UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

Software-defined implementation and practical evaluation of ARQ schemes over Visible Light Communications

Author:
Martí Busquets González

Supervisors:
Dr. Joan Bas
Dr. Alexis Dowhuszko

Professor:
Dr. Ana Isabel Pérez

A thesis submitted in fulfillment of the requirements for the degree of Masters in Telecommunication Engineering

in the

Centre Tecnològic de Telecomunicacions de Catalunya

May 21, 2021

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Abstract

Masters in Telecommunication Engineering

Software-defined implementation and practical evaluation of ARQ schemes over Visible Light Communications

by Martí Busquets González

Nowadays, telecommunications systems have become a very powerful technology. The ability to be connected to the Internet anywhere on the planet, by simply unlocking a mobile phone, has revolutionized the in which we all live. 5G is arriving with a whole new set of applications (e.g., IoT, connected car, smart cities, augmented reality, etc.). In the near future, it will be necessary to implement a more comprehensive wireless system to satisfy the requirements of our society. Visible Light Communications (VLC) is a candidate technology for the next generation of wireless systems, particularly in those situations where a low-cost solution and the license-free spectrum require enabling ultra-dense deployments of small cells indoors. The major shortcoming of VLC is the large drop in power when an element blocks the line-of-sight between transmitter and receiver. To address this, different Automatic Repeat Request (ARQ) protocols have been implemented in a software-defined testbed to ensure a proper reception of the information. Also, the ARQ signalling was used to enable a VLC-based monitoring system, without the need to deploy additional monitor sensors.

Acknowledgements

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of development of this MSc Thesis.
I want to thank my supervisors Dr. Joan Bas and Dr. Alexis A. Dowhuszko, I am sincerely grateful to them for sharing their knowledge and helping me on a number of issues related to the project.
I would also like to thank the teachers who have believed in me throughout my years as a student, who have supported and encouraged me to continue with my studies.
Finally, I want to thank my family and friends who have supported me during all these years. I would not have made it this far if it were not for you.

Contents

List of Figures

List of Tables

Acronyms

ACK Acknowledgement.

AI Artificial Intelligence.

AM Amplitude Modulation.

ANN Artificial Neural Network.

ARQ Automatic Repeat Request.

ASK Amplitude-Shift Keying.

AWGN Additive White Gaussian Noise.

BER Bit Error Rate.

BLER Block Error Rate.

BPSK Binary Phase-Shift Keying.

CRC Cyclic Redundancy Check.

FM Frequency Modulation.

IoT Internet of Things.

LED Light-Emitting Diode.

LoS Line-of-Sight.

NACK Negative-acknowledgement.

PAM Pulse Amplitude Modulation.

PD Photodetector.

PDF Probability Density Function.

PSK Phase-Shift Keying.

QAM Quadrature Amplitude Modulation.

QPSK Quadrature Phase-Shift Keying.

RF Radio Frequency.

RTT Round Trip Time.

SNR Signal-to-Noise Ratio.

SRRC Square Root Raised Cosine.

USRP Universal Software Radio Peripheral.

UVLC Underwater Visual Light Communication.

VLC Visual Light Communication.

VLP Visible Light Positioning.

Wi-Fi Wireless Fidelity.

Chapter 1

Introduction

Visual Light Communication (VLC) is a technology that consists of transmitting information in the frequency range of that is visible for the human eye, between 400 and 800 THz (780 - 375 nm). VLC is oriented to high speed data communication, creating a variation on the instantaneous light intensity that it is undetectable for the human eye. Radio Frequency (RF) wireless communications systems such as Millimeter-Wave (5G) and Tera-Hz bands (6G), require advanced digital signal processing [1] to provide a high speed communications. On the contrary, VLC systems relies on low-cost, energy efficient Light-Emitting Diodes (LEDs) as transmitters and on Photodetectors (PD) at the receiver side.

The current technologies, 2G, 3G and 4G utilize the frequencies range to reach up to 6 GHz [2]. Nevertheless, 5G systems aims to exploit its spectral efficiency by adding carriers in the Millimeter Wave frequency band (i.e., between 24 and 100 GHz) in addition to the contemporary frequency bands. Nowadays, investigators are realizing the growing data rate demands will force to remodel the communications technology again, as an ultra-high definition video may reach 24 Gb/s and some 3D videos may arrive to 100 Gb/s [3]. To achieve this future demands, the next generation of wireless communication technology, 6G, is expected to support transmission rate 100-1000 times higher than 5G [2], velocities difficult to accomplish in the same frequency range as the state of the art technologies. With 400 THz of bandwidth, VLC is a very powerful technology to achieve these extremely high data rates [4].

Unlike RF systems, VLC transmissions can be blocked by the walls. However, it supports Line-of-Sight (LoS) and indirect one (diffuse and specular reflections). In an environment where there may be temporal blockages, it is necessary to implement an Automatic Repeat Request (ARQ) scheme to ensure the correct reception of the information or reallocate the transmission point [5].
Towards this regard, the aim of the MSc Thesis consist on designing a software-defined VLC link with an ARQ protocol to be able to perform a transmission through a visible light signal, with the reliance that a possible blockage do not put the transmission in outage. Instead, the implementation of an ARQ scheme delays the data stream to be transmitted in a buffer until the connectivity is restored.

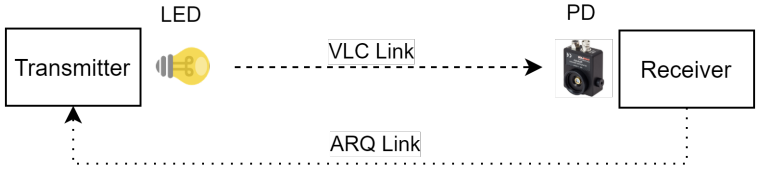


FIGURE 1.1: General overview of the VLC link. The dash line is the light data transmission and the dot line is the automatic request link for frame validation.

At the transmitter side, different modulation schemes have been studied, to send information through the LED and reach the PD. At the receiver side two ARQ schemes have been implemented, namely Stop-and-Wait and Go-Back-N to compare and evaluate the differences between them.

1.1 Motivation

VLC technology is becoming a very powerful candidate for the next generation of mobile communications standards (i.e., beyond 5G) [1]. The ultra-high bandwidths within the frequency range between 400 and 800 THz which, compared to RF bandwidth, will allow creating ultra-densification systems capable to reach high data rate transmissions.

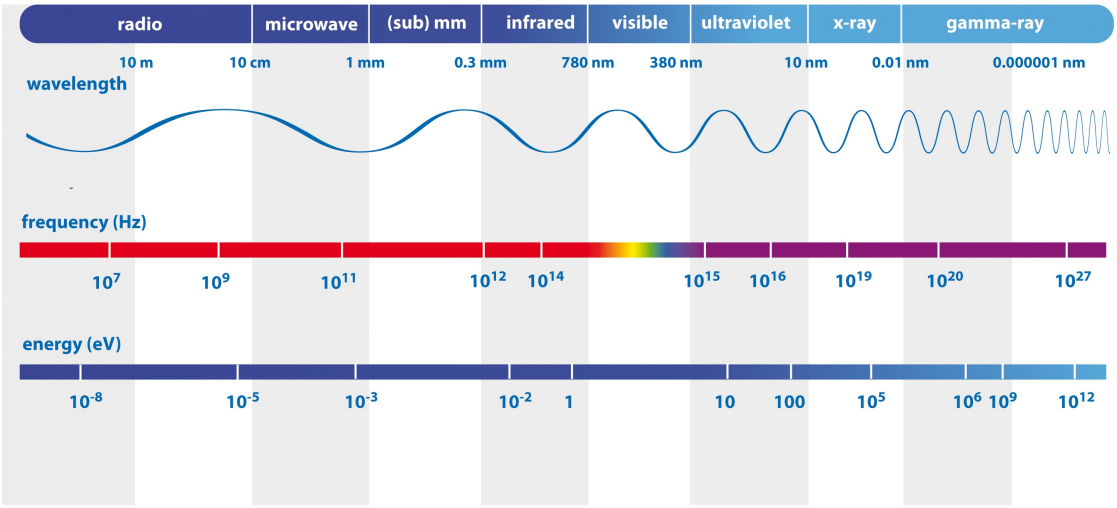


FIGURE 1.2: A scheme of the electromagnetic spectrum with indication of wavelengths, frequencies and energies. [6]

When the demand for data rate increases exponentially, VLC will support 6G to provide superior coverage integrating space, air, underwater and terrestrial networks. New applications will emerge, such as real-time Underwater Visual Light Communication (UVLC) [7], and the utilization of machine learning with the implementation of Artificial Neural Network (ANN) strategies either its to equalize the signal to increase the data rate or extract features of the crossing links to monitor the status of the VLC system. [8].

With the arrival of 5G, the Internet of Things (IoT) will be a reality, and it will be necessary massive connectivity with high reliability, high data rate, high positioning accuracy, low latency, low power consumption and improved security. To address these challenges, VLC is a promising candidate [9], not only for the low cost or the existing lighting infrastructure. Also, for the high speed transmission, the integration with Wireless Fidelity (Wi-Fi) systems [10], the communication security and the offering of high accuracy localization via Visible Light Positioning (VLP) [11].

Nowadays, there are many technologies that monitor the status of an environment. The surveillance camera systems or infrared sensors are the most used, but there are also, RF systems taking advantage of Wi-Fi or 4G nodes to sense and interpret the variations of the electromagnetic waves during propagating. VLC with the LoS restriction, can also be a useful tool to monitoring a room or a hallway. Depending on

the duration of blockage it can be know if there are one or two persons crossing the beam, or if the person going through the beam of light is walking or running, and even the direction it has been followed [12].

1.2 Objectives

The main objective is to implement a VLC link with an ARQ protocol capable to detect and retransmit every data frame whose content becomes corrupted during the transmission. To reach this point, a solid theoretical background, as well as the use of software tools for numerical simulations in both real and non-real time are also required. The specific objectives that stem from the aim of this MSc Thesis can be outlined as follows:

- *Theoretical fundamental*: Understand the theoretical background for VLC at the Physical Layer, with emphasis on the different modulation schemes that can be used to modulate the intensity of the light emitted by an LED transmitter. Moreover, understand the different ARQ schemes and the advantages or disadvantages between them.
- *Numerical simulations (non-real-time)*: Estimate the performance that is achievable with different schemes using non-real time simulation tools (e.g., Matlab). Compare the obtained simulations results with the ones provided by the theoretical background, to have a starting point for the following real time simulations.
- *Software defined VLC (real-time)*: Using GNU Radio and python, create and simulate different VLC transmission schemes comparing the obtained figure of merits with the numerical results to ensure a correct behaviour.
- *Universal Software Radio Peripherals (USRPs) and LED/PD*: Incorporate to the different simulations in *GNURadio* with the USRP and the LED/PD in the loop.
- *Practical validation*: Measurement campaign setup, including hardware requirements such as USRPs, LED, PD and driver. Make an adaptation of measurements to figures of merits that are relevant to the VLC transmission link under study.

1.3 Methodology

In order to reach the objectives, this MSc Thesis has followed the methodologies that are listed below for each chapter:

- In Chapter 2, a theoretical background has been considered. Different modulation schemes have been studied. Moreover, an important point is the creation of a ARQ, for this reason a thorough study on different schemes such as Stop-and-wait and Go-Back-N has been conducted.
- In Chapter 3, different scenarios have been developed in a non-real-time simulation tool. Simulated transmissions with different modulations comparing different parameters such as the Bit Error Rate (BER) or the Block Error Rate (BLER) with their theoretical expressions.
- In Chapter 4, the software defined platform GNU Radio has been used in conjunction with *python* programming language to create different blocks to translate the non-real-time simulations to a real-time scenario. In addition to the BER and BLER, other relevant performance parameters have been measured to determine the efficiency of the ARQ link.

- In Chapter 5, GNU Radio has been combined with the required hardware (i.e., USRPs, LED, PD and LED driver), to create a actual VLC transmission. New functionalities have been implemented, such as automatic gain control to compensate the distance-dependent channel attenuation and stream synchronization to tackle the variable delay that the data symbols experience during the propagation on the VLC channel.
- Finally, Chapter 6 presents the experimental results. The different measures collected in the previous chapters are summarized and compared to the theoretical ones. Future improvements are also discussed.

Chapter 2

Theoretical background

This chapter summarizes the key theoretical concepts that were utilized to develop the communication of digital information over a VLC link. Digital communications deals with the transmission and reception of a stream of bits over a channel. Although there are different types of channels, such as copper wires or optical fibres, the focus of this MSc Thesis is on the use of a wireless communication channel.
Wireless channels have had many applications along history. Modern mobile communications stated in the 1980s with analog communications (1G), and then it evolved to digital communications on the following mobile generations, from 2G to the future 5G and 6G [13].

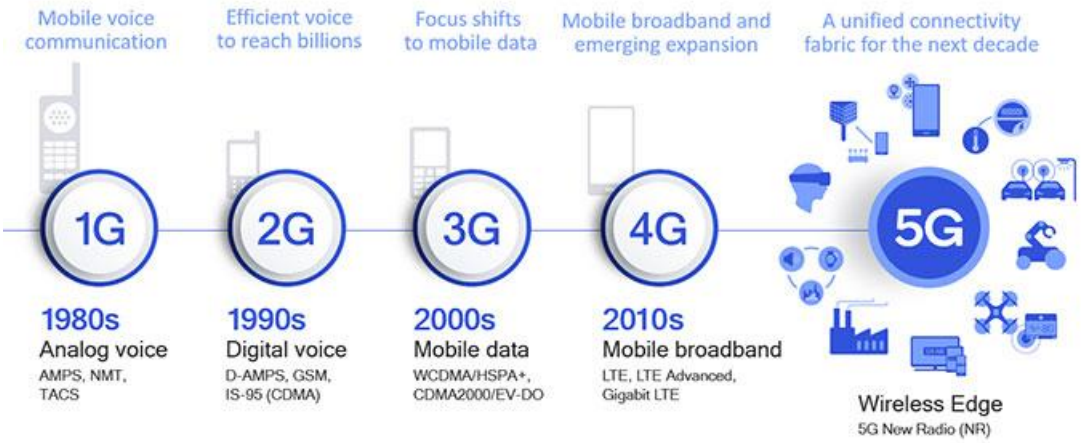


FIGURE 2.1: Every decade, the evolution of mobile wireless communications changes with the arrival of a new generation. The current trend is on ultra-densification and sustainability, where VLC can have a notable role

2.1 Modulation techniques

In order to transmit information, there are different options to vary the properties of a periodic waveform known as wireless carrier signal. This process is known as modulation, and it can take place either in the analog or digital domains. Analog modulation schemes are impressed on the carrier signal and need less processing resources. For example, in Amplitude Modulation (AM) the amplitude of the carrier is modified, whereas in Frequency Modulation (FM) the frequency of the carrier signal is varied according to the information to be transmitted.
Modern systems use digital modulations, where the input information is mapped and converted into a sequence of digital symbols which are used to modulate some properties of the carrier. This methodology increases the resources that are needed to perform the digital signal processing in transmission and reception, but the results are

immensely better. Two examples of digital modulation schemes are Phase-Shift Keying (PSK), which conveys data by the variation of the phase in a constant frequency carrier, and Amplitude-Shift Keying (ASK), which uses the amplitude of the carrier as a parameter to be modified according to the digital data that wants to be transmitted. Each modulation scheme has a different performance, measured by the BER that determines the number of bit errors divided by the total number of transferred bits. This parameter depends on the Signal-to-Noise Ratio (SNR) that experiences the communication channel. Depending on many factors (i.e., noise, interference, distortion, bit synchronization problems, multipath fading, etc.) the quality of the communication channel may be affected and the SNR may be reduced. When this happens, bit errors are more frequent and the BER performance of the digital communication channel is increased. On the other hand, when there is no interference and the strength of the noise that is present in the digital channel is much weaker than the transmitted data-carrying signal, the SNR increases and the bit errors are less likely. In other words, the BER decrease and the transmission of information is more reliable.

2.1.1 Quadrature Amplitude Modulation (QAM)

Nowadays QAM is widely used in digital telecommunication systems. The Quadrature Amplitude Modulation (QAM) scheme consists of two carrier waves at the same frequency with a different phase shift. This condition is known as orthogonality or quadrature, and consists of setting the phase shift between both carriers at precisely 90^o. The In-Phase and Quadrature components of the data-carrying signals are then used to modulate the amplitude of these two waves. In reception, the signal can be decoded thanks to the orthogonality property.

Binary Phase-Shift Keying (BPSK) is the simplest QAM scheme. It uses only two symbols with a 180^o separation between them. Since the two symbols of BPSK are considerably separated, the decision threshold is at the farthest distance of all possible QAM schemes. This makes BPSK one of the most robust modulation schemes, and even with a low SNR it is possible to reach an acceptable BER as shown in

$$\text{BER} = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right), \tag{2.1}$$

where E_b/N_0 is the SNR and $erfc(x)$ is the Gauss complementary error function. Nevertheless, with only 1 symbol per bit is unsuitable for high data-rate applications.

4-QAM, also known as Quadrature Phase-Shift Keying (QPSK), is the following modulation order that could be implemented in QAM schemes. QPSK uses two orthogonal carriers, both modulated by amplitude. For this reason, at reception both carriers can be demodulated independently, and the decision thresholds are the same as BPSK. Although the two modulations follow the same BER probability of (2.1), the accommodation of two bits per symbol makes the bit rate of QPSK two times faster that BPSK.

There are many others QAM methods, such as, 16-QAM, 64-QAM and 256-QAM. However, the higher order modulation scheme that has been considered in this MSc Thesis is the 16-QAM, which transports two bits In-Phase and Quadrature carriers, totalizing 4 bits per symbol. This increment has two consequences: On one hand, it makes the decision thresholds more closely packed, incrementing the probability of having a symbol wrongly detected as started in

$$\text{BER} \approx \frac{3}{8}\text{erfc}\left(\sqrt{\frac{6E_b}{15N_0}}\right). \tag{2.2}$$

On the other hand, the bits of information can be accommodated in fewer symbols, and the data rate increase.

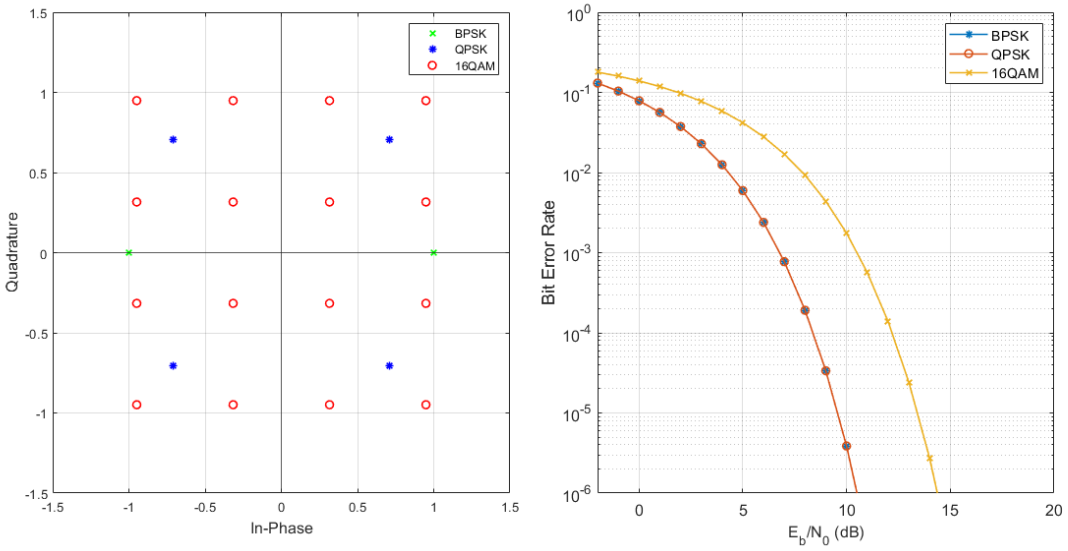


FIGURE 2.2: From the left to the right. The constellations and the BER curves for the BPSK, QPSK and 16-QAM. The symbols are normalized to transmit unitary power.

It is important to highlight that these modulation schemes have been studied to understand their properties, advantages and disadvantages. The QAM schemes are not suitable for VLC transmission due to a LED transmitter is not able to deal with quadrature subcarriers (i.e., coherent communication scheme), as the only parameter that it can be varied is the intensity of the light that is emitted.

2.1.2 Pulse Amplitude Modulation (PAM)

PAM is a signal waveform in which the bit stream information is encoded on the amplitude of a series of signal pulses. At the receiver side, an equivalent demodulator decides which amplitude corresponds to each received sequence of samples.

