

# Diseño, fabricación y caracterización de una guía de luz basada en la geometría del cono de Winston para uso en detectores de partículas

Por: Cristian Canales Miranda.

Departamento de ciencias exactas  
Universidad Andres Bello  
[c.alexiscanalesmiranda@gmail.com](mailto:c.alexiscanalesmiranda@gmail.com)

Tesis presentada a la Facultad de ciencias exactas de la Universidad Andrés  
Bello para optar al título profesional de Ingeniero Físico

Agosto 2023  
Santiago, Chile

**Profesor Guía: Renato Galleguillos**

© 2023, Cristian Canales Miranda

Ninguna parte de esta tesis puede reproducirse o transmitirse bajo ninguna forma  
o por ningún medio o procedimiento, sin permiso por escrito del autor.

Se autoriza la reproducción total o parcial, con fines académicos, por  
cualquier medio o procedimiento, incluyendo la cita bibliográfica del documento

A mi yo del pasado.

## AGRADECIMIENTOS

Me gustaría agradecer, en primer lugar, a mi familia: mi papá Cristian, mi mamá Ximena, mi abuela Teresa; ellos siempre estuvieron ahí para cualquier cosa que yo necesitara durante este periodo de formación académica, pero más importante aún, como formación de una persona con valores. Sin ellos, esto no hubiera sido posible. También agradecer a mis amigos más cercanos: Rubén y Cristóbal. Que de alguna u otra manera me ayudaron de manera incondicional. Por último, pero no menos importante, a mi perrita Kiara, quien me acompañó en momentos de estudio a lo largo de toda la carrera.

Quiero agradecer al Dr. Renato Galleguillos por su excelente apoyo e instrucción como guía de este trabajo y en especial por su paciencia y comprensión. Agradecer de igual manera al Dr. Sergey Kuleshov, quien me permitió ser parte del Instituto Milenio Saphir. Gracias al Instituto Milenio de Física Subatómica en la Frontera de Altas Energías - SAPHIR por financiar esta tesis, por dejarme usar sus instalaciones y por darme la oportunidad de conocer a increíbles profesionales.

También agradezco a Loreto Canales, Dr. Sebastián Olivares, Carlos Flores, Matías Liz, Sebastián Cepeda, Matías Henríquez, Marco Ayala; cada uno me ayudó a lo largo de esta tesis y siempre tuvieron la disposición de resolver mis dudas.

Por último, me gustaría agradecer a la UNAB, a todo el equipo docente y administrativo del Departamento de Física.

## Resumen

Se estudiaron diferentes geometrías de guías de onda para ser aplicadas a detectores de partículas de alta energía. Se realizó la comparación de las geometrías utilizadas tanto con FEM y experimentalmente. Las simulaciones y los experimentos se llevaron a cabo para determinar las geometrías que optimizan la eficiencia de transmisión de fotones. Los resultados de las simulaciones muestran una mejora del 54 % y experimentalmente del 13 % en la focalización de fotones utilizando la geometría del cono de Winston modificado en comparación a usar una barra simple. Por otro lado, la geometría del cono de winston modificado muestra un 6 % de mejora con respecto al cono de winston tradicional. Esta mejora del 6 % es igual tanto para simulaciones como los experimentos realizados. Que la mejora en la focalización obtenida mediante dos procesos distintos sea igual destaca la precisión de la técnica empleada para determinar la carga de un fotoelectrón y el ajuste gaussiano de los histogramas de carga.

Finalmente, los resultados indican que la geometría del cono de Winston modificado supera la eficacia al focalizar fotones de manera más eficiente en un 6 %. Además, se tratan posibles discrepancias entre los porcentajes obtenidos entre las simulaciones y los experimentos para la geometría de la barra. Aunque las causas específicas de estas diferencias no se determinan en este trabajo, se enfatiza la importancia de un análisis detallado de los factores relevantes para futuras investigaciones.

Este trabajo de investigación contribuye al avance en la optimización de sistemas de detección de partículas mediante la implementación de guías de luz basadas en la geometría del cono de Winston que aumentan la cantidad de luz disponible en cada proceso de detección, por tanto sus resultados tienen implicaciones significativas para futuras aplicaciones en el campo de la física de partículas.

## Abstract

Different waveguide geometries were studied for application in high-energy particle detectors. A comparison was made between the geometries used, both through finite element method (FEM) simulations and experimental testing. Simulations and experiments were conducted to determine the geometries that optimize photon transmission efficiency. The simulation results show a 54 % improvement, and experimentally, a 13 % improvement in photon focusing using the modified Winston cone geometry compared to using a simple rod. Furthermore, the modified Winston cone geometry exhibits a 6 % improvement over the traditional Winston cone. This 6 % improvement is consistent in both simulations and experiments, highlighting the precision of the technique used to determine the charge of a photoelectron and the Gaussian fitting of charge histograms.

Finally, the results indicate that the modified Winston cone geometry outperforms in efficiently focusing photons by 6 %. Additionally, possible discrepancies between the percentages obtained from simulations and experiments for the rod geometry are addressed. Although the specific causes of these differences are not determined in this work, the importance of a detailed analysis of relevant factors for future research is emphasized.

This research contributes to advancing the optimization of particle detection systems by implementing light guides based on the Winston cone geometry, increasing the amount of available light in each detection process. Therefore, its results have significant implications for future applications in the field of particle physics.

# Índice general

<b>AGRADECIMIENTOS</b>	<b>I</b>
<b>Resumen</b>	<b>II</b>
<b>Abstract</b>	<b>III</b>
<b>1. Introducción</b>	<b>1</b>
<b>2. Marco Teórico</b>	<b>3</b>
2.1. Conos de winston . . . . .	3
2.1.1. Parabola del cono de Winston y sus Parámetros . . . . .	4
2.1.1.1. Diseñando el modelo de la curva del cono de Winston	5
2.1.1.2. Diseñando las geometrias utilizadas . . . . .	7
<b>3. Simulaciones</b>	<b>10</b>
3.1. Trayectoria de los rayos simulados . . . . .	13
3.1.0.1. Cono de winston . . . . .	13
3.1.0.2. Cono de winston modificado . . . . .	14
3.1.0.3. Barra simple . . . . .	14
<b>4. Resultados y análisis Simulaciones</b>	<b>15</b>
4.1. Histogramas de distribución de energía . . . . .	15
4.1.1. Cono de winston . . . . .	16
4.1.2. Cono de winston modificado . . . . .	17
4.1.3. Barra simple . . . . .	17
4.1.4. Comparación para las tres geometrías . . . . .	18
4.1.5. Tiempo de llegada de los rayos en las geometrías . . . . .	20
4.1.6. Número de fotones detectados en las simulaciones . . . . .	21
<b>5. Comparación experimental de recolección de luz para las tres geometrías en estudio</b>	<b>23</b>
5.0.1. Montaje 1: Detección de luz utilizando un PMT como sensor y resina transparente como material . . . . .	24
5.0.2. Preparación de las geometrías . . . . .	24

