

Arduino ed applicazioni

Componenti, dispositivi e altro

a cura di

prof.ssa Tiziana Marsella

prof. Romano Lombardi

Indice generale

Arduino ed applicazioni	1
LA RESISTENZA.....	1
Arduino e i LED.....	3
Il LED.....	3
Colore della luce emessa.....	3
Costruzione del circuito di prova del LED.....	5
Come funziona il circuito di prova del LED.....	8
Accensione e spegnimento di un LED verde con Arduino.....	9
Accendere/Spegnere il LED con un programma.....	10
Come funziona LedOnOff.pde.....	11
Programma di Esempio: LedOnOff10.pde.....	12
Come funziona LedOnOff10.....	12
Un secondo circuito con i LED	13
Controllare ambedue i LED.....	13
Programma Esempio: Flash2Leds.pde.....	13
Il LED BiColore.....	15
Programma di Esempio: TestBiColorLED.....	17
IL DIODO RGB (anodo comune – catodo comune).....	19
Programmazione di Arduino con RGB.....	22
CODIFICA DEL TELECOMANDO.....	23
Principio di funzionamento.....	23
CONTROLLO MOTORI CON ARDUINO E DRIVER L293D.....	28
Controllo motori con alimentazione diretta	28
IL CHIP L293D.....	30
Codice di programmazione per Arduino.....	32
Programma di test.....	32
DS1620: termometro digitale	35
Descrizione del chip DS1620.....	35
Hardware.....	35
Funzionamento.....	36
Comunicazione sul bus a 3 vie.....	37
Lettura del contatore	38
Codice di programma con Arduino.....	38
INSEGUITORE DI TRACCIA CON 3 CNY70.....	42
Sensori CNY70.....	42
Funzionamento ed utilizzo.....	42
Inseguitore di traccia con Arduino.....	43
Codice di programmazione per Arduino.....	44
Programma di test per 3 sensori CNY70.....	44
Inseguitore di traccia con 3 CNY70 e Arduino.....	44
IL DISPLAY SERIALE LCD.....	48
Il display SerLCD della Spark fun.....	48
Caratteristiche.....	48
Interfaccia di comunicazione.....	48
Configurazione.....	48
Retroilluminazione.....	48
Comandi aggiuntivi.....	49
Splash screen.....	50

Modifica del valore della velocità (baud rate).....	50
Hardware.....	50
Controllo di contrasto.....	51
Pin di controllo dell'alta corrente.....	51
Programmi con Arduino e serLCD.....	51
Test del SerLCD con Arduino.....	51
LCD parallax.....	53
Velocità di invio dati.....	54
Test del display.....	54
Visualizzazione del testo.....	55
Spostamento del cursore.....	55
Controllo del display.....	55
Custom caracters.....	56
Set dei comandi.....	56
IL TSL230R.....	57
Caratteristiche tecniche.....	57
Descrizione.....	57
Funzione dei pin.....	58
Schemi a blocchi.....	58
Valori massimi nel range di temperatura con utilizzo in aria del TSL230.....	58
Condizioni di funzionamento raccomandati.....	58
Caratteristiche elettriche principali.....	59
Informazioni applicative.....	59
Codice di programmazione con Arduino.....	59
I SERVOMOTORI.....	61
I servomotori a rotazione non continua.....	61
Comandare un servomotore.....	61
I servomotori a rotazione continua.....	63
Programmazione Arduino per servomotori a rotazione continua	63
I MICROSWITCH.....	64
Utilizzo.....	64
Codice di programmazione con Arduino.....	65
Modulo di comando di due microswitches.....	66
Schemi elettrici e di realizzazione.....	66
Problematiche	67
Codice di antirimbalzo.....	67
SENSORE RILEVAMENTO GPL.....	69
FOTORESISTENZE.....	71
IL BUZZER.....	75

LA RESISTENZA

Una resistenza è un dispositivo che si oppone al passaggio della corrente.

Ogni resistenza ha un valore che indica con quanta forza resiste al flusso di corrente. Questo valore della resistenza viene indicato in ohm ed il simbolo per gli ohm è la lettera dell'alfabeto Greco - Ω . La resistenza con cui lavorerete in quest'esercizio, è la resistenza da 470 Ω mostrata nella Figura 1.

La resistenza ha due fili chiamati terminali, che escono da ciascun'estremità. Tra i due terminali c'è un contenitore ceramico, ed è questa la parte che oppone resistenza al flusso della corrente.

La maggior parte degli schemi che usano il simbolo della resistenza a sinistra, una linea a zig zag, indicano che la persona che dovrà costruire il circuito, deve usare una resistenza da 470 Ω . E' chiamato simbolo schematico.

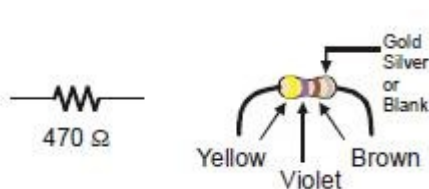


Figura 1
Disegno di una resistenza da 470 Ω

Simbolo Schematico (a sinistra) e disegno del componente (a destra).

Le resistenze come quelle che stiamo usando in quest'Esercizio, hanno fasce colorate che indicano il valore di resistenza. Ciascun valore di resistenza ha una combinazione diversa di colori. Per esempio il codice colori per la resistenza da 470 Ω è giallo, viola, marrone.

Ci può essere una quarta fascia che indica la tolleranza della resistenza. La tolleranza è misurata in percentuale, ed indica di quanto il valore reale si può discostare dal valore nominale. La quarta fascia può essere oro (5%), argento (10%), o essere assente (20%).

Per gli esercizi di questo testo, la tolleranza della resistenza non è importante, ma lo è il valore.

Ciascuna fascia colorata indica una cifra del valore della resistenza e questi valori/cifre sono elencati nella Tabella 2-1. Figura 2

Tabella 2-1: Valori del codice colori delle resistenze

Digit	Color
0	Nero
1	Marrone
2	Rosso
3	Arancio
4	Giallo
5	Verde
6	Blu
7	Viola
8	Grigio
9	Bianco

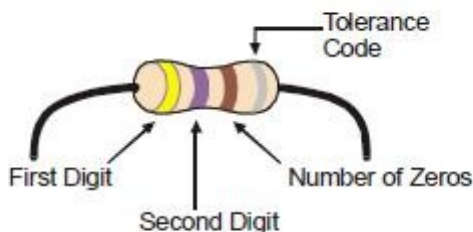


Figura 2
Codice colori delle resistenze

Di seguito c'è un esempio che mostra come usare la Tabella 2-1 e la Figura 2 per sapere il valore di una resistenza e che vi prova che giallo-viola-marrone è una resistenza da 470 Ω :

- La prima fascia è gialla, significa che la prima cifra è 4.
- La seconda fascia è viola, significa che la seconda cifra è 7.
- La terza fascia è marrone, dal momento che marrone è 1, significa aggiungere 1 zero alle prime due cifre.

Giallo-Viola-Marrone = 4-7-0.

Arduino e i LED

La maggioranza degli indicatori luminosi che si vedono nei dispositivi elettronici sono chiamati diodi emettitori di luce indicato nei testi con le lettere LED.

Un circuito con LED può essere collegato alla piattaforma Arduino che può essere programmato per collegare e scollegare l'alimentazione del circuito LED. Questo è molto più semplice che cambiare manualmente i collegamenti del circuito o collegare e scollegare la pila.

Arduino può inoltre essere programmato per fare le seguenti cose:

- Accendere e spegnere un circuito LED ad intervalli differenti
- Accendere e spegnere un circuito LED un determinato numero di volte.
- Controllare più di un LED
- Controllare il colore di un LED bicolore.

II LED

Un diodo è un dispositivo che fa passare la corrente solo in un verso (si chiama per questo componente unidirezionale), un diodo emettitore di luce (LED) emette luce quando la corrente lo attraversa. Il colore di un LED normalmente indica di quale colore s'illuminerà quando la corrente lo attraversa.

Come in molti altri tipi di diodo, il principio di funzionamento del LED si basa sulle proprietà delle giunzioni, le superfici di contatto tra due zone di un cristallo semiconduttore (zona *p* e zona *n*) con caratteristiche diverse (*vedi drogaggio*).

Quando è percorsa da una debole corrente elettrica (valori tipici da 10 a 20 mA), la giunzione emette spontaneamente fotoni di una determinata lunghezza d'onda, e quindi di un determinato colore.

La corrente elettrica che attraversa un LED deve avere un'intensità controllata, per evitare che arrechi danni al componente.

Molto usati come indicatori elettronici, i LED sono reperibili in varie forme e dimensioni.

Anche se è cosa poco nota, i LED sono "macchine reversibili": infatti, se la loro giunzione viene esposta direttamente ad una forte fonte luminosa o ai raggi solari, ai terminali appare una tensione, dipendente dall'intensità della radiazione e dal colore del led in esame (massima per il blu). Questa caratteristica viene abitualmente sfruttata nella realizzazione di sensori, per sistemi di puntamento ([inseguitori solari](#)) di piccoli impianti fotovoltaici o a concentratore e per molti altri scopi

Colore della luce emessa

A seconda del drogante utilizzato, i LED producono i seguenti colori:

- [AlGaAs](#) - rosso ed infrarosso
- [GaAlP](#) - verde
- [GaAsP](#) - rosso, rosso-arancione, arancione, e giallo
- [GaN](#) - verde e blu

- [GaP](#) - rosso, giallo e verde
- [ZnSe](#) - blu
- [InGaN](#) - blu-verde, blu
- [InGaAlP](#) - rosso-arancione, arancione, giallo e verde
- [SiC](#) come substrato - blu
- [Diamante \(C\)](#) - ultravioletto
- [Silicio \(Si\)](#) come substrato - blu (in sviluppo)
- [Zaffiro \(Al₂O₃\)](#) come substrato – blu

Tipologia LED	tensione di giunzione V_f (volt)
Colore infrarosso	1,3
Colore rosso	1,8
Colore giallo	1,9
Colore verde	2,0
Colore arancio	2,0
Flash blu/bianco	3,0
Colore Blu	3,5 V
Colore Ultravioletto	4 ÷ 4,5 V

Tensione di polarizzazione diretta

Il valore della resistenza in serie R_s è calcolato mediante la legge di Ohm e la legge di Kirchhoff conoscendo la corrente di lavoro richiesta I_f , la tensione di alimentazione V_s e la tensione di giunzione del LED alla corrente di lavoro data, V_f .

Nel dettaglio, la formula per calcolare la resistenza in serie necessaria è: $R_s = (V_f - V_s) / I_f$.

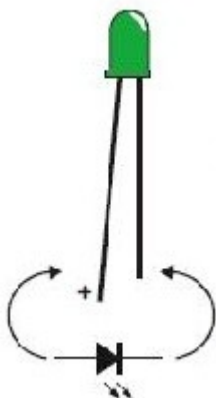
Per quanto riguarda gli assorbimenti di corrente elettrica di alimentazione entrante nel dispositivo, questi variano molto in funzione della tipologia di LED: sono minori nei LED normali usati come indicatori rispetto a quelli ad alta luminosità (led flash e di potenza), secondo la seguente tabella:

Tipologia LED	Assorbimento (mA)
LED basso consumo	3 - 10
LED normali	10 - 15
LED flash	20 - 40
LED di potenza	100 - 20000

La polarità di un LED è contenuta nella sua forma.

Dal momento che un LED è un dispositivo polarizzato, vi dovete assicurare di collegarlo nella giusta maniera, o non funzionerà come voluto.

La Figura a lato mostra lo schema di collegamento di un LED e lo schema pratico.



Un LED ha due terminali. Uno è chiamato **anodo**, l'altro è chiamato **catodo**.

Sullo schema pratico, l'anodo è marcato con il simbolo più (+). Sullo schema elettrico, l'anodo è la parte larga del triangolo.

Sullo schema pratico il catodo è il terminale non contrassegnato, e sullo schema elettrico, il catodo è la linea al vertice del triangolo.

Quando si inizia l'assemblaggio del circuito, occorre controllare e confrontare lo schema elettrico con lo schema pratico. Sullo schema pratico, notare che i terminali del LED sono di lunghezza diversa. Il terminale più lungo è l'anodo del LED, il terminale corto è il catodo. Inoltre se si guarda attentamente il corpo plastico del LED è quasi rotondo con una piccola zona piatta vicino al terminale corto identificando così il catodo. Questo fatto è particolarmente utile se i terminali sono stati tagliati alla stessa lunghezza.

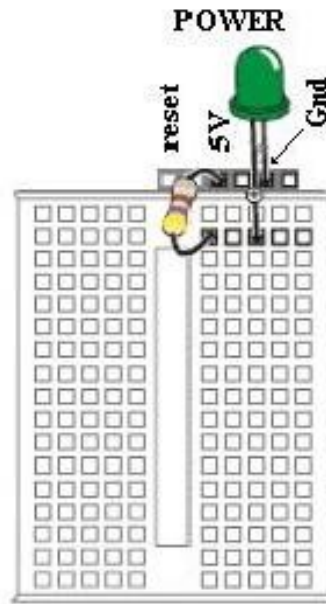
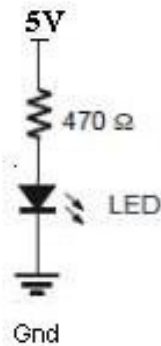
Costruzione del circuito di prova del LED

Dallo schema di Arduino Uno (fig4) , in basso , si individuano i connettori neri relativi alla massa GND e al piedino di uscita di 5V.



Costruirete il circuito inserendo i terminali del LED e della resistenza nei fori della piastra per prototipi (breadboard) come mostrato nella figura

Figura 5



La breadboard è collegata con dei connettori neri sul lato inferiore della piattaforma di Arduino.

I connettori neri sul lato inferiore, sono etichettati: Reset, 3V3, 5V, Gnd, Vin.

Le etichette 5V, Gnd, Vin si chiamano terminali di alimentazione, e saranno usati per fornire elettricità ai vostri circuiti. I connettori neri sul lato superiore di Arduino sono etichettati con le sigle 0,1,2,..13 Queste sono le connessioni che userete per collegare il vostro circuito ai piedini di ingresso/uscita dell'Arduino.

La zona bianca con molti fori, si chiama area senza saldature e viene utilizzata per collegare tra loro i componenti dei circuiti.

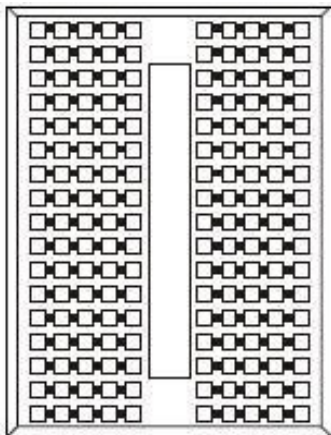


Figura 6

Piastra per prototipi

*Connettori di alimentazione (fori neri sul lato superiore),
Piedini di ingresso/uscita (connettori neri sul lato sinistro),
area prototipi senza saldature (connettori bianchi)*

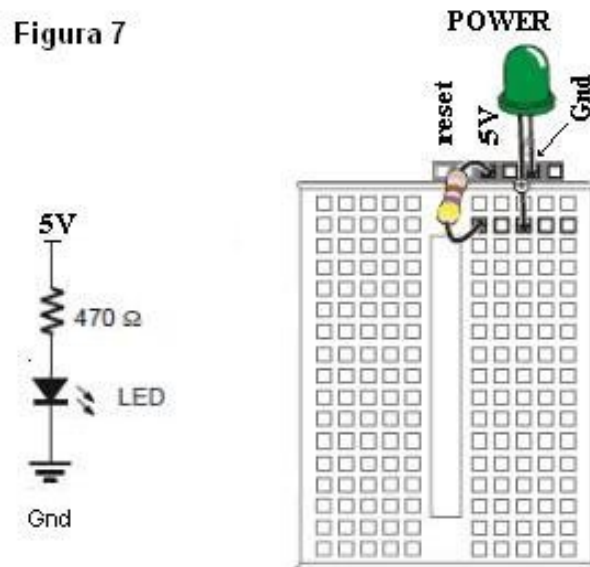
Siete ora pronti per costruire il circuito mostrato in Figura 7 (sotto) inserendo la resistenza ed il LED nei connettori dell'area prototipi. Seguite i seguenti passi:

- Togliere l'alimentazione dalla vostra piattaforma.
- Usare la Figura 5 per decidere quale terminale è collegato al catodo del LED.
- Cercare il terminale più corto e la zona piatta sul corpo plastico del LED.

- Inserire il catodo del LED in uno dei connettori neri etichettati Gnd sul lato inferiore di Arduino.
- Inserire l'anodo del LED (l'altro terminale, il più lungo) nel connettore della porzione di area prototipi mostrata.
- Inserire uno dei terminali della resistenza nella stessa fila di connettori dell'anodo del LED. Questo collegherà insieme i due componenti.
- Inserire l'altro terminale della resistenza nel connettore etichettato 5V.

La polarità è importante per il LED: collegando un LED al contrario, il LED non emetterà luce quando si alimenterà il circuito.

- Rialimentate Arduino
- Controllate che il LED verde si accenda. Dovrebbe illuminarsi di verde.



Se il LED verde non si accende quando alimentate la scheda:

- Alcuni LED sono più luminosi se li guardate da sopra. Provate a guardare direttamente sulla cupoletta del contenitore plastico.
- Se la stanza è fortemente illuminata provate a spegnere l'illuminazione oppure usate le mani per mettere in ombra il LED.

Se ancora non vedete nessuna luce verde, fare questi tentativi:

- Ricontrollate per assicurarvi che l'anodo ed il catodo siano collegati correttamente. Se questo non è, semplicemente rimuovete il LED e reinsertelo al contrario. Non danneggerete il LED se lo collegherete al contrario, semplicemente non si accenderà. Quando sarà collegato per il giusto verso, si accenderà.
- Ricontrollate per assicurarvi che avete assemblato il circuito esattamente come mostrato in Figura 7.
- Se siete in laboratorio, controllate insieme all'insegnante.

Come funziona il circuito di prova del LED.

I terminali 5V e Gnd forniscono tensione elettrica allo stesso modo di come farebbe una pila. Il connettore 5V equivale al polo positivo della pila, ed i connettori Gnd sono uguali al polo negativo. La Figura 8 mostra come applicando una differenza di potenziale ad un circuito con una pila, costringe gli elettroni a fluire attraverso di essa. La corrente elettrica è limitata dalla resistenza. Questa corrente è la causa dell'illuminazione del diodo.

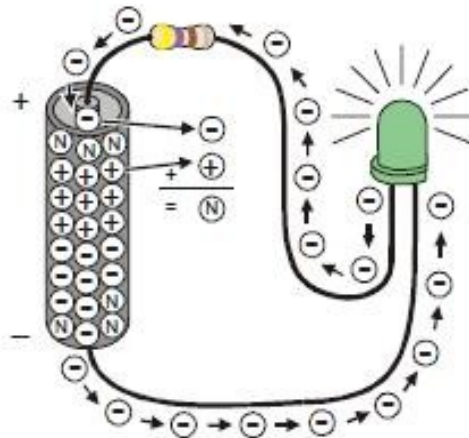


Figura 8
Flusso di elettroni nel circuito LED acceso.

