1 Che cos'è Arduino?

Nella Fig. 1 è riprodotta la scheda Arduino.

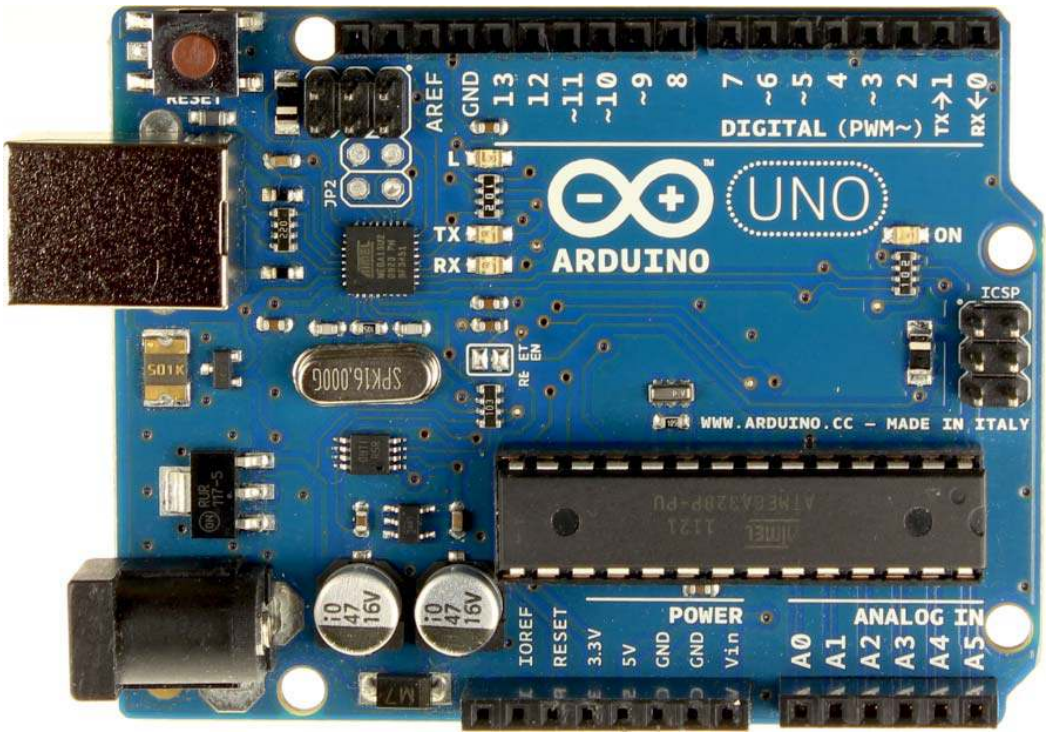


Fig.1
Arduino UNO.

Sul sito ufficiale *www.arduino.cc* si trova la seguente definizione:

> Arduino è una piattaforma elettronica open source basata su software e hardware facili da usare. È pensata per chiunque voglia costruire progetti interattivi.[1]

In realtà la trovate in inglese (vedi nota 1), come tutto il sito di cui non esiste una versione italiana. Già questo ci dice che chi l'ha progettata voleva proporla a tutto il mondo.

Con il termine **piattaforma** si intende un **sistema completo hardware e software** (cioè oggetti e programmi) per costruire dispositivi elettronici di vario genere. Il gruppo di Arduino, infatti, ha progettato la scheda e realizzato anche un software per programmarla. Adesso esistono moltissime schede Arduino ma la più nota e utilizzata è la **Uno R3**[2] riportata in Fig. 1. In Fig. 2 la prima scheda Arduino 10000[3] e l'ultima uscita, **Galileo**, realizzata in collaborazione con Intel (la multinazionale che produce microprocessori per PC, Tablet e Smartphone).

---

[1] *Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.*

[2] R3 sta per *revision 3*, a indicare che è la terza versione della scheda Uno.

[3] Arduino 10000 si chiama così perché ne erano state prodotti appunto 10000 pezzi.

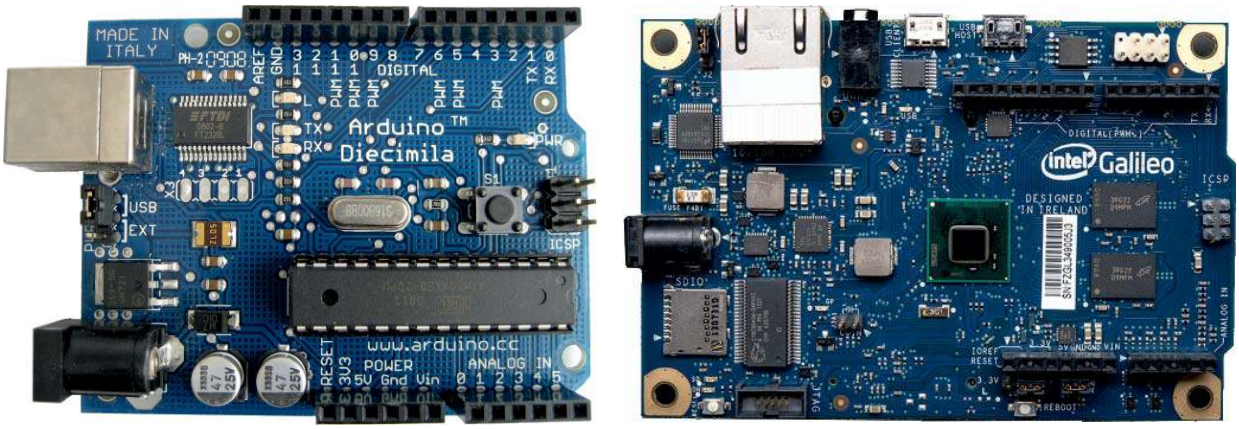


Fig.2
Arduino Diecimila (il capostipite) e la scheda Intel Galileo.

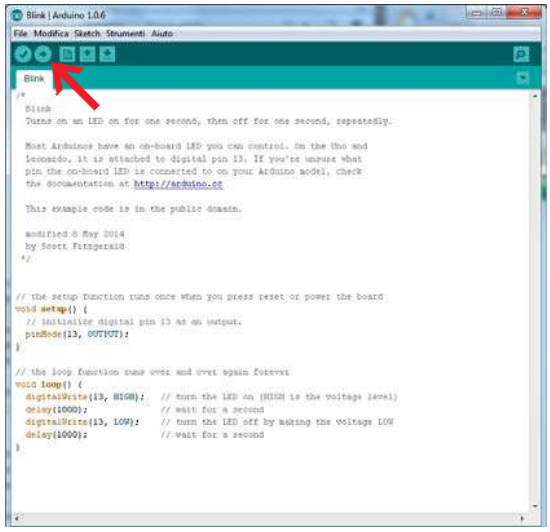


Fig. 3
L'interfaccia del software di programmazione di Arduino.

Il software realizzato dal gruppo Arduino (in Fig. 3 la schermata di avvio) consente di programmare la scheda per realizzare moltissimi tipi di dispositivi. Digitando la parola "Arduino" su *YouTube*, appariranno milioni di video che descrivono le potenzialità della scheda.

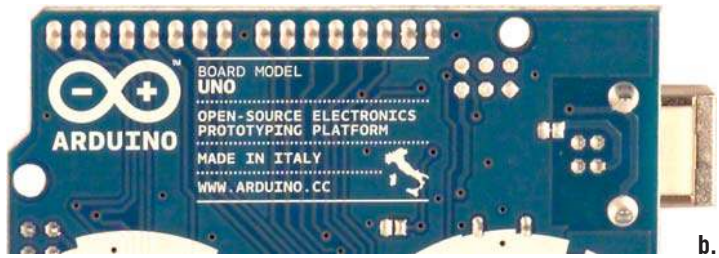
La storia di Arduino

Perché Arduino?

