ISTITUTO TECNICO INDUSTRIALE STATALE
G.GALILEI DI SAN SECONDO

SISTEMI EMBEDDED
Il caso Arduino

"se ascolto dimentico, se vedo ricordo, se faccio capisco"

INDICE

- Installazione e specifiche dell'IDE
- Funzioni principali e le librerie
- Creare una libreria
- I/O digitale e analogico
- Basi di programmazione in Processing

IDE ARDUINO

Download the Arduino Software

The open-source Arduino environment makes it easy to write code and upload it to the i/o board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing, avr-gcc, and other open source software.

THE Arduino SOFTWARE IS PROVIDED TO YOU "AS IS," AND WE MAKE NO EXPRESS OR IMPLIED WARRANTIES WHATSOEVER WITH RESPECT TO ITS FUNCTIONALITY, OPERABILITY, OR USE, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR INFRINGEMENT. WE EXPRESSLY DISCLAIM ANY LIABILITY WHATSOEVER FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR SPECIAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST REVENUES, LOST PROFITS, LOSSES RESULTING FROM BUSINESS INTERRUPTION OR LOSS OF DATA, REGARDLESS OF THE FORM OF ACTION OR LEGAL THEORY UNDER WHICH THE LIABILITY MAY BE ASSERTED, EVEN IF ADVISED OF THE POSSIBILITY OR LIKELIHOOD OF SUCH DAMAGES.

By downloading the software from this page, you agree to the specified terms.

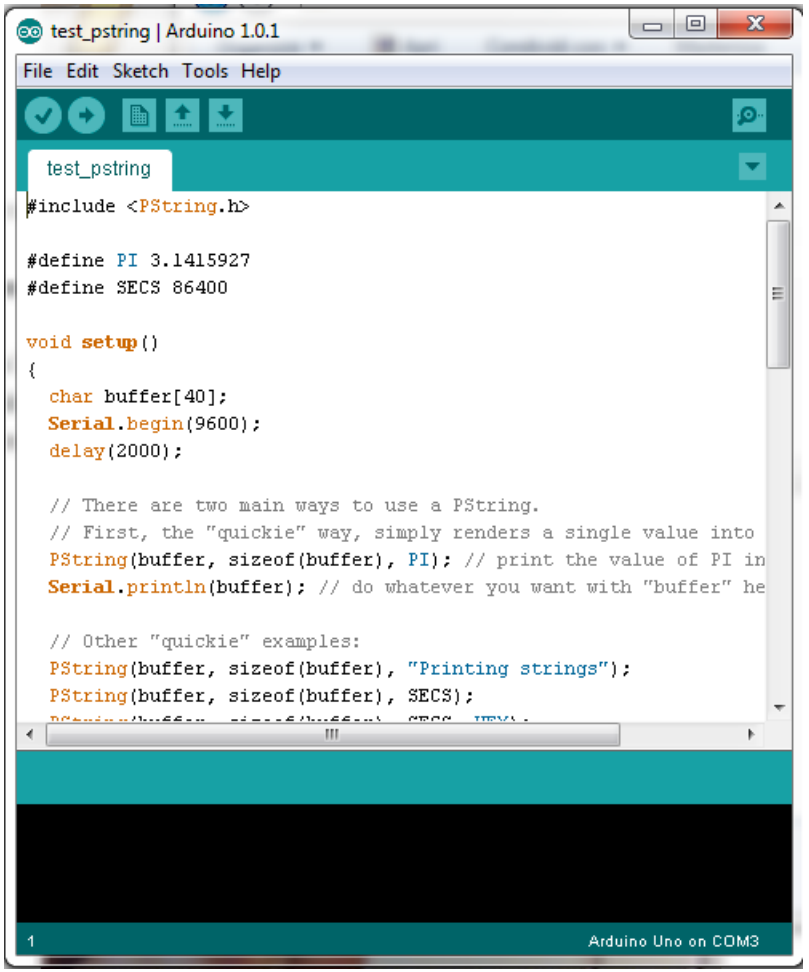
Download

Arduino 1.0.3 (release notes), hosted by Google Code:

- Windows
- Mac OS X

Next steps

Getting Started
Reference
Environment
Examples



http://arduino.cc/en/Main/Software

LE LIBRERIE PRINCIPALI

EEPROM - lettura e scrittura su memoria “permanente”
Ethernet - per connettere Arduino su una rete attraverso l’apposita shield
Firmata - per comunicare con diverse applicazioni sul computer usando lo standard seriale
LiquidCrystal - per controllare display LCD
SD - per leggere e scrivere su schede SD
Servo - per controllare servo motori
SPI - per comunicare con dispositivi che utilizzano un Bus Serial Peripheral Interface
SoftwareSerial - per la comunicazione seriale
Stepper - per controllare motori stepper
WiFi - per collegare Arduino a internet usando la WiFi shield
Wire - Two Wire Interface (TWI/I2C) per mandare e ricevere dati su una rete di sensori o dispositivi

From http://arduino.cc/en/Reference/Libraries

LE LIBRERIE PRINCIPALI

Communication (networking e protocolli):
Messenger - per l'elaborazione di messaggi basati su testo dal computer
OneWire - dispositivi di controllo (da Dallas Semiconductor) che utilizzano il protocollo One Wire.
PS2Keyboard - Leggere caratteri da una tastiera PS2
Simple Message System - inviare messaggi tra Arduino e il computer
SSerial2Mobile - inviare messaggi di testo o e-mail utilizzando un telefono cellulare (con comandi AT attraverso seriale)
Webduino - libreria per un web server
X10 - Invio di segnali X-10 sulla linea di alimentazione AC
XBee - per comunicare con XBees in modalità API

