Maria Ana de Matos Afonso Pereira
EXPERIMENTAL SEMI-DEVICE INDEPENDENT QUANTUM KEY DISTRIBUTION
1 2 9 0
UNIVERSIDADE Đ COIMBRA

1 2 9 0

UNIVERSIDADE Đ
COIMBRA

Maria Ana de Matos Afonso Pereira

EXPERIMENTAL SEMI-DEVICE INDEPENDENT QUANTUM KEY DISTRIBUTION

Dissertação no âmbito do Mestrado Integrado em Engenharia Física, ramo de Instrumentação, orientada pelo Professor Doutor Hugo Zbinden da Universidade de Genebra e coorientado pela Professora Doutora Maria Helena Vieira Alberto apresentada ao Departamento de Física da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Outubro de 2020

1 2 9 0

UNIVERSIDADE D
COIMBRA

Maria Ana de Matos Afonso Pereira

EXPERIMENTAL SEMI-DEVICE INDEPENDENT QUANTUM KEY DISTRIBUTION

Thesis submitted to the
University of Coimbra for the degree of
Master in Engineering Physics

Supervisor:
Prof. Hugo Zbinden

Co-Supervisor:
Prof. Maria Helena Vieira Alberto

Coimbra, October 2020

This work was developed in collaboration with:

Quantum Technologies Group - University of Geneva

UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES

This project was financed by the INSPIRE Potentials - QSIT Master Internship Award.

Resumo

O objetivo da distribuição de chaves quânticas (em inglês QKD, "quantum key distribution") é transferir de forma segura chaves de encriptação entre dois utilizadores através de um canal de comunicação não protegido, com recurso às propriedades da mecânica quântica. As provas de segurança de protocolos padrão de sistemas de distribuição de chaves quânticas, requerem uma caracterização completa das operações de medida e dos estados quânticos preparados. Estas suposições são, no entanto, impraticáveis numa aplicação real devido às imperfeições inerentes aos instrumentos físicos que são utilizados. A solução que surge naturalmente é a aplicação de um sistema de distribuição de chaves quânticas cuja segurança seja assegurada independentemente dos intrumentos experimentais utilizados. No entanto, aplicações práticas deste tipo de protocolos continuam a ser um grande desafio atualmente. A alternativa que surgiu foi uma abordagem semi- independente dos instrumentos utilizados. Nesta situação, os instrumentos utilizados não são caracterizados. A única caracterização a fazer é da informação do produto interno dos estados quânticos que codificam a informação enviada por Alice.

O meu projecto de mestrado tem como objectivo a implementação de um protocolo de distribuição de chaves quânticas semi-independente dos dispositivos usados. A dissertação incicia-se com uma breve exposição do estado da arte e introdução ao tema.

Segue-se um capítulo com a análise de como os estados quânticos serão preparados. A exatidão desta preparação tem um papel fulcral no funcionamento do protocolo. Para assegurar a sua precisão, é necessário fazer uma caracterização dos controladores de polarização utilizados para codificar os estados. Com base nesta caracterização, calcularam-se então os parâmetros necessários para preparar os estados quânticos. Neste segundo capítulo é também abordada a construção dos sistemas necessários para controlar a polarização.

Na terceira parte da dissertação a implementação experimental do protocolo semi-independente de dispositivos é analisada com mais detalhe. Os componentes utilizados são enunciados e a sua escolha é discutida e fundamentada. Os métodos utilizados para controlar toda experiência são também abordados. Isto inclui uma análise das técnicas de caracterização dos pulsos coerentes. Por fim é discutida a implementação experimental do protocolo.

Keywords— Distributição de Chaves Quânticas, Semi-Device Independent, Criptografia, Comunicações Quânticas

Abstract

The goal of quantum key distribution is to safely transfer secret data between two legitimate users through an unreliable network. This is done so by exploiting the properties of quantum mechanics. The security proofs of standard quantum key distribution protocols rely heavily on the characterization of the measurements and prepared quantum states. These assumptions, however, prove to be difficult to meet in real-life implementations. The obvious solution would come as device-independent (DI) security proofs. However, this type of implementation remains a challenge to this day. The alternative to DI found was a semi-device independent approach. Here the devices are non-characterized, and the only assumption made is the inner product information of the sent coherent states. As it is currently one of the most well-established quantum-information technologies, I shall provide a brief introduction and state-of-the-art of quantum key distribution.

In this dissertation, I will expound on the implementation of a semi-device independent quantum key distribution protocol. Firstly, state preparation is discussed. The accuracy of the state preparation as well as the measurement operation will have a great impact on the performance of the protocol based on polarization states encoded on weak coherent light pulses. To ensure these are correctly implemented, a full characterization of the polarization controllers used to encode the states is made. After that, the estimation of the parameters needed to prepare the desired polarization states and their respective optimization is explained. In this chapter, the building of the systems needed to control the polarization is also discussed.

In the second part of the dissertation, the experimental implementation of the semi-device independent protocol is examined in more depth. Here, the components used shall be specified and their choice is explained. The full control of the experimental set-up will also be discussed. This includes an analysis of the alignment procedures and a characterization of the weak coherent pulses. Lastly, we shall discuss the experimental realization of the protocol and the discussion of the obtained results.

Keywords— Quantum Key Distribution, Semi-Device Independent, Cryptography, Quantum Communication

List of Acronyms

DAC - Digital to Analog Converter

DI - Device Independent

EOM - Electro-Optic Modulator

MDI - Measurement-Device Independent

OTP - One Time Pad

PBS - Polarizing Beam Splitter

PD - Photon Detector

PM - Prepare and Measure

QBER - Quantum Bit Error Rate

QKD - Quantum Key Distribution

QND - Quantum Nondemolition

QRNG - Quantum Random Number Generation

SDI - Semi-Device Independent

SDP - Semidefinite Programming

SKC - Secret-Key Capacityy

SPD - Single-Photon Detector

SNSPD - Superconducting Nanowire Single-Photon Detector

TDC - Time-to-Digital Converter

TF - Twin Field

List of Figures

List of Tables

Contents

1 Introduction

Secure communication has been a concern of humanity for thousands of years, with the first use of cryptography being attributed to the Egyptians. Cryptography remained rudimental for a great number of years until World War II, where major advances were made, and the rotor machine Enigma opened the door for more complex and sophisticated means of encoding and decoding.

With the development of integrated circuits in the 1960s, came the Third Generation of computers. Around this time the usage of computers increased greatly and so did the concern of the private sector with security, resulting in an increased demand for security services. This also gave a new dimension to cryptography, distancing it from the military and governmental affairs and bringing it closer to the common public.

Cryptography is now defined as a set of techniques that ensure confidentiality, data integrity, authentication, and non-repudiation of information [1].

1.1 Cryptography

Confidentiality is ensured with the use of a key. This key is combined with the information one wishes to keep secret, resulting in a cryptogram. This process is called encryption. For a cryptosystem to be considered secure, it must only be solved with the use of its corresponding key [2].

In classical cryptography, there are two types of cryptosystems. Symmetrical systems in which Alice and Bob share the same key, and asymmetrical systems, where Alice and Bob use different keys.

Public key

In asymmetrical systems, also known as public-key systems, Bob creates a private key that only he has access to. From this key, Bob creates a public key. The public key is then used by Alice to encrypt her message, which in turn can only be decrypted using Bob's private key.

The first practical application of asymmetric cryptosystem was in 1978 by Rivest, Shamir, and Adleman, in a protocol named after them - RSA. It is based on the mathematical difficulty of factoring integers [3]. This protocol is still largely used to this day. The main idea of public-key systems is then to use one-way functions i.e. functions $z = f(x,y)$ is easy to compute for a given x,y but the inverse function $f^{-1}(z)$ is arduous to obtain with the computational resources at disposal. 'Arduous' means the solving time increases exponentially with the amount of information. So the security of these systems is not based in mathematical principles but rather computational limitations, and time constraints.

However, further developments in quantum computing, will allow for the implementation of Shor's Algorithm, which allows the factorization of integers and the finding of discrete logarithms [4]. It's implementation will reduce greatly the time it would take to solve these factorization problems, allowing for a violation of these public keys.

Private key

In symmetrical systems, also referred to as private-key systems, the same key is used to encode and decode the messages. Therefore, both Alice and Bob must have the said key in their possession [1].

The only truly theoretically unbreakable method of encryption, the One-Time Pad (OTP), is an example of application of a symmetrical system. OTP requires the use of an encryption key that is at least as long as the message one wants to send. Each element of the message will then be combined with an element of the key, thus creating the cipher. To ensure the OTP's security the following conditions must be met:

- The OTP is comprised of truly random elements;
- It should be as long as the message;
- It should never be re-used;
- Its secrecy shall be preserved even after use.

These conditions raise a few problems. The first and most evident one is achieving true randomness. A second issue may arise with the treatment of the key after its first use. The OTP's should be discarded properly in order to assure they remain secret.

From this type of cryptosystems also arises the "key distribution problem". Since both Alice and Bob need to have the same key, it must be securely transferred to both parties. If someone intercepts even part of it during transmission, they can decipher a future message encrypted with that one OTP. Unfortunately, the security of this transmission cannot be completely assured by classical methods [5].

The shortcomings of classical cryptography are evident. Further advances in computational sciences or mathematics may render asymmetric cryptosystems useless. The question is whether there is a way to solve the key distribution problem in symmetric cryptosystems? The answer is yes, using quantum key distribution.

1.2 Quantum Cryptography

The aim of QKD is to distribute a secret key using the properties of quantum mechanics. Unlike classical transmissions, in QKD, the information cannot be copied and stored as a consequence of the non-cloning theorem [6]. This theorem also applies to a potential eavesdropper.

Moreover, when polarization is encoded in non-orthogonal quantum states, any attempt to obtain information on the communication, will result in a high probability of a disturbance of the transmission [7]. This allows the two legitimate participants in the communication, Alice, and Bob, to detect an unauthorized third party who has had access to the information.

A quantum channel will not be used to distribute a message but rather the encryption key, which is no more than a random sequence of bits containing no crucial information. They will then have to communicate through an authenticated classical channel for the safety analysis. If an intrusion is detected Alice and Bob can simply discard the unsafe key and try again [2].

The Qubit

In classical information, the elementary unit is the bit, which can take one of the values 0 or 1. They can be used to represent classical properties for example, 0 if a switch in a circuit is open and 1 if it is closed. Bits can be stored, copied, and read as many times as one desires without the loss of information. However, some physical states cannot be described by a classical bit. Quantum systems do not follow the rules set by deterministic classical mechanics, thus not allowing them to be described by bits. A solution to that is the qubit.

Unlike his classical counterpart, the qubit is a state in a two-dimensional Hilbert space. The qubit can take any value of $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ for α and β complex. Where $\mid \alpha \mid^2 + \mid \beta \mid^2 = 1$. With a measurement operation, one can project the qubit into one of the bases $|0\rangle$ or $|1\rangle$ with a probability of either $\mid \alpha^2 \mid$ or $\mid \beta^2 \mid$ respectively [8].

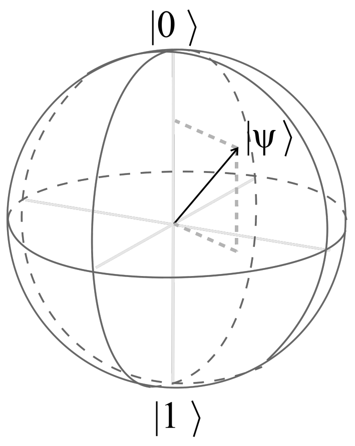


Figure 1.1: Graphical representation of the Bloch sphere.

Qubits can be represented by a vector in 3D space. A common representation is when they fall into a sphere of radius 1 known as the Bloch sphere. Its representation can be seen in Fig.1.1 In the poles we will have the states 0 and 1. Antipodal points in the sphere are orthogonal and form a base. Points in the surface of the sphere represent pure states, any point in the interior will represent a mixed state [9].

1.2.1 Fiber Based QKD

The act of measurement is inherently different in quantum mechanics than it is in classical physics. The main security advantage of QKD relies precisely in the limitations imposed by the act of quantum measurement.

A QKD protocol can be looked at in two steps, the quantum communication and a classical post-processing. In the first phase, Alice encodes classical information in quantum states, creating the optical signal. The states are sent to Bob through a quantum channel. We assume this quantum channel is under Eve's control. Bob receives and measures the states sent by Alice, as a result, he acquires a sequence of classical bits enclosing information.

During classical post-processing, both legitimate parties perform error corrections and privacy amplifications (reducing the amount of information acquired by Eve) and a key sifting. In the key sifting, Alice and Bob communicate through a classic channel to filter out unusable preparation/measurement pairings. After performing both stages, Alice and Bob will have a secret key.

1.2.1.1 BB84

The first QKD protocol was proposed by Bennet and Brassard in 1984, hence the given name BB84 [2].

This protocol has a basic principle using photons and polarization to encode the infor-

Quantum Communication											
Alice Bits	0	0	1	1	1	0	1	1	0	1	0
Encoding Basis	D	R	D	R	R	R	D	R	D	D	R
Photon Polarization	↗	↔	↖	↕	↕	↔	↖	↕	↗	↖	↔
Bob Measuring Basis	R	D	D	R	D	R	R	D	D	D	R
Bob's Bits	1	0	1	1	0	0	0	1	0	0	0
Discussion											
Bob's Base Reveal	R	D	D	R	D	R	R	D	D	D	R
Alice's Basis Confirmation			D	R		R			D	D	R
Shared Bits			1	1		0			0	1	0

Table 1.1: Secret key distribution in the BB84 implementation. Table adapted from the original paper [2]. **D** and **R** stand for diagonal and rectilinear basis respectively.

mation. Alice chooses a random string of bits and an equally . long sequence defining the corresponding polarization basis to be used, which can be either rectilinear, R, or diagonal, D. Alice then encodes the previous bit sequence in a train of photons using the bit's corresponding base.

Bob chooses randomly the basis (rectilinear or diagonal) in which he measures the received photon. He only gets relevant information when he guesses the basis correctly (about 50% of the time). Note that when measuring on a basis different from the one the photon was encoded in, Bob gets a random answer and induces a loss of the previously encoded information.

The next steps of the protocol take place on an authenticated public channel. This is a classic channel susceptible to eavesdropping but no message traveling through it can be altered.

Firstly, Alice and Bob must check which photons Bob received. Afterward, they determine the photons that were measured on the correct basis and discard the others. Alice and Bob can then check for eavesdropping by comparing publicly a small sequence of the remaining bits. These then must be discarded since their secrecy was renounced. If the bits appear undisturbed, free of significant eavesdropping, the sequence of remaining bits can then be used as a secret key.

Further analysis of the protocol can be found in [2].

One limitation of this protocol is that it assumes perfect single-photon sources i.e. sources that emit one photon at a time. These sources are incredibly hard to build and not widely available. A simpler alternative is to use attenuated coherent sources. However, attenuated coherent pulses are comprised of a vacuum and a multi-photon component. It is this multi-photon component that creates complications. An hypothetical eavesdropper may split the multi-photon signals, keeping one to themselves and sending the remaining to Bob [10]. These multi-photon sources then lead to a new type of eavesdropping attacks, photon splitting attacks.

Decoy-States

To correct for this, decoy-state protocols are used [8]. In these protocols, Alice prepares signal states that encode the information to create the key as well as decoy sates that are used to detect eavesdropping attacks. An in-depth analysis can be found in [11].

A decoy-state protocol was first implemented in 2006, achieving distances of 15km and a key rate of 165bits per second [12].

The current record distance using decoy-state protocols was set in 2018 [13]. The protocol was based on a simplification of the BB84 protocol, using two states in the Z basis (to generate the raw key) and one decoy-state in X the basis (to estimate the eavesdropper information) [14]. A. Boaron et. al. achieved a key distribution over 421km with a SKR of 0.25 bits per second, and 6.5 bits per second over 404.9km. Details of the protocol can be viewed in [14][15].

1.2.1.2 Device Independent-QKD

Although the security of QKD is based on the principles of quantum mechanics, any inconsistency in the physical implementation due to technological shortcomings can cause a breach in security. These practical imperfections were first exploited in reference [16] where the authors demonstrated a security breach without it being detected by the legitimate parties. More specifically, they use the vulnerability of single-photon detectors being blinded under strong illumination. By using this method, the gated detectors are converted into classical linear detectors allowing them to be fully remote-controlled by an external party[17].

All the features of the real devices that are not modelled in the security proof will therefore compromise the security of the protocols. When these inconsistencies in the physical implementation are exploited for attacks performed by Eve, they have the name of side-channel attacks. This has led to a cycle of finding security breaches in the implementations and then finding more generalized security proofs to close them [18]. A way to escape this never-ending cycle is using Device-Independent protocols (DI).

A protocol is deemed device-independent if its security is conserved heedless of the quality of the devices used in its implementation, and no assumptions about them are needed. The security is derived only from the classical input-output relationship observed by the legitimate parties, the validity of quantum mechanics, as well as the safety and isolation of

Alice and Bob's physical location. Without this last inference, any type of cryptography will completely loose its meaning [19].

As a fundamental aspect of DI-QKD relies on Bell's theorem. Therefore, to understand the basis of DI-QKD, we must first take a look at the paper where Einstein, Podolsky, and Rosen presented the famous EPR paradox [20]. In this paper, they concluded that quantum mechanics is not a complete theory and its intrinsic "randomness" is justified by the existence of "hidden variables". In 1965 Bell disproved this conclusion and demonstrated it to be incompatible with the statistical predictions of quantum mechanics [21]. A few years later, Clauser, Horne, Shimony, and Holt presented a generalization of Bell's theorem and proved the Bell test to be decisive in the experimental assessment of the theories of "hidden variables". In this paper they also derived one of the most studied of Bell's inequalities, the CHSH inequality [22].

$$S \equiv E_{xy} + E_{x'y} + E_{xy'} - E_{x'y'} \tag{1.1}$$

Where as predicted by the theory of hidden variables:

$$\mid S \mid \leq 2 \tag{1.2}$$

In a Bell experiment, we have two measurement devices A and B, how they work is irrelevant and they can be deemed black boxes. Each device has an input (measurement settings) and an output (measurement results). For every run A and B randomly choose an input (x or y) and measure the corresponding outcome (a or b). After the desired number of trials, A and B compute together the statistics of what they observed [23]. As stated previously, when performing a Bell experiment, quantum mechanics predicts a violation of inequality 1.2.

In 1991 Ekert found an application for the generalized Bell's theorem in QKD [24]. The scheme he proposed exploits the CHSH inequality to test for eavesdropping. A single source emits a pair of particles with spin $\mid \frac{1}{2} \mid$ and sends one to each legitimate user, Alice, and Bob. Once they receive the particles, Alice and Bob measure their spin component concerning one of two avalilable basis (x,x for Alice and y,y for Bob).

After the quantum communication is over, Alice and bob announce the bases used. Any round with a failed measurement is discarded. When both measured in bases with a different orientation, they reveal the measurement outcomes to estimate S. The correlation for when Alice and Bob measure with bases with a different orientation can then be calculated. In this case, due to the mathematical formalism of quantum mechanics for S

we will for a choice of basis which maximizes $\mid S \mid$,

$$\mid S \mid = 2\sqrt{2} \tag{1.3}$$

If S has not been disturbed, then the outcome should be the one of equation 1.3. However, if an eavesdropper has had access to the system, one should measure:

$$-\sqrt{2} \leq S \leq \sqrt{2}$$

Thus, not violating the CHSH bound, revealing this way the presence of the eavesdropper. Based in Ekert's intuition, Acín et. al. [25] proposed a device independent security scenario. In their approach, Alice has three possible measurement basis $\{A_0; A_1; A_2\}$ and Bob two $\{B_0; B_1\}$ and two possible measuring outcomes $a_i, b_j \in \{+1, -1\}$. The key is obtained from the measurement settings $\{A_0,B_1\}$. As a result, the QBER (quantum bit error rate) can be calculated as the probability of the outcomes a_0 and b_1 being different:

$$Q = prob(a_0 \neq b_1)$$

And the secret key rate is given by

$$r \geq 1 - h(Q) - \chi(B_1 : E)$$

Where $\chi(B_1 : E)$ is known as the Holevo quantity between Eve and Bob. The Holevo bound is a measure of the upper bound of the amount of information that can be known about a quantum state. In this case, it relates to what Eve can know about Bob. To find Eve's optimal attack χ must be optimized to the maximum. It can be proved [25] that the largest value for the Holevo quantity between Eve and Bob in a CHSH experiment yielding an S value, is given by:

$$\chi(B_1 : E) \leq \left(\frac{1 + \sqrt{\left(\frac{S}{2}\right)^2 - 1}}{2} \right) \tag{1.4}$$

The key rate can then be plotted for values of both Q and S. This ensures security for a sufficiently large violation of the CHSH bound.

