Universidade de Aveiro Departamento de Electrónica, Telecomunicações e
2014 Informática

Gonçalo Nuno Brás Farias

Sistema de comunicação por Luz Visível de Baixo Débito
Low Data Rate Visible Light Communication System

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e
2014 Informática

Gonçalo Nuno Brás Farias

Sistema de comunicação por Luz Visível de Baixo Débito
Low Data Rate Visible Light Communication System

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Dr. Luís Filipe Mesquita Nero Alves, Professor Assistente do Departamento de Electrónica e Telecomunicações e Informática da Universidade de Aveiro e do Doutor João Paulo Barraca Professor Assistente Convidado do Departamento de Electrónica e Telecomunicações e Informática da Universidade de Aveiro.

Agradecimentos

Aos meus pais, pela oportunidade que me deram, por toda a compreensão e porque sempre estiveram presentes.
À minha irmã, pelo apoio incondicional.
Aos meus avós, por todo o carinho e porque sempre se preocuparam comigo.
À minha madrinha por toda a ajuda no processo de integração numa nova cidade.
À Vanessa, pela paciência e pela companhia.
A todos os amigos com quem partilhei bons momentos durante o meu percurso académico e que o tornaram melhor.
Aos meus orientadores Prof. Luis Nero Alves e Prof. João Paulo Barraca pela ajuda, pela paciência e por todos os conhecimentos transmitidos.
Aos meus colegas no Laboratório de Sistemas e Circuitos Integrados pelo convívio que sempre ajudou a aliviar o stress do trabalho.
A todas as pessoas que não estão aqui referidas e que de uma forma ou de outra estiveram presentes na minha vida ao longo destes anos.

Palavras-chave

Diodos Emissore de Luz, Comunicação por Luz Visível, Smartphone, Comunicações Móveis, Sistemas de Baixo débito

Resumo

Os recentes avanços na tecnologia dos diodos emissores de luz (LED) levaram a que estes conquistassem um lugar muito importante nos sistemas de iluminação. Esta conquista aliada à sua velocidade de comutação levou ao desenvolvimento de sistemas de comunicação por luz visível (VLC), estes incluem-se nos sistemas de comunicação ópticos não guiados. No passado as comunicações ópticas não guiadas restringiam-se ao espectro infravermelho, devido a ineficiência dos LEDs, mas hoje em dia isso está a mudar.

Os sistemas de comunicação por luz visível podem oferecer alternativas viáveis ou complementares aos actuais sistemas de comunicação, devido à sua facilidade de integração em certos meios. Um dos meios em que este tipo de comunicação se pode integrar, e que deu origem a este trabalho, são as comunicações móveis. Os telemóveis antigos possuíam interfaces que permitiam comunicações ópticas como por exemplo IrDA, mas com os avanços tecnológicos estes tornaram-se obsoletos e foram eliminados. Devido a isso os *smartphones* modernos não oferecem qualquer tipo de interface óptica de comunicação. Privilegiam no entanto o uso de câmaras que têm associado um dispositivos de flash baseados em LEDs de alta intensidade. Conseguindo controlar com alguma precisão o flash de um *smartphone* consegue-se implementar um sistema VLC de baixo débito que pode ser usado em sistemas de smart tagging, controlo de remoto de dispositivos electrónicos ou mesmo controlo de acesso a edifícios.

O principal objectivo deste trabalho é o estudo da viabilidade do uso de um *smartphone* como emissor num sistema VLC de baixo débito.

Keywords

Light Emitting Diode, Visible Light Communication, Smartphone, Mobile Communications, Low Data Rate System

Abstract

Recent advances in light emission diodes (LED) technology led them to an important place on lighting systems. This conquer allied to its switching speed permitted the development of new visible light communication systems (VLC), these are included in unguided optical communications. In the past, unguided optical communications were restricted to infrared spectrum due to LEDs inefficiency, but nowadays this is changing.

Visible light communication systems can offer viable or complementary alternatives to the existing communication systems, due to its easy integration in certain environments. One possible integration environment are the mobile communications, and that fact is in the origin of this work. Old mobile phones had interfaces that allowed optical communication, for example IrDA but with the advances of technology these become obsolete and were eliminated. Due to that, modern smartphones do not offer any kind of interface for optical communications. However most of them have one camera that uses a flash device based on high intensity LEDs. Controlling with some precision one smartphone flash allows the implementation of one low data rate VLC systems which can be used for smart tagging, remote control of electronic devices or to control access to buildings.

The main goal of this project is study the viability of the use of one smartphone as emitter in a VLC system.

Contents

List of Figures

List of Tables

Acronyms

ASCII	American Standard Code For Information Interchange
ASK	Amplitude Shift Keying
ATM	Automatic Teller Machines
ADC	Analog-to-Digital-Converter
CES	Consumer Electronics Show
DH-PIM	Dual-Header Pulse Interval Modulation
GPIO	General Purpose Input/Output
GPS	Global Positioning System
IEEE	Institute Of Electrical And Electronic Engineers
IID	Independently and Identically Distributed
IM/DD	Intensity Modulated Direct Detection
IOCTL	Input Output ConTroL
IR	Infra Red
IrDA	Infra Red Data Association
JAR	Java Archive
JNI	Java Native Interface
JVM	Java Virtual Machine
LCD	Liquid Cristal Display
LED	Light Emitting Diode
LOS	Line of Sight
MAC	Medium Access Control
MIMO	Multiple Input Multiple Output

NDK	Native Development Kit
NFC	Near Field Communication
OFDM	Orthogonal Frequency Division Multiplexing
OOK	On-Off Keying
OOK-NRZ	On-Off Keying Non Return to Zero
OOK-RZ	On-Off Keying Return to Zero
OS	Operative System
OW	Optical Wireless
PIM	Pulse Interval Modulation
PPM	Pulse Position Modulation
PSD	Power Spectral Density
PWM	Pulse Width Modulation
RF	Radio Frequency
RFID	Radio-frequency Identification
RGB	Red, Green and Blue
SDK	Software Development Kit
SysFS	Sys File System
VFS	Virtual File System
VLC	Visible Light Communications
VPPM	Variable Pulse Position Modulation
WDM	Wavelength Division Multiplexing
WLED	White Light Emitting Diode
Wi-Fi	Wireless Fidelity

Chapter 1

Introduction

1.1 Motivation for Visible Light Communications (VLC)

With the introduction of mobile technologies in people's life, wireless communications became an important tool either to work or just for fun. Technological advances in the fields of displays, batteries and processing technologies made it possible to build smartphones and tablets that allow the user to access Internet almost everywhere. This kind of devices created an increasing popularity of multimedia services supplied over radio frequency, examples of this services are web browsing and audio/video on demand. This increase on popularity originates an astronomic growth on mobile data traffic. Figure 1.1 shows one prediction on how the global mobile data traffic will behave until 2018, it is expected an compound annual growth rate of 61%.

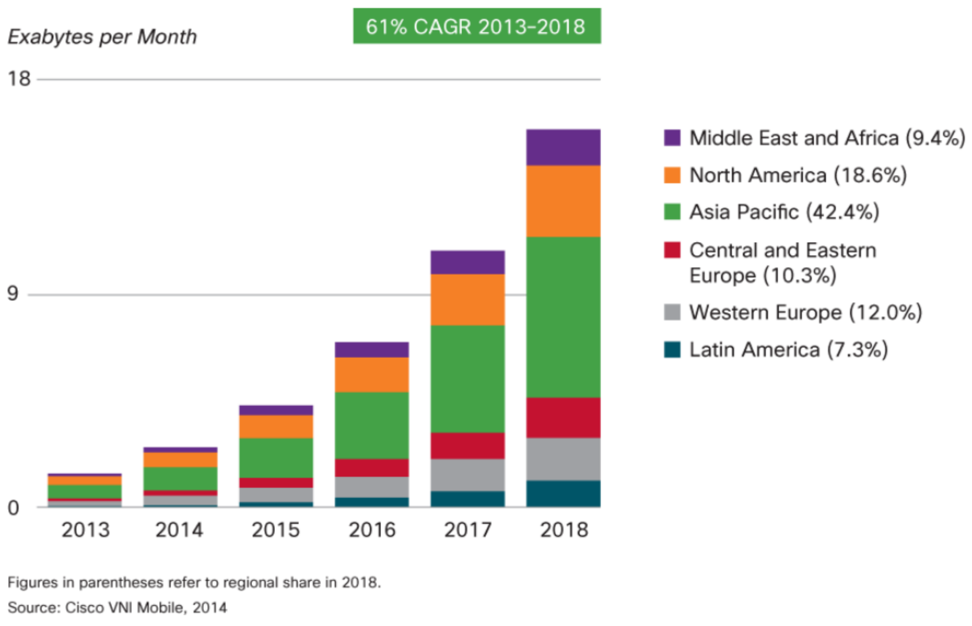


Figure 1.1: Global Mobile Traffic Forecast by Region [1]

Scientific community believes it is only a matter of time until radio frequency spectrum reaches its limits and users start to face long latency problems and line congestion when trying to access the services anteriorly referred, one belief which is supported by the fact of the electromagnetic spectrum is extremely crowded. This fact, known as "spectrum crunch" may cause an increase in the number of dropped calls, slow down data speeds and raise costumer's prices [2]. Figure 1.2 reveals that in a near future the spectral bandwidth available for use will not be sufficient for our needs. This predictions were presented in 2012 and refers to the USA scenario. Nonetheless, this lack of spectral space caused by the increasing number of mobile devices affects all the world.

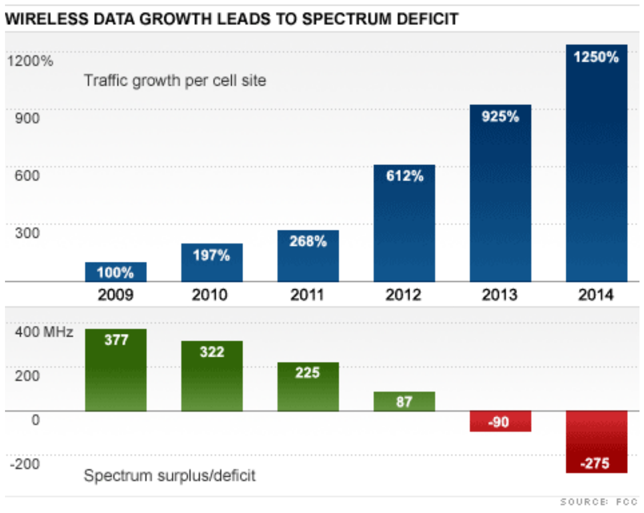


Figure 1.2: Expected Spectrum Deficit [2]

To mitigate that fact cellular systems started to reuse bandwidth but if by one side this technique raised the achievable system capacity by the other side it increased interferences and limited the achievable network throughput. So, finding an radio frequency (RF) orthogonal communication medium to fill the gap between needed and available spectral bandwidth becomes a real need [3]. Despite of the predictions are not encouraging, nowadays we do not reach this spectrum deficit, and we continue to use RF with no problems.
Few years ago researchers realized that white LEDs (WLEDs) could be much more than a lighting device and could be used for wireless communications purposes. The advantages of such technology applications are many. It belongs to the green technologies category when used for lighting purposes, becoming even more environmentally friendly when it supports communication functionality compared to RF alternatives [4].
Light waves at visible and infrared (IR) wavelength are space confined, once they can not penetrate solid materials such as walls. This property allows VLC to perform full bandwidth reuse without interference.
In open environments where high speed communications may be needed, Optical Wireless (OW) revealed to be an sensible solution due to its limited cell size, but today's RF cells do not support so much high capacity users per cell. Multiple high capacity users require multiple cells and originate overlaps and inter-cell interferences. Contrasting with that, OW provides user-sized cells and because of the intrinsically abrupt boundary of these cells inter-cell interference is negligible and carrier reuse is not an issue.
It is expected that in a near future almost all the lighting devices are based on LEDs, this makes VLC a viable option for OW system since LEDs can be used as a wireless communications transmitter [5]. For all the previous mentioned characteristics and for being interference-orthogonal to the cellular RF networks VLC is the best solution to

provide viable or complementary alternatives for RF communications.

1.2 VLC Characteristics

Visible Light Communications use light sources such as LEDs rather than RF transmitters as a transmission source. LEDs can be readily modulated making them suitable for the dual role of illumination and data transmission. For illumination purposes, LEDs generate very little heat and they do not radiate anything but light photons, thus cause much reduced carbon emissions than other illumination technologies. Visible light occupies the range of wavelengths between 380 and 780nm. In Figure 1.3 it is possible to see the position of visible band inside the electromagnetic spectrum.

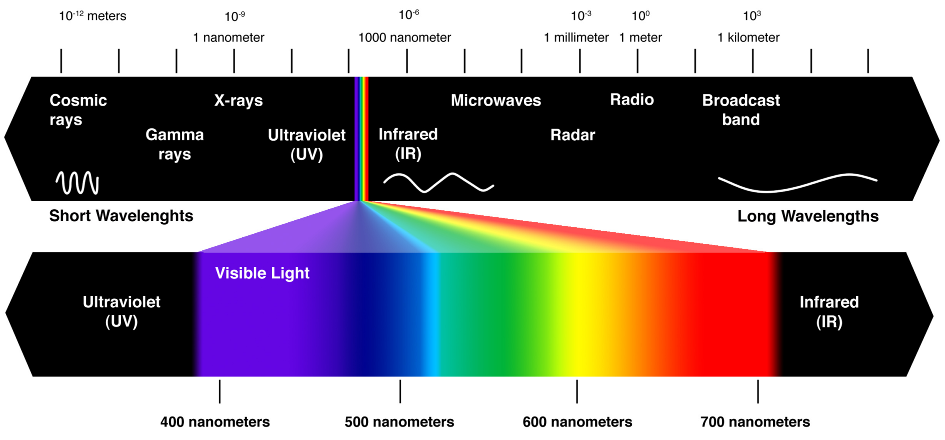


Figure 1.3: Electromagnetic Spectrum [6]

Communications over light can offer many advantages when compared to RF systems, starting by the price of LEDs and photo-detectors, which tend to be cheaper than RF emitters and receivers. Other factor that can reduce the price for VLC is that visible light spectrum is unlicensed and can be used free of charge. Optical communications can easily reuse bandwidth and provide more privacy than RF because light is confined to a certain space and devices out of line of sight could not access data transmitted by the light beam. The bandwidth for this communication technique is virtually unlimited and can achieve high data rates, making it viable for high speed wireless communications. In [7] an experiment with link speeds up tp 3 Gb/s is reported. Since VLC uses visible light beams it is not harmless to human health and does not represent danger for human eyes. As there are already many lighting devices using LED lamps it could be seen as one advantage to VLC, because this lamps could be used as data emitters. All this facts make VLC a suitable option to use in environments where no RF interference is required, such as some medical environments, schools, military facilities and planes [4].
Obviously this technology has some drawbacks once it uses light it become vulnerable to the interference of other light sources. In some VLC applications a line of sight (LOS) between emitter and receiver is required, and in some situations ensuring that can lead to an increase of system complexity and price [4].

1.3 VLC Applications

VLC has many applications, has previously referred. Now a short list af applications and respective explanation will be provided.

Smart Lighting Any light source could be used to provide an VLC access point. The same communications and sensor infrastructure may be used to control lighting. This kind of systems can adjust its brightness according to the natural light they receive, saving money and reducing carbon emissions.

Indoor Positioning Indoor environments do not allow positioning systems such as global positioning system (GPS), so VLC can help in that field. Using multiple LED bulbs and image sensors it is possible to calculate accurate indoor positioning [8].

Hazardous Environments VLC could be used as safe alternative to electromagnetic interferences from RF communications in environments such as mines and petrochemical plants.

Vehicular Networks Nowadays car lights and traffic lights are LED based, this fact can be used to set-up vehicular networks. This networks will be used to exchange information between vehicles or from traffic infrastructures to vehicles [9]. This application might increase drive safeness and reduce traffic jams.

Medical Hospitals and medical installations are generally much more averse to the use of RF technology, mainly due to its lack of security, interference with other electronic equipment by overlapping channels and possible damage to sensitive electronic equipment. Because VLC is fast, safe and secure, and has no RF radiation, medical applications are an obvious choice.

Aviation LEDs are present in aircraft cabins, so VLC could be used to provide in-flight entertainment systems. This entertainment systems could offer audio/video streaming and digital books or magazines, for example.

Underwater Communications Due to strong signal absorption in water, RF use is impractical. Acoustic waves have extremely low bandwidth and disturb marine life. VLC provides a solution for short-range communications.

Toys Lots of toys have built in LEDs, this fact could be used to enable communication and interaction between various toys, using VLC [10].

1.4 Project Motivation

The idea for this project stems from the fact of modern smartphones do not have any kind of optical communication interface. If old mobile phones allowed data transfer using IrDA, new smartphones just allow data exchange over RF. However the presence of high definition cameras, with one flash device associated, is normal in smartphones. This flash is typically based on one high intensity LED, and if we can control this LED we may be able to use it to send data and use one mobile phone as an emitter in a

VLC system.
Previous work on this field revealed that it is possible to make flash light blink at one specific frequency. Experimental data revealed that for today's smartphones the flashlight blinking frequency can be up to 50Hz. However, the frequency is not constant over time because the flashlight is controlled by the operating system, and depending on the process scheduling, the flashlight's light pattern deviates significantly from the intended pattern [10].
With this project it is intended to know if it is really possible to use one smartphone to modulate and send data. For this, various flash control strategies were tested in order to find out its limitations as well as the complexity behind its implementation. During the execution of this work some modulation techniques were also analyzed, because this kind of application requires one modulation technique with one good compromise between data throughput and energy consumption.

1.5 Document Structure

This document has four chapters.
The first one is this chapter, the Introduction. This chapter is divided in five sections:Motivation for Visible Light Communications (VLC), VLC Characteristics, VLC Applications, Project Motivation and Document Structure.

In chapter two, "State of the Art", there are three sections:Some History, with history of LEDs and VLC history;Actual State of VLC Communications and Low Debt VLC Projects.

Chapter three, called "Exploring Smartphone's Flashlight to Transmit Data" is composed by sections: Concept, Flash Control Strategies, Modulation Techniques. There we can find information on how to control flash light and information about modulation techniques that can be used on VLC. In that chapter decisions about chosen modulation technique are justified.

Chapter four, named "Developed Work" and has the analysis of the obtained results.

Last chapter has some brief conclusions and presents some points for future work.

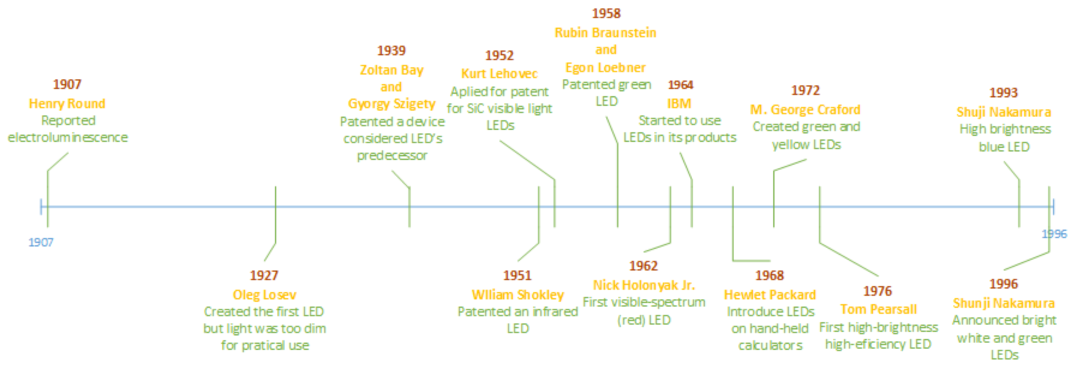
Chapter 2

State of the Art

2.1 Some History

2.1.1 LED History

The history of LED technology dates back to 1907 when Henry Joseph Round noticed that the crystals used in radio receivers emitted light when electricity passed through them. This phenomenon was called electroluminescence. Twenty years after Round's discovery Oleg Losev reported the creation of a LED, but emitted light was too dim for practical use. In 1939 Zoltan Bay and Gyorgy Szigety patented a silicon carbide (SiC) electroluminescent lighting device, considered to be the predecessor of modern LEDs.
The 1950s were very important for LED technology advances, beginning in 1951 with the patent for an infrared LED by William Shockley. In 1952 Kurt Lehovec applied for patent for SiC visible lights LEDs and in 1958 Rubin Braunstein and Egon Loebner patented a green LED.
In figure 2.1 it is possible to see one timeline with important dates and events referred to in this subsection.



1907
Henry Round
Reported electroluminescence
1927
Oleg Losev
Created the first LED but light was too dim for pratical use
1939
Zoltan Bay and Gyorgy Szigety
Patented a device considered LED's predecessor
1951
Wiliam Shokley
Patented an infrared LED
1952
Kurt Lehovec
Aplied for patent for SiC visible light LEDs
1958
Rubin Braunstein and Egon Loebner
Patented green LED
1962
Nick Holonyak Jr.
First visible-spectrum (red) LED
1964
IBM
Started to use LEDs in its products
1968
Hewlet Packard
Introduce LEDs on hand-held calculators
1972
M. George Craford
Created green and yellow LEDs
1976
Tom Pearsall
First high-brightness high-eficiency LED
1993
Shuji Nakamura
High brightness blue LED
1996
Shunji Nakamura
Announced bright white and green LEDs
1907
1996

Figure 2.1: Time line for LED evolution (1907 - 1996)

The next important step was taken by Nick Holonyak Jr. in 1962 with the construction of the first practical visible-spectrum (red) LED. In 1964 IBM used for he firs time LEDs on circuit boards in a early mainframe computer and the year of 1968 was marked by the introduction of LEDs in Hewlett Packard's hand-held calculators. At the same time LEDs also started to appear in digital displays on TVs, radios, watches, and

telephones.
During the 1970s two major events took place, in 1972 M. George Craford created the green and the yellow LED and in 1976 the first high-brightness, high-efficiency LED for optical fiber communications appeared by the hands of Tom Pearsall.
An LED revolution occurred in 1993, Shuji Nakamura introduced the blue to the LED color spectrum with his high brightness blue LED. In 1996 Nakamura announced bright white and green LEDs.
In the early 2000s LEDs started to be used on common use lighting devices [11].

2.1.2 VLC History

Using visible light to transmit data is not a entirely new idea. Fire and smoke signaling were used in ancient civilizations. For example, Roman records indicate that polished metal plates were used as mirrors to reflect sunlight for long distance signaling. Chinese used fire beacons followed by the Romans and American Indians used smoke signals [12].
In Greek history we can find some references to communications using fire. Herodotus described a fire signal sent from Sciathus to Artemisium in 480 BC. Later, Polybius wrote about two devices that used fire to send messages. The first one was developed by Cleoxenus and Democleitus and then perfected by Polybus. This device used a complex system of torches and tables to send messages. The alphabet was divided in five parts and each part was written into a table, then the transmitter of the message raised two torches and waited until the receiver did the same. After that the dispatcher of the message raised the first set of torches on the left side indicating which tablet is to be consulted followed by the raise of a set of torches on the right signaling which letter of the table should be written.