La scheda è stata inventata da un gruppo di ricercatori dell'Interaction Design Institute di Ivrea. La versione ufficiale ne associa il nome a un personaggio storico, *Re Arduino*, piemontese, considerato il primo re d'Italia (vedi fig. 4). Una versione più corrente vuole che il nome derivi dal quello del bar che gli inventori frequentavano.

Fig.4
a. Re Arduino di Ivrea (in una medaglia d'epoca) che si dice abbia dato il nome alla scheda;
b. la serigrafia sulla board che certifica il prodotto come Made in Italy.

a.



b.

Comunque sia, resta il fatto che gli autori volevano dare risalto all'italianità del progetto.[4]

[4] La città di Ivrea, in Piemonte, era considerata il centro tecnologico più avanzato in ambito di macchine programmabili; negli anni '70 infatti, ospitava la Olivetti, azienda, al tempo, leader nel settore.

Open source e Open hardware

Open source e *Open hardware* sono due concetti molto importanti, uno legato al software e l'altro alle schede hardware.

Fig. 5
I simboli dei movimenti *open source* e *open source hardware.*

In entrambi i casi significa che l'inventore lascia la sua invenzione "aperta", cioè lascia che il programma e il dispositivo possano essere copiati, distribuiti, modificati o migliorati.

Open source

Per quanto riguarda i programmi, abbiamo a disposizione il **programma sorgente** (da cui il termine ***source***) scritto nel linguaggio originario. Possiamo leggerlo, studiarlo, modificarlo e crearne una nuova versione che risponda alle nostre esigenze. Quando invece compriamo un programma (un gioco o, per esempio, un software per l'ufficio) compriamo solo il codice binario, cioè il codice numerico che solo il PC capisce perfettamente. In Fig. 6 si può notare la differenza tra il codice sorgente (pur complesso, ma scritto in forma comprensibile, in lingua inglese) e il codice eseguibile (formato solamente da numeri esadecimali).

a.

```
// these constants describe the pins. They won't change:
const int groundpin = 18;            // analog input pin 4 -- ground
const int powerpin = 19;             // analog input pin 5 -- voltage
const int xpin = A3;                 // x-axis of the accelerometer
const int ypin = A2;                 // y-axis
const int zpin = A1;                 // z-axis (only on 3-axis models)

void setup()
{
  // initialize the serial communications:
  Serial.begin(9600);

  // Provide ground and power by using the analog inputs as normal
  // digital pins.  This makes it possible to directly connect the
  // breakout board to the Arduino.  If you use the normal 5V and
  // GND pins on the Arduino, you can remove these lines.
  pinMode(groundpin, OUTPUT);
  pinMode(powerpin, OUTPUT);
  digitalWrite(groundpin, LOW);
  digitalWrite(powerpin, HIGH);
}

void loop()
{
  // print the sensor values:
  Serial.print(analogRead(xpin));
  // print a tab between values:
  Serial.print("\t");
  Serial.print(analogRead(ypin));
  // print a tab between values:
  Serial.print("\t");
  Serial.print(analogRead(zpin));
  Serial.println();
  // delay before next reading:
  delay(100);
}
```

b.

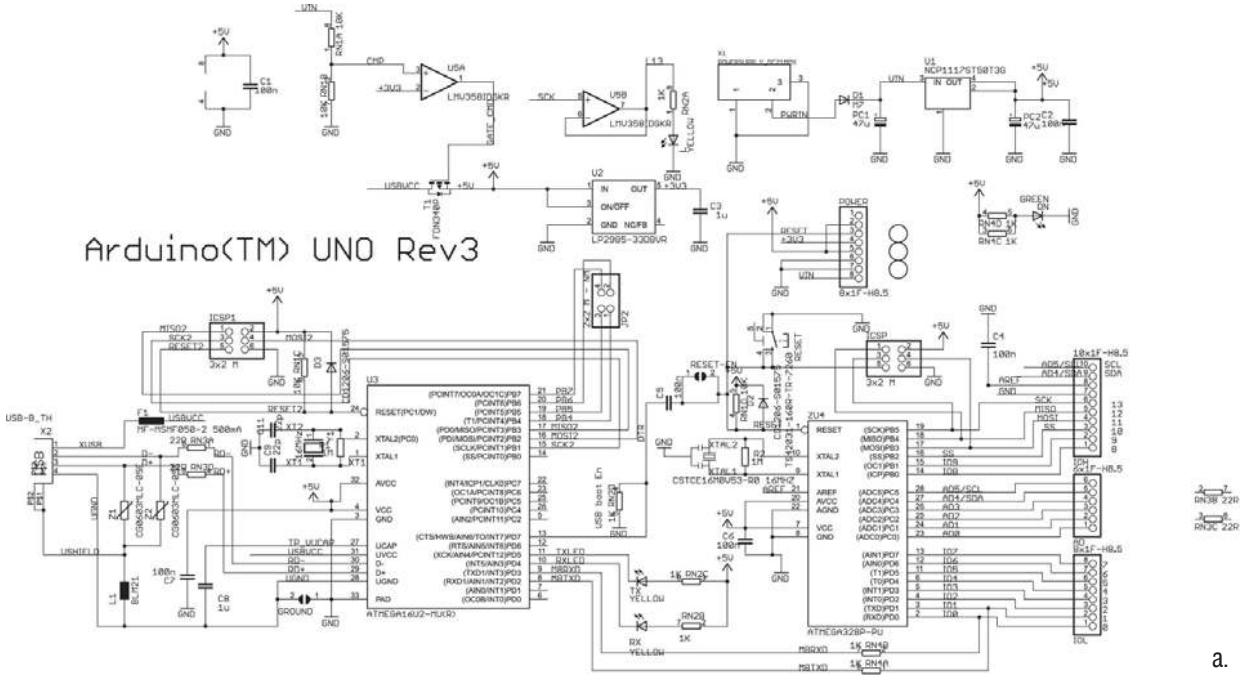
```
 1 0D0A2F2A0D0A204144584C3378780D0A
 2 200D0A2052656164732061E20416E61
 3 6C6F67204465766963657320414E44584C
 4 337878206163636C65726F6D657465
 5 7220616E6420636F6D6D756E69636174
 6 6573207468650D0A20616363656C6572
 7 6174696F6E20746F2074686520636F6D
 8 70757465722E20205468652070696E73
 9 207573656420617265206465736E
10 656420746F20626520656173696C790D
11 0A20636F6D70617469626C6520776974
12 6820746865206272656B6F75742062
13 6F617264732066726F6D20537061726B
14 66756E2C20617661696C61626C652066
15 726F6D3A0D0A20687474703A2F2F7777
16 772E737061726B66756E2E636F6D2F63
17 6F6D6D657263652F63617465676F7269
```

Fig. 6
a. Codice sorgente di un programma;
b. Lo stesso programma compilato (trasformato) per renderlo comprensibile al PC.

