TELECOMUNICAZIONI

Volume 1.3

Versione [2012/2013.v1]

Logiche programmabili ed utilizzo dell'ambiente di sviluppo Arduino

Classi III Informatica
ITTS Vito Volterra
San Donà di Piave

Prof. Mirco Segatello

Indice

Capitolo 1. Cos'è Arduino

Arduino è una piattaforma hardware per lo sviluppo di applicazioni basate sui microcontrollori ATMEL. Creata in Italia nel 2005, Arduino è basata su una semplicissima scheda di I/O e su un ambiente di sviluppo che usa una libreria Wiring per semplificare la scrittura di programmi in C e C++ da far girare sulla scheda.

Wiring è un ambiente di programmazione open-source per impieghi su schede elettroniche pensato per una facile applicazione in campo elettronico. Wiring è un progetto italiano nato all'istituto di Ivrea ed attualmente sviluppato successivamente all'università Los Andes in Colombia.

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.

La piattaforma hardware Arduino è distribuita agli hobbisti sia attraverso la rete internet che tramite fornitori locali ed è disponibile in versione pre-assemblata, ma le informazioni sul progetto hardware sono rese disponibili a tutti, in modo che, chiunque lo desideri, può costruirsi un clone di Arduino con le proprie mani. Il team di Arduino è composto da Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, e David Mellis. Il progetto ha preso avvio in Italia ad Ivrea, nel 2005, con lo scopo di rendere disponibile a progettisti, studenti e semplici hobbisti, un dispositivo di sviluppo facile ed allo stesso tempo più economico rispetto ad altri sistemi di prototipazione equivalenti.

I progettisti sono riusciti nell'intento di creare una piattaforma di semplice utilizzo ma che, al tempo stesso, permettesse una significativa riduzione dei costi rispetto a molti prodotti disponibili sul mercato. A ottobre 2008 erano già stati venduti più di 50.000 esemplari di Arduino in tutto il mondo.

Una scheda Arduino consiste di un microcontroller a 8-bit AVR prodotto dalla Atmel, con l'aggiunta di componenti complementari che ne facilitino il suo utilizzo in altri circuiti. Le Arduino ufficiali usano i chip della serie megaAVR - nello specifico i modelli ATmega8, ATmega168, ATmega328, e ATmega1280 - ma i suo cloni si sono disponibili anche con altri microcontrollori. Molte schede includono un regolatore lineare di tensione a 5volt e un oscillatore al quarzo da 16MHz (o un risonatore ceramico in alcune varianti), sebbene alcune implementazioni, come ad esempio LilyPad, girino a 8Mhz e facciano a meno dello stabilizzatore di tensione.

Inoltre, il controller Arduino è pre-programmato con un bootloader che semplifica il caricamento dei programmi nella memoria flash incorporata nel chip, rispetto ad altri dispositivi che richiedono, solitamente, un programmatore esterno.

A livello concettuale, tutte le schede vengono programmate attraverso un 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 traslatore di livelli che permette la conversione tra il livello della RS-232 e il livello dei segnali TTL.

Le recenti versioni di Arduino vengono gestite via USB, grazie a un'implementazione che usa un chip adattatore USB-seriale come l'FT232 della FTDI. Alcune varianti, come la Arduino Mini e la versione non ufficiale Boarduino, usano una scheda o un cavo adattatore USB-to-serial separato.

Le schede Arduino dispongono di molti connettori di Input/Output usabili quale estensione per altri circuiti esterni. La Diecimila, ad esempio, offre 14 connettori per l'I/O digitale, 6 dei quali possono produrre segnali PWM, mentre 6 sono dedicati a ingressi di segnali analogici. Questi pin sono disponibili sulla parte superiore della scheda, mediante connettori femmina da 0.1 pollici. Inoltre, sono disponibili commercialmente molte schede applicative plug-in, note come "shields".

Le schede Barebones e Boarduino e Seeduino, tre 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.

L’ambiente di programmazione integrato (IDE) di Arduino è un'applicazione multipiattaforma scritta in Java, ed è derivata dallo IDE creato per il linguaggio di programmazione Processing e adattato al progetto Wiring. È concepito per introdurre alla programmazione hobbisiti e neofiti, a digiuno di pratica nello sviluppo di software. Per permettere la stesura del codice sorgente il programma include un editor di testo dotato di alcune particolarità, come il syntax highlighting, il controllo delle parentesi, e l'indentificazione automatica delle istruzioni. L'editor è inoltre in grado di compilare e lanciare il programma eseguibile in una sola passata e con un singolo 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), che rende molto più semplice implementare via software le comuni operazioni input/output. I programmi di Arduino sono scritti in C/C++, ma, per poter creare un file eseguibile, all'utilizzatore non è chiesto altro se non definire due funzioni:

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

loop() – una funzione chiamata ripetutamente fino a che la scheda non viene spenta.

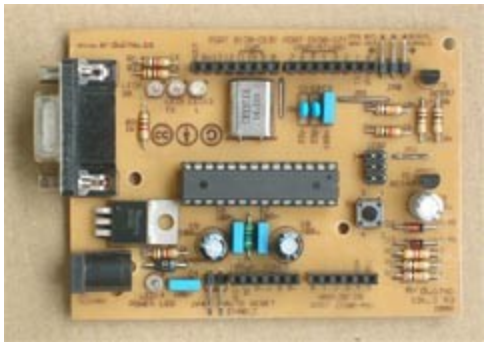
Lo IDE di Arduino usa la GNU toolchain e la AVR Libc per compilare i programmi, mentre usa avrdude per caricarli sulla scheda. L'hardware originale Arduino è realizzato dalla italiana Smart Projects ed alcune schede a marchio Arduino sono state progettate dalla statunitense SparkFun Electronics.

Capitolo 2. Versioni disponibili in commercio

Il progetto Arduino ha subito, dalla sua nascita, una notevole evoluzione che ha riguardato sia l'hardware che il software.

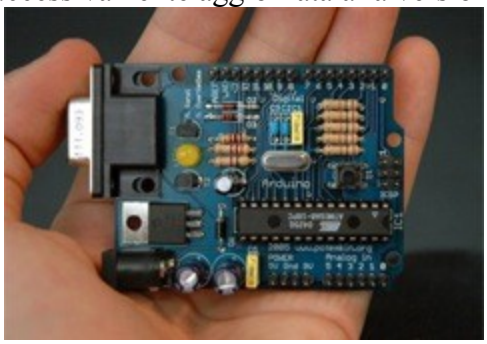
Arduino Single-Sided Serial

La primissima versione, equipaggiata con un ATMega8 e programmabile via seriale. Lo stampato è realizzato a singola faccia con tutti componenti DIP è quindi facilmente realizzabile a livello hobbistico.



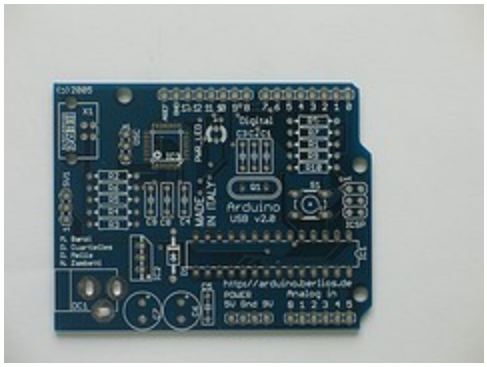
Arduino SERIAL

Costruttivamente migliorata con PCB professionale, programmabile via seriale con microcontrollore ATMega8. Successivamente aggiornata alla versione 2.0.



Arduino USB:

Versione con connessione USB facente uso del convertitore FT232BM. La programmazione avviene connettendola via USB ad un PC. Successivamente aggiornata alla versione 2.0 nella quale viene corretto un problema sulla USB e viene fornita nuova documentazione a corredo.

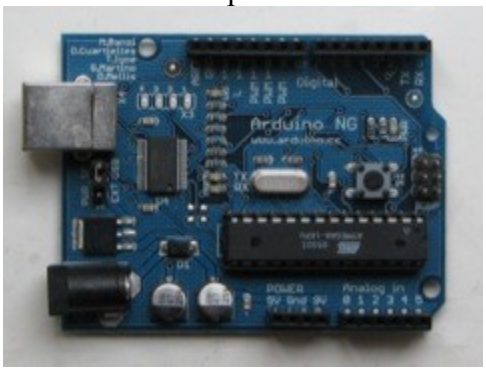


Arduino EXTREME

In questa versione vengono usati più componenti a montaggio superficiale. I connettori sono di tipo femmina a differenza delle prime versioni. Sono installati due LED sulle linee TX ed RX per monitorare il traffico della comunicazione. Anche in questo caso è stata proposta la versione 2.0.

Arduino NG

È la New Generations di Arduino, utilizza il convertitore USB-Seriale di FTDI FT232RL, che richiede meno componenti esterni del FT232BM. Inoltre ha un LED incorporato sul pin 13. La versione plus viene fornita con un ATMega 168 invece di un ATMega8. La versione C di NG non ha il LED incorporato sul pin 13 ma semplicemente una resistenza da 1K, il LED può quindi essere messo all'esterno senza aggiunta di ulteriori componenti.

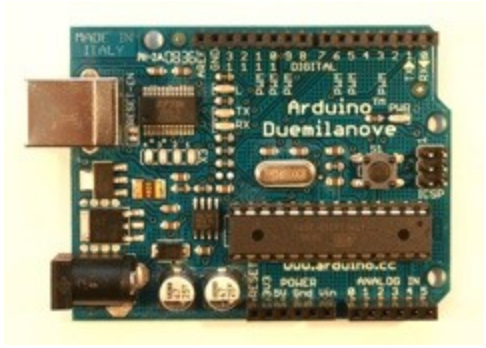


Arduino DIECIMILA

Con interfaccia di programmazione USB e con un ATmega168 in un package DIL28. Il reset può avvenire indifferentemente via software o con pulsante fisico sulla scheda. Viene usato uno stabilizzatore di tensione e l'alimentazione può avvenire indifferentemente via USB o esterna non stabilizzata. Un polyfuse viene messo a protezione dell'alimentazione della USB, è stato montato un LED sul pin 13 per i primi esperimenti.

Arduino DUEMILANOVE

E' la versione aggiornata della diecimila, viene eliminato il selettore per l'alimentazione in quanto uno switch interno commuta in automatico tra alimentazione USB o esterna. Viene eliminata la funzione di autoreset eventualmente ripristinabile con un jumper.



Arduino UNO

In questa nuova versione di Arduino non viene più utilizzato il convertitore USB-Seriale della FDTI bensì un microntrollore ATmega8U2 programmato come convertitore USB-Seriale.

Questo nuovo prodotto della ATMEL è infatti un microcontrollore con a bordo un modulo Transceiver USB liberamente programmabile come ad esempio alcuni PIC della serie 18F della Microchip. La nuova scheda ha anche ricevuto la certificazione FCC sulle emissioni elettromagnetiche. Le diciture "ROHS Compliant" e "Zero carbon footprint" fanno emergere l'interesse dello team Arduino per l'ambiente.