I segni meno nei cerchietti vengono usati per mostrare gli elettroni che scorrono dal polo negativo della pila al suo polo positivo.

Le Reazioni chimiche all'interno della pila forniscono corrente al circuito. Il polo negativo della pila contiene un composto di molecole con più elettroni del normale (rappresentate nella Figura 8 dai segni meno). Il polo positivo della pila è fatto di un composto chimico con molecole a cui mancano elettroni (rappresentate dai segni più). Quando un elettrone lascia una molecola per scorrere attraverso il filo, viene chiamato elettrone libero (anche lui rappresentato da un segno meno). La molecola che ha perso l'elettrone, non ha più una carica negativa; è ora chiamata neutra (indicata da una N). Quando un elettrone giunge al polo positivo, si unisce ad una molecola a cui mancava un elettrone, cosicché quella molecola diventa neutra.

La Figura 9 mostra come il flusso di corrente attraverso il circuito LED venga descritto usando la simbologia schematica.

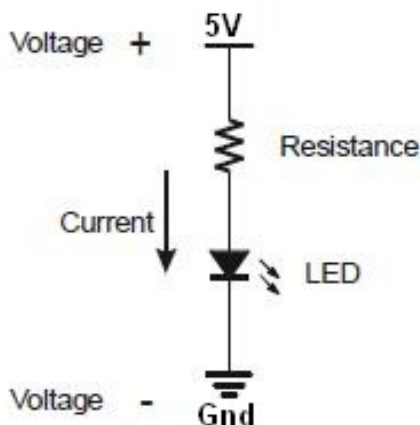


Figura 9
Schema di un circuito LED acceso che mostra la tensione ed il verso convenzionale del flusso della corrente

I segni + e - mostrano la tensione applicata al circuito, e la freccia mostra il verso di scorrimento della corrente attraverso il circuito.

I segni + e – sono usati per mostrare dove la tensione viene applicata al circuito.

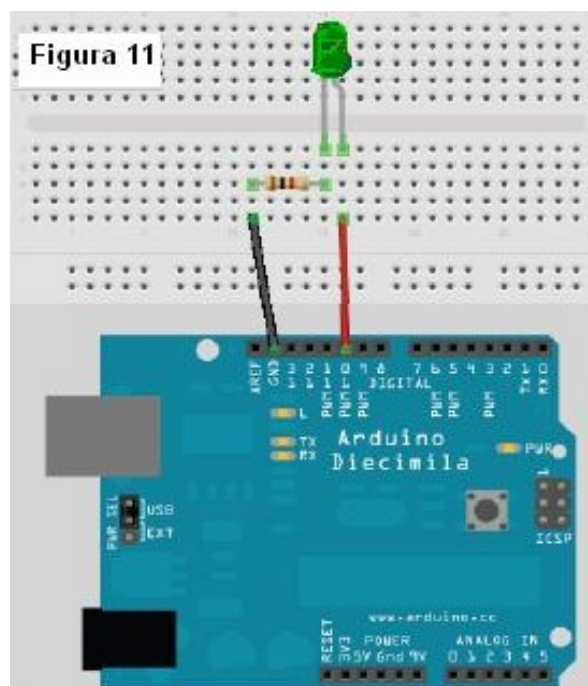
La freccia mostra il verso della corrente nel circuito. Questa freccia punta di solito nel verso opposto al flusso reale degli elettroni.

Si ritiene che Benjamin Franklin non si fosse reso conto degli elettroni quando decise di rappresentare il flusso di corrente come cariche passanti dal polo positivo al polo negativo di un circuito.

Al momento in cui i fisici scoprirono la vera natura della corrente elettrica, tale convenzione era già notevolmente diffusa, per cui è rimasta.

Accensione e spegnimento di un LED verde con Arduino

In quest'esempio il LED viene collegato ad Arduino sul pin 10 e verrà programmato per accenderlo e spegnerlo anche a diverse velocità.



Ci sono due grandi differenze tra cambiare manualmente la connessione ed avere Arduino che lo fa.

- Primo, Arduino non deve togliere l'alimentazione quando cambia il collegamento da 5V a Gnd.
- Secondo, mentre un uomo può fare questo scambio diverse volte al minuto, Arduino lo può fare migliaia di volte al secondo!

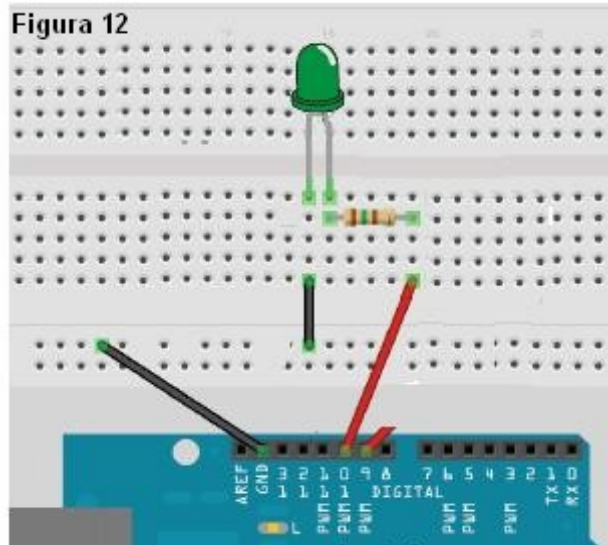
Componenti del circuito di prova del LED

1 LED verde

1 breadboard

1 resistore da 150 Ω

fili jumpers per breadboards



Accendere/Spegnere il LED con un programma.

Il programma esempio fa lampeggiare il LED una volta al secondo.

- Digitate il codice LedOnOff nell'Editor dell'Arduino
- Rialimentate Arduino.
- Lanciate il programma.
- Verificate che il LED lampeggi ad un ritmo di un secondo.
- Quando avete finito, scollegate l'alimentazione.

*// LedOnOff.pde (l'estensione dei file Arduino è pde) accende e spegne un LED.
//Lo ripete 1 volta al secondo indefinitamente.*

```
#define ledPin 10 // direttiva che fissa il led al pin 10
void setup() {
    pinMode(ledPin, OUTPUT); // imposta il pin digitale ledPin come
} // uscita

void loop() {
    digitalWrite(ledPin, HIGH); // accende ledPin
    delay(500); //attende mezzo secondo (500 ms)
    digitalWrite(ledPin, LOW); //spegne ledPin
    delay(500); //attende mezzo secondo
}
```

Come funziona LedOnOff.pde

La direttiva `#define ledPin 10` specifica che il LED che si vuole far lampeggiare è collegato al Pin 10 di Arduino.

Le parentesi `{ }` racchiudono il blocco di codice relative al `setup()` che in questo caso specifica che il pin definito da `ledPin` è impostato come output.

pinMode è una funzione che imposta i pin quali INPUT o OUTPUT.

INPUT e OUTPUT nel linguaggio di Arduino sono parole costanti (hanno valore assegnato nella sintassi di programmazione di Arduino).

loop() è il luogo dove si specifica il comportamento principale del dispositivo interattivo, che verrà ripetuto all'infinito finché non si spegne la scheda.

Il comando `digitalWrite(ledPin, HIGH);` forza Arduino a collegare internamente il Pin I/O definito in precedenza da `ledPin` a livello alto di conseguenza il LED si accende.

Il comando `delay(500)` forza Arduino a non far niente per ½ secondo mentre il LED resta acceso.

Il numero 500, indica al comando `delay` di attendere 500ms.

Il comando `digitalWrite(ledPin, LOW);` forza Arduino a collegare internamente il Pin I/O definito da `ledPin` a Gnd. Questo spegne il LED. Dal momento che questo comando è seguito da un altro comando `delay(500);` il LED resta spento per ½ secondo.

La ragione per cui il codice si ripete all'infinito, è perché è inserito all'interno del blocco `void loop()` un ciclo loop che fa ritornare all'inizio del blocco in questione.

Arduino non è in grado di eseguire contemporaneamente diversi programmi e dai programmi non si può uscire. Quando si accende la scheda, viene eseguito il codice; quando lo si vuol fermare basta spegnerlo.

Un esperimento divertente consiste nel vedere quanto potete accorciare le pause pur continuando a vedere lampeggiare il LED. Quando il LED lampeggia velocemente, ma sembra essere sempre acceso, questo fenomeno si chiama **persistenza visiva**. Di seguito viene indicata la procedura per vedere qual è la vostra soglia di persistenza visiva:

- Provate a modificare ambedue le **Durate** dei comandi `delay` così che siano 100.
- Rilanciate il programma e controllate per vedere un rapido lampeggio.
- Riducete ambedue gli argomenti **Durata** di 5 e provate ancora.
- Continuate a ridurre gli argomenti Durata fino a che il LED appaia sempre acceso senza lampeggi. Dovrebbe accendersi in modo attenuato ma senza lampeggiare.

Commentare una linea di codice:

Mettere un `//` a sinistra di un comando, lo trasforma in un commento. Questo è uno strumento utile perché non dovrete fisicamente cancellare il codice per vedere cosa accade se lo togliete dal programma. È molto più facile aggiungere e rimuovere un `//` che eliminare e ridigitare i comandi. Inoltre è possibile sempre spiegare, passo dopo passo, cosa il programma dovrebbe svolgere.

Come scegliere il nome delle variabili:

1. Il nome non può essere una parola riservata..Alcuni esempi di parole riservate: ,**PAUSE**, **HIGH**, **LOW**, **DO**, **CONTINUE**.
2. Il nome non può contenere uno spazio.

Programma di Esempio: LedOnOff10.pde

Il programma LedOnOff10.pde dimostra come far lampeggiare un LED dieci volte.

- Il circuito di test è quello di fig.12.
- Digitate il codice LedOnOff10 nell'Editor di Aduino.
- Accendete il vostro Arduino .
- Scaricate il programma attraverso il comando di upload.
- Verificate che il LED lampeggi dieci volte.
- Attivate il programma una seconda volta, e verificate che il valore di **conta** mostrato sul monitor segua accuratamente quante volte il LED lampeggi.
- Suggerimento: invece di clickare upload una seconda volta, potete premere il tasto reset sulla vostra scheda.

//LedOnOff10.pde accende e spegne un LED. Lo ripete 1 volta al secondo per 10 volte.

```
#define ledPin 10
```

```
int conta = 0;      /* definizione della variabile del contatore di tipo intera e  
                    relativa inizializzazione */
```

```
void setup() {
```

```
    pinMode(ledPin, OUTPUT);    // imposta il pin digitale ledPin come  
                                // uscita
```

```
}
```

```
void loop() {
```

```
    if(conta<10){  
        digitalWrite(ledPin, HIGH);    // accende ledPin  
        delay(500);                    //attende mezzo secondo (500 ms)  
        digitalWrite(ledPin, LOW);    //spigne ledPin  
        delay(500);                    //attende mezzo secondo  
        conta++;                        // incrementa la variabile conta
```

```
}
```

Come funziona LedOnOff10

Questa dichiarazione

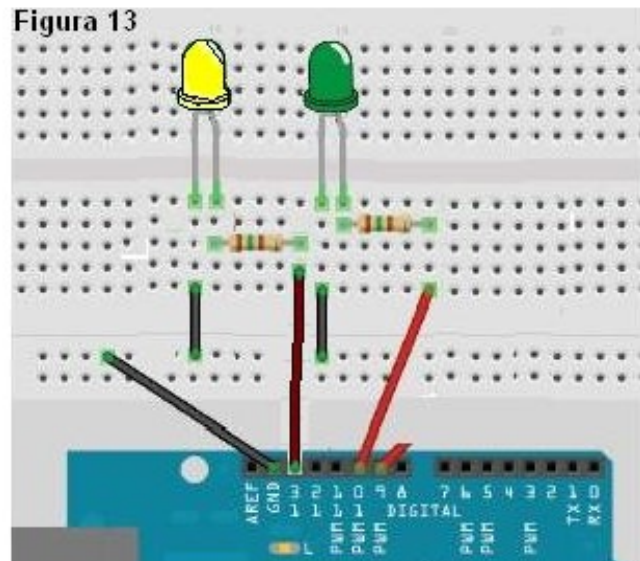
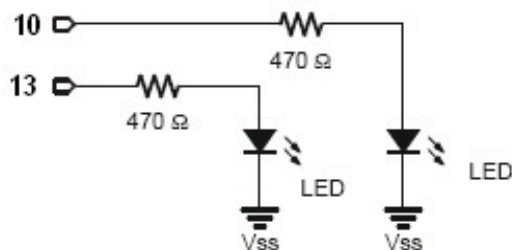
```
int conta=0;
```

specifica che il programma userà la parola **conta** come variabile che può contenere un'informazione grande un intero memorizzabile in una parola di 16 bit, cioè che va da -32768 a 32767

Se $conta=32767$, allora $conta++=32767+1$ non dà 32768 ma fa ripartire il valore a -32768 in quanto la variabile è in overflow.

Attenzione a quando si utilizzano gli incrementi di variabili.

Un secondo circuito con i LED



Controllare ambedue i LED

Far lampeggiare ambedue i LED insieme. Una maniera in cui potete farlo è di usare due comandi

`digitalWrite(ledPin, HIGH);`

prima del comando `delay`. Un comando con argomento **HIGH** imposta P10 alto, ed il successivo comando con argomento **LOW** imposta P13 alto. Dovrete anche inserire due argomenti **LOW** per spegnere entrambi i LED.

I due LED non lampeggeranno esattamente nello stesso momento perché saranno accesi o spenti uno dopo l'altro.

D'altronde, c'è una differenza di non più di un millisecondo tra i due cambiamenti, e l'occhio umano non la noterà.

Programma Esempio: Flash2Leds.pde

- Digitate il codice FlashBothLeds nell'Editor di Arduino.
- Eseguite upload.
- Verificate che ambedue i LED lampeggino nello stesso momento.


```

// flashe2leds accende e spegne un LED verde insieme ad un led giallo. Lo ripete
indefinitamente ogni secondo.
#define ledVerde 10
#define ledGiallo 13

void setup() {
  pinMode(ledVerde, OUTPUT); //imposta il pin digitale ledVerde come uscita
  pinMode(ledGiallo, OUTPUT); //imposta il pin digitale ledGiallo come uscita
}

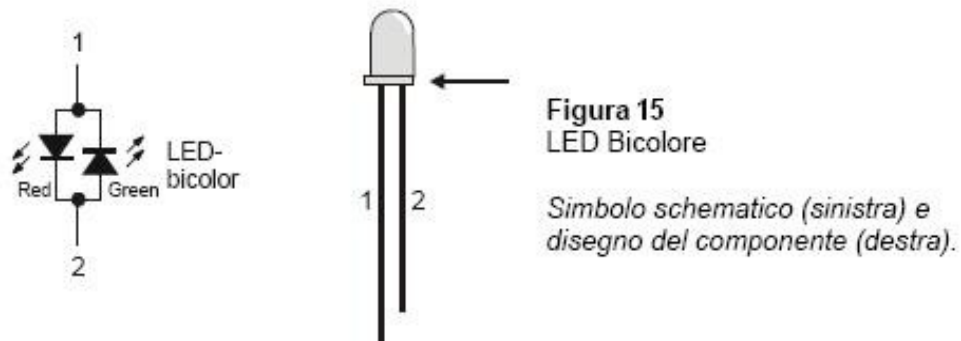
void loop() {
  digitalWrite(ledVerde, HIGH); // accende led verde
  digitalWrite(ledGiallo, HIGH); // accende led giallo
  delay(500); // attende mezzo secondo
  digitalWrite(ledVerde, LOW); // spegne led verde
  digitalWrite(ledGiallo, LOW); // spegne led verde
  delay(500); // attende mezzo secondo
} // end loop

```

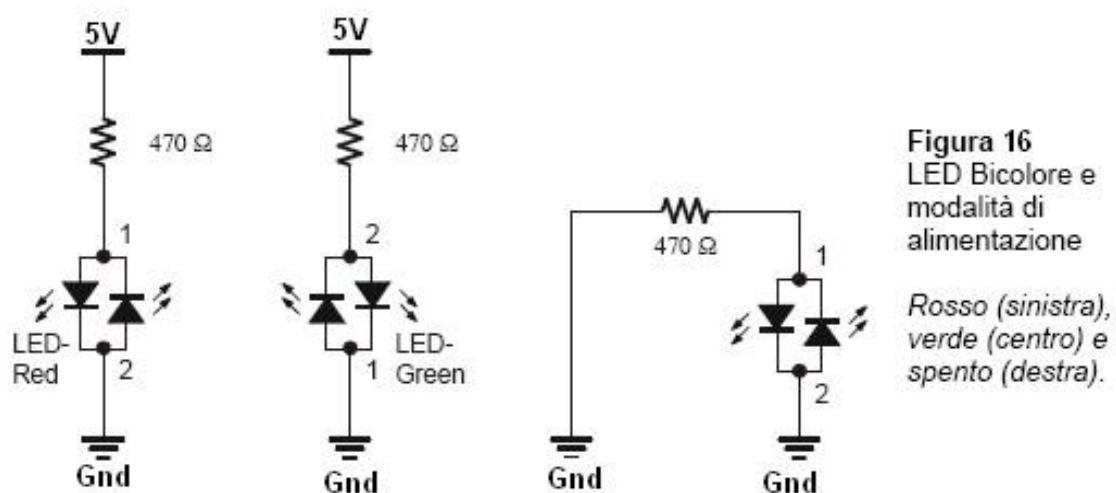
Il LED BiColore

Il simbolo schematico del LED bicolore ed il disegno sono mostrati nella Figura 15

Il LED bicolore in realtà è composto di due LED nello stesso involucro. La Figura 2-18 mostra che potete alimentarlo in un verso e si accenderà il LED rosso. Collegandolo al



contrario, si accenderà il LED verde. Come per gli altri LED, se collegate ambedue i terminali a Gnd, il LED non si accenderà.



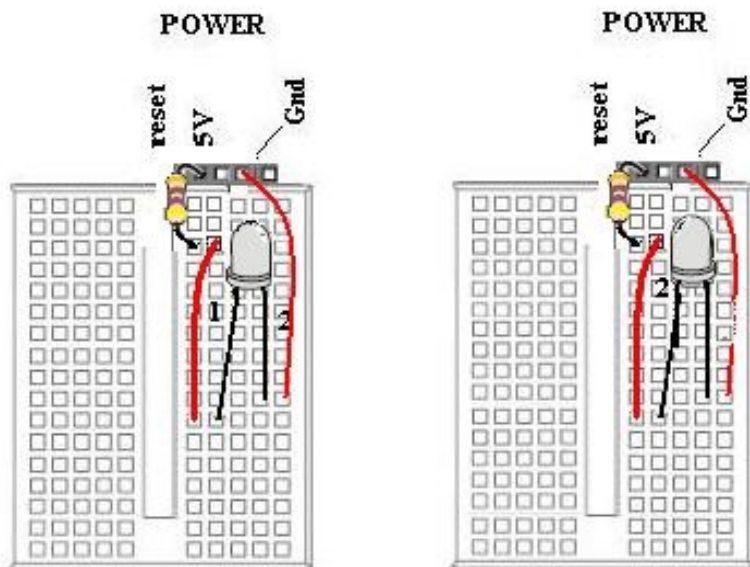


Figura 17
Collaudo Manuale del LED
bicolore
LED Bicolore rosso (sinistra)
e verde (destra).

