Versione N° 1.0

LaurTec

RGB LED Controller

Controller per barre LED dinamiche

Autore : *Matteo Garia*

ID: UP0002-IT

INFORMATIVA

Come prescritto dall'art. 1, comma 1, della legge 21 maggio 2004 n.128, l'autore avvisa di aver assolto, per la seguente opera dell'ingegno, a tutti gli obblighi della legge 22 Aprile del 1941 n. 633, sulla tutela del diritto d'autore.

AVVERTENZE

I progetti presentati non hanno la certificazione CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.

L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.

Si fa inoltre presente che quanto riportato viene fornito così com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Indice

Introduzione

La scheda per LED RGB qui presentata consente il facile controllo di LED o barre RGB in diverse modalità. E' possibile infatti utilizzare un filocomando connesso al connettore apposito, oppure comandare i LED mediante PC. L'interfacciamento con quest'ultimo avviene grazie al convertitore USB-seriale presente sul PCB.
La scheda è in grado di gestire 3 barre indipendenti, fino ad un massimo di 100mA per ogni colore.
Il progetto nasce con l'idea di modificare un kit di illuminazione commerciale, venduto da IKEA con il nome DIODER. Di tale kit vengono utilizzate le barre a led, il comando a filo e l'alimentatore.
Il progetto è comunque adatto a pilotare qualsiasi led entro i limiti di corrente ammessi, e può esser personalizzato per le proprie esigenze.

Specifiche Tecniche

Alimentazione : 12VDC
Assorbimento : 1 A max.
Assorbimento da spento : circa 100mW
Corrente massima per uscita : 100mA (a colore) – 300mA (totale)
Dimensioni : 40 x 50 mm
Product Number: UP0002
Versione PCB : 1.0

Applicazioni

Il progetto in questione può avere le seguenti applicazioni:

- Controllo manuale di LED o barre RGB
- Controllo mediante PC (grazie all'IC adattatore USB-seriale)
- Modalità dinamica con apposito software su PC per creare effetti ambientali per l'utilizzo multimediale (vedere il sistema *Philips AmbiLight*)
- Possibilità di creare facilmente applicazioni su PC per la gestione di effetti di luce

Analisi del progetto

In Figura 1 è riportato lo schema elettrico del progetto, realizzato per mezzo di un PIC16F88.

- È possibile notare come l'alimentazione sia ottenuta mediante il classico regolatore LM7805, mentre le uscite di tipo Open Collector per i Led sono alimentate direttamente dalla tensione in ingresso, è pertanto indispensabile porre adeguati resistori di limitazioni in serie ai led stessi. Molte barre led sono già dotate internamente di tali resistori, per consentire il funzionamento a 12V DC.
  La scelta di non installare i resistori è dovuta alla volontà di rendere flessibili le uscite potendo collegare diversi tipi di LED.

- E' presente un integrato FT232RL (di cui rimando al datasheet per ulteriori informazioni) per la comunicazione al PC. La scelta di questa soluzione è stata dettata dalla scomparsa della porta seriale dalla maggioranza dei PC in commercio ai giorni nostri.
  Il dispositivo, una volta connesso al PC, viene individuato come una porta seriale, consentendone poi un semplice controllo, mediante un set di comandi esposto più avanti.

- Il connettore “EXT_KEYS” è pensato per l'aggiunta di comandi esterni, come un semplice filocomando. Oltre ad alimentazione e massa, è presente un pin I/O digitale e due analogici. I pin analogici possono esser usati per interfacciarsi in modo semplice con un potenziometro, oppure anche con dei pulsanti connessi su un partitore resistivo.
  È utile notare il particolare pin-out del connettore, pensato per evitare guasti nel caso venga inserito al contrario. Viene presentato uno schema di filocomando compatibile con il firmware attuale (versione 1.1)

