Delft University of Technology
Master of Science Thesis in Embedded Systems

# Coloring the Pulse Position Modulation in VLC

**Tolga Parlan**

Embedded Systems

TU Delft Delft University of Technology

# Coloring the Pulse Position Modulation in VLC

Master of Science Thesis in Embedded Systems

Embedded Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

Tolga Parlan
t.parlan@student.tudelft.nl
tolgaparlan@gmail.com

June 2nd 2023

**Author**
Tolga Parlan (t.parlan@student.tudelft.nl)
(tolgaparlan@gmail.com)
**Title**
Coloring the Pulse Position Modulation in VLC
**MSc Presentation Date**
June 2nd 2023

**Graduation Committee**

| | |
|---|---|
| Dr. Marco Zúñiga | Delft University of Technology |
| Dr. Qing Wang | Delft University of Technology |

**Abstract**

This thesis describes the design and implementation of Coloured Multiple Pulse Position Modulation (CMPPM) which is a novel Visible Light Communications (VLC) modulation scheme focusing on expanding upon the standard Pulse Position Modulation (PPM) via taking advantage of the information encoding potential of the color spectrum. The system is able to transmit data uni-directionally at a throughput on par with comparable CSK and multiple times above comparable PPM schemes, solely using commodity multi-colour LEDs and a photodiode. It supports a flexible forward error correction level as well as different colour and optionally dimming outputs during the communications. In order to test the scheme's feasibility and robustness, it is realised in hardware as well as software using the Rust programming language. A transmission speed of 2 kb/s is achieved with simple commodity hardware components.

# Preface

The research area of Visible Light Communications is an exciting field to explore as a student thanks to the low cost of the components involved, which makes it feasible to experiment with new ideas and gives the possibility of converting abstract knowledge about the physics of light and data modulation into tangible systems. In this vein, I have also enjoyed the practically focused nature of many projects in this field, as I appreciated the amount of experience I have personally gained while working on my own thesis project. As an Embedded Systems student with a background in Computer Science, I have felt that I benefited from gaining experience with the different domains involved, including rudimentary hardware design as well as an extensive software implementation of the system including an embedded part, and research into light physics. The thesis project has taken approximately one and a half years to complete.

I would like to thank my parents, my sister as well my girlfriend for their practical and motivational support during my thesis work. Additionally, I would like to thank Hanting Ye and Qing Wang for their support and help during the project.

Tolga Parlan

Delft, The Netherlands
29th May 2023

# Contents

# Chapter 1

# Introduction

As the need and demand for wireless communications constantly increase, the limitations of the available bandwidth increase as well and there is a search for wireless communication mechanisms to be used independently of the licensed frequency bands [26]. The visible light spectrum presents an opportunity in this area, as there are no regulatory limitations on its use unlike most other commonly used communication bands which are strictly regulated. Especially with the ubiquitous usage of communications devices with various types of LEDs and photo sensors as well as the abundance of LED lighting in contemporary spaces, the research over how to integrate these devices' light sensing and transmission capabilities for creating an alternative avenue of data communication have gained momentum, culminating in the field of Visible Light Communications (VLC).

There is a substantial amount of research into integrating VLC-based systems into 6G networks [3]. This stems from the core advantage of VLC-based communication systems, namely the ability to use currently unexploited wide frequency bands of visible light. While the use of radio frequency bands is strictly controlled due to interference between the devices using the same bandwidth range, this is much less of a problem since VLC-based communication systems do not penetrate hard objects such as walls, and therefore spectrum sharing is almost a non-issue.

While it is possible to increase the data throughput of VLC systems by simply switching the LEDs on and off faster, there is also untapped potential in using color for the purpose of achieving increased data rates [25]. Even though VLC systems that utilize color effectively already exist, they typically use a color sensor in order to achieve precise data readings of the color conditions. Color sensors have the associated downside of relatively slow reading frequencies. Another interesting possibility is the usage of faster sensors such as photodiodes, which, even though are not directly designed to sense color, can still be used for this purpose by taking advantage of their inherent physical properties and carefully designing a modulation scheme to accommodate the hardware.

## 1.1 Research Objective

The objective of this thesis is to create a novel VLC modulation system that can achieve high data throughput and low error rates by utilizing the opportunities presented by using the color spectrum in Pulse Position Modulation systems. This should be achieved while using a single simple photodiode as the detector and multi-color LED(s) as the transmitter in order to avoid the cost and complexity of specialized hardware. As part of the research, the system should also be implemented with commodity hardware and software developed for the purpose, and evaluated under different conditions.

## 1.2 Research Challenges

The core challenge of the project is the fact that photodiodes are not designed for color sensing. A photodiode creates an electrical current when light is incident on its sensor. This current can be transformed into a voltage output by integrating an op-amp in the circuit and thus easily measured by an analog-to-digital converter. As photodiodes typically create different (known) electrical currents for different light frequencies (i.e. colors), a modulation scheme can be created to exploit them as limited but blazing-fast color sensors. The challenges faced while creating such a modulation algorithm are listed below:

- The electrical current produced by photodiodes in response to the light varies according to the frequency of the light. However, this relationship is not linear and not uniform across different photodiode models. Therefore a careful selection of a photodiode as well as the fine-tuning of the light frequencies used by the transmitter is necessary. The modulation algorithm should also ideally be flexible enough to accommodate a wide selection of photodiodes with their unique characteristics.
- As smart lighting systems are becoming common, it is becoming more important for a VLC modulation scheme to support different dimming and color changes out of the box [32]. Therefore the VLC system produced in this work should ideally support such features.
- There is a myriad of modulation schemes and ideas in the VLC sphere in use today. However, they usually come with particular strong points and weaknesses. It is important to evaluate the options to choose the appropriate ideas and build upon them.
- Since the target infrastructure of Visible Light Communications is typically lighting systems with the primary aim of illumination [26], it is important to take into account the interactions between the devices and the human body. Especially the potential eye and concentration problems that can be caused by the flickering of the light intensity and color as data transmission takes place are of utmost importance. The modulation scheme should avoid such a flickering effect regardless of the type and content of the data.
- To be able to compare the designed modulation scheme with other systems, an expansive accounting and calculation of the throughput and error rates of the system should be realized.

## 1.3 Contributions

In order to solve these challenges, this work provides multiple contributions. These are listed below:

- A review of the state-of-the-art research is presented on topics of using color in Pulse Position Modulation, different dimming mechanisms in Pulse Position Modulation, and the techniques for using alternative hardware in Color Shift Keying (focused, but not limited to photodiodes). The positives and drawbacks of different methods for each topic are discussed. Existing Pulse Position Modulation variants and algorithms are investigated for their use in a color-coded data communications environment.

- Based on this research, a novel modulation scheme is developed which combines certain aspects of different state-of-the-art schemes, providing efficiency improvements and meanwhile integrating original ideas about how to encode data in the color spectrum. The novel modulation scheme, named Colored Multiple Pulse Position Modulation (CMPPM), builds upon the heuristic encoding/decoding algorithm developed for Multiple Pulse Position Modulation by Wu et al. [32] by expanding it into a system capable of transmitting data in an arbitrary number of simultaneous payloads in the color spectrum, called *channels*.

- The modulation scheme is implemented using commodity hardware. The achieved system is described in detail for reproducibility, and the code is made available open-source with both the modulation algorithm and the implementation details.

- A performance evaluation of the system is presented with theoretical calculations for the *ideal* throughput achiveable with CMPPM. The evaluation includes comparisons with relevant existing modulation schemes, such as Adaptive Multiple Pulse Position Modulation and Color Shift Keying.

## 1.4 Thesis Content

In this thesis work, a broad and complete overview is provided with regards to the state-of-art-systems in the related areas, the theoretical background of the novel modulation scheme, implementation details, as well as the evaluation of the created system in both theoretically achievable as well as the practically achieved throughput and error rates. The contents of each chapter are summarised below:

- **Introduction** has explained the challenges and aims associated with this thesis work.

- **Related Work** is a review of the state-of-the-art research in the relevant fields such as Pulse Position Modulation and Color Shift Keying necessary to take into consideration while creating a novel system. A discussion of crucial VLC constraints such as flickering mitigation is presented.

- **Colored Multiple Pulse Position Modulation** presents the theoretical background for the novel modulation scheme developed as part of this work as well as a detailed explanation of the encoding and decoding algorithms used. The use of the color spectrum in CMPPM for higher throughput is expanded upon.

- **Implementation** provides an overview of the hardware and software implementation of the presented modulation scheme using commodity hardware such as a photodiode, LED, and easily accessible microcontrollers. The PC and embedded codebases necessary to control the microcontrollers are also presented in detail with explanations of important design decisions. The software is implemented using Rust.

- **Evaluation** outlines the throughput achievable with the modulation algorithm under different conditions theoretically. After an investigation of the theoretical performance of this modulation scheme, a series of experiments are presented exploring the throughput as well as the error rate achieved with the practical implementation of the system presented in the previous chapter. A thin application layer for the modulation scheme is also implemented and introduced in this chapter to test the capabilities of the system for transmitting a text file with detection and re-transmission of corrupted and missed packets.

- **Conclusion** summarizes the results while provides ideas for future work based on this project.

# Chapter 2

# Related Work

## 2.1 Visible Light Communication (VLC)

Recent advancements in the power efficiency of LED technology have led to a revolution in the indoor lighting market as well as a rapid increase in the share of illumination provided by LEDs. Compared to earlier illumination technologies, LEDs provide the advantage of being capable of switching rates beyond what was possible in the past, opening up possibilities for using regular illumination installations for data communications without creating any discomfort for the human and little to no extra energy use on top of what is already consumed for illumination purposes [22].

The Visible Light Communication field opens up broad opportunities for overcoming the increasing congestion of radio frequencies as a rapidly increasing number of applications are allocated frequency bands by regulatory bodies for purposes of wireless communications. The promise of the field of VLC mainly lies in the ability to use the light infrastructure already installed in living and office spaces for illumination purposes. However, since unlike radio frequencies the human body is directly perceptive of the changes in illumination, there are certain constraints both theoretically and experimentally shown to be important to avoid causing harm or discomfort to humans. Also as LEDs are commonly used in Smart Lighting systems, in order to see practical adoption the theoretical frameworks for VLC need to integrate seamlessly here as well.

### 2.1.1 Flickering

The most important practical constraint on VLC systems is the perceptibility of the flicker effect caused while transmitting data which inevitably has to regularly change certain properties of the light such as intensity or color in predetermined manners. IEEE 802.15.7 standard defines duty cycles of less than 200Hz as flicker region, where eye safety might be of concern and duty cycles higher than this value should be used in order to make changes between symbols imperceptible to the human eye and avoid any adverse effects [25]. Another important category of flickering is inter-frame flickering, where subsequent data frames might have noticeably different light emissions, causing a perceptible flickering effect. While intra-frame flickering is relatively easy to avoid by

simply using high clock rates while transmitting data, inter-frame flickering requires conscious design choices on the part of the modulation scheme such that the changes in light conditions between frames are not perceptible.

### 2.1.2 Dimming

Since LED light systems are increasingly used for mood regulation and entertainment, it is becoming common for consumers to demand products that can freely change color and dimming levels. The IEEE 802.15.7 standard [25] mentions dimming support as an "important consideration for VLC for power savings and energy efficiency. It is desirable to maintain communication while a user arbitrarily dims the light source". Therefore in this work, great importance is placed on achieving different colors while transmitting data and making the system compatible with the ability to dim the light to arbitrary levels.

### 2.1.3 Pulse Position Modulation

Pulse position modulation (PPM) is a digital modulation technique that conveys data by varying the position of the pulses in a sequence of equally spaced time intervals [24]. In PPM, a pulse is transmitted in a specific time slot, and the information is encoded by changing the position of the pulse within the time slot, as shown in Figure 2.1. The position of the pulse is varied according to the symbol being transmitted. The width of the pulse remains constant, and only its position changes.

PPM has the advantage of being highly efficient with regards to the bandwidth if the desired action is to communicate data through pulses, in a manner that does not produce a visible flickering effect regardless of the modulated data, unlike schemes such as On-Off Keying. However, PPM can be very susceptible to timing errors because it relies on accurate timing to determine the position of the pulse. In the presence of noise or refraction, the timing of the pulses can be distorted, causing errors in the detection of the pulse position and leading to errors in decoding the transmitted data [4]. The transmitter and the receiver systems need to be in near-perfect time synchronization with each other since otherwise, the clock drift in one system can cause the pulse position to drift over time, leading to errors in decoding the transmitted data. It is also therefore preferable that the PPM systems have a mechanism for regaining synchronization once lost.

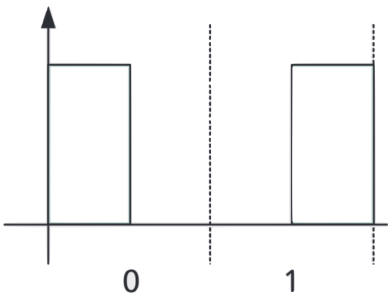


Figure 2.1: **Basic 2-PPM [25]**

## 2.2 Using the Color Spectrum in PPM

In order to determine the state-of-the-art techniques in using the color spectrum in PPM to achieve higher data rates, significant papers in this field are summarised below. The main criterion for choosing a paper as relevant was its discussion of a novel and distinct technique to achieve this.

- **Color in PPM using Lasers** [11]: This is the earliest article I have come across which mentioned the idea of increasing the data rate of PPM communication through the use of color. The authors, in 1985, are not yet concerned with Visible Light Communication applications but with direct-detection optical communication systems using lasers and photodetectors. However, they observe that by using multiple photodetectors each with a color filter, the throughput of the PPM algorithm can be easily scaled up. They also mention the possibility of increased error this might introduce into the system and propose Reed Solomon encoding, arguing that even with this overhead the system can achieve higher throughput.

- **PPM in CSK for Better Synchronization and Added Data Rate** [12]: Identifies an issue with proposed CSK systems, namely that they typically assume perfect synchronization between the transmitter and receiver nodes, even though this is not a trivial task in practice. Therefore the authors propose transmitting CSK constellations at the rising edges of PPM pulses as a method of easy synchronization between the nodes. This also adds extra information to each symbol. However, in practice, since this requires dark sections between pulses where no CSK constellation is transmitted, the increase in data rate is only theoretically achievable by redefining a symbol from a CSK definition of a single pulse, to a PPM definition of a combination of multiple pulses. Furthermore, through the use of the Variable Pulse Position Modulation (VPPM), they achieve dimming support in CSK. VPPM's use in dimming support is investigated further in the next sections.

- **CSK and Complementary PPM** [23]: Focuses on a system of transmission where CSK symbols are transmitted to a single photodiode receiver instead of multiple receivers as envisioned in the IEEE Standard [25], using the time domain to encode further information. The adapted method of utilizing a single photodiode for color information, however, is to implement band-pass filters to correspond with the expected constellations. This is a valid method while using CSK since the frequency band of each constellation is strictly defined, however not a workable solution for modulation systems that aims to achieve greater flexibility concerning color bands to use in tandem with PPM.