- Scollegare l'alimentazione dalla vostra scheda Arduino
- Costruire il circuito mostrato a sinistra della Figura 17.
- Riaccendete e verificate che il LED bicolore si accenda di rosso.
- Spegnete di nuovo.
- Modificate il circuito che corrisponde al disegno a destra della Figura 17
- Riaccendete.
- Verificate che il LED bicolore si accenda di verde.
- Spegnete.

Controllare un LED bicolore con Arduino richiede due pin I/O. Dopo che avrete manualmente verificato che il LED bicolore funzioni, potrete collegarlo ad Arduino come mostrato in Figura 18.

- Collegate il LED bicolore ad Arduino come mostrato nella Figura 18.

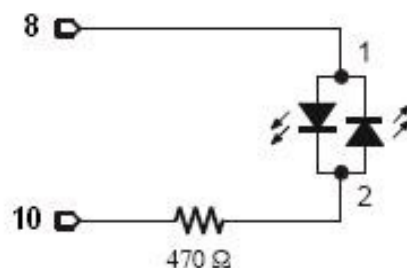


Figura 18
LED Bicolore

La corrente circola nel LED rosso quando 8 è collegato a 5V e 10 è collegato a massa (Gnd). Questo perché il LED rosso farà circolare corrente quando si applica la tensione come mostrato, mentre il LED verde polarizzato inversamente non permetterà alla corrente di circolare. Il LED bicolore si accenderà di rosso.

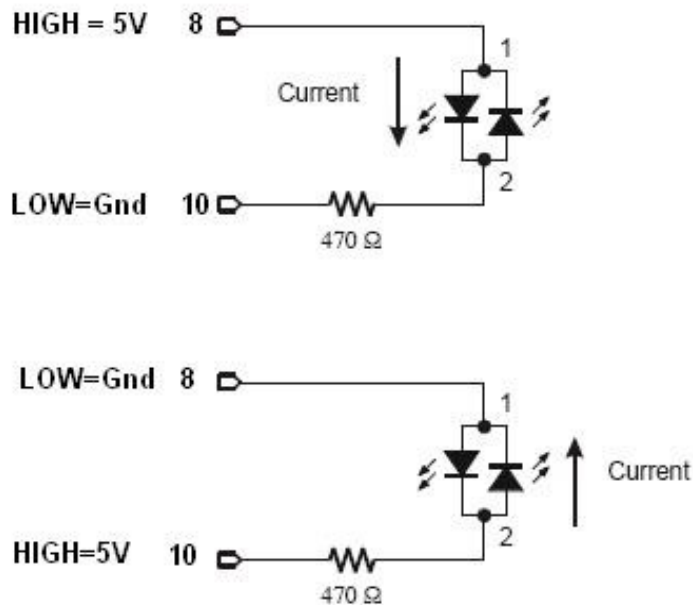


Figura 19
Collaudo manuale
del LED bicolore

La Corrente
attraverso il LED
rosso (sopra) ed
attraverso il LED
verde (sotto).

Lo schema mostra che cosa succede quando 8 è collegato a massa e 10 è collegato a 5V. La tensione è ora invertita. Il LED rosso si spegne e non permette alla corrente di circolare. Nel frattempo il LED verde si accende, la corrente può passare nel circuito nella direzione opposta.

Il LED bicolore si spegnerà anche se invierete un segnale alto ad entrambi i pin. Perché?

Perché la tensione sarà la stessa su questi pin e non importa se imposterete ambedue i pin high o low, in ogni caso non ci sarà differenza di potenziale.

Programma di Esempio: TestBiColorLED

*// TestBiColorLed comanda il lampeggio di un led bicolore a due pin
// il led è collegato tra i due piedini 8 e 10 con in serie un resistore da 470Ω.*

```
void setup() {
    pinMode(8, OUTPUT);           //imposta il pin digitale 8 come uscita
    pinMode(10, OUTPUT);
}

void loop() {
    digitalWrite(8, HIGH);       // accende rosso
    digitalWrite(10, LOW);
    delay(500);                  //attende mezzo secondo (500 ms)
    digitalWrite(8, LOW);        //accende verde
    digitalWrite(10, HIGH);
    delay(500);                  //attende mezzo secondo
```

```
digitalWrite(8, LOW);      //spento
digitalWrite(10, LOW);
delay(500);
} // end loop
```

IL DIODO RGB (anodo comune – catodo comune)



I Led RGB sono led capaci di produrre 3 differenti lunghezze d'onda, quella del Rosso, del Green (verde), e quella del Blu.

Essi non hanno solamente un anodo e un catodo, ma un anodo e 3 catodi (RGB ad anodo comune), o 3 anodi e un catodo (RGB a catodo comune):

RGB Hex Triplet Color Chart

E-mail-ware...What a concept!

If you find this chart helpful, send mail to Doug and say "Thanks!".
jacobson@phoenix.net

	FFFFFF	FFCCFF	FF99FF	FF66FF	FF33FF	FF00FF	
	FFFFCC	FFCCCC	FF99CC	FF66CC	FF33CC	FF00CC	
	FFFF99	FFCC99	FF9999	FF6699	FF3399	FF0099	
EEEEEE	FFFF66	FFCC66	FF9966	FF6666	FF3366	FF0066	00FF00
DDDDDD	FFFF33	FFCC33	FF9933	FF6633	FF3333	FF0033	00EE00
CCCCCC	FFFF00	FFCC00	FF9900	FF6600	FF3300	FF0000	00DD00
BBBBBB	CCFFFF	CCCCFF	CC99FF	CC66FF	CC33FF	CC00FF	00CC00
AAAAAA	CCFFCC	CCCCCC	CC99CC	CC66CC	CC33CC	CC00CC	00BB00
999999	CCFF99	CCCC99	CC9999	CC6699	CC3399	CC0099	00AA00
888888	CCFF66	CCCC66	CC9966	CC6666	CC3366	CC0066	009900
777777	CCFF33	CCCC33	CC9933	CC6633	CC3333	CC0033	008800
666666	CCFF00	CCCC00	CC9900	CC6600	CC3300	CC0000	007700
555555	99FFFF	99CCFF	9999FF	9966FF	9933FF	9900FF	006600
444444	99FFCC	99CCCC	9999CC	9966CC	9933CC	9900CC	005500
333333	99FF99	99CC99	999999	996699	993399	990099	004400
222222	99FF66	99CC66	999966	996666	993366	990066	003300
111111	99FF33	99CC33	999933	996633	993333	990033	002200
000000	99FF00	99CC00	999900	996600	993300	990000	001100
FF0000	66FFFF	66CCFF	6699FF	6666FF	6633FF	6600FF	0000FF
EE0000	66FFCC	66CCCC	6699CC	6666CC	6633CC	6600CC	0000EE
DD0000	66FF99	66CC99	669999	666699	663399	660099	0000DD
CC0000	66FF66	66CC66	669966	666666	663366	660066	0000CC
BB0000	66FF33	66CC33	669933	666633	663333	660033	0000BB
AA0000	66FF00	66CC00	669900	666600	663300	660000	0000AA
990000	33FFFF	33CCFF	3399FF	3366FF	3333FF	3300FF	000099
880000	33FFCC	33CCCC	3399CC	3366CC	3333CC	3300CC	000088
770000	33FF99	33CC99	339999	336699	333399	330099	000077
660000	33FF66	33CC66	339966	336666	333366	330066	000066
550000	33FF33	33CC33	339933	336633	333333	330033	000055
440000	33FF00	33CC00	339900	336600	333300	330000	000044
330000	00FFFF	00CCFF	0099FF	0066FF	0033FF	0000FF	000033
220000	00FFCC	00CCCC	0099CC	0066CC	0033CC	0000CC	000022
110000	00FF99	00CC99	009999	006699	003399	000099	000011
	00FF66	00CC66	009966	006666	003366	000066	
	00FF33	00CC33	009933	006633	003333	000033	
	00FF00	00CC00	009900	006600	003300	000000	

Copyright © 1995 Douglas R. Jacobson
All Rights Reserved



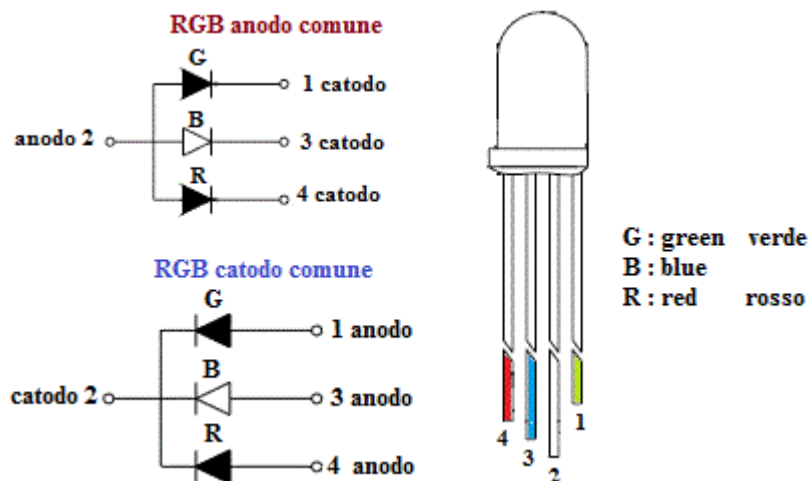
RGB è l'acronimo di Red, Green, Blue (ovvero Rosso, Verde, Blu). Tutti i colori visibili all'occhio umano sono in realtà generati dalla miscela di queste tre tinte fondamentali nelle giuste proporzioni.

Essi sono molto utilizzati quando si vogliono avere colori secondari o terziari, poiché si possono "miscelare" i colori primari applicando tensioni più o meno alte a più elettrodi contemporaneamente.

Hanno lo stesso tipo di involucro, impacchettamento, per cui si possono distinguere solo in fase di polarizzazione.

Praticamente un LED RGB è l'insieme di tre LED: uno rosso, uno verde e uno blu racchiusi in un unico package e invece di avere 6 morsetti, 3 anodi e 3 catodi possono avere 4 connettori: 3 catodi e un anodo comune oppure 3 anodi e un catodo comune.

Gli RGB più comuni sono quelli ad anodo comune e si trovano con un package da 5mm.



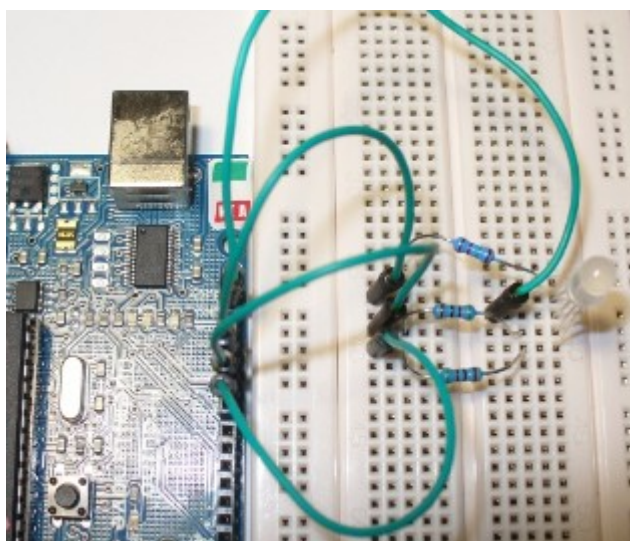
Tensione di polarizzazione diretta:

- Rosso: 2V
- Verde: 3,2V
- Blu: 3,2V

Luminosità:

- Rosso: 2800 mcd
- Verde: 6500 mcd
- Blu: 1200 mcd

Programmazione di Arduino con RGB



da completare

CODIFICA DEL TELECOMANDO

Il telecomando in dotazione nel laboratorio è quello di figura



Figura 1: Telecomando della DEA

Per poterlo utilizzare è necessario codificare il treno di impulsi che invia ad un IR del tipo tsop_18XX.

Principio di funzionamento

Alla pressione di un tasto, il telecomando invia al ricevitore IR un segnale caratterizzato dall'aver, nell'unità base di riferimento, sia un livello alto che un livello basso.

Se con oscilloscopio si visualizza cosa accade all'atto della pressione di un tasto del telecomando in uso potrebbe verificarsi una situazione del genere:

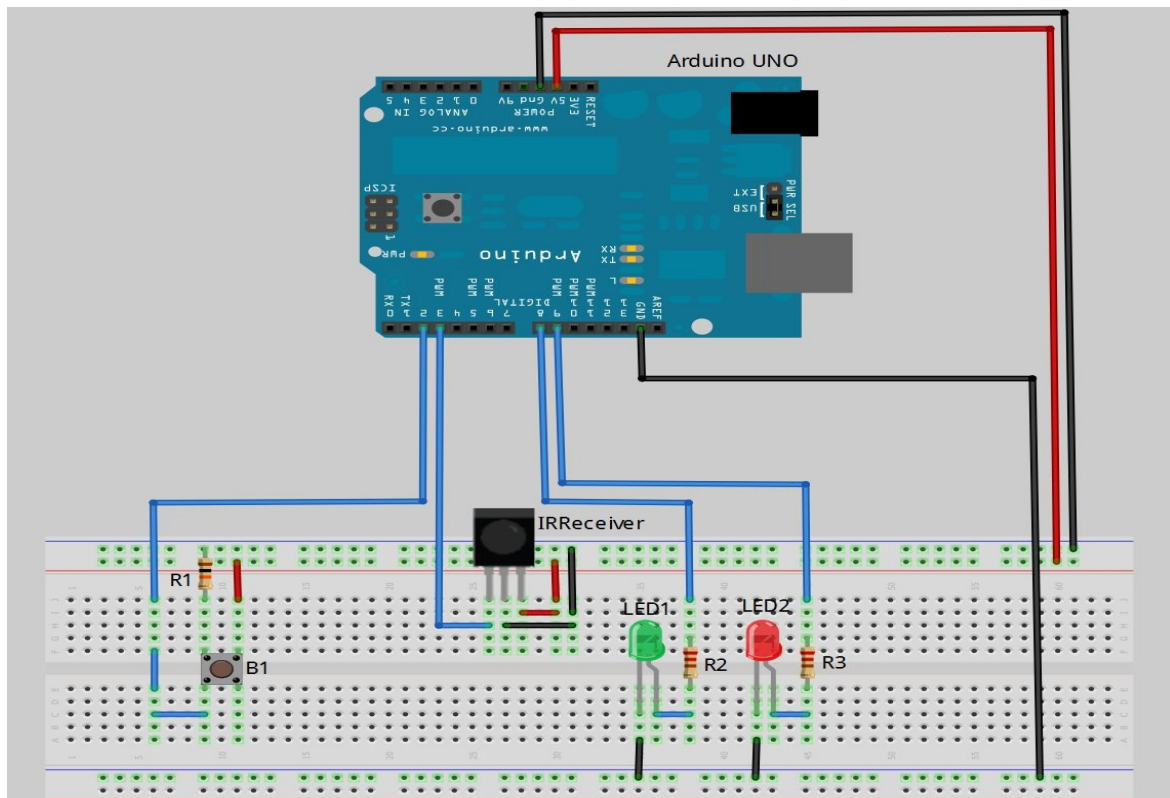
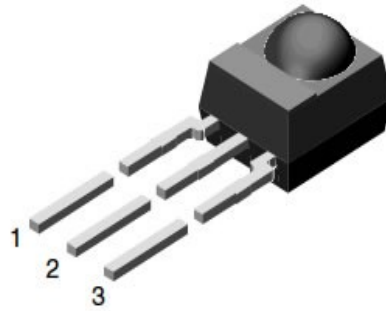


Come si osserva c'è una certa alternanza di valori alti e bassi per un totale di 12 alti e 12 bassi. Situazione analoga si presenta ad ogni pressione di un tasto se ne deduce che si può risalire al tasto premuto andando a memorizzare i 12 valori interi

Tramite il comando pulseIn si va a misurare l'ampiezza di ogni parte alta o bassa del treno di impulsi.

Il codice seguente permette di codificare la pressione di un tasto del telecomando della DEA in dotazione nei nostri laboratori e poter quindi poter associare alla pressione di un tasto un relativo codice numerico.

Il circuito vede un ricevitore IR a 38kHz con il pin 1 collegato al pin 2 digitale di Arduino attraverso un resistore di 220 Ω , il pin 2 collegato a massa mentre il pin 3 collegato all'alimentazione a 5V.



```
int pinIR = 2; // pin1 del sensore collegato al pin 2 di Arduino
// tramite un resistore di 220  $\Omega$ 
int BitStart = 2200; // Soglia per il bit di start in microsecondi
int bin_1 = 1000; // Soglia per il bit 1 in microsecondi.

void setup( ) {
```

```

pinMode(pinIR, INPUT);           // il pinIR in modalità input per ricevere
Serial.begin(9600);              // impostazione comunicazione seriale
}

```

```

void loop() {
    int Codice = trovaCodice();    // chiamata di funzione che restituisce il
                                   // codice in formato di tipo intero

    Serial.print("Codice ricevuto: ");
    Serial.println(Codice);
}

```

*// funzione che restituisce come codifica alla pressione di un tasto del
// telecomando un numero intero*

```

int trovaCodice( ) {
    int data[12];                 // i 12 impulsi che arrivano all'IR a seguito
                                   // della pressione di un tasto vengono
                                   // inseriti nell'array data[ ]

    while(pulseIn(pinIR, LOW) < BitStart) {
        // aspetta che passi la durata del bit di start
    }

    data[0] = pulseIn(pinIR, LOW); // inizia la misura della durata
    data[1] = pulseIn(pinIR, LOW); // degli impulsi bassi a partire
    data[2] = pulseIn(pinIR, LOW); // dal trigger alto-basso
    data[3] = pulseIn(pinIR, LOW);
    data[4] = pulseIn(pinIR, LOW);
    data[5] = pulseIn(pinIR, LOW);
    data[6] = pulseIn(pinIR, LOW);
    data[7] = pulseIn(pinIR, LOW);
    data[8] = pulseIn(pinIR, LOW);
    data[9] = pulseIn(pinIR, LOW);
    data[10] = pulseIn(pinIR, LOW);
    data[11] = pulseIn(pinIR, LOW); // array data completo

    for(int i=0;i<11;i++) {        // verifica tutti i dati
        if(data[i] > bin_1) {     // se il valore supera la soglia dell'1
            data[i] = 1 ;         // allora mette 1 nell'array
        }
        else {

```

```

                data[i] = 0;
            }
        }//for i
        // come fare la conversione binario decimale
        int risultato = 0;
        int Base2 = 1;
        for(int i=0;i<11;i++) {
            if(data[i] == 1) {
                risultato=risultato+Base2;
            }
            Base2 = Base2 * 2;           // questa istruzione permette il calcolo del
                                        // peso in base 2 necessario per la conversione
        }
        return risultato;              // restituisci risultato codice
    }

```

Utilizzando questo codice quindi siamo in grado di associare ad ogni tasto del telecomando che abbiamo in dotazione, in questo caso quello della DEA, un numero intero.

