ARI Civitavecchia

Arduino

Presentazione corso/laboratorio

ARI Civitavecchia

I microcontrollori

Il microcontrollore o microcontroller o MCU (MicroController Unit) è un dispositivo elettronico integrato su singolo chip, nato come evoluzione alternativa al Microprocessore ed utilizzato generalmente in sistemi embedded ovvero per applicazioni specifiche (special purpose) di controllo digitale.

Il microcontrollore è progettato per interagire direttamente con il mondo esterno tramite un programma residente nella propria memoria interna e mediante l'uso di pin specializzati o configurabili dal programmatore.

L'ampia gamma di funzioni di comando e controllo disponibili, sia analogiche che digitali, integrate sullo stesso chip, permette l'impiego delle MCU in sostituzione di schede elettroniche cablate tradizionali ben più complesse e costose.

ARI Civitavecchia

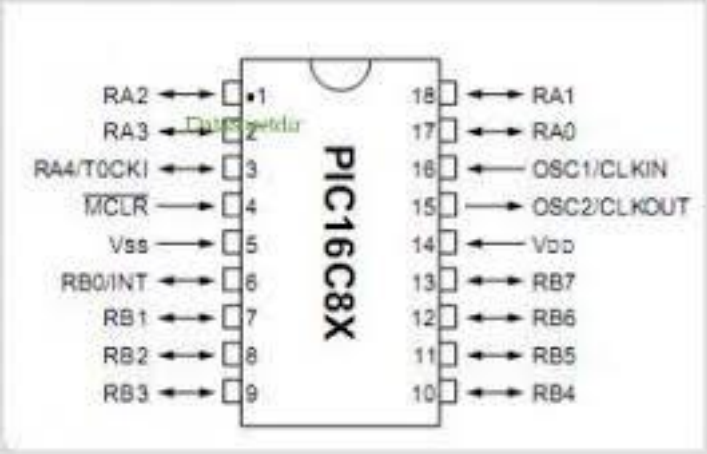
Il primo computer on-chip ottimizzato per applicazioni di controllo è stato il modello 8048 di Intel, rilasciato nel 1975, con RAM e ROM sullo stesso chip. Questo componente è stato utilizzato in più di un miliardo di tastiere per PC e numerose altre applicazioni.

ARI Civitavecchia

Nei primi anni di sviluppo del microcontrollore, la maggior parte dei modelli era commercializzata in due varianti. La più economica era dotata di memoria di programma programmata in fabbrica (ROM) su specifiche del cliente oppure programmabile dall'utente una sola volta (OTP, One Time Programming). La seconda, più costosa, aveva la memoria di programma cancellabile EPROM mediante esposizione a luce ultravioletta del chip tramite la finestrella trasparente che lo sovrastava.

ARI Civitavecchia

Nel 1993, Microchip ha introdotto il modello di MCU PIC16C84, caratterizzato da memoria programma in EEPROM, ovvero cancellabile elettricamente, che permetteva sia lo sviluppo veloce del prototipo, sia la modifica del Firmware a circuito montato (In-System Programming). La semplificazione del contenitore (package), senza finestrella in quarzo, ha contribuito a ridurre il costo finale del componente.



ARI Civitavecchia

Nello stesso anno, Atmel ha rilasciato il primo MCU che utilizzava una memoria di tipo flash, ancora più semplice e veloce da programmare/modificare, più compatta e con un ciclo di vita (cancellazioni) molto più elevato. Questo ha segnato l'inizio del massiccio utilizzo del microcontrollore nelle più disparate applicazioni.

ARI Civitavecchia

Differenze tra microprocessore e microcontrollore

Quando nell'aprile del 1972 Intel ha introdotto sul mercato il primo processore a 8 bit - lo 8008 - esso consisteva di una ALU (Arithmetic & Logical Unit, ALU) per svolgere operazioni logiche e matematiche e di una unità di elaborazione elementare (Control Processing Unit, CPU) per controllare il flusso di dati e indirizzi tra la ALU e i circuiti esterni di supporto.

Successivamente, l'architettura interna del microprocessore (in seguito spesso chiamato semplicemente CPU) si è evoluta velocemente con lo sviluppo di blocchi interni via via più complessi (maggior numero di bit elaborati e movimentati) e con capacità e velocità di calcolo crescenti in modo esponenziale. **Per quanto potente, il microprocessore integra sul chip solo la logica di elaborazione mentre richiede sempre delle unità esterne - memorie, gestori di segnali e dispositivi periferici per poter scambiare informazioni e interagire con l'esterno.**

ARI Civitavecchia

Il microcontrollore è invece un sistema completo, che integra in uno stesso chip il processore, la memoria permanente, la memoria volatile e i canali (pin) di I/O, oltre ad eventuali altri blocchi specializzati.

A differenza dei microprocessori classici, adatti per un uso generale (general purpose), il microcontrollore è inoltre progettato per ottenere la massima autosufficienza funzionale ed ottimizzare il rapporto prezzo-prestazioni in uno specifico campo di applicazioni.

ARI Civitavecchia

Anche l'esecuzione dei programmi applicativi si appoggia su un'architettura hardware diversa da quella tipicamente usata per i microprocessori. **Mentre questi ultimi, soprattutto quando usati per computer e personal computer, eseguono i programmi applicativi appoggiandosi a dispositivi di memoria di massa o a memoria volatile, per i microcontrollori il programma applicativo è tipicamente memorizzato su un dispositivo di memoria ROM (come per esempio una EPROM) ed è quindi in realtà un firmware.**

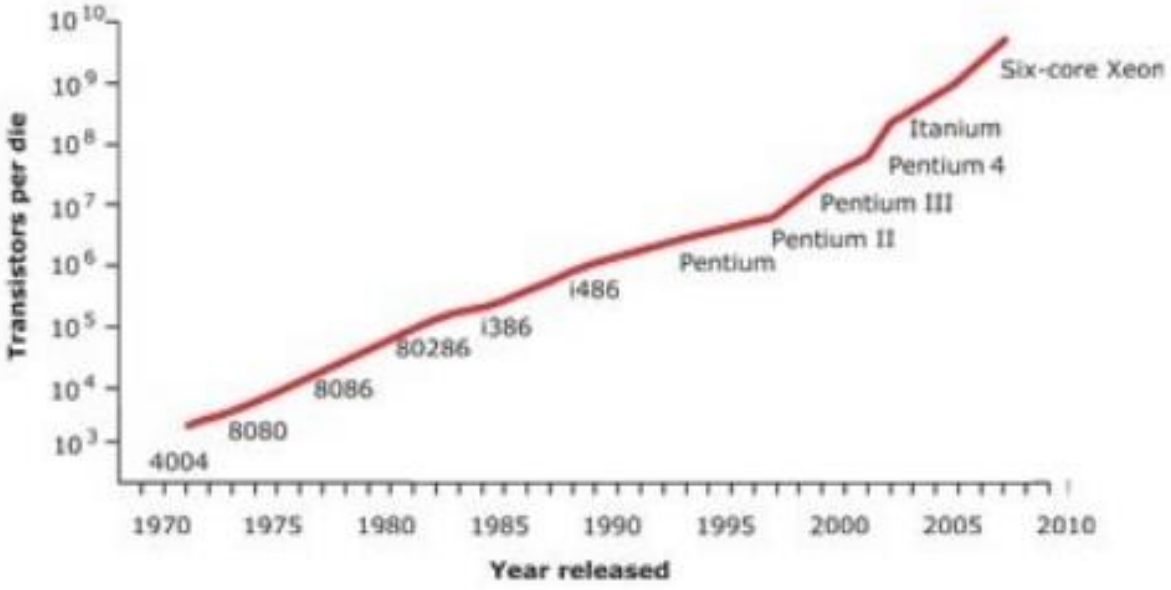
In generale, essendo concepiti per applicazioni specifiche e per l'uso in condizioni ambientali particolari che impongono limitazioni sia in termini di consumi che in termini di dissipazione di potenza, i microcontrollori presentano caratteristiche meno spinte rispetto ai microprocessori ma risultano più economici di questi ultimi sia per l'integrazione in un solo chip di una serie di componenti sia per fattori di economia di scala legati ai volumi di produzione molto più elevati.

ARI Civitavecchia

Tabella 1: confronto microprocessore vs. microcontrollore		
Caratteristica	Microcontrollore	Microprocessore
Velocità massima di clock	200Mhz	4GHz
Capacità elaborativa massima in MegaFLOPS	200	5000
Potenza minima dissipata in Watt (in stato di elaborazione)	0.001	50
Prezzo minimo per singola unità in USD	0.5	50
Numero di pezzi venduti annualmente (in milioni)	11,000	1,000

ARI Civitavecchia

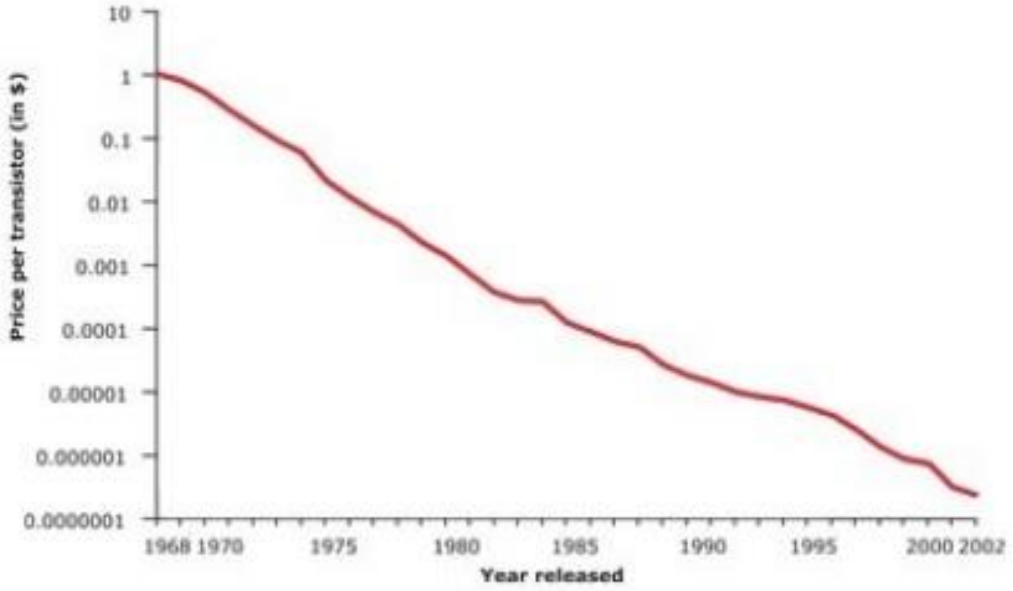
Moore's Law



the graph we are used to seeing

ARI Civitavecchia

Moore's Law, also



per-transistor cost of CPUs

ARI Civitavecchia

Information Processing as Material

1989: $900

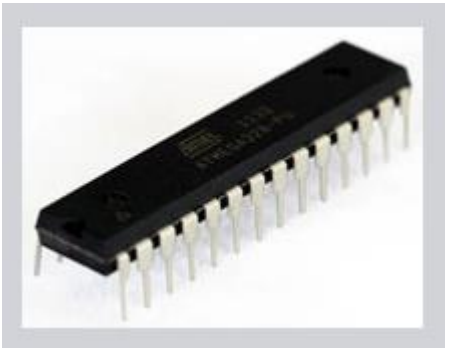
33 MHz, 20 MIPs

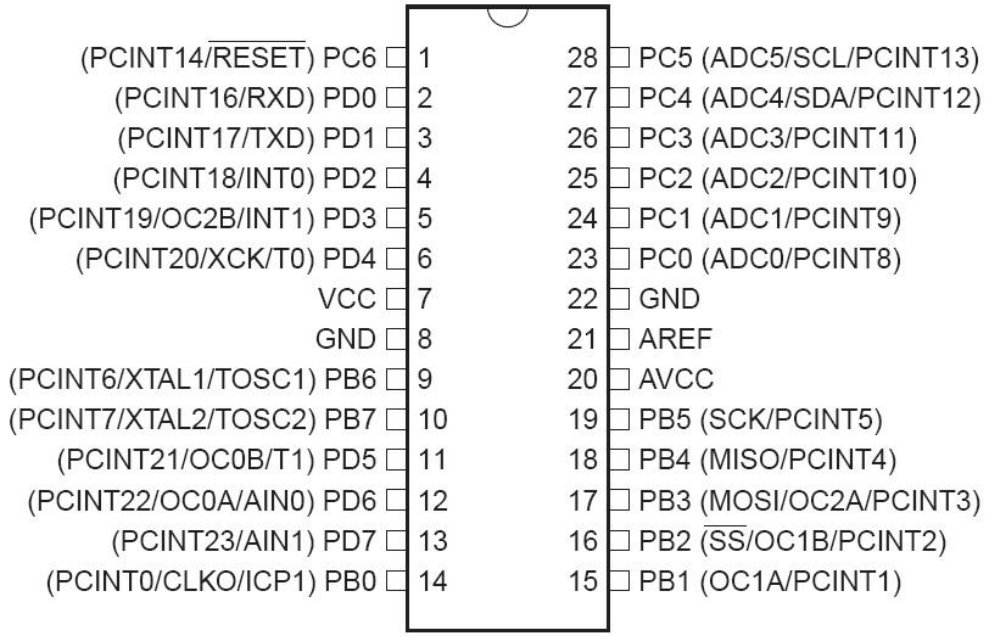
2009: $0.47

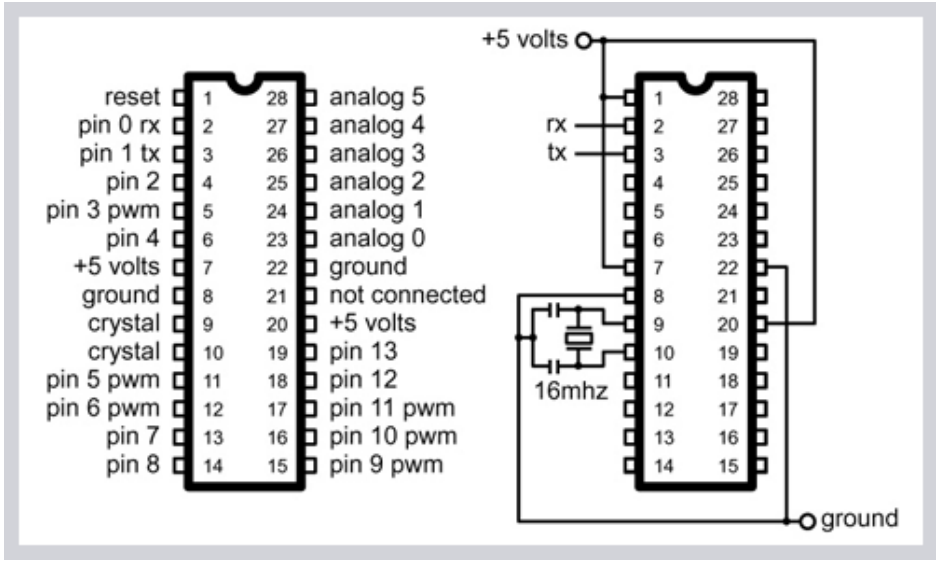
20 MHz, 20 MIPs

We are headed towards computation being a nil cost add-on

ARI Civitavecchia







Arduino vuol dire 3 cose

CC-BY-SA

ARI Civitavecchia

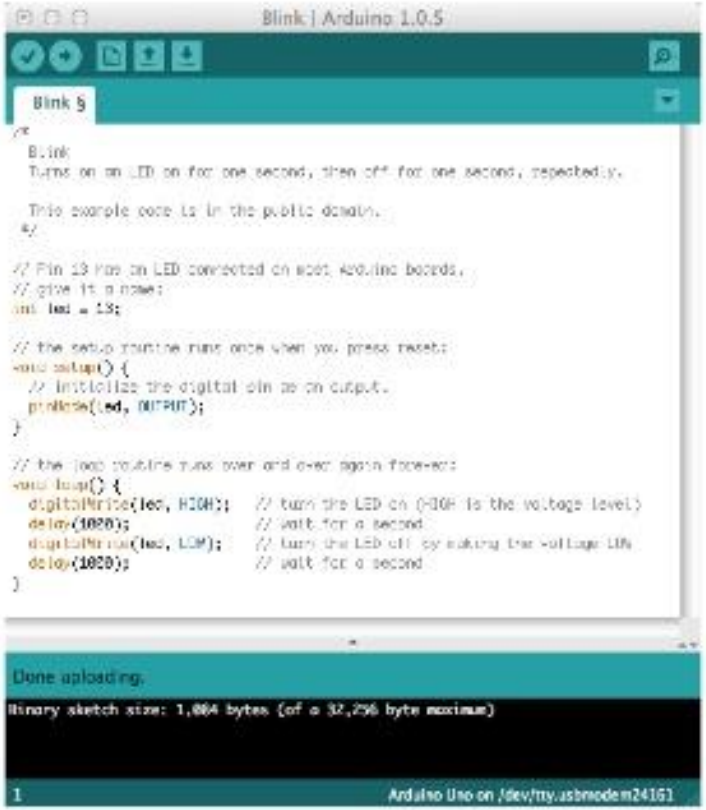
Arduino vuol dire 3 cose 1/3

Un oggetto fisico

CC-BY-SA

ARI Civitavecchia

Un ambiente di sviluppo (di programmazione)



CC-BY-SA

ARI Civitavecchia

Arduino vuol dire 3 cose 3/3

una comunità ed una filosofia di sviluppo

CC-BY-SA

ARI Civitavecchia

Caratteristiche tecniche

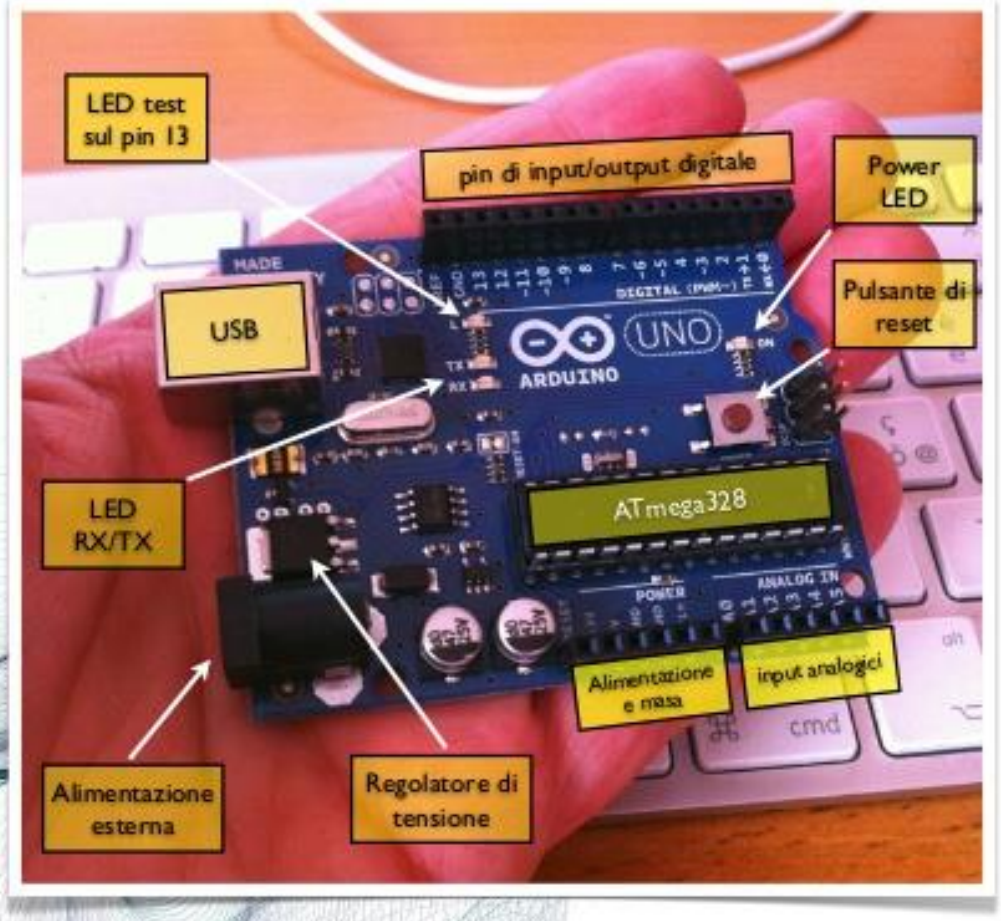
- Microcontroller: ATmega328
- Tensione di lavoro: 5V
- Tensione di ingresso (raccomandata): 7-12V
- Tensione di ingresso (limiti): 6-20V
- Pin digitalio I/O: 14 (di cui 6 forniscono un'uscita PWM)
- Pin analogici: 6
- Corrente Continua per i pin I/O: 40 mA
- Corrente continua per l'uscita a 3.3V: 50 mA
- Flash Memory: 32 KB (ATmega328) di cui 0.5 KB usata per bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Velocità del clock: 16 MHz

CC-BY-SA

ARI Civitavecchia

Hardware Arduino

Riconosciamo le parti fondamentali della scheda



ARI Civitavecchia

Arduino Boards

Explosion of different form-factors

ARI Civitavecchia

FFino a oggi, sono state commercializzate 16 versioni dell'hardware Arduino:1

.Serial Arduino, programmata con una porta seriale DB9. Fa uso del microcontroller ATmega8;

2.Arduino Extreme, con interfaccia di programmazione USB, facente uso del chip ATmega8;

3.Arduino Mini, una versione in miniatura facente uso di un ATmega168 a montaggio superficiale;

4.Arduino Nano, una versione ancor più piccola della Mini, utilizzante lo stesso controller ATmega168 SMD e alimentata tramite USB;

5.LilyPad Arduino, un progetto minimalista (scheda circolare dal diametro di 50 mm, per circa 0.8 mm di spessore), per applicazione su indumenti, con lo stesso ATmega168 in versione SMD;

6.Arduino NG, con un'interfaccia USB per programmare e usare un ATmega8;

7.Arduino NG plus, con interfaccia di programmazione USB, con un ATmega168;

8.Arduino BT, con interfaccia di programmazione Bluetooth e con un ATmega168;

9.Arduino Diecimila, con interfaccia di programmazione USB e con un ATmega168 in un package DIL28;

10.Arduino Duemilanove, facente uso del chip Atmega168 (o Atmega328 nelle versioni più recenti) e alimentata in corrente continua tramite USB, con commutazione automatica tra le sorgenti di alimentazione;

11.Arduino Mega, che fa uso di un ATmega1280 a montaggio superficiale per I/O e memoria addizionale;

12.Arduino Uno, evoluzione della Duemilanove con un differente chip, programmabile e più economico, dedicato alla conversione USB-seriale;

13.Arduino Mega2560, che fa uso di un ATmega2560 (anch'esso a montaggio superficiale) ed è un'evoluzione dell'Arduino Mega;

14.Arduino Due, che fa uso di un Atmel SAM3X8E ARM Cortex-M3 CPU;

15.Arduino Zero Pro, dotata di un microcontrollore Atmel SAMD21 basato su core ARM Cortex M0+ a 32 bit.

16.Arduino Yún, che fa uso di un ATmega32u4 e del processore Atheros AR9331, quest'ultimo compatibile con una distribuzione Linux basata su OpenWrt e chiamata Linino OS.

ARI Civitavecchia

Arduino Shields

"shields" = stackable add-on boards

for sensors, actuators, communication, etc.

Arduino Shield List

Pin usage details for 317 shields from 125 makers, and counting!

Search: blinkm GO

> Home

- 4D Systems (4)
- Adafruit Industries (9)
- AeroQuad (2)
- Andre Concalves (2)
- antrax Datentechnik (1)
- Applied Platonics (1)
- ArduCapSense (1)
- Arduino (3)
- Argent Data Systems (1)
- AsyncLabs (3)
- Batsocks (2)
- Ben Combee (1)
- Bhasha Technologies (2)
- Bliptronics (2)
- Blushing Boy (1)
- Carlos Neves (1)
- Chesters Garage (1)
- Excamera Labs (1)
- Faz Jaxton (1)
- FlamingoEDA (1)
- Freetronics (12)
- Futura Elettronica (2)
- Galileo 7 (4)
- GeekOnFire (1)
- GfxHax (1)
- GinSing (1)
- Gravitech (1)
- Homeroasters (1)
- HW Kitchen (1)
- ITead Studio (8)
- Jee Labs (1)
- Jimmie Rodgers (1)
- John Liu (3)
- Krudsel (1)
- Ocean Controls (1)
- Open Electronics (1)
- PDK Solutions (1)
- Photoduino (1)
- Pololu (1)
- Practical Maker (5)
- Protuino (2)
- Ray's Hobby (1)
- Renbotics (2)
- RepRap Research Foundation (1)
- Ro-Bot-X Designs (1)
- Robot Power (1)
- RobotPirate (1)
- Rocket Scream (1)
- Rogue Robotics (1)
- Rugged Circuits (6)

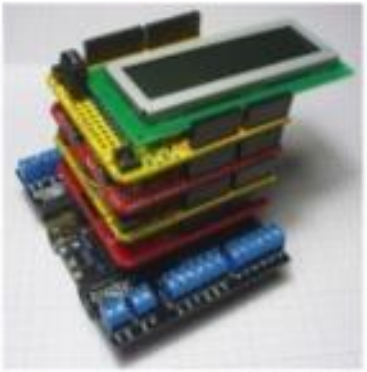


Photo: John Boxall

Tweets by @shieldlist

"317 shields from 125 makers, and counting!"

ARI Civitavecchia

Terminologia

- **sketch**
  il programma che scrivete e fate girare sulla scheda Arduino
- **pin**
  i connettori di input o output
- **digital**
  vuol dire che può assumere solo due valori: **ALTO** o **BASSO** in altro modo **ON/OFF** oppure **0** o **1**
- **analog**
  quando i valori utili che rappresentano un segnale sono continui (infiniti)

ARI Civitavecchia

Arduino UNO R3

E' il microcontrollore che impareremo ad utilizzare, puoi assimilare questa scheda ad un piccolo computer che ti consentirà di realizzare prototipi di circuiti elettronici di controllo a scopo didattico e di prototipazione. Ad essa potrai collegare dei sensori, componenti elettronici in grado di convertire una grandezza fisica in una grandezza elettrica e degli attuatori, componenti elettronici e meccanici in grado di compiere un'azione controllata da Arduino.

