Universidad Nacional Autónoma de México

Facultad de Ciencias

Sistema de adquisición de datos para el conteo de fotones y detección de coincidencias

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

Físico

PRESENTA:

Erick David Duarte Mendieta

TUTOR:

Dr. Jehú López Aparicio

Ciudad Universitaria, CD. MX. , 2018

Hoja de datos del jurado

1. Datos del alumno

Duarte
Mendieta
Erick David
55 85 81 66 96
Universidad Nacional Autónoma de México
Facultad de Ciencias
Física
414013562

2. Datos del tutor

Dr.
Jehú
López
Aparicio

3. Presidente

Dr.
Víctor Manuel
Velázquez
Aguilar

4. Vocal

Dr.
Carlos Javier
Villagómez
Ojeda

5. Secretario

Dr.
Jehú
López
Aparicio

6. Suplente 1

Dr.
Mathieu Christian Anne
Hautefeuille

6. Suplente 2

Dr.
Ruben Yvan Maarten
Fossion

7. Datos del trabajo escrito

Sistema de adquisición de datos para el
conteo de fotones y detección de coincidencias.
94 p.
2019

A mi esposa, a mis hijos y a mis padres,
por todo su apoyo y cariño

Agradecimientos

A mi asesor, el Dr. Jehú López Aparicio, por todo su tiempo, disponibilidad, apoyo y valiosos comentarios.

Al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) IN116716, "*Estudio de materiales con transición de espín: teoría cuántica y experimento*", por la beca recibida.

Todo el manuscrito que se presenta es de mi absoluta responsabilidad y autoría. Manifiesto que no existen párrafos extraídos textualmente de otras referencias en español o traducidos de otro idioma sin que se hayan citado correctamente.

Índice general

Índice de figuras

Resumen

El conteo de fotones individuales es un problema práctico de gran importancia en el área de la Óptica Cuántica, la cual se encarga de estudiar las consecuencias físicas resultantes de considerar a la luz como un ente cuantizado que se encuentra compuesto por fotones, descrito mediante la teoría de la mecánica cuántica, en lugar de estudiarse desde el enfoque de la teoría clásica del electromagnetismo en el que se estudia a la luz como una onda electromagnética.

Para realizar diversos experimentos en esta área como los experimentos relacionados con estudiar la estadística de fotones de una fuente, o el experimento de Hanbury Brown y Twiss (HBT) para estudiar la correlación entre las intensidades recibidas en distintos detectores de fotones, es necesario contar el número de fotones que inciden sobre un detector por unidad de tiempo y detectar cuando el tiempo de llegada de dos o más fotones a diferentes detectores difiere a lo más por cierto umbral conocido como la ventana de coincidencia.

En este trabajo se describen el software, hardware y firmware desarrollados para construir un dispositivo de instrumentación que sea capaz de realizar estas dos tareas de una manera eficiente, amigable, y que facilite el proceso de adquisición de datos para que el usuario pueda enfocarse más en los resultados de los experimentos que en el procesamiento de los datos obtenidos. El dispositivo desarrollado cuenta con 6 canales de entrada independientes, es capaz de obtener tasas de conteo de hasta 5 millones de cuentas por segundo de forma independiente en cada canal y es capaz de detectar coincidencias para ventanas de hasta 5 ns.

En la parte del software (Capítulo 4) se utilizó el lenguaje de programación Python para desarrollar distintos módulos que le permiten a la computadora comunicarse con el dispositivo para realizar el envío de instrucciones o la extracción de los datos recopilados mediante el protocolo de comunicación UART, producir gráficas interactivas que permitan explorar facilmente los datos obtenidos, y generar archivos en formato *.csv* de forma automática para almacenarlos. Además, se diseñó una interfaz gráfica de usuario con ayuda de la librería PyQt con el objetivo particular de mejorar el proceso de adquisición de datos utilizado actualmente en el laboratorio de Óptica Avanzada de la Facultad de Ciencias, UNAM.

En lo que respecta al hardware (Capítulo 2) se utilizó un FPGA Cylone ll de la compañía Altera para implementar las tareas de detección de coincidencias, conteo de pulsos, y la implementación del protocolo de comunicación UART. También se utilizó una tarjeta de desarrollo Teensy 3.2 para facilitar la comunicación entre el FPGA y la computadora, además de abrir una gran cantidad de posibilidades de trabajo a futuro para implementar nuevas funcionalidades al dispositivo sin tener que realizar un cambio abrupto de su arquitectura.

Finalmente el firmware (Capítulo 3) consiste de un conjunto de módulos diseñados y verificados con ayuda del software Quartus (desarrollado por Altera) y del lenguaje de descripción de hardware Verilog, cuya función es implementar las tareas mencionadas anteriormente y programar al FPGA con el diseño desarrollado.

Los resultados obtenidos (Capítulo 5) muestran que el dispositivo tiene un error máximo de conteo del 0.0006 %, valor obtenido al introducir un señal cuadrada con periodo de 100 kHz en uno de sus canales y realizar un experimento de conteo con duración de un segundo. Para frecuencias mayores como 5 MHz el mismo experimento arroja un error porcentual promedio de aproximadamente 0.00055 %.

La eficiencia del proceso de detección de coincidencias no pudo ser evaluada de una forma objetiva debido a la falta de tiempo y de material apropiado, y su adecuada caracterización se deja como trabajo a futuro. Sin embargo los experimentos realizados con el material disponible muestran que es necesario revisar detalladamente el desempeño de esta tarea al utilizar el valor mínimo permitido por el dispositivo (5 ns) de la ventana de coincidencia, pues los resultados obtenidos no coinciden con lo esperado. Sin embargo, los resultados también muestran que para ventanas de coincidencia de al menos 10 ns el dispositivo funciona correctamente.

Introducción

La automatización de procesos permite transferir tareas a un sistema para mejorar la productividad, reducir costos, mejorar la calidad, reducir el error y realizar operaciones difíciles de controlar de manera manual. La automatización e instrumentación está presente en aplicaciones en distintas áreas, tales como domótica, automatización de procesos industriales, automatización e instrumentación de experimentos científicos, entre otros.

La automatización e instrumentación de experimentos científicos facilita la obtención de datos, agiliza el procesamiento y la obtención de resultados, la transferencia de datos a otros dispositivos, su almacenamiento y la generación de gráficos apropiados que faciliten su interpretación.

En los experimentos de estadística de fotones y detección de coincidencias se emplean módulos que permiten obtener una señal eléctrica como resultado de la detección de fotones individuales. Sin embargo se requiere de un sistema electrónico capaz de procesar dichas señales, ya que los pulsos producidos por los detectores utilizados tienen duraciones del orden de nanosegundos y pueden producirse a frecuencias muy altas que hacen que sean difíciles de contar y estudiar manualmente. Por lo tanto, el sistema electrónico debe ser capaz de realizar el conteo de pulsos individuales, así como de detectar coincidencias entre distintas señales para diferentes valores de la ventana de coincidencia, según las necesidades y la configuración de cada experimento de manera automatizada.

En el mercado existen distintos módulos que permiten la obtención de señales eléctricas a partir de la detección de fotones individuales, sin embargo la mayoría de ellos no realiza ningún procesamiento o almacenamiento de los pulsos producidos ya que las dispositivos encargados de realizar estas tareas suelen venderse por separado. Algunos parámetros de interés relacionados con estos detectores como la eficiencia cuántica, el tiempo muerto y la tasa de cuentas oscuras se definen brevemente en la sección 1.4.1. A continuación se presentan algunos módulos que permiten realizar la detección de fotones individuales:

- C11202-100 single pixel photon counting module for low-light-level detection - Hammatsu (1) : Es capaz de detectar fotones en un rango de longitudes de onda (λ) de 320 a 900 nm con una respuesta máxima para $\lambda = 450$

nm, para la cual tiene una eficiencia cuántica típica del 70 % y una tasa de cuentas oscuras máxima de 100 cuentas por segundo. Es capaz de obtener tasas de conteo de hasta 20 millones de cuentas por segundo.

- Single Photon Counting Modules - Laser Components (2): Algunos de sus productos tienen una eficiencia cuántica de hasta el 70 % para $\lambda = 670$ nm y tasas de cuentas oscuras tan bajas como 10 cuentas por segundo. Presentan un tiempo muerto típico de 45 ns.

- Single Photon Counters - ThorLabs (3): son capaces de detectar fotones con longitudes de onda en el rango de 300 a 900 nm, y tienen una eficiencia cuántica máxima del 35 % para $\lambda = 500$ nm. Presentan tasas de cuentas oscuras típicas de entre 25 y 150 cuentas por segundo y tienen un tiempo muerto de entre 35 y 45 ns.

- PDM Series, Photon Counting and Timing - PicoQuant (4).: Son capaces de detectar fotones con longitudes de onda entre 400 y 1100 nm, cuentan con una eficiencia cuántica máxima del 49 % para $\lambda = 550$ nm y con un tiempo muerto típico de 77 ns.

- ID100 Visible Single-Photon Detector - IDQuantique (5): Son capaces de detectar fotones con longitudes de onda entre 350 y 900 nm, cuentan con un tiempo muerto típico de 45 ns y una eficiencia cuántica máxima para $\lambda = 500$ ns.

A pesar de que el nombre de varios de estos dispositivos incluye la palabra 'contador' no todos son capaces de procesar las señales eléctricas producidas, limitándose únicamente a la tarea de detección de fotones y dejando las tareas de conteo de pulsos, detección de coincidencias y almacenamiento/procesamiento de datos a cargo de dispositivos adicionales como:

- TimeHarp 200 (6): Esta tarjeta está diseñada para el conteo de fotones temporalmente correlacionados. Utiliza un bus PCI para comunicarse con otros dispositivos, es capaz de lograr tasas de conteo de hasta 3 millones de cuentas por segundo, y tiene capacidad de ruteo de 4 canales. Entre otras de sus aplicaciones se encuentran la caracterización de tiempos de respuesta de dispositivos optoelectrónicos, la espectroscopía de moléculas individuales y la microscopía de tiempo de vida de imagen flourescente.

- TimeHarp 260 (7): Esta tarjeta tiene aplicaciones similares a las de la tarjeta anterior. Cuenta con 2 canales independientes, un bus de comunicación PCI express y es capaz lograr tasas de conteo de hasta 40 millones de cuentas por segundo.

- DPC-230 Photon Correlator (8): Esta tarjeta cuenta con 16 canales independientes y es capaz de detectar correlaciones con una ventana de coincidencia mínima de 165 picosegundos. Algunas de sus aplicaciones típicas son la espectroscopia de correlación de fluorescencia y la microscopía de tiempo de vida de imagen flourescente.

- Tarjeta PCI desarrollada en la Facultad de Ciencias de la Electrónica de la BUAP con patente No. MXa2013005105: Esta tarjeta cuenta con 6 canales independientes y es capaz de detectar correlaciones entre señales con una ventana de coincidencia mínima de 5 ns. Puede contar hasta 1 millón de pulsos por segundo y puede ser configurada con una computadora mediante una aplicación desarrollada en LabView.

En este trabajo se describe el diseño elaborado de una tarjeta que permite automatizar experimentos de conteo de fotones y detección de coincidencias. En la figura 1 se muestran esquemáticamente las diferentes partes que lo componen, mientras que los motivos que llevaron a desarrollar este dispositivo y a no optar por alguna de las opciones mencionadas anteriormente se discuten en la siguiente sección.

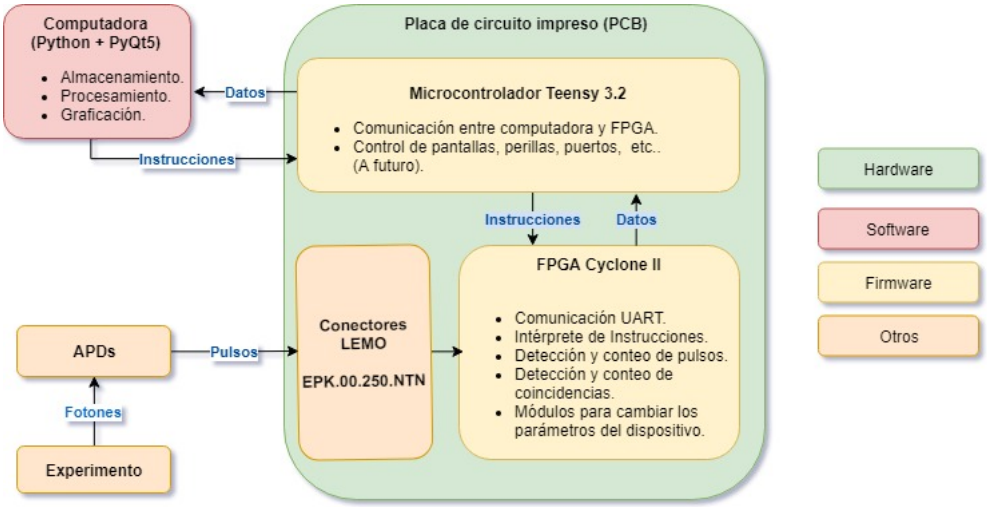

Figura 1: Esquema general del trabajo desarrollado.

Justificación

En el laboratorio de Óptica Avanzada de la Facultad de Ciencias, UNAM, se obtienen datos de distintos experimentos que requieren de grandes cantidades de tiempo y de esfuerzo manual para su obtención y procesamiento. Con el propósito de mejorar la calidad de los resultados obtenidos, el tiempo de procesamiento de los datos y el impacto del error humano sobre los resultados se planteó diseñar un dispositivo para automatizar algunos de los experimentos más comunes que suelen realizarse en el laboratorio de forma habitual, como lo son los experimentos de estadística de fotones y el experimento de Hanbury Brown y Twiss.

Actualmente en el laboratorio de óptica cuántica se cuenta con un contador de fotones SR400 de Stanford Research Systems (9) que cuenta con dos canales de entrada. Sin embargo, este contador no permite almacenar los datos obtenidos en un archivo y el número de entradas disponibles limita la cantidad de experimentos para los que puede ser de utilidad.

Además, se emplean dos tarjetas PCIs diseñadas por la Facultad de Ciencias de la Electrónica de la BUAP. La primera cuenta con una capacidad de conteo de hasta 255000 pulsos por segundo y una ventana de coincidencia mínima de 30ns; la segunda con una capacidad de conteo de hasta 1 millón de pulsos por segundo y con una ventana de coincidencia mínima de 5ns. Ambas tarjetas son controladas mediante una aplicación desarrollada en LabView.

Finalmente, se utiliza una tarjeta propietaria de Altera modelo DE2-115 que permite almacenar la información obtenida y es capaz de detectar coincidencias para ventanas de tiempo de hasta 8ns. Sin embargo, esta tarjeta requiere de hardware adicional para adquirir los datos de los módulos detectores de fotones y utiliza una aplicación desarrollada en LabView para su adquisición y procesamiento.

Como se mencionó en la sección anterior existen distintos módulos que permiten automatizar este tipo de experimentos. Sin embargo, la tarjeta desarrollada tiene las siguientes ventajas:

- Es mucho más económica que las alternativas existentes en el mercado (Aunque las capacidades actuales de esta tarjeta se encuentran más limitadas respecto a las comerciales).
- El software se desarrolló en el lenguaje de programación Python. Python y las diferentes librerías utilizadas son de código abierto por lo que incentivan la colaboración entre diferentes usuarios y facilitan el envío y la publicación de los resultados obtenidos.
- Al ser uno de los lenguajes más utilizados para enseñar programación a principiantes, es relativamente sencillo que el usuario desarrolle sus propios

scripts para comunicarse con el dispositivo y pueda personalizar/automatizar el envío de instrucciones y la extracción de datos de acuerdo a las necesidades individuales de cada experimento.

- Python es un lenguaje multiplataforma por lo que el software es capaz de ejecutarse en distintos sistemas operativos sin tener que modificar su código fuente.
- Al ser un software desarrollado con herramientas de código libre no es necesario comprar ninguna licencia adicional para utilizar el dispositivo, lo que disminuye su costo total.
- La tarjeta de desarrollo Teensy 3.2 abre una gran cantidad de oportunidades para añadir nuevas modalidades y periféricos al dispositivo a medida que sean requeridas por el usuario y el experimento.
- El dispositivo desarrollado es sumamente portátil y para utilizarlo únicamente es necesario contar con computadora personal o una laptop, gracias a su puerto de entrada USB.
- Al conocer cómo se encuentra programado el software, el firmware, y las capacidades y limitaciones del hardware utilizado es posible elaborar módulos y funciones hechas a la medida para satisficer las futuras necesidades del Laboratorio de Óptica Avanzada.

Objetivos

El objetivo principal del siguiente proyecto es ***Diseñar e implementar un sistema de adquisición de datos para el conteo individual de fotones y detección de coincidencias*** y se divide en los siguientes objetivos particulares:

- Diseñar el software en el lenguaje de programación Python.
- Diseñar el Firmware en un FPGA.
- Hacer pruebas funcionales utilizando instrumentos comerciales.
- Diseñar el hardware (Tarjeta de Circuito Impreso).
- Implementar un módulo de comunicación serial.
- Hacer pruebas funcionales en el Laboratorio de Óptica Avanzada.
- Hacer pruebas de replicabilidad y estadística.

- Comparar los resultados obtenidos contra un contador de fotones comercial.
- Publicar resultados.
- Redacción de documento

Capítulo 1

Introducción y fundamentos

En este capítulo se discuten el contexto y los conceptos básicos necesarios para entender el propósito del dispositivo de instrumentación elaborado, algunas de sus aplicaciones y la teoría básica para comprender el papel que juega en el proceso experimental. También se discuten conceptos básicos acerca de la detección y conteo de fotones individuales, la estadística de fotones y la detección de fotones temporalmente correlacionados.

1.1. Electromagnetismo

En esta sección se discuten de manera breve los conceptos básicos de la teoría clásica del Electromagnetismo necesarios para el desarrollo del presente trabajo, como lo son las ecuaciones de Maxwell, el enfoque clásico de la luz como ondas electromagnéticas, el concepto de polarización de una onda y los conceptos de coherencia espacial y coherencia temporal de una fuente luminosa.

1.1.1. Ecuaciones de Maxwell

La teoría del Electromagnetismo en su forma matemática actual fue desarrollada por el científico escosés James Clerk Maxwell en lo que se ahora se conocen como las Ecuaciones de Maxwell. Estas cuatro ecuaciones logran resumir el trabajo de diversos científicos como Ampère, Faraday, Gauss y Coulomb, entre otros, de una manera concisa y elegante. Las ecuaciones de Maxwell en su forma diferencial son (10, eq. (7.56)):

$$\nabla \cdot \vec{D} = \rho_L \tag{1.1}$$

$$\nabla \cdot \vec{B} = 0 \tag{1.2}$$

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{1.3}$$

$$\nabla \times \vec{H} = \vec{J}_L + \frac{\partial \vec{D}}{\partial t} \tag{1.4}$$

Estas ecuaciones establecen relaciones entre el comportamiento de los campos de desplazamiento eléctrico $\vec{D}$, el campo auxiliar $\vec{H}$, el campo magnético $\vec{B}$ y el campo eléctrico $\vec{E}$ y sus respectivas fuentes que se conocen como la densidad de carga libre ρ_L y la densidad de corriente libre $\vec{J}_L$.

Cabe aclarar que estamos utilizando la notación actual en la que $\vec{B}$ denota a el campo vectorial que satisface la ley de Biot-Savart y $\vec{H}$ al campo vectorial que toma en cuenta la respuesta magnética del material. Esta observación es importante debido a que en el pasado algunos textos solían utilizar la convención contraria y solían denotar al campo que satisface la ley de Biot-Savart como $\vec{H}$ (10, p. 285). Así, estos dos campos se encuentran relacionados mediante la siguiente expresión (10, eq. (6.18)):

$$\vec{H} = \frac{1}{\mu_0}\vec{B} - \vec{M} \tag{1.5}$$

donde μ_0 es una constante conocida como la permeabilidad magnética del vacío y $\vec{M}$ es la magnetización del medio, definida como el momento dipolar magnético por unidad de volumen.

Respecto a su comportamiento eléctrico, para medios isotrópicos los dipolos eléctricos generados al someter al material a un campo eléctrico externo $\vec{E}$ apuntan en la misma dirección que éste, y para medios lineales la polarización del material se encuentra caracterizada por la siguiente expresión (10, eq. (4.30)):

$$\vec{P} = \epsilon_0 \chi \vec{E} \tag{1.6}$$

donde $\epsilon_0 = 8.854 \times 10^{-12}$ F m^{-1} se conoce como la permitividad eléctrica del vacío, χ es una constante conocida como la susceptibilidad eléctrica del medio y $\vec{P}$ es el momento eléctrico dipolar por unidad de volumen.

La aproximación lineal de la ecuación 1.6 usualmente solo es válida cuando la amplitud del campo eléctrico externo es pequeño. A medida que la amplitud del campo eléctrico aumenta es necesario tomar en cuenta potencias superiores

de $\|E\|$ de tal forma que entramos en el régimen de la óptica no lineal, en donde (11, eq.(2.56)):

$$\|P\| = \epsilon_0 \chi^{(1)} \|E\| + \epsilon_0 \chi^{(2)} \|E\|^2 + \epsilon_0 \chi^{(3)} \|E\|^3 + \dots \tag{1.7}$$

Al término $\|P\|^{(n)} = \epsilon_0 \chi^{(n)} \|E\|^n$ se le conoce como el término de polarización no lineal de orden n, y al coeficiente $\chi^{(n)}$ se le conoce como la suceptibilidad no lineal de orden n (11, p. 20).

1.1.2. Ondas electromagnéticas

A partir de las ecuaciones de Maxwell y álgebra vectorial (10, p. 398-399) es posible demostrar que estas ecuaciones admiten soluciones ondulatorias, y que éstas se propagan en el vacío con una velocidad de $c = 1/\sqrt{\epsilon_0 \mu_0} = 299\,792\,458\,\mathrm{m/s}$ (11, eq.(2.18)).

Las soluciones ondulatorias usuales son ondas transversales en las que los campos eléctrico y magnético oscilan perpendicularmente entre ellos y a la dirección de propagación de la onda, y cuyos ejes de oscilación son tales que el vector $\vec{E} \times \vec{B}$ apunte en la dirección de propagación.[1] Para el caso particular de una onda electromagnética con longitud de onda λ y frecuencia f que se desplaza sobre el eje z y cuyo campo eléctrico asociado oscila sobre el eje x las ecuaciones que describen a E_x y B_y son (10, eq. (9.48)):

$$\vec{E} = E_0 \cos(kz - \omega t - \phi)\, \hat{x} \tag{1.8}$$

$$\vec{B} = B_0 \cos(kz - \omega t - \phi)\, \hat{y} \tag{1.9}$$

en donde a $k = 2\pi/\lambda$ se le conoce como el número de onda, a $\omega = 2\pi f$ como la frecuencia angular, y la relación entre λ y f está dada por $c = \lambda f$. A este tipo de soluciones se les conoce como ondas electromagnéticas planas.

Las amplitudes de los campos electromagnéticos de una onda monocromática plana no son independientes ya que se encuentran relacionados por la siguiente expresión (10, eq. (9.47)):

$$B_0 = \frac{E_0}{c} \tag{1.10}$$

[1] Siempre y cuando la onda electromagnética se propague en un medio isotrópico.

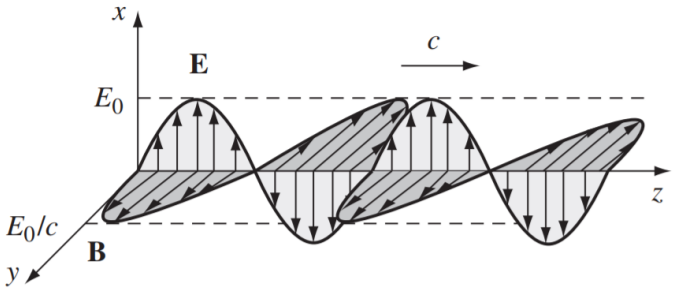

Figura 1.1: Onda electromagnética plana descrita por las ecuaciones 1.8 y 1.9 (10, Fig. 10).

En general suelen utilizarse números complejos para describir a este tipo de ondas cuando viajan en una dirección arbitraria, ya que su representación polar permite simplificar considerablemente los cálculos al considerar la superposición de varias de ellas y además en esta representación la amplitud y la fase de cada onda (cantidades que suelen ser de interés) pueden obtenerse directamente sin realizar cálculos adicionales. En esta representación su amplitud está dada por la parte real de la siguiente expresión (10, eq. (9.18)):

$$\|E\| = \mathrm{Re}\left(E_0 \exp i(\vec{k}\cdot\vec{r} - \omega t - \phi)\right) \tag{1.11}$$

donde $\vec{k}$ es un vector conocido como vector de onda cuya magnitud es igual a el número de onda y cuya dirección coincide con la dirección de propagación para medios isotrópicos.

Se define la intensidad de una onda electromagnética como la potencia promedio por unidad de área que ésta transporta. Para una onda monocromática plana la relación entre su intensidad y su campo eléctrico asociado está dada por (11, eq. (2.28)):

$$I = \frac{nc\epsilon_0}{2}\|\vec{E}\|^2 \tag{1.12}$$

donde n es el índice de refracción del medio en el que se propaga la onda. Así, la intensidad de una fuente monocromática es proporcional al cuadrado de la amplitud de su campo eléctrico.

1.1.3. Polarización

—

La polarización de una onda electromagnética es una propiedad que describe la forma en la que cambia su campo eléctrico asociado a lo largo del tiempo. Esta

puede ser de varios tipos:

- Lineal: La dirección del eje de oscilación del campo eléctrico se mantiene constante a lo largo del tiempo, por lo que el campo solo cambia su magnitud y sentido pero no su dirección.
- Circular: La magnitud del campo eléctrico es independiente del tiempo mientras que su dirección cambia de una forma tal que la punta del vector $\vec{E}$ traza un circulo en cualquier plano perpendicular a la dirección de propagación en un punto dado del espacio.
- Elíptica: Tanto la magnitud y dirección del campo eléctrico cambian de tal forma que la punta del vector $\vec{E}$ traza una elipse en cualquier plano perpendicular a la dirección de propagación en un punto dado del espacio.
- No polarizada: La dirección de $\vec{E}$ en cada instante de tiempo es aleatoria.

Por ejemplo, la onda electromagnética mostrada en la figura 1.1 se encuentra linealmente polarizada sobre el eje $\hat{x}$ mientras que la onda mostrada en la figura 1.2 se encuentra circularmente polarizada. El campo magnético asociado a esta última onda será perpendicular a el campo eléctrico en todo momento, por lo que de forma similar al campo eléctrico el vector $\vec{B}$ trazará círculos en planos perpendiculares a la dirección de propagación a lo largo del tiempo.

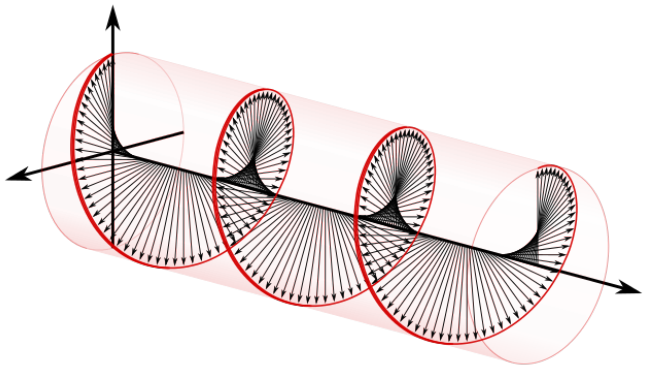

Figura 1.2: Campo eléctrico de una onda electromagnética circularmente polarizada.

1.1.4. Coherencia

Una onda monocromática es aquella que en su representación matemática se encuentra descrita únicamente por una sola frecuencia. Por lo general cualquier fuente luminosa estará compuesta por una superposición de ondas monocromáticas con distintas frecuencias angulares, por lo que muchas veces el parámetro

de interés es el rango de frecuencias angulares $\Delta\omega$ presentes en la fuente cuya contribución es mayor que cierto umbral especificado, conocido como su ancho espectral. Es claro que a medida que el ancho espectral de una fuente tienda a cero ésta tendrá un comportamiento cada vez más similar al de una fuente monocromática ideal.

