UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA ELETTRICA

TESI DI LAUREA MAGISTRALE

REALIZZAZIONE DI UNA SORGENTE A CROMATICITA' VARIABILE

utilizzando la tecnologia LED

RELATORE: Prof. Ing. Pietro Fiorentin

Dipartimento di Ingegneria Elettrica – Laboratorio di Illuminotecnica e Fotometria

CORRELATORE: Ing. Alessandro Scroccaro

Dipartimento di Ingegneria Elettrica – Laboratorio di Illuminotecnica e Fotometria

LAUREANDO: Marco Dal Monico

A.A. 2010 – 2011

Sommario

In questo lavoro di tesi si è realizzata una sorgente luminosa in grado di generare una luce a cromaticità variabile.
Uno strumento di questo genere si può rivelare utile per condurre alcuni esperimenti sulla percezione dei colori, o come strumento di calibrazione di alcuni strumenti di misura presenti nel Laboratorio di Illuminotecnica e Fotometria, in cui questa tesi è stata sviluppata.
La capacità della sorgente di fornire una cromaticità variabile è stata ottenuta utilizzando dei LED dei tre colori primari (rosso, verde, blu): miscelando la luce da essi prodotta in una certa proporzione, si riesce ad ottenere (quasi) qualsiasi colore si desideri.
Per scegliere il colore che vuole riprodurre, l'utente può utilizzare un'interfaccia grafica dal computer, appositamente sviluppata nel linguaggio di programmazione MatLab: una volta scelto il colore, attraverso uno stadio di interfaccia software / hardware, i LED vengono alimentati con un segnale tale per cui la luce risultante dalla miscela dei tre colori avrà la cromaticità desiderata.
Allo stadio attuale di sviluppo, la sorgente è funzionante e fornisce risultati accettabili; nei giorni precedenti la laurea, ma successivi alla stampa di questa tesi, si tenterà di effettuare alcune piccole aggiunte ai programmi in MatLab, per fare in modo che lo spazio colore L*a*b* goda di tutte le funzionalità che al momento possiede solo lo spazio Lxy.
In ogni caso, per rendere la sorgente adeguata agli usi sopra descritti, sarà necessario sistemare alcune imperfezioni causate da un comportamento non ideale di alcuni componenti.

Indice

Capitolo 1 - Cenni di teoria

1.1 La radiazione visibile

La radiazione visibile costituisce solo una piccola parte dello spettro elettromagnetico.

Essa è compresa essenzialmente fra le lunghezze d'onda di 380 e 780 nm, anche se in base alla definizione CIE (Commission Internationale de l'Eclairage), la radiazione visibile è compresa tra 360 e 820 nm.

(1 nm = un nanometro = un miliardesimo di metro).

Verso le lunghezze d'onda più corte, e quindi verso le frequenze (ed energie) maggiori, troviamo la luce blu / viola, fino ad arrivare nell'ultravioletto al di sotto dei 380 nm.
Verso le lunghezze d'onda più lunghe, e quindi verso le frequenze (ed energie per fotone) minori, troviamo invece il rosso, sconfinando nell'infrarosso per lunghezze d'onda maggiori di 700 nm.

400 nm 780 nm

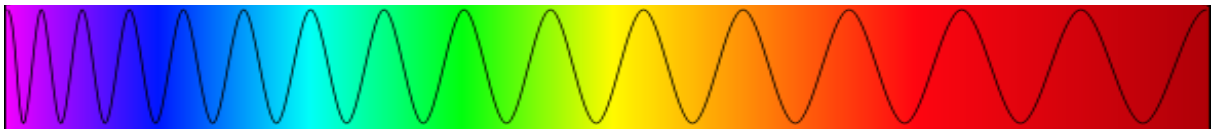


figura 1.1: spettro della radiazione visibile e lunghezze d'onda

1.2 L' occhio umano

L'occhio umano è costituito da molti elementi, ma quello che a noi più interessa, poiché direttamente coinvolto nella percezione dei colori, è sicuramente la retina.

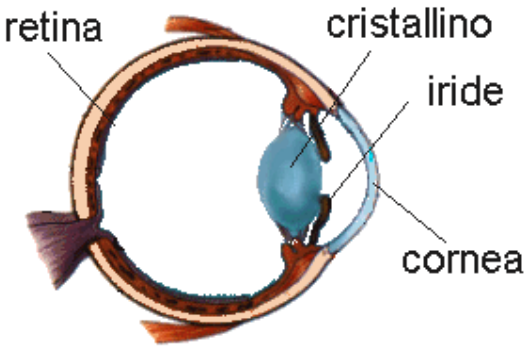


figura 1.2: principali componenti dell'occhio umano

La retina è posizionata sulla parte posteriore dell'occhio, ed è costituita dai fotorecettori, che hanno il compito di trasformare le informazioni ricevute in impulsi elettrochimici da inviare al cervello.
I fotorecettori presenti nella retina sono di due tipi:

- i coni (6 milioni, concentrati nella zona centrale della retina), che contribuiscono alla visione fotopica (ovvero diurna, con elevati livelli di luminanza) e che permettono di distinguere i colori;
- i bastoncelli (120 milioni, distribuiti su tutta la retina ,tranne che nella zona dei coni e in una zona chiamata "angolo cieco"), che contribuiscono alla visione scotopica (ovvero notturna, con bassi valori di luminanza), e che non sono in grado di riconoscere i colori degli oggetti.
  Le zone più periferiche della retina contengono solo bastoncelli, e non distinguono né la forma, né i colori degli oggetti.

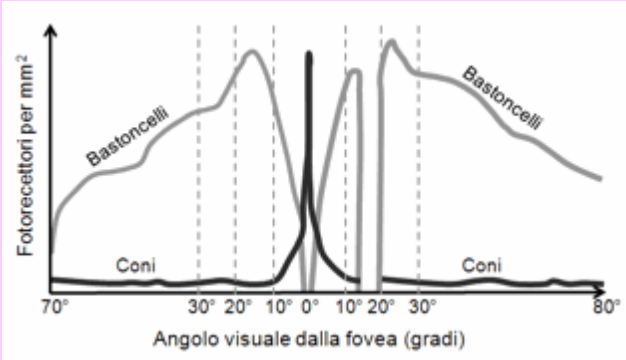


figura 1.3: distribuzione dei fotorecettori (coni e bastoncelli) nella retina

Abbiamo dunque capito che, per quanto riguarda i nostri scopi, sono i coni i fotorecettori che verranno utilizzati.
Nella retina esistono tre tipi di coni, ognuno dei quali è sensibile a diverse lunghezze d'onda: lunghe (L) – luce rossa (R) , medie (M) – luce verde (G), corte (S: Short) – luce blu (B).

Questo spiega all'istante perché tutta la tecnologia visiva si sia sviluppata con l'RGB.

La percezione cromatica di una determinata luce, quindi, è data dalla combinazione delle risposte dei tre tipi di cono: dipenderà quindi sia dalla luce, che dalla sensibilità dei diversi coni.

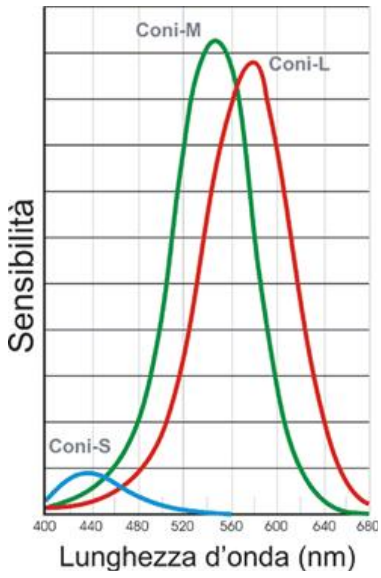


figura 1.4: sensibilità dei tre tipi
di coni presenti nella retina

Dal diagramma appena visto, si intuisce che l'occhio umano è molto sensibile al verde, quindi ci basta poca potenza per vedere un verde luminoso[1]; al contrario, siamo molto meno sensibili al blu, quindi serve molta potenza per avere la sensazione di un blu luminoso. Questo può essere pericoloso, perché prima che una luce molto blu dia fastidio, potrebbe aver già causato danni all'occhio.

Infatti, l'occhio umano è più sensibile alla lunghezza d'onda di circa 555 nm (che corrisponde ad una luce giallo-verde). Ciò significa che due gialli vicini vengono distinti molto bene, mentre due rossi vicini vengono distinti molto più a fatica; infine, due blu vicini saranno praticamente indistinguibili.

1 Si può intuire che questa maggiore sensibilità al colore verde sia dovuta al fatto che, per migliaia di anni, l'uomo ha vissuto all'aperto, in mezzo alla natura, dove i colori preponderanti della luce naturale erano appunto il giallo e il verde; il rosso era poco presente (al tramonto), ed il blu era presente di notte, quando l'occhio non veniva usato.

Questo è più facile da comprendere osservando la figura 1.5 di seguito riportata.

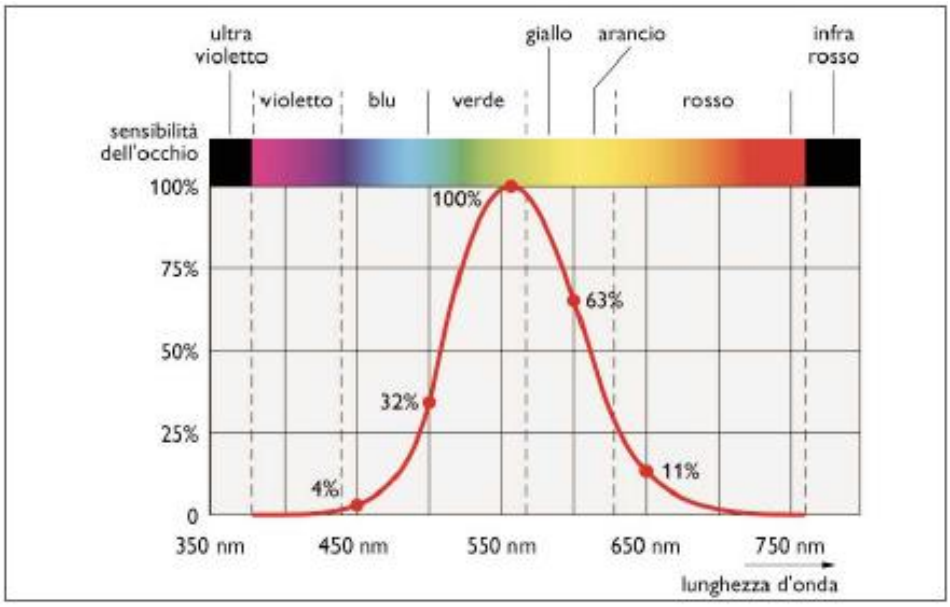


figura 1.5: sensibilità dell'occhio umano alle varie lunghezze d'onda

Volendo quantificare a spanne questa sensibilità, possiamo dire che tra i 500 nm e i 600 nm (quindi tra il giallo e il verde) l'occhio umano è in gradi di distinguere una differenza di lunghezza d'onda di 1-3 nm; verso il rosso si riescono a distinguere due colori con lunghezze d'onda distanti almeno 6 nm.

1.3 Grandezze fotometriche

Ci limiteremo, in questo ambito, a dare le definizioni di base e le unità di misura delle principali grandezze fotometriche che sono state utilizzate nel lavoro di tesi, e alla breve descrizione degli strumenti utilizzati per la loro misura, se necessario.

Ad ogni grandezza fotometrica X_V corrisponde una grandezza radiometrica X_e: la relazione tra le due è che una grandezza fotometrica è la corrispondente grandezza radiometrica, pesata con la sensibilità fotopica dell'occhio umano V(λ)

$$X_v = K_m \cdot \int_{\lambda_{min}}^{\lambda_{max}} X_e(\lambda) \cdot V(\lambda) d(\lambda)$$

dove Km = 683 [lumen/watt]

1.3.1 Flusso luminoso – Φv

Il flusso luminoso è la quantità totale di radiazioni emesse nell'unità de tempo da una sorgente primaria o secondaria, pesate con la sensibilità spettrale dell'occhio umano.

$$X_v = K_m \cdot \int_{\lambda_{min}}^{\lambda_{max}} P(\lambda) \cdot V(\lambda) d(\lambda)$$

L'unità di misura del flusso luminoso è il lumen [lm]

Per misurare il flusso luminoso, si utilizza la *sfera integratrice*: nella sfera integratrice, la sorgente viene messa nel centro di una sfera le cui pareti interne sono trattate con speciali vernici, le quali devono avere coefficiente di riflessione quanto più possibile vicino ad 1 ed essere il più possibile diffondenti; in questo modo, l'illuminamento in ogni punto è prodotto sia dalla sorgente che dagli altri punti della superficie interna della sfera, e quindi ogni punto è illuminato allo stesso modo indipendentemente dalla direzionalità della sorgente (idealmente).
Se mettiamo su una piccola parte della superficie interna della sfera un rivelatore di illuminamento, attraverso il valore letto sarà possibile risalire alla potenza della sorgente in prova. Per fare in modo che la misura di illuminamento non sia influenzata dall'illuminamento diretto prodotto dalla sorgente, si deve interporre tra quest'ultima e il rivelatore un setto che agisca da schermo.

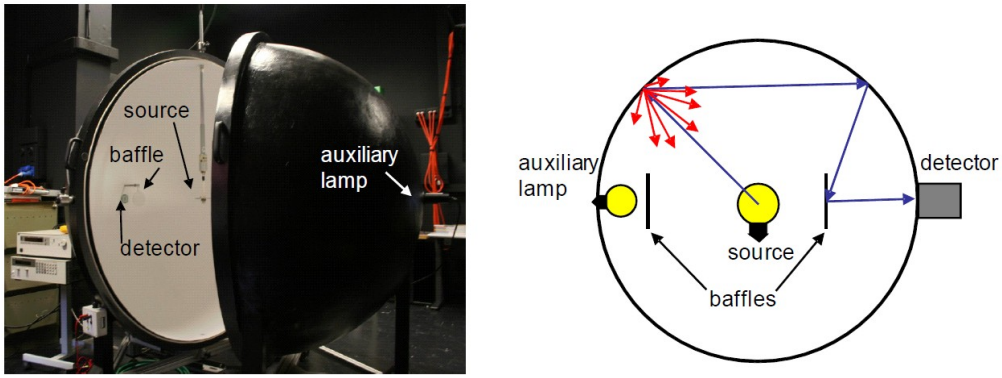


figura 1.6: sfera integratrice e suo principio di funzionamento

Come detto, è importante che le pareti interne della sfera integratrice abbiano un coefficiente di riflessione ρ quanto più possibile vicino ad 1: questo perché più alto è il coefficiente di riflessione, più alta è la luminanza della superficie interna della sfera a parità di flusso luminoso emesso dalla sorgente.

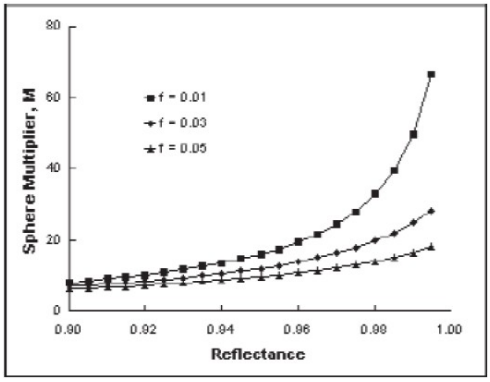


figura 1.7: luminanza della parete interna della sfera integratrice al variare del suo coefficiente di riflessione, e alla dimensione relativa del foro per il sensore

1.3.2 Luminanza – L

La luminanza può essere definita in due modi:

- come rapporto tra l'intensità luminosa emessa da una sorgente luminosa in una data direzione, e l'area apparente di quella superficie

$$L = \frac{dI}{dA_{app}}$$

- come flusso luminoso per angolo solido per area, proiettata perpendicolarmente alla direzione di osservazione

$$L = \frac{d^2\Phi}{d\Omega\, dA\, cos\vartheta}$$

L'unità di misura della luminanza è la candela su metro quadro [cd/m^2].

La grandezza radiometrica corrispondente alla luminanza è la *radianza* **Le**, che può essere sempre definita in entrambi i modi, ma considerando il flusso radiante invece di quello luminoso;
la sua unità di misura è [$W\ sr^{-1}\ m^{-2}$].

La luminanza gode di alcune importanti proprietà, che elenchiamo solamente:

- la luminanza è indipendente dalla distanza di osservazione;
- la luminanza è una quantità che si conserva nel sistema, e questa regola vale anche in presenza di lenti e sistemi ottici (quale potrebbe essere ad esempio l'iride): in pratica, la luminanza della sorgente luminosa e la luminanza della retina sono uguali; poi, la retina reagisca alla densità di flusso e converte la luminanza in illuminamento.

E' per questo motivo che **l'occhio umano è sensibile alla luminanza**.
Ovvero, **la luminanza è la luminosità**.

Per misurare la luminanza si utilizza il *luminanzometro*, che sfrutta la proprietà di conservazione della luminanza: a

valle di un sistema di lenti e di un filtro fotopico, la luminanza dell'elemento sensibile sarà uguale alla luminanza dell'oggetto osservato; questo elemento sensibile converte la luminanza in illuminamento, e quindi in corrente, ed è così possibile avere un valore numerico che indichi la luminanza.

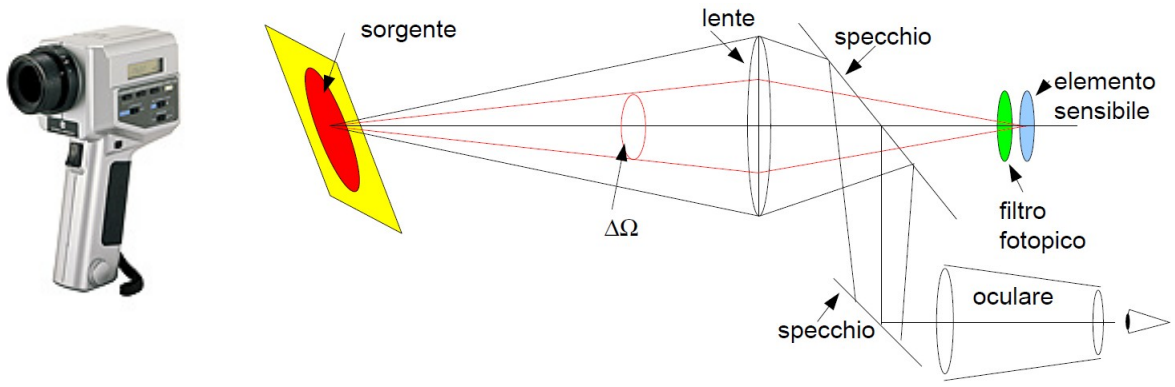


Figura 1.8: luminanzometro e suo principio di funzionamento

Se si omette il filtro fotopico, con la stessa strumentazione si ha una misura della radianza, presa però come valore totale (quindi, non è nota la radianza per ogni lunghezza d'onda).

Per misurare la radianza spettrale, ovvero la radianza per ogni singola lunghezza d'onda, si può invece utilizzare lo **spettroradiometro** (come è stato fatto durante il mio lavoro di tesi).

Lo spettroradiometro da noi utilizzato è il MINOLTA CS-1000:

figura 1.9: spettroradiometro Minolta CS-1000

Quella che proponiamo nella figura 1.10 è una rappresentazione semplificata del principio di funzionamento di uno spettroradiometro:

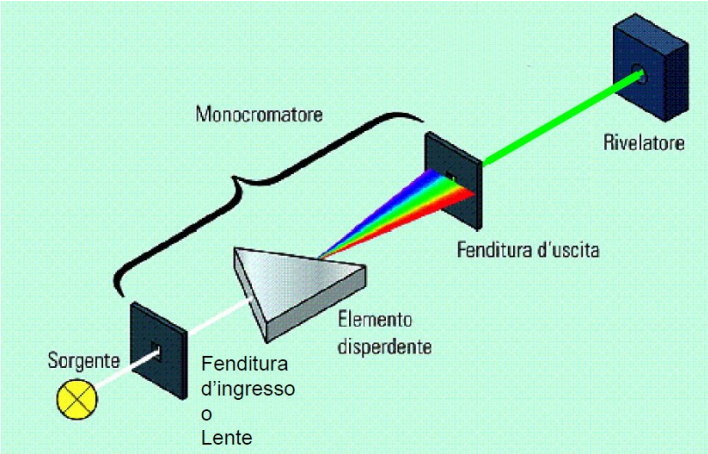


figura 1.10: principio di funzionamento di uno spettroradiometro

Spostando la fenditura di uscita del monocromatore, si analizza una lunghezza d'onda alla volta; in alternativa, si può togliere la fenditura di uscita, ed avere una fila di rivelatori: in questo modo si analizza contemporaneamente tutto lo spettro della sorgente, analizzandolo in 256, 512, 1024... intervalli.

Nello spettroradiometro che abbiamo utilizzato per questa tesi, il Minolta CS-1000, questo è il principio di funzionamento[2]:

"L'energia luminosa passsa attraverso l'obbiettivo. La luce proveniente dell'area di misurazione passa attraverso il foro al centro dello specchio dell'apertura fino ad arrivare alla fibra ottica, mentre la luce rimanente è guidata verso l'ottica nel mirino attraverso lo specchio apertura. Di conseguenza, la parte equivalente all'area di misurazione ha l'aspetto di un cerchio nero quando è osservata tramite il mirino.
La luce che penetra nella fibra ottica viene riflessa ripetutamente, in modo che si mescoli e diventi praticamente uniforme. Passa quindi attraverso l'obbiettivo del collimatore, fino al reticolo di diffrazione piano. Dopo essere stata dispersa dal reticolo, la luce viene focalizzata dall'obbiettivo del condensatore in base alla lunghezza d'onda. Un sensore matrice si trova in questo punto di messa a fuoco.
La quantità di energia rilevata per ciascuna lunghezza d'onda viene quindi convertita in un valore digitale tramite un convertitore A/D, in base al quale la radianza spettrale, la luminanza e il cromatismo vengono calcolati tramite l'elaborazione software."

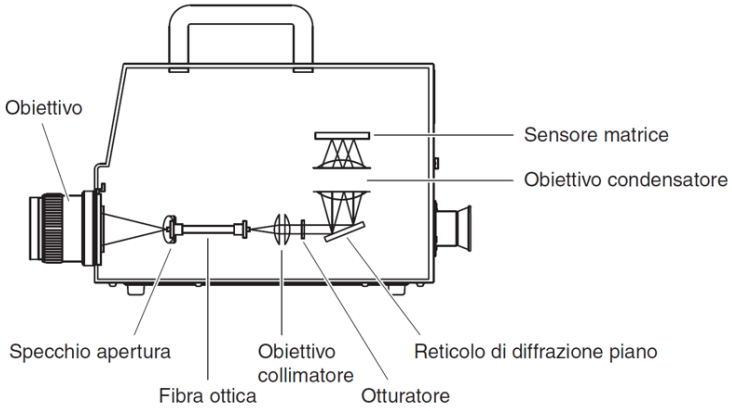


figura 1.11: elementi principali dello spettroradiometro Minolta CS-2000

2 Tratto dal manuale di istruzioni dello spettroradiometro Minolta CS-2000; il principio di funzionamento comunque non cambia.

1.3.3 Illuminamento – Ev

L'illuminamento è il rapporto tra il flusso luminoso ricevuto da un elemento di superficie e l'area di questo elemento.

$$E_v = \frac{d\Phi_v}{dA}$$

L'unità di misura dell'illuminamento è il lux [lx].

Nella caratterizzazione delle superfici, esiste un coefficiente che lega l'illuminamento alla luminanza: il *coefficiente di luminanza*

$$q = \frac{L}{E}[sr^{-1}]$$

Nel caso di riflessione diffondente, questo coefficiente di luminanza vale

$$q = \frac{\rho}{\pi}$$

dove ρ è il *coefficiente di riflessione*.

Perciò, nel caso di superficie perfettamente diffondente (come ad esempio all'interno della sfera integratrice) la luminanza e l'illuminamento saranno legati dalla legge

$$L = E \cdot \frac{\rho}{\pi}$$

questa legge verrà utilizzata più avanti, quando si illustreranno i calcoli fatti per la scelta della potenza della nostra sorgente luminosa.

1.4 Colorimetria

Abbiamo visto, in precedenza, che la presenza di tre diversi tipi di cono nell'occhio umano consente di esprimere un generico colore come combinazione di tre colori primari.

Qualsiasi colore, quindi, può essere rappresentato in uno spazio tridimensionale, in cui le coordinate di un colore sono legate alle quantità di questi primari necessarie per riprodurlo.

1.4.1 Leggi di Grassman (ridefinite da Hunt)

Queste considerazioni hanno portato alla formulazione delle leggi di Grassman:

1. Tre grandezze specificano il colore;
2. La sensazione di colore è proporzionale allo stimolo *(più o meno intenso, quindi dalla potenza del segnale)*;
3. Una mescolanza additiva di colori dipende solo dal loro aspetto *(non dipende dalle radiazioni in gioco, ma solo dalla percezione che noi abbiamo di quelle radiazioni)*.

Queste leggi sono state successivamente ridefinite da Hunt, per renderle più chiare e complete:

1. Una sensazione di colore è completamente specificata da tre grandezze: la tinta (il colore), l'intensità del colore (la saturazione) e l'intensità del bianco (la luminosità);
2. Se una luce varia con continuità, varia con continuità anche la sensazione di colore della mescolanza additiva con una seconda luce fissata;
3. Il risultato di una mescolanza additiva di colori dipende solo dal loro aspetto, e non dalla loro composizioni fisica (il loro spettro). Quindi si può riprodurre una stessa sensazione di colore utilizzando due sorgenti con diversa distribuzione spettrale.

Queste leggi stanno alla base del lavoro sviluppato in questa tesi: infatti, in questa tesi, si sintetizza un colore sfruttando la mescolanza additiva della luce prodotta da tre sorgenti che forniscono sensazioni di colore diverse.
Per assurdo, questo si poteva fare anche sfruttando tre lampadine che fornissero le stesse sensazioni di colore fornite dai nostri LED: questo fenomeno è chiamato *metamerismo* è spiegato nella terza legge.
In realtà, alla fine sono stati utilizzati i LED, grazie alle proprietà di miniaturizzazione di queste sorgenti luminose, che consentono di avere elevate potenze luminose con piccoli ingombri.

Dall'affermazione fatta sopra, si può intuire che il lavoro di questa tesi si basa su una sintesi additiva di colori; per completezza, ricordiamo che esiste anche una sintesi sottrattiva dei colori, nella quale la somma di tutti i colori dà il nero: è quanto avviene, ad esempio, nelle stampanti a getto d'inchiostro.

1.4.2 Gli spazi colore

Analogamente ai tipi di sintesi possibili, esistono spazi colore additivi e sottrattivi: noi tratteremo solamente quelli additivi.

Come detto prima, qualsiasi colore può essere rappresentato in uno spazio tridimensionale, che d'ora in poi chiameremo **spazio colore**.

Il primo spazio colore ad essere stato definito, nel 1931 ad opera della CIE, fu lo spazio **CIE XYZ**; nonostante sia stato il primo, e ne siano poi stati definiti molti altri, questo spazio ha ancora la sua validità: vediamo come è definito e capiremo il perché.

1.4.2.1 Spazio CIE XYZ (CIE 1931)

Lo spazio CIE XYZ venne definito sulla base di tre funzioni primarie fondamentali *x*, *y*, *z*, che danno rozzamente le sensazioni di verde, rosso e blu e che possono essere pensate come la risposta spettrale dei tre tipi di sensori presenti nell'occhio umano; queste funzioni sono chiamate *color matching functions*.

Integrando la radianza spettrale S(λ) di una certa sorgente, pesata con queste tre funzioni, si sono ottenute le coordinate tricromatiche X,Y e Z.

$$X = \int_{\lambda_{min}}^{\lambda_{max}} S(\lambda) \cdot \bar{x}(\lambda)\, d(\lambda)$$

$$Y = \int_{\lambda_{min}}^{\lambda_{max}} S(\lambda) \cdot \bar{y}(\lambda)\, d(\lambda)$$

$$Z = \int_{\lambda_{min}}^{\lambda_{max}} S(\lambda) \cdot \bar{z}(\lambda)\, d(\lambda)$$

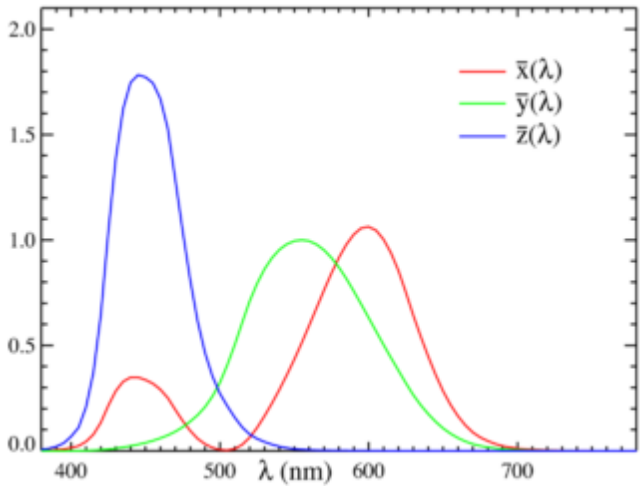


figura 1.12: color matching functions

Avendo scelto che la funzione *y* fosse esattamente uguale alla sensibilità fotopica V(λ) dell'osservatore standard CIE, si ottenne che la funzione Y corrispondesse alla luminanza: quindi la coordinata Y rappresenterà la quantità di bianco, la luminosità del colore.

Vedendo come sono state definite le coordinate di questo spazio, si capisce quale sia la sua più importante proprietà: quella di essere **lineare** rispetto alle sensazioni.
Infatti, in questo spazio valgono le proprietà di somma dei vettori relativi al tristimolo; vale quindi la regola del centro di gravità per la sintesi additiva (ovvero, se sommo due colori con certe coordinate cromatiche, le coordinate cromatiche del colore risultante saranno esattamente a metà strada tra quelle dei due colori iniziali).

Questo spazio non è però uniforme, nel senso che un'uguale distanza tra due punti non indica un'uguale differenza di cromaticità percepita.
Inoltre, anche se Y rappresenta la luminanza, tutti e tre i valori X,Y,Z dipendono dalla potenza del segnale luminoso.

1.4.2.2 Spazio CIE Lxy

Per descrivere la cromaticità in modo indipendente dalla luminanza, si decise di normalizzare ciascuna delle tre coordinate X,Y,Z rispetto alla somma X+Y+Z.

In questo modo si ottenne lo spazio CIE Yxy, chiamato anche CIE Lxy dato che Y rappresenta ancora la luminanza L.

Si può intuire che le tre coordinate cromatiche x,y,z non sono tra loro indipendenti, poiché la loro somma dà sempre 1; è per questo motivo che si utilizzano solo le prime due (x e y) per identificare la cromaticità del colore, tenendo Y come grandezza indipendente ad identificare la luminanza.
Anche dal punto di vista grafico, la rappresentazione dello spazio Lxy è ottenuta con lo stesso procedimento usato per calcolare le coordinate cromatiche: infatti, lo spazio Lxy è la proiezione sul piano (x,y) del piano con
x + y + z = cost = 1.

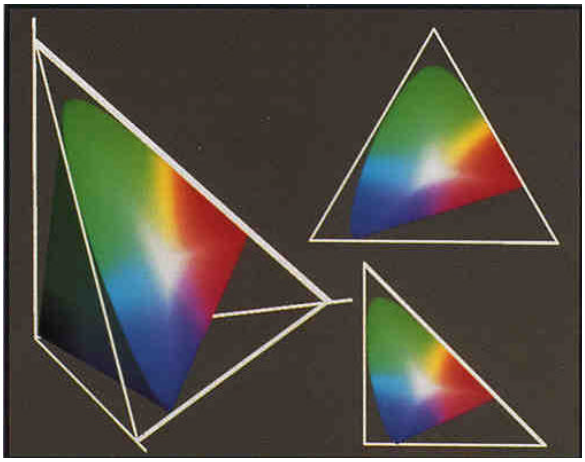


figura 1.13: spazio Lxy - vista tridimensionale, vista del piano x+y+z=1 e proiezione sul piano (x,y)

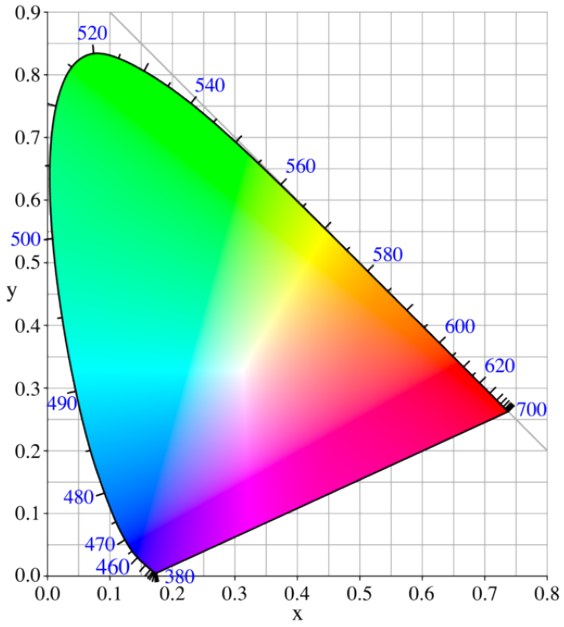


figura 1.14: spazio CIE Lxy con indicate le lunghezze d'onda delle radiazioni monocromatiche costituenti il luogo spettrale

La curva a ferro di cavallo riportata in figura 1.14, che indica il bordo dello spazio Lxy, è chiamata “luogo spettrale” (*spectral locus*), ed è ottenuta riportando nello spazio Lxy le coordinate cromatiche relative a radiazioni monocromatiche con lunghezza d'onda che va da 360 nm (blu) a 820 nm (rosso).

NOTA: d'ora in poi, si utilizzeranno indifferentemente i termini “bordo dello spazio Lxy”, “luogo spettrale” e “curva a ferro di cavallo”.

La linea che congiunge i punti a 360 nm e a 800 nm è chiamata “linea delle porpore” (*purple line*), ed è stata aggiunta per “chiudere” lo spazio: infatti, il color porpora non è un colore che può essere dato da una radiazione monocromatica a una certa lunghezza d'onda.

Durante la definizione di questo spazio, risulta che il “bianco equienergetico” (ovvero quello relativo ad una radiazione ad energia costante su tutto lo spettro) ha coordinate cromatiche
x = y = z = 1/3.

Lo spazio Lxy ha proprietà diverse dallo spazio XYZ, che è lineare: infatti, le coordinate cromatiche x,y,z vengono ottenute attraverso una normalizzazione, che è un'operazione non lineare.
La dimostrazione che non sia uno spazio lineare è data anche dal fatto che le lunghezze d'onda sulla curva del luogo spettrale non sono uniformemente distribuite: sono molto più rade nella zona del blu, e molto più concentrate nella zona del rosso.
Comunque, unendo due punti sul piano Lxy, i punti che cadono sulla linea di connessione rappresentano ancora tutti i colori che possono essere ottenuti miscelando quei due colori, anche se non vale più la legge del baricentro a causa della non linearità di questo spazio colore.

Notiamo che la zona dello spazio Lxy in cui è presente il colore verde è molto più estesa rispetto a quelle dove sono presenti il rosso e il blu: questo comporta che a due punti equidistanti non corrispondono colori ugualmente simili.
Questo fenomeno è stato studiato da Mac Adams: egli ha dimostrato che due punti a una certa distanza nel piano xy vengono riconosciuti facilmente se sono nella zona del rosso; due punti alla stessa distanza, ma nella zona del verde, potrebbero invece non essere distinti.
Per rappresentare questo fenomeno, egli disegnò delle ellissi, le quali includevano tutti i punti i cui colori fossero indistinguibili per l'osservatore: queste ellissi sono più grandi nella zona del verde, infatti per poter riconoscere due punti in quella zona, serve che essi abbiano una grande distanza, ragionando in termini di coordinate cromatiche.

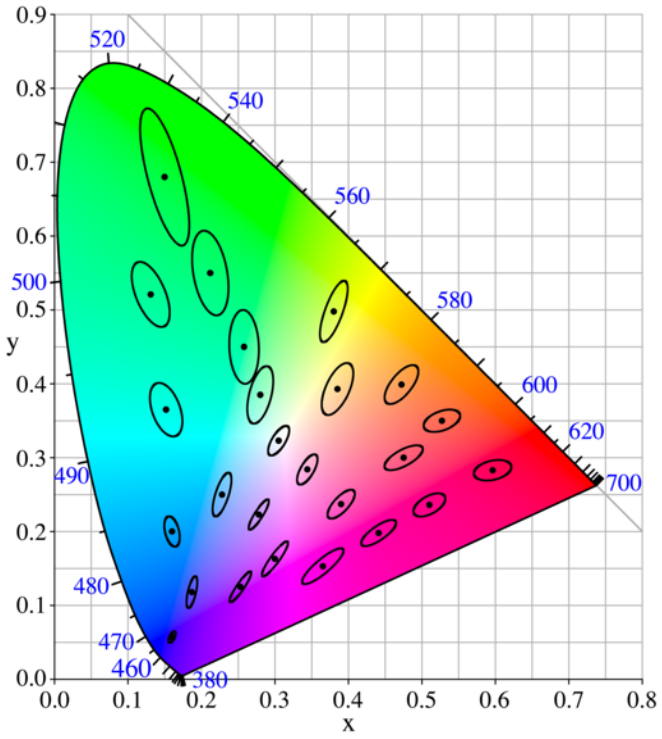


figura 1.15: ellissi di Mac Adams sul piano Lxy

Poiché era importante fornire uno spazio percettivamente più uniforme possibile, si crearono altri spazi colore, nel tentativo di coprire questa lacuna dello spazio Lxy (che viene comunque tuttora utilizzato per rappresentare un colore).

Tra questi spazi accenniamo appena allo spazio Lu'v', che venne creato nel 1976 come trasformazione dallo spazio XYZ; questo spazio venne anche soprannominato UCS (Uniform Color Space) per via della sua maggiore uniformità.

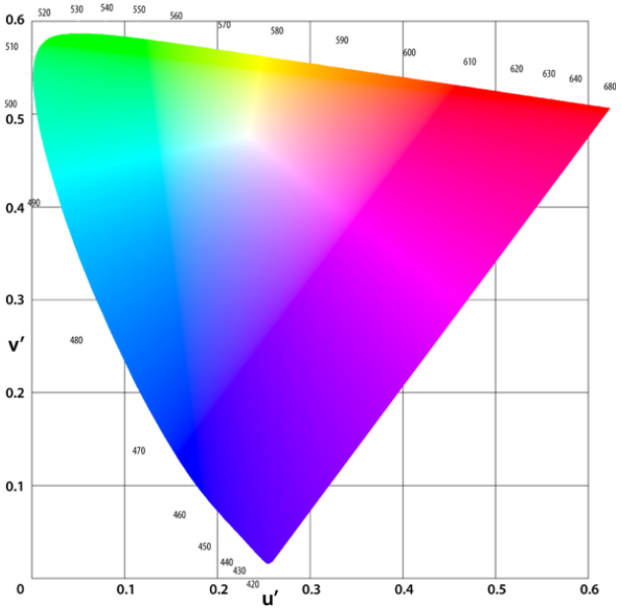


figura 1.16: spazio uniforme L u' v'

1.4.2.3 Spazio CIE Lab (CIE 1964)

Sempre allo scopo di fornire uno spazio percettivamente uniforme, nel 1964 venne creato lo spazio L*a*b*.

La sigla Lab, con la quale viene solitamente abbreviato il nome dello spazio CIE L*a*b*, in realtà può risultare ambigua, poiché esiste anche uno spazio Lab creato da Hunter nel 1948 (le cui coordinate sono L,a e b, mentre lo spazio CIE L*a*b* ha come coordinate L*,a*,b*).
Comunque, dato che lo spazio colore più utilizzato tra i due è proprio lo spazio CIE L*a*b*, la sigla Lab viene usata in relazione ad esso.
Anche nel seguito, quando parleremo di spazio Lab e di coordinate L,a,b intenderemo sempre parlare dello spazio L*a*b* e delle coordinate L*,a*,b* rispettivamente.

La creazione dello spazio Lab venne fortemente influenzata dal Sistema Munsell.

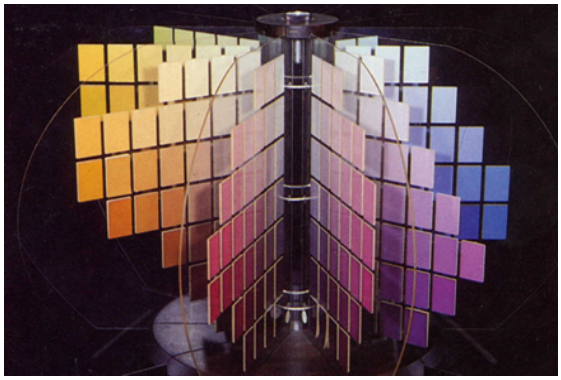


figura 1.17: "albero di Munsell"
(orig. "Munsell Tree")

In questo sistema, i colori sono ordinati in modo che la percezione di differenza di colore, tra due colori adiacenti, sia sempre costante nelle tre direzioni dello spazio:

- tinta (hue) costante – su ogni semipiano a partire dall'asse dei grigi;
- chiarezza (value) costante – su ogni piano orizzontale;
- saturazione (chroma) costante – su ogni superficie cilindrica equidistante dall'asse dei grigi.

Analogamente al sistema Munsell, lo spazio Lab è molto uniforme rispetto alle sensazioni: anche se non lo è completamente, è comunque uno degli spazi colore più uniformi che sia stato definito dalla CIE, ed in ogni caso il più completo: esso descrive tutti i colori visibili all'occhio umano, e fu creato come modello “device independent” per essere usato come riferimento.

Inoltre, dato che ad uguali distanze tra punti corrispondono uguali differenze tra gli stimoli, lo spazio Lab risulta essere uno spazio in cui è definita una metrica: infatti, nello spazio Lab può essere definita la “distanza”, e ciò può essere utilizzato per quantificare la differenza di colore.

Le tre coordinate vengono calcolate con relazioni non lineari dai valori tristimolo X,Y,Z dopo aver definito i valori tristimolo Xn, Yn, Zn relativi al “punto bianco di riferimento”.
Questo punto bianco di riferimento può variare, in base all'uso che deve essere fatto delle coordinate definite in Lab: ad esempio, per la rappresentazione a monitor, come punto bianco di riferimento vengono utilizzati i valori tristimolo

ottenuti con uno spettro D65.

La coordinata L può variare da 0 a 100, mentre le coordinate a,b hanno un intervallo di variazione che dipende dal valore di L; indicativamente, variando a da valori negativi verso valori positivi, il colore varia dal verde al rosso; variando b da valori negativi verso valori positivi, il colore varia dal blu al giallo.

Nonostante lo spazio Lab venga solitamente rappresentato come una sfera, in realtà la sua forma è più complessa: si veda questa rappresentazione, in cui il solido interno è quello degli stimoli forniti dai colori non autoluminosi, mentre l'involucro esterno è ottenuto trasformando nello spazio Lab i colori spettrali monocromatici e quelli rappresentanti la linea delle porpore dello spazio Lxy.

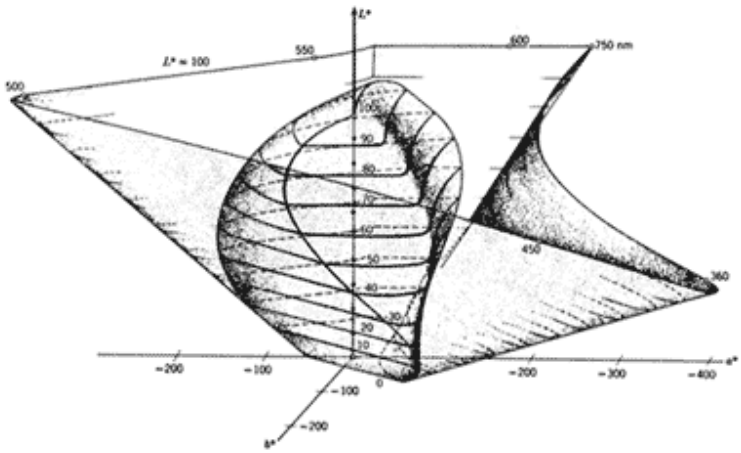


figura 1.18: solidi dei colori autoluminosi (interno) e dei colori spettrali monocromatici (esterno) nello spazio Lab

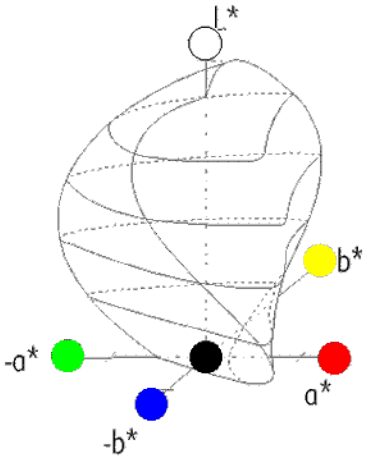


figura 1.19: solido degli stimoli autoluminosi ("Object COlour Stimuli") nello spazio Lab

1.4.2.4 Spazio HSV

Lo spazio HSV (Hue, Saturation, Value), pur non essendo particolarmente uniforme, viene spesso utilizzato perchè è molto intuitivo quando si deve scegliere un colore.

Infatti, lo spazio HSV è una rappresentazione in coordinate cilindriche dello spazio RGB, in cui la tinta (H: Hue) può essere scelta variando l'angolo da 0° a 360° (partendo dal rosso, si incontrano, nell'ordine, il verde, il blu e di nuovo il rosso); la distanza dall'asse del cono rappresenta la saturazione (S: Saturation), variabile tra 0 quando si è in corrispondenza dell'asse del cono e 1 quando si è sul bordo del cono; lo spostamento lungo l'asse rappresenta infine la chiarezza (V: Value), variabile tra 0 sul vertice del cono e 1 sulla sua base.

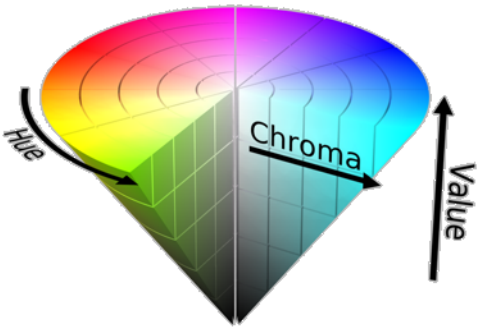


figura 1.20: rappresentazione dello spazio HSV come cono rovesciato

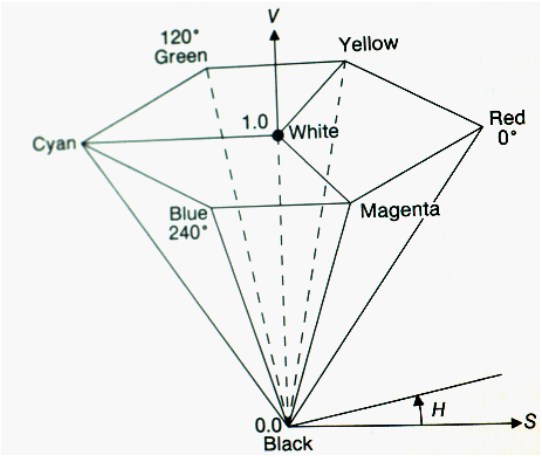


figura 1.21: rappresentazione dello spazio HSV come piramide esagonale

Più spesso, lo spazio HSV viene rappresentato come una piramide esagonale, per via della trasformazione inversa da HSV ad RGB: infatti per effettuare quest'ultima si immagina che lo spazio HSV sia la proiezione su un piano del cubo RGB, posizionato vertice corrispondente al colore nero.