Lista codici associati ai tasti:

tasto avanti=129;

tasto indietro=132;

tasto dietro_orario=128;

tasto dietro_antiorario=130;

tasto avanti_antiorario=131;

tasto avanti_orario=133;

tasto stop=149;

tasto 1=148;

tasto 2=186;

tasto 3=191;

tasto 4=165;

tasto 5=184;

tasto 6=171;

tasto 7=244;

tasto 8=252;

tasto 9=151;

tasto 10=245;

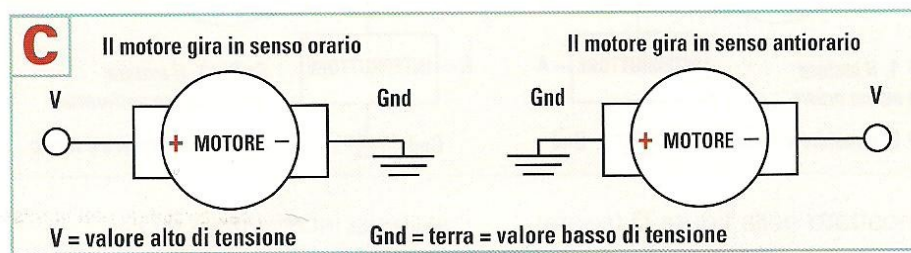
tasto 11=225;
tasto 12=188;
tasto 13=146;
tasto 14=144;
tasto 15=182;
tasto 16=147;
tasto 17=145.
apri=134;
su=135;
giù=137;
chiudi=136;
prg=157;

Detto questo sarà possibile utilizzare questi valori interi per far svolgere le funzioni volute.

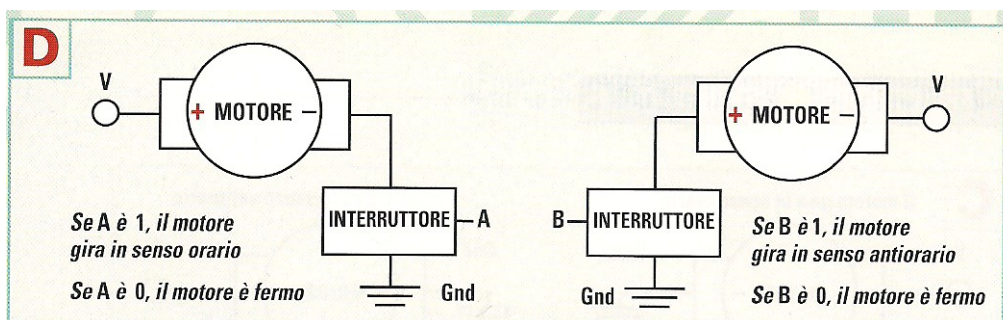
CONTROLLO MOTORI CON ARDUINO E DRIVER L293D

Controllo motori con alimentazione diretta

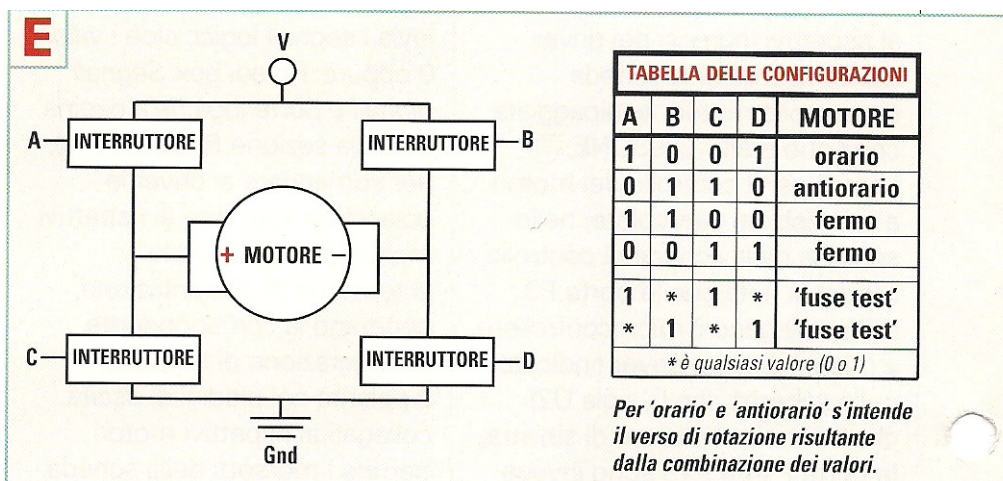
I motori in dotazione nei nostri laboratori di Sistemi sono motorini in cc che invertono il senso di marcia a seconda della polarità dell'alimentazione (principio comune a tutti i motori cc).



Per poterne comandare anche l'arresto si può pensare di utilizzare due interruttori



Proviamo ora a combinare insieme i due circuiti della figura **D**, per ottenerne uno solo grazie al quale sia anche possibile invertire il senso di rotazione di uno stesso motore. Quello che si ottiene è il circuito riportato nella figura **E**, che ha ora quattro interruttori comandati disposti a formare una lettera H: per questo tale configurazione viene detta 'a ponte H'.

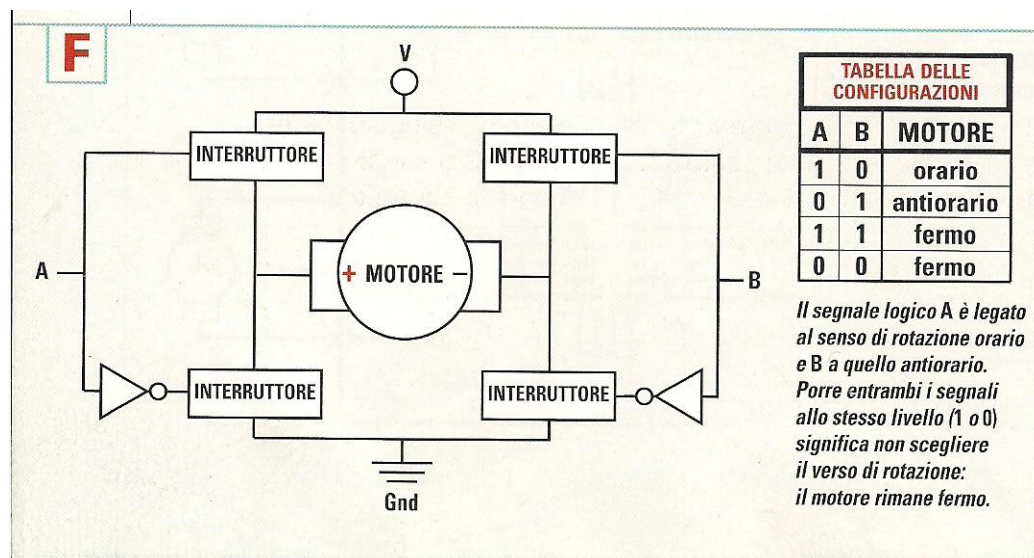


Il funzionamento del motore dipende ora da quattro valori logici (A, B, C e D). La tabella a destra dello schema riassume i possibili comportamenti del motore, a seconda delle diverse configurazioni di ingressi.

Dalla tabella risulta che, se si attivano (ponendo il valore 1) coppie di ingressi incrociati (A e D oppure B e C), si realizza una delle due situazioni rappresentate dagli schemi della figura C, in cui il motore gira in senso orario oppure antiorario. Quando invece si attivano le coppie di ingressi allineati (A e B oppure C e D), il motore resta fermo poiché scollegato dall'alimentazione (V) o dalla terra (Gnd).

Infine, le configurazioni che attivano le coppie di ingressi dello stesso lato (A e C oppure B e D) determinano, nello stesso punto del circuito, valori di tensione diversi, che provocano un cortocircuito, molto dannoso per il circuito.

Per evitare tali configurazioni pericolose, dette di 'fuse test', il circuito deve essere modificato ulteriormente, riducendo i valori logici di comando del ponte H da quattro a due. E' quello che si può osservare nel circuito schematizzato nella figura F



dove è stato inserito un nuovo componente chiamato 'porta NOT'.

Si tratta di una porta logica con funzione di negazione, in grado di garantire che gli ingressi degli interruttori posti sullo stesso lato abbiano sempre e solo valori opposti: se per esempio A è 1 l'interruttore in alto a sinistra riceverà tale valore, mentre quello in basso a sinistra riceverà la negazione del valore di A, cioè 0.

IL CHIP L293D

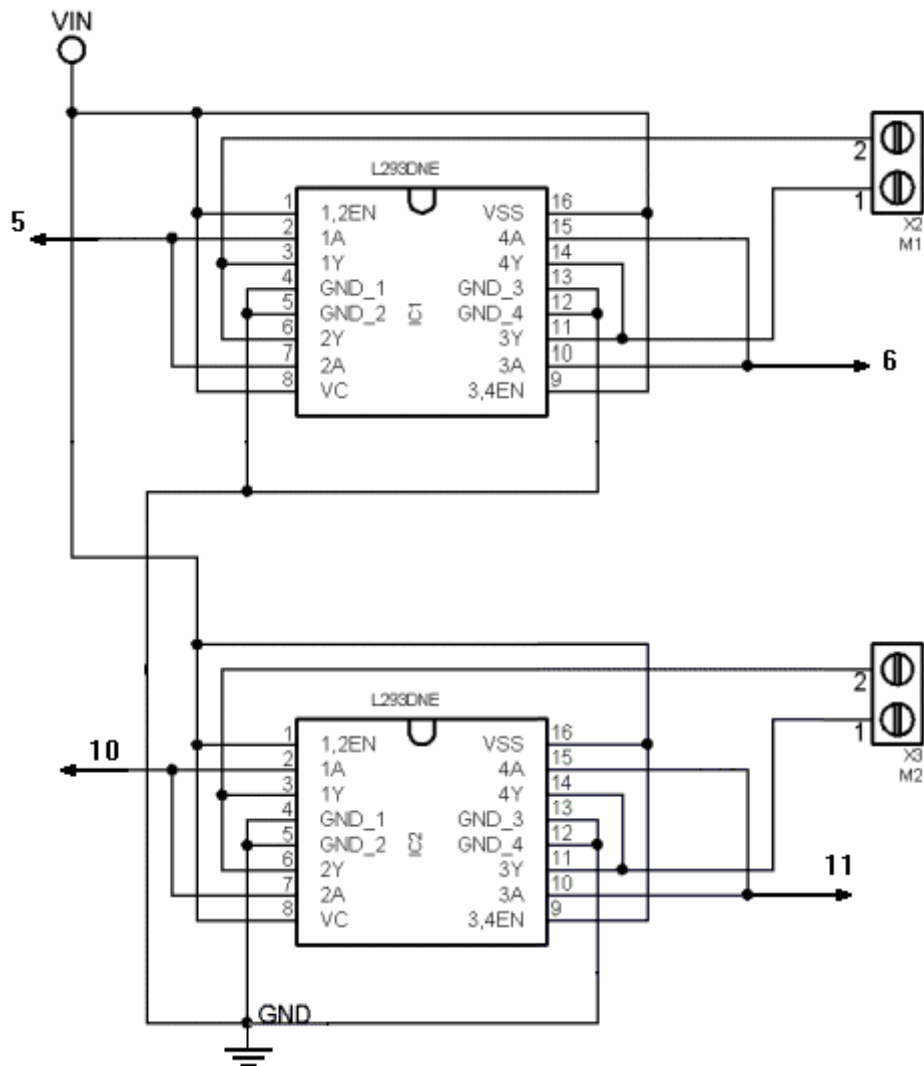
Il chip L293DNE è uno degli elementi che compongono la scheda di controllo motori e che funge da intermediario tra i motori che devono essere posti in movimento e il microcontrollore che invia i comandi necessari, per questo chiamato driver.

I driver L293DNE sono costituiti da 4 mezzi ponti H e sono progettati per fornire corrente bidirezionale fino a 600mA a tensioni tra 4,5V e 36V. Progettato per pilotare carichi induttivi come relè, solenoidi, motori in cc e motori bipolari passo passo, come altri dispositivi alta I/altaV in alimentazione non alternata.

Tutti gli ingressi sono compatibili con la logica TTL.

Lavorano a temperature da 0°C fino a 70°.

lo schema circuitale prevede il pilotaggio di un motore con un proprio drive L293D per non incorrere in problematiche legate al valore di corrente massima dissipabile qualsiasi sia la coppia da fornire.



Codice di programmazione per Arduino

Programma di test

/ I define sono da preferire rispetto all'assegnazione di costanti intere in quanto non occupano spazio in memoria del controllore.
Sono direttive che vengono date al compilatore.
Per esser certi dei collegamenti fare la prova inviando il software e verificando che parta con l'andare in avanti. */*

```
#define MotSxA 5
#define MotSxB 6
#define MotDxA 10
#define MotDxB 11
```

/ Questo programma permette ad un robot di seguire un percorso a nostro piacere assegnando valori differenti di valori di velocità. Permette di variare il numero assegnato ad una data velocità in una sola posizione senza per questo ricercarne altre.*

Vengono richiamati le funzioni di moto.

**/*

```
int normale_dx=95;
int normale_sx=95;           //valori che vanno testati per andare in avanti
int veloce_sx=105;         //valori da valutare di volta in volta
int lento_sx=50;
int lento_dx=50;
int veloce_dx=105;
```

//definizione dei pin di uscita e ingresso

```
void setup(){
    pinMode(MotSxA, OUTPUT);
    pinMode(MotSxB, OUTPUT);
    pinMode(MotDxA, OUTPUT);
    pinMode(MotDxB, OUTPUT);
}
```

```
void loop(){
    avanti();           //chiamata del metodo avanti()
    delay(2000);       //rimane con il comando di avanti per 2 secondi
```

```

fermo();
delay(1000);           // fermo per 1 secondo
orario();
delay(3000);          // gira in senso orario per 3 secondi
fermo();
delay(1000);          // fermo per 1 secondo
antiorario();
delay(3000);          // gira in senso antiorario per 3 secondi
indietro();
delay(2000);          // indietro per 2 secondi
fermo();
delay(1000);
} // loop

void fermo(){
    digitalWrite(MotSxA,LOW);
    analogWrite(MotSxB,LOW);
    analogWrite(MotDxA,LOW);
    digitalWrite(MotDxB,LOW);
} //fermo

void avanti(){
    digitalWrite(MotSxA,LOW);           // sinistro antiorario
    analogWrite(MotSxB,normale_sx);
    analogWrite(MotDxA,normale_dx);     // destro orario
    digitalWrite(MotDxB,LOW);
} //avanti()

void orario(){
    digitalWrite(MotSxA,LOW); //sinistro antiorario
    analogWrite(MotSxB,veloce_sx);
    analogWrite(MotDxA,lento_dx); //destro orario
    digitalWrite(MotDxB,LOW);
} //orario

void antiorario(){
    digitalWrite(MotSxA,LOW);           //sinistro antiorario
    analogWrite(MotSxB,lento_sx);

```

```

    analogWrite(MotDxA,veloce_dx);           //destro orario
    digitalWrite(MotDxB,LOW);
} // antiorario

void indietro(){
    digitalWrite(MotSxA,indietro_Sx);       // sinistro orario
    analogWrite(MotSxB,HIGH);
    analogWrite(MotDxA,HIGH);               // destro antiorario
    digitalWrite(MotDxB,indietro_Dx);
} // indietro

```

Tutte le funzioni utilizzano la funzione analogWrite() che funziona in questo modo:

Sintassi

```
analogWrite(pin,valore);
```

Il parametro pin rappresenta il pin su cui scrivere;

il parametro valore rappresenta il valore del duty cycle tra 0 (sempre off) e 255(sempre on).

Manca il disegno della disposizione elettronica dei driver e della dislocazione dei motori

DS1620: termometro digitale

Descrizione del chip DS1620

Il chip Ds1620 è un termometro digitale completo, capace di sostituire i le normali combinazioni tra sensori di temperatura e convertitori analogico-digitali nella maggior parte delle applicazioni.

Può misurare temperature da -55°C fino a +125°C con una risoluzione di 0,5°C.

I valori di temperatura sono espressi con parole di nove bit secondo la notazione del complemento a due.

Il DS1620 comunica con un microcontrollore così come con un PIC oppure una Stamp attraverso una connessione seriale a tre fili.

Il DS1620 è capace di operare come termostato standalone cioè indipendentemente da altro hardware o software.

Tensione di alimentazione coperta tra 2.7 V e 5.5 V. (Vdd da 3 a 5V)

La temperatura viene convertita in parola digitale in 750ms massimo

Il settaggio termostatico è definitivo e non volatile in quanto va a finire nella EPROM

Le applicazioni includono controlli termostatici, sistemi industriali, termometri e ogni sistema sensibile alla temperatura.

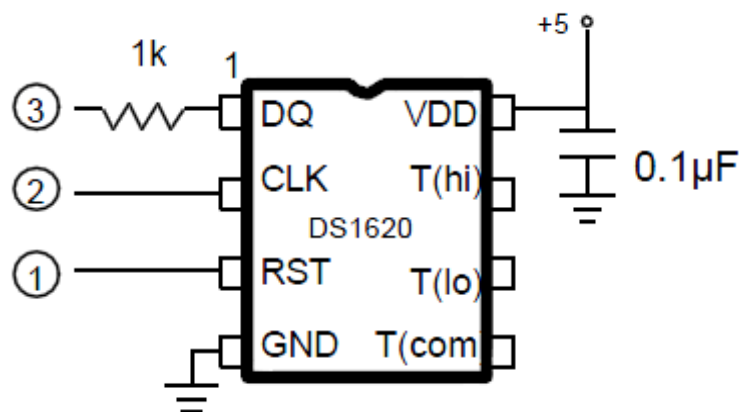
Hardware

Il DS1620 è costituito da una linea dati I/O a 3 vie indicata con DQ, una linea a 3 vie di clock di sincronizzazione indicata con CLK e una linea di reset/select indicata con RST anch'essa a 3 vie (ecco perché si necessita di una parola a 9 bit per leggere la temperatura).

La presenza del condensatore (di valore 0.1microFarad) in figura è dovuta al fatto che l'alimentazione deve risultare costante e ben filtrata. Si consiglia di posizionare il condensatore quanto più vicino possibile all'alimentazione del DS1620.

Sebbene il resistore da 1K non sia strettamente necessario è preferibile metterlo.

Se per esempio il microcontrollore e il DS1620 tentano di inviare dati sulla linea allo stesso istante, il resistore limita la quantità di corrente che potrebbe fluire tra di essi a valori di sicurezza.



Con le tre uscite di allerta il DS1620 può anche funzionare da termostato standalone.

T_{HIGH} è imposto alto se la temperatura è maggiore o uguale di una data temperatura prefissata T_H .

T_{LOW} è imposta alta se la temperatura è minore o uguale a una data temperatura prefissata T_L .

T_{COM} è pilotato a valore alto quando la temperatura del DS1620 eccede T_H e rimane alto fino a quando la temperatura non diminuisce al disotto di T_L .