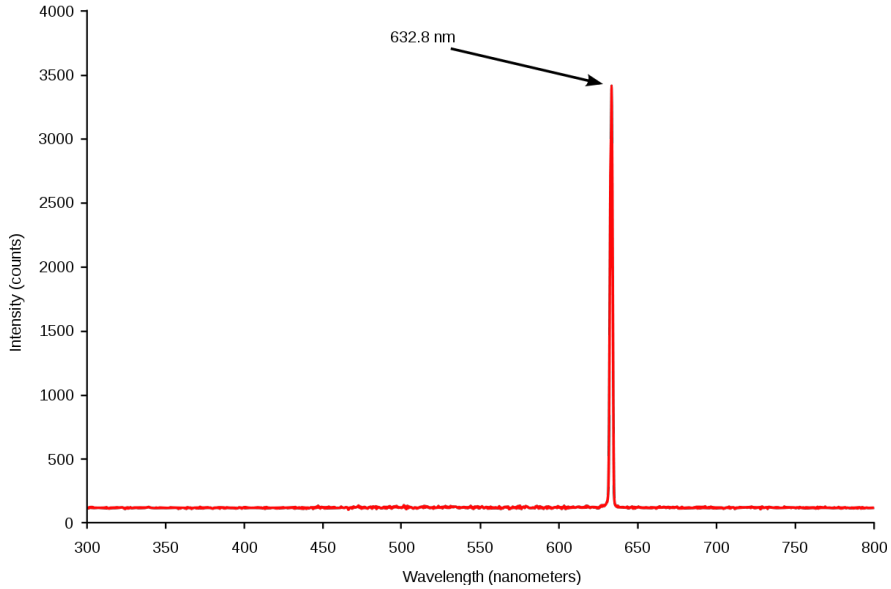

Figura 1.3: Espectro típico de la luz emitida por un láser rojo de Helio-Neón ($\Delta w \approx 5$ nm).

La coherencia es una forma de cuantificar la estabilidad de una fuente en términos de la incertidumbre que existe en la variación de su fase. Existen dos tipos distintos de coherencia conocidos como coherencia temporal y coherencia espacial, que se describen brevemente a continuación.

1.1.4.1. Coherencia temporal

La coherencia temporal es una medida de la correlación que existe entre la fase que tiene una onda en un punto fijo x en un tiempo t y la fase de la misma onda en el mismo punto pero en un tiempo posterior. Esta correlación está cuantificada por dos parámetros conocidos como el tiempo de coherencia τ_c y la longitud de coherencia L_c de la fuente, que se encuentran relacionados por la siguiente ecuación (11, eq.(2.40)):

$$L_c = c\tau_c \tag{1.13}$$

Si inicialmente conocemos la fase de una onda en un punto x en un tiempo t,

el grado de certeza con el que conoceremos la fase en un tiempo posterior $t + \Delta t$ dependerá de que tan grande sea el valor de Δt respecto al tiempo de coherencia. Para $\Delta t \ll \tau_c$ seremos capaces de conocer con un alto grado de certeza la fase de la onda, mientras que en el caso contrario en que $\Delta t \gg \tau_c$ no tendremos certeza alguna de su valor.

En general el tiempo de coherencia está relacionado con el ancho espectral de la fuente de la siguiente manera (11, eq.(2.41)):

$$\tau_c \approx \frac{1}{\Delta\omega} \tag{1.14}$$

Así, el tiempo de coherencia en general está relacionado con la "pureza"de la fuente ya que mientras sea menor su contenido de frecuencias el tiempo de coherencia será cada vez mayor. Para una onda monocromática que tiene un ancho espectral nulo su tiempo de coherencia será infinito, mientras que para una fuente térmica que contiene una gran diversidad de frecuencias angulares será muy bajo. Como puede observarse en la figura 1.3 el ancho espectral de un láser es tan pequeño que prácticamente puede ser considerado como una onda monocromática ideal.

1.1.4.2. Coherencia espacial

La coherencia espacial es una forma de cuantificar la correlación existente entre la fase de una onda electromagnética en diferentes puntos del espacio en el mismo instante de tiempo. Intuitivamente, si en un tiempo t en promedio el conocer la fase de una onda en un punto x es capaz de aportar información acerca del valor de la fase en otro puntos del espacio entonces se dice que la fuente presente una alta coherencia espacial. En caso contrario, si en promedio conocer la fase en un punto dado no guarda ninguna relación con el valor de la fase en otros puntos de la onda la fuente tendrá una coherencia espacial muy baja.

1.2. Interacción entre la luz y la materia

Existen distintos enfoques para resolver problemas que requieran analizar la interacción entre la luz y la materia, los cuales son:

1. Clásico: La luz es tratada como una onda electromagnética usual, mientras que el efecto que tiene sobre la materia es modelado mediante dipolos electromagnéticos clásicos sin que exista ninguna noción de cuantización.

2. Semi-clásico: La luz es tratada clásicamente pero ahora se toma en cuenta la mecánica cuántica para describir a los átomos, por lo que ahora se considera la cuantización de sus estados y energías posibles.

3. Cuántico: La mecánica cuántica sigue siendo utilizada para describir a la materia, y adicionalmente ahora se toma en cuenta que en realidad la luz también se encuentra cuantizada en forma de fotones.

El enfoque cuántico es el único que es capaz de explicar todos los fenómenos observados, sin embargo el enfoque semi-clásico suele ser capaz de explicar satisfactoriamente muchos de ellos. Sin embargo existen otros como la emisión espontánea(12) o el efecto Lamb (13) que únicamente pueden ser explicados cuando se toma en cuenta la naturaleza cuántica de la luz.

La Óptica Cuántica es la rama de la Física que se encarga de estudiar los fenómenos ópticos utilizando el enfoque cuántico descrito anteriormente.

1.3. Absorción

Cuando una onda electromagnética interactúa con la materia pueden ocurrir diferentes cosas dependiendo de la energía de los fotones incidentes y de la configuración interna de los átomos y moléculas que componen al material (14, p. 77).

Si la energía de los fotones incidentes coincide con la energía necesaria para que los electrones del material realicen una transición hacia un estado excitado, es muy probable que los electrones pierdan esta energía adicional al colisionar con los demás átomos del material antes de que puedan realizar una transición a un estado con menor energía y liberarla en forma de fotones. En este caso la energía de los fotones incidentes se convirtió en energía térmica que fue absorbida por el material, y a este proceso se le conoce con el nombre de *absorción disipativa*.

Si la energía de los fotones incidentes es tal que no permite que los electrones del material realicen transiciones hacia estados excitados, la onda electromagnética incidente los hará entrar en movimiento oscilatorio por lo que los electrones comenzarán a liberar radiación electromagnética con la misma frecuencia que la onda incidente.

El mecanismo de absorción disipativa es en el que está basado el funcionamiento de distintos tipos de detectores de fotones, ya que es necesario que los fotones incidentes logren ionizar a los electrones del material para que estos produzcan corrientes eléctricas o diferencias de voltaje medibles experimentalmente.

1.4. Detección y conteo de fotones

La capacidad de detectar y contar fotones individuales es importante en diversas aplicaciones y ramas de la física, entre las cuales destacan:

- Telemetría láser (15).
- Conteo de fotones individuales temporalmente correlacionados.[1]
- Criptografía cuántica.
- Microscopía de tiempo de vida de imágen fluorescente (16).[2]
- Estadística de fotones.

En este trabajo nos centraremos en las áreas de estadística de fotones y en el conteo de fotones temporalmente correlacionados, debido a que los experimentos que suelen realizarse con mayor frecuencia en el Laborartorio de Óptica Avanazada son de estas áreas. Para ser capaces de detectar fotones individuales debemos de estar en un régimen en el que la intensidad de la luz incidente sea lo suficientemente baja. Esto es así debido a que en caso contrario los fotones que componen la luz estarán tan cerca temporalmente que los detectores utilizados, usualmente fotodiodos de avalancha (APD, *Avalanche Photo-Diode*), no serán capaces de distinguirlos por separado.

1.4.1. Fotodiodo de avalancha

Un APD es un dispositivo semiconductor capaz de convertir señales luminosas de muy baja intensidad, como lo son los fotones individuales, en señales eléctricas macroscópicas detectables.

En el Laboratorio de Óptica Avanzada de la FC-UNAM se utilizan los modulos detectores SPCM-AQRH-13-FC y el SPCM-AQ-4C, ambos de la compañía PerkinElmer.

A continuación se describen algunos parámetros y propiedades importantes de los detectores al realizar experimentos de detección y de conteo de fotones:

- Los APDs responden de diferentes maneras a distintas longitudes de onda. Como se describió brevemente en la sección 1.3, en general los materiales

[1] En inglés: TCSPC (Time-Correlated Single Photon Counting)
[2] En Inglés: FLIM (Flourescence Lifetime Imaging Microscopy)

responden de diferentes maneras dependiendo de la relación existente entre la energía de los fotones incidentes y los niveles energéticos de los átomos del material.

- Aunque no incida luz sobre ellos los APDs pueden producir pulsos eléctricos. Esto en su mayoría es debido al ruido térmico. Al número promedio de éstas cuentas falsas por unidad de tiempo se le conoce como la tasa de *cuentas oscuras*.

- Los APDs no pueden producir un pulso eléctrico inmediatamente después de haber producido un pulso anterior. Al tiempo que necesita el detector para poder responder a un nuevo evento se le conoce como su *tiempo muerto*.

- La *eficiencia cuántica* de un fotodetector es la fracción de fotocuentas detectadas entre el número de fotones incidentes. Existen diversas formas de medir experimentalmente esta cantidad. La más sencilla consiste en utilizar un medidor de potencia óptica para medir la potencia de la luz incidente y así estimar el número de fotones por unidad de tiempo que inciden sobre el detector, después de haber filtrado la fuente luminosa para únicamente dejar pasar los fotones con la longitud de onda deseada. El número de fotodetecciones puede ser estimado al medir la corriente producida por el APD al hacer incidir la fuente de luz sobre él y expresarla en unidades de electrones por unidad de tiempo.

A continuación se describen los dos modelos distintos de APDs que se utilizan en este laboratorio. El motivo principal por el que se cuenta con varios tipos de ellos es por que el costo del módulo SPCM-AQ-4Q (que tiene 4 canales independientes) es mucho menor que el de los 4 módulos SPCM-AQRG-13-FC que tendrían que utilizarse en su lugar para contar con el mismo número de canales.

1.4.1.1. SPCM-AQ-4C

Este módulo tiene 4 canales independientes y es capaz de detectar fotones individuales con longitudes de onda entre 400 y 1600 nm. Este módulo utiliza un fotodiodo de avalancha que tiene una eficiencia cuántica del 60 % para $\lambda = 650$ nm. Cada fotón detectado es convertido en un pulso eléctrico de 4.5V con un ancho de 25 ns. Además, este módulo tiene un tiempo muerto de 50 ns. El pulso eléctrico producido por el módulo al detectar un fotón se muestra en la figura 1.4. La hoja de datos de este APD puede consultarse en (17).

1.4.1.2. SPCM-AQRH-13-FC

Este módulo tiene una eficiencia cuántica del 65 % y 55 % para $\lambda = 650$ nm y $\lambda = 810$ nm, respectivamente. A diferencia del módulo anterior cuenta con un único canal y la señal eléctrica generada por este dispositivo tiene una amplitud de 2.5V, un ancho de pulso de 35 ns, y un tiempo muerto de 50 ns.

1.4.2. Tubos fotomultiplicadores

Los tubos fotomultiplicadores son otro tipo de dispositivos muy populares para convertir señales eléctricas de muy baja intensidad en pulsos eléctricos detectables. Estos dispositivos cuentan con una superficie fotosensible, y cuando la luz incide sobre ella se emiten electrones que son acelerados mediante una diferencia de potencial aplicada hacia un electrodo (dínodo) cuyo material tiene la propiedad de que cuando éstos electrones acelerados impactan sobre él se liberan electrones adicionales con energías menores (electrones secundarios).

Estos electrones secundarios son acelerados nuevamente hacia otro dínodo con el mismo propósito, por lo que al repetir este proceso varias veces el número de electrones que se liberaron inicialmente en la superficie fotosensible puede amplificarse considerablemente. Cuando todos estos electrones inciden sobre el ánodo del tubo fotomultiplicador finalmente se produce un pulso eléctrico detectable.

Estos detectores suelen ser muy populares debido a su alta ganancia y bajo ruido, sin embargo no son utilizados en el Laboratorio de Óptica Avanzada debido a que su eficiencia cuántica suele ser dos o tres ordenes de magnitud menor (18) que la de los APDs.

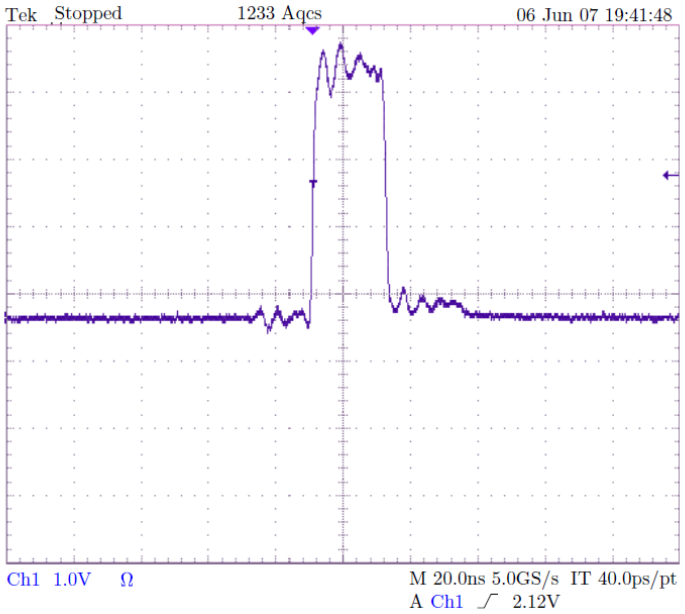

Figura 1.4: Señal eléctrica producida por el módulo SPCM-AQ-4C medida con ayuda de un osciloscopio.

1.5. Estadística de fotones

La estadística de fotones consiste en estudiar las modificaciones estadísticas resultantes de considerar a la luz como un conjunto de fotones en lugar de una onda electromagnética clásica al realizar experimentos de conteo de fotones. Cuando un rayo de luz de baja intensidad incide sobre un fotodetector éste producirá un pulso eléctrico cada vez que un fotón individual sea detectado. En la práctica el tiempo muerto de los detectores establece una cota superior sobre la tasa de conteos máxima que podemos medir con ellos.

A pesar de que la fuente de luz incidente tenga una intensidad promedio fija el número de cuentas detectadas por unidad de tiempo presenterá variaciones respecto a su valor promedio, y estas variaciones serán cada vez mayores al disminuir el tiempo de conteo. Esto es debido a que en realidad no conocemos la forma en la que los fotones se encuentran distribuidos dentro del haz luminoso, por lo que aún al considerar dos secciones del haz de una misma longitud no existirá necesariamente el mismo número de fotones en ambas secciones. El resultado de lo anterior y de considerar que en cualquier sección del haz debe de existir un número entero de fotones tiene como consecuencia la variación en el número de detecciones por unidad de tiempo.

Son estas variaciones las que permiten clasificar a la luz de acuerdo a su

comportamiento estadístico y encontrar uno de esos extraños fenómenos que no tiene una explicación clásica ni semi-clásica, pero que sí puede explicarse al tomar en cuenta la cuantización de la luz.

Cabe aclarar que el hecho de que los fotodetectores produzcan pulsos eléctricos discretos cuando un rayo de muy baja intensidad incide sobre ellos no es evidencia lo suficientemente fuerte para poder concluir la existencia de la cuantización de la luz, ya que se ha demostrado que el enfoque semi-clásico es suficiente para explicar este comportamiento (11, p. 3). Estos experimentos se describen brevemente debido a que suelen realizarse regularmente en el Laboratorio de Óptica Avanzada, y automatizarlos es uno de los principales propósitos del presente trabajo.

1.5.1. Clasificación estadística de la luz

Cualquier fuente luminosa puede ser clasificada en términos de su estadística de fotones de la siguiente manera. Sea $P(n)$ la probabilidad de que n fotones incidan sobre el detector en un intervalo de tiempo fijo T, y sean $\bar{n}$ y Δn el valor promedio y la desviación estándar de ésta distribución, respectivamente.

Considerando detectores idealizados con una eficiencia cuántica del 100 % seríamos capaces de obtener una muestra de esta distribución de probabilidad fijando un tiempo de conteo T en el que registraremos el número de fotodetecciones, repitiendo esta tarea varias veces, y realizando un histograma de las frecuencias relativas de los conteos observados. Una representación esquemática de este procedimiento se muestra en la figura 1.5.

De acuerdo a la relación existente entre $\bar{n}$ y Δn podemos clasificar a una fuente luminosa en las siguientes 3 categorías:

- Luz Sub-Poissoniana: Son aquellas fuentes para las que $\Delta n < \sqrt{\bar{n}}$.
- Luz Poissoniana: Son quellas fuentes para las que $\Delta n = \sqrt{\bar{n}}$.
- Luz Super-Poissoniana: Son aquellas fuentes para las que $\Delta n > \sqrt{\bar{n}}$.

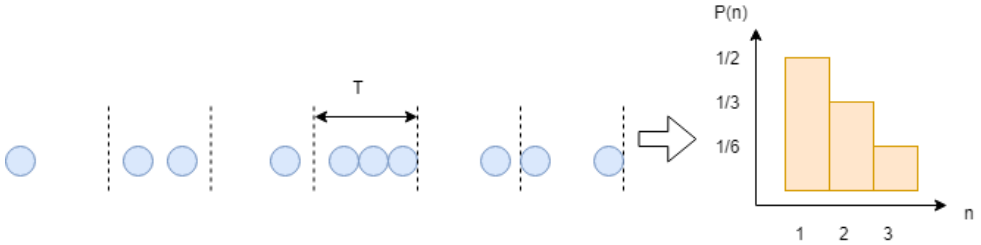

Figura 1.5: Representación esquemática de un experimento de estadística de fotones con duración $6T$ y estimación de la distribución $P(n)$ mediante el histograma de frecuencias relativas de los conteos observados.

Clásicamente la fuente de luz más estable es una fuente monocromática perfectamente coherente con una intensidad constante, para la cual puede demostrarse que su estadística de fotones esperada cae dentro del régimen Poissoniano. Que esta fuente presente una estadística Poissoniana es consecuencia de asumir que si dividimos una sección de longitud L del haz en N subsecciones de longitud L/N existe la misma probabilidad de encontrar un fotón en cualquiera de estas subsecciones. Un ejemplo aproximado de este tipo de luz es un láser cuyo ancho espectral es muy pequeño.

Intuitivamente es claro que en el caso en que la fuente de luz tenga una intensidad dependiente del tiempo, la naturaleza discreta de los fotones tendrá como consecuencia una mayor variación en el número de conteos detectados por unidad de tiempo, por lo que esperamos que este tipo de luz siga una estadística Super-Poissoniana. Como puede mostrarse, un ejemplo de este tipo de luz es la luz térmica.

Sin embargo, la distribución $P(n)$ en el caso Sub-Poissoniano tiene un ancho menor que en el caso Poissoniano por lo que de alguna manera este tipo de luz es más estable que el ideal clásico. Así, al no tener una contraparte clásica, este tipo de luz sólo puede ser explicado al considerar el paradigma cuántico. Diversas observaciones experimentales de este tipo de fuentes luminosas son una prueba contundente de la necesidad de considerar al enfoque cuántico de la luz como el modelo mas completo para describir a los fenómenos ópticos (19).

1.6. Fotones temporalmente correlacionados

En esta sección se discute el experimento realizado por R. Hanbury Brown y R. Q. Twiss en los años cincuenta, en el que se estudió la variación de las fluctuaciones de intensidad dependientes del tiempo de un haz luminoso (20). Este experimento lleva de una manera natural a una caracterización alterna de las fuentes luminosas en términos de lo que se cononce como su función de correlación de segundo orden $g^{(2)}(\tau)$.

De igual forma que con la clasificación estadística, esta nueva clasificación de la luz en términos de $g^{(2)}(\tau)$ trae consigo la aparición de otro fenómeno para el cual la descripción clásica y semi-clásica tampoco es satisfactoria. Este experimento se realiza regularmente en el Laboratorio de Óptica Avanzada para demostrar experimentalmente la necesidad del enfoque cuántico, cuya automatización es uno de los objetivos principales del presente trabajo. El propósito de las siguientes secciones es describir brevemente estos experimentos, así como desarrollar la intuición de qué es lo que se desea estudiar en ellos.

1.6.1. El interferómetro estelar de Michelson

El interferómetro estelar de Michelson fué utilizado por primera vez en el mes de diciembre de 1920 para realizar la primera medición del diámetro de la estrella Betelgeuse (21). Un esquema de este interferómetro se muestra en la figura 1.6.

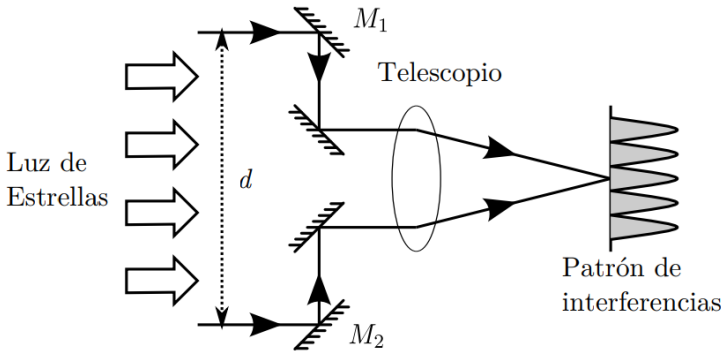

Figura 1.6: Interferómetro estelar de Michelson.

El principio de funcionamiento de este inferferómetro es el siguiente: la luz proveniente de un objeto distante es reflejada por dos espejos M_1 y M_2 que se encuentran separados por una distancia d, en donde posteriormente se dirije a través de rendijas distintas a la entrada de un telescopio. Si la luz reflejada por ambos espejos es coherente, entonces se formará un patrón de interferencia en el plano focal del telescopio. En caso contrario no se formará patrón de interferencia alguno por lo que simplemente se sumarán las intensidades de ambos haces. El experimento consiste en estudiar la variación en la visibilidad del patrón de interferencia a medida que se varía la distancia d entre los espejos.

Es importante recalcar que en este caso el factor determinante para que exista o no un patrón de interferencia es el hecho de que la luz recibida por ambos espejos sea coherente espacialmente. La razón de esto es que al considerar que una estrella no es una fuente puntual de luz sino más bien una fuente extendida, la luz que incide sobre los espejos puede provenir de diferentes puntos de la estrella con diferentes ángulos de incidencia. Cada ángulo de incidencia distinto tendrá un patrón de interferencia correspondiente pero desplazado uno respecto de otro por una distancia que depende de su ángulo. Si no fijamos adecuadamente la distancia entre los espejos d es posible que estos distintos patrones de interferencia se superpongan de tal forma que las línas oscuras de uno coincidan con las líneas brillantes de otro y el patrón de interferencia resultante sea nulo.

El experimento permite encontrar el tamaño angular $\delta\theta_s$ de la estrella observada, por lo que si conocemos la distancia de la tierra a la que se encuentra la

estrella podemos encontrar su diámetro con la siguiente ecuación (11, eq.(6.1)):

$$\delta\theta_s = \frac{D}{L} \tag{1.15}$$

donde L es la distancia de la tierra a la estrella y D el diámetro de la estrella.

Puede demostrarse que para este interferómetro su resolución angular[1] está dada por (11, eq.(6.2)):

$$\delta\theta_r = 1.22\frac{\lambda}{d} \tag{1.16}$$

donde λ es la longitud de onda de la luz incidente y d la separación entre los espejos.

Si $\delta\theta_r \gg \delta\theta_s$ para todos los valores prácticos de d la luz procedente de diferentes puntos de la estrella no será distinguida como diferente por el interferómetro, por lo que la luz incidente será coherente espacialmente y sí existirá un patron de interferencia. En el caso en que existan valores prácticos de d para los que $\delta\theta_s > \delta\theta_r$ la luz procedente de diferentes puntos de la estrella si podrá ser distinguida como proveniente de distintas partes de su superficie por lo que el patrón de interferencia observado será nulo.

Así, al variar d y estudiar la visibildad del patrón de interferencia podremos encontrar $\delta\theta_s$ y obtener el diámetro de la estrella con la ecuación 1.15 siempre y cuando L sea conocida.

1.6.2. El interferómetro de HBT

Como lo indica la ecuación 1.16 una ventaja importante del interferómetro de Michelson es que la distancia entre los espejos d puede ser mayor que la óptica del telescopio, por lo que la resolución angular $\delta\theta_r$ del inteferómetro en principio puede ser mejorada tanto como se desee.

Sin embargo al aumentar d cada vez es más difícil mantener a los espejos M_1 y M_2 lo suficientemente estables como para poder observar franjas de interferencia, debido a que distancias tan grandes cualquier ligera perturbación en la posición de los espejos se verá reflejada en cambios abruptos en el patrón de interferencia resultante. Para resolver este problema Hanbury y Twiss propusieron el arreglo mostrado en la figura 1.7 (20).

Una ventaja de este arreglo respecto al mostrado en la figura 1.6 es que en el nuevo arreglo no es necesario utilizar óptica adicional para observar un patrón de

[1] La resolución angular es la separación angular mas chica tal que dos fuentes puntuales de luz pueden ser distinguidas individualmente por el instrumento.

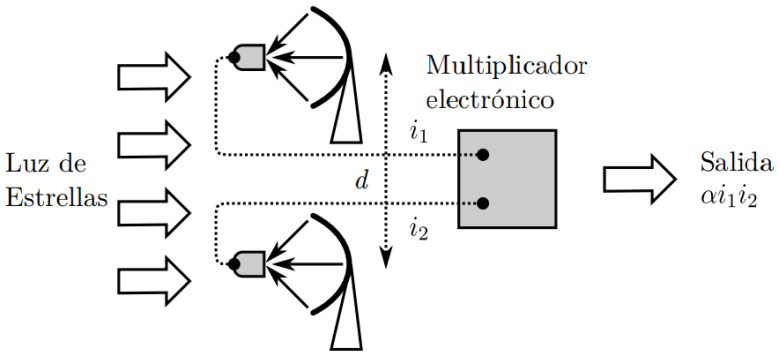

Figura 1.7: Interferómetro de Hanbury Brown y Twiss.

interferencia, ya que en su lugar son utilizados tubos fotomultiplicadores y circuitos eléctricos. Este inteferómetro mide las correlaciones entre las fotocorrientes $i_1(t)$ e $i_2(t)$ producidas por la luz estelar que incide sobre ambos fotomultiplicadores. Esto es realizado mediante un circuito multiplicador de tal forma que la salida es proporcional al promedio temporal de $i_2(t)i_2(t)$, que a su vez es proporcional al producto de las intensidades $I_1(t)$ e $I_2(t)$ de la luz que incide sobre los fotomultiplicadores.

Si d es pequeño entonces $I_1(t)$ e $I_2(t)$ serán prácticamente iguales, pero a medida que d aumente los detectores serán capaces de diferenciar que la luz que incide sobre ellos proviene de diferentes ángulos de la estrella. Así, a medida que d varíe también lo hará la salida del experimento, por lo que la información obtenida de esta manera también nos permitirá encontrar $\delta\theta_s$.

1.6.3. Los experimentos de HBT y las fluctuaciones clásicas de intensidad

Una de las formas con las que Hanbury Brown y Twiss probaron su arreglo fue con la configuración mostrada en la figura 1.8.

En este arreglo la luz correspondiente a la línea 435.8 nm de una lámpara de descarga de mercurio incide sobre un espejo semi-plateado, donde es separada en haces trasmitido y reflejado que inciden sobre dos tubos fotomultiplicadores, generando fotocorrientes $i_1(t)$ e $i_2(t)$.