5.0.3.	Montaje 2: Detección de luz utilizando un MPPC S14160-3050HS y resina transparente como material . . . . .	27
5.0.4.	Montaje 3: Detección de luz utilizando un MPPC S14160-3050HS y acrílico como material . . . . .	30
<b>6. Resultados experimentales</b>		<b>32</b>
6.1.	Montaje 1 . . . . .	33
6.1.1.	Cono de Winston Resina-PMT: . . . . .	33
6.1.2.	Cono de Winston modificado Resina-PMT: . . . . .	34
6.1.3.	Barra simple Resina-PMT: . . . . .	35
6.1.4.	Comparación en la distribución de carga en las geometrías	36
6.2.	Montaje 2 . . . . .	37
6.2.1.	Cono de Winston Resina-MPPC: . . . . .	37
6.2.2.	Cono de Winston modificado-MPPC: . . . . .	38
6.2.3.	Barra simple Resina-MPPC: . . . . .	39
6.2.4.	Comparación en la distribución de carga en las geometrías	40
6.3.	Montaje 3 . . . . .	41
6.3.1.	Cono de Winston Acrílico-MPPC: . . . . .	41
6.3.2.	Cono de Winston modificado Acrílico-MPPC: . . . . .	42
6.3.3.	Barra simple Acrílico-MPPC: . . . . .	43
6.3.4.	Comparación en la distribución de carga en las geometrías	44
6.4.	Ánalisis de la baseline del pulso para obtener la carga de un fotoelectrón . . . . .	46
6.4.1.	Ganancia experimental - Geometría cono de Winston . .	47
6.4.2.	Ganancia experimental - Geometría cono de Winston mod	48
6.4.3.	Ganancia experimental - Geometría cono de barra simple .	48
6.4.4.	Comparación ganancia experimental vs nominal . . . . .	49
<b>7. Conclusion</b>		<b>52</b>
<b>Apéndices</b>		<b>56</b>
<b>A.</b>		<b>56</b>
A0.1.	Código Simulaciones . . . . .	56
A0.2.	Código Montaje 1 . . . . .	71
A0.3.	Código Montaje 2 . . . . .	80
A0.4.	Código Montaje 3 . . . . .	89

# Índice de figuras

2.1.1.Forma del Cono de Winston. Fuente: Merz, J. (2015). Photon detection efficiency measurement of Winston cones for the FAMOUS telescope [Figura 2.5(b), página 8]. . . . .	4
2.1.2.Cono de Winston adherido a una barra de 40 mm . . . . .	7
2.1.3.Cono de Winston . . . . .	7
2.1.4.Cono de Winston visto en el plano XY . . . . .	7
2.1.5.Cono de Winston modificado adherido a una barra de 40 mm . . .	8
2.1.6.Cono de Winston Modificado . . . . .	8
2.1.7.Cono de Winston Modificado visto desde el plano ZX . . . . .	8
2.1.8.Barra 10x10x60.669 [mm]. . . . .	9
 3.0.1.Parámetros utilizados en todas las simulaciones . . . . .	12
3.1.1.Trayectoria de los 2000 rayos en un Cono de Winston . . . . .	13
3.1.2.Trayectoria de los 2000 rayos en un Cono de Winston Modificado . . . . .	14
3.1.3.Trayectoria de los 2000 rayos en una barra simple . . . . .	14
 4.0.1.Energía total de los rayos detectados en distintas geometrías. . . . .	15
4.1.1.Distribución de energía de los rayos detectados en el Cono de Winston. . . . .	16
4.1.2.Distribución de energía de los rayos detectados en el Cono de Winston para una energía mayor a 3[w]. . . . .	16
4.1.3.Distribución de energía de los rayos detectados en el Cono de Winston Mod. . . . .	17
4.1.4.Distribución de energía de los rayos detectados en la Barra. . . . .	17
4.1.5.Comparación de la distribución de energía total de rayos en distintas geometrías. . . . .	18
4.1.6.Comparación de la distribución de energía ( $E_{\min} > 3[w]$ ) de los rayos detectados en distintas geometrías. . . . .	18
4.1.7.Comparación de tiempo de llegada de los rayos detectados en distintas geometrías en un rango de 5[ns]. . . . .	20
4.1.8.Comparación de tiempo de llegada de los rayos detectados en distintas geometrías para un tiempo menor a 0.4[ns] . . . . .	20
4.1.9.Comparación de tiempo de llegada de los fotones detectados y porcentaje de fotones detectados. . . . .	21
4.1.10Comparación de número total de fotones detectados para distintas geometrías. . . . .	22

5.0.1.Diagrama de conexiones del setup utilizado en los montajes realizados.	23
5.0.2.Cono de Winston antes y después del tratamiento de lijado y pulido.	25
5.0.3.Acoples para las geoemtrías de estudio.	25
5.0.4.Montaje de la geometría con los acoplos requeridos para ser conectado al PMT y led.	26
5.0.5.Caja negra conectada al generador de funciones y osciloscopio	26
5.0.6.Detalles del MPPC utilizado.	27
5.0.7.Información datasheet MPPC S14160-3050HS	28
5.0.8.Ensamble de la geometría de estudio con el led y MPPC- Montaje 2.	29
5.0.9.Configuración general del montaje y fuente de alimenntacion de alto voltaje utilizada.	29
5.0.10Proceso de fabricación de la geometría utilizando una fresadora CNC.	30
5.0.11Montaje General	31
6.1.1.Gráfico para los 5000 pulsos - Winston - Resina - Montaje 1.	33
6.1.2.Distribución de cargas para el cono de Winston - Montaje 1.	33
6.1.3.Gráfico para los 5000 pulsos - Winston modificado - Resina - Montaje 1.	34
6.1.4.Distribución de cargas para el cono de Winston modificado - Resina - Montaje 1.	34
6.1.5.Gráfico para los 5000 pulsos - Barra - Resina - Montaje 1.	35
6.1.6.Distribución de cargas para la barra - Resina - Montaje 1.	35
6.1.7.Distribución de carga en las 3 geometrías - Resina - Montaje 1.	36
6.1.8.Comparación de carga total en las 3 geometrías - Resina - Montaje 1.	36
6.2.1.Gráfico para los 5000 pulsos - Winston - Resina - Montaje 2.	37
6.2.2.Distribución de carga cono de Winston - Resina - Montaje 2.	37
6.2.3.Gráfico para los 5000 pulsos - Winston modificado - Resina - Montaje 2.	38
6.2.4.Distribución de carga Winston cone modificado - Resina - Montaje 2	38
6.2.5.Gráfico para los 5000 pulsos - Barra - Resina - Montaje 2.	39
6.2.6.Distribución de carga Barra simple - Resina - Montaje 2.	39
6.2.7.Comparación distribución de carga en las 3 geometrias - Resina - Montaje 2.	40
6.2.8.Comparación de carga total en las 3 geometrias - Resina - Montaje 2	40
6.3.1.Gráfico para los 5000 pulsos - Winston - Acrílico - Montaje 3.	41
6.3.2.Distribución de carga Winston cone - Acrílico - Montaje 3.	42
6.3.3.Gráfico para los 5000 pulsos - Winston modificado - Acrílico - Montaje 3.	42
6.3.4.Distribución de carga Winston cone modificado - Acrílico - Montaje 3.	43
6.3.5.Gráfico para los 5000 pulsos - Barra - Acrílico - Montaje 3.	43
6.3.6.Distribución de carga Barra simple - Acrílico - Montaje 3.	44
6.3.7.Comparación distribución de carga en las 3 geometrías - Acrílico - Montaje 3.	44
6.3.8.Comparación de carga total en las 3 geometrías - Acrílico - Montaje 3.	45
6.4.1.Pulso y peaks detectados en la ROI	46

