

La board ARDUINO UNO

Arduino è un sistema, a basso costo, basato su un microcontrollore della Atmel, realizzato su una piccola basetta.

Sono presenti sul mercato varie versioni del sistema con potenzialità diverse e con tipi differenti di processore (sempre dell'Atmel). Alcune versioni sono specializzate soprattutto per particolari funzioni.

Arduino Uno R3 utilizza il microcontrollore **ATmega328P** (figura 1.1).

I vari tipi di schede esistenti (tranne alcune eccezioni) rendono accessibili i pin del microprocessore su due file laterali di connettori (tipo femmina) a passo 2,54 mm.

Su questi connettori si possono inserire una o più schede periferiche (dette **shield**) ciascuna delle quali è in grado di svolgere particolari funzioni (per esempio shield per il controllo di motori DC o motori Passo Passo, shield per pilotare display LCD, shield Ethernet, shield WiFi, shield GPS, shield XBee, shield per comunicare via Bluetooth, shield per memorie microSD, shield per Data Logging, shield con Relè di potenza, shield MIDI e molte altre ancora).

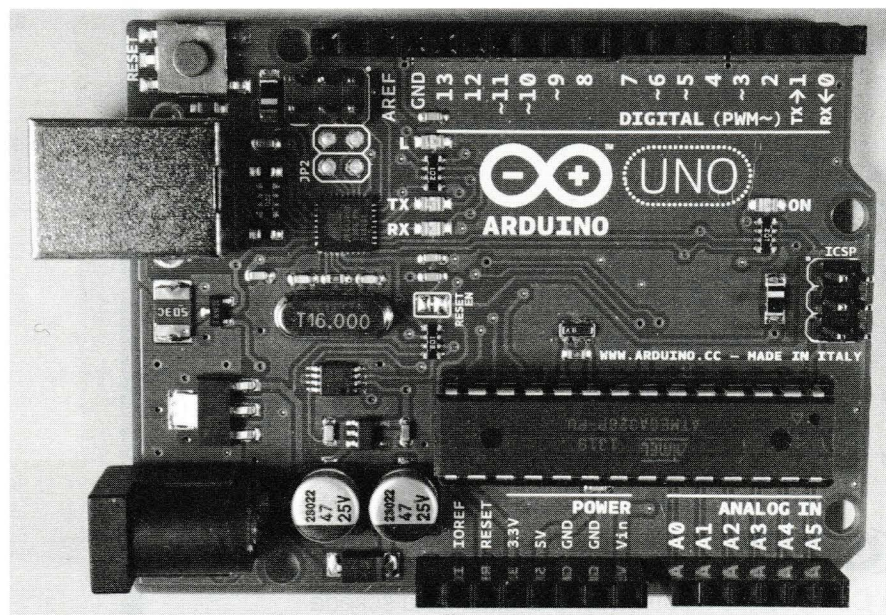


figura 1.1

Arduino viene connesso al PC su una porta USB che provvede anche ad alimentarlo. È possibile tuttavia, quando è richiesta dalle periferiche a esso connesse o dagli **shield**, una corrente superiore a quella che può essere fornita con l'USB, utilizzare un'alimentazione esterna.

La comunicazione Arduino-PC avviene con la USB emulando un collegamento seriale (RS-232).

La programmazione del dispositivo viene fatta sul computer utilizzando un editor appartenente all'**ambiente di sviluppo integrato (IDE)**, installato insieme con i driver di Arduino. L'applicazione dell'IDE (*Integrated Development Environment*), scritta in Java, è multiplatforma (per Windows, Linux e Mac OS X).



figura 1.

I programmi (detti **sketch**), che debbono essere eseguiti dal microcontrollore, sono scritti in C/C++ con l'aggiunta di estensioni che facilitano soprattutto l'utilizzazione delle funzioni di I/O. A corredo del software è fornita un'ampia collezione di **librerie** che possono essere utilizzate collegandole al programma da sviluppare.

1.1 Installazione dei driver della scheda

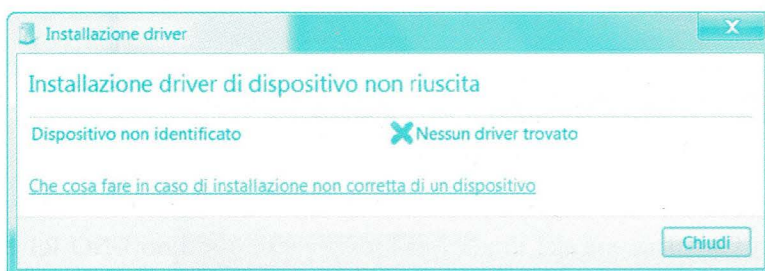


figura 1.2

Inserito il connettore sulla porta USB in Windows 7 e Vista i driver vengono installati automaticamente dal sistema operativo. Nel caso, invece, che esso non venga riconosciuto e compaia il messaggio INSTALLAZIONE DRIVER DISPOSITIVO NON RIUSCITA (figura 1.2), si proceda nel seguente modo:

1. dal menu START si faccia clic con il tasto destro su COMPUTER;
2. dal menu che si apre si scelga GESTIONE e dalla nuova finestra GESTIONE COMPUTER si scelga GESTIONE DISPOSITIVI;
3. sulla destra della finestra si selezioni ALTRI DISPOSITIVI e si faccia clic con il destro su DISPOSITIVO SCONOSCIUTO;
4. dal menu che si apre si scelga AGGIORNAMENTO SOFTWARE DRIVER e si si faccia clic con il tasto destro su di esso (figura 1.3);

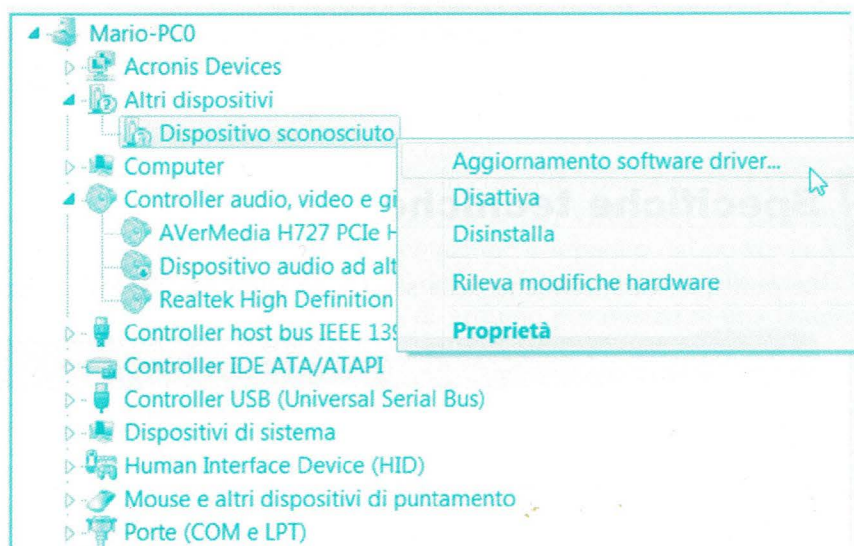


figura 1.3

5. si apre la finestra AGGIORNAMENTO SOFTWARE DRIVER - DISPOSITIVO SCONOSCIUTO. Si scelga CERCA IL SOFTWARE DEL DRIVER NEL COMPUTER e, quindi, nella nuova finestra cercare con SFOGLIA il percorso in cui si trova la cartella ARDUINO-1.0.3\DRIVERS (figura 1.4) e quindi premere il pulsante Avanti;
6. si avvia l'installazione; se è visualizzato l'avviso che non è possibile verificare l'autore del software, confermare l'installazione (INSTALLA IL SOFTWARE DEL DRIVER).

➔ Cerca il software del driver nel computer
Il software del driver verrà individuato e installato manualmente.

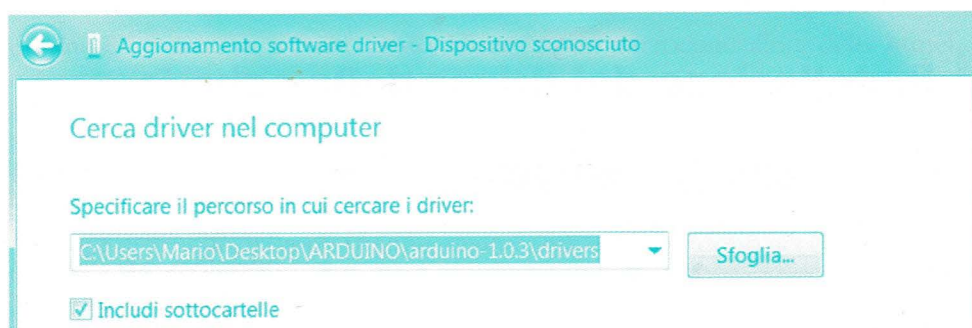


figura 1.4

Al termine viene confermata l'installazione del driver per il dispositivo Arduino UNO R3 è indicata la porta seriale COM 3 (o un'altra porta) con cui è collegato al computer (figura 1.5).

