Capitolo 1

# Benvenuti in Arduino

Il progetto Arduino venne ideato per progettisti e creativi con poca esperienza tecnica. Anche chi non conosceva la programmazione software poteva fare riferimento al progetto Arduino per realizzare sofisticati prototipi e apparecchiature interattive. Non deve pertanto stupire che i primi passi con Arduino siano molto semplici, soprattutto per tecnici esperti di controllo e automazione.

È fondamentale conoscere gli elementi di base di questo sistema di controllo. Si potrà sfruttarne al massimo le potenzialità solo dopo aver appreso il funzionamento della scheda Arduino, dell'ambiente di sviluppo e di altre tecniche particolari, per esempio la comunicazione seriale tra i dispositivi.

Prima di iniziare è bene comprendere il significato di *physical computing*. Chi ha già lavorato nella programmazione dei computer può rimanere perplesso di fronte a questo termine: dopotutto, i computer sono oggetti fisici che accettano segnali (input) da altri oggetti fisici, per esempio da tastiere e mouse, e producono segnali (output) audio e video che vengono riprodotti da altoparlanti e display, anche questi oggetti puramente fisici. Ma allora, perché non affermare che la scienza dei computer è nel suo complesso "physical computing"?

In teoria, la programmazione convenzionale è intesa come un sottoinsieme del physical computing: tastiera e mouse sono *sensori* di segnali di input del mondo reale, mentre display e stampanti sono veri e propri *attuatori*. D'altra parte, è difficile impiegare un computer convenzionale per controllare direttamente il funzionamento di sensori e attuatori, mentre l'utilizzo di una scheda Arduino rende quasi banale il controllo

## In questo capitolo

di dispositivi sofisticati e per certi versi misteriosi. I prossimi capitoli illustreranno le modalità di utilizzo delle schede Arduino, mentre nei prossimi paragrafi verrà introdotto il concetto di physical computing studiando il funzionamento di una scheda di controllo, gli strumenti di lavoro necessari per metterla in funzione e l'installazione e la configurazione di una scheda. Al termine di questo capitolo sarete in grado di affrontare la parte più interessante della trattazione, ovvero lo sviluppo di un primo progetto Arduino.

## Cosa serve

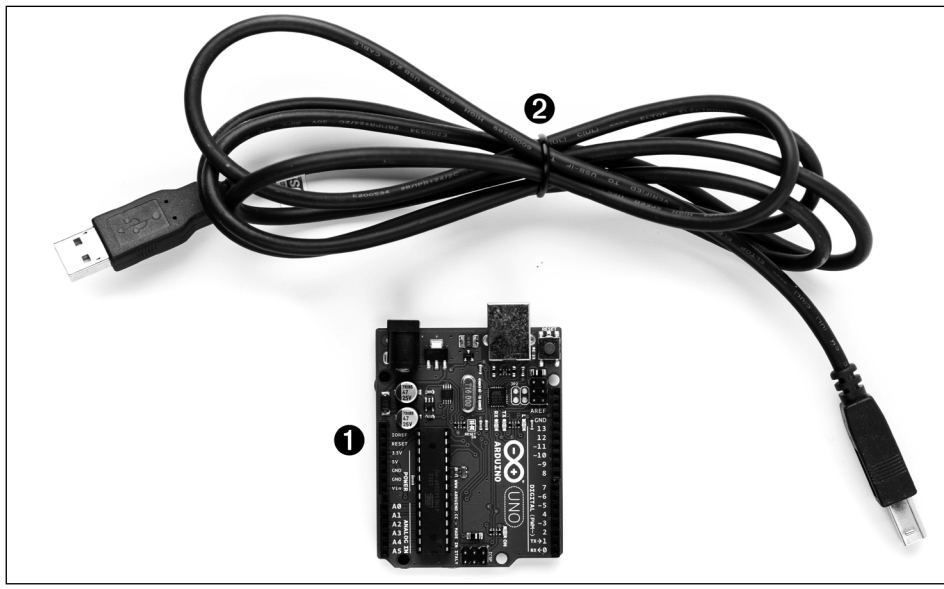


**Figura 1.1** Componenti necessari.

1. Una scheda Arduino, per esempio un modello Arduino Uno, Duemilanove o Diecimila.
2. Un cavo USB per collegare la scheda Arduino al computer.
3. L'IDE Arduino, l'ambiente di sviluppo software che verrà spiegato più avanti in questo capitolo. L'IDE è necessario per realizzare i programmi di tutti i progetti illustrati in questo libro, pertanto la sua presenza non verrà più richiamata esplicitamente.

Troverete figure di questo tipo nella maggior parte dei prossimi capitoli. I numeri nella figura corrispondono ai numeri nell'elenco dei componenti. Nei capitoli successivi le figure non mostreranno i componenti standard, come la scheda Arduino o un cavo USB.

## Cos'è esattamente un progetto Arduino?

A prima vista può non essere chiara la definizione di "progetto Arduino", dato che a un esame preliminare scopriamo nomi stravaganti, per esempio Arduino Uno, Duemilanove,

Diecimila, LilyPad oppure Seeeduino. Il problema nasce dal fatto che non esiste un unico oggetto che possiamo chiamare "Arduino".
Un paio di anni fa il team di Arduino produsse una scheda a microcontrollore che venne rilasciata con una licenza open source. I negozi di elettronica offrono schede di questo tipo già assemblate, ma è sempre possibile scaricare lo schema elettronico di questa scheda (`http://arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf`) e realizzare per conto proprio il sistema a microcontrollore.
Il team di Arduino ha successivamente perfezionato il progetto hardware e ha rilasciato svariate versioni della scheda di controllo, differenti una dall'altra. In genere queste schede sono identificate da un nome italiano, per esempio Uno, Duemilanove o Diecimila; potete trovare online un elenco completo delle schede realizzate dal team di Arduino agli indirizzi `http://arduino.cc/en/Main/Boards` e `http://arduino.cc/en/Main/Products`.
La Figura 1.2 mostra alcune schede Arduino, che risultano diverse nell'aspetto ma che presentano anche molti componenti comuni e possono essere programmate utilizzando gli stessi strumenti e le stesse librerie.

