La piattaforma hardware e software Arduino: parte II

Corso di autoapprendimento

Prof. Angelo Monfroglio

Questa è la seconda parte del corso di autoapprendimento sulla piattaforma Arduino. Il tempo previsto di studio è di 2-3 ore. Un consiglio: prima di iniziare ripassare la parte I.

Se sento dimentico, se vedo ricordo, se faccio capisco (detto orientale)

If you can imagine it, you can make it

(se puoi immaginare una cosa, la puoi realizzare), Make Media, California

Massimo Banzi, creatore di Arduino, afferma che Arduino è una piattaforma per il 'physical computing', una programmazione che comprende sensori, microcontrollori e attuatori, una piattaforma robotica. Ha senso, cioè, solo in un laboratorio di elettronica. Ricordiamo che, con la riforma Gelmini, solo l'ITIS Omar, a Novara, ha laboratori di elettronica, e laboratori di Telecomunicazioni con Informatica.

Il linguaggio base di Arduino, basato sulla sintassi del linguaggio C: qualche dettaglio in più

Strutture generali

setup()

loop()

Strutture di controllo

If esempio if (val == 1) { digitalWrite (LED,HIGH);}

if .. else

for esempio for (int i = 0; i < 10; i ++) {Serial.print("ciao");}

I ++ significa i= i + 1

switch case

Esempio:

```
Switch (sensorValue) {
        case 23:
                digitalWrite(13,HIGH);
                break;
```

```
                case 46:
                        digitalWrite(12,IGH);
                        break;
                default:
                        digitalWrite(12,LOW);
                        digitalWrite(13,LOW);
}
        while                   (0-ciclo)
```

Esempio.

```
//lampeggia per tutto il tempo che il sensore è sotto il valore 512
sensorValue = analogRead(1);
while (sensorValue < 512) {
        digitalWrite(13,HIGH);
        delay(100);
        digitalWrite(13,HIGH);
        delay(100);
        sensorValue = analogRead(1);
}
```

do .. while	(1-ciclo: la condizione è valutata in fondo e il ciclo è eseguito almeno 1 volta)
break	chiude un loop e fa proseguire dall'istruzione successiva al loop
continue	all'interno di un ciclo fa saltare le istruzioni seguenti e riesegue il test del ciclo
return	
goto	

Segni di interpunzione usati

```
;
{ }
//
```

/*/

Direttive

#define

#include

Tipi di dati (variabili)

Void	vuoto, non restituisce nulla
boolean	
char	
unsigned long	da 0 a 4294967295
byte	numero fra 0 e 255
int	2 bytes, fra -32768 e 32767
unsigned int	2 bytes fra 0 e 65535
word	2 bytes
long	intero a 32 bit con segno
unsigned long	intero a 32 bit sempre positivo
float	4 bytes
double	8 bytes, Massimo 1,7976931348623157 per 10 alla 308
string (di caratteri)	
String (oggetto)	
Array	ad esempio int light [6] = {0, 20, 50, 75, 100};

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()

constrain() 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, TRUE, FALSE

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.

LED RGB (a colori)

Da qualche anno sono stati introdotti i LED a colori o RGB. Hanno 4 piedini: la massa (o catodo comune), oppure il + (o anodo comune), il piedino Rosso, il Blu e il Verde. Il sistema di colori è additivo: per somma si possono ottenere tutti i colori. Lo schema di connessione è semplice: il + a 5V (consideriamo la versione a anodo comune) della scheda Arduino, gli altri piedini del LED RGB ai piedini 9, 10, 11, usando resistori di limitazione della corrente di 220 ohm.

