UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA ELETTRICA

TESI DI LAUREA MAGISTRALE

REALIZZAZIONE DEL CONTROLLO DI TEMPERATURA IN UNA SORGENTE A LED A CROMATICITÀ VARIABILE

RELATORE: Prof. Pietro Fiorentin

CORRELATORE: Ing. Alessandro Scroccaro

LAUREANDO: Vettoretto Alberto, matr. 1034583.

A.A. 2014/2015

SOMMARIO

CAPITOLO 1: Cenni di teoria

1.1 La radiazione visibile

La radiazione visibile è una piccola parte dello spettro delle radiazioni elettromagnetiche le quali, come è possibile vedere dalla Figura 1, hanno un campo di impiego molto ampio,variabile a seconda dello scopo, che si differenzia per lunghezze d'onda e quindi energia che viene richiesta.

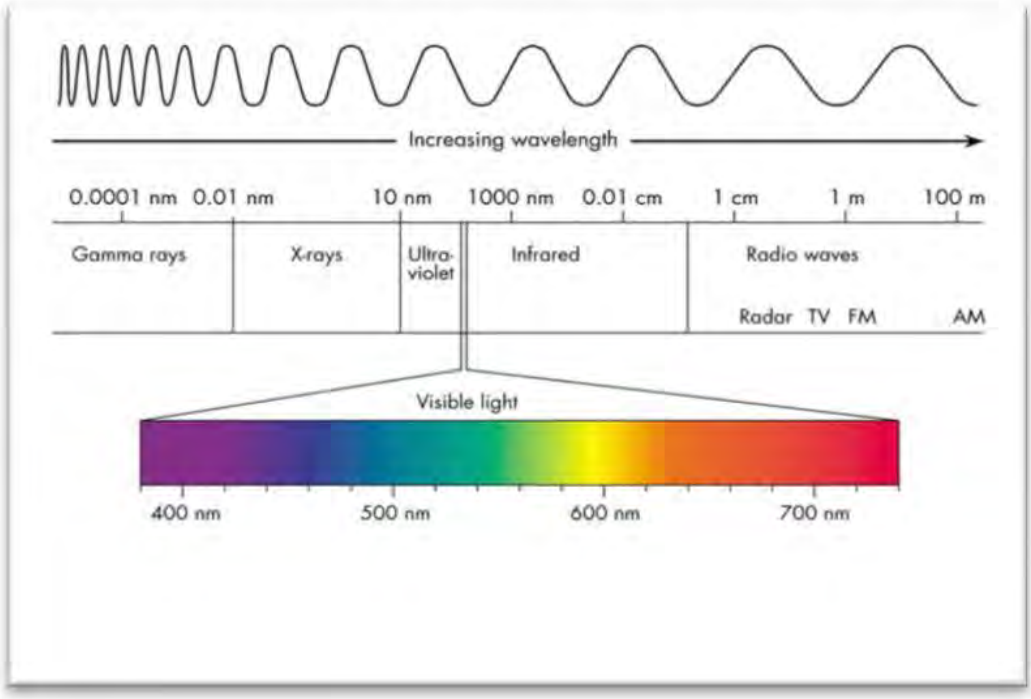


Figura 1: Spettro elettromagnetico

In particolare, secondo la Legge di Plank, la relazione tra energia emessa da un fotone in una radiazione elettromagnetica e la frequenza della stessa (e dunque la sua lunghezza d'onda $\lambda = 1/f$) può essere sinteticamente descritta come:

$$E = h * f$$

dove h è la costante di Plank ed è circa pari a 6,626*10^(-34) J*s ed f è la frequenza della radiazione elettromagnetica [Hz]. Dunque tanto più la lunghezza d'onda è grande, tanto minore è l'energia contenuta nella radiazione e viceversa.

Per questa Tesi, sono di nostro interesse una piccola porzione dello spettro elettromagnetico, centrata in un range di lunghezza d'onda λ copreso tra i 380nm (in cui troviamo il colore viola) e 780 nm (per cui abbiamo il colore rosso) oppure, come descritto dalla CIE

(Commission Internationale de l'Eclairage), tra 360 e 820 nm, che rappresentano il range delle radiazioni visibili dall'occhio umano e che emette energia sottoforma di radiazione luminosa di cromaticità variabile.

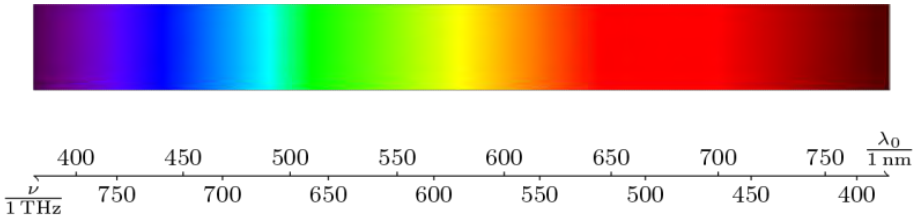


Figura 2: Spettro elettromagnetico visibile all'uomo

1.2 L'occhio umano

L'occhio umano, di cui faremo una descrizione sommaria per capirne intuitivamente il funzionamento, è composto da diversi componenti. Volendo seguire idealmente il percorso della radiazione luminosa quando questa incontra l'occhio umano, andremmo ad incontrare:

- Diottro (o Cornea);
- Lente Biconvessa (o Cristallino) la cui distanza focale può essere modificata mediante le ciglia;
- Iride: Muscolatura che consente di regolare l'entrata della luce all'interno della pupilla.

Questo insieme di elementi compone la lente convergente, la quale ha il compito di inviare le immagini rimpicciolite e capovolte alla retina.

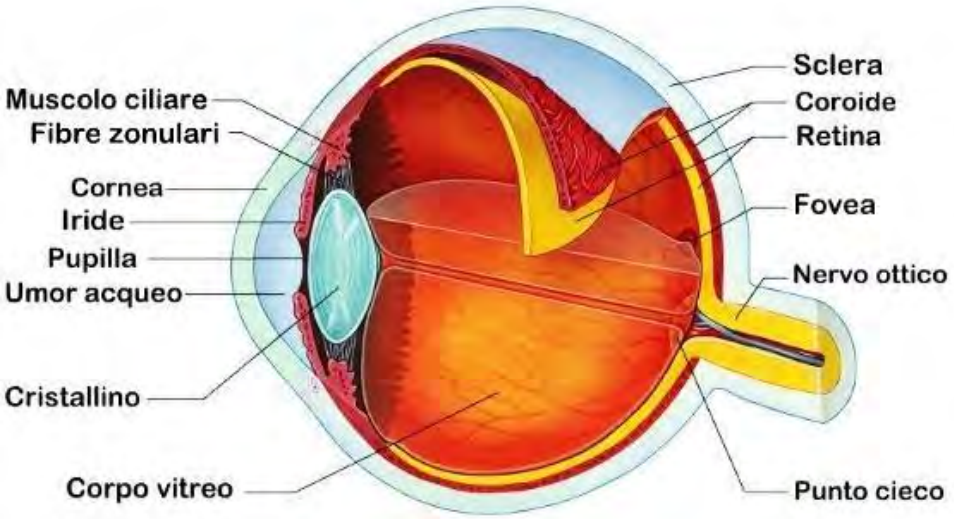


Figura 3: Occhio umano

La retina è composta da un insieme di fotorecettori distinguibili in due categorie:

- Bastoncelli: sono quelli che contribuiscono alla visione scotopica (cioè notturna, con bassi valori di luminanza), e che non riescono a distinguere i colori. Sono circa 120 milioni e sono distribuiti in tutta la retina (a parte nella zona centrale, dove vedremo sono situati i coni, e nella zona definita angolo cieco).
- Coni: Permettono la visione fotopica (diurna con elevati livelli di luminanza) e permettono di distinguere i colori. Sono 6 milioni e sono distribuiti nella parte centrale dell'occhio.

La retina, attraverso i neuroni retinici, portano il segnale al nervo ottico che lo trasporta a sua volta nelle aree del cervello dedite all'elaborazione dello stesso.
Parlando di una sorgente a cromaticità variabile, è facile intuire come la parte anatomica dell'occhio che più ci interessa sono i coni. In particolare questi sono suddivisibili in tre tipologie, che si differenziano in base alle diverse lunghezze d'onda a cui sono più sensibili:

- LUNGHE (L), Luce rossa (R).
- MEDIE (M), Luce Verde (G).
- CORTE (S), Luce Blu (B).

Possiamo quindi dire che la percezione cromatica di una luce è frutto dell'interazione tra la luce stessa e il sistema che abbiamo per percepirla, cioè l'insieme delle tre tipologie di coni che, come possiamo vedere dalla Figura 4, hanno delle curve di sensibilità spettrale diverse.

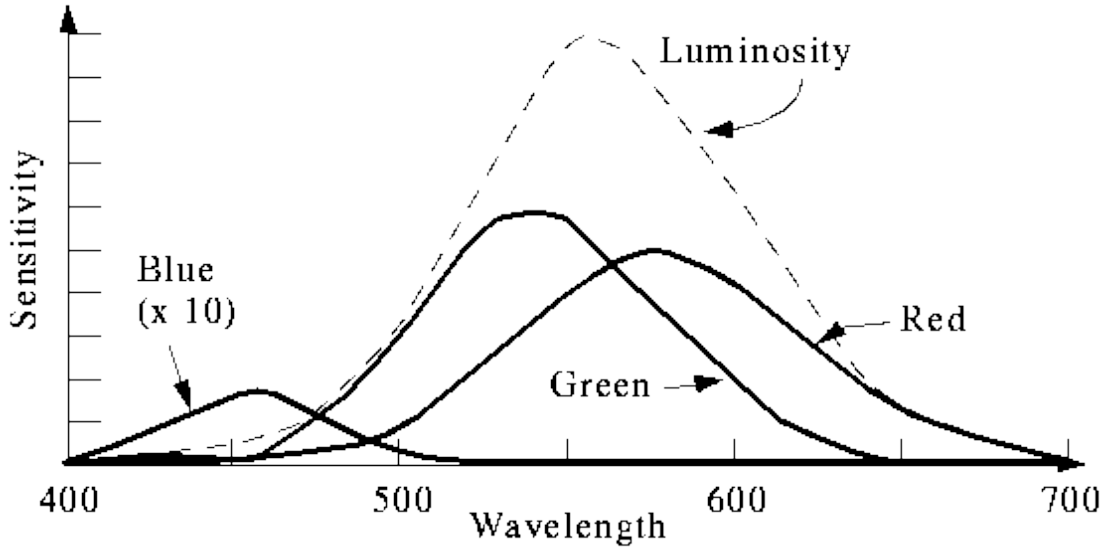


Figura 4: Curve di sensibilità spettrale dei coni.

Questo grafico fa intuire come per poter avere due sorgenti luminose che presentano la stessa luminosità, ma una con lunghezza d'onda legata alla percezione ad esempio del colore verde (in cui c'è la massima sensibilità) e l'altra con lunghezza d'onda legata alla percezione ad esempio del colore blu, necessitiamo di una potenza emessa molto più grande in quest'ultima che non nella sorgente verde, a causa del diverso livello di sensibilità spettrale delle due tipologie di coni.

La curva tratteggiata, che rappresenta la totale sensibilità spettrale dell'occhio umano, permette di intuire che quest'ultimo presenta la massima sensibilità con radiazioni di lunghezza d'onda che si aggirano nell'intorno dei 555 nm. Possiamo inoltre dedurre che, se consideriamo una sorgente centrata su un'unica lunghezza d'onda e operiamo una variazione della stessa, l'occhio umano tenderà a percepire delle variazioni di colore, applicando alla sorgente delle variazioni di lunghezza d'onda sempre più piccole quanto più grande è la sensibilità spettrale dell'occhio.

1.3 Grandezze fotometriche

Quando si parla di sorgenti luminose, le unità di misura del Sistema Internazionale (SI) non sono più sufficienti a descrivere la natura dei fenomeni che ne stanno alla base. È necessario

quindi introdurre delle unità di misura apposite, di cui daremo una breve spiegazione. Inanzitutto c'è da dire che ad ogni grandezza fotometrica x_v corrisponde una grandezza radiometrica x_e; la grandezza fotometrica può essere definita come la corrispondente grandezza radiometrica pesata con la curva di sensibilià fotopica dell'occhio umano V(λ):

$$Xv = Km * \int_{\lambda min}^{\lambda max} Xe(\lambda) * V(\lambda)\ d\lambda$$

dove Km = 683 [lumen/W] è un fattore che consente di avere un legame con l'unità di misura delle candele [cd], unità di misura dell'intensità luminosa.

1.3.1 Flusso Luminoso

Il flusso luminoso può essere definito come la quantità di radiazioni emesse nell'unità di tempo da una sorgente primaria o secondaria pesate con la curva fotopica.

$$Xv = Km * \int_{\lambda min}^{\lambda max} P(\lambda) * V(\lambda)\ d\lambda$$

Il flusso luminoso è quindi associato alla potenza luminosa emessa dalla sorgente pesata con la curva fotopica. Possiamo quindi intuire come una lampada che presenta uno spettro di potenza distribuito anche su lunghezze d'onda in cui la sensibilità fotopica è bassa, non presenta per tali range di λ alcuna utilità ai fini dell'illuminazione vera e propria, ma ci consente comunque di distinguere i colori.

Lo strumento che viene utilizzato per la misura del flusso luminoso è la sfera integratrice, la quale internamente è rivestita di una speciale vernice che consente di avere un coefficiente di riflessione pressochè unitario e permette una diffusione il più possibile uniforme della radiazione riflessa; in questo modo l'illuminamento è idealmente costante in tutti i punti della sfera, perdendo quindi la direzionalità dettata dalla lampada. All'interno della sfera sono inoltre compresi sia un sensore, a cui interponiamo un setto di copertura così da impedere che le radiazioni luminose colpiscano direttamente il sensore stesso, e una lampada ausiliaria di cui conosciamo sia il flusso luminoso che l'illuminamento. Avendo quindi una relazione proporzionale tra flusso emesso e illuminamento ed essendo noti i parametri della lampada ausiliaria (r), posso ricavare il flusso emesso dalla lampada in prova (t: test) come:

$$Et = K * \phi t \qquad\qquad Er = K * \phi r$$

$$\phi t = \frac{Et}{Er} * \phi r$$

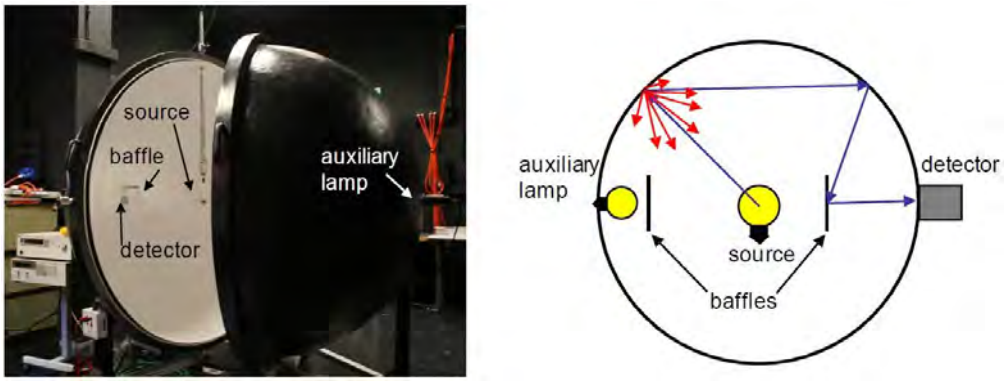


Figura 5: Sfera integratrice

È anche vero che come è possibile vedere dal grafico che segue (Figura 6), le pareti interne devono avere da una parte un coefficiente di riflessione che sia il più possibile vicino all'unità, ma è anche vero che se per qualsiasi motivo il coefficiente di riflessione dovesse decrescere (es. superfici non perfettamente pulite), questo comporterebbe una caduta molto più accentuata che non adottando un coefficiente di riflessione leggermente inferiore ad 1 (specie quando il foro per il sensore risulta molto piccolo). Questo sta inoltre ad indicare come la superficie interna della sfera deve mantenere sempre un livello di pulizia elevato per garantire le prestazioni desiderate.

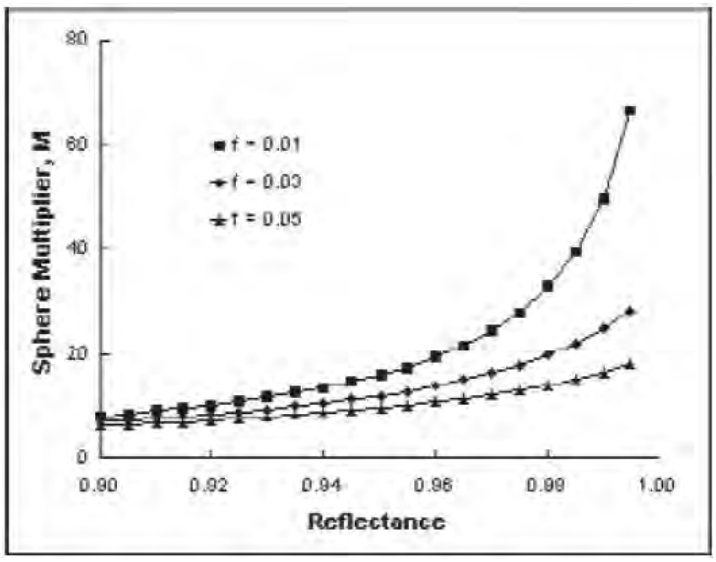


Figura 6: Misura della sfera integratrice in funzione del coefficiente di riflessione (parametrizzato secondo il grado di pulizia)

1.3.2 Luminanza

La luminanza può essere definita come rapporto tra l'intensità luminosa (flusso emesso per unità di angolo solido dΩ) emessa da una superficie e la sua area apparente, cioè la superficie pesata con l'angolo d'osservazione θ:

$$L = \frac{dI}{dApp} = \frac{d\phi^2}{d\Omega(dA * cos\theta)} \ [cd/m^2]$$

La corispondente grandezza radiometrica è la radianza, la quale viene misurata in [W/(sr * m^2)].

Le sorgenti luminose, anche secondarie come ad esempio una superficie riflettente, sono spesso di tipo lambertiano e garantiscono quindi che l'intensità luminosa (dϕ/dω) sia descrescente con l'angolo d'osservazione e garantendo quindi una luminanza costante, eliminando di fatto la dipendenza della stessa con l'angolo d'osservazione.

La luminanza è contraddistinta inoltre da due proprietà molto importanti:

- La luminanza è indipendente dalla distanza.
- Conservazione della luminanza: la luminanza è una quantità che si conserva nel sistema; dunque è sempre uguale, anche nella retina che opera una conversione della lunìminanza in illuminamento, reagendo alla densità di flusso.

Da quest'ultima è facile intuire come la luminanza sia estremamente importante nell'interazione tra la luce e l'occhio umano e che rappresenti di fatto quella che viene definita comunemente luminosità.

Lo strumento che viene utilizzato per la misura della luminananza è il luminanzometro, nel quale la luminanza di una certa superficie viene convogliata attraverso un sistema di lenti ad un elemento sensibile a cui viene interposto un filtro fotopico; tale elemento sensibile, essendo comunque colpito dalla stessa luminanza dell'oggetto in esame produrrà, convertendo tale luminanza in illuminamento, una corrente proporzionale alla luminanza stessa.

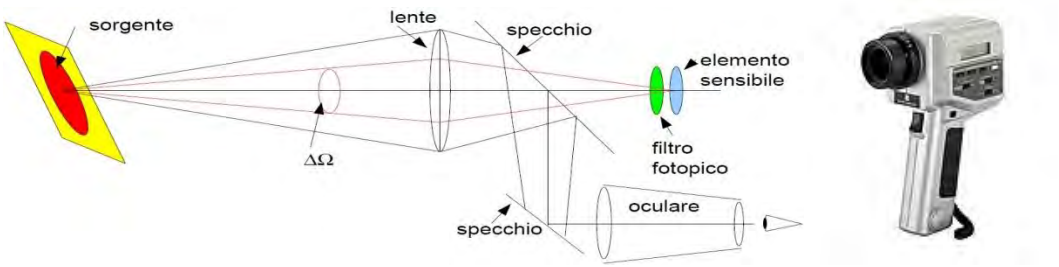


Figura 7: Luminanzometro e suo principio di funzionamento

Questo strumento non consente però di conoscere la radianza spettrale (cioè lunghezza d'onda per lunghezza d'onda), e per fare questo occorre servirsi di uno spettroradiometro. Il principio di funzionamento si basa sulla diffrazione della luce che viene operata attraverso un sistema definito monocromatore, che permetterà inoltre che l'elemento fotosensibile sia di volta in volta colpito da una radiazione di lunghezza d'onda diversa alla volta. Di tale strumento ci siamo serviti anche per alcune prove inerenti a questa tesi, in particolare lo strumento utilizzato è il MINOLTA CS-1000.

Figura 8: Spettroradiometro Minolta CS-1000

1.3.3 Illuminamento

L'illuminamento è definito come il rapporto tra il flusso luminoso incidente su una superficie e l'area della superficie stessa.

$$Ev = \frac{d\phi v}{dA}$$

l'unità di misura è il lux [lx].

Tale grandezza è direttamente legata alla luminanza vista in precedenza attraverso un coefficiente di luminanza q:

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

il quale presenta, nel caso di superficie lambertiana quale è la sfera integratrice, un valore pari a:

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

con ρ coefficiente di riflessione.

Quindi possiamo affermare che, nel caso di un diffusore perfetto, la luminanza può essere valutata come:

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

1.4 Colorimetria

Quando si parla di colore e della percezione umana dello stesso, alla base di tutto come abbiamo visto ci sono le tre tipologie di coni che ci permettono di esprimere un generico colore come la combinazione di tre colori primari. Questo vuol dire che idealmente qualsiasi colore può essere rappresentato in uno spazio tridimensionale, le cui coordinate sono legate alla quantità di colore primario impiegato per realizzarlo.

1.4.1 Leggi di Grassman (ridefinite da Hunt)

Alla base della colorimetria ci sono le tre leggi di Grassman, le quali specificano che:

- Tre grandezze specificano il colore;
- La sensazione di colore è proporzionale allo stimolo (quindi alla potenza);
- Una mescolanza additiva di colori dipende solo dal loro aspetto (non quindi dalle lunghezze d'onda delle radiazioni in gioco, ma della percezione che ha l'occhio umano di esse).

Queste sono state poi ridefinite da Hunt:

- Una sensazione di colore è completamente specificata da tre grandezze: la tinta (il colore), l'intensità del colore (la saturazione) e l'intensità di bianco (la luminosità);
- Se una luce varia con continuità varia anche la mescolanza additiva;
- Il risultato di una mescolanza additiva dipende solo dal suo aspetto e non dalla composizione fisica degli elementi che la compongono (i diversi spettri). Quindi una sensazione di colore può essere riprodotta utilizzando delle sorgenti con distribuzione spettrale diversa.

Queste leggi sono alla base del lavoro che è stato sviluppato per realizzare la sorgente a cromaticità variabile, di cui questo lavoro di tesi si occupa. Infatti le sensazioni di colore sono state ottenute mediante sintesi additiva di tre colori (rosso,verde e blu) ricreati attraverso l'utilizzo di LED.

1.5 Modulazione della larghezza d'impulso (PWM)

La tecnica di modulazione a larghezza d'impulso PWM è una tecnica utilizzata nei convertitori dc-dc per ottenere da una tensione di ingresso continua di valore fisso, una tensione sempre continua ma con valore regolabile.
Tale tecnica si basa, attraverso l'utilizzo di circuiti che confrontano dei segnali di riferimento, sulla variazione del fattore di intermittenza o di utilizzazione (duty cycle) D, che viene definito come il rapporto tra il tempo per cui la tensione d'uscita è perfettamente pari a quella d'ingresso grazie alla chiusura di un interruttore t_{on} e il periodo di commutazione totale Ts, pari alla somma tra il tempo di accensione e di spegnimento.

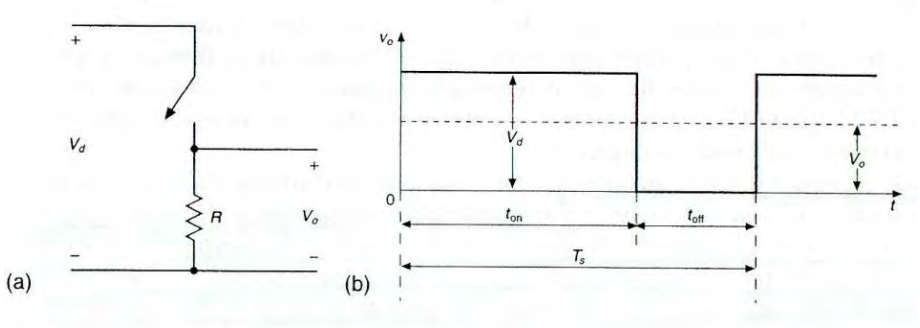


Figura 9: Fuzionamento di base di una PWM ed andamento della tensione al carico puramente resistivo

A questo punto, analizzando un carico puramente resistivo, applicando come in figura una tensione di ingresso continua Vd, la tensione media d'uscita Vo potrà essere calcolata in funzione del duty cycle come:

$$Vo = \frac{1}{Ts} * \int vo(t)dt = \frac{1}{Ts} * \left(\int_0^{ton} Vd\ dt + \int_{ton}^{Ts} 0\ dt \right) = \frac{ton}{Ts} * Vd = D * Vd$$

è possibile vedere come regolando il solo duty cycle è possibile ottenere una tensione media d'uscita che è minore o uguale alla tensione d'ingresso. Il filtro consente di eliminare le componenti di tensione in alta frequenza e di ottenere in uscita una tensione pressochè continua e pari al valore medio Vo precedentemente calcolato.

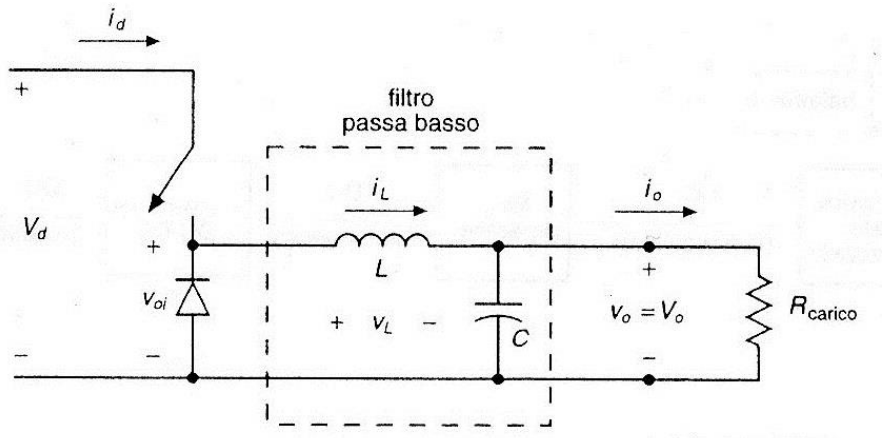


Figura 10: Fuzionamento di base di una PWM con l'inserzione di un filtro passa basso

1.6 Filtri passa basso RC del primo ordine

Il filtro passa basso del primo ordine è un dispositivo, composto da una resistenza ed una capacità posti in serie, a cui viene applicata una tensione di ingresso Vin; ai capi del condensatore una tensione d'uscita Vout.

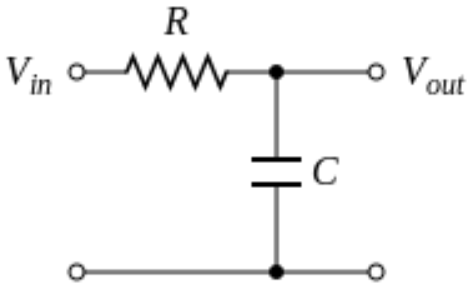


Figura 11: Filtro RC passa basso

Il funzionamento del filtro si basa sulla variazione dell'impedenza che presenta il condensatore al variare della frequenza d'ingresso del segnale; infatti per frequenze elevate il condensatore presenta impedenza ridotta e quindi ai suoi capi abbiamo una tensione nulla, a frequenze basse invece l'impedenza sale e quindi la tensione rilevata avrà valori confrontabili con quella d'ingresso.

La grandezza caratteristica è la frequenza di taglio ft, che rappresenta la frequenza del segnale d'ingresso tale per cui il valore del segnale in uscita si riduce di un valore pari a 3 dB; questo valore è legato alla definizione del modulo di una certa grandezza nell'ambito del tracciamento dei Diagrammi di Bode; infatti il modulo viene definito come:

$$X_{db} = 20 * \log X$$

Nel nostro caso tale valore è pari alla riduzione di 3 dB; facendo l'operazione inversa è possibile ricavare che la variazione legata a 3 dB è pari a:

$$X = 10^{\frac{3}{20}} \sim \sqrt{2}$$

La frequenza di taglio è legata ai parametri RC che vengono scelti e sarà pari a:

$$ft = \frac{1}{2 * \pi * R * C}$$

il filtro passa basso viene utilizzato per filtrare le uscite PWM di Arduino che ci consentiranno di comandare sia i pin ctrl dei driver di alimentazione che forniranno la corrente richiesta ai LED e ai riscaldatori, ma anche per comandare il morsetto gate del MOSFET che ci consentirà di gestire la ventola in modalità ON-OFF. Inoltre viene anche utilizzato per filtrare la tensione che viene misurata ai resistori di shunt, nell'ambito delle misure di corrente

1.7 Sistemi in catena aperta e in catena chiusa

Prendiamo in considerazione un processo descritto, nel dominio delle Laplace trasformate, attraverso la funzione G(s); l'obiettivo nella realizzazione di un sistema controllato è quello di ottenere un controllore C(s) tale che il sistema retroazionato di figura:

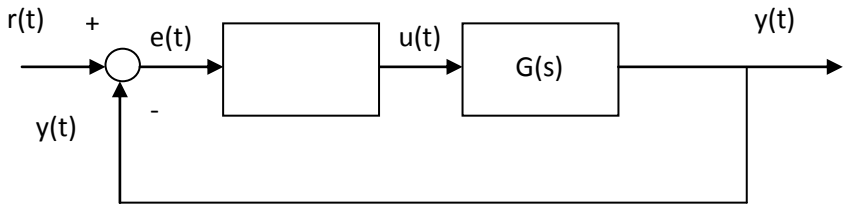


Figura 12: Schema del controllo in retroazione

la cui funzione di trasferimento è:

$$W(s) = \frac{C(s) * G(s)}{1 + C(s) * G(s)}$$

sia BIBO stabile (cioè ad un segnale di ingresso limitato corrisponde un'uscita anch'essa limitata) e rispetti i seguenti requisiti:

- il sistema retroazionato sia di tipo k: quando un sistema è di tipo k segue, con valore di errore di regime permanente costante e non nullo, il segnale canonico:

$$\delta_{-(k+1)}(t) = \frac{t^k}{k!} * \delta_{(-1)}(t)$$

- errore di regime permanente erp^{k+1} non superiore ad un massimo valore tollerato erp^*.
- la risposta al gradino abbia un tempo di salita tr circa pari ad un valore desiderato tr^*.
- una massima sovraelongazione s non superiore ad un massimo valore tollerabile s^* ed un tempo d'assestamento ts non superiore ad un valore massimo prestabilito ts^*.

I primi due obiettivi sono relativi al funzionamento in regime permanente, dunque per valori elevati di tempo t; le ultime due riguardano invece la fase tranitoria e mirano a quantificare la prontezza del sistema rispetto al segnale che maggiormente mette in evidenza tali aspetti, cioè il gradino.

Tali obiettivi possono essere riscritti analizzando il processo non più dal punto di vista temporale ma di quello della risposta in frequenza del sistema e quindi nel dominio delle Laplace trasformate come:

- per ottenere un sistema di tipo k dovrò avere:

1. k=0: W(0)=0
2. k>0: W(0)=1 con $\frac{dW(0)^i}{ds^i} = 0; \frac{dW(0)^k}{ds^k} \neq 0 \; con \; i < k$

- errore di regime permanente $erp^{k+1} \leq erp^*$: l'errore a regime permanente del sistema è pari a:

1. k=0: $erp^1 = 1\text{-}W(0)$;
2. k>0: $erp^{k+1} = -\frac{1}{k!} * \frac{dW(0)^k}{ds^k}$;

- Tempo di salita al di sotto di una certa soglia tr*: questo si traduce nel realizzare un sistema W(s) che abbia una banda passante a 3dB che sia la più ampia possibile. Infatti il tempo di salita è inversamente proporzionale alla banda passante a 3dB del sistema retroazionato.
- Limitazione della sovraelongazione ad un valore massimo s*: questo si traduce nell'avere uno scostamento massimo tra il picco del sistema ad anello chiuso W(jωr) ad una certa pulsazione ωr e il valore assunto in regime permanente W(0), limitato al valore prescelto come scostamento massimo.

Tutto quello che abbiamo appena detto per il sistema in catena chiusa, trova anche una corrispondenza per il sistema in catena aperta, descritto dall'espressione:

$$\tilde{G}(s) = C(s) * G(s)$$

Tale sistema dovrà rispettare infatti alcune specifiche ai fini di ottenere un andamento soddisfacente in catena chiusa:

- Tipo k: il sistema in catena aperta $\tilde{G}(s)$ ha un polo in 0 di molteplicità k.
- $erp^{k+1} \leq erp^*$: l'errore in regime permanente della catena aperta è pari a:

1. k=0: $erp^1 = \frac{1}{Kb+1}$
2. k>0: $erp^{k+1} = \frac{1}{Kb}$

- tr ≤ tr*: Garantire una pulsazione di attraversamento ω_A che abbia un valore paragonabile ad un valore preso come riferimento ω_A*. Tale pulsazione d'attraversamento ω_A*, viene scelta in modo che il sistema retroazionato sia il più reattivo possibile (tanto più elevata è la pulsazione d'attraversamento tanto più il sistema è pronto), senza però che lo stesso presenti un'instabilità nella risposta.
- sovraelongazione s inferiore ad un valore massimo s*: questo si traduce nel garantire un margine di fase mψ che sia al di sopra di un valore minimo mψ* (tipicamente 45°). Quando si parla di margine di fase si intende la quantità:

$$\text{m}\psi = 180 \ + \arg\left(\tilde{G}(j * \omega_A)\right) con\ \omega_A\ pulsazione\ d'attraversamento$$

Per ottenere gli obiettivi appena citati, si opera una sintesi che ci consenta di progettare il controllore che meglio possa approssimare poi nella realtà i risultati che vogliamo ottenere.

