UNIVERSITÁ DEGLI STUDI DI PADOVA

Dipartimento di Ingegneria Elettronica e dell'Informazione

Corso di Laurea Triennale in Ingegneria Elettronica

Relazione per la prova finale

Sviluppo di un controllore digitale per sorgente luminosa con trasduttore ottico

Development of a light fixture digital controller with optical sensor

Relatore:
Ch.mo Prof. Simone Buso

Laureando:
Alberto Pettena
1203700

Anno accademico 2021/2022

In primo luogo voglio ringraziare il professor Buso per avermi dato l'idea e la possibilità di realizzare questo progetto, e il mio amico Enrico per il grande aiuto nella stesura di questa tesi.

Un ringraziamento speciale va ai miei genitori Monica e Massimo per avermi dato la possibilità di intraprendere questo percorso di studi, ai miei fratelli Marina e Alessandro, a zii, cugini e alla nonna Marisa per avermi sostenuto moralmente.

Ringrazio mio cugino Lorenzo per l'esperienza indimenticabile in appartamento a Padova e i miei colleghi di studi Emanuele, Giacomino e Giacomone per aver condiviso sessioni, partite a briscola e lunghe telefonate di studio su zoom.

Ringrazio inoltre Annalisa per avermi dato la possibilità di una seconda esperienza a Padova aiutandomi a superare molti esami in un periodo difficile.

Infine ringrazio gli amici storici, gli amici nuovi, gli amici del ITIS e tutte quelle comparse che hanno contribuito a farmi raggiungere questo traguardo.

Sommario

Questa tesi affronta il dibattito riguardante la sempre maggiore richiesta di una corretta gestione dell'energia elettrica in particolare nell'illuminazione diurna di un ambiente pubblico, lavorativo o privato.

Si pone come obbiettivo lo sviluppo di un controllore in grado di gestire la potenza erogata ad una sorgente luminosa in base all'illuminazione presente nella zona da illuminare, mostrando i valori di luce e potenza attraverso un display.

Indice

1 Panoramica **1**

1.1 Illuminazione diurna e risparmio energetico 1

1.2 Il progetto 4

2 Hardware **5**

2.1 Microcontrollore STM 5

2.1.1 Datasheet 6

2.1.2 ADC 7

2.1.3 I²C 8

2.1.4 Timers 9

2.1.5 PWM 10

2.1.6 STLink V2 11

2.2 Display 1602 12

2.2.1 Interfaccia I²C PCF8574T 14

2.3 Fotoresistore 14

2.4 LED 15

3 Software **17**

3.1 STM32Cube IDE 17

3.2 Pseudocodice 19

3.2.1 Codice del controllo lineare PWM 19

3.2.2 Macchina di Mealy 22

3.2.3 Visualizzazione display 24

4 Realizzazione e risultati **27**

4.1 Circuito 27

4.2 Strumenti di misura 29

4.2.1 Multimetro 29

4.2.2 Oscilloscopio 30

4.3 Comportamento 31
4.3.1 Verifica dei segnali 32
4.3.2 Analisi funzionamento macchina a stati 34

5 Conclusioni **37**

A Codice **39**

Elenco delle figure **45**

Elenco delle tabelle **47**

Elenco dei codici **49**

Bibliografia **51**

Capitolo 1

Panoramica

1.1 Illuminazione diurna e risparmio energetico

L'illuminazione di un ambiente è parte fondamentale della quotidianità moderna. Ciò che invece è stato trascurato per molto tempo, grazie alla comodità ed efficienza di fonti non rinnovabili, è il risparmio energetico. Il dibattito è aperto da ormai parecchio tempo, ma solo di recente, visti gli aumenti dei prezzi e i danni causati dal cambiamento climatico, è iniziata una ricerca verso qualsiasi metodo di risparmio. Il cambiamento però inizia da ognuno di noi, con gesti piccoli come per esempio quello di spegnere la luce se non è necessaria. Quando si parla dell'utilizzo della luce di solito ci si riferisce all'illuminazione notturna di ambienti chiusi o aperti, privati o pubblici allo scopo del proseguimento di lavoro e attività e per un generale aumento della sicurezza quando necessario. Di giorno questo tipo di illuminazione è ovviamente offerto dal sole, fonte di energia ed illuminazione gratuita. Che motivo c'è di illuminare artificialmente un ambiente di giorno? La sicurezza, specialmente sul lavoro e nel pubblico. Svariate sono le leggi che obbligano ad avere un'illuminazione a norma per i luoghi di lavoro e allo stesso tempo leggi che incentivano all'utilizzo di luce naturale.

Secondo il decreto legislativo 81/2008 1.10.1 [1]:

> "A meno che non sia richiesto diversamente dalle necessità delle lavorazioni [...], i luoghi di lavoro devono disporre di sufficiente luce naturale."

Questo decreto è stato emanato sia per incrementare il risparmio energetico, ma ancor più per migliorare il benessere e la salute a lungo termine di tutte le persone che passano molte ore in locali chiusi. Il sole infatti aiuta il corpo umano

a seguire corretti cicli biologici e aumenta la produttività generale di un'attività. Grazie a questo decreto tutti i nuovi edifici industriali e pubblici vengono pensati e costruiti sfruttando al meglio la luce naturale fornita dal sole e dal cielo azzurro. Si utilizzano metodi semplici, come lucernari, fino a metodi più complessi come condotti di luce per portare luce naturale in stanze prive di finestre.

Nonostante il sole offra un'illuminazione ottimale con un rendimento di colore lineare, ci sono ambienti che a causa delle loro caratteristiche ricevono poca illuminazione naturale. Esempi evidenti possono essere locali sotterranei, gallerie, cantine ma anche capannoni, uffici o edifici in generale con scarso numero di finestre. La normativa UNI EN 12464-1 [2] descrive nello specifico che:

> "Al fine di ottenere una corretta illuminazione è necessario soddisfare tre esigenze fondamentali, quali il comfort visivo (sensazione di benessere), la prestazione visiva (svolgimento del compito anche in situazioni difficili e protratte) e la sicurezza."

e contiene diverse tabelle indicanti i valori di Lux[1] da applicare in tutti gli ambienti lavorativi, pubblici e privati. Lo stesso decreto 81/2008 che incita ad un utilizzo della luce naturale ribadisce che tutti gli ambienti di lavoro devono essere illuminati in sicurezza [1]. Come conseguenza di queste due leggi, nel caso la luce naturale non sia sufficiente a garantire un'illuminazione adeguata, rispettando i valori indicati nella normativa [2], si ricorre a complementare l'illuminazione con delle sorgenti artificiali.

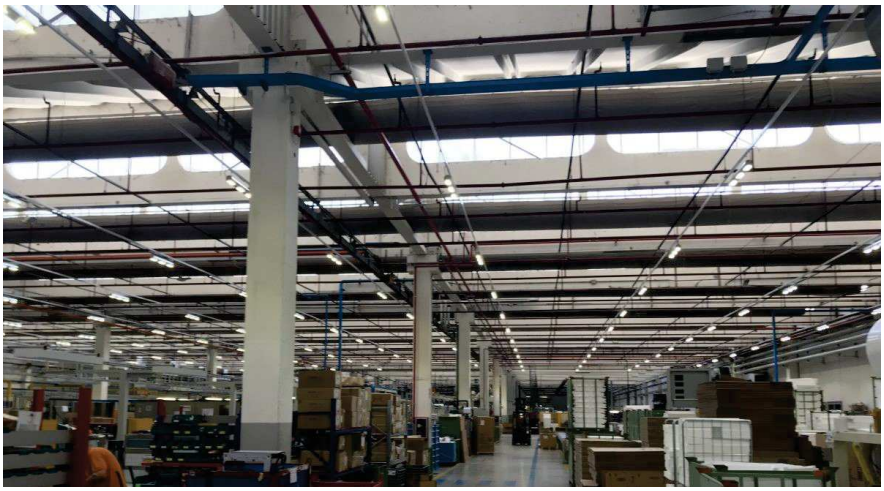


Figura 1.1: Esempio di illuminazione artificiale diurna

[1] Il lux (simbolo lx) è l'unità di misura per l'illuminamento, accettata dal Sistema Internazionale. Un lux è pari a un lumen su metro quadrato.

Lo spreco di energia elettrica diurno deriva proprio da questi casi essendo la luce naturale variabile durante la giornata. Alla mattina, specialmente d'inverno, con l'arrivo dei lavoratori si accendono luci artificiali che vengono lasciate accese anche durante le fasce orarie maggiormente illuminate dal sole. Un altro esempio può essere una stazione all'aperto che deve garantire la sicurezza e un'illuminazione a norma.

L'obbiettivo per rispettare le normative è quindi quello di integrare la luce naturale con quella artificiale utilizzando svariate tecnologie già presenti nel mercato, ad esempio:

- sensori di presenza, che accendono e spengono la luce al passaggio di una persona;
- regolatori di intensità (dimmers) che consentono di controllare e ridurre la potenza in ingresso verso la lampada, riducendo così non solo il consumo di energia ma aumentando anche la vita della lampadina;
- interruttori temporizzati e crepuscolari semplici ed economici, che si usano quando la richiesta di luce artificiale può essere predeterminata;
- fotocellule, che regolano la luce in base all'illuminazione naturale.

In particolare ci si vuole concentrare su lampade intelligenti dotate di fotocellule. Grazie ad un microcontrollore è possibile analizzare il flusso luminoso che investe la fotocellula e dosare la potenza della lampada in modo da risparmiare energia quando la superficie di lavoro è illuminata a sufficienza da luce naturale.

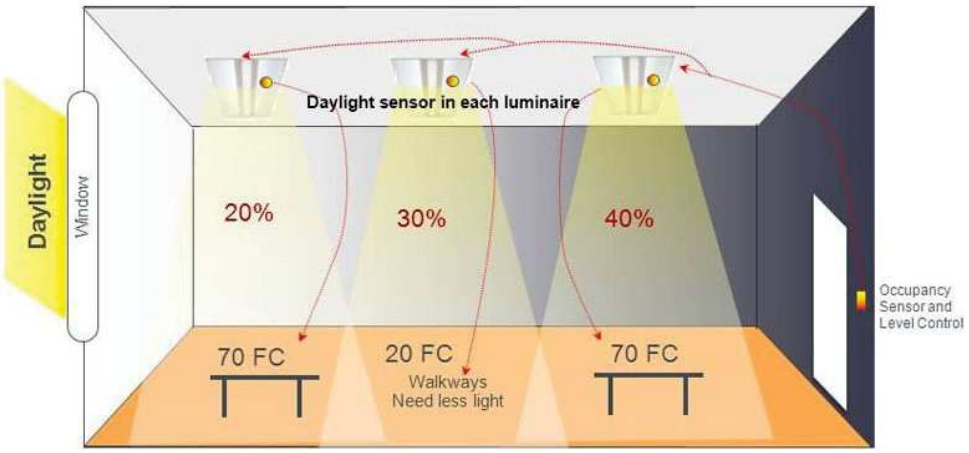


Figura 1.2: Esempio controllo intelligente della luce [3]

1.2 Il progetto

Questo progetto di tesi si pone come obbiettivo quello dello sviluppo di un codice firmware per un microcontrollore e di una progettazione hardware in laboratorio per la visualizzazione del suo corretto funzionamento. È richiesta inoltre l'implementazione di un display che mostri a schermo i valori di luce rilevati dalla fotocellula e il valore di potenza erogata in uscita.

L'implementazione poteva essere fatta con un semplice dispositivo Arduino dotato di ambiente di programmazione proprietario, ma in vista di un possibile sviluppo aziendale si è deciso di scrivere il codice in linguaggio C che permette una programmazione a basso livello. È così possibile l'utilizzo di un microcontrollore di dimensioni e costi ridotti al solo scopo di svolgere le operazioni richieste.

Si è scelto quindi di affidarsi ai dispositivi di STMicroelectronics, azienda leader nei microcontrollori di piccole dimensioni. Il dispositivo scelto è l'STM32F103 il quale viene spesso usato per piccoli progetti scolastici o amatoriali. La programmazione in C viene fatta nell'ambiente STM32CubeIDE , un software proprietario di ST molto intuitivo e con vari strumenti di programmazione.

Successivamente, si è progettato un circuito su breadboard in grado di ospitare il microcontrollore, di connettere le periferiche come il programmatore, il display, il fotoresistore utilizzato come fotocellula e il LED usato per visualizzare la potenza in uscita.

Una volta conclusa la progettazione e la costruzione del circuito sono state eseguite delle prove e simulazioni in laboratorio servendosi delle strumentazioni di misura descritte nel paragrafo 4.2.

Capitolo 2

Hardware

2.1 Microcontrollore STM

Il microcontrollore STM32F103C8T6 appartiene alla famiglia STM32F103x8/xB prodotto dalla casa STMicroelettronics. Il controllore presenta performance medie e monta un processore Arm32 Cortex-M3, con frequenza massima di lavoro di 72 MHz, memoria Flash di 128 *kbyte* e SRAM di 20 *kbyte*. La scheda su cui è montato è progettata per essere alimentata con tensioni dai 3 V ai 5 V ed è possibile alimentarla con una batteria collegandola all'apposito ingresso. Si è scelta questa scheda per le sue dimensioni ridotte, la discreta reperibilità e soprattutto un costo ridotto nonostante la vastità di operazioni che può svolgere. ST fornisce inoltre un programma proprietario, STM32CubeIDE, per programmare ed eseguire debug sulla scheda.