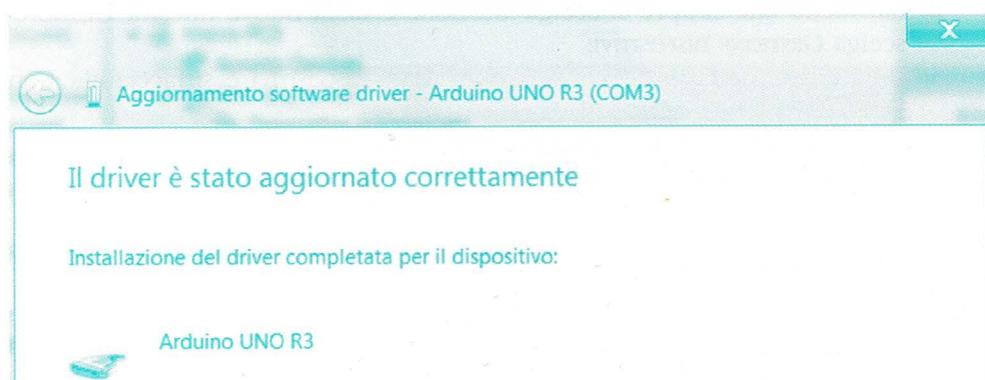


figura 1.5

1.2 Specifiche tecniche

Le principali caratteristiche della scheda **Arduino Uno R3** sono elencate nella **tabella 1.1**.

tabella 1.1

Specifiche Tecniche	
Microcontrollore on board	ATmega328P
Frequenza del clock del microcontrollore	16 MHz
Memoria RAM microcontrollore	2 KB
Memoria FLASH microcontrollore	32 KB [bootloader (0,5 kB) + programma]
Memoria EEPROM	1 KB
Tensione di alimentazione della scheda	7 ÷ 12 V DC
Uscita a + 5 V	I _{MAX} legata alla modalità di alimentazione
Uscita a + 3,3 V	I _{MAX} = 50 mA
6 ingressi analogici (A/D a 10 bit – f _{max} = 9 kHz)	A0 ÷ A5
14 I/O digitali	0 ÷ 13
6 degli I/O digitali possono essere con uscita e di tipo PWM	~3 – ~5 – ~6 – ~9 – ~10 – ~11
Corrente sui pin di I/O	40 mA (Absolute Maximum Ratings)

1.3 Alimentazione della scheda

Quando la board di Arduino Uno è connessa al PC tramite un cavo USB, viene attivata automaticamente l'alimentazione della scheda attraverso la porta USB del computer.

In questo caso la corrente massima disponibile per la scheda e per eventuali carichi esterni è quella fornita dal computer per mezzo della porta USB.

La scheda è poi dotata di un regolatore di tensione che opera quando si utilizzano altri metodi di alimentazione (figura 1.6).

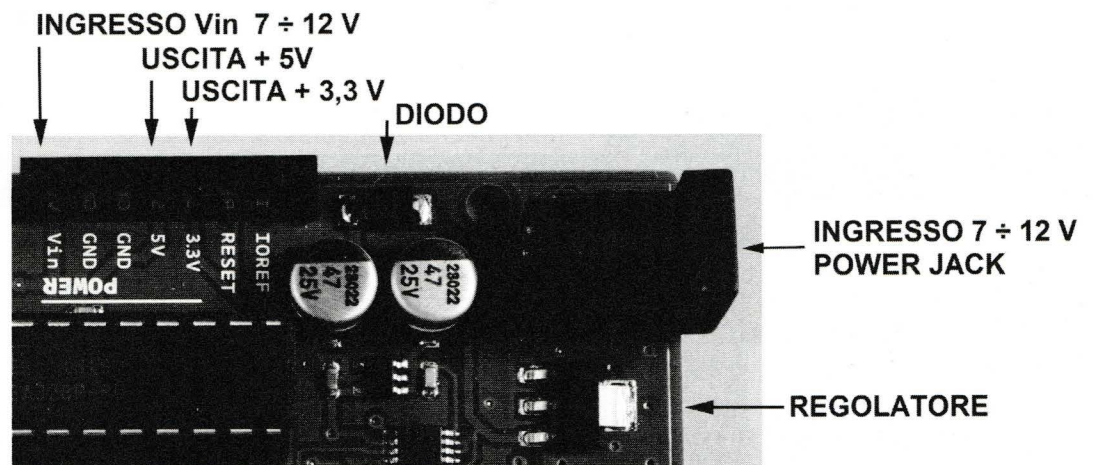


figura 1.6

Oltre l'alimentazione per mezzo della USB è possibile quindi alimentare la scheda:

- utilizzando una sorgente di alimentazione esterna in corrente continua regolata o non regolata con valore compreso tra 7 V e 12 V connessa al **power jack** della scheda con uno spinotto da 2,1 mm con **positivo al centro**;
- con una sorgente di alimentazione esterna in corrente continua regolata o non regolata con valore compreso tra 7 V e 12 V connessa al pin **VIN** posto sul connettore laterale della scheda. Questo ingresso per l'alimentazione è separato dal **power jack** da un diodo ed è collegato direttamente all'ingresso del regolatore presente sulla scheda. **L'ingresso può essere utilizzato per l'alimentazione di Arduino per mezzo di una batteria** (non c'è la caduta di tensione sul diodo rispetto all'ingresso **power jack**).

Tensioni di alimentazione superiore a 12 V sono ammissibili (**max 20 V**) ma non consigliabili in quanto fanno aumentare la dissipazione del regolatore. Una tensione di alimentazione inferiore a 7 V (**min 6 V**) rende instabile la tensione di alimentazione a 5 V della scheda.

1.4 I/O digitali

Sono presenti sulla scheda, accessibili su uno dei connettori laterali, i pin per l'ingresso e l'uscita di dati digitali (figura 1.7). Ogni linea è collegata con i pin di I/O digitale del microprocessore. I livelli d'ingresso e d'uscita delle linee sono riportati nella **tabella 1.2**. Alcuni pin possono avere funzioni diverse, selezionabili con particolari istruzioni.

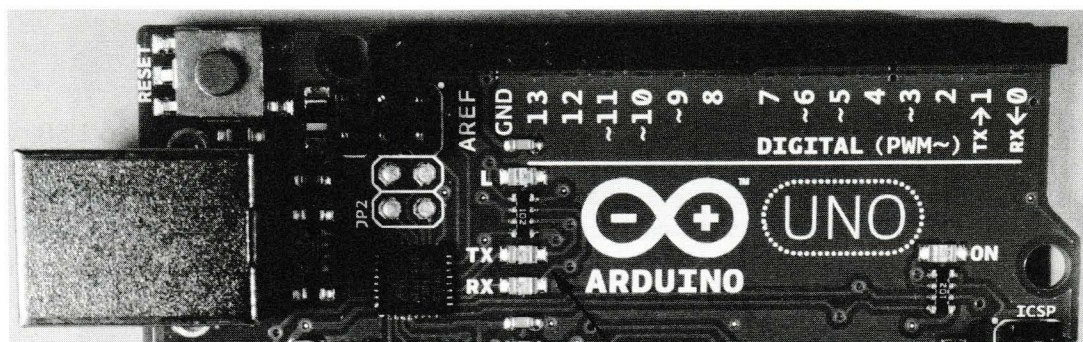


figura 1.7

CONNETTORE
USB TIPO B

LED TX e RX della
comunicazione seriale

tabella 1.2

	Pin	Livello H	Livello L
Out	0 ÷ 13	$\equiv V_{CC}$	$\equiv 0 V$
In	0 ÷ 13	$V_{IH} > 0,6 V_{CC}$	$V_{IL} < 0,3 V_{CC}$
Out PWM	~3 - ~5 ~6 - ~9 ~10 - ~11		

Valori più precisi per i livelli d'ingresso e d'uscita, dedotti dal datasheet dell'ATmega328P (doc8161.pdf), sono riportati nella **tabella 1.3**.

tabella 1.3

	Condizione di alimentazione	Min [V]	Max [V]
V_{IL}	$V_{CC} = 2,4 V \div 5,5 V$	-0,5	$0,3 V_{CC}$ (1)
V_{IH}	$V_{CC} = 2,4 V \div 5,5 V$	$0,6 V_{CC}$ (2)	$V_{CC} + 0,5$
V_{OL}	$V_{CC} = 5 V$ IOL = 20 mA		0,9
V_{OH}	$V_{CC} = 5 V$ IOH = -20 mA	4,2	

1. Si intende il valore più alto per il quale è garantito che il pin sia letto come livello basso.

2. Si intende il valore più basso per il quale è garantito che il pin sia letto come livello alto.

I **pin 0 (RX)** e **pin 1 (TX)**, com'è visibile nella **figura 1.7**, sono utilizzati anche per la **comunicazione seriale** (scheda Arduino ↔ PC o altro). Quindi, questi pin non possono essere adoperati come linee di I/O digitale se viene utilizzata la comunicazione seriale.

