Universidad Autónoma de Querétaro

Facultad de Ingeniería

"Diseño y construcción de un experimento automatizado de Criptografía Cuántica basado en el protocolo BB84"

T E S I S

para obtener el grado de:

Ingenieros Físicos

presentan:

Cristopher Rojas Aguado
Abril Montserrath Vargas Jiménez

Director de tesis

Dr. Neil Vladimir Corzo Trejo

Santiago de Querétaro, Querétaro, Marzo 2023

Diseño y construcción de un experimento automatizado de Criptografía Cuántica basado en el protocolo BB84

por

Cristopher Rojas Aguado

Abril Montserrath Vargas Jiménez

"Soy de las que piensan que la ciencia
tiene una gran belleza. Un científico
en su laboratorio no es sólo un técnico:
también es un niño colocado ante
fenómenos naturales que lo impresionan
como un cuento de hadas".

- Marie Curie

"Me he dado cuenta que incluso las
personas que dicen que todo está
predestinado y que no podemos hacer nada
para cambiar nuestro destino, siguen mirando
a ambos lados antes de cruzar la calle".

- Stephen Hawking

Resumen

Se presenta la construcción, desarrollo y automatización de un prototipo experimental que simula un experimento de criptografía cuántica basado en el protocolo BB84. Este protocolo es uno de los más efectivos y sencillos de implementar en la criptografía cuántica debido a que utiliza la polarización de la luz para codificar los bits de información. El experimento utiliza dos canales de comunicación: un canal cuantico de espacio libre de 60 cm para el envío de los bits y un canal clásico *Bluetooth* para la comunicación entre el emisor (Alice) y el receptor (Bob). Con la implementación del protocolo BB84 se genera una llave secreta que permite cifrar el mensaje y enviarlo de forma segura. Con el prototipo experimental automatizado se logró reducir el tiempo para la creación de la llave en un 84 % y con un costo ligeramente superior, comparado con el kit manual *EDU-QCRY1* de Thorlabs. Nuestro prototipo experimental es una plataforma realista para introducirse al estudio de la criptografía cuántica. Dicha plataforma logra demostrar los principios y conceptos presentes en la criptografía cuántica transmitiendo información privada de forma segura. Como futuros trabajos, se pueden agregar fuentes y detectores de un sólo fotón para obtener un sistema realmente cuántico o implementar otros protocolos de seguridad.

(**Palabras clave:** Criptografía Cuántica, Protocolo BB84, Polarización de la luz, Llave Secreta, Experimento Automatizado)

Abstract

We present the construction, development and automation of an experimental prototype that simulates a quantum cryptography experiment based on the BB84 protocol. This protocol is one of the most effective and simple to implement in Quantum Cryptography because it uses the polarization of light to encode information bits. The experiment used two communication channels: a 60 cm-long free-space quantum channel to send the bits and a Bluetooth channel for classical communication between the sender (Alice) and receiver (Bob). With the implementation of the BB84 protocol, a secret key is generated which allows the message to be encrypted and sent safely. With the automated prototype, the time required to create the key was reduced by 84 % compared with the manual kit *EDU-QCRY1* from Thorlabs at a slightly higher cost. Our prototype is an affordable realistic platform to initiate the study of quantum cryptography. Our experimental prototype demonstrated the principles and concepts presented in quantum cryptography by securely transmitting private information. For future work, single-photon sources and detectors can be added to obtain a truly quantum system or implement other security protocols.

(**Key Words:** Quantum Cryptography, BB84 Protocol, Light Polarization, Secret Key, Automated Experiment)

Agradecimientos

Al Dr. Neil Vladimir Corzo Trejo y a la Dra. Karina Jiménez García por su valioso apoyo y consejos a lo largo de la carrera, por ayudarnos a mejorar académicamente y personalmente, por permitirnos trabajar con ellos en el laboratorio de tecnologías cuánticas y aprender de sus conocimientos. También por permitirnos ser parte de este proyecto el cuál contribuirá a ampliar el conocimiento de las tecnologías cuánticas a la sociedad. También agradecemos al Dr. José Mauricio López Romero por permitirnos trabajar en las instalaciones del Cinvestav Querétaro y ser parte de este proyecto.

A Elías León Ángeles por brindarnos su apoyo y conocimientos a lo largo de la construcción, el diseño y la elaboración del experimento, así como de todos lo buenos momentos que pasamos a su lado. A Karen Rojas Aguado por ayudarnos con el diseño y elaboración de las imágenes.

A todos los profesores de la Universidad Autónoma de Querétaro que nos apoyaron en diferentes ámbitos a lo largo de la carrera. Principalmente a los profesores de Ingeniería Física que nos acompañaron a lo largo de toda la carrera brindándonos gran parte de sus conocimientos para afrontar los diversos retos que surgieron en nuestro día a día. De la misma forma queremos agradecer a nuestros compañeros y amigos que nos apoyaron tanto académica como personalmente y con los cuales salimos adelante.

A la colaboración de CONCYTEQ y CINVESTAV Querétaro por los cuales obtuvimos los recursos y materiales necesarios para el desarrollo del prototipo experimental. Así como al CECEQ por permitir instalar el prototipo experimental en en el museo de ciencia para que nuevas generaciones puedan aprender sobre las tecnologías cuánticas y en específico de la criptografía cuántica.

Por último, pero no menos importante, agradecemos a todos nuestros familiares, amigos y aquellas personas importantes que nos hemos encontrado y que nos han apoyado, nos han levantado cada vez que caemos, nos han inspirado a ser mejores y sobre todo nos han ayudado a superarnos y a luchar por cada uno de nuestros sueños.

Agradecimientos personales:

Abril:

A mis padres, por enseñarme a ser valiente y nunca darme por vencida. Gracias por su paciencia y apoyo incondicional. A mis abuelos, por todos sus consejos y amor. A mis hermanas por ser tan alegres y siempre sacarme una sonrisa. Al resto de mi familia por siempre mostrarme su apoyo y creer en mí.

Al Dr. Neil y la Dra. Karina por su valiosa guía y conocimientos y por ayudarme a encontrar mi pasión en el campo de las tecnologías cuánticas. A mis maestros de la universidad por cultivar en mí el amor por la ciencia y la física y tener curiosidad por el mundo. A mis compañeros de clase, que hicieron que mi paso por la universidad fuera más ameno y con los cuales compartí muchos momentos de felicidad.

A mis amigos de toda la vida Abby, Roberto, Edgar, Emmanuel, que a pesar del tiempo y la distancia siempre me han brindando su apoyo y cariño y son como parte de mi familia. A mi amigo José Carlos, que siempre tuvo la disposición de ayudarme con las dudas en electrónica y por su amabilidad para compartir conmigo sus conocimientos.

A mi compañero de tesis, Cris, por siempre brindarme su apoyo, conocimientos y amistad incondicional y por motivarme a ser mejor persona. Realizar este proyecto no hubiera sido tan increíble de no ser por él, gracias de verdad.

Cristo:

A mis padres, por darnos todo lo que necesitábamos, por haberme apoyado en las decisiones que tomaba y por todo el cariño y amor que me brindaron. De igual forma a mis hermanos, que me brindan mucho cariño y amor y me dieron lecciones para ser mejor en la vida. A mis abuelos o mis otros padres, los cuales me dieron bastante amor desde niño, se dedicaron a hacerme feliz y me apoyaban en cada decisión.

Al Dr. Neil y la Dra. Karina porque desde que llegue sin saber muchas cosas me brindaron su apoyo para trabajar con ellos e incrementar mis conocimientos. Me han apoyado y guiado hasta donde han podido y han despertado en mí el amor por estudiar las tecnologías cuánticas.

A Jose Juan, Erick, Luis Gerardo, Juan Carlos, Guillermo, David e Isabel los cuales me brindaron muchos momentos de felicidad y estuvieron conmigo apoyándome cuando los necesitaba. A Patricia, quien compartió bastante tiempo de su vida conmigo y me enseñó a luchar por lo que quería y a ser una mejor persona. A mi mejor amigo Rodrigo, quien me ha enseñado a vivir la vida, me ha impulsado a ser mejor con cada paso que doy y sobre todo, me ha demostrado el cariño de una amistad.

A mi compañera Abril, porque a pesar de que nos conocimos hace poco, me ha enseñado bastantes cosas de la vida y me ha impulsado a mejorar académicamente y personalmente. Además, es una persona a la que estimo mucho y con la cual pude cumplir un sueño muy importante para mí.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1
Introducción

Hoy en día se necesita una mayor cantidad de dispositivos conectados a internet para realizar nuestras actividades diarias. Constantemente se generan datos e información a través aplicaciones, sitios de internet, redes sociales, etc. De acuerdo al año 2020, se estima que se descargaron más de 71500 millones de aplicaciones en todo el mundo, se enviaron 306400 millones de correos electrónicos y 18700 millones de mensajes de texto cada día (Vuleka, 2021). Mucha de la información que se comparte y se genera es carácter confidencial, por ejemplo al realizar una compra en línea, una transacción bancaria o al compartir información personal. Se requiere tener la certeza de que nuestra información se comparte de forma segura y para realizar este proceso de forma confiable es necesario hacer uso de la criptografía, que junto con la seguridad informática juegan un papel muy importante en nuestra sociedad de hoy en día.

La criptografía se define como el arte de codificar un mensaje para que sólo la persona a quien va dirigido pueda leerlo (Fox, 2006) ó como una técnica que se encarga de cifrar la información y que ésta pueda ser transmitida de manera segura (Boscá Díaz-Pintado, 2017). Se puede decir entonces que la criptografía es el método para compartir información confidencial donde los únicos que tienen acceso a dicha información son el emisor y receptor.

En criptografía se utilizan algoritmos de cifrado que consisten en combinar el mensaje secreto con información adicional conocida como "llave" o "clave" de cifrado, lo que produce un criptograma o mensaje cifrado. Un mensaje cifrado es imposible de

leer sin la clave (Gisin, Ribordy, Tittel, y Zbinden, 2002). Si un espía intercepta el canal de comunicación y obtiene el mensaje cifrado sin tener la clave, obtendría simplemente un conjunto de letras o símbolos sin sentido.

Es difícil situarse en una fecha exacta sobre los principios de la criptografía, sin embargo uno de los primeros métodos de encriptación es el método por sustitución donde cada letra del mensaje original se sustituye por otra letra o símbolo (Davies, 1997). El ejemplo más conocido de este tipo es el cifrado "César". Este cifrado consiste en desplazar cada letra del mensaje "n" posiciones hacia adelante o atrás en el abecedario para obtener el mensaje cifrado (Salam y Al Salah, 2015). Por ejemplo, si n=3, entonces la letra A se sustituye por la letra D, B por E y así sucesivamente. Un mensaje cifrado con este método es fácil de descifrar tomando en cuenta que el número de opciones para desplazar el mensaje se reduce a las 27 letras del alfabeto español y por lo tanto sólo se necesitan probar todas las combinaciones posibles hasta obtener un mensaje coherente.

Otro cifrado importante surgió en Francia (siglo XVI), el cifrado Vigenère. Este cifrado requiere de una clave y un alfabeto, y consiste en agregar la clave al texto original a través de una operación matemática como la suma (se suman los valores de las letras de acuerdo a su posición). Este método tenía el problema que al analizar el mensaje cifrado se podían detectar ciertos patrones, lo que permitía encontrar la clave y descifrar el mensaje. Así surgió el criptoanálisis, el cual se basaba en analizar las frecuencias de los mensajes cifrados. El criptoanálisis dejó obsoletos varios métodos de cifrado haciendo importante el crear sistemas más seguros de encriptación (Davies, 1997).

La repetición de patrones llevo a la conclusión de que se necesitaba incluir el factor aleatorio en los métodos de cifrado, evitando así dejar pistas o patrones reconocibles en los códigos que pudieran ser evidentes para el criptoanálisis. Esto estableció dos formas diferentes para los métodos de cifrado: los "Sistemas Simétricos o de Clave privada" y los "Sistemas Asimétricos o de Clave pública" (Simmons, 1979). Las características principales de estos sistemas son:

1. **Sistema simétrico**: En este sistema se utiliza una clave para cifrar y descifrar el mensaje, la cual comparten previamente el emisor y receptor. Para esto se pueden realizar reuniones privadas en las cuales los interlocutores establecen la clave que utilizarán hasta su próxima reunión, o bien, comparten un "libro de claves" (en inglés *One-Time-Pad* (Rueppel, 1986)) con caracteres aleatorios para cifrar los mensajes. Para encriptar y desencriptar el mensaje primero se elige una página del libro de claves, la cual es conocida entre el emisor y receptor. Cada página del libro se utiliza una sola vez y al finalizar el libro es necesario volver a generar uno nuevo, ya que por seguridad no es posible reutilizar las páginas (Fox, 2006).

   Utilizar un libro de claves conduce principalmente a dos obstáculos. El primero consiste en distribuirlo de forma segura y garantizar que nadie más tenga acceso a él, ya que existe la posibilidad de que un espía obtenga una copia del libro que le permita descifrar los mensajes. El segundo problema consiste en que la clave debe ser tan larga como el mensaje a cifrar. De no ser así, se genera la repetición de patrones que pueden ser detectados por un espía.

2. **Sistema asimétrico**: En este cifrado se utilizan dos claves para el cifrado, una pública y una privada, tal y como lo hace el sistema RSA creado por Rivest, Shamir y Adleman en 1978. La ventaja de utilizar una clave pública es que puede ser distribuida por medio de un canal de comunicación que no necesariamente es un canal privado y cualquier persona puede tener acceso a él. Sin embargo la clave va encriptada y al escucharla una persona indeseada sólo escucharía "ruido" (Rivest, Shamir, y Adleman, 1978).

   Este método utiliza las llamadas "*trap-door one-way function*" (funciones trampa de un sólo sentido) las cuales se consideran unidireccionales porque es fácil implementarlas en un sentido pero significativamente difícil calcular su función inversa (Diffie y Hellman, 2019). El sistema RSA se basa en el problema matemático de que el tiempo necesario para encontrar los factores primos (clave privada) de un número entero "muy grande" (clave pública) aumenta exponencialmente con el número de dígitos (Rivest y cols., 1978). En este caso la clave pública es el producto de dos números primos muy grandes y para desencriptar el mensaje

se necesita realizar el proceso inverso, es decir, factorizar la clave pública. En principio, no es posible deducir la clave privada a partir de la clave pública, ya que el tiempo de cómputo para realizar la factorización supera la capacidad de procesamiento de los ordenares actuales. Sin embargo, en 1994 Peter Shor creó un algoritmo que puede vulnerar los métodos "*trap-door one-way function*", y para ser implementado requiere de una computadora cuántica con cierto número de *qubits*[1]. De poder contar con dicha computadora cuántica se estima que podría resolver este problema matemático en segundos, por lo que este tipo de cifrado podría ser obsoleto en un futuro cercano (Shor, 1994).

Gracias al entendimiento de la mecánica cuántica se pueden utilizar las propiedades de los sistemas cuánticos para manipular y transferir información de forma segura. En esto se basa la criptografía cuántica para distribuir la clave de forma segura y surge como una solución a los problemas presentes en criptografía clásica (Nielsen y Chuang, 2010).

La criptografía cuántica se ha desarrollado experimentalmente tanto en centros de investigación como en el sector privado. Algunas compañías que proporcionan sistemas comerciales de criptografía cuántica son: *Thorlabs* (Thorlabs, 2017), *ID Quantique* (ID Quantique, 2022), *MagiQ Technologies* (MagiQ Technologies, 2004), *Toshiba* (Toshiba, 2004) y *Quintessense labs* (Quintessence Labs, 2022). Estos sistemas son adquiridos principalmente por compañías o instituciones que requieren de un sistema especializado de comunicación segura, pero su costo es elevado.

A partir de esto surge la motivación de la tesis, ya que actualmente existen sistemas comerciales automatizados de criptografía cuántica que son demasiado costosos o requieren de procesos especiales para su manejo adecuado, siendo poco didácticos. También se encuentran sistemas manuales tal como el Kit de demostración de criptografía cuántica EDU-QCRY1 de Thorlabs (kit comercial) (Thorlabs, 2017) que es para uso educativo, sin embargo su costo también es elevado y requiere demasiado tiempo realizar cada prueba experimental debido a que es un sistema manual.

[1]Un *qubit* es un sistema cuántico de dos estados. El *qubit* puede tomar el valor de "0", "1" ó un estado superpuesto de ambos. Es la unidad básica para almacenar o registrar procesos en una computadora cuántica (Sutor, 2019)

La construcción, desarrollo e implementación del software y hardware de un prototipo experimental automatizado de criptografía cuántica, basado en el protocolo BB84, permitirá a diversas instituciones y centros de investigación apoyarse en este documento para construir su propio prototipo experimental con fines académicos, de investigación y/o de divulgación de la ciencia. Este prototipo tiene como características ser un sistema eficaz, rápido y de bajo costo comparado con los experimentos automatizados de fibra óptica y con el kit comercial de Thorlabs. Esto contribuirá a extender el área de la ciencia dedicada a la criptografía cuántica experimental, aportar conocimientos que aún se encuentran en desarrollo en esta área de estudio y realizar demostraciones divulgativas de los diferentes conceptos físicos que se desarrollan en este experimento.

La motivación de este un proyecto es contribuir al crecimiento y desarrollo científico, poniendo a disposición del público general un prototipo experimental con el que puede interactuar y entender conceptos fundamentales de la Física y su aplicación en el desarrollo de Tecnologías Cuánticas. Así mismo permitirá el fomento de vocaciones científicas en los jóvenes, así como la creación de recursos humanos para las tecnologías futuras, dando a conocer la importancia y las principales aplicaciones de las Tecnologías Cuánticas en nuestra actualidad. Este proyecto será de innovación en el tema de comunicaciones, pues muestra las bases para el desarrollo de un sistema de Comunicación Cuántica desarrollado en el Estado de Querétaro, lo que contribuirá a la consolidación del Ecosistema de Ciencia, Tecnología e Innovación del Estado.

Este proyecto forma parte de la colaboración CONCYTEQ - Cinvestav Querétaro y cuenta con los recursos y materiales necesarios para el desarrollo un prototipo experimental de divulgación científica enfocados en criptografía cuántica. El prototipo desarrollado es actualmente exhibido en el Museo de Ciencia del Péndulo de Foucault del CECEQ dentro de la Exposición permanente “Al Encuentro del Saber” ubicado en el centro educativo cultural del estado de Querétaro. Debido a ello, este proyecto tiene una utilidad práctica y metodológica mediante la generación de nuevo conocimiento con la propuesta experimental y es de relevancia social, puesto que el experimento podrá fomentar las vocaciones científicas en ciencias exactas y tecnología y contribuir a la divulgación científica de estas nuevas tecnologías a la población en general.

Capítulo 2
Antecedentes

Se ha mencionado anteriormente que los sistemas de encriptación clásicos tienen algunos obstáculos, ya sean de clave pública o privada. Los sistemas de clave pública serán obsoletos con el desarrollo de una computadora cuántica capaz de implementar algoritmos matemáticos que permitirán calcular las funciones unidireccionales con una increíble facilidad, tal como el algoritmo de Shor (Shor, 1994). Por otra parte, los sistemas de clave privada serían un sistema seguro a largo plazo si se garantiza que la clave sea aleatoria, sea tan larga como el mensaje, se genere una clave distinta para cada mensaje y se comparta de forma segura. Esto fue probado por Shannon (1949) donde menciona que el sistema de cifrado *One-Time-Pad* es teóricamente seguro, es decir cumple con tres de los requerimientos. Sólo es necesario de un método que garantice que la distribución de la clave sea totalmente segura.

Es aquí donde entra la criptografía cuántica, la cual utiliza las leyes de la mecánica cuántica para detectar a cualquier persona indeseada y pueda garantizar que la distribución de la clave sea totalmente segura. A continuación se presentan algunos de los conceptos básicos en criptografía cuántica, así como algunas aplicaciones prácticas y sus avances en la actualidad.

2.1 Distribución de Claves Cuánticas (*QKD*)

En la Figura 2.1 se muestra un diagrama con los personajes[1] y las partes principales de un sistema de comunicación clásico basado en ondas de radio. Alice transmite su mensaje mediante las ondas de radio a través del espacio libre (canal de transmisión) utilizando una torre de radio (fuente de señal). Bob es el receptor de información, que también cuenta con una torre receptora para detectar las señales de radio. Eve es la espía en el canal de comunicación que intercepta las señales de radio para robar el mensaje de Alice. Es posible utilizar distintos tipos de canales y fuentes para el envío de información, tal como fibra óptica o el espacio libre con fotones del cual se hablará más adelante.

Figura 2.1: Representación de un sistema de comunicación clásico. La PC del lado izquierdo representa al emisor "Alice" y a la derecha se representa al receptor "Bob". Ambos transmiten la información encriptada por medio de un canal de transmisión de ondas en el espacio libre. Este es un canal inseguro, en el que "Eve", puede obtener la información sin que los interlocutores se den cuenta de la invasión (Elaboración propia).

[1] Alice y Bob son dos personajes ficticios en criptografía mencionado por primera vez en el trabajo de Rivest y cols. (1978). Alice representa el emisor, Bob el receptor y Eve la espía *(eavesdropper)*, por su significado en inglés: *secretly listen to a conversation.*

A diferencia de un sistema de comunicación clásico donde la información se codifica en una propiedad clásica como las ondas de radio, en un sistema de comunicación cuántico la información se codifica en las propiedades de un sistema cuántico, lo cual permite garantizar que el envío de la información en el canal de comunicación sea seguro. Para garantizar esto, se utilizan los protocolos de Distribución de Claves Cuánticas (*QKD*, por sus siglas en inglés *Quantum Key Distribution*). El objetivo de estos protocolos es generar una clave secreta entre dos partes de forma segura y son las leyes de la mecánica cuántica las que garantizan la seguridad del protocolo (Wolf, 2021). El origen de la *QKD* se debe a Stephen Wisner, quien propuso la idea de aplicar las relaciones de indeterminación que están ligadas con las propiedades complementarias de un sistema cuántico a un sistema de comunicación (Wiesner, 1983). Aunque el primer experimento de *QKD* se le atribuye a Charles Bennett y Gilles Brassard los cuales presentaron un protocolo seguro para detectar la presencia de un espía en el canal de comunicación y así garantizar el envío seguro de la clave (Bennett, Bessette, Brassard, Salvail, y Smolin, 1992).

Para enviar un mensaje de forma segura primero es necesario generar una clave que permita cifrar el mensaje. Así, aunque la información fuese interceptada por alguien indeseado, no podría saber su contenido debido a que se encuentra cifrado. Una clave segura debe de cumplir con los siguientes requisitos (ID Quantique, 2020):

1. **Clave aleatoria:** La clave debe ser aleatoria de tal forma que no existan repeticiones o patrones que Eve pueda adivinar o deducir.

2. **Clave confidencial:** La clave no puede ser interceptada u obtenida por Eve en ningún momento.

El proceso de *QKD* se puede definir como un método que permite a dos usuarios generar una clave privada. Sin embargo *QKD* no evita que Eve interfiera en el canal de comunicación, obteniendo parte o toda la información enviada. En el caso de que Eve robe la información, esta será modificada. Esta alteración puede ser detectada por los interlocutores y determinar si el canal de comunicación es seguro. Una vez que el canal es seguro, se crea la clave con la cual se cifra la información privada para posteriormente ser transmitida por un canal de comunicación clásico (Bennett y Brassard, 1984).

La seguridad de *QKD* está dada por las leyes y principios de la mecánica cuántica. En el caso de que Eve quiera interferir en la comunicación sin que se den cuenta los interlocutores, necesita medir la señal de Alice, extraer la información, hacer una copia exacta y enviarla a Bob para pasar desapercibida. Esto es posible clásicamente, ya que bastaría un dispositivo que mida y replique la información obtenida por Eve. La ventaja de la criptografía cuántica es que al codificar la información en un estado cuántico, las leyes de la mecánica cuántica impiden realizar la medición y obtener una copia exacta sin modificar su estado original. Esta propiedad de los sistemas cuánticos es lo que se conoce como Teorema de No Clonación (Wootters y Zurek, 1982). Por ejemplo, si Eve realiza la medición y extrae la información de la partícula, estará modificando la información que Alice envió originalmente, por lo que es posible que Bob no obtenga la información correcta. Esta modificación que introduce la espía es inevitable y puede ser detectada por Alice y Bob.

Existen diversos protocolos *QKD* para generar la clave de forma segura. Entre los protocolos más conocidos están:

- **Protocolo BB84:** Se basa en la codificación de información en el estado de los fotones[2] individuales polarizados aleatoriamente en dos pares de direcciones de polarización ortogonales. Estos dos estados pueden ser horizontal - vertical ($\rightarrow$, $\uparrow$) ó circular izquierda - circular derecha ($\circlearrowleft$ y $\circlearrowright$). En un principio, los fotones individuales eran difícil de producir experimentalmente, por lo que la primera implementación del protocolo se realizó utilizando pulsos atenuados propagándose en el espacio libre (Bennett y Brassard, 1984).

- **Protocolo BB92:** Este protocolo es similar al protocolo BB84, pero es más simple ya que se utilizan únicamente uno de los dos estados de cada par de polarización para enviar y detectar la información. Debido a esto se descartan más *bits*[3] de información (Bennett, 1992).

[2]Se puede definir al fotón como la partícula de luz que viaja en línea recta desde una fuente. Ésta partícula poseen la característica de comportarse como onda o partícula dependiendo del fenómeno en el que se encuentre involucrada (Pedrotti, Pedrotti, y Pedrotti, 2017).

[3]La unidad mínima para almacenar o registrar procesos en una computadora clásica se llama *Bit* y toma únicamente dos valores posibles "0" ó "1".

- **Protocolo B91:** Este protocolo fue propuesto por Artur Ekert en 1991 en el cual se utilizan pares de fotones entrelazados[4]. Estos pares de fotones primero deben de prepararse y después se distribuyen a los interlocutores. Una vez creados los pares de fotones entrelazados, cada interlocutor mide sus fotones con polarizaciones aleatorias. Cuando ambos coinciden en la orientación se obtiene una medición correcta, pero cuando eligen orientaciones diferentes entonces obtienen un 50 % de probabilidad de obtener una medición correcta, por lo que estos últimos son desechados (Ekert, 1992). Este tipo de sistema tiene una mayor seguridad, debido a que es posible medir las propiedades cuánticas de la otra partícula a partir de la primera. Esto significa que si una de las partículas es medida o alterada en el camino por algún espía es posible detectarlo (Aczel, 2002).

Hasta ahora se ha hablado de los objetivos, características y principios de la criptografía cuántica. A continuación se dará una breve descripción de algunos conceptos básicos como: *One-Time-Pad*, *Qubit*, Polarización, Teorema de no clonación, Protocolo BB84, Autenticación, entre otros necesarios para cifrar el mensaje, crear la clave cuántica y transmitir la información de forma segura.

2.2 Cifrado *One-Time-Pad*

El proceso de *QKD* es un método seguro para la distribución de la clave, pero se requiere de un método de cifrado para generar una clave. Para que una clave sea segura debe de cumplir con ciertos requisitos como: clave aleatoria, al menos tan larga como la del texto a cifrar y utilizarla una sola vez para cada mensaje. Un método de cifrado que cumple con estos requerimiento es el cifrado *One-Time-Pad* (Shannon, 1949).

El cifrado *One-Time-Pad* (ó libreta de un sólo uso) se basa en la propuesta de Gilbert Vernam durante la Primera Guerra Mundial (Vernam, 1926). En este cifrado, el remitente y el receptor comparten un libro de claves, el cual contiene una secuencia

[4]El fenómeno de entrelazamiento cuántico (en inglés *quantum entanglement*) correlaciona dos partículas, del tal forma que dejan de actuar de forma independiente. Es decir, lo que le pase a una le sucederá inmediatamente a la otra. Medimos una y conocemos la otra. Fue predecido en 1935 por Einstein, Podolsky y Rosen (Einstein, Podolsky, y Rosen, 1935).

aleatoria de *bits* (0 y 1) que es tan larga como el mensaje en sí (Fox, 2006). Antes de sumar la clave aleatoria al texto plano (el mensaje con la información confidencial) se necesita convertir el mensaje a una secuencia de *bits*. Para transformar una cadena de texto a dígitos binarios, existen varios protocolos o algoritmos, tal como el *American Standard Code for Information Interchange*, o mejor conocido como *ASCII* (Singh, 2001). De acuerdo a este protocolo, se asignan ocho dígitos binarios a cada letra del alfabeto. Esto nos permite identificar cada letra o símbolo con una secuencia única de ceros y unos. Una vez que el mensaje se ha transformado a su representación binaria, comienza el proceso de cifrado. Para esto basta con sumar el mensaje binario con la clave aleatoria de 0's y 1's para producir una nueva cadena de *bits* y obtener el mensaje cifrado. Luego, el receptor tiene que restar la clave al mensaje cifrado para decodificar el mensaje (Fox, 2006). La suma o resta de binario se realiza mediante la suma *XOR* donde se obtiene un 0 en caso de sumar dos entradas iguales y 1 en caso contrario, observe la Tabla 2.1.

Tabla 2.1: Suma *XOR*. *XOR* es una puerta lógica digital en la que se define al 0 como una entrada falsa y al 1 como una entrada verdadera. Entonces, para dos entradas verdaderas o falsas se obtiene una salida falsa, mientras que si una y sólo una de las entradas es falsa, se obtiene una salida verdadera (Fletcher, 1997).

Entrada A	Entrada B	Salida $A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

A continuación, se muestra un ejemplo simplificado de como funciona el proceso de cifrado y descifrado con cinco *bits*. Primeramente, a cada letra del alfabeto le es asignado un número, iniciando desde 0, entonces A=0, B=1, C=2 etc. Luego, se necesitan 5 *bits* para poder expresar cualquier número entre el 0 y el 25. Luego 0=00000, 1=00001, 2=00010 y así sucesivamente, ver Tabla 2.2.

Tabla 2.2: Ejemplo de asignación de un valor decimal a binario de cinco *bits* para cada letra del alfabeto inglés. Este es el método más simple de asignación, pero es posible asignar un valor diferente o asignar un valor dinámico, todo depende del arte de encriptar (Elaboración propia).

