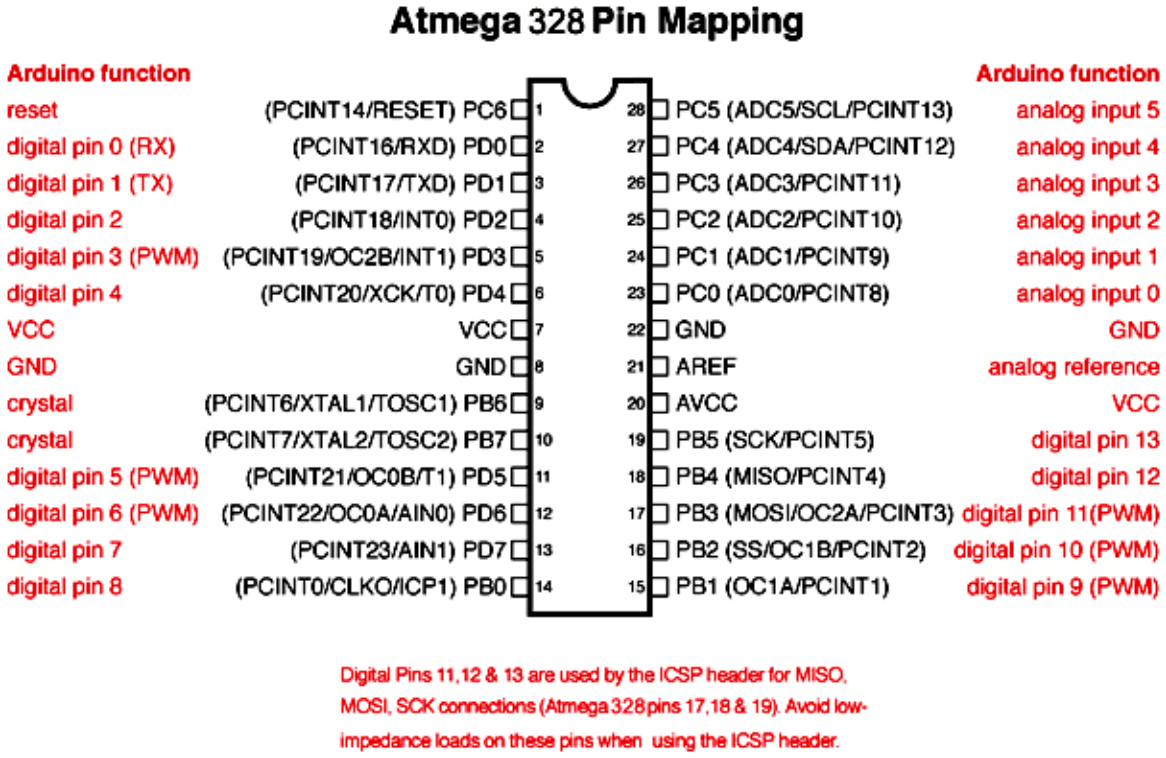

I condensatori elettrolitici e al tantalio ed i led hanno una polarità: il polo “positivo” è sempre quello col pin più lungo; in alcuni casi il “+” o il “-“ sono direttamente stampati sul componente. Vi ricordo che serve un solo condensatore da 10µF, a scelta tra i due.

Connettori (da sinistra a destra):

6 pin femmina (3x2), 7 pin maschio da cs (7x1), 7 pin femmina (7x1), pin maschio singolo


Il chip ATmega328P (ci serve per vedere le posizioni dei pin, i cui numeri sono quelli posti all'interno della figura):



Cap. 2 Procedura per caricare il bootloader su ATmega328P

Nei prossimi passaggi vedremo come portare a buon fine l'operazione; dovrete fare molta attenzione a seguire alla lettera ogni passaggio, non dare niente per scontato, osservare attentamente le foto, una qualsiasi svista non vi permetterà di ottenere il successo sperato. Se avete certezza che i componenti in vostro possesso hanno i valori corretti e sono perfettamente funzionanti abbiate sicurezza che la procedura funzionerà, se non funziona è perché avete sbagliato qualcosa, in questo caso LA PRIMA OPERAZIONE È SCOLLEGARE SUBITO IL CAVO USB, in modo da togliere l'alimentazione al circuito; poi con calma si verificano i collegamenti ed i componenti, per correggere eventuali errori, quindi si può riprovare.

Nota Tecnica iniziale: c'è un concetto che deve essere ben chiaro, su un chip vergine ATmega328P potremo caricare a nostra scelta il bootloader della UNO o della 2009, a prescindere dall'Arduino che usiamo con funzioni di programmatore, infatti il bootloader da caricare nell'ATmega vergine NON viene prelevato dall'Arduino bensì dalle cartelle del software. Inoltre se anche disponiamo di un chip con bootloader precaricato, possiamo sostituirlo tranquillamente, quindi se c'è quello della 2009 possiamo sostituirlo con quello della UNO o viceversa.

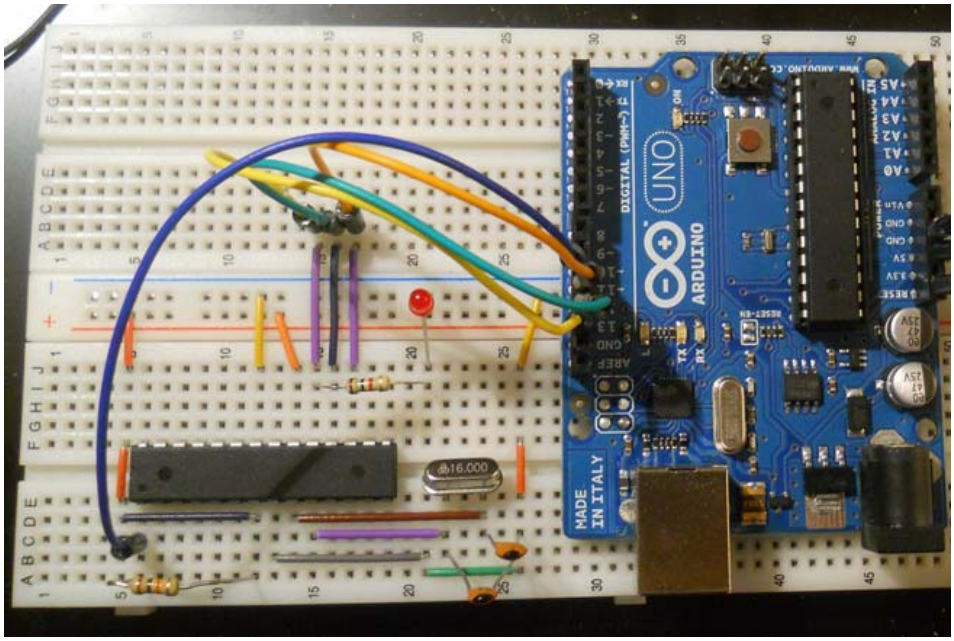
Nelle mie prove ho usato la versione IDE 0022 mentre gli aggiornamenti che troverete riguardano la versione 1.0.1. Sconsiglio vivamente di usare versioni antecedenti alla 0022 e la versione 1.0.

p. 1 – La fase hardware: Arduino e Breadboard

In questa procedura useremo: una board Arduino UNO completa (ma potrebbe essere anche la 2009 o una compatibile, come già detto, le connessioni sono identiche), cioè con a bordo il proprio chip ATmega328P con Bootloader, che svolgerà la funzione di programmatore ISP, una breadboard su cui installiamo il chip ATmega328P vergine (o con un bootloader che vogliamo sostituire) ed i necessari componenti; collegheremo tra loro alcuni pin dell'Arduino UNO e della Breadborad. Vediamo un po' di immagini e descriviamole.

La prima foto è una panoramica del circuito completo; due elementi che possiamo NON montare sono la resistenza da 1 kohm ed il led rosso che vediamo installati nella zona in cui è montato il chip vergine, la cui utilità è esclusivamente quella di rendere visibile l'attività del pin 19 del chip vergine (corrispondente al pin 13 di Arduino), durante le operazioni di scrittura o dopo il caricamento di uno sketch.

Il circuito completo Arduino UNO + Breadboard[2]



Vi elenco i collegamenti da fare, d'ora in poi useremo genericamente i termini **Arduino** riferendoci alla board programmatore e **chip** riferendoci all'ATmega vergine. È

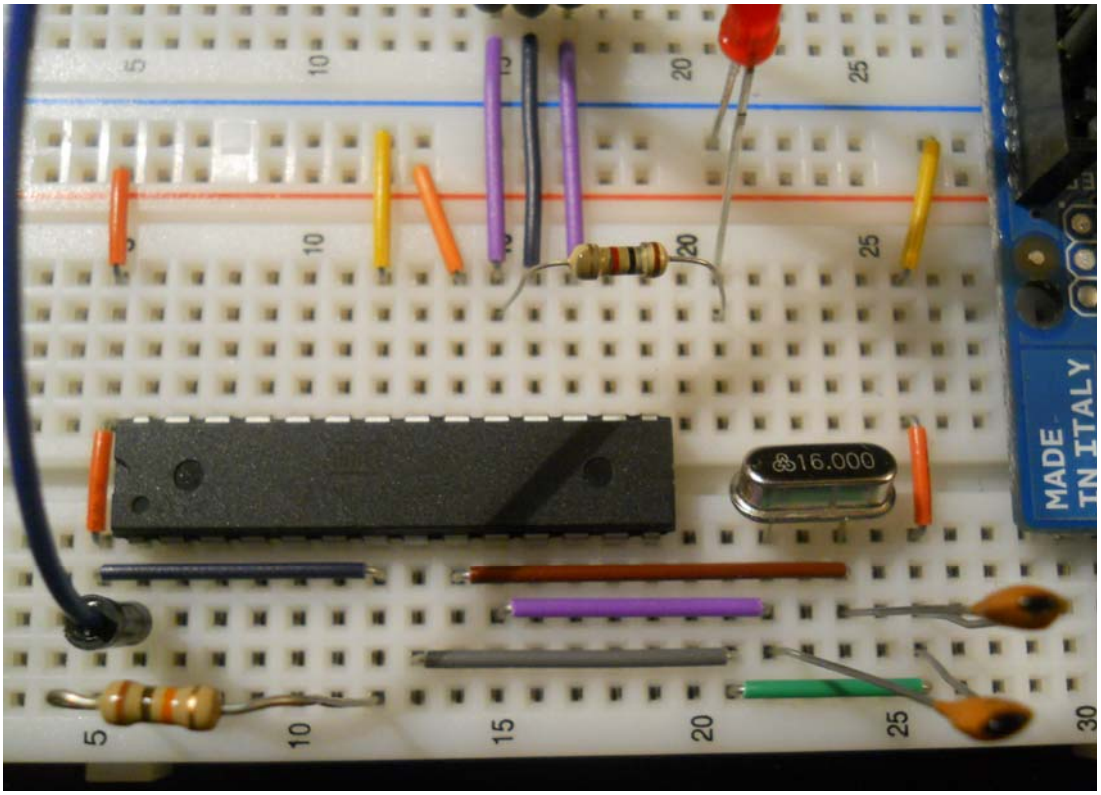
[2] In questa Guida troverete varie versioni di montaggio della breadboard, sono tutte tecnicamente identiche e ugualmente valide e collaudate, ma quella riportata nella sezione dedicata agli sketch è un po' più semplice da realizzare.

preferibile fare prima questi collegamenti e POI collegare Arduino UNO al PC, mediante la porta USB, in modo da operare il meno possibile con le board sotto tensione.[3]

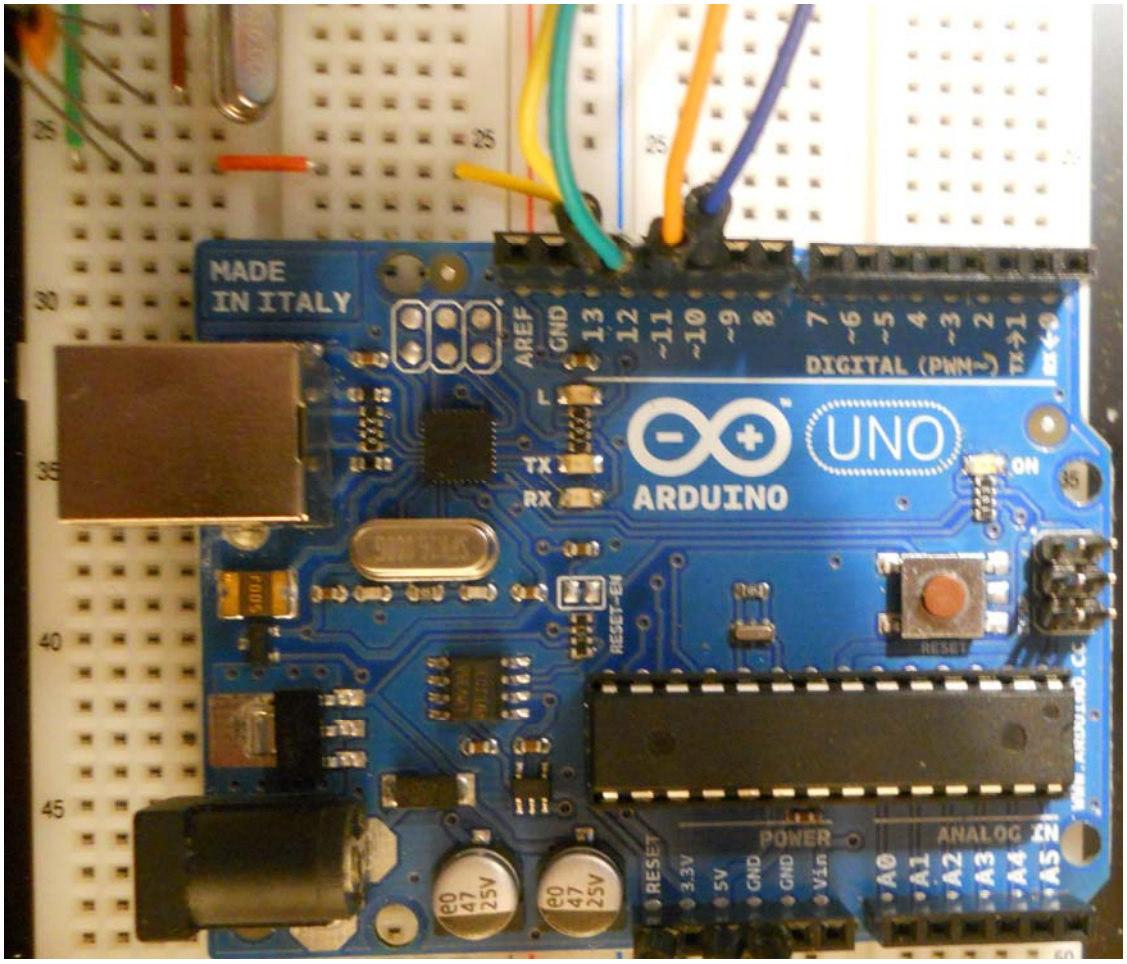
- Il pin 1 del chip (è SEMPRE segnato da un tondino posto alla sinistra della tacca di riferimento del chip, a forma di mezzaluna) va collegato, mediante un filo (nella foto è quello blu), al pin 10 di Arduino. Sempre da questo pin parte una resistenza da 10 kohm che va a finire sui +5V; nell'immagine è collegata al pin 7 dello stesso chip che, a sua volta, va a finire sulla linea +5V (rossa) della breadboard.
- Come appena detto il pin 7 del chip, ma anche il pin 20, vanno collegati ai +5V.
- I pin 8 e 22 del chip vanno collegati sulla linea di massa GND (blu) della breadboard.
- I pin 9 e 10 del chip vanno collegati ai due piedini del quarzo a 16MHz; su ognuno di questi due pin va inoltre collegato un condensatore ceramico da 22pF che con l'altro capo va a massa.
- Il pin 17 del chip va collegato al pin 11 di Arduino (filo arancione).
- Il pin 18 del chip va collegato al pin 12 di Arduino (filo verde).
- Il pin 19 del chip va collegato al pin 13 di Arduino (filo giallo).
- Opzionale: se vogliamo replicare il led 13 di Arduino sul chip colleghiamo al pin 19 del chip una resistenza da 1 kohm, il cui altro capo va in un punto libero della breadboard, qui collegheremo anche il pin lungo del led, il cui pin corto va sulla linea GND (blu).
- A questo punto si può collegare uno dei 3 pin GND di Arduino alla linea GND (blu) della breadboard (filo nero); infine si collega il +5V di Arduino alla linea +5V (rossa) della breadboard (filo rosso).

[3] Nel capitolo 6 di questa stessa Guida troverete le istruzioni per costruire un cavetto ISP, molto più comodo per effettuare i collegamenti tra Arduino e la breadboard; se però non volete realizzarlo seguite questa sezione, funzionerà tutto ugualmente.

I collegamenti del chip



I collegamenti di Arduino UNO



Bene, arrivati a questo punto diamo una ricontrollata (senza fretta!) ai collegamenti, alla posizione del chip vergine (il pin 1 si trova sul capo opposto rispetto alla board Arduino, se lo invertiamo corriamo il rischio di fare seri danni!) e dei componenti aggiuntivi.

È tutto ok? Ottimo, allora abbiamo terminato la fase hardware, **ora colleghiamo la board all'USB del PC** e possiamo passare alla fase software.

Nel prossimo paragrafo vedremo i collegamenti da fare nel caso disponessimo di 2 Arduino (la cosa diventa ancora più semplice); quindi se avete appena realizzato il collegamento tra Arduino e breadboard potete passare direttamente al paragrafo 3, relativo alla fase software; in caso contrario dovete seguire le operazioni del paragrafo 2 e poi passare a quelle del paragrafo 4.

1A – Per chi possiede Arduino Mega (qualsiasi versione)[4]

Queste poche righe sono rivolte ai possessori di una board Arduino Mega (qualsiasi tipo). Dal punto di vista tecnico non cambia nulla, ma le board Mega hanno molti più header a disposizione ed i segnali ISP sono collegati a pin diversi rispetto ad UNO e 2009.

Vediamo rapidamente la corrispondenza dei segnali (per le altre informazioni sui collegamenti del chip far riferimento a quanto scritto finora):

- Il pin 1 del chip al pin 53 di Arduino MEGA (filo blu).
- Il pin 17 del chip va collegato al pin 51 di Arduino (filo arancione).
- Il pin 18 del chip va collegato al pin 50 di Arduino (filo verde).
- Il pin 19 del chip va collegato al pin 52 di Arduino (filo giallo).

Non c'è altro, fatti questi collegamenti il resto è identico.

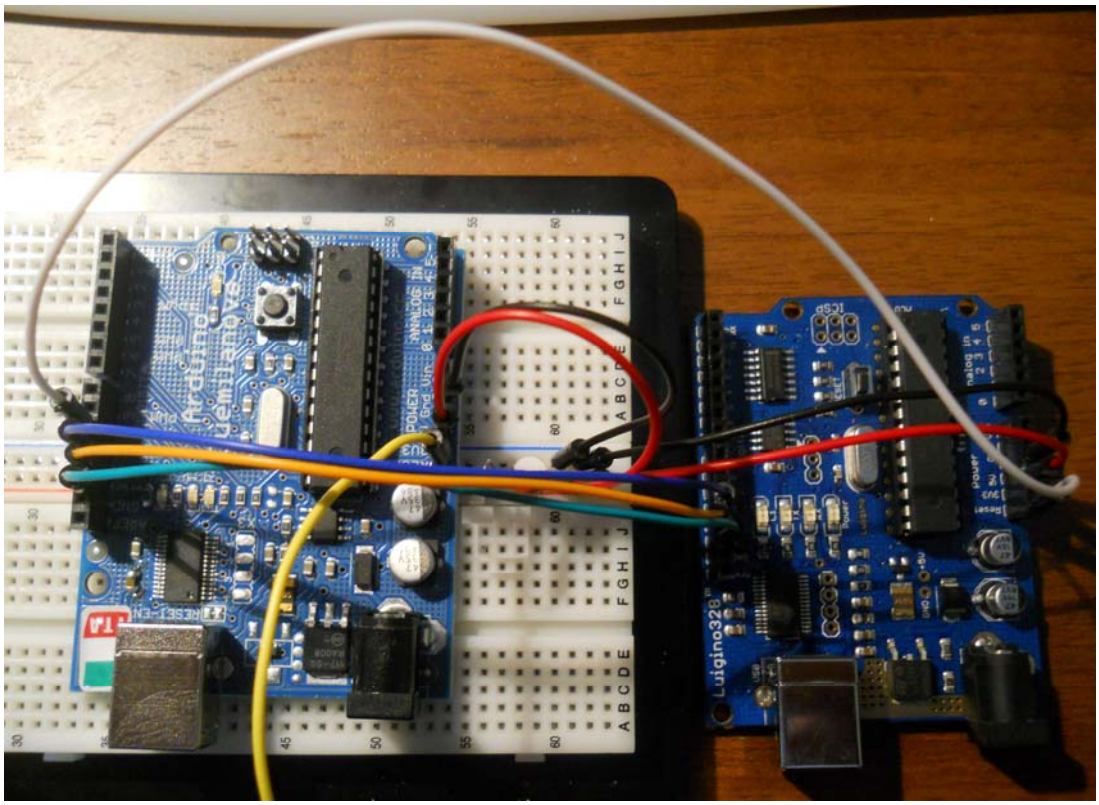
[4] Tra i tanti Topic aperti sul Forum per trattare questi ed altri argomenti, mi è successa una cosa curiosa. Un Utente (nick: **Pablos**), dopo che abbiamo passato alcune serate "virtuali" a risolvere un fastidiosissimo problema di programmazione della sua Arduino Mega ADK, quando finalmente lo risolvemmo mi disse che, siccome nel frattempo aveva comprato una scheda nuova, mi spediva quella oggetto di tante prove, così avrei potuto continuare a sperimentare e magari integrare la Guida con informazioni utili anche per questa scheda. Pensavo scherzasse ed invece eccomi qui a mantenere una promessa fatta ed a ringraziarlo per questo simpatico gesto.

p. 2 – La fase hardware: Arduino & Arduino

In questa procedura useremo: una board Arduino 2009 completa (ma potrebbe essere una UNO o una compatibile), cioè con a bordo il proprio chip ATmega328P con bootloader, che svolgerà la funzione di programmatore, una compatibile (anche in questo caso può essere un altro modello) con funzione "target"; quest'ultima board in pratica svolgerà la stessa funzione della breadboard nel 1° paragrafo, cioè semplicemente di supporto all'ATmega vergine; **per ottenere questa funzione dobbiamo preventivamente togliere dalla compatibile il proprio chip e montare al suo posto** (ATTENZIONE alla posizione del pin1, nel dubbio osservate attentamente le varie foto!!!) **il chip ATmega328P vergine** (o quello con un bootloader che vogliamo sostituire); avere a disposizione una seconda board ci evita molti dei collegamenti della breadboard, in pratica dovremo collegare tra le due board solo alcuni pin. Anche in questo caso faremo uso di immagini altamente esplicative e ben descritte.

Il software non può stabilire da sé qual è la board "programmatore", dobbiamo dirglielo noi, e lo facciamo nell'unico modo possibile, cioè collegando tale board al PC, mediante il cavo USB e poi fornendo le opportune istruzioni software a tale scopo.

Il circuito completo Arduino 2009 (a sinistra) + compatibile con chip vergine (a destra)[5]



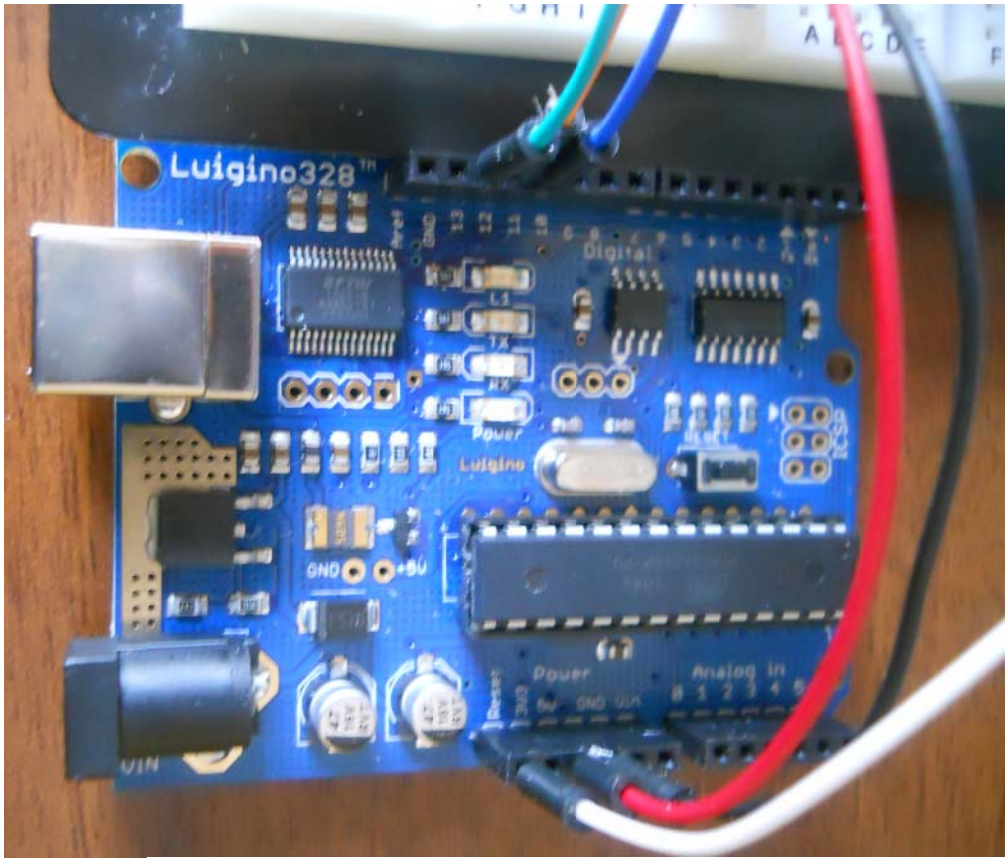
[5] Qui troverete le istruzioni per costruire un cavetto ISP, molto più comodo per effettuare i collegamenti tra due Arduino; se però non volete realizzarlo seguite questa sezione, funzionerà tutto ugualmente.

Da notare che il filo giallo del reset della 2009 è scollegato. Inoltre la in realtà non serve, io l'ho usata solo per come ponte per l'alimentazione.

Vi elenco i collegamenti da fare, d'ora in poi useremo genericamente i termini **2009** riferendoci alla board programmatore e **Compatibile** riferendoci alla board con l'ATmega vergine. È preferibile fare prima questi collegamenti e POI collegare la 2009 al PC, mediante la porta USB, in modo da operare il meno possibile con le board sotto tensione.

- Il pin 10 della 2009 va collegato, mediante un filo (nella foto è quello bianco), al pin Reset della Compatibile.
- Il pin 11 della 2009 va collegato al pin 11 della Compatibile (filo blu).
- Il pin 12 della 2009 va collegato al pin 12 della Compatibile (filo arancione).
- Il pin 13 della 2009 va collegato al pin 13 della Compatibile (filo verde).
- A questo punto si può collegare uno dei 3 pin GND della 2009 alla linea GND (blu) della breadboard (filo nero) e ad uno dei 3 pin GND della Compatibile; infine si collega il +5V della 2009 alla linea +5V (rossa) della breadboard (filo rosso) ed al pin +5V della Compatibile.

I collegamenti della Compatibile



Bene, arrivati a questo punto diamo una ricontrollata (senza fretta!) ai collegamenti, alla posizione del chip vergine (il pin 1 deve trovarsi accanto alla presa ISP della Compatibile, se lo invertiamo corriamo il rischio di fare seri danni!). Se è tutto ok abbiamo terminato la fase hardware, **ora colleghiamo la board all'USB del PC** e possiamo passare alla fase software.

Se abbiamo appena realizzato questa struttura hardware possiamo passare direttamente al paragrafo 4, in cui è esposta la relativa fase software.

2A – Collegare Arduino Mega ad Arduino UNO/2009

Per chi possiede una Arduino Mega (qualsiasi versione) ed una Arduino UNO/2009 esistono due possibilità: usare la prima come Programmatore e la seconda per il chip da programmare o viceversa. Vediamo entrambi i casi:

2A1 – Arduino Mega Programmatore e Arduino UNO/2009 con micro vergine

- Il pin 53 della MEGA va collegato al pin Reset della UNO/2009.
- Il pin 51 della MEGA va collegato al pin 11 della UNO/2009.
- Il pin 50 della MEGA va collegato al pin 12 della UNO/2009.
- Il pin 52 della MEGA va collegato al pin 13 della UNO/2009.
- A questo punto si può collegare uno dei pin GND della MEGA ad uno dei pin GND della UNO/2009; infine si collega il +5V della MEGA al pin +5V della UNO/2009.

2A2 – Arduino UNO/2009 Programmatore e Arduino Mega da programmare

Questo è un caso particolare; infatti le board MEGA non hanno il microcontrollore ATmega328P, bensì un ATmega1280 o 2560, in formato SMD e saldato direttamente sulla board. Questo collegamento quindi serve solo nel caso in cui si danneggi il bootloader dell'Arduino Mega o si sostituisca il suo micro con uno nuovo e quindi vergine.

- Il pin 10 della UNO/2009 va collegato al pin Reset della MEGA.
- Il pin 11 della UNO/2009 va collegato al pin 51 della MEGA.
- Il pin 12 della UNO/2009 va collegato al pin 50 della MEGA.
- Il pin 13 della UNO/2009 va collegato al pin 52 della MEGA.
- A questo punto si può collegare uno dei pin GND della UNO/2009 ad uno dei pin GND della MEGA; infine si collega il +5V della UNO/2009 al pin +5V della MEGA.

2B – Collegare Arduino Mega ad un’altra Arduino Arduino Mega

Per chi possiede due Arduino Mega (qualsiasi versione), una volta scelta quella che fa da programmatore (la chiamiamo MEGA1) e quella che fa da target (la chiamiamo MEGA2), i collegamenti sono questi:

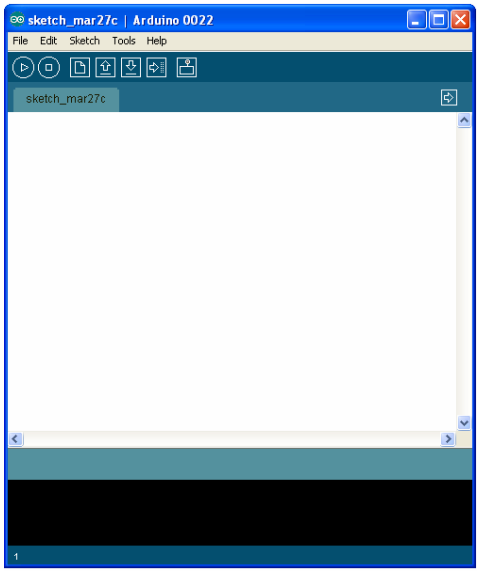
- Il pin 53 della MEGA1 va collegato al pin Reset della MEGA2.
- Il pin 51 della MEGA1 va collegato al pin 51 della MEGA2.
- Il pin 50 della MEGA1 va collegato al pin 50 della MEGA2.
- Il pin 52 della MEGA1 va collegato al pin 52 della MEGA2.
- A questo punto si può collegare uno dei pin GND della MEGA1 ad uno dei pin GND della MEGA2; infine si collega il +5V della MEGA1 al pin +5V della MEGA2.

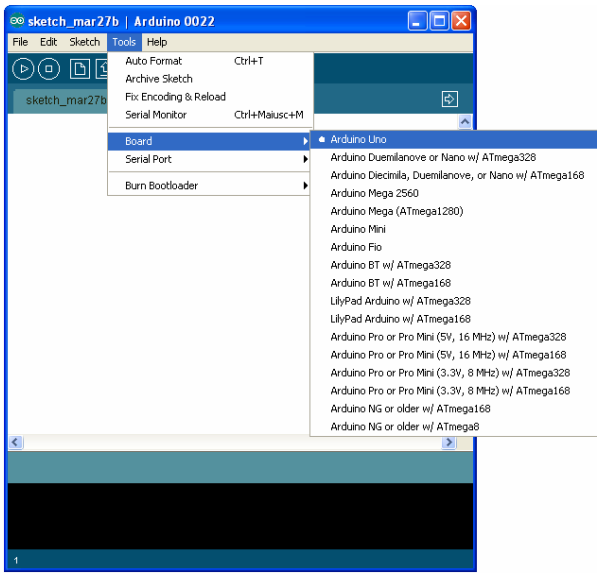
p. 3 – La fase software: Arduino e Breadboard

In questo paragrafo vedremo come effettuare le necessarie operazioni software per arrivare al risultato sperato; in realtà, in caso di possibili (ma previsti) problemi dovremo di nuovo mettere mano all'hardware (tecnica di disabilitazione dell'autoreset).

Questa fase si compone di una serie di passaggi obbligati che vanno eseguiti secondo una sequenza ben precisa, se si tenta di invertire qualche passaggio si potrebbe compromettere il successo dell'operazione.

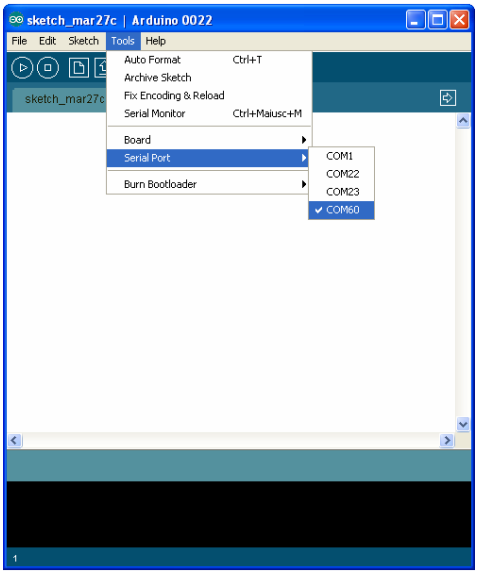
a. Ribadisco che bisogna usare la versione IDE 0022 (v. anche paragrafo specifico se si usa la 1.0.1), che a questo punto andiamo ad eseguire, ottenendo la tipica videata iniziale:





b. Dobbiamo ora selezionare la board che stiamo usando come "programmatore" (quella collegata all'USB del PC), dal menu Tools – Board; nel nostro caso stiamo usando la UNO quindi selezioniamo la relativa voce (v. immagine precedente).

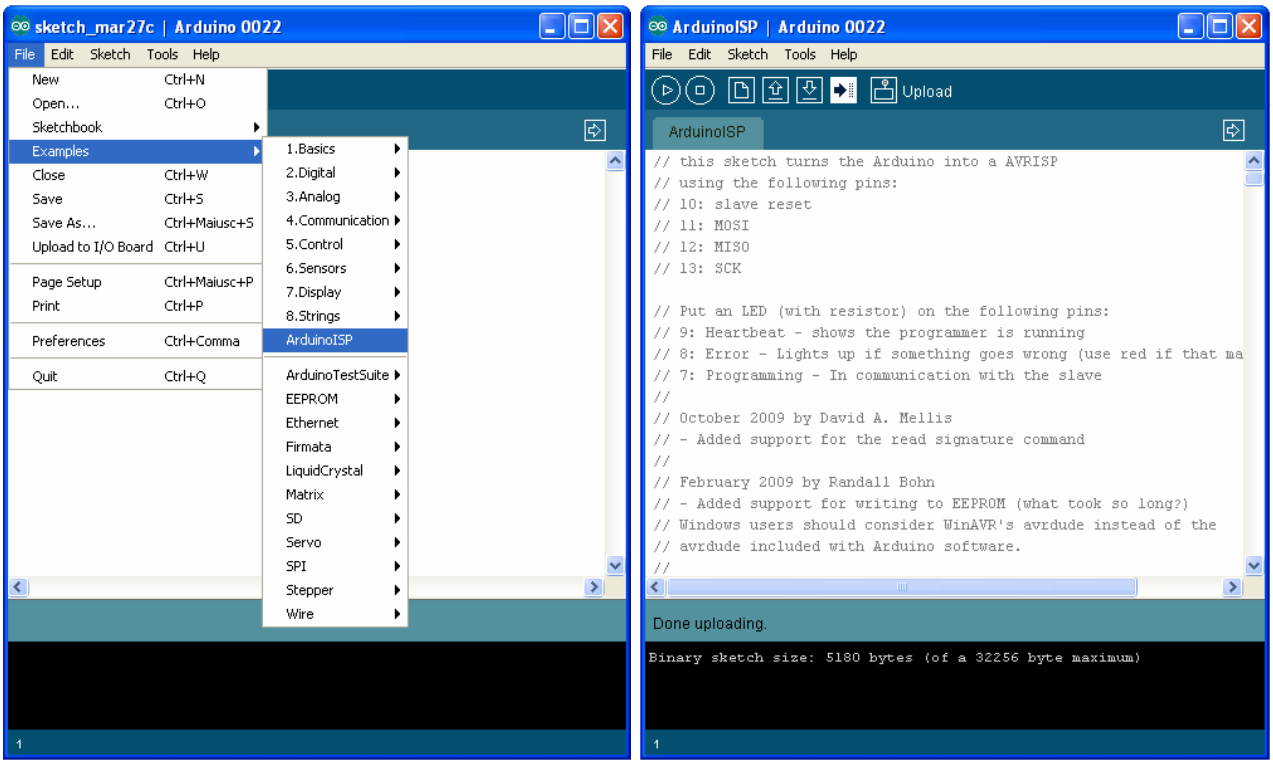
c. A questo punto dobbiamo selezionare la COM che corrisponde alla seriale/USB a cui è collegata la board (in questo nostro esempio è la COM 60)[6].



d. Ora dotiamo la nostra board delle funzionalità

[6] La sigla COM identifica le porte seriali in ambiente Windows; con altri Sistemi Operativi le stesse porte sono denominate diversamente; p.es. in Linux, che distingue tra il chip FT232RL ed il chip ATmega8U2, definisce le COM rispettivamente /dev/ttyUSB o /dev/ttyACM.