From http://arduino.cc/en/Reference/Libraries

LE LIBRERIE PRINCIPALI

Librerie USB(Leonardo, Micro, Due, and Esplora)
Keyboard - mandare caratteri ad un computer come se fosse una tastiera
Mouse - controllare il puntatore del mouse del computer a cui si è collegati

Librerie per Arduino Due
Audio - Play file audio da scheda SD
Scheduler - Gestire blocchi multipli di task
USBHost - Comunicare con perfieriche USB, come tastiera o mouse

Libreria per Arduino Esplora
Esplora - questa libreria consente di accedere facilmente a vari sensori e attuatori montati sulla scheda ESPLORA

From http://arduino.cc/en/Reference/Libraries

CREARE UNA LIBRERIA

```
int pin = 13;

void setup()
{
  pinMode(pin, OUTPUT);
}

void loop()
{
  dot(); dot(); dot();
  dash(); dash(); dash();
  dot(); dot(); dot();
  delay(3000);
}
```

Se si prova ad eseguire questo sketch, si vedrà riprodurre il codice morse per l'SOS sul led del pin 13.

Ma dot() e dash() come sono implementate?

CREARE UNA LIBRERIA

Per creare una libreria sono necessari due file:

<*nome_libreria*>.h – il file header
<*nome_libreria*>.cpp – il file sorgente con l'implementazione del file header

Il cuore del codice HEADER

```
class Morse
{
 public:
   Morse(int pin);
   void dot();
   void dash();
 private:
   int _pin;
};
```

Il cuore del codice SORGENTE

```
Morse::Morse(int pin)
{
 pinMode(pin, OUTPUT);
 _pin = pin;
}

void Morse::dot()
{
 digitalWrite(_pin, HIGH);
 delay(250);
 digitalWrite(_pin, LOW);
 delay(250);
}
```

```
void Morse::dash()
{
 digitalWrite(_pin, HIGH);
 delay(1000);
 digitalWrite(_pin, LOW);
 delay(250);
}
```

CREARE UNA LIBRERIA

IL FILE HEADER

```
#ifndef Morse_h
#define Morse_h

#include ‘’Arduino.h’’

class Morse
{
  public:
    Morse(int pin);
    void dot();
    void dash();
  private:
    int _pin;
};

#endif
```

CREARE UNA LIBRERIA

IL FILE SORGENTE

```
#include "Arduino.h"
#include "Morse.h "

Morse::Morse(int pin)
{
  pinMode(pin, OUTPUT);
  _pin = pin;
}

void Morse::dot()
{
  digitalWrite(_pin, HIGH);
  delay(250);
  digitalWrite(_pin, LOW);
  delay(250);
}
```

```
void Morse::dash()
{
  digitalWrite(_pin, HIGH);
  delay(1000);
  digitalWrite(_pin, LOW);
  delay(250);
}
```

CREARE UNA LIBRERIA

*Una volta creati i due file bisogna inserirli in una cartella **CON LO STESSO NOME DELLA LIBRERIA** e andarla a copiare nella cartella `libraries` dove si trova l'IDE di Arduino.*

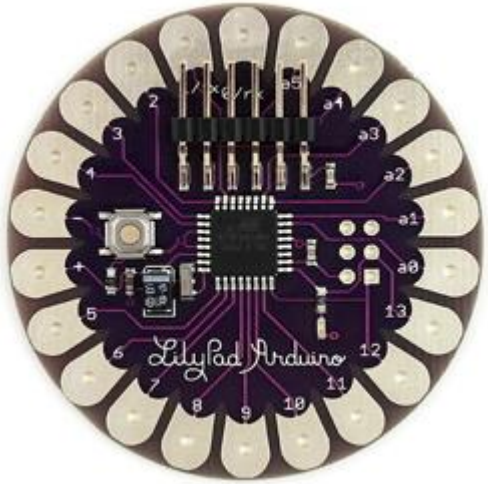
È possibile creare un ulteriore file in cui andare ad inserire le parole chiavi del programma (che poi saranno colorate in modo differente nell'IDE per migliorare la lettura del codice. KEYWORD1 per una colorazione arancio, mentre KEYWORD2 per una colorazione blu)

keywords.txt va salvato nella stessa cartella della libreria. Potrebbe essere scritto come segue:

```
Morse   KEYWORD1
dash    KEYWORD2
dot     KEYWORD2
```

I/O DIGITALE E ANALOGICO

Quasi tutte le schede Arduino hanno 14 pin digitali: sono pin che puoi utilizzare sia in modo INPUT (ingresso), per acquisire il valore di un pulsante ad esempio o di un interruttore a leva, sia in modo OUTPUT (uscita) per inviare segnali logici ad una scheda aggiuntiva, ad un led, ecc…



Siccome si tratta di impulsi di tipo digitale il valore che puoi ricevere/inviare è il classico 0/1 o in linguaggio Arduino LOW/HIGH.