2-PAM, is the baseline. It consists of two symbols modulated in $+1$ and -1. Bit "1" is mapped as symbol $+1$ and bit "0" is mapped as symbol -1. In terms of data rate and BER probability, this modulation is approximating to BPSK (2.1).

4-PAM is a higher-order modulation in which the four possible symbols (that can accommodate 2 bits) are modulated in $+3/\sqrt{5}$, $+1/\sqrt{5}$, $-1/\sqrt{5}$ and $-3/\sqrt{5}$. The bits are mapped in pairs. Bit "00" is map as symbol $+3$, bit "01" is map as symbol $+1$, bit "11" is map as symbol -1, and bit "10" is map as symbol -3. Notice that this mapping is not chosen randomly. It corresponds to Gray mapping, where two successive symbols only differs in one bit. This may reduce the BER formula in

$$\text{BER} \approx \frac{3}{8}\text{erfc}\left(\sqrt{\frac{E_b}{5N_0}}\right). \tag{2.3}$$

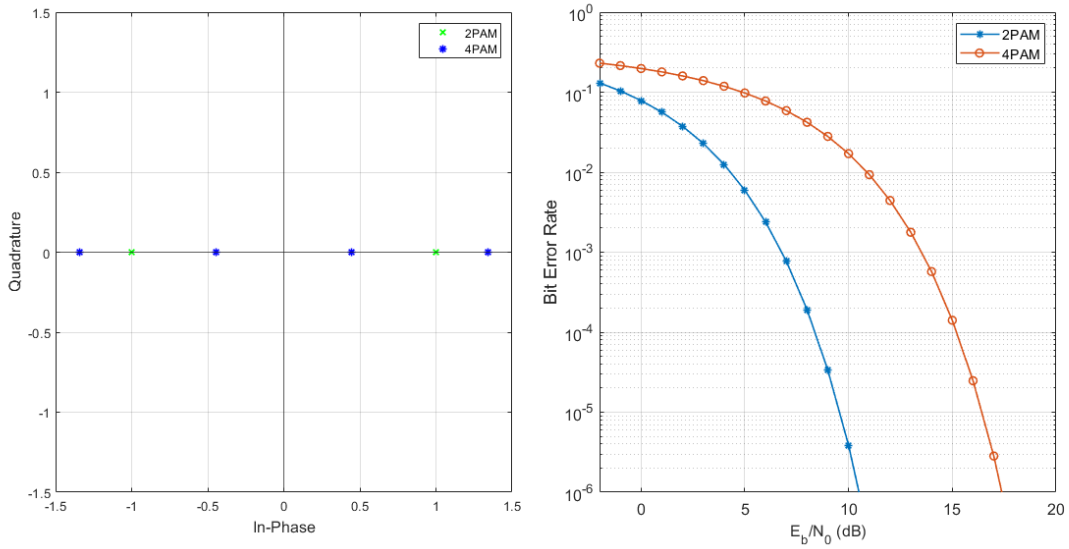


FIGURE 2.3: From the left to the right. The constellations and the BER curves for the studied PAM modulations. It is possible to observe a gap of 7 dB between the 2 and the 4-PAM.

As these modulations interact only with the signal amplitude, it is perfectly suitable to implement a VLC link in baseband. Using LEDs, the intensity is directed related to the signal strength. At this point, it is important the time-varying intensity modulation that is introduced does not create undesired effects to a human's vision, such as blinking.

2.2 Automatic Repeat Request schemes

As it is well known, a wireless link transmission is far away from being perfect. Noise, interferences and other factors make data transmission process variable in a stochastic way. The BER is based on that, if its value is equal to zero, it means a perfect channel communication, with no errors. Instead, if the measured value is 0.5, it means the channel is strongly affected by the noise, with no chances to infer the transmitted bits from the received signal samples correctly.
ARQ is an error control method that ensures the proper reception of the transmitted data. At the transmitter side, all the information to be transmitted is divided in frames of constant length that are sent it through the digital communication channel. At the receiver side, every frame is processed to detect if all the bits arrived correctly and, if so, notify the transmitter about the status of the received data frame. To achieve a proper functionality many elements are introduced.

Cyclic Redundancy Check (CRC) is the code in charge of detecting errors at the receiver side. The simplest error detecting code uses only one parity bit. However, a CRC scheme relies on a bigger number of bits to add redundancy to the system and be able to ensure the detection of errors (n-CRC), been n the number of bits of the CRC codeword. The code is calculated with the division of the frame data bits (payload) and a fixed polynomial [14]. In this computation, the quotient is discarded, and the remainder becomes the useful data to be appended in the frame. This result is added to the end of the frame and transmitted with the payload. In reception, the system computes the same polynomial division with the received frame. Then, if the remainder of this calculation is 0, no error has been detected. However, when there are errors

in the frame, the result is different from 0 and the received frame is discarded.

Acknowledgements (ACKs) and Negative-acknowledgements (NACKs) are the messages sent by the receiver to notify whether the current data frame has been properly received or not. If a frame arrives and the result of the CRC calculation is correct, the system notify an ACK. Nevertheless, if the computation is incorrect, and it is impossible to recover the correct information, the receiver notifies the transmitter with an NACK. This allows the transmitter to know which information has been received properly and which part of the message needs to be retransmitted.

In a ARQ protocol, there is one more possibility to take into account. This situation takes place when the actual value of the channel can not be predicted reliable when a large interference event occurs. This could happen in a VLC system due to, e.g., an unexpected obstacle can blockage the light beam. These problems may generate a total frame lost, and the receiver never detect the presence of a frame. Therefore, since no frame transmission is detected, it never sends an ACK or NACK.
As the transmitter always expects a frame confirmation, a timer, known as timeout, needs to be implemented. The system sends the frame and waits an amount of time. If in this time the transmitter does not receive the correct ACK or NACK referring to this frame, it considers the frame as lost and triggers the retransmission of it. The correct period to wait for the frame defined as a reliable upper bound from the estimated Round Trip Time (RTT), where RTT is the average time that it takes when a frame is sent until its ACK is received.

At this point, it is possible to introduce another parameter to measure the performance of the transmission. At the transmitter side, all the data information is divided into frames to be sent. That is why, in addition to the BER, other value to be taken into account is the BLER. This ratio for uncoded signals is obtained by the probability of receiving all the bits in a frame

$$\mathrm{BLER} = 1 - (1 - \mathrm{BER})^{L}, \tag{2.4}$$

where L is the number of symbols that are accommodated in each frame.

2.2.1 Stop-and-Wait ARQ

Stop-and-Wait is the simplest ARQ mechanism. As the name states, it transmits a frame and waits until an ACK is received. During the waiting period, the system do not transmit any more frames. This is why Stop-and-Wait ARQ is inefficient compared to other ARQ schemes [15].

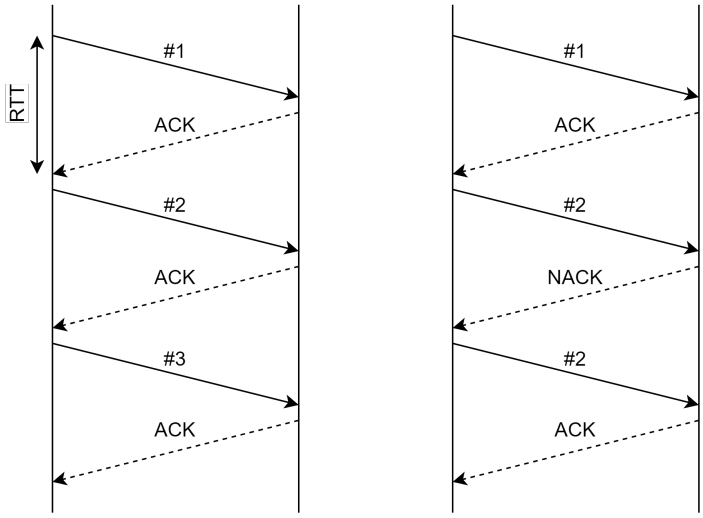


FIGURE 2.4: Examples of different events in a Stop-and-Wait ARQ protocol. Left image, the transmission is working properly. Right image, the information of the second frame was corrupted during the transmission.

As it is shown in the Fig. 2.4, the Stop-and-Wait protocol can handle different events. When the frames are correctly received, the transmitter receives an ACK and sends the following frame. When the frame is wrongly received, the transmitter is notified by a NACK and sends the same frame again until it receives the correct ACK. Finally, the possibility of failure in the reception of the frame is considered. The transmitter waits the equivalent of $2RTT$ (Fig. 2.4). Then, it sends the same frame as if a NACK signal has been received.
To measure the performance of this protocol, it is possible to monitor different parameters. For instance, the average number of transmissions for each frame as

$$M_{\text{tx}} = \frac{1}{1 - \text{BLER}}. \tag{2.5}$$

It consists of the number of times that the same frame needs to be transmitted to arrive correctly.
The utilization, or efficiency, of the channel (U), consists in the percentage of time that the ARQ protocol is transmitting useful information (data information, retransmissions are not considered) divided by the total time out of the communication. The channel utilization can be calculated for an ideal case, with a perfect channel without errors as

$$U_{\text{max}} = \frac{T_{\text{tx}}}{\text{RTT}} = \frac{T_{\text{tx}}}{2T_{\text{prop}} + T_{\text{tx}}}, \tag{2.6}$$

where the result it is considered the maximum efficiency a channel can reach.
For a non-ideal case the BLER is added to the equation as

$$U = \frac{T_{\text{tx}}(1 - \text{BLER})}{\text{RTT}} = \frac{T_{\text{tx}}(1 - \text{BLER})}{2T_{\text{prop}} + T_{\text{tx}}}, \tag{2.7}$$

where the channel affects the transmission and cause random errors.

2.2.2 Go-Back-N ARQ

Go-Back-N is a more complex ARQ mechanism. This protocol tries to reduce the waiting period by increasing the number of frames sent on a continuous basis. When the

system needs to transmit, it generates a sliding window containing a given number of frames (w) to be sent. When these frames reach the receiver, an ACK is generated for every one of them. In the Go-Back-N ARQ protocol, a frame sequence number is included in each frame and each feedback message, in order to ensure that the correct frame has been received. When the transmitter has been notified, the window is moved to transmit the following frames. That is the reason why this mechanism it is also known as sliding window protocol.

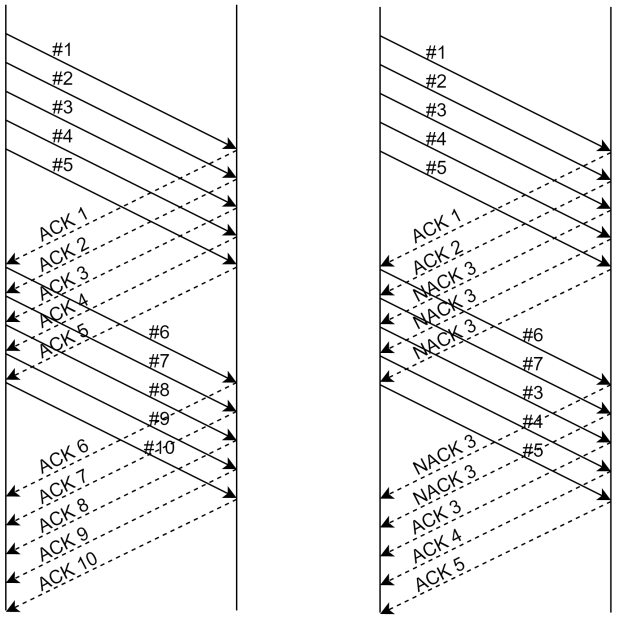


FIGURE 2.5: Examples of different events in a Go-Back-N protocol in a window equal to 5. Left image, the transmission is working properly. Right image, the third frame has arrived corrupted.

Two different scenarios have been considered in Fig. 2.5. On the Left-Hand side, the transmission works properly and there are no errors (NACKs). When the transmitter receives the correct signal, the sliding window moves to the following position to transmit the next frames. On the Right-Hand side of the figure, the third frame is received with errors and, by means of the CRC, this situation is detected. However, the fourth and fifth frames are received properly. As the receiver is waiting for the third frame, any other identification number is discarded. Therefore, the same NACKs asking for the correct frame is generated until it arrives properly at reception.

The utilization of the channel for Go-Back-N ARQ increase compared with Stop-and-Wait ARQ, thanks to the sliding windows [16]. In a theoretical case, for a perfect channel, according to

$$U_{\max} = \frac{wT_{\text{tx}}}{\text{RTT}} = \frac{wT_{\text{tx}}}{2T_{\text{prop}} + T_{\text{tx}}}, \qquad \text{if } wT_{\text{tx}} < \text{RTT}, \tag{2.8}$$

where efficiency increase with the window size. Reaching a point where the windows is optimum, $w_{\text{opt}} = RTT/T_{\text{tx}}$. When this point is reached, $wT_{\text{tx}} >$ RTT , the efficiency is maximum, $U = 1$, and the channel is the most used as possible.

In a non-ideal case scenario, the channel alterations generate errors in reception. As the BLER increases, the frames need more transmissions to be received properly. This increases the total time of the communication for the same amount of information

and consequently, the utilization of the channel is reduced. Therefore, in this scenario the efficiency of the channel can be written as

$$U = \frac{wT_{\text{tx}}(1 - \text{BLER})}{\text{RTT}} = \frac{wT_{\text{tx}}(1 - \text{BLER})}{2T_{\text{prop}} + T_{\text{tx}}}, \qquad \text{if } wT_{\text{tx}} < \text{RTT} \tag{2.9}$$

until an optimum windows if reached. However, after this point the channel still depends on the channel, $U = 1 - \text{BLER}$, when $wT_{\text{tx}} > \text{RTT}$. The purpose of this protocol is to find the optimum w_{opt} following

$$w_{\text{opt}} = \frac{\text{RTT}}{T_{\text{tx}}} = \frac{2T_{\text{prop}} + T_{\text{tx}}}{T_{\text{tx}}}. \tag{2.10}$$

Notice that with a lower w_{opt} the system may have a waiting period such that the channel efficiency is not optimal. On the other hand, with a higher w_{opt}, the channel may be fully used but the required processing resources to transmit continuously are higher than necessary.

Chapter 3

Numerical simulations

This chapter presents the numerical simulations in a non-real-time software (i.e., Matlab) to obtain the first figures of merits and compare them with their theoretical ones. The simulations have been focused on Pulse Amplitude Modulation (PAM), because, as explained in Chapter 2, the QAM schemes are not suitable for the implementation of an intensity modulation VLC link.

Different simulations have been carried out. For the two modulations, 2-PAM and 4-PAM, simulations at the bit level to calculate the BER, and at the frame level to calculate the BLER, have been done. To continue, ARQ simulations have been designed to understand the evolution of the efficiency of the channel in the Stop-and-Wait and Go-Back-N ARQ protocols.

In all the simulations, artificial Gaussian noise has been added to the transmitted signal to emulate a Additive White Gaussian Noise (AWGN) channel. The Gaussian noise samples have an Probability Density Function (PDF) equal to the normal distribution

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 \big/ 2\sigma^2}, \tag{3.1}$$

where μ is the mean, and σ is the standard deviation.

3.1 BER simulations

The bit level simulation, consists of emulating a bit stream mapped as symbols, passing them through an AWGN channel model. This channel adds a Gaussian noise into the symbols array. As a statistic, if the vector of bits to be simulated is longer, more accurate is the obtained simulated result. When this noise is added, the receiver side is simulated comparing the noise contaminated symbols with the initial clean ones.

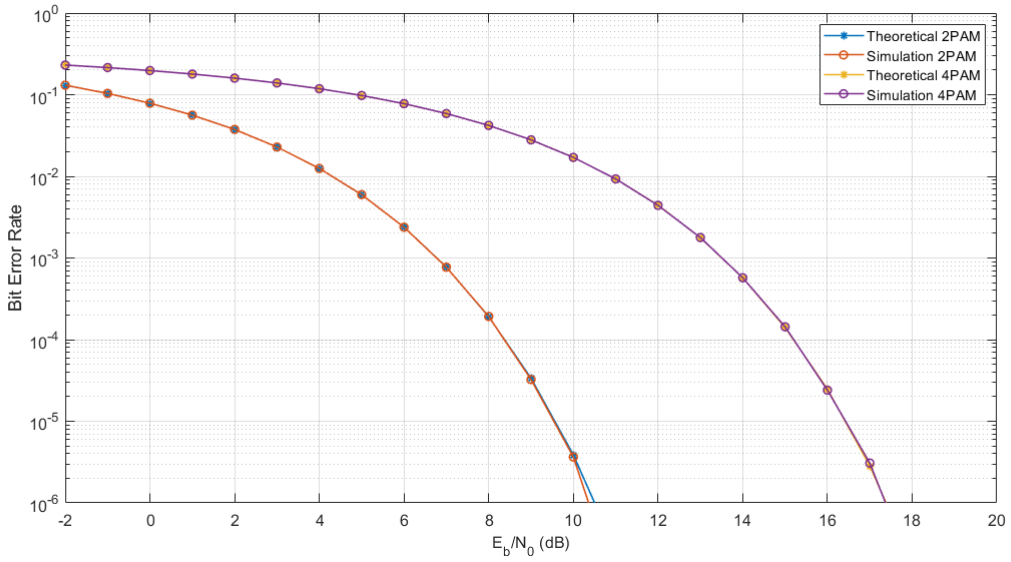


FIGURE 3.1: 2-PAM and 4-PAM theoretical and simulated BER curves in function of the SNR of a channel

For the 2-PAM, the symbols generated to execute the simulations are $+1$ and -1. As there have a unitary amplitude, there is no inconvenience to transmit them without any alteration. However, in the case of the 4-PAM, the constellation symbols are $+3$, $+1$, -1 and -3. These symbols do not have unitary energy and need to be normalized, according to the mean symbol energy. Therefore, in the case of the 4-PAM, to achieve a transmission with unitary energy, the normalized symbols are $(+3, +1, -1, +3)/\sqrt{5}$. Once the signal arrives at reception, the opposite operation is realized to recuperate the original symbols.

3.2 BLER simulations

The simulation at frame level consist of transmitting the information with frames of constant length, L. First, the designed program takes $L - 32$ data bits. Following, the 32-CRC is calculated and added after the payload. As the CRC codeword has a length of 32 bits, the total frame length is L.

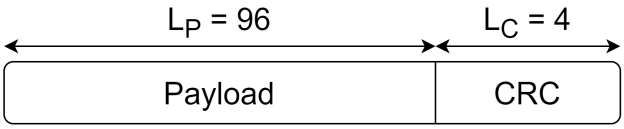


FIGURE 3.2: Overview of the frame structure implemented in the BLER and Stop-and-Wait ARQ simulations. The length of the frame fields is in bytes.

Once the frame is constructed, the bits are mapped into symbols according to the modulation scheme that is used. At this point, AWGN samples are added to the received signal. To continue, the signal is demapped with the proper demodulator and converted to bits.
Finally, the bits of the frame are verified by the CRC check. If the remainder of the calculation is equal to 0, the frame has been received properly. Otherwise, the frame has been corrupted by the Gaussian noise, and it is counted as an error.

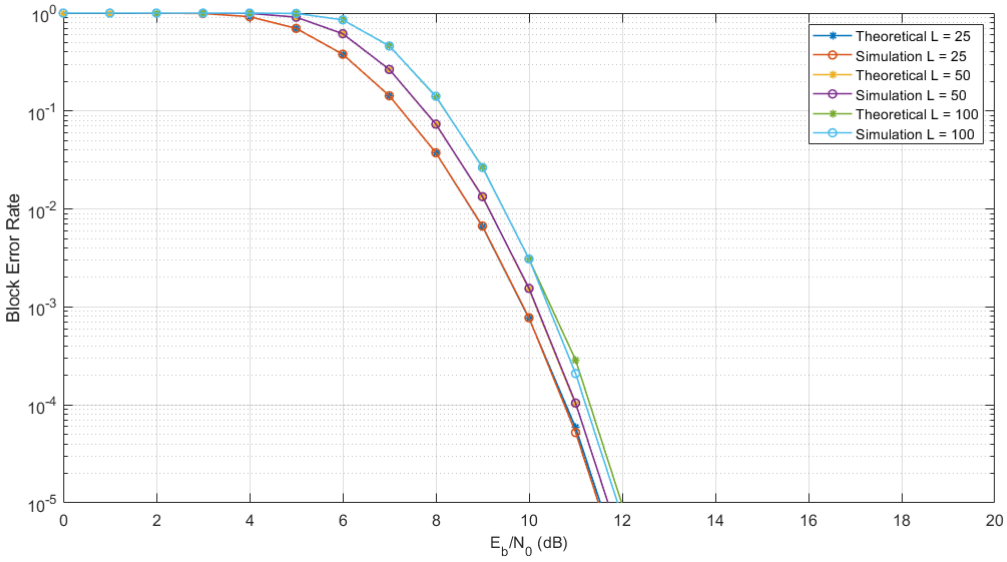


FIGURE 3.3: 2-PAM theoretical and simulated BLER curves in function of the SNR of the channel for different frame sizes, in bytes.