CC-BY-SA

ARI Civitavecchia

Installare Arduino

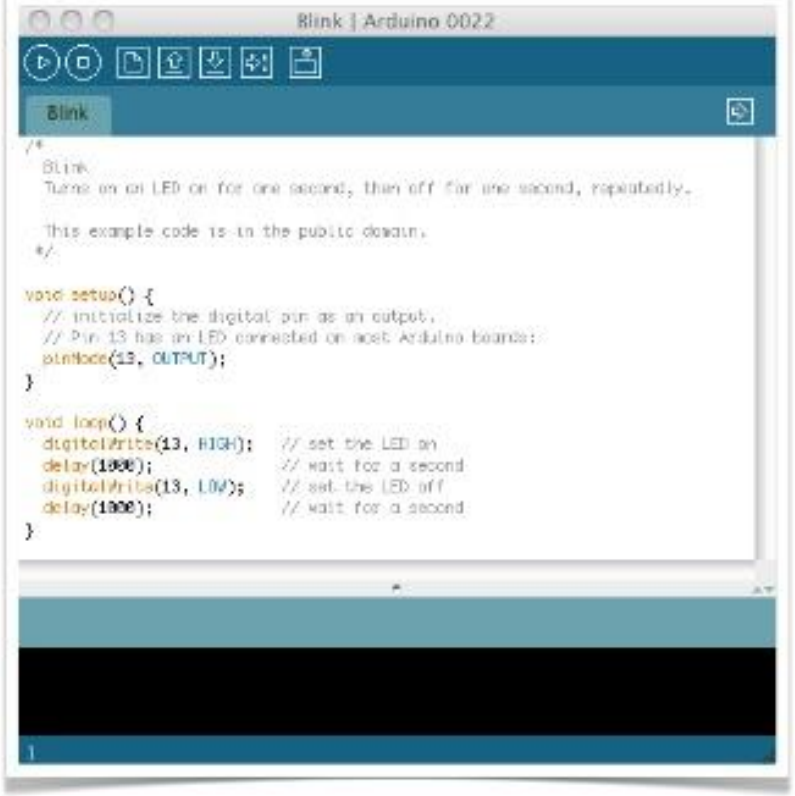
Installazione dell'IDE

1. Collegarsi al sito **arduino.cc**
2. nella sezione **Getting Started** selezionare il sistema operativo (MacOS X, Linux, Windows) su cui si desidera installare l'IDE e seguire i passi indicati.

per approfondimenti sull'installazione in ambiente MacOS X o Ubuntu consultare la sezione Arduino del sito **maffucci.it**

ARI Civitavecchia

Il software

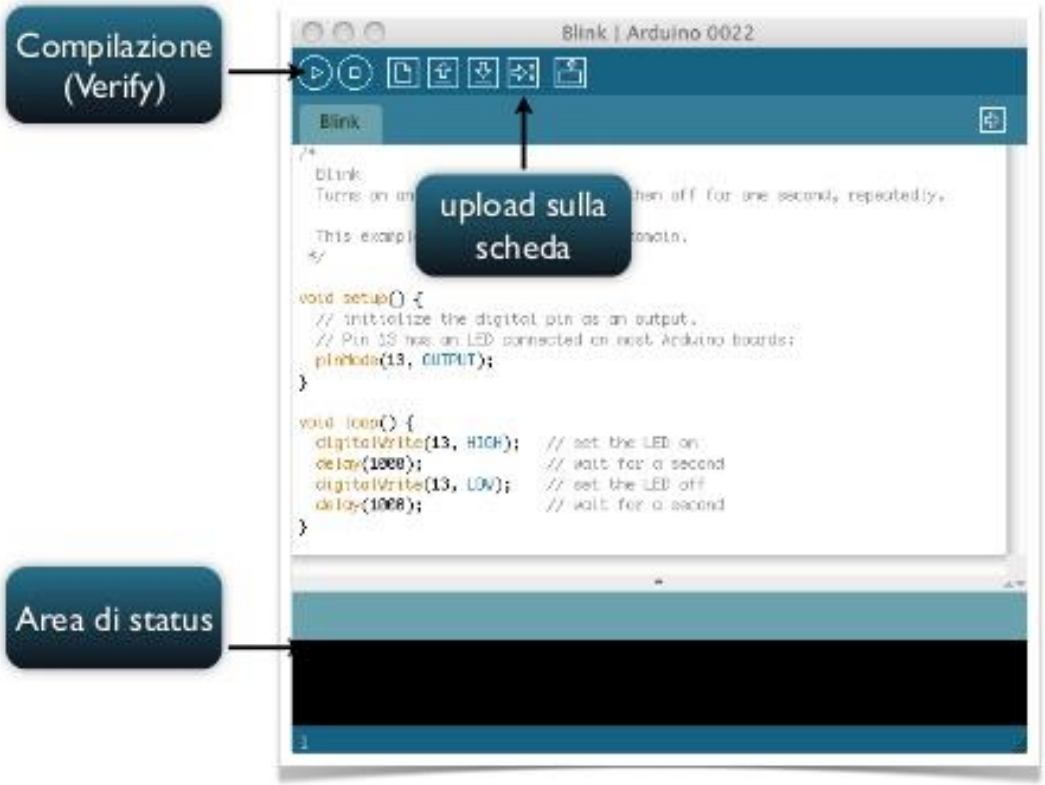


- Simile ad editor di testo;
- potete scrivere, visualizzare, verificare la sintassi;
- potete trasferire il vostro sketch sulla scheda.

ARI Civitavecchia

Installare Arduino

La finestra di programmazione



ARI Civitavecchia

Programmazione

Ciclo di sviluppo

Il ciclo di sviluppo consiste in queste 4 fasi:

ARI Civitavecchia

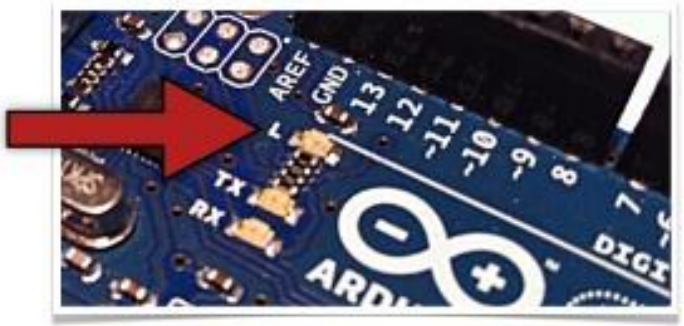
Programmazione

Scriviamo il nostro primo programma: facciamo lampeggiare un LED

Lo sketch che vi consente di far lampeggiare un led, che chiamerebo blink, è il primo programma che dovete realizzare per familiarizzare con l'ambiente di sviluppo

PARTIAMO !

Sulla scheda è presente un piccolo diodo LED indicato da una "**L**" che utilizzeremo come trasduttore per il nostro primo programma



ARI Civitavecchia

Programmazione

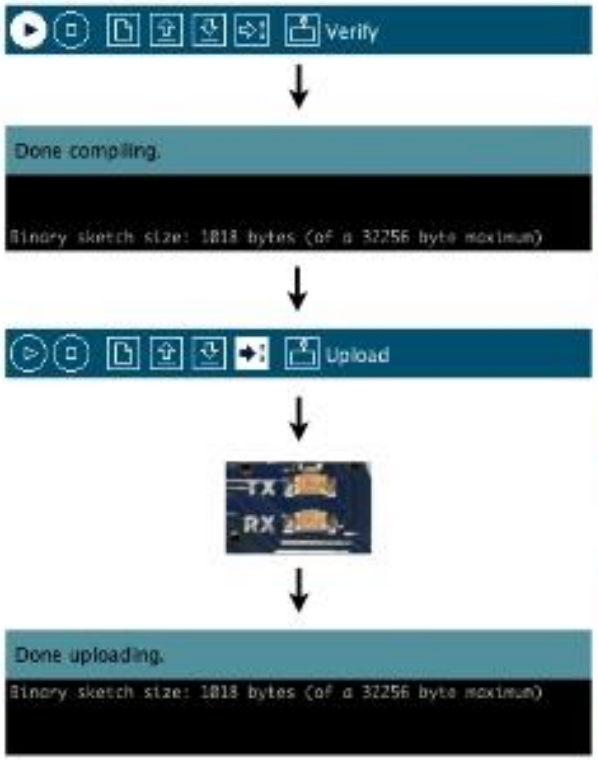
Fasi di sviluppo

```
// Esempio 01: far lampeggiare un led

#define LED 13            // LED collegato al pin digitale 13

void setup() {
  pinMode(LED, OUTPUT);     // imposta il pin digitale come output
}

void loop() {
  digitalWrite(LED, HIGH);  // accende il LED
  delay(1000);              // aspetta un secondo
  digitalWrite(LED, LOW);   // spegne il LED
  delay(1000);              // aspetta un secondo
}
```



- scrivere uno sketch
- click su "Verify" per verificare se sono presenti errori nel programma
- click su "Upload" affinchè lo sketch compilato venga trasferito dal computer ed eseguito

ARI Civitavecchia

Programmazione - analizziamo il codice

La struttura di uno sketch

1. in cima al programma troverete sempre le dichiarazione di variabili
2. `setup()` - inizializzazione dell'uso dei piedini (ingresso uscita) della scheda
3. `loop()` - esecuzione, che funziona ripetutamente dopo aver eseguito il `setup()`

```
// Esempio 01: far lampeggiare un led

  #define LED 13              // LED collegato al pin digitale 13

void setup() {
  pinMode(LED, OUTPUT);       // imposta il pin digitale come output
}

void loop() {
  digitalWrite(LED, HIGH);    // accende il LED
  delay(1000);                // aspetta un secondo
  digitalWrite(LED, LOW);     // spegne il LED
  delay(1000);                // aspetta un secondo
}
```

ARI Civitavecchia

Programmazione - analizziamo il codice

```
// Esempio 01: far lampeggiare un led

  #define LED 13              // LED collegato al pin digitale 13

void setup() {
  pinMode(LED, OUTPUT);       // imposta il pin digitale come output
}

void loop() {
  digitalWrite(LED, HIGH);    // accende il LED
  delay(1000);                // aspetta un secondo
  digitalWrite(LED, LOW);     // spegne il LED
  delay(1000);                // aspetta un secondo
}
```

Cone le due barre "**//**" indichiamo l'inizio di un commento e il testo che lo segue, in questo caso, identifica il titolo del nostro programma.

I commenti non servono per far funzionare il programma ma sono di utilità al programmatore per dettagliare ad esempio il funzionament di alcune parti.

ARI Civitavecchia

Programmazione - analizziamo il codice

Molto spesso è importante indicare subito dopo il titolo, la data dell'ultima variazione e il nome dell'autore, in modo che a distanza di tempo possiamo identificare le versioni di sviluppo ed il nome dell'ultimo programmatore che ha eseguito modifiche:

Ad esempio:

```
// Esempio 01: far lampeggiare un led
// Data ultima modifica: 16.11.2010
// Autore: Michele Maffucci
```

notate che ogni linea di commento deve essere preceduta da "**//**"

Se volete evitare di scrivere per ogni linea "**//**" potete far precedere il testo commento dai caratteri "**/***" e farlo concludere con i caratteri "***/**"

Ad esempio:

```
/* Esempio 01: far lampeggiare un led
Data ultima modifica: 16.11.2010
Autore: Michele Maffucci */
```

ARI Civitavecchia

Programmazione - analizziamo il codice

```
// Esempio 01: far lampeggiare un led

  #define LED 13              // LED collegato al pin digitale 13

void setup() {
  pinMode(LED, OUTPUT);       // imposta il pin digitale come output
}

void loop() {
  digitalWrite(LED, HIGH);    // accende il LED
  delay(1000);                // aspetta un secondo
  digitalWrite(LED, LOW);     // spegne il LED
  delay(1000);                // aspetta un secondo
}
```

`#define` consente di sostituire, in fase di compilazione, all'interno del vostro programma il valore 13 ogni volta che compare la parola LED.

Vi ricordo che la compilazione avviene quando fate click su "Verify" oppure quando effettuate l'upload sulla scheda.

ARI Civitavecchia

Programmazione - analizziamo il codice

```
// Esempio 01: far lampeggiare un led

  #define LED 13              // LED collegato al pin digitale 13

void setup() {
  pinMode(LED, OUTPUT);       // imposta il pin digitale come output
}

void loop() {
  digitalWrite(LED, HIGH);    // accende il LED
  delay(1000);                // aspetta un secondo
  digitalWrite(LED, LOW);     // spegne il LED
  delay(1000);                // aspetta un secondo
}
```

void indica ad Arduino che stiamo dichiarando una funzione di nome **setup()**, cioè un porzione di codice che esegue un'operazione specifica

```
{
```

La **parentesi graffa aperta** indica dove incomincia il codice.

ARI Civitavecchia

Programmazione - analizziamo il codice

```
// Esempio 01: far lampeggiare un led

  #define LED 13              // LED collegato al pin digitale 13

void setup() {
  pinMode(LED, OUTPUT);       // imposta il pin digitale come output
}

void loop() {
  digitalWrite(LED, HIGH);    // accende il LED
  delay(1000);                // aspetta un secondo
  digitalWrite(LED, LOW);     // spegne il LED
  delay(1000);                // aspetta un secondo
}
```

`void loop()` Indica la sezione di codice principale, il nucleo del vostro programma, che verrà ripetuto all'infinito fino a quando non si spegne la scheda.

ARI Civitavecchia

Esercizio 1
Variate il tempo di accensione e spegnimento, con tempi uguali di accensione e spegnimento (superiore ad 1 secondo).

Esercizio 2
Variate il tempo di accensione e spegnimento, con tempi diversi di accensione e spegnimento (superiore ad 1 secondo).

Esercizio 3
Provate ad eseguire il programma con tempi di accensione e spegnimento del LED con tempi di:

- 500 ms
- 250 ms
- 100 ms
- 50 ms

Verificare cosa accade al diminuire del tempo di accensione e spegnimento.

CC-BY-SA

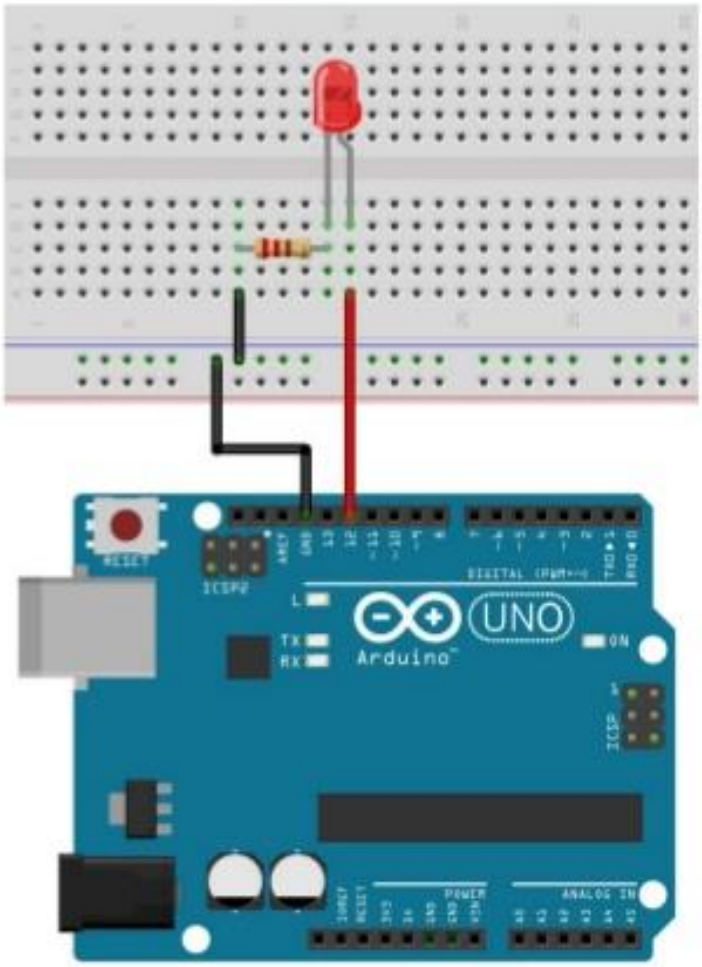
ARI Civitavecchia

esercizio | 2/4 | sketch04

Esercizio 4

Utilizzando un LED esterno (diverso dal LED L della scheda) ed una resistenza da collocare in serie al LED per limitare la corrente del circuito, far funzionare il programma Blink. Utilizzare un qualsiasi pin digitale dal 3 al 12 (nell'esempio è stato usato il pin 12) e verificatene il funzionamento con tempi di accensione e spegnimento a piacere.

R = 220 Ohm



Per la realizzazione dello schema di montaggio è stato utilizzato il software Fritzing, per maggiori informazioni seguire il link.

CC-BY-SA

ARI Civitavecchia

Corso hardware-software Arduino – parte 1

I microcontrollori

Il microcontrollore o microcontroller o MCU (MicroController Unit) è un dispositivo elettronico integrato su singolo chip, nato come evoluzione alternativa al Microprocessore ed utilizzato generalmente in sistemi embedded ovvero per applicazioni specifiche (special purpose) di controllo digitale.

Il microcontrollore è progettato per interagire direttamente con il mondo esterno tramite un programma residente nella propria memoria interna e mediante l'uso di pin specializzati o configurabili dal programmatore.

L'ampia gamma di funzioni di comando e controllo disponibili, sia analogiche che digitali, integrate sullo stesso chip, permette l'impiego delle MCU in sostituzione di schede elettroniche cablate tradizionali ben più complesse e costose.

Il primo computer on-chip ottimizzato per applicazioni di controllo è stato il modello 8048 di Intel, rilasciato nel 1975, con RAM e ROM sullo stesso chip. Questo componente è stato utilizzato in più di un miliardo di tastiere per PC e numerose altre applicazioni.

Nei primi anni di sviluppo del microcontrollore, la maggior parte dei modelli era commercializzata in due varianti. La più economica era dotata di memoria di programma programmata in fabbrica (ROM) su specifiche del cliente oppure programmabile dall'utente una sola volta (OTP, One Time Programming). La seconda, più costosa, aveva la memoria di programma cancellabile EPROM mediante esposizione a luce ultravioletta del chip tramite la finestrella trasparente che lo sovrastava.

Nel 1993, Microchip ha introdotto il modello di MCU PIC16C84, caratterizzato da memoria programma in EEPROM, ovvero cancellabile elettricamente, che permetteva sia lo sviluppo veloce del prototipo, sia la modifica del Firmware a circuito montato (In-System Programming). La semplificazione del contenitore (package), senza finestrella in quarzo, ha contribuito a ridurre il costo finale del componente.

Nello stesso anno, Atmel ha rilasciato il primo MCU che utilizzava una memoria di tipo flash, ancora più semplice e veloce da programmare/modificare, più compatta e con un ciclo di vita (cancellazioni) molto più elevato. Questo ha segnato l'inizio del massiccio utilizzo del microcontrollore nelle più disparate applicazioni.

Differenze tra microprocessore e microcontrollore

Quando nell'aprile del 1972 Intel ha introdotto sul mercato il primo processore a 8 bit - lo 8008 - esso consisteva di una ALU (Arithmetic & Logical Unit, ALU) per svolgere operazioni logiche e matematiche e di una unità di elaborazione elementare (Control Processing Unit, CPU) per controllare il flusso di dati e indirizzi tra la ALU e i circuiti esterni di supporto.

Successivamente, l'architettura interna del microprocessore (in seguito spesso chiamato semplicemente CPU) si è evoluta velocemente con lo sviluppo di blocchi interni via via più

complessi (maggior numero di bit elaborati e movimentati) e con capacità e velocità di calcolo crescenti in modo esponenziale. Per quanto potente, il microprocessore integra sul chip solo la logica di elaborazione mentre richiede sempre delle unità esterne - memorie, gestori di segnali e dispositivi periferici per poter scambiare informazioni e interagire con l'esterno.

Il microcontrollore è invece un sistema completo, che integra in uno stesso chip il processore, la memoria permanente, la memoria volatile e i canali (pin) di I/O, oltre ad eventuali altri blocchi specializzati.

A differenza dei microprocessori classici, adatti per un uso generale (general purpose), il microcontrollore è inoltre progettato per ottenere la massima autosufficienza funzionale ed ottimizzare il rapporto prezzo-prestazioni in uno specifico campo di applicazioni.

Anche l'esecuzione dei programmi applicativi si appoggia su un'architettura hardware diversa da quella tipicamente usata per i microprocessori. Mentre questi ultimi, soprattutto quando usati per computer e personal computer, eseguono i programmi applicativi appoggiandosi a dispositivi di memoria di massa o a memoria volatile, per i microcontrollori il programma applicativo è tipicamente memorizzato su un dispositivo di memoria ROM (come per esempio una EPROM) ed è quindi in realtà un firmware.

In generale, essendo concepiti per applicazioni specifiche e per l'uso in condizioni ambientali particolari che impongono limitazioni sia in termini di consumi che in termini di dissipazione di potenza, i microcontrollori presentano caratteristiche meno spinte rispetto ai microprocessori ma risultano più economici di questi ultimi sia per l'integrazione in un solo chip di una serie di componenti sia per fattori di economia di scala legati ai volumi di produzione molto più elevati.

Tabella 1: confronto microprocessore vs. microcontrollore

Caratteristica	**Microcontrollore**	**Microprocessore**
Velocità massima di clock	200Mhz	4GHz
Capacità elaborativa massima in MegaFLOPS	200	5000
Potenza minima dissipata in Watt (in stato di elaborazione)	0.001	50
Prezzo minimo per singola unità in USD	0.5	50
Numero di pezzi venduti annualmente (in milioni)	11,000	1,000

Componenti del microcontrollore

L'architettura del microcontrollore prevede un insieme di moduli fissi, comuni a tutti i modelli, e una serie di possibili estensioni in funzione del costruttore, del prezzo e della fascia applicativa):

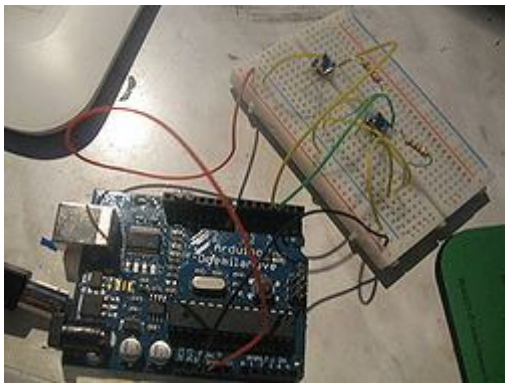
- Unità di elaborazione: CPU
- Memoria di programma: ROM, EPROM, FLASH
- Memoria dati: RAM e EEPROM
- Oscillatore interno o esterno
- Porte di I/O e/o GPIO configurabili
- Gestione *Interrupt*
- Moduli aggiuntivi
    - Contatori e timer
    - Moduli di comunicazione: USART, I2C, SPI, USB, Ethernet, IrDA, CAN, Wi-Fi, Zigbee
    - Interfacce analogiche o a tecnologia mista: ADC, DAC, PWM, Comparatori analogici
    - Interfacce di visualizzazione e controllo: (LCD, Touch sensor)

Piattaforma arduino

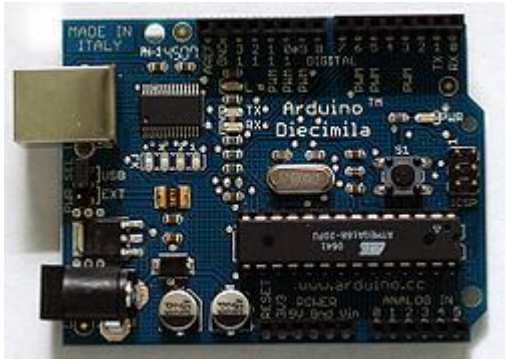
Arduino è un framework open source che permette la prototipazione rapida e l'apprendimento veloce dei principi fondamentali dell'elettronica e della programmazione. E' composto da una piattaforma hardware per il physical computing sviluppata presso l'Interaction Design Institute, un istituto con sede ad Ivrea, fondato da Olivetti e Telecom Italia. Il nome della scheda deriva da quello del bar di Ivrea frequentato dai fondatori del progetto, nome che richiama a sua volta quello di Arduino d'Ivrea, Re d'Italia nel 1002.

Questa si basa su un circuito stampato che integra un microcontrollore con PIN connessi alle porte I/O, un regolatore di tensione e quando necessario un'interfaccia usb che permette la comunicazione con il computer. A questo hardware viene affiancato un ambiente di sviluppo integrato (IDE) multipiattaforma (Linux, Apple Macintosh e Windows). Questo software permette anche ai novizi di scrivere programmi con un linguaggio semplice e intuitivo derivato da C e C++ chiamato Wiring, liberamente scaricabile e modificabile. Arduino può essere utilizzato per lo sviluppo di oggetti interattivi stand-alone ma può anche interagire, tramite collegamento, con software residenti su computer, come Adobe Flash, Processing, Max/MSP, Pure Data, SuperCollider, Vvvv.

Hardware

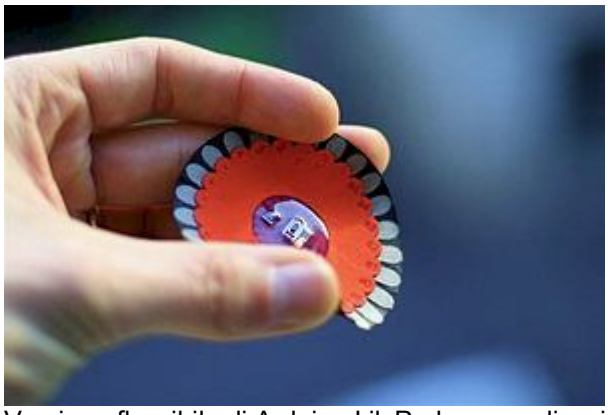

"Arduino Duemilanove" con i connettori di I/O digitale (in alto) e analogico (in basso) collegati ad una breadboard

L'"Arduino Uno".



L'"Arduino Diecimila".

Retro della scheda "Arduino Duemilanove"


Versione flessibile di Arduino LilyPad, per applicazioni su tessuti

L'hardware originale Arduino è realizzato dalla italiana Smart Projects. Alcune schede Arduino sono state progettate dalla statunitense SparkFun Electronics.

Una scheda Arduino tipica consiste in un microcontroller a 8-bit AVR prodotto dalla Atmel, con l'aggiunta di componenti complementari per facilitarne l'incorporazione in altri circuiti. In queste schede sono usati chip della serie megaAVR - nello specifico i modelli ATmega8, ATmega168, ATmega328, ATmega1280 e ATmega2560.

Molte schede includono un regolatore lineare di tensione a 5-volt e un oscillatore a cristallo a 16 MHz, sebbene alcune implementazioni, come ad esempio la piccola LilyPad, abbiano un clock di 8Mhz e facciano a meno dello stabilizzatore di tensione.