Il microcontrollore ATmega8U2 usato come convertitore USB-Seriale può essere facilmente programmato in quanto, al suo interno, è già precaricato il bootloader. In questo caso è possibile utilizzare le appositi piazzole di programmazione dopo aver attivato la modalità di programmazione saldando il piccolo jumper disponibile sul retro della scheda. Il software necessario per lo sviluppo di un proprio programma si chiama Atmel's FLIP software per windows (http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3886) e DFU programmer per Mac OS X and Linux (http://dfu-programmer.sourceforge.net/).

Posizione microcontrollore 8U2.

Jumper per attivare modalità aggiornamento firmware 8U2.

piazzole per la programmazione di 8U2.

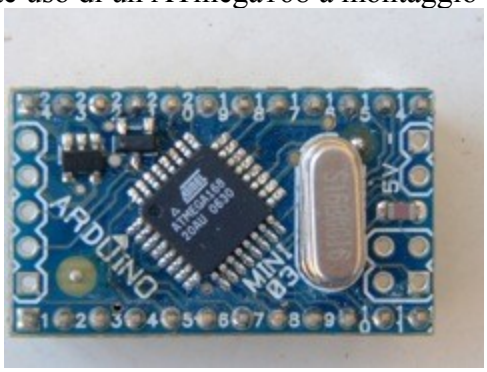
Ai più preparati non passerà inosservato il grosso vantaggio di avere il convertitore USB-Seriale programmabile; infatti sino ad ora Arduino poteva essere visto dal PC solo come una periferica seriale ed infatti i driver del convertitore della FDTI creavano una seriale virtuale. Adesso, invece, potendo programmate il convertitore, Arduino può essere visto dal sistema operativo del PC come una periferica ad-hoc. Nulla vieterebbe di far rilevare Arduino come una stampante e quindi, qual'ora inviaste il comando di stampa da un qualsiasi software, i dati giungerebbero ad Arduino il quale magari controlla una macchina CNC! Allo stesso modo si potrebbe inventare un nuovo sistema di puntamento e, una volta connesso al PC, verrebbe riconosciuto come periferica tipo mouse ed il cursore sullo schermo si muoverebbe tramite la vostra periferica. Non proprio alla portata di tutti ma si lascia immaginare un nuovo ed interessante scenario per quanto riguarda lo sviluppo di nuove periferiche per PC.

Arduino UNO REV3:

Non ci sono sostanziali modifiche rispetto la REV1, sono stati aggiunti i pin SDA e SCL per la comunicazione TWI vicino al pin AREF e vicino al pin RESET altri sono stati collocati altri due nuovi pin, il pin IOREF che consentirà agli shield di adattarsi alla tensione fornita dalla scheda, ed un pin non collegato, riservato per scopi futuri. l'Atmega 16U2 sostotuisce la versione 8U2. La REV3 funziona con tutti gli shield realizzati per le versioni precedenti.

Arduino MINI:

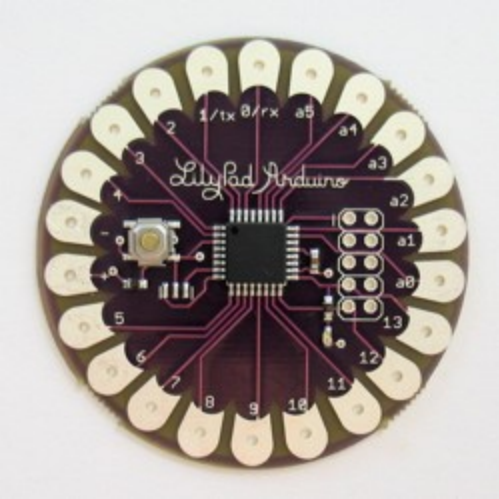
Versione in miniatura facente uso di un ATmega168 a montaggio superficiale.



Arduino NANO:

Versione ancora più piccola della Mini, utilizzante lo stesso controller ATmega168 SMD e alimentata tramite USB

Arduino LILYPAD:



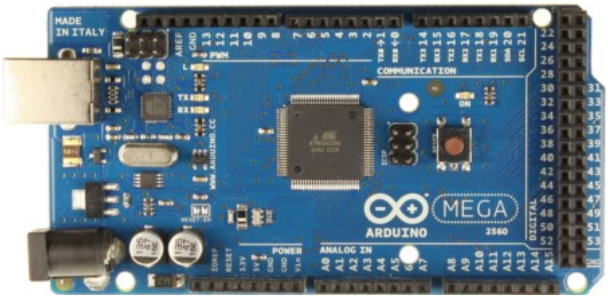
Un progetto minimalista per applicazioni indossabili con lo stesso ATmega168 SMD.

Arduino BT:

Con interfaccia di programmazione Bluetooth e con un ATmega168.

Arduino MEGA:

E' una scheda maggiorata basata su ATmega1280 con ben 54 pin di I/O e 16 canali analogici, anche la memoria programma è maggiorata.



CLONI:

Per il fatto che è possibile per terze parti creare una propria Arduino compatibile con il software originale sono disponibili in commercio diversi 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 per indicare opere derivate senza il permesso. Il documento che esprime la policy ufficiale sull'uso del nome "Arduino" mette l'accento su come il progetto sia aperto ad incorporare lavori altrui nel prodotto ufficiale.

Quale conseguenza di queste convenzioni sulla protezione del nome, molti prodotti simili ad ARDUINO sono presenti sul mercato ma con nome diverso dall'originale come FREEDUNO o SEEEDUINO. Il nome non è però inteso come un marchio commerciale ma è liberamente utilizzabile da chiunque lo desideri.

Tra le schede disponibile sul mercato la SEEDUINO V2.12 propone una valida alternativa all'originale con alcune differenze con essa: Programmazione via USB con connettore micro, alimentazione esterna con connettore JST, completamente assemblata con componenti SMD.

Seeeduino è una scheda compatibile con Arduino Diecimila e basata sul microcontrollore ATmega168. Il pinout, i fori di fissaggio e le dimensioni sono compatibili al 100% con quelle di Arduino Diecimila. La scheda dispone di 14 I/O (di cui 6 possono essere utilizzati come uscite PWM), 8 ingressi analogici, 16 kB di memoria flash, 1 kB di SRAM e 512 byte di memoria EEPROM. Rispetto alla scheda Arduino Diecimila dispone di alcune differenze.

Caratteristiche:

1. Gli stessi ingressi e uscite sono disponibili su due connettori differenti
2. Il microcontrollore ATmega168 versione DIP è stato sostituito con la versione SMD, questo ha permesso di ottenere più spazio sul PCB semplificando l'inserimento delle schede dei vostri prototipi sui connettori del Seeeduino.
3. Per un facile accesso tutti i pulsanti e gli interruttori sono vicini ai bordi del PCB.
4. A causa delle sue dimensioni ingombranti, il connettore USB tipo B è stato sostituito con un connettore Mini USB.
5. A causa delle sue dimensioni ingombranti, la presa di alimentazione (jack da 3,5 mm) è stata sostituita con connettore JST a 2 poli.
6. Dispone di LED indicatore presenza alimentazione e di Reset vicino al pulsante RST.
7. Funzione di Auto-reset selezionabile.

8. Dispone di interruttore di selezione per tensione a 3,3 V o 5 V.
9. Interfaccia UART per FTDI232 che permette di trasferire il bootloader senza la necessità di utilizzare un cavo ISP.
10. Sono state aggiunte 2 uscite ADC.
11. Facile connessione I2C e sensori analogici.
12. Possibilità di alimentazione diretta del Seeeduino a 5 VDC (Attenzione: Usare solo 5 V) mediante ingresso supplementare.
13. Il microcontrollore ATmega168 versione DIP è stato sostituito con la versione SMD, questo ha permesso di ottenere più spazio sul PCB.
15. Riga supplementare di pin a saldare. E' così possibile utilizzare un connettore femmina o maschio a propria scelta.
16. Dispone di un regolatore di tensione da 3,3 V in grado di fornire una maggior corrente di alimentazione (150 mA), piuttosto che i 50 mA forniti dall' FT232.

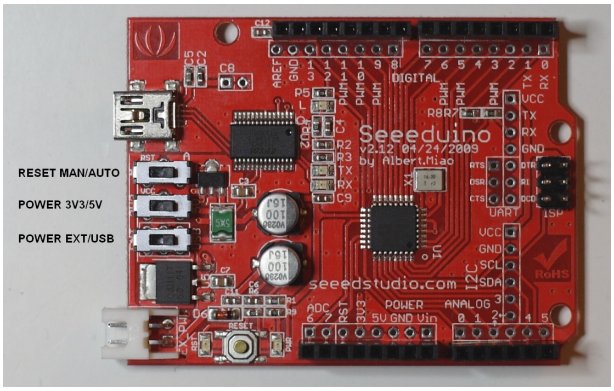
SEEDUINO V3.28

Seeeduino è una scheda compatibile con Arduino Diecimila e basata sul microcontrollore ATmega328. Differisce dal modello Seeeduino V2.12 perchè dispone di maggior memoria flash, EEprom e Sram.

Per quanto riguarda il suo utilizzo, esso non di discosta molto dalla versione originale. Sulla scheda sono disponibili tre piccoli deviatori con le seguenti funzioni:

1) Seleziona se l'alimentazione giunge dalla USB o da fonte esterna. Per le prima applicazione impostiamo l'alimentazione da USB così da non doverci procurare ulteriori alimentatori.
2) Selezione l'alimentazione della logica a 5Volt o 3,3volt.
3) Il terzo deviatore seleziona la modalità di reset. Impostandola su automatico la scheda si resetterà in automatico non appena sarà caricato il firmware.

Per la prima applicazione è opportuno impostare tutti i deviatori con la levetta rivolta verso l'interno della scheda.



La scheda viene connessa al PC con il solito cavo USB (con connettore micro dal lato scheda), la procedura di avvio e di programmazione è identica alla scheda Arduino originaria come descritto in precedenza.

Licenza d'uso

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 disponibili, e concessi in uso, secondo i termini legali contenuti nella licenza GPLv2.

La GNU General Public License è una licenza per software libero. È comunemente indicata con l'acronimo GNU GPL o semplicemente GPL.

Contrapponendosi alle licenze per software proprietario, la GNU GPL assicura all'utente libertà di utilizzo, copia, modifica e distribuzione. La GPL ha incontrato un gran successo fra gli autori di software sin dalla sua creazione, ed è oggi la più diffusa licenza per il software libero.

Come ogni licenza software, la GPL è un documento legale associato al programma rilasciato sotto tale licenza. Come ogni licenza di software libero, essa concede ai licenziatari il permesso di modificare il programma, di copiarlo e di ridistribuirlo con o senza modifiche, gratuitamente o a pagamento. Rispetto alle altre licenze di software libero, la GPL è classificabile come "persistente" e "propagativa".

È "persistente" perché impone un vincolo alla redistribuzione: se l'utente distribuisce copie del software, deve farlo secondo i termini della GPL stessa. In pratica, deve distribuire il testo della GPL assieme al software e corredarlo del codice sorgente o di istruzioni per poterlo ottenere ad un costo nominale. Questa è la caratteristica principe della GPL, il concetto ideato da Richard Stallman e da lui battezzato copyleft. Il suo scopo è di mantenere libero un programma una volta che esso sia stato posto sotto GPL, anche se viene migliorato correggendolo e ampliandolo.