Esempio di programma (sketch) tratto da "Bionic Arduino" (vedi bibliografia parte I)

```
 * Code for cross-fading 3 LEDs, red, green and blue, or one tri-color LED,
using PWM
Passa lentamente dal rosso al verde al blu e di nuovo al rosso

 * The program cross-fades slowly from red to green, green to blue, and blue
to red

// Output
int redPin   = 9;   // Red LED,   connected to digital pin 9
int greenPin = 10;  // Green LED, connected to digital pin 10
int bluePin  = 11;  // Blue LED,  connected to digital pin 11

// Program variables
int redVal   = 255; // Variables to store the values to send to the pins
int greenVal = 1;   // Initial values are Red full, Green and Blue off
int blueVal  = 1;

int i = 0;     // Loop counter
int wait = 15; // 50ms (.05 second) delay; shorten for faster fades
int DEBUG = 0; // DEBUG counter; if set to 1, will write values back via
serial

void setup()
{
  pinMode(redPin,   OUTPUT);   // sets the pins as output
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin,  OUTPUT);
  if (DEBUG) {           // If we want to see the pin values for
debugging...
    Serial.begin(9600);  // ...set up the serial ouput on 0004 style
  }
}

// Main program
void loop()
{
  i += 1;      // Increment counter
  if (i < 255) // First phase of fades
  {
    redVal   -= 1; // Red down
```

```
    greenVal += 1; // Green up
    blueVal   = 1; // Blue low
  }
  else if (i < 509) // Second phase of fades
  {
    redVal    = 1; // Red low
    greenVal -= 1; // Green down
    blueVal  += 1; // Blue up
  }
  else if (i < 763) // Third phase of fades
  {
    redVal  += 1; // Red up
    greenVal = 1; // Green lo2
    blueVal -= 1; // Blue down
  }
  else // Re-set the counter, and start the fades again
  {
    i = 1;
  }

  // we do "255-redVal" instead of just "redVal" because the
  // LEDs are hooked up to +5V instead of Gnd
  analogWrite(redPin,   255 - redVal);   // Write current values to LED pins
  analogWrite(greenPin, 255 - greenVal);
  analogWrite(bluePin,  255 - blueVal);

  if (DEBUG) { // If we want to read the output
    DEBUG += 1;     // Increment the DEBUG counter
    if (DEBUG > 10) { // Print every 10 loops
      DEBUG = 1;     // Reset the counter
      Serial.print(i);       // Serial commands in 0004 style
      Serial.print("\t");    // Print a tab
      Serial.print("R:");    // Indicate that output is red value
      Serial.print(redVal);  // Print red value
      Serial.print("\t");    // Print a tab
      Serial.print("G:");    // Repeat for green and blue...
      Serial.print(greenVal);
      Serial.print("\t");
      Serial.print("B:");
      Serial.println(blueVal); // println, to end with a carriage return
    }
  }
  delay(wait); // Pause for 'wait' milliseconds before resuming the loop
}
```

Ora un programma per controllare il colore (mixer) con un potenziometro collegato al piedino 2

```
 * RGB Pot Mixer
 * -------------
 *
 * Code for making one potentiometer control 3 LEDs, red, grn and blu,
 * or one tri-color LED.
 * The program cross-fades from red to grn, grn to blu, and blu to red.
 *
 * Code assumes you have the LEDs connected in a common-anode configuration,
 * with the LED's anode connected to +5V via a resistor and the cathode
connected
```

```
 * to Arduino pins 9,10,11.
 *
 * Originally by Clay Shirky <clay.shirky@nyu.edu>
 * Modified slightly by Tod E. Kurt <tod@todbot.com>
 *
 */

// INPUT: Potentiometer should be connected to 5V and GND
int potPin = 2;  // Potentiometer output
int potVal = 0;  // Variable to store the input from the potentiometer

// OUTPUT: Use digital pins 9-11, the Pulse-width Modulation (PWM) pins
int redPin = 9;   // Red LED,   connected to digital pin 9
int grnPin = 11;  // Green LED, connected to digital pin 10
int bluPin = 10;  // Blue LED,  connected to digital pin 11

// Program variables
int redVal = 0;   // Variables to store the values to send to the pins
int grnVal = 0;
int bluVal = 0;

int DEBUG = 0;          // Set to 1 to turn on debugging output

void setup()
{
  pinMode(redPin, OUTPUT);   // sets the pins as output
  pinMode(grnPin, OUTPUT);
  pinMode(bluPin, OUTPUT);

  if (DEBUG) {           // If we want to see the pin values for
debugging...
    Serial.begin(19200);  // ...set up the serial ouput in 0004 format
  }
}

// Main program
void loop()
{
  potVal = analogRead(potPin);   // read the potentiometer value at the
input pin

  if (potVal < 341)  { // Lowest third of the potentiometer's range (0-340)
    potVal = (potVal * 3) / 4; // Normalize to 0-255

    redVal = 255 - potVal;  // Red from full to off
    grnVal = potVal;        // Green from off to full
    bluVal = 1;             // Blue off
  }
  else if (potVal < 682) { // Middle third of potentiometer's range (341-
681)
    potVal = ( (potVal-341) * 3) / 4; // Normalize to 0-255

    redVal = 1;            // Red off
    grnVal = 255 - potVal; // Green from full to off
    bluVal = potVal;       // Blue from off to full
  }
  else  { // Upper third of potentiometer"s range (682-1023)
    potVal = ( (potVal-683) * 3) / 4; // Normalize to 0-255

    redVal = potVal;       // Red from off to full
```

```
    grnVal = 1;            // Green off
    bluVal = 255 - potVal; // Blue from full to off
  }

  // "255-" is because we have common-anode LEDs, not common-cathode
  analogWrite(redPin, 255-redVal);   // Write values to LED pins
  analogWrite(grnPin, 255-grnVal);
  analogWrite(bluPin, 255-bluVal);

  if (DEBUG) { // If we want to read the output
    DEBUG += 1;      // Increment the DEBUG counter
    if (DEBUG > 100) { // Print every hundred loops
      DEBUG = 1;     // Reset the counter
      Serial.print("R:");    // Indicate that output is red value
      Serial.print(redVal);  // Print red value
      Serial.print("\t");    // Print a tab
      Serial.print("G:");    // Repeat for grn and blu...
      Serial.print(grnVal);
      Serial.print("\t");
      Serial.print("B:");
      Serial.println(bluVal); // println, to end with a carriage return
    }
  }
}
```