I/O DIGITALE

Per definire come utilizzeremo un dato pin si usa il seguente comando:
`pinMode(pin, modalità):`
pin *= indica il pin che vogliamo settare con la modalità definita nel secondo parametro,*
modalità = *INPUT o OUTPUT.*

Per inviare un segnale al pin si usa il comando:
`digitalWrite(pin,level):`
pin *= indica il pin a cui vogliamo inviare il segnale definito nel secondo parametro,*
level *= LOW o HIGHT (0 o 1).*

Per leggere un valore digitale, magari quello di un interruttore collegato a massa puoi si utilizza il comando pinMode con modalità INPUT, ed il comando
`digitalRead(pin):`
pin *= indica il numero del pin di cui vogliamo conoscere il valore.*

I/O ANALOGICO

Bisogna ricordarsi sempre di settare la modalità del pin prima di eseguire l'invio dei comandi, in questo esempio si utilizza il pin 11 scrivendo:
`pinMode(11,OUTPUT)` indico ad Arduino di utilizzarlo in modo OUTPUT.

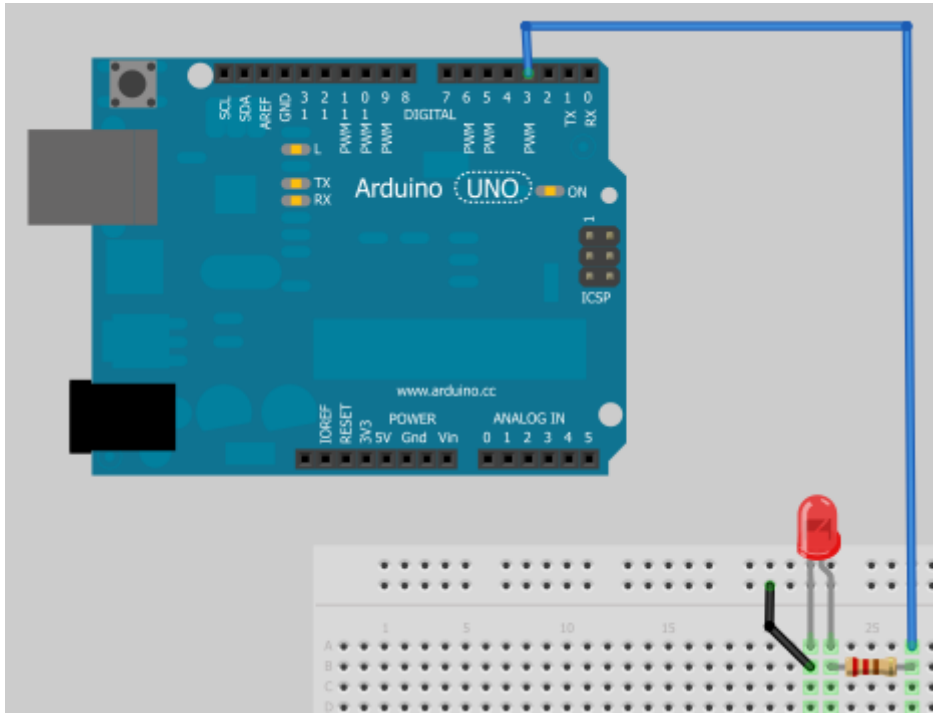
Per esempio posso inviare il mio segnale con duty-cycle al 50%:
`analogWrite(11,127).`
Il valore 127 è esattamente a metà della mia scala (0 – 255) quindi il duty-cycle generato sarà del 50%.

Per i pin analogici non è necessario settare la modalità con cui vengono usati perché sono SOLAMENTE di INPUT.
Per leggere il valore presente sul pin 5, ad esempio, si utilizza il comando:
`analogRead(5)` il valore rilevato può variare da 0 a 1023, all'interno del range indicato ogni variazione del segnale sarà indicata con una variazione di tale valore.