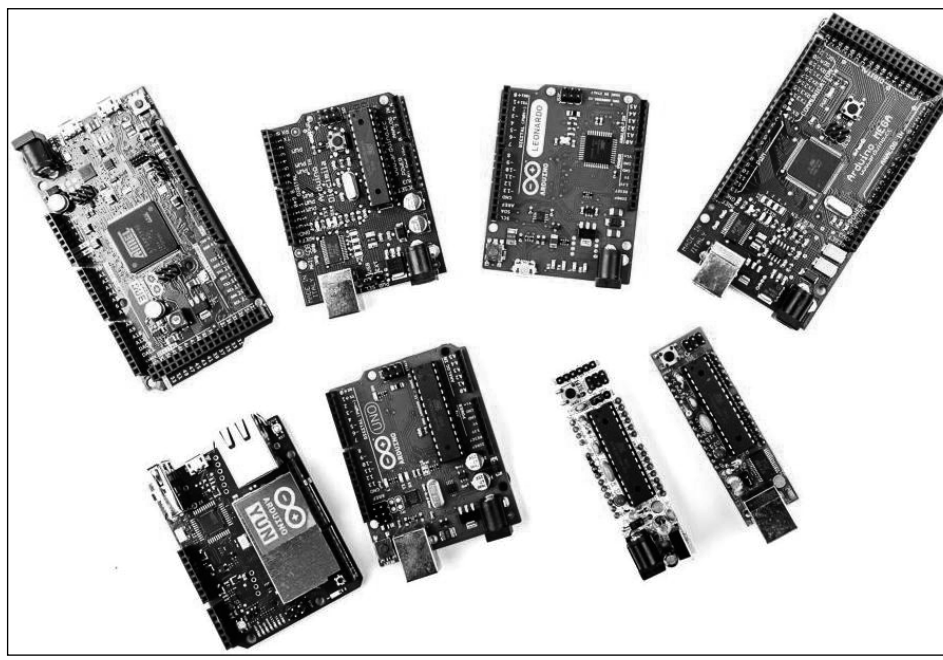


**Figura 1.2** Potete scegliere tra diverse schede Arduino.

Anche se in linea di principio sono identiche, differiscono per alcuni dettagli. La scheda Arduino Due (`http://arduino.cc/en/Main/ArduinoBoardDue`) dispone di un numero di pin di I/O superiore alla maggior parte delle altre schede Arduino e utilizza un potete microcontrollore con core ARM a 32 bit, mentre la scheda Arduino Nano (`http://arduino.cc/en/Main/ArduinoBoardNano`) è stata progettata per l'uso su una breadboard, pertanto non dispone di socket. I principianti dovrebbero iniziare con una delle schede "standard", per esempio la Uno.

Il team di Arduino non si è limitato a migliorare il progetto hardware delle schede a microcontrollore ma ha realizzato anche alcune schede dedicate a un utilizzo particolare; per esempio, la scheda Arduino LilyPad (`http://arduino.cc/en/Main/ArduinoBoardLilyPad`) permette di incorporare la scheda di controllo in un tessuto. Potete così utilizzarla per realizzare t-shirt interattive.
Sul Web potete trovare le schede Arduino "ufficiali" e altri dispositivi cloni. L'utilizzo e la modifica del progetto originale sono consentiti a chiunque e sono molte le persone che hanno realizzato una propria versione di una particolare scheda Arduino.
Potete per esempio trovare schede Freeduino, Seeeduino, Boarduino e l'interessante dispositivo Paperduino (`http://lab.guilhermemartins.net/2009/05/06/paperduino-prints/`), un clone Arduino privo di circuito stampato nel quale tutti i componenti sono incollati a un foglio di carta.
Arduino è un marchio registrato e solo le schede ufficiali sono denominate con questo nome, mentre i diversi cloni hanno in genere un nome che termina con "duino". Per realizzare i progetti illustrati in questo libro siete liberi di utilizzare una qualsiasi scheda clone che sia compatibile con le specifiche originali del progetto Arduino.

## Esplorare la scheda Arduino

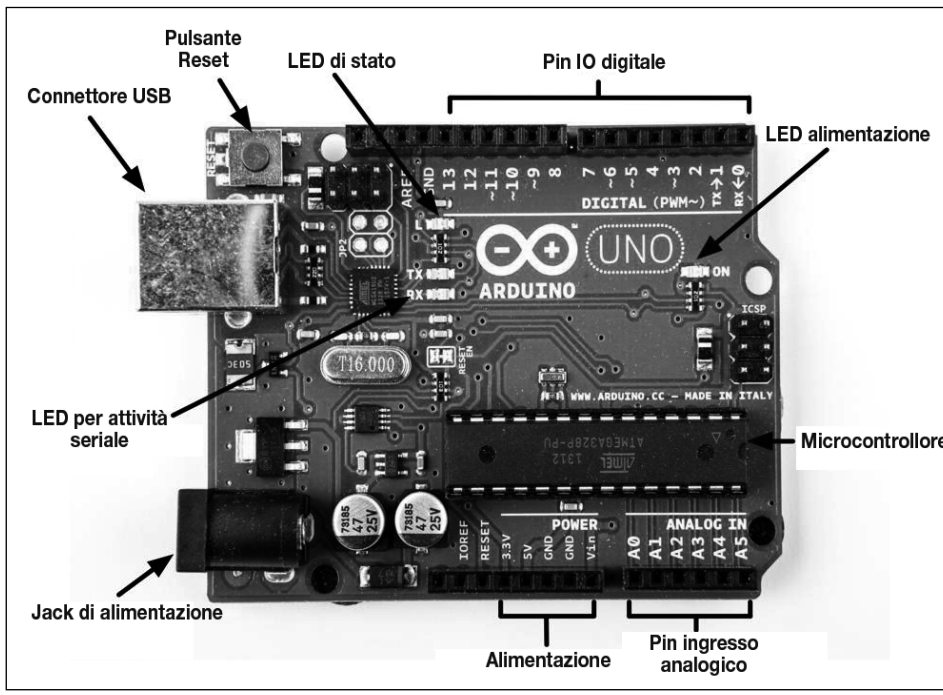


**Figura 1.3** I componenti principali di una scheda Arduino.

La figura mostra una scheda Arduino Uno e i suoi componenti principali. In primo luogo è visibile un connettore USB, che permette di collegare Arduino a un computer tramite

cavo USB. Il tipo di cavo USB dipende dal tipo di scheda Arduino in uso; Arduino Uno è dotata della spina B standard, mentre altre schede, come Arduino Leonardo o Arduino Due, utilizzano le spine micro B.
La connessione tra scheda e computer permette di effettuare le operazioni indicate di seguito.

- Caricare un nuovo programma nella scheda del microcontrollore (come verrà spiegato più avanti in questo capitolo).
- Consentire la comunicazione tra scheda Arduino e computer (si veda il Capitolo 2).
- Fornire alimentazione elettrica alla scheda Arduino.

Arduino è un dispositivo elettronico e in quanto tale deve essere alimentato con energia elettrica. Potete fornire alimentazione collegando per esempio la scheda a un computer tramite la porta USB, anche se questa soluzione non è sempre conveniente. Alcuni progetti non richiedono necessariamente il collegamento della scheda con un computer e sarebbe quantomeno stravagante utilizzare un computer solo per fornire alimentazione elettrica alla scheda del microcontrollore. Occorre anche tenere presente che la porta USB mette a disposizione solo 5 V di alimentazione, ma alcuni progetti richiedono anche tensioni superiori.
In questi casi conviene allora utilizzare un alimentatore in CA da 9 V; in effetti si può impiegare un alimentatore qualsiasi in grado di fornire una tensione compresa tra 7 V e 12 V (`http://www.arduino.cc/playground/Learning/WhatAdapter`). L'alimentatore deve avere un connettore a punta cilindrica di 2,1 mm con positivo centrale: non è necessario comprendere il significato di queste specifiche, potete semplicemente chiedere un alimentatore di questo tipo al vostro fornitore di materiale elettrico. Collegate l'alimentatore al jack corrispondente della scheda Arduino e potete immediatamente mettere in funzione il sistema di controllo, anche se non avete collegato la scheda al computer. La tensione esterna fornita dall'alimentatore è utilizzata dalla scheda anche se collegate Arduino a un computer via USB.