Non è invece presente, a confronto con schede più grandi come Nucleo e Discovery pensate per un ambiente scolastico, un debugger/programmer on-board. ST offre una soluzione con ST-Link esterno che verrà trattato in seguito.

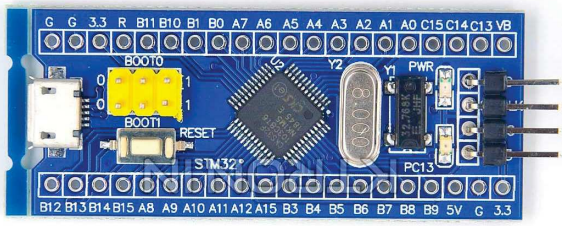


Figura 2.1: STM32F103C8T6 "Blue Pill"

2.1.1 Datasheet

La scheda è dotata di un oscillatore al quarzo da 8 MHz e uno al cristallo da 32 kHz, un regolatore di tensione 5 V – 3.3 V, un tasto reset, due jumper per configurazioni di start-up speciali, un LED che indica l'operatività del dispositivo e un LED programmabile connesso al pin C13. È presente una porta Micro-USB per alimentazione e comunicazione e, dal lato opposto, dei pin per il debug e alimentazione con tensione 3.3 V.

Lungo i bordi longitudinali della scheda sono presenti 40 pin di connessione, la maggior parte dei quali è connessa con i contatti effettivi della scheda. Tra le connessioni si trovano pin di alimentazione 3.3 V, 5 V (proveniente direttamente dall'USB), un ingresso $Vbat$ per l'alimentazione a batteria, un reset e vari pin collegati a massa comune.

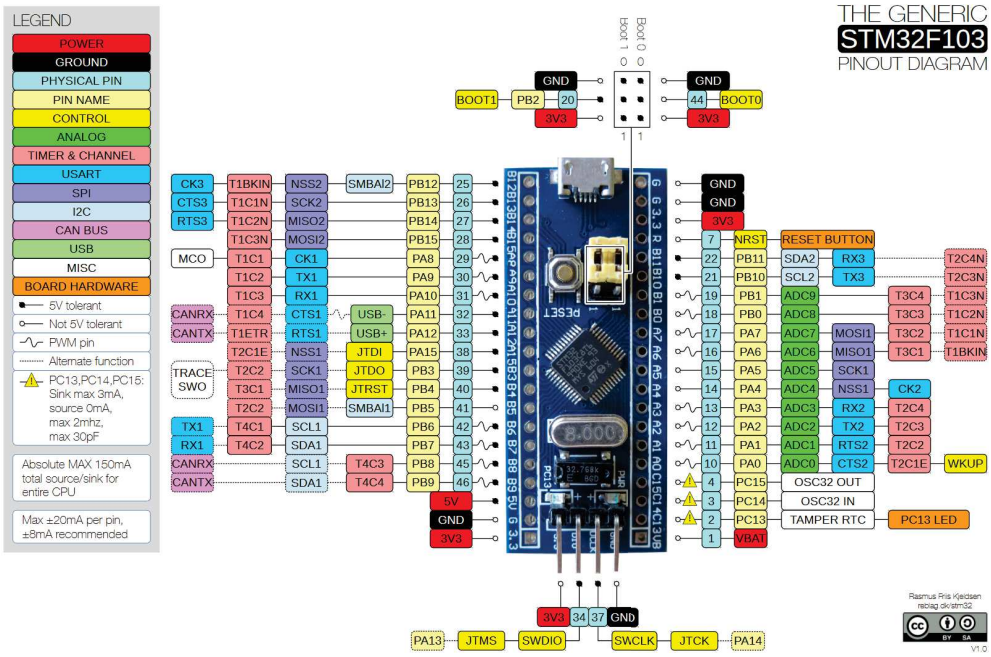


Figura 2.2: Schema Pin Out STM32F103C8T6 [4]

Nel datasheet sono indicate tutte le funzionalità dei pin a cui l'utente si può connettere. Ciascuno ha un numero che si riferisce alla numerazione del chip principale e una dicitura (es. PA6) che aiuta l'utente a configurare correttamente i pin dal software, come si può vedere nella figura 3.1. Alcuni pin sono tolleranti a tensioni fino a 5 V, ciò rende la scheda più robusta a piccole sovratensioni o a picchi di tensione. Altri invece possono avere uscite o ingressi PWM, funzionalità di cui si parlerà in seguito.

2.1.2 ADC

L'Analog to Digital Converter è un dispositivo che utilizza il campionamento e la quantizzazione per convertire un segnale a tempo continuo sia nel tempo che in ampiezza, cioè un segnale analogico, in un segnale a tempo discreto sia nel tempo che in ampiezza, ovvero digitale. Tale periferica, nella catena di acquisizione e distribuzione dati, è utilizzata per convertire i segnali in ingresso derivanti dai sensori, in segnali digitali utilizzabili dal microprocessore. Questo STM dispone di due convertitori a 12-bit che condividono 16 canali esterni (di cui solo 10 sono accessibili sulla scheda) eseguendo conversioni singole o in varie modalità scanner.

La figura 2.3 e la tabella 2.1 mostrano l'accuratezza dell'ADC con V_{ref+} tipica uguale alla tensione di alimentazione.

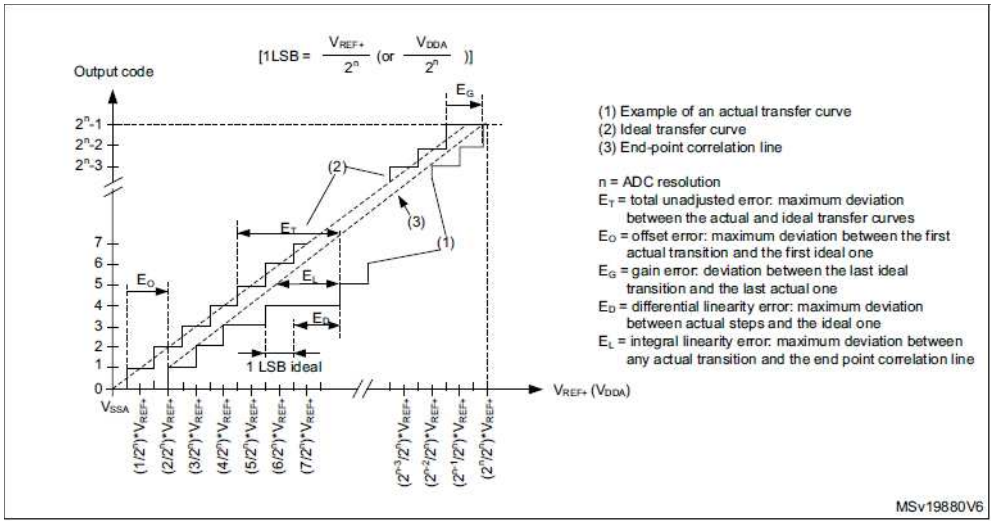


Figura 2.3: Caratteristica e dati dell'ADC1 [5]

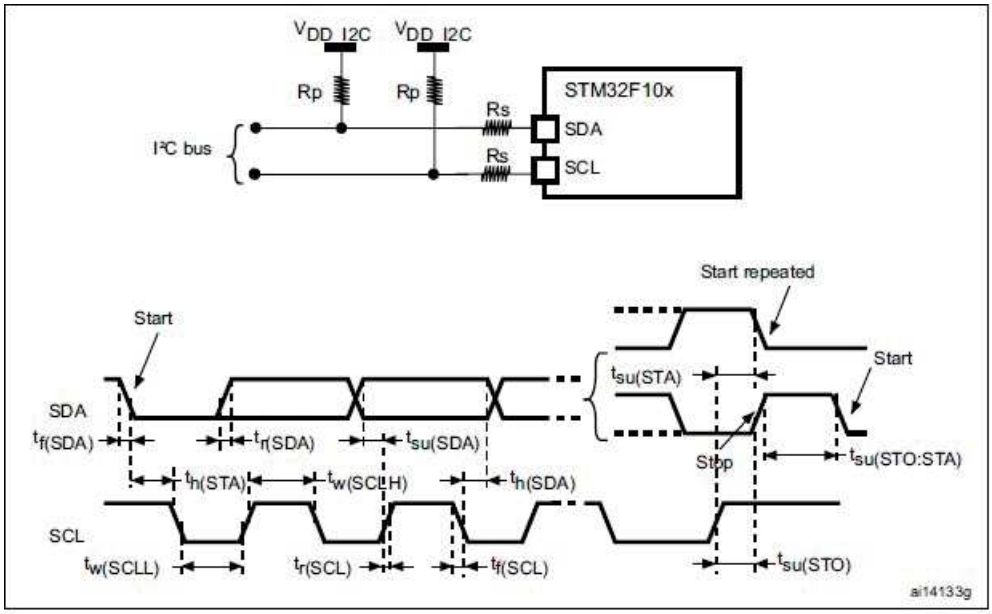
Symbol	Parameter	Test conditions	Typ	Max	Unit
ET	Total unadjusted error	f_{PCLK2} = 56 MHz, f_{ADC} = 14 MHz, R_{AIN} < 10 kΩ, V_{DDA} = 2.4 V to 3.6 V Measurements made after ADC calibration	±2	±5	LSB
EO	Offset error		±1.5	±2.5	
EG	Gain error		±1.5	±3	
ED	Differential linearity error		±1	±2	
EL	Integral linearity error		±1.5	±3	

Tabella 2.1: Caratteristica e dati dell'ADC2 [5]

2.1.3 I²C

Tra i vari sistemi di comunicazione sul bus di cui la scheda è dotata, si è scelto di utilizzare un bus di tipo I^2C [2] per la comunicazione tra il controllore ed il display.

Lo standard di comunicazione I^2C è molto diffuso in ambito industriale. Esso permette il collegamento con una connessione seriale sincrona di molti dispositivi ad un singolo master utilizzando solo due linee di trasmissione, una per i dati e l'altra per il segnale di clock comune [6]. I dati vengono trasferiti 9 bit alla volta, di cui solo l'ultimo è a cura del destinatario in quanto funge da acknowledgment. Questo sistema è fatto in modo da comunicare i dati al corretto destinatario e per assicurarne il corretto invio. Nel caso in cui l'ACK non sia ricevuto dal controllore la comunicazione viene interrotta automaticamente e deve essere riavviata manualmente o con interrupt.



1. Measurement points are done at CMOS levels: 0.3 V_{DD} and 0.7 V_{DD}.
2. Rs = Series protection resistors, Rp = Pull-up resistors, V_{DD_I2C} = I2C bus supply.

Figura 2.4: Schema, SDA e Clock I^2C [5]

Il controllore dispone di due interfacce bus I^2C che possono funzionare in modalità multimaster e slave e supportano le modalità standard 100 $kbit/s$ e veloce 400 $kbit/s$. Il display dispone di un'interfaccia esterna che riceve la comunicazione in standard I^2C e converte il dato ricevuto in analogico per controllare gli

[2] Inter Integrated Circuit, si legge "I quadro C" o più semplicemente "I due C".

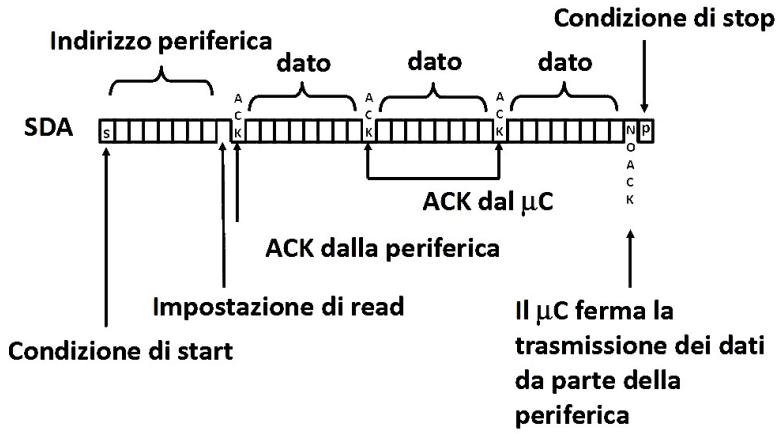


Figura 2.5: Esempio di trasmissione I^2C [6]

ingressi del display stesso. Nella tabella 2.2 si possono osservare le caratteristiche I^2C del controllore.

Symbol	Parameter	Standard mode I²C(1)(2)		Fast mode I²C(1)(2)		Unit
		Min	Max	Min	Max	
$t_{w(SCLL)}$	SCL clock low time	4.7	-	1.3	-	µs
$t_{w(SCLH)}$	SCL clock high time	4.0	-	0.6		
$t_{su(SDA)}$	SDA setup time	250	-	100	-	ns
$t_{h(SDA)}$	SDA data hold time	-	3450(3)	-	900(3)	
$t_{r(SDA)}$ $t_{r(SCL)}$	SDA and SCL rise time	-	1000	-	300	
$t_{f(SDA)}$ $t_{f(SCL)}$	SDA and SCL fall time	-	300	-	300	
$t_{h(STA)}$	Start condition hold time	4.0	-	0.6	-	µs
$t_{su(STA)}$	Repeated Start condition setup time	4.7	-	0.6	-	
$t_{su(STO)}$	Stop condition setup time	4.0	-	0.6	-	µs
$t_{w(STO:STA)}$	Stop to Start condition time (bus free)	4.7	-	1.3	-	µs
C_b	Capacitive load for each bus line	-	400	-	400	pF
t_{SP}	Pulse width of spikes suppressed by the analog filter	0	50(4)	0	50(4)	ns

1. Specified by design, not tested in production.
2. f_{PCLK1} must be at least 2 MHz to achieve standard mode I²C frequencies. It must be at least 4 MHz to achieve fast mode I²C frequencies. It must be a multiple of 10 MHz to reach the 400 kHz maximum I2C fast mode clock.
3. The maximum Data hold time must be met if the interface does not stretch the low period of SCL signal.
4. The minimum width of the spikes filtered by the analog filter is above t_{SP}(max).