Fino a oggi, sono state commercializzate 16 versioni dell'hardware Arduino:

1.Serial Arduino, programmata con una porta seriale DB9. Fa uso del microcontroller ATmega8;

2.Arduino Extreme, con interfaccia di programmazione USB, facente uso del chip ATmega8;

3.Arduino Mini, una versione in miniatura facente uso di un ATmega168 a montaggio superficiale;

4.Arduino Nano, una versione ancor più piccola della Mini, utilizzante lo stesso controller ATmega168 SMD e alimentata tramite USB;

5.LilyPad Arduino, un progetto minimalista (scheda circolare dal diametro di 50 mm, per circa 0.8 mm di spessore), per applicazione su indumenti, con lo stesso ATmega168 in versione SMD;

6.Arduino NG, con un'interfaccia USB per programmare e usare un ATmega8;

7.Arduino NG plus, con interfaccia di programmazione USB, con un ATmega168;

8.Arduino BT, con interfaccia di programmazione Bluetooth e con un ATmega168;

9.Arduino Diecimila, con interfaccia di programmazione USB e con un ATmega168 in un package DIL28;

10.Arduino Duemilanove, facente uso del chip Atmega168 (o Atmega328 nelle versioni più recenti) e alimentata in corrente continua tramite USB, con commutazione automatica tra le sorgenti di alimentazione;

11.Arduino Mega, che fa uso di un ATmega1280 a montaggio superficiale per I/O e memoria addizionale;

12.Arduino Uno, evoluzione della Duemilanove con un differente chip, programmabile e più economico, dedicato alla conversione USB-seriale;

13.Arduino Mega2560, che fa uso di un ATmega2560 (anch'esso a montaggio superficiale) ed è un'evoluzione dell'Arduino Mega;

14.Arduino Due, che fa uso di un Atmel SAM3X8E ARM Cortex-M3 CPU;

15.Arduino Zero Pro, dotata di un microcontrollore Atmel SAMD21 basato su core ARM Cortex M0+ a 32 bit.

16.Arduino Yún, che fa uso di un ATmega32u4 e del processore Atheros AR9331, quest'ultimo compatibile con una distribuzione Linux basata su OpenWrt e chiamata Linino OS.

La programmazione del microcontroller

A livello concettuale, tutte le schede sono programmate attraverso una porta seriale RS-232, ma il modo in cui questa funzionalità è implementata nell'hardware varia da versione a versione. Le schede seriali Arduino contengono un semplice circuito che permette la

conversione tra il livello della RS-232 e il livello dei segnali TTL. Le versioni attuali di Arduino sono gestite via USB: l'ultima versione, la Uno, utilizza un microcontrollore Atmega8U2 programmato come convertitore USB-seriale mentre le precedenti versioni Diecimila e Duemilanove usavano chip adattatori USB-seriale, come gli FT232 di FTDI. Alcune varianti, come la Arduino Mini e la versione non ufficiale Boarduino, usano una scheda o un cavo adattatore USB-seriale separato.

Funzionalità di Input/Output

Per implementare il comportamento interattivo, Arduino è fornita di funzionalità di Input/Output, grazie alle quali essa riceve i segnali raccolti da sensori esterni. In base a tali valori, il comportamento della scheda è gestito dal microcontroller, in base alle decisioni determinate dal particolare programma in esecuzione in quel momento sulla scheda. L'interazione con l'esterno avviene attraverso attuatori pilotati dal programma attraverso i canali di output in dotazione.

Connettori I/O

A tale scopo, Arduino è dotata di molti dei connettori di Input/Output per microcontroller in uso su altri circuiti. Tutti i pin di I/O sono collocati sulla parte superiore della scheda, mediante connettori femmina da 0,1". Inoltre, sono disponibili commercialmente molte schede applicative plug-in, note come "shields".

Le schede Barebones e Boarduino, due cloni compatibili con la Arduino, sono dotate di connettori maschio sul lato inferiore del circuito in modo da poter essere connesse a una breadboard senza necessità di effettuare saldature.

I/O digitale

La Arduino Uno, ad esempio, che ha soppiantato la Duemilanove, offre 14 connettori per l'I/O digitale (numerati da 0 a 13). La direzione di funzionamento, input o output, è decisa dallo sketch programmato sull'IDE.

Sei dei canali I/O possono produrre segnali Pulse-width modulation (PWM). Attraverso i segnali PWM è possibile, ad esempio, regolare l'intensità di luminosità di un LED o la velocità di rotazione di un motorino elettrico. L'hardware di tre dei pin di I/O (9, 10 e 11) implementa la possibilità di gestirli direttamente attraverso la funzione analogWrite(), che permette di controllare la PWM del segnale in uscita in maniera efficiente, senza dover eseguire linee di codice appositamente predisposte. La funzione accetta due parametri, il primo dei quali è il pin pilotato mentre il secondo rappresenta l'intensità della modulazione (espressa su una scala da 0 a 255): così, ad esempio, analogWrite(9, 128) attiverà un led collegato al pin 9 al 50% della sua luminosità.

I/O analogico

Sempre sulla Uno, sono presenti altri 6 connettori specificamente dedicati a ingressi di segnali analogici, cioè valori di tensione letti da sensori esterni i cui valori, fino a un massimo di 5 Volt, sono convertiti in 1024 livelli discreti (da 0 a 1023). Questi 6 connettori possono essere riprogrammati (sempre dal codice dello sketch sull'IDE) per funzionare come normali entrate/uscite digitali.

Nella seguente tabella sono riepilogate le dotazioni dell'hardware dei vari modelli:

		Memoria			Connettori di I/O					
Arduino	**Microcontrollore**	**Flash KB**	**EEPROM KB**	**SRAM KB**	**Pin di I/O digitale**	**...di cui con PWM**	**Pin di Input analogico**	**Tipo interfaccia USB**	**Dimensioni in pollici**	**Dimensioni in millimetri**
Mega	ATmega1280	128	4	8	54	14	16	FTDI	4 × 2,1	101,6 × 53,3
Diecimila	ATmega168	16	0,5	1	14	6	6	FTDI	2,7 × 2,1	68,6 × 53,3
LilyPad	ATmega168V o ATmega328V	16	0,5	1	14	6	6	Nessuno	⌀ 2	⌀ 50
Duemilanove	ATmega168/328P	16/32	0,5/1	1/2	14	6	6	FTDI	2,7 × 2,1	68,6 × 53,3
Nano	ATmega168 o ATmega328	16/32	0,5/1	1/2	14	6	8	FTDI	1,70 × 0,73	43 × 18
Mega2560	ATmega2560	256	4	8	54	14	16	ATmega8U2	4 × 2,1	101,6 × 53,3
Uno	ATmega328P	32	1	2	14	6	6	ATmega8U2	2,7 × 2,1	68,6 × 53,3
Leonardo	Atmega32u4	32	1	2,5	20	7	12	Atmega32u4 integrato	2,7 × 2,1	68,6 × 53,3
Fio	ATmega328P	32	1	2	14	6	8	Nessuno	1,6 × 1,1	40,6 × 27,9
Yun	ATmega32u4	32	1	2,5	20	7	12	Atmega32u4 integrato	4 x 2,1	68,6 x 53,3
Due [9]	Atmel SAM3X8E	512		96	54	12	12	ATmega16U2 + native host	4 × 2,1	101,6 × 53,3

Alimentazione elettrica

L'alimentazione della scheda può avvenire attraverso la porta USB del computer (o attraverso la maggior parte degli alimentatori USB), oppure attraverso un adattatore in corrente continua a 9 volt, con connettore cilindrico (diametro 2,1 mm e positivo centrale). In quest'ultimo caso, la scheda commuta automaticamente sull'alimentazione esterna quando il connettore dell'alimentatore esterno è inserito, mentre commuta autonomamente sull'alimentazione USB in caso di disconnessione del connettore. La Arduino-NG e la Arduino Diecimila, versioni meno recenti, necessitano di essere commutate a mano, azionando uno switch ubicato tra la porta USB e l'ingresso dell'alimentazione esterna.

Schede Arduino compatibili

La quantità enorme e l'estrema variabilita d'uso e di componenti, rende difficile definire univocamente una scheda Arduino compatibile. Solitamente contengono un microcontroller a 8 16 o 32 bit AVR, PIC o ARM, con clock variabile tra gli 1 e 96Mhz. Molte incorporano componenti aggiuntivi pensati per i piu' svariati utilizzi.

Recentemente sono comparse sul mercato schede compatibili Arduino con implementato un processore PIC a 32 bit 80 MHz a prezzi paragonabili alla arduino UNO.

Software

IDE di Arduino

Screenshot dell'IDE di Arduino, che mostra un semplice codice di esempio.

L'ambiente di sviluppo integrato (IDE) di Arduino è un'applicazione multipiattaforma scritta in Java, ed è derivata dall'IDE creato per il linguaggio di programmazione Processing e per il progetto Wiring. È concepita per iniziare alla programmazione artisti e altri neofiti, che siano a digiuno di pratica nello sviluppo di software. Per permettere la stesura del codice sorgente, l'IDE include un editore di testo dotato inoltre di alcune particolarità, come il syntax highlighting, il controllo delle parentesi, e l'indentazione automatica. L'editor è inoltre in grado di compilare e lanciare il programma eseguibile in una sola passata e con un solo click. In genere non vi è bisogno di creare dei Makefile o far girare programmi dalla riga di comando.

L'ambiente di sviluppo integrato di Arduino è fornito di una libreria software C/C++ chiamata "Wiring" (dall'omonimo progetto Wiring): la disponibilità della libreria rende molto più semplice implementare via software le comuni operazioni di input/output. I programmi di Arduino sono scritti in C/C++, ma all'utilizzatore, per poter creare un file eseguibile, non è richiesto di scrivere un programma in C, ma solo di definire due funzioni:

setup()

- – funzione invocata una sola volta all'inizio di un programma che può essere utilizzata per i settaggi iniziali;

loop()

- – funzione chiamata ripetutamente, la cui esecuzione si interrompe solo con lo spegnimento della scheda.

Esempi di codice

Un tipico esempio di programma per iniziare a far pratica con un microcontroller è quello, molto semplice, che permette l'accensione ripetuta di un LED. Nell'ambiente di sviluppo di Arduino, l'utilizzatore potrebbe farlo con un programmino come il seguente:

```
#define LED_PIN 13

void setup () {
   pinMode (LED_PIN, OUTPUT);     // abilita il pin 13 per l'output digitale
}

void loop () {
   digitalWrite (LED_PIN, HIGH);  // accende il LED
   delay (1000);                  // aspetta 1 secondo (1000 millisecondi)
   digitalWrite (LED_PIN, LOW);   // spegne il LED
   delay (1000);                  // aspetta un secondo
}
```

Scritto in questo modo, il codice non verrebbe riconosciuto come programma valido da un compilatore C++; per questo motivo, quando l'utilizzatore seleziona il bottone di comando "Upload to I/O board" (presente nell'IDE), una copia del codice viene automaticamente trascritta in un file temporaneo con l'aggiunta di un extra include header all'inizio del codice e l'implementazione di una semplicissima funzione main() in basso:

```
#include "WProgram.h"
#define LED_PIN 13

void setup () {
   pinMode (LED_PIN, OUTPUT);     // abilita il pin 13 per l'output digitale
}

void loop () {
   digitalWrite (LED_PIN, HIGH);  // accende il LED
   delay (1000);                  // aspetta 1 secondo (1000 millisecondi)
   digitalWrite (LED_PIN, LOW);   // spegne il LED
   delay (1000);                  // aspetta 1 secondo
}

int main(void)
{
   init();

   setup();

   for (;;)
```

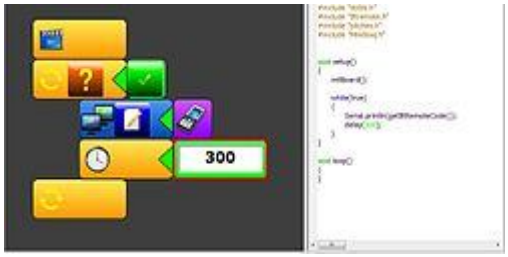
```
        loop();

    return 0;
}
```

"WProgram.h" è lo header file principale per la libreria Wiring. La funzione main() effettua solo tre chiamate, init(), setup() e loop(); le ultime due sono definite dall'utente, mentre la prima è scritta nella libreria stessa. La funzione loop(), come richiesto, è automaticamente nidificata in un loop infinito.

Lo IDE di Arduino usa la GNU toolchain e la AVR Libc per compilare i programmi, mentre usa avrdude per caricarli sulla scheda.

Minibloq



Esempio di codice generato con Minibloq: i dati provenienti da un telecomando a infrarossi RC5 sono inviati alla porta seriale

Un esempio alternativo di ambiente integrato dedicato ad Arduino (ma anche ad altre utilizzazioni) è Minibloq, distribuito con licenza libera, che mette a disposizione un particolare ambiente di sviluppo visuale per la programmazione di Arduino, robot, e altri simili dispositivi di physical computing.

Caratteristiche di Minibloq sono l'interfaccia intuitiva, e la capacità di generare il codice a partire da uno schema grafico a blocchi disegnato dall'utente. Per questa sua impostazione intuitiva, il software è indirizzato ad avvicinare principianti e neofiti alla programmazione di microcontroller, in contesti scolastici ed educativi che vanno dalla scuola primaria all'ambiente liceale.

L'applicazione, compilata con GCC, si presta a essere utilizzata anche su computer a più basse prestazioni, come netbooks, OLPC XO-1, Intel-Classmate.

Android Device Kit

In occasione della conferenza Google I/O di maggio 2011, Google ha presentato ADK (Android Device Kit), un kit di sviluppo basato su Arduino e dedicato alla produzione di accessori Android. Attualmente è disponibile il solo supporto per USB ma nei prossimi mesi arriverà anche il supporto per il Bluetooth.

L'ADK, per ora, si basa sulla scheda Arduino Mega2560.

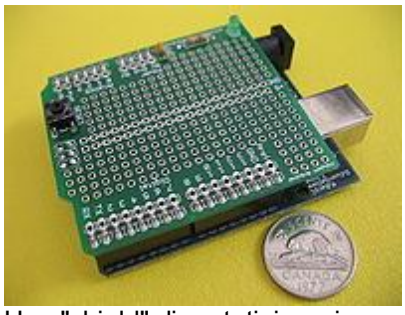
Open hardware e open source

Gli schemi hardware di Arduino sono distribuiti in modo da poter essere utilizzati nei termini legali di una licenza Creative Commons Attribution Share-Alike 2.5, e sono

disponibili sul sito ufficiale Arduino. Per alcune versioni della scheda sono disponibili anche il layout e i file di produzione.

Il codice sorgente per l'Ambiente di sviluppo integrato e la libreria residente sono invece disponibili, e concessi in uso, secondo i termini legali contenuti nella licenza GPLv2.

Accessori hardware



Uno "shield" di prototipizzazione montato su una scheda Arduino.

Sia la scheda originale, sia i cloni, fanno uso di shields, ovvero di espansioni alla Arduino base, realizzate con schede a circuito stampato che possono essere collocate al di sopra della Arduino, inserendosi nei connettori già normalmente presenti su Arduino. Esistono espansioni dedicate a varie funzioni, dal controllo motorio, al breadboarding (prototipizzazione).

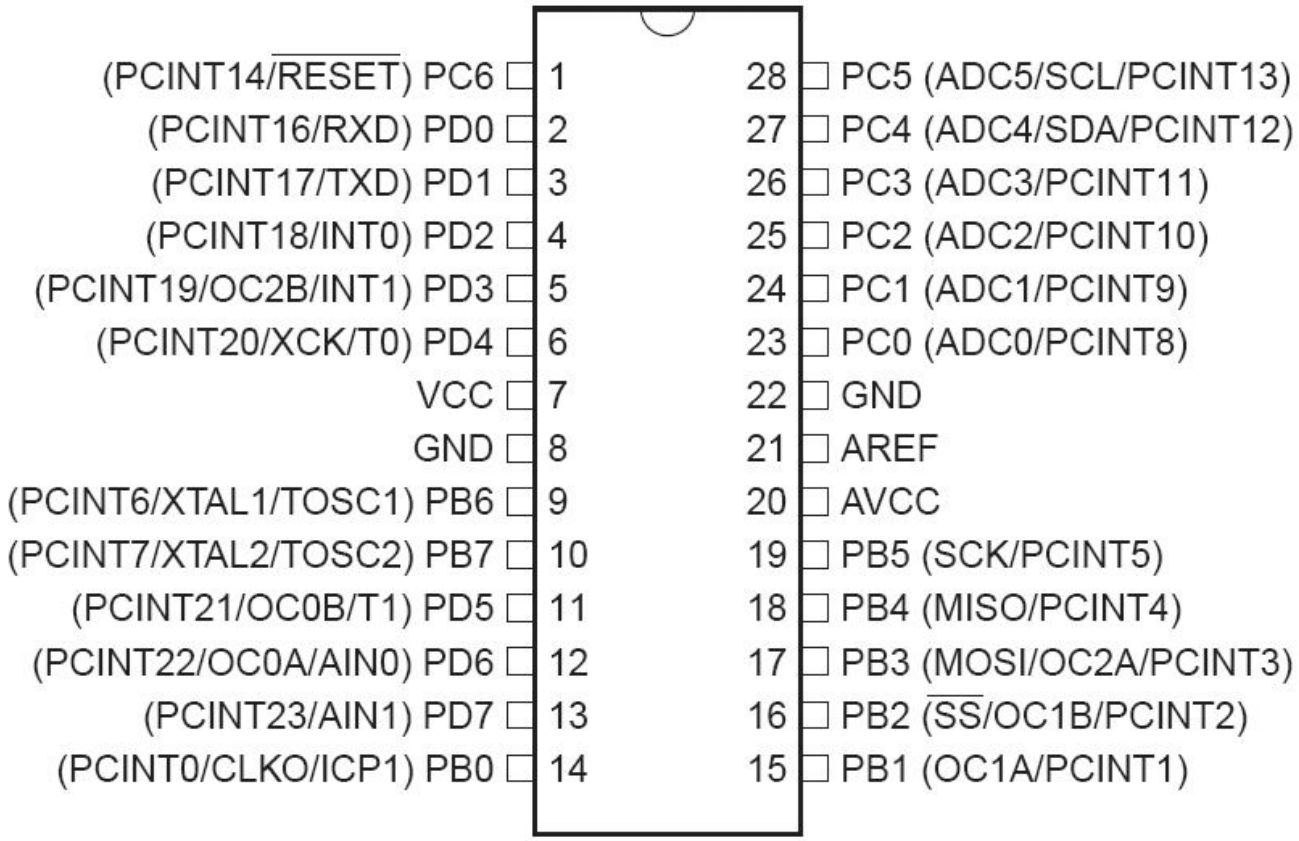
Cloni

Benché i progetti hardware e software siano resi disponibili con licenze copyleft, gli sviluppatori hanno espresso il desiderio che il nome "Arduino" (o suoi derivati) venga riferito solo al prodotto originale e non sia usato, senza permesso, per indicare opere derivate.

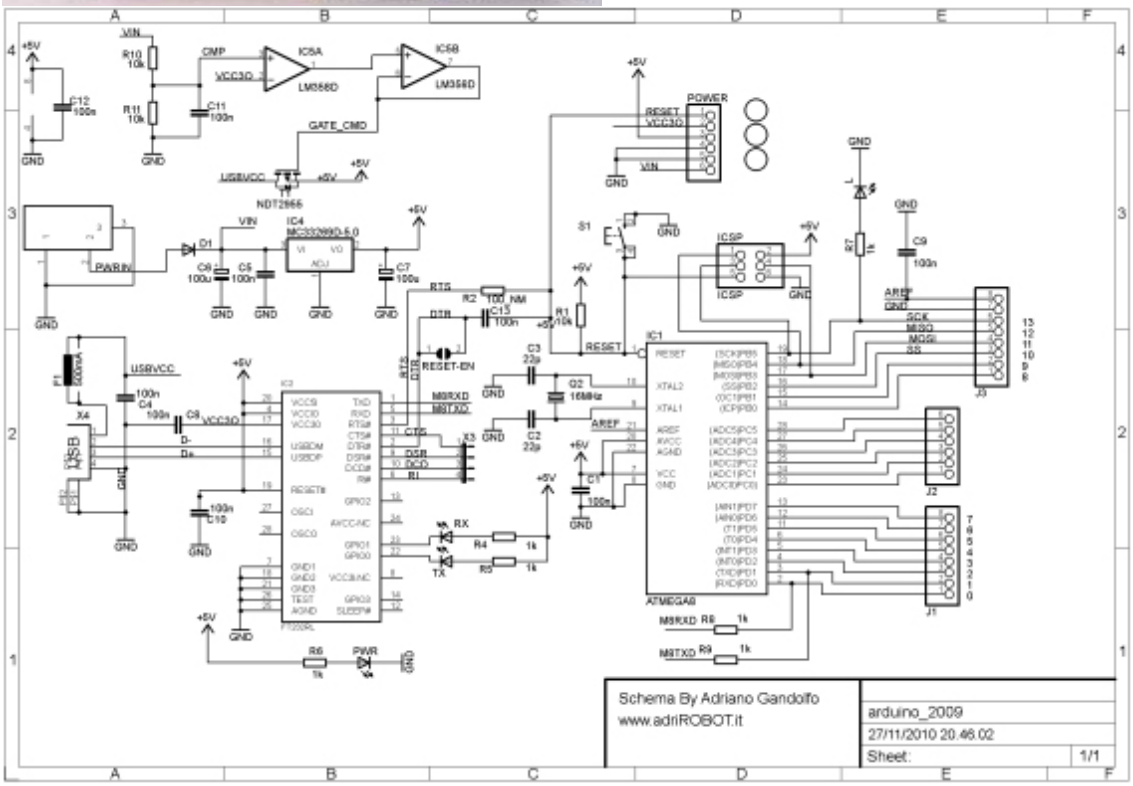
Il documento che esprime la policy ufficiale sull'uso del nome "Arduino" mette l'accento su come il progetto sia aperto a incorporare lavori altrui nel prodotto ufficiale.

Quale conseguenza di queste convenzioni sulla protezione del nome, un gruppo di utilizzatori ha effettuato un "fork" (nel senso esteso del termine) della "the Arduino Diecimila", distribuendo una scheda equivalente chiamata "Freeduino". Il nome "Freeduino" non è però un marchio commerciale ma è liberamente utilizzabile da chiunque lo desideri.

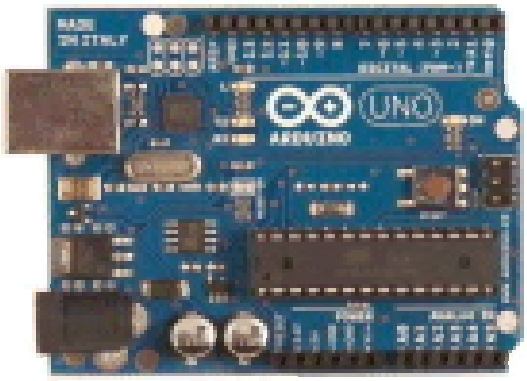
MCU utilizzata su Arduino 2009 e UNO

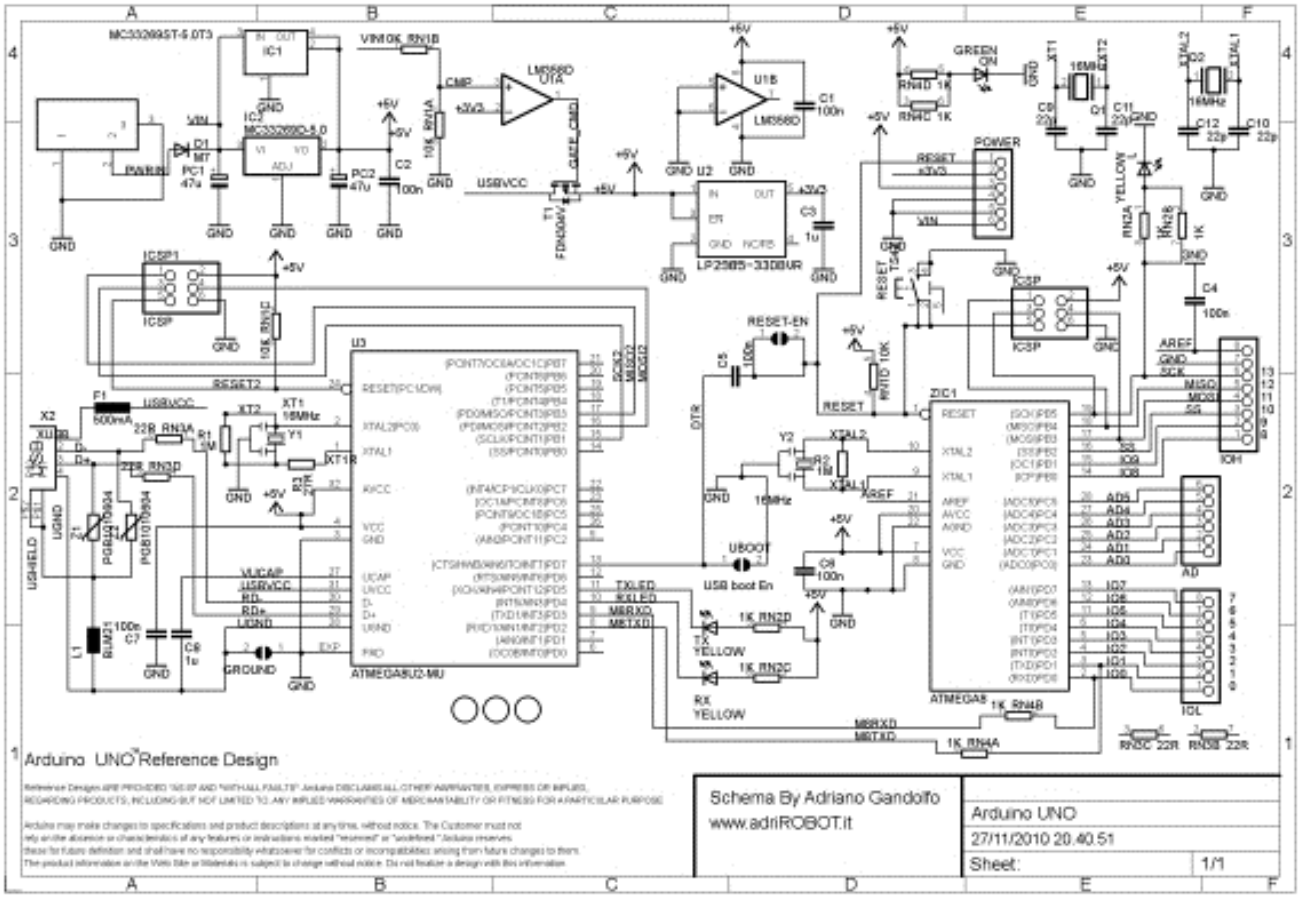


Arduino 2009 e relativo schema elettrico

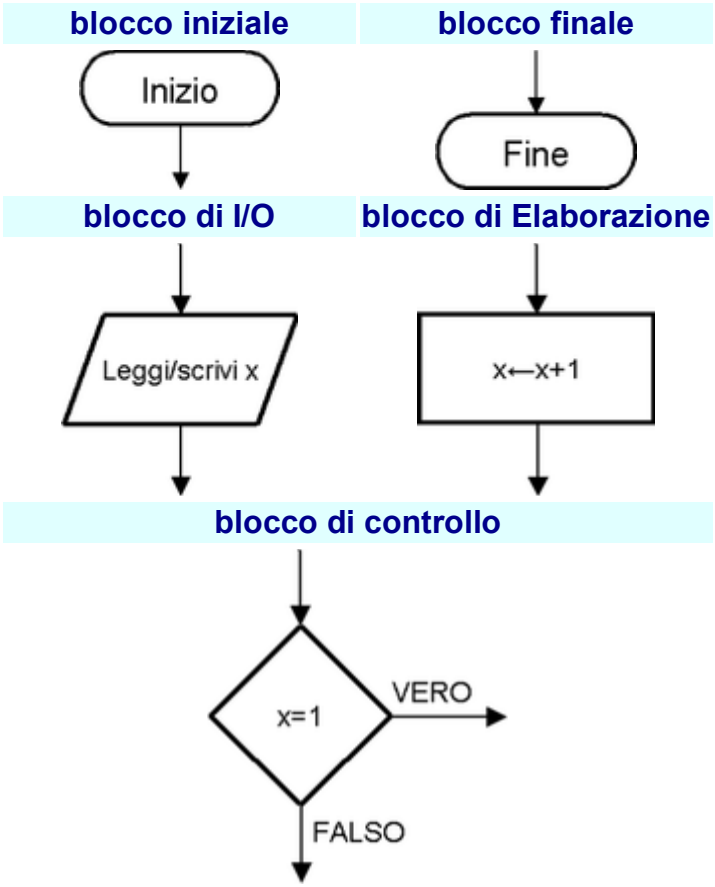


Arduino UNO e relativo schema elettrico





Elementi di diagrammazione a blocchi



Language Reference

Arduino programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

Structure

setup()
loop()

