UNIVERSIDADE D
COIMBRA

João Carlos dos Reis Frazão

QUANTUM COMMUNICATIONS SYSTEM USING QUBITS WITH POLARIZATION ENCODING

Dissertação no âmbito do Mestrado Integrado em Engenharia Física no ramo da Instrumentação
Orientada pelo Professor Doutor Armando Nolasco Pinto
Co-Orientada pela Professora Doutora Maria Helena Almeida Vieira Alberto
Apresentada ao Departamento de Física da Faculdade de Ciências e Tecnologias da Universidade de Coimbra

Julho de 2020

Acknowledgements

A realização desta dissertação de mestrado contou com importantes apoios e incentivos sem os quais não se teria tornado uma realidade e aos quais estarei eternamente grato.

Ao Professor Doutor Armando Nolasco Pinto, pela sua orientação, total disponibliidade, apoio, pelas opiniões, critícas e conhecimento que transmitiu ao longo deste trabalho. A todo o grupo de Comunicações e Tecnologias Ópticas Quânticas do Instituto de Telecomunicações, e à Universidade de Aveiro por toda a ajuda na parte laboratorial desta dissertação.

À professora Doutora Maria Helena Almeida Vieira Alberto, que apesar de não ser a sua área e de não ter estado presencialmente na realização deste trabalho, mostrou sempre disponiblidade e interesse em acompanhar o projecto. Agradeço também toda a dedicação, paciência e correções durante esta última etapa.

Aos meus pais, em especial à minha mãe, que sempre me incentivou a seguir o caminho que sempre quis. Agradeço toda a paciência, companhia, preocupação e por ter conseguido dar-me educação escolar, mesmo em tempos difíceis. Obrigado mãe.

Aos meu colegas de Leiria, em especial ao João Jójó por ter sido sempre uma pessoa com que posso contar. Sempre me motivou, ajudou e manteve o contacto em todos os momentos. Obrigado meu mambo.

Aos meus colegas de curso. João Silva, obrigado pelo apoio, dedicação e paciência para me aturares durante estes últimos cinco anos.

To Byron.

Resumo

Compreende-se criptografia como a prática de princípios e técnicas que permitem uma comunicação segura, na presença de terceiros. Com o desenvolvimento dos computadores quânticos, a utilização de um algoritmo quântico muito eficiente (algoritmo de Shor) para atacar a atual criptografia assimétrica pode transformar-se numa realidade. Isso comprometeria a segurança dos sistemas atuais e futuras trocas de informações. Nesta dissertação é estudada uma implementação do protocolo quântico BB84, que utiliza variáveis discretas com codificação na polarização de fotões.

Na primeira parte deste trabalho, foi estudado o recetor já implementado no laboratório do Instituto de Telecomunicações de Aveiro. De forma a otimizar o processo de recolha e processamento de informação, foi desenvolvida uma solução baseada no Arduino. Conforme foram realizados alguns testes, percebeu-se que seria necessário adicionar um novo Arduino e uma placa periférica para gerir quatro valores de tensão de um controlador de polarização. Foram ainda obtidos resultados testes do Quantum Bit Error Rate (QBER), onde se verifica a estabilidade do sistema.

Por último, de forma a obter ritmos de operação elevados, estudou-se uma solução baseada em moduladores IQ (In-phase Quadrature). Para se conseguir gerar os seis estados de polarização nas três bases não ortogonais (base padrão, base diagonal e base circular) propusemos uma estrutura Dual-IQ. Nesta estrutura, o sinal ótico é dividido em duas partes iguais, cada uma passando por um modulador IQ. Aqui, uma diferença de fase é introduzida em cada um dos sinais, e com a ajuda de um aparelho para a rotação de polarização é possível gerar vários estados de polarização. Foram realizadas simulações de forma a demonstrar que esta estrutura é capaz de gerar os seis estados de polarização necessários.

Keywords – Criptografia Quântica, Recetor, Arduino, Transmissor, Modulador Dual-IQ

Abstract

Cryptography can be understood as the practice of principles and techniques, that allows secure communications, in the presence of unwanted parties. With the development of quantum computers, the use of a very efficient quantum algorithm (Shor algorithm) to attack the current asymmetric cryptography can become a reality. This would compromise the security of the current systems and future information exchanges. In this dissertation, an implementation of the BB84 quantum protocol is studied, which uses polarization encoding on single photons.

In the first part of this work, the already implemented receiver, in the laboratory of Instituto de Telecomunicações de Aveiro, was studied. To optimize the process of collecting and processing information, a solution based in the Arduino was developed. As some tests were carried out, it was perceived that the system needs two Arduinos and a peripheral board to manage four voltage values of a polarization controller. The Quantum Bit Error Rate (QBER) tests results were also obtained, where the stability of the system can be analyzed.

Finally, in order to acquire higher operation rates, a solution based on IQ (In-phase Quadrature) modulators was studied. To generate the six polarization states, in the three non-orthogonal bases (standard base, diagonal base and circular base), we proposed a Dual-IQ structure. In this structure, the optical input signal is divided into two equal parts, each passing on the IQ modulator. Here, a phase difference is introduced in each signal, and with the help of a polarization rotator it is possible to generate several polarization states. Simulations have been performed to demonstrate that this structure is capable of generating the six required polarization states.

Keywords – Quantum Cryptography, Receiver, Arduino, Transmitter, Dual-IQ Modulator

Acronyms

QBER - Quantum Bit Error Rate

IQ - In-Phase Quadrature

RSA - Rivest-Shamir-Adleman

QKD - Quantum Bit Error Rate

DV-QKD - Discrete Variable - Quantum Key Distribution

CV-QKD - Continuous Variable - Quantum Key Distribution

PNS - Photon Number Splitting

I/O - Inputs/Outputs

EPC - Electronic Polarization Controller

PBC - Polarization Beam Combiner

PBS - Polarization Beam Splitter

DAC - Digital to analog converter

APD - Avalanche Photodiode

UART - Universal asynchronous receiver-transmitter

SPI - Serial Peripheral Interface

I2C - Inter-Integrated Circuit

COM - Communication port

QAM - Quadrature Amplitude Modulation

PSK - Phase Shift Keying

BPSK - Binary Phase Shift Keying

QPSK - Quadrature Phase Shift Keying

List of Figures

List of Tables

Table of Contents

Chapter 1

Introduction

The importance of cryptography considerably expanded during and after the two world wars. However, it was the introduction of the Internet that brought the interest of cryptography to the everyday life of common world citizens. Crypto systems now enable emails, phone, and financial communications to be safe on a daily basis. Nowadays it is important to have a method that secures the content of messages between distant parties, preventing the theft of information by any unauthorized party. This represents one of the main goals of cryptography, and can be accomplished by encrypting the transmitted message, in such way, that no external entity can decipher it. Throughout this work the two legitimate parties are mentioned as Alice (transmitter), Bob (receiver), and the eavesdropper as Eve, according to the established convention. The cryptosystems can be divided in two main branches: the secret key cryptography (symmetric) and public key cryptography (asymmetric).

In an asymmetric-key cryptography system, Alice encrypts the message utilizing a public key, and sends it to Bob who possesses the private key [8]. These keys are both mathematically related, and an eavesdropper intercepting the encrypted message cannot decipher it in a reasonable amount of time. The public key cryptosystems have been successful and popular over the last years. One of the first implemented protocols is the RSA (Rivest-Shamir-Adleman) encryption, which is built on the principles of public key exchange and the computationally difficult problem of prime factorization. For a classical computer, an efficient algorithm for finding the prime factors of a composite, in a reasonable time, does not exist. Even so, this assumption was challenged in 1995, when Peter Shor proposed a polynomial-time quantum algorithm for the factoring problem. With the advances in quantum technology's, the security of the asymmetric protocols are starting to being threatened by the

increasing quantum computational power to break the relation between public and private keys.

In the secret key cryptography only one key is used to encrypt and decipher a message. These cryptosystems use the one-time pad encryption technique, that requires the use of a one time pre-shared key [8]. In this scheme, each bit in the message is encrypted by a bit in the key. Alice obtains a scramble text and sends it to Bob, through an authenticated public channel. Bob deciphers the message by subtracting the key. Because the bits of the scrambled text are as random as those of the key, they do not contain any important information. The ciphertex can be considered unconditionally safe if the key is at least the same size of the message, random, only used once and if Alice and Bob are able to share it in complete secrecy. However, the classical ways of sharing the private key are not satisfactory and the problem of distributing secret keys arises.

Quantum mechanics offers a solution to this problem: If a key can be generated and transmitted using a quantum system then, in principle, that key can be distributed securely, without requiring any assumptions of the computational power that an eavesdropper possesses. This is possible because any measurement on a quantum system disturbs its state, and unknown quantum states cannot be copied. And so the quantum key distribution comes into action.

1.1 State of art

Quantum key distribution (QKD) is a method used to exchange encryption keys between two distant parties, who are communicating through a public channel. The intrinsic laws of quantum mechanics allows the unconditional security of the QKD protocols. This security is based on the non-cloning theorem and on the Heisenberg uncertainty principle [18]. The non-cloning theorem claims that an unknown quantum state cannot be copied and therefore the eavesdropper cannot copy qubits to be measured later. This means when a single photon is measured, its state is perturbed and the action of an eavesdropper can be detected. For these systems, the quantum mechanics do not prevent a third party from eavesdropping, but instead

enables Alice and Bob to detect the presence of Eve in the system [14]. If Eve tries to take information from the system, discrepancies between Alice's and Bob's keys are going to show in the post-processing techniques, giving them the choice to discard the keys and repeat the process to generate a new one.

Two standard techniques have been the focus in developing QKD protocols: discrete variable - quantum key distribution (DV-QKD) and continuous variable - quantum key distribution (CV-QKD). The first one uses observables that admits discrete values, based on single photon detection. The latter one uses CV observables of light field that can be measured by shot-noise limited homodyne detection [18]. Since this work is based on a DV-QKD, a look over the evolution of these protocols is presented.

The first protocol for quantum cryptography, was proposed in 1984 by Charles H. Bennett, of IBM and Gilles Brassard, of the University of Montreal, hence the name BB84, as this protocol is known [8]. In this protocol, Alice sends a single photon encoded in a specific polarization using two non-orthogonal states for the bit 0 and two non-orthogonal states for the bit 1 (forming two non-orthogonal bases). Then Alice sends the encoded single photon to a quantum channel, so that Bob can measure it according to a determined orientation of polarization. Bob publicly announces the basis chosen for the measurement, so that him and Alice can keep the bits encoded with the same bases. On the other hand, if the encoded basis does not match the basis used for the measurement, they discard the bits. If Eve tries to intercept the photon sent by Alice, she is going to introduce disturbance in the system, which can be detected by estimating the discrepancy rate of the sifted keys [14].

In 1992, Charles Bennett suggested that only two non-orthogonal states are really needed for a quantum key distribution system [18]. Here Alice prepares the information between two quantum states that are going to correspond to the bit 0 and 1, in two different bases. However, such scheme is only secure if the losses along the process are very low. When only using two non-orthogonal states, some measurements yield the value of the bit being sent, whether other measurements are inconclusive. If she has obtained an inconclusive result, she blocks the signal, while if she has detected the state, she is able to re-send a correct state to Bob because she knows it

with certainty. To compensate the blocked photons, she can send a pulse of higher intensity so that Bob cannot observe any decrease in the expected transmission rate.

The six-state protocol was introduced by Pasquinucci and Nicolas Gisin in 1999. Instead of using four states, a six state one can better respect the symmetry for a better key generation rate and tolerance to noise [8]. The six states form three non-orthogonal bases, reducing the probability that Alice and Bob can match their sequence. However, the symmetry also reduces the crucial information Eve can gain for a given error rate, simplifying the security of this protocol. This extra choice of basis causes the eavesdropper to produce a higher rate of error, when attacking the single photons, thus becoming easier to detect.

In 2004, Scarani, Acin, Ribordy and Gisin, proposed a new variant of the BB84 at the classical communication channel stage. Since the single-photon sources are not perfect, there is some probability for a source to emit multiple photons with identical encoding, in a given run of the protocol. This proves to be a vulnerability to an eavesdropper who employs the photon number splitting attack (PNS) [18]. The essential idea is that Eve can perform a quantum non-demolition measurement to determine the number of photons in a run. Then she could steal one of the excess photons while forwarding the others to Bob. In this way, Bob could not detect the intruder, while waiting for the Alice basis revelation. A natural way against this type of attack is to discount the information revealed regarding the basis used in the prepare/measure stages. The SARG04 does the same initial step of the photon transmission of the BB84 protocol, where Alice sends one of the four states selected randomly from the two bases, and Bob does the measurement. However, when Alice and Bob determine which bits their basis matched, Alice does not directly announce her basis, but instead a pair of non-orthogonal states, one of which being used to encode her bit. Since the two states are non-orthogonal, the PNS attack cannot provide Eve with perfect information on the encoded bit.

Even though there are more QKD protocols, using properties such as Entanglement, Time bins [4] and the continuous variables, different implementations of the BB84 protocol are still being published in the recent years. The newest implementations focus on avoiding loopholes, having higher secret key rates (GHz), aiming for longer

distances (100km -300km), and a straightforward integration into existing infrastructures of communication networks. And so, implementations using intrinsically stable loops [[13],[21],[2]] are often being proposed. Another way to implement such protocol is to use phase modulators, both at the transmitter and receiver side [6], or use just one in the transmitter with a complex polarization beam splitter (PBS) set up in the receiver[10]. However, the performance of these set-ups are hard to replicate, and can be complex/expensive.

1.2 Motivation

Since the quantum technologies are experiencing a rapid growth, when given the chance to do a dissertation about optimizing the laboratory implementation of a QKD, in the Instituto de Telecomunicações de Aveiro, I decided to participate in it.

With the development of quantum computation, the security of classical cryptography will eventually break down in the future. These protocols are known for using cryptographic keys, to encode and decrypt messages. Nevertheless, these keys are mathematically related to one another, meaning a high-end computer with enough power can break the link between them. Another way to implement classical protocols, is to use the same key to encrypt and decrypt a message. Even so, each key should only be used once and should have the same size of the message that is sent. This raises the problem of key distribution, since sharing a large set of keys through a personal meeting is not a viable solution, in general.

Quantum cryptography created a solution for a safe distribution of keys at distance. However, there still are some technology limitations that slow the process of bringing quantum key distribution to the real world.

1.3 Goals

The purpose of this is work is to optimize an existing system in IT-Aveiro. Here the BB84 protocol is used to encode information in the polarization of a single photon.

To implement this protocol, the system is divided in two parts: the transmitter and the receiver, linked by a quantum channel (optical fibers).

Since the apparatus is still in the early stages, the current status can be enhanced, in order to achieve higher key generation rates, low error rates, and stable communications.

Looking at the receiver, it needs a module capable of receiving information, process it and then act accordingly to specific instructions. With this set up Bob would be able to commute between the non-orthogonal bases to do his measurements. The transmitter needs a new design, in order to encode information, using the polarization of single photons at the GHz margin.

1.4 Document structure

This document is divided in 5 chapters, being the current one the introduction. The rest can be summarized here:

- Chapter 2 summarizes the theoretical information needed to understand the concepts of discrete variables quantum key distribution.
- Chapter 3 describes the process of optimizing the receiver implementation, using an Arduino based solution.
- Chapter 4 includes the theoretical study of IQ modulation, to understand and simulate a transmitter solution based on Dual-IQ modulators.
- Chapter 5 summarizes the results obtained in this dissertation and describes the future work.

Chapter 2

Discrete Variables QKD

2.1 Introduction

In this chapter concepts related to quantum optics and quantum key distribution are introduced as background information for the work developed in the dissertation. This chapter starts with the introduction of the unity of information in the quantum world, the qubit. One way to encode the qubits is to use the polarization property of light which can be represented by two formalisms. Then it is explained how these qubits can be measured and how an eavesdropper can affect the system. Since the BB84 protocol is being used in the laboratory, a further look in to it is pertained. Due to the non-ideal world we live in, some technology equipment errors are studied, as well as two different type of implementations using polarization encoding.

2.2 Qubit

In the classical world, the bit represents the unit of information, and it can only have two unique states, $|0\rangle$ and $|1\rangle$ [14]. For example, the state $|0\rangle$ can represent the situation where a system is off and the state $|1\rangle$ represents where a system is on. There are many examples that illustrates this logic, and one can copy, erase and save data, as long as the information is supported by the classical laws of physics. However, there are physical states that do not have a classical correspondence. In the quantum world, the unit of information is the quantum bit, conventionally mentioned as the qubit. The state of a qubit is a normalized vector in a two-dimensional complex vector space, with an inner product $\langle\psi|\psi\rangle = 1$. Unlike classical bits, qubtis do not represent a single state, but instead a linear superposition of the classical states:

$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ where α and β are complex numbers that satisfy $|\alpha|^2 + |\beta|^2 = 1$. The superposition of states relates to the ondulatory nature of particles, that is not perceptible in the macroscopic world. In a quantum system, any measurement performed on the qubit, collapses its state into a classical one of $|0\rangle$ or $|1\rangle$. Thus, the initial state contains more information than the measured one, and the measurement destroys the qubit, turning it into a normal bit. It is also impossible to create an identical copy of an arbitrary unknown quantum state, which is used as a big advantage in QKD protocols.

Qubit states can be represented graphically by vectors on what is known as a Bloch sphere. In this representation the states $|0\rangle$ and $|1\rangle$ are in the poles of the sphere and any states lying on opposite sides are orthogonal and form a basis. Any point on the surface of the sphere (vector length of 1) corresponds to a pure state. Since the coefficients α and β of a general qubit are complex numbers, the equation can be rewritten as:

$$|\psi\rangle = Ae^{i\gamma}\,|0\rangle + B^{i\delta}\,|1\rangle \tag{2.1}$$

Here A, B, γ and δ are real numbers, and the angle $\phi = \gamma - \delta$ can be considered. The term $e^{i\gamma}$ can be use as a common factor for both states, and it does not represent a physical observable effect on the system. Since $A^2 + B^2 = 1$, A can be substituted by $\cos(\frac{\theta}{2})$ and B by $\sin(\frac{\theta}{2})$, leaving the equation with the following final form:

$$|\psi\rangle = \cos(\frac{\theta}{2})\,|0\rangle + e^{i\phi}\sin(\frac{\theta}{2})\,|1\rangle \tag{2.2}$$

The angles θ and ϕ are spherical coordinates of a specific point in the unitary radius sphere, called Bloch Sphere. The basis states $|0\rangle$ and $|1\rangle$ are in the z axis, more specifically in the top and bottom. This is known as the z basis or standard basis.

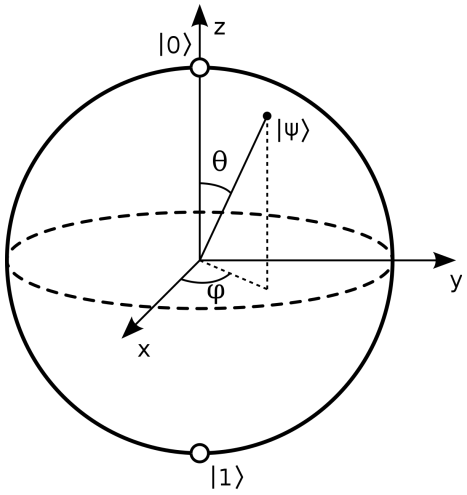

Figure 2.1: Bloch sphere representation. The states $|0\rangle$ and $|1\rangle$ represent the standard basis and are located at the top and bottom of the z-axis. Any other state can be represented on the surface of the sphere.

2.3 Photonic qubits

There are many ways to create and prepare qubits. The polarization of a photon is one of the most important properties when encoding information in quantum cryptography. A single photon is an electromagnetic wave, and its polarization corresponds to the direction of the electric field. Unpolarized radiation occurs when the direction of the electric field fluctuates randomly in time. Many common light sources produce unpolarized light such as LED, halogen lamps, sunlight, etc. However, if the direction of the electric field is well-defined, polarized light can be achieved. One of the most common sources of polarized light is a laser.