È "propagativa" perché definisce nel testo una particolare interpretazione di "codice derivato", tale che in generale l'unione di un programma coperto da GPL con un altro programma coperto da altra licenza può essere distribuita sotto GPL.

Sia la scheda originale che i suoi 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 normalmente forniti. Esistono espansioni dedicate a varie funzioni, dal controllo motorio, al breadboarding (prototipizzazione).

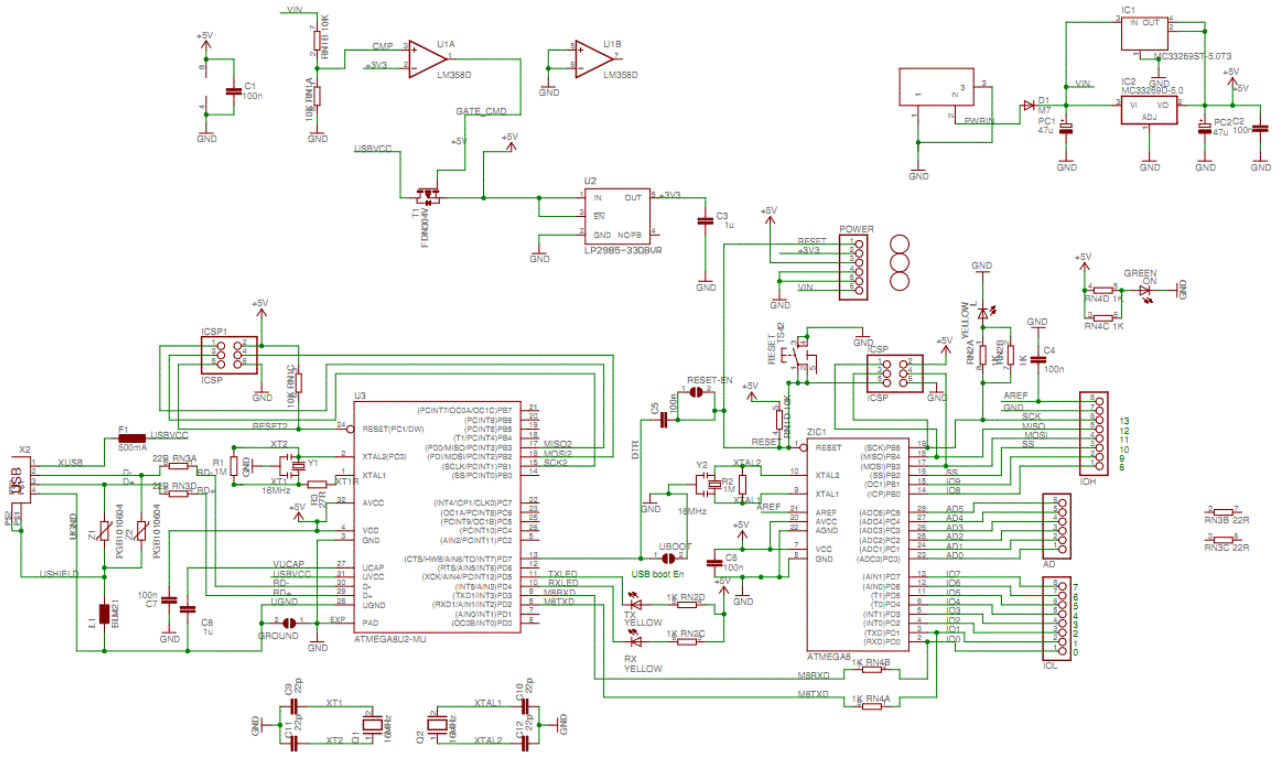
Tutta la documentazione originaria di riferimento costantemente aggiornata è presente sul sito ufficiale in lingua inglese: http://arduino.cc/en esiste anche la versione in italiano ma non costantemente aggiornata http://arduino.cc/it . All’interno potete navigare attraverso alcuni link per accedere alla sezione hardware, alla sezione software, il forum o i tutorial: http://arduino.cc/en/Tutorial/Blink

http://arduino.cc/en/Main/Hardware

http://arduino.cc/it/Main/FAQ

Capitolo 3. Descrizione dell'Hardware

Adesso che abbiamo appreso il senso del progetto ARDUINO entriamo nei dettaglio di uno dei prodotti, per la precisione la scheda Arduino UNO che risulta essere la più recente al momento in cui scriviamo, inoltre è quella che meglio raccoglie la filosofia ARDUINO e ben si presta ad un corso didattico.



Arduino UNO è basato sul microcontrollore ATmega328 (in formato DIP) e dispone di 14 pin di I/O (di cui 6 usabili come uscite PWM), 6 ingressi analogici, un oscillatore a 16MHz, un connettore per la programmazione In-Circuit ed un Plug di alimentazione. Come nelle ultime versioni di Arduino è presente un connettore USB che, semplicemente connesso ad un PC, permette sia di alimentare la scheda sia di programmarla.

Questa versione di Arduino è da intendersi la versione 1.0 con la nuova tecnologia per la connessione alla USB, la pagina di riferimento per comparare tutte le versioni hardware è la seguente: http://arduino.cc/en/Main/Boards.

Caratteristiche di Arduino UNO	
Microcontrollore	ATmega328
Tensione di lavoro	5V
Alimentazione esterna (raccomandata)	7-12V
Alimentazione esterna (limiti)	6-20V
Digital I/O Pins	14 (di cui 6 usabili come PWM output)
Ingressi analogici	6
Corrente per ogni I/O Pin	40 mA
Corrente prelevabile da 3.3V Pin	50 mA

Corrente prelevabile da 5V Pin	500mA (dipendente da Valimentazione)
Flash Memory	32 KB di cui 0.5 KB usati per il bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

L'alimentazione alla scheda può avvenire tramite la porta USB ma è disponibile un connettore plug che accetta, in ingresso, una tensione non regolata con valore compreso tra 7 e 12, in questo caso un semplice alimentatore non stabilizzato impostato sul valore di 9volt è l'ideale ma nulla vieta di alimentare la scheda tramite una batteria a 9 o 12 volt. La sorgente di alimentazione viene riconosciuta in automatico, e nessun deviatore deve essere impostato. La porta USB è comunque protetta da accidentali corto circuiti nella scheda e comunque non vengono prelevati più di 500mA.

Protezione da una sovra-alimentazione della USB

Arduino ha un polifusibile autoripristinante che protegge la porta USB del computer da corto circuiti e sovra-alimentazione. Anche se la maggior parte dei computer già prevedono una loro protezione interna, il fusibile fornisce un ulteriore livello di protezione. Se più di 200 mA attraversano la porta USB, il fusibile automaticamente interromperà la connessione fino a quando il corto o il picco non sia rimosso.

Caratteristiche fisiche

Il PCB della scheda Duemilanove misura 6.8 e 5.33 cm., il connettore USB e l'attacco per l'alimentazione escono leggermente dal profilo dello stampato inoltre, sono presenti tre fori per il fissaggio della scheda ad una superficie o un contenitore.

Approfondimenti sull'hardware

Una prima fonte di alimentazione può essere applicata al plug al quale fa capo un diodo a protezione dell'inversione di polarità ed uno stabilizzatore di tensione a 5volt. L'alimentazione giunge anche tramite il connettore USB ma solo se non è presente l'alimentazione primaria. La scheda commuta automaticamente tra una fonte di alimentazione e l'altra senza dover intervenire su nessun deviatore.

Il clock è ottenuto tramite un quarzo a 16MHz e questo stabilirà anche l'intervallo di tempo per l'esecuzione di una istruzione in quanto quasi tutte le istruzioni necessitano di un ciclo di clock per la loro esecuzione. Nei microcontrollori microchip, ad esempio sono necessari 4 impulsi di clock per eseguire un'istruzione e quindi il numero di istruzioni eseguibili in un secondo (MIPS) sono equivalenti ad un quarto della frequenza di clock.

Notiamo dallo schema che tutti i segnali sono disponibili nei vari connettori, quindi, oltre agli in/out disponiamo del segnale seriale in uscita dall'FT232, i segnali per il programmatore seriale esterno, le tensioni di alimentazione. A completare la scheda il diodoLED PWR acceso in presenza di scheda alimentata ed il diodo LED da usarsi a piacimento.

Descrizione delle funzioni:

Pin di alimentazione nella scheda:

VIN. Questo pin semplicemente replica la tensione fornita in ingresso sul connettore plug. Può essere usato per alimentare altri circuiti che dispongano già di un regolatore di tensione.

5V. Questo pin fornisce i 5volt dello stabilizzatore di tensione interno alla scheda. E' utile per alimentare altri circuiti compatibili con i 5volt.

3V3. Questo pin fornisce I 3,3volt dello stabilizzatore interno alla scheda. E' utili per alimentare circuiti compatibili con tensioni di 3,3Volt. La massima corrente prelevabile è di 50mA.

GND. Pin di massa (GND).

Memoria:

Il microcontrollore ATmega328 dispone di 32 KB di memoria programma di cui 0.5 KB usati per il bootloader. Dispone in oltre di 2 KB di SRAM e 1 KB di EEPROM utilizzabile, quest'ultima, per il salvataggio di dati permanenti (mantiene i dati anche in assenza di alimentazione).

Ingressi/uscite

Ciascuno dei 14 pin può essere usato come pin di input o output e gestisce una corrente massima di 40mA, in oltre dispone di una resistenza di pull-UP del valore di 20-50Kohms (attivabile tramite programmazione).

Seriale: pin TX(1) e RX(0). Questi pin fanno capo all'USART interno al microcontrollore e sono connessi al convertitore USB-Seriale della scheda.

Interrupts esterni: pin 2 e 3. Questi pin possono essere configurati come trigger per eventi esterni come ad esempio il rilevamento di un fronte di salita o di discesa di un segnale in ingresso.

PWM: pin 3, 5, 6, 9, 10, e 11. Questi pin possono essere configurati via software per generare segnali PWM con risoluzione di 8 bit. Tramite un semplice filtro RC è possibile ottenere tensioni continue di valore variabile.

SPI: pin 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Questi pin possono essere programmati per una comuniazione SPI.

LED: pin 13. Questo pin è connesso ad un LED interno alla scheda utile per rapide diagnostiche.

Arduino UNO dispone di 6 ingressi analogici A0,A1,A2,A3,A4 e A5 con risoluzione di 10bit e tensione di ingresso 0-5volt di default; tuttavia è possibile usare l'ingresso Aref per modificare il range di misura.

I2C: 4 (SDA) and 5 (SCL). Pin usati per la comunicazione nello standard I2C (due fili) in abbinamento alla libreria Wire.

Reset. Portato a livello logico basso resetta il microcontrollore. Questa funzione può essere attivata anche tramite il pulsante presente nella scheda.