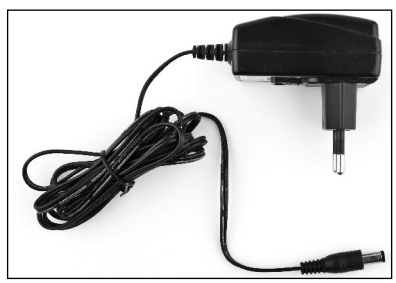

**Figura 1.4** Un tipico alimentatore in CA.

Ricordate però che le prime versioni di schede Arduino (Arduino NG e Diecimila) non commutano automaticamente l'alimentazione elettrica tra alimentatore esterno e tensione USB, ma prevedono un apposito ponticello (*PWR_SEL*) che dovete impostare rispettivamente nella posizione EXT oppure USB, come si può vedere nella Figura 1.5. Ora conoscete due modi per fornire alimentazione elettrica a una scheda Arduino, che peraltro non è avida e permette di condividere con altri dispositivi la tensione che ne

alimenta il funzionamento. Nella parte inferiore della scheda mostrata nella Figura 1.5, si nota che sono presenti più morsetti (o *pin*, in quanto sono collegati internamente ai pin del microcontrollore) relativi a diverse tensioni di alimentazione, come indicato di seguito.

- I pin *3V3 e 5V* permettono di alimentare dispositivi esterni con tensioni di 3,3 V oppure di 5 V.
- I due pin *GND* (*Ground*) condividono il collegamento a massa/terra con la scheda Arduino.
- I progetti che dovranno essere portatili richiedono una tensione di alimentazione fornita da batterie. Potete collegare alla scheda Arduino una sorgente esterna di alimentazione, per esempio una serie di pile o batterie, utilizzando i pin *Vin* e *GND*.

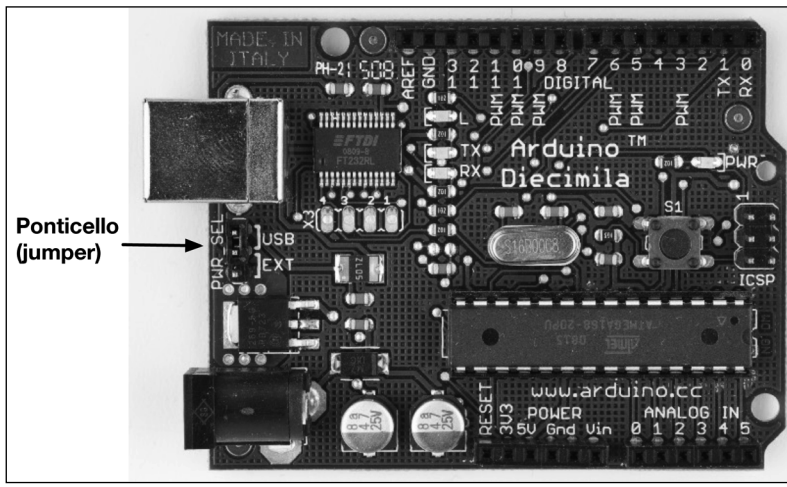


**Figura 1.5** Le schede Arduino meno recenti hanno un ponticello per la selezione della tensione di alimentazione.

Il collegamento di un alimentatore fornisce l'accesso alla tensione di alimentazione della scheda tramite il pin Vin connesso al jack.
Nella parte inferiore destra della scheda potete vedere 6 pin relativi ad altrettanti ingressi analogici denominati da A0 ad A5. Questi pin permettono di collegare più sensori analogici alla scheda Arduino. I dati dei sensori vengono convertiti in un numero compreso tra 0 e 1023. Nel Capitolo 5 si vedrà come utilizzare questa funzionalità per collegare Arduino con un sensore di temperatura.
Nella parte superiore della scheda si possono vedere 14 pin di IO digitale denominati da D0 a D13. In base alle esigenze del progetto questi pin possono controllare segnali di input o di output digitale; potete per esempio leggere lo stato ON e OFF di un pulsante (input digitale) o commutare lo stato del pin da ON a OFF per accendere e spegnere un led (output digitale), come si vedrà nel Capitolo 3. Sei di questi pin (D3, D5, D6, D9, D10, D11) possono anche essere utilizzati come output analogici; in questo caso i pin presentano una tensione analogica che dipende da un valore numerico compreso tra 0 e 255.

## Segnali analogici e digitali

Quasi tutti i fenomeni fisici hanno a che fare con grandezze analogiche. Ogni volta che si osserva un evento naturale, per esempio elettrico o acustico, si sta in realtà ricevendo un segnale analogico. Una delle proprietà fondamentali dei segnali analogici è che sono continui. In un determinato istante di tempo è sempre possibile misurare l'intensità del segnale e da un punto di vista teorico si può rilevare una variazione minima di qualsiasi segnale analogico.

Se la Natura è sostanzialmente analogica, è altrettanto vero che stiamo vivendo nell'era digitale. Fin dalla comparsa dei primi computer, qualche decina di anni fa, ci si rese immediatamente conto che è molto più semplice elaborare le informazioni del mondo reale dopo averle rappresentate in forma numerica (digitale) piuttosto che come segnale analogico di tensione o di volume. Per esempio, è molto più semplice elaborare segnali audio con un computer dopo che le onde sonore sono state memorizzate come sequenza di numeri, dove ogni numero della sequenza descrive l'intensità del segnale in un determinato istante di tempo. Invece di memorizzare il segnale analogico in modo continuo (come veniva fatto per i dischi in vinile), la "misura" del segnale è eseguita solo in particolari istanti di tempo (Figura 1.6); questa tecnica di registrazione prende il nome di *campionamento* e i valori memorizzati prendono il nome di *campioni*. La frequenza di rilevazione dei campioni è chiamata *sampling rate*. Nel caso di un CD audio il sampling rate è di 44,1 kHz, ovvero si memorizzano 44.100 campioni al secondo.

La digitalizzazione di un segnale analogico richiede inoltre la limitazione della misura dei campioni entro un certo intervallo di valori. In un CD audio ogni campione è rappresentato da 16 bit; nella Figura 1.6 si può notare che l'intervallo di campionamento è definito da due linee tratteggiate orizzontali: nell'esempio è stato "tagliato" un picco del segnale visibile all'inizio del campionamento.

Arduino permette di effettuare il collegamento con dispositivi analogici e digitali, anche se in genere non ci si deve occupare più di tanto della forma del segnale da controllare, dato che la scheda Arduino è in grado di eseguire automaticamente la conversione tra segnali analogici e digitali o viceversa.