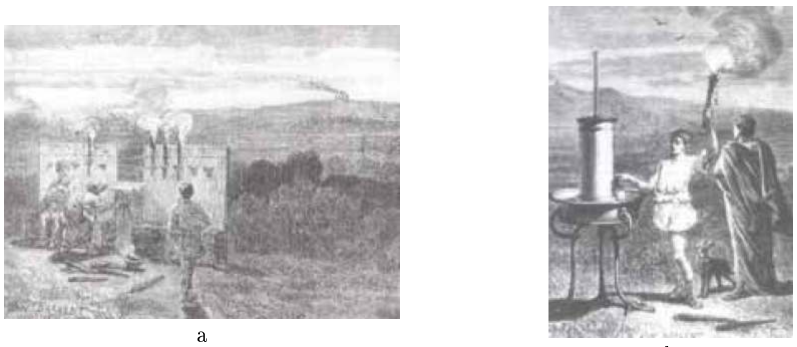
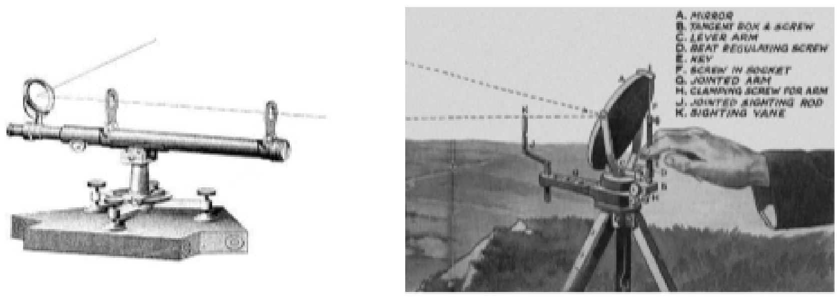


Figure 2.2: Polybus telegraph (a) and Aeneas water clocks (b) [13]

The second one, developed by Aeneas Tacticus was similar to a telegraph and was named *water clocks*. To communicate using *water clocks* every communicating part had exactly the same jar, with a same size hole that was closed and the same amount of water in it. In the jar was a stick with different messages written on. When one part wanted to tell something to the other it made a fire-sign. When the other answered, both of them opened the hole at the same time. And with the help of another fire-sign closed it again at the same time, too. In the end the water covered the stick until the point of the wanted message [14].
By the same time lighthouses appeared to help ships navigate through dangerous

coastal areas, the most famous example is the "Pharos of Alexandria" which was built around 300 BC.
In 1821 Carl Friedrich Gauss invented the Heliotrope, an instrumented that used a mirror to reflect sunlight over great distances. This instrument was used to mark the position of the participants in land surveys and was used since 1822 until the late 1980s when GPS measurements replaced it. The Heliotrope was the start point for the Heliograph. Heliograph was a wireless solar telegraph that signals using Morse code flashes of sunlight reflected by a mirror. The flashes are produced by momentarily pivoting the mirror, or by interrupting the beam with a shutter. The navy often uses blinking lights, i.e. Aldis lamps, to send messages also using Morse code from one ship to another [15].



a b

Figure 2.3: Heliotrope (a) and Heliograph (b)

The first modern device built with VLC purposes was the photophone made by Alexander Graham Bell in 1880. Bell's photophone was based on transmitting sound on a beam of light. A person's voice was projected through an instrument toward a mirror. The vibrations of the voice caused similar vibrations in the mirror. Sunlight was then directed into the mirror, where the vibrations were captured and projected into the photophone's receiver. There they were converted back into sound [16].

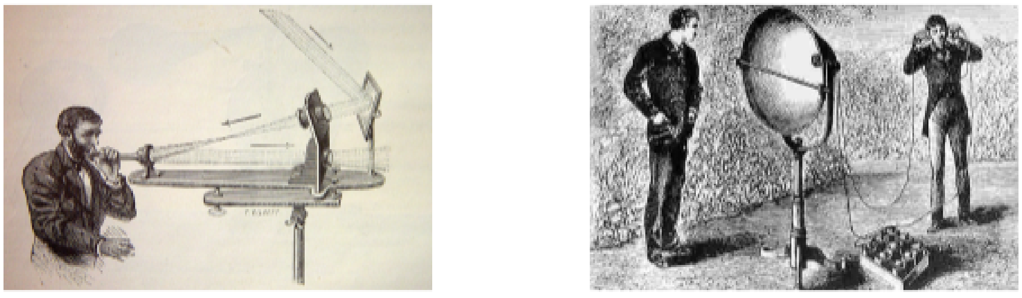

a b

Figure 2.4: Photophone transmitter (a) and receiver (b) [17]

Seventy five years after the photophone, Eugene Polley invented the Flash-Mati,c a TV remote controller that used a light beam to interact with the TV. By pointing the device to one of the four photo-cells located at each corner of the TV users were able change channel, mute and unmute the volume and turn the TV set *ON* or *OFF*. This device was not a great success because it was very sensible to interferences. External light sources, such as solar light or lighting devices were able to produce the same effects as the remote control, leading the TV to trigger the remote control functions when they were not wanted [18].

a

b

Figure 2.5: Flash-Matic (a) and TV set with photo-cells (b)

During the year of 1961 Infrared Industries, an American company presented the Astro-Phone. This device was a kind of walkie talkie that used light instead of RF to communicate. This set was composed by two similar devices and allowed bi-directional communication, each of the devices used a regular light bulb and a filter to block all but infrared light to send data. Ten years after that Nintendo launched the Light Phone, another walkie talkie set based on light communication. The Light Phone used an incandescent torch bulb to transmit voice. Voice was picked up by the microphone, coded and transmitted using the light bulb. On receivers side was a photocell to receive the light pulse, then it was decoded and voice was played through the headphones.

a

b

Figure 2.6: Astro-Phone (a) Light Phone (b) [19]

In the early 2000's Keio University in Japan started his work on data transfer using LEDs and the visible light communication consortium was founded by Japanese tech-companies. Later, in 2011, IEEE workgroup 802.17, Task group 7 finished the standardization of VLC, this work has begun in 2009.
Nowadays there are lots of companies and investigation groups working on VLC.

2.2 Actual State of VLC Communications

In recent years, interest in OW as a promising complementary technology for RF technology has gained new momentum fueled by significant deployments in solid state lighting technology. In this section a brief review about VLC state of the art will be provided. Here three points will be focused: Indoor VLC, Image Sensors as VLC Receivers and VLC under water.

2.2.1 Indoor VLC

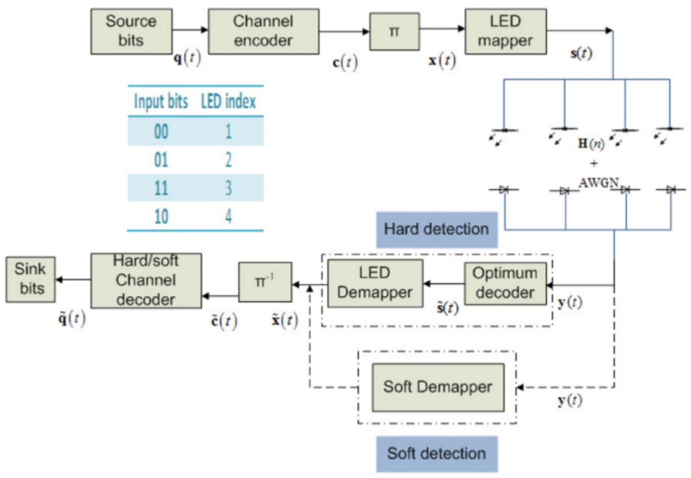
Finding a modulation scheme that offers low bandwidth needs is on of the principal challenges when trying to achieve high data rates for VLC transmission. Researchers have been studying and proposing some solutions for this problem, and some options came out over the last years. Possible solutions are blue optical filtering, spectrally efficient modulation techniques, optical multiple-input multiple-output (MIMO), wavelength division multiplexing (WDM) and their combinations.

OFDM

OFDM was successful used in a European project called OMEGA to transmit data at a rate of 100Mbit/s. This was demonstrated with a broadcast of 4 high definition video streams (about 20 Mbit/s each) from 16 LED ceiling lamps to a photo detector placed within the lit area of $\approx 10\text{m}^2$. The same project was also able to keep traffic on the IP layer at $\approx$ 80Mbit/s [20].
There are still records of laboratory trials capable to achieve data rates of 500Mbit/s using a phosphorescent white LED [21] and 800Mbit/s using an Red Green and Blue (RGB) LED [22]. This results were published in 2010 and 2011, respectively. In 2014 IEEE Photonics Technology Letters published an article referring to an system based on a single 50μm gallium nitride LED capable of transmitting data exceeding 3 Gb/s [7]. To date, this is the fastest wireless VLC system using a single LED. To achieve this values OFDM was employed as a modulation scheme to enable the limited modulation bandwidth of the device to be fully used.

Optical MIMO

Obtained data rates could be increased in VLC, like in RF, by transmitting data in parallel over multiple transmitter and receiver units. MIMO transmission scenarios in indoor VLC could be enhanced by the fact of lighting scenarios require white LED arrays in order to achieve a certain illumination level.
Using a combination of MIMO and OFDM a group of scientists demonstrated that it is possible to transmit data at 220Mbit/s over a 1m, with line of sight (LOS) between transmitter and receiver [23].
To improve power and bandwidth efficiency a new modulation technique called Optical Spatial Modulation (OSM) was proposed. In OSM, multiple transmission are used but only one is active at any given time instance. This transmission units are spatially separated and may be seen as constellation points. In figure 2.7 is presented the model for an VLC system using OSM.



Input bits	LED index
00	1
01	2
11	3
10	4

Figure 2.7: OSM system model [24]

OSM proved to be better than on - off keying (OOK) when achieved transmission speed of 60Mbit/s in a system with a bandwidth of 30MHz, against the 30Mbit/s achieved by OOK [25].

2.2.2 Image Sensors as VLC Receivers

Reception of VLC signals is typically made using photodiodes, but instead of these it is possible to use image sensors (*i.e.*, cameras). First ones are better for high speed applications whereas the second ones can be a good choice in applications where determination of the source location is in the focus. Image sensors are capable to obtain images and receive data at the same time. One camera can acquire data from several sources at the same time, because data from separate LEDs is processed in different pixels on the sensor array. This approach of VLC systems could be used on augmented reality applications as well as indoor location systems. Partners around Keio University have demonstrated a three dimensional position measuring system using VLC determining an object location with *mm* accuracy. Another example is demonstrated example is a VLC system, retrieving the global location information based on an LED identification. This system can be used indoor or outdoor. Detailed information about this location systems could be read in [26]

2.2.3 VLC under water

Due to limitations of RF communications underwater, VLC pointed out to be the researchers choice to allow rapid and successful transmission of data between sub-aquatic nodes.
The company *Penguin Automated Systems Inc.* developed an optical communication system the allows operators to control underwater remotely operated vehicles. This system supports a 120° field of view, data rates up to 20Mbit/s and uses over 70 LEDs.
In 2009 a group of researchers published one paper describing *AquaOptical*, an underwater optical communication system. In clear water *AquaOptical* achieved a data rate of 1.2Mbit/sec at distances up to 30m. and in water with visibility estimated at 3m *AquaOptical* achieved communication at data rates of 0.6Mbit/sec at distances up to

9m [27]. A laboratory experiment reported in [28] demonstrated error-free underwater optical transmission measurements at 1 Gbit/s over a 2 m path in a water pipe with up to 36dB extinction rate.

VLC revealed to be an option not only for high speed communications, but also for systems that require lowest data rates. So, over the last years researchers started to investigate the use of this technology on low data rate portable systems. Most of this investigation use smartphones as emitters and/or as receivers. Next section will provide information about some of this projects.

2.3 Low Debt VLC Projects

2.3.1 LED to LED communication

Because LEDs are so commonly used as light emitters it is easy to forget that they are fundamentally photodiodes, and as such, are light detectors as well. So, back in 2003 a group of researchers from *Mitsubishi Electric Research Laboratories, Inc.* proposed the use of LEDs to send and receive information. They built a bidirectional communication system using only LEDs and one microprocessor.
LEDs are photodiodes that are sensitive to light at and above the wavelength at which they emit (barring any filtering effects of a colored plastic package). Under reverse bias conditions, a simple model for the LED is a capacitor in parallel with a current source which models the optically induced photocurrent. The simplest way to make a photodetector out of a LED and a micro-controller is to tie the anode to ground and connect the cathode to a input/output (I/O) pin driven high. This configuration puts the LED under reverse bias conditions and charges its capacitance, After that the I/O pin must be switched to input mode, allowing the capacitance to discharge down to the digital input threshold. Knowing the time that the capacitance takes to discharge it is possible to compute the photocurrent value as well as the amount of received light. Figure 2.8 shows the three referred situations.

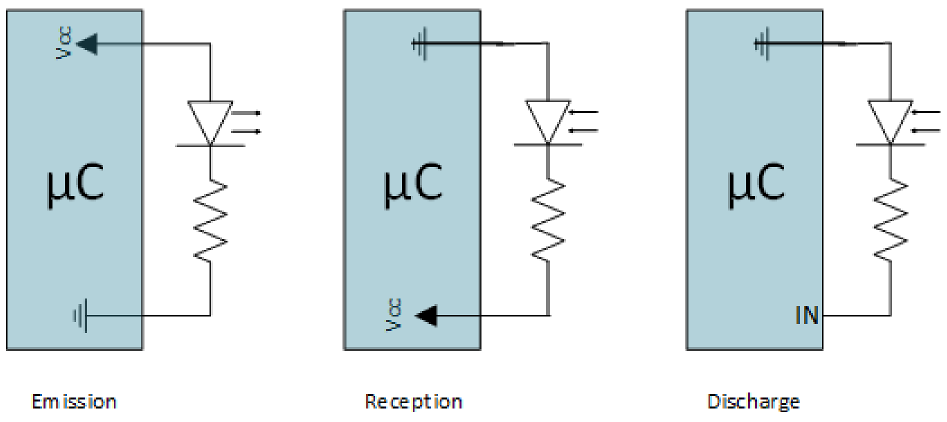


Figure 2.8: Using LED as emitter and receiver

During the elaboration of this project two prototype devices were made, one to be used to control the backlight of remote controls and another that can be used as a smart key for building access or to replace Radio Frequency Identification (RFID). The range

of communication in this project was very short and limited at 250bits/second in each direction [29].
It is important to say that LEDs can not transmit and receive data at the same time. When a LED is being used as receiver for long time periods it will not play its role as lighting device, this may be a problem if VLC is to be integrated with existing lighting devices.

2.3.2 Disney Research

Disney Research labs, in Zürich are working in the field of VLC. They have projects to use this technology in theme parks and for toy networking.
In [30] we can find the explanation of one VLC led-to-led system that can be used for toy networking. A brief explanation of all implementation layers is provided. In terms of data rate this application provides link speed of 100 - 200 bits/second over short distances. The maximum distance presented is 9 cm, with a throughput of less than 50 bits/second. In figure 2.9 we can see the shapes of received data and discharging voltage for one LED being used as receiver.

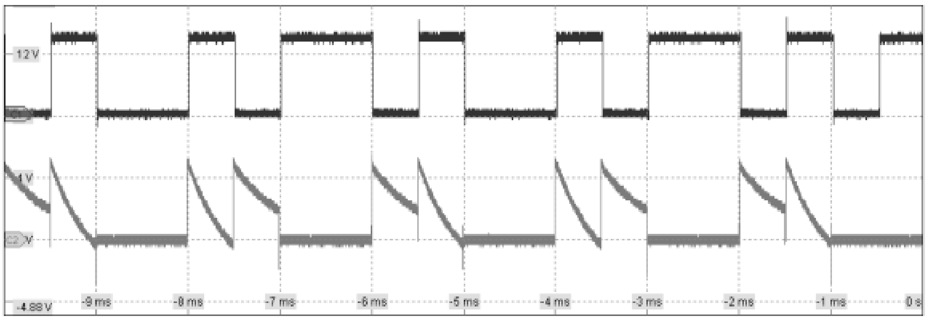


Figure 2.9: Pulse shapes for transmitted data (top) and discharging voltage (bottom) [31]

In July of 2014, IEEE Communications Magazine published an article reporting experiences with several prototypes of practical VLC systems. This systems include LED-to-LED communication, smartphone to LED communication using the flashlight LED and LED to smartphone communication using camera. The system with better performance was the one that used LED-to-LED communication, which achieved a realistic data rate of 8 kbits/s. For communications between smartphone and LED the obtained data rate was 2 bits/s. The LED to smartphone protocol was tested with two different encoding techniques: blink and aliasing. In the Blink method, the LED flickers at rates visible to the human eye (around 10 Hz in our implementation), whereas in the Aliasing method, the LED is modulated at high rates such that the LED is perceived as constantly turned on. Blink scheme achieved 1 bit/s realistic data rate, whereas aliasing scheme allowed communications at 0.5 bit/s [32].

2.3.3 Casio

Casio Computer Co., Ltd. has unveiled its prototype of VLC System Using Smartphones at Consumer Electronics Show (CES) 2012 [33].This prototype has two components: smartphone to smartphone communication and information transmission to smartphones using LED and digital signage. The smartphone to smartphone communication system uses the smartphone's screen to send data that would be decoded by another smartphone camera when it takes a picture. To send data the system flashes smartphone screens. This receiver could acquire data from up to five mobile phones simultaneously and can display messages of up to 120 characters with customizable balloon shapes and image frames. This could also send information about photographed phone owners, such as phone number or social networks link, into the taker's mobile phone.

2.3.4 Using Smartphones as Secure Keys

Nowadays people do not need to regularly go to the bank to deal with economical issues. They can use internet or phone banking as well as ATM machines for performing payments, withdrawals, transfers and even money deposits. With this services the number of magnetic cards in people's wallets is expected to increase. Once smartphones are strongly present in our life, companies and researchers started the quest for one solution for that problem using these equipments. Google and Samsung developed Near Field Communication (NFC), which requires the user to tap the back of his NFC enabled smartphone on the surface of an also NFC enabled card reader to transmit the required credit card information [34]. Square created *Square Reader* a device that is used to accept credit card payments by connecting to a mobile device's audio jack. Square's original reader consisted of a simple read head directly wired to a 3.5 mm audio jack, through which unencrypted, analogue card information was fed to smartphones for amplification and digitization. Both solutions have some drawbacks, such as lack of security, incompatibility or high complexity [35].
A group of researchers thought that maybe VLC could be the solution [36] and created a prototype that used a smartphone flash to send card data to a receiver. The used smartphone was a Samsung device running Android OS 4.2.2 and the receiver was a photodetector connected to an Arduino Mega kit. The Arduino Mega was responsible to decode data and send them to a computer to be displayed. Speaking about modulation and data rates, the OOK modulation scheme failed to correctly receive the transmitted signal. Using Pulse Width Modulation (PWM) the system achieved error-free transmission of the required card information at speeds up to 4.2 and 15 bits/s using light-dependent resistor (LDR) and photodiode detectors, respectively.

Chapter 3

Exploring Smartphone's Flashlight to Transmit Data

3.1 Concept

In this project we are trying to use the flashlight present in one smartphone to act as emitter in one VLC system. This system is to be used for communication purposes with link distance bellow 2 meters and in environments illuminated with LED lamps. The use of LED lamps is required to avoid errors caused by light flickering that is characteristic of incandescent lamps. Once we are planing to use pulse modulation we do not have to concern about amplitude noise. But some cautions with the noise caused by signal transitions are needed, once that kind of noise could affect the system's synchronism and introduce errors on the information decode.
We pretend to implement a low data rate system that can be used in the field of electronic shelf labeling. An electronic shelf label (ESL) system is used by retailers for displaying product pricing on shelves. Typically, electronic display modules are attached to the front edge of retail shelving. These modules use liquid-crystal display (LCD) or similar screen technologies to show the current product price to the customer. A communication network allows the price display to be automatically updated whenever a product price is changed. This communication network can be based on radio, infrared or even visible light communication.
Our project can be used to do an upgrade to exiting shelf labeling systems, allowing costumers to provide feedback about is preferences to retailers. This feedback will be provided using one mobile application, and could be something like *"I like this product"* or *"I do not like this product"*. To make this option viable the existing shelves must be accompanied with photodetectors.

3.2 Flash Control Strategies

In this section we will explore and explain various ways in how the flash could be accessed and controlled.
Almost every smartphones have an high power LED, which is mainly used to provide illumination to the camera, and to serve as a low power flashlight. The advances in VLC technology revealed that this LED can be used for communications, replacing

existing technologies such as RFID and NFC. However the flash sub-system was not designed for that purpose, it was designed and implemented for short time illumination when taking photos in dark environments or for being used as illumination source for extended periods. Despite of that, smartphones have one characteristic that allows the use of the LED as communication interface. They guarantee that applications could access and manipulate the LED state. Knowing the value of LED state, and changing it allow to turn *ON* and *OFF* the flashlight. Some smartphones also allow applications to manipulate the value of LED intensity, with up to 4 possible intensity levels.
The smartphone used for this project was an *HTC One X* which has one LED that can handle currents up to 700mA, allowing both time and amplitude modulation for transmitted data. This project only used the time control. In *HTC One X* an *AAT1271* chip[37], exposed through the *Tegra30* GPIO chip [38], drives the LED. On top of this, the Linux kernel will use the *AAT1271* driver software, and a flashlight driver.
A generic *NVidia Tegra30* based device running the Android OS offers four different strategies to access the flashlight, these strategies are shown in figure 3.1, identified with numbers for 1 to 4. It is important to know, describe and characterize the different strategies of flash access and control that may be used by applications. Each one of the strategies provide different levels abstraction and functionality over the basic function of the LED. The following subsections discuss the available methods for flash control.

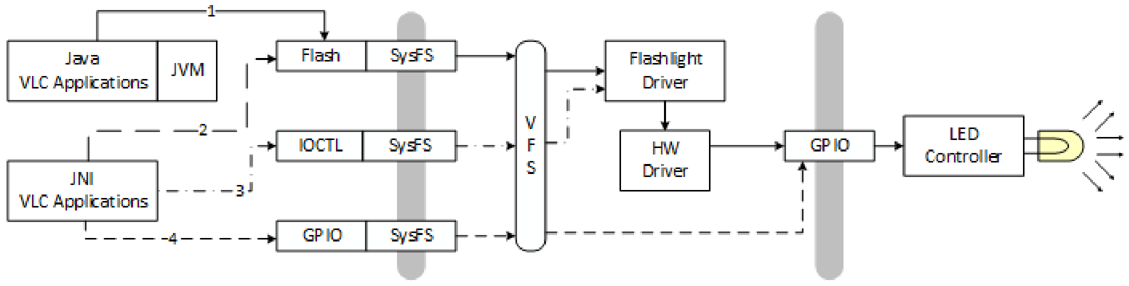


Figure 3.1: Different ways of controlling the flashlight

3.2.1 Java approach

Java is widely used to develop embedded and mobile applications, games, web content and enterprise software. With more than 9 million developers all over the world, Java is present in a huge amount of devices from mobile phones to scientific supercomputers. Java has been tested, refined, extended, and proven by a dedicated community of Java developers, architects and enthusiasts. This allow the use of Java in the development of high-performance applications for a great variety of devices and platforms. With Java it is possible to write software on one platform and run it on virtually any other platform. This facts led to the use of Java in Android OS software development[39].
In spite of the chosen programing language is Java, Android terminals do not run Java code. Programs are commonly written in Java and compiled to bytecode for the Java Virtual Machine (JVM), which is then translated to Dalvik bytecode (.dex or .odex files).
Figure 3.2 shows the process of compiling and installing a Java application into one Android OS terminal.

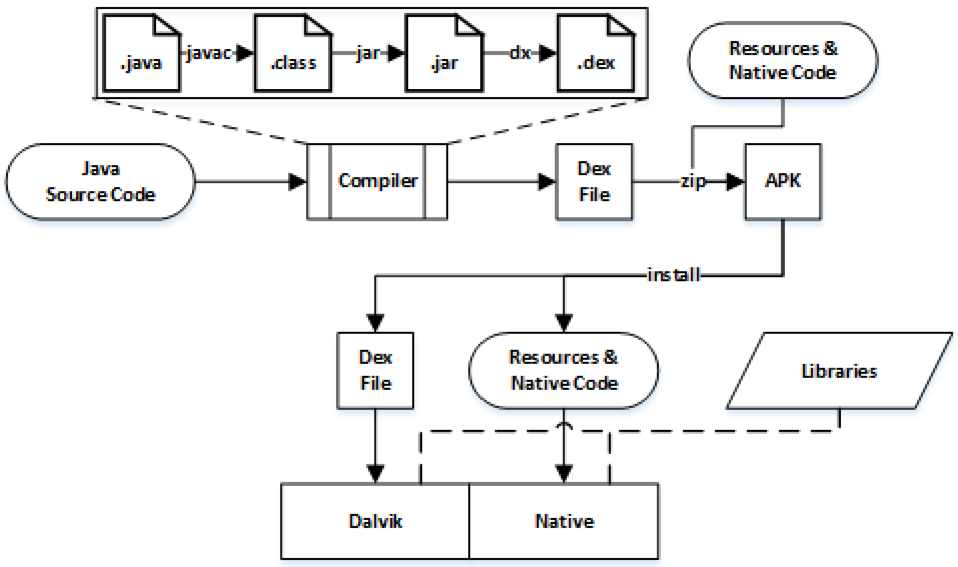


Figure 3.2: Compile and Install Android OS Application

Dalvik is the process virtual machine (VM) in Android OS which runs over the Linux kernel and executes Java applications written for Android. Unlike JVMs, which are stack machines, the Dalvik VM uses a register-based architecture which requires fewer but more complex virtual machine instructions.
A tool called *dx* is used to convert Java *.class* files into the *.dex* format. Multiple classes are included in a single *.dex* file. Duplicate strings and other constants used in multiple class files are included only once in the *.dex* output to save space in memory. Java bytecode is also converted into an alternative instruction set used by the Dalvik VM. An uncompressed *.dex* file is typically a few percent smaller in size than a compressed Java archive (JAR) derived from the same *.class* files.
Programing in Java it is easy to access flashlight control, once it uses the Camera class provided by the Android Java Software Development Kit (SDK). The implementation of this control interface is high level and is well documented. Applications that use this kind of approach can run on any Android OS smartphone because Camera class is natively supported by the OS and abstracts the access to the hardware layer.
The biggest problem of using this approach is the fact that it is a Java interface running inside Dalvik JVM. Despite of granting portability JVM needs to interact with the world outside it and this interaction costs some time. This time costs are due to synchronization mechanisms and context switching.
This approach offers a low level of control over flash LED and imposes great time delays during the execution. Along with the delays we found a enormous dispersion on the obtained time values, making impossible the use of an Java application for communication purpose.
To reduce de impact of all the mechanisms involved in the execution of Java applications on the time precision it was decided to implement the LED control interface using Java Native Interface (JNI). JNI is a Java platform that interacts directly with the machine and can be employed to increase the interaction with the hardware improving efficiency. JNI code uses both Java and native languages, such as *C*, *C++* or *assembly* and is written using the Android Native Development Kit (NDK). The NDK is a toolset that allows developers to implement parts of their applications using native languages. Since

Java code runs on Dalvik and native code is compiled to binaries that run directly on the operating system, JNI ensures correct cooperation between the two worlds.
As it can be seen on figure 3.3 native code runs on top of the Android system, as well as Dalvik VM and both require the system to provide the execution environment. JNI belongs to the Dalvik VM and establishes two way communication between native code and Java code, enabling the use of Java methods by native code and the use of native methods by Java code. During the execution of this project JNI was used to perform the interaction between Java and *C* methods[40].
Using JNI it was possible to access and control the flashlight using three different methods: using Sys File System (SysFS), using the Input Output Control Interface (IOCTL) and using General Purpose Input/Output (GPIO). The use of this three approaches will be now described.