Le linee di uscita **PWM** forniscono un segnale digitale con **duty cycle** variabile con frequenza di default di 490 Hz ($T \approx 2$ ms). Sui pin ~5 e ~6 la frequenza di default è 976 Hz. La durata dell'impulso può essere variata da 0 (uscita sempre a livello basso) a tutto il periodo (uscita sempre a livello alto) con passi di 1/256. I segnali PWM possono essere utilizzati per variare la potenza fornita a un carico. Le istruzioni per utilizzare le **linee digitali** sono riportate nella **tabella 1.4**.

tabella 1.4

sintassi	parametri	
<code>pinMode(pin, mode)</code>	<code>pin</code> : numero del pin per il quale è impostata la modalità	<code>mode</code> : INPUT, OUTPUT, INPUT_PULLUP
<code>digitalWrite(pin, value)</code>	<code>pin</code> : numero del pin su cui deve essere scritto	<code>value</code> : HIGH o LOW
<code>digitalRead(pin)</code>	<code>pin</code> : numero del pin digitale che deve essere letto	ritorna un livello basso o alto (vedere tabella 1.3)
<code>analogWrite(pin, val)</code>	<code>pin</code> : numero del pin digitale PWM	<code>val</code> : valore variabile da 0 a 255

1.5 Input analogici

I sei pin per l'acquisizione di dati analogici sono presenti sulla seconda fila di connettori laterali accanto alla sezione power (**figura 1.8**). Con i canali A0 ÷ A5 è possibile leggere tensioni analogiche, con valori compresi tra 0 e +5 V, usando il convertitore A/D a 10 bit presente all'interno del microcontrollore. Si ha quindi una risoluzione pari a

$$1/2^{10} = 1/1024.$$

Con una $V_{REF} = 5\text{ V}$ la più piccola variazione di tensione che può essere letta è:

$$1\text{ LSB} = V_{REF} / 1024 \cong 4,88\text{ mV}.$$

Usando l'istruzione per la lettura di un canale si ha di ritorno un intero compreso tra 0 e 1023.

La tensione di riferimento di default può essere cambiata via software.

Le istruzioni per utilizzare le **linee analogiche d'ingresso** sono riportate nella **tabella 1.5**.

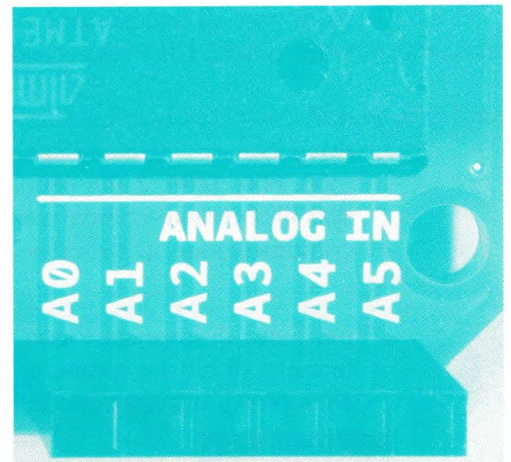


figura 1.8

tabella 1.5

sintassi	parametri
<code>analogRead(pin)</code>	<code>pin</code> : numero del pin analogico che deve essere letto (1). Ritorna un <code>int</code> (0 ÷ 1023)
<code>analogReference(type)</code>	<code>type</code> : DEFAULT : valore di default per la tensione di riferimento coincidente con la tensione di alimentazione + 5 V INTERNAL : valore interno, uguale a 1.1 V per ATmega328 EXTERNAL : la tensione di riferimento esterna (0 ÷ 5 V) deve essere applicata al pin AREF (2)

Note:

- Il tempo di conversione dell'A/D varia da 13 a 260 μs (datasheet dell'ATmega328P doc8161.pdf).
- Se si utilizza una tensione di riferimento esterna bisogna impostare il parametro `type` dell'istruzione `analogReference(type)` su EXTERNAL prima di utilizzare l'istruzione `analogRead()` per non causare possibili danni al microcontrollore.
- I pin analogici A0 ÷ A5 possono essere utilizzati come generici pin di I/O digitale (GPIO), usando le istruzioni `pinMode(AX, OUTPUT/INPUT)` e `digitalWrite(AX, HIGH/LOW)` per l'output oppure `digitalRead(AX)` per l'ingresso con X = 0 ÷ 5.

1.6 Risorse di comunicazione

Si descrivono brevemente i vari metodi di comunicazione implementati nella scheda Arduino che permettono di scambiare dati con periferiche e sistemi esterni.

► Comunicazione seriale

La trasmissione seriale asincrona è realizzata con la UART del microcontrollore ATmega328 con livelli TTL a 5V. Con il chip ATmega16U2-MU (8-bit *Microcontroller with USB Controller*), che affianca il microcontrollore principale, è realizzata una porta virtuale seriale che permette la trasmissione dei dati digitali sulla USB. Il firmware dell'ATmega16U2-MU utilizza i driver standard per la comunicazione USB COM, non richiedendo l'installazione di driver aggiuntivi. **I LED TX e RX lampeggiano quando c'è trasmissione di dati sulla USB verso o dal computer, ma non quando si effettuano trasmissioni seriali con i pin 0 (RX) e 1 (TX).** Le funzioni base di trasmissione seriale sui pin TX e RX sono riconosciute direttamente dal software di ARDUINO semplicemente usando l'inizializzazione della porta con l'istruzione:

```
Serial.begin(speed)
```

dove con **speed** è indicato il valore del baud rate da utilizzare per la comunicazione (300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, o 115200). Il baud specificato deve coincidere con quello utilizzato nel computer o con l'altro dispositivo di comunicazione. L'istruzione:

```
Serial.begin(speed, config)
```

permette di impostare con **config** anche gli altri parametri della comunicazione seriale: numero di bit del carattere, parità inclusa o esclusa, numero dei bit di stop.

Esempio di utilizzazione del parametro **config**:

```
Serial.begin(9600, SERIAL_8N1)
```

inizializzazione della porta seriale con un baud rate pari a 9600, carattere a 8 bit, nessuna parità, 1 bit di stop.

Nella **tabella 1.6** sono elencate alcune delle funzioni per la comunicazione seriale.

tabella 1.6

Serial.begin(speed)	Inizializza la porta seriale con un determinato baud rate.
Serial.begin(speed, config)	Inizializza la porta seriale con un determinato baud rate permettendo di assegnare anche altri parametri di trasmissione. Alcuni tipi di config : SERIAL_8N2: 8 bit, nessuna parità, 2 bit di stop SERIAL_8E1: 8 bit, parità pari (even), 1 bit di stop SERIAL_8O2: 8 bit, parità dispari (odd), 2 bit di stop SERIAL_7N2: 7 bit, nessuna parità, 2 bit di stop.
Serial.available()	Ritorna il numero di byte disponibili per essere letti. I dati in arrivo sono memorizzati in un buffer di ricezione che può contenere 64 byte.
Serial.write(val)	Scriva un valore binario sulla porta seriale. Ritorna il numero di byte scritti. Il dato val è inviato come byte o serie di byte.
Serial.print()	Scriva i dati sulla porta seriale come caratteri ASCII.
Serial.println(val)	val : valore da trasmettere. Ritorna il numero di byte trasmessi. Aggiunge a ogni dato trasmesso un CR LF (ritorno a capo, nuova linea).
Serial.println(val, format)	format : specifica la codifica del dato ASCII trasmesso. Per default se manca format la codifica è decimale. Codifica format : DEC (decimale), HEX (esadecimale), OCT (ottale), BIN (binario).
Serial.read()	Legge i dati che arrivano sulla porta seriale. Ritorna il byte letto o -1 se esso non è disponibile.

tabella

Per utilizzare funzioni più estese per la comunicazione seriale (per esempio uso di pin digitali TX e RX diversi dal pin 0 e pin 1) può essere usata la [libreria standard SoftwareSerial](#).

► SPI (Serial Peripheral Interface)

SPI è un sistema di comunicazione seriale sincrono full duplex, per brevi distanze, che permette lo scambio veloce di dati tra un microcontrollore e una o più periferiche o anche tra due microcontrollori. È un sistema di trasmissione basato su un master (in genere il microcontrollore) e uno o più slave (i dispositivi periferici) che utilizza per lo scambio dei dati quattro conduttori. Tre di essi sono comuni a tutti i dispositivi mentre il quarto è individuale per ogni periferica.