- **Color and Dimming Control with VPPM Coding** [9]: Appends color and dimming control techniques to Variable Pulse Position Modulation through multiple LEDs of different colors and controlling the width of the pulses via PWM controllers. The exact color produced in each VPPM symbol is determined by changing the proportion of pulses produced by each LED. The level of dimming is controlled by the pulse widths, which are uniform for all pulses in a symbol. The color spectrum is not used

for transmitting extra data so the system described in this work would produce the same data rate with only a single color LED but without any flexibility concerning color and dimming levels.

- **PPM over Each Color Component of CSK** [19, 20]: Explores pulse-coded signals over each color component on CSK modulation. This achieves better time synchronization between the nodes, dimming control, and reduces the system complexity while reducing the throughput to a degree. Three bands are chosen from the seven wavelengths available in the CSK bands, and each of these modulates data via PPM. As each symbol contains exactly one pulse per band, the resulting symbols still conform to the constraints of CSK modulation.

- **Color-and-Intensity Shift Keying for VLC** [6]: Takes a different approach and instead of the position of the pulse, it uses the amplitude of the pulse via combining CSK with an amplitude modulation scheme known as UPAM. While this is not entirely relevant to our research, their reasoning for this choice is still valid and important: Combining CSK with any type of PPM scheme can never actually increase the practical data rate of the system. We have already observed that when other related research such as [12] claim increased data rates due to the extra bits added to each CSK symbol thanks to the PPM, this is in reality only possible because they redefine a *symbol* from a CSK constellation to a PPM symbol. In this thesis as well, it will be described in Chapter 5 how the data rates of our system can never be faster than an equivalent CSK scheme. However, it is still possible to achieve similar throughput with additional benefits that the CSK scheme does not provide.

Adding the color spectrum into PPM schemes can serve different purposes such as modulating more data or simply obtaining variable color lighting while maintaining the basic transmission principles of PPM. The techniques for modulating data with color also differ. Combining an altered CSK modulation scheme with PPM is seen as beneficial by many research projects since CSK presents a ready-made solution to the problem of color flickering (which would likely be introduced by a simpler color modulation scheme), while PPM provides advantages concerning time-synchronization between the communication nodes. Overall, the properties seen as desirable in a colored PPM modulation scheme by most research can be listed as:

- Can facilitate different color points and can (in some cases) change the color during the operation. Modern smart LED lighting applications rely increasingly on complicated color features to differentiate themselves in the market [32]. Therefore this feature is crucial for such lighting systems.

- Does not flicker. This is usually achieved by arranging the possible color points in such a way that their average is the desired color over a predictable time frame, such as a symbol. It is also important to adopt a method of data modulation that ensures the frequency of color points is independent of the underlying data. Since real-world data is seldom entirely random, there can be a flickering effect if large blocks of identical

color points are repeated. CSK modulation, for example, solves this issue by scrambling the data before modulation and un-scrambling after demodulation [1].

- Can transmit extra data thanks to the addition of the color spectrum, such that the extra complexity and possibly the extra hardware in the system can be justified. In addition to using the color spectrum effectively, an important consideration seems to be the modifications to the underlying PPM scheme. If a slower PPM scheme has to be used to accommodate the color spectrum, then the benefits of the color spectrum might be lost.

- Provides an easy manner of maintaining time synchronization between the transmitter and receiver nodes. Since the data is transmitted via the rising and falling edges of light pulses, which are easy to detect accurately and precisely, the receiver node in the system can always recover the clock of the transmitter node without any complicated synchronization scheme.

## 2.3 Dimming Support in PPM

On top of increasing the data rate of PPM using the color spectrum, this work aims to achieve this outcome while also allowing arbitrary levels of dimming when the scheme is implemented with the right hardware. Dimming is recognized as one of the main challenges in Visible Light Communications [25], as it enhances power efficiency, and for many users of commodity LED hardware it is an essential feature for personal comfort and entertainment reasons. It was observed that there are different approaches for dimming with inherent trade-offs, and thus some of the more significant methods are discussed below.

- **Overlapping Pulse Position Modulation (OPPM)** [8]: Uses OPPM in order to achieve flexible dimming which can outperform several other PPM schemes. OOPM is a variant of PPM where multiple pulses are arranged in blocks of a given size, and the beginning position of the block is utilized to encode data. The blocks *overlap* in the sense that as long as the rising time of a block in a symbol is distinct from all other symbols, there is no limitation on how long the block can be. Essentially this acts as a way of encoding a similar amount of data to regular PPM while having more control over the length of the pulse since pulses of different symbols can *overlap* on the time domain. Because the length of the pulse is then freely adjustable and directly correlates with the light intensity, this work takes advantage of this property in order to adjust the dimming levels using the length of the pulse. However, OPPM suffers from the problem that it is inherently not capable of achieving high data rates compared to other PPM variants such as MPPM.

- **Overlapping Pulse Position Modulation (OPPM) with Pulse Width Modulation (PWM)** [5, 13]: Have a similar approach to the previously mentioned paper, but in order to achieve dimming, pulse intensity is adjusted using PWM, without having to adjust the properties of the modulation algorithm for dimming purposes. This achieves a much more fine-grained dimming compared to dimming using the parameters of the

OPPM modulation since hardware peripherals usually allow for very precise adjustments of the PWM duty cycle. However PWM requires additional hardware in the system which increases cost and complexity.

- **Variable Pulse Position Modulation (VPPM)** [9]: Uses a variation of PPM that is specifically designed for achieving a dimming level other than the fixed values of the regular PPM. This is achieved by a small modification to the pulse width of the PPM, which can be extended or shrunk in size using PWM. While this can achieve fine-grained dimming rates, a disadvantage of VPPM is that it relies on PWM hardware with the problems, as discussed above.

- **Multiple Pulse Position Modulation (MPPM)** [18]: Uses yet another PPM variant for controlling dimming with modulation variables. MPPM modulation posits known number of pulses in each symbol where each possible combination of pulses map to a certain pattern of bits to express data. The number of pulses in the packet can be increased or decreased to change the dimming level. However unlike the OPPM dimming system proposed in [8], this change has an influence on the data rates. As explained in Section 3.2.3, having more or fewer pulses than 50% of all positions cause a steep decline in the data rates.

- **VPPM with on-the-fly adjustments** [30]: Desires to achieve a system with flexible dimming levels while still using VPPM, since in a regular VPPM system the receiver needs to know the exact dimming level of the signal in advance in order to not miss detection of the pulses. Multiple ideas are presented in the paper for how to overcome this limitation including using neural nets. While this aim is achieved, it is accomplished at the cost of high system complexity.

There exists a wide range of PPM variants that support dimming by modulating the data with fewer pulses. However, this often presents a problem since dimming often comes at the cost of lower throughput since using fewer pulses inevitably decreases the number of possible symbols. A more beneficial approach is to leave the underlying PPM scheme untouched and make the pulses smaller or larger according to the desired dimming. This is achieved either using Pulse Width Modulation (as in VPPM) or by varying the pulse length (as in OPPM). PWM has the slight disadvantage of introducing more complexity to the transmitter circuitry and increasing the cost.

## 2.4 Multiple Pulse Position Modulation

Multiple Pulse Position Modulation (MPPM) is a variation of the regular PPM, using multiple pulses in each symbol at different positions in order to encode more data [29]. While regular PPM uses one pulse per symbol, an MPPM symbol contains multiple pulses, encoding more data per symbol in this way. This results in an increased throughput compared to naive PPM since there are more degrees of freedom in each symbol as shown in Figure 2.2.

The trade-off inherent in using MPPM instead of PPM for achieving higher throughput is that since we relinquish control over the positioning of pulses


Figure 2.2: **(a) 3 separate 2-PPM symbols transmitting 3 bits of data, (b) MPPM symbol where $N = 6, K = 3$ transmitting 4 bits of data**

to increase the degrees of freedom in symbols, we might be causing a flickering effect. The flickering possibility is intuitive when one thinks of a situation where $N = 4, K = 2$, and both pulses are in the first two locations of the symbol. Such a symbol is, in terms of the transmitted light, actually equivalent to a 2-PPM symbol at half the transmission speed. As the effective speed of switching the LED on and off decreases, it becomes more likely that the LED is perceived to be flickering.

## 2.5 Color Shift Keying

Color Shift Keying (CSK) is a modulation scheme specifically designed for usage in VLC communication, proposed in IEEE Standard 802.15.7 [1]. The core factor behind CSK is the relatively high speed of switching that can be achieved with three separate RGB LEDs as opposed to blue LEDs with yellow phosphorous LEDs, as the latter are more restricted in their ability to toggle at high frequencies due to their physical properties [10]. White light can be obtained in this manner by mapping symbols with specific frequencies which averages to white light when transmitted in quick succession. The idea to use multicolor LEDs in combination to create different frequencies of light is at the base of CSK, which is also the aim of this work. Therefore, the fundamentals of CSK modulation are explored and explained with respect to how they relate to similar challenges faced in this study.

In CSK modulation, seven distinct color bands are determined over xy coordinates defined by the CIE 1931 color coordinates, as shown in Figure 2.3. Out of these color bands, combinations of three vertices are used to form constellation triangles, the centers of which form the CSK symbols as described in Figure 2.4. The number of constellations determines the amount of data that can be encoded in an N-CSK symbol. A certain bit pattern is assigned to each constellation, and the number of bits that can be represented by a constellation is $\log_2(N)$. For example, each symbol in a 4-CSK system encodes two bits. Due to the constraints imposed by flickering, the data has to be scrambled before encoding into symbols to ensure randomness [25], and therefore to avoid data with repeating blocks from producing visual effects noticeably by the human eye.

### 2.5.1 CSK with Alternative Hardware

One perceived issue with CSK as defined in the standard is its relative complexity and rigidity in terms of hardware and software blocks required [20] with

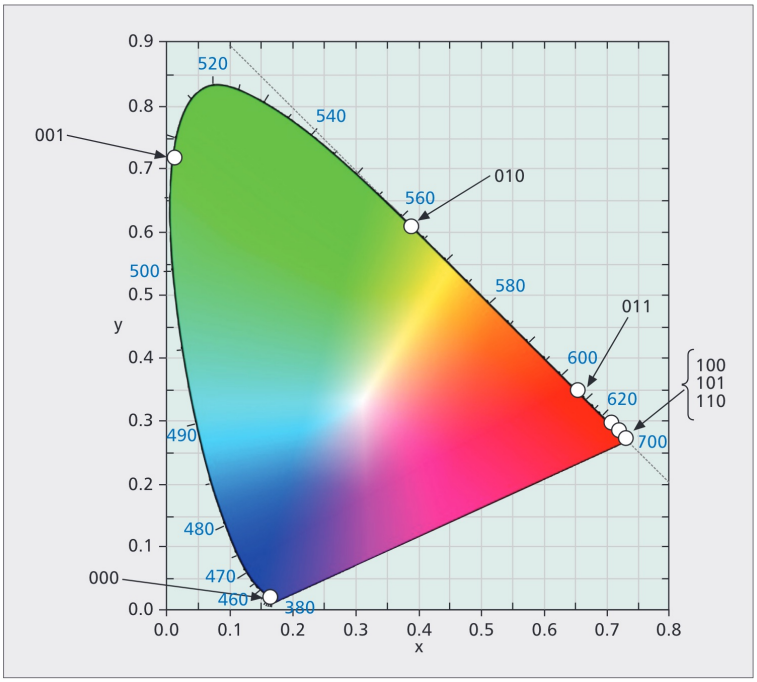


Figure 2.3: **CIE 1931 xy color coordinates. The three-digit values refer to the center wavelength of the seven bands defined in the IEEE 802.15.7 band plan. [25]**

multiple light sources and receivers. Therefore there is a body of work exploring implementing CSK, or modifications of CSK, using alternative hardware setups. As it is in the interest of this work to sense the color spectrum for encoded data using the simplest and most efficient hardware possible, a summary of a wide array of ideas regarding this field is presented in this section.

- **Camera** [16, 15]: Inspects the feasibility of using a camera for CSK communications and develops techniques to deal with the relatively significantly low speed of a camera in sensing an LED in comparison to a dedicated sensor. Most cameras are simply a combination of a matrix of sensors scanned in lines, and this technical detail can be utilized for CSK sensing. A fast-blinking LED leaves an image pattern similar to a barcode, from which the original LED blinking pattern can be recovered. Error correction techniques are also needed to be implemented as there are bound to be gaps between camera frames during which some symbols are lost. While this technique can find exciting applications due to the abundance of cameras in personal devices, it also imposes a steep increase in system complexity. Furthermore, even when cameras are used in this manner, they still have relatively low sensing rates compared to dedicated sensors.

- **LED** [31]: Even though this system uses simple On-Off Keying (OOK) as its modulation scheme, it develops on top of an interesting idea to use an LED not just for transmission but also for light detection. It has the

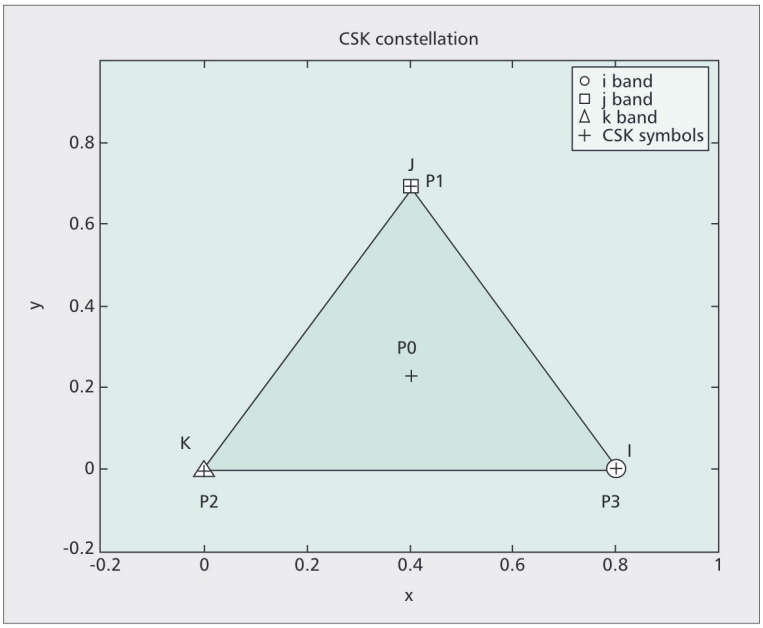


Figure 2.4: **Constellation design rule for 4-CSK. The corners correspond to three points chosen from the seven bands, while P0 is the centroid of the triangle and acts as the constellation. [25]**

advantage of achieving two-way communications between nodes with relatively low levels of complexity since only a single LED is needed per node to implement both ends of the communication. It could potentially also be extended to a CSK system by implementing three white LEDs, all behind color filters of red, green, and blue, and therefore able to transmit and receive the entire color spectrum. However, this forfeits the simplicity benefit of using an LED, since three LEDs would have to be working in tandem for transmitting and receiving. This system can be more complicated than using a single multi-color LED in combination with a color sensor.