La salida de los tubos fotomultiplicadores se encuentra conectada a amplificadores acoplados a corriente alterna cuyas salidas son proporcionales a las variaciones de intensidad Δi_1 e Δi_2. Una de estas salidas fue introducida a un generador de retardo de tiempo electrónico establecido con un valor τ. Por último, la salida de este generador y la salida del otro amplificador fueron introducidas a

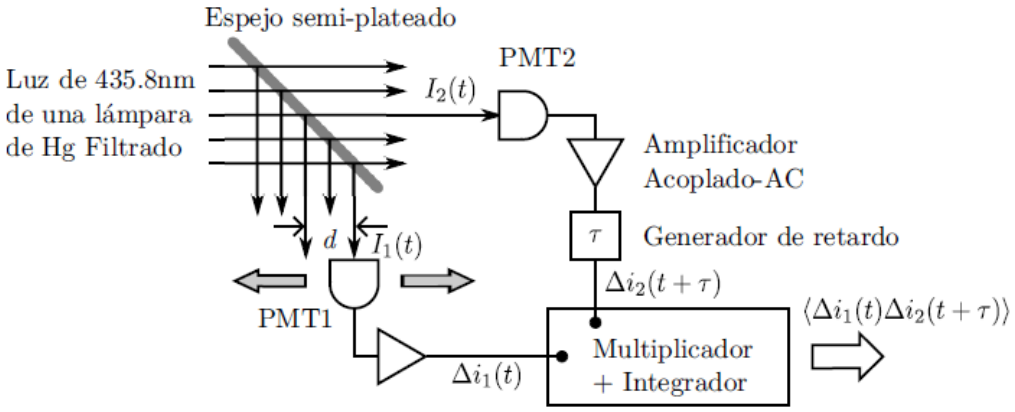

Figura 1.8: Experimento de fluctuación de intensidades de HBT.

un circuito multiplicador-integrador cuya función era multiplicar sus entradas y promediarlas sobre un intervalo de tiempo largo.

Debido a que la fotocorriente producida por cada tubo fotomultiplicador es proporcional a la intensidad de la luz que incide sobre ellos, concluimos que la señal de salida del arreglo total es proporcional a $\langle \Delta I_1(t) \Delta I_2(t+\tau) \rangle$ donde $\langle x \rangle$ indica el promedio temporal de x e $I_1(t)$ y $I_2(t)$ son las intensidades de los haces incidentes sobre los tubos fotomultiplicadores. El resultado principal de este experimento fue observar que la correlación entre las fluctuaciones de intensidad de un haz luminoso está relacionada con su coherencia (22). Si la luz incidente sobre los tubos fotomultiplicadores es coherente entonces las fluctuaciones de intensidad estarán correlacionadas entre sí. De esta manera al medir la correlación entre las fluctuaciones de intensidad podemos obtener información acerca de la coherencia de la luz incidente sin necesidad de realizar experimentos clásicos de interferencia.

Consideremos los resultados que obtendríamos en el arreglo de la figura 1.8 al hacer $d = 0$. Ajustemos el divisor de haz hasta lograr que la intensidad promedio sobre ambos tubos fotomultiplicadores sea la misma. Entonces tendremos que (11, eq.(6.3)):

$$I_1(t) = I_2(t) = \langle I(t) \rangle + \Delta I(t) \tag{1.17}$$

donde $2I(t)$ es la intensidad de la luz incidente sobre el divisor de haz y $\Delta I(t)$ son sus fluctuaciones respecto a su valor promedio.

Si establecemos el tiempo de retardo τ en cero, tendremos que (11, eq.(6.4)):

$$\langle \Delta I(t) \Delta I(t+\tau) \rangle = \langle \Delta I(t)^2 \rangle \tag{1.18}$$

Aunque por definición $\langle \Delta I(t) \rangle = 0$, $\langle \Delta I(t)^2 \rangle \neq 0$ a menos que la fuente de luz no presente variaciones de intensidad. Por otro lado para $\tau \gg \tau_c$ $I(t)$ e $I(t+\tau)$ no estarán correlacionadas temporalmente por lo que su producto cambiará de signo aleatoriamente y su promedio temporal será cero (11, eq.(6.5)):

$$\langle \Delta I(t) \Delta I(t+\tau) \rangle_{\tau \gg \tau_c} = 0 \tag{1.19}$$

Así, este experimento nos permite estimar el tiempo de coherencia de la fuente de luz en función del valor observado de $\langle \Delta I(t) \Delta I(t+\tau) \rangle$.

Originalmente Hanbury Brown y Twiss fijaron $\tau = 0$ y variaron d. De forma análoga al interferómetro de Michelson al aumentar d la coherencia espacial de los haces de luz tenderá a cero igual que $\langle \Delta I(t) \Delta I(t+\tau) \rangle$, de donde podemos observar que efecetivamente este experimento nos brindará información acerca de la coherencia espacial de la fuente.

1.6.4. La función de correlación de segundo orden $g^{(2)}(\tau)$

La función de correlación de segundo orden, denotada como $g^{(2)}(\tau)$, tiene ese nombre debido a que es una manera de cuantificar las variaciones de intensidad que son proporcionales a la segunda potencia del campo eléctrico asociado a la fuente luminosa. Esta función se encuentra definida como (11, eq.(6.6)):

$$g^{(2)}(\tau) = \frac{\langle \vec{E}^*(t) \vec{E}(t) \vec{E}^*(t+\tau) \vec{E}(t+\tau) \rangle}{\langle \vec{E}^*(t) \vec{E}(t) \rangle \langle \vec{E}^*(t+\tau) \vec{E}(t+\tau) \rangle} = \frac{\langle I(t) I(t+\tau) \rangle}{\langle I(t) \rangle \langle I(t+\tau) \rangle} \tag{1.20}$$

donde $\vec{E}$ es el campo eléctrico asociado a la fuente y * denota su complejo conjugado.

Para obtener un poco de intuición acerca de esta función consideremos el valor de $g^{(2)}(\tau)$ para algunos casos simples. Consideremos primero una fuente con una intensidad promedio constante, de tal forma que $\langle I(t) \rangle = \langle I(t+\tau) \rangle$. Así, podemos escribir que (11, eq.(6.7)):

$$I(t) = \langle I \rangle + \Delta I(t) \tag{1.21}$$

donde $\langle \Delta I(t) \rangle = 0$ para todo tiempo.

En este caso particular tenemos que:

$$\begin{aligned}\langle I(t)I(t+\tau)\rangle &= \langle(\langle I\rangle + \Delta I(t))(\langle I\rangle + \Delta I(t+\tau))\rangle \\ &= \langle I\rangle^2 + \langle I\rangle\langle\Delta I(t)\rangle + \langle I\rangle\langle\Delta I(t+\tau)\rangle + \langle\Delta I(t+\tau)\Delta I(t)\rangle \\ &= \langle I\rangle^2 + \langle\Delta I(t+\tau)\Delta I(t)\rangle\end{aligned}$$

Si $\tau \gg \tau_c$ las variaciones de intensidadas no estarán correlacionadas por lo que $\langle\Delta I(t+\tau)\Delta I(t)\rangle = 0$ y en este caso el valor de $g^{(2)}(\tau)$ es (11, eq.(6.9)):

$$g^{(2)}(\tau \gg \tau_c) = \frac{\langle I\rangle^2}{\langle I\rangle^2} = 1 \tag{1.22}$$

Por otro lado si $\tau \ll \tau_c$ si existirá una correlación en las variaciones de intensidad. En particular para $\tau = 0$ tendremos que (11, eq.(6.10)):

$$g^{(2)}(0) = \frac{\langle I(t)^2\rangle}{\langle I(t)\rangle^2} \tag{1.23}$$

En general se puede demostrar (23) que para cualquier forma funcional concebible de $I(t)$ siempre se cumplirá lo siguiente:

$$g^{(2)}(\tau) \geq 1 \tag{1.24}$$

$$g^{(2)}(0) \geq g^{(2)}(\tau) \tag{1.25}$$

Ahora consideremos el caso de una fuente de luz perfectamente monocromática con una intensidad I_0 independiente del tiempo. Es claro que ahora $\langle I(t)I(t+\tau)\rangle = I_0^2 = \langle I_0^2\rangle$, por lo que en este caso tendremos que $g^{(2)}(\tau) = 1$ para todo valor de τ.

Si fuera posible encontrar una fuente de luz tal que $g^{(2)}(\tau) < 1$ para algún valor de τ estaríamos observando un fenómeno cuya explicación no puede estar basada en el enfoque clásico de la luz. De hecho, en (24) se presenta un experimento relativamente sencillo que puede realizarse en un laboratorio de licenciatura que permite observar una fuente luminosa para la cual $g^{(2)}(\tau) < 1$. Este experimento suele llevarse a cabo de forma usual en el Laboratorio de Óptica Avanzada de la Facultad de Ciencias, UNAM.

1.6.5. Experimento de HBT con fotones

Reconsideremos ahora el experimento de HBT tomando en cuenta el comportamiento cuántico de la luz. Ahora un flujo de fotones incide sobre el divisor de haz, donde este flujo se divide de formas iguales en ambas salidas, para después ser detectado por módulos detectores de fotones individuales. Los pulsos generados por estos módulos son alimentados a un módulo contador/temporizador que se encarga de contar individualmente el número de pulsos recibidos en cada entrada y de contar el tiempo transcurrido entre los pulsos recibidos en el canal 1 y el canal 2. El esquema experimental se muestra en la parte a) de la figura 1.9.

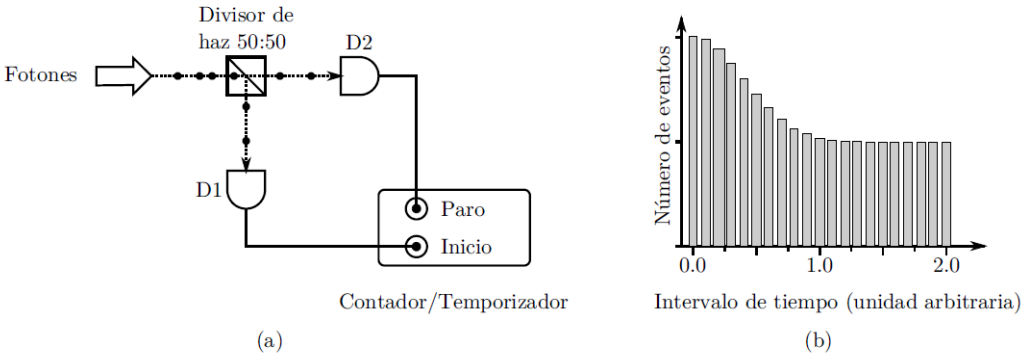

Figura 1.9: Experimento de HBT con fotones.

Los resultados de este experimento suelen mostrarse en un histograma de frecuencias, donde el eje horizontal indica el tiempo τ transcurrido entre la detección de un pulso en una entrada de inicio y de un pulso subsecuente en la entrada de paro, y el eje vertical muestra el número de eventos observados. Una muestra de éstos histogramas se muestra en la parte b) de la figura 1.9.

En la subsección anterior se definió $g^{(2)}(\tau)$ en términos de la intensidad de la fuente de luz. Debido a que el número de cuentas observadas en cada entrada es directamente proporcional a la intensidad de la luz presente podemos reescribir a $g^{(2)}(\tau)$ como (11, eq.(6.16)):

$$g^{(2)}(\tau) = \frac{\langle n_1(t) n_2(t+\tau) \rangle}{\langle n_1(t) \rangle \langle n_2(t+\tau) \rangle} \tag{1.26}$$

donde $n_i(t)$ es el número de cuentas registradas por el detector i-ésimo en el tiempo t.

La ecuación 1.26 muestra que el valor de $g^{(2)}(\tau)$ depende de la probabilidad simultánea de contar fotones en el detector 1 al tiempo t y en el tiempo $t+\tau$ en el detector 2. En otras palabras $g^{(2)}(\tau)$ es proporcional a la probabilidad condicional

de encontrar un foton en el detector 2 al tiempo $t = \tau$ dado que se encontró un fotón en el detector 1 al tiempo $t = 0$. De hecho, el histograma de la figura 1.9 b) es proporcional a esta probabilidad.

Un experimento mental simple es capaz de mostrarnos la diferencia en el comportamiento de $g^{(2)}(\tau)$ al considerar el enfoque de fotones en lugar del enfoque clásico de la luz. Supongamos ahora que la luz se encuentra compuesta de fotones y que la separación temporal entre la detección de éstos es mayor a la resolución del aparato de medición utilizado, de forma tal que es posible distinguir la llegada de cada uno de ellos de manera individual. Cuando el primer fotón llega al divisor por simetría este tiene la misma probabilidad de ser detectado en el canal 1 o en el canal 2. Si el fotón es detectado en el canal 1 entonces iniciara el temporizador del módulo y por lo tanto la probabilidad de detectar un fotón en en el canal 2 en $\tau = 0$ será nula, por lo que el temporizador no registrará ningún evento para este valor de τ.

Cuando el fotón siguiente incida sobre el divisor este puede ser detectado en 1 o en 2 con la misma probabilidad. Si el fotón es detectado en 2 entonces el temporizador será detenido y se contará un evento exitoso para el valor correspondiente de τ. En caso contrario, el fotón será detectado en 1 y deberemos esperar hasta que incida otro fotón sobre el divisor y sea detectado en 2 para detener el temporizador. Así, tenemos una situación en donde no esperamos ningún evento para $\tau = 0$ pero sí esperamos eventos para tiempos posteriores de τ, que claramente contradice los resultados clásicos esperados de las ecuaciones 1.24 y 1.25.

Consideremos ahora el caso en el que los fotones llegan al divisor por racimos. Aproximadamente la mitad de ellos serán detectados en 1 y la otra mitad en 2. Así, la probabilidad de observar un fotón en 2 dado de que un fotón fue observado en 1 será alta para valores $\tau \approx 0$ mientras que esta probabilidad tenderá a cero para valores mayores de τ. En este caso tenemos muchos eventos cercanos a $\tau = 0$ y un número cada vez menor de eventos para tiempos posteriores, lo que es consistente con los resultados clásicos de las ecuaciones 1.24 y 1.25.

Como conclusión, en el enfoque cuántico de la luz lo que importa es la separación temporal entre los fotones que componen al haz, y esta propiedad nos permite clasificar a las fuentes de luz en términos de su valor de $g^{(2)}(0)$.

1.6.6. Clasificación de la luz en términos de $g^{(2)}(\tau)$

Cualquier fuente luminosa puede ser clasificada en términos de el valor de su función de correlación de segundo orden dentro de alguno de los 3 grupos siguientes:

- Luz agrupada: $g^{(2)}(0) > 1$

- Luz coherente: $g^{(2)}(0) = 1$
- Luz antiagrupada: $g^{(2)}(0) < 1$

La luz agrupada y la luz coherente son compatibles con el esquema clásico de la luz, mientras que la luz antiagrupada no tiene ninguna contraparte clásica sino que es un fenómeno puramente cuántico. Así, la observación de este tipo de luz es una prueba conclusiva en favor del esquema de fotones, y en efecto este tipo de luz ha podido observarse experimentalmente (25).

Figura 1.10: Clasificación de la luz en términos de $g^{(2)}(0)$.

En la figura 1.10 se muestra conceptualmente la diferencia entre estos 3 tipos de luz. La luz coherente representa el caso en que los fotones se encuentran distribuidos de manera aleatoria dentro del haz, mientras que para la luz agrupada estos se encuentran agrupados en racimos. Por último para la luz anti-agrupada los fotones están separados regularmente dentro del haz.

Recordemos que el valor de $g^{(2)}(\tau)$ es proporcional a la probabilidad conjunta de detectar un fotón en $t = \tau$ dado que se detectó un fotón en $t = 0$. Así, para la luz agrupada esperamos que $g^{(2)}(\tau)$ sea mayor para valores muy chicos de τ debido a que los fotones se encuentran agrupados en racimos, mientras que para la luz antiagrupada esperamos que $g^{(2)}(\tau)$ sea muy chica para valores pequeños de τ debido a que los fotones se encuentran separados regularmente dentro del haz.

La relación entre las variaciones clásicas de intensidad de una fuente luminosa caótica y el esquema cuántico de fotones agrupados se muestra en la figura 1.11. Debido a que el número de fotones es proporcional a la intensidad instantánea los racimos de fotones están relacionados con aquellas regiones donde la intensidad instantánea es mayor que la intensidad promedio, mientras que las regiones entre racimos están relacionadas con las regiones cuya intensidad instantánea es menor que el promedio.

Finalmente en la tabla 1.1 se muestra una comparación entre el esquema clásico y cuántico de la luz junto con los valores correspondientes de $g^{(2)}(0)$ a manera de resumen.

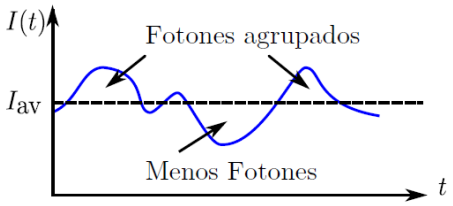

Figura 1.11: Relación entre las variaciones clásicas de intensidad y el agrupamiento de fotones.

Contraparte Clásica	**Distribución de Fotones**	$g^{(2)}(0)$
Luz Caótica	Agrupados	>1
Luz Coherente	Aleatoria	= 1
Ninguna	Anti-agrupados	<1

Tabla 1.1: Resumen de la clasificación de la luz en términos de $g^{(2)}(0)$

1.6.7. Medición de $g^{(2)}(0)$ con 2 y 3 detectores

A continuación se muestran dos expresiones distintas para determinar experimentalmente el valor de $g^{(2)}(0)$ mediante los experimentos descritos en (24), donde se pueden consultar el material y los diagramas experimentales correspondientes. La diferencia entre estas expresiones consiste en el número de detectores utilizados para poder medir las cantidades de interés.

El valor de la expresión obtenida al utilizar dos detectores se denotará como $g^{(2)}_{2D}(0)$, y se le conoce como una medición incondicional de $g^{(2)}(0)$. Para el caso de 3 detectores la medición se denotará como $g^{(2)}_{3D}(0)$, y a esta expresión se le conoce como una medición condicional del valor de $g^{(2)}(0)$ debido a la presencia del detector de validación adicional.

La demostración de ambas expresiones puede encontrarse en (26).

1.6.7.1. Medición de $g^{(2)}(0)$ con 2 detectores

La expresión de $g^{(2)}_{2D}(0)$ se encuentra expresada en términos de probabilidades de observar ciertos eventos de detección, que a su vez pueden estimarse experimentalmente mediante el uso de un contador de fotones.

$$g^{(2)}_{2D}(0) = \frac{P_{TR}}{P_T P_R} \tag{1.27}$$

Donde:

- P_{TR} es la probabilidad de detectar fotones de manera simultánea en los detectores T y R en un mismo intervalo de tiempo fijo Δt conocido como la *ventana de coincidencia*.
- P_T es la probabilidad de detectar fotones en el detector T en el mismo intervalo de tiempo Δt.
- P_R es la probabilidad de detectar fotones en el detector R en el mismo intervalo de tiempo Δt.

Estas probabilidades pueden ser estimadas con la ayuda de un contador de fotones de la manera siguiente. Sea R_R la tasa de detecciones registradas en el detector R por unidad de tiempo, R_T la tasa de detecciones registradas en el detector T por unidad de tiempo y sea R_{TR} la tasa de detecciones simultáneas registradas por ambos detectores por unidad de tiempo. Así:

$$P_T = R_T \Delta t \tag{1.28}$$
$$P_R = R_R \Delta t \tag{1.29}$$
$$P_{TR} = R_{TR} \Delta t \tag{1.30}$$

Sea ΔT el intervalo de tiempo durante el cual los detectores se encuentran en funcionamiento, al cual nos referiremos en este documento y en el software como el *tiempo de prueba*. El contador de fotones nos permite encontrar el número de detecciones observadas por cada detector dentro de este intervalo de tiempo, por lo que podremos estimar el valor de R_R, R_T y R_{TR} como $R_T = N_T/\Delta T$, $R_R = N_R/\Delta T$ y $R_{TR} = N_{TR}/\Delta T$. Así:

$$P_T = \frac{N_T \Delta t}{\Delta T} \tag{1.31}$$
$$P_R = \frac{N_R \Delta t}{\Delta T} \tag{1.32}$$
$$P_{TR} = \frac{N_{TR} \Delta t}{\Delta T} \tag{1.33}$$

Estas ecuaciones serán validas siempre y cuando todas las probabilidades sean menores que 1. Sustituyendo en la ecuación 1.27 tenemos finalmente que:

$$g_{2D}^{(2)}(0) = \frac{N_{TR}}{N_T N_R}\left(\frac{\Delta T}{\Delta t}\right) \tag{1.34}$$

Donde:

- ΔT: Tiempo de prueba
- Δt: Ventana de coincidencia
- N_T: Número de detecciones observadas en el detector T en el intervalo de tiempo ΔT
- N_R: Número de detecciones observadas en el detector R en el intervalo de tiempo ΔT
- N_{TR}: Número de detecciones simultáneas observadas en ambos detectores en el intervalo de tiempo ΔT

1.6.7.2. Medición de $g^{(2)}(0)$ con 3 detectores

En esta medición se utiliza un detector adicional, denotado como G, que es utilizado para validar las fotodetecciones. En este caso la expresión utilizada para calcular $g^{(2)}(0)$ es :

$$g_{3D}^{(2)}(0) = \frac{P_{GTR}}{P_{GT} P_{GR}} \tag{1.35}$$

Donde:

- P_{GT} es la probabilidad de realizar una detección simultánea por los detectores G y T dentro de la ventana de coincidencia Δt.
- P_{GR} es la probabilidad de realizar una detección simultánea por los detectores G y R dentro de la ventana de coincidencia Δt.
- P_{GTR} es la probabilidad de realizar una detección simultánea triple por los 3 detectores G, T Y R dentro de la ventana de coincidencia Δt.

La presencia del detector adicional G nos permite normalizar éstas probabilidades de un modo distinto al caso de dos detectores, por lo que ahora tendremos que:

$$P_{GT} = \frac{N_{GT}}{N_G} \tag{1.36}$$

$$P_{GR} = \frac{N_{GR}}{N_G} \tag{1.37}$$

$$P_{GTR} = \frac{N_{GTR}}{N_G} \tag{1.38}$$

Donde:

- N_G es el número de detecciones registradas por el detector G durante el tiempo de prueba.
- N_{GT} es el número de coincidencias registradas entre los detectores G y T durante el tiempo de prueba.
- N_{GR} es el número de coincidencias registradas entre los detectores G y R durante el tiempo de prueba.
- N_{GTR} es el número de coincidencias triples registradas entre los detectores G, T y R durante el tiempo de prueba.

Sustituyendo estas expresiones en la ecuación 1.35 tenemos finalmente que:

$$g^{(2)}_{3D}(0) = \frac{N_{GTR}N_G}{N_{GT}N_{GR}} \tag{1.39}$$

Cabe aclarar que aunque el tiempo de prueba y la ventana de coincidencia no aparezcan explícitamente en la ecuación 1.39 estos parámetros son utilizados para definir el tiempo dentro del que diremos que dos fotones son coincidentes y para determinar el tiempo durante el cual realizaremos la medición de las cuentas y coincidencias detectadas.

1.7. Conclusión

Como hemos visto el conteo de fotones individuales y la detección de coincidencias en el arribo de las señales correspondientes son problemas prácticos de gran interés para la óptica cuántica. Para desarrollar un dispositivo de instrumentación que sea capaz de realizar estas tareas es necesario diseñar la electrónica del dispositivo (Capítulo 2) de tal forma que sea compatible con los APDs y los

dispositivos lógicos programables utilizados, programarlos para que lleven a cabo las tareas deseadas (Capítulo 3), y establecer una comunicación fluida entre el instrumento y el usuario que permita configurar los parámetros de interés del experimento, realizar la extracción de datos y recibir instrucciones por parte del usuario, entre otras cosas. Por cuestiones de tiempo el instrumento se diseñó para ser configurado con la ayuda de una computadora (Capítulo 4), dejando la arquitectura lista para que en el futuro también se pueda establecer la comunicación a través de la forma estándar utilizando botones, teclados y pantallas.

En los capítulos siguientes se mostrará la solución propuesta a detalle, empezando por el firmware que fue programado en un dispositivo lógico conocido como FPGA y en un microcontrolador, el hardware que consiste de una tarjeta de circuito impreso con la electrónica necesaria para acoplar los APDs y los dispositivos lógicos que contienen el firmware, y el software que fue programado en el lenguaje de programación Python para permitirle al usuario configurar de manera intuitiva el dispositivo de instrumentación y realizar un análisis básico de los datos obtenidos.

Capítulo 2

Diseño del Hardware

En este capítulo se describe el hardware utilizado para construir el dispositivo de instrumentación, el cual consiste de un FPGA (*Field Programmable Gate Array*) Cyclone ll de la compañía Altera, una tarjeta de desarrollo Teensy 3.2 y conectores tipo EPK.00.250.NTN de la compañía LEMO en los que se reciben las señales de entrada procedentes de los APDs, así como de una tarjeta de circuito impreso (PCB) diseñada para unir todos estos componentes en un sistema integral de instrumentación.

El FPGA se encuentra programado con el firmware descrito en el capítulo 3 y es el encargado de realizar las tareas relacionadas directamente con el conteo y detección de coincidencias, mientras que la tarjeta Teensy se encarga de establecer la comunicación serial entre el FPGA y la computadora para habilitar el envío de órdenes y la extracción de datos mediante el protocolo UART.

Un diagrama de bloques del sistema desarrollado se muestra en la figura 2.1.

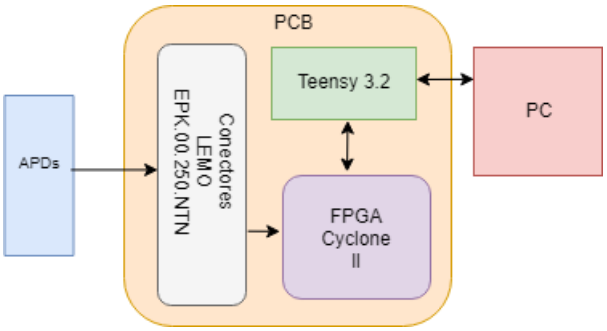

Figura 2.1: Diagrama de bloques del hardware.

2.1. Microcontroladores

Los microcontroladores son dispositivos electrónicos programables que consisten de una unidad de procesamiento, una unidad de memoria y puertos de entrada y salida que le permiten interactuar con el exterior a través de las señales eléctricas enviadas o recibidas por medio de sus terminales. Básicamente uno puede pensar en un microcontrolador como una computadora con dimensiones pequeñas que uno puede programar a voluntad. Un ejemplo de estos dispositivos se muestra en la figura 2.2.

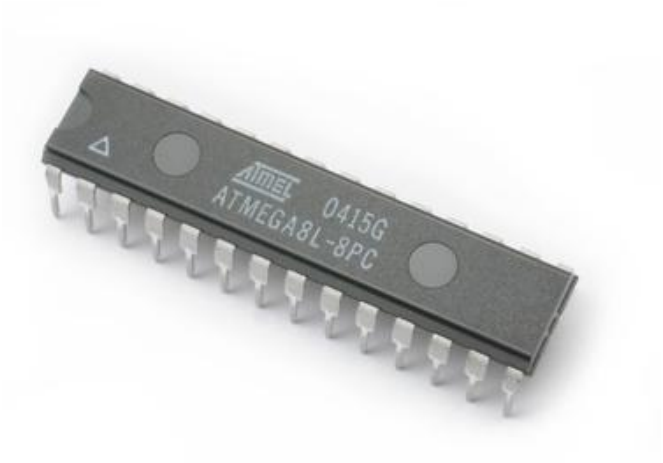

Figura 2.2: Microcontrolador ATMEGA8L de la compañia ATMEL (Ahora Microchip).