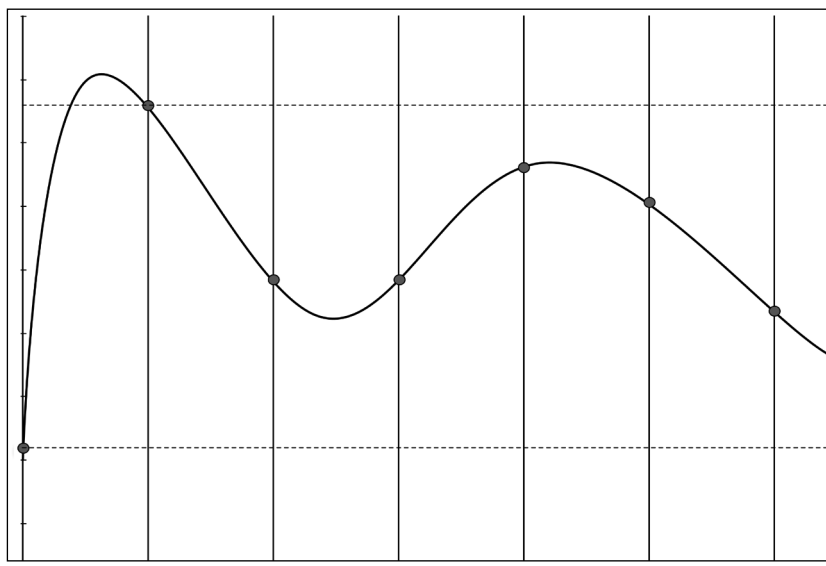

**Figura 1.6** Digitalizzazione di un segnale analogico.

I pin descritti finora sono collegati direttamente a un microcontrollore, che combina le funzioni tipiche di una CPU e di svariati dispositivi periferici, che riguardano per esempio il controllo dei canali IO. Esistono molti tipi di microcontrollore, ma in genere Arduino propone schede che montano un ATmega328, un microcontrollore prodotto da Atmel. Esistono poi modelli di Arduino (per esempio Arduino Mega o Arduino Due) che utilizzano microcontrollori più potenti.
I computer caricano i programmi da eseguire leggendoli da un disco fisso, mentre i microcontrollori devono essere programmati direttamente; ciò significa che dovete caricare il software nel microcontrollore tramite un collegamento via cavo. Dopo aver caricato un programma, questo rimane installato nel microcontrollore fintanto che non viene sovrascritto da un nuovo programma.
È sufficiente fornire alimentazione a una scheda Arduino per mandare in esecuzione il programma attualmente memorizzato nel microcontrollore presente nella scheda. A volte si richiede di avviare Arduino dalla prima istruzione del programma, operazione che si può effettuare utilizzando il pulsante *Reset* visibile lungo il lato destro della scheda. Premete *Reset* per inizializzare il funzionamento della scheda e avviare dall'inizio il programma memorizzato nella scheda, come si vedrà nel Capitolo 3.
Sulla maggior parte delle schede Arduino sono presenti anche un paio di LED, che descriveremo in un paragrafo successivo.

# Installazione dell'IDE Arduino

I responsabili del progetto Arduino hanno realizzato un ambiente di sviluppo, o IDE (*Integrated Development Environment*), che facilita l'apprendimento delle funzioni che si possono svolgere con questo genere di microcontrollore. L'IDE può essere eseguito utilizzando diversi sistemi operativi e deve essere installato rispettando le indicazioni illustrate di seguito.
Nota importante: nel periodo in cui è stato scritto questo libro esistevano due diverse versioni dell'IDE (1.0.6 e 1.6.0; ne esiste anche un'altra per Arduino Galileo all'indirizzo `https://communities.intel.com/docs/DOC-22226`). Ragionevolmente la versione 1.0.x non sarà più supportata in futuro, quindi è preferibile utilizzare la versione 1.6.x limitando l'impiego dalla 1.0.x solo ai casi in cui sono necessarie librerie che non funzionano con la 1.6.x. Le istruzioni di seguito si riferiscono alla versione 1.6.0.

## Installare l'IDE Arduino in Windows

L'IDE funziona con tutte le versioni più recenti di Windows, per esempio Windows 8.1 e Windows 7. Il software è disponibile come programma di installazione Windows o come archivio ZIP autocontenuto; visitate la pagina di download di Arduino (`http://arduino.cc/en/Main/Software`) per ottenere la versione più recente di entrambi i file.
Se disponete dei privilegi di amministrazione sul computer, scegliete l'installer, che oltre all'IDE installa anche tutti i driver necessari. In questo caso di solito non serve eseguire altre operazioni ed è possibile utilizzare subito l'IDE. Se non avete i privilegi di amministrazione, scaricate l'archivio ZIP ed estraete i file contenuti in una directory sul vostro disco fisso.

Prima di avviare l'IDE dovete installare i driver richiesti dalla porta USB di Arduino. La procedura di installazione dipende dal tipo di scheda che avete a disposizione e dalla versione di Windows

## Installare i driver per le schede Arduino attuali

Per installare i driver per le schede recenti, come Arduino Uno, dovete innanzitutto collegare Arduino a una porta USB del computer per avviare la procedura automatica. È raro che questo processo non riesca. Aprite il pannello di controllo del sistema e quindi *Device Manager* (lo trovate nella sezione *System and Security*, `http://windows.microsoft.com/en-us/windows/open-device-manager#1TC=windows-7`). Nella sezione *Ports (COM & LPT)*, vedrete con ogni probabilità la voce *Arduino Uno (COMxx)*.

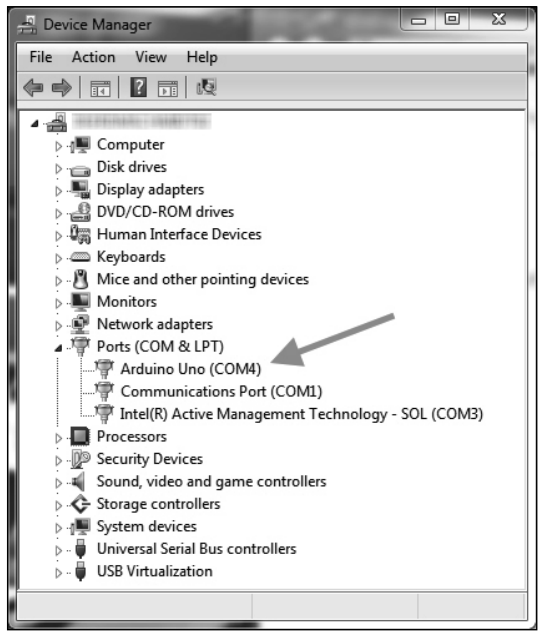


**Figura 1.7** La scheda Arduino è mostrata nella sezione Ports (COM & LPT).

Se non la trovate, cercate *Unknown device* nel menu *Other devices* (Figura 1.8).
Fate clic con il pulsante destro del mouse sulla voce relativa alla scheda Arduino e scegliete *Update Driver Software*. Scegliete l'opzione *Browse My Computer for Driver Software*, aprite la cartella *drivers* dell'archivio che avete estratto e selezionate il file `arduino.inf` (Figura 1.9). Nelle versioni precedenti dell'IDE il file era chiamato `Arduino UNO.inf`.
Dopo aver installato il driver, potete avviare l'IDE di Arduino e lavorare con la scheda (se utilizzate Windows 8.x, dovrete disabilitare alcuni meccanismi di protezione prima di installare il driver, come spiegato in `https://learn.sparkfun.com/tutorials/installing-arduino-ide/windows`).

