

ARI Sezione di Ivrea Serate tecniche 2012

ARDUINO il corso Lezione 3 Output "Video"

IZiMHN e IW1ALX

ARI Sezione di Ivrea Serate tecniche 2012

Prima di iniziare... (che novità!)

- Domande sulla lezione scorsa? (e chi si ricorda!)
- Come è andato il compito a casa? (avete fatto esercizi, vero???)

Output video 1/2

- I led
 - Gestione dei led
 - Gestione della luminosità dei LED
 - LED ad alta potenza
 - LED RGB
 - Barre di LED
 - ...Supercar!
 - Gestire più LED
 - Matrici LED
 - Multiplexing
 - Charlieplexing

Output video 2/2

- Display a 7 segmenti
- LCD
- Display Grafici

Materiale necessario

- Per questa seconda lezione ci servono:
 - La scheda Arduino UNO R3
 - La breadboard e fili
 - 2 led (1 rosso e uno verde)
 - 1 led RGB
 - 2 display a 7 segmenti e 2 BC547B
 - Un pò di resistenze
 - 1 display LCD almeno 16x2
 - 1 potenziometro Lineare da 10K

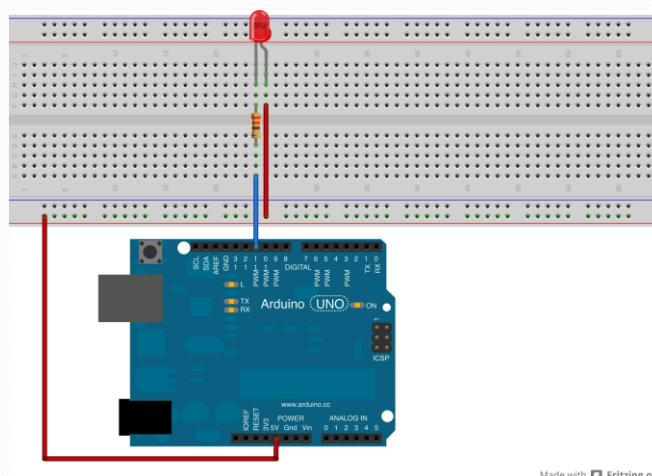
La gestione dei LED

- Digitale
 - Si usa la funzione `digitalWrite(pin, valore)`
 - Valore può essere:
 - HIGH (5V)
 - LOW (0V)
- Analogica
 - Si usa la funzione `analogWrite(pin, value)`
 - Usa la tecnologia PWM (Pulse Width Modulation)
 - Disponibile solo su alcuni PIN

Led "digitali"

- Solo un cenno:
 - Dichiariamo il LED
 - Nel setup lo mettiamo in OUTPUT
 - `digitalWrite (pin,HIGH)` lo accende
 - `digitalWrite (pin,LOW)` lo spegne

Led "digitali": circuito

Made with  Fritzing.org

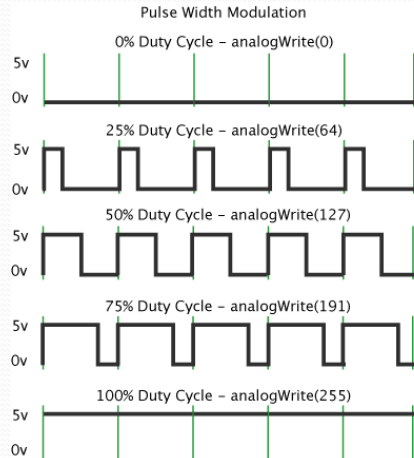
Led “digitali”: lo schetch

```
int led = 11;
void setup () {
  pinMode(led, OUTPUT);
}
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Led “analogici”

- Consentono di variare la luminosità
- L'uscita è “variabile”
- Sfrutta il PWM

Veloci veloci... il PWM



Veloci veloci... il PWM

- In pratica il pin referenziato nella analogWrite genera una quadra con Duty Cycle specifico e continua a generarli fino al successivo “evento” (cioè un'altra chiamata ad una analogWrite, ad una digitalRead o ad una or digitalWrite).
- La frequenza del PWM è circa 490 Hz.

Gestione del PWM

- Si usa la funzione `analogWrite(pin, value)`
 - Value rappresenta il duty cycle
 - Varia tra 0 e 255 (8 bit!)
 - ... vi ricordate le proporzioni, verò???
- Pin disponibili:
 - R3: 3, 5, 6, 9, 10, e 11.
 - Mega: dal pin 2 al pin 13
 - Due: dal pin 2 al pin 13 più I pin DAC₀ e DAC₁

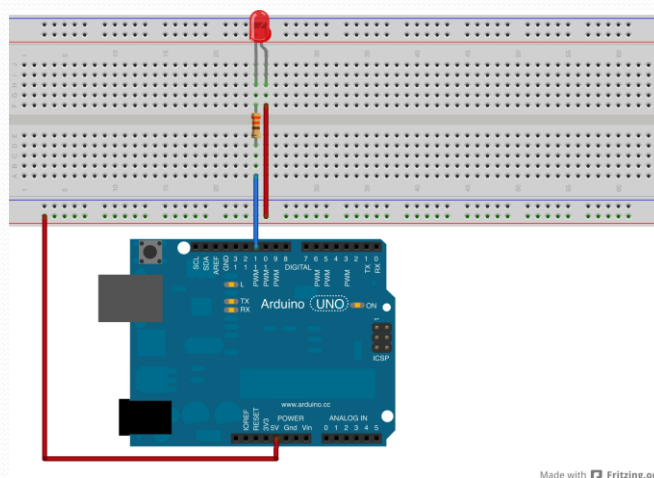
PWM e DAC sulla Due

- Una nota solo sulla Due:
 - 12 pins con PWM a 8-bit come le altre schede
 - 2 pins con un DAC a 12-bit
 - Usando il DAC si arriva a una risoluzione di 4096.
 - Si può impostare la profondità con `analogWriteResolution(bits)`

Gestione della luminosità dei led

- Si varia sfruttando il meccanismo dei PWM
- **ATTENZIONE:** I pin su cui si fa la analogWrite NON devono essere messi in setup come output.
- Il circuito... non varia

Led "analogici": circuito



Led “analogici”: lo schetch