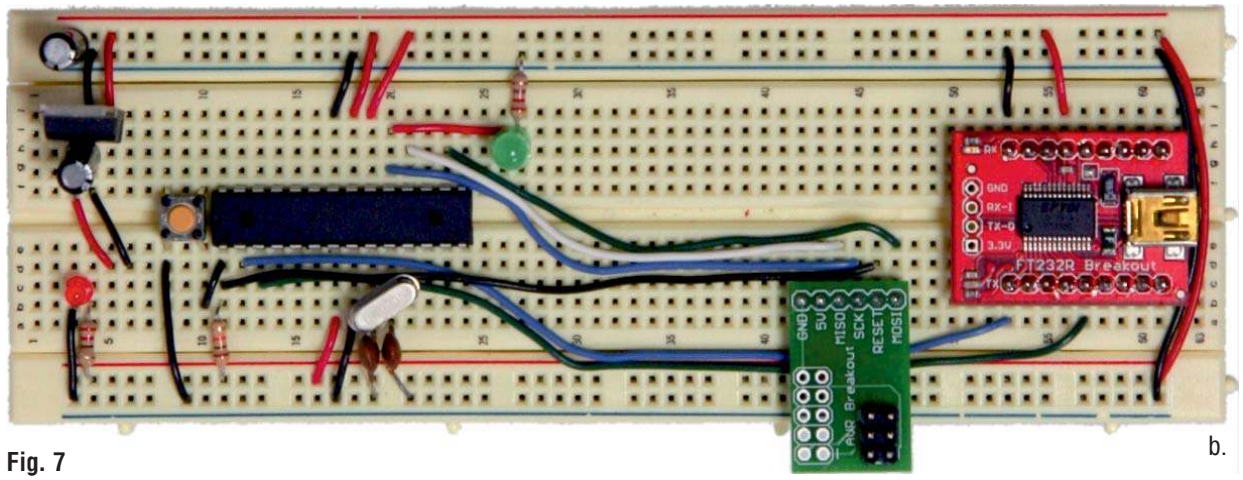
Open source hardware

L'**Open source hardware** è un concetto più recente e praticamente è stato inventato dal gruppo di Arduino. Si tratta di fornire insieme al dispositivo elettronico tutta la documentazione disponibile e il progetto completo. In questo modo è possibile costruire da soli un modello equivalente e persino venderlo, a patto che si dichiari che è basato su Arduino.

In Fig 7.a sono visualizzati gli schemi di Arduino; in Fig 7.b è presentata una versione di Arduino costruita su breadboard assemblando i diversi componenti.



a.



b.

Fig. 7
a. Schema di Arduino Uno rev. 3.
b. Realizzazione su breadboard.

2 Che cosa c'è dentro Arduino?

Le altre board di Arduino

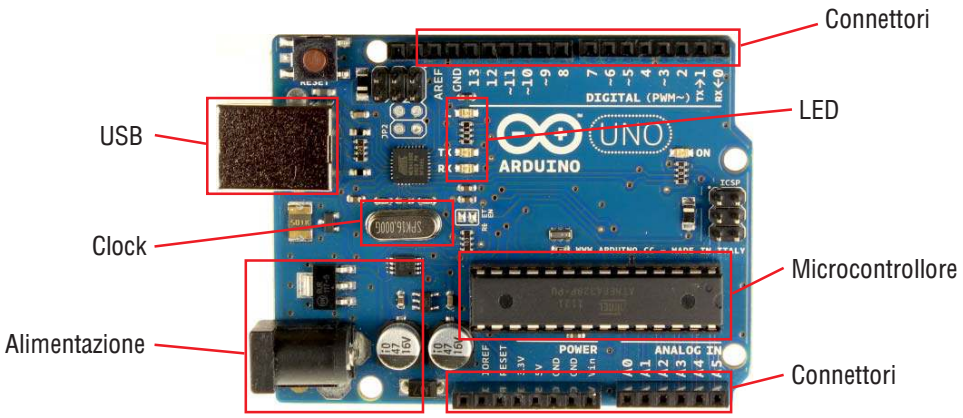
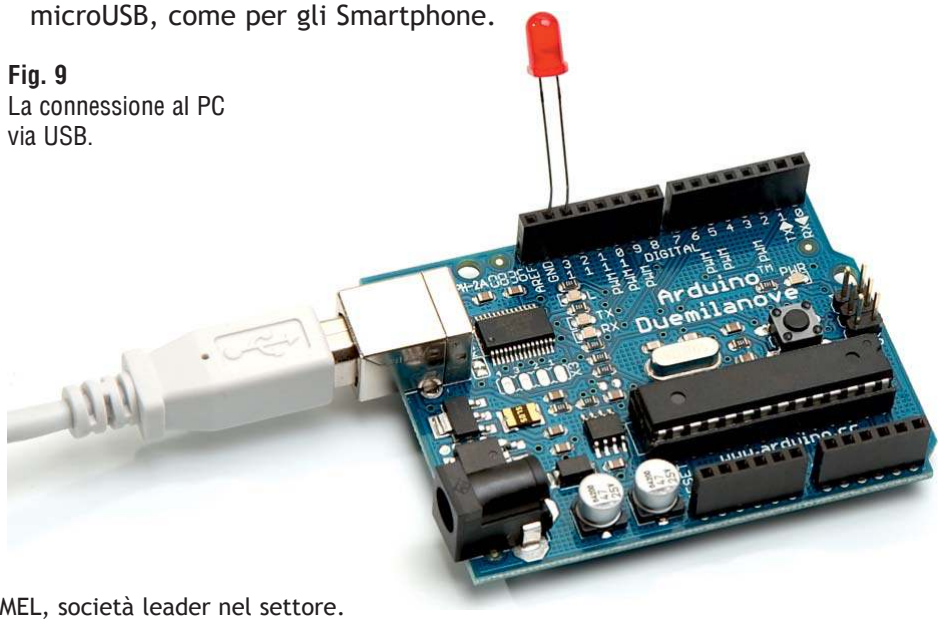


Fig. 8
La scheda Arduino. In evidenza i componenti principali.

Descriviamo brevemente le parti di cui è composta la scheda:

- **Microcontrollore**: è la CPU della scheda[5], cioè il processore che esegue i programmi che si scrivono per la scheda stessa. Rispetto ai processori dei PC è meno potente, ma più completo e contiene tutto quello che serve per eseguire programmi di media complessità: una memoria Flash da 32 KByte (che non si cancella quando si spegne tutto; in pratica è l'equivalente, in forma ridotta, del disco fisso dei PC), una memoria RAM da 2 KByte e altro di cui parleremo più avanti.
- **Clock:** è un generatore di frequenza al quarzo che fornisce il ritmo di lavoro al microcontrollore e rappresenta quindi la velocità di lavoro del processore. Il clock può arrivare fino a 20 MHz, anche se sulla scheda ne è montato uno meno potente, da 16 MHz. Ciò vuol dire che questo può arrivare a eseguire fino a circa 20 milioni di operazioni al secondo.
- **USB**: è l'interfaccia di connessione al PC, attraverso la quale inviare il programma dal PC alla scheda. In Fig. 9 è visualizzata la connessione USB con connettore tipo B. Su alcuni più recenti modelli di schede, la connessione è microUSB, come per gli Smartphone.

Fig. 9
La connessione al PC via USB.



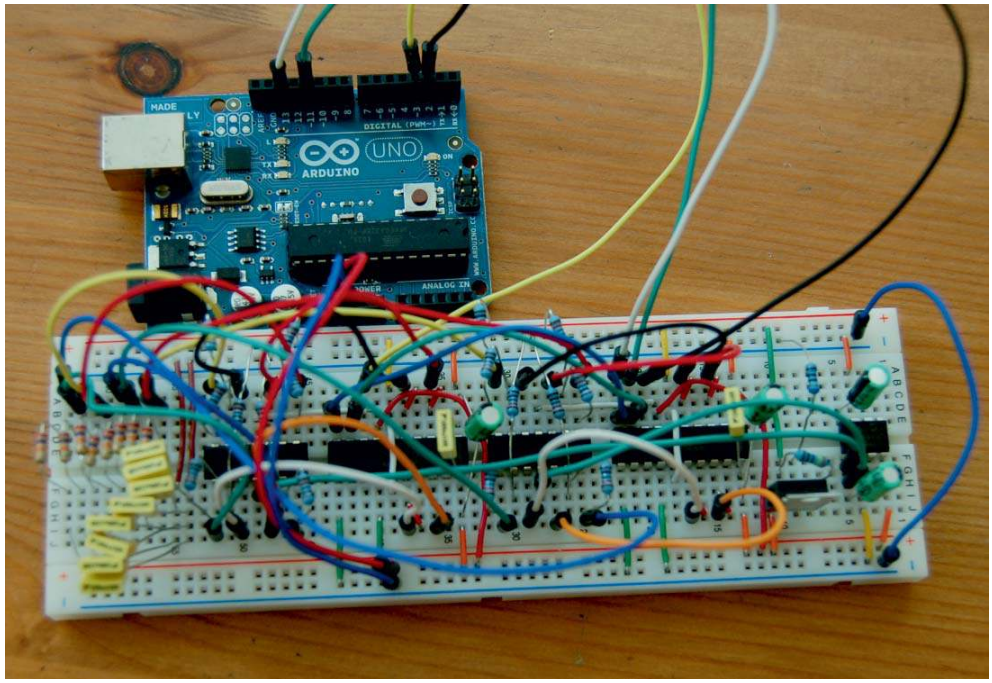
[5] Un *ATmega 328* di ATMEL, società leader nel settore.

