Progetti reali con ARDUINO

Introduzione alla scheda Arduino (parte 4ª)

gennaio 2015 – Giorgio Carpignano

I.I.S. PRIMO LEVI

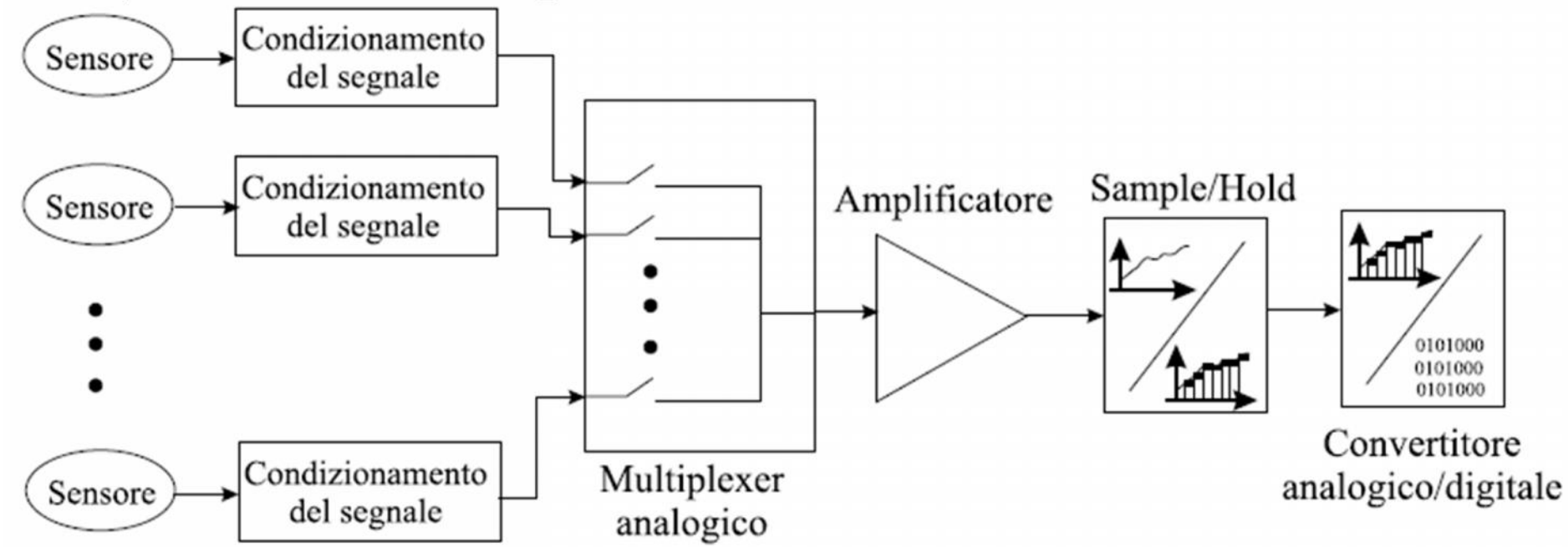
C.so Unione Sovietica 490 (TO)

Materiale didattico:

www.istitutoprimolevi.gov.it

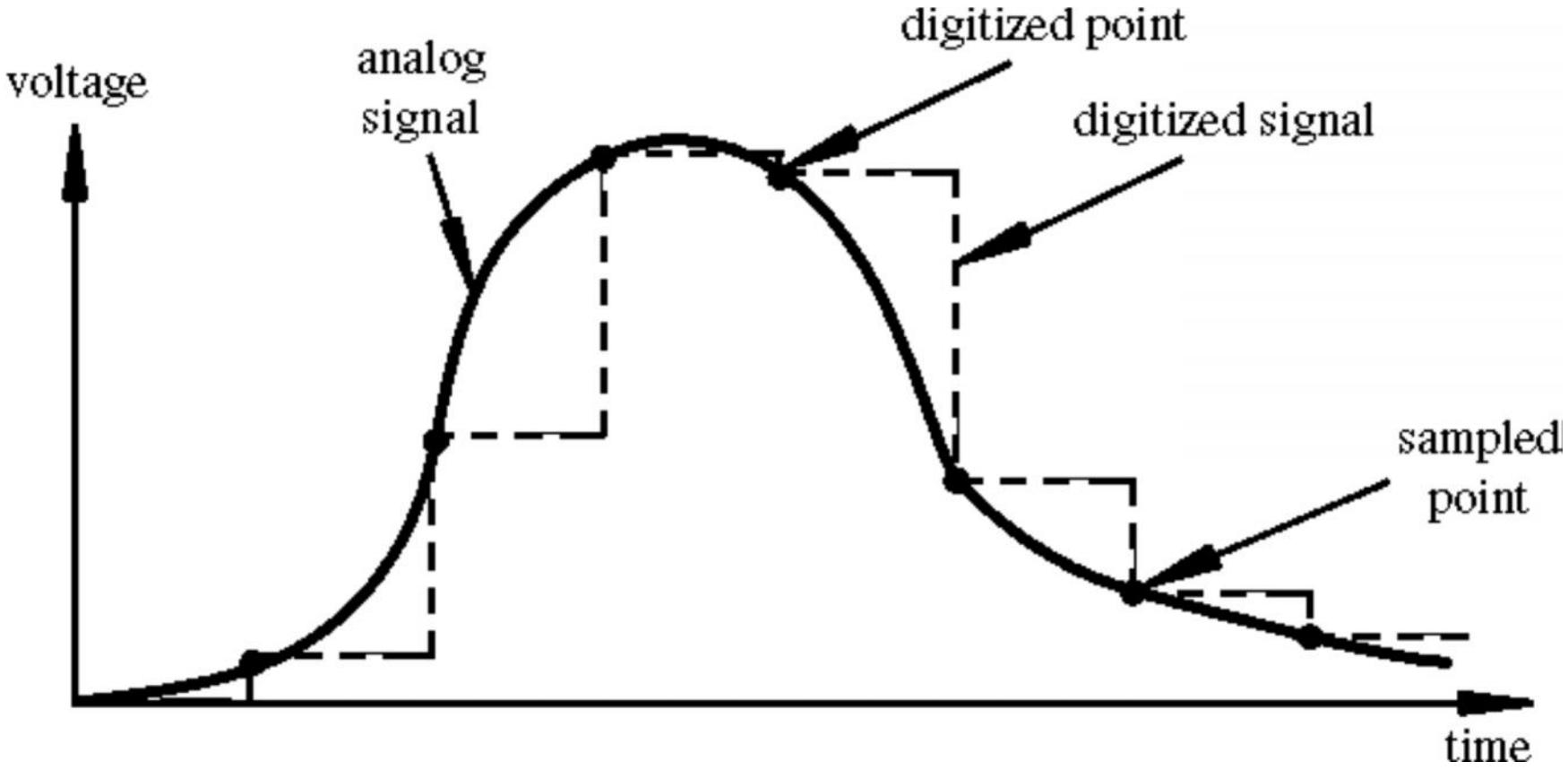
Acquisizione di segnali analogici per l'elaborazione digitale

- Il segnale generato dai trasduttori in genere non è idoneo per la diretta elaborazione da parte del sistema di elaborazione digitale che realizza un algoritmo di controllo, ma occorre interporre una fase di **trattamento del segnale (CONDIZIONAMENTO)** in cui vengono effettuate tutte le operazioni necessarie alla corretta acquisizione del segnale.



Conversione in digitale delle tensioni analogiche

- La **conversione** della tensione da **analogico** a **digitale** (**ADC**) provoca la perdita di alcune informazioni. Nella figura sono stati convertiti (digitalized point) solo alcuni punti della forma d'onda, ovvero nei punti di campionamento (sample point).



Conversione digitale delle tensioni analogiche (ADC) e viceversa (DAC)

- Esistono molti valori per ogni singolo punto campionato e non solo un **livello alto** o **basso**
- Il numero dei valori o stati è definito come la "**risoluzione della conversione analogica**"
- I valori normali più utilizzati sono:
  - **2^8 = 256 (da 0 a 255 stati)**
  - **2^{10} = 1024 (da 0 a 1023 stati) (arduino uno)**
  - **2^{12} = 4096 (da 0 a 4095 stati) (arduino due)**
  - **2^{16} = 65538 (da 0 a 65537 stati)**

Conversione digitale delle tensioni analogiche

- **Arduino UNO** dispone di **6 ingressi analogici**, selezionabili da software, che risultano collegati all'unico convertitore **ADC** (**ADC = convertitore analogico-digitale**) disponibile nel microcontrollore **ATmega328**
- Sono in grado di leggere una tensione analogica compresa tra **0V** (**V_{min}**) e **5V** (**V_{max}**)
- La risoluzione dell'ADC è di **10-bit** (**2^{10}** = **1024 stati**)
- In altre parole, la tensione più piccola riconosciuta dal bit meno significativo che il convertitore (**Arduino UNO**) riesce a discriminare vale:

$V_{LSB} = (V_{max} - V_{min}) / 1024 = (5 - 0) / 1024 =$ 4,88 mV

mentre per **Arduino DUE** [**12-bit** (**2^{12}** = **4096 stati**)] si avrà:

$V_{LSB} = (V_{max} - V_{min}) / 4096 = (3,3 - 0) / 4096 =$ 0,805 mV

che corrisponde alla più piccola variazione di tensione che è possibile misurare o discriminare.

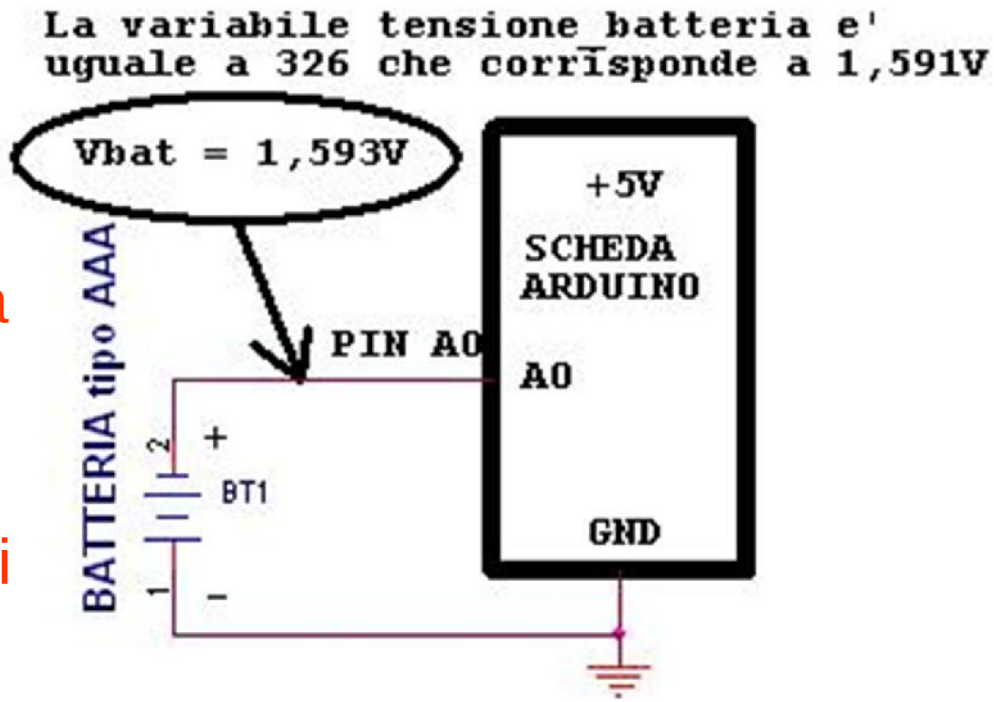
Come determinare il corrispondente valore digitale di una tensione analogica?

Ad esempio: se applico una tensione sul pin analogico A0 della scheda Arduino UNO prelevata da una batteria del tipo AAA (la tensione è stata misurata con un tester digitale che fornisce la seguente indicazione: **Vbat = 1,593V**) a cui corrisponde un valore decimale di **326** nella variabile denominata “tensione_batteria”

ATTENZIONE!

Collegare il polo “**+**” **all’input analogico** e il polo “**-**” a **GND** (non invertire la polarità della batteria per non distruggere la scheda Arduino).

Non collegare tensioni maggiori di +5V o inferiori a 0V (GND) sugli input analogici e/o digitali.



Come determinare il corrispondente valore digitale di una tensione analogica?

Le istruzioni per acquisire la tensione applicata sull'input analogico sono le seguenti:

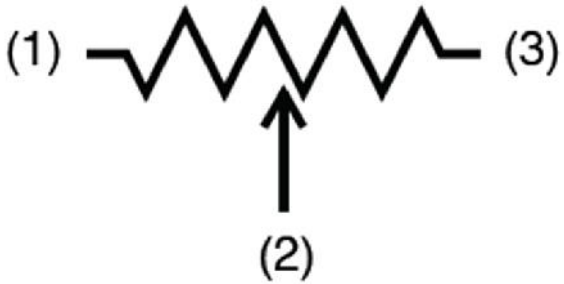
```
// variabile assegnata al valore della tensione di batteria
int tensione_batteria;
// acquisisci il valore di tensione presente sull'input analogico A0
tensione_batteria = analogRead(0);
```

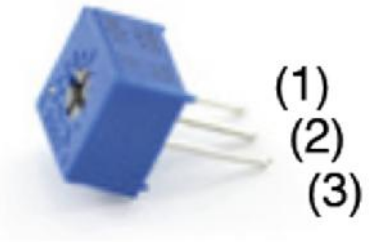
Dopo l'esecuzione delle precedenti istruzioni si avrà che la variabile denominata "**tensione_batteria**" è uguale al numero intero decimale **326**.

$$tensione\_batteria = \frac{1{,}593}{\frac{5-0}{2^{10}}} = 326{,}2464$$

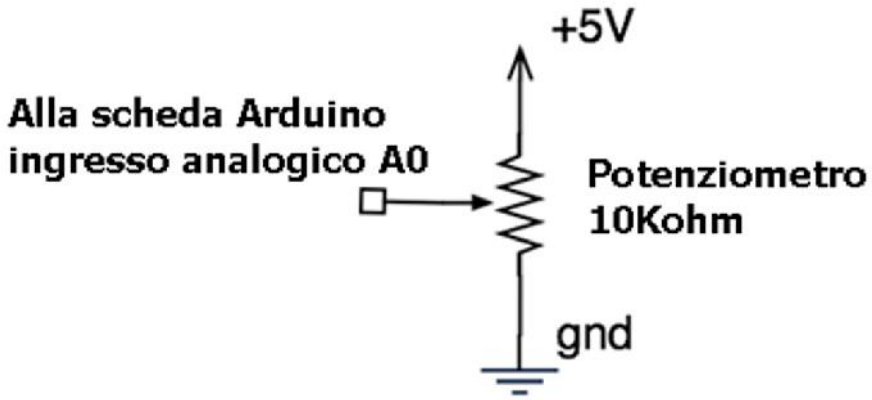
Che viene troncato al numero intero **326**

Potenziometro





Ma come fare per inserire una tensione variabile su uno dei 6 ingressi analogici?