Letra Alfabeto	Asignación numérica	Representación binaria	Letra Alfabeto	Asignación numérica	Representación binaria
A	0	0 0 0 0 0	N	13	0 1 1 0 1
B	1	0 0 0 0 1	O	14	0 1 1 1 0
C	2	0 0 0 1 0	P	15	0 1 1 1 1
D	3	0 0 0 1 1	Q	16	1 0 0 0 0
E	4	0 0 1 0 0	R	17	1 0 0 0 1
F	5	0 0 1 0 1	S	18	1 0 0 1 0
G	6	0 0 1 1 0	T	19	1 0 0 1 1
H	7	0 0 1 1 1	U	20	1 0 1 0 0
I	8	0 1 0 0 0	V	21	1 0 1 0 1
J	9	0 1 0 0 1	W	22	1 0 1 1 0
K	10	0 1 0 1 0	X	23	1 0 1 1 1
L	11	0 1 0 1 1	Y	24	1 1 0 0 0
M	12	0 1 1 0 0	Z	25	1 1 0 0 1

Por ejemplo, para codificar la palabra "UAQ" se considera que $U = 20 = 10100$, $A = 0 = 00000$, $Q = 16 = 10000$, por lo que su representación en binario es la concatenación de las cifras anteriores: UAQ = 101000000010000. Luego, se genera una clave aleatoria de 0's y 1's tomando en cuenta que debe ser tan larga como el mensaje que se desea cifrar, por ejemplo Clave = 110001101010011. Finalmente, se aplica la suma al mensaje binario con la clave como muestra en la Tabla 2.3 (Singh, 2001). Si una persona obtiene el mensaje encriptado y agrupa cada 5 *bits* obtendrá 3 grupos, el Grupo azul = 01100, el Grupo morado = 11010 y el Grupo rosa = 00011, ver Tabla 2.3. Al convertir cada grupo a su letra correspondiente sin antes aplicar la clave, obtendrá como resultado: Grupo azul = M, el Grupo morado = No existe, Grupo rosa = D,

por lo cual es imposible descifrar un mensaje sin la clave, ya que sólo la clave permite obtener el mensaje original (Fox, 2006). Este es un ejemplo sencillo de como encriptar un mensaje, sin embargo existen otros métodos para transformar cualquier carácter existente a código binario.

Tabla 2.3: Ejemplo de encriptación con suma binaria (Elaboración propia).

Palabra	**U**					**A**					**Q**				
Mensaje en binario	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
Clave aleatoria	1	1	0	0	0	1	1	0	1	0	1	0	0	1	1
Mensaje encriptado	0	1	1	0	0	1	1	0	1	0	0	0	0	1	1

El cifrado *One-Time-Pad* es un sistema totalmente seguro debido a que no existen patrones reconocibles en la clave por ser aleatoria y única para cada mensaje (Djordjevic, 2019). El principal problema de este sistema es que para que el mensaje sea indescifrable el emisor y receptor deben de utilizar una clave nueva para cada mensaje que comparten. Es por ello que se requiere de un método para poder enviar la clave de forma segura. Sin embargo, tal método no existe para la tecnología clásica y es aquí donde entra la mecánica cuántica con los *qubits* y la distribución de claves cuánticas.

2.3 *Qubit*: unidad básica de información cuántica

En computación clásica, la unidad mínima para almacenar y registrar procesos en una computadora se conoce como *bit* (abreviación de *binary digit*) y puede tomar uno de los dos valores posibles: “0” ó “1”. Estos valores pueden ser representados por dos estados opuestos o contrarios, como por ejemplo el lanzamiento de una moneda que puede tomar el valor “cara” ó “cruz”. Al lanzar la moneda y observar el resultado siempre se obtendrá sólo uno de los dos valores posibles, independientemente de cómo se realice la medición (Milburn, 2012). En computación cuántica también existe una unidad mínima de información, la cual es conocida como *bit* cuántico o *“qubit”*. El *qubit* es un sistema cuántico de 2 niveles que puede estar en una superposición simultánea de dichos estados (Nielsen y Chuang, 2010).

Los estados posibles de un *qubit* se pueden representar utilizando la notación matemática de Dirac o notación bra-ket, donde para el *bit* 0, el estado correspondiente es $|0\rangle$ mientras que para el *bit* 1, el estado es $|1\rangle$. La superposición de estos dos estados está dada por (Dür y Heusler, 2013):

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.1}$$

donde α y β son las amplitudes de probabilidad de encontrar al sistema en el estado $|0\rangle$ o en el estado $|1\rangle$, respectivamente. Las amplitudes de probabilidad son importantes en *QKD* porque permiten construir los diferentes estados a partir de la propiedad física con la que se crean los *qubits*. La ecuación 2.1 representa una superposición de estados y luego de una medición se produce un resultado específico: se obtendrá el estado $|0\rangle$ con probabilidad $|\alpha|^2$ o el estado $|1\rangle$ con probabilidad $|\beta|^2$. Además, α y β cumplen con la condición de normalización, dada por la ecuación 2.2. Esta ecuación permite visualizar geométricamente al *qubit* en la esfera de Bloch[5], como se observa en la Figura 2.2 (Dür y Heusler, 2013).

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.2}$$

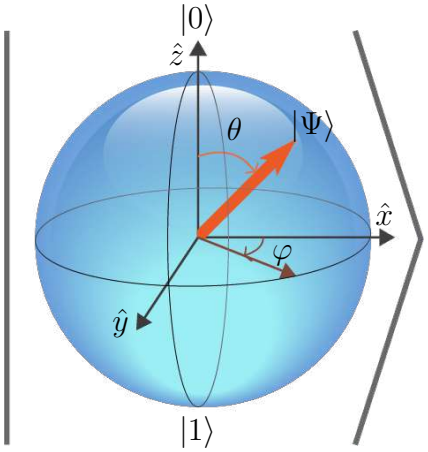


Figura 2.2: La esfera de Bloch. El vector rojo representa el estado $|\Psi\rangle$, uno de los infinitos estados que un *qubit* puede tomar. Cualquier estado del *qubit* tiene una representación en términos de dos ángulos $0 \leq \varphi \leq 2\pi$ y $0 \leq \theta \leq n$ dado por $|\psi\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)e^{i\varphi}|1\rangle$. Los ángulos φ y θ determinan un punto en la esfera (Wolf, 2021).

[5]Es una representación geométrica de la superposición de un estado cuántico de dos niveles. Cada estado cuántico $|\Psi\rangle$ está representado por un punto en la esfera unitaria (Nielsen y Chuang, 2010)

Para codificar nuestra información se necesita un sistema físico que represente los estados posibles de un *qubit*, por lo que surge la siguiente pregunta: ¿Cómo se puede codificar el valor de un *bit* en un sistema cuántico? Existen diferentes sistemas cuánticos que aprovechan las propiedades físicas para la construcción de un *qubit*, algunos ejemplos son (DiVincenzo, 2000): fotones individuales donde los distintos grados de polarización representan los dos estados posibles; también es posible utilizar los estados del espín de una partícula variando el campo magnético externo. En la Tabla 2.4 se muestran algunos ejemplos de sistemas cuánticos para representar *qubits*.

Tabla 2.4: Ejemplos de sistemas cuánticos para representar *qubits* (Fox, 2006).

Sistema cuántico	**Propiedad física**	$\|0\rangle$	$\|1\rangle$
Fotón	Polarización lineal	Horizontal	Vertical
Fotón	Polarización circular	Izquierda	Derecha
Núcleo	Espín	Arriba	Abajo
Electrón	Espín	Arriba	Abajo
Unión tipo Josephson[6]	Carga eléctrica	N pares Cooper	N+1 pares Cooper
Espira superconductora[7]	Orientación de flujo magnético	Arriba	Abajo

Cabe mencionar que el experimento desarrollado en este trabajo no utiliza fotones individuales debido a la dificultad y costo de producirlos. En vez de ello, se utilizan pulsos cortos de luz láser análogos a los fotones individuales, los cuales permiten codificar la información en la polarización de la luz y seguir correctamente todos los pasos del protocolo BB84.

[6]En un material superconductor se forman pares de electrones (llamados pares de Cooper) que actúan como partículas individuales. Cuando se colocan capas delgadas de un superconductor entre capas delgadas de un aislante, se obtiene una unión tipo Josephson. En estas uniones, los niveles de energía de los pares de Cooper en una unión de Josephson son discretos y pueden usarse para codificar *qubits*. Makhlin, Schön, y Shnirman (2001)

[7]En un circuito superconductor con uniones tipo Josephson, es posible representar el *qubit* mediante el flujo magnético externo dentro del circuito (Kockum y Nori, 2019).

2.3.1 Polarización

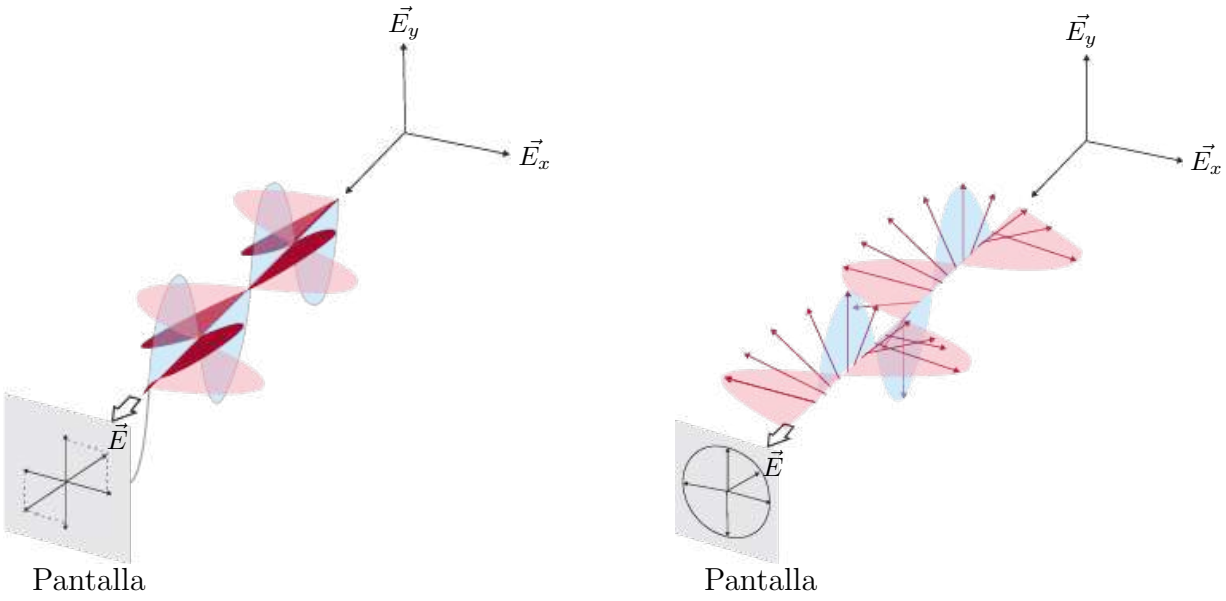
Según la teoría electromagnética clásica descrita por James Maxwell, la luz se describe como una onda transversal formada por un campo eléctrico $\vec{E}$ que oscila de forma perpendicular a un campo magnético $\vec{B}$, siendo ambos perpendiculares a la dirección en la que se propaga la luz. Para una onda electromagnética, la polarización se define como la dirección en la que oscila su vector de campo eléctrico (Pedrotti y cols., 2017). Por ejemplo, para la onda electromagnética descrita por las ecuaciones

$$\overrightarrow{\mathbf{E}} = E_0 \sin(kz - \omega t)\hat{\mathbf{x}} \tag{2.3}$$

$$\overrightarrow{\mathbf{B}} = \left(\frac{1}{c}\right) E_0 \sin(kz - \omega t)\hat{\mathbf{y}} \tag{2.4}$$

lo que significa que está polarizada en la dirección $\hat{x}$, puesto que el vector de campo eléctrico ecuación 2.3 oscila a lo largo del eje $\hat{x}$. Se pueden definir tres tipos de polarización (Hecht, 2002):

- **Polarización lineal:** El vector de campo eléctrico oscila a lo largo de un eje fijo. En la Figura 2.3a se representa la polarización lineal, donde el vector de campo eléctrico oscila en los cuadrantes I y III trazando una línea que forma un ángulo de 45° con el eje $\hat{x}$.
- **Polarización circular:** Las componentes ortogonales del vector de campo eléctrico tienen la misma amplitud, pero difieren en fase y el extremo del vector de campo eléctrico define un círculo. Se tienen dos tipos: polarización circular derecha y polarización circular izquierda. En la Figura 2.3b se muestra la polarización circular derecha donde el vector de campo eléctrico tiene amplitud constante y rota en el sentido de las manecillas del reloj (es decir, siguiendo la regla de la mano derecha) con la misma frecuencia con la que oscila.
- **Polarización elíptica:** Es el tipo de polarización más general. Las componentes ortogonales del campo eléctrico varían tanto en la amplitud como en la fase. Aquí el extremo del vector del campo eléctrico define una elipse.



(a) Polarización lineal

(b) Polarización circular

Figura 2.3: Representación gráfica de la polarización lineal y circular de un campo eléctrico oscilatorio que se propaga a lo largo del eje $\hat{z}$. En la Figura 2.3a el campo eléctrico $\vec{E}$ (onda roja obscura) es la resultante del campo eléctrico en las diferentes componentes, $\vec{E_y}$ (ondas azul) y $\vec{E_x}$ (rojo claro). El campo resultante traza una diagonal en la pantalla, la cual oscila en los cuadrantes I y III. En la polarización circular de la Figura 2.3b el campo resultante (flechas rojas) traza un círculo en la pantalla. En la polarización circular las componentes del vector de campo eléctrico conservan la misma amplitud pero difieren en fase en comparación con la polarización elíptica en la cual difieren tanto en amplitud como en la fase (Hecht, 2002).

Existen diversos dispositivos que permiten cambiar la polarización de la luz, tal como los dispositivos basados en el Efecto Pockels[8], filtros polarizadores y retardadores de onda. En ese trabajo se utilizaron los retardadores de onda $\lambda/2$ debido a que la luz que sale del láser ya está polarizada y nos interesa cambiar la polarización de acuerdo a los diferentes ángulos de las bases de Alice y Bob. Un retardador de onda es un elemento óptico que cambia la polarización de una onda incidente. Como su nombre lo indica, retrasa la fase relativa de las componentes de la onda incidente en una cantidad

[8]Estos dispositivos varían el campo eléctrico cambiando el índice de refracción del medio convirtiéndolo en un medio anisótropo. Este medio genera un desface en las componentes del campo lo que produce una polarización diferente al salir del medio.

determinada, cambiando la polarización de la onda al salir del retardador (Pedrotti y cols., 2017). Dependiendo del ángulo con el que se rote el retardador, la polarización se transforma en otro estado dado el ángulo de rotación. En la Figura 2.4 se muestra como incide luz linealmente polarizada y la polarización cambia al pasar a través del retardador de onda. Un retardador que introduce un diferencia de fase relativa de π o 180° se denomina placa de media onda ($\lambda/2$). También existen otros retardadores como el de onda completa (λ) o los de cuarto de onda ($\lambda/4$), los cuales introducen una diferencia de fase de 2π y $\pi/2$ respectivamente. Un retardador de onda λ mantiene la fase de la onda por lo que no hay ningún efecto observable en la polarización, sin embargo son muy efectivos para corregir cambios en las polarizaciones. Los retardadores de onda $\lambda/4$ permiten cambiar la polarización lineal en circular y viceversa (Hecht, 2002).

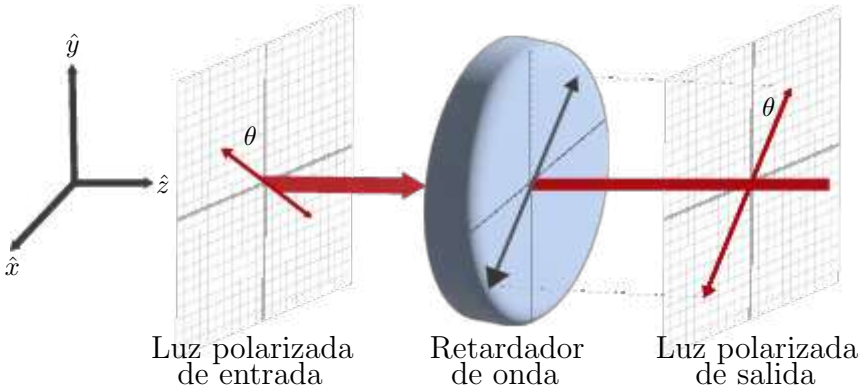


Figura 2.4: Funcionamiento de una placa retardadora $\lambda/2$. Un rayo de luz polarizado incide en la placa retardadora $\lambda/2$, la cual está rotada en un ángulo θ del eje rápido. Posteriormente la luz sale polarizada de acuerdo con el ángulo θ (Hecht, 2002).

Cuando la luz sale a través del retardador de onda se necesita poder diferenciar entre el estado $|0\rangle$ y el estado $|1\rangle$. Para esto se utiliza un cubo divisor de haz polarizado 50:50 (*polarizing beamsplitter cube*, PBS) y dos módulos *LDR*[9]. El PBS es un espejo semi-transparente que divide una onda entrante en dos haces polarizados ortogonalmente dependiendo de su orientación. Cuando la luz de entrada no está polarizada, las intensidades de los dos haces de salida están dadas por la relación 50:50, es decir 1/2 en ambos casos. De esta forma, al salir la luz del PBS se obtienen los diferentes estados

[9]Los módulos *LDR* (*light dependent resistors*) son sensores fotoresistivos que varían la resistencia dependiendo de la intensidad de luz de entrada (Boylestad, Nashelsky, Barraza, y Fernández, 2003).

$|0\rangle$ ó $|1\rangle$ (Fox, 2006). Cuando las partículas de luz, denominadas fotones, pasan a través de un retardador de onda, se puede definir su estado de acuerdo con la ecuación (Boscá Díaz-Pintado, 2017):

$$\Psi = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle . \tag{2.5}$$

lo que significa que los estados $|0\rangle$ y $|1\rangle$ tienen la misma probabilidad de ocurrir y es $\frac{1}{2}$. Entonces, cuando este estado superpuesto incide en el PBS, la mitad de las veces se obtendrá el estado $|0\rangle$ y la otra mitad el estado $|1\rangle$ (Dür y Heusler, 2013). En la Figura 2.5 se muestra el esquema de detección.

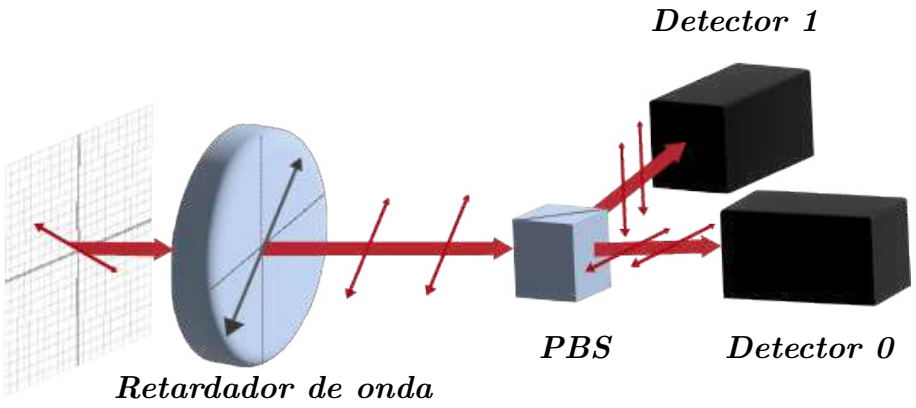


Figura 2.5: Esquema de detección para fotones polarizados. Al principio se tiene un fotón del cual no se conoce su polarización, se encuentra despolarizado. Este fotón incide en un retardador de onda el cual lo polariza y finalmente pasa a través del PBS para ser desviado a los detectores. La desviación hacia los detectores depende de si los fotones vienen con polarización horizontal (0°) o vertical (90°). En caso de que el fotón venga en un estado superpuesto se obtendrá una medición aleatoria de acuerdo a las amplitudes de probabilidad (Fox, 2006).

Codificar la información en la polarización de la luz ha sido ampliamente implementado en *QKD* puesto que presenta una gran ventaja para el envío de información a grandes distancias. Esto es debido a que los fotones no tienen masa ni carga eléctrica, se mueven a la velocidad de la luz y no interaccionan entre ellos, lo que los hace una excelente opción para codificar y transmitir información (Brukner, Żukowski, y Zeilinger, 2002). Sin embargo, una de las mayores dificultades es producir fotones individuales, que es el principal requerimiento para garantizar una *QKD* segura. Se ha logrado producir un sólo fotón polarizado con pulsos de luz atenuados garantizando que

sólo el 1 % contiene más de un sólo fotón por pulso (Boscá Díaz-Pintado, 2017). Otro reto importante es mantener la polarización de la luz durante la transmisión, por lo que se necesita de medio que mantenga la polarización, velocidad y atenuación durante la transmisión (Wiesner, 1983).

En este trabajo se utiliza la polarización lineal, ya que dependiendo de la dirección en la que oscila el campo eléctrico se pueden codificar los *bits* de información. Por ejemplo, se puede interpretar la polarización horizontal como "0" y la polarización vertical como "1". Además, debido a los obstáculos que implica crear fotones individuales, en este trabajo se emplean pulsos de luz clásicos. Cabe mencionar que a pesar de utilizar pulsos de luz clásicos, los principios son exactamente idénticos que si el sistema fuera implementado con fotones individuales. En consecuencia, la configuración experimental con pulsos de luz clásicos es adecuada para un prototipo experimental análogo de *QKD*.

2.3.2 Teorema de no clonación

La seguridad de *QKD* reside en que Eve requiere medir los valores de expectación del qubit para poder obtener la información de él. Este valor medio o valor de expectación proporciona las propiedades físicas (observables) de un sistema cuántico, de acuerdo a los postulados de la interpretación probabilística de la teoría cuántica (Griffiths, 2004). Sin embargo algunos pares de propiedades como el momento y la posición, denominados "canónicamente conjugados" (Bransden y Joachain, 2000), producen que al medir una propiedad altera el resultado de la otra. Esto se explica del principio fundamental de la mecánica cuántica conocido como principio de incertidumbre, descubierto por Werner Heisenberg. Este principio dice que para ciertos pares de propiedades físicas como la posición y el momento, a mayor precisión en la medida de la posición, menor será la precisión en el momento y viceversa. En términos matemáticos el producto de estas dos indeterminaciones no puede ser menor que la constante de Planck (Boscá Díaz-Pintado, 2017).

Lo anterior puede ser comprendido por una analogía utilizada por Chris Bernhardt

en su libro "*Quantum computing for everyone*" (Bernhardt, 2019) donde menciona el caso de medir la velocidad de un auto utilizando una pistola de radar que genera fotones. Estos fotones rebotan en el carro pero como los fotones son insignificantes en proporción con el auto, entonces no modifican el estado del auto. Sin embargo, si se quisiera medir la velocidad de un sólo fotón con el mismo método pasaría algo diferente. Si se utiliza la misma pistola radar se lanzarán millones de fotones que afectaran al único fotón que se requiere medir. Incluso si la pistola de radar arrojará un sólo fotón, provocaría que el sistema cambiase debido a que las dimensiones de energía son las mismas y esto genera un cambio no despreciable.

Las mediciones en mecánica cuántica son procesos no reversibles. Cuando se miden las propiedades de un sistema descrito por la ecuación de onda, la medición provoca que la función colapse a uno de sus estados posibles. Este proceso es irreversible ya que es imposible obtener el estado original de un sistema únicamente a partir del resultado de la medición. Esta es la idea fundamental del Teorema de No Clonación propuesto por Wootters y Zurek, en el cual, debido al principio de indeterminación, no es posible crear un dispositivo capaz de clonar un sistema cuántico. Crear una copia exacta de un sistema cuántico implicaría determinar el estado original del sistema desconocido después del colapso de la función de onda, lo cual es imposible (Wootters y Zurek, 1982).

En criptografía cuántica la información se puede codificar en la polarización de los fotones, los cuales siguen los principios de la mecánica cuántica. Por lo que si Eve interfiere en el canal de comunicación entre Alice y Bob, tendrá que medir la polarización de los fotones, realizar una copia exacta y enviárselos a Bob para no ser descubierta. Esto es una tarea imposible debido al teorema de no clonación. Sin embargo se necesita garantizar que las mediciones se realizan sobre pares de propiedades complementarias. Para garantizar un envío seguro, se pueden establecer dos bases de polarizaciones diferentes, donde cada base está definida por dos polarizaciones ortogonales entre sí, ver Tabla 2.5.

Tabla 2.5: Construcción de las bases y los ángulos de polarización para representar un *bit* (Fox, 2006).

Base	**Ángulo de polarización**	***Bit***
$\oplus$	$\theta = 90°$	1
	$\theta = 0°$	0
$\otimes$	$\theta = 45°$	1
	$\theta = -45°$	0

Siguiendo con el ejemplo anterior, si Eve realiza la medición de la polarización en la base $\otimes$ perderá toda la información que se encuentra en la base $\oplus$ y viceversa. Esto provoca que Eve introduzca errores que pueden ser detectados por Alice y Bob cuando las mediciones se sometan a un análisis de comunicación segura. Como se mencionó anteriormente, existen diversos protocolos que utilizan las propiedades de la mecánica cuántica para el envío seguro de información. Este trabajo se enfocará en el protocolo BB84, el cual es la base para nuestro experimento automatizado de *QKD*.

2.4 Protocolo BB84

El nombre del protocolo BB84 se debe a sus creadores Bennett y Brassard quienes propusieron su trabajo en 1984. En este protocolo es posible codificar los 0's y 1's en los estados de polarización de fotones individuales (Bennett y Brassard, 1984), mostrado en la Tabla 2.5. El protocolo BB84 permite crear una clave privada entre dos usuarios, la cual servirá para usarla junto con un método clásico de encriptación, tal como el *One-Time-Pad* para encriptar el mensaje secreto con la clave y después enviar el mensaje cifrado a través de un canal público de forma segura.

Antes de mencionar los pasos del Protocolo BB84 hay que recordar que existen tres personajes principales. Alice es el emisor de información (la persona que desea compartir el mensaje secreto) y Bob es el receptor (el destinatario del mensaje). Eve es la espía que intenta interceptar el canal de comunicación y robar el mensaje sin que

Alice y Bob se den cuenta. En *QKD* Alice codifica sus *bits* en los estados de polarización de fotones individuales (*qubits*). El protocolo BB84 permite que Alice y Bob verifiquen que su canal de comunicación es seguro y generen una clave privada la cual es una secuencia aleatoria *bits* para codificar y decodificar su mensaje secreto. En la Figura 2.6 se observa la representación de un sistema de comunicación cuántica siguiendo el protocolo BB84.

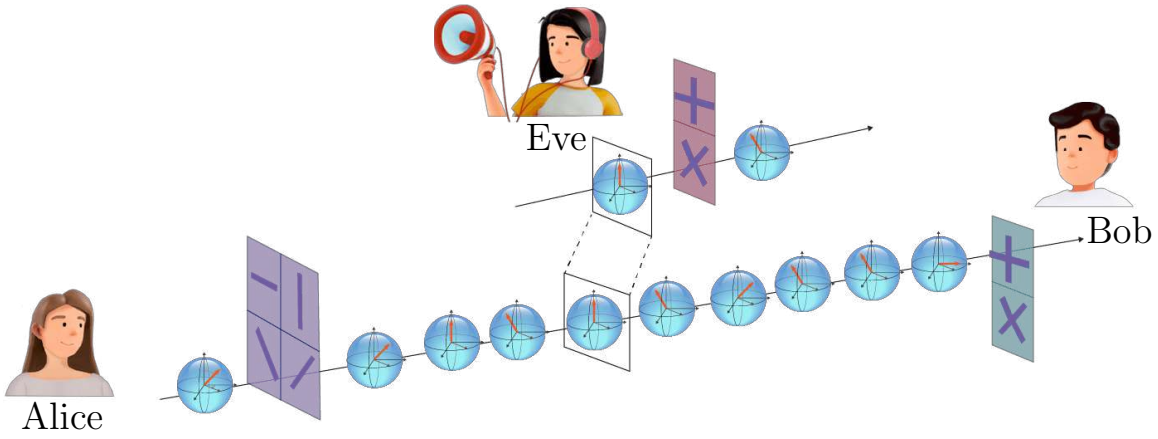


Figura 2.6: Representación gráfica de los pasos simplificados del protocolo BB84. Se encuentran los siguientes personajes: Alice (emisor), Bob (receptor), Eve (espía). Alice codifica los *bits* en los estados de polarización de fotones individuales (*qubits*) representados en las esferas de Bloch. Eve y Bob miden cada *qubit* después de pasar a través de sus retardadores de onda, produciendo que la superposición del *qubit* colapse a uno de los valores clásicos (*bit*) (Elaboración propia).

De acuerdo a Benett, Brassard (1984) y Bernhardt (2019) los pasos del Protocolo BB84 son:

1. **Creación de una secuencia aleatoria de *bits* y bases.** Alice crea una secuencia aleatoria de *bits* y para cada uno de ellos se selecciona aleatoriamente entre las bases posibles ($\bigoplus$ ó $\bigotimes$). Alice envía a Bob su secuencia aleatoria de *bits* cambiando el ángulo de polarización θ de acuerdo a la Tabla 2.5. La longitud de la secuencia de *bits* es en general $4n$, donde n es un número muy grande o al menos de la misma longitud del mensaje original (privado) que se desea transmitir. La razón de escoger $4n$ se verá más adelante.

2. **Selección de las bases para la medición de Bob.** Bob selecciona de forma aleatoria las bases con las cuales realizará su medición. Luego Bob mide los fotones que Alice le envía de acuerdo a las bases que eligió. Bob detecta cada *bit* siguiendo el esquema de detección mostrado en la Figura 2.5. Al final de la transmisión Bob tiene una secuencia de *bits* de longitud $4n$ para un sistema ideal sin perdidas[10] en el canal de comunicación. En este paso Bob habrá medido todos los fotones que envió Alice y la longitud de la secuencia es $4n$.

3. **Comparación de las bases para crear la clave tamizada.** Cuando Alice y Bob utilizan la mismas bases, Bob mide y obtiene con seguridad el *bit* que le envió Alice. Sin embargo, si se escogen bases diferentes, la medición de Bob toma un estado aleatorio. En este paso Alice y Bob se comunican por medio de un canal clásico y comparan sus bases. Luego se quedan con los *bits* donde sus bases coinciden (estos se conocen como la "clave tamizada") y descartan los *bits* donde las bases difieren. Estos últimos *bits* son eliminados ya que no se tiene la seguridad de que los *bits* estén correctos[11]. Cabe mencionar que es posible utilizar un canal clásico para el envío de las bases pues sólo se transmiten datos aleatorios (aún no se comparte la información secreta, sólo se genera la clave).