Estos dispositivos suelen ser muy utilizados en la industria debido a que tienen diversas ventajas como un bajo costo, una alta flexibilidad, y un bajo consumo de energía. Algunas de sus aplicaciones son:

- Sistemas de ignición para automóviles.
- Controles remotos.
- Juguetes.
- Dispositivos médicos.
- Celulares.

En este proyecto se utilizó una tarjeta de desarrollo Teensy 3.2 (27), la cual emplea un microcontrolador MK20DX256VLH7 Cortex-M4 de la compañía NXP, cuya única tarea actualmente es establecer la comunicación serial entre el FPGA

y la computadora. Debido a que en el futuro se planea agregar nuevas funciones como comunicación inalámbrica y dispositivos adicionales al sistema que permitan configurarlo sin la necesidad de utilizar una computadora, se decidió utilizar un microcontrolador en lugar de un circuito integrado que únicamente sirva para este propósito, ya que se tendría que modificar el diseño del hardware para agregar estas nuevas funcionalidades.

Debido a la gran variedad de proyectos en los que suelen ser utilizados existen diversos modelos producidos por distintas empresas, como Atmel (ahora Microchip), Texas Instruments, Motorola e Intel, entre otras.

Algunos de los principales criterios utilizados para seleccionar entre la gran variedad de microcontroladores disponibles son:

- Velocidad de reloj.
- Cantidad de memoria.
- Dimensiones físicas.
- Número de terminales disponibles.
- Costo.
- Bajo consumo energético.

Una de las mayores desventajas de los microcontroladores es que no pueden realizar procesamiento en paralelo de forma natural, por lo que es necesario agregar otro dispositivo adicional al sistema para realizar las tareas de conteo y detección de coincidencias. Más acerca de esto se discute en la sección 2.2.

En el mercado existen tarjetas de desarrollo que integran un microcontrolador junto con circuitos y dispositivos adicionales para facilitar el diseño, prototipado y depuración de nuevos proyectos, como:

- Circuitos adicionales que permiten programar el microcontrolador facilmente a través de un cable USB.
- Módulos de comunicación inalámbrica Wifi/Bluetooth integrados.
- Ranuras para introducir tarjetas de memoria adicionales como SD y microSD.
- Pantallas (Displays) y led indicadores completamente programables.

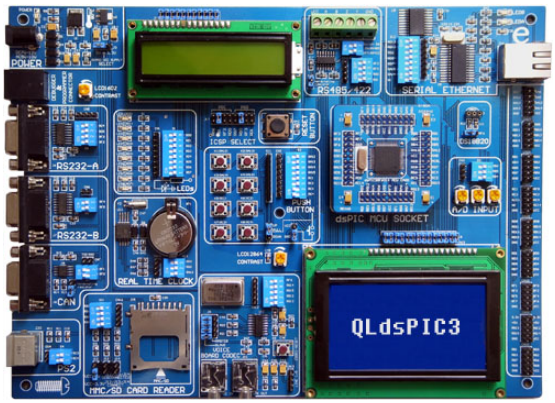

Figura 2.3: Tarjeta de desarrollo QLdsPIC3 para microcontroladores PIC de 16 bits.

Ejemplos de este tipo de tarjetas son las producidas por la compañia Arduino (28), que utiliza microcontroladores de la marca Microchip, las tarjetas Launch-Pad (29) producidas por Texas Instruments que utilizan microcontroladores de la misma compañia y la familia de tarjetas Teensy (30) que utilizan microcontroladores de la compañía NXP.

En este proyecto se eligió la tarjeta de desarrollo Teensy 3.2 debido a que sus dimensiones, la cantidad de terminales disponibles, la frecuencia de su reloj interno de 72 MHz, la cantidad de conversores ADC de 13 bits, la presencia de una salida analógica de 12 bits y la variedad de estándares de comunicación que permite implementar como ISP, I2C y UART, lo convierten en un dispositivo sumamente potente y flexible para permitirnos implementar en el futuro nuevas funcionalidades sin tener que modificar demasiado la arquitectura interna de la tarjeta, y todo esto en un dispositivo cuyas dimensiones son menores que 4cm x 2cm. Las especificaciones técnicas completas de esta tarjeta pueden consultarse en (27).

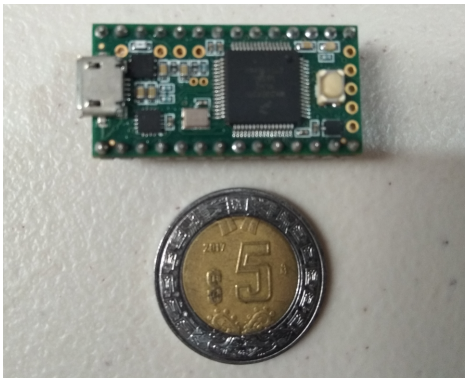

Figura 2.4: Tarjeta de desarrollo Teensy 3.2.

2.2. FPGAs

Los FPGAs son dispositivos electrónicos programables que consisten de varios bloques lógicos cuya funcionalidad e interconexiones puede ser configurada por el usuario en lenguajes especializados conocidos como lenguajes de descripción de hardware (HDL - Hardware Desciption Language). Las tareas que pueden llevar a cabo pueden ser tan sencillas como implementar una compuerta lógica o utilizarse para desarrollar tareas mas complicadas como implementar complejos sistemas combinacionales, ya que en principio un FPGA es capaz de implementar cualquier circuito lógico deseado siempre y cuando éste contenga una cantidad suficiente de bloques lógicos.

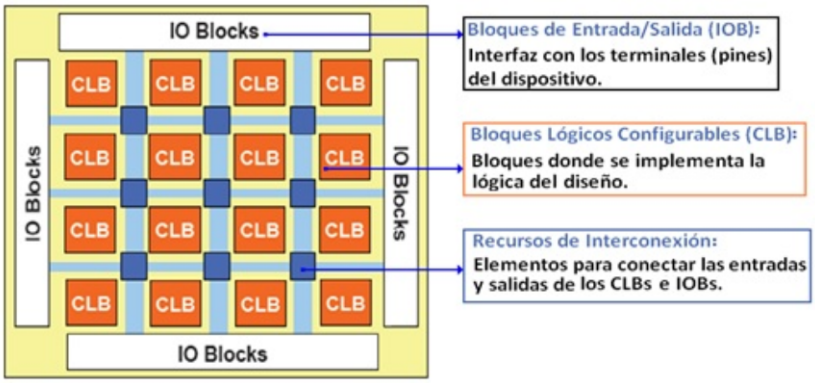

Figura 2.5: Esquema conceptual de la estructura interna de un FPGA.

Estos dispositivos suelen ser utilizados ampliamente en la industria, y algunas de sus principales ventajas respecto a un microcontrolador son:

- Procesamiento en paralelo: su arquitectura interna hace posible realizar tareas que requieran de procesamiento en paralelo de forma natural, por lo que en tareas que requieran grandes cantidades de procesamiento que deba realizarse en el menor tiempo posible suelen ser una de las opciones mas naturales.
- Control del diseño de hardware y flexibilidad: al programar un microcontrolador únicamente se modifica el software que va a ejecutar su unidad de procesamiento pero la arquitectura interna de éste siempre se mantiene fija, mientras que en un FPGA se configuran directamente las conexiones físicas de los bloques lógicos internos.
- Velocidad: los FPGAs son programados directamente a nivel lógico y su funcionamiento únicamente depende de la forma en que el usuario haya

programado las conexiones internas de los bloques lógicos, mientras que en un microcontrolador su unidad de procesamiento es la encargada de administrar el órden en que se realizan las tareas ejecutadas por el software.

Figura 2.6: FPGA Cylone ll de la compañia Altera.

Por estas razones, los FPGAs suelen ser utilizados en aplicaciones que involucren realizar grandes cantidades de procesamiento en tiempo real, sistemas de procesamiento de señales, análisis de imágenes y procesamiento en paralelo, entre otras.

En este proyecto se utilizó un FPGA debido a que los fotones incidentes sobre cada canal pueden llegar de manera independiente, por lo que existe la posibilidad de que los fotones lleguen separados por intervalos de tiempo arbitrariamente pequeños. Así, si utilizaramos un microcontrolador para monitorear de manera secuencial las entradas para observar si no existe algún fotón incidente existirían fotones que no serían detectados por el sistema, ya que como no es posible realizar procesamiento en paralelo en los microcontroladores no se pueden monitorear de manera simultánea todas las entradas y solo seríamos capaces de monitorear un solo canal a la vez.

En el mercado existen diversos fabricantes de FPGAs como Altera, Xilinx y Lattice Semiconductors, entre otras. Análogamente que con los microcontroladores existen diversas tarjetas de desarrollo que contienen dispositivos y circuitos adicionales para facilitar el desarrollo y prueba de nuevos proyectos, siendo uno de sus principales beneficios que le facilitan a nuevos usuarios la tarea de aprender a programarlos y explorar muchas de sus capacidades sin necesidad de armar circuitos adicionales.

Sin embargo, el uso de estas tarjetas en algún proyecto que se planee llevar a la etapa de producción tiene varias desventajas en comparación con la alternativa de utilizar únicamente un FPGA individual e incluir manualmente los circuitos adicionales que van a ser utilizados en el proyecto:

- La cantidad de dispositivos adicionales que se encuentran conectados al FPGA en estas tarjetas reducen el numero de terminales disponibles para ser utilizadas por el usuario.

- La adición de estos dispositivos incrementa el costo del proyecto de manera innecesaria, ya que es muy probable que la mayoría de éstos no sean necesarios en el proyecto de interés.

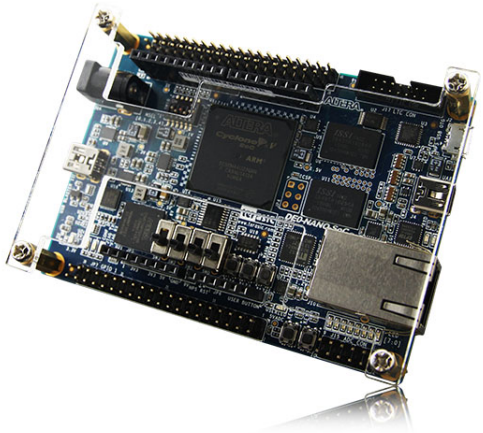

Figura 2.7: Tarjeta de desarrollo Atlas Soc diseñada por Terasic Inc.

Algunos ejemplos son las tarjetas Mojo (31), basadas en FPGAs de la familia Spartan que son producidos por Xilinx, o las distintas familias producidas por Altera (32) como Stratix, Arria, Cyclone, MAX y Soc.

Elegimos el FPGA Cyclone ll debido a que es uno de los dispositivos de menor costo cuyo número de terminales, arquitectura interna y número de unidades lógicas son suficientes para permitirnos implementar el firmware desarrollado e implementar mejoras y nuevas funcionalidades en el futuro sin tener que sustituirlo por una alternativa mas potente. Las especificaciones técnicas de este FPGA pueden consultarse en (33).

La tarjeta de desarrollo Atlas Soc (34) mostrada en la figura 2.7 fue utilizada constantemente en este proyecto para desarrollar y probar rápidamente distintas partes del firmware descrito en el capítulo 3.

2.3. Tarjeta con puertos de entrada

Como se muestra en la figura 2.1 el propósito de esta tarjeta es integrar los distintos dispositivos y componentes que componen al hardware. El diseño de la tarjeta de circuito impresa (PCB) se muestra en la figura 2.8. El diagrama eléctrico correspondiente se muestra en la figura 2.12, que se encuentra al final del capítulo.

Los circuitos y dispositivos que fueron soldados a la tarjeta son:

- 4 conectores header de 14x2 para conectar el FPGA.
- 2 conectores header de 14x1 para conectar el Teensy 3.2.
- Arreglo de LEDS y transistores NPN para monitorear el proceso de transmisión de datos.
- Conectores LEMO tipo EPK.00.250.NTN para conectar los APDs.

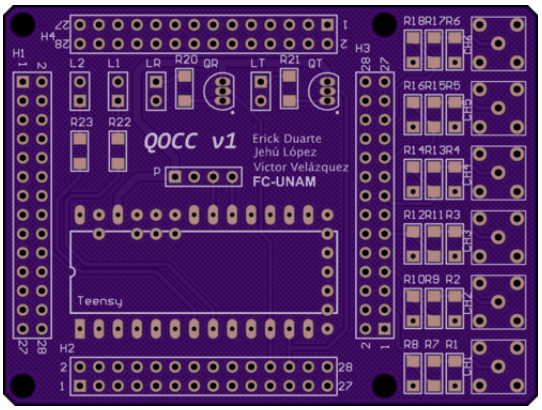

Figura 2.8: Diseño de la tarjeta PCB desarrollada.

La tarjeta con todos los componentes soldados se muestra en la figura 2.9.

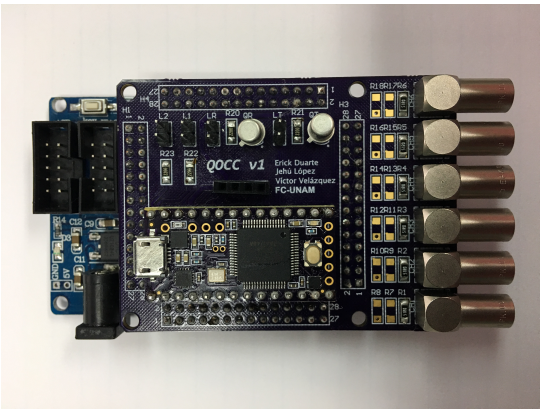

Figura 2.9: Tarjeta PCB con componentes soldados.

Los puertos de entrada de la tarjeta consisten en conectores LEMO tipo EPK.00.250.NTN y un arreglo de resistores de 50 ohms para acoplar la impedancia de salida de los módulos detectores SPCM-AQ-4C con la impedancia de entrada del FPGA. La hoja de datos de estos conectores puede consultarse en (35).

Figura 2.10: Conector Lemo tipo EPK.00.250.NTN.

Debido a que estos módulos proporcionan una señal de salida con una amplitud de 2.5V y un ancho de 35 ns no fue necesario incluir un circuito adicional para modificar su amplitud, ya que las terminales del FPGA Cyclone ll son capaces de tolerar estos voltajes de entrada directamente.

El circuito de LEDS utilizado para monitorear el proceso de transmisión de datos entre la computadora y el Teensy consiste en un conjunto de transistores NPN que son activados con los pulsos de las terminales TX y RX del microcontrolador.

2.4. Diseño final con carcasa

Se diseñó una carcasa para proteger a la tarjeta desarrollada del exterior y darle un acabado más profesional con ayuda del software Fusion 360 (Autodesk), misma que se fabricó con ayuda de una impresora 3D. El resultado final se muestra en la figura 2.11.

Figura 2.11: Vista final de la tarjeta desarrollada dentro de su carcasa.

2.5. Conclusión

En este capítulo se describieron algunas de las ventajas y desventajas de los microcontroladores y los FPGAs, así como la razón por la que es necesario incluir ambos dispositivos en este diseño.

A manera de resumen, principalmente se utiliza un FPGA debido a la capacidad que posee de ejecutar distintas rutinas de código de forma paralela, mientras que el microcontrolador es capaz de permitir extender fácilmente el dispositivo desarrollado con nuevos módulos y funcionalidades como comunicación inalámbrica, permitir agregar pantallas y dispositivos de entrada que permitan controlar al dispositivo y almacenar los archivos de datos directamente en una memoria USB sin la necesidad de contar con una computadora, y muchas otras más. Aunque en principio es posible realizar todas esta tareas programando adecuadamente al FPGA, la gran cantidad de librarías existentes en el ecosistema Arduino permite realizarlas de una manera mucho más sencilla sin necesidad de invertir más tiempo en desarrollar el código correspondiente.

La impresión 3D facilitó considerablemente la elaboración de una carcasa hecha a la medida para la tarjeta, lo que permitió dotar al dispositivo de una apariencia mas profesional y hacerla mas resistente ante golpes, polvo y caídas.

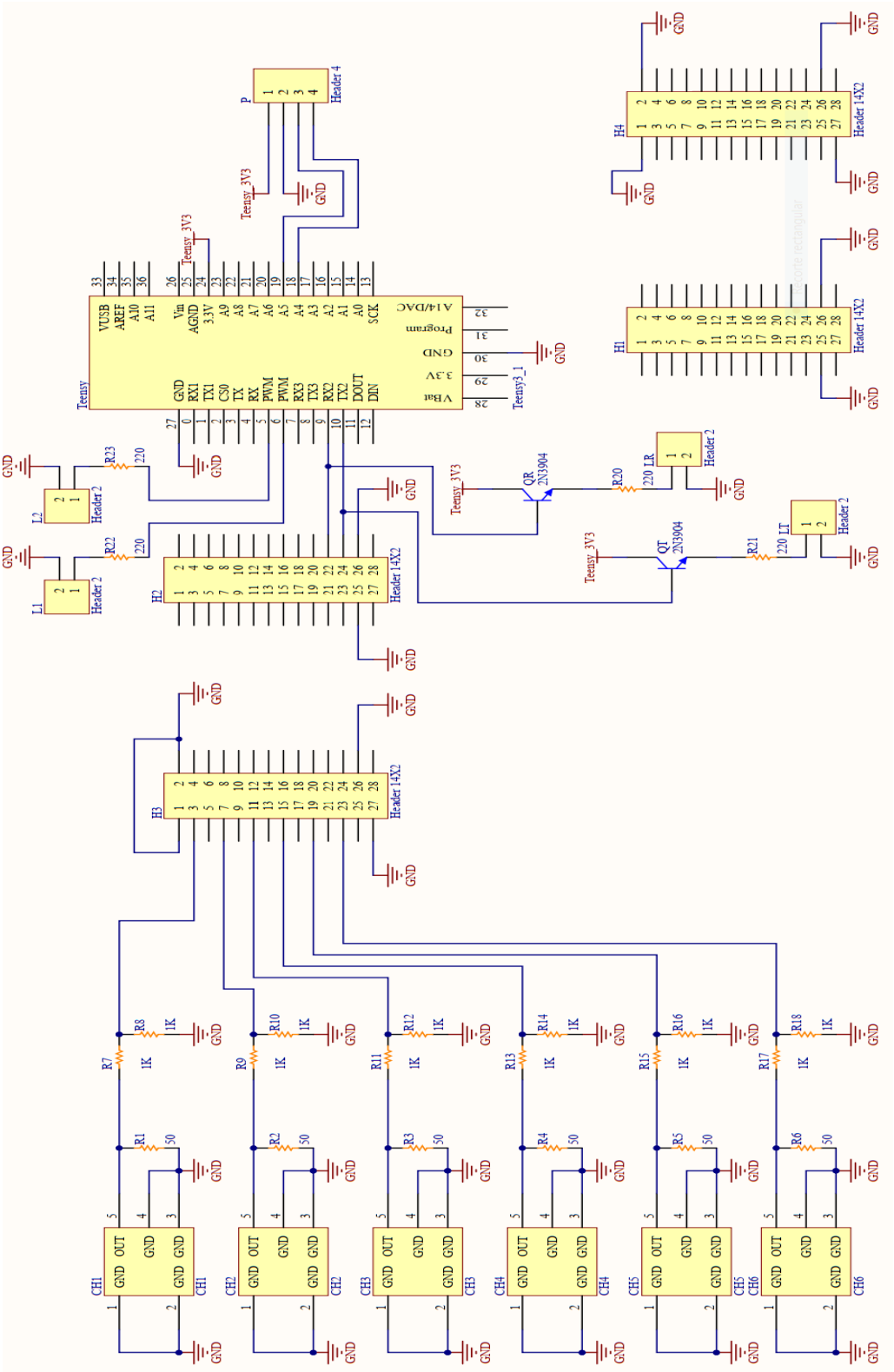

Figura 2.12: Diagrama eléctrico de la tarjeta desarrollada.

Capítulo 3

Diseño del Firmware

El firmware consiste en el conjunto de instrucciones programadas directamente en el FPGA que le indican qué realizar con las señales eléctricas presentes en sus entradas, cómo procesarlas, y cómo comunicarse con el exterior a través de señales eléctricas que producirá en sus terminales de salida. El firmware está compuesto por varios módulos encargados de realizar distintas tareas como establecer la comunicación serial con la computadora a través del microcontrolador o realizar la detección de coincidencias, entre otras. Estos módulos se comunican entre sí para intercambiar información acerca de las tareas que cada uno debe realizar y con el exterior a través de las terminales del FPGA.

En este capítulo se describen cada uno de los módulos que componen el firmware programado en el FPGA: se describen sus entradas, salidas, interconexiones con otros módulos, su funcionamiento y el papel que juega cada uno en el sistema completo de instrumentación. El diagrama por bloques completo del firmware desarrollado se muestra en la figura 3.18, que se encuentra al final del capítulo.

3.1. Introducción

En esta sección se describe la notación y términos que serán utilizados en las secciones siguientes para describir el firmware y su funcionamiento.

El firmware se encuentra compuesto por varios módulos, que se encuentran conectados mediante distintos tipos de conexiones. Cuando tratamos a dos o mas líneas de conexión relacionadas como un solo objeto nos referimos a ellas como un *bus*, y a el número de líneas que lo componen como su tamaño o número de bits.

Si X denota a el valor lógico conjunto de todas las líneas que componen a un bus en un instante de tiempo, utilizaremos la notación X[i] para denotar al valor

de la i-ésima línea y X[j:i] para denotar a el valor conjunto de las líneas i, i+1, . . . , j donde por convención se comienza a enumerar a los bits que componen al bus de derecha a izquierda e iniciando desde el valor 1. Por ejemplo si X = 00110001 entonces X[1] = 1, X[8] = 0 y X[6:4] = 110.

Adicionalmente decimos que el bit que se encuentra más a la izquierda en un bus es su bit más significativo y el que el bit que se encuentra más a la derecha es su bit menos significativo, y suelen utilizarse expresiones como 'los 3 bits más significativos' o 'los 5 bits menos significativos' para referirse a los 3 bits que se encuentran más a la izquierda o a los 5 bits que se encuentran mas a la derecha en un bus, respectivamente.

Por último, existen dos señales que tienen una importancia particular debido a que son utilizadas muy frecuentemente, conocidas como pulso positivo y pulso negativo, cuyas gráficas se muestran en la figura 3.1.

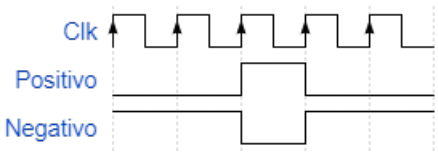

Figura 3.1: Pulso positivo y pulso negativo.

3.2. Generador de reloj - PLL

El propósito de este módulo es generar una señal de reloj de 200 MHz a partir de la señal proporcionada por el reloj interno de 50 MHz. El código de este módulo no fue programado directamente ya que forma parte de un conjunto de módulos diseñados y programados por Altera (36) que son puestos a disposición del programador al adquirir sus productos, con el propósito de minimizar el uso de los recursos disponibles del FPGA y de realizar la tarea correspondiente de la manera más eficiente posible.

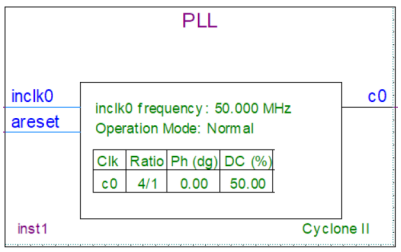

Figura 3.2: Diagrama de bloques del módulo PLL.

El diagrama de bloques de este módulo se muestra en la figura 3.2 y sus terminales y conexiones correspondientes se describen en la tabla 3.1.

Nombre	Tipo	Conexiones
inclk0	Entrada	Reloj interno de 50 MHz
areset	Entrada	*RESETN* - Terminal física
c0	Salida	Señal de reloj para varios módulos

Tabla 3.1: Terminales y conexiones del módulo PLL

3.3. Comunicación

En esta sección se describen las bases del protocolo de comunicación UART (UART, *Universal Asynchronous Receiver-Transmitter*) y su implementación, programada en el lenguaje de descripción de Hardware Verilog, para establecer la comunicación serial asíncrona entre el FPGA y el microcontrolador. Este conjunto de módulos hacen posible la recepción de instrucciones enviadas por el microcontrolador y el envío de los datos de conteos y coincidencias almacenados en el FPGA.

3.3.1. Protocolo UART

En general existen dos métodos para transmitir información entre dos dispositivos, conocidos como *comunicación serial* y *comunicación en paralelo* (37). La comunicación serial consiste en que la información que se desea transmitir es enviada un bit a la vez, uno detrás de otro, mediante la misma línea de comunicación. En contraste, en la comunicación en paralelo los dispositivos comparten mas de una línea de comunicación y es posible transmitir información de forma simultánea en todas las líneas.

El UART, más que un protocolo de comunicación, suele ser considerado como el dispositivo de hardware utilizado para establecer la comunicación serial asíncrona entre dos dispositivos. Sin embargo como en este caso su implementación fue programada en Verilog nos referiremos a él como un protocolo de comunicación.

Al ser un protocolo de comunicación asíncrono el emisor y el receptor no se encuentran sincronizados en el envío y recepción de mensajes, es decir, el emisor puede enviar un mensaje en cualquier instante sin necesidad de que el receptor

esté preparado para recibirlo. Para solucionar este incoveniente el emisor debe de informarle antes al receptor que está a punto de enviar un mensaje para que éste se prepare para recibirlo exitosamente, ya que cuando no se está transmitiendo información la línea de datos se mantiene siempre en alto (1 lógico) por defecto. Para esto, por cada byte de datos que el emisor desee enviar debe de agregar dos bits adicionales al mensaje, que sirven para indicarle al receptor el inicio y el final del mismo. Estos bits adicionales suelen ser conocidos como *bit de inicio* y *bit de paro*, respectivamente.

Adicionalmente ambos dispositivos deben de establecer una velocidad de comunicación fija a la que se van a transmitir y recibir los datos. Esta velocidad se conoce como *baudrate* y suele medirse en bits por segundo (bps).

En la figura 3.3 se muestra el tren de pulsos transmitido al enviar el byte 01100101 mediante el protocolo de comunicación UART, donde D_N denota al N-ésimo bit (contando de derecha a izquierda). En esta figura se etiqueta al bit de inicio con la letra I, y al bit de paro con la letra P.

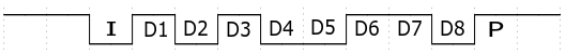

Figura 3.3: Tren de pulsos transmitido al enviar el byte 01100101 mediante el protocolo de comunicación UART.

Como últimas observaciones cabe notar que el byte empieza a transmitirse por sus bits menos significativos, es decir, de derecha a izquierda y que la implementación programada en Verilog está diseñada para trabajar con 2 bits de paro en lugar de uno. Por lo tanto por cada byte de datos que se desee transmitir entre el FPGA y el microcontrolador se enviarán en total 11 bits debido a la presencia del bit de inicio y a los 2 bits de paro adicionales.

Se utilizó Verilog para programar los módulos que implementan la recepción y transmisión de datos utilizando el protocolo UART, diseñados para comunicarse con el microcontrolador a un baudrate fijo de 115200 bps. Como se verá en la sección 3.4 el firmware fue diseñado de tal forma que la información entre el FPGA y el microcontrolador siempre se transmita en paquetes de 4 bytes.

3.3.1.1. Módulo receptor UART

En la figura 3.4 se muestra el diagrama de bloques de este módulo.

Su propósito es recibir órdenes de 4 bytes provenientes del microcontrolador (ARM Cortex-M4 72 MHz), guardarlos en los registros de la salida *Data*, y por último enviar un pulso en alto a la salida *Ready* como un indicador de que los datos se encuentran listos para ser procesados a el módulo Intérprete de Órdenes.

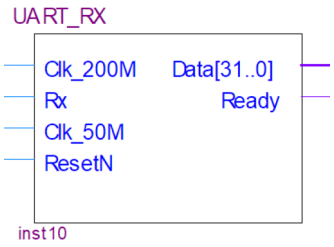

Figura 3.4: Módulo receptor UART.