Il controllore PID (Proporzionale Integrale Derivativo) è una particolare tipologia di controllore che realizza sul segnale di ingresso (nel nostro caso l'errore e(t), frutto della differenza tra il riferimento r(t) e l'uscita del sistema y(t)) la combinazione lineare di tre azioni, appunto proporzionale, integrale e derivativa. Nel dominio del tempo la risposta in uscita dal controllore è:

$$u(t) = Kp * e(t) + Ki * \int \text{e(t) dt} + \text{Kd} * \frac{\text{de(t)}}{\text{dt}}$$

Dunque il controllore PID può essere descritto con una funzione di trasferimento nel dominio delle Laplace trasformate attraverso la funzione:

$$C_{PID}(s) = Kp + \frac{Ki}{s} + Kd * s = \frac{Kp * s + Ki + Kd * s^2}{s} = Ki * \frac{1 + \frac{Kp}{Ki} * s + \frac{Kd}{Ki} * s^2}{s}$$
$$= Kp * \left(1 + \frac{1}{s * Ti} + Td * s\right)$$

Dove Ti = Kp/Ki rappresenta la costante di tempo dell'azione integrale e Td = Kd/Kp quella dell'azione derivativa.
È possibile notare quindi come questa tipologia di controllore consenta di utilizzare il polo all'origine per poter ottenere il tipo di sistema desiderato, il guadagno di Bode Ki per avere

un errore a regime permanente minore o al massimo uguale a quello desiderato e i due zeri per aggiustare la posizione della pulsazione di attraversamento ω_A e il margine di fase $m\psi$, e garantire quindi le prestazioni desiderate durante il transitorio e quindi negli istanti iniziali.

1.8 Tecnologia LED (Light Emitting Diode)

La tecnologia LED (Light Emitting Diode) è una tipologia di sorgente luminosa che può essere paragonata, come metodologia di funzionamento, ad un diodo.
Infatti presenta anch'esso una ginzione p-n che, se polarizzata direttamente, consente la migrazione di una parte di elettroni della banda di conduzione alla banda di valenza, dove avverrà la ricombinazione con le lacune; tale variazione di livello energetico comporta il rilascio di energia sottoforma di fotoni, la cui cromaticità dipende dai materiali impiegati per realizzare la giunzione pn.
Questa tecnologia presenta sicuramente dei vantaggi sia a livello di compattezza ma anche di caratteristiche tecniche; infatti il LED, come noto, ha un'efficienza luminosa alta (80-110 lm/W per il LED bianco), sono regolabili in corrente e producono dei colori pressochè saturi, privi di componenti ultravioletti e infrarossi (inutili ai fini pratici dell'illuminazione artificiale), per un tempo di vita medio molto elevato (50000 ore).
D'altro canto è anche vero che anche questa tecnologia presenta delle lacune; infatti il ridotto flusso emesso rende necessario l'accorpamento di più unità. Inoltre per poter funzionare i LED necessitano di apparecchiature ausiliarie e, cosa più importante per questo lavoro di tesi, è che l'emissione dei LED è influenzata dalla temperatura di esercizio.

1.9 MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor)

Il MOSFET è un dispositivo comandato in tensione, il quale può comportarsi come un circuito aperto tra i morsetti D (DRAIN) e S (SOURCE), quando tra i morsetti G (GATE) E SOURCE viene applicata una tensione al di sotto della tensione di soglia $V_{GS\ th}$, mentre se applichiamo una tensione che superi tale valore, il circuito consentirà, il passaggio di una certo valore di corrente Ids, tanto maggiore quanto più alta è la tensione tra gate e source.

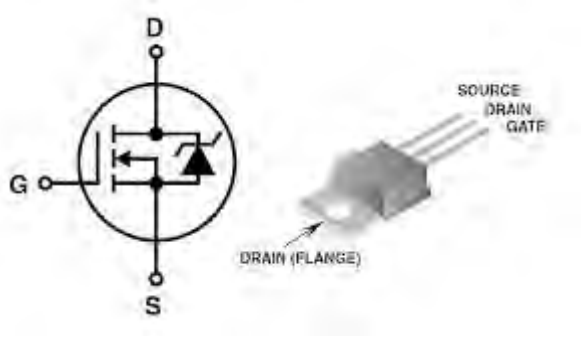


Figura 13: Schema circuitale del MOSFET e piedinatura del MOSFET in dotazione per il lavoro svolto

A differenza di altri tiristori che necessitano solamente di un impulso per poter condurre (ad esempio i GTO), i MOSFET necessitano dell'applicazione continua di tensione tra GATE e SOURCE. Altra cosa fondamentale è che tra GATE e SOURCE c'è la possibilità che in fase di commutazione ci sia circolazione di corrente legata alla scarica o alla carica della capacità di gate. Ai fini pratici è quindi consigliabile introdurre una resistenza di valore elevato (ad esempio 10 kΩ) tra GATE E SOURCE ai fini di evitare che la carica o scarica della capacità di GATE avvenga in modo distruttivo.

Altra cosa importante a livello costruttivo (che spesso troviamo nei MOSFET, anche in quello utilizzato per il comando ON OFF della ventola di raffreddamento) è che la piastra metallica esterna del MOSFET sia collegata al morsetto di DRAIN; è opportuno quindi che se il MOSFET viene fissato ad una superficie metallica (come la piastra di montaggio in alluminio presente nella sorgente in esame), la flangia metallica del MOSFET sia opportunamente isolata, così da evitare che la superficie metallica su cui il MOSFET viene fissato vada in tensione e, se questa è collegata a terra, portare al drenaggio di una parte della corrente, che non circolerà più quindi tra DRAIN e SOURCE.

CAPITOLO 2: Visione d'insieme del sistema preesistente

Questo lavoro di tesi tratta un oggetto già esistente, quindi prima di entrare nel merito del lavoro svolto, è necessario capire il funzionamento di tale sistema e dei componenti che lo formano.
Ciò con cui abbiamo a che fare è essenzialmente una sorgente luminosa che, sfruttando le leggi di Grassman viste in precedenza, consente di ricreare attraverso sintesi additiva una certa gamma di radiazioni luminose di diversa cromaticità. Le sorgenti luminose scelte sono dei LED rossi, verdi e blu inseriti all'interno di una sorta di sfera integratrice che consente il mescolamento delle diverse radiazioni luminose prodotte dai singoli gruppi di LED, creando una soluzione omogenea, la cui cromaticità dipende dal diverso contributo fornito dai LED stessi.

Nella sfera sono montati dei sensori (TAOS TCS230) che rispondono, ad una certa radiazione luminosa, con una segnale di frequenza variabile in uscita ed è inoltre presente nella piastra di montaggio dei LED un sensore di temperatura che ci consente di rilevare la temperatura della piastra di montaggio dei LED, convertendo la stessa in una corrente.

L'alimentazione dei LED avviene attraverso i driver LUXDRIVE 3021 D-E-700 BuckPuck comandati, a seconda della corrente che vogliamo inviare ai LED, con un segnale al morsetto di controllo (ctrl). Tale segnale viene fornito attraverso le uscite PWM, opportunamente filtrate, della scheda Arduino; la stessa scheda Arduino mette a disposizione inoltre degli ingressi analogici che consentono di rilevare dei segnali dall'esterno, opportunamente filtrati, da poter elaborare. Nel caso della sorgente in esame tali ingressi sono stati utilizzati per misurare le tensioni del morsetto ctrl dei driver, le correnti ai LED e la temperatura. La misura della frequenza viene invece realizzata attraverso gli ingressi digitali e la frequenza del segnale viene poi misurata attraverso un contatore.

2.1 Hardware

2.1.1 LED installati

Sono stati installati i LED LUXEON III Star da 3W, in numero diverso a seconda del colore, così da poter ottenere da ognuno dei gruppi un flusso luminoso circa pari a 100 lm:

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

I quali sono stati installati in una piastra di alluminio opportunamente ventilata per impedire alla temperatura di raggiungere dei valori critici; infatti questo potrebbe comportare un aumento di temperatura ai LED che potrebbe risultare dannosa.

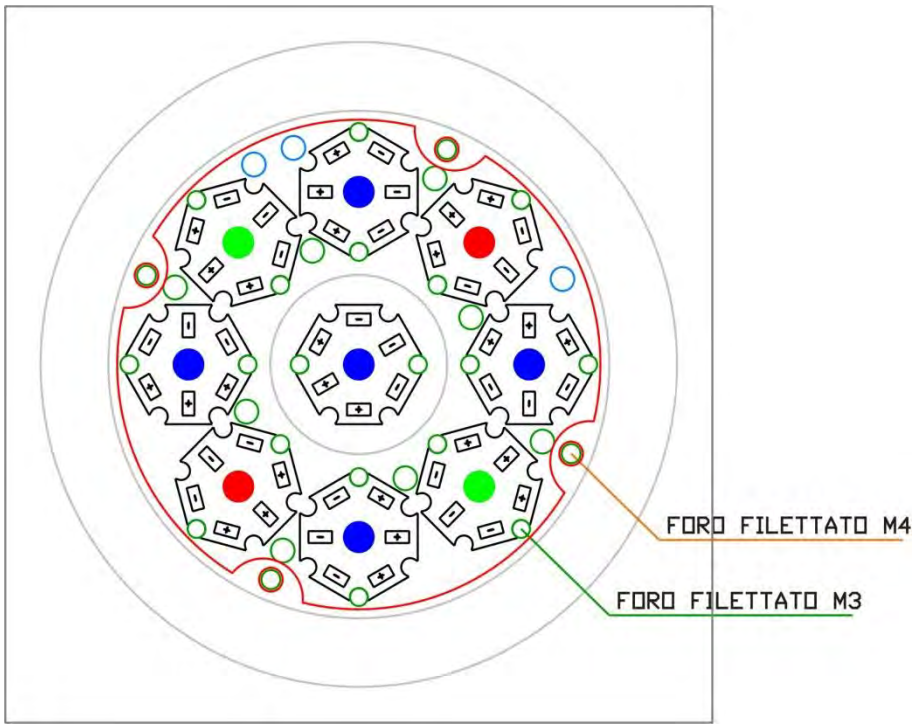


Figura 14: Schema della piastra di montaggio metallica dei LED

Inoltre davanti ai LED è stato installato un setto separatore bianco; l'installazione ha due utilità:

- Impedisce che la radiazione diretta dei LED possa influenzare la lettura da parte dei sensori TAOS.
- Impedisce che l'utente sia colpito dalla radiazione diretta dei LED, guardando attraverso il foro d'osservazione della sfera.

2.1.2 Sfera Integratrice

Per poter ottenere una radiazione che sia il più possibile uniforme, è stato ricreato il principio della sfera integratrice attraverso la creazione di una sfera, dipinta all'interno con uno smalto bianco (con coefficiente di riflessione pressoché unitario) sulla quale viene installato da una parte il sistema PIASTRA-LED precedentemente citato e dall'altra sono stati operati due fori, uno per poter installare i sensori TAOS TCS 230 e l'altro per permettere all'utente di vedere il risultato della sintesi additiva creata.

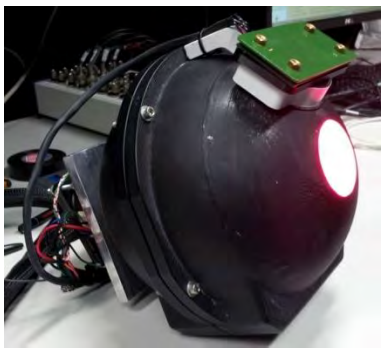


Figura 15: Sfera integratrice creata per generare la mescolanza additiva

2.1.3 Circuito d'alimentazione

Il circuito d'alimentazione dei LED, ha il compito di garantire una regolazione della luminanza in uscita dalla sfera integratrice. Questa grandezza è legata anche alla corrente d'alimentazione del LED, sulla quale si agisce.
L'alimentazione viene realizzata attraverso i driver LUXDRIVE 3021 D-E-700 BuckPuck che consentono di alimentare i LED ad una corrente impressa costante, il cui valore dipende dalla tensione che forniamo al morsetto di CTRL (control) del driver stesso. Il riferimento di questa tensione, è lo stesso di quello utilizzato per alimentare i LED ed è il collegamento al GND (Ground) della scheda Arduino.

Tali driver presentano le seguenti caratteristiche nominali:

- Alimentazione DC da 6 a 32 V;
- Uscita a corrente costante a 700 mA massimi;
- Corrente di uscita variabile dallo 0 al 100% tramite ingresso in tensione da 0 a 5 V;

È stata filtrata inoltre la tensione di alimentazione dei driver attraverso l'impiego di condensatori:

- Un condensatore elettrolitico da 1000 µF – 30 V per filtrare la tensione fornita dagli alimentatori switching;
- Un condensatore elettrolitico da 220 µF – 30 V utilizzato per filtrare ulteriormente la tensione di alimentazione ed eliminare le fluttuazioni della corrente, posto in parallelo all'alimentazione di ogni driver;
- Un condensatore ceramico da 100 nF, uno da 22 nF e un terzo da 103 nF posto in parallelo all'alimentazione di ogni driver;

Per permettere la misura delle correnti inviate ai LED attraverso gli ingressi analogici di Arduino (vd. paragrafo 2.1.6.2 Misura della corrente ai LED) e consentirne quindi il controllo in retroazione, è stata inoltre inserita una resistenza di shunt di 1Ω (parallelepipedi bianchi posti tra i driver e i morsetti verdi di alimentazione).

Figura 16: Scheda di alimentazione dei LED

Come già accennato in precedenza il driver fornisce in uscita una corrente proporzionale alla tensione che viene fornita al morsetto CTRL dello stesso, secondo la caratteristica riportata nei datasheet (Figura 17).

Attraverso il firmware implementato su Arduino (vd. paragrafo 2.2.1.6 Controllo dei driver d'alimentazione dei LED), è stata imposta al driver una condizione di lavoro nel tratto lineare della caratteristica; dunque per ottenere la corrente massima 0,7 A verrà applicata una tensione al morsetto CTRL pari a 1,65 V, mentre per corrente nulla una tensione pari a 4,2 V.

Possiamo quindi esprimere la funzione che lega la corrente in uscita dai driver con la tensione che deve essere fornita al morsetto CTRL degli stessi per ottenere quella determinata corrente:

$$Vctrl = -3{,}6 * Iout + 4{,}2$$

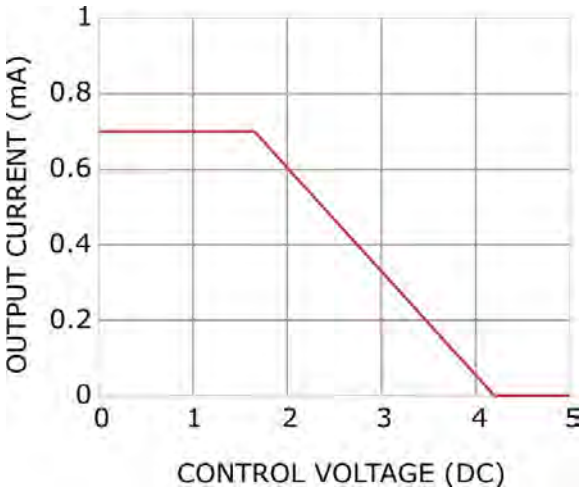


Figura 17: Caratteristica di funzionamento dei driver LUXDRIVE 3021 D-E-700 BuckPuck

La tensione di ctrl viene fornita dalla uscita PWM, opportunamente filtrata, che la scheda Arduino mette a disposizione (vd. paragrafo 2.1.6.1 Alimentazione dei pin CTRL dei driver). Il driver che comanda i LED rossi e verdi sono alimentati a 12 V, quelli blu vengono alimentati a 24 V; tale differenziazione è dovuta al fatto che la tensione di alimentazione del driver deve essere superiore rispetto a quella d'uscita di almeno 2 V (con funzionamento a tensione continua).

2.1.4 Sensore TAOS TCS230

Il sensore TAOS TCS230 consente di convertire, attraverso un sistema di 64 fotodiodi e un convertitore corrente frequenza, una luce colorata in un segnale PWM. Tale segnale presenta una frequenza direttamente proporzionale all'intensità della radiazione luminosa che lo colpisce, con un duty cycle al 50%.
I 64 fotodiodi citati sono disposti in una matrice 8x8 e sono suddivisi in base al filtro che viene loro applicato; infatti di questi 64 fotodiodi, 16 hanno interposto un filtro rosso, 16 un filtro verde, 16 un filtro blu e 16 un filtro neutro. Tutti questi fotodiodi sono posizionati in ordine sparso così da garantire una misurazione il più uniforme possibile su tutta la superficie del sensore da parte di ogni gruppo di fotodiodi.

Il sensore presenta inoltre dei pin di ingresso:

- OE (Output Enable): consente di abilitare l'uscita dei sensori TAOS.

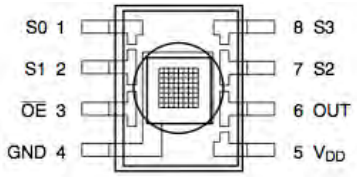
- S0 e S1, S2 E S3: sono degli ingressi la cui combinazione di livelli consente di regolare rispettivamente il fondoscala della frequenza d'uscita e il gruppo di fotodiodi selezionati.

Sono stati inoltre aggiunti sia un filtro ciano davanti al sensore, per ridurre la sensibilità all'infrarosso, che un filtro neutro a 1,5 dB per ridurre la sensibilità dei fotodiodi a tutte le lunghezze d'onda.

Inizialmente il sistema fotodiodi-LED era stato pensato per garantire che, utilizzando le informazioni provenienti dal gruppo di fotodiodi relativo ad un certo colore, si potesse regolare la corrente ai LED di quel specifico colore ed ottenere così tre anelli di controllo diversi per il colore rosso, il verde e il blu.

La cosa però non è stata facile da implementare per diverse ragioni, ma sicuramente tra le più importanti c'è quella per cui gli spettri di emissione dei LED non sono perfettamente comparabili con gli spettri di sensibilità dei filtri dei corrispondenti fotodiodi. Ciò significa che non ho una corrispondenza univoca tra un gruppo di LED di un certo colore ed i fotodiodi con il filtro relativo; quindi se accendo un gruppo di LED di un certo colore facendo circolare un valore di corrente, ottengo una risposta, da tutti quei fotodiodi il cui filtro presenti una sensibilità spettrale alle lunghezze d'onda di emissione dei LED, che sarà proporzionale all'entità della sensibilità spettrale che i filtri stessi hanno per lo spettro di emissione dei LED che sono stati attivati.
Inoltre le sensibilità spettrali dei diversi filtri, spesso si sormontano tra loro e questo comporta l'analogo effetto descritto in precedenza.

Come vedremo sarà compito di questo lavoro cercare di trovare in prima battuta una matrice di trasformazione frequenza- corrente che ci consenta di abbinare, all'accensione di un gruppo di LED di un certo colore, una terna di frequenze rilevate dai sensori TAOS.



S0	S1	Fondo scala frequenza		S2	S3	Fotodiodi selezionati
L	L	Spento		L	L	Rossi
L	H	500 kHz		L	H	Blu
H	L	100 kHz		H	L	Nessun filtro
H	H	10 kHz		H	H	Verde

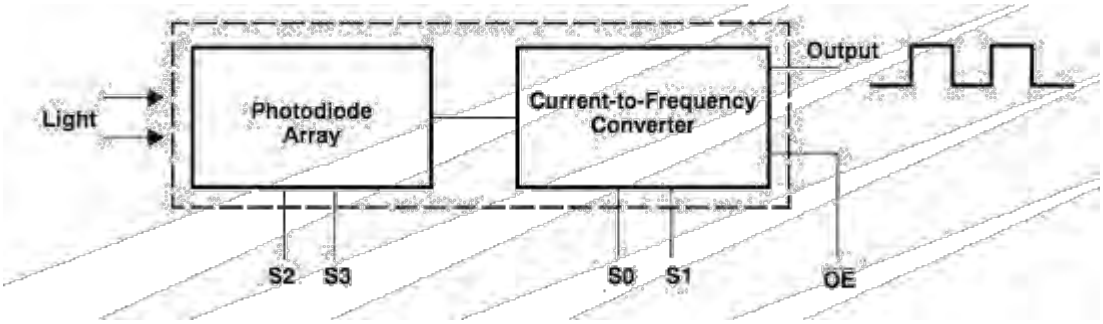


Figura 18: Piedinatura e comandi del sensore TAOS TCS230.

2.1.5 Sensore di temperatura AD590

Nella piastra metallica è stato inoltre installato il sensore di temperatura AD590, il quale consente di produrre una corrente proporzionale alla temperatura rilevata (1 μA/K). Il sensore di temperatura è composto da un morsetto di alimentazione (collegato nel nostro caso ai 5V di alimentazione di Arduino) e da un morsetto sul quale viene resa disponibile il segnale proporzionale alla temperatura. Tale segnale verrà poi portato alla resistenza di shunt che ci consentirà di effettuare la misura (vd. paragrafo 2.1.6.3 Misura della temperatura alla piastra).

Figura 19: Sensore di temperatura AD590

2.1.6 Scheda Arduino e interfaccia con la sorgente luminosa

Arduino è una scheda elettronica dotata di microcontrollore e di altri circuiti accessori che consente:

- di creare dei segnali PWM oppure dei segnali digitali;
- di leggere dei segnali dall'esterno, che verranno poi opportunamente convertiti in segnali digitali gestibili da Arduino stesso, attraverso un convertitore ADC.

Nell'oggetto in questione è stato utilizzato Arduino AT Mega 2560, a 10 bit, che contiene al suo interno il microcontrollore collegato alle porte di ingresso ed uscita, un collegamento al PC attraverso porta USB ed un regolatore di tensione.

Figura 20: Scheda Arduino Mega 2560

La scheda Arduino consente, nel caso in esame, di:

- produrre da una parte i segnali PWM che, una volta filtrati delle componenti a frequenza elevata (vd. paragrafo 2.1.6.1 Alimentazione dei pin CTRL dei driver), servono a comandare i driver di alimentazione dei LED.
- leggere le tensioni alle resistenze di shunt, la cui scelta verrà successivamente discussa, che consentono di misurare le correnti inviate ai LED e di poter inoltre valutare la corrente prodotta dal sensore di temperatura, che sarà proporzionale alla temperatura della piastra di montaggio.
- valutare le frequenze dei segnali in uscita dal sensore TAOS TCS230 attraverso gli ingressi digitali di Arduino (vd. paragrafo 2.1.6.5 Misura della frequenza del segnale in uscita dal sensore TAOS TCS230). Sono stati utilizzati inoltre anche dei canali di uscita digitali che ci consentono di comandare il funzonamento del sensore TAOS TCS230, scegliendone il fondoscala, attivandone l'uscita e scegliendo di volta in volta i fotodiodi da attivare nel momento in cui viene effettuata la misura.

La piattaforma Arduino può essere alimentata sia attraverso il connettore USB ma anche attravero un alimentazione esterna, collegandola al morsetto Vin, con una tensione variabile da 7 a 12 V. Per ciò che riguarda l'alimentazione, i Pin di nostro interesse sono:

- GND: il pin collegato a massa
- 5 V: pin con tensione a 5V fornita dal regolatore di tensione.
- 3,3 V: pin con tensione a 3,3 V fornita dal regolatore di tensione, con corrente fino a 50 mA.
- Vin: pin in cui è possibile collegare l'alimentazione esterna e da cui è possibile misurare anche tale valore di tensione.

Arduino contiene una Memoria Flash da 256 KB su cui è possibile salvare il file sorgente che comanda Arduino, 4KB di memoria EEPROM (richiamabili utilizzando la libreria EEPROM) e 8 KB di SRAM.

Alcuni dei pin di Arduino hanno delle funzioni particolari, elenchiamone alcuni:

- AREF: rappresenta la tensione di riferimento per il convertitore analogico-digitale; tale tensione può essere imposta fisicamente attraverso tale morsetto oppure è possibile impostarla attraverso l'IDE di Arduino tramite il comando analogReference().
- LED: pin che comanda un LED posto sulla scheda. Questo LED viene utilizzato nel nostro caso effettuando un lampeggio ad ogni ciclo così da poter avere subito un riscontro se il file sorgente stia effettivamente facendo il suo compito.

La comunicazione con il PC avviene attraverso una porta seriale, che verrà connessa alla USB del PC e attraverso la quale sarà possibile sia inserire all'interno della memoria Flash il file sorgente creato attraverso l'IDE di Arduino, che comunicare con Arduino stesso per fornire dei comandi oppure ricevere dei dati; Arduino inizialmente gestisce queste informazioni all'interno del buffer, questo poi verrà svuotato da Arduino effettuando le operazioni richieste, oppure dal PC acquisendo i dati resi disponibili da Arduino.

Per poter gestire la sorgente luminosa in esame è stata costruita una scheda elettronica di interfaccia; i componenti fondamentali di tale scheda sono:

- Filtro RC passa basso per eliminare dalla tensione PWM tutte le componenti in alta frequenza e trasformando così il segnale stesso in un segnale continuo variabile da 0 a 4,2 V.

- Filtro RC passa basso per eliminare la componente di rumore in alta frequenza dalle tensioni misurate alle resistenze di shunt.
- Resistenza di shunt da 2,2 kΩ: questa consente di trasformare la corrente prodotta dal sensore di temperatura in una tensione, misurata attraverso l'ingresso analogico.
- Connessione ai sensori TAOS TCS 230.
- Cablaggi per la misurazione della tensione al morsetto ctrl dei driver.

Questa scheda sarà poi opportunamente collegata alla scheda d'alimentazione vista in precedenza per consentire di accedere a tutte le grandezze che dovranno essere misurate attraverso gli ingressi analogici. I cablaggi realizzati consentono inoltre ma di fornire i segnali PWM di Arduino filtrati, ai morsetti CTRL dei driver di alimentazione dei LED. Nei seguenti paragrafi verranno descritte le componenti della scheda di controllo, che fungono da interfaccia tra Arduino e la sorgente.

2.1.6.1 Alimentazione dei pin Ctrl dei Driver

Per l'alimentazione del pin CTRL dei driver abbiamo bisogno di una tensione che sia il più possibile continua; tale tensione avrà valore compreso tra un massimo di 4,2 V (corrispondente a corrente di valore prossimo allo zero da parte dei driver) e un minimo di 1,65 V (corrispondente ad una corrente nulla inviata ai LED).
Il filtro RC passa basso del primo ordine, posto a valle delle uscite PWM di Arduino, avrà il compito di ridurre la frequenza di taglio, così da avere un segnale che sia il più continuo possibile.
Il filtro è composto da una resistenza di valore pari a 270 Ω e un condensatore al tantallio da 100μF, che ci permette di ottenere una frequenza di taglio di:

$$ft = \frac{1}{2 * \pi * R * C} = \frac{1}{2 * \pi * 270 * 100 * 10^{-6}} \sim 6\ Hz$$

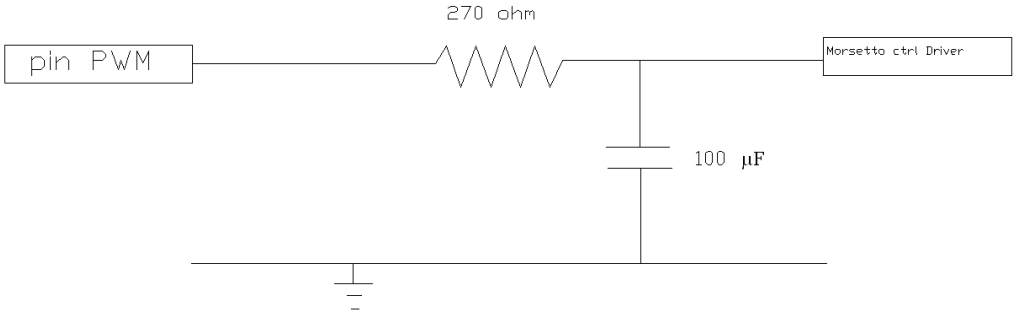


Figura 21: Schema filtro d'uscita PWM.

2.1.6.2 Misura della corrente ai LED

Per misurare la corrente inviata ai LED, è stata installata una resistenza di shunt da 1 Ω che ci consentirà di convertire la corrente in una tensione che, opportunamente filtrata, verrà inviata agli ingressi analogici.

Il filtro presenta una resistenza da 47 kΩ e un condensatore al tantallio da 4,7 µF, che permette di ottenere una pulsazione di taglio pari a :

$$ft = \frac{1}{2 * \pi * R * C} = \frac{1}{2 * \pi * 47 * 10^{3} * 4{,}7 * 10^{-6}} \sim 10\ Hz$$

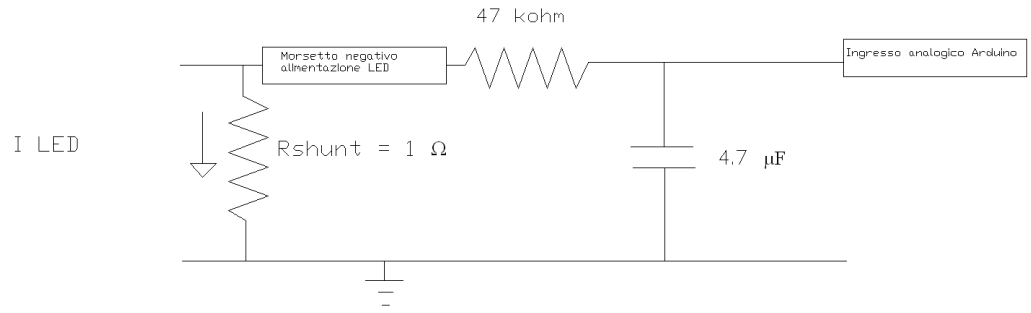


Figura 22: Filtro RC all'ingresso analogico per la misura di corrente

2.1.6.3 Misura della temperatura alla piastra

La misura della temperatura, come precedentemente citato, avviene attraverso un sensore di temperatura (AD590) posto sulla piastra metallica d'installazione dei LED.

Tale sensore, secondo i dati ricavabili dal datasheet, è in grado di produrre una corrente continua proporzionale alla temperatura che questo rileva (1µA/K); questo vuol dire che, nel range di temperature in cui opera la sorgente (intorno ai 300 K), tale sensore produrrà una corrente di valore circa pari a 300 µA.

È stata installata una resistenza da 2,2 kΩ che consente di ottenere ai suoi capi, nell'intorno della temperatura ambiente, dei valori di tensione di 0,7 V.

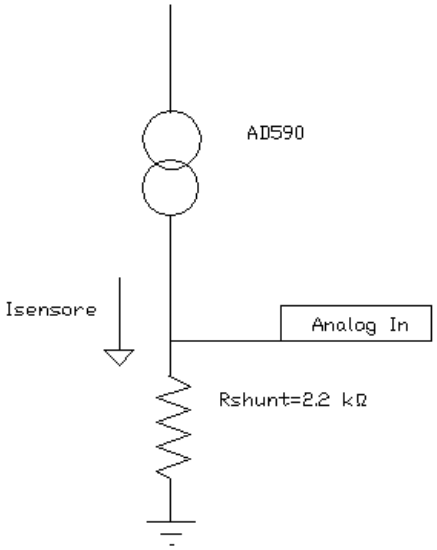


Figura 23: Schema elettrico per la misura di temperatura

2.1.6.4 Alimentazione dei pin di controllo del sensore TAOS TCS230

Come descritto nel paragrafo "2.1.4: sensore TAOS TCS230" per poter abilitare l'uscita del sensore ed attivare un certo gruppo di fotodiodi è necessario inviare un segnale agli ingressi del sensore TAOS TCS230.
Questi segnali vengono prodotti, in base alla misura che il sensore deve operare, attraverso le uscite digitali che Arduino mette a disposizione (vd. appendice "Pin Arduino utilizzati"). La composizione dei diversi segnali consente di regolare il fondoscala della misura ed anche il gruppo di fotodiodi attivati.

2.1.6.5 Misura della frequenza del segnale in uscita dal sensore TAOS TCS230

La misura della frequenza del segnale d'uscita proveniente sensore TAOS TCS230, avviene collegando l'uscita dello stesso ad un pin digitale, programmato per funzionare come un ingresso digitale (vd. appendice "Pin Arduino utilizzati"). Sul segnale che viene misurato, avviene una misura di frequenza attraverso un contatore

2.1.6.6 Misura della tensione ctrl dei driver d'alimentazione dei LED

La tensione rilevata al morsetto CTRL dei driver di alimentazione dei LED sono state connesse a degli ingressi analogici (vd. appendice "Pin Arduino utilizzati"), anche se in realtà non è implementata alcuna function che consenta la misura del reale valore di questa tensione

2.2 Software

La scrittura del firmware di Arduino viene realizzata attraverso l'apposito software, fornito in dotazione, che utilizza un linguaggio di scrittura denominato Wiring, molto simile al linguaggio di programmazione C.
L'IDE di Arduino consente, oltre a programmare la scheda con i comandi opportuni, anche di implementare le librerie che ci consentiranno di gestire Arduino stesso (ad esempio quella della PWM oppure della lettura della frequenza dei segnali in uscita dai sensori TAOS TCS230).

L'acquisizione dei segnali analogici dall'esterno avviene attraverso il comando analogRead, mentre la creazione di segnali di tipo PWM avviene attraverso il comando analogWrite. Per ciò che riguarda i segnali digitali c'è la possibilità sia di poterli generare ma anche di poterli acquisire (digitalWrite o digitalRead).

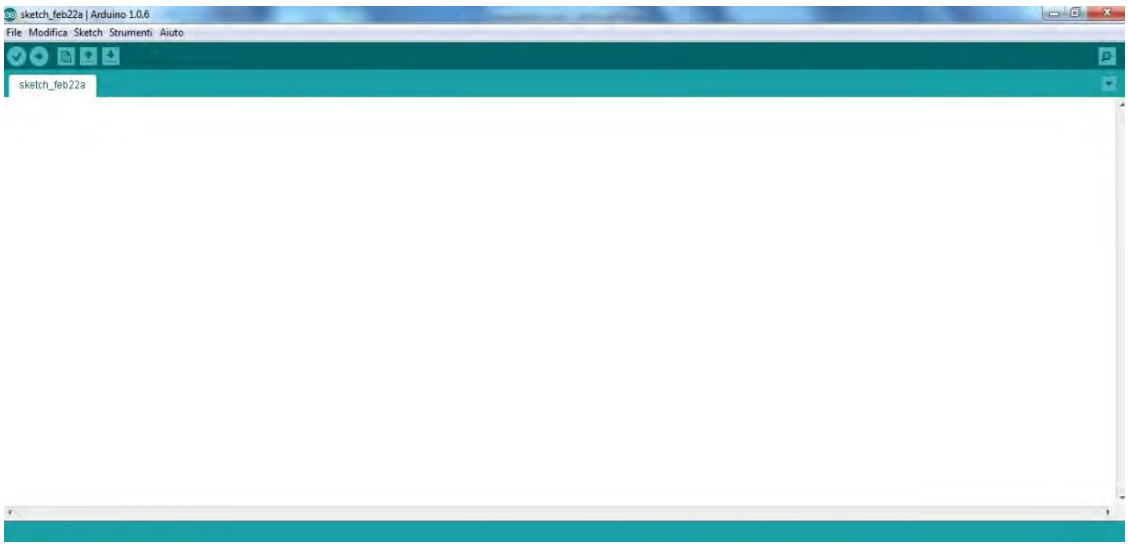


Figura 24: foglio di scrittura del FW di Arduino.

2.2.1 Cenni sul funzionamento del Firmware Arduino preesistente

Per prima cosa sono state richiamate le librerie per la creazione del segnale PWM (TimerThree.h) e per la lettura della frequenza (FreqCount.h).