È supportato dalla [libreria standard SPI](#) che gestisce Arduino come master.

I segnali sono così denominati:

- **MISO** (*Master In Slave Out*) - Linea dello Slave per inviare dati al Master;
- **MOSI** (*Master Out Slave In*) - Linea del Master per inviare dati allo Slave (dispositivo periferico);
- **SCK** (*Serial Clock*) - Uscita del segnale di clock generato dal Master per sincronizzare la trasmissione dei dati;
- **SS** (*Slave Select*) - Pin presente su ogni Slave. Con una linea dedicata il Master seleziona un ben preciso dispositivo periferico.

Nella **tabella 1.7** sono elencati i pin in cui sono disponibili i segnali della SPI.

tabella 1.7

MOSI	MISO	SCK	SS (slave)
11 or ICSP-4	12 or ICSP-1	13 or ICSP-3	10

Il connettore ICSP (*In Circuit Serial Programming*), a 6 pin, è presente su un lato della board Arduino (**figura 1.9**).

Sul connettore sono presenti i tre segnali comuni della SPI (MISO, MOSI e SCK) oltre all'alimentazione V_{CC} e GND.

Il segnale SS per pilotare lo slave è il pin 10 che deve essere [configurato come uscita digitale](#).

Quando sono presenti più slave debbono essere usati altri pin digitali che debbono essere configurati come uscite.



figura 1.9

► I²C

È un sistema di comunicazione seriale a due fili (più la massa ed eventualmente la V_{CC}) per trasmettere e ricevere i dati. La comunicazione avviene tra un [master](#) (in genere un microcontrollore) e uno o più [slave](#). Il ruolo del master può essere svolto anche da più dispositivi, ma solamente uno per volta può essere quello attivo. Ogni dispositivo slave deve avere un indirizzo univoco (a 7 o a 10 bit). Il master conosce gli indirizzi dei dispositivi con cui deve collegarsi. Le due linee di comunicazione sono chiamate [serial clock](#) (SCL) e [serial data](#) (SDA). Sulle linee di comunicazione tra Master e Slave sono connessi due resistori di pull up che le mantengono normalmente alte. Sulla linea SDA transitano i dati (in maniera bidirezionale), ma anche gli indirizzi dei dispositivi slave. La [libreria standard Wire](#) permette di gestire la comunicazione seriale I²C permettendo ad Arduino di assumere il ruolo di Master o di Slave. È possibile pertanto far comunicare tra loro due (o più) board Arduino facendo assumere a una il ruolo di Master e all'altra (o alle altre) quello di Slave.

La programmazione di ARDUINO

1.1 L'IDE di Arduino

Dopo aver estratto dalla cartella compressa i file di Arduino e aver connesso la board con una porta USB del computer, si avvia l'applicazione facendo doppio clic su **arduino.exe** (figura 2.1)

Nome	Ultima modifica	Tipo	Dimensi...
drivers	26/12/2012 00:08	Cartella di file	
examples	26/12/2012 00:08	Cartella di file	
hardware	26/12/2012 00:08	Cartella di file	
java	26/12/2012 00:09	Cartella di file	
lib	26/12/2012 00:09	Cartella di file	
libraries	26/12/2012 00:09	Cartella di file	
reference	26/12/2012 00:09	Cartella di file	
tools	26/12/2012 00:09	Cartella di file	
arduino.exe	25/12/2012 12:44	Applicazione	840 KB
cygiconv-2.dll	25/12/2012 12:44	Estensione de...	947 KB
cygwin1.dll	25/12/2012 12:44	Estensione de...	1.829 KB
libusb0.dll	25/12/2012 12:44	Estensione de...	43 KB
revisions.txt	25/12/2012 12:44	Documento di...	36 KB
rxTxSerial.dll	25/12/2012 12:44	Estensione de...	76 KB

figura 2.1

Si entra nell'ambiente dell'IDE di Arduino mostrato in figura 2.2 su cui andranno scritti gli sketch usando le regole del linguaggio C/C++.

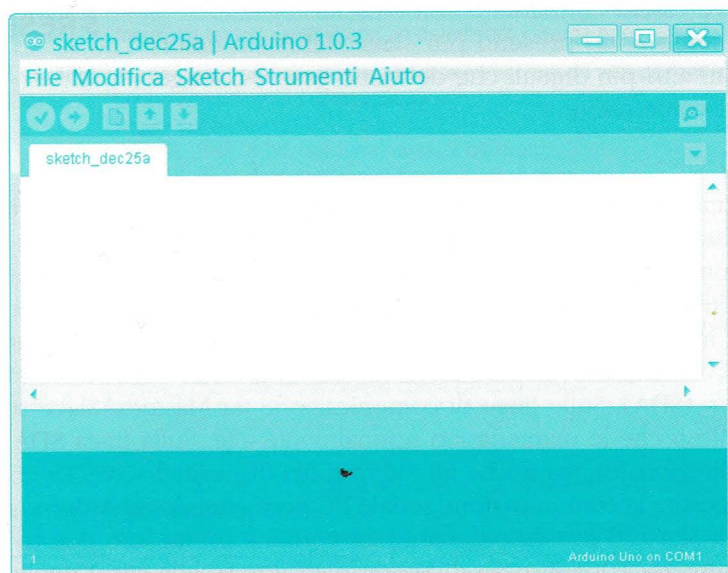


figura 2.2

È possibile ampliare la finestra degli sketch portando il cursore sui lati (o su un angolo) trascinandoli poi fino a ottenere la dimensione desiderata.

Sulla finestra si evidenziano le seguenti zone:

- nella parte alta c'è il **nome** attribuito in automatico allo sketch attuale (da scrivere);
- più sotto c'è una **barra dei menu** e, al di sotto di essa, una **barra dei pulsanti**;
- la parte bianca è utilizzata per la scrittura degli sketch;
- nella zona nera verranno scritte, in fase di caricamento dello sketch in memoria, le informazioni relative all'operazione compiuta o eventualmente gli errori presenti nell'applicazione editata;
- la porta COM, indicata in basso a destra, diviene quella effettiva, su cui è collegata la board, solo dopo che è stata selezionata con il menu STRUMENTI ⇒ PORTA SERIALE (figura 2.3a).

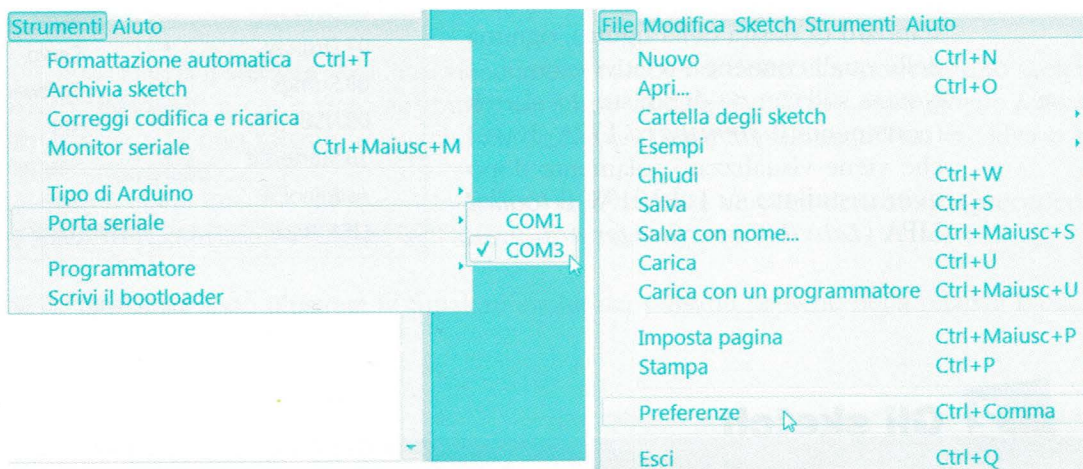


figura 2.3

a)

b)

Con il primo avvio di Arduino viene creata entro Documenti una cartella degli sketch chiamata Arduino.

È possibile utilizzare un'altra cartella modificandone il percorso aprendo il menu **FILE ⇒ PREFERENZE**. Si apre la finestra PREFERENZE di figura 2.4. Inserire il nuovo percorso in: **POSIZIONE DELLA CARTELLA DEGLI SKETCH** usando il pulsante sfoglia, per la ricerca.