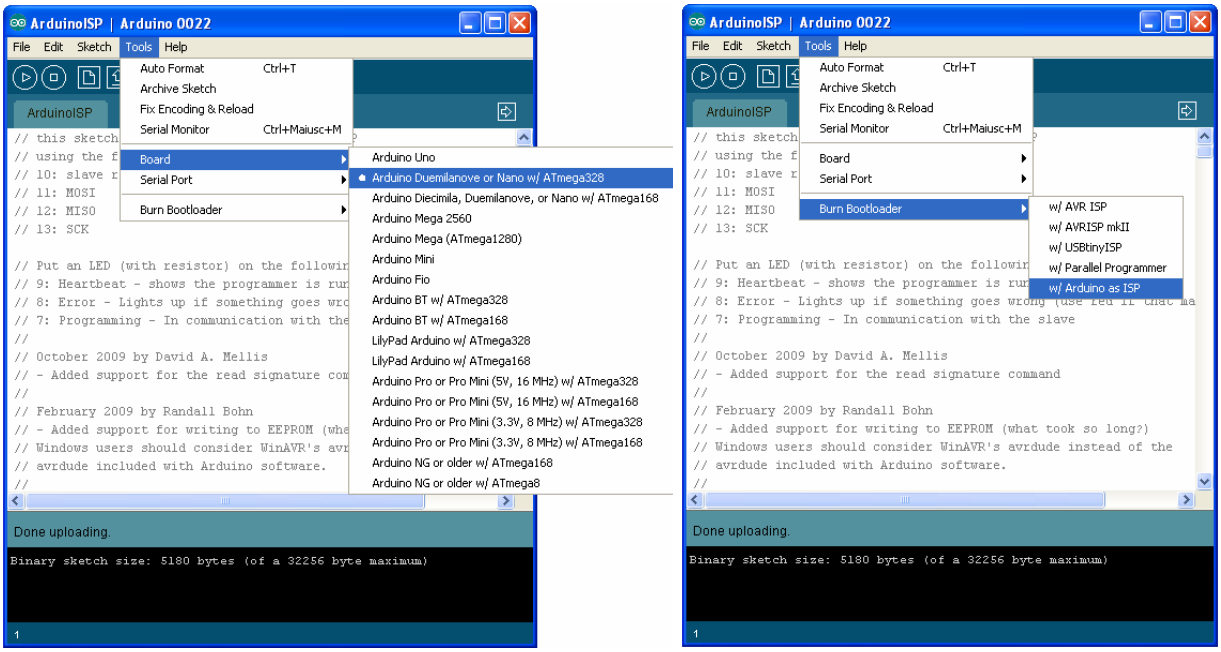
di programmatore, selezioniamo lo sketch (programma) “ArduinoISP” dal menu File – Examples e lo inviamo alla board Arduino programmatore con la normale operazione di UPLOAD.[7]



e. Arrivati a questo punto dobbiamo decidere quale bootloader caricare nell’ATmega vergine; fatta la scelta dobbiamo aprire nuovamente il menu Tools – Board e selezionare la relativa board; poiché qui di solito si genera confusione è meglio spendere qualche parola in più: nel precedente punto “b” abbiamo selezionato Arduino UNO in quanto è il tipo di board che abbiamo fisicamente collegato alla USB del PC ed è quella che usiamo come programmatore; ora invece dobbiamo dire al software qual è la board di destinazione, cioè quella che contiene il chip vergine, in questo momento è una breadboard che sta facendo le “veci” di un’altra board Arduino; scegliendo un modello il software caricherà nell’ATmega vergine il bootloader di quel modello; naturalmente se decidiamo di caricare il bootloader della UNO, nel nostro caso non dobbiamo cambiare niente, in quanto è come se usassimo due Arduino UNO per fare l’operazione; ma noi invece ora vogliamo avere un chip con il bootloader della 2009, ecco perché apriamo il menu Tools – Board e scegliamo Arduino 2009 (v. immagine dopo il punto “f”).

[7] Alcune volte succede che, pur avendo settato correttamente la board di programmazione, non si riesce ad inviarle lo sketch ArduinoISP, ricevendo un errore “avrdude: stk500_getsync(): not in sync: resp=0x00 avrdude: stk500_disable(): protocol error, expect=0x14, resp=0x51”; in questo caso probabilmente avete già collegato il condensatore dell’anti-autoreset, bloccando di fatto il Reset; staccate questo componente e riprovate.

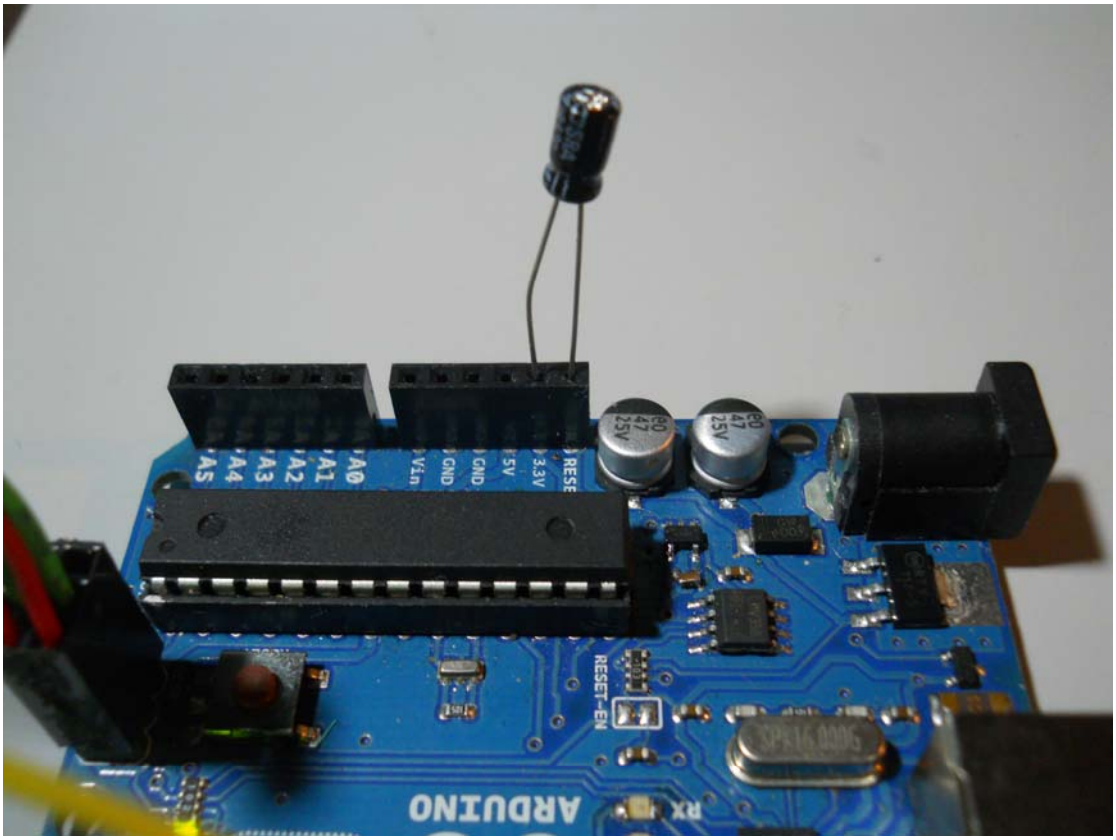
f. Ora c'è da fare l'ultimo passaggio, ordinare al software di eseguire l'operazione di caricamento del bootloader nel chip vergine. Apriamo quindi il menu Tools – Burn bootloader e scegliamo l'opzione "w/Arduino as ISP":



Quasi istantaneamente vedremo un rapido lampeggio dei 3 led della board Arduino UNO e qui possono succedere due cose:

- Dopo qualche istante i 3 led cominciano a lampeggiare allegramente (complessivamente l'operazione può durare da 30 secondi a quasi un minuto, in base alla configurazione hardware ed al bootloader che si è scelto di caricare sul chip vergine), quando tutto si spegne l'operazione è terminata con successo!!!
- Oppure il led 13 fa un solo ulteriore lampeggio e poi non succede più nulla, dopo qualche istante nella sezione "nera" del software appare un messaggio d'errore, che probabilmente è questo: "**avrdude: stk500_getsync(): not in sync: resp=0x15**". Niente paura, questo è l'autoreset che agisce in modo troppo repentino, ma noi abbiamo il rimedio pronto! È arrivato il momento di collegare il condensatore da 10µF col polo + al pin 3,3V dell'Arduino usato come programmatore ed il pin – al pin Reset dello stesso Arduino UNO.

Il sistema anti-autoreset col condensatore da 10µF



g. Ora possiamo ritentare l'operazione ripartendo direttamente dal punto "f", vedrete che questa volta l'"allegro lampeggio" ci sarà! [8]
h. A questo punto il nostro circuito è pronto per programmare un altro chip, possiamo togliere quello appena programmato e sostituirlo con un altro. Questa operazione di sostituzione è bene farla togliendo prima l'alimentazione (quindi scollegando l'USB) per evitare di causare possibili danneggiamenti alla board o al chip stesso. Naturalmente per programmare un altro chip non è necessario rifare tutti i passaggi, basta ricollegare l'USB e ripartire dal punto "e" (se vogliamo cambiare il tipo di bootloader) o dal punto "f" (se vogliamo usare lo stesso bootloader caricato sul chip precedente).

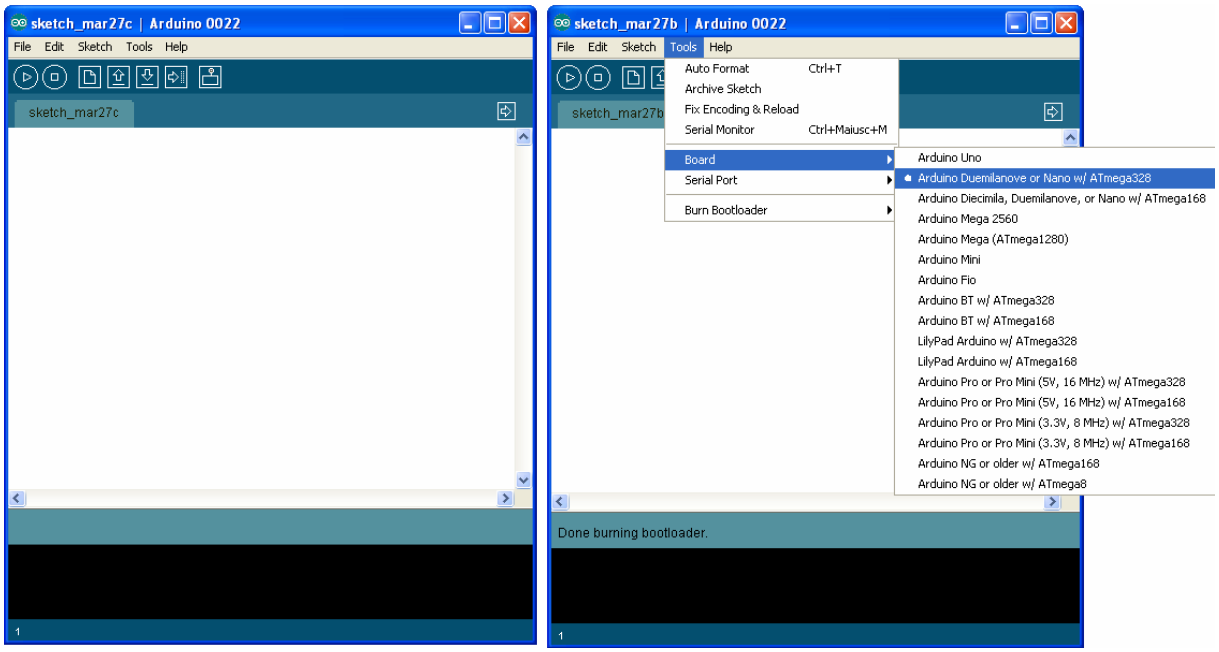
[8] Nei vari Topic visitati ho potuto constatare che tale problema si verifica o non si verifica indifferentemente, su qualsiasi Arduino, per cui Vi assicuro che non è "legge" il fatto che su 2009 non si debba disabilitare l'autoreset e su UNO sì, o viceversa. Semplicemente, se l'operazione non va a buon fine bisogna disabilitare l'autoreset, a prescindere dall'hardware che si sta usando; questo mi hanno insegnato le decine e decine di prove che ho dovuto fare per poter produrre questo lavoro. Chiarisco che se Arduino necessita di questo intervento per poter portare a buon fine l'operazione, non c'è speranza che se ne possa fare a meno, quindi la prossima volta bisognerà montare direttamente il condensatore, senza fare inutili tentativi. Per la verità più di qualcuno ha sperimentato che installando sul micro dell'Arduino UNO/2009 il nuovo bootloader "Optiboot", presente nella versione IDE1.0.1, il problema dell'autoreset si risolve spontaneamente, ma non ho mai provato a fare questa operazione.

p. 4 – La fase software: Arduino & Arduino

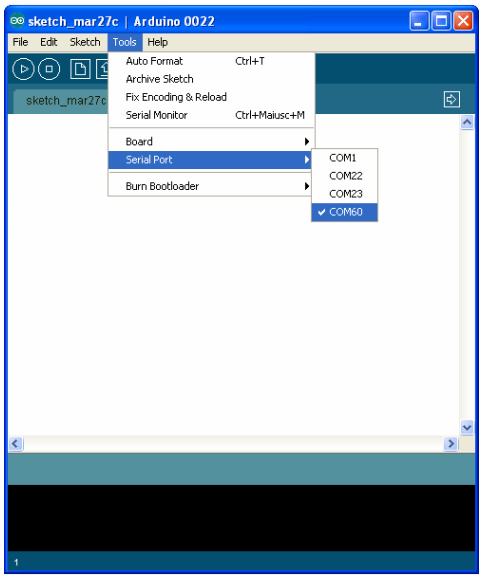
Questo paragrafo non è altro che la replica del precedente applicata al metodo hardware con due Arduino, ma preferisco ripetere (con le dovute modifiche) in modo che ognuno possa eventualmente scaricare o stampare solo le sezioni di interesse.

Vedremo come effettuare le necessarie operazioni software per arrivare al risultato sperato; in realtà, in caso di possibili (ma previsti) problemi dovremo di nuovo mettere mano all'hardware (l'ormai nota disabilitazione dell'autoreset), ma se avremo a portata di mano il condensatore da 10µF non dobbiamo preoccuparci di nulla.

a. Non mi stancherò di ripetere che non bisogna usare versioni dell'IDE antecedenti alla 0022, che a questo punto andiamo ad eseguire, ottenendo la tipica videata iniziale:



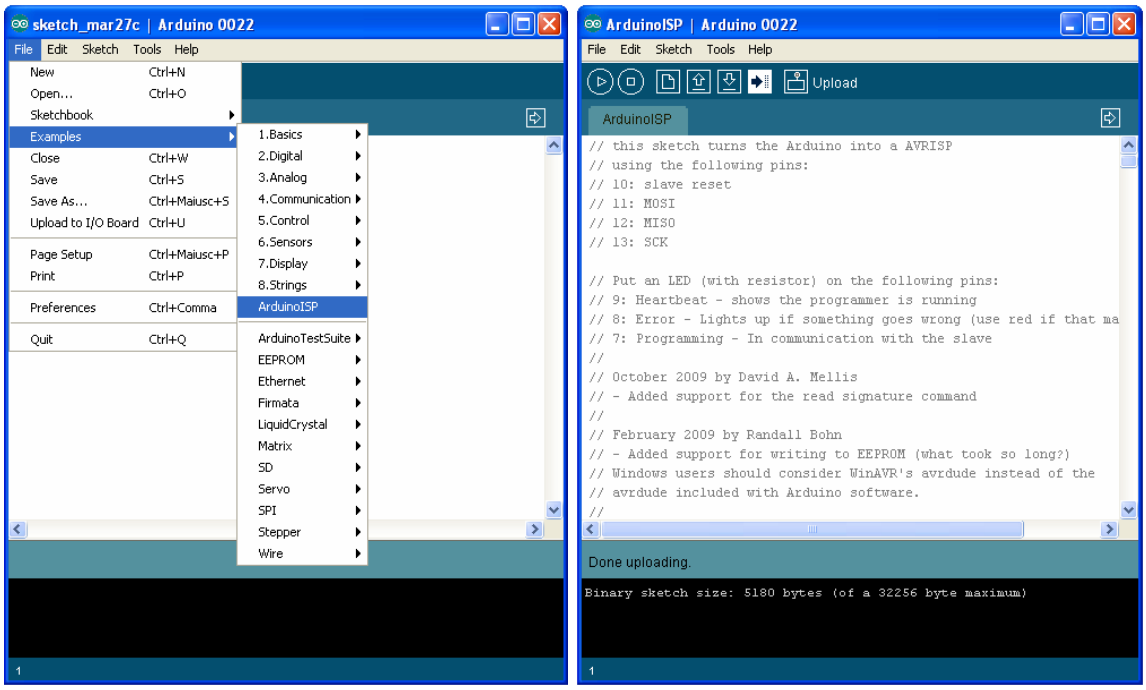
b. Dobbiamo ora selezionare la board che stiamo usando come "programmatore" (Vi ricordo che tra le due è quella fisicamente collegata all'USB del PC), dal menu Tools – Board; nel nostro caso stiamo usando la 2009 quindi selezioniamo la relativa voce (v. immagine precedente).



c. A questo punto dobbiamo selezionare la COM che corrisponde alla seriale/USB a cui è collegata la board (in questo nostro esempio è la COM 61).

d. Ora dotiamo la nostra board delle funzionalità di

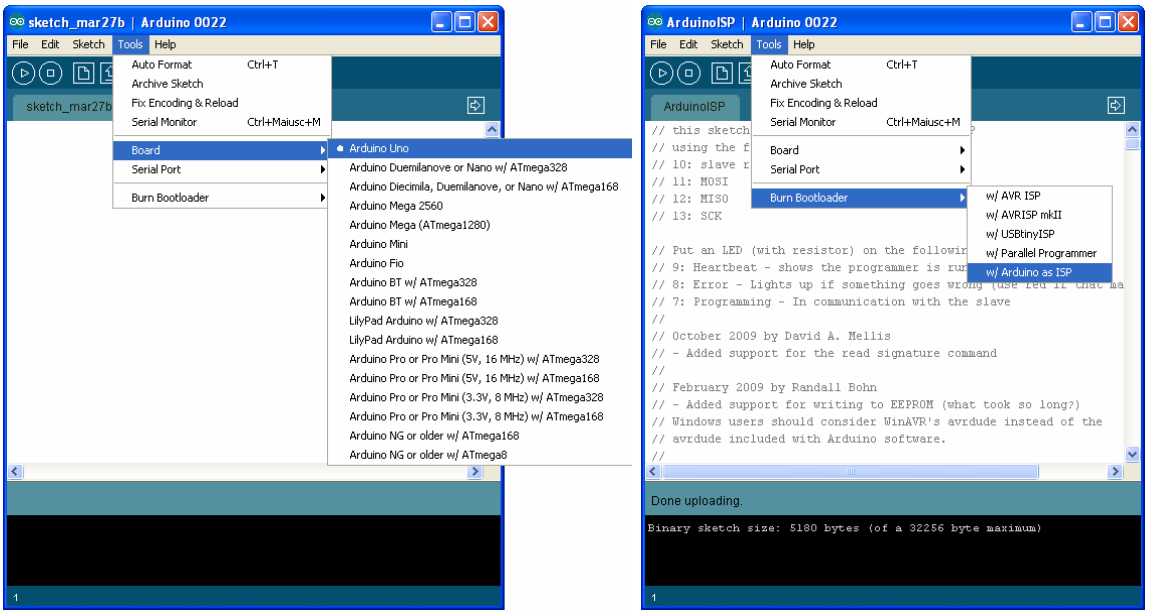
programmatore, selezioniamo lo sketch (programma) “ArduinoISP” dal menu File – Examples e lo inviamo alla board Arduino programmatore con la normale operazione di UPLOAD.



e. Arrivati a questo punto dobbiamo decidere quale bootloader caricare nell’ATmega vergine; fatta la scelta dobbiamo aprire nuovamente il menu Tools – Board e selezionare la relativa Board; poiché anche in questo caso si genera non poca confusione è meglio approfondire la problematica: nel precedente punto “b” abbiamo selezionato Arduino 2009 in quanto è il tipo di board che abbiamo fisicamente collegato alla USB del PC ed è quella che usiamo come programmatore; ora invece dobbiamo dire al software qual è la board di destinazione, cioè quella che contiene il chip vergine, nel nostro caso è una board totalmente compatibile con Arduino 2009, **ma ciò non ha alcuna importanza, è questo il punto della questione!** Dobbiamo infatti avere ben presente che Arduino 2009, Arduino UNO e compatibili sono pressoché intercambiabili tra loro; infatti se io inserisco in una 2009 un chip con bootloader della UNO, il programma è in grado di vedere questa Board SOLO se la configuro come una UNO; viceversa, se metto in una UNO un chip con bootloader della 2009 il programma la vede come una 2009; e così per la Compatibile che ho usato, che viene vista indifferentemente come una 2009 o una UNO **esclusivamente** in funzione del chip che vi monto sopra. Se ci è chiaro questo principio ben comprendiamo come noi non sceglieremo ora LA board realmente usata come target per il chip vergine, bensì QUELLA board di cui vogliamo avere il bootloader sul chip vergine. Infatti,

questa volta, dal menu Tools – Board selezioniamo Arduino UNO (v. immagine dopo il punto “f”).

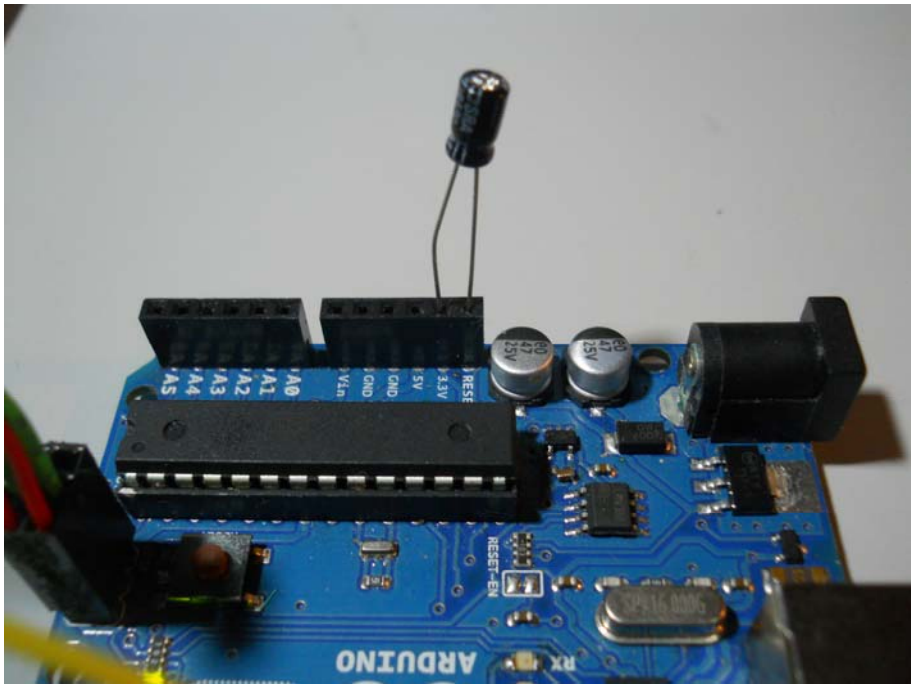
f. Ora c’è da fare l’ultimo passaggio, ordiniamo al software di eseguire l’operazione di caricamento del bootloader nel chip vergine. Apriamo quindi il menu Tools – Burn bootloader e scegliamo l’opzione “w/Arduino as ISP”:



Quasi istantaneamente vedremo un rapido lampeggio dei 3 led della board Arduino 2009 e contemporaneamente del led 13 della Compatibile; qui possono succedere due cose:

- Dopo qualche istante i 3 led (+ led 13 della Compatibile) cominciano a lampeggiare allegramente (complessivamente l’operazione può durare da 15 secondi a quasi un minuto, in base alla configurazione hardware ad al bootloader che si è scelto di caricare), quando tutto si spegne l’operazione è terminata con successo!!!
- Oppure il led 13 fa un solo ulteriore lampeggio e poi non succede più nulla, dopo qualche istante nella sezione “nera” del software appare un messaggio d’errore, che probabilmente è questo: "**avrdude: stk500_getsync(): not in sync: resp=0x15**". Niente paura, questo è l’autoreset che agisce in modo troppo repentino, ma noi abbiamo il rimedio pronto! È arrivato il momento di collegare il condensatore da 10µF col polo + al pin 3,3V dell’Arduino usato come programmatore ed il pin – al pin Reset dello stesso Arduino UNO.

Il sistema anti-autoreset col condensatore da 10µF



g. Ora possiamo ritentare l'operazione ripartendo direttamente dal punto "f", vedrete che questa volta l'"allegro lampeggio" ci sarà![9]

4A – Caricare il bootloader su Arduino Mega

Chi possiede una o due Arduino Mega e desidera caricare il bootloader originale sul micro ATmega1280 o 2560 (in base al modello), deve eseguire le stesse operazioni viste finora, riassunte in:

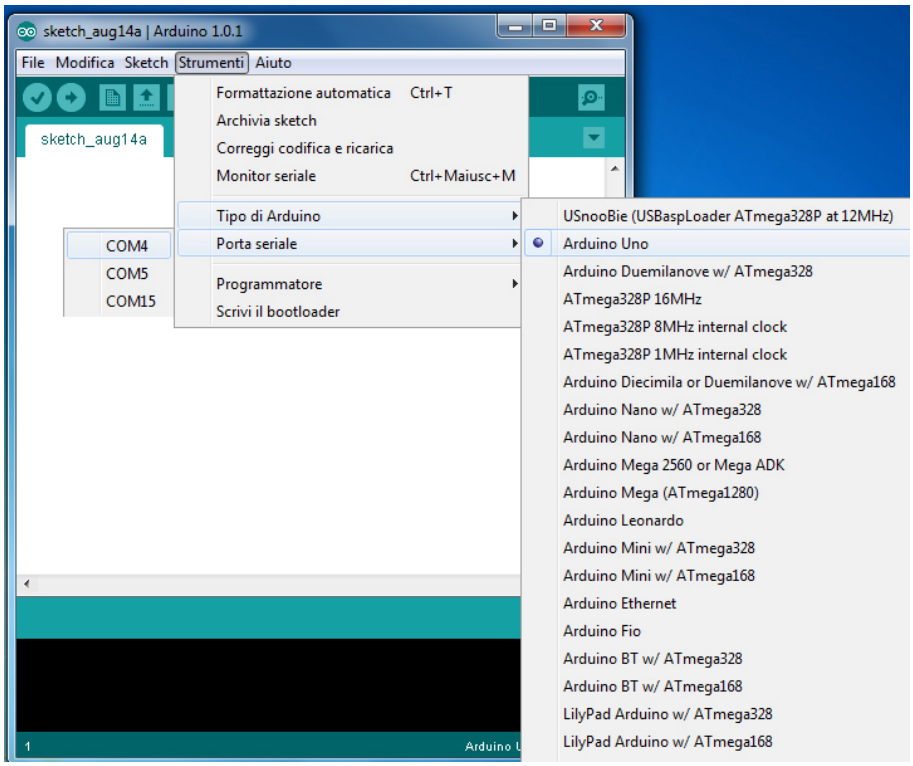
- Selezionare la board usata come Programmatore (quella collegata al PC) e caricarvi lo sketch "ArduinoISP".
- Selezionare il modello di board MEGA di cui si vuole caricare il bootloader sul micro danneggiato o sostituito.
- Aprire il menu Tools – Burn bootloader e scegliere l'opzione "w/Arduino as ISP".

[9] Nei vari Topic visitati ho potuto constatare che tale problema si verifica o non si verifica indifferentemente, su qualsiasi Arduino, per cui Vi assicuro che non è "legge" il fatto che su 2009 non si debba disabilitare l'autoreset e su UNO sì, o viceversa. Semplicemente, se l'operazione non va a buon fine bisogna disabilitare l'autoreset, a prescindere dall'hardware che si sta usando; questo mi hanno insegnato le decine e decine di prove che ho dovuto fare per poter produrre questo lavoro. Chiarisco che se Arduino necessita di questo intervento per poter portare a buon fine l'operazione, non c'è speranza che se ne possa fare a meno, quindi la prossima volta bisognerà montare direttamente il condensatore, senza fare inutili tentativi. Per la verità più di qualcuno ha sperimentato che installando sul micro dell'Arduino UNO/2009 il nuovo bootloader "Optiboot", presente nella versione IDE1.0.1, il problema dell'autoreset si risolve spontaneamente, ma non ho mai provato a fare questa operazione.

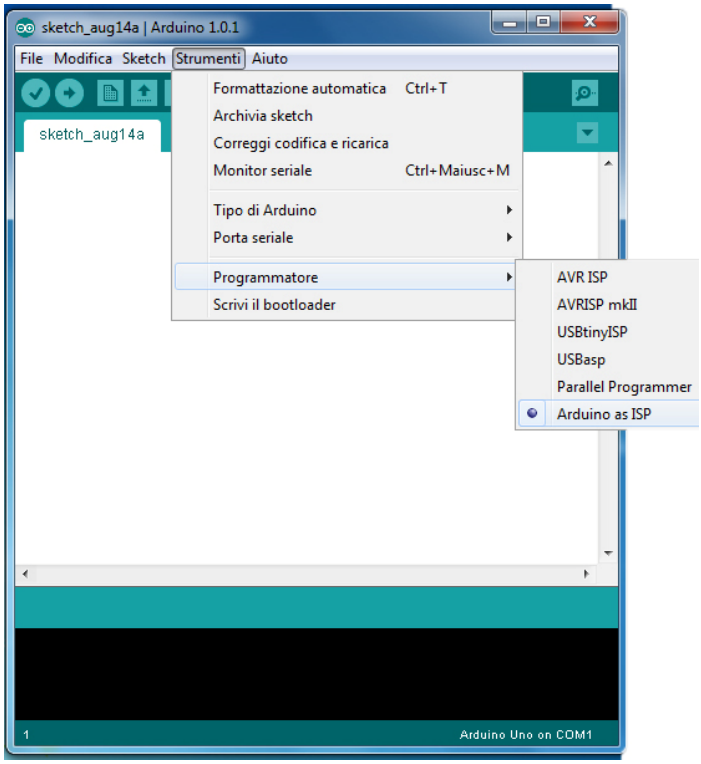
p. 5 – Caricare il bootloader mediante IDE 1.0.1

Questo breve paragrafo rappresenta l'aggiornamento ai due precedenti, che consiglio vivamente di leggere, in quanto molto approfonditi nelle spiegazioni. Qui mi limito a sintetizzare i passaggi già spiegati, con le necessarie variazioni.

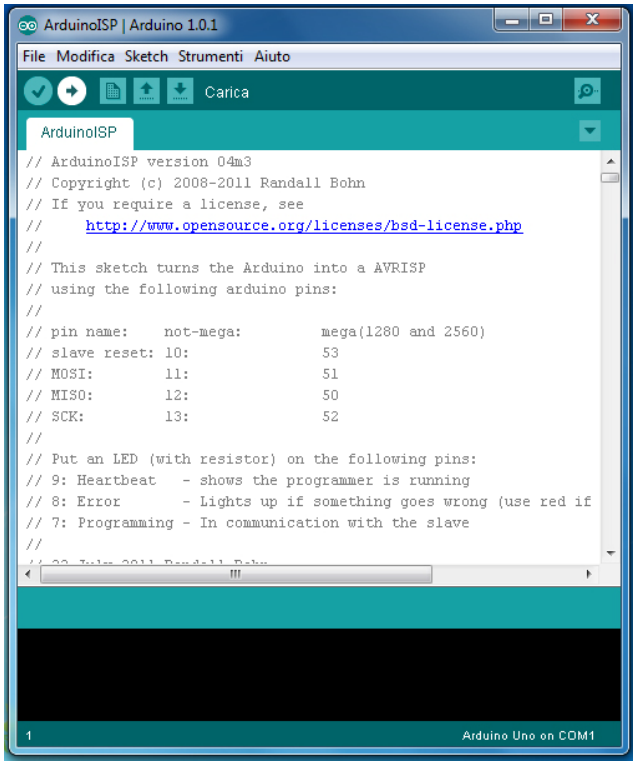
a – Settare la board usata come Programmatore (es. Arduino UNO) e la relativa COM.



b – Settare Strumenti→Programmatore→"Arduino as ISP".

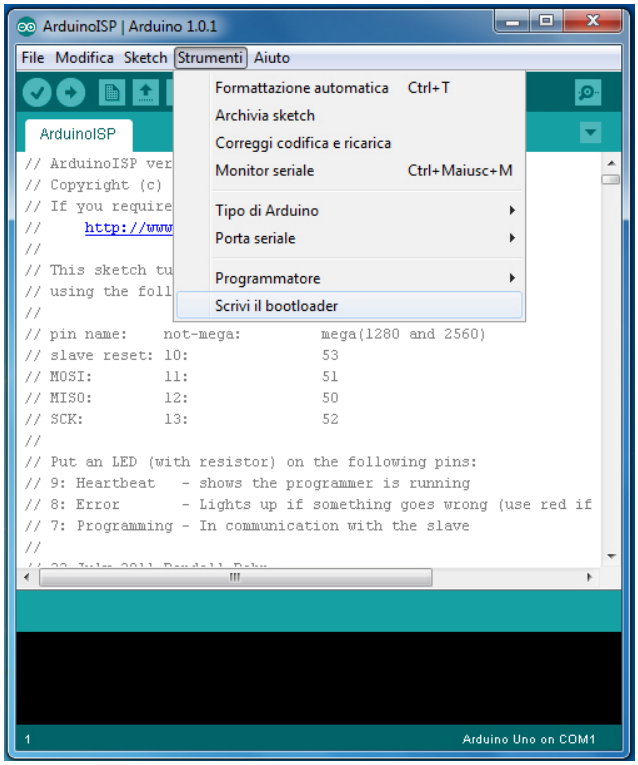


c – Caricare lo sketch ArduinoISP presente negli esempi ed inviarlo all'Arduino UNO



d – Settare ora la board di cui si vuole caricare il bootloader nel micro da programmare (p.es. Arduino UNO, 2009, Mega2560, ecc.).

e – Eseguire il comando Strumenti→Scrivi il bootloader.



Come vedete le operazioni sono pressoché identiche, ma bisogna fare attenzione ad un paio di possibili elementi “di disturbo”. Uno è certamente la versione IDE 1.0, il cui

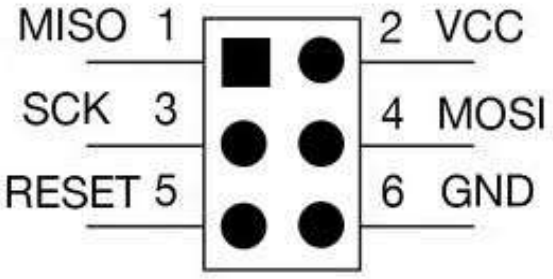
ArduinoISP non funziona e non c’è verso di farlo funzionare, ecco perché dovete scaricare direttamente la versione 1.0.1. L’altro elemento può essere legato alla contemporanea presenza di due diverse versioni di IDE, p.es la 0022 e la 1.0.1. Purtroppo, e non se ne capisce il motivo, le Preferenze di entrambe le versioni, quindi anche la cartella personale dell’IDE, sono scritte nello stesso file di sistema, pertanto si creano conflitti che a volte impediscono il corretto funzionamento di una delle due versioni. Per tale ragione sconsiglio di tenerle entrambe, a meno che non abbiate la giusta dimestichezza per gestirle correttamente.

p. 6 – Semplifichiamo i collegamenti: il connettore ISP

Abbiamo visto finora come i collegamenti tra Arduino e breadboard o tra due Arduino siano effettuati mediante fili con terminali tipo pin; questa pratica è semplice e diretta, ma anche seccante e pericolosa, se scambiamo tra loro i fili dei segnali non succede nulla (ovvio che non funziona), ma se l'errore lo commettiamo con l'alimentazione sono dolori!

Allora, visto che disponiamo di tutto ciò che ci serve (o quasi…) sul **connettore ISP** di Arduino, realizziamo due diversi cavi di connessione (più avanti capiremo perché) che ci permetteranno di collegare rispettivamente due board Arduino o una board Arduino ed una breadboard.