4. **Comparación del 50 % de *bits* y obtención de la clave.** Bob envía a Alice un porcentaje de los *bits* donde coinciden las bases. Bob selecciona estos *bits* aleatoriamente y le envía a Alice los *bits* y las posiciones de estos. Después de recibir los *bits* y las posiciones, Alice los compara con los suyos. En caso de que todos los *bits* coincidan significa que Eve no está presente en la comunicación y el resto de los $2n$ *bits* se usará como la clave para enviar su mensaje secreto[12]. Puesto que se comparan los *bits* donde ambos usaron las mismas bases, el error

[10] En un canal de comunicación existen pérdidas debidas al ruido o factores físicos. Estos errores se pueden corregir mediante algunos métodos, ver la sección Autenticación.

[11] La eliminación de los bits debida a la diferencia de las bases produce que la longitud de la clave se reduzca a $2n$. Como ambas bases son igual de probables, aproximadamente la mitad de las veces se obtendrá una de ellas y la otra mitad la otra base.

[12] La eliminación al comparar el 50 % de los bits de la clave tamizada produce que la clave final tenga una longitud de n. Se puede obtener una clave de menor longitud debido a los ruidos en el canal o pérdidas.

deber ser cero o menor a cierto rango debido a las posibles pérdidas o errores de medición en el canal de comunicación real.

5. **Verificación del canal.** Se calcula la tasa de error (porcentaje de *bits* que difieren donde Alice y Bob utilizaron las mismas bases) para verificar si el canal es seguro o no. Alice puede deducir que el canal de comunicación es seguro si el error es cero o menor al 25 %. Los errores existentes pueden ser atribuidos a fallas en la medición o pérdidas. Si una cuarta parte de los *bits* no coinciden, es decir el error es mayor o igual a 25 %, significa que Eve introdujo un error al interceptar los *qubits* de Alice y el canal de comunicación no es seguro. En este caso no se crea la clave y Alice y Bob tienen que volver a iniciar el protocolo hasta encontrar un canal seguro.

   El porcentaje de error del 25 % se produce debido a la presencia de Eve en el canal de comunicación. Eve medirá los *qubits* de Alice creando modificaciones cada que Eve no acierte en la medición. Eve acertará aproximadamente la mitad de las veces en la elección de las bases al medir los *qubits* de Alice y también Bob acertará la mitad de las veces. De probabilidad se tiene que para dos eventos mutuamente excluyentes la probabilidad total se encuentra dada por el producto de la probabilidad de que suceda cada evento, lo que significa que un cuarto de los *bits* que recibe Bob no coincidirán con los de Alice si Eve esta presente. Lo anterior se puede entender como:

$$\begin{aligned} P_{error} =& P_{Eve\ elige\ la\ base\ incorrecta} \times P_{Bob\ elige\ la\ base\ incorrecta} \\ =& 50\,\% \times 50\,\% = 25\,\% \end{aligned} \tag{2.6}$$

La Tabla 2.6 muestra un ejemplo básico donde el protocolo se cumple exitosamente (Eve no esta presente en el canal de comunicación). Al realizar el análisis de error el porcentaje es cero, por lo tanto se descarta la presencia del espía, el canal de comunicación es seguro y se crea la clave privada. Luego Alice y Bob utilizan la clave para transmitir su mensaje de forma segura. La tabla muestra un ejemplo básico con una pequeña parte de los *bits* analizados, sin embargo para un caso real la clave que se crea es al menos tan larga como la longitud del mensaje que se desea enviar.

Tabla 2.6: Ejemplo básico del protocolo donde Eve no está presente en el canal de comunicación y se crea la clave. Para facilitar la comprensión, sólo se muestra un pequeño conjunto de la secuencia aleatoria de los $4n$ *bits*. Al inicio del protocolo la longitud de los *bits* es 12 y al final del protocolo se genera una clave de 3 *bits*, lo que significa que la clave se reduce. (Bennett y Brassard, 1984).

Protocolo BB84 sin espía												
Bases de Alice	⊗	⊗	⊕	⊕	⊗	⊗	⊕	⊗	⊗	⊕	⊗	⊗
Bits de Alice	1	0	0	1	0	1	0	1	0	1	1	0
Bases de Bob	⊕	⊗	⊕	⊕	⊕	⊗	⊗	⊗	⊕	⊗	⊗	⊗
Bits de Bob	0	0	0	1	1	1	0	0	1	1	1	0
¿Bases iguales?	✗	✓	✓	✓	✗	✓	✗	✗	✗	✗	✓	✓
Bits de Alice		0	0	1		1					1	0
Bits de Bob		0	0	1		1					1	0
Bits a comparar		Sí	Sí	No		No					Sí	No
¿*Bits* son iguales?		✓	✓								✓	
Error	Porcentaje de error = 0 % ∴ No hay espía, se crea la clave											
Llave				1		1						0

En la Tabla 2.7 se muestra el ejemplo para el caso donde Eve esta presente en el canal de comunicación. El análisis de error muestra que el error supera el límite permitido, lo cual ocurre debido a las perturbaciones que introduce Eve al hacer las mediciones en los *qubits*. Como el porcentaje de error es mayor al 25 %, el canal es inseguro y no se genera la clave. El protocolo BB84 se debe de repite nuevamente hasta que Alice y Bob encuentran un canal seguro.

Tabla 2.7: Representación del protocolo donde Eve está presente y no se crea la clave. Al igual que la Tabla 2.6, sólo se muestra un subconjunto de la secuencia de *bits*. En este caso se analizaron 3 *bits* y el porcentaje de error es del 66 %, lo que significa que Eve está presente en el canal de comunicación (Bennett y Brassard, 1984).

Protocolo BB84 sin espía												
Bases de Alice	⊗	⊗	⊕	⊕	⊗	⊗	⊕	⊗	⊗	⊕	⊗	⊗
Bits de Alice	1	0	0	1	0	1	0	1	0	1	1	0
Bases de Bob	⊕	⊗	⊕	⊕	⊕	⊗	⊗	⊗	⊕	⊗	⊗	⊗
Bits de Bob	0	0	1	0	1	1	0	0	1	1	0	0
¿Bases iguales?	✗	✓	✓	✓	✗	✓	✗	✗	✗	✗	✓	✓
Bits de Alice		0	0	1		1					1	0
Bits de Bob		0	1	0		1					0	0
Bits a comparar		Sí	Sí	No		No					Sí	No
¿*Bits* son iguales?		✓	✗								✗	
Error	Porcentaje de error = 66 % ∴ Sí hay espía y no se crea la clave											

2.4.1 ¿Como se envía y se recibe información?

Para un experimento de *QKD* mediante fotones polarizados se necesita:

- Fuente de luz, pueden ser pulsos atenuados o fuentes de un sólo fotón. Los pulsos atenuados pueden ser generados con una fuente de luz láser y un atenuador.
- Canal para transmitir los fotones polarizados. Actualmente existen dos tipos de canales, fibra óptica y espacio libre. En general, cuando se utiliza la fibra óptica se generan pérdidas por absorción y modificaciones en la polarización, mientras que en espacio libre también se generan pérdidas de fotones y se debe tomar en cuenta la decoherencia. Sin embargo, los canales de espacio libre se vuelven favorables cuando se utilizan satélites terrestres (Cao y cols., 2013).
- Generar los grados de polarización del fotón. Esto se puede realizar con un retardador de onda, el cual orienta al fotón de acuerdo a la Tabla 2.5.

- Diferenciar entre la polarización horizontal o vertical para saber si es un *bit* 0 o 1. Para esto se utiliza un PBS que refleja la componente de la luz polarizada verticalmente (90°) y transmite la componente de la luz polarizada horizontalmente (0°) (Fox, 2006).
- Colocar dos detectores que capturan la luz que se refleja o transmite al pasar a través del divisor de haz. Si la polarización de la luz es horizontal (0°), entonces se transmite por el cubo divisor y llega al detector "0", obteniendo un *bit* 0. Si la polarización de la luz es vertical (90°), entonces se refleja por el cubo divisor y llega al detector "1", obteniendo un *bit* 1. En la Figura 2.5 se muestra el esquema de detección.

Una vez implementado el sistema experimental de *QKD* se procede a realizar los siguientes pasos:

1. Crear la clave y verificar que el canal es seguro siguiendo los pasos del protocolo BB84.
2. Cifrar el mensaje con la suma *XOR* de acuerdo a la Tabla 2.1.
3. Transmitir el mensaje. La transmisión del mensaje cifrado se realiza por medio de un canal público. A pesar de que el canal público puede ser escuchado por cualquiera, la información será totalmente segura ya que está cifrada con la clave que se genera mediante *QKD*. Esta clave es totalmente segura pues los únicos que tienen conocimiento de ella son Alice y Bob. Un espía que obtenga la información del canal obtendría datos aleatorios al no tener la clave, tal como se demostró anteriormente en la Tabla 2.3 (Fox, 2006).

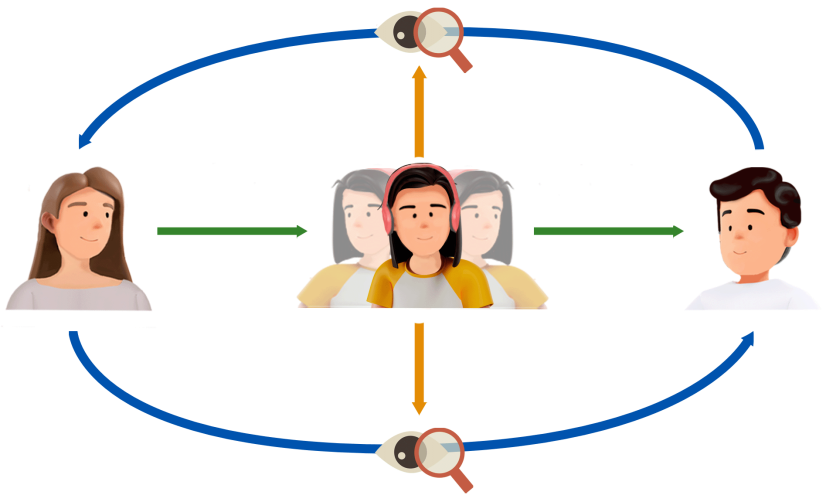
2.5 Autenticación

En principio, el término autenticación se refiere a que la información se debe de "etiquetar" de forma que los interlocutores tengan la certeza de que están hablando con la persona adecuada (Rivest y cols., 1978). Sin embargo, la autenticación va más allá de eso y debe de poder responder dos preguntas (Smart y Smart, 2016):

1. **¿Cómo saber que una entidad es quién dice ser?** En el protocolo BB84 se pueden tener dos escenarios descritos por la Figura 2.7. En un escenario (Fig. 2.7a) Eve trata de obtener el mensaje al medir los fotones polarizados de Alice y posteriormente enviarlos a Bob. Pero la medición de Eve provoca errores que pueden ser detectados por Alice y Bob al comunicarse por el canal clásico y finalmente detectar la presencia de Eve. En la segunda configuración (Fig. 2.7b) Eve suplanta completamente a Bob. Este robo de identidad implica interferir en el canal de comunicación cuántica y clásica. En el canal de comunicación cuántica Eve mide los fotones polarizados y en la comunicación clásica suplanta a Bob realizando el proceso de verificación del espía (Ma, Mink, y Tang, 2009).

2. **¿Cómo saber que la clave entre los interlocutores es idéntica?** Aún cuando Eve no este presente pueden generarse errores debido a fallas en el canal de comunicación, tal como la decoherencia o la despolarización cuando se utilizan fotones polarizados. En teoría, estos errores no excederán el 25 %, pero generan que la clave no sea idéntica entre Alice y Bob, produciendo posibles errores a la hora de encriptar y desencriptar el mensaje con la clave generada (Cao y cols., 2013).

Para resolver la primera pregunta se emplea un método de autenticación clásico en el cual se utiliza una clave precompartida con una función que correlaciona dicha clave con un grupo de la clave tamizada, la cual puede ser la suma *XOR*. Primero se divide la clave tamizada del protocolo BB84 en grupos (Grupo 1, Grupo 2, ..., Grupo N) de igual longitud que la clave precompartida. Después se aplica la suma *XOR* de la clave precompartida con el Grupo 1 de la clave tamizada produciendo la Etiqueta 1. Luego el grupo 1 pasa a ser la clave precompartida y se realiza la suma *XOR* con el grupo 2 y así sucesivamente, observe la Figura 2.8. De esta forma se crea en cada grupo una clave cifrada conocida como "Etiqueta". Al final del protocolo BB84 en la verificación de espía, los interlocutores intercambian un conjunto aleatorio de las etiquetas creadas (Ma y cols., 2009). Al final de la transmisión del mensaje se puede utilizar el último bloque de la clave tamizada como la siguiente clave precompartida para un nuevo mensaje (ID Quantique, 2020). Es importante mencionar que la distribución de la primera clave precompartida necesita ser entregada de forma física entre el emisor y receptor.



(a) ¿Eve está presente?

(b) Suplantación de Bob

Figura 2.7: **(a)** Configuración tradicional descrita por el protocolo BB84. Se observa que la comunicación cuántica (flechas verdes) suceden en un sólo sentido. Eve recibe la los fotones polarizados de Alice y los envía a Bob para no ser descubierta. En la comunicación clásica (flechas azules) interactúan los interlocutores y Eve sólo esta presente escuchando la información. **(b)** Configuración similar a **(a)**, sin embargo en este caso Eve ya no sólo se encuentra escuchando la información del canal clásico, sino que Eve interpreta a Bob para comunicarse con Alice e interpreta a Alice para comunicarse con Bob. Aquí Eve roba la identidad de Alice y Bob para obtener toda la información y enviarla sin el riesgo de ser descubierta. También existe un esquema más sencillo de **(b)** donde Eve suplanta totalmente a Bob y Eve sólo se comunica con Alice (Ma y cols., 2009).

Este procedimiento mantiene las mismas medidas de seguridad presentes en *One-Time-Pad* y en *QKD*, ya que la clave precompartida es aleatoria y del mismo tamaño que cada grupo de la clave tamizada. Además, esta clave precompartida se utiliza una sola vez en el Grupo 1 y luego se sustituye por el grupo anterior en el cual se realiza la suma *XOR*. Estos grupos de la clave tamizada se crean en el protocolo BB84 por lo

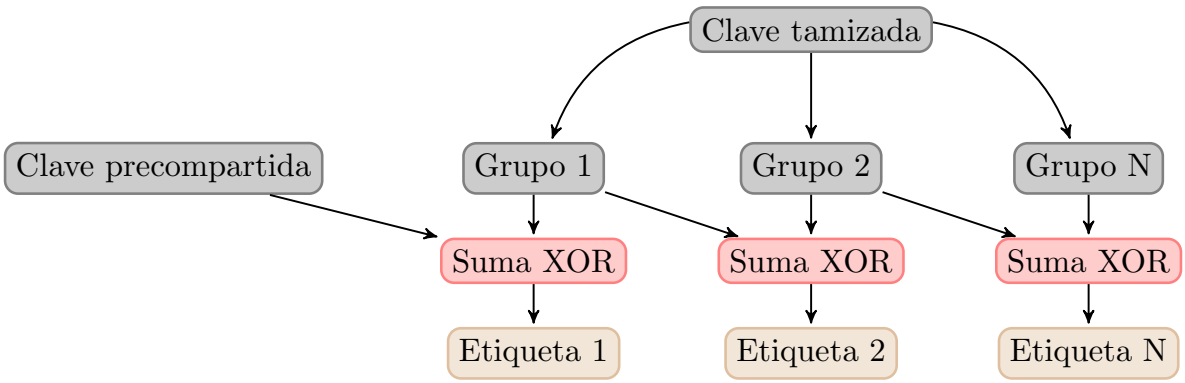


Figura 2.8: Método de autenticación clásico. En este método se crean grupos de la clave tamizada de igual longitud que la clave precompartida. Después se aplica la suma *XOR* para obtener una clave cifrada llamada "Etiqueta N". La clave precompartida sólo se aplica al primer grupo de la clave tamizada, después la clave precompartida pasa a ser el grupo anterior en el que se encuentra. Algunas partes de las etiquetas serán intercambiadas para hacer la verificación del espía en el protocolo BB84 (Ma y cols., 2009).

que son completamente aleatorios. Además, la distribución de clave precompartida se realiza en una reunión física entre los interlocutores, asegurando que se entregue a la persona adecuada y no existan escuchas.

Para resolver la pregunta dos se puede realizar un paso extra en el protocolo BB84. Este paso consiste en realizar permutaciones aleatorias de grupos de *bits*, obteniendo grupos más pequeños de permutaciones de tal forma que sea más probable encontrar un error. Los errores se encuentran cuando Alice y Bob intercambian las paridades de cada grupo que se crea. La paridad se refiere a la suma natural de los *bits*. Por ejemplo, para el grupo 010 la paridad es impar ya que la suma es 1 y el grupo 011 tiene paridad par ya que la suma es 2. Si la paridad es diferente se elimina todo el bloque, pero si son iguales se queda como la clave. Se puede seguir creando grupos más pequeños que permitan determinar si el grupo es correcto de acuerdo a su paridad. Para evitar filtrar información a Eve durante este proceso de reconciliación, Alice y Bob acuerdan descartar el último *bit* de cada bloque cuya paridad hayan revelado quedando al final las claves que utilizarán Alice y Bob (Borrayo y Arias, 2017).

Capítulo 3
Metodología

En este capítulo se describe la construcción de un experimento de criptografía cuántica basado en el Protocolo BB84. La construcción del experimento se dividió en tres partes: La primer sección da una descripción detallada de los componentes de un sistema de criptografía cuántica y los programas a implementar para el control de los dispositivos ópticos y electrónicos; la segunda sección corresponde a la calibración del experimento para evitar fallas en el protocolo o la detección incorrecta de los diferentes estados; y finalmente la tercer sección corresponde al proceso de acondicionamiento para hacer de este experimento un prototipo que cualquier persona pueda comprender y usar para aprender más sobre los fundamentos básicos de la criptografía cuántica.

3.1 Componentes para construir un experimento automatizado de criptografía cuántica

Un experimento de criptografía cuántica utiliza diversas componentes ópticas y electro-ópticas para representar a los personajes presentes en la criptografía cuántica: los interlocutores (Alice y Bob) y el espía (Eve) los cuales se muestran en la Figura 3.1. El experimento utiliza los pulsos de luz para enviar los *bits* de información, a través de la polarización, por lo que se utiliza un diodo láser y un retardador de onda $\lambda/2$, mientras que para recibirla fue necesario de un retardador de onda $\lambda/2$, un divisor de haz polarizado 50:50 y dos módulos *LDR*.

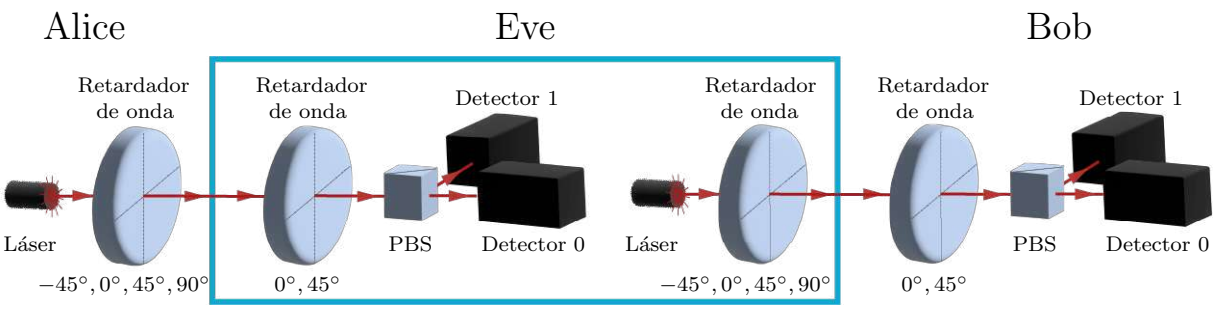


Figura 3.1: Componentes ópticos y electrónicos para un sistema de criptografía cuántica que utiliza la polarización de la luz (Elaboración propia).

El objetivo principal de este trabajo es demostrar experimentalmente cómo funciona un canal de comunicación protegido por un protocolo cuántico, mientras que se mantenga lo más accesible para su entendimiento por niños, jóvenes y adultos. Este experimento tiene la ventaja de que cualquier persona sin conocimientos previos del tema pueda comprender las bases de la criptografía cuántica. Para los fines demostrativos y didácticos del experimento se descarta a Eve, ya que la implementación de Alice y Bob es suficiente para explicar y demostrar todos los conceptos implicados en criptografía cuántica y seguir los pasos del protocolo BB84.

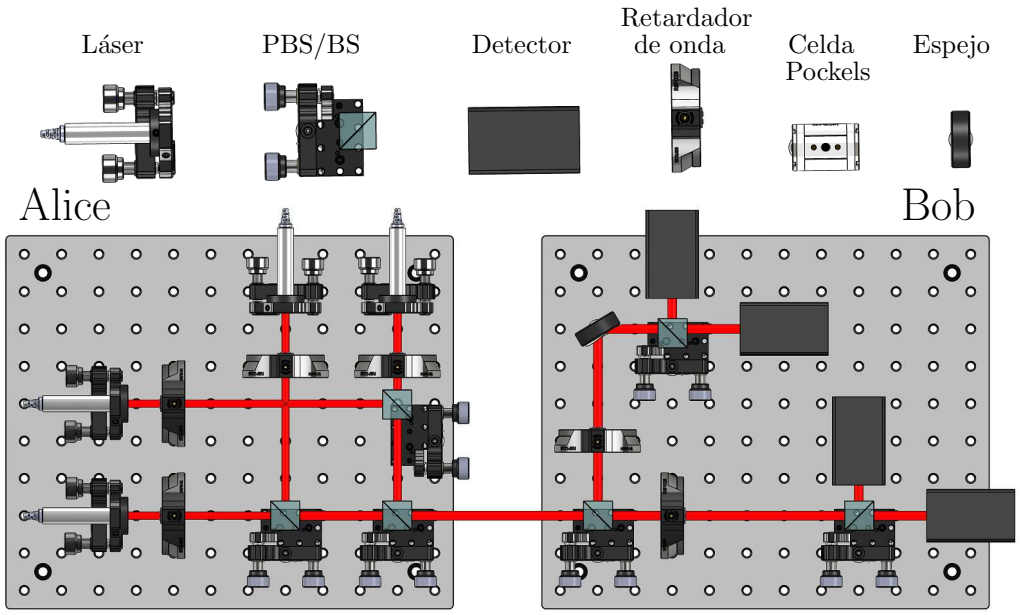
Para la construcción del experimento automatizado de criptografía cuántica se consideraron distintas configuraciones para elegir la opción que permitiera un menor tiempo y un costo asequible. En la Figura 3.2 se aprecian 3 diferentes posibles configuraciones para realizar un experimento automatizado de criptografía cuántica. Cualquiera de estas configuraciones puede implementarse para construir un experimento automatizado, sin embargo algunas requieren de componentes especializadas para implementarlos de forma eficientemente. Por ejemplo, Alice en la Figura 3.2a requiere utilizar 4 láseres y 4 retardadores de onda fijos y orientados a una cierta polarización, lo cual permite seleccionar rápidamente un estado con el encendido aleatorio del láser, mientras que Bob requiere de 2 retardadores de onda y 4 detectores para realizar la medición. Por lo que, comparado con la Figura 3.2c se puede decir que esta configuración sería más rápida y durable porque no requiere de ninguna pieza mecánica. La configuración de la Figura 3.2b es similar a la Figura 3.2c, pero requiere de dos Celdas Pockels en lugar de los retardadores de onda, sin embargo el costo de las celdas es

bastante elevado. A pesar de que la configuración de la Figura 3.2c corresponde a un experimento manual de criptografía cuántica se decidió basarnos en ella ya que es la configuración con menos componentes especializadas para su control. Las componentes extra para automatizar el sistema, tal como los servomotores o los módulos *LDR* son más sencillas de adquirir y tienen un costo más accesible.

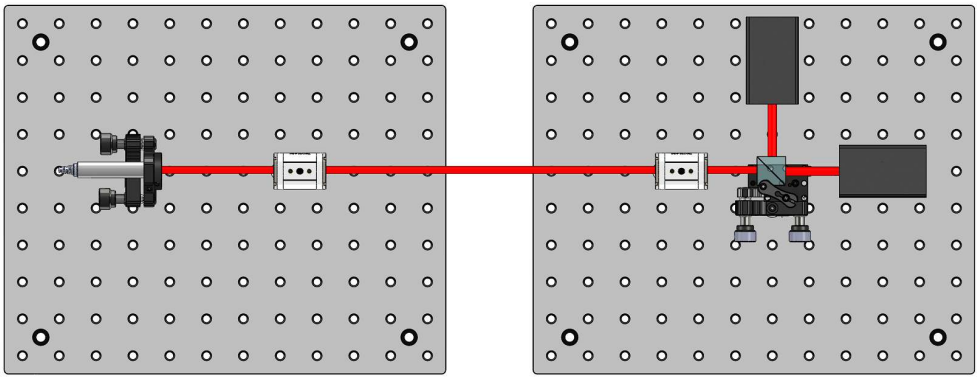
Para automatizar la configuración de la Figura 3.2c, se requiere realizar ciertas modificaciones. Estas modificaciones consisten en rotar el retardador de onda, realizar los pulsos cortos del láser, efectuar las mediciones y ejecutar los pasos del protocolo BB84 para crear la llave y enviar el mensaje. Existen diversas componentes electrónicas que permiten realizar las modificaciones para automatizar el experimento, por lo que en la Figura 3.3 se muestran los cambios necesarios a realizarse.

Lo siguiente fue diseñar y obtener las componentes mecánicas y electrónicas para automatizar el experimento siguiendo el esquema de la Figura 3.3. Se utilizó una tarjeta electrónica que nos permitiera implementar los circuitos y los códigos para el sistema de control. Para el sistema experimental se utilizó la tarjeta de control Raspberry Pi 4B[1] que es una computadora de bajo costo utilizada para realizar diversos proyectos de educación y automatización. Una de las principales ventajas de utilizar dicha tarjeta es que es posible utilizar el lenguaje de programación Python[2]. Este lenguaje de programación es cada vez más popular debido a su simplicidad y flexibilidad lo que permite realizar diversos tipos de proyectos como robótica, aplicaciones, algoritmos, IA, entre otros. Las componentes ópticas, electrónicas y las impresiones 3D utilizadas para automatizar el experimento serán detalladas a continuación.

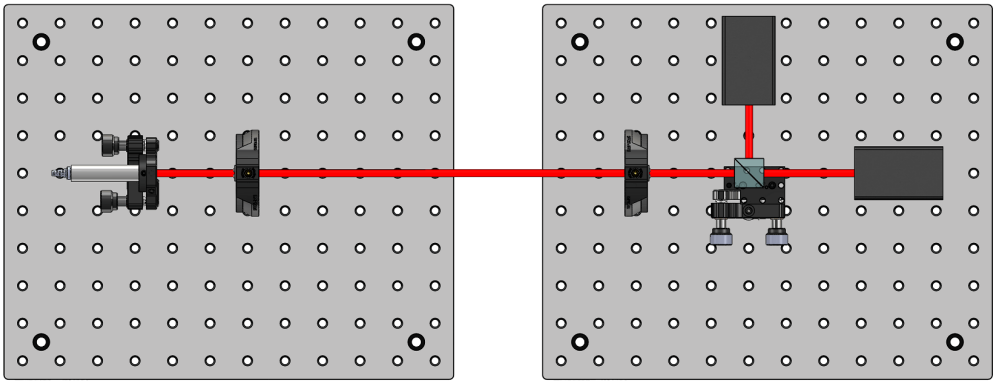
[1]Raspberry Pi 4, modelo B, url: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/
[2]Python, url: python.org



(a) Configuración 1 (Ma y cols., 2009).



(b) Configuración 2 (Bennett y cols., 1991).



(c) Configuración 3 (Thorlabs, 2017).

Figura 3.2: Configuraciones para la construcción de un experimento automatizado de criptografía cuántica. Izquierda componentes Alice, derecha componentes Bob.

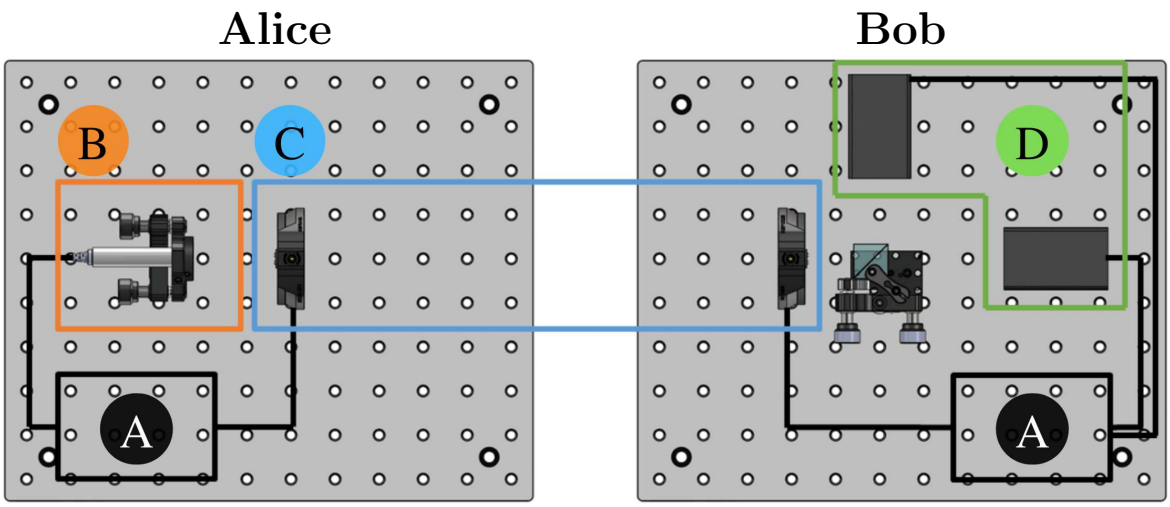


Figura 3.3: Componentes principales de la demostración experimental de criptografía cuántica. **(A)** Sistema para el control de componentes electrónicas y para ejecutar los pasos del protocolo BB84. **(B)** Circuito de control para realizar los pulsos láser. **(C)** Sistema de control para rotar automáticamente los retardadores de onda y cambiar la polarización. **(D)** Sistema de control para realizar las mediciones de los pulsos láser (Elaboración propia).