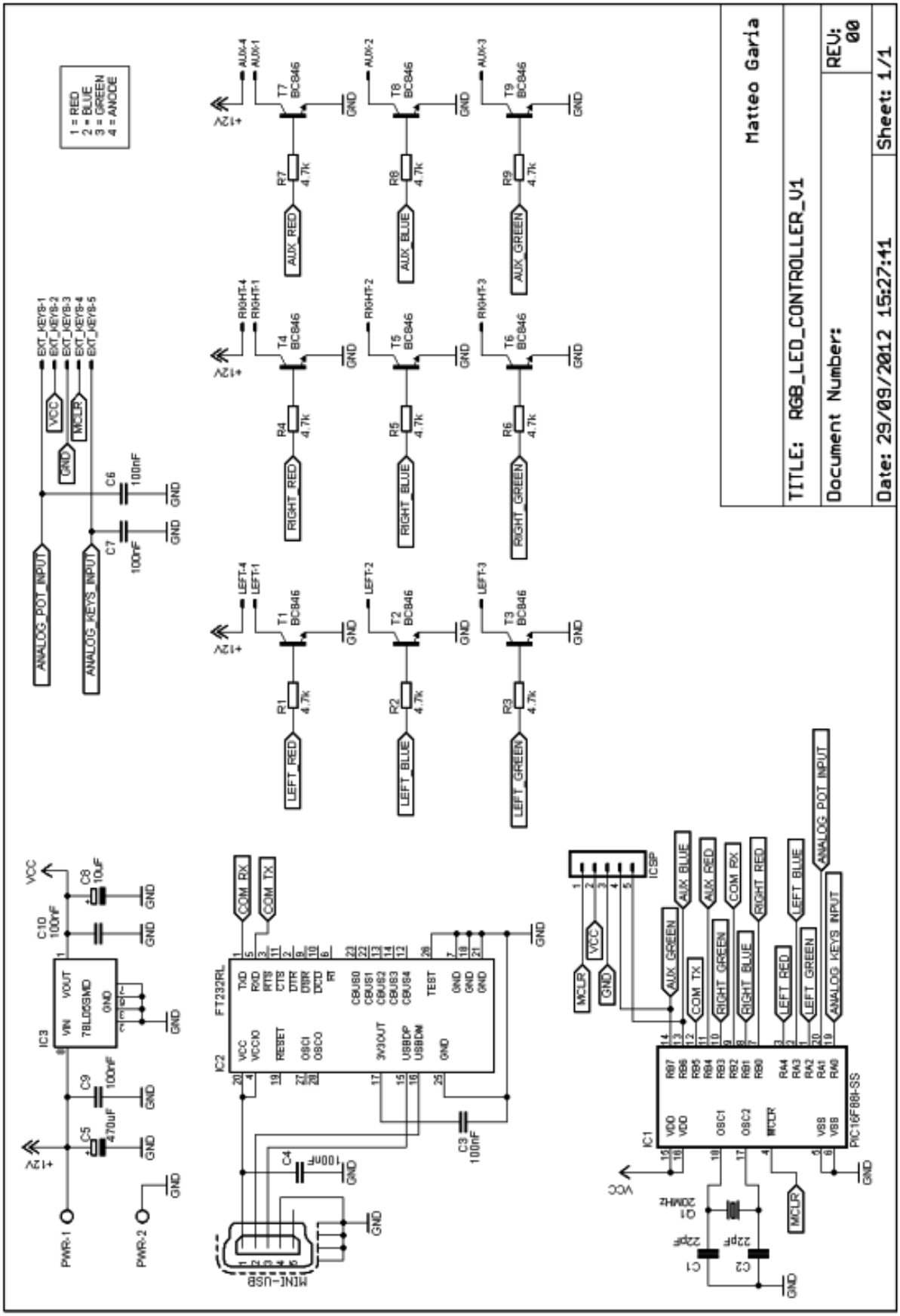

Figura 1: *Schema elettrico dell'RGB LED Controller.*

Lista Componenti

Resistori

R1 = 4.7kΩ %5 1/4W 0603
R2 = 4.7kΩ %5 1/4W 0603
R3 = 4.7kΩ %5 1/4W 0603
R4 = 4.7kΩ %5 1/4W 0603
R5 = 4.7kΩ %5 1/4W 0603
R6 = 4.7kΩ %5 1/4W 0603
R7 = 4.7kΩ %5 1/4W 0603
R8 = 4.7kΩ %5 1/4W 0603
R9 = 4.7kΩ %5 1/4W 0603

Condensatori

C1 = 22pF 0603
C2 = 22pF 0603
C3 = 100nF ceramico 0805
C4 = 100nF ceramico 0805
C5 = 470uF 25V
C6 = 100nF ceramico 0805
C7 = 100nF ceramico 0805
C8 = 10uF 16V
C9 = 100nF ceramico 0805
C10 = 100nF ceramico 0805

Transistor

T1: BC846 NPN
T2: BC846 NPN
T3: BC846 NPN
T4: BC846 NPN
T5: BC846 NPN
T6: BC846 NPN
T7: BC846 NPN
T8: BC846 NPN
T9: BC846 NPN

Circuiti Integrati

IC1 = PIC16F88I-SS
IC2 = FT232RL
IC2 = LM78L05 SMD

Quarzi

Q1 = 20MHz SMD

Connettori

LEFT = molex 4 pin passo 2.54 orizzontale
RIGHT = molex 4 pin passo 2.54 orizzontale
AUX = molex 4 pin passo 2.54 orizzontale
EXT KEYS = molex 5 pin passo 2.54 orizzontale
USB = connettore mini-usb
PWR = morsettiera 2 poli passo 5.08
ICSP = header maschio 5 pin

Connettori del controller

CONNETTORI LED (LEFT-RIGHT-AUX)				
Pin	**Nome**	**Direzione**	**Tipo**	**Funzione**
1	**RED**	**Output**	**Open Collector**	Uscita catodo LED rosso
2	**GREEN**	**Output**	**Open Collector**	Uscita catodo LED verde
3	**BLUE**	**Output**	**Open Collector**	Uscita catodo LED blu
4	**+12V**	**Alimentaz.**	**+12V**	Alimentazione a 12V per l'anodo dei LED

Tabella 1: *Tabella riassuntiva del pinout del connettore per l'interfacciamento ai LED. Le funzioni dei singoli pin si riferiscono al firmware v.1.1*

CONNETTORE EXT KEYS				
Pin	**Nome**	**Direzione**	**Tipo**	**Funzione**
1	**POT**	**Input**	**Ingresso analogico 0-5V**	Ingresso potenziometro di controllo
2	**VCC**	**Alimentaz.**	**+5V**	Alimentazione per il filocomando
3	**GND**	**Alimentaz.**	**Massa**	Massa
4	**MCLR**	**I/O**	**Ingresso/Uscita digitale**	Attualmente non usato, disponibile
5	**KEYS**	**Input**	**Ingresso analogico 0-5V**	Ingresso per i pulsanti

Tabella 2: *Tabella riassuntiva del pinout del connettore per l'interfacciamento al filocomando. Le funzioni dei singoli pin si riferiscono al firmware v.1.1*

Realizzazione su circuito stampato

Per contenere gli ingombri si è scelto di realizzare il controller con componentistica SMD. Tale realizzazione richiede maggiore attenzione rispetto all'uso di componentistica tradizionale, ma il risultato finale è sicuramente soddisfacente.