In ogni caso, data la forma dello spazio HSV, si intuisce subito che il valore della massima saturazione S raggiungibile dipenderà dal valore della chiarezza V: alla massima chiarezza (V=1) si potranno scegliere i colori alla loro massima saturazione, mentre man mano che la chiarezza cala, diminuisce anche la massima saturazione che può essere data al colore.

1.4.3 Trasformazioni matematiche tra i vari spazi colore

Le trasformazioni da uno spazio colore ad un altro sono quasi sempre operazioni non lineari; solitamente, lo spazio di partenza è lo spazio CIE XYZ, i cui valori tristimolo vengono coinvolti in queste trasformazioni per ottenere delle coordinate cromatiche in spazi colore di più semplice lettura, o anche con migliore uniformità percettiva.

Nel programma sviluppato per questa tesi, al contrario, i valori che identificano un certo colore vengono scelti direttamente negli spazi Lxy, L*a*b* e HSV (anche se in realtà, lo spazio HSV è stato adottato a causa di un malinteso: infatti inizialmente si voleva usare lo spazio Lch, ovvero la trasformazione dello spazio L*a*b* in coordinate cilindriche).
Viene quindi applicata una trasformazione inversa, che permette di ottenere i valori tristimolo nello spazio XYZ, dai quali vengono poi calcolati i valori in RGB necessari per alimentare i LED.

Per questo motivo in questo paragrafo si riportano le trasformazioni dirette ed inverse, quindi da e per lo spazio XYZ.

1.4.3.1 XYZ ↔ Lxy

Trasformazione diretta: XYZ → Lxy

$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

$$z = \frac{Z}{X + Y + Z} = 1 - x - y$$

Trasformazione inversa: Lxy → XYZ

$$Y = L$$

$$X = \frac{Y}{y} \cdot x$$

$$Z = \frac{Y}{y} \cdot (1 - x - y)$$

1.4.3.2 XYZ ↔ Lab

Prima di eseguire la trasformazione, deve essere definito il punto bianco di riferimento, con valori tristimolo Xn, Yn, Zn.

Il punto bianco di riferimento può anche essere definito da Yn = 1, xn = yn = 1/3 e da questi valori possono essere ricavati Xn e Zn con le trasformazioni appena viste.

Trasformazione diretta: XYZ → Lab

$$L^* = 116\, f(Y/Y_n) - 16$$

$$a^* = 500\, f[(X/X_n) - (Y/Y_n)]$$

$$b^* = 200\, f[(Y/Y_n) - (Z/Z_n)]$$

dove

$$f(t) = \begin{cases} t^{1/3} & se\ t > (\frac{6}{29})^3 \\ \frac{1}{3}(\frac{29}{6})^2\, t + (\frac{4}{29}) & altrimenti \end{cases}$$

Trasformazione inversa: Lab → XYZ

la trasformazione inversa è espressa in modo semplice, utilizzando la funzione inversa delle funzione f(t) di cui sopra:

$$Y = Y_n\, f^{-1}\left(\frac{L^* + 16}{116}\right)$$

$$X = X_n\, f^{-1}\left(\frac{L^* + 16}{116} + \frac{a^*}{500}\right)$$

$$Z = Z_n\, f^{-1}\left(\frac{L^* + 16}{116} - \frac{b^*}{200}\right)$$

dove

$$f^{-1}(t) = \begin{cases} t^3 & se\ t > (\frac{6}{29}) \\ 3\,(\frac{6}{29})^2\,(t - \frac{4}{29}) & altrimenti \end{cases}$$

1.4.3.3 XYZ ↔ CIE RGB

Trasformazione inversa: XYZ ← CIE RGB

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.91240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

la trasformazione XYZ → RGB, essendo lineare, può essere definita semplicemente invertendo la matrice appena vista.

1.4.3.4 XYZ ↔ RGB LED

Trasformazione diretta: RGB LED → XYZ

La matrice riportata di seguito è stata ottenuta da calcoli sulle radianze spettrali dei LED costituenti la sorgente a cromaticità variabile, e viene riportata dopo essere stata prelevata dal workspace di Matlab.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 4038,2 & 494,05 & 2327,2 \\ 1781,6 & 1312,1 & 1484,9 \\ 1,7594 & 94,216 & 14574 \end{bmatrix} \begin{bmatrix} R_{LED} \\ G_{LED} \\ B_{LED} \end{bmatrix}$$

Trasformazione inversa: XYZ → RGB LED

$$\begin{bmatrix} R_{LED} \\ G_{LED} \\ B_{LED} \end{bmatrix} = 10^{-5} \cdot \begin{bmatrix} 29,561 & -10,871 & -3,6127 \\ -40,431 & 91,642 & -2,8814 \\ 0,2578 & -0,59113 & 6,8807 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

1.4.3.5 HSV ↔ RGB

Per effettuare queste trasformazioni, si pensa che la forma esagonale dello spazio HSV sia data dalla proiezione su un piano del cubo RGB che posa sullo spigolo corrispondente al colore nero:

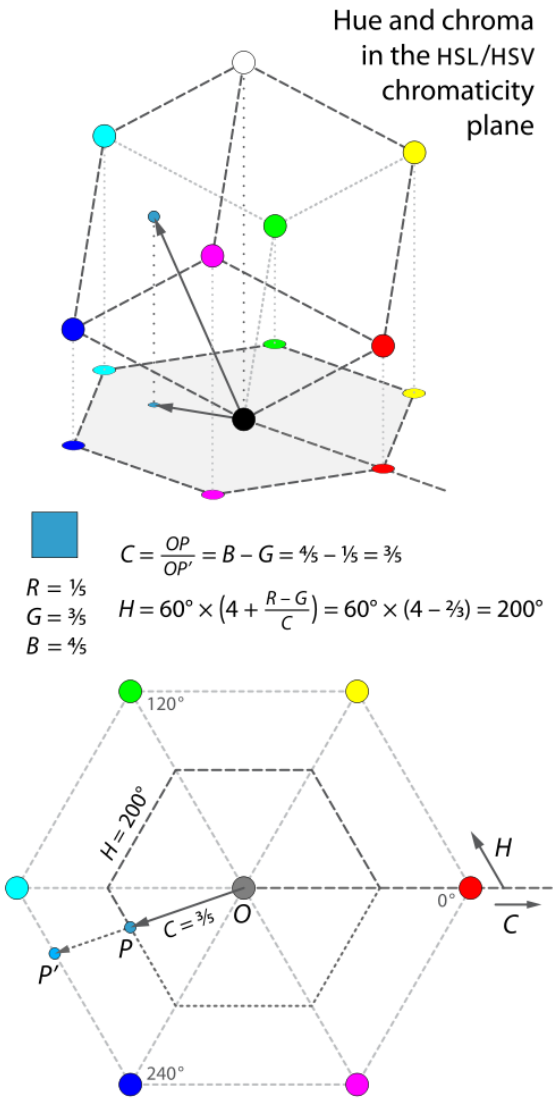


figura 1.22: rappresentazione grafica utile per comprendere la trasformazione da HSV a RGB e le relative grandezze

Trasformazione diretta: HSV → RGB

dato un colore con tinta H compresa nell'intervallo [0,360), saturazione S compresa nell'intervallo [0,1] e chiarezza V compresa nell'intervallo [0,1], si calcola per prima la C (*chroma*):

$$C = V \cdot S$$

quindi, si calcola un punto (R_1, G_1, B_1) giacente su una delle tre facce inferiori del cubo RGB (quelle che hanno, ad un vertice, il colore nero: vedi l'immagine sopra), con la stessa tinta e saturazione del nostro colore, ma più scuro poiché sta su una delle facce inferiori:

$$H' = \frac{H}{60°}$$

$$X = C\,(1 - |H'\,mod 2 - 1|)$$

$$(R_1, G_1, B_1) = \begin{cases} (0,0,0) & se\ H\ non\ definito \\ (C,X,0) & se\ 0 \leq H' \leq 1 \\ (X,C,0) & se\ 1 \leq H' \leq 2 \\ (0,C,X) & se\ 2 \leq H' \leq 3 \\ (0,X,C) & se\ 3 \leq H' \leq 4 \\ (X,0,C) & se\ 4 \leq H' \leq 5 \\ (C,0,X) & se\ 5 \leq H' \leq 6 \end{cases}$$

infine, possiamo trovare i valori R,G,B aggiungendo la stessa quantità ad ogni componente, per raggiungere la corretta posizione del colore all'interno del cubo RGB, e quindi la corretta chiarezza del colore:

$$m = V - C$$

$$(R, G, B) = (R_1 + m, G_1 + m, B_1 + m)$$

Trasformazione inversa: RGB → HSV

si calcolano

$$M = max(R, G, B)$$

$$m = min(R, G, B)$$

C (Chroma), che rappresenta la distanza del punto dall'origine, può essere calcolata come

$$C = M - m$$

la tinta H (Hue) invece viene calcolata in questo modo, passando per la variabile H':

$$H' = \begin{cases} non\ definito & se\ C = 0 \\ \frac{G-B}{C} mod 6 & se\ M = R \\ \frac{B-R}{C} + 2 & se\ M = G \\ \frac{R-G}{C} + 4 & se\ M = B \end{cases}$$

$$H = 60° \cdot H$$

La chiarezza V è V = M

la saturazione viene calcolata come:

$$S = \begin{cases} 0 & se\ C = 0 \\ C/V & altrimenti \end{cases}$$

1.4.3.6 XYZ → sRGB

Questa trasformazione verrà considerata solo da XYZ a sRGB, poiché essa viene utilizzata nel programma per fornire un'anteprima attendibile sullo schermo.

Per calcolare i valori sRGB, vengono dapprima calcolati dei valori intermedi, ovvero dei valori RGB *lineari*, partendo dai valori tristimolo, che devono essere stati precedentemente normalizzati, e quindi devono essere compresi nell'intervallo [0,1]:

$$\begin{bmatrix} R_{lin} \\ G_{lin} \\ B_{lin} \end{bmatrix} = \begin{bmatrix} 3,2406 & 0,3576 & 0,1805 \\ 0,2126 & 0,7152 & 0,0722 \\ 0,0193 & 0,1192 & 0,9505 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

successivamente, a questi valori intermedi viene applicata la seguente trasformazione (dove C rappresenta uno qualsiasi tra R,G o B):

$$C_{sRGB} = \begin{cases} 12,92 \cdot C_{lin} & se\ C_{lin} \leq 0,0031308 \\ (1+a) \cdot C_{lin}^{1/2.4} - a & se\ C_{lin} > 0,0031308 \end{cases}$$

con a = 0,055.

1.4.3.7 RGB → sRGB

La trasformazione appena vista può essere sostituita con la forma semplificata della correzione esponenziale di RGB, che consiste nell'applicare una gamma pari a 2,2 a ciascun valore R,G,B.

$$sRGB = RGB^{2.2}$$

1.5 la tecnologia LED

Nonostante alcuni la definiscano “la fonte luminosa del futuro”, la tecnologia LED non è affatto giovane: infatti, i primi LED apparvero già nella prima metà degli anni '60.

L'acronimo LED sta per Light Emitting Diode (ovvero diodo ad emissione luminosa): infatti un LED ha una struttura simile a quella di un diodo (quindi è composto da una giunzione P-N), e grazie all'utilizzo di particolari materiali per la giunzione si riesce a fare in modo che la ricombinazione elettrone – lacuna emetta dei fotoni.

Il colore della luce emessa dai LED dipende dai materiali utilizzati per la giunzione: i LED rossi furono i primi ad essere costruiti, ed utilizzavano l'arseniuro di gallio (GaAs); successivamente, si costruirono LED che emettevano luce verde; solo negli anni '90 si ottennero dei LED a luce blu, e mischiando questi tre colori si riuscì ad ottenere qualsiasi altro colore.

Molto più recenti sono i LED che emettono luce bianca: questi furono derivati dai LED a luce blu, applicando uno strato di fosfori sulla cupola per assorbire alcune lunghezze d'onda e riemetterle a lunghezze d'onda superiori (quindi a energia minore).

I LED normali (quelli utilizzati come spie luminose) vengono solitamente alimentati con un generatore a tensione costante, utilizzando una resistenza in serie per limitare la corrente che li attraversa.
La corrente assorbita da dei LED di questo tipo è di qualche unità o decina di milliampere.

I Power LED, invece, necessitano di un'alimentazione a corrente costante, poiché solo in quel modo si può garantire un corretto funzionamento anche di fronte ad instabilità della tensione di alimentazione.

La corrente assorbita dai Power LED è di centinaia di milliampere, in base alla potenza del LED ed alla tensione di alimentazione; in qualche caso si può anche superare l' ampere di corrente.

La tensione di alimentazione dei LED (normali o Power LED, indifferentemente) dipende dal materiale di cui è costituita la giunzione, e quindi varia con il colore della luce emessa: i LED a luce blu e bianca richiedono una tensione di alimentazione maggiore (tra i 3 V e i 4 V), mentre i LED a luce gialla e rossa richiedono una tensione di alimentazione tra i 2 V e i 3 V.

I LED hanno molteplici vantaggi: lunga durata, elevata efficienza e quindi basso consumo, miniaturizzazione, elevata luminosità, facilità di convogliamento del fascio luminoso.

Questi vantaggi fanno sì che la tecnologia LED stia prendendo sempre più piede nell'illuminazione, dove contano molto l'assenza di componenti IR e UV e quindi l'alta efficienza luminosa (anche più di 60 lm/W, contro meno di 20 lm/W di una lampada ad incandescenza), e la lunga durata di vita (che, lo ricordiamo, è valutata in certe condizioni di funzionamento spesso difficili da ottenere nel normale funzionamento).

Quello che al momento sembra essere il limite dei LED, nonostante negli ultimi tempi siano usciti prodotti con potenze notevoli (fino a 10W per ogni chip), è il flusso luminoso totale.
Infatti, mediamente il flusso luminoso per un LED di potenza di colore bianco è di 120 lm, mentre una lampada ad incandescenza da 60W emette un flusso di circa 550 lm.

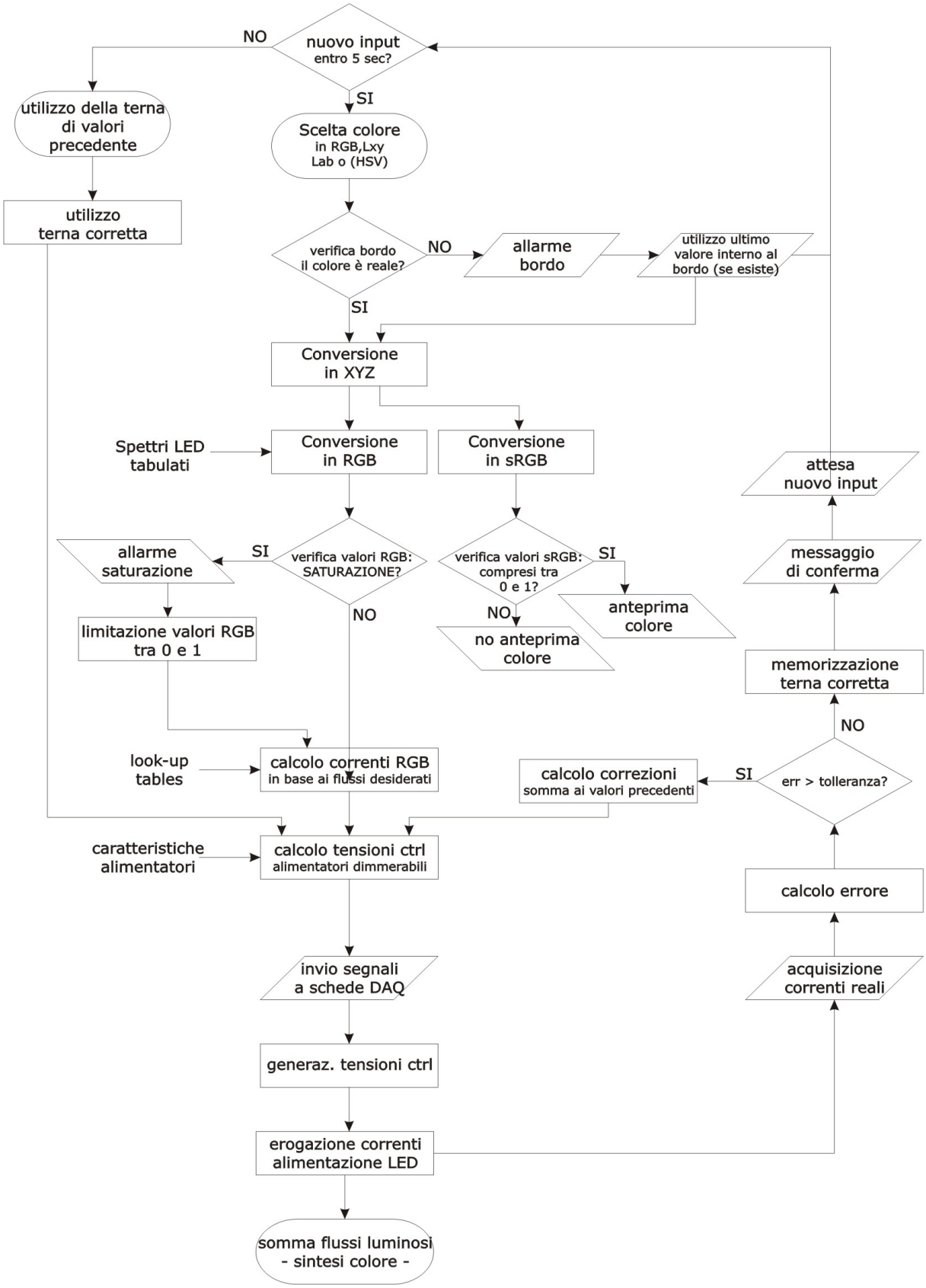
Bisognerà quindi attendere ancora perché i LED possano rappresentare una vera alternativa alle altre sorgenti luminose, specialmente nelle applicazioni in cui sia importante avere flussi luminosi elevati (ad esempio, nell'illuminazione stradale).

In ogni caso, data la grande varietà di colori disponibili, i LED rappresentano la sorgente luminosa più adatta quando si tratti di ottenere particolari effetti cromatici, sia per scopi di decorazione, sia per quanto riguarda l'illuminazione degli ambienti.

Proprio riguardo a quest'ultima applicazione, una nota azienda che si occupa di illuminazione ha messo in commercio delle lampade a LED che permettono di variare la temperatura di colore del bianco prodotto.
In questo modo è possibile riprodurre, con le varie tonalità di bianco, le tonalità che assume la luce solare durante il giorno: più fredda all'alba, bianca a mezzogiorno e più calda alla sera.
Sembra che questo possa avere effetti benefici sul corpo umano, perché aiuta a regolare i bioritmi durante la giornata.

Nonostante non sia questo lo scopo principale di questa tesi, con la sorgente che è stata creata sarebbe possibile gestire effetti di questo genere, anche se sarebbe decisamente una limitazione: infatti, con la nostra sorgente, si può riprodurre (quasi) qualsiasi colore si desideri.

Capitolo 2 - Visione di insieme del progetto:



Descriviamo a grandi linee il funzionamento della sorgente, trattando contemporaneamente sia gli aspetti hardware che quelli software. Nei prossimi capitoli, questi vari aspetti verranno trattati separatamente e più in dettaglio.

Partiamo dal momento in cui l'utente sceglie, dall'interfaccia grafica, il colore: prima di tutto, l'utente sceglie in che spazio colore lavorare; poi, immette tramite le tre caselle di testo o le tre *sliders*, i tre valori che identificano il colore in quello spazio colore.

Appena il programma riceve i tre valori, verifica se il punto identificato da essi ricade all'interno del bordo dello spazio colore (verifica attiva attualmente solo per Lxy, e in via di inserimento per Lab; per RGB non serve). Se il punto ricade al di fuori dello spazio colore, si attiva l'*allarme bordo* e il programma si mette in pausa fino a quando non viene richiesto un punto interno al bordo dello spazio colore.
Se invece il punto ricade all'interno del bordo dello spazio colore, e quindi è realizzabile, i tre valori vengono convertiti dallo spazio colore di partenza in XYZ.
Questi valori in XYZ vengono convertiti in RGB per comandare i LED, e in sRGB per fornire l'anteprima a monitor del colore scelto.

Se la terna RGB ha valori al di fuori del range [0,1] significa che si sta lavorando al di fuori del bordo del triangolo RGB riproducibile con i LED, e quindi il colore non potrà essere riprodotto esattamente: in questo caso, si attiva l'*allarme saturazione*, e i valori della terna vengono limitati nell'intervallo [0,1].

Se la terna sRGB ha valori al di fuori del range [0,1] significa che il colore non può essere riprodotto con la stessa luminanza e con la stessa tinta dal monitor del computer: quindi, viene visualizzato il messaggio “anteprima a monitor non disponibile”. Se invece la terna sRGB ha valori tutti compresi nell'intervallo [0,1], nell'apposita casella viene visualizzata l'anteprima a monitor del colore scelto.

La terna RGB che si ha adesso è una terna di flussi: dato che, nei LED, il flusso e la corrente non sono perfettamente proporzionali, attraverso l'utilizzo di una look-up table per ogni canale viene calcolato il valore di corrente da imporre al LED per ottenere il flusso desiderato.

In base alla terna di correnti di cui si dispone ora, viene calcolata la tensione di controllo da fornire agli alimentatori, per far sì che essi diano in uscita la corrente desiderata.

Questa tensione di controllo viene inviata alla scheda DAQ, la quale la fornisce alle apposite uscite che comandano gli alimentatori; in funzione della tensione di controllo impostata, gli alimentatori generano una corrente costante del valore desiderato, con cui vengono alimentati i LED.

In base alla corrente di alimentazione, ogni LED produce un certo flusso luminoso, che si va a sommare a quello degli altri LED, andando così a miscelare i vari colori per sintetizzare il colore desiderato.

La corrente di alimentazione dei LED viene letta (leggendo la tensione ai capi di una resistenza di shunt per ogni canale), acquisita dalla scheda DAQ ed inviata al programma; in base al valore delle correnti reali, viene calcolato l'errore rispetto alle correnti desiderate, ed un programma di retroazione applica delle correzioni alla terna di correnti, e quindi alle tensioni di controllo degli alimentatori, finché la corrente reale non coincide con quella desiderata, a meno di una tolleranza dello 0,5‰.

Quando tutte e tre le correnti rientrano nella tolleranza, la terna corretta viene memorizzata, e viene dato un messaggio di conferma del termine della retroazione; se entro 50 cicli tutte e tre le correnti non sono rientrate nella tolleranza, viene dato un messaggio in cui si avvisa che la retroazione non è riuscita a convergere.

Se, nel frattempo, dall'interfaccia grafica viene richiesto un nuovo colore, appena finita la retroazione il programma riparte con la nuova terna di valori; se il nuovo colore ricade al di fuori del bordo dello spazio colore, si attiva l'*allarme bordo* e il programma si mette in pausa, ma intanto riutilizza i valori del ciclo precedente (o meglio, l'ultima terna di valori che identificasse un punto interno al bordo dello spazio colore).

Allo stesso modo, se l'utente non dà nessun nuovo input, il programma attende un certo tempo (al momento 5 secondi) e poi riutilizza i valori del ciclo precedente. Dato che per questi valori era già stata trovata la terna “corretta” che forniva correnti reali esattamente uguali a quelle richieste, viene saltata tutta la parte delle trasformazioni matematiche, e il valore che era stato memorizzato viene direttamente riutilizzato.
In questo modo, dopo aver acquisito le correnti di alimentazione dei LED, la retroazione effettua una prima verifica e, solitamente, in un ciclo le correnti reali rientrano nella tolleranza richiesta.

Capitolo 3 - Software

La parte software del progetto è stata implementata completamente nel linguaggio di programmazione Matlab, e più precisamente con la versione r2007b.
Solo nell'ultima parte del lavoro di programmazione si è passati alla versione r2008a, poiché per utilizzare la scheda di interfaccia NI USB-6259 era necessario usare il Data Acquisition Toolbox versione 2.12 (nella r2007b era presente la versione 2.11).

Inizialmente era stata valutata l'ipotesi di utilizzare anche il software di programmazione LabView della National Instruments, dato che si trattava della stessa casa delle schede DAQ utilizzate per l'interfaccia software-hardware. LabView è un programma molto potente, in cui la programmazione viene operata attraverso un'interfaccia grafica abbastanza intuitiva, poiché ogni funzione è rappresentata da un blocco, e le varie funzioni sono collegate in un ideale diagramma di flusso. Il suo difetto principale, almeno per la nostra applicazione, è quello di essere orientato principalmente all'acquisizione ed analisi di dati, ed al controllo di processi, e di utilizzare blocchi preimpostati; noi invece avevamo bisogno di uno strumento matematico potente e configurabile dettagliatamente, poiché si sarebbero dovute effettuare delle trasformazioni matematiche per poter passare dai valori immessi dall'utente alle grandezze fisiche erogate (leggi: corrente di alimentazione dei LED).

Per questo motivo la nostra scelta è caduta su Matlab, anche grazie al fatto che è un programma molto utilizzato nell'ambiente universitario: in questo modo è anche stato possibile riutilizzare programmi, o parti di programma, sviluppati dai tesisti che avevano studiato in precedenza questo argomento.

3.1 l'interfaccia grafica

La scelta di utilizzare Matlab ha comportato qualche difficoltà quando si è trattato di programmare un'interfaccia grafica, necessaria nell'ottica in cui la sorgente venga utilizzata da persone che non conoscono questo linguaggio di programmazione. Queste difficoltà erano dovute soprattutto al fatto che la programmazione di interfacce grafiche non era mai stata trattata nel corso della carriera universitaria; anche per professori e assistenti, questo era un aspetto ancora nuovo, per cui ci si è dovuti affidare quasi completamente all' *help* di Matlab e alle discussioni riportate sul sito internet della MathWorks.

Trascurando tutti i passaggi che hanno portato alla definizione dell'attuale interfaccia, si descrive di seguito il suo funzionamento, anche a favore di chi in futuro si troverà ad utilizzarla.

L'interfaccia grafica è costituita da quattro gruppi di elementi base:

- una finestra sulla sinistra, in cui viene rappresentato lo spazio colore in cui si sta lavorando ed il punto scelto;
- un menù a tendina, tre caselle e tre *sliders*, sulla destra, che consentono di scegliere in che spazio colore lavorare e le coordinate del colore che si desidera sintetizzare;
- dei messaggi di avviso (“allarme bordo” e “allarme saturazione”), posti a sinistra sotto la finestra;
- una casella in cui viene visualizzata l'anteprima del colore selezionato.
- un pulsante “Conferma”, che può anche non essere utilizzato se si usano solamente gli *sliders*.

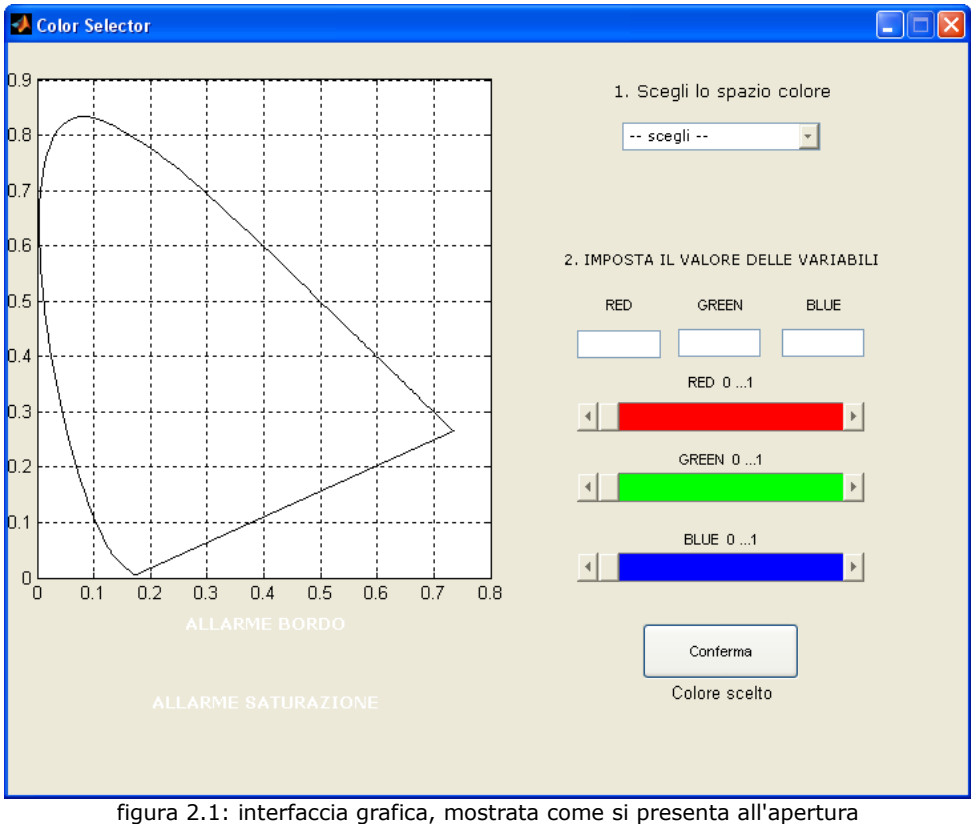


figura 2.1: interfaccia grafica, mostrata come si presenta all'apertura

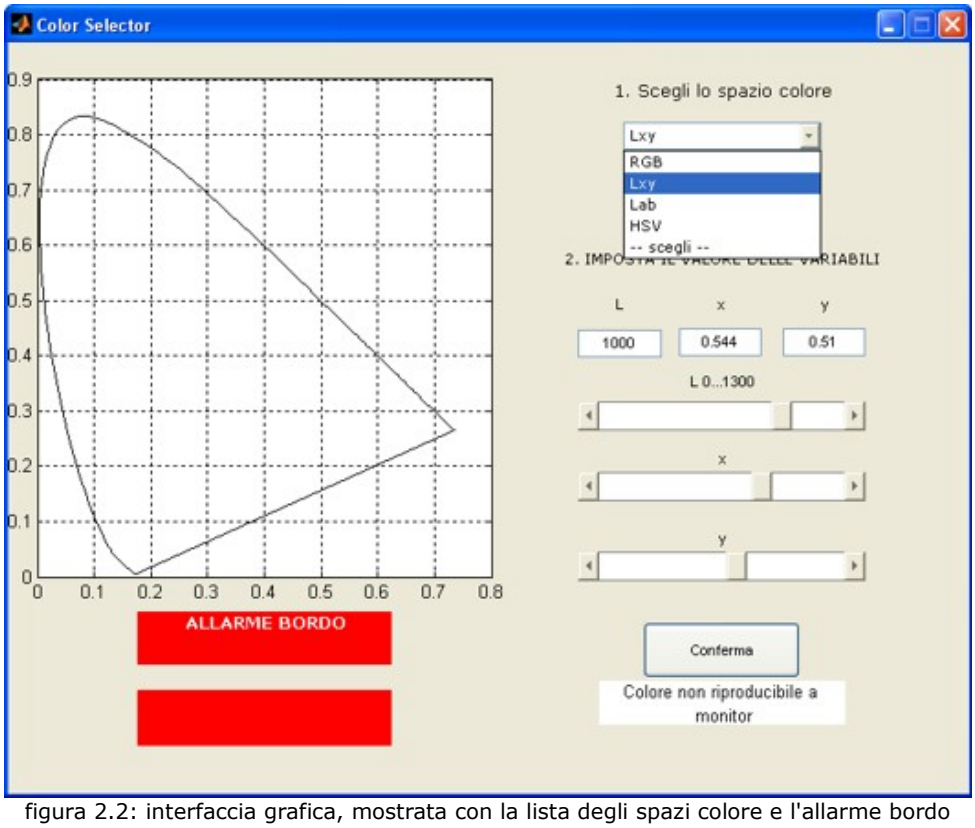


figura 2.2: interfaccia grafica, mostrata con la lista degli spazi colore e l'allarme bordo attivo

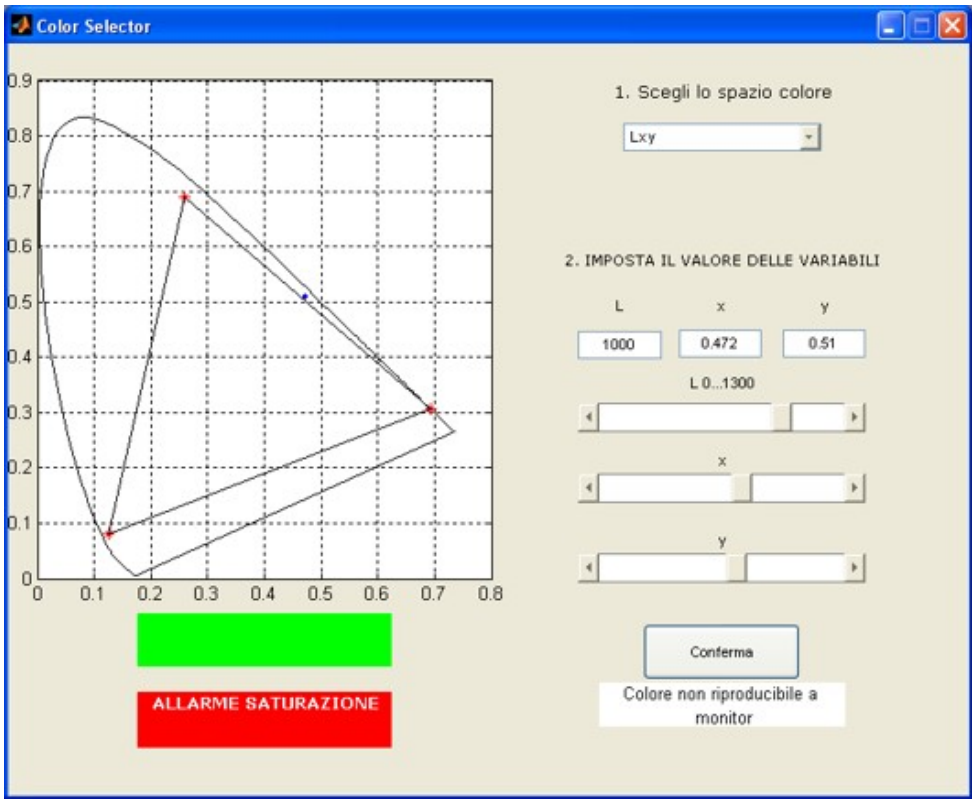


figura 2.3: interfaccia grafica, mostrata con il solo allarme saturazione attivo

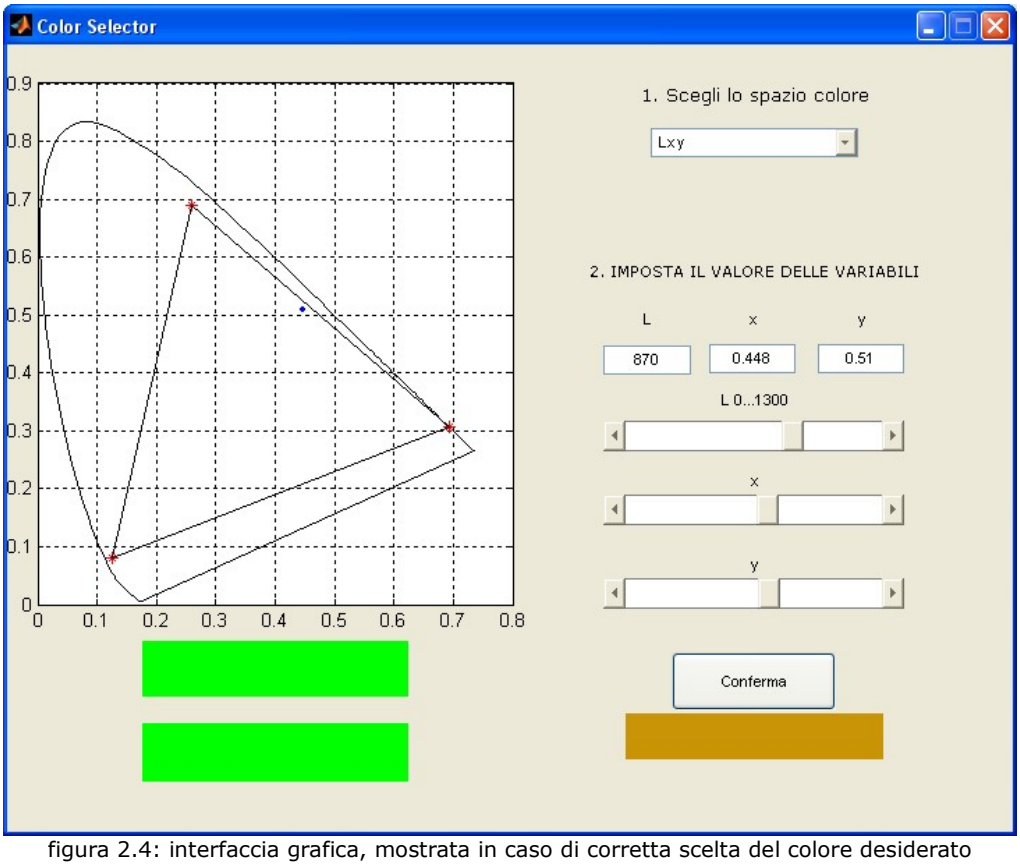


figura 2.4: interfaccia grafica, mostrata in caso di corretta scelta del colore desiderato

L'utente che desideri sintetizzare un colore, deve innanzitutto scegliere in che spazio colore desidera lavorare:

- lo **spazio RGB (1)**, utilizzato per effettuare tutte le prove di funzionamento, è ancora selezionabile, ma si deve tenere conto che utilizzando questo spazio colore, il colore corrispondente ad una certa terna di valori può dare risultati diversi tra l'anteprima e il colore realmente riprodotto con la sorgente a cromaticità variabile: questo perché lo spazio RGB dipende dal dispositivo di riproduzione del colore, e quindi dalle potenze in gioco per i LED dei tre colori, ed utilizzando questo spazio la terna di correnti che viene imposta ai LED non tiene conto della differenza di queste potenze "installate".
  Ad ogni modo, i valori R (Rosso), G (Verde) e B (Blu) possono essere scelti in un intervallo compreso tra 0 e 1, digitando il valore nella casella corrispondente, o trascinando il cursore della barra desiderata.
  La terna [0 0 0] corrisponderà al colore nero, quindi tutti i LED saranno spenti; la terna [1 1 1] non corrisponde al bianco, ma a tutti i LED accesi alla massima potenza (il colore risultante non sarà esattamente bianco ma un violetto molto chiaro, a causa delle diverse potenze luminose in gioco per i tre colori).

- lo **spazio Lxy (2)**, attualmente è quello per cui sono state implementate tutte le funzioni:
  L è la luminanza, che può essere scelta in un intervallo compreso tra 0 e 1300 [cd/m^2];
  x ed y rappresentano, invece, le coordinate cromatiche sul piano Lxy del punto corrispondente al colore scelto. Indicativamente, variando x da 0 al valore massimo, si va dal blu al rosso; variando y da 0 al valore massimo, si va dal blu al verde.

- Lo **spazio Lab (3)**, è lo spazio in cui, una volta che saranno implementate tutte le funzioni, sarà più vantaggioso lavorare: questo perché lo spazio Lab ha la caratteristica di essere quello con maggior uniformità percettiva, come spiegato nella parte di teoria.
  In questo spazio, L rappresenta la luminanza riferita alla luminanza dello sfondo (variabile tra 0 e 100), ovvero la luminosità del colore espressa come una percentuale della luminanza dello sfondo; a varia da -150 (verde) a +150(rosso); b varia da -150 (blu) a +150 (giallo).

- Lo **spazio HSV (4)**, come spiegato in precedenza, è uno spazio non particolarmente uniforme, ma molto intuitivo nella scelta dei colori. In questo spazio, variando l'angolo H da 0 a 360 si percorre in senso orario il "cerchio delle tinte"; variando S da 0 ad 1, si aumenta la saturazione del colore (una bassa saturazione equivale

ad un colore “poco carico”); variando V da 0 ad 1 si aumenta la luminosità del colore.
Lo spazio HSV è stato solo previsto nel menu a tendina, ma non sono ancora stati sviluppati gli aspetti ad esso relativi, per mancanza di tempo e perché sarebbe preferibile utilizzare lo spazio Lch piuttosto che HSV.

Quando si seleziona un qualsiasi spazio colore, viene rappresentato, nella finestra di sinistra, il bordo dello spazio Lxy; una volta scelte le coordinate del colore, viene visualizzato il punto scelto.

Idealmente, la cosa migliore da fare sarebbe visualizzare, per ogni spazio colore selezionato, la rappresentazione grafica dello spazio stesso; questo però avrebbe richiesto un notevole dispendio di tempo, ed essendo l'aspetto di importanza secondaria, è stato tralasciato per poter lavorare su particolari più importanti.

Quindi, al momento, in qualsiasi spazio colore si lavori, il punto scelto viene visualizzato nello spazio Lxy. Questo, comunque, può risultare vantaggioso dal punto di vista della intuitività: infatti, essendo già stato disegnato in Lxy il triangolo del gamut RGB riproducibile con i LED della sorgente, si può sempre avere un'idea di quanto siamo vicini al limite dei LED.

L'allarme bordo

Questo controllo viene effettuato all'interno della function *verifica_inpolygon.m*, al quale vengono mandati i tre valori selezionati, che vengono trattati in modo opportuno a seconda dello spazio colore in cui si sta lavorando.

Scegliendo i valori nello spazio Lxy, il valore massimo selezionabile per x e y è stato scelto pari a 0,8 e 0,9 rispettivamente; si deve però notare che tutti i punti che stanno al di fuori del bordo di Lxy rappresentano dei colori non spettrali, che non è quindi possibile riprodurre: per questo, è stato implementato un "allarme bordo", che avvisa quando la coppia di valori (x,y) non è compresa all'interno del bordo di Lxy.

Quando il punto scelto ricade al di fuori del bordo di Lxy e si attiva l'allarme bordo, il punto non viene visualizzato nel grafico e il programma resta in attesa dell'immissione di un valore realizzabile; se nel ciclo precedente era stato scelto un valore interno al bordo di Lxy, quest'ultimo continua ad essere utilizzato fino a quando non viene inserito un nuovo valore realizzabile.

Lo stesso tipo di controllo si dovrebbe implementare per lo spazio Lab, ma al momento bisogna ancora inserire un codice che fornisca la curva del bordo per il valore di L scelto; quindi, attualmente, lavorando in Lab il controllo del bordo è stato disabilitato.

Data la forma dello spazio HSV, che è rappresentabile con una specie di cono (o come una piramide a 6 facce rovesciata), si capisce che, per basse luminosità del colore, la massima saturazione raggiungibile dal colore è minore di quella che si può ottenere con la massima luminosità. Dovrà quindi essere previsto un controllo che limiti il valore di saturazione in base alla luminosità, o almeno un controllo che indichi se si sta uscendo dallo spazio dei colori reali, analogamente agli spazi appena citati.

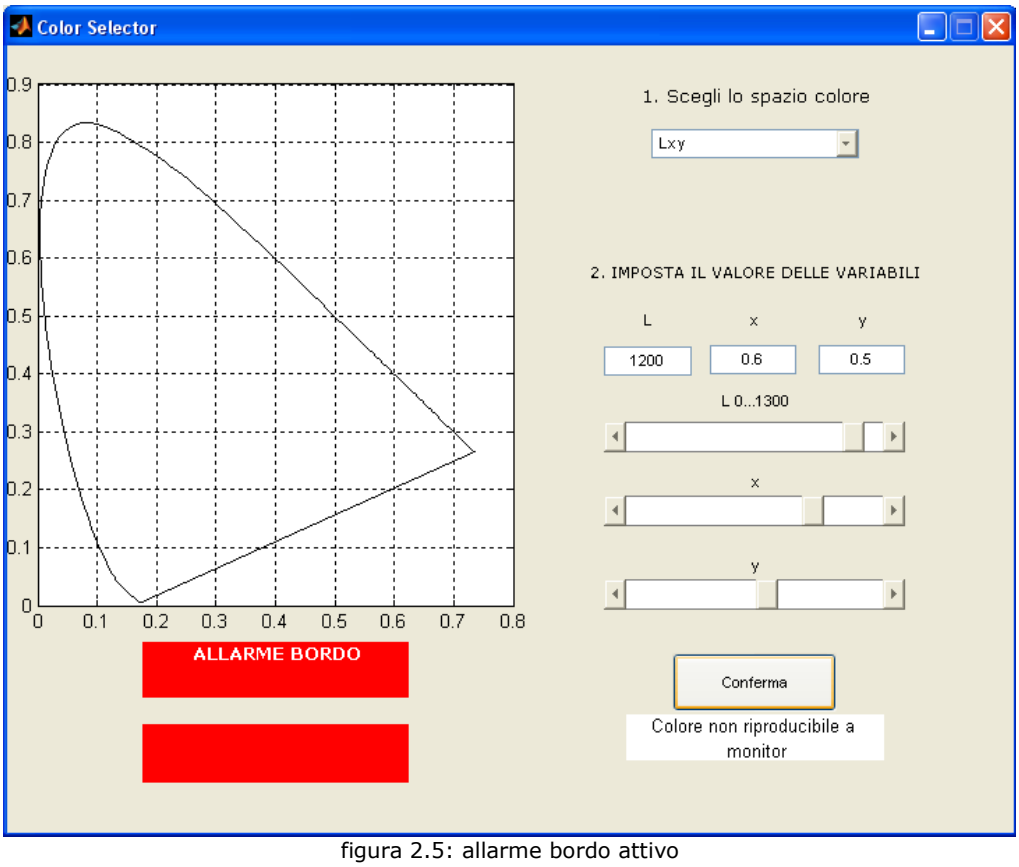


figura 2.5: allarme bordo attivo

L'allarme saturazione

Guardiamo la rappresentazione dello spazio Lxy, con il triangolo RGB al suo interno.