2.4 Jones formalism

The polarization of a photon can be characterized by considering an electromagnetic plane wave, polarized in the direction $\hat{e}_\alpha$, propagating in the y direction : $\vec{E} = E_0 e^{i(ky-wt+\delta)}\hat{e}_\alpha$. This wave can be described as the superposition of two plane waves, polarized in the z and x directions, meaning $\vec{E} = \vec{E}_z + \vec{E}_x$:

$$\begin{cases} \vec{E}_z = E_{0z}\cos(\alpha)e^{i(ky-wt+\delta)}\hat{z} \\ \vec{E}_x = E_{0x}\sin(\alpha)e^{i(ky-wt+\delta)}\hat{x} \end{cases} \tag{2.3}$$

The real components of the $\vec{E}$ vector are:

$$\begin{cases} E_z = E_{0z}\cos(\alpha)\cos(ky-wt+\delta_z) \\ E_x = E_{0x}\sin(\alpha)\cos(ky-wt+\delta_x) \end{cases} \tag{2.4}$$

If δ_z and δ_x aren't equal, the vector $\vec{E} = \vec{E}_z\hat{e}_z + \vec{E}_x\hat{e}_x$ direction would not remain constant. Meaning the tip of vector $\vec{E}$ is going to describe an ellipse, in the perpendicular plane to the polarization direction. Only the phase difference $\phi = \delta_x - \delta_z$ has physical meaning. The following equation can be obtained:

$$\vec{E} = \vec{E}_z + \vec{E}_x \tag{2.5}$$

$$\vec{E} = E_0 e^{i(ky-wt+\delta_z)}[\cos(\alpha)\hat{e_z} + \sin(\alpha)e^{i\phi}\hat{e_x}] \tag{2.6}$$

The polarization of the electromagnetic wave can be characterized by the vector:

$$\hat{e} = \cos(\alpha)\hat{e_z} + \sin(\alpha)e^{i\phi}\hat{e_x} \tag{2.7}$$

Only two parameters, α and ϕ, are necessary to characterize a polarization state. And so the photonic qubit can be represented by a point in the Bloch sphere with spherical coordinates θ and ϕ with $\theta = 2\alpha$, and ϕ equal to the phase difference of the two field components:

$$|\alpha,\phi\rangle = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha e^{i\phi}) \end{pmatrix} = \cos(\alpha)\,|H\rangle + \sin(\alpha e^{i\phi})\,|V\rangle \tag{2.8}$$

Depending on the α and ϕ angles, different types of vectors can be obtained, each representing a polarization state. The table below summarizes the cases of interest for this work:

Table 2.1: Summarized table including the angles, vectors, and symbol information for each polarization state that is necessary for this work.

α	ϕ	Vector	Jones vector	Polarization	Symbol
90°	0°	$\hat{e}_x$	$\begin{pmatrix}0\\1\end{pmatrix}$	Horizontal	$\|H\rangle$
0°	0°	$\hat{e}_z$	$\begin{pmatrix}1\\0\end{pmatrix}$	Vertical	$\|V\rangle$
45°	0°	$\frac{1}{\sqrt{2}}(\hat{e}_z + \hat{e}_x)$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$	Linear + 45 °	$\|D\rangle$
135°	180°	$\frac{1}{\sqrt{2}}(\hat{e}_z - \hat{e}_x)$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$	Linear - 45 °	$\|A\rangle$
45°	90°	$\frac{1}{\sqrt{2}}(\hat{e}_z + i\hat{e}_x)$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\i\end{pmatrix}$	Right Circular	$\|R\rangle$
45°	270°	$\frac{1}{\sqrt{2}}(\hat{e}_z - i\hat{e}_x)$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\-i\end{pmatrix}$	Left Circular	$\|L\rangle$

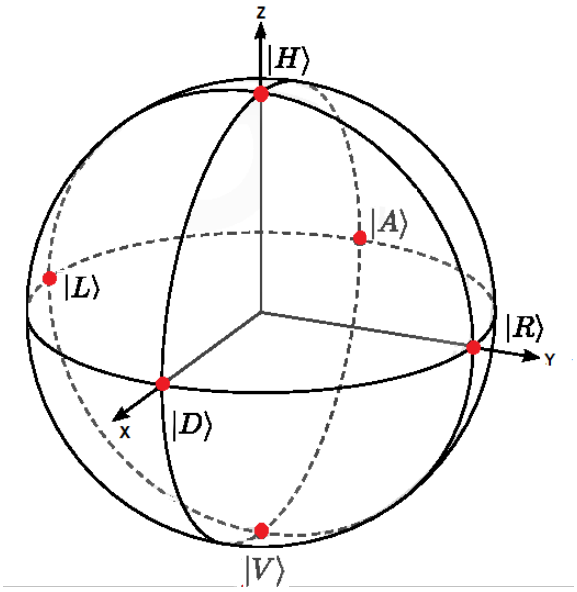

Figure 2.2: Bloch sphere containing the six polarization states in the respective axis.

It can be concluded that the $|V\rangle$ and $|H\rangle$ orthogonal states make the standard basis which can be represented by the $\oplus$ symbol. The $|D\rangle$ and $|A\rangle$ orthogonal states form a basis represented by the $\otimes$ symbol and the $|R\rangle$ and $|L\rangle$ orthogonal states are referred as the y-basis, because both are in the y direction in the Bloch sphere.

2.5 Stokes parameters

In 1852, George Stokes showed that any polarization could be represented by four measurable quantities, known as the Stokes polarization parameters. Unlike Jones vectors, the Stokes parameters are real-valued and can represent both full or partially polarized light [9]. Let us consider two plane waves that are orthogonal to each other at a point in space (x, y, z). Without losing generality, $z = 0$:

$$\begin{cases} E_x(t) = E_{0x}(t)\cos(\omega t + \delta_x(t)) \\ E_y(t) = E_{0y}(t)\cos(\omega t + \delta_y(t)) \end{cases} \tag{2.9}$$

Where $E_{0x}(t)$ and $E_{0x}(t)$ are the instantaneous amplitudes, ω is the instantaneous angular frequency, $\delta_x(t)$ and $\delta_y(t)$ are the instantaneous phase factors [1]. By removing the term ωt, the polarization ellipse equation can be obtained:

$$\frac{E_x^2(t)}{E_{0x}^2} + \frac{E_y^2(t)}{E_{0y}^2} - 2\frac{E_x(t)E_y(t)}{E_{0x}E_{0y}}\cos(\phi) = \sin^2(\phi) \tag{2.10}$$

Where $\phi = \delta_y - \delta_x$ represents the phase difference. To represent this equation in terms of the observables of the optical field, the average over the time of observation must be considered. Since the vibration is so fast, the observation time can be seen as infinite. However, $E_x(t)$ and $E_y(t)$ are both periodical, and the average of the previous equation is only over a single oscillation period. The time average is represented by the angular brackets $\langle\rangle$ and so [1] :

$$\frac{\langle E_x^2(t)\rangle}{E_{0x}^2} + \frac{\langle E_y^2(t)\rangle}{E_{0y}^2} - 2\frac{\langle E_x(t)E_y(t)\rangle}{E_{0x}E_{0y}}\cos(\phi) = \sin^2(\phi) \tag{2.11}$$

With:

$$\langle E_x(t)E_y(t)\rangle = \lim_{T\mapsto\infty}\frac{1}{T}\int_0^T E_x(t)E_y(t)dt \tag{2.12}$$

The average values of equation 2.11 can be calculated using 2.9 and 2.12

$$\langle E_x^2(t)\rangle = \frac{1}{2}E_{0x}^2 \tag{2.13}$$

$$\langle E_y^2(t)\rangle = \frac{1}{2}E_{0y}^2 \tag{2.14}$$

$$\langle E_x(t)E_y(t)\rangle = \frac{1}{2}E_{0x}E_{0y}\cos(\phi) \tag{2.15}$$

Multiplying equation 2.11 by $4E_{0x}^2E_{0y}^2$ and substituting by the average values:

$$(E_{0x}^2 + E_{0y}^2)^2 - (E_{0x}^2 - E_{0y}^2)^2 - (2E_{0x}E_{0y}\cos(\phi))^2 = (2E_{0x}E_{0y}\sin(\phi))^2 \tag{2.16}$$

And so the S0, S1, S2 and S3, real quantities can be introduced with the following relations:

$$S_0 = E_{0x}^2 + E_{0y}^2 \tag{2.17}$$

$$S_1 = E_{0x}^2 - E_{0y}^2 \tag{2.18}$$

$$S_2 = 2E_{0x}E_{0y}\cos\phi \tag{2.19}$$

$$S_3 = 2E_{0x}E_{0y}\sin\phi \tag{2.20}$$

The stokes parameters have a physical meaning in terms of intensity. The S_0 is used to represent the total intensity of light (polarized and unpolarized). The parameter S_1 describes the difference in intensity between the linearly horizontal polarized light and the linearly vertical polarized light [1]. The parameter S_2 represents the intensity of linearly polarized in the directions 45° to the reference plane. The S_3 describes the preponderance of the right circular polarized light over the left circular polarized light. The stokes parameters can form a vector that represents a polarization state.

Table 2.2: Summarized table including the amplitudes of $E_x(t)$, $E_y(t)$, phase difference, stokes vector and symbol information for each polarization state.

E_{0x}	E_{0y}	ϕ	Stokes vector	Polarization	Symbol
1	0	0°	$\begin{pmatrix}1\\1\\0\\0\end{pmatrix}$	Horizontal	$\|H\rangle$
0	1	0°	$\begin{pmatrix}1\\-1\\0\\0\end{pmatrix}$	Vertical	$\|V\rangle$
1	1	0°	$\begin{pmatrix}1\\0\\1\\0\end{pmatrix}$	Linear + 45°	$\|D\rangle$
1	1	180°	$\begin{pmatrix}1\\0\\-1\\0\end{pmatrix}$	Linear - 45°	$\|A\rangle$
1	1	90°	$\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$	Right Circular	$\|R\rangle$
1	1	-90°	$\begin{pmatrix}1\\0\\0\\-1\end{pmatrix}$	Left Circular	$\|L\rangle$

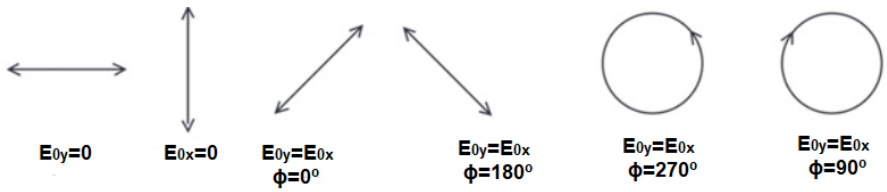

Figure 2.3: Representation of how the electric field oscillates with respect to the amplitude of $E_x(t)$, $E_y(t)$ and the phase difference.

When the stokes parameters are used as coordinates in a three-dimensional space, all physically possible states fall within a sphere of radius one. The Poincaré sphere is a powerful graphical tool that can represent any polarization state as a three-dimensional point. The origin point is considered as the unpolarized light, and each point at the surface represents a pure state of polarization. The right circular and left circular polarization states are represented at the north and south poles (S3 axis), respectively. Points along the equator represent the linearly polarized states of varying orientations, with notable cases at the axes with the horizontal, vertical polarization states (S1 axis) and the linear +45° and the linear -45° (S2 axis). Other points, with intermediate elliptical states are continuously distributed between the equator and the poles. The evolution of the state of polarization can be represented as a continuous point path on the sphere [9]. The graphical Poincaré sphere description, allows a more intuitive approach to polarization mathematics. Here the same analogy can be made as the Jones vector where $|V\rangle$ and $|H\rangle$ orthogonal states make the standard basis which can be represented by the $\oplus$ symbol and the $|D\rangle$ and $|A\rangle$ orthogonal states form a basis represented by the $\otimes$ symbol.

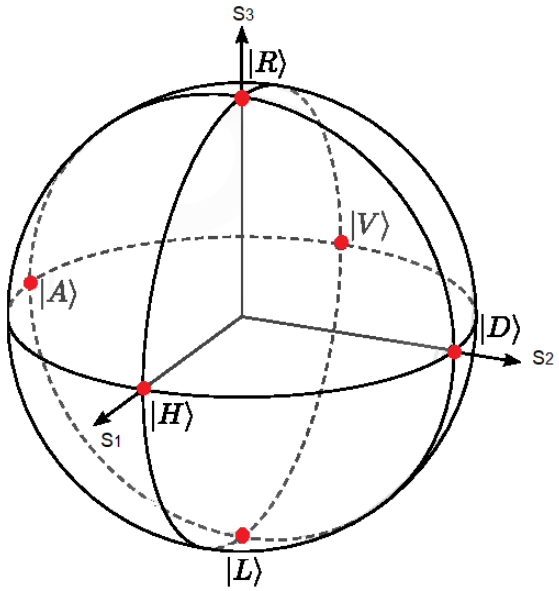

Figure 2.4: Poincaré sphere containing the six polarization states in the respective axis.

2.6 Single photon measurement

Photons are traditionally detected by converting their energy into an electrical signal. One arrangement designed to measure the polarization state of single photon, is shown in figure 2.5 This system consists of a polarizing beam splitter and two single photon detectors, D1 and D2. The PBS has the ability to transmit vertically polarized light, while diverting horizontally polarized light through a 90° angle. Depending on the initial polarization state, the particle is either deflected up or down. For example, if the incoming photon is vertically polarized, the photon is going to be registered in the D1 detector. Similarly, if the incoming photon is horizontally polarized the D2 detector is going to register the photon. In all other cases we have to dissect the polarization vector into the vertical and horizontal components, and check its amplitude probabilities. As mentioned before, the initial state of the photon with an arbitrary polarization angle, can be written as a superposition of the two orthogonal polarization states according to equation 2.8. The probability that the photon is transmitted to D1 is given by $\cos^2(\alpha)$, and the probability that it is transmitted to D2 is given by $\sin^2(\alpha)$. Supposing that Eve was able to intercept the system with her measurement apparatus, she is going to try to determine α and then transmit a similar photon to Bob. In each measurement, Eve is able to check whether detector D1 or D2 registers, with the probabilities previously mentioned. However, when she tries to deceive Bob, by transmitting a similar photon, she can only do it in special cases where $\alpha = 0°$ and $\alpha = 90°$. For all other values, the act of extracting information about the polarization, leads Eve to transmit her photon with a different polarization angle to the first one, reveling her presence in the system [7]. When the detectors receive the single photons, they can output an electrical signal. With a processing unit collecting the data, it is possible to set counters, to see which detector clicked at any given time. Due to the dark counts associated with them, both detectors can either click at the same, or not click at all.

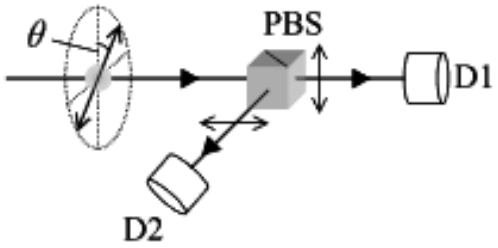

Figure 2.5: Apparatus to measure the polarization state of a single photon using two single photon detectors and a polarizing beam splitter [7].

2.7 BB84 protocol

In the BB84 protocol, Alice and Bob start their communication by exchanging information through the quantum channel. Alice sends her encoded qubits through this channel to Bob, so he can measure them. In the end, Alice and Bob communicate via a classical channel to check if there was any eavesdropper attacking the system. The table example 2.3 illustrates how the process of communication is executed.

Table 2.3: Representative random sequence of data choices to implement the BB84 protocol. The $\oplus$ symbol represents the basis with vertical and horizontal polarization states. The $\otimes$ represents the basis with linear +45° and linear -45° states.

Alice random bit	0	1	1	0	1	0	0	1
Alice random basis	$\oplus$	$\oplus$	$\otimes$	$\oplus$	$\otimes$	$\otimes$	$\otimes$	$\oplus$
Alice Qubit	$\|0_\oplus\rangle$	$\|1_\oplus\rangle$	$\|1_\otimes\rangle$	$\|0_\oplus\rangle$	$\|1_\otimes\rangle$	$\|0_\otimes\rangle$	$\|0_\otimes\rangle$	$\|1_\oplus\rangle$
Bob random basis	$\oplus$	$\otimes$	$\otimes$	$\otimes$	$\oplus$	$\otimes$	$\oplus$	$\oplus$
Bob qubit	$\|0_\oplus\rangle$	?	$\|1_\otimes\rangle$	?	?	$\|0_\otimes\rangle$	?	$\|1_\oplus\rangle$
Basis reconciliation								
Sifted key	0		1			0		1

For the first step Alice generates a random bit sequence, for example, 0 1 1 0 1 0 0 1. She then generates a new bit sequence which identifies the basis $\oplus$ or $\otimes$. For the bit sequence, 00101110, the basis sequence is going to be $\oplus \oplus \otimes \oplus \otimes \otimes \otimes \oplus$.

Alice encodes her qubits according to the previous sequence, and sends them to Bob through the quantum channel (optical fibers). In order to read the qubits, Bob does his measurements in the $\oplus$ basis or in the $\otimes$ basis. To do that, he generates a random bit sequence which is going to identify the basis used to read the qubit. For example, 01110100, means basis $\oplus \otimes \otimes \otimes \oplus \otimes \oplus \oplus$. When Bob finishes his measurements,

he communicates with Alice through the classical channel so they can share the two random bases bit sequences. Now it is public which cases the two of them coincide. Nevertheless, an eavesdropper cannot obtain information of the actual bit value.

The public knowledge of the basis sequence does not allow the private key to be discovered. If there were no errors in the transmitter and receiver side, a mismatch in Alice and Bob's sequence would mean that the intruder, Eve, was able to do measurements in the quantum channel. However, the discrepancies in the nearly identical shared keys, can also be caused by the errors associated to the transmitter and the receiver equipments. At this point, it is still impossible to distinguish between these errors and the ones introduced by Eve. To guarantee the security of the system, it is assumed that all errors are introduced by the eavesdropper. Provided that the error rate between the keys is lower than a certain threshold ($\sim 11\%$), two further steps can be implemented to remove the bit errors and eliminate Eve knowledge of the key.

Information reconciliation is a form of error correction, performed on Alice and Bob's keys, to ensure that they are identical. The cascade protocol is used to correct errors based on a block parity exchanges [20].

Privacy amplification is a method used to eliminate Eve's information about Alice and Bob's key. This information could have been gain by eavesdropping on the quantum channel during the keys transmission, and on the public channel ,during the basis reconciliation. At this point Alice and Bob have acquired identical strings, but they are not completely private. Privacy amplification uses Alice and Bob's key to produce a new shorter key, to eliminate all possible information leaks. This can be done using a universal hash function, chosen at random from a publicly known set. This function takes as input a binary string of length equal to the key and outputs a binary string of a chosen shorter length. The amount by which this new key is shortened, is calculated based on how much information Eve could have gained about the old key (which is known due to the errors introduced in the system). With this method it is possible to eliminate the probability of Eve having knowledge of the new key [18].

2.8 Some system errors

In a quantum key distribution implementation, Eve is not the only one who introduces errors in the system. Since the technology equipments are not perfect, the inaccuracies introduced by instruments needs to be accountable, in order to create an error threshold.

One type of error is the birefringence, and it occurs when the medium in which the photons are traveling is birefringent. This is going to affect the polarization angle, forcing it to change as the photon travels from Alice to Bob. In this case, Bob is going to get the wrong result in his detectors, even though he chose the right basis for the measurement. This type of error can be calibrated out of the system by using classical error-correction algorithms [7]. Optical fibers can also introduce losses, which decays the intensity of the optical beams as they propagate. These propagation losses depend on the wavelength used. There are three common wavelength bands used in fibre optic systems: the 850 nm band, which has larger scattering losses; the 1300 and 1550 nm bands, which have detectors with much higher dark count rates, and are affected by afterpulsing that restricts the transmission bit rate[7]. The gain/loss needs to analysed, to see which wavelength is better supported in the system.

Another type of error is the detector dark counts. It happens when the photon sent by Alice never reaches Bob and the wrong detector randomly registers, due to thermal noise. This error, has the same effect has the birefringence one, but can also be calibrated by using classical error analysis on a portion of the sifted bits [7] .