6.4.2.Histograma de cargas con ajuste, valor de carga, mse, $r^2$ - Geometría Winston . . . . .	47
6.4.3.Histograma de cargas con ajuste, valor de carga, mse, $r^2$ - Geometría Winston modificado . . . . .	48
6.4.4.Histograma de cargas con ajuste, valor de carga, mse, $r^2$ - Geometría barra . . . . .	48
6.4.5.Gráfico de barras para comparar las ganancias obtenidas vs la ganancia nominal . . . . .	49
6.4.6.Gráfico que muestra la relación entre ganancia y overvoltage a una temperatura ambiente de 25°C . . . . .	49
6.4.7.Gráfico que muestra la comparación de fotones en cada una de las geometrías . . . . .	50
6.4.8.Gráfico que muestra la cantidad total de fotones en cada geometría . . . . .	50
6.4.9.Gráfico que muestra el porcentaje relativo de fotones de ambas geometrias con respecto a la geometría del cono de Winston . . . . .	51
A0.1.Universidad Andrés Bello . . . . .	114

# Capítulo 1

## Introducción

En física de partículas, la detección de partículas requiere sistemas ópticos capaces de capturar y transmitir la mayor cantidad de luz posible a los detectores de luz. En este contexto, el diseño y optimización de guías de luz es fundamental para mejorar la eficiencia en la detección de fotones. Es por este motivo que el enfoque de este trabajo es el diseño, fabricación y caracterización de guías de luz basadas en la geometría del cono de Winston. Los conos de Winston<sup>1</sup> son conocidos por su capacidad para focalizar la luz, proporcionando una alta eficiencia de transmisión en un campo de visión limitado, lo cual reduce la posibilidad de que la luz se disperse en diferentes direcciones. Estas características los convierten en candidatos ideales para su uso en detectores de partículas. El objetivo principal de este trabajo fue comparar y analizar el desempeño de diferentes geometrías, utilizando dos materiales diferentes: resina transparente y acrílico. Para esta comparación se utilizaron dos tipos de sensores de fotones: Fotomultiplicador (PMT) y un Multi-Pixel Photon Counters (MPPC). Mediante simulaciones utilizando un software de modelado y análisis computacional y diversos montajes experimentales, se buscó determinar qué geometrías y materiales funcionan mejor como guías de luz. Es decir, cuantificar el número de fotones que llega finalmente al detector para cada geometría en estudio.

El documento se organizará de la siguiente manera: en primer lugar, se presentará una descripción bibliográfica de los conos de Winston, sus principios de funcionamiento y los datos utilizados en este documento. En segundo lugar, se detallarán las geometrías diseñadas en CAD junto a sus características. En tercer lugar, se describirá el funcionamiento y la configuración de las simulaciones realizadas. Posteriormente, se expondrán los resultados obtenidos, así como las conclusiones correspondientes. En cuarto lugar, se proporcionará la descripción y configuración de los tres montajes experimentales realizados: Montaje 1: Detección de luz utilizando un PMT como sensor y resina transparente como material, Montaje 2: Detección de luz utilizando un MPPC S14160-3050HS y resina transparente como material y Montaje 3: Detección de luz utilizando un MPPC S14160-3050HS y acrílico como material. Luego se procedió a realizar un análisis de los datos obtenidos en los montajes mencionados. A continuación, se presentaron los resultados y conclusiones, tanto de esta sección en particular como de manera general. Finalmente, se incluyen todos los códigos realizados en PYTHON, junto con las referencias a otros documentos.

En resumen, este estudio contribuirá al conocimiento y comprensión del uso de guías de luz basadas en la geometría del cono de Winston en detectores de partículas. Los resultados obtenidos permitirán determinar configuraciones óptimas para una mejor recolección de fotones, lo que puede tener importantes implicaciones en el diseño y desarrollo de sistemas de detección de fotones más eficientes y precisos.

# Capítulo 2

## Marco Teórico

### 2.1. Conos de winston

Los conos de Winston son concentradores de luz que se construyen en base a una parábola de revolución. Inicialmente fueron realizados para ser utilizados en telescopios debido a que ofrecen eficiencias de transmisión muy altas. Además de un campo de visión limitado, eliminando la posible dispersión de luz<sup>1</sup>. Además, pueden diseñarse huecos, minimizando el material puesto en el camino de la luz y, por lo tanto, minimizando la posible absorción, especialmente en el espectro UV de la luz de fluorescencia.

La construcción del cono de Winston se realiza definiendo 3 parámetros claves, la apertura de entrada  $r_1$ , la apertura de salida  $r_2$ , y el Ángulo máximo  $\Theta_{max}$ . Dada la apertura de entrada y salida, podemos seguir un rayo de luz, comenzando a la distancia  $r_1$  del eje óptico, que ingresa al cono en  $\Theta_{max}$ . La parábola del cono está diseñada de tal manera que la luz se enfoca en un punto al final del cono con distancia  $r_2$  donde ira conectado el detector encargado de medir la cantidad de fotones que llegan.<sup>1</sup>

### 2.1.1. Parabola del cono de Winston y sus Parámetros

Los parámetros  $r_1$  y  $r_2$  se definen en función de la características físicas del sensor de fotones a utilizar, mientras que  $\Theta_{max}$  está definido por los radios de apertura  $r_1$  y  $r_2$  de la siguiente forma<sup>1</sup>:

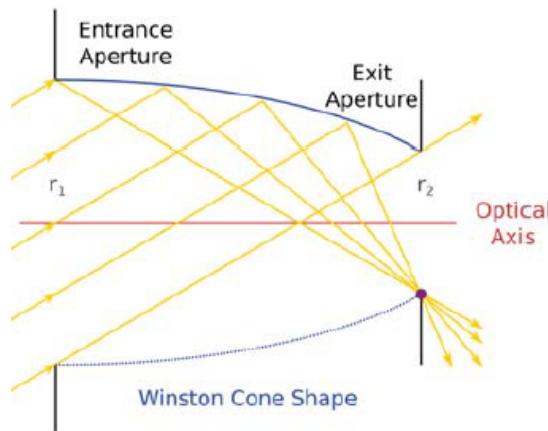
$$\operatorname{sen}\Theta_{max} = \frac{r_1}{r_2} \quad (2.1.1)$$

Ahora para el largo del cono de Winston tendremos lo siguiente:

$$L = \frac{r_1 + r_2}{\tan\Theta_{max}} \quad (2.1.2)$$

Finalmente, podremos expresar la parábola del Cono de Winston de la siguiente forma:

$$r(\Theta) = \frac{1 + \operatorname{sen}\Theta_{max}}{1 - \cos(\Theta + \Theta_{max})} 2r_2 \quad (2.1.3)$$



**Figura 2.1.1:** Forma del Cono de Winston. Fuente: Merz, J. (2015). Photon detection efficiency measurement of Winston cones for the FAMOUS telescope [Figura 2.5(b), página 8].