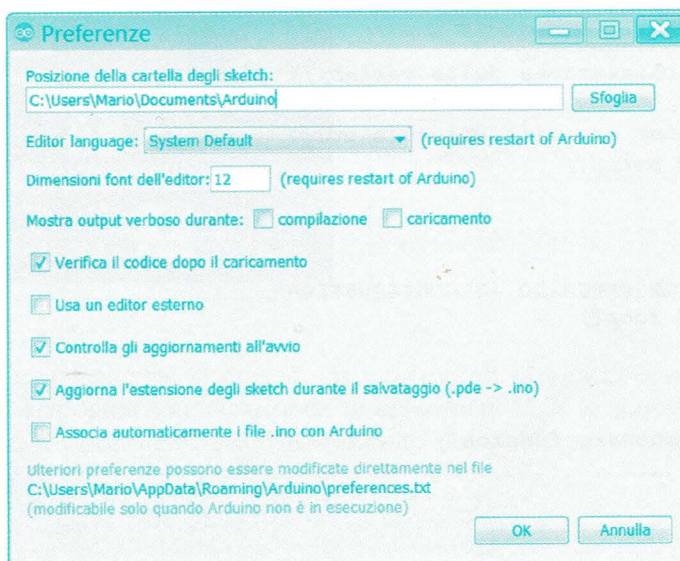


figura 2.4

esempio 2.1

Si vuole leggere la tensione sul pin di ingresso analogico A0 e visualizzarne il valore e il dato ottenuto dopo la conversione su un display LCD collegato ad Arduino.

La tensione presente su A0 è prelevata su un semplice circuito contenente una cella fotoconduttiva (o fotocella o anche fotoresistore → Modulo 1, Unità 5, paragrafo 5.2) in serie con un resistore. Più è alta la luminosità percepita dalla fotocella tanto più è alto il valore della tensione su A0.

Nella **figura 2.6** è riportato lo schema del circuito alimentato prelevando l'alimentazione direttamente dalla board Arduino.

Il punto di collegamento tra resistore e fotocella va collegato con l'ingresso analogico A0 (ingresso del convertitore A/D).

Il resistore R1 in serie con la fotocella ha il valore di 1 k Ω . Il valore della tensione presente su A0, prelevata ai capi del resistore R1 (in serie con la fotocella) è legato al livello di luminosità percepito dalla cella fotovoltaica.

Aumentando la luminosità diminuisce la resistenza R_{fc}, aumenta la corrente che scorre sui due elementi in serie e aumenta la tensione su A0.

Le caratteristiche del fotoresistore utilizzato (VT43N2) sono riportate nella **tabella 2.2**.

Dalla **tabella 2.2** si può osservare come la cella fotoconduttiva con una illuminazione di 10,764 lux ha una resistenza tipica di 16 k Ω . Al buio la resistenza aumenta raggiungendo il valore di 300 k Ω in 30 secondi.

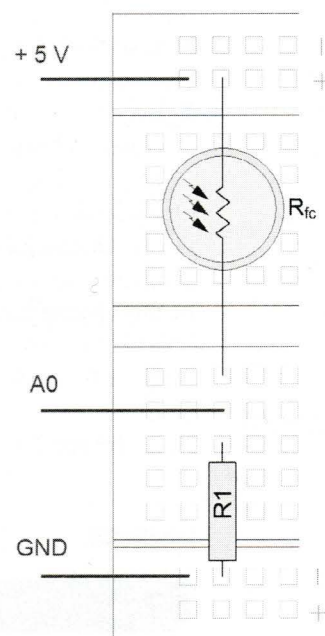


figura 2.6

tabella 2.2

Photoconductive Cell VT43N2				
1 fc a 6500 K			Dark	Time [s]
Min	Tip	Max		
8 k Ω	16 k Ω	24 k Ω	300 k Ω	30

Note

- fc (footcandle) = 10,764 lux
- 6500 K temperatura di colore

Come è possibile osservare dalla **tabella 2.2** i valori assunti dalla resistenza della fotocella, a parità di illuminazione, hanno una dispersione abbastanza rilevante tra un dispositivo e l'altro (min. 8 k Ω , max. 24 k Ω con circa 10 lux).

Per avere a disposizione altri valori di resistenza conviene effettuare delle misurazioni nell'ambiente nel quale sarà utilizzato il dispositivo.

Nella **tabella 2.3** sono riportati alcuni valori rilevati con un ohmetro e luxmetro (non essenziale se non si vogliono determinare anche i livelli di luminosità dell'ambiente).

tabella 2.3

Luminosità	Resistenza Cella fotoconduttiva	V_{A0}
8 lux	10,6 k Ω	0,43 V
11 lux	9,7 k Ω	
30 lux	1,4 k Ω	
280 lux	1,2 k Ω	
4300 lux	250 Ω	
8500 lux	160 Ω	4,3 V

Nota:

- le misure fatte sono solamente indicative in quanto la corrispondenza tra luminosità percepita dal luxmetro e quella captata dal fotoresistore possono essere diverse.
- I valori della tensione V_{A0} sono quelli teorici calcolati con le formule 2.1 e 2.2 facendo l'ipotesi che sia $V_{cc} = 5$ V.

Con i valori misurati può essere calcolata orientativamente la tensione presente ai capi del resistore da 1 k Ω che è poi quella convertita dal convertitore A/D, facendo l'ipotesi che la tensione di alimentazione del circuito sia uguale a 5 V.

Alla luminosità massima dell'ambiente (8500 lux) con 160 Ω si ha:

$$V_{A0} = \frac{V_{cc} \times R_1}{R_1 + R_{fc}} = \frac{5 \times 1000}{1000 + 160} \cong 4,3V \quad [2.1]$$

Mentre alla luminosità minima (8 lux) con 10,6 k Ω :

$$V_{A0} = \frac{V_{cc} \times R_1}{R_1 + R_{fc}} = \frac{5 \times 1000}{1000 + 10,6 \cdot 10^3} \cong 0,43V \quad [2.2]$$

I valori presenti sull'ingresso analogico della scheda Arduino sono trasformati dal convertitore A/D in valori numerici.

Il display LCD del tipo 16 caratteri con 2 righe (16 x 2) è dotato di 16 pin con le funzioni riportate in **tabella 2.4**.

tabella 2.4

pin	nome	funzione
1	V_{SS}	GND
2	V_{DD}	Alimentazione del modulo (+5 V)
3	V_0	Tensione per il contrasto
4	RS	H seleziona il registro dati - L seleziona il registro istruzioni
5	R/W	H modalità lettura - L modalità scrittura
6	E	Segnale di abilitazione per lettura o scrittura
7 ÷ 14	DB0 ÷ DB7	Bus dati bidirezionale
15	LED (A o K)	Alimentazione per retroilluminazione
16	LED (K o A)	Alimentazione per retroilluminazione

Note:

- La tensione V_0 è prelevata dal centro di un trimmer da 20 k Ω collegato tra + 5 V e GND.
- I pin per l'alimentazione del LED per la retroilluminazione (catodo e anodo) dipendono dal modello utilizzato.
- Nel display LCD dell'esempio è possibile cambiare la disposizione del catodo e anodo. Quella di default è A pin 16 (+ 5 V) - K pin 15 (GND).

La libreria standard **LiquidCrystal** controlla in modo trasparente per l'utente la scrittura dei dati sul display a cristalli liquidi. Può essere utilizzata con tutti i dispositivi che utilizzano il driver di controllo HD44780 Hitachi (o altro compatibile). Nell'esempio è stato usato il display **DEM 16226 SYH-LY** (con **controller SPL-C780D1 compatibile**).

La libreria gestisce il collegamento del display con Arduino realizzato solamente con **6 linee**: 4 del bus dati più i segnali RS ed Enable. Il pin R/W è collegato direttamente a massa.

Le connessioni sono realizzate in base a quanto riportato nella **tabella 2.5**. Sulla scheda Arduino possono essere utilizzati tutti i pin di I/O digitali liberi (la scelta è libera).

Per assegnare i pin da utilizzare deve essere usata la funzione di libreria:

```
nome_lcd.begin(RS, E, DB4 ,DB5 ,DB6 ,DB7)
```

Nell'istruzione ai nomi dei pin vanno sostituiti i numeri dei pin utilizzati come descritto nella **tabella 2.5**. L'istruzione quindi diviene:

```
nome_lcd.begin(7, 8, 3 ,4 ,5 ,6)
```

Funzioni della libreria **LiquidCrystal** utilizzate nell'applicazione

Nella **tabella 2.6** sono illustrate alcune funzioni della libreria.

tabella 2.6

LiquidCrystal nome_lcd (RS,E,DB4,DB5,DB6,DB7)	Assegna i pin utilizzati da Arduino: RS, E, DB4, DB5, DB6, DB7 vanno sostituiti con i corrispondenti pin utilizzati.
nome_lcd.begin(col,rig);	Imposta le colonne e righe del display usato.
nome_lcd.Clear()	Pulisce lo schermo del LCD e sposta il cursore nell'angolo in alto a sinistra.
nome_lcd.setCursor(col,rig);	Sposta il cursore nella posizione indicata.
nome_lcd.print(dato);	Invia al display il dato da visualizzare; se dato è una stringa va racchiusa entro doppi apici. Essendo utilizzate solo 4 linee del bus dati, ogni byte da inviare viene suddiviso in due semi-byte.
nome_lcd.print(dato,base);	Con base (DEC, BIN, HEX, OCT) si indica il formato con cui è visualizzato dato .