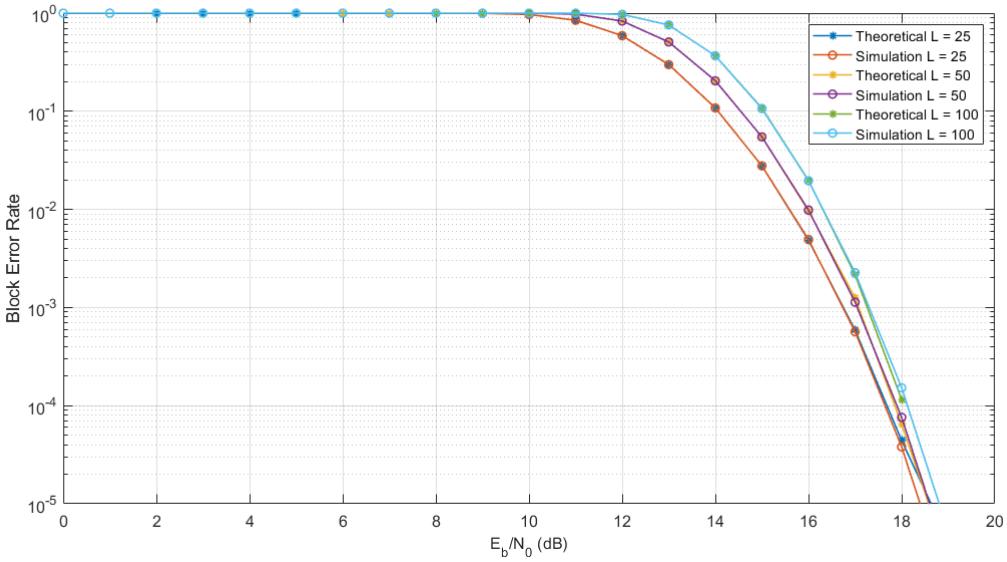


FIGURE 3.4: 4-PAM theoretical and simulated BLER curves in function of the SNR of the channel for different frame sizes, in bytes.

These simulations have been done with different frame lengths, ranging from $L = 25$ to $L = 100$ bytes. Notice that according to (2.4), the BLER depends on the frame length. With more bits per frame, the probability of receiving at least one bit in error increases. This is the reason why, for $L = 25$ bytes, it is needed a lower SNR to obtain a better BLER. Instead, for $L = 50$ and $L = 100$ bytes, the required SNR is higher to achieve the same BLER.

3.3 ARQ simulations

To simulate at the data link layer, different scripts have been prepared. First, Stop-and-Wait ARQ protocol has been implemented. It has the same basis as the BLER

simulation, but with a difference in the behaviour after a frame has been received. When the frame is decoded, the system simulates the emission of a notification signal going through the channel, towards the transmitter. To simplify the design, these signals, ACKs and NACKs are not contaminated by the noise such that they are received without any errors. Once the signals reach their destination, if the received notification is an ACK, the simulation continues with the transmission of the next frame. However, if the notification is negative, the system performs the transmission of the same frame again.

3.3.1 Average frames per transmission

At this point, according to 2.4, the average transmission frames in a total communication is estimated with the aid of numerical simulations. If the channel is more unstable generates more errors, the number of transmissions for each frame becomes higher. However, if the channel noise is constant and weak, then the number of transmission is close to one.

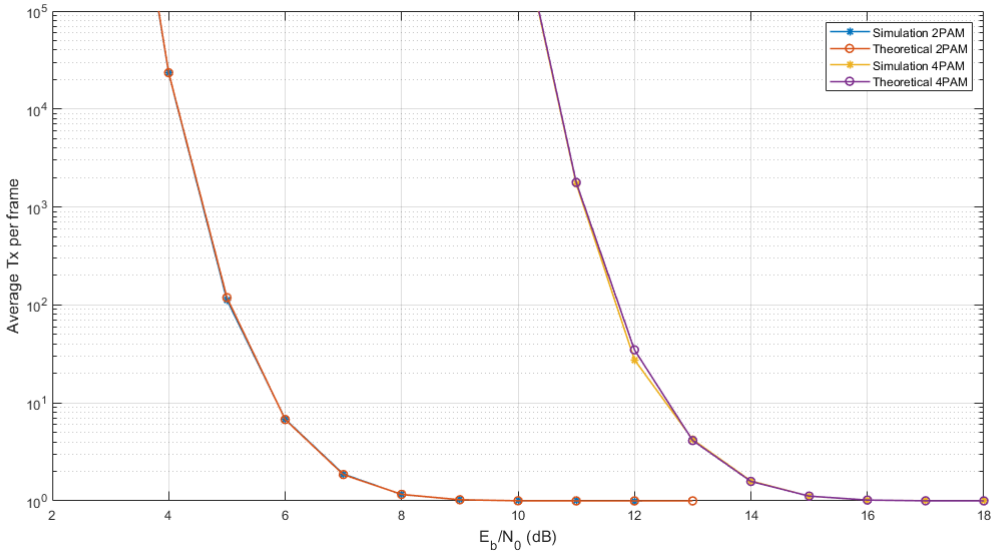


FIGURE 3.5: The average transmissions for each frame versus the SNR of the channel in 2 and 4-PAM. Notice that in a lower SNR bound the transmission average grow asymptotically large.

For this simulation, the timeout event have not been considered for two reasons. First, the script has been implemented in only one computer that takes the two roles, transmitter and receiver. The timeout requires listening continuously to handle the reception of the frame or notify when the designed timer expires. Also, another process is required to handle the main part of the simulation. To solve this problem, there are elements called threads. A thread is a process of sequences chain tasks that can be executed by an operative system. These sequences can be executed in parallel, asynchronously, to carry out more than one task at the same time. Threads can be implemented in advanced software such as Matlab, but it has been found that using them slows down the simulation and alters the results.
The second reason is due to the configuration of the modelled AWGN channel. It is designed to add AWGN, but do not have properties to cause multipath or symbol delays. Therefore, the frames are always detected, and the timeout is not necessary.

These problems are solved in the following chapter with real time simulations.

3.3.2 Utilization of the channel

To continue, a simulation for the Go-Back-N protocol has been designed. The basis is similar to the previous script. Here, a new parameter has been introduced, the sliding window size (w). As explained in section 2.2.2, this protocol takes advantage of the pauses in transmission to increase the number of frames sent continuously. This number of frames is controlled with the value that, the sliding window takes. As the frames can now be received in a continuous form, it may happen that one of them was not detected or out of order. Consequently, the composition of the frame implements an identification number to let the receiver known witch frame is receiving.

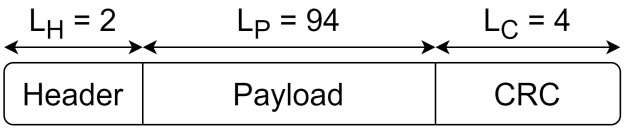


FIGURE 3.6: Overview of the frame structure implemented in the Go-Back-N ARQ simulation. The length of the frame fields is in bytes.

As it was shown in Fig. 2.5, the frames are sent in function of the windows value. Once the reception process the frames, it sends the corresponding notifications back. At transmitter side, the window increments one position for every received ACK. In case a NACK is received, the window stops and the system sends all the frames from this position again.

To design this simulation a new problem as arisen. According to 2.9, the propagation time (T_{prop}), and the frame transmission time (T_{tx}) have to be taken into account. In Matlab, the protocol is running in a non-real-time simulation and due to that, these times are negligible. The propagation time does not exist as the simulations runs in the same computer, and the frame transmission time are very fast and difficult to measure.
To try to solve this problem, the variables have been fixed before the simulation, and the system generates delays and timers with the aim of being as accurate as possible. The transmission data rate have been fixed to $R_b = 1$ Mbps. So the time to generate a frame and be transmitted is calculated as $T_{\text{tx}} = L/R_b = 100$ Bytes $/1$ Mbps $= 0.8$ ms, been L the total frame length. For the propagation time, the time it takes for a frame to travel all the way from the transmitter to the receiver, it has been fixed to $T_{\text{prop}} = 0.5$ ms.

These solutions have helped to run the simulations to obtain the channel utilization (U), for the two different modulation schemes under analysis. Although the simulation is not entirely accurate, the obtained figures have been helpful to understand the protocols and preparing for the change towards real-time simulations.

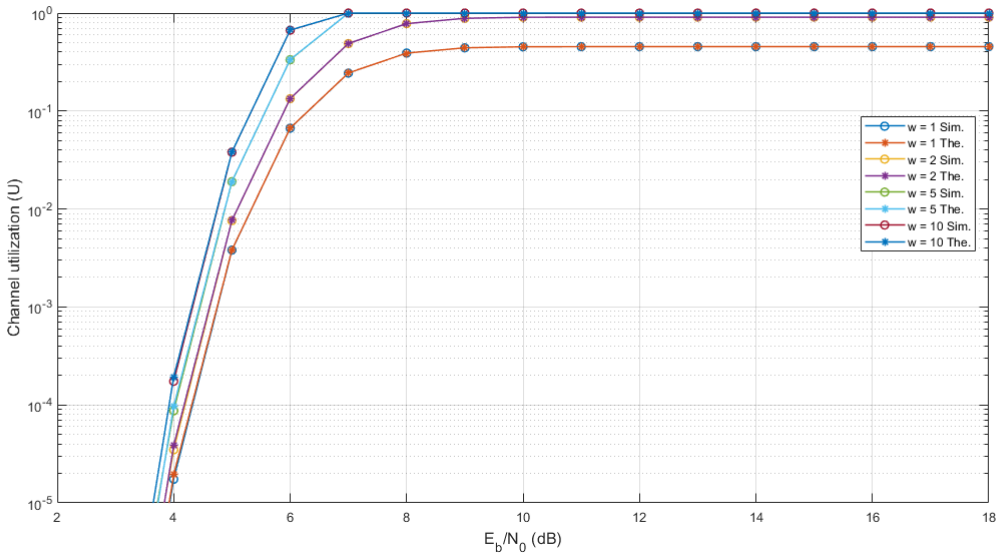


FIGURE 3.7: Utilization of the channel (U) at different windows sizes (w) theoretical and simulated curves for 2-PAM

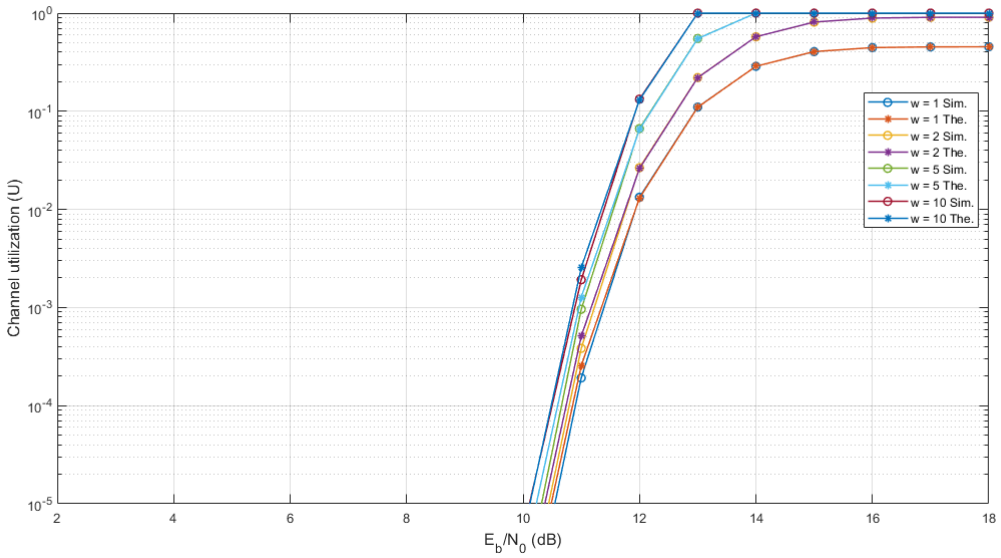


FIGURE 3.8: Utilization of the channel (U) at different windows sizes (w) theoretical and simulated curves for 4-PAM

Fig. 3.7 and Fig. 3.8 have been obtained for 2-PAM and 4-PAM schemes, respectively, with different window sizes. Note that when $w = 1$, a unique frame is transmitted continuously, and the Go-Back-N protocol ends up being the Stop-and-Wait scheme.

In these figures, it is necessary to highlight some values. As it is shown in the equation 2.8, the maximum utilization of the channel $U_{\max}$, is achieved when the BLER is near to zero, and it depends on the values that parameters w, T_{prop} and T_{tx} take. In these cases, for the fixed values enumerated above, the maximum efficiency is $U_{\max} = 0.8w/(1 + 0.8) = 4w/9$. Following this case, for Stop-and-Wait ARQ, $w = 1$ and, due to that, the utilization channel is never 1. However, if the windows size is increase the $U_{\max}$ increases with it, reaching the value expressed after (2.8) when the windows reach its optimum value $w_{\text{opt}} = 3$. Notice that, at this point, $w = 5$, 10 and

further, are useless. The simulation require more computational process and the $U_{\max}$ is saturated in its highest possible value.

Before finishing this chapter, a table that summarizes the different parameters that have been used for the numerical simulations is presented. Note that the payload length, was changed according to the ARQ protocol that was evaluated. More precisely, $L_P = 96$ for Stop-and-Wait ARQ, and $L_P = 94$ for Go-Back-N ARQ allowing the two bytes for the header, L_H. For the sliding windows, different values have been used to simulate different scenarios, in which $w = 1$ corresponds for Stop-and-Wait ARQ, whereas the remaining ones were used for Go-Back-N.
All the frame length values are used for the following chapters. However, the time values have been used to help the simulations and may be subject to change.

Symbol	**Parameter**	**Value**	**Unit**
L	Total frame length	100	Bytes
L_{PR}	Preamble length	22	Bytes
L_P	Payload length	94, 96	Bytes
L_C	CRC length	4	Bytes
L_H	Header length	0, 2	Bytes
R_b	Data bit rate	1	Mbps
T_{tx}	Frame transmission time	0.8	ms
T_{prop}	Propagation time	0.5	ms
RTT	Round Trip Time	1.8	ms
w	Sliding windows	1, 2, 5, 10, ...	

TABLE 3.1: Overview of all the parameters taken into account in the non-real-time simulations

Chapter 4

Real-time simulations

This chapter presents the simulations in a real-time software such as GNU Radio [17]. This development toolkit is a free software that, with the help of signal processing blocks, can implement software-defined radios or, in this case, software-defined VLC systems. The simulation results in this chapter have been obtained without any external hardware. However, in Chapter 5 the software is integrated with the hardware to achieve the main objective of this MSc Thesis.
The GNU Radio real-time simulations obtained here are compared with the ones of Chapter 3, as well as with ones that have been predicted according to the theoretical background. Aim of this is to ensure a good functioning and a smooth integration with the hardware.

4.1 GNU Radio

GNU Radio is composed by processing blocks. Every block has its own function and processes the data in parallel with the others. The set of blocks, connected by wires, forms the layouts.

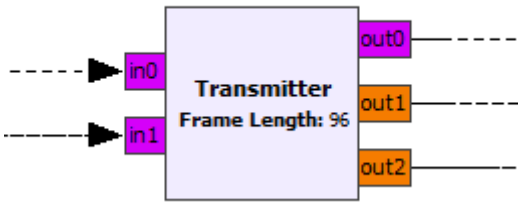


FIGURE 4.1: Overview of a GNU Radio block. On the left side there are two inputs for bytes. On the right side there are three outputs in total: one for bytes and two for number of float type.

Fig. 4.1, shows a sample GNU Radio block. The magenta rectangles on the left side indicate the inputs 0 and 1 for a byte stream. The rectangles on the right side indicate the outputs, magenta for bytes, output 0, and orange for numbers of type float, outputs 1 and 2. Also, different variables can be declared, such as *Frame Length*, to assign the same value in different parts of the system.

To carry out the real-time simulations, a block system has been created. The simplest ones are pre-designed blocks available in GNU Radio. However, for a specific functionality, the software does not have the desired blocks. In these cases, GNU Radio allows to create custom blocks in C++ or in Python. In order to complement the system, different Python blocks have been created.

At this point, different elements have been taken into account. The simulations have been designed to resemble as closely as possible an actual digital communication transmission. First, a preamble has been added before the frame. The preamble, or sync-word, is a sequence of symbols which remains constant in the communication link. The preamble is added at the transmitter side and used at the receiver side to identify the start of the frame and have reference points to deal with the synchronization problems.

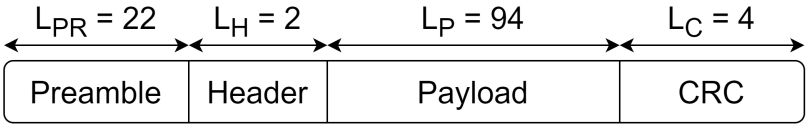


FIGURE 4.2: Overview of the frame structure implemented in the GNU Radio simulations. The length of the frame fields is in bytes.

Secondly, a pulse shape filter has been utilized in transmission. The purpose of this filter is to make the transmit signal better suited for the communication channel, trying to fit as much as possible the frequency response bandwidth. A signal conveying the 2-PAM or 4-PAM symbols may have a very abrupt amplitude variations between consecutive sampling times. This, in the frequency spectrum, leads to a very large bandwidth, that is unfeasible to support in a bandwidth-limited communication channel. To mitigate this effect, a Square Root Raised Cosine (SRRC) filter has been implemented for pulse-shaping purposes in the VLC transmitter.

4.1.1 Block diagram of the VLC transmitter

The transmitter side is in charge of converting the data bytes into bits. After that, the obtained sequence of bits should be accommodated into data blocks, with a CRC that should be appended before the whole frame is sent through the channel. In addition, for the ARQs simulations, the VLC transmitter is also responsible for processing the ACKs and NACKs and, according to the signal event, proceed to (re-)transmit a new (the previous) data frame.

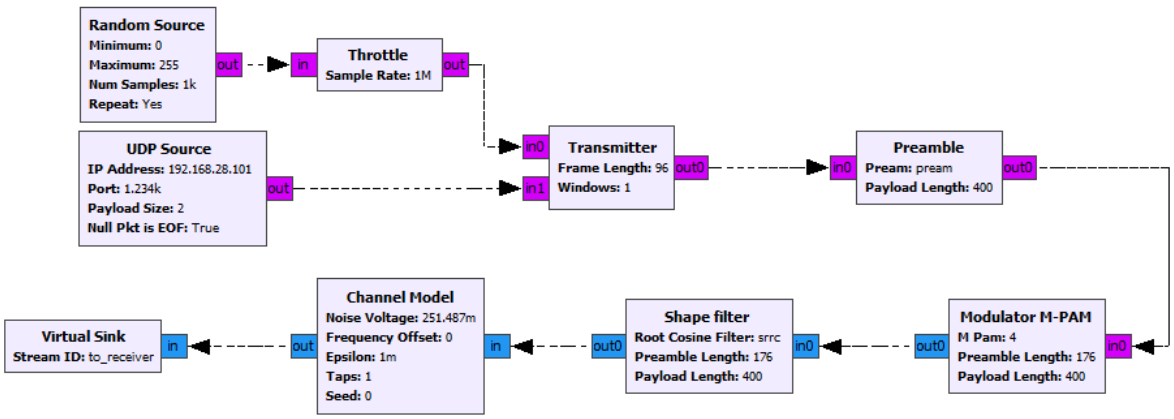


FIGURE 4.3: Overview of the transmitter layout in GNU Radio. The process starts with the creation of random block of bytes, and ends with the signal going through a simulated digital communication channel.

In Fig. 4.3, the layout shows all the pre-designed and custom blocks at the transmitter side. Hereafter, all of them have been explained to understand the proper functionality of the system. The Python code, for the custom designed blocks, have been added in the Annex B of this MSc Thesis.

The Random Source and the Throttle blocks are pre-designed by GNU Radio. The first one generates random bytes to simulate data bits for transmission. The second one is responsible for regulating the speed at which these bytes are sent.