Funzionamento

Il DS1620 misura la temperatura usando un sensore di temperatura basato su bandgap.

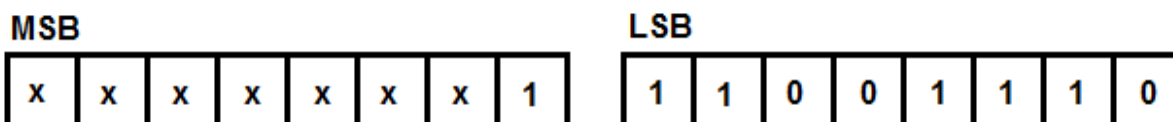
La temperatura letta è fornita da 9 bit, complemento a due rilasciata dal comando di READ TEMPERATURE. Il dato è trasmesso serialmente attraverso un'interfaccia seriale a 3 vie, a partire dall'LSB.

Il Ds1620 rilascia i valori in °C, da -55°C a +125°C.

Poiché i dati sono trasmessi su un bus a 3 vie, i valori letti o scritti sono parole di 9 bit (rendendo basso RST dopo la ricezione del bit MSB), o anche come trasferimento di due byte, con i 7 bit più significativi ignorati oppure posti a 0.

La temperatura è rappresentata in DS1620 in termini di 0,5°C LSB.

Formato di memorizzazione della temperatura TH e TL



T = -25°C

TEMPERATURE/DATA RELATIONSHIPS Table 3

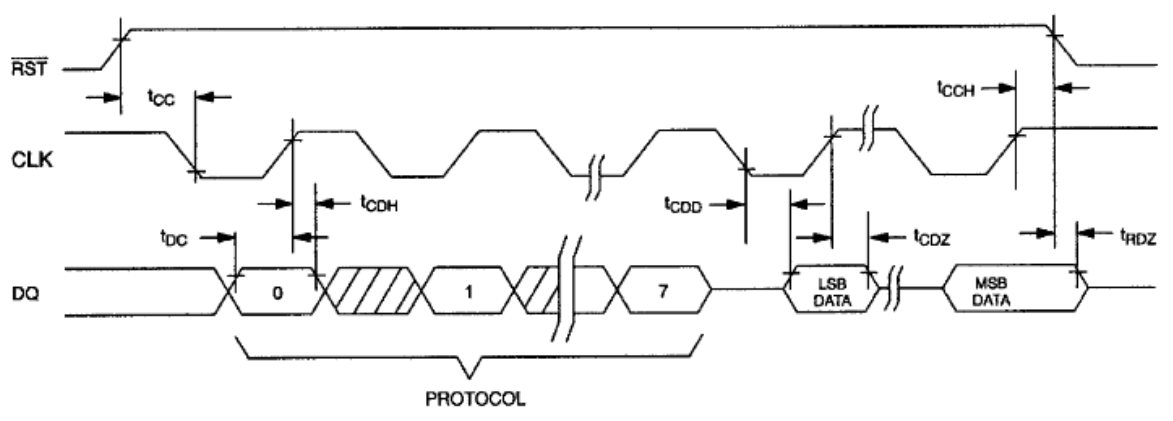
TEMP	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	0 11111010	00FA
+25°C	0 00110010	0032h
+½°C	0 00000001	0001h
+0°C	0 00000000	0000h
-½°C	1 11111111	01FFh
-25°C	1 11001110	01CEh
-55°C	1 10010010	0192h

Comunicazione sul bus a 3 vie

Il bus a tre vie è compreso di tre segnali. Questi sono RST (reset), CLK (segnale di clock) e segnale dati DQ.

Il trasferimento dei dati viene inizializzato pilotando a livello alto l'input di RST.

READ DATA TRANSFER Figure 4



Pilotando a livello basso l'ingresso RST la comunicazione verrà terminata. Un ciclo di clock è una sequenza di fronti di discesa seguiti da fronti di salita tra due valori costanti.

Per i bit dati in ingresso, il dato dovrà essere rimanere valido durante il fronte di salita di un ciclo di clock.

I bit dati sono posti in uscita durante il fronte di discesa del clock e rimangono a disposizione durante il fronte di salita.

Quando leggiamo i dati dal DS1620, il pin DQ dovrà essere in alta impedenza mentre il clock è alto. Rendendo RST basso saranno terminate tutte le comunicazioni e questo

causerà la messa in alta impedenza del pin DQ.

Set dei comandi per la lettura della temperatura del DS1620		
Istruzione	Descrizione	Comando in esadecimale
Letture della temperatura	Questo comando legge il contenuto del registro che contiene il risultato dell'ultima conversione di temperatura. I prossimi 9 cicli di clock restituirà il contenuto di questo registro.	[AAh]
Letture del contatore	Questo comando legge il valore del contatore di byte. I prossimi nove cicli di clock daranno in uscita il contenuto di questo registro.	[A0h]
Letture della pendenza	Questo comando legge il conteggio della pendenza del DS1620. I prossimi 9 cicli di clock restituiranno questo conteggio.	[A9h]
Inizio della conversione di temperatura	Questo comando dà inizio alla conversione della temperatura. Non sono richiesti altri dati. In un colpo solo la conversione della temperatura sarà eseguita e quindi il DS1620 rimarrà in ozio. In modalità di conversione continua questo comando inizierà continuamente una nuova conversione.	[EEh]
Stop della conversione della temperatura	Questo comando blocca la conversione della temperatura. Non sono richiesti altri dati. Questo comando può essere usato per fermare il DS1620 quando di trova in modalità di conversione continua. Dopo l'emissione di questo comando la temperatura corrente misurata sarà completata e quindi il DS1620 rimarrà in ozio fino a che Start Convert T è rilanciato per ricominciare l'operazione continua.	[22h]

Codice di programma con Arduino

/*

Esempio di applicazione del DS1620

Letture della temperatura con il DS1620 in funzionamento di termometro digitale e risultato scritto sul serial monitor del PC.

Collegamenti con Arduino:

Arduino DS1620

pin 3 ---> RST

pin 4 ---> CLK

pin 5 ---> DQ

*/

```
#define RST 3
```

```
#define CLK 4
```

```
#define DQ 5

void setup() {
    pinMode(RST, OUTPUT);           // impostazione dei pin del DS1620
    pinMode(CLK, OUTPUT);           // come OUTPUT
    pinMode(DQ, OUTPUT);
    Serial.begin(9600);              // impostazione comunicazione seriale PC
}
```

```
void loop() {
    RST_basso();
    CLK_alto();
    RST_alto();
    invia_comando(0xEE);            // comando di inizio conversione della
                                    // temperatura

    RST_basso();
    delay(200);
    CLK_alto();
    RST_alto();
    invia_comando(0xAA);            // comando di lettura ultima
                                    // conversione effettuata

    int raw_data = leggi_raw_data(); //i raw-data sono i dati grezzi
    RST_basso();

    Serial.print("temperatura: ");
    Serial.print(raw_data/2);        //in quanto 0.5°C???
    Serial.println(" C");
    delay(100);
}
```

```
void invia_comando (int comando){
```

/ invia 8 bit di comando sull'uscita DQ output, dal LSB preso uno alla volta. Per fare questo basta fare lo shift verso destra del comando proprio della posizione n, si porta il bit dalla posizione n alla posizione 0 e poi si fa & (and tra bit non && logico) con 1 in binario */*

```
for(int n=0; n<8; n++) {
```

```

        int bit = ((comando >> n) & (0x01));    // prendo i bit della parola
                                                // comando uno alla volta
        digitalWrite(DQ, bit);                // e li invio a DQ
        CLK_basso();                          // fornisco un nuovo impulso di clock
        CLK_alto();
    } //for
} // invia comando

```

```

int leggi_raw_data(){
    int bit,n;
    int raw_data=0;
    pinMode(DQ,INPUT);    // ora DQ diventa input dato che con Arduino
                          // si vuol conoscere i valori di temperatura
    for(int n=0;n<9;n++) {
        CLK_basso();
        bit=(digitalRead(DQ));
        CLK_alto();
        raw_data = raw_data | (bit << n);
    }
    pinMode(DQ, OUTPUT);    // DQ di nuovo impostato come uscita per
                          // poter inviare i comandi necessari
    return(raw_data);      // misurazione restituita
}

```

```

void CLK_alto(){
    digitalWrite(CLK,HIGH); //metti a livello alto il pin di clock
} // CLK_alto

```

```

void CLK_basso() {
    digitalWrite(CLK,LOW); //metti a valore basso il pin di CLK
} // CLK_basso

```

```

void RST_alto(){
    digitalWrite(RST,HIGH); // metti a valore alto il pin di RST
} // RST_alto

```

```

void RST_basso() {
    digitalWrite(RST,LOW); // metti a valore basso il pin di RST
}

```

} //RST_basso

INSEGUITORE DI TRACCIA CON 3 CNY70

Sensori CNY70

Il sensore CNY70 è un sensore di tipo ottico.

Al suo interno sono presenti un diodo emettitore ad infrarosso (che lavora su una lunghezza d'onda di 950 nm) ed un fototransistor.

La distanza di lettura si aggira sui 0,3 mm.

Funzionamento ed utilizzo

All'interno della capsula di questo sensore è montato un diodo LED che emette raggi infrarossi, invisibili all'occhio umano. Il diodo è dotato di due terminali, l'anodo (A) ed il catodo (K).

Sulla stessa superficie è ubicato un fototransistor che ha la proprietà di condurre corrente fra l'emettitore (E) e il collettore (C), proporzionale alla quantità di luce che incide sulla base.

Dato che sia l'emettitore sia il ricevitore dei raggi sono disposti sulla stessa superficie, è necessario che davanti ad entrambi sia presente una superficie riflettente, per fare in modo che il fototransistor possa ricevere i raggi che genera il led.

La superficie riflettente deve essere situata a pochi millimetri da quella su cui sono montati emettitore e ricevitore, per far sì che i raggi riflessi abbiano sufficiente intensità.

Electrical Characteristics ($T_{amb} = 25^{\circ}\text{C}$)

Input (Emitter)

Parameter	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Forward voltage	$I_F = 50 \text{ mA}$	V_F		1.25	1.6	V

Output (Detector)

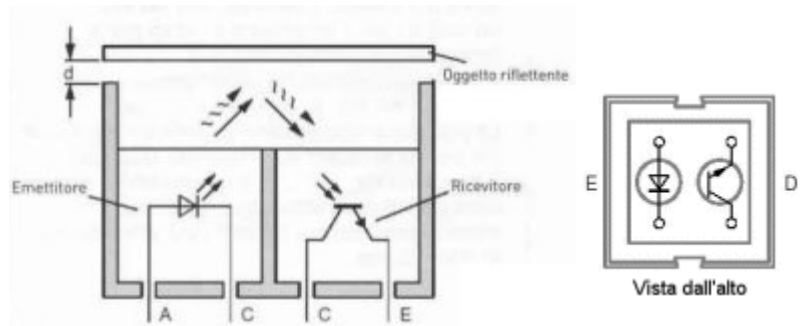
Parameter	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Collector emitter voltage	$I_C = 1 \text{ mA}$	V_{CEO}	32			V
Emitter collector voltage	$I_E = 100 \mu\text{A}$	V_{ECO}	5			V
Collector dark current	$V_{CE} = 20 \text{ V}, I_F = 0, E = 0$	I_{CEO}			200	nA

Coupler

Parameter	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Collector current	$V_{CE} = 5 \text{ V}, I_F = 20 \text{ mA}, d = 0.3 \text{ mm}$ (figure 1)	$I_C^{1)}$	0.3	1.0		mA
Cross talk current	$V_{CE} = 5 \text{ V}, I_F = 20 \text{ mA}$ (figure 1)	$I_{CX}^{2)}$			600	nA
Collector emitter saturation voltage	$I_F = 20 \text{ mA}, I_C = 0.1 \text{ mA}, d = 0.3 \text{ mm}$ (figure 1)	$V_{CEsat}^{1)}$			0.3	V

¹⁾ Measured with the 'Kodak neutral test card', white side with 90% diffuse reflectance

²⁾ Measured without reflecting medium



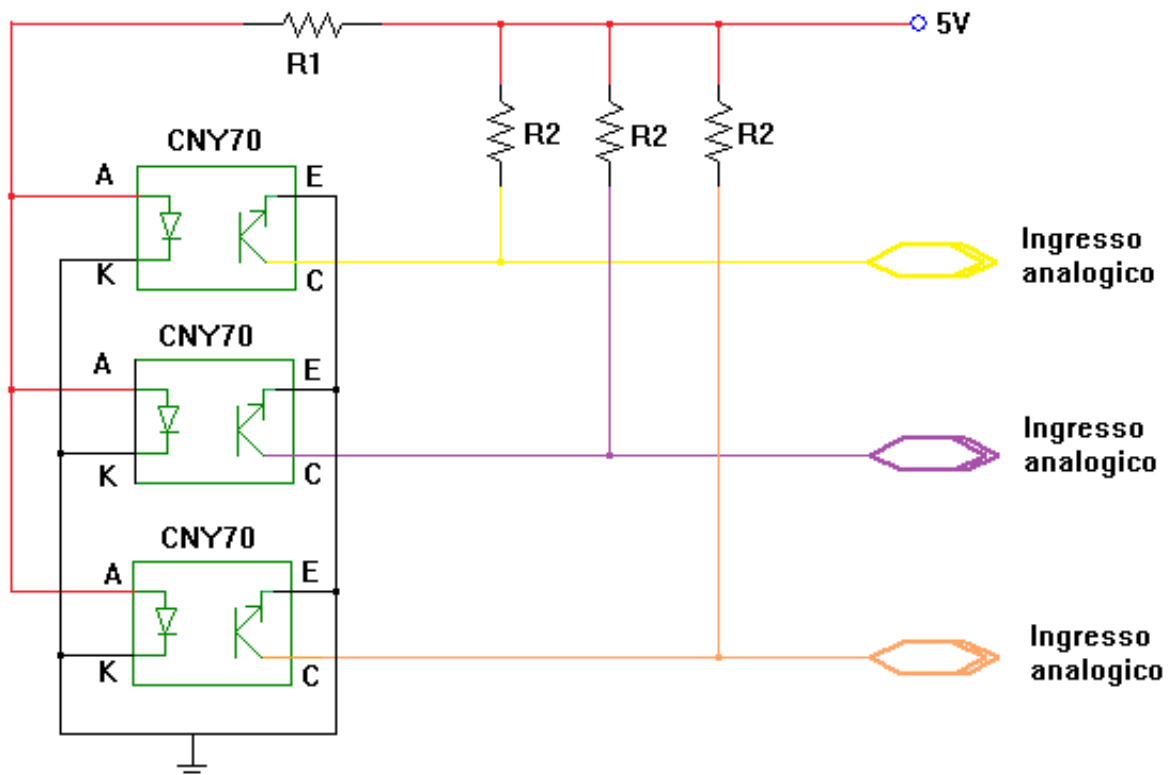
Inseguitore di traccia con Arduino

Il circuito utilizzato per costruire l'inseguitore di traccia segue queste indicazioni:

- il catodo del fotoemettitore e l'emettitore del fototransistor sono sempre collegati a massa;
- l'anodo è alimentato tramite $R1=220\text{ Ohm}$ per cui i fotoemettitori sono costantemente polarizzati direttamente
- il collettore viene alimentato tramite $R2=39\text{kOhm}$ dalla tensione di 5V.

I diodi si trovano sempre ad emettere luce IR, mentre i fototransistor forniscono una tensione variabile in funzione della luce riflessa dalla traccia.

Quando la luce emessa dal diodo incide, per riflessione, la base del transistor circola una I_c che cresce al crescere dell'irradiazione incidente. La V_{ce} di conseguenza diminuisce se



la Ic aumenta.

Codice di programmazione per Arduino

Programma di test per 3 sensori CNY70

/ programma che serve per codificare i valori letti dal CNY70 in corrispondenza della traccia da seguire.*

Questo programma consente di individuare un buon valore di soglia per il riconoscimento del colore della traccia.

*I pin digitali 0 e 1 sono dedicati alla comunicazione seriale tra Arduino e PC: non si possono utilizzare. */*

```
int sensore1=0;           // collettore del CNY70 1 su pin A0 analogico di Arduino
int sensore2=1;           // collettore del CNY70 2 su pin A1 analogico di Arduino
int sensore3=2;           // collettore del CNY70 3 su pin A2 analogico di Arduino
```

```
void setup(){
    Serial.begin (9600);    // impostazione velocità di trasmissione in bps
                          // i pin analogici non vengono impostati nel metodo setup().
}
```

```
void loop(){
    float valore1 = analogRead (sensore1);
    float valore2 = analogRead (sensore2);
    float valore3 = analogRead (sensore3);
    Serial.println("Valore del sensore 1: ");
    Serial.println(valore1);
    Serial.println("Valore del sensore 2: ");
    Serial.println(valore2);
    Serial.println("Valore del sensore 3: ");
    Serial.println(valore3);
    delay(500);           // fermiamo il programma per dar modo di leggere i valori
} //loop
```

Inseguitore di traccia con 3 CNY70 e Arduino

/ I define sono da preferire rispetto all'assegnazione di costanti intere in quanto non occupano spazio in memoria del controllore. Sono direttive che vengono date al compilatore. Non possono essere utilizzate però per ulteriori*

manipolazioni. */

```
#define MotSxA 5 // motore sinistro A collegato con il pin 5 tipo PWM
#define MotSxB 6 // motore sinistro B collegato con il pin 6 tipo PWM
#define MotDxA 10 // motore destro A collegato con il pin 10 tipo PWM
#define MotDxB 11 // motore destro B collegato con il pin 11 tipo PWM
```

// ---- pin di tipo analogico per inseguitore di traccia ----

```
#define line_sx 0
#define line_centro 1
#define line_dx 2
```

/ Questo programma permette ad un robot di seguire un percorso fatto da traccia di colore nera anche se si può estendere anche per altri colori di tracce. Assegnare ad una variabile differenti valori di velocità permette di variarne il valore solo in un punto del codice senza preoccuparsi delle sostituzioni ricorrenti.**

```
int normale_dx=95;
int normale_sx=95; // valori che vanno testati per andare dritto in avanti
int veloce_sx=105; // valori da valutare di volta in volta
int lento_sx=50;
int lento_dx=50;
int veloce_dx=105;
int calibrazione=800; // nel caso di linea nera sembra un buon valore di soglia
float val_sx, val_centro, val_dx; // variabili di tipo float legate alla
// risposta dei CNY70
```

```
void setup(){
    pinMode(MotSxA, OUTPUT);
    pinMode(MotSxB, OUTPUT);
    pinMode(MotDxA, OUTPUT);
    pinMode(MotDxB, OUTPUT);
    // i pin analogici A0, A1 e A2 non vanno impostati nel setup()
}
```

```
void loop(){
    segui_traccia(); // chiamata del metodo segui_traccia
} //loop
```



```

void segui_traccia(){
    val_sx=analogRead(line_sx);           // leggi valore del CNY70 sinistro
    val_centro=analogRead(line_centro); // leggi valore del CNY70 di centro
    val_dx=analogRead(line_dx);           // leggi valore del CNY70 destro

    /* Se tutti e tre i sensori sono interessati da riflessione, quindi si trovano tutti e
       tre sulla striscia il robot deve andare avanti. */

    if(val_sx>calibrazione && val_centro>calibrazione && val_dx>calibrazione){
        avanti();
    }//if

    /* se il sinistro è fuori traccia il robot deve ruotare in senso orario e ritrovare
       la traccia */

    else if( val_dx>calibrazione && val_centro>calibrazione &&val_sx <calibrazione){
        orario();
    }//else

    /* se il destro è fuori traccia il robot deve ruotare in senso antiorario e ritrovare
       così la traccia */
    else if( val_dx<calibrazione && val_centro>calibrazione && val_sx>calibrazione){
        antiorario();
    }//else

    else {
        fermo(); // se il robot perde la traccia si ferma
    }
} //seguì_traccia

void fermo(){
    digitalWrite(MotSxA,LOW);
    analogWrite(MotSxB,LOW);
    analogWrite(MotDxA,LOW);
    digitalWrite(MotDxB,LOW);
} //fermo

```

```

void avanti(){
    digitalWrite(MotSxA,LOW);           // sinistro antiorario
    analogWrite(MotSxB,normale_sx);
    analogWrite(MotDxA,normale_dx);    // destra orario
    digitalWrite(MotDxB,LOW);
}//avanti()

void orario(){
    digitalWrite(MotSxA,LOW);           // sinistro antiorario
    analogWrite(MotSxB,veloce_sx);
    analogWrite(MotDxA,lento_dx);      // destra orario
    digitalWrite(MotDxB,LOW);
}//orario

void antiorario(){
    digitalWrite(MotSxA,LOW);           // sinistro antiorario
    analogWrite(MotSxB,lento_sx);
    analogWrite(MotDxA,veloce_dx);     // destra orario
    digitalWrite(MotDxB,LOW);
}//antiorario

```

IL DISPLAY SERIALE LCD

Il display SerLCD della Spark fun

Questo display è una semplice ed economica soluzione di interfacciamento con un LCD (display a cristalli liquidi).