**2.1.1.1. Diseñando el modelo de la curva del cono de Winston**

Para el montaje del detector se utilizaron los siguientes valores:

$$r_1 = 5[\text{mm}] \quad (2.1.4)$$

$$r_2 = 1,5[\text{mm}] \quad (2.1.5)$$

$$\Theta_{max} = 17,4576 \quad (2.1.6)$$

$$L = 20,6686[\text{mm}] \quad (2.1.7)$$

Finalmente, la curva viene dada por la siguiente expresión:

$$r(\Theta) = \frac{3,9}{1 - \cos(\Theta + 17,4576)} \quad (2.1.8)$$

Para el diseño realizado los valores de  $r_1$  y  $r_2$  son constantes dadas según la configuración del montaje y medidas del área sensible del sensor utilizado. Debido a esto el largo  $L$  y  $\Theta_{max}$  vienen fijados por dichas constantes. Además se busco el punto crítico de la ecuación 2. A continuación, se presenta el desarrollo, primero podemos escribir el largo en función de  $r_1$  y  $r_2$ .

$$L = \frac{r_1 + r_2}{\tan(\Theta_{max})} = \frac{r_1 + r_2}{\tan(\arcsin(\frac{r_1}{r_2}))} \quad (2.1.9)$$

Aplicando derivadas parciales a la ecuación anterior con respecto a las variables  $r_1$  y  $r_2$  obtenemos lo siguiente:

$$\begin{aligned} \frac{\partial L}{\partial r_1} &= \frac{-r_1^3 - r_2^3}{r_1^2 r_2 \sqrt{1 - \frac{r_1^2}{r_2^2}}} \\ \frac{\partial L}{\partial r_2} &= \frac{-r_1^2 + r_1 r_2 + 2r_2^2}{r_1 r_2 \sqrt{1 - \frac{r_1^2}{r_2^2}}} \end{aligned} \quad (2.1.10)$$

(2.1.11)

Ahora, haciendo  $\frac{\partial L}{\partial r_1} = 0$  y  $\frac{\partial L}{\partial r_2} = 0$

$$(r_1 \neq 0 \wedge r_2 = -r_1) \vee$$

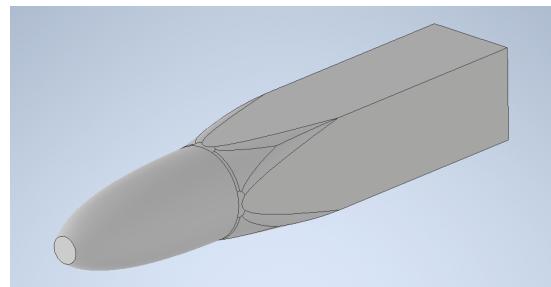
$$(r_1 \neq 0 \wedge r_2 = r_1) \vee$$

$$r_1 = 0 \vee r_2 = 0$$

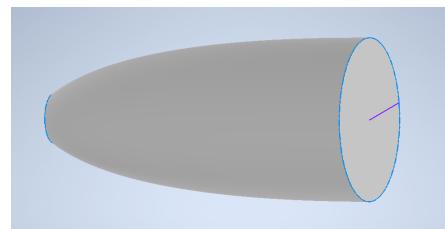
Es por ese motivo que los valores para los radios de entrada y salida del cono no pueden ser iguales, ya que la ecuación (2.1.9) se indeterminaria. Por otro lado, si la elección de  $r_1$  y  $r_2$  son valores muy pequeños implicaría un  $L$  pequeño. Si esto ocurriera ocasionaría problemas de admitancia de luz en la geometría.

### 2.1.1.2. Diseñando las geometrias utilizadas

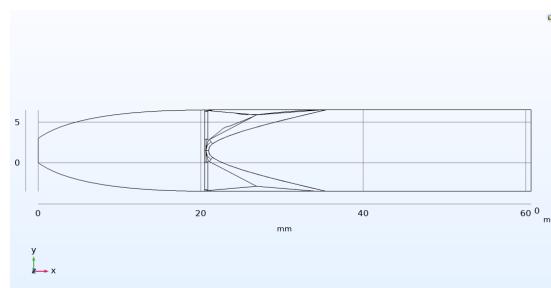
A continuación, se presentarán las 3 geometrías utilizadas a lo largo de todo este documento. Tanto el cono de winston como su modificación están diseñadas con el fin de maximizar la cantidad de fotones detectados por el MPPC, por otro lado la barra simple será usada como control. Ahora, utilizando la ecuación 2.1.8 se procederá a diseñar el modelo del cono de Winston en CAD. Para ello la ecuación 2.1.8 debe rotar con respecto a su eje central, de esta forma se podrá obtener la figura 2.1.3. Por otro lado, a la unión de esta figura con la barra se le realizaron ajustes a sus 4 extremos, esto con el fin de que el cambio de una geometría a otra no sea tan brusco.



**Figura 2.1.2:** Cono de Winston adherido a una barra de 40 mm

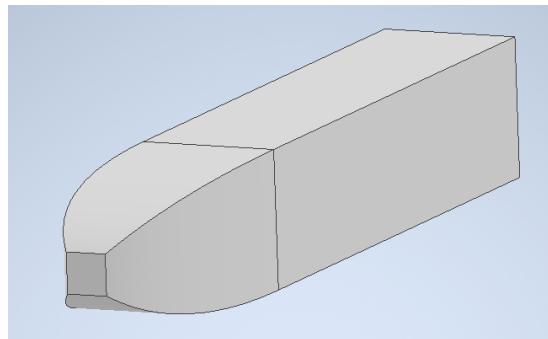


**Figura 2.1.3:** Cono de Winston

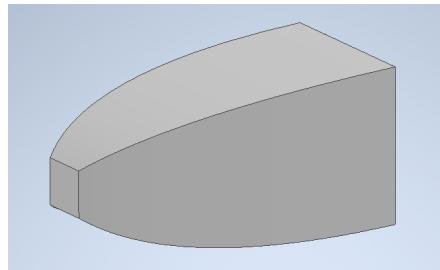


**Figura 2.1.4:** Cono de Winston visto en el plano XY

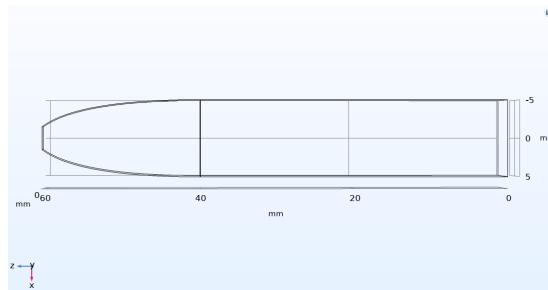
Por otro lado, se utilizo una modificación realizada al cono de Winston original. En función de lograr una comparación más adecuada los radios de apertura y salida serán iguales al cono de Winston. Esta modificación se hace con el fin de aumentar la cantidad de fotones que llegan al sensor(MPPC). A diferencia del cono de winston, la superficie de contacto con el área sensible del mppc tienen la misma geometría. Es decir, un cuadrado de 3x3 mm. De esta forma nos aseguramos de que todos los fotones que llegan pasen por el área de detección.