- **Single Photodiode** [2]: Develops a constellation design strategy for a single photodiode-based CSK system. The focus of the design strategy is to maximize the minimum distance between the symbols, as sensed by a photodiode. As one can only get a single dimension of information from a photodiode (the voltage response), in order to come up with optimal constellation design, complex optimization algorithms are developed. The authors observe that the average white light constraint for CSK can be a limiting factor and offer methods for relaxing the constraint or even adding an extra LED to relax the limitations on the signal-to-noise ratio. While this work concerns CSK constellations, the idea of sensing the color spectrum with a photodiode by carefully crafting the symbols to create voltage responses with a uniform distribution can be applied elsewhere.

- **Light-to-Frequency Converter** [21]: Focuses on a low-cost CSK sys-

tem through the elimination of the need for analog-to-digital converters. Therefore the focus is on adapting a light-to-frequency converter to act as a CSK receiver, as well as a particle swarm optimization algorithm for designing CSK constellations for the limitations of the light-to-frequency converters. The scope of this work is similar but relatively more limited to the previously explored Anwar and Srivastava (2020) [2].

- **Four LEDs** [27]: Enhances the regular CSK modulation by adding an extra LED with the color cyan. The fourth LED allows forming a *quadrilateral* constellation shape instead of the usual triangular constellations of the CSK. Further tweaking the CSK algorithm to use the newly added LED allows the system to achieve a small SNR gain. A higher SNR leads to a better trade-off between the operating range and data rate, and can even lead to reductions in the overall system cost if fewer LED clusters need to be used to achieve the same system.

- **Screens** [14]: Focuses on communications involving a screen and a camera, such as sending data between two smartphones. The work has to cope with the fact that both the screen refresh rate both the camera scanning speed are substantially lower compared to dedicated hardware. Deep convolutional neural networks are implemented to lower the error rate of the system.

One common concern that most papers inspected for examples of implementations of CSK with alternative hardware have to deal with is the need to develop methods for designing constellations that can work with the limitations of the alternative hardware. Originally, CSK constellations were designed with color sensors in mind. Therefore, the original constellation picking algorithm [1] can lead to situations where constellations cannot be distinguished by the receiver if an alternative receiver is used. For example, the distinct color combinations that lead to two distinct constellations can trigger the same voltage response in a photodiode depending on the photodiode's voltage response graph. Therefore new constellation points are picked with the receiver hardware in mind. Furthermore, the CSK limitations around achieving average white light constraint create a significant limitation to calculating new constellation points, and some research papers explore creative ways to relax this constraint while avoiding color flickering.

# Chapter 3

# Coloured Multiple Pulse Position Modulation

The main contribution of this work is to propose a novel scheme that expands Pulse Position Modulation into the color spectrum, while not compromising on the constraints necessary for VLC systems to be compatible with human health and comfort via adverse effects. This is achieved by transmitting multiple streams (called **channels** in this work) of MPPM encoded data simultaneously through the extra dimension the color spectrum provides. Figure 3.2 provides an illustration of this idea. This chapter defines the modulation scheme and the accompanying packet structure to encapsulate data and convert the given data bits into signals in the color spectrum.

## 3.1 System Overview

The modulation scheme described in this chapter is designed to be compatible with simple and cheap commodity hardware, with the transmitter consisting of a single multi-color LED and the receiver a photodiode and op-amp circuit connected to an ADC. Figure 4.1 depicts an example implementation on a high level. The hardware choices impose certain restrictions on the modulation scheme, especially at the receiver end. The primary constraint tackled is to design a modulation scheme that can communicate data in the color spectrum while only using a photodiode since photodiodes are typically significantly faster at sensing than color sensors, even though color sensors are the usual devices for communicating data through the color spectrum. As discussed in relation with the Anwar and Srivastava (2020) [2] paper, one method for this is to define symbols that create photodiode responses with a uniform distribution such that the receiver can easily distinguish the symbols.

A CSK-inspired implementation of this method would create a mapping between photodiode voltage values and bit combinations. However, it should be taken into account what would happen if such a modulation scheme was transmitting a repetitive data section, such as a long chunk of zeroes. Since all the symbols would be the same, there would be a visible change of color and light intensity i.e. flickering. This effect is undesirable, and therefore in our modulation scheme an alternative approach is taken with data encoded in larger blocks using an

algorithm initially invented for MPPM.

Data communications between several parties always require an agreement over the rules of accessing the transmission medium called Medium Access Control (MAC) in order to avoid clashes between the transmitting nodes. In this work, the focus is solely on unidirectional communications between two nodes, therefore MAC or similar higher-layer protocols are not considered while developing the modulation algorithm.

## 3.2 Colored Multiple Pulse Position Modulation

Multiple Pulse Position Modulation (MPPM) was introduced in Section 2.4. Using this scheme as a basis, a new PPM variation is introduced in this work which can use the color space for encoding data without any additional hardware. We call this novel modulation scheme *Colored Multiple Pulse Position Modulation (CMPPM)* as we implement an MPPM variant that can also function through the color space to increase the data throughput.

### 3.2.1 Heuristics Based Algorithm for MPPM Encoding

The common method for translating between bit patterns and PPM symbols is to determine a one-to-one mapping between each possible bit pattern and a symbol. While this method is simple and can be implemented with a look-up table and therefore achieve $O(1)$ encoding and decoding speed, the size of the memory needed for larger symbols increases exponentially and becomes unsustainable. For example, an MPPM packet with $N = 100$ and $K = 50$ will have $\binom{100}{50} \approx 10^{29}$ possible states. Such a mapping is not sustainable to store in a regular computer's memory in the form of a look-up table.

Therefore, at the heart of this project lies the application of the heuristics-based modulation algorithm introduced by Wu et al [32] for encoding large MPPM packets. The starting point of the paper is to recognize the importance of adapting to ambient light conditions for any smart lighting product since such consumer products are becoming commonplace and typically require dimming support. The authors introduce a new scheme called *Adaptive Multiple Pulse Position Modulation (AMPPM)* to support fine-grained dimming levels while maximizing throughput and avoiding flickering. AMPPM is adaptive: It can change its parameters (number of positions $N$ and number of pulses $K$) due to dimming demands on-the-fly and therefore change the light intensity. The number of data bits encoded in a codeword of $N$ positions and $K$ pulses is $\log_2(\binom{N}{K})$. The relation between these variables and throughput is displayed in Figure 3.1.

Instead of mapping bit patterns to symbols, AMPPM uses a heuristic algorithm based on *combinatorial dichotomy* to calculate mappings on-the-fly without storing them in the memory. The encoding algorithm generates the symbol (*codeword*) moving from the least significant bit toward the most significant bit of the codeword. The input data is treated as a numeric value (so 128 data bits are handled as an arbitrary integer of 128 bits). At the beginning of the algorithm, $N$ equals the total number of positions and $K$ equals the total number of desirable pulses in a given codeword. Setting or unsetting each bit in the codeword reduces the number of possibilities that can be represented by the

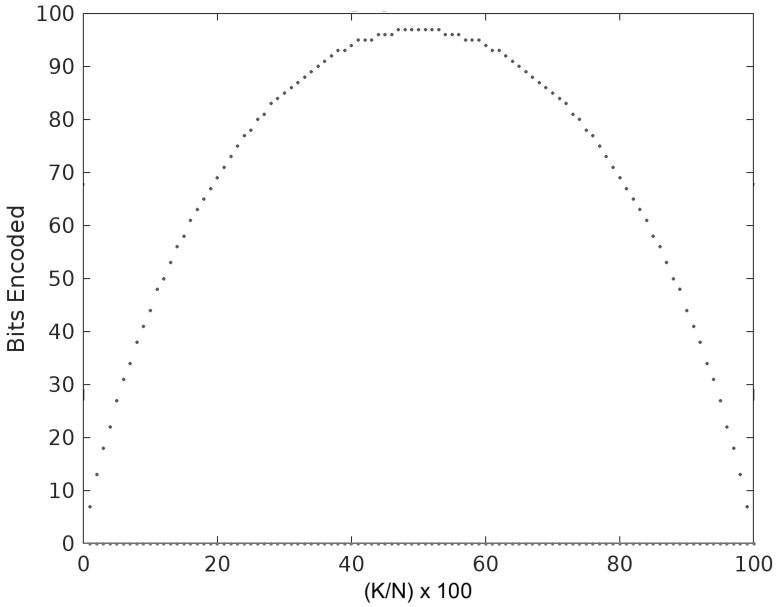


Figure 3.1: **Relation between encoded bits and $K/N$ ratio for MPPM**

rest of the codeword by $\binom{N-1}{K-1}$. This dichotomy is essential to the algorithm and allows the reconstruction of the codeword back into data later. At each step, the algorithm either sets the bit to 1 if the value of the input data is smaller than $\binom{N-1}{K-1}$, or else sets the bit to 0 and subtracts $\binom{N-1}{K-1}$ from the input data. At each step, the $N$ becomes smaller since we have fewer positions to fill, and if the pulse is set to 1, $K$ also becomes smaller. The algorithm does not need to continue further when either of the parameters is completely processed:

- All $N$ positions in the codeword are processed or,
- The prescribed number of pulses ($K$) are encoded in the codeword.

The decoding algorithm functions the opposite way: operating over each bit in the codeword and reversing the decisions taken by the encoding algorithm to rebuild the input data. Since the input data was deconstructed by subtracting the appropriate $\binom{N-1}{K-1}$ value at each 0 bit in the codeword, the decoding node can reconstruct it by adding the correct $\binom{N-1}{K-1}$ at each step whenever there is a 0 bit in the codeword.

Both algorithms are presented with the improvements contributed by this work in Algorithms 2 and 3. The improvements are detailed in the following sections.

### 3.2.2 Simultaneous Multiple Channels of MPPM

This work introduces the idea of transmitting multiple payloads of MPPM-encoded data simultaneously using color space. We name each of these MPPM

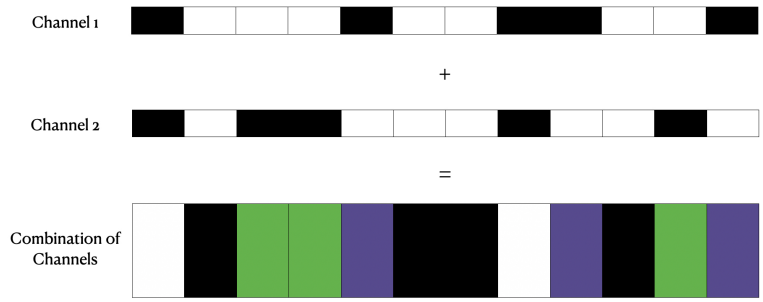


Figure 3.2: **Two channels combined into four channel combinations (expressed as distinct colours)**

symbols a **channel**. One or more channels are transmitted simultaneously multiplying the throughput of the system as illustrated in Figure 3.2. In theory, there is no limit to how many channels a system can have although practically noise represents a soft limit for increasing throughput by adding more channels. The limitations are investigated in Chapter 5.

It is useful to allow each channel to encode data independently of other channels since this achieves the highest possible data rate from such a system. If the channels were constrained by each other, the encoding method would have to take these constraints into account, which would reduce the throughput. However, transmitting data simultaneously on independent channels necessitate the definition of a transformation function that can express a combination of pulses from different simultaneous channels as a single output value in the physical realm. Therefore, we also introduce the concept of **Channel Combination** to define the combined output of any (including zero) number of channels. While channels are an abstract concept, channel combinations have a one-to-one mapping to light frequencies and thus photodiode readings. We define certain constraints on them:

- Any permutation of the channels where one channel is not present more than once, including the lack of any channels, i.e., the empty set ($\emptyset$), must be defined as a channel combination.

- Each channel combination must correspond to a color frequency that produces a distinct voltage response in the photodiode, and the distance between subsequent combinations must be as uniform as possible. This constraint ensures that for the receiver each channel combination is distinguishable from the subsequent channel combinations as much as possible.

- The channel combinations must be ordered from the combination producing the lowest voltage response to the combination producing the highest voltage response.

Through these constraints, we can achieve a modulation algorithm where an arbitrary number of MPPM channels can be transmitted simultaneously, and therefore the data throughput scales linearly with the number of channels.

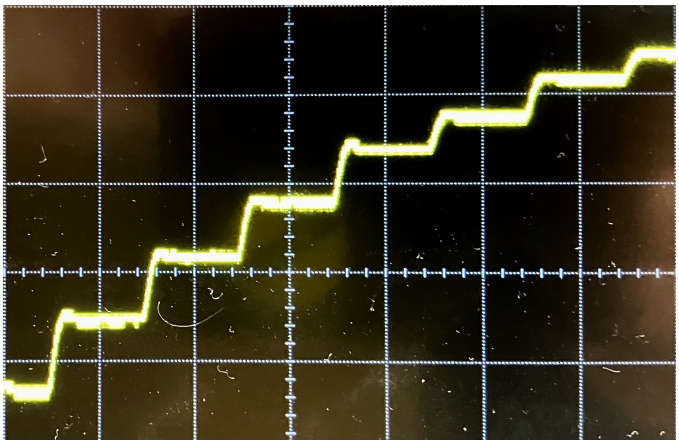

Figure 3.3: **Observed Voltage Response from 3 Channels and 8 Channel Combinations**

In practice, there can be various methods of achieving distinct colors for channel combinations and fulfilling the constraints. There is typically a trade-off between system complexity and control over the exact values of channel combinations. The simplest and cheapest method is to connect multiple LEDs to the transmitter, each with a distinct color. The LED colors and intensities can be chosen to create a distinct and uniform spread between the channel combinations. In Chapter 4, an alternative method is implemented by connecting the colored RGB LEDs to PWM drivers and thus achieving precise control over the intensity of each channel combination.

An example of 4 channel combinations from 2 channels can be seen in Figure 3.2, where different channel combinations are expressed as a distinct color.

### 3.2.3 Improvements on the Heuristic MPPM Algorithm

While implementing the CMPPM scheme, certain weaknesses and possible improvements were identified in the heuristic algorithm for MPPM introduced in the Wu et al [32]. In this section, three improvements are presented with improvements over time complexity, simplifications, and increases in the efficiency of the packet structure by introducing an alternative optional dimming method (which provides that we do not need a special *compensation* part in each packet).

#### Lower Time Complexity

The encoding and decoding algorithms have both $O(N^2)$ time complexity since the algorithm has to iterate over every pulse position in the worst case and calculate a combination value based on input $N$, which takes $O(N)$ time. However, calculating a new combination value for every bit in the codeword from scratch is unnecessary. This operation can be reduced to $O(1)$ complexity using the information we already possess from the last bit. After the initial calculation of $\binom{N}{K}$, every next bit uses a value of either $\binom{N-1}{K-1}$ or $\binom{N-1}{K}$. This fact can be utilized as shown in the Algorithm 1.

