UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

SISTEMA VLC CON MODULAZIONE AD AMPIEZZA D'IMPULSO BASATO SU ARDUINO DUE

Relatore:

Prof. Stefano Tomasin

Laureando:

Alessandro Disarò

ANNO ACCADEMICO: 2023/2024

Data di laurea: 12/03/2024

Ai miei nonni:
Aurelio, Gianni,
Renata e Sandra.

Indice

Sommario

Scopo di questo lavoro è stata la realizzazione di un semplice sistema VLC, ovvero un sistema di comunicazione basato sulla luce visibile, a basso costo, utilizzando una scheda Arduino Due. Come modulazione è stata scelta la PAM (modulazione ad ampiezza d'impulso), una delle più utilizzate per i sistemi VLC. Nell'elaborato vengono presentati i passaggi che hanno portato al completamento del sistema, dalla scelta dei componenti fino all'implementazione della trasmissione di messaggi. I test effettuati hanno evidenziato il corretto funzionamento dei dispositivi costruiti, nonostante le prestazioni siano limitate dalle capacità della strumentazione impiegata.

Introduzione

La sesta generazione della telefonia mobile (6G), la cui distribuzione commerciale è prevista per il 2030, dovrà necessariamente introdurre nuove tecnologie a supporto di quelle attualmente utilizzate, per sostenere il crescente traffico di rete a livello globale. Diverse ricerche si stanno concentrando negli ultimi anni sulla comunicazione attraverso l'uso della luce visibile, per comprendere come possa essere sfruttata nelle reti 6G.

Questa tesi sperimentale si pone come obiettivo la costruzione di un semplice sistema VLC usando una scheda Arduino, una piattaforma hardware open source sviluppata all'*Interaction Design Institute* di Ivrea (TO) nel 2005. Nelle fasi di progettazione e sviluppo, è stato usato un approccio misto, comprendente analisi teoriche, simulazioni con programmi appositi e test reali sui dispositivi.

Nel Capitolo 1 viene presentata la VLC, con un riassunto della sua evoluzione nel tempo, le caratteristiche principali e gli utilizzi attuali e futuri. Tra i diversi studi sull'argomento, viene approfondito il progetto OpenVLC, che è stato di ispirazione per il lavoro svolto. Il Capitolo 2 è dedicato alla parte hardware del sistema realizzato, includendo gli schemi dei circuiti, i componenti usati e i motivi per cui sono stati selezionati. Il Capitolo 3 illustra invece i diversi passaggi che compongono la trasmissione di un messaggio, in particolare la sincronizzazione e le fasi di codifica e decodifica. Nel Capitolo 4 vengono esposti i risultati dei test effettuati sul sistema per verificarne il livello di funzionamento, dedicando le ulti-

me due sezioni ai possibili sviluppi futuri. Infine, nel Capitolo 5 vengono riassunti i punti salienti delle varie fasi di lavoro, con le considerazioni conclusive su quanto realizzato.

Si segnala che, nel periodo di svolgimento di questo progetto, una ricerca simile (con l'uso di una modulazione differente) è stata iniziata dal collega Alessandro Visentin, con cui c'è stato uno scambio di materiale per quanto riguarda i circuiti elettrici (come indicato nel Capitolo 2 in cui gli stessi vengono presentati) e un costruttivo confronto sull'approccio alla sincronizzazione.

Capitolo 1

Visible Light Communication

La *VLC (Visible Light Communication)* è una tecnica di comunicazione che sfrutta la luce visibile per trasmettere messaggi. Questo primo capitolo presenta una visione generale sulla tecnologia VLC, dalle origini fino ai progetti e agli utilizzi attuali.

1.1 Origini ed evoluzione nel tempo

La VLC è una tecnica con origini molto antiche, basti pensare all'accensione di fuochi in posizioni strategiche per comunicare gli avvistamenti di truppe nemiche. Il primo sistema per trasmettere messaggi non prestabiliti è descritto dallo storico greco Polibio (II secolo a.C.) nelle Storie: il numero di torce alzate ai due lati di una torretta forniva le coordinate su una griglia della lettera che si voleva inviare, con segnalazioni speciali per l'inizio e la fine della comunicazione [1].

Nel 1880 Alexander Graham Bell, l'inventore del telefono, inviò il primo messaggio con il suo *fotofono* su una distanza di circa 200 metri. Il fotofono modulava la luce solare attraverso uno specchio vibrante e, collegando alle estremità del sistema un telefono, permetteva a due persone di parlare senza la necessità di con-

nessioni fisiche. Era sufficiente tuttavia qualche nube per rendere l'apparecchio inaffidabile, motivo per cui le ricerche non vennero approfondite [2].

Ulteriori passi in avanti vennero fatti da F. R. Gfeller e G. Bapst che nel 1979 realizzarono un sistema di comunicazione senza fili usando diodi a emissione di luce infrarossa *(LED, Light Emitting Diodes)* [3], aprendo la strada alla ricerca sulle possibilità d'utilizzo dei LED, anche a luce visibile, nelle comunicazioni. Ad esempio, il progetto *RONJA (Reasonable Optical Near Joint Access)*, rilasciato nel 2001, è in grado di trasmettere dati con una velocità di 10 Mbps a una distanza di 1.4 km [4].

Guardando al futuro, ci si aspetta che la VLC possa entrare a far parte di una rete eterogenea con altri strumenti di comunicazione, per supportare lo sviluppo delle tecnologie 6G [5].

1.2 Caratteristiche e applicazioni

Uno dei principali punti di forza della VLC è l'ampiezza dello spettro della luce visibile, circa mille volte maggiore rispetto alle radiofrequenze, il cui spettro è oltretutto sempre più congestionato vista la grande quantità di scopi per cui sono utilizzate. Le frequenze della luce visibile vanno dai 400 THz ai 780 THz, permettendo una comunicazione ad alta velocità. A differenza di altre onde elettromagnetiche, la luce può essere meglio confinata riducendo così l'interferenza, e può essere utilizzata come mezzo trasmissivo in ambienti come aerei e ospedali, in cui tecnologie basate sulle radiofrequenze sono vietate per non disturbare le strumentazioni. Inoltre, i sistemi VLC che impiegano LED come fonte luminosa sono compatti, semplici da implementare, economici ed efficienti dal punto di vista energetico. Un altro vantaggio dei LED è il loro essere innocui per la salute a differenza di laser, pericolosi per gli occhi, e radiofrequenze, il cui impatto sul-

la salute è ancora oggetto di indagine [6]. La VLC, infine, consente trasmissioni più difficili da intercettare, dato che la luce non è in grado di attraversare oggetti opachi e la diffusione dell'informazione può dunque essere limitata ad uno spazio chiuso. Questa caratteristica, tuttavia, rappresenta anche la principale debolezza della VLC, impedendo comunicazioni in caso di ostacoli fisici tra i due dispositivi.

Gli ambiti in cui la VLC può essere utilizzata, oltre ai già citati casi in cui non possono essere usate radiofrequenze, sono molto vari, vista l'enorme diffusione di LED che potrebbero potenzialmente trasmettere informazioni. Una possibile applicazione sono le comunicazioni punto-a-punto tra dispositivi che devono scambiare dati, o l'invio di comandi nella domotica. Negli ambienti esterni, la VLC può essere sfruttata per trasmettere annunci in tempo reale sulle reti stradali, tramite semafori e lampioni. Ulteriori possibilità sono l'invio di informazioni ai passeggeri attraverso gli schermi LED di stazioni e aeroporti; oppure la diffusione di pubblicità o della descrizione di un'opera, usando rispettivamente gli impianti di illuminazione di attività commerciali e musei. Un ultimo impiego interessante della VLC è la localizzazione di dispositivi all'interno di ambienti chiusi, ad esempio per la navigazione autonoma di macchinari industriali che non possono fare affidamento sul GPS [7, pp. 4-6].

1.3 OpenVLC

OpenVLC è una piattaforma di comunicazione basata sulla luce visibile sviluppata all'*IMDEA Networks Institute* di Madrid (Spagna). Si tratta di un progetto open source, il cui obiettivo dichiarato è muovere i primi passi verso la creazione di una comunità di ricerca libera sulla VLC a basso costo [8]. I dispositivi realizzati possono lavorare sia come trasmettitori che come ricevitori e i componenti sono saldati su un circuito stampato, connesso ad una scheda *BeagleBone Black* [9].

Al momento della scrittura di questo elaborato, l'ultima versione ufficiale è la 1.3 (Figura 1.1), che può raggiungere una velocità di trasmissione di 400 Kbps su una distanza superiore ai 3 metri.

Il progetto OpenVLC è stato la principale fonte di ispirazione per questa tesi, che è nata dalla volontà di provare ad implementare un sistema simile usando una scheda differente, con prestazioni decisamente inferiori rispetto alla BeagleBone Black. Inoltre, le informazioni disponibili su OpenVLC sono state studiate per essere usate come base nella progettazione del sistema realizzato.

Figura 1.1: OpenVLC 1.3 (Fonte: www.openvlc.org)

Capitolo 2

Struttura fisica del sistema

Il sistema realizzato è costituito da un trasmettitore e da un ricevitore, a loro volta composti ciascuno da una scheda elettronica dotata di microcontrollore e da un circuito elettrico. In questo capitolo verranno illustrati i vari componenti utilizzati, con un'attenzione particolare alle caratteristiche che li rendono adatti alla realizzazione di un sistema VLC.

I circuiti sono stati progettati a partire da quelli del progetto OpenVLC, cercando di ottenere una semplificazione facile da implementare ed economica, ma allo stesso tempo precisa e ben funzionante.[1] Prima di essere realizzati fisicamente assemblando i componenti su delle *breadboard* (tavolette di montaggio forate con saldature predisposte al loro interno), i circuiti sono stati simulati con *LTspice XVII* per verificare il loro comportamento prima di acquistare i materiali e assicurarsi che le correnti necessarie non fossero troppo elevate per la scheda elettronica.

[1] Si ringrazia il collega Alessandro Visentin per l'elenco dei componenti equivalenti a quelli del progetto OpenVLC in quanto gli originali non erano più disponibili. I componenti utilizzati non sempre corrispondono a quelli indicati a causa delle semplificazioni rispetto al circuito originale.

2.1 Arduino Due

Sia il trasmettiore che il ricevitore hanno come elemento principale una scheda Arduino Due (Figura 2.1). Volendo utilizzare una scheda della famiglia Arduino, la scelta è ricaduta su questa versione poiché le sue caratteristiche [10] la rendono la più adatta alla gestione di segnali.

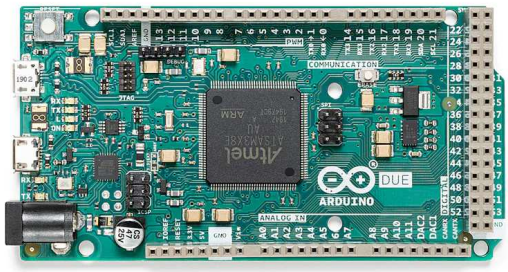

Figura 2.1: Arduino Due (Fonte: store.arduino.cc)

Nonostante Arduino Due non sia progettato in modo specifico per lavorare con i segnali [2], dispone di un processore Atmel SAM3X8E ARM Cortex-M3 a 32 bit con clock di 84 MHz, che permette di eseguire i calcoli necessari in tempi nell'ordine delle centinaia di microsecondi.

La scheda, che può essere facilmente programmata con una versione semplificata di C++, è dotata di pin di input e output (sia analogici che digitali), pin per l'alimentazione dei circuiti e due porte Micro USB per interfacciarsi ad altri dispositivi. Nel progetto realizzato, vengono utilizzati dei pin di input/output come collegamento tra Arduino e i circuiti elettrici e le porte USB per la comunicazione con i PC, come illustrato in Figura 2.2.

