La piattaforma hardware e software Arduino: parte I

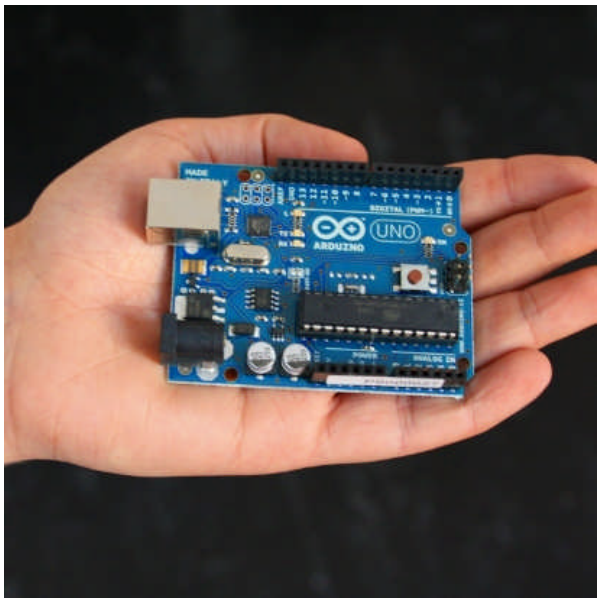
Corso di autoapprendimento

Prof. Angelo Monfroglio

(tempo di apprendimento previsto circa 2-3 ore)

Introduzione

Questa è la prima parte di un corso di base sulla piattaforma open source hardware e software Arduino. Arduino è stato creato da Olivetti nel 2005, è progettato e costruito intermente ad Ivrea (stabilimento di Scarmagno, quello dei famosi PC Olivetti) , e ha conquistato un successo mondiale (100000 schede in tutti i paesi, Corea compresa). La piattaforma è adottata, fra gli altri, dai laboratori di robotica del MIT e della Carnegie Mellon University, i due principali centri di ricerca mondiali in questo settore. E' adatta a molti impieghi, oltre che alla robotica: design, arte, programmazione evoluta. Arduino è presente all'Omar dal 2010 (versioni Arduino Uno e Arduino Mega). Questo corso è diviso in 4 parti, ognuna prevista per circa 3 ore, teoriche e pratiche, di insegnamento e autoapprendimento. La base è Arduino Uno, l'ultimo nato, ma,



Arduino Uno

con poche varianti, si adatta a tutte le versioni.


Il team Arduino: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis.

Open Source significa che viene fornita tutta la documentazione e gli schemi hardware e software, e l'utente può liberamente attuare modifiche e personalizzazioni, riconoscendo i crediti ad Arduino. Così abbiamo fatto nei laboratori di Elettronica, Robotica, Telecomunicazioni e Progettazione all'Omar, che è l'unica scuola novarese dove è presente la piattaforma. Naturalmente, l'invito è ad iscriversi agli indirizzi di Elettronica o Informatica (che sono solo all'Omar).

La scheda Arduino è unanimemente riconosciuta come più potente, più facile da usare e più economica di tutte le concorrenti, come Basic Stamp, BasicX, ecc.

L'ambiente di sviluppo (IDE), ovviamente gratuito, si chiama Sketch ed è scritto in Java ; comprende anche un editor con il syntax highlighting, il controllo automatico delle parentesi e l'indentazione automatica. Il linguaggio di programmazione è basato sul linguaggio C e C++, di cui adotta la sintassi, ma usa una libreria chiamata Wiring, che semplifica enormemente la programmazione, specialmente per l'input output verso sensori, motori, ecc. Il tutto mantenendo un'ottimizzazione delle risorse pari all'assembler, che, così, non è necessario. Inoltre si tratta di un C molto facile da usare. Ogni programma è costituito da due parti

Setup() inizializzazioni

Loop() funzione eseguita fino allo spegnimento della scheda

Arduino adotta i microcontrollori Atmel AVRmega. Arduino Uno ha un'interfaccia USB (del tipo 'stampante') per il collegamento e la programmazione da PC. E' anche possibile collegare un programmatore esterno per AVR, attraverso l'interfaccia e il connettore ISP. Tuttavia, per la presenza di un bootloader precaricato, il programmatore esterno non è indispensabile. Inoltre, si può usare il linguaggio di programmazione 'Processing', sviluppato al MIT e basato su Java. Ovviamente, 'Processing' da solo non è Arduino. Arduino si compone di 3 elementi:

-una scheda elettronica

-un ambiente di programmazione (ambiente di sviluppo sketch, linguaggio Arduino, Wiring ed eventualmente Processing)

-la comunità degli utenti (a Novara il gruppo all'Omar).