Vediamo quali piedini di ATmega (e corrispondenti pin digitali di Arduino) sono collegati al connettore ISP:



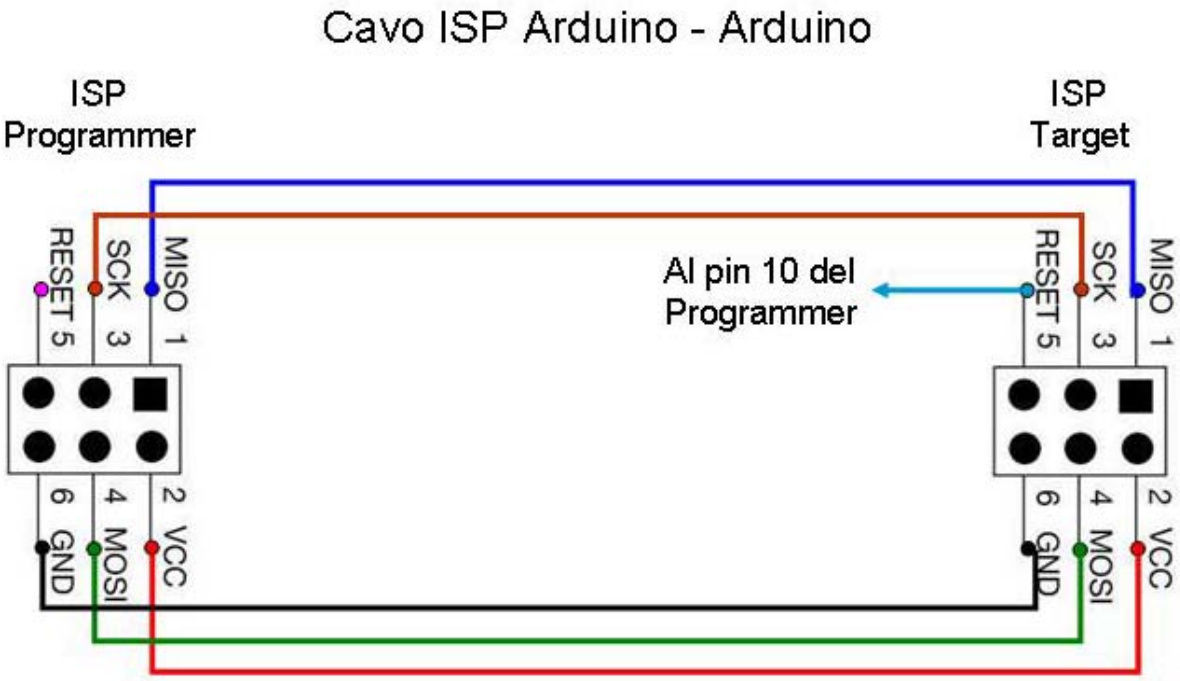
Il pin 1 è facilmente localizzabile su Arduino in quanto è serigrafato sulla board; come vediamo chiaramente abbiamo: il pin MISO (pin digitale 12), l'alimentazione Vcc (pin +5V), il pin SCK (pin digitale 13), il pin MOSI (pin digitale 11), il Reset (pin Reset), la massa (pin GND) .

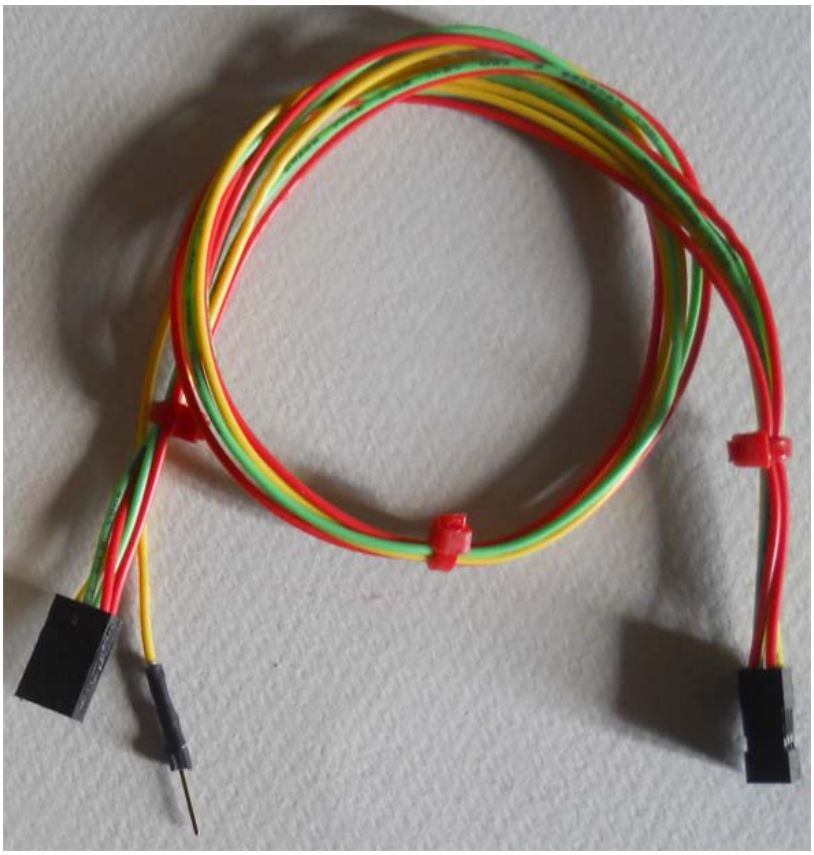
6A - Il cavo Arduino-ISP-Arduino

Iniziamo dal cavo ISP "classico", per collegare due Arduino; dobbiamo tenere in considerazione quanto visto finora, e cioè che mentre su Arduino target ci servono tutti i 6 segnali, su Arduino programmer NON dobbiamo usare il pin Reset; inoltre tutti i segnali vanno collegati pin-to-pin, cioè ogni segnale del target va collegato allo stesso segnale del programmer, ad eccezione del pin Reset del target che, come ormai ben sappiamo, va collegato al pin digitale 10 dell'Arduino programmer; il pin Reset del programmer serve solo, eventualmente, per collegare il condensatore per bloccare l'autoreset.

Per la realizzazione ci servono: due connettori femmina 3x2 a passo 2,54, 6 fili di lunghezza sufficiente (almeno 15 cm), possibilmente tutti di colore diverso, un pin maschio di recupero da un connettore strip.

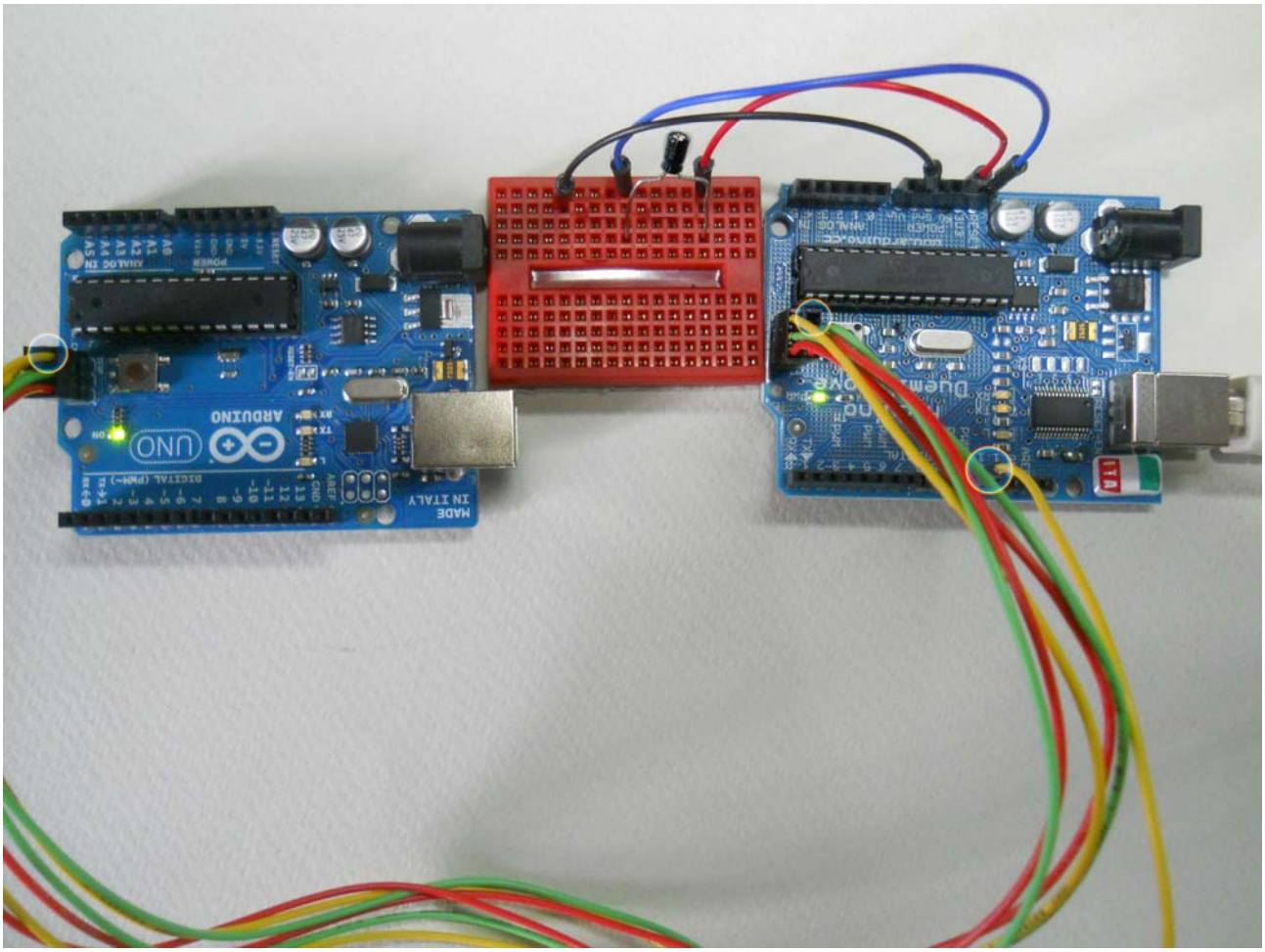
Ecco lo schema pratico dei collegamenti, i colori Vi aiuteranno nella fase di realizzazione, e la foto del cavetto realizzato:





In pratica, dal lato target colleghiamo tutti i pin di ISP, dal lato programmer lasciamo libero il pin Reset; il filo proveniente dal pin Reset del target lo terminiamo con un pin maschio, in modo da poterlo inserire nel pin digitale 10 di Arduino programmer, così:

Sulla sinistra c'è Arduino target, sulla destra Arduino ISP programmer, dove si vedono (cerchiati) il pin Reset di ISP vuoto ed il pin digitale 10 collegato al filo (giallo) proveniente dal Reset di Arduino target.



Poiché non dobbiamo mai trascurare la necessità di dover disabilitare l'autoreset, nello foto ho aggiunto una mini breadboard sulla quale dobbiamo portare (visti da sinistra a destra): GND (opzionale, in questo caso), pin Reset del programmer, +3,3V; al solito collegheremo un condensatore da 10µF con il polo positivo su 3,3V ed il polo negativo sul RESET.

La procedura software è quella già vista in precedenza, non cambia nulla, abbiamo solo semplificato i collegamenti. Tenete presente che entrambi i pin 3,3V e RESET del Programmer sono liberi, quindi in realtà il condensatore (come abbiamo visto nelle figure precedenti, potrebbe essere infilato direttamente negli header di Arduino, eliminando di fatto la piccola breadboard.

Se guardate bene i led tx-rx del programmer sono accesi, il trasferimento del bootloader è già in corso!

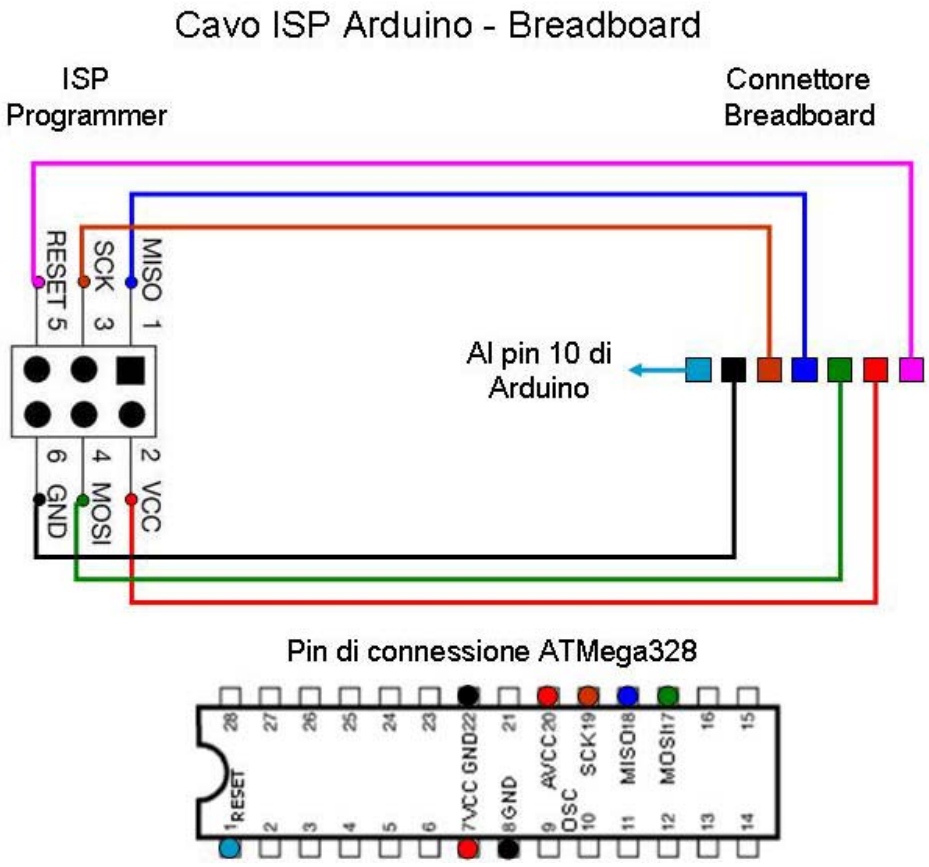
6B - Il cavo Arduino-ISP-Breadboard

E passiamo ora al cavo ISP che ci permetterà di collegare l'Arduino ad una breadboard; ciò ci permetterà di organizzare facilmente una breadboard per fare questa operazione ogni volta che ci serve; io ho usato una piastra grande per fare e mostrare i collegamenti in maniera chiara ma possiamo tranquillamente usarne una da 400 contatti, "sacrificando" pochi euro, e realizzare così un supporto stabile.

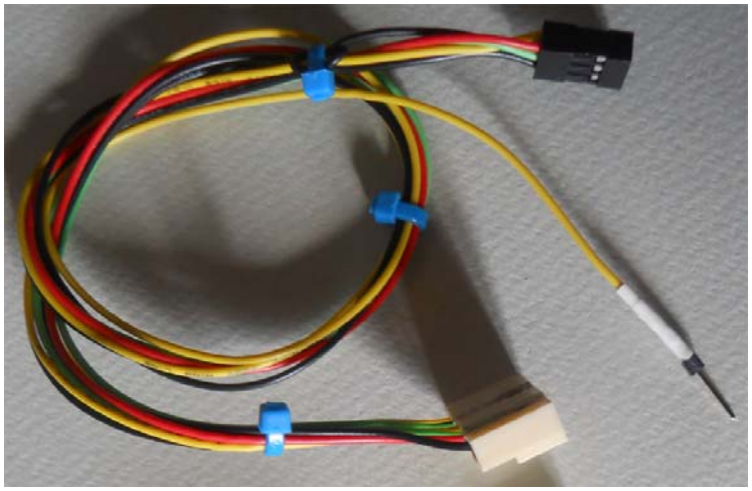
Questa volta, contrariamente a quanto visto nel paragrafo precedente, su Arduino programmer collegheremo tutti i 6 pin dell'ISP e il pin digitale 10 e li porteremo sulla breadboard; qui chiaramente, a motivo dell'organizzazione dei contatti della breadboard, non possiamo replicare lo spinotto ISP di Arduino, quindi dobbiamo realizzare una fila di 7 contatti per portare alla breadboard i 6 pin dell'ISP ed il pin digitale 10 di Arduino.

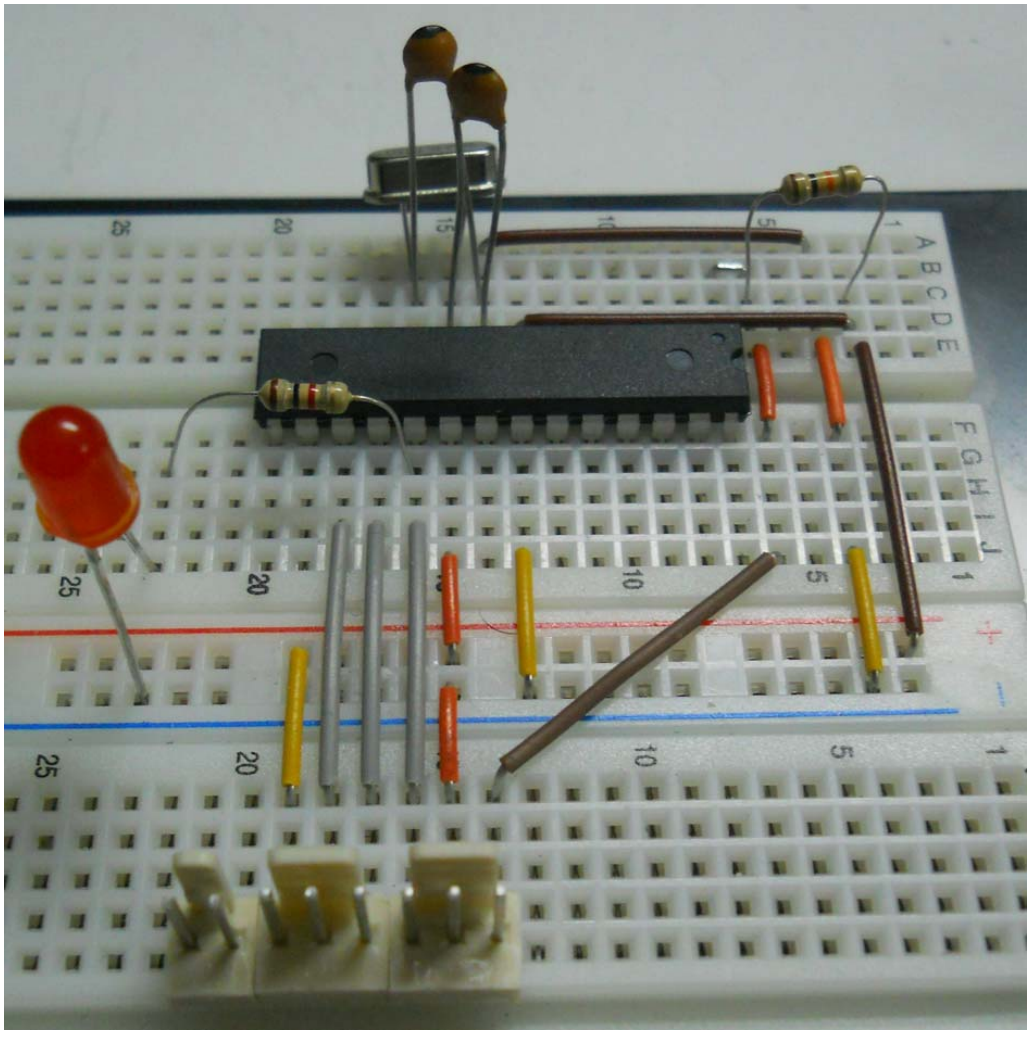
In tal modo predisporremo anche l'ormai noto circuito per la disabilitazione dell'autoreset, se ci dovesse servire avremo il segnale pronto.

Per la realizzazione ci servono: un connettore femmina 3x2, un connettore maschio 7x1 da cs, un connettore femmina 7x1, tutti a passo 2,54, 7 fili di lunghezza sufficiente (almeno 15 cm), possibilmente tutti di colore diverso, un pin maschio di recupero da un connettore strip. Ecco lo schema pratico dei collegamenti, i colori Vi aiuteranno nella fase di realizzazione:

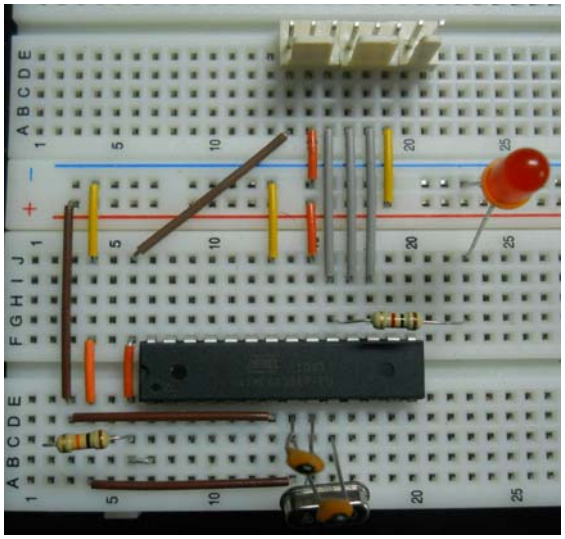
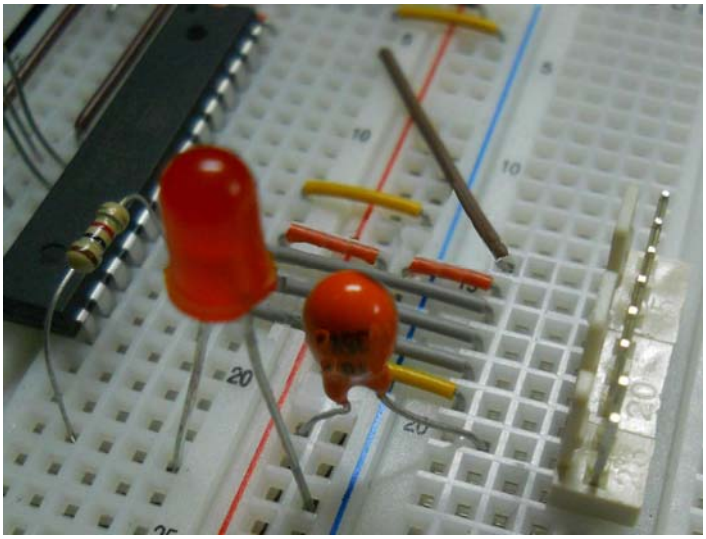


Questa volta ho riportato anche i piedini dell'ATmega328P, così avete anche lo schema di collegamento tra il connettore della breadboard ed il chip vergine; in ogni caso le foto vi aiuteranno a realizzare tutto rapidamente; nelle immagini seguenti il cavetto realizzato e le connessioni della breadboard[10]; le foto riportano le due versioni, con e senza circuito anti-autoreset (unica differenza è il condensatore da 10 µF):

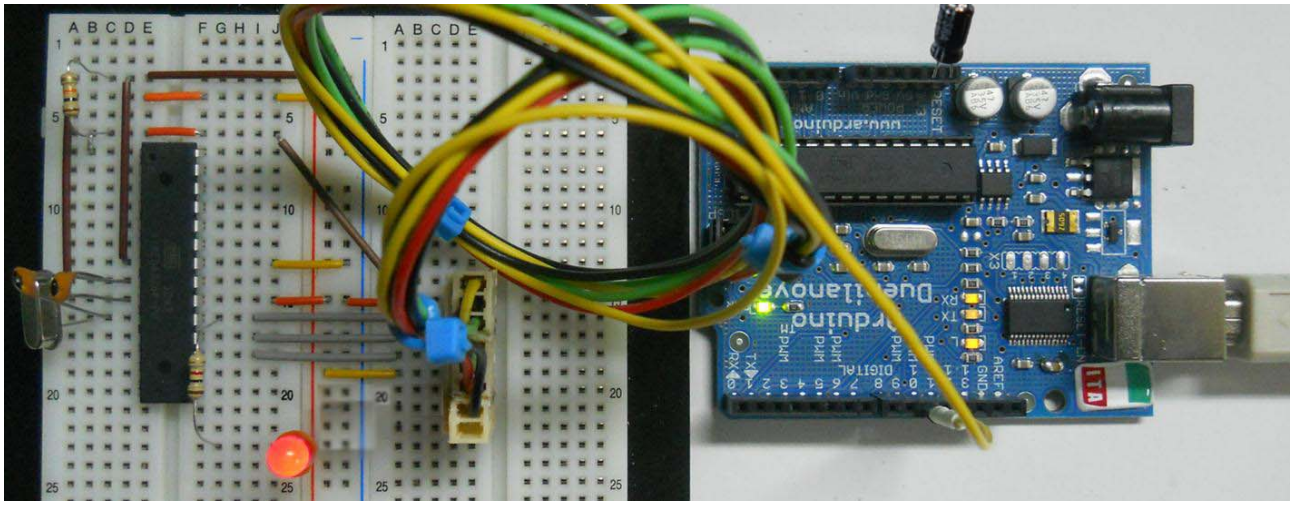




[10] Certamente avrete notato che i pin sono 8 e non 7, ma questo connettore è realizzato con 2 da 3 pin ed uno da 2 pin uniti tra loro, il primo pin a sinistra non è utilizzato.

Con le foto precedenti certamente realizzerete il vostro circuito in pochissimi minuti; ed ecco il nostro Arduino 2009 mentre carica il bootloader (notate il led al lavoro) in un chip montato sulla breadboard:



Notate che in un caso abbiamo usato per l'eventuale anti-autoreset un condensatore al tantalio da 10µF montato sulla breadboard, col polo – al pin che va al RESET dell'Arduino, ed il polo + sui 5V della breadboard. Nell'ultima foto (quella con Arduino) invece abbiamo fatto ricorso al solito condensatore elettrolitico, sempre da 10µF, collegato questa volta sui 3,3V. Se pensate di usare eventualmente questa seconda modalità, viene da sé che il cavetto ISP potrà essere realizzato solo con 6 fili, semplificando tutto. **Ribadisco che questo condensatore, come più volte spiegato in questa Guida, deve essere usato solo se la procedura non va a buon fine**; ricordo ancora che nelle mie prove con Arduino UNO come programmer non ho mai avuto problemi di autoreset, li ho avuti invece con 2009 e Compatibile, e in questi ultimi due casi (è proprio la situazione della foto) ho dovuto aggiungere il componente, risolvendo immediatamente il problema. Ricordo ancora che molti invece hanno il problema opposto, cioè non riescono ad usare

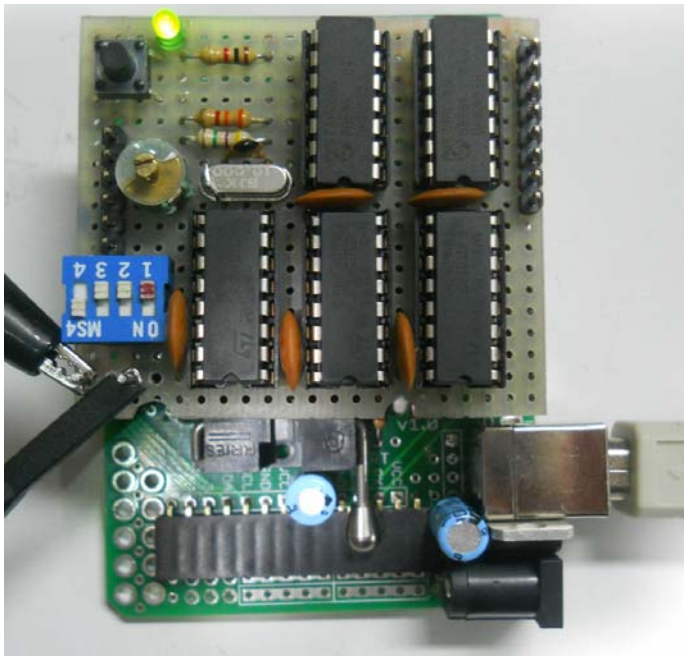
Arduino UNO come programmer, la soluzione è sempre la stessa: aggiungere il condensatore, collegato tra il pin Reset di Arduino e 5V o 3,3V..

Cap. 3 Procedure per caricare uno sketch su ATmega328P

L'unica tecnica nota agli utilizzatori di Arduino alle prime armi, per caricare uno sketch in un chip ATmega328P, da usare poi in modalità stand alone (che, ricordo, consiste nell'usare il chip senza la board Arduino, con l'ausilio di pochissimi componenti esterni, per controllare un circuito specifico), è certamente stata quella di montare tale chip al posto di quello originale di un Arduino ed effettuare la tipica operazione di upload tramite l'IDE.

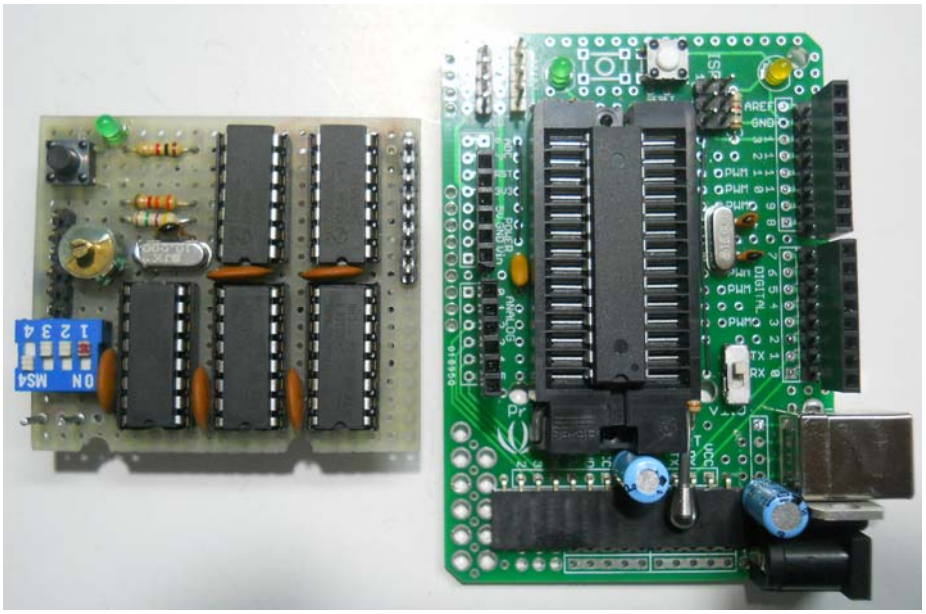
Questa tecnica, per quanto efficace, ha due tipi di problemi: il primo è che costringe l'utente a fare questa operazione ad ogni modifica dello sketch, quindi togliere il chip da circuito stand alone, montarlo su Arduino, programmare il nuovo sketch, togliere il chip da Arduino e rimontarlo sul circuito stand alone; la seconda, più grave, è che alla lunga irrimediabilmente Arduino si danneggia, al minimo si rovina lo zoccolo che contiene il chip, ma può succedere di peggio (come è accaduto a me!) a forza di maneggiare attrezzi per rimuovere ogni volta il chip dallo zoccolo. Inoltre il chip deve obbligatoriamente avere il bootloader, ma questo, arrivati a questo punto della Guida, non è più un problema!

Per tali ragioni molti si sono messi a lavorare alla possibilità di evitare quanto più possibile, di toccare Arduino e, come potrete leggere in seguito, alla fine si è trovato un metodo estremamente soddisfacente. Per la verità nel PlayGround del Forum di Arduino è presente un tutorial che spiega una tecnica seriale, di facile realizzazione, che riporterò in questo capitolo, ma solo perché, così come illustrata da loro, molti scrivevano di non riuscire ad applicarla (io stesso non sono mai riuscito a farla funzionare). In realtà mi è bastato aggiungere un collegamento per risolvere il problema, ma passiamo subito alla pratica.



Prima però, voglio fare un flash-end, come avviene in certi film dove ti fanno vedere come prima scena il finale e poi si ricomincia dall'inizio. Il mio è questo:

Questa foto illustra una pcb su cui è montato un chip ATmega328P in stand alone, nascosto da uno schedino realizzato su millefori, innestato su di essa, tipo shield; nella foto seguente ecco i componenti separati:



La particolarità è che questo circuito in questo momento sta lavorando autonomamente, alimentato con un piccolo alimentatore USB, e nel chip ATmega c'è il solo sketch che pilota lo schedino, SENZA bootloader!

Quindi, per togliere subito ogni dubbio riguardo a questa problematica sappiate che, delle due tecniche che vedremo, quella che ho definito "seriale" ci permette di inviare uno sketch ad un chip ATmega328P in stand alone, a condizione che esso contenga già un qualsiasi bootloader Arduino; la tecnica che ho definito "ISP" invece ci permette di fare questa stessa operazione anche su chip assolutamente vergini, quindi privi del bootloader; chiaramente questo ci dà il vantaggio di poter sfruttare, in caso di sketch molto complessi, anche lo spazio normalmente occupato dal bootloader che, come sappiamo è di 2kB[11] o di 0,5kB a seconda del tipo di bootloader.

Se fossi stato a conoscenza di questa tecnica quando ho realizzato la millefori, certamente avrei montato su di essa anche il chip ATmega328P con i suoi pochi componenti aggiuntivi, ma all'"epoca" (prima di scrivere la prima versione della Guida) ero convinto che fosse indispensabile usare Arduino per farlo funzionare.

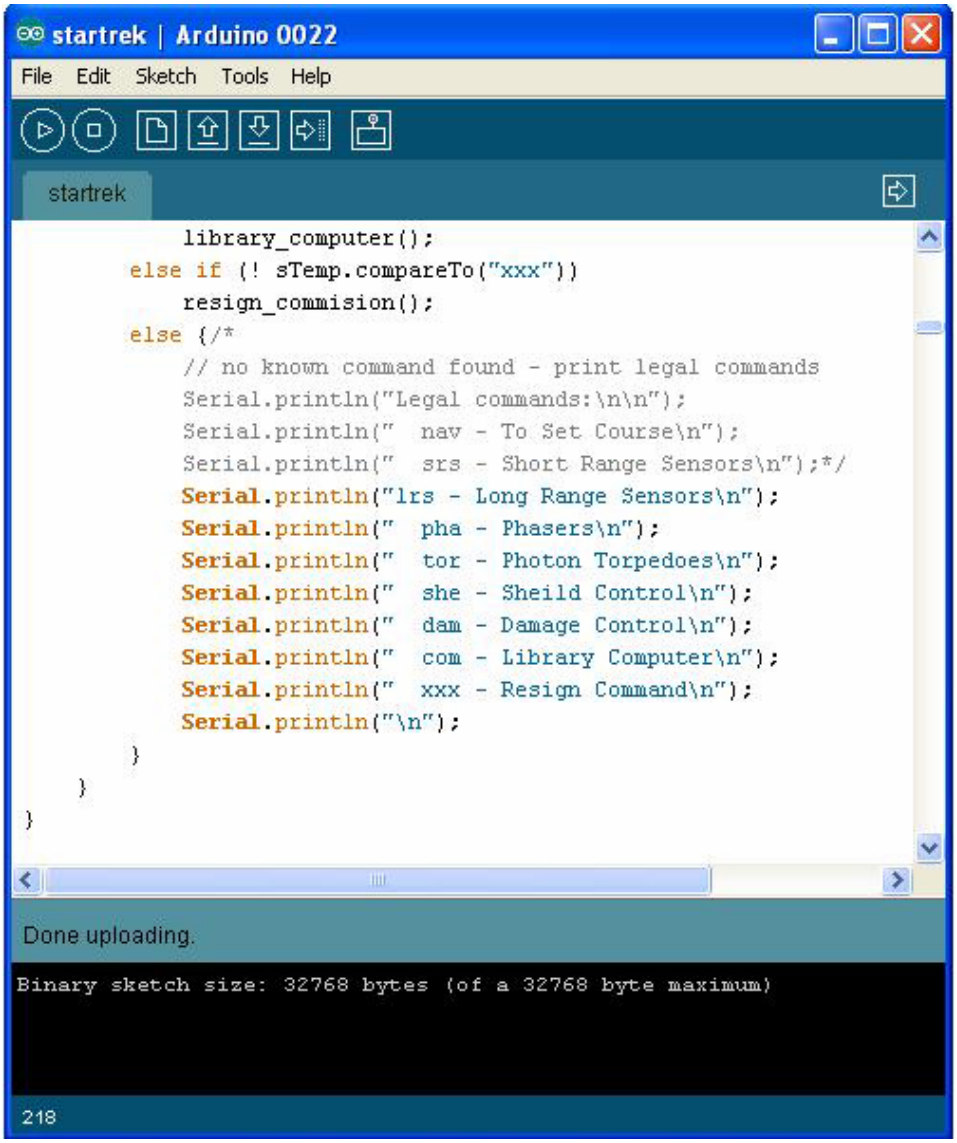
Per farvi capire l'enorme vantaggio che ho tratto da questa tecnica sappiate che questo circuito[12] è in grado di generare sequenze di 8, 12, 16 o 24 bit; le sequenze vengono impostate tramite uno sketch scritto sull'IDE di Arduino, e ben immaginate cosa dovevo fare prima, ogni volta che volevo modificare anche un solo bit.

[11] In realtà il bootloader del 2009 è di circa 1,5kB, ma 2kB è lo spazio che esso occupa nella memoria flash.
[12] Realizzato con l'aiuto del forum, un particolare ringraziamento va ad **Astrobeed** per i preziosi suggerimenti elettronici.

Solo per i più curiosi aggiungo che la board che vedete sulla destra l'ho realizzata su una delle tante proto-shield che si trovano in commercio (ho trovato questa, molto bella, perché prevede già le connessioni del pulsante reset, dei led power e pin 13, del connettore ISP, della presa USB, e altre cose utili di cui parlerò più avanti), con uno zoccolo ZIF (acronimo per indicare zero forza di inserzione del chip) per facilitare le operazioni di sostituzione del chip; normalmente la uso come secondo Arduino per caricare sui chip vergini il bootloader e/o lo sketch.