Nota:

l'istruzione **LiquidCrystal nome_lcd ()** può essere usata anche con 8 linee del bus dati. In questo caso ogni byte è inviato in una sola volta. La sintassi dell'istruzione è:

```
LiquidCrystal nome_lcd (RS, E, DB0, DB1,DB2, DB3, DB4,DB5, DB6, DB7).
```

Struttura dello sketch

Avviato Arduino ed entrati nell'ambiente dell'editore, si scrive lo sketch inserendo innanzitutto l'**include#** della libreria **LiquidCrystal** (menu SKETCH ⇒ IMPORTA LIBRERIA).

I passi per la scrittura dello sketch sono elencati di seguito:

- definizioni delle variabili; ➤
- assegnazione dei pin utilizzati in Arduino corrispondenti a quelli del display LCD;
- setup con indicazione del tipo di display LCD usato;

tabella 2.5

Display LCD		Arduino
N° Pin	Nome	Pin
4	RS	7
5	R/W	GND
6	E	8
11	DB4	3
12	DB5	4
13	DB6	5
14	DB7	6

- ▷ inizio loop;
- ▷ lettura ingresso analogico collegato alla foto-resistenza;
- ▷ attesa fine conversione (1 ms);
- ▷ cancellazione schermo LCD. Il cursore viene posizionato sull'angolo in alto a sinistra sulla prima riga;
- ▷ visualizzazione del dato a 10 bit convertito (N) variabile tra 0 e 1023 sulla prima riga del display;
- ▷ con la formula di conversione (→ [Modulo 4, Unità 1, paragrafo 1.2.2](#)).

$$V_{A0} = \frac{V_{REF} \times N}{2^{10}} = \frac{5 \times N}{1024} \quad [2.3]$$

Si calcola il valore della tensione sull'ingresso A0.

- ▷ Il cursore sul display viene posizionato all'inizio della seconda riga.
- ▷ Viene visualizzato il valore calcolato della tensione (V_{A0}).
- ▷ Nuovo ciclo.

Si fa l'ipotesi che la tensione di riferimento sia uguale a 5 V.










FotoRes_LCD1

```
#include <LiquidCrystal.h>
/*visualizza valore acquisito dal circuito Fotocellula (N) e tensione
su pin A0 sul display LCD*/
int luce_ambiente;           //definizione variabile luce ambiente
int pin_luce_ambiente = 0;    //pin ingresso analogico A0
float N;
float VA0;

LiquidCrystal lcd(7,8,3,4,5,6); //assegna pin usati in Arduino

void setup()
{
    lcd.begin(16,2);           //definisce il tipo di LCD (col, rig)
}

void loop()
{
    luce_ambiente=analogRead(pin_luce_ambiente); //Avvia conversione su A0
    delay(1);                     //attesa fine conversione
    lcd.clear();                  //pulisce schermo display LCD
    lcd.print("N=");             //visualizza N=
    lcd.print(luce_ambiente);     //visualizza su display valore letto su A0
    N=float(luce_ambiente);       //converte in float il dato intero (0-1023)
    VA0=(5*N)/1024;              //calcola valore della tensione su A0
    lcd.setCursor(0,1);          //sposta il cursore su seconda riga
    lcd.print("VA0=");           //visualizza VA0=
    lcd.print(VA0);              //visualizza valore calcolato
    delay(350);                  //attesa
}
```

Dopo la scrittura del codice si compila il sorgente con l'apposito pulsante      Verifica per controllare che non ci siano errori e, se non sono segnalati nella parte bassa dell'IDE, si carica nella memoria di Arduino     Carica.

Si ricordi che la scheda deve essere connessa al PC e che, non appena terminato il caricamento in memoria dello sketch, esso viene immediatamente posto in esecuzione.

Salvare lo sketch realizzato con il nome **FotoRes_LCD1**.

Nella **figura 2.7** è visualizzato il display LCD in fase di lettura connesso con la scheda Arduino. Sulla sinistra è visibile il fotoreistore con in serie il resistore da $1k\Omega$. Il display LCD è stato connesso alla breadboard con un connettore a pettine a 16 pin con terminali a 90° .

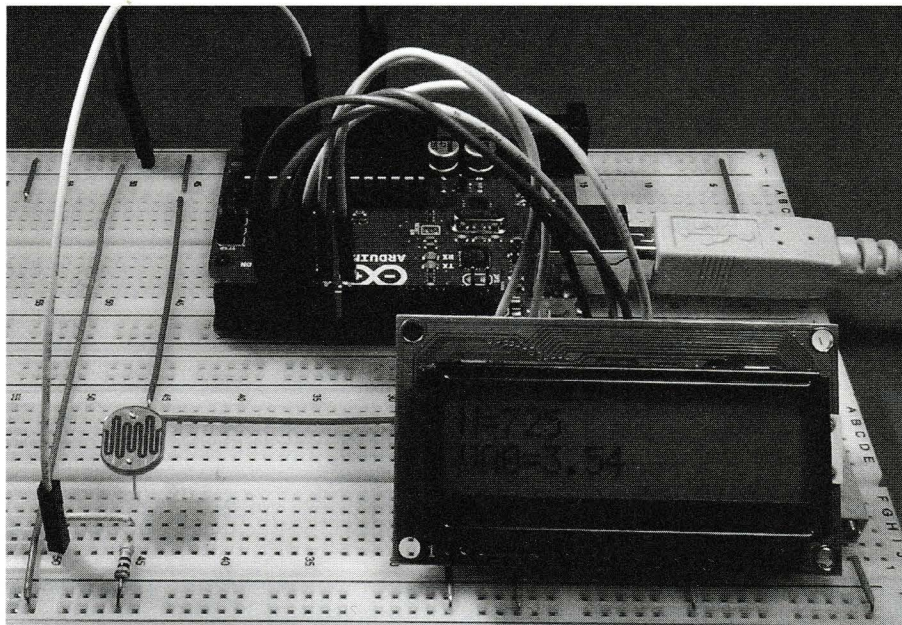


figura 2.7

Nell'**Esempio 2.2** si presenta la modalità di stesura di uno sketch contenente funzioni di input analogico, output digitale e output PWM.

esempio 2.2

Si vuole realizzare un sistema di controllo dell'illuminazione di un ambiente in base alla luminosità percepita da un fotoreistore posto al suo esterno. Si simula il sistema di illuminazione con uno o più LED bianchi. Si utilizza, per il rilievo della luminosità, il circuito presentato nell'Esempio 2.1 (cella fotoconduttiva in serie con resistore da $1k\Omega$). L'ambiente deve essere tanto più illuminato quanto più è bassa la luminosità percepita dal fotoreistore.

Per variare la luminosità dell'ambiente si utilizza un'uscita digitale PWM di Arduino che controlla la luminosità del LED bianco.

Se la luminosità esterna scende al di sotto di un certo livello, si deve accendere il LED bianco alla massima luminosità e segnalare inoltre l'evento con l'accensione di un LED rosso.

Realizzazione del circuito

La tensione sul pin di ingresso analogico A0 (V_{A0}), prelevata dal circuito del fotoreistore e poi convertita in un valore numerico, deve controllare la luminosità del LED bianco. Più è bassa la luminosità percepita dal fotoreistore tanto più deve essere luminoso il LED bianco, la luminosità del quale è controllata, aumentando o diminuendo la tensione ai suoi capi con un'uscita PWM.

Poiché l'accensione del LED bianco deve essere più grande quando il valore di V_{A0} è minimo e più piccola quando invece V_{A0} assume livelli più alti, il controllo del PWM deve avvenire con valori più piccoli quando la tensione V_{A0} è più alta e più alti quando invece essa ha un valore più basso.

Nella **tabella 2.7** è visualizzato il rapporto che intercorre nella variazione delle varie grandezze in gioco.

tabella 2.7

Variazione delle grandezze				
luminosità ambiente	V_{A0} [V]	valore fornito dal convertitore (N)	valore di controllo PWM	uscita PWM [V]
Minima \leftrightarrow Massima	$0 \leftrightarrow 5$	$0 \leftrightarrow 1023$	$255 \leftrightarrow 0$	$5 \leftrightarrow 0$

Note:

- il valore di $V_{A0} = 0$ V è teorico in quanto potrebbe ottenersi solamente con $R_{tc} = \infty$;
- il valore di $V_{A0} = 5$ V è teorico in quanto potrebbe ottenersi solamente con $R_{tc} = 0 \Omega$;
- per questi motivi, anche i valori massimi e minimi delle altre grandezze non sono in pratica raggiungibili;
- per le specifiche del progetto, al di sotto di un certo valore di luminosità ambiente, si assegna al **valore di controllo PWM** un codice pari a 255 ottenendo così per l'**uscita PWM** una tensione di 5V e la massima luminosità del LED bianco.