- **Alimentazione:** come tutti i dispostivi elettronici, il sistema ha bisogno di energia. La tensione richiesta di norma è 5 V anche se il microcontrollore funziona con tensioni da 2 a 5,5 volt circa. Essa può essere alimentata da una batteria a 9 V, da cui la scheda ricava i 5 V necessari, oppure da un opportuno alimentatore. In alternativa, si può usare direttamente la porta USB del computer collegata alla presa sulla scheda. Non appena si collega l'alimentazione, il microcontrollore comincia a eseguire il programma che ha in memoria.

Fig. 10
Alimentazione ausiliaria della scheda.

- **Connettori**: sono i punti dove collegare i vari dispositivi che compongono l'automatismo che si vuole creare (LED, Buzzer, Motori, sensori di vario genere e tutto quello che serve al nostro progetto). Ogni linea corrispondente a una connessione sulla scheda è individuata da un numero che va utilizzato nel programma per specificare a quale linea si riferiscono le istruzioni del programma stesso.

Fig. 11
Circuito realizzato su breadboard che scambia segnali con la scheda Arduino.



- **LED:** sulla scheda sono presenti tre luci a LED (vedi Fig. 8). Un LED marcato con la L è collegato a un punto sul connettore (linea 13) ed è utilizzabile per i primi programmi e per segnalare stati particolari del programma. I LED TX e RX, invece, segnalano la comunicazione tra la scheda e il PC (o altri dispositivi seriali collegati). Se questi lampeggiano, è segno che PC e Arduino stanno comunicando.

3 Che cosa fa Arduino?

Dopo aver detto che cosa contiene, la domanda interessante è: che cosa fa, o meglio, che cosa possiamo farci? Infatti, l'idea alla base di Arduino è che ognuno possa costruire con esso qualcosa di utile o semplicemente di piacevole.
I progetti possibili sono pressoché infiniti. Di seguito, ne elenchiamo alcuni che ci sembrano simpatici o interessanti. In realtà, praticamente tutti i moderni oggetti tecnologici possono essere costruiti con Arduino (entro i limiti delle prestazioni della scheda). Infatti, il sistema comprende tutto ciò che serve (una CPU, una memoria e delle interfacce per sensori e attuatori).
Molti progetti presentati sono pubblicizzati sul sito di informazione *wired.it*, un importante riferimento internazionale legato all'innovazione tecnologica.

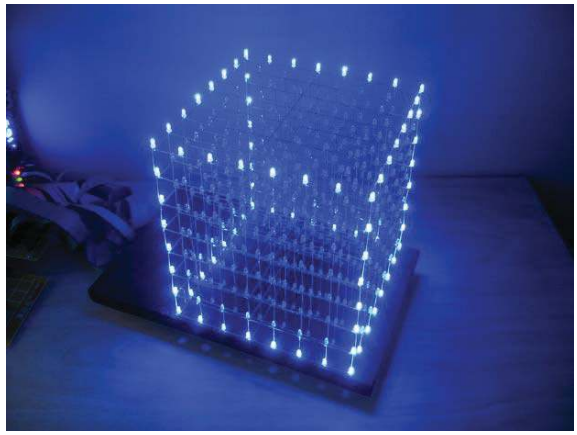


Fig. 12
Cubo a LED.

- **Cubo LED:** è una applicazione di Arduino, in cui la scheda controlla i singoli LED e può produrre effetti molto particolari, come ondulazioni della luce, rotazioni, visualizzazioni di numeri e di immagini.

- **Stampante 3D REP RAP**

http://reprap.org/

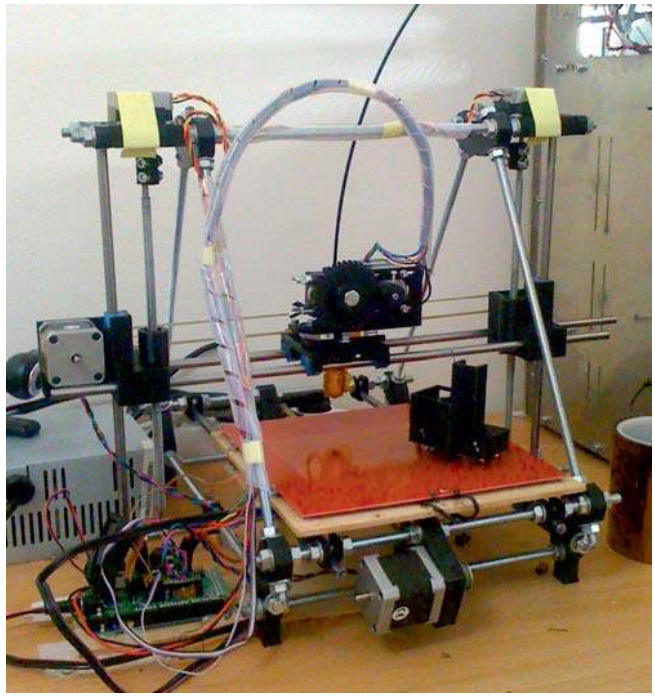


Fig. 13
Stampante 3D REP RAP, costruita in kit.

Con la **stampante 3D REP RAP** (ormai famosa in tutto il mondo) è possibile stampare oggetti tridimensionali a partire dal file del modello 3D. La scheda Arduino controlla i motori e l'estrusore che deposita la plastica (derivata dal mais) in strati successivi, fino al prodotto finito.

• **Rilevatore di fumo, gas, ecc.**

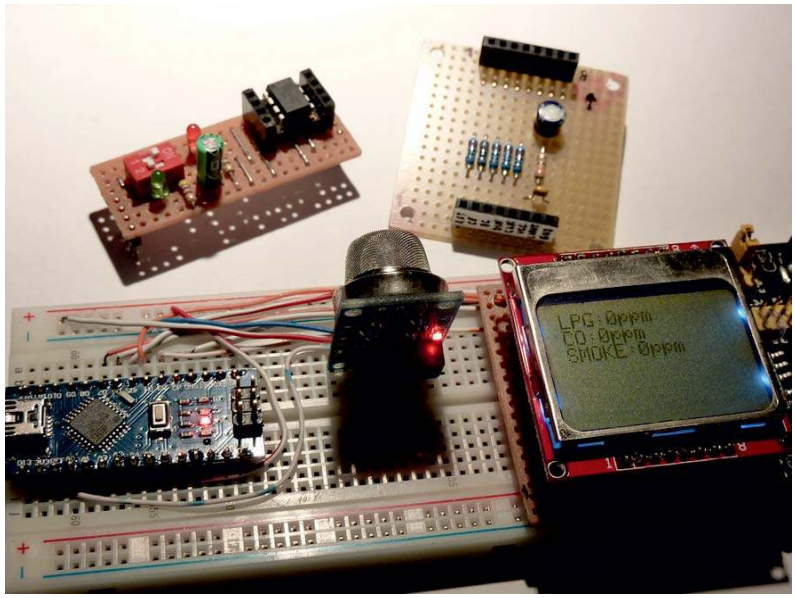


Fig. 14
Realizzazione di un rivelatore di fumo con un sensore di gas tipo MQ2.

