2 Cosa vi occorre

Sapete cos'è Arduino; adesso scopritene la dotazione e iniziate ad allestire la vostra officina creativo-tecnologica che vi permetterà di imparare ad usarlo e a sperimentare con esso tante interessanti applicazioni.

Con Arduino potete trasformare in pratica qualsiasi idea che la vostra creatività vi suggerirà; portò come tutti gli strumenti, anche quelli più potenti e completi, vi richiede un minimo di sforzo per conoscerlo e per imparare a utilizzarlo.
Non pretendete di imparare tutto subito: inizierete col vedere cosa fanno i programmi di esempio delle esercitazioni che via-via vi proporremo, poi procederete modificando gli sketch che qualcuno ha già sviluppato per applicazioni simili a quella che vorrete realizzare; dopo, sicuramente ci prenderete gusto e comincerete a scrivere le vostre prime, timide righe di codice.
Che siate alle prime armi o già collaudati, ricordate che il web offre molti suggerimenti e la condivisione di esperienze con altri utenti è la via probabilmente migliore per crescere alla svelta. Il consiglio è di iscriversi al forum ufficiale Arduino e chiedere agli utenti "esperti"; non abbiate timore né complessi di inferiorità, perché tutti gli esperti sono stati principianti e a loro volta hanno chiesto aiuto, perciò comprenderanno. Naturalmente, non scordate che quando si entra in un forum in Rete bisogna presentarsi e, prima di fare domande, dovete cercare e leggere quanto è stato già detto e scritto nei post degli altri partecipanti; insomma, l'aiuto è gratis ma dovete guadagnarvelo, non aspettare il "piatto pronto".
Per introdurvi nel mondo di Arduino vi faremo lavorare sulla scheda che è attualmente la più diffusa: Arduino Uno Rev. 3. Con questa e un Personal Computer potrete realizzare progetti in base alla vostra fantasia, ma anche cercare in Rete se c'è qualcuno che ha già fatto qualcosa di simile, personalizzarlo secondo le vostre esigenze, quindi, nel rispetto della filosofia open-source, ricambiare rendendo il vostro progetto di pubblico dominio.
Per creare programmi o modificarne di esistenti, ovvero per caricarli in Arduino, vi serve l'IDE (gli esempi di questo volume sono stati provati su IDE v 1.06) che è l'ambiente di sviluppo meglio descritto nel Capitolo 3. Siccome non vogliamo limitarci all'insegnamento teorico ma crediamo che la pratica sia la via più facile per apprendere, abbiamo approntato per voi una serie di applicazioni tipo che realizzeremo insieme utilizzando l'apposito kit ARDUKIT BOOK, contenente una scheda Arduino, una breadboard e tutti i componenti occorrenti.

Arduino Uno

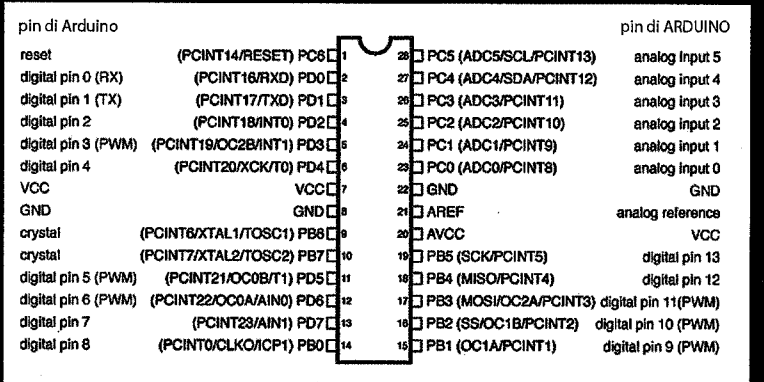
Nata nel 2013, è basata su un microcontrollore della famiglia AVR di Atmel e rappresenta la soluzione più economica e flessibile per realizzare applicazioni elettroniche interattive e tecnologiche. Arduino Uno è basata sul microcontrollore ATmega328P (in formato DIP) e dispone di 14 pin di I/O (di cui 6 utilizzabili come uscite PWM), 6 ingressi analogici, un oscillatore a 16 MHz, un connettore per la programmazione in-circuit ed un plug per l'alimentazione. Come nelle ultime versioni di Arduino, è presente il connettore USB per la connessione al Personal Computer. In Arduino Uno Rev. 3 l'interfaccia USB non ha più il chip FTDI, ma a realizzare la connessione USB è designato un microcontrollore Atmel programmato, tramite un connettore ICSP dedicato posto accanto ad AREF, per fare da transceiver USB/seriale.