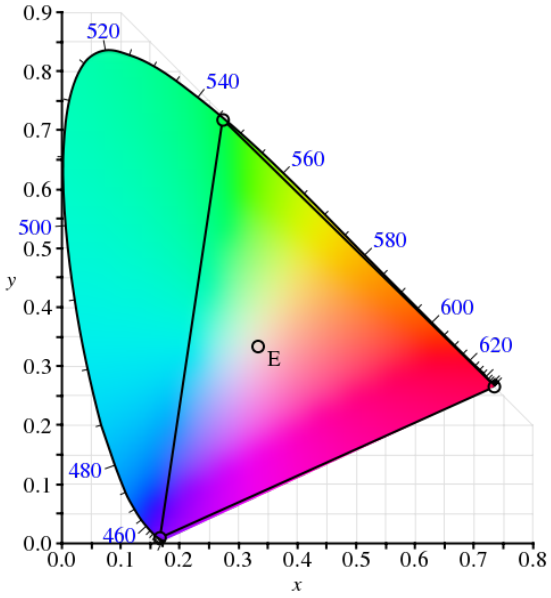


figura 2.6: spazio Lxy e triangolo RGB

Dato che lo spazio Lxy è più ampio del triangolo RGB, ci saranno colori dello spazio Lxy che non si riuscirà a riprodurre esattamente: questo è chiaro vedendo che il triangolo corrispondente al gamut dello spazio RGB è sempre compreso all'interno della curva del luogo spettrale dello spazio Lxy.
Questo è indipendente dal fatto che si usi il gamut RGB riproducibile con i LED, o il gamut di qualsiasi altro spazio colore RGB.

Nella pratica, per rappresentare il colore di un punto posto sul bordo dello spazio Lxy, si può lavorare in due modi:

- "stirare" il triangolo RGB fino al bordo di Lxy: in questo modo, per rappresentare il colore di un punto sul bordo di Lxy si utilizza il colore, con la stessa tinta, che starebbe sul bordo del triangolo RGB (ovvero il colore del punto risultante dall'intersezione tra il segmento che congiunge il punto neutro con il punto sul bordo dello spazio Lxy, con il triangolo RGB);
- mantenere il triangolo RGB inalterato, e al di fuori di questo mantenere il colore costante alla massima saturazione: per riuscire a raggiungere il bordo dello spazio Lxy partendo dal "punto neutro" si modula la saturazione dei colori finché si è all'interno del triangolo RGB, e poi si mantiene il colore alla massima saturazione fino a quando si raggiunge il bordo dello spazio Lxy.

In questo progetto, si è preferito seguire la seconda strada, in modo che sia anche evidente dove sopraggiunge il limite dell'hardware.

Questo comporta che, se il colore scelto ricade al di fuori del triangolo RGB, non verrà riprodotto esattamente, poiché le varie componenti non possono avere valore maggiore di 1 per definizione.
Per avvisare l'utente di questo fatto, è stato inserito l' "Allarme saturazione", che si attiva quando nella terna di valori uno o più elementi ha valore al di fuori del range [0- 1]: la terna di valori viene ugualmente mandata in uscita, dopo essere stata limitata nell'intervallo [0-1], e grazie all'allarme l'utente sa che il colore che vede potrebbe discostarsi anche in maniera significativa da quello desiderato.

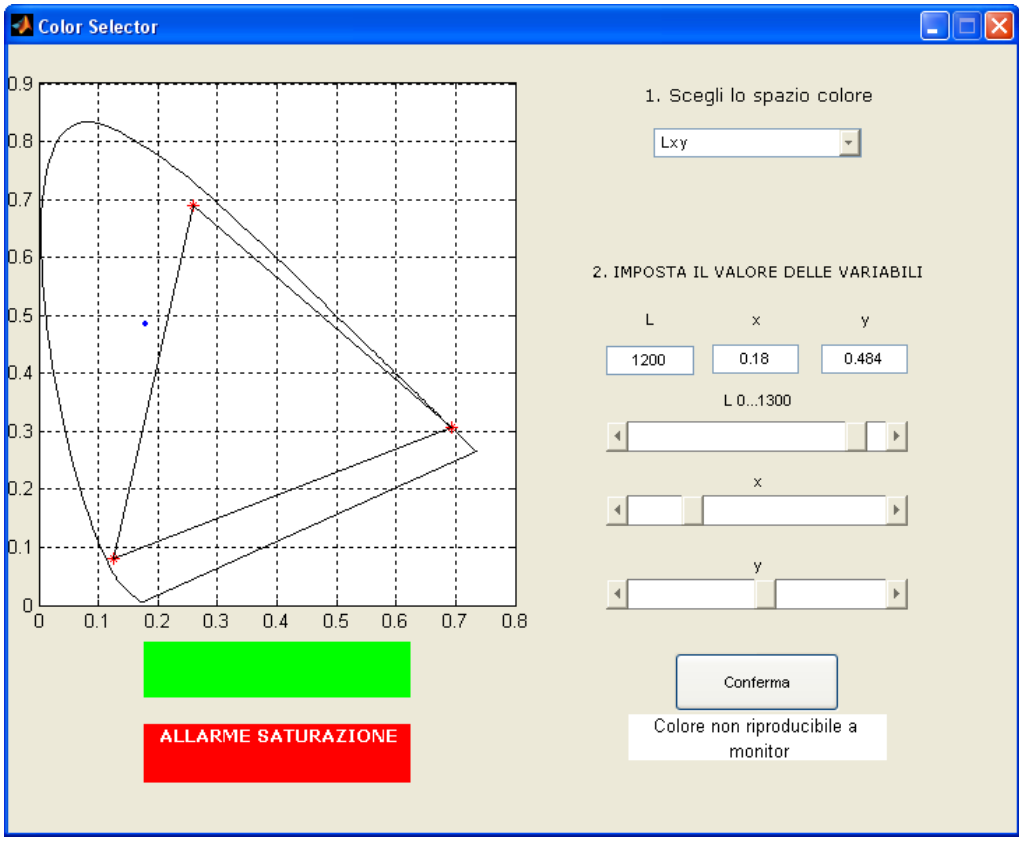


figura 2.7: allarme saturazione attivo

La presenza di uno o più valori non compresi tra 0 ed 1, nella terna risultante dalla trasformazione matematica operata sul colore scelto dell'utente, oltre ad attivare l'allarme saturazione inibisce anche la rappresentazione dell'anteprima del colore: infatti, quest'ultima è rappresentata nello spazio colore sRGB, che altro non è che una correzione esponenziale dello spazio RGB.In verità si tratta di uno spazio RGB diverso da quello dei LED, quindi, potrebbe capitare il caso che il colore sia rappresentabile sul monitor, ma non con il LED

Poiché questo controllo viene effettuato sui valori risultanti dalle trasformazioni matematiche, operate sui tre valori scelti dall'utente per sintetizzare un colore, i concetti appena esposti si possono estendere anche allo spazio Lab.

L'anteprima del colore

Per trasformare la terna di valori dallo spazio colore scelto allo spazio RGB dei LED, si passa attraverso lo spazio XYZ. Contemporaneamente alla trasformazione in RGB, quindi, viene anche effettuata una trasformazione matematica da XYZ a sRGB dei valori scelti.

Questa trasformazione serve per fornire un'anteprima a monitor del colore selezionato, nella casella sottostante alle tre *sliders* nell'interfaccia grafica.

Bisogna qui fare un piccolo appunto: i LED del monitor lavorano in modo diverso dai LED della sorgente, Infatti, sul monitor, per ottenere la luminanza massima, si deve creare un bianco accendendo al massimo i LED R,G e B; nella sorgente, invece, i LED sono stati dimensionati in modo da poter mantenere la luminanza massima per qualsiasi colore.

Quindi, ragionando in tre dimensioni, il triangolo RGB dei LED della sorgente è un prisma con base triangolare; i LED RGB del monitor, invece, formano un cubo, con il vertice superiore che ha le coordinate cromatiche del bianco.

Questo ragionamento è necessario per spiegare che l'anteprima a monitor del colore non può coprire tutti i colori riproducibili con la sorgente: infatti, avendo fatto corrispondere il bianco a luminanza massima della sorgente con il bianco a luminanza massima del monitor, quello sarà l'unico colore a luminanza massima riproducibile dall'anteprima. Man mano che si va verso colori più saturi, la luminanza per cui sarà disponibile l'anteprima diminuirà; se si vuole riprodurre un rosso saturo alla massima luminanza, per esempio, la sorgente sarà in grado di farlo (e l'allarme saturazione non segnalerà problemi, giustamente), ma l'anteprima non sarà disponibile poiché i valori della terna sRGB

andranno al di fuori del range [0-1].
In tali situazioni, al posto dell'anteprima del colore compare la scritta “Colore non riproducibile a monitor”.

L'unico colore che può essere riprodotto con la luminanza massima è il bianco corrispondente al punto bianco del monitor. In realtà, impostando L=1300 i valori sRGB risultano di poco superiori ad 1, ma tanto basta per inibire l'anteprima: probabilmente questo è causato dalla mancanza di qualche cifra significativa in più nelle coordinate cromatiche.
Come mostrato in figura 2.8, il massimo valore di luminanza per cui si ottiene l’anteprima, con le coordinate cromatiche del punto bianco del monitor (x=0,3127 y=0,3290), è L=1299,87.

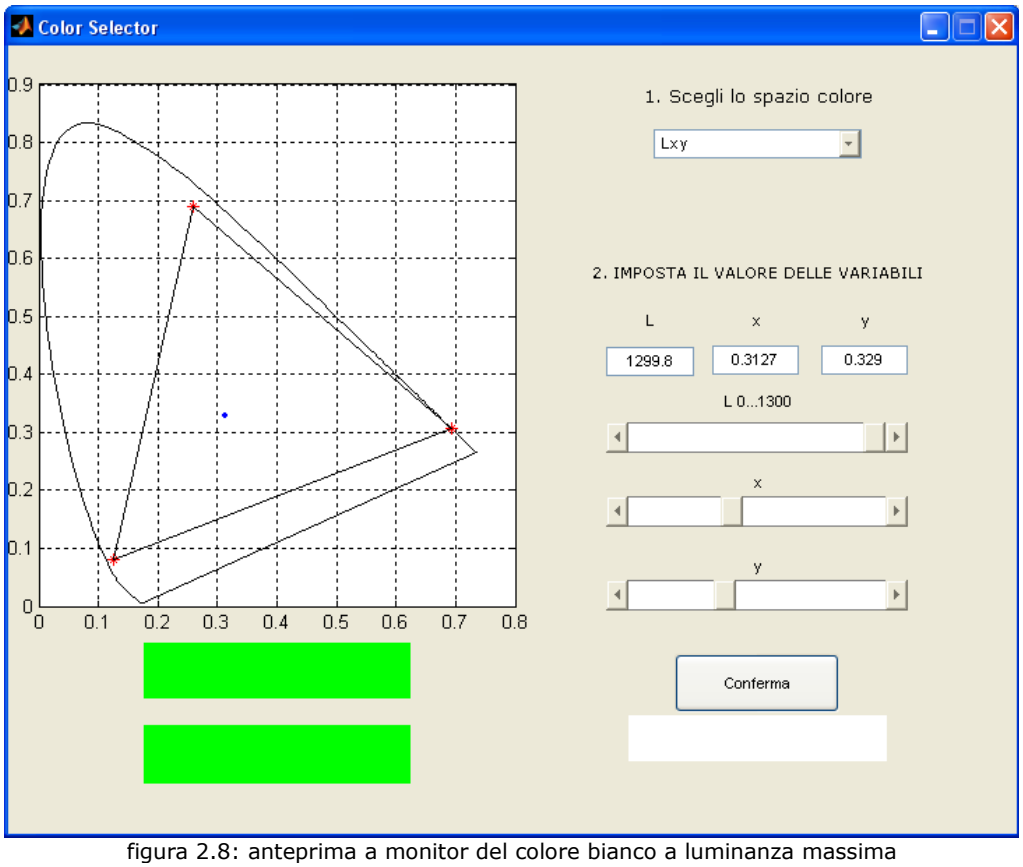


figura 2.8: anteprima a monitor del colore bianco a luminanza massima

Nelle figure 2.9 e 2.10, si mostra come un colore con certe coordinate cromatiche non possa essere reso in anteprima se la luminanza richiesta è superiore ad un certo valore.

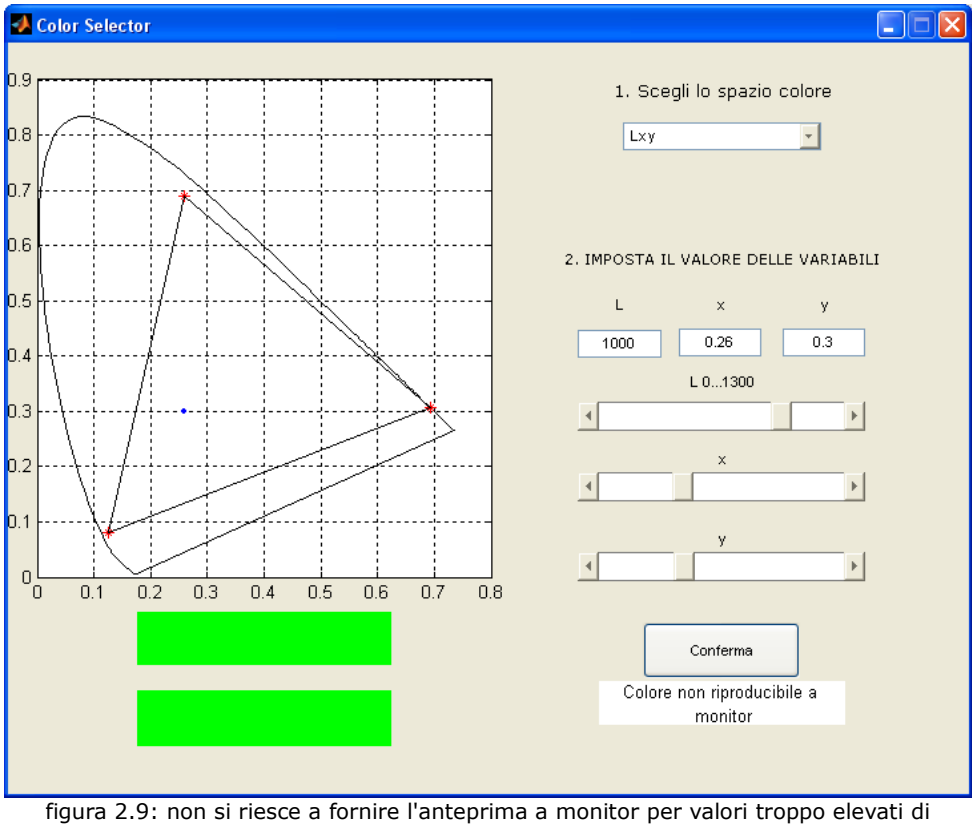


figura 2.9: non si riesce a fornire l'anteprima a monitor per valori troppo elevati di luminanza

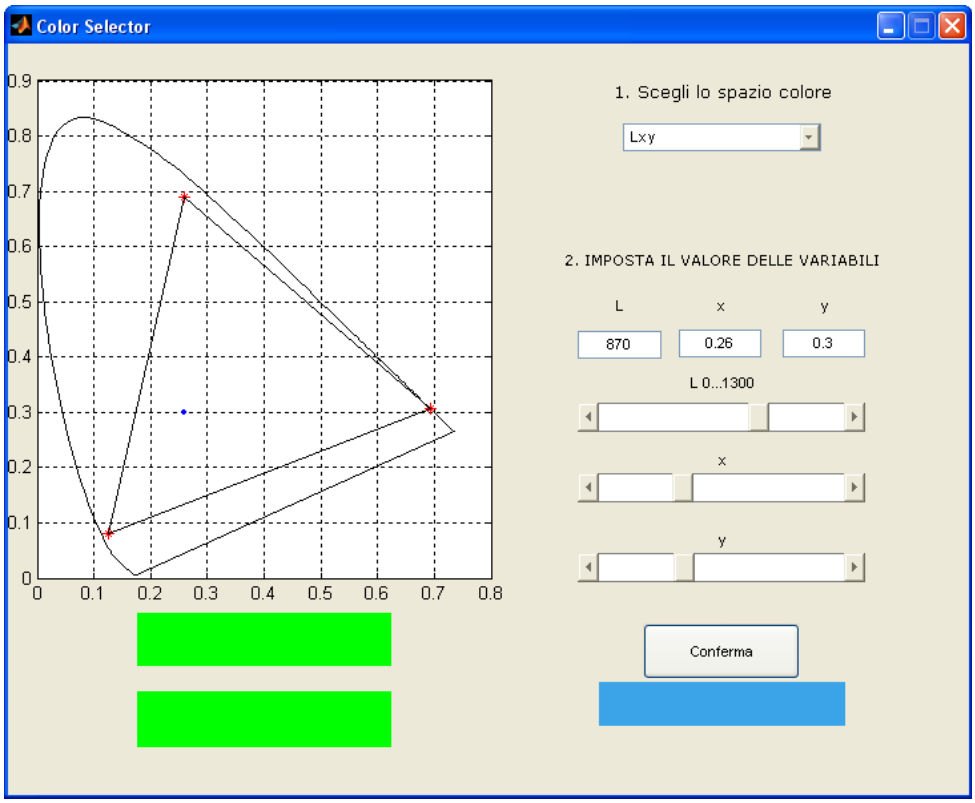


figura 2.10: per luminanze inferiori, si riesce ad ottenere l'anteprima a monitor

Note per l'utilizzo dell'interfaccia grafica

Il programma che genera e acquisisce valori dall'interfaccia grafica attende che siano stati immessi tutti e tre i valori, prima di procedere alla trasformazione matematica e alla sintesi del colore: per fare ciò, il programma attende che esistano contemporaneamente le tre variabili rappresentative dei tre valori scelti.
Se si immettono i valori tramite le caselle di testo, è necessario dare un ulteriore input dopo aver digitato l'ultimo valore (Invio, Tab, click con il mouse, pressione del tasto “Conferma”,...) per dare modo al programma di capire che è terminata la digitazione del valore, e che quindi può acquisire il valore digitato per creare anche la terza variabile.

Quando si sta lavorando in un certo spazio colore, e si decide di cambiare spazio colore, tutti i valori precedentemente impostati vengono azzerati, e il programma messo in pausa. Purtroppo non è possibile eliminare fisicamente le variabili, ma solo azzerarle: per questo motivo, già quando si seleziona il primo dei tre valori nel nuovo spazio colore, il programma esce dalla pausa e la terna viene inviata all'output.
Riuscendo a trovare il modo di eliminare le tre variabili ogni volta che si cambia spazio colore (purtroppo il metodo *delete(handle)* non sembra funzionare, probabilmente perché funziona solo con gli oggetti grafici) si risolverebbe anche questa imperfezione del programma.

Poiché è importante che il colore scelto sia, nei limiti del possibile, riprodotto fedelmente, è stata prevista una retroazione che porta i valori dei flussi luminosi dei tre canali ai livelli corretti per sintetizzare quel colore. Durante questo processo di retroazione (di cui si parlerà più avanti) vengono inviate delle tensioni di comando agli alimentatori, corrette in base all'errore misurato, e vengono misurate le correnti di alimentazione dei LED: per questo, la luminosità e le coordinate cromatiche variano ciclicamente, finché non si raggiunge il valore desiderato.

Per fare in modo che l'osservatore sappia quando è stato raggiunto il colore selezionato nell'interfaccia grafica, si è previsto di far comparire un messaggio a video nel momento in cui il colore selezionato è stato raggiunto e il processo di retroazione termina.

Bisogna attendere che sia finito il ciclo di retroazione, prima di impostare un nuovo valore: in caso contrario, l'input potrebbe essere ignorato dal programma.

Per avere un controllo continuo sui valori di corrente effettivamente circolanti nei LED, in mancanza di nuovi input da parte dell'utente, il programma che comanda l'interfaccia grafica (ColorSelector.m) restituisce il controllo al programma principale (main.m) dopo 5 secondi.

Questo tempo può essere modificato, immettendo il numero di secondi desiderato alla riga 73 del programma ColorSelector.m:

```
uiwait(handles.figure1,5);      %attendo il comando uiresume, che la finestra
                                venga chiusa o che passino 5 secondi
```

Il valore attuale di 5 secondi serve più che altro per le prove di funzionamento, in modo da rendere il programma veloce nell'effettuare i cicli; quando la sorgente dovrà essere utilizzata per esperimenti e prove di laboratorio, si consiglia di adeguare l'intervallo in cui la retroazione non agisce al tempo necessario per l'osservazione.

L'interfaccia grafica resta attiva fino a quando non viene chiusa dal pulsante con la X in alto a destra; in quel caso il programma principale smette di essere eseguito e viene inviata una terna di valori che fa spegnere i LED.

3.2 le trasformazioni negli spazi colore

Una volta che, tramite l'interfaccia grafica, sono stati scelti i tre valori relativi al colore che si desidera sintetizzare[3], interviene il programma che consente di trasformare la terna di valori che identifica quel colore in una terna di valori di corrente da applicare ai LED, per ottenere il colore desiderato.
Contemporaneamente a questa trasformazione, viene anche effettuata la trasformazione nello spazio colore sRGB, per poter dare un'anteprima a monitor del colore scelto.
Infine, viene anche effettuata una trasformazione grafica nello spazio Lxy, per poter visualizzare la posizione del punto scelto.

La function che esegue queste operazioni è chiamata Trasf_2_RGBled.

Come si è detto prima, dall'interfaccia grafica la terna dei valori può essere scelta negli spazi RGB, Lxy, Lab e HSV (questi ultimi non ancora pienamente supportati).

Per ognuno di questi spazi colore, sono definite delle trasformazioni matematiche (viste in precedenza), il più delle volte non lineari, che consentono di mappare un vettore (la terna di valori) da uno spazio colore all'altro.

Lo scopo del programma è quello di trasformare la terna scelta in una terna di valori X,Y,Z , i quali vengono a loro volta trasformati con una trasformazione lineare in una terna di valori nello spazio RGB, ovvero nella terna delle correnti di alimentazione dei LED di ogni canale.

Ovviamente, dovendo passare da uno spazio colore assoluto come XYZ, che non tiene conto del dispositivo di riproduzione, ad uno spazio colore come RGB, in cui questa dipendenza invece esiste, bisogna introdurre le funzioni tristimolo generate dal dispositivo di riproduzione.
Nel nostro caso in particolare, ad un certo punto del programma vengono lette le radianze spettrali dei LED (nell'ordine: rosso, verde, blu) fatti funzionare uno alla volta alla massima potenza. Da queste radianze spettrali, vengono calcolati i valori tristimolo in X,Y,Z prodotti dai LED.
La relazione tra i valori R,G,B e i valori tristimolo X,Y,Z da essi prodotti può essere quindi rappresentata con una matrice che, dopo essere stata invertita, permetterà di calcolare i valori di R,G e B necessari per produrre un certo tristimolo X,Y e Z.

3.2.1 trasformazione da RGB

Nonostante lo spazio RGB non sia molto utile per l'utilizzo della sorgente, si è deciso di lasciarlo, perché a volte può tornare comodo per effettuare delle prove di funzionamento.

In effetti, partendo da RGB e dovendo dare in uscita una terna in RGB, non è necessaria alcuna trasformazione matematica: questo può risultare comodo se si vuole escludere che qualche problema di funzionamento possa essere dovuto alle trasformazioni matematiche.

Le uniche conversioni che vengono effettuate dallo spazio RGB, quindi, sono quelle necessarie per le anteprime a monitor.
Prima di tutto, la terna RGB viene trasformata in XYZ: questa trasformazione, però, non utilizza la matrice ottenuta con i valori tristimolo della sorgente, bensì quella che si basa sulla trasformazione teorica da RGB a XYZ.

Successivamente, i valori in XYZ vengono convertiti in sRGB per l'anteprima del colore, e in Lxy per rappresentare il punto scelto nella finestra grafica di sinistra.

3.2.2 trasformazione da Lxy

Scegliendo di lavorare nello spazio Lxy, la luminanza L può essere variata tra 0 e il massimo (1300); i valori x e y, invece possono variare tra 0 e 1, ma non sempre saranno all'interno del bordo dello spazio Lxy.

Quando i valori x,y sono all'esterno del bordo, significa che si sta cercando di riprodurre un colore non spettrale, cosa che non è possibile: quindi, in tale caso, il programma resta in attesa dell'immissione di una coppia di valori che si trovi all'interno del bordo.

3 Come spiegato in precedenza, se i valori sono stati scelti nello spazio Lxy, come prima cosa viene effettuata la verifica se questi ricadono all'interno del bordo dello spazio: se la verifica dà esito positivo, il programma prosegue, come si sta giust'appunto per spiegare.

Quando i valori x,y sono all'interno del bordo, all'interno della function Trasf_2_RGBled, viene prima effettuata una trasformazione da Lxy in XYZ utilizzando la function CieLxy2CieXYZ.
La terna di valori in XYZ ottenuta viene convertita quindi, tramite la function CieXYZ2RGB_led, nella terna di valori in RGB da mandare all'output.

In Lxy si ritorna per la rappresentazione grafica dello spazio, del triangolo rappresentante il gamut RGB riproducibile, e del punto scelto.
Quest'ultimo è direttamente ottenuto dal secondo e dal terzo valore costituenti la terna Lxy; il bordo dello spazio Lxy è ottenuto tramite la trasformazione da XYZ a Lxy dei valori tristimolo relativi alla sensibilità dell'occhio umano; il triangolo RGB, invece, è ottenuto dalla trasformazione in Lxy dei valori tristimolo in XYZ generati dai LED.

Partendo dalla terna in XYZ, come nel caso precedente, viene effettuata la trasformazione in sRGB per l'anteprima a monitor del colore scelto.

Data la particolare matrice utilizzata per la conversione da XYZ a sRGB, il punto bianco (che è quello a luminanza massima per cui si può dare un'anteprima) ha coordinate cromatiche x=0,3127 y=0,3290 e luminanza L=1299,8.
Per L=1300 i valori vengono leggermente superiori a 1 e quindi l'anteprima non viene visualizzata.

3.2.3 trasformazione da Lab

Poiché in Lab le coordinate sono ottenute confrontando il tristimolo del colore in esame (X,Y,Z) con quello del perfetto diffusore (Xn,Yn=100,Zn), per effettuare la trasformazione da XYZ a Lab è necessario conoscere queste Xn e Zn.

I valori di Xn e Zn dipendono dal tipo di sorgente di riferimento che si sceglie di utilizzare; nel programma Trasf_2_RGBled si può scegliere di utilizzare l'illuminante A (rappresentativa di una lampada ad incandescenza) o l'illuminante D65 (che rappresenta uno spettro simile allo spettro solare diurno, con temperatura del bianco di 6500K, simile a quella del monitor del computer).

Al momento la sorgente di riferimento è la D65, ma questa può essere cambiata, in base alla lampada che verrà utilizzata per illuminare il campo d'esame (quando la sorgente verrà utilizzata nella “cameretta”).

Nel programma, utilizzando quindi lo spettro della sorgente D65, le coordinate del punto neutro in Lab (100,0,0) vengono trasformate nelle corrispondenti coordinate (Xn,Yn,Zn) in XYZ; utilizzando queste ultime, viene effettuata la trasformazione dalla terna di valori scelti in Lab alla corrispondente terna in XYZ.

Successivamente, la terna XYZ viene convertita in RGB per il comando della scheda DAQ, e in sRGB per l'anteprima a monitor del colore.

Dal punto di vista grafico, il punto scelto in Lab viene convertito in XYZ e quindi in Lxy, per poterlo rappresentare nello spazio Lxy, che è più intuitivo.

3.2.3 trasformazione da HSV

Al momento attuale, questa trasformazione non è ancora stata implementata, avendo finora lavorato esclusivamente in RGB, in Lxy e in Lab.

Soprattutto, sembra che la conversione da HSV sia di difficile implementazione in Matlab, poiché essa richiede una funzione di modulo aritmetico, di cui non si trova traccia nemmeno nell'*help* di prodotto.

Probabilmente esiste un metodo più semplice per implementare questa trasformazione, forse usando degli *if*, ma data la mancanza di tempo si lascia da completare questo aspetto.

Nota:

Per maggiori chiarimenti sulle trasformazioni operate tra i vari spazi colore, si consiglia di leggere i listati delle routines in appendice, e il capitolo 1.4.3 nella parte di teoria, che tratta appunto le trasformazioni matematiche negli spazi colore.

3.3 riassunto delle function utilizzate e della loro funzione

main_6259.m: programma principale per utilizzare la sorgente utilizzando la scheda NI USB-6259;

Le altre *functions* vengono elencate in ordine alfabetico:

- **ApriPorte_6259**: apre le porte di comunicazione della scheda NI USB-6259 e dello spettroradiometro; attualmente, le righe che aprono la porta di comunicazione dello spettroradiometro sono commentate;
- **ChiudiPorte_6259**: chiude le porte di comunicazione della scheda NI USB-6259 e dello spettroradiometro; attualmente, le righe che chiudono la porta di comunicazione dello spettroradiometro sono commentate;
- **CieLab2CieXYZ**: effettua la trasformazione dallo spazio Lab allo spazio XYZ;
- **CieLxy2CieXYZ**: effettua la trasformazione dallo spazio Lxy allo spazio XYZ;
- **CieXYZ2CieLab**: effettua la trasformazione dallo spazio XYZ allo spazio Lab;
- **CieXYZ2CieLxy**: effettua la trasformazione dallo spazio XYZ allo spazio Lxy;
- **CieXYZ2RGB_led**: effettua la trasformazione dallo spazio XYZ allo spazio RGB dei LED, per dare i valori dei flussi da richiedere ai LED;
- **CieXYZ2sRGB**: effettua la trasformazione dallo spazio XYZ allo spazio sRGB, per fornire i valori per l'anteprima a monitor del colore scelto;
- **ColorSelector**: è il programma che comanda l'interfaccia grafica: esso viene interpellato ad ogni ciclo del *main.m*, e fornisce in uscita la terna di valori in RGB_LED (valori dei flussi da richiedere) e la variabile Fine, che vale 0 solo quando si chiude l'interfaccia; se entro 5 secondi (valore modificabile) non riceve un nuovo input, rimanda fuori la terna del ciclo precedente;
- **ComandoSchedaDAQ_6259**: è il programma che, partendo dai valori di corrente richiesti, manda alle uscite analogiche della scheda NI USB-6259 i valori delle tensioni di controllo degli alimentatori;
- **creaSpazioLxy**: disegna il bordo dello spazio Lxy appena si sceglie uno spazio colore nell'interfaccia grafica;
- **leggi_x**: script che legge il file *col_mat_fun_x* e calcola la sensibilità spettrale dell'occhio umano rispetto alla Color Matching Function "x";
- **leggi_y**, **leggi_z**: come leggi_x, ma usando i file *col_mat_fun_y* e *col_mat_fun_z* rispettivamente;
- **leggispettro**: carica i file in uscita dallo spettro radiometro Minolta CS-1000 e converte lo spettro in un file di testo;
- **LetturaCorrentiTempProlungata_6259**: acquisisce i valori delle correnti e la tensione relativa al trasduttore di temperatura dalla scheda NI USB-6259; il termine "Prolungata" sta ad indicare che vengono acquisiti 1000 campioni, di cui poi si fa la media; in questo *script* viene anche calcolata la temperatura reale, partendo dalla tensione corrispondente; de-commentando l'ultima riga (*VisualizzaValoriLetti*) si richiama uno *script* che mostra a monitor i valori acquisiti e la temperatura calcolata;
- **Lookup_tables**: riceve in ingresso la terna dei flussi richiesti, e interpolando le *look-up tables*, ne ricava i valori di corrente da imporre ai LED;
- **RetroazionePI_6259**: *function* che effettua la retroazione, prendendo come *set-point* i valori di corrente richiesti e continuando a ciclare finché le correnti misurate non sono uguali al *set-point* per meno di una certa tolleranza; finita la retroazione, la nuova terna di correnti viene memorizzata, per utilizzarla al ciclo successivo se non viene richiesta una nuova terna di flussi;
- **Trasf_2_RGBled**: *function* che riceve in ingresso la terna di valori impostata nell'interfaccia grafica e un numero identificativo dello spazio colore in cui si sta lavorando; tramite vari passaggi, ed utilizzando altre *sub-functions* descritte in precedenza, manda in uscita la terna dei flussi da richiedere (nello spazio RGB) e la terna dei valori in sRGB per l'anteprima a monitor del colore scelto;
- **verifica_inpolygon**: *function* che riceve in ingresso la terna di valori impostata nell'interfaccia grafica e un numero identificativo dello spazio colore in cui si sta lavorando; dopo aver calcolato il bordo dello spazio colore in cui si sta lavorando, determina se il punto scelto ricade all'interno o all'esterno del bordo stesso, e

manda in uscita un valore 0 o 1 (1 se il punto è all'interno) che blocca il ciclo se il valore è esterno al bordo e quindi non è riproducibile; al momento, questa *function* funziona solo con Lxy (in RGB il punto sarà sicuramente interno, mentre per Lab non si ha ancora il vettore dei punti del bordo, quindi al momento per Lab si fa dire al programma che il punto è sempre dentro);

- **VisualizzaValoriLetti**: utilizzando le variabili memorizzate nel *workspace* dallo *script* *LetturaCorrentiTempProlungata_6259*, mostra nella *command window* di Matlab i valori acquisiti e la temperatura calcolata.

Capitolo 4 - Hardware

4.1 progettazione dell'insieme hardware

In linea di principio, dal punto di vista dell'hardware la sorgente doveva essere composta da questi elementi principali:

- un contenitore diffondente, dotato di apertura per l'osservazione;
- un elemento meccanico che permettesse di fissare i LED al contenitore diffondente;
- una sorgente luminosa, composta da LED rossi, verdi e blu, di potenza adeguata per i nostri scopi;
- un circuito di alimentazione dei LED, a corrente costante e regolabile da 0 al 100% per ogni canale separatamente, per poter controllare il flusso emesso per ogni colore.

4.1.1 contenitore diffondente

Il progetto di questa tesi si basa sulla sintesi additiva di colori: partendo dai LED rossi (R), verdi (G) e blu (B), si può teoricamente ottenere qualsiasi colore si desideri.

Era dunque fondamentale, nella progettazione della configurazione hardware dell'insieme, fare in modo che all'osservatore giungesse una luce quanto più possibile uniforme.

Si è fatto quindi riferimento al principio di funzionamento della sfera integratrice, uno strumento utilizzato in fotometria per la misura del flusso luminoso delle sorgenti luminose.

Dato che il costo di una sfera integratrice è molto elevato, si è deciso di utilizzare un “cilindro integratore”, molto più semplice da costruire.
Si è perciò sfruttato un barattolo di latta con un diametro di circa 10 cm, verniciato internamente con uno smalto bianco opaco.

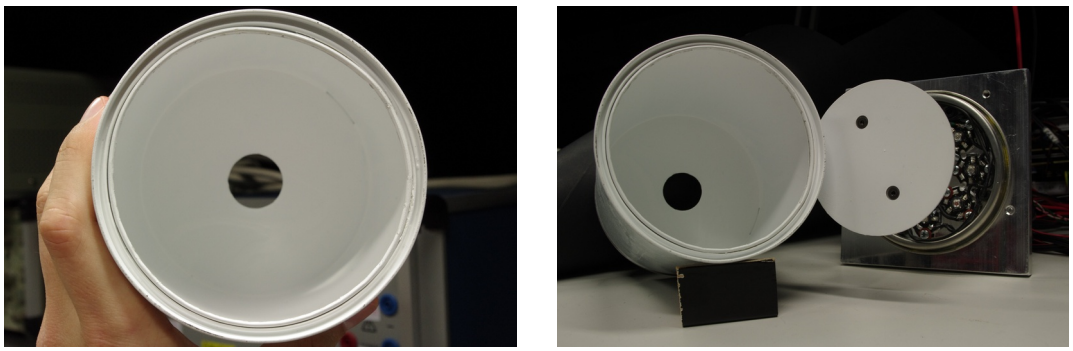


figura 4.1: contenitore diffondente, visto da solo e di fianco alla sorgente

Come nella sfera integratrice, anche in questo caso si necessitava di un setto separatore, che impedisse la visione diretta della sorgente da parte dell'osservatore o dello strumento di misura. Questo setto è stato costruito con dei dischi di plexiglass, verniciati con lo stesso smalto bianco opaco utilizzato per il barattolo e muniti di fori per il fissaggio attraverso delle barre filettate.

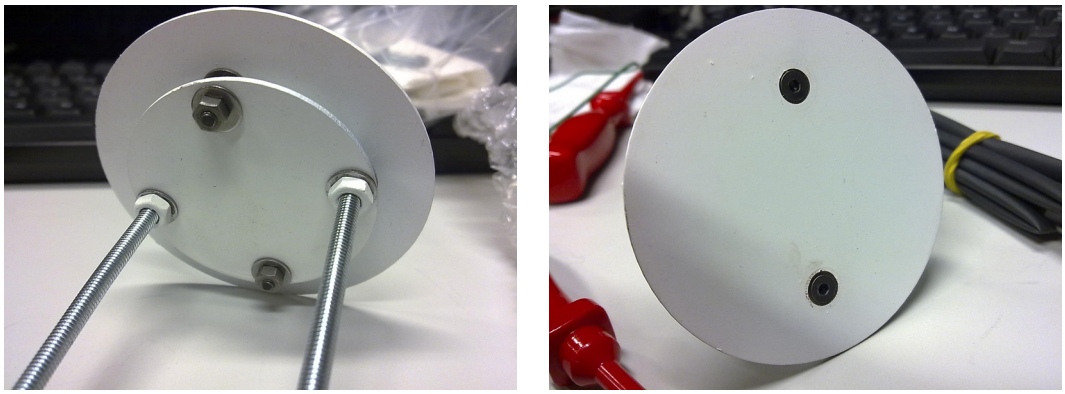


figura 4.2: setto separatore (visto da sotto, ovvero dall'interno, e da sopra)

4.1.2 piastra per montaggio LED

Per l'alloggiamento dei LED è stata costruita una piastra di alluminio, provvista di fori filettati per le viti in nylon di fissaggio delle rosette di dissipazione dei LED, di fori filettati per le barre filettate di sostegno del setto, e di fori per il passaggio dei fili di alimentazione dei LED.

La progettazione della piastra e della disposizione dei vari componenti su di essa è stata fatta con il software di disegno Autocad, dal quale poi si è ricavato direttamente il file per la lavorazione su macchina a controllo numerico.

Dopo la maschiatura manuale dei fori, la piastra è stata lucidata a a specchio, per consentire un migliore scambio termico con le rosette di dissipazione dei LED.

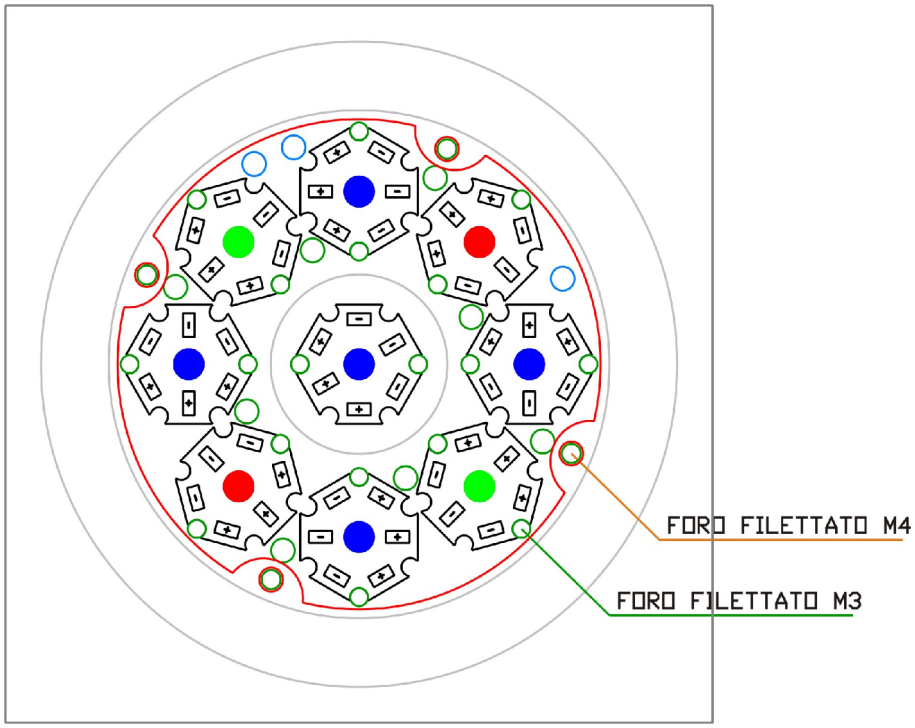


figura 4.3: disegno realizzato in AutoCad per la progettazione della piastra di montaggio dei LED

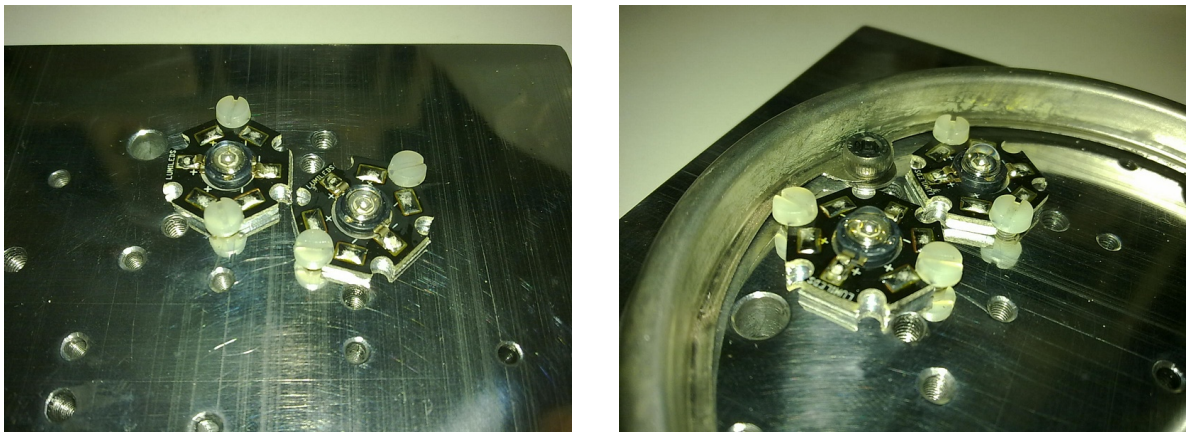


figura 4.4: due led posizionati sulla piastra, a destra con il bordo del coperchio del contenitore, necessario per fissare il contenitore alla piastra

4.1.3 scelta dei LED

Dato che la sorgente verrà successivamente usata in combinazione con la "camera di visualizzazione" (che viene usata per alcuni esperimenti sulla percezione del colore), si è voluto ottenere un livello di illuminamento paragonabile con quello che già si ottiene con le sorgenti installate, ovvero 3000 lux.

Approssimando il comportamento del nostro contenitore a quello di una sfera perfettamente diffondente, ad un illuminamento di 3000 lux corrisponderà una luminanza di

$$L = \frac{E}{\pi} \cdot \rho = 1000\ [cd/m^2]$$

Nella camera di visualizzazione si ottiene una luminanza di 8800 [cd/m^2] con una lampada alogena da 40 W.

Dato che una lampada alogena ha un' efficienza luminosa di circa 13 [lm/W], il suo flusso luminoso è circa di 40x13=520 [lm].

Se noi vogliamo ottenere una luminanza di circa 1000 [cd/m^2] nel barattolo, ci servirà un illuminamento di

520x(1000/8800) = 60 [lm].

Per stare abbondanti, si è deciso di stare sui 100 [lm] per ogni colore, in modo che la luminanza minima si riesca ad ottenere anche usando un solo colore.

Dovendo ottenere dei valori di flusso non irrilevanti per una sorgente a LED, la scelta dei LED è ricaduta quindi sui LUXEON III Star: si tratta di power LED con una potenza di circa 3 W ognuno; essi vengono forniti pre-saldati su una rosetta di materiale conduttore di calore, per facilitare lo smaltimento del calore prodotto dalla giunzione (da qui il suffisso "Star").

Si è deciso di utilizzare questo tipo di LED perché erano già stati utilizzati in precedenti realizzazioni, ed avevano sempre fornito ottimi risultati; inoltre, avendo già in laboratorio altri esemplari dello stesso modello, si è potuto lavorare con la tranquillità di avere a disposizione dei pezzi di scorta.

Per scegliere le proporzioni tra i vari colori, si è deciso di ragionare a parità di flusso per tutti i canali, senza considerare quindi la sensibilità spettrale dell'occhio umano[4].

I LED quindi sono stati installati con le seguenti proporzioni:

- 2 verdi da 55 lm cad. a 700 mA
- 2 rossi da 50 lm cad. a 700 mA
- 5 blu da 20 lm cad. a 700 mA.

In effetti, in seguito alle misure effettuate a sorgente ultimata, si è visto che la luminanza che può essere mantenuta per qualsiasi colore è di oltre 1300 [cd/m^2].

Questa è la piastra in alluminio con i LED montati; il bordo metallico è stato ottenuto intagliando a macchina il coperchio del barattolo, in modo da poter fissare la piastra con i LED al barattolo stesso.

4 Questa scelta è stata fatta nell'ottica di utilizzare la sorgente come calibratore per gli spettroradiometri: in quel caso la sensibilità spettrale dell'occhio umano non conta, conta invece poter avere un flusso minimo garantito per ogni colore si voglia sintetizzare.