On the transmitter side, it is important that Alice generates only one photon to encode the information. If she were to send more than one photon at a time, she could leak information to Eve. For example, if Alice sends light pulses containing two photons instead of one, and Eve is detecting them with the wrong basis, there is a 50 % chance that the two photons are going to be registered by both detectors. Eve would know with certainty that she is using the wrong basis, giving her an advantage, because the error rate introduced by her would decrease. This problem gets worse the more photons Alice sends in a light pulse. If there are three photons per pulse,

Eve can determine both the basis and the bit value, for a significant fraction of the data pulses [7]. The standard procedure to avoid these problems, is to take a pulsed laser and attenuate it so that the mean photon number per pulse is very small. When this value is small, most of the time intervals will contain no photons, a small fraction will contain one photon, and a very small number will contain more than one photon.

2.9 Polarization encoding

A typical quantum key distribution system, using the BB84 four state protocol, utilizes the photon's polarization to encode the information. The transmitter can contain four oriented laser diodes, each one emitting short classical photon pulses, polarized at linear $-45°$, horizontal, linear $+45°$ and vertical states. [9]. The pulses are then attenuated, to reduce the average number of photons, and sent to Bob through the quantum channel. It is important that the pulses remain polarized, so that Bob can extract the correct information encoded by Alice. When the photons reach the receiver, they travel through a set of waveplates to recover the initial polarization state, compensating the errors introduced by the optical fibers. The pulses then reach a symmetric beamsplitter, representing the basis choice step. Transmitted photons are analyzed in the vertical-horizontal basis with a polarizing beamsplitter and two single photon detectors. The polarization state of the reflected photons is first rotated by 45 ° with the help of a waveplate. The photons are then analyzed with a second set of polarizing beamsplitters and detectors, implementing the diagonal basis. Another set up for polarization encoding is to use one single laser diode in combination with two active polarization cells, denominated as Pockels cells. At the transmitter, the modulator is randomly activated for each pulse to rotate the state of polarization to one of the four states. At the receiver, it randomly rotates half of the incoming pulses by 45°, recreating the choice step.

Chapter 3

Receiver Optimization

3.1 Introduction

The quantum communication system implemented in the laboratory uses the polarization property of light to cipher and transmit information. These types of systems can use up to three non-orthogonal bases to encode information: standard linear, diagonal linear and circular polarization's. Necessarily, the receiver side needs to be able to commute between the three non-orthogonal bases, so that Bob can choose one to do his measurements. In this work, an Arduino based solution is proposed, which is going to optimize the reception procedures.

This chapter starts with the project definition, where the goals and the architecture of this implementation are explored. Afterwards, the decision process is discussed, namely the components selection, and how they can be implemented. Next, an overview of the technology used is presented, to specify the details of the electronic equipments. The final implementation using Arduinos is then illustrated and explained. The repository, where all programs are stored, is described, together with the three processes used to have the system working: Signal processing, Communication with the upper level protocol and managing the electronic polarization controller (EPC). Lastly, some QBER results were presented, to check if the receiver module could remain stable for certain amount of times.

3.2 Project definition

In order to start looking at solutions for the problem proposed in this chapter, we needed to describe what was already implemented in the laboratory. This is

an important step, because one can be familiarized with the system limitations, performance and what is really necessary to be achieved. Figure 3.1 illustrates the main components of the receiver in the initial configuration. The clock signal is used to keep both quantum detectors synchronized. In each clock cycle, the detectors open their gates and output a voltage signal if a photon was measured, or output nothing if a photon wasn't measured. However, this feature was not being used. In this implementation, Bob needs to manually adjust the electronic polarization controller, to change the basis for his measurements. Here the state of polarization is converted, so that the polarization beam splitter can distinguish states with the help of the single-photon detectors.

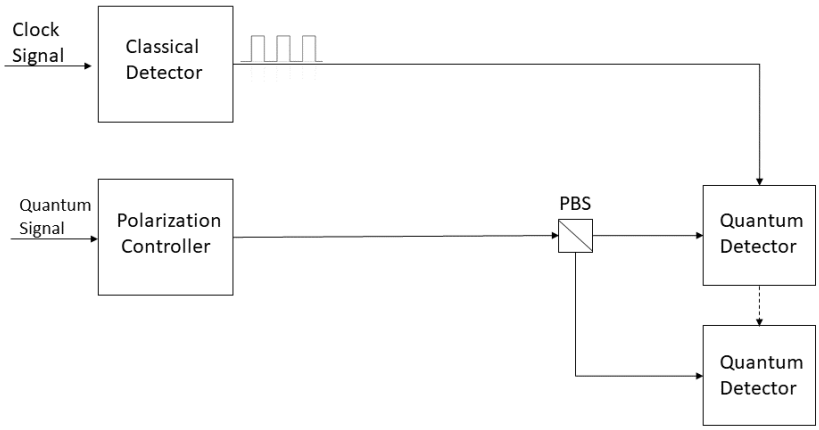

Figure 3.1: Block diagram of the initial components in the receiver. Here the clock signal synchronizes the detectors, and the EPC is adjusted manually. The PBS splits the light between the detectors according to the basis selection.

To improve the receiver side, the following ideas were prioritized: the system needs an autonomous module capable of processing the detectors data, communicate with an upper level protocol, where the error calculations are performed, and act accordingly on the electronic polarization controller.

The receiver side requires a type of electronics that can be controlled automatically, and are capable of implementing the ideas mentioned above.

Basically, the earlier implementation needs a microcontroller board to interpret the data directed to the I/O (input/output) peripherals, and analyse it using the central processor. After the analysis, the board needs to communicate the appropriate actions through the other I/O peripherals. The module chosen for this work was the Arduino DUE, because of its clock speed and the wide range of peripherals. Every time the detectors measure a single photon, they can output a controlled voltage signal through their gates. This gate opens when the clock signal is in the high state and closes when the clock signal in the low state. The Arduino reads the voltage signals, every clock cycle and does a coincidence check. If none of the detectors clicked, the Arduino is going to send an “3” to the upper level protocol. If both clicked, the Arduinos sends a “2”, and depending on if one clicked and the other did not, it sends a “0” or a “1”. The coincidence test information is then forwarded to the upper level protocol, where the QBER is calculated in real time. This protocol also sends back to the Arduino a "0", "1" or "2 representing the three set of voltages to apply on the polarization controller. Each set of voltages represents one of the three non orthogonal basis, that Bob needs to apply to do his measurements.

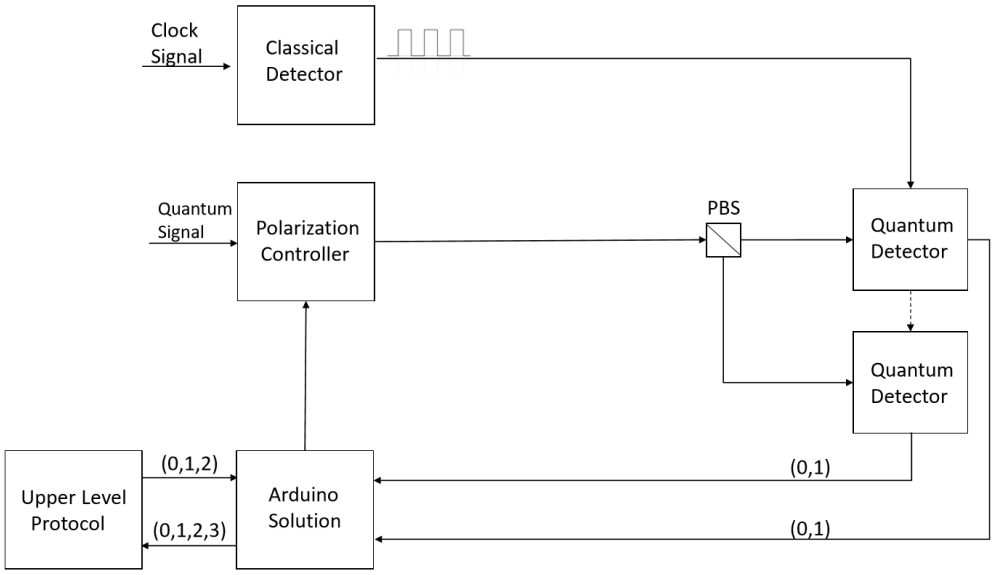

Figure 3.2: Receiver High level diagram block. The receiver module represents all the electronics that are going to be needed for the signal processing, data analysis and for controlling the EPC’s.

3.3 Project decisions

Even though the Arduino DUE is used as the main structure of the receiver module, it was also necessary to make decisions concerning other types of electronics. First, the capability of the Arduino, to measure the voltage signals from the quantum detectors was tested. The Arduino needs to be handled in interruption mode, since these signals have a low duration of 10 ns or 100 ns. This mode implies that Arduino interrupts its actions every time a voltage signal is detected on the digital pins. Each clock cycle, the Arduino is going to check if there is any voltage signal from the quantum detectors gates. After the respective coincidence check is done, the Arduino communicates with the upper level protocol. Therefore, the integers 0, 1, 2, or 3 are going to be sent through the serial port to a computer, via USB connection. The next step is to have a computer run the calculations and send back the information needed to act on the electronic polarization controllers. Even so, this step introduces two new problems regarding the Arduino module. Firstly, the same Arduino cannot be sending and reading information through the same serial port, because it is going to lose bits over time. Secondly, the polarization controller needs 4 voltages in order to be operated, and the Arduino only has two integrated DAC's (Digital to analog converter). For the first problem, instead of one Arduino, a second Arduino is added to the implementation. Arduino DUE 1 is only going to be reading the detectors data, and sending it to the computer. Arduino DUE 2 is going to be reading the data from computer, and act on the polarization controller. For the second problem, an external DAC, compatible with the Arduino DUE, with eight voltage outputs, is going to be used.

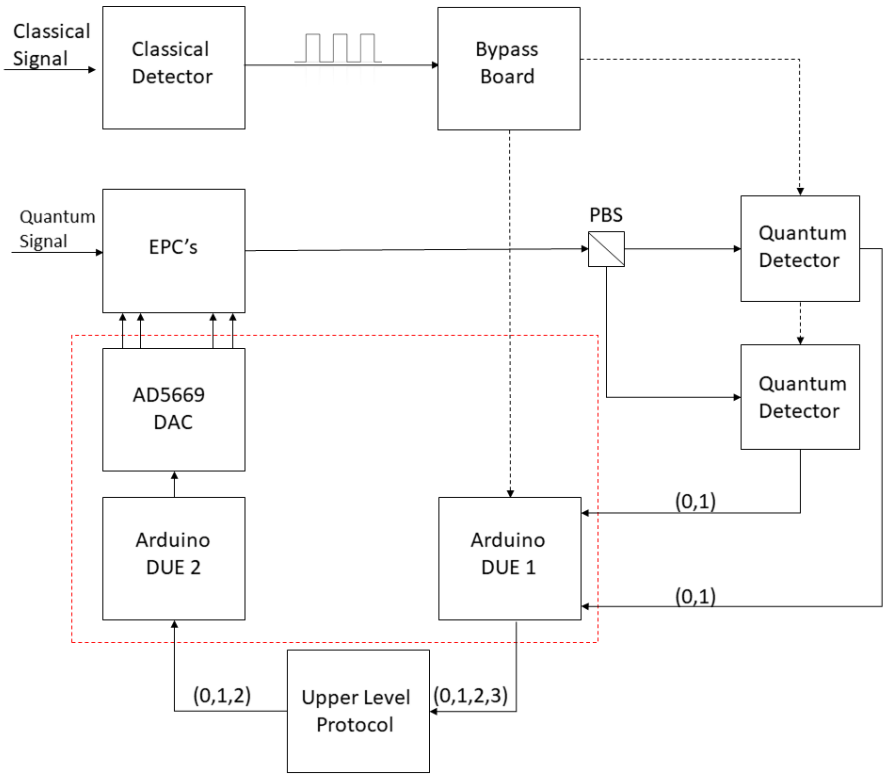

Figure 3.3: Receiver block diagram after the implementation modifications. The components that are needed for the new implementation are highlighted in red: two Arduinos DUE, a computer and an external DAC. This highlighted area represents my contribution to the work in the receiver.

3.4 Technology equipment

A further look in each separate electronic device was made:

- **Detectors**: The photon counting is done by two ID Quantique's single photon detectors modules. The core consists of an InGAaS/InP avalanche photodiode (APD). In order to reduce the dark count probability, the APD is cooled using a thermoelectric cooler. The APD is operated in the gated mode, where a voltage pulse is applied to raise the bias beyond the breakdown point. This type of technology allows a single photon sensitivity up to a wavelength of 1650 nm. The performance of an APD is characterized by the probability for a photon impinging on the photodiode to be detected. At 1550 nm, which is the wavelength used transmitter source, a 25% detection efficiency is typical. In the APD, avalanches can also be triggered by carriers generated in thermal, tunneling or trapping processes taking place in the junction. These

self-triggering effects are called dark counts, and represent the probability as a function of the detection efficiency at a temperature of 220 K. At 10% detection efficiency, the dark count probability is typically $6 \times 10^{-5} ns^{-1}$ or less. The main problem limiting the performance are the afterpulses. This spurious effect arises from trapping of charge carriers during an avalanche by trap levels inside the high field region of the junction where impact ionization occurs [19]

Figure 3.4: ID Quantique single photon detectors

- **Arduino DUE**: Arduino represents an open-source platform that consists of both a physical programmable circuit board, containing a microcontroller, and an Integrated Development Environment that runs is a computer through a USB cable. The Arduino model used for this project was the DUE. It has a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. It also has 54 input/output pins, 12 analog inputs, 4 UARTs (universal asynchronous receiver-transmitter), 84 MHz clock, 2 DAC, a power jack, a SPI (Serial Peripheral Interface) header, a reset button and an erase button. However, this model is limited at 3.3V for the maximum voltage that the I/O pins can tolerate [3].

  According to Arduino official manufactures this language is merely a set of C/C++ functions that can be called from the user's code. The users sketch undergoes minor changes and then is passed directly to a C/C++ compiler.

However, there's currently no support for the standard support library needed for a complete C++ implementation. This imposes a number of restrictions on the C++ programs that can be compiled. Among them are:

- Some of the C++ related standard functions, classes, and template classes are available.
- The operators new and delete are not implemented, attempting to use them will cause the linker to complain about undefined external references.
- Some of the supplied include files are not C++ safe.
- Exceptions are not supported.
- The outputs to the console have to be made through the Arduino serial monitor using the function Serial.print() instead of making use of cout, cin or cerr classes.
- Arduino does not recognize objects of type std::initializer_list<T>, which are a lightweight proxy object that provides access to an array of objects of type const T, so instead vector containers must be used in order to initialize the constructors attributes.

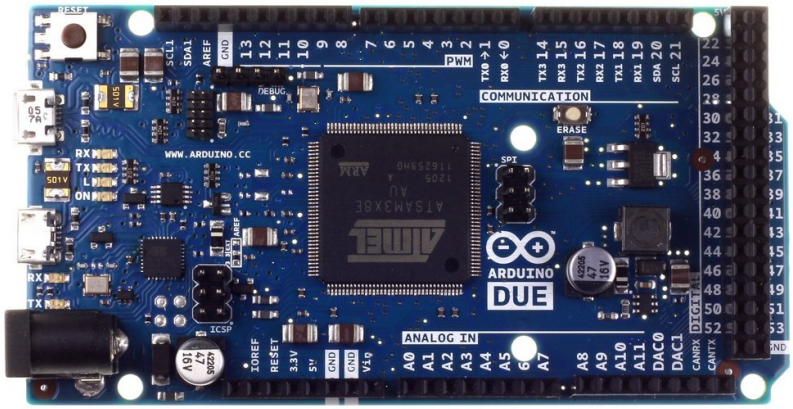

Figure 3.5: Arduino DUE board

- **AD5669** : The AD5669 is a I2C (Inter-Integrated Circuit) high-resolution digital to analog converter capable of generating a 0-5V voltage output. It has a 16-Bit resolution and it can tune the output across 65,536 steps. It is equipped with 8 individual output channels and one floating address line. The AD5669 is capable of 400 kHz communication speed, making it ideal for programmable voltage and current source applications. It is compatible with the Arduino by including the Wire.h library and defining the DAC address in the program. In the Arduino program we set the correct address of the AD5669. It also requires two bytes of data for the DAC and a command byte that controls all the various functions. The clock chosen was 400 kHz, to achieve maximum speed, and four channels were set to output random voltages. Using the Wire library functions, the I2C transmission is established in order to control all four channels accordingly [5].

Figure 3.6: AD5669 DAC board

- **Electronic Polarization controller**: Polarization controllers are normally operated by manual/electrical adjustments or with an automatic feedback. A polarization controller can transform a fixed, known polarization into an arbitrary one. In the reception side, these are used to compensate the random polarization drift from the quantum channel and used to choose in which base the single photons are going to be measured. The ones used in the laboratory need four voltage inputs in order to operate. Since these are electro-mechanical, it is possible to get different results when the same voltages are applied in different occasions.

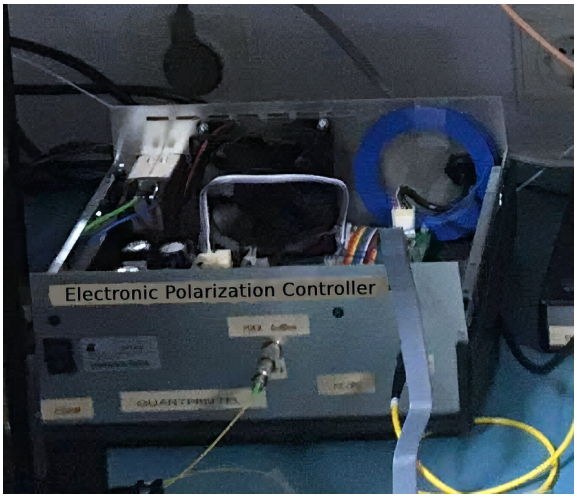

Figure 3.7: Electronic polarization controller at the laboratory

- **Bypass board**: This custom-made board, takes the signal coming from the classic detector and divides it to the two single photon detectors and the Arduino. This signal represents the clock and keeps the system synchronized.

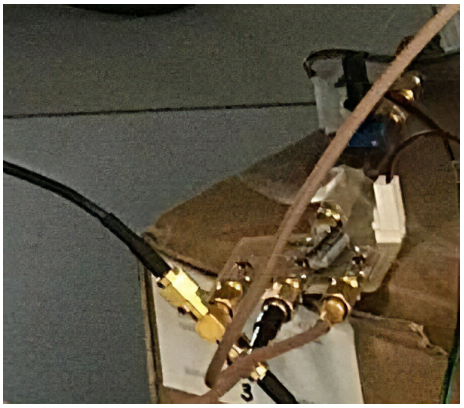

Figure 3.8: Bypass board at the laboratory

- **Polarizing beam splitter**: Polarizing beam splitters use birefringent materials to split light into two beams of orthogonal states. They are designed to split polarization states, rather than dissociation by wavelength or intensity. They are typically manufactured for 0º or 45º angle of incidence with a 90º separation of the beams, depending on the configuration.

3.5 Arduino implementation

In terms of simplification, the implementation can be divided in three steps. The first one represents the signal acquisition, where the Arduino DUE 1 collects the data from the detectors and does the coincidence check. The second stage corresponds to the data analysis, where it is possible to get the QBER reports and the necessary information to act on the EPC. These calculations are performed by a computer located in the laboratory. The third and final step, is to set the correct voltages on the EPC with the help of the Arduino DUE 2 and an external AD5669 DAC.