**Algorithm 1** Find next combination in $O(1)$ time

**Require:** $N > 0$
**Ensure:** $C = \binom{N-1}{K-1}$ or $\binom{N-1}{K}$
  $C \leftarrow \binom{N}{K}$ ▷ This is known from previous iteration
  **if** Both $N$ and $K$ subtracted **then**
    $C \leftarrow C \times K$
  **else if** Only $N$ subtracted **then**
    $C \leftarrow C \times N - K$
  **end if**
  $C \leftarrow \frac{C}{N}$

This method of calculating the combination value requires slightly more bookkeeping as each iteration needs to keep track of the last $\binom{N}{K}$ value. But it can have a significant effect on the run times of both the encoder and decoder algorithms at higher $N$ values and can have the advantage of enabling operations on significantly less powerful and therefore less expensive pieces of hardware.

**Algorithm 2** Optimised encoding algorithm

  Data $\leftarrow$ number representation of input data
  $i_N \leftarrow 1$
  $i_N \leftarrow 1$
  $C \leftarrow \binom{N-1}{K-1}$ ▷ Calculate this only once
  Code $\leftarrow [Off; N]$
  **while** $i_N <= N$ && $i_K <= K$ && $i_N - i_K <= N - K$ **do**
    **if** Data $\geq C$ **then**
      Data $\leftarrow$ Data $-C$
      $C \leftarrow$ decOnlyN$(C, N - i_N, K - i_K)$
    **else**
      Code$[i_N] \leftarrow On$
      $C \leftarrow$ decBothNandK$(C, N - i_N, K - i_K)$
      $i_K \leftarrow i_K + 1$
    **end if**
    $i_N \leftarrow i_N + 1$
  **end while**
  **if** $i_N - i_K \geq N - K$ **then** ▷ fill the remaining bits if need be
    **while** $i_N <= N$ **do**
      Code$[i_N] \leftarrow On$
      $i_N \leftarrow i_N + 1$
    **end while**
  **end if**

### Optional PWM Dimming

The number of data bits that can be encoded in a channel is given by the formula $\log_2(\binom{N}{K})$. Due to the properties of the combination operation, the highest throughput is achieved when $K = \frac{N}{2}$, and the throughput decreases as $K$ deviates towards 0 and $N$. This change is demonstrated in Figure 3.1.

**Algorithm 3** Optimised decoding algorithm

```
Data ← 0
i_N ← 1
i_N ← 1
C ← (N−1 choose K−1)                          ▷ Calculate this only once
Code ← the received codeword
while i_N <= N && i_K <= K && i_N − i_K < N − K do
    if Code[i_N] == Off then
        Data ← Data +C
        C ← decOnlyN(C, N − i_N, K − i_K)
    else
        Code[i_N] ← On
        C ← decBothNandK(C, N − i_N, K − i_K)
        i_K ← i_K + 1
    end if
    i_N ← i_N + 1
end while
```

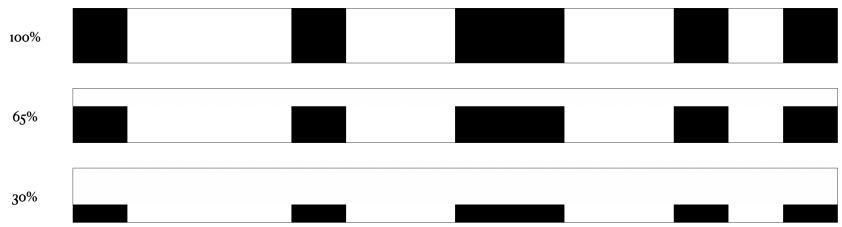


Figure 3.4: **An example of a channel being dimmed at three different PWM duty rates**

AMPPM scheme relies on changing the number of pulses in the codeword to facilitate dimming, which has the advantage of providing fine-grained dimming support in exchange for increased complexity and a header in every packet to announce the $N$ and $K$ values since the receiver cannot know these numbers in advance. However, it is possible to achieve a dimming method with less overhead if the transmitter has access to PWM hardware such that $N$ and $K$ values can stay constant for every packet. Dimming can be provided by connecting the LEDs to the PWM peripherals and transmitting lower PWM duty cycles. This method is similar to the VPPM scheme, applied in a larger domain. A visual representation is given in Figure 3.4.

A disadvantage of dimming in this way is that the maximum brightness achievable from the system is 50% of the capacity of the LED since even with no dimming using PWM, each channel can put out ON pulses only half the time.

But just like the AMPMM dimming method, the method of dimming via changing the PWM duty rate is not free in terms of throughput. If the environmental conditions and the number of channels are kept constant, the upper range of voltage values which can be used for channel combinations is going to decrease with higher dimming levels. Such a change inevitably decreases

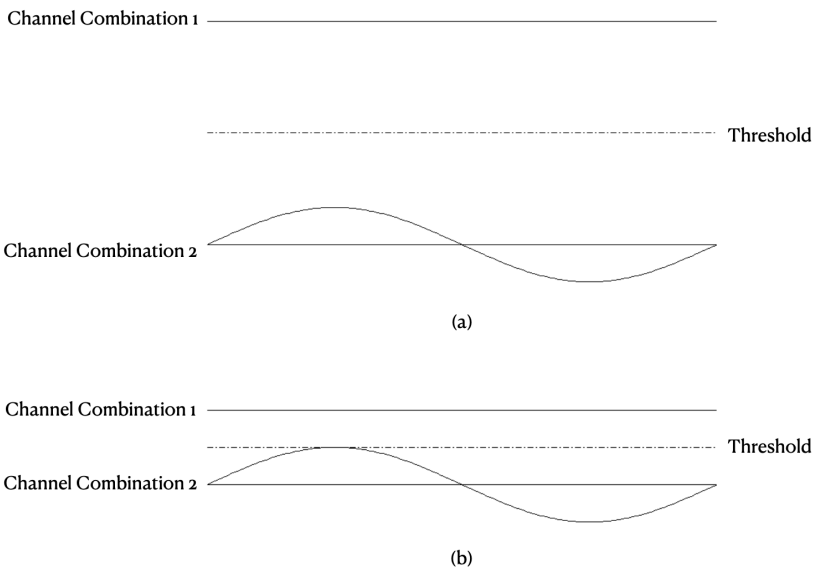


Figure 3.5: **(a) No dimming, (b)** 50% **dimming. Noise is represented with a sin wave. Any value above the threshold interpreted as the first channel combination, while any any value below the threshold is interpreted as the second channel combination**

the noise robustness of the system since the distance between channel combinations becomes smaller, and there is a higher chance that the receiver fails to distinguish between the subsequent channel combinations as demonstrated in Figure 3.5. Therefore, depending on the ambient noise conditions, there is an empirical threshold for dimming where transmitting data using fewer channels increases the throughput due to the positive effect of a higher signal-to-noise ratio overtaking the negative effect of using fewer channels.

### Packets Without Compensation Overhead

A disadvantage of the Adaptive MPPM method adopted by the SmartVLC system is the need to transmit the $N$ and $K$ parameters for each payload in the packet header since these variables can be adapted and changed between packets for dimming purposes. Therefore the receiver needs to be made aware of the parameters before being able to decode the codeword into data. This adds overhead to the transmission twice, both while transmitting the parameters but also later, when a *compensation* section has to be appended to the frame to compensate for the flickering effect from transmitting the parameters as they do not conform to the dimming level of the payload. Especially at lower dimming levels, the pulses required for parameter transmission need to be compensated by long compensation sections to re-attain the correct dimming level inside the packet. The PWM dimming method adopted in this work has the advantage of not sacrificing throughput with a compensation section since the $N$ and $K$ parameters are always the same between subsequent packet payloads. Therefore

| Preamble | Training | Payload | RS |
|---|---|---|---|
| 2 Bytes | $2^c$ bits | X | Y |

Table 3.1: **Packet Structure ($c$ denotes channel count). X and Y are independent from each other and can be any value**

there is no need for a header.

### 3.2.4 Optional Multiplexed Communications

As each channel carries information independently of the others, the transmitter can distribute data between the channels, and the receiver is free to sense some of the channels and perhaps ignore others. This architecture opens up opportunities for multiplexed communications where each channel can encode data targeted for a different receiver, and the receivers record data from only the desired channel(s).

Even though it is possible to create a multi-receiver system with most VLC schemes (due to the large illumination field of view of LEDs), CMPPM is especially suitable because there is no need for any overhead in labeling data for each targeted receiver. The modulation scheme contains multiple channels natively.

## 3.3 Packet Structure

Data is communicated enveloped inside packets to guarantee important properties such as in-order delivery, integrity, and acknowledgments of successfully arrived packets. The structure of the packets is shown in Table 3.1 and explained in this section.

### 3.3.1 Preamble

Before the communication between the nodes starts, it is necessary to synchronize the transmitter and the receiver and inform the receiver of incoming data. The typical method for this is a preamble pattern transmitted at the beginning of each packet. The preamble pattern should be easily distinguishable from regular ambient noise. Therefore it is designed as a series of pulses that rapidly alternate between total darkness and maximum channel combination in order to be noticeable above the unpredictable fluctuations of the ambient light. In order to avoid any coincidences where a regular data payload is mistaken to be the preamble, the preamble section needs to be sufficiently long such that the chances of an identical pattern occurring in the data payload is low. Two bytes (16 pulses) were determined to be sufficient for practical purposes.

### 3.3.2 Training

Factors such as physical distance, angle of incidence, and interference from different light sources or reflections can cause variations in the voltage reading each channel combination creates in the receiving photodiode. It is difficult to predict how these factors vary during communication as the physical conditions should always be assumed to be unpredictable. Schemes such as CSK deal

with this problem by tightly defining the physical properties of the signals. However, there is a great deal of flexibility to be gained by only mandating that the signals differ from each other in *relative* terms and this work choose to follow this path instead. Therefore, in practice, it is necessary to develop a method for identifying each channel combination from the voltage readings of the photodiode even when the levels might be in unpredictable change due to changing environmental conditions.

Typically it is safe to assume that under most conditions, the external physical factors are likely to stay at predictable levels during the communication duration of each packet while showing larger fluctuations between several packets since at high transmission speeds each packet is transmitted in a minuscule time frame. Therefore a useful method for compensating against environment changes is to transmit a **training** sequence in each packet to *train* the receiver node to the correct voltage level for each channel combination.

In the training section, one pulse is transmitted per channel combination from the lowest to the highest order. For example, assuming that a given system has three channels and these correspond to red, green, and blue, and the red channel creates the highest voltage response and blue creates the lowest, the order of channel combinations can be seen in Table 3.2.

| Order | Channel Combination |
|---|---|
| 1 | $\emptyset$ |
| 2 | Blue |
| 3 | Green |
| 4 | Blue, Green |
| 5 | Red |
| 6 | Red, Blue |
| 7 | Red, Green |
| 8 | Red, Green, Blue |

Table 3.2: **An example channel combination ordering where channels correspond to primary colors**

Given that the steps between the channel combinations are uniform, it can be observed that the training sequence contains pulses of each channel for exactly half of the available slots. For example, in the given example four blue, four green, and four red channel pulses are transmitted. Therefore, for each channel pulses are transmitted for exactly half the possible slots ($K = N/2$ ratio). This is the same as in the rest of the packet, therefore the training part of the packet has the same average intensity and color as the rest of the packet and does not cause any flickering. The training sequence of a real packet can be seen in Figure 3.3.

### 3.3.3 Payload

The payload section is obtained by applying the modulation algorithm to enough raw data for each channel. In effect, the payload is several MPPM packets put on top of each other in several channels, and transmitted in parallel. At each pulse of the payload, several channels try to transmit a pulse simultaneously, which is resolved to a *channel combination*. Therefore the payload can also be

conceptually understood as a sequence of channel combinations as depicted in Figure 3.2.

The payload length can be adjusted according to the error characteristics of the communications medium and the desired signal-to-noise ratio. A longer payload can be more efficient in terms of data transmitted per packet but also likely requires more error correction as there is a higher chance that the conditions of the communications medium change during the payload sending. If the communications medium is subject to intermittent total failures such as objects blocking the path between the transmitter and the receiver, then a shorter payload can be more efficient since there is less chance that a long packet is disrupted in half.

### 3.3.4 Forward Error Correction

Errors while communicating through any physical medium are common and have to be accounted for. Therefore it is desirable to correct errors or at least be able to verify the integrity of the signal at the receiver node. Even the detection of an error without the possibility of correction can still facilitate retransmissions. Forward Error Correction schemes achieve this by appending an error correction sequence to the data derived from the data itself. While many different schemes can serve this purpose, most are based on properties that allow for detecting corruption or even recovering the original data through mathematical operations as long as the data is not corrupted beyond a certain threshold. The error correction sequences are essentially overhead, and after the data is corrected or verified, the section can be stripped away.

In this work, Reed-Solomon Encoding is utilized for this purpose. Reed-Solomon encoding adds redundancy to the message by generating a set of symbols based on the payload, each symbol being eight pulses in this case. The system is programmable with regards to how many bytes of forward error correction are appended to the data. This directly affects the error correction potential as detailed in Section 3.3.4.

There are several reasons for choosing Reed-Solomon encoding over similar schemes:

- When a symbol is defined as a series of several pulses, Reed-Solomon encoding is also quite robust to burst errors, which occur when a series of consecutive bits are corrupted [17]. This makes the scheme particularly suitable for this work, as we observed that corruption of a pulse often means an electrical or light distortion also corrupting nearby pulses. Therefore a burst error that affects a contiguous sequence of symbols will likely affect only a small subset of the check symbols, and the remaining check symbols will still be able to correct errors in the message.

- Apart from their correction capabilities, Reed-Solomon codes can also detect corruption at significantly higher corruption levels even when it is not possible to salvage the data anymore. This is useful for avoiding processing corrupted data.

- Although Reed-Solomon encoding is based on advanced mathematics, it has wide use in both academic works and industrial products. This popularity translates itself to the wide availability of well-tested ready-to-use

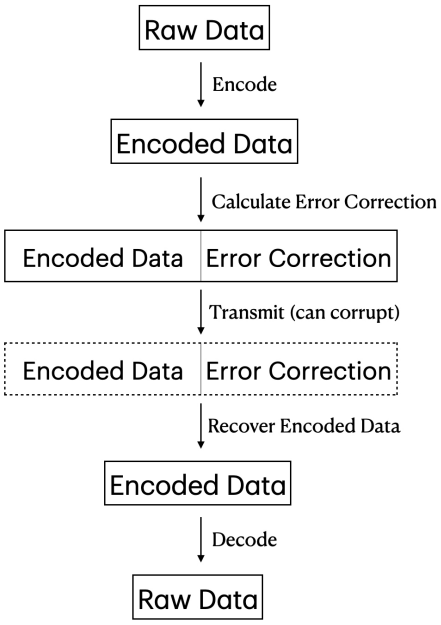


Figure 3.6: **Journey of data through encoding, error correction and decoding steps**

software libraries which can be a significant benefit since the main contribution of this work is not related to the forward error correction schemes.

Each packet contains payloads from several channels, which are encoded and decoded independently. The implementation of the Reed-Solomon encoding also follows this pattern, as the encoding process is applied to each channel separately. The error correction encoding is applied to the already modulated data. Calculating the encoding symbols from raw data would not achieve the desired outcomes since the heuristics-based data modulation and demodulation algorithms are highly unstable to corruption, and it is impossible to correct the result of demodulating corrupted signals. A single bit flip in the encoded payload would result in a completely unpredictable change in the underlying data. Therefore the Reed-Solomon encoding is applied after the modulation step, and the Reed-Solomon correction and verification steps are applied to the received signal before demodulation. The demodulation algorithm is applied to the signal only after it has been corrected and verified, and so the resulting raw data can be presumed to be correct. This is described in Figure 3.6.

