ARI Sezione di Ivrea Serate tecniche 2012

ARDUINO il corso
Lezione 2
Semplici input

IZ1MHN e IW1ALX

ARI Sezione di Ivrea Serate tecniche 2012

Prima di iniziare…

- Domande sulla lezione scorsa?
- Come è andato il compito a casa?

Semplici input

- Pulsanti
- Debounce
- Keypad
- Leggere un valore analogico
  - Leggere più di 6 valori
  - Leggere più di 5 volt

ARI Sezione di Ivrea Serate tecniche 2012

Materiale necessario

- Per questa seconda lezione ci servono:
  - La scheda Arduino UNO R3
  - La breadboard e dei fili rigidi da breadboard
  - 2 led (1 rosso e uno verde)
  - 2 pulsanti n.a. e una tastiera 4x4
  - 2 resistenze da 330 e 2 da 10K
  - 1 potenziometro Lineare da 10K

Pulsanti

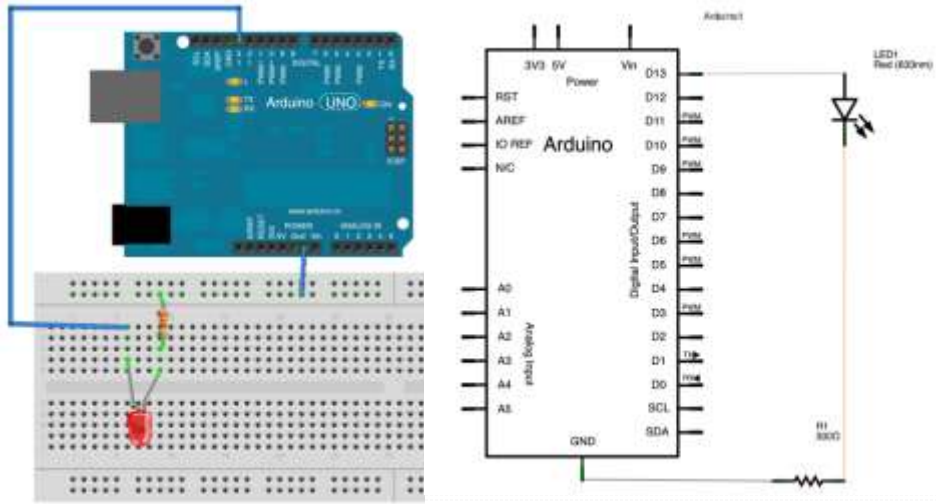
- Uso di un pulsante
- Pulsanti con pull-up interna

ARI Sezione di Ivrea Serate tecniche 2012

Ok… let's go!

- Ripartiamo dal LED acceso/spento con il delay
  - Fritzing VS Schematico
  - Lo schetch
- Aggiungiamo un semplice pulsante

Accendere un led



Al volo Fritzing

- "Fritzing is an open-source hardware initiative to support designers, artists, researchers and hobbyists to work creatively with interactive electronics"
- È un sistema per "disegnare" circuiti con la stessa logica di Arduino: funzionale per chi non ha particolari skill!
- Si parte dal disegno di come metto i fili e ottengo la scheda via posta! (circa 29€ a shield)
- Fritzing.org