ALCUNI ESEMPI CON ARDUINO

Facciamo lampeggiare un led

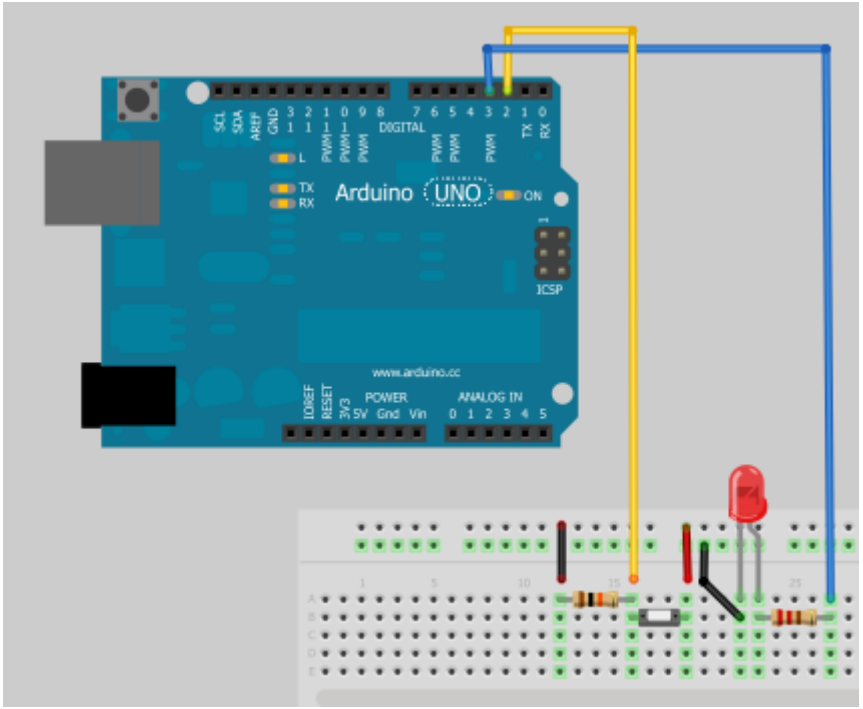
-> Blink



ALCUNI ESEMPI CON ARDUINO

Controlliamo il Led con un bottone

-> Button_Led



Esempio 1: accensione led appena premuto il pulsante

Esempio 2: accendo led quando premo pulsante e spengo quando ripremo

Esempio 3: come il 2 ma con l'antirimbalzo

Esempio 4: led lampeggia se premo e se ripremo si spegne

ALCUNI ESEMPI CON ARDUINO

ESERCIZIO

Simulazione comando delle lampade di una scala

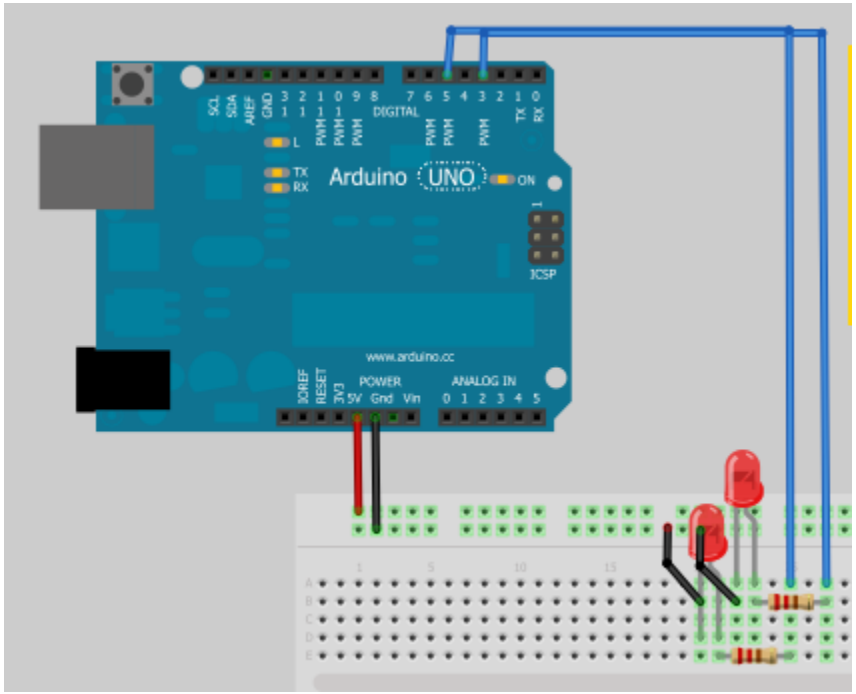
Pulsante 1:
(salita) start,
accensione del led 1,
dopo 2 secondi accensione led 2,
dopo 2 secondi spegnimento led 1,
dopo 2 secondi spegnimento led 2,
end.

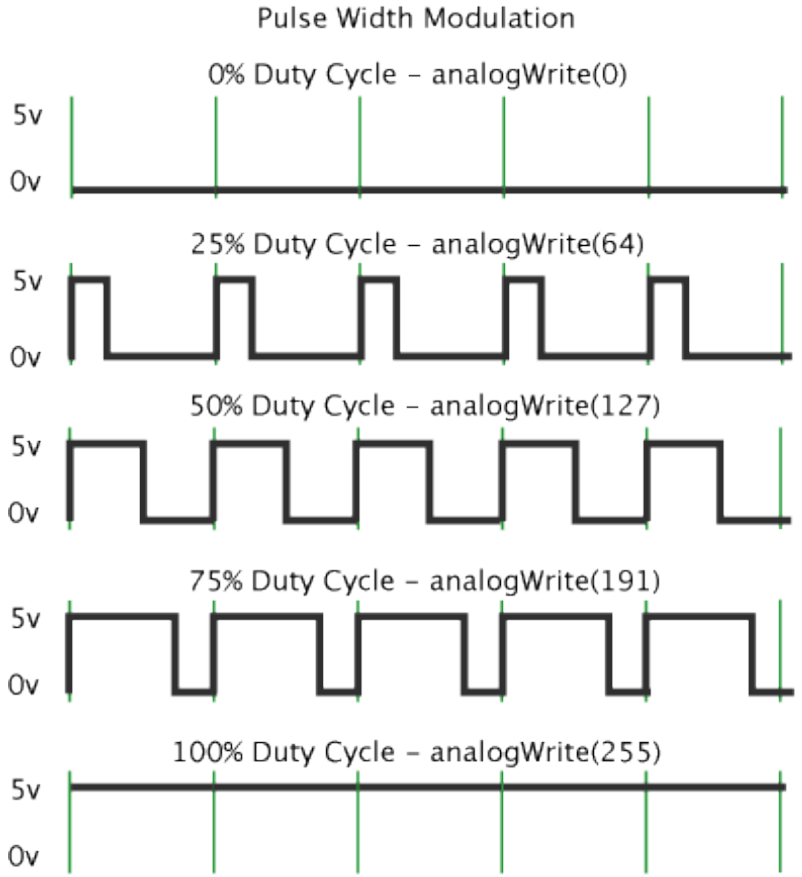
Pulsante 2:
(discesa) start,
accensione del led 2,
dopo 2 secondi accensione led 1,
dopo 2 secondi spegnimento led 2,
dopo 2 secondi spegnimento led 1,
end.