Tabella 2.2: Caratteristica I^2C [5]

2.1.4 Timers

Il concetto generale di un timer è quello di un registro a contatore che viene aggiornato alla scadenza di un determinato intervallo di tempo, ovvero il periodo del

timer, in relazione alla frequenza interna della STM32. La frequenza può essere la massima del controllore (72 MHz) oppure variata configurando i prescaler. Una volta raggiunto il valore massimo del contatore, che viene scelto dall'utente tramite impostazioni, il timer si azzera generando un evento che può essere un trigger o la variazione di un flag.

La famiglia STMF103 è dotata di quattro timer da 16 *bit* per le funzioni principali, di cui il primo supporta funzioni speciali. Per il funzionamento del PWM è sufficiente un timer semplice. Si è scelto quindi di usare $TIM2$.

2.1.5 PWM

Un PWM, dall'acronimo inglese Pulse Width Modulation, è un segnale che sfrutta la modulazione digitale di larghezza dell'impulso. Questo tipo di modulazione permette di ottenere una tensione media variabile dipendente dal rapporto tra la durata dell'impulso positivo T_{ON} e di quello totale T. Tale rapporto in percentuale viene definito duty cycle (D).

$$T = \frac{1}{f} = T_{ON} + T_{OFF} \tag{2.1}$$

$$D_{\%} = \frac{T_{ON}}{T} \cdot 100 = \frac{T_{ON}}{T_{ON} + T_{OFF}} \cdot 100 \tag{2.2}$$

La modulazione a larghezza di impulso è utilizzata per variare la tensione, e quindi la potenza, applicata ad un carico generico. Le applicazioni tipiche riguardano motori elettrici in corrente continua, alimentatori di grossa potenza e, nel caso in studio, la variazione di luminosità di sorgenti luminose in particolare dei LED. Con un duty cycle pari a zero la potenza trasferita è nulla mentre al 100% la potenza corrisponde al valore massimo. Ogni valore intermedio determina una corrispondente fornitura di potenza. Il vantaggio di questa tecnica è di ridurre drasticamente la potenza dissipata dal circuito rispetto all'impiego di transistor o di MOSFET controllati analogicamente. In un semiconduttore, la potenza dissipata è determinata dalla corrente che lo attraversa per la differenza di potenziale presente ai suoi capi. In un circuito PWM, il transistor in un istante conduce completamente, riducendo al minimo la caduta ai suoi capi, oppure non conduce affatto, annullando la corrente. In entrambi i casi, la potenza dissipata è ridotta al minimo.

Nella figura 2.2 sono indicati tutti i pin che possono essere abilitati come uscite (o ingressi) PWM e nell'ambiente di sviluppo sono disponibili diverse impostazioni

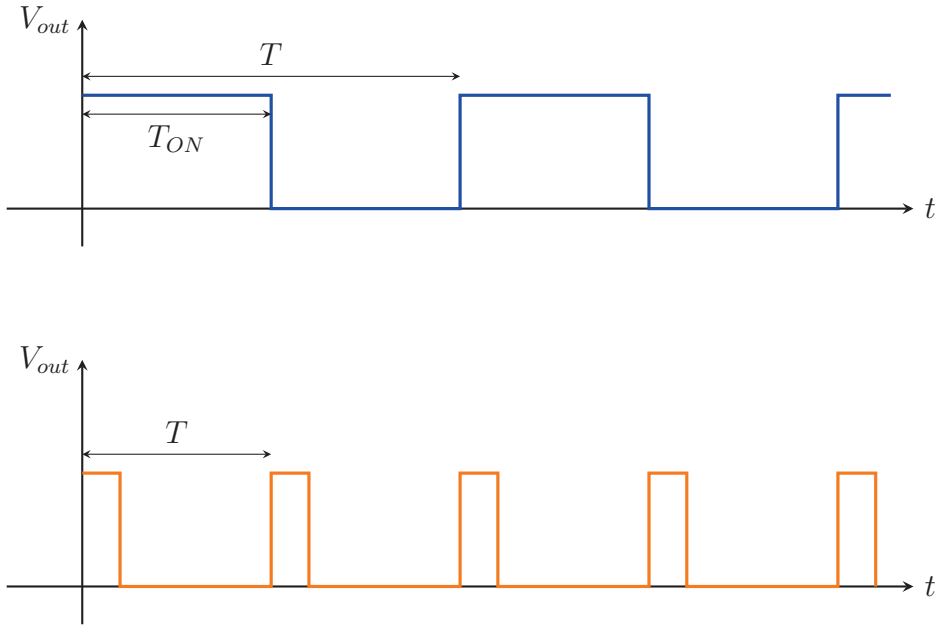


Figura 2.6: Esempi di modulazione PWM

per configurare timers, counters e prescalers.

2.1.6 STLink V2

L'ST-Link V2 è un programmatore esterno in grado di programmare microcontrollori di tipo STM8 e STM32. Dispone di una porta USB-A e di 10 pin posteriori. Tra questi pin troviamo alimentazioni 3.3 e 5 Volt, pin per la comunicazione seriale e il reset del microcontrollore connesso. Il programmatore permette di isolare digitalmente il PC dal microcontrollore ed ha una protezione contro sovratensioni fino a 1000 V. È inoltre in grado di comunicare con l'ambiente di programmazione STM32CubeIDE ed altri programmi di ST.

Figura 2.7: ST Link V2

2.2 Display 1602

Il display LCD1602 è un display a cristalli liquidi che può mostrare 32 caratteri distinti disposti su 2 righe, come suggerisce il nome. Ogni carattere è composto da una griglia di pixel 8x5 con alcuni pixel laterali, come mostrato nella figura 2.9.

Il principio di funzionamento di uno schermo LCD è basato sulle proprietà ottiche di particolari sostanze chiamate cristalli liquidi. In uno schermo LCD i cristalli liquidi sono racchiusi fra due superfici vetrose provviste di numerosi contatti elettrici collegati indipendentemente a ciascun pixel con i quali poter applicare un campo elettrico. Dal lato esterno del vetro è applicata una pellicola polarizzante che polarizza linearmente la luce entrante. La luce riflette sul display che prende un tipico colore verdastro. Quando il campo elettrico viene attivato le molecole del liquido si allineano parallelamente al campo elettrico, limitando la rotazione della luce polarizzata entrante e facendo così apparire il pixel annerito [7]. Annerendo i pixel con determinate combinazioni si crea un carattere, un numero o un simbolo.

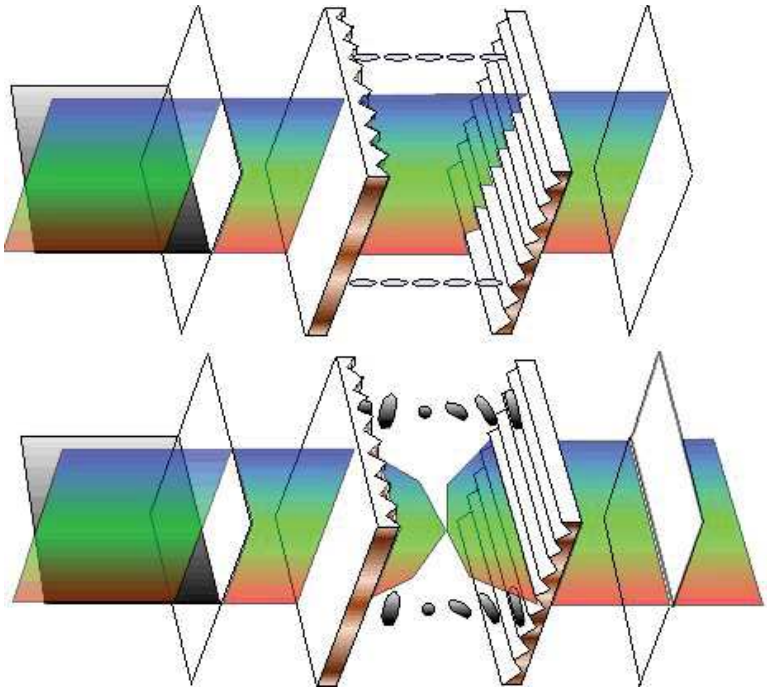
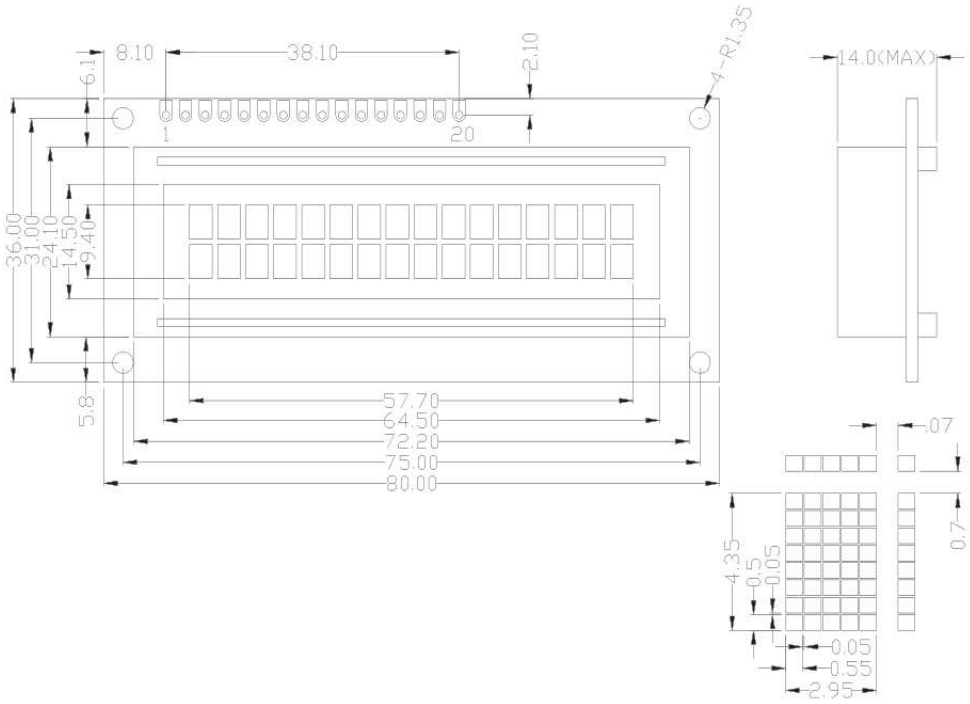


Figura 2.8: Funzionamento dei cristalli [7]

Il principio di funzionamento del display consiste nella lettura dati della memoria interna tramite un processore, anch'esso interno, che si occupa di attivare i corretti pixel per comporre ciascun carattere. È presente quindi un pin I/O per controllare la scrittura di caratteri nella memoria interna o la lettura di quest'ultima. Sono presenti poi svariati pin per l'invio dei dati alla memoria e per il controllo di luminosità e contrasto del display o di altri parametri. I pin in totale sono 16.

Per il corretto utilizzo del display, è necessario che i 16 pin siano connessi al microcontrollore con susseguente progettazione di un firmware per il corretto invio dei caratteri. Dato che questa modalità occuperebbe gran parte dei pin disponibili sul STM, esiste un'interfaccia che sostituisce questo lavoro e permette di comunicare la stringa di caratteri al display, insieme a varie configurazioni, mediante un semplice collegamento con bus I^2C.



1	2	3	4	5	6	7	8
VSS	VCC	VEE	RS	R/W	E	DB0	DB1
9	10	11	12	13	14	15	16
DB2	DB3	DB4	DB5	DB6	DB7	LED+	LED-

Figura 2.9: Specifiche display 1602 [5]

2.2.1 Interfaccia I²C PCF8574T

L'interfaccia si occupa di controllare i pin I/O del display e permette quindi ad un dispositivo, come un microcontrollore, di inviare dati mediante comunicazione I^2C. È dotato di: pin di alimentazione, pin SDA e SCL da collegare al bus I^2C, 16 pin corrispondenti con i 16 pin analogici del display, un LED, un potenziometro per regolare il contrasto del display e tre contatti per impostare manualmente l'indirizzo digitale a cui rispondere durante la comunicazione. L'indirizzo default dell'interfaccia è $0x27$.