La realizzazione del PCB può esser eseguita senza particolari difficoltà con la fotoincisione casalinga, avendo cura di:

- Stampare con una stampante laser alla massima qualità, disattivando l'economia del toner.
- Curare perfettamente l'allineamento dei master top-bottom
- Stampare in modo che il lato stampato sia a contatto con il rame. L'irrisorio spessore del foglio potrebbe sfocare le piste trasferite sul rame, con risultati deleteri trattandosi di piste molto sottili.

Nota:

I master forniti sono già orientati correttamente, è sufficiente stamparli così come sono (no mirror).

Per la saldatura non occorrono strumenti particolari che non siano tipicamente presenti nel laboratorio, in particolare, è essenziale dotarsi di un saldatore con punta a spillo, trecciola dissaldante e pinzette. Utilissima anche una lente ad elevato ingrandimento, in mancanza di essa può esser utile una fotocamera in grado di scattare foto macro (Ingrandendola sul display si possono notare i dettagli delle saldature)

I componenti passivi ed i transistor si saldano prestagnando una piazzola, posizionando il componente e fissandolo con il saldatore alla piazzola stagnata, successivamente si potrà saldare anche l'altra/le

altre e ripassare la prima.

Gli integrati a passo 0.5, per quanto possano sembrare difficoltosi, si saldano con facilità con la seguente tecnica:

- Prestagnare tutte le piazzole, rimuovendo ogni eccesso con la trecciola. Le piazzole devono essere argentee, ma non avere grumi di stagno sopra
- Posizionare l'integrato, fissare con il saldatore qualche pin (anche cortocircuitandoli) in un angolo
- Ora, sul lato opposto, stagnare tutti insieme i pin, ripetere l'operazione dal primo lato.
- Rimuovere con la trecciola ogni eccesso di stagno: i pin si separeranno ed ognuno sarà ben attaccato alla sua piazzola.

Nota:
Fare molta attenzione alle piccole gocce di stagno che potrebbero rimanere sotto i pin, visibili con la lente.

Nel montaggio bisogna porre attenzione a non ostacolarsi la saldatura dei componenti, procedendo con ordine dal centro della scheda:

- Prima i passaggi delle piste top-botton. Le piazzole di tali passaggi vanno forate con punta da 0.6mm, poi di può scegliere se usare gli appositi rivetti (consigliati) o se saldare dei sottili fili di rame
- Successivamente gli IC: il PIC16F88I-SS e l'FT232RL
- A seguire tutti i componenti andando via via ad allontanarsi dal centro
- Per ultimi i componenti through-hole.

In Figura 2 è possibile notare sulla sinistra i connettori di alimentazione, tastiera e USB; sulla destra sono presenti i tre connettori per i LED. Il connettore ICSP è situato sopra al PIC. Nella parte inferiore del PCB è presente il quarzo con i relativi condensatori ed i transistor di uscita.

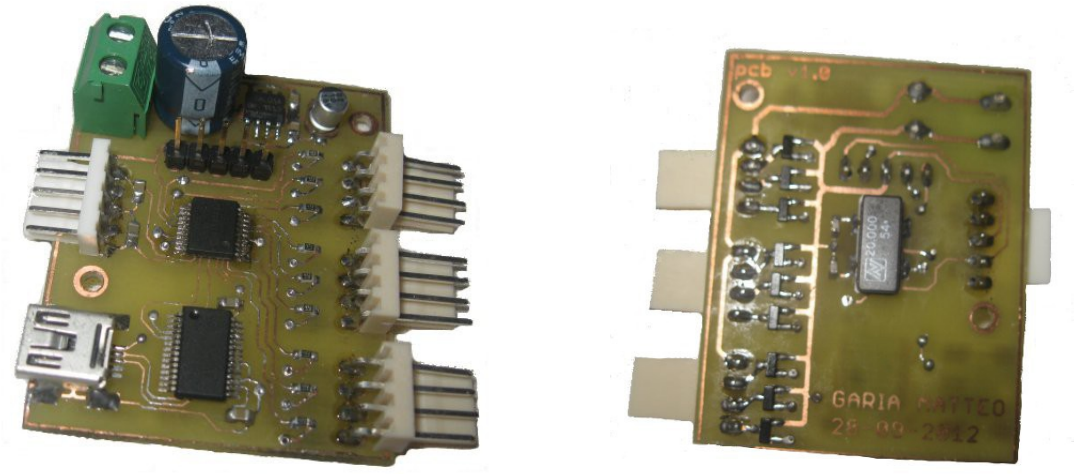

Figura 2: *Fotografie del controller assemblato.*

Filocomando

In Figura 3 è visibile lo schema elettrico del filocomando. Non esiste PCB poiché è stato realizzato modificando il comando originale del KIT IKEA utilizzato nel progetto, è comunque molto semplice e realizzabile su PCB millefori.