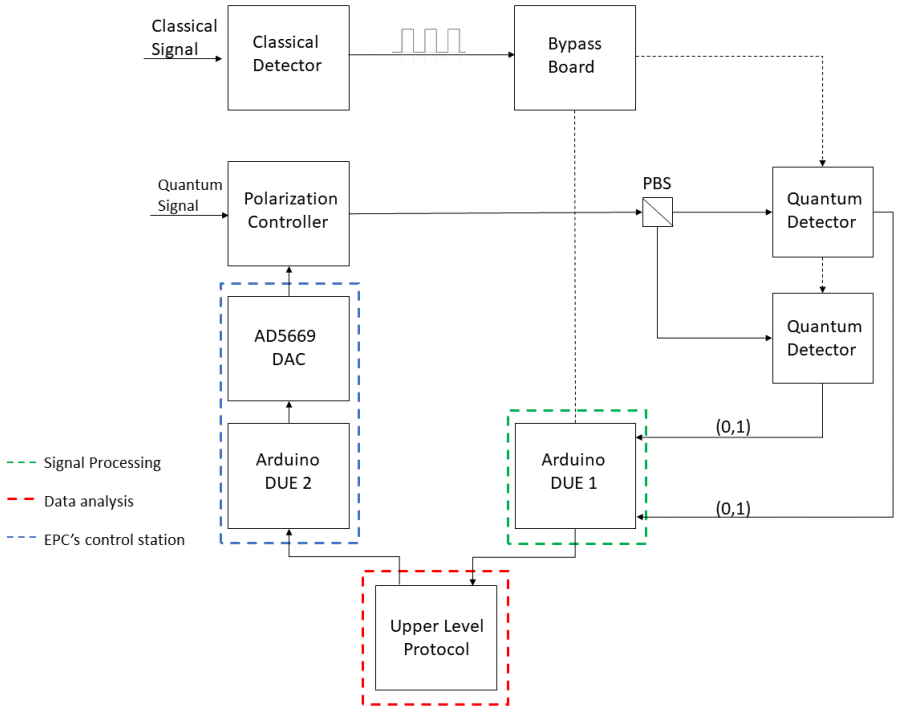

Figure 3.9: Block diagram of the implementation, where the three mains processes are highlighted. The green area is responsible for the signal processing. The red area represents the location where the data is going to be analyzed. Finally, in blue are the components used to control the EPC.

3.6 Repository

Due to the intended nature of this work to be open source, a guide to where all the programs are located is needed. This work was developed in the Netxpto repository. Under the Arduino folder, all the necessary tools to run the software side are

presented. In the “sdf”, the arduino_real_time_receiver folder can be found. Here the three main codes are referenced as ardRece, serial_port_reader_writer-Client and netxpto_qber_estimation-Server. The first one is the Arduino code, that needs to be opened and uploaded to the Arduino DUE 1 and Arduino DUE 2. It is only necessary to run one program, because it is possible to code with two Arduino ports within the same workplace. The second program, is a C++ script that reads data from Arduino DUE 1, and writes data to the Arduino DUE 2. It also communicates, via TCP/IP (Transmission Control Protocol/Internet Protocol), with the netxpto_qber_estimation-Server program. The third one, is continuously receiving the coincidence check data, and outputs a file report each 8000 bits, with the QBER results. For the system to be successfully running, all the three programs need to be running simultaneously. Further detailed explanation is presented in the following section of this chapter.

3.7 Signal acquisition

As mention before, one Arduino is going to read the signals from the quantum detectors, each clock cycle, while sending this data to the upper level protocol. For the signal acquisition two options were considered: polling mode or the interruption mode. Polling proved to be impracticable to implement, since the quantum detectors output a signal with a maximum width of 100 ns. In contrast, the Arduino function, digitalRead(), needs a minimum time of 523 ns to read a voltage signal. This proves the impossibility to read two simultaneous signals, and the use of this function (polling) would lead to the loss of information each cycle. The right approach is to use an interruption every time a rising edge coming from the single photon detectors, or the clock signal is caught. To do this, all the receiving signals are read in the digital input pins of the Arduino DUE board. This type of reading doesn’t require any signal conditioning, which means the detectors gates are directly connected to the Arduino. Since the detectors have the feature to control the output signal voltage, it is important to set it below the 3.3 V Arduino Due threshold.

One important note, is that the clock trigger reaches both the detectors and the Arduino at the same time. In order to detect it, the Arduino does an interruption when it catches a rising voltage and puts a Boolean to true, meaning it is ready to take the signals from the two detectors gates. The same applies to the signal coming from the detectors gates, meaning an interruption function is going to be triggered if the digital pins detect a rising voltage. Each clock cycle, if the Arduino has both detectors Booleans to true, the integer 2 is sent to the computer. If both detectors Booleans are false, the integer 3 is sent to the computer. Depending on if one detector clicked and the other did not, one Boolean is going to be true and the other false, and the Arduino is either going to print a 0 or 1.

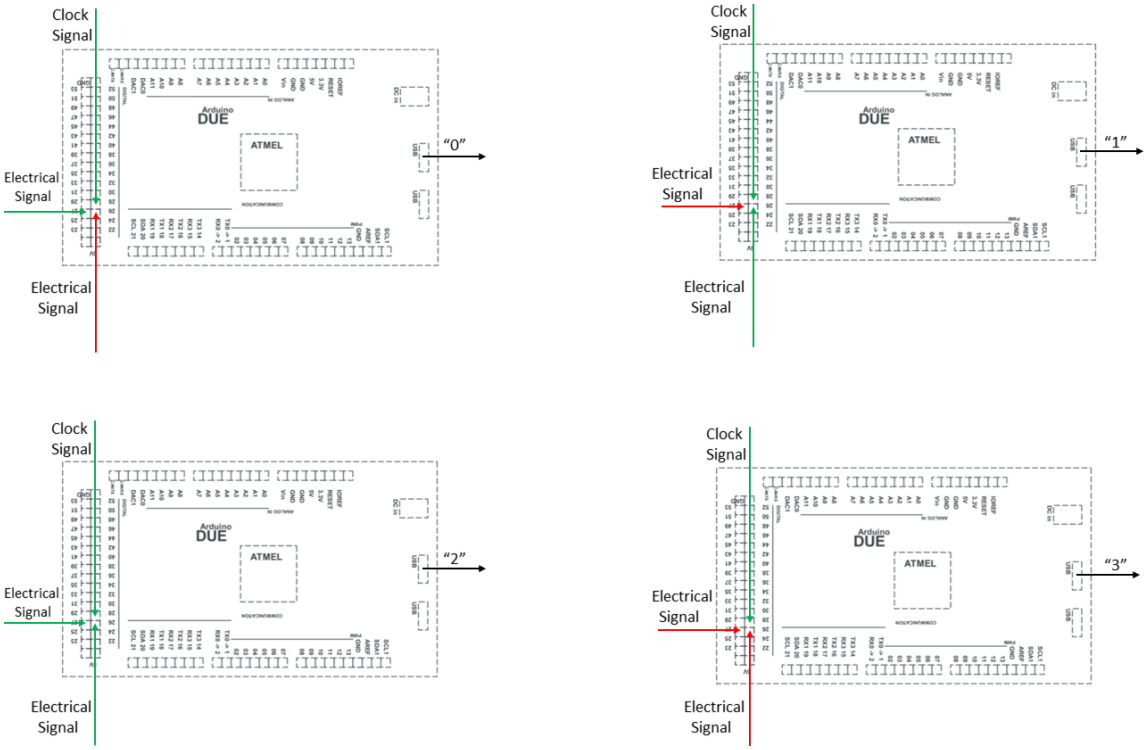

Figure 3.10: Diagram representing all four possibilities for the coincidence check block. The arrows are pointed to the digital inputs 26,27,28 where the detectors and clock are connected. The green error corresponds to the situation where a signal is sent to the Arduino. The red arrow means no signal is sent to the Arduino. Depending on the detector that sends the signal, the Arduino outputs a "0" or a "1" through the USB to the computer. If both detectors clicked a "2" is sent, and if none of them clicked a "3" is dispatched.

3.8 Communication with the computer

The Arduino program has the 26, 27 and 28 pins in the INPUT mode, waiting for an external signal from the clock or the detectors. Each one of the pins, is attached to an interruption that takes place when a rising edge is caught. When an interruption is triggered, the program does a function that puts a Boolean to true. There are four possible results that are going to be printed in the serial COM (communication port). Even though, the attachInterrupt() function takes about 2 microseconds to be completed, this does not represent a problem, because the Arduino saves the register of another event during an interruption. Each clock cycle, the program does one print and resets all the Booleans to false. After printing a value of the coincidence block, the data must be delivered to a computer, so that the QBER can be calculated in real time.

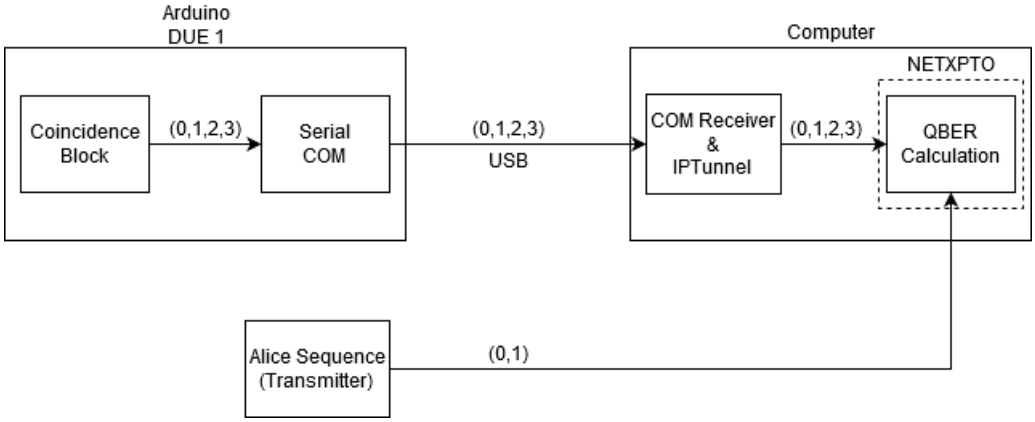

Figure 3.11: Block diagram of the communication between the Arduino DUE 1 and the computer server. After the coincidence check, the Arduino writes the data in the Serial COM. A program on the computer is going to be reading this COM and buffering the data. When the buffer is full the data is sent to the QBER program, via TCP/IP.

To have this implemented in the system, the Arduino must send the data one by one to a C++ program on the computer. The serial_port_reader_writer.sln program (client side), waits for a connection with the Arduino Serial port, in order to start receiving the data. Once the communication between the Arduinos and the computer is established, the program burns the first 10 seconds of information, because the Arduino tends to send unnecessary bits when it starts writing in the serial COM. The program puts the receiving information in a buffer, and when it's

full, all the data is sent to the QBER program. This is achieved by a TCP/IP connection between the program reading the Arduino data and the one that does the QBER estimations. The size of the buffer is 4000 characters. To always have the correct values of the QBER, the sequence the Arduino DUE is sending needs to be synchronized with the sequence from the Alice side (it's written in a text file). The netxpto_qber_estimation program runs between two modes, the synchronized one, and the unsynchronized. After burning the unnecessary data, the first 1000 bits are checked in the buffer and compared with the transmitter sequence. If they are different, the program will discard the first bit that entered the sequence and add a new one (FIFO, first in first out).

This occurs until a match in the sequences is found, and the program will run normally in the synchronized mode. If, for some reason, the program loses the synchronization, it will do the same process previously explained. In the two modes the program always output a file, the midreports, every 8000 bits, with the results of the QBER block calculations.

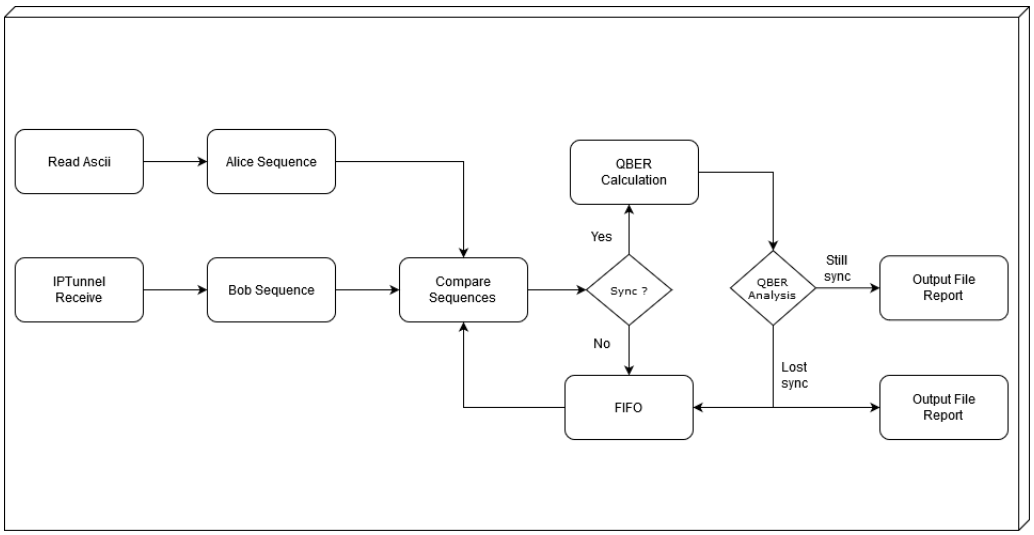

Figure 3.12: High level block diagram of the serial_port_reader_writer.sln script. The program tries to connect to the Arduino ports to be able to read and write in the serial COMs. When the connection is made, it burns the first 10 seconds of data. Then it starts to put the data into a buffer and sends it to the QBER program through an TCP/IP protocol. A counter is increasing each time a max length buffer is sent to the IPTunnel. When it reaches 500, the program writes random values of voltages to the Arduino Due 2.

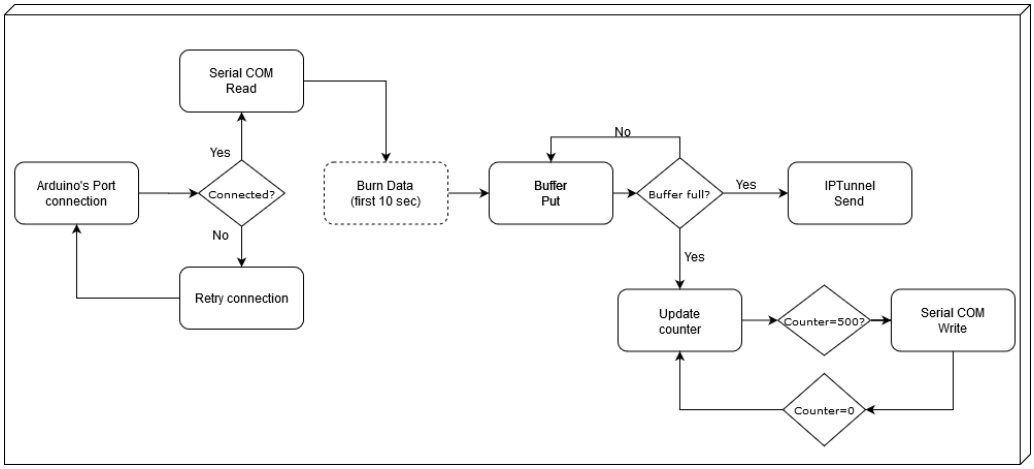

Figure 3.13: High level block diagram of the netxpto_qber_estimation-Server script. The program takes the first 1000 bits of data from the buffer (that was sent from the serial_port_reader_writer.sln script) and compares it with the Alice sequence (this sequence is in a file called binary). If both sequences matched, the QBER calculations are performed. If not, we take the first bit that was put in the Bob buffer, and replace it with a new one, until we have a match. The QBER analysis, also provides a way to check if synchronism was kept and provides information by outputting a file.

3.9 Electronic polarization controller

Bob also needs to control in which basis he is going to measure Alice qubits. To have the bases commuting between them, the EPC needs a set of four voltages. As mentioned before, for this step an extra Arduino board coupled with an external DAC is needed. The Arduino DUE 2 is only focused on receiving data from the computer and set the voltages on the AD5669 DAC. Since the algorithm for the base selection, and to compensate the polarization drifts is not developed, only the synchronization of the communication processes can be tested. However, the Arduino DUE 2, is still going to receive four random voltage values, every time the counter reaches 500 (the counter increases each time a full buffer is sent to the QBER program), simulating the real process. The Arduino communicates with the DAC through an I2C protocol. In the Arduino code, the wire library needs to be included, in order to use the DAC functions to begin the transmission, set the clock speed, write the voltage data, and to end the transmission. The process of sending data through the computer to the serial COM is also limited by its performance. This means, at certain frequencies of the clock, the computer might be doing to many internally operations, affecting

the synchronization. The speed limit is around 2.5 kHz, which does not represent a problem at the moment, because the system is working at 500 Hz.

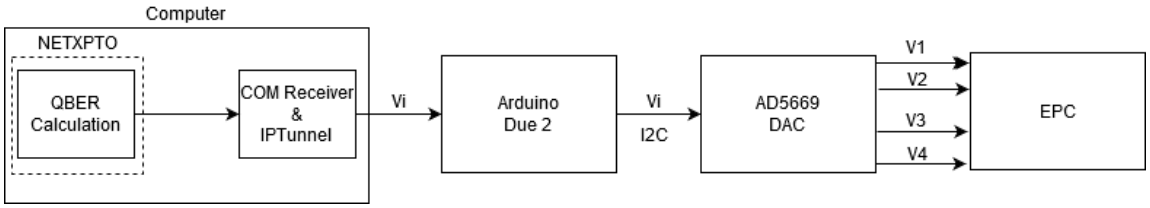

Figure 3.14: Block diagram representing the communication between the computer and the Arduino DUE 2. The COM receiver program is responsible to give the voltage values, Vi, to the Arduino. The Arduino starts the I2C connection with the AD5669 DAC to set the voltages on the EPC.

3.10 QBER results

To check if the system is able to keep the synchronization throughout all the receiver processes, some QBER results were obtained. The transmitter was working at the 500 Hz speed, and the DAC was not operating on the EPC. However, the upper level protocol, was still sending random voltage numbers to the Arduino DUE 2, and consequently to the DAC, simulating the real process. Each 8000 bits, a file containing the QBER, the number of errors, the number of double clicks, the upper and lower confidence bounds for 95% confidence level and the number of double/no clicks is created in real time. By analyzing each file it is possible to obtain the QBER progression in relation to the elapsed time. In theory, the sequence Alice is sending, should be the same Bob is getting. However the errors associated to the devices doesn't allow the QBER to be the ideally 0%. The QBER values obtained range from 3 to 6 %, due to the non ideal behaviour of the equipments.

The first test was performed for the 30-minute mark. Here the QBER values don't oscillate from high to low values, meaning the receiver module is able to keep the communications with the Arduinos, the upper level protocol and the DAC synchronized over the 30 minutes check mark.

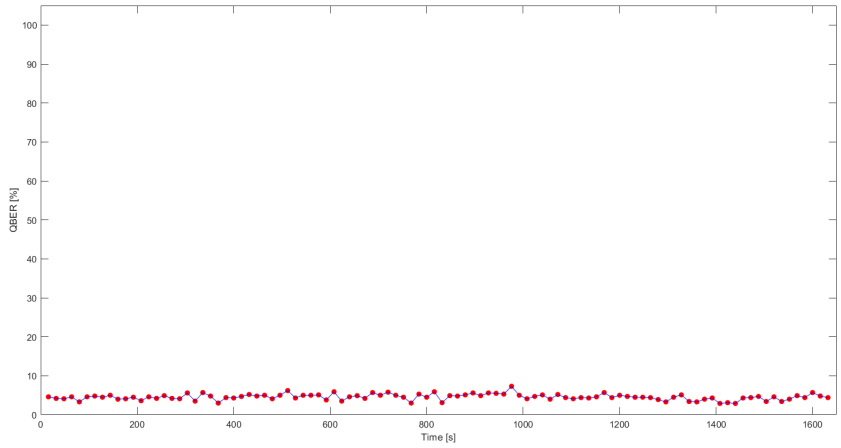

Figure 3.15: QBER progression for about half an hour.

The second test was performed for about 2 hours. Here the QBER values also don't oscillate from high to low values, meaning the receiver module is able to keep the communications with the Arduinos, the upper level protocol and the DAC synchronized for about 2 hours.

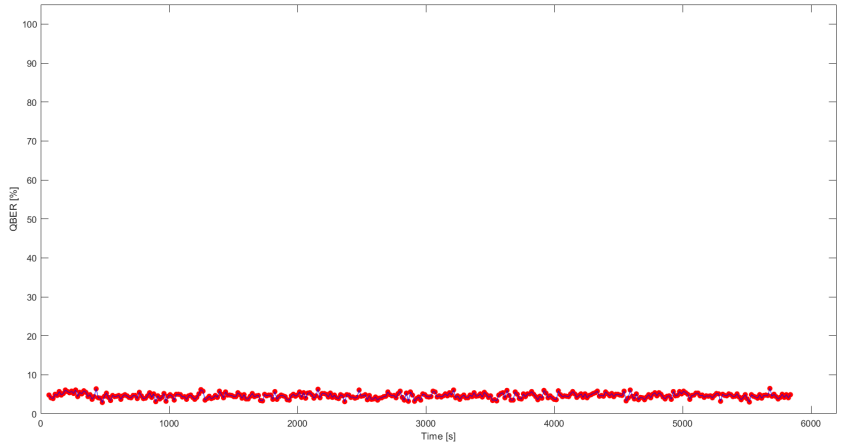

Figure 3.16: QBER progression for about two hours.