[2]Sono disponibili sul mercato schede di aziende differenti, come la Delfino™ LaunchPad™ di Texas Instruments che esegue le moltiplicazioni in virgola mobile 16 volte più rapidamente rispetto ad Arduino Due [11]

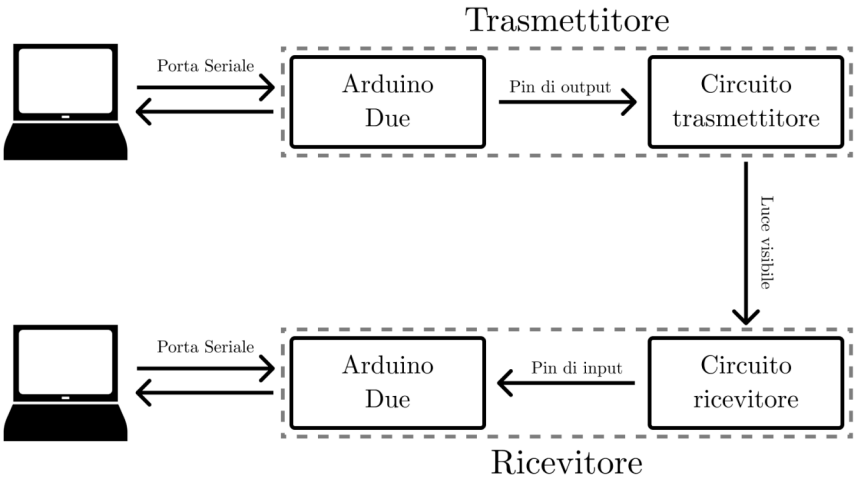

Figura 2.2: Comunicazione tra le diverse parti del progetto

Per il trasmettitore, viene utilizzato il convertitore digitale-analogico *(DAC, Digital to Analog Converter)* incluso nella scheda per impostare la tensione da fornire al LED. La presenza di un DAC integrato è il motivo principale che ha portato alla scelta di Arduino Due per il progetto, data la possibilità di variare la tensione in uscita senza dover ricorrere a componenti esterni. Il DAC è accessibile attraverso due canali indipendenti, ha una risoluzione di 12 bit (4096 livelli) e fornisce tensioni tra 0.55 V e 2.75 V.

Specularmente, il ricevitore fa uso del convertitore analogico-digitale *(ADC, Analog to Digital Converter)*, anch'esso integrato nella scheda, per la lettura dei valori forniti dal circuito di ricezione. L'ADC è accessibile tramite dodici canali indipendenti, ha una risoluzione di 12 bit (come il DAC) e permette di misurare tensioni comprese tra 0 V e 3.3 V.

2.2 Trasmettitore

Il circuito del trasmettitore (Figura 2.3) è piuttosto semplice, essendo composto solamente da un resistore e da un LED in serie (Tabella 2.1).

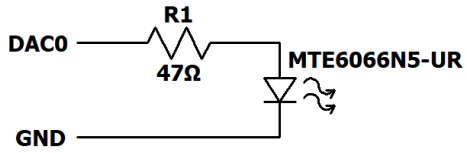

Figura 2.3: Schema elettrico del trasmettitore

Il polo positivo del LED è collegato al pin DAC0 della scheda Arduino tramite la resistenza R1, in modo da limitare la corrente che lo attraversa per evitare di danneggiarlo. Il polo negativo del LED, invece, è connesso a uno dei pin di messa a terra *(GND, ground)*. In questo modo, alla variazione della tensione fornita dal pin DAC0, regolata tramite il software, corrisponde una variazione della corrente che attraversa il LED con una conseguente modifica della sua luminosità.

Tabella 2.1: BOM trasmettitore

Indice	**Nome**	**Descrizione**	**Quantità**
1	A000062	Arduino Due ATSAM3X8E	1
2	MTE6066N5-UR	Led rosso 660 nm	1
3	R1	Resistore 47 Ω ± 1%	1

LED MTE6066N5-UR

Per il trasmettitore è stato scelto il LED MTE6066N5-UR [12], prodotto dall'azienda statunitense Marktech Optoelectronics. Si tratta di un LED rosso che ben si presta ad applicazioni ottiche come questa grazie alla sua elevata intensità luminosa, che permette di distinguere più facilmente il segnale trasmesso dal disturbo, e ad un tempo di salita e di discesa pari a 30 ns, che essendo nettamente inferiore rispetto al periodo di campionamento (ordine di grandezza dei microsecondi) consente di ottenere al ricevitore forme d'onda rettangolari in cui il fronte di salita appare verticale.

La scelta del colore rosso per il LED è dovuta all'impossibilità di trovare in commercio fotodiodi progettati appositamente per la luce visibile,[3] che ha reso necessario l'acquisto di un fotodiodo per luce infrarossa. La radiazione elettromagnetica infrarossa ha una lunghezza d'onda compresa tra 780 nm e 1 mm, superiore a quella della luce visibile, e il colore che più le si avvicina è proprio il rosso che ha una lunghezza d'onda di circa 700 nm [13]. Il LED utilizzato ha la massima emissione di energia luminosa in corrispondenza della lunghezza d'onda di 660 nm (Figura 2.4), mentre LED di colori differenti avrebbero avuto il picco di energia a lunghezze d'onda inferiori [14] (Figura 2.5) e quindi, a parità di energia luminosa emessa, il segnale ricevuto sarebbe stato minore.

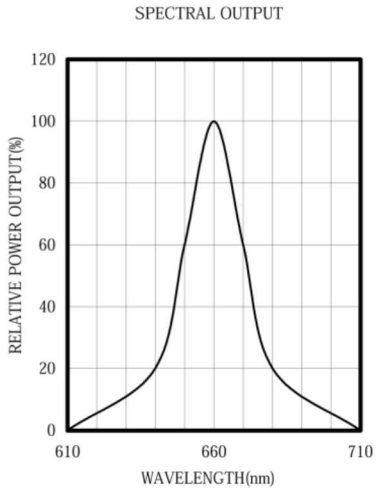

Figura 2.4: Emissione spettrale del LED utilizzato (Fonte: Marktech Optoelectronics)

La relazione tra tensione e corrente del LED in polarizzazione diretta può essere definita a tratti: la corrente è trascurabile fino a V_s, detta *tensione di soglia*, per poi crescere in modo esponenziale fino ad un valore massimo oltre il quale la giunzione p-n viene danneggiata. Come indicato dal produttore, la tensione di soglia del LED in uso è $V_s = 1.65\ V$ e la corrente massima tollerata è $I_{max} = 50\ mA$ (Figura 2.6).

[3] Al momento dell'acquisto dei componenti (marzo 2023) gli unici fotodiodi adatti con picco di rilevamento nella luce visibile avevano tempi di attesa troppo elevati.

Generalmente, nemmeno la relazione tra corrente e potenza ottica emessa è lineare e il modo migliore per descrivere questo rapporto è uno *sviluppo in serie di Volterra*, di cui è spesso complesso ricavare i coefficienti [7, pp. 22-23]. Per il LED utilizzato, tuttavia, questa funzione è pressoché lineare nell'intervallo operativo (Figura 2.7), dunque la relazione tra la tensione fornita dal pin DAC0 (ovvero l'input del circuito) e la potenza ottica emessa (l'output) è, a meno di un coefficiente, la stessa presente tra la tensione applicata ai capi del LED e la corrente che lo attraversa.

Uno dei più importanti parametri da considerare nell'utilizzo dei LED è la corrente di funzionamento, ovvero la corrente massima che dovrebbe attraversare il componente. Pur non essendo il limite assoluto, che è rappresentato da I_{max}, è bene non superare la corrente di funzionamento suggerita dal produttore per garantire la massima durata del LED. La resistenza R1 serve proprio per limitare la corrente nel circuito in modo da preservare il LED, e il suo valore può essere ricavato attraverso la *legge di Ohm*. Indicando con V_A la tensione massima fornita dal pin DAC0, V_F

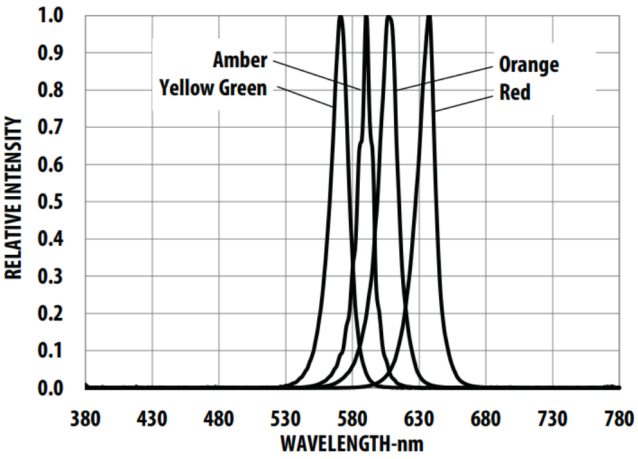

Figura 2.5: Emissioni spettrali di LED con colori differenti (Fonte: Broadcom Inc.)

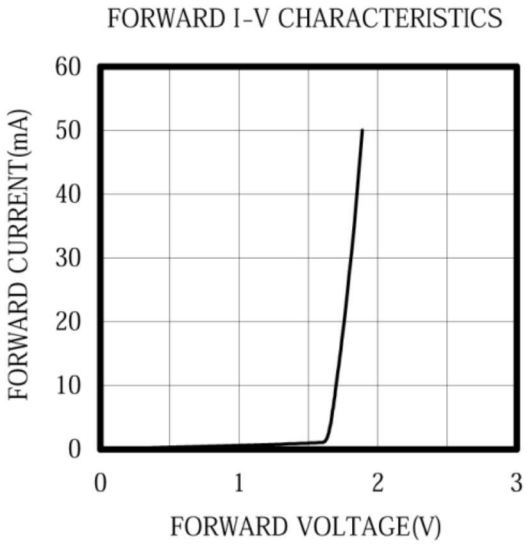

Figura 2.6: Relazione tra tensione e corrente del LED utilizzato (Fonte: Marktech Optoelectronics)

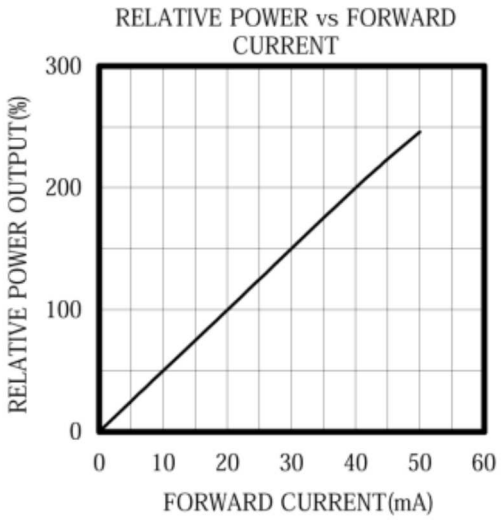

Figura 2.7: Relazione tra potenza relativa e corrente del LED utilizzato (Fonte: Marktech Optoelectronics)

la caduta di tensione tipica del LED e I_F la corrente di funzionamento, si ha

$$R1 = \frac{V_A - V_F}{I_F}. \tag{2.1}$$

Sostituendo nella (2.1) i valori corretti, ovvero $V_A = 2.75\ V$, $V_F = 1.8\ V$ e $I_F =$

20 mA, si ottiene $R1 = 47.5\ \Omega \approx 47\ \Omega$.

2.3 Ricevitore

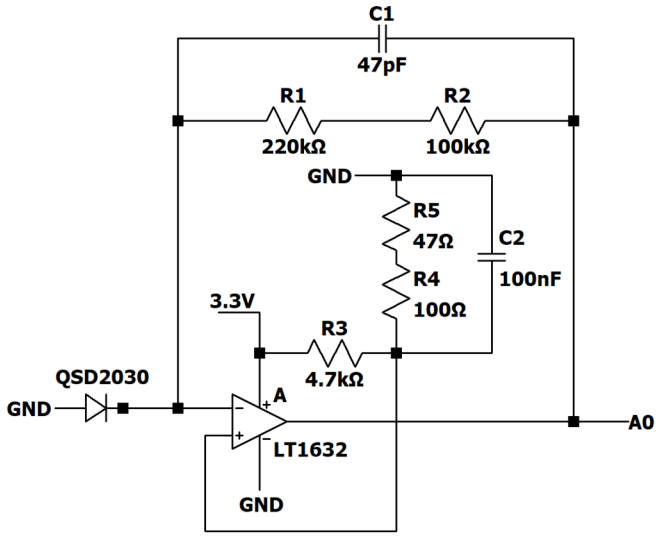

Figura 2.8: Schema elettrico del ricevitore

Il circuito del ricevitore (Figura 2.8) è decisamente più complesso rispetto a quello del trasmettitore, avendo al suo interno un fotodiodo, un amplificatore operazionale *(opamp)*, resistori e condensatori (Tabella 2.2).