If the packet is appended with an error correction section, it should be compensated to achieve a 50% pulse ratio like the rest of the packet to avoid flickering.

### Error Correction Ratio

As described in Section 3.3.4, the symbol used for Reed-Solomon Encoding in this work is bytes (eight pulses), which means errors are corrected and detected on the level of bytes instead of individual bits. Therefore the calculations

regarding the number of correctable symbols are also presented using bytes.

The general formula for Reed-Solomon error correction capability is known to be $t = (n - k)/2$ where,

- $n$ is the total number of symbols in the encoded message, therefore the payload plus the RS encoding symbols.
- $k$ is the number of original data symbols in the encoded message.
- $t$ is the maximum number of errors that the code can correct.

Reed-Solomon encoding can also detect up to $2t$ symbols of error even though such levels of error cannot be corrected. Error detections is valuable for discarding the corrupted packet and demanding a repeat sending. It is also likely that a packet corrupted beyond $2t$ symbols will fail the Reed-Solomon code check. However, this is not guaranteed and the payload can be labelled as correct in rare cases.

## 3.4 Average Light Intensity

While describing the packet structure in the previous section, it is noted for each packet section how the pulses add up such that for each channel, the 50% ratio is maintained between the available slots $N$ and realized pulses $K$. Therefore the average light intensity of the entire packet stays constant between different sections and as well as between subsequent packets. Furthermore, the average color of each section and each packet is also constant as each channel is represented an equal number of times. Therefore the human eye does not perceive any flickering in either the intensity or color output of the system during data transmission.

# Chapter 4

# Implementation

This work also includes an implementation of the CMPPM scheme. The implementation's goal is to showcase the viability of the modulation idea in a performant real-world system, and ultimately provide a springboard from which further development on this topic can take place. All software created as part of this project is available and open-source, with accompanying notes on how to build and run the code on `https://gitlab.com/Master_thesis4`.

The implementation is divided into two parts: **peripheral** and **host**. Host refers to the computers on both ends of the communication sending and receiving files or other data. Conceptually the hosts could be any nodes sending and receiving any data using the communication interface introduced in this work. However, in this section, they are implemented as code that is supposed to run on an operating system. The peripherals are the microcontrollers responsible for outputting the information to the physical medium via LEDs and sensing the data from the physical medium via a photodiode. The classification between receiver and transmitter is then orthogonal, as both the receiver and the transmitter consist of one host and one peripheral node.

## 4.1 Hardware Design

This section introduces the hardware design used in the implementation of this work. The overall structure of the hardware is depicted in Figure 4.1.

### 4.1.1 Transmitter

The transmitter hardware consists of a consumer-grade RGB LED connected to three Pulse Width Modulation (PWM) outputs of a NUCLEO-H743ZI development board. The board is connected to the host PC via USB, transmitting UART messages between the host and the microcontroller.

The host is responsible for dividing the source file of the transmission into chunks and encoding these chunks into packets. The transmitter peripheral, connected to the host via UART, receives the encoded data for each channel. The responsibility of the transmitter peripheral is to receive the bytes through UART and broadcast this information with precise timing to the LED. As standard UART messages consist of bytes, each channel is divided into chunks of eight

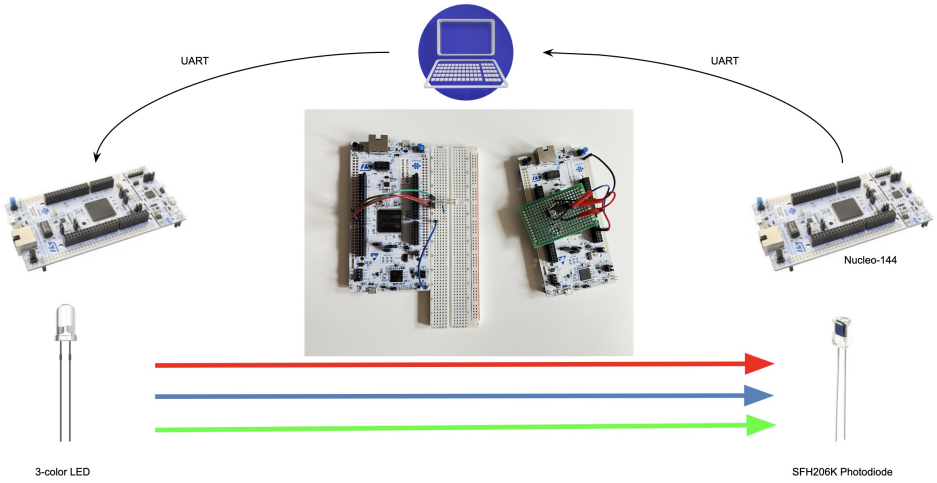


Figure 4.1: **Hardware Components**

ON-OFF pulses collected into bytes. Instead of sending all of the pulses from one channel, and then proceeding to send the next channel, the system transmits one byte from each channel before proceeding to the next byte. This structure has the advantage of more closely resembling the nature of the transmission itself, where all the channels transmit simultaneously. This structure translates to less need for buffering and wasted time between consecutive packets.

Different RGB LEDs typically have different characteristics concerning frequencies and intensities of the light they can transmit. The transmitter peripheral is designed such that the specific characteristics of the LED is not a limiting factor for the system, and any RGB LED hardware (or even 3 separate single-color LEDs) can be used in the transmitter circuit with tuning. The backbone of this flexibility is the ability to tune each LED color's PWM duty cycle for each channel combination separately in the software. Therefore the differences in LED and photodiode characteristics can be eliminated by adjusting the PWM duty cycles for each channel combination with respect to the voltage response it creates in the photodiode output. The ideal case is to aim for a perfectly separated set of steps in the voltage response, as exemplified in Figure 3.3. Such a structure has the benefit of creating the smallest possible risk for noise to cause an error in the receiver system, as each step stands at the maximum possible distance from each other.

However, the perfect distancing between channel combinations might not always be possible due to the available hardware. For example, one can alternatively design an implementation that does not rely on PWM hardware to create separation between channel combinations. In this case, one could choose LEDs and photodiode such that combining colors still creates channel combination steps, or attempt to regulate the brightness of each LED by changing the resistor values attached. The resulting steps can be less uniform, increasing the error rates. But for this work, the PWM hardware simplified the implementation and evaluation.

### 4.1.2 Receiver

A photodiode is a semiconductor component that converts light into an electrical current. It is a type of photodetector that operates by absorbing photons of light and generating a flow of electrons. By adding op-amps into the circuit, the minuscule amount of current generated by a photodiode can be transformed into a voltage value. Since the voltage output of this circuit scales with the light incident on the photodiode, the output can be sampled by an analog-to-digital converter.

Photodiodes are commonly found in various applications such as light detection, optical communication, and sensing. In this work, they are preferred over other types of photodetectors because of their high sensitivity, low noise, and fast response time. Their faster response times in comparison to color sensors are crucial for choosing photodiodes in this work. While the modulation scheme described in this work can also be implemented using a color sensor, the achievable transmission speeds would be considerably slower. Color sensors typically consist of multiple photodiodes with different color filters or spectral sensitivity. By comparing the responses of these photodiodes, the color sensor can determine the color of the incident light. However, the step of analyzing different photodiode responses can introduce delays.

The receiver consists of a photodiode connected to the ADC input of a NUCLEO-H743ZI development board. Mirroring the transmitter architecture, the microcontroller connects to the host via UART to transmit information about the received packets.

The receiver peripheral has only two responsibilities:

- It continuously attempts to sense the preamble of a packet. When a preamble is successfully received, a special byte indicating the start of a packet as well as the type of the packet (further explained in Section 4.3) is transmitted to the host, followed by the training values.

- After the preamble is complete and the exact length of the incoming packet is known, the peripheral is simply responsible for reading the voltage values outputted by the photodiode in regular intervals, in synchronization with the transmission interval of the transmitter peripheral. These values are transmitted to the host via UART when sensed. The receiver peripheral does not match voltage readings to color combinations. Instead, this operation is deferred to the host, since with a high number of channels, processing the voltage values to obtain the correct color combination can be a slow operation while sending a byte through the UART peripheral is predictable. If sampling the ADC and processing the value, and transmitting through UART takes too long, the timer interrupt might repeat before the last interrupt routine completes. This should be avoided at all costs.

Oversampling the input and filtering the data before processing is typical in communications systems. However, since the ADC peripheral of the NUCLEO board does this itself, this step is not included in our work.

Photodiodes do not create an equal voltage response to all light frequencies even when the intensity is the same. This fact is significant for our work, since otherwise it would be impossible to distinguish colors using a single photodiode. The relative spectral sensitivity of a photodiode depends on the specific

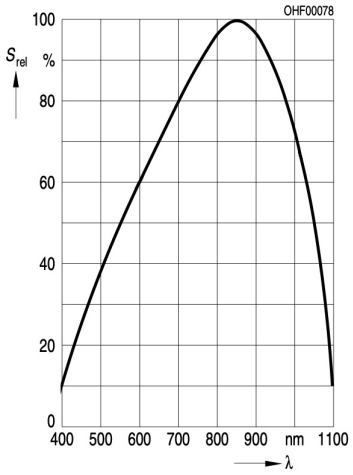


Figure 4.2: **Relative Spectral Sensitivity of SFH206K photodiode**

hardware. The mapping between light frequency and voltage response for the SFH206K photodiode used in this work is shown in Figure 4.2. However, the fact that light color as well as light intensity produces distinct voltage responses can be used as the basis of a color-based modulation implementation such as this work, while still taking full advantage of the speed benefits of photodiodes.

## 4.2 Software Design

The entire codebase for this project, including the peripheral as well as the host code was created using Rust. This choice was made on account of several distinct advantages of Rust over the more common solution of creating embedded software using C and the host software in a higher-level language:

- Rust offers a modern, ergonomic, and commonly accepted package manager solution that increases the ease of code reuse. The packet manager was quite beneficial in this project since the embedded and the host programs share a significant amount of code. Furthermore, the project required the use of several computationally complex data types. These include a big-integer-type to implement the modulation algorithm described in Algorithms 2 and 3, a first-in-first-out buffer at the transmitter peripheral and the mathematically involved algorithm behind Reed-Solomon Encoding discussed in Section 3.3.4. Rust's package manager cargo has readily available open-source libraries for all of these use cases, which simplified the development process substantially.

- While it is possible to create the host program using C, due to its archaic memory management and error handling styles, it is generally accepted that developing PC software using C is only desirable for the most performance-sensitive applications as the potential trade-offs in development speed and error propensity of the code is relatively high. On the other hand, it is possible to develop PC software in garbage-collected languages such as Java or Python. However, this would have the downsides

of unpredictable run-time speeds due to the inherent unpredictability of garbage collection on top of the overhead of developing and maintaining programs in two different programming languages which means the loss of easy code re-use. Rust eliminates this negative trade-off by offering a modern development experience on both platforms while approaching the run-time speeds and predictability of C or C++.

- Rust has a very active community working for the adoption of the language as the default programming language of embedded systems due to its desirable memory and type safety properties. An active community leads to the positive side effect of an abundance of well-maintained Rust hardware abstraction libraries for various families of popular microcontrollers. This work has benefited from an open-source library for easy interactions with the chosen microcontroller as well.

The software is organized into four separate codebases (*crates* as in Rust parlance). The host code for the transmitter and receiver are in a single crate, while the peripheral codebases are separated into two crates for technical reasons to be discussed. Since there is significant overlap in the fundamental structures and configurations used in all crates, a separate crate is created to function as a *common library*. The main design principle while separating functionality into crates was to move as much of the functionality as possible to the host crates, to benefit from the higher computing capacity of PCs over embedded microcontrollers.

### 4.2.1 Host

The host crate is where a large part of the functionality is implemented. The underlying assumption behind this decision is that the host code is expected to be invoked from machines with significantly more processing power and memory, and has access to abstractions such as a file system, dynamic memory allocations, and threads. These reduce development time while implementing multiple features to showcase the contributions of this work:

- A practical way to demonstrate the capabilities of the entire system in terms of transfer speed and correctness, is to send a large file from one node to another. Therefore a file system is a useful feature.

- The adopted method of keeping track of successfully sent packets and transmitting the remaining packets in the optimal order is to store the packets in the memory on both sides while the communication is still happening. However, this method requires large amounts of memory to store all the required packets, especially without a very complicated buffering system. Furthermore, the buffer's memory size depends on the file size. Therefore the ability to allocate large amounts of memory at run time is advantageous. This operation would be inadvisable in an embedded microcontroller.

- The modulation and demodulation algorithms introduced in Chapter 3 can be computationally heavy, especially since the algorithm can require arithmetic operations using a big-integer type if there are too many bits

to fit one of the integer types provided by the compiler. Arithmetic operations on big integers are inherently expensive since they cannot directly take advantage of a processor's circuitry, therefore tasks such as divisions can be substantially slower. As the peripherals transmit and sense information with hard real-time constraints, any computation with the potential to take unpredictable time is not acceptable to run on the peripheral. In a scenario where consecutive packets are being transmitted, if one of the peripherals cannot finish the algorithm during a timer interrupt due to this expensive computation, the receiver and transmitter might lose synchronization with each other. Therefore running the expensive algorithms on the host provides predictable performance where it matters.

- To test the integrity of the files transmitted during evaluation, an acknowledgment can be sent for each successfully received message from the receiver to the transmitter. Since during the evaluation the communication was taking place between two processes running on the same PC, the two processes communicating via the standard input and output simplifies testing. In order to not unnecessarily slow down the transmitter process modulating data and sending packets to the peripheral, a separate thread is spawned whose sole purpose is to monitor the standard input.

- Rust allows a crate to contain more than one file with a *main* function, although in this case, the programmer has to select which file is to be the basis of the compilation while invoking the compiler. Since they share significant functionality, the transmitter and receiver codebases are located in the same crate but within different main files.

### 4.2.2 Peripheral

The role of the peripheral software is to perform the tasks which require real-time performance as well as interacting with the hardware.

Both of the peripheral crates take advantage of *RTIC* and *probe-run* frameworks:

- **RTIC** (Real-Time Interrupt-driven Concurrency) is a Rust framework for building embedded applications that require real-time responsiveness. It is a Rust-based imitation of Real-time Operating Systems (RTOS) commonly used in embedded software. However, unlike most RTOS systems, the tasks do not maintain separate stacks and share the processor time as controlled by the RTOS. RTIC relies on associating each task with an interrupt available in the microcontroller and uses the interrupt to achieve periodical task invoking. While this can be a limitation, it was not a blocking problem for this project since neither the transmitter nor the receiver creates a large number of tasks. Therefore there was no need for a large number of interrupts and some could be spared for RTIC functionality.

- **probe-run** is a custom run script to simplify the process of building, flashing, and running Rust code as firmware on microcontrollers, as well as accompanying features such as testing on hardware and debug logging. It allows for running embedded software using the same processes as regular software.