Vengono inoltre dichiarate il nome e il tipo delle variabili in gioco all'interno del Firmware, che vengono inoltre inizializzate.
Il firmware Arduino preesistente presenta successivamente un ciclo, il quale consente di valutare continuamente se nel buffer di Arduino sono presenti dei dati. In particolare viene valutata la presenza dei comandi, inviabili attraverso terminale (Teraterm), relativi al imposizione di uno specifico setpoint di corrente ai LED oppure alla misura di alcune delle grandezze in gioco.
I comandi che possono essere implementati sono dunque fondamentalmente di due tipologie:

- comandi di impostazione del setpoint di corrente: tali comandi iniziano tutti con la lettera 's' e comprendono i comandi spr, spg, spb per impostare rispettivamente i setpoint di corrente ai LED rossi verdi e blu;
- comandi di misura: questi comandi iniziano tutti con la lettera 'm' e comprendono:

    1. Misura delle correnti inviate ai LED: i comandi in questione consentono di misurare la corrente inviata ai LED di un solo colore (mcr, mcg, mcb per misurare rispettivamente la corrente ai LED rossi, verdi e blu) oppure su tutti i LED (attraverso il comando mct).
    2. Misura delle tensioni al morsetto CTRL dei driver: in questo caso non operiamo una misura vera e proporia ma viene semplicemente stampato a video la tensione di ctrl che viene calcolata nel controllo delle correnti. C'è la possibilità comunque di stamapare a video sia la tensione del morsetto di CTRL ad un driver che comanda un solo gruppo di LED di un certo colore (mtr, mtg, mtb per misurare la tensione CTRL del driver che comanda i LED rossi, verdi e blu) oppure tutte e tre queste tensioni assieme (con il comando mtt);
    3. Misura delle frequenze prodotte dai sensori TAOS TCS230: anche in questo caso la misura può riguardare un solo gruppo di fotodiodi (mfr, mfg, mfb per misurare la frequenza prodotta dai fotodiodi con installati rispettivamente i filtri rossi, verdi e blu) oppure tutti (attraverso il comando mft).
    4. Misura della temperatura alla piastra metallica i montaggio dei LED: attraverso il comando 'mte'.

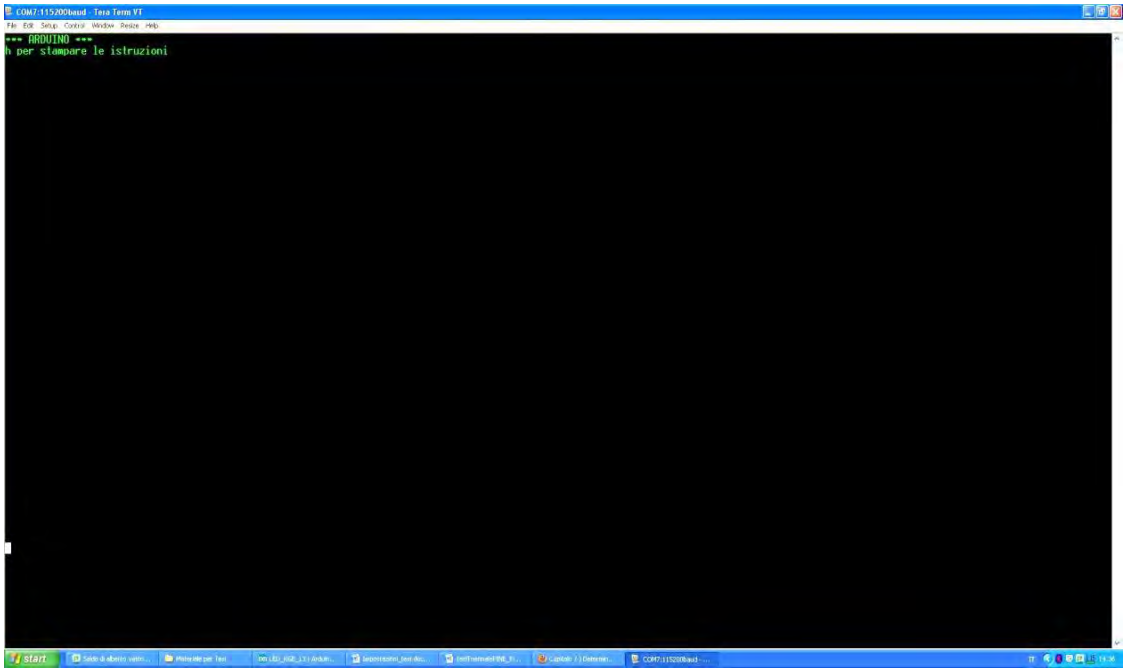


Figura 25: Screenshot della schermata di Teraterm

In caso sia di presenza o meno di un comando a video, viene comunque operata ad ogni ciclo l'inversione dello stato del LED installato sulla scheda Arduino, attraverso la funzione LampeggioLed(). In questo modo è possibile capire fin da subito se effettivamente il programma stia operando o meno; inoltre viene fatto sempre un controllo, di tipo PID, sulle corrente inviate ai LED (attraverso la function ControlloPID()) . Vediamo nel dettaglio alcune delle parti fondamentali del firmware.

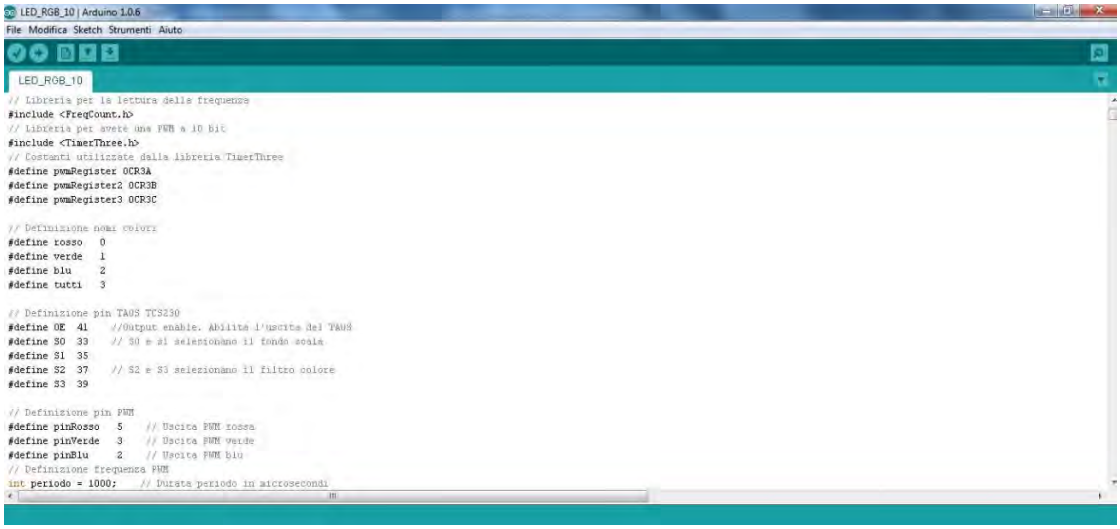


Figura 26: L'ambiente di progettazione del Firmware Arduino

2.2.1.1 Setup iniziale

Attraverso questa funzione viene impostato il funzionamento di Arduino. In questa function viene:

- Impostato il LED installato su Arduino come uscita.
- La velocità di trasmissione della porta seriale (baudrate).
- La tensione di riferimento per il convertitore ADC (INTERNAL1V1, comando che consente di avere tensione di riferimento dell'ADC pari a 1,1 V).
- Viene impostato il timer del microcontrollore: viene cioè impostato il periodo della PWM (Timer3.initialize(periodo)).
- Vengono impostati i pin di uscita digitali che consentono di controllare il sensore TAOS TCS230; in particolare viene definito, attraverso i morsetti S0 e S1 del sensore, il fondoscala del sensore stesso (impostato a 10 kHz) e attraverso il morsetto OE viene abilitata l'uscita del sensore.
- Viene inoltre impostato il tempo di gate del frequenzimetro, cioè il tempo entro cui viene calcolata la media della misura della frequenza. Questo comando consente di definire la precisione della misura stessa. Impostando come in questo caso un GateTime=1000 [ms] significa avere una risoluzione nella misura di:

$$\frac{1}{1000 * 10^{-3}} = 1\ Hz$$

```
void SetupIniziale()
{
  pinMode(ledPin, OUTPUT);            // Imposta il pin relativo al led presente su Arduino come pin di output. Con l'oscilloscopio è possibile controllare la durata del loop()
  Serial.begin(baudRate);             // Inizializza la porta seriale impostando la velocità di connessione.
  analogReference(INTERNAL1V1);     // Imposta la tensione di riferimento dell'ADC a 1.1 V
  Timer3.initialize(periodo);         // Inizializza il Timer 3 del microprocessore e imposta il periodo della PWM.

  // pinMode(OUT, INPUT);  Non può essere cambiato. E' impostato direttamente dalla libreria FreqCount.
  pinMode(OE, OUTPUT);     // Inizializza il sensore TAOS TCS 230 configurando i pin di Arduino come uscite.
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
```

```
  digitalWrite(OE, LOW);  // OE = output enable. Abilita l'uscita del TAOS TCS 230
 delay(10);

 // Fondo scala della frequenza del TAOS TCS 230
 //(s0=s1=LOW Uscita frequenza spenta)
 //digitalWrite(S0, LOW);
 //digitalWrite(S1, LOW);
 //(s0=LOW s1=HIGH fondo scala frequenza 10kHz)
 digitalWrite(S0, LOW);
 digitalWrite(S1, HIGH);
 //(s0=HIGH s1=LOW fondo scala frequenza 100kHz)
 //digitalWrite(S0, HIGH);
 //digitalWrite(S1, LOW);
 //(s0=HIGH s1=HIGH fondo scala frequenza 500kHz)
 //digitalWrite(S0, HIGH);
 //digitalWrite(S1, HIGH);

 FreqCount.begin(Gate);    // Inizializza il frequenzimetro e imposta il gatetime
 InizioProgramma();        // Definisce la fine del setup iniziale e l'inizio del programma
}
```

Figura 27: Setup iniziale di Arduino

Vengono stampate a video le istruzioni che è possibile digitare attraverso TeraTerm per impostare i setpoint di corrente ai LED oppure per avere dei valori frutto della misura delle grandezze precedentemente citate. Le istruzioni sono:

- strxxxx: imposta la corrente sul canale rosso (con xxxx un valore da 0 a 1000);
- stgxxxx: imposta la corrente sul canale verde (con xxxx un valore da 0 a 1000);
- stbxxxx: imposta la corrente sul canale blu (con xxxx un valore da 0 a 1000);
- mcr: misura la corrente sul canale rosso;
- mcg: misura la corrente sul canale verde;
- mcb: misura la corrente sul canale blu;
- mtr: misura la tensione sul canale rosso;
- mtg: misura la tensione sul canale verde;
- mtb: misura la tensione sul canale blu;
- mfr: misura la frequenza del canale rosso;
- mfg: misura la frequenza del canale verde;
- mfb: misura la frequenza del canale blu;
- mte: misura la temperatura.

2.2.1.2 Gestione delle misure di corrente

Le tensioni misurate alle resistenze di shunt da 1 Ω (proporzionali alle correnti di alimentazione dei LED), dopo essere state opportunamente filtrate, vengono connesse ai rispettivi ingressi analogici.
La media di 100 misure di tensione all'ingresso analogico, viene convertita, attraverso l'ADC a 10 bit presente all'interno di Arduino, in un numero da 0 a 1023 in base al valore di tensione presente all'ingresso analogico. La conversione avviene sapendo che al valore di tensione di riferimento (impostato in precedenza a 1,1 V) scelto per l'ADC, corrisponde la massima risoluzione dell'ADC stesso (10 bit: 2^{10}-1=1023). L'operazione di conversione sarà quindi una semplice proporzione; infatti se alla tensione di riferimento corrisponde un valore di 1023, ad una tensione generica d'ingresso Valim, corrisponde un valore:

$$\frac{1023}{1{,}1} * Valim$$

questo sarà il valore che assume la tensione allo shunt convertita in base alla risoluzione dell'ADC di Arduino e che, divisa per la corrispondente resistenza di shunt, fornisce la corrente circolante.

Questa operazione viene effettuata attraverso la function MisuraCorrente(), che in base al colore scelto, opera questa valutazione all'ingresso analogico corrispondente.

```
void MisuraCorrente(int colore)     // Funzione che misura la corrente erogata dai driver. Bisogna passargli il colore del quale si desidera avere la corrente
{
 // Variabili locali
 int letture = 100;                 // Numero di letture di corrente da effettuare
 float risoluzione = 0.0009775171;  // Reciproco della risoluzione dell'ADC. Viene dato il reciproco per evitare di fare divisioni. Impegnative per il microcontrollore.
 int pin = 0;                       // Memorizza il pin di input dell'ADC da leggere.

 correnteMisurata[colore] = 0;    // Inizializza la corrente misurata

 if (colore == rosso)    // Istruzioni condizionali che inpostano il pin dell'ADC da leggere
 {
```

```
  pin = correnteRosso;
 }
 if (colore == verde)
 {
  pin = correnteVerde;
 }
 if (colore == blu)
 {
  pin = correnteBlu;
 }

 for (int i = 0; i < letture; i++)    // Ciclo for che interroga il pin dell'ADC selezionato
 {
  correnteMisurata[colore] += analogRead(pin);
 }

 correnteMisurata[colore] /= letture;    // Viene memorizzata la tensione media
 correnteMisurata[colore] *= risoluzione * VREF;      // Viene memorizzata la tensione convertita in Volt
 correnteMisurata[colore] /= resistenza[colore];     // Viene calcolata la corrente. Il valore della resistenza è definito all'inzio del programma.
}
```

Figura 28: function per la misura delle correnti ai LED e ai resistori

2.2.1.3 Gestione delle misure di frequenza

Per la lettura della frequenza, viene prima di tutto richiamata all'inizio del firmware un'apposita libreria (FreqCount.h), che consente di valutare la frequenza del segnale che viene misurato ad un specifico ingresso digitale.

La misura della frequenza viene realizzata attraverso la funzione LeggiFrequenza(). Questa funzione prima di tutto imposta, in base ai fotodiodi che voglio utilizzare per la misura, la combinazione dei livelli delle uscite digitali S2 e S3 corrispondenti.

Successivamente viene inizializzato il frequenzimetro; la risoluzione con cui viene effettutata la lettura della frequenza è legata all'impostazione del tempo di gate (tempo di gate=1000 ms=1 s risoluzione di 1Hz).

Fintanto che la misura non è stata effettuata e sono disponibili dati nel buffer, avviene la lettura e il dato rilevato sarà salvato in un vettore; la posizione di allocamento, dipende essenzialmente da quali fotodiodi sono stati impiegati:

- attivazione dei fotodiodi rossi: dato allocato nella prima colonna.
- attivazione dei fotodiodi verdi: dato allocato nella seconda colonna.
- attivazione dei fotodiodi blu: dato allocato nella terza colonna.

```
void LeggiFrequenza(int colore)    // Funzione che legge la frequenza del colore selezionato
{
 // Variabili locali
 int endfreq = 0;

 if (colore == rosso)    // Imposta il filtro per misurare solo il rosso
 {
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  FreqCount.end();
 }
 if (colore == verde)    // Imposta il filtro per misurare solo il verde
 {
  digitalWrite(S2, HIGH);
  digitalWrite(S3, HIGH);
  FreqCount.end();
 }
 if (colore == blu)    // Imposta il filtro per misurare solo il blu
 {
  digitalWrite(S2, LOW);
  digitalWrite(S3, HIGH);
  FreqCount.end();
 }
 if (colore == tutti)    // Imposta il filtro per misurare tutti i colori. NON UTILIZZATO
 {
  digitalWrite(S2, HIGH);
```

```
  digitalWrite(S3, LOW);
  FreqCount.end();
 }

 FreqCount.begin(Gate);    // Inizializza il frequenzimetro
 delay(5);
 while(endfreq == 0)    // Quando riceve i dati legge la frequenza.
 {
  if (FreqCount.available())
  {
   freq[colore] = FreqCount.read();
   endfreq = 1;
  }
 }
}
```

Figura 29: Function per la misura della frequenza in uscita dal sensore TAOS TCS230

2.2.1.4 Gestione delle misure di temperatura

All'ingresso analogico di Arduino viene portato in questo caso la tensione che viene rilevata alla resitenza di shunt da 2,2 kΩ. Attraverso la funzione MisuraTemperatura(), tale tensione viene anch'essa convertita dall'ADC in base alla modalità vista per la misura di corrente. La resistenza di shunt per la misura della temperatura è stata impostata nel firmware a 2,2 Ω, questo vuol dire che la corrente calcolata ha un valore mille volte più grande del reale (quindi di fatto misurato in mA). Dunque il fattore di conversione da K a °C (273,15), dovrà essere diviso per 1000.
Quando viene richiesta la misura di temperatura, bisogna moltiplicare il dato ottenuto per 1000; in questo modo si ottiene di fatto un valore di corrente espresso in μA che, essendo il fattore di conversione del sensore di temperatura 1 μA/K, è direttamente proporzionale alla temperatura. L'operazione di moltiplicazione per mille viene fatta attraverso la funzione StampaFloat().

void MisuraTemperatura*() // Funzione che misura la temperatura fornita dal sensore AD590 montato sulla base di alluminio della sorgente.*

```
{
  // Variabili locali
  float risoluzione = 0.0009775171;    // Risoluzione dell'ADC

  temperatura = analogRead(pintemperatura);   // Viene memorizzata la lettura del pin dell'ADC
  temperatura *= risoluzione * VREF;         // Viene convertita la misura in Volt
  temperatura /= resistenzaTemp;             // Viene calcolata la corrente erogata dall'AD590. La corrente corrisponde alla temperatura (Come specificato nei DataSheet)
  temperatura -= (273.15/1000.);             // Opero una conversione da K a °C

}


void StampaFloat(float dato)    // Funzione che serve per la stampa a video dei risultati
{
  // Variabili locali
  float datoMoltiplicato = 0;

  datoMoltiplicato = dato * 1000.;     // moltiplico per 1000 il dato che viene passato alla funzione e lo stampa. La funzione non modifica le variabili globali
  Serial.print(datoMoltiplicato);
}
```

Figura 30: Function per la misura della temperatura

2.2.1.5 Gestione della misura della tensione ctrl del driver

La tensione di ctrl fornita al driver viene collegata ai rispettivi ingressi analogici. In realtà però non viene operata una msiura reale, ma semplicemente una stampa a video della variabile che rappresenta la tensione di ctrl che viene calcolata di volta in volta nell'ambito del controllo delle correnti ai LED. La stampa avviene facendo una richiesta a video attraverso i comandi:

- mtr: stampa a video il valore della tensione di ctrl al driver che alimenta i LED rossi.
- mtg: stampa a video il valore della tensione di ctrl al driver che alimenta i LED verdi.
- mtb: stampa a video il valore della tensione di ctrl al driver che alimenta i LED blu.
- mtt: stampa a video il valore della tensione di ctrl di tutti i driver.

2.2.1.6 Controllo dei driver di alimentazione dei LED

Il controllo dei driver LUXDRIVE 3021 D-E-700 BuckPuck viene fatta in base al setpoint di corrente che vogliamo impostare ai LED; tale valore si mantiene pari al valore inizializzato (zero) se da terminale non impostiamo alcun valore oppure avrà un valore compreso in un range tra 0-0,7 A.

Il controllo viene fatto ciclicamente attraverso il ciclo (void loop()) del Firmware di Arduino, nel quale viene richiamata la funzione ControlloPID(), per ognuna delle tre famiglie di LED.
La funzione ControlloPID() opera una differenza tra il setpoint di corrente impostato e la corrente che viene effettivamente misurata (tensione alla resistenza di shunt che viene convertita in un segnale digitale gestibile da Arduino); questa differenza rappresenta l'errore proporzionale del ciclo in esame. Successivamente verrà calcolato l'errore integrale che sarà la somma tra l'errore proporzionale del ciclo attuale e l'errore proporzionale del ciclo precedente; infine l'errore derivato sarà pari a zero nel primo ciclo, mentre nei successivi sarà calcolato come velocità di variazione della corrente misurata tra ciclo attuale e precedente (rapporto tra la variazione della corrente e il tempo impiegato per compiere un intero ciclo).
Una volta pesati i singoli contributi di errore con le relative costanti moltiplicative del controllore PID, viene sommato il contributo totale di errore con il setpoint di corrente impostato. Questo nuovo livello di corrente consente di compensare, di volta in volta, l'errore tra setpoint impostato e corrente realmente circolante, per arrivare ad ottenere l'effettiva corrente che abbiamo scelto. In base alla corrente che si vuole ottenere, viene calcolata la tensione da imporre al morsetto ctrl del driver (attraverso la funzione descrittiva del tratto lineare tensione ctrl-corrrente, vista nel paragrafo 2.1.3 Circuito di alimentazione).
La tensione di ctrl dei driver è stata inoltre limitata al campo di valori per cui esiste una relazione diretta tra tensione ctrl e corrente inviata ai morsetti d'uscita; tale intervallo è 1,65-4,2 V. Il primo valore sarà quindi il valore di tensione al morsetto ctrl che consente di ottenere una corrente in uscita pari a 0,7 A; mentre il secondo sarà il valore per cui la corrente in uscita è pari a 0 A.

Il valore di tensione di ctrl ottenuto viene convertito in base alla risoluzione dell'ADC di Arduino (10 bit: 2^{10}-1=1023); a tale valore è associato il valore di tensione al morsetto di ctrl per il quale si ottiene la corrente nulla erogata dagli stessi (4,2 V secondo quanto detto in precedenza). La funzione CalcolaPWM() consente di ricavare il valore, pesato con la risoluzione dell'ADC, attraverso una proporzione:

$$Vctrl * \frac{1023}{4{,}2}$$

Tale valore verrà impostato al corrispondente morsetto dell'uscita PWM, attraverso la funzione ImpostaPWM().

```
void ControlloPid(int colore) // Funzione che implementa un controllore PID
{

 correnteMisurataVecchia[colore] = correnteMisurata[colore]; // Salva il valore precedente di corrente misurata.

 MisuraCorrente(colore);  // Richiama la funzione che si occupa di misurare la corrente e salva i dati nella variabile globale misuraCorrente[].
 errore[colore] = ternaRGB[colore] - correnteMisurata[colore];  // Calcola l'errore come differenza tra il set point di corrente e la corrente misurata.

 erroreIntegrale[colore] += errore[colore];  // calcola l'errore integrale come somma dell'errore.

 erroreDerivato[colore] = (correnteMisurata[colore] - correnteMisurataVecchia[colore]) / dt;  // calcola l'errore derivato

 if (primociclo == 1)  // condizione if. se è il primo ciclo imposta l'errore derivato a 0
 {
  erroreDerivato[colore] = 0;
  primociclo = 0;
 }
```

```
 correzione[colore] = errore[colore] * kp + erroreIntegrale[colore] * ki + erroreDerivato[colore] * kd;    // Calcola il valore della correzione.

 correzione[colore] += ternaRGB[colore];    // Somma alla terna di correnti il valore della correzione da apportare.

 tensioneCtrlCalcolata[colore] = mDriver * correzione[colore] + qDriver;    // Calcola la tensione da applicare al pin CTRL utilizzando la caratteristica fornita dai datasheet

 if (tensioneCtrlCalcolata[colore] < 0)    // Blocca la caratteristica. per valori inferiori a 1.65 e maggiori a 4.2 viene impostato un valore definito.
 {
  tensioneCtrlCalcolata[colore] = 1.65;
 }
 if (tensioneCtrlCalcolata[colore] > 4.2)
 {
  tensioneCtrlCalcolata[colore] = 4.2;
 }

 CalcolaPWM(colore);    // Richiama la funzione che calcola il duty cicle della PWM
 ImpostaPWM(colore);    // Richiama la funzione che configura la PWM
}

void CalcolaPWM(int colore)    // Funzione che calcola il duty cicle della PWM.
{
 float k = 243.5714286;      // Per evitare le divisioni questo valore è stato calcolato e riportato come costante. è uguale alla risoluzione dell'ADC (1023) diviso il valore massimo di tensione applicabile al pin CTRL (4.2)

 ternaPWM[colore] = tensioneCtrlCalcolata[colore] * k;    // Memorizza un numero che va da 0 a 1023

}
```

```
void ImpostaPWM(int colore)    // Configura la PWM sul colore selezionato
{
  // Variabili locali
  int pinPWM = 0;    // Memorizza il pin sul quale modificare la PWM

  if (colore == rosso)    // Istruzioni condizionali che inpostano il pin sul quale modificare la PWM. Tutte le variabili sono state definite all'inizio del programma
  {
    pinPWM = pinRosso;
  }
  if (colore == verde)
  {
    pinPWM = pinVerde;
  }
  if (colore == blu)
  {
    pinPWM = pinBlu;
  }
  Timer3.pwm(pinPWM, ternaPWM[colore]);      // Richiama la funzione contenuta nella libreria TimerThree che configura la PWM sull'uscita selezionata.
}
```

Figura 31: Controllo delle correnti ai LED

CAPITOLO 3: Analisi preliminare

Il primo obiettivo era quello di migliorare l'interfaccia grafica del sistema che, in partenza, si basava sul programma TeraTerm. Attraverso questo terminale è possibile implementare dei comandi per accendere una delle tre famiglie di LED, scegliendone l'intensità luminosa (da 0 a 1000, che corrisponde ad un range di corrente ai rispettivi LED di 0-0,7 A).
L'idea è quella di modificare l'interfaccia così da poter immettere dalla Command Window di Matlab le frequenze che vogliamo ottenere in uscita dai sensori TAOS TCS230, verificandone poi il valore realmente rilevato.

3.1 Acquisizione dati ed invio comandi

Per consentire la comunicazione tra Arduino e Matlab attraverso la porta seriale, sono stati sfruttati degli appositi script Matlab, già esistenti, di apertura e chiusura della seriale ma anche degli script di scrittura e lettura dei dati.

Per l'apertura e la chiusura della seriale sono stati utilizzati gli script esistenti in tutte le simulazioni fatte; mentre gli script realizzati per la scrittura e la lettura hanno subito varie modifiche frutto di ragionamenti fatti sulla comunicazione del PC con Arduino e il suo funzionamento in termini di acquisizione e fornitura dati ma anche, in alcuni casi, pensando ad eventuali sviluppi futuri.

Le necessità primarie erano:

- Svicolare l'operazione di lettura da quella di scrittura.
- Gestire l'operazione di lettura dei dati presenti nel buffer di Arduino, conseguenti ad un comando di scrittura, in modo che la lettura fosse fatta fintanto che all'interno del buffer stesso fossero presenti dei dati. Questo consente di garantire lo svuotamento completo del buffer.

Per il primo punto è bastato togliere il comando di lettura del buffer, conseguente al comando di scrittura (nell'appendice Listati Matlab è possibile notare che, rispetto alla function ScriviSeriale1.m, nella function ScriviSeriale2.m manca il comando di lettura).

Per ciò che riguarda la lettura del buffer, il problema può essere affrontato in due modi:

- Utilizzando un comando di lettura della porta seriale che consente non solo di leggere il dato, ma di rendere disponibile un messaggio d'errore qualora il buffer non contenga dati (vd. nell'appendice Listati Matlab, LeggiSeriale1.m e LeggiSeriale2.m).
- Acquisire i dati in maniera diversa a seconda del comando che viene fornito in ingresso. Infatti attualmente i comandi che vengono immessi, possono portare ad avere in risposta uno o tre dati che vengono messi a disposizione nel buffer d'uscita di Arduino. Dunque, a seconda del comando che viene fornito, verranno operate una oppure tre letture del buffer d'uscita (vd. nell'appendice Listati Matlab, LeggiSeriale3.m).

La prima modalità consente di operare continuamente la lettura del buffer, indipendentemente dai dati che vengono messi a disposizione in uscita, fintanto che non compare il messaggio d'errore relativo all'assenza di dati nel buffer. Quindi da un punto di vista di elasticità nella lettura del buffer d'uscita risulta migliore (non essendo vincolata ad un numero predefinito di letture); il problema di questa modalità di lettura sta nel fatto che il messaggio d'errore relativo all'assenza di dati nel buffer viene fornito in un tempo troppo lungo e questo rallenta notevolmente l'acquisizione.
Al momento al buffer d'uscita possiamo avere solamente uno oppure tre dati, quindi verrà utilizzata la seconda metodologia di lettura della porta seriale. Questa è sicuramente meno elastica ma sicuramente molto pratica e veloce. I dati che vengono acquisiti vengono salvati all'interno di un vettore di dimensione 1x1 oppure 1x3 (a seconda del comando inviato).

3.2 Rilevazione delle frequenze in catena aperta

Il primo obiettivo che si voleva raggiungere era l'introduzione di una matrice di trasformazione; questa consentirà di passare da una terna di frequenze che desideriamo rilevare ai sensori TAOS, ad una terna di correnti da inviare ai LED.
Per poter realizzare la matrice di trasformazione frequenze-correnti, verifichiamo inanzitutto che le correnti misurate da Arduino siano realmente quelle circolanti ai LED. La misura viene realizzata mediante l'utilizzo di un multimetro; questo, inserito tra il polo positivo del morsetto di alimentazione dei LED e il connettore collegato all'anodo del primo LED che compone la serie degli stessi, consentirà di misurarne la corrente circolante.

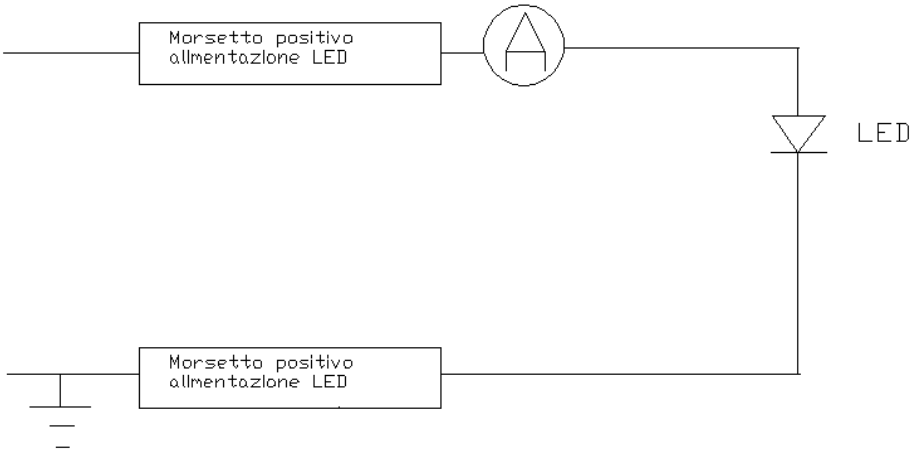


Figura 32: Misura della corrente inviata ai LED

Le rilevazioni effettuate hanno portato alla luce una grossa discrepanza tra la corrente misurata dal multimetro e la relativa tensione di shunt misurata da Arduino. In particolare la corrente misurata dal multimetro risulta minore rispetto a quella rilevata da Arduino. Impostando infatti il setpoint di corrente a 0,7 A c'era uno scostamento tra le due misure:

	Corrente misurata con amperometro	Corrente misurata da Arduino
LED ROSSI	0,68 A	0,7 A
LED VERDI	0,675 A	0,7 A
LED BLU	0,595 A	0,7 A

La causa di questo scostamento potrebbe essere l'introduzione nel firmware di Arduino preesistente di valori di resistenze di shunt non corrispondenti a quelle reali.
Per poterne rilevare il valore reale, è stato utilizzato un secondo multimetro che consenta di misurare la tensione ai capi della resistenza di shunt. Ciò che si è potuto notare è la grossa discrepanza tra il valore di tensione rilevato alla resistenza di shunt e il valore che rilevo al polo negativo del morsetto d'alimentazione dei LED.

	Tensione allo shunt	Tensione al polo negativo del morsetto d'alimentazione
LED ROSSI	0,658 V	0,686 V
LED VERDI	0,655 V	0,665 V
LED BLU	0,625 V	0,651 V

Questi valori dovrebbero coincidere, visto che i punti sono tra loro connessi (vd. schema di connessione al paragrafo 2.1.6.2 Misura della corrente). La causa è probabilmente da

attribuire alla presenza di grosse resistenze di contatto all'interno del morsetto d'alimentazione del LED.
È stata realizzata quindi dapprima una pulizia di tutti i contatti dei morsetti d'alimentazione. Successivamente sono stati sostituiti anche quelli che presentavano maggiori differenze tra i valori di tensione precedentemente rilevati (rossi e blu). Questo ha portato alla riduzione delle resistenze di contatto a valori accettabili. Infatti impostando il setpoint di corrente per ogni canale a 0,63 A, le tensioni alla resistenza di shunt e al polo negativo del morsetto d'alimentazione dei LED diventano:

	Tensione allo shunt	Tensione al polo negativo del morsetto d'alimentazione
LED ROSSI (0,63 A)	0,626 V	0,638 V
LED VERDI (0,63 A)	0,636 V	0,640 V
LED BLU (corrente max 0,592A)	0,610 V	0,597 V

A questo punto è possibile ricavare il valore della resistenza di shunt, che verrà anche implementata all'interno del firmware di Arduino.

$$R_{shunt\_r} = \frac{Tensione\ allo\ shunt}{Corrente\ circolante} = \frac{0{,}626}{0{,}630} = 0{,}994\ \Omega$$

$$R_{shunt\_g} = \frac{Tensione\ allo\ shunt}{Corrente\ circolante} = \frac{0{,}636}{0{,}630} = 1{,}009\ \Omega$$

$$R_{shunt\_b} = \frac{Tensione\ allo\ shunt}{Corrente\ circolante} = \frac{0{,}597}{0{,}592} = 1{,}008\ \Omega$$

Una volta inseriti questi valori all'interno del firmware, si è verificato che la misura di corrente operata da Arduino è paragonabile con quella operata dal multimetro. Nonostante le piccole differenze legate alla presenza delle resistenze di contatto, i risultati possono ritenersi soddisfacenti.
Impostando il setpoint di corrente a 0,7 A si è inoltre notato che le correnti realmente circolanti sono inferiori. Dunque il valore di 0,7 A, indicato nei datasheet del driver come massima corrente erogabile, è ottimistico. Infatti allo stato attuale i driver riescono a garantire una corrente massima erogabile pari a:

- 0,680 A per il driver d'alimentazione dei LED rossi.
- 0,675 A per il driver d'alimentazione dei LED verdi.

- 0,595 A per il driver d'alimentazione dei LED blu.

Per il calcolo della matrice di trasformazione, la corrente ai LED blu deve essere limitata a questo valore. Per una questione d'ordine, poniamo un setpoint di corrente uguale per tutte e tre le serie di LED e pari a 0,560 A (spr0800, spg0800, spb0800).

Per la sua realizzazione sono state effettuate tre prove:

- solo LED rossi alimentati a 560 mA.
- solo LED verdi alimentati a 560 mA.
- solo LED blu alimentati a 560 mA.

Per ognuna di queste tre prove, è stata operata una misura di frequenza ai tre diversi gruppi di fotodiodi del sensore TAOS e delle correnti inviate ai LED, per un tempo pari a circa 20 minuti.