The Transmitter is a custom designed block. Two inputs are implemented to receive the data bytes at the input 0, and the notifications signals at the input 1. Depending on the ongoing modulation, if only the BER or the BLER are calculated, and the ARQ protocols are disabled, the input 1 is inactive. The block has a buffer to save the frames in case a retransmission is needed. In the input 0, the block receives a byte stream of constant length, L_P. If an ARQ protocol is activated, the block waits for a notification signal from input 1. If this notification signal is an ACK, the script discard the last frame and ask for a new one. If the notification signal is a NACK, the script searches into the buffer to retransmit the data frame that was signalled as wrongly received. In the possible event of not receiving any notification, a fixed timeout is implemented to ensure that a retransmission is triggered in such circumstance.
Finally, in all the cases, the script obtains the next right frame, calculates the CRC, appends it after the payload bytes and sends the frame.

The Preamble block is a custom designed block. The received byte frame is converted to bits, and the constant preamble is added at the beginning of the bit stream. At the input, the length of the frames was set to $L = 100$ bytes. At the output, the frame has been converted into bits $L = 100 * 8 = 800$ bits, and the preamble was added. Finally, the total length becomes $L_T = L + L_{PR} = 800 + 176 = 976$ bits.

The Modulator M-PAM block is a custom designed block. Depending on the selected modulation index, the block modulates the payload input bits into 2-PAM symbols, or 4-PAM normalized symbols. The preamble is always modulated with a 2-PAM to increase the options of detection. Notice that at the output of this block, different symbol lengths can be possible for the data frame. When using a 2-PAM, the modulation is one bit per symbol, so the length does not change. However, for 4-PAM, the modulation scheme accommodates two bits per symbol, so the length of the data frame is reduced to half its original size.

The pulse-shaping filter is a custom designed block. It is in charge of applying the SRRC to the input sequence of symbols to control the bandwidth and make the signal more suitable for the transmission. For the filter parameters, different values have been assigned, all of them can be found in the Table 4.1. The pulse-shaping filter is convolutioned with the input sequence of symbols and applies an oversampling factor of $OSF = 5$, to improve the resolution of the time-domain signal and mitigate the aliasing.

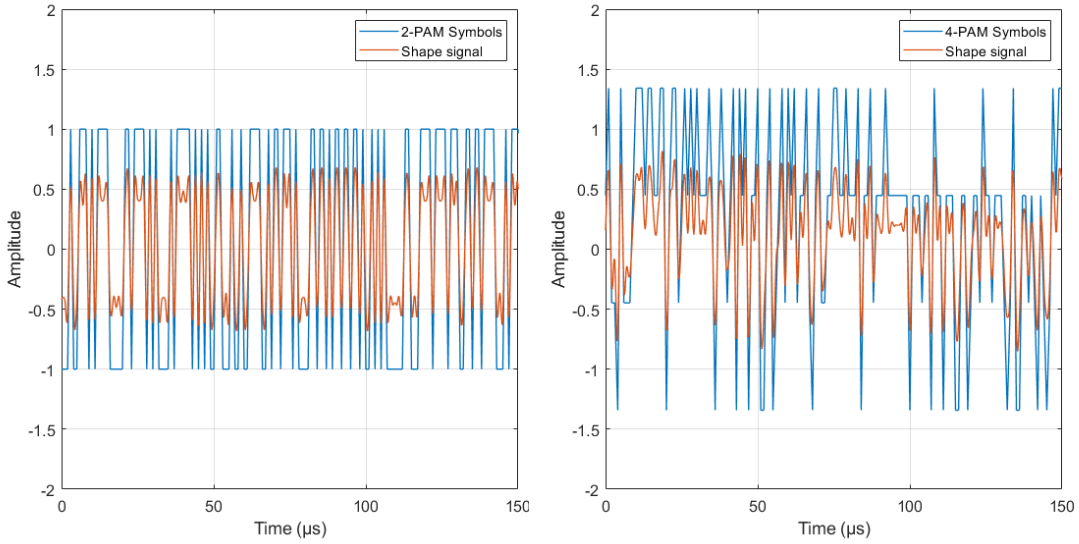


FIGURE 4.4: Overview of the signals generated by the GNU Radio for 2-PAM (left-hand side panel) and 4-PAM (right-hand side panel). The blue signals are at the output of the modulator, and the red signals are at the output of the pulse shape filter.

These signals are 2-PAM on the left-hand, and 4-PAM on the right-hand of the Fig. 4.4. The blue signals are the symbols generated by the Modulator M-PAM, converting the bit stream in to a real-valued digital wave. The red signals are the symbols processes by the shape filter, oversampled and ready to be transmitted. In the 4-PAM it is possible to notice that the symbols never reach the values of $+3$ and -3. This is due to the normalization of the symbols, to transmit the signal with unitary power.

To finish the transmission layout, the Channel Model is a pre-designed block to modify the signal according to the proposed channel model. This block adds AWGN, as well as a time offset so better simulate a real channel. The last block, known as the Virtual Sink, only moves the channel to the Source Sink at the receiver side without any more alteration.

4.1.2 Block diagram of the VLC receiver

The receiver side is in charge of receiving the signal, finding the position of the preamble and synchronizing the received signal samples to frame level. In addition, for the ARQs simulations, the VLC receiver block is also responsible for creating the notification signals and send them to the transmitter.

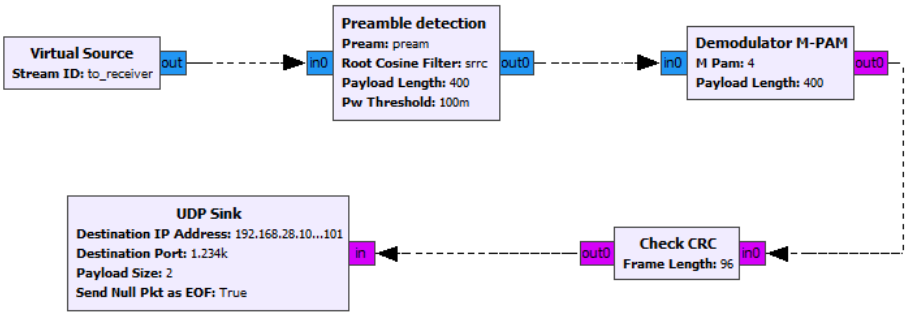


FIGURE 4.5: Overview of the receiver layout in GNU Radio. The block diagram of the receiver starts with the signal reception from the channel, and ends with the verification of CRC and the sending of corresponding ARQ signalling (ACK/NACK).

The Preamble detection is the most complex block to be designed. It is in charge of listening the channel and identifying when a frame has arrived. To achieve this goal, the original preamble is oversampled and convolved twice with the pulse-shaping filter. The intention of this is to have a preamble as similar as possible to the received one.
When the signal arrives, first it goes thought a matched-filter. Then, this outcome is correlated with the modified preamble to find its most likely position in the received signal samples. If the correlation result is higher than a threshold, then a frame is detected. At this point, all the samples in the buffer are processed and oversampled again with a second oversampling factor, OSF_2, to obtain more resolution in the synchronization. The processed samples are analysed to obtain the position where the preamble correlation output has its maximum energy.
Once this position is obtained, the preamble and the frame are synchronized. The received signal samples of the preamble are also used to estimate the attenuation of the channel. This calculated attenuation is used to scale the received signal samples, trying to compensate them to its original power.
Finally, the frame samples are sent to the following block and the other receives samples are discarded.

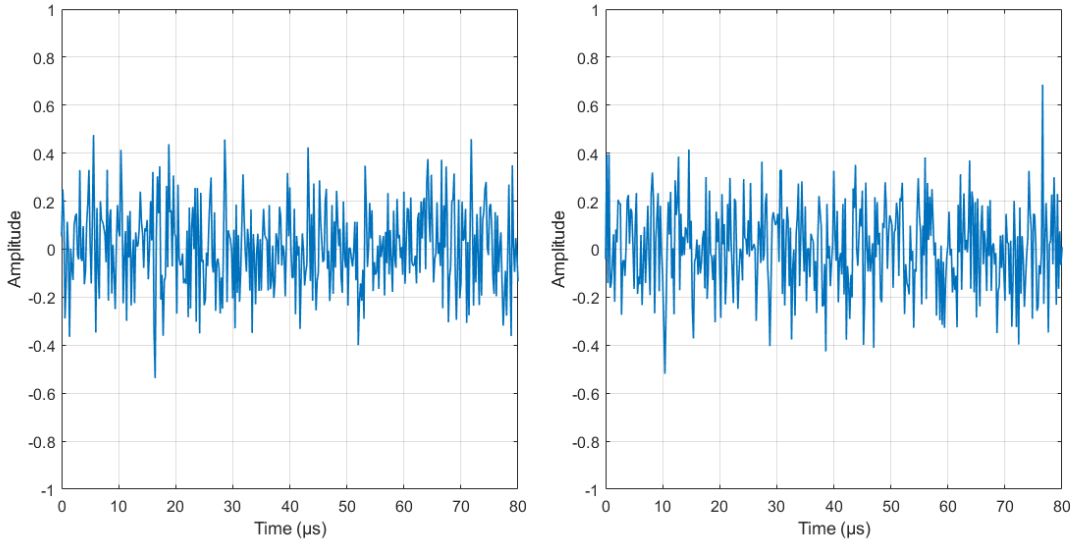


FIGURE 4.6: Overview of the received signals contaminated with noise at the receiver side generated by the GNU Radio. 2-PAM (left-hand side panel) and 4-PAM (right-hand side panel).

The Demodulator M-PAM is a custom designed block. It receives the signal carrying of the information, without the preamble, and demaps the symbols to the corresponding bits. Depending on the modulation, the symbols are demodulated in one or in two bits per symbols. Also, in the case of the 4-PAM, the block denormalizes the signals, scaling the received signals samples with a factor $\sqrt{5}$ to retrieve the original ones.
The obtained bits are packed to bytes and send for its CRC verification.

The Check CRC and the UDP Sink blocks work together. The first one is a custom designed block. It receives a byte stream and realize the CRC division to check whether its transmission has been successful or not. For a bit-level simulation, a predefined text file of information is sent. Then, this block compares the received bytes with the file content to obtain the BER curves. The BLER and ARQ simulations are at a frame level, so the block check the CRC of the same received frame.
For the ARQ simulations, notification signals are generated, and the UDP Sink block is responsible for sending them to the transmitter UDP Source block.

In the GNU Radio simulations, many values have been assigned, and they can be observed in Table 4.1. The different lengths of the frame were kept constant, having usually a payload length of $L_P = 96$, and a header length of $L_H = 0$. In exception to the Go-Back-N simulation, where the payload length was set to $L_P = 94$, the header was $L_H = 2$, and the sliding windows took different values verifying $w > 1$.
The filter values have been chosen to have a unitary power and to avoid large processing times. This is the reason why the number of components is only $N_{\text{TAPS}} = 31$, to minimize the transmitted signal length and to reduce the detection and synchronization time at the receiver.

Symbol	Parameter	Value	Unit
L	Total frame length	100	Bytes
L_{PR}	Preamble length	22	Bytes
L_P	Payload length	94, 96	Bytes
L_C	CRC length	4	Bytes
L_H	Header length	0, 2	Bytes
R_b	Sample rate	1	Mbps
R_s	Symbol rate	250	Kbps
OSF	Oversampling factor	5	
OSF_2	Oversampling factor 2	7	
β	Roll-off factor (SRRC filter)	0.35	
N_{TAPS}	Number of filter components	31	
F_G	Filter gain (SRRC filter)	0.447	
w	Sliding window size	1, 2, 5, 10, ...	

TABLE 4.1: Overview of all the parameters taken into account in the GNU Radio simulations

4.2 BER Simulations

To check that the software-defined transceiver implemented in GNU Radio is prepared for an integration with the hardware, different simulations have been carried out and the obtained simulation results have been compared with the theoretical ones. The same GNU Radio layout that was described in Section 4.1, adapted to perform different kinds of simulations. First, it has been verified that the BER and the BLER have a properly functioning.

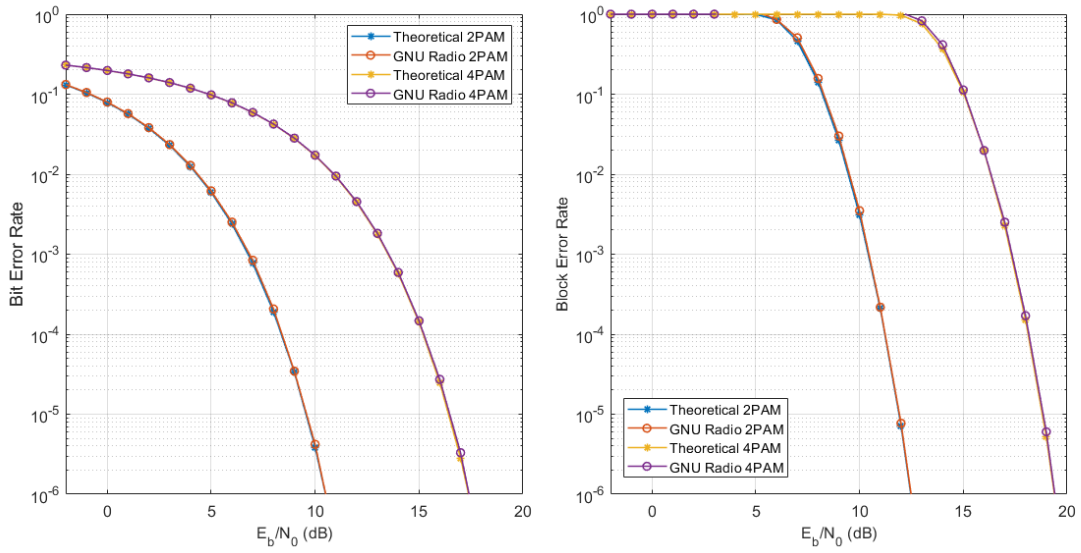


FIGURE 4.7: BER and BLER simulations in GNU Radio for 2-PAM (right-hand side panel) and 4-PAM (left-hand side panel).

To perform these simulations, the UDP blocks that are used to convey the ARQ signalling have been disabled as there is no need to have a feedback communication link.

For the BER simulation, the transmitter block obtains the information to send of a previously established file. This is due to the fact that there is no possible way to compare the bits with only the received frame, unless there is a pre-defined stream of bytes to make the bit-by-bit comparison for the BER computation. However, for the BLER simulation, there is no need to rely on a predefined file to be transmitted. This is because due to the implementation of the CRC, the receiver can verify whether any of the bits of the data frame has been corrupted by just checking the last 4 bytes.
Once the layout was programmed, the curves have been measured with different simulations varying the noise power in the Channel Model block.

4.3 ARQ Simulations

The last set of simulations that were carried out before the hardware integration were the ones related to the ARQ protocols. For this purpose, the UDP blocks where activated to implement a feedback link between the receiver and the transmitter. Note that the encapsulation of ARQ signalling messages over UDP took the role of a virtual link, since the two layouts were running in the same computer. The destination IP address fixed in the blocks was the one of the network interface card of the desktop computer in which GNU radio was running. Therefore, the system sent the notification signals to itself to perform the ARQ simulations.

The first simulation for this section is the Stop-and-Wait ARQ scheme. The sliding window size parameter in the Transmitter block was set to $w = 1$, for a bursty transmission pattern of frames in which only one frame per window was sent. The figure of merit that was used to study the performance of this ARQ scheme was the average number of transmissions needed for each frame until it was properly received in the destination.

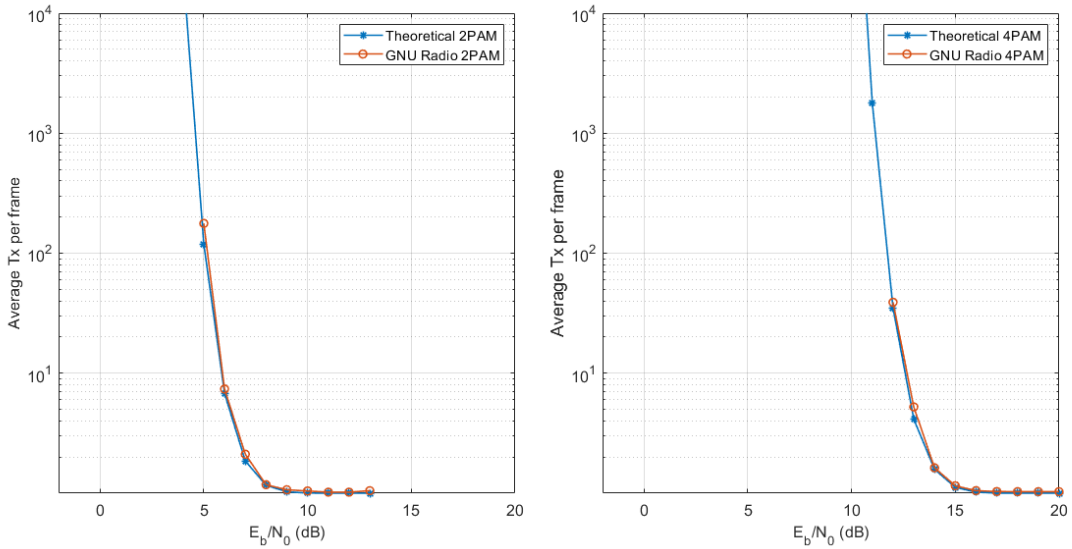


FIGURE 4.8: Average transmissions for each frame GNU Radio simulations in 2-PAM (right-hand side panel) and 4-PAM (left-hand side panel).

To measure the average (re-)transmissions needed per frame, the transmitter counts every sent frame. Then, when an ACK arrives, another counter in incremented. If the notification is a NACK, the same frame is retransmitted but only the total counter is

updated. When the transmission finishes, the total frame counter is divided by the ACK counter to obtain the average. Notice that in the low E_b/N_0 region, the mean number of retransmissions that is needed grows asymptotically large. This has been a problem in the simulations, due to the fact that the probability to obtain only one successful frame is very low.

For the Go-Back-N ARQ simulation setting, a problem was identified which complicated its implementation in practice. Usually, in an application-specific implementation of this ARQ protocol, the processing time in the receiver side is very low due to the use of an asynchronous processing strategy. Therefore, the frames that arrive in the receiver can be processed at the time of arrival, in parallel, and there is no need for a buffer.
In GNU Radio, this methodology is not possible. Though the different blocks work in parallel, the signal processing in the same block is always working synchronously. The possibility to implement threads in Python and use different computer cores has been considered, but the mismatch between Python and GNU Radio libraries has made this implementation unfeasible during the time span of this MSc Thesis.

At this point, a modified Go-Back-N protocol has been implemented. As expected, the first window frame to be sent is the one that takes the longer processing time in the receiver (i.e., preamble correlation, frame synchronization). Then, the main goal of this ARQ protocol is to reduce the processing time of the remaining frames, by sending them in a continuous manner with a sliding window. The rest of the window frames take a shorter processing time as the preamble and the synchronization have already been performed.

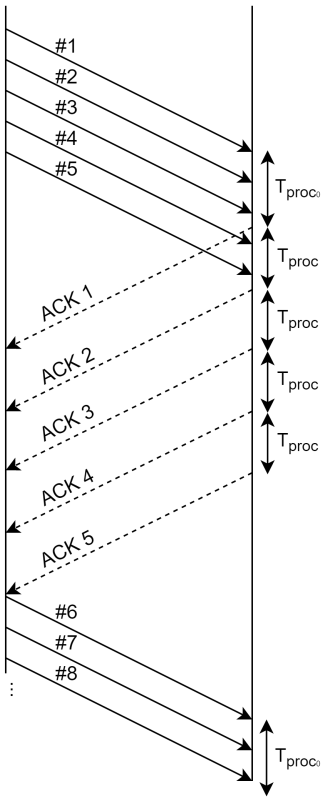


FIGURE 4.9: Example of a modified Go-Back-N protocol in a window equal to $w = 5$. Notice that the process time (T_{proc}) is higher for the first frame of the window.

To evaluate the performance of this modified version of the Go-Back-N protocol, it is possible to adapt the figure of merit in (2.9) to obtain

$$U_m = \frac{wT_{\text{tx}}(1-\text{BLER})}{\text{RTT}} = \frac{wT_{\text{tx}}(1-\text{BLER})}{2T_{\text{prop}} + T_{\text{tx}} + T_{\text{proc}_0} + (w-1)T_{\text{proc}}}, \tag{4.1}$$

which is valid only if $wT_{\text{tx}} < \text{RTT}$ is verified. Notice that the largest processing time (T_{proc_0}) is always present here. On the other hand, the other processing time (T_{proc_0}) is present depending on the window size.
In Stop-and-Wait ARQ ($w = 1$), this problem is present too because the value T_{proc} takes also affect on the overall transmission time. However, the problem has not been noticed in the simulations since the data frame was unique in the sliding window and due to that processing time did not delay any other frames. New figures have been obtained to compare the differences between the theoretical (U) and modified Go-Back-N ARQ channel utilization (U_m).