Control Structures

if
if...else
for
switch case
while
do... while
break
continue
return
goto

Further Syntax

; (semicolon)
{} (curly braces)
// (single line comment)
/* */ (multi-line comment)
#define
#include

Arithmetic Operators

= (assignment operator)
+ (addition)
- (subtraction)
* (multiplication)
/ (division)
% (modulo)

Comparison Operators

== (equal to)
!= (not equal to)
< (less than)
> (greater than)
<= (less than or equal to)
>= (greater than or equal to)

Boolean Operators

Variables

Constants

HIGH | LOW
INPUT | OUTPUT
true | false
integer constants
floating point constants

Data Types

void
boolean
char
unsigned char
byte
int
unsigned int
word
long
unsigned long
float
double
string - char array
String - object
array

Conversion

char()
byte()
int()
word()
long()
float()

Variable Scope & Qualifiers

variable scope
static
volatile
const

Utilities

sizeof()

Functions

Digital I/O

pinMode()
digitalWrite()
digitalRead()

Analog I/O

analogReference()
analogRead()
analogWrite() - *PWM*

Advanced I/O

tone()
noTone()
shiftOut()
shiftIn()
pulseIn()

Time

millis()
micros()
delay()
delayMicroseconds()

Math

min()
max()
abs()
constrain()
map()
pow()
sqrt()

Trigonometry

sin()
cos()
tan()

Random Numbers

randomSeed()

&& (and)
|| (or)
! (not)

Pointer Access Operators

* dereference operator
& reference operator

Bitwise Operators

& (bitwise and)
| (bitwise or)
^ (bitwise xor)
~ (bitwise not)
<< (bitshift left)
>> (bitshift right)

Compound Operators

++ (increment)
-- (decrement)
+= (compound addition)
-= (compound subtraction)
*= (compound multiplication)
/= (compound division)
&= (compound bitwise and)
|= (compound bitwise or)

random()

Bits and Bytes

lowByte()
highByte()
bitRead()
bitWrite()
bitSet()
bitClear()
bit()

External Interrupts

attachInterrupt()
detachInterrupt()

Interrupts

interrupts()
noInterrupts()

Communication

Serial

Istruzioni Standard Arduino

Qui di seguito sono riportate le istruzioni standard supportate dal linguaggio di programmazione di Arduino**.**

STRUTTURA

Il codice di qualsiasi programma per Arduino è composto essenzialmente di due parti:

void setup() - Questo è il posto dove mettiamo il codice di inizializzazione. Inizializza tutte le impostazioni e le istruzioni della
scheda (gli INPUT e OUTPUT) prima che il ciclo principale del programma si avvii.

void loop() - E' il contenitore del codice principale del programma. Contiene una serie di istruzioni che possono essere ripetute
una dopo l'altra fino a quando non spegniamo la scheda Arduino.

COSTANTI

Nella scheda Arduino è inserita una serie predefinita di parole chiave con valori speciali. High e Low sono usati per esempio quando si vuole accendere o spegnere un Pin di Arduino. INPUT e OUTPUT sono usate per definire se uno specifico Pin deve essere un dato di entrata o un dato di uscita. True e False indicano il rispettivo significato italiano: se abbiamo un'istruzione, la condizione può essere vera o falsa.

VARIABILI

Sono aree della memoria di Arduino dove si possono registrare dati e intervenire all'interno del programma. Come dice il nome stesso, le variabili possono essere cambiate tutte le volte che vogliamo. Quando si dichiara una variabile bisogna dichiararne anche il tipo. Questo significa dire al processore le dimensioni del valore che si vuole memorizzare. Ne esistono di diversi tipi:

boolean - Può assumere solamente due valori: vero o falso.

char - Contiene un singolo carattere. L'Arduino lo registra come un numero (ma noi vediamo il testo). Quando i caratteri sono usati per registrare un numero, possono contenere un valore compreso tra -128 e 127.

byte - Può contenere un numero tra 0 e 255. Come un carattere usa solamente un byte di memoria.

int - Contiene un numero compreso tra -32768 e 32767. E' il tipo di variabile più usata e usa 2 byte di memoria.

unsigned int - Ha la stessa funzione di int, solo che non può contenere numeri negativi, ma numeri tra 0 e 65.535.

long - E' il doppio delle dimensioni di un int e contiene i numeri da -2147483648 a 2147483647.

unsigned long - Versione senza segno di long va da 0 a 4294967295.

float - Può memorizzare numeri con la virgola. Occupa 4 bytes della RAM.

double - A doppia precisione in virgola mobile con valore massimo di 1797693134862315 7x10^308.

string - Un set di caratteri ASCII utilizzati per memorizzare informazioni di testo. Per la memoria, usa un byte per ogni carattere

della stringa, più un carattere NULL che indica ad Arduino la fine della stringa. Esempio:

```
char string1[] = "Hello"; // 5 caratteri+carattere NULL
char string2[6]="Hello"  // La stessa cosa di sopra
```

N.B. ogni istruzione deve sempre terminare con ";" in tale linguaggio. Inoltre "//" è usato per inserire commenti che aiutano a comprenderlo.

array - un elenco di variabili accessibili tramite un indice. Vengono utilizzate per creare tabelle di valori facilmente accessibili. Come esempio se si vuole memorizzare diversi livelli di luminosità di un LED possiamo creare 4 variabili, una per ogni livello di luminosità.
Si può utilizzare una semplice array come:

```
int Luce[5]={0,25,50,100};
```

Nel tipo della variabile la parola "array" non si dichiara, ma si usano i simboli [] e {}.

STRUTTURE DI CONTROLLO

Il linguaggio di Arduino include parole chiave per controllare il progetto logico del nostro codice.

If…else - Permette di prendere delle decisioni all'interno del programma, ma deve essere seguito da una domanda sotto forma di espressione tra parentesi. Se la domanda è vera tutto ciò che segue verrà eseguito. Se falso verrà eseguito tutto il codice che segue else. If è possibile usarlo senza usare necessariamente else. Esempio:

```
if (val=1){digitalWrite(LED, HIGH);} // (val=1) è la domanda se è vera esegue ciò che è fra parentesi
```

For - Ripete il codice per un numero predefinito di volte.

```
for(int i=0;i<10;i++){Serial.print("Ciao");}  //stampa 10 volte "Ciao"
```

Switch - E' come un interruttore nel corso del programma. Fa prendere al programma diverse direzioni in base al valore della
variabile (il suo nome deve essere messo tra parentesi dopo switch). E' utile perché può sostituire lunghe serie di if.

```
 switch(valore sensore)
{case 38:digitalwrite(12, High);break;
case 55:digitalwrite(3, High);break;
default: // si usa per indicare tutti i casi in cui non è ne 38 ne 55
digitalwrite(12, Low); digitalwrite(13, Low);}
```

While - Esegue un blocco di codice fino a quando una certa condizione posta tra le parentesi è vera.

```
while(valore sensore<500){digitalWrite(13, HIGH); delay(100); digitalWrite (13, HIGH);
delay(100); Valoresensore=analogRead(1);
```

Do…While - E' uguale a while solo che il codice è avviato prima che la condizione sia verificata. Si usa quando si vuole eseguire il
codice almeno una volta prima che la condizione sia valutata. Esempio:

```
do {digitalWrite(13,HIGH); delay(100); digitalWrite(13,HIGH); DELAY (100);
valore sensore=analogread(1); }
```

Break - Questo termine consente di bloccare il ciclo e continuare ad eseguire il codice fuori dal ciclo. Viene utilizzato anche per separare le varie condizioni nella funzione Switch.

Continue - Questo comando fa saltare il resto del codice all'interno del ciclo, e riavvia il ciclo. Esempio:

for(luminosità=0; luminosità<200; luminosità++)f((x>120) && (x<180)) continue;
analogWrite(PWMPin, luminosità); delay(20); }

Return - Ferma una funzione che si sta eseguendo e restituisce un risultato. E' possibile infatti usarlo per restituire un valore da una funzione. Esempio chiama una funzione "calcolaumidità" e ritorna il valore dell'umidità.

Int calcolaumidità() {Int umidità=0; umidità0(analogread(0)+45/100)/100;return umidità;}

OPERAZIONI ARITMETICHE

Si può usare Arduino per compiere operazioni matematiche complesse con una semplice sintassi: + e – indicano addizione e sottrazione, * indica la moltiplicazione, e / la divisione.
C'è un operatore in più in questo linguaggio chiamato "Modulo" che è un comando che restituisce il resto di una divisione. Esempio:

*a=3+3; luminosità=((12*valore sensore)/4);*

OPERATORI DI COMPARAZIONE

Quando si specificano delle condizioni ci sono vari operatori che tu puoi usare:

== Uguale a

> maggiore di

< minore di

!= diverso da

<= minore o uguale

>= maggiore o uguale

OPERATORI BOOLEANI

Sono usati quando si vogliono combinare più condizioni, ad esempio se vogliamo verificare se il valore di un sensore è tra 1 e 5
basta scrivere:

if(sensore=>1) && (sensore=<=5);

Esistono tre tipi di operazioni booleane: &&(And), ||(Or), !(Not).

OPERATORI COMPUTAZIONALI

Servono a ridurre la mole di un codice e a renderlo più semplice e chiaro per operazioni semplici come incrementare o decrementare una variabile.

Esempio: val=val+1; è come dire val++

incremento (++) e decremento (--)

++ e -- incrementano/decrementano una variabile di 1; lo stesso è applicabile a +=, -=, *=, /= .

Esempio: le seguenti espressioni sono equivalenti:

val=val+5;

Val+=5;

FUNZIONI INPUT E OUTPUT

Arduino include funzioni per la gestione degli Input e degli Output.

pinMode(pin,mode) - Riconfigura un pin digitale a comportarsi come uscita o come entrata.

pinMode(13,INPUT) - imposta il pin 13 come Input.

digitalWrite(pin,value) - imposta un pin digitale ad ON o a OFF.

digitalWrite(7,HIGH) - imposta come 1 digitale il pin 7.

int digitalRead(pin) - Legge lo stato di un input Pin, ritorna HIGH se il Pin riceve della tensione oppure LOW se non c'è tensione
applicata.

```
Val=digitalRead(7); // legge il pin 7 dentro a val
```

Int analogRead(pin) - Legge la tensione applicata a un ingresso analogico e ritorna un numero tra 0 e 1023 che rappresenta le
tensioni tra 0 e 5 V.

```
val=AnalogRead(0); // legge l'ingresso analogico 0 dentro a val
```

analogWrite(pin,value) - Cambia la frequenza PWM su uno dei pin segnati PWM, nella voce pin si può mettere 11 10 9 6 5 3,
value invece può essere un valore da 0 a 255 che rappresenta la scala da 0 a 5 V.

```
analogWrite(9,128);
```

shiftOut(dataPin, clock, Pin, bit, Order, value) - Invia i dati ad un registro. Questo protocollo usa un pin per i dati e uno per il
clock. bitOrder indica l'ordine dei bytes (least significant byte=LSB, most significant byte=LMB) e value è il byte da inviare. Esempio:

```
shiftOut(dataPin, Clock Pin, LSBFIRST, 255);
```

unsigned long pulseIn(pin, value) - misura la durata degli impulsi in arrivo su uno degli ingressi digitali. E' utile ad esempio per
leggere alcuni sensori a infrarossi o alcuni accelerometri che emettono impulsi di diversa durata.

```
Tempo=pulsin(8,HIGH);
```

FUNZIONI DI TEMPO

Arduino include alcune funzioni per misurare il tempo trascorso e anche per mettere in pausa il nostro programma.

unsigned long millis() - Ritorna il numero in millisecondi trascorsi dall'inizio del programma, esempio:

```
durata=millis()-tempo // calcola il tempo trascorso prima di "tempo"
```

delay(ms) - Mette in pausa il programma per un numero di millisecondi specificato.

```
delay(1000); //stoppa il programma per 1 secondo
```

delayMicroseconds(us) - Come delay mette in pausa il programma ma l'unità di misura è molto più piccola, parliamo di microsecondi.

```
delayMicroseconds(2000); // aspetta per 2 millisecondi (1000us=1ms)
```

FUNZIONI MATEMATICHE

Arduino include molte funzioni matematiche comuni. Servono, per esempio, per trovare il numero max o il numero min.

min (x,y) - Ritorna il più piccolo fra x e y. Esempio:

```
Val= min(5,20);  // val adesso è 5
```

max(x,y) - Ritorna il più grande fra x e y.

abs(x) - Ritorna il valore assoluto di x, ossia trasforma i numeri negativi in numeri positivi. Se x fosse 5 ritorna 5, ma anche se x
fosse -5 ritorna sempre 5. Esempio:

```
Val= abs(-5) // val vale 5
```

constrain(x,a,b) - Ritorna il valore "x" costretta tra "a" e "b". Ciò vuol dire che se "x" è minore di "a" ritornerà semplicemente
"a" e se x è maggiore di "b" restituirà semplicemente il valore di "b".

map(value, fromLow, fromHigh, toHigh) - Associa un valore che sta nel range fromLow e maxlow in un nuovo range che va
da toLow a toHigh. E' molto utile per processare valori provenienti da sensori analogici. Esempio:

```
val=map(analogRead(0),0,1023,100,200);  // associa il valore analogico 0 ad un valore tra 100 e 200
```

double pow(base,exponent) - Restituisce come risultato la potenza di un numero. Si deve indicare la base e l'esponente.

Double sqrt(x) - Restituisce la radice quadrata di un numero x.

Double sin(rad) - Restituisce il seno dell'angolo specificato in radianti. Esempio:

```
Double sine= sine(2); // circa 0.909297370
```

Double cos(rad) - Restituisce il coseno dell' angolo specificato in radianti.

Double tan(rad) - Restituisce il valore della tangente di un angolo specificato in radianti.

FUNZIONI NUMERI RANDOM

Se si ha bisogno di generare numeri random (a caso), Arduino ci viene incontro con alcuni comandi standard per generarli.

randomSeed(seed) - Anche se la distribuzione di numeri restituita dal comando random() è essenzialmente casuale, la sequenza è prevedibile. randomSeed(seed) inizializza il generatore di numeri pseudo-casuali, facendola partire da un punto arbitrario nella sua sequenza casuale.

Long random(min,max) - Restituisce un valore long intero di valore compreso fra min e max -1. Se min non è specificato il suo
valore minimo è 0. Esempio:

```
long random= random(13); // numero compreso fra 0 e 12
```

COMUNICAZIONE SERIALE

Queste sono le funzione seriali cioè quelle funzioni che Arduino usa per comunicare tramite la porta Usb del nostro Pc.

Serial.begin(speed) - Prepara Arduino a mandare e a ricevere dati tramite porta seriale. Possiamo usare generalmente 9600 bits per secondo con la porta seriale dell'Arduino, ma sono disponibili anche altre velocità, di solito non si supera i 115.200 bps.

```
Serial.print(data)Serial.begin(9600);
```

Serial.print(data,codifica) - Invia alcuni dati alla porta seriale. La codifica è opzionale.

```
Serial.print(32);  // stampa 32

Serial.Print(32, DEC);   // stampa 32 come sopra

Serial.Print(32, OCT);  // 40  (stampa10 in ottale)

Serial.Print(32 , BIN);  // 100000 (stampa 10 in binario)

Serial.Print(32 , BYTE);  // "Space" valore associato nella tabella ASCII
```

Int Serial.available() - Ritorna quanti bytes non ancora letti sono disponibili sulla porta Serial per leggerli tramite la funzione read(). Dopo aver read() tutti i bytes disponibili Serial.Available restituisce 0 fino a quando nuovi dati non giungono sulla Porta.

Int.Serial.read() - Recupera un byte di dati in entrata sulla porta Seriale.

```
int data= Serial.read();
```

Poichè i dati possono giungere nella porta seriale prima che il programma li possa leggere(per la velocità), Arduino salva tutti i dati in un buffer. Se è necessario ripulire il buffer e aggiornarlo con i dati aggiornati, usiamo la funzione flush().

```
Serial.flush();
```

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from **Sketch > Import Library**.

Standard Libraries

EEPROM - reading and writing to "permanent" storage
Ethernet - for connecting to the internet using the Arduino Ethernet Shield
Firmata - for communicating with applications on the computer using a standard serial protocol.
LiquidCrystal - for controlling liquid crystal displays (LCDs)
SD - for reading and writing SD cards
Servo - for controlling servo motors
SPI - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
SoftwareSerial - for serial communication on any digital pins
Stepper - for controlling stepper motors
Wire - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

These libraries are compatible Wiring versions, and the links below point to the (excellent) Wiring documentation.

Matrix - Basic LED Matrix display manipulation library
Sprite - Basic image sprite manipulation library for use in animations with an LED matrix

Contributed Libraries

If you're using one of these libraries, you need to install it first. To do so, download the library and unzip it. It should be in a folder of its own, and will typically contain at least two files, one with a .h suffix and one with a .cpp suffix. Open your Arduino sketchbook folder. If there is already a folder there called **libraries**, place the library folder in there. If not, create a folder called **libraries** in the sketchbook folder, and drop the library folder in there. Then re-start the Arduino programming environment, and you should see your new library in the **Sketch > Import Library** menu.

For details, see the page on the Arduino environment.

Communication (networking and protocols):

Messenger - for processing text-based messages from the computer
NewSoftSerial - an improved version of the SoftwareSerial library
OneWire - control devices (from Dallas Semiconductor) that use the One Wire protocol.
PS2Keyboard - read characters from a PS2 keyboard.
Simple Message System - send messages between Arduino and the computer
SSerial2Mobile - send text messages or emails using a cell phone (via AT commands over software serial)
Webduino - extensible web server library (for use with the Arduino Ethernet Shield)
X10 - Sending X10 signals over AC power lines
XBee - for communicating with XBees in API mode
SerialControl - Remote control other Arduinos over a serial connection

Sensing:

Capacitive Sensing - turn two or more pins into capacitive sensors
Debounce - for reading noisy digital inputs (e.g. from buttons)

Displays and LEDs:

Improved LCD library fixes LCD initialization bugs in official Arduino LCD library
GLCD - graphics routines for LCD based on the KS0108 or equivalent chipset.
LedControl - for controlling LED matrices or seven-segment displays with a MAX7221 or MAX7219.

LedControl - an alternative to the Matrix library for driving multiple LEDs with Maxim chips.
LedDisplay - control of a HCMS-29xx scrolling LED display.

Frequency Generation and Audio:

Tone - generate audio frequency square waves in the background on any microcontroller pin

Motors and PWM:

TLC5940 - 16 channel 12 bit PWM controller.

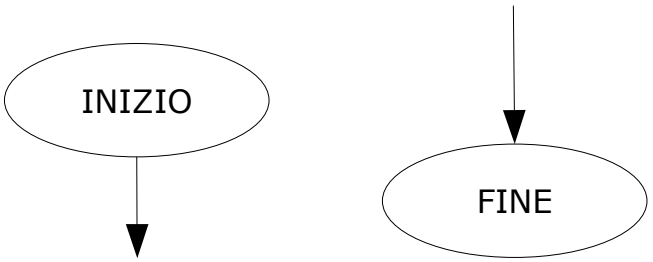
Timing:

DateTime - a library for keeping track of the current date and time in software.
Metro - help you time actions at regular intervals
MsTimer2 - uses the timer 2 interrupt to trigger an action every N milliseconds.

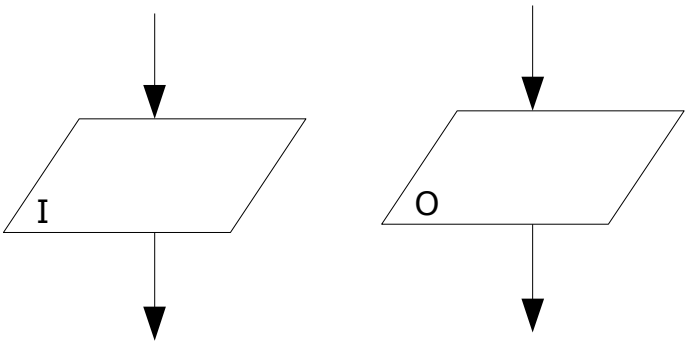
Utilities:

TextString aka String - handle strings
PString - a lightweight class for printing to buffers
Streaming - a method to simplify print statements

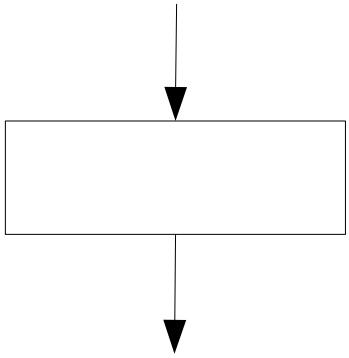
Diagrammi a blocchi: elementi di base



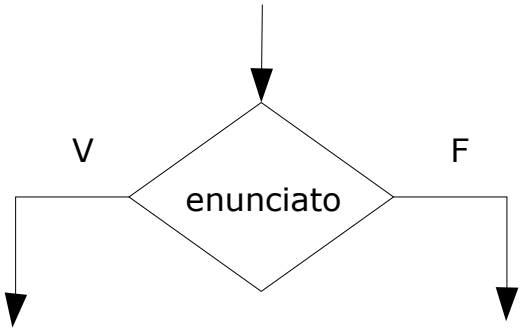
Blocchi terminali



Blocchi di I/O

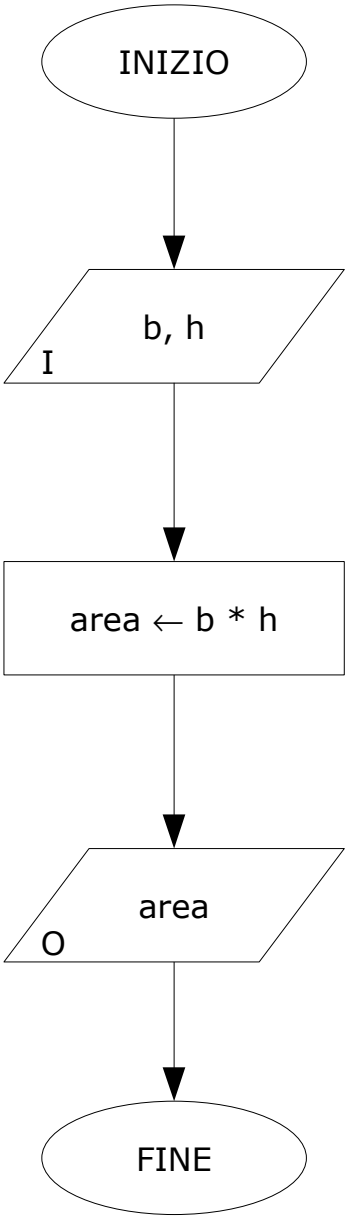


Blocco esecutivo



Blocco decisionale

Esempio: area di un rettangolo

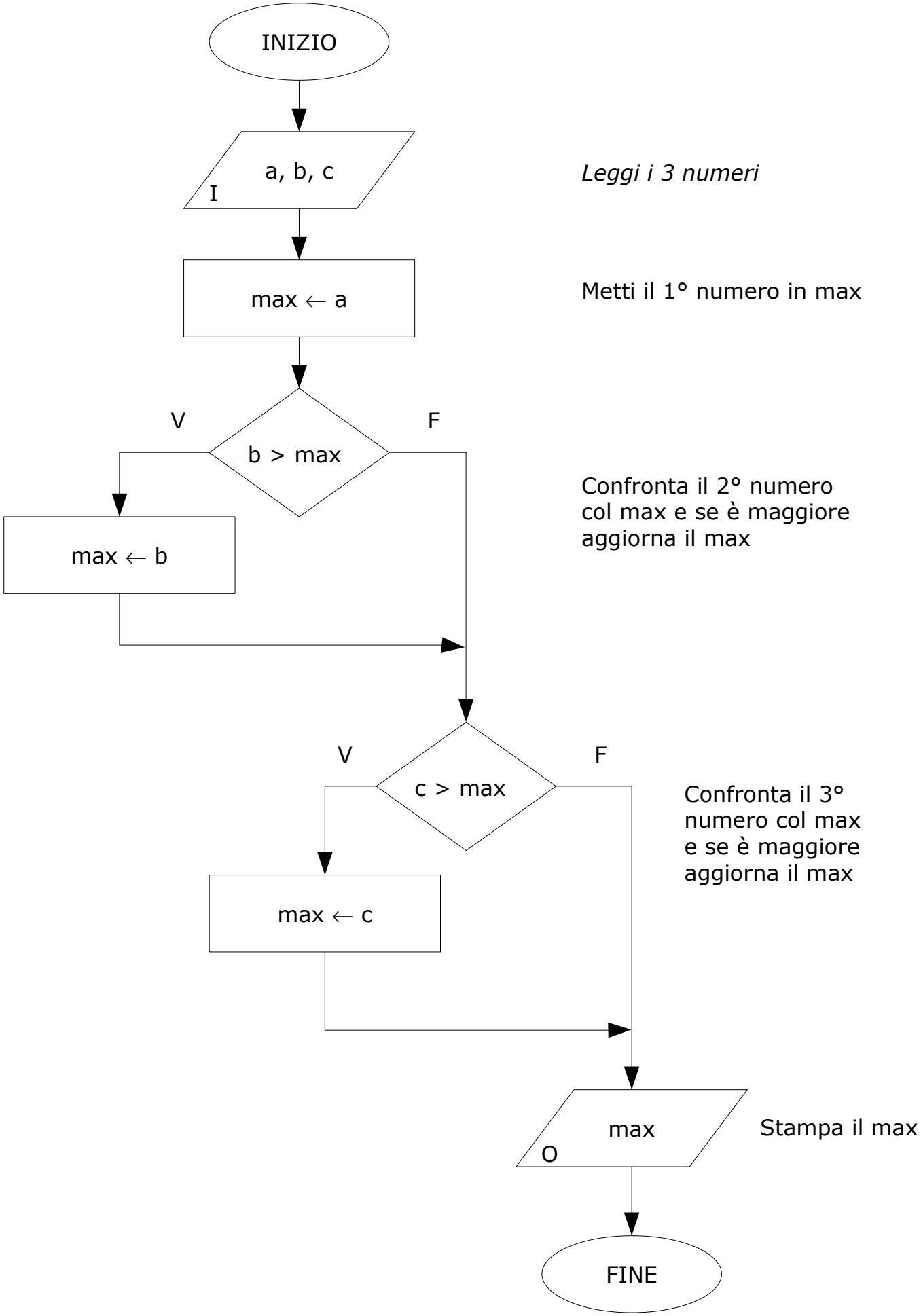


Leggi la base e l'altezza e memorizzale nelle variabili b *e* h

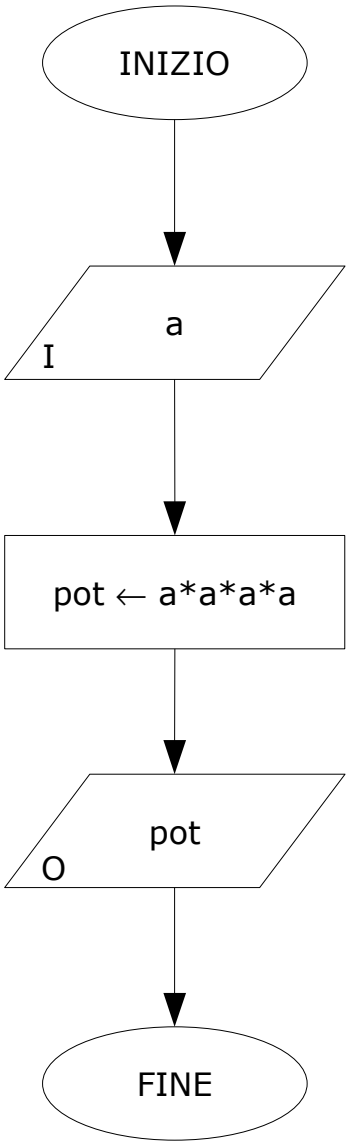
Calcola b *per* h *e memorizza il risultato nella variabile* area