3.1.1 Alice

De acuerdo a la Figura 3.2c se observa que se necesita automatizar el láser y el retardador de onda para la configuración de Alice. Las componentes para el láser son las siguientes y se observan en la Figura 3.4:

1. Diodo Láser clase 2 de 635 nm, 1.2 mW, voltaje de operación: 4.9 a 5.2 V. (Thorlabs CPS635R-C2).
2. Fuente de alimentación del diodo láser.
3. Montura para el diodo láser (Thorlabs KM100).
4. Adaptador para colocar el láser a la montura (Thorlabs AD11NT).
5. Base y poste para fijar el diodo láser (Thorlabs UPH2 y TR2).
6. Módulo relevador de 5 V a 10 A.

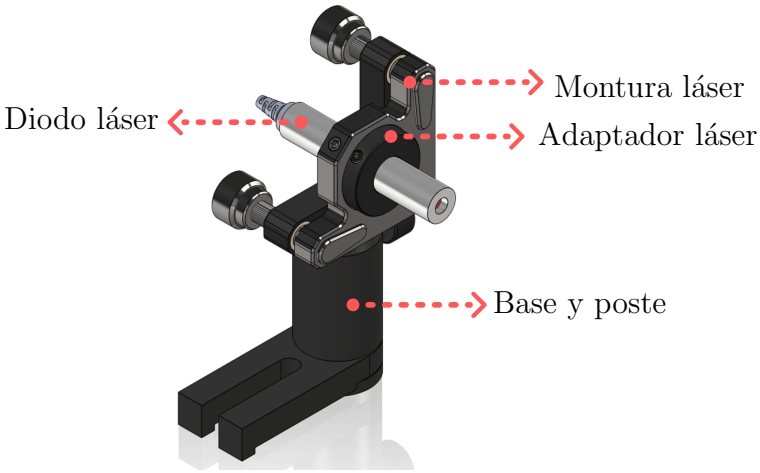


Figura 3.4: Diseño 3D de las componentes para el diodo láser (Elaboración propia).

Además se realizó un circuito para controlar los pulsos de luz láser que consistió en conectar un módulo relevador de 5 V a 10 A entre el láser, su fuente de poder y la Raspberry Pi 4. El relevador es un elemento electromecánico que funciona a partir de un electroimán mediante el accionamiento de sus contactos que pueden ser normalmente abierto o normalmente cerrado. De esta forma es posible controlar la corriente alterna con una señal pequeña de corriente directa (Olvera, 2016). Esto permite que mediante una señal de baja potencia como por ejemplo una de las salidas de la Raspberry Pi 4, accione el relevador permitiendo el control del paso de corriente del láser produciendo los pulsos de luz. Puede visualizarse el diagrama del circuito para el láser en la Figura B.1 del apéndice A.

Para automatizar el retardador de onda es necesario un sistema mecánico que controle la rotación para obtener las distintas polarizaciones, por lo que se utilizó un servomotor. Un servomotor es un motor eléctrico que permite controlar la posición angular y velocidad de su eje con buena precisión en un rango determinado. El servomotor utilizado tiene un rango de 180° el cual es suficiente para cambiar entre las diferentes bases de Alice $(-45°, 0°, 45°, 90°)$. El servomotor es alimentado por una fuente externa, por lo que fue necesario utilizar un módulo relevador para su control con la Raspberry Pi. Con ayuda del software Solidworks se diseñaron las piezas 3D necesarias para acoplar el servomotor al poste y a la base. Uno de estos diseños consiste en una rueda con cinta dentada en su circunferencia la cual se acopla al eje del servomotor. Esta rueda

funciona como engranaje y permite la rotación de la placa retardadora $\lambda/2$ a cualquier ángulo θ.

Las componentes para el servomotor son las siguientes y se observan en la Figura 3.5:

1. Servomotor 3001HB (Power HD).
2. Fuente de alimentación para servomotor 5 V.
3. Montura para sostener el servomotor, impresa en 3D.
4. Rueda con cinta dentada para acoplar el servomotor a la montura del retardador de onda.
5. Base y poste para fijar el servomotor (Thorlabs UPH2 y TR2).
6. Módulo relevador de 5 V a 10 A.
7. Transistor 2N2222.
8. Resistencia de 4.7 KΩ.

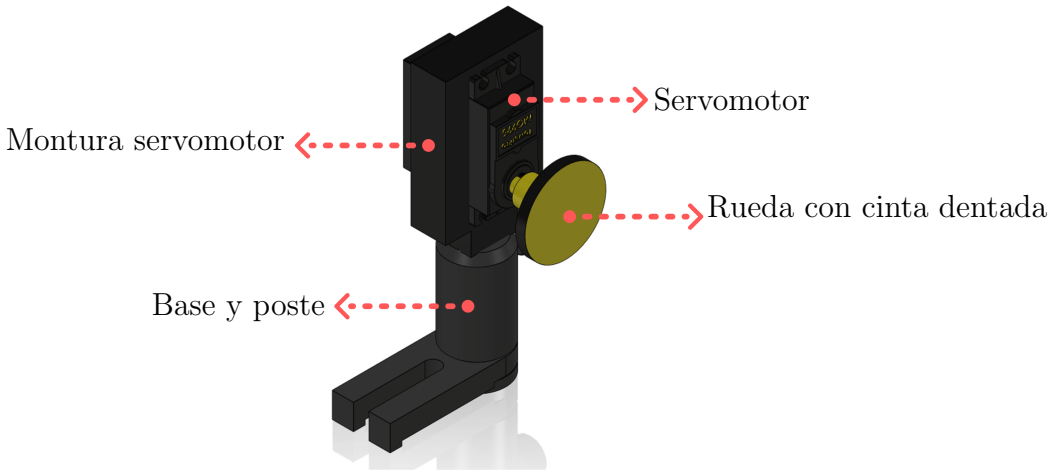


Figura 3.5: Diseño 3D de las componentes para el servomotor (Elaboración propia).

También se realizó un circuito para controlar la rotación del motor que hace mover la placa retardadora $\lambda/2$ a las diferentes polarizaciones, ver Apéndice A. Este circuito consiste en un transistor 2N2222a unido a una resistencia de 4.7 KΩ el cual funciona como un interruptor. Cuando pasa la corriente suficiente, el transistor se satura

provocando el encendido del relevador y moviendo el servomotor. Cuando la resistencia no satura el transistor todo el voltaje se manda a tierra por el mismo transistor (Horowitz, Hill, y Robinson, 1989). Este circuito sirve para activar y desactivar correctamente los pines GPIO[3] de la Raspberry Pi 4. Puede visualizarse el diagrama del circuito para el láser en la Figura B.1 del apéndice A.

Se diseñó una configuración mecánica que permite rotar el retardador de onda de acuerdo a los diferentes ángulos de polarización de las bases de Alice. Esta configuración utiliza un balero para rotar el retardador de onda. El balero se encuentra sujeto a una montura en la parte exterior y en la parte interior se encuentra un adaptador que sostiene la placa retardadora $\lambda/2$ a presión. El adaptador consta de dos partes, la adaptación interior y la adaptación exterior, entre las cuales se coloca la placa retardadora $\lambda/2$. Al igual que en el servomotor se colocó una cinta dentada en el adaptador exterior la cual funciona como un engranaje que gira dependiendo del movimiento del servomotor. La rueda del servomotor es del mismo tamaño que el adaptador exterior para crear la relación 1:1. Las componentes para la montura son las siguientes y la configuración puede visualizarse en la Figura 3.6:

1. Retardador de onda ($\lambda/2$) de 25 mm de diámetro y longitud de onda de 633 nm (Thorlabs WPH10E-633).

2. Adaptador para fijar la placa retardadora $\lambda/2$ a un sistema mecánico movible de 25.5 mm de diámetro interior y 35 mm radio exterior, impresión 3D.

3. Balero de diámetro 35 mm interior y 62 mm para permitir la rotación del retardador de onda.

4. Montura para sostener el balero de diámetro interior de 62 mm y de diámetro exterior de 82 mm, impresión 3D.

5. Base y poste para fijar la montura (Thorlabs UPH2 y TR2).

[3]Los pines GPIO (General Purpose Input/Output), son los pines de entrada y salida de uso general de la Raspberry Pi. Estos puertos pueden configurarse como entradas o salidas digitales según se necesite (Raspberry Pi, 2021).

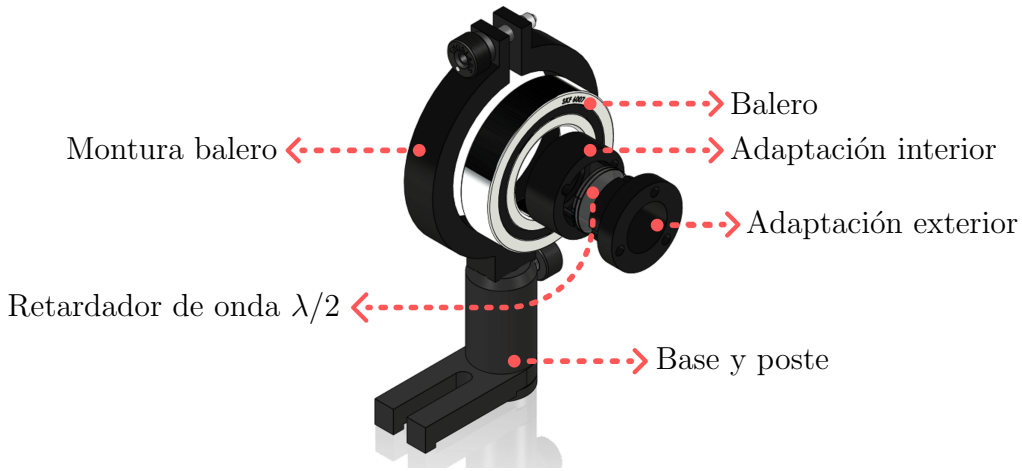


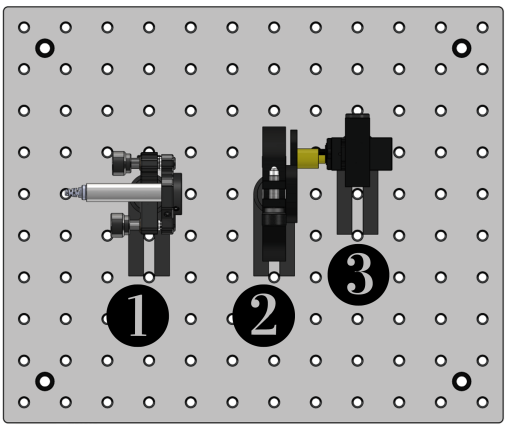
Figura 3.6: Diseño 3D de las componentes para el retardador de onda (Elaboración propia).

Para automatizar la configuración de Alice se necesitaron las siguientes componentes: láser, retardador de onda y servomotor. La Figura 3.7 muestra el diseño final de las componentes de Alice montadas sobre una base óptica (*breadboard*). La *breadboard* nos permite fijar las componentes ópticas y mecánicas para reducir las vibraciones por movimientos externos y que el sistema experimental sea estable y funcione correctamente.

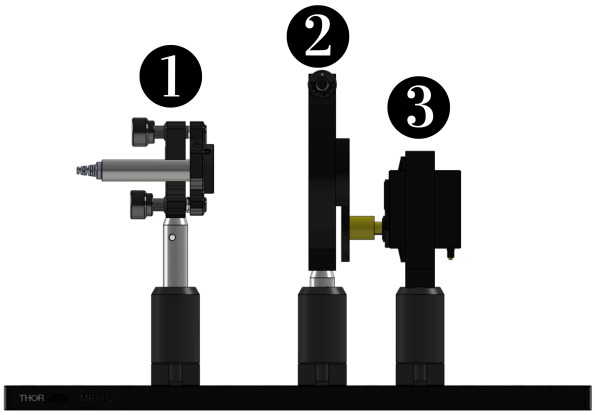
Finalmente se implementó un código en Python para Alice que permite realizar los pasos del protocolo BB84 de criptografía cuántica junto con los diferentes subcódigos de las componentes electrónicas a automatizar. Estos códigos consisten en:

1. **Código para realizar los pulsos del diodo láser mediante el relevador.** Este código permite indicar al relevador en que momento abrir o cerrar el circuito para enviar los pulsos de luz láser cada cierto intervalo de tiempo. El relevador funciona como un interruptor que cuando está activado permite el encendido del láser y cuando esta desactivado lo mantiene apagado. En el apéndice C se describe el código para realizar los pulsos láser.

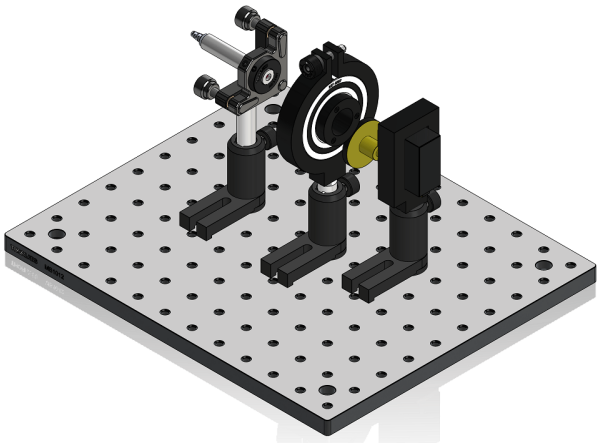
2. **Código para rotar el retardador de onda $\lambda/2$ mediante el servomotor.** En la configuración de Alice, el retardador de onda alterna entre sus 4 bases o polarizaciones $(-45°, 0°, 45°, 90°)$. Este código implementa el control del relevador y la rotación del servomotor. Antes de entrar en el bucle para alternar entre las



(a) Vista superior.



(b) Vista lateral.



(c) Vista isométrica.

Figura 3.7: Vista superior, lateral e isométrica de las componentes de Alice. (1)Láser, (2)Retardador de onda, (3)Servomotor (Elaboración propia).

4 polarizaciones, se necesita enviar una señal para activar el relevador y permitir el funcionamiento del servomotor. Después el servomotor recibe la señal de la Raspberry Pi para moverse de acuerdo a los grados dados por las polarizaciones. Después de completar la secuencia de los grados para las polarizaciones, se desactiva el pin GPIO del relevador para apagar el servomotor. En el apéndice D se describe el código para realizar la rotación del servomotor.

Estos dos códigos se utilizan cuando Alice le envía la información codificada en los pulsos de luz polarizados a Bob, en la comunicación cuántica del Protocolo BB84. En el experimento debe de moverse primero el servomotor al ángulo de polarización determinado. Después se enciende y apaga el láser rápidamente generando el pulso de luz. Finalmente el pulso de luz sale del retardador de onda $\lambda/2$ polarizado.

3.1.2 Bob

De acuerdo a la Figura 3.2c es necesario automatizar la rotación del retardador de onda $\lambda/2$ y la detección de los pulsos mediante módulos *LDR* para la configuración de Bob. Al igual que Alice, se utilizó un sistema mecánico que orienta el retardador de onda a los diferentes ángulos de polarización de las bases mediante un servomotor. Las monturas para el retardador de onda y el servomotor utilizan las mismas componentes que Alice, por lo que la Figura 3.5 y 3.6 muestran el diseño del retardador de onda y servomotor para Bob. En esta sección se describirá la automatización de los detectores y las componentes del PBS, el cual permite desviar los pulsos de luz dependiendo de la polarización con la que llegan. Las componentes para el PBS son las siguientes y se observan en la Figura 3.8:

1. Divisor de haz polarizado de 20 mm con rango de longitud de onda de 420 nm a 680 nm (Thorlabs PBS201).
2. Montura para el PBS (Thorlabs KM100PM).
3. Base y poste para fijar el PBS (Thorlabs UPH2 y TR2).

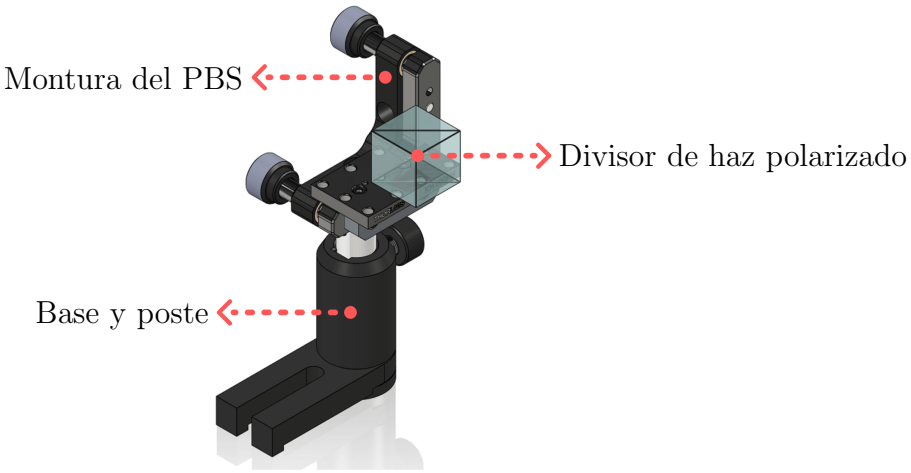


Figura 3.8: Componentes para el cubo divisor de haz (PBS) (Elaboración propia).

Posteriormente se diseñó la parte de los detectores para los cuales se utilizó un módulo sensor *LDR*. Una fotoresistencia o *LDR* (*light dependent resistors*) es dispositivo cuya resistencia varia en función de la luz recibida (Boylestad y cols., 2003). Las *LDR* están diseñadas para detectar cualquier longitud de onda visible, por lo que la luz de alguna fuente externa puede afectar las mediciones. Debido a esto se diseñó una caja 3D oscura con la finalidad de que la luz que incide sobre los detectores no sea vea afectada por la luz natural y la detección sólo se deba a los pulsos del láser. Es importante recordar que este es un prototipo experimental análogo ya que no utilizamos una fuente de un sólo fotón, sino que se generaron pulsos cortos de luz láser. Por lo tanto no se requiere de detectores especializados y se pueden utilizar *LDR*s. Para realizar la detección es necesario comparar la intensidad de luz incidente en cada *LDR*. El valor que se obtiene al medir las *LDR* es una señal analógica. Sin embargo la Raspberry Pi 4 no cuenta con pines para realizar lecturas analógicas (McManus y Cook, 2021), por lo que se utilizó un convertidor analógico a digital (ADC). Las componentes para los detectores se muestran a continuación, observe la Figura 3.9:

1. Módulo Sensor *LDR*, voltaje de operación: 3.3 V a 5 V DC.
2. MCP3008 convertidor analógico a digital.
3. Caja obscura para el módulo sensor *LDR*. Consta de dos piezas: La caja y una cubierta, impresas en 3D.

4. Base y poste para fijar la caja obscura. (Thorlabs UPH2 y TR2).

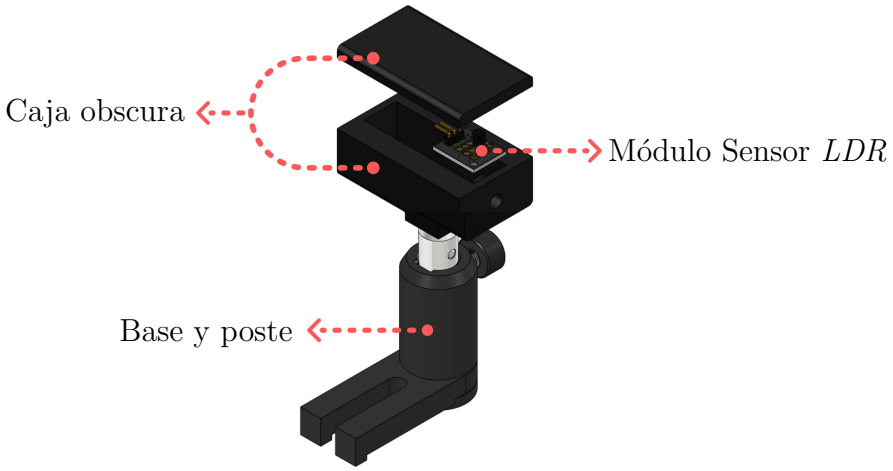
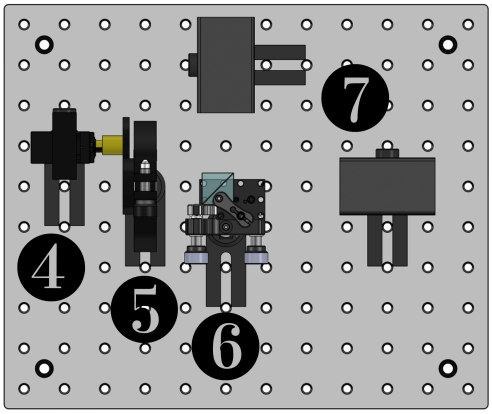


Figura 3.9: Diseño y montura para detectores.

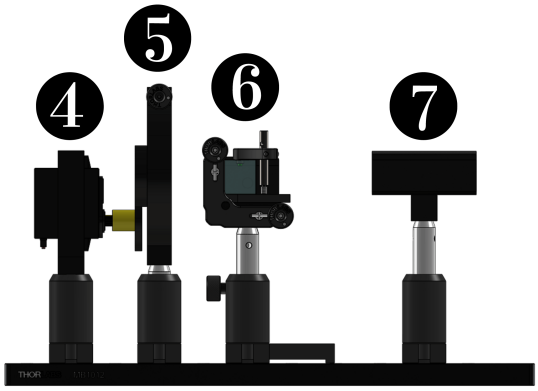
También se realizó un circuito con las conexiones que se realizaron entre el MCP3008 y la Raspberry Pi para la lectura de entradas analógicas. En este circuito también se incluyen las conexiones para el servomotor utilizado por Bob. El diagrama del circuito para los detectores se muestra en la Figura B.1 del apéndice B. Para automatizar la parte de Bob se necesitan las siguientes componentes: servomotor, retardador de onda $\lambda/2$, PBS y los dos detectores. La Figura 3.10 muestra el diseño final de las componentes de Bob montadas sobre la mesa óptica.

De la misma forma que Alice, para Bob se implementó un código en Python que permite realizar los pasos del protocolo BB84 de criptografía cuántica junto con los diferentes subcódigos de las componentes electrónicas a automatizar. Estos códigos consisten en:

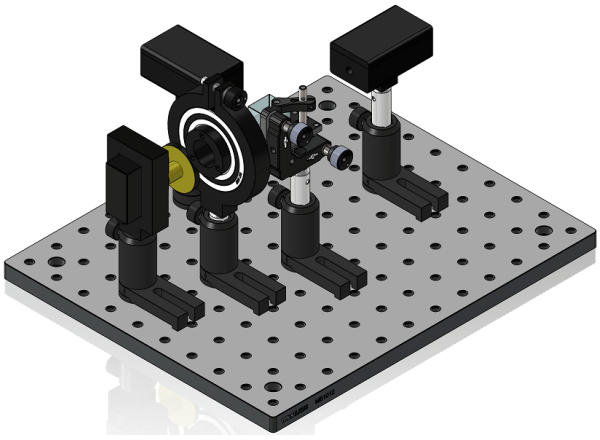
1. **Código para rotar el retardador de onda $\lambda/2$ mediante el servomotor.** En la configuración de Bob el servomotor tiene que moverse únicamente entre 2 polarizaciones (0° y 45°). En el apéndice E se describe la modificación que debe realizarse en el código del apéndice D para la rotación del servomotor de Bob.



(a) Vista superior.



(b) Vista lateral.



(c) Vista isométrica.

Figura 3.10: Vista superior, lateral e isómetrica de las componentes de Bob. (4)Servomotor, (5)Retardador de onda, (6)Cubo divisor de Haz, (7)Detectores (Elaboración propia).

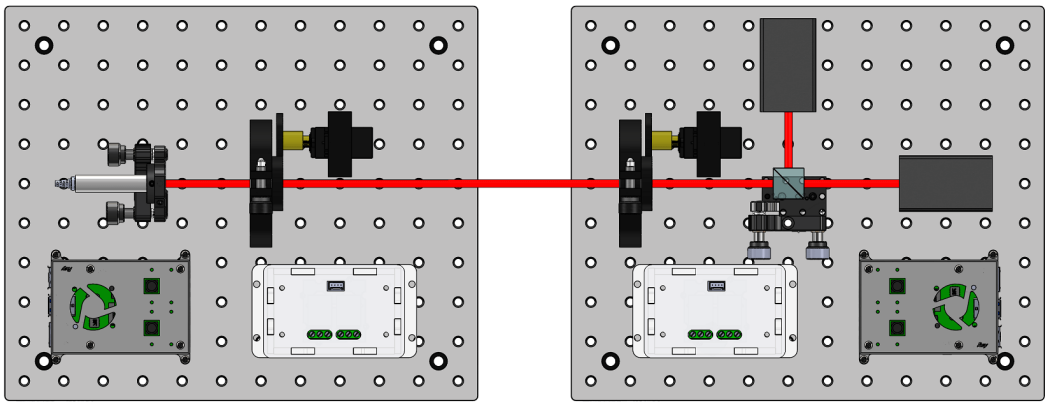
2. **Código para realizar las mediciones de los detectores utilizando el MCP3008.** Este código permite leer, comparar y registrar las intensidades de los módulos *LDR*[4] para obtener los *bit* de información. En el caso que ambos sensores registren el mismo valor, se elije aleatoriamente el valor del *bit*. En el apéndice F se describe el código para realizar las mediciones de los detectores.

Estos dos códigos se agregan en la comunicación cuántica del protocolo BB84, cuando Bob recibe la información codificada en los pulsos de luz polarizados de Alice. El código debe de mover primeramente el servomotor de Bob al ángulo de polarización determinado, después entran los pulsos de luz láser en el retardador de onda $\lambda/2$ y luego los pulsos láser polarizados inciden en el PBS a través del cual son desviados hacia los detectores dependiendo de la polarización del pulso de luz láser.

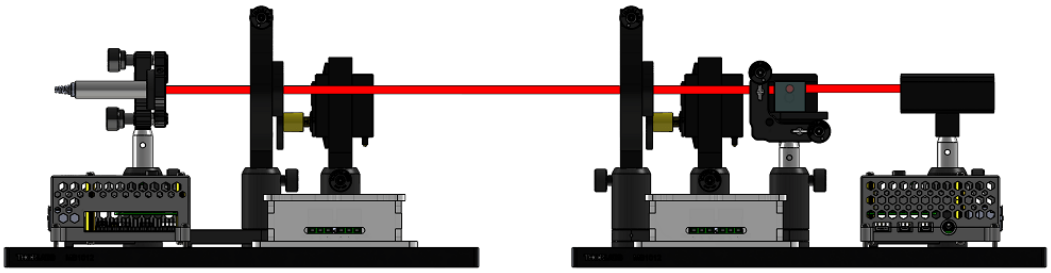
Nota: *El prototipo experimental automatizado no requiere de componentes especiales para las monturas, postes o mesa óptica. Inicialmente contamos con el kit comercial por lo que se utilizaron las monturas, postes y mesa óptica del kit. Es posible remplazar estas componentes por impresiones 3D, lo cual puede reducir significativamente el costo. Sin embargo no es recomendable sustituir las componentes ópticas y el láser debido a que la óptica utilizada en este trabajo funciona para esa longitud de onda en particular. El cambio de estas componentes puede implicar un mal funcionamiento del experimento.*

Finalmente al ensamblar todas las componentes de Alice y Bob se obtuvo la configuración final del experimento, tal como se muestra en la Figura 3.11.

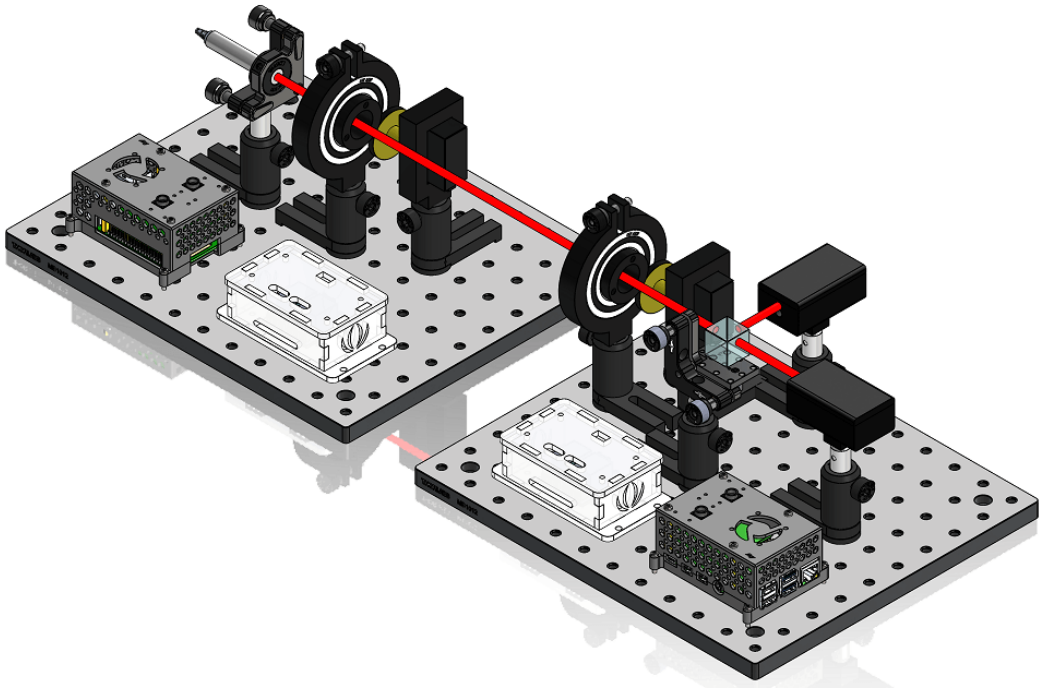
[4]Los módulos *LDR* pueden detectar una variación de resistencia mínima en el rango de 20 a 100 ms dependiendo del modelo (Llamas, 2018), por lo que para velocidades menores a 20 ms es necesario considerar algún otro sensor.



(a) Vista superior.



(b) Vista lateral.



(c) Vista isométrica.

Figura 3.11: Vista superior, lateral e isómetrica de las todas las componentes de Alice y Bob (Elaboración propia).