Prima di iniziare lo studio di questa tecnica vi mostro un'immagine "rara", il buon fine ("done uploading") del caricamento di uno sketch da 32768 byte, abbiamo riempito fino all'ultimo byte la memoria flash dell'ATmega328P!!! Lo sketch non è mio, me l'ha gentilmente prestato **leo72**, seguendo le sue indicazioni l'ho modificato in modo da avere esattamente la dimensione che mi serviva:

Notate la perfetta corrispondenza tra i byte dello sketch ed il valore massimo!



p. 7 – Le tecniche per caricare uno sketch su un chip in stand alone

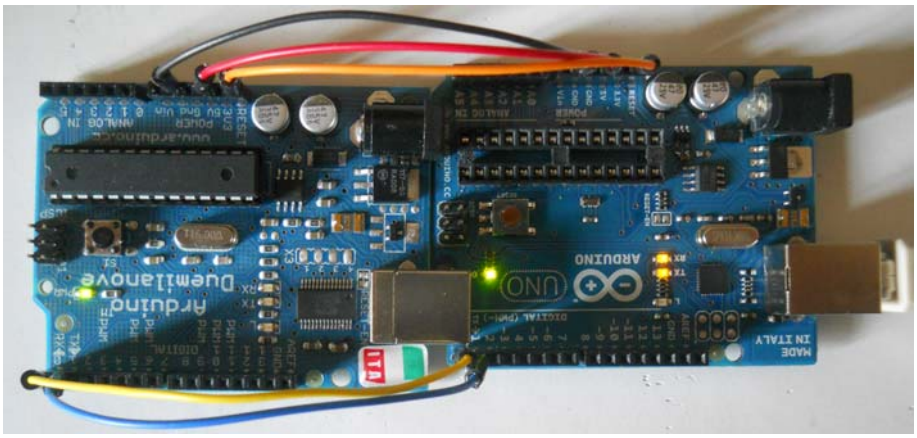
7A – La tecnica seriale

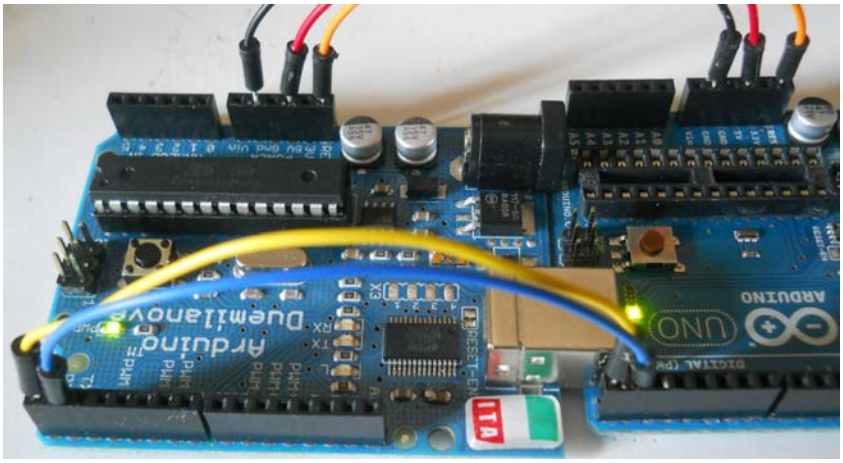
La prima tecnica che vedremo è quella seriale, ricavata dal tutorial ufficiale, ma con l'aggiunta di un collegamento indispensabile affinché tutto funzioni.[13]

Essa consiste fondamentalmente nel **togliere il chip originale dall'Arduino** e collegare la board (che assume le funzioni di programmatore) ad un altro Arduino o ad una breadboard, con funzione "target", cioè su cui è installato il chip da programmare. I collegamenti da fare sono di tipo pin-to-pin, cioè ogni segnale di Arduino va collegato al corrispondente segnale del circuito target. I segnali da collegare sono: +5V, GND, tx, rx e reset.

Nelle foto seguenti riporto i collegamenti tra l'Arduino senza chip ed i tre possibili circuiti target (le foto sono state scattate sempre durante il regolare caricamento dello sketch, come potete vedere dai led tx/rx accesi):

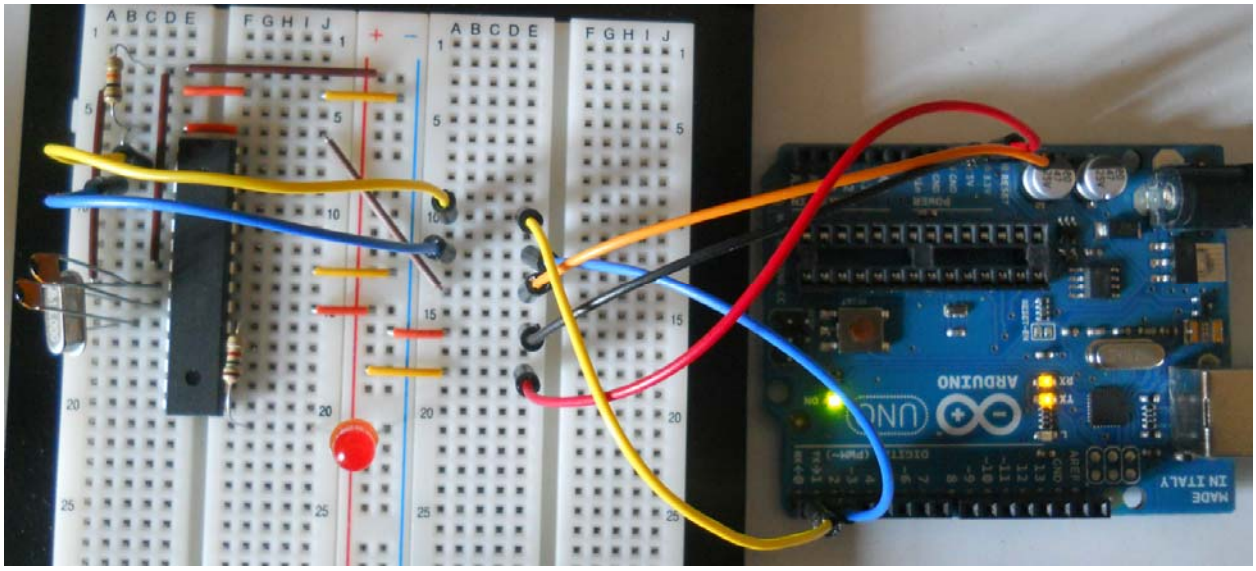
Due Arduino (non importa il modello o la combinazione, è indifferente) da diverse angolature:

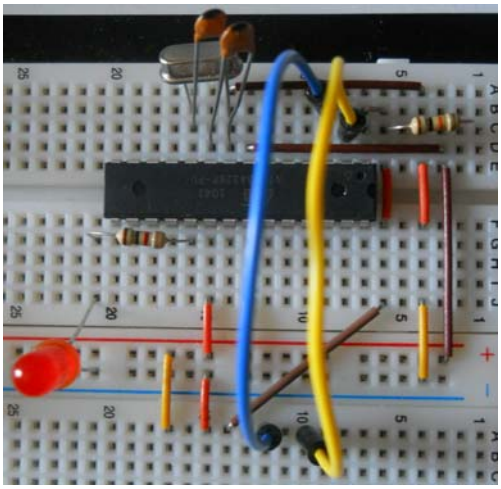


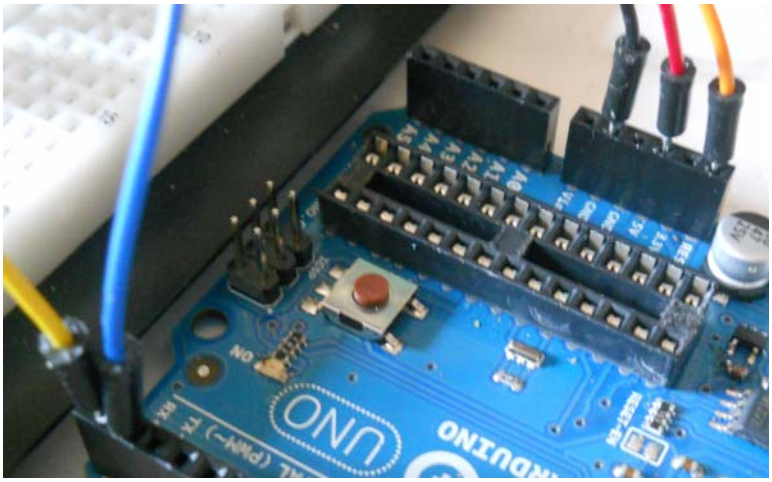


[13] Non so se il tutorial ufficiale nel frattempo sia stato corretto, e non so nemmeno se la mancanza del collegamento del RESET sia un errore, forse non era indispensabile nelle versioni di Arduino esistenti quando esso è stato scritto.

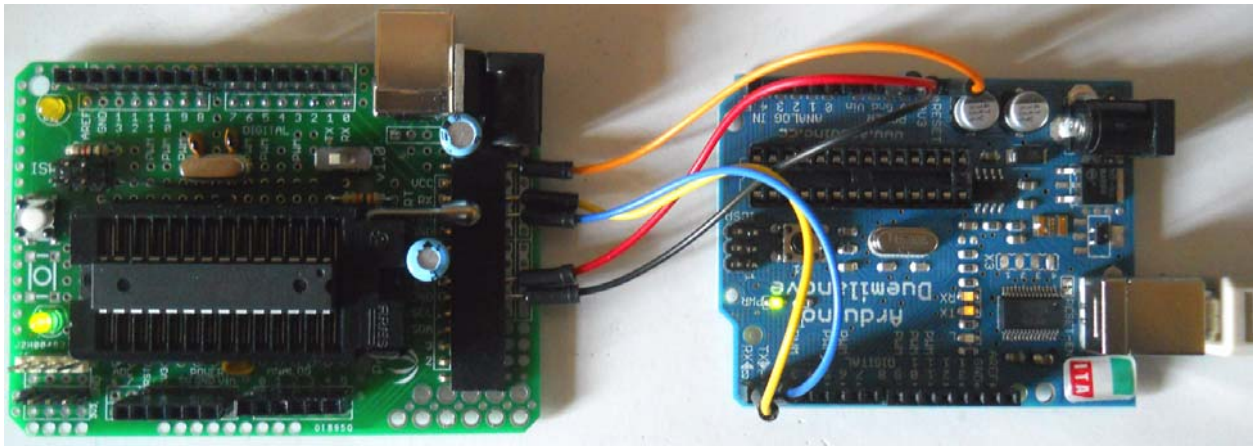
Arduino e breadboard:







Arduino e PCB (in questa da me realizzata avevo già previsto un connettore laterale con tutti i segnali del collegamento seriale)



Come potete vedere, con soli 5 fili si realizza rapidamente il collegamento, **NON dimentichiamo di togliere il chip originale dall'Arduino usato come interfaccia,**

altrimenti l'IDE tenterà di caricare lo sketch su entrambi i chip e inevitabilmente ci darà errore!

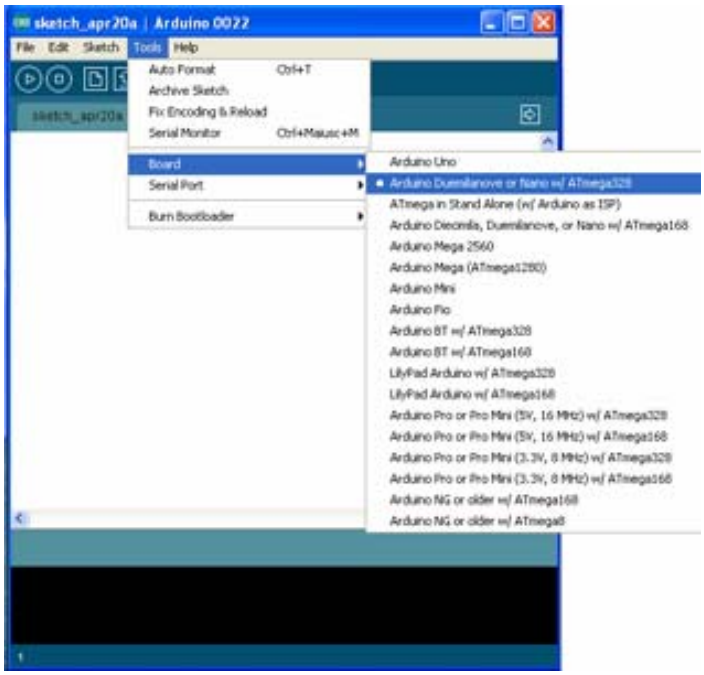
Ricordo inoltre che il chip su cui vogliamo caricare uno sketch DEVE contenere obbligatoriamente il bootloader (la versione non ha importanza, come vedremo).

Se analizziamo bene la tecnica ci accorgiamo che questo semplice collegamento in realtà non è altro che una specie di "prolunga" di Arduino, che ci permette di aggiornare comodamente lo sketch sul nostro chip da lavoro, lasciandolo nel suo circuito stand alone.

Una volta effettuate le connessioni[14] possiamo collegare il cavo USB proveniente dal PC e passare alla fase software, quella del caricamento dello sketch.

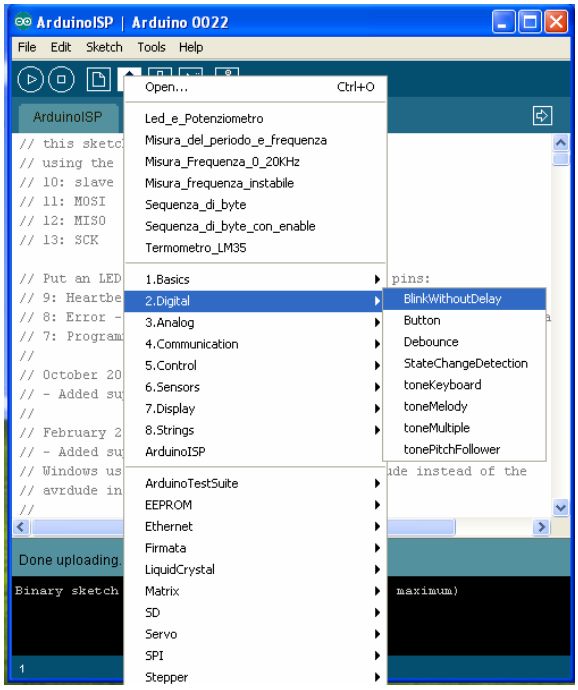
L'operazione è estremamente semplice:

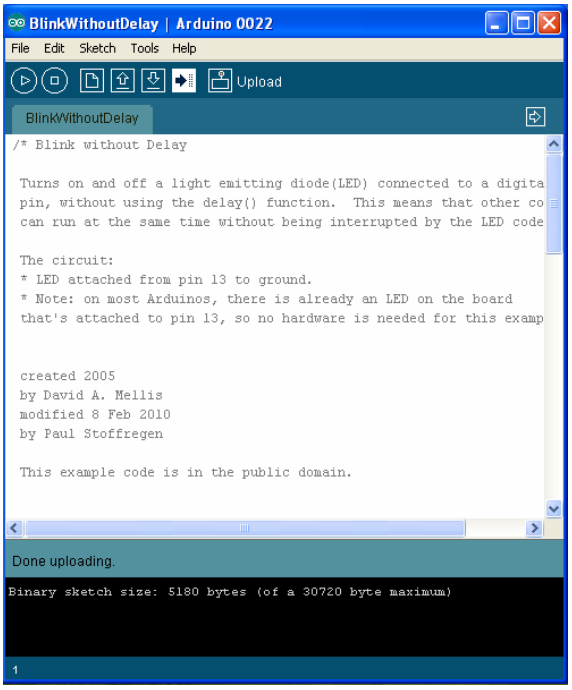
1 – Aprire l'IDE e selezionare la COM di connessione della board Arduino; **NOTA IMPORTANTE**: il modello di board da selezionare NON è quella fisicamente collegata al PC, bensì il tipo corrispondente al bootloader caricato nel chip da programmare. P.es. se sul chip avete precedentemente caricato il bootloader di Arduino UNO, ora come board dovete scegliere proprio questo modello anche se, p.es., state usando una 2009 collegata al PC; in pratica è la stessa situazione di quando montate su un'Arduino un chip con bootloader di un altro modello, se volete che funzioni dovete impostare l'IDE sul modello di bootloader del chip e NON sul tipo di board Arduino che lo ospita. Nelle immagini seguenti supporremo di usare una board Arduino UNO per programmare un chip con bootloader della 2009:



[14] ATTENZIONE! Come sempre è buona norma realizzare i collegamenti SENZA alimentazione, in caso contrario si corre il rischio di fare qualche danno irreversibile!

2 – Caricare lo sketch ed inviarlo con upload al chip stand alone:





Naturalmente questa semplice procedura è identica per tutte le modalità di circuito target, cioè sia esso un'altra Arduino o una PCB stand alone o una breadboard preparata appositamente.

7A1 – Una strana scoperta

Avevo già segnalato nella versione precedente della Guida questa curiosità, che però ha una sua buona valenza tecnica e quindi è degna di nota. Mentre stavo per chiudere la versione precedente della Guida, mi imbatto in un Topic del Forum, nel quale un newbie[15] (almeno nel numero di post), spinto dall'odio che prova per l'estrazione dei chip dagli zoccoli, cominciò a fare una serie di prove, sulla carta assurde, e invece scoprì che si può usare la tecnica "seriale" per caricare uno sketch su ATmega in breadboard, SENZA TOGLIERE IL CHIP DA ARDUINO!

Un minimo di teoria, ma spiegata in maniera semplificata per chi non ha conoscenze specifiche in questo campo: per grandi linee la programmazione seriale consiste nell'inviare uno sketch dal PC al pin "rx" del chip, mediante l'interfaccia di Arduino (tramite il chip FT232RL o ATmegaXXu2, secondo il tipo di board); questa trasmissione avviene a "pacchetti", cioè a gruppi di byte, e presuppone una verifica dei dati inviati, che avviene tramite il pin "tx" del chip; in poche parole, per ogni pacchetto dati ricevuto il chip invia un

[15] Il suo nick è **superkulak**, lo ringrazio pubblicamente per avermi concesso di pubblicare questa curiosità tecnica.

segnale di “ok” all’interfaccia tramite il pin “tx”; se per una qualsiasi ragione il pacchetto ricevuto non è corretto, il PC lo invia nuovamente.

La tecnica “seriale” richiede di togliere il chip da Arduino in quanto, diversamente, i due chip vengono collegati con i 3 pin “reset”, “tx” e “rx” in parallelo; la trasmissione dati avviene normalmente verso entrambi, ma poi ognuno invia il proprio “ok” all’interfaccia e questo causa un errore nell’IDE, impedendo che l’operazione si concluda; il motivo è dettato principalmente dal fatto che i due chip non sono perfettamente sincronizzati.

La tecnica “scoperta” dal nostro amico è semplicissima: eliminare il collegamento “tx” tra Arduino e breadboard. In tal modo entrambi i chip ricevono i dati tramite il pin “rx”, ma solo quello di Arduino risponde all’IDE e l’operazione va a buon fine, io stesso ho fatto moltissime prove e garantisco il funzionamento di questa semplice tecnica.

Chi è interessato alla storia può leggere il topic.

Ciò che rende “non scientifica” la tecnica è il fatto che, in linea teorica, non si può mai sapere se il chip su breadboard è stato programmato o meno finché non lo si prova, proprio perché in caso di errori non ci sono segnali di ritorno tramite “tx”. In effetti, facendo alcuni esperimenti su un chip ATmega328P, mi è capitato di danneggiarlo ed ho approfittato per sperimentarlo subito. Come pensavo la trasmissione “ufficialmente” è andata a buon fine (“done uploading”) ma il chip continuava a non funzionare; provando a programmarlo direttamente montato su Arduino invece è uscito l’errore.

In conclusione, resto dell’avviso che la tecnica “ISP” è comoda, semplice, tecnicamente perfetta, e quindi consiglio vivamente di metterla in pratica, oltretutto consente di caricare sia il bootloader che gli sketch su un chip, come abbiamo visto.

Però per fare delle prove “al volo”, su un circuito che, magari con un led, dia subito segnalazione del buon funzionamento del chip, è certamente una tecnica comoda e semplice da realizzare, specialmente per chi non vuole costruirsi il cavetto ISP.

7B – La tecnica ISP

La seconda tecnica, definita ISP, è decisamente più comoda e potente, i vantaggi li ho già descritti nella parte introduttiva di questo paragrafo, uno su tutti il fatto di NON dover togliere il chip dall'Arduino usato come programmatore.

NOTA IMPORTANTE: Questo metodo permette di programmare il nostro chip stand alone inviandogli un bootloader (come abbiamo già visto) oppure con uno sketch; ma quando si manda lo sketch l'eventuale bootloader viene cancellato. Se si vuole creare un chip che contenga entrambi il caricamento dello sketch deve avvenire col metodo seriale oppure mettendo il chip direttamente su Arduino.

Chiarito questo punto fondamentale iniziamo dai collegamenti hardware: al solito useremo un'Arduino come programmatore ISP, mentre il circuito stand alone potrà essere costituito da un altro Arduino, una PCB o una breadboard. Il collegamento sarà effettuato con la tecnica del cavo ISP, vista nei capitoli specifici, visto che in realtà la tecnica usata per inviare un bootloader è molto simile e richiede gli stessi collegamenti.

NOTA IMPORTANTE: Leggete attentamente questa nota, è fondamentale ai fini del buon funzionamento dei micro in cui caricherete solo uno sketch con la tecnica ISP. Tutti i microcontrollori della ATMEL escono di fabbrica settati per lavorare con clock interno a 1MHz. Come vedremo più avanti, se decidiamo di far lavorare il micro a questa velocità possiamo usare la tecnica ISP direttamente per caricare lo sketch. Se invece, come vedremo tra poco, vogliamo emulare un Arduino creando uno stand-alone a 16MHz, bisogna modificare i valori di alcuni byte che si chiamano "fuses". Questa operazione è fattibile solo creando una board virtuale con i valori dei fuses che servono e caricare PRIMA il bootloader e POI lo sketch. Infatti l'operazione di caricamento del bootloader è l'unica che permette di variare i fuses del micro; invece il caricamento dello sketch non è in grado di compiere questa operazione. Allora, se si va a caricare uno sketch con una board settata a 16MHz, in un micro settato per lavorare a 1MHz o 8MHz, i tempi del firmware risulteranno completamente sballati; naturalmente vale anche la condizione opposta.

7B1 – La modifica del file boards.txt

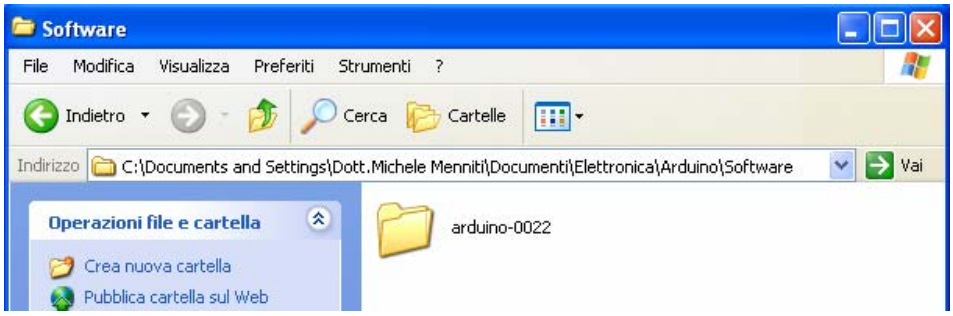
Affinché questa tecnica funzioni prima di ogni altra operazione dobbiamo creare un "nuovo Arduino", naturalmente virtuale! Questa operazione ci serve per far capire all'IDE che deve inviare lo sketch al target e non al programmatore, visto che stiamo lavorando con due chip contemporaneamente! L'operazione, piuttosto semplice anche per un neofita, consiste nell'inserire una serie di righe nel file **boards.txt** che è quello contenente

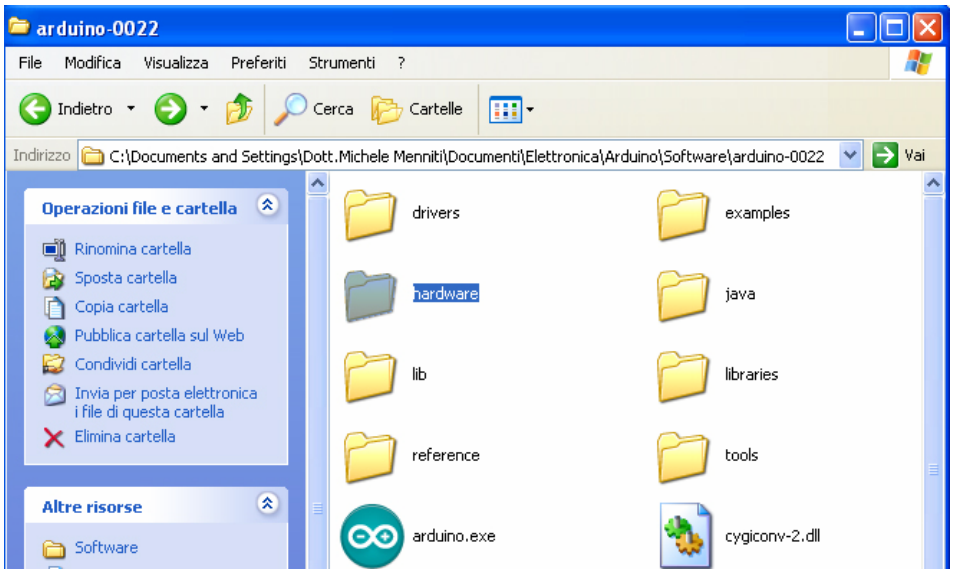
tutti i tipi di board che vengono poi mostrati dall'IDE. Bisogna "scovare" tale file, aprirlo con un qualsiasi editor di testo, inserire le righe che tra poco Vi mostrerò e salvarlo senza cambiargli il nome.

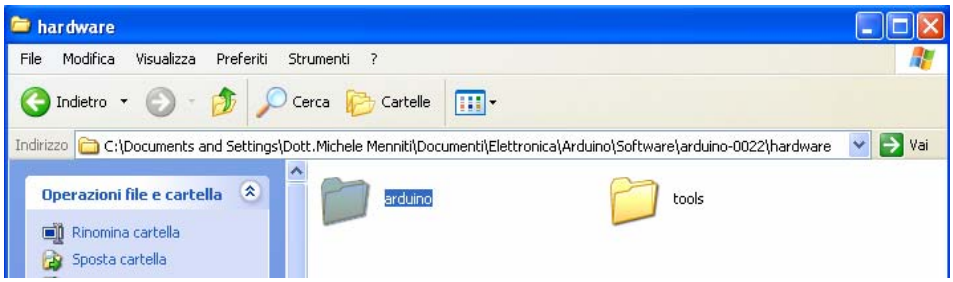
L'unica informazione non standard è quella della posizione del software IDE, ma ognuno di noi ben saprà dove l'ha messo prima di iniziare ad usarlo! A partire da tale posizione il percorso completo per arrivare al file boards.txt è:

X:\MyPath\arduino-0022\hardware\arduino

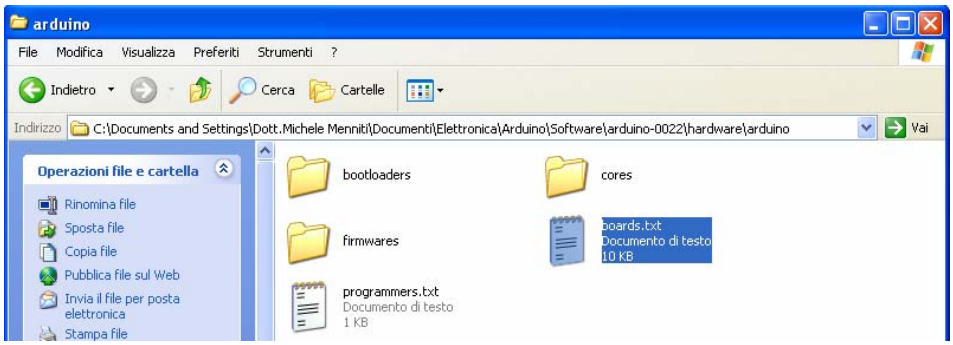
In cui **X** è la lettera del disco su cui è installato l'IDE, **MyPath** il percorso necessario per arrivare alla cartella che contiene l'IDE (nell'es. la **0022**). Trovata questa cartella (arduino-0022) occorre aprirla con doppio clic, quindi trovare ed aprire la cartella "hardware", infine trovare ed aprire la cartella "arduino". Solo per i più inesperti una sequenza di immagini sul percorso da seguire:







Ed ecco finalmente il file board.txt!



E queste sono le nuove righe da inserire (atmega3216……) nel file:

atmega3216.name=ATmega in Stand Alone 16MHz (w/ Arduino as ISP)
atmega3216.upload.protocol=stk500
atmega3216.upload.maximum_size=32768
atmega3216.upload.speed=115200
atmega3216.upload.using=arduino:arduinoisp
atmega3216.bootloader.low_fuses=0xff
atmega3216.bootloader.high_fuses=0xdf
atmega3216.bootloader.extended_fuses=0x07
atmega3216.bootloader.path=optiboot
atmega3216.bootloader.file=optiboot_atmega328.hex
atmega3216.bootloader.unlock_bits=0x3F
atmega3216.bootloader.lock_bits=0x0F
atmega3216.build.mcu=atmega328p
atmega3216.build.f_cpu=16000000L
atmega3216.build.core=arduino
##

La posizione è indifferente, portate il cursore sotto una qualsiasi linea "#########" e scrivete TUTTE le righe che iniziano con atmega3216. Dopo l'ultima riga aggiungete una nuova linea di "#########", quindi esattamente come quella sotto cui avete iniziato a scrivere.

Confrontando queste righe con le prime presenti nel file, potrete notare che esse sono quasi identiche a quelle della board Arduino UNO, con alcune variazioni: la terza riga (upload.maximum_size) che aumenta lo spazio massimo a **32768** byte, appunto a 32KB, cioè l'intera memoria flash dell'ATmega328P; una nuova riga, in quinta posizione (upload.using=**arduino:arduinoisp**) che "spiega" all'IDE che deve usare Arduino come

programmatore ISP; in settima posizione il valore dell'high_fuses ora è "**DF**", mentre nella riga successiva il valore dell'extended_fuse ora è "**07**". Importantissima è la sequenza di caratteri iniziali (in questo caso "**atmega3216**"), che rappresenta un "codice" univoco per identificare la scheda; ho usato questa ma potete scrivere quello che volete, l'importante è che sia diverso da tutte le altre board presenti nel file.

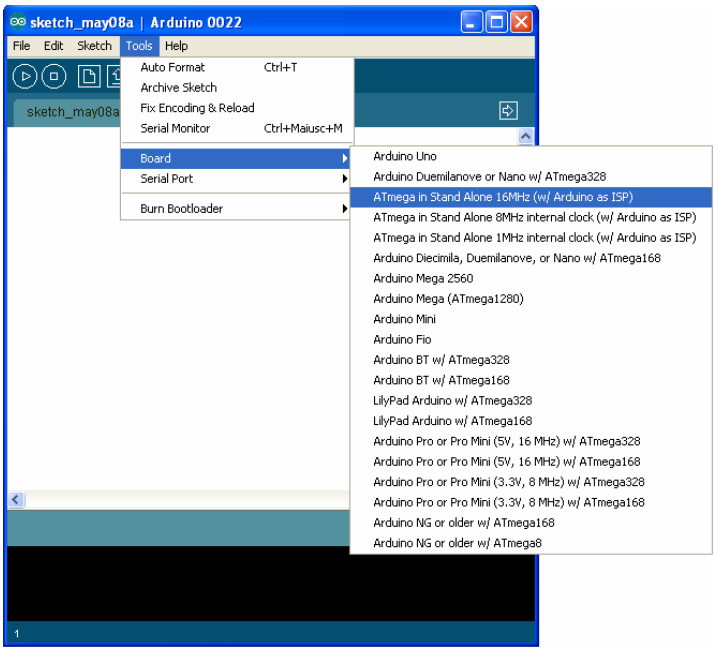
Rifacendomi alla **NOTA IMPORTANTE** descritta poco fa, vi chiarisco che questa board è fatta per programmare un micro che dovrà lavorare a 16MHz; se esso non è già settato (p.es. se è nuovo) per lavorare a questa frequenza, la prima operazione da fare è caricarvi il bootloader, seguendo le operazioni viste nello specifico capitolo. SOLO che questa volta dovrete selezionare questa board, nel passaggio finale. In questo caso si caricherà il bootoloader della UNO, ma comunque ciò non è importante in quanto, come detto, il caricamento dello sketch cancellerà il bootloader.

La nuova nuova board io l'ho inserita in terza posizione, cioè dopo le board UNO e 2009, ma non ha alcuna importanza, la posizione rappresenta solo il punto in cui la board "Stand Alone 16MHz" apparirà nell'elenco dell'IDE. A questo punto chiudiamo il file salvandolo, e siamo pronti! Infatti se ora apriamo l'IDE (se era già aperto lo dobbiamo chiudere e riaprire) ed andiamo a visualizzare l'elenco delle board disponibili, ecco la "nostra" in terza posizione:[16]

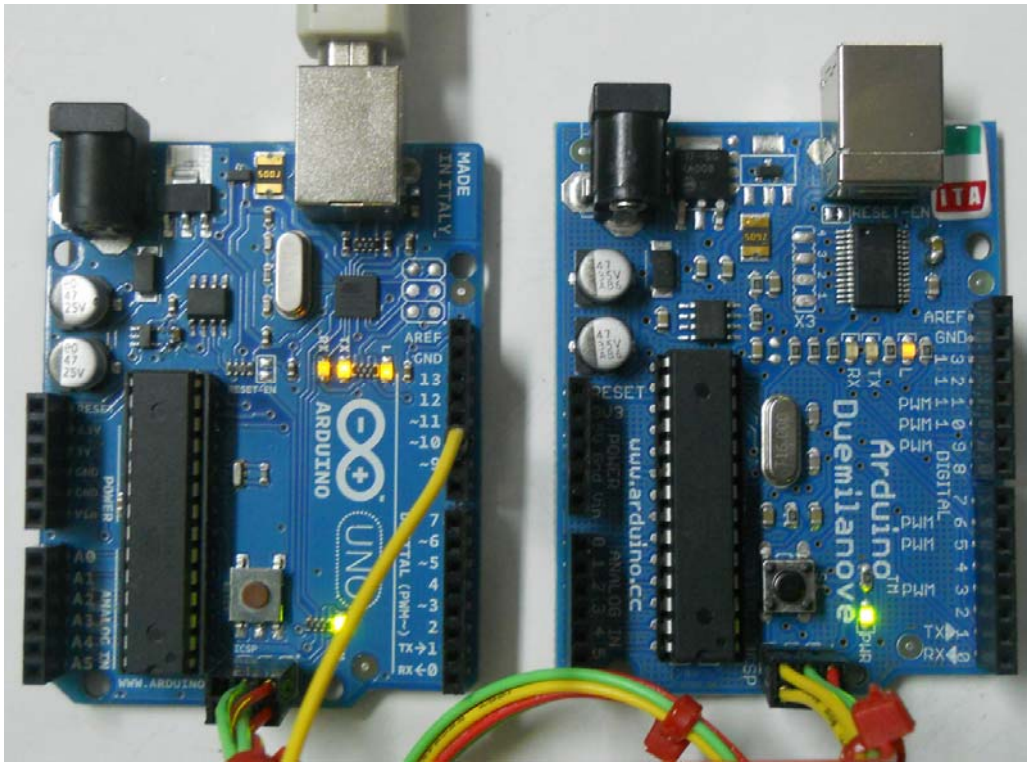
7B2 – Caricare uno sketch con Arduino-ISP-Arduino

Iniziamo dal cavo ISP "classico", per collegare due Arduino; questo il collegamento:

Arduino UNO (a sinistra) è il programmatore – Arduino 2009 (a destra) è il target

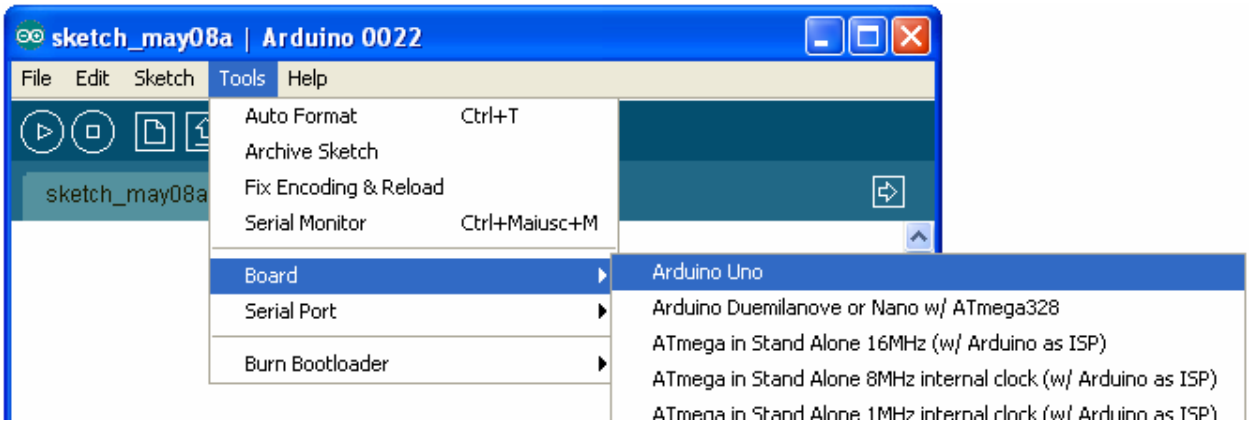


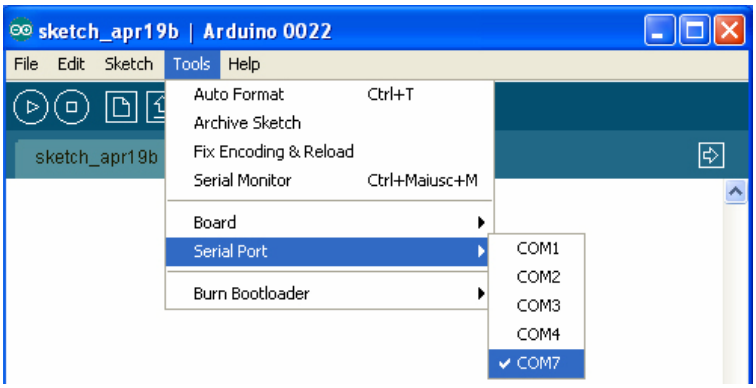
[16] Nell'immagine state vedendo già le altre due board virtuali che ci servono per il prossimo capitolo, ma ne parliamo dopo.