Con un opportuno sensore è facile costruire un rilevatore di fumo e altri gas (metano, ecc.). Dotando il dispositivo di un cicalino (*buzzer*) è possibile realizzare un allarme che suona se viene superata una certa concentrazione di fumo o gas nell'ambiente. Il sensore (in Fig. 14, somigliante a un piccolo microfono) rileva la concentrazione di vari gas visualizzata sul display. Cambiando il tipo di sensore si può costruire un etilometro (soffiando sul sensore sappiamo se ci è consentito guidare oppure no).

• **Serratura a combinazione**

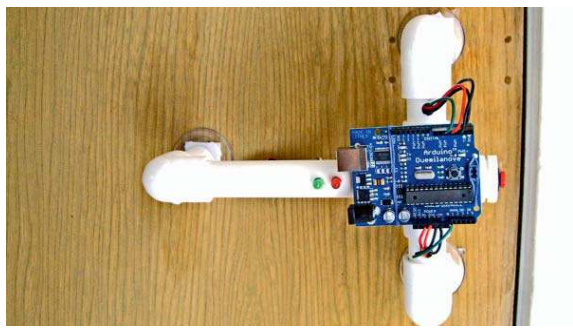


Fig. 15
Realizzazione di un sistema di protezione con sensore piezoelettrico.

La porta si apre solo se si bussa rispettando una certa sequenza, come nei film di spionaggio. Grazie ad un *knock sensor,* costituito da un elemento piezoelettrico che rileva le vibrazioni meccaniche, possiamo far leggere ad Arduino la sequenza di colpi e confrontarla con una "combinazione" memorizzata.

• **Tweet a watt**

Fig. 16
Sistema di controllo dell'energia assorbita.

Per monitorare l'energia impegnata, il dispositivo invia un messaggio via *Twitter* con il consumo rilevato.

• **Smart tosaerba**

Fig. 17
Robot tosaerba.

Il robot tosaerba si muove e taglia l'erba in tutto il giardino. Ne esistono versioni con pannelli solari, con funzioni di apprendimento (impara il percorso e lo riproduce su comando) e con sensore di luce e di pioggia.

• **Arduino cura la pianta**

Fig. 18
Gestione dell'irrigazione di una pianta con rilevatore di umidità e connessione di rete.

La scheda regola l'irrigazione e la luminosità ambientale sulla base di un programma preimpostato e in funzione del grado di umidità rilevato nel terreno. Dotando la scheda di una connessione a Internet, è possibile rilevare, da ogni luogo del mondo, le condizioni della pianta e impostare opportunamente (da PC, Tablet o Smartphone) i cicli di irrigazione e illuminazione.

Concludiamo con due progetti molto diffusi e i cui prototipi sono stati sviluppati con Arduino:

• **Smartwatch**
Lo *Smartwatch* è un orologio con funzionalità aggiuntive, in genere abbinato a uno Smartphone da cui riceve messaggi e segnalazioni.

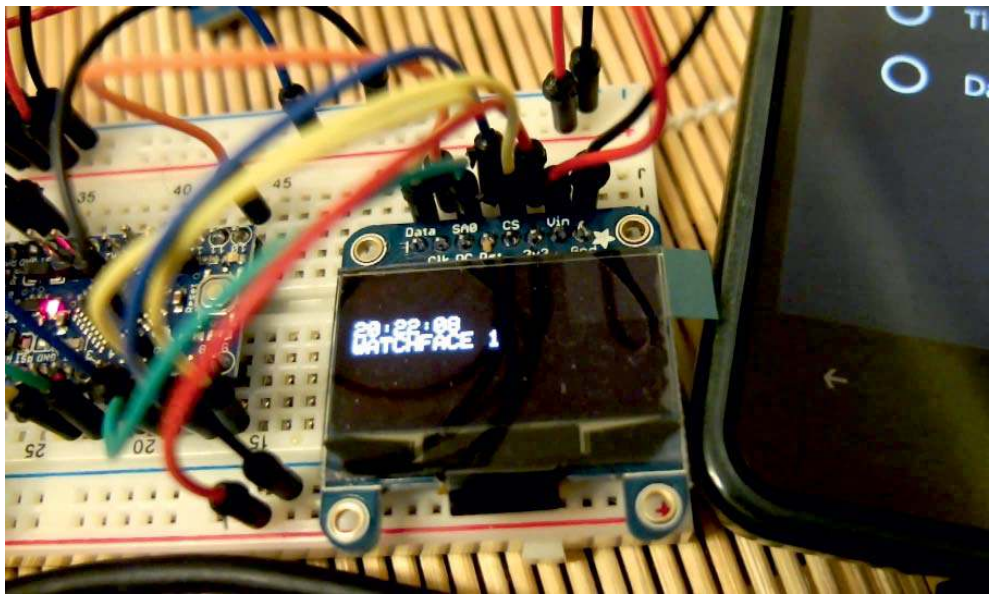


Fig. 19
Prototipo di *smartwatch* con Arduino.

• **Quadricottero**

Fig. 20
Quadricottero controllato da Arduino.

Il quadricottero è un aeromodello dotato di quattro motori e relative eliche che può eseguire le manovre tipiche di un elicottero.

Risorse

Il Web offre numerose risorse per utilizzare Arduino: materiali, tutorial, video, guide, ecc.
Si può cercare ciò che interessa su un qualsiasi motore di ricerca e trovare migliaia di risposte. Questa diffusione è una delle principali ragioni del successo di Arduino. La sua semplicità e la sua impostazione *open* hanno fatto in modo che i progetti disponibili venissero immediatamente condivisi tra tutti.
Di seguito, sono elencati alcuni principali link:

Progetti per cominciare

- *www.arduino.cc* – Il sito originale, da dove partire.
- *scuola.arduino.cc* - Il sito che raccoglie alcuni tutorial del team Arduino.
- *playground.arduino.cc* - Il forum, dove trovare risposte a vari quesiti.
- *fritzing.org/home* - Il programma *Fritzing* è un software (*open source*) per disegnare circuiti basati su Arduino. Esso contiene moltissimi esempi già svolti e molto semplici da provare.
- *learn.adafruit.com/category/learn-arduino* - *Adafruit* è un importante rivenditore americano di componenti elettronici. La sua sezione di apprendimento (*learning*) è davvero ottima.
- *learn.sparkfun.com* - come il precedente, *Sparkfun* vende componenti in tutto il mondo.
- *www.instructables.com/howto/arduino* - letteralmente, istruzioni per ... quasi tutto.

4 Il sistema minimo

Arduino è un sistema minimo, che richiede solo un cavo USB per la connessione al computer. Sulla scheda c'è l'oscillatore per il clock del microcontrollore, l'interfaccia USB per la programmazione seriale del microcontrollore stesso e la sezione di alimentazione (direttamente dal cavo USB o dall'alimentatore). Alcuni semplici componenti possono essere collegati direttamente alla board (Fig. 21).

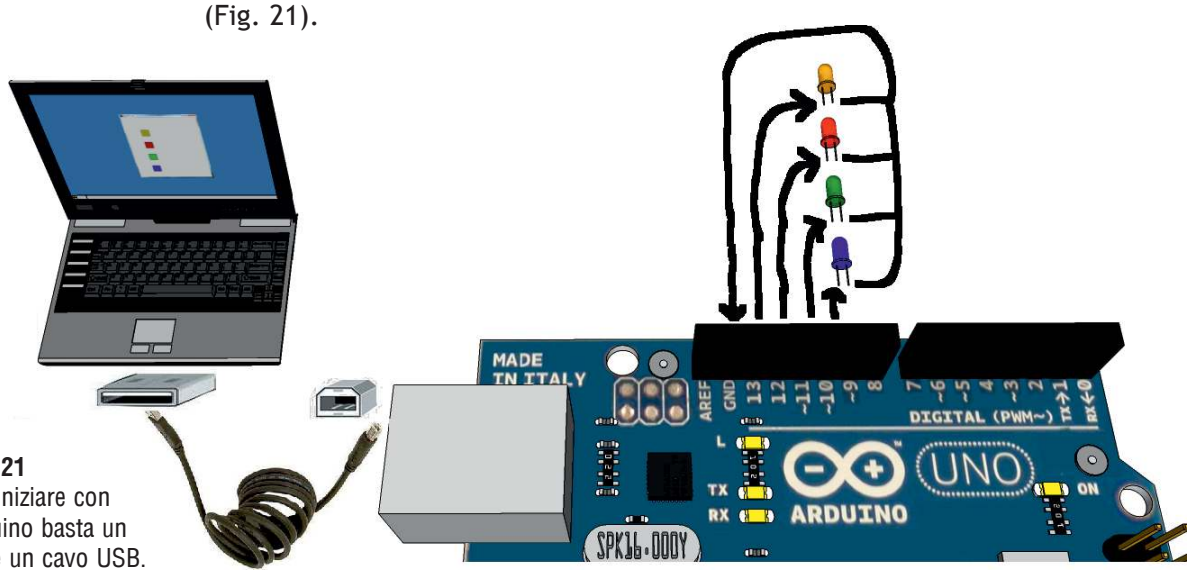


Fig. 21
Per iniziare con Arduino basta un PC e un cavo USB.