To successfully implement a DI protocol, one must close simultaneously the *locality loophole* and the *detection loophole*. These loopholes are nothing more than flaws in the experimental implementation that compromise the validity of a Bell test. The first loophole, the *locality loophole*, must be closed due to concerns regarding communication between Alice and Bob during the experiment. If, for example, Bob, can know the measurement choices of Alice in real-time, then the violation of Bell's inequality becomes inconsequential. This

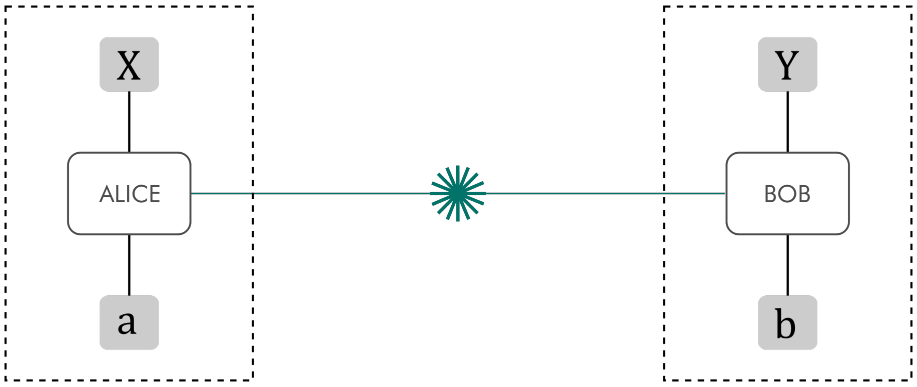


Figure 1.2: Device independent protocol where x and y are the randomly chosen measurement settings and a and b are the measured outputs.

loophole can be closed by performing the measurements with space separation.

The *detection loophole* however, proves to be more troublesome to be closed. It refers to the losses in the quantum channel resulting in not all photons being detected. This effect can be neglected in Bell tests with the fair sampling assumption. However, it is naive to assume that an eavesdropper has no malicious intent. As a result, the closing of this loophole is mandatory for any DI implementation [26]. For that, a total efficiency of over 82.8% must be ensured to dismiss possible attacks based on the detection loophole [27]. This transmission problem may be bypassed by performing QND (quantum-nondemolition measurements) [28], using quantum repeaters, or using a heralded Qubit amplifier [29].

Nonetheless, DI protocols, have not yet been successfully implemented due to the hardship of single-photon detection and generation of entangled states. Below a certain detection threshold, no key can be extracted. Satisfactory key extraction rates have been obtained but in turn, their security has been compromised [30][31].

1.2.1.3 Measurement Device Independent-QKD

As DI protocols prove to be impractical, since they require near-unity detection efficiency and QND (quantum non-demolition) measurements, another solution to remove the concern with detector side-channel attacks was presented in 2012. Lo et. al. [32] proposed a Measurement-Device-Independent (MDI) protocol. Unlike "full" DI-QKD, this protocol requires the assumption that Alice and Bob, must have almost perfect control over the preparation of their quantum states. This removes any concern one may have with the measurement apparatus.

In this approach, both Alice and Bob prepare weak coherent pulses in one of the four polarization states used in BB84 and send them to an untrusted third party, Charlie. This middle party, Charlie, can be the eavesdropper usually call Eve. It is this middle party that performs the Bell state measurements and must announce the outcomes to Alice and

Bob. Note the states sent by Alice and Bob are statistically independent. Fig.1.3 presents a schematic representation of a MDI protocol.

Its properties make MDI-QKD ideal for communication networks with star-type topology, with all users connected to the middle node and having the measurement device accessible to all on-demand. As a result, MDI networks are more cost-effective and simpler to implement than their ”Prepare and Measure” counterparts, where all users are required to have both a sender and receiver module. With their added appeal, MDI-QKD boosts the commercial viability of quantum communications.

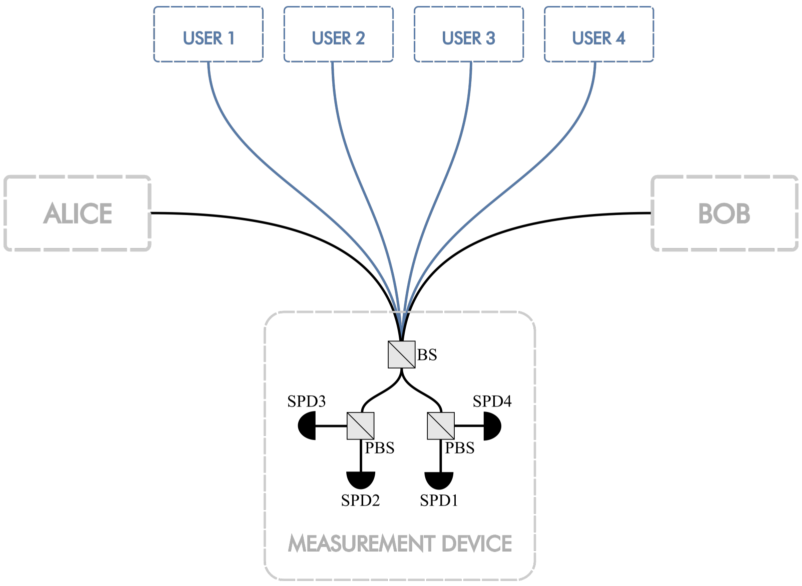


Figure 1.3: Simplified scheme of a SDI protocol. In light blue users were added to demonstrate a star-type topology.

1.2.1.4 Twin Field-QKD

A considerable focus of research in QKD is increasing the transmission distance as well as key rates. However, as it was proven by Pirandola et. al. in [33] it is impossible to overcome a certain limit without the use of quantum repeaters. The maximal acquirable secret-key is bounded by the secret-key capacity (SKC) of a quantum channel. This bound is known as the PLOB (Pirandola-Laurenza-Ottaviani-Banchi) bound:

$$R_{PLOB} = -\log_2(1-\eta)$$

η being the transmissivity of the communication channel.

The SKC of a quantum channel quantifies the maximum amount of information that can be transmitted in QKD.

Up until the proposal of Twin-Field QKD [34], no existing QKD scheme had surpassed this SKC bound. In [34] it was predicted that TF-QKD subjected to realistic parameters would overcome the ideal repeaterless bound in optical fiber at a distance of 340Km. This distance advantage comes from the fact that the proposed TF protocol scales at $\eta^{\frac{1}{2}}$ with transmittance instead of η like conventional QKD protocols.

MDI-QKD	**TF-QKD**
Alice and Bob send two photons to Charlie	Alice and Bob send two optical fields to Charlie
Two-photon interference	Single-photon interference
Coincidence detection	Single-photon detection

Table 1.2: Comparison table between Measurement Device Independent and Twin Field QKD.

Similarly to MDI-QKD, in TF-QKD, Alice and Bob are both transmitters and a middle node, Charlie is responsible for the measurements. In Table 1.2 a comparison between TF and MDI can be seen. Alice and Bob each have one light source and one interferometer arm. Charlie then makes the two pulses interfere on a beam splitter to later be detected by a single-photon detector.

Recently, experimental implementations of variations of the original TF protocol have been carried out, and in [35], the fundamental rate-distance limit of QKD was finally overcome at a distance of 300km. The record distances attained with TF protocols were 509km in fiber [36].

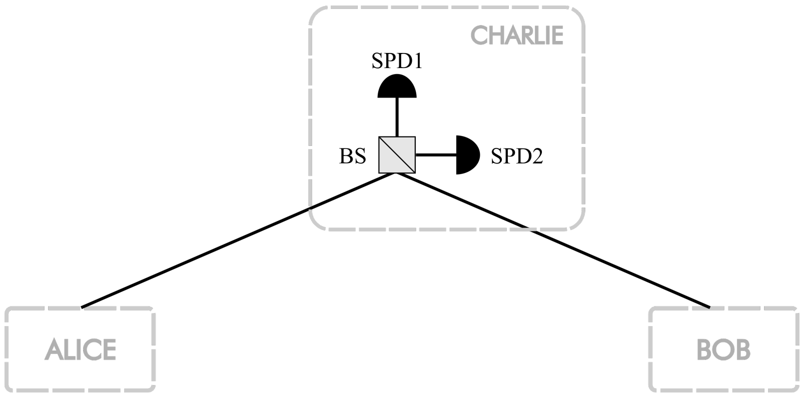


Figure 1.4: Schematic representation of a Twin-Field QKD protocol.

1.2.2 Satellite QKD

So far only optical fiber quantum channels have been mentioned in this review. However, QKD protocols are also suited to be applied in free space. Satellites will play a major role in quantum communications as they do currently with classical information. Satellites, while creators and distributors of a secret key, will allow communications between separated networks on the ground [37].

A single-photon source on a satellite was first established in 2003 [37]. The experiment used the Matera Laser Ranging Observatory, belonging to the Italian Space Agency, to both detect and transmit photons. The ground station, the Matera Laser Ranging Observatory, emitted weak laser pulses towards the satellite. The satellite equipped with retroreflectors reflected a fraction of the pulse (less than one photon per pulse) towards the receiver at the ground.

This technique matured and, quantum communication between a satellite and ground was established, as they demonstrated the preservation of single-photon polarization over a greater length when compared to ground experiments [38]. This procedure presents itself as a possible candidate for space-to-earth QKD.

Asia has been leading the investment in satellite quantum communications. In 2016 China launched the Micius Satellite. The satellite featured a Quantum Optics laboratory that allows it to generate coherent entangled states and measure received qubits [39].

A quantum teleportation uplink was established between a ground station in Tibet and the Micius satellite. This setup achieved a fidelity of 0.80 $\pm$ 0.01 [40]. QKD using a decoy-state version of the BB84 protocol was performed obtaining a key rate of over 10kbps with channel loss of 22dB [41]. The satellite also demonstrated entanglement-based QKD between two ground stations separated by 1200km with a key rate of 0,5 bps [42].

The experiments mentioned have taken place during the night-time, but they can also be performed during the day, as long as one strongly rejects background radiation, and reduce the temporal integration intervals of the incident qubits [43]. Another important aspect to consider in satellite-to-ground and ground-to-satellite quantum communications is the effect of atmospheric turbulence. When an uplink is established the turbulence is at the beginning of the path. This results in beam diameter broadening and scintillation that will affect the link transmissivity. On the other hand, when we have a downlink, photons will only suffer atmospheric effects by the end of their path. This will result in the diffraction of the photons and significative less scintillation.

1.3 Semi-device Independent-QKD

Before looking at our Semi Device Independent (SDI) protocol let us quickly review the main characteristics of the types of QKD previously mentioned. In a full device-dependent protocol, the users must characterize completely the prepared states, the measurement apparatus, and the security proofs only hold if the states sent are qubits. Trying to stray away from these limitations, the device-independent approach was proposed. In this method, the security of the implementations is guaranteed based on an analysis of the input/output relationship while also requiring a Bell test violation. Since the implementation of pure DI-QKD proved to be difficult, a MDI approach was proposed. the MDI approach was proposed. Here we have a relaxation of the assumptions on the measurement apparatus however still needing well-characterized state preparation.

1.3.1 The security

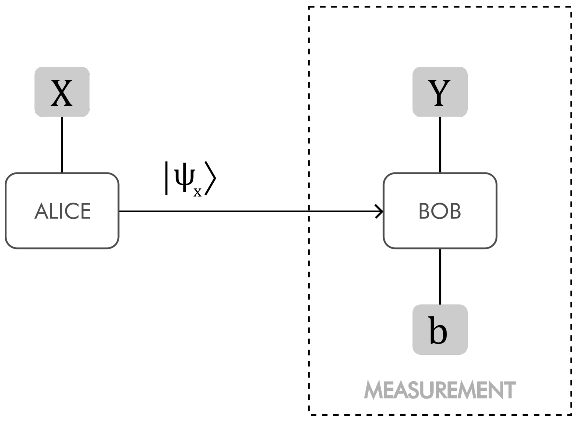


Figure 1.5: Schematic representation of the semi-device independent protocol.

The approach presented in this thesis (Fig.1.5) can be deemed SDI since its security analysis does not require full knowledge of either the operation of the network or of the measurements. For encoding the quantum state prepared by Alice $|\psi_z\rangle$, this method only requires the characterization of of a matrix known as Gram matrix, G, that can be derived from inner products of the quantum states [44].

$$G = \sum_{z,z'=1}^{n} G^{z,z'} \otimes |e_z\rangle\langle e_{z'}|$$

$$for \sum_{z,z'=1}^{n} G^{z,z'} = \langle\psi_z|S_i^\dagger \cdot S_j|\psi_z\rangle$$

$\sum_{z,z'=1}^{n} G^{z,z'}$ is the inner product of vectors $\langle\psi_z|S_i^\dagger$ and $|\psi_z\rangle S_j$. S_j and S_i are operators of the set $S = \{S_1, ..., S_n\}$. $|e_z\rangle_z^n$ are representations of a standard orthonormal basis in $\mathbb{R}^n$.

This feature presents an added benefit when compared to previous SDI methods [45] [46], for the dimension of the system of encoding (often difficult to fix) is no longer necessary to bound the security of the protocol. The inner-product knowledge is enough to characterize the quantum-set as it tells how non-orthogonal the encoded states are. This approach can be used since while working in a high-dimensional Hilbert space, the transmission channel can be seen as an isometric evolution. This means that, even though the dimension and other properties of the sent state $|\psi_z\rangle$ may change in transmission, the inner product remains the same:

$$\langle\psi_z|\psi'_z\rangle = \langle\phi_z|\phi'_z\rangle = \lambda_{zz'}$$

Note that the entropy of a variable, in information theory, is a measurement of the amount of information one can extract from it [47]. It is then fundamental that the amount of information Eve can extract from her measurements is less than what Bob can extract. Therefore, the security is bound by:

$$H_{min}(A \mid E) – H(A \mid B) \geq 0$$

where $H(A \mid E)$ is the amount information produced from A, for measurement E, and $H_{min}(A \mid E)$ is given by:

$$H_{min}(A \mid E) = -\log_2(p_g(e = x))$$

Where p_g is the probability of Eve guessing correctly the state prepared by Alice in a valid round - we will see what makes a round valid a few paragraphs down.

To bound the security of our key distribution, then, we must get an estimation of p_g, based on the expected statistics. This characterization problem is, however, unmanageable to solve directly. The solution is then to use semidefinite programming (SDP), more specifically, a computational toolbox introduced by Wang. et. al. in [48] to get an approximation of p_g.

The security analysis has also bound the amount of losses allowed in the quantum channel. As seen in Fig.1.6, the lowest transmission of the quantum channel allowed in order to generate key for $\frac{\pi}{4}$ is 55%.

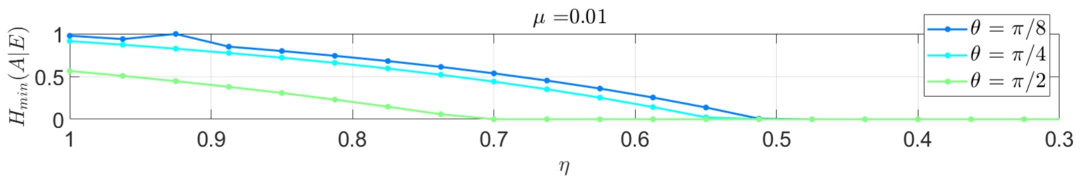


Figure 1.6: Plot of Alice-Eve entropy bound in function of channel losses.

1.3.2 The protocol

B92

Before expounding on the protocol we have implemented, it is worth going back to 1992 to Bennett's paper [49], where he demonstrated that QKD could be performed with any two nonorthogonal quantum states.

In the protocol B92, Alice prepares randomly one of two nonorthogonal states. Let us consider states $|0\rangle$ and $|+\rangle$, representing bit 0 and 1 respectively. Note that these states are non-orthogonal. Bob will choose randomly the basis for his projective measurements, either X or Z. The possible outcomes can be seen in Table 1.3.

Bob \ Alice		$\vert + \rangle$	$\vert 0 \rangle$
X	$\vert + \rangle$	1	$\vert \frac{1}{2} \vert^2$
	$\vert - \rangle$	0	$1- \vert \frac{1}{2} \vert^2$
Z	$\vert 0 \rangle$	$\vert \frac{1}{2} \vert^2$	1
	$\vert 1 \rangle$	$1- \vert \frac{1}{2} \vert^2$	0

Table 1.3: Representation of the probabilities of the possible outcomes measured by Bob, given Alice prepared state $|+\rangle$or $|0\rangle$, and Bob measured in basis X or Z, assuming no losses.

From Table 1.3, it is easy to see that if Alice prepares $|+\rangle$, Bob will never measure the state $|-\rangle$ and the same happens for prepared state $|0\rangle$ and measured state $|1\rangle$. Knowing this, if Bob measures the states $|1\rangle$ and $|-\rangle$, he can infer that Alice had to prepare the states $|+\rangle$ and $|0\rangle$ respectively, thus acquiring bits 1 and 0, in this order. Results where anything else is measured, or nothing is measured. are considered inconclusive and should be discarded by both users in the sifting.

Semi-device Independent

The SDI scenario we will implement, can be seen as a generalization of protocol B92. Here Alice begins by randomly choosing the "basis" in which she will prepare the states among 4 possible alternatives. Each basis corresponds to a different combination of two states 0 - $\{\psi_1,\ \psi_0\}$; 1 – $\{\psi_2,\ \psi_1\}$; 1 - $\{\psi_3,\ \psi_2\}$; 3 – $\{\psi_3,\ \psi_0\}$. After that, Alice picks one of the available states from the chosen basis, prepares it, and sends it to Bob. Bob picks one basis to perform his measurement in $M_{b|y}$ for $y \in \{0,1,2,3\}$ and $b \in 1,2$. In Table 1.4 we can see the possible outcomes of this measurement.

After the quantum communication stage, only for when $b = 2$ in $M_{b|y}$, Alice is asked to reveal her preparation basis. In other instances, the round is discarded. Then Bob compares his measurement basis y with the available states from Alice's preparation basis. If y is equal to one of the possible i, then the round is kept, if the requirement is not met, the round is scrapped. Let's assume Bob measured in $M_{2|0}$ and Alice announces she prepared her state in basis 0. Then, Bob could infer that, if he detected something, Alice must have prepared state ψ_1. This is how the information of the key will be extracted.

To test this protocol, we shall implement the experimental scheme shown in Fig.1.7. A

B \ A		0		1		2		3	
		ψ_0	ψ_1	ψ_1	ψ_2	ψ_2	ψ_3	ψ_3	ψ_0
0	ψ_0	1	$\mid \delta \mid^2$	╲	╲	╲	╲	$\mid \delta \mid^2$	1
	$\psi_0 *$	0	1-$\mid \delta \mid^2$	╲	╲	╲	╲	1-$\mid \delta \mid^2$	0
1	ψ_1	$\mid \delta \mid^2$	1	1	$\mid \delta \mid^2$	╲	╲	╲	╲
	$\psi_1 *$	1-$\mid \delta \mid^2$	0	0	1-$\mid \delta \mid^2$	╲	╲	╲	╲
2	ψ_2	╲	╲	$\mid \delta \mid^2$	1	1	$\mid \delta \mid^2$	╲	╲
	$\psi_2 *$	╲	╲	1-$\mid \delta \mid^2$	0	0	1-$\mid \delta \mid^2$	╲	╲
3	ψ_3	╲	╲	╲	╲	$\mid \delta \mid^2$	1	1	$\mid \delta \mid^2$
	$\psi_3 *$	╲	╲	╲	╲	1-$\mid \delta \mid^2$	0	0	1-$\mid \delta \mid^2$

Table 1.4: Probabilities of Bob detecting an event, given Alice prepared state ψ_i, for $i \in \{0;1;2;3\}$, and Bob measured in basis y, for $y \in \{0;1;2;3\}$.
Note that $M_{1|y} = \psi_y = |\psi_y\rangle\langle\psi_y|$ and $M_{2|y} = \psi_y * = \mathbb{1} - |\psi_y\rangle\langle\psi_y|$ and $\mid \langle\psi_y|\psi_x\rangle \mid^2 = \mid \delta \mid^2$

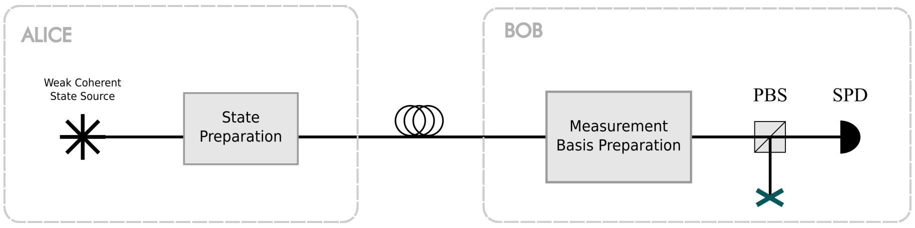


Figure 1.7: Simplified schematic of the proposed SDI-QKD protocol.

telecom pulsed laser will be used as the source of weak coherent pulses, where the key will be encoded. A polarization controller will be used to prepare the states Alice sends to Bob. Similarly, another polarization controller, with the help of a polarizing beam splitter, will be used to perform Bob's measurement. The photons resulting from the measurement operation will then be detected at a single-photon detector with high-efficiency.