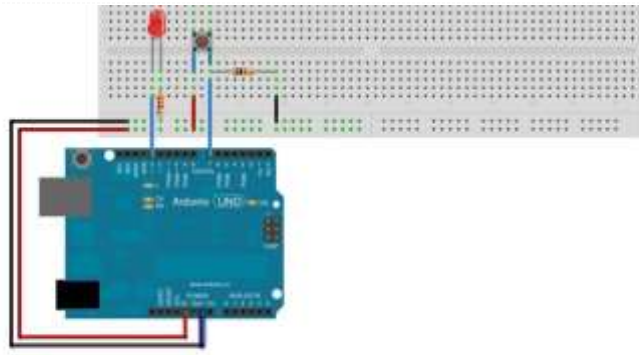
Accendere un led: lo sketch

```
// sketch minimale per lampeggio pin 13

int led = 13;

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

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


Aggiungiamo un pulsante

In pratica vogliamo un pulsante connesso ad esempio al pin 7 in modo che, finché lo si preme, il LED si accende!

Il circuito



Lo sketch

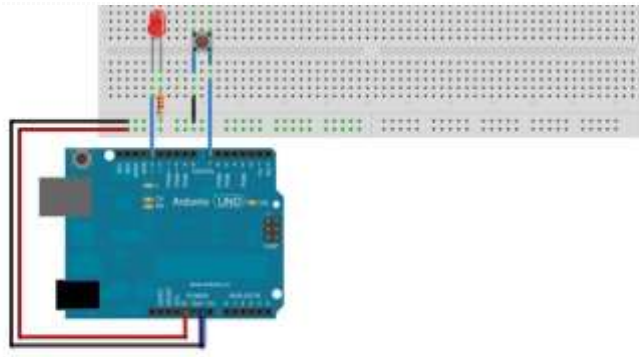
```
int bottone = 7;
int led = 13;

void setup (){
  pinMode(bottone, INPUT);
  pinMode(led, OUTPUT);
}

void loop(){
  int leggistato = digitalRead(bottone);
  if (leggistato == HIGH)
  {
    digitalWrite(led, HIGH);
  }
  else
  {
    digitalWrite(led, LOW);
  }
}
```

Un'alternativa: pull-up interno

- Come abbiamo visto i digital pin devono avere una resistenza per "portare" il pin a un valore valido quando il pulsante non è premuto.
- Arduino ha una resistenza interna di pull-up che può venire abilitata impostando a HIGH un pin che è impostato a INPUT.
- In pratica:

```
Void setup(){
    pinMode(pinInput, INPUT):
    digitalWrite(PinInput, HIGH);
    }
```


Pull-UP interno



Pull-UP interno: lo sketch

```
int led = 13;
int bottone = 7;

void setup(){
  pinMode (led, OUTPUT);
  pinMode (bottone, INPUT);
  digitalWrite(bottone, HIGH); //attiva la resistenza interna di pull-up
}

void loop (){
  int leggistato = digitalRead(bottone);
  if (leggistato == HIGH)
  {
    digitalWrite(led, HIGH);
  }
  else
  {
    digitalWrite(led,LOW);
  }
}
```


Mentre ci siamo: if-else

Il costrutto **If** consente di valutare le condizioni.

La sintassi è:

```
If (condizione)
     {istruzioni se la condizione è vera}
```

Oppure:

```
If (condizione)
     {istruzioni se la condizione è vera}
Else
     {istruzioni se la condizione è falsa}
```

If-else

È anche possibile mettere delle "condizioni multiple":

La sintassi è:

```
If (condizione)
      {istruzioni se la condizione è vera}
Else if (nuova_condizione)
      {istruzioni se la nuova_condizione è vera}
…
Else
      {istruzioni se tutte le condizioni precedenti
non sono vere}
```


Gli operatori di confronto

- Le condizioni vengono espresse per mezzo di operatori di confronto: in pratica si dice fai una cosa se una certa variabile è più grande di un certo valore
- Gli operatori consentono sia confronti tra grandezze numeriche che stringhe
- È possibile concatenare tra loro più condizioni per mezzo di operatori logici.

Operatori numerici

Operatore	Test	Esempio

==	Uguale a	2==2 //vero
		2==3 //falso

!=	Non uguale a	2!=2 //falso 2!=3 //vero
>	Strettamente maggiore	2>1 //vero 2>2 //falso
>=	Maggiore o uguale	2>=2 //vero
<	Strettamente minore	2<3 //vero 2<2 //falso
<=	Minore o uguale	2<=2 //vero

Operatori logici

Operatore	Funzione logica	Descrizione
&&	AND Logico	Vera se ENTRAMBE le condizioni sono vere
||	OR Logico	Vera se ALMENO UNA delle condizioni è vera
!	NOT	Vera se la condizione è falsa e viceversa

Confronto fra stringhe

- Si sfrutta la funzione **`strcmp`**, che restituisce o se le due stringhe sono uguali.
- Ad esempio