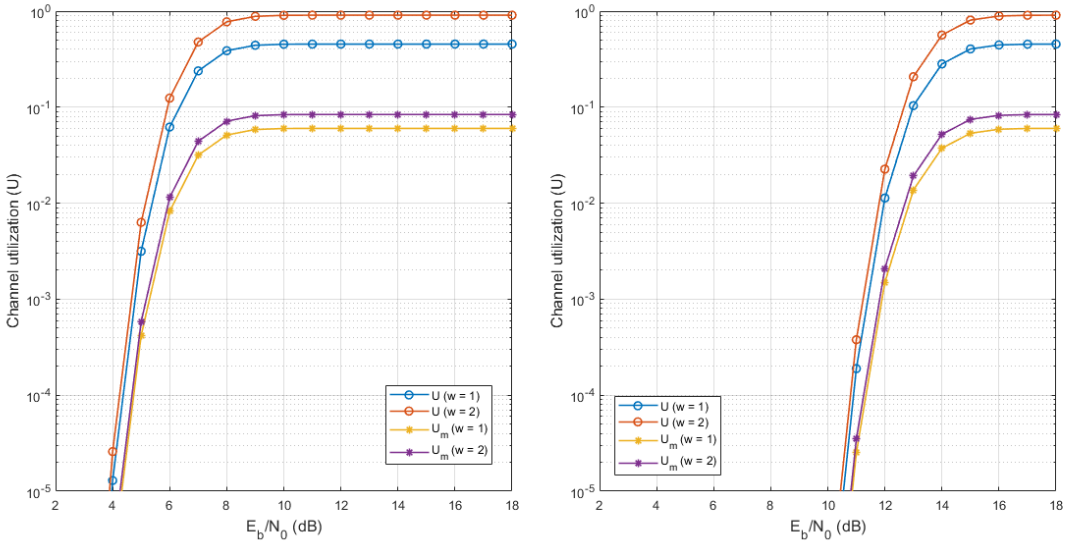


FIGURE 4.10: Comparison between theoretical (U) and Go-Back-N modified ARQ channel utilization (U_m) for 2-PAM (right-hand side panel) and 4-PAM (left-hand side panel).

Notice that, with an increase of the window size, in the high E_b/N_0 regions, the difference is more relevant in the theoretical (U) case. For the modified ARQ scheme, according to (4.1), there is an influence of the windows size in the denominator. This effect mitigates the effect of the windows size in the figure of merit that was used to evaluate performance of this ARQ scheme.

Finally, these calculations have been done with measured times obtained in the GNU Radio simulations. As the transmitter and the receiver layout was executed in the same computer, the propagation time was negligible in practice. To make the effect of this parameter more visible in the simulations, a delay has been added explicitly in the transmission chain.

Symbol	Parameter	Value	Unit
T_{proc_0}	First frame processing time	12	ms
T_{proc}	Non-first frame processing time	6	ms
T_{tx}	Frame transmission time	0.8	ms
T_{prop}	Propagation time	0.5	ms

TABLE 4.2: Overview of all the measured times for the GNU Radio to run an ARQ simulation. Notice that the secondary frames require half of the time to be processed.

Chapter 5

Integration and evaluation

This chapter aims at merging together all the previous simulations with the hardware integration. To begin with, the USRPs have been added. These devices, are designed to connect a host computer through a high speed link. With this connection, the computer uses a software, usually open source (e.g., GNU Radio), to transmit and/or receive information. At the transmitter side, the desktop computer sends the information and the USRP transforms it to an optical signal with the aid of a LED lamp (with a LED driver). At the receiver side, these optical signals are optical-to-electrical converted with a PD and, after that, the USRP is in charge of transforming them into digital information to be read by the GNU Radio.

5.1 Coaxial cable

The first integration is designed with a coaxial cable. The purpose is to observe how the implemented GNU Radio system behaves in a controlled channel. As the LED and PD channel is more unstable and attenuating. This test is a previous check before going forward.

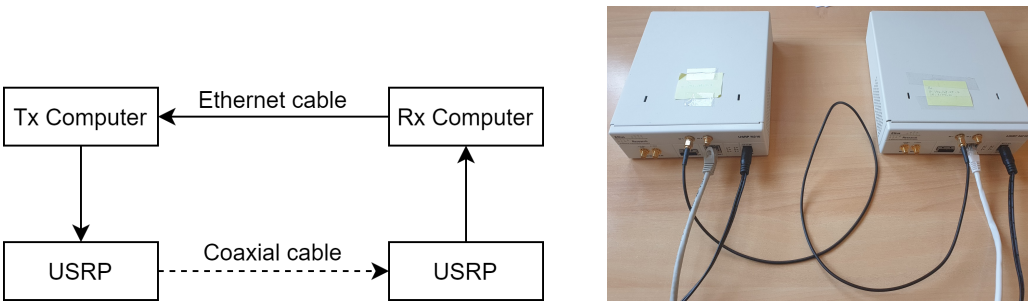


FIGURE 5.1: Overview and picture of the hardware integration. The computers are connected to the USRPs using Ethernet and the USRPs are connected among them by coaxial cable. The feedback channel is also with an Ethernet cable.

For this implementation, as it is shown in Fig. 5.1, the transmitter computer is connected with a high speed link to the USRP. At the same time, this one is connected to the other USRP with a coaxial cable, creating the communication channel. Afterwards, the receiver computer also uses a high-speed link to read the incoming information. Finally, the feedback channel that is needed for the ARQ protocols is implemented with an Ethernet cable connecting directly the two computers.

Before this test, a pre-designed block for the reception and detection of the frame was chosen. However, with the analysis of this system, the block began to behave incoherently and inconsistently. The source code was searched to understand how it works, but even with this support it was impossible to understand the behaviour of

the block. Finally, the Preamble detection block show in Fig. 4.5 was implemented.

The implemented block relies on the preamble detection to synchronize the frame. In the channel created by the coaxial cable, this was a problem. The block was designed to always listen to the channel looking for the preamble, and when the USRPs are turned on and the transmission is stopped, the receiver always collects samples of noise. As the correlation calculation is slower than the reception of samples, the block buffer was quickly full of useless noise, which makes the preamble detection process very unreliable.
To solve this problem, the first implementation was to change the python library in charge of the correlation. The *scipy* library is five times faster than the *numpy* library with this calculation. However, as the problem still remained, the calculation time was not fast enough.
Finally, the solution was to implement a low-pass filter to limit the noise, such that only when the total power of the received signal exceeded a pre-defined threshold, the received samples were correlated with the expected preamble samples; otherwise, the samples were tagged as noise and discarded.

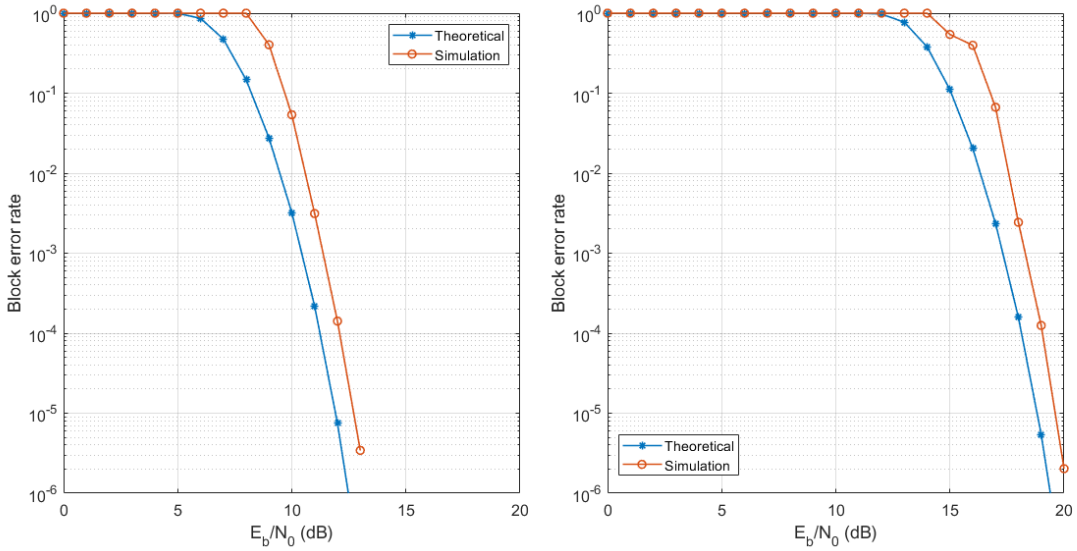


FIGURE 5.2: BLER curves in 2-PAM and 4-PAM, respectively, obtained in the first integration test.

5.2 LED/PD in the loop

Finally, the integration with the LED and PD in the loop was carried out. For this purpose, the coaxial cable was replaced with a LED with a driver at the transmitter side. In reception, a PD was connected into the USRP to capture the variations of the intensity of the received light beam. The LED driver has been designed by another student in his MSc Thesis, and the details of its implementation can be found in [18]. The LED driver was in charge of converting the electric waves coming from the USRP into a more suitable signal to drive the current of the LED. The LED interpreted these signals as a variation in the light intensity, changing the light fast enough to become imperceptible for the human eye.

At the receiver side, two PD have been tested. Firstly, the PDA8A [19] model from Thorlabs that had a fixed gain that did not give the flexibility to regulate the received power. This was a notable drawback of this PD model as it greatly limited the distance

between transmitter and receiver. Secondly, the PDA36A [20] from Thorlabs, with an adjustable gain, that could be tuned according to the desired coverage distances. This is the reason why the chosen PD for the last integration was the PDA36A, which was able to reach longer distances when the regulated gain was properly increased. However, it is important to highlight that the augments of the gain reduces the electrical response bandwidth of the PD and, consequently, this may reduce the achievable data rate of the VLC link.

5.2.1 Performance evaluation of a direct transmission scenario

The complete hardware integration results were measured in a room, where the LED was placed in a fixed position in one extreme of the room, while the PD was arranged on a movable table to give flexibility in the selection of the separation from the transmitter. The measurements were carried out by sending $N = 10\,000$ frames between computers in each transmission. For every distance, the BLER and ARQ protocols with different sliding window sizes were tested.

FIGURE 5.3: Overview and picture of complete hardware integration with the LED and PD in the loop.

In Fig. 5.3, it is possible to see the designed system. The feedback link between the receiver desktop computer and the transmitter desktop computer was finally implemented with the aid of a UDP cable to ensure the correct reception of the signalling frames. The option to implement another optical wireless link (possible on infrared) link for the feedback communication was discarded because, if a blockage occurs, the two links are compromised and the ARQ protocol would become unreliable. Note that this is the reason why, in a practical system implementation, it would be recommended to use another communication media, such low-power radio standards like Bluetooth or ZigBee for the feedback link.

The procedure followed in the measurements started studying the figure of merits with short distances and small gains in the PD. After that, the distance started to increase gradually until the communication became unfeasible. At this point, the gain at reception was increased by 10 dB and the process was continued, until the maximum gain of 30 dB in the PD was achieved. With the collected results, different figures of merits have been obtained.

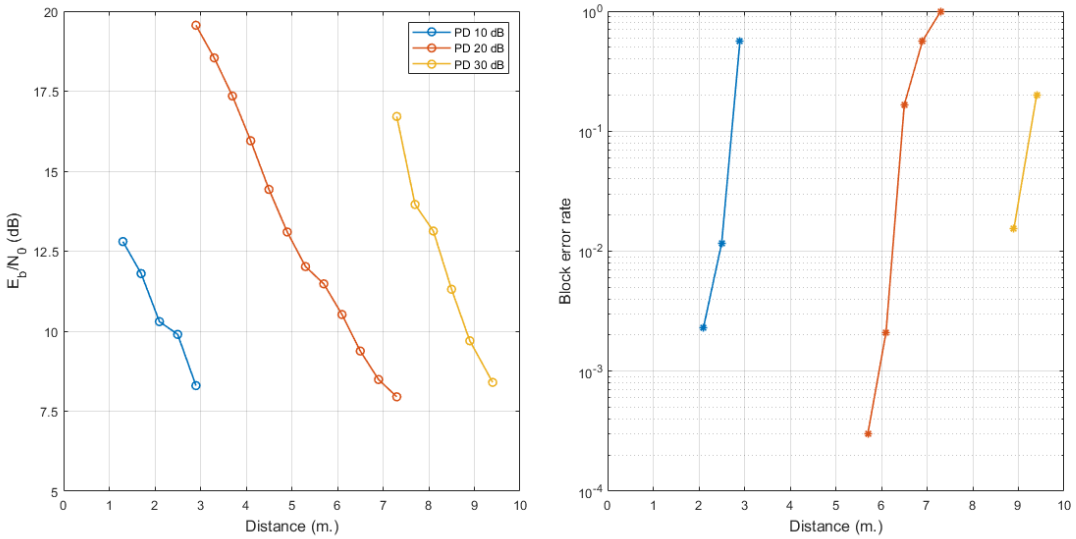


FIGURE 5.4: E_b/N_0 (left-hand side of the picture) and BLER (right-hand side of the picture) for the 2-PAM. These curves were obtained with different PD gains in comparison with the distances.

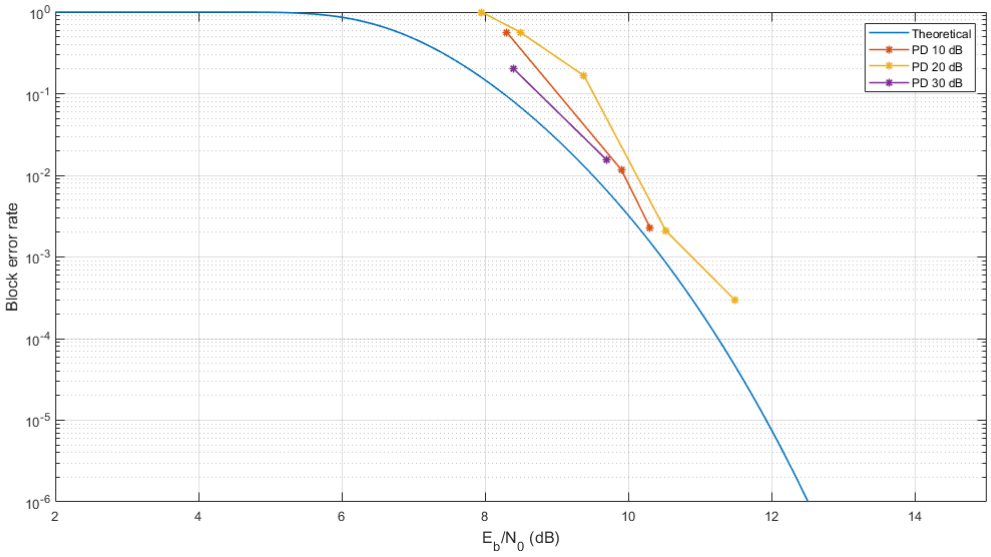


FIGURE 5.5: BLER curves compared with the calculated E_b/N_0 for 2-PAM depending on the distance with different PD gains.

Fig. 5.4 shows the behaviour of the E_b/N_0 and the BLER for different distances. In the right-hand side picture, the BLER that was obtained was acceptable for all the possible PD gains when the distances were less than 10 meters. However, with the increase of 0.4 meters in each new measurement, the value grew asymptotically, making a very clear distance threshold for every PD gain. Also, in the left-hand side of the Fig 5.4, it is possible to appreciate an increment of approximately 10 dB for every time the PD gain was changed.
In Fig. 5.5, the same behaviour can be observed. That is, for a low E_b/N_0 values, the BLER easily goes higher than expected and the transmission is unfeasible. However, with the increase of the E_b/N_0 (i.e., decrease of distance), there is a clear separation distance breakpoint after which the transmission behaves correctly.

In the same experiments, the total time of the transmission was measured for different sliding windows sizes. In the Table 5.1, the utilization of the channel has been calculated and compared with the theoretical presented in (4.1).

Window size	**LED/PD Total time (s.)**	**LED/PD Efficiency (U)**	**Theoretical Efficiency (U)**
1	140.92	0.042	0.056
2	75.04	0.079	0.083
5	61.58	0.096	0.119
10	55.80	0.106	0.139
100	53.13	0.111	0.163

TABLE 5.1: Measured times and calculated channel utilization (U) for a different sliding windows sizes.

Note that, in the first increments of the sliding window size, the total time and the efficiency are significantly improved (i.e., the total time is reduced in half at $w = 2$). However, if the windows continues increasing, the improvement in every step becomes less noticeable.
Another thing to highlight is the poor impact of the sliding windows size for this system; for example, by increasing the window size 100 times a maximum efficiency of $U = 0.111$ can be obtained. This is due to the processing time in reception. To avoid an overload, the transmitter needs to wait a long time period to send the following frames and, consequently, the channel is unused most of the time.

Finally, in the process of these experiments, some values have been adjusted or further improved for better performance. Notice that, the propagation time T_{prop}, has been modified to be more in line with the reality of this direct light transmissions.

Symbol	**Parameter**	**Value**	**Unit**
T_{proc_0}	First frame processing time	14	ms
T_{proc}	Non-first frame processing time	5	ms
T_{tx}	Frame transmission time	0.8	ms
T_{prop}	Propagation time	1	μs
R_b	Sample rate	1	Mbps
w	Sliding window size	1, 2, 5, 10	
TO_1	ARQ protocol Timeout ($w = 1$)	25	ms
TO_w	ARQ protocol Timeout ($w > 1$)	$(w-1)T_{\text{proc}} + TO_{w1}$	ms

TABLE 5.2: Overview of all the measured and selected values for all the integration experiments.

Table 5.2 summarizes the used equation for the timeout in the Stop-and-Wait ($w = 1$) and the Go-Back-N modified ($w > 1$) protocols. In the case of Stop-and-Wait ARQ protocol, the total timeout was fixed $TO_1 = 25$ ms. However, with the increase of the sliding window size, all the frames in a window took a longer time to arrive, although the total transmission and the frame mean time were lower. This is the reason why the timeout implements an equation to calculate the required time depending on the sliding window size.

5.2.2 VLC-based indoor monitoring experiment

Other application for the VLC transmissions is the monitoring of the environment. In many buildings where LEDs are already installed, the infrastructure could be exploited to implement transmissions for monitoring through the light.
With the help of the ARQ protocol, the time an obstacle is blocking the light beam can be measured. Recollecting this information for many events, it may be possible to find patterns for each one of them, knowing when a person pass by walking or running. Using this same principle, it could also be sensed when two persons pass by the light beam at the same time. It would also be possible to detect the direction in which the moving persons are crossing [12].

The last part of this MSc Thesis was focused in collecting information for different events. A circuit has been designed, where a person crosses the light beam walking, running or walking jointly with another person.

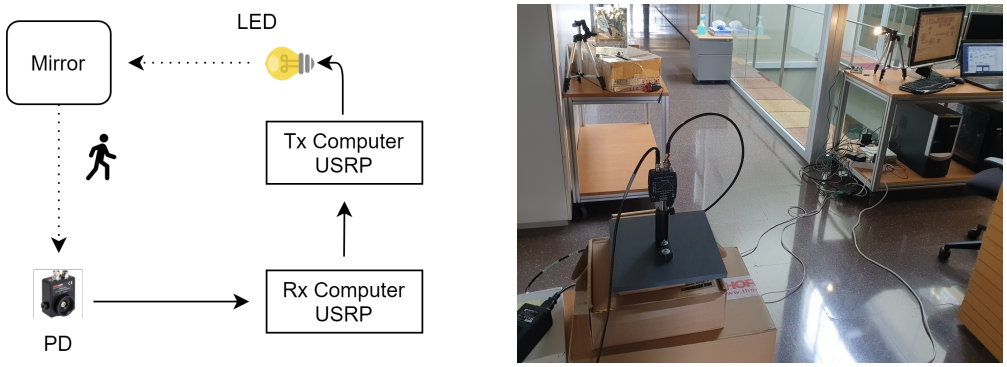


FIGURE 5.6: Overview and picture of the hardware integration for monitoring measurements. A person cross the light beams between the mirror, the LED and the PD.

As it is shown in Fig. 5.6, the experimental layout was implemented with a mirror. The first beam exit from the LED and was reflected in the mirror towards the PD. The intention was to analyse the behaviour of the system when the light is reflected by a mirror and also, to have two beams of light to pass through. Considering that the person always crosses the two beams depending on its velocity, the time gap between the two blockages is different.
It is necessary to highlight that, in these experiments, the timeout at the transmitter side was very related to the resolution of the results. When a blockage occurred, the timeout was able to control how long the system had to wait until the frame was transmitted again. When the timeout was high, the retries took longer and the collected information was lower. In these gaps between retransmissions, the person was moving between beams, releasing one beam and blocking the other. Because of that, it had to adjust as much as possible to match the frame plus ACK time.