```
int led = 11;
int luminosita = 0;
int passo = 1;

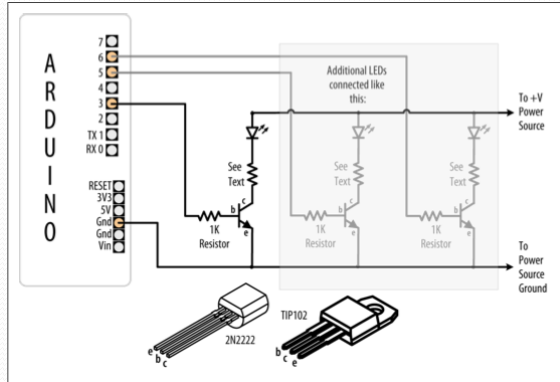
void setup () {
  // non mettere il PIN
}

void loop() {
  if (luminosita > 255) {
    passo = -1;
  }
  else if (luminosita < 1) {
    passo = 1;
  }
  luminosita = luminosita + passo;
  analogWrite(led, luminosita);
  delay (20);
}
```

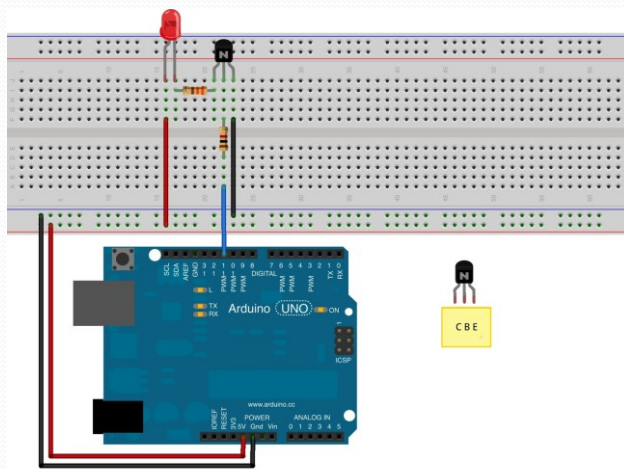
Led e corrente

- Ciascun PIN può pilotare al massimo 40 mA.
- Il PIN +5V di Arduino può portare fino a 400 mA al massimo SE si alimenta Arduino dalla USB, altrimenti il limite è in realtà dato dalla corrente che si mette in alimentazione e dal calore dissipato dal regolatore di tensione.
- Si mette un transistor tra il PIN di Arduino e il LED.
- Ad esempio un BC547B o un 2N2222

Led e corrente: lo schema



Led e corrente: Il circuito



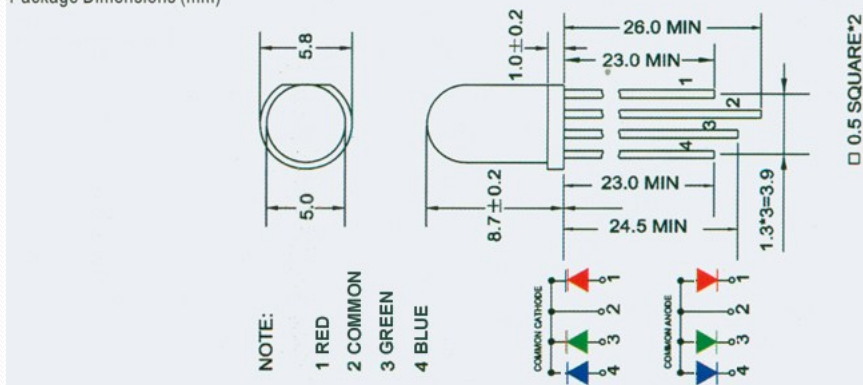
Made with Fritzing.org

Led RGB

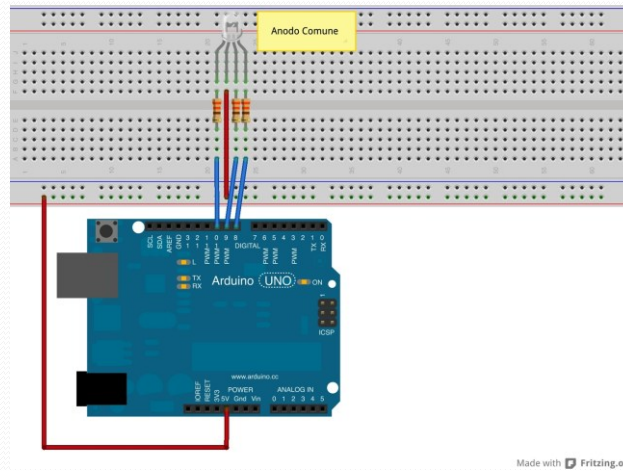
- Sono in pratica tre LED in uno
 - Catodo comune
 - Anodo comune
- Consentono di generare tutti i colori variando in PWM l'intensità del "singolo" LED
- Attenzione alla corrente: molti led RGB sono ad alta luminosità!!!

Led RGB

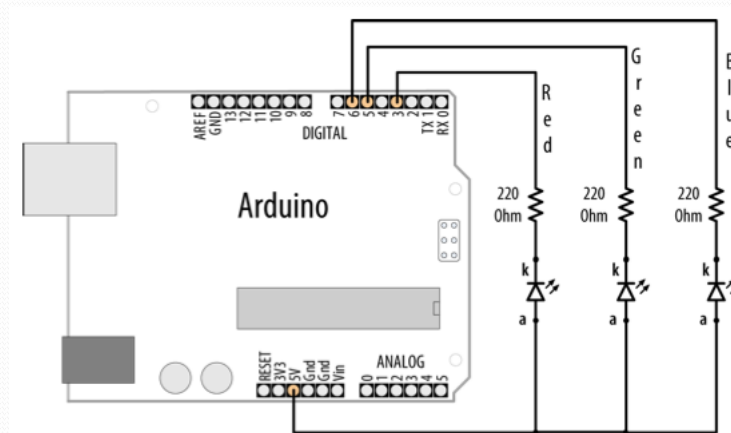
Package Dimensions (mm)



Led RGB: il circuito



Led RGB: lo schema



'spett un moment...

- Definiamo solo cosa è Hue!
- “the degree to which a stimulus can be described as similar to or different from stimuli that are described as red, green, blue, and yellow”

Fonte: en.wikipedia.org/wiki/Hue

'spett ancor un moment...

- Voi come lo fareste?
- Esistono infiniti modi per affrontare il problema...
- In pratica...
 - Si divide il colore per portarlo in RGB
 - Si modula i tre valori in PWM

Led RGB: un esempio 1/3