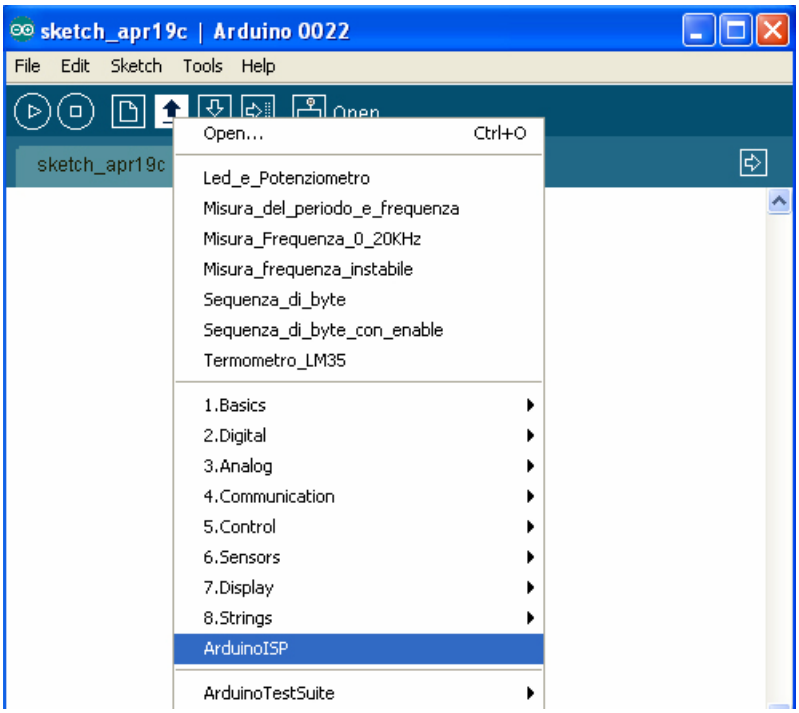
Non spendo troppe parole, la realizzazione del cavetto ISP è già stata spiegata qui, con questo cavetto il collegamento tra le due board è semplice e immediato; i pochi minuti di pazienza che servono per realizzarlo saranno immediatamente ripagati al primo utilizzo; Comunque se qualcuno non volesse ricorrere a questo metodo potrà sempre collegare le due board con 6 semplici fili pin-pin (maschio-maschio), seguendo le immagini e la descrizione dei collegamenti presentati qui. Possiamo quindi passare subito alla fase software:

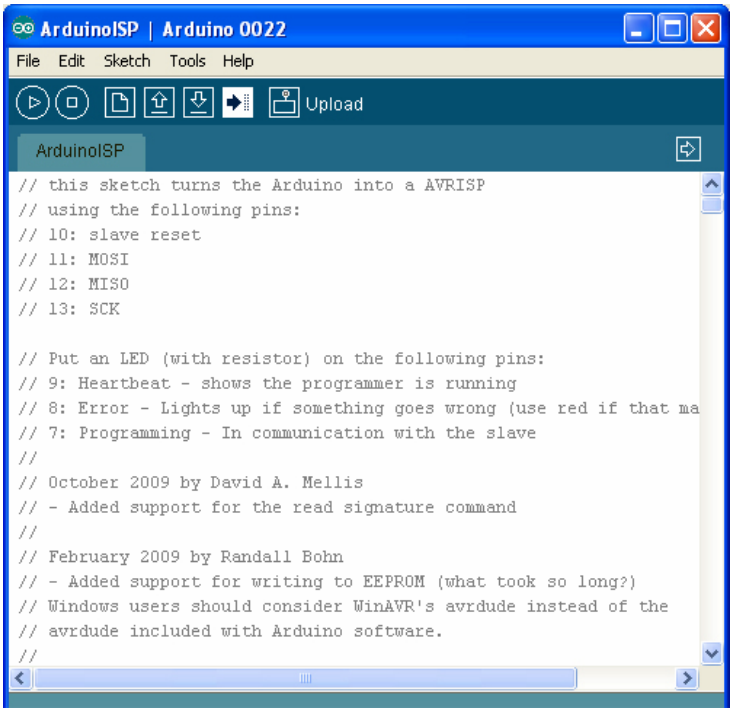
1 – Aprire l'IDE, selezionare la board che abbiamo collegato al PC tramite la porta USB (nella nostra foto è Arduino UNO) e la relativa COM.



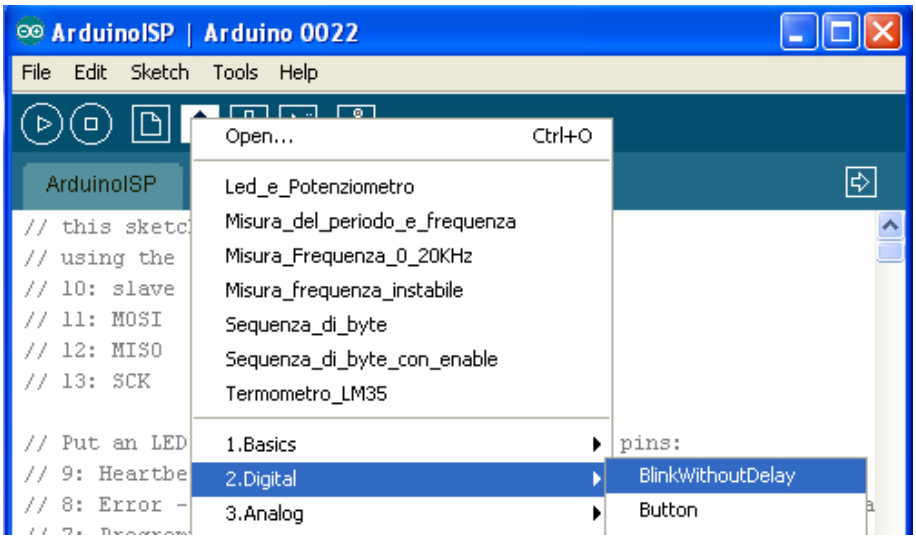


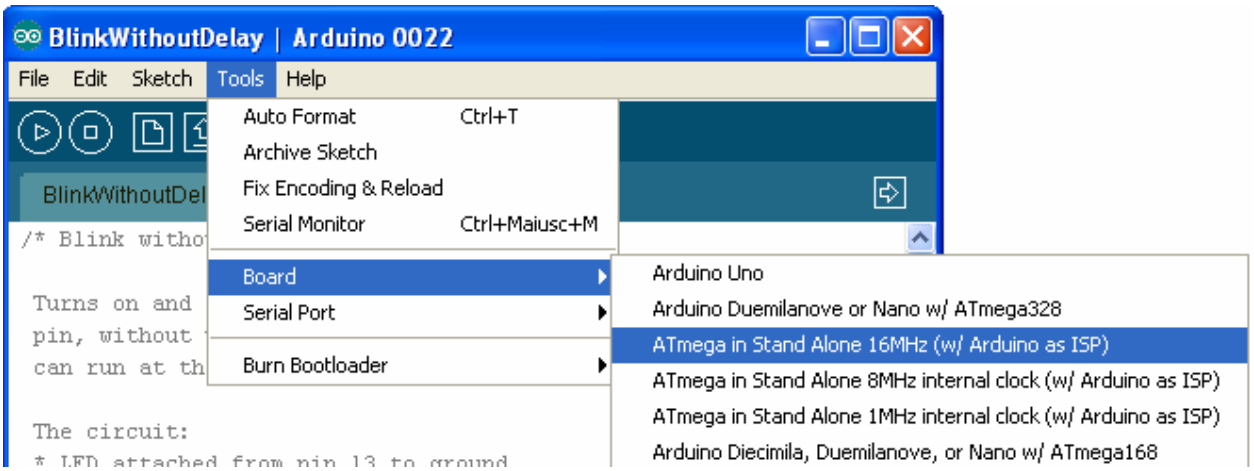
2 – Caricare lo sketch ArduinoISP e inviarlo con Upload ad Arduino UNO:

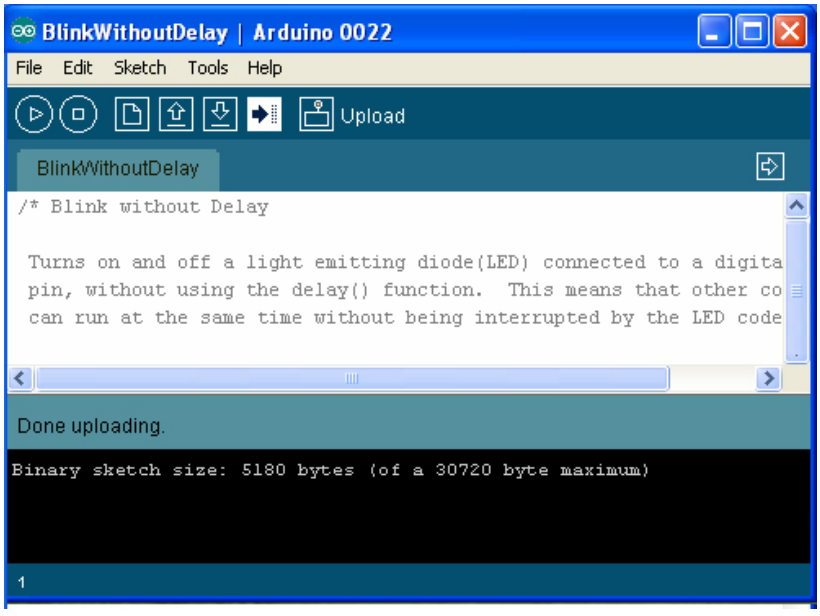




3 – Aprire ora lo sketch da inviare al chip montato sulla board target, quindi selezionare la board “**ATmega in Stand Alone (w/ Arduino as ISP)**” ed eseguire Upload:



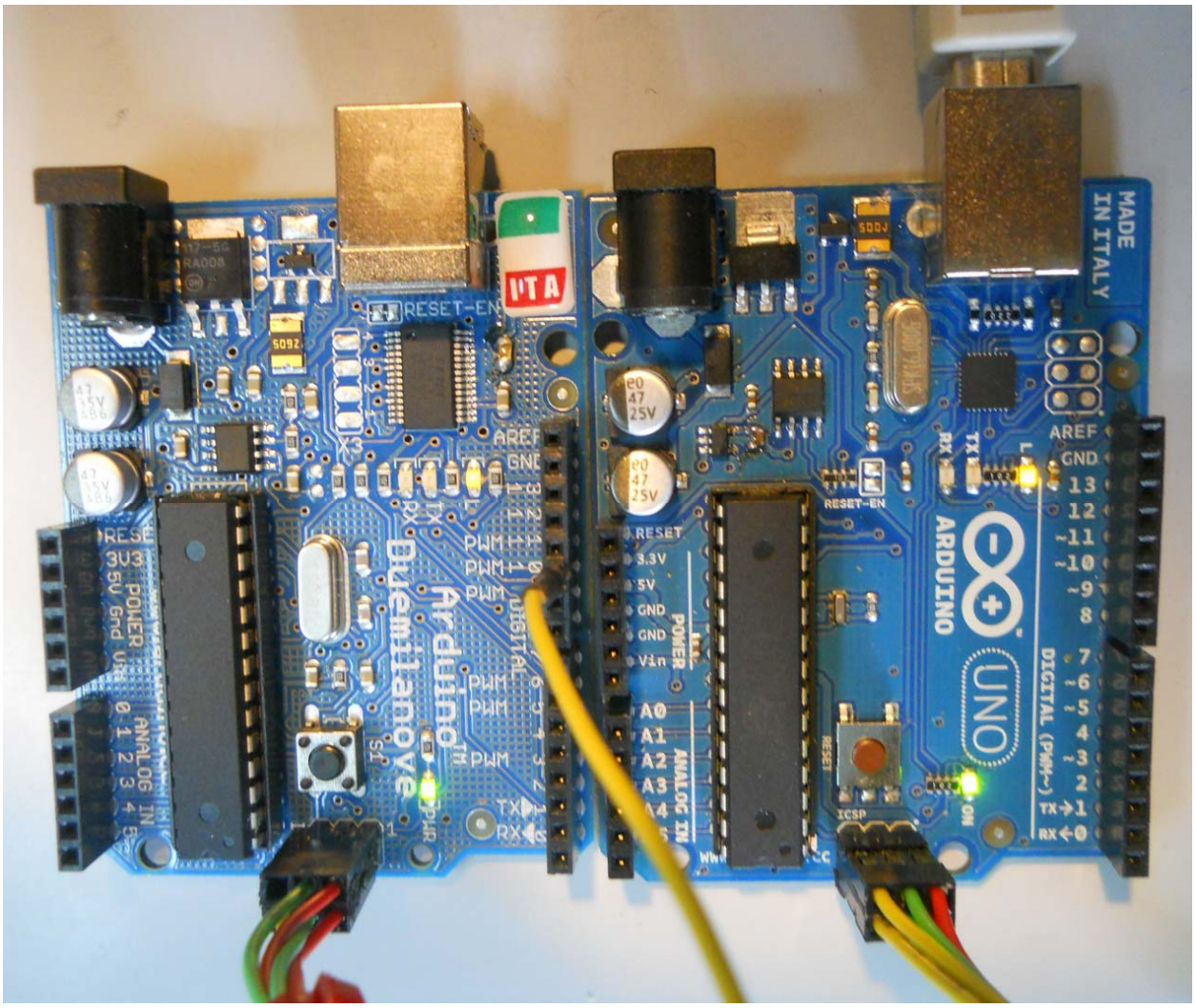




A questo punto l’operazione è terminata, il chip è pronto per essere montato nel suo circuito Stand Alone.

Ma questo paragrafo non può terminare così, non dobbiamo infatti mai dimenticarci dell'autoreset che, quando si verifica, ci impedisce di portare a buon fine le tecniche illustrate in questa Guida; come più volte spiegato il problema si può verificare indifferentemente con una qualsiasi delle board Arduino e la relativa soluzione va applicata solo se il problema si verifica effettivamente. Allora ora invertiamo le parti e usiamo Arduino 2009 come programmatore ISP e Arduino UNO come target, ecco la foto:

A sinistra Arduino 2009 ISP programmer, a destra Arduino UNO target



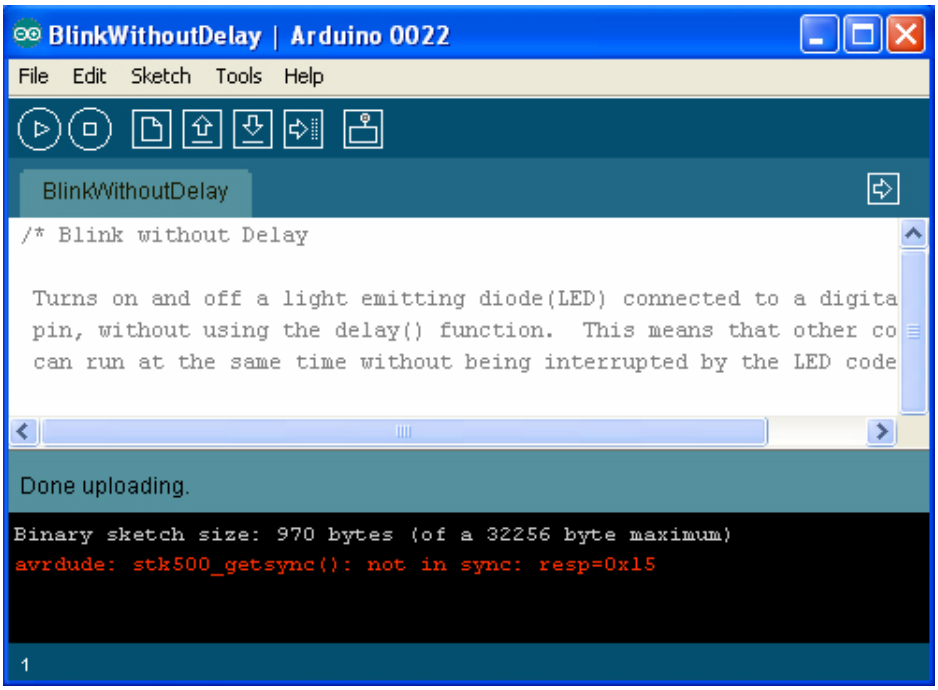
Rieseguiamo tutta la parte software, naturalmente con le necessarie variazioni, in breve:

1 – Apro l'IDE e seleziono la board Arduino 2009 e la relativa COM

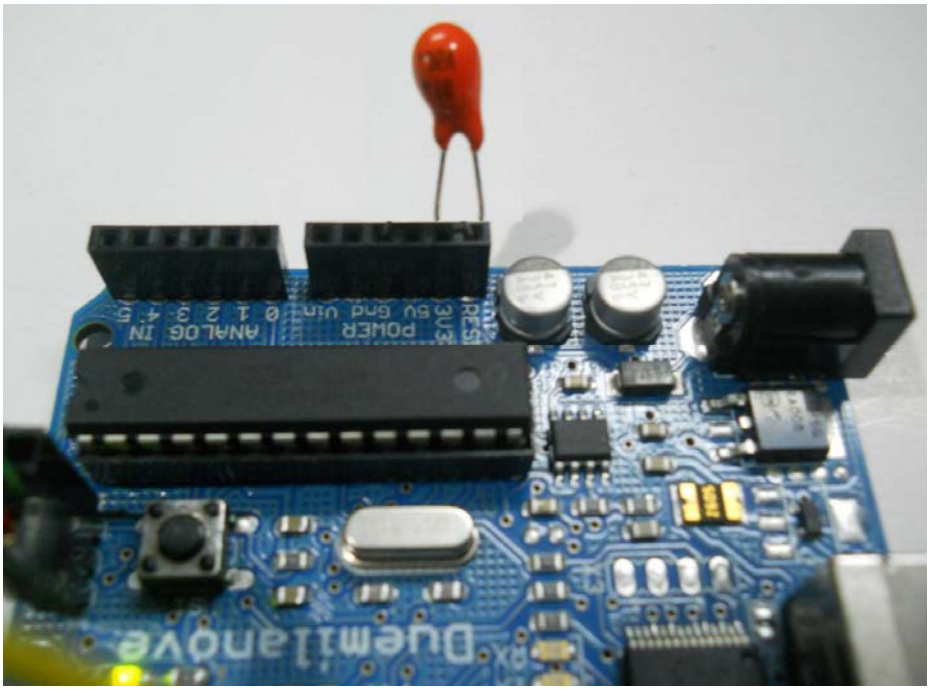
2 – Carico lo sketch ArduinoISP e lo invio alla 2009

3 – Apro lo sketch da inviare al chip montato sulla board target, quindi seleziono la board "**ATmega in Stand Alone (w/ Arduino as ISP)**" ed eseguo Upload.

A questo punto si verifica il problema, lo sketch non viene caricato ed esce questo errore:



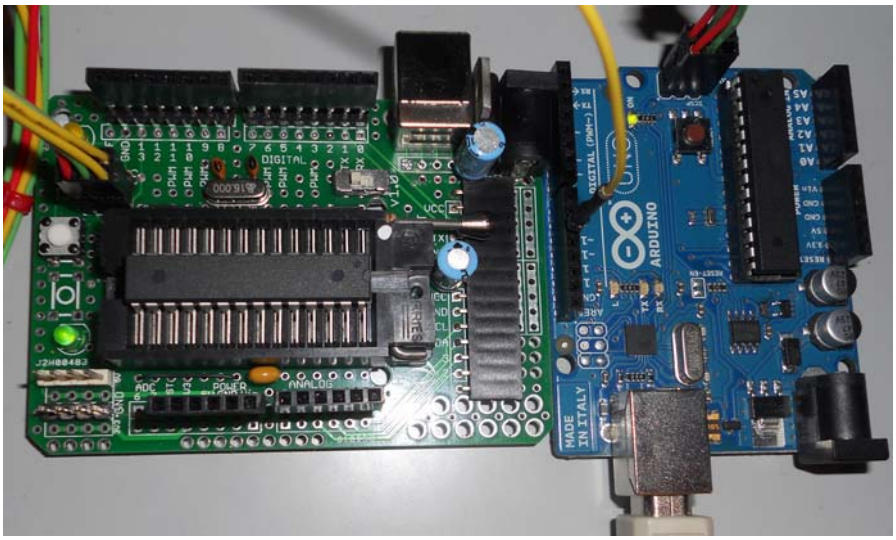
Per risolvere questo problema dobbiamo ricorrere al “solito“ anti-autoreset, come descritto qui; nello specifico ho risolto infilando un condensatore al tantalio da 10µF, direttamente nei pin di Arduino 2009, con il polo positivo su 3,3V ed il polo negativo su RESET:

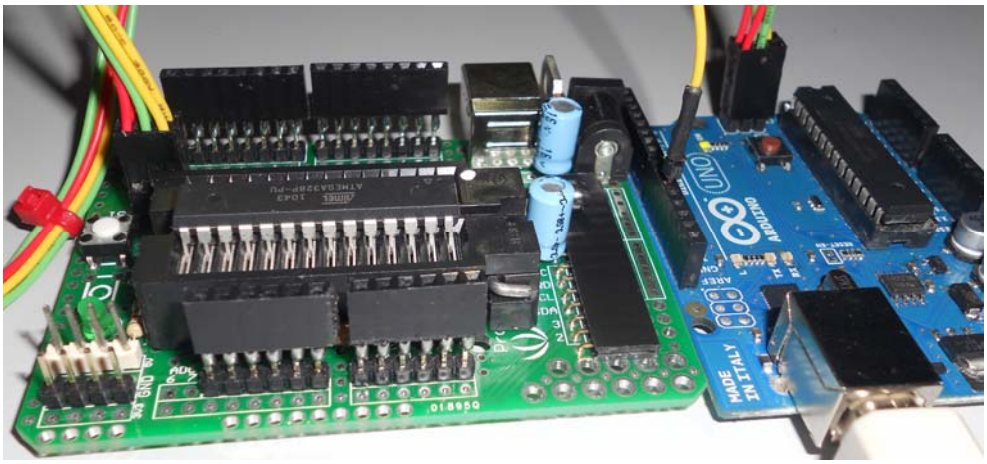


L’innesto del condensatore dovrebbe essere fatto, per buona norma, con Arduino scollegato dall’USB in modo che non sia alimentato, ma non dovrebbe creare problemi nemmeno con l’alimentazione, io stesso l’ho fatto più volte.

7B3 – Caricare uno sketch con Arduino-ISP-PCB o Arduino-ISP-Breadboard

A questo punto in realtà ho già spiegato tutta la tecnica di programmazione ISP di uno sketch, quindi ciò che resta da fare è solo mostrare alcune immagini sui collegamenti tra Arduino ISP programmer ed una generica PCB o una breadboard. Iniziamo dalla PCB:

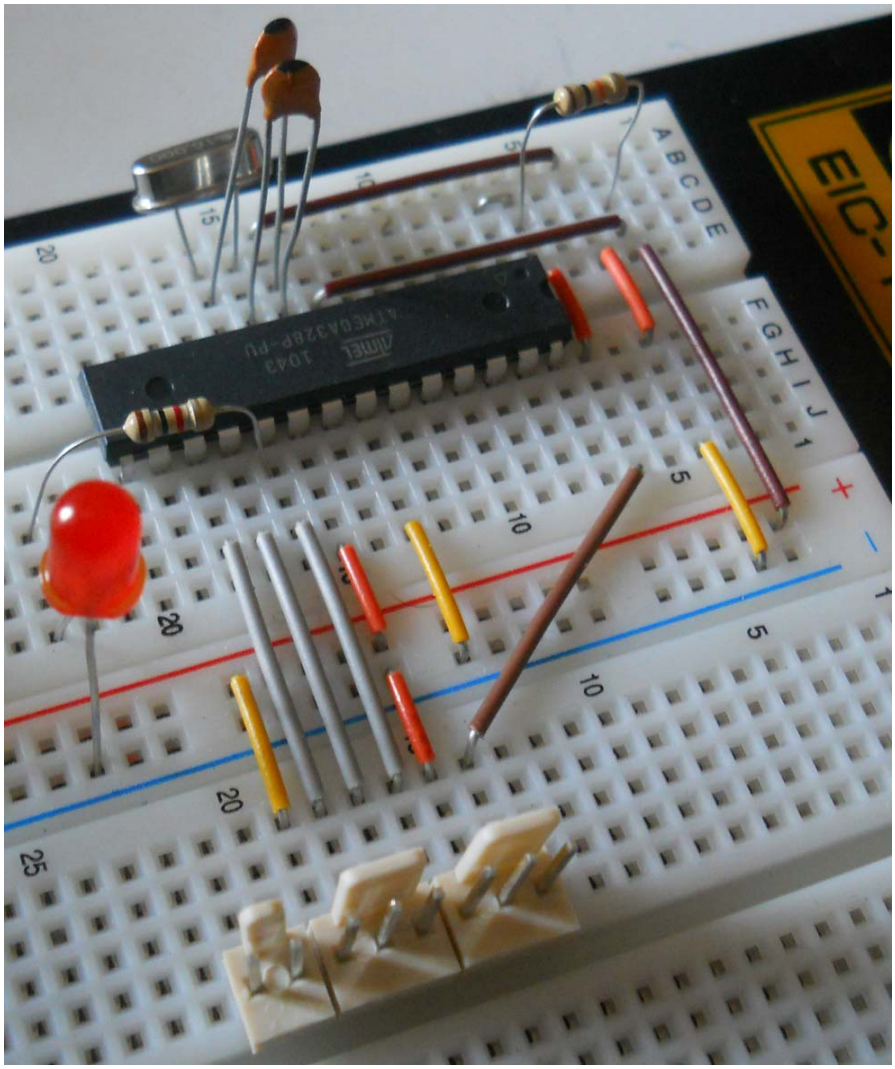


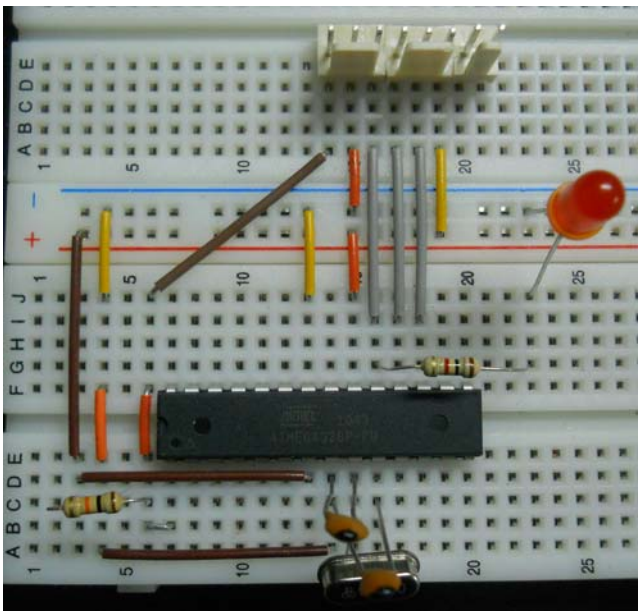


Le foto che state vedendo illustrano la PCB di cui Vi ho parlato nell'introduzione al paragrafo 6, qui; ripeto, l'ho realizzata prevedendo tutte le comodità possibili, proprio per usarla sia al posto di una breadboard per le operazioni di programmazione che come Stand Alone per collegarci shield commerciali o da me realizzate (ne ho mostrata una sempre nell'introduzione); ecco perché ho previsto sia il connettore ISP che le connessioni seriali, così come le alimentazioni da USB o da alimentatore esterno, per le applicazioni hardware che richiedono più di 500mA di corrente. Lo zoccolo ZIF costa tanto, ma si compra una volta sola e permette di inserire ed estrarre il chip con una facilità incredibile, senza ricorrere ad arnesi vari e soprattutto evitando di danneggiare col tempo lo zoccolo classico. La replica dei connettori laterali di Arduino mi permette appunto di salvaguardare anche quelli delle board originali e, soprattutto, mi evita di bruciare una bella scheda Arduino, nel caso lo shield dovesse avere qualche serio errore progettuale. A questo

punto vediamo anche le immagini del collegamento tra Arduino ISP programmer ed una breadboard, ma ormai le abbiamo viste tante volte ì:

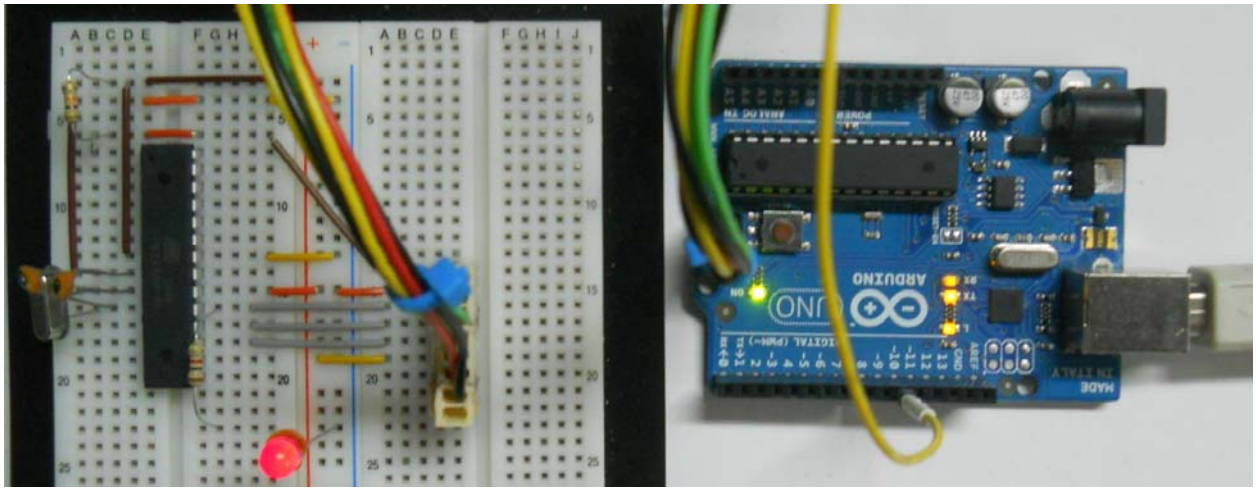
Una bella immagine del montaggio della breadboard

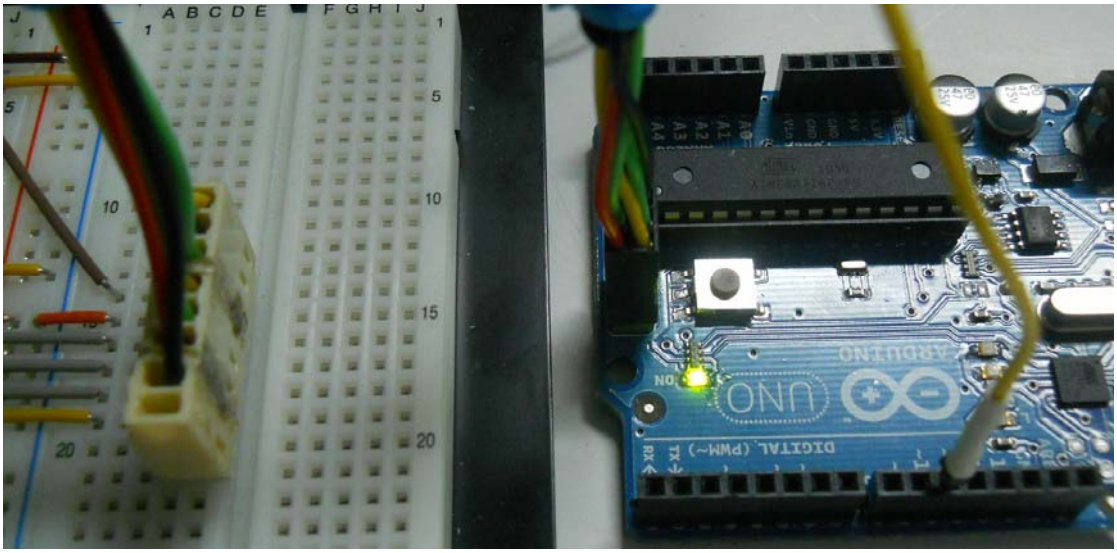




ed una immagine vista dall'alto

Il collegamento tra Arduino UNO e breadboard mediante il cavetto ISP, al lavoro!





Una volta effettuato il tipo di collegamento desiderato non resta altro da fare che ripetere la sequenza software già descritta più volte in questo stesso paragrafo e tutto andrà a buon fine.

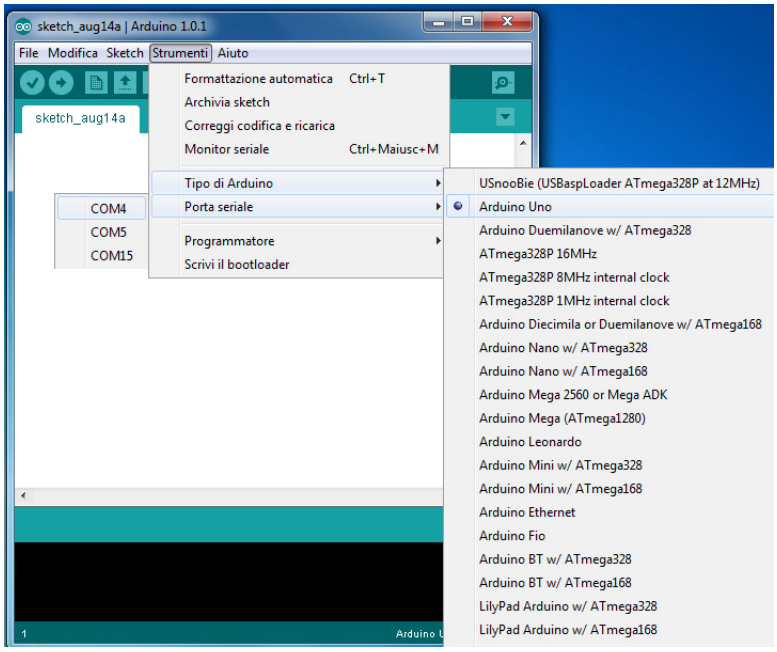
7B4 – Caricare uno sketch con IDE 1.0.1 mediante tecnica ISP

Questo paragrafo è un aggiornamento a quanto visto finora, in quanto vedremo come effettuare il caricamento di uno sketch tramite tecnica ISP, disponendo della nuova versione IDE 1.0.1. Vediamo anche in questo caso i passaggi in sequenza (continueremo ad "operare" in modalità stand-alone a 16MHz):

a – Creare la board viruale per l'IDE 1.0.1. Le modalità sono le stesse viste in precedenza. Le righe da aggiungere sono queste:

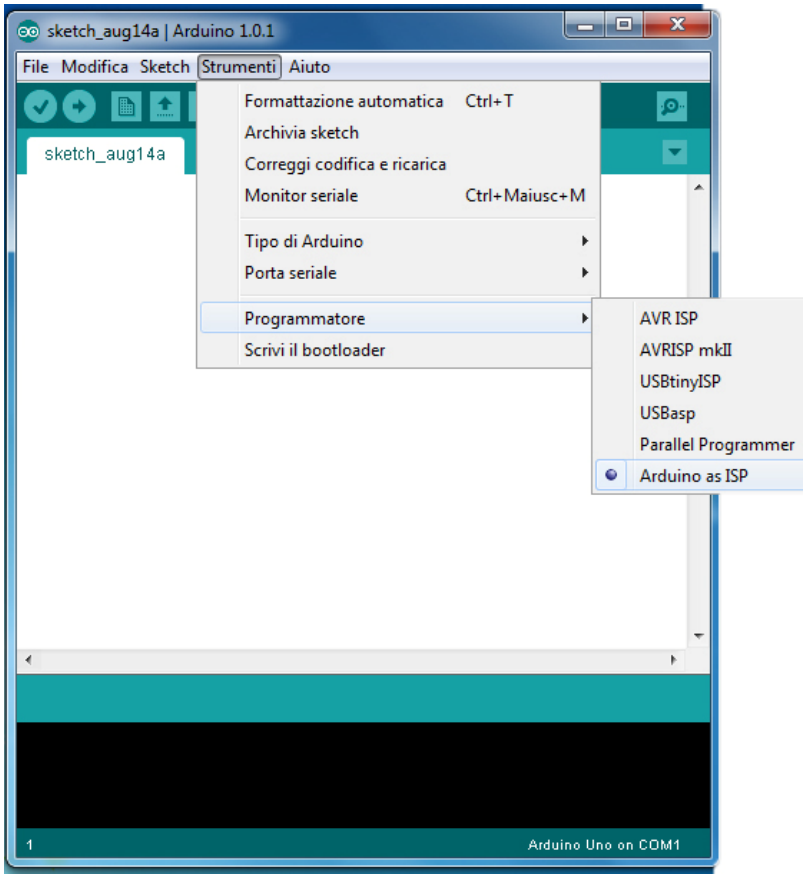
```
##############################################################
atmega3216.name=ATmega328P 16MHz
atmega3216.upload.protocol=arduino
atmega3216.upload.maximum_size=32768
atmega3216.upload.speed=115200
atmega3216.bootloader.low_fuses=0xff
atmega3216.bootloader.high_fuses=0xdf
atmega3216.bootloader.extended_fuses=0x07
atmega3216.bootloader.path=optiboot
atmega3216.bootloader.file=optiboot_atmega328.hex
atmega3216.bootloader.unlock_bits=0x3F
atmega3216.bootloader.lock_bits=0x0F
atmega3216.build.mcu=atmega328p
atmega3216.build.f_cpu=16000000L
atmega3216.build.core=arduino
atmega3216.build.variant=standard
##############################################################
```

Se le confrontate con quelle preparate per l'IDE 0022 noterete svariate differenze; fondamentalmente il motivo risiede nel fatto che questa nuova versione ora contempla il caricamento diretto di uno sketch tramite tecnica ISP; infatti non è più necessario aggiungere la riga che indica l'uso di Arduino come Programmatore ISP, ma le altre variazioni sono identiche: high_fuses="**DF**", extended_ fuses="**07**", il codice "**atmega3216**".