E ora un programma un po' più sofisticato per controllare la tonalità di colore: trasforma un valore di tonalità in RGB. Ricordiamo che lo spazio sottrattivo HSV (Hue, Saturation, Value) è una versione più sofisticata dello spazio RGB, da cui si può ricavare.

```
* RGB Pot Mixer2
 * --------------
 *
 * Code for making one potentiometer control 3 LEDs, red, grn and blu,
 * or one tri-color LED.
 *
 * Uses a purportedly correct algorithm for converting a hue number into RGB
values
 *
 * Code assumes you have the LEDs connected in a common-anode configuration,
 * with the LED's anode connected to +5V via a resistor and the cathode
connected
 * to Arduino pins 9,10,11.
 *
 * Tod E. Kurt <tod@todbot.com>, serial debug by Clay Shirky
 *
 */

// INPUT: Potentiometer should be connected to 5V and GND
int potPin = 2;  // Potentiometer output
int potVal = 0;  // Variable to store the input from the potentiometer

// OUTPUT: Use digital pins 9-11, the Pulse-width Modulation (PWM) pins
int redPin = 9;   // Red LED,   connected to digital pin 9
int grnPin = 11;  // Green LED, connected to digital pin 10
int bluPin = 10;  // Blue LED,  connected to digital pin 11
```

```
// Program variables
int redVal = 0;   // Variables to store the values to send to the pins
int grnVal = 0;
int bluVal = 0;

int DEBUG = 0;          // Set to 1 to turn on debugging output

void setup()
{
  pinMode(redPin, OUTPUT);   // sets the pins as output
  pinMode(grnPin, OUTPUT);
  pinMode(bluPin, OUTPUT);

  if (DEBUG) {           // If we want to see the pin values for
debugging...
    Serial.begin(19200);  // ...set up the serial ouput in 0004 format
  }
}

// Main program
void loop()
{
  potVal = analogRead(potPin);   // read the potentiometer value at the
input pin
  potVal = potVal / 4;           // convert from 0-1023 to 0-255

  hue_to_rgb( potVal );          // treat potVal as hue and convert to rgb
vals

  // "255-" is because we have common-anode LEDs, not common-cathode
  analogWrite(redPin, 255-redVal);   // Write values to LED pins
  analogWrite(grnPin, 255-grnVal);
  analogWrite(bluPin, 255-bluVal);

  if (DEBUG) { // If we want to read the output
    DEBUG += 1;      // Increment the DEBUG counter
    if (DEBUG > 100) { // Print every hundred loops
      DEBUG = 1;     // Reset the counter
      Serial.print("R:");    // Indicate that output is red value
      Serial.print(redVal);  // Print red value
      Serial.print("\t");    // Print a tab
      Serial.print("G:");    // Repeat for grn and blu...
      Serial.print(grnVal);
      Serial.print("\t");
      Serial.print("B:");
      Serial.println(bluVal); // println, to end with a carriage return
    }
  }
}

/*
 * Given a variable hue 'h', that ranges from 0-252,
 * set RGB color value appropriately.
 * Assumes maximum Saturation & maximum Value (brightness)
 * Performs purely integer math, no floating point.
 */
void hue_to_rgb(byte hue)
{
    if( hue > 252 ) hue = 252;
    byte hd = hue / 42;   // 42 == 252/6,  252 == H_MAX
```

```
    byte hi = hd % 6;   // gives 0-5
    byte f  = hue % 42;
    byte fs = f * 6;
    switch( hi ) {
        case 0:
            redVal = 252;     grnVal = fs;      bluVal = 0;
           break;
        case 1:
            redVal = 252-fs;  grnVal = 252;     bluVal = 0;
            break;
        case 2:
            redVal = 0;       grnVal = 252;     bluVal = fs;
            break;
        case 3:
            redVal = 0;       grnVal = 252-fs;  bluVal = 252;
            break;
        case 4:
            redVal = fs;      grnVal = 0;       bluVal = 252;
            break;
        case 5:
            redVal = 252;     grnVal = 0;       bluVal = 252-fs;
            break;
    }
}
```