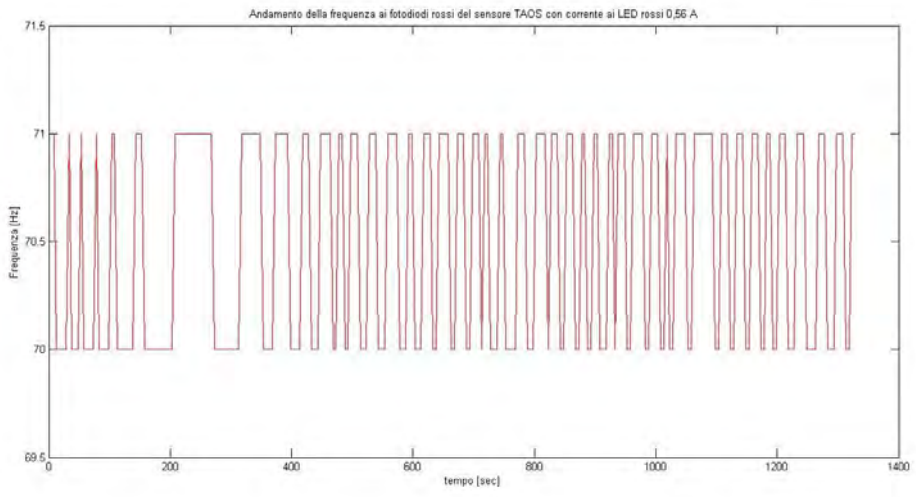


Figura 33: andamento della frequenza ai fotodiodi rossi del sensore TAOS alimentando i soli LED rossi

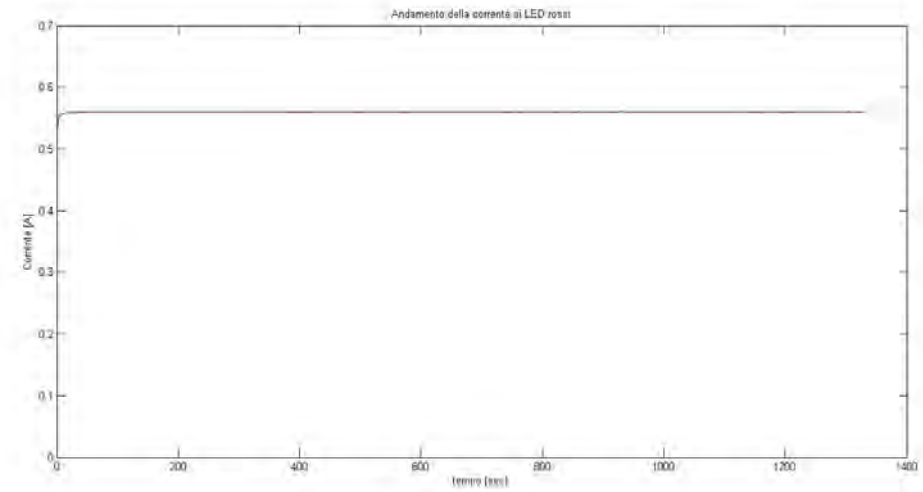


Figura 34: Andamento della corrente ai LED rossi

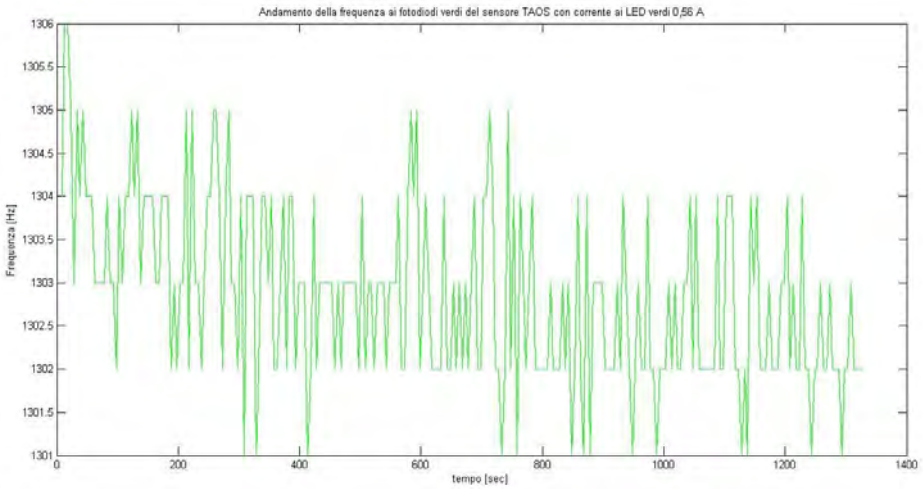


Figura 35: andamento della frequenza ai fotodiodi verdi del sensore TAOS alimentando i soli LED verdi

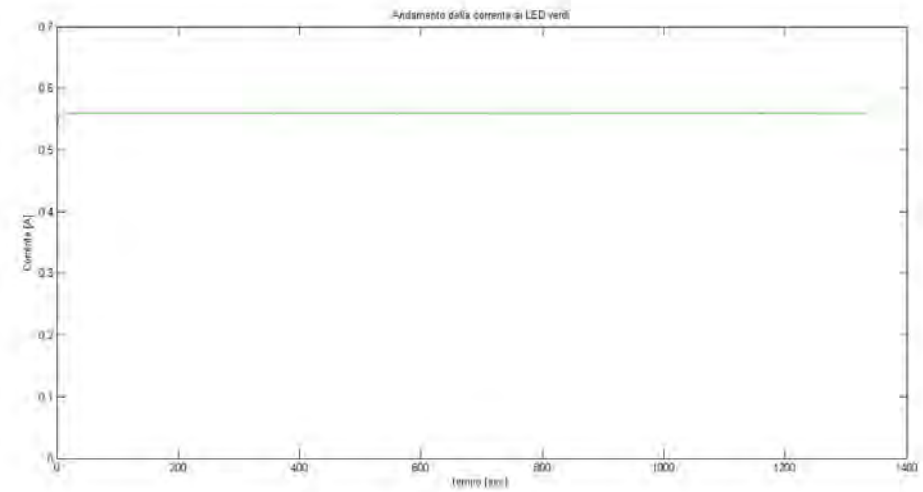


Figura 36: Andamento della corrente ai LED verdi

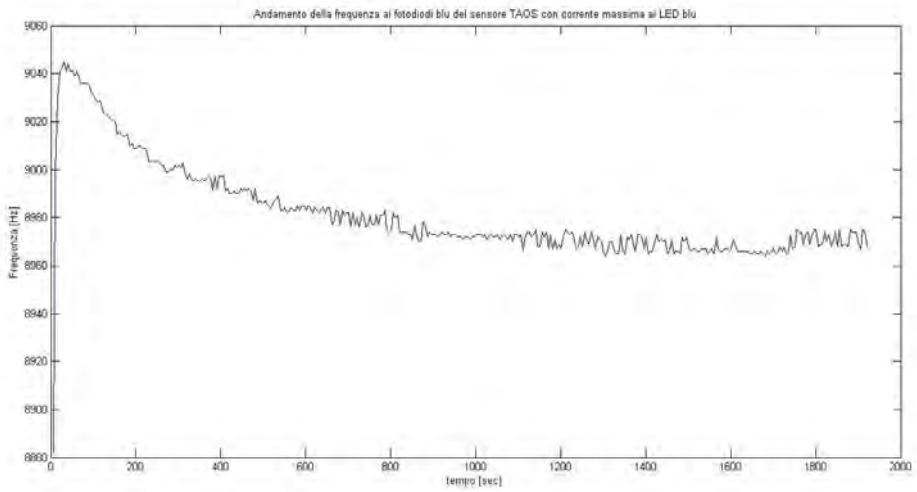


Figura 37: andamento della frequenza ai fotodiodi blu del sensore TAOS alimentando i soli LED blu

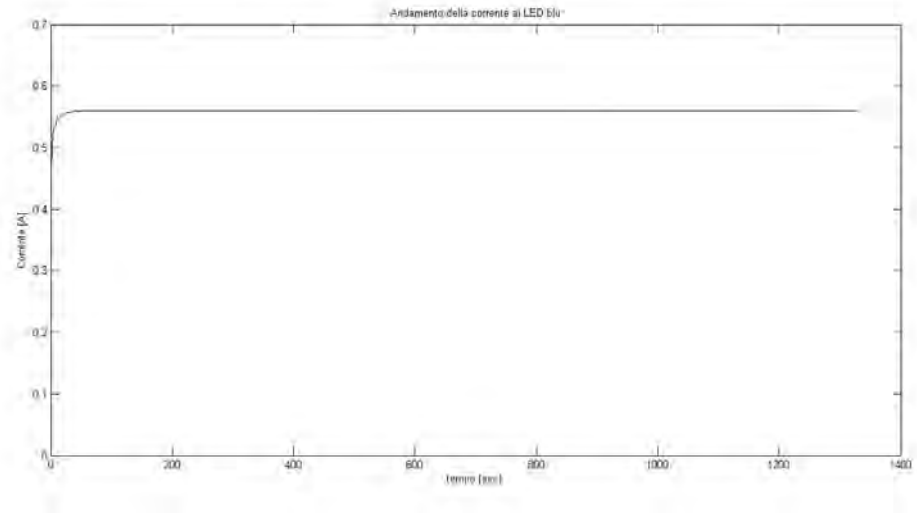


Figura 38: Andamento della corrente ai LED blu

Il fatto di rilevare una variazione della frequenza misurata ai fotodiodi blu e verdi del sensore TAOS alimentando rispettivamente i soli LED blu e verdi rappresenta un problema, perchè questo sta a dimostrare come ad una determinata corrente ai LED, non corrisponde un univoco segnale di frequenza. Dunque la realizzazione della matrice di trasformazione frequenze correnti non può essere realizzata allo stato attuale.

3.3 Analisi delle possibili cause di riduzione della frequenza rilevata ai sensori TAOS

È stata esclusa come causa una possibile variazione della corrente di alimentazione ai LED che, come si è potuto vedere nel precedente paragrafo, si mantiene costante.

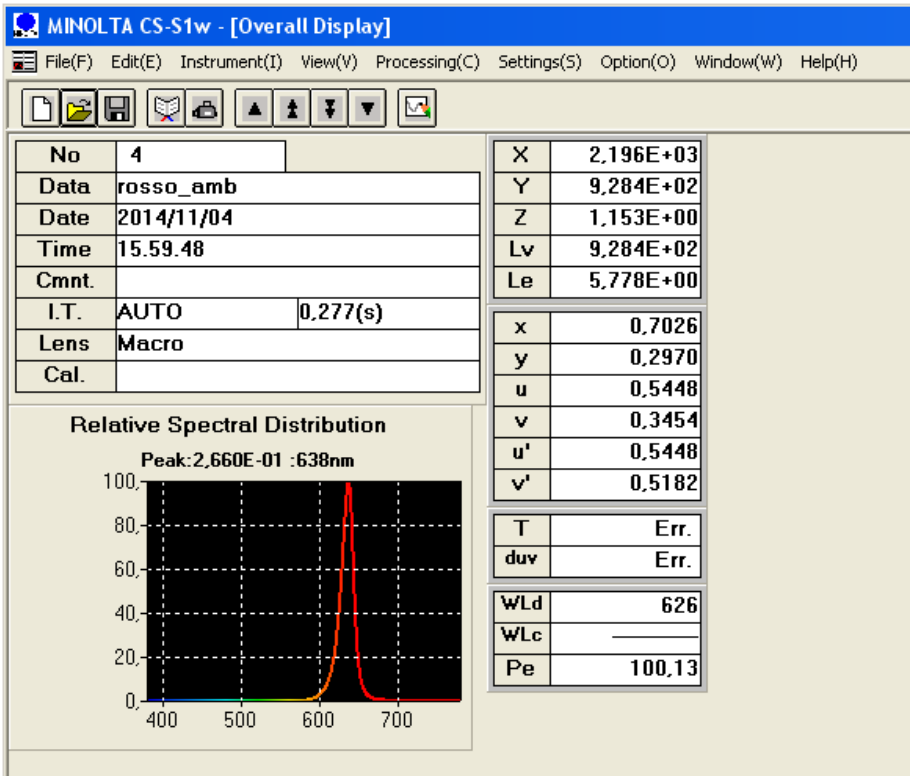
Per analizzare le possibili cause che portano a questo divario nella rilevazione delle frequenze al sensore TAOS TCS230, è necessario operare delle prove.
Le prove operate consistono nell'effettuare delle misure di:

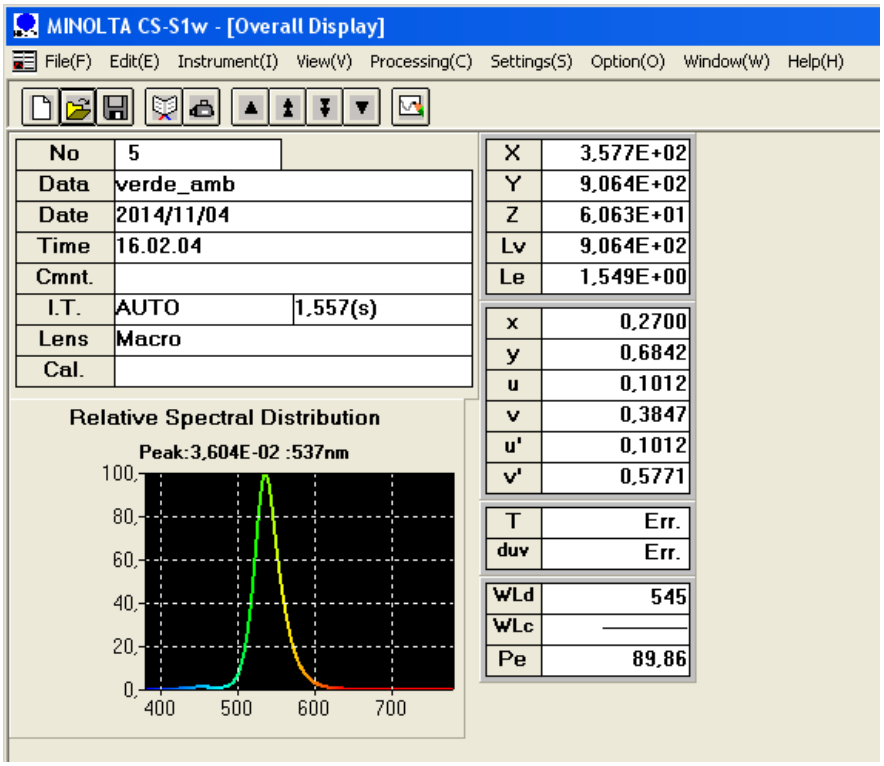
- Temperatura alla piastra: mediante il comando mte da Teraterm.
- Tensione alla serie delle famiglie dei LED (con l'ausilio di un oscilloscopio).
- Spettro della radiazione emessa (mediante l'ausilio di uno spettroradiometro).
- Frequenze in uscita dal sensore TAOS.

Nel primo caso queste misure sono state operate a temperatura ambiente (25-26°C) ed alimentando prima i soli LED rossi, poi i soli LED verdi ed infine i soli LED blu alla massima potenza.

0 min	TAOS R,G,B	Temperatura [°C]	Tensione LED [V]
SPR1000	85,9,1	25,06	4,934
SPG1000	55,1424,373	24,58	7,2
SPB1000	263,1773,9154	25,06	17,782

Figura 39: Risultati di misura ai sensori TAOS TCS230, al sensore di temperatura e tensione ai LED a temperatura ambiente.





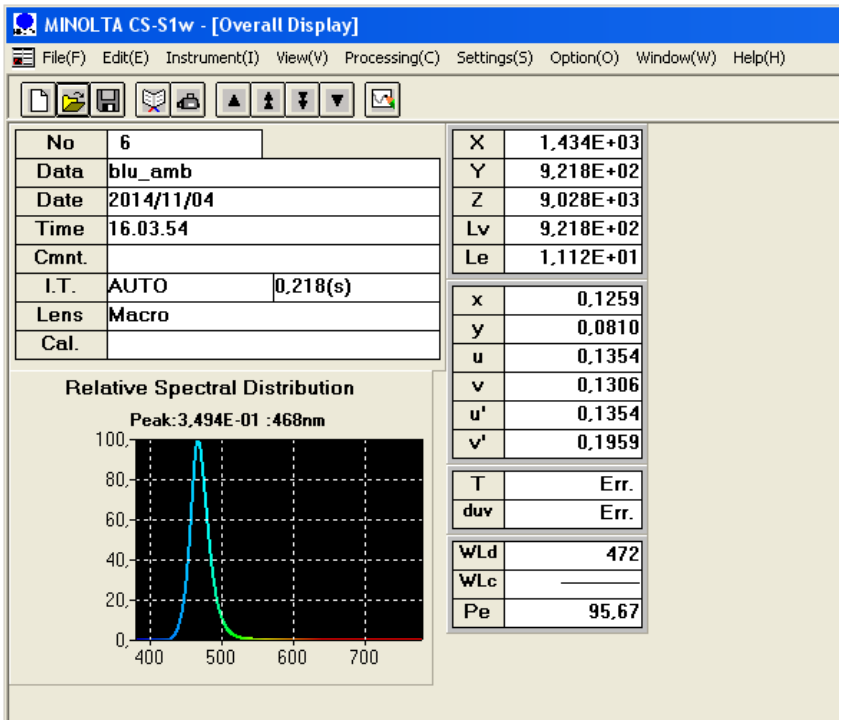
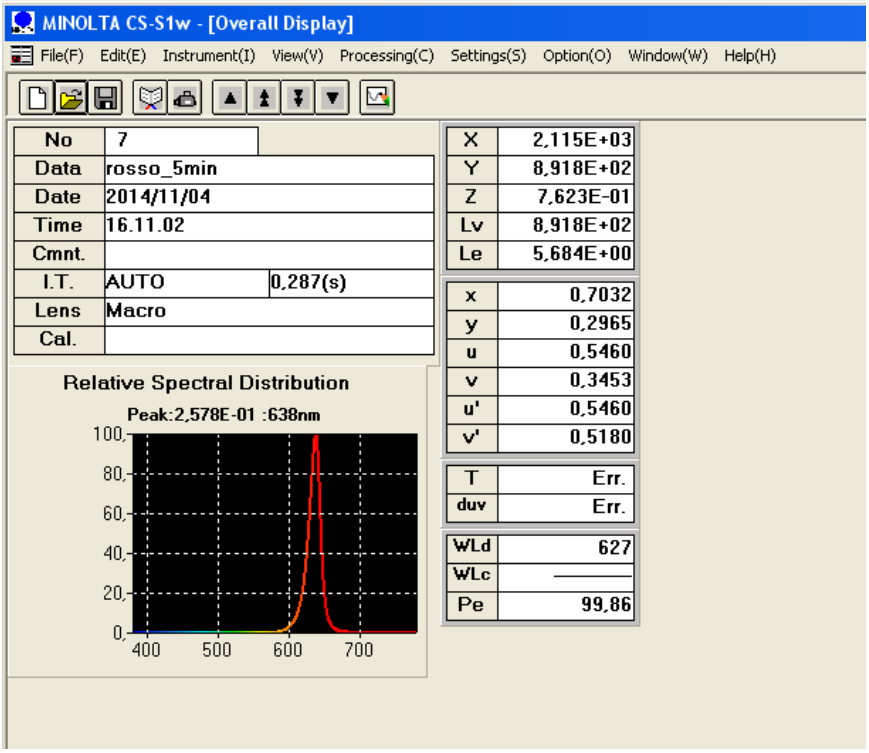


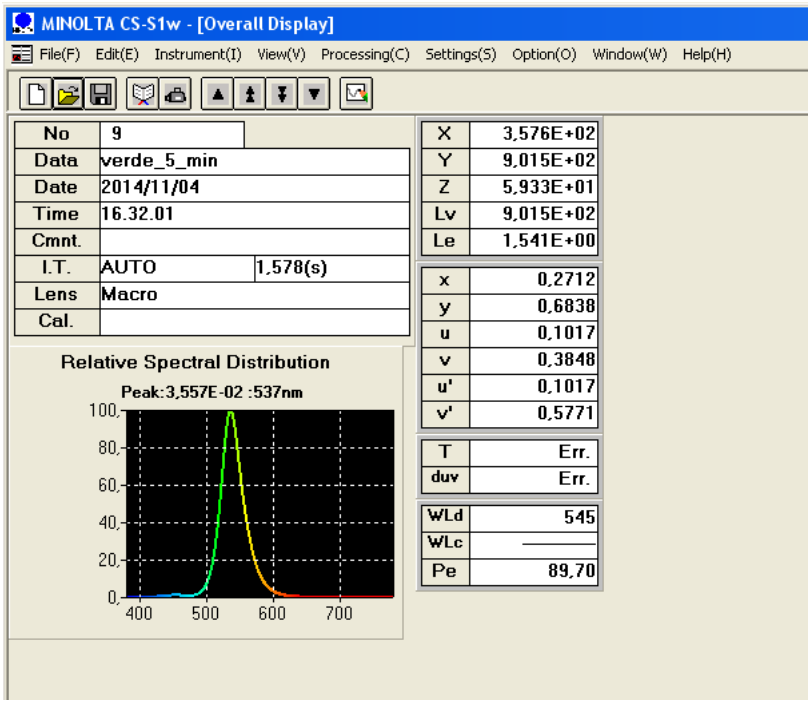
Figura 40: Risultati dello spettroradiometro con analoghe condizioni di temperatura alimentando rispettivamente i soli LED rossi, verdi e blu.

Dopo aver alimentato tutti i LED alla massima corrente per un tempo pari a 5 minuti (per cui si suppone terminato il transitorio termico dei LED), sono state realizzate, per la seconda prova, delle misure analoghe alle precedenti.

5min	TAOS R,G,B	Temperatura [°C]	Tensione LED [V]
SPR1000	81,8,1	29,36	4,85
SPG1000	54,1406,368	29,36	7,068
SPB1000	261,1781,9096	29,84	17,608

Figura 41: Risultati di misura ai sensori TAOS TCS230, al sensore di temperatura e tensione ai LED a temperatura di regime del sistema piastra ventola.





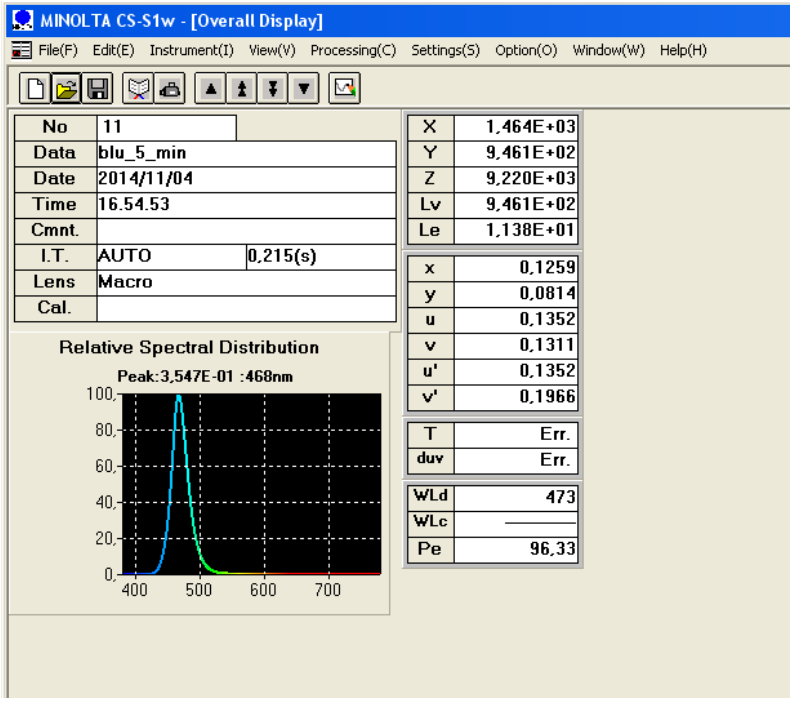
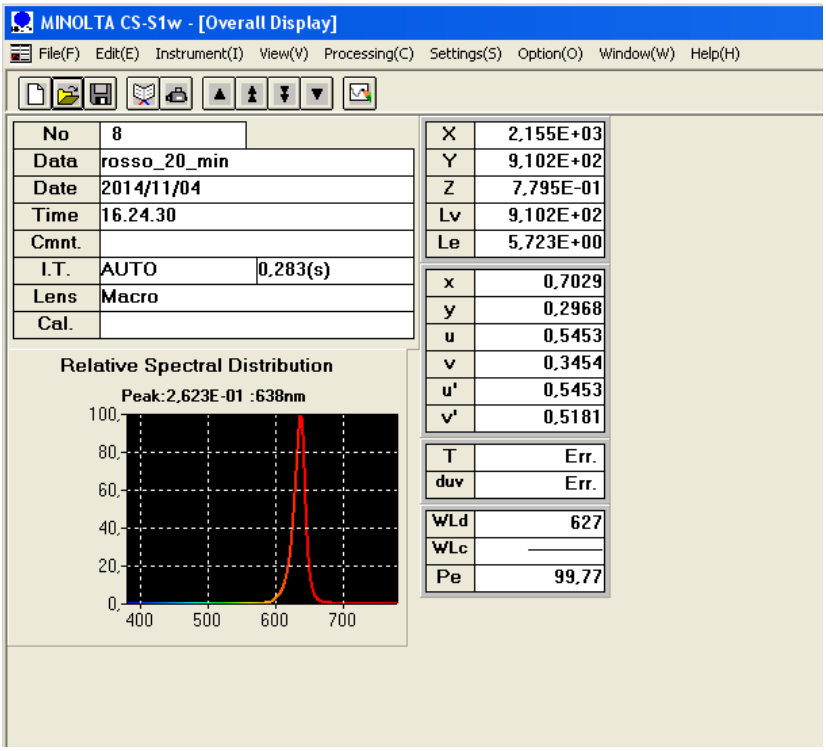


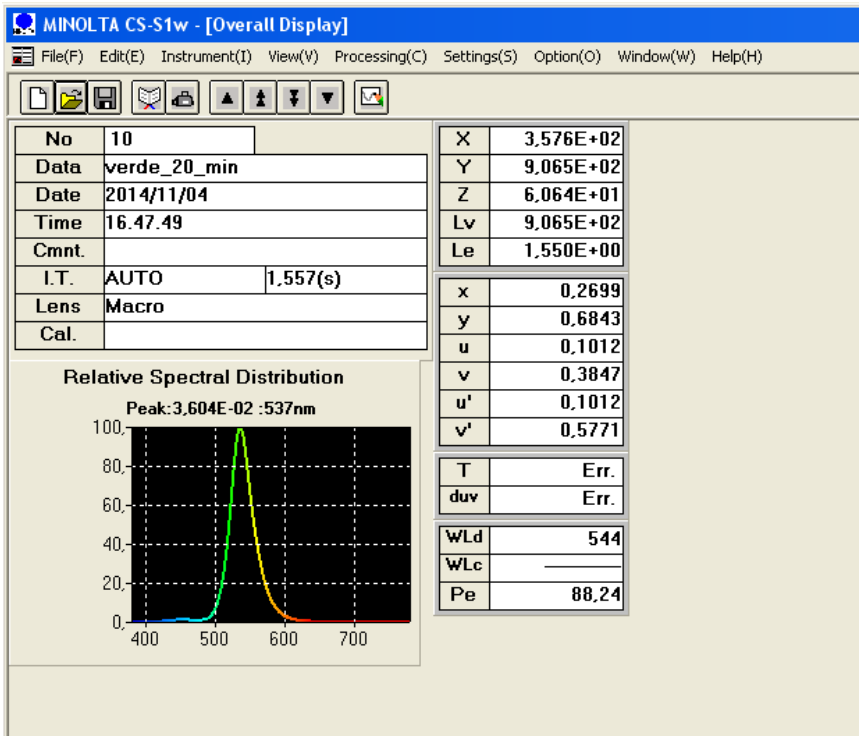
Figura 42: Risultati dello spettroradiometro con analoghe condizioni di temperatura alimentando rispettivamente i soli LED rossi, verdi e blu.

Partendo dal regime termico dei LED realizzato nella seconda prova, sono stati alimentati in successione i LED rossi, verdi e blu alla massima corrente per un tempo pari a 20 minuti (per cui si suppone terminato il transitorio termico della piastra); trascorso questo tempo sono state effettuate le misure citate per la prima e la seconda prova.

20min	TAOS R,G,B	Temperatura [°C]	Tensione LED [V]
SPR1000	83,8,1	25,54	4,85
SPG1000	55,1404,368	26,49	7,068
SPB1000	254,1734,8838	28,88	17,608

Figura 43: Risultati di misura ai sensori TAOS TCS230, al sensore di temperatura e tensione ai LED a temperatura che il sistema piastra ventola assume accendendo alternativamente i LED.





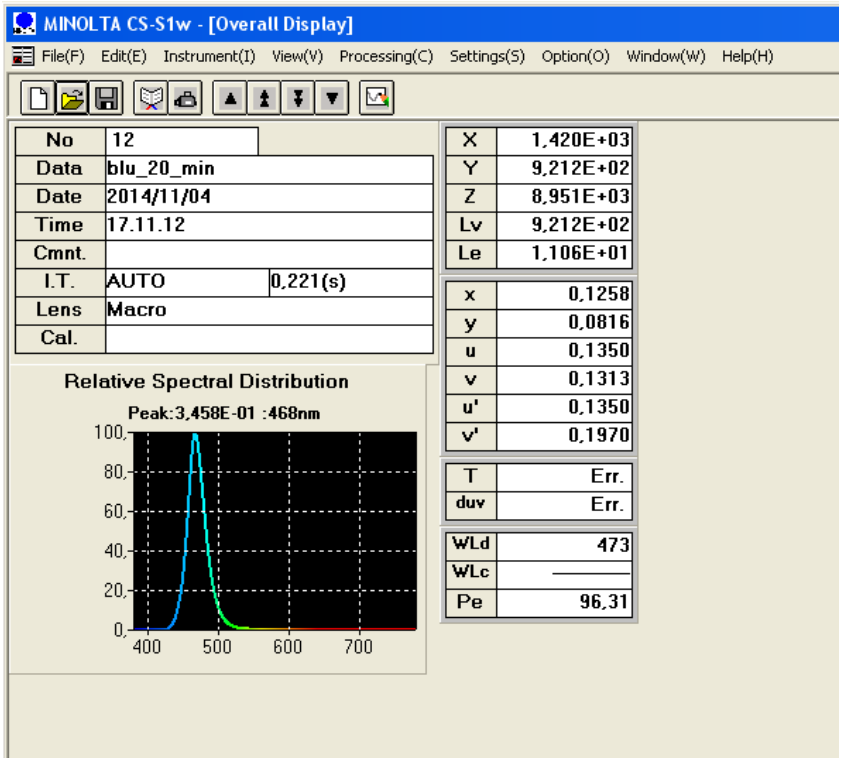


Figura 44: Risultati dello spettroradiometro con analoghe condizioni di temperatura alimentando rispettivamente i soli LED rossi, verdi e blu.

Le prove effettuate hanno lo scopo di valutare il comportamento del sistema una volta esauriti i transitori termici che regolano il sistema:

- Quella dei LED: che è una dinamica termica molto rapida (che si suppone terminata dopo 5 minuti).
- Quella della piastra metallica: dinamica invece molto più lenta (che si suppone termintata dopo 20 minuti).

Queste hanno evidenziato che le tensioni rilevate a capo delle famiglie di LED si mantengono costanti; dunque non sono queste le variabili che giocano sulla variazione dell'emissione dei LED.

D'altra parte è stato constatato che partendo dalla condizione di temperatura di regime della piastra, accendendo i soli LED rossi oppure verdi la temperatura alla piastra tende a scendere a valori paragonabili a quelli rilevati nella prima prova; mentre accendendo i soli LED blu la piastra tende a mantenersi ad una temperatura pari a quella di regime che si ottiene con tutti i LED accesi.
Il confronto tra la prima e la seconda prova hanno avuto lo scopo di valutare la dinamica termica dei LED. L'esaurimento della stessa ha portato ad avere:

- Riduzione della frequenza rilevata ai fotodiodi rossi del sensore TAOS, alimentando i soli LED rossi, del 5%. Lo spettroradiometro rileva una riduzione del picco del 3%.
- Riduzione della frequenza rilevata ai fotodiodi verdi del sensore TAOS, alimentando i soli LED verdi, del 1,3%. Stessa riduzione viene valutata anche dallo spettroradiometro.
- Riduzione della frequenza rilevata ai fotodiodi blu del sensore TAOS, alimentando i soli LED blu, del 0,6%. Lo spettroradiometro rileva un aumento dell'1,5%.

Il confronto tra la seconda e la terza prova hanno lo scopo di verificare il comportamento della sorgente una volta terminato il transitorio termico della piastra. In questo frangente è stato rilevato:

- Aumento della frequenza rilevata ai fotodiodi rossi del sensore TAOS, alimentando i soli LED rossi, del 2,5% rispetto alla seonda prova. Lo spettroradiometro rileva un aumento dell'1,7%.

- Riduzione della frequenza rilevata ai fotodiodi verdi del sensore TAOS, alimentando i soli LED verdi, del 0,13% rispetto alla seconda prova. Lo spettroradiometro rileva un aumento dell'1,3%.
- Riduzione della frequenza rilevata ai fotodiodi blu del sensore TAOS, alimentando i soli LED blu, del 2,9% rispetto alla seconda prova. Lo spettroradiometro rileva una riduzione del 2,6%.

Il confronto tra le misure effettuate con lo spettroradiometro e il sensore TAOS, porta ad avere in alcune occasioni dei risultati discordanti.

Ciò che però è interessante è vedere che comunque nell'emisione dei LED non gioca un ruolo importante solo la dinamica termica dei LED, ma anche la dinamica termica della piastra. Infatti, alimentando i soli LED rossi e verdi la temperatura di regime termico della piastra tende a portarsi a valori inferiori rispetto alla temperatura ambiente. Ciò sta a significare che l'asportazione dell'energia prodotta sottoforma di calore da parte dei LED viene per la maggior parte asportata dal sistema piastra ventola (con dinamiche termiche lente), senza di fatto portare ad aumenti sostanziali della temperatura di giunzione dei LED e quindi ad un degrado significativo delle prestazione degli stessi.

La situazione più gravosa riguarda la rilevazione ai fotodiodi blu alimentando i soli LED blu. In questo caso la dissipazione di potenza per effetto Joule risulta più alta (sono installati infatti 5 LED blu, mentre sia i LED rossi che verdi sono 2), rispetto all'accensione dei soli LED rossi e verdi, e questo consente di mantenere, tra seconda e terza prova, una temperatura della piastra sempre uguale.
Questa condizione significa che lo scambio di calore tra LED e il sistema piastra non consente di asportare totalmente il calore; questo si tramuta in un aumento di temperatura di regime della piastra e di conseguenza anche dei LED (con conseguente degrado prestazionale).

Eliminando il transitorio termico della piastra e portando la stessa a funzionare a temperatura costante, il degrado prestazionale dei LED dovrebbe essere legato alla sola dinamica termica degli stessi, che però è molto più veloce. Dunque la rilevazione dei sensori TAOS, una volta terminato il transitorio termico dei LED, dovrebbe mantenersi costante.

3.4 Problematiche sorte nel realizzare le misure

Volendo intensificare il campionamento delle frequenze rilevate ai fotodiodi blu, quello che abbiamo potuto constatare è che, ponendo la frequenza di campionamento pari a quella massima, la frequenza del segnale in uscita al fotodiodo blu assume dei valori che crescono esponenzialmente fino ad un valore di regime, per poi scendere ad un nuovo valore.