3.2 Programas para el control del experimento

Además de los códigos para la automatización de las componentes de Alice y Bob se incluyeron otros programas que ejecutan procesos específicos de acuerdo a los pasos del protocolo BB84. Estos programas consisten en:

1. **Comunicación por *Bluetooth***. Existen diversos tipos de comunicación en la Raspberry Pi por los cuales se puede hacer el envío de información, por ejemplo: comunicación *SPI*, comunicación *UART*, comunicación en red y comunicación *Bluetooth* (Moya Fernández, 2017). Inicialmente se consideró la comunicación a través de una conexión en red, dado que es posible enviar datos desde cualquier parte de forma inalámbrica entre las Raspberrys Pi. Sin embargo, el lugar donde se instalaría el experimento no contaba con conexión a la red. La Raspberry Pi 4 cuenta con Bluetooth 5.0 ("Datasheet Raspberry Pi 4 Model B", 2019), lo que permite establecer una conexión a una distancia de separación de hasta 100 metros (Vainio y cols., 2000). Esto es suficiente para el prototipo experimental por lo que se optó por dicha comunicación. La conexión *Bluetooth* funciona como un canal de comunicación clásica el cual es inseguro, sin embargo hay que recordar que toda la información transmitida por este canal es información aleatoria que no tiene significado sin la llave. Además, como medida de seguridad la conexión *Bluetooth* se encuentra apagada cuando Alice y Bob no comparten información y se exige la comunicación por medio de la dirección *MAC*[5] (Vainio y cols., 2000). Así mismo, la dirección *MAC* permite verificar la legitimidad de los interlocutores (Ali y cols., 2018). En el apéndice G se describe el código para realizar la comunicación *Bluetooth*.

2. **Sincronización de los pulsos láser de Alice con las mediciones de los detectores de Bob.** Este código se utiliza para indicar al servomotor de Alice y de Bob los ángulos para orientar el retardador de onda $\lambda/2$. Después se activa el láser generando un pulso de luz e inmediatamente se realiza la medición. Final-

[5] *Media Access Control* (Control de acceso a medios). Este es un identificador único para las interfaces de redes inalámbricas *Ethernet* y 802.11 (Ali, Kim, Kim, y Park, 2018).

mente, se apagan los detectores y el láser. Este proceso se hace consecutivamente hasta haber transmitido todos los *bits*. En el apéndice H se describe el código para realizar la sincronización de los interlocutores.

3. **Conversión de caracteres a binario y la suma *XOR* para el cifrado y descifrado del mensaje.** Tal y como se mencionó anteriormente, existen diversos códigos que permiten realizar esta conversión. Uno de los más conocidos se llama código *Unicode*, el cual es utilizado para representar caracteres, símbolos, signos y textos (Korpela, 2006). Una vez que Alice transforma los caracteres a binario, se realiza la suma *XOR* con la clave para encriptar el mensaje y posteriormente ser descifrado por Bob. En el apéndice I se describe el código para realizar tanto la conversión de caracteres a binario como la suma *XOR*.

4. **Importación y guardado de la llave.** El programa general consta de tres opciones: Creación de la llave, envío del mensaje y las dos anteriores en una sola opción. En el caso de elegir directamente la opción de enviar el mensaje, es necesario tener una llave, la cual se guarda previamente. Si el canal de comunicación es seguro, se crea la llave y se guardan tanto las bases como los *bits* de la llave. Una vez que el mensaje se envía, se importa la llave para cifrar y descifrar el mensaje. En el apéndice J se muestra el código para guardar e importar los datos.

5. **Calibración.** Este programa permite realizar las mediciones de los detectores por cada grado que avanzan los retardadores de onda. Los datos obtenidos de cada detector son guardados para después realizar un ajuste. A partir del ajuste se obtienen los ángulos a los cuales hay que orientar el servomotor para calibrar el sistema. En el apéndice K se muestra el código para realizar y guardar las mediciones de los detectores.

Los pasos del protocolo BB84 para cada interlocutor se muestran en el diagrama de flujo de la Figura 3.12, mientras que los pasos para el envío del mensaje se muestran en el diagrama de flujo de la Figura 3.13.

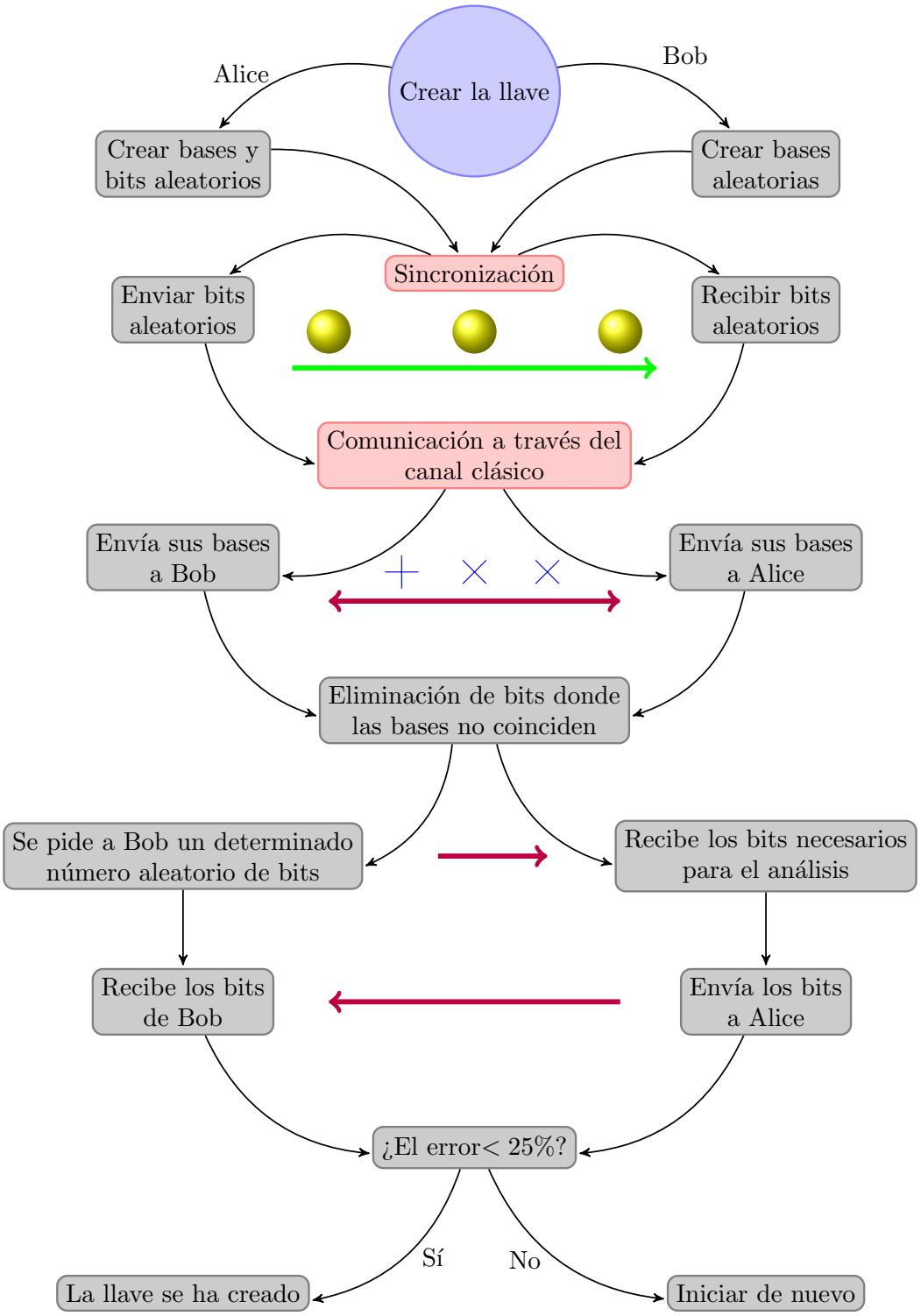


Figura 3.12: Diagrama de flujo de los pasos básicos a implementar en el protocolo BB84 para crear la clave en Python. Los recuadros grises indican los pasos del protocolo BB84; los recuadros rojos indican la comunicación que se debe realizar para el control automatizado; la flecha verde indica la comunicación en el canal cuántico; y las flechas purpura indican la dirección de la comunicación clásica (Elaboración propia).

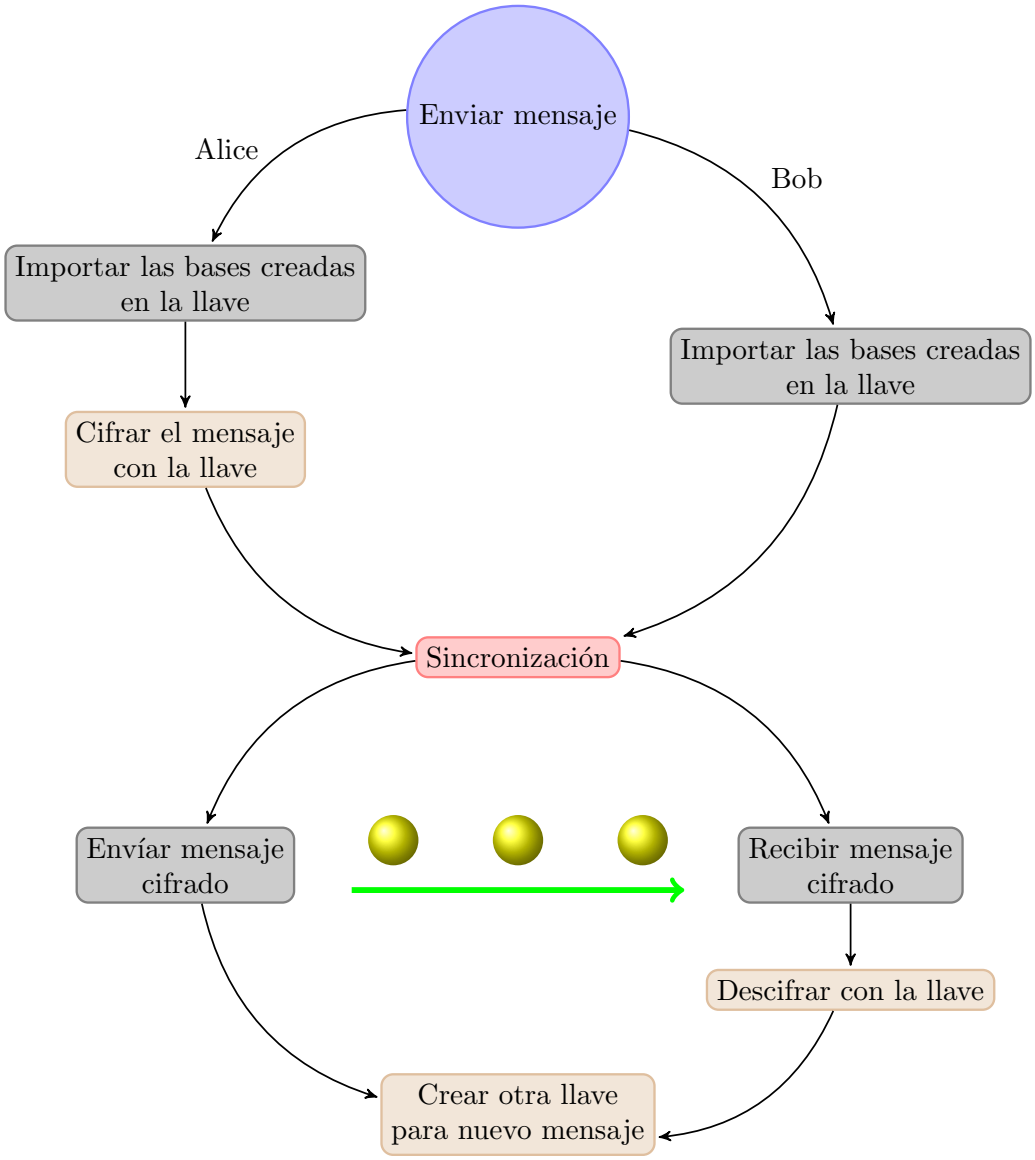


Figura 3.13: Diagrama de flujo de los pasos a implementar en programación para el envío de un mensaje secreto en Python. Los recuadros grises indican los pasos pertenecientes al protocolo BB84; el recuadro rojo indica la comunicación que se debe de realizar para el control automatizado; la flecha verde indica la comunicación en el canal cuántico; y los recuadros café indican los pasos pertenecientes al cifrado *One Time Pad* (Elaboración propia).

3.3 Calibración del experimento

Para que el experimento funcione correctamente es necesario realizar la calibración y alinear las componentes ópticas. En la calibración se pueden utilizar dos instrumentos: los detectores *LDR* del experimento o un medidor de potencia. En este apartado se explicará la calibración utilizando los detectores *LDR*, la cual es una forma sencilla y económica de calibrar cuando no se cuenta con un medidor de potencia. Los pasos para calibración utilizando los detectores *LDR* son:

1. **Fijar el láser.** El diodo láser se ajusta a cierta altura y sobre una línea a la mesa óptica. En este punto el láser se encuentra encendido en modo continuo para para establecer la línea que seguirán las demás componentes.

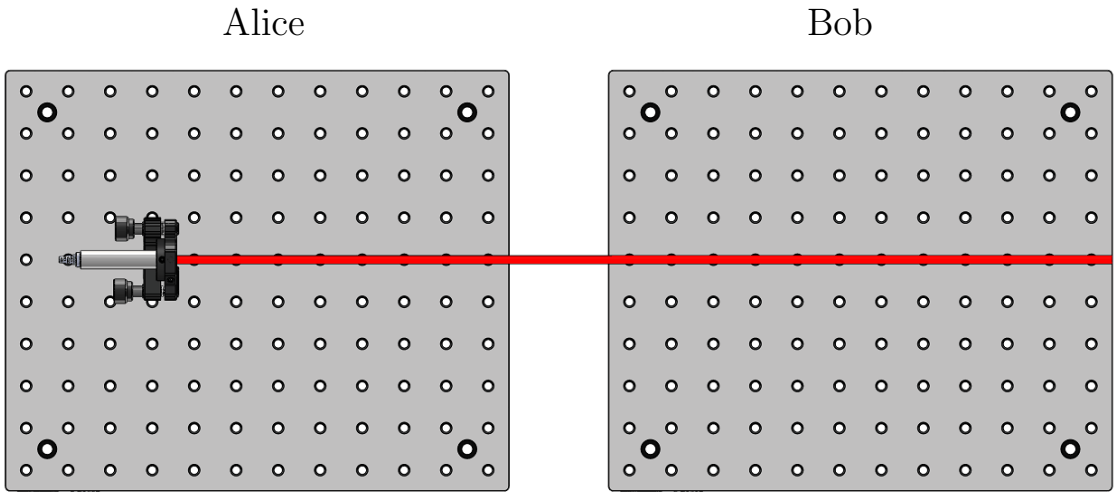


Figura 3.14: Paso 1 de la calibración: Fijar el láser (Elaboración propia).

2. **Ajustar la polarización del láser.** Fijar el cubo divisor de haz y los detectores *LDR* en la segunda mesa óptica. Estos se colocan en el camino óptico del láser, de tal forma que el haz incide en el centro del cubo divisor de haz. Los detectores *LDR* se colocan perpendicularmente al haz láser de tal forma que la luz incide directamente en los orificios de los detectores. Como el láser se encuentra polarizado, se orienta el láser de forma que al pasar a través del cubo divisor de haz el detector 0 registre la mayor intensidad de luz.

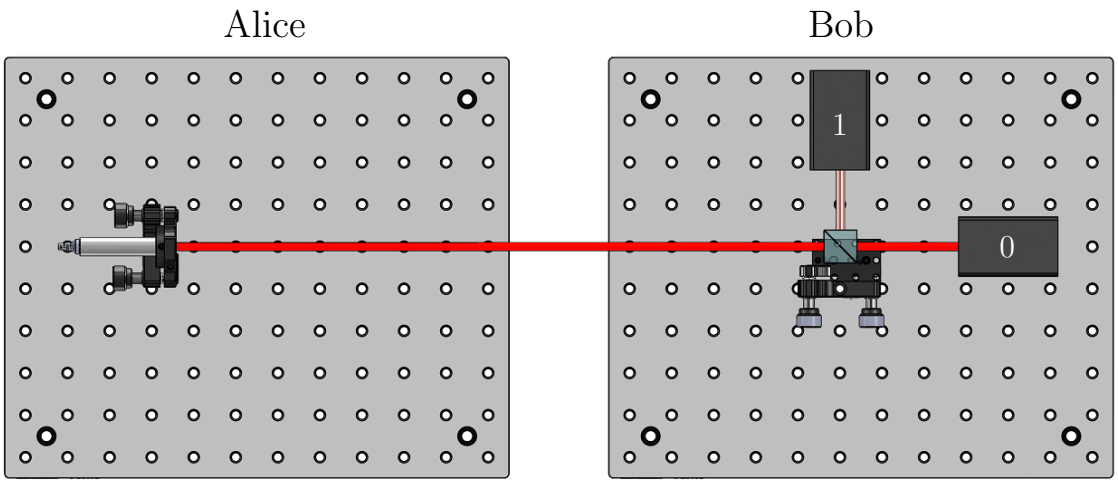


Figura 3.15: Paso 2 de la calibración: Ajustar la polarización del láser (Elaboración propia).

3. **Orientar el retardador de onda $\lambda/2$ de Alice.** Colocar el retardador de onda $\lambda/2$ en la mesa óptica de Alice de tal forma que el haz láser incide en el centro. En este punto los sensores registran diferente intensidad de luz. Manualmente se gira el retardador de onda $\lambda/2$ hasta que el detector cero registra la mayor intensidad de luz, el cual corresponde al ángulo 0° de polarización del retardador de onda de Alice. Luego se coloca el servomotor para rotar el retardador de onda a los distintos grados de polarización.

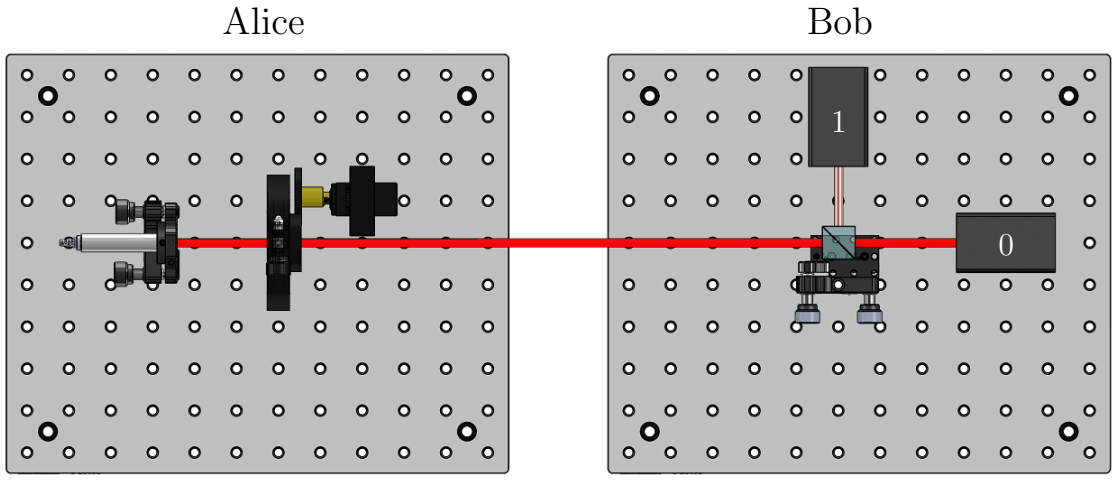


Figura 3.16: Paso 3 de la calibración: Orientar el retardador de onda $\lambda/2$ de Alice (Elaboración propia).

4. **Orientar el retardador de onda $\lambda/2$ de Bob.** Fijar el retardador de onda en la mesa óptica de Bob. De igual forma el haz láser debe incidir en el centro del retardador de onda $\lambda/2$. Nuevamente se gira manualmente el retardador de onda hasta que el detector *LDR* cero registra la mayor intensidad de luz. La posición final corresponde al ángulo 0° de polarización del retardador de onda de Bob. Luego se coloca el servomotor para rotar el retardador de onda $\lambda/2$ a los distintos grados de polarización.

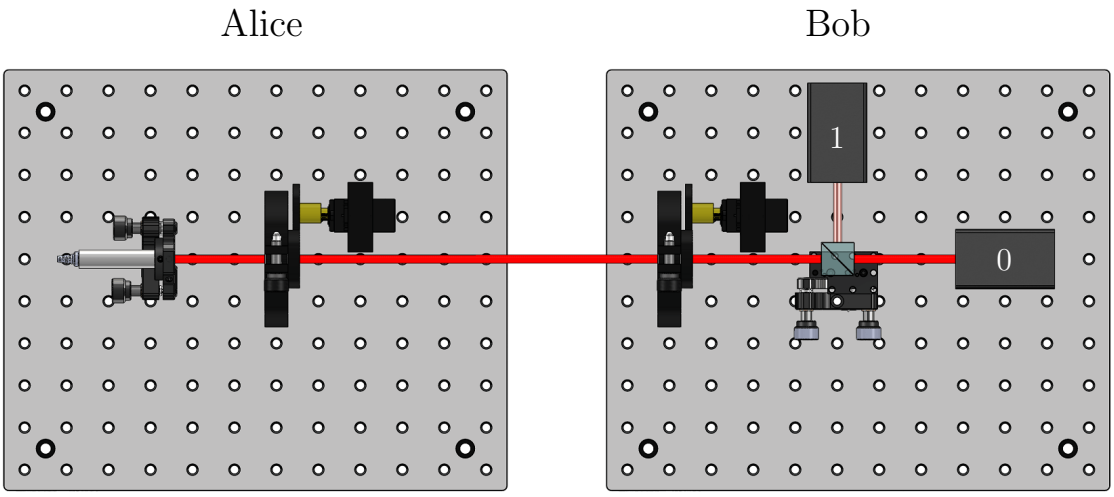


Figura 3.17: Paso 4 de la calibración: Orientar el retardador de onda $\lambda/2$ de Bob (Elaboración propia)

Luego se comprueba que se cumplen los casos mostrados en la Tabla 3.1 para las distintas polarizaciones. Cuando Alice y Bob utilizan la misma base, uno de los detectores registrará el 100 % de probabilidad de medir correctamente. Por ejemplo, si Alice envía la información en la base $\otimes$ con la polarización 45° y Bob mide en la base $\otimes$ con polarización de 45°, entonces la polarización resultante es 90°. Por lo que la luz, al pasar por el PBS será desviado hacia el detector 1. En el caso de que se utilicen bases distintas, los detectores registrarán con una probabilidad del 50 %. Es decir, si Alice envía la información en la base $\otimes$ con la polarización 45° y Bob mide en la base $\oplus$ con la polarización 0°, entonces la polarización resultante es 45°. Luego la luz polarizada al pasar por el cubo divisor de haz es desviada hacia hacia los detectores con la misma probabilidad. La Tabla 3.1 muestra las mediciones que se obtienen para un sistema ideal. Sin embargo, los ángulos reales se encuentran obteniendo las mediciones de los detectores *LDR* y trazando un gráfico superpuesto para las distintas polarizaciones.

Tabla 3.1: Mediciones que se obtienen de acuerdo a las configuraciones posibles de las bases y los ángulos de polarización teóricos (Thorlabs, 2017).

Alice			**Bob**			
Base	*Bit*	Ángulo	Base	Ángulo	Detector 0	Detector 1
⊕	0	0°	⊕	0°	100 %	0 %
⊕	1	90°	⊕	0°	0 %	100 %
⊗	1	45°	⊕	0°	50 %	50 %
⊗	0	−45°	⊕	0°	50 %	50 %
⊕	0	0°	⊗	45°	50 %	50 %
⊕	1	90°	⊗	45°	50 %	50 %
⊗	1	45°	⊗	45°	0 %	100 %
⊗	0	−45°	⊗	45°	100 %	0 %

Para encontrar los ángulos de polarización de Bob se utilizaron los códigos del apéndice K. Además, para encontrar los ángulos de Bob es necesario que en la configuración de Alice el láser se mantenga en continuo y el retardador de onda $\lambda/2$ se mantenga en el ángulo 0. Bob varía su retardador de onda $\lambda/2$ de grado en grado y al mismo tiempo registra cada medición de los detectores *LDR*. Al final se obtienen las mediciones experimentales de cada detector, los cuales se grafican y se realiza un ajuste de interpolación. Una vez encontrados los ángulos para Bob, se tiene que volver a la configuración dada por la Figura 3.17 para encontrar los ángulos de Alice. Para esto, se deja fijo el retardador de onda de Bob en el ángulo 0° y se mueve el retardador de onda de Alice en pasos de un grado. Mientras tanto se obtienen las mediciones y se genera un segundo gráfico para analizar y encontrar los ángulos de Alice. Para el ajuste de interpolación es necesario normalizar los datos experimentales. El gráfico que se obtiene es de tipo sinusoidal y para encontrar los ángulos de cada retardador de onda es necesario identificar los máximos, mínimos y las intersecciones de ambos detectores *LDR*.

3.4 Implementación del proyecto en el museo "El péndulo de Foucault"

El prototipo experimental de criptografía cuántica se diseñó con el objetivo de ser utilizado y atendido por el público en general, desde niños, jóvenes y adultos. Debido a esto se construyó tomando en cuenta las características de un prototipo didáctico, atractivo e interactivo para público no especializado. Para esto se realizó una interfaz gráfica que permite a los usuarios interactuar con el experimento y comprender todos los procesos implicados en distribución de claves cuánticas (QKD).

Mediante programación en Python se diseñó la interfaz gráfica que permite visualizar los pasos del experimento de criptografía cuántica. Para crear la interfaz se utilizó la librería $PySimpleGUI$, que es un paquete para crear interfaces gráficas atractivas para los usuarios. La Raspberry Pi 4 cuenta con conexión *HDMI*, lo que permitió conectar una pantalla para mostrar los pasos del Protocolo BB84 y el mensaje recibido con la interfaz gráfica. También se conectó un teclado para cada uno de los interlocutores, el cuál permite interactuar con el experimento y elegir entre las diferentes opciones de la interfaz. El programa de la interfaz se divide en tres opciones que se describen a continuación:

- **Opción 1: Verificar si hay espía y generar la llave:** Esta opción permite visualizar los pasos del protocolo BB84 para generar y distribuir la llave. Para la creación de la clave se utiliza un mensaje de prueba, el cual debe ser mayor a uno y menor a 10 caracteres. Esta restricción se debe a que entre mayor número de caracteres, mayor es el tiempo necesario en la creación de la llave. De esta forma es posible que más personas puedan utilizar el experimento. Además se realiza el análisis de error para detectar la presencia del espía. En el caso de que el porcentaje de error exceda el 25 %, se detecta la presencia de Eve y no se genera la llave. En caso contrario, el canal es seguro y se guarda la llave para posteriormente enviar el mensaje en la opción 2. Debido a que no existe ruido o pérdidas en el canal de comunicación y a que no existe la presencia de algún espía, el error siempre será 0 %. Sin embargo, es posible simular al espía y obtener un porcentaje de error

mayor a 25 % colocando un dispositivo que modifique la información enviada en el canal de comunicación entre Alice y Bob. Por ejemplo, se puede colocar un dispositivo como un retardador de onda que cambie la polarización de los pulsos de luz como lo haría un espía estando presente en el canal de comunicación. Al final de esta opción se muestran los *bits* y las bases elegidas de Alice y Bob y la llave creada.

- **Opción 2: Enviar mensaje a Bob:** Después de verificar si el canal es seguro, podemos elegir la siguiente opción, que es enviar el mensaje secreto. En esta opción se convierten los caracteres del mensaje a binario y se realiza la suma *XOR* con la llave creada en la opción 1. En esta opción se ajusta la longitud del mensaje con la llave para cifrar la información y finalmente ser enviado por el canal clásico.

  Bob recibe el mensaje que le ha enviado Alice utilizando la llave y las bases creadas en el paso anterior, por lo cual es necesario que se haya ejecutado al menos una vez la primera opción: *"Verificar el canal, generar una llave y enviar mensaje"* o la tercera opción: *"Verificar si hay espía y generar una llave"*.

  Para el envío del mensaje cifrado se puede utilizar un canal de comunicación clásico, por ejemplo a través de la comunicación *Bluetooth*. Sin embargo, para fines demostrativos se utilizó los pulsos de luz a través del aire para transmitir el mensaje cifrado, que es el mismo por el cual se envió la llave.

- **Opción 3: Verificar el canal, generar la llave y enviar mensaje:** Esta opción realiza los dos pasos anteriores en una sola opción. Esta opción es más tardada que las dos anteriores porque que realiza todos los pasos necesarios para distribuir la clave cuántica y posteriormente enviar el mensaje.

Esta división se realizó con la finalidad de comprender los pasos que implica cada opción y que el usuario pueda elegir la opción a realizar de acuerdo al tiempo que dispone. Dentro de las opciones de la interfaz gráfica, se pide al usuario que interpreta a Alice introducir un mensaje prueba que sirve para generar la llave o para enviar el mensaje a Bob. En el mensaje no se puede escribir vocabulario inapropiadas debido a que cuenta con la librería de Python *Spanlp*, la cual bloquea este tipo de palabras (Puentes, 2021).