Comunicazione

Il microcontrollore ATmega328 utilizza il modulo UART interno per comunicare, con livelli logici 0-5volt, via seriale con altri dispositivi o con il PC. Questi segnali sono disponibili sui pin esterni (TX e RX) e sono connessi anche al convertitore USB-Seriale della scheda permettendo una comunicazione tramite la porta USB del PC. A differenza del chip della FDTI per il quale era necessario installare appositi driver, con l'utilizzo dell'integrato ATmega8U2 questo non è più necessario in quanto vengono usati i driver comuni della periferica USB già disponibili con il sistema operativo. Tuttavia, con sistemi operativi Windows per la corretta creazione di una porta COM virtuale, è necessario installare un driver aggiuntivo. Arduino UNO è compatibile con sistemi operativi Windows, Mac OS X e Linux.

Il microocntrollore ATmega368 della scheda Aruino Duemilanove ha già un bootloader pre-caricato che permette di caricare nuovo codice senza la necessità di uno specifico programmatore esterno e comunica utilizzando il protocollo originale STK500. Si può naturalmente evitare l'utilizzo del bootloader e programmare la ATmega368 attraverso il connettore ICSP (In-Circuit Serial Programming).

Capitolo 4. Installazione ed utilizzo pratico

Ma entriamo nel cuore dell'applicazione ovvero il software di sviluppo necessario alla programmazione del microcontrollore, infatti la scheda appena acquistata non fa assolutamente nulla. Per poterla programmare da remoto senza alcun specifico programmatore il microcontrollore viene fornito preprogrammato con un specifico bootloader che instraderà in modo corretto il firmware nell'apposita area di memoria EEPROM durante la fase di programmazione. Faremo riferimento alla scheda Arduino diecimila e duemilanove ma in modo simile potremmo operare per altre schede Arduino (tipo USB) e cloni sempre con USB.

Vediamo ora per passi come rendere operativa una scheda Arduino. Per prima cosa dobbiamo procurarci una scheda Aduino Diemilanove ed un cavo cavo USB con connettore standard A-B, quello normalmente utilizzato per collegare una stampante USB al computer.

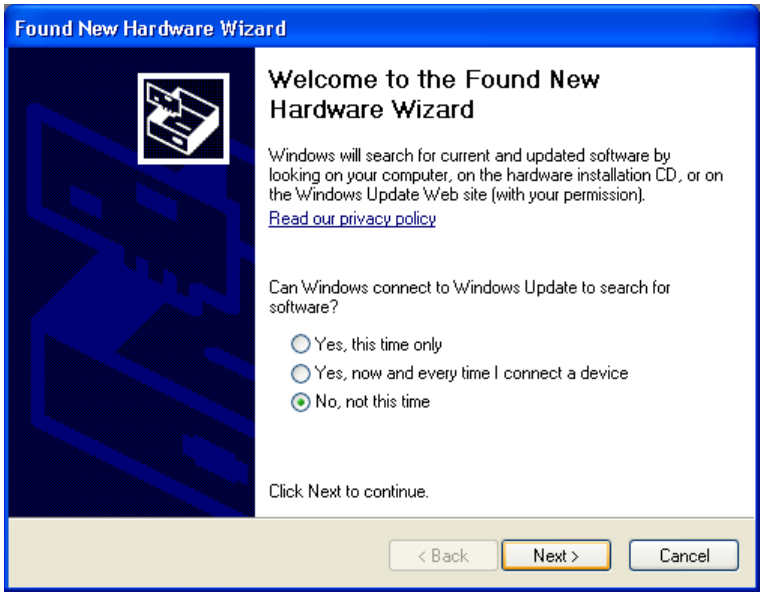
Scaricare il software Arduino:

Per programmare la scheda Arduino è necessario disporre dell'ambiente di sviluppo (IDE) per Arduino. Tutte le versioni dalla 00 alla 23 sono delle pre release, mentre la prima versione ufficiale è la 1.0 seguita poco dopo dalla versione 1.0.1 alla quale facciamo riferimento. Il software è disponibile per Windows, MAC e Linux e non necessita di installazione, potremmo pensarlo come una versione portable.

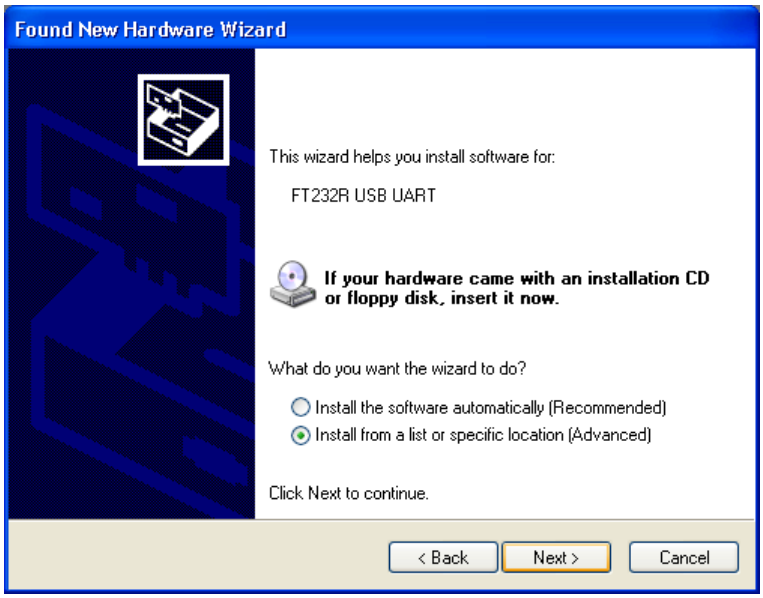
Decomprimete il file scaricato assicurandovi di conservare la struttura delle cartelle. Nelle varie cartelle è compreso oltre al sistema di sviluppo tutti i file java necessari , i driver per il convertitore seriale-USB e gli esempi.

Connettere la scheda:

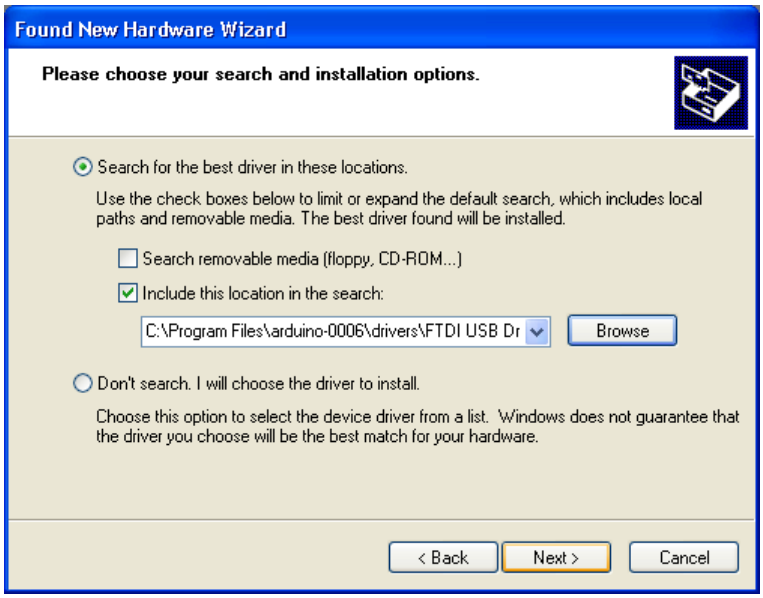
Come primo applicazione suggeriamo di alimentare la scheda direttamente dalla USB, per fare questo è sufficiente inserire il cavo tra la porta USB del PC e la scheda. Non ci sono jumper o deviatori da impostare, il LED di alimentazione (PWR) dove illuminarsi. Vediamo come avviene l'installazione dei driver con una vecchia versione di Windows e con ambiente di sviluppo di Arduino antecedente la versione 1.0.



Impostazione manuale del percorso dei driver.



Specificare il percorso in cui trovare i driver, nel nostro caso la cartella FDT USB driver contenuta nei file di Arduino.



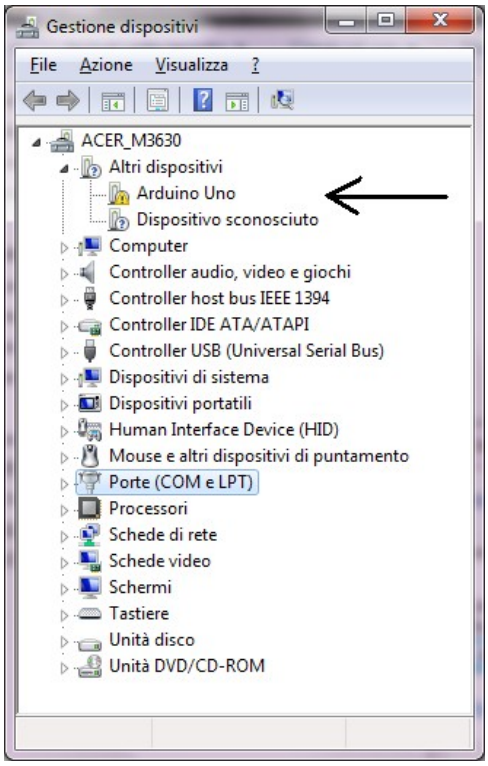
Avviare l'installazione dei driver.

Installazione driver avvenuta con successo.

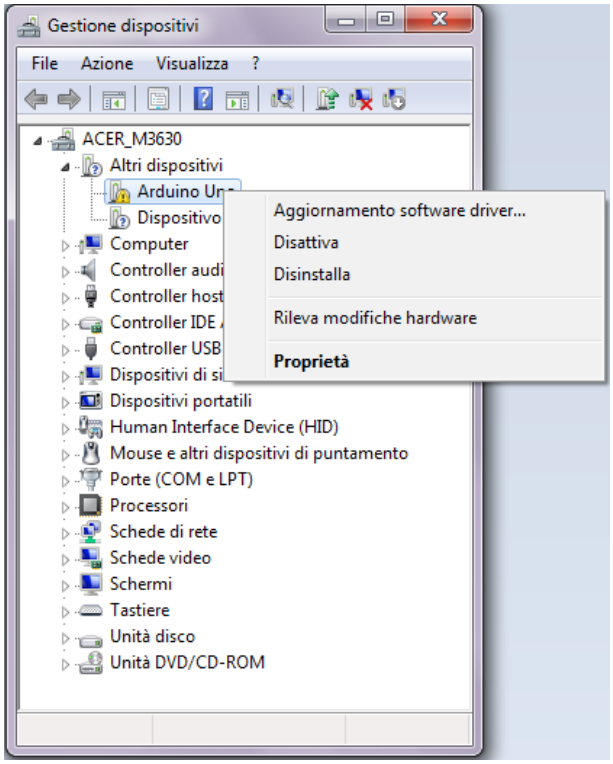
Se si utilizza l'hardware Arduino UNO la procedura è leggermente differente.

Per il corretto utilizzo di questa nuova versione di Arduino è opportuno disporre dell'ultima versione del software gratuitamente all'indirizzo http://arduino.cc/en/Main/Software la quale contiene sia i driver per Arduino UNO, sia i Driver per il chip FTDI degli Arduino precedenti. Con sistema operativo Windows, appena connettete Arduino al PC, esso verrà riconosciuto come nuovo hardware e vi verrà chiesto l'installazione del driver; non dovete far altro che specificare come percorso la cartella "driver" del software Arduino (Arduino UNO.inf); il sistema operativo provvederà alla sua installazione ed alla creazione della porta COM virtuale.