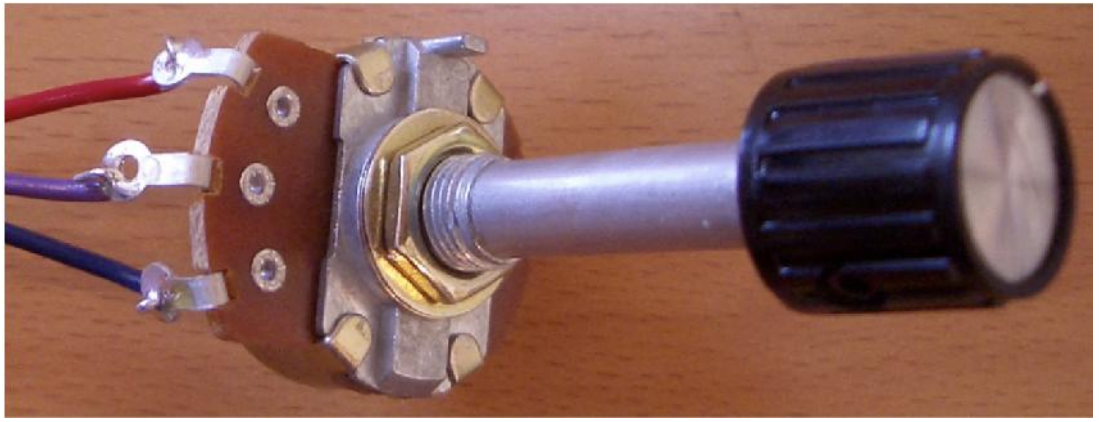
Trimmer multigiri

Utilizzare un potenziometro o trimmer del valore di 10KΩ.

Cavo rosso = +5V (VCC)

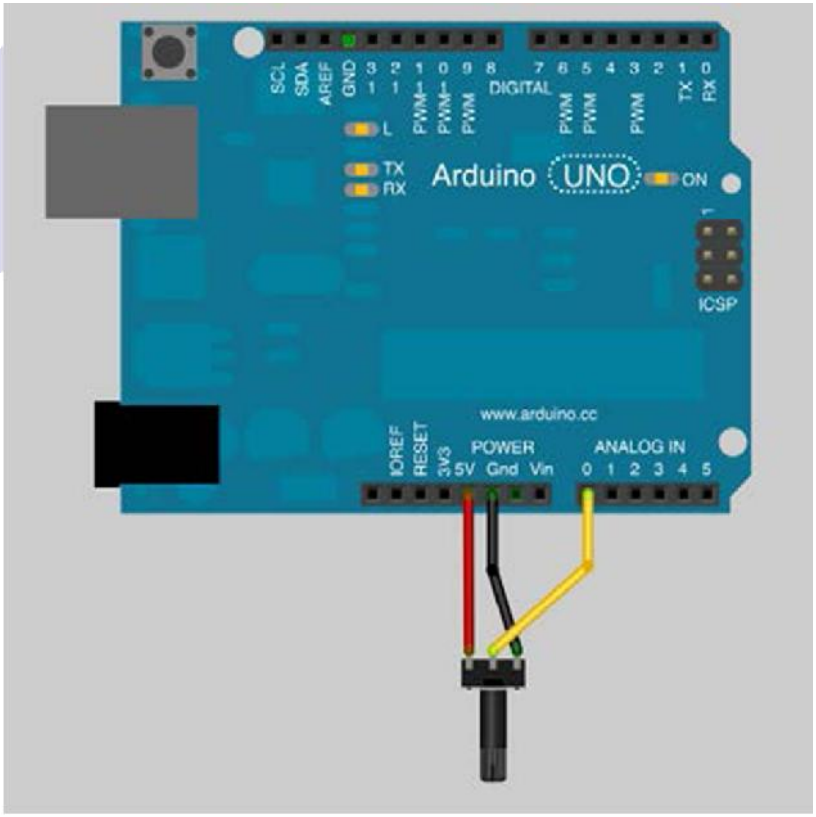
Cavo viola = al pin A0

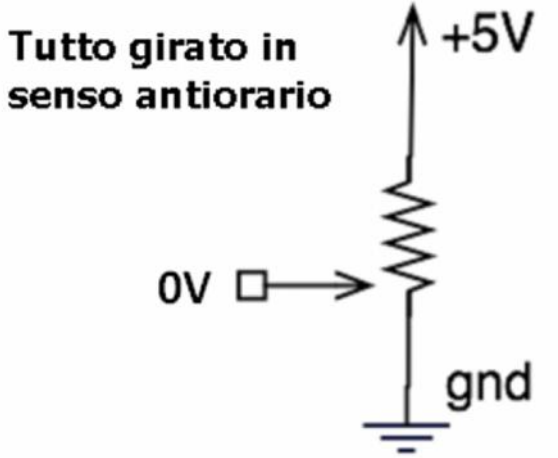
Cavo blu = 0V (GND)

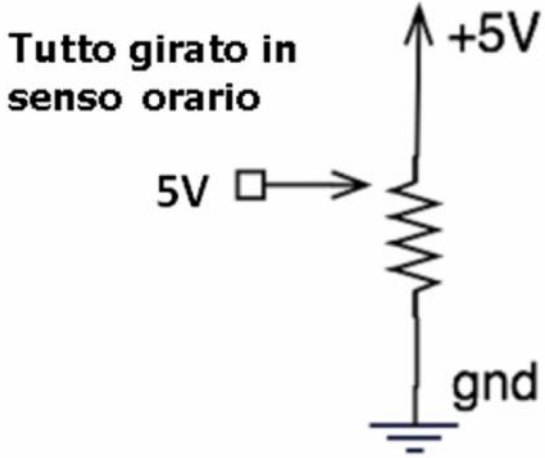


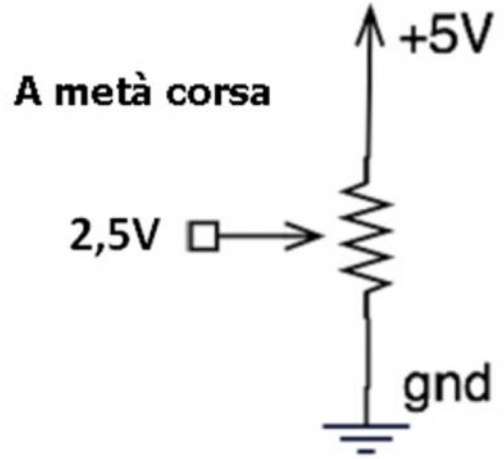
Potenziometro

Ruotando la manopola verso destra o sinistra si ha la possibilità di regolare la tensione presente sul piedino centrale del potenziometro (cursore) tra un valore minimo di **0V (GND)** e un valore massimo di **+5V (VCC).**









Potenziometri e reostati

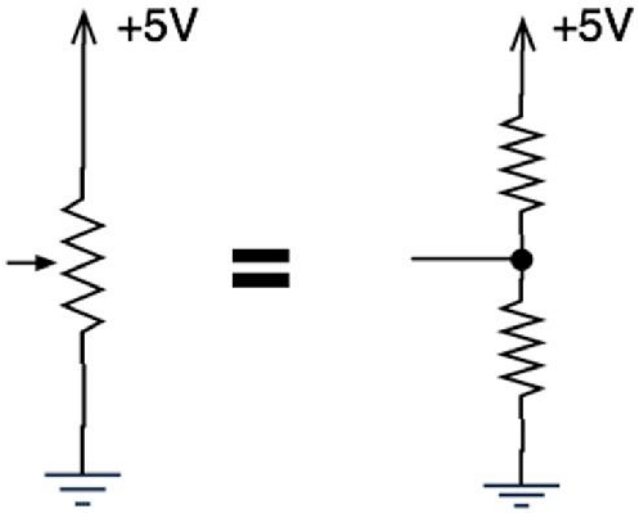
Quando conviene utilizzare i potenziometri?

- Ogniqualvolta occorre fornire una serie di valori compresi tra 0 e 1023 (tra 0 e 4095 per la scheda Arduino DUE) con una semplice rotazione dell'albero cursore.
- Per misurare l'angolo di rotazione dell'alberino (circa 270° di escursione)
- L'alberino può essere collegato per rilevare il movimento di una rotella, pendolo, ecc.

Potenziometri e reostati

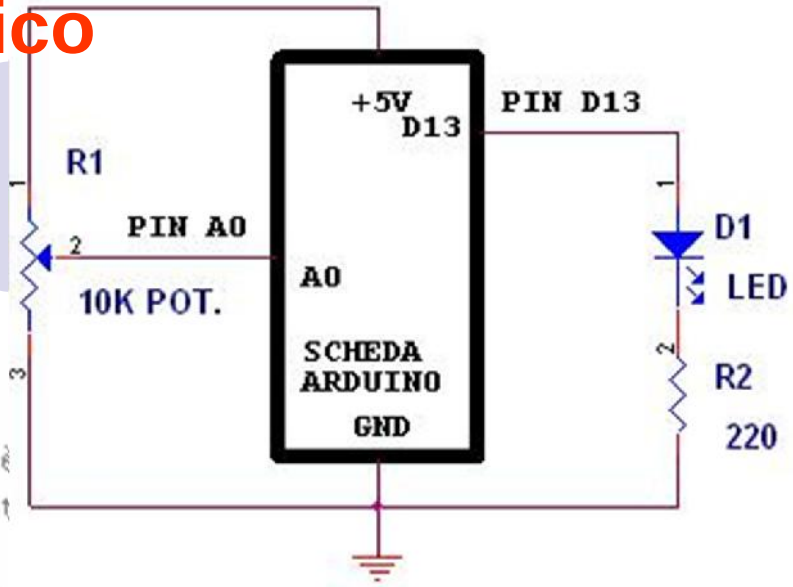
• I potenziometri sono un semplice esempio di partitore di tensione

• La tensione viene suddivisa in due se la l'alberino si trova esattamente a metà, ma può essere modificata semplicemente variando la rotazione dell'alberino.



Software per l'ingresso analogico

- Girare lentamente la manopola per modificare la velocità di lampeggio del led.
- Notare la mancanza della funzione **pinMode()** dentro **setup()** per l'input analogico




```
/*  I.I.S. Primo LEVI - Torino
  Esercizio N. 4
  Progetto: AnalogReadSerial_1      Autore: Questo
  Descrizione: Lettura di un input analogico con pot
  collegato al pin A0)       Data: 28/01/2012 */
int ritardo;  // variabile utilizzata per il ritardo
int num_input_analogico = 0; // numero pin utilizzato per leggere l'input analogico
int lettura_potenziometro; // variabile utilizzata per la lettura dell'input analogico
void setup()   // funzione di inizializzazione della seriale RS232
{
  pinMode(13, OUTPUT); // inizializza il pin 13 della sche
}
void loop()    // programma principale (main) --> ciclo infinito (loop)
{ // acquisisci il valore dell'input analogico pin 0 nella variabile "lettura_potenziometro"
  int lettura_potenziometro = analogRead(num_input_analogico);

  digitalWrite(13, HIGH);   // accendi il LED forzando un livello ALTO sul pin 13
  delay(lettura_potenziometro);  // funzione di ritardo con tempo modificato
                                 // dalla lettura della tensione applicata al pin A0
  digitalWrite(13, LOW);    // spegni il LED forzando un livello BASSO sul pin 13
  delay(lettura_potenziometro);  // funzione di ritardo con tempo modificato
                                 // dalla lettura della tensione applicata al pin A0
}
```

AnalogReadSerial_1.ino

Ma quanto tempo si impiega per acquisire un ingresso analogico?