Il collegamento seriale raggiunge i 38400 baud, l'hardware utilizza un nuovo chipset a larga diffusione e costo contenuto. Per quanto riguarda il controllo basta inviare sulla linea di comunicazione il testo che si vuole visualizzare per vederlo comparire sullo schermo.

Per interfacciarsi con questo LCD sono necessari solo 3 fili: l'alimentazione a 5V, la massa e il filo per il segnale.

Caratteristiche

- Il pic 16f688 utilizzato sulla scheda migliora la precisione della comunicazione;
- velocità di comunicazione selezionabile di 2400, 4800, 9600 (default), 14400, 19200 e 38400;
- velocità di processione fino a 8MHz
- veloce tempo di accensione
- accensione/spegnimento del display via firmware

Interfaccia di comunicazione

Il SerLCD è controllato utilizzando caratteri ASCII. Questo significa che se invii il carattere 'r' , 'r' verrà visualizzato. A meno di due eccezioni però: il comando dei caratteri decimali 254 (0xFE) e 124 (0x7C).

Configurazione

Tutte le impostazioni sono memorizzate sulla EEPROM presente sulla board e caricate durante l'accensione

Retroilluminazione

Il SerLCD modula la retroilluminazione tramite un transistor BJT a 1A. Questo permette di impostare 30 differenti valori di intensità luminosa.

Backlight Brightness	
Value	Brightness
128	Off
140	40% On
150	73% On
157	Fully On
158	Not Valid

Inviando il comando speciale 0x7C (124 decimale) seguito dai numeri che vanno da 128 a 157 si imposterà il valore della PWM legata alla retroilluminazione.

Questo risulta comodo quando il consumo di potenza dell'unità deve essere minimizzato. Riducendo l'intensità luminosa il consumo di corrente verrà ridotto.

Il tipo in dotazione è un LCD 2x16 .

Comandi aggiuntivi

Comandi LCD estesi

Azzeramento del display	0x01
Sposta il cursore a destra di un carattere	0x14
Spostare il cursore a sinistra di un carattere	0x10
Effettua uno scroll verso destra	0x1C
Effettua uno scroll verso sinistra	0x18
Accende il display	0x0C
Spegne in display	0x08
Sottolinea il cursore	0x0E
Elimina la sottolineatura del cursore	0x0C
Attiva il lampeggiamento del cursore	0x0D
Disattiva il lampeggiamento del cursore	0x0C
Impostare la posizione del cursore	0x80 +

Cancellare il display e impostare la posizione del cursore sono i due comandi più ricorrenti: per mezzo di questi comandi si può modificare la posizione del cursore.

Questi cambiamenti sono sempre controllati dal firmware.

Il cursore si muove all'interno di tutta l'area guardabile per cui la sua posizione dovrà essere determinata con precisione.

Per far compier al cursore un certo movimento, nell'area osservabile, occorre seguire diversi passi:

- Occorre determinare la corretta posizione decimale dove muovere. Per esempio la posizione tre sulla seconda linea di un display a 16 caratteri corrisponde al numero 66 decimale.
- Setta a 1 il bit MSB di quel numero. Posizione $66+128=194$
- Ora trasmetti il carattere speciale 254 per dire al SerLCD che vuoi inviare un

comando.

- Alla fine, trasmetti 194. Il cursore adesso si troverà sulla terza posizione della seconda linea.

16 Character Displays	
Line Number	Viewable Cursor Positions
1	0-15
2	64-79
3	16-31
4	80-95

Splash screen

Il SerLCD visualizza lo splash screen per default. Questo significa che per default verifica che tutte le unità vengano alimentate e lavorino correttamente.

Lo splash screen è visualizzato per 500ms alla partenza e può essere richiamato se desiderato.

Per disabilitare questa funzione inviare il comando speciale 0x7C (124 decimale) all'unità facendolo seguire dal numero 9. Ogni volta che questo comando è inviato al display verrà variata la condizione dello splash Screen. Per cui inviando i comandi 0x7C 0x09 verrà disabilitato splash screen durante il prossimo boot. Inviando nuovamente 0x7C 0x09 splash screen verrà abilitato nuovamente.

Modifica del valore della velocità (baud rate)

Per default la velocità di comunicazione del SerLCD è 9600 baud ma può essere variato anche a differenti velocità.

Per cambiare la velocità prima si scrive il comando speciale 124, quindi per:

- 2400 baud, inviare "<control>k"
- 4800 baud, inviare "<control>l"
- 9600 baud, inviare "<control>m"
- 14400 baud, inviare "<control>n"
- 19200 baud, inviare "<control>o"
- 38400 baud, inviare "<control>p"

Se il SerLCD si trova in uno stato sconosciuto, o non puoi comunicare con esso alla velocità di settaggio, invia il comando "<control>r" a 9600 baud mentre splash screen è attivo e l'unità verrà resettata a 9600 baud.

Hardware

Questo dispositivo può essere alimentato solo a 5V in corrente continua. Una tensione

maggiore di 5.5V causerà il danneggiamento del PIC, dell'LCD e della retroilluminazione se collegata.

Il SerLCD utilizza 3mA con la retroilluminazione spenta e 60mA quando invece è attivata.

Controllo di contrasto

Il SerLCD è equipaggiato con un potenziometro di 10K che controlla il contrasto dell'LCD. Questo è settato durante l'assemblaggio e il collaudo ma può essere se necessaria una correzione per il proprio modulo LCD.

La temperatura e la tensione di alimentazione possono provocare effetti sul contrasto dell'LCD.

Pin di controllo dell'alta corrente

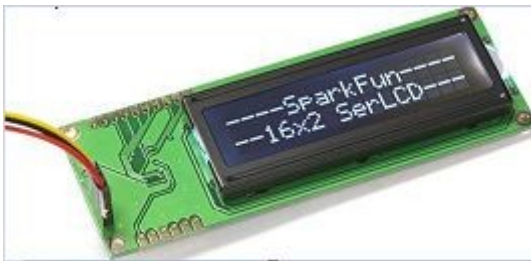
Il SerLCD utilizza un transistor NPN a 1A per controllare la retroilluminazione.

Programmi con Arduino e serLCD

Test del SerLCD con Arduino

In questo esempio non viene specificato alcun pin in quanto i dati vengono inviati all'LCD tramite il pin TX di Arduino quindi tramite un pin già predisposto per questo:

collegamento pin dell'LCD con Arduino



Vdd ---> 5V su Arduino

GND ---> GND su Arduino

Rx ---> Tx di Arduino

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

```

void loop()
{
    selezionaPrimaLinea(); // questa funzione dirà al cursore di spostarsi
                          // alla prima posizione valida della prima linea
                          // dell'LCD
    delay(100);           // aspetta 1ms
    Serial.print(millis()); // invia al LCD il valore in ms del tempo dall'inizio
                          // run
    selezionaSecondaLinea(); // cursore spostato nella seconda linea
    delay(100);
    Serial.print(millis()/2); // invia al LCD il valore di millis() /2
    delay(100);
}

void selezionaPrimaLinea() { // posiziona il cursore nella posizione 0,0
                          // prima linea, prima posizione.
    Serial.print(0xFE, BYTE); // il comando 254 (FEh), secondo il protocollo
                          // di comunicazione, pone l'LCD in attesa di
                          // ricevere un comando
    Serial.print(128, BYTE); // posizione 0 : al numero 0 in byte si
                          // imposta 1 come MSB, ciò corrisponde al
                          // valore 128 in decimale
}

void selezionaSecondaLinea() { // posiziona il cursore al primo carattere
    della seconda linea
    Serial.print(0xFE, BYTE); // opzione di comando
    Serial.print(192, BYTE); // posizione 64 → 64+128=192 come da
                          // protocollo
}

void retroilluminaOn() { // modifica la retroilluminazione dell'LCD
    Serial.print(0x7C, BYTE); // l'opzione di comando per agire sulla
                          // retroilluminazione è l'invio del 124 -->7Ch
    Serial.print(157, BYTE); // livello luminosità da 128 a 157
}

```

Altre funzioni utili:

```

void goTo(int position) { //position = line 1: 0-15, line 2: 16-31, 31+ defaults back

```

```

to 0
if (position<16){ Serial.print(0xFE, BYTE); //command flag
    Serial.print((position+128), BYTE); //position
}else if (position<32){Serial.print(0xFE, BYTE); //command flag
    Serial.print((position+48+128), BYTE); //position
} else { goTo(0); }
}

void clearLCD(){
    Serial.print(0xFE, BYTE); //command flag
    Serial.print(0x01, BYTE); //clear command.
}

void retroilluminaOff(){ //turns off the backlight
    Serial.print(0x7C, BYTE); //command flag for backlight stuff
    Serial.print(128, BYTE); //light level for off.
}

void serCommand(){ //a general function to call the command flag for issuing
all other commands
    Serial.print(0xFE, BYTE);
}

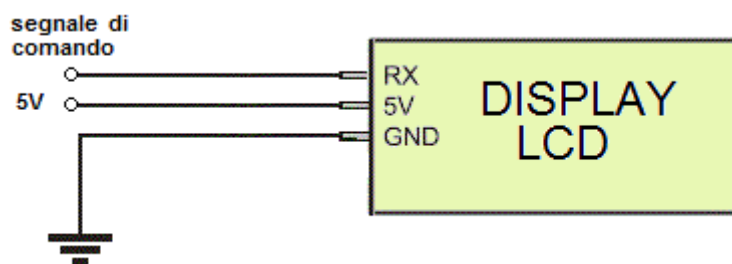
```

LCD parallax

L'LCD della parallax

- Lavora a velocità di 2400, 9600 e 19200 baud.
- É possibile spostare il cursore in qualsiasi posizione del display con un singolo comando
- Aggiunge le linee automaticamente per visualizzare facilmente sul display la stringa di testo
- cancella con un singolo comando quanto visualizzato

Connettere il display ad Arduino è veramente semplice basta seguire lo schema seguente:



Assorbe una corrente di 20mA con retroilluminazione spenta e di 80mA con retroilluminazione accesa.

Velocità di invio dati

Prima di connettere l'LCD seriale al dispositivo quale un microcontrollore, occorre selezionare la velocità con cui inviare i dati. Tre sono i valori possibili: 2400, 9600 e 19200 baud.

Per impostare la velocità occorre spostare le levette degli interruttori SW dislocati sul retro del display nella posizione corretta secondo quanto segue:

Mode	SW1	SW2
TEST	OFF	OFF
4200	ON	OFF
9600	OFF	ON
19200	ON	ON

Test del display

La condizione di test permette di verificare che alimentazione e massa siano correttamente collegate. Questa condizione deve essere verificata prima dell'invio dei dati.

Nella condizione di test devono essere visualizzate sul display le seguenti scritte:

Parallax, Inc.

www.parallax.com.

Se la scritta non si vede bene occorre agire con un cacciavite sul potenziometro "Increase contrast" anch'esso posto sul retro del dispositivo.

Se invece la scritta non si vede meglio allora conviene controllare le connessioni elettriche e riprovare.

Dopo aver completato con successo il test si procede ad abbassare le levette degli interruttori SW1 e SW2 in corrispondenza alla velocità prescelta.

Visualizzazione del testo

per visualizzare i caratteri del testo sul display LCD seriale basta inviare i caratteri in codice ASCII sulla porta seriale collegata all'LCD alla velocità stabilita.

Quando un carattere viene ricevuto, l'LCD lo visualizza nella posizione corrente del cursore e quindi muove il cursore di una posizione verso destra.

La prima posizione valida è quella relativa in fondo a sinistra sulla prima riga.

Quando la prima riga è completa il cursore scenderà nella posizione più a sinistra della seconda riga.

Quando anche la seconda sarà completa il cursore tornerà all'inizio della prima riga.

Funziona come ci si aspetta che funzioni.

La posizione del cursore viene visualizzata dalla piccola barra in fondo al carattere.

In questo modo possono essere inviati caratteri senza limite di numero.

Spostamento del cursore

Quando viene spedito un carattere al display seriale esso viene visualizzato nella posizione corrente del cursore.

Ci sono pochi modi per spostare il cursore sul display. Dopo la ricezione di un carattere il cursore si sposta automaticamente di una posizione. Oltre questa modalità ci sono altri comandi che includono backspace (torna indietro), ritorno a capo (carriage return), nuova linea (line feed).

Il comando backspace/sinistra (Dec8) sposta il cursore di un posto verso sinistra mentre il comando Dec 9 sposta il cursore di un posto verso destra. Questi possono essere utilizzati per sovrascrivere del testo già esistente.

Il comando Line feed , Dec 10, sposta il cursore sulla prossima linea del display senza modificare la posizione orizzontale del cursore.

Il comando Carriage Return , Dec 13 sposta il cursore all'inizio della linea successiva.

Il comando Form Feed, dec 12, cancella l'intero display e sposta il cursore nella posizione a sinistra sulla linea 0, come quando inizia il primo turno di scrittura. Per eseguire questo comando occorrono 5ms di pausa.

Ci sono altri comandi che consentono lo spostamento del cursore in ogni posizione del display. I comandi nel range Dec 128 fino a Dec 143 e Dec 148 fino a Dec 163 spostano il cursore nelle 16 posizioni differenti su ognuna delle due linee.

Controllo del display

Il comando display-off (Dec 21) spegne il display in modo che scompaiono tutti i caratteri. I caratteri non vengono cancellati dal display per cui si può continuare a scrivere nuovi caratteri anche se questi non vengono visualizzati.

Il comando display-on (da Dec 22 a Dec 25) riporta il display nella condizione di on e così controlla quanto si sta visualizzando e/o rende i caratteri del cursore lampeggianti. Il cursore è quella piccola barra evidenziata subito dopo l'ultimo carattere inserito. L'opzione di lampeggiamento (blink) rende il carattere del cursore lampeggiante o non lampeggiante.

Custom characters

L'LCD seriale ha la capacità di memorizzare fino ad 8 caratteri personalizzati. Questi caratteri vengono memorizzati nella RAM per cui devono essere ridefiniti se viene a mancare l'alimentazione. Si possono visualizzare i caratteri custom utilizzando i comandi da Dec 0 a Dec 7, come mostrato nella tabella dei comandi. I caratteri custom (personalizzati) vengono visualizzati nella posizione corrente del cursore.

Set dei comandi

Dal datasheet del dispositivo è possibile andare a individuare tutti i comandi del display LCD seriale utilizzato.

IL TSL230R

Caratteristiche tecniche

- Alta risoluzione di conversione intensità di luce in frequenza senza alcun componente esterno
- Sensibilità e fondo scala programmabile
- Comunicazione diretta con un microcontrollore
- Tolleranza di uscita di $\pm 5\%$ (TSL230B)
- Errore di non linearità di 0.2% a 100kHz
- Coefficiente di temperatura di 100 ppm/°C
- Single-Supply Operation Down to 2.7 V, With Power-Down Feature

Descrizione

I TSL230, TSL230 e TSL230B sono dispositivi programmabili che convertono luce in frequenza attraverso una certa combinazione di fotodiodi e di conversione corrente frequenza su un singolo circuito integrato CMOS.

Le uscite di tutti i dispositivi sono un treno di impulsi ad onda quadra (duty cycle del 50%) con frequenza direttamente proporzionale all'intensità luminosa.

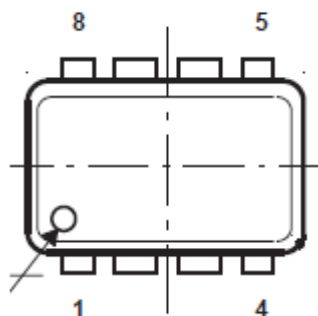
È possibile selezionare la sensibilità dei dispositivi in 3 range, fornendo due decadi di aggiustamento.

Il fondo scala in frequenza può essere scalato da uno a quattro valori di preset.

Tutti gli ingressi e tutte le uscite sono TTL compatibili permettendo così una comunicazione in entrambi i sensi con i microcontrollori.

Se viene abilita l'uscita OE negata si mette l'uscita del dispositivo in condizioni di alta impedenza for multiple-unit sharing of a microcontroller input line.

Pin 1	S0
Pin 2	S1
Pin 3	OE
Pin 4	GND
Pin 5	V _{CC}
Pin 6	OUT
Pin 7	S2
Pin 8	S3



Il TSL230 è un dispositivo che genera un segnale avente frequenze che cambia in proporzione alla quantità dell'incidenza di luce che arriva sul dispositivo stesso.

Il dispositivo è un chip a 8 pin che può fornire impulsi di clock con frequenza che rappresenta la specifica intensità.