2

State Preparation

The prepared states and the measurement bases are a crucial part of the experiment. The accuracy of the states determined by theory and the agreement between prepared states and the measurement basis will have a great impact on the performance of the protocol. As a result, it is required testing and full characterization of the Polarization controllers used.

All the codes used for the microcontrollers were written with Arduino Software (IDE) version 1.8.10. All Python programs mentioned were written using Python 3.8.5.

2.1 The Poincaré Sphere

There are many many degrees of freedom to a qubit, but for this work, we opted to use the polarization states of photons. The polarization of a photon, like in any electromagnetic wave, is the direction of the wave's electric field. The photon can be linearly, circularly, or elliptically polarized, depending on the oscillation of the electric field.

A more convenient way to describe polarized light was given by Sir George Gabriel Stokes. He proposed a characterization based on four measurable properties that allowed the portrayal of polarized and unpolarized light. These four parameters are now known as the Stokes parameters S_0, S_1,S_2,S_3.

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

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

$$S_2 = 2E_{0x}E_{0y}\cos(\delta)$$

$$S_3 = 2E_{0x}E_{0y}\sin(\delta)$$

S_0 represents the total intensity of the light, S_1 S_2, and S_3 the amount of linear horizontal and vertical, linear antidiagonal and diagonal, and circular left and right in the light beam, respectively. The Stokes parameters can also be written as:

$$S_1 = S_0 \cos(2\theta)\cos(2\eta)$$

$$S_2 = S_0 \cos(2\theta)\sin(2\eta)$$

$$S_1 = S_0 \sin(2\theta)$$

and follow the relationship:

$$S_0^2 \geq S_1^2 + S_2^2 + S_3^2$$

Where E_{0x} and E_{0y} are the amplitudes of the electric field in the x and y directions, respectively. The angle δ is the initial phase difference between the two components of the electric field.

This relationship turns to $S_0^2 = S_1^2 + S_2^2 + S_3^2$ when we have completely polarized light.

One can use the angles θ and η to represent the polarization of an optical beam in spherical coordinates. The polarization state can then be represented as a point in a sphere with a unitary radius. The origin will correspond to unpolarized light and in the surface is represented completely polarized light. The spere depicted in 2.11 is named Poincaré sphere.

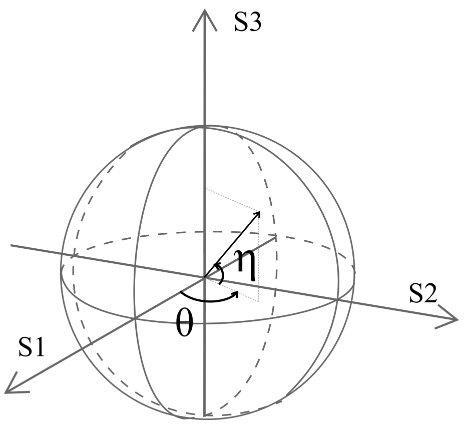


Figure 2.1: The Poincaré sphere.

As a result of these properties, any polarization state of monochromatic light can be represented as a single point on the Poincaré sphere. The degenerate states shown in 2.2 are Linear: Horizontal $|H\rangle$, Vertical $|V\rangle$, Diagonal $|D\rangle$, Antidiagonal $|A\rangle$, and circular: Left $|L\rangle$, Right $|R\rangle$. The linear polarized states lie among the equator and the circularly polarized can be found in the two poles. All other points represent states with elliptical polarization.

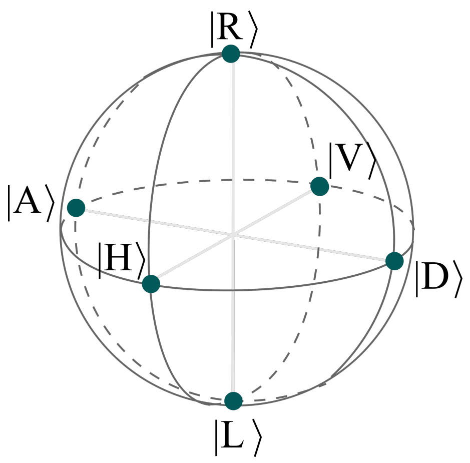


Figure 2.2: Degenerate polarization states shown in the Poincaré sphere.

2.2 Controlling the Polarization

To prepare the 4 polarization states needed for Alice's encoding and Bob's basis selection, we used two General Photonics' PolaRITE^{TM} III polarization controllers. These polarization controllers were chosen due to their low insertion loss (0.05 dB without connectors and 0.3 with connectors).

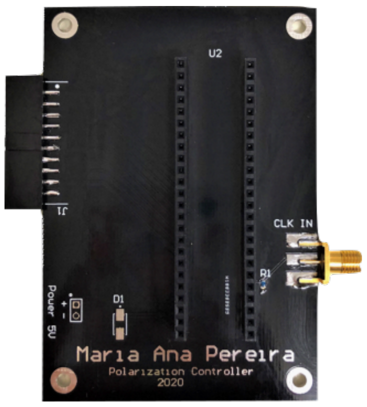
2.2.1 Operation

The polarization controller is comprised of 4 piezoelectric fiber squeezers. The squeezers, driven by voltage, deform the fiber resulting in a linear birefringence that will alter the polarization of passing light.

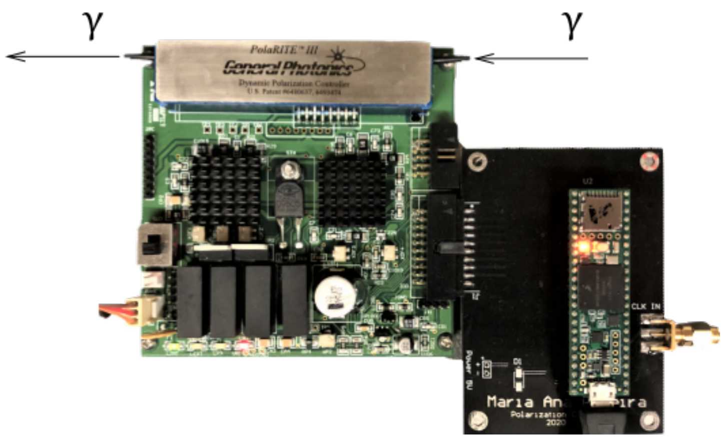
Since the squeezers have different orientation, pressing each one will result in a different shift in polarization. In principle, an increase and decrease of voltage in entry channels 1 and 3, results in a clockwise and counterclockwise rotation around $|A\rangle$ respectively. A similar rotation, orthogonal to the first one is seen around $|H\rangle$ when tension is applied to entry channels 2 and 4. With only two orthogonal channels it is then possible to achieve any desired polarization. These shifts in polarization can be visualized as rotations in the Poincaré sphere 2.2.

The polarization controller is then mounted on piezoelectric driver boards (MPD-001). The board is controlled by a digital input signal. The digital inputs accept TTL levels for the read/write, reset, chip select, 2 channel, and 12 data pins. We used a microcontroller (Teensy 3.6) to provide this digital signal. The digital input line operates sequentially, the 12-bit voltage is written to one channel at a time. The timing sequence should go as follows: the TTL levels should be set for the read/write followed by the channel pins and then the data lines, it finishes by pulling the chip select pin low for 50ns.

(a) Back view of the developed printed circuit boards



(b) Front view of the developed printed circuit boards



(c) The MPD-001 driver board with the PCB attached. The Teensy3.6 can be seen mounted on the PCB.

Figure 2.3: Electronic interface used to control the polarization comprised of a MPD-001 driver board, the developed PCB and a microcontroller (Teensy 3.6).

To establish the communication between the microcontroller, the clock signal (responsible for the synchronization of the experiment), and the polarization controller, a printed circuit board (PCB) had to be created. The PCB was designed using using Circuit Maker from Altium. The PCB was made to accommodate two 22 positions male pin connector for the Teensy, 1 side-mounted 20-positions male connector for the polarization controller, and an 11mm SMA connector for the clock connection. The schematics and PCB layout can be found in Appendix A. Pictures of the PCB and driver board can be found in Fig.2.3.

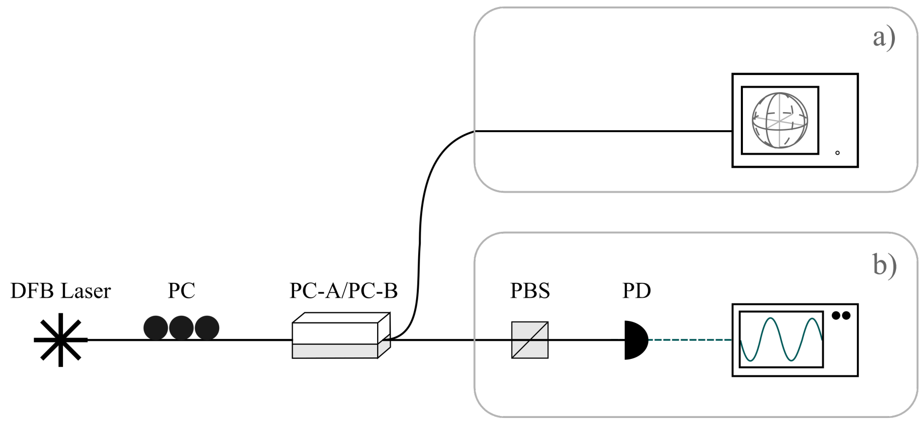


Figure 2.4: Setup used for the characterization of the polarization controllers. DFB laser - distributed feedback laser; PC - manual polarization controller PC-A/PC-B - piezoelectric polarization controllers; PBS - polarizing beam splitter; PD - photodiode.

2.2.2 Characterization

Setup

To characterize the polarization controllers the setup shown in Fig.2.4 was used. For the optical signal source we used a distributed feedback laser at 1550nm. The light then passes through a manual polarization controller before entering the piezoelectric polarization controllers. Then two variations of the setup were used. In *a)* a polarimeter was used to measure the angles θ and η of the polarization states. In setup *b)* a polarizing beam splitter (PBS) was placed after the polarization controller followed by a photodiode in order to measure the intesity of the ligth coming out of one the PBS' arm. The manual polarization controller allows us to align the polarization of the incoming photons with both the desired states in the Poincaré sphere seen in the polarimeter, and the transmission axis of the PBS.

Angle with Applied Voltage

Since the piezoelectic fiber squeezers are not perfectly aligned, the rotation they apply to polarization state does not correspond to what was expected in the specifications. Consequently, a characterization of the angle as a function of the applied voltage was done. For that, the input polarization was aligned with one of the rotation axes and then increments of 10 were added to the DAC values of an orthogonal channel. Note the voltage values written in the microcontroller are already set to match the range of the digital input. This was repeated for all channels of both polarization controllers.

With the aid of a polarimeter (Thorlabs - Profile PAT 9000 Polarimeter & Polarizer) the polar angle and the azimuthal angle (θ,η) of each polarization were saved (refer to Fig.2.1).

The readings of the polarimeter were written into a file with the help of a Python program. The results were plotted and processed using the computing environment, MATLAB®.

To obtain the function that correlates the displacement angle to the applied voltage, the measured (θ,η) pairs were converted into vectors with origin in the center and end on the surface of the Poincaré sphere.

Another important aspect to consider is the fluctuation of the polarization state measurement. This results in slight variations of the angles measured by the polarimeter. The polarimeter will then output an average of these different angles for each polarization state. Consequently, the pairs of angles measured will not create a perfect circumference in turn of the rotation angle. It will look more like a zig-zag along the surface of the sphere. Therefore, one cannot measure the distance between two consecutive vectors. The angle given will be larger than the actual rotation angle. The solution is then to project these vectors to the same plane and only then measure the angle on the projected plane.

Firstly, we should find the rotation angle needed to project the circumference formed by the vectors into z=0, the equator of the sphere. This is done by iterating over all the possible values of θ and calculating for which ones the elevation is minimal.

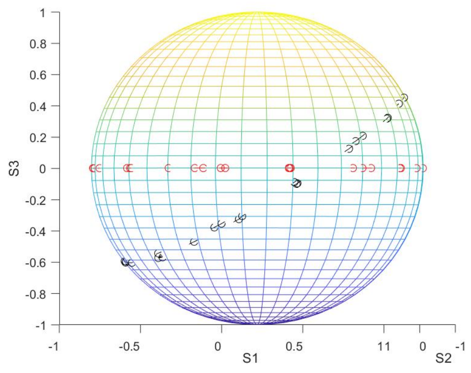
$$Rx(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \qquad Ry(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

$$Rz(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1) \end{bmatrix}$$

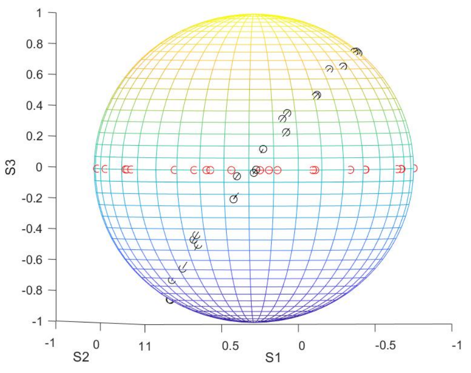
After finding the optimal rotation angles, it is now time to apply the rotation matrices Rx, Ry, and Rz to each of the vectors. Once the set of vectors is rotated, we calculate the projection of each of them onto the equator of the sphere. In figure 2.5 can be seen the polarization rotation before and after corrections.

Now we are only left to find the relationship between the applied voltage and the rotation angle. The vector for zero applied voltage was used as the reference vector. The angle between all other vectors and this reference vector was then calculated. The outcome relation between angle and applied voltage was used to perform the curve fitting to create the characterization function of the channels.

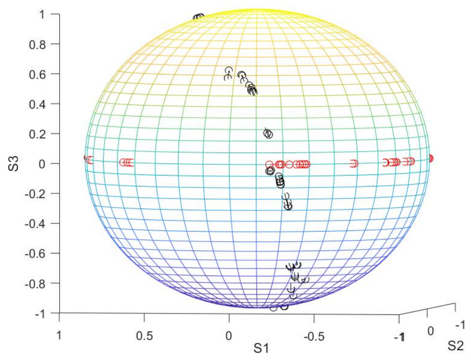
Now we are only left to find the relationship between the applied voltage and the rotation



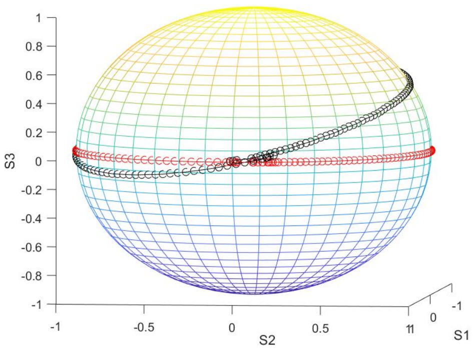
(a) PC-A: Channel 0



(b) PC-A: Channel 1



(c) PC-B: Channel 0



(d) PC-B: Channel 1

Figure 2.5: The states before and after the correction are represented in black and red respectively.

angle.

The vector for zero applied voltage was used as the reference vector. The angle between all other vectors and this reference vector was then calculated. The outcome relation between angle and applied voltage was used to perform the curve fitting to create the characterization function of the channels. An example of curve fitting can be seen in Fig.2.6.

Speed with Applied Voltage

The fiber squeezers are a mechanical component of the polarization controller and thus do not have an instantaneous effect. As a result, the polarization will spend time in an intermediate state between two desired polarization states. All the photons passing through the polarization controller during this rise and fall time cannot be used in the experiment since they will be prepared with an improper state. As a result, the length of this process is what is going to determine the maximum speed of the state preparation.

The microcontroller was programmed so that, using one channel, two states were prepared

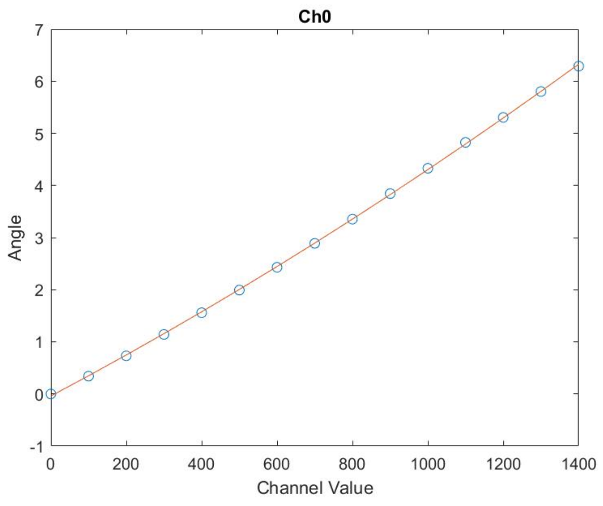


Figure 2.6: Plot of angle shift as a function of applied voltage for channel 0 of PC-A, with the corresponding curve fitting.

alternately, separated by a rotation of π and 2π. The voltage values needed to perform these rotations were taken from the *angleXvoltage* function determined in the previous step. For this characterization setup *b)* of Fig.3.1 was used. To measure this feature, the output of the polarization controller was connected to a PBS that in turn was connected to a photodiode (Thorlabs, DET01CFC - InGaAs FC/PC-Coupled Photodetector). The electrical signal of the detector was viewed in an oscilloscope (OWON SDS6062). A PBS divides the incident light into two polarized beams orthogonal to each other making use of a phenomenon known as polarization splitting. The portion of incident light with the same polarization as the transmission axis of the polarizer will be transmitted while its orthogonal counterpart is reflected. The transmitted and reflected light is, as a result, polarized to a very high degree. This PBS is thus used to study the change of polarization of the incoming light.

For both polarization controllers, the rise and fall time measured was of 325μs for a rotation of the polarization states of $\frac{3}{2}\pi$, which corresponds to the maximum displacement performed in the experiment. The results can be seen in Fig.2.7. This is in accordance with the specification sheet that state the rise and fall time for a displacement angle of π (10% to 90% transition), has an upper bound of 400μs. From this measurements we can infer the maximum frequency for the state preparation is 1.4kHz.

Stability

Next, we proceeded to analyse the overall polarization drift of all the fibered components. For that, the setup in Fig.2.4 *a)* was used, the polarization controllers were left preparing 4 random states overnight for a period of 7 hours. The prepared states were recorded by a polarimeter and then ploted using Matlab. Three different plots were made. In one the

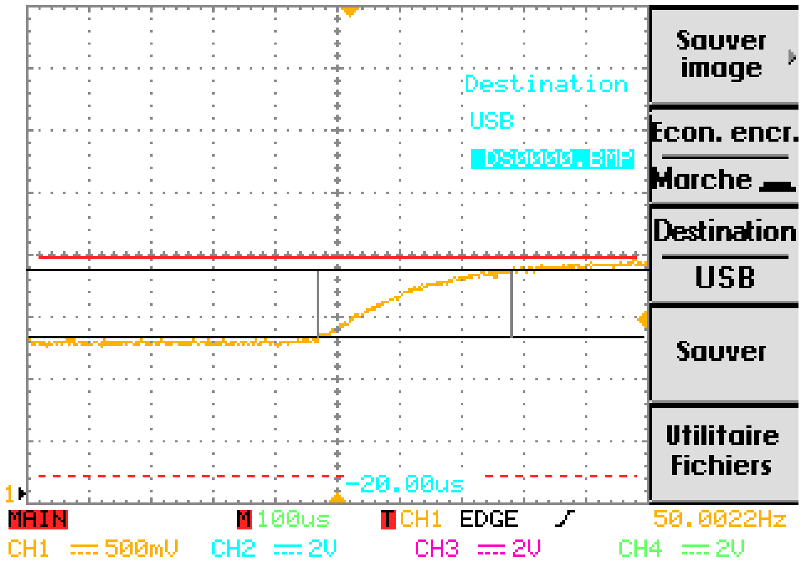


Figure 2.7: Rise time of a shift in polarization of $\frac{3}{2}\pi$ using only one piezoelectric fiber squeezer.

polarization states were represented in the Poincaré sphere, Fig.2.8a. In Fig.2.8b η and θ were ploted in function of time.