Misura_tempo_conversione_A_D.ino

```
/*      nome progetto:  Misura della durata del TEMPO Di CONVERSIONE impiegato
            dal convertitore A/D interno alla scheda Arduino espresso in microsecondi
        Specifiche del progetto:
 Il candidato determini il tempo necessario alla conversione dell'A/D a 10-bit interno alla scheda Arduino.
 Il risultato dovra' essere espresso in microsecondi e visualizzato sul monitor del PC.
        creato da:      G. Carpignano           data:   01/02/2015
        compilatore:    compilato con la scheda Arduino ver. 1.0.5
 */
 int valore, canale;
 unsigned long start_1, stop_1, durata;
void setup()
{
  Serial.begin(9600); // settaggio del baud rate della seriale: 9600 baud
}
void loop()
{
  for (int x=3; x>0; x--)
  {
    Serial.print("Tempo di attesa (in secondi) prima della conversione A/D: ");
    Serial.println(x);
    delay(1000);
  }
  // si ricorda che il tempo della prima conversione risulta maggiore dei successivi.
  // Si consideri sempre che il tempo medio della conversione di 116 microsecondi e'
  // dovuto anche alla somma dei seguenti eventi:
  // 1) memorizzazione del tempo di START in una variabile di tipo "unsigned long" (4 byte) nella memoria
  // 2) settaggio del tempo dovuto al multiplexer analogico interno che deve spostarsi sul canale prescelto
  // 3) tempo di conversione reale necessario per ottenere memorizzato il risultato della conversione A/D
```

Ma quanto tempo si impiega per acquisire un ingresso analogico?

```
// 4) memorizzazione del tempo di STOP in una variabile di tipo "unsigned long" (4 byte) nella memoria
  start_1 = micros(); // memorizza il valore iniziale prima che l'evento conversione venga attivato
  valore = analogRead(canale); // aquisisci la tensione presente sull'input analogico A0
  stop_1 = micros(); // memorizza il valore finale dopo che l'evento conversione e' terminato
  // esegui la sottrazione per determinare la durata dell'evento della conversione.
  durata = stop_1 - start_1;
  Serial.print("Tempo della conversione [microsec]: ");
  Serial.println(durata); // visualizza la durata in microsec. sul PC
  delay(2000); // pausa di 2 secondi
  Serial.print("Valore della conversione: ");
  Serial.print(valore);
  Serial.print("  --> acquisito sul canale: ");
  Serial.println(canale);
  Serial.println();
  canale++;  // incremento unitario del num. canale
  if (canale == 6) // se e' stato raggiunto il num. max di canali
    canale=0; // riparti dal primo canale
}
```

Misura_tempo_conversione_A_D.ino

Sensori ottici

I sensori di luce convertono la luce in una corrente, tensione o frequenza che possono poi essere ulteriormente elaborate.

Fotoresistore (LDR)

Questi sono disponibili in una varietà di disegni e materiali. Alcuni tipi ben noti e popolari sono basate sul cadmio, un metallo pesante tossico, che è vietata secondo le ultime linee guida dell'elettronica (RoHS).
I vantaggi dei LDR sono la loro semplicità, robustezza e abbastanza grande gamma dinamica di KΩ al MΩ). Il maggiore svantaggio è il valore molto basso di reazione da pochi millisecondi al minuto. La conversione da corrente a tensione viene eseguita attraverso un semplice partitore di tensione.

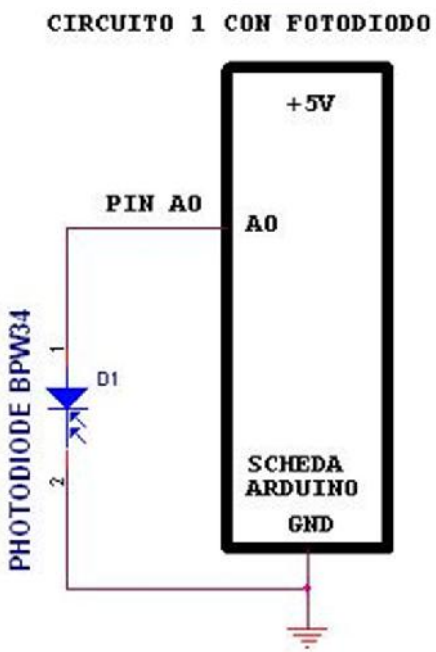
Fotodiodo

Questi sono molto veloci (~10 nanosecondi di commutazione), ma forniscono solo piccolissime correnti (nA o «A). Per la misura dell'illuminamento ci sono tipi di diodi / transistor il cui colore e sensibilità sono molto simili alla percezione umana.

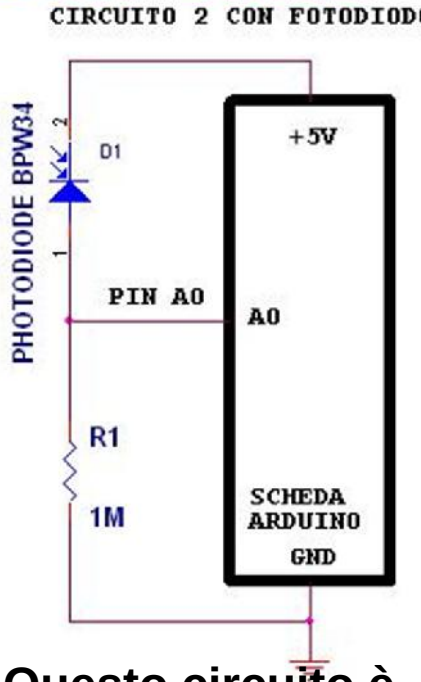
Fototransistor

I tempi di commutazione sono nella gamma del microsecondo, quindi considerevolmente più lento dei fotodiodi, ma consentono il controllo o la commutazione di correnti relativamente più grandi (µA o mA). A seconda del circuito può essere raggiunto una frequenza di taglio di qualche decina di KHz.

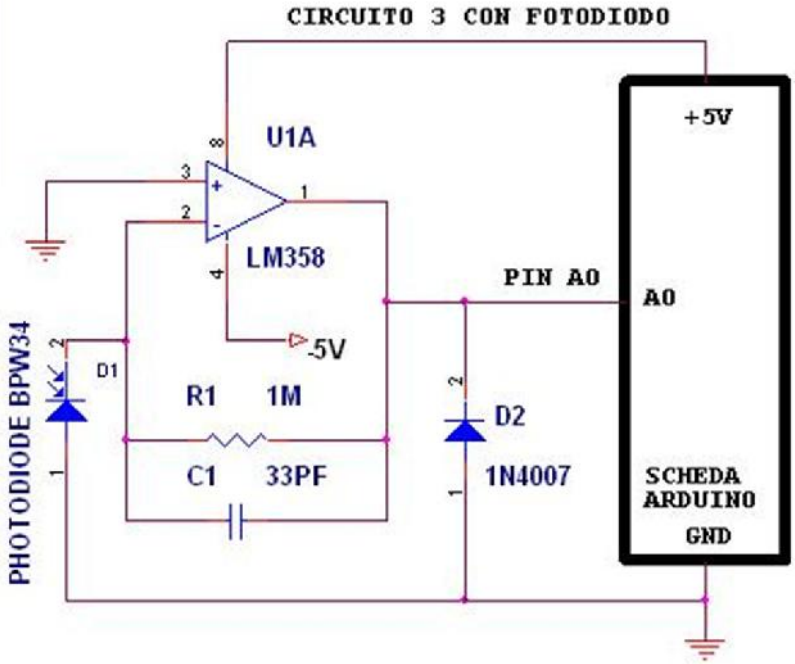
Sensori ottici - Fotodiodo



Il fotodiodo opera come una cella solare. Questo circuito è usato raramente. La tensione di uscita è il logaritmo della luce incidente e corrisponde alla caratteristica diodo inverso (~ 0,7 V). E' molto dipendente dalla temperatura.



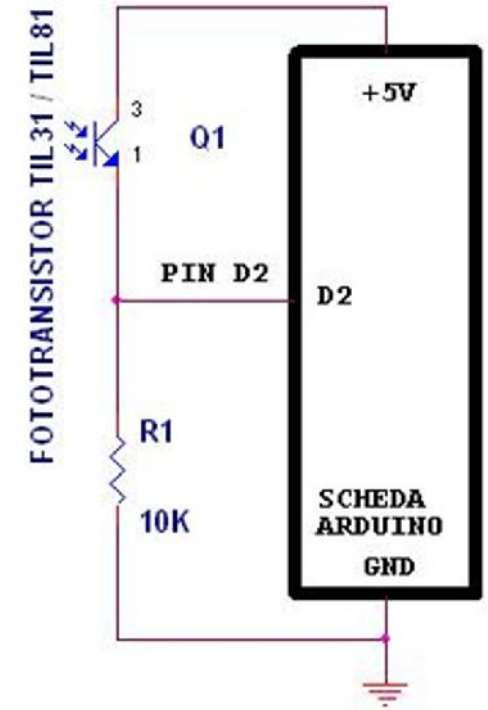
Questo circuito è molto semplice e utilizza un divisore di tensione avente una resistenza relativamente alta. La tensione di uscita è linearmente proporzionale alla potenza luminosa, tuttavia, è abbastanza lento (< 8KHz) .



Questo circuito è veloce (<72KHz). Utilizza un amplificatore operazionale con ingressi JFET quindi con una resistenza molto elevata. Richiede una alimentazione duale +5V e -5V ed il diodo D2 è posto a protezione dell'input analogico A0.

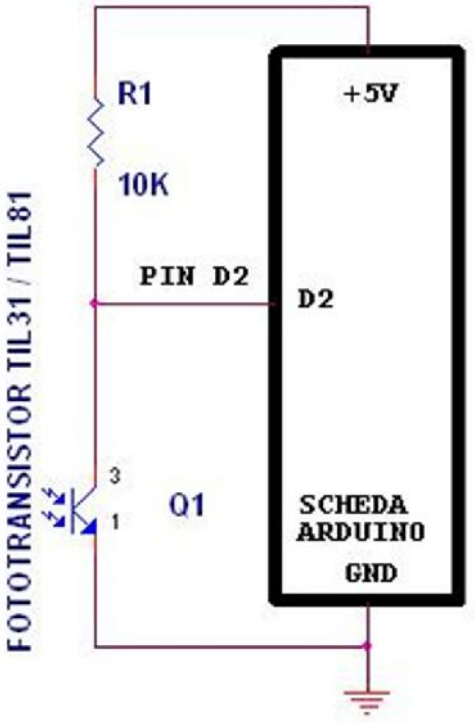
Sensori ottici - Fototransistor

CIRCUITO 1 CON FOTOTRANSISTOR



Il fototransistor Q1 lavora come un interruttore controllato dalla luce.
Nel circuito 1, se Q1 è colpito da un fascio di luce si avrà la saturazione del transistor che determina una tensione Vce = V31 = 0,2V (PIN D2 = HIGH). Al contrario quando il Q1 è oscurato la Vce=V31=5V (PIN D2 = LOW).

CIRCUITO 2 CON FOTOTRANSISTOR



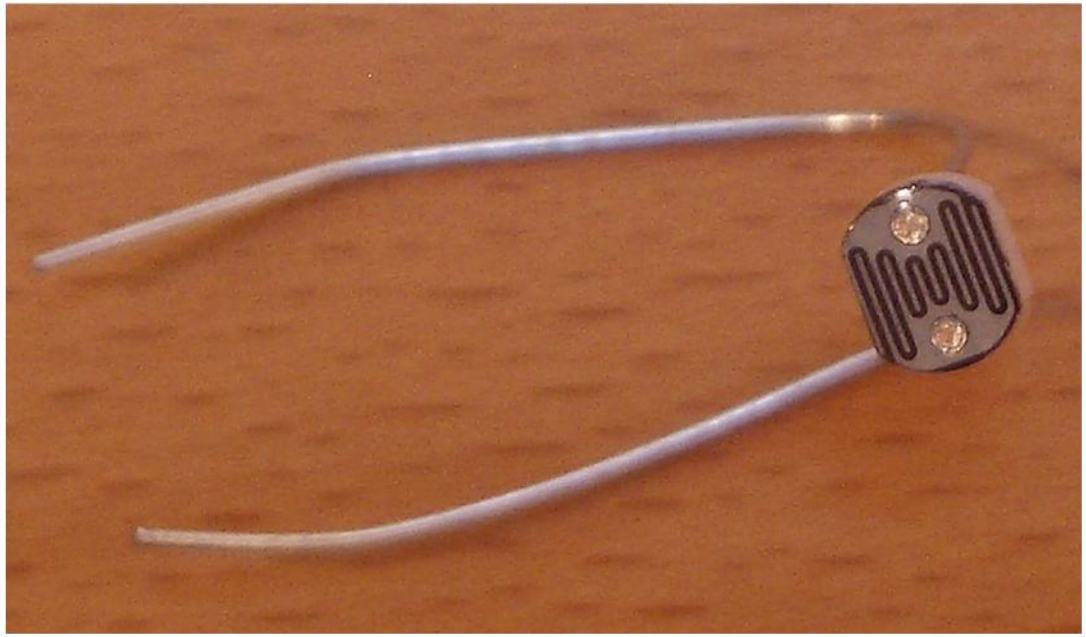
Il fototransistor Q1 lavora come un interruttore controllato dalla luce.
Nel circuito 2, se Q1 è colpito da un fascio di luce si avrà la saturazione del transistor che determina una tensione Vce = V31 = 0,2V (PIN D2 = LOW). Al contrario quando il Q1 è oscurato la Vce=V31=5V (PIN D2 = HIGH).

Sensori ottici - Fotoresistore

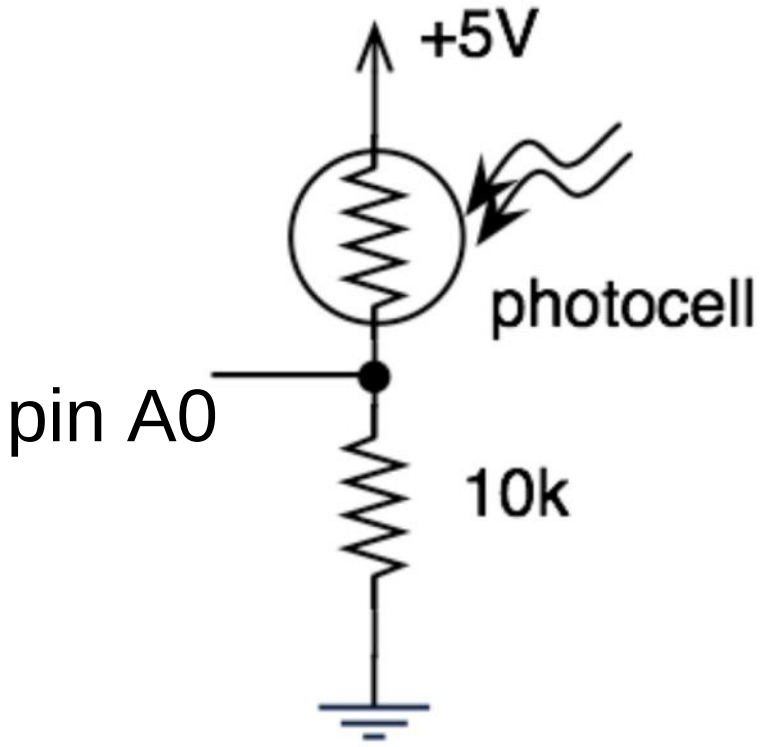
- La fotoresistenza è una resistenza il cui valore è dipendente dalla luce incidente.