```
const int redPin = 3; //usare 3 pin che vanno in PWM
const int greenPin = 5;
const int bluePin = 6;

const boolean invert = true; // impostare a TRUE se anodo comune,
altrimenti false

int color = 0; // la uso per rappresentare il valore di HUE

int R, G, B; //è il valore di HUE diviso nei tre componenti
/* li dichiaro qui per usarli globalmente
quindi li posso usare nella funzione sotto
senza dover gestire una funzione che restituisce dei valori */

void setup(){
  // nb: i PIN usati per analogWrite NON vanno dichiarati
}
```

Led RGB: un esempio 2/3

```
void loop(){
  int brightness = 255; // imposto a 255 la massima
  luminosità
  hueToRGB( color, brightness); // richiamo la funzione
  che c'è sotto
  // write the RGB values to the pins
  analogWrite(redPin, R);
  analogWrite(greenPin, G);
  analogWrite(bluePin, B );

  color++; // incremento il colore
  if(color > 255){ //quando sono a 255 reinizio
    color = 0;
  }
  delay(10);
}
```

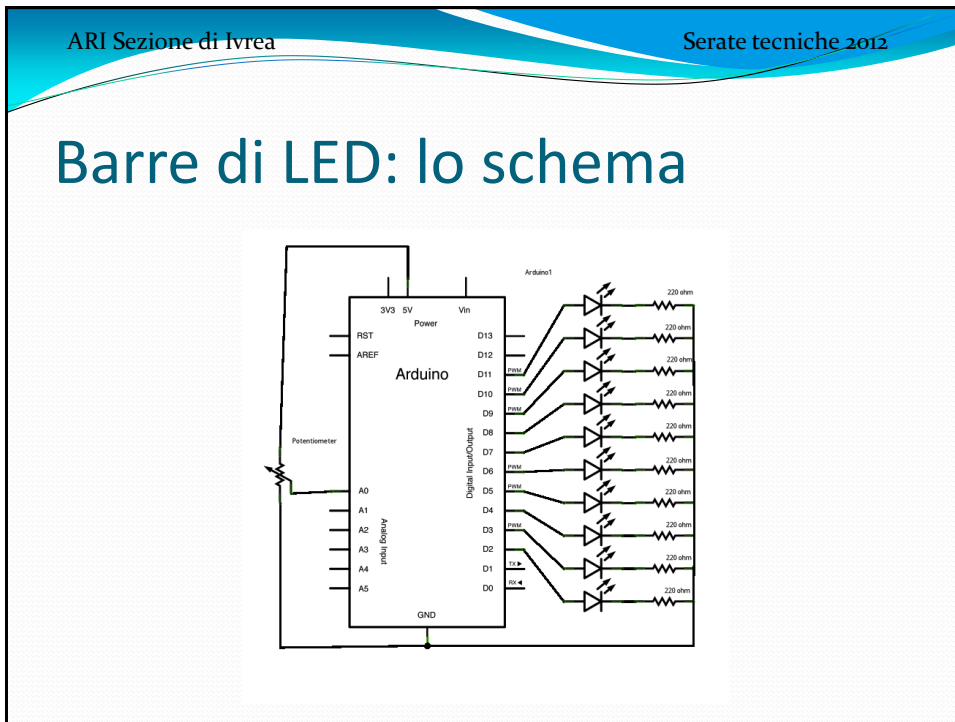
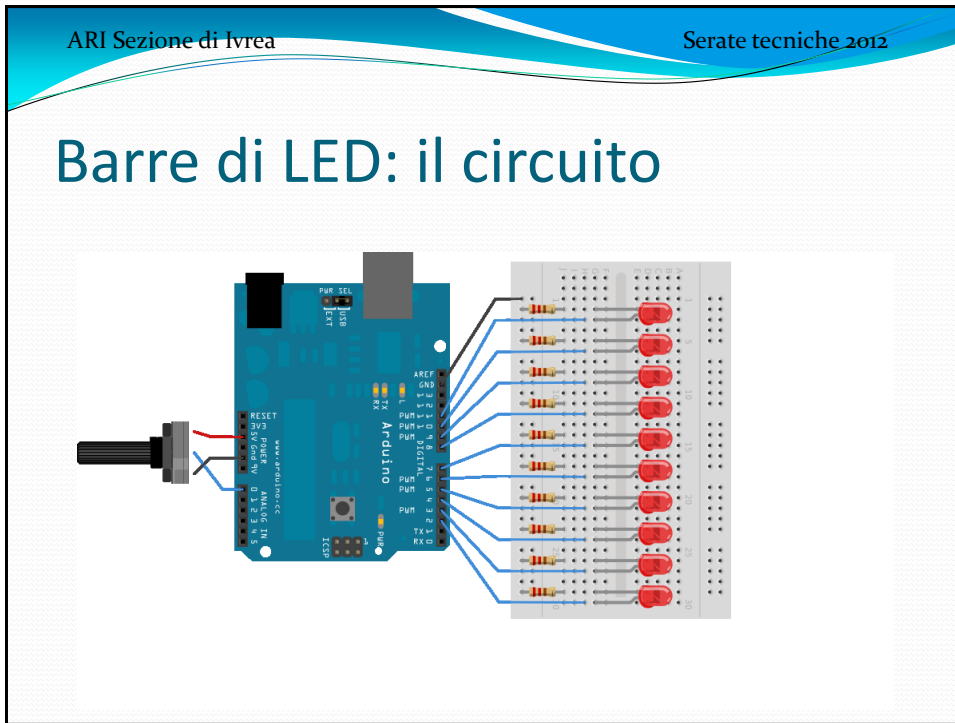
Led RGB: un esempio 3/3