En la tabla 3.2 se describen las terminales de este módulo junto con sus conexiones correspondientes:

Nombre	Tipo	Conexiones
Clk_200M	Entrada	Salida *c0* - PLL
Rx	Entrada	Terminal TX - Teensy
Clk_50M	Entrada	Reloj Interno de 50MHz
ResetN	Entrada	Salida de compuerta AND con entradas *Clear_UARTN* - Intérprete de Órdenes y *RESETN* - Terminal Física
Data	Salida - 32 Bits	Bus de entrada *Data* - Intérprete de Órdenes
Ready	Salida	Entrada *Rdy* - Intérprete de Órdenes

Tabla 3.2: Terminales y conexiones del módulo receptor UART.

3.3.1.2. Módulo transmisor UART

En la figura 3.5 se muestra el diagrama de bloques de este módulo.

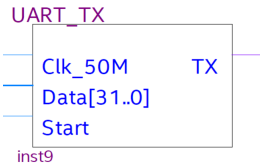

Figura 3.5: Módulo transmisor UART.

Su propósito es enviar los datos que se encuentren en su bus de entrada *Data* al microcontrolador en el momento en que su entrada *Start* reciba un pulso en alto. Las terminales de este módulo y sus conexiones se describen en la tabla 3.3.

Nombre	Tipo	Conexiones
Clk_50M	Entrada	Reloj interno de 50MHz
Data	Entrada - 32 Bits	Bus de salida *Data* - Administrador de Datos
Start	Entrada	Salida *Out* - Ensanchador de Pulsos
TX	Salida	Terminal RX - Teensy

Tabla 3.3: Terminales y conexiones del módulo transmisor UART.

3.4. Intérprete de órdenes

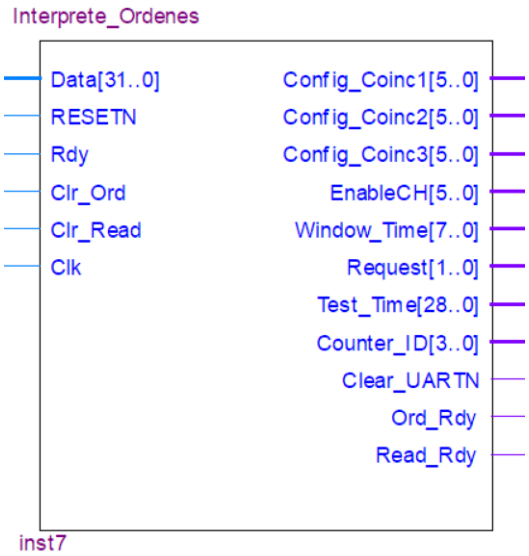

Figura 3.6: Módulo intérprete de órdenes.

El papel de este módulo es procesar la instrucción que se encuentre en su bus de entrada *Data* en el momento en que su entrada *Rdy* reciba un pulso positivo, y dependiendo del tipo de instrucción recibida, guardar un subconjunto de los bytes recibidos en los registros del bus de salida asociado con el tipo de instrucción.

Adicionalmente en el caso de recibir una petición de lectura de contadores también se enviará un pulso positivo mediante la salida *Read_Rdy* para informarle al módulo Administrador de Datos que se desea realizar la lectura del valor de alguno de sus contadores, mientras que si se recibe una orden para iniciar un experimento o para reiniciar el valor de los contadores a cero se realizará la acción análoga con la salida *Ord_Rdy* y el módulo Habilitador de Contadores.

Después de que cada instrucción sea procesada la salida *Clear_UARTN* enviará un pulso negativo al módulo receptor UART para limpiar el valor de sus registros internos. Cuando su entrada *Clr_Ord* reciba un pulso positivo se reiniciará el valor de los registros del bus de salida *Request* a cero, y la situación análoga sucederá con su entrada *Clr_Read* y el bus de salida *Counter_ID*.

El módulo utiliza el valor de los 4 bits más significativos de los datos recibidos para reconocer el tipo de instrucción, a los que nos referiremos como su *código de instrucción*. El diagrama de bloques de este módulo se muestra en la figura 3.6, y sus terminales y conexiones correspondientes se muestran en la tabla 3.4.

A continuación se describen las instrucciones reconocidas por el módulo, su función, el subconjunto de los datos de entrada extraídos por cada una y el bus de salida correspondiente donde se almacenan.

3.4.1. Lectura de contadores

Esta instrucción sirve para leer el valor actual de alguno de los contadores del módulo Contador de Pulsos, y su código de instrucción es 0x6. Cada uno de los contadores tiene un código de contador asociado que consiste en un conjunto de 4 bits que lo identifican unívocamente, denotado por la letra Z en su representación hexadecimal. La instrucción correspondiente para leer el valor del contador con código Z es 0x6Z000000, y los códigos de contador definidos se muestran en la tabla 3.5.

3.4.2. Habilitación de canales

Esta instrucción sirve para seleccionar simultáneamente los canales que se desean habilitar y deshabilitar en el experimento, y su código de instrucción es 0x2. Los bits restantes que componen a la instrucción incluyen la información acerca de la configuración de canales deseada, y se construyen de la siguiente manera:

- $X[32:29] = 0x2$ (Código de instrucción).
- $X[i] = 1$ si se desea habilitar el canal i-ésimo , 0 en otro caso ($i \in \{1, \ldots, 6\}$).

Nombre	Tipo	Conexiones
Data	Entrada - 32 Bits	Bus de salida *Data* - Receptor UART
RESETN	Entrada	RESETN - Terminal física
Rdy	Entrada	Salida *Ready* - Receptor UART
Clr_Ord	Entrada	Salida *Clr_Ord* - Habilitador de Contadores
Clr_Read	Entrada	Salida *Clr_Read* - Administrador de Datos
Clk	Entrada	Reloj PLL de 200 MHz
$Config\_Coinc_N$	Salida - 6 Bits	Bus de entrada *Enable_Coinc* - Detector de Coincidencias N
EnableCH	Salida - 6 Bits	Entrada *EnableCHS* - Habilitador de Canales
Window_Time	Salida - 8 Bits	RESETN - Terminal física
Request	Salida - 2 Bits	Bus de Entrada *Order* - Habilitador de Contadores
Test_Time	Salida - 24 Bits	Bus de Entrada *Test_Time* - Generador de Tiempo de Prueba
Counter_Id	Salida - 4 Bits	Bus de Entrada *Code_ID* - Administrador de Datos
Clear_UARTN	Salida	Entrada de compuerta AND que alimenta a la entrada *RESETN* - Receptor UART
Ord_Rdy	Salida	Entrada *Ord_Rdy* - Habilitador de Contadores
Read_Rdy	Salida	Entrada *Read_Rdy* - Administrador de Datos

Tabla 3.4: Terminales y conexiones del módulo Contador de Pulsos.

- $X[i] = 0$ para todos los bits restantes.

Como ejemplo, si los 6 bits menos significativos de una instrucción para habilitar canales son 011010 entonces se habilitarán los canales 2, 4 y 5 y se deshabilitarán los canales 1, 3 y 6.

Contador	Código	
	Hexadecimal	Binario
Contador de Pulsos CH1	1	0001
Contador de Pulsos CH2	2	0010
Contador de Pulsos CH3	3	0011
Contador de Pulsos CH4	4	0100
Contador de Pulsos CH5	5	0101
Contador de Pulsos CH6	6	0110
Contador de Coincidencias 1	7	0111
Contador de Coincidencias 2	8	1000
Contador de Coincidencias 3	9	1001

Tabla 3.5: Códigos de contador reconocidos por el módulo Intérprete de Órdenes.

3.4.3. Configuración del tiempo de prueba

Esta instrucción sirve para establecer el tiempo de prueba que se desea utilizar en el experimento, que por definición es el tiempo durante el que se procesarán los pulsos incidentes sobre los canales que se encuentren habilitados, y su código de instrucción es 0x4. Sea T el tiempo de prueba que se desea establecer expresado en unidades de 100 ns (0.1 μs), y sea K su correspondiente representación numérica en el sistema binario.

El firmware solo utiliza 29 bits para representar a K por lo que se restringió el software para que T solo pueda tomar valores desde 0 hasta 1.5 segundos. Los bits de la instrucción correspondiente para establecer el tiempo de prueba deseado se construyen de la siguiente manera:

- $X[32:29] =$ 0x4 (Código de instrucción).
- $X[24:1] = K$.
- $X[i] = 0$ en otro caso.

Por ejemplo si se desea establecer un tiempo de prueba de 1 s, que medido en unidades de 100 ns es igual a 1×10^7, los 24 bits menos significativos de la instrucción correspondiente serán 100110001001011010000000 y todos los bits restantes con excepción de su código de su instrucción serán cero.

3.4.4. Configuración de la ventana de coincidencia

Esta instrucción sirve para establecer la ventana de coincidencia a utilizar en el experimento por los módulos detectores de coincidencias, y su código de instrucción es 0x3. Sea τ la ventana de coincidencia que se desea establecer expresada como múltiplo de 5 ns, y sea K su representación númerica en el sistema binario.

El firmware solo utiliza 8 bits para representar la ventana de coincidencia por lo que se restringió el software para poder seleccionar únicamente valores de la ventana de coincidencia que estén entre 5 y 1250 ns, ya que el Laboratorio de Óptica Avanazada especificó que estos valores eran adecuados para sus propósitos. Los bits de la instrucción correspondiente para establecer la ventana de coincidencia deseada se construyen de la siguiente manera:

- $X[32:29] = 0x3$ (Código de instrucción).
- $X[8:1] = K$.
- $X[i] = 0$ en otro caso.

Por ejemplo si se desea establecer un tiempo de prueba de 565 ns, cuyo valor en unidades de 5 ns es 113, los 8 bits menos significativos de la instrucción correspondiente para establecer esta ventana de coincidencia serán 01110001 y todos los bits restantes con excepción de su código de instrucción serán cero.

3.4.5. Configuración de detectores de coincidencias

Esta instrucción sirve para indicarle a cada módulo Detector de Coincidencias (MDC) los canales sobre los cuales se desea que verifique si existen señales de entrada coincidentes. El firmware contiene a 3 de estos módulos y cada uno de ellos tiene asociado un código de instrucción particular, los cuales se muestran en la tabla 3.6.

Sea Z el código de instrucción del módulo detector de coincidencias que se desea configurar representado en el sistema hexadecimal. Los bits que componen a una instrucción general para configurar el módulo correspondiente se construyen de la siguiente manera:

- $X[32:29] = Z$.
- $X[i] = 1$ si se desea que el canal i-ésimo participe en la coincidencia, 0 en otro caso ($i \in \{1, \ldots, 6\}$).

Número de Detector	Código de Instrucción	
	Hexadecimal	Binario
1	A	1010
2	B	1011
3	C	1100

Tabla 3.6: Códigos de instrucción utilizados para configurar los MDC's.

- $X[i] = 0$ para todos los bits restantes.

Por ejemplo si queremos que el módulo Detector de Coincidencias 1 se encargue de detectar coincidencias triples entre los canales 1, 2 y 6 los 4 bits mas significativos de la instrucción enviada serán 1010 (0xA), sus 6 bits menos significativos serán 100011 y todos sus bits restantes serán 0.

3.4.6. Iniciar experimento y reiniciar contadores

Este conjunto de instrucciones sirve para indicarle al dispositivo cuando se desea iniciar un nuevo experimento y/o reiniciar el valor de todos sus contadores a cero. Ambas instrucciones tienen asignado el mismo código de instrucción 0x5. Los bits de estas instrucciones se construyen de la siguiente manera:

- $X[32:29] = 0$x5 (Código de instrucción).
- $\mathrm{X[2:1]} = \begin{cases} 01 & \text{para reiniciar contadores.} \\ 10 & \text{para reiniciar contadores e iniciar un nuevo experimento.} \end{cases}$
- $X[i] = 0$ para todos los bits restantes.

3.5. Detector de pulsos

El diagrama de bloques de este módulo se muestra en la figura 3.7.

El propósito de este módulo es recibir como entradas las señales de salida de los APDs en su bus de entrada *Signals* y convertirlas en pulsos con una duración de 5 ns, que serán transmitidos a través de su bus de salida *Pulses*. Esto se hace debido a que las señales de los APDs tienen un ancho de aproximadamente 25 ns, y es útil hacer esta conversión para hacer mas sencillo su procesamiento. Además,

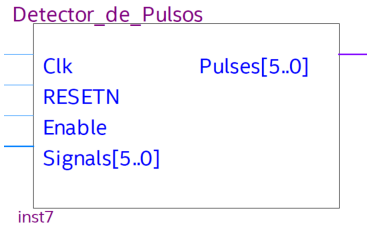

Figura 3.7: Diagrama de bloque del módulo Detector de Pulsos.

para la detección de coincidencias únicamente es de importancia el momento de llegada de las señales, y al trabajar con ventanas de coincidencia que son múltiplos de 5 ns se elimina la posibilidad de registrar dos veces la misma coincidencia.

Mientras el valor de su entrada *Enable* se encuentre en alto el módulo estará en funcionamiento, y en caso contrario el valor de todas las líneas que componen al bus de salida *Pulses* será 0 independientemente de la existencia de pulsos de entrada en las terminales correspondientes. Las terminales de este módulo y sus conexiones se describen en la tabla 3.7.

Nombre	Tipo	Conexiones
Clk	Entrada	Salida *c0* - PLL
RESETN	Entrada	Salida de compuerta AND con entradas *RESETN* - Terminal física y NOT(*Reset*) - Habilitador de Contadores
Enable	Entrada	Salida *Enable_Test* - Generador de Tiempo de Prueba
Signals	Entrada - 6 Bits	Salidas de los APDs
Pulses	Salida - 6 Bits	Bus de entrada *Pulses* - Habilitador de Canales

Tabla 3.7: Terminales y conexiones del módulo Detector de Pulsos.

El funcionamiento lógico del módulo para uno de sus canales se muestra en el diagrama de tiempos de la figura 3.8.

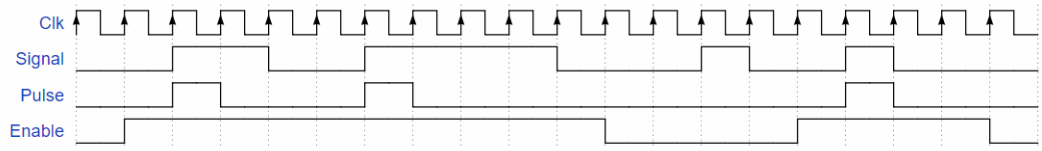

Figura 3.8: Diagrama de tiempos del módulo Detector de Pulsos.

3.6. Habilitador de canales

El diagrama de bloques de este módulo se muestra en la figura 3.9.

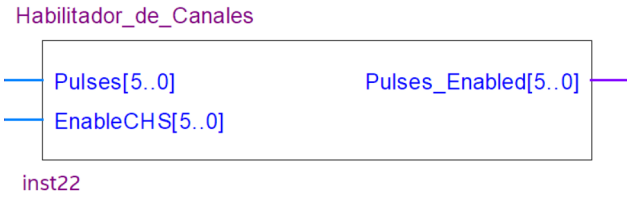

Figura 3.9: Diagrama de bloques del módulo Habilitador de Canales.

Su propósito es tomar los pulsos producidos por el módulo Detector de Pulsos, presentes en su entrada *Pulses*, y únicamente dejarlos pasar a través de su bus de salida *Pulses_Enabled* si los canales correspondientes se encuentran activados. Las terminales de este módulo y sus conexiones se muestran en la tabla 3.8.

Nombre	Tipo	Conexiones
Pulses	Entrada - 6 Bits	Bus de Salida *Pulses* - Detector de Pulsos
EnableCHS	Entrada - 6 Bits	Bus de Salida *EnableCH* - Intérprete de Órdenes
Pulses_Enabled	Salida - 6 Bits	Bus de entrada *In_Pulses* - Detectores de Coincidencias y primeros 6 Bits del bus de Entrada *In_Pulses* - Contador de Pulsos

Tabla 3.8: Terminales y conexiones del módulo Habilitador de Canales.

Este módulo no necesita un reloj de entrada debido a que su única tarea es evaluar la siguiente función lógica para cada una de sus entradas y generarlo en las salidas correspondientes, donde el subíndice n denota al n-ésimo bit del bus correspondiente:

$$Pulses\_Enabled_n = Pulses_n \wedge EnableCHS_n \tag{3.1}$$

Recordemos que el bit n-ésimo del bus de entrada *EnableCHS* será 1 únicamente si el canal correspondiente se encuentra habilitado y 0 en otro caso, por lo que la expresión lógica 3.1 tomará el valor uno si y solo si existe un pulso de entrada en el canal correspondiente y además el canal se encuentra habilitado.

Este módulo es necesario por que mientras el módulo Detector de Pulsos se encuentre habilitado dejará pasar los pulsos entrantes en todos sus canales independientemente de que se encuentren o no habilitados, dejándole al módulo Habilitador de Canales la tarea de permitirle el paso únicamente a los pulsos presentes en los canales que si lo estén.

3.7. Detector de coincidencias

El diagrama de bloques de este módulo se muestra en la figura 3.10.

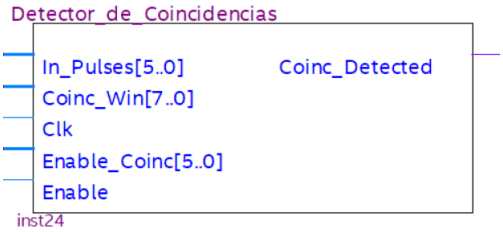

Figura 3.10: Diagrama de bloques del módulo Detector de Coincidencias.

El propósito de este módulo es recibir como entradas las pulsos procedentes del Habilitador de Canales y el valor de la ventana de coincidencia procedente del Intérprete de Órdenes en sus buses de entrada *In_Pulses* y *Coinc_Win*, respectivamente, y producir un pulso de 5 ns en su salida *Coinc_Detected* al detectar una coincidencia entre los canales configurados para participar en la detección de coincidencias.

El bus de entrada *Coinc_win* sirve para introducir la representación binaria de la ventana de coincidencia en unidades de 5 ns, la cual se utilizará para definir si dos señales son o no coincidentes. El bus de entrada *Enable_Coinc* tiene la función de indicarle al módulo qué canales deben de participar en la coincidencia y cuales deben de ser ignorados, lo que permite configurar al módulo para detectar coincidencias entre 2, 3, 4, 5 y hasta 6 señales[1]. La entrada *Enable* permite activar

[1]La arquitectura del módulo se puede cambiar fácilmente para poder detectar coincidencias

o desactivar el funcionamiento de este módulo dependiendo de si el valor lógico de su entrada es 1 o 0, respectivamente. El firmware desarrollado incluye 3 de estos módulos; sus terminales y conexiones se muestran en la tabla 3.9.

Nombre	Tipo	Conexiones
In_Pulses	Entrada - 6 Bits	Bus de salida *Pulses_Enabled* - Habilitador de Canales
Coinc_Win	Entrada - 8 Bits	Bus de Salida *Window_Time* - Intérprete de Órdenes
Clk	Entrada	Salida *c0* - PLL
Enable_Coinc	Entrada - 6 Bits	Buses de salida *Config_Coinc1*, *Config_Coinc2* o *Config_Coinc3* dependiendo del módulo - Intérprete de Órdenes.
Enable	Entrada	Salida *Out* - Retardador de Apagado.
Coinc_Detected	Salida	Bit 9, 8 o 7 del bus de entrada *In_Pulses* dependiendo del módulo - Contador de Pulsos.

Tabla 3.9: Terminales y conexiones de los módulos Detectores de Coincidencias.

Conceptualmente la forma de realizar la detección de coincidencias es la siguiente: el ancho de los pulsos de entrada presentes en cada canal configurado para participar en la coincidencia será modificado para tener un ancho igual a la ventana de coincidencia, y estas señales serán alimentadas como entradas de una compuerta AND. En el instante en que la salida de esta compuerta sea 1 se habrá detectado una coincidencia. Cuando esto suceda se producirá un pulso positivo en la salida *Coinc_Detected* y el módulo se reiniciará inmediatamente para estar disponible para detectar coincidencias futuras. Un ejemplo del método desarrollado para realizar la detección de coincidencias se muestra en la figura 3.11.

3.8. Contador de pulsos

El diagrama de bloques de este módulo se muestra en la figura 3.12.

entre un número arbitrario de señales. Debido a el número de canales presentes en el diseño es posible realizar, a lo más, coindencias entre 6 señales.

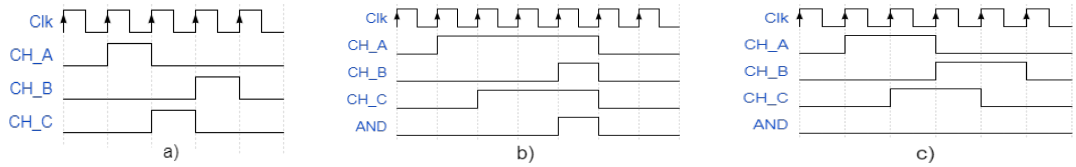

Figura 3.11: Ilustración del método utilizado para realizar la detección de coincidencias entre 3 señales. a) Pulsos entrantes al módulo Detector de Coincidencias. b) Diagrama de tiempos de los pulsos internos generados por el módulo cuando la ventana de coincidencia es de 20 ns. En este caso se detecta una coincidencia. c) Diagrama de tiempos de los pulsos internos generados por el módulo cuando la ventana de coincidencia es de 10 ns. En este caso no se detecta ninguna coincidencia.

Su propósito es contar los pulsos detectados en cada canal que se encuentre habilitado junto con las coincidencias detectadas por cada uno de los 3 Detectores de Coincidencias presentes en el diseño, que están asociados con las cuentas GT, GR y GTR (o TR) al realizar experimentos de Hanbury-Brown y Twiss. Las terminales y conexiones de este módulo se muestran en la tabla 3.10.

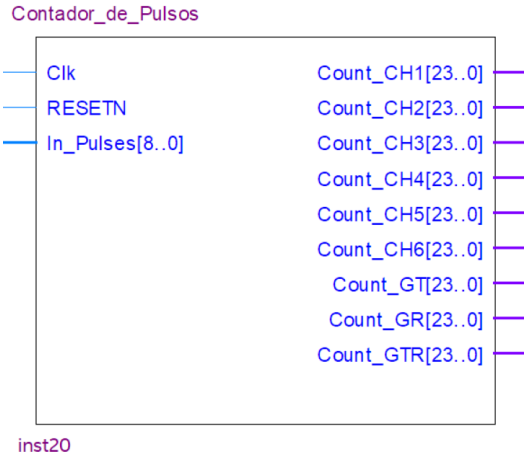

Figura 3.12: Diagrama de bloques del módulo Contador de Pulsos.

Cabe aclarar que los 6 bits menos significativos del bus de entrada *In_Pulses* están conectados a los bits correspondientes del puerto con el mismo nombre del Habilitador de Canales, mientras que los 3 bits más significativos se encuentran conectados a las salidas *Coinc_Detected* de los módulos Detectores de Coincidencias.

Nombre	Tipo	Conexiones
Clk	Entrada	Salida *c0* - PLL
RESETN	Entrada	Salida de compuerta AND con entradas *RESETN* - Terminal física y NOT(*Reset*) - Habilitador de Contadores
In_Pulses	Entrada - 9 Bits	Bus de Salida *Pulses_Enabled* de 6 Bits - Habilitador de Canales y salidas *Coinc_Detected* - Detectores de Coincidencias
Count_CH$_N$	Entrada - 24 Bits	Bus de entrada *CH$_N$* - Administrador de Datos
Count_GT	Entrada - 24 Bits	Bus de entrada *GT* - Administrador de Datos
Count_GR	Entrada - 24 Bits	Bus de entrada *GR* - Administrador de Datos
Count_GTR	Entrada - 24 Bits	Bus de entrada *GTR* - Administrador de Datos

Tabla 3.10: Terminales y conexiones del módulo Contador de Pulsos.

3.9. Administrador de datos

En la figura 3.13 se muestra el diagrama de bloques de este módulo. Su propósito es responder a las peticiones de lectura de contadores e indicarle al módulo Transmisor UART los datos correspondientes que debe de enviar hacia el microcontrolador.

Su funcionamiento es el siguiente: cuando su entrada *Read_Rdy* reciba un pulso positivo el módulo analizará el código del contador que se desea leer, presente en su bus de entrada *Code_ID*, y guardará las cuentas y el código de contador correspondiente en los registros de su bus de salida *Data*.

Un ciclo de reloj después el módulo producirá un pulso positivo en su salida *Clr_Read*, lo que le indicará al módulo transmisor UART (a través del módulo Ensanchador de Pulsos) que debe de iniciar el envío de los datos correspondientes , y que al mismo tiempo reiniciará el valor de los registros del bus de Salida *Request* del módulo Intérprete de Órdenes a través de su entrada *Clr_read*. Las terminales y conexiones de este módulo se describen en la tabla 3.11.

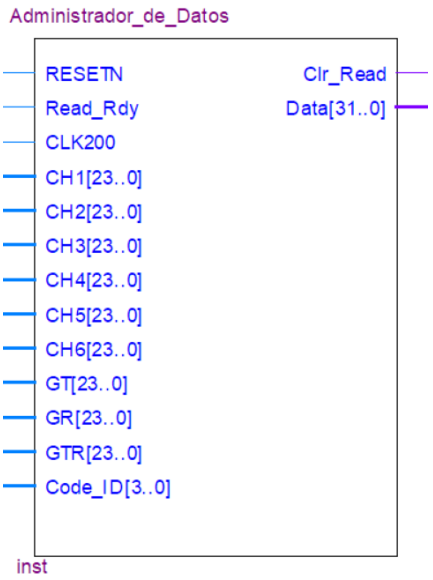

Figura 3.13: Diagrama de bloques del módulo Administrador de Datos.

3.10. Generador de tiempo de prueba

El diagrama de bloques de este módulo se muestra en la figura 3.14.

Su propósito es habilitar a los módulos Detector de Pulsos y Retardador de Apagado durante una cantidad de tiempo igual al tiempo de prueba deseado.

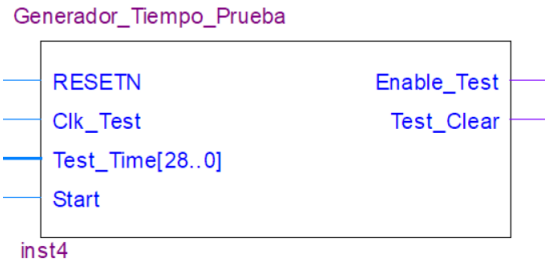

Figura 3.14: Diagrama de bloques del módulo Generador de Tiempo de Prueba.

Cuando su entrada *Start* reciba un pulso positivo el módulo producirá una señal en alto en su salida *Enable_Test* durante una cantidad de tiempo igual al especificado en su bus de entrada *Test_Time*. Al finalizar, el valor de su salida *Enable_Test* regresará a cero, mientras que en su salida *Test_Clear* se producirá un pulso positivo para informarle al Habilitador de Canales que el tiempo de prueba ha finalizado. Las terminales y conexiones de este módulo se describen en la tabla 3.12.

Nombre	Tipo	Conexiones
RESETN	Entrada	*RESETN* - Terminal física
Read_Rdy	Entrada	Salida *Read_Rdy* - Intérprete de Órdenes
CLK200	Entrada	Salida *c0* - PLL
CH_N	Entrada - 24 Bits	Bus de salida $Count\_CH_N$ - Contador de Pulsos
GT	Entrada - 24 Bits	Bus de salida *Count_GT* - Contador de Pulsos
GR	Entrada - 24 Bits	Bus de salida *Count_GR* - Contador de Pulsos
GTR	Entrada - 24 Bits	Bus de salida *Count_GTR* - Contador de Pulsos
Code_ID	Entrada - 4 Bits	Bus de salida *Counter_ID* - Intérprete de Órdenes
Clr_Read	Salida	Entrada *Clr_Read* - Intérprete de Órdenes y entrada *In* - Ensanchador de Pulsos
Data	Salida - 32 Bits	Bus de entrada *Data* - Transmisor UART

Tabla 3.11: Terminales y conexiones del módulo Administrador de Datos.