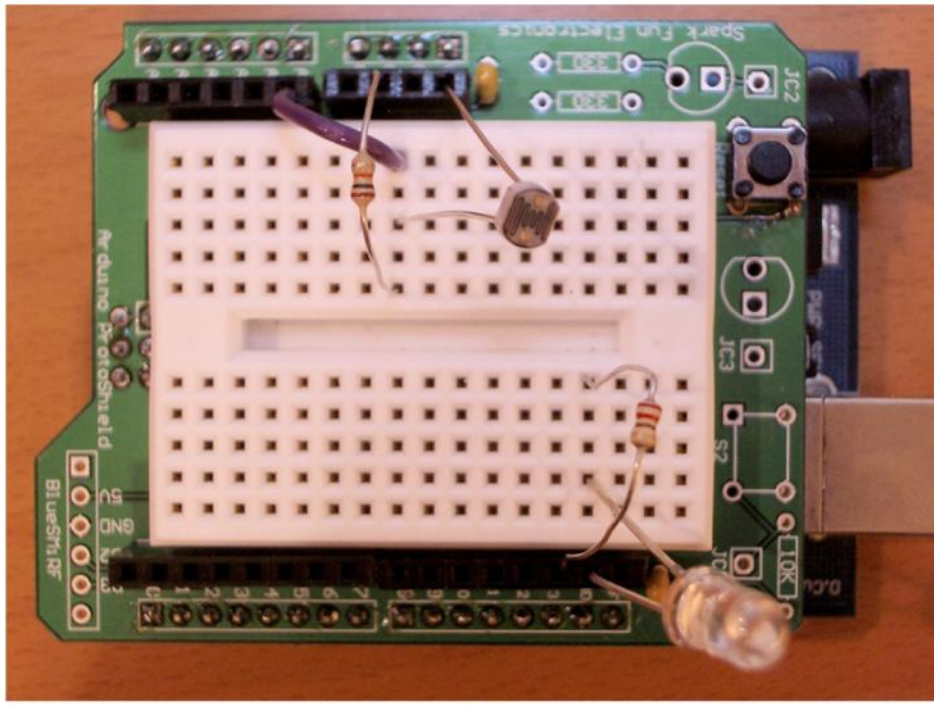
Nei due casi estremi:

- Fotoresistenza **completamente oscurata** (al buio totale) offre una resistenza **> 1MΩ**
- Fotoresistenza **esposta alla luce solare diretta** offre una resistenza **< 1000 Ω**



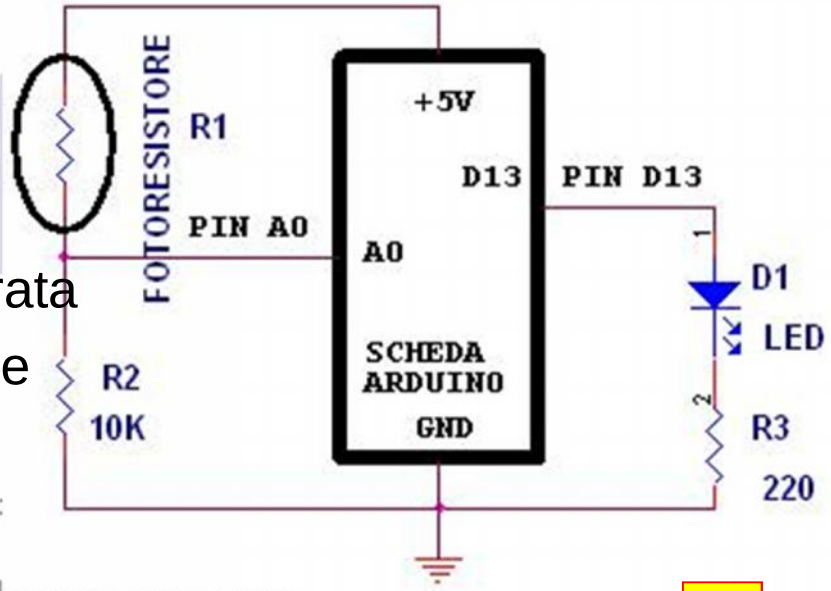
Circuito con fotoresistenza applicata alla scheda Arduino Uno





Light frequency modulator

- Lampeggio veloce → fotoresistenza oscurata
- Lampeggio lento → fotoresistenza alla luce



AnalogReadSerial_2.ino

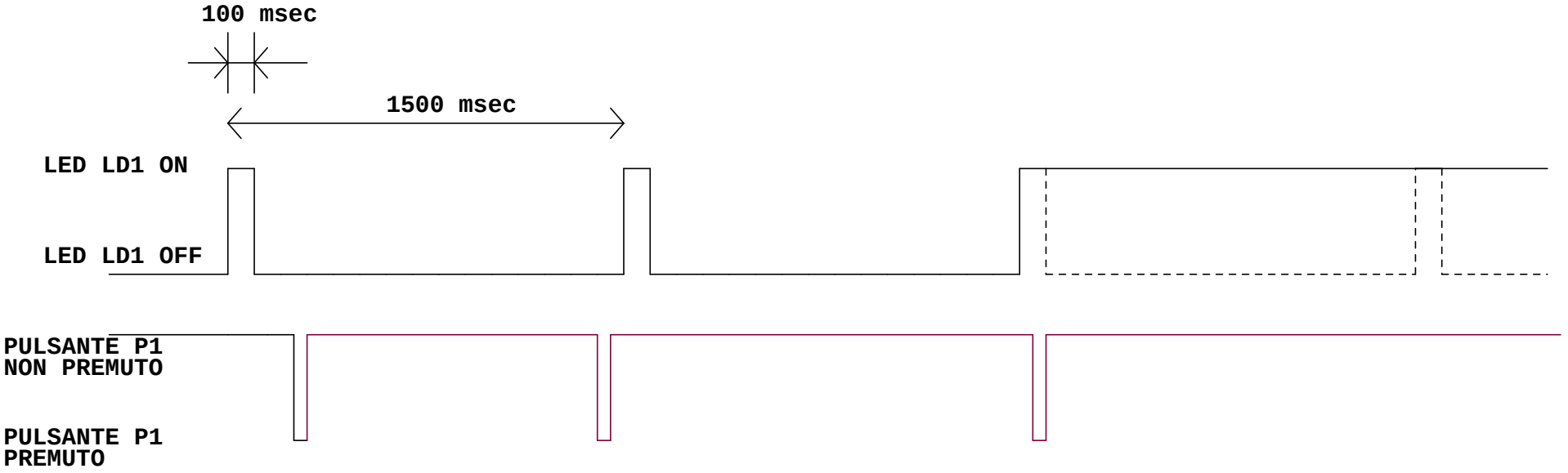
```
/*  I.I.S. Primo LEVI - Torino
  Esercizio N. 5
  Progetto: AnalogReadSerial_2      Autore: Questo è un esemp
  Descrizione: Lettura di un input analogico collegato ad un
  fotoresistore collegato a +5V e la resistenza da 10Kohm collegata verso GND.
  Il centro del partitore e' collegato all'input analogico A0       Data: 29/01/2012 */
int ritardo;  // variabile utilizzata per il ritardo
int num_input_analogico = 0; // numero pin utilizzato per leggere l'input analogico
int lettura_fotoresistore; // variabile utilizzata per la lettura dell'input analogico
void setup()   // funzione di inizializzazione della seriale RS232
{
  pinMode(13, OUTPUT); // inizializza il pin 13 della scheda Arduino come OUTPUT (LED)
}
void loop()    // programma principale (main) --> ciclo infinito (loop)
{ // acquisisci il valore dell'input analogico pin 0 nella variabile "lettura_fotoresistore"
  // se il fotoresistore e' esposto alla luce si ottengono valori elevati di tensione sul pin A0,
  // quindi valori alti della variabile "lettura_fotoresistore", al contrario se il fotoresistore
  // e' al buio si ottengono valori bassi nella variabile.
  int lettura_fotoresistore = analogRead(num_input_analogico);
  digitalWrite(13, HIGH);   // accendi il LED forzando un livello ALTO sul pin 13
  delay(lettura_fotoresistore);  // funzione di ritardo con tempo modificato se fotoresistore alla luce/buio
  digitalWrite(13, LOW);    // spegni il LED forzando un livello BASSO sul pin 13
  delay(lettura_fotoresistore);  // funzione di ritardo con tempo modificato se fotoresistore alla luce/buio
}
```

Calcolo del tempo? (game)

La ditta Claviere Toys richiede la progettazione di un circuito in grado di effettuare dei ragazzi.

Per la realizzazione un test sulla capacità di valutare le sequenze temporali si utilizzi la scheda Arduino **per far lampeggiare un led denominato LD1 ad esempio ogni 1,5 secondi**.

- Premendo un pulsante di tipo n.a. (denominato P1) al momento giusto il led LD1 rimarrà acceso.
- Il led rimane acceso solo per 100 msec e poiché il tempo di reazione della persona si aggira intorno ai 250 msec, quindi non è possibile "**congelare**" il led mentre è acceso se non dopo svariati tentativi di prova.



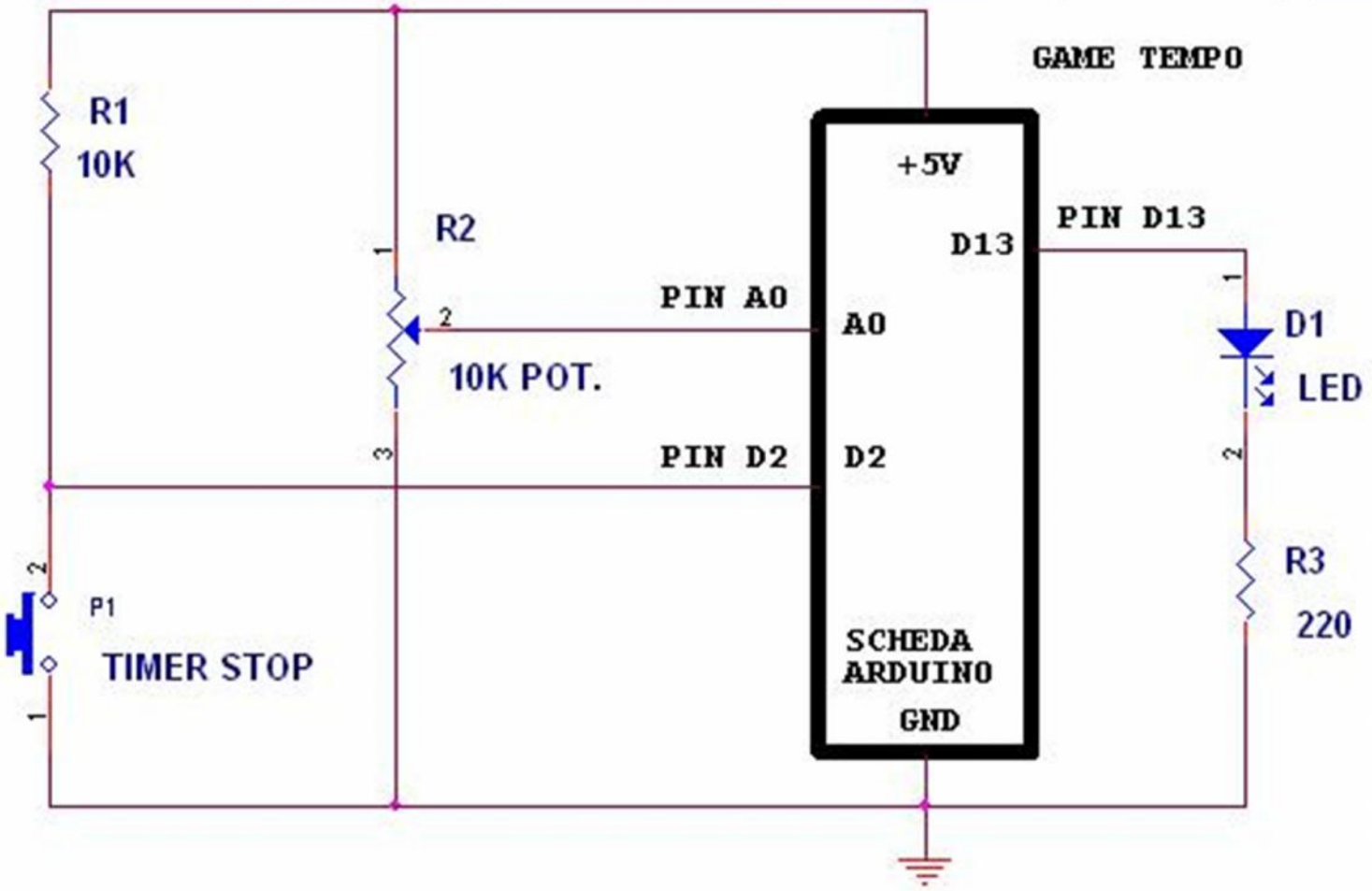
Calcolo del tempo? (game)

Prima di premere il pulsante è quindi necessario valutare, usandolo come riferimento, il tempo che è passato dallo spegnimento del led.