Scrivi il contenuto della variabile area

Esempio: max numero tra 3 letti



Esempio: potenza 4 di un numero

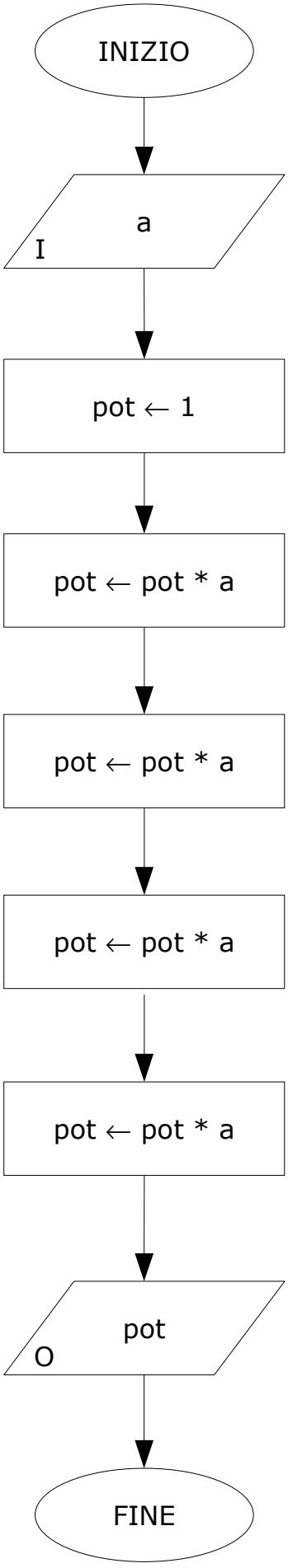


Leggi numero

Calcola la potenza

Stampa la potenza

Esempio: potenza 4 di un numero (v. 2)



Leggi numero

Poni la potenza uguale a 1

Calcola il numero alla 1

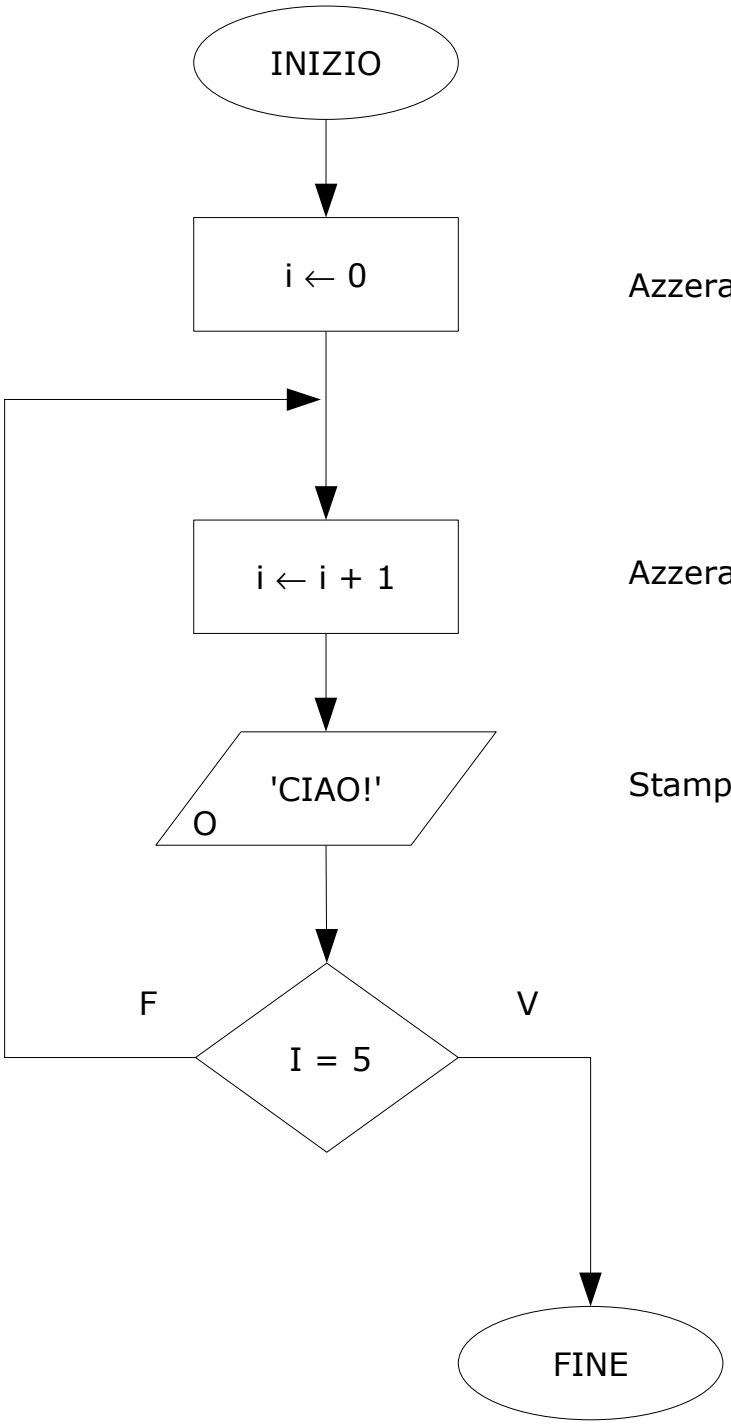
Calcola il numero alla 2

Calcola il numero alla 3

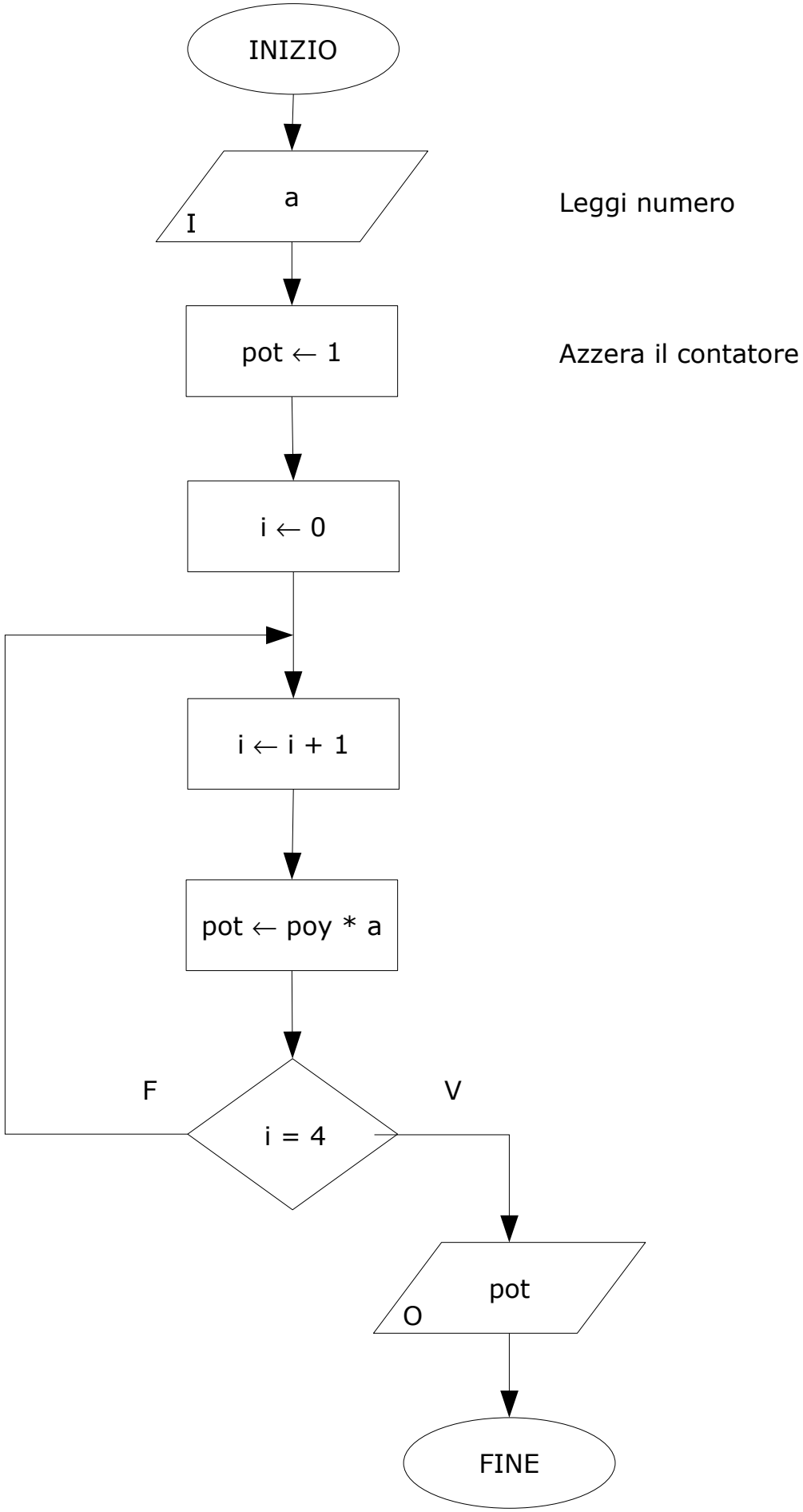
Calcola il numero alla 4

Stampa la potenza

Ciclo FOR: per 5 volte stampa *CIAO!*



Esempio: potenza 4 di un numero (v. 3)

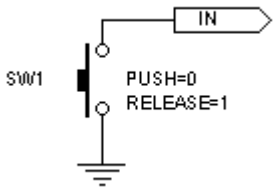


Ingressi

I circuiti di ingresso si applicano ai pin di ARDUINO configurati come ingressi.

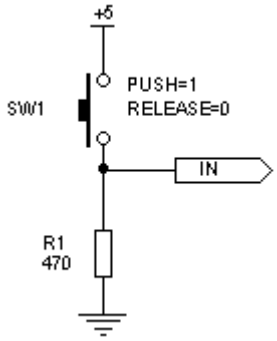
Ingressi digitali

Pulsante o interruttore (verso massa)



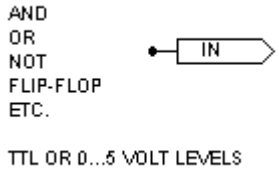
Il modo più semplice di collegare un pulsante o un interruttore ad ARDUINO è quello fra ingresso e massa (GND). Gli ingressi di ARDUINO sono provvisti di una resistenza interna che li mantiene a livello positivo. Perciò a pulsante rilasciato l'ingresso vale 1. Premendo il pulsante si collega l'ingresso a massa, perciò si leggerà uno 0.

Pulsante o interruttore (verso il positivo +5V)

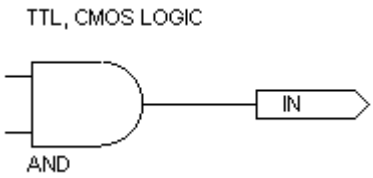


E' possibile collegare un pulsante anche verso il positivo; in tal caso è necessario contrastare l'azione della resistenza interna di *pull-up* che è incorporata in Arduino. La resistenza R1 da 470 ohm (collegata verso massa, cioè come *pull-down*) svolge proprio questa azione.

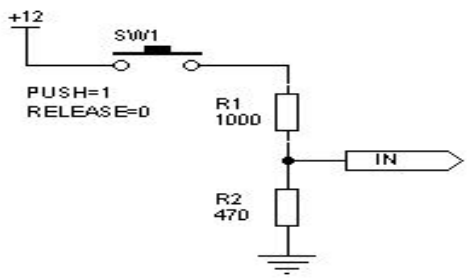
Porte logiche



Tutti gli ingressi possono essere collegati alle uscite di porte logiche TTL o porte logiche CMOS se alimentate a 5V. L'esempio a destra mostra come collegare l'uscita di una porta AND ad un ingresso di Arduino.



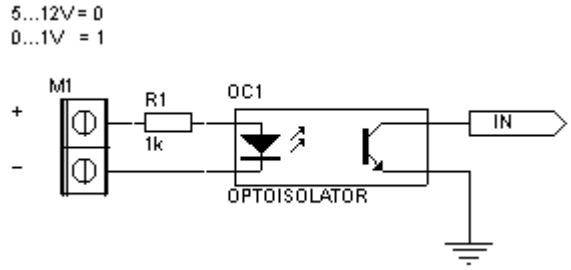
Ingressi a 12 volt (24 volt)



Se gli ingressi devono essere collegati ad un circuito che fornice in uscita una tensione (**continua**) più alta di 5 volt, la possiamo ridurre con un partitore come in figura.

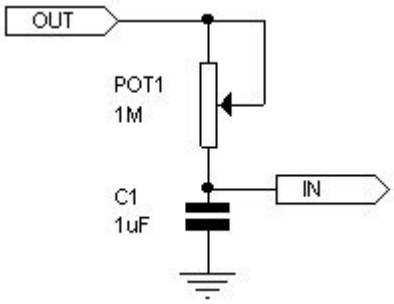
Questo circuito offre una buona **resistenza ai disturbi**, che vengono attenuati dal partitore di ingresso. E' possibile migliorare ulteriormente questa caratteristica aggiungendo un condensatore da **100 nF** in parallelo ad R2.
Per tensioni di **24 volt** utilizzare per R1 il valore di **2200** ohm.

Ingresso optoisolato



Quando è necessario garantire l'isolamento galvanico fra i segnali esterni ed il circuito di Arduino, si ricorre ad un optoisolatore. Con il morsetto scollegato o con una tensione minore di 1 volt il ARDUINOlegge 1. Quando il LED interno all'optoisolatore si illumina, il transistor conduce l'ingresso di Arduinoè condotto a GND: l'ingresso quindi vale zero.

Potenziometro (1)

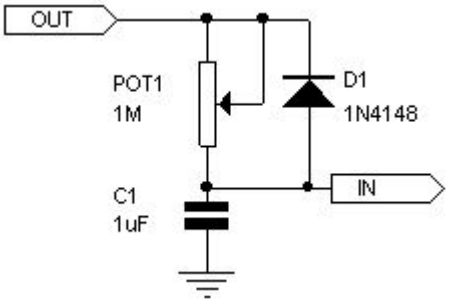


Usate questo circuito per:

- misurare una resistenza
- impostare tempi regolabili

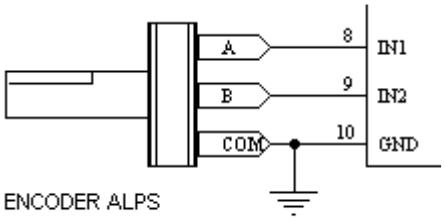
Funziona così:
si mette a zero l'uscita per un tempo lungo a sufficienza per scaricare del tutto il condensatore C1. Sarà necessario attendere almeno il doppio del massimo tempo impostabile.
Poi si rimette l'uscita a uno. L'ingresso tornerà a uno solo quando il potenziometro, caricando il condensatore, gli avrà fatto raggiungere almeno 2 volt.
Nota: se il circuito viene utilizzato con il cursore a zero (resistenza nulla) è consigliabile aggiungere una resistenza da 220 ohm in serie con il potenziometro per limitare la corrente attraverso il condensatore.

Potenziometro (2)

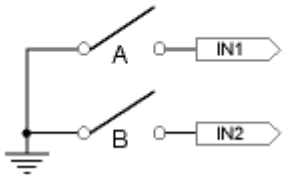


E' una versione migliorata del circuito precedente, in cui il condensatore viene scaricato quasi istantaneamente grazie al diodo D1. Il periodo di scarica, in cui si tiene bassa l'uscita, potrà essere ridotto al minimo. Per il resto funziona come il circuito precedente.

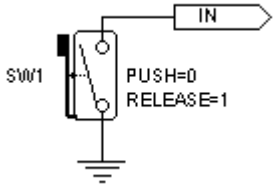
Encoder



Un encoder meccanico equivale ad una coppia di interruttori: i contatti di solito sono solo tre perchè al piedino COM è collegato un polo di entrambi gli interruttori. Alcuni modelli di encoder prevedono che uno o entrambi gli interruttori siano chiusi anche in condizioni di riposo.

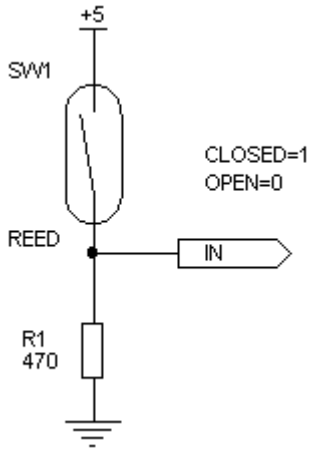


Finecorsa



Un finecorsa ("microswitch") non è altro che un tipo speciale di interruttore a deviatore. Lo si collega ad Arduinocome se fosse un pulsante, preferibilmente verso massa o verso il positivo. Se il filo di collegamento dovesse essere lungo, meglio usare il collegamento verso il positivo a 12 volt ed inserire un condensatore per eliminare eventuali disturbi captati dal filo.

Sensore magnetico (verso il positivo)

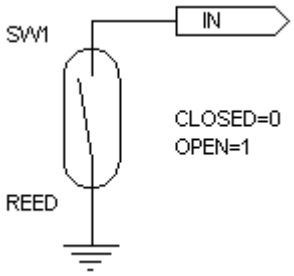


I sensori magnetici (usati spesso negli antifurto) sono dei piccoli interruttori che si aprono o si chiudono quando vengono avvicinati ad una calamita. I più diffusi sono quelli che restano chiusi a riposo (cioè quando la calamita è vicina). Con un contatto normalmente chiuso l'allarme scatta anche se si tenta di tagliare i fili, perchè il taglio apre il circuito interrompendo la corrente.

Un sensore magnetico si collega come un comune interruttore. Volendolo collegare al polo positivo, è necessario contrastare l'azione della resistenza interna di *pull-up* che è incorporata in Arduino. La resistenza R1 da 470 ohm (collegata verso massa, cioè come *pull-down*) svolge proprio questa azione.

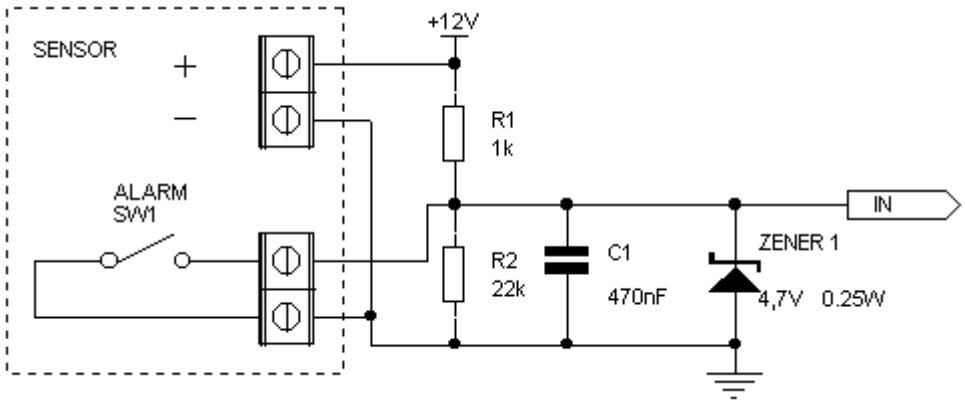
Se il filo di collegamento dovesse essere lungo, meglio usare il collegamento verso il positivo a 12 volt o quello per sensori antifurto ed inserire un condensatore per eliminare eventuali disturbi captati dal filo.

Sensore magnetico (verso GND)



Il modo più semplice di collegare un sensore magnetico ad Arduino è quello fra ingresso e massa (GND). Gli ingressi di Arduino sono provvisti di una resistenza interna che li mantiene a livello positivo. Perciò a contatto aperto l'ingresso vale 1. Chiudendo il contatto si collega l'ingresso a massa, perciò si leggerà uno 0.
Se utilizzate il sensore magnetico **in un antifurto**, allora è meglio usare il modello normalmente chiuso con il circuito che segue, una versione che migliora parecchio la tenuta ai disturbi captati dai fili.

Sensore antifurto



I sensori antifurto hanno questi morsetti:

- un'alimentazione a 12 volt (proveneinte dalla centrale)

- un contatto normalmente chiuso, che si apre quando il sensore scatta

talvolta un secondo contatto di sicurezza (tamper 24 ore) che si apre quando si apre il coperchio

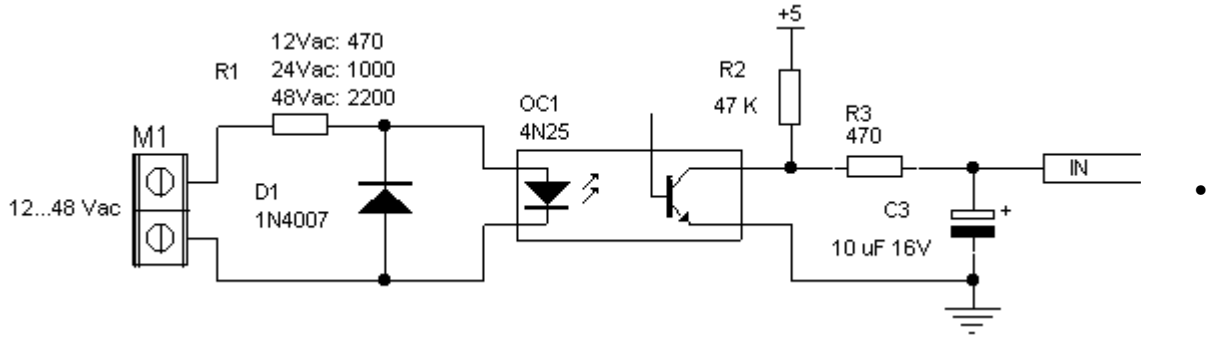
Con un alimentatore a doppia uscita (+12V per il sensore, +5V per il ARDUINO, stesso negativo per entrambi) questo schema permette di collegare e alimentare il sensore con solo tre fili. Inoltre la presenza di un generoso condensatore e del diodo zener spianano eventuali disturbi captati da fili di collegamento, che di solito sono piuttosto lunghi.

Ingresso in corrente alternata

A volte occorre prelevare segnali in corrente alternata:

- per "sentire" un campanello esistente: i trasformatori per campanelli sono di solito a 12 o a 24 volt
- per collegarsi al quadro elettrico di controllo di una macchina

per "sentire" finecorsa, galleggianti, teleruttori, microswitch esistenti

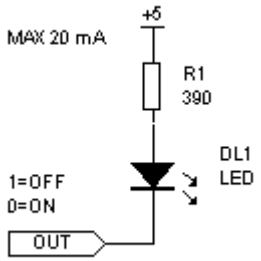


Questo schema è adatto per tensioni alternate da 12 fino a un massimo di 48 volt, cambiando il valore della resistenza R1.