```
// converte un colore nei suoi componenti RGB.
void hueToRGB( int hue, int brightness){
  unsigned int scaledHue = (hue * 6);
  unsigned int segment = scaledHue / 256; // segment 0 to 5 around the color wheel
  unsigned int segmentOffset = scaledHue - (segment * 256); // position within the segment
  unsigned int complement = 0;
  unsigned int prev = (brightness * (255 - segmentOffset)) / 256;
  unsigned int next = (brightness * segmentOffset) / 256;
  if(!invert){
    brightness = 255-brightness; complement = 255;
    prev = 255-prev;
    next = 255-next;
  }
  switch(segment) {
    case 0: // red
      R = brightness; G = next; B = complement;
      break;
    case 1: // yellow
      R = prev; G = brightness; B = complement;
      break;
    case 2: // green
      R = complement; G = brightness; B = next;
      break;
    case 3: // cyan
      R = complement; G = prev; B = brightness;
      break;
    case 4: // blue
      R = next; G = complement; B = brightness;
      break;
    case 5: // magenta default:
      R = brightness; G = complement; B = prev;
      break;
  }
}
```

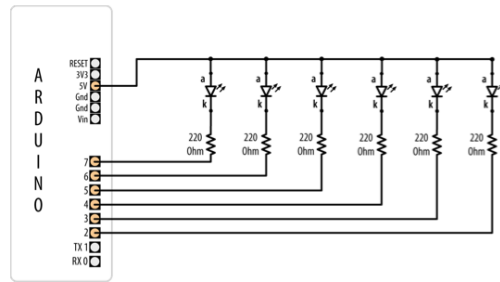
Barre di LED

- Ok, in pratica quello che facevano una volta gli impianti stereo...
- Ci si costruisce un superled multiplo a anodo (o catodo) comune!





Barre di LED: alternativa



Barre di LED: lo sketch 1/2

```

const int analogPin = A0; // Pin per il potenziometro
const int ledCount = 10; // numero di LED per la barra
int ledPins[] = {2, 3, 4, 5, 6, 7,8,9,10,11 }; //
Piedini dei LED

void setup() {
  // Imposto i LED come output
  for (int thisLed = 0; thisLed < ledCount; thisLed++) {
    pinMode(ledPins[thisLed], OUTPUT);
  }
}

```

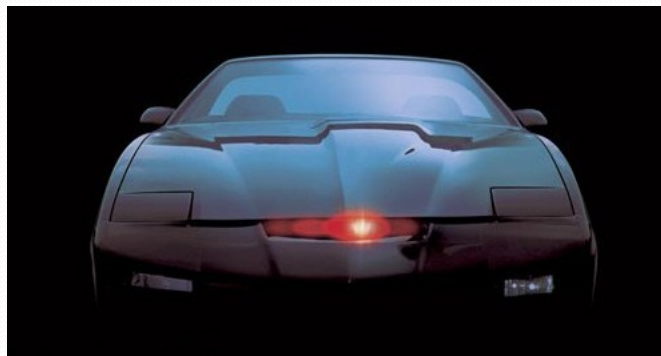
Barre di LED: lo sketch 2/2

```
void loop() {
  // leggi dal potenziometro
  int sensorReading = analogRead(analogPin);
  // scalo la lettura in base al numero di LED
  int ledLevel = map(sensorReading, 0, 1023, 0, ledCount);

  // loop
  for (int thisLed = 0; thisLed < ledCount; thisLed++) {
    // se sono sopra al livello accendo il LED corrispondente
    if (thisLed < ledLevel) {
      digitalWrite(ledPins[thisLed], HIGH);
    }
    // se sono sotto lo spengo
    else {
      digitalWrite(ledPins[thisLed], LOW);
    }
  }
}
```

Supercar

- Vi ricordate Kit...



Supercar: il codice

```
int pinArray[] = {2, 3, 4, 5, 6, 7}; //dichiaro un'array per i led
int count = 0;
int timer = 100;