Il fotodiodo è connesso ad una rete di amplificazione in transimpedenza che trasforma la corrente prodotta dalla giunzione p-n, troppo piccola per poter essere letta da Arduino, in una tensione di dimensioni rilevabili. Il circuito è interamente alimentato dalla scheda Arduino, con la tensione di 3.3 V che viene fornita dall'apposito pin. Per limitare la corrente attraverso i singoli pin della scheda, sono presenti tre collegamenti ad altrettanti pin GND invece che un'unica connessione. Il valore di tensione, amplificato dal circuito, viene letto da Arduino tramite il pin A0.

Tabella 2.2: BOM ricevitore

Indice	Nome	Descrizione	Quantità
1	A000062	Arduino Due ATSAM3X8E	1
2	QSD2030	Fotodiodo 880 nm	1
3	LT1632CN8	Opamp ad alta velocità	1
4	R1	Resistore 220 $k\Omega \pm 1\%$	1
5	R2	Resistore 100 $k\Omega \pm 1\%$	1
6	R3	Resistore 4.7 $k\Omega \pm 1\%$	1
7	R4	Resistore 100 $\Omega \pm 1\%$	1
8	R5	Resistore 47 $\Omega \pm 1\%$	1
9	C1	Condensatore ceramico 47 pF	1
10	C2	Condensatore ceramico 100 nF	1

Fotodiodo QSD2030

Il fotodiodo selezionato per il ricevitore è il QSD2030 [15], prodotto dalla società statunitense ON Semiconductor. Come accennato nella sezione 2.2, questo fotodiodo è progettato per rilevare la luce infrarossa; tuttavia, grazie ad un ampio intervallo di sensibilità, è in grado di rilevare lunghezze d'onda comprese tra 400 nm e 1100 nm, anche se la corrente generata diminuisce più ci si avvicina agli estremi. Il picco di ricezione in corrispondenza degli infrarossi permette alla luce del LED rosso di essere rilevata e, contemporaneamente, fa sì che la luce visibile di altri colori risulti attenuata, riducendo così parte dei disturbi provenienti dall'esterno. Per contro, il sistema risulta molto sensibile ai disturbi provocati da raggi infrarossi e non va dunque collocato in zone "a rischio" (ad esempio nelle vicinanze di un televisore, dato che rileverebbe il segnale del telecomando).

Un'altra caratteristica del fotodiodo che concorre alla protezione dai disturbi è il ristretto angolo di visione, pari a $\pm 20°$, che rende il sistema praticamente cieco alle fonti luminose non frontali. Questo, però, significa anche che il LED e il

fotodiodo dovranno essere il più possibile allineati per avere una buona ricezione.

Quando la giunzione p-n del fotodiodo viene illuminata, l'energia trasportata dai fotoni è convertita in una piccola corrente elettrica. Una volta fissata la tensione di polarizzazione inversa,[4] l'intensità della corrente dipende dall'energia luminosa assorbita con un andamento pressoché lineare per buona parte dell'intervallo di funzionamento (Figura 2.9).

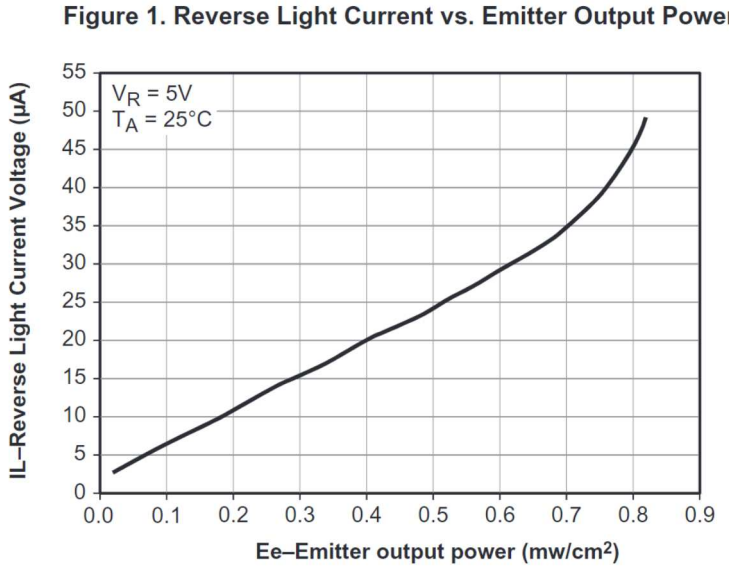

Figura 2.9: Corrente inversa al variare della potenza rilevata (Fonte: ON Semiconductor)

Anche quando non ci sono fotoni che raggiungono la giunzione p-n, è comunque presente una corrente di buio che però, avendo valore massimo di 1.2 nA, può essere trascurata.

Opamp LT1632CN8

Per amplificare il segnale di corrente del fotodiodo è stato utilizzato un opamp LT1632CN8 [16], fabbricato dall'azienda statunitense Analog Devices. Non sarebbe stato possibile infatti convertire la corrente del fotodiodo in tensione utilizzando

[4]Per funzionare correttamente come fotorilevatore, un fotodiodo deve avere la tensione maggiore in corrispondenza della zona N della giunzione p-n

un semplice carico resistivo, poiché si sarebbe venuto a formare un circuito RC con costante di tempo τ troppo elevata per la frequenza del segnale da leggere. Per questo motivo, è stato scelto un opamp con un elevato tasso di salita, per poter rispondere quasi istantaneamente alle variazioni del segnale da amplificare, e con un buon prodotto guadagno-larghezza di banda *(GBWP, gain-bandwidth product)*, in modo da avere un guadagno sufficiente anche a frequenze elevate.

Oltre ad essere prestazionale, questo amplificatore è adatto a circuiti con alimentazione a bassa tensione essendo *Rail-to-Rail*, potendo quindi sfruttare in uscita tutto l'intervallo di tensione compreso tra le due tensioni di alimentazione. Inoltre, la bassa corrente di alimentazione permette alla scheda Arduino di alimentare direttamente l'opamp, senza la necessità di inserire fonti esterne.

Nonostante l'opamp utilizzato disponga di un doppio stadio, ne è stato sfruttato solamente uno nella realizzazione del circuito. Pertanto, quello rappresentato in Figura 2.8 è lo stadio A dell'amplificatore.

Resistori e condensatori

La rete di amplificazione a supporto dell'opamp è stata realizzata a partire da un manuale di Texas Instruments [17], adattando i valori di partenza alle caratteristiche dei componenti utilizzati e gli obiettivi del circuito alle prestazioni attese del sistema. In particolare, i parametri da tenere in considerazione sono:

- corrente in ingresso compresa tra $I_{min} = 0\ A$ e $I_{max} = 10\ \mu A$;[5]
- tensione in uscita compresa tra $V_{min} = 0.1\ V$ e $V_{max} = 3.2\ V$;
- larghezza di banda $f_p = 10\ kHz$;
- tensioni di alimentazione $V_+ = 3.3\ V$ e $V_- = 0\ V$;

[5]Il basso valore di I_{max} rispetto al grafico in Figura 2.9 è dovuto alla bassa tensione di alimentazione del circuito.

- tensione di riferimento $V_{ref} = 0.1\ V$.

Il primo passo è il calcolo del resistore di guadagno R_G, dato dalla formula

$$R_G = \frac{V_{max} - V_{min}}{I_{max} - I_{min}} = \frac{3.2\ V\ -\ 0.1\ V}{10\ \mu A} = 310\ k\Omega. \tag{2.2}$$

Non disponendo di un resistore con tale valore, sono stati utilizzati due resistori $R_1 = 220\ k\Omega$ e $R_2 = 100\ k\Omega$ in serie, per ottenere un valore di R_G pari a 320 $k\Omega$, vicino a quello richiesto e semplice da implementare.

Successivamente, va calcolata la capacità del condensatore del filtro passa-basso per garantire la larghezza di banda richiesta con la disequazione

$$C_1 \leq \frac{1}{2\pi \cdot R_G \cdot f_p} = \frac{1}{2\pi \cdot 320\ k\Omega \cdot 10\ kHz} = 49.7\ pF. \tag{2.3}$$

È stato perciò utilizzato un condensatore $C_1 = 47\ pF$ per rispettare la diseguaglianza.

La tensione di riferimento può essere ottenuta a partire dalla tensione di alimentazione attraverso un partitore di tensione composto dai resistori R_A e R_B, legati dalla relazione

$$R_A = \frac{V_+ - V_{ref}}{V_{ref}} \cdot R_B = \frac{3.3\ V\ -\ 0.1\ V}{0.1\ V} \cdot R_B = 32 \cdot R_B. \tag{2.4}$$

I due resistori, tra quelli disponibili, che più si avvicinano a questa uguaglianza sono $R_A = R_3 = 4.7\ k\Omega$ e $R_B = R_4 + R_5 = 100\ \Omega + 47\ \Omega$. Inoltre, è possibile aggiungere un condensatore alla rete appena costruita per filtrare la tensione di riferimento, ripulendola da parte del rumore. La frequenza di taglio è inversamente proporzionale alla capacità del condensatore utilizzato, perciò per filtrare la maggior parte delle frequenze è stato scelto un condensatore $C_2 = 100\ nF$, il più grande tra quelli a disposizione.

Capitolo 3

Trasmissione di un messaggio

L'invio di un messaggio, nella sua forma più semplice, è composto dalla successione di due operazioni complementari: una modulazione, al trasmettitore, e una demodulazione, al ricevitore [18]. La modulazione consiste nella conversione di una sequenza di bit in un segnale fisico, in questo caso un segnale di tensione a cui corrisponde un segnale elettromagnetico emesso dal LED. La demodulazione è il processo opposto: il segnale elettromagnetico viene trasformato in un segnale di tensione dal fotodiodo e, successivamente, in una sequenza di bit che, in assenza di errori, sarà identica a quella di partenza.

Il terzo capitolo di questo elaborato affronta tutti gli aspetti della trasmissione di un messaggio, dalla definizione teorica della modulazione utilizzata fino alle strategie di implementazione. È inoltre trattata la sincronizzazione tra i due dispositivi del sistema, un passaggio imprescindibile quando non è presente un clock comune a scandire il ritmo delle operazioni.

3.1 Modulazione ad ampiezza di impulso

La modulazione ad ampiezza di impulso *(PAM, pulse amplitude modulation)* è una tra le modulazioni maggiormente utilizzate nella VLC, spesso insieme ad altre tecniche per migliorarne la robustezza [7, p. 17].

3.1.1 Definizione di M-PAM

La M-PAM trasmette l'informazione attraverso l'ampiezza del segnale: a ciascuno degli M valori di ampiezza usati corrisponde una parola di codice, ognuna delle quali è lunga $\log_2(M)$ bit (quando M è una potenza di 2 e la codifica è a lunghezza fissa) [19, pp. 307-309].

Sia $h(t)$ l'ampiezza dell'onda portante, dopo la modulazione le possibili onde avranno la forma

$$s_n(t) = \alpha_n h(t) \tag{3.1}$$

con $\alpha_n = 2n - 1 - M$, per $n = 1, 2, ..., M$.

La M-PAM è una modulazione unidimensionale la cui base ortonormale, detta E_h l'energia dell'onda portante, è il segnale

$$\phi_1(t) = \frac{h(t)}{\sqrt{E_h}}. \tag{3.2}$$

La costellazione è composta dai vettori $s_n = \alpha_n \sqrt{E_h}$ e le regioni di decisione sono delimitate dagli assi dei segmenti che uniscono due punti della costellazione consecutivi (Figura 3.1).

3.1.2 Descrizione della modulazione utilizzata

Generalmente il segnale processato dal ricevitore non è identico a quello prodotto dal trasmettitore, a causa delle attenuazioni introdotte dal canale (oltre che dalla

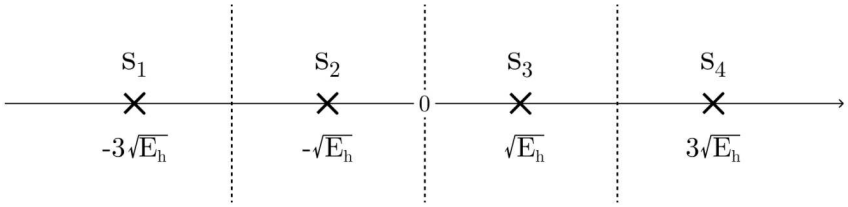

Figura 3.1: Costellazione e regioni di decisione per una 4-PAM

presenza del rumore). Tuttavia, la non linearità del LED, illustrata nel Capitolo 2, fa sì che ci sia un'asimmetria tra le forme d'onda al trasmettitore e al ricevitore. Se venisse usata una M-PAM al trasmettitore, scegliendo valori di tensione equidistanti tra loro, il segnale rilevato al ricevitore non sarebbe compatibile con lo stesso tipo di modulazione.