ALCUNI ESEMPI CON ARDUINO

PWM

-> Fade

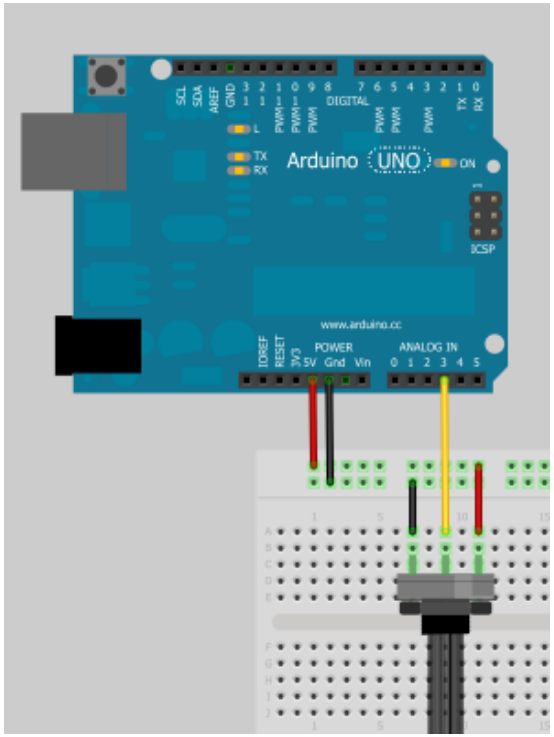




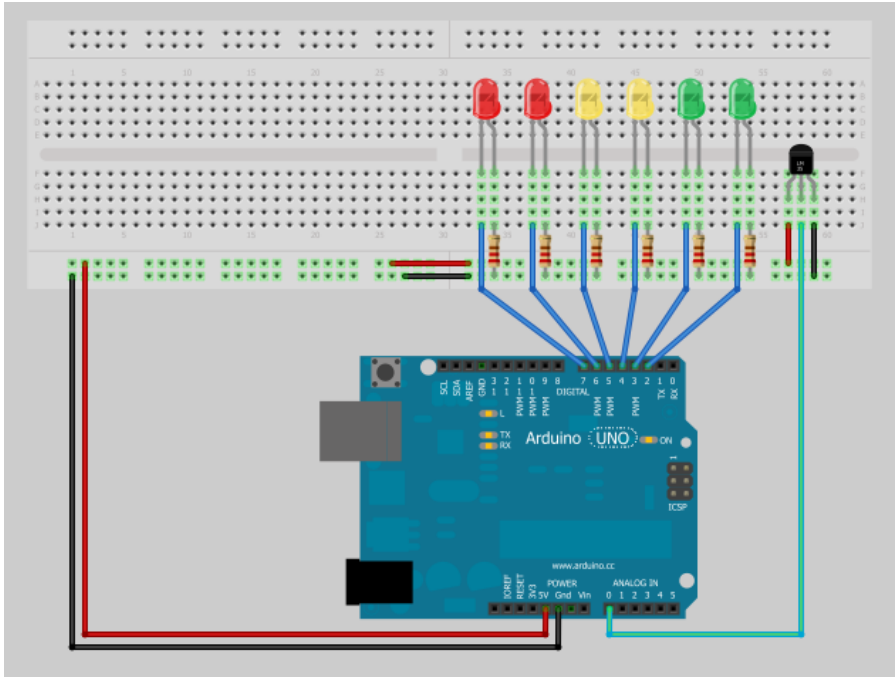
ALCUNI ESEMPI CON ARDUINO

Identifichiamo le variazioni di stato di un sensore analogico

-> AnalogPot



-> Temperature



ALCUNI ESEMPI CON ARDUINO

ESERCIZIO

Simulazione di un impianto di riscaldamento

Riutilizzare l'esempio del termometro ed aggiungere un potenziometro per impostare la temperatura. Simulare con un led l'accensione/spegnimento del termostato e con un altro led (utilizzando l'esempio Fade) mostrare con l'intensità luminosa, quanto vale il valore del potenziometro.

Se il valore letto dal potenziometro è minore della temperatura della stanza accendere il termostato, altrimenti lasciarlo spento

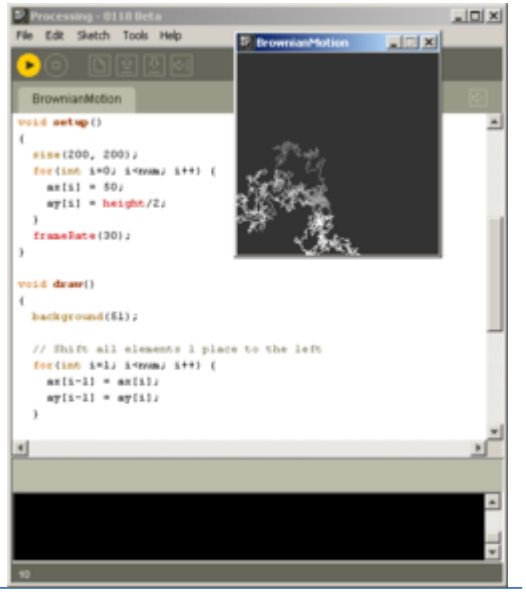
PROCESSING

Il linguaggio Processing è basato su Java. L'ambiente di sviluppo di Arduino è scritto in Processing. Processing deve essere installato dopo averlo scaricato dal sito di Processing. La programmazione è più semplice del linguaggio Java usuale. E' particolarmente adatto per programmi di grafica ed è stato usato anche in ambiente artistico. Per familiarizzarsi conviene caricare un esempio già pronto.