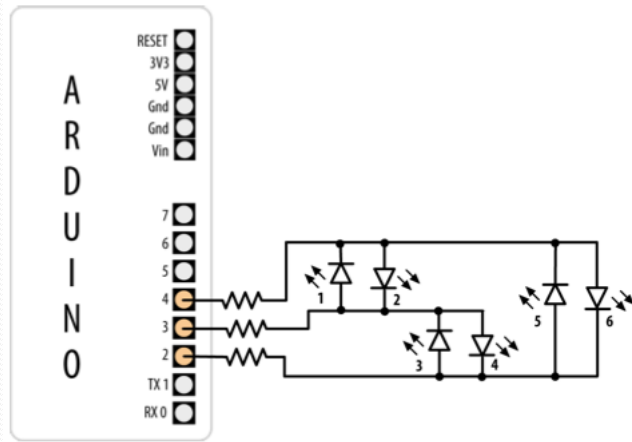
void setup() {
  for (count=0;count<6;count++) {
    pinMode(pinArray[count], OUTPUT);
  }
}

void loop() {
  for (count=0;count<6;count++) {
    digitalWrite(pinArray[count], HIGH);
    delay(timer);
    digitalWrite(pinArray[count], LOW);
    delay(timer);
  }
  for (count=5;count>=0;count--) {
    digitalWrite(pinArray[count], HIGH);
    delay(timer);
    digitalWrite(pinArray[count], LOW);
    delay(timer);
  }
}
```

Gestire più LED: Charlieplexing

- Le soluzioni per matrici e multiplexing, hanno comunque il difetto di consumare PIN
- Il Charlieplexing consente di diminuire significativamente il numero di PIN che servono:
 - 6 LED si pilotano con 3 PIN
 - 12 LED con 4
 - 20 LED con 5
- “Inventata” da Charlie Allen sfrutta la logica che un LED funziona solo in un verso!

Gestire più LED: Charlieplexing

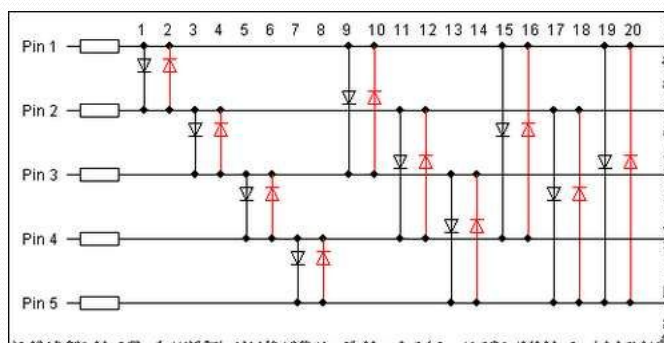


Gestire più LED: Charlieplexing

- In pratica bisogna pensare come se fosse una tabella di verità
- In pratica si basa sul concetto che un PIN può avere tre stati:
 - LOW
 - HIGH
 - INPUT: un PIN impostato a INPUT... lo disconnette effettivamente dal circuito.
- La tabella diventa:

Pins			LEDs					
4	3	2	1	2	3	4	5	6
L	L	L	0	0	0	0	0	0
L	H	i	1	0	0	0	0	0
H	L	i	0	1	0	0	0	0
i	L	H	0	0	1	0	0	0
i	H	L	0	0	0	1	0	0
L	i	H	0	0	0	0	1	0
H	i	L	0	0	0	0	0	1

Gestire più LED: Charlieplexing



Gestire più LED: Charlieplexing

- La cosa simpatica di questa tecnica è che... esiste già una bella libreria!
- <http://playground.arduino.cc/code/charlieplex>
- Per usarla:
 - Si scarica
 - Si copia nella \Libraries
 - Si rilancia l'ADE

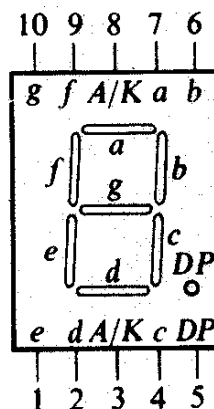
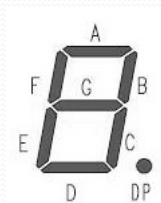
Gestire più LED: Charlieplexing

- Per creare un “oggetto”:
 - `Charlieplex(byte* userPins, byte numberOfUserPins`
- Per creare un “PIN”:
 - `charliePin led = { 0 , 2 }; //HIGH when current flows from 11 to 13`

Display a 7 segmenti

- Anche se “vecchi” i display a 7 segmenti sono ancora usati in diverse applicazioni.
- Fondamentalmente sono un insieme di led (normalmente 8 e non 7: c'è anche il punto!)
- Ogni segmento rappresenta una “linea” che compone i diversi numeri
- Possono essere a anodo o catodo comune

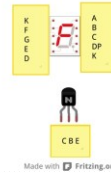
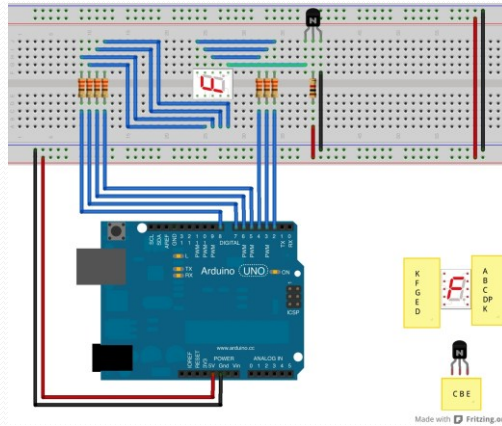
Display a 7 segmenti



La gestione dei 7 segmenti

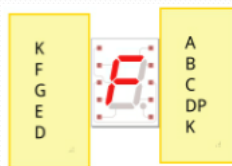
- In pratica si tratta di andare e “costruire” i diversi numeri accendendo o spegnendo i singoli segmenti
- Nell'esempio di prima...
 - 7 = a, b, c accesi e d, e, f, g spenti
 - 6 = a, c, d, e, f, g accesi e b spento
 - ...
- Non è difficile, solo... noioso ;-):
 - O si fa una libreria e la si modifica solo se si usano dei componenti diversi
 - O la mettiamo nella parte globale e la usiamo a seconda di quello che ci serve

7 segmenti: il circuito



Made with fritzing.org

La tabellina delle corrispondenze



Segmento	Pin
A	2
B	3
C	4
D	5
E	6
F	7
G	8

Il che vuol dire...

Numer o	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
0	HIGH	HIGH	HIGH	HIGH	HIGH	HIGH	
1		HIGH	HIGH				
2	HIGH	HIGH		HIGH	HIGH		HIGH
3	HIGH	HIGH	HIGH	HIGH			HIGH
4		HIGH	HIGH			HIGH	HIGH
5	HIGH		HIGH	HIGH		HIGH	HIGH
6	HIGH		HIGH	HIGH	HIGH	HIGH	HIGH
7	HIGH	HIGH	HIGH				
8	HIGH	HIGH	HIGH	HIGH	HIGH	HIGH	HIGH
9	HIGH	HIGH	HIGH	HIGH		HIGH	HIGH

Che in Arduinese diventa...