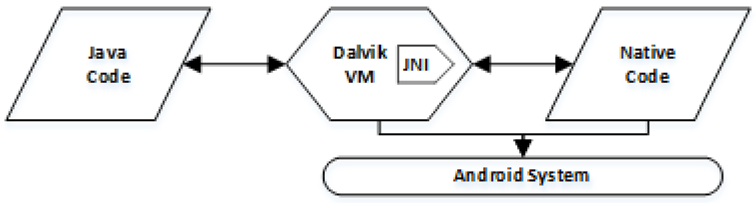


Figure 3.3: Relationship between Java code, Dalvik VM and native code

3.2.2 Accessing LED Through SysFS

SysFS is a virtual file system provided by the Linux kernel. SysFS is used to export information about several kernel subsystems, hardware devices and associated device drivers from kernel's device model to user space. This file system has an strict hierarchical directory organization, based on the kernel's internal data structures organization. SysFS creates mostly ASCCII files with one value per file, to ensure that the exported data is accurate and easily accessible. This makes SysFS one of the most intuitive and useful features of the Linux kernel. It provides two components: a kernel programming interface for exporting data via SysFS, and a user interface to view and manipulate these items that maps back to the kernel objects which they represent. Table 3.1 shows the mapping between internal kernel constructs and their external (user space) SysFS mappings[41].

Kernel	**User Space**
Kernel Objects	Directories
Object Attributes	Regular Files
Object Relationships	Symbolic Links

Table 3.1: Mapping between kernel and user space

SysFS is typically mounted in */sys*. In the smartphone used for this project, the Linux kernel exposes the flash LED in a directory available at */sys/class/leds/flashlight*. Inside this folder there is a special file named *brightness* that can be written in order to control the flash brightness, this file supports integers between 0 and 255. A 0 will turn the flash OFF, while a number between 1 and 255 turns ON the flash. Despite the big range of supported values, only a few brightness levels are allowed, while other values may result on a blinking LED or blinking screen. Every time we need to change

the value present on *brightness* it is necessary to open the file (*fopen()*), write a new value to it (*fprintf()*) and then close the file (*fclose()*), once the changes only take effect when file is closed.
This control interface provides lower latency and jitter once it avoids the JVM synchronization mechanisms. The use of SysFS does not offer portability once each smartphone model has a different kernel structure. Moreover, using SysFS interface implies that some software layers are still present, such as Virtual File System (VFS), the Flashlight driver, and the LED software driver.
This approach allowed to get finer control over T_{ON} and T_{OFF}, but with some unwanted delays.

3.2.3 Using Input Output Control Interface to control the LED

Additionally to the read and write functions, file based mechanisms offer the possibility of additional control commands which are supported by the Input Output ConTroL (IOCTL) method. The IOCTL mechanism is implemented as a single system call which multiplexes different commands to the appropriate kernel space function. A call to ioctl has three arguments: a file descriptor, a number identifying the command, and a data argument. The multiplexing is done based on the file descriptor or on the number of the command. Conceptually it would be possible to use any number for new IOCTL commands, but this is strongly discouraged, and a system wide unique number should be used instead. This ensures that it is not possible to execute an IOCTL on a wrong device leading to unexpected behavior.
Using this approach the obtained results that are very similar to the SysFS control. Despite of IOCTL calls are directed to the Flashlight driver, access to the brightness control file requires opening the file, writing a value and closing the file, as the value is only committed when the file is closed. Each of these operations requires at least one IOCTL, totaling three interactions between the control application and the kernel. This approach also does not offer portability and have unwanted delays with almost the same values as the SysFS approach.

3.2.4 Controlling the LED With General Purpose Input Output

The last tested approach takes advantage of the GPIO interface. GPIO means "General Purpose Input/Output" and is a special pin present in some chips that can be set as input or output and used to set a signal high or low (in output mode) or to get the signal current status (in input mode). Usually this pin is directly managed by kernel modules but there is an easy way to manage these pins also from user space.
This approach offers almost unrestricted control because it allows sending commands directly to LED driver chip, through the GPIO chip. This access can be made in two ways, one is through SysFS and other is by accessing directly the mapped memory space. It provides great access, but requires deep knowledge about how hardware is structured in each smartphone. Applications made using this approach are good for specific uses but do not permit portability between various smartphone models. Fortunately, since GPIOs are identified by numbers, the Linux kernel provide human

readable alias, enabling dynamic discovery of the relevant GPIO to use to drive the LED state. This method was the one that allowed to control the LED with shorter pulses. Nevertheless it also imposes some unwanted delays, much smaller than other approaches. If in previous flash control strategies we had delays in the order of milliseconds that are different for T_{ON} and T_{OFF}, here we have a delay of approximately $100\mu s$ that is equal for both LED states.

3.3 Modulation Techniques

Some pulse modulation techniques are presented in this section. Pulse modulation techniques were used because it is easier to control the time that the flashlight is *ON* or *OFF* than control the amplitude of pulses.

3.3.1 IEEE Standard for VLC

The standardization of VLC was in charge of IEEE 802.15.7 Visible Light Communication Task Group. This task group defined the physical (PHY) and a medium access control (MAC) layer for short-range optical wireless communications using visible light in optically transparent media. The standardization process begun in 2009 with the inaugural meeting for task group 7, where it was chartered to write standards for free-space optical communication using visible light. In December 2011 the MAC and PHY standards for VLC where released, this standardization was very important since it allowed researchers and developers to work in products that will be compliant with the international standard.
Despite of the standardization cover PHY and MAC layer, in this section information about only PHY layer will be provided, once this is the one where light is really used. Although MAC layer is rely important because it handles physical layer management issues such as addressing, collision avoidance and data acknowledgment protocols. Three main topologies are supported by MAC layer: peer to peer, star configuration and broadcast mode. Figure 3.4 shows this three scenarios.

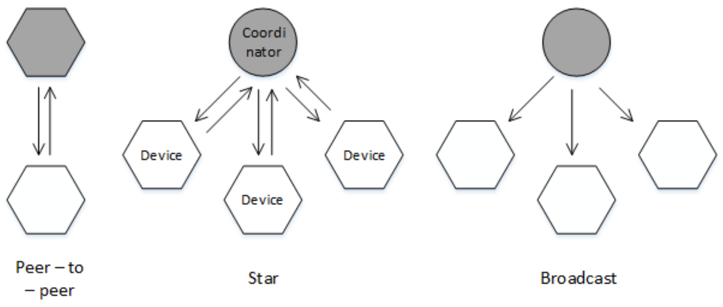


Figure 3.4: MAC Topologies

The physical layer is divided in three types and employs a combination of different modulation schemes: On-off keying (OOK), Variable pulse position modulation (VPPM) and Color shift keying (CSK).
OOK is used because it is simple to code and decode. The 802.15.7 standard uses Manchester coding for low bit rates, once Manchester coding doubles the bandwidth

needed for OOK transmission and uses run length limited (RLL) for higher bit rates. Pulse Position Modulation (PPM) encodes data using the position of a pulse inside a time slot, it requires the duration of the time slot to be long enough to permit the identification of different pulse positions. VPPM es almost equal to PPM but it allows the pulse width to be controlled for light dimming support. CSK needs the use of RGB LEDs on the illumination system, since it codes data using by the color of emitted radiation. Mixing of the red, green and blue primary sources produces the different colors which are coded as information bits. The disadvantage of this system is the complexity of both the transmitter and the receiver.
The three physical layers have different specs for different uses. PHY I is used for data rates between 11.67 and 266.6 kb/s and is designed for outdoor applications. Convolutional and Reed Solomen codes can be used for forward error correction, and OOK or VPPM are used for modulation. PHY II is designed to operate indoor with data rates between 1.25 and 96 Mb/s. It may use Reed Solomen codes to forward error correction and uses the same modulation techniques as PHY I. PHY III can only be used when RGB senders and receivers are available and provides data rates between 12 and 96 Mb/s. CSK is the obvious choice for modulation scheme and Reed Solomen codes are used again for forward error correction. Detailed information about the IEEE 802.15.7 standard can be found at[42].

3.3.2 On Off Keying

On-off keying (OOK) is the simplest form of amplitude-shift keying (ASK) modulation. It codes information as the presence or absence of a carrier wave. In its simplest form, the presence of a carrier for a specific duration represents a binary one, while its absence for the same duration represents a binary zero. Some more sophisticated schemes vary these durations to convey additional information. Due to its simplicity OOK is widely used for intensity modulation/direct detection in optical communications. Two OOK codification schemes could be used, return-to-zero (RZ) and non-return-to-zero (NRZ). Figure 3.5 presents the mapping of OOK-NRZ and OOK-RZ with a duty cycle γ=0.5.

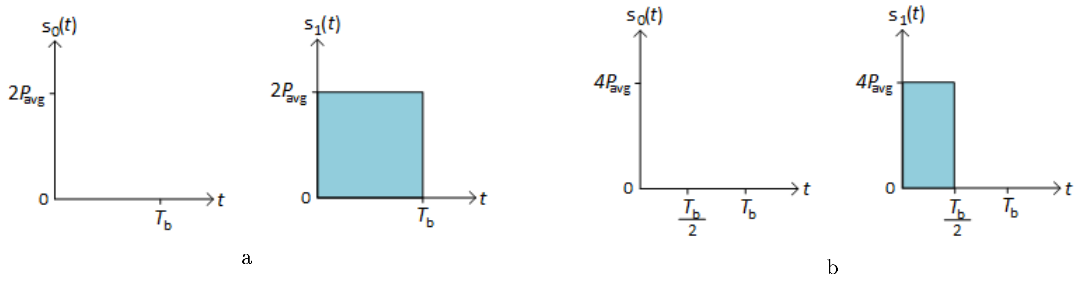


Figure 3.5: Transmitted wave forms for OOK-NRZ (a) and OOK-RZ (b)

The NRZ scheme is the one that codes the bit one with a pulse that occupies all the bit duration slot, RZ scheme uses a pulse that partially occupies the bit duration.
The envelope for OOK-NRZ is given by

$$p(t) = \begin{cases} 2P_r & t \in [0, T_b] \\ 0 & \text{elsewhere} \end{cases} \tag{3.1}$$

where P_r is the mean power and T_b is the bit duration.
OOK is used in many commercial optical wireless systems, due to its simplicity. It is used in IrDA and Fast IR links, for example.
Assuming independently and identically distributed ones and zeros it is possible to compute electrical power spectral densities for OOK-NRZ and OOK-RZ (γ=0.5) using following equations, $\delta()$ represents the Dirac delta function.

$$S_{OOK-NRZ}(f) = (P_r R)^2 T_b \left(\frac{\sin \pi f T_b}{\pi f T_b} \right)^2 \left[1 + \frac{1}{T_b} \delta(f) \right] \tag{3.2}$$

$$S_{OOK-RZ(\gamma=0.5)}(f) = (P_r R)^2 T_b \left(\frac{\sin\left(\frac{\pi f T_b}{2}\right)}{\frac{\pi f T_b}{2}} \right)^2 \left[1 + \frac{1}{T_b} \sum_{n=-\infty}^{\infty} \delta(f - \frac{n}{T_b}) \right] \tag{3.3}$$

Using MatLab® it is possible to plot the theoretical PSDs for the OOK modulation scheme, as shown in figure 3.6.

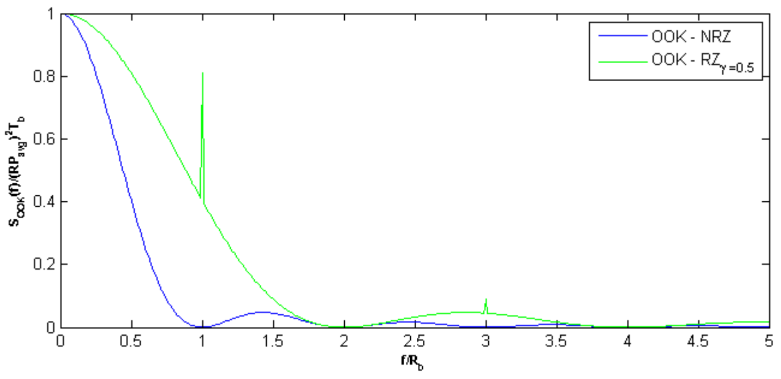


Figure 3.6: Power Spectral Density of OOK-NRZ and OOK-RZ(γ=0.5)

For baseband modulations the bandwidth required is generally defined by the span between DC and the first null of the transmitted signal PSD. Observing Fig. 3.6 it is possible to infer that OOK-NRZ has low bandwidth requirements than OOK-RZ (γ=0.5). The curve representing OOK-RZ power spectral density has discrete impulses at the odd multiples of the bit rate, those impulses could have an important role in clock recovery mechanisms. On the other hand, OOK-NRZ requires the introduction of some nonlinearities to achieve clock recovery, due to the spectral nulls present at the multiples of the bit rate. Both OOK-NRZ and OOK-RZ have significant power at low frequencies, it means that high-pass filtering will not be very effective removing noise introduced by external light sources.

3.3.3 Pulse Position Modulation

Pulse position modulation (PPM) is an attractive option to use in line of sight optical wireless links where bandwidth is not the main concern, due to its power efficiency. If by one side PPM improves the power efficiency of OOK by the other side it increases bandwidth requirements and system complexity.
An L-PPM symbol consists of a pulse of constant power occupying one slot duration L within possible time slots, leaving the other slots empty. The value of L is 2^M where M

is the bit resolution, an integer greater than zero. Information is encoded within the pulse position, which represents the decimal value of the M-bit input data. A sequence of two different PPM symbols is shown in figure 3.7.

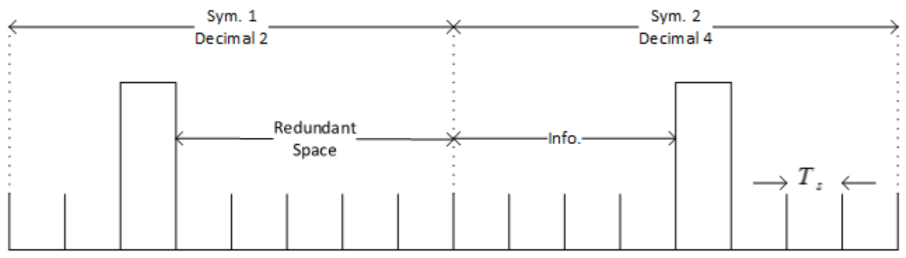


Figure 3.7: Structure of Two PPM symbols (M=3)

If we want the same throughput as in OOK the PPM time slot T_{s_PPM} will be shorter than the OOK bit duration (T_b) by a factor L/M. The following formula is used to compute the value of PPM time slot

$$T_{sPPM} = \frac{T_b M}{L} \tag{3.4}$$

Equations 3.5 and 3.6 can be used to compute the pulse shape for L-PPM and one PPM symbol sequence, respectively

$$x(t) = \begin{cases} 1 & t \in [(m-1)T_{s\_PPM}, mT_{s\_PPM}] \\ 0 & \text{elsewhere} \end{cases}, m \in \{1, 2, ...L\} \tag{3.5}$$

$$x(t)_{PPM} = LP_{avg} \sum_{k=0}^{L-1} c_k p\left(t - \frac{kT_{symb}}{L}\right) \tag{3.6}$$

where $C_k \in \{c_0, c_1, c_2, ..., C_{l-1}\}$ is the PPM symbol sequence, *p(t)* is the pulse shaping function of unitary height and duration $T_{symb}/L, T_{symb}(= T_b M)$ is the symbol duration and $LP_a vg$ is the peak optical power of PPM symbol.
All PPM signals are equidistant and the minimum spacing between them is given by:

$$d_{min-PPM} = min = i \neq j \int [x_i(t) - x_j(t)]^2 dt = 2LP^2 \log_2\left(\frac{L}{R_b}\right) \tag{3.7}$$

The substitution of OOK modulation by PPM will increase the system complexity due to the need of slot and symbol synchronization in the receiver, to demodulate the received signal. Despite of that PPM was the choice in many optical wireless communication systems because it is power effective. We can find PPM in deep space laser communications and portable devices, applications where power consumption is a great concern. Once we are using a smartphone to send data, power consumption is a great concern. Smartphones are battery powered devices and less power consumption means more battery time.
Using some mathematical formulas it is possible to the PSD for PPM.

$$S_{PPM}(f) = |P(f)|^2 [S_{c,PPM}(f) + S_{d,PPM}(f)] \tag{3.8}$$

$P(f)$ represents the Fourier transform of the pulse shape and $S_{c,PPM}$ and $S_{d,PPM}$ are the continuous and discrete components, respectively.

$$S_{c,PPM}(f) = \frac{1}{T_{symb}} \left[\left(1 - \frac{1}{L}\right) + \frac{2}{L} \sum_{k=-\infty}^{L-1} \left(\frac{k}{L} - 1\right) \cos\left(\frac{k2\pi t_{symb}}{L}\right) \right] \tag{3.9}$$

$$S_{d,PPM}(f) = \frac{2\pi}{T_{symb}^2} \sum_{k=-\infty}^{\infty} \delta\left(f - \frac{kL}{T_{symb}}\right) \tag{3.10}$$

Once again MatLab® is a great help to plot the PSD for several values of L.

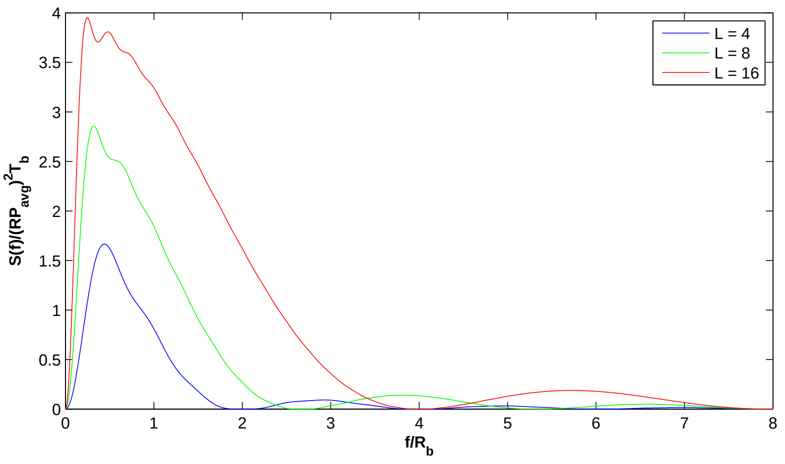


Figure 3.8: Power Spectral Density of PPM for L=4, 8 and 16

As it is possible to observe in figure. 3.8 the position of the first null increases when L increases, it represents that greater L's require more bandwidth.

3.3.4 Pulse Interval Modulation

Pulse interval modulation encodes data by inserting empty slots between two pulses. As it has built-in symbol synchronization PIM offers reduced complexity compared to PPM. The simplest version of PIM is digital pulse interval modulation (DPIM). DPIM is an anisochronous pulse time modulation and offers improved performance compared with PPM by removing redundant space. This results into higher transmission capacity to DPIM when compared with PPM.

Each DPIM symbol is composed by a pulse of one slot duration followed by a variable number of empty slots. There are L possible symbols $\{s(n),\ 0 < n \leq L\}$ of different length and each block of M $(= \log_2 L)$ input data bits $\{d_i,\ i = 1,\ 2,\ ...,\ M\}$ is mapped to one of these symbols. The decimal value of the M-bit data stream being encoded defines the number of empty slots. Once the number of empty slots is variable the symbol duration varies between T_s and LT_s, where T_s is the slot duration. In order to reduce inter symbol interference effect a guard band composed by one or more empty slots can be added to each symbol. This guard band is introduced immediately after the pulse and will change minimum and maximum symbol duration, for example, with one guard slot (1GS) the minimum duration will be $2T_s$ and maximum duration will be $(L + 1)T_s$.

Variations on the value of L allows DPIM to increase either bandwidth efficiency or power efficiency, compared to PPM. If we increase bit resolution, for a fixed average bit rate and fixed bandwidth, we can achieve better power efficiency, once again compared to PPM.
Figure 3.9 shows the mapping for 4-DPIM with no guard slot (NGS) and with one guard slot (1GS).

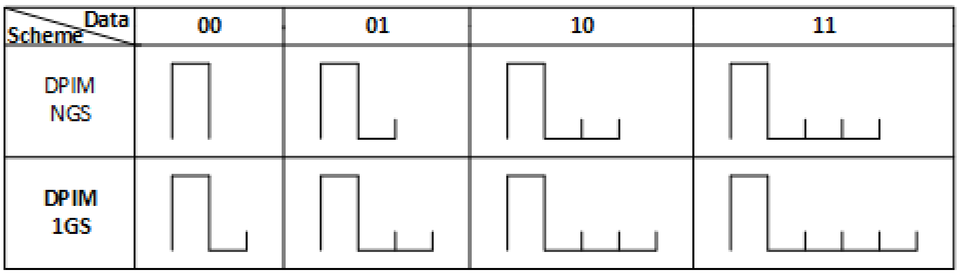


Figure 3.9: Mapping of DPIM Symbols

It is possible to compute slot duration of a DPIM system in a way that the mean symbol duration is equal to the time taken to transmit the same number of bits using OOK or PPM. To perform this calculus the next equation could be used:

$$T_s = \frac{T_b M}{\bar{L}_{DPIM}} \tag{3.11}$$

$\bar{L}_{DPIM}$ is the average symbol length and is given by $(2M+1)/2$ for DPIM(NGS) and by $(2M+3)/2$ for DPIM(1GS).
Assuming random and uniformly distributed symbol length it is possible to compute the value for the bit rate (R_b):

$$R_b = \frac{M}{\bar{L}_{DPIM} T_s} \tag{3.12}$$

The higher transmission capacity of DPIM could be used in several ways for optical wireless communications. It allows to transmit the same average data rate of PPM at near half of the slot frequency, improving the bandwidth efficiency but decreasing the power efficiency. On the other hand side the number of bits per symbol could be increased without an increase of the slot frequency, improving this way the power efficiency. Finally this increased capacity could be used to introduce redundancy into the code. Redundancy will provide error correction and detection mechanisms to the code, allowing the same bit error rate to be achieved at lower transmit power.
This fact could be best appreciated taking a look on the spectral properties of DPIM. One DPIM single sequence is modulated by:

$$S_{DPIM}(t) = \bar{L}_{DPIM} P_r \sum_{k=\infty}^{\infty} c_k p(t - kT_s - \tau_n) \tag{3.13}$$

where c_k is a random variable which represents the presence or absence of a pulse in the *nth* time slot and τ_n is the random jitter within a time slot at the threshold crossing time slot at the receiver. The power spectral density (PSD) model of the DPIM is given as:

$$S_{DPIM}(f) = \frac{1}{T_{s\_DPIM}}|P(f)|^2[S_{c,DPIM} + S_{d,DPIM}] \tag{3.14}$$

where *P(f)* is the Fourier transform of th pulse shape, $S_{c,DPIM}$ is the continuous component and $S_{d,DPIM}$ is the discrete component. This components are given by:

$$S_{c,DPIM}(f) = \sum_{k=-5L}^{5L} \left(R_k \frac{1}{\bar{L}_{DPIM}^2} \right) \mathrm{e}^{-j2\pi k f T_{s\_DPIM}} \tag{3.15}$$

$$S_{d,DPIM}(f) = \frac{2\pi}{T_{s\_DPIM}\bar{L}_{DPIM}^2} \sum_{k=-\infty}^{\infty} \delta\left(f - \frac{2\pi k}{T_{s\_DPIM}}\right) \tag{3.16}$$

When working with rectangular pulses with unit-amplitude and duration T_{s_DPIM} the Fourier transform can be computed as:

$$P(f) = T_{s\_DPIM}\frac{\sin(\pi f T_{s\_DPIM})^2}{\pi f t_{s\_DPIM}} \tag{3.17}$$

The component R_k present in the continuous component is th slot autocorrelation. This value varies according to the type of DPIM. Next will be presented the formulas of R_k for DPIM(NGS) and DPIM(1GS).

$$R_{k-DPIM(NG)} = \begin{cases} \frac{2}{L+1} & k = 0 \\ \frac{2}{L^k}(L+1)^2 & 1 \leq k \leq L \\ \frac{1}{L}\sum_{i=1}^{L} R_{k-i} & k > L \end{cases} \tag{3.18}$$

$$R_{k-DPIM(NG)} = \begin{cases} L_{avg}^{-1} & k = 0 \\ 0 & k = 1 \\ \left(\frac{L_{avg}^{-1}L^{-1}}{\sqrt{1+4L^{-1}}}\right)\left[\left(\frac{1+\sqrt{1+4L^{-1}}}{2}\right)^{k-1} - \left(\frac{1-\sqrt{1+4L^{-1}}}{2}\right)^{k-1}\right] & 2 \leq k \leq L+1 \\ \frac{1}{L}\sum_{i=1}^{L} R_{k-l-i} & k > L+1 \end{cases} \tag{3.19}$$

In figure 3.10 graphics for PSD of DPIM(NGS) and DPIM(1GS) are presented.