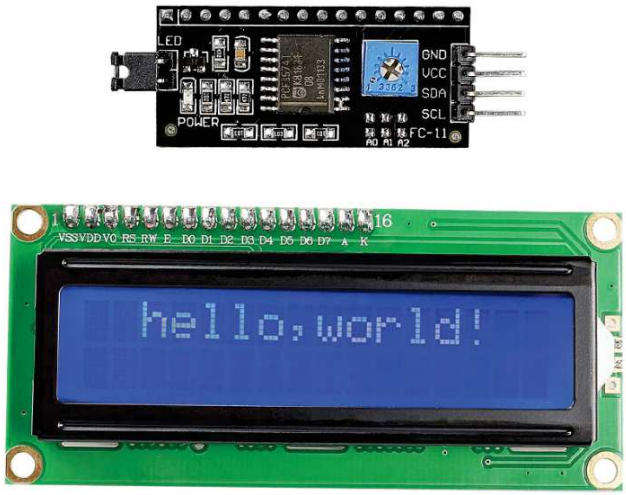


Figura 2.10: Interfaccia PCF8574 (sopra) e display 1602 (sotto)

2.3 Fotoresistore

Un fotoresistore è un resistore fatto di un materiale semi-conduttore che varia la su resistenza in base alla variazione di luminanza che incide sulla superficie. Può essere fabbricato in differenti forme e con differenti aree sensibili alla luce.

Come mostrato nella tabella 2.3, con un basso valore di luminanza la resistenza ai capi del fotoresistore sarà molto elevata, tipicamente nell'ordine dei $M\Omega$. Viceversa con un elevata luminanza il valore sarà contenuto, tipicamente nell'ordine degli Ω.

Il fotoresistore utilizzato per la tesi in oggetto presenta i valori di impedenza presenti nella tabella 2.4. I valori sono stati misurati utilizzando il multimetro Agilent U1241A, spiegato nel capitolo 4.2.1. Nonostante le misure siano state fatte con un valore imprecisato di *Lux* (flusso luminoso), è comunque evidente che la caratteristica del componente non è lineare.

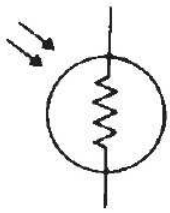


Figura 2.11: Simbolo fotoresistore

T_A = 25°C. 2854°K tungsten light source

Parameter	Conditions	Min.	Typ.	Max.	Units
Cell resistance	1000 lux 10 lux	- -	400 9	- -	Ω kΩ
Dark resistance	-	1.0	-	-	MΩ
Dark capacitance	-	-	3.5	-	pF
Rise time 1	1000 lux 10 lux	- -	2.8 18	- -	ms ms
Fall time 2	1000 lux 10 lux	- -	48 120	- -	ms ms

1. Dark to 110% R_L
2. To 10 × R_L

R_L = photocell resistance under given illumination.

Tabella 2.3: Valori tipici fotoresistore

Illuminazione	Resistenza
Assente	$\geq 10\ M\Omega$
Artificiale Indiretta	$100\ k\Omega$
Artificiale Diretta	$10\ k\Omega$
Naturale	$\leq 1\ k\Omega$

Tabella 2.4: Valori del fotoresistore utilizzato

2.4 LED

Come già esposto, l'obbiettivo di questa tesi è di progettare un firmware che piloti la potenza data ad una sorgente luminosa per adattare la quantità di luce emessa. Questa sorgente nei reali campi di applicazione può avere una potenza di decine o centinaia di Watt e richiede, oltre ad una struttura per l'accomodamento e per il raffreddamento, la progettazione di un circuito di potenza con Mosfet e regolatori.

Figura 2.12: Diodo LED

Allo scopo di vedere concretamente i risultati e il corretto funzionamento del firmware si è deciso di utilizzare un semplice diodo LED collegandolo direttamente all'uscita del controllore.

Il LED è trasparente di luce rossa. La sua tensione nominale è di 1.8 V e va posto in serie ad una resistenza tipicamente di 330 Ω, per evitare sovracorrenti che danneggino il diodo. È stata scelta una resistenza da 390 Ω per via della disponibilità in laboratorio.

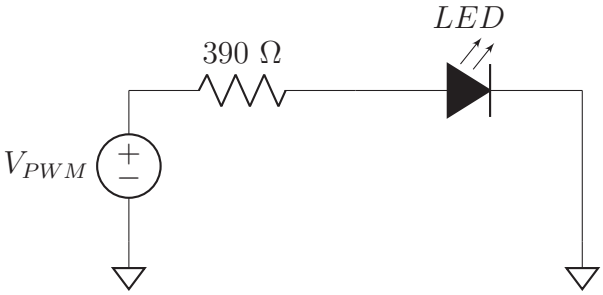


Figura 2.13: Schema tipico diodo LED

Capitolo 3

Software

3.1 STM32Cube IDE

STM32CubeIDE è un ambiente di programmazione disponibile per i sistemi operativi principali in commercio e fa parte dell'ecosistema software STM32Cube. Contiene un'avanzata piattaforma di scrittura C/C++ in grado eseguire la configurazione delle periferiche, la generazione del codice sorgente, la compilazione e il controllo degli errori. Inoltre dispone di applicazioni per il debug dei dispositivi basati su STM32 [8]. È completo di tutorials e di librerie per tutti i chip STM32, in modo da poter generare un codice sorgente adattato allo specifico dispositivo e rendere l'esperienza il più semplice e intuitiva possibile.

Una volta selezionato il Microcontrollore corretto viene visualizzata una schermata (Figura 3.1) per la configurazione delle periferiche. Allo scopo del progetto esposto si è scelto di abilitare un ingresso ADC e di configurare 2 pin per la comunicazione I^2C. Si è successivamente abilitato il Timer TIM2 in modo da generare un uscita PWM per il controllo del LED.

Si passa poi alla schermata per la configurazione del clock e dei prescaler (Figura 3.2). Si è scelta una configurazione base utilizzando l'oscillatore al quarzo da 8 MHz dall'ingresso HSE con il moltiplicatore PPL posto a 2 in modo da avere 16 MHz sulla maggior parte delle periferiche.

Impostata la frequenza dei timer a 16 MHz si deve impostare correttamente il prescaler (PSC) ed il counter period (ARR) del timer TIM2.

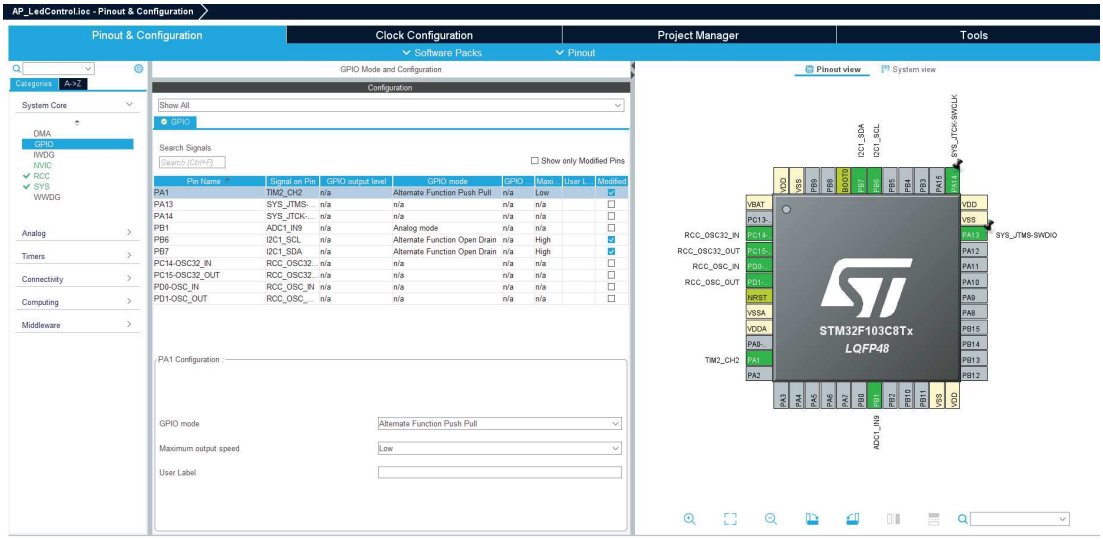


Figura 3.1: Configurazione delle periferiche

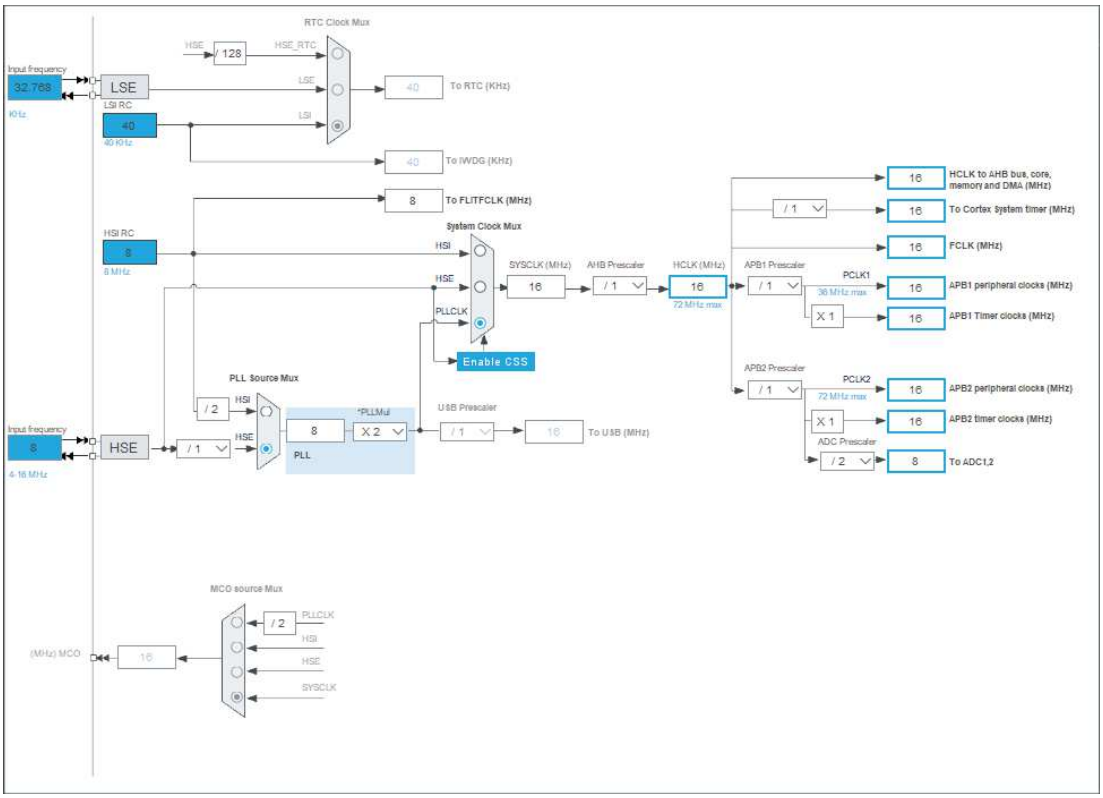


Figura 3.2: Configurazione del clock

Il periodo e la frequenza del PWM si calcolano con le formule:

$$F_{PWM} = \frac{F_{CLK}}{(PSC+1)\cdot(ARR+1)} \qquad [Hz] \tag{3.1}$$

$$T_{PWM} = \frac{1}{F_{PWM}} = \frac{(PSC+1)\cdot(ARR+1)}{F_{CLK}} \qquad [ms] \tag{3.2}$$

Si sono impostati $PSC = 127$ e $ARR = 999$. In questo modo il periodo del PWM è di $\approx 8\ ms$ equivalente ad una frequenza $F_{PWM} \approx 125\ Hz$. Ci si aspetta che la forma d'onda in uscita assomigli alla figura 3.3. Le misurazioni in laboratorio sono illustrate nel capitolo 4.3.1.

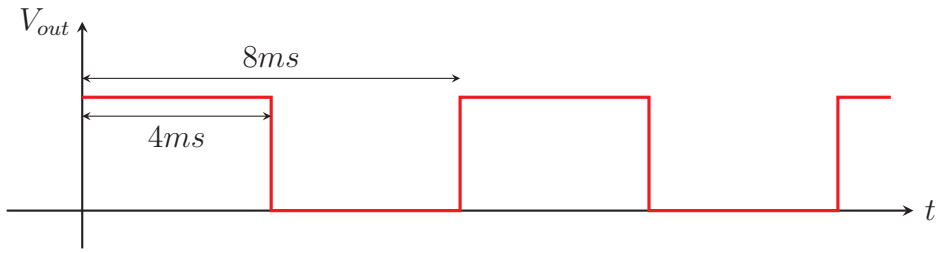


Figura 3.3: Forma d'onda attesa con $T_{on} = 50\%$

3.2 Pseudocodice

3.2.1 Codice del controllo lineare PWM

Considerando il range di luminosità a cui viene esposto il fotoresistore e assumendo che il componente sia soggetto a tolleranze che il Microcontrollore non trascura, si è pensato ad una soluzione per avere un controllo lineare della potenza luminosa solo per un certo range di valori acquisiti dall'ADC.

Eseguendo alcuni test in laboratorio esponendo il fotoresistore a differenti quantità di luce, si sono osservati i seguenti valori in 12_{bit} (ovvero su una scala da 0 a 4096).