Il circuito dell'esempio precedente (fotocellula in serie a resistore $R1 = 1 \text{ k}\Omega$) va completato con:

- LED **bianco** con in serie il resistore **R2** per limitare la corrente;
- LED **rosso** con in serie il resistore **R3** per limitare la corrente.

I pin utilizzati nel sistema di controllo sono riportati nella **tabella 2.8**:

tabella 2.8

pin Arduino		funzione
Analogico A0	Ingresso analogico	Acquisisce la tensione ai capi di R1
Digitale ~9	Uscita PWM	Controlla tensione di alimentazione del LED bianco
Digitale 2	Uscita L/H	Controlla l'accensione del LED rosso

Lo schema del circuito è riportato in **figura 2.8**.

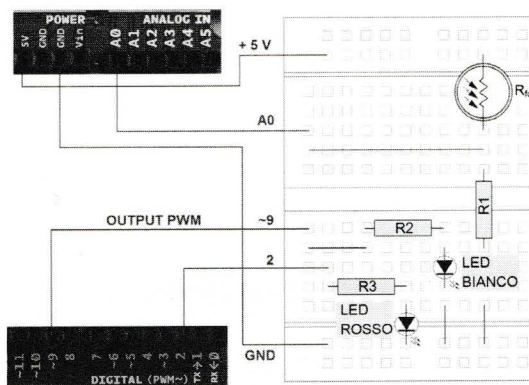


figura 2.8

I resistori R2 e R3 limitano la corrente che scorre nei LED e il loro valore è legato alle caratteristiche del tipo utilizzato.

Con un LED bianco da 5 mm con corrente $I_F = 15 \text{ mA}$ (0,015 A) e $V_F = 3,6 \text{ V}$, si ha

$$R2 = (5 - 3,6) / 0,015 = 93 \Omega \text{ (100 } \Omega \text{)}$$

con un LED rosso da 3 mm con corrente $I_F = 2 \text{ mA}$ (0,002 A) e $V_F = 1,8 \text{ V}$, si ha

$$R3 = (5 - 1,8) / 0,002 = 1600 \Omega \text{ (1,8 k}\Omega \text{)}.$$

Si tenga presente che la tensione di alimentazione del LED bianco è variabile e, in teoria, potrebbe assumere valori compresi da 0 V a circa 5 V. In realtà il suo valore è legato alla tensione presente su A0 prelevata ai capi del resistore R1 e quindi è legato al livello di luminosità percepito dalla cella fotovoltaica.

Struttura dello sketch

- Definizioni delle variabili.
- Assegnazione dei pin di I/O e dei pin utilizzati in Arduino corrispondenti a quelli del display LCD.
- Setup pin output e indicazione del tipo di display LCD usato.
- Lettura ingresso analogico collegato alla foto-resistenza.
- Attesa fine conversione (1 ms).
- Cancellazione schermo LCD. Il cursore viene posizionato sull'angolo in alto a sinistra sulla prima riga.
- Visualizzazione del dato a 10 bit convertito (N), variabile tra 0 e 1023, sulla prima riga del display.
- Cambiamento e inversione di scala dal valore letto ($0 \div 1023$), al nuovo valore $255 \div 0$. Il cambio di scala è necessario perché il valore acquisito con il convertitore A/D varia da valori al di sopra di 0 (minima luminosità) a valori inferiori a 1023 (massima luminosità), mentre il controllo del segnale PWM può andare da 0 (segnale in uscita a 0 V) a 255 (segnale d'uscita a 5 V). Osservando il circuito di **figura 2.6** si osserva che la tensione in ingresso al pin A0 del convertitore è prelevata su un resistore da 1 k Ω . Per quanto precedentemente detto, in base ai valori rilevati per la luminosità dell'ambiente, il valore massimo fornito dal convertitore è inferiore a 1023, non essendo la tensione d'ingresso uguale a 5 V e superiore al valore 0, con tensione d'ingresso superiore a 0 V.
- Il cursore sul display viene posizionato in avanti sulla prima riga.
- Viene visualizzato il codice di controllo del PWM.
- Il cursore sul display viene posizionato all'inizio della seconda riga.
- Calcola il valore della tensione sull'ingresso A0 (V_{A0}) con la formula di conversione (\rightarrow **Modulo 4, Unità 1, paragrafo 1.2.2**)

$$V_{A0} = \frac{V_{REF} \times N}{2^{10}} = \frac{5 \times N}{1024}$$

- Viene visualizzato il valore calcolato della tensione V_{A0}
- Controllo livello luminosità.
- Se **luminosità inferiore** a livello prefissato:
 - accendere LED rosso;
 - accendere LED bianco alla massima luminosità.
- Se **luminosità superiore** a livello prefissato:
 - spegnere LED rosso;
 - accendere LED bianco in base al valore rilevato della luminosità ambiente.
- Nuovo ciclo.

Sketch FotoResLED_LCD2

```

/*Fotocellula LED - visualizza N,VA0 e codice controllo PWM dopo lettura
luce ambiente. Dopo accensione LED rosso, LED bianco massima luminosità*/

#include <LiquidCrystal.h>
int luce_ambiente;           //definizione variabile luce ambiente
int lumin_LED_bianco;        //definizione variabile LED bianco
int pin_luce_ambiente = 0;    //pin ingresso analogico A0
int pin_LED_bianco = 9;       //pin uscita digitale PWM ~9
int pin_LED_rosso = 2;        //pin uscita digitale 2
LiquidCrystal lcd(7,8,3,4,5,6);
float N;
float VA0;
void setup()
{
  lcd.begin(16,2);
  pinMode(pin_LED_bianco,OUTPUT); //setta pin LED bianco come uscita
  pinMode(pin_LED_rosso,OUTPUT); //setta pin LED rosso come uscita
}

```

```

void loop()
{
  luce_ambiente = analogRead(pin_luce_ambiente);
  delay(1); //attesa fine conversione
  lcd.clear(); //pulisce schermo display LCD
  lcd.print("N="); //visualizza N=
  lcd.print(luce_ambiente); //visualizza su display valore letto su A0
  lumin_LED_bianco = map(luce_ambiente,0,1023,255,0); //converte val. acquisito
  lcd.setCursor(7,0); //sposta cursore in avanti su prima riga display
  lcd.print("n_PWM="); //visualizza n_PWM=
  lcd.print(lumin_LED_bianco); //visualizza valore di controllo del PWM
  N=float(luce_ambiente); //converte in float luce_ambiente (0-1023)
  VA0=(5*N)/1024; //calcola valore della tensione su A0
  lcd.setCursor(0,1); //sposta il cursore sul display
  lcd.print("VA0="); //visualizza VA0=
  lcd.print(VA0); //visualizza valore calcolato
  if(luce_ambiente < 200) //se livello di luminosità inferiore alla soglia
  {
    digitalWrite(pin_LED_rosso ,HIGH); //accende LED rosso
    analogWrite(pin_LED_bianco,255); //accende LED bianco massima luminosità
  }
  else
  {
    digitalWrite(pin_LED_rosso, LOW); //se sopra la soglia spegne LED rosso
    analogWrite(pin_LED_bianco,lumin_LED_bianco); //LED bianco luminosità var.
  }
  delay(350);
}

```

Si noti come la visualizzazione del codice da inviare al PWM è mantenuta sul display LCD (N PWM) anche quando si accende il LED rosso, ma esso non è utilizzato in quanto come codice di controllo è utilizzato il valore 255.

Dopo la scrittura del codice si compila il sorgente e, se non sono segnalati errori, si carica nella memoria di Arduino.

Si ricordi che la scheda Arduino deve essere connessa al PC e che, non appena terminato il caricamento in memoria dello sketch, esso viene immediatamente posto in esecuzione.

Salvare lo sketch realizzato con il nome **FotoResLED_LCD2**.

Nella **figura 2.9** è visualizzato il display LCD con un basso valore di **N** e alto valore di **n_PWM** (scarsa luminosità). Con questi valori è acceso il LED rosso e il codice di controllo per il PWM è 255 (e non 242).