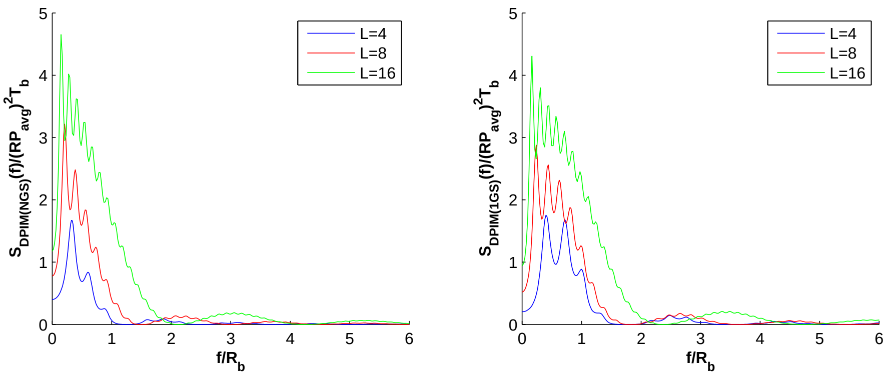


Figure 3.10: PSD for DPIM(NGS) (left) and PSD for DPIM(1GS) (right)

All curves were plotted for the same average optical power, using rectangular-shaped pulses occupying the full slot duration. The power axis is normalized to the average electrical power multiplied by the bit duration and the frequency axis is normalized to the bit rate R_b. Observing the areas under the curves it is possible to see an increment in detected electrical power as L increases. By the observation of null points it is easy to infer that DPIM(1GS) requires more bandwidth than DPIM(NGS).

3.3.5 Dual-Header Pulse Interval Modulation

Dual-Header Pulse Interval Modulation (DH-PIM) is a digital pulse time modulation technique that offers increased transmission capacity requiring less bandwidth and has built-in frame and slot synchronization. In DH-PIM, the nth symbol $S_n(h_n, d_n)$ starts with a header h_n of duration $T_h = (\alpha+1)T_s$ followed by a sequence of d_n empty slots. T_s represents the slot duration and $\alpha > 0$ is an integer. Two headers are possible, depending on the value of the most significant bit (MSB) of the input word. If MSB=0 header one (H1) will be used, if MSB=1 the chose will be header two (H2). H1 and H2 have pulse durations of $0.5 * \alpha T_s$ and αT_s, respectively.

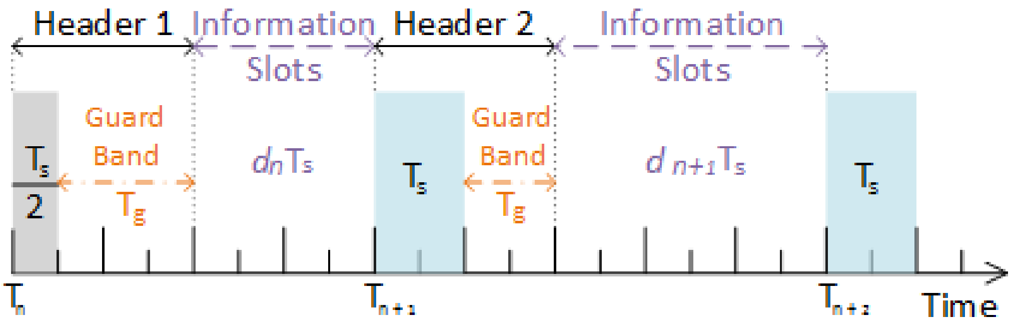


Figure 3.11: DH-PIM Symbol Structure With Two Headers ($\alpha = 1$)

After each pulse there is one guard band of variable length $T_g \in \{(\alpha/2+1)T_s, T_s\}$ to cater for symbols representing zero. The value of $d_n \in \{0, 1, \ldots 2^{M-1} - 1\}$ is the decimal value of the input codeword for symbols which start with H1 or the value of 1's complement for the input codeword when symbols start with H2. Built-in symbol synchronization is possible because the header has the role of symbol initializer and time synchronizer. In addition to the removal of the redundant space in PPM, DH-PIM also reduces the average symbol length of DPIM. This to facts leads to an increased data throughput.
The average symbol length $\bar{L}_{DH-PIM}$ and slot duration T_{s_DH-PIM} of DH-PIM$_\alpha$ are given by:

$$\bar{L}_{DH-PIM} = \frac{2^{M-1} + 2\alpha + 1}{2} \tag{3.20}$$

$$T_{SDH-PIM_\alpha} = \frac{2M}{(2^{M-1} + 2\alpha + 1)R_b} \tag{3.21}$$

Mathematically, a DH-PIM pulse can de represented by:

$$x(t) = A \sum_{k=0}^{\infty} \left\{ p\left[\frac{2(t-T_k)}{\alpha T_s} - \frac{1}{2}\right] + h_n p\left[\frac{2(t-T_k)}{\alpha T_s} - \frac{3}{2}\right]\right\} \tag{3.22}$$

with $A = \frac{4\bar{L}_{DH-PIM}P_r}{3\alpha}$ representing the peak transmitted optical power and $h_n \in \{0,1\}$ indicating H_1 or H_2 respectively.
Table 3.2 shows the mapping of symbols for OOK, PIM(1GS) and DH-PIM

OOK	**PIM(1GS)**	**DH-PIM$_{(\alpha=2)}$**
000	10	100
001	100	1000
010	1000	10000
011	10000	100000
100	100000	110000
101	1000000	11000
110	10000000	1100
111	100000000	110

Table 3.2: Symbol mapping for DPIM(NGS) and DPIM(1GS)

Now some spectral characteristics will be analyzed. The Fourier transform of DH-PIM is given by:

$$X_N(\omega) = V\sum_{n=0}^{N-1}\int_{-\infty}^{infty}\left\{rect\left[\frac{2(t-T_n)}{\alpha T_s}-\frac{1}{2}\right]+h_n rect\left[\frac{2(t-T_n)}{\alpha T_s}-\frac{1}{3}\right]\right\}\mathrm{e}^{-j\omega t}dt \tag{3.23}$$

So,

$$X_N(\omega) = \frac{V}{j\omega}\mathrm{e}^{-j\omega T_0}\left(1-\mathrm{e}^{-\frac{-j\omega T_s\alpha}{2}}\right)\sum_{n=0}^{N-1}\left[(1+h_n\mathrm{e}^{-j\omega T_s\alpha/2})\mathrm{e}^{-j\omega T_s n(\alpha+1)}\mathrm{e}^{-j\omega T_s\sum_{k=0}^{n-1}d_k}\right] \tag{3.24}$$

The PSD of DH-PIM pulse train could be computed by:

$$P(\omega) = \begin{cases} \dfrac{4V^2\cos^2\left(\frac{\alpha\omega T_s}{4}\right)\left\{\begin{array}{c}\left[5-4\sin^2\left(\frac{\alpha\omega T_s}{4}\right)\right] \\ +\left[9-8\sin^2\left(\frac{\alpha\omega T_s}{4}\right)\right]Re\left(\frac{\psi}{1-\psi}\right)\end{array}\right\}}{\omega^2T_s(2^{M-1}+2\alpha+1)} & \omega \neq \frac{2\pi K}{T_s} \\ 0 & \omega = \frac{2\pi K}{T_s} \begin{array}{c} K \text{ even} \\ \text{or } \alpha \text{ even}\end{array} \\ \infty & \omega = \frac{2\pi K}{T_s} \begin{array}{c} K \text{ odd} \\ \text{and } \alpha \text{ odd}\end{array} \end{cases} \tag{3.25}$$

where K is a positive integer and ψ given by

$$\psi = \frac{1}{2^{M-1}}\{1+\mathrm{e}^{-j\omega T_s}+\mathrm{e}^{-j2\omega T_s}+\ldots+\mathrm{e}^{-j\omega(2^{M-1}-1)T_s}\}\cdot\mathrm{e}^{-j\omega(\alpha+1)T_s} \tag{3.26}$$

So, the spectrum consists of a *sinc* envelope when $\omega T_s/2\pi$ is not integer, distinct frequency components at the slot frequency and its harmonics when $\alpha\omega T_s/2\pi$ is an odd integer and nulls when $\alpha\omega T_s/2\pi$ is an even integer. Depending on the values of α the slot component and its harmonics may coincide the nulls of the *sinc* envelope. By consequence the existence of the slot components and the locations of nulls are affected by the pulse shape. The presence of the slot frequency component is useful for synchronization at the receiver, once it could be extracted using a phase-locked loop circuit.
Fig. 3.12 presents the MATLAB graphics of PSD for DH-PIM with fixed α and fixed M.

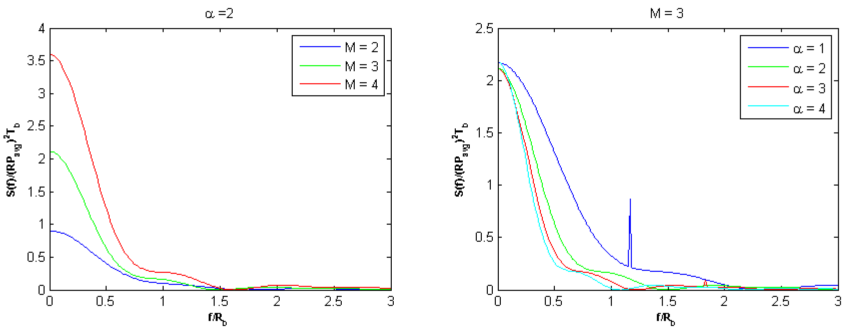


Figure 3.12: PSD for DH-PIM with constant α (left) and PSD for DH-PIM with constant M (right)

Observing fig. 3.2 it is possible to find out that increasing the value of M will increase the detected optical power, for a constant α. It is also possible to see that increasing the value of α for a constant M will cause a decreasing of needed bandwidth. To plot this graphics the value of R_b was normalized to 1.

3.3.6 Comparison of the modulation techniques

Two important points when talking about modulation schemes are the power efficiency and the bandwidth requirement. Since the optical wireless channel is mostly power limited due to the safety issue, the power efficiency is the foremost issue. Power requirement is defined as the average optical power required by an ideal system to guarantee a certain bit rate and error probability.
Using MatLab® we simulated and plotted the average optical power requirements and bandwidth requirements of OOK (NRZ and RZ), PPM, DPIM and DH-PIM. For DH-PIM, DPIM and PPM an increment in the value of L leads to a decrease of the average optical power requirements together with one increase of the bandwidth requirements. In OOK-RZ scheme, lower duty cycles have better power performance but by the other side bandwidth increases for small duty cycles. If we are transmitting symbols from a very little alphabet, more specifically and alphabet with 4 symbols(L=2), the best option is the use of OOK-NRZ, as can be seen on figure 3.13.

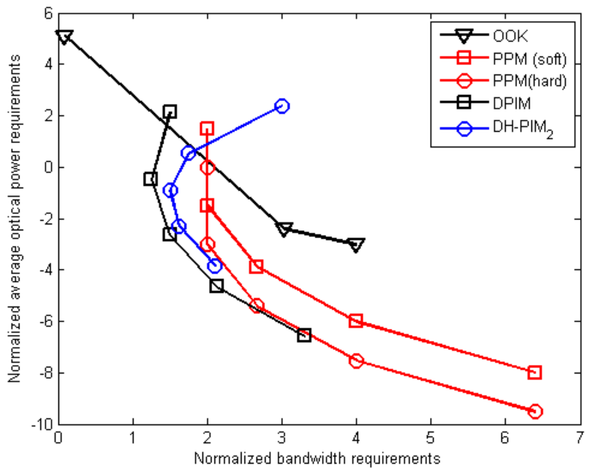


Figure 3.13: Normalized Optical Power Requirements vs. Normalized Bandwidth Requirements [43]

In terms if transmission capacity, DH-PIM$_2$ is the scheme that offers better performance. For low bit rates we can say that DPIM is better that DH-PIM$_1$, but this difference disappears for M $>$ 6. Figure 3.14 shows transmission capacity for several modulation schemes.

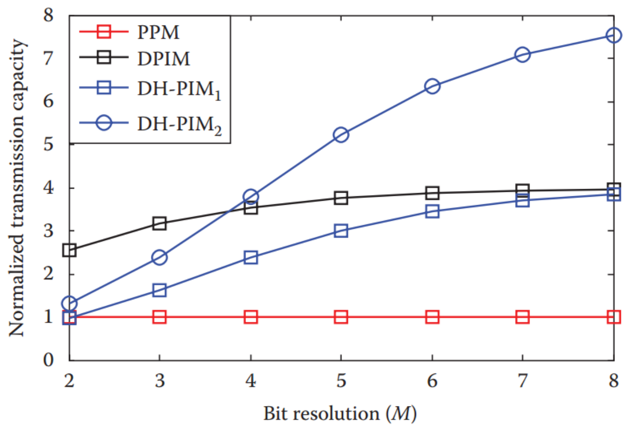


Figure 3.14: Normalized Transmission Capacity vs. Bit Resolution [43]

In an anisochronous modulation scheme such as DPIM, the symbol length is variable. Since errors are not confined to the symbol in which they occur, it is convenient to base the study on the packet transmission rate rather than the bit rate.
In figure 3.15 we can see the curves of the packet transmission rate of PPM, DPIM, DH-PIM$_1$ and DH-PIM$_2$ normalized to that of PPM versus M. As we can see the worst option, based on packet transmission rate is PPM and the best is DH-PIM$_2$. For higher values of M DPIM and DH-PIM$_1$ achieve almost the same transmission rate

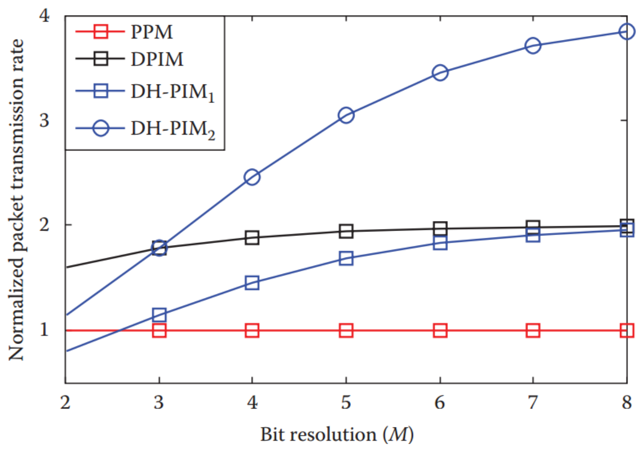


Figure 3.15: Normalized Packet Transmission Rate vs. Bit Resolution [43]

Based on the obtained results we can say that the best modulation scheme is DH-PIM$_2$, and this is the reason why DH-PIM was the chosen modulation for our project.
A deeper analysis on the modulation techniques can be found on [43].

Chapter 4

Developed Work

4.1 Introduction

The experiments made to obtain the results that will be presented were made using one mobile phone as information sender and one device based on an Arduino board as receiver. The experimental set-up could be seen on figure 4.1.

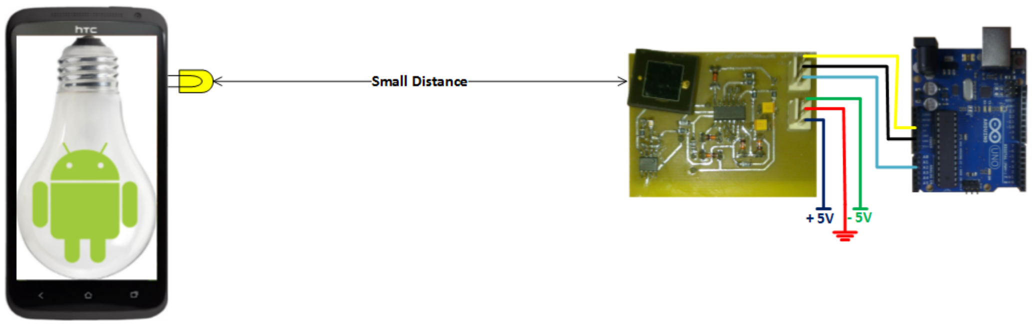


Figure 4.1: Experimental Set-up

This experimental set-up was closed inside a protected box during all the experiments, to avoid external interferences caused by light sources. Both, the emitter and the receiver, were fixed to the extremities of the box to guarantee that the path between the devices were equal for all the tests. The mobile phone used as sender was a HTC One X[44] running Android OS 4.3.1. This smartphone has one Nvidia Tegra 3 chipset, this chipset is a quad-core CPU, with 5th battery-saver core and maximum frequency of up to 1.7 GHz in single core mode or 1.6 GHz as quad-core[45].
To get better performance and to guarantee access to some root functions some modifications where made to the mobile phone's software. The bootloader was unlocked and one custom ROM was installed. The chosen one was *CyanogenMod version 10.2.1-endeavoru*.

The receiver is composed by a photo-detector followed by a transimpedance amplifying stage. The received signal is threshold compared and feed in digital format to an Arduino Uno digital port for further processing, as shown in figure 4.2

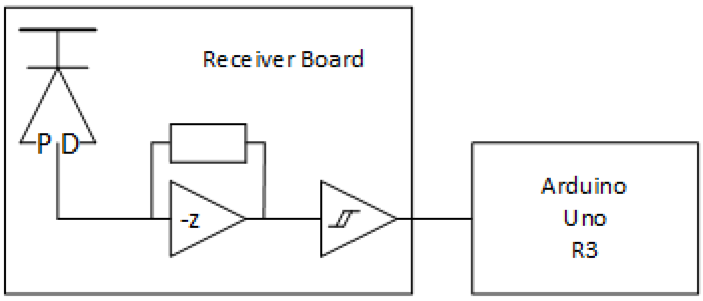


Figure 4.2: Schematic of the receiver set-up

This Arduino is a microcontroller board based on ATmega328. It has 14 digital pins that could be used as input or output and 6 of them could be used as PWM outputs. Furthermore it has 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an in circuit serial programming header, and a reset button[46].
The *loop()* function of Arduino is pooling the digital pin used to receive the signal. When we are gathering times for the *ON* pulse an internal Arduino function is used to obtain the pulse width. The same is made for *OFF* pulses. Figure 4.3 shows one example of what is made in the Arduino function that measures the time intervals.

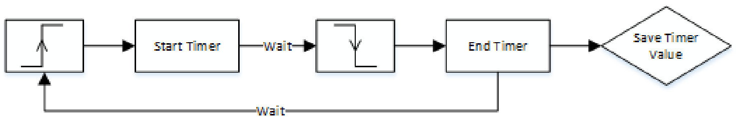


Figure 4.3: Example of the Arduino's Work Flow

Every time a pulse is measured its value is stored into one array, this array is then dumped through the serial port into the PC. To connect the PC with the Arduino we use one USB cable. All the collected data is analyzed using MatLab® .

4.2 Data Analysis

In this section we will evaluate the level of temporal control that can be achieved for *ON* and *OFF* flash pulses.This will allow the characterization of the statistical properties associated with the duration of flash pulses in the proposed communication scheme.
For all results discussed in this section, the tests were made with the flash set to send 30000 consecutive DH-PIM symbols to the Arduino based receiver. The symbols where randomly chosen from the alphabet of *N=8* different symbols shown in table 4.1. For each time value 3 tests were made measuring T_{ON} and 3 measuring T_{OFF}. To control pulse duration a base time (b_t) was defined, the expected values for *ON* are b_t or $2b_t$ depending on the use of header 1 or header 2, respectively. For information slots we expect times equal to nb_t with $n \in [1, 2, 3, 4]$.Figure 4.4 shows an example of the structure of DH-PIM symbols used in this project.

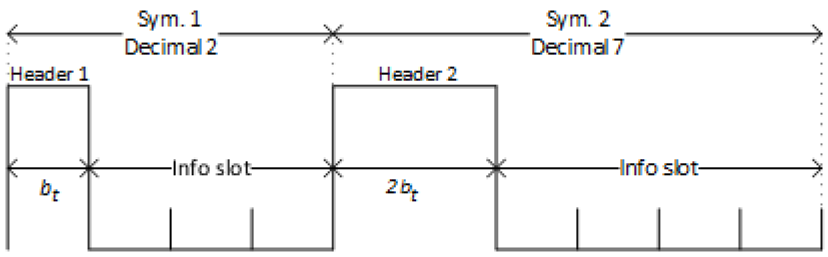


Figure 4.4: DH-PIM Symbol Structure

Decimal	Binary	DH-PIM
0	000	10
1	001	100
2	010	1000
3	011	10000
4	100	110
5	101	1100
6	110	11000
7	111	110000

Table 4.1: DH-PIM used alphabet

4.2.1 Accessing LED Through SysFS

We knew by previous work that using java we can turn *ON* and *OFF* the flash but we don not have any kind of control over the pulse duration, so new approaches were required. The first one we thought was the SysFS approach, previously described. To see if we can precisely control the pulse duration we made tests for base times between 500μs and 900μs using the scheme described above. Here will be presented the analysis of the obtained results for $b_t = 700\mu s$, and some considerations about the general comportment of the system for all the tested times.
In the experience that will be analyzed symbols must start with a *ON* pulse of 700 or 1400μs, depending on H_1 or H_2. This pulses must be followed by empty slots of duration $n{\cdot}700\mu s$, with $n \in \{1, 2, 3, 4\}$. The results provided by Arduino where used to build histograms and to fit them with Gaussian shapes, using MatLab®. figure 4.5 shows the obtained results. It is possible to see that headers and information slots have almost the comportment of Gaussian random variables. For the headers we obtain a mean value μ_1 for H_1 and $T_p + \mu_1$ for H_2 and very similar standard deviations (σ_1). For information slots we have mean values of μ_0 for $n = 1$ and $\mu_0 + k \cdot b_t, k = 2, 3, 4$ for $n = 2, 3, 4$ (n represents the number of empty slots).

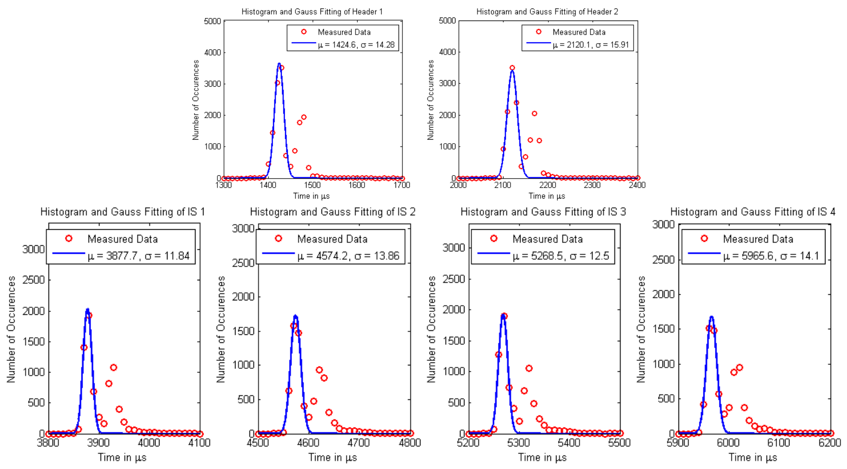


Figure 4.5: Histograms and Gaussian fit for headers and time slots

In table 4.2 it is possible to observe the uncertainty parameters obtained for three consecutive runs, along with difference between expected and obtained mean values for headers and empty slots.
As it is possible to see the distance between peaks is almost equal to b_t, which means we are controlling the time that the LED stays *ON* and *OFF*.