Per i progetti successivi può bastare una normale breadboard (basetta) e qualche cavo per le connessioni (Fig. 22a).
La **breadbord** è una scheda per montaggi sperimentali di elettronica, che comprende una serie di fori con connettori elastici, collegati tra loro, come indicano le linee grigie di Fig. 22b. Ciò permette di inserire e collegare i diversi componenti, di alimentarli collegando le coppie di linee poste ai margini superiore e inferiore della breadboard rispettivamente alla tensione elettrica e a massa direttamente dai connettori della scheda Arduino.

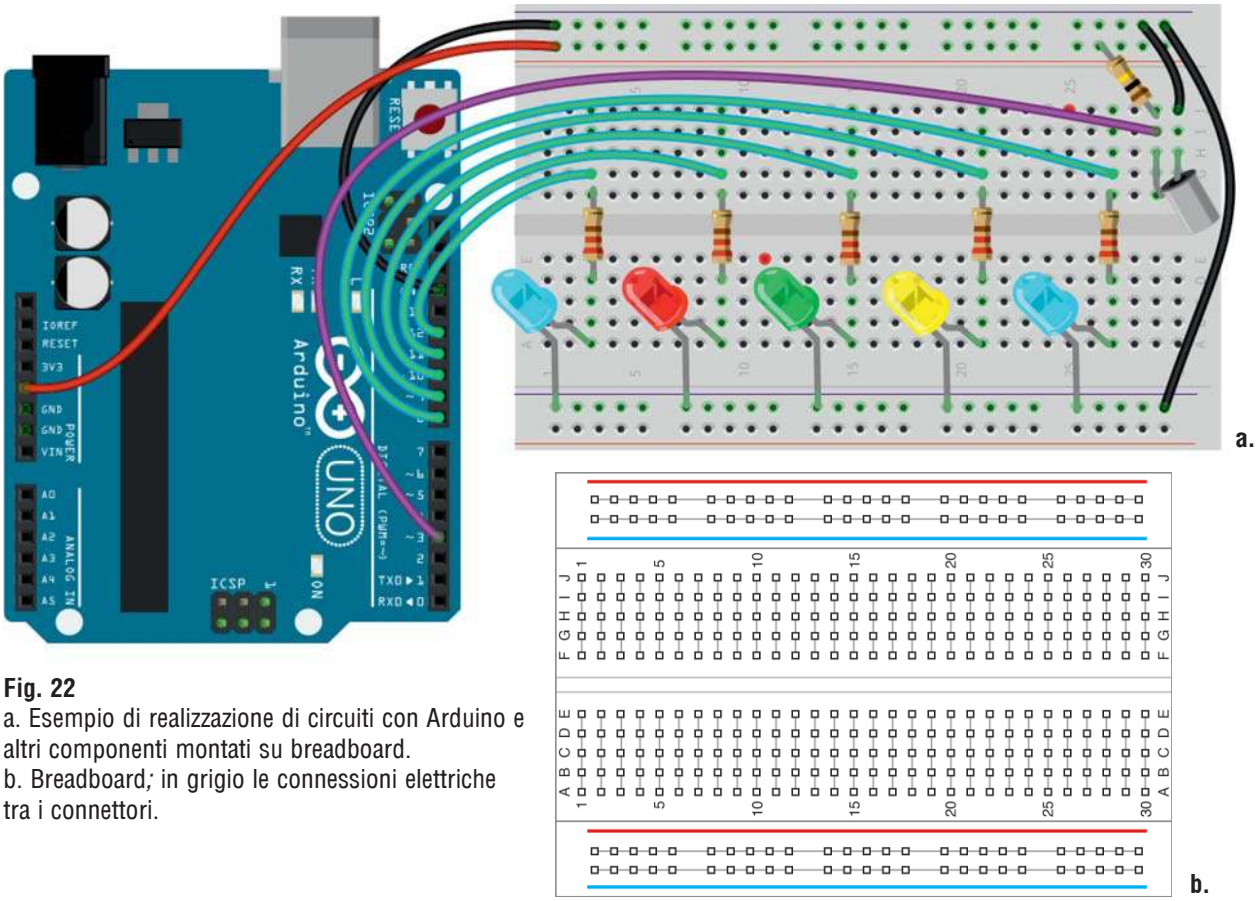


Fig. 22
a. Esempio di realizzazione di circuiti con Arduino e altri componenti montati su breadboard.
b. Breadboard; in grigio le connessioni elettriche tra i connettori.

Prima di iniziare a programmare la scheda Arduino, esaminiamo le funzioni dei diversi tipi di connettori disponibili sulla scheda, per l'ingresso dei dati e l'uscita dei risultati.

- I pin da 0 a 13 sono previsti per ricevere segnali digitali in ingresso (IN) o fornirli in uscita (OUT). Il segnale digitale può avere due soli valori: 0 (livello basso = 0 V) oppure 1 (livello alto = 5 V); in generale, è meglio non usare le linee 0 e 1 (sono linee di comunicazione tra PC e Arduino) e la linea 13 (connessa al LED montato sulla board e in genere usata solo a quello scopo).
- i pin A0 ÷ A5 sono previsti per segnali analogici in entrata. I segnali analogici possono assumere valori di tensione compresi tra 0 e 5 V in base alla informazione che trasportano. In realtà, con il codice adeguato, tali linee possono essere usate anche come linee digitali.

5 Il primo programma in C

Cominciamo subito a realizzare una prima semplice applicazione con Arduino: un **LED lampeggiante**.
Per fare questo non dobbiamo costruire alcun circuito esterno alla scheda di Arduino, in quanto sulla scheda stessa è già montato un LED che può essere opportunamente comandato dal programma in esecuzione. Questo esercizio è anche un buon modo per verificare se tutto funziona (scheda, collegamento al PC e programmazione).

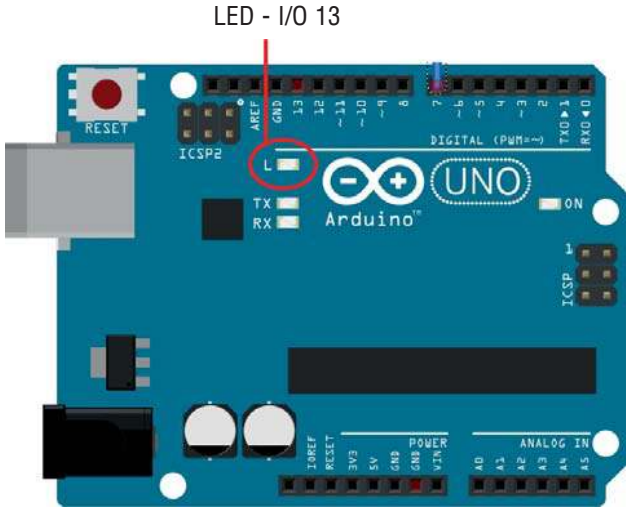


Fig. 23
Primo semplice progetto con Arduino: LED lampeggiante, realizzato con *Fritzing*.