Debido a que el sistema es interactivo con el usuario, se implementó un sistema de protección en el prototipo experimental. Se diseñaron dos cajas de acrílico para colocar todas las componentes de Alice y Bob, esto con el objetivo de proteger las componentes ópticas y electrónicas y evitar cualquier modificación que pueda alterar el sistema. También se diseñaron tomando en cuenta que el láser puede ser dañino si una persona expone directamente la vista a la luz del láser, por lo que las cajas también sirven de protección para el usuario. El diseño de las cajas se muestra en la Figura 3.18

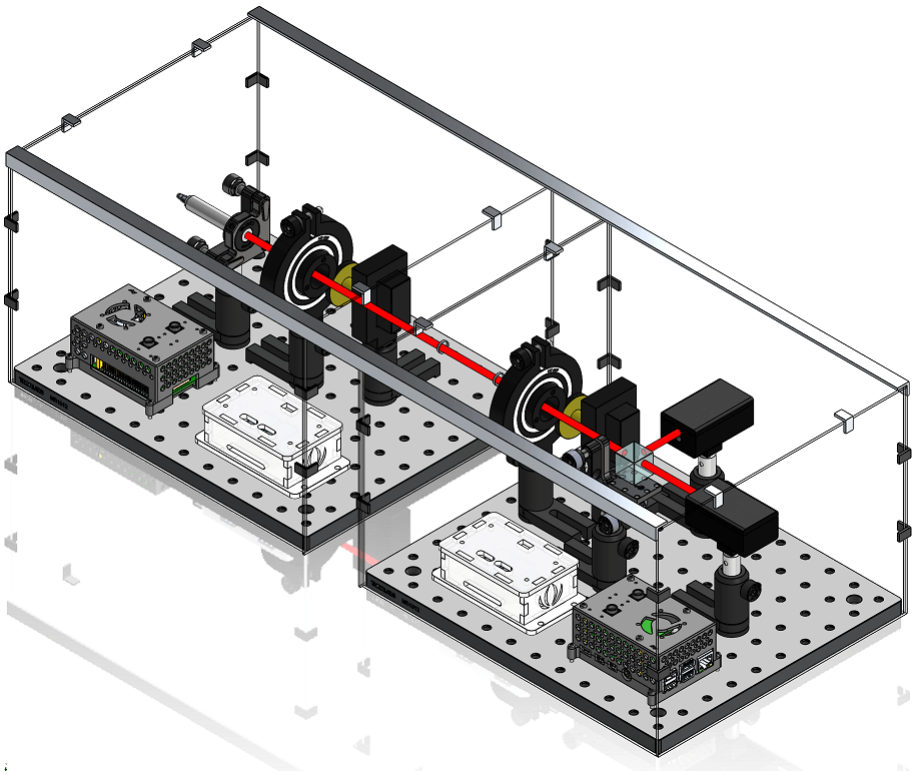


Figura 3.18: Prototipo experimental de criptografía cuántica con las cajas de protección (Elaboración propia).

Se diseñaron carteles científicos para acompañar la demostración y para la comprensión del funcionamiento del prototipo experimental de criptografía cuántica. Los carteles contienen información de las componentes, los pasos del Protocolo BB84, el proceso de cifrado y los procesos físicos implicados en *QKD*. Además, se realizó un manual de operación para los encargados del prototipo en el museo. El manual contiene las componentes utilizadas, el encendido y apagado del prototipo, la explicación de cada paso en el experimento y como usarlo correctamente.

Capítulo 4
Resultados y Discusiones

En esta sección se muestran los resultados obtenidos del prototipo experimental automatizado de criptografía cuántica. Primero se describen las mejoras implementadas de las componentes para nuestro prototipo experimental y se comparan sus características con los otros sistemas experimentales de *QKD*. Principalmente se disminuyó el tiempo en la generación de la clave comparado con el kit comercial. Luego se muestran los ángulos obtenidos en la calibración para cada uno de los retardadores de onda de los interlocutores. Finalmente se muestran todas las componentes y los carteles del experimento instaladas en el museo del péndulo de Foucault.

4.1 Componentes del experimento automatizado de Criptografía Cuántica

Se ha mencionado que existen sistemas comerciales de criptografía cuántica, tal como *"Cerberis XG QKD System"* de *ID Quantique* o *"EDU-QCRY1"* de Thorlabs (kit comercial). El sistema *Cerberis XG QKD System* es un sistema comercial automatizado de costo elevado, por lo que se propuso construir un sistema que demuestre los principios físicos, tecnológicos y computacionales presentes en un experimento de criptografía cuántica, tal como el kit comercial. Sin embargo, el kit comercial es un prototipo manual donde se vuelve tedioso y complicado realizar las pruebas experimentales. Por ejemplo, para generar una llave de 40 *bits* de longitud, es necesario enviar 80 *bits* y se necesita

en promedio un tiempo de 13 minutos. Estas pruebas se realizaron con dos personas para el control de cada interlocutor y utilizando programas para el registro y análisis de los datos.

Para reducir el tiempo en cada prueba es necesario implementar ciertas componentes como la parte mecánica de los retardadores de onda y los detectores para poder automatizar el sistema de criptografía cuántica. Es posible implementar otros detectores para realizar las mediciones, como los sensores de potencia, los cuales reaccionan a una longitud de onda en particular, o mediante detectores térmicos que se basan en medir la energía térmica de la luz. La desventaja de ambos es que tienen un costo elevado y particularmente los sensores de potencia necesitan de instrumentos especiales para su lectura. En la Tabla 4.1 se observan algunas de las características de los detectores considerados para el experimento de criptografía cuántica.

Tabla 4.1: Características de los detectores considerados para las mediciones de Bob (Elaboración propia).

Detectores	Sensor térmico TD15A	Sensor de potencia S122C	Módulo *LDR*
Marca	Thorlabs	Thorlabs	Comercial
Medición	Energía térmica	Potencia óptica	Resistividad
Banda muerta	0.6 s	$< 0{,}6\ \mu$s	20 ms

De acuerdo a la Tabla 4.1, el detector con el menor tiempo de reacción para registrar cambios en el detector (banda muerta) es el sensor de potencia. Sin embargo, el tiempo mínimo para la detección no solo depende del sensor, sino también de los servomotores. Debido a que se utilizó el servomotor HB3001, el cual tiene una velocidad promedio de 60 grados/s, no se necesita de tanta resolución por lo que descartamos el sensor de potencia. Por ese motivo, los módulos *LDR* son buenas opciones. La desventaja de los detectores *LDR* es que miden la intensidad de luz que incide en el detector y pueden ser afectados por cualquier longitud de onda. Debido a ello fue necesario diseñar sistema para no alterar la medición debido a fuentes de radiación externa. Otro factor

significativo fue la diferencia de costos, siendo los módulos *LDR* la mejor opción para este trabajo.

Para el control mecánico del retardador de onda se analizaron diferentes opciones que se muestran en la Tabla 4.2. En esta tabla se observa que la opción con mayor velocidad, precisión y durabilidad es la montura ELL14K. Además, esta opción permite su control a través de diversos dispositivos. Un factor importante que representa una desventaja para la montura ELL14k es que el proyecto estará en uso permanente en el museo, lo que significa que el control mecánico debe de tener la mayor duración pero a un costo accesible. Sin embargo, el precio de la montura ELL14k es superior al de las otras opciones, lo que vuelve costoso su mantenimiento al finalizar su vida útil. Por otra parte, el servomotor 3001HB cuenta con una vida útil similar a la de la montura ELL14K y son mucho más económicos, volviéndolos un buen candidato para realizar los mantenimientos. No obstante, el servomotor presenta tres desventajas: el grado máximo de giro es de 180° lo que vuelve más complicada la calibración; no cuenta con una montura para sostener el retardador de onda $\lambda/2$; y las velocidades son mucho menores que las alcanzadas por la montura ELL14K. Sin embargo, para este experimento no es necesario exceder el ángulo máximo de giro y es posible realizar un acoplamiento mecánico con el retardador de onda $\lambda/2$, tal como se mostró anteriormente. Estas adaptaciones con el servomotor son la opción más económica. Considerando que el experimento de criptografía cuántica se utiliza con fines demostrativos, es suficiente con mejorar el tiempo de realización de cada prueba en comparación con las del kit comercial. Es por esto que el servomotor 3001HB representa la mejor opción para este trabajo.

En la Figura 4.1 se observa el tiempo promedio que se requiere para el envío de 80 *bits* con el kit comercial y nuestro prototipo experimental. Se observa que nuestro prototipo tarda en promedio 2 minutos para realizar el envío de la clave. Sin embargo, los datos obtenidos consideran el tiempo de espera de cada pantalla donde se muestra al usuario el paso que se está realizando. Es posible disminuir el tiempo que toma realizar todo el experimento, pero por fines didácticos se alargó la duración del experimento para que los usuarios tuvieran suficiente tiempo de visualizar los pasos y leer la explicación en las pantallas. Sin la interfaz gráfica el tiempo total puede reducirse a la mitad.

Tabla 4.2: Características los sistemas de rotación considerados para los retardadores de onda $\lambda/2$ de Alice y Bob (Elaboración propia).

Sistema de rotación	Montura ELL14K	Montura PRM1Z8	Montura con servomotor 3001HB
Marca	Thorlabs	Thorlabs	Power HD
Velocidad de rotación	430 grados/s	25 grados/s	60 grados/s
Grado máximo de giro	720°	359°	180°
Movimiento incremental mínimo	±0,002°	±1°	±1°
Vida útil	600K revoluciones	-	500K revoluciones
Lenguaje de programación	Teléfono/ Software Elliptec/ Aplicación personalizada (C#, C++)	C++	Python/C++/...
Requiere controlador externo	-	K-Cube^{TM} Strain Gauge Reader	Adaptaciones 3D y mecánicas

La Figura 4.2 muestra el tiempo promedio que requiere cada experimento para enviar una secuencia de caracteres (2-10). En esta Figura se compara nuestro prototipo experimental y el sistema de *ID Quantique*. El eje $\hat{x}$ empieza a partir de dos caracteres, ya que puede no generarse una clave si se envía solamente uno, por el proceso de eliminación de *bits* en la verificación del canal. Los datos correspondientes al sistema *QKD* de la empresa *ID Quantique* se obtuvieron considerando que la velocidad para la generación de la llave es de (2 KB/s) (ID Quantique, 2022). Por lo tanto, el tiempo promedio para el envío de 10 caracteres es de 0.04 segundos. Para nuestro prototipo

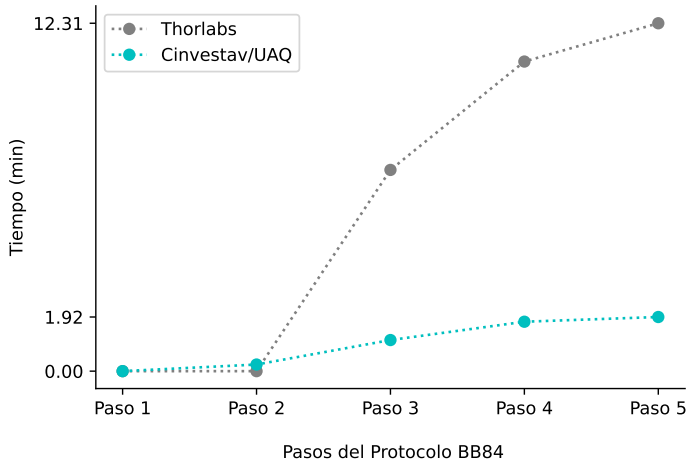


Figura 4.1: Tiempo necesario para crear una clave de longitud aproximada de 40 *bits* realizando todos los pasos del Protocolo BB84. Para crear una clave de esta longitud se requiere enviar un mínimo de 80 *bits*. El eje $\hat{x}$ representa los pasos de Protocolo BB84 y el eje $\hat{y}$ muestra el tiempo total de cada sistema para realizar todos los pasos (Elaboración propia).

experimental el tiempo para el envío de caracteres se obtuvo del promedio de realizar 20 experimentos. Durante el envío de los *bits* se realiza un proceso para verificar que nuestro sistema siga calibrado, lo que genera fluctuaciones en el tiempo del envío, observe la Figura 4.3.

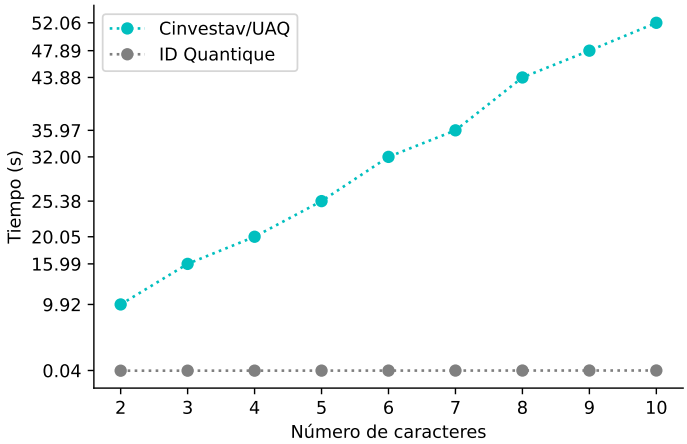


Figura 4.2: Comparación del tiempo promedio que se necesita para enviar una secuencia de caracteres (2-10) con el sistema *ID Quantique* y nuestro prototipo experimental. Cada carácter equivale a enviar 8 bits (Elaboración propia).

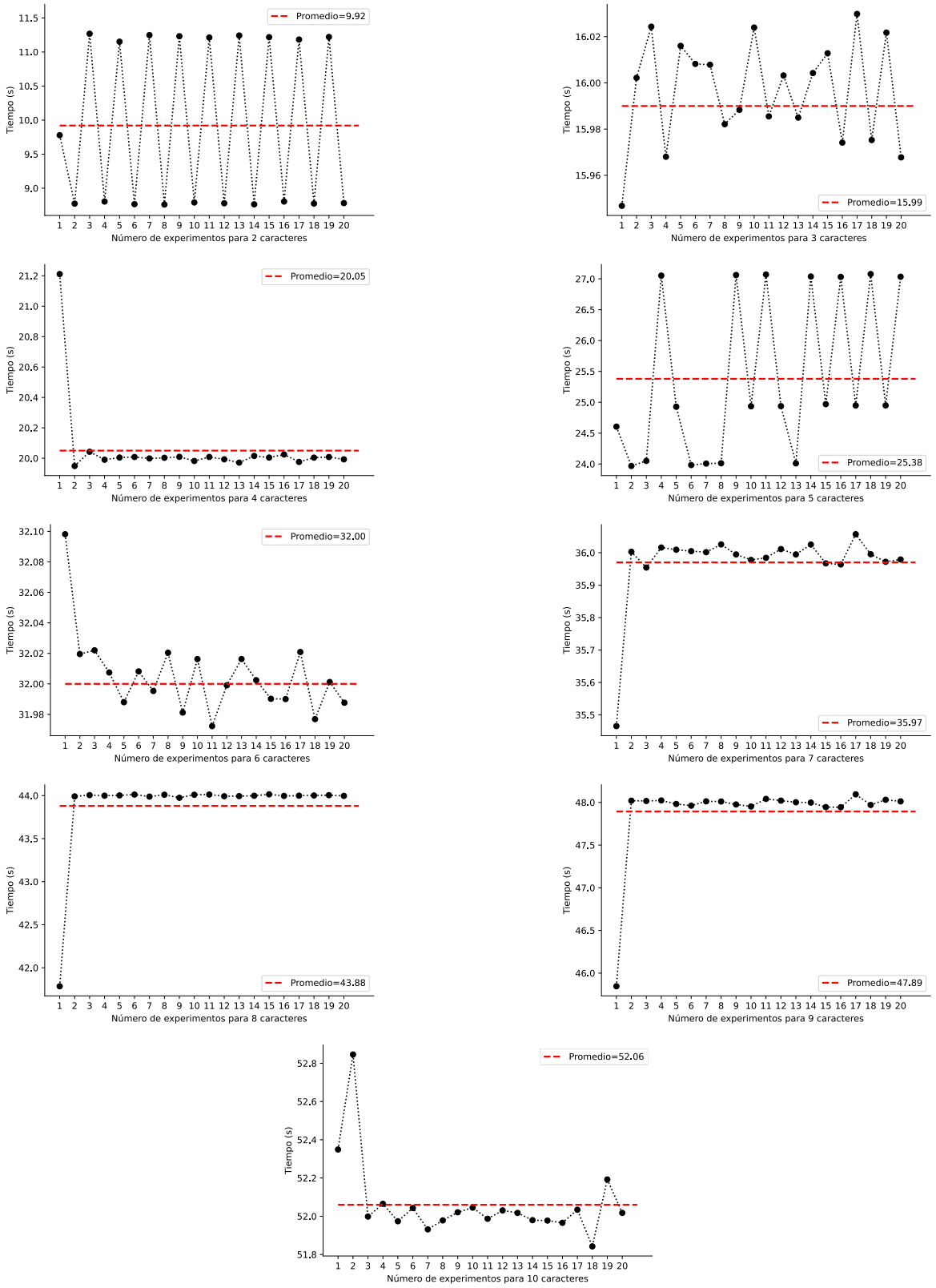
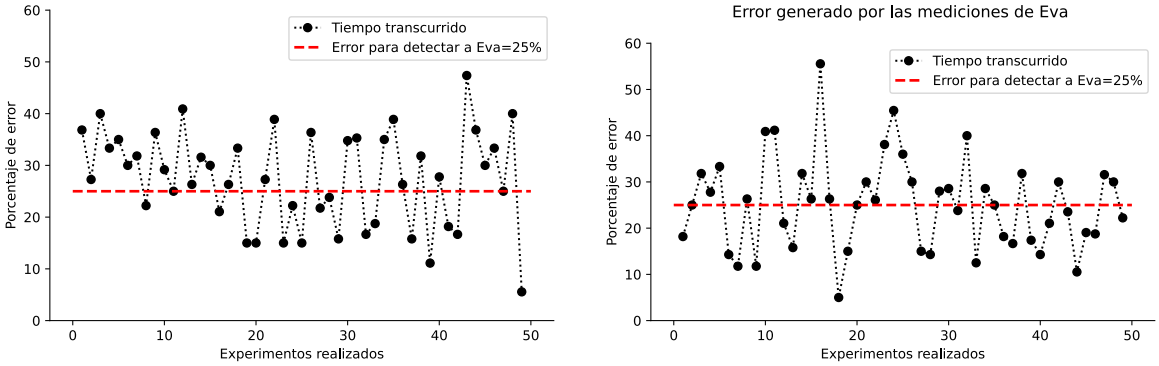


Figura 4.3: Tiempo necesario para enviar un determinado número de caracteres (2-20) promediado de 20 pruebas experimentales para cada secuencia (Elaboración propia).

Algo importante a mencionar es que para generar los números aleatorios se pueden implementar distintas librerías con este propósito. *Qiskit* es una librería que trabaja con computadoras cuánticas para simular e implementar un sistema cuántico en Python. Esta librería permite implementar números aleatorios basados en una simulación de un sistema cuántico, lo cual mejora la aleatoriedad en comparación con los números pseudoaleatorios que se generan con algoritmos clásicos. Un sistema con números pseudoaleatorios puede generar coincidencias en las bases aleatorias, lo cual provocaría que no se detecte al espía en el proceso. En la Figura 4.4 se muestran los porcentajes de error obtenidos por la intrusión del espía con dos librerías distintas para la generación de las bases y *bits* aleatorios de 50 experimentos. En la Figura se observa que se obtienen mejores resultados con la librería *Qiskit*, ya que se detecta más veces la presencia de Eve. Sin embargo, para obtener los números aleatorios con esta librería se requiere más tiempo computacional, en comparación con la librería *Random*. Debido a que nuestro experimento no implementa a Eve y a que se requiere el menor tiempo en la generación de los bits y bases aleatorias, se eligió la librería *Random*. En un sistema real es necesario utilizar un método para generar valores completamente aleatorios y así garantizar que el envío de la clave sea seguro.



(a) Librería cuántica "Qiskit"

(b) Librería clásica "Random"

Figura 4.4: Porcentaje de error generado por la intrusión de Eve simulado con dos librerías en Python: *Qiskit* y *Random*. En cada experimento se generaron y simularon un total de 80 *bits* aleatorios para crear la clave. La librería cuántica detectó al espía en 32 de 50 experimentos, mientras que la librería clásica detectó al espía sólo 28 de 50 experimentos (Elaboración propia).

Las características de los sistemas de *QKD* mencionados anteriormente se muestran en la Tabla 4.3. Nuestro prototipo mejora significativamente la velocidad para la transferencia de *bytes* en comparación con el kit comercial, debido a que se automatizaron los procesos que se hacían manualmente como: pulsos del láser, rotación de los retardadores de onda $\lambda/2$, registro de las mediciones y análisis de error. Sin embargo, la velocidad de transmisión de datos de el *Cerberis XG QKD System* es mayor que la del nuestro. La desventaja del *Cerberis XG QKD System* es su costo elevado lo que implica que sea más difícil adquirirlo. El costo de nuestro prototipo experimental toma en cuenta no sólo el sistema *QKD* sino también las pantallas, teclados y mouse, por lo que es posible reducir el costo con otro tipo de *hardware* para visualizar los datos. También es posible utilizar piezas 3D o tarjeta electrónica de menor costo. El alcance de nuestro prototipo no debía de ser mayor a 1 m por lo que la comunicación *Bluetooth* funcionaba perfecto para este propósito. Debido a que se utilizó la comunicación *Bluetooth*, la distancia máxima posible son 100 m de distancia entre los interlocutores. Sin embargo es posible cambiar este tipo de comunicación para aumentar la distancia hasta el alcance del láser.

Tabla 4.3: Comparación de las características de dos sistemas de criptografía cuántica comerciales y nuestro prototipo experimental (Elaboración propia).

Características de los sistemas de *QKD*			
Nombre	EDU-QCRY1	Cerberis XG QKD System	Prototipo QKD
Empresa	Thorlabs	ID Quantique	Cinvestav/UAQ
Canal cuántico	Espacio libre	Fibra óptica	Espacio libre
Canal clásico	-	Conexión a la red	Conexión *Bluetooth*
Velocidad de trasmisión (Byte/s)	0.02 B/s	2 KB/s	0.2 B/s
Propiedad Física	Luz Polarizada	Estados coherentes	Luz Polarizada

El sistema automatizado *Cerberis XG QKD System* utiliza los estados coherentes para el envío de información. Esto presenta una gran diferencia con nuestro prototipo experimental construido, y lo que marca la diferencia de precios entre un sistema de la industria y un prototipo experimental. Nuestro prototipo experimental funciona como un sistema de criptografía clásico debido a que utilizamos la polarización de pulsos de luz láser. Esto debilita la seguridad de nuestro sistema en caso de que existiera un dispositivo que interfiriera en el canal de comunicación. Para realizar un sistema cuántico en nuestro sistema, se podría agregar un atenuador en la configuración de Alice para producir pulsos de luz que en promedio contienen un sólo fotón, mientras que para Bob tendría que sustituirse los módulos *LDR* por detectores de un sólo fotón.

4.2 Calibración del experimento

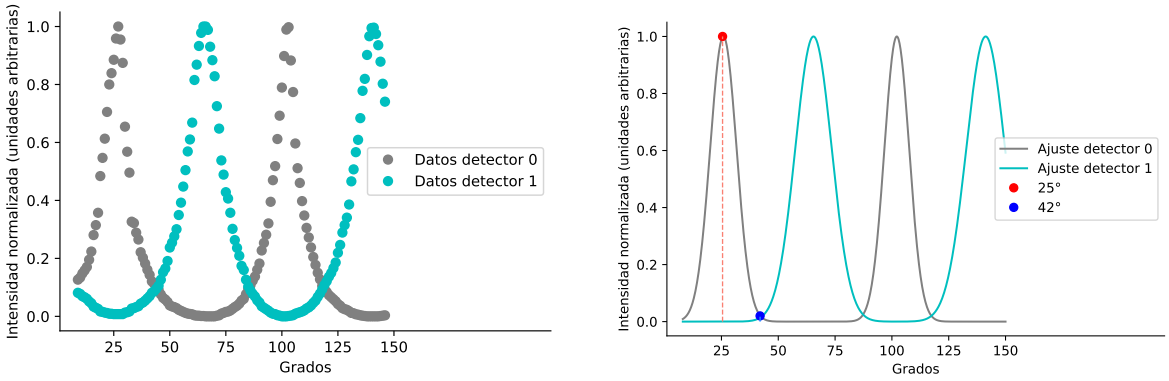
Después de montar las componentes y el sistema de control es necesario realizar la calibración. De acuerdo a la teoría del protocolo BB84 se requieren dos pares de bases ortogonales, por ejemplo las bases $\bigoplus$ $(0°, 90°)$ y $\bigotimes$ $(-45°, 45°)$. Los ángulos reales para los que se debe de rotar el retardador de onda $\lambda/2$ no corresponden con los ángulos teóricos. Estos ángulos se obtuvieron midiendo las intensidades en los detectores *LDR* para los distintos ángulos. Una vez obtenidos los datos, estos se normalizaron y se realizó un ajuste para encontrar los puntos de interés (máximos, mínimos e intersecciones). De acuerdo a los datos obtenidos, se realizó un ajuste por mínimos cuadrados en Python. La función modelo para nuestros datos es:

$$f_{a,b,c,d}(x) = e^{-(x-a)^2/b^2} + e^{-(x-c)^2/d^2} \tag{4.1}$$

Los parámetros obtenidos para el ajuste de Bob se muestran en la Tabla 4.4. Calculando los parámetros de la función se pueden determinar los máximos, mínimos e intersecciones. Esto permite encontrar los valores experimentales a los cuales hay que orientar el retardador de onda $\lambda/2$ para obtener los valores de la Tabla 3.1. Los puntos de interés de las funciones para los ángulos de Bob están dados por 25° y 42°. Estos representan a los ángulos teóricos 0° y 45°, respectivamente. En la Figura 4.5 se pueden observar los datos obtenidos por los detectores de Bob y el ajuste realizado.

Tabla 4.4: Parámetros obtenidos para la función 4.1 con el ajuste de datos para Bob (Elaboración propia).

Parámetros	**a**	**b**	**c**	**d**
Detector 0	25.848 ± 0.090	8.227 ± 0.104	102.157 ± 0.087	7.706 ± 0.100
Detector 1	65.5 ± 0.075	11.341 ± 0.086	141.255 ± 0.174	12.085 ± 0.212



(a) Datos obtenidos de los detectores.

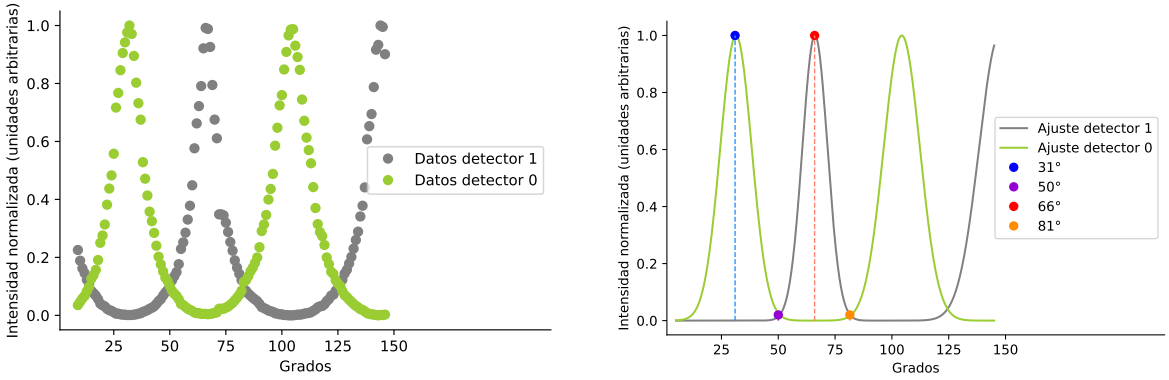
(b) Ajuste de datos.

Figura 4.5: La Figura 4.5a muestra los datos de las mediciones obtenidas por los detectores para la calibración del retardador de onda $\lambda/2$ de Bob. La toma de datos se realizó tomando la medición de grado en grado. La Figura 4.5b muestra el ajuste realizado y los puntos mostrados en el gráfico corresponden a los ángulos a los cuales hay que orientar el retardador de onda. Bob sólo necesita dos ángulos, esto corresponde al punto donde la función del detector 0 es máximo y el punto donde se intersectan ambas funciones (Elaboración propia).

Los parámetros obtenidos para el ajuste de Alice se muestran en la Tabla 4.5. Los puntos de interés de las funciones para el ajuste de Alice están dados por los ángulos 31°, 50°, 66° y 81° que corresponden a los ángulos teóricos 0°, 45° y 90°, -45°. Los datos de las mediciones de los detectores y el ajuste con los puntos de interés se muestran en la Figura 4.6.

Tabla 4.5: Parámetros obtenidos para la función 4.1 con el ajuste de datos para Alice (Elaboración propia).

Parámetros	**a**	**b**	**c**	**d**
Detector 0	66.141 ± 0.084	7.819 ± 0.097	147.424 ± 0.392	12.828 ± 0.453
Detector 1	31.394 ± 0.072	9.785 ± 0.083	104.360 ± 0.077	11.260 ± 0.089



(a) Datos obtenidos de los detectores.

(b) Ajuste de datos.

Figura 4.6: La Figura 4.6a muestra los datos de las mediciones tomadas por los detectores para la calibración del retardador de onda de Alice. La Figura 4.6b muestra el ajuste realizado y los puntos mostrados en el gráfico corresponden a los ángulos a los cuales hay que orientar el retardador de onda. Alice necesita cuatro ángulos para generar dos bases, estos corresponden a los puntos donde la función del detector 0 es máximo, el detector 1 es mínimo y a los puntos donde se intersectan ambas funciones. (Elaboración propia).

Los módulos *LDR* son sensores que detectan la intensidad de la luz que incide sobre la fotoresistencia. Cuando el pulso de luz se divide en dos al pasar a través del cubo divisor de haz y llega a ambos detectores, las intensidades que registran ambos detectores son altas, provocando que la intersección de las gráficas sea en un punto cercano a cero.

4.3 Implementación del proyecto en el museo "El péndulo de Foucault"

A continuación se muestran las pantallas de la interfaz cuando el usuario interactúa con el prototipo experimental de criptografía cuántica. Al inicio del programa se muestra la pantalla de bienvenida, observe la Figura 4.7.

Figura 4.7: Pantallas de bienvenida de la interfaz gráfica (Elaboración propia).

Como Alice es el emisor de información, en su pantalla aparecerá el botón de *"Comenzar"*. Este botón permite dar inicio al programa para las dos pantallas. Después de presionar el botón, la pantalla de Bob cambiará a una imagen de espera mientras que la pantalla de Alice cambiará a una nueva en la cual se puede elegir diferentes opciones, observe la Figura 4.8.

Figura 4.8: Pantalla para la selección de opciones y de la pantalla de espera (Elaboración propia).

Si se selecciona la primera opción *"Verificar si hay espía y generar una llave"*, en Alice se muestra un recuadro para ingresar un mensaje de prueba con el cual se generará la llave. La pantalla de Bob permanece igual, observe la Figura 4.9.

Figura 4.9: Pantalla para ingresar el mensaje prueba (Elaboración propia).

Después de seleccionar el botón de *"Generar llave"*, la pantalla mostrará el proceso que se realiza a lo largo del experimento. A continuación se muestran algunos de estos pasos para la pantalla de Alice, observe la Figura 4.10.