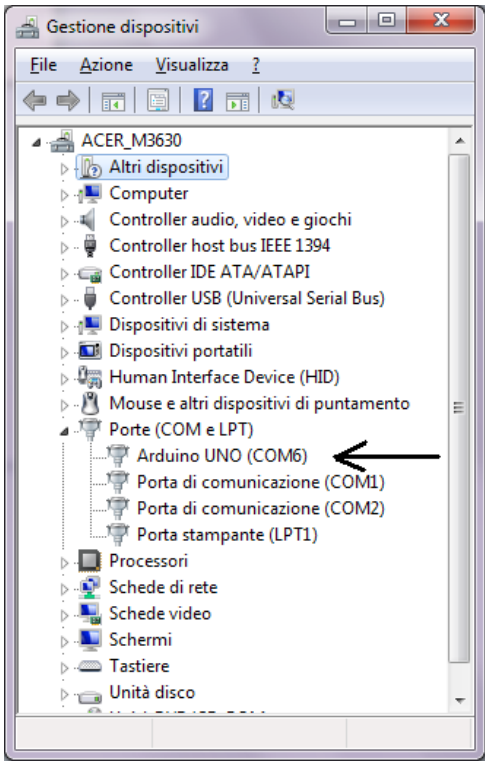
Nel caso incontraste dei problemi è possibile verificare manualmente lo stato di installazione dei driver. In Windows Seven dovete accedere a "Gestione dispositivi" e verificare lo stato delle periferiche:



Se la periferica non è correttamente installata cliccateci sopra con il tasto sinistro del mouse e selezionate la voce proprietà.



Utilizzate la funzione "Aggiornamento software driver" per installare manualmente i driver corretti. Se la procedura è andata a buon fine nella lista delle COM vi ritroverete la voce Arduino con indicata la porta assegnata.

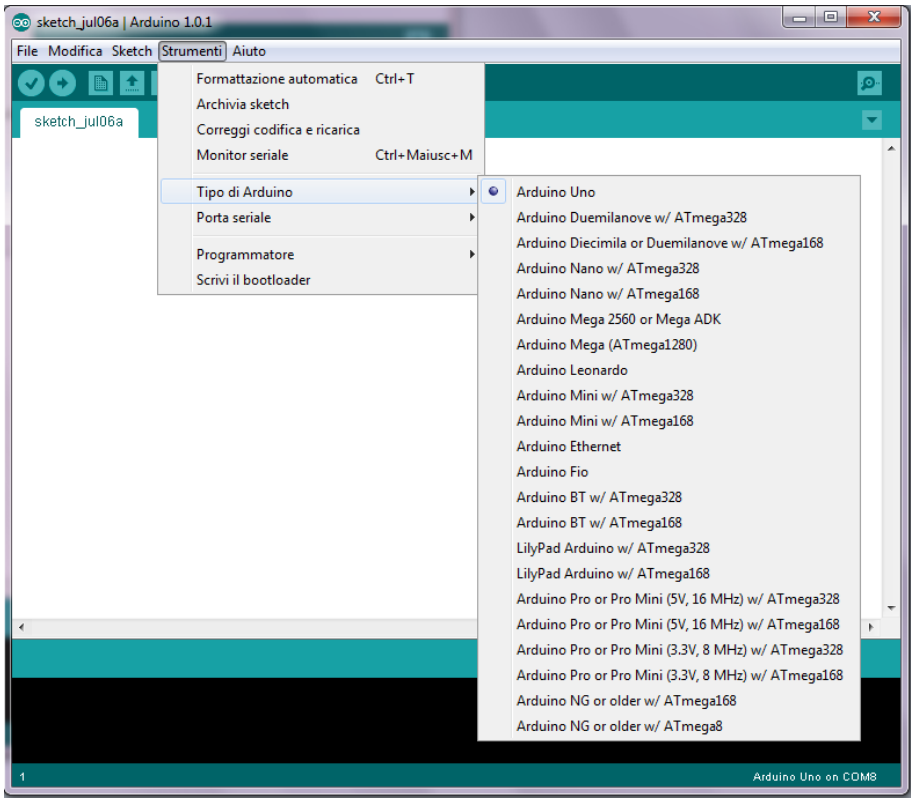


Grazie al Bootloader preinstallato a bordo non è necessario utilizzare alcun programmatore esterno ne è necessario rimuovere il microcontrollore, la connessione USB tra PC e Arduino è sufficiente per permettere la programmazione e la gestione della comunicazione. La funzione di autoreset interna alla scheda permette la programmazione con un solo click del mouse.

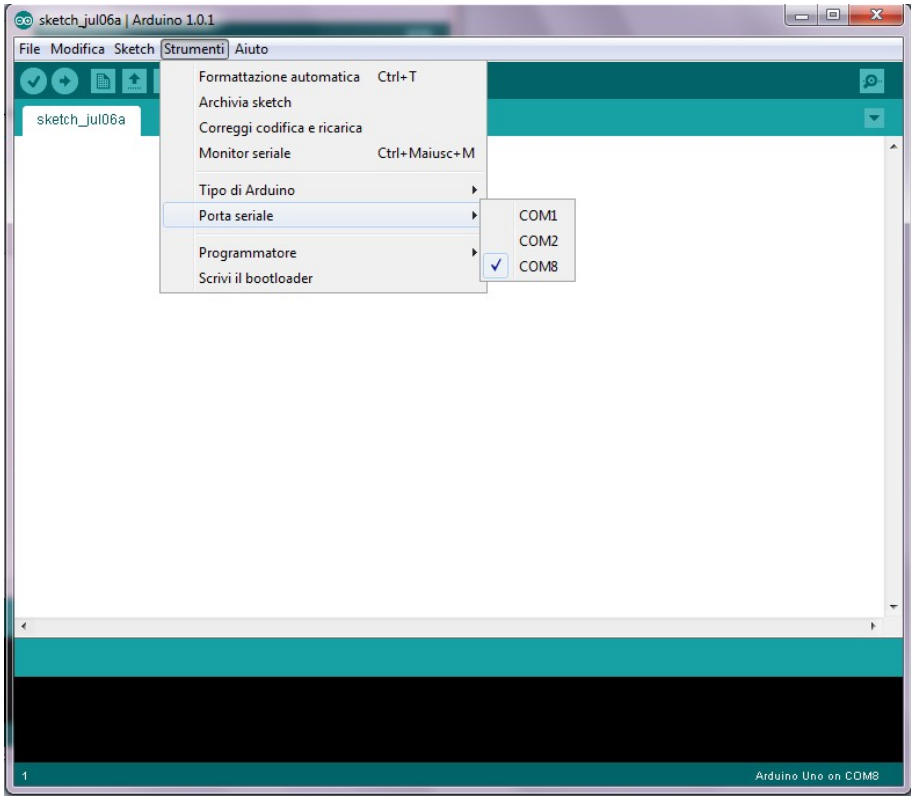
Sulla versione UNO di Arduino è presente un nuovo Bootloader basato sul protocollo STK-500 che occupa un quarto di memoria rispetto il precedente. Adesso sono sufficienti 512b di memoria al posto dei 2K della precedente versione, in oltre, può gestire la velocità di comunicazione sino a 115K contro i 57,6K della precedente versione.

Tutta la documentazione sia hardware che software, compresi i sorgenti, sono disponibili per il download sul sito di Arduino.

A questo punto avviamo il software di sviluppo di Arduino, aprendo la cartella appena decompressa e facendo doppio clic sull'icona dell'eseguibile, dalla schermata principale, procedete alla selezione della periferica appena installata. In "Strumenti-Tipo di Arduino" selezionate la scheda in vostro possesso.



Da "Strumenti-Porta seriale" selezionate la COM alla quale è connessa la periferica.



A questo punto siete perfettamente operativi e si conclude la parte relal tiva alla messa in servizio.

Capitolo 5. Le funzioni disponibili

//	Commento monoriga es.: // Il mio primo programma per Arduino
/*	Apertura commento multiriga; es.: /* * Questo è il mio primo * programma per Arduino */
*/	Chiusura commento Multiriga; es.: vedi sopra
void	dichiarazione di funzione senza ritorno, ossia la funzione esegue delle operazioni ma non restituisce alcun valore dopo l'elaborazione
setup()	funzione base (obligatoria) di ogni programma Arduino
loop()	funzione base (obligatoria) di ogni programma Arduino
int	dichiarazione di variabile di tipo *integer* (intero); utilizzato anche per dichiarare le funzioni che restituiscono valori interi
pinMode(*pin*,*mode*)	funzione di configrazione di un piedino Arduino, in modo che sia utilizzabile come INPUT o come OUTPUT
digitalWrite(*pin*, *level*)	invia un comando di output sul piedino selezionato (pin) di tipo digitale, pe cui 0 o 1; in sostituzione dello 0 e dell'1 si possono utilizzare le costanti LOW (0) e HIGH (1)
delay(*second*)	funzione utile per introdurre un dalay (attesa) tra una istruzione e la successiva, il paramero *second* è esperesso in millesimi di secondo.

Strutture

- void setup()
- void loop()

Controllo

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

- goto

Sintassi

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

Operazioni Aritmetiche

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

Comparazione

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

Operazioni booleane

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

Operatori

- ++ (increment) -- (decrement)
- += (compound addition)
- -= (compound subtraction)
- *= (compound multiplication)
- /= (compound division)

Costanti

Constants are particular values with specific meanings.

- HIGH | LOW
- INPUT | OUTPUT
- true | false
- Integer Constants

Tipi di dati

Variables can have various types, which are described below.