3.11. Ensanchador de Pulsos

El diagrama de bloques de este módulo se muestra en la figura 3.15 . Su propósito es convertir los pulsos de 5 ns producidos por el módulo Administrador de Datos en su salida *Clr_Read* en pulsos con un nuevo ancho de 20 ns. La necesidad de incluir este módulo en el diseño se describe a continuación.

Figura 3.15: Diagrama de bloques del módulo Ensanchador de Pulsos.

Nombre	Tipo	Conexiones
RESETN	Entrada	*RESETN* - Terminal física
Clk_Test	Entrada	Salida *c0* - PLL
Test_Time	Entrada - 29 Bits	Bus de salida *Test_Time* - Intérprete de Órdenes
Start	Entrada	Salida *StartTest* - Habilitador de Contadores
Enable_Test	Salida	Entrada *In* - Retardador de Apagado y entrada *Enable* - Detector de Pulsos
Test_Clear	Salida	Entrada de compuerta NOT con salida *ResetTest* - Habilitador de Contadores

Tabla 3.12: Terminales y conexiones del módulo Generador de Tiempo de Prueba.

Debido a que el módulo transmisor UART funciona con una frecuencia de reloj menor que el módulo Administrador de Datos (50 y 200 MHz respectivamente), por lo que es necesario ensanchar los pulsos de salida del administrador para que el transmisor UART sea capaz de detectarlos. Las terminales y sus conexiones correspondientes se muestran en la tabla 3.13.

Nombre	Tipo	Conexiones
In	Entrada	Salida *Clr_Read* - Administrador de Datos
Clk	Entrada	Salida *c0* - PLL
Out	Salida	Entrada *Start* - Transmisor UART

Tabla 3.13: Terminales y conexiones del módulo Ensanchador de Pulsos.

3.12. Habilitador de contadores

El diagrama de bloques de este módulo se muestra en la figura 3.16 . Su propósito es indicarle al módulo Generador de Tiempo de Prueba cuando se desea inciar un nuevo experimento. Además, este módulo se encarga de realizar un reinicio de los módulos Contador de Pulsos y Detector de Pulsos antes de comenzar cada experimento o al recibir la instrucción correspondiente del Intérprete de Órdenes. El diagrama de bloques de este módulo se muestra en la figura 3.16.

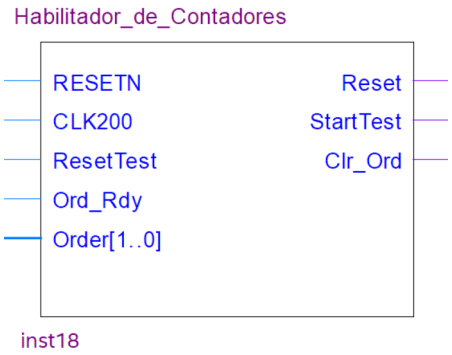

Figura 3.16: Diagrama de bloques del módulo Habilitador de Contadores.

Cuando su entrada *Ord_Rdy* reciba un pulso positivo el módulo analizará la información presente en su bus de entrada *Order* para saber que tipo de instrucción se desea realizar. En el caso de que la instrucción leída sea una instrucción de reinicio únicamente se limpiaran los registros de los módulos Contador de Pulsos y Detector de Pulsos al enviar un pulso negativo a través de su salida *Reset*.

En caso de que la instrucción recibida sea una instrucción para comenzar un nuevo experimento primero se realizará el proceso correspondiente a una instrucción de reinicio y después se enviará un pulso positivo a través de su salida *StartTest* para indicarle al módulo Generador de Tiempo de Prueba que debe de comenzar su funcionamiento. Los terminales y conexiones correspondientes de este módulo se muestran en la tabla 3.14.

3.13. Retardador de Apagado

El diagrama de bloques de este módulo se muestra en la figura 3.17, y su propósito es ensanchar un par de ciclos de reloj adicionales la señal presente en su entrada *In*. Las señales en sus terminales *In* y *Out* son prácticamente iguales con la diferencia de que la señal de salida tardará un par extra de ciclos de reloj en regresar al nivel lógico 0 que la correspondiente señal de entrada.

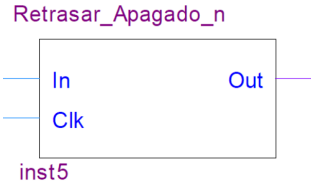

Figura 3.17: Diagrama de bloques del módulo Retardador de Apagado.

Nombre	Tipo	Conexiones
RESETN	Entrada	*RESETN* - Terminal física
Clk200	Entrada	Salida *c0* - PLL
ResestTest	Entrada	Salida de compuerta NOT con entrada *Test_Clear* - Generador de Tiempo de Prueba
Ord_Rdy	Entrada	Salida *Ord_Rdy* - Intérprete de Órdenes
Order	Entrada - 2 Bits	Bus de salida *Request* - Intérprete de Órdenes
Reset	Salida	NOT(*Reset*) es una entrada de una compuerta AND que alimenta a las entradas *RESETN* - Detector de Pulsos y *RESETN* - Contador de Pulsos
StartTest	Salida	Entrada *Start* - Generador de Tiempo de Prueba
Clr_Ord	Salida	Entrada *Clr_Ord* - Intérprete de Órdenes

Tabla 3.14: Terminales y conexiones del módulo Habilitador de Contadores.

Este módulo es necesario debido a que los módulos Detectores de Coincidencias necesitan estar habilitados un par de ciclos de reloj adicionales que el módulo Detector de Pulsos para que sean capaces de terminar de procesar las señales que llegaron justo antes de deshabilitarlo. Las terminales y conexiones de este módulo se muestran en la tabla 3.15.

Nombre	Tipo	Conexiones
In	Entrada	Salida *Enable_Test* - Generador de Tiempo de Prueba
Clk	Entrada	Salida *c0* - PLL
Out	Salida	Entrada *Enable* - Detectores de Coincidencias

Tabla 3.15: Terminales y conexiones del módulo Retardador de Apagado.

3.14. Conclusión

En este capítulo se mostraron todos los diferentes módulos que componen al firmware y se describió el papel que tiene cada uno de ellos en el sistema completo de instrumentación. El lenguaje de descripción de hardware Verilog fue muy útil para acelerar el desarrollo de todos estos módulos, ya que permite diseñarlos con un nivel de abstracción mucho mayor que el correspondiente a la programación explícita de las compuertas lógicas que los componen.

A partir del software Quartus de Altera se pudieron construir simulaciones lógicas de todos los módulos desarrollados, lo que permitió disminuir el tiempo de programación y verificar que se comportaban de acuerdo a lo esperado. Debido a que todos los módulos se ejecutan de forma simultánea (en paralelo) es posible realizar diversas tareas que no podríamos realizar con otros dispositivos programables como los microcontroladores, como realizar la detección de pulsos de entrada en los 6 canales de forma simultánea sin necesidad de estar iterando entre cada uno de ellos.

La manera en que se programaron estos módulos permite extender fácilmente el firmware desarrollado para realizar nuevas tareas a medida que estas sean necesarias, así como cambiar fácilmente distintos parámetros internos del dispositivo como lo son el número máximo de pulsos que se pueden contar por canal, el número de canales de entrada y el número de señales sobre las cuales se desea detectar coincidencias, entre otros.

Además, la mayoría de los módulos que componen al firmware son independientes del hardware en el que éste se implemente, por lo que en caso de tener acceso a FPGAs con una mayor velocidad de reloj, mayor número de compuertas lógicas disponibles, o distintas funcionalidades adicionales el diseño desarrollado podría adaptarse fácilmente para poder implementarse en ellos.

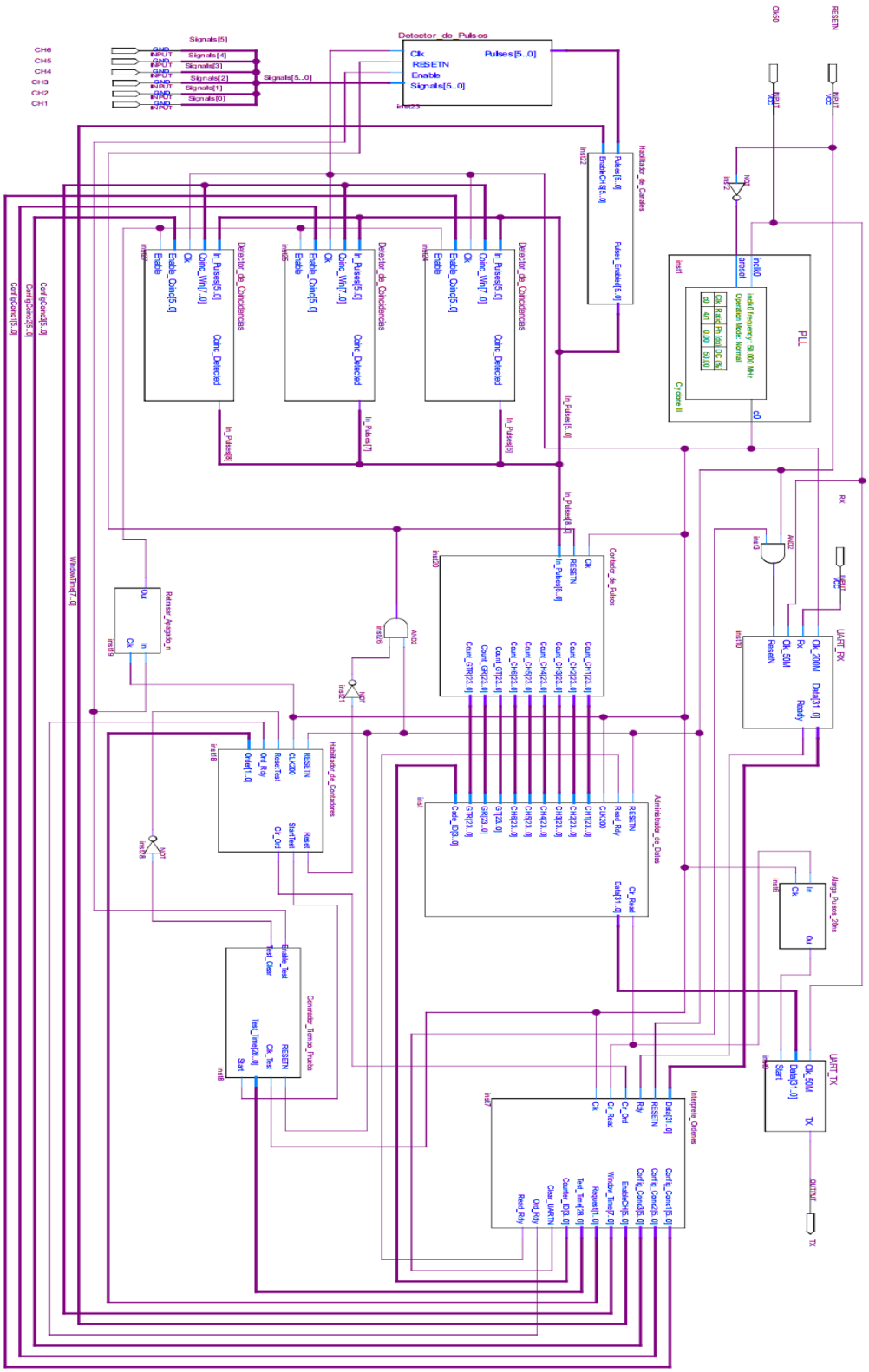

Figura 3.18: Diagrama por bloques del firmware desarrollado.

Capítulo 4

Diseño del Software

En este capítulo se describen los requisitos, las herramientas y la metodología utilizada para desarrollar el software que se encarga, entre otras cosas, de establecer la comunicación bidireccional entre el dispositivo de instrumentación y la computadora, con el propósito de realizar el envío de órdenes hacia el instrumento y realizar la extracción de los datos experimentales recopilados, producir de forma automática las gráficas más adecuadas para el tipo de experimento realizado y generar archivos con formato *.csv* de forma automática con los datos recopilados.

Se discute también la motivación que llevó a tomar cada una de las decisiones que impactaron las características y propiedades del software final desarrollado, las ventajas y desventajas de las herramientas utilizadas respecto a otras alternativas, y posibles mejoras a implementar en la siguiente actualización del software.

El objetivo del software es permitirle al usuario controlar el dispositivo de instrumentación desarrollado sin tener que preocuparse de la implementación o la arquitectura interna de éste, siendo así posible que el usuario sea capaz de poder modificar a voluntad los parámetros del experimento y realizar diversas pruebas sin que sea necesario que el usuario tenga conocimientos en el área de programación.

El software debe de estar diseñado de tal forma que le permita a una persona familiarizada con el desarrollo de software poder modificarlo para que éste se ajuste fácilmente a sus necesidades. Para esto se utilizó un diseño modular en el cual se creó un módulo que contiene funciones generales para enviar cada una de las instrucciones mencionadas en la sección 3.4 y administrar la comunicación serial entre la computadora personal (PC, *Personal Computer*) y el dispositivo, y otro en el cual se encuentra programada la implementación específica desarrollada en este trabajo que se describe en la sección 4.4.

4.1. Requisitos del software

El software a desarrollar debe de cumplir con los siguientes requisitos:

- Implementar el protocolo de comunicación UART para establecer la comunicación entre la computadora y el dispositivo de instrumentación.
- Definir funciones que permitan realizar el envío de cualquiera de las órdenes descritas en la sección 3.4 al dispositivo.
- Realizar la extracción de los datos experimentales almacenados en el dispositivo.
- Crear archivos con formato *.csv* de forma automatizada para almacenar los datos extraídos.
- Producir la gráfica correspondiente al experimento realizado.
- Realizar un análisis básico de los datos experimentales recopilados y calcular el valor de las variables de interés del experimento.

4.2. Metodología de desarrollo

La metodología utilizada para desarrollar el software se conoce con el nombre de *modelo de cascada*. En esta metodología se deben seguir los siguientes pasos:

- Definición de requisitos.
- Análisis y diseño del software.
- Implementación y prueba de módulos.
- Integración y prueba del software.
- Operación y mantenimiento.

Al iniciar un nuevo proyecto estos pasos deben de realizarse de manera secuencial desde el inicio hasta llegar al paso final. Al terminar, el software se mantiene en chequeo constante, y cuando se encuentre alguna deficiencia o se quiera agregar una nueva funcionalidad se vuelve a comenzar el método desde el paso que se considere adecuado y se realizan de nuevo todos los pasos posteriores. Esta metodología se encuentra resumida en la figura figura 4.1.

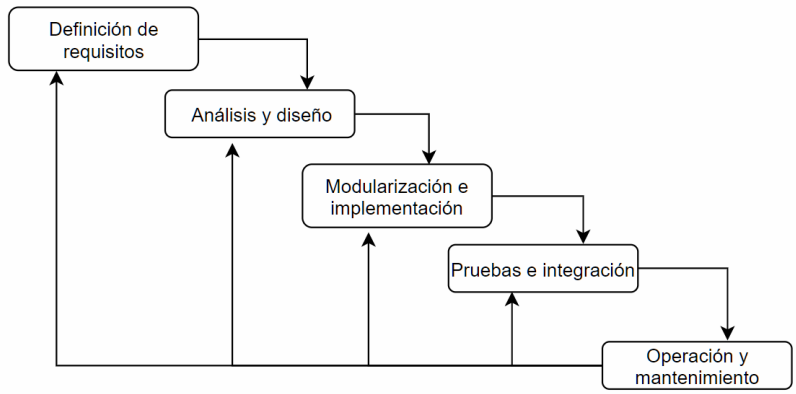

Figura 4.1: Metodología de desarrollo.

4.3. Herramientas utilizadas

Una vez definidos los requisitos que debe satisfacer el software, se tomó una decisión respecto a las herramientas que se utilizaron para desarrollarlo; en este caso se eligieron, entre otras cosas:

1. El lenguaje de programación.
2. El formato de guardado de los archivos de datos obtenidos.
3. El formato de guardado de las gráficas producidas.

Respecto al punto 1 se decidió escoger el lenguaje de programación Python (38). Python es un lenguaje de programación interpretado multiparadigma de código abierto, que actualmente es utilizado en una gran variedad de tareas como:

- Análisis numérico y programación.
- Desarrollo de aplicaciones web.
- Ciencia de datos y aprendizaje máquina.
- Automatización de procesos.
- Procesamiento de lenguaje natural.
- Desarrollo de interfaces gráficas de usuario.

Se decidió utilizar Python debido a que es un lenguaje de alto nivel que tiene diversas ventajas: es relativamente sencillo de aprender, es uno de los lenguajes de programación con mayor cantidad de usuarios por lo que la documentación existente es muy amplia, las aplicaciones desarrolladas en Python son fácilmente portables a diversos sistemas operativos y existe una gran variedad de librerías desarrolladas por sus usuarios para realizar todo tipo de tareas. En particular en este proyecto se utilizaron, entre otras, las siguientes librerías:

- PySerial (39): posee todo tipo de funciones muy útiles que permiten inicializar y configurar la comunicación serial con un dispositivo externo y recibir/enviar datos por el puerto serie.
- Csv (40): permite leer y guardar archivos con formato csv.
- PyQt (41): permite desarrollar interfaces gráficas de usuario con la librería gráfica QT.
- Matplotlib (42): permite, entre otras cosas, producir gráficas de manera sencilla y exportarlas en una gran variedad de formatos de salida.
- Seaborn (43): librería grafica basada en Matplolib que permite, entre otras cosas, mejorar considerablemente la apariencia de las gráficas producidas.
- Numpy (44): contiene estructuras de datos y funciones optimizadas para trabajar con grandes volúmenes de datos de una forma sencilla y eficiente.

Respecto al punto 2 se decidió utilizar el formato .csv debido a que la mayoría de los lenguajes de programación, e incluso excel o un editor de texto, pueden leer y modificar este tipo de datos, y respecto al punto 3 la librería Matplotlib es una elección natural ya que nos permite exportar las gráficas producidas en prácticamente cualquier formato de salida deseado.

Además, en el pasado ya se habían utilizado estas herramientas en diversos proyectos relacionados, por lo que la experiencia adquirida sirvió para agilizar el desarrollo de este software.

4.4. Descripción del software desarrollado

Como se mencionó en la sección anterior se utilizó el lenguaje de programación Python junto con las librerías PyQt, Numpy, Csv, Matplotlib y Pyserial, entre otras, para desarrollar una interfaz gráfica de usuario encargada de controlar al dispositivo.

La interfaz se diseñó con el propósito de automatizar dos tipos de experimentos que suelen llevarse a cabo de forma regular en el Laboratorio de Óptica Avanzada de la Facultad de Ciencias, UNAM: experimentos de conteo de fotones y el experimento de HBT. El propósito de éstos experimentos usualmente es estudiar la estadística de fotones del haz incidente y encontrar el valor de su función de correlación de segundo orden en $t = 0$, con la intención de demostrar experimentalmente la existencia de fuentes sub-poissonianas y el antiagrupamiento de fotones.

El software se comunica a través del puerto USB de la computadora con el dispositivo a una velocidad de comunicación constante de 115200 bps, permite automatizar el guardado de los archivos de datos medidos en formato *.csv*, produce distintas gráficas dependiendo del tipo de experimento realizado y permite exportarlas en distintos formatos.

Todas las gráficas producidas son interactivas por lo que se puede cambiar el nombre de los ejes, el título principal, aplicar transformaciones log o logit sobre los ejes, y amplificar o trasladar las gráficas a voluntad con el propósito de agilizar el análisis exploratorio de los datos obtenidos.

La interfaz realiza una validación de los parámetros y opciones seleccionadas por el usuario. Por ejemplo, no se puede introducir una ventana de coincidencia que no sea un múltiplo de 5ns o que no se encuentre dentro del intervalo [5, 1250] debido a que el hardware fue diseñado para operar únicamente con estos valores. Constricciones similares se realizan con los demás parámetros y opciones de la interfaz para disminuir en la medida de lo posible los errores de usuario y mejorar su experiencia al utilizar la interfaz.

Las gráficas producidas por el software son interactivas (pueden ser trasladadas y las regiones de interés pueden amplificarse a voluntad) y al hacer click derecho sobre ellas o al hacer click sobre los íconos correspondientes se pueden modificar algunos de sus parámetros y exportarlas en una gran variedad de formatos como *jpg*, *png*, *eps* y *pdf*, entre otros.

A continuación se describen las distintas variaciones de estos experimentos que podemos realizar con esta interfaz, los resultados obtenidos en cada caso, y las gráficas que serán producidas para cada tipo de experimento. La ventana principal de la interfaz desarrollada se muestra en la figura 4.2.

En las secciones siguientes se utilizará la notación T_p y V_c para denotar al tiempo de prueba y a la ventana de coincidencia, cuyas definiciones y papel que juegan en el experimento pueden consultarse en la sección 1.6.7.

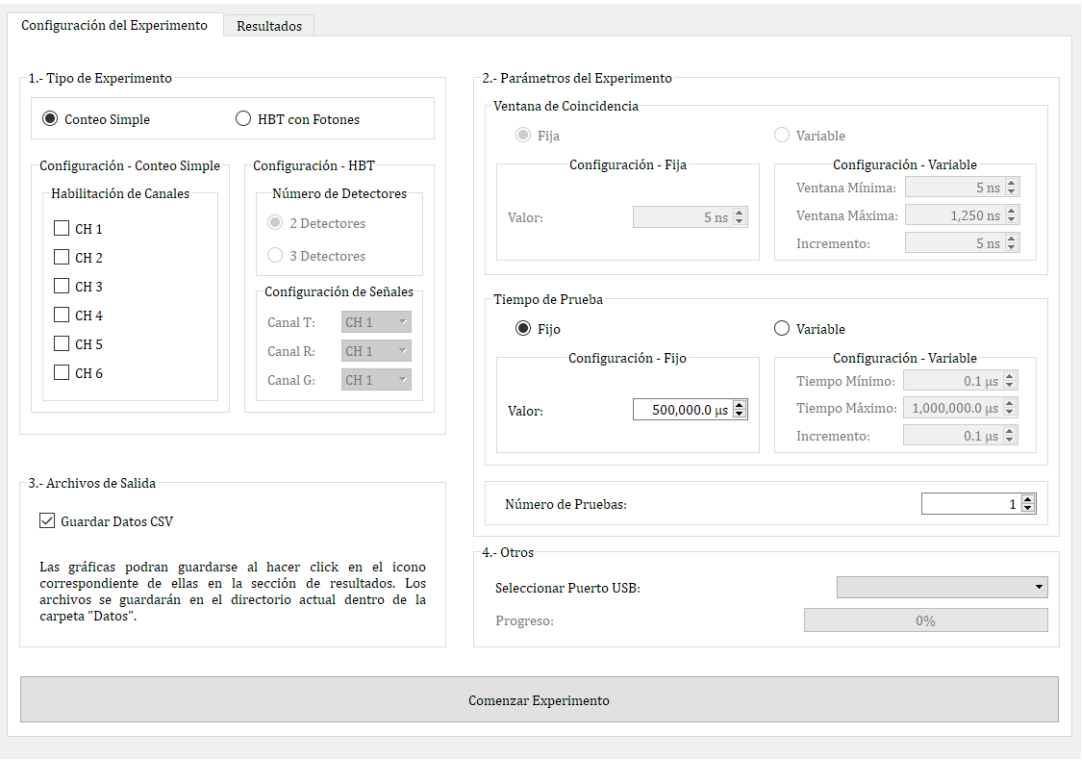

Figura 4.2: Ventana principal de la interfaz desarrollada.

4.4.1. Diagrama de Flujo

En la figura 4.8 que se encuentra al final de capítulo se muestra el diagrama de flujo que resume de manera conceptual el funcionamiento del software. Los paralelogramos naranjas corresponden a entradas que debe de proporcionar el usuario, mientras que los morados corresponden a salidas del programa. Es interesante observar que el programa está diseñado para realizar varios experimentos de forma sucesiva, por lo que el programa no finalizará a menos que el usuario lo indique explícitamente al cerrar la ventana correspondiente.

4.4.2. Experimentos de conteo simple

Para realizar experimentos de conteo simple de fotones se deben de realizar los siguientes pasos:

- Seleccionar la opción ‘Conteo Simple’ del apartado ‘Tipo de Experimento’.

- Seleccionar los canales de entrada en los cuales contaremos los pulsos procedentes de los APDs de la lista mostrada en el apartado 'Habilitación de Canales'.
- Seleccionar los parámetros a utilizar en el experimento dependiendo de si se desea realizar el conteo con un tiempo de prueba fijo o con un tiempo de prueba variable. La diferencia entre estas dos variaciones del experimento se describe en las secciones 4.4.2.1 y 4.4.2.2.
- Escoger el número de veces que se repetirá el conteo en el apartado 'Número de pruebas'.
- Seleccionar el nombre del puerto USB en el que se encuentra conectado el dispositivo de la lista mostrada en la sección 'Otros'.

Un ejemplo de las gráficas producidas por el software al realizar este tipo de experimentos se muestra en la figura 4.3. Los datos graficados fueron obtenidos mediante una simulación computacional en Python en la que se obtuvo una muestra de una de una variable aleatoria Poisson con parámetro $\mu = 75$.

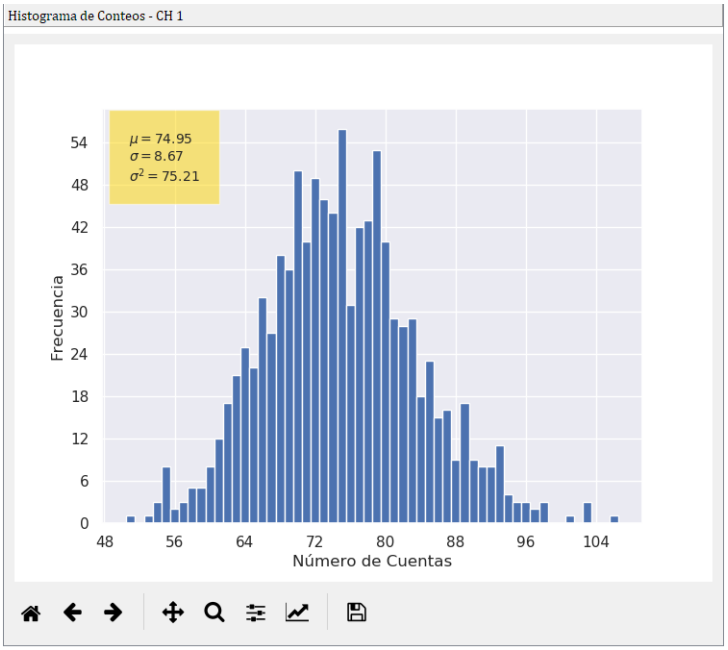

Figura 4.3: Gráfica producida al realizar un experimento de conteo.

A continuación se describe en qué consisten los experimentos de conteo con tiempo de prueba fijo y tiempo de prueba variable.

4.4.2.1. Experimentos con T_p fijo

Este tipo de experimentos consiste en contar el número de fotones detectados durante un tiempo T_p en los canales que se encuentren habilitados. Si el número de pruebas seleccionado es N_p en total se realizarán N_p conteos para cada canal, todos de manera simultánea.

Al finalizar en la sección de resultados se mostrará un histograma de frecuencias de los conteos obtenidos, junto con el cálculo de la media y la desviación estándar de la distribución observada para cada canal.

4.4.2.2. Experimentos con T_p variable

Al realizar este tipo de experimentos debemos de escoger el tiempo de prueba mínimo T_p^{min}, el tiempo de prueba máximo T_p^{max} y el incremento ΔT_p que se desea utilizar. Sea T^* el tiempo de prueba mas grande de la forma $T_p^{min} + n\Delta T_p$ tal que $T^* \leq T_p^{max}$. Si el número de pruebas seleccionadas es N_p entonces para cada tiempo de prueba dentro del conjunto $A = \{T_p^{min}, T_p^{min} + \Delta T_p, T_p^{min} + 2\Delta T_p, \ldots, T^*\}$ se realizarán N_p experimentos de conteo simple.