**Figura 2.1.5:** Cono de Winston modificado adherido a una barra de 40 mm

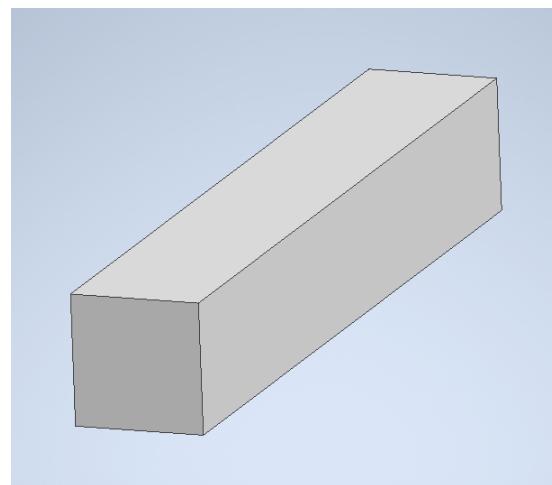


**Figura 2.1.6:** Cono de Winston Modificado



**Figura 2.1.7:** Cono de Winston Modificado visto desde el plano ZX

Por último, se utilizará una barra rectangular como medida de comparación de la cantidad de fotones detectados en el área sensible con respecto a las otras geometrías.



**Figura 2.1.8:** Barra 10x10x60.669 [mm].

Una vez diseñadas las geometrías en CAD, deberán ser exportadas en formato .igl en la mayor calidad posible para ser utilizados en algún software de modelado y análisis computacional.

# Capítulo 3

## Simulaciones

Las simulaciones de óptica geométrica(gop)entregadas por el software nos permite realizar diversas configuraciones para el estudio de ray tracing de nuestras geometrías. Podemos basar nuestro estudio en el cálculo de intensidad, energía o ambas. Para este trabajo, se realizó un estudio basado en la energía de los rayos. Es decir, la simulación nos indica cómo decrece la energía de cada rayo a medida que avanza por la geometría utilizada. En esta configuración cada rayo simula un haz de luz, que representa un conjunto de fotones. Por este motivo, se cuantificó la cantidad de energía final correspondiente a cada rayo que en un periodo de 5 ns llegó al área sensible donde está ubicado el sensor de fotones(MPPC O PMT). Este rango de 5 ns fue establecido debido a que en dicho rango se encuentra la información relevante para el estudio. Luego utilizando el hecho de que la energía y el número de fotones son cantidad proporcionales podemos hacer una conversión de energía a número de fotones. Para ello se utilizó la siguiente ecuación.

$$\frac{E_f}{E_i} = \frac{x}{1000} \frac{\text{fotones}}{\text{fotones}} \quad (3.0.1)$$

Donde la cantidad 1000 representa una cantidad arbitraria de fotones, al ser cantidades proporcionales podemos hacer esto.  $E_f$  representa a la energía final de cada rayo detectado en el sensor.  $E_i$  corresponde a la energía inicial de cada rayo, en la configuración de la simulación se indicó un valor de 10.000 [w] como energía total para los 2000 rayos. De esta forma cada rayo tiene una energía inicial de 5 [w], si cada rayo tiene una cantidad de 1000 fotones podemos decir que inicialmente hay una cantidad de 2.000.000 de fotones en la simulación.

Como bien se mencionó anteriormente, es posible realizar un estudio basado en la intensidad de los rayos. Para ello hay que considerar los parámetros de stokes, estos son un conjunto de valores que describen el estado de polarización de la radiación electromagnética. En este contexto el parámetro  $S_0$  es el más relevante para la simulación por qué representa la intensidad del haz. De esta forma, se podría determinar cuál de todas las geometrías maximiza la intensidad del haz en el detector. El problema de esta configuración es que debido a la naturaleza de las geometrías utilizadas(focalizan los rayos incidentes) se producirán aumentos y disminuciones en la intensidad cuando un rayo sea focalizado o a medida que divergen, respectivamente. Por otro lado, esto también se ve afectado por la reflexión, la refracción y los medios atenuantes.

Producto de lo anterior, esta configuración no es la más adecuada para este caso y por ese motivo no fue incluida en este trabajo<sup>7</sup>. Además de todo lo anteriormente señalado, se deberá de considerar que cuando el haz pase por el material, una cierta parte siempre será absorbida. Debido a esto habrá que considerar el índice de refracción real del material y también su índice imaginario el cual está relacionado con la absorción de luz por el material. Al considerar una configuración de estudio basada en el cálculo de energía de nuestros rayos los cálculos de la simulación solo se ven afectados por la reflexión, la refracción y los medios atenuantes. Por este motivo, esta configuración resulta ser la mejor opción. A continuación, se presenta una expresión para determinar el índice imaginario de refracción  $k^8$ .

$$k = \frac{\lambda}{4\pi\epsilon} \quad (3.0.2)$$

Donde  $\epsilon$  corresponde al *attenuation length*. A continuación se presenta una tabla con los parámetros utilizados en las simulaciones, donde el valor del *attenuation length* para una longitud de onda de 405[nm] fueron obtenidos del estudio<sup>10</sup>. Por otro lado, el valor del índice real para el acrílico fueron obtenidos de la hoja de datos del material, utilizando esta información se determinaron los otros parámetros indicados en la figura 3.0.1.