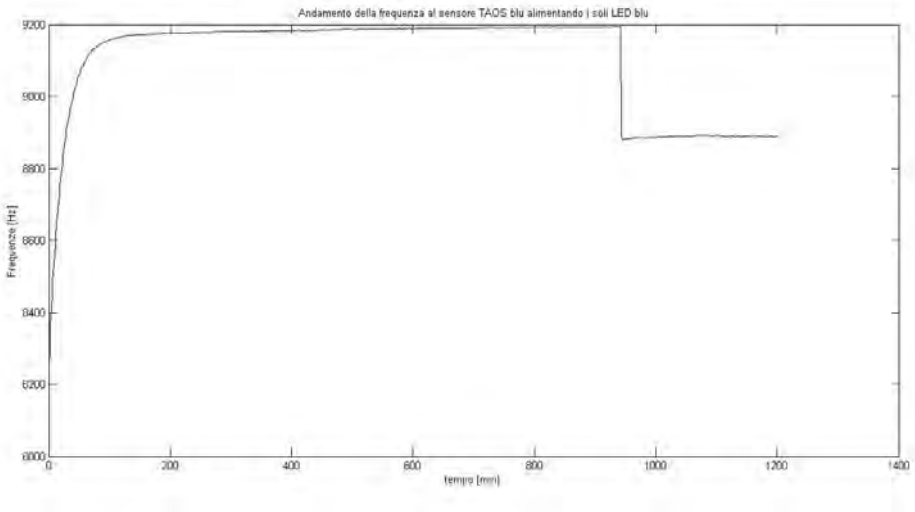


Figura 45: Andamento della frequenza al fotodiodo blu intensificando al massimo il campionamento

quello che si è voluto successivamente voluto verificare è se tale variazione di frequenza sia da attribuire ad una variazione delle correnti ai LED. Per fare ciò è stato realizzato un file Matlab (vd. Appendice Listati Matlab lettura_rgb3.m) che consenta, oltre alla rilevazione delle frequenze misurate dal sensore TAOS blu, di rilevare la corrente inviata ai LED blu.

È stato fatto un campionamento alla massima velocità possibile sia per la frequenza sia per la corrente così da poter avere un numero di campioni che fosse il più ampio possibile.

Ciò che è stato riscontrato è, per prima cosa, che la variazione brusca della frequenza è da attribuire alla variazione della corrente inviata ai LED. In secondo luogo, la richiesta continua di informazioni ad Arduino, che nel frattempo sta operando un controllo delle correnti inviate ai LED, comporta una rilevazione di valori di corrente diverse a quella che ci aspetteremo.

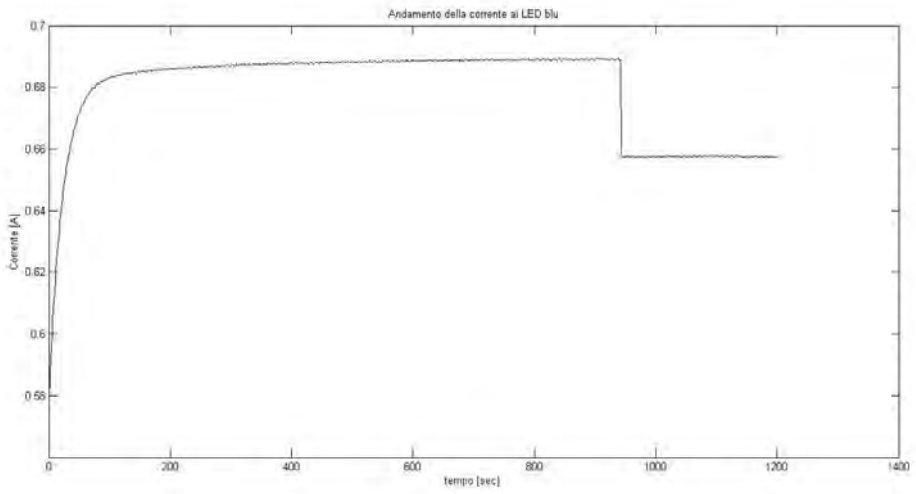


Figura 46: Andamento della corrente al LED blu

La corrente rilevata, oltre ad avere un valore errato legato ai problemi riscontrati al paragrafo "3.2 Rilevazione delle frequenze in catena aperta", presenta un andamento di certo non soddisfacente e non in linea con l'andamento che ci si aspetta di trovare.

La causa è con ogni probabilità da attribuire al fatto che Arduino deve gestire un controllo di correnti; durante il controllo vengono richieste delle misure di frequenze che in termini temporali risultano gravose. Questo vuol dire che, per poter rendere disponibile al buffer d'uscita il dato relativo alla misura richiesta, Arduino abbandona momentaneamente il controllo di corrente. Se l'operazione è a livello temporale poco gravosa rispetto al transitorio delle correnti ai LED, il controllo avviene senza alcun problema. Se invece, come nel caso della misura delle frequenze, risulta gravosa a livello temporale, il controllo delle correnti ai LED viene perso.

Le soluzioni applicabili a tale problematica sono due:

- Attribuire la misura della frequenza (estremamente gravosa in termini temporali) ad una seconda scheda Arduino, così da eliminare il problema dell'impossibilità di Arduino di essere multitasking.
- Rallentare l'acquisizione delle frequenze, e quindi accontentarci di un numero di campioni ridotto rispetto a quelle che sono le reali capacità computazionali, mantenendo una sola scheda Arduino.

Anche se la prima soluzione risulterebbe la più corretta per garantire una misura di frequenza migliore, ci accontentiamo, per praticità, di rilevare un numero di campioni di frequenza ridotto e di mantenere l'unica scheda Arduino.

CAPITOLO 4: Controllo della temperatura alla piastra

Avendo individuato nell'aumento della temperatura dei LED la causa che porta al degrado prestazionale degli stessi, cerchiamo di individuare una soluzione che consenta di eliminare questa riduzione in termini di emissione dei LED.

Il problema potrebbe essere risolto uniformando, qualsiasi sia la condizione di lavoro, la temperatura alla piastra. In questo modo non dovremmo vedere una riduzione della frequenza rilevata ai sensori TAOS una volta raggiunto il regime termico dei LED, e quindi i LED non dovrebbero subire alcun degrado prestazionale.

Per poter realizzare un sistema che garantisca la stabilità della temperatura, l'idea è quella di agire in due modalità:

- Riscaldando la piastra: se la temperatura scende al di sotto di una certa soglia minima.
- Raffreddando la piastra: se la temperatura va al di sopra di una soglia massima.

Per il riscaldamento della piastra vengono utilizzati dei resistori che dissipano energia per effetto Joule; questa viene trasferita per conduzione alla piastra metallica d'installazione dei LED, garantendo l'aumento della temperatura della piastra. Per ciò che riguarda il raffreddamento agiremo attraverso l'accensione e lo spegnimento della ventola già installata.

L'attivazione dei due sistemi viene gestito attraverso un'isteresi, che consenta di mantenere la piastra ad una temperatura attorno ai 29 °C. L'entità della variazione attorno a questo valore sarà definito dall'ampiezza dell'isteresi implementata. Questa temperatura è stata scelta perchè rappresenta la temperatura che raggiunge la piastra a regime quando ai LED viene inviata la massima corrente, mantenendo la ventola sempre accesa (condizione più critica da un punto di vista termico).

La piastra presenta una costante termica molto alta; quindi prima di poter mettere in esercizio il sistema è necessario attendere che la piastra raggiunga la temperatura obiettivo che è stata scelta.

4.1 Scelta dei valori di resistenza

Per garantire alla piastra il raggiungimento della temperatura prefissata, è necessario che i riscaldatori siano in grado di produrre la stessa energia che viene dissipata dai LED sottoforma di calore.

Supponendo che tutti i LED abbiano una tensione di giunzione pari a quella dei LED blu (3,4 V, invece di 1,7 V che realmente abbiamo ai LED verdi e rossi), alla corrente di alimentazione massima dei LED di 1 A avremmo una potenza di:

$$P = V * I = 3{,}4 * 1 \cong 3\ W$$

ciò vuol dire che alla massima corrente che posso fornire con gli alimentatori LUXDRIVE 3021 D-E-700 BuckPuck (suppongo di avere 700 mA ai LED, sovradimensionando in realtà la potenza dissipata) ho una potenza ai LED di:

$$P = V * I = 3 * 0{,}7 \cong 2\ W$$

Supponendo inoltre che tale potenza sia interamente dispersa in calore da parte del LED (cosa che in realtà non è perchè ne viene dispersa solo 3/4) ciò vuol dire che, con tutti i LED accesi alla massima potenza, ho:

$$Ptot = nled * P = (2led_r + 2led_v + 5led_b) * 2 \cong 20\ W$$

la resistenza di cui necessito per smaltire l'intera potenza è pari a:

$$R = \frac{P}{I^2} = \frac{20}{0{,}7^2} \cong 40$$

con tensione ai suoi capi pari a:

$$V = R * I = 40 * 0{,}7 = 28\ V\ (> 24\ Vmax \text{dall alimentatore})$$

Utilizzando due alimentatori LUXDRIVE 3021 D-E-700 BuckPuck (700 mA), posso ripartire equamente la potenza da dissipare (così da poterla anche distribuire in più punti della piastra) in due resistenze ognuna di valore pari a:

$$R = \frac{P}{I^2} = \frac{10}{0{,}7^2} = 20{,}4$$

La resistenza commerciale più vicina sarebbe da 22 ma, avendo però a disposizione delle resistenze da 50Ω, adattiamo la corrente su tale valore di resistenza; quindi la corrente che dovremmo inviare per ottenere una dispersione totale di 20 W sarà pari a:

$$I = \sqrt{\frac{P}{R}} = \sqrt{\frac{20}{(50+50)}} = 0.447 \text{ A}$$

Con una tensione ai suoi capi pari a:

$$V = R * I = 50 * 0{,}447 = 22.35 \text{ V} (< 24\ Vmax\ dall\ alimentatore)$$

Il fatto di avere a disposizione una resistenza da 50 ha rappresentato da un lato anche un vantaggio. Infatti in questo modo la corrente risulta molto più bassa a parità di potenza da dissipare quindi, essendo la corrente massima di 0,7 A un valore ottimistico ottenibile dai driver stessi, realisticamente ottenibile dai driver stessi.

4.2 Modifiche apportate al Firmware di Arduino

Per poter operare il controllo della temperatura rilevata alla piastra d'installazione dei LED è stato necessario apportare delle modifiche al Firmware preimplementato in Arduino.

4.2.1 Introduzione del ciclo d'isteresi

L'implementazione del ciclo di isteresi è stata scelta per impedire che ci fosse una continua variazione del comando di accensione della ventola e dei resitori, quando la temperatura non fosse perfettamente uguale a quella prefissata come temperatura di esercizio (29 °C).
Sono state poste due soglie di temperatura:

- Superiore: pari 30 °C.
- Inferiore: pari a 28 °C.

Il concetto è abbastanza intuitivo; se viene rilevata una temperatura inferiore alla soglia inferiore vengono attivati i resistori e spenta la ventola fintanto che la temperatura non

supera i 30°C; altrimenti se viene superata la soglia superiore, vengono spenti i resistori ed attivata la ventola fintanto che la temperatura non scende al di sotto della soglia inferiore.

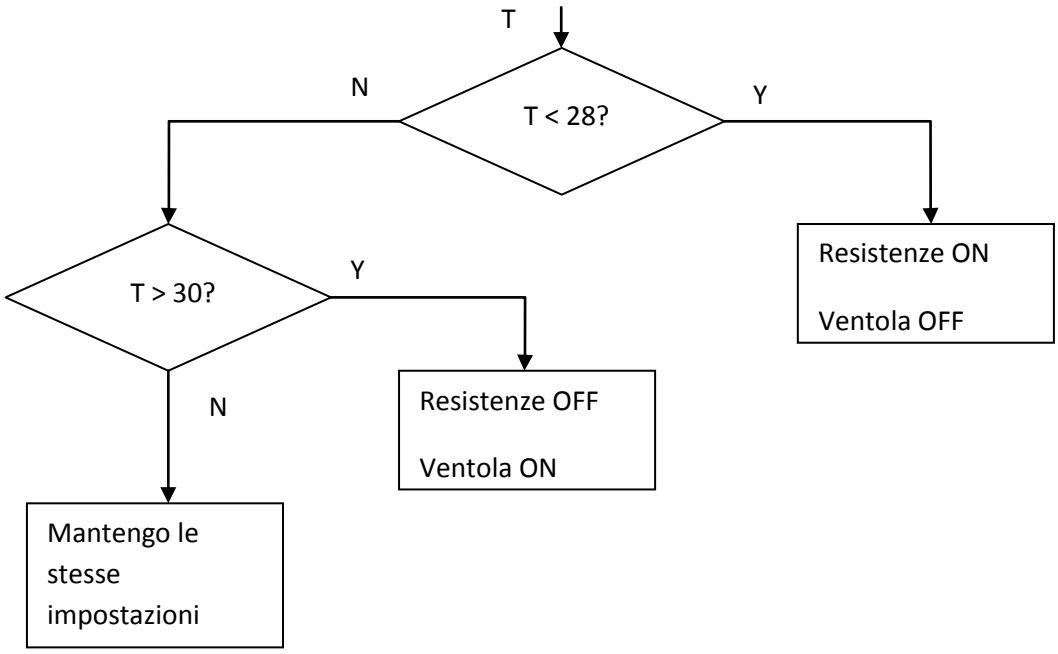


Figura 47: Schema di funzionamento del ciclo d'isteresi

L'accensione e lo spegnimento della ventola avviene rispettivamente portando a 4,2 V o a 0 V il valore della tensione PWM al morsetto corrispondente. Questo segnale consentirà come vedremo di comandare il morsetto gate del mosfet, che fungerà da interuttore per la ventola.

Per quanto riguarda i resistori l'accensione avviene semplicemente impostando il setpoint di corrente che vogliamo far circolare. Il controllo della corrente verrà poi realizzato, come vedremo nel paragrafo "4.2.2 Controllo delle correnti di alimentazione dei resistori", in catena chiusa.

Lo spegnimento invece non viene semplicemente realizzato impostando il setpoint di corrente a zero, ma impostando anche la tensione di ctrl dei driver a 4,2 V (così da garantire la circolazione di corrente nulla). Questa tecnica consente di ottenere una corrente erogata nulla in tempi minori rispetto a quelli che otterrei impostando semplicemente il setpoint a zero. Se addottassi quest'ultima tecnica infatti dovrei attendere un tempo legato al transitorio della corrente di alimentazione dei resistori.

Questa scelta comporta anche, ad ogni spegnimento, una nuova inizializzazione delle variabili che rappresentano i contributi di errore (proporzionale, integrale e derivativo) e il setpoint di corrente che si imposta.

```
if(temperatura < 28)          // Isteresi sul valore massimo di temperatura ammissibile, nel caso in cui superi l'isteresi sul valore massimo non invio corrente ai riscaldatori e comando la chiusura del mosfet con il comando al gate
  {

    tensioneCtrlVent = 0;              // Imposto il valore dell'uscita PWM a zero così da impedire la circolazione di corrente tra i contatti di drain e source del MOSFET
    setPointRes = setPointResImp;

  }
    else
    {
      if(temperatura > 30)

      {
        setPointRes = 0;          // Azzero il setPoint di corrente impostato

        tensioneCtrlCalcolataRes_1 = 4.2;  //spengo in modo immediato le resistenza 1
        erroreRes_1 = 0.1;
        erroreIntegraleRes_1 = 0;  //Azzero tutti i parametri d'errore così da ripartire con la successiva accensione con tutti i parametri a zero come effettivamente è
        erroreDerivatoRes_1 = 0;
        primociclo_Res_1 = 1;
        tempoPrecedente_1 = 0;
        correzioneRes_1 = 0;
        CalcolaPWMRes_1();      // Richiama la funzione che calcola il duty cicle della PWM
        ImpostaPWM_Risc(resistenza_1);    // Richiama la funzione che configura la PWM

        tensioneCtrlCalcolataRes_2 = 4.2;  //spengo in modo immediato le resistenza 2
        erroreRes_2 = 0.1;
        primociclo_Res_2 = 1;
        erroreIntegraleRes_2 = 0;  //Azzero tutti i parametri d'errore così da ripartire con la          successiva accensione con tutti i parametri a zero come effettivamente è
        erroreDerivatoRes_2 = 0;
```

```
        tempoPrecedente_2 = 0;
        correzioneRes_2 = 0;
        CalcolaPWMRes_2();      // Richiama la funzione che calcola il duty cicle della
PWM
        ImpostaPWM_Risc(resistenza_2);    // Richiama la funzione che configura la PWM

        tensioneCtrlVent = 4.2;       // Imposto il valore dell'uscita PWM in base alle
caratteristiche di accensione del MOSFET

      }


    }

 ComandoVentola();            // Attivo il comando della ventola
 ControlloResistenze();       // Attivo il controllo delle resistenze di riscaldamento
  }
```

Figura 48: Ciclo d'isteresi implementato nel FW di Arduino

Il ciclo di isteresi appena descritto è stato implementato all'interno di loop() così da verificare ciclicamente il valore della temperatura ed operare conseguentemente.

4.2.2 Controllo delle correnti di alimentazione dei resistori

Il controllo delle correnti di alimentazione delle resistenze è stato realizzato seguendo la stessa filosofia adottata per l'alimentazione dei LED descritta in precedenza. Dunque è stato implementato un controllo in catena chiusa, utilizando un controllore di tipo PID.

Dovendo poi fisicamente controllare due driver differenti per alimentare i resistori, sono state create due differenti function, ControlloResistenza_1() e ControlloResistenza_2(); queste realizzano un controllo PID sulla corrente inviata ai resistori. Le due function per comodità vengono inglobate all'interno di un'unica function chiamata ControlloResistenze(). Quando viene richiamata, questa function come prima cosa calcola il tempo che intercorre tra una chiamata e la successiva; questo servirà a calcolare il contributo derivativo dell'errore da parte controllore PID.

Prima di misurare la corrente circolante, viene memorizzata in un'altra variabile la misura fatta nel ciclo precedente. Successivamente alla misura della corrente circolante (misurata come tensione alla resistenza di shunt al rispettivo ingresso analogico, vd. paragrafo 4.3.1.3 Misura delle correnti ai riscaldatori), si calcolano i contributi di errore:

- errore proporzionale: differenza tra la rilevazione di corrente al ciclo attuale e quella del ciclo precedente.
- errore integrale: somma tra l'errore proporzionale del ciclo attuale e l'errore proporzionale del ciclo precedente.
- errore derivato: pari a zero nel primo ciclo, mentre nei successivi sarà calcolato come velocità di variazione della corrente misurata tra ciclo attuale e precedente (rapporto tra la variazione della corrente e il tempo impiegato per compiere un intero ciclo).

Una volta pesati i singoli contributi di errore con le relative costanti moltiplicative del controllore PID, viene sommato il contributo totale di errore con il setpoint di corrente impostato. Questo nuovo livello di corrente consente di compensare, di volta in volta, l'errore tra setpoint impostato e corrente realmente circolante, per arrivare ad ottenere l'effettiva corrente che abbiamo scelto. In base alla corrente che si vuole ottenere, viene calcolata la tensione da imporre al morsetto ctrl del driver (attraverso la funzione descrittiva del tratto lineare tensione ctrl-corrrente, vista nel paragrafo "2.1.3 Circuito di alimentazione").
La tensione di ctrl dei driver è stata inoltre limitata al campo di valori per cui esiste una relazione diretta tra tensione ctrl e corrente inviata ai morsetti d'uscita; tale intervallo è 1,65-4,2 V. Il primo valore sarà quindi la tensione al morsetto ctrl che consente di ottenere una corrente in uscita pari a 0,7 A; mentre il secondo sarà il valore per cui la corrente in uscita è pari a 0 A.

Il valore di tensione di ctrl ottenuto viene convertito in base alla risoluzione dell'ADC di Arduino (10 bit: 2^{10}-1=1023); a tale valore è associato il valore di tensione al morsetto di ctrl per il quale si ottiene la corrente nulla erogata dagli stessi (4,2 V secondo quanto detto in precedenza). La funzione CalcolaPWMRes_1() e CalcolaPWMRes_2() consentono di ricavare il valore, pesato con la risoluzione dell'ADC, attraverso una proporzione:

$$Vctrl * \frac{1023}{4{,}2}$$

Una volta fatto questo il valore di tensione verrà impostato al rispettivo morsetto PWM di uscita in base al dispositivo che si vuole andare a controllare. Questo viene realizzato attraverso la function ImpostaPWM_Risc(int dispositivo).

```
void ControlloResistenze() // Funzione che raggruppa le funzioni che implementano un controllore PID differente per il controllo delle due resistenze
{
  tempoPrecedente_1 = tempo_1;    // Memorizza tempo precedente
  tempo_1 = millis() * 0.001;    // La funzione millis() fornisce un tempo in millisecondi, moltiplicati per 0.001 danno un tempo in secondi.
  dt_1 = tempo_1 - tempoPrecedente_1;    // Calcolo il dt trascorso tra due cicli
  ControlloResistenza_1();

  tempoPrecedente_2 = tempo_2;    // Memorizza tempo precedente
  tempo_2 = millis() * 0.001;    // La funzione millis() fornisce un tempo in millisecondi, moltiplicati per 0.001 danno un tempo in secondi.
  dt_2 = tempo_2 - tempoPrecedente_2;    // Calcolo il dt trascorso tra due cicli
  ControlloResistenza_2();
}

void ControlloResistenza_1() // Funzione che implementa un controllore PID per la resistenza 1
{

  correnteMisurataVecchiaRes_1 = correnteMisurataRes_1; // Salva il valore precedente di corrente misurata alla resistenza 1.


  MisuraCorrente_Risc(resistenza_1);    // Richiama la funzione che si occupa di misurare la corrente e salva i dati nella variabile globale correnteMisurataRes1.

  erroreRes_1 = setPointRes - correnteMisurataRes_1;    // Calcola l'errore come differenza tra il set point di corrente e la corrente misurata.

  erroreIntegraleRes_1 += erroreRes_1;        // calcola l'errore integrale come somma dell'errore.
```

```
  erroreDerivatoRes_1 = (correnteMisurataRes_1 - correnteMisurataVecchiaRes_1) / dt_1;
// calcola l'errore derivato

 if (primociclo_Res_1 == 1)    // condizione if. se è il primo ciclo imposta l'errore derivato a
0
 {
  erroreDerivatoRes_1 = 0;
  primociclo_Res_1 = 0;
 }

 correzioneRes_1 = erroreRes_1 * kp_Res_1 + erroreIntegraleRes_1 * ki_Res_1 +
erroreDerivatoRes_1 * kd_Res_1;    // Calcola il valore della correzione.

 correzioneRes_1 += setPointRes;     // Somma al set point di corrente delle resistenze il
valore della correzione da apportare per la resistenza 1.

 tensioneCtrlCalcolataRes_1 = mDriver * correzioneRes_1 + qDriver;      // Calcola la
tensione da applicare al pin CTRL utilizzando la caratteristica fornita dai datasheet

 if (tensioneCtrlCalcolataRes_1 < 0)    // Blocca la caratteristica. per valori inferiori a 1.65
e maggiori a 4.2 viene impostato un valore definito.
 {
  tensioneCtrlCalcolataRes_1 = 1.65;
 }
 if (tensioneCtrlCalcolataRes_1 > 4.2)
 {
  tensioneCtrlCalcolataRes_1 = 4.2;
 }

 CalcolaPWMRes_1();    // Richiama la funzione che calcola il duty cicle della PWM
 ImpostaPWM_Risc(resistenza_1);    // Richiama la funzione che configura la PWM
}
```

```
void ControlloResistenza_2() // Funzione che implementa un controllore PID per la resistenza 2
{

 correnteMisurataVecchiaRes_2 = correnteMisurataRes_2; // Salva il valore precedente di corrente misurata alla resistenza 2.

 MisuraCorrente_Risc(resistenza_2);    // Richiama la funzione che si occupa di misurare la corrente e salva i dati nella variabile globale correnteMisurataRes2.

 erroreRes_2 = setPointRes - correnteMisurataRes_2;    // Calcola l'errore come differenza tra il set point di corrente e la corrente misurata.

 erroreIntegraleRes_2 += erroreRes_2;        // calcola l'errore integrale come somma dell'errore.

 erroreDerivatoRes_2 = (correnteMisurataRes_2 - correnteMisurataVecchiaRes_2) / dt_2; // calcola l'errore derivato

 if (primociclo_Res_2 == 1)    // condizione if. se è il primo ciclo imposta l'errore derivato a 0
 {
  erroreDerivatoRes_2 = 0;
  primociclo_Res_2 = 0;
 }

 correzioneRes_2 = erroreRes_2 * kp_Res_2 + erroreIntegraleRes_2 * ki_Res_2 + erroreDerivatoRes_2 * kd_Res_2;    // Calcola il valore della correzione.

 correzioneRes_2 += setPointRes;      // Somma al set point di corrente delle resistenze il valore della correzione da apportare per la resistenza 2.

 tensioneCtrlCalcolataRes_2 = mDriver * correzioneRes_2 + qDriver;       // Calcola la tensione da applicare al pin CTRL utilizzando la caratteristica fornita dai datasheet
```

```
 if (tensioneCtrlCalcolataRes_2 < 0)    // Blocca la caratteristica. per valori inferiori a 1.65 e maggiori a 4.2 viene impostato un valore definito.
 {
  tensioneCtrlCalcolataRes_2 = 1.65;
 }
 if (tensioneCtrlCalcolataRes_2 > 4.2)
 {
  tensioneCtrlCalcolataRes_2 = 4.2;
 }

 CalcolaPWMRes_2();    // Richiama la funzione che calcola il duty cicle della PWM
 ImpostaPWM_Risc(resistenza_2);    // Richiama la funzione che configura la PWM
}

void CalcolaPWMRes_1()    // Funzione che calcola il duty cicle della PWM alla resistenza 1.
{
 float k = 243.5714286;      // Per evitare le divisioni questo valore è stato calcolato e riportato come costante. è uguale alla risoluzione dell'ADC (1023) diviso il valore massimo di tensione applicabile al pin CTRL (4.2)

 PWM_Res_1 = tensioneCtrlCalcolataRes_1 * k;    // Memorizza un numero che va da 0 a 1023

}

void CalcolaPWMRes_2()    // Funzione che calcola il duty cicle della PWM alla resistenza 2.
{
 float k = 243.5714286;      // Per evitare le divisioni questo valore è stato calcolato e riportato come costante. è uguale alla risoluzione dell'ADC (1023) diviso il valore massimo di tensione applicabile al pin CTRL (4.2)

 PWM_Res_2 = tensioneCtrlCalcolataRes_2 * k;    // Memorizza un numero che va da 0 a 1023
```

```
}

void ImpostaPWM_Risc(int dispositivo)

{

  if (dispositivo == resistenza_1)
 {

  analogWrite(pinRes_1,PWM_Res_1);
 }
 if (dispositivo == resistenza_2)
 {

  analogWrite(pinRes_2,PWM_Res_2);
 }

 if (dispositivo == ventola)
 {

  analogWrite(pinVent,PWM_Vent);
}

}
```

Figura 49: function che gestiscono controllo di corrente ai resistori

4.2.3 Comando di accensione e spegnimento della ventola

Per ciò che riguarda la ventola di raffreddamento il discorso è più semplice volendo di fatto comandare la ventola in modalità ON-OFF.
Per permettere ciò, come vedremo al paragrafo "4.3.3 Mosfet IRF530N e alimentazione della ventola", è stato installato un MOSFET. Il funzionamento in ON-OFF tra DRAIN e SOURCE, viene garantito imponendo la tensione GATE-SOURCE rispettivamente ad un valore pari a 4,2 V (scelto valutando dai datasheet come si comporta il MOSFET variando la tensione Vgs) oppure a zero.

Per ognuno dei casi comunque il valore di tensione che vogliamo impostare alla uscita PWM corrispondente dev'essere convertita in base alla risoluzione di Arduino (10 bit: 2^{10}-1=1023). Al valore 1023 (massima risoluzione) è associato il valore di tensione al morsetto di GATE del MOSFET, per il quale il MOSFET stesso presenta una resistenza tra drain e source bassa (4,2 V secondo quanto detto in precedenza). La funzione CalcolaPWMVent() consente di ricavare il valore di tensione, pesato con la risoluzione dell'ADC, attraverso una proporzione:

$$Vctrl * \frac{1023}{4{,}2}$$

void CalcolaPWMVent() *// Funzione che calcola il duty cicle della PWM alla ventola (anche se noi in realtà la ventola la utilizziamo in ON-OFF questa funzione può essere utile nel momento in cui si decida di effettuare una modulazione del comando al gate del mosfet).*

```
{
 float k = 243.5714286;      // Per evitare le divisioni questo valore è stato calcolato e riportato come costante. è uguale alla risoluzione dell'ADC (1023) diviso il valore massimo di tensione applicabile al pin CTRL (4.2)

  PWM_Vent = tensioneCtrlVent * k;    // Memorizza un numero che va da 0 a 1023

}
```

Figura 50: Calcolo della PWM da impostare all'uscita relativa al comando della ventola

Una volta fatto questo il valore di tensione verrà impostato al rispettivo morsetto PWM di uscita in base al dispositivo che si vuole andare a controllare. Questo viene realizzato attraverso la function ImpostaPWM_Risc(int dispositivo).

void ImpostaPWM_Risc*(int dispositivo)*

```
{

   if (dispositivo == resistenza_1)
  {

   analogWrite(pinRes_1,PWM_Res_1);
```

```
 }
 if (dispositivo == resistenza_2)
 {

   analogWrite(pinRes_2,PWM_Res_2);
 }

if (dispositivo == ventola)
 {

   analogWrite(pinVent,PWM_Vent);
 }

}
```

Figura 51: Impostazione del comando PWM alla relativa uscita

4.2.4 Comando relativo alla misura di temperatura

Come abbiamo visto in precedenza, al comando 'mte' Arduino risponde mettendo a disposizione nel buffer d'uscita il dato relativo alla misura di temperatura alla piastra.
Nel firmware precedentemente implementato, a questo comando corrispondeva ogni volta una misura di temperatura richiamando la function MisuraTemperatura().
Come abbiamo visto nel paragrafo "4.2.1 Introduzione del ciclo d'isteresi", la misura della temperatura viene operata ad ogni ciclo; dunque è inutile ogni volta operare una nuova misura se il dato relativo alla misura di temperatura è già disponibile. È per questo che il firmware è stato modificato affinchè al comando 'mte' non corrisponda più una misura reale, ma semplicemente viene messo a disposizione nel buffer d'uscita il valore già misurato in precedenza. Questo consente di snellire notevolmente il comando di misura di temperatura.

4.2.5 Misura di corrente ai resistori

Rispetto al firmware precedentemente implementato, è stato necessario introdurre anche una function che consenta la misura delle correnti ai resistori.
La function introdotta è void MisuraCorrente_Risc(int dispositivo) e consente di fatto, in base al resistore sul quale vogliamo fare la misura di corrente, di rilevare la tensione alla rispettiva resistenza di shunt. Questa viene misurata attraverso il rispettivo ingresso analogico. Viene fatta una media su 100 misure di tensione, la quale viene convertita,

attraverso l'ADC a 10 bit presente all'interno di Arduino, in un numero da 0 a 1023 in base al valore di tensione presente all'ingresso analogico. La conversione avviene sapendo che al valore di tensione di riferimento (impostato a 1,069 V, che è il valore reale della tensione di riferimento legata ad un'imposizione a 1,1 V) scelto per l'ADC, corrisponde la massima risoluzione dell'ADC stesso (10 bit: 2^{10}-1=1023). L'operazione di conversione sarà quindi una semplice proporzione; infatti se alla tensione di riferimento corrisponde un valore di 1023, ad una tensione generica d'ingresso Valim, corrisponde un valore:

$$\frac{1023}{1{,}069} * Valim$$

questo sarà il valore che assume la tensione allo shunt convertita in base alla risoluzione che ha l'ADC di Arduino e che, divisa per la corrispondente resistenza di shunt, fornisce la corrente circolante.

void MisuraCorrente_Risc(int dispositivo)

```
{

  // Variabili locali
  int letture = 100;                // Numero di letture di corrente da effettuare
  float risoluzione = 0.0009775171;   // Reciproco della risoluzione dell'ADC. Viene dato il reciproco per evitare di fare divisioni. Impegnative per il microcontrollore.
  int pin = 0;                      // Memorizza il pin di input dell'ADC da leggere.

    if (dispositivo == resistenza_1)
  {
    pin = correnteRes1;

    for (int i = 0; i < letture; i++)    // Ciclo for che interroga il pin dell'ADC selezionato
    {
      correnteMisurataRes_1 += analogRead(pin);
    }

    correnteMisurataRes_1 /= letture;    // Viene memorizzata la tensione media
    correnteMisurataRes_1 *= risoluzione * VREF;     // Viene memorizzata la tensione convertita in Volt
    correnteMisurataRes_1 /= rRes_1;      // Viene calcolata la corrente. Il valore della resistenza è definito all'inzio del programma.
  }

  if (dispositivo == resistenza_2)
  {
    pin = correnteRes2;
```

```
    for (int i = 0; i < letture; i++)    // Ciclo for che interroga il pin dell'ADC selezionato
   {
    correnteMisurataRes_2 += analogRead(pin);
   }

   correnteMisurataRes_2 /= letture;    // Viene memorizzata la tensione media
   correnteMisurataRes_2 *= risoluzione * VREF;     // Viene memorizzata la tensione convertita in
Volt
   correnteMisurataRes_2 /= rRes_2;      // Viene calcolata la corrente. Il valore della resistenza è
definito all'inzio del programma
  }

}
```

Figura 52: Misura delle correnti ai resistori

Sempre in materia di misura di corrente, sono stati introdotti due nuovi comandi che consentono di misurare la corrente inviata ai resistori. Questi comandi sono ‘mcs’ e ‘mci’, che consentono rispettivamente di ottenere una stampa a video del valore assunto dalla corrente ai due resistori.

4.2.6 Problemi sorti nella realizzazione del FW di Arduino e verifica del funzionamento dello stesso

Uno dei primi problemi è stata la discrepanza tra i valori di temperatura misurati mediante il comando 'mte' da Teraterm e l'effettiva temperatura ambiente. Per poter trovare una soluzione a tale problema è stato imposto, mediante l'ausilio di un generatore di tensione continua, un valore noto di tensione (1 V) all'ingresso analogico di misura della temperatura per verificare che l'indicazione fornita dal Teraterm fosse coerente. Sapendo che Arduino AT Mega 2560 contiene al suo interno un convertitore ADC a 10 bit ($0 \div 2^{10}$-1= $0 \div 1023$ cifre rappresentabili), imponendo come tensione di riferimento interna per l'ADC di 1,1 V, in risposta ad una tensione all'ingresso analogico di 1 V dovremmo ottenere un numero pari a:

$$\frac{1023}{1{,}1} * 1 = 930$$

ciò che invece otteniamo a video è un valore pari a 966 (e quindi un errore circa pari al 3%). Dopo aver operato più prove imponendo all'ingresso analogico diversi livelli di tensione, si è potuto constatare come l'errore che si otteneva fosse tanto più piccolo quanto più si riduceva il valore della tensione all'ingresso analogico e tendeva inoltre ad annullarsi quando

andavamo ad imporre un valore di tensione pari ad 1,1 V (fondoscala). Questi risultati hanno portato ad ipotizzare che:

- la diversa entità dell'errore al variare della tensione impostata all'ingresso analogico, sia dovuta al fatto che alla tensione misurata si faccia corrispondere un valore, pesato con la risoluzione dell'ADC e la tensione di riferimento dello stesso, sovrastimato rispetto a quello che realmente dovrebbe essere. Tale differenza sarà tanto più marcata quanto più alto è il livello di tensione. Questo potrebbe essere dovuto ad una sovrastima della reale tensione di riferimento dell'ADC.
- dall'altra il fatto di ottenere un valore esatto per il valore di fondoscala 1,1 V è legato probabilmente ad una saturazione del convertitore ADC, per valori prossimi al fondoscala.