Illuminazione	**ADC (12_{bit})**	**Resistenza**
Assente	700	$\geq 10\ M\Omega$
Artificiale Indiretta	1600	$100\ k\Omega$
Artificiale Diretta	3500	$10\ k\Omega$
Naturale	4090	$\leq 1\ k\Omega$

Tabella 3.1: Valori digitali misurati dal microcontrollore

Si osserva che in condizione di luce assente, per cui la lampada dovrebbe dare il massimo della luminosità, il microcontrollore rileva un valore maggiore di `zero` e fluttuante simile a `700`. La stessa cosa si osserva all'estremo superiore dove già con eccesso di luce artificiale si ha un valore di `3500`. È quindi necessario imporre degli spezzamenti nella transcaratteristica per avere dei range ove la potenza della lampada è massima e minima rispettivamente. Si è impostato come valori degli estremi:

- $x_A = 1500$ al di sotto del quale l'ambiente è scarso di luce e deve essere illuminato. La lampada sarà accesa al massimo.
- $x_B = 3800$ al di sopra del quale l'ambiente è già illuminato e non serve luce artificiale. La lampada sarà spenta.

Imposti questi due punti si calcola il coefficiente di pendenza m della retta che li congiunge:

$$m = \frac{x_B - x_A}{y_B - y_A} = \frac{3800 - 1500}{0 - 1000} = -2.3 \tag{3.3}$$

La transcaratteristica, illustrata in figura 3.4, rispetterà quindi la seguente formula dove $Sens$ è il valore in 12_{bit} misurato dall'ADC e PWM_{target} è il valore di *Duty Cycle* (Paragrafo 2.1.5) che la lampada deve raggiungere a regime:

$$\begin{cases} PWM_{target} = 1000 & Sens < 1500 \\ PWM_{target} = 1000 - \frac{Sens-1500}{2.3} & 1500 \leq Sens \leq 3800 \\ PWM_{target} = 0 & Sens > 3800 \end{cases} \tag{3.4}$$

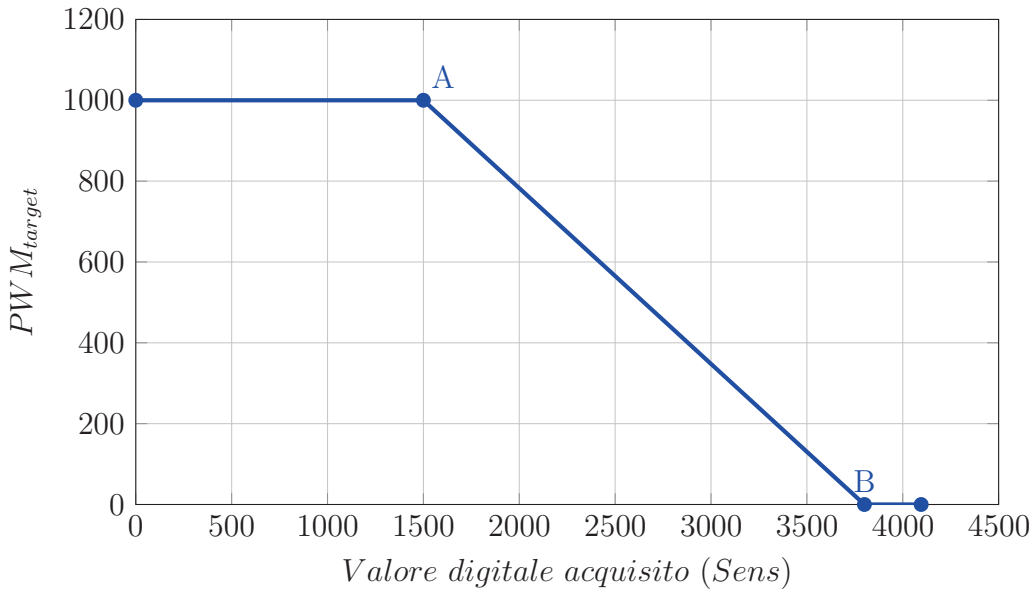


Figura 3.4: Transcaratteristica Ingresso-Uscita

La transcaratteristica viene implementata con il seguente codice 3.1. Si nota che la variabile è rinominata $PWM1$ invece di PWM_{target}.

```
    //parte led PWM

    //Se c'e' poca luce, potenza massima
    if  ( Sens < 1500 )
    {
     PWM1 = 1000 ;
    }
    //Se c'e' sufficiente luce, spegni (oppure si imposta un valore minimo da
    mantenere)
    else if ( Sens > 3800 )
    {
     PWM1 = 0 ;
    }
    //Controllo lineare della potenza del led, opportunamente rapportata
    else
    {
     PWM1 = 1000 - (( Sens - 1500 )  /2.3 ) ;
    }
```

Codice 3.1: Casistica della transcaratteristica

3.2.2 Macchina di Mealy

Il funzionamento del programma segue il modello di una macchina a stati detta anche *Macchina di Mealy* che prende il nome da George H. Mealy, matematico promotore statunitense il quale fu il primo a descriverla nel 1955 [9]. Una macchina di Mealy (versione semplificata della macchina di Moore [10]) è un automa a stati finiti i cui valori di uscita sono determinati dallo stato corrente, che cambia in base alla variazione di ingressi, contatori e variabili. Gli stati di una macchina di Mealy vengono illustrati in un *diagramma di stato*, ovvero uno schema a blocchi che descrive la logica di funzionamento. Ad ogni ciclo la macchina controlla gli ingressi e in base ad una casistica programmata imposta lo stato successivo.

La macchina deve controllare due ingressi e un'uscita:

Input/Output	**Nome**
IN	PWM_{target}
IN	PWM_{set}
OUT	PWM_{set}

Tabella 3.2: Ingressi e uscite della Macchina di Mealy

La variabile PWM_{set} rappresenta il valore di *Duty Cycle* definitivo che andrà poi assegnato al pin di potenza del microcontrollore.

L'obbiettivo di questo diagramma è di variare la potenza del LED (ovvero la variabile PWM_{set}) in modo graduale senza che l'operatore sia soggetto a brusche variazioni di luminosità. È richiesto inoltre che la potenza aumenti velocemente se si trova al di sotto del 50% per evitare situazioni di carenza di luce. L'obbiettivo è raggiunto con un controllo con macchina a stati dove gli stati che variano il valore di PWM_{set} ne aumentano o diminuiscono il valore di un solo punto ad ogni ciclo (viene aumentato di 5 nel caso la potenza sia minore del 50%).

Di seguito viene illustrato il diagramma di stato dove si è indicato PWM_{set} con P_S e PWM_{target} con P_t.

Stato	$\boldsymbol{PWM_{set}}$
ON	1000
OFF	0
DOWN	$PWM_{set} = PWM_{set} - 1$
UP_{fast}	$PWM_{set} = PWM_{set} + 5$
UP_{slow}	$PWM_{set} = PWM_{set} + 1$
STOP	$PWM_{set} = PWM_{set}$

Tabella 3.3: Tabella degli stati

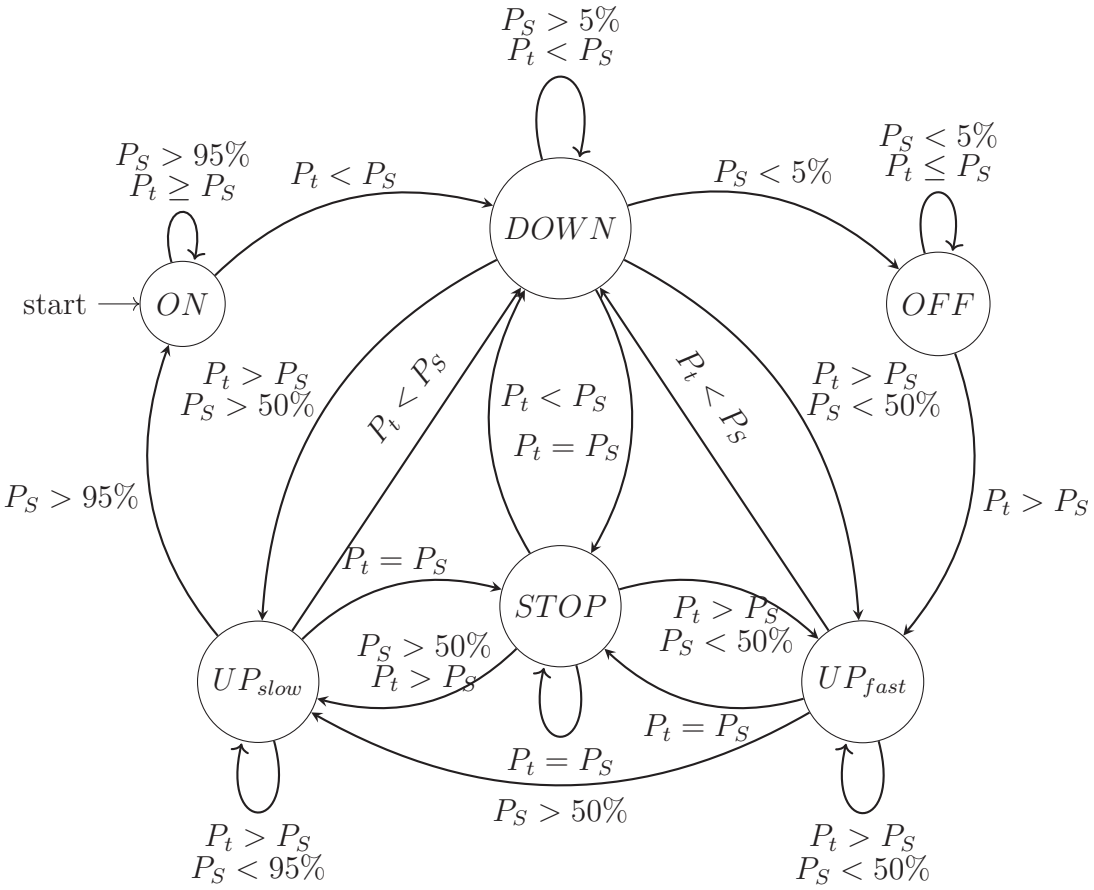


Figura 3.5: Diagramma di stato

Il precedente diagramma è implementato con il seguente codice C.

```
185     //Macchina a Stati
186
187      //caso target minimo e potenza minima = MANTIENI SPENTO
188      if   ( PWMset < 10 && PWM1 < 25 ) {PWMset = 0;}
189      //caso target massimo e potenza massima = MANTIENI MASSIMO
190      else if ( PWMset > 990 && PWM1 > 950 ) {PWMset = 1000;}
191      //caso target maggiore e potenza minore del 50% = AUMENTA VELOCE
192      else if ( PWM1 > PWMset && PWMset < 500 )  { PWMset = PWMset + 5; }
193      //caso target maggiore = AUMENTA LENTO
194      else if ( PWM1 > PWMset )  { PWMset = PWMset + 1;}
195      //caso target minore = DIMINUISCI LENTO
196      else if ( PWM1 < PWMset )  { PWMset = PWMset - 1;}
197      //caso target uguale = MANTIENI
198      else {}
```

Codice 3.2: Casistica della macchina a stati

3.2.3 Visualizzazione display

Nei paragrafi 2.2 e 2.2.1 si è spiegato come il display è dotato di un interfaccia in grado di controllare i pin I/O e permettere la comunicazione con protocollo I^2C. I comandi e le informazioni vengono quindi inviati dal microcontrollore mediante pacchetti I^2C ben definiti. Per semplicità nella programmazione si sono importate due librerie, `liquidcrystal_i2c.c` e `liquidcrystal_i2c.c`, in cui sono definiti i comandi per controllare il display: ad ogni comando è associato un codice esadecimale il quale varia uno o più bit della stringa inviata al display rispettivamente per l'operazione che si vuole svolgere. La stringa viene inviata subito dopo l'invocazione di un comando sia per modificare impostazioni al display sia per l'invio di un testo.

Una volta semplificata la programmazione, si è scritto il codice per la scrittura dei dati relativi al valore digitale acquisito e alla potenza percentuale erogata dal microcontrollore.

```
211     if (refresh <= 0)
212     {
213
214     //parte display
215
216     //cancella i dati precedenti a schermo
217      HD44780_Clear();
```

```
218     //imposta il cursore sul display dove scrivere,se non e' abilitato e'
    invisibile (colonna , riga)
219     HD44780_SetCursor(0,0);
220     //Stampa una stringa sul displey da dove inizia il cursore, la stringa e
    ' convertita automaticamente
221     HD44780_PrintStr("ADC 16bit:");
222     HD44780_SetCursor(0,1);
223     HD44780_PrintStr("Power:   % (   )");
224     itoa(PWMVal, out, 10);
225     HD44780_SetCursor(6,1);
226     HD44780_PrintStr(out);
227     //funzione che converte i valori interi in array contenenti i caratteri
    corrispondenti (per display)
228     itoa(Sens, out, 10);
229     HD44780_SetCursor(12,0);
230     //Stampa un'array che corrisponde ad una stringa (non stampa valori int,
     uint, double, float ecc...)
231     HD44780_PrintStr(out);
232     itoa(PWM1, out, 10);
233     HD44780_SetCursor(12,1);
234     HD44780_PrintStr(out);
243
244     refresh = 25;
251    //con questo 'else' il display non viene aggiornato
252    else
253    {
254     refresh--;
255    }
```

Codice 3.3: Codice per il controllo del display

Come si può notare questo frammento di codice è racchiuso dentro a una condizione che controlla la variabile `refresh`. Fin dalle prime versioni del codice si è notato come il display abbia una latenza dal momento del ricevimento del testo all'effettiva illustrazione di questo a schermo. Se l'esecuzione del codice è troppo veloce il display non ha il tempo di mostrare il testo a schermo. Inizialmente si è pensato ad un semplice `delay` ma ciò avrebbe rallentato anche il resto del codice compresi ADC e macchina a stati. La soluzione si è quindi trovata inserendo la variabile `refresh` che funge da contatore, permettendo al codice di lavorare ad un'elevata velocità. Il display viene aggiornato circa una volta al secondo.