Parámetros			
Nombre	Expresión	Valor	Descripción
p	3.141592653	3.1416	pi
L	0.57[m]	0.57 m	attenuation length
lambda	405[nm]	4.05E-7 m	longitud de onda del haz
indice_R	1.4970	1.497	índice real
indice_I	(lambda)/(4*p*L)	5.6542E-8	índice imaginario
coef_abs	(4*p*L)/(lambda)	1.7686E7	coeficiente de absorción

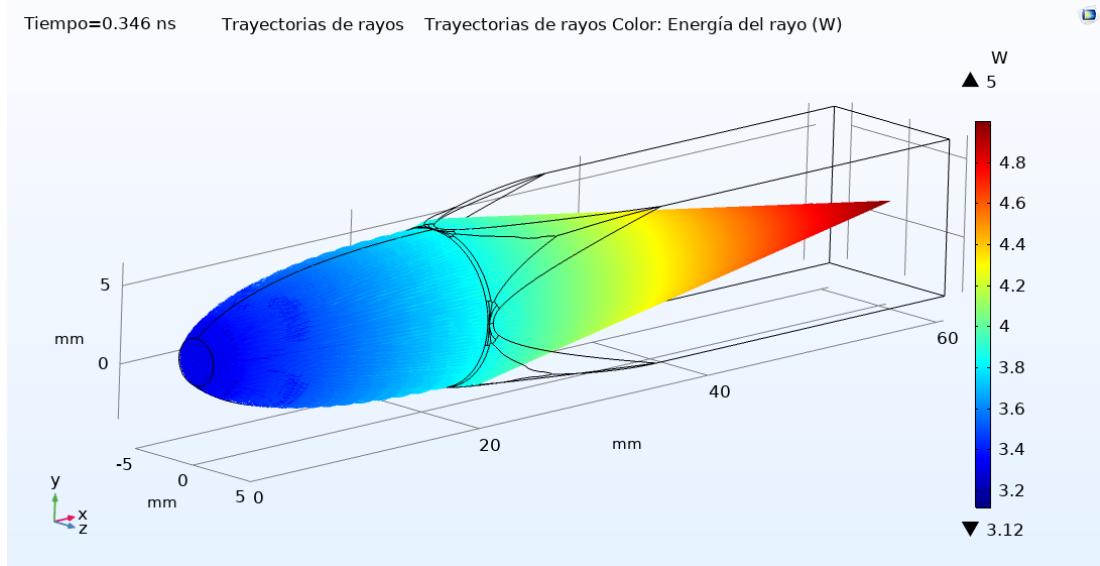
**Figura 3.0.1:** Parámetros utilizados en todas las simulaciones

Finalmente, para todos los casos se simularon 2000 rayos con una energía inicial de 5[W]. Además de considerar un ángulo de radiación incidente de 8 grados en una de las superficies, esto con el fin de hacer lo más real posible la simulación. Por otro lado, el material simulado corresponde al Poly(methyl methacrylate), PMMA) de la marca Bodurov, asociado a la fórmula molecular (C<sub>5</sub>H<sub>8</sub>O<sub>2</sub>)<sub>n</sub>. Por último, se considerará una reflexión especular en las superficies para todas las geometrías.<sup>11</sup>

### 3.1. Trayectoria de los rayos simulados

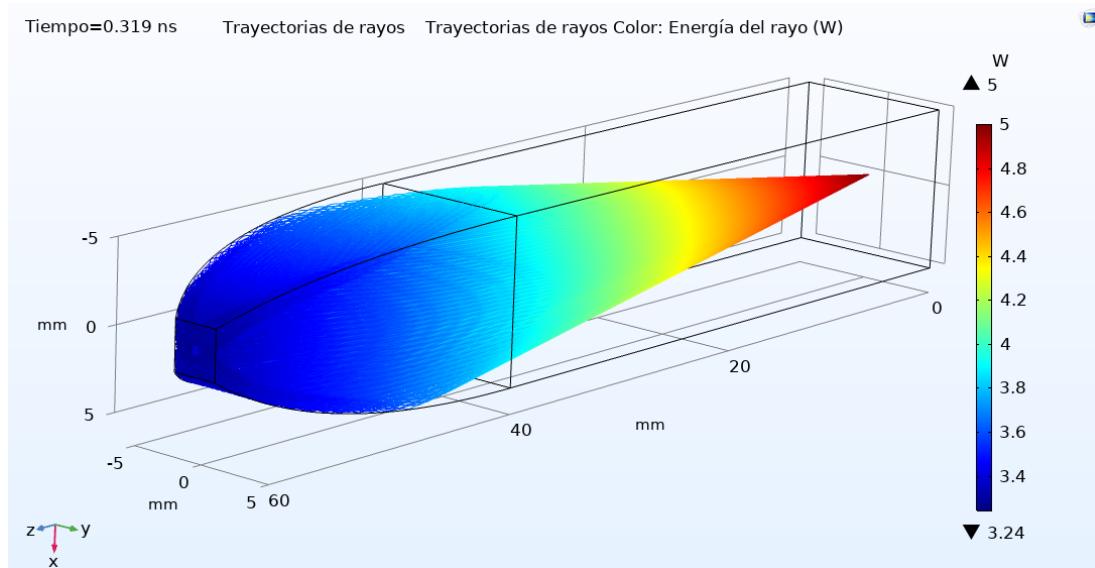
A continuación, se presentan 3 figuras las cuales representan la trayectoria de los rayos en las distintas simulaciones. Podemos apreciar como disminuye la energía de los rayos en función de la distancia que recorre en la geometría mediante la escala de color. Donde el color rojo representa la energía máxima (energía inicial 5[W]) y el color azul la energía mínima de la simulación.

#### 3.1.0.1. Cono de winston



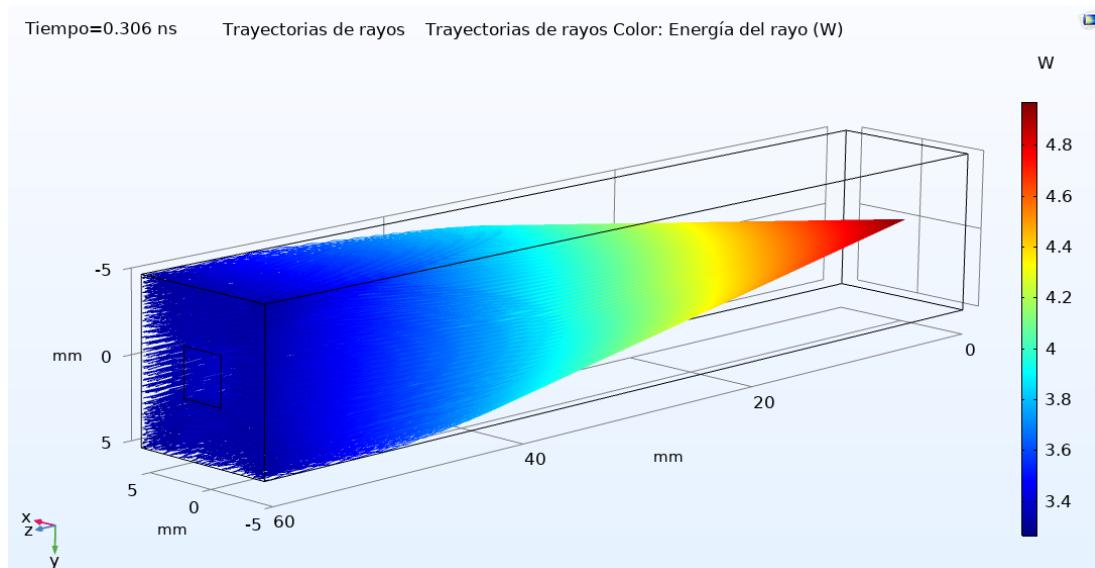
**Figura 3.1.1:** Trayectoria de los 2000 rayos en un Cono de Winston