Arduino Uno non richiede il programmatore -come gli altri microcontrollori- in quanto una volta scritto il programma e compilato il codice su Personal Computer, lo si carica nella memoria Flash del microcontrollore tramite il collegamento USB: tutto merito del bootloader, che permette al micro di "arrangiarsi da solo" e "sbrigare le pratiche" con il PC. Se la vostra applicazione non richiede grande corrente, Arduino Uno può essere alimentata direttamente dal computer, sempre tramite USB; altrimenti bisogna alimentarla tramite il connettore plug con un alimentatore esterno che fornisca da 7 a 12V, tensione che viene poi ridotta a 5V e a 3,3V dai regolatori presenti nella scheda. I 5 volt vengono resi disponibili sui connettori di espansione insieme alla tensione d'ingresso (la corrente prelevabile non deve superare 500 milliampere) per alimentare gli shield.

Arduino Uno si limita a fare calcoli e dispone di un minimo di periferiche per generare ed acquisire segnali analogici; dispone inoltre di alcuni bus di comunicazione; per il resto deve essere collegata ad elettronica esterna. Possiamo paragonare la scheda a un Personal Computer, rispetto al quale può interfacciarsi in ingresso e in uscita con segnali elettrici non solo digitali, ma anche analogici, grazie ai suoi connettori. Ciò significa poter gestire con Arduino praticamente qualsiasi cosa: se funziona con l'elettricità, collegandolo direttamente o mediante interfacce, mentre se è meccanico, gestirlo pilotando dispositivi elettromeccanici o acquisendone il movimento tramite sensori elettrici ed elettronici.

Dove non arriva direttamente, Arduino Uno provvede con l'aiuto degli shield, che gli permettono di rapportarsi ad esempio con utilizzatori di potenza (lampade, motori e tutto quello che ha bisogno di più di 5V e poche decine di milliampere) o anche con elettroniche complesse, sistemi meccanici ecc.

Qui sotto vedete la corrispondondenza tra i pin del microcontrollore ATmega328P e le linee presenti sui connettori di espansione di Arduino Uno; alcune sono ripetute ad esempio sul connettore ICSP, ma per l'uso che farete della scheda vi interessa solo il loro significato sui connettori di espansione.



Input e Output, Analogico e Digitale

Il primo passo fatto dagli sviluppatori di Arduino Uno è stato trasportare i piedini dell'ATmega 328P su connetto-

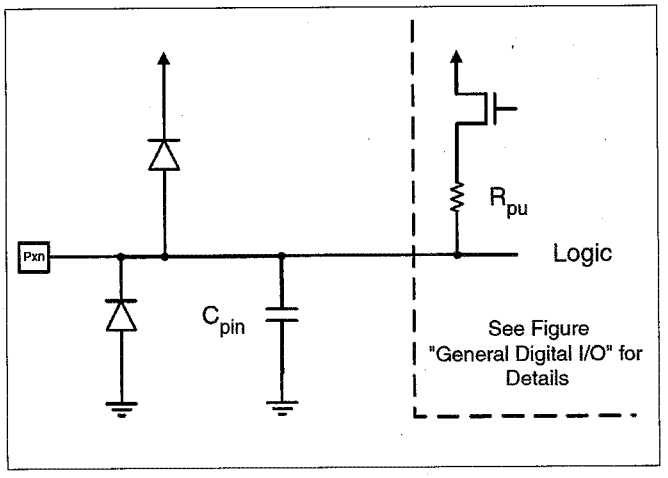


Fig. 2.1 - Le linee digitali di Arduino Uno possono funzionare sia da ingresso che da uscita: dipende da come vengono impostate nello sketch. Inoltre le uscite digitali possono assumere tre stati: alto, basso e alta impedenza (open). Qui ne vedete lo schema semplificato.

ri, ciascun contatto dei quali è stato chiamato con un nome che ne chiarisce la funzione.
Le tipologie di collegamento che vi offre Arduino Uno sono ingresso e uscita, per segnali digitali o analogici, ma anche connessioni dati su bus. Per comprenderle analizziamo le linee di ingresso/uscita (I/O) dividendole per categoria.

I/O digitali

I pin digitali di Arduino, ovvero dell'ATmega328, sono progettati per essere bidirezionali, ovvero in grado di funzionare da ingresso (Input) o uscita (Output); l'impostazione della modalità si deve effettuare all'avvio del microcontrollore mediante istruzioni che provvedono alla scrittura di opportuni valori nei registri di controllo (aree di memoria speciali interne al microcontrollore). Fisicamente, ogni I/O digitale è collegato a una serie di interruttori allo stato solido che dirottano il pin verso un registro d'ingresso o di uscita gestito dalla CPU dell'ATmega328P.
Come ingressi, i pin di Arduino Uno possono leggere due livelli logici: 0 -da zero a 2,5 volt- e 1, da 3 a 5 volt; tra 2,5 e 3 volt non è garantito con certezza il livello che Arduino Uno identifica. Gli ingressi hanno elevata impedenza (si parla di alcuni Mohm) per non caricare i dispositivi che li pilotano, che di norma sono in grado di erogare pochissima corrente (gate logici).
Quando sono in modalità Output, i pin possono assumere i livelli logici 0 e 1, corrispondenti rispettivamente a 0 e 5 volt; esiste anche un terzo stato detto Open (l'uscita va ad alta impedenza, come fosse interrotta) che si può impostare per fare come se il pin fosse sconnesso dal circuito. Per ottenere questa condizione è sufficiente definire il pin come ingresso e non leggerlo; questo stato risulta utile quando si vogliono collegare più dispositivi con un unico filo comune (o bus) per scambiare informazioni: chi vuole comunicare trasmette sul pin in modalità output i livelli logici e chi non deve far nulla o vuole ricevere i dati si pone in modalità input tri-state.
In uscita, i pin di Arduino Uno hanno dei circuiti di pilotaggio CMOS (Complementary MOS) che sono simmetrici, ossia dispongono di un MOSFET N ed uno P posti in cascata, con i gate in comune: l'uscita è prelevata dai drain.

