

Arduino

L'utilizzo di un microcontrollore presenta talvolta alcune difficoltà per chi si avvicina per la prima volta alla progettazione.

I costruttori, per agevolare il lavoro ai progettisti, in genere mettono a disposizione sistemi di sviluppo costituiti da piattaforme hardware e ambienti software per la programmazione.

Nel 2005 un gruppo di progettisti dell'*Interaction Design Institute di Ivrea* ha messo a punto un sistema di sviluppo, chiamato **Arduino**, con l'obiettivo di offrire la massima economicità e semplicità di utilizzo, consentendone l'impiego anche a persone estranee al settore elettronico-informatico.

Questo sistema si è poi diffuso in tutto il mondo, soprattutto in ambito didattico e hobbistico.

Arduino mette a disposizione:

- una piattaforma hardware;
- un ambiente di sviluppo integrato.

La **piattaforma hardware** è costituita da schede che hanno come elemento centrale un *microcontrollore Atmel*.

La scheda è provvista anche di altri dispositivi che ne semplificano l'utilizzo e l'interfacciamento (ad esempio il quarzo per la generazione del clock, il regolatore di tensione, l'interfaccia per la comunicazione USB, connettori di ingresso e uscita per segnali analogici e digitali, LED).

La figura 1 riproduce la scheda **Arduino Uno** (alla quale si riferisce la descrizione) e la più moderna e potente **Arduino Due**.

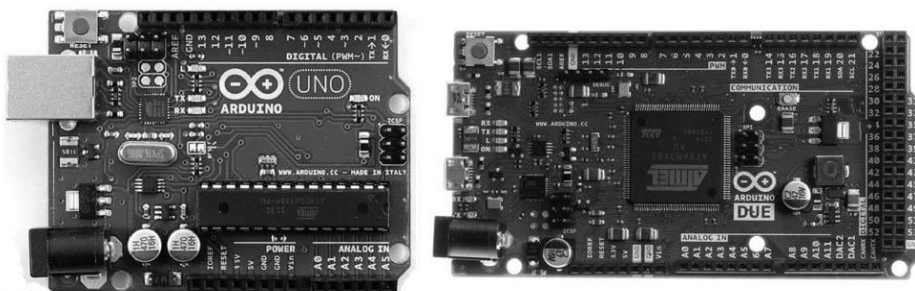


Figura 1 Schede Arduino Uno e Arduino Due

La tabella 1 confronta le caratteristiche della scheda Arduino Uno con la più recente Arduino Due.

Arduino	Microcontrollore	Flash (KB)	EEPROM (KB)	SRAM (KB)	Pin di I/o digitale	...di cui con PWM	Pin di input analogico
Uno	ATmega328P	32	1	2	14	6	6
Due	Atmel SAM3X8E	512		96	54	12	12

Tabella 1 Confronto tra le caratteristiche di Arduino Uno e Arduino Due

L'**ambiente di sviluppo integrato** si avvale del linguaggio *Wiring*, derivato dal C/C++.

Gli strumenti messi a disposizione vengono utilizzati per i seguenti scopi:

- scrivere il programma (denominato *sketch*) e controllarne la sintassi (*editor*);

- convertire il programma dal linguaggio sorgente al codice macchina (*compilatore*);
- caricare il codice nella memoria del microcontrollore (*loader*);
- controllare l'esecuzione del software;
- correggerne eventuali errori (*debugging*).

L'interfaccia grafica del programma si presenta come in figura 2.

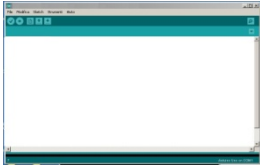


Figura 2 Interfaccia grafica del programma

Pregi del sistema Arduino

I pregi principali della piattaforma Arduino sono:

- l'economicità dell'hardware e la gratuità del software, scaricabile dal sito web di Arduino;
- la scheda che contiene tutto ciò che serve per un utilizzo di base;
- la semplicità di collegamento alla scheda, sia verso il computer (USB) sia verso gli altri circuiti elettronici, mediante i connettori sulla scheda;
- la relativa semplicità della programmazione in C, che non richiede quindi la conoscenza del linguaggio assembler del microcontrollore;
- la disponibilità di librerie di funzioni che semplificano il pilotaggio di display LCD, motori, trasduttori, servomotori;
- l'abbondanza di software già sviluppato per Arduino;
- la disponibilità di schede aggiuntive (denominate *shield*), realizzate da vari produttori, per interfacciare la scheda Arduino a motori, a carichi di potenza, a trasmettitori wireless, alla rete internet.