	Headers				Information Slots							
	H_1		H_2		D=1		D=2		D=3		D=4	
Run	μ_1	σ_1	μ_1	σ_1	μ_0	σ_0	μ_0	σ_0	μ_0	σ_0	μ_0	σ_0
1	1424.4	16.1	2119	16.4	3877.2	11.8	4573.2	14	5268.3	12.7	5964.7	15.7
2	1424.6	14.3	2120.1	15.9	3876.7	10.9	4573	13.7	5267.9	11.4	5964.5	14
3	1440.7	45.7	2136.1	45.3	3877.7	11.8	4574.2	13.9	5268.5	12.5	5965.6	14.1
Mean	1429.9	25.4	2125.1	25.9	3877.2	11.5	4573.5	13.9	5268.2	12.2	5964.9	14.6
Wanted	700	-	1400	-	700	-	1400	-	2100	-	2800	-
Diff.	729.9	-	725.1	-	3177.2	-	3173.5	-	3168.2	-	3164.9	-

Table 4.2: Measured uncertainty parameters, with $T_{ON} = T_{OFF} = 700\mu s$ for headers and empty slots

To find out if the difference between defined and obtained times is dependent or not of the time we want the graphic from figure 4.6 was drawn.

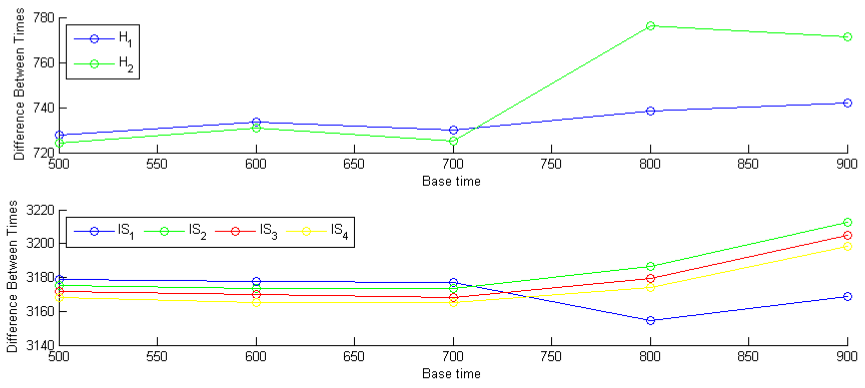


Figure 4.6: Difference between defined and obtained times for $b_t = 500, 600, 700, 800$ and $900\mu s$

Figure 4.6 reveals that for times between 500 and $700\mu s$ the difference between the time we want and the time we obtain is almost constant. When defined time is higher than

$700\mu s$ this difference suffers some variations.This differences might be forced by the *kernel* that imposes minimum times for distance between transitions. On figure 4.7 there it is studied the impact of *ON* and *OFF* nominal timings on the uncertainty parameters.

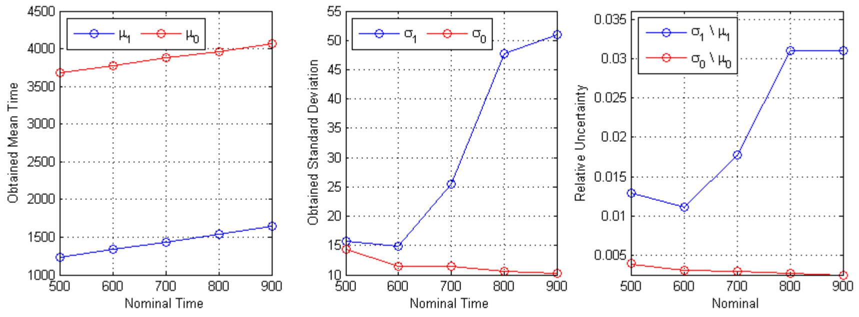


Figure 4.7: Uncertainty parameters for increasing nominal ON and OFF timings (*mean, standard deviation and relative uncertainty*)

Analyzing the graphics it is possible to see that an increase on b_t may lead to decrease on data quality. This decrease can be seen in the graphs of relative uncertainty ($\sigma\backslash\mu$) and is represented by the increase of relative uncertainty for low data rates (*i. e.* for higher *ON* and *OFF* nominal times). This fact can be favorable to this project's purposes because allows data transmission at higher rates with less errors.
In the graphics it is also shown that ones and zeros have similar statistical properties and $\Delta_0 = |T_{OFF} - \mu_0|$ is higher than $\Delta_1 = |T_{ON} - \mu_1|$. After this analysis we made three new runs of the test, but this time defining a very low b_t. This test was made to discover what is the minimum duration for one *ON/OFF* pulse. The obtained values can be seen on figure 4.8 and table 4.3. The difference between defined and obtained values is caused by the system governor, once the minimum obtained values for T_{ON} and T_{OFF} are almost the same as the encountered differences. The minimum possible value for one *ON/OFF* pulse is about $4m_s$

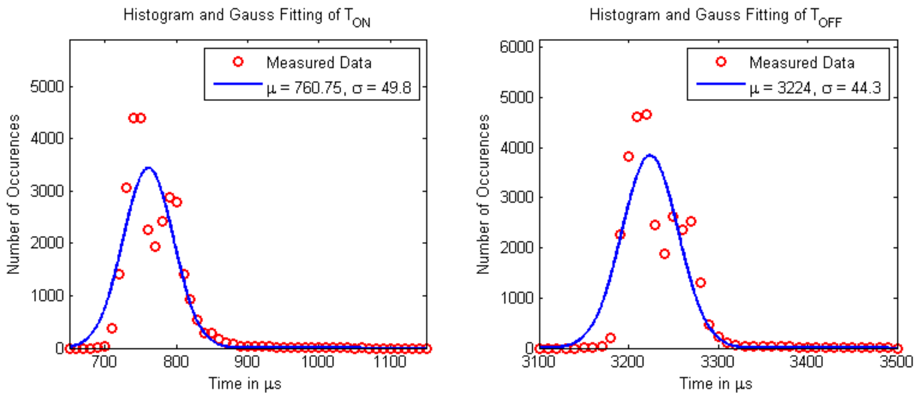


Figure 4.8: Minimum values for T_{ON} and T_{OFF}

	T_{ON}		T_{OFF}	
Run	μ_1	σ_1	μ_0	σ_0
1	761.6	49.8	3224.1	44.7
2	760.8	49.8	3224	44.3
3	760.2	49.5	3224.6	44.6
Mean	760.7	49.7	3224.2	44.5

Table 4.3: Obtained parameters for T_{ON} and T_{OFF}

4.2.2 Using Input Output Control Interface to control the LED

In this section we made tests for the same time values we used on SysFS approach, once the level of control must be similar. This time we tried to find out if there was some minimum limits for T_{ON} and T_{OFF} before starting the tests. This information was important because allowed to know which kind of values we must expect. To do that, the application on the mobile phone was set to turn *ON* and *OFF* the flash without time restrictions. The obtained parameters for T_{ON} and T_{OFF} are synthesized on figure 4.9 and table 4.4.

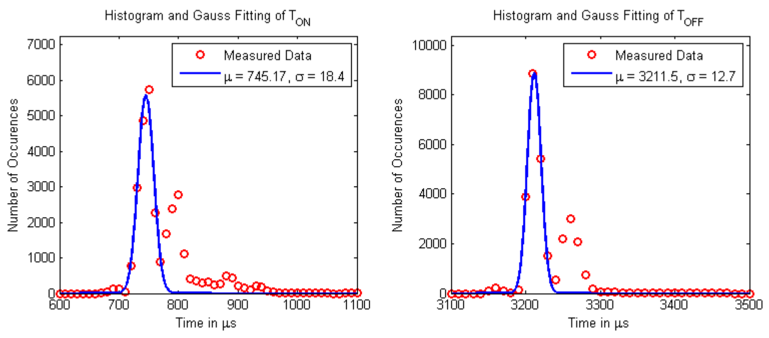


Figure 4.9: Minimum values for T_{ON} and T_{OFF}

	T_{ON}		T_{OFF}	
Run	μ_1	σ_1	μ_0	σ_0
1	745.17	18.4	3211.5	12.7
2	742.96	17.4	3211	12.2
3	742.49	17.6	3211.1	11.9
Mean	743.54	17.8	3211.2	12.267

Table 4.4: Obtained parameters for T_{ON} and T_{OFF}

Observing the obtained results we can see that the minimum possible time for an *ON-OFF* pulse is about 4ms.
After knowing this values the tests with imposed times started. Here we will only make a deep analysis for $b_t = 700\mu s$ and the same considerations about general comportment that were made for SysFS will be made for this approach. Defined durations for H_1 and H_2 are 700 and 1400μs respectively and for information slots defined times are $n{\cdot}700\mu s$, with $n \in \{1, 2, 3, 4\}$.
After gathering the data, MatLab®was used to perform some analysis, in figure 4.10 we present timing histograms and its respective Gaussian, fit for the headers and information slots.

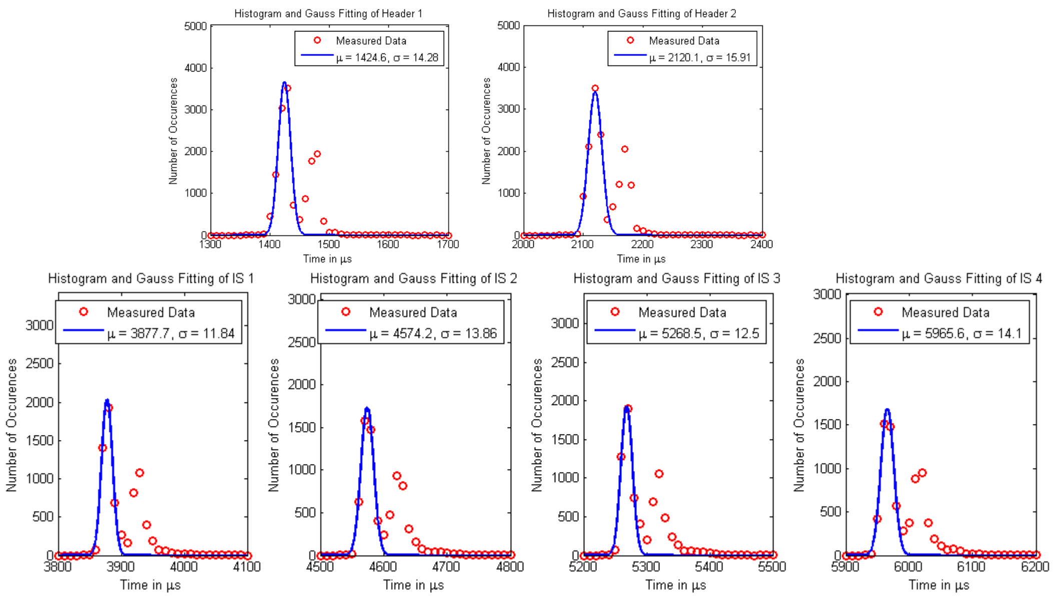


Figure 4.10: Histograms and Gaussian fit for headers and time slots

As we can see the headers and information can be approached by Gaussian random variables with mean value μ_1 for H_1, $T_p+\mu_1$ for H_2 and μ_0 when $n = 1$ and $\mu_0+k\cdot b_t, k = 2, 3, 4$ when $n = 2, 3, 4$ (n represents the number of empty slots). In table 4.5 we have more detailed information about mean and standard deviation for all the tests. The last two lines show the expected value and the difference between this and the obtained mean times.

	Headers				Information Slots							
	H_1		H_2		D=1		D=2		D=3		D=4	
Run	μ_1	σ_1	μ_1	σ_1	μ_0	σ_0	μ_0	σ_0	μ_0	σ_0	μ_0	σ_0
1	1519.3	20.9	2215.4	23.7	3974.6	14.8	4670.1	16.3	5365.9	16	6061.5	15.4
2	1518.3	21.5	2214.3	23.5	3975.5	14.6	4671.1	15.8	5366.8	16.1	6063	15.9
3	1518.5	22.4	2214.7	24	3974.9	14.3	4670.9	15.9	5366.9	16.1	6062.3	15.1
Mean	1518.7	21.6	2214.8	23.7	3975	14.6	4670.7	16	5366.5	16.1	6062.3	15.5
Wanted	700	-	1400	-	700	-	1400	-	2100	-	2800	-
Diff.	818.7	-	814.8	-	3275	-	3270.7	-	3266.5	-	3262.3	-

Table 4.5: Measured uncertainty parameters, with $T_{ON} = T_{OFF} = 700\mu s$ for headers and empty slots

Table 4.5 reveals that the difference between the obtained times and the expected ones is roughly equal to the minimum values for T_{ON} and T_{OFF} previously obtained. To better see this fact figure 4.11 shows the difference between expected and obtained times for headers and information slots. If we look carefully over the picture it confirms what is seen on table 4.5, the difference between the expected and obtained times is very close to the minimum time imposed by the device for flash transitions. It is also possible to see that this difference decreases as nominal times increase.

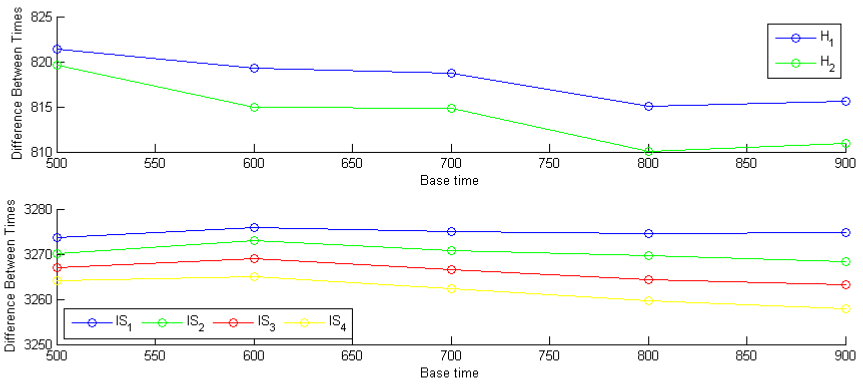


Figure 4.11: Difference between defined and obtained times for $b_t = 500, 600, 700, 800$ and $900\mu s$

Once again the study of uncertainty parameters was done, and the resultant graphics are displayed on figure 4.12

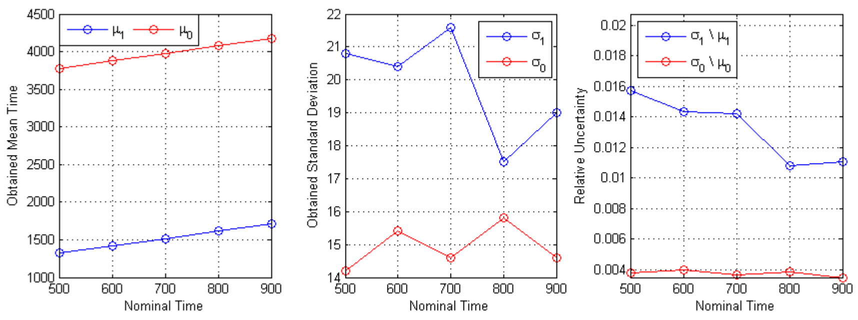


Figure 4.12: Uncertainty parameters for increasing nominal ON and OFF timings (*mean, standard deviation and relative uncertainty*)

For this flash control strategy the relative uncertainty decreases when nominal times increase, which means that higher data rates are more likely to produce errors during transmissions. The standard deviations are stable and do not vary too much with variations of nominal times.
Ones and zeros have different statistical properties and $\Delta_0 = |T_{OFF} - \mu_0|$ is higher than $\Delta_1 = |T_{ON} - \mu_1|$.

4.2.3 Controlling the LED With General Purpose Input Output

GPIO seemed to be the flash control strategy for our project purposes, because it is the one that guarantees lower values for *ON* and *OFF* pulses. To confirm that, several tests were made, as in the other control strategies.
The values presented next were obtained with pulses *ON* and *OFF* with 200 μs of base time. This time definition generated symbols started with *ON* pulses with duration $200\mu s$ or $400\mu s$ (depending on the used header) followed by a series of *OFF* pulses with duration $n{\cdot}200\mu s$, with $n \in \{1, 2, 3, 4\}$. Figures 4.13 and 4.14 show timing histograms fitted to Gaussian shapes for both headers and for all data slots.

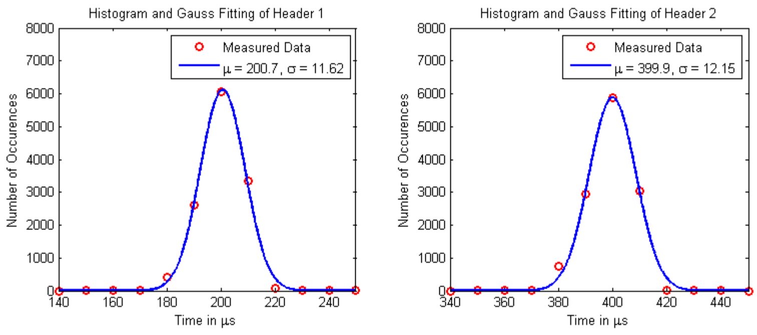


Figure 4.13: Histograms for H_1 and H_2 and the correspondent Gaussian fitting

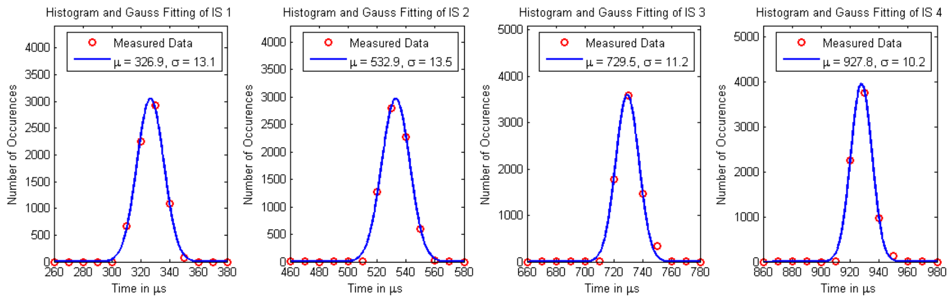


Figure 4.14: Histograms for n = *1, 2, 3 and 4* and the correspondent Gaussian fitting

As it can be seen the duration of headers and data slots behave like Gaussian random variables. Headers are characterized by the temporal mean μ_1 (for H_1) and $T_p + \mu_1$ (for H_2), and have similar standard deviations (σ_1). Data slots have mean (μ_0) and variance (σ_0) that do not depend on the number of empty slots. Results are shown in table 4.6, for three consecutive runs. Ensemble averages for each case and parameter are also given. In the end of the table it is possible to see the values of wanted time as well as the difference between the mean obtained time and the wanted time.

	Headers				Information Slots							
	H_1		H_2		D=1		D=2		D=3		D=4	
Run	μ_1	σ_1	μ_1	σ_1	μ_0	σ_0	μ_0	σ_0	μ_0	σ_0	μ_0	σ_0
1	200.7	11.6	399.9	12.2	326.9	13.1	532.9	13.5	729.5	11.2	927.8	10.2
2	199.8	11.7	410.9	14.6	326.9	12.8	529.9	15.1	727.2	12.6	925.7	12.5
3	199.1	11.7	402.5	17.7	326.9	12.9	534.5	13.3	731.0	12.3	929.6	11.3
Mean	199.9	11.7	404.4	14.8	326.9	12.9	532.4	14	729.2	12	927.7	11.3
Wanted	200	-	400	-	200	-	400	-	600	-	800	-
Diff.	0.1	-	4.4	-	126.9	-	132.4	-	129.2	-	127.7	-

Table 4.6: Uncertainty parameters for $T_{ON} = T_{OFF} = 200\mu s$ for headers and empty slots

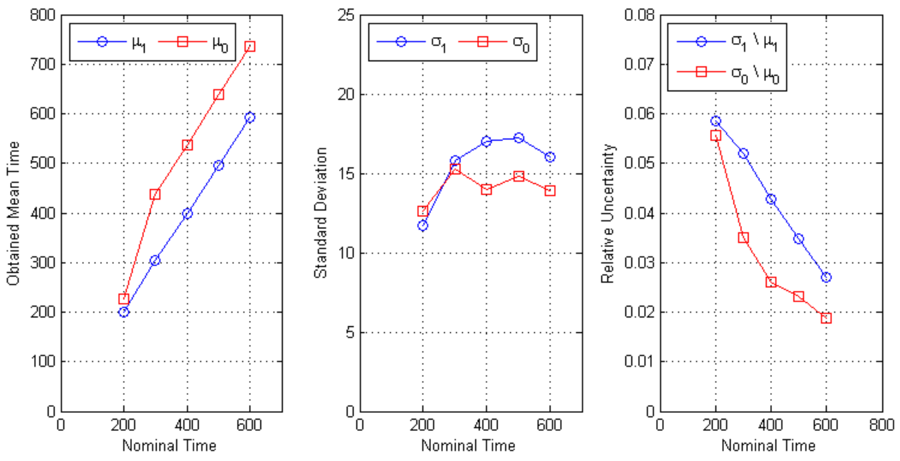


Figure 4.15: Uncertainty parameters for increasing nominal ON and OFF timings (*mean, standard deviation and relative uncertainty*)

The experiment was repeated for increasing nominal times between $200\mu_s$ and $600\mu_s$ in order to perceive the impact of *ON* and *OFF* nominal timings on the uncertainty parameters. figure 4.15 shows absolute and relative uncertainty parameters. Observing the graphics it is possible to see that uncertainty increases almost linearly with the *ON* and *OFF* nominal times while standard deviations are more stable and do not seem to depend on those times. This means that the relative uncertainty $(\sigma\backslash\mu)$increases when the system is set for higher data rates.
Results also show that ones and zeros have similar statistical properties, although $\Delta_0 = |T_{OFF} - \mu_0|$ is a little higher than $\Delta_1 = |T_{ON} - \mu_1|$.

4.3 Timing Uncertainty on DH-PIM

Once we have the ability to produce results with predictable *ON* and *OFF* pulses, some considerations will be taken about time uncertainty.
This analysis will be made assuming a generic DH-PIM system. Each block of M input data bits is mapped to one of possible DH-PIM symbols, as shown in figure 4.16.

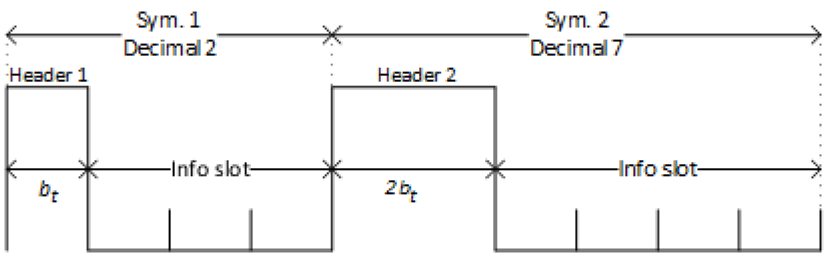


Figure 4.16: DH-PIM Symbol Structure showing H1 and H2

The nominal slot duration is b_t and pulses have amplitude A. For each symbol that is transmitted we have a rectangular pulse with nominal width b_t or $2b_t$, depending on the used header, followed by a sequence of empty slots containing information. The nominal width for the empty slots is equal to nb_t, with $n \in \{1, \ldots, 2^{M-1}\}$. Headers have the role of data decoding and symbol synchronization, but almost all the information is coded in empty slots. From the receiver perspective there is a much higher probability of receiving one '0' than one '1'. This probabilities are given by *p(0)* and *p(1)*, respectively, and can be computed using the following formulas:

$$p(1) = 1/\overline{L}, p(0) = (\overline{L} - 1)/\overline{L}, \text{with } \overline{L} = (2^{M-1} + 3)/2 \tag{4.1}$$

Even in systems where additive noise is negligible, the uncertainty associated with *ON* and *OFF* timings may cause errors. This uncertainty can affect DH-PIM system's performance due to decoding errors in both headers and information slots.
During header reception many errors can occur:

I) H_1 is transmitted but not detected (erasure error)
II) H_1 is transmitted but H_2 is detected (header detection error)
III) H_1 is transmitted but not detected (erasure error)
IV) H_2 is transmitted but H_1 is detected (header detection error)

In information slots errors are caused by an incorrect count of the number of zeros. Considering P_{he} is the header error probability and P_{ie} is the information slot's error probability, the symbol error probability is given by:

$$\begin{gathered} P_{se} = p(1)P_{he} + p(0)P_{ie} = \\ p(1)p(H_1)p(0|H_1) + p(H_2|H_1)] + p(H_2)[p(0|H_2) + p(H_1|H_2)]] + p(0)p(z_n)p(z_m|z_m) \end{gathered} \tag{4.2}$$

Where $p(H_i)$ stands for the probability of transmitting H_i, $p(x|H_i))$ is the conditional probability of detecting x when H_i was transmitted, $p(z_n)$ is the probability of transmitting n zero slots, and $p(z_m|z_n)$ is the probability of detecting m zero slots when n zero slots were transmitted.
Next some considerations about the impact of time uncertainty on header and information slots will be made separately. To do that the probability of detecting an empty slot when H_2 is transmitted will be considered to be vanishingly small, so that $p(0|H_2) \approx 0$. As the impact of Additive White Gaussian Noise will not have a significant impact on this system's performance, it will not be discussed here.