Caricare dagli esempi
→ sezione Motion
→ Bounce.

Premere Run.

La finestra che compare è un'applet Java.



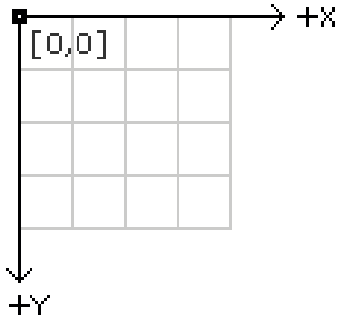
PROCESSING

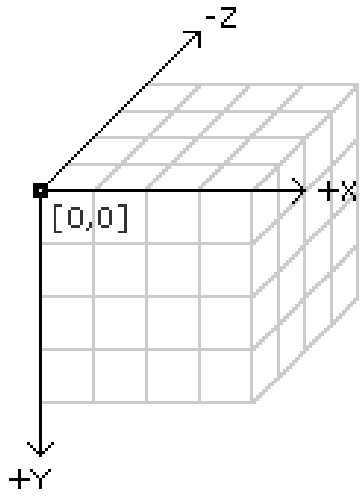
Il bytecode prodotto da Processing è un programma di tipo speciale, detto applet, che può essere inserito all'interno di una pagina web con un'apposito marcatore `<applet>`.
In questo modo, il programma una volta compilato è **inserito all'interno di una pagina web** e, al momento della visualizzazione della pagina, l'applet è eseguito.
L'unica cosa necessaria per la visualizzazione degli applet è che sul calcolatore sia installato l'interprete Java, (JRE o JVM).
Attenzione, Processing crea l'applet solo se si esegue il comando "export".
In tal caso, esso genera il bytecode dell'applet ed una semplice pagina web di nome index.html contenente un riferimento all'applet mediante il marcatore omonimo.
Processing può anche creare un'applicazione se si esegue il comando "export application". In tal caso, esso genera oltre al bytecode dell'applet un file eseguibile che avvia a sua volta l'esecuzione dell'applet. L'applicazione può essere generata per i sistemi Windows, Mac e Linux.

PROCESSING – Le basi

Per disegnare qualche cosa dentro questa finestra è necessario prima di tutto comprendere il concetto di coordinate. L'asse delle X scorre in orizzontale mentre l'asse delle Y scorre in verticale.
L'angolo in alto a sinistra della nostra finestra ha le coordinate (0, 0). L'angolo in basso a destra ha coordinate (X_Max, Y_Max). L'angolo in alto a destra ha coordinate (X_Max, 0) mentre l'angolo in basso a sinistra ha coordinate (0, Y_Max).

```
//Imposto le dimensioni della
finestra di lavoro
size(320, 240);

//Imposto il colore dello sfondo
background(255);
```





PROCESSING – Iniziare a disegnare

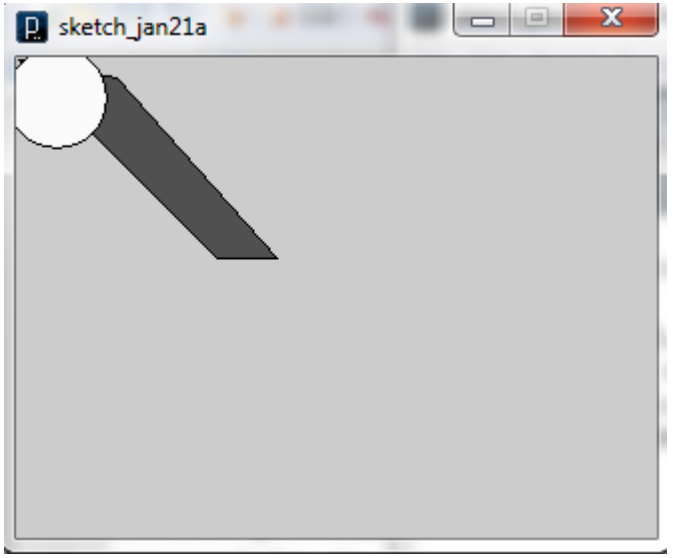
Processing mette a disposizioni alcune funzioni per disegnare le figure fondamentali a schermo.

```
//UN PUNTO
point(X, Y);

//UNA LINEA – coordinata d'inizio e di fine
line(X1, Y1, X2, Y2);

//UN TRIANGOLO – coordinate dei vertici
triangle(X1, Y1, X2, Y2, X3, Y3);

//UN QUADRATO – coordinate dei vertici
quad(X1, Y1, X2, Y2, X3, Y3, X4, Y4);

//UN RETTANGOLO – coordinate dei vertici e dimensioni
rect(X, Y, width, height);

//UN ELLISSE – coordinata del centro di simmetria e dimensioni degli assi
ellipse (X, Y, width, height);
```

PROCESSING – Iniziare a disegnare

Se voglio modificare il colore o la forma dei miei disegni utilizzo queste funzioni:

```
//questa istruzione deve precedere il comando per la generazione della
forma geometrica.
fill(int r, int g, int b, int alpha);

//ESEMPIO
size(320, 240);
//Creiamo un quadrilatero
fill(80,80,80);
quad(1, 1, 50, 10, 130, 100, 100, 100);
//mettiamo sopra al quadrilatero un cerchio
fill(250,250,250);
ellipse(20, 20, 50, 50);
```