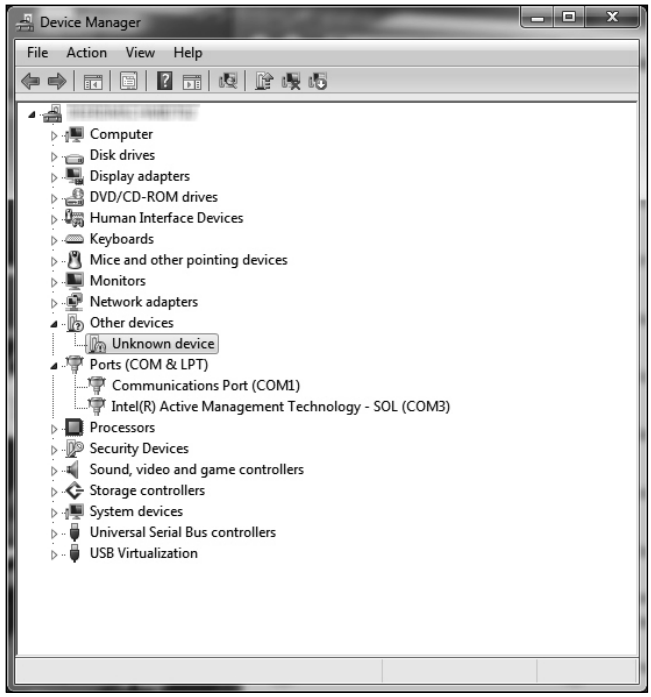


**Figura 1.8** A volte la scheda Arduino non viene riconosciuta.

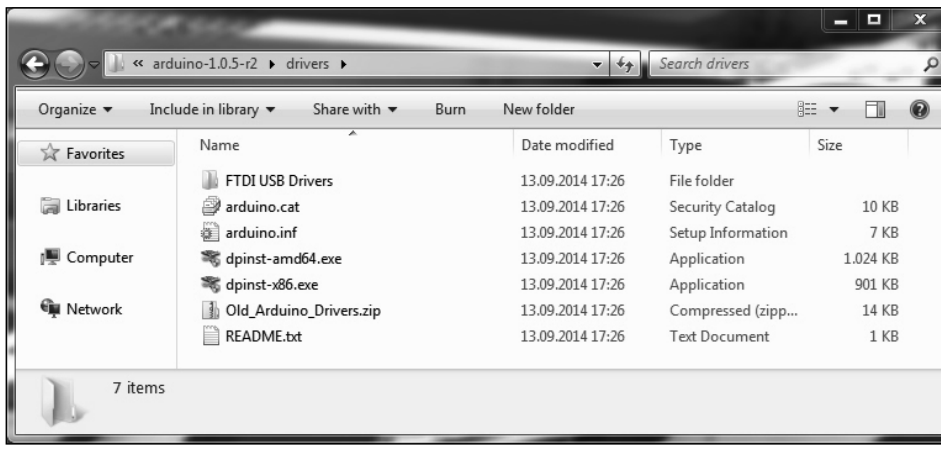


**Figura 1.9** Il contenuto della cartella drivers.

## Installare i driver per le schede Arduino precedenti

L'installazione dei driver per le schede precedenti (Duemilanove, Diecimila o Nano) è leggermente diversa. Ovviamente, per prima cosa dovete sempre collegare la scheda.

Windows Vista dovrebbe avviare automaticamente l'installazione del driver. Non dovete far altro che osservare i messaggi visualizzati dalla procedura e attendere che il computer segnali la possibilità di utilizzare il nuovo hardware USB.
Windows 8.x, Windows 7 e Windows XP potrebbero non rilevare automaticamente i driver tra quelli disponibili nei siti di aggiornamento di Microsoft. A un certo punto la procedura di installazione chiede di indicare il percorso per individuare i driver USB, dopo aver escluso l'installazione automatica dei driver da Internet. A seconda del tipo di scheda Arduino, dovrete individuare la posizione giusta all'interno della directory *drivers/FTDI USB Drivers*.
Dopo aver installato i driver potete avviare con un doppio clic il file eseguibile `Arduino` che trovate nella directory principale dell'archivio estratto in precedenza. Tenete presente che i driver USB non vengono modificati spesso, a differenza delle versioni di IDE, che possono essere aggiornate più frequentemente. Ogni volta che installate una nuova versione dell'ambiente di sviluppo è bene verificare se sono presenti anche nuovi driver USB da installare, operazione che in genere non è però necessaria.

## Installare l'IDE Arduino in OS X

L'IDE Arduino per OS X è disponibile come archivio ZIP (`http://arduino.cc/en/Main/Software`). L'IDE Arduino dipende dalla Java Virtual Machine e nel momento in cui vengono scritte queste righe è disponibile per Java 6 (raccomandato) e Java 7 (sperimentale). Scaricatela, apritela con un doppio clic e trascinate l'icona di Arduino nella cartella *Applicazioni*. Se non avete ancora installato Java OS X vi chiederà il permesso di farlo.
Chi utilizza Arduino Uno oppure Arduino Mega 2560 può avviare direttamente l'IDE. Dovete invece installare i driver relativi alla porta seriale di Arduino nel caso in cui stiate utilizzando una scheda meno recente, per esempio Duemilanove, Diecimila o Nano. Potete trovare la versione più recente online (`http://www.ftdichip.com/Drivers/VCP.htm`). Scaricate il pacchetto per la piattaforma in uso (di solito ha un nome simile a `FTDIUSBSerialDriver_10_4_10_5_10_6.mpkg`), fate doppio clic sulla sua icona e seguite le istruzioni di installazione fornite a video.
L'installazione di una nuova versione dell'IDE non richiede in genere l'installazione di nuovi driver, che va effettuata solo quando diventa disponibile una versione aggiornata degli stessi.

## Installare l'IDE Arduino in Linux

In Linux le procedure di installazione possono risultare differenti. L'IDE Arduino funziona correttamente con quasi tutte le versioni recenti di Linux, ma la procedura di installazione dipende dalla distribuzione che si sta impiegando. Va inoltre ricordato che spesso è necessario installare elementi software aggiuntivi (per esempio la Java Virtual Machine), che sono invece preinstallati in altri sistemi operativi.
Si suggerisce di verificare la documentazione ufficiale del progetto (`http://www.arduino.cc/playground/Learning/Linux`) per individuare le istruzioni che riguardano la distribuzione Linux che vi interessa. Dopo aver installato driver e IDE si può iniziare a esaminare le caratteristiche di questo ambiente di sviluppo.

## Conoscere l'IDE Arduino

L'IDE Arduino è semplice rispetto ad altri ambienti IDE quali Eclipse, Xcode oppure Microsoft Visual Studio. È costituito sostanzialmente da un editor, un compilatore, un loader e un monitor seriale, come si può vedere nella Figura 1.10, o meglio ancora avviando l'IDE sul vostro computer.

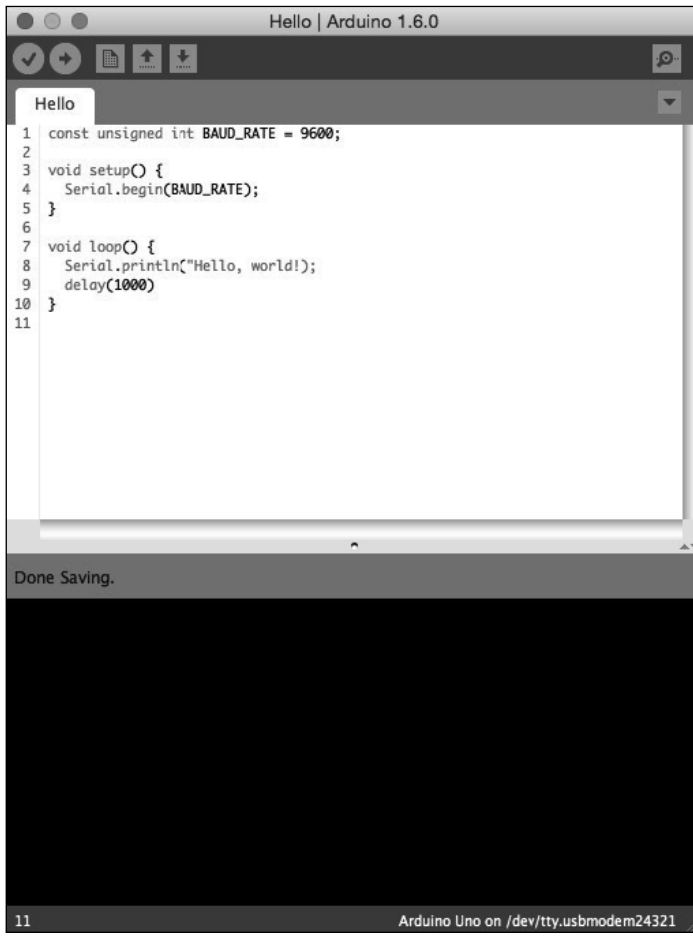


**Figura 1.10** L'IDE Arduino si presenta in modo chiaro.

Non sono presenti altre funzioni avanzate, per esempio un debugger o strumenti di completamento automatico delle istruzioni. Potete modificare una piccola serie di preferenze e, analogamente alle applicazioni Java, l'IDE Arduino non si integra pienamente con le funzioni della scrivania OS X. L'ambiente può comunque essere utilizzato a piacere e mette a disposizione un supporto adeguato per la gestione dei progetti Arduino. La Figura 1.11 mostra la barra degli strumenti dell'IDE che permette di accedere alle funzioni principali.