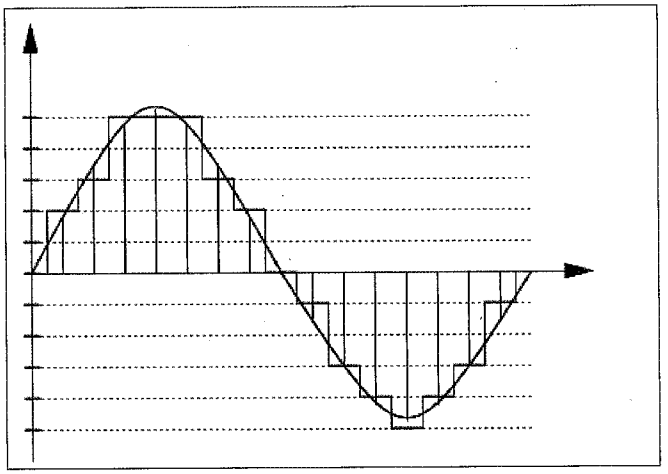
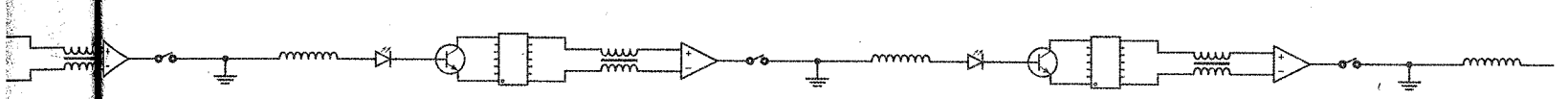


Fig. 2.2 - Il campionamento dell'ADC avviene prelevando alcuni valori momentanei di un segnale e convertendoli in numeri binari.

Questa soluzione consente di generare dei livelli 0 e 1 non solo con carichi ad alta impedenza (e quindi bassissimo assorbimento), ma anche su carichi che assorbono un massimo di 40 mA, fino a raggiungere un totale complessivo, per tutti i vari pin di 150 mA. L'avere due transistor consente di presentare la stessa impedenza e quindi erogare (quando l'uscita è a 1 logico) o assorbire (stato logico 0) la stessa corrente, coerentemente con il comportamento che la parola simmetria fa presagire.
I driver di uscita permettono ad Arduino Uno di collegare direttamente LED, cicalini, servocomandi, o anche sensori, per accenderli e spegnerli al bisogno, riducendo il consumo energetico.



Ingressi Analogici

Arduino Uno può leggere delle tensioni analogiche mediante gli appositi ingressi siglati A0, A1, A2, A3, A4, A5; siccome la CPU lavora con segnali logici, occorre una conversione, che nell'ATmega328P viene operata da un dispositivo chiamato convertitore analogico/digitale (A/D converter o ADC). Si tratta di un circuito che periodicamente campiona (mediante un dispositivo chiamato S&H - Sample & Hold) il segnale, ovvero lo fa passare a periodi, e mediante una serie di comparatori ne trasforma il livello nella commutazione di stati logici, che poi formano un certo numero di bit. Quindi l'A/D converter effettua misurazioni istantanee che converte in un valore binario.

Del convertitore A/D ci interessano due parametri: la risoluzione e la frequenza di campionamento; la prima rappresenta il numero di livelli in cui viene diviso l'intervallo di tensioni misurabile ed è espressa in bit. Se il convertitore è a 8 bit, potrà dividere la tensione in 256 valori; se è a 16 bit, in 65.536. Ne deriva che maggiore è la risoluzione, migliore sarà la capacità dell'ADC di distinguere con precisione i valori di tensione assunti dall'ingresso analogico. Arduino ha un convertitore A/D a 10 bit, quindi può distinguere 1.024 gradini di tensione.

Notate che la risoluzione non va confusa con la finestra di tensione campionabile, tuttavia più cresce la differenza tra massima e minima tensione misurabile, minore è la precisione dell'ADC a parità di bit di risoluzione. La scala di rappresentazione binaria della lettura dell'A/D è lineare, quindi nel caso di Arduino Uno, che ha una risoluzione di 1.024 campioni, ammettendo di fornire al convertitore una tensione di riferimento AREF di 5 V, il valore binario 512 corrisponde a 2,5V, mentre 128 corrisponde a 0,625 volt e così via. Un singolo livello è pari a 0,0048 V: questa è la risoluzione teorica dell'ADC della Arduino Uno.