Uscite

*I circuiti che seguono si applicano ai pin di ARDUINO configurati come uscite.Tipici dispositivi di uscita sono LED, relè, cicalini, ma non mancano neppure porte logiche o display a 7 segmenti. Per il collegamento di **lampade** o **motori** un relè è la soluzione più semplice e diretta*

LED (1)

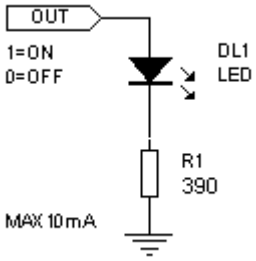


Questa è la configurazione che di solito si preferisce per pilotare dei LED con delle uscite digitali. Si tratta di un'eredità "storica", quando le uscite digitali erogavano più corrente quando erano a livello logico zero di quanto non riuscissero fare qundo erano a livello uno. Il LED si accende quando l'uscita è a zero (assorbimento di corrente) e si spegne quando l'uscita è a uno (corrente pressochè nulla).

Collegando il LED con il catodo (la parte fatta a freccia) verso il polo positivo la corrente scorre quando l'uscita si trova a zero. La resistenza

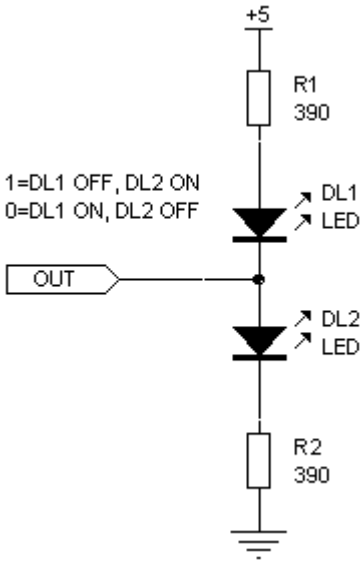
limita la corrente che scorre attraverso il LED e che transita sull'uscita. Con una resistenza da 390 ohm si ottiene una corrente di circa 10mA, un valore ottimale per il pilotaggio di un LED.

LED (2)



Se si preferisce che il LED si accenda quando l'uscita è a uno anzichè quando è a zero si collega il catodo del LED all'uscita e l'anodo a GND attraverso una resistenza. La resistenza sarà da 390 ohm perchè in configurazione positiva il ARDUINO può erogare al massimo 10 mA.

LED Doppio e LED bicolore



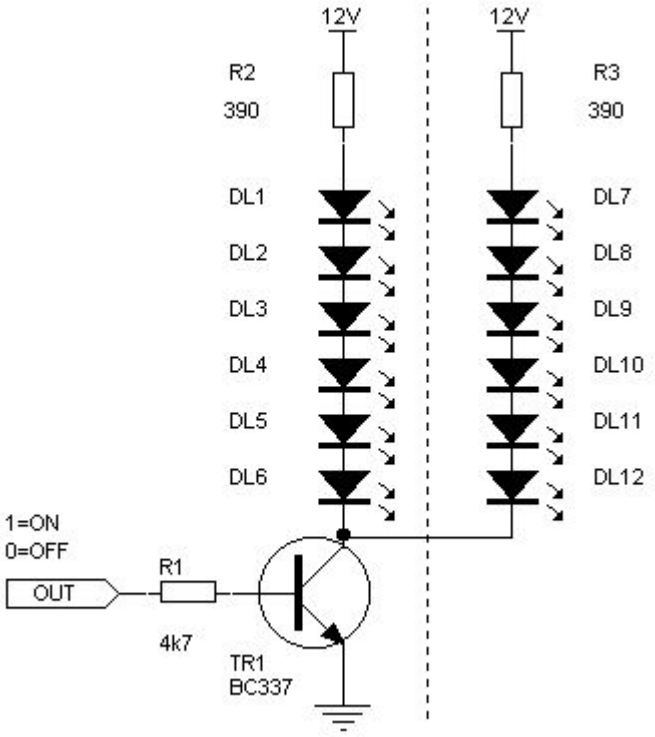
Con questo collegamento si hanno due LED collegati ad una sola uscita, in modalità tale che uno esclude l'altro. E' l'ideale per pilotare LED di colori diversi, ad esempio mettendo un LED verde per DL1 ed uno rosso per DL2 abbiamo che la luce cambia colore a seconda dello stato dell'uscita. Alcuni LED bicolore presenti in commercio contengono al loro interno una coppia di diodi LED collegati esattamente come DL1 e DL2, cioè con una coppia catodo-anodo in comune, e si possono usare per questo circuito.

Quando l'uscita è alta, il LED DL2 è acceso ed il LED DL1 è spento. Se invece l'uscita di Arduino diventa bassa, sarà il LED DL1 ad accendersi, mentre il LED DL2 si spegnerà.

Le resistenza sono entrambe da 390 ohm. La corrente totale sarà sempre di 10 mA circa (e non di 20 mA come si potrebbe a prima vista supporre sommando la corrente di R1 con quella di R2) perchè è acceso sempre e solo un LED per volta.

Nota: Un comportamento interessante di questo circuito è che se provate a collegare solo i LED e le resistenze, ma senza collegare il circuito al piedino di uscita di Arduino, allora vedrete accendersi entrambi i LED ad intensità ridotta!

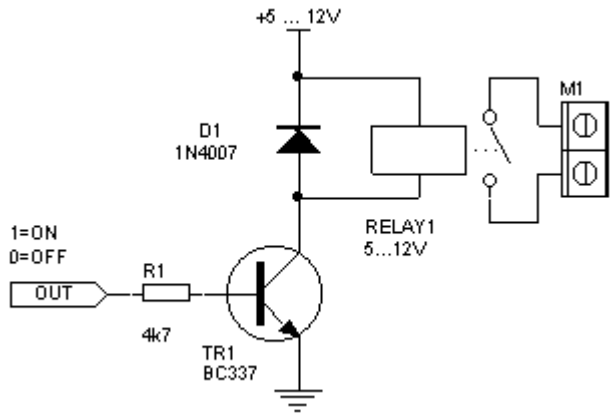
Gruppi di LED



Per realizzare giochi di luce, decorazioni natalizie, scritte lampeggianti, è necessario accendere molti LED per volta.
Questo circuito può pilotare fino a 12 LED per ogni uscita (potete però montare anche solo una delle due catene di 6 led separate dal tratteggio). Se una della catene di LED dovesse avere meno di 6 LED, la resistenza in serie dovrà essere aumentata rispetto agli attuali 390 ohm. in pratico la corrente che la attraversa dovà essere di circa 10 mA per non sovraccaricare i LED.

N. di LED	Resistenza
5	470
4	680
3	750
2	820
1	1000

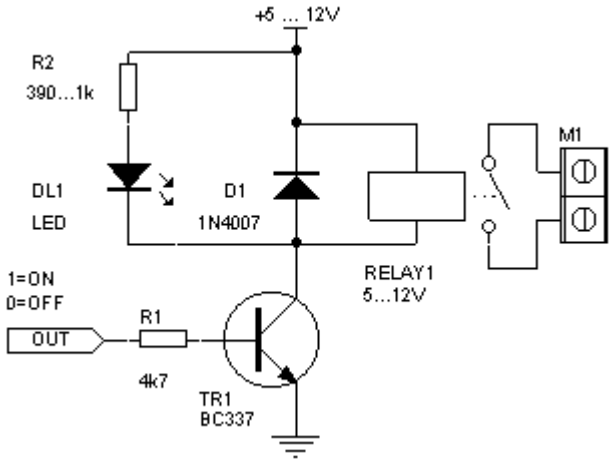
Relè



Assieme ai LED, i relè sono i dispositivi più usati assime ai ARDUINO. Tramite il relè il ARDUINO può comandare luci, motori elettrici, elettrovalvole, trasformatori, riscaldatori, frigoriferi, pompe, e mille altri dispositivi. Se il relè è da 5 volt, lo collegheremo alla stessa alimentazione di Arduino ; se invece ha una tensione maggiore (9 o 12 volt) ci servirà un alimentatore con doppia tensione. Non è necessario che l'uscita a 9 o 12 volt sia stabilizzata.

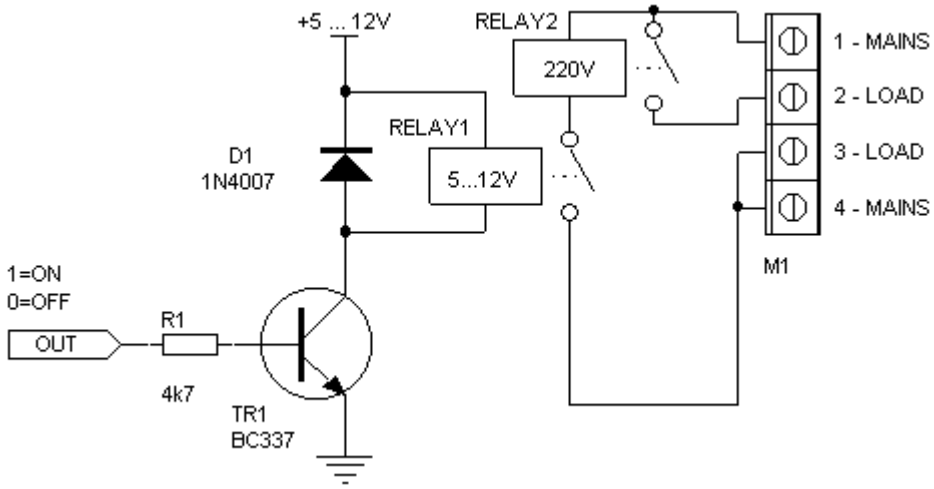
Il relè ha il grande vantaggio di isolare il carico esterno (collegato al morsetto M1) dal circuito elettronico di comando (ARDUINO).
Attenzione: siccome spesso il carico è alimentato con tensioni pericolose per l'uomo, è necessario porre la *massima attenzione* per evitare che i contatti del relè e del morsetto M1 entrino in contatto con altre parti del circuito.

Relè con indicazione a LED



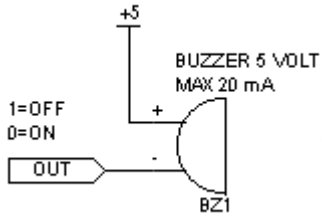
E' **molto comodo** avere un LED in modo che si accende quando il relè è eccitato. Utilizzando un relè da 5 volt montare una resistenza R2 da **390 ohm**. Per 9 volt utilizzare una **820 ohm** e per 12 volt una **1000 ohm**.

Relè di potenza a 220V



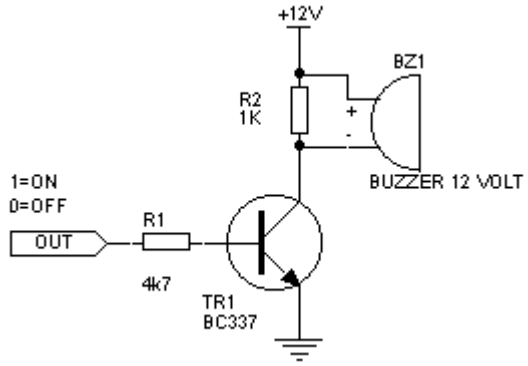
I relè di potenza a 220 volt (Relay 2 nello schema) ed i teleruttori si collegano tramite il contatto di un relè pilota di minor potenza (Relay 1). Se l'amperaggio di Relay 1 può non essere elevato, la tensione alternata applicabile ai suoi contatti deve essere maggiore di 250V. La rete (mains) si applica ai morsettii 1 e 4; il carico (ad esempio un motore elettrico o una lampada) invece ai morsetti 2 e 3. **Pericolo:** fate la massima attenzione nell'effettuare questi cablaggi per via della presenza della tensione di rete a 220 volt.

Buzzer a 5 volt



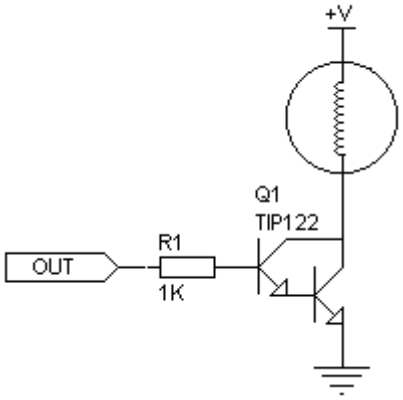
Un cicalino a 5 volt può essere pilotato direttamente da una delle uscite di Arduino. Bisogna sempre scegliere i cicalini di tipo elettronico, in quanto i cicalini elettromeccanici assorbono molto più dei 40 mA massimi disponibili.
La maggior parte dei cicalini a 9 volt funziona ugualmente bene a 5 volt, sebbene ad intensità ridotta.

Buzzer a 12 volt e buzzer di potenza



I buzzer a 12 volt sono i più diffusi. A volte il livello sonoro è tale da potere parlare di vere e proprie minisirene. Il circuito illustrato è adatto per buzzer fino a 100 mA di assorbimento. Per dispositivi più potenti prendete in considerazione l'utilizzo di un relè.

Lampade a bassa tensione

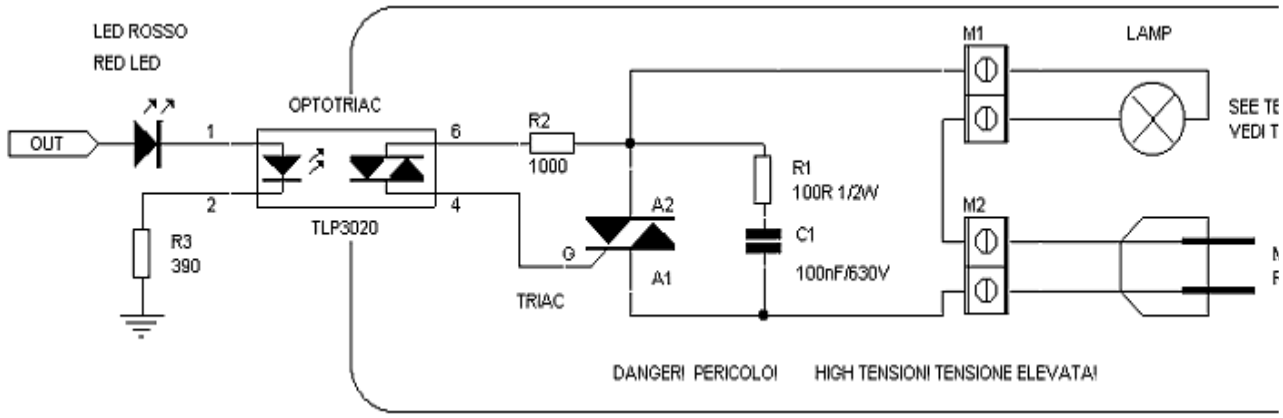


Le lampade da 3 fino a 24 volt funzionanti in corrente continua si pilotano tramite un transistor darlington di potenza (il simbolo è quello di due transistor!), come il TIP122. La tensione +V è quella specificata per le lampade, e non deve superare i 40 volt.

Suggerimento:se si pilota l'uscita con dei veloci inpulsi on/off si può modulare l'intensità luminosa della lampada.

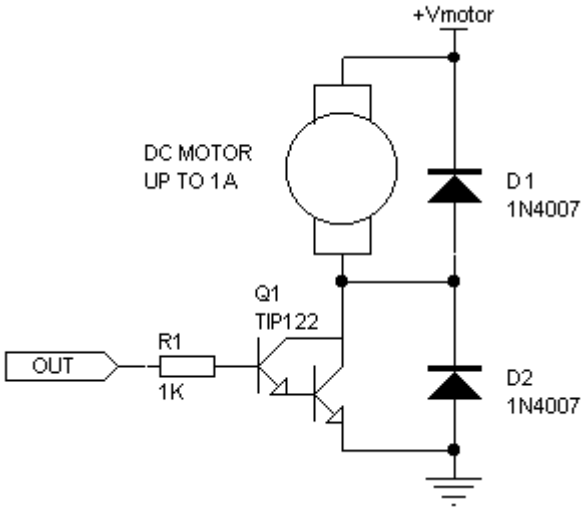
Lampade a 220V con optoTRIAC

Nei giochi di luce si utilizzano quasi sempre i TRIAC. Un relè infatti non è indicato per azionarsi più volte in un secondo, e a lungo andare si romperebbe.
Questo schema utilizza un optoTRIAC per isolare il circuito di Arduino dalla parte di circuito soggetto alla tensione di rete (tutta la zona con l'indicazione di "PERICOLO!"). Un secondo TRIAC, eccitato dall'optoTRIAC, pilota la lampada.
Questo circuito si può usare anche per lampade a corrente alternata a 12 e 24 V (come le lampade alogene), ma NON per pilotare motori, trasformatori, lampade al neon.



Normalmente si usano TRIAC da almeno 600V e 5A, con i quali si potrà pilotare un carico massimo di 200W a 220Vca. Per carichi più grandi sarà necessario utilizzare un TRIAC a maggior corrente e montarlo su una aletta di raffreddamento **accuratamente isolata.**

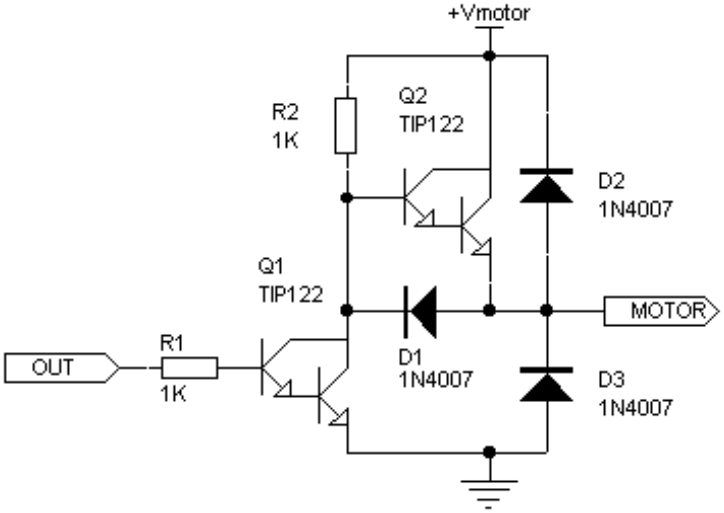
Motori ON/OFF



Basta un transistor di tipo Darlington per pilotare un motore in corrente continua fino a 48 Volt. Se l'assorbimento è minore di 600 mA, non servirà l'aletta di raffreddamento. Il motore si aziona se l'uscita = 1, si spegne se l'uscita = 0.

Nota: il simbolo di un transistor Darlington assomiglia a due transistor in cascata, ma si tratta di un solo componente.

Motori: schema base (mezzo ponte)

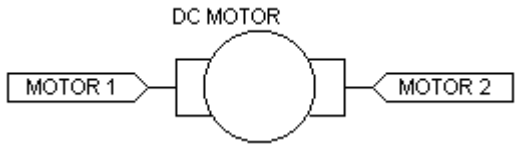


Questo schema, più elaborato, è la base per pilotaggi più sofisticati:

- motori in continua con frenatura
- motori in continua con inversione
- motori passo-passo

Per ogni azionamento serviranno uno più di questi circuiti, ciascuno pilotato da una differente uscita di Arduino.
Questo circuito viene usualmente chiamato *mezzo ponte H.*

Motori avanti/stop/indietro

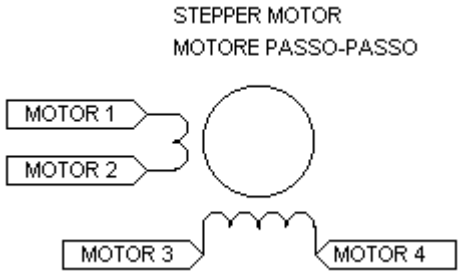


Per invertire il senso di marcia di un motore DC serve un ponte H, che si realizza con due circuiti a mezzo ponte illustrati sopra. Ogni circuito fornisce uno dei due fili MOTOR. Due uscite di Arduino sono necessarie per pilotare le due metà del ponte. Le combinationi delle uscite danno luogo ai seguenti azionamenti:

motor 1	*motor 2*	*movimento*
0	*0*	*stop*
0	*1*	*avanti*

1	*0*	*indietro*
1	*1*	*stop*

Motori passo-passo

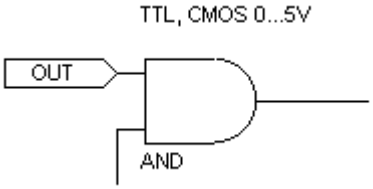


Questo circuito è adatto per stepper bipolari a 4 fili, che richiede 4 circuiti mezzo ponte. Ecco la sequenza di pilotaggio:

motor1	*motor 2*	*motor 3*	*motor 4*
1	*0*	*0*	*0*
0	*0*	*1*	*0*
0	*1*	*0*	*0*
0	*0*	*0*	*1*

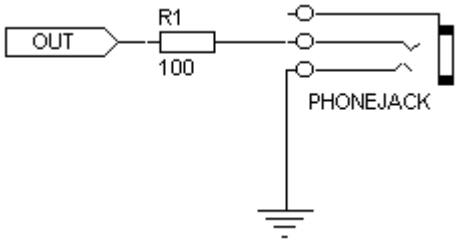
Per invertire la rotazione, basta eseguire la sequenza al contrario

Porte logiche



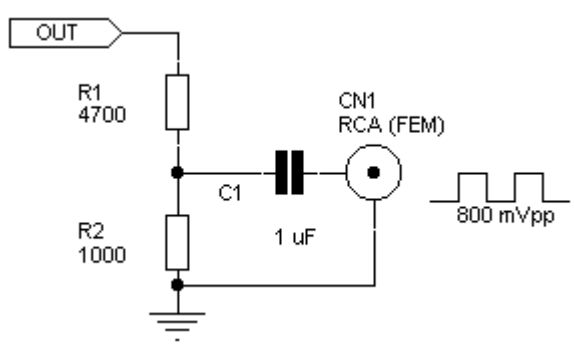
Tutti le uscite possono pilotare gli ingressi di porte logiche TTL o porte logiche CMOS se alimentate a 5V.
Questo esempio mostra come collegare ARDUINO all'ingresso di una porta AND. Se il circuito logico è esterno, ricordate di collegare in comune i GND.

Cuffie stereo



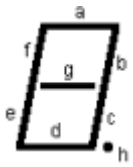
Questo circuito pilota le comuni cuffiette per walkman. Notate che il collegamento più esterno è lasciato libero. In questo modo i due padiglioni sono in serie fra di loro e suoneranno entrambi.

Amplificatore audio



Questo circuito inietta il segnale nell'ingresso ausiliario degli impianti stereo. Se il vostro amplificatore è sprovvisto di questa entrate, potete utilizzare quella per un CD, una piastra (TAPE) o un sintonizzaotore (TUNER) esterni. Alla presa RCA collegate l'apposito cavetto reperibile nei negozi di Hi-Fi. Lo schema si riferisce ad un solo canale (LEFT o RIGHT), ma potete collegarli entrambi mettendo una seconda presa RCA in parallelo alla prima.

Display LED

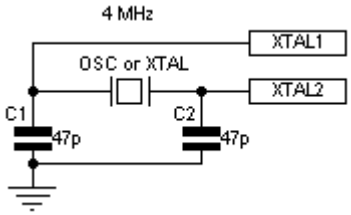


Un display costituito da 8 LED, ognuno dei quali accende un diverso "segmento" della cifra. I segmenti si indicano con le lettere a,b,c,d,e,f,g, mentre h corrisponde al puntino. Per diminuire il numero dei piedini, tutti i catodi (o gli anodi) sono connessi allo stesso pin: parliamo perciò di display a "catodo comune" (oppure "anodo comune"). Sebbene i segmenti si possano illluminare singolarmente allo stesso modo di un LED (basta mettere una resistenza in serie), di solito si utilizza un integrato apposito di decodifica, come ad esempio il CD4511 .

Oscillatori

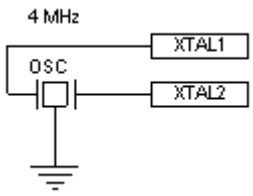
Il ARDUINO necessita di un clock 16 MHz per funzionare. L'oscillatore contenuto nel chip richiede un quarzo od un risuonatore ceramico esterni per funzionare.

Oscillatore al quarzo / Oscillatore con risuonatore ceramico a 2 piedini



Quarzi ed oscillatori ceramici a 2 piedini si collegano allo stesso modo. I quarzi sono indispensabili per costruire timer per lunghi periodi sufficientemente precisi.
I condensatori impediscono che l'oscillatore si inneschi ad una frequenza più alta dei 4 MHz richiesti.
Può capitare di imbattersi in un quarzo che richieda condensatori di 100pF.
I quarzi hanno un tempo di partenza più lungo degli oscillatori ceramici, perciò quando si utilizza il quarzo occorre sempre prevedere un circuito di reset RC o con l'apposito integrato.

Oscillatore con risuonatore ceramico a 3 piedini



Il risuonatore da a tre piedini non richiede componenti supplementari per oscillare.
Il piedino da connettere a massa è quello centrale.

La precisione della frequenza è solitamente migliore dell'1%. Per i brevi periodi questo errore è trascurabile, ma per periodi lunghi e ripetitivi diventa sensibile. Un orologio che avanza di un minuto ogni cento accumula un errore di più di tre giorni in un anno!

Modulo oscillatore /oscillatore esterno

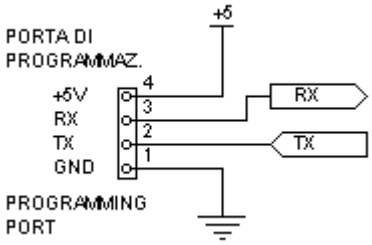
Se disponete di moduli oscillatori al quarzo (si trovano talvolta su schede di recupero) potete utlizzarli collegandone l'uscita al piedino XTAL1 di Arduino. Il pin XTAL2 resta scollegato.

Allo stesso modo potete utilizzare il segnale da una sorgente di clock esterna.

Interfaccia per PC

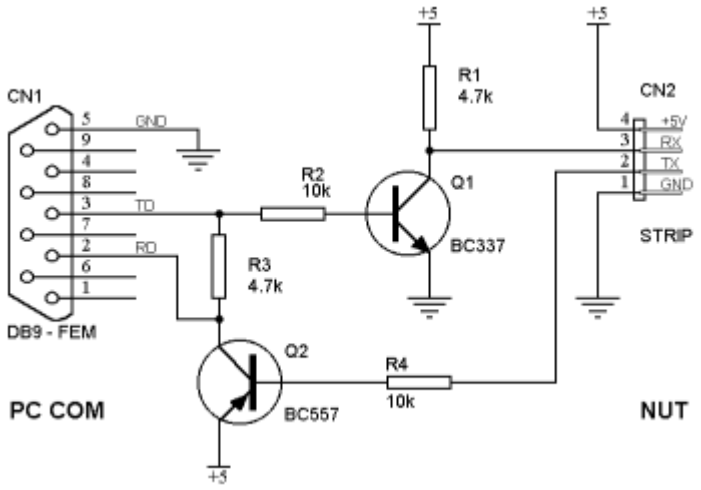
Il ARDUINO dialoga col PC tramite i piedini RX e TX. Nella basetta originale e' incorporato un convertitore USB-Seriale. Il protocollo seriale e' 111500 bps, N, 8, 1. Se vogliamo riprogrammare un chip Arduino montato su un circuito stand-alone, dovremo prevedere il connettore per l'interfaccia PC oppure un'interfaccia PC completa (si tratta di 2 soli transistor) sulla scheda stessa.