The drifts seen at the beginning of the measurement can be attributed to the effect on the optical fibers of temperature shifts in the laboratory, due to improper lab temperature management (AC or people going around). Looking only at data collected later in the night, the polarization is considerably more stable.

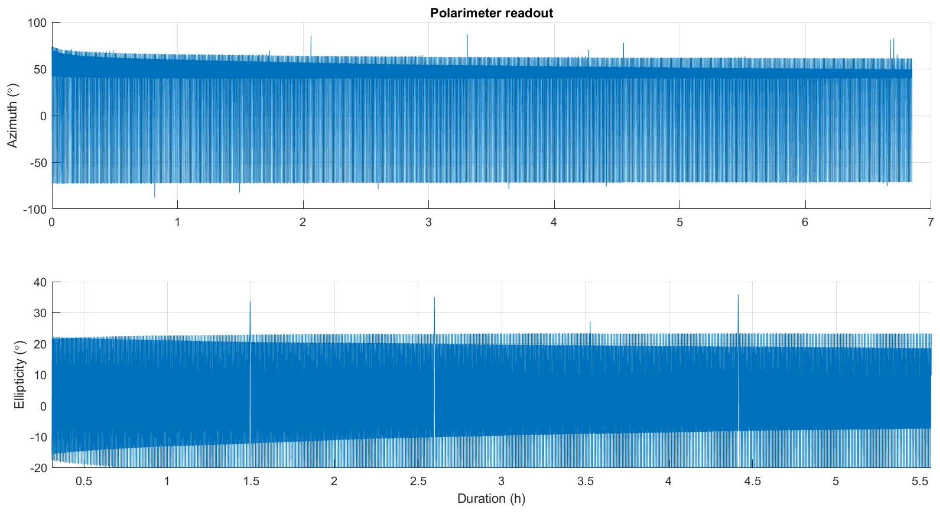
2.3 Finding the Four States

For this protocol, four states placed in the same plane with an elevation of $\frac{1}{4}\pi$ had to be prepared. With the help of the *angleXvoltage* function that we previously calculated, the voltage values for π , $\frac{1}{2}\pi$, $\frac{3}{2}\pi$ for channel 0 and $\frac{1}{4}\pi$ for channel 1 of each polarization controller were obtained.

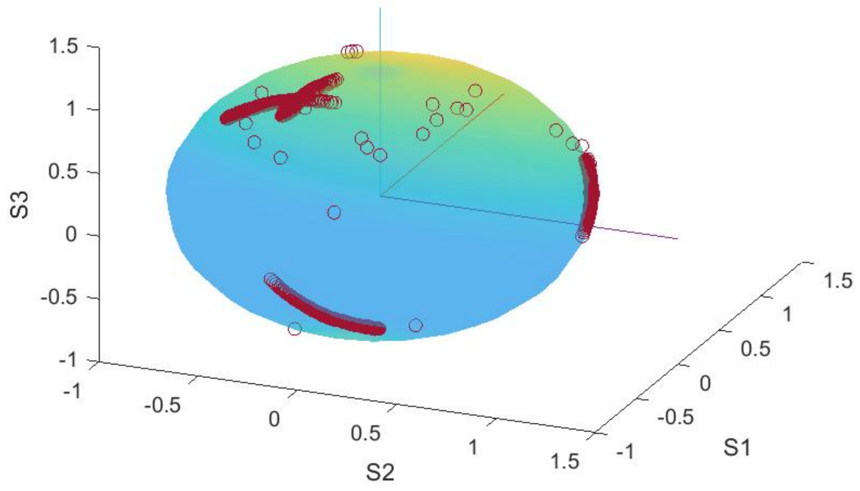
2.3.1 Test

To test the quality of the voltages obtained, we used the same setup shown in Fig.2.4 *b)*, composed of a telecom laser, a manual polarization controller, and one of the piezoelectric polarization controllers (PC-A or PC-B), followed by a PBS and a photodetector (PD). The signal of the PD is viewed in a digital oscilloscope. The test is performed in one polarization controller at a time.

To prepare the 4 desired states, we align the polarization controller input polarization with the rotation axis of channel 0. Then we apply a voltage on channel 1 to "rise" the



(a) Angles characterizing the four polarization states over time.



(b) Four random polarization states ploted in the Poincaré sphere.

Figure 2.8: Stability of state preparation over a period of seven hours.

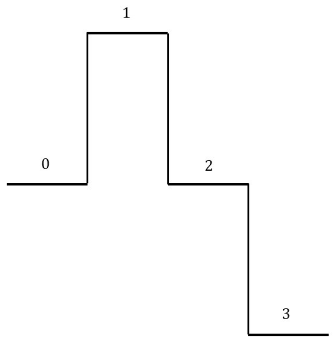
polarization to $\frac{1}{4}\pi$. After that, we turn around channel 0 by applying the voltages we calculated for π , $\frac{1}{2}\pi$, $\frac{3}{2}\pi$, 2π. Table 2.1 shows the voltage values corresponding to each state in both polarization controllers.

We prepare the four polarization states using the voltage combinations from Table 2.1 at a rate of 100Hz. Using the manual polarization controller, we rotate the incoming polarization to align one of the prepared states, with the transmission axis of the PBS output connected to the oscilloscope. Consequently, the state aligned with the PBS will

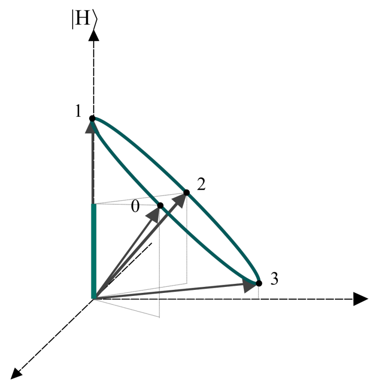
	PC-A				PC-B			
State	CH0	CH1	CH2	CH3	CH0	CH1	CH2	CH3
0	170	0	0	0	183	0	0	0
1	170	454	0	0	183	485	0	0
2	170	900	0	0	183	850	0	0
3	170	1286	0	0	183	1270	0	0

Table 2.1: DAC voltage values to prepare each of the four sates for both polarization controllers.

display the highest intensity in the oscilloscope and the other states will display as steps of lower intensities.



(a) The four polarization states seen in the oscilloscope.



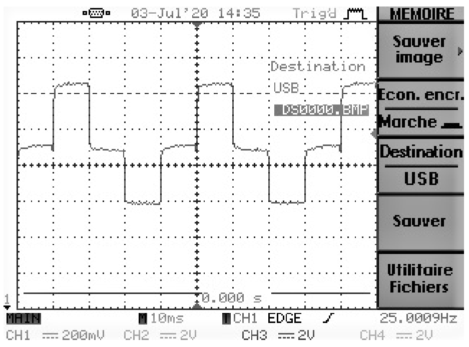
(b) The four polarization states in the Poincaré sphere.

Figure 2.9: The four polarization states. If the polarization of state 1 is aligned with the polarization of the transmission axis of the PBS, then state 1 will have a full projection onto the axis. States 0 and 2, equidistant from state 1, will have an equal projection onto the axis, and state3 will have the smallest projection.

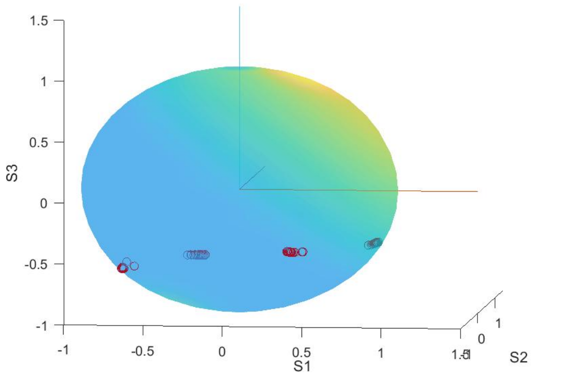
The results are shown in Fig.2.10a. As predicted, the same pattern can be seen on both polarization controllers.

Using the setup of Fig.2.4 *a)*, by connecting the output of the polarization controller directly to the polarimeter we saw the polarization of the states being prepared in the Poincaré sphere. The results were ploted using MATLAB® and they can be seen in Fig.2.10b. With the data from both Fig.2.10a and Fig.2.10b we conclude the states prepared are appropriate to be used in the experiment.

We took this opportunity to also re-test the maximum speed of the state preparation. For this we used the setup shown if Fig.2.4 *b)*. After aligning the states, we gradually increased the frequency of the state preparation until the states seen in an oscilloscope lost their rectangular shape. When the signal approaches the shape of a sinusoidal function,



(a) The four polarization states seen in the oscilloscope, being prepared at 100Hz.



(b) The four polarization states in the Poincaré sphere.

Figure 2.10: The four polarization states prepared at 100Hz

the desired states can no longer be prepared. This threshold marks the speed limit. Here we confirmed once again that the maximum allowed speed was around 1.4kHz as seen in Fig.2.11.

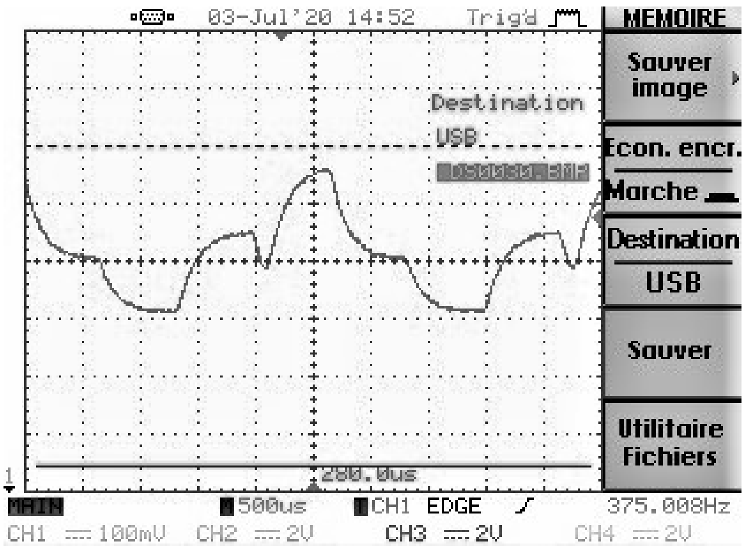


Figure 2.11: The four polarization states being prepared at 1.5kHz.

2.3.2 Optimization

The higher the voltage difference between the two states, the longer the rise and fall time will be. The method previously mentioned only uses two channels. As a result, greater voltage amplitudes are needed in order to achieve the same polarization shift than using the four channels would achieve. To reduce this effect, a different method of preparing the 4 states was needed. However, as the pairs of channels are not perfectly aligned, it is not possible to predict the movement the polarization state would make in the Poincaré sphere when the voltage was applied on the 4 channels simultaneously.

To overcome this issue we performed an optimization to find the set of voltages needed in order to prepare a desired polarization state using the four channels. The four states chosen as the optimal states corresponded to the angle pairs (0,15); (-45°,15°); (45°,15°); (90°,15°) with (θ,η). Every optimization sequence runs and optimizes for one of the four states at a time. The outcome of every run of the optimization program is a DAC value that represents the voltages applied to the piezoelectric squeezers, needed to prepare one of the chosen polarization states. This process is repeated until the voltages for all preparation states are obtained, and for both polarization controllers.

The polarimeter outputs are read and processed with the help of a python program. The python program also sends and receives from the microcontroller (Teensy) the voltages that should and are being applied to each channel. The optimization function was written with the computing environment MATLAB® R2018b and is called by the Python program.

A manual polarization controller was placed before the piezoelectric polarization controller to align the incoming light polarization to $|L\rangle$(0°,0°)). The voltage applied in channel 1 is gradually increased until the rotation of 2π is achieved. The polarimeter readout and corresponding applied voltage are saved into a 30x2 matrix. The readings of the polarimeter are converted into a vector of the Poincaré sphere. After that, with the help of the optimization function, the angle between the optimal state, and every vector of the matrix is calculated.

The voltage with the smallest corresponding angle in then sent to the microcontroller. The microcontroller, in turn, writes this voltage value to the polarization controller as the voltage of the current channel. Afterward, the previous process is repeated for the next piezoelectric squeezer. This loop continues until it converges to a state separated from the optimal state by an angle smaller than 0.1°.

The optimization code can be found in Appendix B. The corresponding block diagram can be seen in Fig.2.12.

One of the polarization controllers, PC-A, converged to the voltage configuration shown in Table 2.2. However, PC-B did not converge. Since this optimization task lasts for over 7 hours, drifts in the input polarization are experienced, limiting the accuracy of he optimized polarization states. They can be a result of varying temperature and moving optical fibers.

Nonetheless, we tested the speed of the state preparation for the obtained voltage configuration for PC-A. The same method was used as in 2.3.1. For this we measured a rise time of 150μs (see Fig.2.13), about 2.3 times faster than the preparation using only two

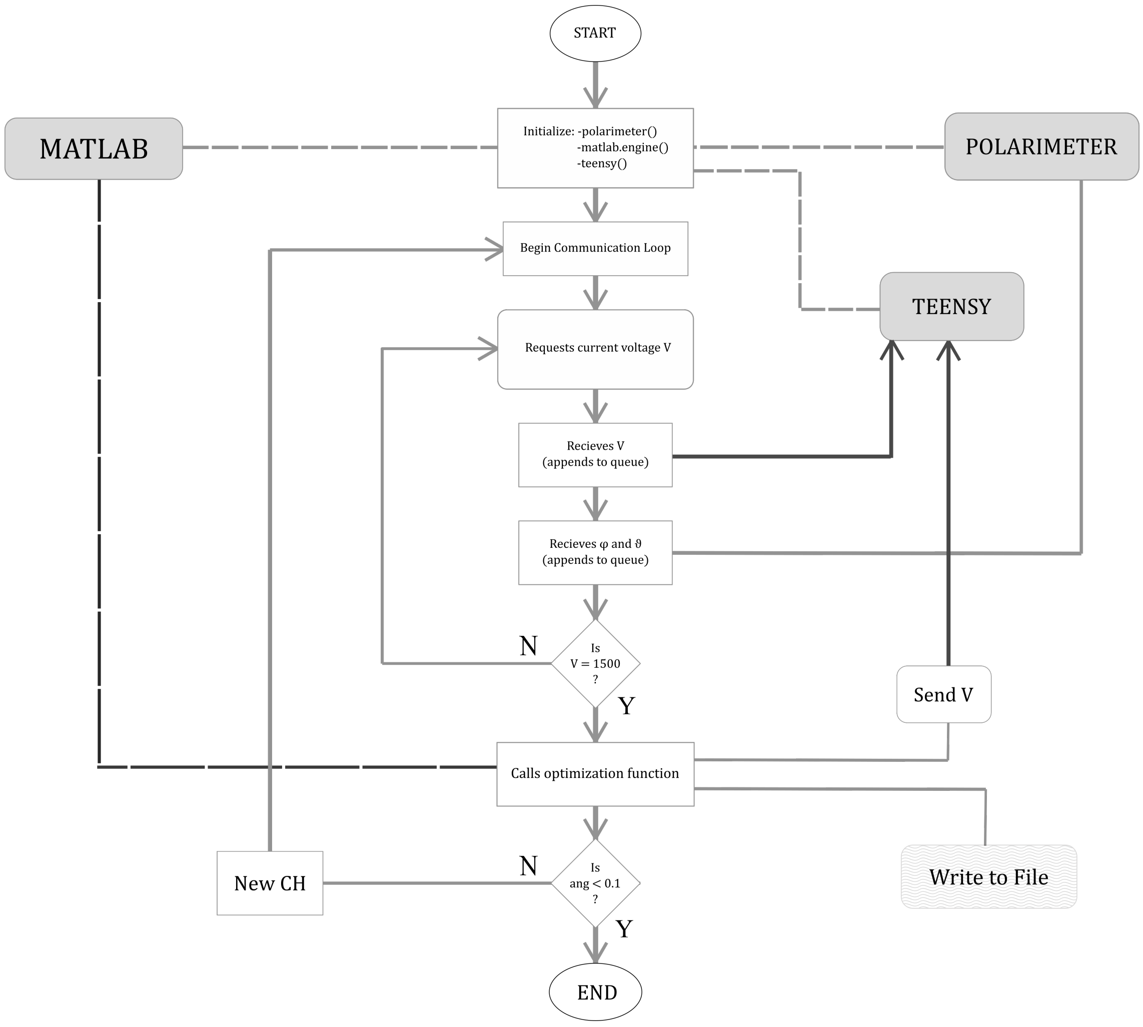


Figure 2.12: Block diagram of the optimization.

piezoelectric channels. As a result, the maximum frequency at what the experiment could be ran is 6.6kHz.

State	CH0	CH1	CH2	CH3
0	950	600	1200	1250
1	50	150	150	150
2	150	350	1450	350
3	150	750	600	600

Table 2.2: Optimized DAC voltage values to prepare each of the four sates, for polarization controller PC-A.

2.4 Conclusion

In this part of the thesis project the systems needed to control the polarization controllers were built. With them we mounted the optical setup needed to characterize the state

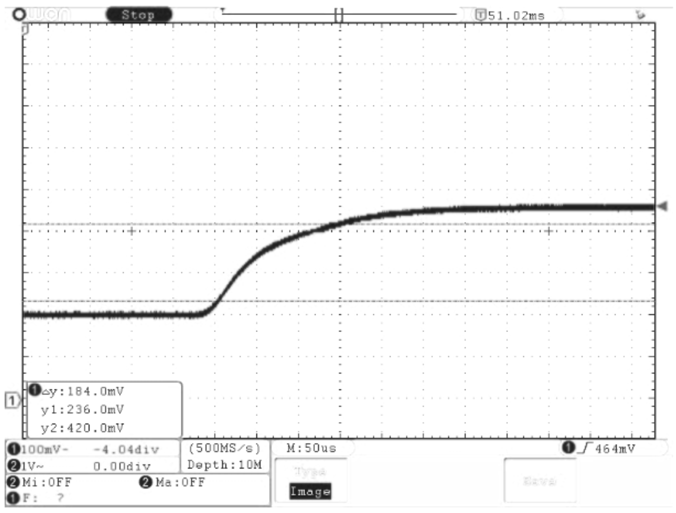


Figure 2.13: Rise time of the transition between two opposite states.

preparation. The polarization controllers were characterized in terms of their speed and the relationship voltage vs. angle of each one of their channels. With the polarization controllers characterized, we then calculated the parameters needed to prepare the four polarization states required by the protocol, as well as the maximal achievable speed of 1.4kHz.

In order to increase the speed of the state preparation, an optimization program was ran. This optimization proved to be successful in only one of the polarization controllers not allowing us to use these optimal voltage configurations in the end. In order to reduce the effects of polarization drifts, the speed of the polarization process would have to be increased, so that we could obtain the desired states. After testing this new technique we could increase the frequency of the state preparation up to 6.6kHz.

3

The Experiment

In the previous chapter we discussed the characterization of the polarization controllers and defined how the polarization states are going to be prepared. In Chapter 1 we have already presented a simplified layout of the experiment. In Chapter 2 we described how we built built the hardware needed to control the polarization controllers and characterized the state preparation. In Chapter 3, we will examine a more in-depth implementation of this experimental set-up, namely how the necessary alignments and characterization of coherent are performed and how it is controlled.