Iniziare a lavorare con Arduino

Si evidenziano di seguito le principali operazioni che consentono di iniziare a lavorare con il sistema Arduino:

- scaricare l'ultima versione del software Arduino dal sito <http://arduino.cc>;
- installare il software e i driver;
- collegare la scheda Arduino ad una presa USB per alimentarla e programmarla; si accende il LED verde ON; una volta programmato Arduino potrà funzionare scollegato dal computer (stand-alone), fornendo l'alimentazione da 7 V a 12 V sull'apposito connettore (il + è il contatto interno);
- lanciare il software Arduino;
- aprire lo sketch di esempio che fa lampeggiare il LED L da *File > Examples > 1.Basics > Blink*;
- selezionare la scheda corrispondente al modello collegato (ad esempio Arduino Uno) da *Tools > Board*;
- selezionare la porta USB utilizzata da *Tools > Serial Port*; se l'upload non dovesse funzionare selezionare un'altra porta seriale;

- compilare e caricare (*Upload*) su Arduino lo sketch utilizzando il pulsante (); dovrebbero lampeggiare i LED RX e TX; al termine dell'uploading (messaggio *Done uploading*) lo sketch caricato (*Blink*) dovrebbe far lampeggiare il LED L collegato al pin 13;
- selezionare *Help > Troubleshooting* in caso di problemi;
- caricare altri esempi premendo il pulsante Open ();
- selezionare *Help > Reference* per la sintassi del linguaggio di programmazione di Arduino.

Caratteristiche della scheda Arduino Uno

La scheda *Arduino Uno* presenta le seguenti caratteristiche salienti:

- presa USB per il collegamento al computer durante la programmazione e la messa a punto del programma;
- presa per l'alimentazione (da 7 V a 12 V) per consentire il funzionamento autonomo (*stand-alone*) una volta che la scheda viene scollegata dal computer e inserita nel circuito a cui è destinata;
- regolatore di tensione interno che produce la tensione 5 V stabilizzata necessaria per il funzionamento degli integrati;
- tensioni presenti sui 6 ingressi analogici acquisite e convertite in digitale a 10 bit mediante opportune istruzioni;
- possibilità di impiego dei 14 pin digitali come ingressi o come uscite mediante opportuna programmazione;
- possibilità di utilizzare alcuni pin digitali (3, 5, 6, 9, 10, 11) come uscite pilotate in PWM mediante opportune istruzioni per il controllo di alcuni attuatori come motori, lampade, riscaldatori, servomotori;
- frequenza del segnale di clock 16 MHz.

Note

- 1) La modulazione PWM modifica il duty-cycle di un'onda rettangolare (dallo 0% al 100%) in base al valore della grandezza modulante, un numero compreso tra 0 e 255 impostato utilizzando l'istruzione *analogWrite*.
- 2) Per definire come ingresso o come uscita i pin digitali (che operano con tensioni comprese tra 0 V e 5 V e con una corrente massima di 40 mA) si utilizzano le istruzioni *pinMode()*, *digitalWrite()* e *digitalRead()*.
- 3) Le tensioni presenti sugli ingressi analogici sono normalmente comprese tra 0 V e 5 V; mediante il pin AREF e l'istruzione *analogReference()* è possibile cambiare il fondoscala.
- 4) Una visione più dettagliata della scheda riportata in figura 3 mostra in particolare la suddivisione dei terminali in *Digital*, *Power* e *Analog*.



Figura 3 Visione dettagliata della piedinatura della scheda

Programmazione

Si riportano di seguito alcune informazioni di carattere generale sulla programmazione di Arduino:

- per verificare la *correttezza sintattica* del programma scritto si deve selezionare l'opzione \checkmark (*verify*) posta in alto a sinistra nell'IDE;
- tutte le istruzioni (tranne `#define` e `#include <nome-libreria>`) terminano con `;` (punto e virgola);
- il simbolo `//` significa *commento singola linea*;
- il simbolo `/*...*/` significa *commento linee multiple*;
- per richiamare le parentesi graffe `{}` si deve premere `Alt+123` e `Alt+125`.