The results of stability are satisfactory at this stage of the system. With speeds up to 2.5k kHz the system should be able to reproduce the same results with no problems.

Chapter 4

Transmitter Design Optimization

4.1 Introduction

One important task to implement the BB84 protocol, is to generate and commute different states of polarization in the transmitter. Along the years, there has been a strive to improve the implementations of the quantum key distribution protocols. It's still necessary to overcome a number of technology limitations, namely, the increase of key generation rates and reduce the cost/complexity aspect of these systems. In this chapter a solution is studied, in order to improve the current implementation in the laboratory, which uses an Electronic polarization controller (EPC) to commute between different bases. This EPC consists of four individual fiber squeezers: the first and third have the main axis align with 0°, and the second and fourth have the main axis aligned with 45°. Due to the mechanical nature of the fiber squeezers and environmental disturbances, there is an instability in the direction of the fiber squeezers along the time. Moreover, it has a slow response, which does not allow the transmission of information over 500 bit/s. This chapter is introduced by the study carried out, based on IQ modulators. Afterwards, a simulation of the solution chosen is performed, in order to check if the needed polarization's states can be generated.

4.2 IQ Modulation

To understand how an IQ modulator operates, it is important to learn the theory and principles behind it. In any event, the designation "IQ" comes from the abbreviation for In-phase and Quadrature [12]. For instance, these terms cannot be mentioned individually, because something can only be in phase or out of phase with reference

to another signal or an established reference point. These terms refer to two sinusoidal waves that have the same frequency and are 90° out of phase. The In-phase signal can either be assigned as the sine or the cosine waveform, as long as, the Quadrature signal is shifted by 90° relative to the In-phase one. The IQ signals are not modulated by frequency or phase, but instead they are always individually modulated by amplitude. In IQ modulation, the wave forms are modulated by signals that can have positive or negative voltage values, and consequently the amplitude modulation can result in a phase shift. One important note to take is that the I and Q signals are individually treated and, when the wave forms are added together, phase modulation can be achieved by simply varying the amplitude of these signals [11]. For example, multiplying the amplitude of the I signal by 1 and the amplitude of the Q signal by 0, would result in the waveform of the I signal. Going the other way around the result would be a 90° shift. Multiplying both amplitude signals by 1, would result in a 45° shift when both waves are added together. As mention before this modulation is not limited by null or positive values, and different results are possible to be obtained.

4.3 Quadrature amplitude modulation

Quadrature amplitude modulation (QAM) is a form of modulation that utilizes both amplitude and phase components to reach high spectral efficiencies [15]. A QAM signal represents two carriers that are shifted in phase by 90° and are modulated by amplitude and phase variations. When using QAM, the constellation points are usually arranged in a square grid with equal vertical and horizontal spacing. For that reason, the number of points in the grid is usually a power of 2. Depending on the multiplicity of the digital signals, m, the resulting QAM is going to have 2^m possible states. By using higher order modulations formats, more points the constellation is going to have and more bits per symbol can be transmitted. However, points that are closer to each other are more susceptible to noise and data errors.

Table 4.1: Table containing some QAM forms, depending on the number of bits/symbol

Bits/Symbol	QAM
m=2	QPSK
m=4	16 QAM
m=6	64 QAM
m=6	256 QAM

4.4 Phase shift keying

Phase shift keying (PSK) can be regarded as a special case of QAM, and corresponds to the digital modulation technique in which the phase of the carrier signal is changed by varying the sine and cosine inputs at a particular time. PSK can have a finite number of phases, each assigned to an unique pattern of binary sequences. Each pattern of bits forms the symbol, and it is represented by the particular phase. A convenient way to represent PSK schemes is by a constellation diagram. This shows the points in the complex plane where, the real and imaginary axis are referred as the In-phase and Quadrature terms, respectively due to their 90° difference. The constellation points chosen are usually positioned with uniform angular spacing around a circle. They are positioned in this way so that they can all be transmitted with the same energy.

4.5 Quadrature phase shift keying

Quadrature phase shift keying (QPSK) uses four points on the constellation diagram spaced equally around a circle [11]. With four phases, it can encode two bits per symbol, doubling the data rate compared with a Binary phase shift keying system (BPSK). BPSK is a two phase modulation scheme, where the binary data, 0 and 1, represents two different phase states in the carrier: 0° for a binary 0 and 180° for a binary 1. While the data is doubled, QPSK, maintains the same bandwidth and the bit error rate. A QPSK symbol does not represent a 0 or 1, but instead it represents two bits-per-symbol: 00, 01, 10 or 11. Considering the IQ signals to have the following equations:

$$I = A\sin(wt) \tag{4.1}$$

$$Q = A\cos(wt) \tag{4.2}$$

$$Binary \quad 0 => A = -1 \tag{4.3}$$

$$Binary \quad 1 => A = 1 \tag{4.4}$$

The following table can be generated:

Table 4.2: Summarize phase shifts that can be obtained by applying a 2 bit sequence to the In-phase and Quadrature wave forms, considering [-180°,180°]

I Data	Q Data	I waveform	Q wave form	I+Q	Phase
0	0	-sin(ωt)	-cos(ωt)	$\sqrt{2}\sin(\omega t - 135°)$	-135°
0	1	-sin(ωt)	cos(ωt)	$\sqrt{2}\sin(\omega t + 135°)$	+135°
1	0	sin(ωt)	-cos(ωt)	$\sqrt{2}\sin(\omega t - 45°)$	-45°
1	1	sin(ωt)	cos(ωt)	$\sqrt{2}\sin(\omega t + 45°)$	+45°

As well as the constellation point diagram:

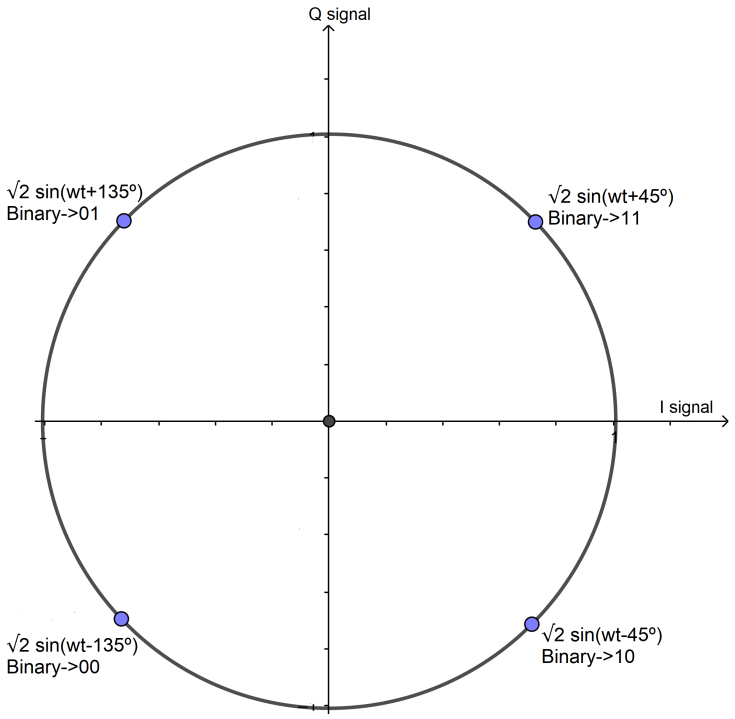

Figure 4.1: QPSK constellation point diagram

Even though we are only varying the the amplitudes of the I and Q signals, it is

possible to get four different phase outputs, when adding the two signals together. With this method we do not need to directly vary the phase of the carrier.

4.6 IQ modulator

An IQ modulator is a structure composed of phase modulators and two Mach-Zehnder interferometers [16]. The incoming radiation is equally split into two arms, the In-phase (I) and the Quadrature (Q) arm. In both paths, a field amplitude modulation is performed by operating the Mach-Zehnder interferometers in the push-pull mode at the minimum transmission point. In the second Mach-Zehnder interferometers, the signal goes through a 90° phase shift to form the Quadrature (Q) signal. This way, any constellation point can be reached in the complex IQ plane after recombining the radiation of both branches. The most popular form of modulation used is the above-mentioned QPSK in which an IQ modulator is used to create 4 symbols (each uniquely located in 2D phase-intensity space).

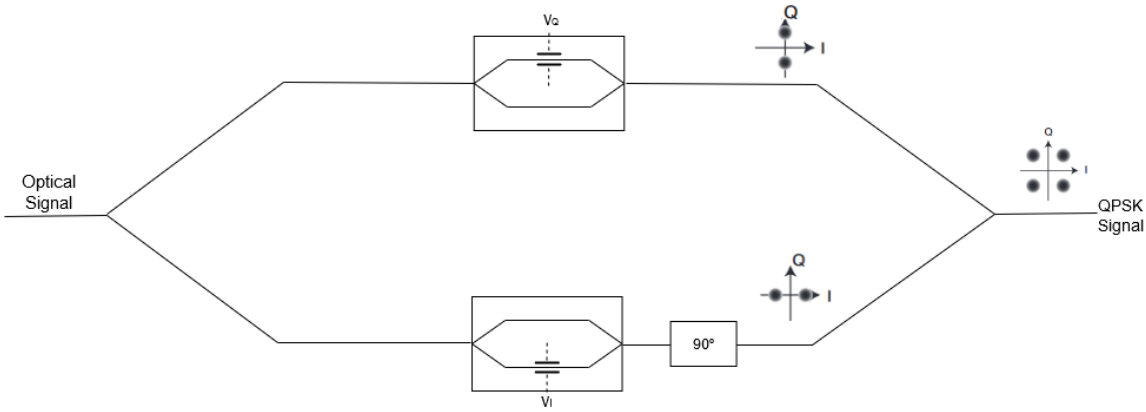

Figure 4.2: Structure of an IQ modulator, composed by two Mach-Zehnder interferometers and a phase modulator.

4.6.1 Phase modulator

An electro-optic modulator is a device which can be used for controlling the power, phase or polarization of light with an electrical control signal[17]. It typically contains two Pockels cells, and additional optical elements such as polarizers. A Pockels cell is a device consisting of an electro-optical crystal through which light can be transmitted. The phase delay in the crystal can be modulated by applying an elec-

tric voltage. The Pockels cell thus acts as a voltage-controlled waveplate, and the principle of operation is based on the Pockels effect. Pockels effect is the name given to the occurrence of birefringence, and to the change in existing birefringence phenomena in an electric field, linearly proportional to the electric field strength. These devices can be used in fiber implementations, where the Pockels cell is placed between two fiber collimators. Since the interest is not to modulate directly the phase of the laser beam, Pockels cells can also be used for other types of modulation. For instance, the electro-optic phase modulator can be used in one arm of a Mach-Zehnder interferometer in order to obtain amplitude modulation.

4.6.2 Mach-Zehnder

Mach-Zehnder interferometers are used for controlling the amplitude of an optical wave. The input waveguide is split up into two waveguide interferometer arms. In each arm there is a phase modulator, where a voltage is applied to induce a phase shift in the wave passing through that arm. It can apply phase shifts in both arms, or only in one of them, where the other will be the reference arm. Then, the two arms are recombined, and the phase difference between the two waves is converted to an amplitude modulation. The output is the result of the interference between the two signals that pass the arms, ranging from constructive (the phase difference between the two signal is zero) to destructive (the phase difference between the two signal is 180°). The Mach-Zehnders are operated in the push-pull operation, where the polarities of optical phase shifts are opposite to each other so that the optical output can be amplitude modulated without any parasitic phase modulation.

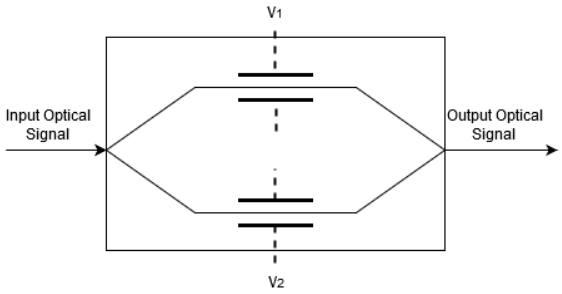

Figure 4.3: Mach-Zehnder interferometer structure.

4.6.3 Single-mode optical fiber

Single-mode fiber, also known as mono-mode fibers ideally only supports one mode to propagate in it. Single-mode fibers are capable of wide bandwidths and are suited for telecommunications over 1 kilometer. They are generally used for wavelengths of 1300 nm to 1550 nm, where the attenuation is low and sources and detectors are available. In conventional single mode fiber, the fundamental mode consists of two orthogonal polarization modes. These modes may be chosen arbitrarily between horizontal and vertical polarization's, in the x and y direction respectively. The electric field of light propagating along the fiber is a linear superposition of these two polarization modes and depends on the polarization of the light at the launching point. And so a single mode optical fiber can support two quasi-degenerate orthogonal modes.

4.6.4 Dual polarization-IQ modulator

The capacity of coherent transmission can be further doubled by using two orthogonal states of polarization, each using a separate IQ modulator. Dual polarization states are created by using two IQ modulators in combination with a polarization rotator and a polarization beam combiner [16]. This technique allows the two orthogonal polarization's of a laser beam to be combined with QPSK modulation on each polarization component. The laser beam is split equally to the two Mach-Zehnder modulators. Both Mach-Zehnders structures are operated by two DAC's, one for the In-phase component and the other for the Quadrature component (shifted 90° by a phase modulator). The DAC's act accordingly to one bit of data, meaning that the system needs a 4 bit sequence to obtain the desirable effect. After the Mach-Zehnder modulations, in one of the arms there is a 90° polarization rotator, so that each polarization component can be achieved. The two signals are then recombined forming the desired polarization state. This modulator can operate at transmission rates over the GHz.

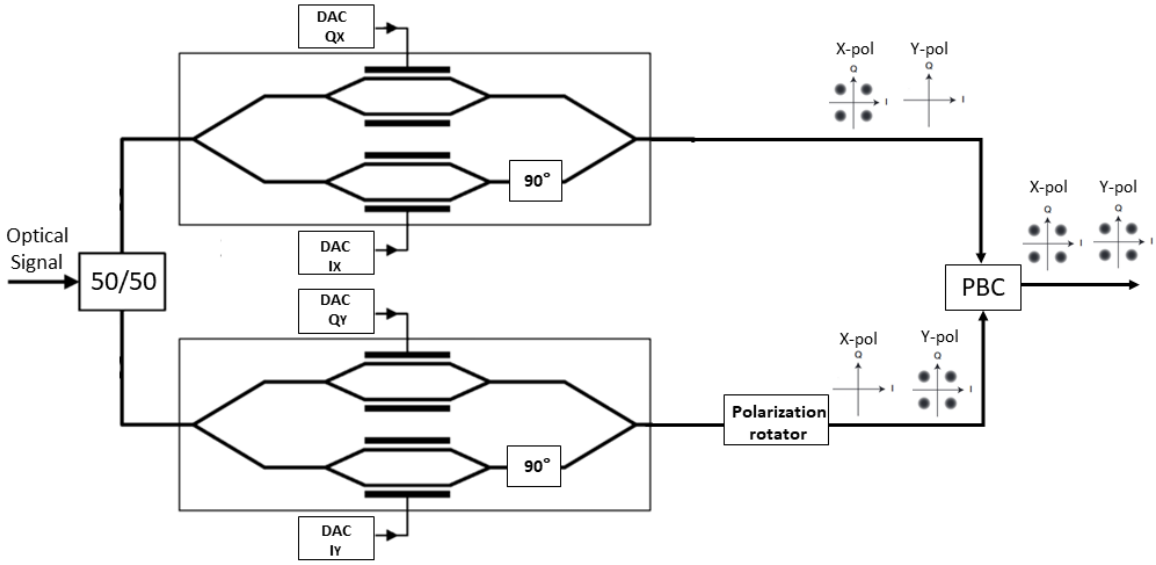

Figure 4.4: Dual-Polarization IQ modulator structure. It is composed by a 50/50 beam splitter, two IQ modulators (one for each polarization) and one polarization rotator. Can take a maximum of 4 bits per symbol.

Considering that the optical input is in the horizontal state the following table can be made:

Table 4.3: Summarized table containing a sequence of bits that can generate a polarization state. Each sequence can induce a different phase shift.

Ix bit	Qx bit	Iy bit	Qy bit	δx	δy	$\delta y - \delta x$	P. rotator	Polarization
0	0	-	-	-135°	-	0°	90°	Horizontal
-	-	0	0	-	-135°	0°	90°	Vertical
0	1	0	1	135°	135°	0°	90°	Linear +45°
0	0	1	1	-135°	45°	180°	90°	Linear -45°
1	0	1	1	-45°	45°	90°	90°	Right Circular
0	1	1	1	135°	45°	-90°	90°	Left Circular

4.7 Simulation

One important task, is to validate the study by performing a simulation capable of corresponding to the necessary requirements. For this work, the challenge was to create a Matlab simulation by feeding a 4-bit sequence to the modulator and try to generate a polarization state represented by the phase plot and the Poincaré sphere.

The first step is to define two binary sequences, one for the X-polarization component and the other to the Y-polarization. Each sequence is composed of two bits, that needs to be transformed into the corresponding positive or negative values, for the amplitude modulation. The sequence reshape takes two bits as input, and transforms the bit 0 into a negative integer of 1 and the bit 1 to a positive integer of 1. Now the I and Q data are ready to be processed into the wave forms. For this simulation, the In-phase component is represented by the sine function and the Quadrature component by the cosine function. Depending on the frequency chosen, a time vector can be implemented into the wave forms, creating a signal with amplitude and time parameters. The I and Q data (1 or -1) is then multiplied by the sine and cosine functions. After the amplitude modulation, for each polarization component, the In-phase and Quadrature signals are added together, forming the X and Y signals. Depending on the input bit sequence, different phases can be obtained, but they only make sense when comparing the X and Y signals. The respective plots for the In-phase, Quadrature, X signal and Y signal are then displayed.

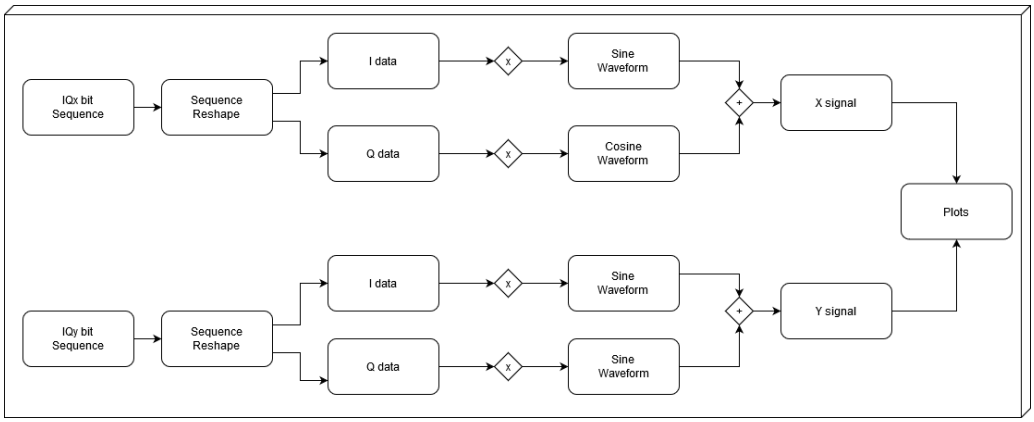

Figure 4.5: Diagram containing the bit and waveforms reshape process. Bit 1 corresponds to the integer 1 and bit 0 to -1.

To make the phase plot, the gradient of the X and Y signals needs to be calculated. This function returns the dimensional numerical derivative of the X and Y signals (both are 1 dimensional vectors). The quiver Matlab function, takes the first two arguments as the (X, Y), origin of each arrow, and uses the derivative of the X and Y signals as the relative direction of each arrow. With this, the phase plot corresponding to a polarization state can be obtained.