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

Conversioni

- int()
- long()
- float()

Funzioni

Digital I/O

- pinMode(pin, mode)
- digitalWrite(pin, value)
- int digitalRead(pin)

Analog I/O

- int analogRead(pin)
- analogWrite(pin, value) - *PWM*

Advanced I/O

- shiftOut(dataPin, clockPin, bitOrder, value)
- unsigned long pulseIn(pin, value)

Time

- unsigned long millis()
- delay(ms)
- delayMicroseconds(us)

Math

- min(x, y)

- max(x, y)
- abs(x)
- constrain(x, a, b)
- map(value, fromLow, fromHigh, toLow, toHigh)
- pow(base, exponent)
- sq(x)
- sqrt(x)

Trigonometry

- sin(rad)
- cos(rad)
- tan(rad)

Random Numbers

- randomSeed(seed)
- long random(max)
- long random(min, max)

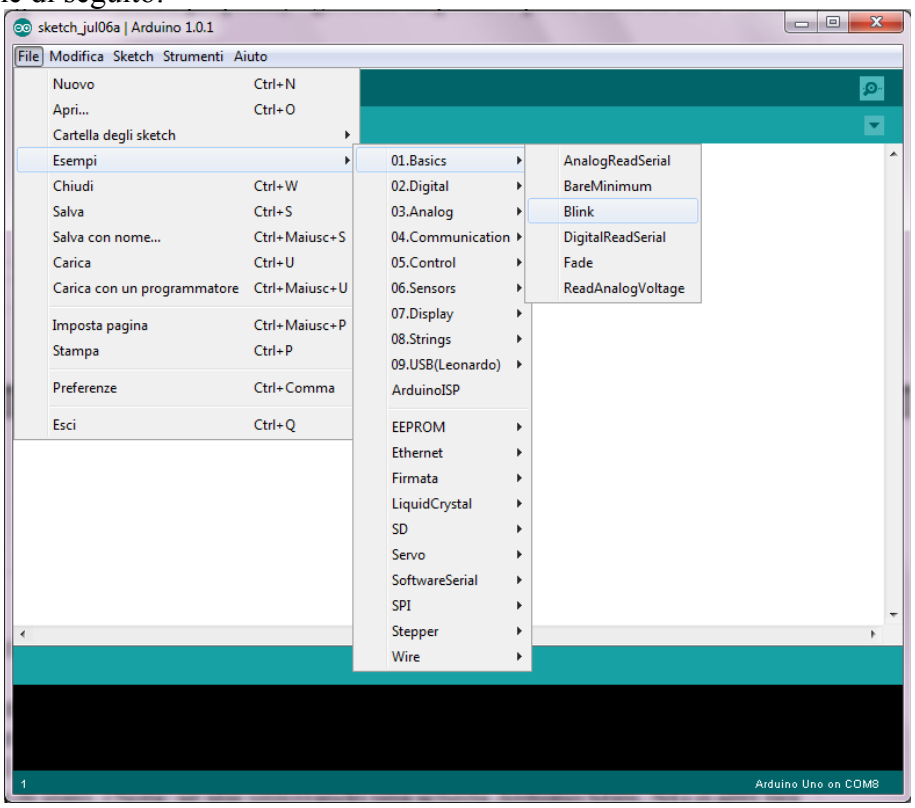
Serial Communication

Usate per comunicare tra schede arduino oppure tra schede arduino ed il PC. Vengono usati i pin TX ed RX facenti capo al modulo USART del microcontrollore.

- Serial.begin(speed)
- int Serial.available()
- int Serial.read()
- Serial.flush()
- Serial.print(data)
- Serial.println(data)

Capitolo 6. Primo esempio: lampeggio di un LED

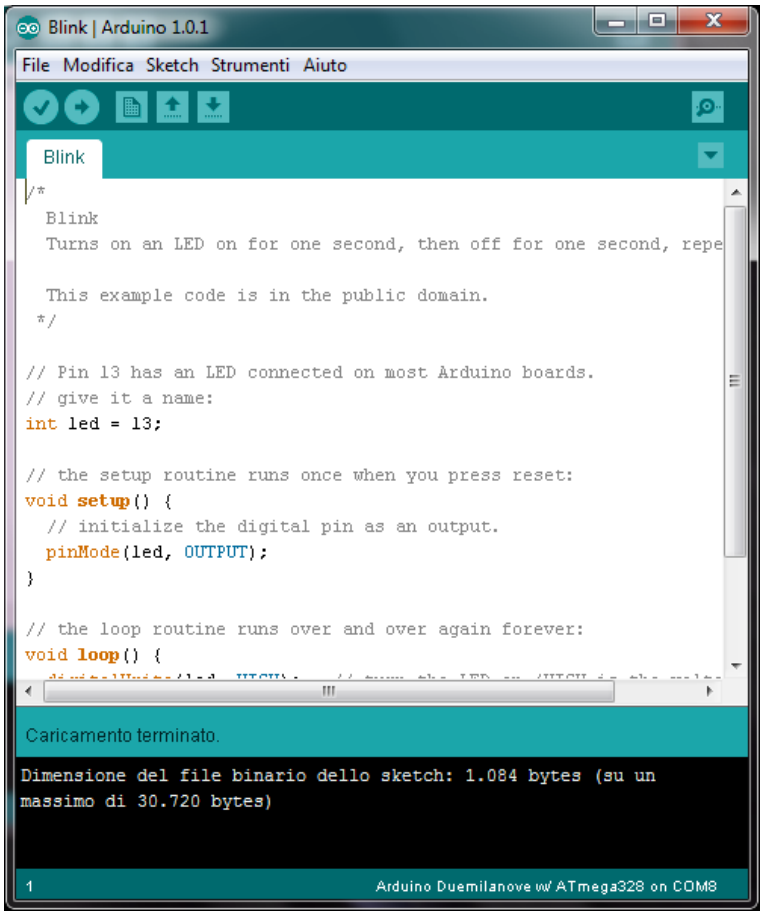
Sempre dalla schermata principale del software di sviluppo aprite il programma di esempio che fa lampeggiare il LED di sistema connesso al pin 13. Ecco come appare il codice del programma di esempio che farà lampeggiare il LED. Aprite "File-Esempi-01.Basic-Blink", vi apparirà una schermata come di seguito:



Ora basta semplicemente clic sul pulsante "Carica" nel software. Dopo qualche secondo dovrebbero lampeggiare molto velocemente i LED RX e TX sulla scheda. Significa che la scheda sta comunicando con il computer. Se la procedura è andata a buon fine apparirà il messaggio "Caricamento terminato" in basso a sinistra nella barra di stato. Nota: utilizzando una scheda Arduino Mini, NG o altri tipi di schede, si dovrà premere fisicamente il pulsante di reset della scheda appena prima di fare clic su "Upload" nel software, nelle altre schede il reset è automatico.)

Viene anche riportato il numero di byte occupati dal programma rispetto la massima capacità della memoria.

Il software di sviluppo non ha la necessità di creare file intermedi ma partendo direttamente dal codice sorgente crea immediatamente il codice macchina da inserire nel microcontrollore il tutto in un unico passaggio. Esiste comunque il pulsante di compilazione che provvede a verificare la correttezza del codice.



Qualche secondo dopo il termine del processo di upload, vedrete il LED color ambra lampeggiare sulla scheda. Congratulazioni! avete una scheda Arduino connessa e funzionante.

Capitolo 7. Collegamento e gestione di un LED

Vediamo ora come sia possibile gestire un LED esterno alla scheda Arduino. Il LED è un particolare tipo di diodo che emette luce se percorso da corrente. Come in molti altri tipi di diodo, il principio di funzionamento del LED si basa sulle proprietà delle giunzioni, le superfici di contatto tra due zone di un cristallo semiconduttore (zona p e zona n) con caratteristiche diverse (vedi drogaggio). Quando è percorsa da una debole corrente elettrica (valori tipici da 10 a 20 mA), la giunzione emette spontaneamente fotoni di una determinata lunghezza d'onda, e quindi di un determinato colore. La corrente elettrica che attraversa un LED deve avere un'intensità controllata, per evitare che arrechi danni al componente. Molto usati come indicatori elettronici, i LED sono reperibili in varie forme e dimensioni. Anche se è cosa poco nota, i LED sono "macchine reversibili": infatti, se la loro giunzione viene esposta direttamente ad una forte fonte luminosa o ai raggi solari, ai terminali appare una tensione, dipendente dall'intensità della radiazione e dal colore del led in esame (massima per il blu). Questa caratteristica viene abitualmente sfruttata nella realizzazione di sensori, per sistemi di puntamento (inseguitori solari) di piccoli impianti fotovoltaici o a concentratore e per molti altri scopi. Il colore della luce emessa dipende dal drogante utilizzato:

- Al Ga As - rosso ed infrarosso
- Ga Al P - verde
- Ga As P - rosso, rosso-arancione, arancione, e giallo
- Ga N - verde e blu
- Ga P - rosso, giallo e verde
- Zn Se - blu
- In Ga N - blu-verde, blu
- In Ga Al P - rosso-arancione, arancione, giallo e verde
- Si C come substrato - blu
- Diamante (C) - ultravioletto
- Silicio (Si) come substrato - blu (in sviluppo)
- Zaffiro (Al 2O3) come substrato – blu

I LED si comandano in corrente a seconda del tipo utilizzato e dell'intensità luminosa richiesta:

tipo di LED	corrente ID
LED basso consumo	2-10mA
LED normali	10-20mA
LED flash	20-40mA
LED di potenza	100-2000mA

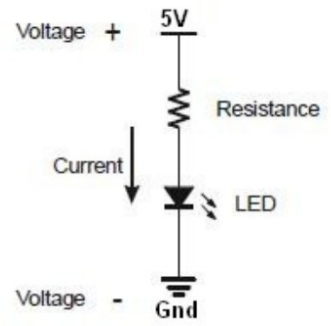
Quando un LED è acceso ai suo capi è presente una tensione detta di polarizzazione diretta (Vf) che dipende dal tipo di drogaggio e quindi dal tipo di luce emessa.

Colore del LED	Tensione diretta Vf
infrarosso	1,3v
rosso	1,8v
giallo	1,9v
verde	2,0v
bianco	3,0v
blu	3,5v
ultravioletto	4-4,5v

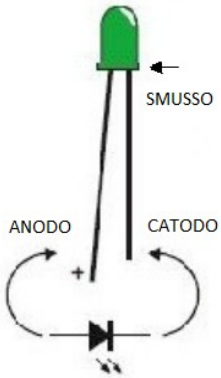
Per poter lavorare correttamente anche a tensioni diverse è necessario inserire in serie al diodo una resistenza di valore opportuno che dipende dal tipo di diodo usato e dalla tensione di alimentazione (Va) utilizzata:

R=(Va-Vf)/ID

Per un diodo normale di colore verde alimentato a 5v la resistenza da aggiungere in serie vale:
$R=(5-2)/10\cdot10^{-3}=300ohm$
Un valore commerciale di 330 ohm potrà andare più che bene.
Lo schema per il corretto funzionamento del LED è il seguente:

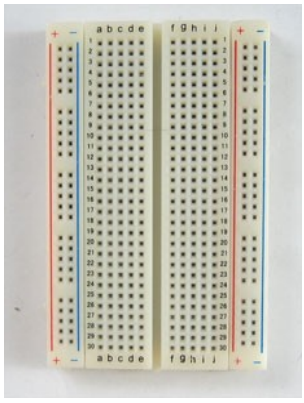


Il LED ha due terminali, uno è chiamato anodo, l'altro è chiamato catodo.
L'anodo è indicato con il simbolo (+) ed è il terminale più lungo ed è rivolto verso il positivo di aliementazione. Il catodo è indicato con il simbolo (-) ed è il terminale più corto ed è rivolto verso il negativo di alimentazione. Inoltre se si guarda attentamente il corpo plastico del LED è quasi rotondo con un piccola zona piatta vicino al terminale corto identificando così il catodo. Questo fatto è particolarmente utile se i terminali sono stati tagliati alla stessa lunghezza.

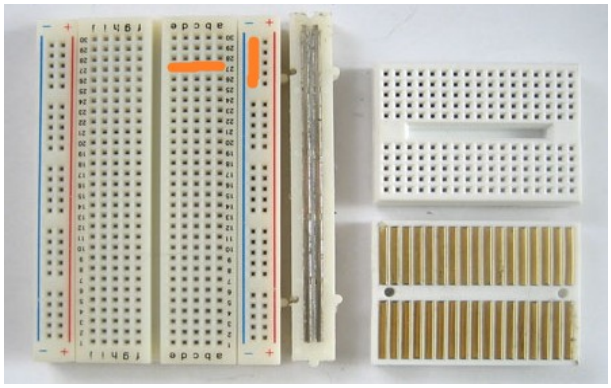


Se viene connesso al contrario non circolerà corrente ed il diodo non si accende. Se viene inserito in un circuito con una tensione maggiore di 5v è molto probabile che si danneggi a causa della tensione inversa troppo elevata.