Arduino, come tutti i sistemi programmabili, può essere programmato in diversi modi. Il primo che affrontiamo consiste nell'utilizzare l'ambiente di sviluppo (IDE)[6] progettato dallo stesso team che ha inventato Arduino. Questo ambiente prevede la programmazione in **linguaggio C** (è sostanzialmente una variante del C standard): i comandi da impartire alla scheda vengono codificati attraverso una sequenza di istruzioni scritte in un file di testo (vedi Fig. 24) che vengono successivamente compilate[7] e inviate alla scheda.

Fig. 24
Esempio di programma C per Arduino.

```
// 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
}
```

Il listato può sembrare molto complesso, ma nell'applicazione Arduino sono compresi esempi pronti che possono essere utilizzati e modificati.

[6] IDE: *Integrated Development Environment*.

[7] La compilazione consiste nel trasformare il programma C in una sequenza di codici binari comprensibili per il microcontrollore.

Installazione dell'ambiente di programmazione

Dal sito *http://arduino.cc/en/Main/Software* si può scaricare l'ambiente di programmazione per Arduino e installarlo sul proprio PC seguendo la procedura guidata.[8]

Fig. 25
Finestre proposte dal software di installazione.

A un certo punto dell'installazione verrà chiesto se installare il driver USB necessario per la comunicazione tra PC e Arduino (via USB appunto). Clic su *Installa* per procedere.

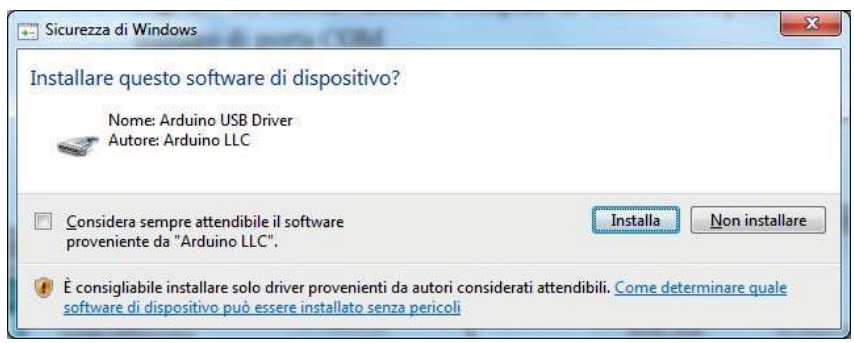


Fig. 26
Richiesta per l'installazione driver USB.

Se non si dovesse installare il driver USB, lo si può fare successivamente da *Pannello di controllo*, usando i driver che si trovano nella relativa cartella (*drivers/*).
Per l'utilizzo dell'ambiente di programmazione (dopo averlo installato)[9] è necessario collegare la scheda a una porta USB del computer e verificare da *Gestione dispositivi* (in *Pannello di controllo*) che la scheda sia stata correttamente riconosciuta e associata a una porta seriale COM (Fig. 27).

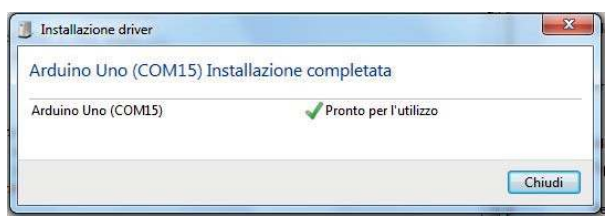


Fig. 27
La scheda Arduino riconosciuta dal sistema con il relativo numero di porta COM.

[8] Sono disponibili anche le versioni per MAC e Linux.

[9] Download da http://*arduino.cc/en/Main/Software*.

Aprendo l'ambiente di programmazione Arduino è opportuno controllare che la scheda Arduino in uso sia inserita negli **Strumenti** del software, e che la **Porta** di collegamento sia la stessa indicata nel **Pannello di controllo**, come illustrato in Fig. 28.

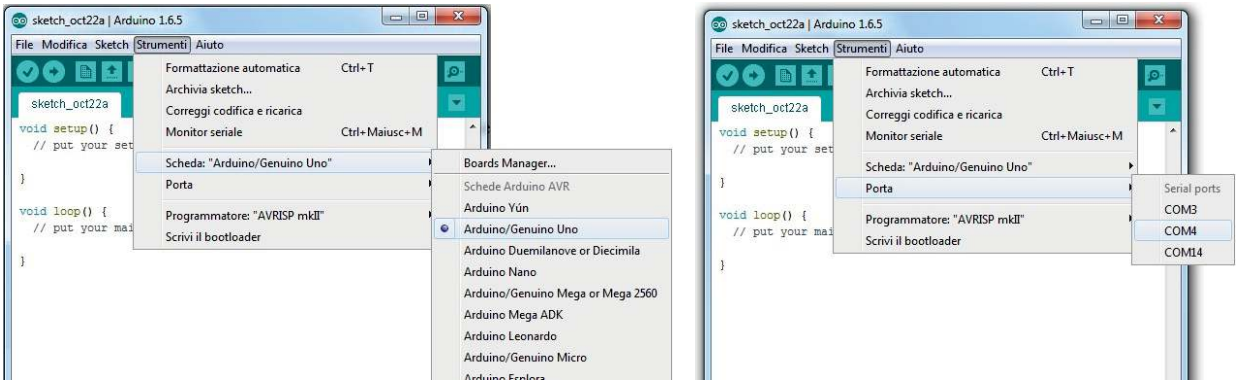


Fig. 28
Settaggio della scheda e della COM nel software di programmazione Arduino.

In genere è comodo partire da uno degli esempi già presenti e procedere poi alla modifica del programma per adattarlo alle proprie esigenze.
L'esempio che può fare al caso nostro è il programma *Blink* del menu *01.Basics* (vedi Fig. 29).

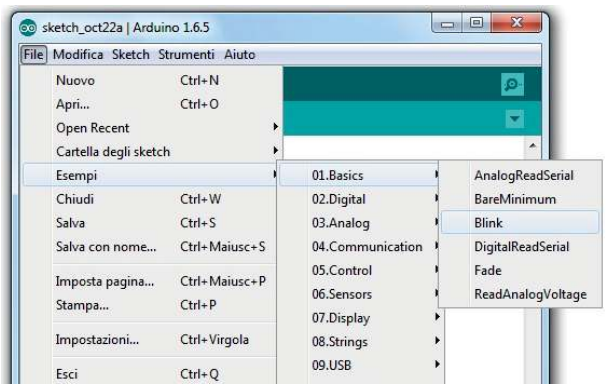


Fig. 29
Il menu esempi fornisce molti programmi da cui partire.

L'**ambiente IDE** di Arduino contiene le funzioni essenziali e presenta alcune icone che rimandano a tutte le azioni eseguibili.
In Fig. 30 vengono presentate le funzioni dei diversi pulsanti e zone dell'IDE.

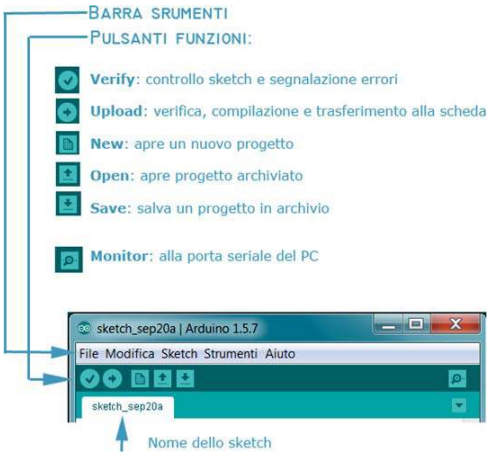
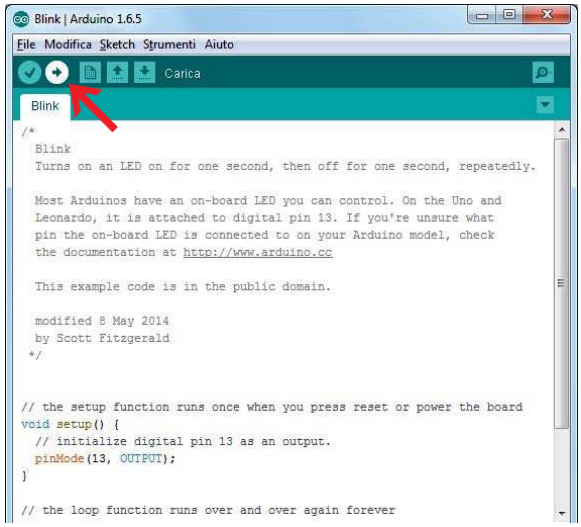


Fig. 30
Strumenti e funzioni nella parte alta dell'IDE.