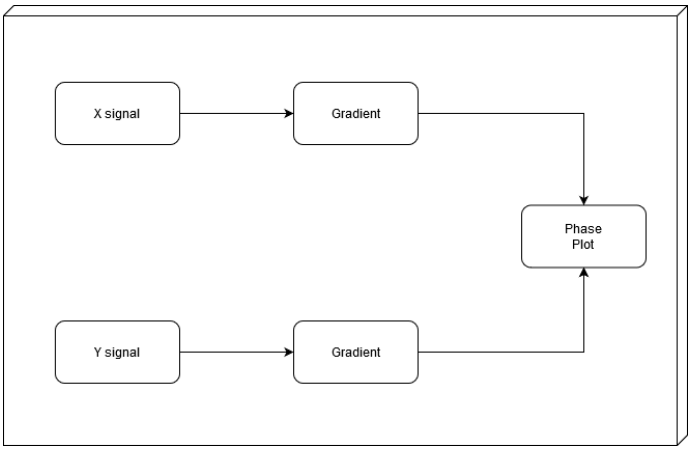

Figure 4.6: Program diagram to obtain the phase plot.

In order to get the phase difference angle between the two signals, the DC component was removed from them. The fast Fourier transform of each signal was calculated, and by getting the absolute max value for X and Y its possible to use the angle function. This function returns the phase angle in the [-180°,180°] interval. The phase difference can be calculated by subtracting the angle of Y to the X one.

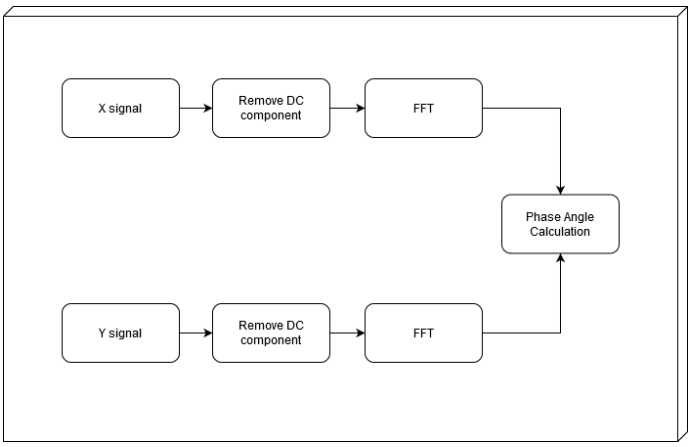

Figure 4.7: Program diagram to calculate the exact phase difference angle.

Lastly, from the X signal, Y signal and the phase difference, the values of S0, S1, S2, S3 can be obtained using the stokes equations. These values form a vector that corresponds to a point in the Poincaré sphere.

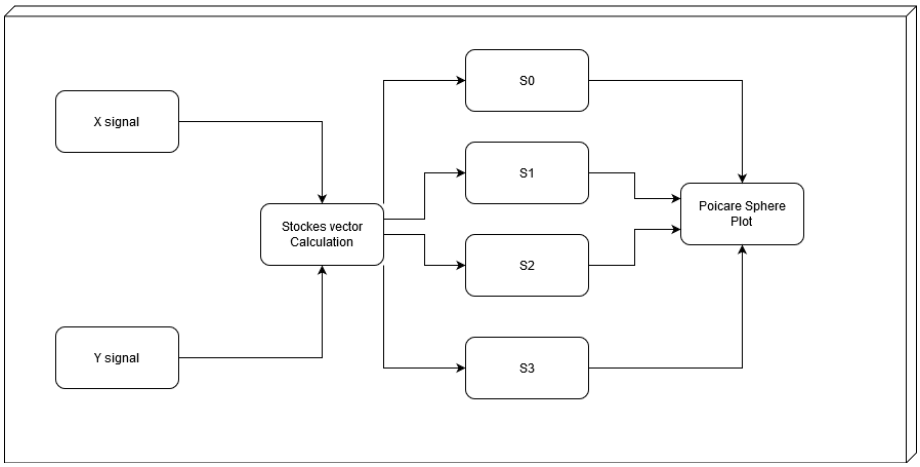

Figure 4.8: Program diagram to calculate the stokes parameters and plot the respective Poincaré Sphere.

4.8 Simulation results

The results obtained for this simulation can be divided for each polarization state from table 4.3. Both the individual behavior of the In-phase/Quadrature signals and the result from adding them together is represented by the respective amplitude/-time graphics, for each polarization component (X and Y). The polarization states are reproduced by the phase difference plots and by the specific stokes vector point in the Poincaré sphere. It is important to note that the vertical and horizontal polarization's, only need a 2 bit sequence. Meaning, for the horizontal state, only the Mach-Zehender without the polarization rotator, at its exit, is going to work. To generate the vertical polarization state, only the Mach-Zehender with the polarization rotator, at its exit, is going to operate. To generate all other states, the system works normally to reach both polarization components.

4.8.1 Horizontal polarization

Sequence: [00–]

To generate this state, only the X component matters, as the Y component is set to zero. For the X component, both the In-phase and the Quadrature signals are multiplied by -1, leading to a -135° shift.

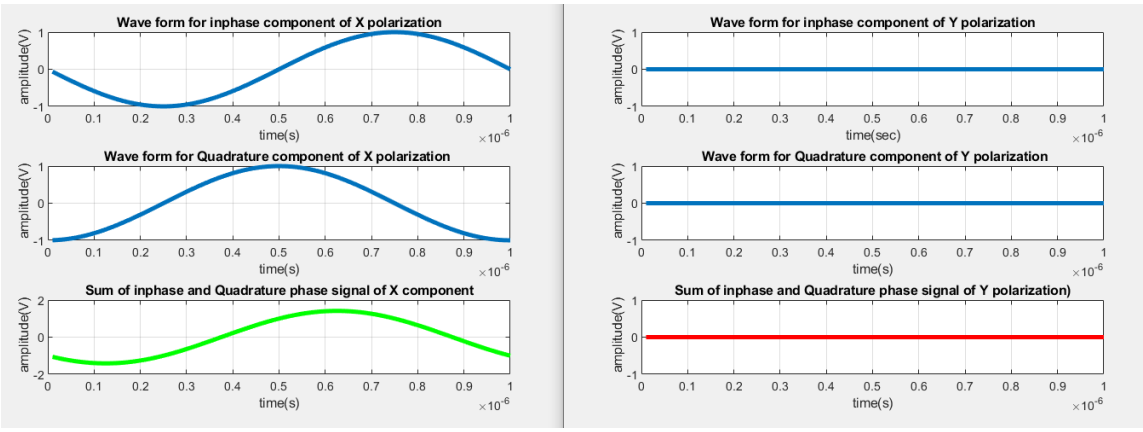

Figure 4.9: In-phase and Quadrature signals behavior for the horizontal polarization sequence. The sum of the signals is represented in the last graphic.

Even though the X signal shifts -135°, it doesn't matter because there isn't another signal to analyze the phase difference. The component electric field oscillates with direction to the horizontal x-axis.

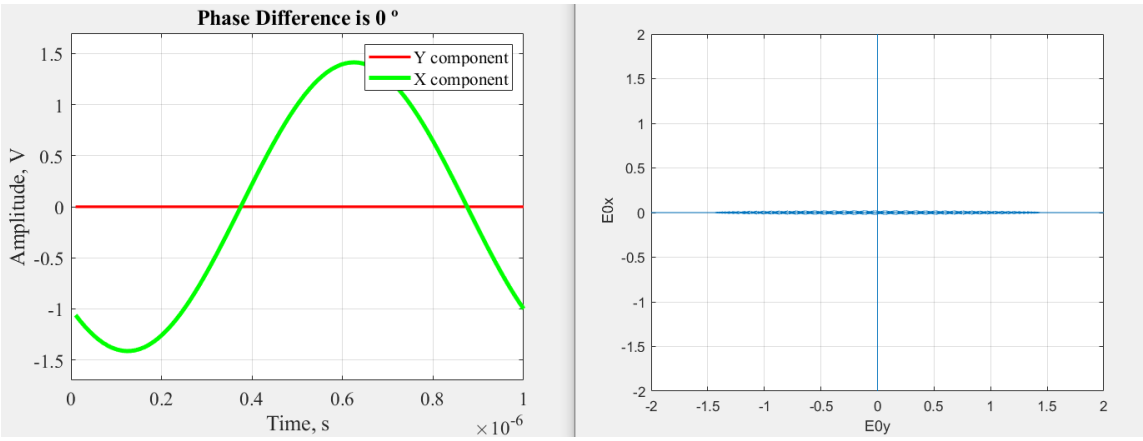

Figure 4.10: Phase difference and phase graphic for the horizontal state.

By analyzing the amplitudes of the X and Y signal, it is possible to normalize the values and obtain the stokes parameters. For this case, only the X signal matters,

and the phase difference is considered to be zero. The horizontal polarization is located at the far positive side of the S1 axis, in the Poincaré Sphere.

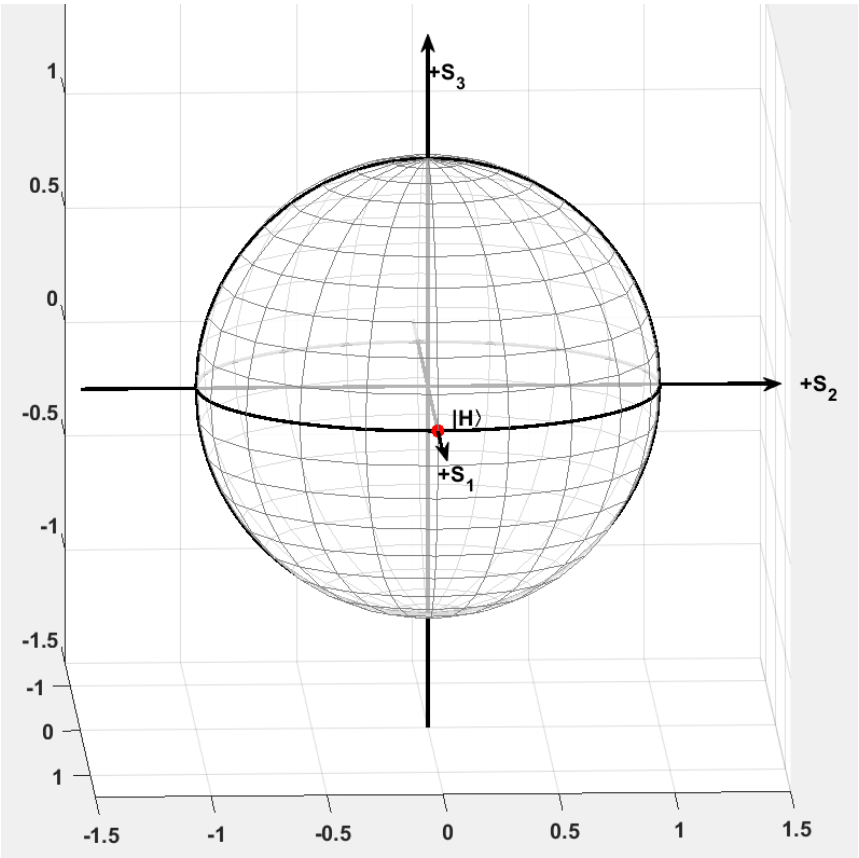

Figure 4.11: Poincaré sphere containing the stokes vector for the horizontal polarization state.

4.8.2 Vertical polarization

Sequence: [–00]

To generate this state, only the Y component matters, as the X component is set to zero. For the Y component, both the In-phase and the Quadrature signals are multiplied by -1, leading to a -135° shift.

Since the plot functions take as inputs (x,y) coordinates, there is no need to simulate the polarization rotator, because the 90° shift is automatically set.

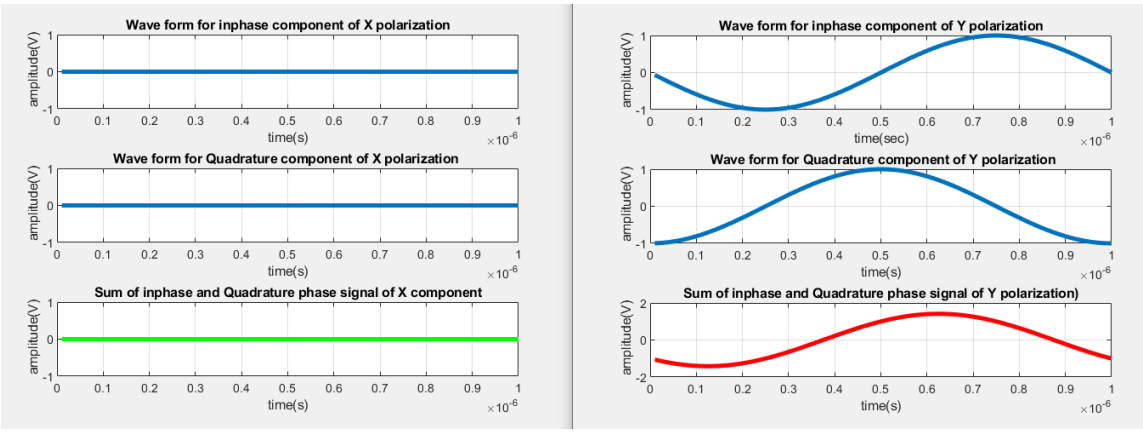

Figure 4.12: In-phase and Quadrature signals behaviour for the vertical polarization sequence. The sum of the signals is represented in the last graphic.

Even though the X signal shifts -135°, it doesn't matter because there isn't another signal to analyze the phase difference. The component electric field oscillates with direction to the vertical y-axis.

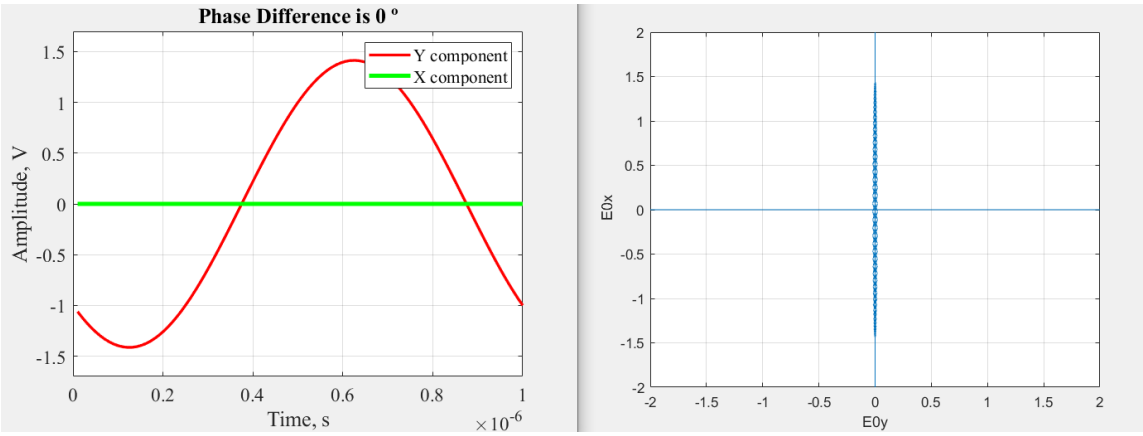

Figure 4.13: Phase difference and phase graphic for the vertical state.

By analyzing the amplitudes of the X and Y signal, it is possible to normalize the values and obtain the stokes parameters. For this case, only the Y signal matters, and the phase difference is considered to be zero. It can be seen that in the Poincaré sphere the vertical polarization state is in the far negative side of the S1 axis. In this axis both the horizontal and the vertical form an orthogonal basis, as expected.

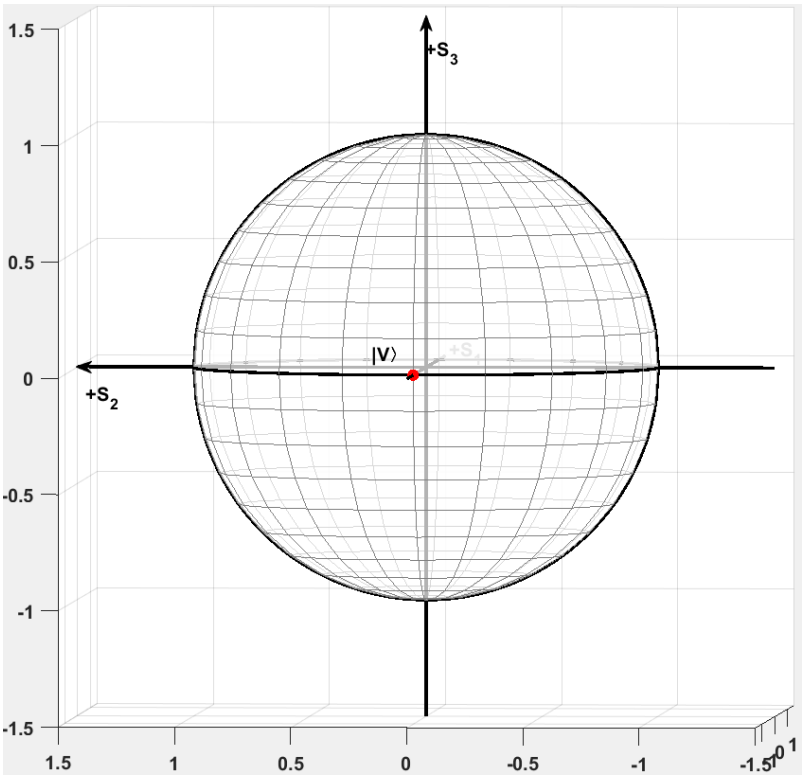

Figure 4.14: Poincaré sphere containing the stokes vector for the vertical polarization state.

4.8.3 Linear + 45°

Sequence: [0101]

To generate this state, both X and Y components are modulated by the same sequence. Meaning a phase shift of 135° is going to be introduced in both signals.

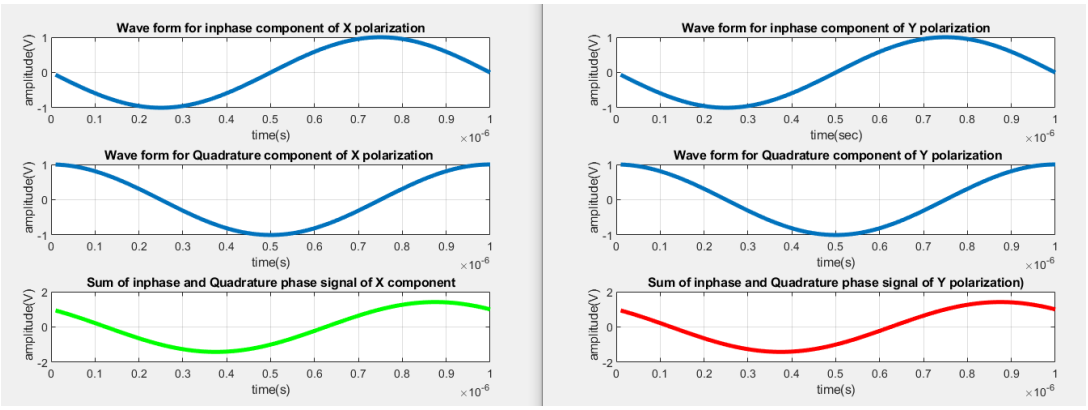

Figure 4.15: In-phase and Quadrature signals behaviour for the linear +45° polarization sequence. The sum of the signals is represented in the last graphic.

Because the X and Y signals are shifted by 135°, no phase difference is introduced when compared one to another. The component electric field oscillates with a 45° between the x and y-axis.

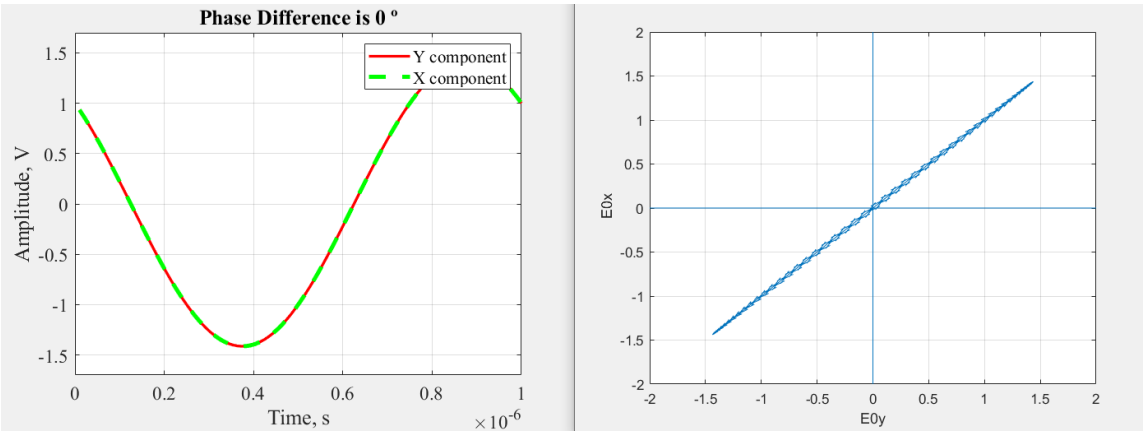

Figure 4.16: Phase difference and phase graphic for the linear +45° state.