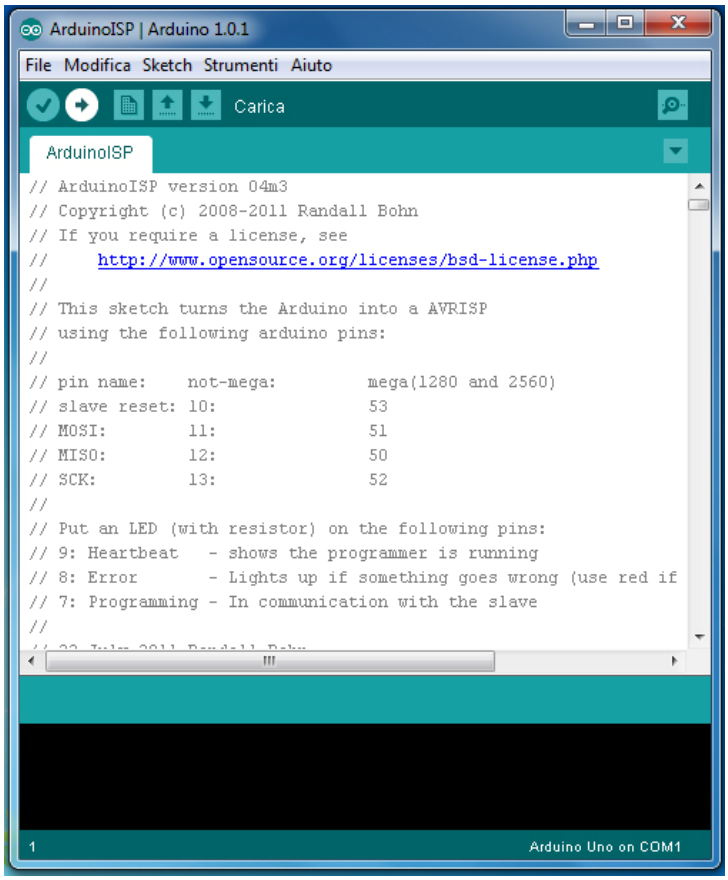


b – Settare la board usata come Programmatore (es. Arduino UNO) e la relativa COM.

c – Settare Strumenti→Programmatore→"Arduino as ISP".



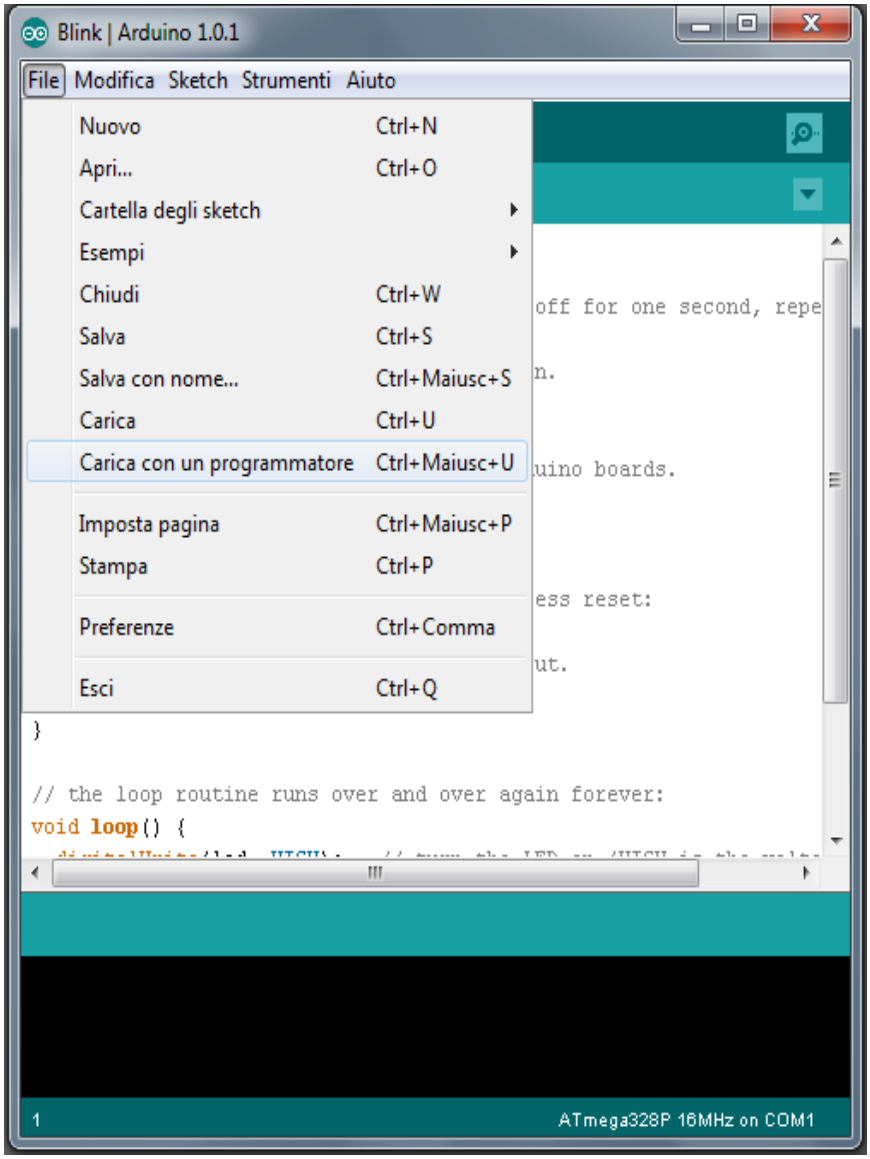
d – Caricare lo sketch ArduinoISP presente negli esempi ed inviarlo all'Arduino UNO



e – Caricare lo sketch da inviare al micro da programmare

f – Settare ora la board virtuale creata per caricare lo sketch nel micro da programmare

e – Eseguire il comando File - Carica con un programmatore.



Per il resto valgono i consigli già forniti a proposito del caricamento del bootloader con la versione 1.0.1.

Cap. 4 Programmare ATmega328P Stand Alone configurato a 8MHz o 1MHz

Vediamo ora come realizzare un circuito Stand Alone usando l'oscillatore interno a 8MHz del chip ATmega328P e potendo così escludere dal circuito il quarzo da 16MHz ed i due condensatori da 22pF collegati tra i pin 9 e 10 del chip e il negativo dell'alimentazione (GND);[17] vedremo inoltre come con una semplicissima modifica è possibile dividere per 8 tale frequenza interna, facendo lavorare così il chip a 1MHz.

Ma qualcuno osserverà «Come? Nell'epoca in cui ogni nuovo chip è più veloce del precedente torniamo indietro invece di andare avanti?»; la ragione dello studio di questa tecnica non va ovviamente nella direzione della velocità del chip, bensì serve a ridurre i consumi che, come ben sappiamo, sono direttamente proporzionali alla velocità.

Nelle varie prove ho misurato la corrente assorbita a riposo (intendendo con ciò che il consumo non era gravato da led o altri componenti esterni al chip), nella tipica configurazione a 16MHz e alimentazione a 5V, ho rilevato dai 20 ai 30mA; «niente!» dirà qualcuno, ma chi avesse necessità, come l'ho avuta io, di realizzare un circuito da alimentare con una batteria da 12V tipo telecomandi, 30mA significherebbero doverla sostituire ogni paio di mesi!

Quando invece sono andato a misurare il mio chip alimentato a 3,3V e impostato per lavorare ad 1MHz di clock, ed ho misurato un assorbimento di SOLI 2,5mA ho urlato ciò che urlò di Doctor Frankenstein (o Frankenstin, come voleva farsi chiamare prima di seguire le tracce del proprio nonno), nel celeberrimo film di Mel Brooke: "**SI PUÒ FARE!!!**"; con questa battuta intendevo dire che ora la strada era aperta, ed infatti nei mesi successivi mi sono messo al lavoro per sperimentare una serie di tecniche software per ridurre i consumi, nei periodi di inattività del micro, arrivando a leggere circa 50nA, praticamente 0!!! In questa sezione creeremo due nuove board virtuali, la tecnica ormai la conosciamo, vedremo come programmare i chip in modo che possano funzionare a bassa velocità.

Useremo ancora le due tecniche, ormai note come "seriale" e "ISP".

[17] Il Tutorial ufficiale "elimina" anche la resistenza da 10K collegata al pin 1 (Reset) del chip; questo non c'entra nulla col fatto che si fa lavorare il chip a frequenza inferiore, comunque ho fatto molte prove e non ci sono stati problemi con il Reset.

p. 8 – Le tecniche per programmare un chip in stand alone, con oscillatore interno a 8MHz o 1MHz

8A – La modifica del file boards.txt

Questa volta iniziamo dalla modifica del file, e c'è una ragione ben precisa; il chip opera di default con l'oscillatore quarzato esterno a 16MHz, affinché possa lavorare con il clock interno a 8MHz, o con questo diviso per 8, cioè ad 1MHz, bisogna effettuare alcune modifiche alle impostazioni interne dei fuses; queste modifiche, nel nostro caso, si possono effettuare in un solo modo: programmare il chip con una board virtuale creata precedentemente con i necessari valori.[18]

La procedura di creazione delle due nuove board virtuali (una per ogni clock) ormai è nota, l'abbiamo vista nello specifico qui; ho inserito questi due gruppi di righe subito sotto quelle della board virtuale precedente, quindi in quarta e quinta posizione, ribadisco però che ciò non ha importanza, la posizione rappresenta solo il punto in cui le varie board "Stand Alone" appariranno nell'elenco dell'IDE.

Queste le righe da inserire nel file board.txt per la board a 8MHz :

```
mega3208.name=ATmega in Stand Alone 8MHz internal clock (w/ Arduino as ISP)
mega3208.upload.protocol=stk500
mega3208.upload.maximum_size=32768
mega3208.upload.speed=115200
mega3208.upload.using=arduino:arduinoisp
mega3208.bootloader.low_fuses=0xe2
mega3208.bootloader.high_fuses=0xdf
mega3208.bootloader.extended_fuses=0x07
mega3208.bootloader.path=optiboot
mega3208.bootloader.file=optiboot_atmega328.hex
mega3208.bootloader.unlock_bits=0x3F
mega3208.bootloader.lock_bits=0x0F
mega3208.build.mcu=atmega328p
mega3208.build.f_cpu=8000000L
mega3208.build.core=arduino
##############################################################
```

Notate che rispetto alla stand-alone 16MHz vista in precedenza, qui cambiano il valore del low_fuses ("**E2**") ed il valore della riga mega3208.build.f_cpu che è "**8000000L**" invece di "16000000L", oltre naturalmente ai caratteri "codice" che sono "mega3208".

[18] Questi valori si calcolano impostando correttamente alcuni parametri in un programma specifico che si trova a questo link: http://www.engbedded.com/fusecalc/
Poiché l'uso di questo programma non rientra negli obiettivi di questa Guida, sappiate che è stato utilizzato per calcolare i valori che da impostare nelle varie board virtuali che stiamo descrivendo.

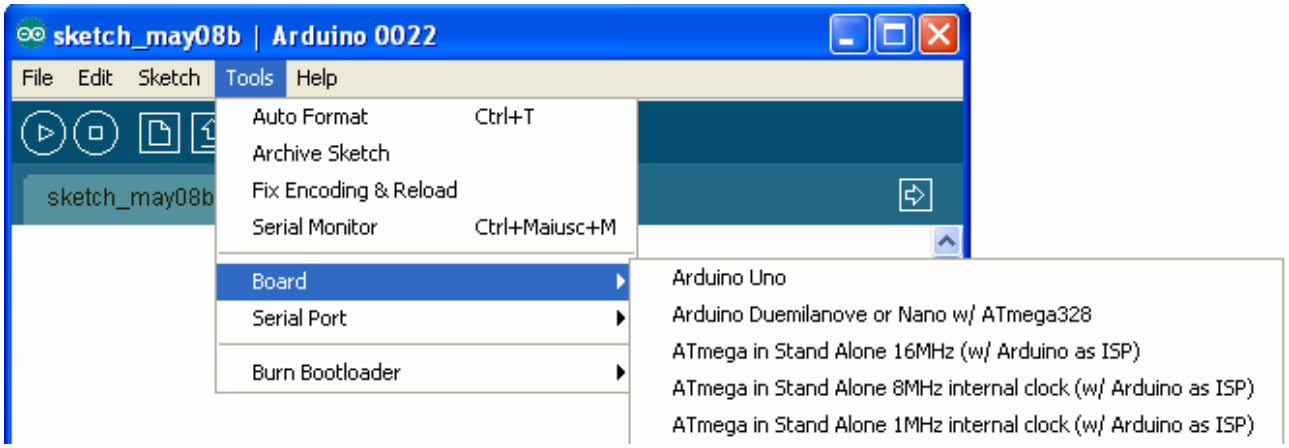
E queste invece le righe da inserire per la board a 1MHz:

```
mega3201.name=ATmega in Stand Alone 1MHz internal clock (w/ Arduino as ISP)
mega3201.upload.protocol=stk500
mega3201.upload.maximum_size=32768
mega3201.upload.speed=115200
mega3201.upload.using=arduino:arduinoisp
mega3201.bootloader.low_fuses=0x62
mega3201.bootloader.high_fuses=0xdf
mega3201.bootloader.extended_fuses=0x07
mega3201.bootloader.path=optiboot
mega3201.bootloader.file=optiboot_atmega328.hex
mega3201.bootloader.unlock_bits=0x3F
mega3201.bootloader.lock_bits=0x0F
mega3201.build.mcu=atmega328p
mega3201.build.f_cpu=1000000L
mega3201.build.core=arduino
##############################################################
```

Anche in questo caso le variazioni sono: low_fuses = “**62**”, build.f_cpu= “**1000000L**”, codice= “**mega3201**”. Ormai è chiaro che il valore build.f_cpu corrisponde alla velocità di clock del micro, espressa in Hz.

N.B.: le righe delle board virtuali possono essere inserite in un punto qualsiasi del file, a vostro piacimento, ma sempre al di sotto di una linea di “#########”, in pratica si dovranno trovare singolarmente racchiuse tra due linee di “#########”, in caso contrario non saranno riconosciute e potrebbero addirittura “disturbare” le board adiacenti nel file.

Dopo questa semplice operazione, nella quale bisogna ovviamente fare moltissima attenzione a non sbagliare nemmeno un carattere, altrimenti qualcosa potrebbe non funzionare, sarà sufficiente chiudere e salvare il file e quindi aprire nuovamente l’IDE; dovreste vedere le due nuove board, oltre a quella creata nel capitolo precedente, così:



8B – La tecnica ISP

Vedremo come sempre i collegamenti hardware; non vedremo più la PCB perché ormai è chiaro che è l'equivalente di una breadboard, vedremo invece la breadboard in configurazione minimal.

Nota importante: nelle immagini seguenti vedremo anche la configurazione Arduino-ISP-Breadboard minimal, con la quale si potrà effettuare il caricamento di sketch, ma deve essere ben chiaro che il buon fine di questa tecnica presuppone che il **chip abbia già i fuses settati per lavorare a 8MHz o 1MHz**; ciò significa, in poche parole, che la prima volta che si prepara un chip a lavorare con una di queste frequenze, l'operazione deve essere fatta o con due Arduino o con Arduino ed una breadboard in configurazione completa, cioè con quarzo, condensatori, ecc.; questo vale sia che il chip sia vergine, sia che il chip abbia già un bootloader standard a 16MHz, con o senza uno sketch.

Inoltre, **se una qualsiasi operazione su chip a 8MHz/1MHz non va a buon fine è molto probabile che non si riesca più a programmarlo con la breadboard minimal**, bisogna quindi ripartire dal caricamento del bootloader con breadboard standard o con Arduino target.[19]

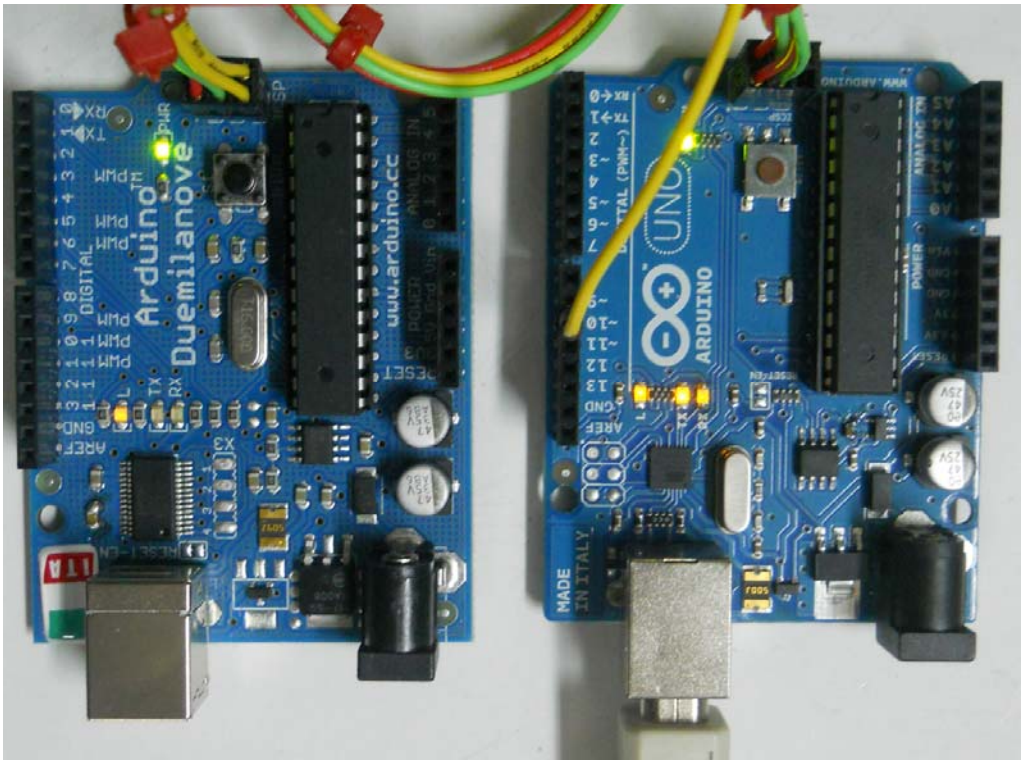
È bene tener conto di queste indicazioni, diversamente si andrà incontro ad una sicura serie di insuccessi!

[19] Per la verità, in alcuni casi l'errore dell'operazione provoca solo un "danneggiamento" dello sketch ArduinoISP caricato nel chip di Arduino, quindi è sufficiente ricaricare su Arduino questo sketch e ripetere l'operazione di programmazione del chip target e tutto funzionerà.

8B1 – Caricare sketch con Arduino-ISP-Arduino

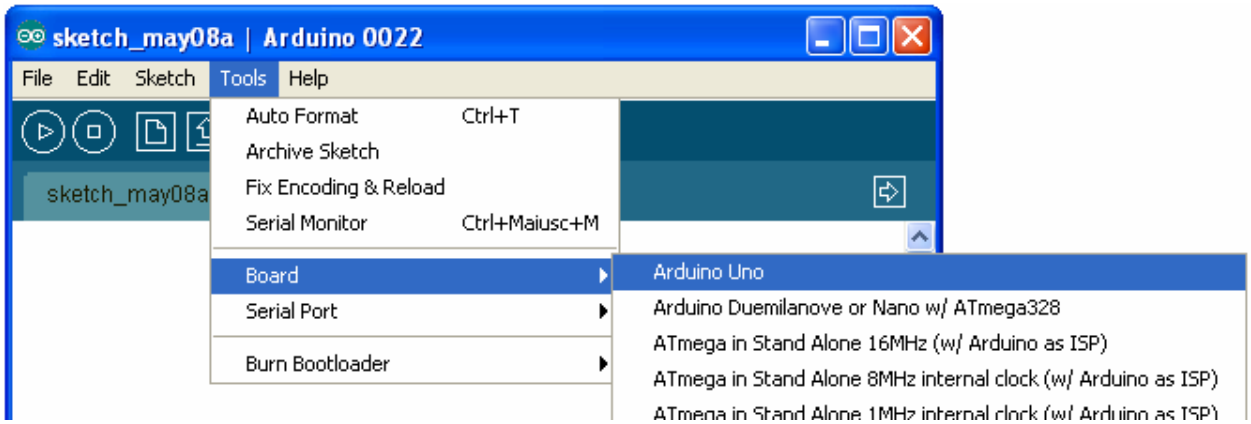
Iniziamo dai collegamenti hardware:

Una coppia di Arduino, tipica immagine con la connessione mediante cavo ISP[20]



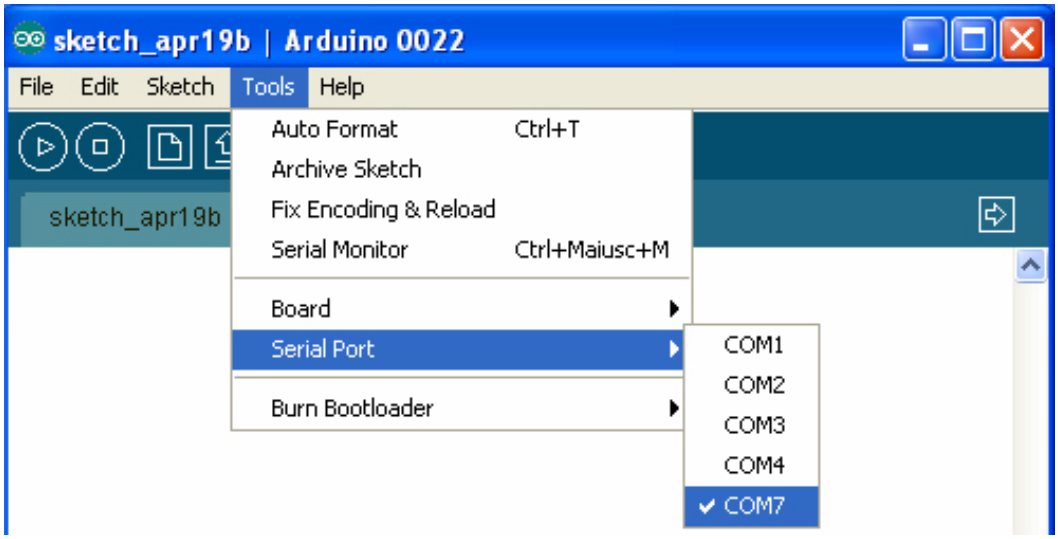
In questa foto abbiamo Arduino UNO in veste di ISP programmer e Arduino 2009 in veste di target;[21] basta ora eseguire questi semplici passaggi:

1 – Aprire l'IDE, selezionare la board che abbiamo collegato al PC tramite la porta USB (nella nostra foto è Arduino UNO) e la relativa COM.

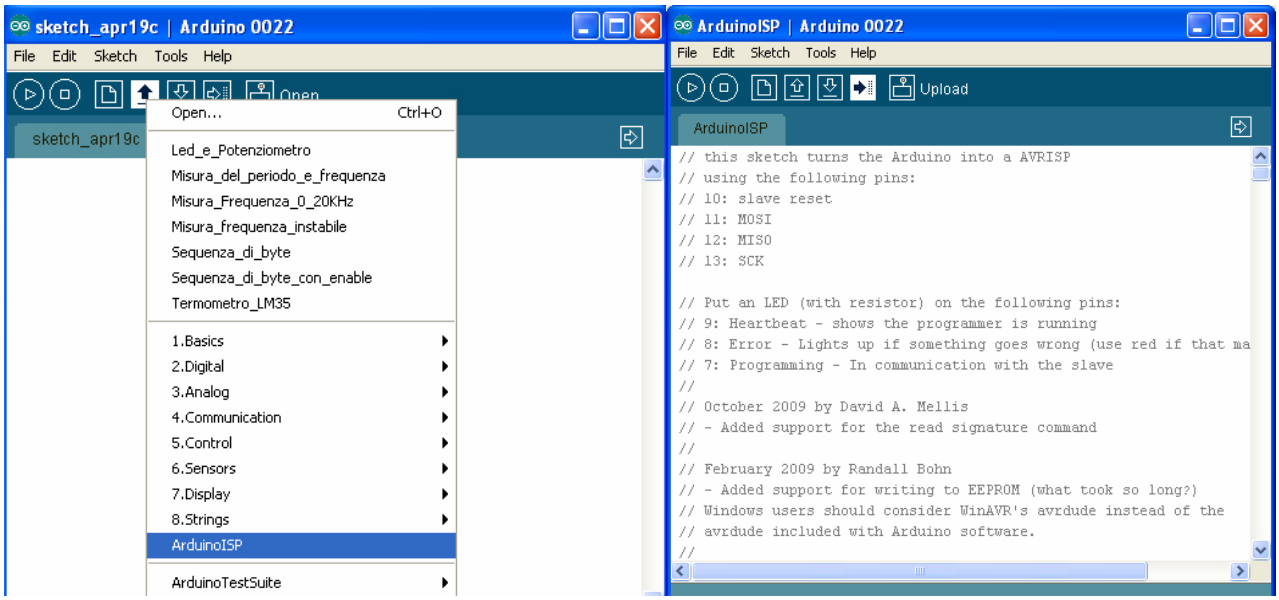


[20] Come sempre è possibile ricorrere ai collegamenti a fili separati già visti più volte in questa Guida; la realizzazione di tale cavetto è però fortemente consigliata in quanto garantisce la certezza di non avere falsi contatti o, peggio, contatti "striscianti", sempre più possibili man mano che si usurano i contatti dei connettori di Arduino per le varie prove con collegamenti di shield o altri circuiti esterni.

[21] Il fatto che anche la board Arduino target sia configurata col quarzo a 16MHz non crea problemi a questo tipo di operazione; in realtà, al caricamento del bootloader, se così non fosse, il chip non verrebbe programmato correttamente

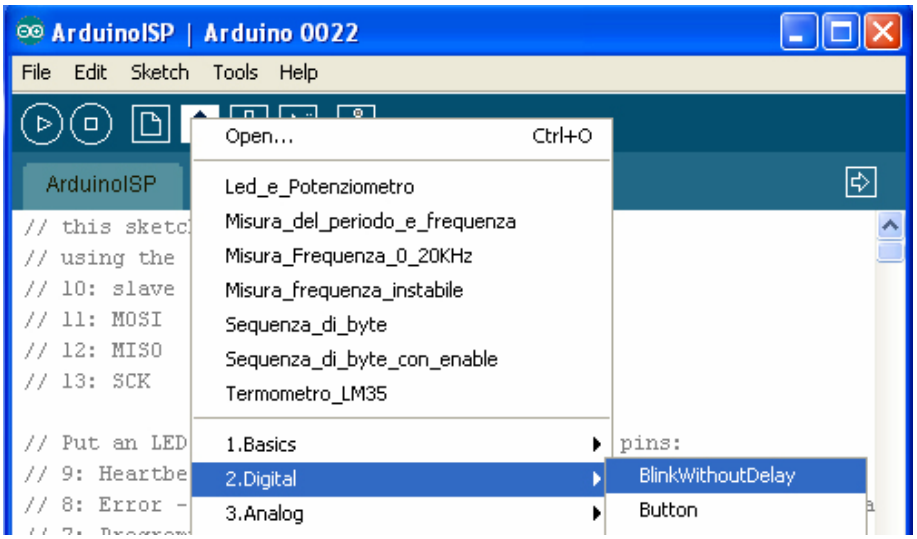


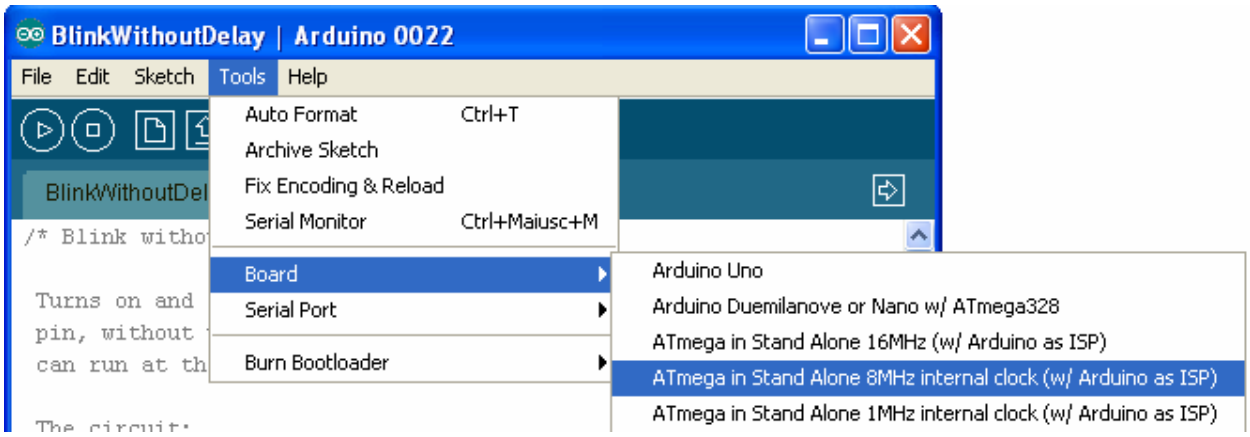
2 – Caricare lo sketch ArduinoISP e inviarlo con Upload ad Arduino UNO:

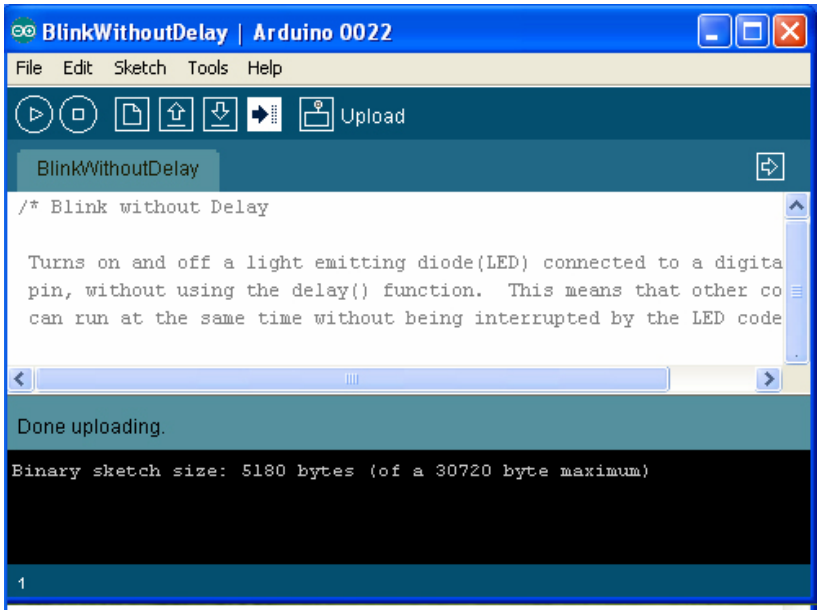


3 – Aprire ora lo sketch da inviare al chip montato sulla board target, quindi selezionare la board "**ATmega in Stand Alone 8MHz internal clock (w/ Arduino as ISP)**" oppure "**ATmega in Stand Alone 1MHz internal clock (w/ Arduino as ISP)**", chiaramente il tipo di board deve essere identico a quello selezionato per il burn bootloader; eseguire Upload:[22]

[22] Il "blink" (con o senza Delay) è lo sketch più semplice ma anche quello che ci permette di vedere subito se il chip funziona nel suo circuito Stand Alone minimal.





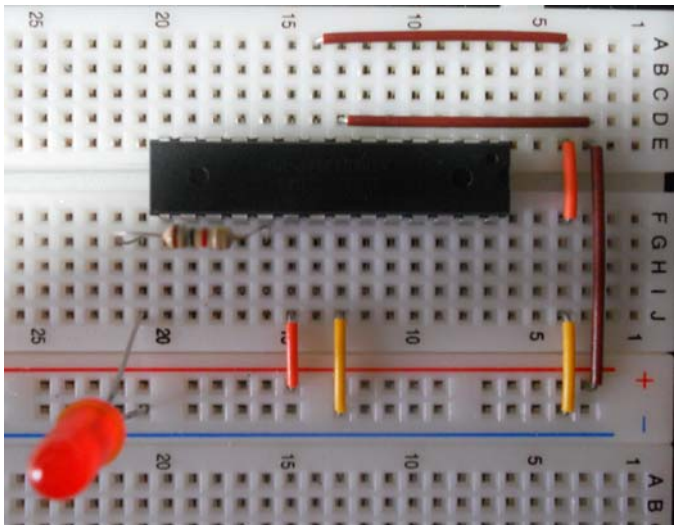


A questo punto il chip può essere montato su un circuito stand alone minimal (v. prossimo paragrafo) e funzionerà subito.

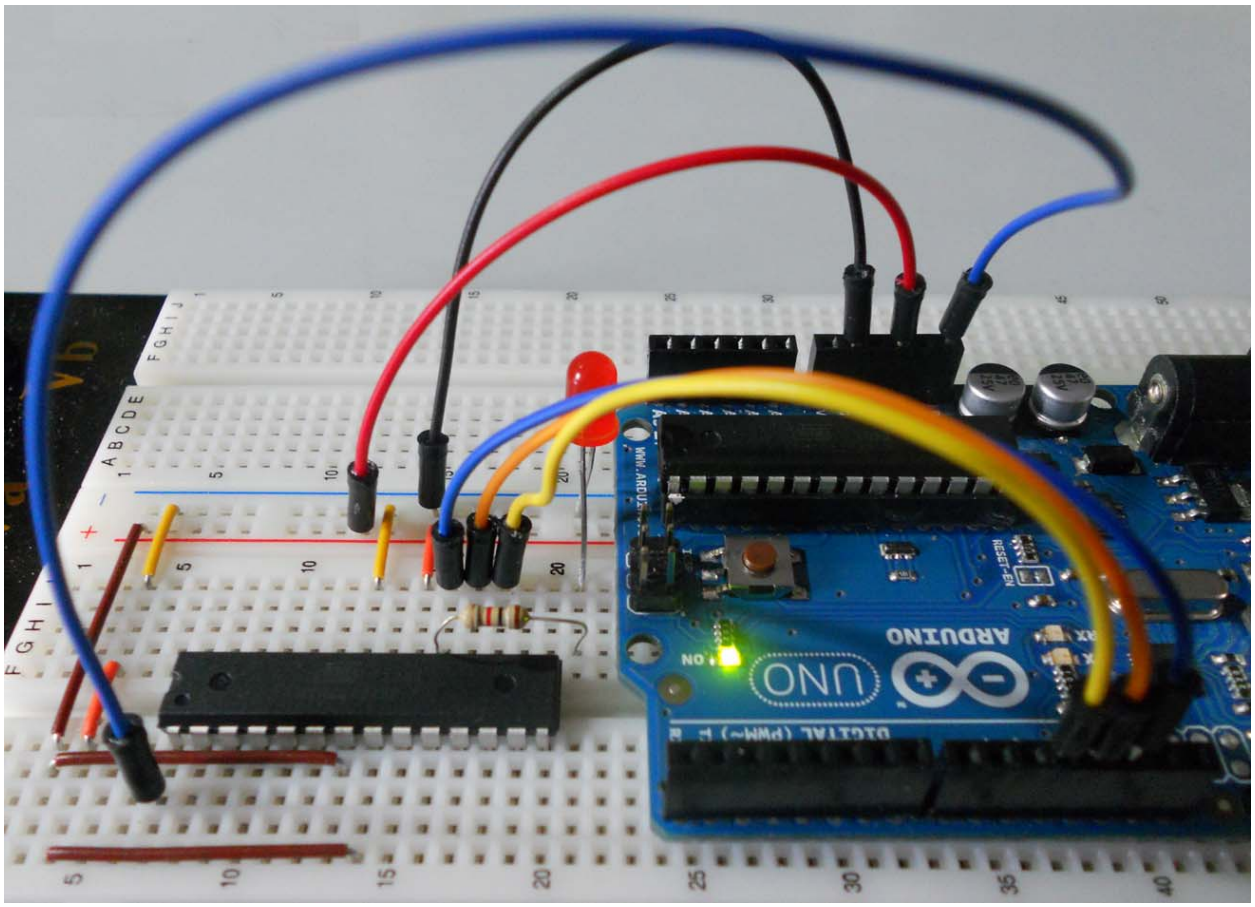
8B2 – Caricare sketch con Arduino-ISP-Breadboard minimal

Al solito prepariamo prima l'hardware:

Davvero "minimal" la breadboard! Si possono abolire anche la resistenza ed il led



Il collegamento ISP tra Arduino e breadboard in configurazione minimal[23]

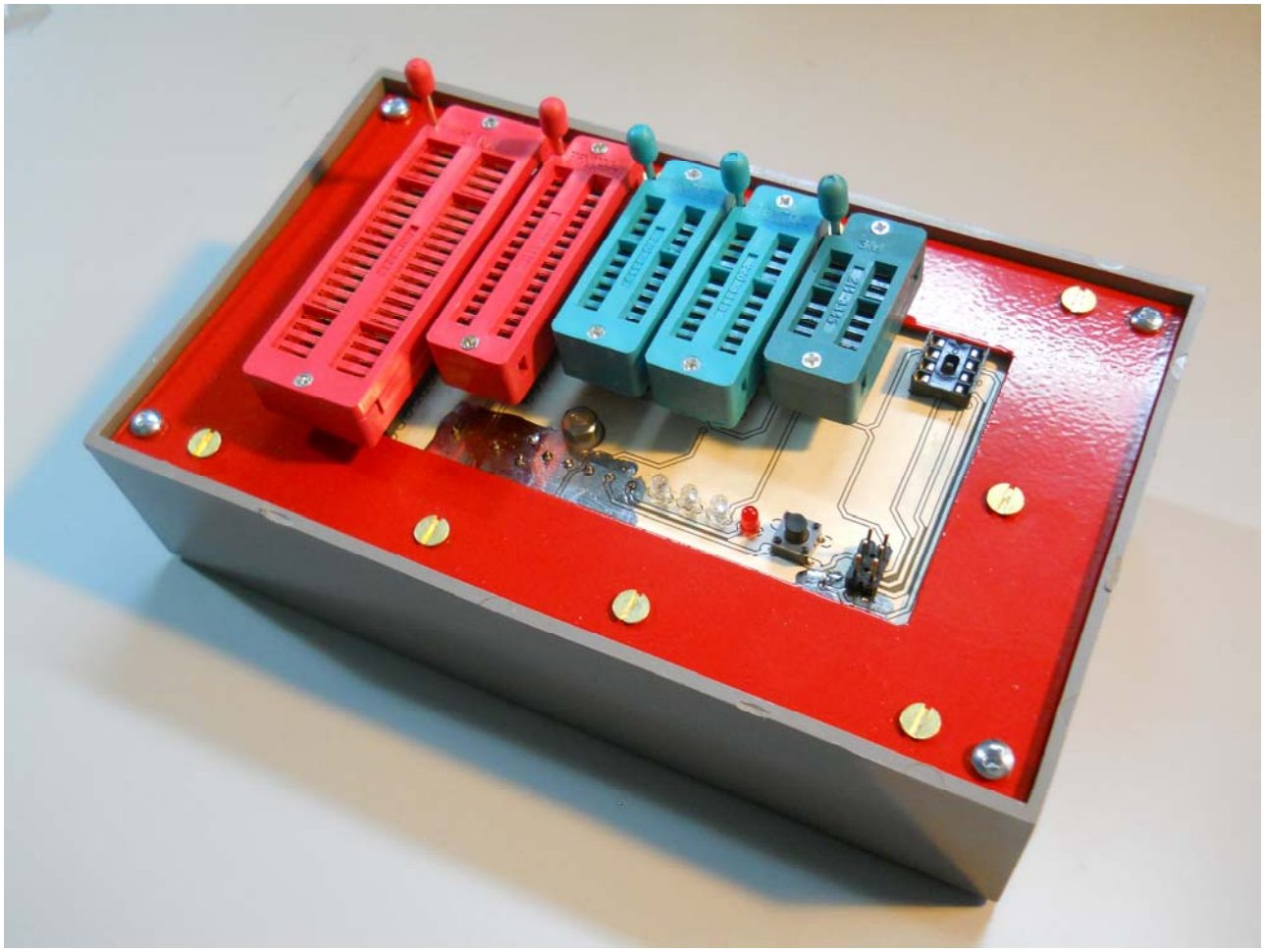


[23] Vi ricordo che questo collegamento può essere realizzato anche sotto forma di cavetto, rendendo l'operazione più semplice e pressoché immune da errori, come visto in precedenza.