Arduino, una volta programmato, può funzionare autonomamente, alimentato da una batteria, oppure collegato a un PC e alimentato dalla USB.

Le caratteristiche della scheda Arduino uno

La scheda Arduino Uno, ultima nata, nel 2010, ha le seguenti caratteristiche:

Microcontrollore	ATmega328
Tensione di lavoro	5V
Tensione di ingresso raccomandata	7-12 V
Ingressi-Uscite	14 (di cui 6 possono fornire il comando PWM di motori)
Ingressi analogici	6
Memoria Flash	32 KB
SRAM	2 KB
EEPROM	1 KB
Clock	16 MHz

Il linguaggio base di Arduino, basato sulla sintassi del linguaggio C

Strutture generali

- setup()
- loop()

Strutture di controllo

- if
- if .. else
- for
- switch case
- while
- do .. while
- break
- continue
- return
- goto

Segni di interpunzione usati

- ;

{ }

//

/*/

Direttive

#define

#include

Tipi di dati

void

boolean

char

insigne char

byte

int

insigne int

word

long	intero a 32 bit con segno
unsigned long	intero a 32 sempre positivo

float

double

string (di caratteri)

String (oggetto)

Array

Funzioni per la gestione dell'hardware

Digitali

pinMode()

digitalWrite()

digitalRead()

Analogici

analogReference()

analogRead()

analogWrite()

Avanzate

tone()

noTone()

shiftOut()

pulseIn()

Gestione del tempo

millis()

micros()

delay()

delayMicroseconds()

Funzioni matematiche

min()

max()

abs()

contrain() costringe un valore in un intervallo a, b

operazioni + - * / % cioè modulo;

map() trasforma un intero da un intervallo ad un altro

pow() Potenza

sqrt() radice quadrata

sin()

cos()

tan()

Numeri casuali

randomSeed()

random()

Operazioni sui bit e i bytes

- lowByte ()
- highByte()
- bitRead()
- bitWrite()
- bitSet()
- bitClear()
- bit()
- &(and su bit)
- |(or su bit)
- ^(xor)
- ~(not su bit)
- <<(shift a sinistra)
- >>(shift a destra)

Costanti predefinite

LOW HIGH, INPUT OUTPUT

Utilità

- sizeof() fornisce le dimensioni in byte di una variabile

Gestioni interrupt esterni

- attachInterrupt()
- noInterrupt()

Interni

- interrupts()
- noInterrupts()

Comunicazione seriale

- Serial

Come si vede, queste istruzioni, che consentono di operare a livello di singoli bit e piedini, e l'ottimizzazione che Arduino assicura per il codice macchina generato, rendono inutile il ricorso all'assembler.

confronti = = != (diverso) < > <= >=

booleani && (and) || (or) ! (not)

Si noti che == significa confronto mentre = è usato per le assegnazioni.

Per cominciare

-prendere una scheda Arduino (nel nostro caso Arduino UNO)

-scaricare il software Arduino dal sito arduino.cc

-scompattare il file zip

-installare sketch

-collegare Arduino alla USB

-al riconoscimento del nuovo hardware, installare il driver (sul sistema operativo Vista è anche necessario copiare un piccolo file dal sito, secondo le istruzioni)

-lanciare Arduino (e la festa comincia!).

Primo uso di Arduino

L'IDE è diviso in 3 zone:

-zona bottoni e menu

-zona editor

-zona di stato (messaggi di errore o successo)

Nel menu File ci sono già pronti alcuni esempi di prova, da caricare e provare senza neanche scrivere una riga di codice.

Occorre, prima di tutto, selezionare la scheda (board) nel menu Tools: nel nostro caso Arduino Uno; e la porta seriale. Per quest'ultima, andare sul Pannello di Controllo, Sistema, Gestione dispositivi, porte seriali. Si vede quale COM è stata assegnata (esempio COM10). Tornare all'IDE di Arduino e al menu Tools e selezionare Serial port.

Ora nel menu Preferences selezionare una directory di lavoro (creata precedentemente).

Il mio primo programma

Il primo programma che si insegna per un linguaggio generale è di solito quello che visualizza Ciao mondo (hello world). Con Arduino, evidentemente, questo non avrebbe senso alcuno. Invece, accenderemo (e spegneremo) un LED. La scheda Arduino ha un LED di prova al pin 13, quindi il test si può fare senza nessun componente aggiuntivo.