PROCESSING – Iniziare a disegnare

Il cerchio che abbiamo creato nel precedente Sketch ha il bordo nero, questo colore è di default e può essere cambiato mettendo, sempre prima dell'istruzione che genera la forma geometrica , l'istruzione:

```
stroke(255,255,255);
```

Un modo più corretto per fare scomparire il bordo da una forma geometrica è l'apposito comando

```
noStroke(0);
```

così come possiamo fare scomparire il riempimento con

```
noFill(0);
```

Inoltre possiamo anche decidere lo spessore del bordo delle forme geometriche (e delle linee) attraverso la seguente istruzione dove il valore tra parentesi indica lo spessore del bordo espresso in pixel.

```
strokeWeight(4);
```

PROCESSING – Le animazioni

Un'animazione è composta da **fotogrammi** (immagini fisse) successivi. L'esecuzione dei fotogrammi ad una velocità sufficiente genera un effetto di animazione nel nostro cervello. Processing è un software che ci permette di disegnare attraverso le istruzioni grafiche e ci permette anche di **disegnare in movimento**. Riconfiguriamo ora gli Sketch realizzati precedentemente dandoci un nuovo obiettivo. Produrre attraverso un'animazione dei punti successivi che compaiono progressivamente. Per ottenere questo nuovo effetto dobbiamo utilizzare le istruzioni che permettono di gestire le funzioni:

```
// configuro i parametri
// di esecuzione
void setup () {
     size (320,240);
     background (255);
}

int intervallo = 0;
```

```
// definisco la funzione
// dentro la funzione
// metto le istruzioni ricorsive
void draw (){
          //Velocità dell'animazione,
          1fps. Di default è 60
          frameRate (1);
          point(intervallo, 120);
          intervallo = intervallo + 10;
}
```

PROCESSING – Il testo

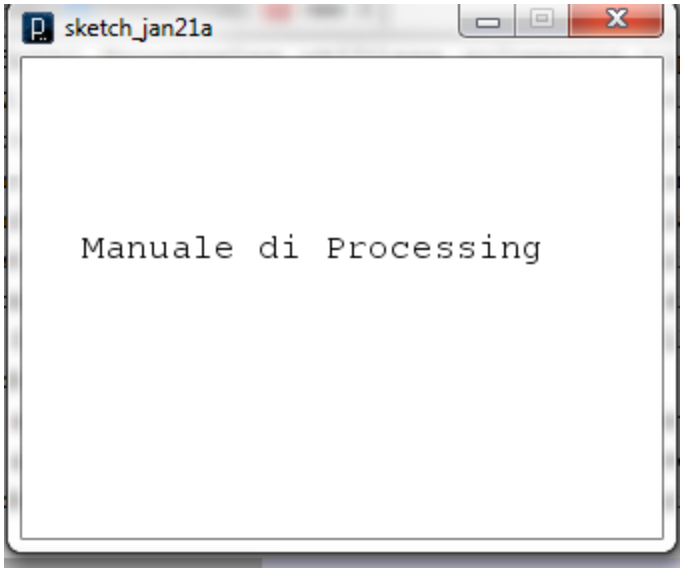
Processing utilizza i font che sono stati precedentemente dichiarati. A differenza dei normali programmi di scrittura, dove troviamo il menù a tendina con tutti i font disponibili, Processing, essendo un ambiente di sviluppo permette di utilizzare qualunque font che sia stato precedentemente impostato.

Infatti Processing utilizza solamente tipi di carattere in **formato VLW**. Quando all'interno di uno Sketch sviluppato in Processing intendiamo utilizzare un font dobbiamo prima di tutto importare il font stesso attraverso il "`Menù Tool`" comando "`Create Font`".

Dando questo comando si apre una finestra nella quale compare la lista dei caratteri installati nel proprio computer. Si seleziona il carattere e la relativa dimensione che si vuole importare e cliccando su OK verrà automaticamente creato un file che poi potrà essere richiamato all'interno dello Sketch. Se ad esempio vogliamo utilizzare il font `Courier` nella dimensione 12 all'interno del nostro Sketch dopo avere generato il file dovremo richiamare un file di questo nome: "CourierNewPSMT-20.vlw".

PROCESSING – Il testo

```
size (320, 240);
background (255);
// Inizializzo la variabile
PFont carattere;
// Carico il font nello Sketch
carattere = loadFont("CourierNewPSMT-20.vlw");
// Stabilisco il font che sto per usare
textFont(carattere);
textSize(18);
fill(0);
// Scrivo il testo
text("Manuale di Processing", 30, 100);
```