Quanto alla frequenza di campionamento, rappresenta la frequenza con cui la lettura può essere ripetuta nel tempo: più è alta, maggiore è la frequenza dei segnali che possiamo misurare. Diciamo subito Arduino Uno può arrivare al massimo a registrare della voce con una fedeltà limitata, mentre per la musica è necessario uno shield dedicato. La frequenza massima di lettura senza usare "trucchi" è di circa 9 kHz e questo permette di campionare frequenze fino a 4,5 kHz.

Il convertitore A/D di Arduino è unico ma può essere assegnato (in rapida sequenza un pin alla volta) sulle sei linee analogiche A0÷A5.

L'uscita analogica e il PWM

Sebbene abbia delle uscite considerate analogiche, Arduino Uno in realtà non genera tensioni analogiche.

Per "analogica" si intende che la tensione può assumere qualsiasi valore compreso tra due margini, con continuità.

Le uscite analogiche di Arduino sono in realtà delle PWM (Pulse Width Modulation), perché producono impulsi periodici (ossia che si ripetono uguali nel tempo) a livello logico 1 o 0 (sono quindi digitali sotto l'aspetto del livello di tensione), però la loro frequenza e larghezza può variare entro due margini assumendo qualsiasi valore intermedio.

Per comprendere il discorso va introdotto il concetto di segnale elettrico ciclico (periodico): un segnale periodico varia ciclicamente e ripete il ciclo costantemente nel tempo con una certa frequenza; la frequenza è il numero

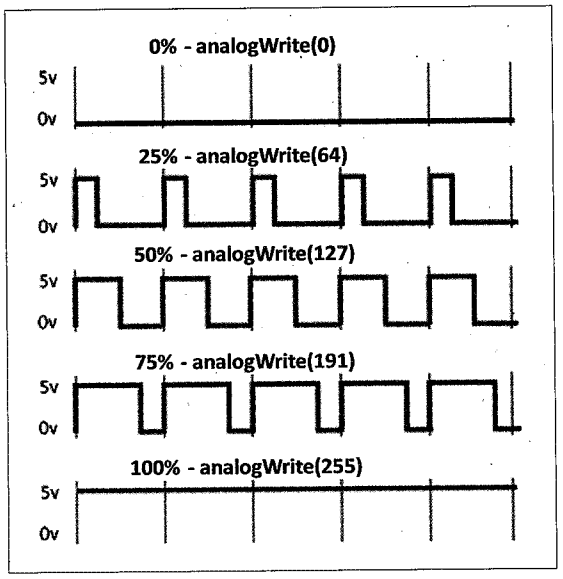


Fig. 2.3 - Il segnale PWM consente di ottenere una tensione di valore medio proporzionale al suo duty-cycle: si ottiene con l'istruzione analogWrite.

di volte che il ciclo si ripete in un secondo. Nel caso del segnale logico, il periodo o ciclo è costituito dall'alternarsi degli stati logici uno e zero; per segnali del genere si usa definire il duty-cycle (letteralmente, ciclo di lavoro) che corrisponde al rapporto tra la durata del livello alto e quella del periodo, ossia della durata del livello alto sommata a quella del livello basso (**Fig. 2.3**).
Il trucco che permette di ottenere da un segnale digitale uno analogico sta nel valore medio, ossia nella media nel tempo del valore di tensione prodotto dagli impulsi: se disegniamo su un foglio di carta i singoli impulsi e sommiamo le aree definite in un periodo, otteniamo il valore medio.
Ne deriva che con un PWM a duty-cycle del 25%, il valore medio è 1/4 dell'ampiezza degli impulsi; con il 50% è la metà. Se gli impulsi hanno ampiezza di 5 volt, nel primo caso il valore medio è 1 volt e nel secondo 2,5V.
Il modulo PWM di Arduino Uno è in grado di spaziare da 0 al 100% in 256 passi (il controllo del PWM viene realizzato dalla CPU dell'ATmega328 impostando valori ad 8 bit), quindi il 50% corrisponde al valore binario 128.
Le uscite PWM vanno bene per regolare la velocità di un motore elettrico in continua o la luminosità di una lampadina o di un LED (quest'ultimo possiamo pilotarlo direttamente da Arduino Uno); nel controllo di fattispecie, bisogna impostare una frequenza che superi i 24 Hz corrispondenti alla persistenza del nostro occhio, altrimenti vedremo il LED pulsare.
Dalle uscite PWM è possibile ricavare una tensione pressoché continua inserendo un filtro passa-basso, tipicamente RC (**Fig. 2.4**): il condensatore si carica all'ampiezza degli impulsi 5V e la resistenza rallenta la scarica quando l'uscita PWM assume lo zero logico. I valori di R (resistenza) e C (condensatore) vanno scelti con la formula: $f = 2\pi/RC$, dove f è la frequenza del PWM espressa in Hz ed R e C sono espressi rispettivamente in Mohm e microfarad. Se il circuito cui va fornita la tensione analogica assorbe una corrente tale da far cadere molta tensione sulla resistenza, si deve collegare l'uscita del filtro (Vi) a un buffer: per esempio un operazionale o un transistor montato a inseguitore di emettitore.