Scriveremo con l'editor

Esempio 1

```
      void setup(){
            pinMode(13,OUTPUT);
      }
      void loop(){
            digitalWrite(13,HIGH);
            delay(1000);
            digitalWrite(13,LOW);
            delay(1000);
      }
```

-salvare con un nome

-compilare, premendo il bottone compila in alto a sinista (triangolino in un cerchio)
-se non ci sono errori, nel menu File selezionare Upload to I/O Board per trasferire l'eseguibile nella scheda.
-verificare il funzionamento.
-Fine: facile ?

Ora, come esercitazione, provare a comprendere, scrivere ed eseguire il seguente programma un po' più complicato.

Esempio 2

```
const int ledPin =  13;      // il numero del piedino corrispondente al LED

int ledState = LOW;

long previousMillis = 0;        // memorizza il tempo dell'ultimo
aggiornamento

// long perchè il tempo è in millisecondi


long interval = 1000;           // interval at which to blink (milliseconds)

void setup() {

  pinMode(ledPin, OUTPUT); // metto il piedino in Output
}

void loop()
{

  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis > interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;

    // se il LED è On mettilo OFF e viceversa:
    if (ledState == LOW)
      ledState = HIGH;
    else
      ledState = LOW;
```

```
    digitalWrite(ledPin, ledState);
  }
}
```

Come avrete visto, i programmi di Arduino, la cui sintassi è quella del linguaggio C, hanno i blocchi che sono delimitati dalle parentesi graffe aperta { (si ottiene col tasto Alt e 123) e chiusa } (alt 125). Le istruzioni sono terminate dal punto e virgola e le funzioni sono sempre seguite da una tonda aperta e poi chiusa.

Errori comuni
Un errore frequente è l'errata scelta della COM. In ogni caso, il testo rosso nell'area messaggi dell'IDE riassume gli errori.

Un programma che usa un'uscita analogica

Ora vediamo un programma che usa un valore analogico da mandare al pin 9 e a un LED. Occorre procurarsi un LED e un resistore di valore appropriato (e un breadboard). Lo schema di collegamento è elementare.

Esempio 3

```
// varia la luce del LED
int value = 0
int ledpin = 9

void setup()
{
// nulla da inizializzare
}
void loop()
{
        for(value = 0; value <= 255; value+=5)          // cresce
        {
        analogWrite(ledpin,value);
        delay(30);
        }
        for(value = 255; value >= 0; value-=5)          // diminuisce
        {
        analogWrite(ledpin,value);
        delay(30);
        }
}
```

Si noti, per chi non è abituato al linguaggio C, che
value+=5 è la forma sintetica per
value = value + 5
si può anche scrivere così, se si preferisce.

Esempio 4

Ora controlleremo l'accensione di un LED con un bottone

```
int  ledPin = 13;
int  inputPin = 2;
```

```
int val = 0;

void setup(){
	pinMode(ledPin,OUTPUT);
	pinMode(inputPin,INPUT);
}

void loop(){
	val = digitalRead(inputPin);
	if(val == HIGH){
	digitalWrite(ledPin,LOW);
	}else{
	digitalWrite(ledPin,HIGH);
	}
}
```

Esempio5

Infine, scriviamo un programma che, a seconda del valore letto sul pin 2, lampeggia o varia la luminosità di un LED collegato al pin 9.

```
int ledPin = 9;
int input Pin = 2
int val = 0;
intfadeval = 0;

void setup(){
	pinMode(ledPin,OUTPUT);
	pinMode(inputPin,INPUT);
}

void loop(){
	val = digitalRead(input Pin);
	if(val == HIGH)  {
		digitalWrite(ledPin, LOW);
		delay(50);
		digitalWrite(ledPin,HIGH);
		delay(50);
	}
	else{
	for(fadeval = 0;fadeval <= 255;fadeval +=5){
		analogWrite(ledPin,fadeval);
		delay(10);
	}
	for(fadeval = 255;fadeval = 255;fadeval >= 0;fadeval -=5){
		analogWrite(ledPin,fadeval);
		delay(10);
	}
}
```

Fine parte I

Buon lavoro

Bibliografia

-Banzi M., Getting started with Arduino, Make Books, 2009

-Gadre V. Dhananjay, Programming and customizing the AVR Microcontroller, McGraw-Hill, New York, 2000

-Tod E. Kurt, Bionic Arduino, Introduction to Microcontrollers with Arduino, 2007