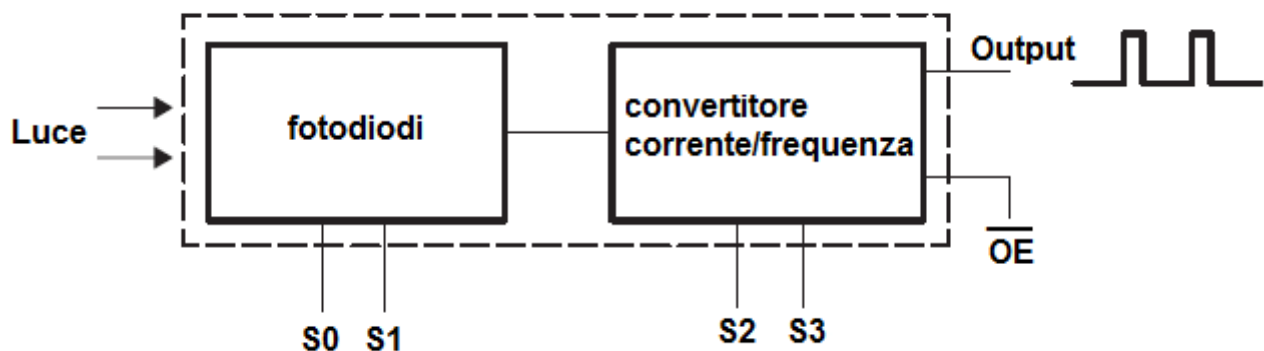
La luce entrante nel chip e colpisce una serie, un array di fotorilevatori, questo array produce una tensione che pilota un oscillatore controllato in tensione sugli 8 pin del chip.

Funzione dei pin

Pin	I/O	Descrizione
GND	4	Massa
\overline{OE}	3	I Abilitazione f0(attivo basso)
OUT	6	O Frequenza di uscita scalata (f0)
S0,S1	1,2	I input di selezione della sensibilità
S2,S3	7,8	I Ingressi selezione per scalatura di f0
VDD	5	Alimentazione

S1	S0	SENSIBILITA'	S3	S2	f0 scaling divisione per
L	L	Potenza minima	L	L	1
L	H	1x	L	H	2
H	L	10x	H	L	10
H	H	100x	H	H	100

Schemi a blocchi



Valori massimi nel range di temperatura con utilizzo in aria del TSL230

- VDD 6,5V
- range di tensione su ogni input -0.3V fino a VDD+0.3V
- range di temperatura di normale funzionamento -25°C a 70°C
- range di temperatura memorizzabile -25°C a 85°C
- massima temperatura sopportabile per 10s 260°C

Condizioni di funzionamento raccomandati

		MIN	NOM	MAX	UNIT
Tensione di alimentazione, VDD		2,7	5	6	V
Tensione di input, livello alto VIH	VDD=4.5 a 5.5V	2		VDD	V

Tensione di input, livello alto V _{IL}	V _{DD} =4.5 a 5.5V	0		0,8	V
Range di temperatura con funzionamento libero		-25		70	°C

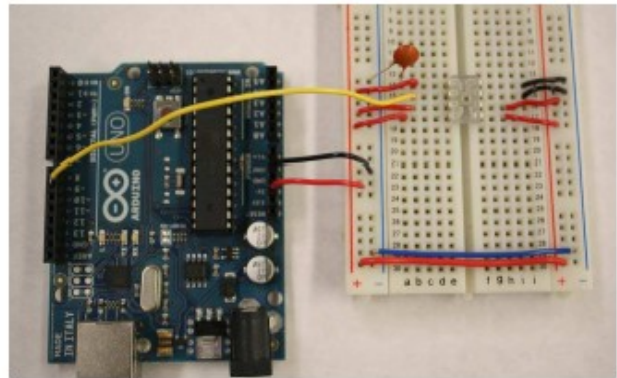
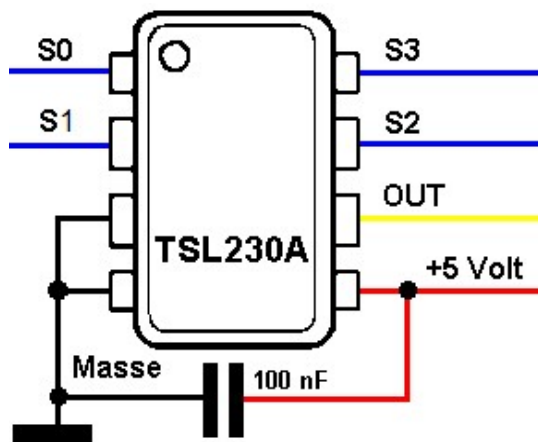
Caratteristiche elettriche principali

- **corrente assorbita di alimentazione:** 100µA in condizioni di minima potenza, tra 2 e 3mA in normali funzionamento
- **valore di fondoscala della frequenza:** 1.1 Mhz

Informazioni applicative

Per ottimizzare la risposta del dispositivo, la linea di alimentazione deve essere disaccoppiata per mezzo di un condensatore di valore 0.01 µF a 0.1µF con terminale a massa.

Codice di programmazione con Arduino



/ Questo programma permette di ottenere impulsi di larghezza proporzionale all'intensità della luce che colpisce il TSL230.*

*Tramite il comando pulseIn si può misurare la durata alta o bassa dell'impulso */*

```
int TSL = 9; // pin 9 di Arduino collegato con il pin6 del
// TSL che fornisce f0
unsigned long durata_impulso; // impulso proporzionale all'intensità
// della luce che colpisce il TSL230R
void setup(){
    Serial.begin (9600); // impostazione velocità di comunicazione seriale
    pinMode(TSL, INPUT); // definizione del pin collegato con f0
}
void loop(){
```

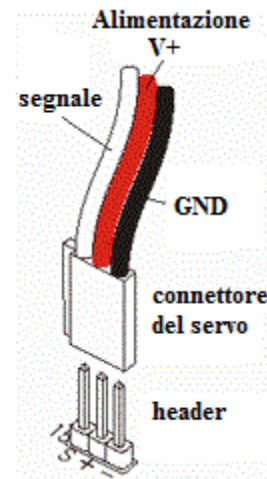
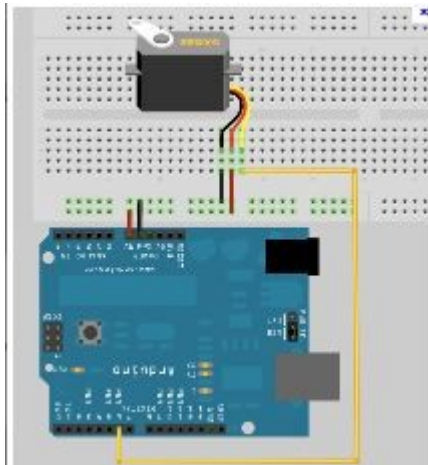
```
durata_impulso = pulseIn(TSL, HIGH);  
Serial.println (durata_impulso);    // scrive la durata alta dell'impulso con  
                                     // duty-cycle del 50%  
} //loop
```

Con questa configurazione il dispositivo lavora a massima sensibilità e senza divisione di frequenza.

I valori possono variare da valori bassi come 0 o alti come 3000 a seconda dell'intensità luminosa.

Ad alta intensità di luce corrispondono bassi valori mentre a bassa intensità luminosa corrispondono alti valori.

I SERVOMOTORI



I servomotori a rotazione non continua

Il servomotore è un motore che ha delle caratteristiche particolari. E' molto usato nel modellismo.

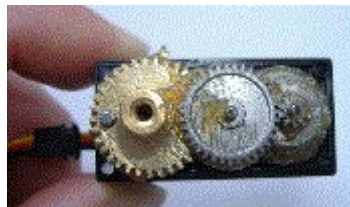
Il servo è costituito da un motore completo di riduzione meccanica, un sistema di feedback per la posizione dell'asse di uscita e tutta l'elettronica di controllo racchiusi in un unico contenitore.

Tramite un sistema di comando opportuno è possibile far ruotare l'asse di uscita del servo e posizionarlo in una precisa posizione voluta.

I diversi tipi di servomotori si differenziano in funzione della forza sviluppata e dalla qualità dei componenti meccanici utilizzati nella costruzione.

Dalla scatola del servo fuoriescono l'asse della riduzione meccanica al quale collegare il dispositivo da mettere in movimento, i due fili per l'alimentazione del servo e un filo per il controllo.

La parte meccanica del servo altro non è che una serie di ingranaggi che ha lo scopo di ridurre il numero dei giri del motore ad un valore utilizzabile dal dispositivo che dobbiamo comandare.



Notare bene che l'albero di uscita del servo può ruotare soltanto di **180°**. Per superare questo limite occorre modificare il servo

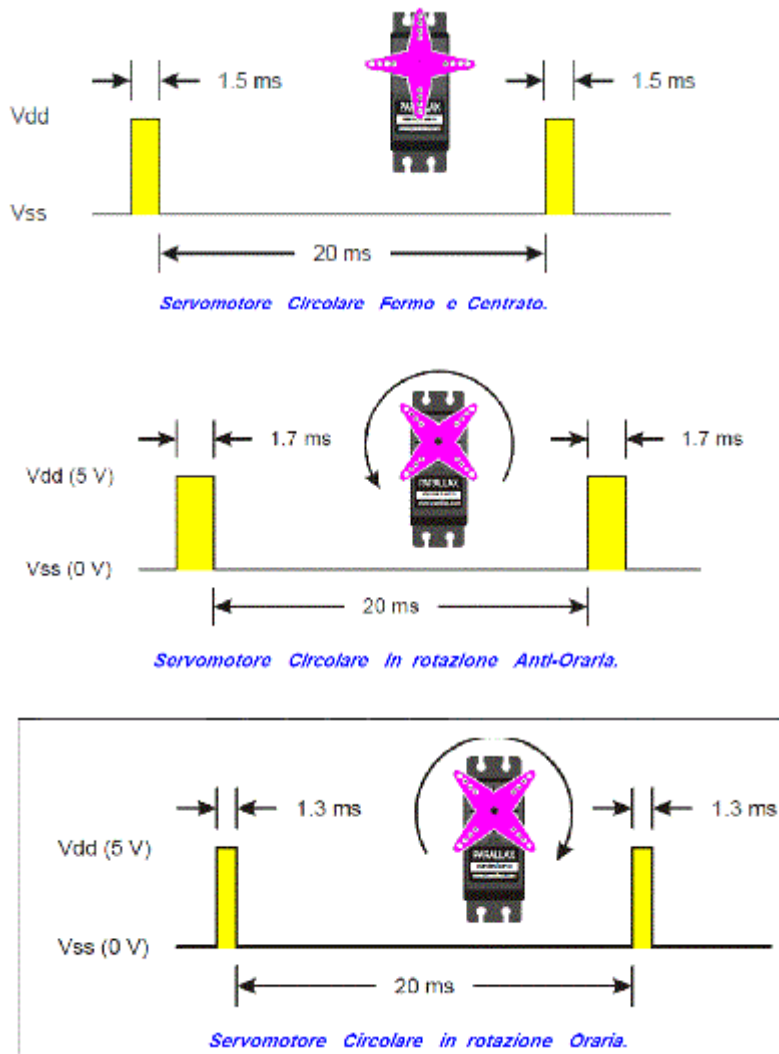
Comandare un servomotore

Per comandare un servomotore occorre inviare ad esso, tramite l'apposito filo di controllo,

una serie di impulsi positivi TTL (livello a 5 Volt).

La durata del singolo impulso determina la posizione dell'asse di uscita del servo mentre il tempo di pausa tra un impulso e l'altro può variare entro certi limiti senza provocare perdita di controllo del servo.

La durata dell'impulso può variare tra un minimo di **1 mS** ad un massimo di **2 mS**.



A seconda della durata di questo impulso, il servo farà ruotare il suo perno di uscita fino ad assumere una posizione ben precisa.

I valori sono indicativi e possono variare a seconda del modello.

Le durate minimo e massimo dell'impulso corrisponderanno alle posizioni estreme dell'asse del servo e, ovviamente, l'impulso di durata 1,5 mS porterà il servo ad assumere la posizione centrale.

Questo impulso va continuamente ripetuto, altrimenti il servo si metterà a riposo e non raggiungerà la posizione desiderata.

La durata della pausa tra un impulso e l'altro deve essere maggiore di 10 mS ma non deve superare i 40 mS.

La durata tipica della pausa tra gli impulsi è di 20 mS (50Hz).

Alla fine, per comandare un servo, si va a generare un treno di impulsi che ha una fortissima somiglianza con un segnale PWM ma non ha nulla a che vedere con esso.

Per comandare un servo serve una serie di impulsi in cui è fondamentale la durata del singolo impulso per determinare la posizione dell'asse ma non è critico il tempo di pausa tra un impulso e l'altro (seppur nei limiti impostati).

Il PWM è tutto un altro genere di segnale dove è importante anche il suo periodo e serve per altre cose, anche se simili, come comandare motori.

In molti siti di internet e in vari forum viene affermato che un servo si comanda con un segnale PWM. Ciò non è affatto vero!!! Anche se le forme d'onda del segnale PWM e del treno d'impulsi per comandare il servo possono risultare identici all'oscilloscopio, essi hanno una natura completamente diversa.

I servomotori a rotazione continua

Le caratteristiche di un servomotore a rotazione continua, tipo quelli in dotazione nei laboratori, sono:

- **Rotazione Continua Bidirezionale.**
- **Risposta Lineare tra 0 e 50 RPM.**
- **Possibile Pilotaggio tramite PWM** per generare rapidamente delle rampe.
- Semplice e facile gestione delle risorse.
- Il Servo a Rotazione Continua della Parallax è controllato da un impulso, di ampiezza variabile, in cui il senso e la velocità di rotazione dell'albero sono determinate dalla durata dell'impulso.

Per ottenere rotazioni regolari, il **Servo**, richiede una pausa di **20 ms** tra gli impulsi. Di seguito c'è un diagramma temporale, di esempio, che mantiene il **Servo Fermo e Centrato**.

Appena la durata dell'impulso decresce da **1,5 ms** il **Servo** ruoterà, in senso **Orario**, con una velocità gradualmente proporzionale alla riduzione così come è illustrato dalla figura seguente

Programmazione Arduino per servomotori a rotazione continua

/ Questo semplice programma fa girare un servomotore della parallax a rotazione continua.*

*Con il delay <2ms gira in senso orario, con valori superiori inverte il senso di rotazione. Conviene sempre provare con il servomotore in dotazione */*

```
int servo=3;
```

```
// pin di connessione del servomotore
```

```

void setup(){
    pinMode(servo,OUTPUT);           // va specificato il pin in quanto si sta
                                     // utilizzando in modalità digitale
}

void loop(){
    digitalWrite(servo,HIGH);        // mette alto lo stato del servo
    delay(1.5);                       // mantiene alto lo stato per 1.5ms
    digitalWrite(servo,LOW);         //abbassa lo stato del servo
    delay(10);                         //mantiene basso il valore per 10ms
}

// sperimentalmente variando il primo delay si ha solo l'inversione
// variando il secondo delay si modifica la velocità. Da 10ms a 40ms
// Più è basso il secondo delay più la velocità è maggiore

```

I MICROSWITCH

I microswitch in dotazione nei nostri laboratori sono dotati di una levetta in metallo. Dispongono di un contatto aperto (N.O) e di uno chiuso (NC). I terminali sono da saldare. Fissaggio mediante viti o collante.

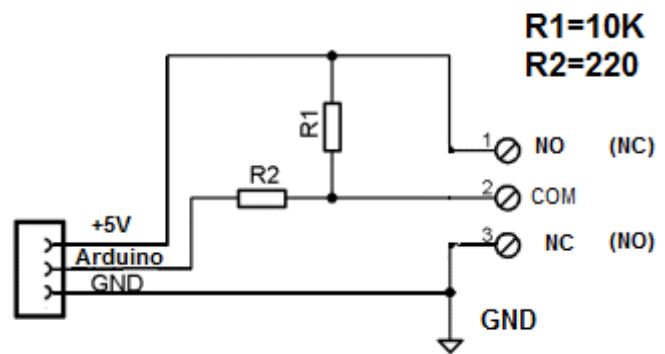
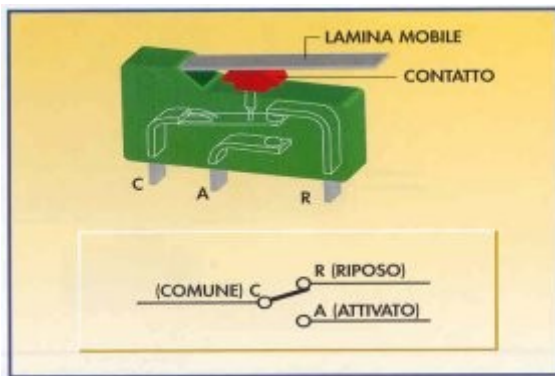
- **Resistenza di contatto:** 30 Mohm
- **Resistenza di isolamento:** 100 Mohm a 500 Vdc
- **Resistenza dielettrica:** 1000 Vac per 1 minuto
- **Vita media parti meccaniche:** 1.000.000 operazioni
- **Vita media circuito elettrico:** 200.000 operazioni
- **Dimensioni:** 10,2 x 19,8 x 6,4 mm
- **Corrente/Tensione:** 5 A / 125 Vac, 3A / 250 Vac



Utilizzo

I microswitches vengono utilizzati quali sensori di contatto: a seconda del sensore attivato è possibile provvedere di conseguenza all'azione più appropriata da eseguire.

Ad esempio se il microswitches è posto sul un robot che si muove, la sua attivazione potrebbe comandare al robot l'inversione di marcia evitando così un ostacolo.



I contatti C, A e R corrispondono direttamente al comune, al NO(normalmente aperto) e al NC (normalmente chiuso).

Il pin di Arduino andrà collegato al comune: sarà una scelta nostra poi definire quale sia la condizione da associare in caso di pressione della levetta.

Ad esempio si può considerare che quando la levetta sia premuta la linea I/O che va ad Arduino è messa a massa per cui la linea vede 0V; quando la levetta non è premuta (condizione di NO) la tensione ai pin I/O è invece 5V. Esempio del disegno su.

Può anche essere il viceversa: dipende da come si effettuano i collegamenti ai morsetti dei NO e NC.

I pin di Arduino verranno collegati direttamente a ciascun interruttore rilevando la tensione alla resistenza da 10kΩ di pull-up.