Il candidato, formulando di volta in volta le ipotesi aggiuntive che ritiene necessarie:

- effettui una regolazione dell'intervallo compreso tra 0,2 e 16,484 secondi corrispondente alla tensione presente sul cursore di un potenziometro (denominato R2) per regolare la durata dell'intervallo di tempo.
- disegni uno schema a blocchi del progetto completo ricordando di porre particolare attenzione agli input e output.
- disegni lo schema elettrico dei collegamenti dimensionando i componenti.
- Si implementi il software in linguaggio C per la scheda Arduino in modo da leggere l'input per gestire l'output necessari secondo quanto richiesto in precedenza dal testo.
- Ogni volta che si riesce a "congelare" il led sullo stato accesso, tale situazione permane in modo permanente fino alla pressione del pulsante di reset del microcontrollore.

Calcolo del tempo? (game)



Schema elettrico dei collegamenti da effettuare con la scheda Arduino

Calcolo del tempo? (gestione del game con interrupt)

Calcolo_del_tempo_trascorso.ino

Parte 1ª

```
/*  I.I.S. Primo LEVI - Torino - Lab. Sistemi Classe 4BN
  Progetto: Calcolo_del_tempo_trascorso      Autore: G. Carpi
  Data: 15/04/2010 */
int ritardo;  // variabile utilizzata per il ritardo
int input_trimmer = 0; // numero pin utilizzato per leggere l'input analogico collegat  da 10K
int lettura_trimmer; // variabile utilizzata per la lettura dell'input analogico (trimmer)
int pulsante = 2;  // pin collegato al pulsante di tipo N.A. tramite una R di pull-up
int ldl = 13; // pin collegato al LED
/*************************************************************************************/
void setup()    // funzione di inizializzazione della seriale RS232
{
 // definisci il numero di Interrupt = 0, definisci il nome della funzione di gestione
 // dell'interrupt = int0_ISR, definisci che ogni volta che il pin2 (INT0) passa dal
 // livello logico HIGH al livello logico LOW genera un interrupt
  attachInterrupt(0, int0_ISR, FALLING);
  pinMode(ldl, OUTPUT); // inizializza il pin 13 della scheda Arduino come OUTPUT (LED1)
  pinMode(pulsante, INPUT); // inizializza il pin 2 della scheda Arduino come INPUT (pulsante)
  digitalWrite(pulsante, HIGH);    // attiva la resistenza interna di pullup
  digitalWrite(ldl, LOW);    // spegni il led
}
/*************************************************************************************/
void loop()     // programma principale (main) --> ciclo infinito (loop)
{ // acquisisci il valore dell'input analogico nella variabile "lettura_trimmer"
  // se il trimmer e' circa a meta' corsa sul pin A0 si ottiene una tensione di circa 2,5V,
  // che verrà convertita in un valore circa a meta' scala tra 0 e 1023 dei possibili valori.
  int lettura_trimmer = analogRead(input_trimmer); // lettura del cursore del trimmer
  // il range richiesto si ottiene con il cursore del trimmer tutto girato in senso antiorario (0,2 sec)
  // o in senso orario (16,484 sec.). Si ricorda che 16,384 sec. e' uguale 16384 msec. Inoltre volendo
```

Calcolo del tempo? (game)

```
  // considerare l'esempio di 16,484 sec. sec. (15484 msec) si ottiene che il led rimarra' spento (Toff)
  // per 16384 msec e sara' acceso (Ton) solo per 100 msec.
  // Quindi il T = Toff(variabile dalla posizione del trimmer) + Ton(fisso a 100 msec)
  // Pero' il convertitore A/D fornisce un valore compreso tra 0 e 1023 a cui deve corrispondere
  // un tempo compreso tra 200 e 16484 msec. Occorre effettuare il seguente calcolo della
  // costante moltiplicativa: (16484 - 100) / 1024 = 16
  lettura_trimmer = lettura_trimmer * 16; // calcolo per adeguamento alla scala del tempo
  digitalWrite(ledl, LOW);    // spegni il ledl  forzando un livello BASSO sul pin 13
  delay(lettura_trimmer);  // funzione di ritardo con tempo modificato in funzione del trimmer
  digitalWrite(ledl, HIGH);   // accendi il ledl forzando un livello ALTO sul pin 13
  delay(100);  // funzione di ritardo coincidente con 0,1 sec. = 100 msec
}
/*****************************************************************************/
// funzione di gestione dell'interrupt collegato al pin 2 (INT0).
// Questa funzione viene richiamata ogni volta
// che il pin 2 passa dal livello logico "HIGH" al livello logico "LOW" ovvero ogni
// volta che viene premuto il pulsante n.a. collegato al pin 2
void int0_ISR(void)
{
  if (digitalRead(ledl) == 1)  // controlla il ledl se e' ACCESO
    while (1); // blocca il programma se il ledl e' acceso.
    // N.B. Ricorda di premere il tasto di RESET per uscire dal loop infinito
  else
   delay(5); // ritardo per eliminare i rimbalzi dei contatti sul pulsante
}
/*****************************************************************************/
```

Calcolo_del_tempo_trascorso.ino

Parte 2ª

Tensione analogica in uscita

Il complemento della funzione di lettura in ingresso di una tensione analogica **«analogRead()»**, è la funzione **«analogWrite()»** che permette di fornire una **tensione analogica in uscita** compresa tra 0 e 5V.

Questa funzione, se applicata al circuito dei led, permette di modificare la loro luminosità da un livello minimo (led spento → 0V) fino ad un livello massimo (led completamente acceso → 5V) utilizzando una scala di $2^8 = 256$ valori differenti.

La tensione analogica PWM **«Pulse Width Modulation»** in uscita dalla scheda Arduino è disponibile solo sui pin digitali: **3**, **5**, **6**, **9**, **10** e **11**.

Pulse Width Modulation

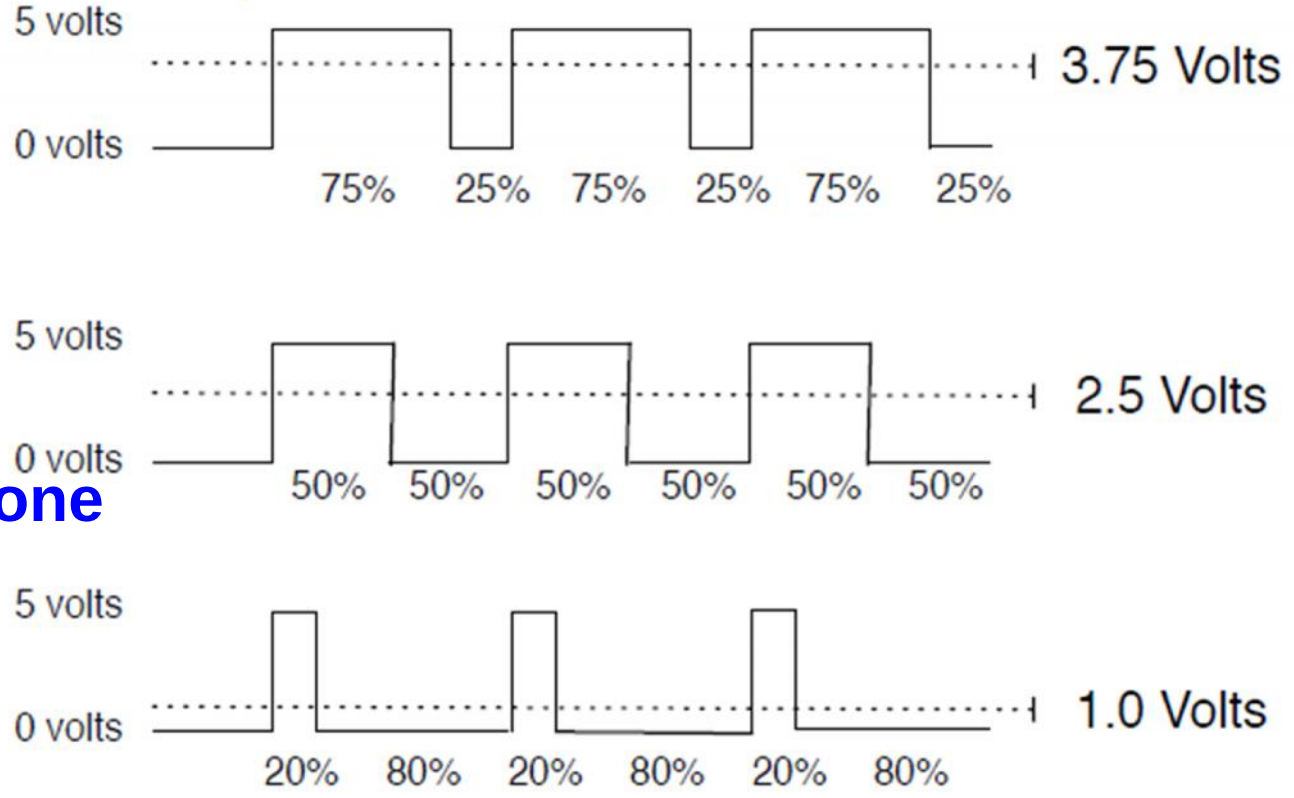
- Modulazione a larghezza di impulso: molto comunemente è chiamato "**PWM**“
- Il microcontrollore della scheda Arduino UNO non può fornire tensioni analogiche (disponibili con appositi integrati denominati **DAC = Digital Analog Converter**), ma solamente delle tensioni di tipo digitale (**0** volt oppure **5** volt).
- Comunque utilizzando una onda quadra con un opportuno **duty cycle** (rapporto tra la semi-onda positiva fratto il tempo totale dell’onda) permette di raggiungere la stessa potenza di una corrispondente tensione continua applicata al carico.

PWM

La tensione di uscita è la media del tempo che rimane a livello alto (**Ton**) fratto il tempo totale (**Ttot**), dove **Ttot = Ton + Toff** (livello logico basso)

Voutput = (Ton / Ttot) * Vmax

Dove:
Ton = semiperiodo positivo
Toff = semiperiodo negativo
Vmax = max. tensione di uscita



PWM

È usato per:

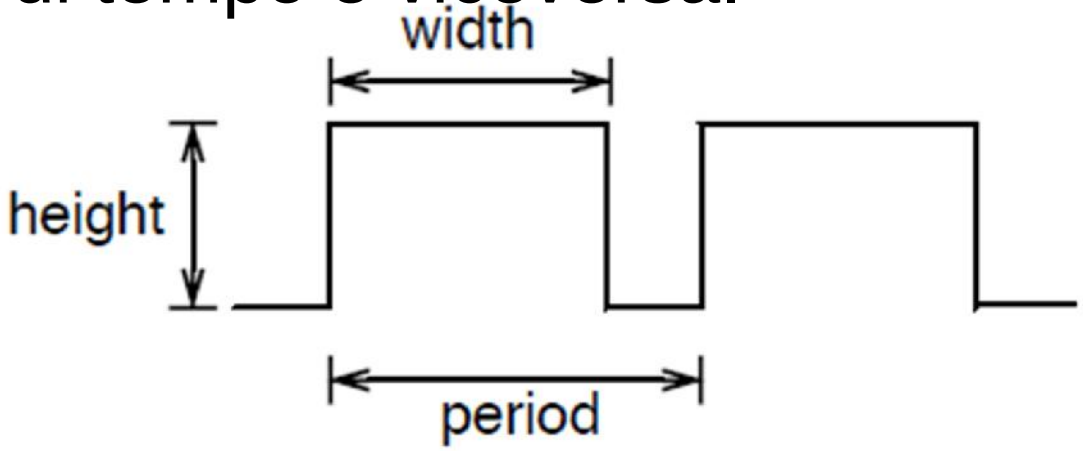
- Dimmer (regolazione luminosità) per le lampade, controllo della velocità dei motori, alimentatori.

Alcune caratteristiche dei segnali PWM

- **Il tempo totale** (**Ttot = Ton + Toff**) **non varia mai**.
- Anche la frequenza non varia
- La tensione di uscita può assumere solo due livelli: **HIGH** e **LOW** (5V e GND)

Per mantenere invariata la frequenza si dovrà:

- se il **Ton aumenta** il **Toff deve diminuire** della stessa quantità di tempo e viceversa.



PWM

Arduino UNO è dotato di PWM su **sei** pin:

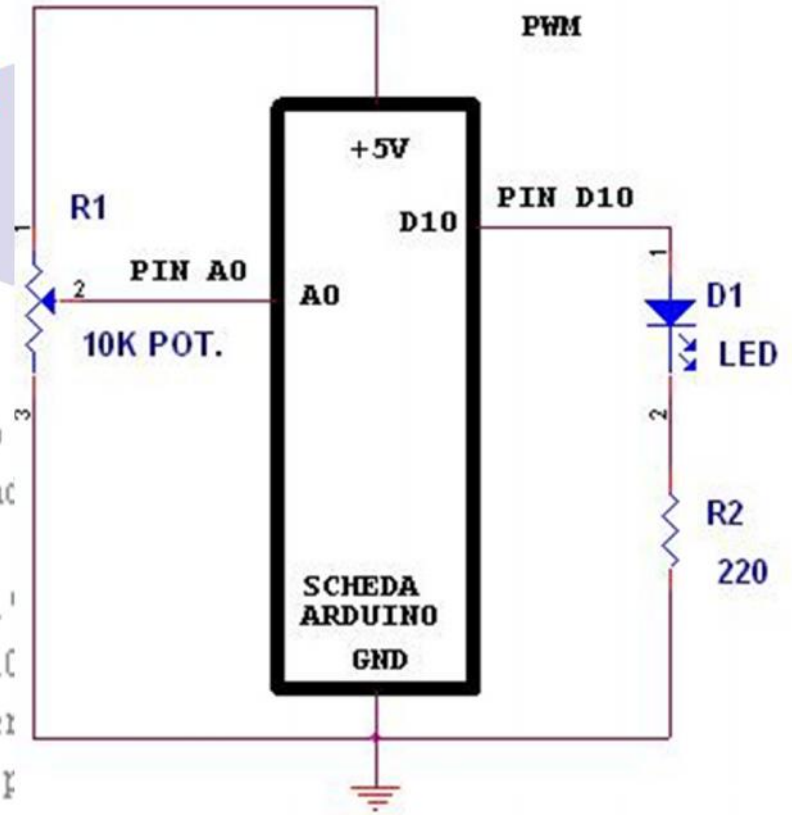
3, 5, 6, 9, 10 e 11

L'istruzione «**analogWrite(pin, valore)**» lavora ad una frequenza elevata ma fissa (quindi non è utilizzabile per i servi motori) ma più che adeguata per i LED e per i motori.