Connettore di programmazione



Su questo connettore si innesta l'interfaccia a transistor per il PC. Il connettore accoglie una strip maschio passo 2,54 mm che è montata sull'interfaccia. E' quindi possibile montare un connettore box femmina a 4 poli, una strip femmina o addirittura metà di uno zoccolo per integrati da 4+4 pin.

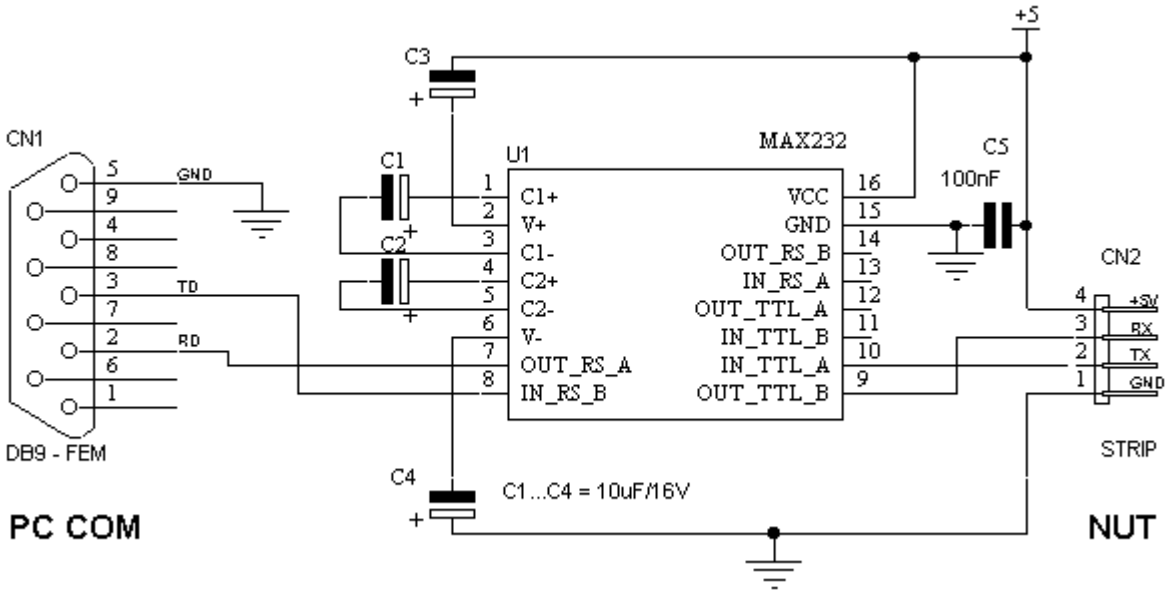
Interfaccia a transistor



E' l'interfaccia più semplice ed economica. Si può utilizzare praticamente qualsiasi transistor PNP ed NPN per piccoli segnali.
Il connettore CN2 è una strip passo 2,54 che si innesta nel *connettore di programmazione*. Se il circuito viene utilizzato **senza** che il PC sia connesso (come accade se si incorpora l'interfaccia nel circuito finale), è consigliabile porre una resistenza da 10k fra la base del transistor Q1 e la massa.

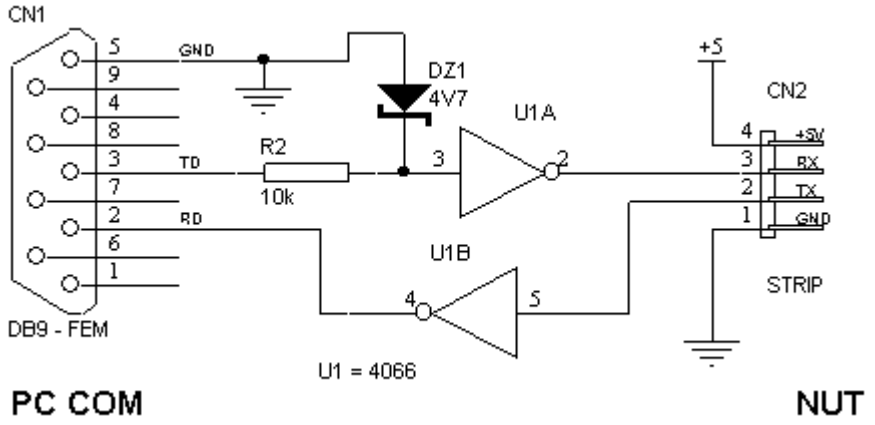
Interfaccia alternativa: con MAX232

Questa interfaccia utilizza un integrato apposito e fornisce livelli di uscita **standard** adatti per ogni tipo di PC. Trattandosi di un integrato specifico, ha anche la migliore reiezione ai distrurbi e la protezione contro scariche e corto circuiti.Per contro è una soluzione più costosa e ingombrante delle soluzioni che utilizzano i transistor o semplici porte logiche.

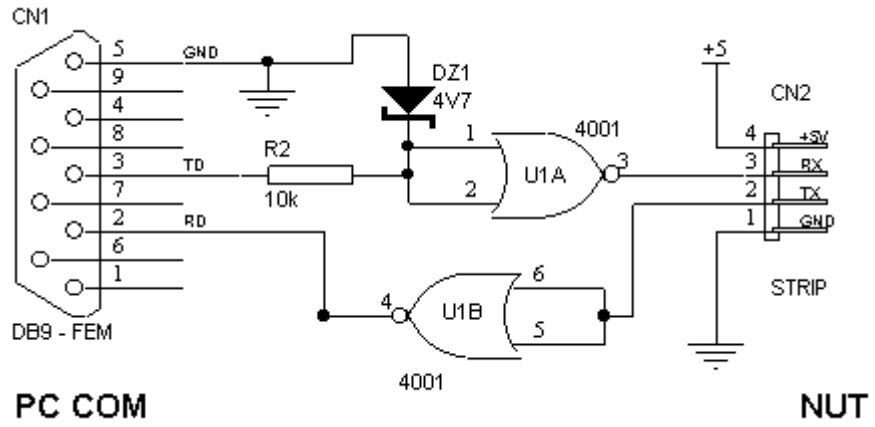


Interfaccia alternativa: con porte logiche

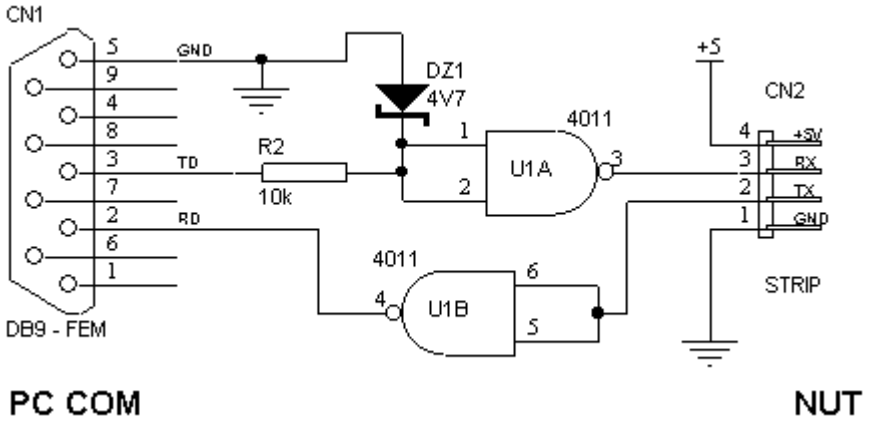
Se nel vostro circuito ci sono alcune porte logiche d'avanzo, potete riutilizzarle come in questo esempio che usa due dei sei inverter di un 4066:



Utilizzando integrati CMOS della serie 4000, scegliete sempre le versioni B *(buffered)* che offrono la maggiore corrente di uscita. Qualsiasi inverter va bene, anche ricavato da porte logiche come NAND e NOR negli esempio che seguono che utilizzano due delle quattro porte di un 4001 o di un 4011:



Non dimenticate di alimentare gli integrati logici collegando gli opportuni piedini al +5V e a GND!



Di tutti i circuiti presentati, questo è il più spartano. Non eroga la tensione negativa richiesta dallo standard, ma soltanto la necessaria l'**inversione** logica. E' un dato di fatto fatto che la maggioranza dei PC funziona ugualmente bene con un segnale del genere, ma ricordate che si tratta di una piccola forzatura! Perciò **non** è consigliabile usarlo per progetti dove sia richiesta la massima affidabilità e flessibilità, dove i cavi di collegamento siano lunghi e se non conoscete a priori quale PC verrà usato per la programmazione.

Interfaccia incorporata nella scheda o esterna?

Non c'è una regola valida per tutti.

- Se il vostro circuito deve eseguire una funzione che **non dovrà essere più modificata**, il chip potrà essere programmato **prima** del montaggio sul circuito. Potete quindi semplificare il circuito finale eliminando qualsiasi genere di interfaccia per il PC.

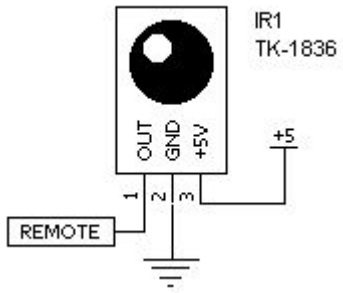
- Se sarete sempre voi a programmare il circuito, allora potete realizzare **una sola interfaccia esterna** da utilizzare per **tutti** i circuiti, collegandola al connettore **STRIP** consigliato (potrebbe bastare anche soltanto metà di uno zoccolo per integrati) che prevederete sulla scheda.
  Questa è la **soluzione preferita dagli hobbysti** perchè consente di sperimentare nuove soluzioni senza togliere il chip dal circuito.

- Se invece il circuito dovrà essere programmato da altre persone e con un PC che non sia noto a priori, allora è consigliabile montare **tutta l'interfaccia** sulla scheda ed uscire con il connettore a **9 poli standard** per porta seriale. In questo caso, potrete utilizzare qualsiasi PC per riprogrammare il chip: basterà un **comune cavo seriale**, reperibile ovunque.
  Questa è la soluzione **scelta dai professionisti** quando vogliono che si il loro cliente ad effettuare l'aggiornamento del chip.

- Ovviamente potrete anche decidere di montare il chip su zoccolo e di sfilarlo ogni volta che volete programmarlo: questa soluzione è **obbligatoria** in quei casi dove sul circuito in funzione siano presenti **tensioni pericolose**, come nelle accensioni a triac non isolate.

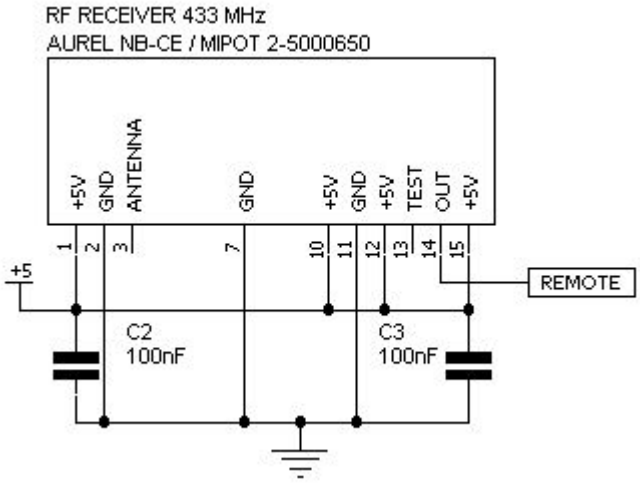
Ricevitori per telecomando

Ricevitore telecomando a infrarossi



La figura si riferisce al ricevitore TSOP1836, marcato TK1836, di produzione Temic/Telefunken/Vishay. Altri ricevitori utilizzabili sono SFH5110 (Siemens, Sharp) LT1059 (Liteon), che però hanno una piedinatura **diversa**.

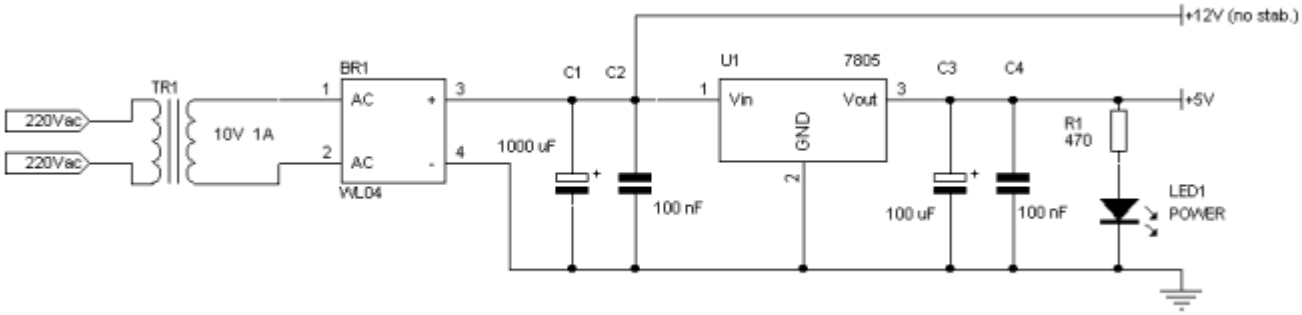
Ricevitore telecomando a radiofrequenza



I migliori risultati si ottengono con i moduli ricevitori Aurel NB-CE, RX-4M50RR30SF, RX-4M50SA60SF, ma il modulo MIPOT 2-5000650 è più economico. Se vogliamo riutilizzare il modulo potremo inserirlo in una lunga strip femmina. Il pin 5 è presente solo sul modulo Aurel, il 12 solo sul Mipot. Al pin ANTENNA potremo collegare uno spezzone di filo rigido di 20...40 cm di lunghezza.

Alimentatori

Alimentatore 5V + 12V



Questo alimentatore fornisce, oltre alla tensione di 5 volt stabilizzata per ARDUINO, anche una tensione non stabilizzata di 12 volt per alimentare ad esempio i relè collegati come dall'apposito schema (vedi).
Se non usate l'uscita a 12 volt allora potrete utilizzare qualsiasi trasformatore con uscita fra 7,5 e 14 volt invece di quello da 10V indicato sullo schema . L'integrato U1 va raffreddato con un'aletta.
Prestate la massima attenzione ai collegamenti del trasformatore, che funziona a 220 volt e quindi sottoposto a tensioni potenzialmente letali. E' consigliabile proteggere il circuito con un fusibile ed un interruttore.

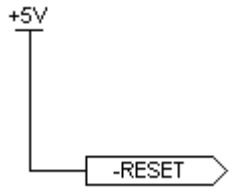
Alimentatore recuperato da PC

I mobili per PC usati o di vecchio tipo sono venduti a prezzi convenienti, e quasi sempre comprendono l'alimentatore originale. E' un alimentatore **molto potente**, capace di erogare decine di ampere, e che fornisce le tensioni di +5 volt, +12 volt (ed anche -12 volt e - 5 volt). Cercate possibilmente un modello con interruttore incorporato. Normalmente i cavi che escono sono tutti in parallelo, il cavo nero a massa, il cavo rosso a +5V ed il cavo giallo a +12V (ma controllate sull'etichetta che questi collegamenti corrispondano). Se non avete problemi di spazio, utilizzate il mobile per intero.

Reset

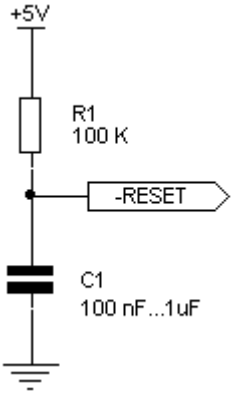
Avere un buon RESET è importante. Se un RESET improprio impedisce al chip di partire, le anomalie che possono presentarsi durante il funzionamento possono bloccare il chip se non sono accompagnate da un opportuno impulso di RESET.
Un tipico esempio è il *brown-out*, cioè quando la tensione di alimentazione scende sotto il livello minimo di funzionamento pur non andando a zero. Durante il *brown-out* il RESET deve essere tenuto a zero, altrimenti il chip può perdere i controllo, bloccarsi, e addirittura richiedere una nuova programmazione.

Reset sperimentale



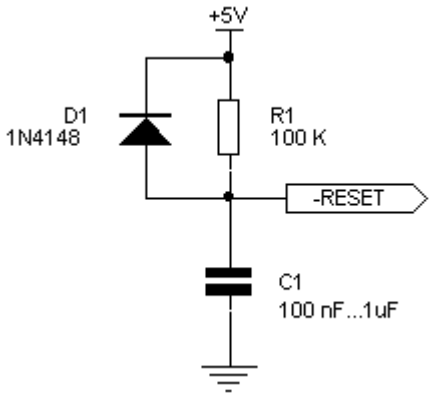
Per rapidi prototipi e per progetti che abbiano funzione di gioco e non debbano essere usati ininterrottamente per lunghi periodi, il piedino di RESET può essere semplicemente portato al positivo. Per farlo potete usare anche una resistenza da 100 kohm per sfruttare la piccola capcità parassita sempre pesente sul piedino.

Reset RC



Una resistenza ed una capacità esterna costituiscono un circuito facilmente realizzabile ed assieme efficace. Il diodo (opzionale) aiuta il condensatore a scaricarsi rapidamente in caso di un "buco" di tensione. Infatti la condizione di instabilità più pericolosa è quando il RESET e l'alimentazione non sono nè alti nè bassi.

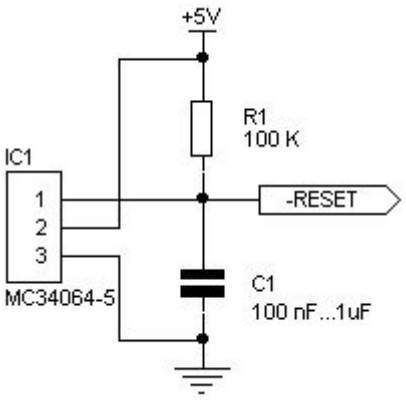
Reset RCD

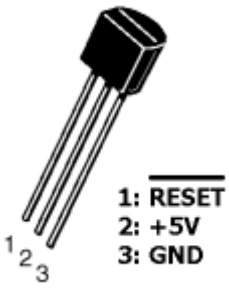


E' una versione migliorata del circuito precedente, che funziona meglio in presenza di interruzioni (o micro-interruzioni) dell'allimentazione. Il diodo D1 infatti provvede a scaricare velocemente la capacità C1 appena la tensione di alimentazione scende.

Reset a integrato

Un'integrato appositamente progettato per il RESET è ovviamente la soluzione migliore. L'MC34064 inoltre interviene con precisione scaricando il condensatore quando la tensione si abbassa, anche di poco, sotto i 4,5 volt. Quando la tensione di alimentazione non è ancora stabile e quindi sotto i 4,5 volt (succede tutte le volte che accendiamo o spegniamo l'apparecchio, ed ad ogni black-out) possiamo essere certi che il piedino di RESET viene mantenuto basso.

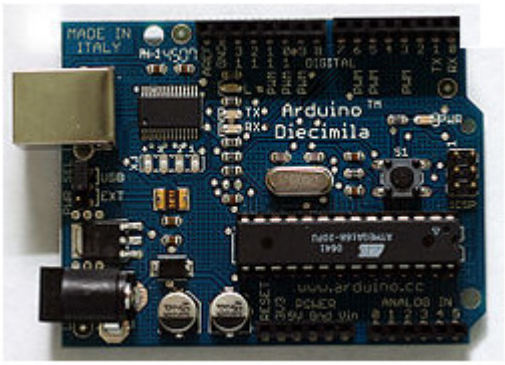




VeroBoArduino, costruire una board compatibile con Arduino a basso costo

11 Lug 2011

- Microcontrollori
- Arduino
- fai da te
- microcontroller



Avete creato il vostro circuito partendo da Arduino su una breadbord ed ora volete renderlo permanente ma così facendo dovrete prendere un nuovo Arduino per i prossimi esperimenti. Allora vediamo come creare una board compatibile con Arduino di modo da non dover ricomprare un nuovo Arduino ogni volta.

Utilizzare Arduino per i propri "esperimenti" elettronici è oggi una pratica largamente diffusa peccato che ogni qual volta siamo sodisfatti del risultato per ottenere il circuito permanente dobbiamo "sacrificare" Arduino. Vediamo allora come ovviare al problema in modo economico.

Il materiale necessario è il seguente:

- Un microcontrollore ATMega con il bootloader per Arduino preinstallato (semplifica il tutto)

- Una stripboard come base per il circuito permanente con almeno 23x14 fori

- Un LM7805

- 2 condensatori elettrolitici a 25V da 100 micro Farad (C1,C2 nello schematico)

- 2 condensatori ceramico da 100 nano Farad (C3,C4 nello schematico)

- 2 condensatori ceramici a disco (C5,C6 nello schematico)

- un socket Dual in-line (DIL) con 28 pin (o equivalente adatto per l'ATMega)

- un connetore femmina a 6 pin

- switch per attivare reset del controllore

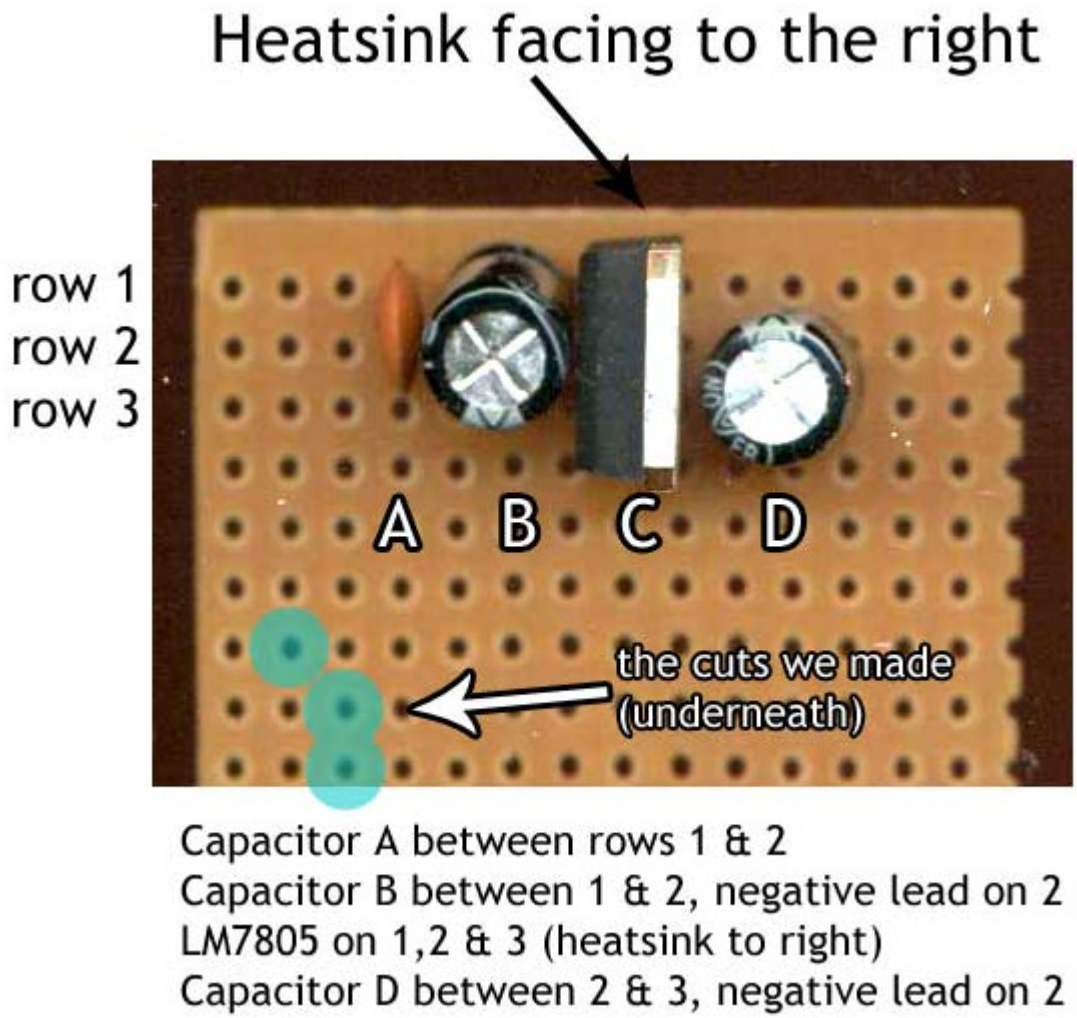
- due resistenze da 10K Ohm (marrone,nero,arancione) (R1,R2 nello schematico)

- 2 cavetti isolati di collegamento in rame o allminio

- quarzo da 16MHz (X1 nello schematico)

- cavo USB - seriale

Partiamo dalla realizzazione del sistema di alimentazione. Come per Arduino vorremmo poter alimentare la board con un batteria od altra sorgente tra 7 e 12 V. Usando i tre condensatori e l'LM7805 effettuare i collegamenti sulla striboard riportati in figura sottostante.