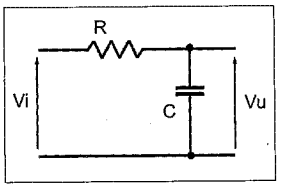


Fig. 2.4 - Ponendo all'uscita analogica un filtro RC passa-basso, si ottiene una tensione continua (Vu) di valore proporzionale al duty-cycle del PWM (Vi).

Significato dei contatti di Arduino Uno

Per sapere come collegare la scheda all'elettronica da gestire, bisogna conoscere la piedinatura e il significato dei contatti; vediamo dunque come sono piazzati fisicamente i pin. Per aiutare l'utente, il team di Arduino ha previsto che sullo stampato le linee siano raggruppate per funzione, quindi abbiamo tutti gli ingressi analogici vicini, gli I/O digitali a parte ecc. Nella Arduino Uno Rev.3, che è la scheda cui ci riferiamo in questo libro, i pin di collegamento con gli shield sono ripartiti in quattro blocchi: due a sinistra e due a destra (immaginando la scheda appoggiata sul lato opposto a quello del connettore USB); a sinistra troviamo due connettori femmina, uno da 6 e l'altro da 8 poli, mentre a destra ci sono un connettore femmina a 8 poli ed uno a 10.

Il lato digitale

Sul lato destro della scheda si trovano quattordici I/O (**Fig. 2.6**) definiti "Di-

gital" e affiancati in alcuni casi dal segno "~". Questi pin possono essere utilizzati indifferentemente come ingressi o uscite tramite le funzioni **pinMode()**, **digitalWrite()** e **digitalRead()**; i 14 contatti sono situati, i primi 8 sul connettore femmina a otto poli e i restanti 6 su quello a 10 poli; i quattro contatti rimanenti di quest'ultimo connettore corrispondono alla massa (GND, collegato insieme al GND della sezione POWER descritta più avanti nel capitolo) e AREF, che corrisponde alla tensione di riferimento da assegnare all'A/D converter della sezione analogica di cui parleremo più avanti in questo stesso capitolo. I due contatti più in alto sono, rispettivamente, SDA (contatto 17) e SCL (contatto 18) dell'I²C-Bus implementato dall'ATmega328; notate che a causa della vicinanza ad essi del connettore 2x3 poli a passo 2,54 mm dell'ICSP per il microcontrollore dell'interfaccia USB, nella scheda Arduino ufficiale i nomi di queste linee sono serigrafati dal lato saldature del circuito stampato (**Fig. 2.8**).

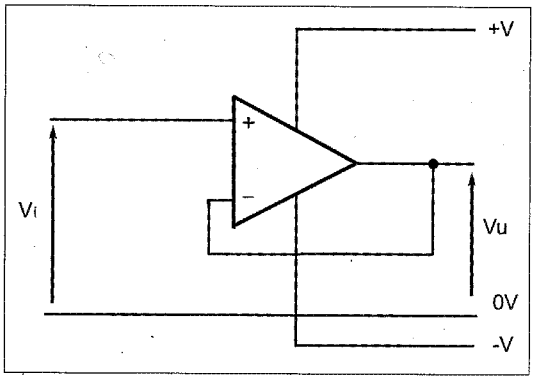


Fig. 2.5 - Buffer a operazionale da collegare all'uscita del filtro RC (Vu) applicato al pin PWM.

Ciascun pin digitale può erogare o assorbire al massimo 40 mA a seconda che sia a 0 o 1 logico.

Se impostato come input, ciascun pin può essere tenuto a livello alto tramite una resistenza di pull-up. Se non viene fornito qualche comando specifico in fase di inizializzazione (setup), tutti i pin digitali sono in input e le resistenze di Pull-up sono scollegate.

Alcuni pin digitali hanno più funzioni che si assegnano in alternativa, in base allo sketch caricato e al setup, descritte qui di seguito.

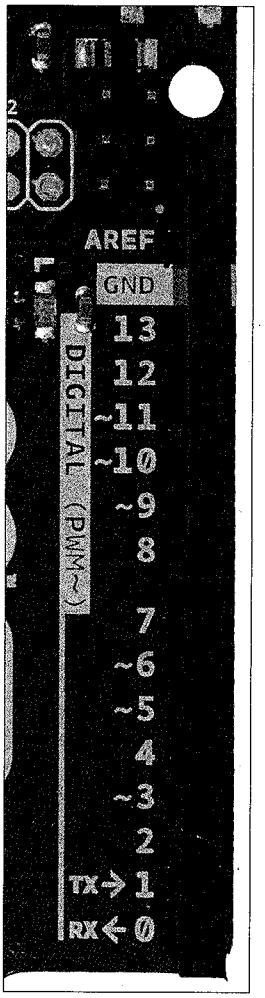


Fig. 2.6

UART (Porta Seriale): fa capo ai pin digital 0 (RX) e pin 1 (TX) e permette di dialogare in seriale -con livelli TTL- con altri dispositivi ricevendo sul pin 0 e trasmettendo sul pin 1. I due pin sono collegati ai corrispondenti dell'ATmega328P.