Utilizza un apposito hardware per i circuiti PWM integrati nel chip ATmega328, senza bisogno di altro software.

La frequenza del PWM utilizzata nell'istruzione «**analogWrite()**» è impostata a circa **480 Hz** e **non risulta modificabile**.

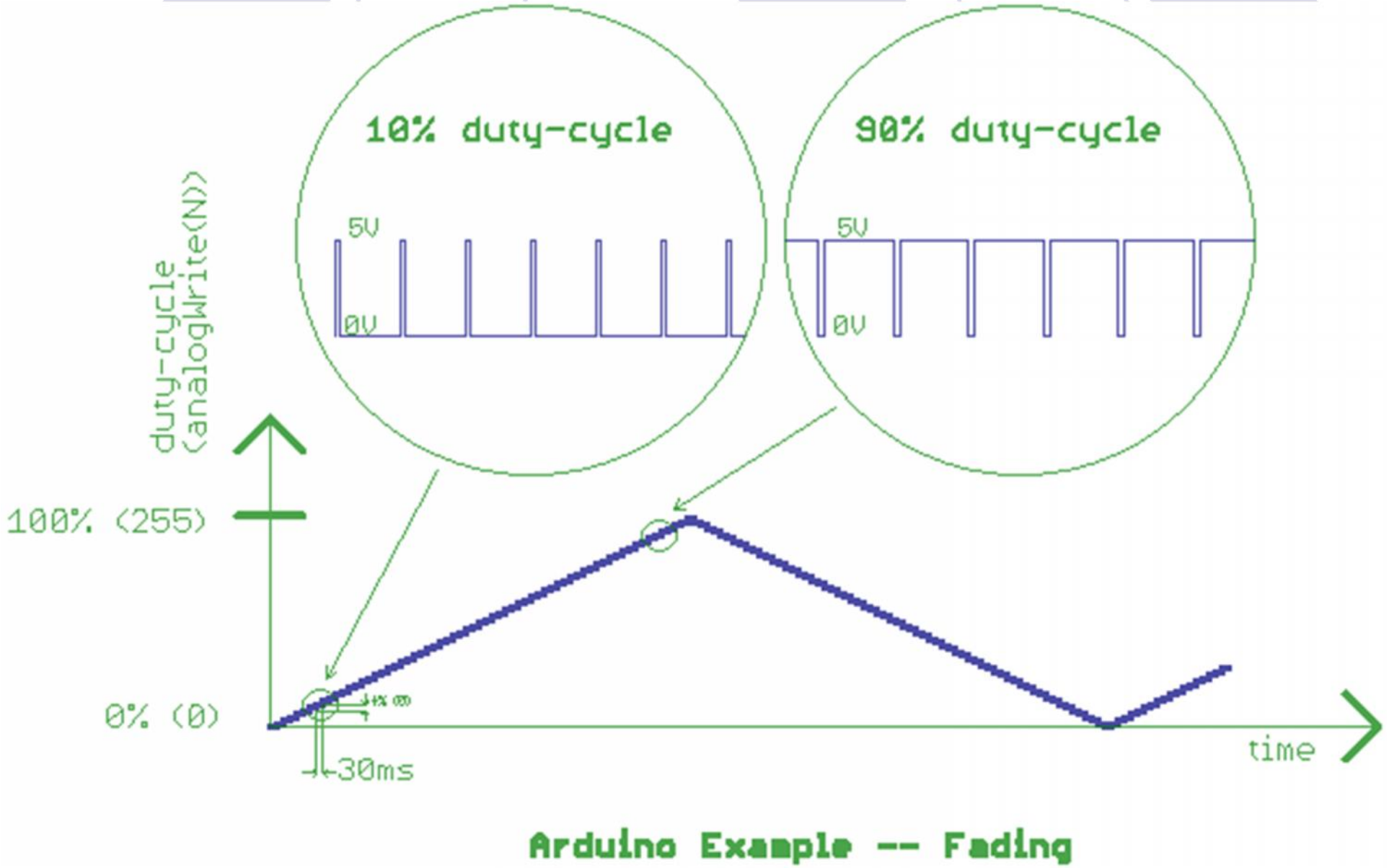
Variazione continua della luminosità di un Led



AnalogRead_analogWrite.ino

```
/*  I.I.S. Primo LEVI - Torino
  Esercizio N. 6
  Progetto: AnalogRead_AnalogWrite_1      Autore: Questo
  Descrizione: Lettura di un input analogico collegato ac
  collegato con gli estremi a +5V e l'altro a GND.
  Il cursore centrale del potenziometro e' collegato all'
  che controlla la luminosità del led collegato al pin 1(
int num_input_analogico = 0; // numero pin utilizzato per
int num_output_analogico = 10; // numero pin utilizzato p
void setup()    // funzione di inizializzazione
{
  pinMode(10, OUTPUT); // inizializza il pin 10 della scheda Arduino come OUTPUT (LED)
}
void loop()     // programma principale (main) --> ciclo infinito (loop)
{
  // regola la luminosita' del LED leggendo la tensione analogica dal potenziometro (valori compresi
  // tra 0 e 1023, che vengono divisi per 4 in modo da rientrare nel range 0-255)
  analogWrite(num_output_analogico, analogRead(num_input_analogico) / 4);
}
```

PWM

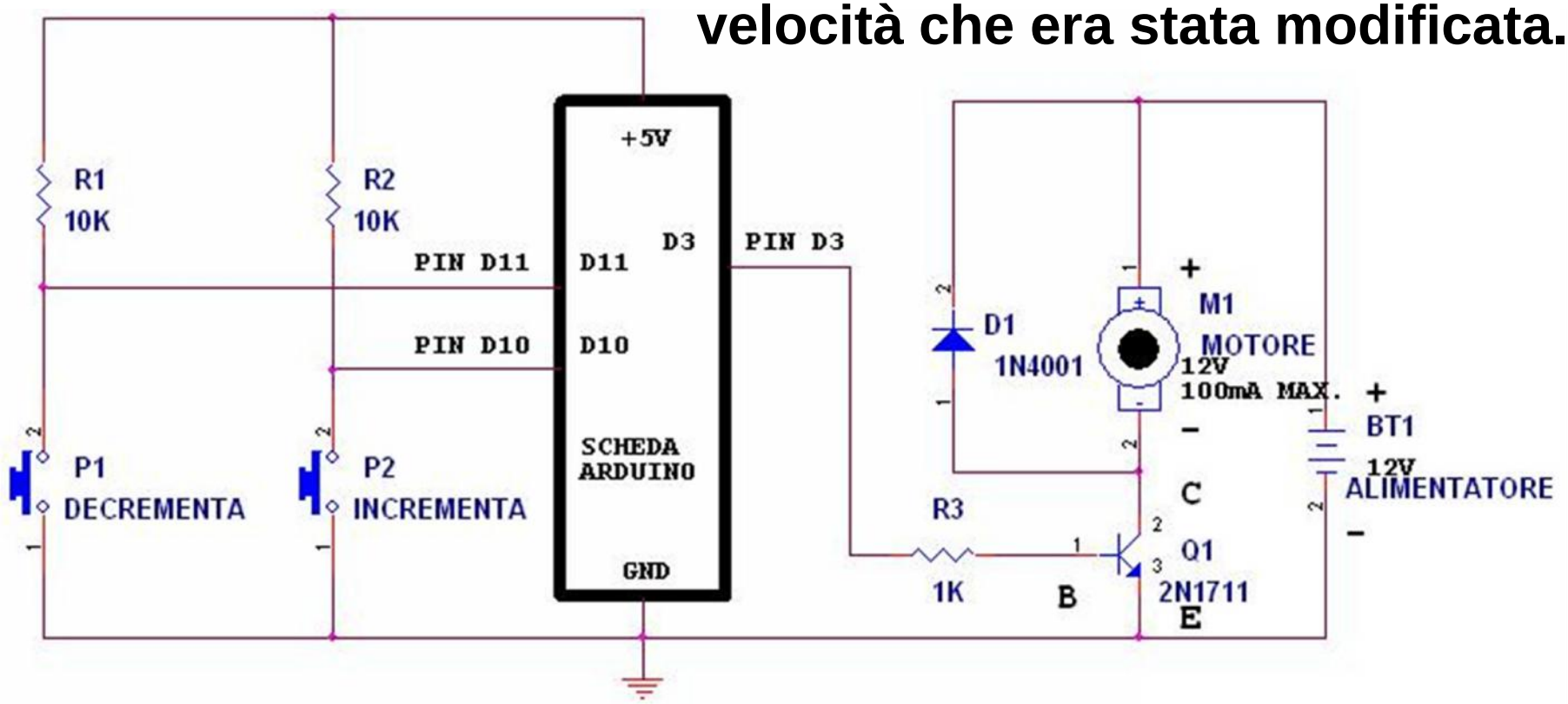


Arduino Example -- Fading

Motore con controllo velocità a due pulsanti

Permette di regolare la velocità di rotazione di un piccolo motore in continua (ad esempio: una ventolina dei computer) tramite la pressione su due differenti pulsanti per incrementare o decrementare la velocità.

Se non vengono premuti i pulsanti il software deve memorizzare e mantenere costante costante la precedente velocità che era stata modificata.



Motore con controllo velocità a due pulsanti

Motore_in_cc_con_2_pulsanti_Up_Down.ino

```
int Val_PWM = 0; // Valore per il PWM (compreso tra 0 e 255)
const int P1 = 11;    // Pulsante decremento (DOWN)
const int P2 = 10;    // Pulsante incremento (UP)
const int MOTORE =  3;  // Motore da 12V c.c.
void setup()
{
  Serial.begin(9600); // inizializza la seriale a 9600 baud rate
  Serial.println("Start del sistema");
  pinMode(P1, INPUT);  // Impostazione Pulsante P1 come ingresso (DOWN)
  pinMode(P2, INPUT);  // Impostazione Pulsante P2 come ingresso (UP)
  pinMode(MOTORE, OUTPUT);  // Impostazione PIN come uscita motore
  digitalWrite(P1, HIGH);  // Abilitazione pull-up
  digitalWrite(P2, HIGH);  // Abilitazione pull-up
}
void loop()
{ // Lettura stato pulsante 1 (Decremento velocità)
  if (digitalRead(P1) == LOW) // se P1 e' premuto
  { // controlla se il valore rientra nel campo tra 0 e 255
    if ((Val_PWM > 0) && (Val_PWM <= 255))
    { // effettua il decremento della velocità
      Val_PWM=Val_PWM-1; // decremento unitario
      Serial.print("Valore PWM: "); // stampa nuovo valore
      Serial.println(Val_PWM, DEC);
    }
  }
  // Lettura stato pulsante 2 (Incremento velocità)
  if (digitalRead(P2) == LOW) // se P2 e' premuto
  { // controlla se il valore rientra nel campo tra 0 e 255
    if ((Val_PWM >= 0) && (Val_PWM < 255))
    { // effettua l'incremento della velocità
      Val_PWM=Val_PWM+1;  // incremento unitario
      Serial.print("Valore PWM: "); // stampa nuovo valore
      Serial.println(Val_PWM, DEC);
    }
  }
  analogWrite(MOTORE, Val_PWM);  // Imposta la nuova velocità
  delay(50);  // Attesa di 50msec.
}
```