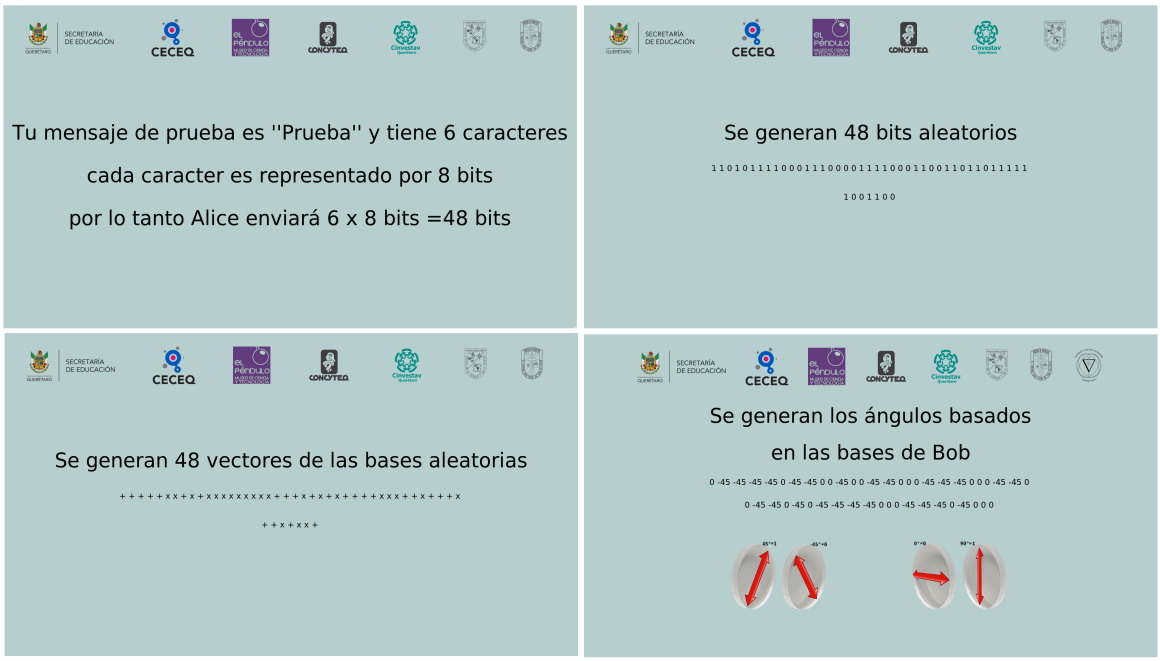


Figura 4.10: Ventanas que aparecen en la pantalla de Alice al seleccionar el botón de *"Generar llave"* (Elaboración propia).

La pantalla de Bob se crean ventanas similares a la Figura 4.10, con la diferencia de que Bob muestra la cantidad de caracteres y número de *bits* a recibir. Después de estás pantallas, Alice y Bob empiezan la transferencia de los *bits* y el experimento empieza moverse automáticamente. En ese momento las pantallas de Alice y Bob muestran el porcentaje de la comunicación, observe la Figura 4.11.

Al finalizar la transferencia de *bits*, Alice y Bob se comunican por medio de

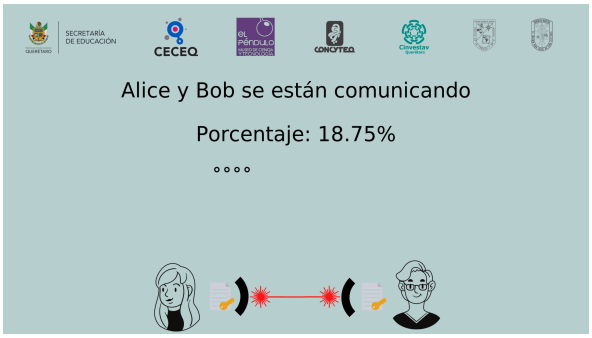


Figura 4.11: Pantalla del progreso de comunicación para la transferencia de *bits* entre Alice y Bob (Elaboración propia).

Bluetooth para verificar si el canal es seguro y crear la llave. En la pantallas se muestran los pasos de verificación del Protocolo BB84, observe la Figura 4.12.

Figura 4.12: Pantallas de Alice para la verificación del canal (Elaboración propia).

Alice envía a Bob el porcentaje de error. Si el canal es seguro, el porcentaje de error es cero y se crea la llave. Si se obtiene un error mayor al 25 % se le indica al usuario que el canal es inseguro y es necesario crear nuevamente la clave y las bases, tal como en la Figura 4.13. Después se reinicia el experimento.

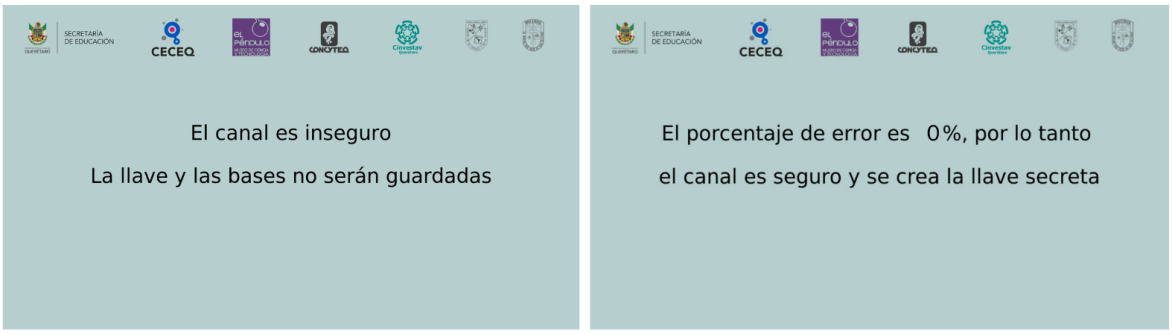


Figura 4.13: Resultados del porcentaje de error para la verificación del canal. Se muestran los casos cuando se crea la clave y el canal es seguro (derecha) y cuando no se crea la clave y el canal es inseguro (izquierda) (Elaboración propia).

En el caso de que el canal sea seguro entonces se muestra al usuario en la pantalla las bases, los *bits* y llave obtenidos en el proceso del protocolo BB84, observe la Figura 4.14.

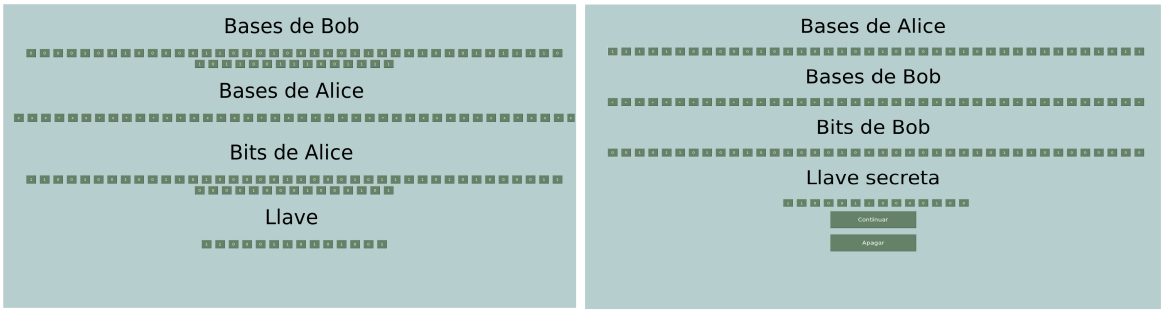


Figura 4.14: Pantallas de ambos interlocutores cuando el canal es seguro. Se muestran las bases y los *bits* de cada interlocutor y la llave generada por medio del Protocolo BB84 (Elaboración propia).

Aquí finaliza la opción 1. En la pantalla de Bob seleccionamos el botón de *Continuar* para volver al inicio donde puede seleccionar la opción 2 o 3 como se muestra en la Figura 4.8. La segunda opción realiza el mismo procedimiento de los pasos de la opción 1 para enviar los pulsos de luz y enviar la información con la diferencia que al final se muestra el mensaje secreto descifrado del emisor al receptor. La tercera opción muestra los pasos de las dos opciones anteriores en una sola.

También se elaboraron carteles científicos para dar una explicación de la demostración experimental el cual se compone de dos partes, observe la Figura 4.16:

1. Explicación breve de los principios teóricos del protocolo BB84, conceptos físicos, probabilísticos e informáticos presentes en el experimento de criptografía cuántica.

2. Explicación de las componentes para Alice y Bob, así como su funcionamiento para el sistema experimental de criptografía cuántica.

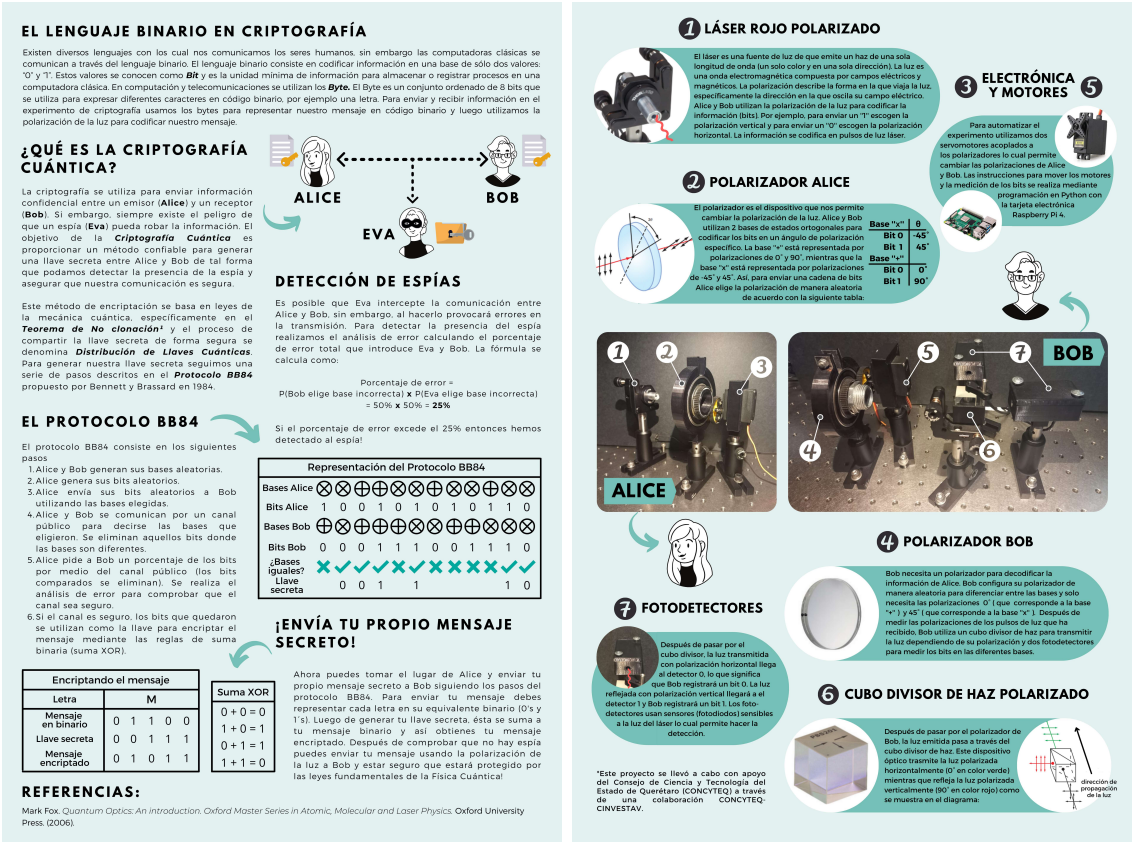


Figura 4.15: Carteles científicos para acompañar la demostración experimental. Izquierda: cartel con la explicación de los conceptos físicos y el protocolo BB84. Derecha: cartel con la descripción de los componentes del sistema experimental (Elaboración propia)

Para montar el experimento en el museo se asignó un espacio con una mesa para fijar las componentes ópticas y electrónicas. Con el espacio disponible se determinó la disposición de las componentes del experimento y las pantallas como se visualiza en la Figura 4.16.

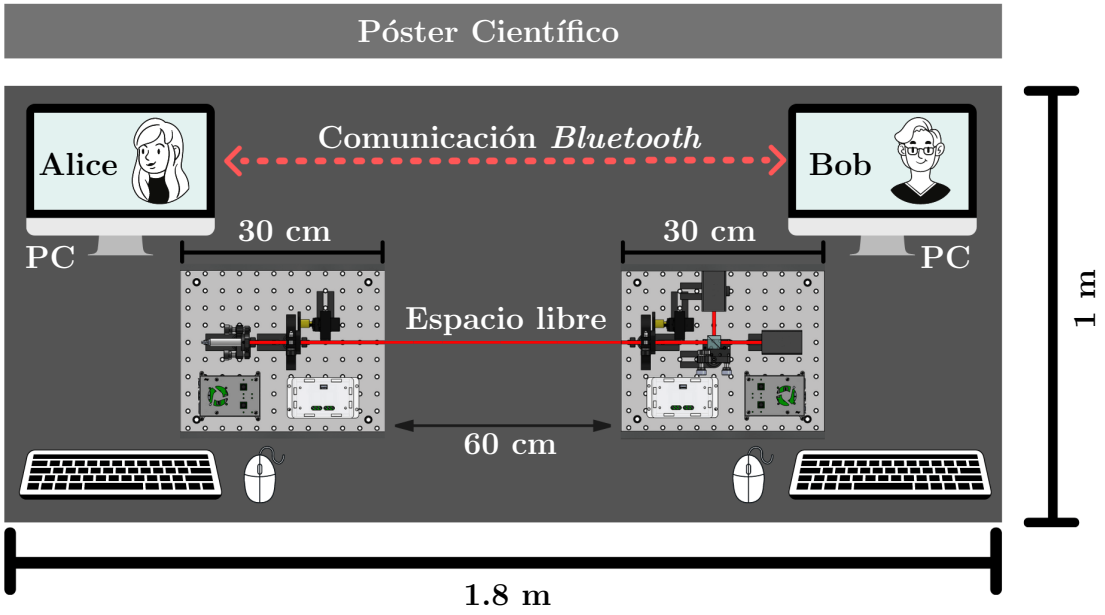
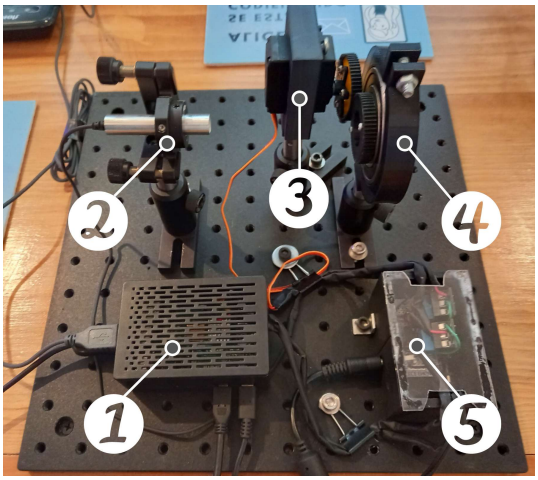
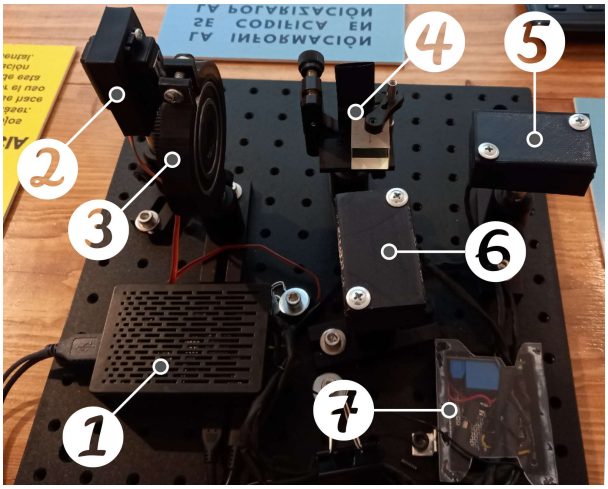


Figura 4.16: Disposición de las componentes del experimento de criptografía cuántica sobre la mesa. El acomodo de las componentes está considerado para ser controlado por uno o dos usuarios (Elaboración propia).

A partir del esquema anterior se colocaron las componentes del prototipo experimental en la mesa óptica (ver Fig. 4.17) y en el espacio asignado dentro del museo (ver Fig. 4.18).

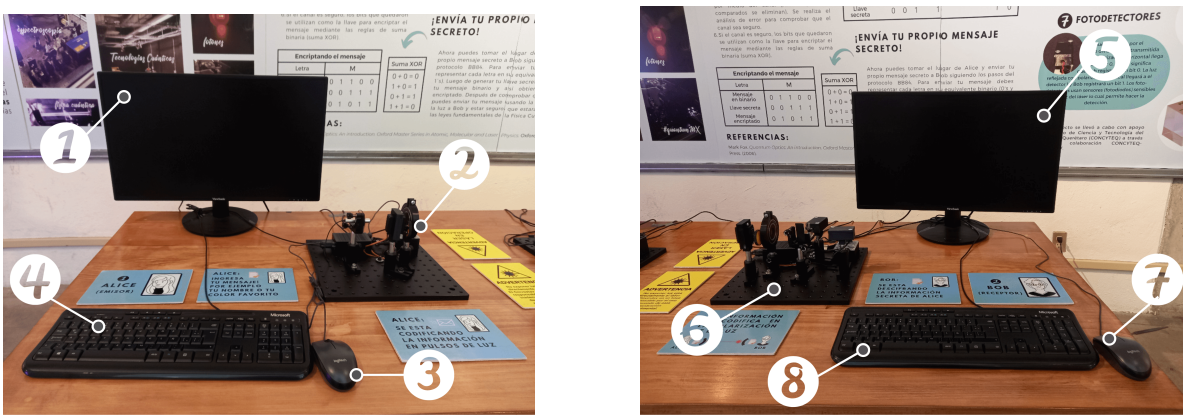


(a) **Alice.** (1) Raspberry Pi (2) Láser, (3) Servomotor, (4) Retardador de onda, (5) Circuito electrónico.



(b) **Bob.** (1) Raspberry Pi (2) Servomotor, (3) Retardador de onda, (4) PBS, (5) Detector "0", (6) Detector "1", (7) Circuito electrónico.

Figura 4.17: Componentes del prototipo experimental para Alice y Bob.



(a) **Alice.** (1) Monitor (2) Breadboard con componentes, (3) Mouse, (4) Teclado

(b) **Bob.** (5) Monitor, (6) Breadboard con componentes, (7) Mouse, (8) Teclado

Figura 4.18: Experimento montado en el museo para Alice y Bob.

En la Figura 4.19 se observa el experimento final de *QKD* junto con todas las componentes para su explicación y funcionamiento dentro en el museo.

Figura 4.19: Prototipo experimental automatizado de criptografía cuántica final con todas las componentes ópticas y electrónicas, el sistema de protección y carteles científicos para su explicación.

Como parte de la implementación del experimento en el museo también se brindaron capacitaciones presenciales tanto al personal encargado del museo como a los estudiantes del servicio social para explicar y resolver dudas sobre el funcionamiento del experimento. Este proyecto se presentó en la ceremonia de reapertura del Museo de Ciencia y Tecnología, Péndulo de Foucault donde participaron diversas instituciones educativas como escuelas de nivel primaria y secundaria y público en general. En esta exposición se realizaron demostraciones experimentales del prototipo y explicación de los conceptos implicados en Criptografía Cuántica.

Figura 4.20: Demostración del experimento a estudiantes de primaria en el Museo de Ciencia y Tecnología del Péndulo de Foucault.

Capítulo 5
Conclusiones

En este trabajo se logró construir un prototipo automatizado que simula los principios de un experimento de Criptografía Cuántica basado en el Protocolo BB84. Es importante mencionar que este prototipo de criptografía cuántica es un experimento clásico en el sentido de que no se utilizó una fuente de fotones únicos, sino que se utilizaron pulsos cortos de luz polarizada. El prototipo automatizado puede ser modificado para volverse un sistema cuántico utilizando atenuadores que permitan generar pulsos al nivel de un sólo fotón y sustituyendo los módulos *LDR* por detectores de un sólo fotón. Sin embargo, a pesar de que ya existen fuentes de fotones únicos deterministas, tal como las fuentes basadas en la conversión descendente paramétrica (*parametric down-conversion*) o más recientemente en puntos cuánticos de semiconductores (Senellart, Solomon, y White, 2017), estas siguen siendo muy complicadas de implementar y requieren de un presupuesto elevado para su desarrollo.

La automatización del prototipo experimental permitió mejorar el tiempo empleado en generar una clave entre dos usuarios en un 84 % en comparación con el kit comercial, el cual se basa en los mismos principios que nuestro prototipo automatizado. A pesar de que el costo de nuestro experimento es ligeramente superior comparado con el de *Thorlabs*, se logró una mejora significativa en el tiempo de cada prueba experimental. Por otra parte, nuestro prototipo experimental no se puede comparar con las altas velocidades de transferencia de *bytes* como lo hacen los sistemas automatizados comerciales actuales de *QKD*. Por ejemplo, *Cerberis XG QKD System* de la empresa *ID Quantique* es 99.98 % más rápido en la transferencia de *bits*, pero con un costo

proporcionalmente elevado. Teniendo un mayor presupuesto se podrían adquirir celdas tipo Pockels o moduladores acusto-ópticos para incrementar la velocidad de nuestro sistema. La propuesta descrita en este trabajo es mucho más accesible y puede servir de guía para que futuras instituciones o centros de investigación puedan construir su propio prototipo experimental automatizado de criptografía cuántica.

El desarrollo y construcción de este experimento se realizó utilizando componentes similares del kit comercial, por lo que es posible reducir los costos modificando algunas componentes. Por ejemplo, las bases y postes utilizadas en nuestro prototipo experimental son de la empresa *Thorlabs*, pero es posible sustituirlas por piezas impresas en 3D. La óptica seleccionada tiene las mismas medidas que las del kit, pero existen componentes ópticas de menor tamaño que pueden ser adquiridos a un menor costo sin cambiar las propiedades del sistema. Los procesadores Raspberry Pi 4 también pueden ser sustituidos por tarjetas de control más económicas.

La desventaja del prototipo experimental es la distancia de alcance para la trasferencia de datos, ya que no se puede superar distancias mayores a 100 m debido al límite de alcance de la comunicación *Bluetooth* utilizada para la verificación del canal. Es posible incrementar la distancia hasta la del alcance del láser si se cambia esta comunicación por otra, como la conexión de red. Otra opción es utilizar fibra óptica la cual permite incrementar aún más la distancia con la misma fuente, aunque es necesario realizar correcciones en la polarización. Para los propósitos de nuestro sistema experimental, la distancia máxima alcanzada con la comunicación *Bluetooth* es suficiente para el lugar asignado dentro del museo del Péndulo de Foucault.

En el área de la investigación y desarrollo, el prototipo experimental propuesto podrá servir como modelo para que otras personas interesadas en desarrollar criptografía cuántica experimental puedan construir su propio experimento automatizado y comprender los principios de la criptografía cuántica. También podría servir para desarrollar un experimento que pueda implementar mejoras y obtener nuevo conocimiento en el área de criptografía cuántica. Actualmente, el prototipo experimental de criptografía cuántica permite que diversas personas tengan un acercamiento a las ciencias, específicamente en el área de las tecnologías cuánticas y conozcan los desarrollos en

estas nuevas tecnologías comprendiendo sus fundamentos de forma educativa e interactiva. Tan sólo se estima que de Febrero a Julio de 2022 se ha contado con un número estimado de 1139 adultos y 1300 niños de diversos niveles educativos visitando e interactuando con la demostración experimental "Criptografía Cuántica: ¿Qué es y cómo funciona?".

A partir de este trabajo se podrían originar otros temas de investigación como implementar mejoras al sistema o agregar otras características. Por ejemplo, se podría agregar atenuadores para generar los fotones individuales y obtener un sistema cuántico. También podría sustituirse el diodo láser por una fuente de fotones individuales e implementar un código que permita realizar la autenticación de datos. Inclusive, debido a que el canal de comunicación es por el espacio libre se podría implementar una comunicación por satélite entre los interlocutores permitiendo un mayor alcance, con una mayor rapidez y una comunicación segura.

Referencias

Aczel, A. D. (2002). *Entanglement: the greatest mystery in physics*. Raincoast Books.

Ali, R., Kim, S. W., Kim, B.-S., y Park, Y. (2018). Design of MAC layer resource allocation schemes for IEEE 802.11 ax: Future directions. *IETE Technical Review*, *35*(1), 28–52.

Bennett, C. H. (1992). Quantum cryptography using any two nonorthogonal states. *Physical review letters*, *68*(21), 3121.

Bennett, C. H., Bessette, F., Brassard, G., Salvail, L., y Smolin, J. (1991). Experimental quantum cryptography. En *Advances in Cryptology—EUROCRYPT'90: Workshop on the Theory and Application of Cryptographic Techniques Aarhus, Denmark, May 21–24, 1990 Proceedings 9* (pp. 253–265).

Bennett, C. H., Bessette, F., Brassard, G., Salvail, L., y Smolin, J. (1992, jan). Experimental quantum cryptography. *Journal of Cryptology*, *5*(1), 3–28. Descargado de http://link.springer.com/10.1007/BF00191318 doi: 10.1007/BF00191318

Bennett, C. H., y Brassard, G. (1984, dec). Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, *560*(P1), 7–11. doi: 10.1016/j.tcs.2014.05.025

Bernhardt, C. (2019). *Quantum computing for everyone*. Mit Press.

Borrayo, V., y Arias, D. (2017). Sistema servidor-cliente seguro para la verificación de autenticidad de archivos por medio de algoritmos digest hash reestadarizados. En *2017 ieee central america and panama student conference (conescapan)* (pp. 1–5).

Boscá Díaz-Pintado, M. (2017). Computación, teleportación y criptografía cuántica. la segunda revolución cuántica. España: RBA Contenidos Editoriales y Audiovisuales, S.A.U.

Boylestad, R. L., Nashelsky, L., Barraza, C. M., y Fernández, A. S. (2003). *Electrónica: teoría de circuitos y dispositivos electrónicos* (Vol. 8). PEARSON educación.

Bransden, B., y Joachain, C. (2000). *Quantum Mechanics*. Pearson Education),(Original work published 1989).

Brukner, Č., Żukowski, M., y Zeilinger, A. (2002). Quantum communication complexity

protocol with two entangled qutrits. *Physical Review Letters*, *89*(19), 197901.

Cao, Y., Liang, H., Yin, J., Yong, H.-L., Zhou, F., Wu, Y.-P., ... others (2013). Entanglement-based quantum key distribution with biased basis choice via free space. *Optics express*, *21*(22), 27260–27268.

Datasheet raspberry pi 4 model b (Manual de software informático n.º Release 1). (2019, 6).

Davies, D. (1997). A brief history of cryptography. *Information Security Technical Report*, *2*(2), 14–17.

Diffie, W., y Hellman, M. E. (2019). New directions in cryptography. En *Secure communications and asymmetric cryptosystems* (pp. 143–180). Routledge.

DiVincenzo, D. P. (2000). The Physical Implementation of Quantum Computation. *Fortschritte der Physik*, *48*(9-11), 771–783. doi: 10.1002/1521-3978(200009)48:9/11⟨771::aid-prop771⟩3.0.co;2-e

Djordjevic, I. B. (2019). *Physical-Layer Security and Quantum Key Distribution* (1st ed.). Springer International Publishing.

Dür, W., y Heusler, S. (2013). What we can learn about quantum physics from a single qubit. *arXiv preprint arXiv:1312.1463*.

Einstein, A., Podolsky, B., y Rosen, N. (1935, May). Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? *Phys. Rev.*, *47*, 777–780. doi: 10.1103/PhysRev.47.777

Ekert, A. K. (1992). Quantum Cryptography and Bell's Theorem. , 413–418. doi: 10.1007/978-1-4615-3386-3_34

Fletcher, W. I. (1997). *An engineering approach to digital design*. Prentice Hall PTR.

Fox, M. (2006). Quantum optics an introduction. En (1.ª ed., cap. 4). New York: Oxford University Press Inc.

Gisin, N., Ribordy, G., Tittel, W., y Zbinden, H. (2002, Mar). Quantum cryptography. *Rev. Mod. Phys.*, *74*, 145–195. doi: 10.1103/RevModPhys.74.145

Griffiths, D. J. (2004). *Introduction to Quantum Mechanics* (2nd ed.). Pearson Prentice Hall.

Hecht, E. (2002). *Optics* (5.ª ed.). Pearson Education India.

Horowitz, P., Hill, W., y Robinson, I. (1989). *The art of electronics* (Vol. 2). Cambridge

university press Cambridge.

ID Quantique. (2020). *Understanding Quantum Cryptography* (Inf. Téc.). Descargado de https://www.idquantique.com/quantum-safe-security/products/cerberis-xg-qkd-system/

ID Quantique. (2022, 05). *Cerberis XG QKD System* (Inf. Téc.). Descargado de https://www.idquantique.com/quantum-safe-security/products/cerberis-xg-qkd-system/

Kockum, A. F., y Nori, F. (2019). Quantum bits with josephson junctions. *Fundamentals and Frontiers of the Josephson Effect*, 703–741.

Korpela, J. K. (2006). *Unicode explained*. .º'Reilly Media, Inc.

.

Llamas, L. (2018). Medir nivel de luz con Arduino y fotoresistencia LDR (GL55). *Arduino*. Descargado de https://www.luisllamas.es/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/

Ma, L., Mink, A., y Tang, X. (2009). High speed quantum key distribution over optical fiber network system. *Journal of research of the National Institute of Standards and Technology*, *114*(3), 149.

MagiQ Technologies. (2004). *Ultimate Cryptography Solution for Network Security* (Inf. Téc.). Descargado de https://www.magiqtech.com/solutions/network-security/

Makhlin, Y., Schön, G., y Shnirman, A. (2001). Quantum-state engineering with Josephson-junction devices. *Reviews of modern physics*, *73*(2), 357.

McManus, S., y Cook, M. (2021). *Raspberry Pi for dummies*. John Wiley & Sons.

Milburn, G. (2012). *Quantum Optics*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-642-19409-2_18

Moya Fernández, F. (2017). Desarrollo en Python. *Taller de Raspberry Pi*. Descargado de https://franciscomoya.gitbooks.io/taller-de-raspberry-pi/content/es/

Nielsen, M., y Chuang, I. (2010). Quantum Computation and Quantum Information. En (cap. 12). United Kingdom: Cambridge University Press.