L'analisi presentata in questa sezione riguarda la modulazione ottenuta al ricevitore, mentre le tecniche utilizzate al trasmettitore verranno illustrate successivamente nel capitolo.

La modulazione scelta è una 4-PAM, che è stata ritenuta un buon compromesso tra la velocità di trasmissione e la flessibilità del sistema in termini di variazione di rapporto segnale-rumore *(SNR, Signal to Noise Ratio)*. Valori di M maggiori avrebbero incrementato la velocità di trasmissione (ad esempio con $M = 8$ la velocità sarebbe stata maggiore del 50%) ma allo stesso tempo avrebbero aumentato la probabilità di ottenere errori dovuti al rumore, a causa della minore dimensione delle regioni di decisione. Come onda portante è stata usata un'onda rettangolare, di larghezza pari al tempo di simbolo, essendo la più semplice da implementare con il DAC di Arduino Due.

Vale la pena infine fare alcune precisazioni sul canale di trasmissione. A livello teorico la M-PAM viene presentata considerando un canale AWGN *(Additive White Gaussian Noise, Rumore Gaussiano Bianco Additivo)* ma nella realtà le

cose non sono così semplici. In una comunicazione VLC dobbiamo prendere in considerazione diversi tipi di rumore, legati alle caratteristiche fisiche della luce, utilizzata come mezzo di trasporto delle informazioni, all'ambiente in cui avviene la trasmissione o agli strumenti elettronici usati. Il principale rumore dovuto al segnale luminoso è il rumore quantico, generato dal fatto che il LED emette fotoni a un tasso che è costante sul lungo periodo ma variabile se si considerano intervalli brevi. L'ambiente che circonda il sistema di comunicazione difficilmente è privo di fotoni in movimento (si pensi a una stanza illuminata da una lampadina oppure a una finestra da cui entrano dei raggi di sole) e questi possono colpire la superficie del fotodiodo, causando un rumore di fondo AWGN. Imputabili agli strumenti elettronici usati sono invece il rumore termico, presente in tutti i conduttori elettrici a temperature superiori allo zero assoluto; il rumore rosa, maggiore a basse frequenze; e le correnti di buio, piccole correnti presenti nel fotodiodo quando non riceve luce. Nella maggior parte delle applicazioni, tuttavia, il contributo al rumore fornito dall'ambiente è nettamente maggiore rispetto agli altri menzionati, pertanto il canale viene considerato AWGN come se fosse presente solo il rumore di fondo. Tale semplificazione è stata utilizzata anche in questo progetto, per facilitare l'uso delle conoscenze teoriche sulla M-PAM.

3.2 Sincronizzazione

In un sistema VLC, non è presente un clock condiviso che permetta a trasmettitore e ricevitore di inviare e leggere i dati simultaneamente. Questo significa che i due dispositivi cominciano a trasmettere e ricevere informazioni ad una frequenza regolare concordata, ma con istanti di inizio $t_{TX,0}$ e $t_{RX,0}$ differenti. Prima di avviare lo scambio di messaggi vero e proprio, il trasmettitore e il ricevitore vanno dunque sincronizzati, ricordando che un allineamento impreciso porterà inevitabilmente a

errori nella comunicazione [20].

3.2.1 Sincronizzazione di trama

Le prime cose che il ricevitore deve individuare in una trasmissione sono l'inizio e la fine di un messaggio, effettuando una *sincronizzazione di trama.*

Per quanto riguarda l'inizio del messaggio, una prima idea potrebbe essere la ricerca di un netto aumento dei valori in ingresso al ricevitore, come indicatore dell'accensione del LED. Questo metodo tuttavia è molto fragile poiché una qualsiasi interferenza luminosa sufficientemente intensa, ad esempio l'accensione di una lampada nella stanza, verrebbe interpretata come l'inizio del messaggio.

Un metodo più robusto consiste nel trasmettere, prima dei simboli corrispondenti ai bit, una sequenza concordata tra trasmettitore e ricevitore, in modo che l'arrivo di tale sequenza di sincronizzazione sia l'indicatore dell'inizio del messaggio. In questa fase il ricevitore non sa ancora a quali valori di tensione corrispondano i diversi simboli trasmessi, dunque non ha modo di capire se sta ricevendo la sequenza di sincronizzazione analizzando solo l'ampiezza del segnale. Ciò che viene sfruttato nella sincronizzazione di trama è la forma della sequenza di sincronizzazione, indipendentemente dall'altezza "assoluta" dei segnali rettangolari. Il grado di similitudine tra la forma di due segnali può essere stimato con la cross-correlazione, definita per due sequenze discrete $x[n]$ e $y[n]$ come la sequenza

$$R_{xy}[l] = \sum_{n=-\infty}^{\infty} x[n]y[n-l] \tag{3.3}$$

dove l indica lo spostamento della sequenza $y[n]$ rispetto all'altra che rimane fissa [21]. Per trovare l'istante corrispondente alla ricezione del primo simbolo della sequenza di sincronizzazione, è quindi sufficiente individuare il picco massimo nella cross-correlazione tra il segnale ricevuto e la sequenza di sincronizzazione stessa.

Teoricamente sarebbe possibile utilizzare una qualsiasi sequenza per la sincronizzazione di trama, tuttavia esistono sequenze particolari che, per la loro struttura, risultano più adatte allo scopo. Le sequenze PN *(pseudo-noise)* sono sequenze deterministiche con caratteristiche spettrali simile a quelle di un rumore bianco. Fanno parte di questa categoria le sequenze di lunghezza massima *(MLS, Maximal-Length Sequences)*, sequenze binarie facilmente ottenibili in modo ricorsivo. Con un registro a scorrimento di r bit è possibile generare una MLS di periodo $L = 2^r - 1$ attraverso operazioni di scorrimento e di somma binaria senza riporto. Ad esempio, con un registro a scorrimento di un byte ($r = 8$) si può produrre una MLS di lunghezza $L = 2^8 - 1 = 255$ a partire dagli otto valori iniziali $p(l-8), ..., p(l-1)$, secondo lo schema riportato in Figura 3.2.

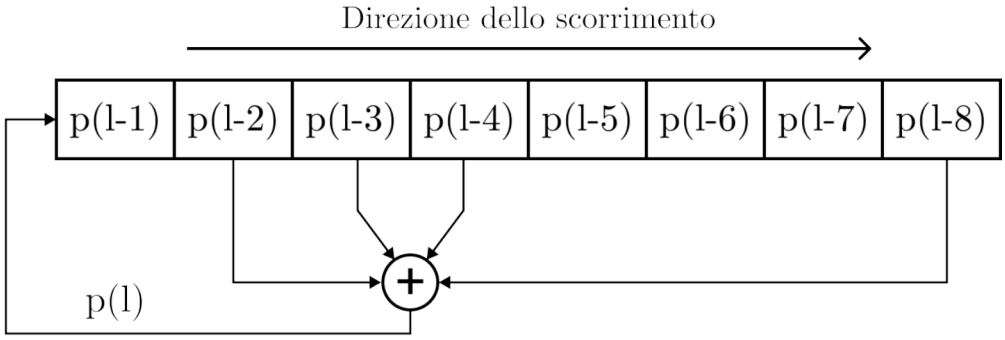

Figura 3.2: Generazione di una MLS con r=8

Le MLS sono candidate ideali al ruolo di sequenze di sincronizzazione avendo, dopo aver mappato i bit $\{0, 1\}$ nei valori $\{-1, 1\}$, un'autocorrelazione[1] pari a L quando l'allineamento è completo e -1 in tutti gli altri casi [22]. Tenendo conto che per la sincronizzazione viene usata la cross-correlazione tra la MLS e un segnale in cui ai lati della MLS si trovano valori nulli (corrispondenti allo 0 binario), si ottiene il grafico in Figura 3.3, in cui è evidente la presenza di un picco quando le sequenze sono allineate.

[1] L'autocorrelazione è definita come la cross-correlazione tra una sequenza e sé stessa.

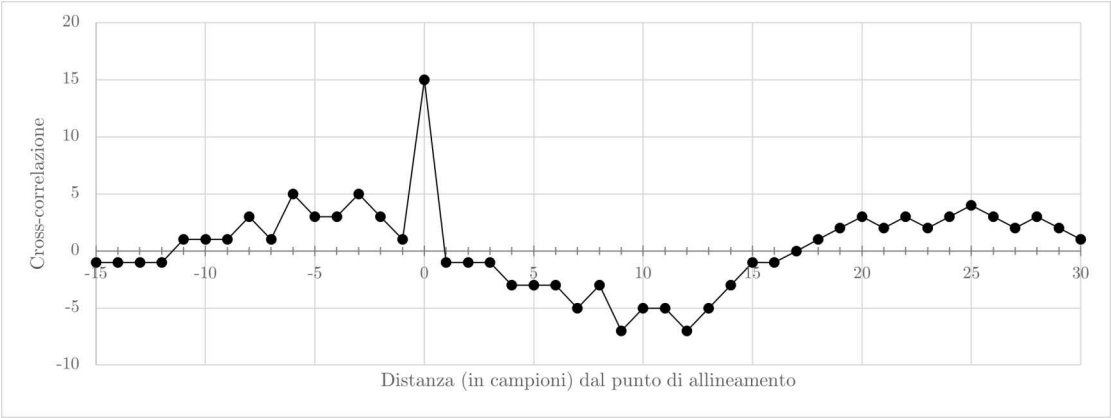

Figura 3.3: Cross-correlazione nella sincronizzazione (MLS con r=4)

Per capire invece quando il messaggio termina sono possibili diverse strategie tra cui l'uso di messaggi a lunghezza fissa predefinita, l'inserimento a inizio messaggio della lunghezza dello stesso oppure l'utilizzo di sequenze terminatrici [23]. Nel sistema realizzato è stata scelta la prima opzione, pertanto dopo aver ricevuto il numero atteso di simboli il ricevitore ritorna a cercare la sequenza di sincronizzazione.

3.2.2 Sincronizzazione di simbolo

Una sincronizzazione più fine può essere ottenuta sovracampionando ogni simbolo di un fattore S. La ricerca del campione iniziale di ogni simbolo prende il nome di *sincronizzazione di simbolo*, e può essere realizzata usando la stessa MLS introdotta per la sincronizzazione di trama. Anche ripetendo S volte i valori per ogni simbolo, il grafico della cross-correlazione non cambia di molto rispetto a quello mostrato in Figura 3.3: sono presenti più punti e i valori di cross-correlazione risultano maggiori, ma la forma e le proporzioni rimangono inalterate (inclusa la posizione del massimo).

Infine, dato che entrambi i dispositivi lavorano a tempo discreto, non ci sono generalmente campioni ricevuti che corrispondono esattamente al fronte di salita

di un rettangolo e il campione individuato è quello immediatamente successivo al reale punto di inizio del simbolo (Figura 3.4).

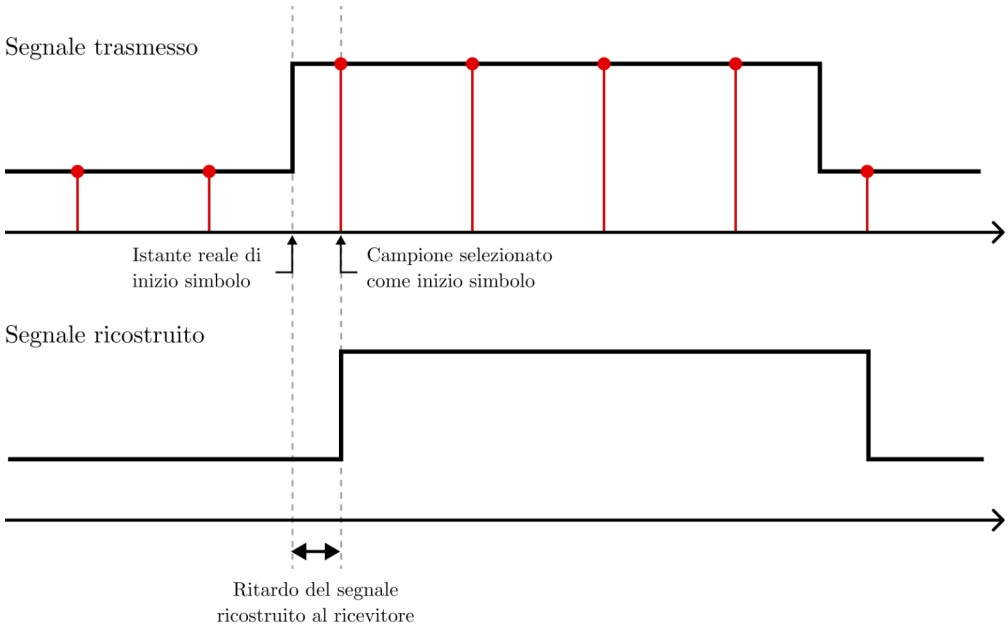

Figura 3.4: Ritardo del segnale ricostruito al ricevitore

3.2.3 Implementazione

Il primo passo per implementare la sincronizzazione è assicurarsi che il campionamento avvenga con un periodo costante T_{RX} e che i simboli siano inviati dal trasmettitore con un periodo $T_{TX} = S \cdot T_{RX}$. Per il campionamento è stata impiegata la libreria *DueAdcFast*, che consente di effettuare letture tramite l'ADC di Arduino Due al tasso desiderato, salvando i valori letti in un buffer mediante PDC[2] [25]. La frequenza di campionamento può essere impostata usando un prescaler ma, per agevolare la fase di test in cui la frequenza viene cambiata spesso, è stato usato il campionamento standard della libreria a $667kHz$, lasciando il compito di stabilire la frequenza di lavoro ad un'altra libreria.