```
char string1[ ] = ”primo”;
char string2[ ] = ”secondo";
if(strcmp(string1, string2) == 0)
   Serial.print(”le stringhe sono uguali”);
```

ARI Sezione di Ivrea Serate tecniche 2012

Confronto tra stringhe

- Utilizzando variabili di tipo String è possibile utilizzare gli operatori ==, !=, >, <, >=, <= per effettuare confronti ALFABETICI tra due variabili.
- Se le prime due lettere sono uguali, si passa alla seconda e così via.
- Il confronto è case sensitive. Fa eccezione la funzione equalsIgnoreCase().

Esercizio

Modificare lo sketch per far sì che il led cambi stato ad ogni pressione del bottone.

Soluzione: lo sketch

```
//variabili globali
int bottone = 7;
int led = 13;

void setup (){
  pinMode(bottone, INPUT);
  pinMode(led, OUTPUT);
}

void loop(){
  int leggistato = digitalRead(bottone);
  if (leggistato == HIGH)
    digitalWrite(led, !digitalRead(led));
}
```

Mentre ci siamo: il toggle

```
digitalWrite(led, !digitalRead(led));
```

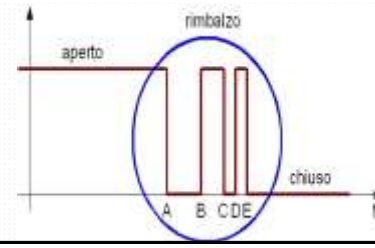
- Questa particolare funzione prende il nome di Toggle.
- In pratica, se il led è acceso lo spegne, se è spento lo accende.
- Il punto esclamativo è l'operatore NOT che posto davanti ad uno stato (HIGH o LOW) ne fa il contrario.

Debounce

- Il problema
- Una possibile soluzione
- Lo sketch

Il problema

- Quando premiamo un pulsante, il contatto non si chiude con un livello stabile, ma a causa delle vibrazioni, continua per qualche decimo ad aprirsi e chiudersi.
- Di conseguenza lo stato varia in continuazione.
- Per risolvere il problema si usano le tecniche di antirimbalzo o debounce.



Una possibile soluzione

- Il “trucco” consiste nel rifare la lettura dopo un ritardo di debounce.
  - Se il valore rimane uguale al precedente lo consideriamo valido.
  - Se il valore è diverso, resettiamo il timer e usiamo il nuovo valore come nuovo valore di confronto e riavviamo il processo.

Lo sketch: il dettaglio

```
…
Int attesa = 10; //durata in ms del debounce
boolean debounce(int pin)
{
  boolean stato;
  boolean statoprecedente;
  statoprecedente = digitalRead(pin); //leggo lo stato
  for (int counter=0; counter < attesa; counter++)
  {
    delay(1);  //aspetto 1 ms
    stato = digitalRead(pin); //leggo lo stato
    if (stato != statoprecedente)
    {
      counter = 0; //reset del contatore
      statoprecedente = stato; //imposto il nuovo stato
    }
   return stato; //restituisco dalla funzione
  }
}
```


Lo sketch

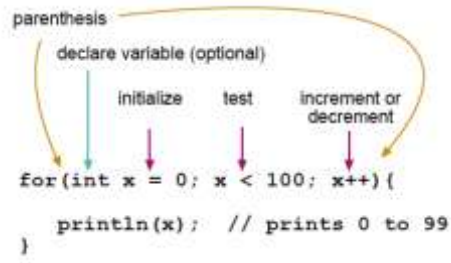
```
int led = 13;
int bottone = 7;
Int attesa = 10; //durata in ms del debounce
boolean debounce(int pin)
…
void setup(){
  pinMode (led, OUTPUT);
  pinMode (bottone, INPUT);
}

void loop (){
  if (debounce(bottone)) //chiamo la funzione passando il PIN
  {
    digitalWrite(led,HIGH);
  }
}
```

Mentre ci siamo: il ciclo for

Il costrutto **for** consente di ripetere un blocco di istruzioni fino al raggiungere una condizione.

La sintassi è:

```
for (dichiarazione; condizione; incremento)
     {istruzioni fino alla condizione}
```



Il ciclo for: esempio

```
int x = 0;
void setup () {
  Serial.begin(9600);
}
void loop () {
  Serial.print("Contatore");
  Serial.print("\t");

  for (x=0; x<9; x++){
    Serial.print(x);
    Serial.print("\t");
    delay(50);
  }
}
```

Mentre ci siamo: il ciclo while

Il costrutto **while** ripete un blocco di istruzioni fino a quando la condizione diventa falsa.

La sintassi è:

```
While (condizione)
      {istruzioni}
```

La condizione può variare o con un contatore che si incrementa o con una condizione esterna (un sensore o un pulsante!).

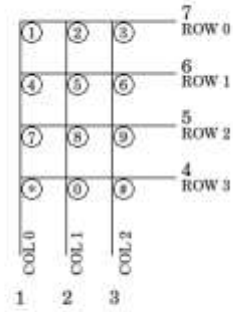
Il ciclo while: esempio

```
int x = 0;
void setup () {
  Serial.begin(9600);
}
void loop () {
  Serial.print("Contatore");
  Serial.print("\t");

  while (x<10){
    Serial.print(x);
    Serial.print("\t");
    x++;
    delay(50);
  }
}
```

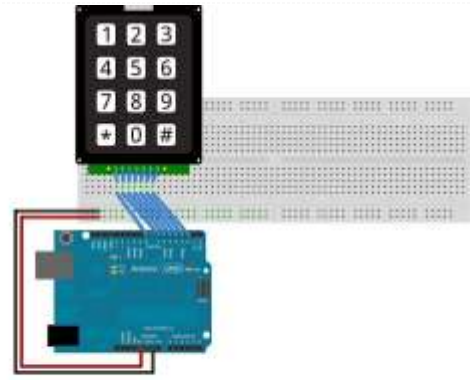
Keypad

- Cosa sono e come funzionano
- Pilotare una keypad in Arduino
  - Il circuito
  - La libreria
  - Un esempio

Cos'è un keypad



Il circuito



La libreria

- È possibile gestire direttamente un keypad (e ci sono molti esempi in giro per farlo).
- Molto più semplicemente possiamo utilizzare una comoda libreria!
- Al momento non fa parte delle librerie installate di default nell'IDE.
- NB:
  - è una libreria generica. Non funziona solo per i 4x3!
  - NB: è possibile istanziare più di un keypad contemporaneamente!

Installazione della libreria

- Scarichiamo dal playground:
  - http://arduino.cc/playground/Code/Keypad#Functions
- Chiudiamo l'IDE
- Una volta scaricato lo ZIP va scompattato nella cartella arduino\libraries\
- Attenzione: crea una cartella che si chiama keypad. Questa contiene altre due cartelle Keypad e Keypad_I2C che vanno spostate in modo da essere in *arduino*\libraries\. Dopo averle spostate si può cancellare quella originale.
- Rilanciando l'IDE selezioniamo Sketch -> Import Library e in basso troveremo le due nuove librerie!

Uso della libreria

- Quando si sceglie la libreria, l'IDE aggiunge automaticamente nello sketch la riga necessaria per includerla:
  - `#include <Keypad.h>`
- Ora è possibile definire un oggetto di tipo Keypad che ha come costruttore:
  - `Keypad(makeKeymap(userKeymap), row[], col[], rows, cols)`
- Dopo aver dichiarato l'oggetto possiamo sfruttare tutti i metodi che la libreria ci mette a disposizione

La “creazione” di un keypad