Per connettere il diodo LED e la resistenza ad Arduino è necessario procurasi una breadboard ed alcuni filetti.

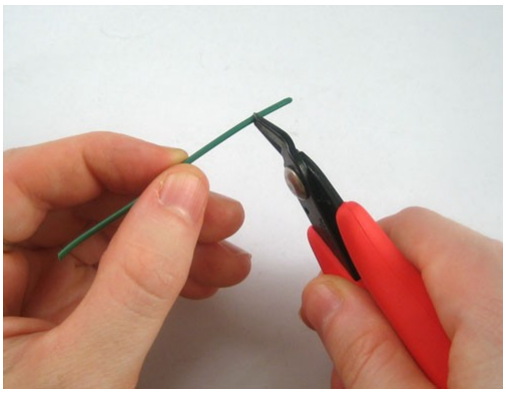


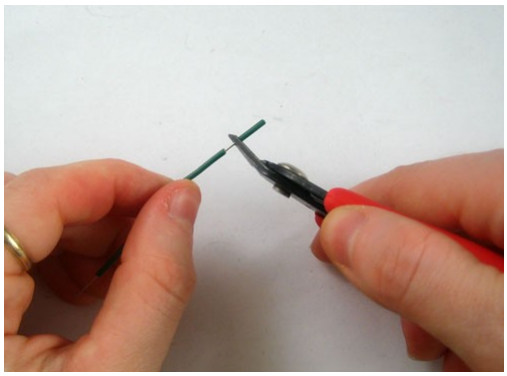
In sostanza la breadboard è formata da una piastra non conduttiva forata entro la quale inserire i componenti ed una piastra sottostante conduttiva che provvede a mettere in contatto i componenti inseriti.

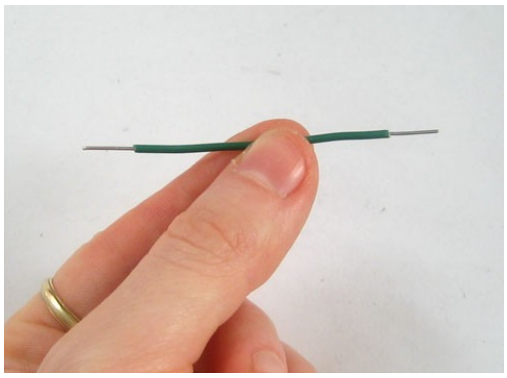


Il contatto avviene seguendo un ordine preciso indicato nella figura con il tratto arancione.

I filetti per il collegamento possono essere ottenuti da del filo in rame rigido isolato.







Esistono in commercio dei filetti flessibili e già terminali adatti all'inserimento in breadboard.

Supponiamo ora di voler comandare il nostro LED tramite il pin 12 di Arduino.

schema elettrico dei componenti e del loro collegamento

Schema pratico di montaggio

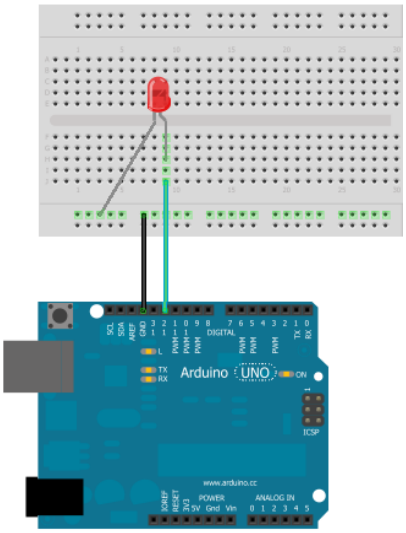
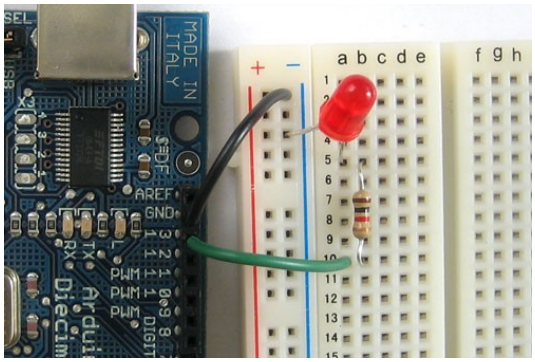


Foto dei collegamenti



Adesso occorre agire a livello software, il programma è simile al precedente ma occorre specificare l'uso del LED sul pin 12.

```
/*
        Titolo: LED_01
        Funzione: Lampeggio LED su pin 12.
        Autore: Marco Rossi
 */

int ledPin =  12;    // LED connesso al pin digitale D12

void setup()   {
  // inizializza il pin del LED come uscita digitale:
  pinMode(ledPin, OUTPUT);
```

```
}

// Viene dichiarato un loop senza uscita
// Le istruzioni interne al loop vengono continuamente eseguite

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

Dopo averlo compilato e trasferito aa Arduino vedrete il LED immediatamente lampeggiare. Analizziamo il software riga per riga almeno per questo primo esempio:

Racchiuso tra /* e */ troverete dei commenti che riportano il nome del programma, la funzione svolta ed il nome dell'autore. Ulteriori commenti su di una riga possono essere scritti iniziandola con //.

Nella prima riga “int ledPin = 12” si dichiara una variabile denominata ledPin associata al pin 12. La riga di codice *pinMode(ledPin, OUTPUT)* racchiusa nella struttura “Void Setup()” specifica che la variabile ledPin associata al pin 12 è utilizzata come uscita. All’interno di questa struttura andremo sempre a specificare la funzione associata per ogni pin utilizzato, infatti ciascun pin digitale di Arduino può essere usato come pin di uscita i pin di ingresso.

La seconda ed ultima struttura di dati si chiama “Void loop()” e rappresenta semplicemente un loop infinito, ovvero le istruzioni al suo interno vengono eseguite in successione partendo dalla prima istruzione dopo la parentesi graffa aperta “{“ sino alla fine della struttura delimitata dalla parentesi graffa chiusa “}” per poi essere rieseguite nuovamente all’infinito. Come vedete, quindi, l’impostazione di un programma risulta assai agevole e semplice. L'istruzione *digitalWrite(ledPin, HIGH)* è la riga che permette di accendere il LED mentre la riga di programma *digitalWrite(ledPin, LOW)* lo spegne.

Commentare una linea di codice:

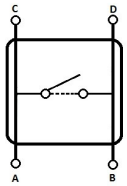
Mettere un // a sinistra di un comando, lo trasforma in un commento. Questo è uno strumento utile perché non dovrete fisicamente cancellare il codice per vedere cosa accade se lo togliete dal programma. È molto più facile aggiungere e rimuovere un // che eliminare e ridigitare i comandi. Inoltre è possibile sempre spiegare, passo dopo passo, cosa il programma dovrebbe svolgere.

Capitolo 8. Collegamento e gestione di un pulsante

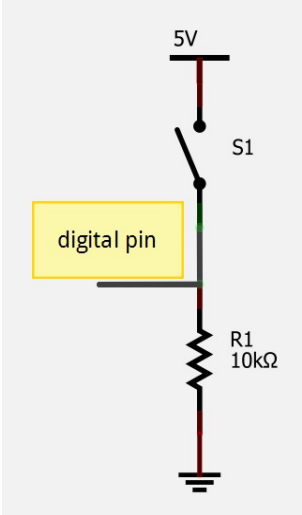
Vediamo adesso come poter usare un dispositivo di input con la scheda Arduino. Il più semplice componente di input è sicuramente un pulsante. Nella sua forma minimale essa è costituito da un contatto che si chiude per azione meccanica.



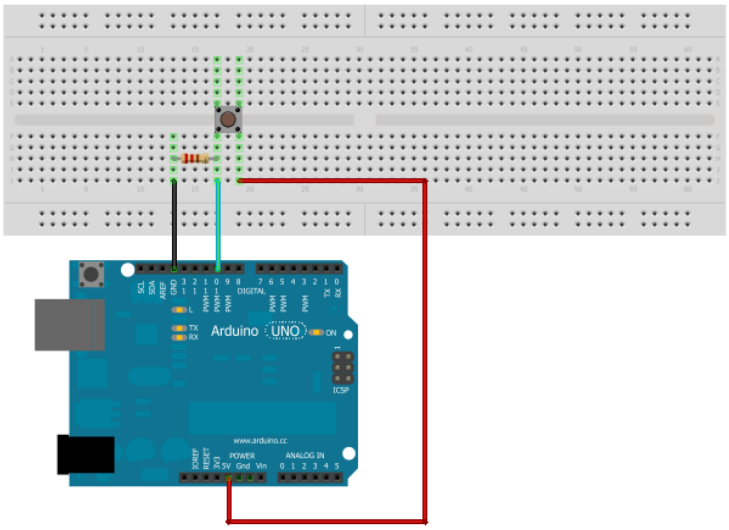

Il simbolo elettrico si suddivide a seconda del numero di terminali presenti. Nella versione a 4 terminali per uso su circuito stampato i due terminali aggiuntivi servono solo per assicurare un miglior ancoraggio allo stampato ed aumentarne la resistenza meccanica alla pressione.



Per il collegamento elettrico occorre tenere conto che ogni logica necessita di due livelli logici "UNO" e "ZERO" corrispondenti ai due livelli di tensione "0v" e "5v".



Quanto il pulsante non è premuto la resistenza R1 assicura che al pin digitale arrivi la tensione di 0v ovvero uno "zero" logico, la resistenza svolge quindi una funzione di mantenere bassa la tensione, in gergo tecnico si dice resistenza di pull-down. Quando invece premiamo il pulsante la tensione di 5v giunge al pin digitale ovvero un "uno" logico. A livello pratico il cablaggio è il seguente:



Dal punto di vista della programmazione è sufficiente utilizzate lo sketch: esempi-02.digital-button. La funzione svolte è molto semplice, finché premo il pulsante il LED rimane acceso.

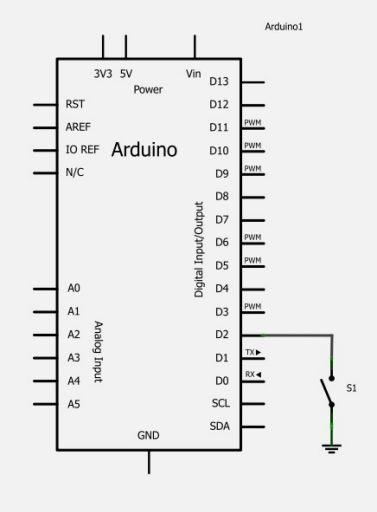
```
/*
  Button
 */
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
```

```
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

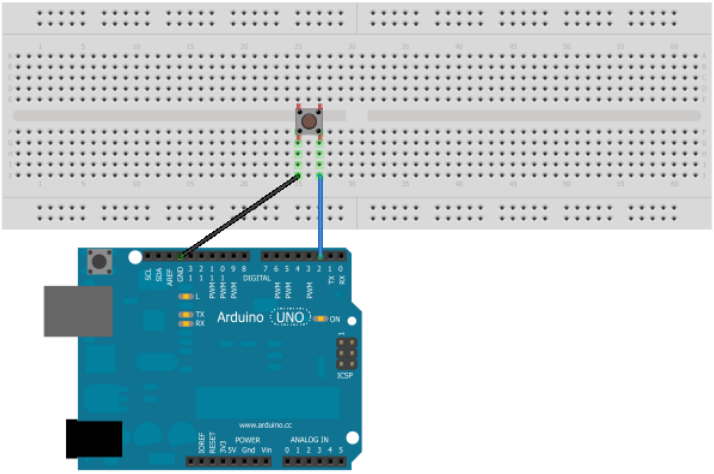
In questo listato la riga di codice *pinMode(buttonPin, INPUT)* specifica che il pin 2 è usato come ingresso. La riga di codice *buttonState = digitalRead(buttonPin)* permette di conoscere il livello logico del pin di ingresso ovvero se il pulsante è premuto o meno.