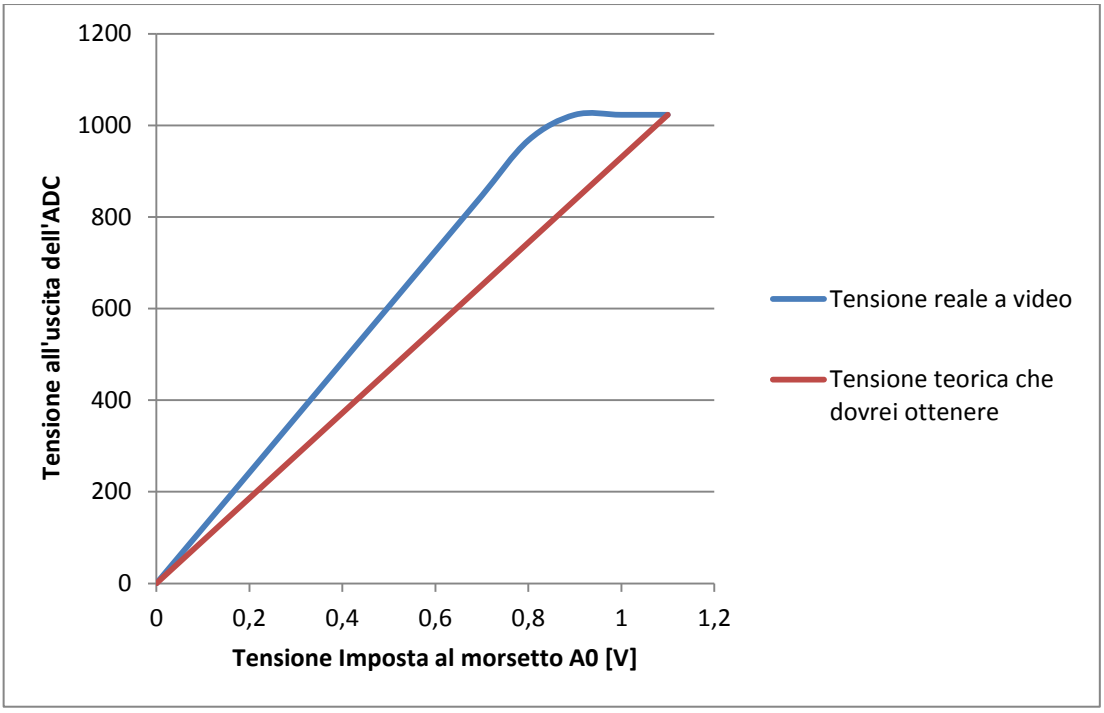


Figura 53: Confronto tra la tensione che ottengo a video e quella reale prima della variazione della tensione di riferimento dell'ADC

Operando la misura della tensione di riferimento del convertitore ADC reale (morsetto AREF di Arduino), si è potuto notare come il valore reale (1,069 V) discosti rispetto a quello impostato all'interno del firmware (1,1 V) del 3%; questo ci permette, in prima battuta, di affermare che scostamento del 3% che ottenavamo precedentemente, possa essere riconducibile ad una sovrastima della tensione di riferimento dell'ADC nelc firmware di Arduino per il calcolo delle tensioni misurate agli ingressi analogici.

In effetti riducendo tale valore, lo scostamento precedentemente rilevato non è più presente.

È stato inoltre verificato che il Firmware modificato, secondo le nostre necessità, fosse idoneo.
In primo luogo si è verificato il funzionamento del controllo delle correnti ai resistori; con l'ausilio di un generatore di tensione continua, sono state simulate le tensione agli shunt relativi alle correnti circolanti ai resistori. Queste tensioni vengono misurate dagli ingressi analogici corrispondenti alla misura delle correnti ai resistori. Una volta impostato un setpoint di corrente che vogliamo ottenere ai resistori, si è verificato che:

- se la tensione all'ingresso analogico relativo alla misura di corrente, corrisponde ad una corrente circolante inferiore al setpoint impostato, la tensione all'uscita PWM tende a crescere.
- se la tensione all'ingresso analogico relativo alla misura di corrente, corrisponde ad una corrente circolante maggiore al setpoint impostato, la tensione all'uscita PWM tende a ridursi.
- se la tensione all'ingresso analogico relativo alla misura di corrente, corrisponde ad una corrente circolante che ha un valore prossimo al setpoint impostato, la tensione all'uscita PWM tende a stabilizzarsi.

Nonostante le dinamiche non siano soddisfacenti e verranno solo in seconda battuta regolate attraverso la variazione delle costanti moltiplicative dei diversi contributi di errore, il funzionamento del controllo è soddisfacente.

In seconda battuta era necessario verificare se, al variare della temperatura della piastra (variando la tensione all'ingresso analogico corrispondente, attraverso un generatore di tensione continua), corrispondesse l'accensione e lo spegnimento dei resistori e della ventola, secondo il ciclo d'isteresi implementato.

Per calcolare la tensione all'ingresso analogico che corrisponde alla temperatura di 29°C sappiamo che:

- la corrente prodotta dal sensore di temperatura circola in una resistenza di shunt di 2,2 kΩ.
- il segnale di corrente proveniente dal sensore di temperatura è pari a 1 μA/K.

Dunque la tensione necessaria a simulare una temperatura di 30°C all'ingresso analogico è pari a:

$$1\left[\frac{\mathrm{A}}{\mathrm{K}}\right] * (273{,}15 + 30)[\mathrm{K}] * 2{,}2\ [\mathrm{k\Omega}] = 0.6669\ \mathrm{V}$$

mentre per simulare la soglia inferiore di 28°C:

$$1\left[\frac{\mathrm{A}}{\mathrm{K}}\right] * (273{,}15 + 28)[\mathrm{K}] * 2{,}2\ [\mathrm{k\Omega}] = 0.6625\ \mathrm{V}$$

Attraverso l'ausilio di un multimetro è possibile verificare che al di sopra della prima soglia, abbiamo l'accensione della ventola (tensione al pin dell'uscita PWM 4, relativa al comando della ventola, pari a 4,2 V) e la tensione di CTRL dei driver di alimentazione delle resistenze anch'essa pari a 4,2 V (che significa correnti nulle inviate dai driver alle resistenze).

Al di sotto della seconda soglia abbiamo lo spegnimento della ventola (tensione al pin dell'uscita PWM 4, relativa al comando della ventola, pari a 0 V) e la tensione di CTRL dei driver di alimentazione delle resistenze pari al valore relativo al setpoint di corrente impostato.
Tra le due soglie di tensione (corrispondenti alle due soglie dell'isteresi) invece non viene operata nessuna variazione; se, partendo da una tensione minore rispetto alla soglia inferiore, la tensione all'ingresso analogico relativo alla misura di tempertura sta crescendo, viene rilevato che:

- le tensioni di uscita PWM che comandano i driver di alimentazione dei resistori si mantengono pari al valore relativo al setpoint di corrente impostato (resistori accesi);
- il segnale PWM che comanda il MOSFET è pari a zero (ventola spenta).

Al contrario, se parto da una tensione maggiore rispetto al valore relativo alla soglia superiore e riduco la tensione:

- le tensioni di uscita PWM che comandano i driver di alimentazione dei resistori si mantengono pari al valore relativo al setpoint nullo di corrente (4,2 V, resistori spenti);
- il segnale PWM che comanda il MOSFET è pari a 4,2 V (ventola accesa).

Ovviamente non avendo inserito le costanti moltiplicative del controllore PID più idonee al nostro caso, la tensione in uscita dai PWM relative alle resistenze ha una dinamica sicuramente non soddisfacente, come già precedentemente esplicitato.

4.3 Sistema di stabilizzazione della temperatura

4.3.1 Scheda di controllo ed alimentazione

Per operare il controllo della temperatura alla piastra con la metodologia precedentemente descritta, è stata creata una scheda supplementare. Prima di tutto sono stati connessi i pin di Arduino di nostro interesse:

- Gli ingressi analogici per la misura delle correnti inviate ai resistori (A7 E A8).
- Le uscite PWM per il comando ctrl dei driver d'alimentazione dei resistori e per il comando del GATE del MOSFET per l'accensione e lo spegnimento della ventola (6,7 e 4 rispettivamente).

Una volta fatto ciò, la scheda è stata sovrapposta a quella già esistente. Il principio di funzionamento di tale scheda è riconducibile al funzionamento della scheda che è stata creata per l'alimentazione dei LED.

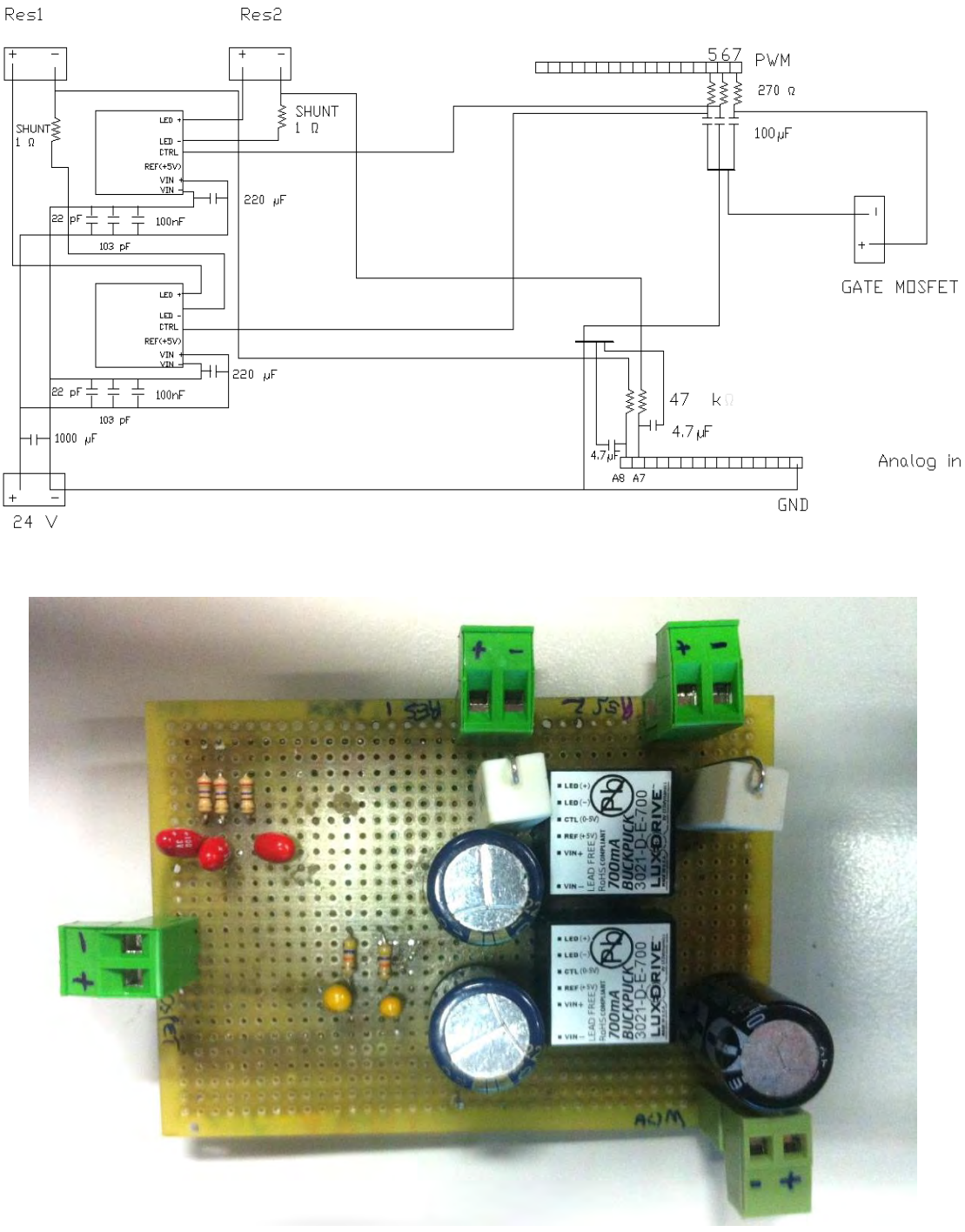


Figura 54: Schema elettrico e foto della scheda di controllo della temperatura

4.3.1.1 Alimentazione e controllo dei driver di alimentazione dei resistori

L'alimentazione delle resistenze avviene, come per i LED, attraverso i driver LUXDRIVE 3021 D-E-700 BuckPuck (vd. Paragrafo “2.1.3 Circuito d’alimentazione”) la cui tensione

d'alimentazione sarà posta a 24 V. Il segnale prodotto attraverso le uscite PWM di Arduino (dai canali 4 e 6), viene filtrato dalle componenti ad alta frequenza attraverso un filtro RC del primo ordine. Questo filtro è composto da una resitenza da 270 Ω e un condensatore al tantallio da 100μF, che porta ad avere una pulsazione di taglio ft pari a:

$$ft = \frac{1}{2 * \pi * R * C} = \frac{1}{2 * \pi * 270 * 100 * 10^{-6}} \sim 6\ Hz$$

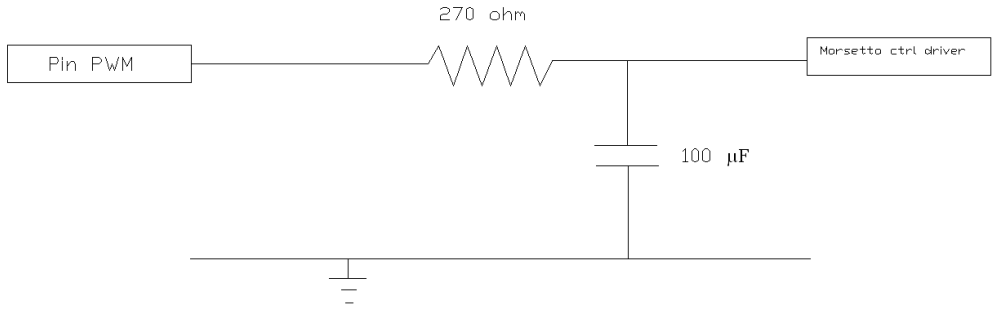


Figura 55: Alimentazione pin CTRL dei driver dei riscaldatori

4.3.1.2 Alimentazione del morsetto Gate del MOSFET

Anche in questo caso viene utilizzato il segnale PWM (dal canale 7), il quale viene filtrato sempre attraverso un filtro RC del primo ordine composto da una resitenza da 270 Ω e un condensatore al tantallio da 100μF.

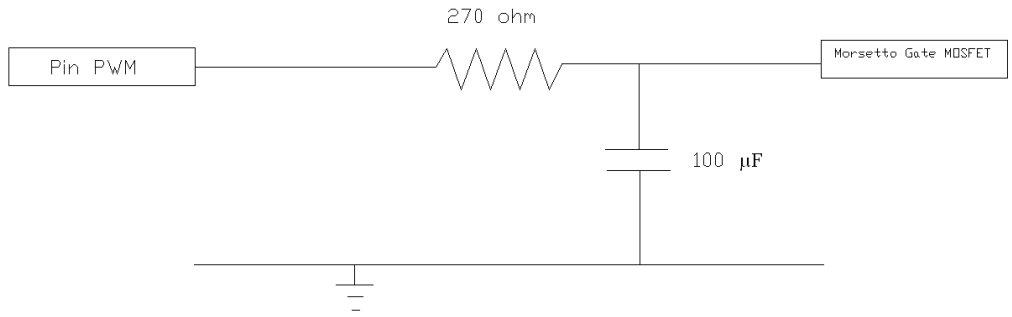


Figura 56: Schema di collegamento per l'alimentazione del morsetto GATE del MOSFET

4.3.1.3 Misura delle correnti ai riscaldatori

Per poter operare un controllo in catena chiusa delle correnti ai riscaldatori, abbiamo bisogno di misurare la corrente circolante sugli stessi.

Per fare ciò sono state installate due resistenze di shunt da 1 sulle quali circolano le correnti d'alimentazione ai resistori per il riscaldamento della piastra. Il valore di tensione ai capi di tali resistenze di shunt, vengono filtrate delle componenti ad alta frequenza attraverso un filtro RC del primo ordine. Questo è composto da una resistenza pari a 470 k e capacità al tantallio da 4,7 μF. Il filtro presenta una frequenza di taglio ft pari a:

$$ft = \frac{1}{2 * \pi * R * C} = \frac{1}{2 * \pi * 47 * 10^{3} * 4{,}7 * 10^{-6}} \sim 10\ Hz$$

Realizzato il filtraggio, si opera la connessione al rispettivo ingresso analogico di Arduino (A7 e A8).

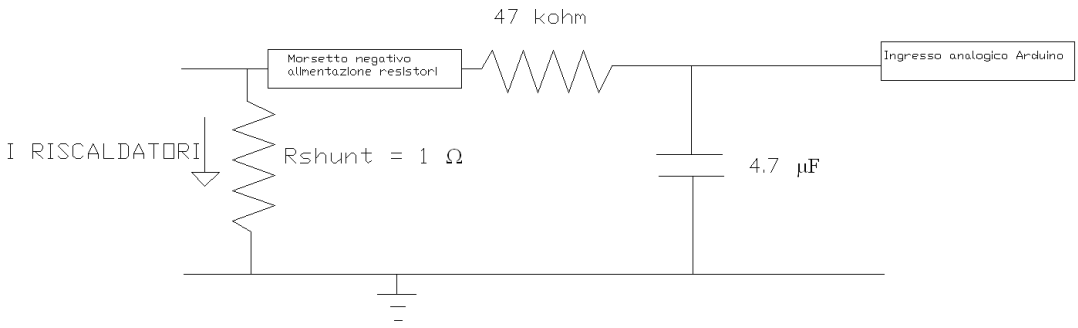


Figura 57: Misura delle correnti inviate ai resistori

La scelta della resistenza di shunt non è casuale; questo valore consente infatti di avere, con le correnti in gioco, una tensione che sia sempre inferiore alla tensione di riferimento dell'ADC di Arduino.

4.3.2 Riscaldatori e loro installazione

La scelta della tipologia di resistori doveva adattarsi al sistema già esistente.

Tra le diverse possibilità, la scelta è ricaduta su resistori di potenza MHP20-500 J.

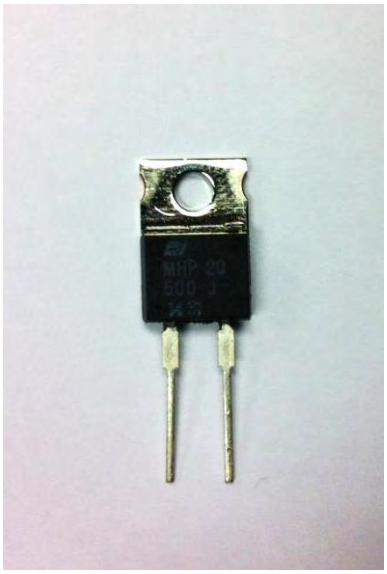


Figura 58: Resistore MHP20-500J

I motivi alla base di questa scelta sono:

- Le loro ridotte dimensioni che consentono di avere molta libertà di installazione.
- Il foro posto all'estremità superiore della flangia di dispersione del resistore, può essere utilizzato per inserire delle viti utili al fissaggio dei resistori alla piastra metallica. La piastra metallica è dotata di un radiatore aggiuntivo, fissato attraverso due viti filettate alla piastra stessa. Dilatando leggermente i due fori presenti sulla flangia metallica dei resistori, è possibile installare i resistori attraverso le viti di fissaggio del radiatore. Questo consente di non operare nuovi lavori di foratura.

Dai datasheet è possibile inanzitutto vedere come la dissipazione della potenza avviene maggiormente nella piastra metallica posteriore (che metteremo quindi a stretto contatto con il radiatore aggiuntivo) e come la temperatura massima che potremmo raggiungere alla piastra metallica d'installazione dei LED, dissipando la potenza a noi necessaria sia circa pari a 90°C, ben al di sopra delle nostre necessità.

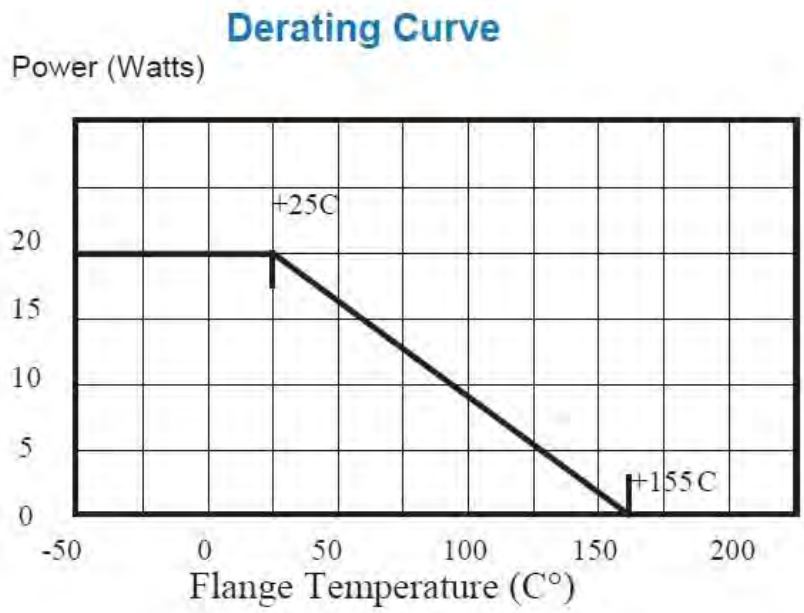


Figura 59: Potenza dissipata dalla flangia posteriore della resistenza in funzione della temperatura della stessa

Ovviamente avendo una potenza massima dissipabile ai resistori di 20 W (con temperatura alla piastra fino a 25°C) è facile intuire che la corrente massima circolante debba essere pari a:

$$I = \sqrt{\frac{P}{R}} = \sqrt{\frac{20}{50}} \sim 0{,}63\ A$$

limite che è stato impostato attraverso il firmware. In realtà nel nostro caso questa rappresenta una semplice precauzione; infatti, alimentando i driver a 24 V, potremmo arrivare ad una corrente massima erogabile dai driver stessi che è pari a:

$$I = \frac{V}{R} = \frac{24}{50} = 0{,}48\ A$$

Questo vorrebbe dire avere una potenza dissipata pari a:

$$P = R * I^2 = 50 * 0{,}48^2 \sim 11{,}5\ W$$

Questa potenza dissipata può essere sopportata dal resistore nel momento in cui la piastra presenta una temperatura che sia al massimo nell'intorno degli 80 °C. Non raggiungendo mai questa temperatura alla piastra, sono sicuro di garantire ai resistori un funzionamento entro i loro limiti.

Nel momento in cui si decidesse di alzare la tensione d'alimentazione dei driver, la corrente erogabile dagli stessi cresce e potrebbe superare il limite massimo ammissibile per i resistori. Dunque la limitazione che viene impostata attraverso il firmware torna utile.

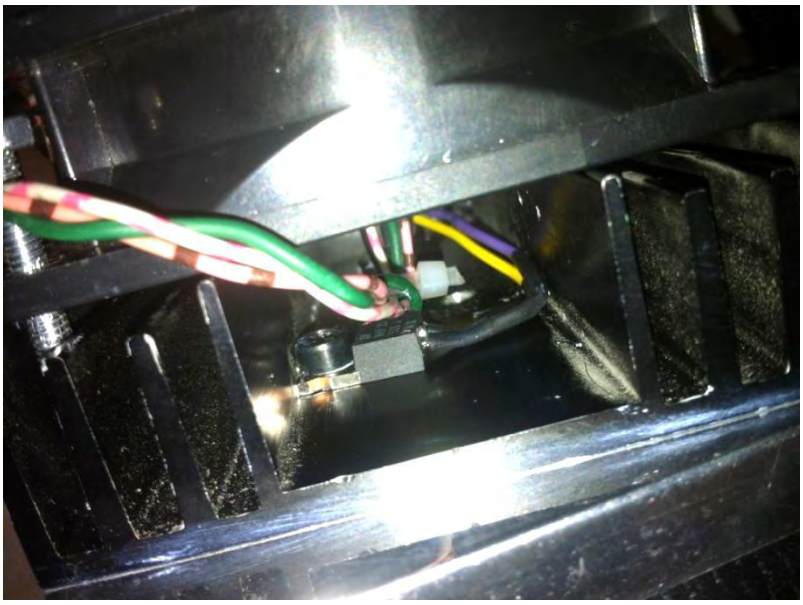


Figura 60: Installazione di una delle due sulla piastra posteriore

4.3.3 Installazione del MOSFET IRF530N ed alimentazione della ventola

Per poter comandare la ventola in modalità ON-OFF quando la temperatura vada al di sopra della soglia prestabilità, c'è bisogno di un interruttore che consenta, attraverso un comando, di poter interrompere o abilitare l'alimentazione della ventola.
Il dispositivo che per caratteristiche risponde alle nostre esigenze è il MOSFET; in realtà questo dispositivo consente non solo un funzionamento come semplice interruttore ma consente la regolazione della corrente circolante tra drain e source, comandando opportunamente il gate. Nel nostro caso però verrà utilizzato come semplice interuttore.
La scelta è ricaduta sul MOSFET IRF530N che, secondo i dati disponibili da datasheet, consentono di far circolare delle correnti molto più grandi di quella assorbita dalla ventola (0,18 A) imponendo una tensione di gate pari a 4,2 V.

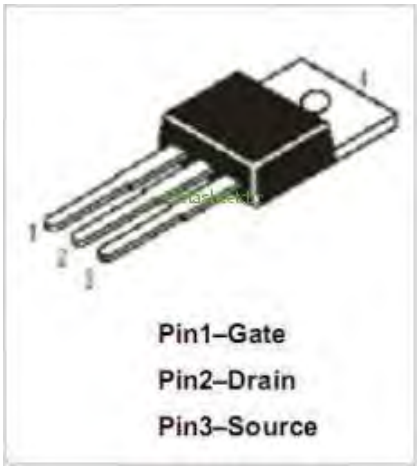


Figura 61: Rappresentazione della piedinatura del MOSFET IRF530N

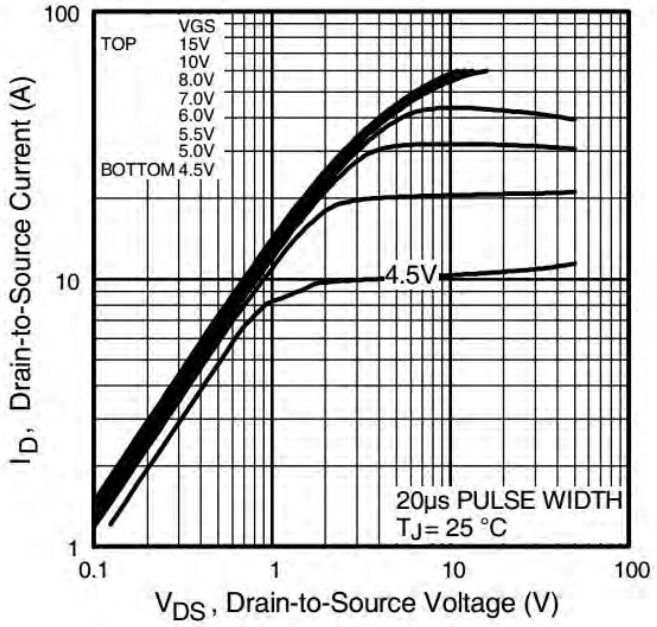


Figura 62: Andamento della corrente Drain-Source in funzione della tensione Drain-Source (curve parametrizzate con la tensione Gate-Source Vgs)

Per poter avere un riscontro effettivo del suo funzionamento, il MOSFET è stato comunque testato; attraverso un generatore di tensione continua sono stati imposti dei valori di tensione gate-source, verificando come variasse la corrente circolante su una resistenza di valore noto, collegata tra l'alimentazione e il morsetto DRAIN del MOSFET. Abbiamo potuto constatare che per tensione nulla, il mosfet si comporta come un circuito aperto, mentre per tensione di gate intorno ai 4,2 V il mosfet si comporta come un cortocircuito.

Attraverso un multimetro (effettuando una misura di resistenza tra flangia e morsetto drain) è stato inoltre possibile verificare che il morsetto drain è direttamente collegato alla flangia posteriore del MOSFET; per evitare quindi che ci sia un cortocircuito a terra attraverso la

piastra metallica di montaggio dei LED (che si trova al potenziale di terra) è necessario quindi, nel fissaggio del MOSFET alla piastra stessa, interporre uno strato di materiale isolante ai fini di evitare questo problema.

Altro punto fondamentale nell'installazione del mosfet è quella di interporre tra gate e source, una resistenza elevata (nel nostro caso è stata scelta da 10kΩ) per consentire alla capacità parassita all'ingresso del gate di scaricarsi quando allo stesso non ho più alcun comando. In questo modo sono sicuro che al comando di tensione nulla al gate, corrisponda l'effettivo spegnimento della ventola.

Cosa da non trascurare, oltre agli aspetti di funzionamento, è il fatto che la sua forma ridotta e il foro posto nella parte superiore della flangia metallica consentono una rapida installazione, attraverso una vite filettata, sfruttando un foro già presente.

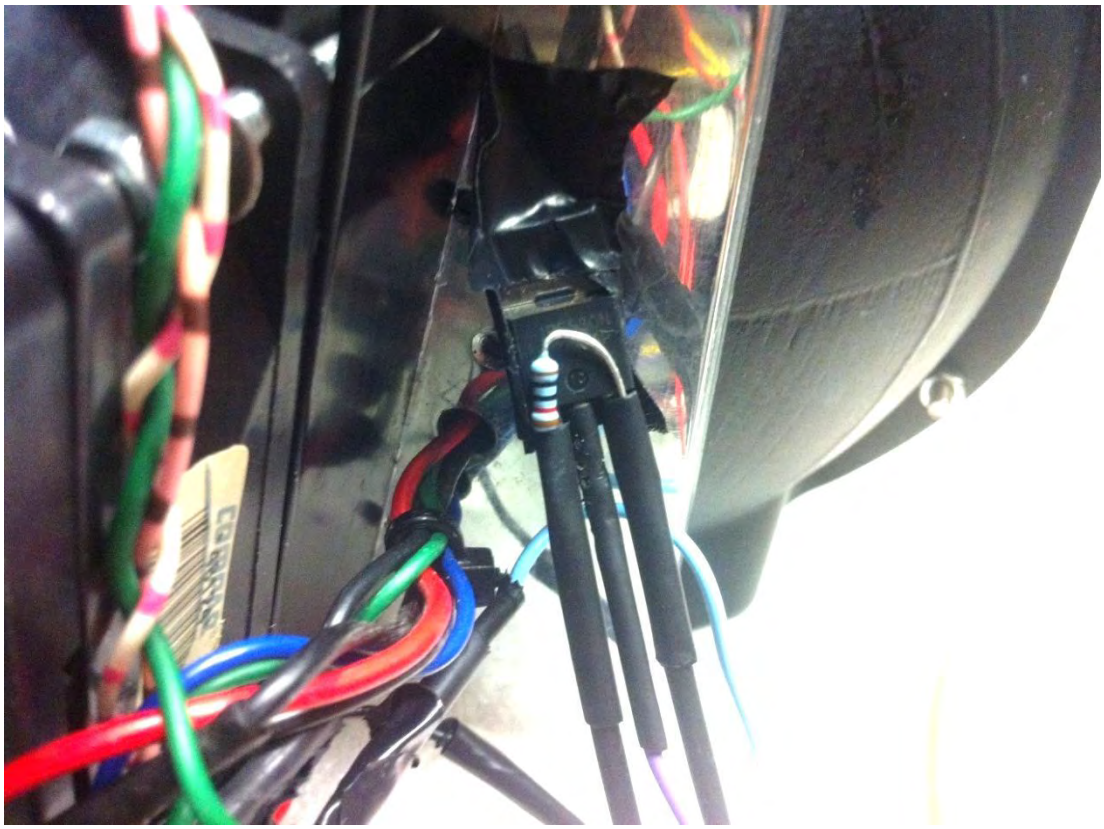


Figura 63: Installazione del MOSFET nella parte posteriore della piastra

La ventola viene connessa tra il morsetto a potenziale 12 V e il drain del mosfet; il source dello stesso viene connesso al riferimento.

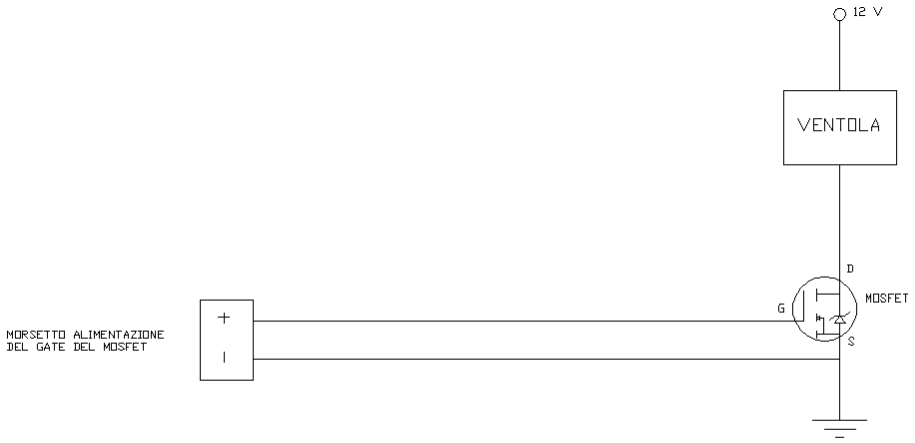


Figura 64: schema di connessione del MOSFET e della ventola

4.3.4 Problemi sorti nella realizzazione del sistema di stabilizzazione della temperatura

Prima di realizzare il sistema resistenze-ventola che consente di lavorare ad una temperatura costante, qualsiasi sia la condizione di funzionamento a cui si portano a funzionare i LED, è stata fatta una verifica dei componenti da utilizzare (in particolare i driver) per verificare il corretto funzionamento degli stessi.

Per fare ciò, utilizzando il generatore di tensione continua presente in laboratorio, si alimentano i driver e i morsetti ctrl degli stessi ad una tensione nota e priva praticamente di disturbi; l'uscita del driver, come sappiamo, dovrebbe essere una corrente imposta che, teoricamente, decresce linearmente da 0,7 a 0 A imponendo rispettivamente una tensione al morsetto CTRL da 4,2 a 1,65 V.

Collegando l'uscita del driver ad una resistenza di valore noto (1 Ω) è possibile verificare la variazione della corrente imposta in uscita dai driver semplicemente misurando, con l'ausilio di un multimetro, la tensione ai capi della stessa.

Quello che non si prevedeva di vedere era un mancato controllo iniziale delle correnti inviate ad un carico resistivo noto (non le resistenze che serviranno per scaldare la piastra). Leggendo dei lavori fatti in precedenza sull'oggetto in esame, il motivo di questa mancata regolazione è da attribuire al fatto che i driver sono stati alimentati inizialmente con una tensione troppo bassa; infatti per garantire il controllo della corrente in uscita, la tensione di alimentazione deve essere di almeno 2 V superiore (funzionamento in DC) rispetto a quella

che ho in uscita. Portando la tensione di alimentazione dei driver ad un valore più elevato il riscontro della prova è stato positivo.