[2]L'uso di un controller DMA periferico *(PDC, Peripheral DMA Controller)* consente di trasferire dati da alcuni moduli, tra cui l'ADC, alla memoria del dispositivo richiedendo un contributo minimo al processore. Questa tecnica permette di aumentare la frequenza di campionamento riducendo il tempo necessario per il salvataggio dei valori letti [24].

La costanza nelle operazioni di trasmissione e ricezione è stata ottenuta con un approccio basato sugli *interrupt* attraverso la libreria *DueTimer*, che usa i timer presenti nella scheda per invocare l'esecuzione di un metodo ogni T microsecondi [26]. Il tasso di sovracampionamento scelto per il progetto è $S = 5$, perciò la frequenza di lavoro del ricevitore è il quintuplo di quella del trasmettitore. Ogni T_{RX} microsecondi, il ricevitore richiama un metodo di DueAdcFast che ritorna l'ultimo valore letto disponibile nel buffer, permettendo così di avere una frequenza di campionamento utile facilmente regolabile, mentre viene mantenuta fissa la frequenza di campionamento fisico.

La sincronizzazione con MLS richiede la trasmissione di due soli simboli, indipendentemente dal tipo di modulazione utilizzata nelle fasi successive. Per massimizzare la distanza tra i due simboli, sono stati usati quelli associati agli stati "completamente acceso" e "spento" del LED. La sequenza usata non è una MLS pura, bensì si tratta di una MLS con $r = 8$, inizializzata con il byte 11001100, seguendo lo schema riportato in Figura 3.2, a cui sono state apportate alcune modifiche dopo aver sostituito il simbolo 0 con -1. Il principale cambiamento è l'eliminazione dell'ultimo valore della sequenza per renderla a media nulla, dato che le MLS, per la loro definizione, hanno sempre media 1.[3] Questo accorgimento fa sì che quando il segnale in ingresso al ricevitore è un segnale costante, ad esempio se il LED è spento e la luminosità della stanza non varia, la cross-correlazione sia mediamente nulla. Il secondo cambiamento è invece la replicazione dei simboli nella sequenza usata come riferimento al ricevitore a causa del sovracampionamento, con le conseguenze descritte nella Sottosezione 3.2.2. Come risultato, la sequenza di sincronizzazione finale ha dimensione

$$L = [(2^r - 1) - 1] \cdot S = [(2^8 - 1) - 1] \cdot 5 = 1270. \tag{3.4}$$

[3]Le MLS hanno lunghezza dispari e il numero di simboli 1 è sempre maggiore di un'unità rispetto a quello di simboli -1 (o 0 se non sono stati mappati).

Per permettere la sincronizzazione, il trasmettitore deve soltanto inviare, prima dei simboli del messaggio, i simboli associati alla sequenza di sincronizzazione concordata.

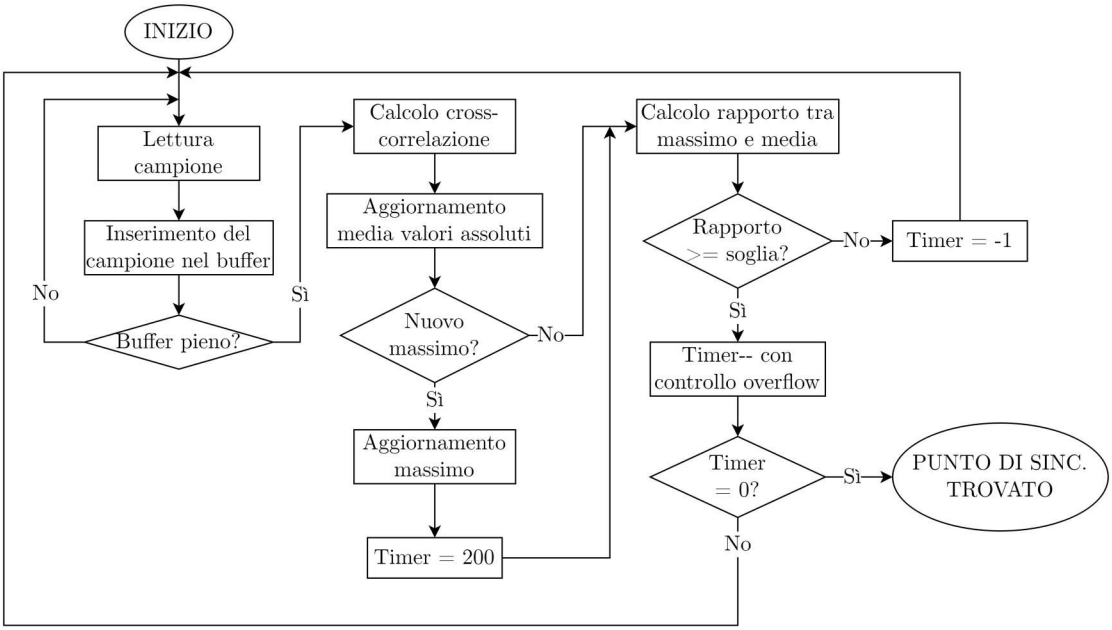

Figura 3.5: Passaggi della sincronizzazione al ricevitore

Il ricevitore, d'altra parte, deve completare diversi passaggi per individuare il punto di sincronizzazione corretto (Figura 3.5). Innanzitutto, i campioni vengono salvati in un buffer circolare di dimensione pari alla lunghezza della sequenza di sincronizzazione. Non avrebbe ovviamente senso cercare la sequenza di sincronizzazione prima di aver ricevuto un numero di campioni pari almeno alla sua lunghezza, quindi fino al riempimento del buffer non vengono effettuate altre operazioni.

Una volta che il buffer è pieno può iniziare la ricerca del punto di sincronizzazione, mentre il buffer continua ad aggiornarsi inserendo i nuovi campioni e scartando quelli più vecchi. Per ogni campione ricevuto, deve essere calcolata la cross-correlazione tra i valori presenti nel buffer (presi in ordine di arrivo) e la sequenza di sincronizzazione di riferimento. L'approccio iniziale prevedeva l'uso della libreria *CMSIS DSP*, che fornisce una serie di funzioni per l'analisi di segnali

tra cui alcune per la cross-correlazione [27]. L'idea è stata però scartata poiché, per evitare di lavorare con numeri in virgola mobile lenti da calcolare[4], andavano convertiti i valori in formato Q31[5], aggiungendo complessità al codice e riducendo la velocità del sistema. È stato quindi realizzato da zero un metodo per calcolare la cross-correlazione, in modo da poterne ottimizzare le prestazioni conoscendo in partenza le caratteristiche delle sequenze. La cross-correlazione è esprimibile come una sommatoria di prodotti (come da (3.3)), che è stata implementata con dei *cicli for*. Per aumentarne la velocità, è stato usato un *loop unrolling* ripetendo 31 volte[6] all'interno di ogni ciclo il codice che moltiplica tra loro due valori delle sequenze coinvolte, riducendo il numero di esecuzioni dei cicli per velocizzare l'esecuzione del programma.

Il massimo attuale viene salvato in una variabile per essere confrontato con i nuovi valori di cross-correlazione ed eventualmente aggiornato. Non è possibile considerare il primo massimo locale trovato come il punto di sincronizzazione, poiché la presenza di rumore nel segnale ricevuto rende normale la comparsa di oscillazioni nei valori della cross-correlazione. Non basta nemmeno cercare un massimo che sia maggiore di un certo valore prefissato di soglia, perché il ricevitore non sa ancora quali valori di tensione aspettarsi per i due simboli della sequenza di sincronizzazione e non può quindi stimare il valore di cross-correlazione associato al punto di perfetto allineamento tra le sequenze.

Per stabilire se un massimo locale sia anche il punto di sincronizzazione, viene

[4] Arduino Due non dispone di un modulo hardware dedicato alle operazioni in virgola mobile.

[5] Il formato Q31 è un formato a virgola fissa usato per rappresentare numeri tra 1 e -1 con un bit per il segno e 31 bit per la parte frazionaria [28].

[6] Il valore 31 non è stato scelto casualmente, bensì è stato ottenuto calcolando il numero medio di esecuzioni di cicli, data la dimensione del buffer pari a 1270, e prendendo il valore che lo minimizza.

confrontato con la media[7] del valore assoluto delle cross-correlazioni: se il loro rapporto è inferiore a un certo coefficiente β il picco viene scartato. L'avere un rapporto con la media maggiore o uguale a β è condizione necessaria ma non sufficiente perché il massimo in esame sia il punto cercato, poiché ci sono due situazioni in cui essa è rispettata ma il picco non è il punto di sincronizzazione: quando le sequenze di sincronizzazione trasmessa e di riferimento iniziano a sovrapporsi e quando aumenta bruscamente la luminosità della stanza. Nel primo caso sono presenti massimi locali molto bassi rispetto al reale picco di sincronizzazione ma nettamente maggiori della media in quel momento, nel secondo ci sono dei massimi locali di altezza simile tra loro dovuti al fatto che alla sequenza di riferimento si sta sovrapponendo un segnale in ingresso con forma a gradino (una volta che il fronte di salita supera la sequenza di riferimento, la media nulla della sequenza usata riporta attorno allo zero la cross-correlazione). Perché un massimo sia considerato il punto di sincronizzazione, il rapporto con la media deve essere maggiore o uguale a β per tutti i 200 campioni successivi, in modo che la media abbia il tempo di crescere se i valori di cross-correlazione si mantengono alti. Il valore β è stato scelto pari a 9 in base a diverse simulazioni effettuate con un foglio di calcolo, in cui sono state provate varie configurazioni di luminosità dell'ambiente, rumore e intensità del LED; e successivamente confermato dai test con il sistema reale. Nei grafici in Figura 3.6 sono riportati i risultati di un'esecuzione della procedura di sincronizzazione, in cui si nota come all'inizio della ricezione della sequenza di sincronizzazione corrisponde un netto picco dei valori di cross-correlazione.

[7] La media usata è una media mobile in cui i valori più recenti hanno un peso maggiore rispetto a quelli precedenti, in modo da reagire adeguatamente a grosse variazioni improvvise. Detto n l'ultimo valore disponibile, la media è calcolata come $m_n = (1 - \alpha) \cdot m_{n-1} + \alpha \cdot n$ dove il coefficiente α determina la velocità di reazione (dopo alcune prove è stato scelto $\alpha = 0.01$).