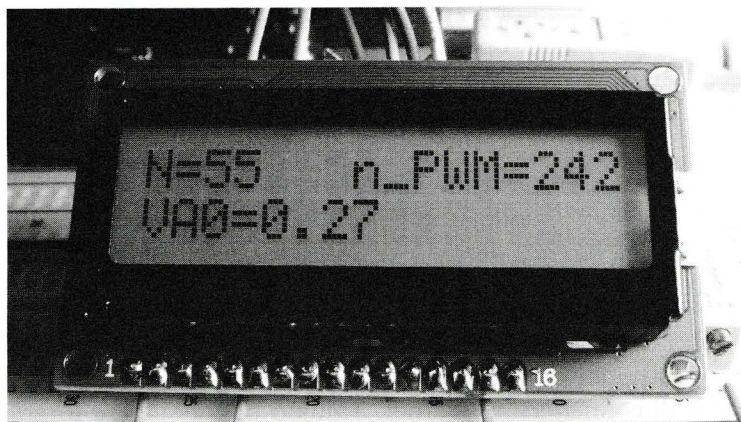


figura 2.9

2.3 Le librerie

A corredo del software di Arduino sono state inserite un buon numero di librerie (dette **librerie standard**) che possono essere collegate agli sketch con la direttiva `include#` (`<nome_libreria.h>`). Con il menu **SKETCH** ⇒ **IMPORTA LIBreria** (figura 2.10), scegliendo una libreria viene automaticamente inserito il relativo `include#` all'inizio dello sketch.

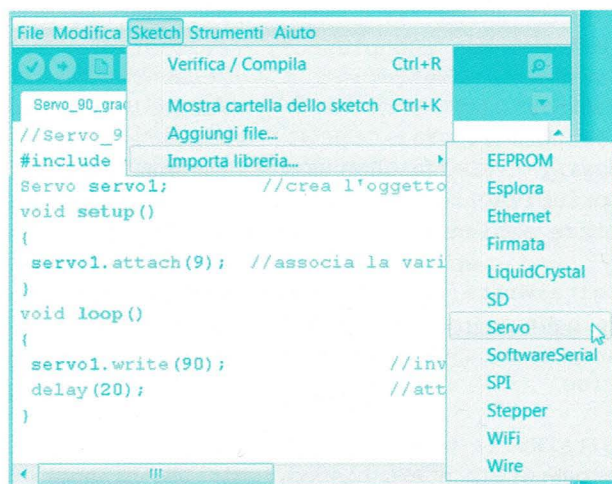


figura 2.10

► Librerie Standard

- **Servo** - usata per il controllo dei servomotori. Contiene una serie di funzioni che permettono di controllare la posizione dell'albero dei servo, utilizzando direttamente l'angolo di rotazione. Viene generato il segnale PWM, con la frequenza e il duty cycle necessari per posizionare l'asse d'uscita del servomotore (→ [Modulo 10, Unità 4](#)).
- **LiquidCrystal** - per controllare i display a cristalli liquidi. Può essere utilizzata con tutti i dispositivi che utilizzano il driver di controllo HD44780 Hitachi (o altro compatibile). Gestisce comunemente display con 2 righe da 16 caratteri. Per l'invio dei dati al display possono essere utilizzate otto o quattro linee digitali oltre quelle di controllo (due o tre).
- **TFT** - questa libreria permette alla scheda Arduino di comunicare con i TFT LCD per disegnare testo e immagini. Il display, per comunicare con Arduino, utilizza la libreria SPI che deve essere inclusa negli sketch.
- **SD** - per la lettura o scrittura su schede di memoria di tipo SD (*Secure Digital*). La libreria supporta anche schede di tipo SDHC (*Secure Digital High Capacity*) con più alta capacità di memorizzazione (superiore ai 2 GByte fino a 32 GByte). Le SD e SDHC sono divise in classi in base alla velocità di lettura e scrittura dei dati: Classe 2 (minimo 2 MByte/s), 4 (minimo 4 MByte/s), 6 (minimo 6 MByte/s) e 10 (minimo 10 MByte/s). La libreria supporta come *file system* di formattazione delle schede la FAT16 (SD) e la FAT32 (SDHC). La comunicazione tra il microcontrollore e la scheda è effettuata con il sistema di comunicazione seriale SPI (*Serial Peripheral Interface*). Per i file sono usati i nomi corti (al massimo 8 caratteri per il nome più 3 per l'estensione). Esistono in commercio anche schede SD con incluso un RTC (*Real Time Clock*) che permettono di realizzare dei compatti **data logging** (*shield Adafruit Assembled Data Logging*). Un lettore di schede SD è presente anche negli shield Ethernet.
- **Ethernet** - la libreria è di supporto agli shield Ethernet permettendo così ad Arduino di connettersi a Internet. Arduino può assumere il ruolo di Server o di Client.
- **WiFi** - è di supporto agli shield WiFi per connettere Arduino a Internet. Ha molte fun-

zioni simili alla libreria Ethernet. Perché si possa realizzare il collegamento, è necessario che la rete alla quale ci si deve connettere trasmetta esplicitamente il suo SSID (*Service Set Identifier*). La libreria accetta la crittografia WEP e WPA2 Personal (con metodo a chiave condivisa) ma non il WPA2 Enterprise (con server di autenticazione).

- **GSM** - di supporto agli shield GSM per la connessione con reti GSM/GRPS.
- **Stepper** - libreria per controllare i motori passo-passo.
- **EEPROM** - libreria di supporto per la lettura e scrittura della memoria di tipo EEPROM non volatile contenuta all'interno del microcontrollore sulla scheda Arduino. L'ATmega328P ha un 1 kByte di tale memoria. Per scrivere un byte sulla memoria occorrono 3,3 ms. La EEPROM ha un massimo di vita di 100000 cicli di scrittura/cancellazione. Si faccia quindi attenzione a non usare dei loop di scrittura/cancellazione sulla memoria, perché potrebbero facilmente esaurirla. Non inserire direttamente nel `void loop()` operazioni di scrittura/cancellazione su tale memoria.
- **Firmata** - è una libreria che implementa il protocollo omonimo che permette a un computer host di comunicare con Arduino sulla porta seriale (emulata con la USB). Nel computer host deve essere installata un'analoga libreria con il protocollo Firmata. Sul sito www.firmata.org sono presenti le librerie per vari sistemi operativi e per diverse applicazioni. Particolarmente interessante, per il collegamento e controllo di Arduino, è **Processing**, ovvero un linguaggio con un PDE (*Processing Development Environment*) che ha molti punti di contatto con l'IDE di Arduino (i programmi scritti con Processing sono, per esempio, detti sketch). È possibile controllare la scheda Arduino con Processing senza scrivere codice su Arduino stesso.
- **SoftwareSerial** - questa libreria, utilizzando la UART interna al microprocessore permette di effettuare la comunicazione seriale con tutti i pin digitali. È possibile aprire diverse porte seriali anche se, solo una alla volta, è in grado di effettuare la trasmissione dei dati.
- **SPI** - la libreria è di supporto al bus SPI (*Serial Peripheral Interface*) per gestire un sistema di comunicazione seriale sincrono full duplex in cui Arduino assume il ruolo di master.
- **Wire** - la libreria permette di gestire la comunicazione seriale I²C permettendo ad Arduino di assumere il ruolo di Master o di Slave.

Librerie non Standard

Sono reperibili sul WEB un gran numero di librerie, non direttamente caricabili dall'IDE di Arduino, prodotte da terzi. Molte di esse sono di supporto agli shield presenti sul mercato. È necessario che queste librerie, prima di essere utilizzate, vengano installate nell'IDE. A partire dalla **versione 1.0.5** è possibile eseguirne un'installazione automatica.

Una volta scaricata la libreria in formato .zip (senza scompattarla), usando il menu **SKETCH ⇒ IMPORTA LIBRERIA ⇒ ADD LIBRARY** (figura 2.11) nella finestra che si apre (SELECT A ZIP FILE OR A FOLDER CONTAINING THE LIBRARY YOU'D LIKE TO ADD), bisogna ricercare e selezionare la libreria premendo poi il pulsante APRI. La nuova libreria verrà automaticamente aggiunta in basso al menu **SKETCH ⇒ IMPORTA LIBRERIA**.

Seguendo le istruzioni riportate sul sito ufficiale di Arduino ([tutorial on writing your own libraries](#)) è anche possibile creare librerie personali.

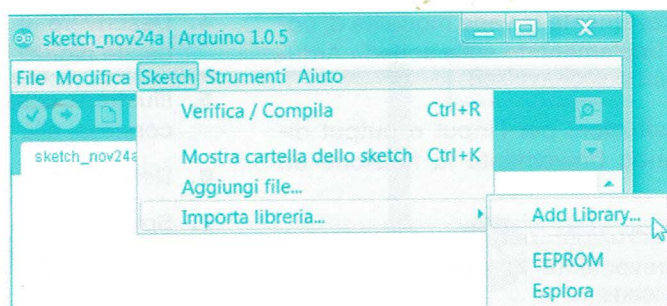


figura 2.11