Dopo aver caricato il programma (o averlo scritto personalmente), clic sul pulsante *Carica* (*Upload*) per trasferirlo alla board e avviarne l'esecuzione.

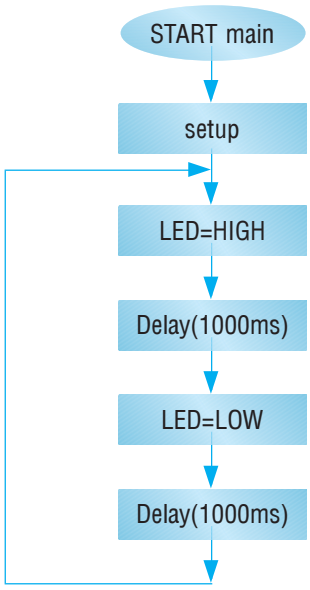
Fig. 31
Immagine presa dal software.



Analizziamo adesso il programma *Blink*:
Il programma accende il LED montato sulla board per un secondo e lo spegne per un secondo, e così via all'infinito.

Vediamo il codice:

Programma Blink




```
int LED = 13;

void setup() {
   pinMode(LED, OUTPUT);

}void loop() {
   digitalWrite(LED, HIGH);
   delay(1000);
   digitalWrite(LED, LOW);
   delay(1000);
}
```

Fig. 32
Diagramma di flusso di lavoro e codice di *Blink*.

La prima linea

```
int LED = 13;
```

dichiara la variabile LED in cui viene caricato il valore 13 che corrisponde al pin del processore al quale, sulla board Arduino, è effettivamente connesso un LED.

Seguono le due funzioni base sempre presenti in un progetto Arduino:
la funzione *setup()* e la funzione *loop()*.

- La funzione *setup()* è una particolare struttura che viene eseguita solo una volta, al reset della scheda. È tipica dei sistemi con microcontrollore per il controllo di numerosi dispostivi. Quando si accende il sistema, si devono eseguire alcune inizializzazioni. Per esempio, accendendo un distributore di bevande, bisognerà impostare preliminarmente il display, definire ingressi e uscite del sistema (riscaldatori, valvole, ecc) e dare loro il valore iniziale (normalmente tutto spento tranne il display). Queste operazioni servono solo all'avvio della macchina e non vanno più ripetute durante il normale funzionamento.

Nel caso specifico la funzione *setup()* esegue solo un'istruzione:

```
pinMode(LED, OUTPUT);
```

tale istruzione serve per "settare" il pin 13 (il valore di LED) come uscita (OUTPUT nel programma). La funzione *pinMode* imposta la modalità di funzionamento dei pin (che possono essere uscite o ingressi, non le due cose contemporaneamente). Se volessimo collegare un LED al pin 10 dovremmo mettere nel setup:

```
pinMode(10, OUTPUT);
```

In alternativa, si può inizializzare LED a 10 (`int LED = 10`).

- La funzione *loop()* comprende una parte di programma la cui esecuzione viene ripetuta continuamente. Una volta che il programma ha terminato l'ultima istruzione, riparte da capo ed esegue nuovamente le varie istruzioni di seguito, all'infinito. Questo tipo di struttura è inconsueto per la programmazione di PC. In informatica, i cicli infiniti sono considerati un errore da evitare, ma nel nostro caso il programma gestisce una macchina reale (come abbiamo già detto, un *firmware*) e l'unico evento che interrompe il funzionamento del programma è lo spegnimento di Arduino.

```
void loop() {
   digitalWrite(LED, HIGH);
   delay(1000);
   digitalWrite(LED, LOW);
   delay(1000);
}
```

La funzione loop() è molto semplice:

```
digitalWrite(LED, HIGH);
```

scrive sul pin 13 un livello alto di tensione, quindi accende il LED.

- L'istruzione `delay(1000)` attende un secondo (la funzione è subito disponibile, non è necessario includere librerie o altro), quindi il valore tra parentesi indica il ritardo in millisecondi prima che inizi l'istruzione successiva.
- La successiva istruzione `digitalWrite(LED, LOW)`, spegne il LED e dopo altri 1000 ms si ricomincia da capo.

Il codice completo è riportato nella precedente Fig. 32.

Alcune osservazioni:

- Il programma viene salvato con estensione *.ino*. In realtà è un normale file di testo. Quando si salva, il programma crea in automatico una cartella con lo stesso nome del file, separando i vari progetti in diverse cartelle.[10]
- Ogni programma per Arduino deve contenere almeno due funzioni: *setup()* e *loop()*. La prima serve per impostare il funzionamento dell'hardware del micro (input, output, comunicazioni seriali, ecc.); la seconda, invece, contiene il codice che verrà eseguito ciclicamente (da cui il nome *loop*). Non è quindi necessario (come in altri micro, per esempio *PIC*) inserire un ciclo infinito nella funzione principale (*main*).
- Per poter usare le linee digitali è necessario definire se sono input o output. A questo scopo si usa la funzione interna:

```
pinMode(numero Linea, INPUT o OUTPUT).
```

6 Il primo programma con Scratch

Vediamo un esempio di programmazione con Arduino usando *Scratch*.

Scratch è un ambiente di programmazione visuale per computer. Poiché Arduino è di fatto un piccolo computer, esiste una versione di *Scratch* realizzata appositamente per Arduino[11].


S4A

È innanzitutto necessario scaricare la versione per Arduino, che si chiama **Scratch for Arduino** e ha la sigla *S4A*.[12] Si può trovare sul sito *http://s4a.cat/*.
Una volta scaricato e installato il programma S4A (Fig. 33), è necessario installare sulla scheda un *firmware*[13] speciale per Arduino, che consente la comunicazione con *Scratch*. In *S4A*, infatti, il programma che scriviamo non viene semplicemente mandato ad Arduino, ma comunica continuamente con la scheda anche durante l'esecuzione del programma. Quindi dall'interfaccia di *Scratch* potremo vedere in tempo reale i valori degli ingressi e delle uscite della scheda. Si tratta ovviamente di un programma che va scaricato su Arduino usando l'ambiente di programmazione originale.

Fig. 33
Sequenza da seguire per l'installazione di S4a.

[10] Fa in automatico ciò che dovrebbe fare ogni buon programmatore: separa i diversi progetti in diverse cartelle.

[11] In realtà è una versione di *Scratch* adattata per Arduino e, come si vede poco oltre nel testo, richiede alcune modifiche iniziali.

[12] *S4A: Scratch four Arduino:* in queste sigle si gioca sul fatto che in lingua inglese *four* o *for* hanno la stessa pronuncia o quasi (come *2b or not 2b: to be or not to be*, la frase iniziale del famoso monologo di Amleto di W. Shakespeare).

[13] *Firmware:* è un tipo di software che determina il funzionamento di un dispositivo (ad es. uno smartphone, un elettrodomestico, ecc.). Normalmente viene caricato dall'azienda che lo produce, da cui il nome composto da *firm:* azienda (anche fisso rigido perché non viene generalmente modificato) e software (programma).