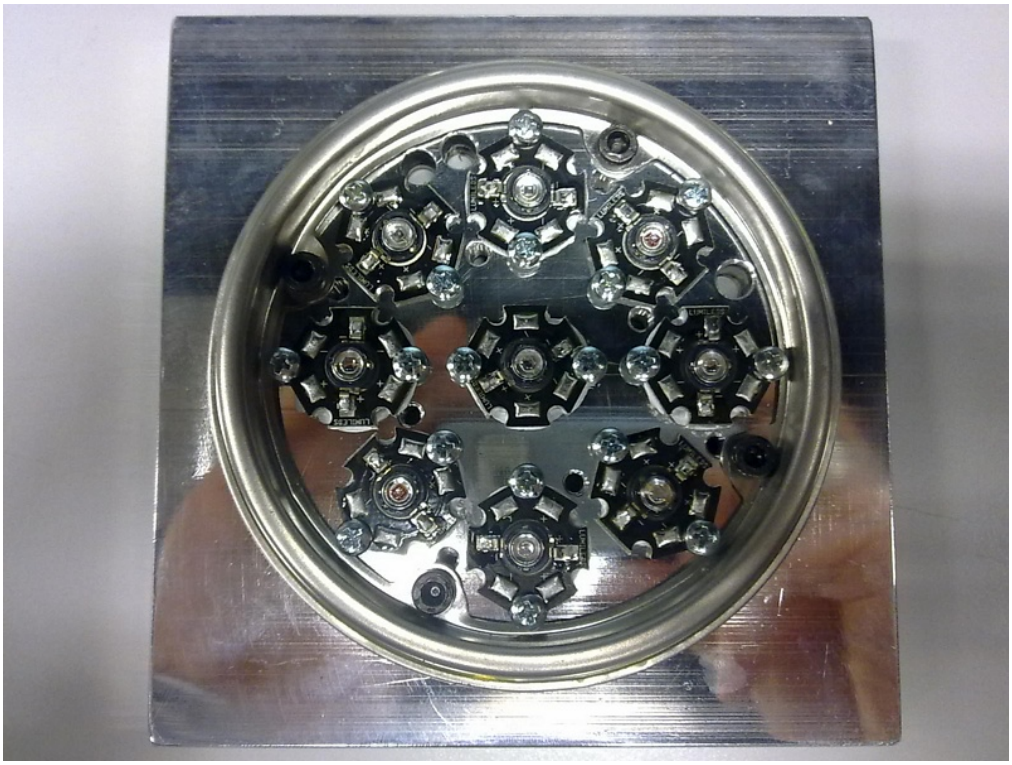


figura 4.5: piastra con tutti i LED montati e con il bordo del operchio fissato

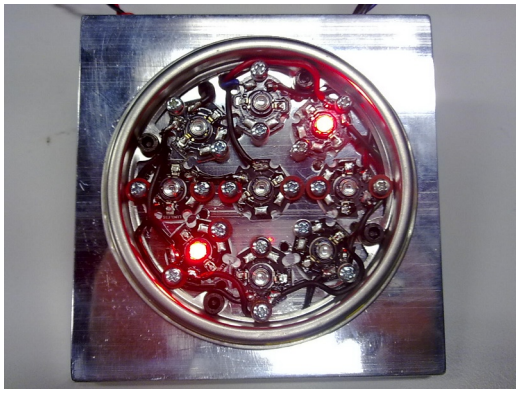


figura 4.6: LED rossi accesi

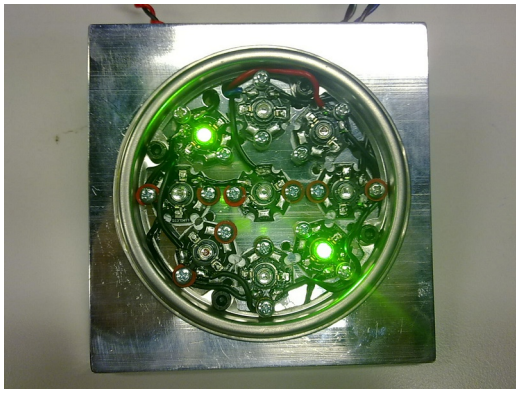


figura 4.7: LED verdi accesi

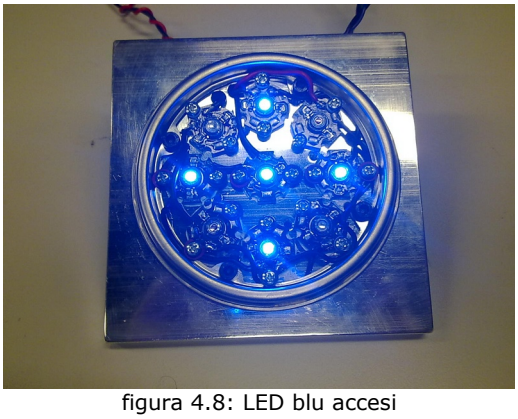


figura 4.8: LED blu accesi

4.1.3 circuito di alimentazione dei LED

Per ottenere le varie combinazioni di colori, era necessario poter regolare la luminosità dei LED canale per canale (volendo essere più precisi, il loro flusso luminoso; ma parlare di luminosità è più intuitivo, quindi d'ora in poi si utilizzerà questo termine).

A causa della loro caratteristica tensione-corrente molto ripida, in seguito alla quale una piccola variazione della tensione di alimentazione potrebbe provocare variazioni di corrente così grandi da distruggere il LED, solitamente i power LED vengono alimentati a corrente costante.

Ne consegue che, per regolare la luminosità dei LED, bisogna variare il valore di questa corrente imposta: questo si può fare con degli alimentatori dimmerabili con uscita a corrente costante.

La scelta è ricaduta, dopo varie ricerche, sui LUXDRIVE 3021 D-E-700 BuckPuck, che rispondevano a tutte le nostre esigenze:

- uscita in corrente costante a 700 mA massimi;
- alimentazione a corrente continua
- dimmerabili 0÷100% tramite un ingresso in tensione 0÷5 V
- buona tolleranza sull'uscita (5%) e buona efficienza (95%)

Essendo degli alimentatori da circuito stampato, si è inizialmente provato a montare tutta la componentistica su una basetta millefori; per avere un sistema più semplice e ordinato, si è però preferito disegnare e stampare con acido un circuito stampato.

Tale circuito stampato doveva comprendere, inizialmente:

- i tre alimentatori (uno per canale)
- due morsetti per canale (uno per l'alimentazione dei LED e uno per il controllo degli alimentatori)
- un morsetto per il bus dc di alimentazione degli alimentatori, con un condensatore elettrolitico da 1000 μF – 30 V
- un condensatore ceramico da 100 nF in parallelo all'alimentazione di ogni alimentatore

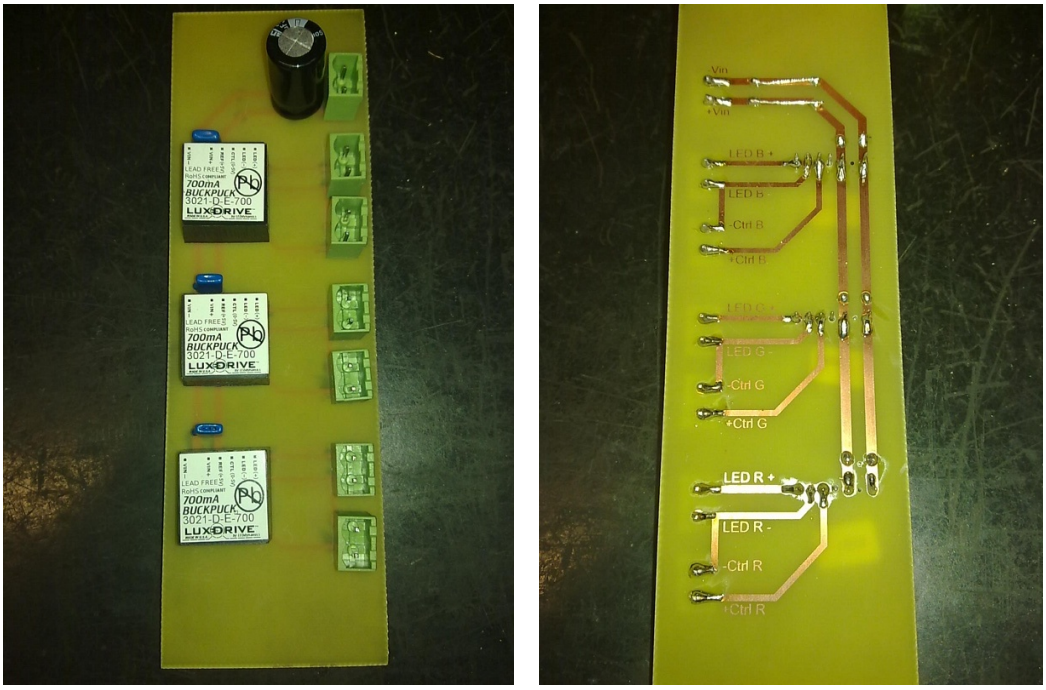


figura 4.9: circuito stampato (vista fronte e retro) per l'alimentazione dei LED

I LED di ogni canale sono stati collegati in serie, per garantire la stessa corrente a tutti, secondo lo schema (10) proposto nel data-sheet degli alimentatori:

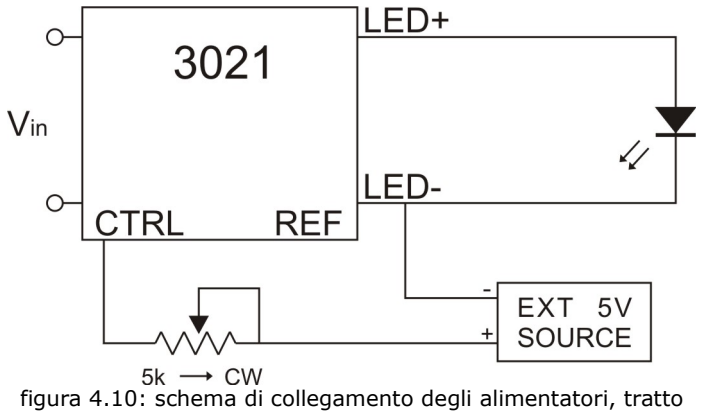


figura 4.10: schema di collegamento degli alimentatori, tratto dal data-sheet

Nel nostro caso specifico, anziché utilizzare una sorgente esterna a tensione fissa ed una resistenza variabile per regolare la tensione di controllo degli alimentatori, si è utilizzata una sorgente esterna a tensione regolabile da 0 a 5 V. Questa sorgente esterna di tensione è rappresentata dalle uscite analogiche della scheda DAQ, di cui si parlerà successivamente.

Dato che la tensione di alimentazione degli alimentatori deve essere di almeno 1 V superiore a quella della serie di LED, nel nostro caso è il circuito dei LED blu ad imporre la condizione: ogni LED blu ha una tensione Vf = 3,7 V, quindi alla serie di 5 LED blu deve essere applicata una tensione di 18,5 V. Di conseguenza, come tensione di alimentazione degli alimentatori servono almeno 20 V. Noi abbiamo impostato l'alimentatore stabilizzato a 24 V.

4.2 prime prove e risoluzione dei problemi

Dalle prime prove di funzionamento della sorgente, si è notata una certa fluttuazione del valore medio delle correnti assorbite dai LED: si è quindi provveduto a filtrare ulteriormente l'alimentazione dei LED, collegando un condensatore elettrolitico da 220 µF in parallelo all'alimentazione di ogni alimentatore.

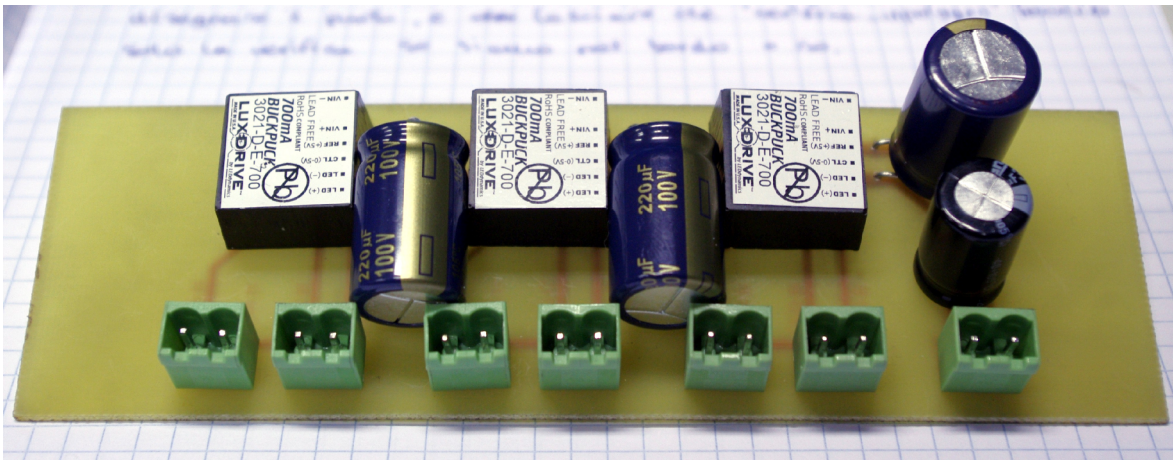


figura 4.11: circuito stampato con condensatori di filtro sul bus di alimentazione

Dopo aver così eliminato le fluttuazioni, nel corso di altre prove si è notato come la caratteristica tensione/corrente degli alimentatori non fosse lineare, specialmente a basse correnti; si è reso perciò necessario inserire in serie ai LED delle resistenze di shunt da 1 Ω, per poter acquisire il valore della corrente e provvedere quindi ad una retroazione.

Le resistenze sono visibili nella foto seguente: sono quei parallelepipedi bianchi tra gli alimentatori e i morsetti.

figura 4.12: circuito stampato con aggiunte le resistenze di shunt per l'acquisizione delle correnti di alimentazione dei LED

Misurando il valore delle correnti di alimentazione dei LED, si è notato un certo rumore nelle acquisizioni, generato principalmente dagli alimentatori switching.

Allora si è provveduto a realizzare dei filtri passa-basso: poiché gli alimentatori switching lavorano con una frequenza di commutazione di 10 kHz, i filtri sono stati realizzati con due resistenze da 1,5 kΩ e un condensatore ceramico da 100 nF.

La frequenza di taglio dei filtri è quindi di:

$f_T = 1 / 2\pi RC \approx 530$ Hz

In effetti il rumore delle acquisizioni, dopo l'applicazione dei filtri passa-basso, si è ridotto: anche se sono rimaste delle componenti dovute alla frequenza di commutazione degli switching, il loro valore picco-picco è di circa 5 mA su 600 mA, quindi meno dell'1%.

Comunque, la misura del valore di corrente viene poi calcolata facendo una media su 1000 campioni: quindi il rumore assume un valore irrilevante, dato che viene diviso ulteriormente per √1000 (regola per la quantificazione del rumore in una misura mediata).

Oltre a quanto sopra detto, restava comunque il problema dello sbandamento di emissione luminosa dei LED: l'emissione luminosa si riduce con l'aumentare della temperatura. Questa riduzione di potenza luminosa è reversibile e non ha influenza sul decadimento del flusso nel corso della vita.

Inoltre, la massima temperatura di funzionamento per i LED è normalmente dì 100 °C e non deve essere superata né tantomeno raggiunta, altrimenti si rischia di abbreviarne la vita.

Per tale motivo, è stato montato un dissipatore sul retro della piastra su cui erano fissati i LED, fissandolo con due viti e interponendo uno strato di pasta all'ossido di argento, in modo da garantire un buon scambio termico; su questo dissipatore è stata montata una ventola, per facilitare lo smaltimento del calore.

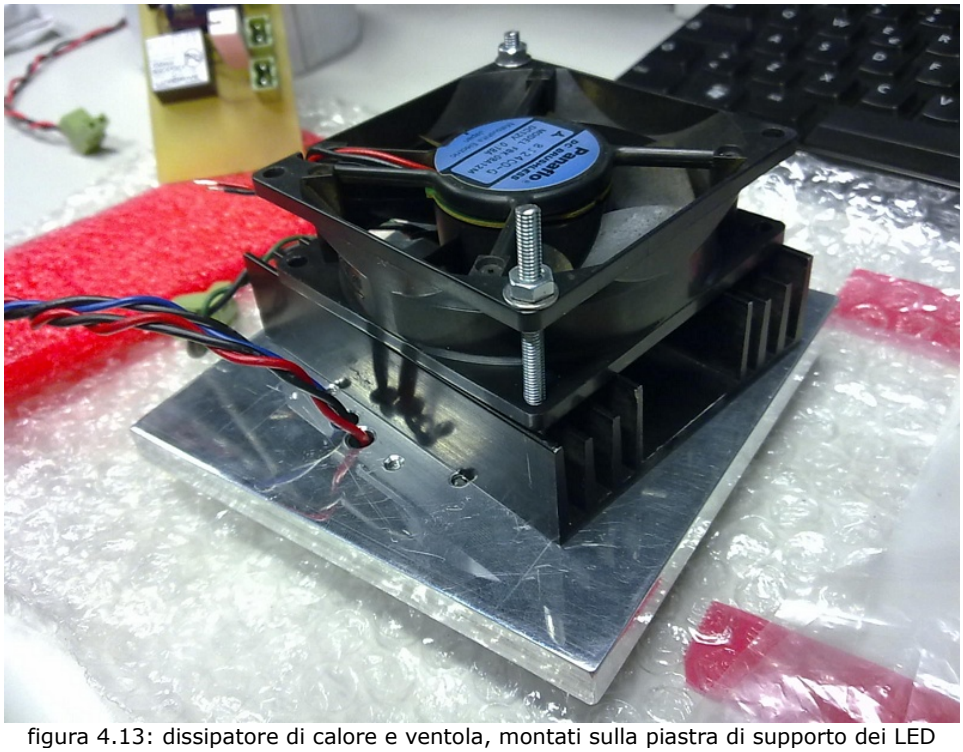


figura 4.13: dissipatore di calore e ventola, montati sulla piastra di supporto dei LED

Per tenere sotto controllo la temperatura, è stato anche inserito (in un nuovo foro ricavato nella piastra) un trasduttore di temperatura di tipo integrato, la cui uscita viene acquisita per visualizzare sul pc il valore della temperatura.

figura 4.14:
trasduttore di
temperatura Analog
Devices AD590

Dalle misure di temperatura effettuate si è potuto vedere che, mantenendo tutti i LED accesi alla massima potenza e con la ventola disalimentata, quindi sfruttando solo il raffreddamento dovuto alla naturale circolazione dell'aria, nel giro di una mezz'ora la temperatura della piastra si stabilizza poco al di sopra di 58 °C.
Alimentando a quel punto la ventola, in una decina di minuti la temperatura della piastra si stabilizza su un valore di poco superiore ai 29°C.

Supponendo che il salto di temperatura tra la giunzione dei LED e la piastra sia dell'ordine dei 30÷40 °C, si può dire tranquillamente che, grazie al dissipatore installato e alla ventola, si può far lavorare la sorgente a piena potenza per periodi di tempo illimitati senza rischiare di raggiungere temperature dannose per la vita dei LED stessi.

Attualmente la ventola è sempre alimentata direttamente da un alimentatore stabilizzato, ma si potrebbe fare in modo che, fino al raggiungimento di una certa temperatura (pari a quella di regime) la ventola resti spenta, velocizzando così il raggiungimento del regime termico della sorgente.

Eventualmente, dato che è stato provato che esiste uno sbandamento delle tensioni di comando degli alimentatori e delle coordinate cromatiche con la temperatura (dovute alla riduzione del flusso luminoso dei LED a parità di corrente, all'aumentare della temperatura), per velocizzare il raggiungimento del regime termico e per mantenere la temperatura costante indipendentemente dalla potenza dei LED utilizzata, si può prevedere l'uso di una resistenza corazzata, comandata da un termostato e utilizzata per scaldare la piastra.

Capitolo 5 - Interfaccia software-hardware

Per poter comandare la sorgente attraverso il computer, si è realizzato uno stadio di interfaccia software-hardware, utilizzando due schede DAQ prodotte dalla National Instruments: le NI USB-6008.

In laboratorio era disponibile anche una scheda NI USB-6259, dotata di un maggior numero di ingressi ed uscite, ma per i nostri scopi le due 6008 erano più che sufficienti: infatti garantivano un numero adeguato di ingressi ed uscite, sia digitali che analogici, fino a 12 bit, dando anche la possibilità di utilizzarli in modalità differenziale anziché single-ended.

Le due schede DAQ sono state collegate al PC tramite USB, e riconosciute mediante il software NI Device Monitor. Questo software in realtà è servito solamente per ottenere l'identificazione delle schede, perché poi il loro comando è avvenuto esclusivamente attraverso Matlab e l'apposito Data Acquisition Toolbox.

Le schede NI USB-6008 non dispongono di un generatore di clock interno, per cui è stato possibile comandare gli alimentatori solamente in modalità statica, ovvero inviando o acquisendo uno o più campioni alla volta; non è quindi possibile inviare alla scheda dei segnali nel dominio del tempo, ma per noi questo non è stato un grosso limite.

Inizialmente, le schede DAQ dovevano essere utilizzate solo per fornire la tensione di regolazione degli alimentatori dimmerabili; successivamente, si sono sfruttati i canali analogici in ingresso per la lettura dei valori della corrente di alimentazione dei LED, e della temperatura della piastra di supporto dei LED.

NOTA: nonostante tutta la prima parte del lavoro di progettazione sia stata fatta utilizzando le due schede 6008, verso la fine del lavoro si è passati alla 6259: in questo modo si è potuto sfruttare il pieno supporto di questa scheda da parte di Matlab (la 6008 non era pienamente supportata, infatti non si riusciva ad esempio a settare il range delle tensioni in ingresso).
Oltre a questo, il grosso vantaggio nell'utilizzare la 6259 si è avuto nell'incremento della risoluzione nella acquisizione dei valori di corrente, ma soprattutto nella maggiore frequenza di campionamento: questo ultimo particolare ha reso molto più rapide le acquisizioni di 1000 campioni, e ciò è andato a vantaggio soprattutto del processo di retroazione, che è diventato notevolmente più veloce.

Nella figura 5.1, una scheda NI USB-6008, utilizzata nella prima parte del lavoro di tesi; nella figura 5.2, la scheda NI USB-6259 utilizzata successivamente.

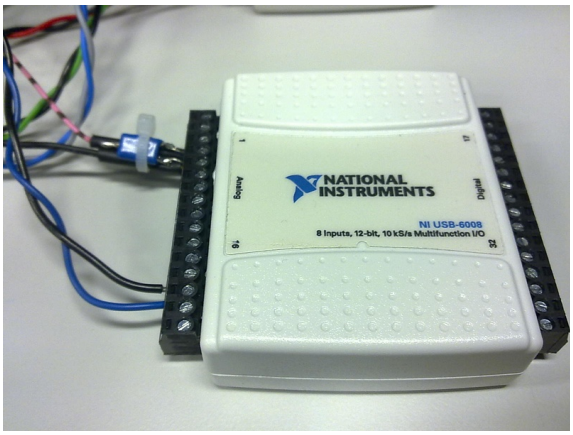


figura 5.1: scheda di acquisizione dati National Instruments NI USB-6008

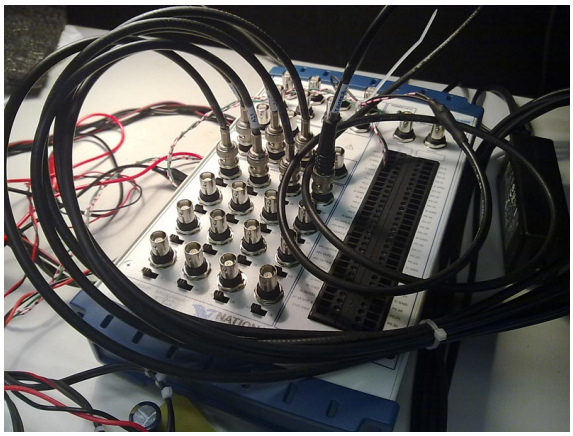


figura 5.2: scheda di acquisizione dati National Instruments NI USB-6259

5.1 comando degli alimentatori dimmerabili

Per alimentare i LED con una determinata corrente, bisogna fornire all'ingresso di controllo degli alimentatori dimmerabili una tensione appropriata.

Questa tensione viene generata dalle schede DAQ, in base al valore richiesto di corrente ed alla caratteristica tensione di controllo/corrente di uscita degli alimentatori.

Come valore richiesto della corrente di alimentazione dei LED, per semplicità si è inizialmente lavorato con valori compresi tra 0 ed 1: a 0 il LED deve essere spento, a 1 il LED deve essere acceso alla massima potenza; a valori intermedi, la corrente di alimentazione del LED deve essere proporzionale al valore desiderato.

La caratteristica tensione di controllo / corrente di uscita degli alimentatori è riportata sul data-sheet degli alimentatori stessi:

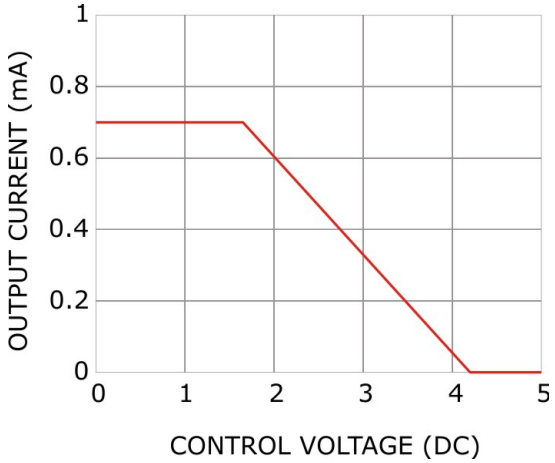


figura 5.3: caratteristica tensione di controllo / corrente di uscita degli alimentatori

Tra gli altri dati, venivano specificati la tensione di spegnimento, pari a 4,2 V, e la tensione al di sotto della quale la corrente di uscita è massima (pari a 1,65 V).

Dovendo ricavare la tensione di controllo dal valore di corrente desiderato, si è calcolata la funzione inversa della caratteristica sopra raffigurata, semplicemente come l'equazione di una retta passante per i punti (1,65;1),(4,2;0).

L'equazione quindi è risultata:

$$TensControllo = -2.55 \times CorrenteDesiderata - 4.2$$

Calcolata la tensione da generare per ogni canale, è quindi sufficiente aprire le porte di comunicazione tra pc e schede DAQ (script "ApriPorte"), inviare alle schede DAQ i valori di tensione desiderati (script "ComandoSchedaDAQ") e successivamente chiudere le porte di comunicazione (script "ChiudiPorte").
Con l'utilizzo della scheda NI USB-6259, i programmi appena citati sono stati modificati, e chiamati rispettivamente "ApriPorte_6259", "ComandoSchedaDAQ_6259", "ChiudiPorte_6259".

5.2 acquisizione delle correnti di alimentazione dei LED

Purtroppo, gli alimentatori da noi utilizzati presentavano una marcata non linearità della caratteristica, specialmente per bassi valori di corrente richiesta.
Infatti, si notava anche ad occhio come, al variare dei valori impostati da computer, le variazioni del colore prodotto fossero diverse a seconda che si lavorasse vicino allo zero o vicino al massimo.

Bisognava quindi prevedere un'acquisizione del valore delle correnti di alimentazione dei LED, canale per canale: per fare questo, si sono poste delle resistenze di shunt da 1 Ω in serie al circuito di alimentazione dei LED di ogni canale; la tensione misurata ai loro capi è stata riportata a tre ingressi analogici delle schede DAQ.

Valutazioni effettuate usando le schede NI USB-6008:

Si è preferito utilizzare tali ingressi in modalità differenziale, per evitare che un rumore di modo comune prodotto dagli alimentatori potesse falsare le misure; dunque, il convertitore A/D per questi canali ha risoluzione di 11 bit.

Poiché il range di tensioni degli ingressi analogici delle schede DAQ è 0÷10 V in modalità differenziale, ed essendo la tensione sulle resistenze di shunt di circa 700 mV, si evince che l'acquisizione viene fatta sfruttando al massimo 8 bit. Dunque, la risoluzione di questa acquisizione è di 1/256=0,4%, valore comunque sufficiente per i nostri scopi.

Dalle prime acquisizioni di corrente era emerso un notevole rumore, anche del 10% (in valore picco-picco) rispetto al valore medio di corrente assorbita dai LED: tale rumore era imputabile principalmente agli alimentatori switching.
Il problema è stato risolto con l'apposizione di filtri passa-basso con frequenza di taglio a circa 530 Hz, che hanno ridotto il rumore a un valore picco-picco di poco meno dell'1% rispetto al valore medio di corrente assorbita.

Per limitare ulteriormente il contributo del rumore, che è particolarmente pericoloso quando si fanno misure su un solo campione, si è preferito acquisire 1000 campioni per ogni misura, e calcolare il valore della corrente assorbita come la media su questi 1000 campioni.
In questo modo, il valore delle corrente è risultato essere sufficientemente stabile, con variazioni di un paio di millivolt su centinaia di millivolt: più o meno lo stesso risultato che si otterrebbe lavorando con un multimetro a 3 ½ digit.

Lo script Matlab che effettua le operazioni di acquisizione mediante le due schede NI USB-6008 è "LetturaCorrentiTempProlungata".

Valutazioni effettuate usando la scheda NI USB-6259:

Utilizzando la scheda NI USB-6259, si è avuta a disposizione una maggiore risoluzione, e la possibilità di impostare il range degli ingressi analogici.

Lo script Matlab che effettua le operazioni di acquisizione mediante la scheda NI USB-6259 è "LetturaCorrentiTempProlungata_6259".

Poiché la corrente dei LED non può essere superiore a 700 mA e le resistenze di shunt hanno valore di 1 Ω, la massima tensione ai loro capi può essere di 700 mV: per tale motivo, come range degli ingressi analogici della scheda DAQ, si è scelto -1÷1 V (non è possibile utilizzare un range del tipo 0÷1 V). Il convertitore A/D della NI USB-6259 lavora a 16 bit indipendentemente dal range della tensione di ingresso: quindi la risoluzione, utilizzando questo range, è di $2/2^{16} = 30$ µV.

In effetti, questa maggiore risoluzione, ottenibile con la scheda NI USB-6259 rispetto alla NI USB-6008, porta un notevole beneficio sulla precisione della misura delle correnti di alimentazione dei LED.

Nonostante ciò, l' acquisizione delle caratteristiche degli alimentatori era stata effettuata utilizzando le due schede NI USB-6008, e mantiene comunque la sua validità, poiché serviva solo per dare un'idea di massima sulla caratteristica degli alimentatori.

Avendo la possibilità di misurare le correnti di alimentazione dei LED, con lo script in Matlab "TestCaratteristicaDAQ" è stata infatti misurata la caratteristica tensione di controllo / corrente di uscita degli alimentatori dimmerabili.
Si è potuto vedere che i tre alimentatori hanno caratteristiche leggermente diverse, sia come linearità che come valore massimo di corrente che sono in grado di erogare.

Per ovviare a questo problema, si è deciso di ricorrere ad un circuito di retroazione che imponesse il valore della corrente desiderata.

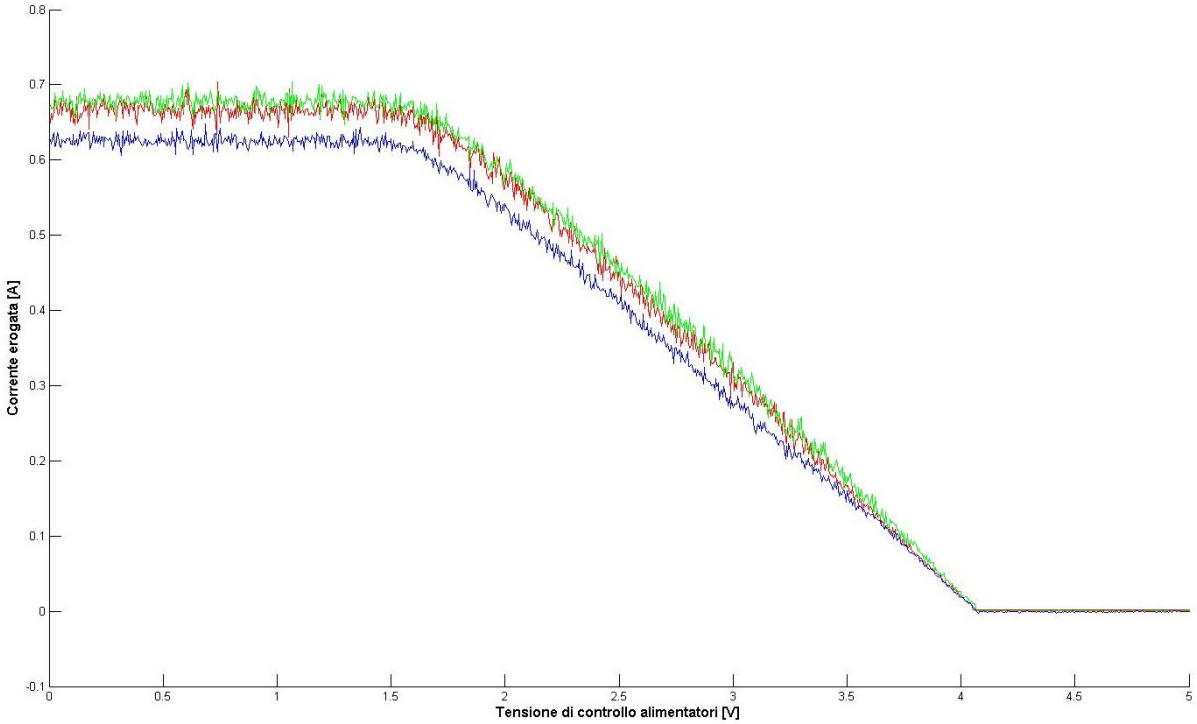


figura 5.4: acquisizione delle caratteristiche tensione di controllo / corrente di uscita degli alimentatori

(nota: le caratteristiche mostrate nella figura 5.4 sono quelle misurate prima dell'introduzione dei filtri passa-basso: per questo motivo esse appaiono molto “sporcate” dal rumore)

5.3 retroazione per il controllo della corrente di alimentazione dei LED

Dato che la caratteristica tensione di controllo / corrente di uscita degli alimentatori non è lineare, come appena visto, ed è inoltre diversa tra i tre alimentatori, ci si è resi conto che non era sufficiente impostare il valore di corrente desiderato per avere la certezza che questo fosse il valore reale della corrente di alimentazione dei LED.

L'unico modo di avere questa certezza era quello di trasformare il circuito "a catena aperta" in un circuito "a catena chiusa", quindi dotato di un sistema di retroazione.

La retroazione è effettuata da uno script ("RetroazionePI", modificato poi in "RetroazionePI_6259" per utilizzare la scheda NI USB-6259) che prende come Set Point il valore di corrente richiesto (la variabile Terna_correnti_RGB, che ha valore compreso tra 0 e 1 per ogni canale); questo valore viene confrontato con il feedback, proporzionale al valore effettivo della corrente di alimentazione dei LED, acquisito come visto prima.

Questa retroazione viene effettuata all'interno di un ciclo in Matlab, che si ferma solo quando l'errore in tutti e tre i canali è inferiore allo 0,5‰ del valore massimo ottenibile (ovvero 600mA, corrispondenti ad un valore di Terna pari a 1).

Il valore massimo di 600 mA per la corrente è stato scelto in base all'acquisizione mostrata prima: il canale con corrente più bassa (il blu) aveva una corrente di poco superiore ai 600 mA, quindi scegliendo di erogare al massimo 600 mA su tutti i canali si ha la sicurezza di non andare mai oltre le capacità di ciascuno degli alimentatori.

Inizialmente, quando si utilizzavano le schede NI USB-6008, come tolleranza minima richiesta si era preso un valore dell'1% rispetto al valore di corrente richiesto: ecco che, quando si richiedevano basse correnti, l'errore non riusciva a diventare minore di una tolleranza diventata piccolissima, e il programma continuava a girare.
Questo succedeva perché l'errore calcolato ha una certa risoluzione, che in buona approssimazione è la stessa risoluzione dell'acquisizione di corrente: avendo quindi una risoluzione per l'errore dello 0,4% e una tolleranza (se si richiedeva ad esempio una corrente di 0,2) dello (0,2 x 1%) = 0,2%, è ovvio che non si riuscisse mai a ricadere nella fascia di tolleranza, ma la retroazione continuasse ad oscillare.

Per evitare questo problema, bisognava prendere una "isteresi" pari almeno al doppio della risoluzione sull'errore: si è presa quindi come "isteresi" del valore dell'1% sul massimo valore ottenibile (ed essendo quest'ultimo 1, la tolleranza dell'1% è da riferirsi all'errore assoluto),

Passando successivamente alla scheda NI USB-6259, la risoluzione dell'acquisizione è notevolmente migliorata, e quindi si è potuto anche spingere sulla precisione della retroazione: come detto prima, è stata scelta una tolleranza di 0,3 mA (pari allo 0,5‰ del valore massimo impostabile, ovvero 600 mA): dato che la risoluzione dell'acquisizione è di circa 30 μV, corrispondenti a 30 μA, questa tolleranza garantisce una "isteresi" di 10 volte la risoluzione, per cui non ci sono problemi di non convergenza della retroazione.

Inizialmente si era prevista solo un'azione correttiva proporzionale all'errore (controllore tipo P), ma si è subito visto che si presentava il problema tipico di tali sistemi di retroazione: la presenza di un errore a regime.
Per eliminare questo errore a regime, si è dovuto implementare anche un controllo integrale dell'errore. Solitamente, in applicazioni a tempo continuo, tale integrale viene fatto per l'appunto nel dominio del tempo; nel nostro caso, essendo il controllo a tempo discreto, l'unico modo di fare l'integrale era fare la somma degli elementi del vettore degli errori. Quello che è stato realizzato, quindi, può essere definito un "controllore PI a tempo discreto".

Il valore dell'errore integrale, in condizioni di normale funzionamento, arriva a circa 1,2; per evitare che, in seguito a una perturbazione temporanea sul circuito, tale valore diventasse troppo grande, e richiedesse quindi molto tempo per essere "scaricato" una volta passata la perturbazione, si è imposto un limite pari a 5 per questo valore. In questo modo, bastano una decina di cicli per "scaricare" l'errore integrale e riuscire a riportare la corrente a regime.

Una parte delicata del processo di progettazione della retroazione è stata la scelta del valore dei parametri moltiplicativi con cui pesare il segnale di errore.
Infatti, il valore di tali parametri può assicurare un raggiungimento più rapido del valore desiderato, ma c'è anche il rischio che il sistema abbia una risposta instabile.

Dopo varie prove, e dopo aver valutato anche la possibilità di implementare anche il controllo differenziale dell'errore (cosa però difficilmente attuabile, poiché la derivata viene calcolata in ritardo, e per giunta come rapporto incrementale della differenza tra il valore di due campioni), finalmente si sono trovati dei valori delle costanti moltiplicative che fornissero una risposta abbastanza veloce nell'andare a regime, ma soprattutto che convergesse per qualsiasi valore del set-point.

Tali valori sono Kp = 0,8 e Ki = 0,5.

Quando viene richiesta una terna di correnti del tipo [0 1 0.5] (ovvero, con uno o più valori pari a zero), la corrente dei LED nel canale che è stato impostato a zero è effettivamente nulla (o comunque piccolissima, qualche decimo di microampere).
A causa di un fenomeno di offset, che si è scoperto affliggere le misure effettuate con la scheda NI USB-6259 e di cui non è ancora ben nota la causa (anche se sembra essere causato da cross-talking tra i canali), la lettura sul canale con corrente nulla non è zero, ma viene letta una tensione di qualche decimo di millivolt.
Questa lettura falsata ricadeva al di fuori della tolleranza per la retroazione, ma essendo in realtà la corrente nulla, la retroazione non poteva fare niente per correggerla, e continuava a ciclare; nei canali con corrente non nulla, invece, pur misurando in modo leggermente falsato, la retroazione portava la corrente al valore desiderato.

Per risolvere questo problema, almeno fino a quando non si scoprirà ed eliminerà la vera causa dell'offset, si è aumentata la tolleranza per i canali in cui viene impostato un valore nullo di corrente: per quei canali, quindi, la tolleranza viene impostata a 5×10^{-3}. Questo permette di non risentire dell'offset nemmeno quando raggiunge i valori più alti (il che avviene quando si richiede la terna [0 0 1]).

Vedendo che, con i valori di Kp e Ki detti prima, la retroazione normalmente converge in meno di una decina di cicli, si è messo un limite di 50 cicli per la retroazione stessa: se al cinquantesimo ciclo l'errore non è ancora rientrato all'interno della tolleranza, compare a video il messaggio “Colore non raggiunto: retroazione interrotta” e la retroazione si interrompe.
Se invece la retroazione converge prima dei 50 cicli, il messaggio è “Colore raggiunto: retroazione conclusa in N cicli”.

Per fare in modo che la retroazione sia sempre pronta ad aggiustare il valore delle correnti in caso di una qualche perturbazione sul circuito, si è fatto in modo che, se anche non si immette nessun nuovo valore, dopo un tempo impostabile a piacere (al momento è di 5 secondi) il programma dell'interfaccia grafica rimanda fuori i valori del ciclo precedente.

Se questi valori sono uguali a quelli del ciclo precedente, il programma della retroazione prende come terna di correnti da richiedere agli alimentatori quella che era venuta fuori dal precedente intervento di retroazione. In questo modo, vengono erogate fin da subito le correnti esatte, e la retroazione deve fare al massimo un ciclo o due, per farle eventualmente rientrare all'interno della tolleranza richiesta.

5.4 acquisizione del valore di temperatura

Per poter tenere sotto controllo la temperatura dei LED, dato che una temperatura di giunzione superiore ai 100 °C pregiudica la vita dei LED stessi, l'unico modo è quello di misurare la temperatura della piastra su cui sono alloggiati i LED, ipotizzando che il salto di temperatura tra giunzione e piastra si attesti sui 30-40 °C.

Per misurare la temperatura della piastra, è stato effettuato un foro sulla piastra stessa, in cui è stato alloggiato un trasduttore di temperatura ANALOG DEVICES AD590.

Questo sensore di temperatura, quando è alimentato ad una tensione continua compresa tra i 4 V e i 30 V, fornisce una corrente costante e variabile con la temperatura (aumento di 1 µA per ogni 1 °C di aumento di temperatura).

L'alimentazione del trasduttore è stata inizialmente prelevata da un alimentatore stabilizzato; poi, quando si è passati ad usare la scheda NI USB-6259, si è sfruttato il generatore di tensione a 5 V a bordo della scheda.

Il trasduttore è tarato per fornire una corrente di 298,2 µA a una temperatura di 298,2 K (ovvero 25 °C); ad esempio, ad una temperatura di 28 °C, la corrente erogata sarà di
298,2 + (28 – 25) = 301,2 µA.

E' stato quindi sufficiente, per poter acquisire una tensione proporzionale alla temperatura, porre in serie con il circuito di alimentazione dell'integrato una resistenza di shunt.
Per sfruttare quanto più possibile la risoluzione della scheda DAQ, si è scelto di usare una resistenza da 1 kΩ, prevedendo che la temperatura della piastra possa raggiungere i 60 °C (a cui corrisponde una corrente di 298.2 + (60 - 25) = 333,2 µA), e di utilizzare il canale di acquisizione in modalità differenziale.

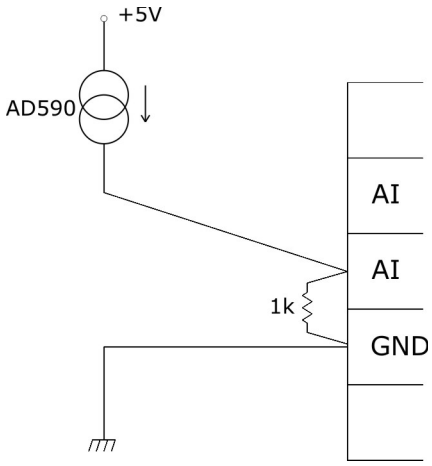


figura 5.5: schema di collegamento del trasduttori di temperatura per l'acquisizione del valore di temperatura della piastra

Utilizzando la scheda NI USB-6259 con gli ingressi analogici in modalità differenziale, e con gli ingressi riferiti a terra (GS: Ground Source) anziché alla massa della scheda (FS: Floating Source), si sono escluse le resistenze da 5 kΩ presenti tra ogni ingresso e la massa della scheda, per cui la resistenza apparente è pari a quella reale, ovvero 1 kΩ.

Per la precisione, la resistenza di shunt è stata misurata ed è risultata pari a 0,972 kΩ.

La formula per calcolare la temperatura è quindi:

$$temperatura = TensMisurata \times (1000/0{,}972) - 273{,}15$$

Dopo aver eliminato l'errore di guadagno, che dipende dal valore della resistenza, si è provveduto ad inserire, nel calcolo della temperatura, un parametro che tenesse conto dell'offset di temperatura introdotto dal trasduttore: dopo aver lasciato la sorgente spenta per un giorno, è stata effettuata una doppia misurazione della temperatura della piastra (sia con il programma che con una termocoppia da multimetro).
La temperatura indicata dalla termocoppia era di 21,9°C, ed in effetti la temperatura ambiente era di 21,8°C; la temperatura calcolata dal programma era invece di 24,4°C; quindi, si è introdotta nel calcolo una correzione dell'offset pari a -2,5°C.

Quindi, l'equazione definitiva per il calcolo della temperatura è diventata:

$$temperatura = TensMisurata \times (1000/0{,}972) - 273{,}15 - 2.5$$

In effetti, questo valore di offset è compatibile con le specifiche dichiarate dal produttore del trasduttore di temperatura: per questo particolare modello, infatti, il costruttore dichiara un possibile errore di calibrazione di ±5°C.

Come per le acquisizioni di corrente, anche l'acquisizione della temperatura è stata effettuata come media sul valore di 1000 campioni, in modo da evitare il contributo del rumore.

Ciò nonostante, l'offset che interessa le misure di corrente (che vengono sempre effettuate con la scheda NI USB-6259) interessa anche la temperatura: per questo motivo, la lettura precisa si ha quando i LED sono spenti, e quindi non ci sono correnti che possano determinare un *cross-talking* con il canale di acquisizione della temperatura.

Capitolo 6 - Misure

Durante e dopo la realizzazione della sorgente, le misure sono state uno strumento insostituibile per verificare la presenza, nella risposta dei vari elementi costituenti la sorgente in oggetto, di problemi o di scostamenti dal valore ideale.

Durante la progettazione di un insieme di componenti, infatti, si parte sempre dall'ipotesi che essi abbiano un comportamento ideale (caratteristiche lineari, assenza di disturbi emessi, immunità ai disturbi introdotti dall'ambiente di lavoro); poi, mano a mano che il progetto prende forma, si può andare ad esaminare, blocco per blocco, di quanto il comportamento reale si discosti da quello ideale: se tale differenza non è trascurabile, si deve agire per renderla minima.

Nell'ambiente industriale, dove bisogna contenere quanto più possibile i costi e dove il tempo è denaro, il processo appena descritto viene seguito esclusivamente in fase di simulazione: infatti, rimediare ad una “non idealità” del comportamento in fase di simulazione è decisamente meno costoso che farlo su un modello reale, sia pure esso un prototipo.
Però, anche al livello industriale non è possibile tenere in conto di ogni non idealità prima della realizzazione di un prototipo, soprattutto quando non si è mai avuta esperienza diretta di tutti i componenti da utilizzare.

I problemi causati dalle non idealità sono poi maggiormente importanti in uno strumento di misura, quale è quello che è stato realizzato nella tesi.

Nel nostro caso, trattandosi di un esemplare singolo, che è costituito da una parte software, una parte hardware e una parte di interfaccia, riuscire a modellizzare tutti i componenti sarebbe stato alquanto oneroso in termini di tempo: si è preferito quindi, come detto prima, effettuare delle misure che consentissero di “aggiustare il tiro” mano a mano che si avanzava nel progetto, quando i problemi più importanti erano stati risolti e si poteva dedicarsi ad aspetti fino a quel punto trascurati.

Tutte le misure sono state eseguite con questo *experimental set-up*:

- Tensione di alimentazione bus DC degli alimentatori: 24V;
- Tensione di alimentazione della ventola: 12V;
- Tensione di alimentazione del trasduttore di temperatura: 5V, prelevati dalla scheda NI USB-6295;
- Schede di acquisizione dati NI USB-6008 e successivamente NI USB-6259;
- Spettroradiometro Minolta CS-1000, connesso tramite porta RS232 al PC, comandato dal programma dedicato CS-S1w per misure manuali, o comandato dagli script Matlab per l'acquisizione dati all'interno di una routine.