Questo però non è l'unico modo di procedere, un'alternativa potrebbe essere quella di invertire nello schema il pulsante con la resistenza. In questo caso quando il pulsante non è premuto la resistenza assicura che al pin digitale arrivi la tensione di 5v ovvero, "UNO" logico, in questo caso la resistenza svolge la funzione di pull-up. A pulsante premuto, al pin digitale, arriverà una tensione di 0v ovvero "ZERO" logico.

Questa è la configurazione più idonea anche se il ragionamento avviene in logica negativa: se premo il pulsante c'è "UNO" se non premo c'è "ZERO". Spesso nelle logiche programmabili la resistenza di pull-UP è già presente all'interno del cip per cui è sufficiente cablare il solo pulsante:



Mentre lo schema pratico di montaggio diventa:



Vediamo ora come procedere con la programmazione, facciamo prima riferimento allo schema con pulsante e resistenza il pin digitale interessato è il numero 2.

Avviate Arduino ed aprite lo sketch esempi-02.digital-button. Per prima cosa dobbiamo aggiungere la riga di codice che abilita la resisteza di pull-up sul pin D2. In questo esempio oltre all'ingresso 2 viene usato il LED di sistema connesso al pin 13. quando si preme il pulsante e finché rimane premuto il LED si accende, viceversa rimane spento.

```
/*
  Button2
 */
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT_PULLUP);
}

void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == LOW) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
```

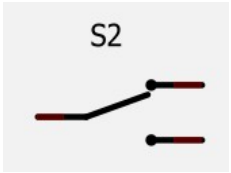
```
    digitalWrite(ledPin, LOW);
  }
}
```

In questo nuovo listato è stata modificata la riga che imposta il pin D2 come ingresso aggiungendo la parola INPUT_PULLUP ovvero imposta il pin come ingresso ed abilita la resistenza di pull-up interna. Lavorando in logica negativa e volendo che il LED si accenda quando si preme il pulsante è necessario modificare anche l'istruzione if: *if (buttonState == LOW) then ...*

Varianti all'uso dei pulsanti

In commercio esistono molte altre forme di contatto elettrico a seconda dell'impiego se ne vuole fare. Il finecorsa, in inglese micro-switch. è simile ad un pulsante ma è predisposto per essere premuto da un oggetto da non da un dito. La presenza di tre reofori è giustifica dal fatto che questo componente dispone di un contatto di scambio (deviatore).



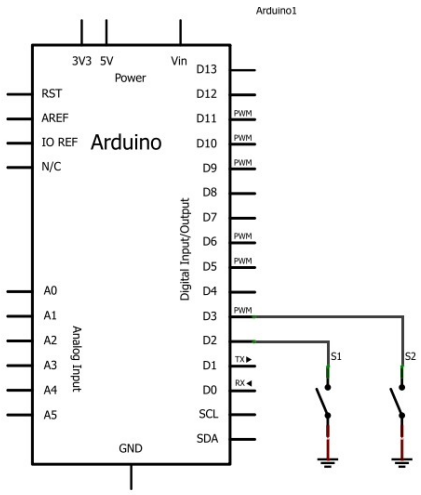


Il contatto read è composto da un'ampolla nella quale sono presenti i due contatti e da una gocia di mercurio. Solo quando il componente si trova in una determinata posizione la goccia di mercurio crea il contatto elettrico tra i due contatti.

I contatti magnetici sono formati da un contatto che si chiude quando nelle vicinanze è presente una calamita. Sono di solito impiegati negli impianti d'allarme per identificare l'apertura di porte o finestre.


Capitolo 9. Funzione AND con due pulsanti

Vediamo ora come sia possibile implementare la funzione logica AND con due pulsanti. La funzione logica che vogliamo ottenere è la seguente: Se il pulsante S1 è premuto E il pulsante S2 è premuto si accende il LED. Per fare questo dovremmo necessariamente connettere due pulsanti ad Arduino, mentre come LED è sufficiente quello già presente sulla scheda.



I pulsanti sono connessi agli ingressi D2 e D3 in logica negativa. Lo sketch è il seguente:

```
/*
     Titolo: AND
     Funzione: Funzione AND tra due pulsanti.
     Autore: Marco Rossi
 */

const int s1Pin = 2;
const int s2Pin = 3;
const int ledPin =  13;

int s1State = 0;
int s2State = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(s1Pin, INPUT_PULLUP);
   pinMode(s2Pin, INPUT_PULLUP);
}

void loop(){
  s1State = digitalRead(s1Pin);
  s2State = digitalRead(s2Pin);

  if ((s1State == LOW) && (s2State == LOW))  {
    digitalWrite(ledPin, HIGH);
```

```
  }
  else {
    digitalWrite(ledPin, LOW);
  }
}
```

La funzione AND viene implementata a livello software tramite l'operatore && e la riga di codice che lo usa è: *if ((s1State == LOW) && (s2State == LOW)) then...*

Capitolo 10. Funzione OR con due pulsanti

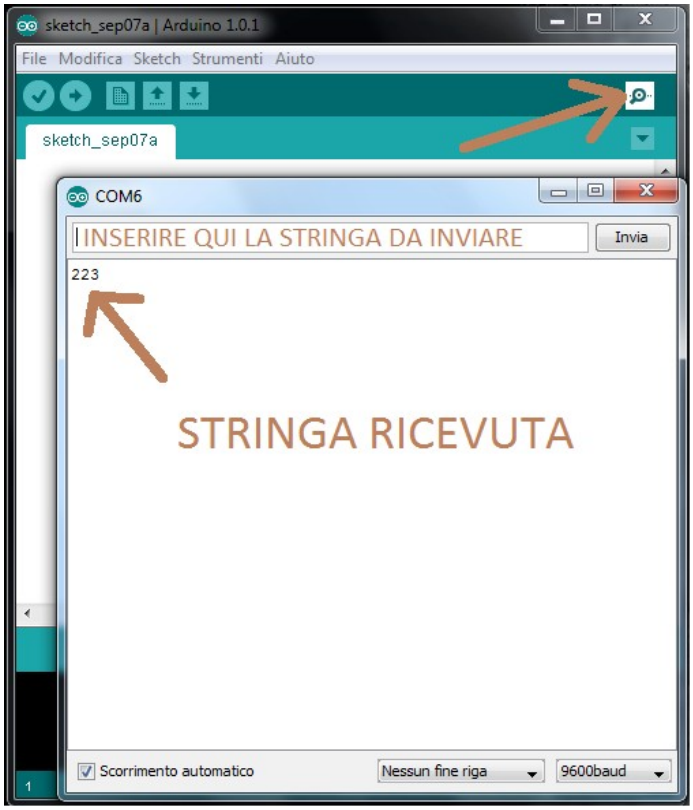
Vediamo ora come sia possibile implementare la funzione logica OR con due pulsanti. La funzione logica che vogliamo ottenere è la seguente: Se il pulsante S1 è premuto OPPURE il pulsante S2 è premuto si accende il LED. Lo schema elettrico è lo stesso dell'esercizio precedente in quanto la funzione richiesta è implementata a livello software. Lo sketch è il seguente:

```
/*
     Titolo: AND
     Funzione: Funzione AND tra due pulsanti.
 */

const int s1Pin = 2;
const int s2Pin = 3;
const int ledPin =  13;

int s1State = 0;
int s2State = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(s1Pin, INPUT_PULLUP);
  pinMode(s2Pin, INPUT_PULLUP);
}

void loop(){
  s1State = digitalRead(s1Pin);
  s2State = digitalRead(s2Pin);

  if ((s1State == LOW) || (s2State == LOW))  {
    digitalWrite(ledPin, HIGH);
  }
  else {
    digitalWrite(ledPin, LOW);
  }
}
```

La funzione OR viene implementata a livello software tramite l'operatore || e la riga di codice che lo usa è: *if ((s1State == LOW) || (s2State == LOW)) then...*

Capitolo 11. Comunicazione con il PC

La scheda Arduino si programma via USB, ed una volta terminata la programmazione il collegamento USB può essere usato per far dialogare la scheda Arduino con il PC per il trasferimento di qualsivoglia dato. Per inviare e ricevere semplici caratteri ASCII è possibile usare la funzione serial monitor già disponibile sull'IDE di Arduino ed attivabile tramite il pulsante posto in alto a destra.



Una prima semplice operazione potrebbe essere quella di leggere lo stato di un ingresso digitale e visualizzarne il valore sul PC con la funzione serial monitor; lo sketch da caricare si chiama DigitalReadSerial e si trova già pronto nella sezione 01.Basic degli esempi.

```
/*
  DigitalReadSerial
 Reads a digital input on pin 2, prints the result to the serial monitor
 */

// digital pin 2 has a pushbutton attached to it. Give it a name:
int pushButton = 2;

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  // make the pushbutton's pin an input:
  pinMode(pushButton, INPUT);
}
```

```
// the loop routine runs over and over again forever:
void loop() {
  // read the input pin:
  int buttonState = digitalRead(pushButton);
  // print out the state of the button:
  Serial.println(buttonState);
  delay(100);        // delay in between reads for stability
}
```

Analogamente è possibile inviare semplici caratteri dal PC ad Arduino con lo scopo di eseguire da remoto determinati comandi. Nel seguente esempio inviando il carattere 'h' viene acceso il Led posto sul pin 13 mentre inviando il carattere 'l' viene spento:

```
/*
  Reading a serial char to control LED 13.

 */

// pins for the LEDs:
const int LEDpin = 13;

void setup() {
  // initialize serial:
  Serial.begin(9600);
  // make the pins outputs:
  pinMode(LEDpin, OUTPUT);
}

void loop() {
  // if there's any serial available, read it:
  while (Serial.available() > 0) {

    // look for the next valid integer in the incoming serial stream:
    byte incomingByte = Serial.read();

    if (incomingByte == 'h') digitalWrite(LEDpin, HIGH);
    if (incomingByte == 'l') digitalWrite(LEDpin, LOW);
  }
}
```

Si conclude questa prima parte dedicata all'uso di Arduino per applicazioni digitali.