Análogamente a los experimentos realizados con T_p fijo, en la sección de resultados se mostrará un histograma de los conteos obtenidos junto con su media y su distribución estándar para cada canal habilitado, con la diferencia de que ahora podremos escoger de manera interactiva para qué tiempo de prueba queremos observar los resultados descritos. Esto es, en realidad podremos observar $|A|$ gráficas distintas para cada canal habilitado, una por cada tiempo de prueba dentro del conjunto A.

Este tipo de experimento es útil porque nos permite estudiar fácilmente la relación entre el tiempo de prueba utilizado y la variación en la estadística de fotones observada.

4.4.3. Experimento de HBT con fotones

Para realizar experimentos de HBT con fotones se deben de realizar los siguientes pasos:

- Seleccionar la opción 'HBT con fotones' del apartado 'Tipo de Experimento'.
- Seleccionar el número de detectores que se utilizarán en el experimento en el apartado 'Número de detectores'.

- Dependiendo de si se desea realizar un experimento con 2 o 3 detectores, indicar qué canal se encuentra asociado con los detectores testigo, reflejado y transmitido (G, R, o T respectivamente).
- Escoger el número de veces que se repetirá el experimento, esto con el propósito de tener un mejor estimador del valor real de $g^{(2)}(0)$ al promediar el valor obtenido para cada prueba individual.
- Seleccionar el nombre del puerto USB en el que se encuentra conectado el dispositivo de la lista mostrada en la sección 'Otros'.

A continuación se describe la diferencia entre los experimentos de HBT con fotones para los casos en los que V_c y T_p son ambos fijos o cuando alguno de ellos es variable.

4.4.3.1. Experimentos con V_c y T_p fijos

Cuando el valor de ambos parámetros es fijo entonces se realizarán N_p repeticiones del experimentos de HBT descrito en la sección 1.6.7, todas con los mismos valores de T_p y V_c.

Un ejemplo de las gráficas producidas en este tipo de experimentos se muestra en la figura 4.4. Debido a que en ese momento no se contaba con material adicional como un generador de pulsos, láseres o APDs la gráfica fue obtenida al agregar un módulo adicional al firmware que se encargaba de producir pulsos con un periodo de 25 ns cuya salida se conectó a todas las entradas del módulo Detector de Pulsos.

4.4.3.2. Experimentos con V_c variable

Cuando realizamos experimentos de este tipo debemos escoger el valor de la ventana de coincidencia mínima V_c^{min}, su valor máximo V_c^{max} y el incremento ΔV_c que se desea utilizar. Sea V^* el tiempo de prueba mas grande de la forma $V_c^{min} + n\Delta V_c$ tal que $V^* \leq V_c^{max}$. Si el número de pruebas seleccionadas es N_p entonces para cada ventana de coincidencia dentro del conjunto $A = \{V_c^{min}, V_c^{min} + \Delta V_c, V_c^{min} + 2\Delta V_c, \ldots, V^*\}$ se realizarán N_p experimentos de HBT, todos con el mismo tiempo de prueba T_p.

Este tipo de experimentos son interesantes por que nos permiten estudiar los cambios en el comportamiento de $g^{(2)}(0)$ al variar gradualmente la ventana de coincidencia.

Un ejemplo de las gráficas producidas en este tipo de experimentos se muestra en la figura 4.5. De forma análoga a la gráfica anterior, esta gráfica se obtuvo al utilizar el módulo auxiliar generador de pulsos.

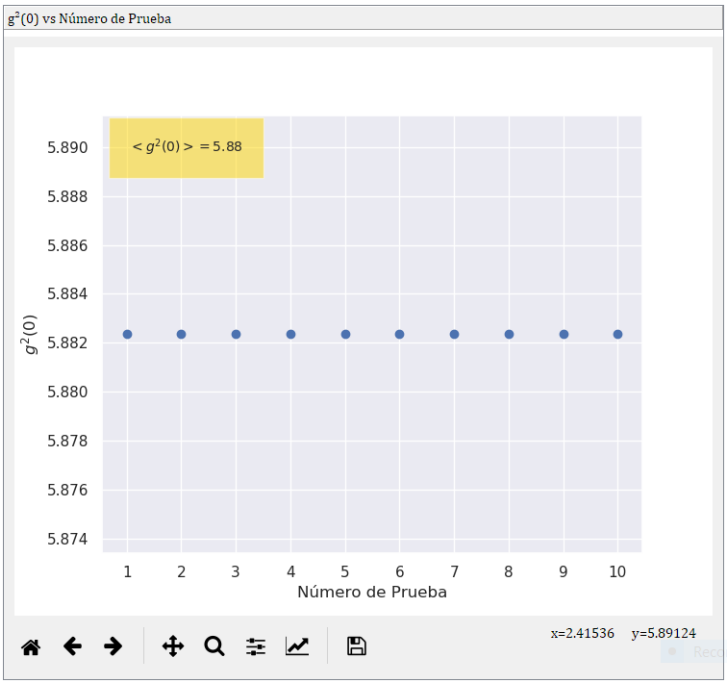

Figura 4.4: Gráfica producida al realizar un experimento de HBT con T_p y V_c fijos.

4.4.3.3. Experimentos con T_p variable

Al realizar este tipo de experimentos debemos de escoger el tiempo de prueba mínimo T_p^{min}, el tiempo de prueba máximo T_p^{max} y el incremento ΔT_p que se desea utilizar. Sea T^* el tiempo de prueba mas grande de la forma $T_p^{min} + n\Delta T_p$ tal que $T^* \leq T_p^{max}$. Si el número de pruebas seleccionadas es N_p entonces para cada tiempo de prueba dentro del conjunto $A = \{T_p^{min}, T_p^{min} + \Delta T_p, T_p^{min} + 2\Delta T_p, \ldots, T^*\}$ se realizarán N_p experimentos de HBT, todos con la misma ventana de coincidencia V_c.

Este tipo de experimentos son interesantes por que nos permiten estudiar los cambios en el comportamiento de $g^{(2)}(0)$ al variar gradualmente el tiempo de prueba.

Un ejemplo de las gráficas producidas en este tipo de experimentos se muestra en la figura 4.6. De forma análoga que con las gráficas anteriores, esta gráfica se obtuvo al utilizar el módulo auxiliar generador de pulsos.

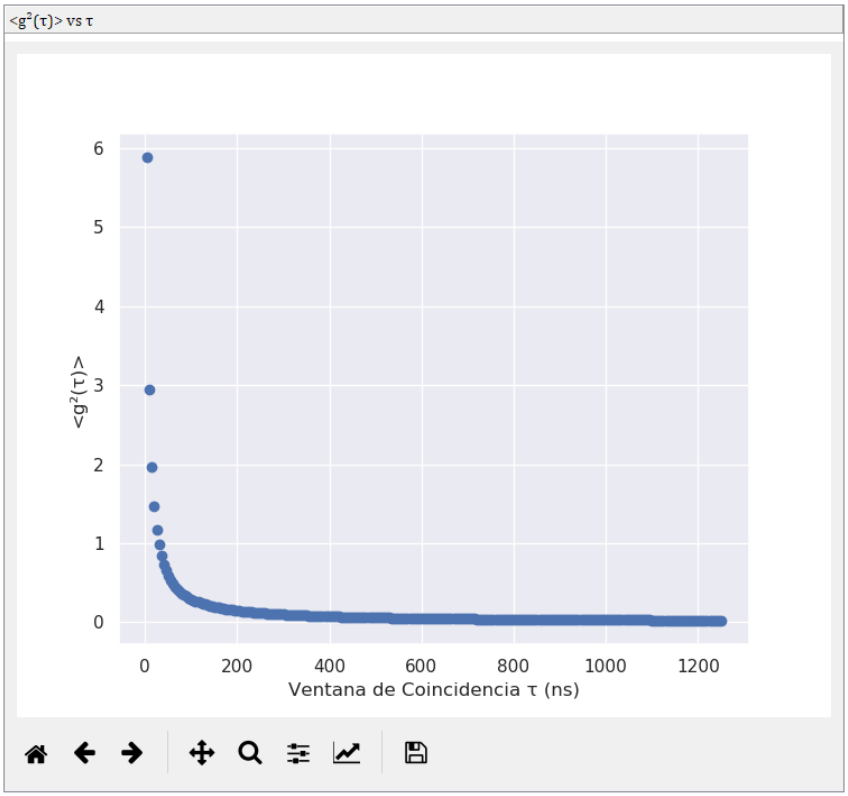

Figura 4.5: Gráfica producida al realizar un experimento de HBT con V_c variable.

4.4.4. Almacenamiento de datos

Si esta opción se encuentra habilitada, al terminar cada experimento el software creará de forma automática los archivos de datos con formato *.csv* correspondientes al tipo de experimento realizado.

Sin importar si se realizó un experimento de conteo o de HBT se creará automáticamente una estructura de folders a partir de la fecha y hora en que se haya realizado. Un ejemplo de la estructura de folders creada por el software se muestra en la figura 4.7.

En el caso de haber realizado un experimento de conteo simple se creará un único archivo de datos con el nombre 'Cuentas' cuyas columnas contienen a los conteos registrados en cada canal habilitado. En el caso de realizar un experimento de conteo con T_p variable se crearán varios de estos archivos, uno por cada tiempo de prueba utilizado. Para poder distinguir qué archivo de datos le corresponde a cada tiempo de prueba éste se incluye en el nombre de los archivos generados.

En el caso de realizar un experimento de HBT con T_p y V_c fijos el archivo de datos contendrá en sus columnas a las cuentas y coincidencias relevantes para

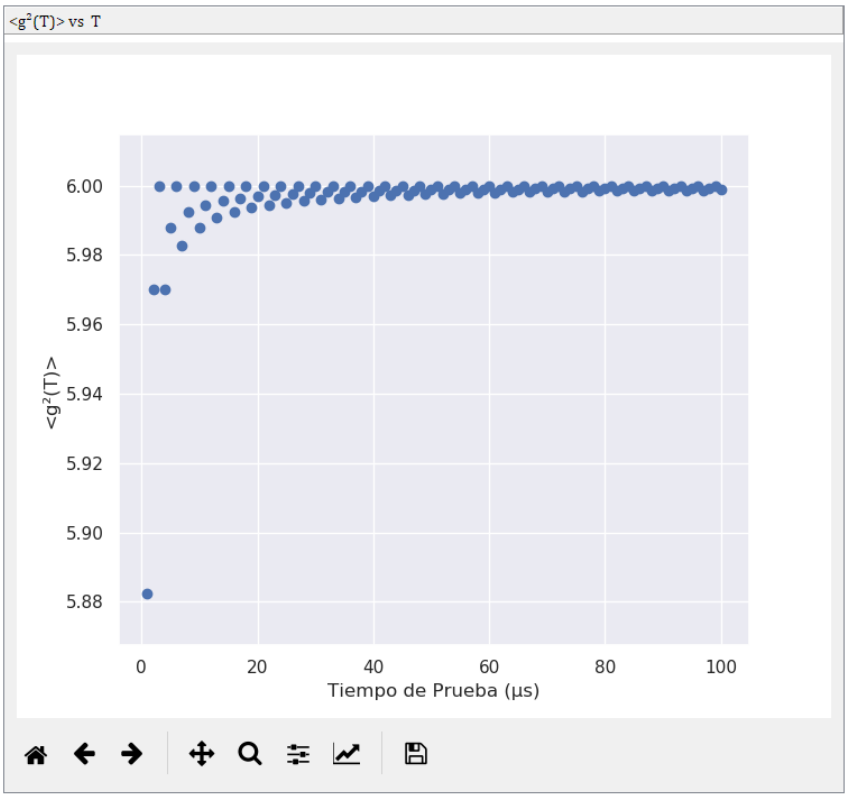

Figura 4.6: Gráfica producida al realizar un experimento de HBT con T_p variable.

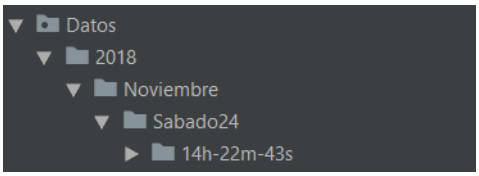

Figura 4.7: Estructura de folders creada por el software.

hallar el valor de $g^2(0)$ junto con su valor calculado por el software. Si se realiza un experimento de HBT con T_p o V_c variable el software creará varios de estos archivos, uno por cada pareja (T_p, V_c) utilizada en el experimento, y el nombre de los archivos será modificado de tal forma que sea posible distinguir qué archivo le corresponde a cada pareja.

4.5. Conclusión

La interfaz desarrollada servirá para automatizar los experimentos de conteo de fotones y el experimento de HBT que suelen llevarse a cabo de manera regular en el Laboratorio de Óptica Avanzada de la Facultad de Ciencias, UNAM. Sin embargo, la interfaz desarrollada no es general en el sentido de que se diseñó teniendo en mente estos dos experimentos particulares. La integración de la librería Matplotlib con PyQt para poder producir las gráficas mostradas presentó algunas complicaciones, pero junto con la librería Seaborn la flexibilidad y la apariencia visual de las gráficas resultantes lo tienen bien justificado.

Como trabajo a futuro, se propone crear una nueva interfaz más general que la desarrollada en el presente trabajo que le entregue al experimental un control total sobre el dispositivo, lo que serviría para ampliar considerablemente el rango de experimentos para los que es utilidad. Sin embargo, en caso de realizar la interfaz de esta nueva manera la tarea de automatizar el experimento de interés estaría en las manos del usuario.

Debido a las herramientas seleccionadas para desarrollar el software y al enfoque modular con que se programó la interfaz esto no debería de ser una tarea difícil para alguien con conocimientos en el lenguaje de programación Python, siendo este lenguaje uno de los mas utilizados actualmente y uno de los principales lenguajes en los que se suele enseñar a usuarios inexperimentados a programar por primera vez.

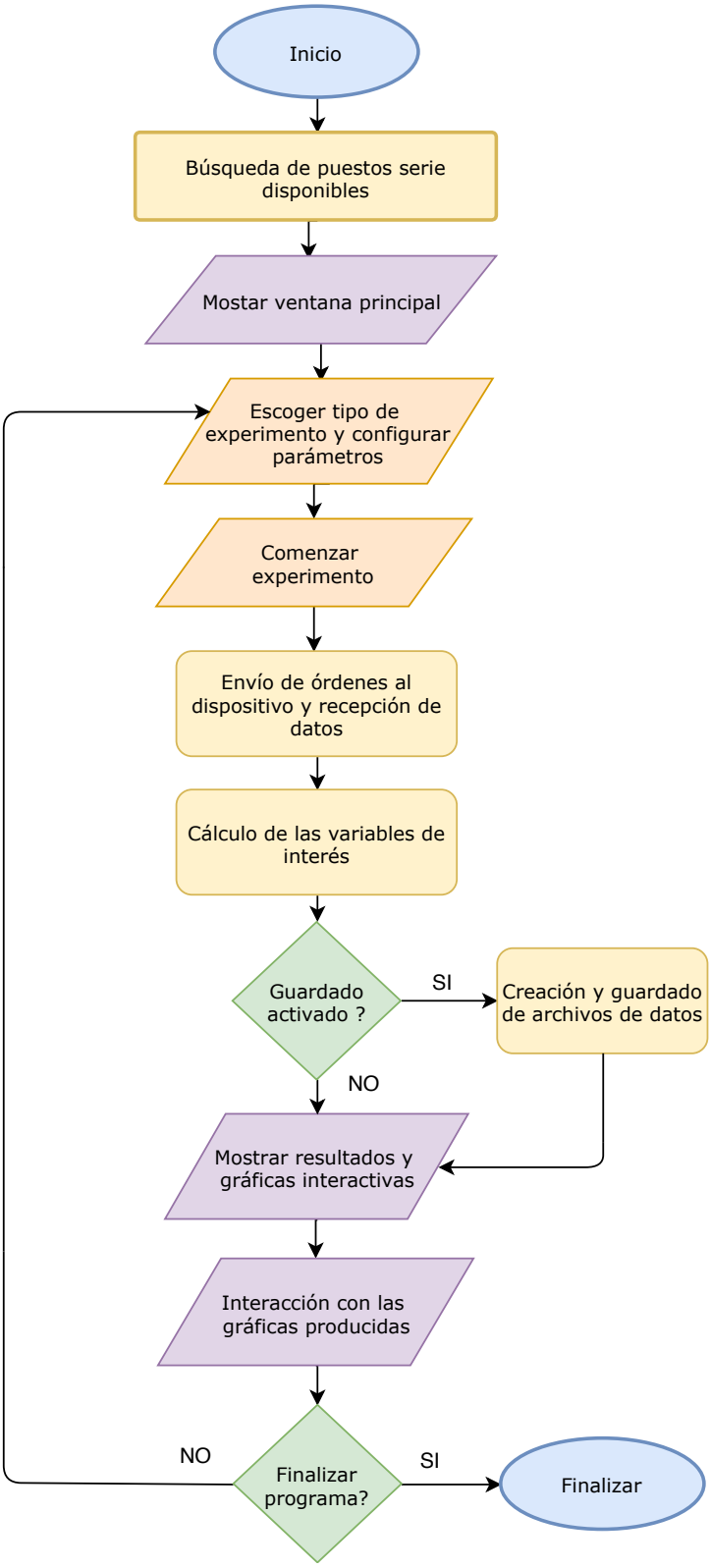

Figura 4.8: Diagrama de flujo de la interfaz desarrollada.

Capítulo 5

Resultados

En esta sección se muestran las simulaciones y pruebas experimentales realizadas para verificar el correcto funcionamiento del software, hardware y firmware desarrollados. En resumen, las simulaciones muestran que el comportamiento de los módulos involucrados en las tareas de conteo de pulsos y detección de coincidencias coincide con lo esperado, mientras que en la práctica el error porcentual máximo obtenido entre los conteos esperados y los observados al utilizar un generador de señales y el dispositivo desarrollado fue de 0.0006 %. Debido a la falta de tiempo y de material apropiado no fue posible realizar una prueba experimental objetiva y rigurosa del funcionamiento de los módulos detectores de coincidencias, pero los resultados observados al realizar pruebas de carácter exploratorio con el material disponible parecen indicar que este módulo también funciona de forma correcta.

5.1. Simulaciones

Las simulaciones siguientes se realizaron con ayuda del software Quartus ll de Altera que permite simular el desempeño lógico de los módulos que componen al firmware, y su propósito es mostrar que la relación entre las señales lógicas de entrada y salida de los módulos es consistente con lo que se describió en el capítulo 3. Se realizaron varias simulaciones para verificar el comportamiento de cada uno de los módulos individuales, pero por cuestiones de espacio únicamente se muestran las simulaciones correspondientes a dos circuitos que fueron diseñados para estudiar el desempeño conjunto de éstos al conectarlos de tal forma que se pudieran emular las tareas de conteo de pulsos y detección de coincidencias.

5.1.1. Conteo de pulsos

En la figura 5.1 se muestra el diagrama por bloques del circuito diseñado para probar el funcionamiento de los módulos involucrados en el conteo de pulsos.

El valor del bus de entrada *EnableCHS* se fijó en 0b000111 por lo que sólo se habilitaron los canales 1, 2 y 3, mientras que el valor del bus de entrada *Test_Time* se fijó en 0x50 (80 decimal). Adicionalmente se introdujo una señal de reloj con periodo de 5 ns y ciclo de trabajo del 50 % en la terminal *Clk*, por lo que al iniciar al módulo Generador de Tiempo de Prueba se simula un experimento de conteo con una duración de 400 ns.

Los resultados se muestran en la figura 5.11, que se encuentra al final del capítulo.

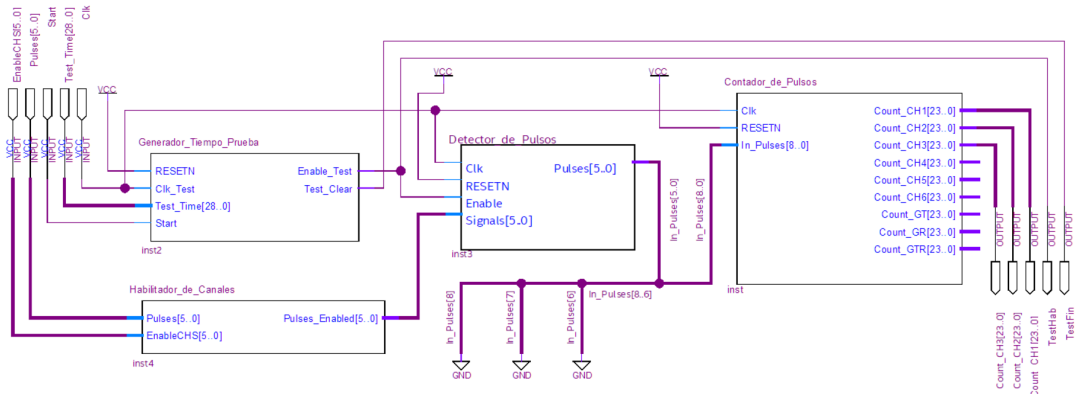

Figura 5.1: Circuito utilizado en la simulación de conteo de pulsos.

5.1.2. Detección y conteo de coincidencias

En la figura 5.2 se muestra el diagrama por bloques del circuito diseñado para probar el funcionamiento de los módulos involucrados en el proceso de detección y conteo de coincidencias.

De forma similar a la simulación pasada se fijó el valor de los buses de entrada *EnableCHS* y *Test_Time* con los valores 0b000111 y 0x50 respectivamente, y en la entrada *Clk* se introdujo una señal de reloj con un periodo de 5 ns y ciclo de trabajo del 50 %. Se utilizaron valores de 5, 10 y 15 ns para la ventana de coincidencia. Los resultados obtenidos se muestran en la figura 5.11 que se encuentra al final del capítulo.

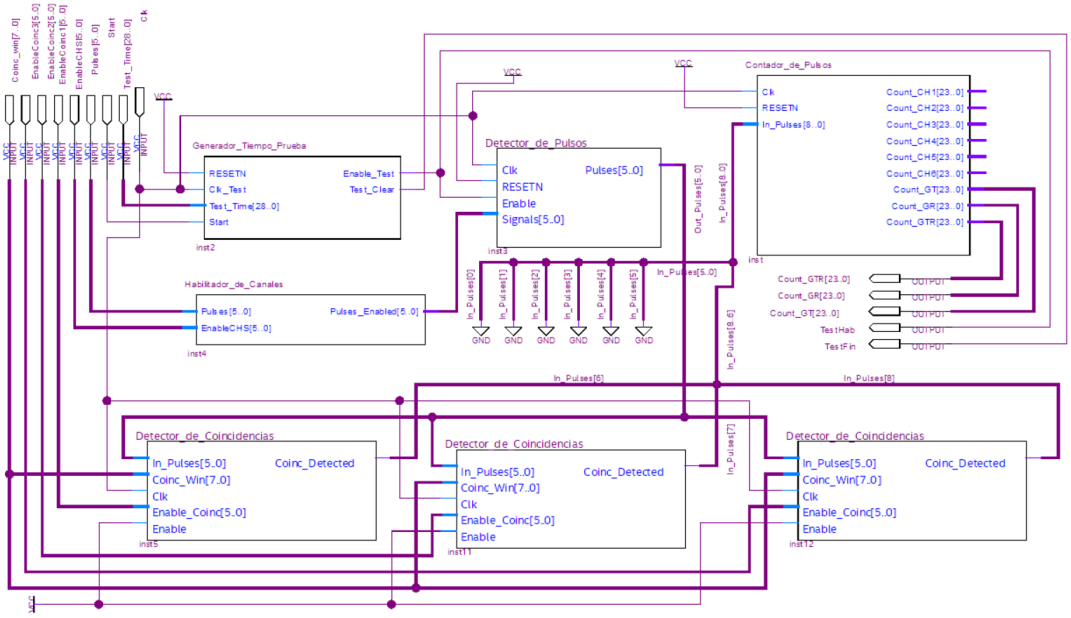

Figura 5.2: Circuito utilizado en la simulación de la detección y conteo de coincidencias.

5.2. Pruebas Experimentales

5.2.1. Conteo de pulsos

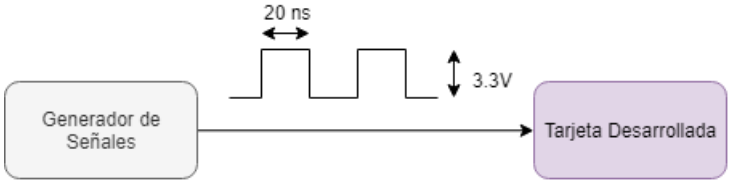

Figura 5.3: Diagrama experimental utilizado para las pruebas de conteo de pulsos.

Se configuró un generador de ondas arbitrarias Agilent 33220A (45) para producir un tren de pulsos con una amplitud de 3.3 V y un ancho de pulso de 20ns. Esta señal se inyectó en un canal del dispositivo y se realizó un experimento de conteo simple con 10 repeticiones y duración de un segundo para cada frecuencia mostrada en la tabla 5.1. Esto corresponde a realizar un experimento de conteo simple con los parámetros $T_p = 1$ s y $N_p = 10$ con la interfaz desarrollada para cada una de ellas.El diagrama experimental correspondiente se muestra en la figura 5.3, y los resultados obtenidos en la tabla 5.1.

Frecuencia (Hz)	Cuentas promedio	Desviación	Error	Error porcentual
100	100	0	0	0 %
500	500	0	0	0 %
1×10^3	1×10^3	0	0	0 %
5×10^3	5×10^3	0	0	0 %
1×10^4	1×10^4	0	0	0 %
5×10^4	49999.8	0.49	0.2	4×10^{-4} %
5×10^5	99999.4	0.49	0.6	6×10^{-4} %
5×10^5	499997.2	0.49	2.8	6×10^{-4} %
1×10^6	999994.4	0.49	5.6	5.6×10^{-4} %
5×10^6	4999972.4	0.49	27.6	5.5×10^{-4} %

Tabla 5.1: Resultados experimentales de pruebas de conteo (10 repeticiones).

5.2.2. Medición del tiempo de prueba

Se modificó el firmware para que la salida *Enable_Test* del módulo Generador de Tiempo de Prueba también fuera asignada a una de las terminales del FPGA y fuera posible observar con el osciloscopio el tiempo durante el cual esta terminal se mantiene en alto, donde se considera que la señal está en alto cuando su valor es mayor a el 50 % de su amplitud máxima. Se realizó esta medición para tiempos de prueba de 0.1μs, 1μs y 2.5μs. El diagrama experimental correspondiente se muestra en la figura 5.4 y los resultados obtenidos en la figuras 5.5, 5.6 y 5.7.

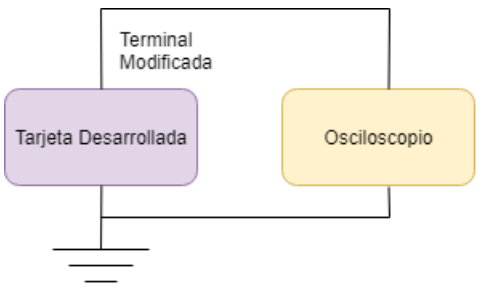

Figura 5.4: Diagrama experimental utilizado para la medición del tiempo de prueba.

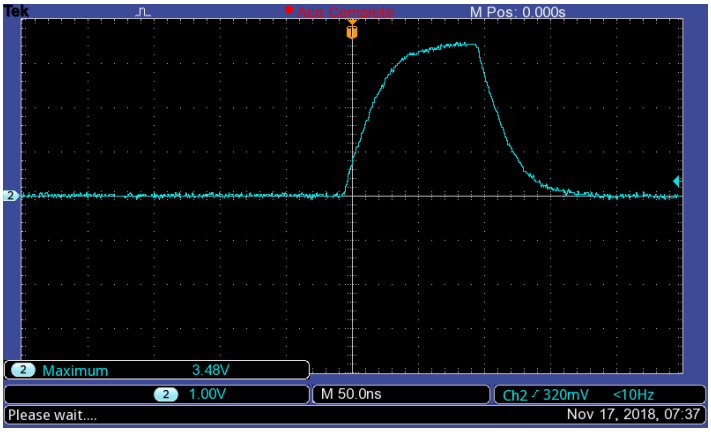

Figura 5.5: Tiempo de prueba generado para $T_p = 0.1\mu$s.