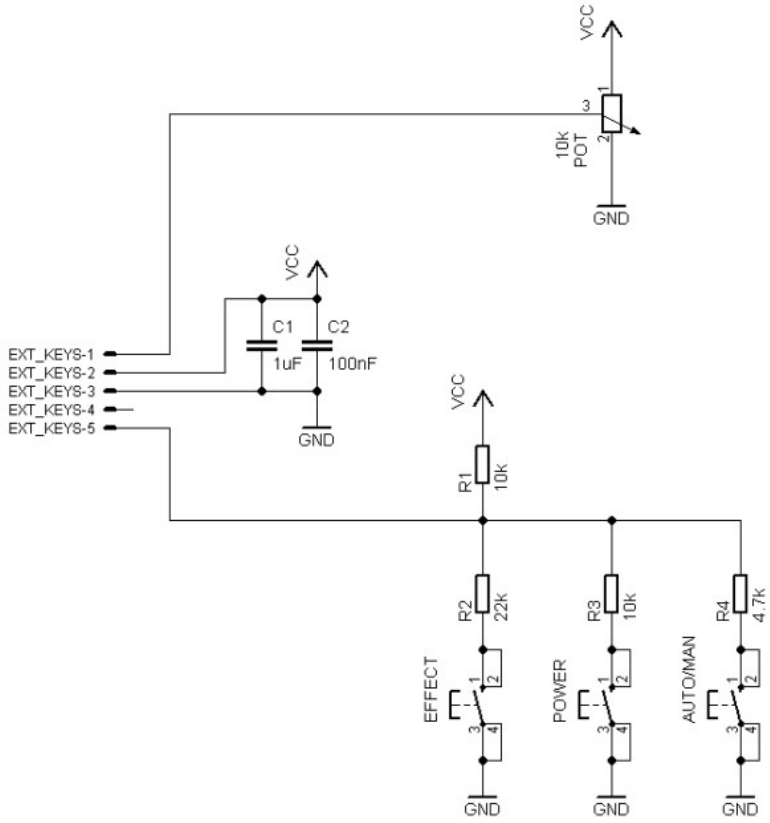

Figura 3: *Schema elettrico del filocomando. Rispettare i valori di resistenza indicati.*

I pulsanti svolgono le seguenti funzioni:

- **Effect**: cambia il gioco di luce eseguito da controller. Sono presenti due effetti: arcobaleno e soft.
  L'effetto arcobaleno esegue la ruota dei colori RGB o permette di impostare con il potenziometro un qualsiasi colore.
  L'effetto soft esegue il lampeggio “respiro” cambiando colore ad ogni lampeggio.
- **Power**: accende e spegne il controller.
- **Auto/Man**: cambia la modalità degli effetti. In automatico il potenziometro imposta la velocità dell'effetto, in manuale l'effetto è statico ed il potenziometro imposta il colore.

Prima messa in funzione dopo il montaggio

Una volta ultimato il montaggio, procedere con i seguenti step di verifica, prima di rendere operativo il progetto. **Non alimentare** finché non richiesto.

- Collegare l'USB ad un PC, verificare (previa installazione del driver FTDI) che il controller sia correttamente rilevato come porta seriale.
- Alimentare (attenzione alla polarità) il circuito a 9V. Verificare che la tensione VCC in uscita dal regolatore sia 5V (±0.25V).
- A questo punto alimentare il circuito a tensione nominale (12V), collegare il programmatore e scaricare il firmware sul PIC.
- Una volta programmato, collegare il filocomando ed i LED e verificare il funzionamento del sistema. Provare anche i comandi da PC con un Software come "*RS232 Terminal*"[1].

Nota:

Non alimentare mai per nessuna ragione il dispositivo dal programmatore, si danneggerà irrimediabilmente il regolatore di tensione.

[1] E' possibile scaricare il programma "*RS232 Terminal"* dal sito www.LaurTec.it .

Firmware

Il firmware è sviluppato in ambiente MPLAB-X con compilatore Hitech C per PIC16. Si compone di un singolo file .C accompagnato dal relativo Header file.
Vediamo nel dettaglio l'implementazione delle funzioni principali. Per la comprensione dell'intero progetto, si rimanda alla lettura del codice con relativi commenti.

Struttura principale del programma

Il programma è temporizzato utilizzando il Timer0 del PIC. Tale Timer è impostato per fornire un interrupt ogni 0.05ms. Il tempo così ridotto è necessario per consentire di generare i segnali PWM con una frequenza sufficientemente alta da non produrre sfarfallio sui led; essendo necessari 9 segnali PWM indipendenti non è stato possibile usare il modulo CCP interno al PIC.

Tale condizione impone però che il resto del Software sia ottimizzato per esser eseguito entro tale tempo, viceversa si andrebbe ad influenzare la funzione PWM. Per questa ragione le parti del codice che non hanno priorità alta vengono eseguite saltando alcuni cicli ed eseguendone solo una per ciclo.
Gli effetti di luce sono creati mediante appositi calcoli per la generazione dei PWM, ed usano come input un valore a 10bit.

Entriamo nel dettaglio analizzando la funzione di **interrupt**
Come ogni buon programmatore sa, tale funzione deve essere quanto più breve possibile.

```
12  void interrupt isr(){
13      if (TMR0IF)
14      {
15  //Ricarico TMR0 con il preload stabilito
16          TMR0=TMR0_PRELOAD;
17  //Imposto il flag di avvenuto interrupt
18          main_int = 1;
19  //Azzero l'interrupt flag di TMR0
20          TMR0IF=0;
21      }
22
23      if(RCIF)
24      {
25  //Leggo il buffer seriale
26          rx_chars[rx_counter] = RCREG;
27  //Incremento il contatore caratteri
28          rx_counter++;
29  //NOTA: il flag di interrupt si azzera in automatico dopo la lettura
30      }
31
32  }
```

Gli interrupt gestiti sono due: Timer0 e ricezione di caratteri dall'UART.

- L'interrupt di Timer0 viene utilizzato per settare un flag, usato poi per temporizzare il ciclo main del programma.
- Quando invece si verifica l'interrupt di ricezione, il PIC salva il carattere ricevuto nell'array rx_chars, e incrementa il contatore per esser pronto a ricevere il carattere successivo. Notare come il flag di Timer0 viene azzerato a mano, viceversa, il flag RCIF si azzera automaticamente.