Interrupt Esterni: pin 2 e pin 3. Questi due pin permettono di attivare un interrupt (Capitolo 1) quando il loro livello subisce una variazione secondo una delle quattro modalità previste, ossia LOW (livello basso), CHANGE (tutte le volte che il pin digitale cambia stato), RISING (l'interrupt avviene quando il pin passa da 0 a 1) e FALLING (il pin passa da 1 a 0). La funzione con cui gestire l'interrupt è **attachInterrupt()**.

PWM (~): pin 3, 5, 6, 9, 10, e 11. Questi pin supportano la generazione di segnali PWM direttamente gestiti dall'hardware ATmega; la frequenza predefinita dell'onda quadra generata non è la stessa per tutti i pin: per 5 e 6 è 976 Hz, mentre per i restanti PWM è 488 Hz. Potrebbe essere modificata, ma per farlo occorrerebbe reimpostare i prescaler (divisori di frequenza) dei timer interni al microcontrollore, operazione che non è scopo di questo volume. La funzione per gestire il PWM su questi pin è **analogWrite()**.

SPI: pin 10 (SS), 11 (MOSI), 12 (MISO) e 13 (SCK). Questi pin supportano il protocollo di comunicazione seriale sincrono SPI (Serial Peripheral Interface) che viene utilizzato per comunicazioni su brevi distanze verso una periferica o un altro microcontrollore. La gestione dell'SPI è demandata a un'apposita libreria inclusa nell'IDE di Arduino.

LED: pin 13. Ripete la linea che fa accendere il LED L sulla scheda.

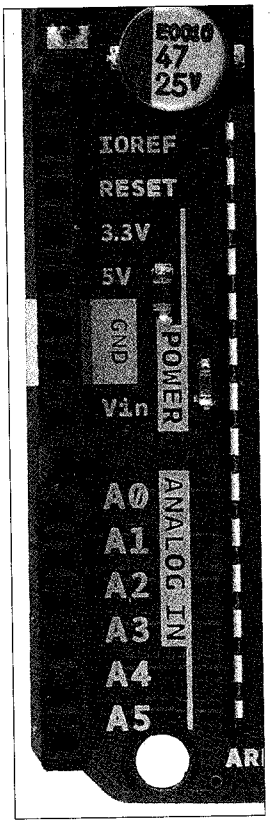


Fig. 2.7

Fig. 2.8 - I quattro contatti superiori del connettore a 10 vie portano le linee GND, IOREF e il bus I²C di Arduino. In evidenza, il connettore per la programmazione in-circuit del microcontrollore che fa da convertitore TTL/USB.

Le linee analogiche

Nella parte inferiore sinistra di Arduino Uno Rev.3 si trovano i sei pin degli ingressi analogici A0÷A5 (ANALOG IN) i quali sono predisposti per acquisire tensioni tra 0 e 5 volt; è possibile leggere ampiezze minori applicando la tensione limite al pin AREF che si trova in alto a destra, sopra il pin digitale 13, e la funzione **analogReference()** dove tra parentesi definite il valore (per esempio, INTERNAL, che è un riferimento interno di Arduino Uno a 1,1 volt; in questo caso non va applicato nulla ad AREF). Per avere la massima definizione possibile, impostate sempre un'AREF di poco superiore alla tensione da misurare e ricordate che la definizione è pari al rapporto tra AREF e 1.024. La tensione di riferimento su AREF non deve superare i 5V o essere più negativa di 0,7 V sotto il potenziale di GND.

Due degli input analogici possono essere assegnati alla funzione **TWI**: si tratta di A4 (SDA) e A5 (SCL), ripetuti come linee dedicate sopra il pin AREF (**Fig. 2.8**) e siglati SDA e SCL sul lato saldature della Arduino Uno Rev. 3; permettono la comunicazione dati con i dispositivi I²C e TWI in modalità a due fili. Il bus I²C è molto utilizzato per comunicare con vari dispositivi (memorie, display ecc.); la linea SDA (Serial DAta) trasporta i dati in modo bidirezionale, mentre SCL (Serial CLock, a volte chiamato SCK) è il clock che scandisce la comunicazione. Entrambe sono riferite a massa (GND). La comunicazione sul bus ha livelli 0/5 V.