```
#include <Keypad.h>

const byte ROWS = 4; //numero di righe
const byte COLS = 3; //numero di colonne
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'#','0','*'}
};
byte rowPins[ROWS] = {5, 4, 3, 2}; //Righe
byte colPins[COLS] = {8, 7, 6}; //Colonne

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
```


I metodi della libreria 1/3

- `char waitForKey()`: This function will wait forever until someone presses a key. Warning: It blocks all other code until a key is pressed. That means no blinking LED's, no LCD screen updates, no nothing with the exception of interrupt routines.
- `char getKey()`: Returns the key that is pressed, if any. This function is non-blocking.

I metodi della libreria 2/3

- `KeyState getState()`: Returns the current state of any of the keys. The four states are IDLE, PRESSED, RELEASED and HOLD.
- `boolean keyStateChanged()`: Let's you know when the key has changed from one state to another. For example, instead of just testing for a valid key you can test for when a key was pressed.

I metodi della libreria 3/3

- `setHoldTime(unsigned int time)`: Set the amount of milliseconds the user will have to hold a button until the HOLD state is triggered.
- `setDebounceTime(unsigned int time)`: Set the amount of milliseconds the keypad will wait until it accepts a new keypress/keyEvent. This is the "time delay" debounce method.
- `addEventListener(keypadEvent)`: Trigger an event if the keypad is used.

Lo sketch

```
#include <Keypad.h>
const byte ROWS = 4;
…
…
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){
  Serial.begin(9600);
}
void loop(){
  char key = keypad.getKey();

  if (key != NO_KEY){
    Serial.println(key);
  }
}
```

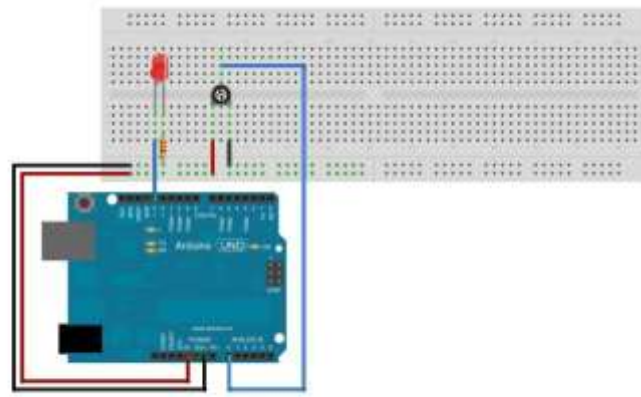

Leggere un valore analogico

- Analog In
- Il circuito
- Il problema
- analogRead()
- Lo sketch
- Cambiare le scale di lettura
- I limiti

Analog In

- Le schede Arduino sono dotate di un buon numero di ingressi analogici:
  - La Arduino Uno R3 ha 6 pin
  - La Arduino Mega ha 16 pin
  - La Arduino Due ha 12 pin
- Questi pin consentono la lettura di valori compresi tra 0V e 5 V (la Due si ferma a 3.3V).

Il circuito



Il problema

- Supponiamo di voler far lampeggiare un led con una frequenza proporzionale al valore di un potenziometro.
- E vogliamo anche “stampare” su seriale il valore!

analogRead()

- Consente di leggere da uno dei 6 (o 16 o 12) PIN analogici un valore compreso tra 0V e 5V.
- Il valore viene “traslato” su una scala da 0 a 1023, quindi in pratica 1024 valori.

Lo sketch

```
int PinPotenziometro = 0;
int PinLed = 13;
int lettura = 0;