L'intera applicazione è contenuta in un ciclo infinito, prima di essa vi è la configurazione dei registri del PIC. La prima parte del loop è costituita dalla gestione dei comandi:

```
46  //Verifico se la comunicazione seriale si è interrotta (timeout)
47              if(rx_counter > 0){com_timeout++;}
48              if(com_timeout == TIMEOUT){
49                  rx_counter = 0;
50                  com_timeout = 0;
51                  ERROR;
52              }
53  //Passo a decodificare i comandi ricevuti (se presenti)
54              if(rx_counter == 4)
55              {
56                  com_timeout = 0;
57                  PcCommands();
58              }
```

- Un contatore (com_timeout) viene incrementato quando si riceve un carattere, esso continua ad esser incrementato a meno che il comando sia ricevuto completamente (4 byte). Se tale contatore raggiunge il valore di TIMEOUT il PIC invia un messaggio di errore ed azzera il buffer di ricezione (azzerandone l'indice rx_counter).
- Se invece il comando viene ricevuto completamente il timeout viene azzerato e viene chiamata la funzione di gestione dei comandi, costituita da un blocco switch-case che interpreta i comandi.

La gestione della comunicazione viene eseguita ad ogni ciclo. Come già spiegato, per ragioni di tempo, alcune parti del codice vengono eseguite a cicli alterni: l'acquisizione dei valori analogici, la gestione dei pulsanti, la gestione del potenziometro e il calcolo dei valori RGB per gli effetti.
Per gestire queste funzioni è usato un contatore dedicato.

Il cuore del codice è in realtà la parte più semplice, ovvero la *generazione del segnale PWM.* Tale segnale è generato ad 8 bit, confrontando il valore voluto con un conteggio ad incremento.
L'uscita è alta finché il conteggio non supera il valore impostato, dopodiché diventa bassa. Quando il contatore torna a zero, il ciclo si ripete.

```
129  void SoftwarePwm(unsigned char pwm0, unsigned char pwm1, unsigned char pwm2,
130                   unsigned char pwm3, unsigned char pwm4, unsigned char pwm5,
131                   unsigned char pwm6, unsigned char pwm7, unsigned char pwm8){
132      if (pwm0 > pwm_counter || pwm0 == 255)
133      {PWM_OUT_0 = HIGH;}
134      else
135      {PWM_OUT_0 = LOW;}
```

Ovviamente sono presenti 9 “if” per altrettante uscite.

I comandi locali, sia da potenziometro che da pulsanti, sono esclusivamente gestiti con il modulo ADC, pertanto è necessario leggere 2 canali. Quando si usano più canali dell'ADC è fondamentale ricordare che, dopo aver selezionato il canale corretto, bisogna lasciare un piccolo ritardo per avere una lettura attendibile.

In questo caso particolare l'acquisizione è suddivisa in 4 step, eseguiti in successione ad ogni chiamata della funzione:

```
239 void AdConversion(){
240 //Primo step della conversione: leggo il potenziometro, confermo la lettura
241 //e passo al canale dei pulsanti
242     if (AdcOperation == 0){
243         potValue = ADC_RESULT;
244         potIsValid = TRUE;
245         SELECT_ADC_CHANNEL(BTN_INPUT);
246     }
247 //Secondo step: avvio conversione
248     if (AdcOperation == 1){
249         TMR0IE = 0;
250         GO = 1;
251         TMR0IE = 1;
252         potIsValid = FALSE;
253     }
254 //Terzo step: leggo i pulsanti, confermo la lettura e passo al canale del pot.
255     if (AdcOperation == 2){
256         btnValue = ADC_RESULT;
257         btnIsValid = TRUE;
258         SELECT_ADC_CHANNEL(POT_INPUT);
259     }
260 //Quarto step: avvio conversione
261     if (AdcOperation == 3){
262         TMR0IE = 0;
263         GO = 1;
264         TMR0IE = 1;
265         btnIsValid = FALSE;
266     }
267 //Incremento il contatore dello step di conversione
268     AdcOperation++;
269     if (AdcOperation == 4){AdcOperation = 0;}
270 }
```

I pulsanti restituiscono un valore analogico, tramite il quale si può dedurre quale pulsante è premuto. I valori teorici sono:

- Effect **1,6V**; lettura ADC **327**
- Power **2,5V**; lettura ADC **512**
- Auto/Man **3,4V**; lettura ADC **704**

Nella realtà i valori saranno prossimi a quelli teorici, ma non esatti. Per ovviare al problema è previsto un range di tolleranza di +/- 30 , regolabile nell'header file.

```
275     if(btnValue >= BTN1_MIN && btnValue <= BTN1_MAX && !cmdOk){
276 //Se il pulsante è appena stato premuto lo imposto come selezionato e azzero
277 //il conteggio antirimbalzo
278         if (btnSelected != BTN1){
279             btnSelected = BTN1;
280             btnDebounceCounter = 0;
281         }
282 //Viceversa incremento il conteggio
283         else{
284             btnDebounceCounter ++;
285         }
286     }
```

Qualora la lettura risultasse all'interno dei valori di un pulsante, viene incrementato il conteggio dell'antirimbalzo, utilizzato per prevenire falsi comandi. Una variabile memorizza quale pulsante è premuto. Quando il conteggio incrementa al tempo di antirimbalzo voluto, in questo caso 3 cicli, si

passa alla gestione vera e propria dei comandi:

```
307  //Quando il conteggio arriva al valore stabilito, passo ad eseguire il comando
308      if (btnDebounceCounter == DEBOUNCE_COUNT){
309  //Azzero il contatore e setto il flag del comando eseguito
310          btnDebounceCounter = 0;
311          cmdOk = 1;
312  //Controllo quale pulsante è stato premuto ed eseguo l'operazione associata
313          switch (btnSelected){
314              case EFFECT:
315                  effect =! effect;
316                  break;
317              case POWER:
318                  power =! power;
319                  TogglePower();
320                  break;
321              case MODE:
322                  mode =! mode;
323                  break;
324          }
325      }
326  //Quando il pulsante viene rilasciato azzero il flag per ricevere un nuovo
327  //comando
328      if (btnValue >= DEFAULT_VALUE){
329          cmdOk = 0;
330      }
```

Un flag (cmdOk) indica che i comandi sono stati eseguiti, questo per evitare che venga ripetuto il comando se il pulsante resta premuto.
Tramite uno Switch si ricava il pulsante premuto e vengono modificati i parametri relativi a quel comando. Quando si rilascia il pulsante, la lettura dell'ADC arriva a fondoscala. A questo punto il flag viene resettato per consentire di ricevere nuovi comandi.