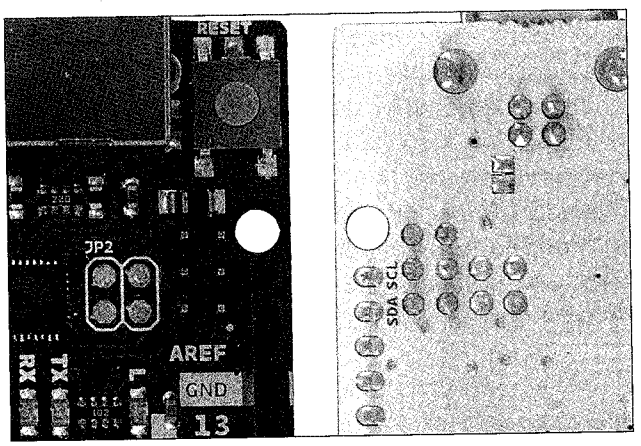
Per gestire il relativo protocollo, l'IDE di Arduino Uno mette a disposizione la libreria standard chiamata Wire.

Sempre a sinistra, sopra il connettore degli ingressi analogici si trova la serie di collegamenti riguardanti l'alimentazione (**Fig. 2.7**), il reset e il riferimento per gli I/O (IOREF). Sono descritti qui di seguito.

VIN: è collegato tramite un diodo al plug per l'alimentazione esterna di Arduino Uno, quindi rende disponibile agli shield l'alimentazione esterna.

5V: vi arriva l'alimentazione regolata e stabilizzata, usata per il microcontrollore e la circuiteria presente sulla scheda, se Arduino Uno è alimentata dal plug; se è alimentata dalla presa USB, giunge da lì.

3V3: da questo contatto può essere prelevata la tensione continua stabilizzata a 3,3 volt generata dal regolatore della scheda Arduino Uno. La corrente massima disponibile è 50 mA.



GND: è la massa di alimentazione e riferimento della scheda (0 volt).

RESET: è l'ingresso da mettere a massa per generare il segnale di reset in alternativa a quello generato con il pulsante (**Fig 2.8**); è utile per resettare Arduino dagli shield o ripetere su uno shield il tasto quando la scheda viene coperta da esso.

IOREF: è un pin comparso con la Rev. 3 di Arduino Uno e fornisce il riferimento di tensione utile per gli stadi di uscita degli shield, affinché possano generare tensioni compatibili con Arduino Uno Rev. 3. Sopra

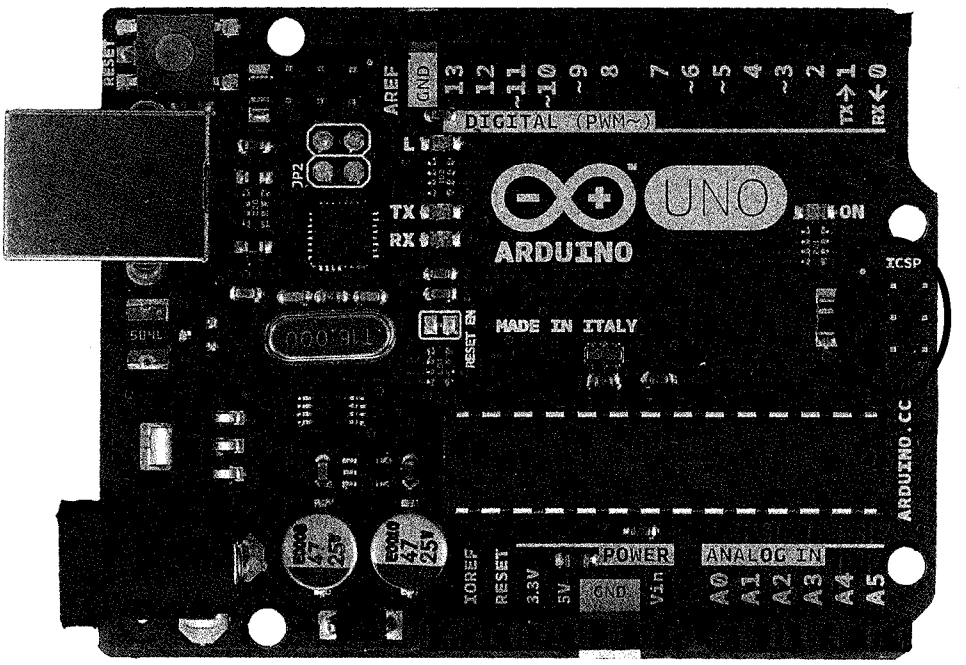


Fig. 2.9
Il connettore ICSP per la programmazione dell'ATmega328 di Arduino Uno Rev. 3.

IOREF si trova un ulteriore contatto che al momento non è assegnato ad alcuna funzione, ma che il team ha previsto di tenere a disposizione per futuri sviluppi della scheda.