Altro problema che si è verificato è stato nel morsetto di ingresso che ci permette di rilevare la temperatura (A0); infatti, rilevando la temperatura si è verificato che l'indicazione che veniva fornitaa video in risposta al comando mte (485 K) era completamente differente dalla temperatura reale presente. A tal proposito sono state effettuate delle misure verificando che, anche togliendo tutte le schede presenti e facendo delle misure solo sul morsetto A0 di Arduino, veniva rilevata una tensione circa pari a 4,9 V; dallo sketch di Arduino è possibile verificare che dall'indicazione mte è possibile risalire a ritroso per individuare la tensione che ho all'ingresso analogico:

$$\frac{485}{1000} * resistenzaTemp * \frac{1}{risoluzione * VREF} \sim 1000$$

Questa è l'indicazione fornita in uscita dal convertitore ADC di Arduino, da cui è possibile ricavare la tensione relativa a tale indicazione, ricordando che il convertitore è a 10 bit (il fondoscala è 2^{10}-1=1023), come:

$$1000 * \frac{Valim}{1023} = 4{,}88\ V$$

Che corrisponde di fatto al valore che viene rilevato attraverso il multimetro. È stato quindi riscontrato che il problema non è nel firmware implementato ma è da ricondurre all'hardware.

Sapendo che gli ingressi di Arduino sono ad elevata impedenza, una possibilità era sicuramente quella che la tensione in esame non fosse in realtà imposta da Arduino, ma bensì una tensione parassita; per verificare ciò è stata introdotta una resistenza molto elevata (10 k) per scaricare l'eventuale capacità parassita presente (sullo stesso principio del gate del Mosfet, vd. paragrafo "4.3.3 Installazione del MOSFET IRF530N ed alimentazione della ventola") tra il morsetto A0 e il potenziale di massa (GND) verificando la tensione ai capi di questa resistenza. Se fosse presente una tensione parassita, progressivamente si dovrebbe vedere la tensione ai capi della resistenza scendere seguendo il tempo di scarica di tale capacità. Avendo avuto un riscontro negativo da tale prova, è stata seguita un'altra strada cercando di imporre dei valori di tensione all'ingresso analogico A0. Tali valori di tensione, imposti attraverso il generatore di tensione continua presente in laboratorio, sono stati scelti

così da essere proporzionali alle temperature in gioco; anche in questo caso però l'indicazione del morsetto rimaneva invariata.
A questo punto, probabilmente, la tensione presente ai capi dell'ingresso analogico A0 è dovuta ad un danneggiamento della scheda Arduino; tale danneggiamento è dovuto con ogni probabilità al contatto accidentale tra il morsetto A0 stesso e l'alimentazione della ventola (12 V).
Quindi per poter continuare con le misure è stato necessario modificare la scheda di controllo, in modo che la tensione alla resistenza di shunt relativa alla misura di temperatura, non venga più portata all'ingresso analogico A0 ma ad un morsetto differente (la scelta è ricaduta sul primo libero, cioè il morsetto A9).

4.4 Scelta del setpoint di corrente ai resistori

La scelta della corrente da inviare ai resistori è stata fatta cercando ottimizzare l'andamento della temperatura rilevata al sensore; infatti fornendo una corrente troppo elevata ai resistori c'è il rischio che la potenza dissipata dagli stessi sia troppo elevata. Questo provocherebbe, certamente un aumento più rapido della temperatura, con il rischio però di superare il limite superiore del ciclo d'isteresi con un valore troppo elevato; questo significherebbe mantenere la ventola accesa per molto tempo per smaltire il calore accumulato dalla piastra e far tornare la temperatura all'interno dell'isteresi. In realtà quello che si vuole ottenere è un sistema che garantisca la stabilità della temperarura, senza operare però continue accensioni e spegnimenti dei sistemi di riscaldamento e raffreddamento.
La prova da effettuare consiste nel valutare come varia la temperatura della piastra imponendo una corrente ai resistori di 0,3 A (leggermente inferiore rispetto alla massima corrente erogabile dai driver), alimentando tutti i LED alla massima corrente. La situazione implementata rappresenta di fatto la situazione più critica in termini di dissipazione di potenza per effetto Joule.
Questa prova ci consente di valutare se la corrente al resistore che è stata imposta porti ad avere nelle peggiori condizioni di riscaldamento della piastra da parte dei LED, il superamento della soglia superiore di isteresi, con una temperatura troppo elevata. Con il firmware precedentemente implementato, possiamo vedere che tra due campioni successivi c'è una variabilità di 0,5 K. Cosa molto strana dato che comunque la rilevazione della temperatura viene fatta con la massima velocità di campionamento (1 campione ogni 0,063 s e quindi circa 15 campioni al secondo) e la dinamica termica della piastra è molto lenta.

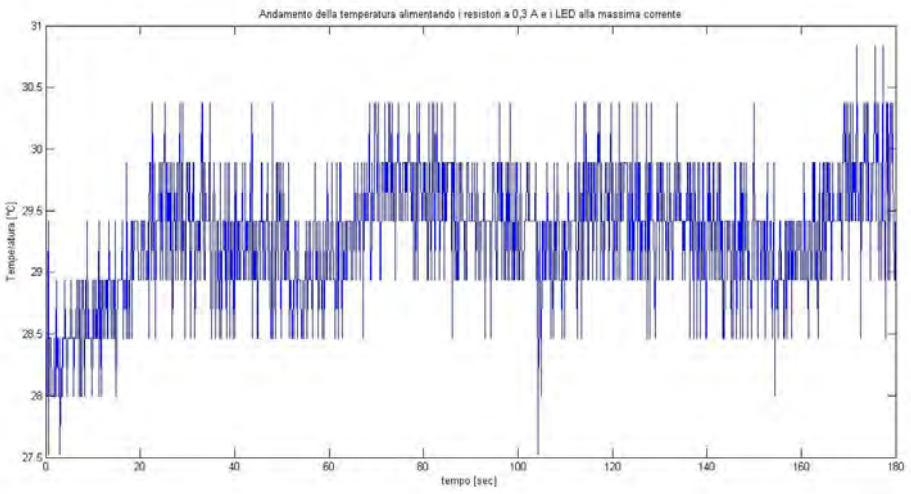


Figura 65: andamento della temperatura alla piastra alimentando i resistori a 0,3 A e i LED alla massima corrente

La variabilità di 0,5 K è giustificata dal fatto che con il convertitore ADC a 10 bit installato e la tensione di riferimento a 1,069 V, la sensibilità di Arduino in termini di misura di tensione all'ingresso analogico è pari a:

$$\frac{1.069}{1023} = 0.00104\ V$$

A questo valore di tensione corrisponde una corrente circolante alla resistenza di shunt per la misura di temperatura pari a:

$$\frac{0.0011}{2.2 * 10^3} \sim 0.474\ \mu A$$

Questa corrisponde ad una misura di temperatura al sensore pari a:

$$\frac{0.474}{1\left[\frac{\mu A}{K}\right]} = 0.474\ K$$

Questo valore è sicuramente troppo elevato per il controllo ad isteresi della temperatura che deve essere operato.

Per poter aumentare la sensibilità di Arduino in termini di misura di tensione, una soluzione potrebbe essere quella di modificare la function del Firmware di Arduino che gestisce la

misura di temperatura. Tale modifica viene realizzata in modo che la temperatura non sia più frutto di una singola misura, ma la media di un certo numero di misure realizzate.
In questo modo la sensibilità si dovrebbe ridurre di un fattore pari alla radice quadrata del numero di letture; ciò significa che se opero 100 letture la sensibilità passa da 0,5 K a 0,05 K, se ne opero 400 passa da 0,5 K a 0,025 K ecc. Non sapendo se un numero di letture troppo elevato fosse gravoso in termini temporali per il sistema, è stato scelto un numero di letture pari a quelle operate per la misura della corrente ai LED e ai resistori (100 letture).

Così facendo è possibile vedere come l'andamento della temperatura risulti più comprensibile.

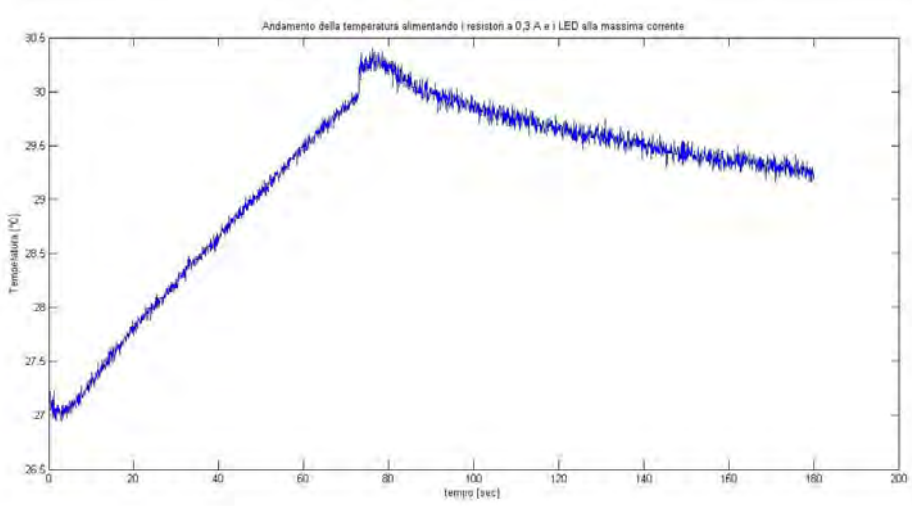


Figura 66: andamento della temperatura alla piastra alimentando i resistori a 0,3 A e i LED alla massima corrente (riducendo la sensibilità a 0,05 K)

Per eliminare anche il rumore presente nella tensione all'ingresso analogico per la misura di temperatura, è stato connesso un condensatore in poliestere da 1μF.

L'implementazione della corrente di 0,3 A ai resistori non risulta così gravosa come si temeva in termini di temperatura massima raggiunta. Questo ha portato alla scelta di questo valore come setpoint di corrente per i resistori.

Inizialmente è stato operato anche una modifica alla function LeggiSeriale() del firmware di Arduino per permettere l'implementazione di un comando di setpoint per la corrente ai riscaldatori; questo serviva più che altro per poter verificare se avevamo effettivamente un controllo sulle correnti oppure no.

Avendo scelto un setpoint fisso, di fatto questo comando risulta inutile ai fini pratici ed il setpoint viene direttamente impostato all'interno dell'isteresi di temperatura; in particolare il setpoint viene impostato nel momento in cui la temperatura scende al di sotto della soglia minima dell'isteresi (vd. function void loop() del Firmware Arduino).

4.5 Stabilizzazione dell'andamento della corrente inviata ai resistori

La corrente che viene inviata ai resistori deve essere soddisfacente anche in termini dinamici. Infatti ad una variazione del setpoint di corrente, la variazione della corrente reale deve compiersi seguendo una dinamica che non presenti sovraelongazioni al di sopra di un certo valore (solitamente 1-2% del valore di regime), e che allo stesso garantisca dei tempi di assestamento accettabili.
I parametri si cui si può agire sono le costanti moltiplicative dei singoli contributi di errore, presenti all'interno del controllore PID.
Inizialmente queste sono state poste uguali a quelle implementate per il controllo delle correnti inviate ai LED. Attraverso l'ausilio di un oscilloscopio si sono misurate le tensioni alle resistenze di shunt sulle quali circolano le correnti inviate ai resistori, ricavando però degli andamenti non soddisfacenti. Facendo delle valutazioni sulle costanti di tempo implementate nel controllo delle correnti ai LED, si è potuto verificare che probabilmente queste sono state implementate in modo che:

- Uno zero del controllore PID vada ad annullare il polo introdotto dal filtro RC del primo ordine posto all'ingresso analogico.
- Il secondo zero del controllore PID introduce una costante di tempo più bassa (0,11 s) rispetto a quella del filtro RC del primo ordine (0,22 s). Di fatto questo zero avrebbe il compito di alzare la pulsazione di attraversamento del sistema in catena chiusa, aumentando la prontezza temporale del sistema stesso.

In prima battuta potrebbe quindi sembrare che il problema dell'overshoot sia legato all'introduzione di una pulsazione di attraversamento troppo elevata, per la quale il margine di fase non è sufficiente a garantire la stabilità del sistema.

Era necessario quindi impostare una formulazione delle costanti moltiplicative del PID affinchè queste potessero essere scritte in funzione della costante di tempo che definisce la nuova pulsazione di attraversamento. Rifacendoci alla formulazione del controllore PID al

paragrafo "1.7 Sistemi in catena aperta ed in catena chiusa", è possibile ricondurre il polinomio di secondo grado nella variabile s, in due polinomi di primo grado:

$$\frac{Ki}{s} * \underbrace{\left(1 + \frac{Kp}{Ki}s + \frac{Kd}{Ki}s^2\right)}$$

$$(1 + s * \tau 1) * (1 + s * \tau 2)$$

$$\tau 1 * \tau 2 = \frac{Kd}{Ki}$$

$$\tau 1 + \tau 2 = \frac{Kp}{Ki}$$

Fissato τ2 = 0,22 s (costante di tempo del filtro RC posto a monte dell'ingresso analogico) e il guadagno di Bode Ki = 0,02, si ottiene:

$$Kd = Ki * \tau 1 * \tau 2 = 0{,}02 * 0{,}22 * \tau 1 = 0{,}0044 * \tau 1$$

$$Kp = Ki * (\tau 1 + \tau 2) = 0{,}02 * \tau 1 + 0{,}0044$$

Implementando dei valori τ1 più alti rispetto al precedente, si è potuto notare l'aumento dell'overshoot fino ad avere, per costanti di tempo elevate un andamento sinusoidale smorzato. Provando al contrario a ridurre la costante di tempo τ1, l'overshoot non diminuisce. Imponendo una variazione del setpoint di corrente da 0,07 a 0,14 A si ottiene, misurando la tensione ai capi della resistenza di shunt da 1 Ω:

Valori scelti per τ1	Andamento transitorio della corrente ai resistori
0,0000011	Overshoot con picco a 286 mV
0,000011	Overshoot con picco a 286 mV
0,00011	Overshoot con picco a 286 mV
0,0011	Overshoot con picco a 286 mV
0,011	Overshoot con picco a 286 mV
0,11	Overshoot con picco a 286 mV
1	Overshoot con picco a 296 mV
10	Sinusoidale smorzato

Si è inoltre verificato che imponendo una variazione identica del setpoint di corrente (0,07 A), la dinamica del sistema è differente a seconda che:

- Si opera una variazione del setpoint di corrente da 0 a 0,07 A.
- Si opera una variazione del setpoint di corrente da 0,07 a 0,14 A.

Si è tentata inoltre un'altra via cercando di agire non solo sulla costante di tempo $\tau 1$ che regola la nuova pulsazione d'attraversamento, ma anche sul guadagno di Bode Ki. L'azione sul guadagno di Bode, variando il valore rispetto a quello preimpostato e pari a 0,02, porta a dei comportamenti diversi a seconda dei casi:

- Riducendolo di molto rispetto al valore impostato, a parità di costante $\tau 1$, i picchi di sovraelongazione crescono.
- Aumentandolo di molto invece i picchi di sovraelongazione tendono a ridursi ma con andamento oscillante.

Dunque partendo dal valore Ki=0,02 (che garantisce un buon compromesso tra la stabilità dell'andamento e il picco massimo della sovraelongazione), ritorniamo di fatto alla situazione vista in precedenza.

In base a queste osservazioni, possiamo concludere che:

- Probabilmente il sistema non è lineare; dunque, a seconda dello stimolo di variazione del setpoint di corrente che viene impostato, la risposta del sistema risulta differente.
- Sarà necessario agire su tutte e tre le costanti moltiplicative del regolatore PID per ottenere un andamento soddisfacente.

Dunque la dinamica della corrente ai resistori viene regolata per il caso particolare di variazione del setpoint di corrente in esame (da 0 a 0,3 A e viceversa, che corrisponde all'accensione e allo spegnimento dei resistori). L'azione di regolazione avviene su tutte e tre le costanti moltiplicative del controllore PID.

Partiamo inizialmente con un controllo proporzionale. È stata implementata una costante Kp inizialmente pari a 0,02, che permette di avere dei valori soddisfacenti di corrente a regime.
Per sistemare la dinamica della corrente è stato introdotto un controllo PI che presenta uno zero con costante di tempo pari a 25 s (per eliminare di fatto l'overshoot che viene rilevato

implementando il solo controllo proporzionale). Nel controllore PI la costante Ki rappresenta il nuovo guadagno di Bode e sarà quindi impostato a 0,02, mentre Kp assume un valore pari a 0,5.

È stata inoltre introdotta un'azione derivativa che ha il compito di velocizzare il più possibile il sistema riducendo la costante di tempo del sistema in catena chiusa. In prima battuta è stato introdotta una costante Kd=Kp=0,02, che ha portato ad avere una risposta al gradino di setpoint 0-0,3 A lenta.

Per cercare di migliorare la prontezza del sistema è stata aumentata la costante Kd. Per diversi valori di Kd è stato rilevato:

Kd	Andamento del transitorio
0,025	Senza sovraelongazione
0,03	Leggero picco di sovraelongazione
0,04	Inizia ad oscillare
0,05	Andamento oscillante

Aumentando la costante Kd ottengo un sistema sicuramente molto più veloce ma anche molto instabile. Per trovare un compromesso tra la prontezza del sistema e l'andamento della risposta al gradino, si è scelta una costante Kd =0,04.

I valori impostati sono:

- Kp = 0,125.
- Ki = 0,02.
- Kd = 0,04.

Tali valori garantiscono la dinamica della corrente ai resitori rappresentata in figura.

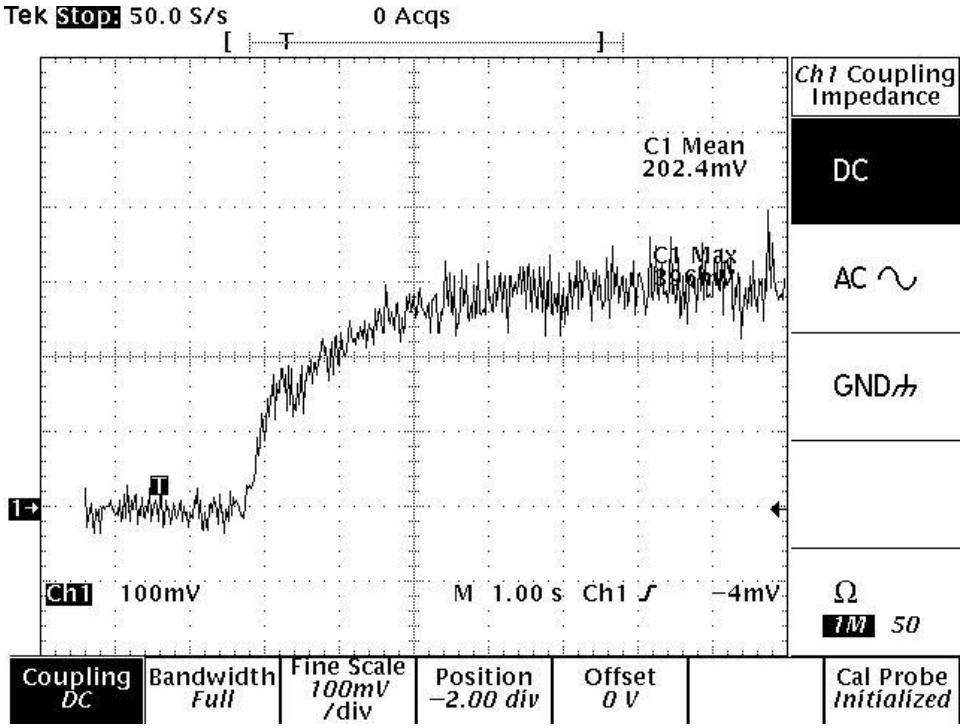


Figura 67: Andamento della corrente ai riscaldatori (misurata come tensione alla resistenza di shunt)

CAPITOLO 5: Risultati ottenuti con la stabilizzazione della temperatura

Per valutare gli effetti della stabilizzazione della temperatura alla piastra, è stata fatta una misura delle frequenze rilevate alle tre diverse tipologie di fotodiodi del sensore TAOS TCS230 e della temperatura alla piastra. In particolare l'analisi si è focalizzata sulla rilevazione tramite i fotodiodi blu del sensore TAOS, accendendo i soli LED blu ad una corrente pari a quella implementata nella rilevazione delle frequenze fatta nel paragrafo "3.2 Rilevazione delle frequenze in catena aperta" (0,56 A). (situazione che presentava maggiore criticità)

L'andamento della frequenza rilevata risulta sicuramente più stabile rispetto all'assenza del controllo di temperatura, ma presenta delle oscillazioni indesiderate.

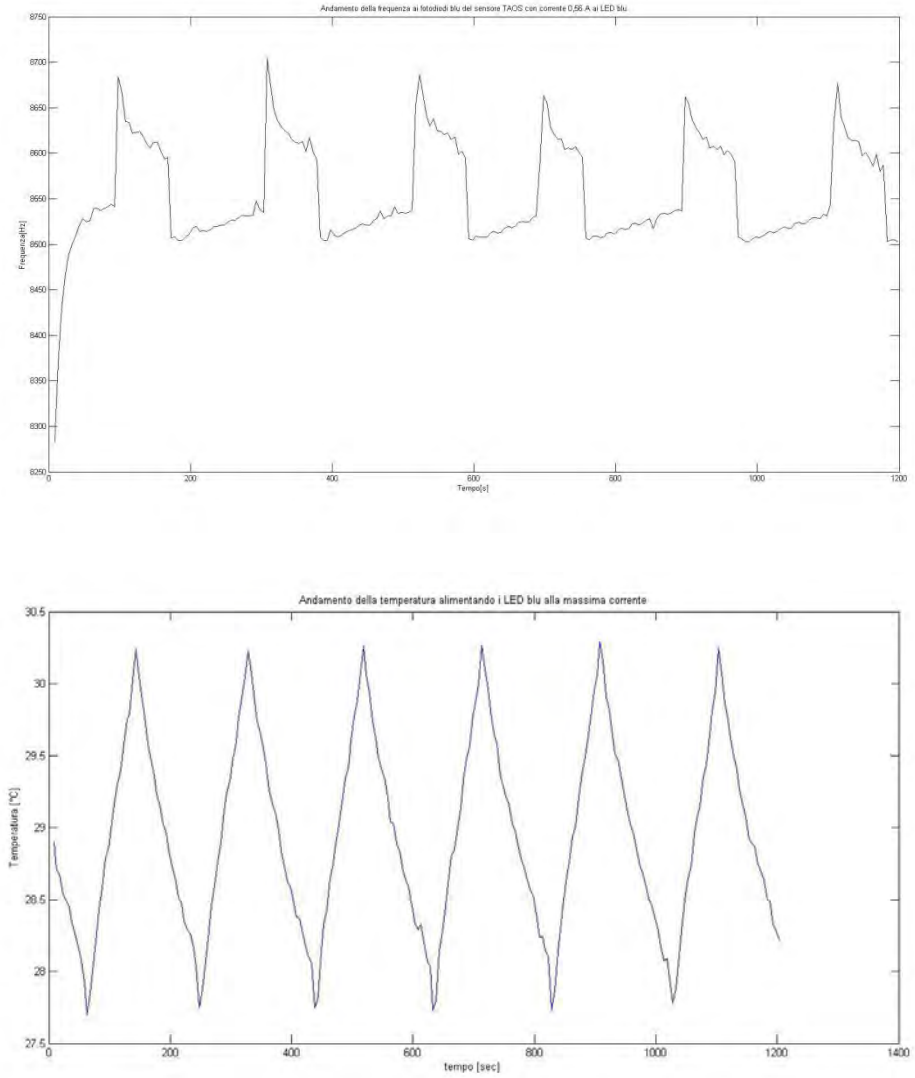


Figura 68: andamento della frequenza rilevata al sensore TAOS blu e della temperatura alla piastra, con LED blu alimentati a 0,56 A.

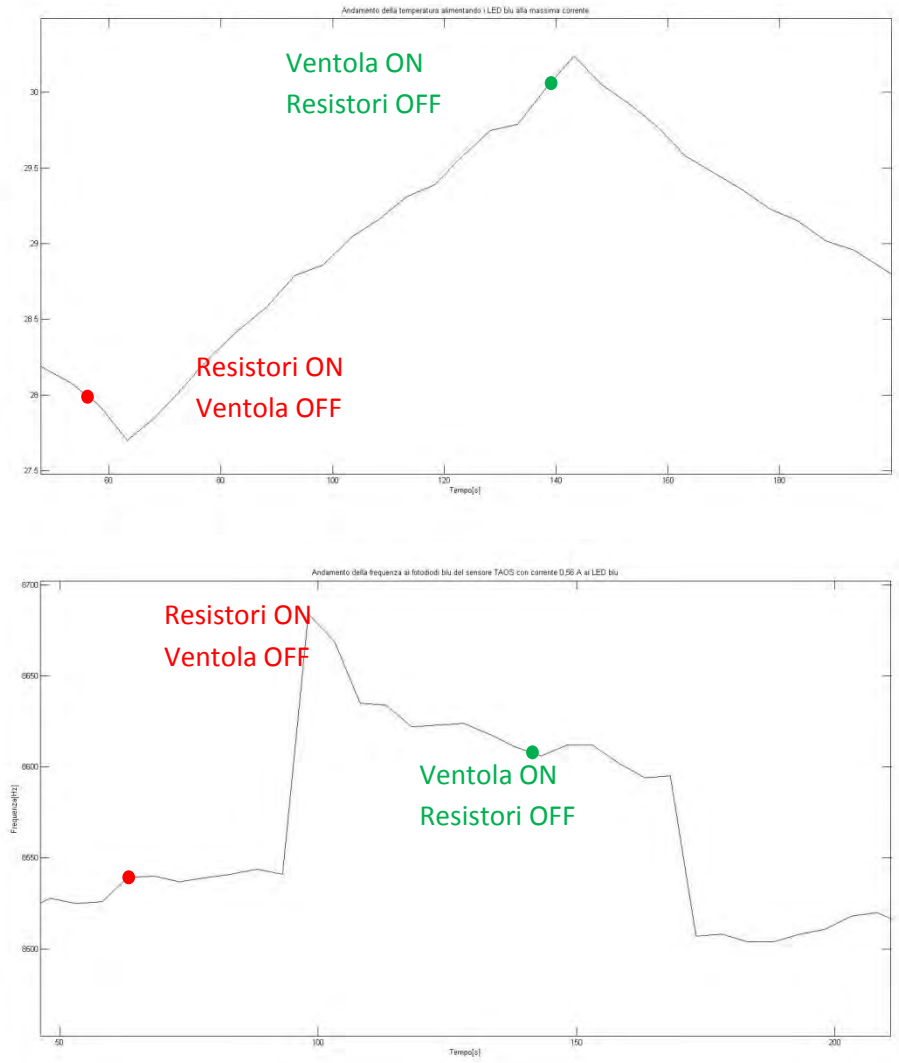


Figura 69: zoom degli andamenti di Figura 68

In particolare il confronto tra l'andamento della frequenza al sensore TAOS e quello della temperatura alla piastra, rilevano che la crescita del valore di frequenza (di valore circa pari al 2%) si verifica per istanti temporali successivi all'accensione dei resistori (e spegnimento della ventola). Quando viene superata la soglia superiore dell'isteresi di temperatura (con conseguente spegnimento dei resistori ed accensione della ventola) la frequenza rilevata al sensore sta dimunendo (sempre del 2%).

Il comportamento è senz'altro anomalo rispetto a quello che ci si aspetterebbe; infatti negli istanti di tempo in cui i resistori sono accesi (e la ventola spenta), la frequenza dovrebbe ridursi in quanto la temperatura alla piastra tende ad aumentare (peggiorando lo scambio termico LED-piastra, con aumento della temperatura di giunzione dei LED); mentre con ventola accesa e resistori spenti, dovrei avere un aumento della frequenza rilevata, in quanto

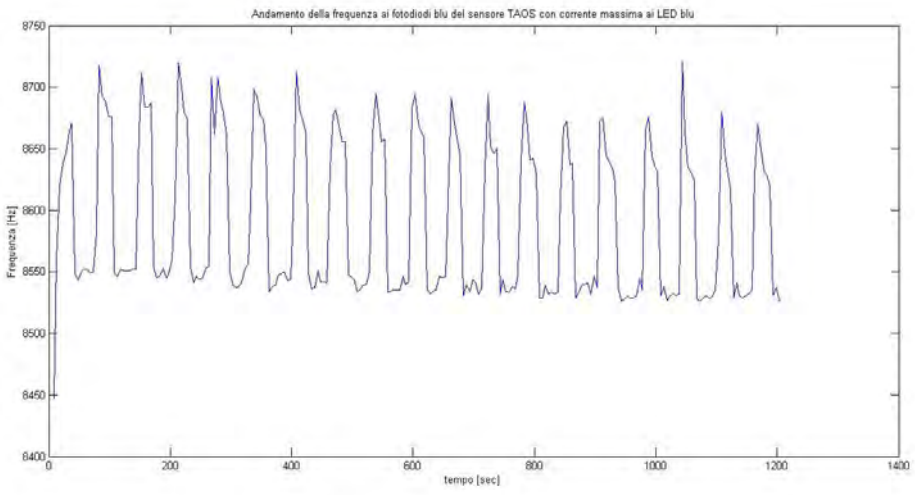
ho una riduzione di temperatura misurata alla piastra (e di fatto ho un miglioramento dello scambio termico LED-piastra).

Sembrerebbe che l'azione di regolazione della temperatura alla piastra, non comporti una variazione congruente della frequenza rilevata al sensore TAOS nell'istante in cui l'azione stessa viene compiuta. Probabilmente l'azione di regolazione di temperatura per stabilizzare l'emissione ai LED è legata alla dinamica termica della piastra, la quale risulta molto lenta. Dunque l'operazione di riscaldamento e raffreddamento operati sulla piastra, non risultano immediatamente efficaci sui LED a causa della presenza di resistenze termiche (sulla piastra metallica, sulla piastrina di montaggio del LED) che rallentano lo scambio termico tra LED e piastra. Dunque l'azione che viene operata sulla piastra metallica risulta efficace ai LED con un certo ritardo.

Per cercare invece di risolvere il problema relativo ai picchi di rilevazione è stata ridotta l'isteresi di temperatura, verificando inoltre se questa operazione sia gravosa in termini di accensione e spegnimento troppo repentini della ventola e dei resistori.

Sono stati portati i livelli dell'isteresi a:

- 28,75 °C: Limite inferiore.
- 29,25 °C: Limite superiore.



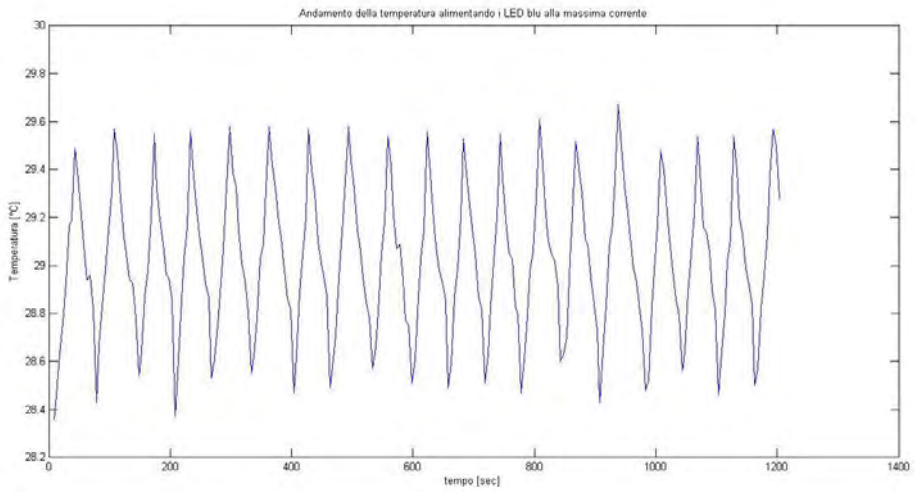


Figura 70: andamento della frequenza rilevata al sensore TAOS blu e temperatura alla piastra con LED blu alimentati con la massima corrente (isteresi ridotta a 0,5°C).

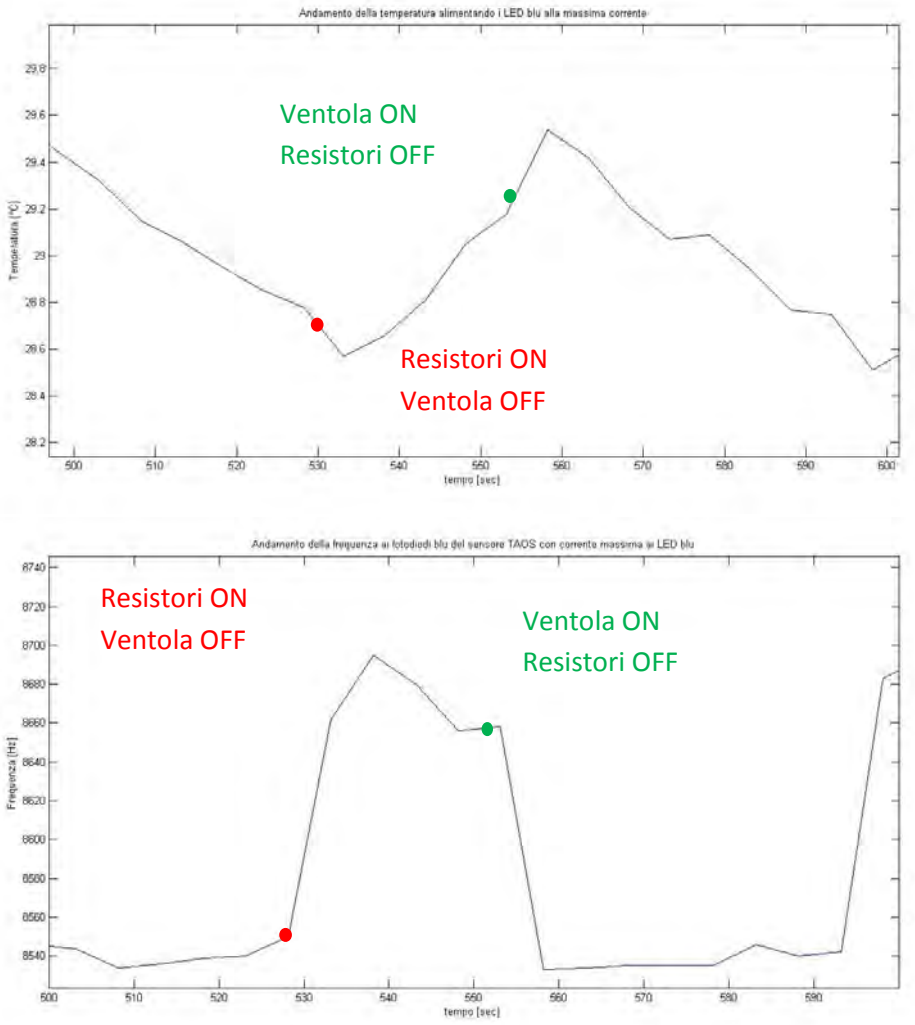


Figura 71: zoom degli andamenti della Figura 70.