3.1 The Experimental Set-Up

The expanded experimental realization can be found in Fig.3.1. The optical signal is generated by a DFB laser at 1559nm on Alice's side (Mitsubishi, FU-68 PDF-5 - 1.58 μm (L-Band) DFB-LD Module With Polarization Maintaining Fiber Pigtail). The laser, being unable to maintain the lasing threshold when triggered at 1kHz, had to be triggered at 1MHz, while creating 90ps pulses. To remove the unwanted pulses, an Electro-optic modulator (EOM) was placed after the laser. It is used as an almplitude modulator that attenuates 999 pulses out of the 1000 created. The polarization encoding is done with the help of the voltage-driven piezoelectric polarization controllers (General Photonics PCD-M02) described in 2. The encoded pulses then go through a variable attenuator to reduce the mean photon number per pulse to the desired sum. The output of Alice is connected to the input of Bob.

Bob makes an active choice for the measurement basis by preparing one of the four available polarization states with the help of a polarization controller (General Photonics PCD-M02). The pulses then pass through a polarizing beam splitter (PBS).

The detector used for all the preliminary tests is an InGaAs avalanche photodiode (ID Quantique id210) set with 10% efficiency and 25μs deadtime and dark count rate of 2kHz. The detector used in runs of the experiment is a superconducting nano-wire single photon detector (SNSPD) from IDQuantique with 90% efficiency and dark count rate of 200Hz.

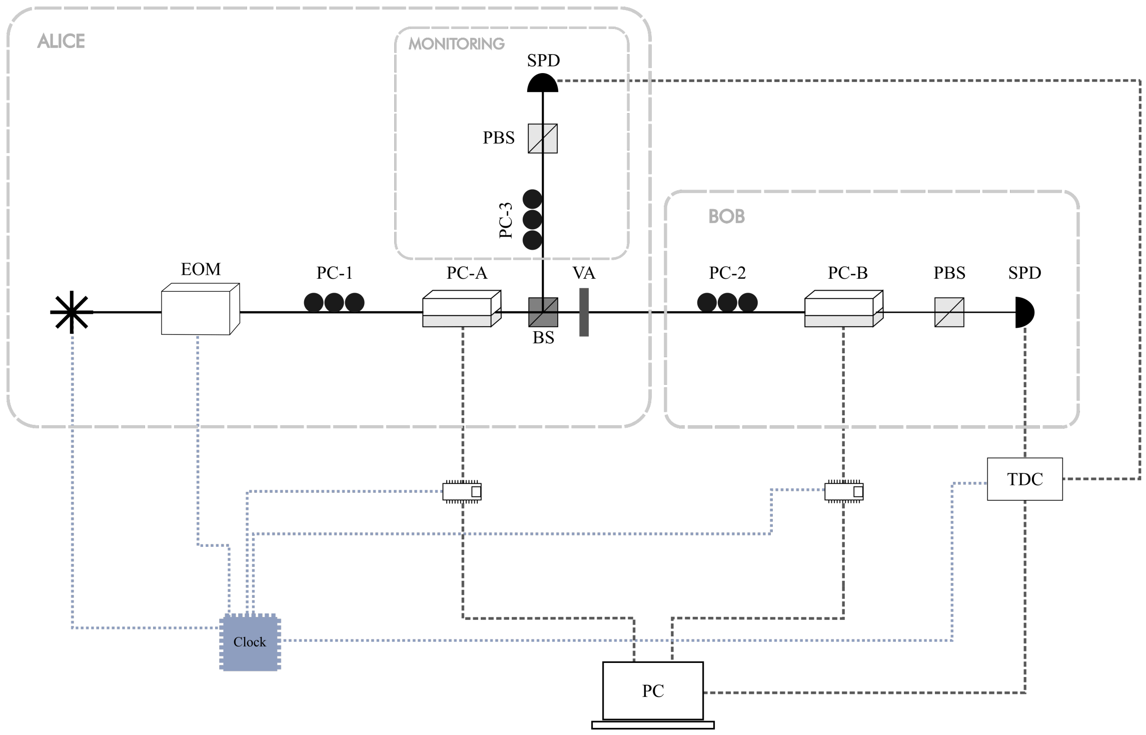


Figure 3.1: The experimental set-up. EOM: Electro Optical Modulator; PC-1: Manual polarization controller 1; PC-2: Manual polarization controller 2; PC-A: Alice's polarization controller; PC-B: Bob's polarization controller; VA: Variable attenuator; PBS: Polarizing beam-splitter; SPD: Single photon detector; TDC - Time-to-digital converter; PC - Personal Computer.

The fibers were spliced from the output of Alice's setup to the detector's input to reduce losses in the channel. The quantum channel has an overall attenuation of -0.45dB (90% transmission) from the output of Alice to the detector input. The detector is connected to a Time-to-digital converter (id800-TDC 8-channel time-to-digital converter). As the name entails, the TDC converts the events caught by the detector into a digital representation of their timestamp (time recorded in 81ps bins). The time stamps are thus forwarded to the PC and read by a Python program.

To monitor the stability of the prepared states, on the output of Alice's polarization controller, is placed a 50/50 beamsplitter that will veer 50% of PC-A's output beam into the monitoring setup. This monitoring system is comprised of a manual polarization controller, a polarizing beamsplitter and a single photon detector connected to the TDC. The detector used is an InGaAs avalanche photodiode (ID Quantique id210) set with 10% efficiency and 25μs deadtime and dark count rate of 2kHz.

PC-3 is used to uniformly align all the incoming polarized states to the PBS's transmission axis so that only the phase varies between the four preparations. As a result, if an oscilloscope is placed after the PBS, we should be able to see a flat line when the four states are being prepared. This results in a constant photon flux to be detected at the

SPD. Any fluctuations in the count rate will thus indicate a misalignment of the states being prepared.

Note that all time driven components of the set-up are synchronized with an external clock (Silicon Labs, Clock & Timer Development Tools SI5341-D-EVB).

3.2 Control

The experiment is controlled by a Python program, developed previously by the group for past QKD experiments and adapted here. It is responsible for generating the states used in the experiment, for the communication with the microcontrollers of Alice and Bob, and for receiving and analyzing the data received from the detector. Among these features, the Python Gui also allows the user to synchronize and align the states used in the experiment and respective measurements.

Since we were unable to conclude the optimization described in 2.3.2 for PC-B, the set of voltage values used in experiment will be the ones seen in Table 2.1. By using this method, a $\frac{3}{2}\pi$ rotation is performed with channel 1. We opted to implement the experiment at a rate of 1kHz, to ensure the photon pulses were prepared in the desired polarization states. The bigger the difference between the transition times and the time where the polarization is flat, the easier it will be to align the photon pulses with the respective polarization state. Also, potential delays in the software will not cause critical misalignments.

3.2.1 Microcontrollers

Alice and Bob's polarization controllers are controlled by one microcontroller, Teensy 3.6, each. The Teensy is responsible for sending the preparation states and the measurement basis to Alice and Bob at a rate of 1KHz. The Teensys are connected to the clock through an interrupt pin, and every 1ms, receive a trigger to prepare a state. Each teensy has a buffer (*statesBuffer*) of 200 polarization states following a FIFO (first-in, first-out) scheduling. When in need, each Teensy asks the computer for a new queue of 100 states, and recieves them through Serial communication.

The Teensy uses an interrupt function to ensure adequate timing of the state preparation. When the Teensy recieves a trigger from the clock through the digital interrupt pin, the interrupt flag is set to TRUE and the Teensy enters the state preparation loop. In this loop, the Teensy, with the help of a counter, runs through the statesBuffer and extracts the state to be prepared, deleting it from the buffer. Once the state is known, the corresponding DAC values for each channel can be obtained. Then, iterating over the four channels, the binary value of the DAC voltage is sent sequentially to the polarization controller. Whenever the counter is 0 or 100 (every 100ms) Alice and Bob send a request to the

Python program to receive more states. If the counter is at 0 then the new 100 state queue is put on the second half of the buffer. Otherwise, the states are written to the first half. The new states queue is received as a Serial communication from the computer.

The Teensy can also receive new voltage values to set as the new voltages of a state. It can also receive a message to request emptying the buffer. The full code can be found in Appendix C.

3.2.2 Data Collection

The states that are sent to the microcontrollers are saved in the python memory in two queues (one for Alice and one for Bob) to later be used for the key sifting.

The Python program collects the time stamps by the TDC and shows them in a histogram on the display window while the code is running, updated every 5 seconds. The collected timestamps are converted into the format *an bn dn*, where n ranges from 0 to 3. This data is then saved to a file every 5 second. The file log can be read as:

- an – Alice prepares state n
- bn - Bob measures in basis n
- d0 - detector 1 clicks
- d1 - detector 2 clicks
- d2 - coincidence detection (both detectors click)
- d3 - no detector clicks

Giving origin to a matrix:

$$M = \begin{array}{c} \\ b0 \\ b1 \\ b2 \\ b3 \end{array} \begin{array}{c} \begin{array}{cccc} a0 & a1 & a2 & a3 \end{array} \\ \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} \end{array}$$

Where x is the number of detections on detector 1 when state *an* is prepared and *bn* is the measurement basis. Note that the detections in detector 1 are the ones that will be used for the key sifting.

The system is aligned so that, when Alice and Bob prepare and measure coincidentally on the same basis, one arm of the PBS will have no light. Following the security analysis of the protocol, we are interested in collecting data that comes from the "0%" arm as we may call it. Hence, this is the only output connected to the detector. We only used the values from d0 and d3 for the data analysis for the key. The values from the second detector will be used to monitor the stability of the state preparation.

One experimental challenge we faced was having Alice and Bob's polarization controllers preparing the correct state pairs. If we assume two sequences of n random states, such that $(A_i;B_j)\ \forall i,j = \{0;n\}$ is the pair of polarization states being prepared simultaneously, it is indispensable that $i = j$. This stems from the fact that the state combination used for the key sifting is generated in the software and extracts elements with the same index from the respective state preparation queues.

To correct for that, the user can also add/remove states and detections from their respective sifting queue. The display window also allows the user to choose the sequences sent to Alice and Bob between a list of fixed pre-made sequences and a random sequence. In this interface, the user is also able to choose the detection window, in number of time bases, and set the delay of the detector, also in time bases. On the display window, it also shows the normalized sum of the diagonal of matrix seen above divided by the normalized sum of the non-diagonal elements of the matrix 3.1. This will later be used for aligning the detections in time and shall be discussed later. An image of the display window can be found in 3.2.

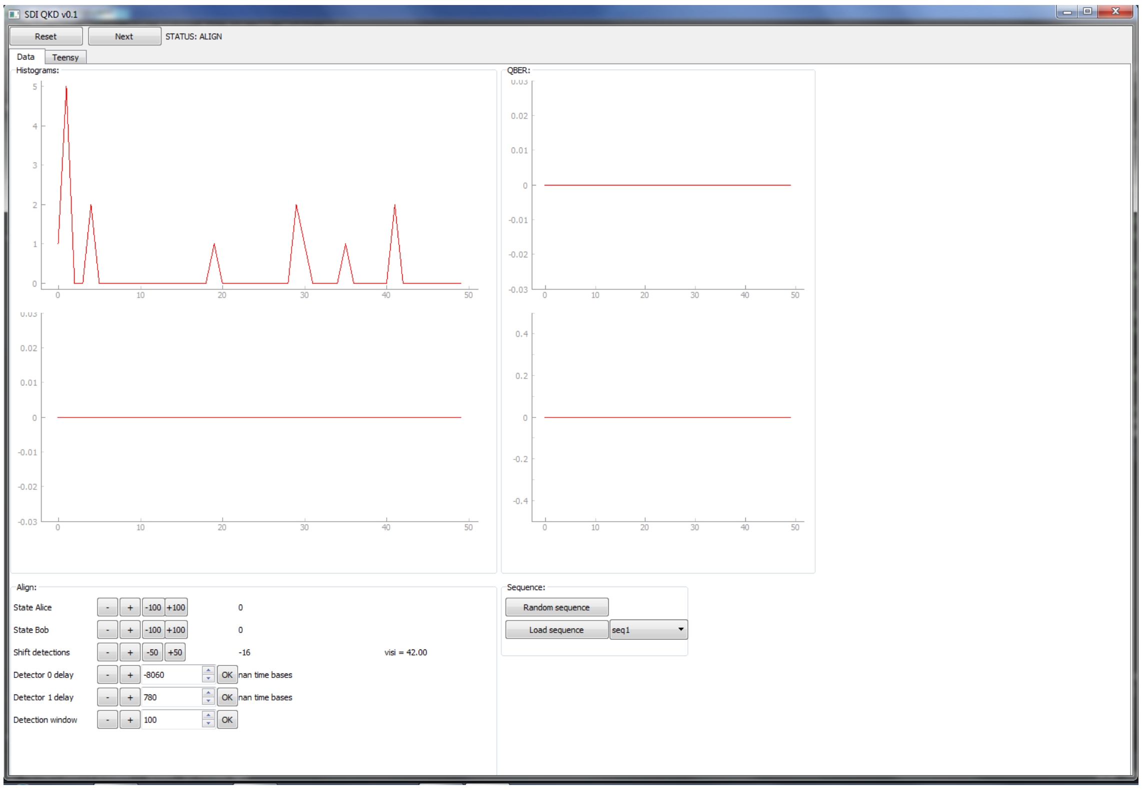
$$V = \frac{\frac{1}{4}\sum_{(n=m):0}^{3} an.bm}{\frac{1}{12}\sum_{(n\neq m):0}^{3} an.bm} \tag{3.1}$$

3.3 Alignment

To correct the alignment in time, with the Python code running, we used the experimental setup seen in Fig.3.3 and repetitive fixed sequences of Alice and Bob's states. The output of the PBS can be monitored either with an APD, ploted and viewed in a histogram with the help of quTAU's user's interface Fig.3.3 *a)*, or, with PD and viewed in an oscilloscope Fig.3.3 *b)*.

3.3.1 Time

When the Python program starts and the serial communication with the microcontrollers is established, the pointer of the state's buffer is set to 0 on both microcontrollers. However, this restart is done sequentially, first in PC-A and then on PC-B, which may result in the pointers not being coincidental. This can in turn result in a misalignment in time of the



STATUS: ALIGN

Align:

State Alice 0

State Bob 0

Shift detections -16 visi = 42.00

Detector 0 delay -8060 nan time bases

Detector 1 delay 780 nan time bases

Detection window 100

Sequence: seq1

Figure 3.2: Display window of the Python code used in this experiment.

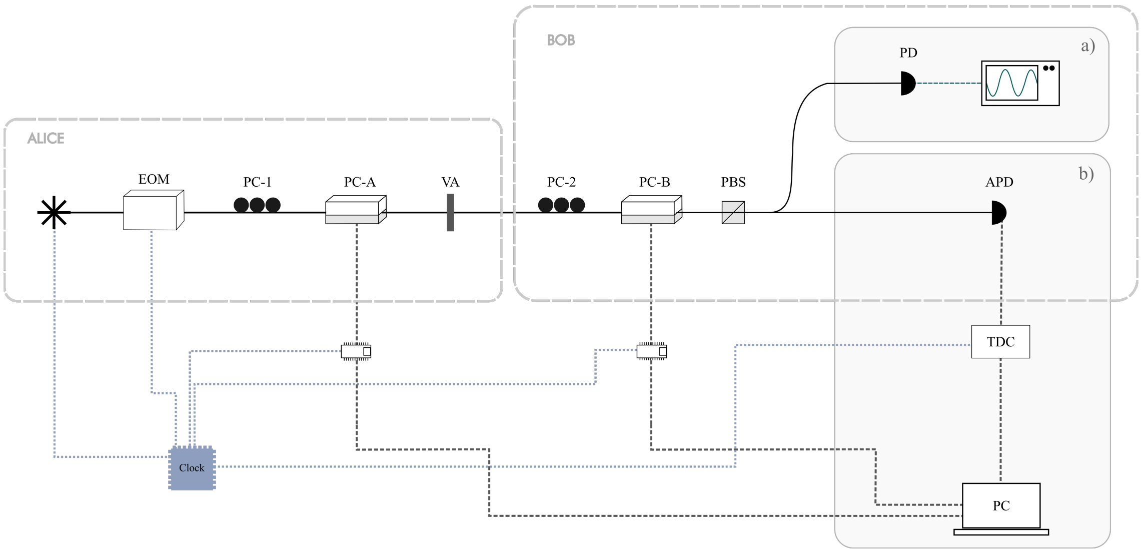


Figure 3.3: Setup used for aligning the state preparation. EOM - Electro-optic modulator; PC-1/PC-2 - manual polarization controller PC-A/PC-B - piezoelectric polarization controllers; VA - variable attenuator; PBS - polarizing beam splitter; PD- photodiode; APD - Avalanch photodector; TDC - Time-to-digital converter; PC - Personal Computer.

state preparation. The separation between pulses will depend on how close in time the initialization of both microcontrollers is done.

To correct this, a *remove_state/add_state* function was needed in the Python software. This function allows the removal of states from Alice's and Bob's state queues that are being sent to their Teensy. When a state is removed it is also removed from the sifting queue.

With the Python code running, a sequence is uploaded to both Alice and Bob. It is comprised of ninety-nine state0, and one state1. In the oscilloscope, we will be able to see one pulse if the state preparation is aligned in time and two if it is misaligned.

3.3.2 Polarization

Once we are sure the states are aligned in time, it is important to make sure the polarization states prepared by both Alice and Bob are the same i.e. they are defined by the same ellipticity and elevation (θ,η) in the Poincaré sphere. To check for this, we used the same setup described in Fig.3.3 *a)*. We send to Alice and Bob a sequence of 100 states following the order 0123. As a result, Bob should always measure in the same basis that Alice prepared her states in. We see this alignment either as a signal of 0V in the oscilloscope, meaning we are looking at the PBS's output orthogonal to the axis of PBS's polarizer, corresponding to 0% transmission. Alternatively, the value measured by the PD can take its maximum value, meaning the prepared states are aligned with the polarizer's axis, resulting in 100% transmission.

With the help of the manual polarization controllers, we then align the polarization of the incoming light so that the steps seen in the output signal are no longer visible and the signal is a straight line. This means both polarization controllers are correctly preparing the same polarization state.

After making sure the preparation states and the measurement basis are in agreement, it is important to also analyze the relationship between different states. The state sequences sent to Bob and Alice are the following:

A = 0000111122223333(...)

B = 0123012301230123(...)

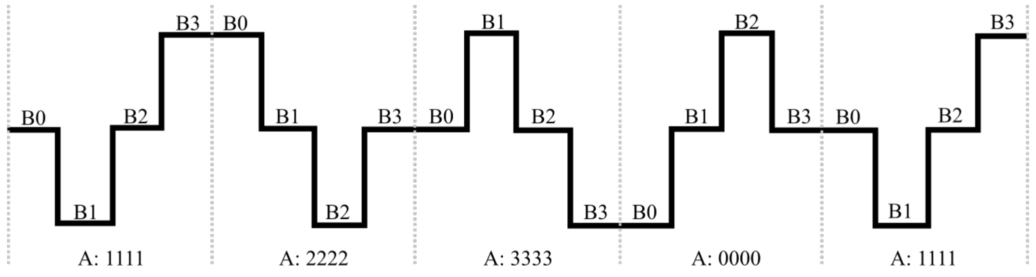


Figure 3.4: Relationship between all prepared states.

If the state preparation of both polarization controllers is aligned in both time and polarization, in the oscilloscope, looking at the output where the alignment results in 0% transmission, we would, in theory, see the sequence in Fig.3.4.

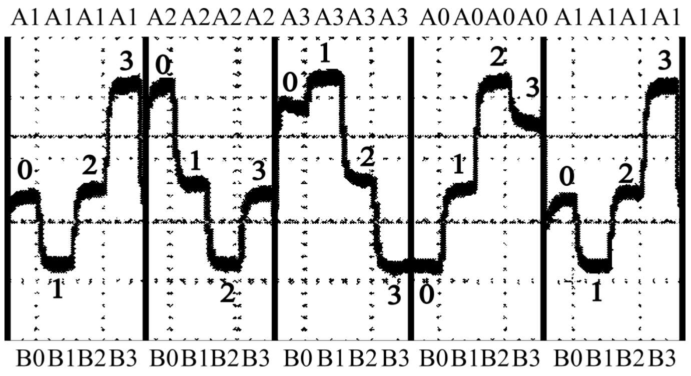


Figure 3.5: Relationship between all prepared states displayed in the oscilloscope.

The oscilloscope display is shown in Fig.3.5. All the combinations were distinguishable. From this, we can confirm the relationship between the prepared states and the measurement basis is correct.