### 3.1.0.2. Cono de winston modificado



**Figura 3.1.2:** Trayectoria de los 2000 rayos en un Cono de Winston Modificado

### 3.1.0.3. Barra simple

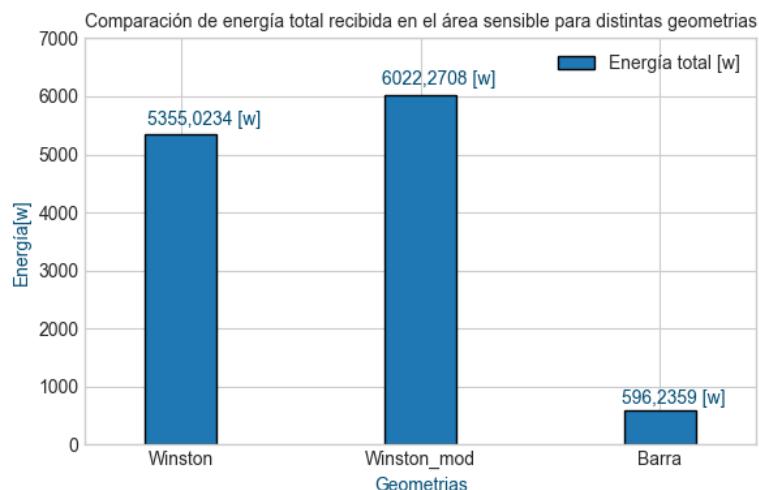


**Figura 3.1.3:** Trayectoria de los 2000 rayos en una barra simple

## Capítulo 4

# Resultados y análisis Simulaciones

De las simulaciones podemos obtener los siguientes resultados:



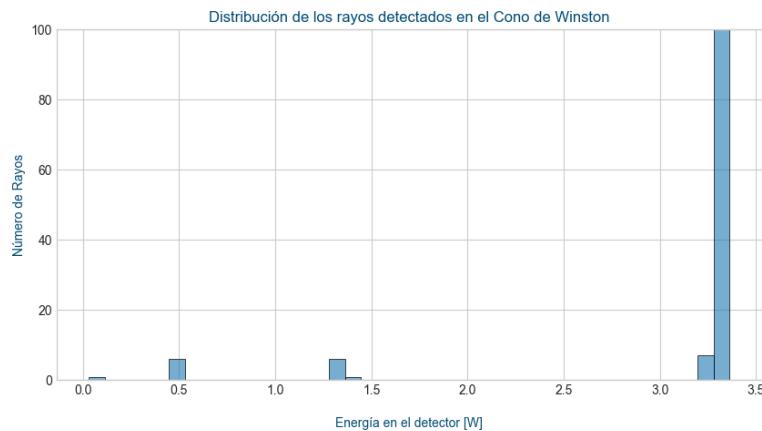
**Figura 4.0.1:** Energía total de los rayos detectados en distintas geometrías.

### 4.1. Histogramas de distribución de energía

A continuación, se puede apreciar distintos histogramas. Cada una de las figuras nos enseña cómo se distribuye la energía de los rayos para distintas geometrías. Notar que no se están incluyendo los rayos no depositados en el detector.

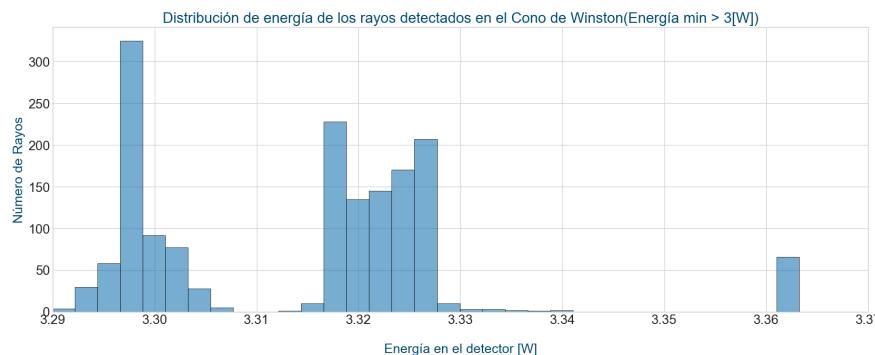
### 4.1.1. Cono de winston

En la figura 4.1.1 se ve cómo se distribuye la energía de todos los rayos para la geometría del cono de Winston. Se hizo un ajuste en la escala vertical de la gráfica, de tal forma que se pueda apreciar que hay rayos que están llegando al área sensible con muy poca energía.



**Figura 4.1.1:** Distribución de energía de los rayos detectados en el Cono de Winston.

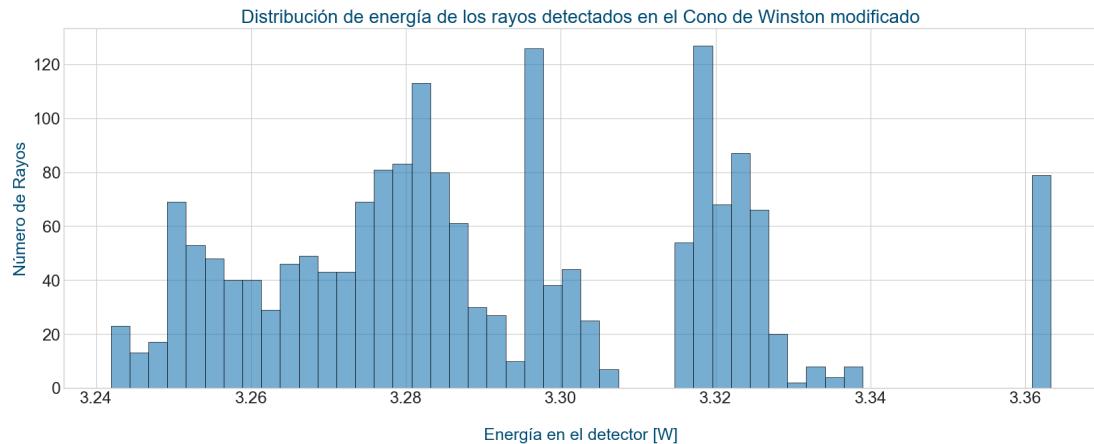
En la figura 4.1.2 nuevamente se aprecia la distribución de energía de los rayos para la geometría del cono de Winston. Esta vez no se considera el ajuste en la escala del eje vertical, pero solo se considera una energía mínima de  $3[W]$  debido a que en este rango se encuentran gran parte de los rayos.



**Figura 4.1.2:** Distribución de energía de los rayos detectados en el Cono de Winston para una energía mayor a  $3[W]$ .