By analyzing the amplitudes of the X and Y signal, it is possible to normalize the values and obtain the stokes parameters. For this case, the X and Y amplitude matters, and the phase difference is considered to be zero. The diagonal polarization is located at the far positive side of the S2 axis, in the Poincaré Sphere.

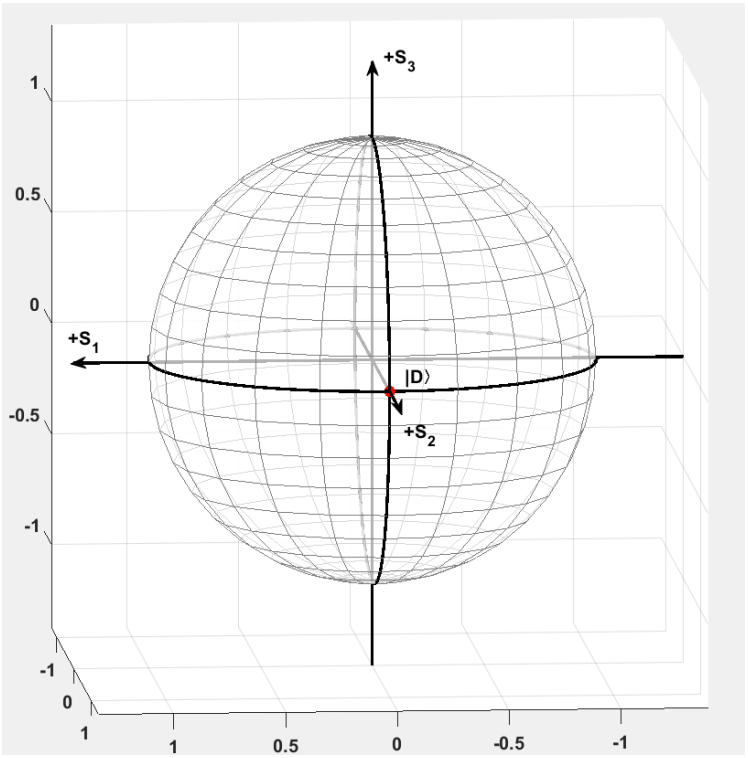

Figure 4.17: Poincaré sphere containing the stokes vector for the linear +45°.

4.8.4 Linear - 45°

Sequence: [0011]

To generate this state, the X and Y components are modulated by a different 2 bit sequence. Meaning a phase shift of -135° is going to be introduced in the X signal, and a phase shift of 45° in the Y signal when the In-phase and Quadrature components are added together.

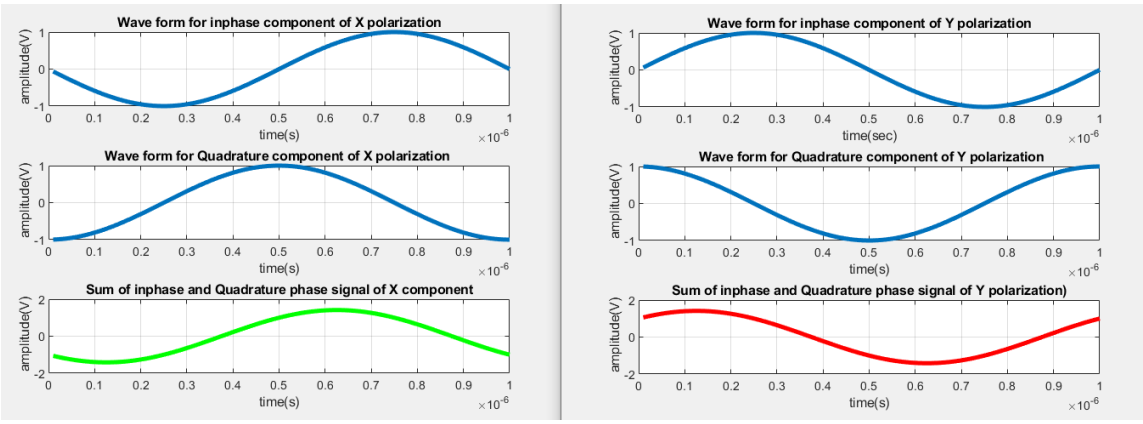

Figure 4.18: In-phase and Quadrature signals behaviour for the linear -45° polarization sequence. The sum of the signals is represented in the last graphic.

The phase difference introduced is going to be 180°. The electric field oscillates with a -45° between the x and y-axis.

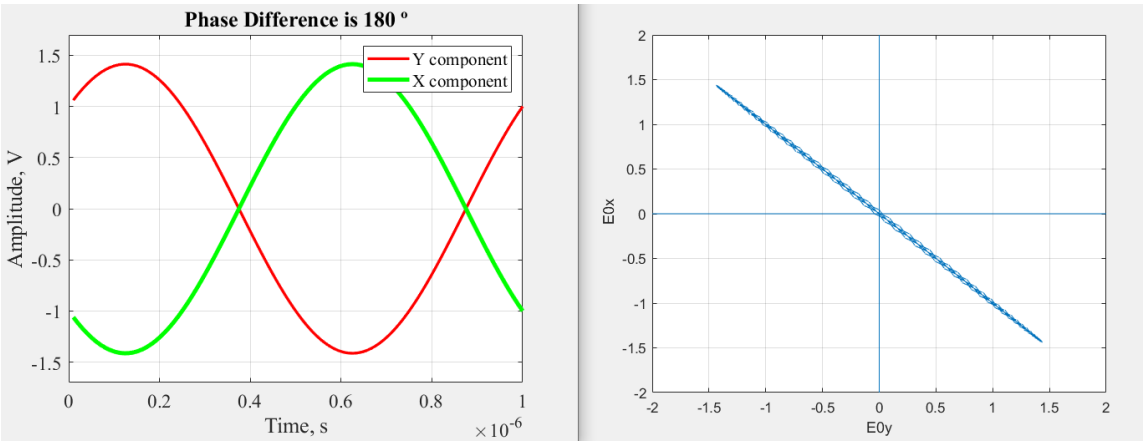

Figure 4.19: Phase difference and phase graphic for the linear -45° state.

By analysing the amplitudes of the X and Y signal, it is possible to normalize the

values and obtain the stokes parameters. For this case, the X and Y amplitude matters, and the phase difference is considered to be 180°.

It can be seen that in the Poincaré sphere the linear -45° polarization state is in the opposite side of the S2 axis. In this axis both the linear +45° and the linear -45° form an orthogonal basis, as expected.

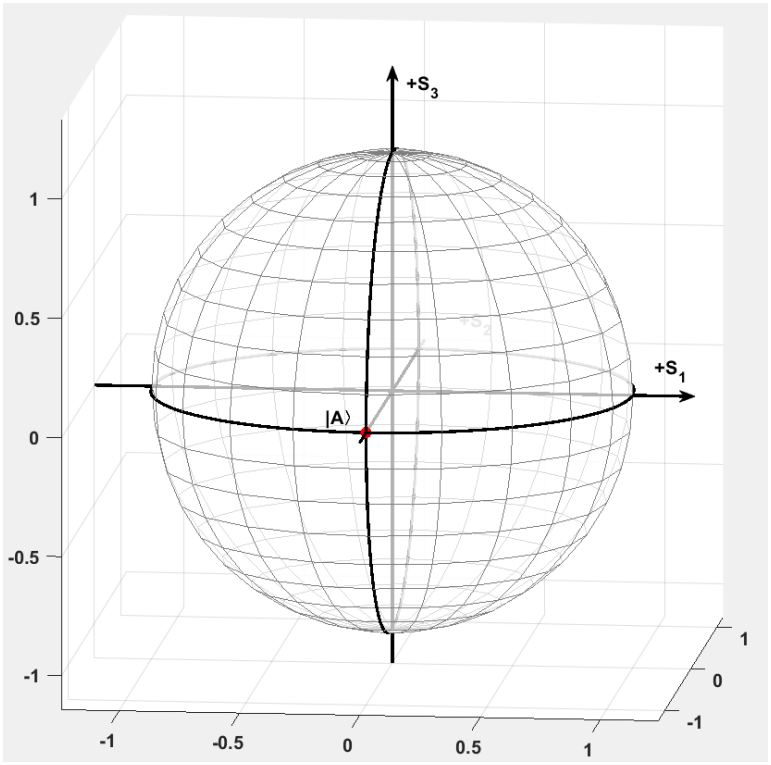

Figure 4.20: Poincaré sphere containing the stokes vector for the linear -45°.

4.8.5 Right circular polarization

Sequence: [1011]

To generate this state, the X and Y components are modulated by a different 2 bit sequence. Meaning a phase shift of -45° is going to be introduced in the X signal, and a phase shift of 45° in the Y signal when the In-phase and Quadrature components are added together.

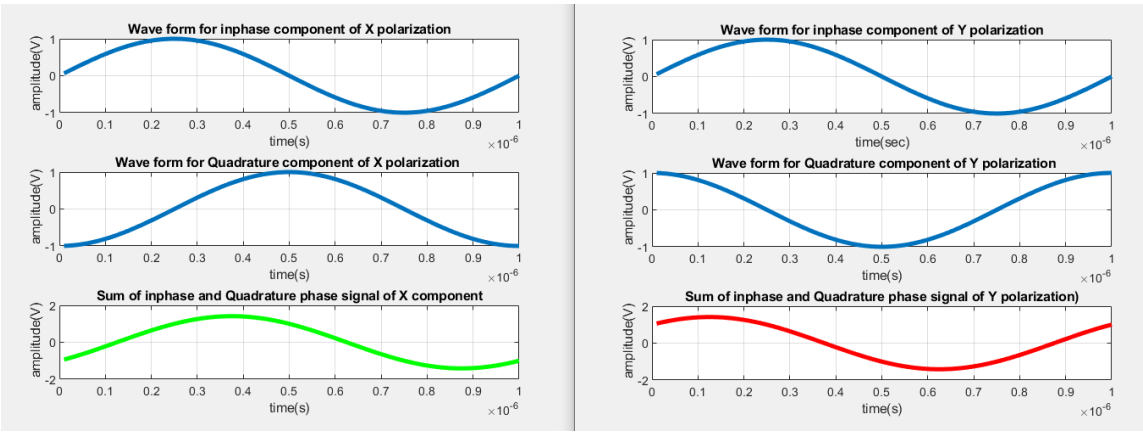

Figure 4.21: In-phase and Quadrature signals behaviour for the right circular polarization sequence. The sum of the signals is represented in the last graphic.

The phase difference introduced is going to be 90°. The resulting electric field rotates in the right-hand sense with respect to the direction of propagation.

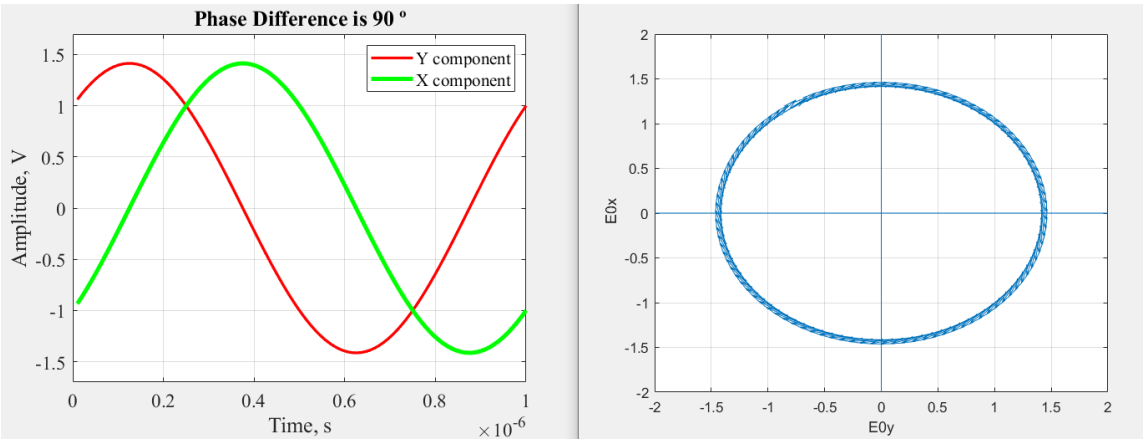

Figure 4.22: Phase difference and phase graphic for the right circular state.

By analyzing the amplitudes of the X and Y signal, it is possible to normalize the values and obtain the stokes parameters. For this case, the X and Y amplitude matters, and the phase difference is considered to be 90 °. The right circular polarization is located at the far positive side of the S3 axis, in the Poincaré Sphere.

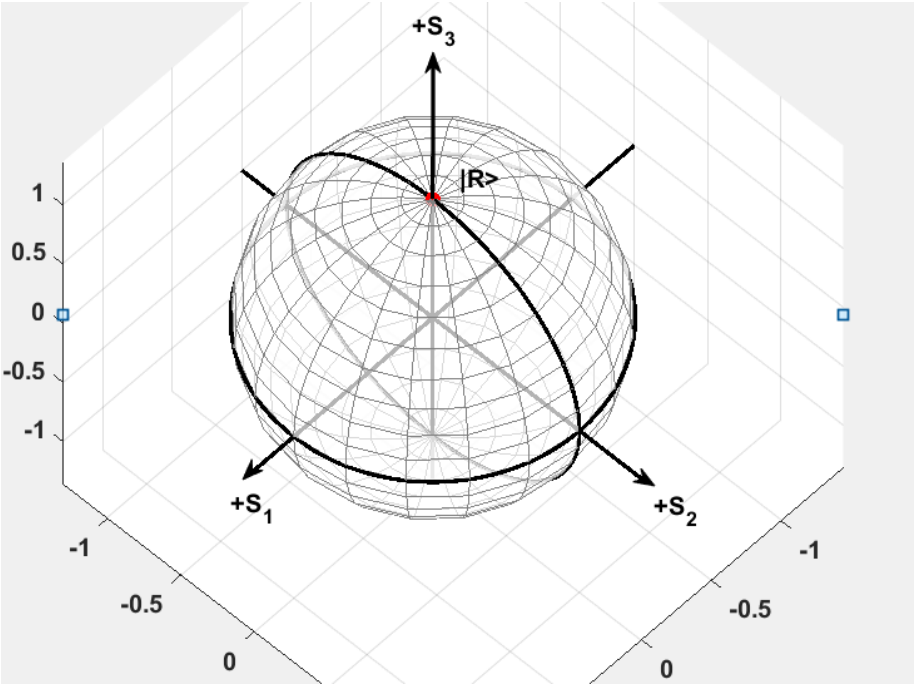

Figure 4.23: Poincaré sphere containing the stokes vector for the right circular polarization state.

4.8.6 Left circular polarization

Sequence: [0111]

To generate this state, the X and Y components are modulated by a different 2 bit sequence. Meaning a phase shift of 135° is going to be introduced in the X signal, and a phase shift of 45° in the Y signal when the In-phase and Quadrature components are added together.

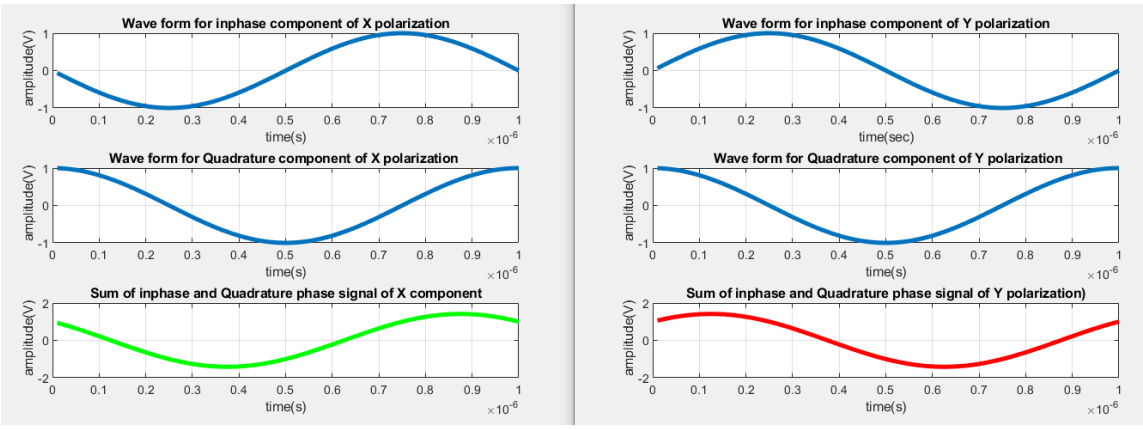

Figure 4.24: In-phase and Quadrature signals behaviour for the left circular polarization sequence. The sum of the signals is represented the the last graphic.

The phase difference introduced is going to be -90°. The resulting electric field rotates in the left-hand sense with respect to the direction of propagation.

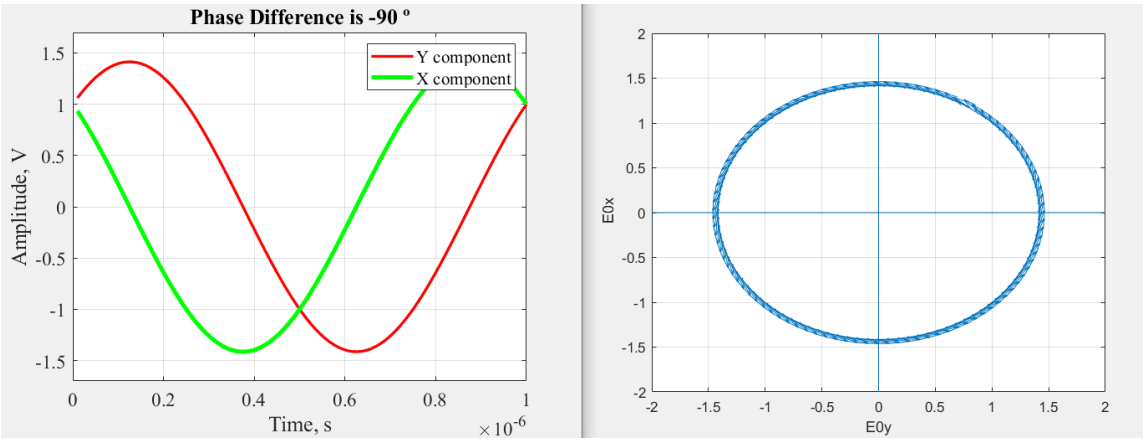

Figure 4.25: Phase difference and phase graphic for the left circular state.

By analysing the amplitudes of the X and Y signal, it is possible to normalize the values and obtain the stokes parameters. For this case, the X and Y amplitude matters, and the phase difference is considered to be -90°. The left circular polarization is located at the far negative side of the S3 axis, in the Poincaré Sphere. In this axis, both the right and left circular states form an orthogonal basis, as expected.

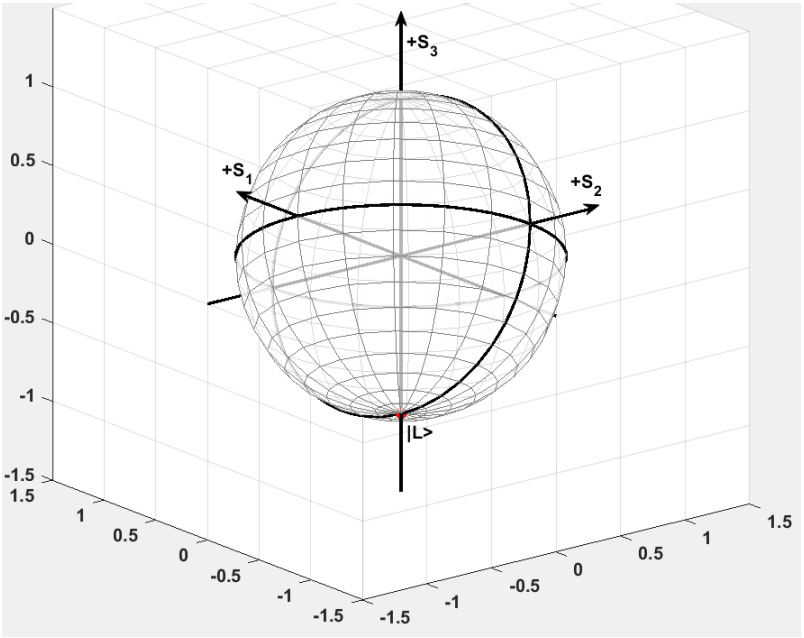

Figure 4.26: Poincaré sphere containing the stokes vector for the left circular polarization state.