3.3.3 Detections

Another misalignment to account for was detections with prepared states. The Python script starts receiving detections from the TDC's internal buffer, before the microcontrollers start to in fact prepare the states. The received detections are saved in a queue that later will be used for the key sifting. The same happens with the states prepared by Alice and the measurement basis chosen by Bob. When the computer is initializing all the controllers and models at the beginning of every run, it is receiving data from TDC. However, during this time, the serial connection with the microcontrollers is blocked. As a result, the sifting queues for Alice and Bob's states will not be aligned with the detection sifting queue. To correct this, a *remove_detection* function is needed. This function removes n detections from the detection queue, with n an arbitrary integer number chosen by the user.

To align the detections in time, we used the setup shown in 3.1 and made use of Python's user interface created for this experiment. As mention in section 3.2.2, the interface displays in real time (updated every 5 seconds) the normalized sum of the diagonal of matrix divided by the normalized sum of the non-diagonal elements of the matrix described by eq.3.1.

As it is described in section 3.2.2,we align Alice and Bob's state preparation and sent to both queues of randomly generated states. If we examine matrix M again with more detail, it is easy to realise that under these conditions, in the diagonal of the matrix *(a0b0 ; a1b1 ; a2b2 ; a3b3)* will never be recorded detections. Consequently V should be zero.

However, if the Alice and Bob's sifting queues are not aligned with the detection sifting queue, then V is close to unity. This is how we use V as a real-time indicator of the detections' alignment in time. Using the *remove_detections* function we then remove the necessary number of detections from the beginning of the sifting queue until V is zero.

3.4 Weak Coherent Pulses

Since perfect single photon sources are not widely available, we opted to use a weak coherent laser source that generates a pulse with probability of having n photons given by:

$$P(n) = \frac{\mu^n}{n!}\exp(-\mu)$$

Where μ is the average photon number per pulse. Thus, the smaller the μ the least information an eavesdropper will be able to extract from the communication [50]. Therefore we must ensure that the weak coherent states we prepare for this experiment are appropriate.

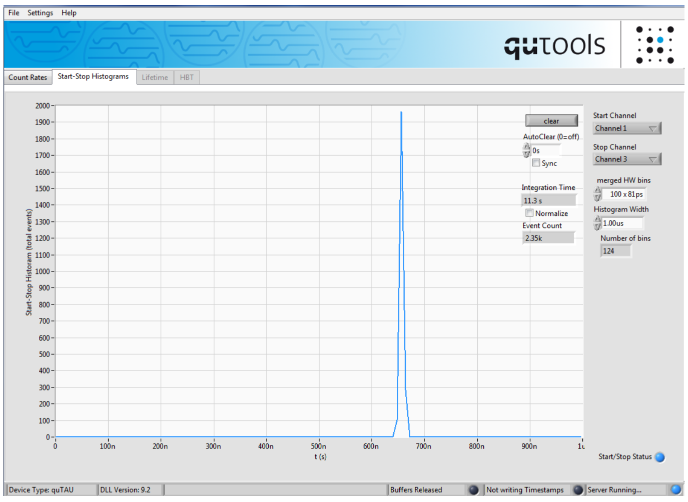
The characterization of the coherence pulses where the information is encoded in the experiment will be discussed in this section.

3.4.1 The Laser

The Power Meter (Thorlabs - Digital Optical Power and Energy Meter + Thorlabs - Fiber Photodiode Power Sensors S154C) used to measure the output power of Alice is not sensitive enough to measure the power when the EOM is on, i.e. when the laser pulses pass through at 1kHz. And depending on the frequency at which the EOM is triggered its attenuation may vary, that is why we cannot measure with the EOM ON triggered at 1MHz. Consequently, to measure the number of photons per pulse, μ, at the output of Alice's system, we needed to measure with the EOM OFF (1 MHz pulsed laser frequency) and then take into account the attenuation the EOM causes in the pulses that are not filtered out.

To measure the EOM's attenuation, the laser output was connected to the EOM which in turn was connected to an SPD (ID Quantique id210) linked to a TDC. With the help of the graphical user interface quTAU, we were able to view the detection's timestamps on the computer which were being plotted in a histogram (Fig.3.6). Choosing an integration time of 500s, we then measured the number of counts in the peak that corresponded to the photon pulses.

We did the same measurement but this time with the EOM ON, resulting in laser pulses with a 1KHz frequency. We measured once again for a period of 500s. Dark counts were also measured for both samplings, 1KHz, and 1MHz, with the optical fiber unplugged from the detector. We measured a transmission of 91%.



Start Channel: Channel 1
Stop Channel: Channel 3
AutoClear (0=off): 0s
Sync
merged HW bins: 100 x 81ps
Integration Time: 11.3 s
Normalize
Histogram Width: 1.00us
Event Count: 2.35k
Number of bins: 124
Start/Stop Status

Device Type: quTAU | DLL Version: 9.2 | Buffers Released | Not writing Timestamps | Server Running...

Figure 3.6: Display of quTAU's histogram interface showing the time histogram of the weak coherent state measured by the SPD.

3.4.2 Test

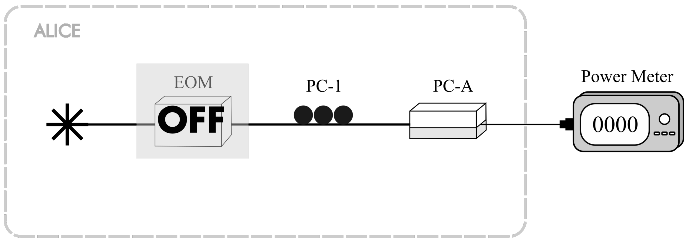


Figure 3.7: Setup used to measure the intensity of light leaving Alice when the EOM is off.

As mentioned previously, the mean photon number had to be 0.01 for optimal performance. To achieve this value, a variable attenuator must be used to reduce the number of photons in each pulse.

Firstly, we quickly measured μ at the output of Alice's setup with the power meter with the laser at 1MHz and the EOM OFF (Fig.3.7). As mentioned previously, with the EOM triggered at 1kHz the power is too low to be detected by the power meter. Without the attenuator it yields 80nW. Which, considering the EOM attenuation, corresponds to 5,671 $\times$ 10^5 photons per pulse. Note that the average mean photon number μ is given by:

$$\mu = \frac{\frac{P}{N_{pulses}}}{E_P}$$

so that $E_P = \frac{hc}{\lambda}$ is the photon energy , P is the beam power and N_{pulses} is the number of pulses per time unit

We want to attenuate to 0.01 photons per pulse as a result, an attenuation of -77dB is needed. For the attenuation defined as:

$$atenuation = 10.log_{10}\left(\frac{\mu_{out}}{\mu_{in}}\right)$$

We must then confirm that the attenuation calculated is also valid for a repetition rate of 1KHz. For that, we used once again the SPD and placed it at the output of Alice's setup (Fig.3.8). It is important to mention that the free running detector used has a dead time of 25 μs and 10% efficiency.

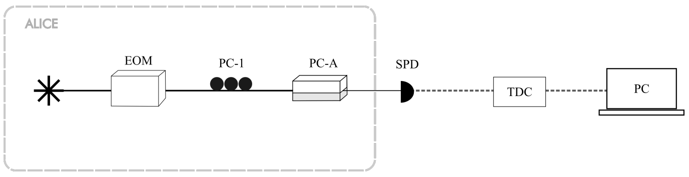


Figure 3.8: Setup used to measure the number of photons per pulse.

Note that dead time is defined as the time after the detection of one event, in which the system is not able to perform any other detection. As a result, we must consider the dead-time correction as well as dark-counts when calculating the number of photons that reach the detector. The event rate given by the detector corrected for dead time minus the dark counts should yield the number of photons arriving at the detector per second. So, if N is the number of photons that reach the detector, d is the number of detections, Td is the dead-time and DC the dark counts, then, we have an N of:

$$N = \frac{d}{1 - dxT_d} - DC$$

The detector was connected to the TDC and the user interface provided by quTAU (3.9) was once again used to collect the timestamps of the detections and show the photon counts over an integration time of 500s. We measured both the detection rate of DC and photons. The results can be found in below:

$$d = 17.9k \;\; photons \quad (500s)$$

$$dc = 15.9k \;\; photons \quad (500s)$$

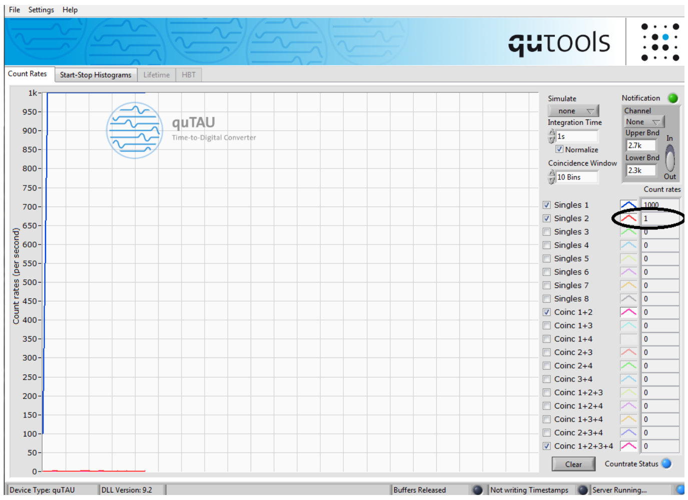
$$Td = 25\mu s$$

$$N = 1.201 \;\; photons/s \qquad (10\% \;\; efficiency)$$

$$N = 12.01 \;\; photons/s \qquad (100\% \;\; efficiency)$$

Measuring

$$\mu = \frac{12.01}{1000} = 0.0120 \;\; photons/pulse$$



Simulate: none
Integration Time: 1s
[✓] Normalize
Coincidence Window: 10 Bins
Notification
Channel: None
Upper Bnd: 2.7k
Lower Bnd: 2.3k
In / Out

	Count rates
[✓] Singles 1	1000
[✓] Singles 2	1
Singles 3	0
Singles 4	0
Singles 5	0
Singles 6	0
Singles 7	0
Singles 8	0
[✓] Coinc 1+2	0
Coinc 1+3	0
Coinc 1+4	0
Coinc 2+3	0
Coinc 2+4	0
Coinc 3+4	0
Coinc 1+2+3	0
Coinc 1+2+4	0
Coinc 1+3+4	0
Coinc 2+3+4	0
[✓] Coinc 1+2+3+4	0

Countrate Status

Device Type: quTAU | DLL Version: 9.2 | Buffers Released | Not writing Timestamps | Server Running...

Figure 3.9: Display of quTAU's interface.

The mean photon number was tested a second time with the same detector, gated at 1KHz, with a detection window of 20ns and again set to 10% efficiency. We used the setup shown in Fig.3.8 for this measurement.

Using quTAU the collected time stamps were displayed in a histogram. Increasing the coincidence window to 100 bins allows all the pulse photons to be acquired on a solo peak in the histogram. With this, the number of detected photons can directly be measured from the peak's hight. Since the detector is gated, we will not need to use dead time corrections.

With the gated detector, 115 photons were measured over an integration time of 100s of the histogram. Since this corresponds to a detector of 10% efficiency, the number of photons from the pulse arriving at the detector over a 100s period is 1150. This corresponds to a mean photon number of

$$\mu = 0.0115 \quad photons/pulse$$

3.5 Experimental Results

Firstly, we ran test measurements with the INGAS single photon detector with 10% efficiency and compensated the quantum channel losses by increasing the mean photon number. Here we collected some preliminary probability distribution matrices to assess the set up before running the experiment with the high efficiency detector.

We ran the experiment for a period of 3 hours, using the experimental setup pictured in Fig.3.1. The single photon detector used by Bob for his measurements was the 90%

efficiency SNSPD discussed earlier in Section 3.1. The total transmission from Alice's output to Bob's detector measured 92%, and the mean photon number was prepared to be $\mu = 0.01$ For a dark count probability of 3.24×10^{-7} and 1.144125×10^{7} total trials, we obtained the following detection probability matrix:

$$prob = \begin{bmatrix} 1.00845280 \times 10^{-4} & 1.89569084 \times 10^{-3} & 4.23695029 \times 10^{-3} & 2.72414919 \times 10^{-3} \\ 2.32493803 \times 10^{-4} & 1.07624722 \times 10^{-4} & 1.53587076 \times 10^{-3} & 3.76759283 \times 10^{-3} \\ 3.54147433 \times 10^{-3} & 1.56870202 \times 10^{-4} & 5.46026537 \times 10^{-5} & 1.65985771 \times 10^{-3} \\ 1.99347159 \times 10^{-3} & 3.74898445 \times 10^{-3} & 1.94928230 \times 10^{-3} & 6.57426330 \times 10^{-5} \end{bmatrix}$$

And the matrix of total number of clicks:

$$clicks = \begin{bmatrix} 72 & 1355 & 3029 & 1952 \\ 1664 & 77 & 1099 & 2693 \\ 2528 & 1122 & 39 & 1188 \\ 1426 & 2681 & 1394 & 47 \end{bmatrix}$$

Remember the data was registered in the format of matrix M (please refer to Section 3.2.2).

Alice's monitoring results are plotted in Fig.3.10. Allowing us to infer that the state preparation was stable throughout the run of the experiment.

3.6 Discussion

For a total transmission of the setup of 82.5% and a mean photon number of $\mu = 0.01$ and the measured dark count probability of 3.24×10^{-7}, we calculated the probability distribution matrix that is foreseen by theory and can be seen below.

$$prob_t = \begin{bmatrix} 3.24000000 \times 10^{-7} & 2.07317268 \times 10^{-3} & 4.14172465 \times 10^{-3} & 2.07317268 \times 10^{3} \\ 2.07317268 \times 10^{-3} & 3.24000000 \times 10^{-7} & 2.07317268 \times 10^{-4} & 4.14172465 \times 10^{-3} \\ 4.14172465 \times 10^{-3} & 2.07317268 \times 10^{-3} & 3.24000000 \times 10^{-7} & 1.65985771 \times 10^{-3} \\ 2.07317268 \times 10^{-3} & 4.14172465 \times 10^{-3} & 2.07317268 \times 10^{-3} & 3.24000000 \times 10^{-7} \end{bmatrix}$$

As one can see, $prob_{theory}$ differs from the measured probability distribution. Since in matrix *prob* the order of magnitude of the diagonal elements is much greater than the dark count probability, we can infer dark counts are not the main prompter of the discrepancies seen in the matrices. For that, we must look for another phenomenon capable of inducing

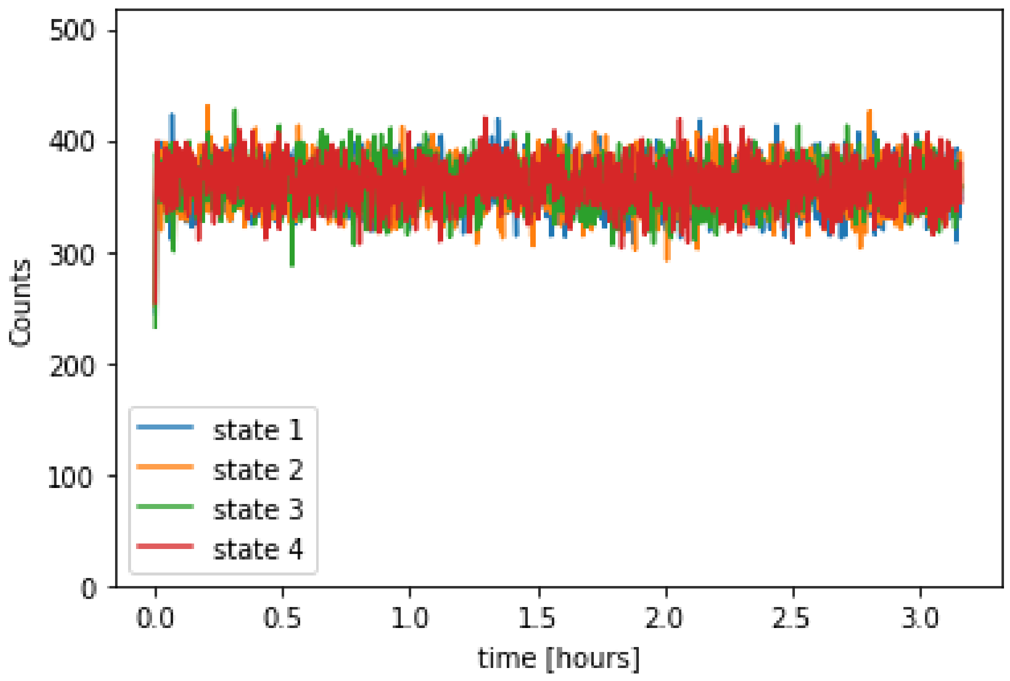


Figure 3.10: Alice's monitoring results over a span o 3 hours. Each colored line represents one of the four states being prepared.

such outcomes.

The difference in the diagonal states stems from the states prepared in both polarization controllers not being exactly the same, as well as imperfect polarization alignment at the beginning of the experiment. The discrepancies of the probabilities in other elements of the matrix are likely due to an incorrect estimation of μ or faulty assessment of the detector's efficiency and fiber transmission.

The collected data was sent to the theory group to perform the key sifting and security analysis.

4

Conclusion

The aim of this project was to experimentally implement the testing of a semi-device independent QKD protocol. With that goal in mind, we were able to realize an experimental set-up that allowed a stable manipulation of the polarization states needed for the encoding as well as measurement.

We built the hardware needed to control the polarization controllers responsible for the state preparation and measurement basis selection. These polarization controllers were characterized for speed and polarization shift with applied voltage. With this information, we were then able to estimate the parameters needed to prepare the four desired polarization states. We also showed that, using an optimization function, we could in principle increase the state preparation frequency up to 6.6 kHz. However, to go any further, we would have to fully redesign the experimental setup. One idea would be to use time-bin based encoding. For that the different states would be set using phase modulators that can work in the GHz range.

The full experiment was then implemented. Using a SNSPD with 90% detection efficiency, we could achieve a quantum channel with 83% transmission efficiency. The probability distribution was thus measured for the following parameters, $\mu = 0.01$ and dark count probability of 3.24×10^{-7}, and compared with the theoretical distribution.

This work did however not provide a complete result of the key sifting nor the security analysis, since the theory behind the security proof is still an ongoing investigation by the theory group.

To conclude, in this thesis, is expounded the experimental implementation for a novel approach to a semi-device independent QKD protocol with four-coherent-state polarization-encoding. The work discussed stands as a starting ground for future works. The following stages of this project will focus on complementing the experimental realization with real-time key sifting. The results will then also be analysed with latest security proof of the protocol.

Bibliography

[1] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, "Handbook of applied cryptography, crc press series on discrete mathematics and its applications., crc press, boca raton, fl," vol. 1997.

[2] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical Computer Science*, vol. 560, p. 7–11, Dec 2014.

[3] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 26, p. 96–99, Jan. 1983.

[4] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, no. 5886, pp. 802–803, 1982.

[5] P. Reuvers and M. Simons, "One-time pad (otp) - the unbreakable code," May 2011.

[6] W. K. Wootters and W. H. Zurek, "The concept of transition in quantum mechanics," *Foundations of Physics*, vol. 1, no. 1, pp. 23–33, 1970.

[7] J. L. Park, "The concept of transition in quantum mechanics," *Foundations of Physics*, vol. 1, no. 1, pp. 23–33, 1970.

[8] J. Preskill, "Lecture notes for physics: Quantum information and computation," September 1998.

[9] M. A. Nielsen and I. L. Chuang, *Quantum Computation Quantum Infomation.* Cambridge University Press Textbooks, 1900.

[10] N. Lütkenhaus, "Secret keys from quantum correlations," *Informatik - Forschung und Entwicklung*, vol. 21, pp. 29–37, 10 2006.

[11] N. Lütkenhaus, "Secret keys from quantum correlations," *Informatik - Forschung und Entwicklung*, vol. 21, pp. 29–37, 10 2006.

[12] Y. Zhao, B. Qi, X. Ma, H.-K. Lo, and L. Qian, "Experimental quantum key distribution with decoy states," *Phys. Rev. Lett.*, vol. 96, p. 070502, Feb 2006.

[13] Y. Zhao, B. Qi, X. Ma, H.-K. Lo, and L. Qian, "Simulation and implementation of decoy state quantum key distribution over 60km telecom fiber," pp. 2094 – 2098, 08 2006.

[14] A. Boaron, G. Boso, D. Rusca, C. Vulliez, C. Autebert, M. Caloz, M. Perrenoud, G. Gras, F. Bussières, and M.-J. e. a. Li, "Secure quantum key distribution over 421 km of optical fiber," *Physical Review Letters*, vol. 121, no. 19, p. 190502, 2018.