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

void loop ()
{
  lettura=analogRead(PinPotenziometro);
  Serial.println(lettura);
  digitalWrite(PinLed,HIGH);
  delay(lettura);
  digitalWrite(PinLed,LOW);
  delay(lettura);
}
```


Cambiare le scale di lettura

- Arduino offre una comoda funzione per riscalare le letture della analogRead().
- La funzione **map** consente di spostare gli estremi dagli originali 0 e 1023.
- La sintassi è:

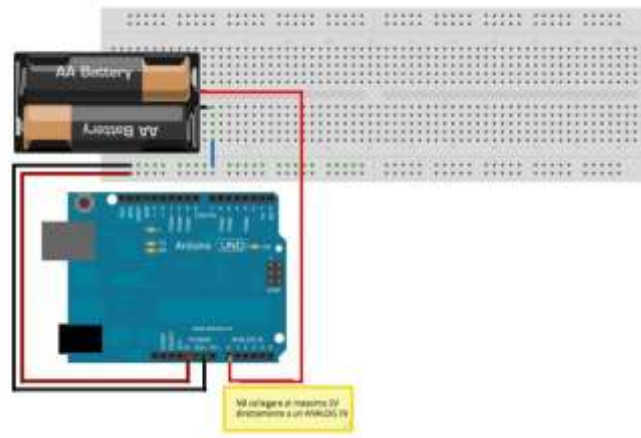
  `map(variabile,0,1023,`**`nuovoMIN,nuovoMAX`**`)`
- Ad esempio

```
Lettura=analogRead(PinPotenziometro);
//convertiamo il valore in %
Percento=map(lettura,0,1023,0,100);
```

Esercizio

Scrivere uno sketch che consenta di visualizzare (sul serial monitor) una tensione compresa tra 0 e 5 V.

Un voltmetro da 0 a 5V: ciruito



Un voltmetro da 0 a 5V: sketch

```
float VRef = 5.0;
float Volt = 5.0;
int VPIN = 0;
int VLetta = 0;

void setup(){
  Serial.begin(9600);
}

void loop(){
  VLetta = analogRead(VPIN);
  //converto: Volt=(lettura/passi) * VRef
  Volt = (VLetta/1024) * VRef;
  Serial.println(Volt);
}
```

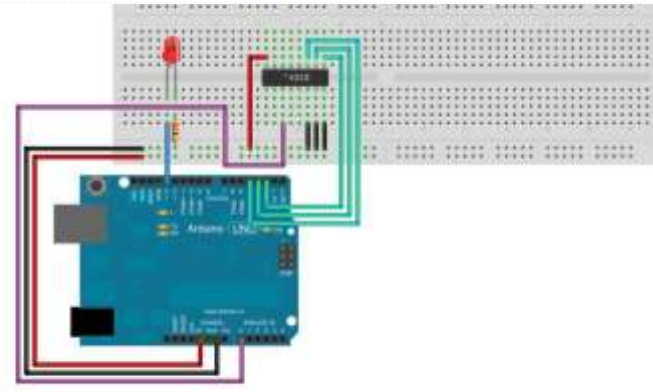

I limiti

- I PIN sono limitati: 6 per la UNO, 16 per la MEGA
- Il range di tensione va da 0 a 5 Volt!

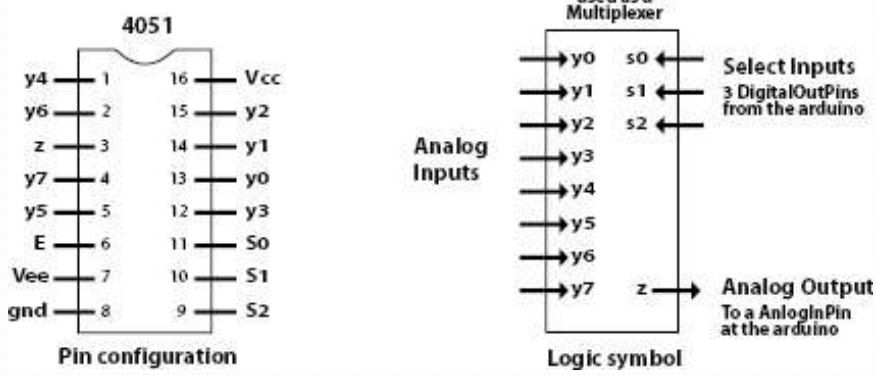
Leggere più di 6 valori

- Il trucco è semplice: leggere più valori su un solo PIN per mezzo di un multiplexer pilotato dai PIN digitali!
- Nel nostro caso useremo un 4051
- Un link… ferroviario:
  - http://dawson-station.blogspot.it/2010/01/mux-demux-cd4051-parlor-tricks.html

Il circuito



Il 4051 in (molto) breve



Il multiplexer

- I 3 PIN (2,3,4) consentono di scegliere quale ingresso del Multiplexer deve essere "passato" al pin analog 0 di Arduino.
- La scelta viene fatta in base al valore che in binario assumono i 3 PIN digitali

S2	S1	S0	OUT
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Lo sketch

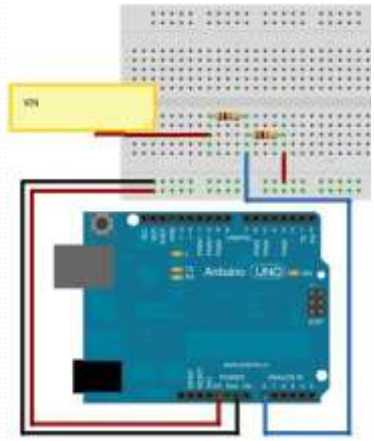
```
int r0 = 0;
int r1 = 0;
int r2 = 0;