Infine, per estetica, si è aggiunta una variabile contatore `cont` la quale dopo 1000 cicli spegne la retroilluminazione dello schermo.

Capitolo 4

Realizzazione e risultati

4.1 Circuito

Il circuito è stato realizzato su una tavoletta *breadboard*. Le tavolette breadboard si usano molto nei laboratori scolastici in quanto non richiedono la stagnatura dei componenti rendendo più veloce e versatile la realizzazione di piccoli progetti elettronici. Si è cercato di rendere ben visibili i collegamenti tra i componenti, anche per avere comodità nell'eseguire le misurazioni con gli strumenti di misura.

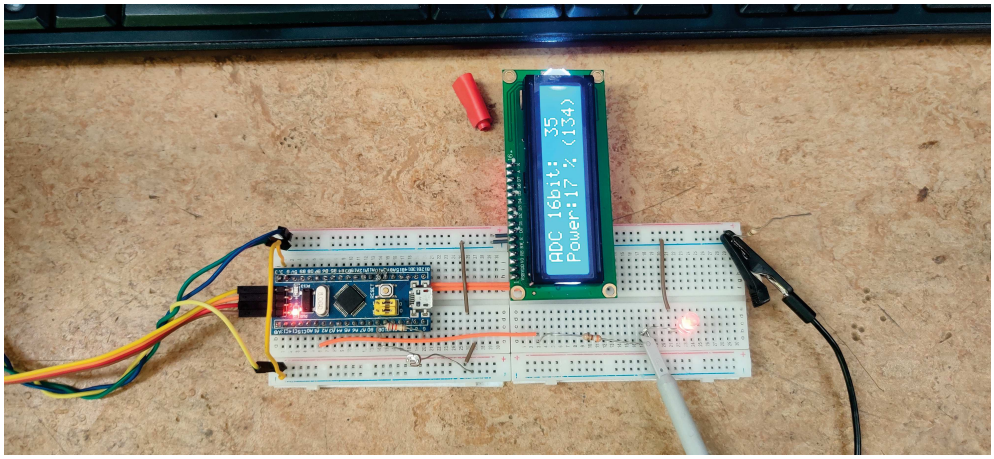


Figura 4.1: Realizzazione in Laboratorio

La scheda viene alimentata dal programmatore ST-Link V2 il quale è alimentato tramite USB da un computer. Si utilizzano tensioni di 3.3 V per alimentare microcontrollore e fotoresistenza e di 5 V per alimentare il display. Si sarebbe potuto alimentare il microcontrollore direttamente da USB e alimentare il resto dei componenti collegandoli ai pin interni alimentati ma, oltre ad essere superfluo, espone il PC al rischio di sovratensioni, nel caso di errori accidentali durante

le misurazioni. La fotoresistenza è collegata in configurazione “Pull-Down”. Il bus I^2C è ovviamente collegato nel modo più diretto possibile. L’uscita PWM è portata distante dal Microcontrollore in modo da sfruttare meglio lo spazio.

Il seguente schema è stato realizzato con Eagle 7.7.0 Circuit Designer.

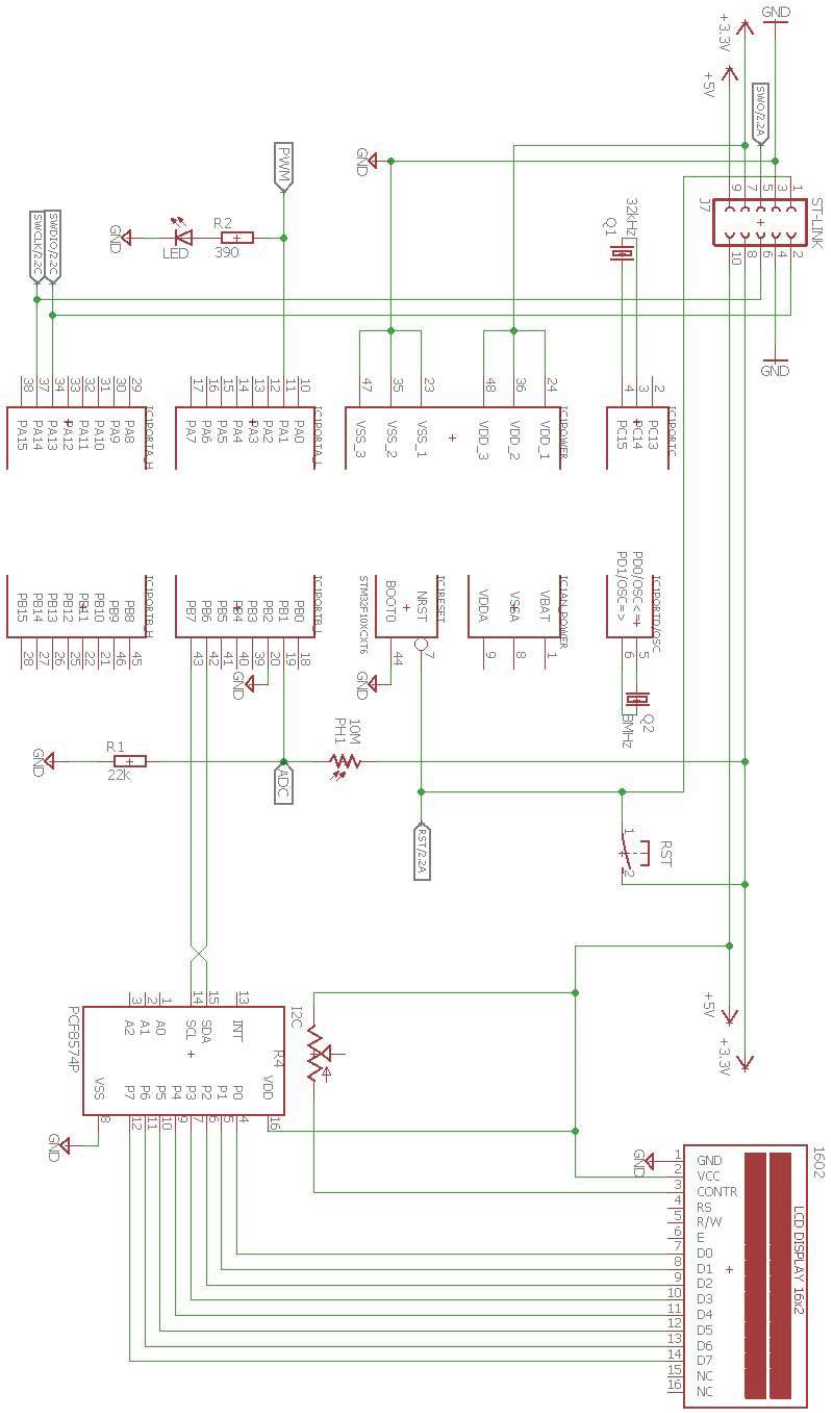


Figura 4.2: Schema elettrico

4.2 Strumenti di misura

4.2.1 Multimetro

Il multimetro digitale è uno strumento di misura per le grandezze elettriche sempre presente nei laboratori di elettronica. Combina le funzionalità di più strumenti di misura come Voltmetro, Amperometro, Wattmetro, Ohmmetro e controllo di continuità per diodi o cortocircuiti. In particolare il modello utilizzato in laboratorio è l'Agilent U1241A. Verrà utilizzato principalmente come Ohmmetro per misurare i valori del fotoresistore, visto al paragrafo 2.3, esposto a diverse condizioni di luminosità. Sarà utile in generale per il controllo della correttezza dei collegamenti e delle tensioni della scheda.

Figura 4.3: Multimetro Agilent U1241A

Nel manuale istruzioni fornito da Agilent [11] si trovano le seguenti tabelle 4.1 e 4.2 indicanti la precisione dello strumento.

Tabella 6-1 Specifiche CC con precisione ± (% valore di lettura + N. di cifre meno significative)

Funzione	Portata	Risoluzione	Corrente di test/ Caduta di tensione	Precisione	
				U1241A	U1242A
Tensione	1000,0 mV	0,1 mV	-	0,09% + 5	
	10,000 V	0,001 V	-	0,09% + 2	
	100,00 V	0,01 V	-		
	1000,0 V	0,1 V	-	0,15% + 5	
Corrente	1000,0 µA	0,1 µA	< 0,06 V (50 Ω)	0,1%+3	
	10000 µA	1 µA	< 0,55 V (50 Ω)	0,1%+3	
	100,00 mA	0,01 mA	< 0,18 V (0,5 Ω)	0,2%+3	
	440,0 mA	0,1 mA	< 0,8 V (0,5 Ω)	0,5%+3	
	10,000 A	0,001 A	< 0,4 V (0,01 Ω)	0,6%+5	

Tabella 4.1: Precisione di Voltmetro ed Amperometro [11]

Tabella 6-3 Specifiche di resistenza con precisione ± (% valore di lettura + N. di cifre meno significative)

Funzione	Portata	Risoluzione	Corrente di test/ Caduta di tensione	Precisione
Resistenza	1000,0 Ω	0,1 Ω	0,5 mA	0,3% + 3
	10.000 kΩ	0,001 kΩ	50 µA	
	100,00 kΩ	0,01 kΩ	4,91 µA	
	1000,0 kΩ	0,1 kΩ	447 nA	
	10,000 MΩ	0,001 MΩ	112 nA	0,8% + 3
	100,00 MΩ	0,01 MΩ	112 nA	1,5% + 3

Tabella 4.2: Precisione dell'Ohmmetro [11]

4.2.2 Oscilloscopio

Keysight DSOX1102G è un oscilloscopio professionale e fa parte della famiglia Keysight InfiniiVision X1000. Dispone di due ingressi in grado di acquisire segnali periodici con frequenze fino a 100 MHz e di memorizzare fino a 2 GS/s. Oltre alle fondamentali impostazioni tipiche di un oscilloscopio, questo dispositivo è in grado di eseguire svariate misurazioni in tempo reale, misure picco-picco, calcolo di frequenza e periodo del segnale, calcolo del Duty-Cycle per segnali PWM nonché la misurazione di T_{rise}, T_{fall}, e altre misurazioni per segnali a gradino. Le misurazioni saranno fatte con una sonda 10:1 compatibile con l'oscilloscopio.

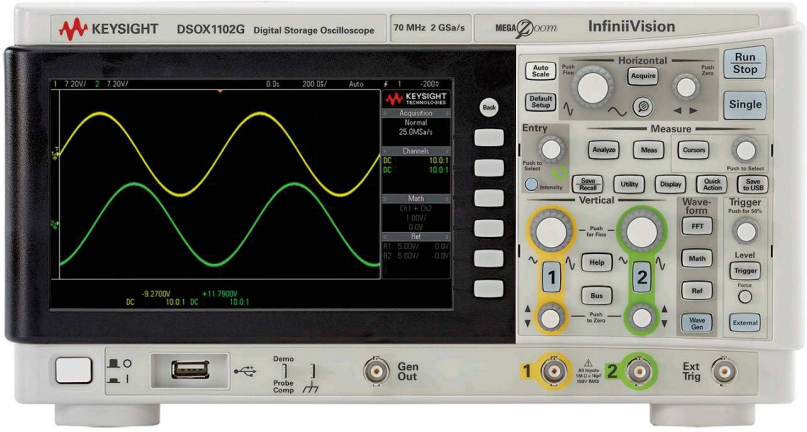


Figura 4.4: Oscilloscopio Keysight DSOX1102G [12]

Le tabelle relative alla sensibilità e ai tempi di risposta dello strumento si trovano nel relativo manuale istruzioni *InfiniiVision 1000 X-Series Oscilloscopes* [12] al paragrafo "*Performance Characteristics*".

Volendone citare alcuni di rilievo, lo strumento ha un range verticale (asse Voltmetrico) da 500 $\mu V/div$ a 10 V/div con una risoluzione di 8_{bit}, mentre l'asse dei tempi presenta un range da 5 ns/div a 50 s/div con una risoluzione di $2ps$.

4.3 Comportamento

Di seguito sono elencate le fasi del comportamento dell'ultima versione del codice caricato, ovvero dopo essere stato sottoposto a simulazioni e perfezionamenti. Si è ipotizzato un utilizzo del dispositivo simile ad una lampadina, ovvero il dispositivo verrà azionato da un interruttore a muro che fornisce e toglie alimentazione di rete all'intero circuito.

Ad ogni accensione la sequenza degli eventi è la seguente:

1. Inizialmente la scheda si trova in uno stato di assenza di alimentazione, simulando una lampadina spenta.

2. Una volta che viene data tensione al circuito, il microcontrollore ha bisogno di circa 1 secondo per avviarsi in cui non succede niente.

3. Successivamente inizia l'esecuzione del codice A caricato che, per prima cosa, configura le periferiche esterne ed instaura la comunicazione I^2C con il display. Per aggiungere un po' di estetica, il microcontrollore eroga potenza gradualmente per 0.5 secondi fino al raggiungimento della potenza massima. Questa parte di codice viene eseguita solo durante la prima accensione e non interferisce con la macchina a stati.

4. Dopo l'accensione il LED si trova alla massima potenza ($PWM_{set} = 1000$) e il controllore entra nel ciclo `while` principale. In questa fase si ripetono le operazioni principali del codice:

   (a) Acquisizione ADC e conversione del dato digitale.