- Il pulsante *Verify* compila il programma attualmente presente nell'editor. Da un certo punto di vista il nome del pulsante è improprio, dato che l'attivazione di questa funzione non si limita a verificare la sintassi del programma ma lo converte in una forma compatibile con la scheda Arduino. Potete chiamare questa funzione utilizzando la scelta rapida da tastiera MELA+R su OS X o Ctrl+R su tutti gli altri sistemi.
- Fate clic sul pulsante *Upload* (MELA+U o Ctrl+U) per compilare il programma e caricarlo sulla scheda Arduino che avete configurato tramite il menu *Tools* > *Serial Port*, come si vedrà più avanti in questo capitolo.
- Il pulsante *New* (MELA+N o Ctrl+N) crea un nuovo programma svuotando il contenuto della finestra dell'editor. Prima di eseguire questa operazione l'IDE offre la possibilità di memorizzare le modifiche apportate dopo l'ultimo salvataggio.
- Con il pulsante *Open* (MELA+O o Ctrl+O) selezionate e aprite un programma già memorizzato nel computer.
- Il pulsante *Save* (MELA+S o Ctrl+S) memorizza il programma presente nell'editor.
- La scheda Arduino può comunicare con un computer attraverso la porta seriale. Fate clic sul pulsante *Serial Monitor* (Shift+MELA+N o Ctrl+Maiusc+N) per aprire una finestra omonima che permette di esaminare i dati inviati da Arduino e trasmettere dati alla scheda.

L'utilizzo dell'IDE è abbastanza semplice, ma si possono sempre verificare problemi di funzionamento oppure si può voler eseguire operazioni particolari. In questi casi conviene fare riferimento al menu *Help* per consultare le risorse utili offerte dal sito web del progetto Arduino e studiare le soluzioni proposte per risolvere non solo i problemi più comuni ma, anche per avere a disposizione altra documentazione e tutorial.

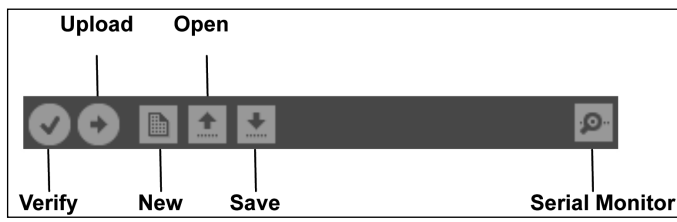


**Figura 1.11** La barra degli strumenti dell'IDE permette di accedere velocemente alle funzioni principali.

## Hello, World!

Per iniziare a familiarizzare con le funzioni principali dell'IDE provate a realizzare un semplice programma che fa lampeggiare un led (*Light-Emitting Diode*). Un led è una sorgente di luce efficiente ed economica, già presente anche su ogni scheda Arduino, dove un led segnala l'alimentazione elettrica della scheda, mentre altri due lampeggiano per segnalare la trasmissione o la ricezione di dati attraverso la connessione seriale (Figura 1.12).

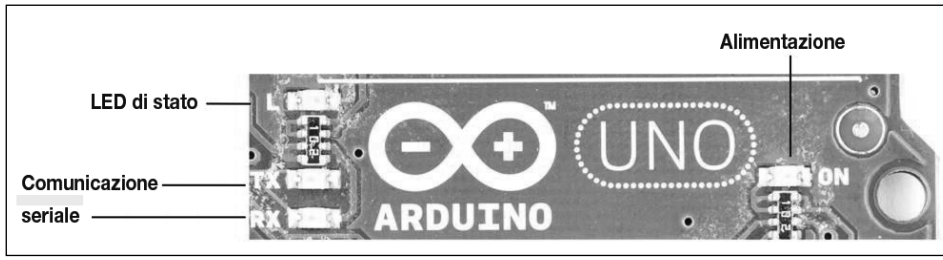


**Figura 1.12** La scheda Arduino presenta una serie di led.

Questo primo progetto si propone di far lampeggiare il led di stato della scheda Arduino. Il led di stato è collegato al pin di IO digitale 13. I pin digitali possono essere considerati una sorta di interruttore che si può trovare in uno dei due stati HIGH oppure LOW. Lo stato HIGH corrisponde alla presenza di 5 V di tensione sul pin, il che permette di far scorrere una corrente sul led che si può così accendere. Lo stato LOW interrompe il passaggio di corrente e il led si spegne. Non è necessario approfondire ora lo studio del comportamento elettrico di un led, ma se vi interessa l'argomento potete consultare direttamente l'Appendice A.
Aprite l'IDE e digitate nell'editor le istruzioni riportate di seguito.

Welcome/HelloWorld/HelloWorld.ino

```
Riga 1  const unsigned int LED_PIN = 13;
     -  const unsigned int PAUSE = 500;
     -
     -  void setup() {
     5    pinMode(LED_PIN, OUTPUT);
     -  }
     -
     -  void loop() {
     -    digitalWrite(LED_PIN, HIGH);
    10    delay(PAUSE);
     -    digitalWrite(LED_PIN, LOW);
     -    delay(PAUSE);
     -  }
```

Conviene studiare il funzionamento del programma esaminando il significato delle singole istruzioni. Nelle prime due righe si definiscono le costanti `unsigned int` tramite la parola chiave `const`. `LED_PIN` indica il pin digitale IO che si sta utilizzando, mentre `PAUSE` imposta l'intervallo di tempo (in millisecondi) di accensione del led.
Ogni programma Arduino richiede la presenza di una funzione `setup`, che in questo primo progetto ha inizio alla riga 4. Di seguito è riportata la sintassi tipica che definisce una funzione:

```
<tipo del valore restituito> <nome funzione> '(' <elenco di parametri> ')'
```

In questo programma la funzione è denominata `setup` e il valore restituito è di tipo `void` in quanto non viene restituito alcun valore. L'istruzione `setup` non si aspetta alcun argomento e per questo motivo l'elenco dei parametri è vuoto. Prima di proseguire è bene saperne di più sui tipi di dati accettati da Arduino.

## Tipi di dati in Arduino

Qualsiasi dato presente in un programma Arduino deve essere associato a un tipo di dato. Scegliete il tipo che meglio soddisfa le esigenze del programma di controllo tra le opzioni indicate di seguito.