Come già spiegato in precedenza questo tipo di collegamento richiede che sulla breadboard sia montato un chip precedentemente preparato con bootloader a 8MHz/1MHz; infatti con questa tecnica potremo solo inviare o aggiornare uno sketch o, al limite, caricare nuovamente il bootloader per la stessa frequenza.

Un nuovo caricamento del bootloader sarebbe necessario solo in caso di variazione della frequenza di clock oppure in caso di errore, ecco perché consiglio fortemente, a chi pensa di poter adoperare molto queste tecniche, di comprare un secondo Arduino o di realizzare una PCB “dotata di tutti i comforts” e quindi anche di una presa 6 poli ISP; a fronte della spesa non elevata e (nel secondo caso) di un po’ di tempo da dedicare alla realizzazione, i vantaggi sono infiniti, come sto sperimentando io ogni giorno.

A tal proposito segnalo un bellissimo progetto, che ho realizzato e pubblicato sul numero 164 di Elettronica In: un Programmatore ISP molto bello e potente; collegato col solito cavetto ad una qualsiasi board Arduino, permette di effettuare programmazioni ISP di moltissimi microcontrollori ATMEL, ma per approfondimenti Vi rimando alla lettura della Rivista. Qui mi limito a pubblicarvi la foto:



In basso a sinistra potete vedere il classico connettore ISP.

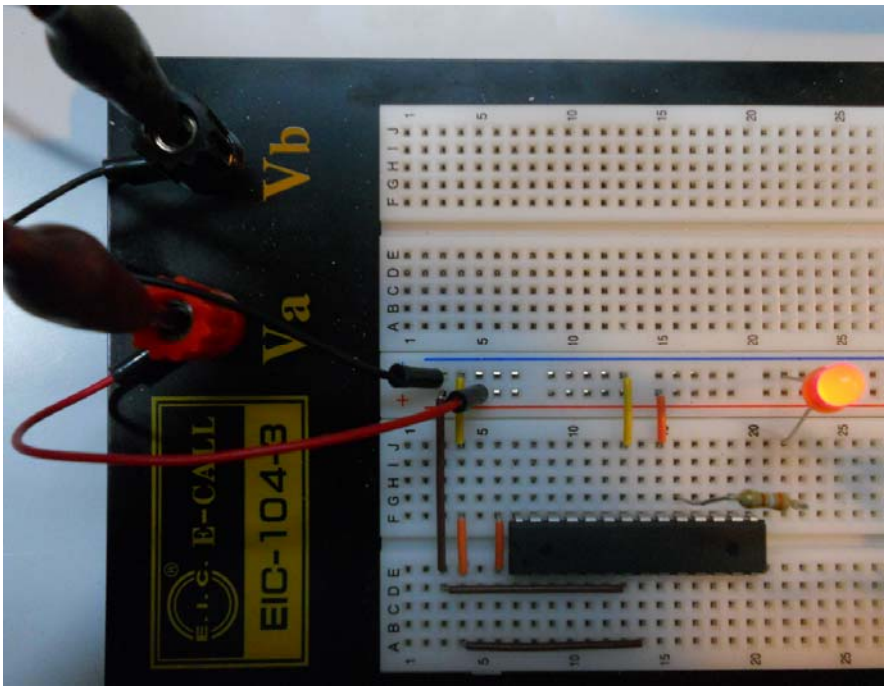
La procedura di caricamento dello sketch è la solita, vediamola in breve:

1 – Aprire l'IDE, selezionare la board che abbiamo collegato al PC tramite la porta USB (nel nostro caso è Arduino UNO) e la relativa COM.

2 – Caricare lo sketch ArduinoISP e inviarlo con Upload ad Arduino UNO.

3 – Aprire ora lo sketch da inviare al chip montato sulla breadboard, quindi selezionare la board "**ATmega in Stand Alone 8MHz internal clock (w/ Arduino as ISP)**" oppure "**ATmega in Stand Alone 1MHz internal clock (w/ Arduino as ISP)**", chiaramente il tipo di board deve essere identico a quello usato per il burn bootloader, quando si è preparato il chip; eseguire Upload.

Il chip ATmega328P su breadboard in configurazione minimal, al lavoro col blink



8C – La tecnica seriale

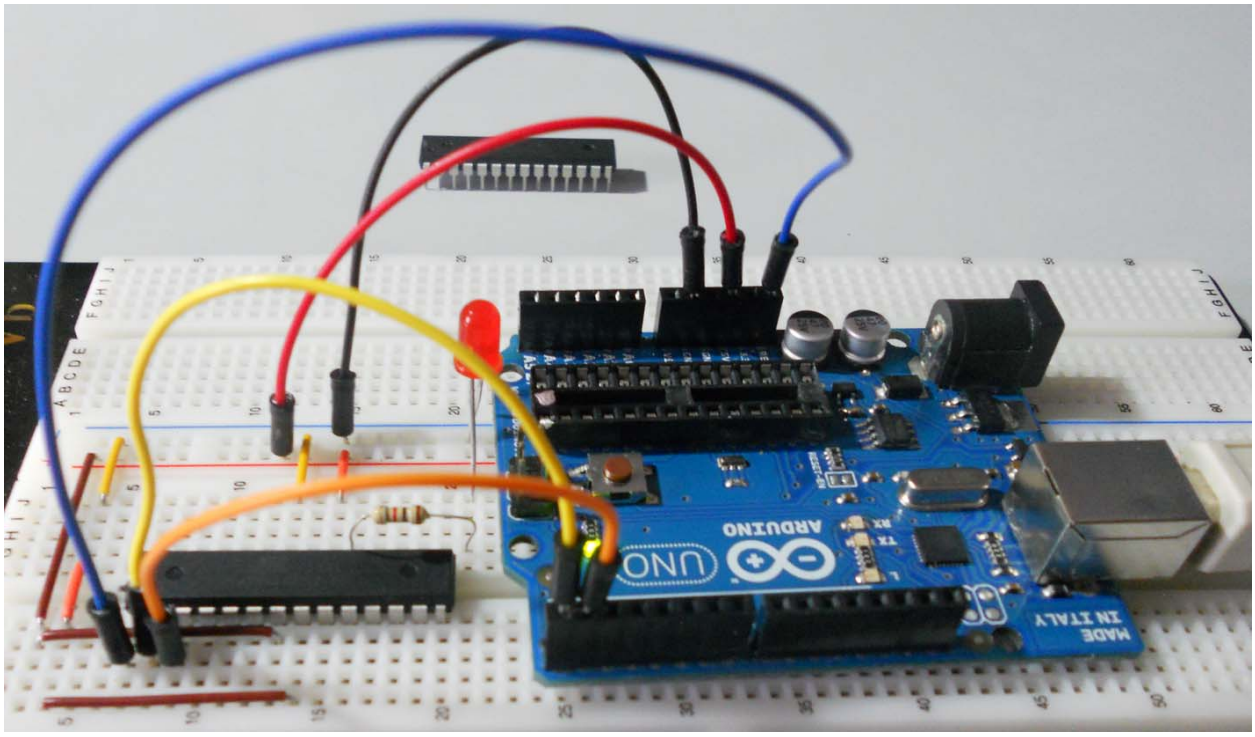
Ho già spiegato che **per poter lavorare con questi chip a frequenza inferiore è indispensabile che abbiano precaricato il bootloader specifico per la frequenza desiderata**, ma abbiamo visto come questa operazione si possa effettuare agevolmente.

La tecnica seriale, applicata ai chip programmati per lavorare a frequenze inferiori allo standard, fondamentalmente consiste nel **togliere il chip originale dall'Arduino** e collegare la board (che assume le funzioni di programmatore) ad una breadboard in configurazione minimal, con funzione "target", cioè su cui è installato il chip da programmare.

Non avrebbe infatti alcun senso usare due Arduino, ecco perché NON vedremo questa tipologia hardware.

Al solito i collegamenti da fare sono di tipo pin-to-pin, cioè ogni segnale di Arduino va collegato al corrispondente segnale del circuito target. I segnali da collegare sono: +5V, GND, tx, rx e reset.

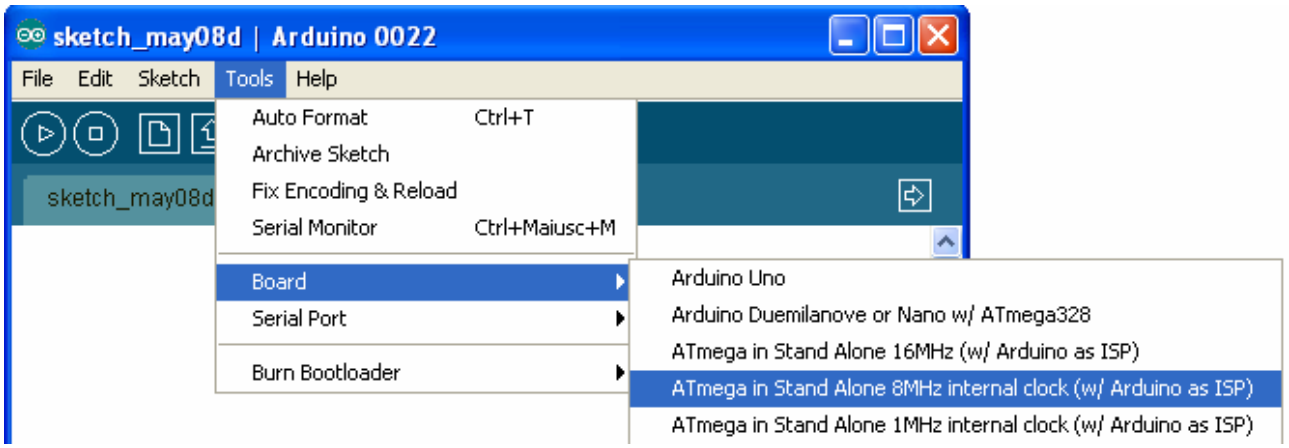
Arduino e breadboard in configurazione minimal e collegamento seriale:
(da notare il chip originale tolto da Arduino e messo da parte….per non dimenticare…)

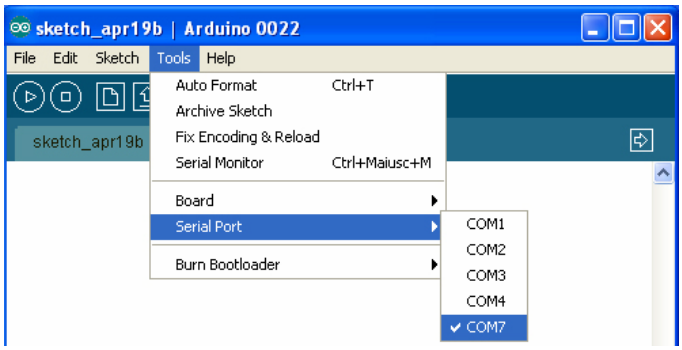


E passiamo subito alla fase software, quella del caricamento dello sketch.

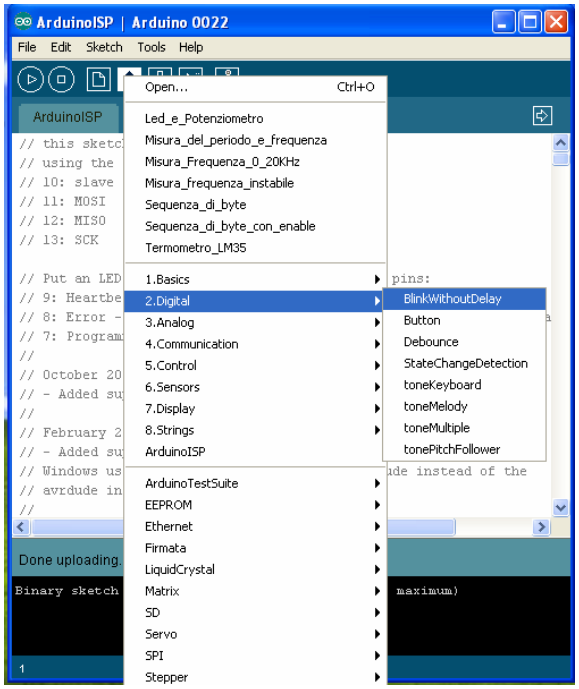
1 – Aprire l'IDE e selezionare la COM di connessione della board Arduino; **NOTA IMPORTANTE**: invece la board da selezionare NON è quella fisicamente collegata al PC, bensì il tipo corrispondente al bootloader caricato nel chip da programmare. P.es. se sul chip avete precedentemente caricato il bootloader "Stand Alone a 1MHz", ora come board

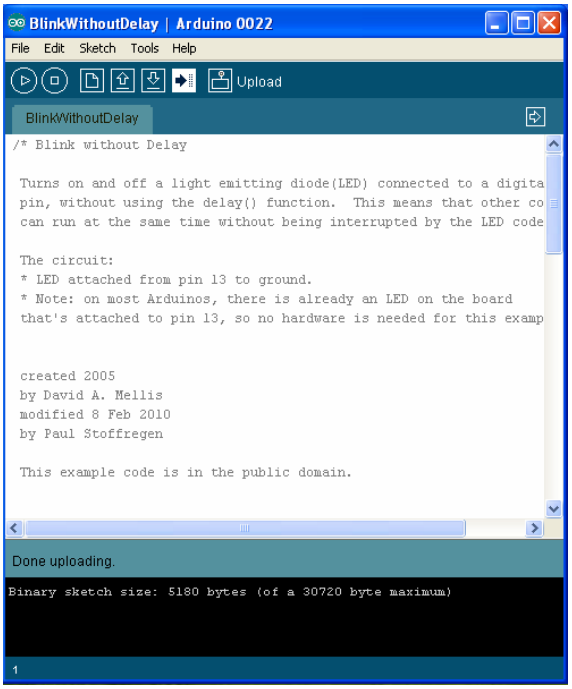
dovete scegliere proprio questo modello, ignorando quindi il tipo di Arduino che state usando come programmatore.





2 – Caricare lo sketch ed inviarlo con upload al chip stand alone:





8D – Indicazioni tecniche sui chip programmati a 8MHz o 1MHz

Chiudiamo questa sezione con alcune indicazioni tecniche, riscontrate durante le moltissime prove effettuate per preparare la Guida, relativamente a questa sezione.

Dopo aver programmato un chip per lavorare col clock interno a 8MHz, o con lo stesso diviso per 8 (1MHz), se lo lasciate su Arduino o su una breadboard in configurazione standard, il circuito che farà riferimento all'oscillatore interno e ignorerà la presenza del quarzo esterno..

Questa è una cosa importante, in quanto in realtà, usando due Arduino o un Arduino ed una breadboard in configurazione standard (e NON minimal!) è possibile caricare il bootloader, inviare gli skecth e fare tutte le necessarie prove di funzionamento, senza ricorrere al circuito minimal, utilissimo invece per lo schema finale, per risparmiare spazio e consumo ed anche per risparmiare il costo dei componenti aggiuntivi.

L'altra cosa da tenere ben presente riguarda i tempi di esecuzione ed i comandi "temporali", come delay, millis, ed altri che fanno riferimento al clock.

Se riscontrate tempi strani rispetto a quelli che vi aspettate, certamente avete omesso di programmare i fuse, mediante caricamento del bootloader, prima di caricare lo sketch, oppure avete caricato il bootloader usando una board impostata per una frequenza diversa, rispetto a quella usata per caricare lo sketch. In questi casi conviene rifare completamente la doppia operazione bootloader & sketch e tutto si sistemerà.

Cap. 5 Come programmare altri micro della famiglia ATMEL

Una volta che, con la versione precedente della Guida, sono stati sviscerato i modi principali per programmare i microcontrollori ATmega328P, ovvio che si è dedicato tempo per allargare gli orizzonti, cioè per riuscire ad usare altri tipi di microcontrollori. In questo capitolo non farò una trattazione completa, non basterebbe un'altra Guida intera! Però fornirò gli elementi necessari per comprendere le modalità e gli elementi necessari per poter programmare altri microcontrollori.

La problematica non è assoluta, ATMEL mette a disposizione strumenti hardware e software molto potenti per programmare tutti i microcontrollori che produce; i problemi sorgono quando si vogliono programmare questi chip usando Arduino e l'IDE.

Le informazioni indispensabili per la programmazione di un microcontrollore sono racchiuse in una serie di file che sono raggruppati sotto il nome di "**core**". Tali informazioni sono fondamentalmente di natura hardware e riguardano appunto la struttura del microcontrollore, p.es.: la piedinatura, la disposizione dei pin I/O, degli eventuali pin del convertitore ADC (i cosiddetti pin analogici), dei pin con funzione PWM, il numero e le caratteristiche dei timer del microcontrollore, e svariate altre che non ha importanza descrivere ora.

Solo disponendo di queste informazioni l'IDE di Arduino è in grado di programmare tali microcontrollori.

Altro dato importante riguarda il file (in dotazione all'IDE) AVRDUDE.CONF che fornisce informazioni all'altro file AVRDUDE.EXE per la compilazione del firmware da caricare nel micro. Il file .CONF (si trova in arduino-xxxx\tools\avr\etc) contiene una serie di altre informazioni, tra cui p.es. la signature di ogni microcontrollore: essa è costituita da tre byte, memorizzati in un'area protetta della flash memory, che identificano univocamente ogni modello di micro ATMEL.

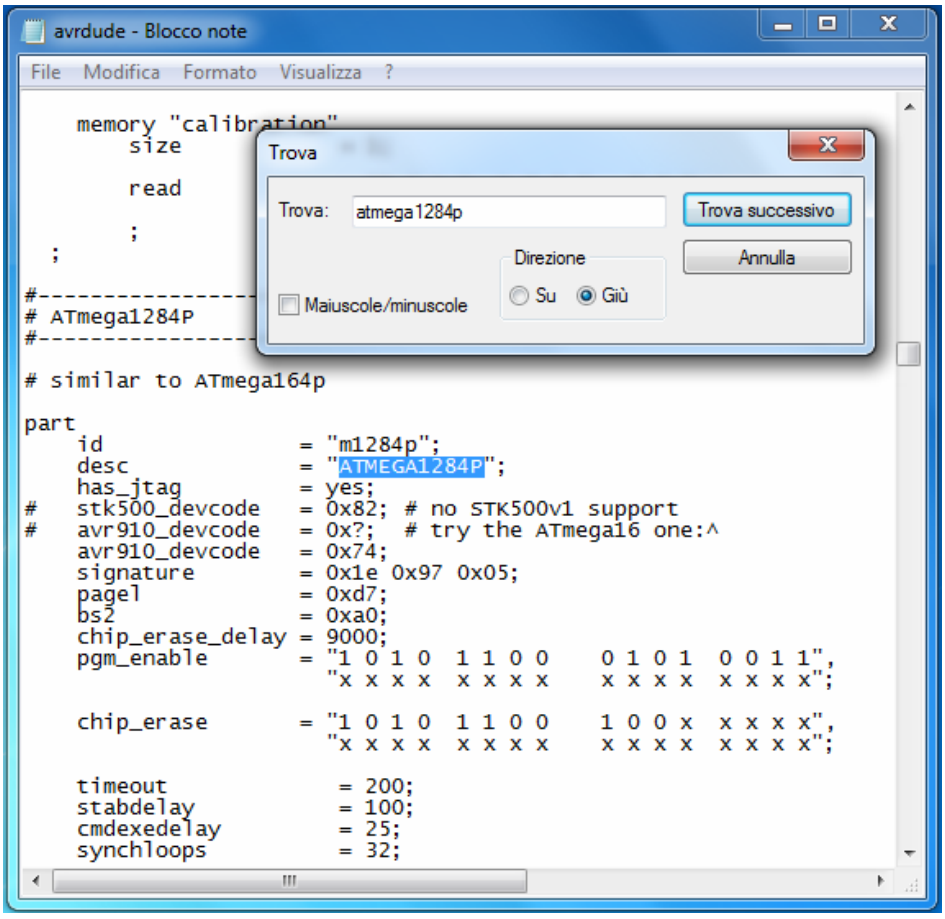
Nel nostro esempio immaginiamo di disporre di un ATmega1284P (un grosso chip ATMEL a 40W pin) e di volerlo far lavorare in stand-alone con clock interno a 8MHz. I "**core**" sono reperibili gratuitamente su Internet, non sempre sono completi ed affidabili, ma in genere fanno il loro onesto lavoro e, nella stragrande maggioranza dei casi, li trovate forniti anche delle board virtuali per le varie velocità.

Se disponete dell'IDE 0022 andate su Internet all'indirizzo:
http://www.leonardomiliani.com/?page_id=374 (il sito del mio carissimo amico Leonardo Miliani, noto sul Forum come Leo72) e cliccate sulla voce "Arduino1284"; scaricate scompattate la cartella e copiatela all'interno di arduino-0022\hardware.

Se invece disponete dell'IDE 1.0.1 andate su Internet all'indirizzo: http://maniacbug.wordpress.com/2011/11/27/arduino-on-atmega1284p-4/, trovate la voce "1.Download the ZIP File", cliccate, scaricate e scompattate la cartella e copiatela all'interno di arduino-101\hardware.

A questo punto siete forniti di core, bootloader e board virtuali, che però sono predisposte per inserire un bootloader nel micro, mentre noi vogliamo creare uno stand-alone puro, per cui andremo comunque a modificare il file boards.txt in dotazione a questi core.

Ora dovete sincerarvi di disporre delle informazioni del micro nel file AVRDUDE.CONF; aprite con un qualsiasi elaboratore testi e cercate "atmega1284p".



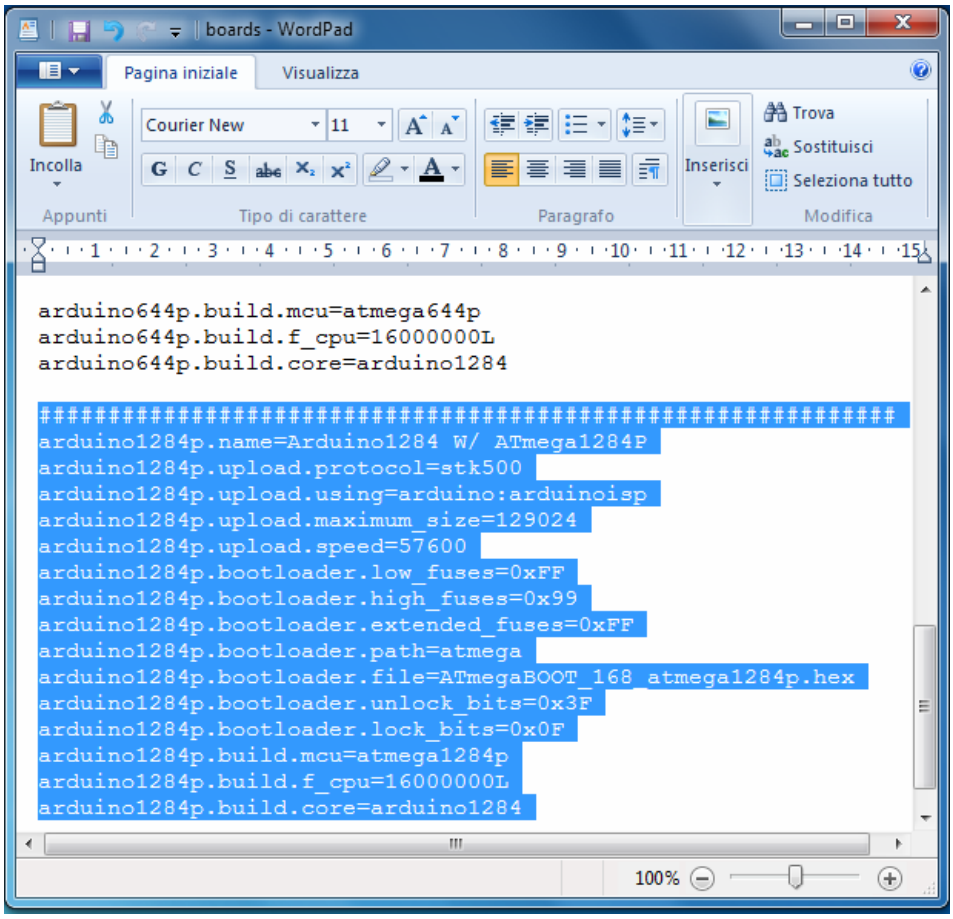
Se nel vostro file AVRDUDE.CONF non trovate il micro dovete necessariamente cercarne una versione più recente, diversamente il solo core non vi sarà utile; comunque la versione del file in dotazione all'IDE 1.0.1 è molto aggiornata e può essere copiata ed utilizzata anche nella 0022.

Andiamo ora ad aprire e modificare i file boards.txt appena scaricati, per aggiungere la board che ci serve. Vi fornisco le relative indicazioni, sia per IDE 0022 che per IDE 1.0.1.

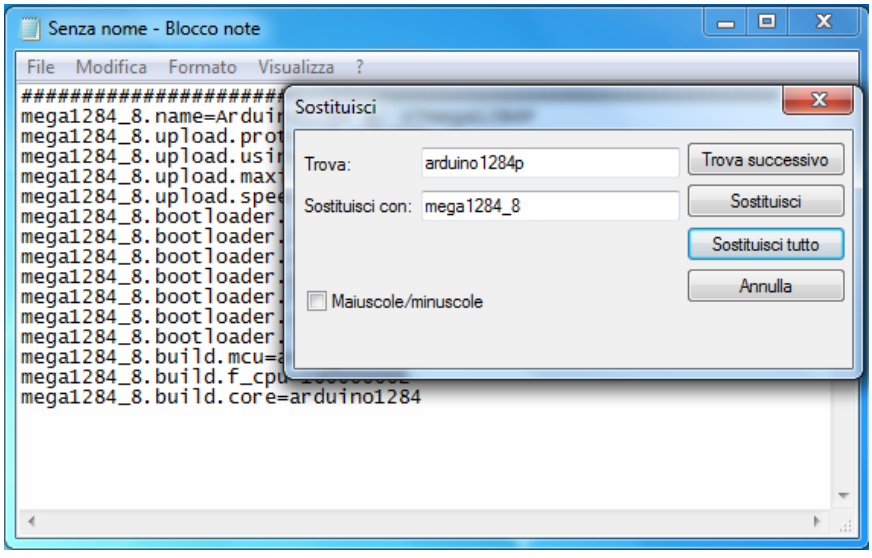
p. 9 – Creare una board virtuale per IDE 0022

Vi fornisco i passaggi in sequenza:

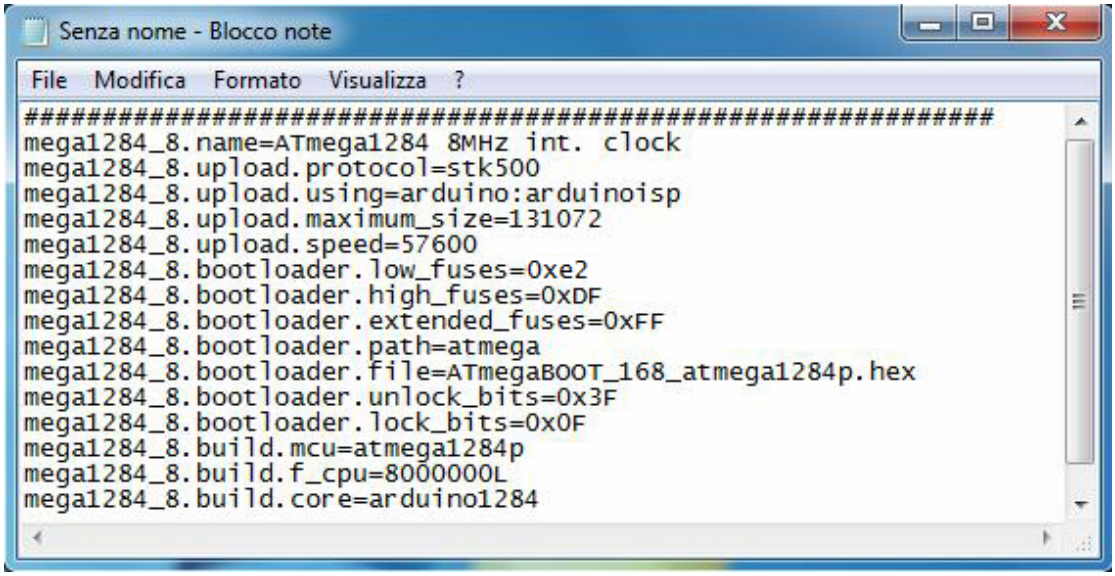
1. Aprire il file boards.txt appena scaricato con il core e copiare le righe della board denominata “Arduino1284 W/ ATmega1284P”, avendo cura di selezionare la riga superiore di “#########”:



2. Incollare queste righe in un programma di elaborazione testi e sostituire il codice “arduino1284p” con “mega1284_8”:



3. Nella prima riga, dopo “name=” scrivere **ATmega1284 8MHz int. clock**; questo è un nome arbitrario, è quello che uscirà nell’elenco delle board dell’IDE, quindi potete modificarlo a piacimento.
4. La terza riga è: **mega1284_8.upload.using=arduino:arduinoisp**; è indispensabile e serve, come abbiamo già visto, per informare l’IDE del fatto che programmeremo questo micro usando Arduino come Programmatore ISP; non dovete variare nulla.
5. Nella quarta riga, dopo “maximum_size=” scrivete **131072**; questa è la massima quantità di memoria flash disponibile per questo microcontrollore.
6. Modificare i valori dei tre fuses in: low=**E2**, high=**DF**, extended=**FF** (i valori devono essere sempre preceduti da 0x e si possono scrivere in minuscolo o maiuscolo, indifferentemente).
7. Nella penultima riga, dopo “build.f_cpu=” scrivete **8000000L**; questo è il fattore di clock espresso in Hz; questo il risultato finale:

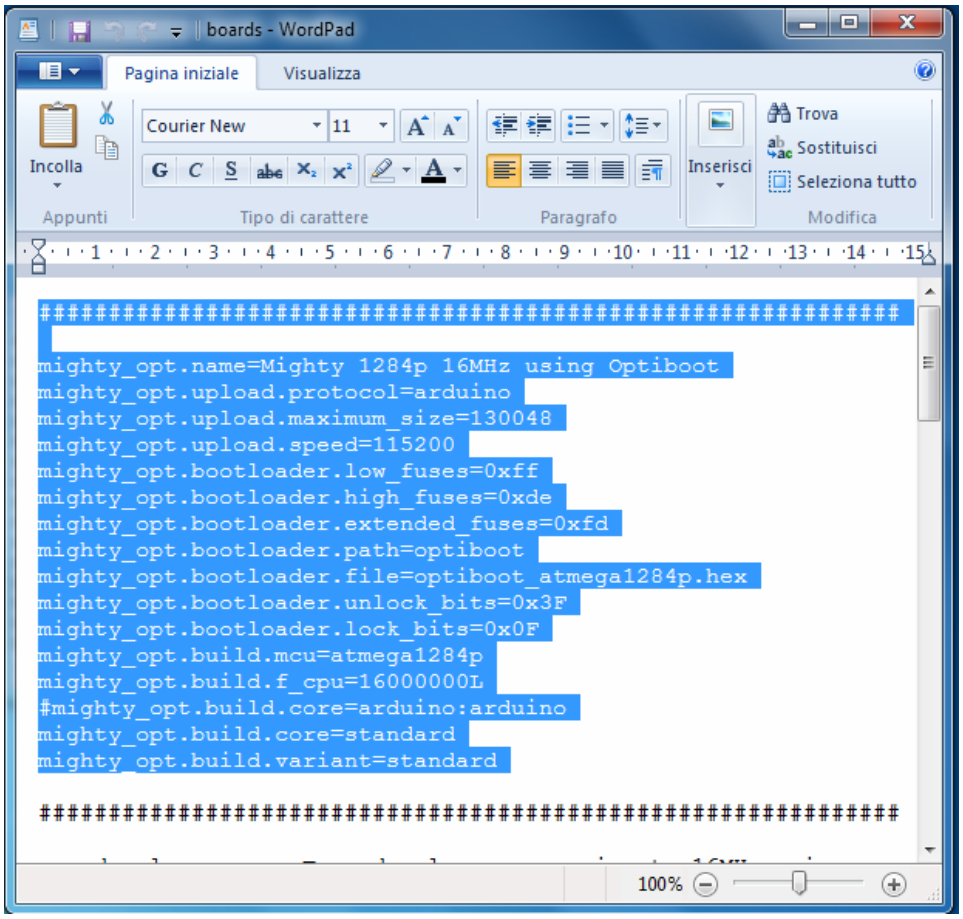


8. A questo punto copiate tutte queste righe ed incollatele alla fine del file boards.txt originale (quello da cui le avete prelevate per modificarle), quindi salvate e chiudete il file.
9. Ora siete pronti, potete aprire l’IDE 1.0.1 ed usare il micro con le consuete modalità della tecnica ISP. Naturalmente non dovete scordare che per settare correttamente i fuse dovete prima caricare il bootloader e successivamente potete caricare il vostro sketch.