La riduzione dell'ampiezza dell'isteresi porta ad avere un'alternanza nell'accensione e nello spegnimento dei dispositivi che è naturalmente più frequente, avendo stretto la fascia d'isteresi. È possibile vedere che i tempi di salita (accensione dei resistori e spegnimento della ventola) e di discesa (accensione della ventola e spegnimento dei resistori) della temperatura, non sono tanto brevi da risultare dannosi per i dispositivi installati.
Per quanto riguarda la rilevazione invece dei picchi di frequenza la riduzione dell'isteresi non ha portato alcun beneficio.

Inoltre le variazioni della rilevazione di frequenza al sensore fanno pensare sia che l'azione della ventola risulti troppo spinta (grossi aumenti della frequenza rilevata al sensore TAOS), ma anche che la corrente implementata (0,3 A) sia ottimale da un punto di vista di variazione di temperatura alla piastra (vd. Paragrafo "4.4 Scelta del setpoint di corrente ai resistori"), ma non lo risulti altrettanto per quanto riguarda le variazioni nella rilevazione di frequenza che vengono realizzate (riduzioni della frequenza troppo spinte). Però mentre la ventola viene alimentata costantemente a 12 V e di fatto la sua azione di raffreddamento risulta difficoltosa da modificare, è possibile ridurre la corrente implementata ai resistori per cercare di ridurre l'energia dissipata sottoforma di calore. Questo dovrebbe comportare una accensione anche meno ripetuta della ventola e quindi di fatto anche l'aumento della frequenza rilevata dovrebbe ridursi. Per poter garantire una migliore efficienza nella riduzione delle oscilazioni di frequenza è stata inoltre aumentata la fascia d'isteresi della temperatura così da rendere meno frequenti le accensioni dei dispositivi.

Per fare questo, avendo ottimizzato i coefficienti del regolatore PID per una variazione di setpoint di corrente ai resistori da 0-0,3 A, è stato modificato il FW. Questa modifica consente di implementare la corrente ai resistori non attraverso un controllo in catena chiusa, ma impostando direttamente il valore della tensione al morsetto ctrl che porta ad avere una corrente circolante ai resistori di 0,1 A (2,88 V).

È stato fatto in questo caso un campionamento di 5 minuti (tempo per cui è stato verificato che la frequenza rilevata al sensore tende a stabilizzarsi) per verificare se l'andamento fosse migliorato. È stata introdotta un'isteresi di 1° C (limite inferiore 28,5°C e superiore 29,5°C).

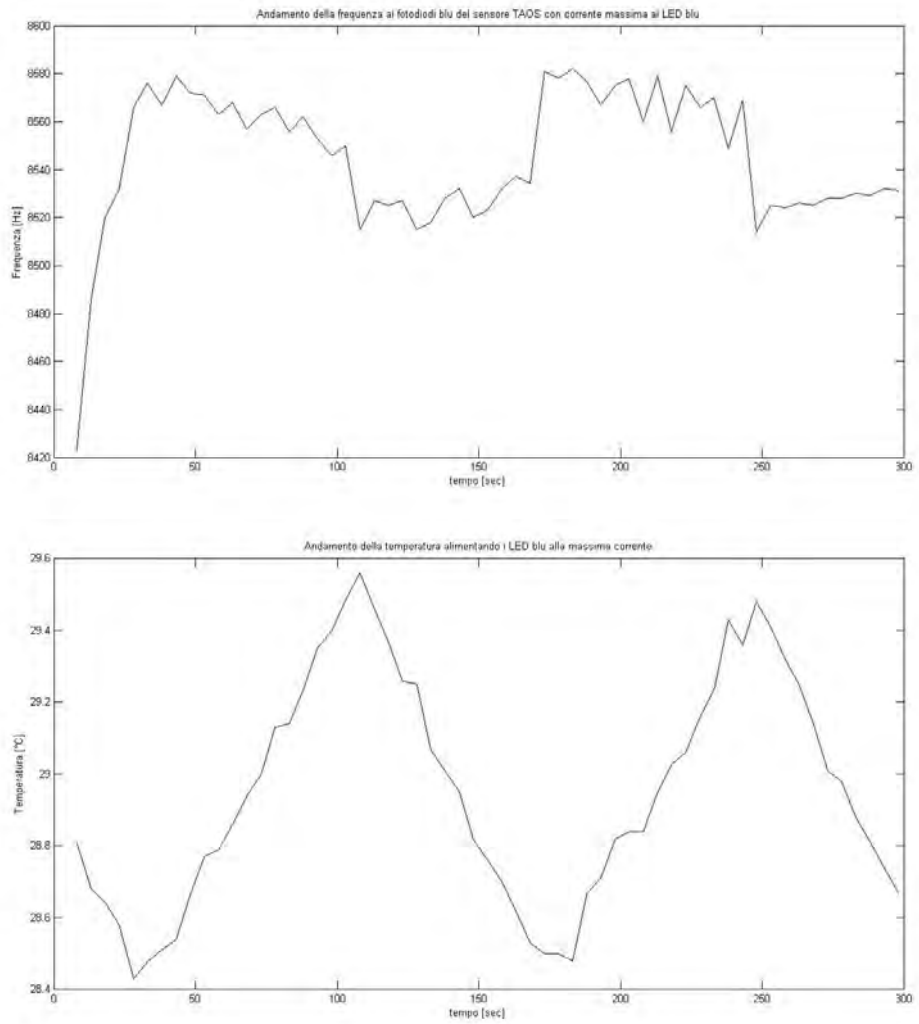
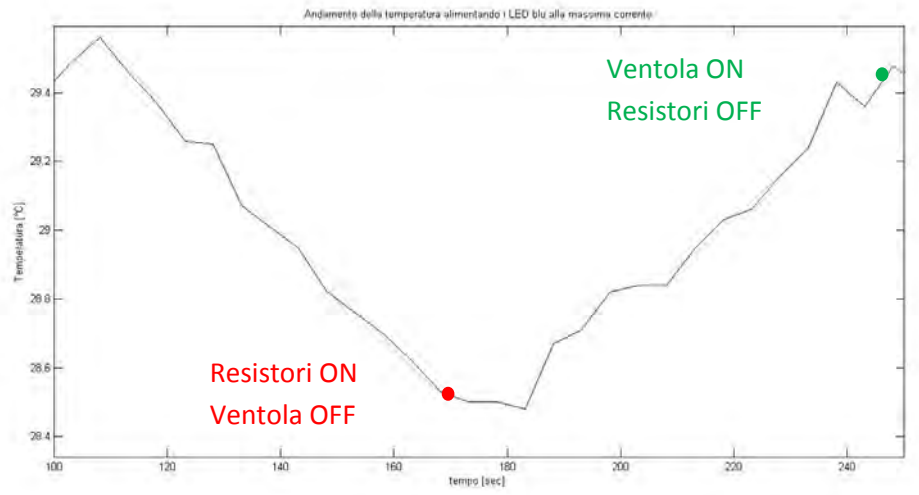


Figura 72: andamento della frequenza rilevata al sensore TAOS blu e temperatura alla piastra con LED blu alimentati con la massima corrente (isteresi 1°C e impostazione diretta tensione ctrl per avere 0,1 A ai resistori).



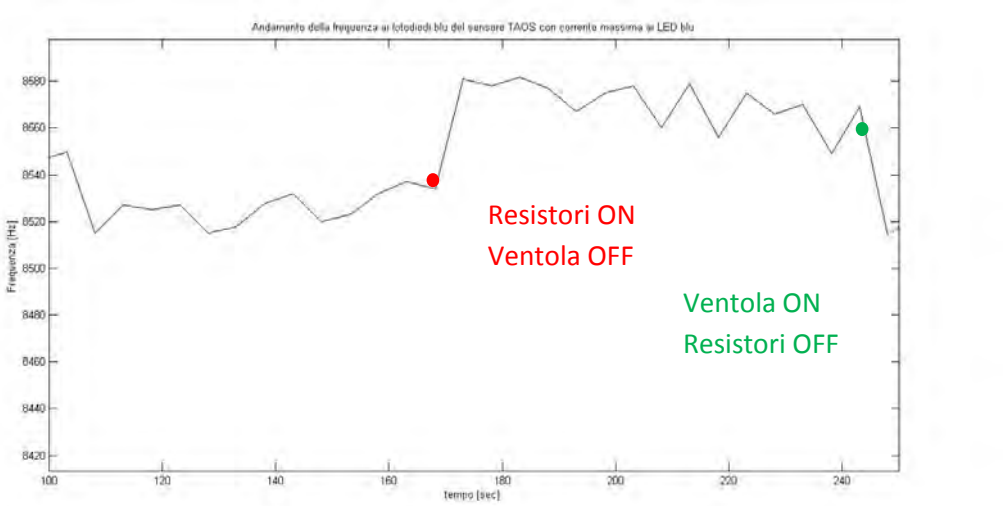


Figura 73: zoom degli andamenti di Figura 72.

Com'è possibile vedere la riduzione della corrente ai resistori ad un terzo del valore precedente, ha portato a rilevare una variazione della frequenza più basso; in particolare questo si è ridotto ad un valore pari a circa 0.8%.

Quanto emerso ci permette di capire che la corrente scelta in precedenza per i resistori (0,3 A) risulta troppo elevata per il controllo della temperatura e che quindi deve essere opportunamente ridotta per ridurre l'entità dele oscillazioni nella rilevazione di frequenza al sensore.

Purtroppo a causa di un malfunzionamento del dispositivo, di cui non si è avuto tempo di rilevarne il motivo, non si è avuto modo di verificare se il controllo di temperatura operato fosse effettivamente idoneo.

La prova che volevo realizzare successivamente era di fatto un confronto finale tra il comportamento del sistema senza il controllo di temperatura e con il controllo di temperatura.
Questa prova consisteva nell'implementare, con e senza il controllo di temperatura, tre livelli di corrente singolarmente per ogni famiglia di LED:

- Corrente pari al suo valore massimo.
- Corrente dimezzata rispetto al suo valore massimo.
- Corrente pari ad un quarto della corrente massima.

Per ognuna di queste tre prove bisognava misurare la frequenza del segnale proveniente dal TAOS, relative all'attivazione del gruppo di fotodiodi corrispondente (accensione LED rossi rilevazione di frequenza ai fotodiodi rossi, accensione LED verdi rilevazione di frequenza ai fotodiodi verdi, accensione LED blu rilevazione di frequenza ai fotodiodi blu).
Una volta realizzate queste prove viene valutato il rapporto tra la frequenza rilevata al sensore e la corrispondente corrente circolante ai LED, per ognuna delle prove fatte per le diverse famiglie di LED.

$$K_{rmax} = \frac{R_{\,TAOS\,r}max}{Imax_r} \qquad K_{r1/2} = \frac{R_{\,TAOS\,1/2,r}}{I_{1/2,r}} \qquad K_{r1/4} = \frac{R_{\,TAOS\,1/4,r}}{I_{1/4,r}}$$

$$K_{gmax} = \frac{R_{\,TAOS\,g}max}{Imax_g} \qquad K_{g1/2} = \frac{R_{\,TAOS\,1/2,g}}{I_{1/2,g}} \qquad K_{g1/4} = \frac{R_{\,TAOS\,1/4,g}}{I_{1/4,g}}$$

$$K_{bmax} = \frac{R_{\,TAOS\,b}max}{Imax_b} \qquad K_{b1/2} = \frac{R_{\,TAOS\,1/2,b}}{I_{1/2,b}} \qquad K_{b1/4} = \frac{R_{\,TAOS\,1/4,b}}{I_{1/4,b}}$$

Dove:

- R_{TAOS} sta ad indicare la frequenza rilevata ai fotodiodi (r: rossi, g: verdi, b: blu) ai diversi livelli di corrente implementati (max: corrente massima, 1/2 : metà del valore massimo, 1/4 : un quarto del valore massimo).
- I: indica la corrente circolante (max: corrente massima, 1/2 : metà del valore massimo, 1/4 : un quarto del valore massimo) ai LED (r: rossi, g: verdi, b: blu).

Quello che ci si aspetta di vedere è:

- Senza il controllo di temperatura: che il coefficiente assuma, in maniera diversa da colore a colore, un valore tanto più piccolo quanto maggiore è la corrente ai LED. In realtà si è visto come, per i LED rossi e verdi l'accensione dei LED con un valore di corrente pari alla massima implementabile, la temperatura di regime della piastra si mantiene circa pari a quella ambiente. Per quanto riguarda i LED blu invece tende a salire intorno ad un valore pari a 29°C. Di fatto quindi il decadimento prestazionale dei LED, e quindi la riduzione del coefficiente K precedentemente visto, dovrebbe essere ridotto per i LED rossi e verdi mentre dovrebbe subire un decadimento maggiore per i LED blu.
  Questo sta a significare che per potenza dissipata crescente per effetto Joule dai LED, corrisponde una temperatura di giunzione diversa, e questo comporta un diverso degrado delle prestazioni dei LED.

- Con il controllo di temperatura: che il coefficiente K, abbia un valore stabile qualsiasi sia la corrente implementata ai LED e quindi qualsiasi sia la potenza dissipata dagli stessi. Certamente il valore che assume il coefficiente K in alcuni casi più basso rispetto ai valori assunti senza controllo di temperatura (prestazioni ridotte dei LED). In particolare ci si dovrebbe aspettare che:
    1. Per i LED rossi e verdi il coefficiente K si mantiene comunque costante, ma pari ad un valore inferiore rispetto a quello in assenza di controllo di temperatura. Questo perché di fatto si è aumentata la temperatura di regime della piastra che viene raggiunto quando sono alimentati singolarmente i LED rossi e verdi (sempre pari all'incirca a quella ambiente, qualsiasi sia la corrente inviata ai LED); questo implica che anche la temperatura di regime dei LED risulta più elevata con il controllo di temperatura e questo degrada di fatto le prestazioni degli stessi.
    2. Per i LED blu invece, si dovrebbe ottenere un coefficiente K costante, con valore più basso rispetto a quelli ottenuti senza il controllo di temperatura per valori di corrente ai LED bassi. Per correnti più elevate invece il coefficiente dovrebbe assumere dei valori prossimi a quelli che si ottengono senza il controllo di temperatura.

       Il controllo di temperatura comporta quindi una variazione del regime termico per valori di corrente ridotti, mentre si mantiene pressoché uguale a quello senza controllo di temperatura per valori di corrente nell'intorno del valore massimo. Quindi le prestazioni dei LED risultano, per correnti ridotte peggiorate, ma la stabilizzazione della temperatura della piastra ci garantisce che l'emissione dei LED sia indipendente dalla dinamica termica della piastra, che viene di fatto eliminata.

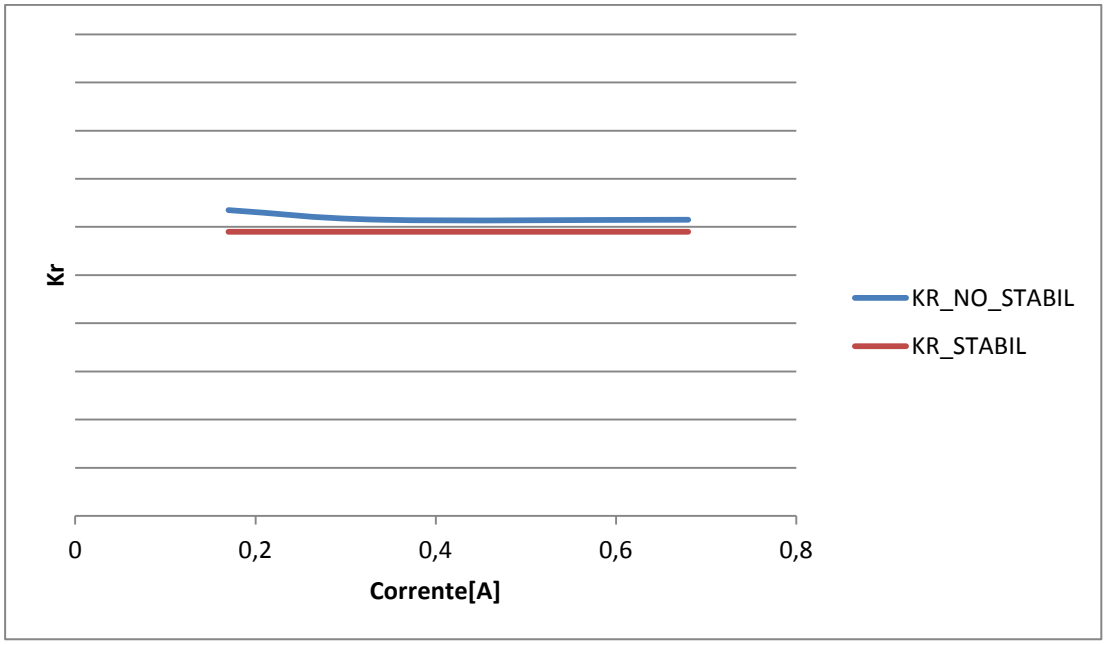


Figura 74: andamento qualitativo del coefficiente K che ci si aspetta di ricavare dalla stabilizzazione della temperatura (alimentazione LED rossi, rilevazione fotodiodi rossi TAOS)

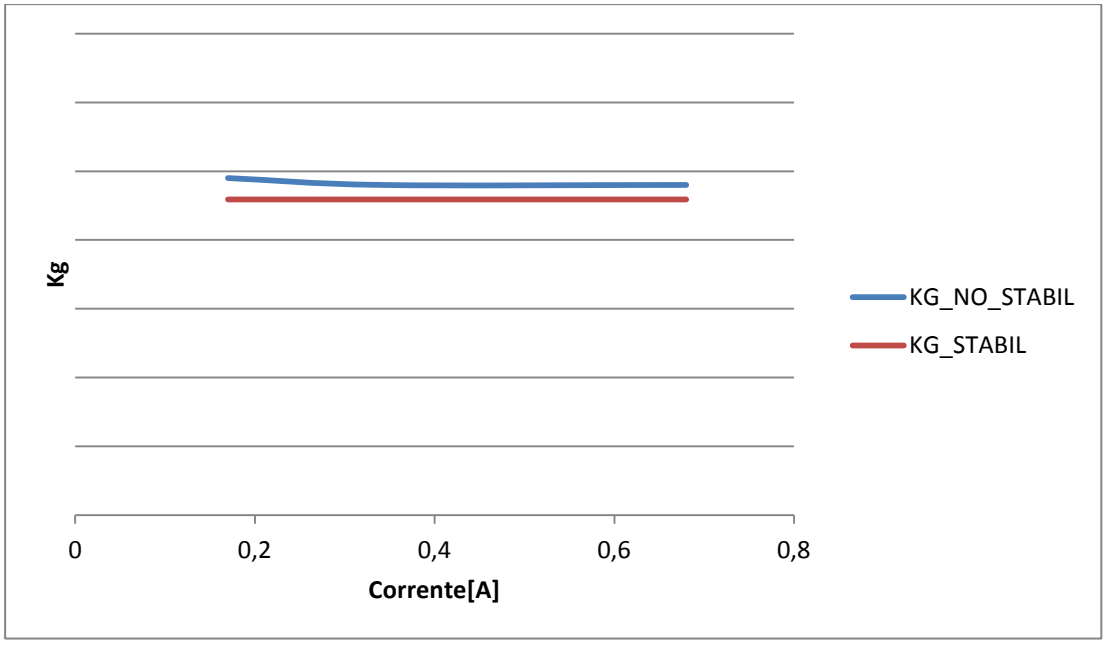


Figura 75: andamento qualitativo del coefficiente K che ci si aspetta di ricavare dalla stabilizzazione della temperatura (alimentazione LED verdi, rilevazione fotodiodi rossi TAOS)

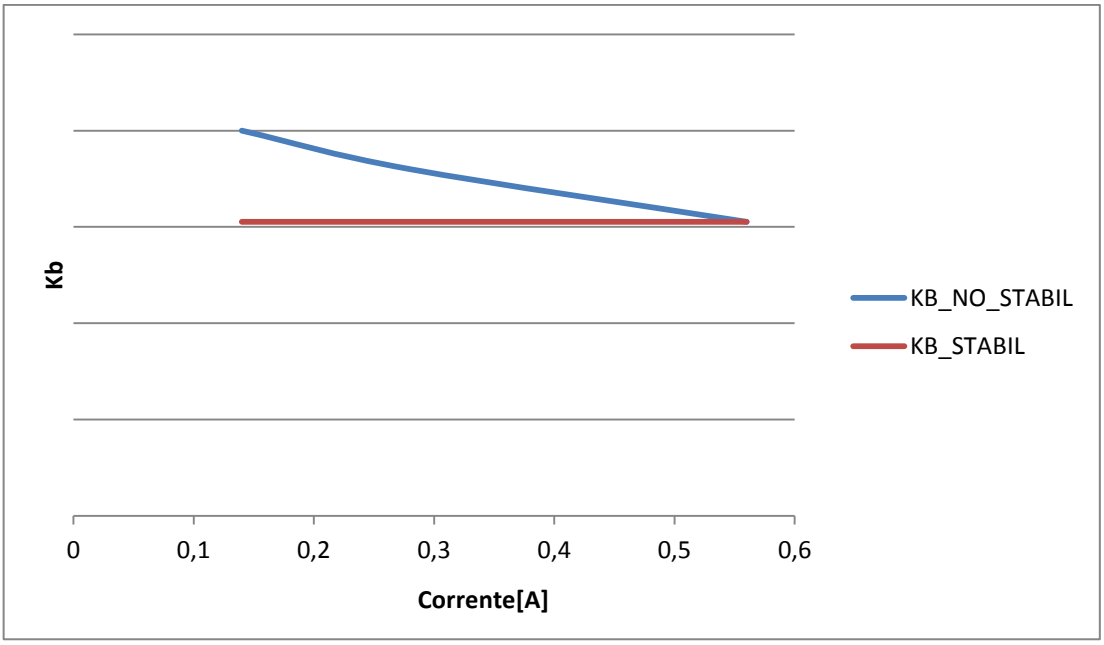


Figura 76: andamento qualitativo del coefficiente K che ci si aspetta di ricavare dalla stabilizzazione della temperatura (alimentazione LED blu, rilevazione fotodiodi blu TAOS)

Queste prove hanno il compito di verificare se effettivamente la stabilizzazione della temperatura alla piastra garantisca un buon funzionamento della sorgente. Quello che si voleva verificare con queste prove, e che purtroppo non è stato possibile realizzare, era se la riduzione dell'intensità della radiazione fosse sempre proporzionale alla variazione di corrente. Questa situazione è verificata nel momento in cui il coefficiente K precedentemente visto si mantiene costante qualsiasi sia la corrente implementata.

Questo sta a significare che, con il controllo di temperatura alla piastra, non esiste nessun fattore di influenza sulla radiazione emessa dai LED e che quindi le coordinate cromatiche che descrivono la tinta della radiazione emessa di ogni singolo gruppo di LED si mantengono costanti. Questo vorrebbe dire non avere variazioni cromatiche della radiazione emessa. L'unica variazione che subisce sarà in termini di intensità luminosa che però è direttamente proporzionale alla corrente implementata ai LED. Ovviamente le prove descritte dovranno essere accompagnate da misure attraverso lo spettroradiometro, per verificare la veridicità dei risultati ottenuti.

Di certo l'esito di queste prove possono rappresentare un punto di partenza per verificare se effettivamente il controllo di temperatura operato sulla piastra metallica sia efficiente o meno.

Se il riscontro fosse positivo, si potrebbe procedere alla realizzazione della matrice di trasformazione frequenze-correnti che consentirebbe di trasformare una terna di frequenze

che vogliamo ottenere al sensore TAOS TCS230 in una corrispondente terna di correnti da inviare ai LED.

CAPITOLO 6: Aspetti da sviluppare

Gli aspetti migliorativi che possono essere applicati al lavoro fin qui fatto sono sicuramente molteplici.

Una volta che è stata individuata la causa del malfunzionamento del sistema, un aspetto da migliorare riguarda il problema delle frequenze rilevate dai sensori TAOS una volta operata la stabilizzazione della temperatura. Il punto da cui partire è legato al fatto che l'andamento delle frequenze deve essere ottimizzato agendo sia sulle correnti implementate ai resistori che sui livelli della fascia d'isteresi, per cercare di ottimizzare l'andamento. Sicuramente questo potrà rappresentare un punto di partenza per coloro che decideranno di affrontare una Tesi di Laurea sul dispositivo analizzato.

Altro aspetto sicuramente da migliorare è il controllo di temperatura; infatti ora l'attivazione dei riscaldatori e della ventola viene fatta in modalità ON-OFF. Un'idea potrebbe essere quella di poter regolare a seconda delle necessità sia il sistema di raffreddamemento che di riscaldamento, così da poter stabilizzare la temperatura in modo ottimale.

Un ulteriore aspetto, che purtroppo non si è avuto modo di affrontare, è la realizzazione delle prove per confrontare il comportamento della sorgente con e senza la stabilizzazione della temperatura.
Da queste prove sarà possibile, nel caso i risultati siano soddisfacenti, passare alla realizzazione della matrice di trasformazione frequenze-correnti.

Una volta fatto questo sarà possibile migliorare l'interfaccia grafica; l'idea di partenza, in parte anche abbozzata, era quella di realizzare un'interfaccia grafica che permettesse di inserire, dalla Command Window di Matlab, le frequenze che si desidera ottenere ai sensori TAOS. Una volta fatto questo si passava a calcolare le corrispondenti correnti che dovranno circolare ai LED per garantire di ottenere dai sensori TAOS le frequenze implementate, attraverso la matrice di trasformazione frequenze-correnti.
Oltre a migliorare la bozza di interfaccia già realizzata, sarebbe quindi utile, a mio avviso, cercare di implementare un'interfaccia grafica che consenta di scegliere ad esempio un colore con delle specifiche coordinate tricromatiche CIE (con la possibilità ad esempio di poter anche cambiare il sistema di coordinate CIE a cui si fa riferimento L,x,y , Lab, L*a*b*, Lu'v' ecc) oppure attraverso la scelta intuitiva attraverso mouse o tastiera di un colore all'interno

della rappresentazione grafica che descrive il particolare sistema di coordinate tricromatiche scelto.

Altro aspetto, con il quale ho avuto modo di scontrarmi durante questo lavoro, è la poca elasticità che il sistema presenta a livello fisico; ad esempio il cablaggio tra la scheda di controllo dei LED e quella di alimentazione degli stessi è risultato troppo corto e questo limitava molto spesso l'agilità di manovra quando si operavano delle misure sul campo attraverso multimetri o oscilloscopio.

CAPITOLO 7: Conclusioni

Nonostante le notevoli difficoltà e le migliorie che il lavoro svolto dovrà subire, sono soddisfatto di quanto fatto.

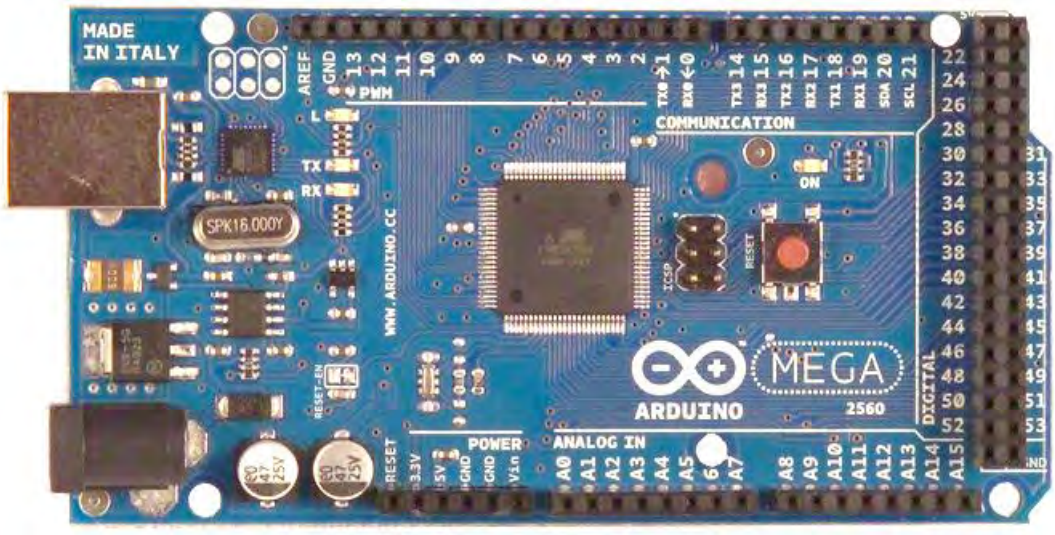
Credo che l'esperienza di Tesi in laboratorio sia un'esperienza utile che vada a completamento di concetti, spesso fatti solo a livello teorico, sviluppati in tutti questi anni di Università (conoscenze di Controlli Automatici, Elettronica di Potenza, Illuminotecnica e Fotometria).

Tale lavoro mi ha consentito inoltre di prendere manualità (nel campo ridotto del lavoro svolto) con strumenti come ad esempio il saldatore a stagno, la scheda Arduino, con le misure attraverso lo spettroradiometro e l'oscilloscopio.

Credo inoltre che l'esperienza pratica mi sia stata d'aiuto per sviluppare un maggior senso critico verso ciò che si andava a realizzare e cercare di capire, spesso dopo lunghe riflessioni e prove, perchè in certi casi le cose non andavano per il verso giusto e di cercare una soluzione per risolvere oppure ovviare al problema.

Quindi alla fine di questa esperienza sento di dire che oltre ad essere stata un'esperienza pratica che sicuramente sarà di grande aiuto anche nel campo del lavoro, in cui mi accingo ad entrare, è stata anche un'esperienza di vita che aiuta a crescere sia come ingegnere che come uomo.

Pin Arduino Utilizzati



I pin Arduino utilizzati nella realizzazione di questo progetto sono:

Ingresso analogico	Utilizzo
A1	Misura Corrente LED rosso
A2	Misura Corrente LED verde
A3	Misura Corrente LED blu
A4	Misura tensione ctrl driver LED rossi
A5	Misura tensione ctrl driver LED verdi
A6	Misura tensione ctrl driver LED blu
A7	Misura Corrente Riscaldatore 1
A8	Misura Corrente Riscaldatore 2
A9	Misura Temperatura

Uscite PWM	Utilizzo
2	Tensione ctrl driver rosso
3	Tensione ctrl driver verde
4	Tensione ctrl driver Riscaldatore 2
5	Tensione ctrl driver blu
6	Tensione ctrl driver Riscaldatore 1
7	Tensione Gate MOSFET

Pin Digitali	Utilizzo
33	Pin TAOS S0
35	Pin TAOS S1
37	Pin TAOS S2
39	Pin TAOS S3
41	Pin TAOS OE

Listati Matlab

ApriSeriale.m: consente di aprire la comunicazione tra PC ed Arduino attraverso la porta seriale.

ChiudiSeriale.m: consente di chiudere la comunicazione tra PC ed Arduino.

Convertinstringa.m: consente di trasformare la terna di correnti, relativa ad una terna di frequenze implementate, in una stringa di caratteri comprensibili ad Arduino. (non usata)

CreaMatriceTrasf_freq_corr.m: consente di realizzare e salvare la matrice di trasformazione frequenze correnti, prelevando i valori di corrente inviate ai LED e di frequenze al sensore TAOS dai file contenenti le rilevazioni fatte dopo aver stabilizzato la temperatura.

LeggiSeriale2.m: file di lettura del buffer d'uscita di Arduino fintanto che questo non è stato completamente svuotato (lo svuotamento del buffer viene comunicato attraverso un messaggio d'errore).

LeggiSeriale3.m: effettua una oppure tre letture del buffer d'uscita a seconda del comando richiesto, salvando i dati all'interno di un vettore.

lettura_tot.m: consente, specificando il setpoint di corrente ai LED nella variabile comando, di effettuare delle misure di frequenza ai tre diversi gruppi di fotodiodi del sensore TAOS, di corrente alle tre famiglie di LED e di temperatura alla piastra (richiamando la funzione Rileva__campioni _frequenzacorr_tot.m).

lettura_rgb4.m: permette di ottenere la temperatura alla piastra e gli istanti in cui vengono effettuate le rilevazioni, richiamando la funzione Rileva_campioni_temp.m.

LimitiCorrenteLED.m: consente di valutare se la terna di frequenze che vogliamo ottenere al sensore TAOS siano associabili ad una terna di correnti ai LED effettivamente riproducibile con il sistema installato (non usata).

Rileva_campioni_frequenzacorr_tot.m: permette in prima battuta di scegliere una finestra temporale su cui effettuare la misura. All'interno di tale frequenza temporale vengono realizzate delle misure di corrente ai LED (alla massima velocità di campionamento), di frequenza ai fotodiodi del sensore TAOS e di temperatura (rallentando il campionamento per non rendere instabile il controllo di corrente ai LED).

Rileva_campioni_temp.m: consente in prima battuta di scegliere la finestra temporale su cui effettuare la misura e l'intervallo tra un campionamento e l'altro. Una volta implementati i comandi di setpoint di corrente ai LED e ai resistori, consente di campionare la temperatura alla piastra alla massima velocità di campionamento possibile. Ovviamente per poter impostare il setpoint di corrente ai resistori è necessario togliere il setpoint di corrente fisso impostato tramite Arduino.

ScriviSeriale1.m: consente di fornire un comando ad Arduino attraverso la porta seriale e di leggere automaticamente il dato in risposta a tale comando messo a disposizione nel buffer d'uscita di Arduino.

ScriviSeriale2.m: consente la sola scrittura del comando sul buffer di Arduino.

ValutazioneCorrenteLED.m: richiamando tale funzione è possibile immettere a video le frequenze che si desidera ottenere dal sensore TAOS. Viene valutata la reale possibilità di realizzazione attraverso la funzione LimitiCorrenteLED.m e, in caso di reale possibilità di realizzazione, si opera passa alla creazione della stringa di comando di Arduino relativa al setpoint di corrente che si vuole impostare. (non usata)

Firmware Arduino

LED_RGB_14: utile a ricreare la condizione di lavoro implementata prima del controllo di temperatura.
LED_RGB_15: realizza il controllo di temperatura attraverso l'isteresi, con un controllo PID delle correnti ai resistori.
LED_RGB_16: realizza il controllo di temperatura attraverso l'isteresi, implementando la corrente ai resistori attraverso un comando diretto al morsetto ctrl del driver.

Bibliografia

Mohan Ned, Undeland Tore P., Robbins William P., "Elettronica di Potenza" Ed. Hoepli.

Dispense del corso di Illuminotecnica e Fotometria, prof. Pietro Fiorentin.

M.E. Valcher, M. Bisiacco, "Controlli Automatici ... tutto quello che avreste voluto sapere a riguardo ma non avete mai osato chiedere", Ed. Libreria Progetto.

Sitografia

www.**arduino**.cc: Informazioni sul funzionamento software e hardware di Arduino.
http://it.mathworks.com/help/matlab/: Informazioni di funzionamento Matlab

Ringraziamenti

Il ringraziamento più importante va sicuramente alla mia famiglia e agli amici, sempre vicini anche durante i periodi di maggior sconforto.
Ringrazio infinitamente il prof. Fiorentin e l'ing. Scroccaro per il grande aiuto dato nella realizzazione di questo lavoro di Tesi e la loro disponibilità.