- I tipi `boolean` occupano un byte di memoria e assumono il valore `true` oppure `false`.
- Le variabili `char` occupano un byte di memoria e memorizzano valori numerici compresi tra -128 e 127. I numeri corrispondono a caratteri espressi nel codice ASCII; per esempio, di seguito sono riportate due variabili `c1` e `c2` che hanno lo stesso valore:

```
char c1 = 'A';
char c2 = 65;
```

  Osservate l'utilizzo della virgoletta singola per indicare letterali di tipo `char`.
- Le variabili `byte` occupano un byte e memorizzano valori numerici compresi tra 0 e 255.
- Una variabile `int` occupa due byte di memoria e permette di memorizzare valori numerici compresi tra -32.768 e 32.767. Anche la variabile senza segno `unsigned int` occupa due byte e può memorizzare valori compresi tra 0 e 65.535.
- Valori numerici più elevati richiedono l'impostazione di tipo `long`. Questa variabile occupa quattro byte di memoria e memorizza valori compresi tra -2.147.483.648 e 2.147.483.647. La variabile senza segno `unsigned long` richiede sempre quattro byte e accetta valori compresi tra 0 e 4.294.967.295.
- Le variabili `float` e `double` assumono al momento il medesimo significato sulla maggior parte delle schede Arduino e le potete utilizzare per memorizzare numeri a virgola mobile. Queste variabili occupano quattro byte di memoria e accettano valori compresi tra -3,4028235E+38 e 3,4028235E+38. Sulla scheda Arduino Due, i valori `double` sono più precisi e occupano otto byte di memoria.
- Utilizzate il tipo `void` solo nelle dichiarazioni delle funzioni. Questo tipo imposta una funzione che non restituisce alcun valore.
- Gli array memorizzano insiemi di valori dello stesso tipo:

```
int values[2];              // Un array di due elementi
values[0] = 42;             // Imposta il primo elemento
values[1] = -42;            // Imposta il secondo elemento
int more_values[] = { 42, -42 };
int first = more_values[0];  // first == 42
```

  Gli array `values` e `more_values` contengono gli stessi elementi e le istruzioni evidenziano semplicemente due modalità diverse di inizializzazione di un array. Ricordate che l'indice di un array inizia con il valore 0 e che eventuali elementi di un array privi di inizializzazione contengono valori inaffidabili.
- Una stringa è un array di valori `char`. L'ambiente Arduino supporta la creazione di stringhe con una sintassi piuttosto "elastica"; tutte le dichiarazioni riportate di seguito impostano stringhe che hanno lo stesso contenuto:

```
char string1[8] = { 'A', 'r', 'd', 'u', 'i', 'n', 'o', '\0' };
char string2[]  = "Arduino";
char string3[8] = "Arduino";
char string4[]  = { 65, 114, 100, 117, 105, 110, 111, 0 };
```

Le stringhe devono sempre concludersi con un byte a zero. L'inserimento del byte a zero è automatico nel caso in cui creiate una stringa utilizzando le virgolette doppie; dovete però aggiungere un byte quando determinate la lunghezza reale dell'array corrispondente a una determinata stringa.

Nel Capitolo 11 si vedrà come utilizzare la nuova classe `String` di Arduino, che permette di lavorare con le stringhe in modo più comodo e sicuro.

## Funzioni di Arduino

Arduino chiama la funzione `setup` quando viene collegata l'alimentazione elettrica; nel precedente esempio *Hello, World!* è stata impiegata la medesima funzione per inizializzare la scheda e l'hardware collegato a questa. Nel programma si utilizza il metodo `pinMode` per configurare il pin 13 come pin di output. In questo modo il pin è in grado di fornire la corrente necessaria per accendere il led. Lo stato di default di un pin è `INPUT`; le parole *INPUT* e *OUTPUT* sono costanti predefinite, come si può vedere consultando la documentazione ufficiale del progetto (`http://arduino.cc/en/Tutorial/DigitalPins`).

Un'altra funzione obbligatoria è chiamata `loop` e nel primo progetto di esempio ha inizio alla riga 8. Questa funzione contiene la logica principale del programma e viene utilizzata da Arduino per creare un loop infinito. La logica di questo programma deve in primo luogo accendere il led collegato al pin 13. Per effettuare questa operazione si utilizza il metodo `digitalWrite` passando come parametri il numero del pin e la costante `HIGH`. In questo modo il pin fornisce in uscita 5 V fino alla modifica successiva e il led collegato al pin si accende.

A questo punto il programma chiama il metodo `delay` e attende 500 millisecondi senza eseguire alcuna operazione. Durante questo intervallo di tempo il pin 13 rimane in stato `HIGH` e il led è sempre acceso. Il led si spegne quando si imposta di nuovo a `LOW` lo stato del pin utilizzando ancora una volta il metodo `digitalWrite`. Il programma attende ancora per 500 millisecondi prima di concludere la funzione `loop`. Il programma Arduino viene avviato di nuovo e il led torna ad accendersi. Nel prossimo paragrafo si vedrà come mettere in funzione il programma e trasferirlo nella scheda Arduino.

# Compilare e caricare i programmi

Prima di compilare e caricare un programma è necessario configurare due impostazioni dell'IDE, che riguardano il tipo di scheda Arduino che si vuole impiegare e la porta seriale di collegamento tra scheda e computer.

A partire dalla versione 1.6.0, l'IDE cerca di identificare automaticamente la scheda connessa al computer. Questo meccanismo funziona abbastanza bene, ma non sempre. Per questo è utile saper identificare manualmente la scheda Arduino e la porta seriale utilizzate. È facile identificare il tipo di scheda Arduino, dato che questa informazione è stampata sulla scheda. I tipi più diffusi sono chiamati Uno, Duemilanove, Diecimila, Nano, Mega, Mini, NG, BT, LilyPad, Pro oppure Pro Mini. In alcuni casi dovete anche verificare il tipo di microcontrollore installato sulla scheda, che in genere corrisponde a un componente ATmega328. Il modello di microcontrollore è stampato sul corpo del componente. Identificate il tipo esatto di Arduino a disposizione e selezionate il modello corrispondente nel menu *Tools > Board*.

A questo punto dovete scegliere la porta seriale di collegamento tra Arduino e computer dal menu *Tools* > *Serial Port*. In OS X il nome delle porta seriale inizia solitamente con */dev/tty.usbserial* oppure */dev/tty.usbmodem* (nel MacBook Pro dell'autore di questo libro la porta è identificata come */dev/tty.usbmodem24321*).
Nei sistemi Linux si dovrebbe identificare la porta */dev/ttyUSB0*, */dev/ttyUSB1* o una porta analoga; il numero finale dipende dalla quantità di porte USB disponibili nel computer.
In Windows dovete utilizzare *Device Manager* per trovare la porta seriale giusta. In *Device Manager*, individuatela dal menu *Ports* (*COM & LPT*), come si può vedere nella Figura 1.7. In genere la porta seriale è denominata *COM1*, *COM2* o qualcosa di simile.
Dopo aver selezionato correttamente la porta seriale fate clic sul pulsante *Verify* del menu IDE, in modo da visualizzare un messaggio simile a quello che segue (ricordate che l'IDE Arduino identifica i programmi chiamandoli *sketch*):

```
Build options changed, rebuilding all Sketch uses 1,030 bytes (3%) of program storage
space. Maximum is 32,256 bytes. Global variables use 9 bytes (0%) of dynamic memory,
leaving 2,039 bytes for local variables. Maximum is 2,048 bytes.
```