Chapter 5

Conclusions

The work accomplished in this dissertation was organized in three steps. The first one, was to study the fundamentals of DV-QKD protocols and how they can be implemented. The other two are related to the optimization of the receptor module and the new design of the transmitter, in order to achieve higher key transmission rates.

For the receptor, the Arduino solution is capable of reading the detectors electrical signal, process this information and then send it to an upper level protocol. Even though the algorithm to set the correct voltages is yet to be completed, the upper level can send four random voltages to the second Arduino. This Arduino can successfully set the random voltages into the external DAC, and not lose the synchronization up to 2.5 kHz speeds. This doesn't represent a problem, because the transmitter is currently operating at the 500 Hz margin. The Arduino solution can also operate for periods of time around the 2-hour mark, without loosing synchronization as seen in the QBER reports.

The new transmitter design, using IQ modulators, is expected to operate at GHz rates. However, speed was not the only requirement, because the Dual-IQ modulator needs to be able to generate the desirable polarization states. From the study, it can be concluded that a 4 bit sequence can introduce a phase shift to the X and Y polarization components of an optical input signal. Depending on the sequence, the vertical, horizontal, linear $+45°$, linear $-45°$, right circular and left circular states (forming the three non-orthogonal bases) can theoretically be generated. This statement is further validated by simulating the steps of this type of modulation.

5.1 Future work

This work can be used as future reference to continuing the optimization of the system implemented in IT-Aveiro laboratory. Using the Arduino layout and the study performed on the Dual-IQ modulators, the objective of having the system operating at high speeds and with high stability can be achieved.

For the Arduino implementation, the correct voltage values to apply on the EPC, based on the different bases, are still unknown. The algorithm in the upper level protocol, which chooses the voltage values to compensate the random polarization drifts and commute the bases, still needs to be finished. More tests, regarding the stability of the Arduino set-up, should be done for longer periods of time, in order to test its limits. Eventually, the transmitter will be operating at higher speeds, meaning a solution to solve the 2.5 kHz limitation is going to be needed.

For the transmitter side, it was shown that the new design can generate the desirable polarization states. Nevertheless, its still needed to study how the Dual-IQ structure can commute between the different states. It is important to test in the laboratory if this sequences lead up to the simulation results. To do this, the binary sequences need to be transformed to voltage values, which for now are still unknown. Simple tests can be achieved with the help of polarimeter structure and software analysis, to see the points in the Poincaré sphere, given a 4 bit sequence. If all this can be accomplished, the last step is to implement the new design in the laboratory system.

References

[1] F. optics 4 sale, *The derivation of stokes polarization parameters*, Last accessed July 2020, . [Online]. Available: https://www.fiberoptics4sale.com/blogs/wave-optics/102492742-stokes-polarization-parameters (cit. on pp. 12, 13).

[2] C. Agnesi, M. Avesani, A. Stanco, P. Villoresi and G. Vallone, 'All-fiber self-compensating polarization encoder for quantum key distribution', Mar. 2019 (cit. on p. 5).

[3] *Arduino due specs*, Last accessed July 2020, . [Online]. Available: https://store.arduino.cc/arduino-due (cit. on p. 26).

[4] A. Boaron, B. A. Korzh, R. Houlmann, G. Boso, D. Rusca, S. C. Gray, M.-j. Li, D. Nolan, A. Martin and H. Zbinden, 'Simple 2.5ghz time-bin quantum key distribution', 2018 (cit. on p. 4).

[5] *Dac specs*, Last accessed July 2020, . [Online]. Available: https://store.ncd.io/product/ad5669-16-bit-8-channel-digital-to-analog-converter-i2c-mini-module/ (cit. on p. 28).

[6] A. Duplinskiy, V. Ustimchik, A. Kanapin, V. Kurochkin and Y. Kurochkin, 'Low loss qkd optical scheme for fast polarization encoding', *Opt. Express*, vol. 25, no. 23, pp. 28 886–28 897, Nov. ts. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-25-23-28886 (cit. on p. 5).

[7] M. Fox, *Quantum optics: an introduction*, ser. Oxford Master Series in Atomic, Optical and Laser Physics. Oxford: Oxford Univ. Press, 2006 (cit. on pp. 16, 17, 19, 20).

[8] N. Gisin, G. Ribordy, W. Tittel and H. Zbinden, 'Quantum cryptography', *Rev. Mod. Phys.*, vol. 74, pp. 145–195, 1 Mar. 2002. DOI: 10.1103/RevModPhys.74.145. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.74.145 (cit. on pp. 1–4).

[9] D. Goldstein, *Polarized Light*. CRC Press, 2017, ISBN: 9781439830413. [Online]. Available: https://books.google.pt/books?id=w6PMBQAAQBAJ (cit. on pp. 12, 15, 20).

[10] F. Grünenfelder, A. Boaron, D. Rusca, A. Martin and H. Zbinden, 'Simple and high-speed polarization-based qkd', *Applied Physics Letters*, vol. 112, 2018. DOI: 10.1063/1.5016931 (cit. on p. 5).

[11] N. Instruments, *Iq modulation*, Last accessed July 2020, . [Online]. Available: http://www.ni.com/tutorial/4805/en/ (cit. on pp. 39, 40).

[12] *Iq modulation*, Last accessed July 2020, . [Online]. Available: https://www.allaboutcircuits.com/textbook/radio-frequency-analysis-design/radio-frequency-demodulation/understanding-i-q-signals-and-quadrature-modulation/ (cit. on p. 38).

[13] X. Liu, C. Liao, J. Mi, J. Wang and S. Liu, 'Intrinsically stable phase-modulated polarization encoding system for quantum key distribution', *Physics Letters A - PHYS LETT A*, vol. 373, pp. 54–57, Dec. 2008 (cit. on p. 5).

[14] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667 (cit. on pp. 3, 7).

[15] E. notes, *Qam*, Last accessed July 2020, . [Online]. Available: https://www.electronics-notes.com/articles/radio/modulation/quadrature-amplitude-modulation-what-is-qam-basics.php (cit. on p. 39).

[16] ntua, *Coherent optical dsp*, Last accessed July 2020, . [Online]. Available: https://www.photonics.ntua.gr/OptikaDiktyaEpikoinwnias/Lecture_4_CoherentOptical_DSP.pdf (cit. on pp. 42, 44).

[17] RP-photonics, *Phase modulator*, Last accessed July 2020, . [Online]. Available: https://www.rp-photonics.com/pockels_cells.html (cit. on p. 42).

[18] S. Pirandola, U. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. Pereira, M. Razavi, J. Shaari, M. Tomamichel, V. Usenko, G. Vallone, P. Villoresi and P. Wallden, 'Advances in quantum cryptography', English, *arXiv*, 2019 (cit. on pp. 2–4, 18).

[19] I. Quantique, 'Single-photon detector module', 2011 (cit. on p. 26).

[20] B. Rijsman, *A cascade information reconciliation tutorial*, Last accessed July 2020, . [Online]. Available: https://hikingandcoding.wordpress.com/2020/01/15/a-cascade-information-reconciliation-tutorial/ (cit. on p. 18).

[21] J. Wang, X. Qin, Y. Jiang, X. Wang, L. Chen, F. Zhao, Z. Wei and Z. Zhang, 'Experimental demonstration of polarization encoding quantum key distribution system based on intrinsically stable polarization-modulated units', *Opt. Express*, vol. 24, no. 8, pp. 8302–8309, Apr. 2016. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-24-8-8302 (cit. on p. 5).

Appendix A
Arduino Code

```
#include <Wire.h>

const byte interruptPin_Trigger = 26;
const byte interruptPin_Detect1 = 27;
const byte interruptPin_Detect2 = 28;

long int countT = 0;
int flag1 = 0;
int flag2 = 0;

volatile bool stateT = false;
volatile bool state2 = false;
volatile bool state3 = false;
#define Addr 0x56


void setup() {
  Wire.begin();
  Wire.setClock(4000000);
  Serial.begin(115200);
  pinMode(interruptPin_Trigger, INPUT);
  pinMode(interruptPin_Detect1, INPUT);
  pinMode(interruptPin_Detect2, INPUT);

  attachInterrupt(digitalPinToInterrupt(interruptPin_Trigger),
   triggerFunction , RISING);
  attachInterrupt(digitalPinToInterrupt(interruptPin_Detect1),
   triggerFunction2, RISING);
  attachInterrupt(digitalPinToInterrupt(interruptPin_Detect2),
   triggerFunction3, RISING);
}

int incomingByte1 = 0;
int incomingByte2 = 0;
int incomingByte3 = 0;

bool state = true;
unsigned int dataA[2] = {0x00, 0x00};
unsigned int dataB[2] = {0x00, 0x00};
unsigned int dataC[2] = {0x00, 0x00};
unsigned int dataD[2] = {0x00, 0x00};

void loop() {
  if (stateT) {


    if (state2 && state3) Serial.print("2");
```

```
45      else if (state2 && !state3) Serial.print("0");
46      else if (!state2 && state3) Serial.print("1");
47      else if (!state2 && !state3) Serial.print("3");
48      stateT = false;
49      state2 = false;
50      state3 = false;
51
52      incomingByte1 = Serial.read();
53      incomingByte2 = Serial.read();
54      incomingByte3 = Serial.read();
55
56      if (state) {
57        dataA[0] = 0x00;
58        dataA[1] = 0x00;
59
60        dataB[0] = 0x00;
61        dataB[1] = 0x00;
62
63        dataC[0] = 0x00;
64        dataC[1] = 0x00;
65
66        dataD[0] = 0x00;
67        dataD[1] = 0x00;
68        state = false;
69      } else {
70        dataA[0] = 0xff;
71        dataA[1] = 0x00;
72
73        dataB[0] = 0x7f;
74        dataB[1] = 0x00;
75
76        dataC[0] = 0x3f;
77        dataC[1] = 0x00;
78
79        dataD[0] = 0x1f;
80        dataD[1] = 0x00;
81        state = true;
82      }
83
84      // Start I2C transmission
85
86      Wire.beginTransmission(Addr);
87      Wire.write(0b00100000);
88      Wire.write(dataA[0]);
89      Wire.write(dataA[1]);
90      Wire.endTransmission();
91
92      Wire.beginTransmission(Addr);
93      Wire.write(0b00100001);
94      Wire.write(dataB[0]);
95      Wire.write(dataB[1]);
96      Wire.endTransmission();
97
98      Wire.beginTransmission(Addr);
99      Wire.write(0b00100010);
100     Wire.write(dataC[0]);
101     Wire.write(dataC[1]);
```

```
    Wire.endTransmission();

    Wire.beginTransmission(Addr);
    Wire.write(0b00100011);
    Wire.write(dataD[0]);
    Wire.write(dataD[1]);
    Wire.endTransmission();

  }

}
void triggerFunction()
{
  stateT = true;
}
void triggerFunction2()
{
  state2 = true;
}
void triggerFunction3()
{
  state3 = true;
}
```

Listing A.1: Arduino Code

Appendix B
Simulation code

```
clear all;
close all;

x=[1,0];
datax=[x];
y=[1,1];
data=[y];
brx=10.^6;
fx=brx;
Tx=1/brx;
tx=Tx/99:Tx/99:Tx;

figure(1)

stem(data, 'linewidth',3), grid on;
title('  Information before Transmiting ');
axis([ 0 11 0 1.5]);

figure(2)

stem(data, 'linewidth',3), grid on;
title('  Information before Transmiting ');
axis([ 0 11 0 1.5]);

data_NZRx=2*datax-1;
s_p_datax=reshape(data_NZRx,2,length(datax)/2);
x=[];
x_in=[];
x_qd=[];

data_NZR=2*data-1;
s_p_data=reshape(data_NZR,2,length(data)/2);
br=10.^6;
f=br;
T=1/br;
t=T/99:T/99:T;

y=[];
y_in=[];
y_qd=[];


for(i=1:length(datax)/2)
    x1=s_p_datax(1,i)*sin(2*pi*fx*tx);
    x2=s_p_datax(2,i)*cos(2*pi*fx*tx);
    x_in=[x_in x1];
    x_qd=[x_qd x2];
```

```
    x=[x x1+x2];

    %x_qd=x_qd*0;
    % x_in=x_in*0;
    % x=x*0;

end
for(i=1:length(data)/2)
    y1=s_p_data(1,i)*sin(2*pi*f*t);
    y2=s_p_data(2,i)*cos(2*pi*f*t) ;
    y_in=[y_in y1];
    y_qd=[y_qd y2];
    y=[y y1+y2];

    % y_qd=y_qd*0;
    % y_in=y_in*0;
    % y=y*0;
end

maxX=max(x, [], 'all');
maxY=max(y, [], 'all');

Tx_sigx=x;
ttx=Tx/99:Tx/99:(Tx*length(datax))/2;
Tx_sig=y;
tt=T/99:T/99:(T*length(data))/2;

figure(3)

subplot(3,1,1);
plot(tt,y_in,'linewidth',3), grid on;
title(' Wave form for inphase component of Y polarization ');
xlabel('time(sec)');
ylabel(' amplitude(V)');
subplot(3,1,2);
plot(tt,y_qd,'linewidth',3), grid on;
title(' Wave form for Quadrature component of Y polarization');
xlabel('time(s)');
ylabel(' amplitude(V)');
subplot(3,1,3);
plot(tt,Tx_sig,'r','linewidth',3), grid on;
title('Sum of inphase and Quadrature phase signal of Y polarization)
    ');
xlabel('time(s)');
ylabel(' amplitude(V)');

figure(4)

subplot(3,1,1);
plot(ttx,x_in,'linewidth',3), grid on;
title(' Wave form for inphase component of X polarization ');
xlabel('time(s)');
ylabel(' amplitude(V)');
subplot(3,1,2);
plot(ttx,x_qd,'linewidth',3), grid on;
title(' Wave form for Quadrature component of X polarization  ');
xlabel('time(s)');
```

```
ylabel(' amplitude(V)');
subplot(3,1,3);
plot(ttx,Tx_sigx,'g','linewidth',3), grid on;
title('Sum of inphase and Quadrature phase signal of X component');
xlabel('time(s)');
ylabel(' amplitude(V)');

figure(5)
u = gradient(x);
v = gradient(y);

quiver(x, y, u, v)
axis([-2 2 -2 2])
xL = xlim;
yL = ylim;
line([0 0], yL);
line(xL, [0 0]);
xlabel('E0y');
ylabel(' E0x');
grid

figure(6)
thetax=angle(x);
thetay=angle(y);
thetafinal=thetax-thetay;
stem(ttx,thetafinal/pi)
xlabel 'Frequency (Hz)'
ylabel 'Phase / \pi'
grid

figure (7)
x = x - mean(x);
y = y - mean(y);

X=fft(x);
Y=fft(y);

[mag_x idx_x] = max(abs(X));
[mag_y idx_y] = max(abs(Y));

px = angle(X(idx_x))
py = angle(Y(idx_y))

phase_lag = py-px

%amplitude_ratio = mag_y/mag_x
PhDiff = phase_lag*180/pi

plot(tt, Tx_sig, 'r', 'LineWidth', 2)

grid on
hold on
plot(tt,Tx_sigx, 'g', 'LineWidth', 3)
xlim([0 0.000001])
ylim([-1.7 1.7])
set(gca, 'FontName', 'Times New Roman', 'FontSize', 14)
xlabel('Time, s')
```

```
ylabel('Amplitude, V')
title(['Phase Difference is ',num2str(PhDiff),' \circ'])

legend('Y component', 'X component')

figure (8)

dat_t = 0;
dat_0 = round(maxX^2+maxY^2);
dat_1 = round(maxX^2-maxY^2);  % Stokes vector S1
dat_2 = round((2*maxX*maxY*cos(PhDiff*pi/180))); % Stokes vector S2
dat_3 = round((2*maxX*maxY*sin(PhDiff*pi/180)));  % Stokes vector S3
dat_4 = 1;  % DOP

if dat_0 >0
    dat_0 = (maxX^2+maxY^2)/(maxX^2+maxY^2);
end
if dat_1 >0
    dat_1 = (maxX^2-maxY^2)/(maxX^2-maxY^2);
end
if dat_2 >0
    dat_2 = (2*maxX*maxY*cos(PhDiff*pi/180))/(2*maxX*maxY*cos(PhDiff
   *pi/180));
end
if dat_3 >0
    dat_3 = (2*maxX*maxY*sin(PhDiff*pi/180))/(2*maxX*maxY*sin(PhDiff
   *pi/180));  % Stokes vector S3
end
if dat_0 <0
    dat_0 = -(maxX^2+maxY^2)/(maxX^2+maxY^2);
end
if dat_1 <0
    dat_1 = -(maxX^2-maxY^2)/(maxX^2-maxY^2);
end
if dat_2 <0
    dat_2 = -(2*maxX*maxY*cos(PhDiff*pi/180))/(2*maxX*maxY*cos(
   PhDiff*pi/180));
end
if dat_3 <0
    dat_3 = -(2*maxX*maxY*sin(PhDiff*pi/180))/(2*maxX*maxY*sin(
   PhDiff*pi/180));  % Stokes vector S3
end

x = dat_1;
y = dat_2;
z = dat_3;
% plot data
figure('Position',[183 70 500 600]);

[X,Y,Z] = sphere(20);
X = X;
Y = Y;
Z = Z;
Hs = mesh(X,Y,Z,'facecolor','w','edgecolor',[0.5 0.5 0.5]);
caxis([1.0 1.01]);
alpha(0.70);
axis equal;
```

```
set(gcf,'Renderer','opengl');
hold on;

Hx = plot3([-1.5 1.5], [0 0], [0 0],'k-');
set(Hx,'linewidth',2,'linestyle','-','color','k');
ht_x = text(1.75,0,0,'+S_1','fontweight','bold','fontsize',12,'
    fontname','arial');

Hy = plot3([0 0], [-1.5 1.5], [0 0],'k-');
set(Hy,'linewidth',2,'linestyle','-','color','k');
ht_y = text(0.1,1.6,0,'+S_2','fontweight','bold','fontsize',12,'
    fontname','arial');
Hz = plot3([0 0], [0 0], [-1.5 1.5],'k-');
set(Hz,'linewidth',2,'linestyle','-','color','k');
ht_z = text(-0.05,0,1.35,'+S_3','fontweight','bold','fontsize',12,'
    fontname','arial');
ht_rcp = text(-0.05,0.0,-1.1,'|L>','fontweight','bold','fontsize'
    ,12,'fontname','arial','color','k');

% Draw a bold circle about the equator (2*epsilon = 0)
x_e = (-1:.01:1);
for i = 1:length(x_e)
    z_e(i) = 0;
    y_e_p(i) = +sqrt(1 - x_e(i)^2);
    y_e_n(i) = -sqrt(1 - x_e(i)^2);
end
He = plot3(x_e,y_e_p,z_e,'k-',x_e,y_e_n,z_e,'k-');
set(He,'linewidth',2,'color','k');
% Draw a bold circle about the prime meridian (2*theta = 0, 180)
y_pm = (-1:.01:1);
for i = 1:length(x_e)
    x_pm(i) = 0;
    z_pm_p(i) = +sqrt(1 - y_pm(i)^2);
    z_pm_n(i) = -sqrt(1 - y_pm(i)^2);
end
Hpm = plot3(x_pm,y_pm,z_pm_p,'k-',x_pm,y_pm,z_pm_n,'k-');
set(Hpm,'linewidth',2,'color','k');
% Now plot the polarimetry data
H = plot3(x,y,z,'m.');
set(gca,'fontweight','bold','fontsize',12,'fontname','arial');
set(H,'markersize',25,'markeredgecolor','r','markerfacecolor','r','
    color','r','linewidth',0.5);

view(135,20);
```

Listing B.1: Matlab simulation