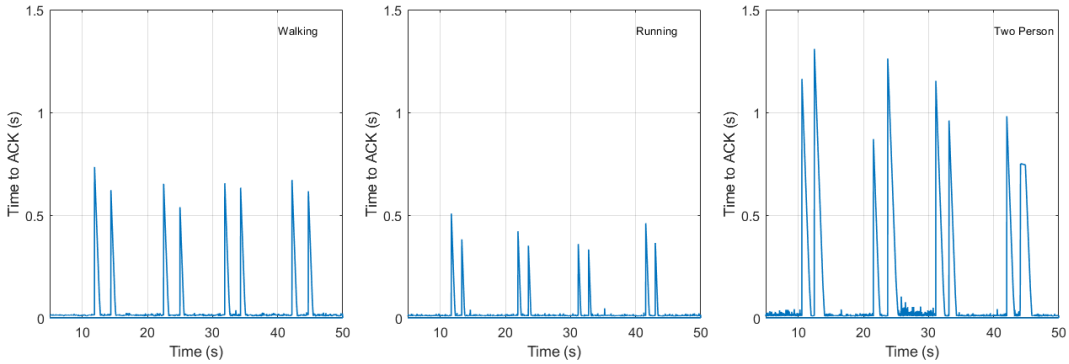


FIGURE 5.7: Time-to-ACK when the light beam between the LED-mirror and the mirror-PD is blocked by different events. Walking (left-panel), running (centre-panel), and two-people walking (right-panel).

Fig. 5.7summarizes the different time-to-ACK that were obtained in the monitored events. The peaks in the figures corresponds to the time frames that were not able to be received correctly due to blockages. When finally the obstacle stopped blocking the corresponding beam, the retransmitted frame arrived correctly and the transmission flow was able to be continued smoothly. In the experiments, the person crossed the two beams, waited approximately for ten seconds, and returned to its started point again in the reverse direction. This routine is repeated twice for each experiment, giving as result that the same beam is crossed four times.
The patterns between the three figures can be perfectly seen. In the second figure, when a person crosses running, it is possible to appreciate a lower time of blockage, as the peaks are smaller. Also, a lower time gap between the two consecutive beams can be appreciated in this situation. In the last figure, two persons have cross the beam together, making a bigger obstacle and, consequently, increasing the time for the blocked frame to be properly received.

Chapter 6

Conclusions

The use of ARQ protocols is very helpful in VLC to prevent data loss due to the channel blockages. These unexpected obstructions, that are likely to happen in a VLC link, cause the loss of synchronization due to that, the receiver needs to spend time resources to recover from this loss of timing. To solve this problem, a hybrid system that uses the wide-area coverage of 4G or 5G synchronization signals may be used, helping in the areas where VLC has disadvantages.

On other hand, the advantage of VLC to be used in an existing infrastructure using LEDs may solve some blockages problems. For example, when a full-blockage event takes place, the ARQ protocol notifies it to the transmitter, and this may decide to use another LED in the room as an alternative point of transmission [5].
This so-called Joint Transmission Coordinated Multi-Point scheme represents an alternative to the Hybrid-ARQ protocols used nowadays for radio communications. In these protocols, when a frame is erroneous, it is sent with an error control coding with variable code rate (redundancy) to ensure its correct reception. As it has been studied, when an object blocks the light beam the SNR asymptotically decreases turning the channel useless. Therefore, these hybrid protocols are not suitable for VLC schemes but, the ARQ signalling that is generated in such circumstance can be used to trigger the process of finding an alternative transmission point.

Another element to note in experiments that were carried out, is the lower impact of the Go-Back-N ARQ protocol. Usually, the sliding window of ARQ protocols are implemented in environments where the propagation time is high, helping to fill the gaps when the information is travelling. In the VLC transmissions, this time is short, and the optimum sliding window is close to one.
Finally, the monitoring measurements carried out in the last chapter of the MSc Thesis have become an interesting topic of study, which paved the way for the writing on a conference paper [12], which has been submitted for evaluation to the Global Communications Conference, which will be held in Madrid in Dec. 2021.

As a future line of research, it is interesting to note that further studies are required in the field of VLC monitoring. With a higher resolution system, the events between the beam of light can be more easily determined. If a lot of information is extracted from these events, an Artificial Intelligence (AI) system may be capable to read them and find additional patterns. At some point, it may be able to find elements to identify the gait of each person and, therefore, identify the person crossing the beam.

Bibliography

[1] W. Jiang, B. Han, M. Habibi, and H. Schotten, "The road towards 6G: A comprehensive survey," *IEEE Open Journal Communications Society*, vol. 2, pp. 334–366, Feb. 2021.

[2] P. Yang, Y. Xiao, M. Xiao, and S. Li, "6G wireless communications: Vision and potential techniques," *IEEE Network*, vol. 33, pp. 70–75, Jul./Aug. 2019.

[3] A. S. Cacciapuoti, K. Sankhe, M. Caleffi, and K. R. Chowdhury, "Beyond 5G: THz-based medium access protocol for mobile heterogeneous networks," *IEEE Communications Magazine*, vol. 56, pp. 110–115, Jun. 2018.

[4] N. Chi, Y. Zhou, Y. Wei, and F. Hu, "Visible light communication in 6G," *IEEE Vehicular Technology Magazine*, vol. 15, pp. 93–102, Dec. 2020.

[5] A. A. Dowhuszko and A. I. Pérez-Neira, "Achievable data rate of coordinated multi-point transmission for visible light communications," *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–7, 2017.

[6] ESA / AOES Medialab, "The electromagnetic spectrum," https://sci.esa.int/s/8OQPZpw, 2012.

[7] M. C., . Zou, L. Zhang, and N. Chi, "Demonstration of a 2.34 Gbit/s real-time single silicon-substrate blue led-based underwater vlc system," *IEEE Photonics Journal*, vol. 12, pp. 1–12, Feb. 2020.

[8] P. A. Haigh, Z. Ghassemlooy, S. Rajbhandari, I. Papakonstantinou, and W. Popoola, "Visible light communications: 170 Mb/s using an artificial neural network equalizer in a low bandwidth white light configuration," *Journal of Lightwave Technology*, vol. 32, pp. 1807–1813, May. 2021.

[9] S. Singh, G. Kakamanshadi, and S. Gupta, "Visible light communication-an emerging wireless communication technology," *2015 2nd International Conference on Recent Advances in Engineering Computational Sciences (RAECS)*, pp. 1–3, 2015.

[10] C.-M. Lee, P.-H. Chen, and Z.-A. Chen, "Visible light communication system with arq applied to smart mobile device," *2017 International Conference on Applied System Innovation (ICASI)*, pp. 280–283, 2017.

[11] H. Yang, W. Zhong, C. Chen, and A. Alphones, "Integration of visible light communication and positioning within 5G networks for internet of things," *IEEE Network*, vol. 34, pp. 134–140, Sept. 2020.

[12] J. Bas, J. A. Ortega, M. Busquets, and A. Dowhuszko, "Indoor monitoring system based on ARQ signaling generated by a visible light communication link," *IEEE Global Communications Conference*, vol. 1, pp. 1–6, Dec. 2021. Under evaluation.

[13] X. Li, A. Gani, R. Salleh, and O. Zakaria, "The future of mobile wireless communication networks," *International Conference on Communication Software and Networks*, vol. 1, pp. 554–557, Feb. 2009.

[14] I. O. for Standardization, "IEEE International Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 2: Logical Link Control," *ISO/IEC 8802-2:1998 ANSI/IEEE Std 802.2, 1998 edition (Incorporating ANSI/IEEE Stds 802.2c-1997, 802.2f-1997, and 802.2h-1997)*, pp. 1–256, 1998.

[15] O. Hasan and S. Tahar, "Performance analysis and functional verification of the stop-and-wait protocol in hol," *Journal of Automated Reasoning*, vol. 1, pp. 1–33, Sept. 2008.

[16] E. G. Varthisa and D. I. Fotiadis, "A comparison of stop-and-wait and go-back-n arq schemes for ieee 802.11e wireless infrared networks," *Computer Communications*, vol. 1, pp. 1015–1025, Jul. 2005.

[17] Gnuradio community, "GNU Radio," https://gnuradio.org, 2001.

[18] J. A. Ortega, "Design and practical evaluation of an led driver circuit for a software-defined visible light communication system," Master's thesis, Universitat Politècnica de Catalunya, 2021.

[19] *PDA8A Operating Manual - Silicon Amplified Detector*, ThorLabs, 12 2018, rev B.

[20] *PDA36A(-EC) Si Switchable Gain Detector*, ThorLabs, 7 2017, rev E.

Appendix A

Contributions

As the outcome of this project, in collaboration with the supervisors of this MSc Thesis, the following paper has been submitted to the IEEE Global Communications Conference, 2021.

J. Bas, J. A. Ortega, **M. Busquets**, and A. Dowhuszko, "Indoor monitoring system based on ARQ signaling generated by a visible light communication link", *IEEE Global Communications Conference*, pp. 1–6, Dec. 2021. Under evaluation.

Indoor monitoring system based on ARQ signaling generated by a Visible Light Communication link

Joan Bas*, Jose Antonio Ortega †, Martí Busquets†, and Alexis A. Dowhuszko‡
*Centre Tecnológic de Telecomunicacions de Catalunya (CTTC/CERCA), 08860 Castelldefels, Spain
†Universitat Politècnica de Catalunya (UPC), 08034 Barcelona, Spain
‡Department of Communications and Networking, Aalto University, 02150 Espoo, Finland
Email: joan.bas@cttc.es; {jose.antonio.ortega,marti.busquets}@estudiantat.upc.edu; alexis.dowhuszko@aalto.fi

***Abstract*—Visible Light Communications (VLC) is a candidate technology that complements the benefits of radio communication networks, particularly in those situations where a low-cost solution using license-free spectrum is required to enable ultra-dense deployments of small cells indoors. However, the main drawback that VLC has when compared to wireless communications on RF bands is that, in presence of obstacles between the transmitter and the receiver, full-blockage events are likely to happen as the power received on higher-order reflections is much weaker than the power of the blocked line-of-sight link. In this paper, we take advantage of this phenomenon and study the effect that different activities performed by people in the service area to-be-monitored have on the Automatic Repeat Request (ARQ) signaling of the VLC link. Based on the presented experimental studies, it is possible to conclude that different relevant events are able to be detected correctly according to the statistics of the ARQ signaling that is collected from the ongoing VLC transmission.**

***Index Terms*—Visible Light Communications; Automatic Repeat Request; Indoor sensing; Activity recognition; Software-defined demonstration; Line-of-sight blockage.**

I. Introduction

Visible Light Communication (VLC) systems encode data into fast changes on the intensity of a visible light source, which are imperceptible to the human eye. Unlike conventional Radio Frequency (RF)-based wireless systems that require advanced digital signal processing, particularly when using Millimeter-Wave (5G) and Tera-Hertz (6G) RF bands [1], a VLC system utilizes low-cost, mass-produced, energy-efficient LEDs as transmitter [2]. At the receiver side, a VLC system relies on a Photodetector (PD) that senses the changes on the intensity of the *light signal* to estimate the data symbols that were transmitted. Apart from data communication and illumination services [3], VLC technology can also be used to monitor the status of the indoor environment by sensing the effect that people create on the received optical signal [4].

Nowadays, there is a wide range of technologies that could be used to monitor the status of an indoor environment. Apart from well-known camera based-surveillance systems and infrared sensors, there are also RF-based solutions that take advantage of the ubiquitous availability of Wi-Fi and 4G/5G nodes to sense the variations that the RF signals experience during propagation [5]. Most of these RF-based solutions require a wireless-enabled device on the target object to perform *active* monitoring, but there are also radio communication systems that carry out a *passive* sensing of the effect that moving obstacles (like people) create on the Received Signal Strength Indicator (RSSI) and/or Channel State Information (CSI) of an ongoing wireless data transmission [6]. We note that both RSSI and CSI signaling belong to the PHY-layer of the wireless communication standard and that, so far, no passive monitoring solution has studied in detail the effect that people's activities create on the Data Link-layer signaling of an active wireless link (*e.g.*, ARQ messages and events).

Automatic Repeat Request (ARQ) is an error control mechanism that wireless communication systems use to ensure the delivery of packetized data in the correct sequence. In an ARQ scheme, Cyclic Redundancy Check (CRC) is used to detect whether the received packet is in error or not. Therefore, when the Signal-to-Noise-plus-Interference power Ratio (SNIR) of the wireless link falls below the sensitivity threshold of the terminal, the payload of the received packet will be likely corrupted (even after forward error correction decoding). In this situation, a Negative Acknowledgment (NACK) is feedback to the transmitter, and the re-transmission of the wrongly received packet is scheduled. If the transmitter does not receive a positive Acknowledgement (ACK) before the *time-out* timer expires, the packet is also re-transmitted [7]. ARQ is used by the MAC protocol of Wi-Fi to detect collisions in the contention-based (shared) radio channel, and can be combined with forward error correction coding of variable rates to maximize the chances of a reliable communication when multiple re-transmissions are triggered due to deep fading and/or strong interference events in mobile networks (4G/5G) [8]. Note that when the frequency of the wireless carrier moves beyond RF bands into the VLC bands, the presence of full-blockage events become more likely, and the ARQ signaling can also be used for indoor monitoring purposes.

In this paper, we study the effect that different people's activities create on the ARQ signaling of an ongoing VLC data link. In the experimental setting, a Phosphor Converted LED and a silicon PIN-based PD were used in the VLC transmitter and receiver, respectively. Both LED and PD were pointing to a large white wall, resembling the indirect illumination scenario presented in [9]. When people passed by the (direct) LED-wall link and (indirect) wall-PD link at different paces, a different pattern of ARQ signaling (*i.e.*, ACKs, NACKs, and time-outs) was generated. These representative ARQ signaling patterns were studied in detail for different people's activities,

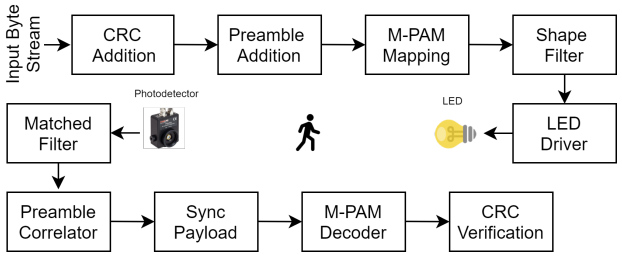


Fig. 1: Overview of the VLC link based on single-carrier M-PAM symbols and a Stop-and-Wait ARQ scheme. The statistics of the ACKs, NACKs, and time-out events in the transmitter are used to identify the presence of people carrying out different activities.

showing notable variations that pave the way for using this information to train advanced machine learning classifiers.

The rest of this paper is organized as follows: Section II explains the software-defined VLC system that was implemented, whereas Section III provides the details of the ARQ signaling scheme and the geometry of the different people's activity recognition problems that we aim to solve. Section IV explains the demonstration setting and carries out the analysis of the ARQ signaling that was collected on the different experiments. Finally, conclusions are drawn in Section V.

II. Implementation Details of the VLC Link

The block diagram of the software-defined VLC link that was implemented to collect the ARQ signalling is shown in Fig. 1. It consists of a software-defined single-carrier M-PAM transmitter that generates a real-valued baseband signal, one LED driver that adapts the output voltage of the USRP to the input current of an LED array with 7 LUXEON Rebel Plus LXML-PWC1-0100 (cool-white) white LEDs [10], and a Thorlabs PDA100A2 detector with switchable gain Transimpedance Amplifier (TIA) [11] that interfaces the (weak) current coming from the silicon PIN diode to the input voltage that the USRP needs to obtain the received signal samples.

A. Signal processing in the VLC transmitter and receiver

The VLC transmitter divides the input byte stream into payload packets of length L_{pl} bytes. After that, a known preamble sequence of length L_{p} bytes is added at the beginning, and CRC of length L_{crc} bytes is computed (based on the payload bytes) and appended after the payload. Finally, a guard band of L_{g} bytes is also added at the end of the packet to guarantee a minimum separation between consecutive packets (see Fig. 2)

The bytes of each VLC packet are divided into symbols and, after that, each of them are first mapped into points of an M-PAM constellation. Then, samples are pulse-shaped using a Square-Root Raised-Cosine (SRRC) filter. Finally, the signal is sent to the LED driver, which interfaces the output of the USRP to the electric current that is needed to control accordingly the intensity of the light emitted by the LED array.

The optical signal that reaches the sensitive area of the PD generates a current that, after being amplified by the

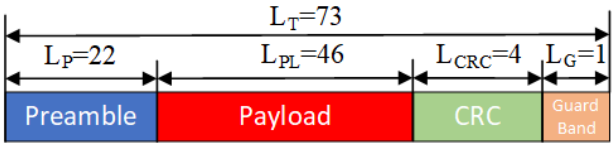


Fig. 2: Overview of the frame structure that was implemented in the software-defined VLC system that relies on the collected ARQ signaling to monitor the status of the indoor environment. The length of the frame fields is in bytes.

TIA, is sampled at a regular sampling time T_{s}. These signal samples go first through a SRRC matched-filter with the same roll-off factor used in the pulse-shaping filter of the VLC transmitter. Afterwards, the energy of the received signal samples is computed to identify whether there is an ongoing transmission or not, due to the bursty data traffic pattern that the implementation of an ARQ scheme generates at the PHY-layer. Therefore, when the energy of the received signal samples is higher than a pre-defined threshold, the most likely position of the preamble is estimated from the received sample sequence. After the position of the preamble is detected, the received signal samples that correspond to the payload and CRC of the VLC frame are demodulated. Finally, the CRC for the received payload bytes is computed and compared with the received CRC bytes, to check whether the received packet is corrupted or not. Note that when implementing an ARQ sheme, the outcome of this CRC verification process is used to issue ACKs (or NACKs) to inform to the transmitter that the current data packet was properly received (or not).

B. Modelling the elements of the VLC channel

An LED is composed of a PN junction which, after the application of a forward voltage, experiences a reduction on the potential barrier of the junction such that the movement of injected minority carriers (*i.e.*, electrons and holes) is induced. This movement results in electron-hole recombination events which emit photons centered on different wavelengths according to the specific semiconductor materials that were used to manufacture the LED chip.

There are different LED technologies that could be used to generate white light with various Correlated Color Temperatures (CCTs) and suitable illuminance (or luxes) for different indoor environments. In this paper, we consider the Phosphor-Converted (PC) white LED technology due to the low-cost, implementation simplicity, and good ability that it has to render colors accurately [9]. As shown in Fig. 3a, a PC-LED consists of a Blue-Chip and a Yellow-phosphor layer, whose interaction generates white light with a CCT that depends on the balance between the blue and yellow light is emitted [3].

In order to model the electrical response of an LED, it is necessary to find an equivalent DC model that attains the form of the well-known Shockley equation, *i.e.*,

$$I_{\mathrm{d}} = I_{\mathrm{s}} \left[\exp\left(\frac{V_{\mathrm{d}}}{N\, V_{\mathrm{t}}} \right) - 1 \right], \tag{1}$$

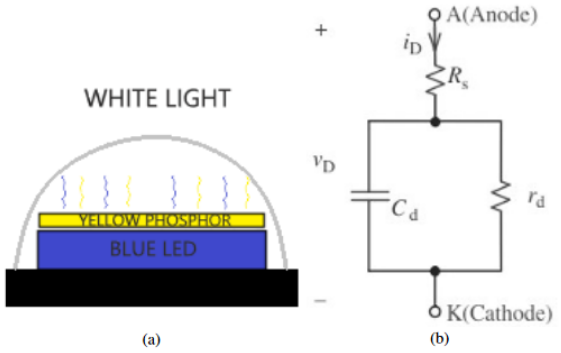


Fig. 3: a) The CCT of the white light emitted by a PC-LED depends on the photons coming from the Blue Chip and Yellow-phosphor layer. b) Equivalent small-signal circuit of a commercial LED used to model its frequency response. Here, R_s, C_d, and r_d are the series resistance, diffusion capacitance, and diffusion resistance, respectively.