Questo messaggio segnala che l'IDE ha compilato con successo il codice sorgente convertendolo in 1030 byte di codice macchina da caricare nella scheda Arduino. Se compare un messaggio di errore dovete controllare di aver digitato correttamente le istruzioni; nel dubbio, potete scaricare il codice degli esempi collegandovi alla pagina web dedicata al libro (`http://www.pragprog.com/titles/msard2`). Il numero massimo di byte riportato nel messaggio dipende dal tipo di scheda Arduino a disposizione; per esempio, Arduino Duemilanove indica in genere 14336 byte. Anche le dimensioni dello sketch potrebbero essere leggermente diverse in base alla versione dell'IDE Arduino.
A questo punto fate clic sul pulsante *Upload*. Dopo alcuni secondi dovrebbe comparire un messaggio simile a questo:

```
Sketch uses 1,030 bytes (3%) of program storage space. Maximum is 32,256 bytes. Global
variables use 9 bytes (0%) of dynamic memory, leaving 2,039 bytes for local variables.
Maximum is 2,048 bytes.
```

Si tratta dello stesso messaggio visualizzato al termine della compilazione del programma, ma questa volta segnala che 1030 byte di codice macchina sono stati trasferiti con successo sulla scheda Arduino. In caso di errore dovete controllare nel menu *Tools* di aver selezionato correttamente il tipo di scheda e di porta seriale.
In fase di caricamento del programma i led TX ed RX lampeggiano per alcuni secondi; ciò è normale e si verifica ogni volta che Arduino e computer comunicano attraverso la porta seriale. Il led TX si accende quando Arduino invia informazioni al computer, mentre si accende il led RX quando Arduino riceve bit di dati. La comunicazione è decisamente rapida, pertanto i led lampeggiano e non è possibile identificare la trasmissione di ogni singolo byte.
Il programma è eseguito non appena si conclude la trasmissione del codice su Arduino. In questo programma ciò diventa evidente quando il led inizia a lampeggiare e rimane acceso per mezzo secondo, spento per il successivo mezzo secondo e così via.
La Figura 1.13 mostra un grafico che rappresenta la tensione fornita dal pin 13 durante l'esecuzione del programma. Il pin si trova inizialmente in stato LOW e non genera alcuna corrente in uscita. Viene utilizzato `digitalWrite` per impostare lo stato HIGH e

fornire 5 V in uscita per 500 millisecondi. Successivamente, il software imposta lo stato LOW per 500 millisecondi, poi ripete la medesima procedura.
Ecco fatto! Avete appena realizzato il vostro primo progetto di *physical computing*.
Bisogna ammettere che il led di stato non è particolarmente spettacolare. Nel Capitolo 3 collegheremo "veri" LED alla scheda Arduino.

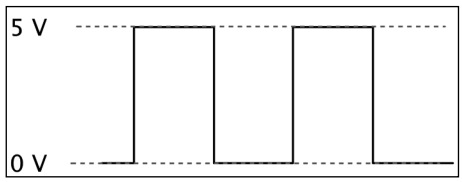


**Figura 1.13** Cosa succede nel pin 13 quando il led lampeggia.

I concetti e le operazioni illustrate in questo capitolo sono utili per realizzare quasi tutti i progetti Arduino. Nel prossimo capitolo vedrete come controllare in modo più accurato il funzionamento dei led e imparerete a utilizzare altre funzioni dell'IDE.

## Cosa fare se non funziona?

L'impostazione non corretta della porta seriale è l'errore più comune durante i primi esperimenti con una scheda Arduino. Un messaggio "Serial port already in use" in fase di caricamento di un programma significa che dovete controllare la porta seriale che avete selezionato nel menu *Tools > Serial Port*. Messaggi quali "Problem uploading to board" oppure "Programmer is not responding" richiedono invece di controllare l'impostazione della scheda Arduino nel menu *Tools > Board*.
Analogamente a qualsiasi altro codice informatico, anche i programmi Arduino possono contenere bug; il compilatore rileverà gli errori di scrittura e di sintassi. Nella Figura 1.14 è riportato un tipico messaggio di errore segnalato dal compilatore.
Invece di digitare **pinMode** è stato scritto **pinMod**: dato che il compilatore non ha trovato una funzione denominata in questo modo, la procedura si è interrotta ed è stato emesso un messaggio di errore. L'IDE Arduino evidenzia la riga che ha bloccato la compilazione, mostra l'istruzione su sfondo in colore giallo e visualizza un messaggio che permette di identificare la causa dell'errore.

**RIFERIMENTO**

Altri bug possono risultare più complessi da identificare e richiedono di studiare accuratamente il codice o di utilizzare sofisticate tecniche di debugging. Il libro di Paul Butcher *Debug It! Find, Repair, and Prevent Bugs in Your Code* (The Pragmatic Bookshelf, 2009) fornisce ampie delucidazioni su questo argomento.

# Esercizi

- Generate diverse modalità di accensione/spegnimento del led impostando più intervalli di pausa e modificate la durata delle singole pause, che non devono necessariamente essere identiche tra loro. Provate a impostare pause molto brevi che facciano lampeggiare il led a frequenza elevata. Sapete spiegare l'effetto che si produce in questi casi?
- Fate in modo che il led di stato visualizzi il vostro nome in codice Morse (`http://it.wikipedia.org/wiki/Codice_Morse`).

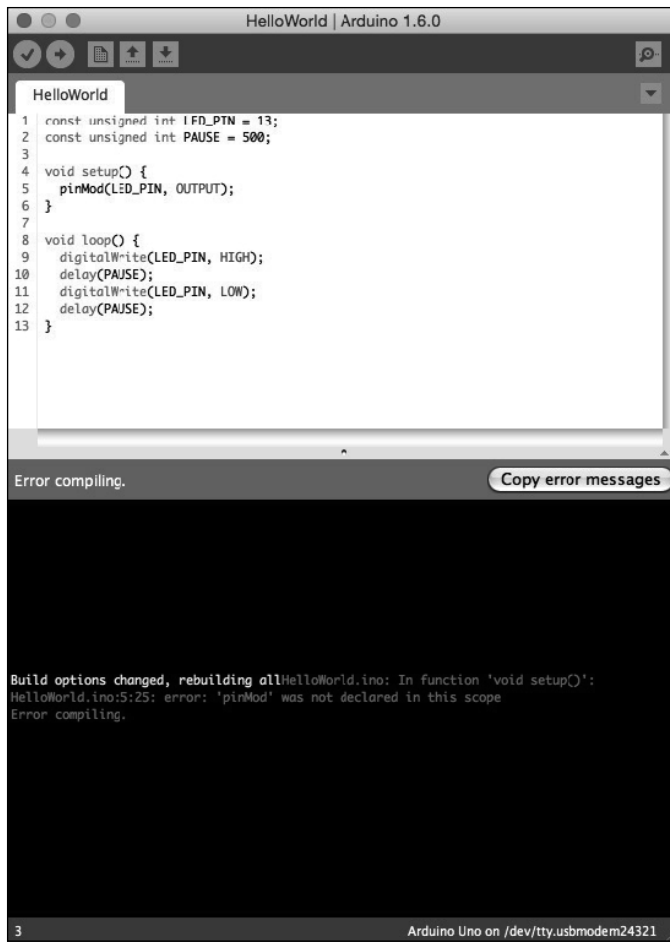


**Figura 1.14** L'IDE Arduino visualizza espliciti messaggi di errore.