int PinAnalogico = 0;
int cont = 0;
int lettura = 0;

void setup(){
  pinMode(2, OUTPUT);     // s0
  pinMode(3, OUTPUT);     // s1
  pinMode(4, OUTPUT);     // s2
}
```

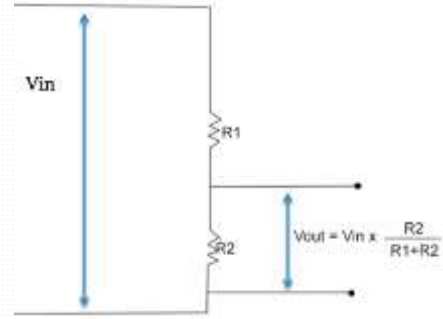
```
void loop () {
  for (cont=0; cont<=7; cont++) {
    // converto in binario cont
    r0 = bitRead(cont,0);
    r1 = bitRead(cont,1);
    r2 = bitRead(cont,2);
    digitalWrite(2, r0);
    digitalWrite(3, r1);
    digitalWrite(4, r2);

lettura=analogRead(PinAnalogico);
    Serial.println(lettura);
  }
}
```


Leggere più di 5 Volt

- Ok Arduino su Analog IN accetta al massimo 5V (in alcune schede a basso consumo 3.3V).
- E se devo misurare un valore più ampio?
- Uso un partitore! E lo metto direttamente nei calcoli!

Partitore: il circuito





Partitore

- In pratica prima avevo:

```
volt=(lettura/passi) * Vref;
```

- Adesso userò:

```
int R1=10000; //valore di R1
int R2=10000; //valore di R2
float fattoreR=1024/(R2/(R2+R1);
volt=(lettura/fattoreR) * Vref;
```

Voltmetro “multiscala”

- È solo un’idea, ma keypad per selezionare i range attraverso un…
- Multiplexer con collegati diverse coppie R1 e R2 per realizzare un…
- Partitore con cui effettuare la misura!

Q & A

Compiti a casa

- Rivedere tutta la lezione.
- Inventarsi come determinare per quanto tempo si è premuto un tasto.
- Inventarsi come accendere un LED quando la tensione scende sotto una soglia.

Next time: Output “video”!

- I LED
  - Gestione dei led
  - Gestione della luminosità dei LED
  - LED RGB
  - Gestire più LED: Matrici LED e Multiplexing
- Display a 7 segmenti
- LCD
- Display Grafici

Che ci servirà?

Oltre alla solite cose:

- Un po' di LED.
- Un led RGB o una matrice di LED (che ci interessa di più?
- Due display a 7 segmenti e 1 74HC595
- Un display LCD 16x2.