Unlike the host crate, the peripheral codebases are separated into two different crates. This is due to a limitation of the probe-run tool. The run configuration system of the tool can be hard-coded with a serial ID of the desired microcontroller, which makes it considerably easier to work with two identical microcontrollers connected to the same host computer, as they can be distinguished with their serial ID number as flashing targets. However, the configuration is only available on a per-crate basi. Therefore it is not possible to take advantage of this functionality while keeping both of the peripheral codebases in a single crate.

Both peripherals handle most of their application logic inside the timer interrupt, which runs at the same frequency for both nodes. Inside this interrupt, the receiver continuously checks for new preambles, and when a preamble is received, subsequent readings are forwarded via UART to the host. The transmitter checks if it has a packet in the queue to transmit, and if there is data available, this is modulated with the correct frequency inside the timer interrupt. The transmitter peripheral additionally enables the UART interrupt to fetch data from the host, which is immediately placed in a queue to be processed later.

### 4.2.3 Common Library

Since two peripherals as well as the host software share a significant amount of functionality and configurations, a separate common library crate was created to be imported by all other crates and provide the code structures for the basic underlying building blocks of the system. These blocks are:

- **Channel**: A struct to encapsulate a channel.
- **Channel Combination**: Represents a combination of various channels. This is a useful abstraction since the receiver peripheral detects incoming messages as a series of channel combinations. For $n$ channels, there are a total of $2^n$ possible channel combinations, since the combinations also include the empty set.
- **Preamble**: The preamble is used for denoting the beginning of a packet and its type. Since sensing a successful preamble (and the training values associated with it) requires a stateful operation through multiple timer interrupts, this is implemented as a state machine. This state machine can be seen in Figure 4.3. For the transmitter node, the iterator pattern of Rust serves as an ergonomic and well-suited structure, iterating over the channel combinations required to complete the preamble. The advantage of collecting the software for sending and receiving the preamble in the same crate is the ease of creating integration tests for this complicated operation, such that a faulty change in the preamble process at the transmitter or the receiver does not cause an unchecked regression.
- **Configurations**: The codebase is implemented to be flexible with regards to configurations such as the number of transmission channels, length of the packet preamble, length of the payload and error correction sections, etc. The reasoning behind this flexibility is to provide the ability to test and evaluate different configurations concerning their effect on the theoretical bandwidth of the system as well as robustness to noise and error.

While these properties can vary between different setups, in a given setup they should be shared between all the components and are not prone to change during communications. Therefore it is a pragmatic solution to place the configuration variables in the library shared by all the components and define them as Rust constants.

### 4.2.4 Flow Control

In Section 4.1, it was mentioned that the communications between the peripheral and the host devices happen through the UART protocol. The typical method for handling incoming UART messages is to place the incoming bytes into a buffer for later processing by the program. In the host code, the buffering is handled by the operating system itself, and therefore the user code does not have to concern itself with buffering solutions. Furthermore, a PC's processing power and memory capacity are high enough that it is difficult to cause a buffer overflow when a PC is receiving UART messages. On the other hand, in embedded systems, the programmer has to take care of buffering. Due to the memory-constrained environment and the lack of a Memory Management Unit, this is often achieved by defining a statically sized buffer. Since the clock speeds of embedded processors tend to be significantly below that of a computer, this can mean that when receiving UART messages from a PC the embedded system might experience buffer overflows which can cause dropping data and corruption when the buffer is full. Flow control techniques are needed to avoid this outcome.

The flow control in this work consists of a boolean state maintained by the peripheral and communicated to the host. When the internal buffer of the peripheral reaches a high watermark, it communicates to the host (through a predetermined UART message) that the flow control should be turned off. The host is then responsible for periodically checking for this message and pausing with the transmission until the transmitter peripheral processes and transmitted enough data such that its internal buffer reaches a low watermark. At this point, another predetermined UART message is transmitted to communicate to the host that the peripheral is available to process more data. The benefit of turning the flow control on or off at watermark levels, and not when the buffer is full or empty, is to avoid situations where the buffer stays empty until a new packet is received, which causes the transmitter LED to stay off momentarily and therefore flicker.

## 4.3 File Length Packet

When transmitting a file by breaking the file into equally sized chunks and encoding them into packets, there is likely not enough raw data to fill the last packet. Since all packets are supposed to be the same length, it is still desirable to send a whole packet, but pad raw bytes with zeroes before modulation. However, if the receiving node does not know the exact amount of bytes to be received as part of the transfer, it also would not be capable of distinguishing padding from data. This limitation becomes a problem when transmitting an arbitrary file. Meanwhile, it is not an issue if the structure of the data contained in each packet is known.

For evaluation purposes, it is desirable to transmit files between PCs. When receiving a file, the receiver must be informed of the size of the file before the communications. To make this process dynamic and avoid hardcoding the file length, we can send a special packet at the beginning of each file transfer with the file length. As the file length can fit into a simple integer data type, this packet is defined to have smaller data and Reed-Solomon encoding parts. An extra bit is appended at the end of the preamble to indicate that this is a file-length packet.

### 4.3.1 Preamble and Training

The role of the preamble and training sections in the modulation scheme is described in Section 3.3.1. This section is going to detail the implementation. Preamble sending and receiving are both implemented entirely in the peripherals since this must be a real-time operation. Both sections are implemented together since they serve similar purposes and are transmitted adjacent to each other.

#### Transmitter

The preamble controller for the transmitter is implemented as an iterator that outputs a Channel Combination describing what should be transmitted in each pulse. It maintains an internal counter that is incremented after each call to the iterator and is used by the internal logic to determine the next pulse. The transmitter peripheral software also maintains an internal mapping of each channel combination to the appropriate PWM duty rates for each LED. These duty rates are determined empirically through measurements with the photodiode.

#### Receiver

Receiving the preamble is a relatively more complicated operation since the preamble can *fail* halfway if an unexpected reading is recorded. Since the preamble is simply an alternation between maximum and minimum channel combinations, the distance between subsequent pulses should be sufficiently large. The typical practice for implementing operations with a state changed through outside events is to define an explicit state machine. There exists a popular open-source crate in Rust called *smlang* that can generate code for the states and the events of a given state machine at compile-time, without the need to use heap memory and is therefore compatible with the embedded software of the peripherals.

The UML diagram of the preamble state machine is displayed in Figure 4.3. The assumption behind the state machine is that the receiver is constantly monitoring a noisy but relatively stable light channel, which does not trigger an ON-OFF series by accident, defined as a sudden jump (above a set threshold) in the observed voltage value. Even if such a change happens once stochastically, the chances of this event occurring multiple times in the given pattern by coincidence are low enough to be disregarded. The state machine has a fail condition if it receives an unexpected value. A fail resets all the internal variables and waits for a new complete preamble. After the preamble and the packet type are received, the next pulses to be received are the training values. Finally, the peripheral software has all the information necessary for correctly sensing the payload of the packet.

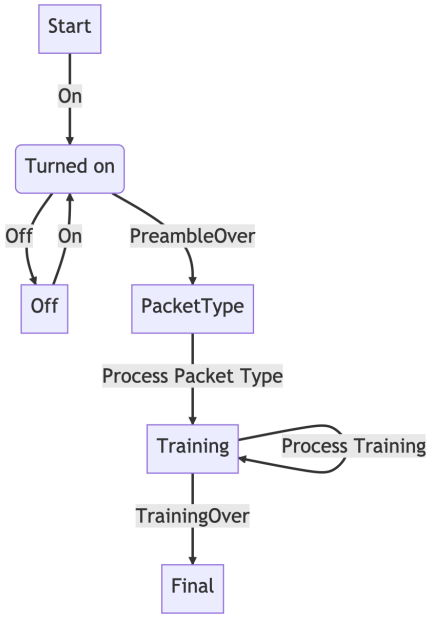


Figure 4.3: **The state machine of the Preamble receiving logic**

# Chapter 5

# Evaluation

## 5.1 Theoretical Throughput

To facilitate comparisons with similar works and modulation schemes such as MPPM, AMPPM, and CSK, this section calculates the *theoretical* throughput of CMPPM. The calculations do not take into account the optional implementation details such as the packet indexes, error correction, or preambles for any of the schemes. Perfect transmission conditions and hardware are assumed.

An important factor while considering the throughput of a system is the Symbol Error Rate (SER) since higher error rates mean one has to add more error-correcting code to each transmission and facilitate re-transmissions. While it is possible to theoretically derive this through techniques such as Monte Carlo Simulations [20], such calculations are not a part of this work. However, one should keep in mind that the SER inevitably increases as the number of channels increases and the channel combination voltage values approach each other, getting mistaken more by the receiver due to small variations introduced by environmental fluctuations. The effects of this are investigated in the next Section.

### 5.1.1 Single Channel

While using a single channel, this work is functionally equivalent to MPPM where a relation of $K = \frac{N}{2}$ exists between the number of pulse positions $N$ and pulses $K$. The number of bits that can transmitted in a packet with $N$ positions are plotted in Figure 5.1, using the formula $y = \log_2(\binom{N}{N/2})$. There is a linear relation between the two. Therefore for the calculations it is unimportant which $N$ value is chosen.

### 5.1.2 Multiple Channels

With multiple transmission channels, the number of bits that can be fitted in each packet also scales almost linearly. The only caveat is that the training section of the packet (Section 3.3.2) has to get longer for each extra channel which slightly reduces the data rate as some of the pulses are used in the training section, therefore the exact formula becomes $y = c \times \log_2(\binom{(N-2^c)}{(N-2^c)/2})$ where $c$ stands for channel count.

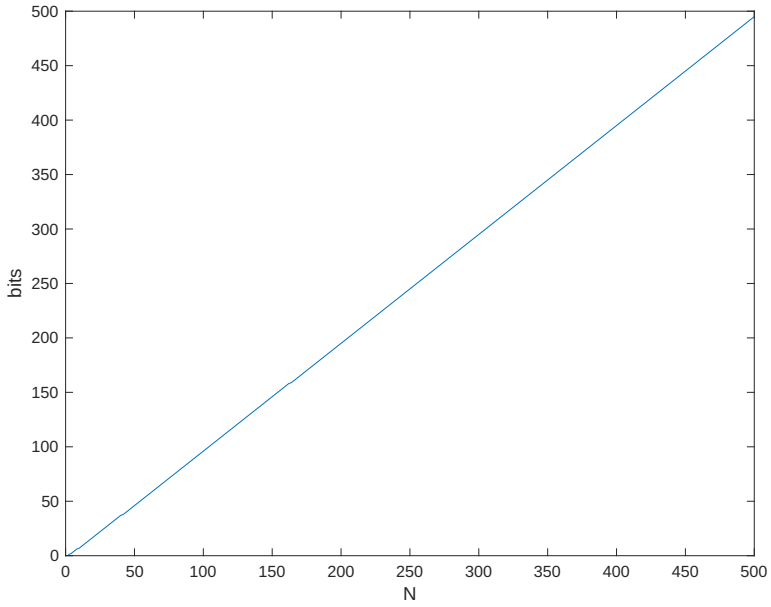


Figure 5.1: **The relation between $N$ and bits per packet in MPPM where $K = N/2$**

### 5.1.3 Truncation

As covered in Chapter 3, CMPPM can encode and decode data as non-byte aligned number of bits. Communicating with non-byte aligned bits increases the software complexity substantially since computers preserve data in bytes, and the change between two formats creates buffering and alignment issues. One solution to this problem for evaluation purposes is to truncate the number of bits contained in the raw data of each channel down to the nearest byte. The truncation has a slight negative effect on the practical throughput of the evaluated system. When the effect of the truncation is factored into the formula, the bit count becomes $y = c \times \lfloor \log_2(\binom{(N-2^c)}{(N-2^c)/2})/8 \rfloor \times 8$. Figure 5.2 showcases this formula with different $N$ and channel counts. It should be noted that at lower values of $N$, because of the increased training overhead, using extra channels can bring the throughput down.

### 5.1.4 Comparison with CSK and AMPPM

The bit rate of $N$-CSK is straightforward to calculate, as every *pulse* transmits the same amount of data: $\log_2(N)$ bits. Furthermore, there is no overhead for informing the receiver node of constellation points since they are predetermined and standardized. A comparison between the throughput of 8-CSK and 3-channel CMPPM is shown in Figure 5.3. The truncation and training cause a slight performance loss over CSK, but CMPPM still has advantages over CSK depending on the use case:

- CSK limits the possible constellation points and always aims to produce white light, unlike CMPPM. CMPPM can achieve a broad range of per-

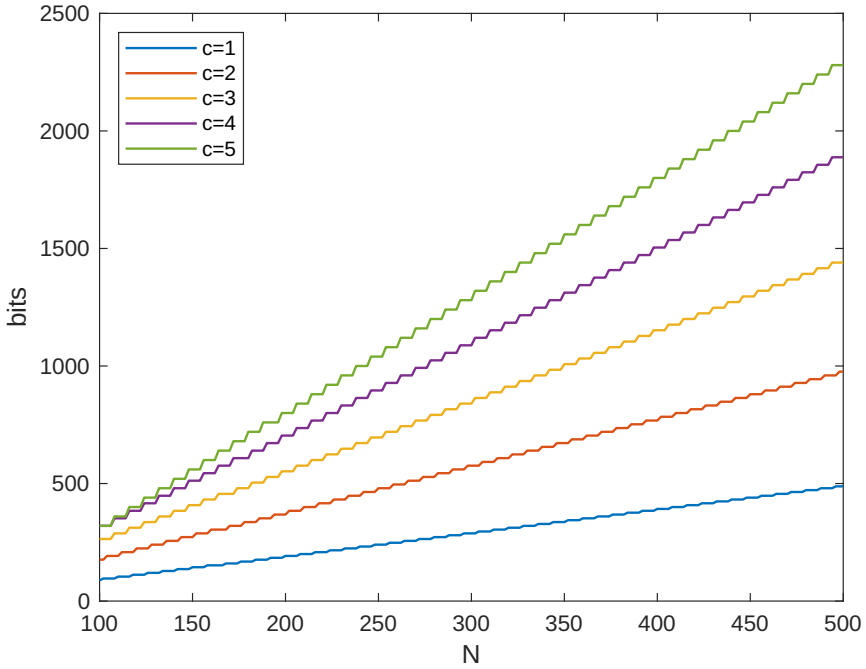


Figure 5.2: **The relation between $N$ and bits per packet achieved in CMPPM for different channel counts**

ceived colors without a prior agreement between the nodes. Each packet contains the channel combination values in the training section.

- PPM-based schemes have time synchronization advantages, as discussed in Section 2.5.1.

- As each channel in CMPPM transmits data independently of each other, it is possible to devise a single-transmitter multiple-receiver system where each receiver only pays attention to only one channel. Such segmented communications are not possible with CSK without a higher layer for labeling data meant for each receiver.

- Hardware requirements are much simpler compared to CSK, since the receiver can consist of a single photodiode.

Therefore, it can be concluded that the throughput cost of sending a training section as part of every packet bears certain benefits in other areas.