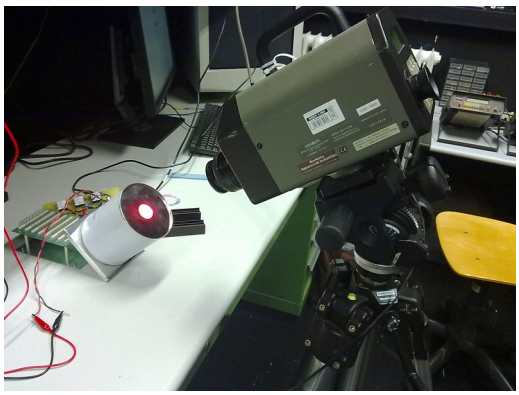


figura 6.1: misurazione con spettroradiometro

6.1 ricerca della configurazione geometrica ottimale

Il modello a cui si è fatto riferimento, per la realizzazione del contenitore diffondente per la sorgente, è quello della sfera integratrice.
In questo modello, la sorgente occupa una porzione relativamente piccola della totale superficie interna della sfera, ed è occultata alla vista dell'osservatore da un setto, anch'esso piccolo rispetto al diametro della sfera.
Una delle proprietà fondamentali della sfera, necessaria per il suo funzionamento, è che la luminanza di tutti i punti della superficie interna sia uguale: dunque, qualsiasi sia la direzione di osservazione (comunque limitata dall'apertura dedicata all'osservatore, che dev'essere sufficientemente piccola), la luminanza vista non cambia.

Nel nostro caso, non avendo a disposizione una sfera integratrice ma un “cilindro integratore”, bisognava verificare che tale caratteristica di invarianza della luminanza fosse verificata, eventualmente con una certa tolleranza rispetto al caso ideale della sfera.

Inoltre, nel nostro caso, essendo la sorgente relativamente grande rispetto al contenitore diffondente e all'apertura di osservazione, bisognava utilizzare un setto di dimensioni relativamente grandi rispetto al diametro del cilindro.

Questo fatto ha introdotto un limite rispetto alla sfera integratrice, poiché nel nostro caso al variare della distanza del setto dalla sorgente e dal foro di osservazione, e al variare del suo diametro, variavano il valore massimo e l'uniformità della luminanza sulla superficie osservata.

Oltre a questo, erano stati preparati due contenitori diffondenti: delle stesse dimensioni, ma il foro di osservazione era da 1” su uno e da 2” sull'altro.
Come era presumibile, anche questo parametro si è rivelato molto influente sul valore della luminanza del setto (specialmente sul suo valore massimo, poiché, come è intuibile, con un foro più largo è maggiore la quantità di luce dispersa).

Dunque, nonostante alcune prove siano state effettuate con il foro da 2”, alla fine si è deciso di utilizzare il barattolo con il foro da 1”, che garantisce un maggior valore di luminanza.
Inoltre, essendo il foro di osservazione piccolo, anche osservando all'interno del contenitore con un angolo abbastanza ampio rispetto alla normale, non si vedono le pareti del contenitore, dove sono invece maggiormente visibili i contributi separati dei tre colori.

Per trovare la miglior configurazione distanza del setto dal foro di osservazione, sono state effettuate varie misure, per verificare l'andamento del valore massimo e dell'uniformità della luminanza:

- misure con setto diametro 80 mm posizionato alla massima distanza dalla sorgente (e quindi a 50 mm dal foro di osservazione);
- misure con setto diametro 80 mm in posizione intermedia tra la sorgente e il foro di osservazione (a 77 mm dal foro di osservazione).

Tali misure sono state effettuate in quattro punti della superficie di osservazione, per verificare l'uniformità tra il centro, il bordo e le diagonali relative al posizionamento dei LED verdi e rossi. In particolare, sono stati scelti quattro punti di misura:

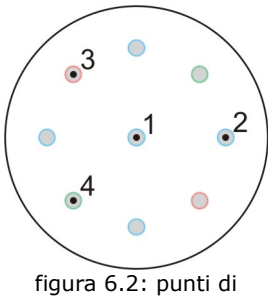


figura 6.2: punti di misura della luminanza

I punti 3 e 4 sono stati scelti in quella precisa posizione perché le misure sono state effettuate accendendo il solo canale verde al 25% della potenza: il punto 3 si trova in corrispondenza del LED rossi, distante dai LED verdi, mentre il punto 4 si trova in corrispondenza dei LED verdi.
Dunque, in caso di non uniformità della luminanza, era probabile che la luminanza misurata nel punto 4 fosse più alta di quella misurata nel punto 3 (che, essendo un punto distante da entrambi i LED verdi, teoricamente avrebbe dovuto

essere quello a luminanza più bassa tra i quattro).

Da quelle misure si è notato che il valore massimo della luminanza diminuisce arretrando il setto (ovvero avvicinandolo alla sorgente e allontanandolo dal foro di osservazione), però l'uniformità della luminanza aumenta, come era prevedibile.

In base a queste conclusioni, si è deciso di mantenere il setto in posizione arretrata, utilizzando però il barattolo con il foro da 1" per sopperire alla diminuzione del valore massimo di luminanza.

Scelta la configurazione, si è successivamente proceduto alla misurazione del valore della luminanza, al centro del setto, per i vari colori:

- misura per la terna [1 0 0], ovvero solo canale rosso alla massima potenza;
  valori ottenuti: x=0,6937 y=0,3060 L=1782 [cd/m^2]
- misura per la terna [0 1 0], ovvero solo canale verde alla massima potenza;
  valori ottenuti: x=0,2600 y=0,6905 L=1312 [cd/m^2]
- misura per la terna [0 0 1], ovvero solo canale blu alla massima potenza;
  valori ottenuti: x=0,1266 y=0,0808 L=1485 [cd/m^2]

Si vede che la luminanza più bassa si è ottenuta con il canale Verde: 1312 [cd/m^2]

Quindi, questo valore è quello massimo che siamo sicuri di poter ottenere, indipendentemente dal colore scelto: perciò, nell'interfaccia grafica, il valore massimo selezionabile per L quando si lavora nello spazio colore Lxy è stato impostato a 1300 [cd/m^2].

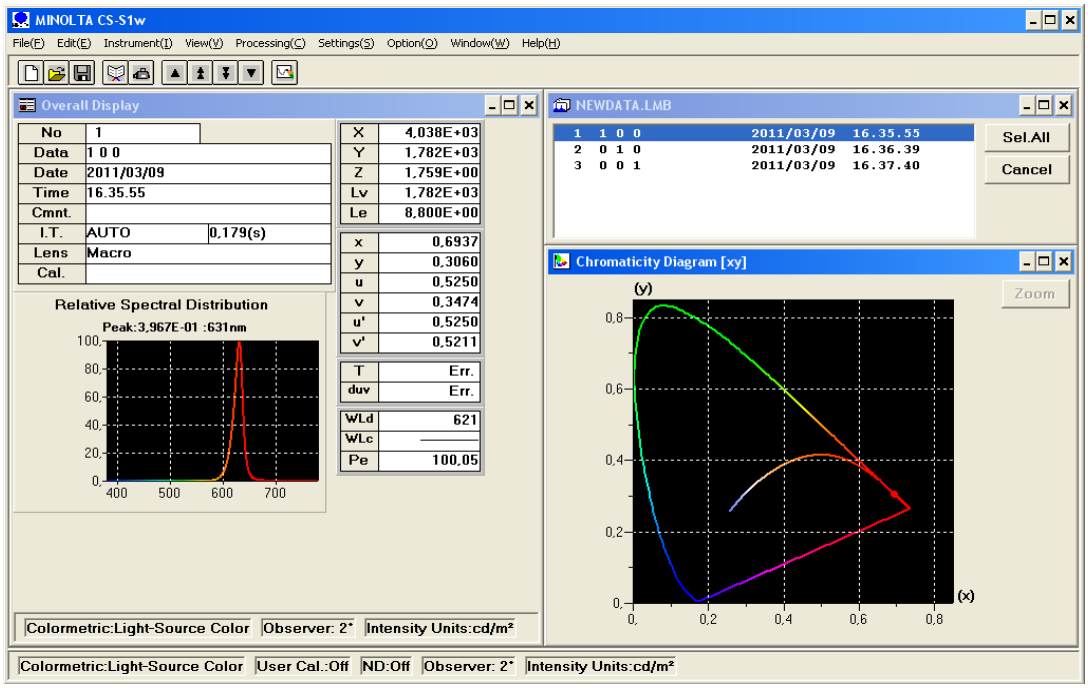


figura 6.3: screenshot dell'acquisizione del valore di luminanza con la terna [1 0 0]

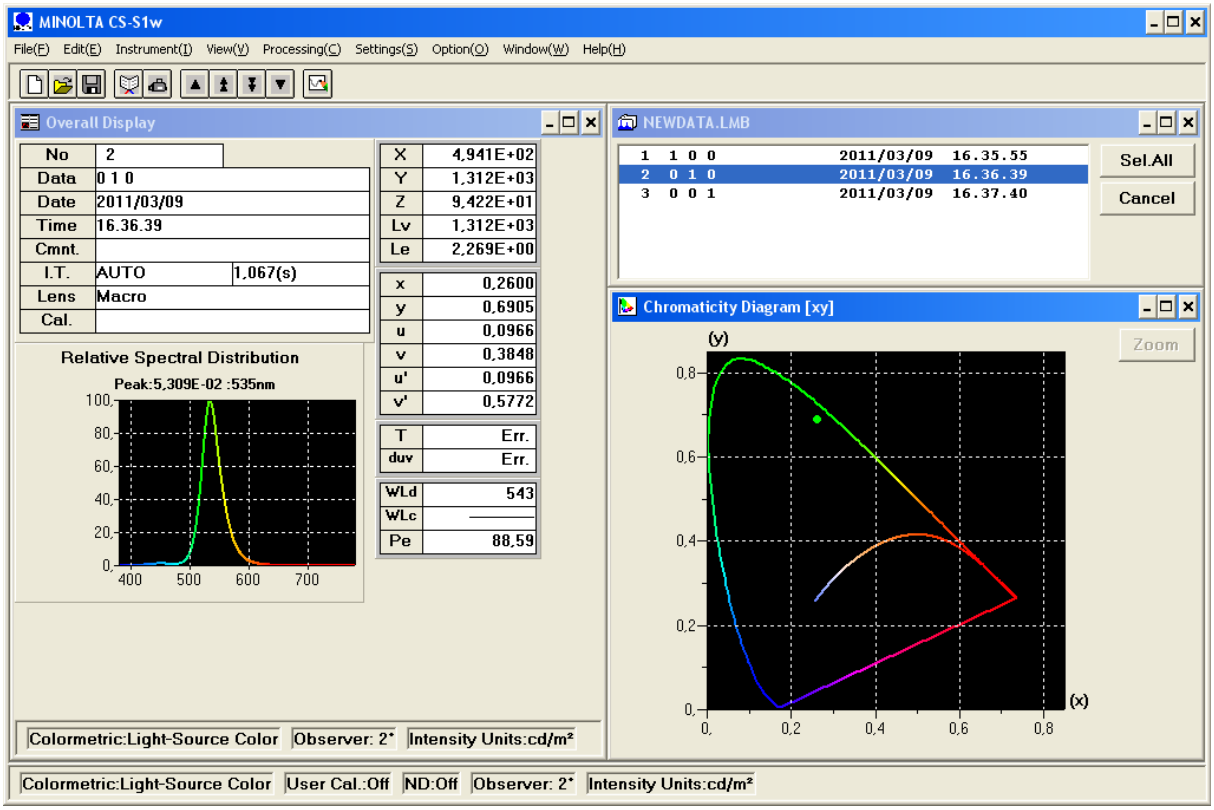


figura 6.4: screenshot dell'acquisizione del valore di luminanza con la terna [0 1 0]

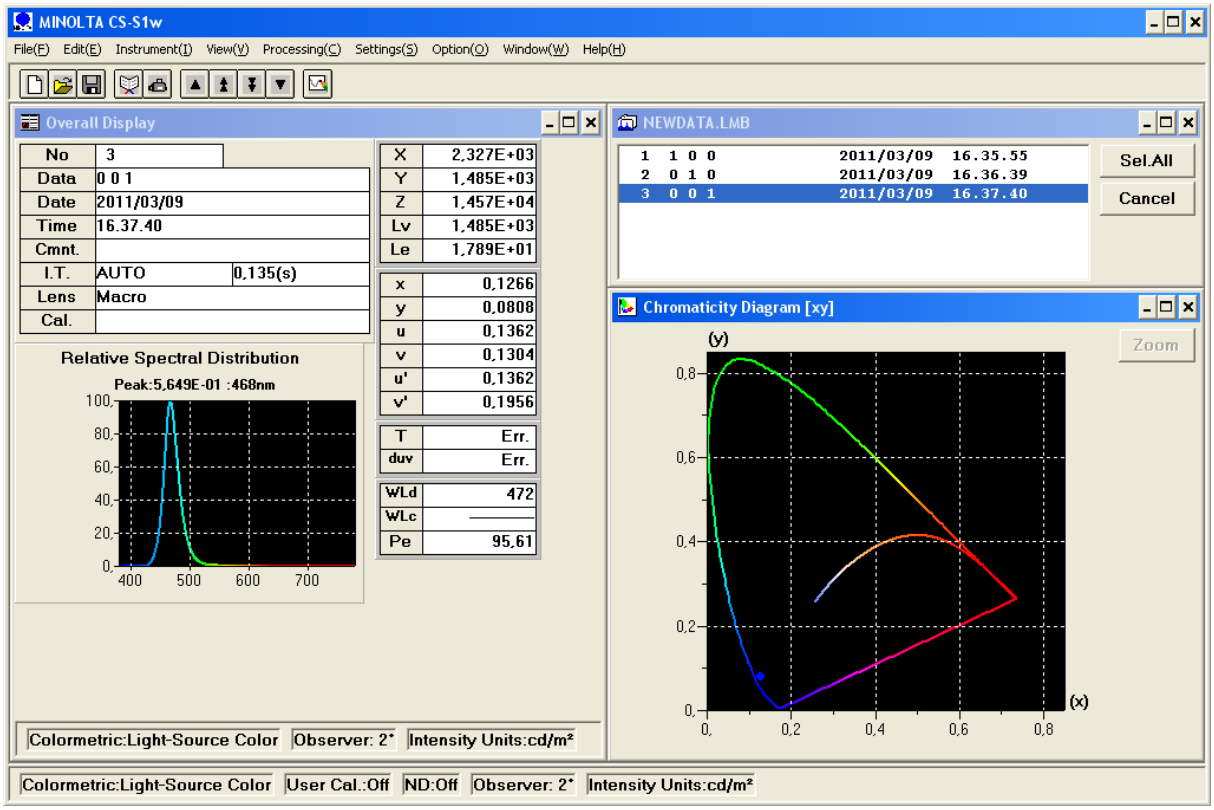


figura 6.5: screenshot dell'acquisizione del valore di luminanza con la terna [0 0 1]

6.2 misure di uniformità delle coordinate cromatiche

Una volta decisa la configurazione finale “diametro del setto / distanza del setto / diametro foro di uscita”, che soddisfaceva le nostre richieste in quanto a valore massimo e uniformità della luminanza sulla superficie di osservazione, si è proceduto alla verifica dell'uniformità delle coordinate cromatiche su tale superficie di osservazione.

Infatti, a causa della configurazione della sorgente, che ha i LED blu posizionati a croce (quindi con disposizione perfettamente simmetrica) e i LED verdi e rossi posizionati sulle due diagonali, c'era il rischio che in corrispondenza di queste diagonali vi fosse un contributo troppo pesante di verde o di rosso rispettivamente.

I punti di misurazione sono rimasti gli stessi delle misure precedenti, ed a variare sono state invece le potenze dei vari canali di colore: sono state utilizzate infatti le terne [1 1 1], [1 0 0], [0 1 0] e [0 0 1].

Per essere sicuri di effettuare la misura sempre nello stesso punto, per ognuno dei punti di misurazione sono state provate le varie terne, per poi passare al punto successivo.

La figura sottostante rappresenta le coordinate cromatiche relative alle varie terne che sono state provate: si può notare che, nonostante ogni terna sia stata provata in quattro punti, le sue coordinate cromatiche praticamente non varino (lo si capisce dal fatto che si vede solo un punto per le quattro misure con ogni colore).

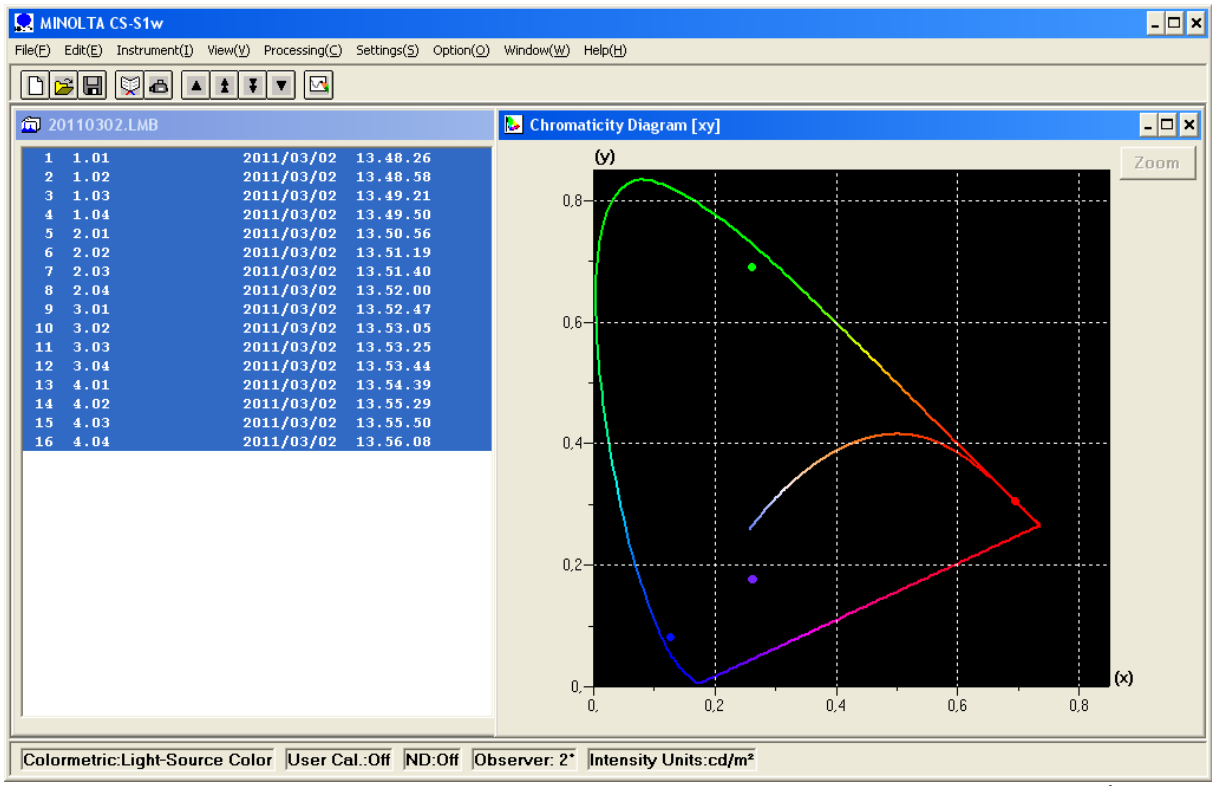


figura 6.6: screenshot delle acquisizioni delle coordinate cromatiche per verificarne l'uniformità

Da queste misure infatti è risultato che le coordinate cromatiche, per la stessa terna di correnti ma nei vari punti di misurazione, rimangono molto stabili: la massima variazione infatti è stata riscontrata per la terna [1 1 1], tra il punto 1 e il punto 4:

- punto 1: x=0,2616 y=0,1763
- punto 4: x=0,2628 y=0,1764

per la coordinata cromatica x la variazione è apprezzabile ma comunque molto contenuta, infatti è dello 0,46%; per la coordinata cromatica y, la variazione è praticamente trascurabile.
(nota: queste misure sono state effettuate prima di scoprire che le coordinate cromatiche subivano delle variazioni con la temperatura: dunque, questa variazione potrebbe essere stata causata da quel fenomeno).

Si è potuto anche vedere che, con i LED a nostra disposizione, si riesce a coprire un gamut discretamente ampio all'interno dello spazio Lxy.

Infine, la “prova del nove” è stata fatta impostando dall'interfaccia grafica gli stessi valori delle coordinate cromatiche ottenute dalle misure: in particolare, i punti più rappresentativi sono i vertici del triangolo RGB; si è visto che, con solo qualche millesimo di scostamento rispetto ai valori che erano stati ottenuti, ci si è trovati esattamente nei vertici del triangolo RGB.
Segno, questo, che anche le trasformazioni matematiche all'interno del programma lavorano con buona precisione e senza errori.

6.3 acquisizione della radianza spettrale dei LED

Come spiegato nella parte relativa al software, in particolare quella relativa alle trasformazioni negli spazi colore, per eseguire la trasformazione da XYZ a RGB bisogna tener conto della sorgente di cui disponiamo.

Per fare questo, è necessario conoscere i valori tristimolo generati dai LED: per ottenerli, si è operato come segue:

Alimentando i LED, un canale per volta alla massima potenza (nell'ordine: rosso, verde, blu), si è eseguita una misura per ogni colore utilizzando lo spettroradiometro, ed acquisendola attraverso il programma dedicato CS-S1w.

Il file contenente la misura è stato salvato in formato .LMT (file spettriRGB.LMT); i dati in esso contenuti vengono letti nella function di Matlab che effettua le trasformazioni, ed utilizzati per la costruzione della matrice di trasformazione da RGB ad XYZ; invertendo questa matrice, partendo dalla terna X,Y,Z necessaria per riprodurre il colore desiderato, si calcolano i valori R,G,B di corrente da imporre ai LED.

Le acquisizioni sono mostrate nelle figure 6.3, 6.4 e 6.5.

6.4 misure delle correnti di alimentazione dei LED

Dopo aver modificato il circuito per permettere la misurazione delle correnti di alimentazione dei LED, la prima misura da fare era sicuramente quella dei valori delle correnti quando si impostava al massimo la potenza richiesta, per verificare se le correnti fossero uguali per i tre canali.

Purtroppo, si è visto che così non era: infatti, i valori medi erano diversi per le correnti dei tre canali, nonostante si stesse richiedendo la piena potenza a tutti i LED indifferentemente:
il canale rosso aveva un valore medio della corrente di 650 mA, il canale verde aveva un valore medio della corrente di 670 mA e il canale blu aveva un valore medio della corrente di circa 600 mA.

Inoltre, dalle acquisizioni si notava un forte rumore: infatti, si avevano variazioni dei valori di corrente di quasi 80 mA picco-picco (più del 10% rispetto al valore medio!).

Sono stati quindi realizzati e collegati dei filtri passa-basso agli ingressi delle schede DAQ, che hanno permesso di migliorare notevolmente la situazione: infatti, dopo la loro inserzione, il valore picco-picco del rumore si è ridotto a 5 mA, quindi meno dell'1% rispetto al valore medio.

Qui di seguito mostriamo le acquisizioni delle correnti dei tre canali: prima le acquisizioni senza filtri passa-basso, e successivamente le acquisizioni con filtri passa-basso.

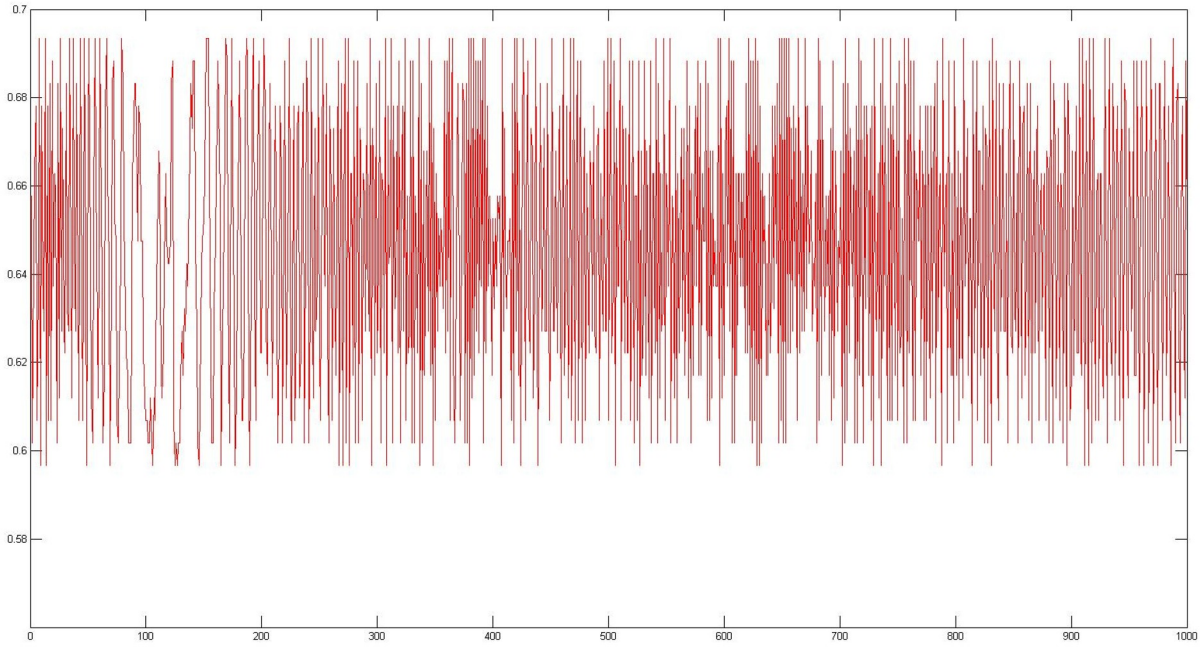


figura 6.7: acquisizione della corrente di alimentazione dei LED rossi, senza filtro passa-basso

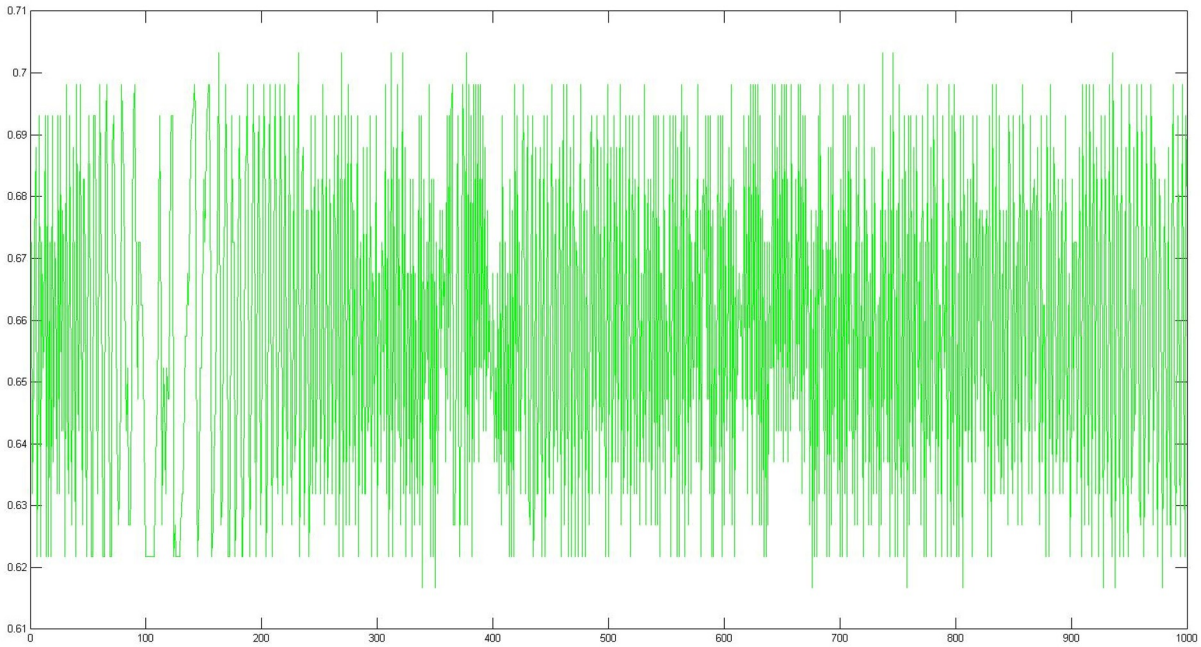


figura 6.8: acquisizione della corrente di alimentazione dei LED verdi, senza filtro passa-basso

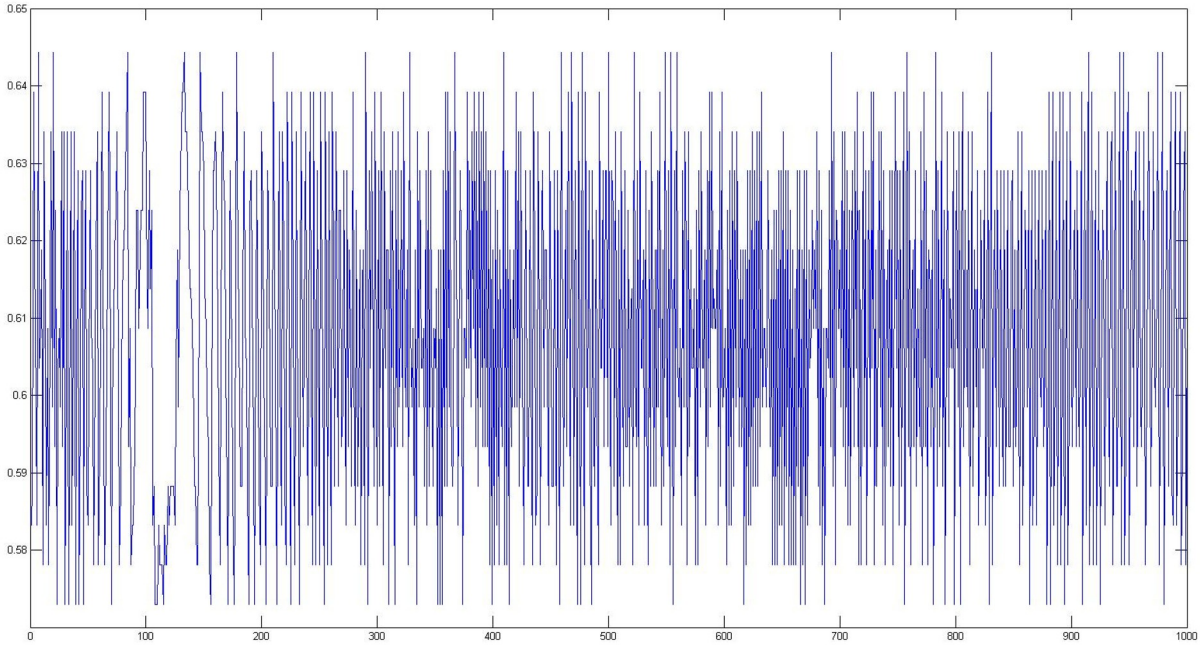


figura 6.9: acquisizione della corrente di alimentazione dei LED blu, senza filtro passa-basso

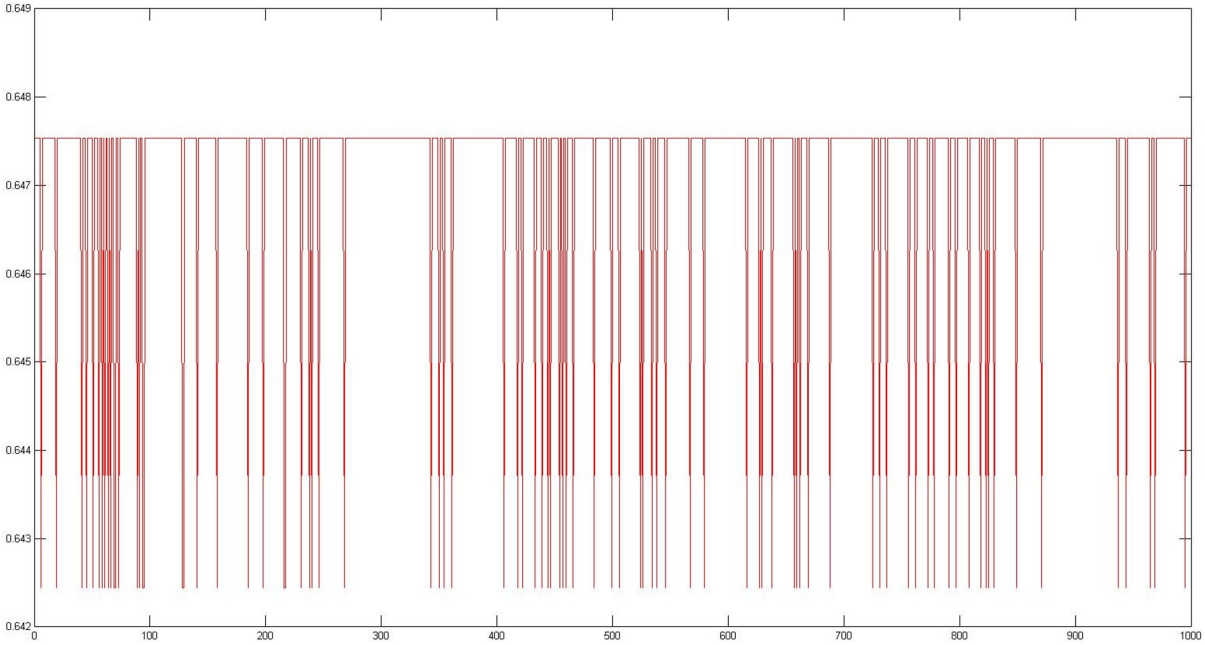


figura 6.10: acquisizione della corrente di alimentazione dei LED rossi, con filtro passa-basso

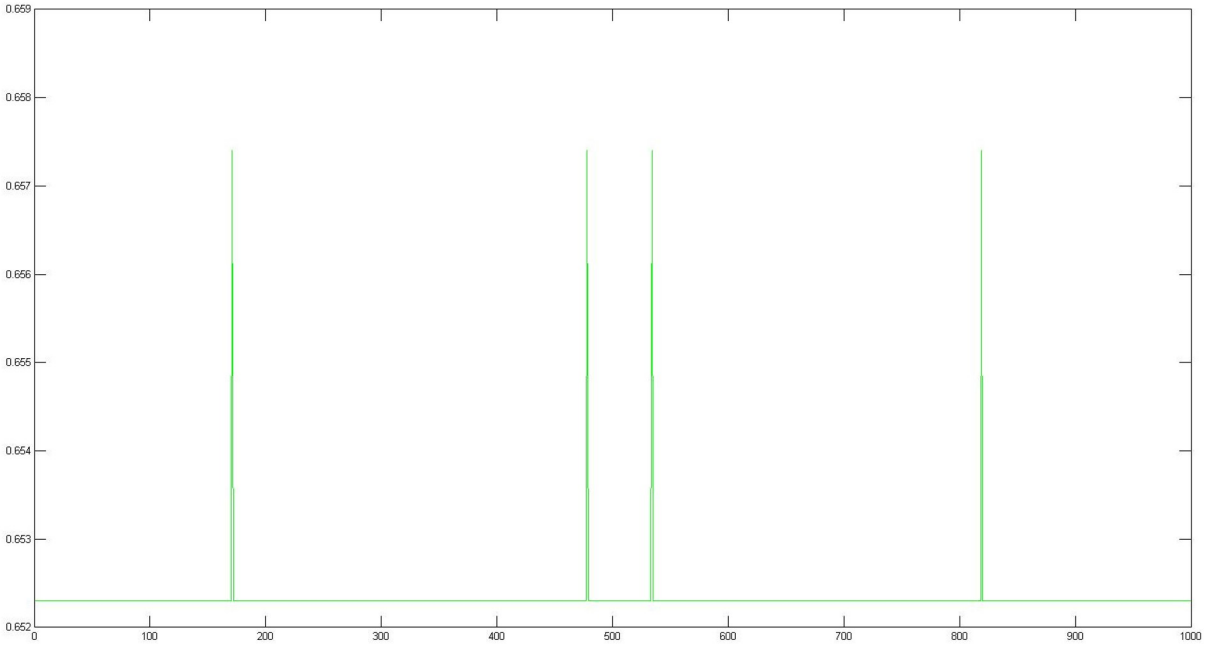


figura 6.11: acquisizione della corrente di alimentazione dei LED verdi, con filtro passa-basso

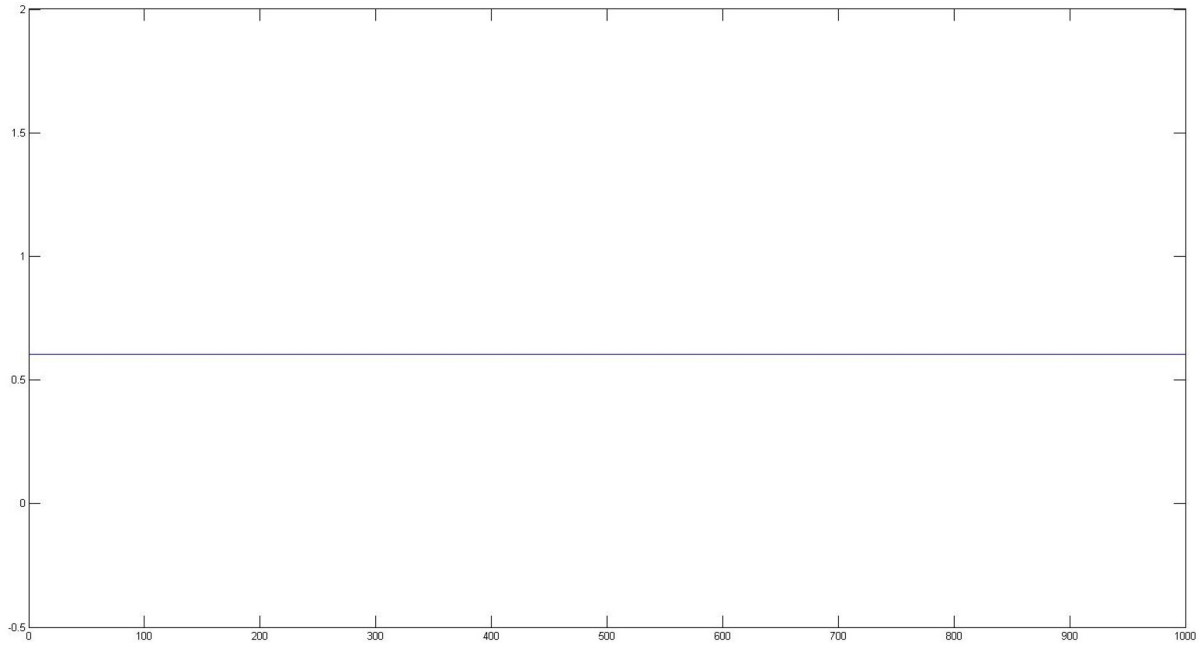


figura 6.12: acquisizione della corrente di alimentazione dei LED blu, con filtro passa-basso

Anche se era stato risolto il problema del rumore, rimaneva sempre il fatto che le correnti di alimentazione dei tre canali non erano uguali, e quindi non avevano lo stesso coefficiente di proporzionalità rispetto al valore di flusso luminoso richiesto.

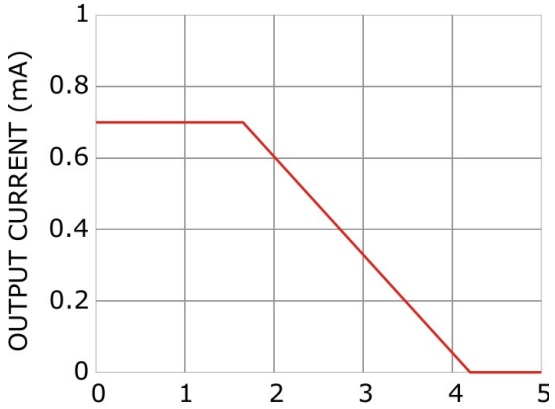
6.5 acquisizione delle caratteristiche degli alimentatori

Dopo aver verificato che le correnti di alimentazione dei LED dei tre canali non erano uguali a piena potenza, si è voluto verificare se il problema dipendesse dagli alimentatori, o da qualche altro componente del sistema.

Si è quindi scritto uno script in Matlab che permettesse di verificare la caratteristica tensione di controllo / corrente di uscita degli alimentatori dimmerabili.

Tramite questo script sono state inviate agli alimentatori tensioni di controllo da 0 a 5 V, con un passo di 5 mV, e per ognuno dei valori di tensione applicati è stata misurata la corrente di alimentazione dei LED.

Sotto vengono mostrate la caratteristica ideale, tratta dal data-sheet, e le caratteristiche misurate dei tre alimentatori (misure effettuate dopo l'inserimento dei filtri passa-basso).



caratteristica ideale degli alimentatori:
tensione di controllo / corrente di uscita

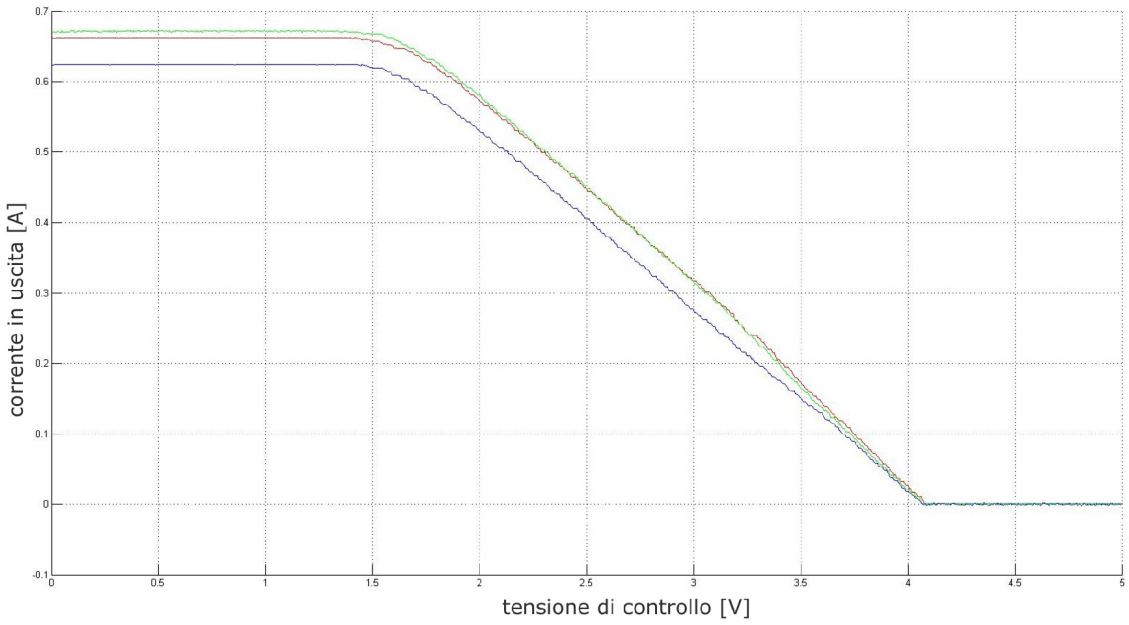


figura 6.13: acquisizione delle caratteristiche reali tensione di controllo / corrente di uscita degli alimentatori, con filtro passa-basso

Non è stato molto confortante vedere che le tre caratteristiche non coincidevano, specialmente quella dell'alimentatore del canale blu.

Soprattutto, si è avuta la conferma che i 700 mA di corrente di uscita dichiarati nel data-sheet sono parecchio ottimistici.

Per questo motivo, si è dovuti ricorrere a un circuito di retroazione, che assicurasse nei limiti del possibile una linearità tra la potenza richiesta e la corrente di alimentazione dei LED.
Con questo circuito di retroazione, per ovviare al problema delle diverse correnti a piena potenza, si è deciso di fare in modo che, richiedendo la massima potenza, vengano erogati 600 mA: in questo modo si è sicuri di poter raggiungere tale valore con tutti i canali.

6.6 acquisizione della relazione corrente/flusso luminoso dei LED

Poiché è noto che la caratteristica corrente / flusso luminoso dei LED non è lineare, ma soggetta a una saturazione avvicinandosi alla corrente massima, un altro passo necessario per “correggere le non linearità del sistema” era quello di creare una look-up table da dare in pasto al Matlab.

Fino ad ora infatti, impostando ad esempio la potenza dei LED al 50% dall'interfaccia grafica, si è fatto in modo che fosse la corrente di alimentazione dei LED ad essere il 50% di quella massima; grazie all'utilizzo di questa look-up table, invece, sarà il flusso luminoso ad essere il 50% di quello massimo, com'è giusto che sia.

Essendo la sorgente già costruita e definitiva, era impensabile mettersi a scollegare i LED per fare delle prove individuali in una sfera integratrice: quindi, supponendo che la relazione tra il flusso luminoso dei LED e la luminanza della superficie di osservazione sia lineare, si è deciso di misurare quest'ultima, utilizzando lo spettroradiometro.

In realtà, è più conveniente misurare la radianza che la luminanza, poiché la radianza esce direttamente come valore dalle misure effettuate con lo spettroradiometro; la luminanza, in fondo, è solamente la radianza pesata con la curva di sensibilità dell'occhio umano.

Quindi, si è scritto uno script in Matlab per ogni canale, che imponesse una corrente di alimentazione dei LED, da zero al massimo, suddividendo il range in 250 intervalli.
Ovviamente, la corrente imposta non è quella con cui vengono realmente alimentati i LED, per cui è stata anche fatta l'acquisizione (sempre come media di 1000 campioni per ogni valore) della reale corrente di alimentazione dei LED.

Utilizzando dei programmi già esistenti, si è misurata la radianza spettrale, ovvero la radianza per ogni lunghezza d'onda: la radianza è stata poi calcolata come l'integrale su tutte le lunghezze d'onda della radianza spettrale.

Avendo acquisito anche le correnti reali di alimentazione dei LED, non si è reso necessario l'utilizzo del circuito di retroazione, che avrebbe allungato ulteriormente i tempi necessari per l'acquisizione.

Comunque, avendo tra le variabili memorizzate anche i valori impressi di corrente, si è potuto notare che la relazione tra questi ultimi e la radianza non è perfetta: in particolar modo per il canale rosso, dove per un certo intervallo di valori si nota un abbassamento della caratteristica.

Qui di seguito mostriamo le relazioni esistenti tra le correnti imposte (che, quando è stata effettuata la prova, erano coincidenti con i flussi richiesti dato che ancora mancavano le look-up tables) e le radianze misurate con lo spettroradiometro.