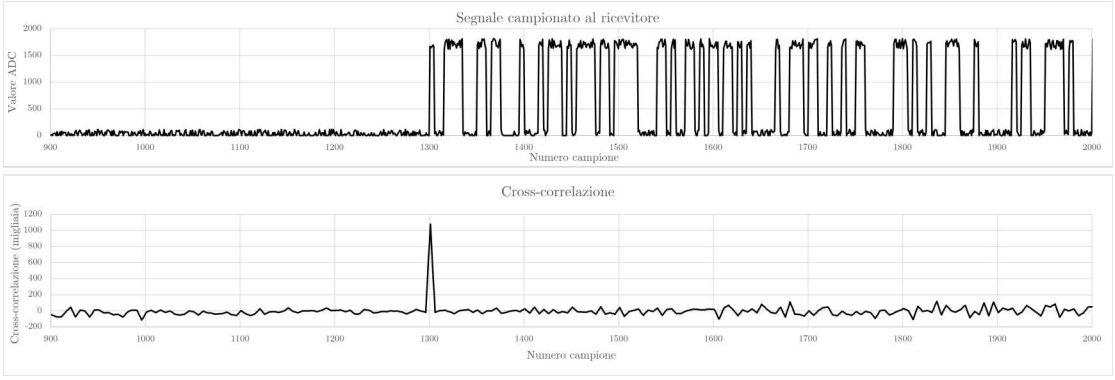

Figura 3.6: Segnale ricevuto e cross-correlazione corrispondente

3.3 Codifica e decodifica dei simboli

Una volta completata la sincronizzazione, è possibile trasmettere il messaggio vero e proprio. Anche in questo caso, è il ricevitore a svolgere la maggior quantità di lavoro, dovendo stabilire le soglie limite delle regioni di decisione e associare ai valori ricevuti i corrispondenti bit. Il trasmettitore deve invece convertire i bit in arrivo dal PC in valori di tensione per ottenere l'intensità luminosa del LED corretta per ogni simbolo.

3.3.1 Codifica al trasmettitore

Dal momento dell'accensione, il trasmettitore si mette in ascolto sulla porta seriale, in attesa di un messaggio da inviare. La comunicazione tra PC e Arduino Due ha come unità il byte, ma la modulazione usata (4-PAM) permette di associare ad ogni simbolo solamente una coppia di bit. Il byte ottenuto deve quindi essere diviso in quattro segmenti di uguale lunghezza, prima di inserire in un buffer i valori del DAC corrispondenti a ciascuna coppia di bit. Il DAC, come detto nella Sezione 2.1, ha 4096 livelli esprimibili con numeri interi tra 0 e 4095, quindi la modulazione consiste semplicemente nell'associare a ognuna delle quattro possibili coppie un opportuno numero intero in tale intervallo.

Per ottenere una 4-PAM al ricevitore, non si possono scegliere quattro valori ugualmente distanziati per il DAC, altrimenti la non-linearità del LED eliminerebbe la condizione di equidistanza dei vettori della costellazione. Il grafico in Figura 3.7 mostra i risultati di un test effettuato aumentando gradualmente il livello del DAC, evidenziando come i valori letti dall'ADC non seguano un andamento lineare e siano quasi nulli fino al raggiungimento della tensione di soglia del LED. I livelli da

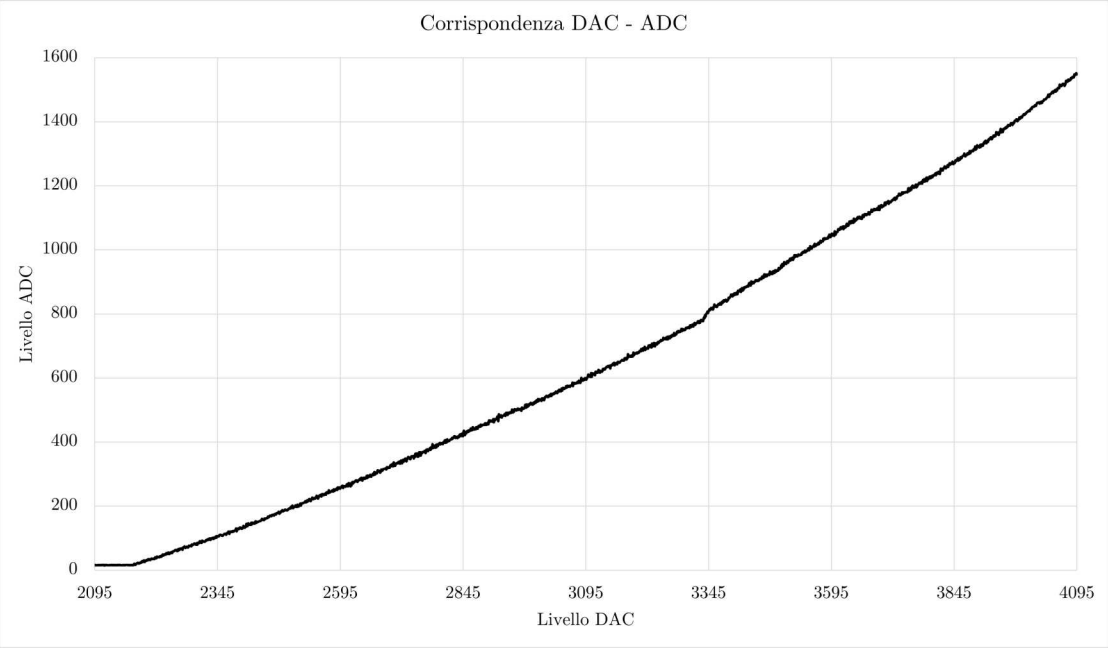

Figura 3.7: Relazione tra il valore impostato al DAC e quello letto dall'ADC

utilizzare al DAC sono stati decisi proprio partendo da questo grafico, individuando prima quattro valori equidistanti sull'asse delle ordinate $\{17, 527, 1037, 1547\}$ e poi i corrispondenti valori su quello delle ascisse $\{0, 2992, 3585, 4095\}$ (dato che fino alla tensione di soglia è presente solo la componente dovuta alla luminosità ambientale, è stato preso 0 in modo da ridurre il dispendio di corrente).

I simboli sono stati associati alle coppie di bit con un codice Gray, un codice binario in cui tra due posizioni consecutive c'è un solo bit di differenza [19, pp. 312-313]. La codifica di Gray è ampiamente utilizzata nelle comunicazioni poiché, in caso di errori nella demodulazione dovuti al rumore, il numero medio di bit errati

è inferiore rispetto ad una codifica che segue la notazione posizionale degli interi binari.

La Tabella 3.1 riassume i valori e i simboli coinvolti nella fase di codifica al trasmettitore.

Tabella 3.1: Schema generale della codifica

Simbolo	**Valore DAC**	**Coppia di bit**
1	0	00
2	2992	01
3	3585	11
4	4095	10

Ci sono, infine, altri tre aspetti da considerare sui compiti del trasmettitore. Il primo è che tra la sequenza di sincronizzazione e l'invio del messaggio viene trasmessa un'ulteriore sequenza in cui si alternano 20 simboli 1 e 20 simboli 4, il cui scopo verrà illustrato nella Sezione 3.3.2. Il secondo riguarda invece la fine del messaggio che, come già discusso nella Sezione 3.2.1, è a lunghezza fissa: dopo aver trasmesso il numero prefissato di simboli, il trasmettitore torna allo stato iniziale, eventualmente inviando subito una nuova sequenza di sincronizzazione se sono presenti ancora valori nel buffer. Il terzo aspetto è legato alla gestione di quelle situazioni in cui, per svariate ragioni, il buffer di trasmissione si svuota prima della fine del messaggio. Se al momento di leggere il simbolo successivo da trasmettere non sono presenti valori nel buffer, viene trasmesso il simbolo 1.[8]

[8]Dovendo scegliere arbitrariamente un simbolo di default da trasmettere in caso di buffer vuoto, è stato preferito il simbolo che comporta il minor dispendio di corrente, ovvero quello associato al LED spento e alla minima tensione.

3.3.2 Decodifica al ricevitore

Il ricevitore, per ottenere la sequenza di bit da inviare al PC, deve per prima cosa capire quali simboli sono stati trasmessi. Le soglie che delimitano le regioni di decisione dipendono dall'intervallo di valori ricevuti (che varia in base a fattori come la distanza e il grado di allineamento tra i dispositivi) e non possono essere quindi stabilite a priori. Per determinare correttamente le soglie di decisione viene usata la sequenza di 40 simboli trasmessa dopo quella di sincronizzazione. Innanzitutto, viene calcolata la media aritmetica dei campioni per entrambi i tipi di simbolo (1 e 4) in modo da ottenere i valori corrispondenti al simbolo massimo e a quello minimo. A partire da questi, vengono ricavati i valori associati agli altri due simboli (2 e 3) che saranno tali da rendere i simboli equidistanti tra loro. A questo punto si possono finalmente stabilire le soglie di decisione, posizionate nei punti medi tra i simboli 1-2, 2-3 e 3-4.

A partire da questo momento, e fino al raggiungimento della lunghezza prestabilita del messaggio, i simboli vengono decodificati calcolando la media aritmetica di ogni gruppo di cinque valori, confrontando il risultato ottenuto con le soglie e selezionando infine la coppia di bit corrispondente. Quando sono state decodificate quattro coppie di bit, vengono assemblate in un unico byte e inviate al PC tramite la porta seriale.

Il sistema proposto presenta però una grossa criticità: è sufficiente variare la luminosità dell'ambiente nella fase di lettura del messaggio per compromettere la correttezza dei bit ottenuti. Le soglie vengono infatti calcolate tenendo conto della luminosità al momento dell'avvenuta sincronizzazione ed eventuali cambiamenti successivi comportano l'innalzamento o l'abbassamento dei campioni, ma non delle soglie stesse. La soluzione consiste nel centrare inizialmente le soglie in zero, senza alterarne l'ampiezza, e sottrarre ai campioni la media dei valori ricevuti fino a quel

momento[9], in modo da ottenere un segnale a media nulla centrato in zero come le soglie. La sequenza usata per il rilevamento delle soglie, essendo un'alternanza regolare di simboli 1 e 4, permette di stabilizzare la media iniziale al valore medio dei simboli. Questo metodo non può garantire che non ci siano errori dovuti alla variazione di luminosità, poiché se la media fosse troppo sensibile cambierebbe di continuo assumendo il valore dell'ultimo campione letto. Tuttavia, l'impatto di tale variazione viene mitigato e reso nullo dopo un certo numero di simboli, portando a una perdita solo parziale dell'informazione. Ovviamente la strategia appena descritta non è adatta a messaggi con lunghe sequenze dello stesso simbolo, dato che andrebbero a spostare la media causando errori nelle letture successive.

[9]Anche questa media, come quella usata per la sincronizzazione, è una media mobile, con un coefficiente $\alpha = 0.005$ inferiore per evitare troppe oscillazioni dovute alla naturale distanza tra i diversi simboli.

Capitolo 4

Risultati e sviluppi futuri

Nel quarto capitolo di questo elaborato, vengono illustrati e analizzati i risultati dei test effettuati sul sistema. Sono inoltre presenti una sezione dedicata alle possibilità di miglioramenti futuri e un riepilogo dei componenti utilizzati.

4.1 Risultati sperimentali

Dopo aver disegnato e assemblato i circuiti, progettato gli algoritmi e sviluppato il software, sono stati eseguiti alcuni test volti a valutare le prestazioni del sistema costruito. Per agevolare l'esecuzione dei test, in cui era necessario spostare spesso i dispositivi, i circuiti e le schede sono stati montati su dei supporti in legno in modo da evitare scollegamenti accidentali dei cavi e facilitare il corretto allineamento (Figura 4.1).