Olvera, D. (2016). Plataforma didáctica para adquisición de datos a través de un

microcontrolador. *Tesis de licenciatura: Instituto Politécnico Nacional*.

Pedrotti, F. L., Pedrotti, L. M., y Pedrotti, L. S. (2017). *Introduction to Optics*. Cambridge University Press. doi: 10.1017/9781108552493

Puentes, J. (2021). Spanlp. *The Python Package Index (PyPI)*. Descargado de https://pypi.org/project/spanlp/

Quintessence Labs. (2022). *QuintessenceLabs Sits at the Intersection of Quantum Cyber* (Inf. Téc.). Descargado de https://www.quintessencelabs.com/products#qkd

Raspberry Pi. (2021). Raspberry Pi hardware. *Raspberry Pi Ltd*. Descargado de https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/README.md

Rivest, R. L., Shamir, A., y Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, *21*(2), 120–126. Descargado de https://doi.org/10.1145/359340.359342 doi: 10.1145/359340.359342

Rueppel, R. A. (1986). Stream ciphers. *Analysis and Design of Stream Ciphers*, 5–16.

Salam, A. A., y Al Salah, R. M. (2015). Vertically Scrambled Caesar Cipher Method. *International Journal of Computer Applications*, *118*(21).

Senellart, P., Solomon, G., y White, A. (2017). High-performance semiconductor quantum-dot single-photon sources. *Nature Nanotechnology*, *12*(11), 1026–1039. doi: 10.1038/nnano.2017.218

Shannon, C. E. (1949). Communication theory of secrecy systems. *The Bell system technical journal*, *28*(4), 656–715. doi: 10.1002/j.1538-7305.1949.tb00928.x

Shor, P. (1994). Algorithms for quantum computation: discrete logarithms and factoring. En *Proceedings 35th Annual Symposium on Foundations of Computer Science* (p. 124-134). doi: 10.1109/SFCS.1994.365700

Simmons, G. J. (1979). Symmetric and asymmetric encryption. *ACM Computing Surveys (CSUR)*, *11*(4), 305–330.

Singh, S. (2001). *The code book: How to make it, break it, hack it, crack it, for young people* (1st Edition ed.). Delacorte.

Smart, N. P., y Smart, N. P. (2016). *Cryptography made simple*. Springer.

Sutor, R. S. (2019). *Dancing with qubits: How quantum computing works and how it can change the world* (1.ª ed.). Packt Publishing.

Thorlabs. (2017, 05). *EDU-QCRY1 EDU-QCRY1/M Quantum Cryptography Demonstration Kit. Manual* (Inf. Téc. n.º MTN005660-D02). Descargado de https://www.thorlabs.com/thorproduct.cfm?partnumber=EDU-QCRY1

Toshiba. (2004). *Quantum Key Distribution* (Inf. Téc.). Descargado de https://www.global.toshiba/ww/products-solutions/security-ict/qkd.html

Vainio, J. T., y cols. (2000). Bluetooth security. En *Proceedings of helsinki university of technology, telecommunications software and multimedia laboratory, seminar on internetworking: Ad hoc networking, spring* (Vol. 5).

Vernam, G. S. (1926). Cipher printing telegraph systems: For secret wire and radio telegraphic communications. *Journal of the AIEE*, *45*(2), 109–115.

Vuleka, B. (2021). How much data is created every day? *SeedScientific*. Descargado de https://seedscientific.com/how-much-data-is-created-every-day/

Wiesner, S. (1983). Conjugate coding. *ACM Sigact News*, *15*(1), 78–88.

Wolf, R. (2021). *Quantum key distribution: An introduction with exercises (lecture notes in physics)* (1st ed. ed.). Springer. doi: 10.1007/978-3-030-73991-1

Wootters, W. K., y Zurek, W. H. (1982). A single quantum cannot be cloned. *Nature*, *299*, 802–803.

Apéndice A

Diagramas para las conexiones de Alice

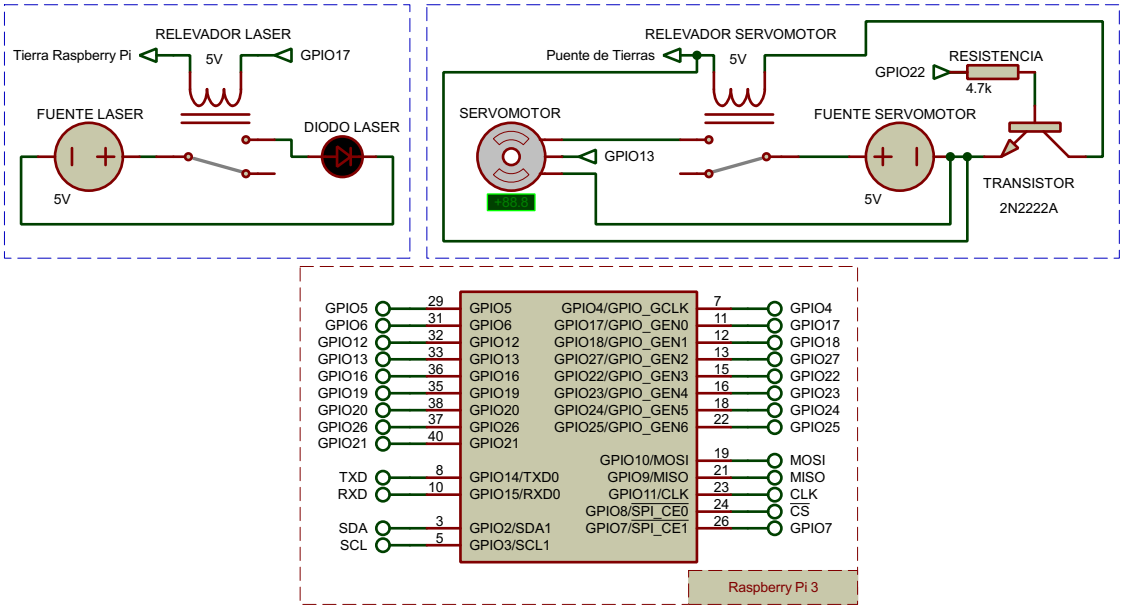


Figura A.1: Diagrama de componentes y conexiones para Alice. En el lado izquierdo se observa el circuito del láser, a la derecha el circuito del servomotor y en medio la tarjeta de programación Raspberry Pi. Para obtener mayores velocidades se puede conectar el servomotor a una fuente de 6V. Sin embargo, para nuestro propósito fue suficiente la fuente de 5V. Conectamos un transistor y resistencia entre el pin GPIO22 y el servomotor. De no realizar estas últimas conexiones el servomotor produce pequeños movimientos que hacen reducir su vida útil (Elaboración propia).

Apéndice B

Diagramas para las conexiones de Bob

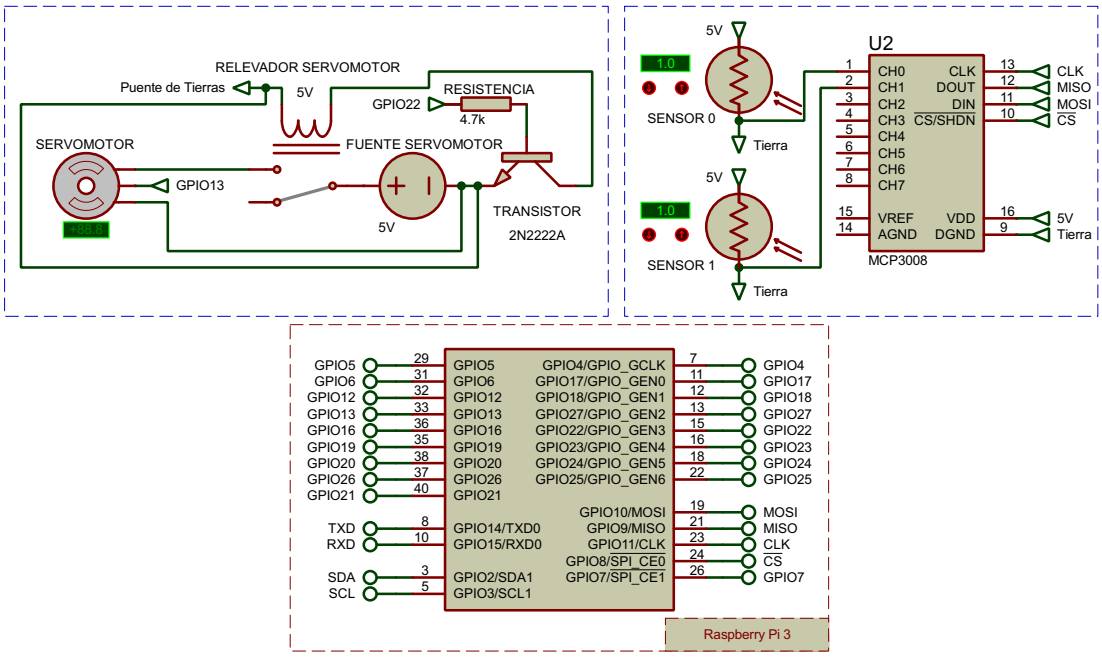


Figura B.1: Diagrama de componentes y conexiones para Bob. En el lado izquierdo se observa el circuito del servomotor, a la derecha el circuito de los detectores y en medio la tarjeta de programación Raspberry Pi. El circuito del servomotor tiene los mismos componentes que Alice. Los detectores *LDR* muestran las conexiones al MCP3008 que recibe las señales analógicas de los módulos LDR y las convierte a señales digitales para después ser leídas por la Raspberry Pi (Elaboración propia).

Apéndice C

Código para realizar los pulsos del láser

```
#Este programa realiza los pulsos de luz láser cada determinado
    intervalo de tiempo
#Librerías-----------------------------------------------------------
import RPi.GPIO as GPIO
from time import sleep

#Variables-----------------------------------------------------------

GPIO.setmode(GPIO.BOARD)    #Permite nombrar los puertos
GPIO.setup(17, GPIO.OUT)    #Establece el puerto 17 como salida
Key_length=100              #Tamaño de la llave a enviar

#Main----------------------------------------------------------------
i=0
while i<Key_length:
    GPIO.output(7,True)     #Permite encender el láser
    sleep(1)                #Tiempo de encendido del láser
    GPIO.output(7,False)    #Permite apagar el láser
    sleep(1)                #Tiempo de apagado del láser
    i+=1
#sleep() recibe como entrada un valor en segundos y es necesario
    establecer los dos sleeps para poder encender y apagar. Si sólo se
    pone uno, se quedará encendido o apagado y no generan los pulsos.
```

Apéndice D

Código para el movimiento del servomotor

```
#Este programa permite mover el motor de acuerdo a los ángulos de
   polarización
#Librerías-----------------------------------------------------------
import RPi.GPIO as GPIO
from time import sleep
import random
import math

#Variables-----------------------------------------------------------
GPIO.setmode(GPIO.BOARD)    #Permite nombrar los puertos
GPIO.setup(13, GPIO.OUT)    #Servomotor: puerto 13 como salida
GPIO.setup(22, GPIO.OUT)    #Relevador: puerto 22 como salida
servo=GPIO.PWM(13, 50)      #Frecuencia del servomotor: 50Hz
servo.start(2.0)            #Iniciar el servomotor en ángulo 0
Key_length=100              #Tamaño de la llave
Bases=[]                    #Vector de Bases aleatorias
Bits=[]                     #Vector de Bits aleatorias
Angles=[]                   #Vector de ángulos de polarización

#Main----------------------------------------------------------------
for i in range(Key_length): #Bases aleatorias (0:base +; 1:base x)
    Bases.append(random.randint(0,1))

```

```
for i in range(Key_length): #Bits aleatorias (0:bit 0; 1:bit 1)
    Bits.append(random.randint(0,1))

#Dependiendo de la base elegida y el bit seleccionado, se tendrá el á
   ngulo de polarización:
#Base 0 || Bit 0 || Ángulo 0
#Base 0 || Bit 1 || Ángulo 90
#Base 1 || Bit 1 || Ángulo 45
#Base 1 || Bit 0 || Ángulo 135
for i in range(Key_length):
    if Bases[i]==0:
        if Bits[i]==0:
            Angles.append(0)
        elif Bits[i]==1:
            Angles.append(90)
    elif Bases[i]==1:
        if Bits[i]==0:
            Angles.append(135)
        elif Bits[i]==1:
            Angles.append(45)

i=0
GPIO.output(22,True)    #Activar relevador para función del servo
while i<Key_length:

    Grade=((1.0/18.0)*Angles[i])+2.0    #Formula
    servo.ChangeDutyCycle(Grade)        #Ciclo de trabajo

    #El tiempo de espera para mover al siguiente ángulo depende del
   servomotor. Si se establece una velocidad menor no llegará a la
   posición adecuada.
    sleep(0.3)
    i+=1

GPIO.output(22,False)   #Desactivar relevador
```

Apéndice E

Código para el movimiento del servomotor de Bob

La primer modificación que debe de realizarse es en el apartado de las variables. Bob crea una variable para guardar las mediciones de los pulsos de luz que llegan a los detectores y se descartan las líneas 16, 23 y 24 del código del apéndice D. La segunda modificación es dentro del *Main* para la selección de las bases aleatorias, Bob sólo se necesitan dos bases. Por lo que se eliminaron las lineas 26-41 del código del apéndice D y se sustituyeron por el siguiente código:

```
1    #Dependiendo de la base seleccionada se tiene un ángulo de
     polarización:
2    #Base 0 || Ángulo 0
3    #Base 1 || Ángulo 45
4    for i in range(Key_length):
5        if Bases[i]==0:
6            Angles.append(0)
7        elif Bases[i]==1:
8            Angles.append(45)
```

Apéndice F

Código para realizar las mediciones de los detectores

```
#Programa para realizar las mediciones de la luz polarizada de los
   detectores
#Librerías-----------------------------------------------------------
from gpiozero import MCP3008    #Para leer datos del ADC (MCP3008)
from time import sleep
#Variables-----------------------------------------------------------
Key_length=100          #Tamaño de la llave a recibir
Bits = []               #Vector de las mediciones de los LDR
Sensor0 = MCP3008(channel=0)    #Detector 0. Conectado al canal 0
Sensor1 = MCP3008(channel=1)    #Detector 0. Conectado al canal 0
#Main----------------------------------------------------------------
i=0
while i<Key_length:
    Medicion_0 = 1-Sensor0.value    #Funcionamiento inverso corregido
    Medicion_1 = 1-Sensor1.value    #Funcionamiento inverso corregido

    if Medicion_Sensor0<Medicion_1:   #Si detector 0>1: Guarda un 0
        Bits.append(0)
    if Medicion_Sensor0<Medicion_1: #Si detector 1>0: Guarda un 0
        Bits.append(1)

    sleep(1)
    i+=1
```

Apéndice G

Código para la comunicación *Bluetooth*

Lo primero es importar las librerías correspondientes. En Python, para establecer la comunicación *Bluetooth* sólo hace falta agregar la siguiente línea de código:

```
#Librerías-----------------------------------------------------------
import bluetooth
```

Después, se creó un pequeño programa que permite enviar los mensajes utilizando la comunicación *Bluetooth*. Para esto se considera que el tipo de elementos que se envían son vectores. Para enviar vectores, se realizó una función de Python la cual recibe 2 entradas: la cantidad de elementos a enviar (x) y el vector a enviar (y). Después de establecer la comunicación con la dirección *MAC* del otro dispositivo, se envía por medio de un bucle *for* cada elemento del vector y al finalizar se envía la palabra clave "*exit*" que permite terminar la comunicación *Bluetooth*.

```
#Código para enviar mensajes----------------------------------------
def Send_Message(x,y):
    sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
    sock.connect(('XX:XX:XX:XX:XX:XX', 1))
    while True:
        for i in range(x):
            sock.send(str(y[i]))
        sock.send('exit')
        break
```

También se necesita una función que permita permita recibir el mensaje. Para esta función sólo hace falta una entrada, la cual es el vector donde se guardan los valores. Cada elemento recibido es evaluado por si se recibe la palabra clave "*exit*". Una vez recibida la palabra clave se termina la conexión *Bluetooth* y en caso contrario, los elementos se agregan a un vector.

```
#Código para recibir mensajes----------------------------------------
def Receive_Messages(x):
    server_sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
    server_sock.bind(("",1))
    server_sock.listen(1)
    client_sock,address=server_sock.accept()
    while True:
        data = client_sock.recv(1024)
        a=data.decode("utf-8")
        if a=='exit':
            break
        else:
            x.append(int(a))
```

Apéndice H

Código para realizar la sincronización de los interlocutores

Lo primero a tener en cuenta es que la sincronización se realiza en el canal de comunicación a través del cual se envían los pulsos de luz polarizada, pero antes se agregan las librerías para el control de las componentes electrónicas y para realizar la sincronización con la comunicación *Bluetooth*.

Alice:

```
import RPi.GPIO as GPIO
from time import sleep
import bluetooth
import math
```

Bob:

```
from gpiozero import MCP3008
import RPi.GPIO as GPIO
from time import sleep
import bluetooth
import math
```

Después se realiza una primer sincronización mediante la comunicación *Bluetooth*. Aquí se envía cualquier información (en este caso el caracter 1) para indicar cuando comienza el envío de los pulsos de luz.

Alice:

```
#Variables-------------------------------------------------------------
GPIO.setmode(GPIO.BOARD)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(13, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)
servo_A=GPIO.PWM(13, 50)
servo_A.start(2.0)
Key_length=100
Bases_Alice=[]
Bits_Alice=[]
Angles_Alice=[]
Exit = [1]  #Información para enviar los pulsos de luz polarizados
#Main------------------------------------------------------------------
Send_Message(1,Exit)
Send_key()
```

Bob:

```
#Variables-------------------------------------------------------------
GPIO.setmode(GPIO.BOARD)
Sensor0 = MCP3008(channel=1)
Sensor1 = MCP3008(channel=0)
GPIO.setup(13, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)
servo_B=GPIO.PWM(13, 50)
servo_B.start(2.0)
Key_length=100
Bases_Bob=[]
Bits_Bob=[]
Angles_Bob=[]
Exit = []   #Información para enviar los pulsos de luz polarizados
#Main------------------------------------------------------------------
Receive_Messages(Exit)
Send_key()
```

Como se observa en el código anterior, primero Alice le envía un valor cualquiera a Bob para indicarle que los pulsos de luz serán enviados. Después de esto ambos entran

a la función llamada "*Send_key()*", la cual reinicia el tiempo de la Raspberry para sincronizar ambas Raspberry y empezar a contar los segundos iguales en el proceso de ejecución del experimento.

```
os.system("sudo date --set 00:00:00")
```

Luego el programa entra dentro de un bucle *while* que envía todos los *bits* de la clave. El bucle *while* contiene la secuencia de paro "$i <= Bits$" debido a que primero se deben mover los retardadores de onda $\lambda/2$ y después enviar y detectar los pulsos de luz. Entonces, dentro del bucle se indica a los servomotores los ángulos para orientar los retardadores de onda $\lambda/2$ y también la parte para enviar y medir los pulsos de luz.

Alice:

```
#Funcion para el envío de la llave-----------------------------------
def Send\_key():
    os.system("sudo date --set 00:00:00")   #Restablecer hora
    GPIO.output(13, GPIO.HIGH)              #Activar relevador
    i=0
    while i<=Bits:
        if i!=0 and i%9==0: #Proceso de calibración cada 9 bits
            window.read(timeout=0)
            while True:
                s=datetime.now().second
                if s%4==0:
                    break

        if i==0:    #Orientar el retardador de Alice
            servo_A.ChangeDutyCycle(((1.0/18.0)*Angles_Alice[i])+2.0)
            sleep(0.23)
            i+=1
            continue

        elif i<Bits:    #Encender láser/mover retardador
            GPIO.output(11,True)
            sleep(0.3)
            GPIO.output(11,False)
            servo_A.ChangeDutyCycle(((1.0/18.0)*Angles_Alice[i])+2.0)
            sleep(0.23)
```

```
        elif i==Bits:   #Encender láser
            GPIO.output(11,True)
            sleep(0.3)
            GPIO.output(11,False)

        i+=1
    GPIO.output(15,GPIO.LOW)    #Desactivar relevador
```

Bob:

```
#Funcion para el envío de la llave-----------------------------------
def Send\_key():
    os.system("sudo date --set 00:00:00")   #Restablecer hora
    GPIO.output(13, GPIO.HIGH)              #Activar relevador
    i=0
    while i<=Bits:
        if i!=0 and i%9==0: #Proceso de calibración cada 9 bits
            window.read(timeout=0)
            while True:
                s=datetime.now().second
                if s%4==0:
                    break

        if i==0:    #Orientar el retardador de Bob
            servo_Bob.ChangeDutyCycle(((1.0/18.0)*Angles_Bob[i])+2.0)
            sleep(0.23)
            i+=1
            continue

        elif i<Bits:    #Realizar mediciones/mover retardador
            sleep(0.1)
            l,m= Sensor0.value,Sensor1.value
            if l>m:
                Bob_Bits.append(1)
            elif m>l:
                Bob_Bits.append(0)
            elif l==m:
                aleatorio=random.randint(0,1)
                Bob_Bits.append(aleatorio)
```

```
30            servo_Bob.ChangeDutyCycle(((1.0/18.0)*Angles_Bob[i])+2.0)
31            sleep(0.23)
32            sleep(0.2)
33
34        elif i==Bits:   #Realizar mediciones
35            sleep(0.1)
36            l,m= Sensor0.value,Sensor1.value
37            if l>m:
38                Bob_Bits.append(1)
39            elif m>l:
40                Bob_Bits.append(0)
41            elif l==m:
42                aleatorio=random.randint(0,1)
43                Bob_Bits.append(aleatorio)
44            sleep(0.2)
45
46        i+=1
47    GPIO.output(13,GPIO.LOW)     #Desactivar relevador
```

La función *Send_key()* realiza la sincronización tomando en cuenta todas las variables necesarias para la automatización del experimento. De los códigos anteriores, se crea el vector "Angles" el cual necesita las bases y los bits para la parte de Alice y de las bases para la parte de Bob. Puede consultarse el código D para la creación de dicho vector.

Apéndice I

Conversión de caracteres a binario y suma XOR para encriptar y desencriptar mensajes

Para cifrar el mensaje y posteriormente enviarlo, Alice primero tiene que convertir la información que quiere enviar de caracteres a binario. Para esto utiliza el siguiente código:

```
def Unicode_to_Binary():
    for i in range(len(Message)) :
        a = ord(Message[i])      #Número decimal a Unicode.
        for j in range(8):       #Conversión de decimal a binario
            b = a % 2
            Alice_Bits.insert(j,b)
            a = a//2

    Alice_Bits.reverse()         #Funcionamiento inverso corregido
```

El mensaje en binario se almacenan en un vector al igual que la clave para luego realizar la suma XOR. Para este paso, ya se realizaron los pasos del protocolo BB84 que permiten la creación de la clave. Además, la clave debe de ser del mismo tamaño que el mensaje, por lo que se tiene que ajustar su longitud. Una vez que se tienen los caracteres en binario, se le suma la clave mediante la suma XOR. La suma XOR se implementa como:

```
def XOR(Mensaje en binario):
    for i in range(len(Mensaje en binario)):
        #Condiciones de la suma XOR, ver Tabla 2.1
        if Clave[i]==Mensaje en binario[i]:
            Mensaje_encriptado.append(0)
        else:
            Mensaje_encriptado.append(1)
```

Finalmente, se envía el mensaje encriptado a Bob. Para que Bob descifre el mensaje tiene que sumar la clave utilizando el mismo código anterior y después convertir de binario a Unicode. El código para transformar de binario a caracteres es:

```
Message_received = []   #Guardado del mensaje
def Binary_to_Unicode():    #Binario a caracteres
    cantidad=int(Bits/8)    #Generación de grupos de 8 bits
    palabra=''  #Agregar el mensaje desencriptado a un string

    #Bucle que permite pasar cada grupo de 8 bits a su caracter
    for i in range(cantidad):
        #Cada bit es multiplicado por la constante correspondiente
   para obtener el número decimal del carácter
        a = Mensaje_sin_clave[8*i+7]*1
        b = Mensaje_sin_clave[8*i+6]*2
        c = Mensaje_sin_clave[8*i+5]*4
        d = Mensaje_sin_clave[8*i+4]*8
        e = Mensaje_sin_clave[8*i+3]*16
        f = Mensaje_sin_clave[8*i+2]*32
        g = Mensaje_sin_clave[8*i+1]*64
        h = Mensaje_sin_clave[8*i+0]*128
        #Suma de variables para obtener el número decimal
        suma=a+b+c+d+e+f+g+h
        #La función chr() pasa de decimal a caracteres y es agregado
   al mensaje desencriptado
        palabra = palabra + chr(suma)
    Message_received.append(palabra)
```

Apéndice J

Código para guardar e importar datos

Debido a la configuración del experimento, es posible elegir la opción donde sólo se verifica el canal y genera la clave sin utilizarla. Debido a esto, es necesario un código para guardar la clave y después importarla cuando se requiera. El código para guardar e importar la clave es el siguiente:

```
#Función para guardar la clave en un archivo .txt
import os                          #Librería
os.chdir('.../.../')               #Directorio

#Recibe 3 entradas: nombre del archivo (x), mensaje dentro del
    documento para descripción (y) y el vector que se desea guardar (z)
    .
def Save_files(x,y,z):
    Document_name = x +".txt"      #Nombre del documento
    f=open(Document_name,"w")      #Se abre documento para escribir
    f.write(y)                     #Se agrega la descripción

    for i in z:
        f.write(str(i))            #Escribiendo
        f.write("\n")              #Se agrega un salto de linea

    f.close()                      #Se cierra el documento
```

Cuando se requiera enviar un mensaje entonces se importa la clave y las bases seleccionadas para orientar los retardadores de onda y posteriormente cifrar y descifrar el mensaje. El código que permite realizar la importación de los datos es:

```
#Código que permite importar los datos desde un archivo .txt
import numpy as np                          #Librería
data = np.loadtxt('/.../.txt',skiprows=1) #Cargar de archivos .txt

#Se equiere dos valores: los datos a importar (x) y el lugar de
   guardado (y)
def Import_files(x,y):
    for i in range(len(x)):
        y.append(int(x[i]))
```

Cada que se crea una nueva llave se actualiza el contenido que hay dentro del documento. Lo que significa que borra los datos anteriores en el documento. Una forma sencilla de realizarlo es borrando todos los archivos con extensión ".txt". El código se muestra a continuación

```
#Comando para abrir la carpeta del lugar donde se quieren eliminar los
    archivos
files = glob.glob('/Dirección de la carpeta de donde se eliminaran los
    archivos/*.txt')
    #Se establece un bucle para encontrar los archivos .txt de la
   carpeta
    for f in files:
        os.remove(f)        #Comando para eliminar los documentos
```

Apéndice K

Código para realizar las mediciones de los detectores

Una vez instalado el prototipo experimental en las mesas ópticas, se necesita calibrar el experimento. Por lo que, Alice debe de encender el láser en continuo y colocar el retardador de onda $\lambda/2$ en el ángulo 0° para encontrar los ángulos a los cuales se orienta el retardador de onda $\lambda/2$ de Bob. Luego de calibrar a Bob, se deja fijo su retardador de onda $\lambda/2$ y se rota el de Alice. Para esto se necesitan dos códigos, uno para Alice y otro para Bob. El código para Alice es:

```
#Librerías-----------------------------------------------------------
import RPi.GPIO as GPIO
from time import sleep

#Variables-----------------------------------------------------------
GPIO.setmode(GPIO.BOARD)
#Se indica el puerto del servomotor y su frecuencia de trabajo y el á
   ngulo con el que inicia
GPIO.setup(13, GPIO.OUT)
Servo_Alice = GPIO.PWM(13, 50)
Servo_Alice.start(0)
#Se habilita el puerto del láser
GPIO.setup(11, GPIO.OUT)

#Main----------------------------------------------------------------
```

```
#Se establece encendido el puerto del láser
GPIO.output(11,True)

'''
#Una vez encontrados los ángulos de Bob, se descomentan las lineas del 18 y 27 para determinar los ángulos de Alice.

i=0
#El bucle va de 0° hasta 180° debido al rango de movimiento del servomotor
while i<180:
    Servo_Alice.ChangeDutyCycle(((1.0/18.0)*i)+2.5)
    sleep(0.2)
    i+=1
'''
```

El código para Bob es:

```
#Librerías-----------------------------------------------------------
from gpiozero import MCP3008
import RPi.GPIO as GPIO
from time import sleep

#Funciones-----------------------------------------------------------
#Función que permite guardar los datos obtenidos de los detectores
def Save_files(x,z):
    now=datetime.now()
    Document_name = x + ".txt"
    f=open(Document_name,"w")
    for i in z:
        f.write(str(i))
        f.write(", ")
    f.close()

#Variables-----------------------------------------------------------
GPIO.setmode(GPIO.BOARD)
Detector_0 = []
Detector_1 = []
#Detectores
```

```
22 Sensor0 = MCP3008(channel=1)
23 Sensor1 = MCP3008(channel=0)
24 #Se indica el puerto del servomotor, su frecuencia de trabajo y el á
   ngulo con el que inicia
25 GPIO.setup(13, GPIO.OUT)
26 Servo_Bob = GPIO.PWM(13, 50)
27 Servo_Bob.start(0)
28
29 #Main-----------------------------------------------------------------
30 i=0
31 #El bucle va de 0° hasta 180° debido al rango de movimiento del
   servomotor
32 while i<180:
33
34     #Realizar mediciones
35     Detector_0.append(Sensor0.value)
36     Detector_1.append(Sensor0.value)
37
38     #Una vez encontrados los ángulos de Bob, se comenta la linea 39 y
   se reinicia el código para determinar los ángulos de Alice.
39     Servo_Bob.ChangeDutyCycle(((1.0/18.0)*i)+2.5)
40
41     sleep(0.2)
42     i+=1
43
44 #Finalmente se guardan los datos para posteriormente realizar el
   ajuste de los datos con el método deseado.
45
46 Save_files("Detector0B",Detector_0)
47 Save_files("Detector1B",Detector_1)
48
49 #Una vez que se hayan encontrado los ángulos de Bob, se sustituyen los
    nombres de los archivos por "Detector0A" y "Detector1A" para
   encontrar los ángulos de Alice.
```