Interfaccia seriale: configurazione e comandi

Una delle caratteristiche principali dell'RGB LED Controller è la presenza del convertitore USB-seriale, allo scopo di poter comandare il controller da PC. Di seguito è esposta la configurazione ed i comandi:

Parametri di configurazione della porta seriale

La comunicazione va impostata nel seguente modo:

Baudrate: 38400
Bit: 8
Bit di stop: 1
Parità: nessuna

Sintassi predefinita dei comandi

- Ogni comando valido è costituito da **4 byte**, inviati in sequenza, senza alcun carattere di terminazione.
- Nel caso il controller non riceva tutti i 4 byte entro un tempo di 10ms, esso annullerà i byte ricevuti e invierà notifica di **timeout**.
- Il primo byte è il comando da eseguire, i 3 successivi sono i parametri

ACK e NACK

- Qualora si invii un qualsiasi comando al controller, fatta eccezione per i comandi di controllo diretti delle barre (l, r, x), esso risponderà con il messaggio "**ACK**" ad indicare il riconoscimento del comando, e con il messaggio "**NACK**" ad indicare un comando errato.

Comandi di lettura dati

Il controller supporta alcuni comandi in lettura, per la richiesta di informazioni dallo stesso. I comandi disponibili sono i seguenti:

- **Leggi Sistema**

Richiede il nome del dispositivo. L'unico byte significativo è il primo.

Comando ASCII: sXXX (con X -> indifferente)
Comando HEX: 0x73 0x00 0x00 0x00

- **Leggi Autore**

Richiede il nome dell'autore del progetto. L'unico byte significativo è il primo.

Comando ASCII: aXXX (con X -> indifferente)
Comando HEX: 0x61 0x00 0x00 0x00

- **Leggi Versione Firmware**

Richiede la versione firmware. L'unico byte significativo è il primo.

Comando ASCII: vXXX (con X -> indifferente)
Comando HEX: 0x76 0x00 0x00 0x00

- **Leggi Versione Hardware**

Richiede la versione hardware. L'unico byte significativo è il primo.

Comando ASCII: hXXX (con X -> indifferente)

Comando HEX: 0x68 0x00 0x00 0x00

• **Leggi Data Ultima Modifica**

Richiede la data dell'ultima modifica al firmware. L'unico byte significativo è il primo.

Comando ASCII: dXXX (con X -> indifferente)
Comando HEX: 0x64 0x00 0x00 0x00

In Figura 4 è riportato un dettaglio dei rispettivi comandi inviati per mezzo di *RS232 Terminal*.

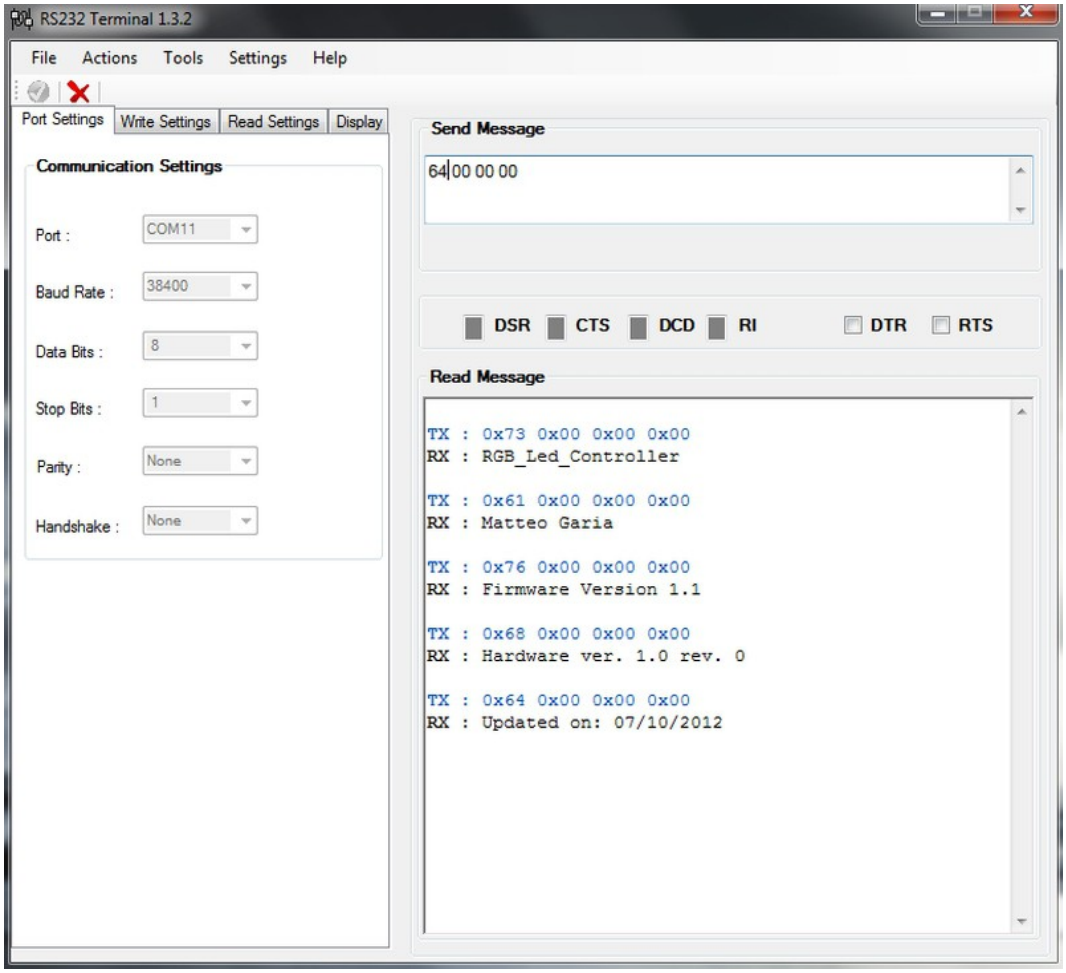

Figura 4: *Schermata di RS232 Terminal dove è possibile vedere la comunicazione con l' RGB LED controller.*