4.3.1 Impact on Header Decoding

According to experimental results, the uncertainty when generating a sequence of '1's or '0's will be considered not to depend on the absolute sequence time duration. So, H_1 pulses are considered to have random width equal to τ_1, H_2 pulses are considered to have random width equal to $\tau_1 + b_t$, and empty slots have random width equal to $\tau_0 + (n-1)2b_t$. It is considered that τ_0 and τ_1 are Gaussian variables with mean (μ_0 and mu_1) and standard deviation (σ_0 and σ_1).The absolute time difference between mean values and the correspondent nominal times will be therefore represented by $\Delta_0 = |\mu_0 - 2b_t|$ and $\Delta_1 = |\mu_1 - b_t|$.
To evaluate the uncertainty impact on header detection, it is necessary to consider four different situations:

I) H_1 is transmitted with $\mu_1 < b_t$
II) H_1 is transmitted with $\mu_1 > b_t$
III) H_2 is transmitted with $\mu_1 < b_t$
IV) H_2 is transmitted with $\mu_1 > b_t$

Assuming that μ_1 can be smaller or larger than b_t with equal probabilities and that H_1 and H_2 also have equal probabilities of occurring,t is possible to write the header error probability

(P_{he}) as:

$$\begin{aligned} P_{he} = \frac{1}{2}\left[Q\left(\frac{A((b_t/2)-\Delta_1)}{\sigma_1}\right) + Q\left(\frac{A((b_t/2)+\Delta_1)}{\sigma_1}\right)\right] + \frac{1}{2}\left[\frac{1}{2}Q\left(\frac{A((b_t/2)-\Delta_1)}{\sigma_1}\right)\right] \\ = \frac{3}{4}Q\left(\frac{A((b_t/2)-\Delta_1)}{\sigma_1}\right) + \frac{1}{2}Q\left(\frac{A((b_t/2)+\Delta_1)}{\sigma_1}\right) \end{aligned} \tag{4.3}$$

where A represents the amplitude if the received signal.

4.3.2 Impact on Information Decoding

Uncertainty in the measurements of information slot's duration depends on:

I) the uncertainty associated with the time reference used in the measurement
II) the uncertainty associated with the empty slot's duration

If the time reference used is perfect there will not be synchronization error and the counter will measure a Gaussian random time interval equal to $t_i = (n-1)2b_t + \tau_0$, with mean $\mu_i = (n-1)2b_t \pm \mu_0$ and standard deviation $\sigma_i = \sigma_0$.
The value for zero error probability can be computed as:

$$P_{e0|\overline{SE}} = Q\left(\frac{b_t - \Delta_0}{\sigma_0}\right) + Q\left(\frac{b_t + \Delta_0}{\sigma_0}\right) \tag{4.4}$$

And the information error probability (P_{ie}) is given by:

$$P_{ie} = p(SE)P_{e0|SE} + p(\overline{SE})P_{e0|\overline{SE}} = \frac{1}{4}P_{e0|SE} + \frac{3}{4}P_{e0|\overline{SE}} \tag{4.5}$$

Time uncertainty is better explained in[47], present in the appendixes of this thesis.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

When analyzing the obtained values it is important to have in mind that these results are for three runs only and were obtained with a particular transmitter. Different transmitters (different smartphones, with different flash control capabilities) may exhibit different absolute and relative uncertainty parameters. Thus, the receiver must cope with significant timing uncertainty. In fact, the receiver will never be adequately matched to the transmitted pulse, which depends on the flash controllability of each used smartphone.
With the observed results we can say that this smartphone has some potential for VLC uses. It has good performance using GPIO, but for systems that do not require high transmission rate any control interface could be used.
Since the relative uncertainty decreases with the increase of data rate for SysFS flash control strategy, SysFS might be better than IOCTL to transmit data.

5.2 Future Work

- Implement one communication system based on this results
- Find out the viability of the communication system, based on link speed and transmission error rate
- Make one application that could be used on several smartphones
- Test the application in real scenarios

Bibliography

[1] Cisco, "Cisco visual networking index : Global mobile data traffic forecast update , 2013 - 2018," 2014.

[2] David Goldman, "Sorry, america: Your wireless airwaves are full," February 2012, [Accessed 25 November 2014]. [Online]. Available: http://money.cnn.com/2012/02/21/technology/spectrum_crunch/

[3] H. Burchardt, N. Serafimovski, D. Tsonev, S. Videv, and H. Haas, "Vlc: Beyond point-to-point communication," *Communications Magazine, IEEE*, vol. 52, no. 7, pp. 98–105, July 2014.

[4] M. Kavehrad, "Sustainable energy-efficient wireless applications using light," *Communications Magazine, IEEE*, vol. 48, no. 12, pp. 66–73, December 2010.

[5] M. Kavehrad and M. Chowdhury, "An archipelago of high-bandwidth islands by optical wireless systems - a solution to the usa wireless airwaves spectrum crunch," in *Photonics Society Summer Topical Meeting Series, 2012 IEEE*, July 2012, pp. 92–93.

[6] Faeza, "The electro magnetic spectrum," [Accessed 22 October 2014]. [Online]. Available: http://faezams.edublogs.org/6scg/

[7] Tsonev, Dobroslav; Chun, Hyunchae; Rajbhandari, Sujan; McKendry, Jonathan J. D.; Videv, Stefan; Gu, Erdan; Haji, Mohsin; Watson, Scott; Kelly, Anthony E.; Faulkner, Grahame; Dawson, Martin D.; Haas, Harald; O'Brien, Dominic, "A 3-gb/s single-led ofdm-based wireless vlc link using a gallium nitride μled," *IEEE Photonics Technology Letters*, vol. 26, 04 2014.

[8] M. S. Rahman, M. M. Haque, and K.-D. Kim, "Indoor positioning by led visible light communication and image sensors," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 1, no. 2, pp. 161–170, 2011.

[9] N. Kumar, L. A. Nero, and R. L. Aguiar, "Visible light communication for advanced driver assistant systems," 2009.

[10] G. Corbellini, K. Aksit, S. Schmid, S. Mangold, and T. Gross, "Connecting networks of toys and smartphones with visible light communication," *Communications Magazine, IEEE*, vol. 52, no. 7, pp. 72–78, July 2014.

[11] World of LEDs, "History of leds," [Accessed 25 October 2014]. [Online]. Available: http://www.worldofleds.co.uk/history-of-leds.htm

[12] Pure VLC, "Differences between radio & visible light communications: A technical guide," [Accessed 28 October 2014]. [Online]. Available: http://www.purevlc.com/pureVLC_RadioTech_v1.0.pdf

[13] M. Lahanas, "Ancient greek communication methods," [Accessed 25 November 2014]. [Online]. Available: http://www.hellenicaworld.com/Greece/Technology/en/Communication.html

[14] Michael Lahanas, "Ancient greek communication methods," [Accessed 2 November 2014]. [Online]. Available: http://www.hellenicaworld.com/Greece/Technology/en/Communication.html

[15] C. H. Sterling, *Military Communications: From Ancient Times to the 21st Century.* ABC-CLIO, 2008.

[16] A. G. Bell, "The photophone," *Science*, vol. 1, no. 11, pp. pp. 130–134, 1880. [Online]. Available: http://www.jstor.org/stable/2900889

[17] M. Kavehrad, "Sustainable energy-efficient wireless applications using light," *Communications Magazine, IEEE*, vol. 48, no. 12, pp. 66–73, 2010.

[18] Zenith, "Operating guide and technical manual to help you enjoy your new zenith flash-matic remote control."

[19] Erik V, "Nintendo light telephone," 2011, [Accessed 25 November 2014]. [Online]. Available: http://blog.beforemario.com/2011/06/nintendo-light-telephone-lt-1971.html

[20] Bouchet, O; O'Brien, D; Kamalakis, T; Nerreter, S; Vucíc, J; Walewski, J. W., "Deliverable d4.6a smart wireless optics (swo) – final evaluation report," [Accessed 2 November 2014]. [Online]. Available: http://www.ict-omega.eu/fileadmin/documents/deliverables/Omega_D4.6a.pdf

[21] Vucic, J. and Kottke, C. and Nerreter, S. and Langer, K. D. and Walewski, J.W., "513 mbit/s visible light communications link based on dmt-modulation of a white led," *Lightwave Technology, Journal of*, vol. 28, no. 24, pp. 3512–3518, Dec 2010.

[22] Vucic, J. Kottke, C. Habel, K. and Langer, K. D., "803 mbit/s visible light wdm link based on dmt modulation of a single rgb led luminary," in *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference*, March 2011, pp. 1–3.

[23] Stefan, I.; Elgala, H.; Mesleh, R.; O'Brien, D. and Haas, H., "Optical wireless ofdm system on fpga: Study of led nonlinearity effects," in *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, May 2011, pp. 1–5.

[24] H. E. Raed Mesleh and H. Haas, "Optical spatial modulation," *Journal of Optical Communications and Networks*, vol. 3, no. 3, pp. 234–244, March 2011.

[25] Mesleh, R.; Elgala, H. and Haas, H., “Optical spatial modulation,” *Optical Communications and Networking, IEEE/OSA Journal of*, vol. 3, no. 3, pp. 234–244, March 2011.

[26] S. Haruyama, “Visible light communications: Recent activities in japan,” *Smart Spaces: A Smart Lighting ERC Industry - Academia Day at BU Photonics Center*, p. 49, February 2011.

[27] M. Doniec, I. Vasilescu, M. Chitre, C. Detweiler, M. Hoffmann-Kuhnt, and D. Rus, “Aquaoptical: A lightweight device for high-rate long-range underwater point-to-point communication,” in *OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future: Global and Local Challenges*, Oct 2009, pp. 1–6.

[28] F. Hanson and S. Radic, “High bandwidth underwater optical communication,” *Applied optics*, vol. 47, no. 2, pp. 277–283, 2008.

[29] P. Dietz, W. Yerazunis, and D. Leigh, “Very low-cost sensing and communication using bidirectional leds,” in *UbiComp 2003: Ubiquitous Computing*. Springer, 2003, pp. 175–191.

[30] S. Schmid, G. Corbellini, S. Mangold, and T. R. Gross, “An led-to-led visible light communication system with software-based synchronization,” in *Globecom Workshops (GC Wkshps), 2012 IEEE*. IEEE, 2012, pp. 1264–1268.

[31] S. Schmid, G. Corbellini, S. Mangold, and T. R. Gross", “Led-to-led visible light communication networks,” in *Proceedings of the fourteenth ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2013, pp. 1–10.

[32] G. Corbellini, K. Aksit, S. Schmid, S. Mangold, and T. Gross, “Connecting networks of toys and smartphones with visible light communication,” *Communications Magazine, IEEE*, vol. 52, no. 7, pp. 72–78, July 2014.

[33] L. Casio Computer Co., “Vlc system using smartphones,” 2012, [Accessed 22 October 2014]. [Online]. Available: http://world.casio.com/news/2012/0115_VisibleLightcomm/

[34] Google, “Google wallet,” 2012, [Accessed 27 November 2014]. [Online]. Available: https://www.google.com/wallet/

[35] S. Up, “Square reader,” [Accessed 27 November 2014]. [Online]. Available: https://squareup.com/reader

[36] M. M. Galal, A. A. El Aziz, H. a. Fayed, and M. H. Aly, “High speed data transmission over a visible light link employing smartphones Xenon flashlight as a replacement of magnetic cards,” *2013 High Capacity Optical Networks and Emerging/Enabling Technologies*, pp. 157–160, Dec. 2013.

[37] Skyworks, “AAT1271 datasheet - 1.5A Step-Up Current Regulator for Flash LEDs,” 2012.

[38] N. Corporation, “Tegra 3 Technical Reference Manual version 1.0,” 2011.

[39] Oracle, "Learn About Java Technology," 2014. [Online]. Available: https://www.java.com/en/about/

[40] F. Liu, *Android Native Development Kit Cookbook.* Packt Publishing Ltd, 2013.

[41] P. Mochel, "The sysfs filesystem," in *Linux Symposium*, 2005, p. 313.

[42] "Ieee standard for local and metropolitan area networks–part 15.7: Short-range wireless optical communication using visible light," *IEEE Std 802.15.7-2011*, pp. 1–309, Sept 2011.

[43] Z. Ghassemlooy, W. Popoola, and S. Rajbhandari, *Optical wireless communications: system and channel modelling with Matlab®.* CRC Press, 2012.

[44] HTC, "Htc one x," 2014, [Accessed 7 September 2014]. [Online]. Available: https://www.htc.com/us/smartphones/htc-one-x/

[45] Nvidia, "Tegra mobile processors - tegra 3," 2014, [Accessed 11 October 2014]. [Online]. Available: http://www.nvidia.com/object/tegra-3-processor.html

[46] Arduino, "Arduino Uno R3," 2014, [Accessed 29 September 2014]. [Online]. Available: http://arduino.cc/en/Main/arduinoBoardUno

[47] Gonçalo Farias, João Paulo Barraca, Mónica Figueiredo, Luis Nero Alves, "Using the smartphone flash for visible light communication purposes: Limitations and design tradeoffs," *Submited to IET Optoelectronics*, 2014.

Appendices

Appendix A

Emitter Control

Here a generic Java method will be presented, followed by some Java snippets showing specific options for each flash control strategy. This options are related with the kind of kernel objects we need to guarantee access.
After the Java code the native *C* methods used to control the flash light will be displayed.

A.1 Java Code

A.1.1 Generic Java Method

```
import java.io.DataOutputStream;
import java.io.IOException;
import android.app.Activity;
import android.content.Context;
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnKeyListener;
import android.widget.Button;


public class MainActivity extends Activity {

  Button send;
  Process sh;

  static {
    System.loadLibrary("native-methods");
  }

  public native void flashFlicker(); //Native methods that
   control flash
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    android.os.Process
        .setThreadPriority(android.os.Process.
   THREAD_PRIORITY_URGENT_AUDIO); //Increase thread priority

    try {
      sh = Runtime.getRuntime().exec("su"); //Guarantee root
   privileges, equal for every approach
      /*Here we need to define some options, according
      to the used flash control strategy. This options
      will be shown next.*/

    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }

    send = (Button) findViewById(R.id.myBt); //Create Button
    send.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {

        flashFlicker(); //Call native methods

        try {
          Uri notification = RingtoneManager
              .getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
    //Sound notification when test is done
          Ringtone r = RingtoneManager.getRingtone(
              getApplicationContext(), notification);
          r.play();
        } catch (Exception e) {
          e.printStackTrace();
        }
      }
    });

  }
}
```

A.1.2 IOCTL Specific Options

```
try {
  sh = Runtime.getRuntime().exec("su");
  DataOutputStream os = new DataOutputStream(sh.getOutputStream
   ());
  //Unlock AAT1271 chip for reading and writing
  os.writeBytes("chmod 777 /dev/aat1271\n");
  os.flush();
  os.writeBytes("exit\n");
  os.flush();
  sh.waitFor();
} catch (IOException e) {
  e.printStackTrace();
} catch (InterruptedException e) {
  e.printStackTrace();
}
```

A.1.3 GPIO Specific Options

```
try {
  sh = Runtime.getRuntime().exec("su");
  DataOutputStream os = new DataOutputStream(sh.getOutputStream
   ());
  //Unlock GPIOs that control flash light for torch mode and
   flash mode
  os.writeBytes("chmod 777 /sys/class/gpio/gpio219/value\n");
  os.flush();
  os.writeBytes("chmod 777 /sys/class/gpio/gpio137/value\n");
  os.flush();
  os.writeBytes("exit\n");
  os.flush();
  sh.waitFor();
} catch (IOException e) {
  e.printStackTrace();
} catch (InterruptedException e) {
  e.printStackTrace();
}
```

A.2 Native methods

A.2.1 SysFS

```
#include <jni.h>
#include <sequence-jni.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <sys/types.h>
#include <pthread.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <sys/ioctl.h>

#define BASE_TON 700  //Base time for ON pulses
```

```
#define BASE_TOFF 700 //Base time for OFF pulses
#define SYMS 30000    //Number of symbols to send

int sTs;
struct time_param {
  unsigned int param_on;
  unsigned int param_off;
};

void transmitter(struct time_param mTimes[]) {
  int i;
  FILE *flash;
  for (i = 0; i < sTs; i++) {
    flash = fopen("/sys/class/leds/flashlight/brightness", "w");
    fprintf(flash, "1\n");
    fclose(flash);
    usleep(mTimes[i].param_on);
    flash = fopen("/sys/class/leds/flashlight/brightness", "w");
    fprintf(flash, "0\n");
    fclose(flash);
    usleep(mTimes[i].param_off);
  }
}

JNIEXPORT void JNICALL
   Java_com_example_genericJavaMethod_MainActivity_flashFlicker(
    JNIEnv *env, jobject thiz) {

  setpriority(PRIO_PROCESS, 0, -16); //Process priority

  int i, symbol;
  sTs = SYMS;
  struct time_param mTimes[sTs];
  srand(time(NULL));
  i = 0;

  while (i < sTs) {
    symbol = rand() % 8;
    switch (symbol) {
    case 0:
    case 1:
    case 2:
    case 3:
      mTimes[i].param_on = BASE_TON;
      mTimes[i].param_off = BASE_TOFF * (symbol + 1);
      break;
    case 4:
    case 5:
    case 6:
```

```
    case 7:
      mTimes[i].param_on = BASE_TON * 2;
      mTimes[i].param_off = BASE_TOFF * (symbol - 3);
      break;
    }
    i++;
  }
  transmitter(mTimes);
}
```

A.2.2 IOCTL

```
#include <jni.h>
#include <Flash.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <android/log.h>
#include <sys/stat.h>
#include <sys/syscall.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/resource.h>
#include <sys/time.h>

#define BASE_TON 700  //Base time for ON pulses
#define BASE_TOFF 700 //Base time for OFF pulses
#define SYMS 30000    //Number of symbols to send

struct nvodmimager_param {
  int param;
  unsigned int sizeofvalue;
  unsigned long *p_value;
};

struct time_param {
  unsigned int param_on;
  unsigned int param_off;
};

int dev, sTs;
uint32_t value;
struct nvodmimager_param led_mode;

//IOCTL command to write to AAT1271
#define AAT1271_IOCTL_PARAM_WR _IOW('o', 4, struct
   nvodmimager_param)
```

```
void transmitter(struct time_param mTimes[]) {
  memset(&led_mode, 0, sizeof(led_mode));
  led_mode.param = 10;
  led_mode.sizeofvalue = sizeof(value);
  led_mode.p_value = &value;

  int i;
  for (i = 0; i < sTs; i++) {
    value = 1;
    ioctl(dev, AAT1271_IOCTL_PARAM_WR, &led_mode);
    usleep(mTimes[i].param_on);
    value = 0;
    ioctl(dev, AAT1271_IOCTL_PARAM_WR, &led_mode);
    usleep(mTimes[i].param_off);
  }
}

JNIEXPORT void JNICALL
    Java_com_example_genericJavaMethod_MainActivity_flashFlicker(
      JNIEnv *env, jobject thiz) {

  setpriority(PRIO_PROCESS, 0, -16); //Process priority

  dev = open("/dev/aat1271", O_RDWR);
  sTs = SYMS;
  int i, symbol;
  struct time_param mTimes[sTs];

  srand(time(NULL));

  i = 0;
  memset(&mTimes, 0, sizeof(mTimes));
  while (i < sTs) {
    symbol = rand() % 8;

    switch (symbol) {
    case 0:
    case 1:
    case 2:
    case 3:
      mTimes[i].param_on = BASE_TON;
      mTimes[i].param_off = BASE_TOFF * (symbol + 1);
      break;
    case 4:
    case 5:
    case 6:
    case 7:
      mTimes[i].param_on = BASE_TON*2;
      mTimes[i].param_off = BASE_TOFF * (symbol - 3);
      break;
    }
    i++;
  }
  transmitter(mTimes);
}
```

A.2.3 GPIO

```
#include <jni.h>
#include <flash-gpio.h>
#include <android/log.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>
#include <pthread.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <sys/ioctl.h>

//TEGRA_GPIO_PBB1
#define FLASH "/sys/class/gpio/gpio219/value"
//TEGRA_GPIO_PR1
#define TORCH "/sys/class/gpio/gpio137/value"

int f;
#define sTs 30000 //Symbols to simulate
#define BASE_TIME 200 //Base time for DH-PIM
char on = '1';
char off = '0';
static void gpio_direction_output(const char* dev, char value) {
  int f = open(dev, O_WRONLY);
  if (f < 0) {
    return;
  }
  write(f, &value, 1);
  close(f);
}
static void flashlight_turn_off(void) {
  gpio_direction_output(FLASH, '0');
  gpio_direction_output(TORCH, '0');
}
struct time_param {
  unsigned int param_on;
  unsigned int param_off;
};
void transmitter(struct time_param mTimes[]) {
  int i;
  for (i = 0; i < sTs; i++) {
    write(f,&on,1);
    usleep(mTimes[i].param_on);
    write(f,&off,1);
    usleep(mTimes[i].param_off);
  }
}
JNIEXPORT void JNICALL
   Java_com_example_genericJavaMethod_MainActivity_flashFlicker(
   JNIEnv *env,
    jobject thiz) {
  setpriority(PRIO_PROCESS, 0, -16); //Process priority
  f = open(FLASH, O_WRONLY);
  int symbol, i;
  struct time_param mTimes[sTs];
  srand(time(NULL));
    i = 0;
    while (i < sTs) {
      symbol = rand() % 8;
      switch (symbol) {
      case 0:
      case 1:
      case 2:
      case 3:
        mTimes[i].param_on = BASE_TIME;
        mTimes[i].param_off = BASE_TIME * (symbol + 1);
        break;
      case 4:
      case 5:
      case 6:
      case 7:
        mTimes[i].param_on = BASE_TIME * 2;
        mTimes[i].param_off = BASE_TIME * (symbol - 3);
        break;
      }
      i++;
    }
    transmitter(mTimes);
  write(f, &off, 1); close(f);
  }
```

Appendix B

Arduino Control

Here the Arduino program responsible for acquiring data will be presented.

```
#define SAMPLES 500  //Number of Samples
#define PIN_IN A2  //Input pin
unsigned short valores[SAMPLES];  //array for value storage

void setup()
{
  pinMode(PIN_IN,INPUT);
  for(int i=0;i<SAMPLES;i++)
    valores[i]=0;
  Serial.begin(9600); //start serial communication
  Serial.println("OK");
}
int j=0; int m = 0; unsigned short num = 0; long a = 0;
void loop()
{
  short v = pulseIn(PIN_IN, HIGH)/10;
  //for measure TOFF must be short v = pulseIn(PIN_IN, LOW)/10;
  if(v >= SAMPLES)
   v = SAMPLES-1;
  if(num > 0 && (v == 0 || m == 0xFFFF))
  {
    for(int i = 0; i < SAMPLES ; i++)
    {
      Serial.println(String(i)+"␣:␣"+String(valores[i]));
      valores[i] = 0;
    }
    Serial.println("Total:␣"+String(num));
    m = 0;  num = 0;
  }
  if(v == 0)
    return;
  num++;  valores[v]++;
  if(valores[v] > m)
    m = valores[v];
}
```

Appendix C

Submitted Paper

Using the Smartphone Flash for Visible Light Communication Purposes: Limitations and Design Tradeoffs

Gonçalo Farias[1], João Paulo Barraca[1,2], Mónica Figueiredo[2,3], Luis Nero Alves[1,2]
[1]Universidade de Aveiro, Departamento de Eletrónica Telecomunicações e Informática, Aveiro, Portugal.
[2]Instituto de Telecomunicações, Aveiro, Portugal.
[3]Instituto Politécnico de Leiria, Escola Superior de Tecnologia e Gestão, Leiria, Portugal.
goncalofarias@ua.pt, jpbarraca@ua.pt, monica.figueiredo@ipleiria.pt, nero@ua.pt

Submitted to IET Optoelectronics

Abstract

There are several design tradeoffs regarding visible light communication systems supported on currently available smartphones. Current smartphones do not include means for optical communications, rather than a CCD camera and a flash LED. Neither of these devices is conceived for communication purposes, nevertheless several attempts have been made in other to extend their functionalities. This paper discusses the usage of the flash LED as a transmitting device in low bit rate applications. For analysis purposes it was used an HTC One X with Android 4.3.1 Operating System (OS). Android is open source, thus enabling several flash control possibilities, from an application level down to the hardware access level. Achieved results reveal that, communication means using the flash LED are limited on data rate and link distance. It was also disclosed that the best approach to mitigate data rate problems is to rely on DHPIM modulation formats, which are able to improve data rate due to their non-uniform symbol timings.