4.1.1 Probabilità di errore

La probabilità di errore è un parametro fondamentale nelle comunicazioni, dato che generalmente è inutile realizzare sistemi con elevata velocità o grandi di-

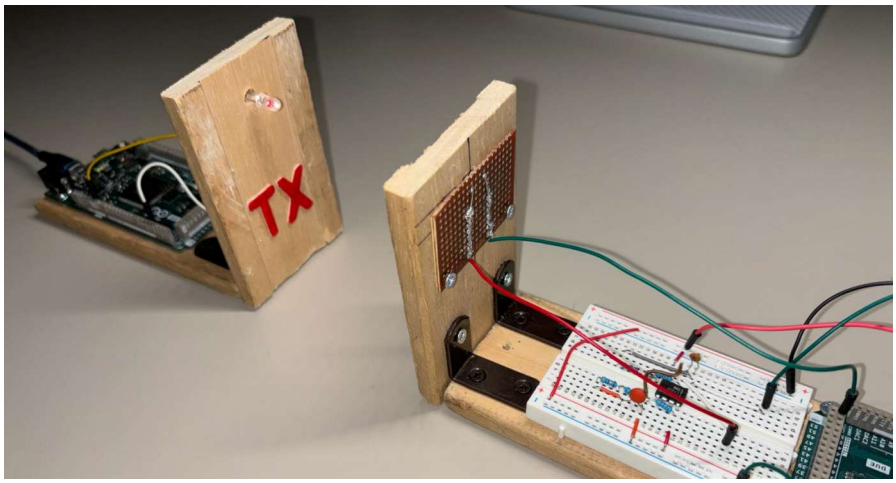

Figura 4.1: Dispositivi in posizione per la fase di test

stanze supportate, se poi la maggior parte delle informazioni non viene ricevuta correttamente.[1]

La probabilità di errore sui simboli P_e è la probabilità che il simbolo trasmesso $s_{n,TX}$ e il simbolo interpretato dal ricevitore $s_{n,RX}$ siano diversi. Per una M-PAM su canale AWGN questa probabilità può essere espressa come funzione dell'energia media della segnalazione

$$E_s = \frac{M^2 - 1}{3} E_h \tag{4.1}$$

e dell'energia media del rumore E_ω, secondo la formula

$$P_e = 2\left(1 - \frac{1}{M}\right) Q\left(\sqrt{\frac{6}{M^2 - 1}\frac{E_s}{E_\omega}}\right) \tag{4.2}$$

dove Q indica la funzione di distribuzione complementare della variabile gaussiana standard $Q(a) = 1 - \Phi(a) = P(x > a)$ [19, pp. 311-312]. Per stimare la probabilità di errore sui simboli reale del sistema, è stato trasmesso un messaggio predefinito con simboli equiprobabili (condizione che viene assunta vera nella (4.2)). Al ricevitore, sono stati calcolati i valori medi per ogni tipo di simbolo della segnalazione,

[1] Quando la quantità di errori è limitata, possono essere usate tecniche di codifica di canale per individuare, e in alcuni casi anche correggere, tali errori [19, pp. 383-394].

in modo da sottrarli poi ai campioni ricevuti per ottenere il rumore. L'energia del rumore E_ω è stata calcolata sommando i quadrati dei valori rimasti dopo il passaggio precedente, mentre quella del segnale E_s è stata ricavata svolgendo la stessa operazione con i valori medi dei simboli. Il fatto che il segnale fosse prestabilito ha permesso al ricevitore di eseguire automaticamente il calcolo del numero di simboli errati[2]. I risultati sono illustrati nel grafico in Figura 4.2, in cui i punti neri rappresentano la probabilità di errore sui bit in funzione del rapporto tra le energie di segnale e rumore $SNR = E_s/E_\omega$ espresso in decibel.

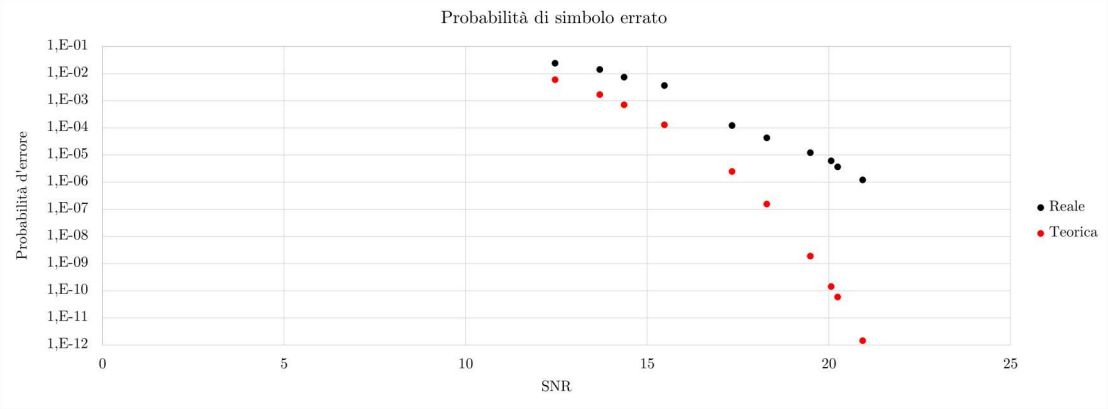

Figura 4.2: Probabilità di errore sui simboli e SNR

I punti rossi indicano invece i valori teorici ottenuti dalla (4.2), per lo stesso SNR. Si nota subito che la probabilità di errore effettiva non rispetta le previsioni teoriche, pur mantenendo la caratteristica di diminuire al crescere del SNR. I fattori che potrebbero spiegare il motivo di questa discrepanza sono molteplici e di nature differenti. Innanzitutto, va considerato che il rumore è stato assunto AWGN ma, come esposto nella Sezione 3.1.2, nella realtà sono presenti diverse forme di rumore coinvolte in un sistema VLC (in Figura 4.3 è riportata la distribuzione del rumore stimata nella trasmissione di un messaggio).

[2]Per il test non è stato usato l'aggiustamento rispetto alla media dei valori letti dall'ADC, per evitare che ciò influisse sulla probabilità di errore.

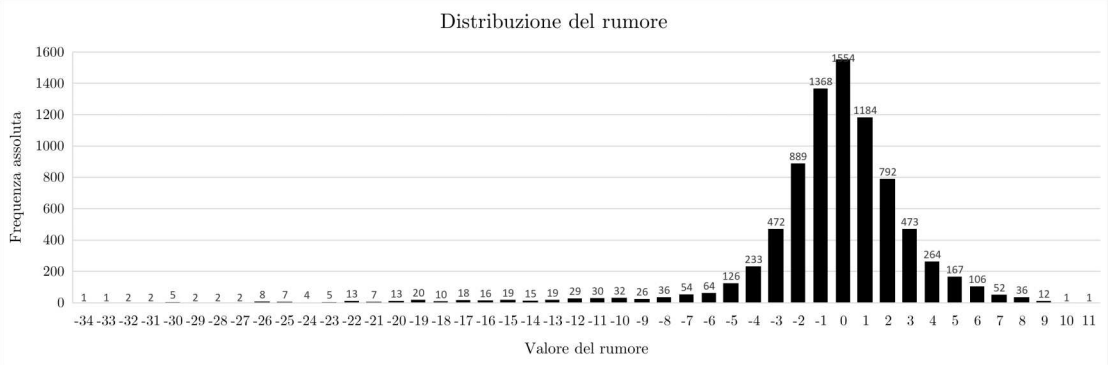

Figura 4.3: Rumore stimato in un test di trasmissione

In particolare, la natura economica e non professionale di cavi di collegamento e saldature potrebbe aver variato notevolmente le caratteristiche del rumore, influenzando sia l'accuratezza delle letture al ricevitore che la stabilità della luminosità del LED al trasmettitore. Un altro fattore di cui tenere conto è la difficoltà di ottenere simboli perfettamente equidistanti data la strumentazione usata, che diventa incisiva soprattutto quando i simboli della segnalazione sono molto vicini tra loro. A tal proposito va anche osservato che Arduino Due utilizza una scala discreta di valori con 4096 livelli per l'ADC, ma quando i simboli sono ravvicinati possono esserci anche solo un paio di livelli a separarli ed eventuali valori nel mezzo vengono inclusi nel livello più vicino, motivo per cui non sono stati possibili test con SNR inferiori rispetto a quelli riportati. Si nota infine che non sono stati svolti test nemmeno con valori maggiori di SNR, poiché certi valori di probabilità di errore sarebbero raggiungibili solo trasmettendo un gran numero di simboli, cosa difficile da realizzare a causa dei limiti di memoria di Arduino Due che permettono di effettuare i calcoli necessari solo con una quantità moderata di simboli per messaggio. Nel test sono stati trasmessi messaggi di lunghezza 2 KB[3], corrispondenti a 8192 simboli, ripetuti ciascuno 100 volte per rilevare probabilità di errore dell'ordine di 10^{-6}. Le prove in cui il SNR è risultato maggiore di 21 hanno indicato

[3]La lunghezza indicata è quella dei dati effettivi, senza tenere conto delle sequenze di sincronizzazione e rilevamento soglie.

un numero di errori pari a zero, anche se probabilmente aumentando le ripetizioni si sarebbero ottenuti valori non nulli.

4.1.2 Distanza massima

Supportare trasmissioni su distanze elevate non è necessariamente un requisito fondamentale per un sistema VLC; l'importanza di questo parametro dipende dalle applicazioni pratiche a cui è destinato. È comunque interessante osservare qual è la distanza massima del sistema realizzato, per capire cosa la limita e come potrebbe essere aumentata.

In Figura 4.4 sono riportati i valori letti dall'ADC in corrispondenza dell'emissione di simboli con il LED alla massima intensità al variare della distanza. Essendo tali valori proporzionali all'energia luminosa assorbita dal fotodiodo, tendono a diminuire in modo non lineare all'aumentare della distanza tra i dispositivi.

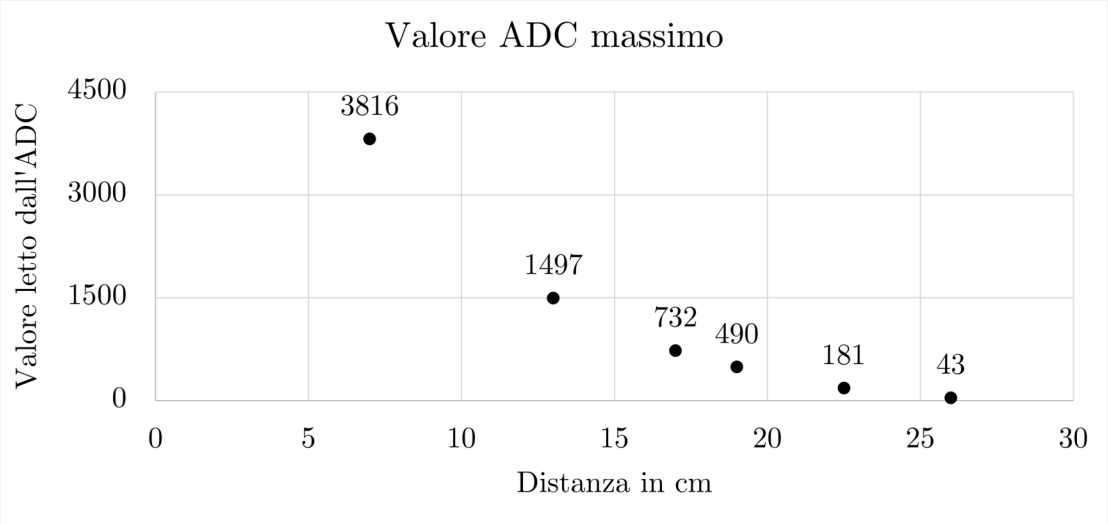

Figura 4.4: Valori letti dall'ADC con il LED completamente acceso a diverse distanze

Tra i diversi valori rilevati, tre meritano particolare attenzione. Il primo è quello più elevato, 3816, ottenuto ad una distanza di 7 cm, che analizzando le medie per i diversi simboli nella segnalazione è risultato essere dovuto al *clipping*, un fe-

nomeno che si verifica quando il valore teorico supera l'intervallo dell'amplificatore. Osservando i livelli medi associati ad ogni simbolo, riportati nella Tabella 4.1, si nota infatti che i simboli 1,2 e 3 sono equidistanti, mentre il simbolo 4 è nettamente più vicino al precedente.

Tabella 4.1: Trasmissione a distanza 7 cm

Simbolo	**Valore ADC**	**Distanza dal precedente**
1	35	- - -
2	1704	1669
3	3373	1669
4	3816	443

Il secondo campione è quello con valore 181, poiché la distanza corrispondente, pari a 22.5 cm, è stata l'ultima su cui si è ottenuta una reale trasmissione con la 4-PAM: allontanando ulteriormente i dispositivi l'accensione al minimo del LED non viene più rilevata dal ricevitore, con i simboli 0 e 1 che risultano sovrapposti. Il terzo valore, infine, è quello ottenuto ad una distanza di 26 cm, l'ultimo per cui è stato possibile effettuare almeno la sincronizzazione (che a differenza della trasmissione del messaggio sfrutta solo i simboli esterni della costellazione).

Per aumentare la distanza supportata, un primo intervento potrebbe essere la sostituzione dei resistori nel circuito del ricevitore (e di conseguenza dei condensatori) seguendo i passaggi illustrati nella Sezione 2.3 ma con una corrente I_{max} minore. Così facendo, aumenterebbe la capacità di amplificazione dell'opamp, amplificando maggiormente il segnale ma anche il rumore. Una soluzione più efficace sarebbe la sostituzione del LED usato con uno più potente, alimentato con una fonte esterna, in modo da avere a disposizione una potenza maggiore di quella che può fornire Arduino Due, e pilotato tramite un opamp o un transistor.