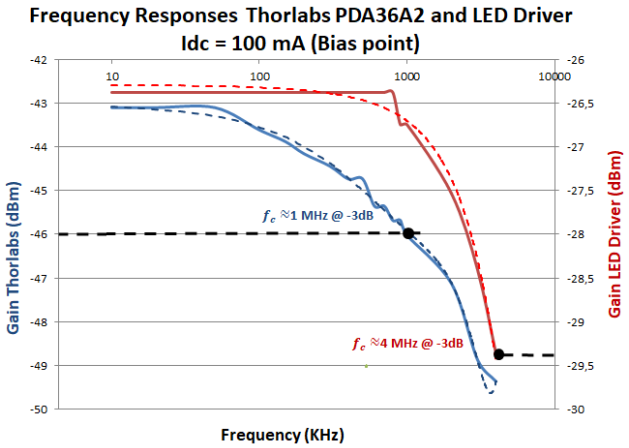


Fig. 4: Frequency response of the LED driver when connected to seven PC-LEDs at 100 mA of DC current. Red lines: Electrical response (i_{led} vs. v_{in}). Blue lines: Optical response (v_{pd} vs. v_{in}). Solid lines: Measured values. Dashed lines: Polynomial approximation. Cutoff frequencies at 3 dB are shown for electrical and optical responses.

where V_{d} is the voltage on the PN junction, N is the parameter known as the emission coefficient, $V_{\mathrm{t}} = 25.8\,\mathrm{mV}$ is the typical thermal voltage at a room temperature of 300 K, and I_{s} is the reverse current of the LED.

The DC constant signal model of an LED consists of a series resistance R_{s} and a diffusion capacitance C_{d}, which take different values according to the forward biasing conditions that are set. In order to determine these values, some of the parameters of the Shockley equation are needed to match the voltage and current pairs that are measured from the LED electrical response. In order to determine the emission coefficient N and the the reverse current I_{s} of a given LED sample under analysis, we use curve fitting methods to minimize the Mean Square Error (MSE) between measured (V_{led},I_{led}) pairs with the ones predicted in (1). For further information about the details of this process, please refer to [12].

On the other hand, in order to study the AC response of the LED at different electrical frequencies, the small-signal model should be considered instead. For this purpose, the diffusion capacitance C_{d} and resistance r_{d} of the LED should be determined, such that the measured frequency response in the electrical domain can accurately match the one approximated by the equivalent circuit in Fig. 3b.

To carry out the practical validation of VLC-based monitoring system using ARQ signaling, an *ad hoc* LED driver (or VLC transmitter front-end) was designed and implemented to interface the voltage signal that comes from the USRP to the current that modulates the intensity of the light emitted by the PC-LEDs. The LED driver was designed to maximize the excursion of the output AC signal at different DC-bias levels, maximizing as much as possible the bandwidth of the input signal (*i.e.*, the cutoff frequency at 3 dB). The LED driver was based on a high-power MOSFET transistor, and included a pre-emphasis circuit that aimed at compensating the strong attenuation that the high-frequency components of the data-carrying signal experience due to the low-pass response of the Yellow-phosphor layer of the PC-LED. Fig. 4 shows the electrical response of the LED driver (red lines), as well as the optical response (blue lines) that includes the low-pass response of the yellow phosphor (measured at the PD output).

III. VLC-BASED MONITORING USING ARQ SIGNALING

When compared to RF communications, VLC systems suffer from more notable line-of-sight blockage when an obstacle (such as a person) is placed between the transmitter and receiver. According to the pace that this person takes when moving across the VLC services area, different ARQ signaling profiles will be generated. In this section, we first give the details of the ARQ scheme that was implemented. After that, we present the geometry of the sensing problem that was considered, which justifies the ARQ statistics that were collected when a person performed different activities.

A. Principles of Stop-and-Wait ARQ

Though different types of ARQ can be found in the literature, the most popular ones are the *Stop-and-Wait* ARQ, the *Go-Back-N* ARQ, and *Selective Repeat* ARQ. In this paper we focus on the Stop-and-Wait scheme, the most basic form of an ARQ protocol, which transmits one data block at a time and waits until a positive (ACK) or negative (NACK) confirmation of the reception of the data packet is signaled to the transmitter.

When the ARQ signaling is received by the transmitter, there are three possible scenarios: (a) The frame was successfully received with no errors (ACK), so the transmitter can send the following data block; (b) The data block that was received had errors (NACK), so the transmitter must re-send the same data block that was hold in a buffer, aiming at having better luck in the new attempt; (c) The data packet is never detected at the intended receiver and, due to that, the (N)ACK is never issued. To solve this problem, the Stop-and-Wait ARQ scheme implements a waiting time period known as *time-out*, giving enough time to receive the (N)ACK signaling under

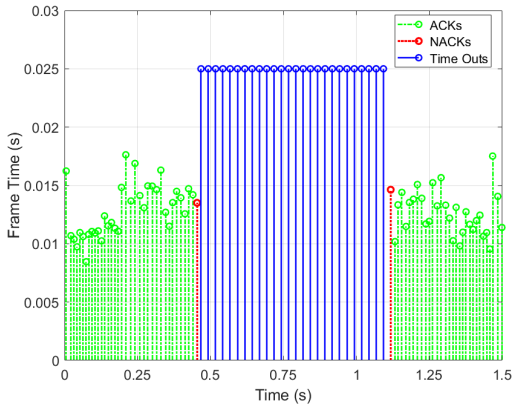


Fig. 5: Transmission time (time-to-ACK) for different data frames when the VLC link implements the Stop-and-Wait ARQ. ACKs are plotted in green dashed-dotted lines, whereas NACKs and time-outs are shown with red dotted and blue continuous lines, respectively.

normal working conditions. If the (N)ACK confirmation is not received before the time-out timer expires, the data block is considered lost, and a re-transmission process is triggered.

We note that in presence of a normal working conditions, where no obstacle is placed between the VLC transmitter and receiver, ACKs are mainly received. However, in presence of a partial VLC link blockage, the SINR of the received signal will be reduced, and NACKs will start to appear according the probability of bit error that the VLC link has. Finally, in case of full-blockage, the preamble of the VLC frame will not be detected and, due to that, no (N)ACKs will be issued by the receiver; in this situation, the time-out timer of the transmitter will expire continuously until the VLC link can be recovered. This effect can be visualized in the central part of Fig. 5, where blue lines show the presence of time-out events during the whole duration of the full-blockage event.

B. Geometry of the Indoor Monitoring based on ARQ

As it has been previously stated, communications based on light suffer from blockage in presence of obstacles between the transmitter and the receiver. This fact causes a deep fading to the transmitted frames making difficult their detection at the receiver side. This initial inconvenient is an advantage for monitoring a point-to-point link. Specifically, it is measured the variations in the time-out of ARQ to determine: i) if it has cut the link a person or two, ii) the direction of walking, and iii) if the person is walking or running when it cuts the link. Toward this regard, the scenario for monitoring consists on an indirect communication from a LED to a Photodetector through a wall (See Fig. 6). The LED and Photodetector are rotated an angle θ_{LED} and θ_{PD} respectively. They are separated from the wall a distance of y, and between them x m. The LED has a exposition angle ψ that focus the light from the LED to the wall in these angular region. On the contrary, the reflection of the light from the wall to the Photodetector is diffuse. That means that the angle in which the light is reflected from the wall does not match with its incident one.

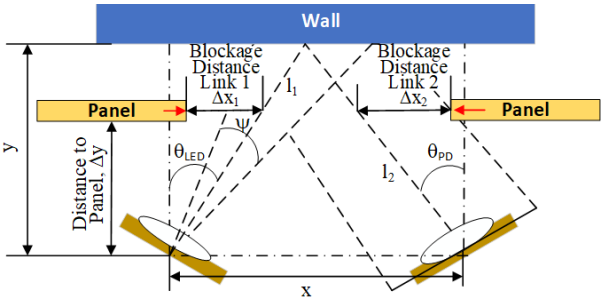


Fig. 6: VLC link geometry for ARQ indoor monitoring. θ_{led} and θ_{pd} represent the rotation angles of the LED and Photodetector respectively, whereas Ψ is the aperture emission angle of the LED.

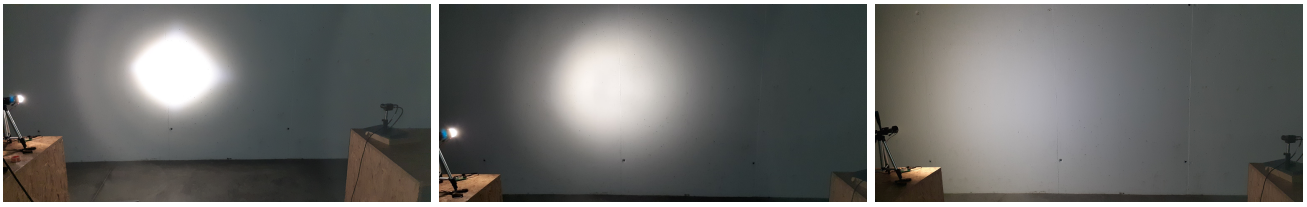
As a result, the size of the second link (i.e. from the wall to the Photodetector) it is expected to be much larger than the first one (i.e. from the LED to the wall). To determine the blocking region of these two links, denoted as Δx_{k}, we have moved a panel from outside of the two links to their center until the VLC receiver does not detect any transmitted data. Thus, if the time-out duration is denoted as T_{out} and the average speed in which it is cut the k-th link is v_{k}, then the expected number of time-outs of the k-th, $N_{\mathrm{T_{out},k}}$, link will be: $N_{\mathrm{T_{out},k}}$=$\Delta x_{\mathrm{k}}/(T_{\mathrm{out}} \cdot v_{\mathrm{k}})$ being the index k equal to one, i.e. $k = 1$, for the link between the LED and the wall, and two, i.e., $k = 2$, for the link between the wall and the Photodetector. Next, it is possible to define the ratio between the number of time-outs of both links, denoted as ζ, as $\zeta = N_{\mathrm{T_{out},2}}/N_{\mathrm{T_{out},1}}$.

This relationship will indicate if its possible to detect the direction, number of persons that cut a link or if the person moves low or fast when cuts the link for a particular setting of the scenario (See Fig. 6). If the value of ζ is close to one, then the time-outs of both links will be quite similar. In this case, it will not be possible to infer any conclusions from them. On the contrary, it will permit us to monitor the link by means of VLC technology. However, the value of ζ depends on several parameters such as the distance of the panel to the LED and Photodetector, the cutting speed in both link (i.e. it could be different), the exposition angle of the LED, and if the Photodetector use lens or not to name a few of them. As a result, it has been considered the number of time-outs as a random variable, which can be characterized by a probability density function. Then, the conditional probabilities of the number of time-outs, $N_{\mathrm{T_{out},k}}$, will be equated as:

$$p(N_{\mathrm{T_{out}}}|N_{\mathrm{T_{out},k}}) = f(N_{\mathrm{T_{out},k}}, \mathbf{\Phi}) \tag{2}$$

being $\mathbf{\Phi}$, the parameters that define the number of time-outs of the link k-th. Then, if $d_{N_{T_{out}}}$, represents the decision frontier of the number of time-outs of the two links, then the error probability in choosing the link wall-Photodetector (link 2) instead of the link LED-wall (link 1) is:

$$p(l_2|N_{\mathrm{T_{out}}} = N_{\mathrm{T_{out},1}}) = \int_{d_{N_{\mathrm{T_{out}}}}}^{\infty} p(N_{\mathrm{T_{out}}}|N_{\mathrm{T_{out},1}}) dN_{T_{out}} \tag{3}$$



(a) Illuminated area of the LED over the wall for an exposition angle of $\psi = 12$ deg.

(b) Illuminated area of the LED over the wall for an exposition angle of $\psi = 50$ deg.

(c) Illuminated area of the LED over the wall for an exposition angle of $\psi = 120$ deg.

Fig. 7: Illuminated area of the LED over the wall when its exposition angles are : $\psi = \{12,50,120\}$ deg.

where in (3) has been assumed that the number of time-outs of link wall-Photodetector is much larger than the corresponding ones for the link LED-wall. Similarly the error probability for choosing the link 1 instead of the link 2 will be:

$$p(l_1|N_{\mathrm{T_{out}}} = N_{\mathrm{T_{out},2}}) = \int_0^{d_{N_{\mathrm{T_{out}}}}} p(N_{\mathrm{T_{out}}}|N_{\mathrm{T_{out},2}})dN_{\mathrm{T_{out}}} \tag{4}$$

Finally, after introducing the VLC based monitoring systems, the next section details the obtained experimental results.

IV. Experimental Results

This section presents the practical results that have been obtained. Initially, it is explained the settings of the experimental tests and next the performance of the obtained practical results.

A. Setting

Table I summarizes space and temporal metrics using in the VLC system for indoor monitoring. The rotation angle of the LED and Photodetector are $\theta_{\mathrm{LD-wall}}$=60 deg. and $\theta_{\mathrm{wall-PD}}$=45 deg. Regarding the exposition angle of the LED, Fig. 7 shows the illuminated are at the wall when the exposition angle is $\psi \in$ [12,50,120] deg. From all of them we have considered the exposition angle of the LED is ψ=50 deg. since for this configuration the blockage distances from the center of the LED-wall and wall-PD links are quite different. Toward this regard, Table II shows such blockage distance when the panel is separated at $\Delta y \in [28.5,38.5,48.5,58.5]$ cm from the LED and Photodetector (See section III-B). Note that the larger the separation distance from the panel to the LED/Photodetector, Δy, the greater the blockage distance, denoted as Δx. Furthermore the blockage distance of the link between the wall to Photodetector is much bigger that the corresponding one for the link between the LED and wall. As a result, if the velocity in which it is cut the two links does not vary too much, it will be possible to infer valid conclusions about the implemented activities from the number of ARQ's time-outs and/or NACKs.

Finally, after introducing the settings of the VLC indoor sensing communication system, the following section shows the main performance results in activity monitoring from ARQ patterns.

TABLE I: Parameters of the VLC experimentation scenario and numerology of the single-carrier M-PAM frame.

Symbol	VLC Link Parameter	Value	Unit
$d_{\text{led-wall}}$	Distance LED to wall	1	m
$d_{\text{wall-pd}}$	Distance wall to Photodetector	1	m
$d_{\text{led-pd}}$	Distance between LED and Photodetector	2.5	m
f_s	USPR sampling rate	1×10^6	Sa/s
T_{F}	Frame durationn	0.576	ms
T_{out}	Time-out duration	25	ms

TABLE II: Blockage distance to the centers of LED-wall and wall-PD links, including their ratio for different separations. Distances are in centimeters (cm). The aperture angle of the LED emission is $\psi = 50$ deg.

Distance Panel, Δy	Blockage Distance Link LED-wall, Δx_1	Blockage Distance Link wall-PD, Δx_2	Ratio ζ
28.5	6.2	15.5	2.5
38.5	9	24.5	2.72
48.5	10.5	33	3.14
58.5	11.5	35	3.04

B. Performance analysis

This section shows the results in activity monitoring from ARQ signaling. Specifically, it has been monitoring the following use cases from ARQ pattern: i) a person cuts the links between the LED-wall and wall-PD walking, ii) a person cuts the two links running, iii) a person with crutches cuts the links walking, and iv) two people cuts the two links walking. Towards this regard, Fig. 8 shows their corresponding time-to-ACK. From there it is possible to distinguish the four ARQ use cases. Specifically, if a person walks then the duration of his/her time-to-ACK augments respect its running case (See Fig. 8a and 8b). For the cases of a person with crutches and two persons walking (See Fig. 8c and 8d), the time-to-ACK increases notably respect to the other two cases. However, the pattern of their time-to-ACK is quite different since the speed of the person with crushes trend to be much more irregular than in the case of two persons. Next, Fig. 9 shows the CDF of the number of time-outs, N_{to}, and NACKs, N_{nacks}, when a person cuts the links LED-wall and wall-PD walking. From there it is possible to obtain several conclusions. Firstly, the number of time-outs of the link between the LED and wall is lower than the corresponding one from the link between the wall and Photodetector. Nevertheless, if it is measured the number of NACKs, it happens just the opposite. The reason of that is because the blocking region of the first link is much lower than the second one (See Table II). Consequently, it is expected that the dominant blockage effect for the first link

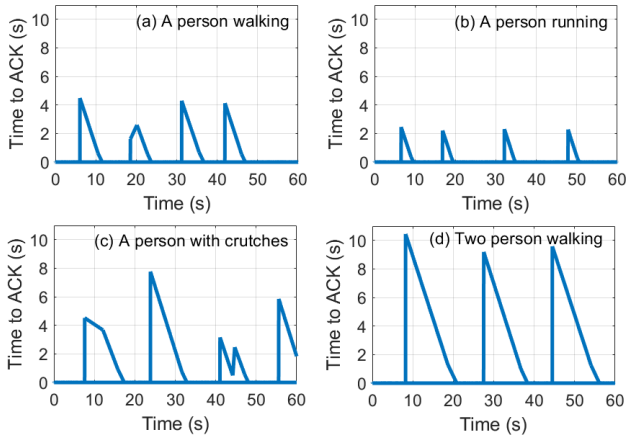


Fig. 8: Time-to-ACK when the links between the LED-wall and wall-PD are cut by (a) a person walking, (b) running, (c) a person walking with crutches, and (d) two person walking

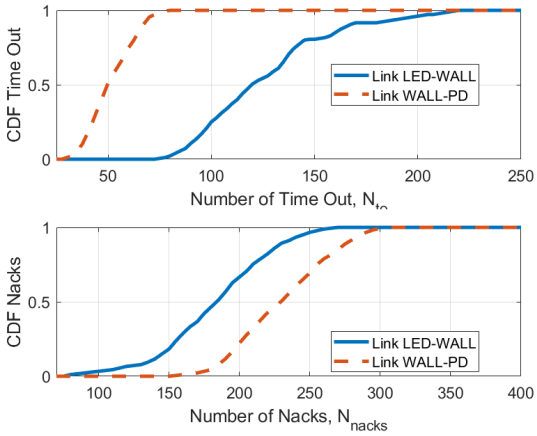


Fig. 9: CDF of the time-outs and NACKs for the link LED-wall and wall-Photodetector

will be *full-blockage*, whereas in the second one, the dominant one will be the *partial-blockage*. In practical terms, it means that in the *full-blockage* case the VLC receiver is not able to detect the preamble sequence. Then, the transmitter does not obtain any feedback from the VLC receiver and time-out signaling is activated. On the contrary, in the *partial-blockage*, the receiver is able to detect the preamble. However, the attenuation is so high that some received symbols may be incorrectly demapped. In this situation the receiver activates the NACK procedure to inform the VLC transmitter that the frame has been incorrectly received. Note that in both cases the frames are re-transmitted but the concept for which they are re-transmitted is different. By doing so, it can be used the signalling of ARQ to determine the monitored activity's direction.

Another interesting result is that the distance between the CDFs of the number of time-outs for the two links is much larger than their corresponding CDF for the number of NACKs. So, it means that there will be less error probability in classifying the activity when it is measured the number of time-outs instead of the number of NACKs.

V. Conclusions

This paper studied the effect that different people's activities generates on the ARQ signaling of an ongoing VLC transmission. The geometry of the scenario consists on an indirect communication between a LED and Photodetector through a wall. As a result, the dominant blockage effect that experiments both links is different. In the first link the dominant blockage is the *full-blockage* whereas in the second one the most important one is the *partial-blockage*. Thus, the concept for which the frames are re-transmitted is different. By doing so, it is possible to detect if a person or two cuts the links, if they walk or run, if they bring or not crutches and the link that has been cut. The obtained results pave the way for deriving future machine learning algorithms that improve the classification of the different activities.

References

[1] W. Jiang, B. Han, M. Habibi, and H. Schotten, "The road towards 6G: A comprehensive survey," *IEEE Open J. Commun. Society*, vol. 2, pp. 334–366, Feb. 2021.

[2] D. Karunatilaka, F. Zafar, V. Kalavally, and R. Parthiban, "LED based indoor visible light communications: State of the Art," *IEEE Commun. Surv. & Tut.*, vol. 17, no. 3, pp. 1649–1678, 3Q 2015.

[3] A. Dowhuszko, M. Ilter, and J. Hämäläinen, "Visible light communication system in presence of indirect lighting and illumination constraints," in *Proc. IEEE Int. Conf. Commun.*, June 2020, pp. 1–6.

[4] ——, "Visible light communications for indoor monitoring (VLADIMIR)," *Public deliverable of ATTRACT final conference*, pp. 1–5, Sept. 2020, [Online]. Available: https://attract-eu.com/wp-content/uploads/2019/05/VLADIMIR.pdf.

[5] F. Zafari, A. Gkelias, and K. Leung, "A survey of indoor localization systems and technologies," *IEEE Commun. Surv. Tut.*, vol. 21, no. 3, pp. 2568–2599, Apr. 2019.

[6] S. Yousefi, H. Narui, S. Dayal, S. Ermon, and S. Valaee, "A survey on behavior recognition using WiFi channel state information," *IEEE Commun. Mag.*, vol. 55, no. 10, pp. 98–104, Oct. 2017.