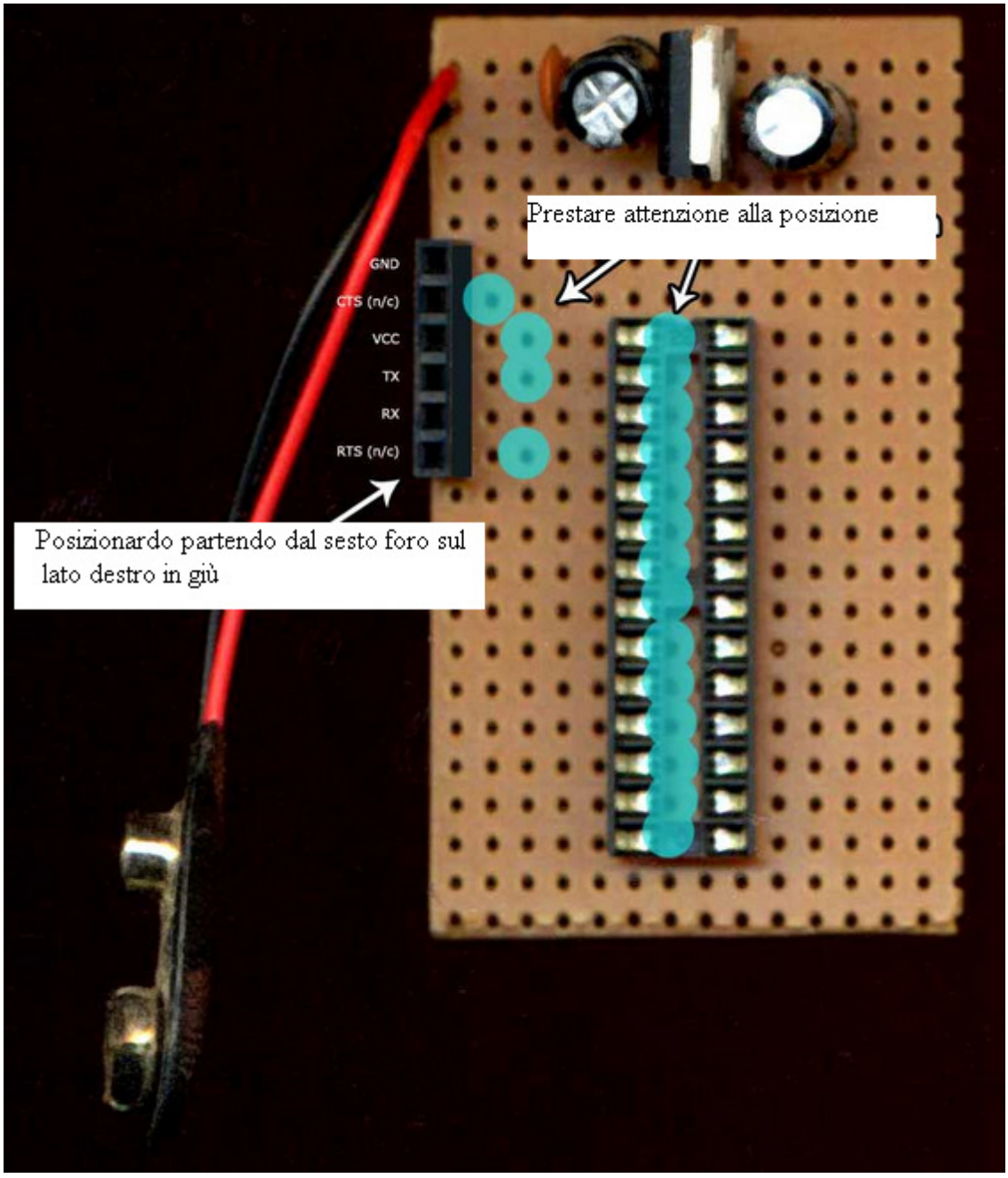
Prestare molta attenzione alla polarità dei condensatori seguendo le istruzioni in figura sopra.

L'alimentazione andrà applicata alle righe (row) 1 (positiva) e 2 (negativa). Se tutto è andato bene sulla riga 2 avremo l'uscita in tensione negativa e sulla riga tre l'uscita positiva (5V). Fare molta ATTENZIONE a non invertire la polarità dell'alimentazione, ciò causerebbe la rottura del regolatore di tensione LM7805. Si potrebbe aggiungere un diodo per prevenire questo problema ed un fusibile per prevenire sovra alimentazioni ma tali accortezze non sono lo scopo di questo articolo.

Se disponete di un multimetro o di un voltmetro potete verificare il corretto funzionamento del sistema di alimentazione misurando la tensione di input (alimentazione), provare con una batteria da 9V per esempio, e l'output atteso (tra riga 2 e 3) sarà circa 5V.

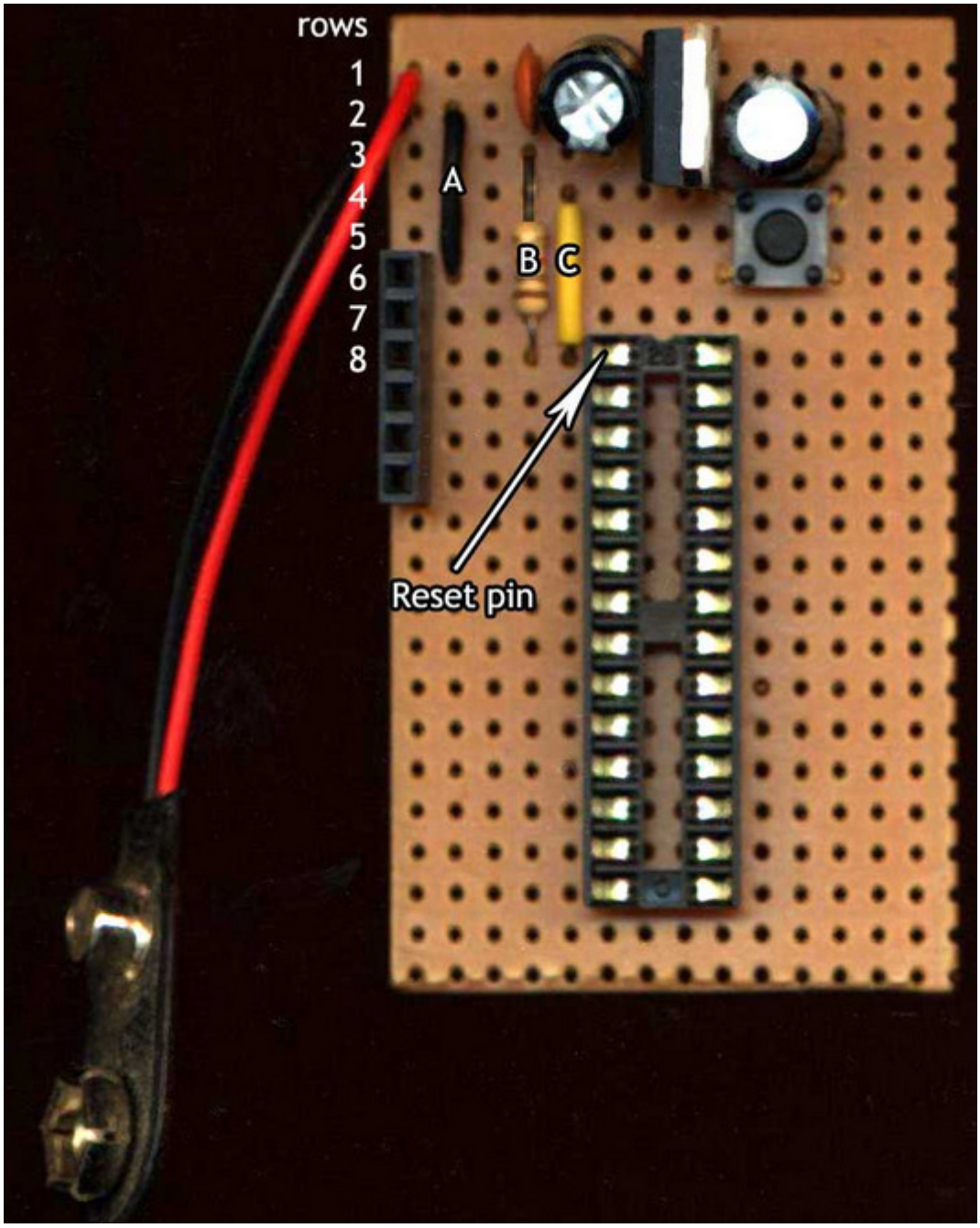
Passiamo al posizionamento del controllore...

Saldare il socket DIL ed il connottore a 6 pin come mostrato in figura sottostante con la stessa posizione sulla stripboard



Il connettore seriale femmina sarà usato per programmare l'ATMega tramite la IDE di Arduino.

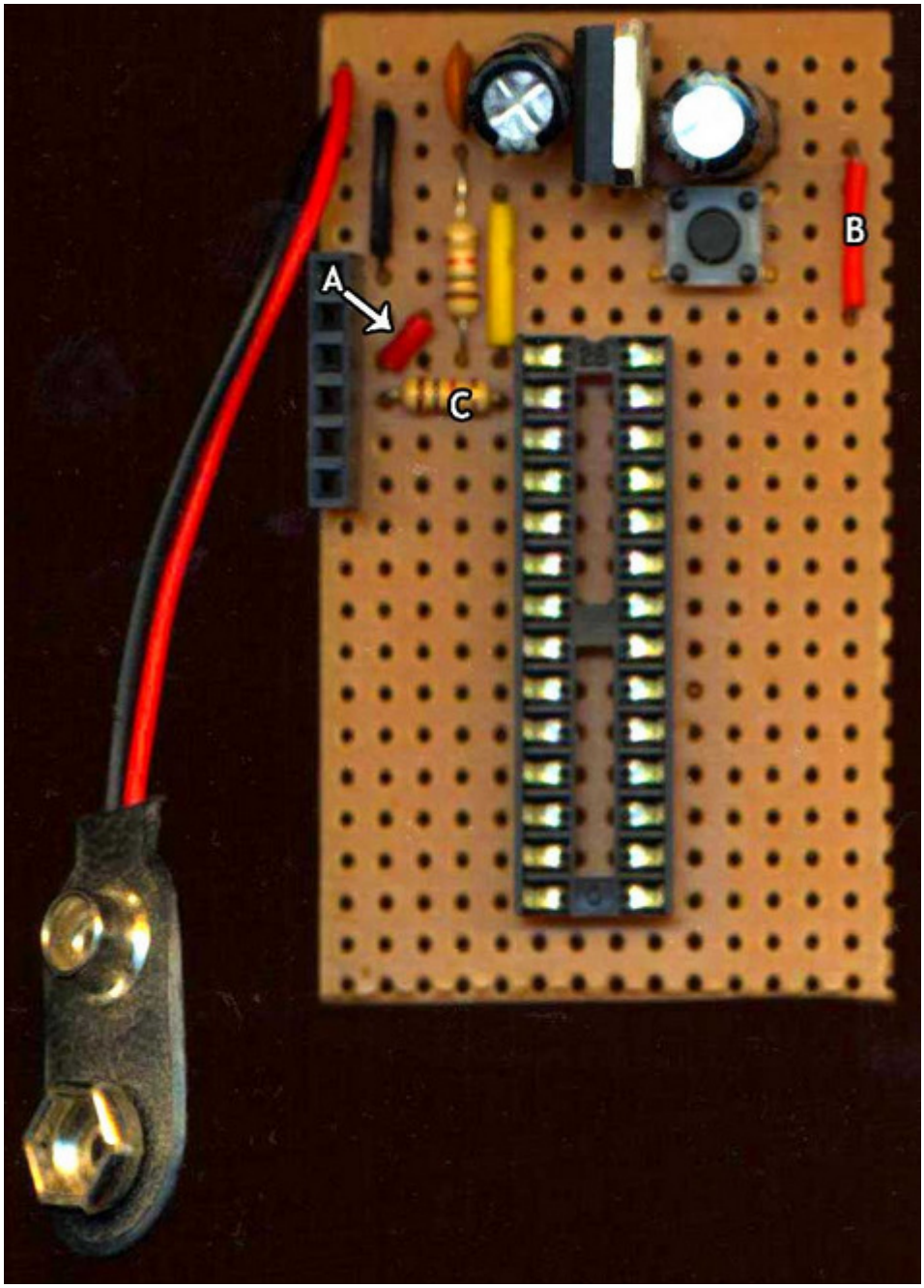
Creiamo adesso il circuito per resettare l'ATMega. Saldare la resistenza tra riga 3 (+5V) e 8 (per mantenere lo stato di reset del ATMega alto) e lo switch di modo che colleghi la riga 6 con la 2 quando premuto. Vedere figura sottostante.



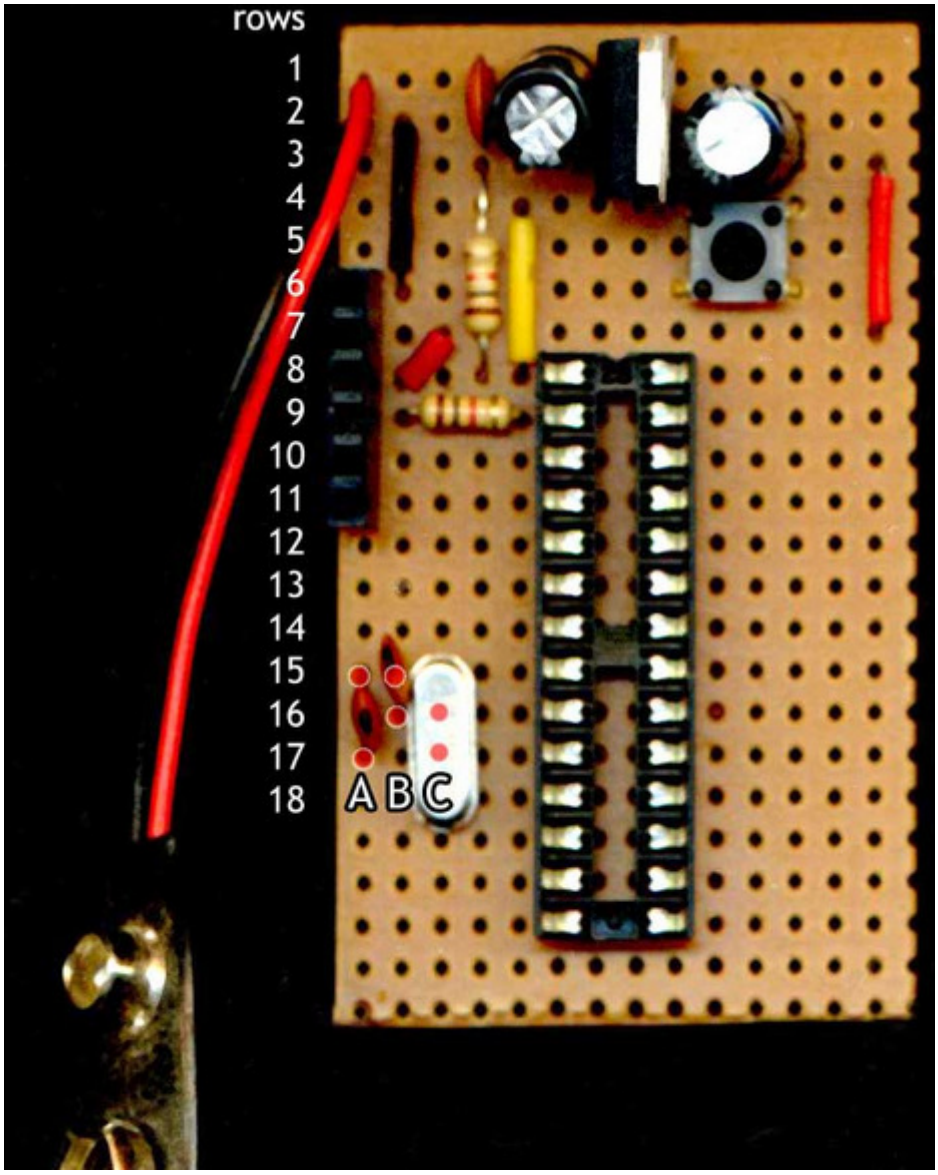
Collegando con un cacetto riga 4 e 8, agendo sullo switch si esclude la resistenza collegata a +5V resettando l'ATMega.

Adesso colleghiamo il connettore seriale che sarà usato per caricare i programmi (sketches) sull'ATMega con la IDE di Arduino.

Dobbiamo collegare l'ingresso di tensione del connettore VCC, l'ingresso di trasmissione TX al pin RX del ATMega ed al contrario per RX del connettore. Usando dei cavetti ed una resistenza i collegamenti da effettuare sono illustrati nella figura sottostante.



Aggiungiamo un oscillatore al quarzo necessario al'ATMega per scandire gli intervalli di tempo. Saldare quindi l'oscillatore al quarzo ed i due condensatori ceramici come in figura sottostante.



Collegare:

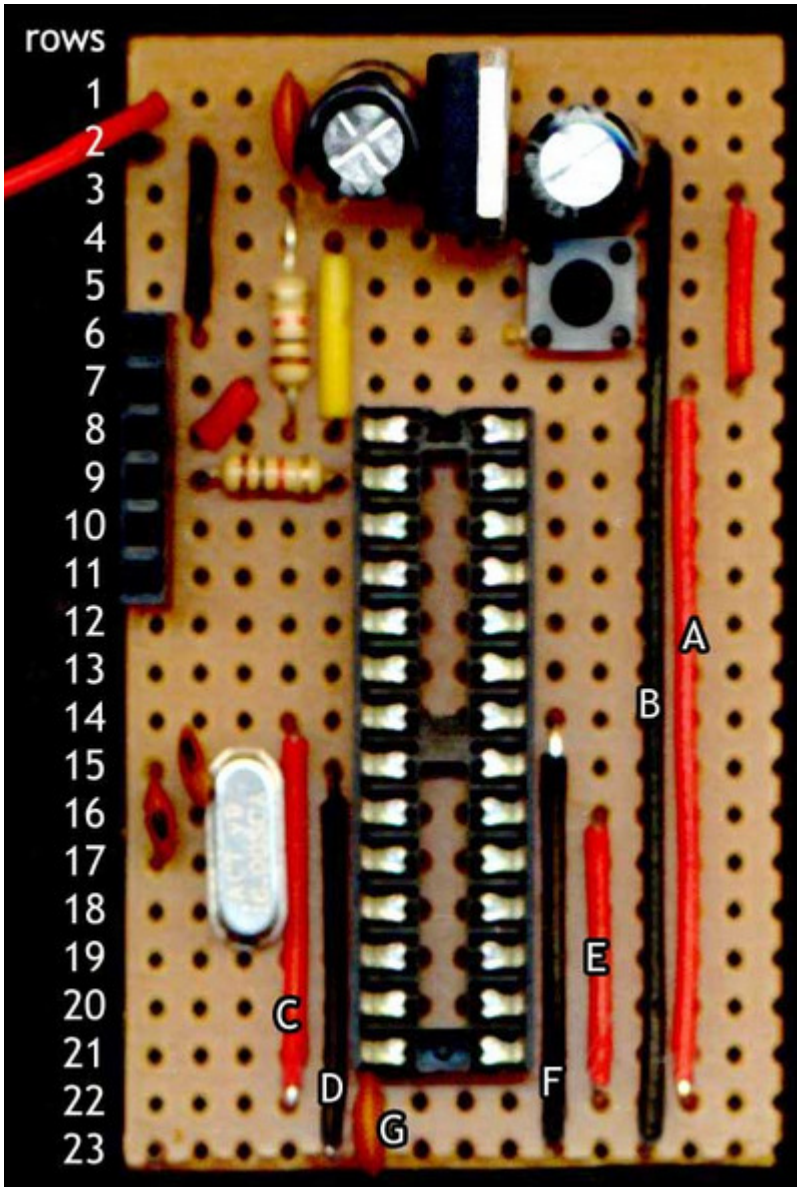
- condensatore A da riga 15 a 17

- condensatore B da riga 15 a 16

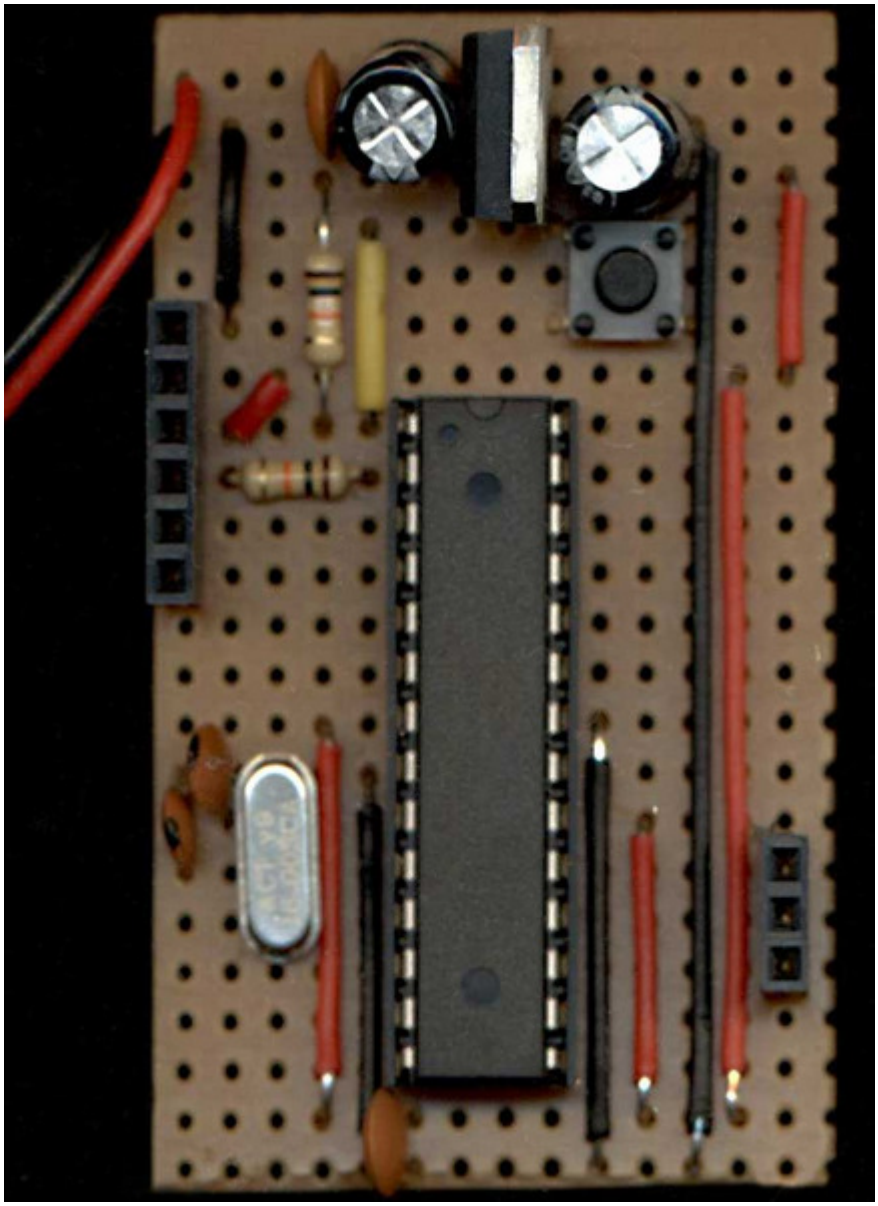
- quarzo C da riga 16 a 17

nessuno di questo ha una polarità specifica quindi non preoccuparsi sul collegamento esatto dei pin.

Completiamo l'opera con alcuni collegamenti tra le varie parti illustrati in figura sotto.

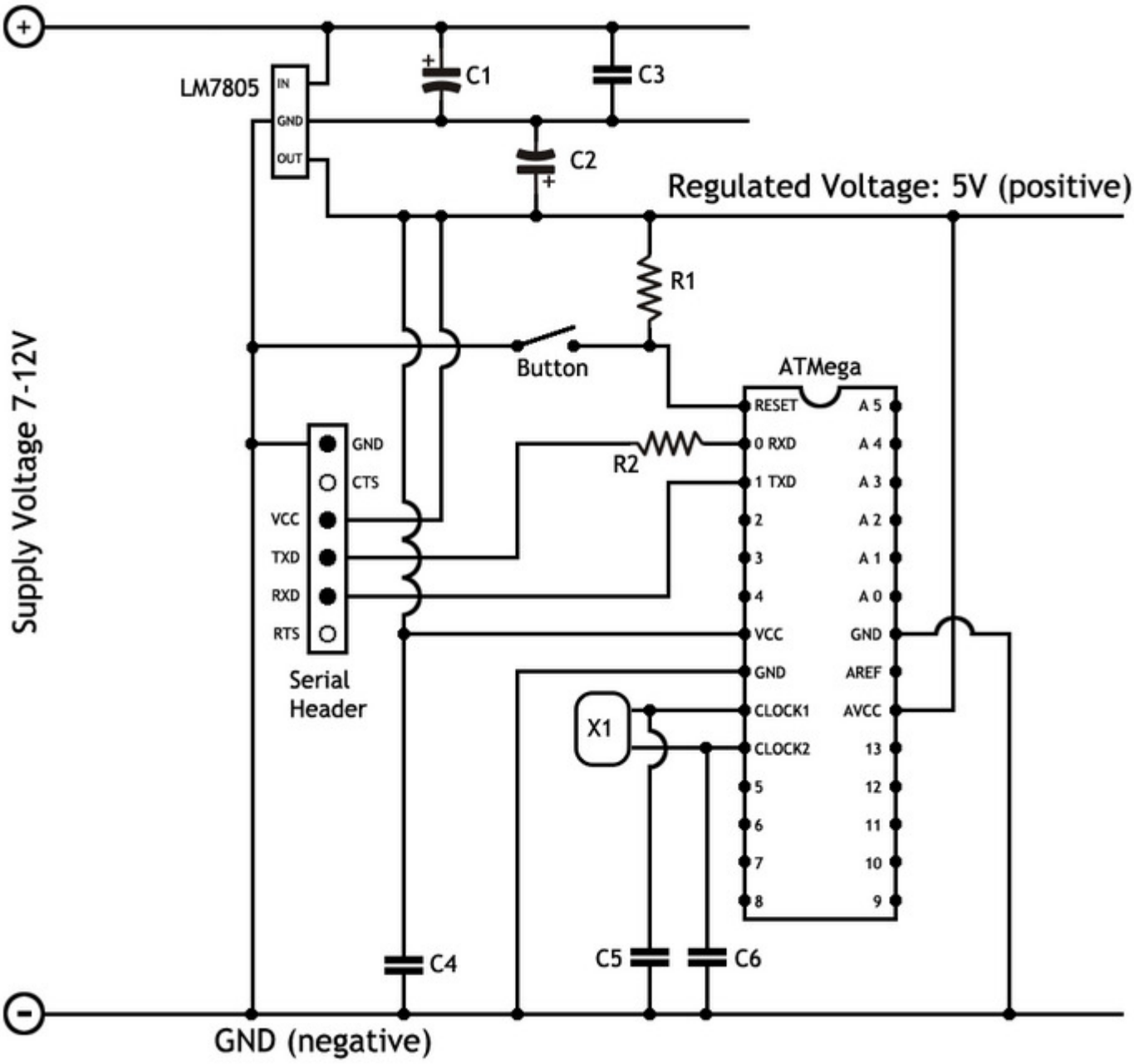


A questo punto possiamo montare l'ATMega e vogliamo, ma non è indispensabile, si può aggiungere un altro connettore femmina a tre pin per leggere l'uoput del ATMega, per il collegamento vedere la figura sotto.



Ora non resta che caricare il nostro programma usando il cavo USB - seriale e la IDE di Arduino ed il circuito è pronto!

Per completezza si riporta in figura sotto lo schema del circuito realizzato.



Abbiamo visto come rendere permanenti i risultati dei nostri "esprimenti" con Arduino senza dover sacrificare un Arduino ogni volta, con ovvio rispario