```
// definisco la tabella numeri-pin
```

```
int A=2;
int B=3;
int C=4;
int D=5;
int E=6;
int F=7;
int G=8;
```



```
//preparo i vari numeri
void zero() {
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
}
```

7 segmenti: lo schetch 1/2

```
// definisco la tabella numeri-pin //preparo i vari numeri
int A=2;                               void zero(){
int B=3;                               digitalWrite(A, HIGH);
int C=4;                               digitalWrite(B, HIGH);
int D=5;                               digitalWrite(C, HIGH);
int E=6;                               digitalWrite(D, HIGH);
int F=7;                               digitalWrite(E, HIGH);
int G=8;                               digitalWrite(F, HIGH);
                                       digitalWrite(G, LOW);
                                       }

void setup(){
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(E, OUTPUT);
  pinMode(F, OUTPUT);
  pinMode(G, OUTPUT);
}

void uno(){
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
}
```

7 segmenti: lo schetch 2/2

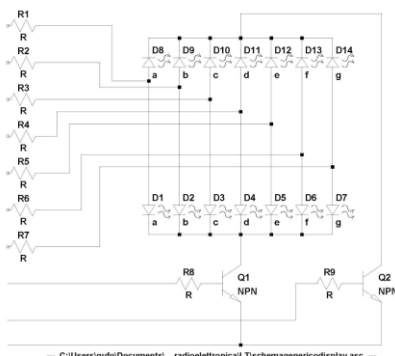
```
void loop(){
  uno();
  delay(500);
  due();
  delay(500);
  tre();
  delay(500);
  quattro();
  delay(500);
  cinque();
  delay(500);
  sei();
  delay(500);
  sette();
  delay(500);
  otto();
  delay(500);
  nove();
  delay(500);
}
```

Gestire più di 7 segmenti

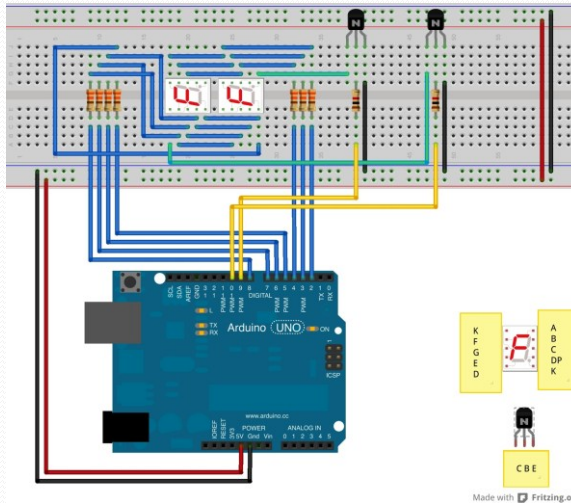
- Si può fare sia con 2 componenti che con “blocchetti” apposti da 2, 3, 4 ... cifre.
- Nel modo più semplice usiamo un transistor (BC547) come select per il display che vogliamo accendere.

Gestire più di 7 segmenti

schema semplificato relativo a due display a catodo comune



Gestire più di 7 segmenti



Lo sketch: alcuni trucchi

- Definire i Pin dei diversi segmenti per non dover impazzire a ricordarsi quale pin accende cosa

```
// definisco la tabella tra segmenti e pin
int segA = 2;
int segB = 3;
int segC = 4;
int segD = 5;
int segE = 6;
int segF = 7;
int segG = 8;
// definisco i pin di select per Decine e Unità
int UN = 9;
int DE = 10;
```

Lo sketch: alcuni trucchi

- Definire un vettore dei pin

```
// vettore dei pin
int tabSegmentiPin[7] = {
    2, //seg a
    3, //seg b
    4, //seg c
    5, //seg d
    6, //seg e
    7, //seg f
    8 //seg g
};
```

Lo sketch: alcuni trucchi

- Usiamo una matrice per i segmenti

```
// Matrice numeri -> segmenti
int tabNumeriSegmenti[10][7] =
{
    //a,b,c,d,e,f,g
    {1,1,1,1,1,1,0}, //zero
    {0,1,1,0,0,0,0}, //uno
    {1,1,0,1,1,0,1}, //due
    {1,1,1,1,0,0,1}, //tre
    {0,1,1,0,0,1,1}, //quattro
    {1,0,1,1,0,1,1}, //cinque
    {1,0,1,1,1,1,1}, //sei
    {1,1,1,0,0,0,0}, //sette
    {1,1,1,1,1,1,1}, //otto
    {1,1,1,1,0,1,1} //nove
};
```

Lo sketch: principali funzioni