   (b) Controllo lineare: viene calcolato il valore di PWM_{target} secondo la formula 3.4.

   (c) Macchina a stati: la macchina a stati decide lo stato corrente, ovvero se aumentare, diminuire o mantenere costante la variabile PWM_{set}.

   (d) Visualizzazione display: il display mostra a schermo i valori:

      - valore acquisito in formato decimale;

- valore di Potenza Percentuale e variabile PWM_{target} racchiusa tra parentesi.

(e) Estetica: viene decrementata una variabile che dopo circa 1000 secondi spegne il display.

4.3.1 Verifica dei segnali

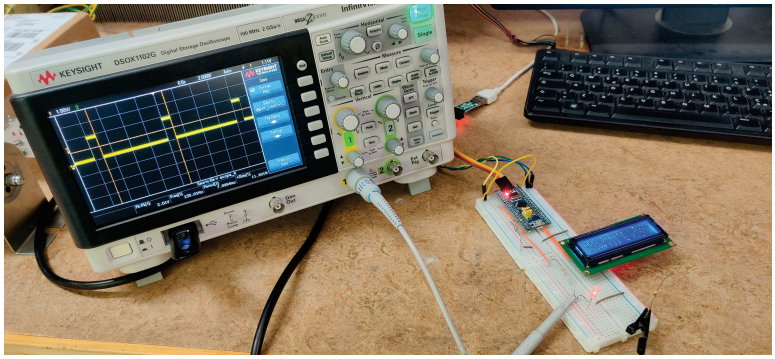


Figura 4.5: Misurazione con oscilloscopio in laboratorio

Per prima cosa si è voluto controllare la risposta della fotoresistenza a variazioni brusche di luminosità e visualizzare il valore di tensione all'ingresso analogico del microcontrollore.

È stato impostato un valore di misura per l'asse dei tempi il più elevato possibile. La figura 4.6 rappresenta quindi un intervallo di tempo di 5 secondi. Come si può vedere la tensione media massima raggiunta non supera i 3 V, evitando quindi sovratensioni in ingresso.

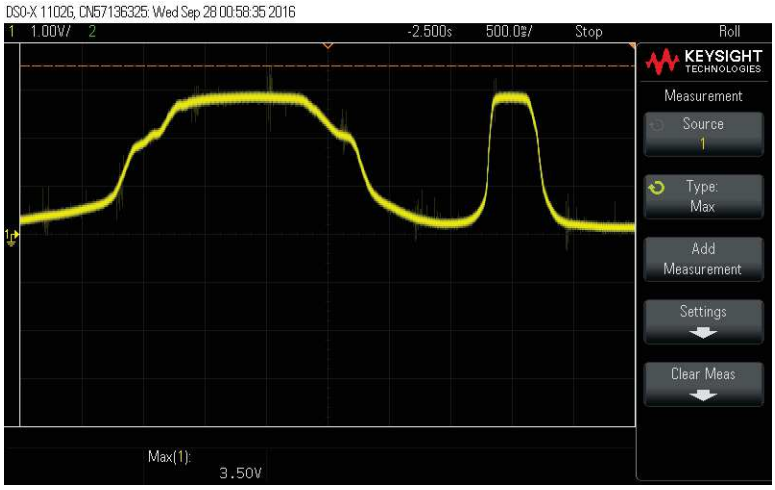
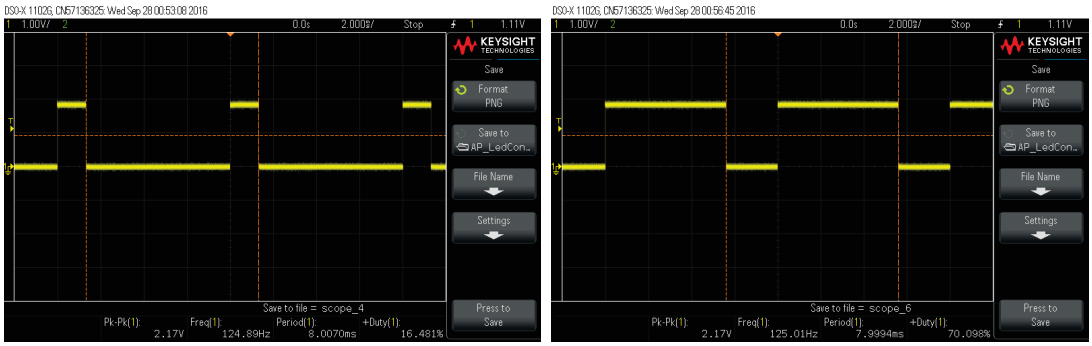


Figura 4.6: Visualizzazione della variazione di tensione in ingersso

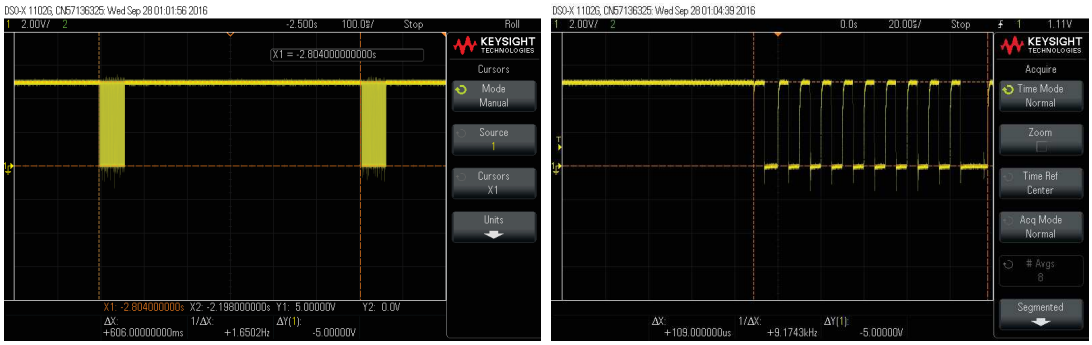
Successivamente si è controllata l'onda e la tensione in uscita. In base ai valori di PWM impostati ci si aspetta un'onda rettangolare simile a quella in figura 3.3. Si collega la sonda all'anodo del LED e si imposta nell'oscilloscopio l'asse dei tempi a 2 ms/div e l'asse delle ampiezze a 1 V/div.



(a) Duty Cycle ≈ 16% **(b)** Duty Cycle ≈ 70%

Figura 4.7: Onda modulata PWM

Infine si è osservato all'oscilloscopio il funzionamento del bus I^2C. Dato che il bus invia migliaia di bit al secondo, un'acquisizione normale dell'oscilloscopio non offriva una visualizzazione utile. Si è quindi utilizzata una visualizzazione singola.



(a) **(b)**

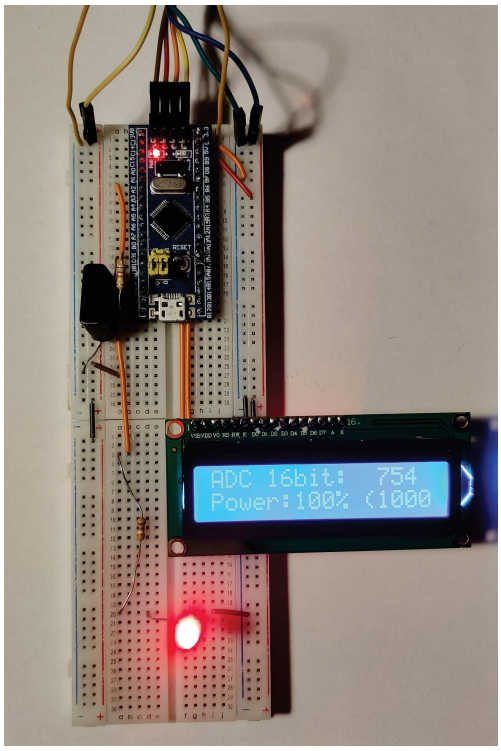
Figura 4.8: Frammento di stringa di comando I^2C

4.3.2 Analisi funzionamento macchina a stati

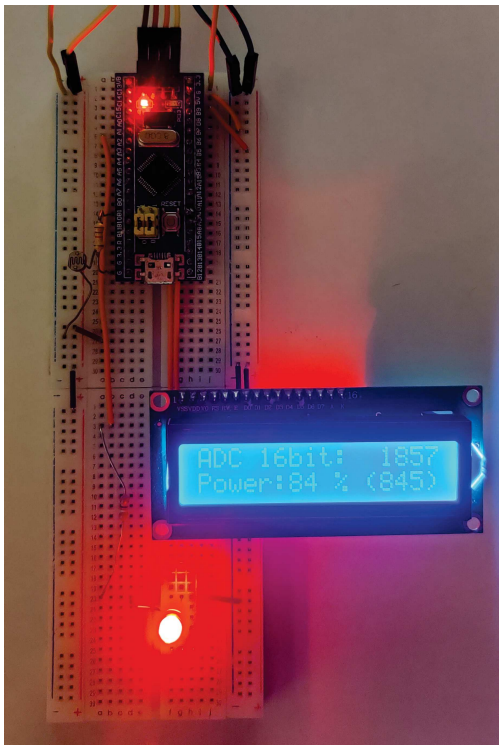
Una volta controllato il corretto funzionamento dei componenti hardware, si procede con una simulazione per controllare il funzionamento della macchina a stati. Durante le simulazioni sono state fatte modifiche al codice ogni qual volta il microcontrollore reagisse diversamente da quanto richiesto o per modificare i parametri di risposta.

Dopo una serie di versioni del codice si è riusciti ad ottenere un funzionamento equivalente al diagramma di stato 3.5. Successivamente sono state aggiunte alcune righe di codice atte a migliorare l'esperienza di utilizzo per l'utente finale, come per esempio un'accensione iniziale graduale molto rapida.

Di seguito alcune simulazioni di luce con relativa risposta. Non potendo illustrare un filmato in questa tesi, la potenza erogata al LED è in equilibrio con la potenza richiesta.

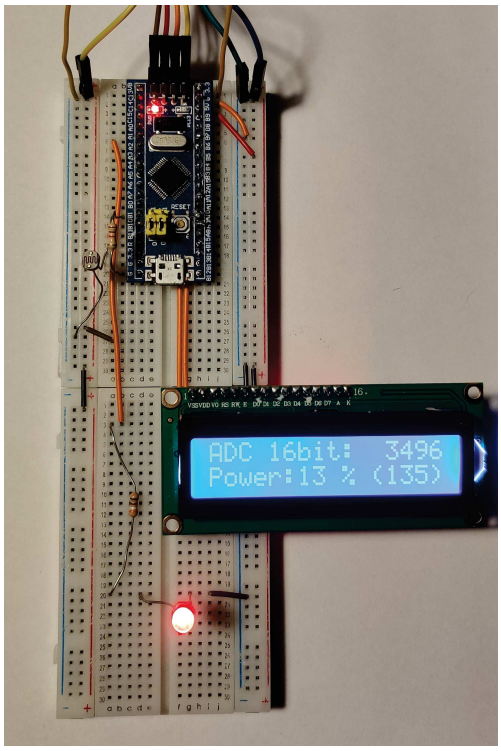


(a) Assenza di luce (LDR[3] oscurata)

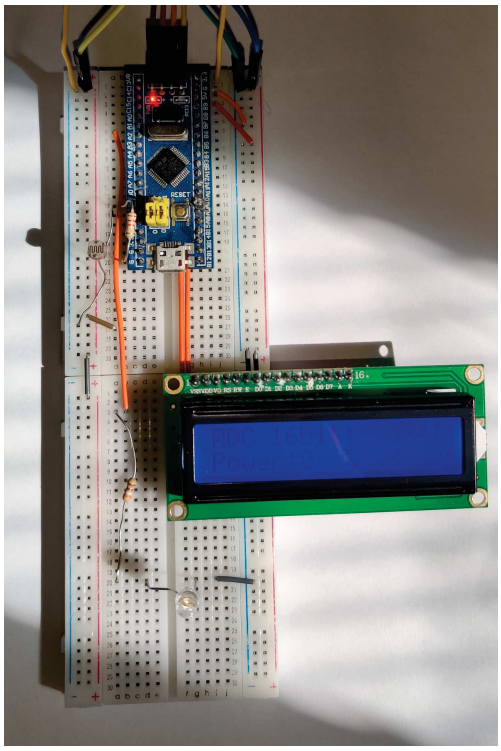


(b) Luce artificiale indiretta

[3]Light Dipendent Resistance, nome esteso inglese di Fotoresistenza.