Comandi di scrittura dati

A seguire l'elenco dei comandi in scrittura. Tutti i byte sono significativi, ad eccezione del comando "c" che richiede un solo parametro.

Nota:

Come valore del singolo parametro si intende l'invio di un byte con quel preciso valore. Per cui un valore numerico viene inviato in ***un solo byte*** *e non inviando il numero in notazione ASCII (per cui un byte per ogni cifra). Tenere conto di ciò nella realizzazione di interfacce da PC.*

In VB.Net per inviare il valore direttamente e non la sequenza ASCII è possibile eseguire la conversione con:

Chr(Valore)

• Comando RGB barra "left"

In PC MODE assegna il colore alla barra "left"

Comando HEX: 0x6C RED GREEN BLUE (dove RED GREEN BLUE = valori 0-255)
Esempio stringa VB: "l" & Chr(RED) & Chr(GREEN) & Chr(BLUE)

• Comando RGB barra "right"

In PC MODE assegna il colore alla barra "right"

Comando HEX: 0x72 RED GREEN BLUE (dove RED GREEN BLUE = valori 0-255)
Esempio stringa VB: "r" & Chr(RED) & Chr(GREEN) & Chr(BLUE)

• Comando RGB barra "aux"

In PC MODE – AUX BAR assegna il colore alla barra "aux"

Comando HEX: 0x78 RED GREEN BLUE (dove RED GREEN BLUE = valori 0-255)
Esempio stringa VB: "x" & Chr(RED) & Chr(GREEN) & Chr(BLUE)

• Comando "c"

Consente di impartire alcuni comandi al controller

Comando HEX: 0x63 0x00 0x00 CMD
Esempio stringa VB: "c" & Chr(00) & Chr(00) & Chr(CMD)

I comandi riconosciuti sono:

- 0 -> disabilita il controllo da pc, passa al comando manuale
- 1 -> abilita il controllo da pc per le sole barre LEFT e RIGHT, mentre AUX rimane manuale
- 2 -> abilita il controllo da pc per tutte le barre
- 3 -> spegne il controller
- 4 -> accende il controller

• Stato Power ON

Quando il controller passa da Power OFF a Power ON, sia da pulsante, sia da porta COM, esso si avvierà sempre e comunque in modalità manuale. E' necessario quindi scegliere successivamente la modalità operativa con gli appositi comandi sopra citati.

Un esempio di invio comandi di scrittura è riportato in Figura 5.