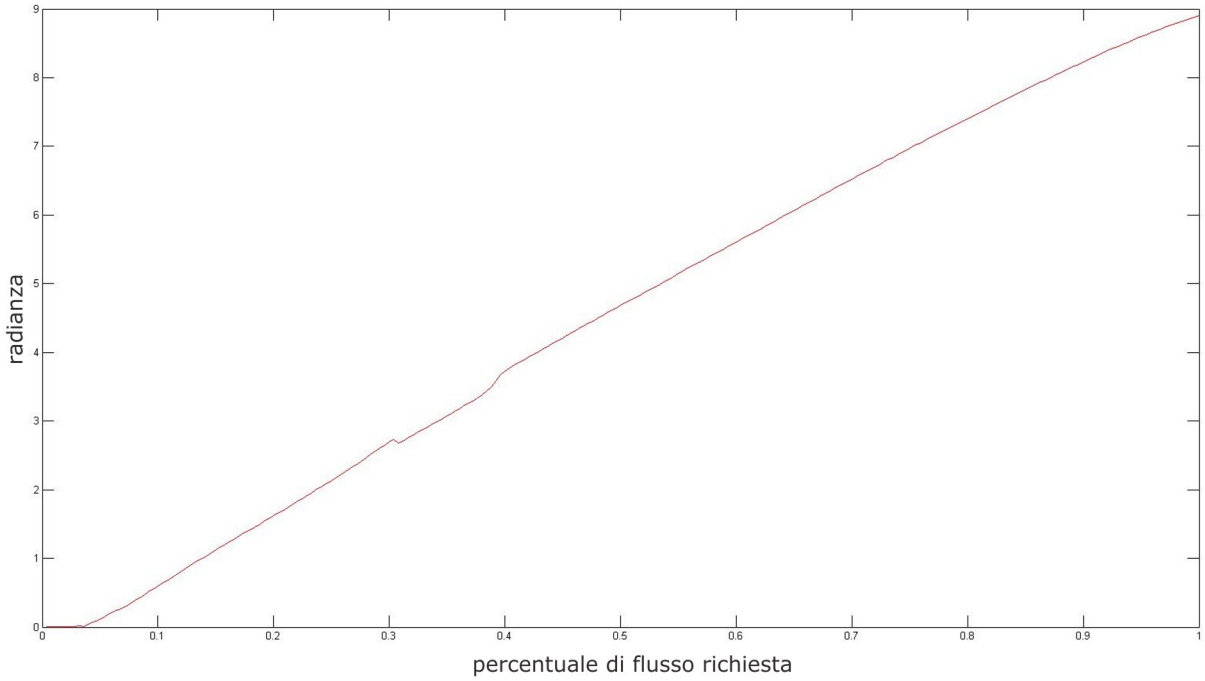


figura 6.14: relazione flusso richiesto / radianza per i LED rossi

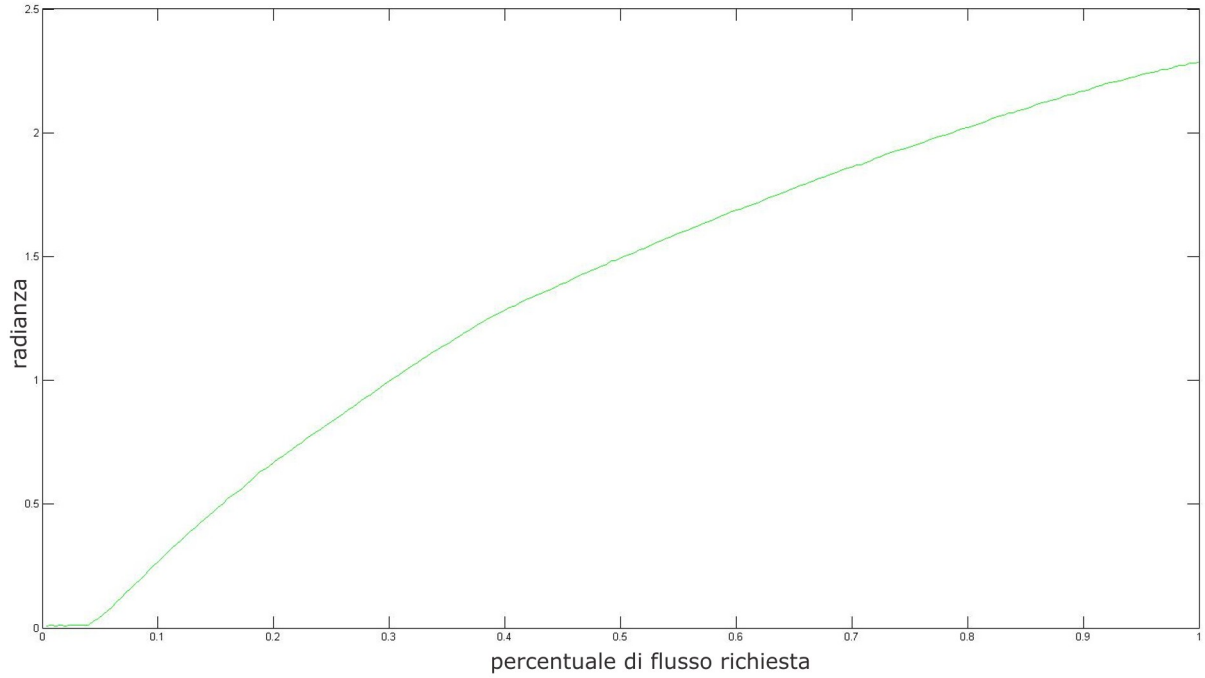


figura 6.15: relazione flusso richiesto / radianza per i LED verdi

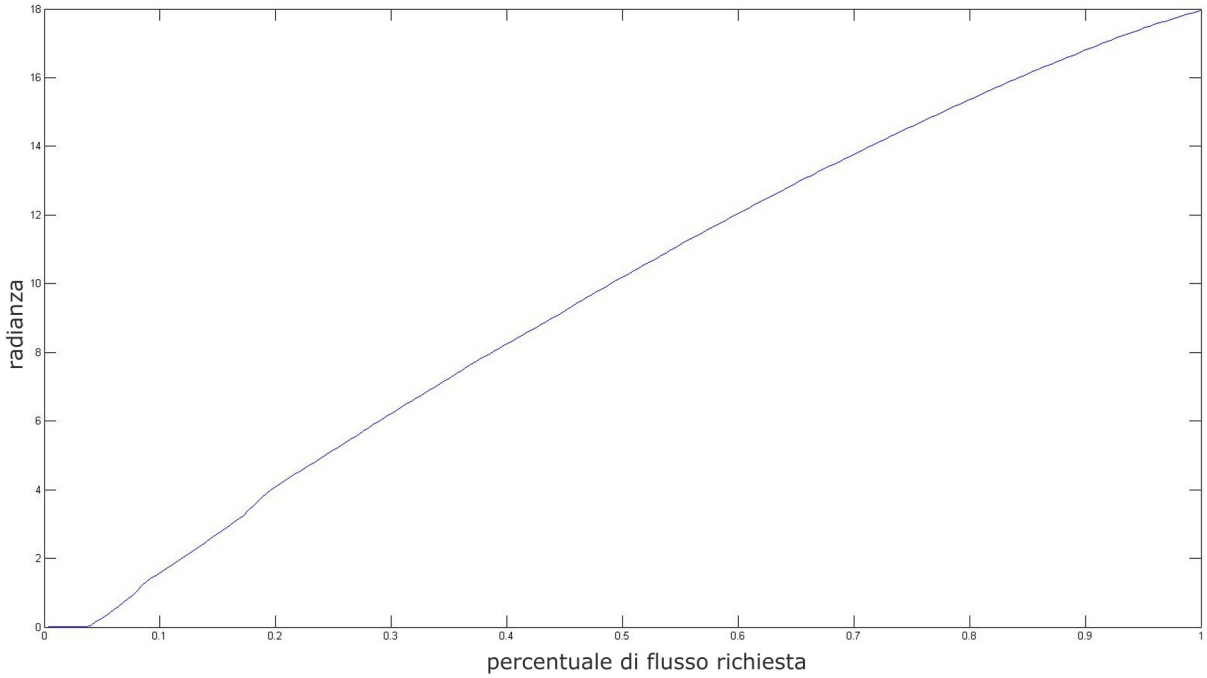


figura 6.16: relazione flusso richiesto / radianza per i LED blu

E' stata effettuata anche una acquisizione per verificare la presenza di un'isteresi nel comportamento dei LED: questa isteresi effettivamente compare, come si può vedere dalla caratteristica qui sotto riportata, ma probabilmente essa è data dall'aumento della temperatura durante la prova, con la conseguente riduzione di flusso luminoso dei LED:

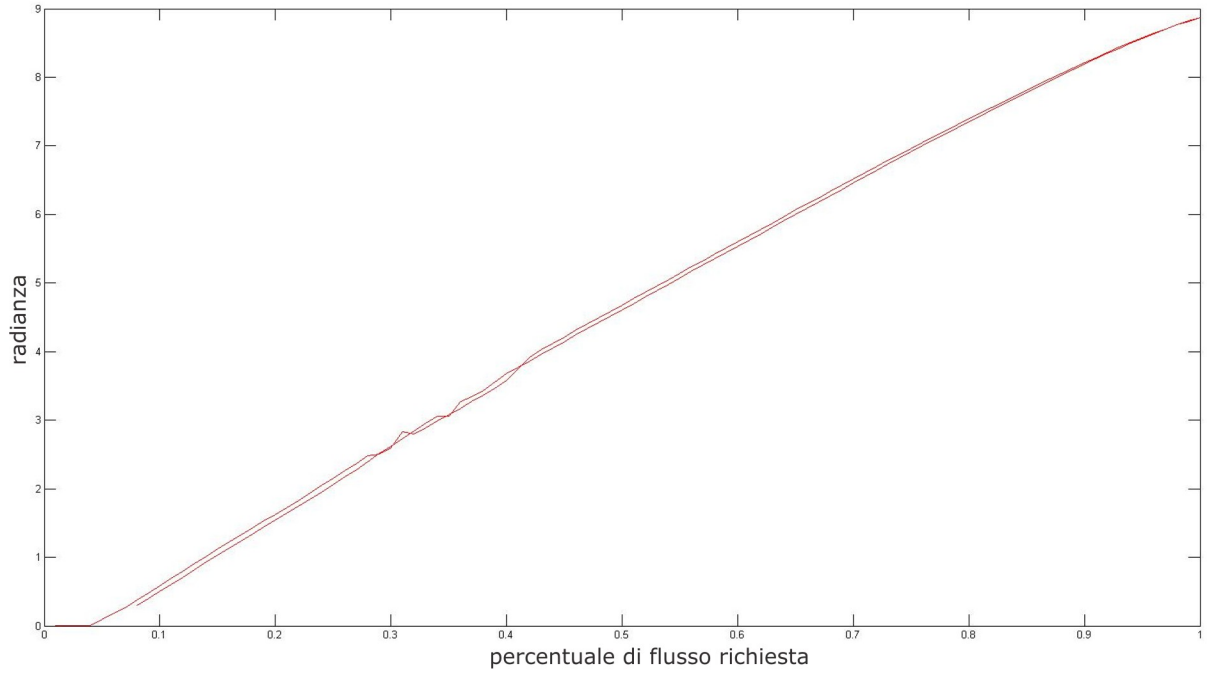


figura 6.17: relazione flusso richiesto / radianza per i LED rossi, effettuando la prova prima da LED spenti aumentando la corrente, e successivamente da LED a piena potenza riducendo la corrente fino a zero

Quando verrà aggiunta la resistenza per mantenere a temperatura costante la sorgente, sarebbe il caso di effettuare nuovamente queste misurazioni, per poter usare delle look-up tables definite a una certa temperatura, pari a quella che verrà mantenuta durante il normale funzionamento.

In ogni caso, dopo aver effettuato le acquisizioni, si è agito in questo modo:

i valori delle correnti misurate e dei flussi sono stati normalizzati (in modo che, quando richiedo flusso=1 ovvero 100%, la corrente erogata sia il 100%);

i valori delle correnti misurate sono poi stati ordinati in ordine crescente, memorizzando gli indici di permutazione in un vettore esterno; usando questo vettore, anche i flussi sono stati ordinati, in modo che ogni flusso corrispondesse nuovamente al valore a cui corrispondeva prima dell'ordinamento;

i vettori delle correnti misurate e dei flussi sono stati copiati all'interno della function *lookup_tables.m*.

All'interno di questa function, a cui vengono passati i valori dei tre flussi richiesti (in percentuale), viene utilizzata la funzione **interp1** la quale, prendendo il vettore dei flussi, il vettore delle correnti e il valore di flusso richiesto, fornisce il valore di corrente corrispondente.

Utilizzando questa funzione si ha il vantaggio che, anche se la relazione flusso-correnti è definita solo a punti, se si cerca di ottenere un valore intermedio tra due valori acquisiti, lo si otterrà come media tra i due valori tra cui è compreso.

Per poter fare funzionare correttamente la funzione interp1 che, come indica il nome, *interpola*, è stato necessario modificare i primi valori dei vettori ponendoli a zero: infatti, se non si effettuava questa operazione, quando nella terna di valori richiesta uno era pari a zero, il programma avrebbe dovuto estrapolare il valore di corrente corrispondente, e questo non poteva essere fatto con questa funzione.

Un'altra piccola modifica ai valori che è stata fatta, stavolta per non avere valori al di fuori dei range [0,1], è stata quella di modificare i primi due valori del vettore delle correnti del canale blu.
Infatti, come spiegato prima, a causa dell'offset che si verifica sulle misure di corrente effettuate con la scheda NI USB-6259, i primi due valori erano risultati negativi (anche se molto vicini allo zero); allora il primo valore è stato posto a zero, e il secondo è stato posto pari alla media tra zero e il terzo valore del vettore.

Comunque, i primi valori dei vettori erano tutti molto piccoli, quindi averli approssimati a zero non introduce problemi: questo è ancor più vero se si pensa che, a quei bassissimi valori, i LED probabilmente sono completamente spenti a causa della loro caratteristica.

6.7 variazione tensione di controllo e coordinate cromatiche con la temperatura

Come detto in precedenza, una volta fatta funzionare in modo corretto e preciso la retroazione, si è notato che, pur riportando ad ogni ciclo le correnti di alimentazione dei LED al valore esatto, la tensione di comando degli alimentatori dimmerabili variava lentamente.

In primo luogo non si capiva a cosa fosse dovuta questa variazione, ma poi si è scoperto che essa dipende dalla temperatura; ed in effetti, si può fare il seguente ragionamento.

I LED utilizzati in questa sorgente hanno una relazione tra temperatura e corrente assorbita quasi lineare e con coefficiente negativo: ovvero, all'aumentare della temperatura diminuisce la corrente nei LED. Quindi, per mantenere sempre allo stesso valore la corrente di alimentazione dei LED, gli alimentatori dimmerabili richiedono in ingresso una tensione di controllo più bassa, in modo da dare in uscita una corrente più alta.

Quindi, la prima volta che si fa funzionare la sorgente, la tensione di controllo necessaria per ottenere (ad esempio) 600 mA ha un certo valore; man mano che la sorgente si scalda, andando verso la temperatura di regime termico, questa tensione diminuisce man mano.

Per verificare questa teoria, sono state effettuate delle acquisizioni dei valori di tensione di controllo partendo da una situazione di sorgente fredda, prima tenendo la ventola alimentata, e successivamente tenendo la ventola spenta, in modo da forzare il riscaldamento della sorgente stessa; questo è quello che si è ottenuto:

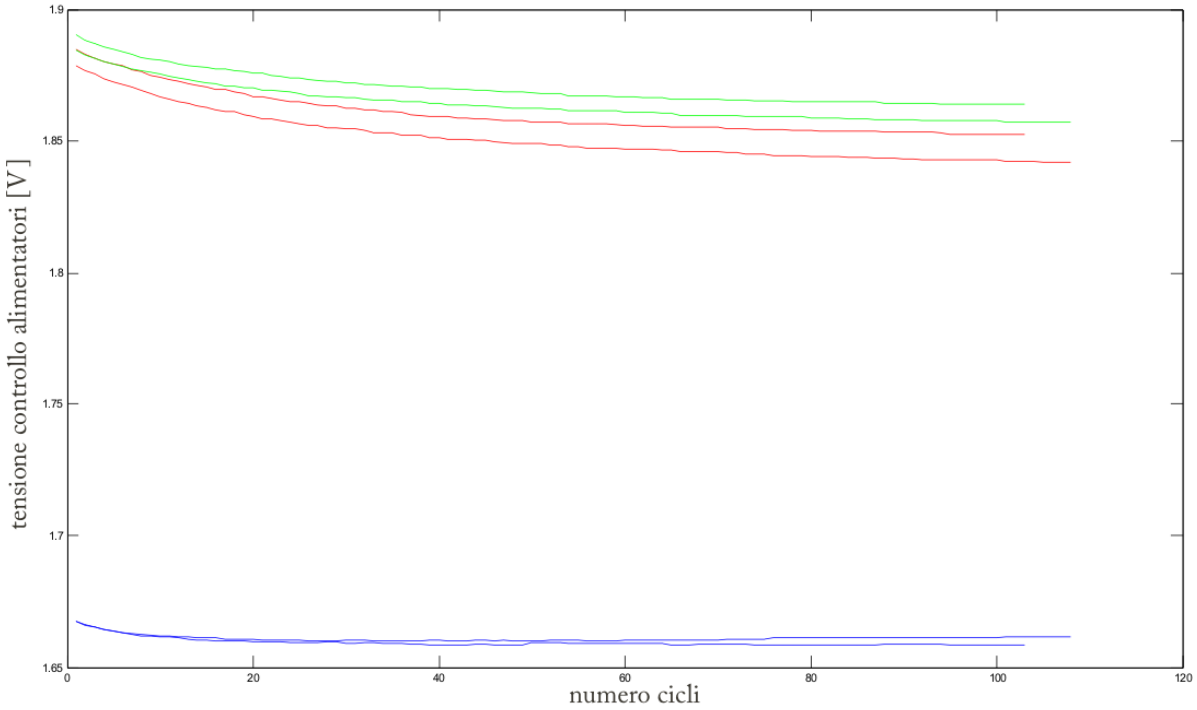


figura 6.18: variazione delle tensioni di controllo degli alimentatori, richiedendo una stessa corrente costante, al variare della temperatura

Questa figura riunisce gli esiti delle due prove: la prima con la ventola accesa, e la seconda con la ventola spenta.
I tratti più bassi, per ciascun alimentatore, rappresentano le tensioni di controllo nella prova con ventola spenta: quindi, si è verificato che più si alza la temperatura e più si abbassa la tensione di controllo per fornire la stessa corrente in uscita.

Ad ogni modo, finché la corrente dei LED viene mantenuta al livello corretto dalla retroazione, il fatto che le tensioni di controllo varino non ci preoccupa più di tanto.

Piuttosto, il fatto a cui prestare più attenzione è la variazione delle coordinate cromatiche con la temperatura.
Infatti, il flusso luminoso dei LED diminuisce all'aumentare della temperatura; questa diminuzione, però, non è uguale per i LED dei tre colori: quindi, la luce ottenuta dalla somma dei tre contributi potrà avere una variazione (in termini di coordinate cromatiche) al variare della temperatura.

Si è quindi effettuata la stessa prova di prima, acquisendo però, invece delle tensioni, le coordinate cromatiche utilizzando lo spettroradiometro. Le coordinate sono state acquisite dopo un ciclo di retroazione, quindi con la certezza che la corrente di alimentazione dei LED fosse quella corretta.

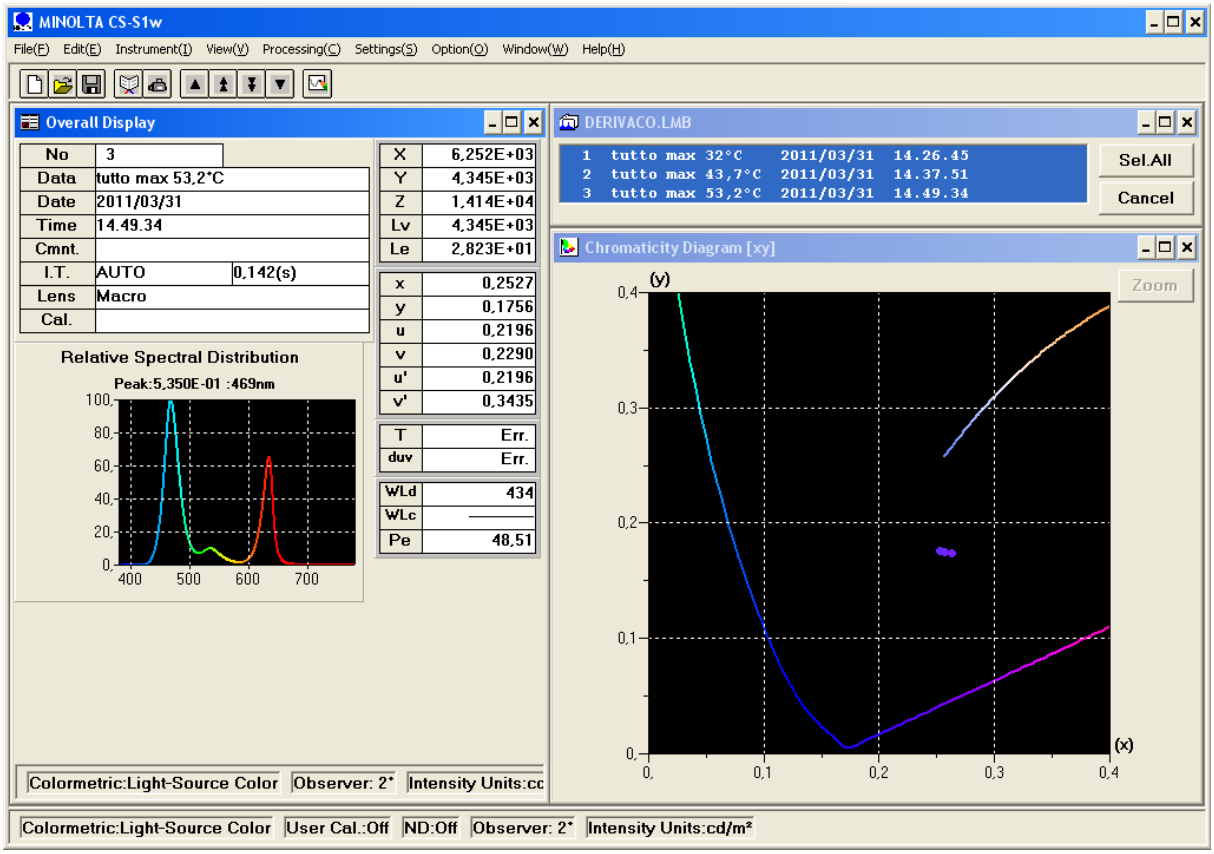


figura 6.19: variazione delle coordinate cromatiche ottenute con tutti i LED a piena potenza, al variare della temperatura

Le misure sulla terna [1 1 1] hanno dato i seguenti risultati:

- terna [1 1 1] @ 33,7°C → x = 0,2537 y = 0,1713
- terna [1 1 1] @ 34,7°C → x = 0,2534 y = 0,1715
- terna [1 1 1] @ 43,7°C → x = 0,2507 y = 0,1723
- terna [1 1 1] @ 56,0°C → x = 0,2437 y = 0,1736

Calcolando le variazioni delle coordinate cromatiche nei vari casi, risulta che esse sono mediamente del -0,12%/°C per la coordinata cromatica **x** e +0,12%/°C per la coordinata cromatica **y**.

Ciò significa che, se la temperatura varia di 10°C, le variazioni di ogni coordinata cromatica possono essere intorno all'1,2%.

Per verificare il contributo di ogni singolo canale alla variazione delle coordinate cromatiche, sono state effettuate altre misure: prima a piastra fredda, con un flusso del 10%, del 50% e del 100%, per ogni colore alternativamente; successivamente ad una temperatura pari circa a quella di regime termico, sempre con un flusso del 10%, del 50% e del 100%, per ogni colore alternativamente.
Purtroppo non si è riusciti ad effettuare tutte le misure alla stessa temperatura esatta, poiché accendendo i LED per fare la misura la temperatura saliva durante la misura; inoltre, per effettuare le prove con sufficiente velocità, non si è atteso che la piastra raggiungesse la stessa temperatura della base dei LED, per cui una variazione di qualche decimo di grado sulla piastra può corrispondere ad una variazione più ampia della temperatura della base dei LED.
La differenza è data dalla dinamica del sistema termico (capacità termica della piastra e resistenze termiche).

I risultati ottenuti dalle misure sono i seguenti:

- terna [0.1, 0, 0]@ 26,7°C → x = 0,6927 y = 0,3070
- terna [0.1, 0, 0] @ 33,8°C → x = 0,6937 y = 0,3061
- terna [0.5, 0, 0] @ 26,7°C → x = 0,6928 y = 0,3069
- terna [0.5, 0, 0] @ 33,4°C → x = 0,6939 y = 0,3059
- terna [1, 0, 0] @ 27,5°C → x = 0,6933 y = 0,3063
- terna [1, 0, 0] @ 33,6°C → x = 0,6942 y = 0,3055

- terna [0, 0.1, 0] @ 26,7°C → x = 0,2886 y = 0,6890
- terna [0, 0.1, 0] @ 33,8°C → x = 0,2909 y = 0,6870
- terna [0, 0.5, 0] @ 26,9°C → x = 0,2719 y = 0,6905
- terna [0, 0.5, 0] @ 33,4°C → x = 0,2737 y = 0,6897
- terna [0, 1, 0] @ 27,5°C → x = 0,2611 y = 0,6907
- terna [0, 1, 0] @ 33,4°C → x = 0,2627 y = 0,6901

- terna [0, 0, 0.1] @ 26,8°C → x = 0,1213 y = 0,0888
- terna [0, 0, 0.1] @ 33,4°C → x = 0,1211 y = 0,0898
- terna [0, 0, 0.5] @ 26,8°C → x = 0,1239 y = 0,0850
- terna [0, 0, 0.5] @ 33,4°C → x = 0,1236 y = 0,0863
- terna [0, 0, 1] @ 27,6°C → x = 0,1270 y = 0,0805
- terna [0, 0, 1] @ 33,7°C → x = 0,1268 y = 0,0819

Per il canale rosso, la massima sensibilità alla variazione di temperatura si è registrata per la terna [0.5, 0, 0], per la quale le variazioni sono:

per la coordinata cromatica x: 0,02369 % / °C;

per la coordinata cromatica y: -0,04879 % / °C.

variazione totale coordinate cromatiche: 0,046 % / °C

Per il canale verde, la massima sensibilità alla variazione di temperatura si è registrata per la terna [0, 0.1, 0], per la quale le variazioni sono:

per la coordinata cromatica x: 0,11225 % / °C;

per la coordinata cromatica y: -0,04100 % / °C.

variazione totale coordinate cromatiche: 0,12 % / °C

Per il canale blu, la massima sensibilità alla variazione di temperatura si è registrata per la terna [0, 0, 1], per la quale le variazioni sono:

per la coordinata cromatica x: -0,02586 % / °C;

per la coordinata cromatica y: 0,2581 % / °C.

variazione totale coordinate cromatiche: 0,286 % / °C

Si può quindi intuire che la responsabilità sulla variazione delle coordinate cromatiche globali non sia di un solo canale, e che dipenda anche dalla percentuale di flusso richiesta ad ogni canale.

Si è notato che, variando il flusso dei LED, anche un solo canale per volta, le coordinate cromatiche hanno una leggera variazione, pari al massimo allo 0,2%:

Nota: queste acquisizioni sono state effettuate utilizzando gli ingressi analogici della scheda NI USB-6259 in modalità Single Ended; solo successivamente ci si è accorti che, usandola in modalità differenziale e adeguando il valore della resistenza nel calcolo, le temperature assumevano valori più realistici.
Ciò non toglie che il fenomeno sia presente e sia quindi degno di attenzioni.

Poiché una variazione di qualche punto percentuale delle coordinate cromatiche può essere distinguibile dall'occhio umano, si consiglia sempre di dare il tempo alla sorgente di raggiungere la temperatura di regime termico, prima di effettuare qualsivoglia test.

Per fare in modo che la sorgente raggiunga tale temperatura di regime termico, è sufficiente accendere l'alimentatore stabilizzato che alimenta il tutto: finché non si fa partire il programma principale, i LED rimangono accesi alla massima potenza, e anche la ventola viene alimentata alla sua tensione nominale.

In tali condizioni, si è stimato che il tempo necessario per raggiungere il regime termico sia di una ventina di minuti.

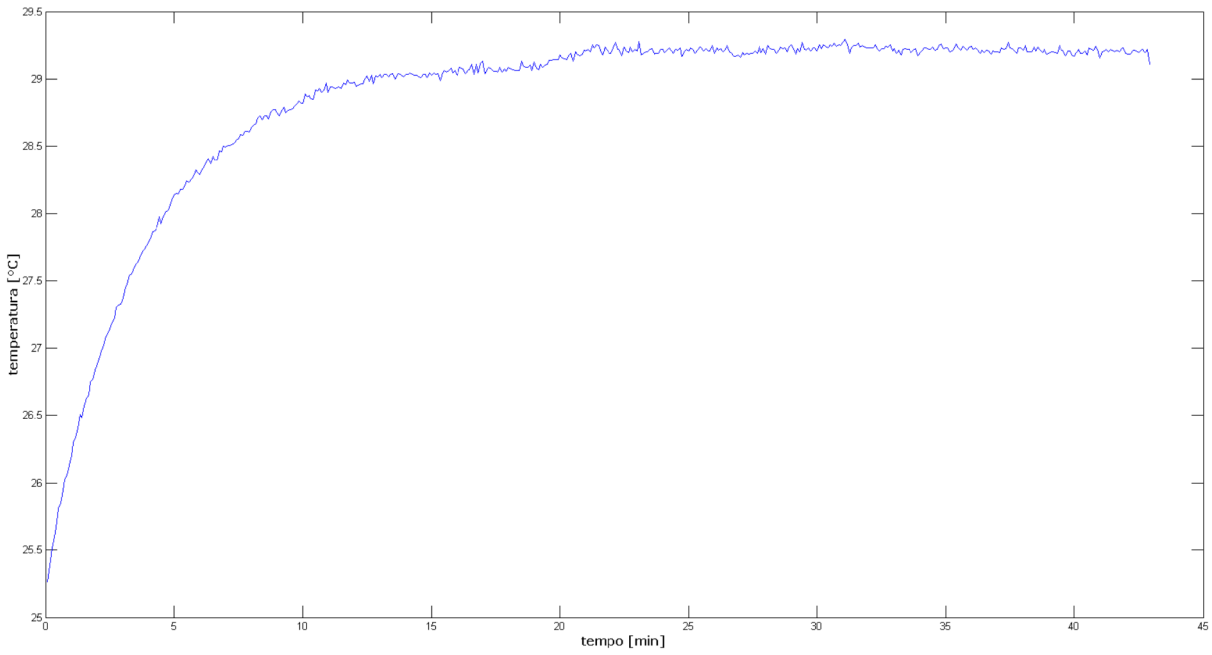


figura 6.20: andamento della temperatura in funzione del tempo, con LED completamente accesi e ventola alimentata

Per accelerare il raggiungimento del regime termico, si può togliere l'alimentazione alla ventola, mantenendo accesi i LED: questo garantisce che in circa 5 minuti si raggiunga il regime termico, mentre lasciando passare più tempo si raggiungono temperature superiori, fino ad arrivare a circa 58°C.

Inoltre, poiché al variare della potenza richiesta ai LED la temperatura può variare, sarebbe necessario introdurre un elemento riscaldante, da alimentare in base al valore misurato della temperatura.

6.8 verifica delle coordinate cromatiche realizzate

Prima di concludere il lavoro, era necessario effettuare delle misure per verificare che, quando viene richiesta una certa coordinata cromatica dall'interfaccia grafica, questa venga ottenuta con una buona precisione.

A questo scopo, si è deciso di utilizzare l'interfaccia grafica lavorando in Lxy per impostare le coordinate cromatiche, e una volta concluso il ciclo di retroazione effettuare una misura con lo spettroradiometro.

Dapprima sono state effettuate 30 misure: tre livelli di luminanza (98%, 50% e 2%) per ognuno di 10 punti "importanti" all'interno dello spazio Lxy.

I dieci punti sono stati così scelti:

i tre vertici del triangolo RGB:

	Rosso:	x= 0,6935	y=0,3060
	Verde:	x=0,2600	y=0,6904
	Blu:	x=0,1266	y=0,0808
il punto	Bianco	x=0,3333	y=0,3333

tre punti a metà strada tra il punto bianco e i vertici del triangolo RGB:

semi-Rosso:	x=0,5124	y=0,3197
semi-Verde:	x=0,2967	y=0,5119
semi-Blu:	x=0,2300	y=0,2071

e infine, tre colori complementari:

Porpora:	x=0,4	y=0,2
Azzurro:	x=0,2	y=0,33
Giallo:	x=0,45	y=0,5

I tre valori di luminanza per cui è stata effettuata la prova per ogni punto sono stati:

98% (L=1274), 50% (L=650), 2% (L=30) (in realtà il 2% corrisponderebbe a L=26, ma più si abbassa il valore della luminanza e più fatica fa la retroazione a convergere).

Successivamente, sono state effettuate delle nuove acquisizioni, a più livelli di luminanza:
98% (L=1274), 50% (L=650), 25% (L=325), L=150, L=100, L=50, L=30.

Dalle acquisizioni eseguite, si è notato che per luminanze pari al 98% e al 50% di quella massima, le coordinate cromatiche restavano abbastanza stabili e si discostavano dal valore richiesto, come si può vedere dagli esempi riportati di seguito.
Per valori inferiori di luminanza, quasi tutti i colori risentono di una variazione non trascurabile (fino anche al 25%, calcolata come distanza sul piano xy tra il punto richiesto e quello ottenuto) delle coordinate cromatiche: proseguendo, ne spiegheremo anche la motivazione.

Rosso (coordinate cromatiche impostate: x=0,6935 y=0,3060).

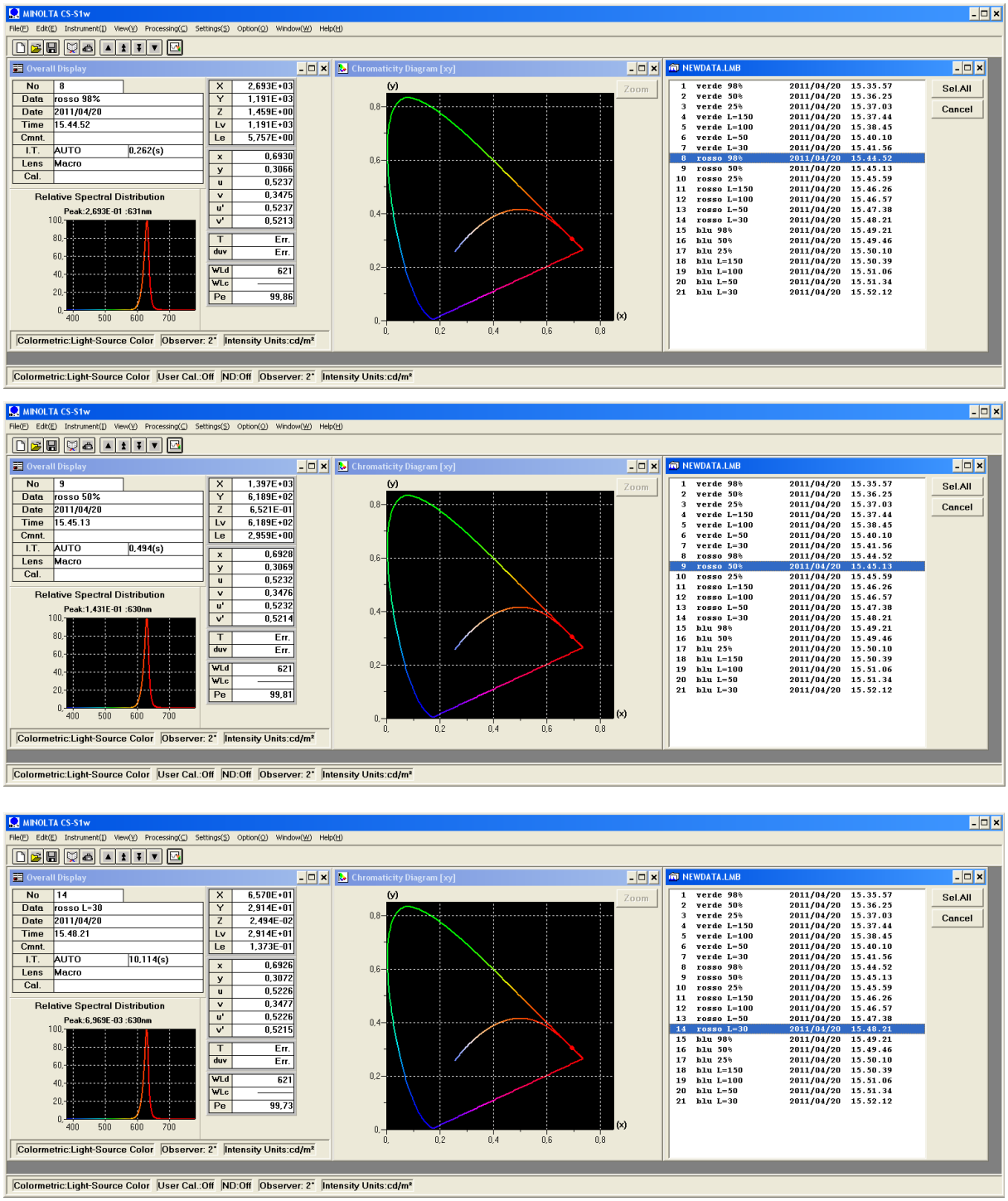


figura 6.21: coordinate cromatiche ottenute per il colore rosso, richiedendo luminanze del 98%, 50% e 2%

Variazioni relative (assolute) rispetto alle coordinate cromatiche richieste:

Luminanza 98%: Δx = -0,07% (-0,0005) Δy = 0,2% (0,0006) → variazione relativa tot. = 0, 21%

Luminanza 50%: Δx = -0,1% (-0,0007) Δy = 0,29% (0,0009) → variazione relativa tot. = 0, 31%

Luminanza 2%: Δx = -0,13% (-0,0009) Δy = 0,39% (0,0012) → variazione relativa tot. = 0, 41%

Blu (coordinate cromatiche impostate: x=0,1266 y=0,0808).

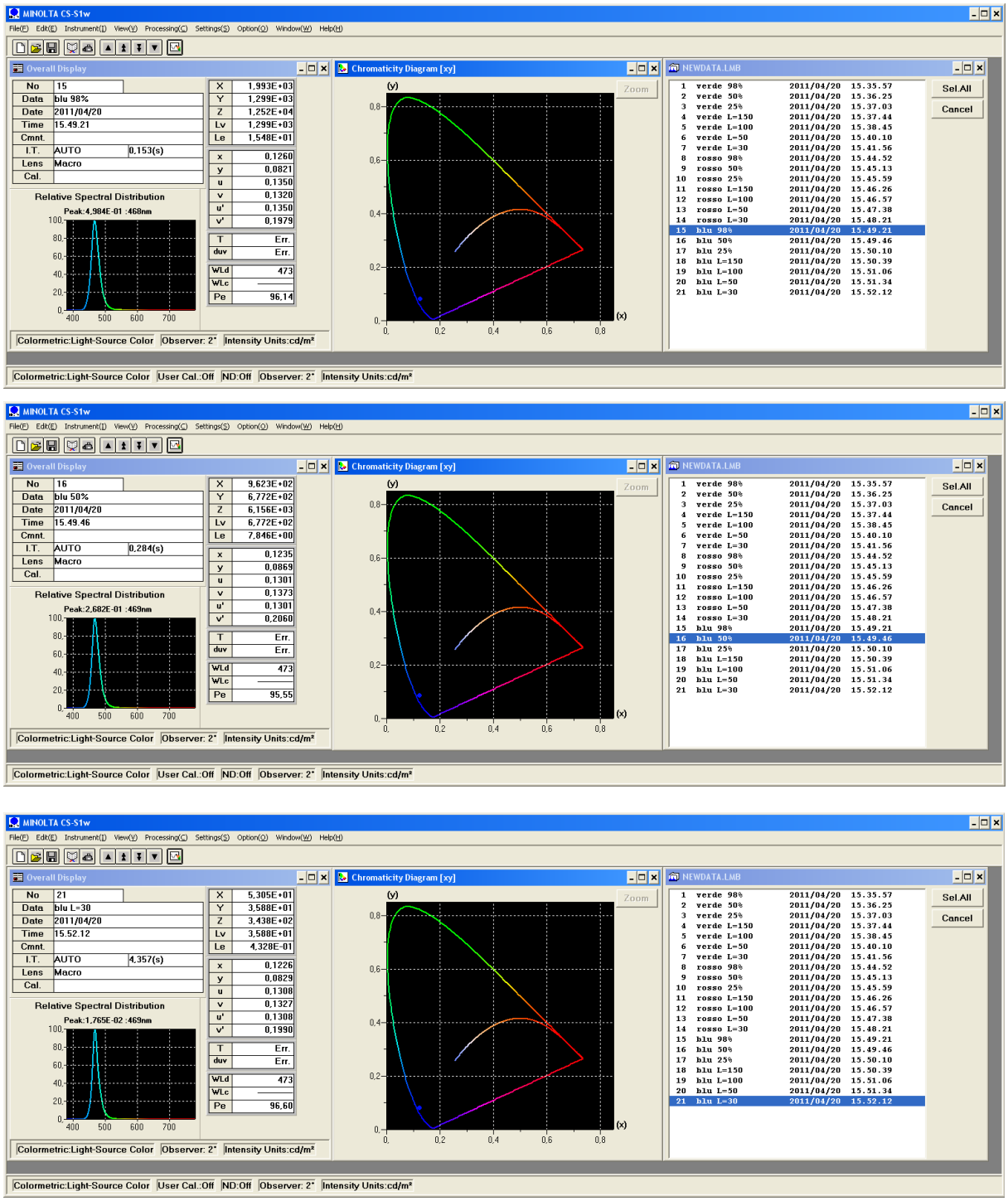


figura 6.22: coordinate cromatiche ottenute per il colore blu, richiedendo luminanze del 98%, 50% e 2%

Variazioni rispetto alle coordinate cromatiche richieste:

Luminanza 98%: Δx = -0,47% (-0,0006) Δy = 1,61% (0,0013) → variazione relativa tot. = 1,68%

Luminanza 50%: Δx = -2,51% (-0,0031) Δy = 7,5% (0,0061) → variazione relativa tot. = 7,9%

Luminanza 2%: Δx = -3,26% (-0,0040) Δy = 2,6% (0,0021) → variazione relativa tot. = 4,17%

Gli alti valori relativi per le variazioni, comunque, sono dati dal valore piccolo delle coordinate cromatiche: in termini assoluti, la variazione è di qualche centesimo.

Verde (coordinate cromatiche impostate: x=0,2600 y=0,6904).

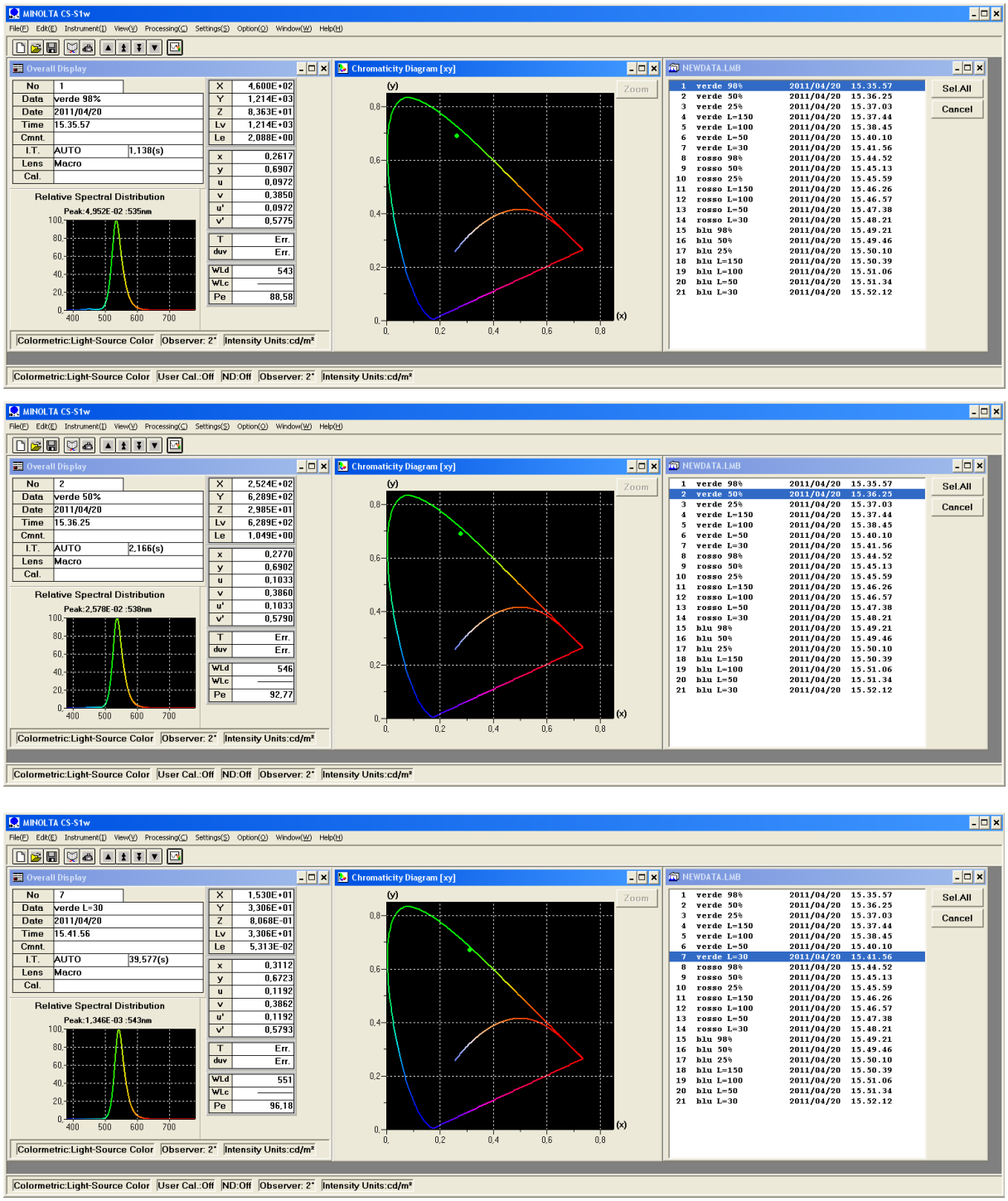


figura 6.23: coordinate cromatiche ottenute per il colore verde, richiedendo luminanze del 98%, 50% e 2%

Variazioni rispetto alle coordinate cromatiche richieste:

Luminanza 98%: Δx = 0,65% (0,0017) Δy = 0,04% (0,0003) → variazione relativa tot. = 0, 65%

Luminanza 50%: Δx = 6,5% (0,0170) Δy = -0,03% (-0,0002) → variazione relativa tot. = 6,5%

Luminanza 2%: Δx = 19,7% (0,0512) Δy = -2,7% (-0,0181) → variazione relativa tot. = 19,9%

Per il verde, la variazione per bassi valori di luminanza diventa addirittura inaccettabile; si è notata anche una variazione di 8 nm della lunghezza d'onda alla quale viene emesso il picco.