[15] A. Boaron, B. Korzh, R. Houlmann, G. Boso, D. Rusca, S. Gray, M.-J. Li, D. Nolan, A. Martin, and H. Zbinden, "Simple 2.5ghz time-bin quantum key distribution," *Applied Physics Letters*, vol. 112, no. 17, p. 171108, 2018.

[16] I. Gerhardt, Q. Liu, A. Lamas-Linares, J. Skaar, C. Kurtsiefer, and V. Makarov, "Full-field implementation of a perfect eavesdropper on a quantum cryptography system," *Nature Communications*, vol. 2, p. 349, Jun 2011.

[17] L. Lydersen, C. Wiechers, C. Wittmann, D. Elser, J. Skaar, and V. Makarov, "Hacking commercial quantum cryptography systems by tailored bright illumination," *Nature Photonics*, vol. 4, p. 686–689, Aug 2010.

[18] D. Gottesman, H. . Lo, N. Lutkenhaus, and J. Preskill, "Security of quantum key distribution with imperfect devices," 2004.

[19] R. Arnon-Friedman, R. Renner, and T. Vidick, "Simple and tight device-independent security proofs," *SIAM Journal on Computing*, vol. 48, p. 181–225, Jan 2019.

[20] A. Einstein, B. Podolsky, and N. Rosen, "Can quantum-mechanical description of physical reality be considered complete?," *Physical Review*, vol. 47, pp. 777–780, 1935.

[21] J. S. Bell, "On the einstein podolsky rosen paradox," *Physics Physique Fizika*, vol. 1, pp. 195–200, Nov 1964.

[22] J. F. Clauser, M. A. Horne, A. Shimony, and R. A. Holt, "Proposed experiment to test local hidden-variable theories," *Phys. Rev. Lett.*, vol. 23, pp. 880–884, Oct 1969.

[23] V. Scarani, "The device-independent outlook on quantum physics (lecture notes on the power of bell's theorem)," 2015.

[24] A. K. Ekert, "Quantum cryptography based on bell's theorem," *Phys. Rev. Lett.*, vol. 67, pp. 661–663, Aug 1991.

[25] A. Acín, N. Brunner, N. Gisin, S. Massar, S. Pironio, and V. Scarani, "Device-independent security of quantum cryptography against collective attacks," *Phys. Rev. Lett.*, vol. 98, p. 230501, Jun 2007.

[26] P. M. Pearle, "Hidden-variable example based upon data rejection," *Phys. Rev. D*, vol. 2, pp. 1418–1425, Oct 1970.

[27] J. F. Clauser, M. A. Horne, A. Shimony, and R. A. Holt, "Proposed experiment to test local hidden-variable theories," *Phys. Rev. Lett.*, vol. 23, pp. 880–884, Oct 1969.

[28] N. Sangouard, C. Simon, H. de Riedmatten, and N. Gisin, "Quantum repeaters based on atomic ensembles and linear optics," *Rev. Mod. Phys.*, vol. 83, pp. 33–80, Mar 2011.

[29] N. Gisin, S. Pironio, and N. Sangouard, "Proposal for implementing device-independent quantum key distribution based on a heralded qubit amplifier," *Phys. Rev. Lett.*, vol. 105, p. 070501, Aug 2010.

[30] A. Acín, N. Brunner, N. Gisin, S. Massar, S. Pironio, and V. Scarani, "Device-independent security of quantum cryptography against collective attacks," *Phys. Rev. Lett.*, vol. 98, p. 230501, Jun 2007.

[31] J. Barrett, R. Colbeck, and A. Kent, "Memory attacks on device-independent quantum cryptography," *Phys. Rev. Lett.*, vol. 110, p. 010503, Jan 2013.

[32] H.-K. Lo, M. Curty, and B. Qi, "Measurement-device-independent quantum key distribution," *Phys. Rev. Lett.*, vol. 108, p. 130503, Mar 2012.

[33] S. Pirandola, R. Laurenza, C. Ottaviani, and L. Banchi, "Fundamental limits of repeaterless quantum communications," *Nature Communications*, vol. 8, p. 15043, Apr 2017.

[34] M. Lucamarini, Z. L. Yuan, J. F. Dynes, and A. J. Shields, "Overcoming the rate–distance limit of quantum key distribution without quantum repeaters," *Nature*, vol. 557, no. 7705, pp. 400–403, 2018.

[35] S. Wang, D.-Y. He, Z.-Q. Yin, F.-Y. Lu, C.-H. Cui, W. Chen, Z. Zhou, G.-C. Guo, and Z.-F. Han, "Beating the fundamental rate-distance limit in a proof-of-principle quantum key distribution system," *Phys. Rev. X*, vol. 9, p. 021046, Jun 2019.

[36] J.-P. Chen, C. Zhang, Y. Liu, C. Jiang, W. Zhang, X.-L. Hu, J.-Y. Guan, Z.-W. Yu, H. Xu, J. Lin, M.-J. Li, H. Chen, H. Li, L. You, Z. Wang, X.-B. Wang, Q. Zhang, and J.-W. Pan, "Sending-or-not-sending with independent lasers: Secure twin-field quantum key distribution over 509 km," *Phys. Rev. Lett.*, vol. 124, p. 070501, Feb 2020.

[37] P. Villoresi, T. Jennewein, F. Tamburini, M. Aspelmeyer, C. Bonato, R. Ursin, C. Pernechele, V. Luceri, G. Bianco, and A. e. a. Zeilinger, "Experimental verification of the feasibility of a quantum channel between space and earth," *New Journal of Physics*, vol. 10, no. 3, p. 033038, 2008.

[38] G. Vallone, D. Bacco, D. Dequal, S. Gaiarin, V. Luceri, G. Bianco, and P. Villoresi, "Experimental satellite quantum communications," *Physical Review Letters*, vol. 115, p. 040502, Jul 2015.

[39] H. Xin, "Chinese academy takes space under its wing," *Science*, vol. 332, no. 6032, pp. 904–904, 2011.

[40] J.-G. Ren, P. Xu, H.-L. Yong, L. Zhang, S.-K. Liao, J. Yin, W.-Y. Liu, W.-Q. Cai, M. Yang, and L. e. a. Li, "Ground-to-satellite quantum teleportation," *Nature*, vol. 549, no. 7670, pp. 70–73, 2017.

[41] S.-K. Liao, W.-Q. Cai, W.-Y. Liu, L. Zhang, Y. Li, J.-G. Ren, J. Yin, Q. Shen, Y. Cao, Z.-P. Li, and et al., "Satellite-to-ground quantum key distribution," *Nature*, vol. 549, p. 43–47, Aug 2017.

[42] J. Yin, Y. Cao, Y.-H. Li, S.-K. Liao, L. Zhang, J.-G. Ren, W.-Q. Cai, W.-Y. Liu, B. Li, H. Dai, G.-B. Li, Q.-M. Lu, Y.-H. Gong, Y. Xu, S.-L. Li, F.-Z. Li, Y.-Y. Yin, Z.-Q. Jiang, M. Li, J.-J. Jia, G. Ren, D. He, Y.-L. Zhou, X.-X. Zhang, N. Wang, X. Chang, Z.-C. Zhu, N.-L. Liu, Y.-A. Chen, C.-Y. Lu, R. Shu, C.-Z. Peng, J.-Y. Wang, and J.-W. Pan, "Satellite-based entanglement distribution over 1200 kilometers," 2017.

[43] L. Shengkai, H.-L. Yong, C. Liu, G.-L. Shentu, D.-D. Li, J. Lin, H. Dai, S.-Q. Zhao, b. li, J.-Y. Guan, W. Chen, Y.-H. Gong, Y. Li, Z.-H. Lin, G.-S. Pan, J. Pelc, M. Fejer, W.-Z. Zhang, W. Liu, and J.-W. Pan, "Long-distance free-space quantum key distribution in daylight towards inter-satellite communication," *Nature Photonics*, vol. 11, p. 509–513, 07 2017.

[44] L. Liu, Y. Wang, E. Lavie, C. Wang, A. Ricou, F. Z. Guo, and C. C. W. Lim, "Practical quantum key distribution with non-phase-randomized coherent states," *Phys. Rev. Applied*, vol. 12, p. 024048, Aug 2019.

[45] M. Pawłowski and N. Brunner, "Semi-device-independent security of one-way quantum key distribution," *Phys. Rev. A*, vol. 84, p. 010302, Jul 2011.

[46] J. Bowles, M. T. Quintino, and N. Brunner, "Certifying the dimension of classical and quantum systems in a prepare-and-measure scenario with independent devices," *Phys. Rev. Lett.*, vol. 112, p. 140407, Apr 2014.

[47] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 4, pp. 623–656, 1948.

[48] Y. Wang, I. W. Primaatmaja, E. Lavie, A. Varvitsiotis, and C. C. W. Lim, "Characterising the correlations of prepare-and-measure quantum networks," *npj Quantum Information*, vol. 5, p. 17, Feb 2019.

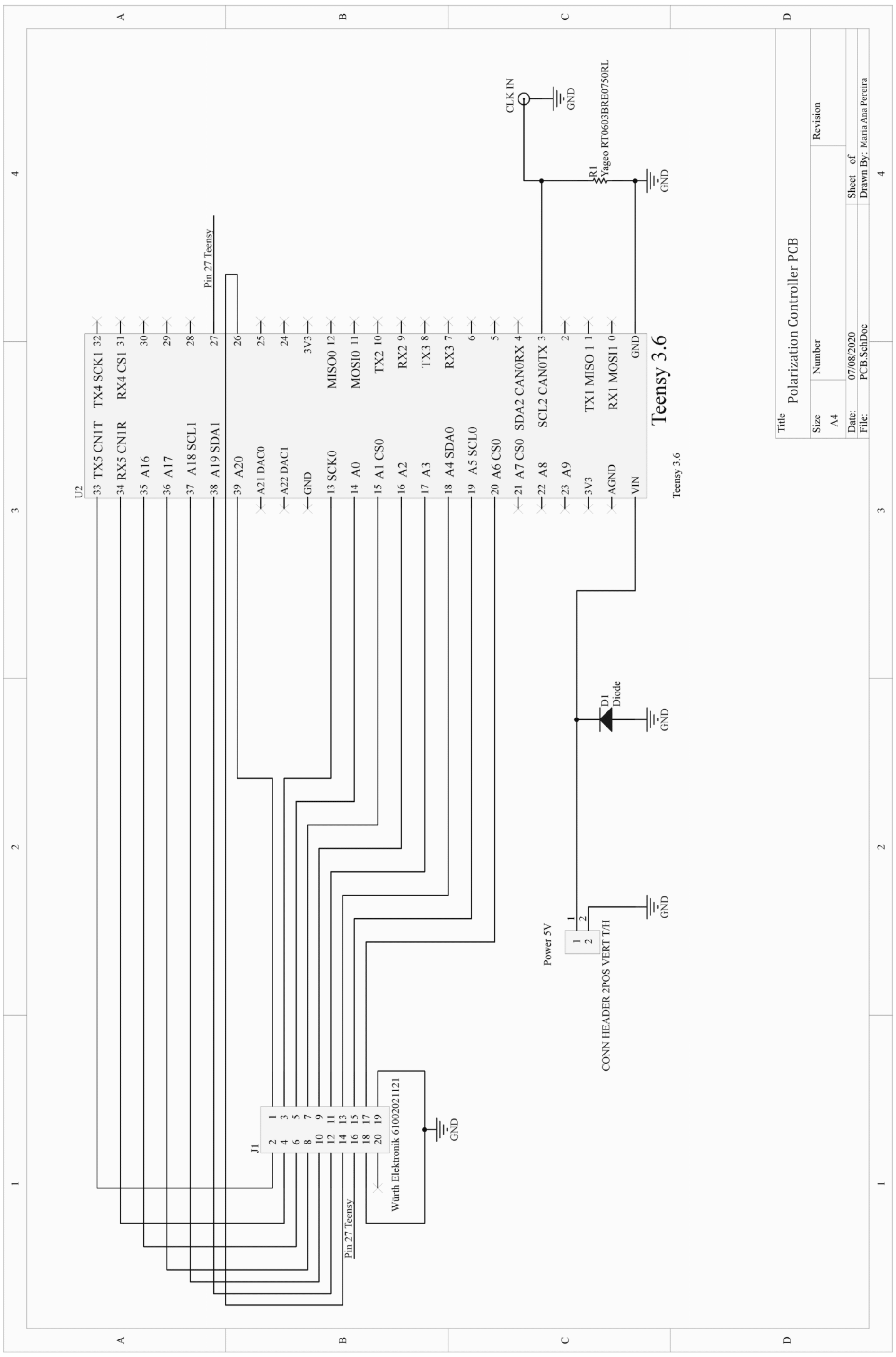
[49] C. H. Bennett, “Quantum cryptography using any two nonorthogonal states,” *Phys. Rev. Lett.*, vol. 68, pp. 3121–3124, May 1992.

[50] N. Lütkenhaus, “Security against individual attacks for realistic quantum key distribution,” *Phys. Rev. A*, vol. 61, p. 052304, Apr 2000.

Appendices

A

First appendix
PCB's Schematics


Teensy 3.6
U2
CLK IN
GND
R1
Yageo RT0603BRE0750RL
Pin 27 Teensy
D1
Diode
Power 5V
CONN HEADER 2POS VERT T/H
J1
Würth Elektronik 61002021121
Title
Polarization Controller PCB
Size
A4
Number
Revision
Date: 07/08/2020
File: PCB.SchDoc
Sheet of
Drawn By: Maria Ana Pereira

B

Second appendix
State Optimization

Teensy code:

```
//current voltage value
int voltage = 0;

//current channel
int ch = 0;

//optmial voltage values per channel
int optV [] = {0,0,0,0};



// === PINS ===

// Digital Signal Bits
const int DB0 =  35; //LSB
const int DB1 =  15;
const int DB2 =  36;
const int DB3 = 16;
const int DB4 = 37;
const int DB5 = 17;
const int DB6 = 38;
const int DB7 = 18;
const int DB8 = 26;
const int DB9 = 19;
const int DB10 = 27;
const int DB11 = 20; //MSB

// Channel control bits
int Ch1 = 34;
int Ch2 = 14;
```

```
//Read/Write
int RW = 39;

//Chip Select
int CS = 33;


// === GLOBAL VARIABLES ===

// Array with the digital signal input in binary
int digitalSig[13];

// Array with the channel # in binary
int digitalCh[3];

// Array with the digital signal pins
int digPin_array  [13] = {DB11, DB10, DB9, DB8, DB7, DB6, DB5, DB4,
DB3, DB2, DB1, DB0};

// Array with the channel pins
int ChPin_array  [3] = {Ch1, Ch2};

// Array with the channel #s in binary
int digitalChs[4][2] = {{0,0},{1,0},{0,1},{1,1}};

// Clock Flag Variable
volatile int stateFlag;

//  Array with Voltage Values per State
//  To call -> StatesxVoltages [voltage][state]
int StatesxVoltages[4][4] = {{209,455,0,0}};


// States Queue in Arduino
int statesBuffer[2000];

// Message Recieved from Python (Array)
char serialArray[200];

// Message Recieved from Python (String)
String serialMessage(serialArray);

// Delay
int microsecs = 0;
```

```
void setup() {////////////////////////////////////////////////////////

  Serial.begin(9600);
  Serial.println("Arduino Ready");

  // ———————— INITIALIZE DIGITAL PINS AS OUTPUTS ————————
  for (int i = 0 ; i < 12 ; i++){
    pinMode(digPin_array[i], OUTPUT);
  }

  pinMode(Ch1, OUTPUT);
  pinMode(Ch2, OUTPUT);

  pinMode(RW, OUTPUT);

  pinMode(CS, OUTPUT);

  // ——————— INITIALIZE CHANNEL VALUES ———————

  //—————————————— CH = 0 ——————————————
  digitalCh[0] = 0;
  digitalCh[1] = 0;

  for (int i = 0; i < 2; i++) { //channel
    if (digitalCh[i] == 0) {
      digitalWrite(ChPin_array[i], LOW);
    }
    else if (digitalCh[i] == 1) {
      digitalWrite(ChPin_array[i], HIGH);
    }
  }
    for (int i = 0; i < 12; i++) { //value
      digitalWrite(digPin_array[i], LOW);
  }

  digitalWrite(CS, LOW);
  delayMicroseconds(1);
  digitalWrite(CS, HIGH);


  //—————————————— CH = 1 ——————————————
  digitalCh[0] = 1;
```

```
digitalCh[1] = 0;

for (int i = 0; i < 2; i++) { //channel
  if (digitalCh[i] == 0) {
    digitalWrite(ChPin_array[i], LOW);
  }
  else if (digitalCh[i] == 1) {
    digitalWrite(ChPin_array[i], HIGH);
  }
}

for (int i = 0; i < 12; i++) { //value
    digitalWrite(digPin_array[i], LOW);
}

digitalWrite(CS, LOW);
delayMicroseconds(1);
digitalWrite(CS, HIGH);



//---------------- CH = 2 ----------------
digitalCh[0] = 0; //<*
digitalCh[1] = 1; //<*

for (int i = 0; i < 2; i++) { //channel
  if (digitalCh[i] == 0) {
    digitalWrite(ChPin_array[i], LOW);
  }
  else if (digitalCh[i] == 1) {
    digitalWrite(ChPin_array[i], HIGH);
  }
}

for (int i = 0; i < 12; i++) { //value
    digitalWrite(digPin_array[i], LOW);
}

digitalWrite(CS, LOW);
delayMicroseconds(1);
digitalWrite(CS, HIGH);

//---------------- CH = 3 ----------------
digitalCh[0] = 1; //<*
digitalCh[1] = 1; //<*
```

```
  for (int i = 0; i < 2; i++) { //channel
    if (digitalCh[i] == 0) {
      digitalWrite(ChPin_array[i], LOW);
    }
    else if (digitalCh[i] == 1) {
      digitalWrite(ChPin_array[i], HIGH);
    }
  }

  for (int i = 0; i < 12; i++) { //value
      digitalWrite(digPin_array[i], LOW);
  }

  digitalWrite(CS, LOW);
  delayMicroseconds(1);
  digitalWrite(CS, HIGH);

}//////////////////////////////////////////////////////////////////////


void loop() {#########################################################

  if(Serial.available() > 0) {
        Serial.readBytesUntil('\n',serialArray,50);
        String serialMessage(serialArray);

        if(serialMessage.substring(0,7) == "VOLTAGE"){
             Serial.print(voltage);
             delayMicroseconds(1);
             Serial.flush();

             //write current voltage value to ch
             convertBinary (voltage, digitalSig);
             writeState (digitalChs[ch][0], digitalChs[ch][1],
             ChPin_array, digitalSig, digPin_array);

             //increases voltage value
             voltage = voltage + 50;

        }
        else if(serialMessage.substring(0,4) == "VAL"){
           int newV = serialMessage.substring(5,9).toInt();

           //save opt Voltage to array
```

```
            optV[ch] = newV;

            //write current voltage value to ch
            convertBinary (newV, digitalSig);
            writeState (digitalChs[ch][0], digitalChs[ch][1],
            ChPin_array, digitalSig, digPin_array);

        }

        //Quando chega a 2PI recome a para um novo ch
        if (voltage > 1500){
          voltage = 0;
          if (ch == 3) {ch = 0;}
          else {ch=ch+1;}
        }
  }
}###########################################################################

//————————————————————————————————————————————————————————————————————————(1)
void convertBinary (int val, int digitalSig[]) {

  //converts the voltage values to binary (stored in an Array
  digitalSig[])
  int remainders[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
  int cont = 0;
  int i = 0;

  while ( val > 0 ) {
    remainders[cont] = val % 2; //saves the remainders all in one
                                        list
    val = val / 2; //keeps the value given divided by 2
    cont++; //adds to counter
  }

  int a = 0;
  for (int i = 11; i >= 0; i--) {
    digitalSig[i] = remainders[a];
    a++;
  }
}//-----------------------------------------------------------------------

//———————————————————————————————————————————————————————————————————— (2)
void writeState (int digitalCh0, int digitalCh1, int ChPin_array[],
int digitalSig[], int digPin_array[]){
  //write channel # and voltage value to the PolCont
```

```
      for (int i = 0; i < 2; i++) { //channel
        if (digitalCh0 == 0) {
          digitalWrite(ChPin_array[i], LOW);
        }
        else if (digitalCh1 == 1) {
          digitalWrite(ChPin_array[i], HIGH);
        }
      }

     for (int i = 0; i < 12; i++) { //digPin_array
        if(digitalSig[i]==0){
          digitalWrite(digPin_array[i],LOW);
        }
        else if(digitalSig[i]==1){
          digitalWrite(digPin_array[i],HIGH);
        }
     }

   digitalWrite(CS, LOW);
   for (int i = 0 ; i<10000; i++){};
   digitalWrite(CS, HIGH);

}//----------------------------------------------------------------
```