```

void visualizzaNumero(int numero)
{
  //sfrutto la funzione modulo:
  // supponiamo che il numero sia 29
  // avremo che:
  // u = 29 % 10 = 9
  // d = (29/10) % 10 = 2.9 % 10 = 2

  int u = numero % 10;
  int d = (numero / 10) % 10;

  boolean cifraPrecedente = false;

  if (d > 0)
  {
    accendiDecine(d);
    cifraPrecedente = true;
  }
  else
  {
    digitalWrite(DE, LOW);
    cifraPrecedente = false;
  }

  accendiUnita(u);
}

```

Lo sketch: principali funzioni

```

// Funzione gestione unita
void accendiUnita(int numUnita)
{
  spegniSegmenti();
  digitalWrite(UN, HIGH);
  digitalWrite(DE, LOW);
  spegniSegmenti();
  attivaSegmenti(numUnita);
}

// Funzione gestione decine
void accendiDecine(int numDecine)
{
  spegniSegmenti();
  digitalWrite(UN, LOW);
  digitalWrite(DE, HIGH);
  spegniSegmenti();
  attivaSegmenti(numDecine);
}

// Funzione lookup pin -> segmento
int ottieniPin(int seg)
{
  return tabSegmentiPin[seg];
}

// Funzione accensione segmenti
void attivaSegmenti(int n)
{
  for (int i=0; i< 7; i++)
  {
    int segAttivo = tabNumeriSegmenti[n][i];
    if (segAttivo == 1)
    {
      digitalWrite(ottieniPin(i), HIGH);
    }
  }
}

```

Espandere lo sketch

- Se vogliamo aumentare i numeri...
 - Aumentiamo i transistor
 - La visualizzaNumero diventa:
 - `int u = numero % 10;`
 - `int d = (numero / 10) % 10;`
 - `int c = (numero / 100) % 10;`
 - ...
- Esistono anche dei display multinumero direttamente con interfaccia I2C

I display LCD

- In questa parte del corso ci occupiamo dei display paralleli. Altre forme di connessione (es I2C) le vedremo più avanti.
- In particolare cerchiamo di utilizzare dei dispositivi compatibili con i driver Hitachi HD44780.
- Li useremo in modalità 4 Pin.

I display LCD

- Disponibili sia come “componente” che come shield
- Tipicamente gli shield LCD incorporano anche 6 pulsanti (gestiti tramite analog in per risparmiare Pin)



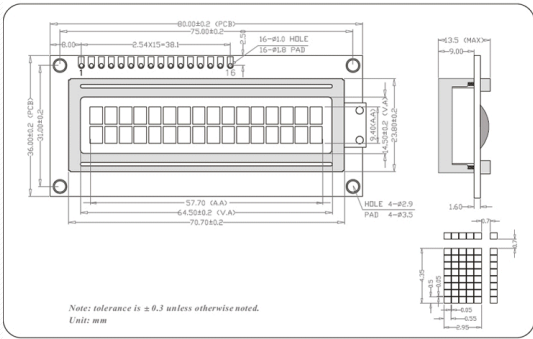
HD 44780

- L'HD44780 è un ASIC sviluppato da Hitachi, che consente di pilotare display a caratteri.
- È considerato uno standard de facto grazie a tre caratteristiche:
 - Semplicità
 - Diffusione
 - Basso costo
- Esistono in diversi formati: 8x1, 8x2, 16x2, 20x2, 16x4, 20x4, ecc.

ARI Sezione di Ivrea Serate tecniche 2012

HD 44780

■ DIMENSIONS/DISPLAY CONTENT



Note: tolerance is ±0.3 unless otherwise noted.
Unit: mm

■ PIN CONFIGURATION

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Vss	Vcc	V ₀	RS	RW	E	DB ₀	DB ₁	DB ₂	DB ₃	DB ₄	DB ₅	DB ₆	DB ₇	LEDA	LEDK

Pin n.	Funzione
1	Vss (Massa)
2	Vdd (+5V)
3	V ₀ (Contrasto)
4	R/S (o comando, 1 dato)
5	R/W (o scrive, 1 legge)
6	E (Enable: inizia scrittura o lettura)
7	DB ₀ (non usato se in modo 4 bit)
8	DB ₁ (non usato se in modo 4 bit)
9	DB ₂ (non usato se in modo 4 bit)
10	DB ₃ (non usato se in modo 4 bit)
11	DB ₄
12	DB ₅
13	DB ₆
14	DB ₇
15	LED A (anodo retroilluminazione)
16	LED K (catodo retroilluminazione)

ARI Sezione di Ivrea Serate tecniche 2012

HD 44780 funzionamento 1/2

- Protocollo di comunicazione
 - Impostare R/S a 1 per inviare dati, a 0 per inviare comandi
 - Impostare E a 1
 - Caricare i 4 msbit
 - Impostare E a 0
 - Impostare E a 1
 - Caricare i 4 lsbit
 - Portare E a 0
 - NB: Usare un ritardo di 2ms tra ogni operazione
 - NB: se si usa il modo a 8 bit si caricano i dati tutti in una volta

HD 44780 funzionamento 2/2

- Inizializzazione del display
 - Appena acceso, il display, va inizializzato, inviando i comandi base. Primo fra tutti la selezione del modo di comunicazione a 4 o 8 bit.
- Inviare caratteri al display
 - Dopo l'inizializzazione è possibile inviare i caratteri trasmettendo il loro valore ASCII.
 - Il display incrementa automaticamente la posizione, quindi non è necessario fare nulla tra un carattere ed il successivo

HD 44780: alcuni link al volo

- Datasheet: ovunque... ad esempio:
www.sparkfun.com/datasheets/LCD/HD44780.pdf
- Tutorial: lcd-linux.sourceforge.net/pdffocs/lcd1.pdf e
lcd-linux.sourceforge.net/pdffocs/lcd2.pdf
- Simulatore:
www.geocities.com/dinceraydin/djlcdsim/djlcdsim.html

HD 44780 e Arduino

- Moolto semplice:
 - `#include <LiquidCrystal.h>`
- Cioè... esiste una libreria specifica per la sua gestione!
 - “This library allows an Arduino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with in either 4- or 8-bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).”

LiquidCristal: il Costruttore

- `LiquidCrystal(rs, e, d4, d5, d6, d7)`
- `LiquidCrystal(rs, rw, e, d4, d5, d6, d7)`
- `LiquidCrystal(rs, e, d0, d1, d2, d3, d4, d5, d6, d7)`
- `LiquidCrystal(rs, rw, e, d0, d1, d2, d3, d4, d5, d6, d7)`
- Consente di creare una “variabile” di tipo LCD display sia a 4 bit, sia a 8 bit.
- In entrambi i casi il parametro rw è opzionale.
- Vanno specificati i Pin che connettono il display ad Arduino

LiquidCristal: un esempio

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);

void setup()
{
  lcd.print("hello, world!");
}

void loop() {}
```