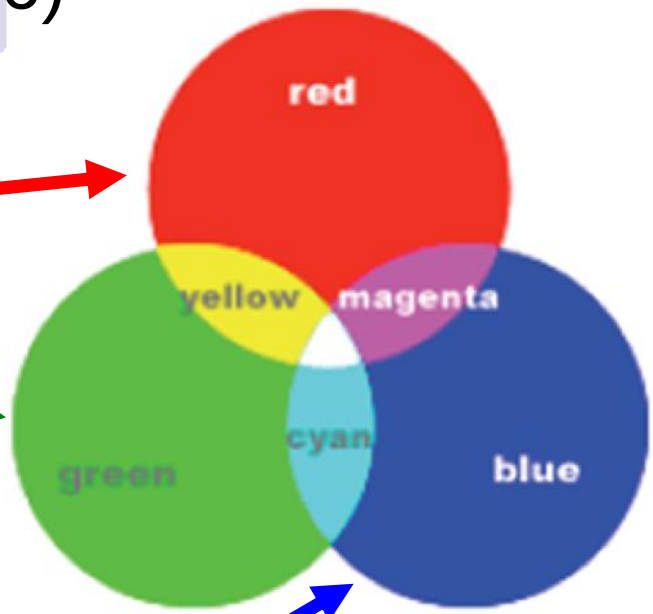
Il LED RGB

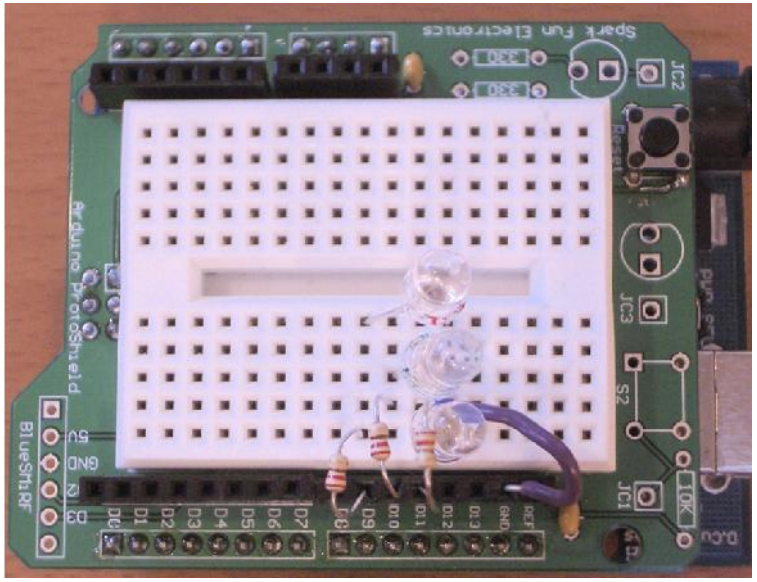
- Con il led **RGB** è possibile ottenere più di **16 milioni di colori** ($2^8 \times 2^8 \times 2^8 = 16.777.216$)
- Si richiedono 3 uscite PWM per i 3 colori primari:

R = RED

G = GREEN

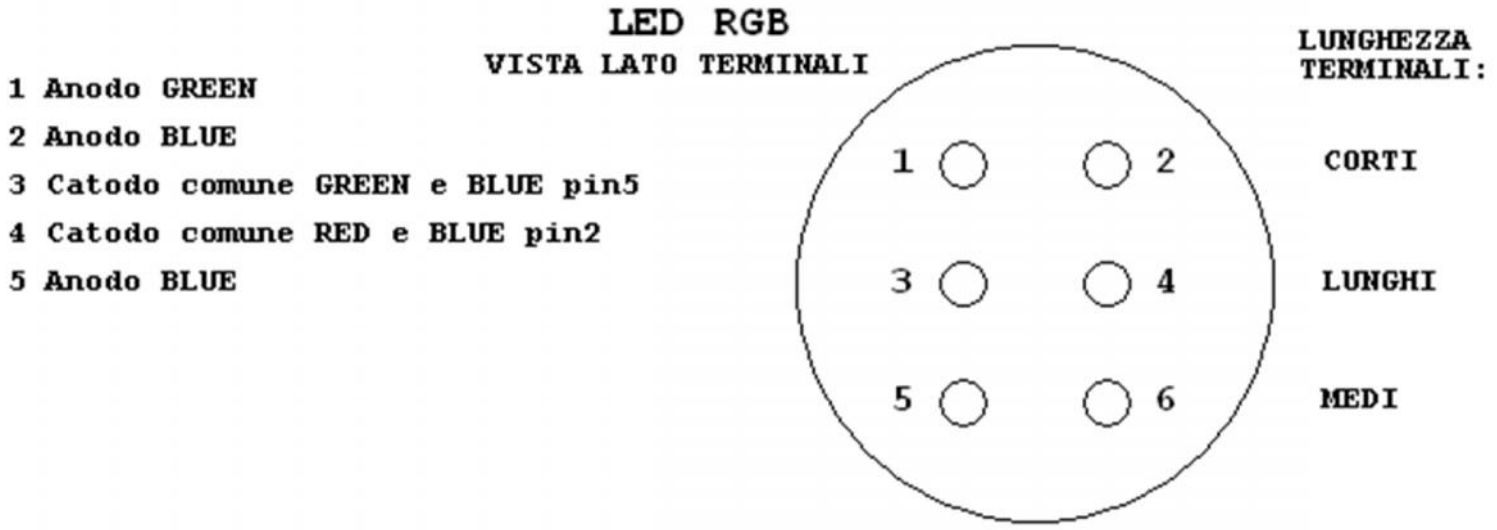
B = BLUE





Il LED RGB

- Caratteristiche del Led RGB della Kingbright LF819EMBGMBC
- $I_{Forward}$ (tipica) = 20mA
- $I_{Forward}$ (max.) = 30mA(rosso) / 25mA(verde) / 30mA(blu)
- $V_{Forward}$ (tipica) = 2V(rosso) / 2,2V(verde) / 4,5V(blu)
- $V_{Reverse}$ (max.) = 5V
- Intensità (min.) = 80mcd(rosso) / 100mcd(verde) / 20mcd(blu)
- Intensità (max.) = 200mcd(rosso) / 200mcd(verde) / 60mcd(blu)
- Ang. di osservazione = 30 gradi
- Lungh. d'onda di picco = 625nm(rosso) / 565nm(verde) / 430nm(blu)
- Temperatura di funzionamento da -40°C a +85°C
- Lunghezza (corpo) 13,5mm - Ø 10mm - Passo terminali 2,54mm

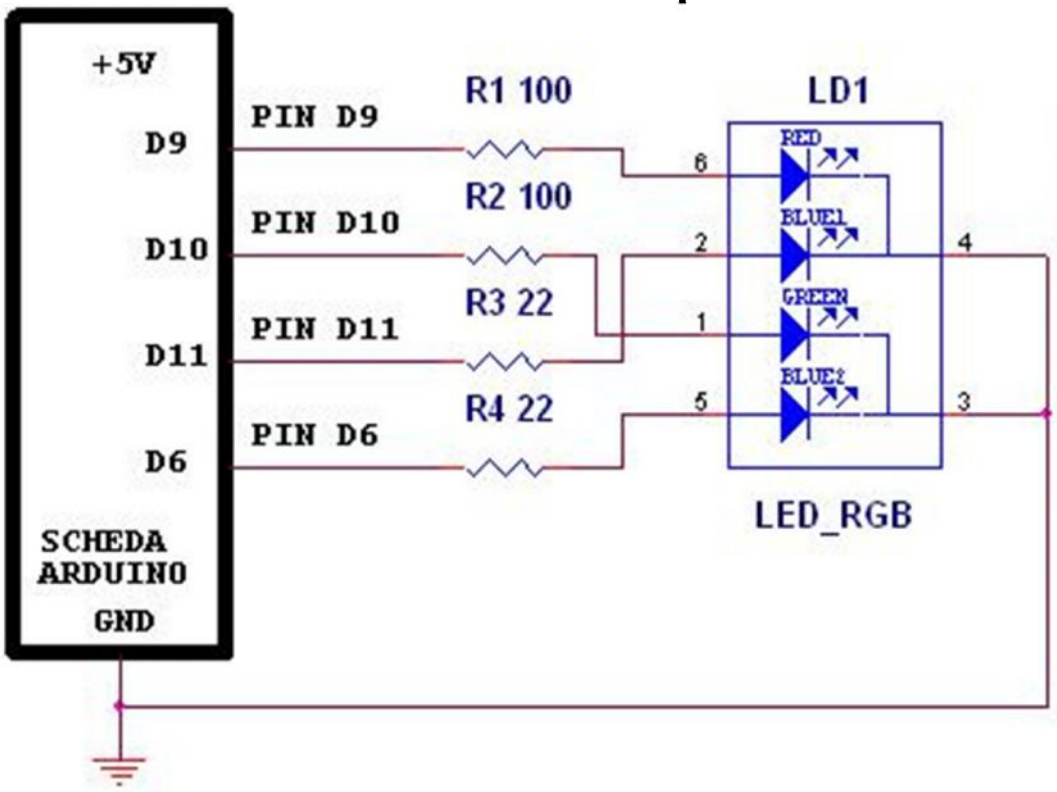


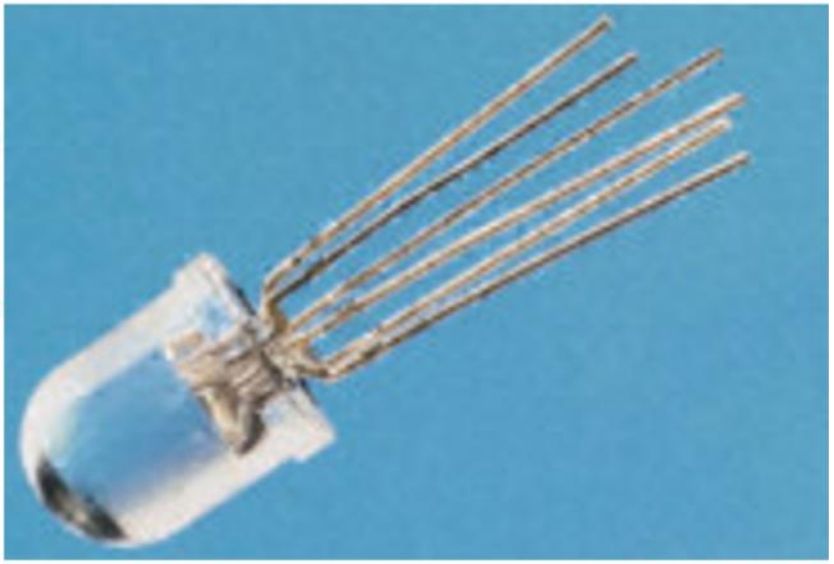
RGB Color Fading

- Il software proposto permette la lenta dissolvenza del colore e la sua miscelazione.

Led_RGB.ino

- Inoltre trasmette anche i valori del colore visualizzato sulla porta seriale.



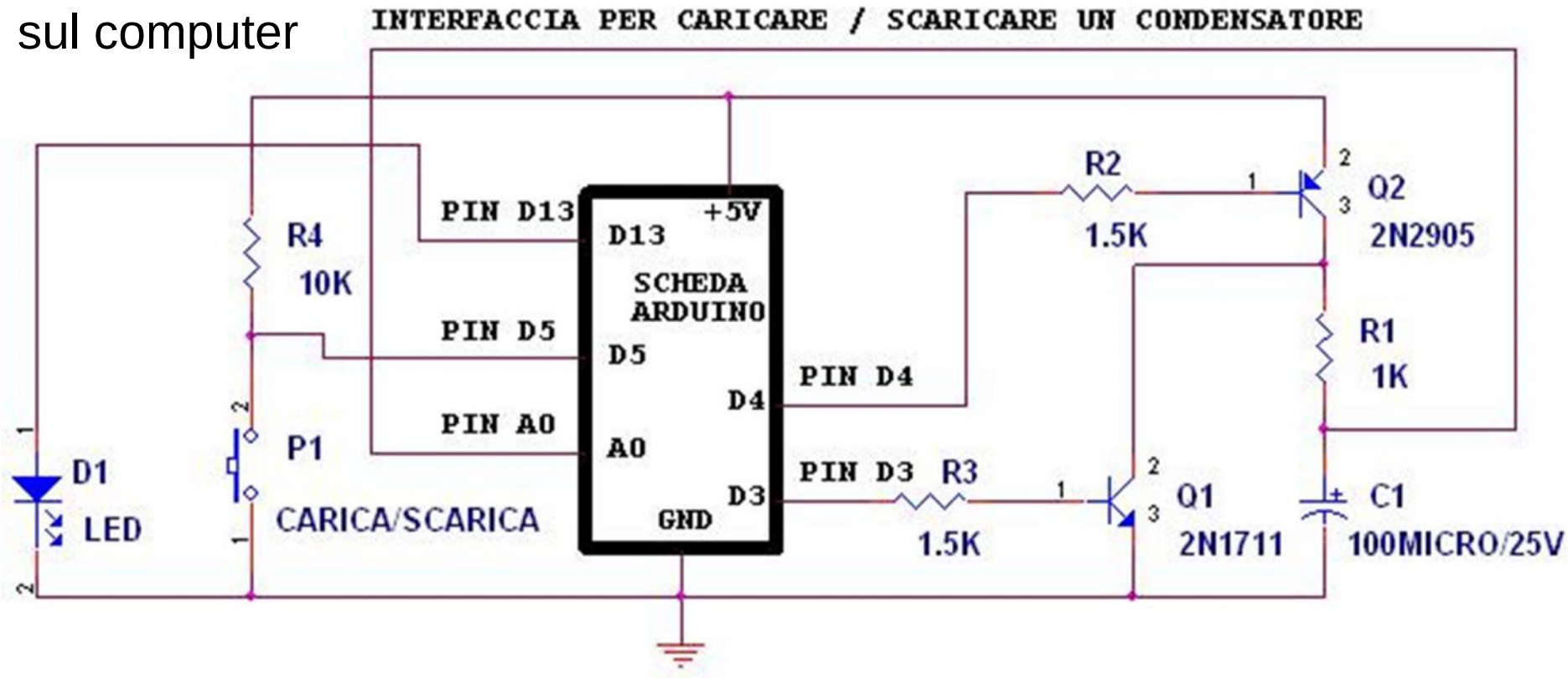


Rilievo della tensione di carica / scarica di un condensatore

Rilievo della tensione di carica e scarica ai capi di un condensatore con trasmissione dei dati sulla seriale del PC con possibilità di visualizzazione dei dati sulla seriale (e loro memorizzazione per ottenere un grafico tramite il software).

È possibile scrivere un programma su Arduino e uno sul computer denominato **Processing** sempre in linguaggio C.