The frame format of AMPPM is displayed in Table 5.1. To facilitate a comparison with the best possible case of AMPPM, we can assume that the length and pattern of the packet are known by the receiver, while the preamble or CRC sections are not used. In this case, the compensation and sync bits are also not needed. Therefore only the payload is taken into account. In this case, the scheme is practically equivalent to CMPPM if we are using a single channel since we have removed all the *adaptive* properties of AMPPM. However, this comparison obscures all the improvements introduced by CMPPM:

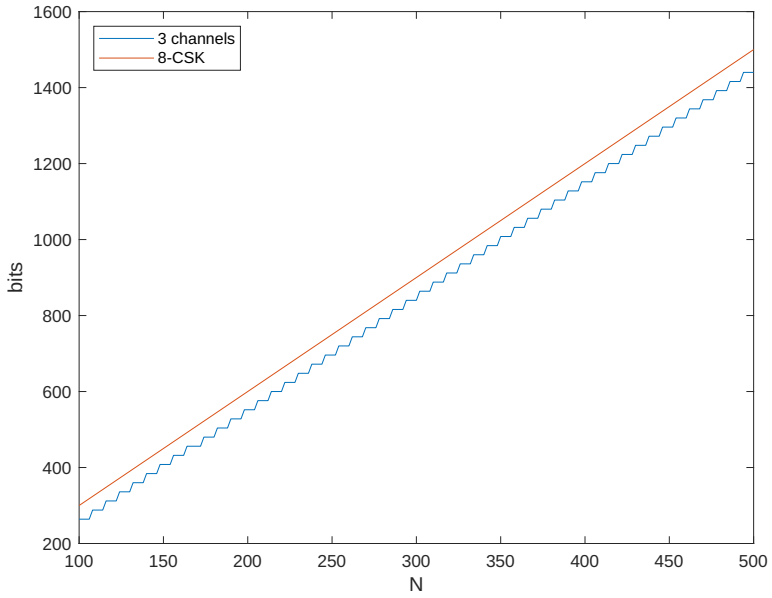


Figure 5.3: **8-CSK throughput compared it with CMPPM using 3 channels**

| Preamble | Length | Pattern | Compensation | Sync | Payload | CRC |
|---|---|---|---|---|---|---|
| 3 Bytes | 2B | 4B | X bits | 1 bit | 0-MAX B | 2B |

Table 5.1: **AMPPM Frame Format [32]**

- **Multiple channels**: AMPPM is limited to solely toggling an LED, and cannot use the color domain for data transmission.
- **No need for compensation**: Using PWM for dimming the output light to the desired level has the advantage that all sections of the packet can transmit at the same average light intensity as the data payload, including training, error correction training, and preamble. Therefore there is no need for a compensation sector in each packet.
- **Dimming without losing throughput**: Ensures that the throughput achieved by optimal 50% dimming in AMPPM can be maintained over a large range of dimming values before the SER becomes too large.
- **Fixed length**: Since AMPPM uses the dithering technique to achieve fine-grained dimming values, it needs to transmit the length of each packet since the payloads can have different lengths. The advantage of CMPPM is that since the payload length is fixed, transmitting the length does not take up space and does not need compensation.

Therefore a proper comparison should include the entire packet necessary for both modulation schemes at different dimming levels and channel counts. Since the number of bits required for compensation depends on the bit pattern used for the preamble, length, pattern, and CRC and therefore depends on the

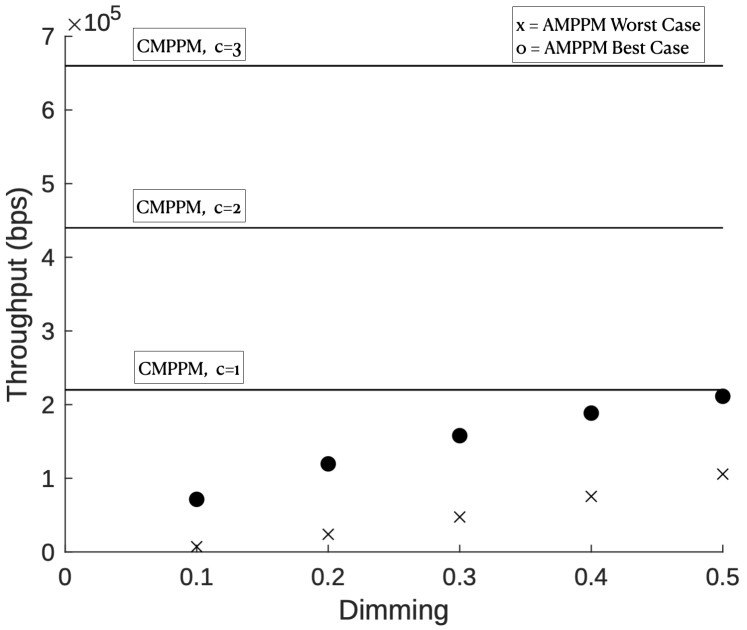


Figure 5.4: **AMPPM's best case (no compensation required) vs worse case (maximum compensation required) compared with CMPPM (1, 2, or 3 channels) when the payload is** 100 **pulses long for both. Preambles are not taken into account.** $f_{tx} = 250$ **kHz**

contents of the packet, a best-case scenario (no compensation) and a worst-case scenario (all the bits of these arbitrary parts are the same and therefore need to be compensated back to the dimming level) are calculated. The comparison can be seen in Figure 5.4. The actual throughput for AMPPM is somewhere between the best case and worst case. However, it is clear that at any dimming level, CMPPM can achieve a higher throughput than AMPPM even with a single channel.

## 5.2 Error Rates

While theoretical SER calculations are not within the scope of this work, it is still useful to gain insight into how the error rates evolve under different conditions, as this can give insights into the inherent capabilities of the system. All the experiments are conducted by sending the same pattern in the payload, without any error correction. Distance between the nodes, transmission angle, and number of channels are the analyzed parameters. The SER is calculated over the payloads of packets that reached the destination correctly with a correct preamble. If the channel combination of a pulse was ascribed wrongly to another channel combination, this is considered an error.

The aim of this evaluation is threefold:

1. To understand the practically achievable communication distance with the current hardware implementation under different conditions.

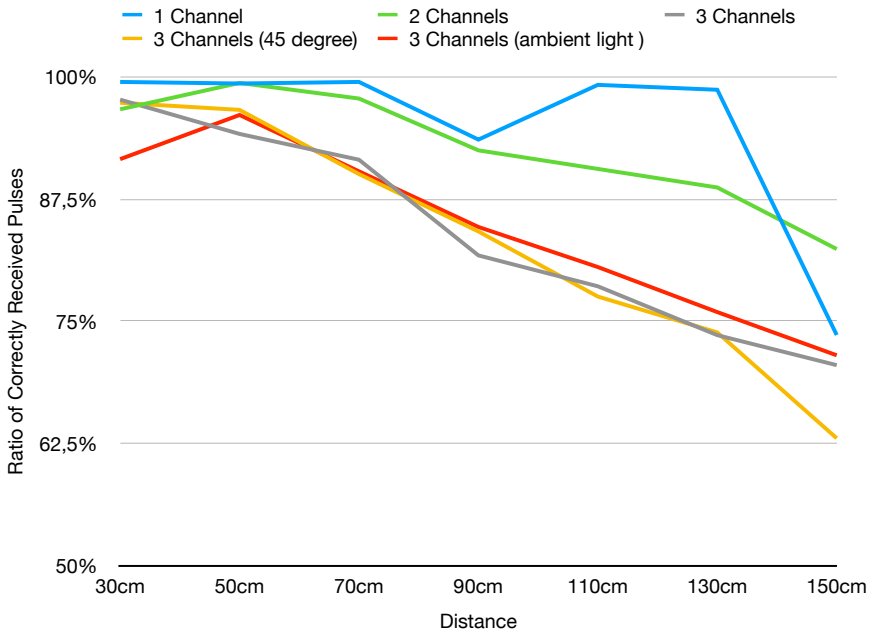


Figure 5.5: **Ratio of correctly received pulses per distance and channel count/transfer conditions**

2. To collect empirical data concerning the increase in SER as the channel count increases.

3. To observe if non-orthogonal angles of incidence between the transmitter and receiver are possible since multiple receivers standing at different angles towards the transmitter would be crucial for achieving a multi-receiver multiplexed communications system as described in Section 3.2.4.

Tests are conducted under five different conditions. Initially, a series of experiments were conducted using one, two, and three data channels with a direct angle of incidence between the nodes and no ambient light interference. Additionally, one set of separate tests was conducted with a 45 degree angle between the nodes and another set with moderate ambient light conditions (a sufficiently bright room, no direct light incident on the photodiode). Conditions with a high amount of ambient light were not included in testing since it was observed that the system was not able to function in this case. For each set of tests, the distance between the nodes was changed at set intervals, and the percentage of correctly received pulses for each channel was recorded. It should be noted that even with a 0% SNR, approximately 50% of pulses are still going to be received correctly since a pulse can either be high or low. Therefore 50% correct is the base case. For each leg of the experiment, a total of 100.000 pulses were sent and the number of correctly received pulses was recorded in the receiver node. The results are presented in Figure 5.5. A photo of the test setup can be seen in Figure 5.6, with the transmitter and receiver connected to the same laptop and separated by marked distances along a straight line.

The experiment displays a clear trend where adding extra channels to the

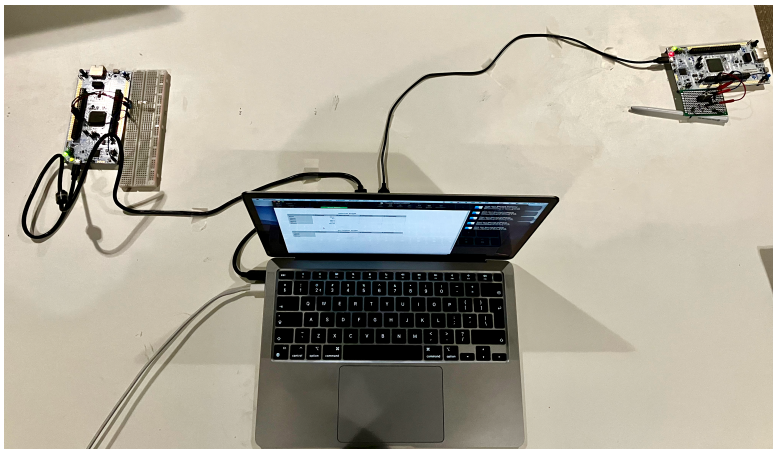

Figure 5.6: **The setup for error rate testing**

communication as well as distance increases the error rates. Since an LED is used as the light source, it is to be expected that the light carrying the signals diffuses with distance and leads to corruption. While a larger number of channels does not cause corruption in itself, it becomes more difficult to sense the correct channel combinations when the distance between channel combinations becomes smaller. However, even under worse conditions, the system can conduct communications effectively up to a distance of approximately 1.5 m. It is also of note that a 45 degree angle of incidence, as well as moderate ambient light, has no discernible effect on the error rates.

## 5.3 File Sending Speed

To demonstrate a practical use case of the CMPPM system, a series of additional experiments were designed, showcasing a file transfer between two nodes using different Reed-Solomon configurations and channel counts in optimal environment conditions: 10 cm distance between the nodes and no light noise. To achieve file transfer a *thin application layer* was created by appending an index number in the data payload of each packet, such that lost or corrupted packets can be re-transmitted until the file is complete, approximating the effects symbol error rate on the throughput. The average communications time for five large files ($\sim$ 50 kB) for each configuration was recorded. The results are presented in Figure 5.7.

The results are within the bounds of reasonable expectations, as each additional channel corresponds to an additional linear increase in the throughput. We observe that increasing the size of the error-correcting code in relation to the payload typically causes a lower performance due to higher overhead. However, this entails a trade-off since less error checking also implies a higher probability of undetected errors and corruption in the transfer.

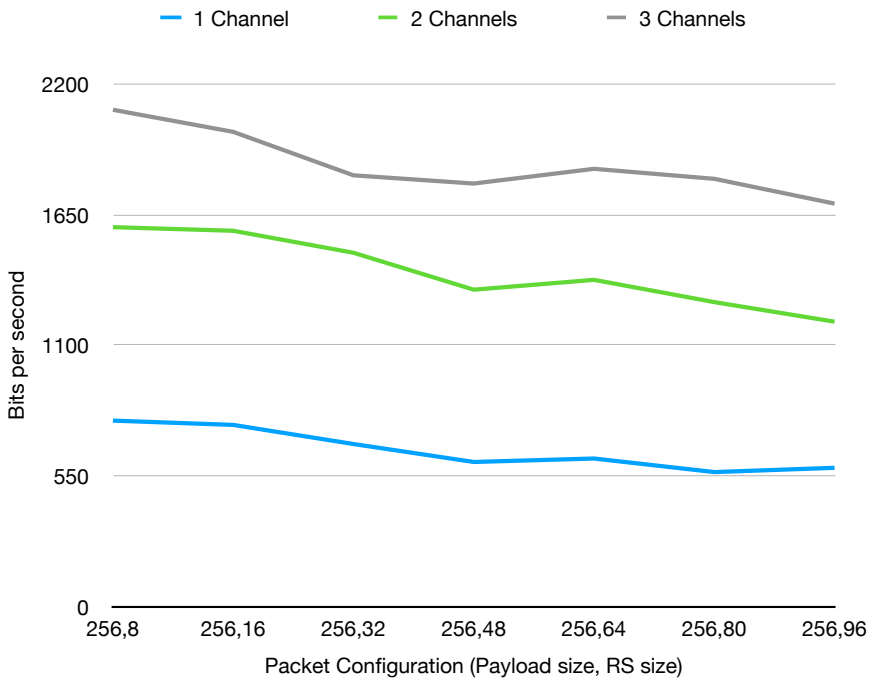


Figure 5.7: **Throughput while sending files at varying channel counts and packet configurations**

### 5.3.1 Thin Application Layer

To facilitate a mechanism for re-transmitting lost or corrupted packets, the transmitter side needs to be aware of exactly which packets have arrived successfully. Associating an index number with each packet is therefore necessary for transmitting a complete file where losing or corrupting any byte can irredeemably corrupt the entire file. On the other hand, there can also be applications where packet loss is tolerable, or re-transmitting old packets does not yield useful information. Therefore the packet index is an optional element that applications can use depending on their specific requirements and it is not hardcoded as a packet section as part of CMPPM. Instead, the packet index is encoded inside the payload alongside the data. This greatly simplifies the implementation in two ways:

- As described in Section 3.3.4, the modulated data in the packet is protected with a Forward Error Correction scheme. As the index is also a crucial part of the information contained in the packet, it should be error corrected as well. Receiving a packet with an intact payload but a corrupted index can have breaking consequences. Therefore it is a pragmatic solution to error-correct the index with the rest of the payload by encoding them together.

- There are implementation benefits to minimizing the variation between the structures of any two packets, for the sake of simplicity. If the subsequent packets could differ in length from each other, this would increase

the complexity of the system exponentially with the number of different configurations. Therefore it is beneficial to take a principled approach to limit the different possible configurations of packets in a scheme.