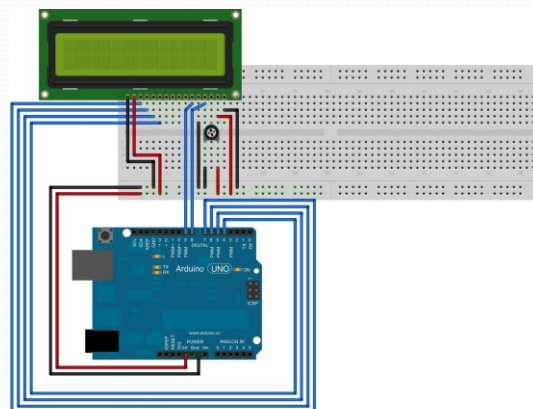
LiquidCristal: alcuni metodi

- `begin()`: specifica la dimensione del display (colonne, righe).
- `clear()`: pulisce il display e riposta il cursore in alto a sx.
- `home()`: sposta il cursore in alto a sx ma NON pulisce.
- `setCursor()`: posiziona il cursore (colonna, riga).
- `write()`: scrive un carattere sul display
- `print()`: scrive un testo sul display

LiquidCristal: write VS print

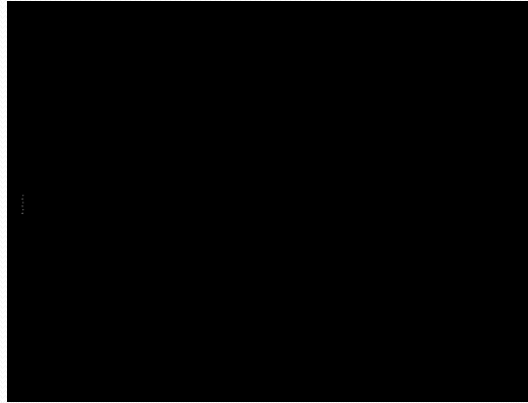
- Tecnicamente parlando la differenza è minima.
- La print converte in stringhe, mentre la write lascia il dato grezzo.
- La write ha senso solo se si usano LCD che accettano anche i comandi di contrasto, luminosità o simili.
- La write si usa se si devono utilizzare caratteri custom (creati con la createChar).
- La print richiama internamente la write e questo introduce un minimo di overhead.

LCD: il circuito



LCD: lo schema

- NB: attenzione i Pin differiscono dal circuito



LCD: lo sketch

- PIN per il nostro esempio:
 - LCD RS pin 8
 - LCD Enable pin 9
 - LCD D4 pin 4
 - LCD D5 pin 5
 - LCD D6 pin 6
 - LCD D7 pin 7
 - LCD R/W pin ground
 - LCD VSS pin ground
 - LCD VDD pin VCC
 - LCD V0 10K
- La piedinatura varia da shield a shield: verificare sempre piedini con Tester

LCD: lo sketch

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
void setup() {
  lcd.begin(16, 2);
  lcd.print("Ciao mamma!");
}
void loop() {
  lcd.setCursor(0, 1);
  lcd.print(millis()/1000);
}
```

I display grafici

- Usano sempre e comunque delle librerie
- Possono essere a colori o in B/N
- Possono essere touch screen
- Molti diffusi grazie al basso costo sono quelli derivati da display di telefonini (la parte da leone la fanno i display dei Nokia 5110)
- Ps... un piccolo consiglio... usate la carta millimetrata!

ARI Sezione di Ivrea Serate tecniche 2012

Q & A

ARI Sezione di Ivrea Serate tecniche 2012

Compitino

- Questa volta due compiti
 - Leggere un valore di tensione su un pin analogico e stamparlo su un LCD.
 - Implementare i classici 6 bottoni in analogico e gestirli sul display.

Next time: i sensori

- Movimento
- Luce
- Distanza
- Suono
- Rotazione
- Posizione (GPS)
- Accelerazione

Che ci servirà?

Oltre alla solite cose:

- Parliamone!
 - Distanza?
 - Luce?
 - ...