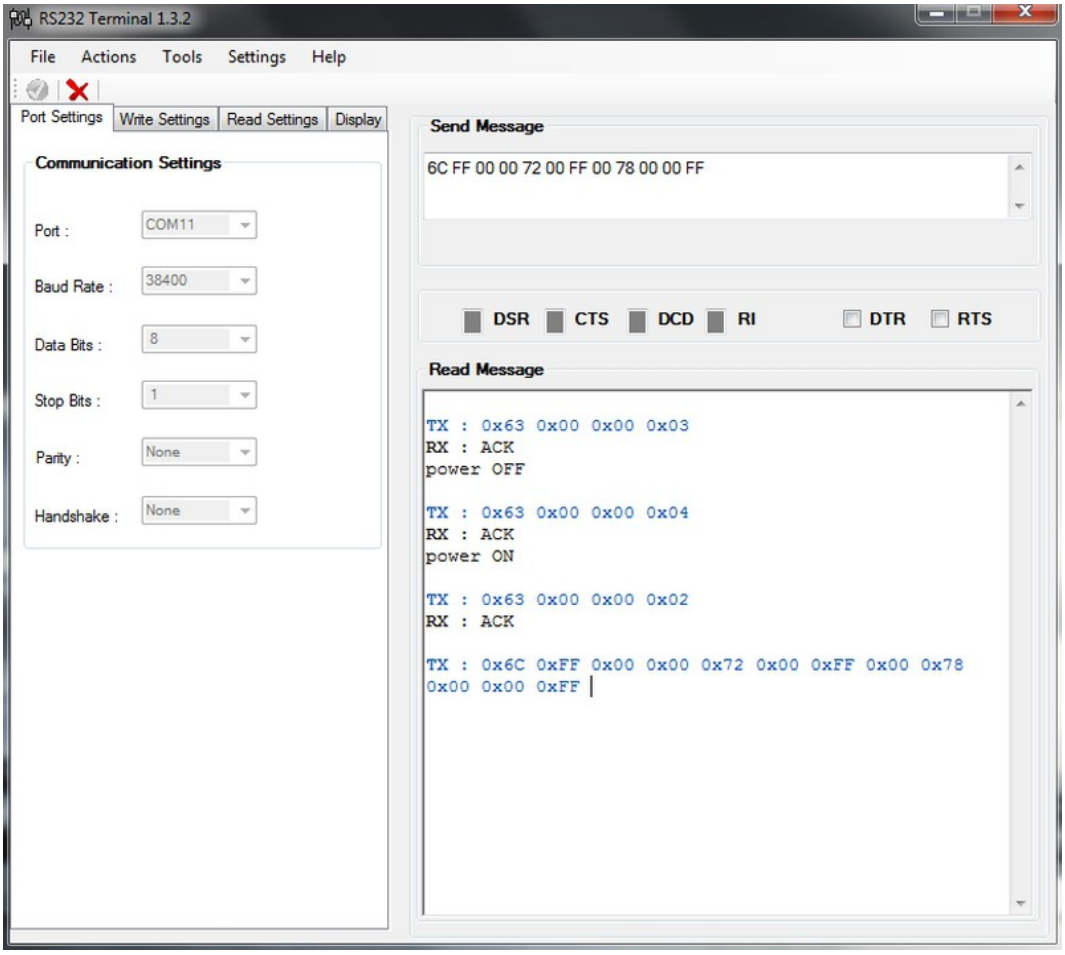

Figura 5: *Esecuzione di comandi da parte dell' RGB LED Controller: da stato acceso viene passato a spento, successivamente si abilita il comando completo da PC (comando 02) e infine ogni barra viene accesa con un colore primario. Il risultato è visibile qui sotto:*

In Figura 6 è riportata un'immagine del sistema pilotato con i precedenti comandi.

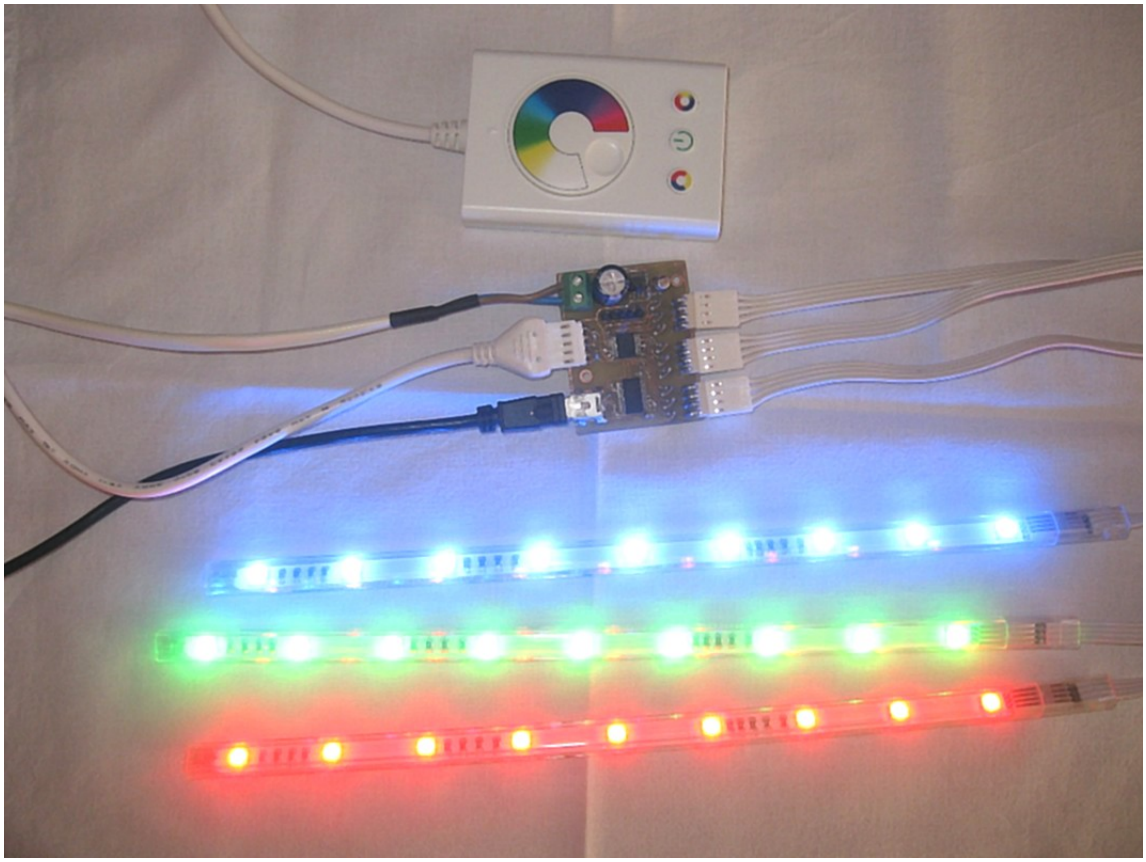

Figura 6: *Led accesi secondo il comando inviato da pc.*

Analisi finale

Il controller presentato può esser adottato per applicazioni in cui sia voluta un'illuminazione "creativa". La possibilità di comando da PC permette di realizzare anche effetti particolari, come è possibile vedere provando il software allegato.

La scheda ed il firmware non sono del tutto esenti da difetti dovuti a scelte costruttive ed allo sviluppo del firmware, in particolare:

- La corrente di 100mA a colore limita la quantità di led e la loro potenza, è comunque sufficiente sostituire i transistor per avere maggiore corrente disponibile.
- Per migliorare il firmware si potrebbe spostare la gestione del PWM all'interrupt di Timer1, permettendo di gestire il resto del codice senza rispettare le tempistiche strette necessarie.

Indice Alfabetico

Bibliografia

[1] www.LaurTec.com : Sito dove poter scaricare aggiornamenti dell'articolo e "*RS232 Terminal*".
[2] www.microchip.com : Sito dove scaricare i datasheet del PIC16F88 e MPLAB-X.
[3] www.ftdichip.com : Sito dove poter scaricare il datasheet del convertitore USB-UART FT232RL.
[4] www.usb.org : Sito ufficiale del consorzio USB.

History

Data	Versione	Nome	Descrizione Cambiamento
20.10.12	1.0	Matteo Garia	Versione Originale.