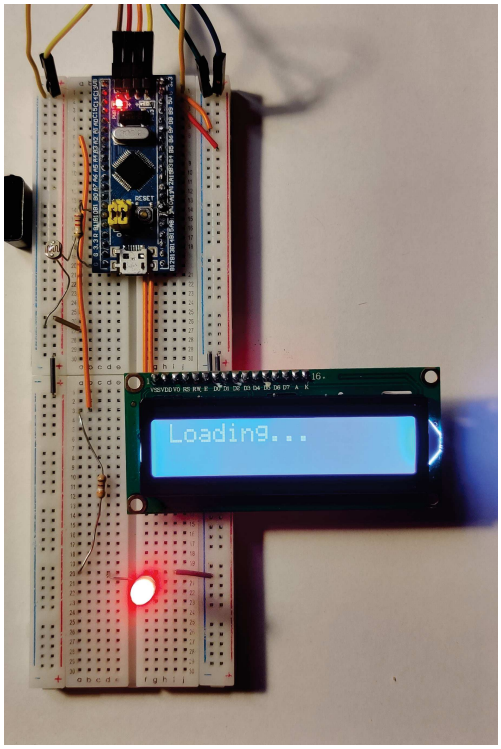


(c) Luce artificiale diretta

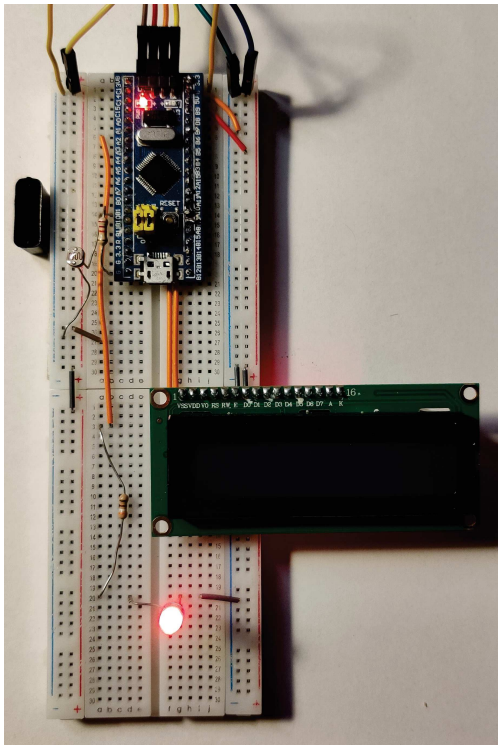


(d) Eccesso di luce

Figura 4.10: Esposizione a diverse intensità luminose.



(e) Schermata in accensione



(f) Spegnimento LCD

Figura 4.11: Estetica aggiuntiva nel comportamento.

Capitolo 5

Conclusioni

Volendo passare da un progetto di laboratorio ad un prodotto commerciabile, c'è molto spazio per miglioramenti e funzionalità aggiuntive. Una volta raggiunto l'obbiettivo principale sorgono infatti problemi inserendo l'ipotesi di un utilizzo intensivo e la possibilità di malfunzionamenti da parte del microcontrollore o dei dispositivi periferici. Si è notato che in generale nel caso si verifichi un malfunzionamento, la potenza viene portata a zero spegnendo la lampada. Ciò può essere un vantaggio dal punto di vista della sicurezza elettrica ma costringe l'utente a sostituire l'intero dispositivo illuminante, nonostante il problema sia di natura informatica o elettronica. Una più semplice soluzione consiste nell'utilizzo di una logica negativa realizzata con transistor PNP o mosfet P-MOS. In questo caso la sorgente rimarrebbe sempre accesa quando è presente tensione e il microcontrollore andrebbe ad agire limitando l'afflusso di corrente.

In ogni caso l'idea iniziale di questa tesi era di progettare un dispositivo in grado di controllare la potenza erogata da una sorgente luminosa, integrando un'illuminazione naturale con una artificiale riducendo così gli sprechi di energia elettrica. Grazie alla realizzazione del progetto si ritiene raggiunto tale obbiettivo.

Appendice A

Codice

Di seguito il codice integrale caricato nel microcontrollore.

```
/* Includes ------------------------------------------------------------------
    */
#include "main.h"

/* Private includes ----------------------------------------------------------
    */
/* USER CODE BEGIN Includes */

#include "liquidcrystal_i2c.h"
```

```
ADC_HandleTypeDef hadc1;

I2C_HandleTypeDef hi2c1;

TIM_HandleTypeDef htim2;

/* USER CODE BEGIN PV */

  //Duty Cicle targhet convertito dal valore rilevato
 uint16_t PWM1=0;
  //Duty Cicle in millesimi
 uint16_t PWMset=1000;
  //Configurazione ADC
 ADC_ChannelConfTypeDef sConfigPrivate = {0};
```

```
//parte display

  //Valore letto dall ADC in 16 bit
 uint16_t Sens;
  //Stringa da stampare
```

```
 char out[4];
  //Valore del PWM in percentuale
 uint16_t PWMVal;
  //variabile per cicli
 short  i=0;
  //Sostituisce un banale delay in modo che mentre il display mostra
    informazioni, il pwm lavora
 uint8_t refresh=0;

// altro

  //conteggio per lo standby dell'LDC
 uint16_t cont=1000;
```

```
  //parte display

   //estetica finche' il PWM aumenta da 0 a 1000
    HD44780_Init(2);
    HD44780_SetCursor(0,0);
    HD44780_PrintStr("Loading...");


   //parte PWM

    HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2);

   //estetica: invece di accendere bruscamente il led lo si accende
    gradualmente
    for(i=0; i<1000;i++)
    {
     __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2, i);

     HAL_Delay(0);
     //durata stimata ciclo for = 1 secondo
    }


  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
     //parte ACD

```

```
161      //Ad ogni ciclo inizializza la lettura dell'ingresso analogico
162      HAL_ADC_Start(&hadc1);
163      HAL_ADC_PollForConversion(&hadc1,1000) ;
164      //Legge il valore istantaneo dell'ADC in 16bit
165      Sens = HAL_ADC_GetValue(&hadc1) ;
166
167     //parte led PWM
168
169     //Se c'e' poca luce, potenza massima
170     if  ( Sens < 1500 )
171     {
172      PWM1 = 1000 ;
173     }
174     //Se c'e' sufficiente luce, spegni (oppure si imposta un valore minimo da
        mantenere)
175     else if ( Sens > 3800 )
176     {
177      PWM1 = 0 ;
178     }
179     //Controllo lineare della potenza del led, opportunamente rapportata
180     else
181     {
182      PWM1 = 1000 - (( Sens - 1500 )  /2.3 ) ;
183     }
184
185     //Macchina a Stati
186
187      //caso target minimo e potenza minima = MANTIENI SPENTO
188      if   ( PWMset < 10 && PWM1 < 25 ) {PWMset = 0;}
189      //caso target massimo e potenza massima = MANTIENI MASSIMO
190      else if ( PWMset > 990 && PWM1 > 950 ) {PWMset = 1000;}
191      //caso target maggiore e potenza minore del 50% = AUMENTA VELOCE
192      else if ( PWM1 > PWMset && PWMset < 500 )  { PWMset = PWMset + 5; }
193      //caso target maggiore = AUMENTA LENTO
194      else if ( PWM1 > PWMset )  { PWMset = PWMset + 1;}
195      //caso target minore = DIMINUISCI LENTO
196      else if ( PWM1 < PWMset )  { PWMset = PWMset - 1;}
197      //caso target uguale = MANTIENI
198      else {}
199
200    //calcolo valore PWM in percentuale (per display)
201     PWMVal = ( PWMset * 100 ) /1000;
202    //PWM
203     __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_2, PWMset);
```

```
204
205     HAL_Delay(20);
206     //Valore di delay che regola gli intervalli di variazione del PWM
207    //Questo delay moltiplicato per la variabile 'refresh' definisce
208    //il delay totale per l'aggiornamento dei valori a display
209
210
211     if (refresh <= 0)
212     {
213
214     //parte display
215
216     //cancella i dati precedenti a schermo
217      HD44780_Clear();
218      //imposta il cursore sul display dove scrivere,se non e' abilitato e'
    invisibile (colonna , riga)
219      HD44780_SetCursor(0,0);
220      //Stampa una stringa sul displey da dove inizia il cursore, la stringa e
    ' convertita automaticamente
221      HD44780_PrintStr("ADC 16bit:");
222      HD44780_SetCursor(0,1);
223      HD44780_PrintStr("Power:   % (   )");
224      itoa(PWMVal, out, 10);
225      HD44780_SetCursor(6,1);
226      HD44780_PrintStr(out);
227      //funzione che converte i valori interi in array contenenti i caratteri
    corrispondenti (per display)
228      itoa(Sens, out, 10);
229      HD44780_SetCursor(12,0);
230      //Stampa un'array che corrisponde ad una stringa (non stampa valori int,
     uint, double, float ecc...)
231      HD44780_PrintStr(out);
232      itoa(PWM1, out, 10);
233      HD44780_SetCursor(12,1);
234      HD44780_PrintStr(out);
235
236      //altro
237     // estetica: dopo "cont" cicli il display si spegne
238      if  (cont==0)
239      {
240       HD44780_NoBacklight();
241      }
242      else cont--;
243
```

```
244      refresh = 25;
```

```
250     }
251     //con questo 'else' il display non viene aggiornato
252     else
253     {
254      refresh--;
255     }
```

Codice A.1: Codice sorgente completo

Elenco delle figure

1.1 Esempio di illuminazione artificiale diurna 2
1.2 Esempio controllo intelligente della luce 3

2.1 STM32F103C8T6 “Blue Pill” 5
2.2 Schema Pin Out STM32F103C8T6 6
2.3 Caratteristica e dati dell’ADC1 7
2.4 Schema, SDA e Clock I^2C 8
2.5 Esempio di trasmissione I^2C 9
2.6 Esempi di modulazione PWM 11
2.7 ST Link V2 11
2.8 Funzionamento dei cristalli 12
2.9 Specifiche display 1602 13
2.10 Interfaccia PCF8574 (sopra) e display 1602 (sotto) 14
2.11 Simbolo fotoresistore 15
2.12 Diodo LED 16
2.13 Schema tipico diodo LED 16

3.1 Configurazione delle periferiche 18
3.2 Configurazione del clock 18
3.3 Forma d’onda attesa con $T_{on} = 50\%$ 19
3.4 Transcaratteristica Ingresso-Uscita 21
3.5 Diagramma di stato 23

4.1 Realizzazione in Laboratorio 27
4.2 Schema elettrico 28
4.3 Multimetro Agilent U1241A 29
4.4 Oscilloscopio Keysight DSOX1102G 30
4.5 Misurazione con oscilloscopio in laboratorio 32
4.6 Visualizzazione della variazione di tensione in ingersso 32

4.7 Onda modulata PWM 33
4.8 Frammento di stringa di comando I^2C 33
4.10 Esposizione a diverse intensità luminose. 35
4.11 Estetica aggiuntiva nel comportamento. 35

Elenco delle tabelle

2.1 Caratteristica e dati dell'ADC2 7
2.2 Caratteristica I^2C 9
2.3 Valori tipici fotoresistore 15
2.4 Valori del fotoresistore utilizzato 15

3.1 Valori digitali misurati dal microcontrollore 20
3.2 Ingressi e uscite della Macchina di Mealy 22
3.3 Tabella degli stati 23

4.1 Precisione di Voltmetro ed Amperometro 29
4.2 Precisione dell'Ohmmetro 30

Elenco dei codici

3.1 Casistica della transcaratteristica 21
3.2 Casistica della macchina a stati 24
3.3 Codice per il controllo del display 25
A.1 Codice sorgente completo 43

Bibliografia

[1] Governo italiano. *Decreto Legislativo 81/2008*. Vol. Allegato IV. 81. 2008, p. 15 (cit. alle pp. 1, 2).

[2] Ente nazionale italiano di unificazione (UNI). *UNI EN 12464-1*. 12464. 2021 (cit. a p. 2).

[3] Director Sajol Ghoshal. «LMigrating from "power-centric" to "space-sensing" daylighting systems». In: *Power Systems Design* (2013). URL: https://www.powersystemsdesign.com/articles/migrating-from-power-centric-to-space-sensing-daylighting-systems/36/5528 (cit. a p. 3).

[4] Rasmus Friis Kjeldsen. *THE GENERIC STM32F103 PINOUT DIAGRAM*. URL: https://commons.wikimedia.org/wiki/File:Stm32f103_pinout_diagram.png (cit. a p. 6).

[5] STMicroelettronics. *Mainstream Performance line, Arm Cortex-M3 MCU with 64 Kbytes of Flash memory, 72 MHz CPU, motor control, USB and CAN*. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html (cit. alle pp. 7–9, 13).

[6] Simone Buso. *Microcontrollori e DSP*. Cap. Comunicazione seriale, p. 43 (cit. alle pp. 8, 9).

[7] Engicam. *Introduzione alla tecnologia LCD e utilizzo con il modulo Engicam GEA M6425*. URL: https://it.emcelettronica.com/introduzione-alla-tecnologia-lcd-e-utilizzo-con-modulo-engicam-gea-m6425 (cit. a p. 12).

[8] STMicroelettronics. *STM32CubeIDE user guide*. URL: https://www.st.com/resource/en/user_manual/um2609-stm32cubeide-user-guide-stmicroelectronics.pdf (cit. a p. 17).

[9] Wikipedia. *Macchina di Mealy*. URL: https://it.wikipedia.org/wiki/Macchina_di_Mealy (cit. a p. 22).

[10] Wikipedia. *Macchina di Moore*. URL: https://it.wikipedia.org/wiki/Macchina_di_Moore (cit. a p. 22).

[11] Agilent Technologies. *Agilent U1241A e U1242A Multimetri digitali palmari*. URL: https://it.rs-online.com/web/p/multimetri/0501348 (cit. alle pp. 29, 30).

[12] Keysight Technologiess. *InfiniiVision 1000 X-Series Oscilloscopes*. URL: https://www.keysight.com/us/en/support/DSOX1102G/oscilloscope-70-100-mhz-2-analog-channels.html (cit. a p. 30).