4.1.3 Velocità di trasmissione

Per fornire un indicatore sulla velocità di trasmissione del sistema che fosse indipendente da fattori come la lunghezza del messaggio o la presenza di codici a correzione d'errore, è stato considerato il numero di simboli trasmessi nell'unità di tempo. Per testare il limite superiore del sistema è stato diminuito progressivamente il periodo di simbolo, fino a quando non è stato più in grado di funzionare correttamente. La velocità massima raggiunta è stata di $R_s = 1388.89$ simboli al secondo, corrispondenti ad un periodo di simbolo $T_s = 720\ \mu s$. Considerando che la modulazione usata è una 4-PAM, in cui ogni simbolo porta due bit di informazione, la velocità massima può essere espressa anche come $R_b = 2777.78\ bps$.

Il collo di bottiglia è stato individuato nella procedura di sincronizzazione, che richiede il maggior numero di operazioni. Per aumentare la velocità massima di trasmissione sarebbe sufficiente accorciare la sequenza di sincronizzazione, per ridurre i calcoli necessari, notando però che questo influirebbe negativamente sulla capacità di sincronizzazione del sistema.

4.1.4 Robustezza alla variazione di luminosità ambientale

Come descritto nella Sezione 3.3.2, il sistema sviluppato è dotato di una tecnica per ridurre l'impatto delle variazioni di luminosità ambientale durante la trasmissione di un messaggio. Per testare l'efficacia di tale soluzione sono stati analizzati gli errori sui simboli in tre differenti situazioni: luminosità della stanza stabile, accensione di una fonte luminosa e spegnimento della stessa fonte (Figura 4.5).

Quando la luce ambientale rimane stabile, l'aggiustamento dei campioni usando la media mobile dei valori non comporta alcuna differenza,[4] mentre nelle situazio-

[4]Il messaggio usato per il test era costruito in modo da avere un'alternanza regolare dei diversi simboli, in caso di messaggi con lunghe sequenze di simboli associati a valori elevati (o, al contrario, ridotti) il sistema avrà un numero maggiore di errori con la tecnica basata sulla media.

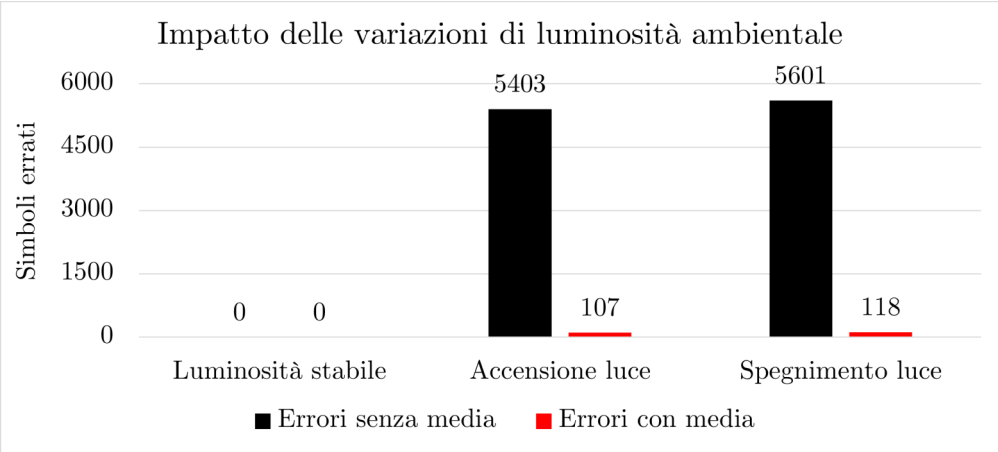

Figura 4.5: Test con variazione di luminosità ambientale

ni di accensione e spegnimento della luce i suoi effetti sono ben visibili. Senza l'aggiustamento, quando la luce è stata accesa poco dopo l'inizio del messaggio, il ricevitore ha iniziato ad interpretare tutti i simboli come quello a valore maggiore, portando ad una lettura errata del 66% dei simboli del messaggio. Con l'aggiustamento il ricevitore ha cominciato a comportarsi allo stesso modo subito dopo l'accensione della luce, adattandosi però gradualmente alla nuova luminosità ambientale e sbagliando complessivamente solo l'1.3% dei simboli. Risultati simili sono stati ottenuti anche nel test in cui la luce, inizialmente accesa, è stata spenta a messaggio iniziato: è stato interpretato erroneamente il 68.4% dei simboli senza l'aggiustamento, contro l'1.4% con il suo utilizzo.

4.2 Sviluppi futuri

Nonostante il sistema sviluppato possa considerarsi funzionante allo stato attuale, sono certamente possibili interventi per migliorarne le prestazioni.

Per quanto riguarda la parte hardware, si potrebbero modificare i circuiti di entrambi i dispositivi sostituendo o aggiungendo componenti. Nel trasmettitore, andrebbe installato un LED ad alta potenza per aumentare la distanza massima

supportata. Il ricevitore, invece, potrebbe essere ridisegnato per filtrare meglio il rumore e stabilizzare il segnale in ingresso, usando schemi più complessi come quello del progetto OpenVLC. Su entrambi i dispositivi andrebbero anche rifatti, in modo più professionale, i collegamenti tra i componenti e le saldature.

A livello software si può sicuramente ottimizzare il codice, in particolar modo del ricevitore, sfruttando maggiormente le funzionalità avanzate di Arduino Due come il PDC e la programmazione diretta dei registri. Inoltre, l'intero programma potrebbe essere convertito in una libreria facilmente integrabile in altri progetti, permettendo agli utilizzatori di variare i parametri del sistema (ad esempio il tipo di M-PAM, il periodo di simbolo, la lunghezza della sequenza di sincronizzazione o il tasso di sovracampionamento).

Infine, volendo aumentare le possibilità del sistema, potrebbero essere aggiunte tecniche di codifica di canale e addirittura raddoppiati i componenti dei circuiti, rendendo entrambi i dispositivi in grado sia di inviare che di leggere i messaggi per una comunicazione bidirezionale.

4.3 Versioni dei software

Per dare a chi lo desiderasse la possibilità di replicare il sistema realizzato, vengono riportati nella Tabella 4.2 tutti i software usati con le relative versioni. Per quanto riguarda la parte hardware, tutte le indicazioni necessarie (schemi, componenti e calcoli per la scelta di resistori e condensatori) sono contenute nel Capitolo 2. I codici, invece, sono disponibili al link `https://github.com/aledisa/arduinoVLC`.

Tabella 4.2: Versione dei software utilizzati

Software	**Versione**
Sistema operativo PC	Microsoft Windows 10 Home v. 10.0.19045 Build 19045
Arduino IDE	2.2.1, CLI 0.34.0
Arduino Sam Boards core	1.6.12
Libreria DueTimer	1.4.8
Libreria DueAdcFast	1.2.0

Capitolo 5

Conclusione

I dispositivi realizzati, descritti nel Capitolo 2, sono semplici da assemblare ed economici, con un costo totale di circa 112 euro (IVA esclusa). Inoltre, la modulazione PAM, la cui implementazione è stata approfondita nel Capitolo 3, è risultata adatta al tipo di sistema costruito. Le prestazioni sono nettamente inferiori rispetto a progetti più complessi come OpenVLC, come esposto nella Sezione 4.1, ma comunque sufficienti a scopi didattici o per trasmissioni di piccole quantità di dati su brevi distanze. Sono inoltre ampie le possibilità di miglioramento con interventi su hardware e software, come illustrato nella Sezione 4.2.

In definitiva, nonostante i limiti presenti, questo lavoro ha dimostrato che la realizzazione di un sistema VLC, con modulazione PAM, basato sulla scheda Arduino Due è certamente possibile.

Bibliografia

[1] C. Schick, *Le Storie di Polibio*, vol. 2, pp. 235–239. Arnoldo Mondadori Editore, 1955.

[2] M. Bellis, "Alexander graham bell's photophone was an invention ahead of its time," *ThoughtCo*, marzo 2019. https://www.thoughtco.com/alexander-graham-bells-photophone-1992318.

[3] F. Gfeller and U. Bapst, "Wireless in-house data communication via diffuse infrared radiation," *Proceedings of the IEEE*, vol. 67, pp. 1474 – 1486, 12 1979.

[4] Twibright Labs, "Ronja project." http://ronja.twibright.com/about.php [Consultato il 2 marzo 2024].

[5] N. Chi, Y. Zhou, Y. Wei, and F. Hu, "Visible light communication in 6g: Advances, challenges, and prospects," *IEEE Vehicular Technology Magazine*, vol. 15, no. 4, pp. 93–102, 2020.

[6] AIRC, "L'uso dei cellulari può causare un tumore al cervello?." https://www.airc.it/cancro/informazioni-tumori/corretta-informazione/luso-dei-cellulari-puo-causare-un-tumore-al-cervello [Consultato il 2 marzo 2024].

[7] Z. Wang *et al.*, *Visible Light Communications: Modulation and Signal Processing*. Hoboken, NJ: John Wiley & Sons, Ltd, 2017.

[8] IMDEA Networks Institute, "Openvlc." `http://www.openvlc.org/home.html` [Consultato il 29 dicembre 2023].

[9] BeagleBoard.org Foundation, "Beaglebone black." `https://www.beagleboard.org/boards/beaglebone-black` [Consultato il 2 marzo 2024].

[10] Arduino S.r.l., "Arduino due." `https://store.arduino.cc/products/arduino-due` [Consultato il 21 dicembre 2023].

[11] M. Reyes, P. Melín, E. Espinosa, M. Rivera, and J. Suárez, "An experimental comparison between an arduino due and a dsp-based delfino launchpad board," in *2022 41st International Conference of the Chilean Computer Science Society (SCCC)*, pp. 1–8, 2022.

[12] Marktech Optoelectronics, *MTE6066N5-UR datasheet*, aprile 2012.

[13] G. Stark, "light," *Encyclopedia Britannica*, dicembre 2023. `https://www.britannica.com/science/light`.

[14] Broadcom Inc., *HL3P-Nx45/HL3P-Bx60 datasheet*, settembre 2017.

[15] Semiconductor Components Ind., *QSD2030 datasheet*, novembre 2018.

[16] Linear Technology Corporation, *LT1632 datasheet*, 1998.

[17] Texas Instruments, *Photodiode Amplifier Circuit*, febbraio 2019.

[18] J. Lehnert *et al.*, "telecommunication," *Encyclopedia Britannica*, febbraio 2024. `https://www.britannica.com/technology/telecommunication`.

[19] N. Benvenuto and M. Zorzi, *Principles of Communications Networks and Systems*. John Wiley & Sons Ltd, 2011.

[20] A. Falaschi, *Trasmissione dei segnali e sistemi di telecomunicazione*, pp. 477–482. 2022.

[21] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing*, pp. 118–222. Pearson Education, 1996.

[22] N. Benvenuto, G. Cherubini, and S. Tomasin, *Algorithms for Communications Systems and their Applications*, pp. 99–101. John Wiley & Sons Ltd, 2021.

[23] Y. Lin, R. Hwang, and F. Baker, *Computer Networks: An Open Source Approach*, pp. 127–128. McGraw-Hill, 2012.

[24] Atmel, *SAM3X / SAM3A Datasheet*, marzo 2015.

[25] A. Previtali, "Dueadcfast." `https://github.com/AntonioPrevitali/DueAdcFast` [Scaricato il 17 maggio 2023].

[26] I. Seidel, "Duetimer." `https://github.com/ivanseidel/DueTimer` [Scaricato il 10 aprile 2023].

[27] Arm Software, "Cmsis5." `https://github.com/ARM-software/CMSIS_5` [Scaricato il 6 maggio 2023].

[28] S. Kumar, "Representing decimal data in q-format." `https://www.pathpartnertech.com/representing-decimal-data-in-q-format/` [Consultato il 9 luglio 2023].

Ringraziamenti

Mi è doveroso dedicare questo spazio finale della mia tesi ai ringraziamenti verso tutte le persone che hanno contribuito, direttamente e indirettamente, alla sua realizzazione.

Innanzitutto desidero ringraziare il mio relatore, il Prof. Stefano Tomasin, per i suoi fondamentali consigli e il suo costante supporto in tutte le fasi di questo lavoro.

Ringrazio poi mamma e papà, i miei fratelli Giacomo e Leonardo, i nonni e tutti i componenti della mia famiglia che da sempre mi supportano con infinita pazienza, ciascuno a proprio modo.

Ringrazio gli amici con cui ho condiviso mille avventure e che ci sono sempre stati nel momento del bisogno, in particolare Jessica che mi sopporta quotidianamente da anni.

Infine, voglio ringraziare tutti coloro che mi hanno aiutato a completare questo elaborato con le loro preziose indicazioni quando mi sono trovato in difficoltà di fronte ad argomenti per me nuovi.