In Arduino: l'acquisizione della tensione di carica / scarica permette di inviare i dati, tramite porta seriale USB, come byte al software presente sul computer



Rilievo della tensione di carica / scarica di un condensatore

La tensione di carica massima del C1 sarà:

$V_{c1} = V_{cc} - V_{ecsatQ2} = 5 - 0,2 =$ **4,8V**

La tensione di scarica minima del C1 sarà:

$V_{c1} = V_{cesatQ1} =$ **0,2V**

	Q2 PNP	Q1 NPN	Descrizione
STAND-BY	**Interdizione (off)**	**Interdizione (off)**	**Condizione di inizio lavoro**
CARICA	**Saturazione (on)**	**Interdizione (off)**	**Fase di carica del condensatore C1**
SCARICA	**Interdizione (off)**	**Saturazione (on)**	**Fase di scarica del condensatore C1**
IMPOSSIBILE	**Saturazione (on)**	**Saturazione (on)**	**ATTENZIONE! Cortocircuito dell'alimentazione**

Tensione_di_carica_scarica_condensatore_nel_tempo_Arduino.ino

Parte 1ª

```
/*  I.I.S. Primo LEVI - Torino - Lab. Sistemi Classe 5BN
  Progetto: Tensione_di_carica_scarica_condensatore_nel_tempo      Autore: G. Carpignano
  Data: 10/05/2013 */
int durata_acquisizione = 100;  // variabile che determina ogni quanti millisecondi occorre acquisire
int input_tensione_c1 = 0; // pin 0 analogico collegato al C1
int lettura_tensione_c1; // variabile utilizzata per la lettura dell'input analogico (ai capi di C1)
int transistor_npn = 3;  // pin 3 configurato come Output, collegato al transistor Q1 (2N1711)
int transistor_pnp = 4;  // pin 4 configurato come Output, collegato al transistor Q2 (2N2905)
int pulsante = 5; // pin 5 pulsante N.A. serve come Start della fase carica e della fase scarica
int led = 13; // pin 13 configurato come Output da collegare al led
long tempo; // variabile utilizzata per calcolare il tempo es
/*****************************************************
void setup()   // funzione di inizializzazione della seriale
{
  pinMode(transistor_npn, OUTPUT); // inizializza l'output del transistor NPN Q1
  pinMode(transistor_pnp, OUTPUT); // inizializza l'output del transistor PNP Q2
  digitalWrite(transistor_npn, LOW); // interdizione del NPN Q1 forzando un livello BASSO
  digitalWrite(transistor_pnp, HIGH); // interdizione del PNP Q2 forzando un livello ALTO
  pinMode(led, OUTPUT); // inizializza il pin 13 come l'output del led
  digitalWrite(led, LOW); // spegni il led forzando un livello BASSO
  pinMode(pulsante, INPUT); // inizializza il pin 5 come l'input del pulsante n.a.
  digitalWrite(pulsante, HIGH); // attiva la R di pullup interna (10K) forzando un livello ALTO
  Serial.begin(115200);  // inizializza la seriale RS232 con 115200 baud, 8 bit dati, nessuna parità
}
/*************************************************************************/
void loop()    // programma principale (main) --> ciclo infinito (loop)
{ // finchè il pulsante non viene premuto continua a verificare il pulsante
  // Serial.print("\nPREMI IL PULSANTE PER INIZIARE LA FASE DI CARICA");
```

Software per Arduino (1ª parte)

Tensione_di_carica_scarica_condensatore_nel_tempo_Arduino.ino **Parte 2ª**

```
while(digitalRead(pulsante) == HIGH) // nell'attessa della pressione del pulsante
 { // fai lampeggiare il led
  digitalWrite(led, HIGH); // accendi il led forzando un livello ALTO
  delay(100);  // ritardo di 100 msec.
  digitalWrite(led, LOW); // spegni il led forzando un livello BASSO
  delay(100);  // ritardo di 100 msec.
 };
// FASE DI CARICA DEL CONDENSATORE
// Serial.print("\nCARICA DEL CONDENSATORE");
digitalWrite(led, HIGH); // accendi il led forzando un livello ALTO
digitalWrite(transistor_npn, LOW); // interdizione del NPN Q1 forzando un livello BASSO
delay(10); // ritardo di 10 msec
digitalWrite(transistor_pnp, LOW); // saturazione del PNP Q2 forzando un livello BASSO
// acquisisci il valore dell'input analogico con un tempo espresso nella variabile
// denominata "durata_acquisizione", espresso in millisecondi
tempo = 0; // reset del tempo
for(int x=0; x < 100; x++) // tempo totale = 100 * 10 = 1000 msec = 1 sec
{
  lettura_tensione_c1 = analogRead(input_tensione_c1); // lettura della tensione ai capi di C1
  tempo = tempo + durata_acquisizione; // aggiorna il tempo incrementandolo
  delay (durata_acquisizione); // ritardo tra una acquisizione e la successiva
  // Serial.print(tempo); // trasmetti il dato tempo
  // Serial.print("\t"); // trasmetti il dato TAB
  Serial.println(lettura_tensione_c1); // trasmetti i dati acquisiti in formato decimale
}
// Serial.print("\nFINE ACQUISIZIONE");
// FASE DI STAND-BY
digitalWrite(transistor_npn, LOW); // interdizione del NPN Q1 forzando un livello BASSO
```

Software per Arduino (2ª parte)

Tensione_di_carica_scarica_condensatore_nel_tempo_Arduino.ino **Parte 3ª**

```
digitalWrite(transistor_npn, LOW); // interdizione del NPN Q1 forzando un livello BASSO
digitalWrite(transistor_pnp, HIGH); // interdizione del PNP Q2 forzando un livello ALTO
// Serial.print("\nPREMI IL PULSANTE PER INIZIARE LA FASE DI SCARICA");
while(digitalRead(pulsante) == HIGH) // nell'attessa della pressione del pulsante
 { // fai lampeggiare il led
  digitalWrite(led, HIGH); // accendi il led forzando un livello ALTO
  delay(100);  // ritardo di 100 msec.
  digitalWrite(led, LOW); // spegni il led forzando un livello BASSO
  delay(100);  // ritardo di 100 msec.
 };
// FASE DI SCARICA DEL CONDENSATORE
// Serial.print("\nSCARICA DEL CONDENSATORE");
digitalWrite(led, LOW); // spegni il led forzando un livello BASSO
digitalWrite(transistor_pnp, HIGH); // interdizione del PNP Q2 forzando un livello ALTO
delay(10); // ritardo di 10 msec
digitalWrite(transistor_npn, HIGH); // saturazione del NPN Q1 forzando un livello ALTO
// acquisisci il valore dell'input analogico con un tempo espresso nella variabile
// denominata "durata_acquisizione", espresso in millisecondi
tempo = 0; // reset del tempo
```

Software per Arduino (3ª parte)

Tensione_di_carica_scarica_condensatore_nel_tempo_Arduino.ino | **Parte 4ª**

```
  for(int x=0; x < 100; x++) // tempo totale = 100 * 10 = 1000 msec = 1 sec
  {
    lettura_tensione_c1 = analogRead(input_tensione_c1); // lettura della tensione ai capi di C1
    tempo = tempo + durata_acquisizione; // aggiorna il tempo incrementandolo
    delay (durata_acquisizione); // ritardo tra una acquisizione e la successiva
    // Serial.print(tempo); // trasmetti il dato tempo
    // Serial.print("\t"); // trasmetti il dato TAB
    Serial.println(lettura_tensione_c1); // trasmetti i dati acquisiti in formato decimale
  }
  // Serial.print("\nFINE ACQUISIZIONE");
  // FASE DI STAND-BY
  digitalWrite(transistor_npn, LOW); // interdizione del NPN Q1 forzando un livello BASSO
  digitalWrite(transistor_pnp, HIGH); // interdizione del PNP Q2 forzando un livello ALTO
}
/*****************************************************************************/
```

Software per Arduino (4ª parte)

Tensione_di_carica_scarica_condensatore_nel_tempo_Processing.pde **Parte 1ª**

```
/*  I.I.S. Primo LEVI - Torino          Data: 10/01/2012
    ACQUISISCE, MEMORIZZA E VISUALIZZA I DATI RELATIVI ALLA CARICA/SCARICA
    DI UN CONDENSATORE E RESISTENZA COMPRESI TRA 0 e 5V (10 bit = 0-1023 valori)
    Progetto: Acquisizione, memoria e visualizzazione dati      Autori: Classe 5BN
    Descrizione:  Legge dalla seriale i valori Di un R/C (0-1023) trasmessi
    dalla scheda Arduino, li visualizza su un grafico e li memorizza su file.    */
// importazione del file per gestire la seriale
import processing.serial.*;
Serial mia_porta_rs232;   // la definizione del nome relativo alla porta seriale
int xPos = 1;         // posizione orizzontale del grafico in pixel
  boolean fine_acquisizione = false; // fine acquisizione, salva e chiudi il file dei dati
// definisce l'uscita su file
PrintWriter output;
/*********************************************/
void keyPressed()  // se viene premuto un tasto "" (Close)
{
  if (key == 'c' || key == 'C') // se e' stato premuto il carattere "C" oppure "c"
   { // fine acquisizione, salva e chiudi il file dei dati
     fine_acquisizione = true;
   }
}
/*********************************************/
void setup ()
{
  // dimensioni della finestra grafica x, y
  size(1000, 400);
  // Crea un nuovo file denominato "dati.txt" nella cartella dove si trova lo scketch
  output = createWriter("dati.txt");
```

Software per Processing (1ª parte)

Tensione_di_carica_scarica_condensatore_nel_tempo_Processing.pde **Parte 2ª**

```
  // stampa la lista delle porte seriali che sono attive
  println(Serial.list());
  // viene attivata la prima della lista denominata "0" alla velocita' di 115200 baud
  mia_porta_rs232 = new Serial(this, Serial.list()[0], 115200);
  // non generare un serialEvent() se ricevi una nuova linea (C.R. Carriage Return):
  mia_porta_rs232.bufferUntil('\n');
  // setta i valori di background:
  background(0);
}
/*********************************************/
void draw ()
{
    if (fine_acquisizione) // fine acquisizione, salva e chiudi il file dei dati
    {
     output.flush(); // Scrivi i dati rimanenti rimasti nel buffer
     output.close(); // Chiudi il file
     exit(); // ESCI dal programma
    }
}
/*********************************************/
void serialEvent (Serial mia_porta_rs232)
{ // acquisisci la stringa di caratteri ASCII:
  String inString = mia_porta_rs232.readStringUntil('\n');
  if (inString != null) // se la stringa e' diversa da 0 caratteri
  {
    // cancella ogni eventuale spazio:
    inString = trim(inString);
    // converti la stringa in un numero reale
```

Software per Processing (2ª parte)

Tensione_di_carica_scarica_condensatore_nel_tempo_Processing.pde

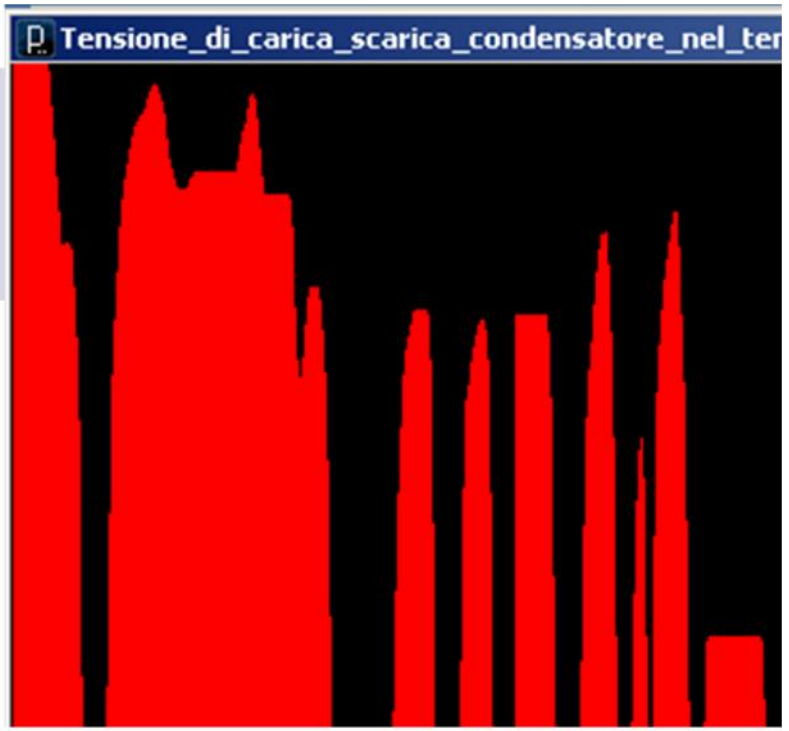
Parte 3ª

```
    float inByte = float(inString);
    // Scrivi i dati appena ricevuti nel file su una riga differente
    // con un carattere "\t" (TAB) tra ogni dato inserito
    // output.println(xPos + "\t" + inByte);
    // output.println(inByte); // per memorizzare in valori reali
    output.println(int (inByte)); // per memorizzare in valori interi
    // rimappa il numero in un valore compreso tra 0 e 1023 per essere visualizzato sul grafico
    inByte = map(inByte, 0, 1023, 0, height);
    // disegna la linea sul grafico:
    stroke(255,0,0); // colore nel formato Red, Green, Blue (range tra 0 e 255 per ogni colore)
    line(xPos, height, xPos, height - inByte);
    // controlla se sei alla fine del grafico, in tal caso cancella e inizia:
    if (xPos >= width)
     {
      xPos = 0;
      background(0);
     }
    else
     {
      // incrementa la posizione orizzontale di un pixel:
      xPos++; // xPos = xPos + 1; incrmento unitario
     }
  }
}
```

Software per Processing (3ª parte)

Rilievo della tensione di carica / scarica di un condensatore

Software Arduino e Processing con grafico (Excel) e file dati.



dati.txt

Tensione_di_carica_scarica_condensatore_nel_tempo_Processing.pde

dati.txt - Blocco note

File Modifica Formato Visualizza ?

1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1015
980
947
909
873
840
840
841
840
834
795
744