[7] R. Cam and C. Leung, "Throughput analysis of some ARQ protocols in the presence of feedback errors," *IEEE Trans. Commun.*, vol. 45, no. 1, pp. 35–44, Jan. 1997.

[8] S. Lindner, J. D. Kroening, P. N. Tran, C. Petersen, and A. Timm-Giel, "Analytic study of packet delay from 4g and 5g system arqs using signal flow graphs," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–5.

[9] A. Dowhuszko, M. Ilter, P. Pinho, R. Wichman, and J. Hämäläinen, "Effect of the color temperature of LED lighting on the sensing ability of visible light communications," in *Proc. IEEE Int. Conf. Commun. Workshops*, June 2021, pp. 1–6.

[10] Lumileds, "LUXEON Rebel — General purpose white," Apr. 2016, DS64 LUXEON Rebel General Purpose Product Datasheet, [Online]. Available: https://www.lumileds.com/wp-content/uploads/files/DS64.pdf.

[11] Thorlabs, "PDA100A2 Si switchable gain detector – User guide," May 2019, [Online]. Available: https://www.thorlabs.com/drawings/e10b5a31b41beb6f-17B4555E-A230-2E4C-BB120B00C3A97F91/PDA100A2-Manual.pdf.

[12] E. F. Schubert, *Light-Emitting Diodes*, 2nd ed. Cambridge University Press, 2006.

Appendix B

GNU Radio software

For a better understanding of this MSc Thesis. It is possible to find attached on the following pages the code designed in Python for the correct implementation of the system in GNU Radio.

```
"""
    Tx
    Transmitter Block
"""

import zlib
import math
import numpy as np
from gnuradio import gr
import time
import threading
import json
from scipy.io import savemat
from datetime import datetime

class blk(gr.basic_block):
    def __init__(self, packet_length = 96, windows = 2):
        gr.basic_block.__init__(
            self,
            name='Create CRC',
            in_sig=[np.uint8, np.uint8],
            out_sig=[np.uint8]
        )
        self.packet_len = packet_length - 1
        self.crc_len = 4

        self.tx_count = 0
        self.frame_count = 1
        self.timeout_count = 0
        self.noindex_count = 0

        self.ack_required = 0
        self.init_time = time.time()
        self.tx_time = self.init_time

        self.configure_flag = 0
        self.configure_count = 0
        self.set_output_multiple(self.packet_len + self.crc_len)

        #Go And Back
        self.current_windows = 0
        self.max_windows = windows
        self.rtt_w1 = 0.025
        self.rtt = (self.max_windows - 2) * 0.006 + self.rtt_w1
        self.max_packets = 10000

        self.ack_recv = 0
        self.window_time = np.zeros((self.max_windows, self.
            max_packets))

    def forecast(self, noutput_items, ninput_items_required):
        ninput_items_required[0] = self.packet_len
        ninput_items_required[1] = 0


    def general_work(self, input_items, output_items):
        data_len = len(input_items[0])
        ack_len = len(input_items[1])

        nowTime = time.time()
        total_time = nowTime - self.init_time
        frame_time = nowTime - self.tx_time
```

```
        if self.configure_flag == 1:
            self.init_time = time.time()
            self.configure_flag = 2
        elif self.configure_flag == 3:
            print("Transmission is completed")
            self.configure_flag = -1
            return 0
        elif self.configure_flag == -1:
            return 0


        if self.ack_required == 1 and ack_len > 0:
            self.analize_acks(input_items[1])
            self.consume(1, ack_len)
            return 0

        if self.ack_required == 1 and frame_time >= self.rtt:
            self.timeout_count = self.timeout_count + 1
            self.current_windows = 0
            self.ack_required = 0
            self.ack_recv = 0
        elif self.ack_required == 1:
            return 0

        #Send new packet
        if self.current_windows == self.max_windows:
            self.ack_required = 1
            return 0

        w_data = input_items[0][:self.packet_len]
        w_data[0:2] = np.array([self.frame_count + self.
            current_windows], dtype=np.uint16).view("uint8")
        output_items[0][:self.packet_len] = w_data
        output_items[0][self.packet_len:self.packet_len + self.
            crc_len] = self.get_crc(w_data)

        self.tx_time = time.time()
        self.tx_count = self.tx_count + 1
        self.current_windows = self.current_windows + 1
        return self.packet_len + self.crc_len + 1


    def analize_acks(self, input_ack):
        for idx in range(0, len(input_ack), 2):
            ack = np.array(input_ack[idx:idx + 2], dtype=np.uint8).
                view("uint16")[0]
            if ack == self.max_packets:
                self.configure_flag = 3

            if ack == self.frame_count:
                self.sum_acks(ack)
                self.frame_count = self.frame_count + 1
                self.consume(0, self.packet_len)
            elif ack > 0:
                self.noindex_count = self.noindex_count + 1
            else:
                self.sum_acks(0)

    def sum_acks(self, ack):
        self.required_w1 = 0
        self.ack_recv = self.ack_recv + 1
        if self.ack_recv == self.max_windows:
            self.ack_recv = 0
            self.current_windows = 0
            self.ack_required = 0
```

```
    def get_crc(self, input):
        crc_int32 = zlib.crc32(input)
        crc_np = np.array([crc_int32], dtype="int32")
        return crc_np.view("uint8")
```

```
"""
    Tx
    Preamble Block
"""

import numpy as np
from gnuradio import gr
import pmt


class blk(gr.basic_block):

    def __init__(self, pream = np.zeros(176), payload_length = 0):
        gr.basic_block.__init__(
            self,
            name='Preamble',
            in_sig=[np.uint8],
            out_sig=[np.uint8]
        )
        self.pream = pream
        self.pream_len = len(pream)
        self.payload_len = payload_length
        self.output_len = self.payload_len + self.pream_len
        self.set_output_multiple(self.output_len)


    def forecast(self, noutput_items, ninput_items_required):
        ninput_items_required[0] = self.payload_len


    def general_work(self, input_items, output_items):
        output_items[0][:self.pream_len] = self.pream
        output_items[0][self.pream_len:self.output_len] = input_items
            [0][:self.payload_len]
        self.consume(0, self.payload_len)
        return self.output_len
```

```
"""
    Tx
    Modulator M-PAM Block
"""

import numpy as np
from gnuradio import gr
import math

class blk(gr.basic_block):
    def __init__(self, m_pam = 2, preamble_length = 176,
        payload_length = 800):
        gr.basic_block.__init__(
            self,
            name='Modulator M-PAM',
            in_sig=[np.uint8],
            out_sig=[np.complex64]
        )
        self.m_pam = m_pam
        self.bits_per_sym = int(self.m_pam / 2)
        self.preamble_len = preamble_length
        self.payload_len = payload_length
        self.sym_len = self.payload_len / self.bits_per_sym
        self.set_output_multiple(self.preamble_len + self.sym_len)

    def forecast(self, noutput_items, ninput_items_required):
        ninput_items_required[0] = self.preamble_len + self.
            payload_len

    def general_work(self, input_items, output_items):
        pream = input_items[0][:self.preamble_len]
        pream_syms = np.where(pream > 0, 1, -1)

        if self.m_pam == 2:
            bits = np.array(input_items[0][self.preamble_len:self.
                preamble_len + self.payload_len])
            payload_syms = np.where(bits > 0, 1, -1)
            output_items[0][0:self.preamble_len] = pream_syms
            output_items[0][self.preamble_len:self.preamble_len +
                self.sym_len] = payload_syms
            self.consume(0, self.preamble_len + self.payload_len)
            return self.preamble_len + self.sym_len
        elif self.m_pam == 4:
            bits = np.array(input_items[0][self.preamble_len:self.
                preamble_len + self.payload_len])
            b = np.resize(bits, (self.sym_len, self.bits_per_sym))
            idx00 = np.argwhere((b[:, 0] == 0) & (b[:, 1] == 0))
            idx01 = np.argwhere((b[:, 0] == 0) & (b[:, 1] == 1))
            idx10 = np.argwhere((b[:, 0] == 1) & (b[:, 1] == 0))
            idx11 = np.argwhere((b[:, 0] == 1) & (b[:, 1] == 1))

            idx00 = np.squeeze(idx00)
            idx01 = np.squeeze(idx01)
            idx10 = np.squeeze(idx10)
            idx11 = np.squeeze(idx11)

            payload_syms = np.zeros(self.sym_len, dtype=np.complex64)
            payload_syms[idx00] = 3 / math.sqrt(5)
            payload_syms[idx01] = 1 / math.sqrt(5)
            payload_syms[idx10] = -3 / math.sqrt(5)
            payload_syms[idx11] = -1 / math.sqrt(5)

            output_items[0][0:self.preamble_len] = pream_syms
```

```
            output_items[0][self.preamble_len:self.preamble_len +
                self.sym_len] = payload_syms
            self.consume(0, self.preamble_len + self.payload_len)
            return self.preamble_len + self.sym_len
        return 0
```

```
"""
    Tx
    Pulse-Shape Filter Block
"""

import numpy as np
from gnuradio import gr
import math
from datetime import datetime

class blk(gr.basic_block):
    def __init__(self, root_cosine_filter = np.zeros(200),
        preamble_length = 0, payload_length = 0):
        gr.basic_block.__init__(
            self,
            name='Shape filter',
            in_sig=[np.complex64],
            out_sig=[np.complex64]
        )
        self.oversampling_factor = 5
        self.root_cosine = root_cosine_filter
        self.packet_len = preamble_length + payload_length
        self.set_output_multiple(self.packet_len * self.
            oversampling_factor + self.root_cosine.size * 2)
        self.count = 0


    def forecast(self, noutput_items, ninput_items_required):
        ninput_items_required[0] = self.packet_len


    def general_work(self, input_items, output_items):
        real = np.array(input_items[0][:self.packet_len]).real
        real_ovsam = self.oversampling_input(real)
        real_out = np.convolve(self.root_cosine, real_ovsam)
        len_out = len(real_out)
        output_items[0][:len_out] = real_out
        self.consume(0, self.packet_len)
        return len_out


    def oversampling_input(self, input_p):
        input_sup = np.zeros(input_p.size * self.oversampling_factor)
        input_sup[::self.oversampling_factor] = input_p
        return input_sup
```

```
"""
    Rx
    Preamble detection Block
"""

import zlib
import numpy as np
from gnuradio import gr
import math
import warnings
import pmt


class blk(gr.basic_block):
    def __init__(self, pream = np.zeros(176), root_cosine_filter = np
        .zeros(31), payload_length = 400, pw_threshold = 0):
        gr.basic_block.__init__(
            self,
            name='Preamble detection',
            in_sig=[np.complex64],
            out_sig=[np.complex64]
        )
        warnings.simplefilter(action='ignore', category=FutureWarning
            )
        self.crc_len = 4
        self.pw_threshold = pw_threshold
        self.set_output_multiple(payload_length)
        self.pream = (np.array(pream) * 2) - 1
        self.hintfil = self.get_intertpolate_filter_27()
        self.oversampling_factor = 5
        self.oversampling_total = self.oversampling_factor * self.
            oversampling_factor2
        self.Lh = root_cosine_filter.size
        self.Npup = self.pream.size * self.oversampling_factor
        self.N = self.Npup + self.Lh - 1 + self.Lh - 1

        self.payload_len = payload_length
        self.payload_oversamp = self.payload_len * self.
            oversampling_factor
        self.frame_len = self.Npup + self.payload_oversamp
        self.process_pream(root_cosine_filter)
        self.pream_pw = np.sum(self.preamble_m * self.preamble_m)

        self.pream_detected = 0
        self.py_lfilter = 0
        self.scipy_signal = 0

        self.buffer_atn = np.zeros(50)
        self.atn = 0
        self.count_atn = 0
        self.snr = 0

        #Go Back N
        self.continue_frame = 0
        self.windows = 1

        self.load_matlabLib()


    def forecast(self, noutput_items, ninput_items_required):
        ninput_items_required[0] = self.frame_len


    def general_work(self, input_items, output_items):
```

```
        real = np.array(input_items[0]).real

        if self.pream_detected == 1:
            payload = self.detect_payload(real)
            output_items[0][:self.payload_len] = payload
            self.consume(0, self.frame_len)
            return self.payload_len

        res = self.measure_power(real)
        self.consume(0, res)
        return 0


    def measure_power(self, input_real):
        conv_real = np.convolve(self.matched_filter, input_real)
        pw = np.sum(np.power(conv_real, 2))
        if pw > self.pw_threshold:
            pos = self.correlate(conv_real)
            return pos
        return input_real.size - self.frame_len


    def correlate(self, received):
        if self.continue_frame == 1:
            r_pos = self.detect_continue_frame(received)
            if r_pos != 0:
                return r_pos
        valcorr = self.scipy_signal.fftconvolve(received, self.
            preamble_m[::-1], mode='valid')
        argmax = valcorr.argmax()
        detect = valcorr[valcorr > self.pw_threshold]
        if len(detect) == 0:
            return received.size - self.frame_len
        self.pream_detected = 1
        self.continue_frame = 1
        return int(argmax)


    def detect_continue_frame(self, received):
        corr = np.sum(received[self.Lh - 1:self.N + self.Lh - 1] *
            self.preamble_m)
        if corr > self.pw_threshold:
            self.windows = self.windows + 1
            self.pream_detected = 1
            return self.Lh - 1
        self.windows = 1
        self.continue_frame = 0
        return 0

    def detect_payload(self, input_real):
        self.pream_detected = 0
        frame = np.convolve(self.matched_filter, input_real)
        y2c = frame[self.Lh - 1:]
        z = self.oversampling_rx(y2c)
        y3c = self.matlab_filter(self.hintfil, 1, z)
        if self.windows == 1:
            pot = np.zeros(self.oversampling_total + self.hintfil.
                size)
            for nq in range(self.oversampling_total + self.hintfil.
                size):
                y3b = y3c[nq:nq + self.Npup * self.
                    oversampling_factor2]
                y3d = y3b[:self.pream.size * self.oversampling_total:
                    self.oversampling_total]
                pot[nq] = np.sum(y3d * self.pream)
```

```
                self.pot_max = pot.argmax()

        nmax = self.pot_max
        y2d2 = y3c[nmax:nmax + self.Npup * self.oversampling_factor2]
        y2d = y2d2[0::self.oversampling_total]
        attn = np.sum(y2d * self.pream) / self.pream.size
        self.atn = self.set_attenuation(attn)
        preamblesest = y2d / self.atn
        self.calc_noise(y2d)

        rx_prem = np.where(preamblesest > 0, 1, -1)
        npream = np.sum(rx_prem == self.pream)

        total_len = ((self.pream.size + self.payload_len) * self.
            oversampling_factor) * self.oversampling_factor2
        y3b = y3c[nmax:nmax + total_len] / self.atn
        y3d = y3b[0::self.oversampling_total]
        header = y3d[self.pream.size:self.pream.size]
        payload = y3d[self.pream.size:self.pream.size + self.
            payload_len]
        return payload


    def calc_noise(self, preamble):
        noise = abs(preamble) - self.atn
        var2 = np.sum(noise ** 2) / self.pream.size
        self.snr = 10 * math.log10(self.atn / (4 * var2))
        return self.snr

    def set_attenuation(self, attn):
        if self.count_atn == 49:
            self.buffer_atn[:self.count_atn - 1] = self.buffer_atn[1:
                self.count_atn]
            self.buffer_atn[self.count_atn] = attn
            mean = np.mean(self.buffer_atn)
            return mean
        self.buffer_atn[self.count_atn] = attn
        self.count_atn += 1
        mean = np.sum(self.buffer_atn) / self.count_atn
        return mean



    def process_pream(self, root_cosine_filter):
        pream_sup = self.oversampling_preamble()
        eq_filter = np.convolve(root_cosine_filter,
            root_cosine_filter)
        self.matched_filter = root_cosine_filter
        self.preamble_m = np.convolve(eq_filter, pream_sup)

    def oversampling_preamble(self):
        pream_sup = np.zeros(self.pream.size * self.
            oversampling_factor)
        pream_sup[::self.oversampling_factor] = self.pream
        return pream_sup

    def oversampling_rx(self, rx_data):
        rx_sup = np.zeros(len(rx_data) * self.oversampling_factor2)
        rx_sup[::self.oversampling_factor2] = rx_data
        return rx_sup

    def matlab_filter(self, a, b, c):
        return self.py_lfilter(a, b, c)

    def load_matlabLib(self):
```

```
        from scipy import signal
        from scipy.signal import lfilter
        self.scipy_signal = signal
        self.py_lfilter = lfilter

    def get_intertpolate_filter_27(self):
        self.oversampling_factor2 = 7
        hintfil = np.array([-0.0249, -0.0466, -0.0622, -0.0683,
            -0.0619, -0.0401, -0.0000, 0.1524, 0.3170, 0.4845, 0.6457,
             0.7914, \
        0.9125, 1.0000, 0.9125, 0.7914, 0.6457, 0.4845, 0.3170,
            0.1524, -0.0000, -0.0401, -0.0619, -0.0683, -0.0622,
            -0.0466, -0.0249])
        return hintfil
```

```
"""
    Rx
    Demodulator M-PAM Block
"""

import zlib
import numpy as np
from gnuradio import gr
import math
import warnings


class blk(gr.basic_block):
    def __init__(self, m_pam = 4, payload_length = 400):
        gr.basic_block.__init__(
            self,
            name='Demodulator M-PAM',
            in_sig=[np.complex64],
            out_sig=[np.uint8]
        )
        self.m_pam = m_pam
        self.bits_per_sym = int(self.m_pam / 2)
        self.sym_len = int(payload_length)
        self.byte_len = int((payload_length * self.bits_per_sym) / 8)

    def forecast(self, noutput_items, ninput_items_required):
        ninput_items_required[0] = self.sym_len


    def general_work(self, input_items, output_items):
        symbols = np.array(input_items[0][0:self.sym_len])
        if self.m_pam == 2:
            pam2_bytes = np.where(symbols > 0, 1, 0)
            output_items[0][:self.byte_len] = pam2_bytes
            self.consume(0, self.sym_len)
            return self.byte_len
        elif self.m_pam == 4:
            pam4_bytes = self.demodule_4PAM(symbols, 1e-4)
            output_items[0][:self.byte_len] = pam4_bytes
            self.consume(0, self.sym_len)
            return self.byte_len
        return 0


    def demodule_4PAM(self, payload, pn):
        result = np.zeros(self.sym_len*self.bits_per_sym, dtype=int)
        symbols = np.array([-3, -1, 1, 3]) / math.sqrt(5)
        result[::2] = (-np.sign(payload)) > 0
        r1 = -np.power(payload - symbols[0], 2)
        r2 = -np.power(payload - symbols[3], 2)
        res1 = np.exp(-np.power(payload - symbols[0], 2) / pn) + np.
            exp(-np.power(payload - symbols[3], 2) / pn)
        res2 = np.exp(-np.power(payload - symbols[1], 2) / pn) + np.
            exp(-np.power(payload - symbols[2], 2) / pn)
        result[1::2] = np.where(res2 > 0, 1, 0)
        res_pack = np.packbits(result)
        return res_pack
```

```
"""
    Rx
    Check CRC Block
"""

import zlib
import struct
import math
import numpy as np
from gnuradio import gr
from gnuradio import digital
import time


class blk(gr.basic_block):

    def __init__(self, frame_length = 96):
        gr.basic_block.__init__(
            self,
            name='Check CRC',
            in_sig=[np.uint8],
            out_sig=[np.uint8]
        )
        self.packet_len = frame_length
        self.crc_len = 4
        self.wrongs = 0
        self.corrects = 0

    def forecast(self, noutput_items, ninput_items_required):
        ninput_items_required[0] = self.packet_len + self.crc_len


    def general_work(self, input_items, output_items):
        res = self.check_crc(input_items[0][:self.packet_len + self.
            crc_len])
        num = np.array(input_items[0][:2], dtype=np.uint8).view("
            uint16")[0]

        if res == 0:
            self.corrects = self.corrects + 1
            output_items[0][0:2] = input_items[0][:2]
        else:
            self.wrongs = self.wrongs + 1
            output_items[0][0:2] = [0, 0]

        per = (self.wrongs / float(self.corrects + self.wrongs))
        output_items[1][0] = per
        output_items[2][0] = self.wrongs
        output_items[3][0] = self.corrects + self.wrongs
        self.consume(0, self.packet_len + self.crc_len)
        return 2


    def check_crc(self, input):
        packet = input[:self.crc_len * -1 - 1]
        packet_crc = input[self.packet_len - 1:self.packet_len + self
            .crc_len - 1]
        crc_int32 = zlib.crc32(packet)
        crc_np = np.array([crc_int32], dtype="int32")
        crc_int8 = crc_np.view("uint8")
        res1 = np.sum(packet_crc != crc_int8)
        return res1
```