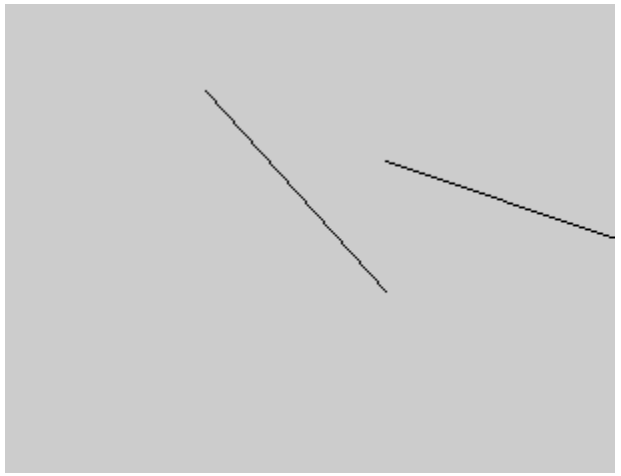
PROCESSING – Traslazione

Attraverso l'istruzione `translate()` abbiamo la possibilità di traslare degli oggetti. Per esempio se voglio creare due linee traslate di un certo valore posso fare così:

```
size(320, 240);
translate(100, 25); //Quello che disegno dopo sarà spostato
                      di 100px in basso e 25px a destra
line(10, 20, 100, 120);
translate(80, 15); //Quello che disegno dopo sarà spostato
                      di 80px in basso e 15px a destra
line(20, 40, 200, 100);
```

La prima line() potrebbe essere riformulata così:

```
line(10 + 100, 20 + 0, 100 + 100, 120 + 0);
```



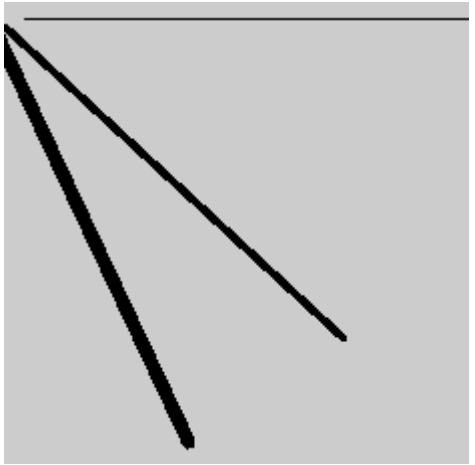
PROCESSING – Rotazione

I principi della traslazione valgono anche nel caso della rotazione. L'istruzione da utilizzare è `rotate()`. L'unità di misura della rotazione è il radiante. La corrispondenza tra i gradi e i radianti e molto semplice:

- due radianti (2 * π) sono 360 gradi
- un radiante (π) e 180 gradi
- mezzo radiante (π/2) e 90 gradi

In processing π si scrive PI.

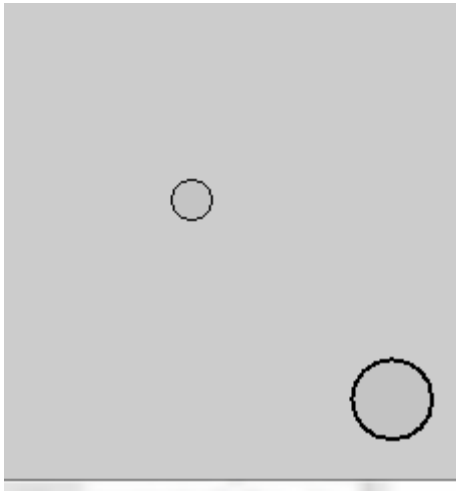
```
size (240, 240);
line(0, 0, 240, 0);
//Ruoto l'oggetto successivo
strokeWeight(4);
rotate(PI/4);
line(0, 0, 240, 0);
//Ruoto l'oggetto successivo
strokeWeight(7);
rotate(PI/8);
line(0, 0, 240, 0);
```



PROCESSING – Scalare

Per ingrandire o rimpiccolire un oggetto si utilizza l'istruzione `scale()`.

```
size (240, 240);
fill (100,20);
// Ingrandisco del doppio l'oggetto successivo
scale (2);
ellipse (100, 100, 20, 20);
// Diminuisco della metà l'oggetto successivo
scale (0.5);
ellipse (100, 100, 20, 20);
```



PROCESSING E ARDUINO

```
/* Simple Read
Read data from the serial port and change the color of a rectangle when a switch connected
to a Wiring or Arduino board is pressed and released. This example works with the Wiring /
Arduino program that follows below. */
```

```
import processing.serial.*;
String portname = "COM8"
Serial port; // Create object from Serial class
int val=100; // Data received from
           the serial port, with an initial value
```

```
void setup(){
  size(400, 400);
  colorMode(HSB, 255);
  ellipseMode(CENTER); // draw from center out
  noStroke();
  frameRate(30);
  smooth();
  // Open the port the board is connected to
  port = new Serial(this, portname, 19200);
}
```

```
void draw(){
  // If data is available,
  if (port.available() > 0) {
     val = port.read();
     // read it and store it in val
  }
  background(99);
  // Draw the shape
  fill(val,255,255); // we're in HSB mode,
                     so first value is color
  ellipse(width/2, height/2, 250,250);
```

RIFERIMENTI

ARDUINO

- Tutorial di arduino (link)
- PWM Tutorial (link)

PROCESSING

- Guida in Italiano (pdf)
- Esercizi e guida di riferimento (esercizi, guida)

RIFERIMENTI

- M. Banzi, *Getting started with Arduino*, O'Reilly, Cabridge,Beijing, 2009
- Tod E. Kurt, Bionic *Arduino, MachineProject*, 2007
- S. Monk, *30 Arduino Projects*, McGrawHill, New York,2010
- M. Schmidt, Arduino, *A Quick-Start Giude, The Pragmatic Programmers*, 2011
- C. Reas and Ben Fry, *Getting Started with Processing*, O'Reilly, 2010

- www.arduino.cc
- http://www.maffucci.it/
- www.shiffman.net/itp

RIFERIMENTI

www.mancio90.it

mirkomancin90@gmail.com