Nonostante il blu e il rosso non abbiano apprezzabili variazioni di coordinate cromatiche al variare della luminanza richiesta, la variazione della luminanza reale non è esattamente pari a quella richiesta.

Infatti, diminuendo la luminanza richiesta con il colore **rosso**, il valore di luminanza che si ottiene è sempre inferiore a quello richiesto.

Infatti, richiedendo una luminanza L=1274, si ottengono 1191 [cd/m^2];

richiedendo una luminanza L=650, si ottengono 618,9 [cd/m^2];

richiedendo una luminanza L=325, si ottengono 306 [cd/m^2];

richiedendo una luminanza L=150, si ottengono 143,2 [cd/m^2];

richiedendo una luminanza L=100, si ottengono 95,3 [cd/m^2];

richiedendo una luminanza L=50, si ottengono 52,6 [cd/m^2];

richiedendo una luminanza L=30, si ottengono 29,1 [cd/m^2];

Diminuendo la luminanza richiesta con il colore **blu**, invece, la luminanza che si ottiene è sempre leggermente superiore rispetto al valore richiesto.

Infatti, richiedendo una luminanza L=1274, si ottengono 1299 [cd/m^2];

richiedendo una luminanza L=650, si ottengono 677,2 [cd/m^2];

richiedendo una luminanza L=325, si ottengono 342,7 [cd/m^2];

richiedendo una luminanza L=150, si ottengono 162 [cd/m^2];

richiedendo una luminanza L=100, si ottengono 98,4 [cd/m^2];

richiedendo una luminanza L=50, si ottengono 54,4 [cd/m^2];

richiedendo una luminanza L=30, si ottengono 35,9 [cd/m^2];

Questo scostamento tra luminanze richieste e luminanze ottenute si traduce in una variazione di coordinate cromatiche, al variare della luminanza richiesta, per il colore porpora, che viene ottenuto miscelando il rosso e il blu (teoricamente, sempre nella stessa proporzione; in pratica, questa proporzione cambia al variare della luminanza richiesta).

Nelle prossime pagine mostreremo le variazioni subite dalle coordinate cromatiche del colore porpora al variare della luminanza richiesta, specificando per iscritto solo quella di maggiore entità, che si verifica con una luminanza richiesta L=30, pari a circa il 2% della massima luminanza ottenibile.

Successivamente, mostreremo anche le variazioni delle coordinate cromatiche degli altri due colori complementari, ovvero l'azzurro e il giallo, sempre specificando per iscritto solo la variazione che si verifica richiedendo L=30, poiché è sempre quella di maggiore entità in termini relativi.

Porpora (coordinate cromatiche impostate: x=0,4 y=0,2)

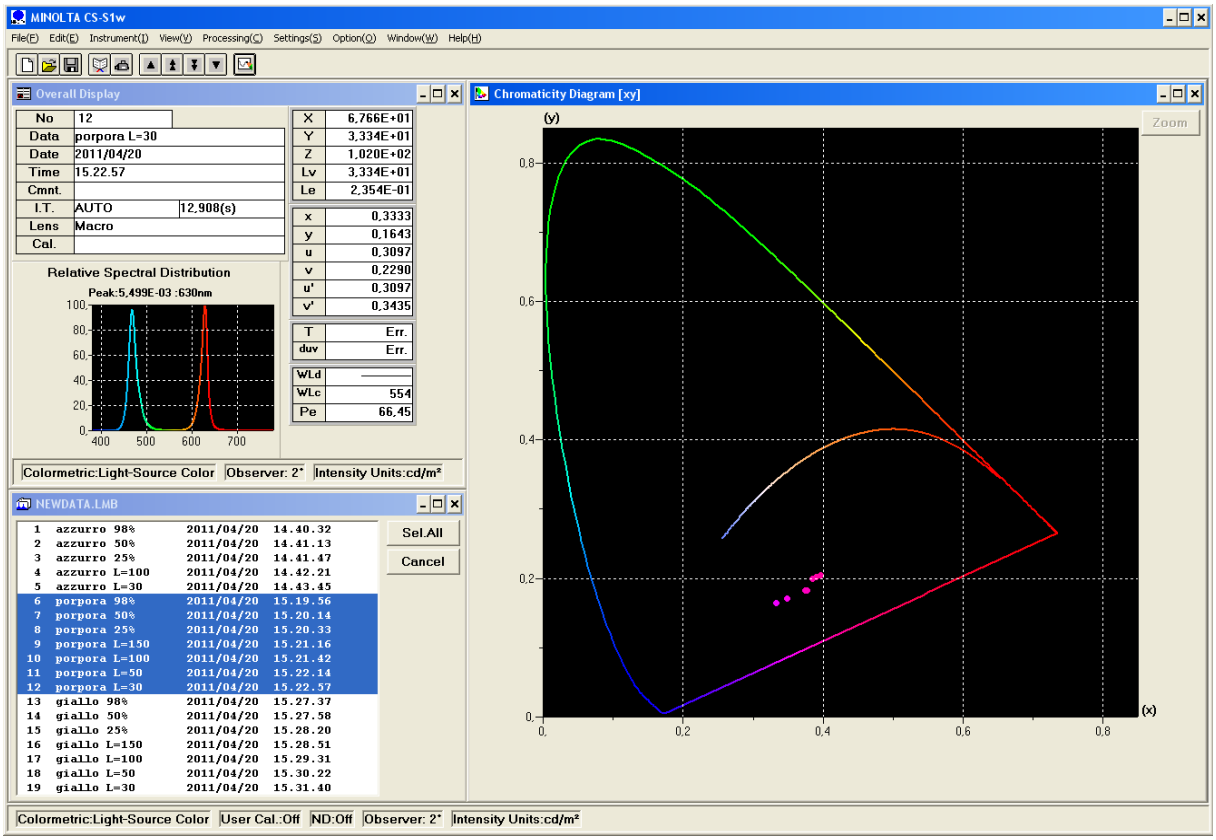


figura 6.24: variazione delle coordinate cromatiche per il colore porpora al variare della luminanza richiesta

Richiedendo una luminanza L=30, pari a circa il 2% di quella massima ottenibile, le coordinate cromatiche ottenute sono state x=0,333 e y=0,1643.

Quindi le variazioni assolute sono $\Delta x = -0,067$ e $\Delta y = -0,0357$
e quelle relative rispetto al valore impostato, quindi, dell' 16,75% per la x e 17,85% per la y;

in totale quindi, la variazione, calcolata come distanza nel piano xy, è pari al 24,5%.

Analoghe variazioni di coordinate cromatiche si possono riscontrare con gli altri due colori complementari, ovvero con l'azzurro (verde+blu) e con il giallo (verde+rosso), e sono dovute soprattutto alla variazione di coordinate cromatiche del verde.

Azzurro (coordinate cromatiche impostate: x=0,2 y=0,33)

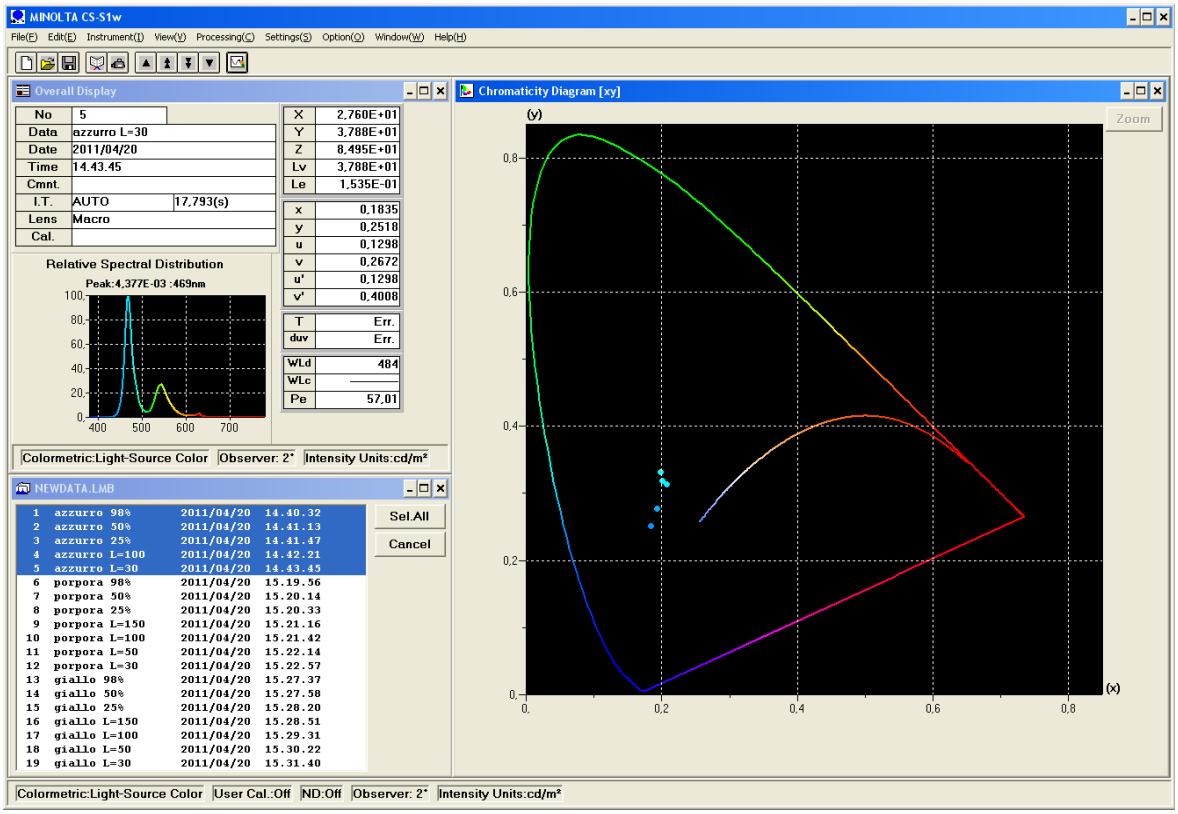


figura 6.25: variazione delle coordinate cromatiche per il colore azzurro al variare della luminanza richiesta

Richiedendo una luminanza L=30, pari a circa il 2% di quella massima ottenibile, le coordinate cromatiche ottenute sono state x=0,1835 e y=0,2518.

Quindi le variazioni assolute sono $\Delta x = -0,0165$ e $\Delta y = -0,0782$
e quelle relative rispetto al valore impostato, quindi, dell' 8,25% per la x e 23,7% per la y;

in totale quindi, la variazione, calcolata come distanza nel piano xy, è pari al 25,1%.

Giallo (coordinate cromatiche impostate: x=0,45 y=0,5)

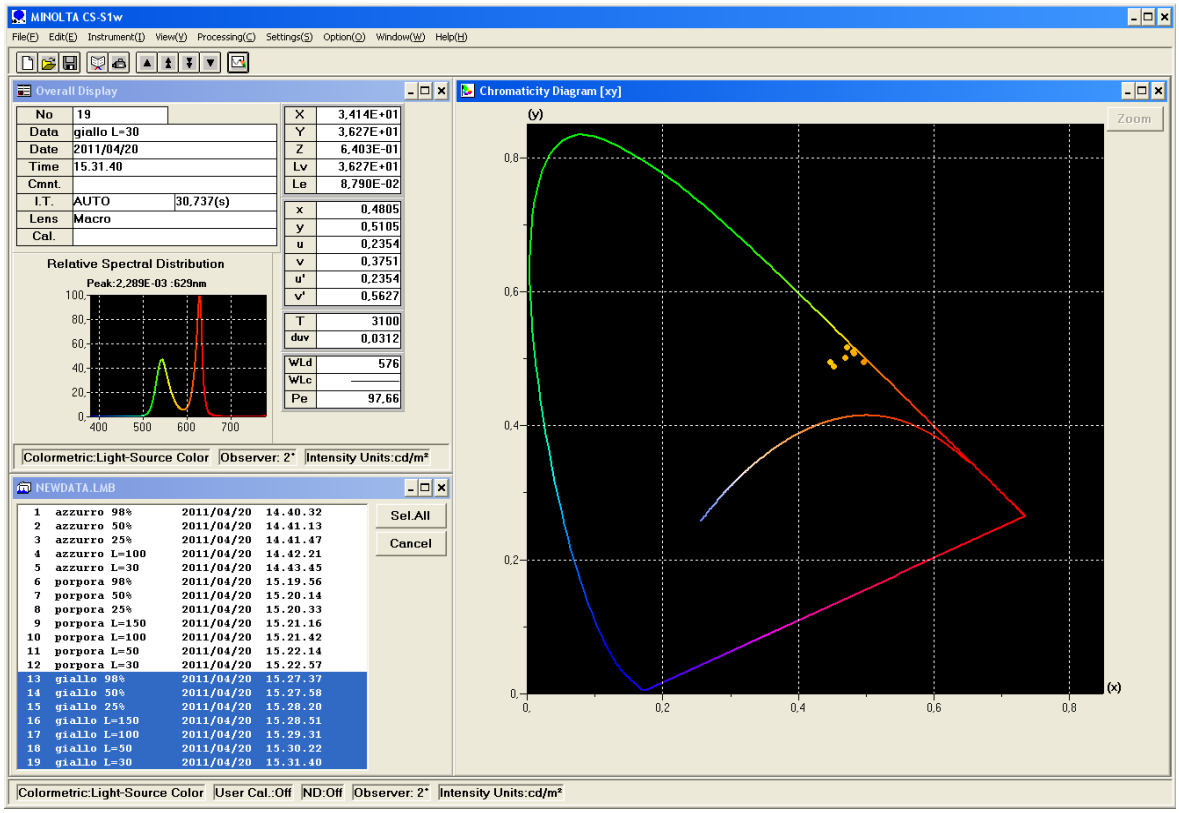


figura 6.26: variazione delle coordinate cromatiche per il colore giallo al variare della luminanza richiesta

Richiedendo una luminanza L=30, pari a circa il 2% di quella massima ottenibile, le coordinate cromatiche ottenute sono state x=0,4805 e y=0,5105.

Quindi le variazioni assolute sono $\Delta x = 0{,}0305$ e $\Delta y = 0{,}0105$
e quelle relative rispetto al valore impostato, quindi, del 6,78% per la x e 2,1% per la y;

in totale quindi, la variazione, calcolata come distanza nel piano xy, è pari al 7,1%, dunque molto più contenuta rispetto a quella subita dagli altri due colori complementari.

Successivamente, dati i risultati ottenuti, sono state effettuate altre misure soprattutto per il verde, avendo cura di effettuarle dopo aver lasciato il tempo alla piastra di raggiungere la temperatura della base dei LED, e quindi con ragionevole certezza di essere a temperatura praticamente costante.

Queste nuove misurazioni, purtroppo, non hanno fatto che confermare che i LED verdi, variando il flusso richiesto, variano coordinate cromatiche, e addirittura variano la lunghezza d'onda alla quale è presente il picco di emissione. Tale variazione non è affatto trascurabile, dato che passando dal 100% al 2% della luminanza richiesta, si passa da 535 nm a 543 nm.

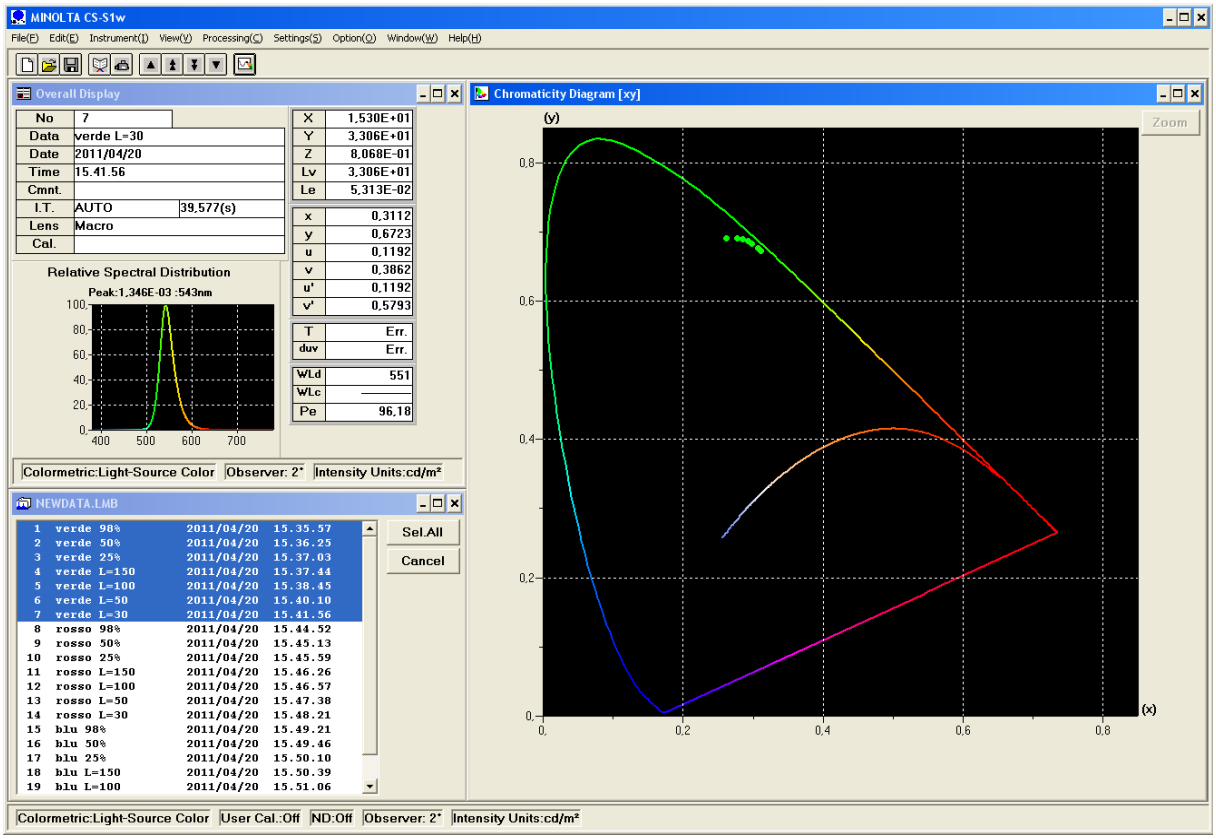


figura 6.27: variazione delle coordinate cromatiche per il colore verde al variare della luminanza richiesta

Il problema da noi avuto con i LED verdi non era mai stato rilevato, in precedenti esperienze avute in laboratorio; tuttavia, mi è stato detto che, quando si utilizzano LED verdi, non è raro che si presenti qualche comportamento anomalo: infatti, pare che i LED verdi siano più “delicati” rispetto a quelli di altri colori, a causa della loro particolare costituzione (in particolare, a causa delle miscele di elementi utilizzati per ottenere il colore verde, e della non perfetta congruenza tra i vari reticoli cristallini).

Purtroppo, su questo aspetto noi non possiamo fare niente, se non sostituire i LED con altri dal comportamento più lineare: infatti, anche realizzando una retroazione sulle coordinate cromatiche, queste non potrebbero essere modificate nel caso in cui si voglia realizzare un verde saturo.

Capitolo 7 - Possibili utilizzi della sorgente a cromaticità variabile

I possibili utilizzi di una sorgente a cromaticità variabile sono molti, soprattutto se si pensa al settore del design.

La nostra sorgente a cromaticità variabile, però, è stata costruita con un certo criterio e con certe attenzioni per poterla utilizzare come uno strumento di laboratorio.

Infatti, nell'utilizzo come semplice sorgente luminosa decorativa, non servirebbero a nulla tutti i circuiti di controllo e retroazione che sono stati implementati: questi servono invece per poter avere una ragionevole certezza sulle precisione della riproduzione di un certo colore.

Dopo questa piccola premessa, andiamo ad illustrare i principali utilizzi per la sorgente a cromaticità variabile all'interno del laboratorio di Illuminotecnica e Fotometria.

Queste possibili applicazioni sono:

1. Utilizzo all'interno della cameretta per l'esperimento "Identificazione dei colori unici"
2. Strumento di calibrazione per colorimetri
3. Sorgente luminosa per la calibrazione (tramite altro strumento calibrato) dello spettroradiometro.

Andiamo ora ad illustrare brevemente questi aspetti.

Il primo modo di utilizzare la sorgente è quello di inserirla come sorgente luminosa nella camera di visione che viene utilizzata per varie **prove sulla percezione dei colori**.

figura 7.1: campioni per l'esperimento di identificazione dei "colori unici"

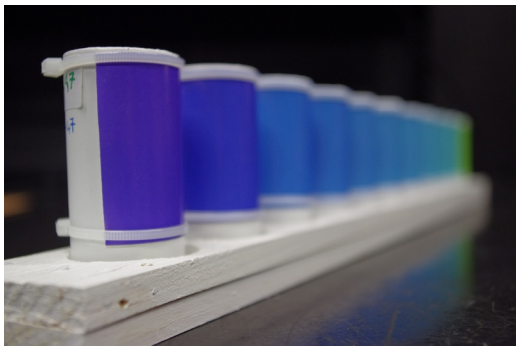


figura 7.2: sequenza di campioni per l'esperimento di identificazione del "colore unico BLU"

In questo modo, si può ottenere una variante di un esperimento che viene svolto per determinare la percezione dei “colori unici” e la capacità di distinguerli sotto diverse sorgenti luminose.

I “colori unici” sono sei: rosso, giallo, verde, blu, bianco e nero, e stanno alla base di una possibile nuova definizione dell'Indice di Resa Cromatica (CRI) a cui si sta lavorando nel Laboratorio di Illuminotecnica e Fotometria.

Per identificare questi colori unici, nella camera di visione vengono posizionati dei campioni di vari colori (ad esempio, una fila di campioni di colore variabile dal verde al viola, passando per il blu, come è mostrato nella figura 7.2): questi campioni vengono illuminati con una determinata sorgente luminosa, e il soggetto deve scegliere il campione che, secondo lui, è “quello blu”, né blu-violastro, né blu-verdastro. Lo stesso procedimento si applica con gli altri campioni, relativi agli altri colori unici. Gli stessi campioni vengono ripresentati successivamente, illuminati da una diversa sorgente luminosa, e il soggetto deve effettuare nuovamente la scelta.

Essendo l'apparenza dei colori dipendente dallo spettro della sorgente che li illumina, può succedere che le scelte effettuate nei due casi siano diverse, come è anche possibile che il campione scelto sia quello scelto in precedenza. Tutto dipende da come le due sorgenti rendono differentemente i colori dei campioni.

L'utilizzo della sorgente a cromaticità variabile in questo ambito sostituirebbe completamente i campioni colorati, illuminati da una sorgente neutra.

Quindi, invece di utilizzare dei campioni colorati illuminati da una certa sorgente luminosa, si potrebbe regolare direttamente la cromaticità della luce emessa, e variarla fino a quando il colore della luce emessa non corrisponde al

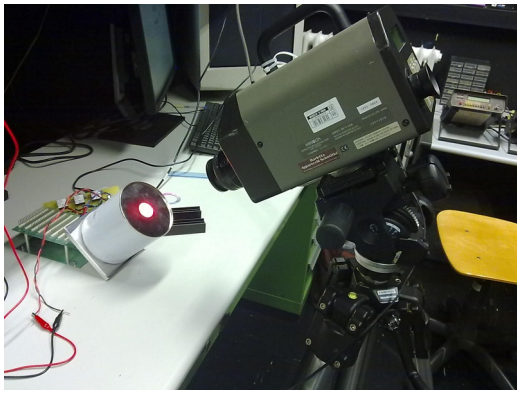
"colore unico".

Il fatto di utilizzare la sorgente anziché i campioni consentirebbe di avere una risoluzione molto maggiore nella regolazione del colore, poiché quando si utilizzano i campioni fisici, la necessità di non averne un numero troppo elevato implica una certa differenza di cromaticità tra due campioni adiacenti.

Un secondo utilizzo per la sorgente a cromaticità variabile prevede la sua utilizzazione come **calibratore per colorimetri**: infatti, utilizzando lo spazio Lxy per scegliere i valori, è possibile sintetizzare un colore con delle coordinate cromatiche ben precise.

Effettuando una misura con il colorimetro, si può vedere se le coordinate cromatiche misurate corrispondono con quelle impostate.

Un terzo possibile utilizzo della sorgente a cromaticità variabile è nella **calibrazione dello spettroradiometro**.



Infatti, sintetizzato un certo colore con la sorgente, lo spettro dell'emissione luminosa è ben noto, sia come distribuzione spettrale che come ampiezze.

Si potrebbe utilizzare la sorgente per calibrare direttamente lo spettroradiometro, ma il modo migliore è quello di utilizzarla semplicemente come sorgente luminosa: con uno strumento campione si leggono le distribuzioni spettrali e le ampiezze delle radianze spettrali, e si verifica che le stesse grandezze, misurate dallo spettroradiometro in calibrazione, siano esatte.

Capitolo 8 - Aspetti da completare e possibili sviluppi futuri

Data la mancanza di tempo per il completo sviluppo della sorgente, restano da completare alcuni aspetti per renderla completa.

In particolare, lo spazio colore HSV è già stato previsto nel menu a tendina dell'interfaccia grafica, e anche nella parte iniziale del programma che la comanda (ColorSelector.m).
Per questo spazio colore, bisogna però ancora sviluppare:

- nel programma ColorSelector.m, i *Callback* delle tre caselle e delle tre *sliders*;
- nel programma Trasf_2_RGBled, la trasformazione matematica da HSV a RGB per il comando dei LED, e da RGB agli spazi opportuni per l'anteprima grafica del punto scelto e per l'anteprima a monitor del colore selezionato.

Eventualmente, con poca fatica, si potrebbe sostituire allo spazio HSV lo spazio Lch, ovvero la notazione in coordinate polari dello spazio uniforme Lab.
In alternativa, lo spazio Lch potrebbe essere aggiunto, senza eliminare lo spazio HSV.

Si potrebbe poi visualizzare nella finestra grafica, quando si lavora in un certo spazio colore, la sua rappresentazione (eventualmente in tre dimensioni, e con un'anteprima dei colori); sarebbe poi molto intuitivo poter scegliere i valori muovendosi all'interno della rappresentazione dello spazio colore (ad esempio selezionando con il mouse un certo punto, o spostandosi con le frecce della tastiera).

Un'altra aggiunta potrebbe essere quella di definire, per ogni spazio colore, dei “punti di maggior interesse” (ad esempio, alcuni punti lungo la “linea dei bianchi” nello spazio Lxy) a cui far agganciare il cursore per poterne selezionare rapidamente le coordinate cromatiche.

Tra gli aspetti da considerare per il completamento della sorgente, c'è anche la risoluzione di alcuni piccoli problemi che sono ancora presenti allo stato attuale di sviluppo.

Tra questi, ricordiamo la variazione delle tensioni di comando degli alimentatori, e delle coordinate cromatiche, in relazione alla temperatura.

Le possibili soluzioni a questo problema sono due, e non è detto che si escludano a vicenda:

la prima soluzione, forse più semplice, è quella di aggiungere una resistenza comandata da termostato, con la funzione di portare la piastra alla temperatura di regime termico e mantenerla in un certo range nell'intorno di questa.
La seconda soluzione potrebbe essere quella di effettuare una retroazione, oltre che sull'anello “interno” della corrente, anche sulle coordinate cromatiche: si potrebbero utilizzare dei sensori RGB, che danno un segnale proporzionale, rispettivamente, agli stimoli rosso, verde e blu forniti dai LED; questi valori potrebbero essere confrontati con il valore richiesto e la loro differenza utilizzata per correggere il valore di corrente da imprimere ai LED.

Tale retroazione, pur essendo apparentemente complicata da effettuare, garantirebbe la piena sicurezza di ottenere ciò che si vuole: infatti, al momento, noi controlliamo la corrente di alimentazione dei LED, ma quello che succede tra l'alimentazione del LED e la visione della luce è al di fuori del nostro controllo.
Realizzando questa retroazione, come detto nel capitolo precedente, si risolverebbe (almeno in parte) il problema della variazione delle coordinate cromatiche al variare del flusso richiesto.

Un altro problema ancora presente è l'offset che affligge le misure effettuate con la scheda NI USB-6259: sembra che esso sia provocato da un *cross-talking* tra i canali di acquisizione, anche se le modalità con cui si presenta questo fenomeno sono abbastanza difficili da comprendere. Una volta che si sarà risolto questo problema, la riproduzione del colore scelto sarà sicuramente più accurata, e si potrà imporre una tolleranza più restrittiva anche nei casi in cui un valore sia impostato a zero.

Infine, tra gli aspetti da sistemare, seppur di poco conto, rimane il fatto di riuscire a cancellare le variabili dei tre valori impostati, ogni volta che si cambia spazio colore: in questo modo il programma attenderebbe che venissero reimmessi tutti e tre prima di mandare la terna all'output.
Cosa ancora migliore da fare, sarebbe fare in modo che, se ad esempio sto scegliendo il colore in RGB e passo in Lxy, il valore preimpostato in Lxy sia quello corrispondente al colore che ho scelto prima. Per fare questo, bisognerebbe effettuare ogni volta la tasformazione in TUTTI gli spazi colore selezionabili, in modo che sia pronta la terna di valori da inserire, in caso di cambio di spazio colore, nelle caselle.

Un ulteriore sviluppo futuro, che era già programmato sin dall'inizio del lavoro, potrà poi essere quello di utilizzare le conoscenze acquisite e i programmi sviluppati per realizzare una seconda sorgente a cromaticità variabile, la quale utilizzi però LED di 6 colori anziché 3.

In questo modo, si potrà riprodurre un gamut di cromaticità molto più ampio e in modo più completo, consentendo anche una migliore gestione del problema dei “colori unici”, a fronte però di un raddoppio di canali I/O e di componenti hardware.
Anche la maggior parte dei programmi in Matlab dovrà essere adattata, da quello che comanda l'interfaccia grafica al programma che effettua le trasformazioni matematiche.

Capitolo 9 - Conclusioni

Allo stadio di sviluppo a cui si trova, la sorgente a cromaticità variabile può essere già utilizzata per i primi test; comunque, anche se qualche miglioria può ancora essere apportata, ritengo che essa costituisca un buon punto di partenza per ulteriori sviluppi.
Sicuramente, è stato fatto tutto il possibile per ottenere un progetto ben funzionante, anche se di fronte ad alcuni comportamenti dei componenti (vedi LED verdi) non è stato possibile fornire una soluzione in tempo utile per poterla implementare.

Al termine di questo lavoro di progettazione, posso sicuramente dire di essere soddisfatto di aver scelto questo argomento come tesi.

Infatti, è stato molto più gratificante creare un progetto da zero, occupandomi anche della parte pratica di realizzazione della sorgente, che non effettuare solamente degli studi al calcolatore.

Sicuramente posso dire che, nonostante esternamente la sorgente possa sembrare di semplice funzionamento, le conoscenze che ho dovuto mettere in campo sono state molteplici.

Infatti, partendo dalla parte pratica, l'utilizzo di AutoCad mi ha notevolmente semplificato il lavoro di progettazione della piastra di sostegno dei LED, permettendomi successivamente di effettuare delle lavorazioni su macchine a controllo numerico utilizzando direttamente il file fornito da AutoCad.

Durante il montaggio dei LED e di tutte le apparecchiature elettroniche, ho acquisito una certa manualità nelle operazioni di stagnatura (sembra una sciocchezza,ma io penso che un ingegnere non possa avere solo conoscenze teoriche!).

Durante la parte di sviluppo dei programmi, poi, sono state richieste conoscenze sia di tipo matematico (per lo sviluppo delle trasformazioni matematiche tra i vari spazi colore), sia nell'ambito dei controlli automatici (per la messa a punto della retroazione).

Infine, alla fine del lavoro di progettazione, mi sono reso conto che durante il processo di sviluppo della sorgente ho appreso un metodo di lavoro: infatti, sono state prima costruite le basi, e poi studiati e risolti i vari problemi, di importanza via via decrescente, fino ad avere un insieme perfettamente funzionante.

Ringraziamenti

Un ringraziamento particolare a tutte le persone che mi hanno seguito durante il lavoro di tesi:

al Prof. Fiorentin, per il suo spirito critico che mi ha consentito di risolvere alcuni problemi della sorgente che nemmeno io avevo notato;

all'Ing. Scroccaro, per l'infinita assistenza prestatami in laboratorio e per i numerosi consigli pratici che mi ha dispensato;

ad Elena, per la pazienza che ha dovuto portare dato che le ho praticamente requisito il computer che usava di solito;

a mio papà e alla ditta *e.dalmonico* per avermi dato modo di realizzare in modo impeccabile tutte le parti meccaniche della sorgente (la piastra di alluminio di sostegno dei LED, i setti in plexiglass, i barattoli con il foro di uscita realizzati al tornio...);

a mia zia Elena e alla ditta *Targochimica* per avermi dato la possibilità di stampare il circuito stampato;

alla ditta *mondoenergia* in cui lavoro, per avermi permesso di avere un orario di lavoro flessibile, necessario per portare a termine questo lavoro di tesi.

Bibliografia

[1] C. Oleari, “Misurare il colore” seconda edizione, Editore Ulrico Hoepli MILANO, 2008.

[2] dispense del corso di Illuminotecnica e Fotometria, prof. Fiorentin Pietro

[3] Wyszecki & Stiles, “Color Science: concepts and methods, quantitative data and formulae” 2^ edizione, ed. Wiley Classics Library, 2001.

[4] Da Pos, Fiorentin, Maistrello, Pedrotti, Scroccaro, “New Assessment Criteria for Colour Quality”

Sitografia

[1] “CIE 1931 color space - Wikipedia”, http://en.wikipedia.org/wiki/CIE_1931_color_space, ultima modifica 30.09.2010;

[2] “Lab color space – Wikipedia”, http://en.wikipedia.org/wiki/Lab_color_space
ultima modifica 01.10.2010;

[3] “HSL and HSV – Wikipedia” http://en.wikipedia.org/wiki/HSL_and_HSV
ultima modifica 09.04.2011;

[4] “Sistema CIELAB 1976 – WikiColore”
http://boscarol.com/wiki/index.php?title=Sistema_CIELAB_1976
ultima modifica 26.04.2009;

[5] “CIE L*a*b* Color Scale – HunterLab” http://www.hunterlab.com/appnotes/an07_96a.pdf, 2008

[6] “About the Lab Gamut – Bruce Lindbloom” - http://brucelindbloom.com/LabGamutDisplayHelp.html, ultima modifica 18.04.2007

[7] “CIELab Color Space – Gernot Hoffman”
http://www.fho-emden.de/~hoffmann/cielab03022003.pdf, 15.09.2009

[8] “Spazio del colore, diagrammi di cromaticità e problemi di non uniformità di scala”
http://www.fis.unipr.it/home/fernando.fermi/GCA_Prato_1.htm , 1996

[9] “Color reproduction using CIE”
http://hyperphysics.phy-astr.gsu.edu/hbase/vision/colrep.html

[10]”Colorspace Transforms – by Pascal Getreuer”
http://www.mathworks.com/matlabcentral/fx_files/28790/5/content/colorspace/colorspace.html , 2010

[11] “MathWorks – Support” http://www.mathworks.com/support/, 2011

[12]”MathWorks Support – Handle Graphics and Properties Guide”
http://www.mathworks.com/support/tech-notes/1200/1205.html, 2010

Appendice A – Listati Matlab

Si riportano i listati dei programmi Matlab, in ordine alfabetico.

ApriPorte_6259.m

```
% uscite alla schede di acquisizione dati 6259:
%     AO0: tensione comando LED rosso
%     AO1: tensione comando LED verde
%     AO2: tensione comando LED blu

fprintf('\nApertura porte di comunicazione\n');

ao1 = analogoutput('nidaq','Dev2');

Tens_chan = addchannel(ao1,0:2,
{'Tens_comando_Rosso','Tens_comando_Verde','Tens_comando_Blu'});

set(ao1,'SampleRate',75.00);

% ingressi dalla scheda di acquisizione dati 6259:
%     AI0: corrente LED rosso
%     AI1: corrente LED verde
%     AI2: corrente LED blu
%     AI3: temperatura

ai1 = analoginput('nidaq','Dev2');
ai1.InputType = 'Differential';


set(ai1,'SampleRate',10000);

ch_cur = addchannel(ai1,0:2,{'Corrente_Rosso','Corrente_Verde','Corrente_Blu'});
ai1.Channel.InputRange = [-1 1];

ch_temp = addchannel(ai1,3,{'TensTemperatura'});
% ai1.Channel(4).InputRange = [-10 10];


% %-------------------------------------------------
% %controllo dello spettroradiometro Minolta CS-1000
% %
% %apertura della porta seriale
% %
% s1 = serial('COM1', 'BaudRate', 19200, 'DataBits', 8,'Parity', 'none', 'FlowControl',
'hardware', 'Terminator', 'CR', 'RequestToSend', 'on', 'ReadAsyncMode', 'continuous')
% %
% fopen(s1)
% fprintf(s1,'RMT,1') %prende il controllo dello strumento
% idn = fscanf(s1);
```

ChiudiPorte_6259.m

```
% chiusura porte di scrittura

delete(ao1);
clear ao1;

% chiusura porte di acquisizione

delete(ai1);
clear ai1;

% %-------------------------------------------------
% %controllo dello spettroradiometro Minolta CS-1000
%
% %chiusura della porta seriale
%
% fprintf(s1,'RMT,0') %chiude il controllo remoto dello strumento
% fclose(s1)
```

CieLab2CieXYZ.m

```
% da CIE L*a*b* a CIE XYZ
%la variabili di ingresso sono vettori riga

function z=CieLab2CieXYZ(Las,aas,bas,X_n,Y_n,Z_n)

X=X_n*f_1Lab((Las+16)/116+aas/500);
Y=Y_n*f_1Lab((Las+16)/116);
Z=Z_n*f_1Lab((Las+16)/116-bas/200);

z=[X;Y;Z];

function f_1=f_1Lab(t)

f_1(t>6/29)=t(t>6/29).^3;
f_1(t<=6/29)=3*(6/29)^2*(t(t<=6/29)-4/29);
```

CieLxy2CieXYZ.m

```
%da CIE Lxy a CIE XYZ

function z=CieLxy2CieXYZ(L,x,y)

X=(L./y).*x;
Y=L;
Z=(L./y).*(1-x-y);

z=[X;Y;Z];
```

CieXYZ2CieLab.m

```
%trasformazione da CIE XYZ a CIE L*a*b*

function z=CieXYZ2CieLab(X,Y,Z,Xn,Yn,Zn)

%ingresso
% X Y Z vettori delle coordinate tristimolo in CIE XYZ
% Xn Yn Zn coordinate tristimolo della sorgente di riferimento
%uscta
% Las aas bas vettori delle coordinate in CIE L*a*b*

%-----------------------------------------------------------------

%sorgente di riferimento
% Xn=(x1*D65')/(y1*D65');
% Yn=1;
% Zn=(z1*D65')/(y1*D65');

xr=(X/Xn);
yr=(Y/Yn);
zr=(Z/Zn);

fx=zeros(size(xr));
fy=zeros(size(yr));
fz=zeros(size(zr));

fx(xr>(6/29)^3)=xr(xr>(6/29)^3).^(1/3);
fx(xr<=(6/29)^3)=1/3*(29/6)^2*xr(xr<=(6/29)^3)+4/29;
fy(yr>(6/29)^3)=yr(yr>(6/29)^3).^(1/3);
fy(yr<=(6/29)^3)=1/3*(29/6)^2*yr(yr<=(6/29)^3)+4/29;
fz(zr>(6/29)^3)=zr(zr>(6/29)^3).^(1/3);
fz(zr<=(6/29)^3)=1/3*(29/6)^2*zr(zr<=(6/29)^3)+4/29;

Las=116*fy-16;
aas=500*(fx-fy);
bas=200*(fy-fz);

clear fx fy fz

z=[Las;aas;bas];
```

CieXYZ2CieLxy.m

```
%da CIE XYZ a CIE Lxy
%la variabili di ingresso sono vettori riga

function z=CieXYZ2CieLxy(X,Y,Z)

L=Y;
x=X./(X+Y+Z);
y=Y./(X+Y+Z);
z=[L;x;y];
```

CieXYZ2RGB_led.m

```
%trasformazione da CIE XYZ a RGB_LED
%le variabili di ingresso X, Y, Z sono vettori riga
%e
%la matrice di trasformazione
%le coordinate RGB_LED sono le colonne della matrice di uscita

function z=CieXYZ2RGB_led(X,Y,Z,daXYZaRGB)

RGB=daXYZaRGB*[X; Y; Z];

z=RGB';
```

CieXYZ2sRGB.m

```
% da CIE XYZ a sRGB per anteprima a monitor

function Terna_sRGB=CieXYZ2sRGB(X,Y,Z)

normalizzatore=1300;
XYZ=[X Y Z];
XYZnorm=XYZ./normalizzatore;

%matrice di conversione da XYZ a sRGB per l'anteprima a monitor (in
%riferimento al bianco D65)

daCieXYZaSRGB=[3.2406 -1.5372 -0.4986;-0.9689 1.8758 0.0415;0.0557 -0.2040 1.0570];

RGBlin=daCieXYZaSRGB*XYZnorm';
Rlin=RGBlin(1);
Glin=RGBlin(2);
Blin=RGBlin(3);

RsRGB=Rlin*f_sRGB(Rlin);
GsRGB=Glin*f_sRGB(Glin);
BsRGB=Blin*f_sRGB(Blin);

Terna_sRGB=[RsRGB;GsRGB;BsRGB];


function f=f_sRGB(Clin)

a=0.055;

f(Clin<=0.0031308)=12.92*Clin;
f(Clin>0.0031308)=(1+a)*(Clin^(1/2.4))-a;
```