Keywords: Optical Communications, Data Communications, Light Emitting Diodes.

1 Introduction

Modern smartphones do not support optical wireless communications means. Previous generations included a dedicated IrDA port able to provide an easy access communication link with neighbor equipment. This is in general not present in most smartphones in the market. Thus, the optical communications means available on these devices are restricted to the camera and flash LED. These devices can exploited for communications on the framework of the IEEE Standard 802.15.7 [1], tackling short range communication employing visible light communication concepts. However, in order to address this service integration, some challenges posed by the usage of smartphone's cameras and flash LEDs need to be considered.

Camera and flash LED peripherals were not intended for communication purposes. Nevertheless, it is possible to exploit their usage as receiving and transmitting devices [2-10]. The camera is limited in terms of acquisition time to acquisition speeds around 30fps (60fps in some devices), which poses a direct limitation on the receiving data rates. However, being a multiple input device (pixels), it is possible to exploit more sophisticated modulation formats able to take improve data rates. Previous contributions available on the literature focus on visible content bidirectional communication links [2, 3]. In 2012 Casio unveiled visible light communication concepts to allow augmented reality scenarios involving the camera and flash LED of smartphone [4]. On the other hand, the flash LED is normally used as flashlight for the camera or even, as general purpose lighting device. Normal applications do not require fast switching devices, but rather, power devices. Still, power drivers offer reasonable fast response times, potentially enabling several VLC scenarios, after the system is correctly characterized. Thus using the flash LED for communication purposes poses some challenging considerations, normally not very well specified in the literature. Adding to this, the wide variety of smartphones models running different OS and relying on different hardware platforms, contribute to make the picture more complex.

Previous contributions on this topic focused on application oriented cases, highlighting applications such as secure cash transactions [5, 6], communication with interactive tabletop equipment [7], and electronic shelf labelling [8-10]. The first mentions to the use of visible light to communicate with electronic shelf labelling equipment were reported in [8, 9]. There, the authors discuss the possibility of wireless optical access to the devices inside the store for programming purposes. In [10] the authors propose the usage of the LED flash as a means to communicate with electronic shelf labelling devices to enable fast feedback loops between consumers and retailers. This is seen as a promising technology given the current trends on massive advertising campaigns. It also copes with the trends on these kind of devices, which are currently seen as part of sensor networks, thus extend their normal usage to other important scenarios (such as consumer habits profiling) [11]. Other possibilities are directed to the toys market. Disney research labs are currently investing on visible light communications products linking toy control to smartphones [12, 13]. Other possible applications may resort to secure key lockers or general communication interface. On another edge, high speed short range communication links are seen as a relevant technology for fast download/upload of media content. The Infra Red Data Association (IrDA) is advancing technology able to tackle communication speeds on the range of 10Gbps, seen as ideal for instantaneous data transfer in point-to-point links [14, 15].

Though promising, these application oriented contributions are not able to address the limitations posed by both the camera and the flash LED when exploited as communication devices. This paper addresses the limitations and tradeoffs posed on the usage a LED flash, in a current smartphone, as transmitting device. In order to identify the sources of data rate limitations, a detailed study on different methodologies to control the flash LED is presented. As it will be disclosed, limitations are mostly posed by the software and OS (Operating System) of the smartphone rather than the LED itself. It should be highlighted that normal battery savings and temperature control are high priority tasks in a smartphone. Thus the software assumes by default conservative timings for short pulses on the LED flash, typically restricting how fast two consecutive pulses may appear. The net result affects pulse timing definitions, on both the average and standard deviation, with obvious impairments for communication.

Depending on the control methodology, different pulse statistics can be expected. This is linked to the normal operation of the smartphone, where communication tasks and network search have higher priority than controlling the flash LED timings. For the aforementioned applications, the communication link is mostly line-of-sight and short range, thus it is reasonable to assume that amplitude induced noise has negligible impact on performance.

Clearly, this will be the case where nearby lighting sources induce only white noise on the receiver. For these cases, communication performance will be mostly affected by the timing statistics of the transmitter, which appear to the receiver as uncertainty on the signal edges.

The present work proposes the usage of DH-PIM [16, 21] as a means to fight data rate limitations. DH-PIM modulation format has two relevant characteristics for the application: i) it is a pulsed modulation scheme with reduced average duty-cycle, which is important considering both LED temperature and energy; ii) the symbol duration is not uniform, which implies a considerable gain on the achievable data rates, obviously dependent on the symbol composition. Achieved results for 8DH-PIM disclose that performance depends greatly on the ability to accurately generate ON and OFF pulses with the smartphone flash, as time uncertainty is the major impairment to data throughput in such systems. Also, uncertainty is shown to increase for higher data rates. However, because each smartphone has shown to exhibit a significant static uncertainty, higher throughputs may be attained by resorting to dynamically adjusted receivers.

The paper is divided into five sections. Section 2 discusses the available flash LED control schemes in a HTC One X smartphone with Android 4.3.1 OS [17], and some experimental results that illustrate the timing uncertainty problem in this system. Section 3, presents an in deep analysis to evaluate the impact of timing uncertainty in DH-PIM receiver error performance. Section 4 discusses the theoretical results regarding the symbol error probability induced by timing uncertainty. Finally, section 5 draws the final conclusions.

2 Flash Control Strategies

It is common for smartphones to have a high power LED, which is mainly used to provide illumination to the camera, and to serve as a low power flashlight. As discussed, the LED can be used for communications, instead of other technologies such as RFID and NFC. In particular, because it enables more devices to communicate under short distance scenarios [18]. However, the LED flash sub-system was not designed for this. Applications trigger the LED for a short instant, in order to enable capturing a photo in dark environments or for extended periods in order to provide a constant illumination source. The main characteristic of smartphones, which enables the use of the LED as a communication interface, is the fact that applications can have direct access to the LED state. In some smartphones, it is also possible to have access to the intensity of the LED, with up to 4 intensity levels being available. In the smartphone used in this work, the LED can be safely driven with currents up to 700mA, enabling time and amplitude modulation of signals. In the smartphone we used, an AAT1271 chip [19], exposed through the Tegra30 GPIO chip [20], drives the LED. On top of this, the Linux kernel will use the AAT1271 driver software, and a flashlight driver. Each layer provides a better abstraction and functionality over the basic function of the LED. The following subsections discuss the available methods for flash control, as well as their experimental characterization.

2.1 Flash LED Control Methodologies

While applications have access to the LED, there are multiple interfaces through which control can be applied. Interfaces that may have be implemented at a software level, having multiple abstraction layers between the LED driver and the application. Therefore, it is vital to describe and characterize the different flash control strategies that can be used by applications. Considering a generic NVidia Tegra30 based device running the Android OS, four different approaches are readily available, and were considered. These approaches are identified in Figure 1 and identified by numbers 1 to 4.

The first interface is through the Camera class provided by the Android Java SDK. This is the most readily used interface as it is high level and well documented. It provides a common interface to applications, enabling development of applications that are able to control the LED of any Android smartphone. The Camera class abstracts access to the hardware and applications do not need to have any knowledge about the smartphone. The main drawback is that this is a Java interface, available for applications that run inside a Java Virtual Machine (JVM). While the use of a JVM is great for portability, interaction with the world outside the JVM will impose synchronization mechanisms and added context switching. Moreover, Java is not a top performance programming language, at least not when comparing with carefully optimized C code.

The second interface is through the sys filesystem (SysFS), a mechanism provided by the Linux kernel, allowing swift communication between applications and drivers residing in kernel space. SysFS is a file system that is typically mounted in /sys and exposes interfaces for most drivers. In the case of the flash LED in a smartphone, the Linux kernel will expose a directory /sys/class/leds/flashlight. A special file named brightness, present at this directory, allows applications to set the brightness level of the LED by writing an integer to it. Writing 0 will turn the LED off while other values will set the LED to a different brightness level. Because only a few levels are available, only a few values are allowed, and some may provide complex behavior, such as turning on the LED for 0.5s and then turning it off. A clear advantage is that a developer can use a wide range of languages to interact with the LED. Another advantage is that control can be enforced outside the JVM, through Java Native Interface (JNI) integrated C functions, which potentially provides lower latency and jitter by bypassing the JVM. However, controlling the LED of a specific phone will imply knowing which brightness values are available. Moreover, using the SysFS interface implies that multiple software layers are still present, such as the Virtual File System (VFS), the Flashlight driver, and the LED software driver.

The third interface is similar to the previous, but makes use of the IOCTL mechanism of the Linux kernel. The level of control is similar, but Input Output ConTroL (IOCTL) calls are directed to the Flashlight driver, bypassing the VFS. Moreover, access to the brightness control file requires opening the file, writing a value and closing the file, as the value is only committed when the file is closed. Each of these operations requires at least one IOCTL, totaling three interactions between the control application and the kernel.

Finally, the forth interface bypasses more layers and is the GPIO interface. Accessing a GPIO provides almost unrestricted control as it allows sending commands directly to the LED driver chip, through the GPIO chip. Such access can be obtained either through the SysFS interface or through direct access to the mapped memory space. GPIO access is at the same time, the most flexible and most specific method. It provides great access, but requires deep knowledge about how hardware is structured in each specific smartphone. This makes such solution important for specific applications, but not so relevant for an application aiming to be generic and support a wide range of devices. Fortunately, while numbers identifies GPIOs, the Linux kernel provide human readable alias, enabling dynamic discovery of the relevant GPIO to use to drive the LED state.

From all the above methodologies, last three were tested, as they are the most promising in terms of performance. The first was discarded because from our experience, and previous experiments, it is known that latency values were a magnitude higher than the other cases. Achieved results demonstrate that with the second and third methodologies, although there is a fine control of the LED, there is limited control on LED pulse duration. Both the ON and OFF durations have a minimum value set by default, and enforced by the driver. For the smartphone under test, these times were 0.8ms for the minimum ON pulse and 3.2ms for the minimum OFF pulse which put a limit of 4ms for the minimum ON-OFF pulse occurrence. In practice, the driver tries to set a delay a little higher than 1ms, but as it is software timing, in a concurrent OS, using the usleep function, some uncertainty is always present. Furthermore, it was ascertained that the timing uncertainty using the second method, was worse than for the third method, with standard deviations reaching the order of 300ms. This was expected, but it would further reduce the achievable data rates, even with DH-PIM modulation schemes. The forth method revealed superior performance with better timing control of the pulse durations, allowing switching the LED with pulses below 100µs.

2.2 Timing Uncertainty in Flash-Based Communications

Before conducting a formal system analysis, it is necessary to evaluate first the uncertainty associated with ON and OFF flash pulses with a suitable experimental setup. This will allow the characterization of the statistical properties associated with the duration of flash pulses in the proposed communication scheme. This was achieved with the set-up of figure 2. The transmitter side is composed by one HTC One X smartphone, configured according to the third methodology previously described, while the receiver is composed by a photo-detector followed by a transimpedance amplifying stage.

The received signal is threshold compared and feed in digital format to an Arduino Uno digital port for further processing. The Arduino board includes an ATmega328 microprocessor, used here for the purpose of measuring the duration of each impulse. The microprocessor samples the incoming signals at the rate of $1\mu s$ and builds two timing histograms, one for ON pulses and another for OFF pulses. The flash was set to transmit 30000 consecutive DH-PIM symbols to the Arduino based receiver, from an alphabet of N=8 different symbols. Both the transmitter and receiver were fixed, the distance between them was limited, and the setup was protected with an enclosure to remove the influence from external light sources. Thus, the impact of external noise sources can be disregarded when analyzing presented results. Moreover, because the receiving front-end regenerates the signal coming out of the photo-detector, additive noise is not a concern in this system.

In this particular experiment, $200\mu s$ ON and OFF pulses were used. Thus, symbols start with ON pulses with duration $200\mu s$ or $400\mu s$ (header one or two, respectively), followed by a series of OFF pulses (empty slots) with time duration $D \cdot 200\mu s$, with $D \in \{1,2,3,4\}$. Timing histograms were curve fitted to Gaussian shapes, as shown in Figure 3 for both headers (H_1 and H_2). As it can be seen, the duration of H_1 and H_2 pulses approximately behaves like a Gaussian random variable characterized by the temporal mean μ_1(for H_1) and $T_p + \mu_1$ (for H_2), and similar standard deviations (σ_1). The duration of empty slots has also shown to behave like Gaussian random variables with mean (μ_0) and variances (σ_0) that do not depend on the number of consecutive empty slots. Timing histograms for empty slots were also curve fitted to Gaussian shapes, and uncertainty parameters obtained. Results are shown in Table 1, for three consecutive runs. Ensemble averages for each case and parameter are also given.

To investigate the impact of ON and OFF nominal timings on the uncertainty parameters, the experiment was repeated for increasing nominal times, from $100\mu s$ to $600\mu s$. Figure 4 shows the absolute and relative uncertainty parameters. Note that these results were obtained for a single run and thus, exhibit natural deviations from the ensemble average that characterizes this smartphone. Nevertheless, it is possible to see that the uncertainty increases almost linearly with the ON and OFF nominal times while standard deviations are more stable and do not seem to depend on those times. This means that the relative uncertainty increases when the system is set for higher data rates (smaller ON and OFF nominal timings).

Results also show that ones and zeros have similar statistical properties, although $\Delta_0 = |T_{OFF} - \mu_0|$ is a little higher than $\Delta_1 = |T_{ON} - \mu_1|$. Nevertheless, it must be recalled that these results are for three runs only and were obtained with a particular transmitter. Different transmitters (different smartphones, with different flash control capabilities) may exhibit different absolute and relative uncertainty parameters. Thus, the receiver must cope with significant timing uncertainty. In fact, the receiver will never be adequately matched to the transmitted pulse, which depends on the flash controllability of each smartphone used. Thus, it is important to analyze the impact of uncertainty in DH-PIM symbol error probability.

3 Timing Uncertainty Analysis

In the previous section, the ability to produce accurate and predictable ON and OFF flash pulses was shown to depend greatly on the particular execution environment provided by smartphones. This section will analyze timing uncertainty in flash based VLC systems in order to evaluate its impact in symbol error probability.

3.1 Introduction

To evaluate the impact of timing uncertainty in DH-PIM symbol error probability, a generic DH-PIM system will be considered here. Each block of M input data bits is mapped to one of $n = 2^M$ possible symbols, as shown in Figure 5. The nominal slot duration is equal to $T_s = 2T_p$ and pulses have amplitude A. Each transmitted symbol starts with a header rectangular pulse, with nominal width equal to $d_h T_p$, with $d_h \in \{1,2\}$, followed by a series of empty slots (information slots), with nominal width equal to $d_i T_s$, with $d_i \in \{1, \dots, 2^{M-1}\}$. Note that headers are used both for data decoding and symbol synchronization, but information is conveyed mainly in empty slots. Thus, from the receiver perspective, the probability of receiving a one is much lower that the probability of receiving a zero. These probabilities are shown in (1), where $\bar{L}$ is the average DH-PIM symbol length.

$$p(1) = 1/\bar{L}\ ,\ \ p(0) = (\bar{L} - 1)/\bar{L} \quad for \ \ \bar{L} = (2^{M-1} + 3)/2 \tag{1}$$

The optimum DH-PIM receiver consists of a matched filter, which will integrate the symbol energy, followed by two parallel paths: one to detect and decode the header and one to decode information slots (Figure 6). Here, the detection scheme is based on the popular hard decision threshold detector because DH-PIM symbol boundaries are not known prior to detection. Information slots are decoded using a simple counter, followed another hard decision threshold detector. This is the usual scheme in DH-PIM receivers because information is carried in the duration of empty slots and not on the amplitude of samples therein [21].

The filter has an impulse response $h(t)$, matched to the transmitted pulse shape in H_1 or H_2. If it is matched to H_2, the second peak (as shown in Figure 7) can be used to synchronize the local clock generator with the incoming data sequence. Also, the peak sample y_p can be used to detect the type of header received. The threshold in this detector is set to the optimum level considering the probabilities of receiving H_1 or H_2. In this case, both headers have the same probability and thus, the threshold should be set midway between H_1 and H_2 levels. These levels are AT_p and AT_s, respectively, as shown in Figure 7.

Even in the absence of additive noise, errors may occur due to the uncertainty associated with ON and OFF timings. This uncertainty can affect DH-PIM system's performance due to decoding errors in both the headers and information slots. If a header is transmitted, an error occurs when: i) H_1 is transmitted but not detected (erasure error); ii) H_1 is transmitted but detected as H_2 (header detection error); iii) H_2 is transmitted but not detected (erasure error); or iv) H_2 is transmitted but detected as H_1 (header detection error). If zeros are transmitted an error occurs when a wrong number of information slots are detected. If P_{he} is the header error probability and P_{ie} is the information slot's error probability, the expression for symbol error probability can be written as shown in (2).

$$P_{se} = p(1) \cdot P_{he} + p(0) \cdot P_{ie} = p(1) \cdot \big[p(H_1) \cdot [p(0|H_1) + p(H_2|H_1)] + p(H_2) \cdot [p(0|H_2) + p(H_1|H_2)]\big] + p(0) \cdot p(z_n) \cdot p(z_m|z_n) \tag{2}$$

Here, $p(H_i)$ is the probability of transmitting H_i, $p(x|H_i)$ is the conditional probability of detecting x when H_i has been transmitted, $p(z_n)$ is the probability of transmitting n zero slots, and $p(z_m|z_n)$ is the probability of detecting m zero slots when n were transmitted. Next, the error probabilities in header and information slot's detection are analyzed separately, in the presence on timing uncertainty. In this analyzes, the probability of detecting an empty slot when H_2 is transmitted will be considered to be vanishingly small, so that $(p(0|H_2) \approx 0$. As explained before, the impact of Additive White Gaussian Noise will not be discussed here, as it will not have a significant impact on this system's performance. Nevertheless, the interested reader can find a thorough analysis in [21].

3.2 Impact on Header Decoding

According to experimental results shown in section 2.2, the uncertainty when generating a sequence of '1's or '0's will be considered not to depend on the absolute sequence time duration. Thus, H_1 and pulses are considered to have a random width equal to τ_1, H_2 pulses have a width equal to $\tau_1 + T_p$, and the transmitted zeros (empty slots) to have a random width equal to $\tau_0 + (d_i - 1)T_s$. Random variables τ_1 and τ_0 are considered to follow a Gaussian distribution, characterized by mean (μ_1 and μ_0) and standard deviation (σ_1 and σ_0) parameters. The absolute time difference between mean values and the correspondent nominal times will be therefore represented by $\Delta_1 = |\mu_1 - T_p|$ and $\Delta_0 = |\mu_0 - T_s|$.

To evaluate the uncertainty impact on header detection, the matched filter output has to be considered for four different situations: H_1 is transmitted with $\mu_1 < T_p$; H_1 is transmitted with $\mu_1 > T_p$; H_2 is transmitted with $\mu_1 < T_p$; and H_2 is transmitted with $\mu_1 > T_p$. These situations are illustrated in Figure 8. It shows that the second peak in the filter's output occurs at T_s in three out of four situations and thus, a synchronization offset will occur only when H_2 is transmitted with $\mu_1 > T_p$. This will impact the error probability in the information slot's detection and thus, will be considered in the next section.

Figure 8 also shows that, as long as the peak detector introduces no error, the peak sample (y_p) will be a constant value equal to AT_s when H_2 is transmitted and $\mu_1 > T_p$. In the other three situations, y_p will be a random variable with a conditional probability density function (PDF) that depends upon the presence of H_1 or H_2 pulses. Figure 9 shows curves of the conditional probabilities $p_y(y_p|H_i)$ along with the proposed decision thresholds, midway between expected levels for zero, H_1 and H_2.

Based in (1) and (2), and the shaded areas in Figure 9, it is possible to write the header error probability P_{he}, as shown in (3). This equation is based on the assumption that μ_1 can be smaller or larger than T_p, with equal probabilities, and that H_1 and H_2 occur with the same probability.

$$P_{he} = \frac{1}{2} \cdot \left[Q\left(\frac{A\left((T_p/2)-\Delta_1\right)}{\sigma_1}\right) + Q\left(\frac{A\left((T_p/2)+\Delta_1\right)}{\sigma_1}\right)\right] + \frac{1}{2} \cdot \left[\frac{1}{2} \cdot Q\left(\frac{A\left((T_p/2)-\Delta_1\right)}{\sigma_1}\right)\right] =$$

$$= \frac{3}{4} \cdot Q\left(\frac{A\left((T_p/2)-\Delta_1\right)}{\sigma_1}\right) + \frac{1}{2} \cdot Q\left(\frac{A\left((T_p/2)+\Delta_1\right)}{\sigma_1}\right) \quad (3)$$

3.3 Impact on Information Decoding

Uncertainty in the measurement of information slot's duration depends on: i) the uncertainty associated with the time reference used in measurement; and ii) the uncertainty associated with the empty slot's duration. If the time reference is perfect (no synchronization error), the counter will measure a Gaussian random time interval equal to $t_i = (d_i - 1)T_s + \tau_0$, with mean $\mu_i = (d_i - 1)T_s \pm \mu_0$ and standard deviation $\sigma_i = \sigma_0$. Thus, as before, the error probability is given by the shaded areas in the conditional probability density functions shown in Figure 10 and the expression for zero error probability can be written as shown in (4).

$$P_{e0|\overline{SE}} = Q\left(\frac{(T_s/2)-\Delta_0}{\sigma_0}\right) + Q\left(\frac{(T_s/2)+\Delta_0}{\sigma_0}\right) \quad (4)$$

On the other hand, if there is a timing error in the filter's output peak, the receiver is affected by this synchronization uncertainty. Thus, the counter will start later than it should and one (or more) empty slots may be missed by the detector, even if no uncertainty exists in the transmitted zero slots. Finally, if both uncertainties exist (in synchronization and empty slots), the mean of t_i becomes equal to $\mu_i = (d_i - 1)T_s \pm \mu_0 - \mu_1$ and the standard deviation equal to $\sqrt{{\sigma_0}^2 + {\sigma_1}^2}$. Note that the sum of variances must be taken due to the assumption that uncertainty in zeros and ones is not correlated. Thus, when synchronization error exists, the error probability is given by (5).

$$P_{e0|SE} = \frac{1}{2} \cdot Q\left(\frac{(T_s/2)-(\Delta_1+\Delta_0)}{\sqrt{{\sigma_0}^2+{\sigma_1}^2}}\right) + \frac{1}{2} \cdot Q\left(\frac{(T_s/2)-(\Delta_1-\Delta_0)}{\sqrt{{\sigma_0}^2+{\sigma_1}^2}}\right) \quad (5)$$

Finally, the information error probability P_{ie} is given by (6), considering that synchronization errors may occur in one out of the four situations depicted in Figure 8.

$$P_{ie} = p(SE) \cdot P_{e0|SE} + p(\overline{SE}) \cdot P_{e0|\overline{SE}} = \frac{1}{4} \cdot P_{e0|SE} + \frac{3}{4} \cdot P_{e0|\overline{SE}} \quad (6)$$

4 Results and Discussion

The present section analyzes the symbol error probability for different uncertainty parameters using the expressions derived in section 3. Figure 11a) shows the symbol error probability (P_{se}) for an increasing time deviation from nominal time and different standard deviations, with $T_s = 2T_p = 200\mu s$ and M=3 (8 symbols). Statistical parameters were considered to be proportional to nominal times and thus, $\mu_0 = 2\mu_1$ and $\sigma_0 = 2\sigma_1$. The particular ON and OFF timings used in the experiments described in section 2 ($T_s = T_p = 200\mu s$) was also evaluated. Results are shown in Figure 11b), where $\mu_0 = \mu_1$ and $\sigma_0 = \sigma_1$. These are both valid DH-PIM schemes, although the symbol mean time duration in the first is a little shorter than in the second. According to (1), the first exhibits $\bar{L} = 7T_s/2 = 700us$ while the second exhibits $\bar{L} = 4T_s = 800us$.