Python code:

```
# -*- coding: utf-8 -*-
import os
import polarimeter as pol
import time
from datetime import datetime
import numpy as np
import matplotlib.pyplot as plt
import matlab.engine
import serial

# timing parameters
sleeptime = 5
t0 = datetime.utcfromtimestamp(0) # epoch

# initialize data array
T = np.array([])
theta = np.array([])
eta = np.array([])
voltages = np.array([])
```

```
#initialize polarimeter
polo = pol.polarimeter()

#initialize matlab.engine
eng = matlab.engine.start_matlab()

#initialize serial connection
teensy = serial.Serial('COM4',9600)
time.sleep(1) #give the connection a second to settle

ch = 0

# === COMMUNICATION LOOP ===
try:
    while True:
        # ----- TEENSY LOOP ------
        while True:
            # Recieves voltage values from arduino ==> Reads polarimeter
            teensy.write('VOLTAGE') # request voltage
            time.sleep(0.5)# waits for reply

            data = teensy.readline()[:-2] #the last bit gets rid of the
            new-line chars

                if data:

                #saves current voltage value
                voltages = np.append(voltages,data)

                #saves polarimeter data
                az,el = polo.read_polarimeter()
                t1 = datetime.now()
                ts1 = (t1-t0).total_seconds()
                T = np.append(T,ts1)
                theta = np.append(theta,az)
                eta = np.append(eta,el)

                time.sleep(sleeptime)

            # Quando 2PI termina while: True
            if int(data) == 1500 :
                break

        # ----- END TEENSY LOOP ------
        # =========== AFTER 1st 2PI =====================

        # —— OPTIMIZATION FUNCTION ——
        volt,ang = eng.V_min(voltages,theta,eta)
        f.write(ch, volt) #saves ch and corresponding opt voltage

        teensy.write('VAL',volt)
        ch=ch+1

        # ——— END OF OPTIMIZATION ———
        if ang < 0.1
            break
# === End Data Acquisition Loop ===

        #reset voltages,theta,eta
        voltages.clear()
        theta.clear()
        eta.clear()


except KeyboardInterrupt: #close connection
    polo.close_polarimeter()
    f.close()
```

Matlab function:

```
function [volt,ang] = V_min(voltages,theta,eta)
%[LINHA][COLUNA]

%voltages[100] array com as tens es preparadas neste canal
%angles[100][2] array com os angulos correspondentes as tensoes

%% − SELECT REFERENCE STATE −
state = 1;
if state == 1
    stateTheta = 0;
    stateEta = 15;
elseif state == 2
     stateTheta = −45;
     stateEta = 15;
elseif state == 3
    stateTheta = 45;
    stateEta = 15;
else
    stateTheta = 90;
    stateEta = 15;
end

%% CREATING 3 ELEMENT VECTORS
% -- REFERENCE VECTOR --
x0 = cos(stateTheta.*pi./90).*cos(stateEta.*pi./90);
y0 = sin(stateTheta.*pi./90).*cos(stateEta.*pi./90);
z0 = sin(stateEta.*pi./90);

Vec0 = [x0 y0 z0];



% -- LIST RECIEVED VECTORS --
x = cos(theta.*pi./90).*cos(eta.*pi./90);
y = sin(theta.*pi./90).*cos(eta.*pi./90);
z = sin(eta.*pi./90);

Vecs = [x y z];

%% CALCULATE ANGLES
size = length(voltages);
a = zeros(size,1);

% -- CALCULATE ANGLES IN RELATION TO THE REFERENCE VECTOR --

for i = 1:size
    a(i) = atan2(norm(cross(Vec0,Vecs(i,:))),dot(Vec0,Vecs(i,:)))
    % Angle in radians
end

[M,I] = min(a)



ang = a(I);
volt = voltages(I);
```

C

Third appendix Teensy's State Preparation Software

```
    #define BUFFER_SIZE 200

// ===== PINS =====

// Digital Signal Bits
const int DB0 =  35; //LSB
const int DB1 =  15;
const int DB2 =  36;
const int DB3 = 16;
const int DB4 = 37;
const int DB5 = 17;
const int DB6 = 38;
const int DB7 = 18;
const int DB8 = 26;
const int DB9 = 19;
const int DB10 = 27;
const int DB11 = 20; //MSB

// Channel control bits
int CH1 = 34;
int CH2 = 14;

//Read/Write
int RW = 39;

//Chip Select
int CS = 33;
```

```
// ===== GLOBAL VARIABLES =====

// - - - WRITE TO PC - - -

// Array with the digital signal pins
int SIG_PIN_ARRAY  [13] = {DB11, DB10, DB9, DB8, DB7, DB6, DB5, DB4,
DB3, DB2, DB1, DB0};

// Array with the channel pins
int CH_PIN_ARRAY  [3] = {CH1, CH2};

// Array with the digital signal input in binary
int digitalSig[13];

//Array with the channel number in binary
int digitalCh[2];

//  Array with Voltage Values per State
//  To call -> STATE_VOLTAGES [voltage][state]

// = = = = = PC1 = = = = =
//        (COM 20)  BOB
int STATE_VOLTAGES[5][4] = { {183,    0,  0, 0},      //0 226
                             {183,  485, 0, 0},       //1
                             {183,  850, 0, 0},       //2
                             {183, 1270, 0, 0},       //3
                             {  0,    0, 0, 0}};




// = = = = = PC2 = = = = =
//        (COM 16)  ALICE
//int STATE_VOLTAGES[5][4] = { {170,    0, 0, 0},      //0   219
//                             {170,  454, 0, 0},      //1
//                             {170,  900, 0, 0},      //2   old(948)
//                             {170, 1286, 0, 0},      //3
//                             {  0,    0, 0, 0}};



// Voltage of each CH of a State
int ch_voltage[4]; //[0 1 2 3]
```

```
// − − − BUFFER − − −
// States Queue in Arduino
int statesBuffer[BUFFER_SIZE];

// State Preparation Pointer
int cont = 0;

// State to be Prepared
int state;

// Marker for Buffer Update
int marker;


// − − − SERIAL COM − − −
// Message Recieved from Python (Array)
char serialArray[BUFFER_SIZE*3/4];

// Message Recieved from Python (String)
String serialMessage(serialArray);

// States to be Added to the Buffer
int newStates[BUFFER_SIZE/2];

// Delay
int delay_micros = 0;


// −−− INTERRUPT −−−
// Clock Flag Variable
volatile int STATE_FLAG;



void setup() {//****************************************************
  Serial.begin(912600);

 // −−−−−−−−− INITIALIZE DIGITAL PINS AS OUTPUTS −−−−−−−−
    for (int i = 0 ; i < 12 ; i++) {
      pinMode(SIG_PIN_ARRAY[i], OUTPUT);
    }

    pinMode(CH1, OUTPUT);
    pinMode(CH2, OUTPUT);
    pinMode(RW, OUTPUT);
    pinMode(CS, OUTPUT);


 // ———— INITIALIZE DIGITAL PIN AS INTERRUPT ————
    attachInterrupt(digitalPinToInterrupt(3), ISR_state, RISING);


 // ———— INITIALIZE CHANNEL VOLTAGES ————
    setUp(CH_PIN_ARRAY, SIG_PIN_ARRAY, digitalSig);


 // ——— initialize state queue ———
    for (int j = 0 ; j < BUFFER_SIZE ; j++) {
      if (j % 2 == 0){
        statesBuffer[j] == 1;
      }
      else{
        statesBuffer[j] == 0;
      }
   }
}

//****************************************************************

void loop() { //%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  // ==================== STATE PREPARATION ====================

    // ————— Reads States from Buffer —————
    boolean request = false;

    if (STATE_FLAG) {

      STATE_FLAG = 0;
      //delayMicroseconds(delay_micros); //delay is now done externally
      state = statesBuffer[cont]; // State to be prepared

      // ——— Prepares State ———
        prepareState(state, ch_voltage); // Set corresponding state
        value to each Pol Cont channel
        for (int channel = 0 ; channel < 4 ; channel++) {
        // ——— Iterates over channels ———
          convertBinary (ch_voltage[channel], digitalSig);
          // Converts voltage to binary
          writeState (channel, CH_PIN_ARRAY, digitalSig, SIG_PIN_ARRAY);
        }

        // Deletes preapared state from Buffer
        //statesBuffer[cont] = 4;

      // ----- Checks for states -----
        //if (cont == BUFFER_SIZE/4 or cont == BUFFER_SIZE*3/4) {
        if (cont == 0 or cont == BUFFER_SIZE/2) { //ALICE
        //if (cont == 10 or cont == (BUFFER_SIZE/2)+10) { //BOB
          marker = cont;
          request = true;

        }
        cont++;
        if (cont == BUFFER_SIZE) {
          cont = 0;
        }
    }

    // ------------ Request More States ------------
    if (request) {
      request = false;
      Serial.write("G");


    }

  //===================================================================



  //======================== PYTHON COM ========================
    if (Serial.available() > 0) {
      Serial.readBytesUntil('\n', serialArray, 1012);
      String serialMessage(serialArray);

       // ---------------        START      ----------------------
         if(serialMessage.substring(0,5) == "START"){
            cont = 0;
         }

      // --------------- STATES QUEUE ----------------------
         if(serialMessage.substring(0,9) == "SETSTATES"){
            int serialMessage_array[110];
            for(int i = 0 ; i < 100 ; i++) {
               newStates[i] = serialArray[i+10]-48;
               //Array with States int(0,1,2,3) -> to be added to BUFFER
            }
            writeToBuffer(marker, statesBuffer, newStates);
         }
      // ----------- SET NEW VOLTAGE VALUES ----------------
         else if (serialMessage.substring(0, 11) == "SETVOLTAGES") {
           setNewVoltages(serialMessage);
         }
      // -------------- EMPTY BUFFER ------------------------
         else if (serialMessage.substring(0, 5) == "RESET") {
           emptyBuffer(statesBuffer);
         }
      // --------------- ADD DELAY --------------------------
         else if (serialMessage.substring(0, 5) == "DELAY") {
           delay_micros = serialMessage.substring(6, 10).toInt(); //
         }
    }

}//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




//######################## --- FUNCTIONS --- ########################



//---------------------------------------------------------------(1)
void ISR_state() {
  STATE_FLAG = 1;
}//-------------------------------------------------------------



//---------------------------------------------------------------(2)
void convertBinary (int val, int digitalSig[]) {

  //converts the voltage values to binary (stored in an Array
  digitalSig[])
  int remainders[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
  int cont = 0;
  int i = 0;

  while ( val > 0 ) {
    remainders[cont] = val % 2; //saves the remainders all in one list
    val = val / 2; //keeps the value given divided by 2
    cont++; //adds to counter
  }

  int a = 0;
  for (int i = 11; i >= 0; i--) {
    digitalSig[i] = remainders[a];
    a++;
  }
}//-------------------------------------------------------------



//———————————————————————————————————————————————————————————————(3)
void prepareState (int state, int ch_voltage[]) {

  //. . . . . . . . . . . .
  //
  // SETS CHANNEL VOLTAGE VALUES DEPENDING ON THE SELECTED STATE
  // STATE_VOLTAGES[state][voltage]
  //
  //. . . . . . . . . . . .

  if (state == 0) {
    ch_voltage[0] = STATE_VOLTAGES[0][0];
    ch_voltage[1] = STATE_VOLTAGES[0][1];
    ch_voltage[2] = STATE_VOLTAGES[0][2];
    ch_voltage[3] = STATE_VOLTAGES[0][3];
  }else if (state == 1) {
    ch_voltage[0] = STATE_VOLTAGES[1][0];
    ch_voltage[1] = STATE_VOLTAGES[1][1];
    ch_voltage[2] = STATE_VOLTAGES[1][2];
    ch_voltage[3] = STATE_VOLTAGES[1][3];
  } else if (state == 2) {
    ch_voltage[0] = STATE_VOLTAGES[2][0];
    ch_voltage[1] = STATE_VOLTAGES[2][1];
    ch_voltage[2] = STATE_VOLTAGES[2][2];
    ch_voltage[3] = STATE_VOLTAGES[2][3];
  } else if (state == 3) {
    ch_voltage[0] = STATE_VOLTAGES[3][0];
    ch_voltage[1] = STATE_VOLTAGES[3][1];
    ch_voltage[2] = STATE_VOLTAGES[3][2];
    ch_voltage[3] = STATE_VOLTAGES[3][3];
  } else if (state == 4) {
    ch_voltage[0] = STATE_VOLTAGES[4][0];
    ch_voltage[1] = STATE_VOLTAGES[4][1];
    ch_voltage[2] = STATE_VOLTAGES[4][2];
    ch_voltage[3] = STATE_VOLTAGES[4][3];
  }




}//----------------------------------------------------------------




//------------------------------------------------------------------(4)
void writeState (int channel, int CH_PIN_ARRAY[], int digitalSig[],
int SIG_PIN_ARRAY[]) {
  //write channel # and voltage value to the PolCont
  digitalWrite(RW,LOW); //load data

  if (channel == 0){
  //------- CH = 0 ----------
    digitalCh[0] = 0;
    digitalCh[1] = 0;
  }else if (channel == 1){
  //------- CH = 1 ----------
    digitalCh[0] = 1;
    digitalCh[1] = 0;
  }else if (channel == 2){
  //------- CH = 2 ----------
    digitalCh[0] = 0;
    digitalCh[1] = 1;
  }else if (channel == 3){
  //------- CH = 3 ----------
    digitalCh[0] = 1;
    digitalCh[1] = 1;
  }

  for (int i = 0; i < 2; i++) { //CHANNEL
    if (digitalCh[i] == 0) {
      digitalWrite(CH_PIN_ARRAY[i], LOW);
    }
    else if (digitalCh[i] == 1) {
      digitalWrite(CH_PIN_ARRAY[i], HIGH);
    }
  }
  for (int i = 0; i < 12; i++) { //VOLTAGE
    if (digitalSig[i] == 0) {
      digitalWrite(SIG_PIN_ARRAY[i], LOW);
    }
    else if (digitalSig[i] == 1) {
      digitalWrite(SIG_PIN_ARRAY[i], HIGH);
    }
  }

  digitalWrite(CS, LOW);
  //for (int i = 0 ; i < 150000000; i++) {}; //DELAY
  digitalWrite(CS, HIGH);



}//--------------------------------------------------------------




//---------------------------------------------------------------(5)
void setNewVoltages(String serialMessage) {

  int state = serialMessage.substring(12).toInt(); //   -> state #

  // assumi tens es V.V (3char)
  int V1 = serialMessage.substring(14, 17).toInt();
  int V2 = serialMessage.substring(18, 21).toInt();
  int V3 = serialMessage.substring(22, 25).toInt();
  int V4 = serialMessage.substring(26, 29).toInt();

  int newVoltages[] = {V1, V2, V3, V4};

  for (int i ; i < 4 ; i++) {

    STATE_VOLTAGES [i][state] = round((newVoltages[i] * 4096) / 150);

  }
  // -> STATE / V1 / V2 / V3 / V4 / NEWVOLTAGES [] / STATE_VOLTAGES [][]
}//--------------------------------------------------------------
```

```
//------------------------------------------------------------------(6)
void emptyBuffer(int statesBuffer[]) {

  for (int i = 0 ; i < sizeof(statesBuffer) ; i++) {
    statesBuffer[i] = 0;
  }
}//-----------------------------------------------------------------



//------------------------------------------------------------------(7)
void writeToBuffer(int marker, int statesBuffer[], int newStates[]) {

  //if (marker == BUFFER_SIZE/4) {
  //puts in the "second part" of the buffer
  if (marker == 0) {
  //puts in the "second part" of the buffer        -- ALICE
  //if (marker == 10) {
  //puts in the "second part" of the buffer     -- BOB
    for (int j = BUFFER_SIZE / 2 ; j < BUFFER_SIZE ; j++) {
      statesBuffer[j] = newStates[j - BUFFER_SIZE/2];
    }
  }
  //else if (marker == BUFFER_SIZE*3/4) {
  //puts in the "1 part" of the buffer
  else if (marker == BUFFER_SIZE/2) {
  //puts in the "1 part" of the buffer               -- ALICE
  //else if (marker == (BUFFER_SIZE/2)+10) {
  //puts in the "1 part" of the buffer       -- BOB

    for (int j = 0 ; j < BUFFER_SIZE / 2 ; j++) {
      statesBuffer[j] = newStates[j];
    }
  }

  marker = 0;
}//-----------------------------------------------------------------



//------------------------------------------------------------------(8)
void setUp(int CH_PIN_ARRAY[], int SIG_PIN_ARRAY[], int digitalSig[]) {
```

```
//------------------ CH = 0 ------------------
    digitalCh[0] = 0;
    digitalCh[1] = 0;

    for (int i = 0; i < 2; i++) { // WRITES CHANNEL # TO PC
      if (digitalCh[i] == 0) {
        digitalWrite(CH_PIN_ARRAY[i], LOW);
      }
      else if (digitalCh[i] == 1) {
        digitalWrite(CH_PIN_ARRAY[i], HIGH);
      }
    }

    //PC1 - 208
    //PC2 - 209
    convertBinary (0, digitalSig); // converts the voltage value to
    binary

    for (int i = 0; i < 12; i++) { // WRITES VOLTAGE VAL TO PC
      if (digitalSig[i] == 0) {
        digitalWrite(SIG_PIN_ARRAY[i], LOW);
      }
      else if (digitalSig[i] == 1) {
        digitalWrite(SIG_PIN_ARRAY[i], HIGH);
      }
    }

    digitalWrite(CS, LOW);
    delayMicroseconds(1);
    digitalWrite(CS, HIGH);


//------------------ CH = 1 ------------------
    digitalCh[0] = 1;
    digitalCh[1] = 0;

    for (int i = 0; i < 2; i++) { // WRITES CHANNEL # TO PC
      if (digitalCh[i] == 0) {
        digitalWrite(CH_PIN_ARRAY[i], LOW);
      }
      else if (digitalCh[i] == 1) {
        digitalWrite(CH_PIN_ARRAY[i], HIGH);
      }
    }
    for (int i = 0; i < 12; i++) { // WRITES VOLTAGE VAL TO PC
      digitalWrite(SIG_PIN_ARRAY[i], LOW);
    }
    digitalWrite(CS, LOW);
    delayMicroseconds(1);
    digitalWrite(CS, HIGH);


//―――――――――――――――― CH = 2 ――――――――――――――――
    digitalCh[0] = 0;
    digitalCh[1] = 1;

    for (int i = 0; i < 2; i++) { // WRITES CHANNEL # TO PC
      if (digitalCh[i] == 0) {
        digitalWrite(CH_PIN_ARRAY[i], LOW);
      }
      else if (digitalCh[i] == 1) {
        digitalWrite(CH_PIN_ARRAY[i], HIGH);
      }
    }

    for (int i = 0; i < 12; i++) { // WRITES VOLTAGE VAL TO PC
      digitalWrite(SIG_PIN_ARRAY[i], LOW);
    }

    digitalWrite(CS, LOW);
    delayMicroseconds(1);
    digitalWrite(CS, HIGH);


//―――――――――――――――― CH = 3 ――――――――――――――――
    digitalCh[0] = 1;
    digitalCh[1] = 1;

    for (int i = 0; i < 2; i++) { // WRITES CHANNEL # TO PC
      if (digitalCh[i] == 0) {
        digitalWrite(CH_PIN_ARRAY[i], LOW);
      }
      else if (digitalCh[i] == 1) {
        digitalWrite(CH_PIN_ARRAY[i], HIGH);
      }
    }

    for (int i = 0; i < 12; i++) { // WRITES VOLTAGE VAL TO PC
      digitalWrite(SIG_PIN_ARRAY[i], LOW);
```

```
        }

        digitalWrite(CS, LOW);
        delayMicroseconds(1);
        digitalWrite(CS, HIGH);
}//----------------------------------------------------------------


//######################################################################
```