ColorSelector.m

```
function varargout = ColorSelector(varargin)
% ColorSelector M-file for ColorSelector.fig
%      ColorSelector, by itself, creates a new ColorSelector or raises the existing
%      singleton*.
%
%      H = ColorSelector returns the handle to a new ColorSelector or the handle to
%      the existing singleton*.
%
%      ColorSelector('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ColorSelector.M with the given input arguments.
%
%      ColorSelector('Property','Value',...) creates a new ColorSelector or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before ColorSelector_OpeningFcn gets calLED.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to ColorSelector_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ColorSelector

% Last Modified by GUIDE v2.5 03-Feb-2011 17:12:27

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ColorSelector_OpeningFcn, ...
                   'gui_OutputFcn',  @ColorSelector_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before ColorSelector is made visible.
function ColorSelector_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for ColorSelector
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% UIWAIT makes ColorSelector wait for user response (see UIRESUME)

% se sono già stati impostati tutti e tre i valori, anche se non vengono
% fatte modifiche dopo 5 secondi questi valori vengono di nuovo passati in
% uscita
if isfield(handles, 'PrimoDitre') && isfield(handles, 'SecondoDitre')...
    && isfield(handles, 'TerzoDitre') && handles.bordo && ~handles.pausa
    %questa condizione è vera solo quando sono stati inseriti tutti e tre i
    %valori, quando il punto selezionato è all'interno del bordo di Lxy e
```

```
    %quando il programma non è in pausa perchè ho appena cambiato spazio
    %colore
    uiwait(handles.figure1,5); %in questo caso, attendo il comando uiresume, che la fine-
stra venga chiusa o che passino 5 secondi

else uiwait(handles.figure1); %attendo che vengano impostati tutti e tre i valori prima
di mandarli in uscita
end


% --- Outputs from this function are returned to the command line.
function varargout = ColorSelector_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);

display 'sto mandando fuori i valori'

% Get default command line output from handles structure
if isfield(handles, 'figure1') % durante il normale funzionamento mando fuori questi va-
lori
    varargout{1} = handles.Terna;
    varargout{2} = handles.Fine;
else % quando chiudo il programma mando fuori questi valori
    varargout{1} = [0 0 0];
    varargout{2} = 0;
end



% --- Executes on selection change in ScegliSpazioColore.
function ScegliSpazioColore_Callback(hObject, eventdata, handles)

handles.Spazio = get(handles.ScegliSpazioColore,'Value');
guidata(gcbo,handles);

switch handles.Spazio

    case 1 %Spazio RGB
        display 'Stai scegliendo il colore nello spazio RGB'

        %imposto i testi sulle caselle, colori e range sulle sliders
        set(handles.text2,'String','RED');
        set(handles.text3,'String','GREEN');
        set(handles.text4,'String','BLUE');
        set(handles.text7,'String','RED 0...1');
        set(handles.text8,'String','GREEN 0...1');
        set(handles.text9,'String','BLUE 0...1');
        set(handles.slider1,'Min',0,'Max',1,'BackgroundColor',[1 0 0])
        set(handles.slider2,'Min',0,'Max',1,'BackgroundColor',[0 1 0])
        set(handles.slider3,'Min',0,'Max',1,'BackgroundColor',[0 0 1])

        if isfield(handles, 'PrimoDitre') && isfield(handles, 'SecondoDitre') &&
isfield(handles, 'TerzoDitre') %questa condizione è vera solo quando sono stati
                                inseriti tutti e tre i valori

% quando cambio spazio colore, reimposto tutto a zero e metto in pausa
        set (handles.slider1,'Value',0)
        set (handles.Casella1,'String',0)
        set (handles.slider2,'Value',0)
        set (handles.Casella2,'String',0)
        set (handles.slider3,'Value',0)
        set (handles.Casella3,'String',0)

        handles.PrimoDitre=str2double(get(handles.Casella1,'String'));
        handles.SecondoDitre=str2double(get(handles.Casella2,'String'));
        handles.TerzoDitre=str2double(get(handles.Casella3,'String'));
        handles.Terna=[handles.PrimoDitre handles.SecondoDitre handles.TerzoDitre];
        handles.pausa=1;
```

```
        guidata(gcbo,handles);

        end


    case 2 %Spazio Lxy
        display 'Stai scegliendo il colore nello spazio Lxy'

        %imposto i testi sulle caselle, colori e range sulle sliders
        set(handles.text2,'String','L');
        set(handles.text3,'String','x');
        set(handles.text4,'String','y');
        set(handles.text7,'String','L 0...1300');
        set(handles.text8,'String','x');
        set(handles.text9,'String','y');
        set(handles.slider1,'Min',0,'Max',1300,'BackgroundColor',[1 1 1])
        set(handles.slider2,'Min',0,'Max',0.8,'BackgroundColor',[1 1 1])
        set(handles.slider3,'Min',0,'Max',0.9,'BackgroundColor',[1 1 1])

        if isfield(handles, 'PrimoDitre') && isfield(handles, 'SecondoDitre') &&
isfield(handles, 'TerzoDitre') %questa condizione è vera solo quando sono stati
                        inseriti tutti e tre i valori

% quando cambio spazio colore, reimposto tutto a zero e metto in pausa
            set (handles.slider1,'Value',0)
            set (handles.Casella1,'String',0)
            set (handles.slider2,'Value',0)
            set (handles.Casella2,'String',0)
            set (handles.slider3,'Value',0)
            set (handles.Casella3,'String',0)

            handles.PrimoDitre=str2double(get(handles.Casella1,'String'));
            handles.SecondoDitre=str2double(get(handles.Casella2,'String'));
            handles.TerzoDitre=str2double(get(handles.Casella3,'String'));
            handles.x_Lxy=handles.SecondoDitre;
            handles.y_Lxy=handles.TerzoDitre;
            handles.Terna=[handles.PrimoDitre handles.SecondoDitre handles.TerzoDitre];
            handles.pausa=1;

            guidata(gcbo,handles);

        end


    case 3 %Spazio Lab
        display 'Stai scegliendo il colore nello spazio Lab'

        %imposto i testi sulle caselle, colori e range sulle sliders
        set(handles.text2,'String','L');
        set(handles.text3,'String','a');
        set(handles.text4,'String','b');
        set(handles.text7,'String','L 0...110');
        set(handles.text8,'String','a -150...150');
        set(handles.text9,'String','b -150...150');

        %i range delle sliders sono da AGGIORNARE quando avrò il bordo Lab
        set(handles.slider1,'Min',0,'Max',100,'BackgroundColor',[1 1 1])
        set(handles.slider2,'Min',-150,'Max',150,'BackgroundColor',[1 1 1])
        set(handles.slider3,'Min',-150,'Max',150,'BackgroundColor',[1 1 1])

        if isfield(handles, 'PrimoDitre') && isfield(handles, 'SecondoDitre') &&
isfield(handles, 'TerzoDitre') %questa condizione è vera solo quando sono stati inseriti
tutti e tre i valori

% quando cambio spazio colore, reimposto tutto a zero e metto in pausa
        set (handles.slider1,'Value',0)
        set (handles.Casella1,'String',0)
        set (handles.slider2,'Value',0)
```

```
        set (handles.Casella2,'String',0)
        set (handles.slider3,'Value',0)
        set (handles.Casella3,'String',0)

        handles.PrimoDitre=str2double(get(handles.Casella1,'String'));
        handles.SecondoDitre=str2double(get(handles.Casella2,'String'));
        handles.TerzoDitre=str2double(get(handles.Casella3,'String'));
        handles.Terna=[handles.PrimoDitre handles.SecondoDitre handles.TerzoDitre];
        handles.pausa=1;

        guidata(gcbo,handles);

        end


    case 4 %Spazio HSV
        display 'Stai scegliendo il colore nello spazio HSV'

        %imposto i testi sulle caselle, colori e range sulle sliders
        set(handles.text2,'String','H');
        set(handles.text3,'String','S');
        set(handles.text4,'String','V');
        set(handles.text7,'String','H 0°...360°');
        set(handles.text8,'String','S 0...1');
        set(handles.text9,'String','V 0...1');

        set(handles.slider1,'Min',0,'Max',359,'BackgroundColor',[1 1 1])
        set(handles.slider2,'Min',0,'Max',1,'BackgroundColor',[1 1 1])
        set(handles.slider3,'Min',0,'Max',1,'BackgroundColor',[1 1 1])

        if isfield(handles, 'PrimoDitre') && isfield(handles, 'SecondoDitre') &&
isfield(handles, 'TerzoDitre') %questa condizione è vera solo quando sono stati
                        inseriti tutti e tre i valori

% quando cambio spazio colore, reimposto tutto a zero e metto in pausa
        set (handles.slider1,'Value',0)
        set (handles.Casella1,'String',0)
        set (handles.slider2,'Value',0)
        set (handles.Casella2,'String',0)
        set (handles.slider3,'Value',0)
        set (handles.Casella3,'String',0)

        handles.PrimoDitre=str2double(get(handles.Casella1,'String'));
        handles.SecondoDitre=str2double(get(handles.Casella2,'String'));
        handles.TerzoDitre=str2double(get(handles.Casella3,'String'));
        handles.Terna=[handles.PrimoDitre handles.SecondoDitre handles.TerzoDitre];
        handles.pausa=1;

        guidata(gcbo,handles);
        end
end



% --- Executes during object creation, after setting all properties.
function ScegliSpazioColore_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundCol-
or'))
    set(hObject,'BackgroundColor','white');
end



function Casella1_Callback(hObject, eventdata, handles)

set(handles.slider1,'Value',str2double(get(handles.Casella1,'String')));
handles.PrimoDitre=str2double(get(handles.Casella1,'String'));
```

```
guidata(gcbo,handles);

switch handles.Spazio
    case 1 %Spazio RGB
        handles.R_RGB=handles.PrimoDitre;
    case 2 %Spazio Lxy
        handles.L_Lxy=handles.PrimoDitre;
    case 3 %Spazio Lab
        handles.L_Lab=handles.PrimoDitre;
end
guidata(gcbo,handles);

%--------------------------------------------------------------------------
if isfield(handles, 'PrimoDitre') && isfield(handles, 'SecondoDitre') && isfield(handles,
'TerzoDitre')
    handles.pausa=0;
    display 'I colori sono stati scelti'
    handles.Fine = 1;
    guidata(gcbo,handles);

    handles.bordo=verifica_inpolygon(handles.Spazio, handles.PrimoDitre, handles.Secondo-
Ditre, handles.TerzoDitre);

    if handles.bordo %vale 1 se il punto è all'interno del bordo
        set(handles.text10,'BackgroundColor',[0 1 0],'ForegroundColor',[0 1 0]);
        [handles.Terna handles.Terna_sRGB] = Trasf_2_RGBled(handles.Spazio,
handles.PrimoDitre, handles.SecondoDitre, handles.TerzoDitre);

        if max(handles.Terna)<=1 && min(handles.Terna)>=0
            set(handles.text11,'BackgroundColor',[0 1 0],'ForegroundColor',[0 1 0]);
        else set(handles.text11,'BackgroundColor',[1 0 0],'ForegroundColor',[1 1 1]);

        handles.Terna(1)=min(max(handles.Terna(1),0),1);
        handles.Terna(2)=min(max(handles.Terna(2),0),1);
        handles.Terna(3)=min(max(handles.Terna(3),0),1);
        end

        if max(handles.Terna_sRGB)<=1 && min(handles.Terna_sRGB)>=0
            set(handles.text6,'BackgroundColor',handles.Terna_sRGB,'String','');
        else set(handles.text6,'BackgroundColor','white','String','Colore non riprodu-
cibile a monitor');
        end

        uiresume;

    else set(handles.text10,'BackgroundColor',[1 0 0],'ForegroundColor',[1 1 1]);
        set(handles.text11,'BackgroundColor',[1 0 0],'ForegroundColor',[1 0 0]);
    end

guidata(gcbo,handles);
uiresume;
end
%--------------------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function Casella1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundCol-
or'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on slider1 movement.
function slider1_Callback(hObject, eventdata, handles)

set(handles.Casella1,'String',num2str(get(handles.slider1,'Value')));
handles.PrimoDitre=get(handles.slider1,'Value');
```

```
guidata(gcbo,handles);
switch handles.Spazio
    case 1 %Spazio RGB
        handles.R_RGB=handles.PrimoDitre;
    case 2 %Spazio Lxy
        handles.L_Lxy=handles.PrimoDitre;
    case 3 %Spazio Lab
        handles.L_Lab=handles.PrimoDitre;
end
guidata(gcbo,handles);

%--------------------------------------------------------------------------
if isfield(handles, 'PrimoDitre') && isfield(handles, 'SecondoDitre') && isfield(handles,
'TerzoDitre')
    handles.pausa=0;
display 'I colori sono stati scelti'
handles.Fine = 1;
guidata(gcbo,handles);

handles.bordo=verifica_inpolygon(handles.Spazio, handles.PrimoDitre, handles.SecondoDi-
tre, handles.TerzoDitre);
if handles.bordo %vale 1 se il punto è all'interno del bordo
            set(handles.text10,'BackgroundColor',[0 1 0],'ForegroundColor',[0 1 0]);
            [handles.Terna handles.Terna_sRGB] = Trasf_2_RGBled(handles.Spazio,
handles.PrimoDitre, handles.SecondoDitre, handles.TerzoDitre);

            if max(handles.Terna)<=1 && min(handles.Terna)>=0
                set(handles.text11,'BackgroundColor',[0 1 0],'ForegroundColor',[0 1 0]);
            else set(handles.text11,'BackgroundColor',[1 0 0],'ForegroundColor',[1 1 1]);

            handles.Terna(1)=min(max(handles.Terna(1),0),1);
            handles.Terna(2)=min(max(handles.Terna(2),0),1);
            handles.Terna(3)=min(max(handles.Terna(3),0),1);
            end

            if max(handles.Terna_sRGB)<=1 && min(handles.Terna_sRGB)>=0
                set(handles.text6,'BackgroundColor',handles.Terna_sRGB,'String','');
            else set(handles.text6,'BackgroundColor','white','String','Colore non ripro-
ducibile a monitor');
            end

            uiresume;

        else set(handles.text10,'BackgroundColor',[1 0 0],'ForegroundColor',[1 1 1]);
            set(handles.text11,'BackgroundColor',[1 0 0],'ForegroundColor',[1 0 0]);
        end

guidata(gcbo,handles);
uiresume;
end
%--------------------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function Casella2_Callback(hObject, eventdata, handles)

set(handles.slider2,'Value',str2double(get(handles.Casella2,'String')));
handles.SecondoDitre=str2double(get(handles.Casella2,'String'));
guidata(gcbo,handles);
switch handles.Spazio
    case 1 %Spazio RGB
```

```
        handles.G_RGB=handles.SecondoDitre;
    case 2 %Spazio Lxy
        handles.x_Lxy=handles.SecondoDitre;
    case 3 %Spazio Lab
        handles.a_Lab=handles.SecondoDitre;
end
guidata(gcbo,handles);

%--------------------------------------------------------------------------
if isfield(handles, 'PrimoDitre') && isfield(handles, 'SecondoDitre') && isfield(handles,
'TerzoDitre')
    handles.pausa=0;
display 'I colori sono stati scelti'
handles.Fine = 1;
guidata(gcbo,handles);

handles.bordo=verifica_inpolygon(handles.Spazio, handles.PrimoDitre, handles.SecondoDi-
tre, handles.TerzoDitre);
if handles.bordo %vale 1 se il punto è all'interno del bordo
            set(handles.text10,'BackgroundColor',[0 1 0],'ForegroundColor',[0 1 0]);
            [handles.Terna handles.Terna_sRGB] = Trasf_2_RGBled(handles.Spazio,
handles.PrimoDitre, handles.SecondoDitre, handles.TerzoDitre);

            if max(handles.Terna)<=1 && min(handles.Terna)>=0
                set(handles.text11,'BackgroundColor',[0 1 0],'ForegroundColor',[0 1 0]);
            else set(handles.text11,'BackgroundColor',[1 0 0],'ForegroundColor',[1 1 1]);

            handles.Terna(1)=min(max(handles.Terna(1),0),1);
            handles.Terna(2)=min(max(handles.Terna(2),0),1);
            handles.Terna(3)=min(max(handles.Terna(3),0),1);
            end

            if max(handles.Terna_sRGB)<=1 && min(handles.Terna_sRGB)>=0
                set(handles.text6,'BackgroundColor',handles.Terna_sRGB,'String','');
            else set(handles.text6,'BackgroundColor','white','String','Colore non ripro-
ducibile a monitor');
            end

            uiresume;

        else set(handles.text10,'BackgroundColor',[1 0 0],'ForegroundColor',[1 1 1]);
            set(handles.text11,'BackgroundColor',[1 0 0],'ForegroundColor',[1 0 0]);
        end

guidata(gcbo,handles);
uiresume;
end
%--------------------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function Casella2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundCol-
or'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider2 movement.
function slider2_Callback(hObject, eventdata, handles)

set(handles.Casella2,'String',num2str(get(handles.slider2,'Value')));
%guidata(gcbo,handles);
handles.SecondoDitre=get(handles.slider2,'Value');
guidata(gcbo,handles);
switch handles.Spazio
    case 1 %Spazio RGB
```

```
        handles.G_RGB=handles.SecondoDitre;
    case 2 %Spazio Lxy
        handles.x_Lxy=handles.SecondoDitre;
    case 3 %Spazio Lab
        handles.a_Lab=handles.SecondoDitre;
end
guidata(gcbo,handles);

%-------------------------------------------------------------------------
if isfield(handles, 'PrimoDitre') && isfield(handles, 'SecondoDitre') && isfield(handles,
'TerzoDitre')
    handles.pausa=0;
display 'I colori sono stati scelti'
handles.Fine = 1;
guidata(gcbo,handles);

handles.bordo=verifica_inpolygon(handles.Spazio, handles.PrimoDitre, handles.SecondoDi-
tre, handles.TerzoDitre);
if handles.bordo %vale 1 se il punto è all'interno del bordo
            set(handles.text10,'BackgroundColor',[0 1 0],'ForegroundColor',[0 1 0]);
            [handles.Terna handles.Terna_sRGB] = Trasf_2_RGBled(handles.Spazio,
handles.PrimoDitre, handles.SecondoDitre, handles.TerzoDitre);

            if max(handles.Terna)<=1 && min(handles.Terna)>=0
                set(handles.text11,'BackgroundColor',[0 1 0],'ForegroundColor',[0 1 0]);
            else set(handles.text11,'BackgroundColor',[1 0 0],'ForegroundColor',[1 1 1]);

            handles.Terna(1)=min(max(handles.Terna(1),0),1);
            handles.Terna(2)=min(max(handles.Terna(2),0),1);
            handles.Terna(3)=min(max(handles.Terna(3),0),1);
            end

            if max(handles.Terna_sRGB)<=1 && min(handles.Terna_sRGB)>=0
                set(handles.text6,'BackgroundColor',handles.Terna_sRGB,'String','');
            else set(handles.text6,'BackgroundColor','white','String','Colore non ripro-
ducibile a monitor');
            end

            uiresume;

        else set(handles.text10,'BackgroundColor',[1 0 0],'ForegroundColor',[1 1 1]);
            set(handles.text11,'BackgroundColor',[1 0 0],'ForegroundColor',[1 0 0]);
        end

guidata(gcbo,handles);
uiresume;
end
%-------------------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end




function Casella3_Callback(hObject, eventdata, handles)

set(handles.slider3,'Value',str2double(get(handles.Casella3,'String')));

handles.TerzoDitre=str2double(get(handles.Casella3,'String'));
guidata(gcbo,handles);
%assegno i valori alle variabili, in base allo spazio colore in cui sto
%lavorando
switch handles.Spazio
```

```
    case 1 %Spazio RGB
        handles.B_RGB=handles.TerzoDitre;
    case 2 %Spazio Lxy
        handles.y_Lxy=handles.TerzoDitre;
    case 3 %Spazio Lab
        handles.b_Lab=handles.TerzoDitre;
end
guidata(gcbo,handles);

%-------------------------------------------------------------------------
if isfield(handles, 'PrimoDitre') && isfield(handles, 'SecondoDitre') && isfield(handles,
'TerzoDitre')
    handles.pausa=0;
display 'I colori sono stati scelti'
handles.Fine = 1;
guidata(gcbo,handles);

handles.bordo=verifica_inpolygon(handles.Spazio, handles.PrimoDitre, handles.SecondoDi-
tre, handles.TerzoDitre);
if handles.bordo %vale 1 se il punto è all'interno del bordo
            set(handles.text10,'BackgroundColor',[0 1 0],'ForegroundColor',[0 1 0]);
            [handles.Terna handles.Terna_sRGB] = Trasf_2_RGBled(handles.Spazio,
handles.PrimoDitre, handles.SecondoDitre, handles.TerzoDitre);

            if max(handles.Terna)<=1 && min(handles.Terna)>=0
                set(handles.text11,'BackgroundColor',[0 1 0],'ForegroundColor',[0 1 0]);
            else set(handles.text11,'BackgroundColor',[1 0 0],'ForegroundColor',[1 1 1]);

            handles.Terna(1)=min(max(handles.Terna(1),0),1);
            handles.Terna(2)=min(max(handles.Terna(2),0),1);
            handles.Terna(3)=min(max(handles.Terna(3),0),1);
            end

            if max(handles.Terna_sRGB)<=1 && min(handles.Terna_sRGB)>=0
                set(handles.text6,'BackgroundColor',handles.Terna_sRGB,'String','');
            else set(handles.text6,'BackgroundColor','white','String','Colore non ripro-
ducibile a monitor');
            end

            uiresume;

        else set(handles.text10,'BackgroundColor',[1 0 0],'ForegroundColor',[1 1 1]);
            set(handles.text11,'BackgroundColor',[1 0 0],'ForegroundColor',[1 0 0]);
        end

guidata(gcbo,handles);
uiresume;
end
%-------------------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function Casella3_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundCol-
or'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function Grafico_CreateFcn(hObject, eventdata, handles)

CreaSpazioLxy

% --- Executes on Slider3 movement.
function slider3_Callback(hObject, eventdata, handles)

set(handles.Casella3,'String',num2str(get(handles.slider3,'Value')));
```

```
handles.TerzoDitre=get(handles.slider3,'Value');
guidata(gcbo,handles);
switch handles.Spazio
    case 1 %Spazio RGB
        handles.B_RGB=handles.TerzoDitre;
    case 2 %Spazio Lxy
        handles.y_Lxy=handles.TerzoDitre;
    case 3 %Spazio Lab
        handles.b_Lab=handles.TerzoDitre;
end
guidata(gcbo,handles);

%--------------------------------------------------------------------------
if isfield(handles, 'PrimoDitre') && isfield(handles, 'SecondoDitre') && isfield(handles,
'TerzoDitre')
    handles.pausa=0;
display 'I colori sono stati scelti'
handles.Fine = 1;
guidata(gcbo,handles);

handles.bordo=verifica_inpolygon(handles.Spazio, handles.PrimoDitre, handles.SecondoDi-
tre, handles.TerzoDitre);
if handles.bordo %vale 1 se il punto è all'interno del bordo
            set(handles.text10,'BackgroundColor',[0 1 0],'ForegroundColor',[0 1 0]);
            [handles.Terna handles.Terna_sRGB] = Trasf_2_RGBled(handles.Spazio,
handles.PrimoDitre, handles.SecondoDitre, handles.TerzoDitre);

            if max(handles.Terna)<=1 && min(handles.Terna)>=0
                set(handles.text11,'BackgroundColor',[0 1 0],'ForegroundColor',[0 1 0]);
            else set(handles.text11,'BackgroundColor',[1 0 0],'ForegroundColor',[1 1 1]);

            handles.Terna(1)=min(max(handles.Terna(1),0),1);
            handles.Terna(2)=min(max(handles.Terna(2),0),1);
            handles.Terna(3)=min(max(handles.Terna(3),0),1);
            end

            if max(handles.Terna_sRGB)<=1 && min(handles.Terna_sRGB)>=0
                set(handles.text6,'BackgroundColor',handles.Terna_sRGB,'String','');
            else set(handles.text6,'BackgroundColor','white','String','Colore non ripro-
ducibile a monitor');
            end

            uiresume;

    else set(handles.text10,'BackgroundColor',[1 0 0],'ForegroundColor',[1 1 1]);
        set(handles.text11,'BackgroundColor',[1 0 0],'ForegroundColor',[1 0 0]);
end

guidata(gcbo,handles);
uiresume;
end
%--------------------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

guidata(gcbo,handles);
```

```
uiresume;

% --- Executes during object creation, after setting all properties.
function text6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns calLED


% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns calLED


% --- Executes during object deletion, before destroying properties.
function figure1_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

display 'Chiusura del Color Selector'



% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
delete(hObject);
```

ComandoSchedaDAQ_6259.m

```
% Questa è la parte di script che manda fuori i valori delle tensioni da
% applicare agli alimentatori, in modo che la corrente che essi erogano sia
% pari a quella richiesta dal colore scelto.

fprintf('\nmando i valori alla scheda NI USB-6259\n');

TernaRGB=Terna_correnti_RGB;

%Ricostruisco la caratteristica di uscita degli alimentatori dimmerabili
%LuxDrive 3021 BuckPuck
CtrlTensRed=-2.55*TernaRGB(1)+4.2;
CtrlTensGreen=-2.55*TernaRGB(2)+4.2;
CtrlTensBlue=-2.55*TernaRGB(3)+4.2;

putsample(ao1,[CtrlTensRed CtrlTensGreen CtrlTensBlue]);
```

CreaSpazioLxy.m

```
% disegno il bordo dello spazio Lxy

[lam X Npunti]=leggi_x;
[lam Y]=leggi_y;
[lam Z]=leggi_z;

x=X./(X+Y+Z);
y=Y./(X+Y+Z);
z=Z./(X+Y+Z);
```

```
CoordLxy(1,:)=lam;
CoordLxy(2,:)=x;
CoordLxy(3,:)=y;
CoordLxy(4,:)=Y;

%questi valori mi servono per chiudere la curva
CoordLxy(1,Npunti+1)=lam(1);
CoordLxy(2,Npunti+1)=x(1);
CoordLxy(3,Npunti+1)=y(1);
CoordLxy(4,Npunti+1)=Y(1);

plot(CoordLxy(2,:),CoordLxy(3,:),'k');
grid on
```

leggi_x.m

```
function [lam,V,Nl] = leggi_x

%sensibilità dell'occhio umano V(lambda)
fid=fopen('Col_mat_fun_x','r');
for n=1:8
   line=fgetl(fid);
end
Nl=str2double(fgetl(fid));
a=fscanf(fid,'%g %g',[2,inf]);
fclose(fid);
lam=a(1,:); %lunghezza d'onda banda visibile (m)
V=a(2,:); %sensibilità spettrale dell'occhio umano (-)
clear fid line a
%V=V(21:Nl);
```

leggi_y.m

```
function [lam,V] = leggi_y

%sensibilità dell'occhio umano V(lambda)
fid=fopen('Col_mat_fun_y','r');
for n=1:8
   line=fgetl(fid);
end
Nl=str2double(fgetl(fid));
a=fscanf(fid,'%g %g',[2,inf]);
fclose(fid);
lam=a(1,:); %lunghezza d'onda banda visibile (m)
V=a(2,:); %sensibilità spettrale dell'occhio umano (-)
clear fid line a
%V=V(21:Nl);
```

leggi_z.m

```
function [lam,V] = leggi_z

%sensibilità dell'occhio umano V(lambda)
fid=fopen('Col_mat_fun_z','r');
for n=1:8
   line=fgetl(fid);
end
Nl=str2double(fgetl(fid));
a=fscanf(fid,'%g %g',[2,inf]);
fclose(fid);
lam=a(1,:); %lunghezza d'onda banda visibile (m)
V=a(2,:); %sensibilità spettrale dell'occhio umano (-)
clear fid line a
%V=V(21:Nl);
```

leggispettro.m

```
%Caricamento file in uscita dallo spettroradiometro Minolta CS-1000:
%conversione in formato ".txt" --> "spettro_sorgente.txt"

function [XSP,nome]=leggispettroragiometro(filemisure);
fid=fopen([filemisure],'r');

na=0;
line=fgetl(fid);

while line~=-1


    na=na+1;

    for k=1:2
        line=fgetl(fid);
    end

    data(na,:)=line(2:length(line)-1);
    line=fgetl(fid);
   % ora(na,:)=line(2:length(line)-1);

    line=fgetl(fid);
    line=fgetl(fid);

    n=1;
    while line(n)~=','
        n=n+1;
    end
    n=n+2;
    n1=n;
    while line(n)~='"'
        n=n+1;
    end
    n2=n-1;
    it(na)=str2double(line(n1:n2)); %durata dell'intervallo di tempo di integrazione
    for k=1:6
        line=fgetl(fid);
    end

    XSP(na,:)=fscanf(fid,'%g',[1,401]);
    line=fgetl(fid);
    line=fgetl(fid);
end %while

fclose(fid);
```

LetturaCorrentiTempProlungata_6259.m

```
% grandezze utili per il calcolo della temperatura
R_Shunt_Temp=0.972; %[Ohm]
CorrezioneOffsetTemperatura=-2.5; %[°C]


% Questa è la parte di script che acquisisce i valori, e li trasforma nelle
% grandezze corrispondenti
start(ai1)
tic;
Acquisition=getdata(ai1);
t=toc;
CorrenteRossoMis=Acquisition(:,1);
```

```
CorrenteVerdeMis=Acquisition(:,2);
CorrenteBluMis=Acquisition(:,3);

CorrenteRossoMed=mean(CorrenteRossoMis);
CorrenteVerdeMed=mean(CorrenteVerdeMis);
CorrenteBluMed=mean(CorrenteBluMis);

% acquisizione tensione della sonda di temperatura

TensTemperaturaMis=Acquisition(:,4);

%calcolo della temperatura partendo dalla tensione acquisita

TemperaturaReale=mean(TensTemperaturaMis)*(1000/R_Shunt_Temp)-273.15 + CorrezioneOffset-
Temperatura;
%il valore 0.972 è la resistenza, in kOhm, misurata ai capi della resistenza
%di shunt per l'acquisizione di temperatura. La resistenza sarebbe da 1
%kOhm, ma bisogna considerare l'impedenza del cavo coassiale con cui è
%collegata alla scheda DAQ.

% wait(ai1,0.5)
stop (ai1)

%VisualizzaValoriLetti
```

Lookup_tables.m

Nota: non ho riportato i vettori complete per questioni di spazio

```
function Terna_correnti=Lookup_tables(Terna_flussi)

% definizione dei vettori delle correnti misurate (ordinate) e delle
% radianze (ordinate secondo la corrispondente corrente);
% intanto li scrivo così, perchè fare un fscan con tutte queste righe era
% improponibile.

% il primo valore di Correnti_R era 0.00210311469435100 ma l'ho messo a
% zero altrimenti dovevo fare anche l'estrapolazione
Correnti_R=[0,...,1]

% il primo valore di Radianze_R era 0.000874290637269546 ma l'ho messo a
% zero altrimenti dovevo fare anche l'estrapolazione
Radianze_R[0,...,1];

% il primo valore di Correnti_G era 0.000814025390771249 ma l'ho messo a
% zero altrimenti dovevo fare anche l'estrapolazione
Correnti_G=[0,...,1];

% il primo valore di Radianze_G era 0.00490497882171380 ma l'ho messo a
% zero altrimenti dovevo fare anche l'estrapolazione
Radianze_G[0,...,1];

% il primo valore di Correnti_B è stato impostato zero, perchè inizialmente
% era negativo (anche se molto piccolo)
Correnti_B[0,...,1];

% il primo valore di Radianze_R era 0.000557249129830264 ma l'ho messo a
% zero altrimenti dovevo fare anche l'estrapolazione
Radianze_B[0,...,1];



% interpolazione per ottenere il valore di corrente richiesto
```

```
Corr_R_LED=interp1(Radianze_R,Correnti_R,Terna_flussi(1));
Corr_G_LED=interp1(Radianze_G,Correnti_G,Terna_flussi(2));
Corr_B_LED=interp1(Radianze_B,Correnti_B,Terna_flussi(3));

Terna_correnti=[Corr_R_LED Corr_G_LED Corr_B_LED];
```

main_6259.m

```
%programma principale, che fa funzionare iterativamente la GUI e dà i
%valori in uscita al programma per il comando della scheda DAQ

clear
clc

fprintf('Inizio il programma\n');

Fine=1;
ciclo=0;

Terna=[0 0 0]; %inizializzo i valori a zero, così appena parte questo programma i LED re-
stano spenti
Terna_correnti_RGB=Terna; %inizializzo la terna delle correnti a zero
Terna_vecchia=Terna;

ApriPorte_6259
ComandoSchedaDAQ_6259


while Fine==1
    if Fine==0 break;
    end
        ciclo=ciclo+1;
    fprintf('\n\ninizio ciclo numero %d',ciclo);
    fprintf('\n\n');

    [Terna_flussi,Fine] = ColorSelector;
    fprintf('\nricevuta Terna_flussi nel main\n');

    Terna_nuova=Terna_flussi;

    if Terna_nuova==Terna_vecchia
        fprintf('\n la terna è uguale a quella del ciclo precedente\n')
        Terna_correnti_RGB=Terna_corretta;
        fprintf('\n la terna RGB che invio alla scheda DAQ è\n');
    Terna_correnti_RGB
        terna_cambiata=0;
    else terna_cambiata=1;
    fprintf('\n la nuova terna di flussi richiesta è\n');
    Terna_flussi

    Terna_correnti_RGB=Lookup_tables(Terna_flussi);

    fprintf('\n la nuova terna RGB che invio alla scheda DAQ è\n');
    Terna_correnti_RGB
    end

    fprintf('\nI valori sono stati scelti per la %d',ciclo);
    fprintf('° volta');

    ComandoSchedaDAQ_6259

    LetturaCorrentiTempProlungata_6259

%     VisualizzaValoriLetti

    if ciclo==1
        Terna_corretta=Terna_correnti_RGB; %inizializzo Terna_corretta
    end

    RetroazionePI_6259

%     le righe commentate qui sotto servono per monitorare l'andamento
%     delle tensioni di controllo degli alimentatori in base alla temperatura;
%     le lascio, ma ora non servono più.
```

```
%     Temp(ciclo)=TemperaturaReale(1);
%     TensR(ciclo)=CtrlTensRedAgg;
%     TensG(ciclo)=CtrlTensGreenAgg;
%     TensB(ciclo)=CtrlTensBlueAgg;

    Terna_vecchia=Terna_nuova;

end

ChiudiPorte_6259

fprintf('\n\nfine programma\n');
```

RetroazionePI_6259.m

```
if terna_cambiata
    SetPoint=Terna_correnti_RGB;
    fprintf('\ncambio Set Point\n');
end

toll=3e-4*ones(1,3);   %tolleranza ammessa

err=ones(1,3); %inizializzo l'errore, con un valore superiore a toll, in modo tale che si
entri nel ciclo while

err_int=0*ones(1,3); %inizializzo l'errore integrale, che poi aumenterà ciclo dopo ciclo

Kp=0.8;
Ki=0.5;
Ka=0.6;  %per avere 600mA con RGB=1
beta=1;

ciclo_retroaz=0;
max_cicli=50;

if ~isequal(SetPoint,[0 0 0])   %per fare in modo che la retroazione non venga eseguita
alla chiusura del programma

% se la corrente richiesta è zero, aumento la tolleranza,
% perchè a LED spenti l'offset della scheda DAQ porterebbe la retroazione a non finire
mai.
    if SetPoint(1)==0
        toll(1)=5e-3;
    end
    if SetPoint(2)==0
        toll(2)=5e-3;
    end
    if SetPoint(3)==0
        toll(3)=5e-3;
    end

    %entro nel ciclo di retroazione se almeno una delle correnti è al di
    %fuori della tolleranza
    while (abs(err(1))>=toll(1)||abs(err(2))>=toll(2)||abs(err(3))>=toll(3))&&ciclo_re-
troaz<max_cicli

        LetturaCorrentiTempProlungata_6259
        VisualizzaValoriLetti

    feedback=[CorrenteRossoMed CorrenteVerdeMed CorrenteBluMed].*(beta/Ka);

    err=(SetPoint-feedback);

    if max(abs(err))>=min(toll)
```

```
%         err_rel=err./SetPoint;

    fprintf('\nerrore oltre la tolleranza: necessaria correzione\n');

        ciclo_retroaz=ciclo_retroaz+1;

        err_int=err_int+err; %calcolo l'errore integrale, come somma di tutti gli errori
misurati precedentemente
        err_int=min(err_int,5*ones(1,3)); %limito l'errore integrale, perchè altrimenti
ci mette una vita a "scaricarlo"

        Correz=err.*Kp+err_int.*Ki;

%         Terna_corretta_new=Ka.*(Terna_corretta+Correz) usando quella
%         sotto al posto di quella sopra, la retroazione non inizia se
%         richiedo la stessa terna
        Terna_corretta_new=Terna_corretta+Correz

        CtrlTensRedAgg=-2.55*(Terna_corretta_new(1))+4.2;
        CtrlTensGreenAgg=-2.55*(Terna_corretta_new(2))+4.2;
        CtrlTensBlueAgg=-2.55*(Terna_corretta_new(3))+4.2;

        putsample(ao1,[CtrlTensRedAgg CtrlTensGreenAgg CtrlTensBlueAgg]);

        if ciclo_retroaz>0
        Terna_corretta=Terna_corretta_new;
        end

    else fprintf('\nerrore entro la tolleranza: correzione non necessaria\n');
    end


    end

    if terna_cambiata
    Terna_corretta=Terna_corretta_new;
    end
    fprintf('\nla terna corretta è \t');
    Terna_corretta

    if ciclo_retroaz==max_cicli
    fprintf('\nColore non raggiunto - retroazione interrotta\n');
    else fprintf('\nColore raggiunto - retroazione conclusa in %2.0f',ciclo_retroaz);
        fprintf(' cicli\n');
    end

else fprintf('\nSetPoint nullo: non effettuo retroazione\n');
    Terna_corretta=Terna_correnti_RGB;
end
```

Trasf_2_RGBled.m

```
%in questa function si trasformano i tre valori di ingresso (ovvero i
%valori impostati tramite le caselle o gli sliders) nella terna RGB che
%serve per comandare i LED

function [Terna Terna_sRGB] = Trasf_2_RGBled (SpazioColore, Val1, Val2, Val3)


radice=cd;
k=length(radice);
while radice(k)~='\'
   k=k-1;
   radice=radice(1:k);
end

Km=683; %(lm W-1)
nm=1e-9;%nanometro
```

```
%funzione x
fid=fopen([radice 'dati\Col_mat_fun_x'],'r');
for n=1:8
   line=fgetl(fid);
end
Nl=str2double(fgetl(fid));
a=fscanf(fid,'%g %g',[2,inf]);
fclose(fid);
lam=a(1,:); %lunghezza d'onda banda visibile (m)
x1=a(2,:); %sensibilità spettrale dell'occhio umano (-)
clear fid line a

%sensibilità dell'occhio umano V(lambda) - funzione y
fid=fopen([radice 'dati\Col_mat_fun_y'],'r');
for n=1:8
   line=fgetl(fid);
end
Nl=str2double(fgetl(fid));
a=fscanf(fid,'%g %g',[2,inf]);
fclose(fid);
lam=a(1,:); %lunghezza d'onda banda visibile (m)
y1=a(2,:); %sensibilità spettrale dell'occhio umano (-)
clear fid line a

%funzione z
fid=fopen([radice 'dati\Col_mat_fun_z'],'r');
for n=1:8
   line=fgetl(fid);
end
Nl=str2double(fgetl(fid));
a=fscanf(fid,'%g %g',[2,inf]);
fclose(fid);
lam=a(1,:); %lunghezza d'onda banda visibile (m)
z1=a(2,:); %sensibilità spettrale dell'occhio umano (-)
clear fid line a
%y1=y1(21:421);

%coordinate X
fid=fopen([radice 'dati\Chrom_coor_x'],'r');
for n=1:8
   line=fgetl(fid);
end
Nl=str2double(fgetl(fid));
a=fscanf(fid,'%g %g',[2,inf]);
fclose(fid);
lamC=a(1,:); %lunghezza d'onda banda visibile (m)
X1=a(2,:); %sensibilità spettrale dell'occhio umano (-)
clear fid line a

%coordinate Y
fid=fopen([radice 'dati\Chrom_coor_y'],'r');
for n=1:8
   line=fgetl(fid);
end
Nl=str2double(fgetl(fid));
a=fscanf(fid,'%g %g',[2,inf]);
fclose(fid);
lamC=a(1,:); %lunghezza d'onda banda visibile (m)
Y1=a(2,:); % (-)
clear fid line a

%coordinate Z
fid=fopen([radice 'dati\Chrom_coor_z'],'r');
for n=1:8
   line=fgetl(fid);
end
Nl=str2double(fgetl(fid));
```

```
a=fscanf(fid,'%g %g',[2,inf]);
fclose(fid);
lamC=a(1,:); %lunghezza d'onda banda visibile (m)
Z1=a(2,:); % (-)
clear fid line a

%densità spettrale iluminante D65 CIE
fid=fopen([radice 'dati\I_D65-CIE-1986_v001'],'r');
for n=1:8
   line=fgetl(fid);
end
Nl=str2double(fgetl(fid));
a=fscanf(fid,'%g %g',[2,inf]);
fclose(fid);
lamD=a(1,:); %lunghezza d'onda banda visibile (m)
D65=a(2,:); %sensibilità spettrale dell'occhio umano (-)
clear fid line a

D65=interp1(lamD,D65,lam);

%densità spettrale iluminante A CIE
fid=fopen([radice 'dati\I_A-CIE-1986_v001'],'r');
for n=1:8
   line=fgetl(fid);
end
Nl=str2double(fgetl(fid));
a=fscanf(fid,'%g %g',[2,inf]);
fclose(fid);
lamA=a(1,:); %lunghezza d'onda banda visibile (m)
A=a(2,:); %sensibilità spettrale dell'occhio umano (-)
clear fid line a

A=interp1(lamA,A,lam);

clear lamC lamD lamA

%_____________________________________________________________________

%_____________________________________________________________________

%_____________________________________________________________________

%identificazione della sorgente di riferimento
%!!!!!!!!!!!!!!!!! va scelta in funzione dell'illuminazione della
%!!!!!!!!!!!!!!!!! camera di osservazione
%_________ temporaneamente è D65
%_____________________________ da controllare lunghezza vettori
Sor_rif=D65; %radianza spettrale della sorgente di riferimento
% Sor_rif=A; %radianza spettrale della sorgente di riferimento
%lam_rif= ??????????????????????????????????????????????????????
%X_rif=(Sor_rif*interp1(lam,[x1;y1;z1]',lam_rif))'; ??????????????????

XYZ_rif(1,1)=x1*Sor_rif'; %valore tristimolo X della sorgente di riferimento
XYZ_rif(2,1)=y1*Sor_rif'; %valore tristimolo Y della sorgente di riferimento
XYZ_rif(3,1)=z1*Sor_rif'; %valore tristimolo Z della sorgente di riferimento
XYZ_rif=XYZ_rif*1000/XYZ_rif(2,1);

%---- matrice di trasformazione da CIE XYZ a RGB_LED ---
%     matrice di trasformazione da RGB_LED a CIE XYZ
%          lettura della radianza spettrale dei LED RGB della sorgente
           nomefile=[radice 'dati\spettriRGB.LMT'];
           nomefile_teor=[radice 'dati\SORG_RGB.LMT'];
           lam_s=[380:780]*nm;
           XSP=leggispettro(nomefile);
           XSP_teor=leggispettro(nomefile_teor);
           %calcolo dei valori tristimolo di RGB
           %valori tristimolo CIE XYZ
           XYZ_rgb=Km*(XSP(1:3,:)*interp1(lam,[x1;y1;z1]',lam_s))';
```

```
            XYZ_RGB_teor=Km*(XSP_teor(1:3,:)*interp1(lam,[x1;y1;z1]',lam_s))';

%       Matrice di trasformazione da RGB_LED a XYZ
            daRGBaXYZ=XYZ_rgb; %Le colonne sono i valori tristimolo di R,G,B

%       Matrice inversa per la trasformazione da CIE XYZ a RGB_LED
            daXYZaRGB=inv(daRGBaXYZ);

%       Matrice di trasformazione da RGB_teor a XYZ
            daRGB_teor_aXYZ=XYZ_RGB_teor; %Le colonne sono i valori tristimolo di R,G,B

%       Matrice inversa per la trasformazione da CIE XYZ a RGB_teor
            daXYZaRGB_teor=inv(daRGB_teor_aXYZ);

            %trasformazione da CIE XYZ a CIE Lxy (grafica)
Lxy_rgb=CieXYZ2CieLxy(daRGBaXYZ(1,:),daRGBaXYZ(2,:),daRGBaXYZ(3,:)); %i vertici del
triangolo RGB riportati nello spazio Lxy?

% (questa parte commentata l'ho spostata sotto: se funziona tutto, posso
% cancellarla)
% hold on
% grid on
% plot([X1 X1(1)],[Y1 Y1(1)],'k')    % disegna il bordo di Lxy (viene già
%                                    % fatto durante verifica_inpolygon)
% plot(Lxy_rgb(2,:),Lxy_rgb(3,:),'r*') %i vertici del triangolo RGB riportati nello spa-
zio Lxy

% hold off

switch SpazioColore
    case 1 %spazio RGB (lo lascio solo per comodità)
        Terna=[Val1, Val2, Val3];

        XYZ=XYZ_RGB_teor*Terna'; %conversione in XYZ per anteprima a monitor
        X=XYZ(1,:);
        Y=XYZ(2,:);
        Z=XYZ(3,:);

        RGBinLxy=CieXYZ2CieLxy(X,Y,Z); %conversione in Lxy per far vedere il punto nello
spazio Lxy

        plot(RGBinLxy(2),RGBinLxy(3),'.');


    case 2 %spazio Lxy
        XYZ=CieLxy2CieXYZ(Val1, Val2, Val3);
        X=XYZ(1,:);
        Y=XYZ(2,:);
        Z=XYZ(3,:);

        plot(Val2,Val3,'.'); %disegno sul grafico il punto selezionato, poi sotto disegno
tutto lo spazio Lxy


    case 3 %spazio Lab

  % trasformazione da CIE L*a*b* a CIE XYZ del punto neutro
    Lab_t=[100 0 0]';
    %Lab_t=[76.06925 0 0]';
    XYZ_t=CieLab2CieXYZ(Lab_t(1),Lab_t(2),Lab_t(3),XYZ_rif(1,:),XYZ_rif(2,:),XYZ_rif(3,:)
); %a cosa serve questa riga se poi non uso più il valore?

        XYZ=CieLab2CieXYZ(Val1, Val2, Val3,XYZ_t(1,:),XYZ_t(2,:),XYZ_t(3,:));
        X=XYZ(1,:);
        Y=XYZ(2,:);
        Z=XYZ(3,:);
```

```
        Lxy_Lab=CieXYZ2CieLxy(X,Y,Z); %conversione in Lxy del punto scelto in Lab, pas-
sando per XYZ
        plot(Lxy_Lab(2),Lxy_Lab(3),'.'); %disegno sul grafico il punto scelto in Lab,
nello spazio Lxy


    case 4 %spazio HSV

        Terna=HSV2RGB(Val1/360, Val2, Val3);
end

if SpazioColore~=1 && SpazioColore~=4
    %Ora che, partendo da qualsiasi spazio eccetto RGB e HSV, siamo arrivati allo spazio
XYZ, faccio la trasformazione da XYZ a RGB_LED
Terna=CieXYZ2RGB_led(X,Y,Z,daXYZaRGB);
end


%CONVERSIONE IN sRGB PER L'ANTEPRIMA A MONITOR

%Prova 1: XYZ -> sRGB (vengono valori di sRGB al di fuori di (0:1)
Terna_sRGB=CieXYZ2sRGB(X,Y,Z);

%%Prova 2: RGB -> sRGB esatta
% Terna_sRGB=RGB2sRGB(Terna);

% %Prova 3: RGB-> sRGB approssimata con esponenziale
% Terna_sRGB=(Terna).^(1/2.2);


%Questa parte disegna il bordo dello spazio Lxy e il gamut (triangolo RGB) riproducibile
con i nostri LED
hold on
grid on
plot([X1 X1(1)],[Y1 Y1(1)],'k')    % disegna il bordo dello spazio Lxy
plot(Lxy_rgb(2,:),Lxy_rgb(3,:),'r*') %i vertici del triangolo RGB riportati nello spazio
Lxy
plot([Lxy_rgb(2,:) Lxy_rgb(2,1)],[Lxy_rgb(3,:) Lxy_rgb(3,1)],'k') %il bordo del triangolo
RGB riportato nello spazio Lxy
hold off
```

verifica_inpolygon.m

```
function bordo = verifica_inpolygon (SpazioColore, val1, val2, val3)

switch SpazioColore
    case 1 %spazio RGB
        bordo=1;
    case 2 %spazio Lxy

        % 1. creazione della curva del bordo dello spazio Lxy
        [lam X Npunti]=leggi_x;
        [lam Y]=leggi_y;
        [lam Z]=leggi_z;

        x=X./(X+Y+Z);
        y=Y./(X+Y+Z);
        z=Z./(X+Y+Z);

        CoordLxy(1,:)=lam;
        CoordLxy(2,:)=x;
        CoordLxy(3,:)=y;
        CoordLxy(4,:)=Y;

        %questi valori mi servono per chiudere la curva
```

```
        CoordLxy(1,Npunti+1)=lam(1);
        CoordLxy(2,Npunti+1)=x(1);
        CoordLxy(3,Npunti+1)=y(1);
        CoordLxy(4,Npunti+1)=Y(1);

        %le righe qui sotto plottano la curva nello spezio Lxy e il punto
        %nella posizione selezionata. Probabilmente sarebbe meglio mettere
        %questi comandi in una function diversa, che faccia solo il plot
        %della posizione del punto. La rottura è che, anche in quella
        %function, dovrei rileggere leggi_x ecc. e rallenterebbe il
        %programma!

        plot(CoordLxy(2,:),CoordLxy(3,:),'k');
        grid on
        hold on
%         plot(xPunto,yPunto,'.');
%         hold off

    % 2. verifico se il punto desiderato è all'interno o meno del bordo dello
    % spazio Lxy (bordo=0 se è fuori, bordo=1 se è dentro)
    xPunto=val2;
    yPunto=val3;
    bordo=inpolygon(xPunto,yPunto,CoordLxy(2,:),CoordLxy(3,:));


    case 3 %spazio Lab
        bordo=1;

    case 4 %spazio HSV
end
```

VisualizzaValoriLetti.m

```
fprintf('\n\nValori acquisiti:');
    fprintf('\ncorrente media canale rosso %2.4f',CorrenteRossoMed);
    fprintf('\tA');

%     figure(11)
%     base=1:size(CorrenteRossoMis);
%     plot(base,CorrenteRossoMis,'r');

    fprintf('\ncorrente media canale verde %2.4f',CorrenteVerdeMed);
    fprintf('\tA');


%     figure(12)
%     plot(base,CorrenteVerdeMis,'g');

    fprintf('\ncorrente media canale blu %2.4f',CorrenteBluMed);
    fprintf('\tA');


%     figure(13)
%     plot(base,CorrenteBluMis,'b');

    fprintf('\ntensione sensore di temperatura %2.4f',TensTemperaturaMis(1));
    fprintf('\tV');
    fprintf('\ntemperatura calcolata %3.1f',TemperaturaReale(1));
    fprintf('\t°C');
```

Appendice B - Data sheets dei componenti

I data sheets dei componenti verranno stampati ed allegati alla copia cartacea, perché sono tutti in pdf e quindi non sono importabili nel file di testo.