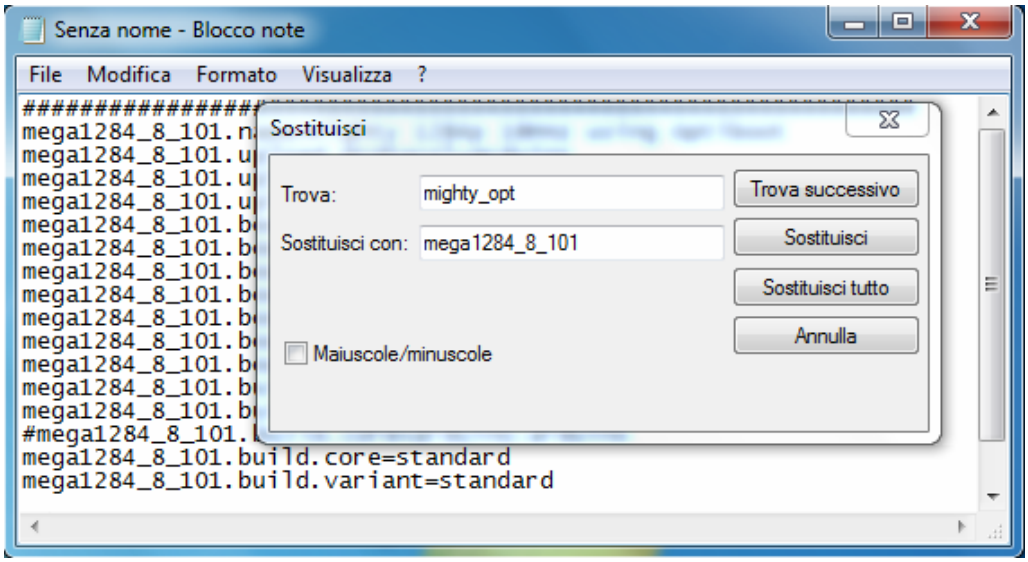
p. 10 – Creare una board virtuale per IDE 1.0.1

Vi fornisco i passaggi in sequenza:

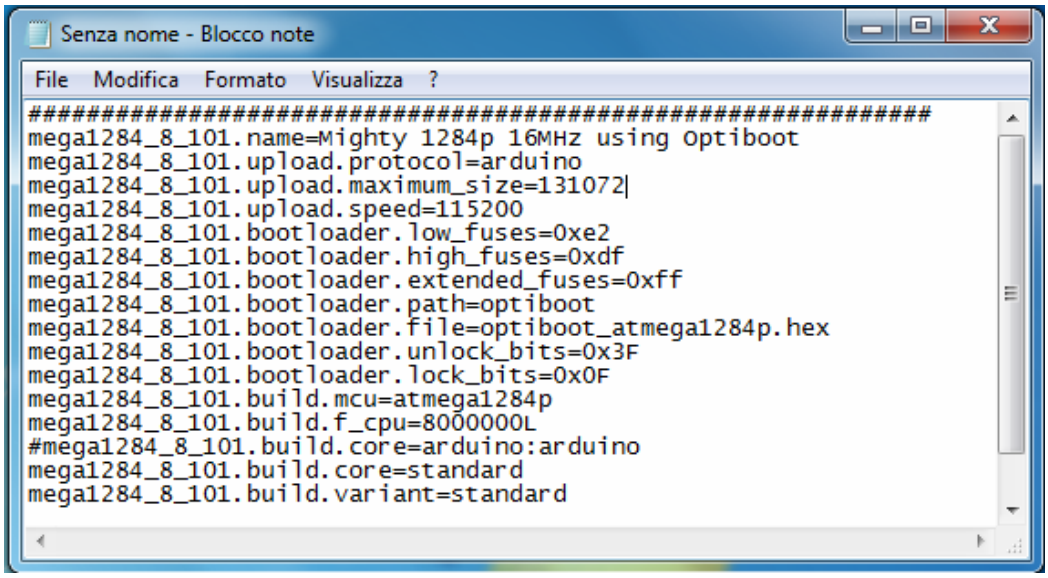
1. Aprire il file boards.txt appena scaricato con il core e copiare le righe della prima board, denominata “Mighty 1284p 16MHz using Optiboot”, avendo cura di selezionare la riga superiore di “#########”:



2. Incollare queste righe in un programma di elaborazione testi e sostituire il codice “mighty_opt” con “mega1284_8_101”:



3. Nella prima riga, dopo “name=” scrivere **ATmega1284 8MHz int. clock**; questo è un nome arbitrario, è quello che uscirà nell’elenco delle board dell’IDE, quindi potete modificarlo a piacimento.
4. In questo caso non c’è una riga **upload.using=arduino:arduinoisp** in quanto l’IDE 1.0.1 prevede già la possibilità di usare Arduino come Programmatore ISP.
5. Nella terza riga, dopo “maximum_size=” scrivete **131072**; questa è la massima quantità di memoria flash disponibile per questo microcontrollore.
6. Modificare i valori dei tre fuses in: low=**E2**, high=**DF**, extended=**FF** (i valori devono essere sempre preceduti da 0x e si possono scrivere in minuscolo o maiuscolo, indifferentemente).
7. Nella quart’ultima riga, dopo “build.f_cpu=” scrivete **8000000L**; questo è il fattore di clock espresso in Hz; questo il risultato finale:



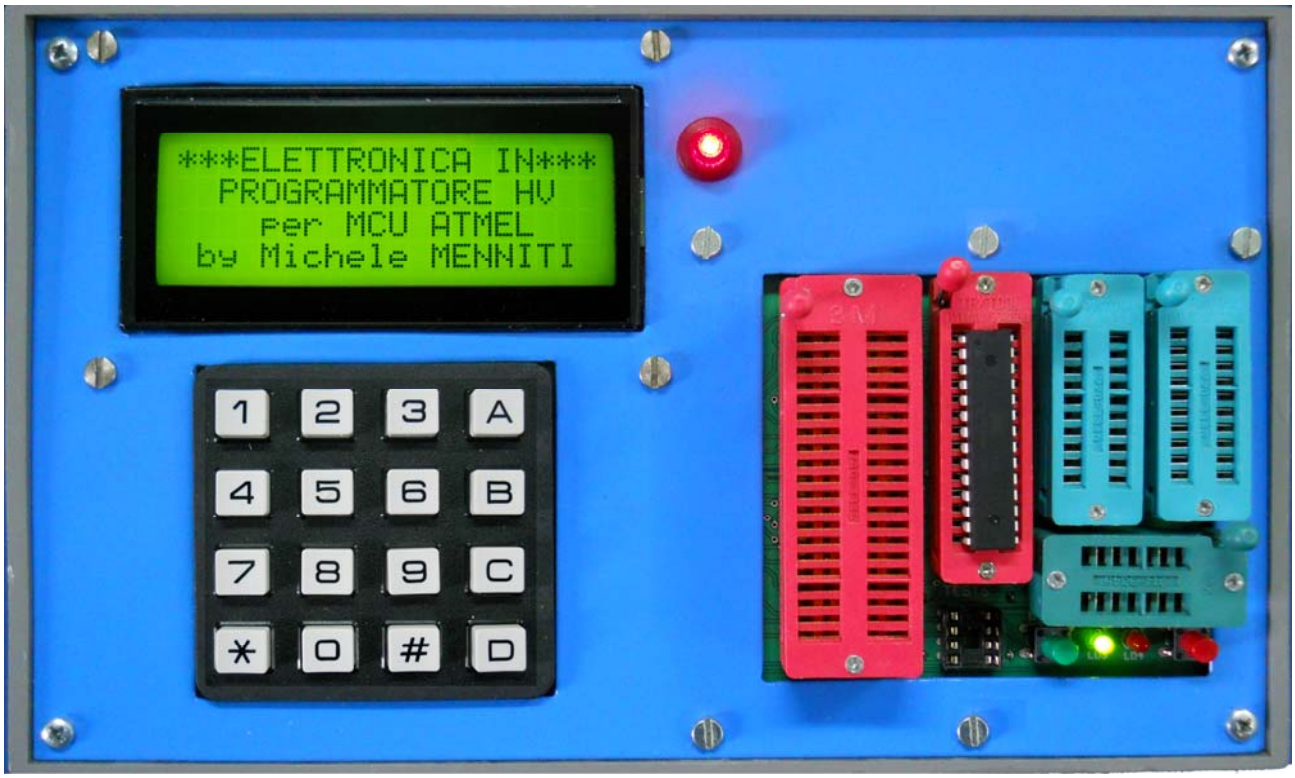
8. A questo punto copiate tutte queste righe ed incollatele alla fine del file boards.txt originale (quello da cui le avete prelevate per modificarle), quindi salvate e chiudete il file.
10. Ora siete pronti, potete aprire l’IDE 1.0.1 ed usare il micro con le consuete modalità della tecnica ISP. Naturalmente non dovete scordare che per settare correttamente i fuse dovete prima caricare il bootloader e successivamente potete caricare il vostro sketch.

Cap. 6 Cenni sui fuse bits

Quello dei fuse è certamente un argomento ignoto a moltissimi Utenti che, quando vogliono fare qualche sperimentazione con nuove board, una volta compresa la tecnica di creazione delle stesse, si scontrano immediatamente con la necessità di calcorare questi due-tre valori che vanno sotto il nome di low_fuses, high_fuses ed extended_fuses.

Moltissime benedizioni a colui che ha ideato e aggiorna il programma gratuito on-line denominato FuseCalc (http://www.engbedded.com/fusecalc/), che permette appunto di calcolare rapidamente questi valori, ma a patto di sapere cosa si sta facendo, diversamente si può andare incontro come minimo a delusioni (micro che non funzionano dopo la programmazione), ma all'estremo opposto c'è il pericolo "chip bricked". Determinate (ed errate) combinazioni di fuse possono dare origine ad una condizione di blocco per la quale non è più possibile sbloccare ed usare il micro, se non ricorrendo alla Programmazione HV.

Anche in questo caso ho realizzato e presentato su Elettronica In (numeri 165, 166, 167, 168) un bellissimo progetto di un **Programmatore HV**, di cui vi mostro un'immagine, che permette di programmare ogni funzione dei micro ATMEL in modo estremamente semplice.



Su quelle stesse pagine ho scritto moltissima teoria, davvero consiglio a tutti, se hanno possibilità, di leggere quei quattro articoli, frutto di mesi di studio intenso dei

principali data-sheet ATMEL. Tanto studio approfondito mi ha permesso di comprendere appieno l'organizzazione e la funzione di ogni singolo bit di questi tre byte, e quindi di spiegarne la programmazione. Non mi dilungherò sulla tematica in questo capitolo, per ragioni di correttezza nei confronti della Rivista, con cui ho obblighi contrattuali sulla non diffusione del materiale che pubblico per un anno dal momento dell'uscita della Rivista.

Però alcuni cenni di massima non si negano a nessuno, inoltre sono sempre presente sul Forum e, come me, altri amici molto esperti, che in pochi secondi sono in grado di fornire valori per qualsiasi tipo di micro.

Tutti i micro ATMEL possiedono due fuse byte: low e high, alcuni possiedono un terzo fuse byte: extended. Per spiegarveli useremo come esempio quelli dell'ATmega328P, sappiate che in altri micro alcuni bit hanno funzione diversa ed in altri ancora certe funzioni non esistono affatto.

Il **low fuse byte** è costituito da 8 bit così organizzati (dal più significativo al meno significativo, tra parentesi la posizione):

- CKDIV8 (7): divide il clock impostato x 8
- CKOUT (6): porta il segnale di clock su un'uscita digitale del micro
- SUT1-0 (5-4): Seleziona il tempo di Start-Up (max 4 combinazioni), fondamentalmente il numero di cicli di clock che vengono eseguiti dopo un RESET o un ritorno da uno stato di SLEEP power-down
- CKSEL3÷0 (3÷0): Seleziona il tipo di clock: oscillatore interno o esterno, clock esterno (max 16 combinazioni moltiplicate per le combinazioni SUT).

L'**high fuse byte** è costituito da 8 bit così organizzati (dal più significativo al meno significativo, tra parentesi la posizione):

- RSTDSBL (7): disabilita il pin di RESET del micro (pericoloso!)
- DWEN (6): abilita il Debug wire sulla linea del pin di RESET
- SPIEN (5): abilita la programmazione seriale e la lettura dei dati dalla memoria del micro. Disabilitare questo fuse significa non poter più programmare il micro né leggerne il contenuto
- WDTON (4): attiva il Watch Dog Timer
- EESAVE (3): preserva il contenuto della memoria EEPROM interna del micro, durante le operazioni di Chip_Erase (cancellazione totale mediante comando software)

- BOOTSZ1-0 (2-1): riserva al bootloader una parte della sezione superiore della memoria flash. L'impostazione di questi bit viene ignorata se il bit BOOTRST è disabilitato.
- BOOTRST (0): se disabilitato imposta il puntamento del vettore di RESET al primo indirizzo della memoria flash (0x0000); se abilitato lo imposta al primo indirizzo dell'area riservata al bootloader mediante i bit BOOTSZ1-0.

L'**extended fuse byte** è costituito da 3 bit (nel caso dell'ATmega328P) così organizzati (dal più significativo al meno significativo, tra parentesi la posizione):

- BODLEVEL2-0 (2÷0): abilita o disabilita il Brown-out Detect Level, in base a diversi livelli di tensione, da intendersi come tensione minima di alimentazione, sotto la quale il micro si pone in perenne stato di RESET.

Secondo il criterio ATMEL tali bit sono programmati (abilitati) quando vengono messi a 0, mentre si intendono non programmati quando sono posti a 1. Di conseguenza ricordate che se nel FuseCalc mettete una spunta su un bit, significa che lo state abilitando e quindi ponendo a 0. E' molto importante comprendere questo meccanismo altrimenti vi usciranno valori sballati nel calcolo manuale, ma anche in questo vi viene incontro FuseCalc, perché in fondo alla pagina fornisce i due-tre valori calcolati in base ai bit spuntati.

Ho spiegato il significato dei fuse bit dell'ATmega328P in base alla loro posizione nei rispettivi byte, come descritti nella sezione "Manual Configuration"; ricavare il valore dei fuse da usare ricorrendo all'impostazione manuale dei singoli bit è operazione che richiede grande dimestichezza e ottima conoscenza di questi parametri, in caso contrario il chip bricked è assicurato!

Il FuseCalc permette anche un diverso metodo di impostazione, infatti nella sezione denominata "Feature Configuration" la situazione è un po' mischiata, ma in compenso le molte combinazioni di clock e start-up sono raggruppate in forma descrittiva ed aiutano non poco nella scelta di quella corretta.; ma comunque anche qui è possibile sbagliare, a volte la scelta non è semplice, anche avendo competenza in materia.

Non aggiungo altro ma questa carrellata certamente chiarirà le idee a molti; a chi volesse approfondire consiglio vivamente la lettura dei miei due articoli sul Programmatore HV, usciti a Giugno e Luglio/Agosto (n. 167 e 168), nei quali ho ampiamente trattato questi argomenti, unitamente ai lock bits, con alcuni esempi di interpretazione.

Cap. 7 I Convertitori USB→Seriale

Ho dedicato diverso tempo a questa tematica, lavorando con diversi circuiti, voglio quindi darvi alcune indicazioni e fare una breve panoramica su questa tipologia di strumento. Il Convertitore USB→Seriale è un circuito che preleva i segnali D+ e D- dalla porta USB del PC e li converte nei tipici segnali di comunicazione seriale bidirezionale: TX, RX, DTR (o RTS); la porta USB inoltre porta l'alimentazione a 5V proveniente dal PC e, in genere, questa viene riportata pari pari sui pin di uscita. In tal modo il convertitore è anche in grado di alimentare il circuito con cui dialoga.

Nell'Arduino 2009 tale convertitore è costituito dal famoso chip FT232RL, estremamente potente e versatile; nei modelli successivi è stato sostituito con i micro ATMEL della famiglia **XXu2**, programmati per svolgere tale funzione.

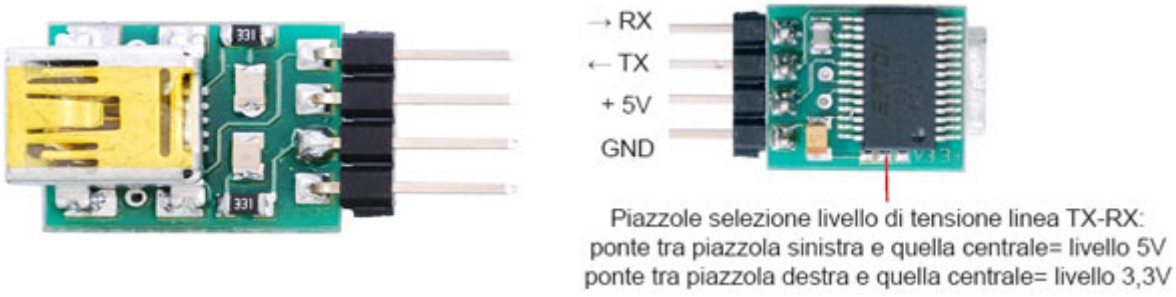
Un tale tipo di convertitore con Arduino non serve praticamente a nulla, se non nel malaugurato caso dovesse rompersi quello integrato; può diventare invece estremamente utile quando si realizzano circuiti in stand-alone con micro che devono mantenere il bootloader a bordo, oppure con quella serie di prodotti commerciali che sono predisposti per la programmazione seriale. E' quindi FONDAMENTALE che il micro da programmare possieda il bootloader, in caso contrario non sarà possibile programmarlo via seriale!!

Abbiamo già visto come sia possibile usare lo stesso Arduino come programmatore seriale, quindi non riprenderemo quel tipo di tecnica, vedremo invece una serie di modelli commerciali già pronti e due possibili applicazioni DIY (Do It Yourself, il nostro "fai da te").

p. 11 – I modelli commerciali

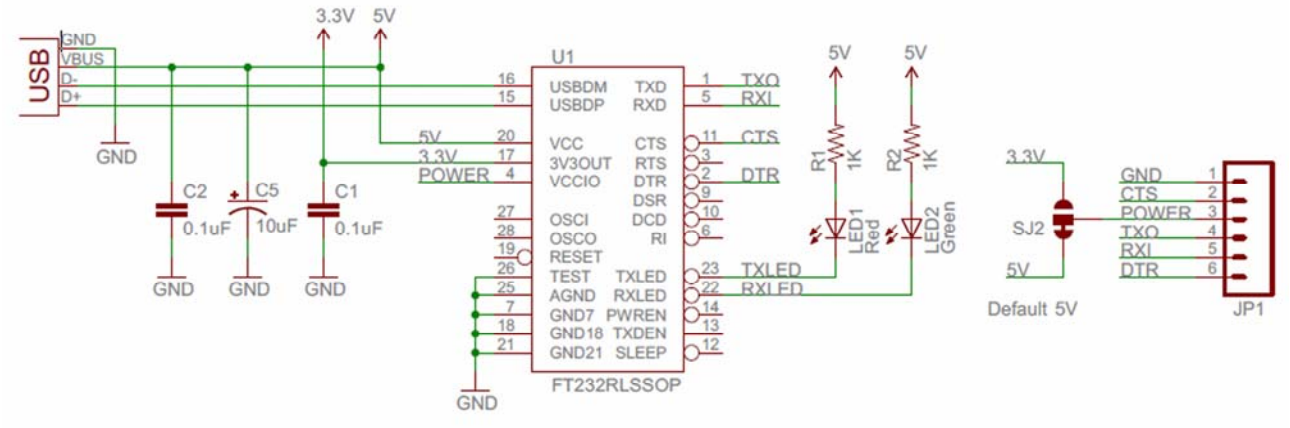
Iniziamo con una panoramica di prodotti, alcuni noti, altri meno, ma che comunque ben rappresentano ciò che si trova sul mercato:

Un convertitore USB→Seriale INUTILE per noi!



La didascalia in questo caso parla chiaro! Questo convertitore non va bene per i nostri usi, infatti, come potete vedere nell'immagine di destra, manca il segnale DTR (o RTS), indispensabile per programmare correttamente il nostro microcontrollore stand-alone.

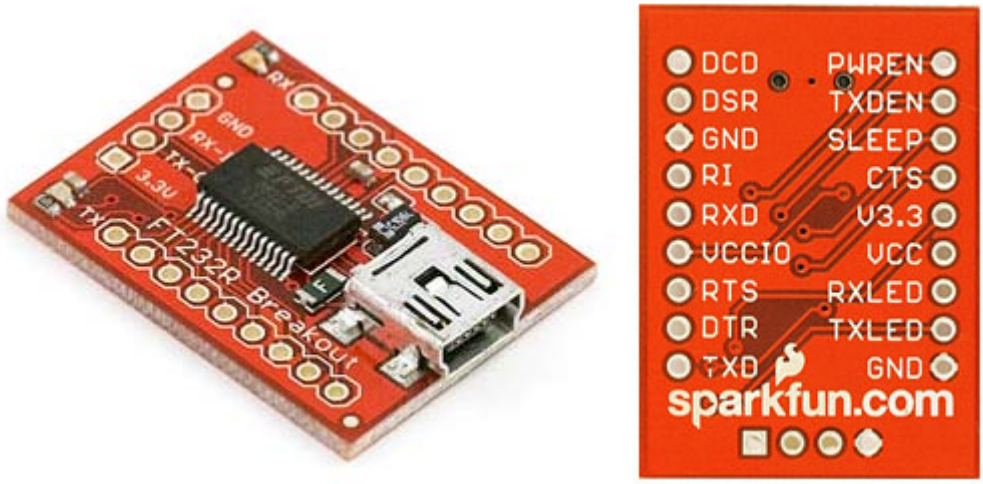
Un buon convertitore USB→Seriale con cambio tensione

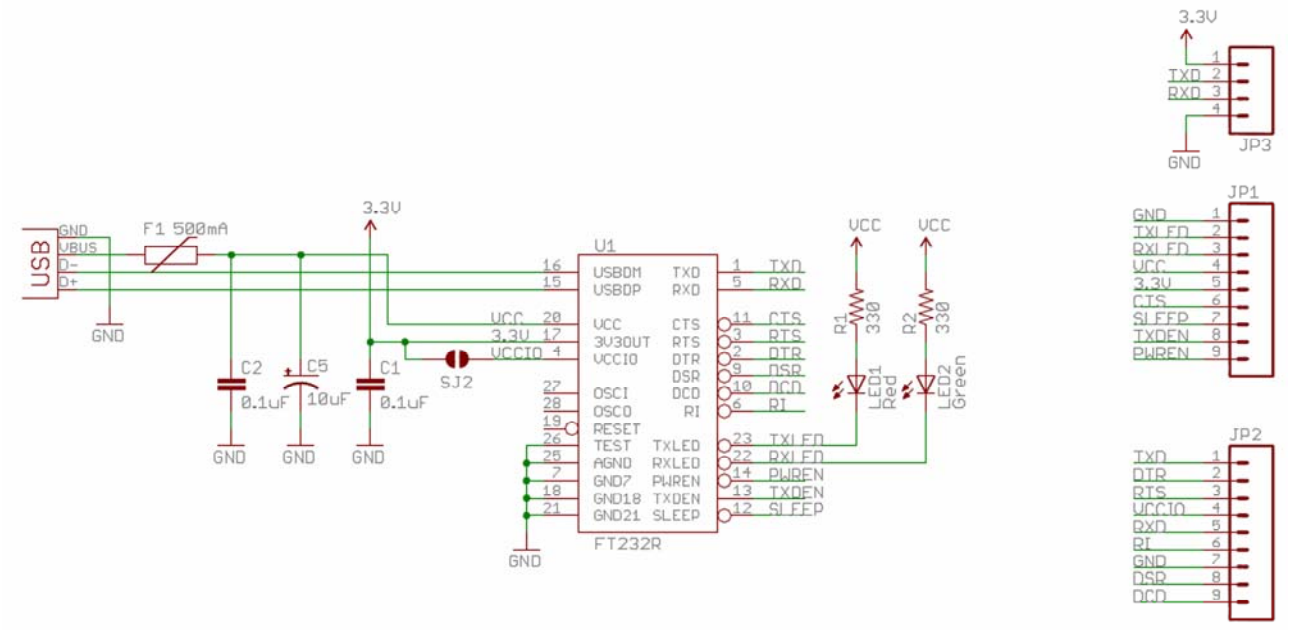


Questo bel prodotto della Sparkfun è basato sull'FT232RL, come il precedente, ma in questo caso sono stati portati sui pin tutti i segnali necessari: DTR, RXI (Input), TXO (Output), 5V, GND; c'è anche un CTS che a noi non serve, quindi lo ignoreremo. L'altra buona dotazione consiste nella possibilità di settare questo strumento per poter lavorare a 3,3V, indispensabile nei casi in cui ci si debba interfacciare con circuito commerciali o

micro stand-alone impostati per lavorare a tale tensione. Infatti il segnale TX proveniente dal convertitore potrebbe danneggiare irrimediabilmente il micro da programmare, se quest'ultimo fosse alimentato a 3,3V. VI spiego brevemente il funzionamento di questa interessante possibilità, basta seguire lo schema elettrico. La porta USB fornisce in ogni caso 5V che arrivano al pin 20 (VCC) dell'FT232RL; all'interno di questi integrato c'è un regolatore che abbassa la tensione a 3,3V restituendola sul pin 17 (3V3OUT); queste due tensioni sono applicate ai due capi del jumper a saldareSJ2 che si trova sul lato inferiore della scheda, normalmente settato sui 5V. Il centrale di tale jumper rientra nel chip dal pin 4 (VCCIO) e quindi stabilisce il livello dei segnali seriali; nel caso in cui il circuito dovesse lavorare a 3,3V basterà interrompere il collegamento 5V-VCCIO e mettere una goccia di stagno tra 3,3V e VCCIO.

Un convertitore USB→Seriale professionale con cambio tensione





Quest'ultimo prodotto si chiama breakout board ed il significato di tale termine, in uso per qualsiasi tipo di integrato, è che la scheda su cui è montato porta sugli header esterni tutti i segnali I/O ed alimentazioni dell'integrato. Infatti questa scheda, sempre della Sparkfun, non è altro che la stessa della precedente con in aggiunta tutti i segnali che è in grado di gestire l'FT232RL, che nella precedente erano stati ignorati. Anche in questo caso esiste la possibilità di settarla a 3,3V ma questa volta vedete che normalmente il pin VCCIO è scollegato esternamente (condizione di default per 5V), mentre l'eventuale goccia di stagno va sempre depositata sul jumper a saldare SJ2 nel caso in cui si voglia lavorare con segnali a 3,3V. Vi ho parlato di quest'ultima versione non solo per farvi la panoramica completa dei Convertitori su scheda, ma anche per accennarvi al fatto che esiste da tempo una tecnica di auto-programmazione del bootloader di Arduino, denominata BitBang. Ho scritto un bell'articolo su questa tecnica, in pubblicazione sul numero di Settembre 2012 di Elettronica In, e proprio in occasione dello studio di tale tecnica, ho verificato che servono altri tipi di segnali, comunque generati dall'FT232RL ma purtroppo non dagli ATMEL XXu2; tali segnali sono: CTS, DSR, DCD, RI e, al solito, l'alimentazione. Nel caso di Arduino 2009 la tecnica si esegue con grande facilità proprio per la presenza di un connettore che rende disponibili questi segnali; nel caso degli altri Arduino la tecnica è applicabile sfruttando un convertitore professionale come questo, che dispone appunto di tali segnali.

Vi ricordo che per usare una qualsiasi di queste schede, ma anche altri tipi in generale, bisogna disporre del driver del chip; gli Utenti di Arduino lo hanno come driver di

installazione della board 2009, comunque è facilmente reperibile presso i fornitori delle schede stesse.

Per questo primo paragrafo di questo capitolo mi è d'obbligo ringraziare **Futura Elettronica** dal cui sito Internet ho prelevato le immagini che avete appena visto.

p. 12 – Il cavo Nokia CA-42

Ed ora vediamo ulteriori informazioni per coloro i quali amano lavorare di saldatore e fasi le cose in casa a tutti i costi, magari andando contro il proprio interesse economico, ma si sa, il divertimento non ha prezzo!

Partendo da un cavetto **NOKIA tipo CA-42** (NON altri, non posso fornire alcuna garanzia), che si usava nei vecchi telefoni NOKIA, per dotarli di una connessione USB, è possibile realizzare un buon Convertitore USB→Seriale standard. Infatti tale cavetto, all'interno dello spinotto che si collega alla porta USB del PC, ha un microcircuito, basato sull'integrato **PL-2303**, che opera in modo molto simile all'FT232RL.

Io l'ho fatto, ma davvero non lo rifarei, a meno di non trovare un cavo NOKIA ORIGINALE, purtroppo mi sono imbattuto in un prodotto cinese. La differenza? quello originale si apre con grande facilità, avendo un involucro ad incastro, il tipo cinese invece era racchiuso in una massa di plastica fusa e poi solidificata che per fortuna alla fine è venuta via senza danneggiare nulla!

La necessità di smontare tutto è quella di localizzare le connessioni TX, RX, GND, 5V e poi di recuperare il segnale DTR (pin 2 del PL2303), che nella connessione NOKIA è sostituito da un altro segnale; bisogna quindi spostare il collegamento di un filo dal segnale originale al segnale DTR e localizzare tutti gli altri. Lo spinotto lato Nokia ovviamente va eliminato, tranciando di netto il cavo, e poi sostituito con un header maschio o femmina, secondo il tipo di collegamento che dovete realizzare.

L'ultima manovra da fare per chiudere l'operazione è quella di saldare un condensatore da 100nF tra il terminale del filo DTR ed il pin relativo dell'header finale. Non dobbiamo infatti dimenticare che bisogna trasformare il segnale DTR in un impulso di RESET per il microcontrollore e tale operazione è svolta appunto dal condensatore in serie al segnale DTR; peraltro la stessa capacità è usata anche sulle board Arduino, con identica funzione.

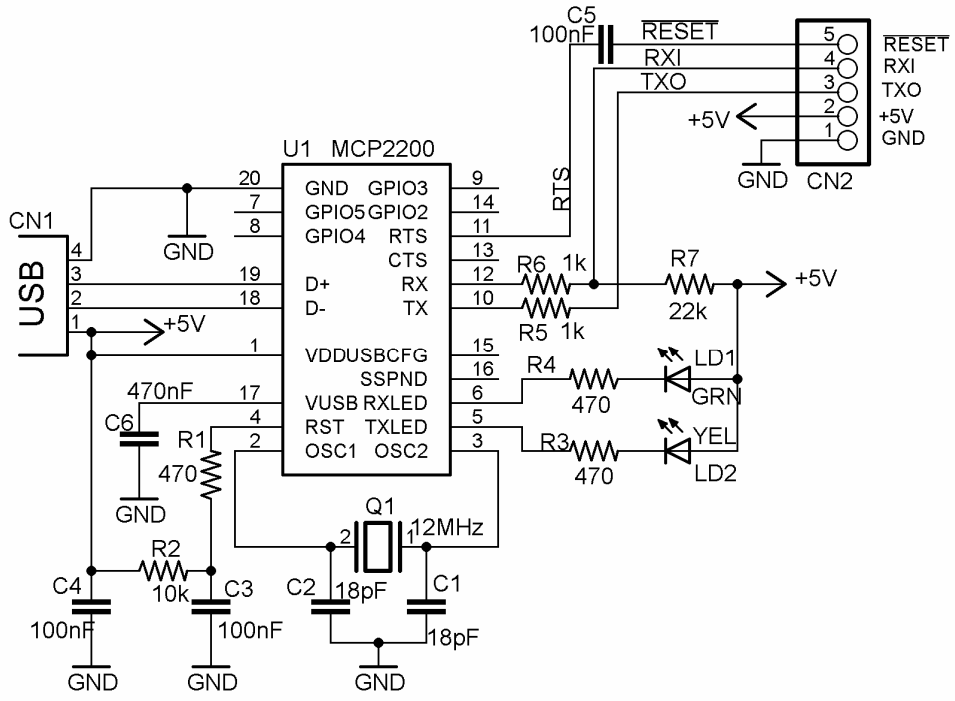
Insomma non è un lavoro da principianti, bisogna operare (è proprio il caso di dirlo) col data-sheet davanti, l'integrato è in package SMD SOIC, quindi piuttosto difficile da saldare e non sempre il segnale DTR è portato su una piazzola (come nel caso del cavo originale) che rende tutto più semplice. Per questa ragione e per il fatto che questi cavetti si trovano in vari formati, non sto a spiegare tutti i passaggi e nemmeno metto foto, però se qualcuno volesse cimentarsi sono a disposizione per le necessarie informazioni.

p. 13 – Il micro PIC MCP-2200

Sempre nelle varie sperimentazioni, grazie ad una dritta[24], ho imparato ad usare l'integrato della Microchip **MCP2200**, che è un PIC programmato per l'uso di Convertitore USB→Seriale, più o meno un equivalente dell'ATmega8u2, infatti anch'esso fornisce i soli segnali standard della comunicazione seriale, ma con una differenza: al post del segnale DTR usa il segnale RTS. Ora questi due segnali sono molto simili; in passato l'Arduino 2009 li prevedeva entrambi, poi la scelta è caduta sul DTR. L'unica differenza tangibile è che l'apertura del Serial Monitor NON provoca il RESET automatico di Arduino, ma per molti questo è un bene e non un male. Invece come programmatore seriale è perfetto! Io l'ho usato con successo sul mio Programmatore HV, preferendolo all'FT232RL per due validissime ragioni: costa molto meno ed è in formato SMD SOIC, quindi saldabile molto più facilmente.

Se andate sul sito di Microchip potete scaricare il firmware, il driver ed uno schema applicativo, un po' striminzito, per la verità, ma funzionante. Ho apportato alcune modifiche che ne hanno migliorato il funzionamento, come dimostrano i miei molti test durante la progettazione del Programmatore HV.

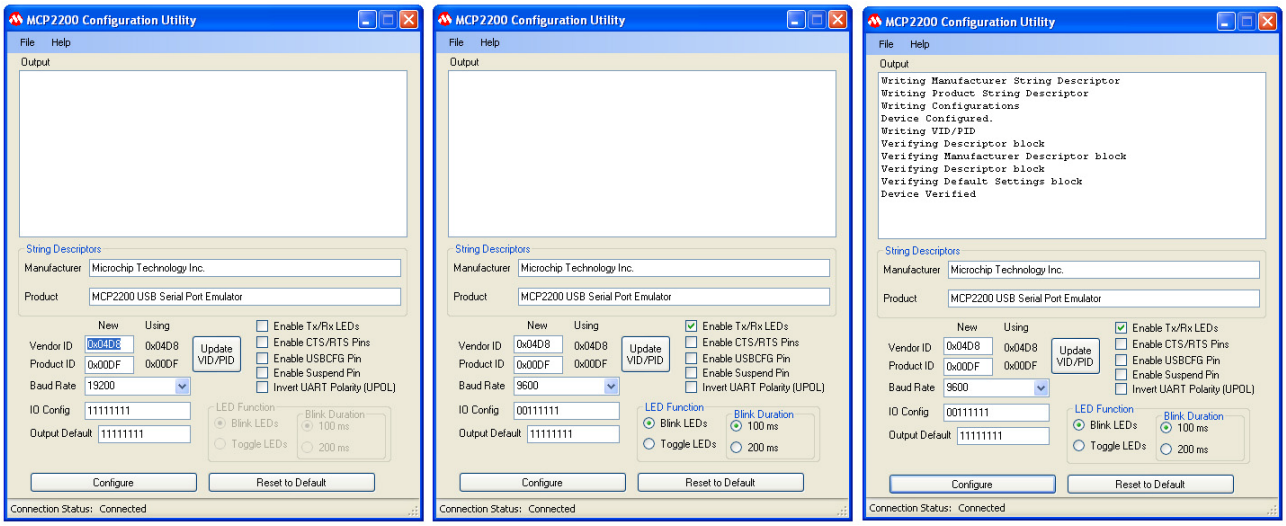
Vi riporto lo schema elettrico della mia versione, per chi volesse utilizzare questo integrato; purtroppo non ho disegnato un PCB, in quanto l'ho integrato nel Master del mio progetto, comunque sono pochissimo componenti.



[24] ringrazio il "solito" Astrobeed per avermi "presentato" questo chip.

L'integrato non è programmato di fabbrica quindi l'operazione dovete farla Voi, seguendo le prossime semplici istruzioni.

Il primo passaggio da fare è eseguire il file “MCP2200 Configuration Utility Setup” che si trova nella cartella “Configuration Utility”; questa operazione avrà il duplice scopo di installare l’utility di configurazione ed i driver necessari per la gestione della nuova seriale; appena finita l’installazione collegate la scheda ad una presa USB del PC, mediante un normale cavo USB. La scheda sarà subito riconosciuta come periferica seriale e vi sarà chiesto di fornire il driver: dovrete semplicemente confermare l’installazione automatica, ma se questa procedura non andasse a buon fine allora fornite il percorso della cartella “MCP2200 Win INF” e confermate le successive richieste, dopo alcuni istanti la nuova porta seriale sarà installata tra le periferiche del vostro PC, come “USB serial port (COMxx)”. A questo punto nei programmi troverete “MCP2200 Configuration Utility”, eseguitelo e Vi apparirà la maschera visibile a sinistra nella figura.



Dovete solo effettuare le due semplici modifiche visibili nell’immagine centrale e cioè: impostare il Baud Rate a 9600 e attivare il campo “Enable Tx/Rx LEDs”, quindi cliccate sul pulsante “Configure”, i messaggi visibili nell’immagine a destra vi indicheranno che tutto è andato a buon fine, a questo punto potete chiudere la maschera. Vi sconsiglio decisamente di fare sperimentazione con questa utility, alcune combinazioni bloccano il chip e posso assicurarvi che è un’impresa riuscire a sbloccarlo; limitatevi a quanto vi ho consigliato non avrete alcun problema.

p. 14 – Collegare un Convertitore USB→Seriale ad un micro da programmare

A questo punto non resta che spiegare come si collega uno dei qualsiasi Convertitori, commerciali o autocostruiti, visti finora, al microcontrollore da programmare.

L'operazione è estremamente semplice:

- Il segnale TX (TXO) del Convertitore va al pin RX del micro (il pin 2 nel caso dell'ATmega328P).
- Il segnale RX (RXI) del Convertitore va al pin TX del micro (il pin 3 nel caso dell'ATmega328P).
- Il segnale DTR (o RTS) va al pin RESET del micro (il pin 1 nel caso dell'ATmega328P) ma è FONDAMENTALE che i due segnali siano separati da un condensatore da 100nF. Come già detto bisogna trasformare il segnale DTR in un impulso di RESET per il microcontrollore e tale operazione è svolta appunto dal condensatore in serie al segnale DTR o RTS; peraltro la stessa capacità è usata anche sulle board Arduino, con identica funzione. In alcuni casi le schede incorporano tale condensatore, ma sono rari. P.es. nello schema dell'MCP-2200 io l'ho previsto, infatti sull'header di uscita trovate il segnale RESET, non l'RTS.
- 5V e GND vanno collegati ai rispettivi poli di alimentazione del vostro circuito.

Ora potete collegare il Convertitore alla porta USB (do' per scontato che abbiate già installato e fatto riconoscere i driver!), aprite l'IDE, selezionate come board l'Arduino il cui bootloader è caricato nel micro da programmare; selezionate la COMx con cui è stato riconosciuto ed installato il vostro convertitore; caricate uno sketch e fate l'upload, vedrete che dopo pochi secondi lo sketch sarà stato regolarmente inserito nel micro.

Conclusione

Cari amici, anche questa volta siamo giunti alla fine di questa Guida, alla sua quarta versione; avete ormai compreso come in realtà non ci sia nulla di difficile nel caricare il bootloader o anche il solo sketch in un chip vergine o già usato, con bootloader o senza, visto che anche nei casi in cui si verificano errori il rimedio è piuttosto banale e, soprattutto, non invasivo; terminata l'operazione, scollegati fili e componenti, e rimessi i chip al loro posto, ogni board ritornerà esattamente come "nuova"!

L'altra cosa importante è aver compreso che le board Arduino, almeno per quanto riguarda le 2009 e le UNO, e tutte le varie compatibili XXXX(u)ino che si trovano in commercio, a parte le differenze riguardanti i connettori ed in alcuni casi anche le differenze circuitali, sono pressoché TOTALMENTE intercambiabili tra loro, dal punto di vista software, basta montarci un chip con il bootloader desiderato. Una importante conseguenza di ciò? Si è visto che spesso i micro stand-alone con bootloader UNO tendono a bloccarsi (anche se con l'OPTIBOOT presente nella versione IDE 1.0.1 il problema sembra risolto), cosa che non avviene mai con quelli contenenti il bootloader 2009. Chi dispone solo di un Arduino UNO fino ad oggi è stato costretto a comprare chip pre-caricati per questo scopo; invece da oggi potrà prepararseli da soli i chip con bootloader 2009; in realtà, come abbiamo visto, nella stragrande maggioranza dei casi, si può tranquillamente fare a meno del bootloader senza perdere alcuna funzionalità!

Infine abbiamo ben visto come è possibile realizzare piccoli circuiti, quando non necessitano di molta velocità o di molta memoria, riducendo notevolmente i consumi.

P.S. Naturalmente, come sempre invito tutti ad inviare suggerimenti, correzioni ed implementazioni, per il bene della Comunità Arduino; potete tranquillamente scrivermi in forma privata per qualsiasi chiarimento; io continuerò a studiare nuove tecniche e se raggiungerò altri risultati soddisfacenti e, soprattutto, praticabili da tutti, certamente li inserirò in una successiva versione di questa Guida che spero starete già apprezzando e, naturalmente, usando!