Sending and receiving a file intact in an environment with potential packet loss or corruption requires being able to re-send packets and store the successfully received packets until the entire file is received. Therefore two separate but related libraries are developed to store the packets in the memory with order as long as they are needed.

### Transmitter Packet Repository

The packet repository for the transmitter is responsible for storing each packet in memory until the transmitter receives confirmation that the packet is successfully received. This can be implemented with an interface for storing packets until confirmation of their successful sending, through their index value.

The packet repository for the transmitter also has the more complicated responsibility of providing an iterator over all the packets which were not transmitted successfully yet. An iterator provides a practical interface for the main program using the packet repository since it only needs to ask the packet repository for new packets without knowing about the internal state. When the packet repository does not return a new packet, the program knows the entire file was successfully transmitted. Such an interface follows the semantics of Rust iterators, where if an iterator returns an empty result once, this indicates the iterator is finished and should not provide any more results. Successfully transmitted packets can be removed from the repository to reduce memory usage. Therefore the packet repository can be used in a simple loop producing packets.

Determining which packet to transmit next depends on the purpose of the application transmitting the data. As part of this work, two modes are implemented which can cover a broad range of applications:

- If a file is not meaningful to the receiver until it arrives in its entirety, the simplest ordering of packets suffices. The packets can be sent from the lowest index to the highest index while keeping the last sent index in memory. The last sent index is useful for searching the packet repository for the next packet with the lowest index that was not yet sent. In this scheme, if a packet was not successfully transmitted, it is re-sent only after all the proceeding packets are transmitted once as well. As there is a delay between sending a packet and receiving confirmation of a successful transfer,

- For certain file formats, it can be desirable for the receiver to partially decode or utilize the file information incrementally while the transmission is ongoing. For example, a browser might want to decode a partially downloaded *jpeg*, or stream a *mp3*. In this case, the previously described re-transmission scheme would not lead to desirable results since there is little utility in transmitting proceeding packets while leaving a gap in the file due to a lost packet. Therefore the second transmission mode is introduced to transmit packets in chunks. Each chunk needs to be successfully transmitted in its entirety before continuing to the next chunk. This mode of operation introduces inefficiencies since the acknowledgment

of successful packet transmissions is prone to delays and meanwhile a packet might be re-sent multiple times needlessly.

#### Receiver Packet Repository

The receiver packet repository is responsible for storing the received packets in the correct order and providing them to the host program for decoding and writing to the target file. Since the receiver cannot assume the packets are sensed in the correct order, a logic layer is needed between receiving the packets in the possibly incorrect order and saving them to the file in the correct order. This repository has a relatively simpler structure compared to the transmitter packet repository and also implements an iterator returning the received packets in the correct order. In Rust terms, this is an un-fused iterator meaning that after returning an empty item, it can still return more items in the future. This is a common occurrence as lost packets create gaps, which are later filled with re-transmissions.

# Chapter 6

# Conclusions

In this work, a novel Visible Light Communications modulation scheme called Colored Multiple Pulse Position Modulation (CMPPM) was introduced. CMPPM expands upon existing PPM modulation schemes which only utilize the on and off states of an LED, by introducing the possibility to modulate data also in the color space and therefore achieve a throughput comparable to Color Shift Keying while retaining advantages inherent in pulse-based VLC systems. Similar to CSK, exploiting the color space more deeply leads to almost linear increases in throughput with CMPPM, but the system has to face up to the limitations caused by the increased signal error rate deteriorating the real throughput.

To test the feasibility of the modulation scheme, a hardware and software implementation of transmitter and receiver nodes was created and it was successfully demonstrated that the system can indeed accomplish complicated tasks such as sending a file with error correction, and can be combined with higher level layers such as re-transmission of faulty packets.

A theoretical and practical evaluation of the designed system was presented. On commercial hardware, up to 2 kb/s file transmission speed was achieved using the CMPPM scheme, and tolerable error rates were demonstrated up to 1.5 m distance between the transmitter and the receiver. Theoretically, it was demonstrated that the modulation scheme has the potential to achieve throughput on par with equivalent CSK systems and achieve throughput multiple times faster than equivalent MPPM communication schemes.

## 6.1 Future Work

Certain areas were not the main points of focus for this thesis work, even though they could or would be welcome additions to a more fully-fledged application or research on top of this work. The reasons for not developing in these areas were mainly constraints of time and expertise. Especially when it comes to hardware, lacking a background in Electrical Engineering, I have chosen a more convenient path of using simpler circuits instead of developing custom solutions.

### 6.1.1 More Performant Hardware

Limitations concerning transmission speed as well as noise have prevented the evaluation results from reaching more impressive practical data throughput rates. The limitation in the transmission pipeline of this project mainly stemmed from the photodiode and Op-Amp circuit capturing the modulated light. Amplifying the minuscule amounts of current produced by a photodiode when light is incident upon it and transforming this to voltage detectable by an Analog-to-Digital Converter is a process with substantial potential for interference from electrical noise. It also requires complicated circuitry to achieve the rise and fall times required for faster transmission speeds. Therefore a future iteration of this project can focus on developing custom circuitry to improve the signal-to-noise ratio while improving the rise and fall times of the photodiode sensing process.

### 6.1.2 Bidirectional Communications

Research projects in Visible Light Communications often focus on unidirectional communication applications, which limits the level of practical adoption such an application can achieve. Most common methods of data communications at least expect acknowledgments of successful arrivals from the receiving end of the communications link. Developing hardware capabilities and the theoretical groundwork for bidirectional communications between systems both equipped with LEDs and photodiodes would allow the implementation of applications with significantly more relevant use cases.

### 6.1.3 Quality of Service Adaptations

In wireless data transfer communications, rapid changes in the quality of service due to unpredictable conditions of the physical environment is a common problem. This is typically overcome in protocols such as IEEE 802.11e by defining a set of modulation schemes along a trade-off gradient of robustness and speed. The transmitter node is then responsible for switching to the most optimal scheme to provide the highest throughput when the environment changes and affects the signal-to-noise ratio. In CMPPM, the number of channels used for communications in the color domain can fulfill a similar role. A higher number of channels imply higher throughput but also higher susceptibility to noise. Detecting the changes in the environmental conditions and allowing reactive changes in the channel count would allow the definition of QoS levels for CMPPM.

### 6.1.4 Change Color and Dimming on the Fly

CMPPM scheme has no low-level support for changing the color and the dimming levels during the communications, as the receiver and the transmitter need to agree on the number of channels in the transfer. However, with the increasing importance of smart lighting systems, such capabilities are becoming crucial for the adoption of Visible Light Communications [32]. Therefore this capability can be implemented at a higher layer, for example with special packets that instruct the receiver device to expect a different number of channels to accomplish the new dimming level or the color, or at a lower level with extra hardware

such as differently calibrated LEDs or a PWM peripheral to change values on demand.

### 6.1.5 Theoretical SER Calculations

As part of this work, the symbol error rate was not calculated using formal methods such as Monte Carlo simulations commonly used in similar research papers [7, 28]. Such a calculation would be an important addition as it would allow for a systemic categorization of the limits of adding extra channels to increase the data rates of the system.

# Bibliography


[1] IEEE Standard for Local and Metropolitan Area Networks–Part 15.7: Short-Range Wireless Optical Communication Using Visible Light. Technical report, IEEE.

[2] Dil Nashin Anwar and Anand Srivastava. Constellation Design for Single Photodetector Based CSK With Probabilistic Shaping and White Color Balance. *IEEE Access*, 8:159609–159621, 2020.

[3] Sri Ariyanti and Muhammad Suryanegara. Visible Light Communication (VLC) for 6G Technology: The Potency and Research Challenges. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 490–493, London, United Kingdom, July 2020. IEEE.

[4] Muhammad Salman Bashir and Sajid Sheikh Muhammad. Time Synchronization in Photon-Limited Deep Space Optical Communications. *IEEE Transactions on Aerospace and Electronic Systems*, 56(1):30–40, February 2020.

[5] Bo Bai, Zhengyuan Xu, and Yangyu Fan. Joint LED dimming and high capacity visible light communication by overlapping PPM. In *The 19th Annual Wireless and Optical Communications Conference (WOCC 2010)*, pages 1–5, Shanghai, May 2010. IEEE.

[6] Wenjun Chen, Zhengpeng Li, and Ming Jiang. Color-and-Intensity Shift Keying for Visible Light Communication. *IEEE Communications Letters*, 22(9):1790–1793, September 2018.

[7] Yu Chen, Zhongyu Li, Peng Liu, and Song Liu. The BER performance comparison of MSK and GMSK schemes for short-range visible light communication. In *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, pages 611–614, Guangzhou, May 2017. IEEE.

[8] Ata Chizari, Mohammad Vahid Jamali, Sajjad Abdollahramezani, Jawad A. Salehi, and Akbar Dargahi. Visible light for communication, indoor positioning, and dimmable illumination: A system design based on overlapping pulse position modulation. *Optik*, 151:110–122, December 2017.

[9] Kyungmook Choi, Yunseon Jang, Jingyoung Noh, MinChul Ju, and Youngil Park. Visible light communications with color and dimming control by

employing VPPM coding. In *2012 Fourth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 10–12, Puket, Thailand, July 2012. IEEE.

[10] Faisal A Dahri, Sajjad Ali, and Muhammad Moazzam Jawaid. A Review of Modulation Schemes for Visible Light Communication. 2018.

[11] F. Davidson. Direct-Detection Optical Communication with Color Coded Pulse Position Modulation Signaling. *IEEE Transactions on Communications*, 33(3):273–276, 1985.

[12] Francisco A. Delgado Rajo, Victor Guerra, Jose A. Rabadan Borges, Julio Rufo Torres, and Rafael Perez-Jimenez. Color Shift Keying Communication System With a Modified PPM Synchronization Scheme. *IEEE Photonics Technology Letters*, 26(18):1851–1854, September 2014.

[13] John E. Gancarz, Hany Elgala, and Thomas D.C. Little. Overlapping PPM for band-limited visible light communication and dimming. *Journal of Solid State Lighting*, 2(1):3, December 2015.

[14] Alex Cartagena Gordillo. CNN- Based Demodulation of Color Shift Keying in Screen Camera Communications. In *2022 First International Conference on Computer Communications and Intelligent Systems (I3CIS)*, pages 117–122, Jijel, Algeria, November 2022. IEEE.

[15] Safa Halawi, Elias Yaacoub, Saadallah Kassir, and Zaher Dawy. Performance Analysis of Circular Color Shift Keying in VLC Systems With Camera-Based Receivers. *IEEE Transactions on Communications*, 67(6):4252–4266, June 2019.

[16] Pengfei Hu, Parth H. Pathak, Xiaotao Feng, Hao Fu, and Prasant Mohapatra. ColorBars: Increasing data rate of LED-to-camera communication using color shift keying. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, pages 1–13, Heidelberg Germany, December 2015. ACM.

[17] Sanjeev Kumar and Ragini Gupta. Bit Error Rate Analysis of ReedSolomon Code for Efficient Communication System. *International Journal of Computer Applications*, 30(12):11–15, September 2011.

[18] Kwonhyung Lee and Hyuncheol Park. Modulations for Visible Light Communications With Dimming Control. *IEEE Photonics Technology Letters*, 23(16):1136–1138, August 2011.

[19] J.M. Luna-Rivera, R. Perez-Jimenez, V. Guerra-Yañez, C. Suarez-Rodriguez, and F.A. Delgado-Rajo. Combined CSK and pulse position modulation scheme for indoor visible light communications. *Electronics Letters*, 50(10):762–764, May 2014.

[20] Jose Martin Luna-Rivera, Cristo Suarez-Rodriguez, Victor Guerra, Rafael Perez-Jimenez, Jose Rabadan-Borges, and Julio Rufo-Torres. Low-complexity colour-shift keying-based visible light communications system. *IET Optoelectronics*, 9(5):191–198, October 2015.

[21] Roger Alexander Martínez-Ciro, Francisco Eugenio López-Giraldo, Andrés Felipe Betancur-Perez, and Jose Martín Luna-Rivera. Design and Implementation of a Multi-Colour Visible Light Communication System Based on a Light-to-Frequency Receiver. *Photonics*, 6(2):42, April 2019.

[22] Parth H. Pathak, Xiaotao Feng, Pengfei Hu, and Prasant Mohapatra. Visible Light Communication, Networking, and Sensing: A Survey, Potential and Challenges. *IEEE Communications Surveys & Tutorials*, 17(4):2047–2077, 2015.

[23] Stefano Pergoloni, Mauro Biagi, Stefano Rinauro, Stefania Colonnese, Roberto Cusani, and Gaetano Scarano. Merging Color Shift Keying and Complementary Pulse Position Modulation for Visible Light Illumination and Communication. *Journal of Lightwave Technology*, 33(1):192–200, January 2015.

[24] Angga Pradana, Nur Ahmadi, Trio Adiono, Willy Anugrah Cahyadi, and Yeon-Ho Chung. VLC physical layer design based on Pulse Position Modulation (PPM) for stable illumination. In *2015 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pages 368–373, Nusa Dua Bali, Indonesia, November 2015. IEEE.

[25] Sridhar Rajagopal, Richard Roberts, and Sang-Kyu Lim. IEEE 802.15.7 visible light communication: Modulation schemes and dimming support. *IEEE Communications Magazine*, 50(3):72–82, March 2012.

[26] Saeed Rehman, Shakir Ullah, Peter Chong, Sira Yongchareon, and Dan Komosny. Visible Light Communication: A System Perspective—Overview and Challenges. *Sensors*, 19(5):1153, March 2019.

[27] Ravinder Singh, Timothy OFarrell, and John P. R. David. An Enhanced Color Shift Keying Modulation Scheme for High-Speed Wireless Visible Light Communications. *Journal of Lightwave Technology*, 32(14):2582–2592, July 2014.

[28] Jian Song, Tian Cao, and Hongming Zhang. A low complexity NOMA scheme in VLC systems using pulse modulations (invited paper). In *2020 29th Wireless and Optical Communications Conference (WOCC)*, pages 1–6, Newark, NJ, USA, May 2020. IEEE.

[29] H. Sugiyama and K. Nosu. MPPM: A method for improving the band-utilization efficiency in optical PPM. *Journal of Lightwave Technology*, 7(3):465–472, March 1989.

[30] Kubilay Muhammed Sünnetci and Mehmet Sönmez. Variable pulse position modulation receivers for visible light communication systems without the knowledge of dimming level. *Transactions on Emerging Telecommunications Technologies*, 33(5), May 2022.

[31] Qing Wang, Domenico Giustiniano, and Daniele Puccinelli. OpenVLC: Software-defined visible light embedded networks. In *Proceedings of the 1st ACM MobiCom Workshop on Visible Light Communication Systems*, pages 15–20, Maui Hawaii USA, September 2014. ACM.

[32] Hongjia Wu, Qing Wang, Jie Xiong, and Marco Zuniga. SmartVLC: Co-Designing Smart Lighting and Communication for Visible Light Networks. *IEEE Transactions on Mobile Computing*, 19(8):1956–1970, August 2020.