Nella tabella 2 vengono riportati i principali operatori aritmetici, logici e di comparazione utilizzati nel linguaggio di programmazione.

Operatori aritmetici	
=	assegnazione
+	Somma
-	Differenza
x++	incremento di x
x--	decremento di x
*	Prodotto
/	Quoziente
%	resto della divisione tra interi
Operatori logici	
&&	prodotto logico
	somma logica
!	negazione
Operatori di comparazione	
x==y	uguaglianza tra x e y
x!=y	x diverso da y
x<y	x minore di y
x>y	x maggiore di y
x<=y	x minore o uguale a y
x>=y	x maggiore o uguale a y

Tabella 2 Principali operatori aritmetici, logici e di comparazione

Nella tabella 3 vengono riportati i principali tipi di variabile utilizzati nel linguaggio di programmazione.

Char	carattere ASCII	1byte	
Int	intero	2 byte	da -32768 a 32767
Long	intero	4 byte	da -2147483648 a 2147483647
Float	virgola mobile	4 byte	da 3.4028235E+38 a -3.4028235E+38
Array	vettore		<u>Esempio</u> <code>int myArray[10]={9,3,2,4,3,2,7,8,9,11}</code> vettore di 10 elementi numerati da 0 a 9 ed elencati tra graffe (richiamando <code>myArray[9]</code> si ottiene il valore 11)

Tabella 3 Principali tipi di variabile

Struttura di uno sketch

La struttura di un programma (*sketch*) è la seguente:

```
/*
Titolo
Descrizione
(commento di più righe)
*/
Definizioni
-----
Esempi
#define ledPin 3 (il compilatore sostituisce il valore 3 a ledPin nel programma)
int ledPin = 13; (LED connesso al pin 13)
#include <LiquidCrystal.h> (includi la libreria display LCD)
-----
void setup() {
  codice di SETUP
  (eseguito una sola volta e mantenuto fino al caricamento di un altro sketch)
  .....; //commento
  .....; //commento
}
void loop() {
  codice del PROGRAMMA PRINCIPALE
  (eseguito ripetutamente finché la scheda non viene spenta o riprogrammata)
  .....; //commento
  .....; //commento
}
eventuali FUNCTIONS richiamate nel programma
(fine dello sketch)
```

Le funzioni *void setup()* e *void loop()* sono essenziali per questo linguaggio in quanto consentono rispettivamente di definire:

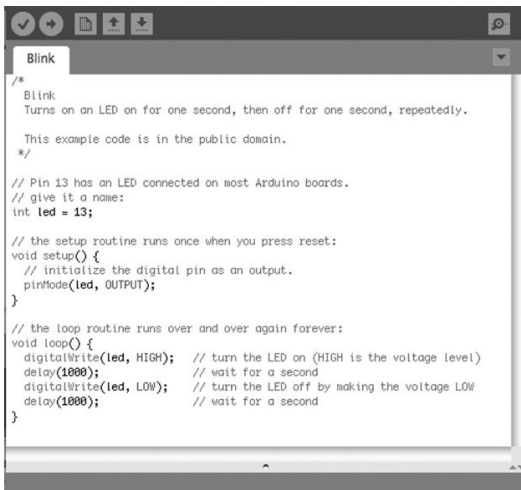
- le impostazioni necessarie per il funzionamento dei dispositivi collegati con la scheda;
- i blocchi di istruzioni necessari per svolgere le funzioni che vengono richieste al programma.

La funzione **setup()** è la prima ad essere eseguita dopo ogni accensione o reset del sistema.

Viene utilizzata in particolare per inizializzare le variabili, per impostare lo stato dei pin, per l'impostazione delle comunicazioni seriali.

La funzione *loop()* esegue ciclicamente il programma definito al suo interno.

La figura 4 riporta un esempio di sketch scritto nell'IDE che produce il lampeggio del LED montato sulla scheda e collegato al pin 13.



```
Blink
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Figura 4 Esempio di programma