Codice di programmazione con Arduino

/ Questo programma accende un diodo led verde posto sul pin 10 durante la pressione della levetta del micorswitch.*

Il microswitch è collegato in questo modo:

*il comune va al pin 2 digitale di Arduino, il morsetto NO è collegato a 5 V,il morsetto NC è collegato a massa insieme a quella di Arduino */*

```

const int Switch = 2;           // pin che riceve il comando dal microswitch
const int verde = 10;          // pin dove è connesso il LED

int StatoSwitch = 0;           // variabile che legge lo stato della levetta dello switch

void setup() {
    pinMode(verde, OUTPUT);     // inizializzazione del pin che comanda il led
                                // verde quale OUTPUT
    pinMode(Switch, INPUT);     // e quello dello switch quale INPUT
}

```

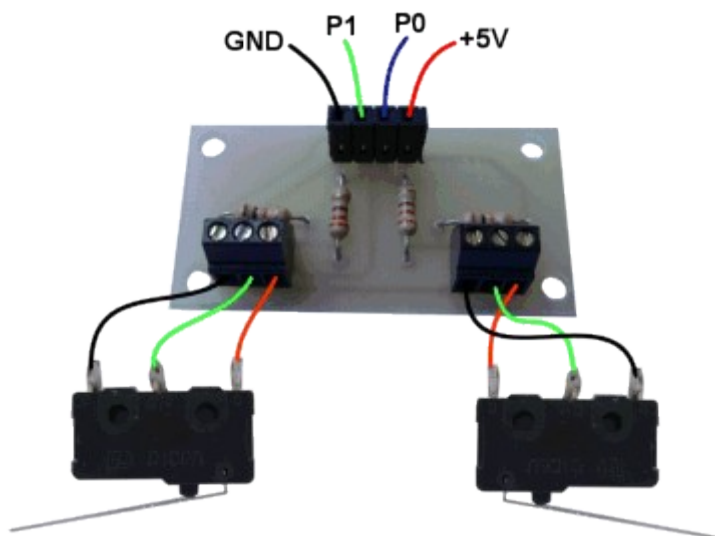
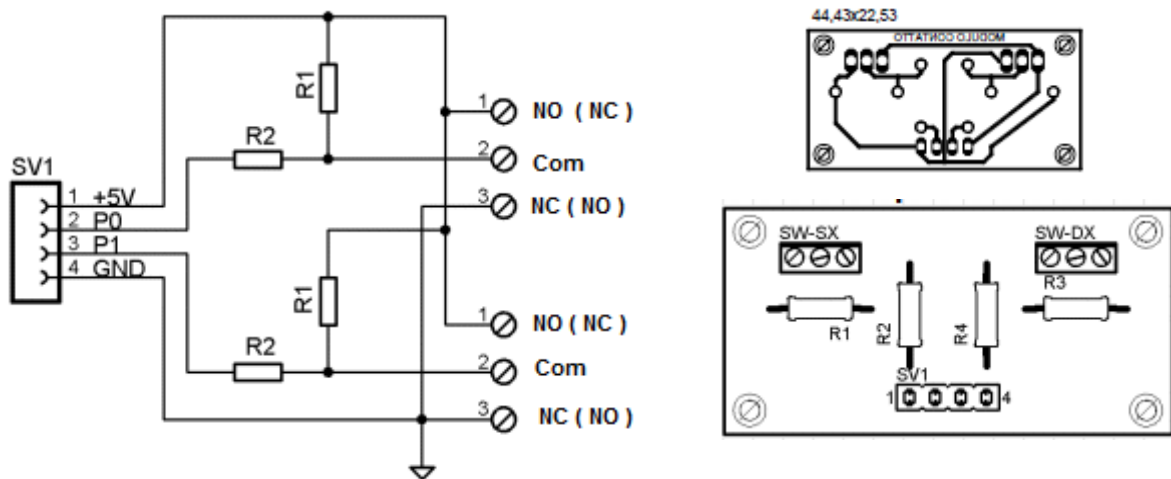
```

void loop( ){
  StatoSwitch = digitalRead(Switch);           // leggi lo stato dello switch
  if ( StatoSwitch == HIGH) {                 // verifica se lo stato è alto (il NO si trova a 5V)
    digitalWrite(verde, HIGH);               // accendi il led verde sul pin 10
  }
  else {
    digitalWrite(ledPin, LOW);               // spegni il diodo
  }
}

```

Modulo di comando di due microswitches

Schemi elettrici e di realizzazione



Problematiche

Questi piccoli pulsante sono dispositivi molto semplici: due parti di metallo separate da una molla. Quando premete il pulsante, i due contatti si toccano e l'elettricità può fluire. Questo suona facile, ma nella vita reale il collegamento non è così perfetto, specie quando il pulsante non è premuto completamente, e genera dei segnali spuri detti simbalzi (Bouncing)

Quando il pulsante produce questi segnali, Arduino vede una sequenza molto rapida di segnali acceso-spento. Per evitarlo sono state sviluppate molte tecniche, ma in questo semplice codice ho notato che normalmente basta aggiungere un ritardo di 10/50 millisecondi quando il codice rileva una transizione.

Un problema noto dell'utilizzo dei pulsanti, dei microswitch è che all'atto della pressione del tasto, il segnale in uscita non subisce una variazione istantanea dall'alto al basso e viceversa, si verifica una sorta di rimbalzo del segnale come si può vedere dalla foto seguente che mostra cosa accade quando viene premuta la levetta del microswitch.

Questo disturbo può risultare molto fastidioso nel caso in cui sia molto importante contare quante volte sia avvenuta la pressione di un tasto e gestire il comportamento del pulsante in fase di rilascio.

Per risolvere questo problema viene utilizzata la tecnica del debouncing detta anche di antirimbalzo.

Il debounce, o antirimbalzo, può esser utilizzato anche per eliminare disturbi sulle linee, dato che controlla lo stato del segnale per un certo periodo prima di fornire in uscita lo stato logico corretto.

Questa tecnica si può risolvere sia dal punto di vista hardware che dal punto di vista software.

Codice di antirimbalzo

/ Questo programma mantiene un led acceso variandone la luminosità in funzione del tempo di pressione di un pulsante, evitando problemi legati all'effetto di rimbalzo del segnale */*

```
#define Led 9           // Pin del Led
#define Pulsante 7     // Pin di Input del pulsante
int Valore = 0;       // Conserva lo stato del pin input
int Vecchio_Valore = 0; // Conserva il valore precedente di "Valore"
int Stato = 0;       // 0 = Led spento mentre 1 = Led acceso
int luminosita = 128; // Conserva il valore della luminosità
unsigned long startTime = 0; // quando è cominciata la pressione ?

void setup() {
    pinMode(Led, OUTPUT); // Indica ad Arduino che Led è un Output
```

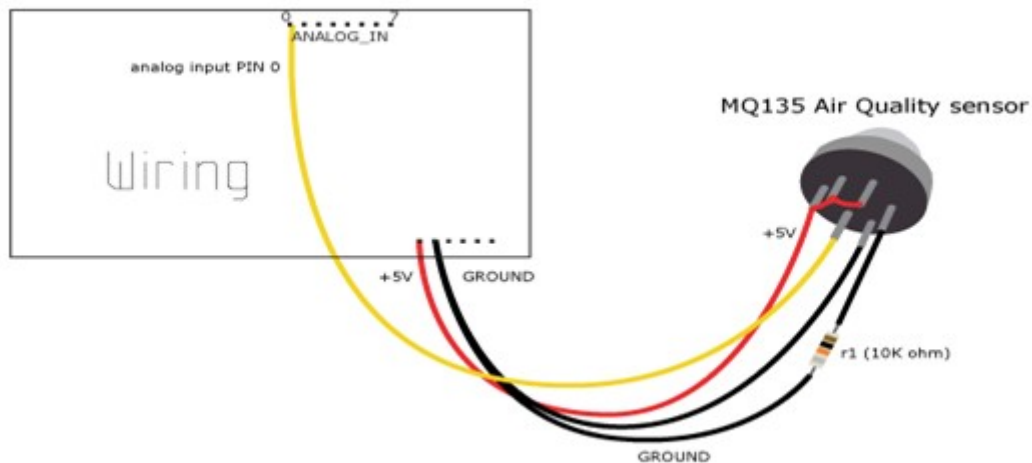
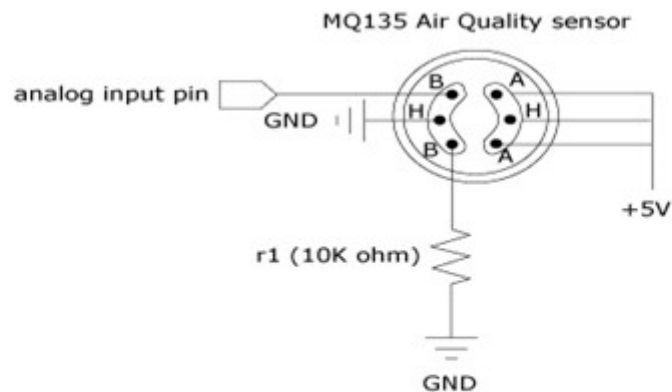
```

    pinMode(Pulsante, INPUT);    // e che Pulsante è un Input
}

void loop() {
    Valore = digitalRead(Pulsante);    // Legge il valore dell'input e lo conserva
    if ((Valore == HIGH) && (Vecchio_Valore == LOW)){ // Controlla se è
                                                //avvenuta una transizione
        Stato = 1 - Stato;    // Cambio lo stato del led
        startTime = millis();    // millis() restituisce quanti millisecondi sono
// passati da quando è stata resettata
// ( Questa riga ricorda quando è stato premuto il pulsante l'ultima volta )
        delay(10);
        if ((Valore == HIGH) && (Vecchio_Valore == HIGH)){ // Verifica che
                                                            // il pulsante sia ancora premuto
            if (Stato == 1 && (millis() - startTime) > 500){
// Se il pulsante è premuto per più di 500Ms
                luminosita++;    // Incremento di luminosità di 1
                delay(10);    // Ritardo per evitare aumenti
                                // troppo veloci
                if (luminosità > 255) {    // 255 è la luminosità massima
                    luminosità = 0;    // Se va oltre 255 torna a 0
                } //>255
            } //if >500ms
        } //if transizione
        Vecchio_Valore = Valore;    // Memorizza il valore precedente di Valore
        if (Stato == 1) {
            analogWrite(Led, luminosità);    // Accende il led al livello corrente di
                                                //luminosità
        } else {
            analogWrite(Led, 0);    // Spegne il Led
        }
    }
}

```

SENSORE RILEVAMENTO GPL



Questo sensore fornisce in uscita un segnale legato strettamente alla quantità di gas rilevato nell'aria.

```
int sensorValue;
```

```
void setup(){
```

```
    Serial.begin(9600); // sets the serial port to 9600
```

```
}
```

```
void loop() {
```

```
    sensorValue = analogRead(0); // read analog input pin 0
```

```
    Serial.print(sensorValue, DEC); // scrive il valore letto in formato
```



```
Serial.print(" ");  
delay(100);  
}  
  
// decimale  
// prints a space between the numbers  
// wait 100ms for next reading
```

FOTORESISTENZE

Le **fotoresistenze**: si tratta di resistenze il cui valore dipende dall'intensità e dal colore della luce che le colpisce; in genere sono dei sottili film di solfato di cadmio su un supporto rigido, chiusi in involucri protettivi trasparenti. Data la struttura fisica, si comprende come questi non siano quasi mai elementi di potenze elevate; valori caratteristici della massima potenza dissipabile sono sui 50mW per le più piccole, circa 1 W per le più grosse.

Le fotoresistenze sono caratterizzate dalla curva di sensibilità, cioè dal colore al quale sono maggiormente sensibili e dai valori della resistenza al buio e alla luce forte, dette valore di buio e valore di luce; si ha indicativamente:

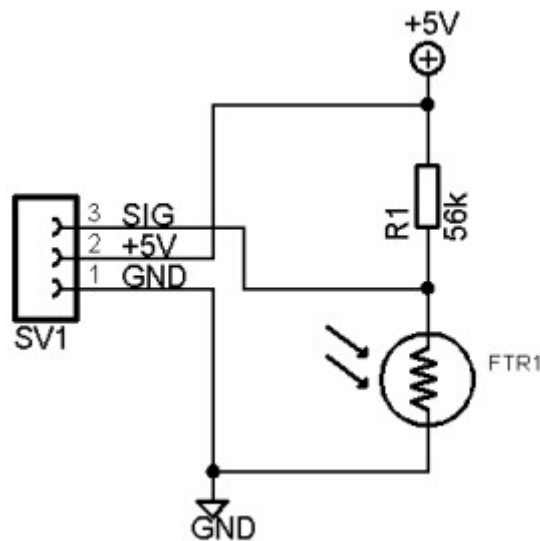
valore di buio: qualche Mohm

valore di luce: intorno al Kohm

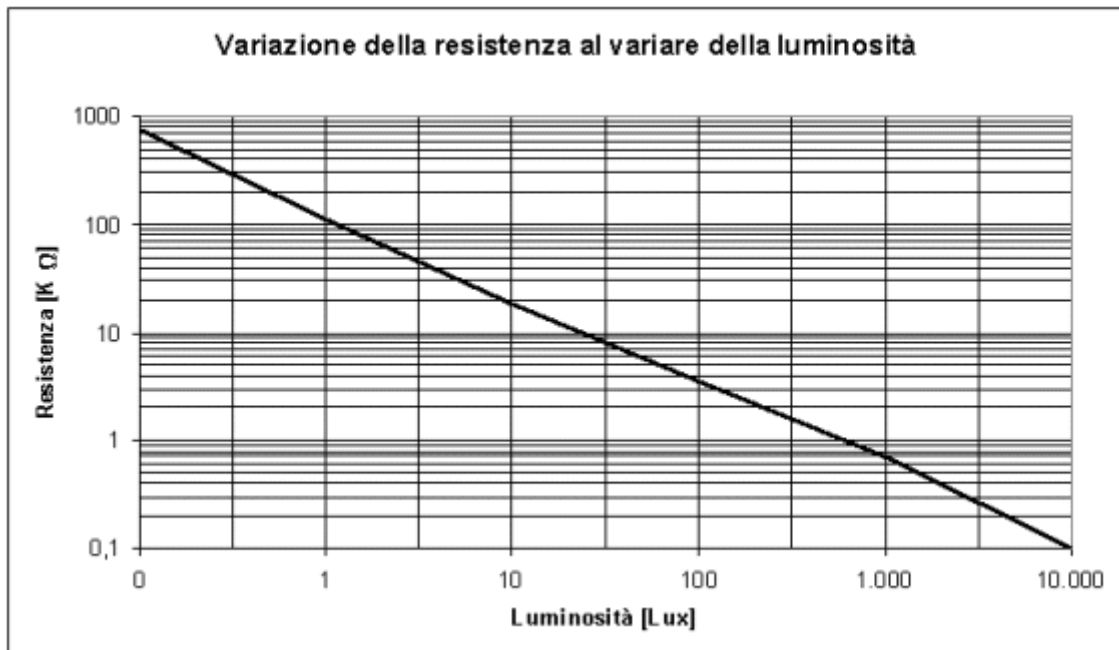
Il campo di variabilità è quindi molto elevato, dell'ordine di 1000.

La loro caratteristica di illuminazione è definita dall'equazione

$$R = A I^\alpha$$



Quindi, date la resistenza di buio e di luce, si può tracciare una caratteristica rettilinea (in scala logaritmica) che approssima abbastanza bene quella reale (vedi grafico). Bisogna però tener presente che questi elementi sono "lenti" (variazione di circa 200 Kohm/s) cioè se la luce varia rapidamente non è detto che il valore della resistenza la segua con la stessa legge.



Il programma sotto riportato prevede l'utilizzo oltre che del sensore di luce connesso alla porta AD0 la connessione di un display LCD seriale, in questo caso quello costruito sulla base del progetto della SparkFun.

Il display è connesso alla porta digitale 3.

Il programma in base al valore rilevato dalla fotocellula provvede anche all'accensione del led connesso alla porta digitale 13, led presente sulla scheda.



/*

Letture_Fotocellula_lcd.pde

Il programma monitorizza una fotocellula collegata alla porta AD0 e riporta il valore sul LCD, lo stesso valore è anche inviato al PC dove può essere visualizzato sul Serial monitor. In base al valore letto viene anche acceso il led collegato alla porta digitale pin 13.

Vengono utilizzati i seguenti pin:

Pin +5V

```

Pin GND
Pin Digital 3      Trasmissione dati seriali
Pin Analogico 0    lettura fotocellula
*/

// Inclusione della libreria SoftwareSerial in modo da poter utilizzare le sue
// funzioni:

#include <SoftwareSerial.h>
#define rxPin 4
#define txPin 3

int fotoresistenzaPin = 0 ; // il centro del partitore lo colleghiamo al pin 0
int fotoresistenzaReading; // il valore che conterrà la lettura
int valore =90;

// Imposta una nuova porta seriale
SoftwareSerial mySerial = SoftwareSerial(rxPin, txPin);
byte pinState = 0;

void setup() {
  // definisce i pin per tx, rx:
  pinMode(rxPin, INPUT);
  pinMode(txPin, OUTPUT);
  pinMode(13, OUTPUT);

  // imposta la velocità della porta per LCD
  mySerial.begin(9600);
  // imposta la velocità della porta per PC
  Serial.begin(9600);
}
// Resetta il display, annullando qualsiasi scorrimento e la rimozione di tutto il
// testo
void clearLCD(){
  mySerial.print(0xFE, BYTE); //command flag
  mySerial.print(0x01, BYTE); //Comando clear //
  delay(50);
}

```

```

// Avvia il cursore all'inizio della prima linea
void firstrow(){
  mySerial.print(0xFE, BYTE);
  mySerial.print(128, BYTE);
  delay(2);
}

// Avvia il cursore all'inizio della seconda linea
void secndtrow(){
  mySerial.print(0xFE, BYTE);
  mySerial.print(128+64, BYTE);
  delay(2);
}

// Routine lettura e stampa valore su PC o LCD
void loop (){
  fotoresistenzaReading = analogRead(fotoresistenzaPin);
  // lettura grezza dall'adc

  // Accensione led
  if (fotoresistenzaReading > valore) {
    digitalWrite (13, HIGH);
  }
  else if (fotoresistenzaReading < valore) {
    digitalWrite (13, LOW);
  }

  //Invia dati su porta seriale PC
  Serial.print("Lettura analogica: = ");
  Serial.println(fotoresistenzaReading); //stampiamo il valore

  // Stampa su LCD
  clearLCD();
  delay (50);
  firstrow();
  mySerial.print( "Lettura LDR" );
  secndtrow();
  mySerial.print("Valore: = ");
  mySerial.print(fotoresistenzaReading); //stampiamo il valore

```

```
delay(1000); // Pausa tra le letture
```

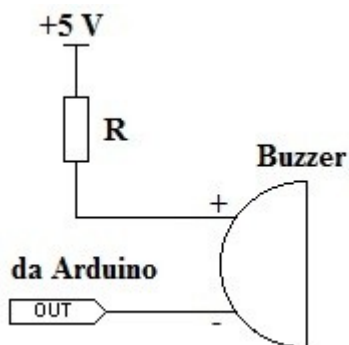
IL BUZZER

Il buzzer è un trasduttore acustico che trasforma un segnale elettrico, normalmente una corrente, in un suono.

Il segnale di ingresso, inviato da Arduino, è un'onda quadra avente una certa frequenza (secondo la tecnica del PWM).

Il suono emesso dipende dalla frequenza di questo segnale di ingresso.

Il circuito di controllo del dispositivo è il seguente:



Il buzzer utilizzato è il MT12D02 che presenta le seguenti caratteristiche:

Codice	MT12D02
Tensione di alimentazione (v)	5
Valori limite di alimentazione (v)	3~8
Consumo di corrente (mA)	40 max
Resistenza della bobina (Ω)	47 \pm 52
Impedenza della bobina (Ω)	80
Livello di pressione sonora a 10cm	85dB min
Frequenza di risonanza	2400Hz
Temperatura operativa.($^{\circ}$ C)	-25~ + 70
Peso	2.0 g