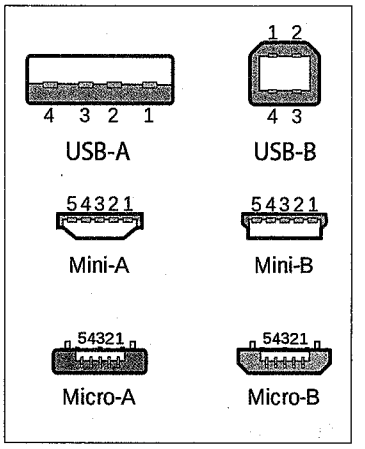
Il connettore ICSP

ICSP (In-Circuit Serial Programming) è un connettore da sei pin (2 file da tre a passo 2,54 mm) situato in centro nella parte bassa della scheda ed è utilizzabile per programmare direttamente –attraverso l'apposito programmatore– il microcontrollore ATmega 328. Questo genere di operazione non è tra gli scopi di questo libro e viene eseguita in fabbrica per caricare il bootloader. Esiste un secondo connettore ICSP, del quale abbiamo accennato qualche paragrafo indietro, che riguarda il micro utilizzato per l'USB.
Il connettore ICSP dell'ATmega328 ci interessa perché in alcune applicazioni le linee che gli fanno capo, avendo più di una funzione, possono essere inizializzate nel firmware per implementare ad esempio la comunicazione con alcuni shield (ad esempio quelli ethernet).

USB

Da anni la porta di comunicazione standard dei Personal Computer e accessori è l'USB (Universal Serial Bus), che oggi è giunta alla sua terza versione (retro compatibile con la versione 2.0 e 1.0). Si tratta di un bus a due fili (D+ e D-) cui possono essere applicati in parallelo 127 dispositivi; la porta contiene anche l'alimentazione a 5 volt (+5V e GND) che permette al computer (host) di alimentare varie periferiche, purché non assorbano nel complesso più di 500 mA, quindi anche Arduino Uno.
Nel collegare la vostra scheda a un PC, accertatevi di avere il cavo giusto, perché nel tempo, pur restando uguale la connessione elettrica (a parte nell'USB 3.0, dove il link D+/D- è sdoppiato) invariata, sono nate varie forme fisiche del connettore che riportiamo in **Fig. 2.10**. Il cavetto con cui collegherete la vostra Arduino Uno dovrà avere da un lato (quello di Arduino) l'attacco USB-B e dall'altro quello del vostro computer.

Fig. 2.10



Quanti pin! E ora cosa ci collego?!

Ad Arduino possiamo collegare tutto quello che ci serve per farlo interagire con il mondo esterno, quindi sensori per vedere (fotoresistori, fotocellule ecc.), sentire suoni (microfoni), rilevare odori (sensori di gas) o percepire la prossimità o il contatto con oggetti e persone (sensori capacitivi a trasferimento di carica, P.I.R. e sensori magnetici); possiamo anche applicare dispositivi per rilevare la posizione e lo spostamento (accelerometri e giroscopi) e per misurare le distanze (radar a ultrasuoni o infrarossi).
Oltre che di quelli per fargli percepire il mondo che lo circonda, possiamo dotare Arduino di dispositivi per agire sul mondo esterno: motori, servocomandi, luci, generatori di radiofrequenza e segnali audio o ultrasuoni, e quant'altro ci viene in mente.
Va ricordato che Arduino Uno (la scheda usata in questo libro), pur essendo molto flessibile, presenta dei limiti in termini di porte disponibili per gli ingressi e le uscite, nonché di proprietà delle stesse porte stesse. Per questo motivo, prima di scegliere il tipo di scheda per il proprio progetto conviene verificare quanti I/O ha disponibili e di che tipo sono, ovvero che funzioni possono svolgere (la tabella a pagina 7 fornisce utili indicazioni).

Un kit, tanti esperimenti

Per insegnarvi a utilizzare Arduino abbiamo predisposto un kit basato sulla scheda Arduino Uno e contenente tutti i componenti che abbiamo selezionato per compiere le esercitazioni didattiche descritte in questo libro.
Il kit si chiama ARDUKITBOOK, viene commercializzato dalla Futura Elettronica (www.futurashop.it) e contiene:

- una scheda Arduino Uno Rev. 3;
- un cavo USB;
- una Breadboard 400 contatti;
- una confezione da 20 jumper maschio-maschio (2 colori - 10 per tipo);
- una fotoresistenza;
- 5 LED rossi da 5 mm;
- 3 minipulsanti da C.S.;
- 3 MOSFET 50 V-0,5A BS170;
- 1 potenziometro 10 kohm lineare;
- 1 potenziometro slider 10 kohm lineare;
- 5 resistenze da 330 ohm 1/4 di watt;
- 5 resistenze da 180 ohm 1/4 di watt;
- un LED RGB da 5 mm;
- un cicalino piezo senza elettronica da circuito stampato;
- un connettore strip maschio 40 poli - passo 2,54 mm;
- un servocomando micro 9g -23x12,5x30 mm.

Oltre a questo, esiste un ulteriore kit "avanzato" (7300-ARDUKITV3) contenente altri componenti come ad esempio un display LCD 16x2 a interfaccia parallela, un ricevitore ad infrarossi a 38 kHz, un sensore di movimento e un motore elettrico in continua.