As expected, there is only a small difference in P_{se} when comparing both scenarios, which is noticeable only for small Δ. In this case, the impact of σ_0 and $\sigma_1$is higher and thus, the scheme with lower global variances is the one with lower error probability. Moreover, as it can be seen in both graphics that the error probability increases fast when the pulse's duration is higher than $\mathrm{T_s}/2$, as this is the threshold level at the information hard-decision detector. Also, the impact of the standard deviation is as expected - it increases the error probability when Δ is small and decreases it for higher Δ.

Also interesting is the behavior of the symbol error probability for different ‘1’s and ‘0’s time differences and standard deviations. In the following analysis it was considered the general case where $T_s = 2T_p = 200\mu s$. Figure 12a) shows that uncertainty in ‘1’s (header pulses) have a higher impact than in ‘0’s, for large Δ. This is due to the fact that uncertainty in ‘1’s affects both the header and information slot’s decoding. On the contrary, Figure 12b) shows that for a given standard deviation, any deviation from the nominal slot duration in ‘0’s has a greater impact in symbol error probability than in ‘1’s. This makes sense if we remember that the probability of receiving ‘0’s is higher than the probability of receiving ‘1’s.

As a final comment on these results, it is important to highlight that error probabilities are unusually high in these graphics because we considered unusually high timing uncertainty parameters. This large amount of variability may indeed occur is the described communication system because the receiver cannot know the transmitter’s timing characteristics in advance. Thus, it will never be perfectly matched to the transmitter and high deviations from nominal times may occur. Future work to alleviate this problem should include the development of a dynamically adjusted matched filter in the receiver to mitigate the static uncertainty introduced by the particular characteristics of different transmitters.

5 Conclusions

This paper discussed limitations and tradeoffs on the usage of the smartphone flash LED as transmitting device, employing visible light communication concepts. It was ascertained that timing uncertainties constrain the achievement of high data rates. Timing uncertainties depend on the smartphone model, its hardware resources, and operating systems. Results are expected to change for different devices, but maintain similar trends, due to the fact that the flash LED will have similar specifications. Although the relative uncertainty was shown to increase with data rates, it was disclosed the possibility to achieve a switching speed in the range of 100us, which will open up space for several low data rate applications. Also, the receiver could be designed to tolerate higher uncertainty levels by resorting to dynamic automatic threshold adjustment.

6 References

[1] IEEE Standard 802.15.7- IEEE Standard for Local and Metropolitan Area Networks-Part 15.7: Short-Range Wireless Optical Communication Using Visible Light, 2011

[2] M. Xie, L. Hao, K. Yoshigoe, J. Bian, “CamTalk: A Bidirectional Light Communications Framework for Secure Communications on Smartphones”, Security and Privacy in Communication Networks, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Volume 127, 2013.

[3] W. Hu, H. Gu, Q. Pu, “LightSync: Unsynchronized Visual Communication over Screen-Camera Links”, Proceedings of the ACM International Conference on Mobile Computing & Networking, New York, USA, 2013.

[4] "Casio Unveils Prototype of Visible Light Communication System Using Smartphones at CES," 14 January 2012. [Online]. Available: http://world.casio.com/news/2012/0115_VisibleLightcomm/. [Accessed on November 15, 2014].

[5] M.M. Galal, H.A. El Fayed, A.A. Aziz, M.H. Aly, “Smartphones for Payments and Withdrawals Utilizing Embedded LED Flashlight for High Speed Data Transmission”, IEEE International Conference on Computational Intelligence, Communications Systems and Networks, Madrid, Spain, 2013.

[6] M.M. Galal, A.A. El Aziz, H.A. Fayed, ; M.H. Aly, "High Speed Data Transmission over a Visible Light Link Employing Smartphones Xenon Flashlight as a Replacement of Magnetic Cards", International Conference on High Capacity Optical Networks and Enabling Technologies, Magosa, Cyprus, December, 2013.

[7] T. Hesselmann, N. Henze, S. Boll, “Flashlight: Optical Communication Between Mobile Phones and Interactive Tabletops”, ACM Proceedings of the International Conference on Interactive Tabletops and Surfaces, New York, USA, 2010.

[8] J.A. Otto, "Methods and Apparatus for an Electronic Shelf Label Communication System", USA Patent US 6496121 B2, December 2002.

[9] H. Hong, Y. Ren, R. Tian, L. Xiao, “Electronic Shelf Label System Based On Public Illuminating Network”, IEEE Asia Pacific Conference on Circuits and Systems, Macao, China, December, 2008.

[10] João Paulo Barraca, Luis Nero Alves, Mónica Figueiredo, "Electronic Shelf Labeling Employing Visible Light Communication Concepts," IEEE/IET International Symposium on Communication Systems, Networks & Digital Signal Processing, Manchester, United Kingdom, July 2014.

[11] P. DeMil, B. Jooris, L. Tytgat, R. Catteeuw, I. Moerman, P. Demeester, A. Kamerman, “Design and Implementation of a Generic Energy-Harvesting Framework Applied to the Evaluation of a Large-Scale Electronic Shelf-LabelingWireless Sensor Network”, EURASIP Journal on Wireless Communications and Networking, July 2010.

[12] G. Corbellini, K. Aksit, S. Schmid, S. Mangold, T.R. Gross, "Connecting Networks of Toys and Smartphones with Visible Light Communication," IEEE Communications Magazine, Vol. 52, Nº 7, July 2014.

[13]. S. Mangold, “Visible Light Communications for Entertainment Networking”, IEEE Photonics Society Summer Topical Meeting Series, Seatle, USA, July 2102.

[14] A. C. Boucouvalas and P. Huang, "Gbit/s Data Rate IrDA Protocol Performance Evaluation", The Mediterranean Journal of Computers and Networks, Vol. 1, Nº 2, October 2005.

[15]. “Multi-Gigabit Communications (Giga-IR™) Using Infrared Technology”, IrDA Organization, www.irda.org, [Accessed on November 15, 2014].

[16] N.M. Aldibbiat, Z.F. Ghassemlooy, R. McLaughlin "Performance of dual header-pulse interval modulation (DH-PIM) for optical wireless communication systems", Proceedings of SPIE 4214 on Optical Wireless Communications III, Vol. 4214, February 2001.

[17] HTC One X datasheet, https://www.htc.com/us/smartphones/htc-one-x/, [Accessed on November 15, 2014].

[18] GSM Arena Phone Finder, http://www.gsmarena.com/search.php3, [Accessed on November 15, 2014].

[19] “AAT1271 datasheet - 1.5A Step-Up Current Regulator for Flash LEDs”, Skyworks, June 2012.

[20]. NVidia Corporation, Tegra 3 Technical Reference Manual version 1.0, November 2011.

[21] Z.F. Ghassemlooy, W. Popoola, S. Rajbhandari, "Optical Wireless Communications: System and Channel modelling with Matlab”, CRC Press, Taylor and Francis Group, 2013.

Figures

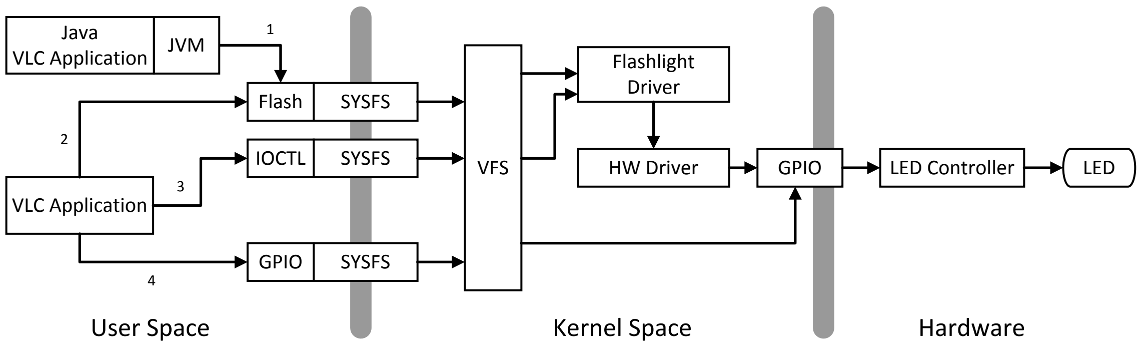


Figure 1 – Layered stack for flash control in Android smartphones.

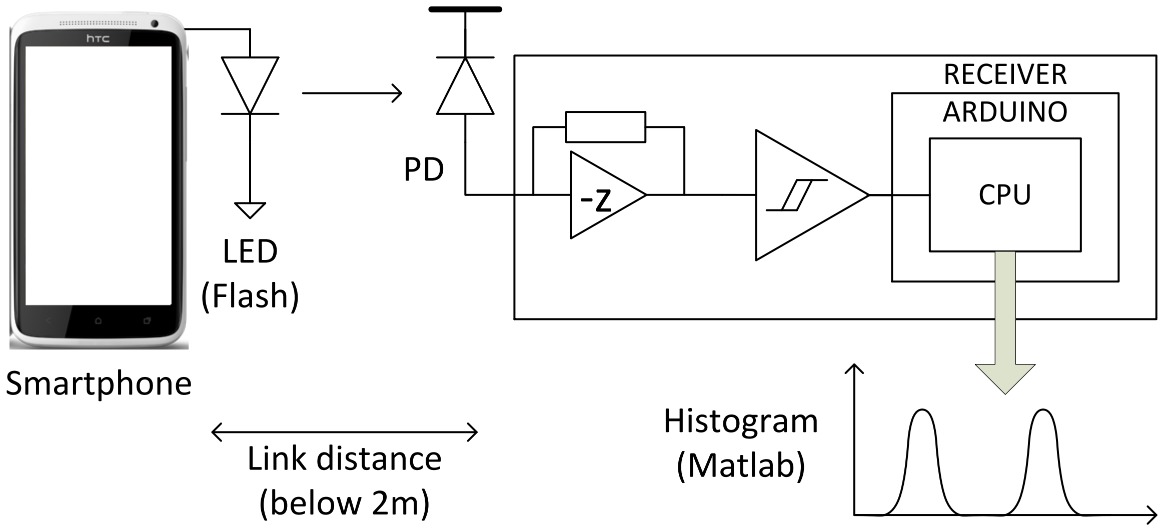


Figure 2 – Experimental set-up for LED timing characterization

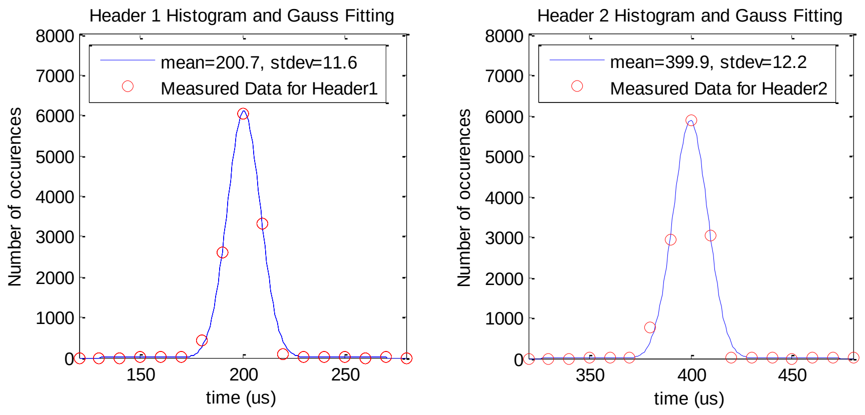


Figure 3 – Histograms for H_1 and H_2 and the correspondent Gaussian fitting

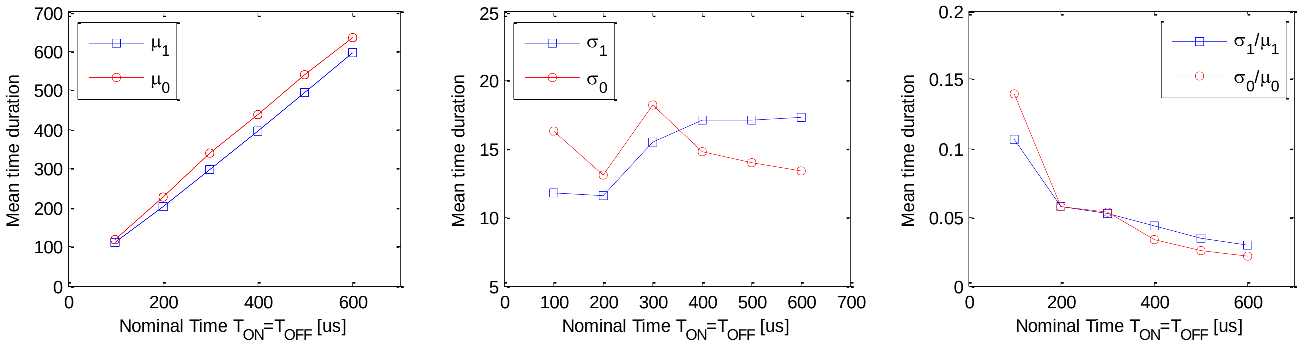


Figure 4 – Uncertainty parameters for increasing nominal ON and OFF timings: a) mean; b) standard deviation; and c) relative uncertainty.

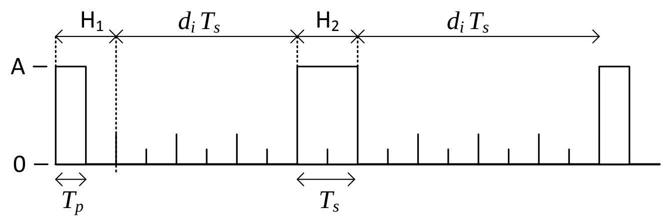


Figure 5 - DH-PIM symbol structure showing both headers (H1 and H2)

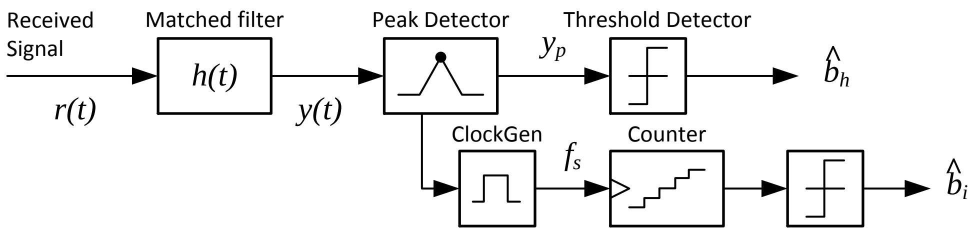


Figure 6 - DH-PIM optimum receiver

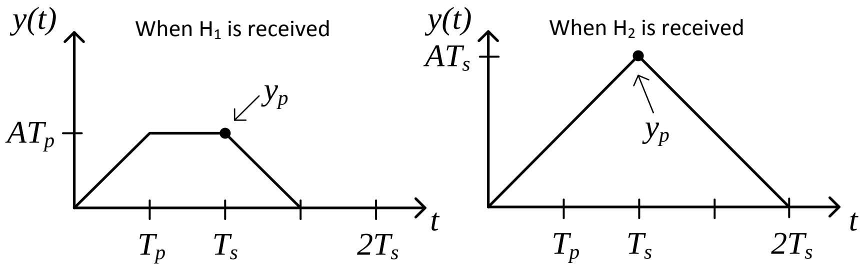


Figure 7 - Matched filter output when receiving H1 and H2 pulses

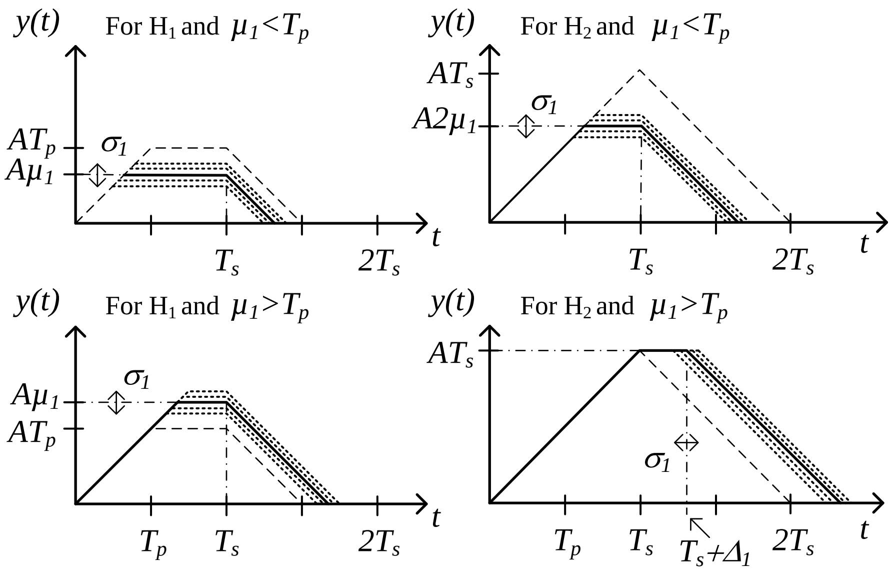


Figure 8 - Matched filter output time uncertainty in H_1 and H_2.

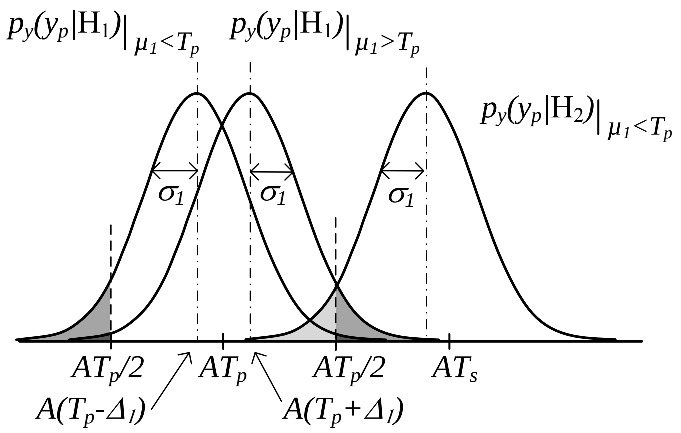


Figure 9 - Conditional PDFs with decision thresholds and error probabilities.

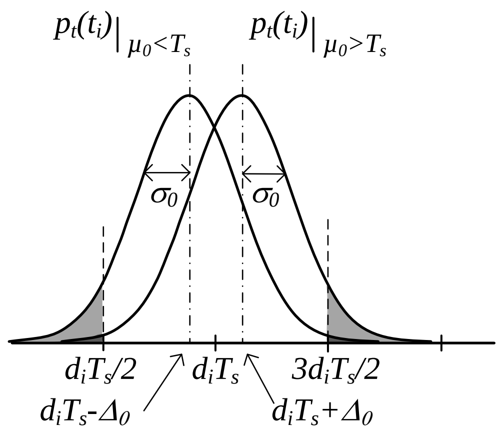


Figure 10 - Conditional PDF with decision thresholds and error probabilities.

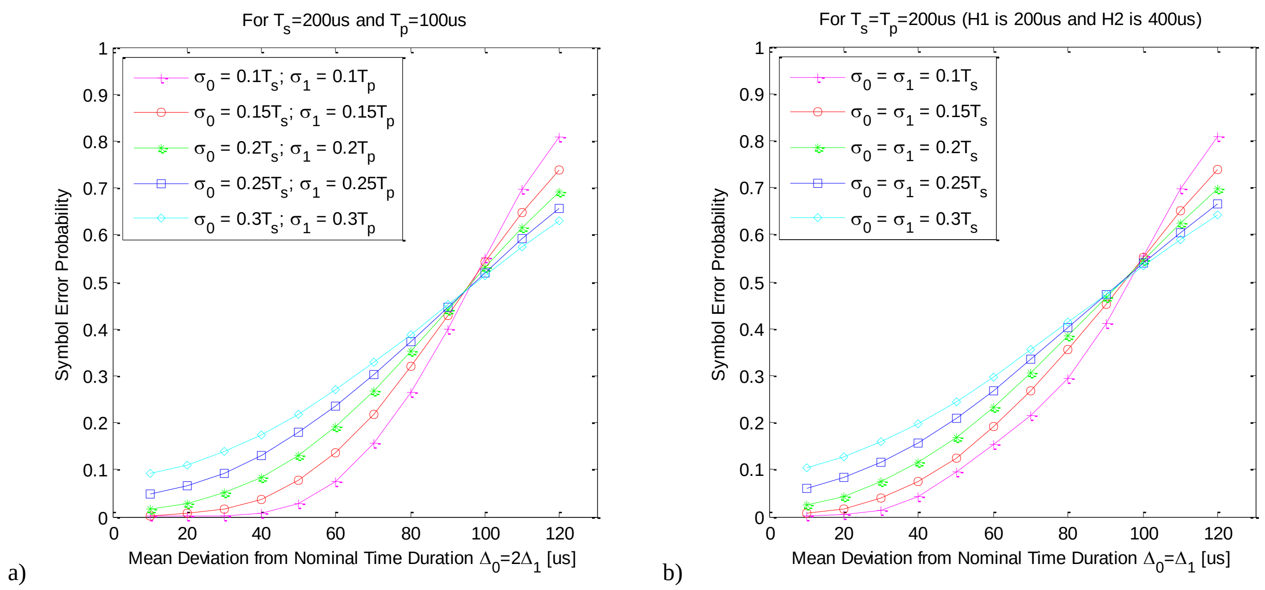


Figure 11 – Symbol error probability for M=3, for increasing mean deviation from nominal time duration and different standard deviation parameters. Two schemes are shown: a) $T_s = 2T_p = 200\mu s$; and b) $T_s = T_p = 200\mu s$.

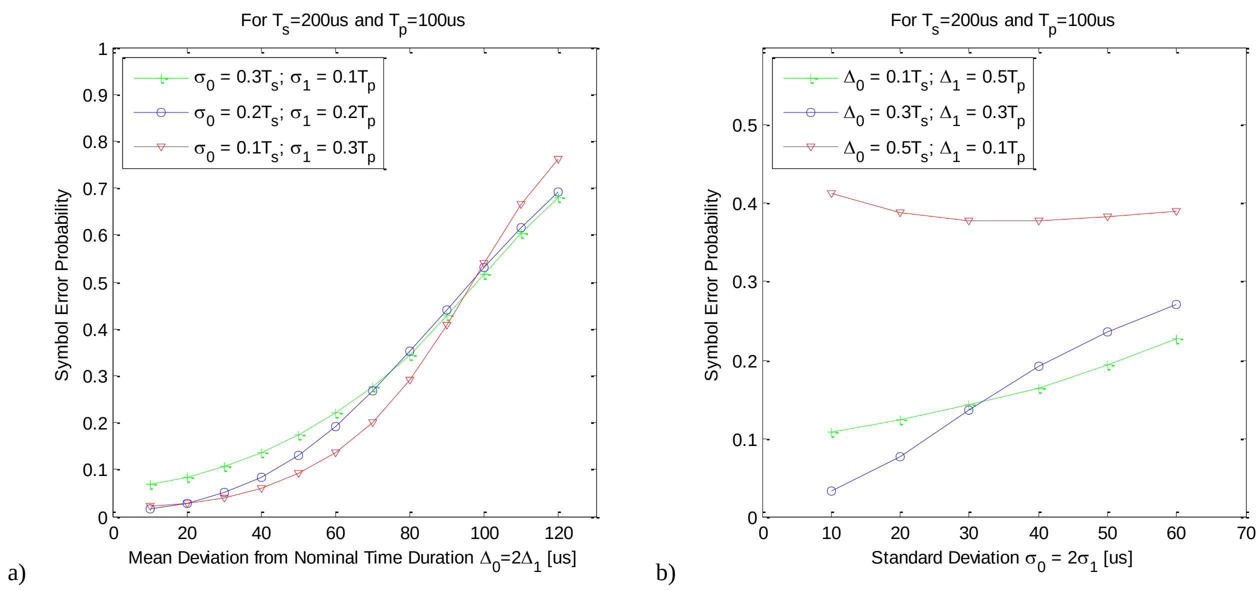


Figure 1 - Symbol error probability for: a) increasing Δ_0= $2\Delta_1$ and different standard deviations σ_1 and σ_0; b) increasing standard deviation and different time differences Δ_1 and Δ_0.

Tables

Table 1 – Measured uncertainty parameters, with T_{ON}=T_{OFF}=200us for headers and empty slots.

	Headers				Information Slots							
	H_1		H_2		D=1		D=2		D=3		D=4	
Run	μ_1	σ_1	μ_1	σ_1	μ_0	σ_0	μ_0	σ_0	μ_0	σ_0	μ_0	σ_0
1	200.7	11.6	199.9	12.2	226.9	13.1	232.9	13.5	229.5	11.2	227.8	10.2
2	199.8	11.7	210.9	14.6	226.9	12.8	229.9	15.1	227.2	12.6	225.7	12.5
3	199.1	11.7	202.5	17.7	226.9	12.9	234.5	13.3	231.0	12.3	229.6	11.3
Mean	**199.9**	**11.7**	**204.4**	**14.8**	**226.9**	**12.6**	**232.4**	**14.0**	**229.2**	**12.0**	**227.7**	**11.3**