Generatore di codice Morse

Colleghiamo un LED al piedino 12 con una resistenza di limitazione (ad esempio di 220 o 270 ohm). In questo esempio mandiamo il codice Morse per SOS.

Programma 1

```
int ledPin = 12;
int durations[] = {200, 200, 200, 500, 500, 500, 200, 200, 200};

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  for (int i = 0; i < 9; i++)
  {
    flash(durations[i]);
    if (i == 2)
    {
      delay(300);
    }
  }
  delay(1000);                  // wait 1 second before we start again
}

void flash(int duration)
{
  digitalWrite(ledPin, HIGH);
  delay(duration);
  digitalWrite(ledPin, LOW);
```

```
  delay(duration);
}
```

Programma 2

Ora un programma più completo che traduce in Morse un testo

```
// Listing. Project 3.
int ledPin = 11;

char* letters[] = {
  ".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..",    // A-I
  ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.",  // J-R
  "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--.."          // S-Z
};

char* numbers[] = {"-----", ".----", "..---", "...--", "....-", ".....", "-
....", "--...", "---..", "----."};

int dotDelay = 200;

void setup()
{
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  char ch;
  if (Serial.available())      // is there anything to be read from USB?
  {
    ch = Serial.read();        // read a single letter
    if (ch >= 'a' && ch <= 'z')
    {
      flashSequence(letters[ch - 'a']);
    }
    else if (ch >= 'A' && ch <= 'Z')
    {
      flashSequence(letters[ch - 'A']);
    }
    else if (ch >= '0' && ch <= '9')
    {
      flashSequence(numbers[ch - '0']);
    }
    else if (ch == ' ')
    {
     delay(dotDelay * 4);      // gap between words
    }
  }
}

void flashSequence(char* sequence)
{
   int i = 0;
   while (sequence[i] != NULL)
   {
       flashDotOrDash(sequence[i]);
       i++;
   }
   delay(dotDelay * 3);    // gap between letters
```

```
}

void flashDotOrDash(char dotOrDash)
{
  digitalWrite(ledPin, HIGH);
  if (dotOrDash == '.')
  {
    delay(dotDelay);
  }
  else // must be a -
  {
    delay(dotDelay * 3);
  }
  digitalWrite(ledPin, LOW);
  delay(dotDelay); // gap between flashes
}
```

Buon lavoro !

Fine della parte II del Corso di autoapprendimento sulla piattaforma Arduino

Bibliografia (oltre a quella della prima parte)

-Monk Simon, 30 Arduino Projects for the evil genius, Mc Grraw Hill, New York, 2010

-Schmidt Maik, Arduino, A Quick- Start Guide, Pragmatic Bookshelf, Raleigh, North Carolina, 2011