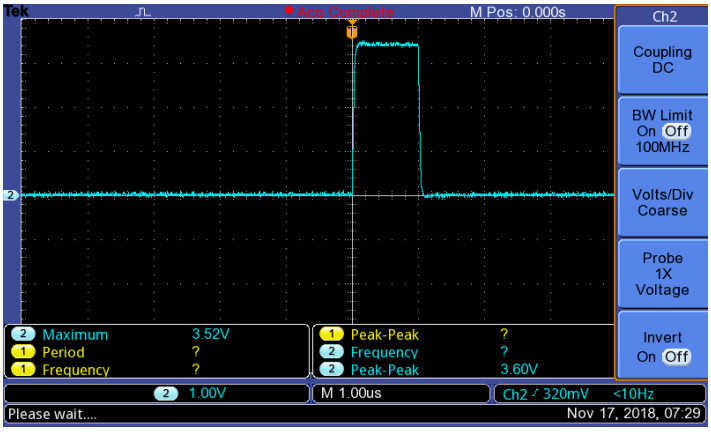

Figura 5.6: Tiempo de prueba generado para $T_p = 1\mu$s.

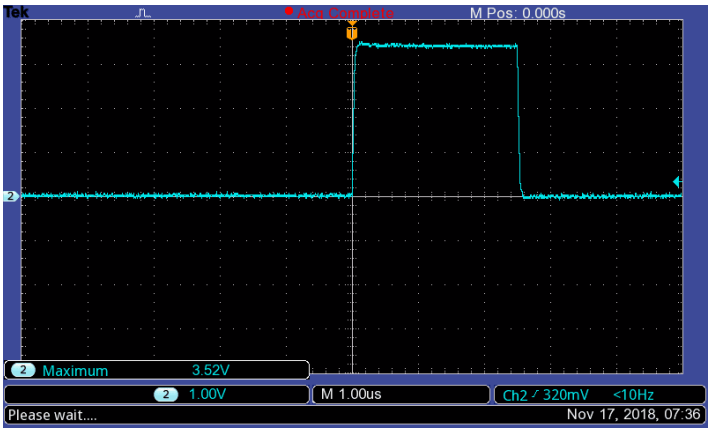

Figura 5.7: Tiempo de prueba generado para $T_p = 2.5$s.

En la figura 5.5 se puede observar que el tiempo durante el cual el pulso tiene una amplitud mayor que el 50 % de su amplitud máxima es de aproximadamente 100 ns (0.1 μs). A este último tiempo se le suele conocer como el ancho de pulso, por lo que esta figura muestra que el pulso producido por el FPGA tiene el ancho deseado. Adicionalmente en las figuras 5.6 y 5.7 se muestra que para tiempos de prueba mayores los tiempos de transición entre estados son cada vez menores respecto a la duración total de la señal, por lo que el pulso producido se asemeja cada vez más a un pulso cuadrado.

5.2.3. Detección de coincidencias

En esta prueba se configuró el generador de señales para producir pulsos con una frecuencia de 100 Hz. Se utilizaron diversos cables de diferente longitud que se encontraban etiquetados con el tiempo que le toma a una señal propagarse por cada uno de ellos. No fue posible medir este tiempo experimentalmente con el osciloscopio (Tektronix TDS1102BEDU) que se tenía debido a que el dispositivo desarrollado tiene una impedancia de entrada de 50 Ω y el osciloscopio utilizado cuenta con una impedancia de entrada de $1M\Omega$, por lo que la señal que se pudiera visualizar en el osciloscopio no sería una representación fiel de la señal que se alimentaría al dispositivo.

Se utilizó un conector tipo T para conectar dos de estos cables a la salida del generador y a 2 canales del dispositivo. El diagrama experimental correspondiente se muestra en la figura 5.8.

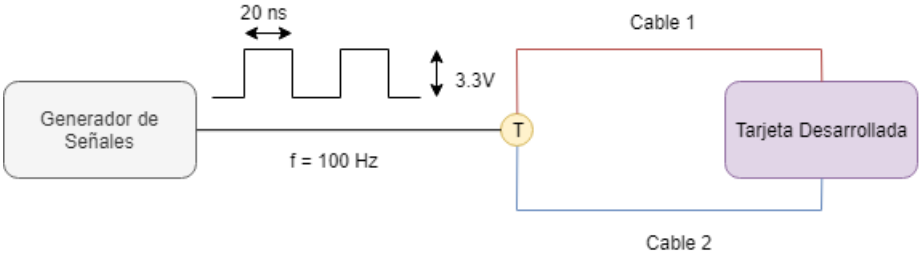

Figura 5.8: Diagrama experimental utilizado para las pruebas de detección de coincidencias.

Se realizó un experimento de conteo de coincidencias con duración de un segundo para dos diferentes configuraciones de cables. Las configuraciones utilizadas y los resultados obtenidos se muestran en la tabla 5.2.

Retraso		Coincidencias		
Cable 1	Cable 2	Vc = 5ns	Vc = 10ns	Vc = 15ns
5ns	8ns	33	100	100
5ns	10ns	0	34	100

Tabla 5.2: Resultados de pruebas de conteo y detección de coincidencias.

Estos resultados muestran que la eficiencia del proceso de detección de coincidencias para ventanas de 5ns es de aproximadamente 33 %. Sin embargo en este momento no se tiene la información suficiente para poder concluir que esto sea una consecuencia del diseño elaborado o de la señal que se inyectó al dispositivo, esto debido a la falta de equipo apropiado para realizar pruebas adicionales.

Aún así, se puede observar que la eficiencia del proceso de detección de coincidencias para ventanas con una duración de al menos 10 ns es del 100 %.

5.2.4. Estadística de fotones

Se hizo incidir un láser ultravioleta de 495 mW sobre un cristal no lineal de beta borato de bario (BBO). El haz resultante se hizo incidir sobre un APD que se conectó al dispositivo desarrollado, y se realizó un experimento de conteo simple con los parámetros $T_p = 100$ μs y $N_p = 1500$. En la figura 5.10 se muestra el histograma de los conteos obtenidos junto con los respectivos valores del promedio, la varianza y la desviación estándar. Estos valores fueron calculados de forma automática por el software. El diagrama experimental correspondiente se muestra en la figura 5.9.

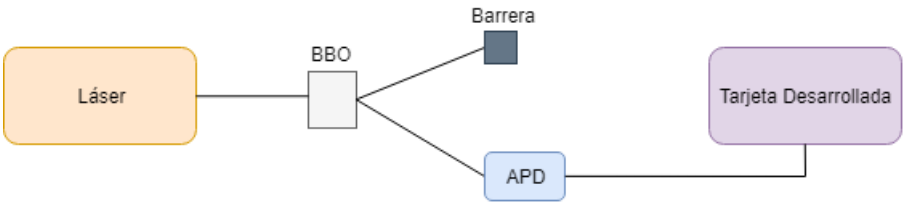

Figura 5.9: Diagrama experimental utilizado en la prueba de estadística de fotones.

Se utilizó esta configuración debido a que en ese momento se estaba realizando un experimento en el Laboratorio de Óptica Avanzada y fue la forma mas rápida y sencilla de probar el funcionamiento del dispositivo sin interferir con el

experimento. Sin embargo, teóricamente el haz de fotones sobre el cual se realizo la estadística conserva la distribución de la fuente original (Poisson).

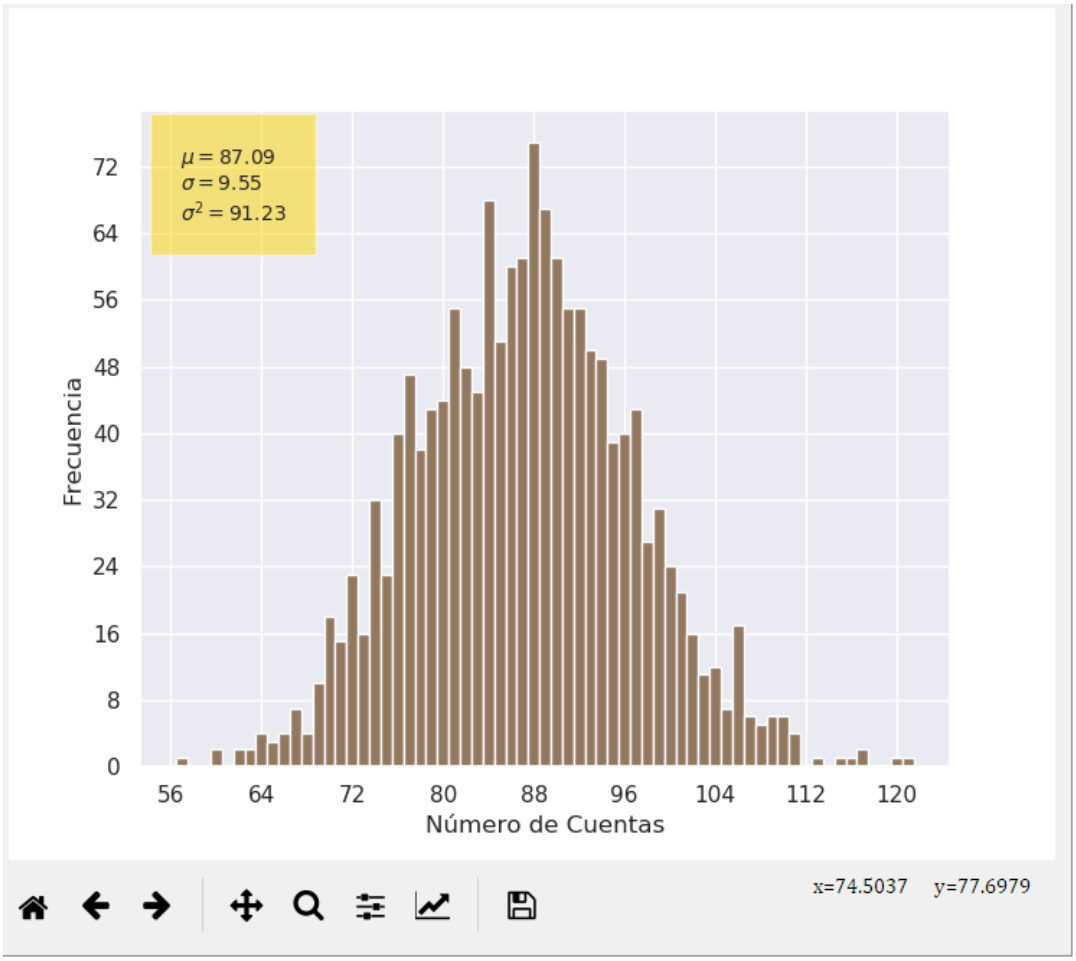

Figura 5.10: Estadística de fotones del arreglo experimental mostrado en la figura 5.9.

5.3. Conclusión

Los experimentos descritos en este capítulo muestran que los módulos encargados de realizar la comunicación UART con el microcontrolador funcionan correctamente. Debido a que las simulaciones lógicas de los módulos desarrollados muestran que éstos funcionan de acuerdo a lo esperado, se propone que el error porcentual obtenido al realizar experimentos de conteo de pulsos se debe a que aún es necesario realizar una etapa adicional conocida como *Timing Closure*,

cuyo propósito es modificar el proceso de compilación utilizado por el software para cumplir con todos los requisitos de sincronización especificados. Más acerca de esto se discute en el capítulo 6.

Para poder realizar pruebas mas rigurosas acerca del funcionamiento del Módulo Detector de Coincidencias es necesario utilizar un osciloscopio cuya impedancia se encuentre acoplada con la de los conectores utilizados por la tarjeta desarrollada (50 Ω), así como medir experimentalmente el tiempo de propagación de los cables utilizados. Esto es necesario para que la señal observada en el osciloscopio sea una representación fiel de la señal que se está introduciendo en los canales de la tarjeta, y así poder distinguir si los resultados mostrados en este capítulo se deben a un funcionamiento incorrecto del módulo para $V_c = 5$ ns o a que la forma de la señal utilizada no coincide con lo esperado. También existe la posibilidad de que estos resultados se deban a una sincronización incorrecta de los módulos internos, misma que podría solucionarse al completar el proceso de Timing Closure.

Todos estos experimentos se realizaron con ayuda del software desarrollado en Python, por lo que estos resultados también son útiles para comprobar el correcto funcionamiento de éste. Aún es necesario realizar más experimentos para caracterizar completamente el estado de funcionamiento actual del dispositivo. Sin embargo, experimentos subsecuentes realizados en el Laboratorio de Óptica Avanzada por diversos estudiantes e investigadores han arrojado resultados similares a los obtenidos con las tarjetas comerciales que se utilizaban anteriormente, que sirven como una evidencia indirecta del adecuado funcionamiento del dispositivo desarrollado.

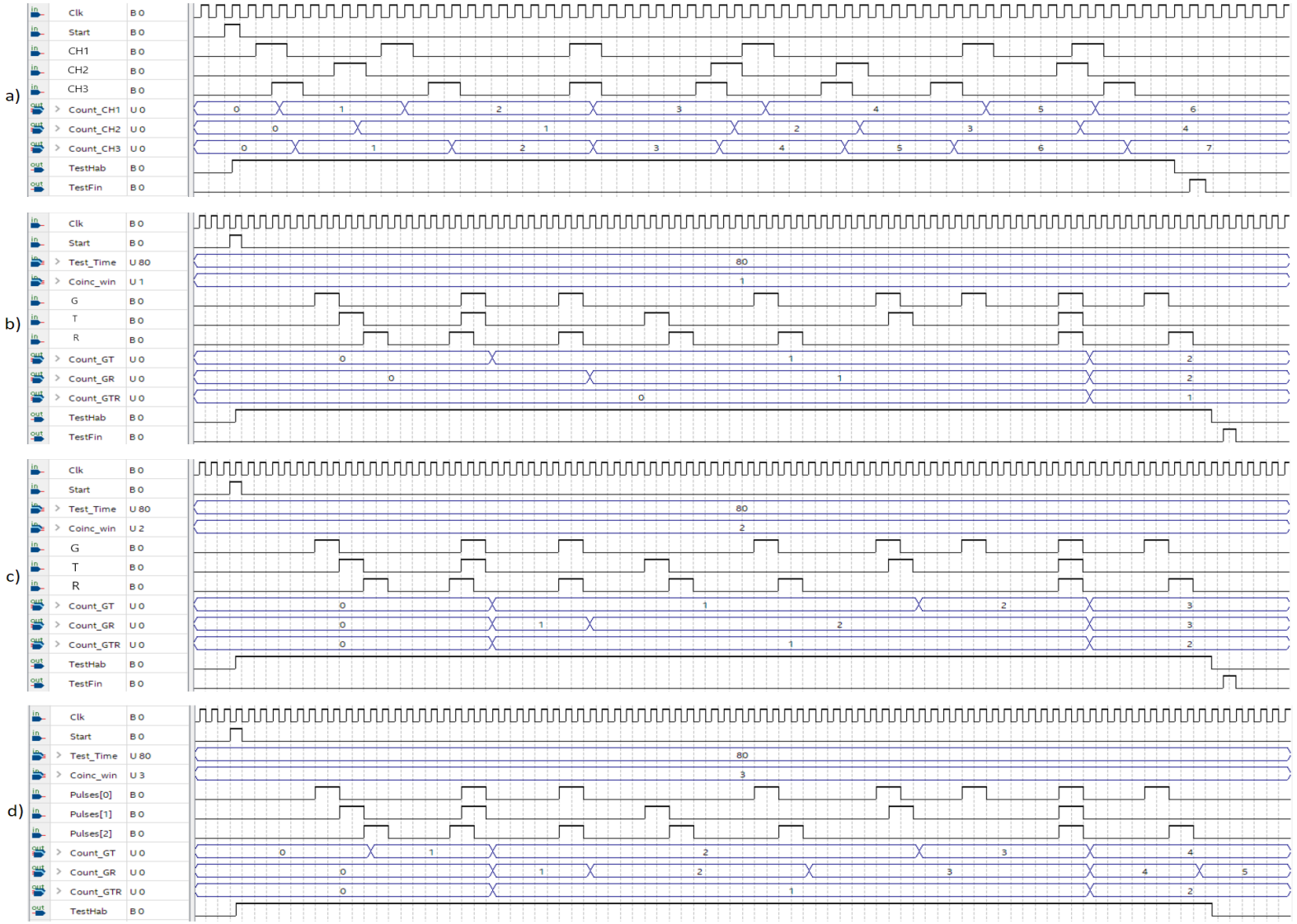

Figura 5.11: Resultados de las simulaciones. a) Conteo de pulsos. b), c) y d) Detección y conteo de coincidencias para $V_c =$ 5ns, 10ns y 15ns respectivamente.

Capítulo 6

Conclusiones

Se pudo observar que el software, el hardware y el protocolo de comunicación UART establecido entre el dispositivo y la computadora funcionan de la forma esperada. En lo que respecta a el proceso de conteo de pulsos se obtuvo un error porcentual máximo del 0.0006 %. Sin embargo, los resultados obtenidos para el proceso de detección de coincidencias muestran que es necesario realizar pruebas adicionales con el propósito de verificar el correcto funcionamiento del dispositivo al utilizar una ventana de coincidencia de 5ns. Estas pruebas se proponen como trabajo a futuro debido a la falta de tiempo y de material adecuado. Aún así se pudo observar que en su estado actual el dispositivo es capaz de realizar este proceso de forma correcta para ventanas de coincidencia de al menos 10 ns.

El lenguaje de programación Python y la multitud de librerías disponibles fueron muy útiles para poder agilizar el desarrollo de este proyecto. Además, al ser un lenguaje de código abierto el usuario es capaz de programar cualquier experimento de interés de una forma amigable y distribuir el código fuente de forma libre y gratuita. Además Python tiene la ventaja de ser un lenguaje multiplataforma, por lo que el código desarrollado puede ejecutarse en casi todos los sistemas operativos sin necesidad de modificar el código fuente.

La presencia de la tarjeta de desarrollo Teensy 3.2 abre una gran cantidad de posibilidades que pueden desarrollarse a futuro como la implementación de protocolos de comunicación inalámbricos, la incorporación de pantallas y dispositivos de entrada y salida para poder configurar el dispositivo sin la necesidad de contar con una computadora y procesar los resultados obtenidos directamente en el dispositivo, entre otras.

La principal limitante para poder reducir aún más la ventana de coincidencia mínima y lograr que el dispositivo funcione sin error alguno es el hecho de que aún es necesario realizar un procedimiento conocido como *Timing Closure*, el cual consiste en toda una metodología para modificar el proceso de compilación del diseño de tal forma que todas las señales dentro del dispositivo cumplan con las

especificaciones deseadas.

Este procedimiento es necesario debido a que al utilizar lenguajes de descripción de hardware como Verilog o VHDL el compilador tiene la tarea de traducir este código a su representación equivalente en compuertas lógicas y realizar el mapeo y la optimización relevante, para que al implementar físicamente el diseño en el dispositivo y tomar en cuenta distintos factores como la velocidad de propagación de las señales dentro de la tarjeta y la velocidad de respuesta de sus compuertas, entre otras, las señales se encuentren correctamente sincronizadas y el desempeño físico del diseño sea el mismo que el observado en las simulaciones lógicas. Por ejemplo, si este procedimiento no es realizado es posible que el mismo diseño implementado en dos diferentes tipos de tarjetas funcione de manera distinta.

Para especificarle al compilador las constricciones de sincronización y los demás parámetros de operación relevantes se utilizan archivos con extensión *.sdc (Synopsis Design Constraints)*. Debido a que se necesitan conocimientos técnicos que van mas allá del alcance de este trabajo esta tarea se propone como trabajo a futuro.

En conclusión, se logró construir un dispositivo de instrumentación de bajo costo capaz de realizar conteo de pulsos y detección de coincidencias con una ventana mínima de 10 ns. Aunque el dispositivo se diseñó con el propósito de utilizarse en experimentos de óptica cuántica, éste puede ser de utilidad en cualquier proceso que requiera contar pulsos o que dependa del retraso entre distintas señales, como los mencionados en 1.4. En lo personal me sorprendió lo mucho que se puede lograr en el área de la instrumentación científica al utilizar las herramientas adecuadas, el costo de los dispositivos desarrollados con esta metodología en comparación con los disponibles en el mercado, y la flexibilidad que le otorga al físico experimental el ser capaz de desarrollar sus propios instrumentos para perseguir sus objetivos de investigación.

Bibliografía

[1] Hammatsu, "Single pixel photon counting module for low-light-level detection." https://www.hamamatsu.com/jp/en/product/type/C11202-100/index.html. Consultado: 29-Ene-2019. XVII

[2] LaserComponents, "Single pixel photon counting module for low-light-level detection." https://www.lasercomponents.com/de-en/photon-counter/single-photon-counting-modules/. Consultado: 29-Ene-2019. XVIII

[3] ThorLabs, "Single pixel photon counting module for low-light-level detection." https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=5255. Consultado: 29-Ene-2019. XVIII

[4] PicoQuant, "Single pixel photon counting module for low-light-level detection." https://www.picoquant.com/products/category/photon-counting-instrumentation. Consultado: 29-Ene-2019. XVIII

[5] IDQuantique, "Single pixel photon counting module for low-light-level detection." https://www.idquantique.com/single-photon-systems/products/id100/. Consultado: 29-Ene-2019. XVIII

[6] P. Quant, "Timeharp200 specifications." https://sites.fas.harvard.edu/~phys191r/Bench_Notes/D4/TimeHarp200.pdf. Consultado: 30-Ene-2019. XVIII

[7] P. Quant, "Timeharp260 specifications." https://www.picoquant.com/images/uploads/downloads/timeharp260.pdf. Consultado: 30-Ene-2019. XVIII

[8] P. Solutions, "Dpc-230 photon correlator." https://www.photonicsolutions.co.uk/product-detail.php?prod=6607. Consultado: 30-Ene-2019. XIX

[9] S. R. Systems, "Sr 400 gated photon counter." https://www.thinksrs.com/products/sr400.html. Consultado: 19-Mar-2019. xx

[10] D. J. Griffiths, *Introduction to electrodynamics; 4th ed.* Boston, MA: Pearson, 2013. Re-published by Cambridge University Press in 2017. 1, 2, 3, 4

[11] M. Fox, *Quantum optics: an introduction*, vol. 15. OUP Oxford, 2006. 3, 4, 6, 7, 13, 16, 18, 19, 20, 21

[12] R. Photonics, "Spontaneous emission." https://www.rp-photonics.com/spontaneous_emission.html. Consultado: 26-Sep-2018. 8

[13] J. Branson, "The lamb shift." https://quantummechanics.ucsd.edu/ph130a/130_notes/node476.html. Consultado: 26-Sep-2018. 8

[14] E. Hecht, *Optics - Global Edition*. Pearson, 2017. 8

[15] I. Prochazka and F. Yang, "Photon counting module for laser time transfer via earth orbiting satellite," *Journal of Modern Optics*, vol. 56, no. 2-3, pp. 253–260, 2009. 9

[16] W. Becker, A. Bergmann, and C. Biskup, "Multispectral fluorescence lifetime imaging by tcspc," *Microscopy research and technique*, vol. 70, no. 5, pp. 403–409, 2007. 9

[17] E. Technologies, "Single photon counting modules." http://www.excelitas.com/Pages/Product/Single-Photon-Counting-Modules-SPCM.aspx. Consultado: 22-Ene-2019. 10

[18] E. Hergert, "Detectors: Guideposts on the road to selection." https://www.photonics.com/Articles/Detectors_Guideposts_on_the_Road_to_Selection/a25535. Consultado: 22-Mar-2019. 11

[19] R. Short and L. Mandel, "Observation of sub-poissonian photon statistics," *Physical Review Letters*, vol. 51, no. 5, p. 384, 1983. 14

[20] R. H. Brown and R. Twiss, "A test of a new type of stellar interferometer on sirius," *Nature*, vol. 178, no. 4541, pp. 1046–1048, 1956. 14, 16

[21] C. Chant, "Betelgeuse: How its diameter was measured," *Journal of the Royal Astronomical Society of Canada*, vol. 15, p. 133, 1921. 15

[22] R. H. Brown and R. Q. Twiss, "Correlation between photons in two coherent beams of light," *Nature*, vol. 177, no. 4497, pp. 27–29, 1956. 18

[23] R. Loudon, *The quantum theory of light*. OUP Oxford, 2000. 20

[24] J. Thorn, M. Neel, V. Donato, G. Bergreen, R. E. Davies, and M. Beck, "Observing the quantum behavior of light in an undergraduate laboratory," *American journal of physics*, vol. 72, no. 9, pp. 1210–1219, 2004. 20, 24

[25] H. J. Kimble, M. Dagenais, and L. Mandel, "Photon antibunching in resonance fluorescence," *Physical Review Letters*, vol. 39, no. 11, p. 691, 1977. 23

[26] L. Mandel and E. Wolf, *Optical coherence and quantum optics*. Cambridge university press, 1995. 24

[27] PJRC, "Teensy technical specifications." https://www.pjrc.com/teensy/techspecs.html. Consultado: 26-Ene-2019. 30, 32

[28] Arduino, "Arduino - home." https://www.arduino.cc/. Consultado: 23-Ene-2019. 32

[29] T. Instruments, "Launchpad development kits." http://www.ti.com/tools-software/launchpads/overview.html. Consultado: 23-Ene-2019. 32

[30] PJRC, "Teensy usb development board - pjrc." https://www.pjrc.com/teensy/. Consultado: 23-Ene-2019. 32

[31] E. Micro, "Mojo: Digital design for the hobbyist by embedded micro — kickstarter." https://www.kickstarter.com/projects/1106670630/mojo-digital-design-for-the-hobbyist?lang=es. Consultado: 23-Ene-2019. 35

[32] Intel, "Intel fpga products." https://www.intel.com/content/www/us/en/products/programmable/fpga.html. Consultado: 23-Ene-2019. 35

[33] Intel, "Cyclone ll device handbook." https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyc2/cyc2_cii5v1.pdf. Consultado: 26-Ene-2019. 35

[34] T. Inc, "Terasic - de main boards - de0-nano-soc kit/atlas-soc kit." https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=941. Consultado: 23-Ene-2019. 35

[35] LEMO, "Epk.00.250.ntn - technical details." https://www.lemo.com/pdf/EPK.00.250.NTN.pdf. Consultado: 23-Ene-2019. 36

[36] Intel, "Introduction to intel fpga ip cores." https://www.intel.com/content/www/us/en/programmable/documentation/mwh1409960636914.html. Consultado: 22-Ene-2019. 42

[37] QUANTIL, "Data transmission – parallel vs serial." https://www.quantil.com/content-delivery-insights/content-acceleration/data-transmission/. Consultado: 22-Ene-2019. 43

[38] P. S. Foundation, "Welcome to python." https://www.python.org/. Consultado: 22-Ene-2019. 67

[39] C. Liechti, "pyserial's documentation." https://pythonhosted.org/pyserial/. Consultado: 22-Ene-2019. 68

[40] P. S. Foundation, "Csv documentation." https://docs.python.org/3/library/csv.html. Consultado: 22-Ene-2019. 68

[41] Riverbank, "What is pyqt?." https://riverbankcomputing.com/software/pyqt/intro. Consultado: 22-Ene-2019. 68

[42] M. development team, "Matplotlib documentation." https://matplotlib.org/. Consultado: 22-Ene-2019. 68

[43] M. Waskom, "Seaborn documentation." https://seaborn.pydata.org/. Consultado: 22-Ene-2019. 68

[44] N. Developers, "Numpy documentation." http://www.numpy.org/. Consultado: 22-Ene-2019. 68

[45] A. Technologies, "Agilent 33220a user's guide." http://ecelabs.njit.edu/student_resources/33220_user_guide.pdf. Consultado: 23-Ene-2019. 81