### 4.1.2. Cono de winston modificado

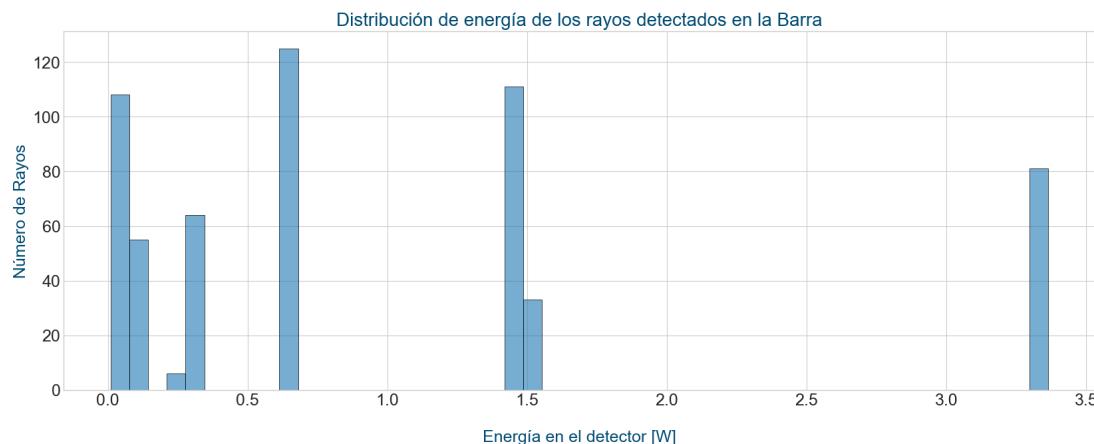
En la figura 4.1.3 se aprecia la distribución de energía de los rayos para la geometría de el cono de Winston modificado. En este caso todos los rayos detectados en el área sensible se encuentran en este rango de energías.



**Figura 4.1.3:** Distribución de energía de los rayos detectados en el Cono de Winston Mod.

### 4.1.3. Barra simple

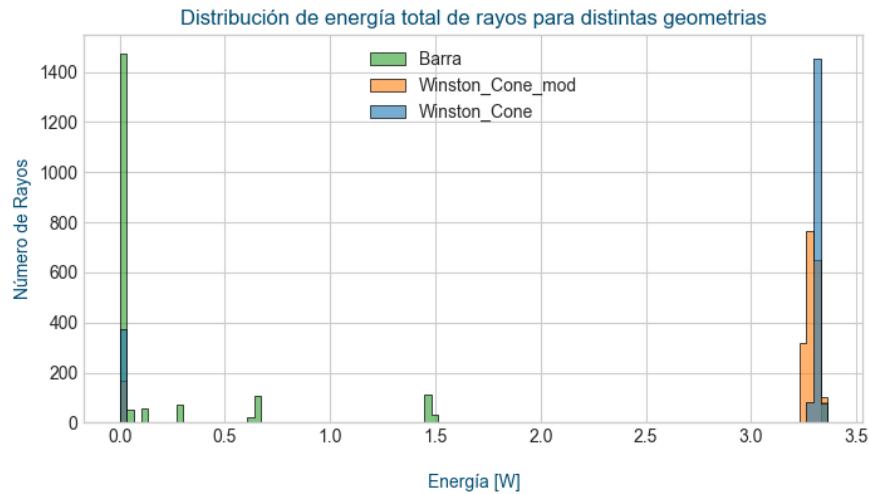
En la figura 4.1.4 se aprecia la distribución de energía de los rayos para la geometría de la barra.



**Figura 4.1.4:** Distribución de energía de los rayos detectados en la Barra.

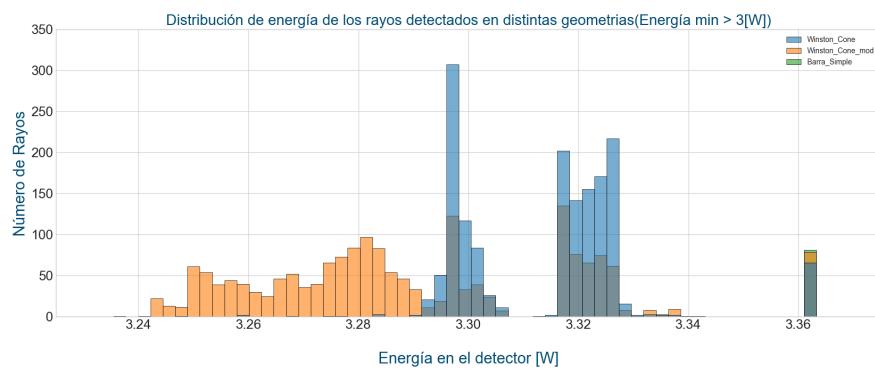
#### 4.1.4. Comparación para las tres geometrías

De forma que se pueda realizar una comparación adecuada, se graficó la distribución de energía de todos los rayos en las geometrías utilizadas. Notar que solo se consideran en el análisis los rayos que inciden en el detector.



**Figura 4.1.5:** Comparación de la distribución de energía total de rayos en distintas geometrías.

La siguiente figura corresponde a la comparación de la distribución de energía de los rayos que sí llegaron al área sensible en las tres geometrías. Visto de otra forma, corresponde a un "zoom" de la figura 4.1.5 en una zona de interés.



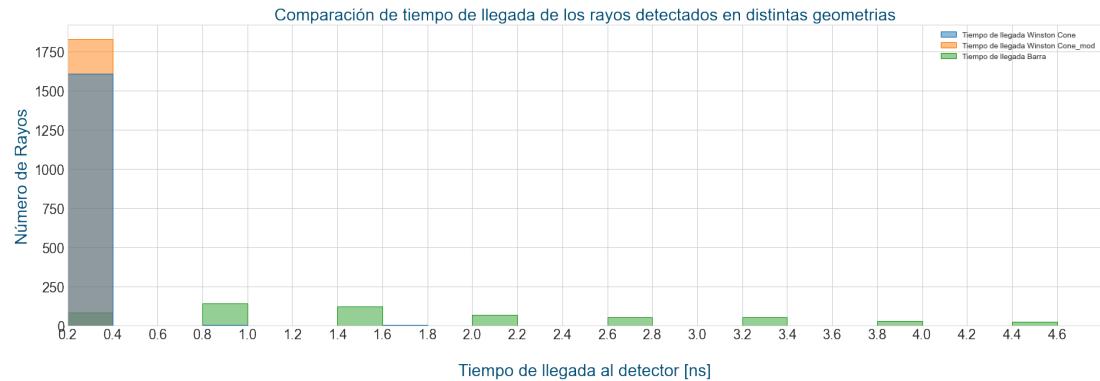
**Figura 4.1.6:** Comparación de la distribución de energía ( $E_{\min} > 3[\text{w}]$ ) de los rayos detectados en distintas geometrías.

En las simulaciones los resultados obtenidos corresponden a lo que se hubiera podido esperar desde un comienzo. Tal como la figura 4.1.6 indica, podemos apreciar que el cono de Winston presenta un rango más acotado para la energía

de sus rayos. Por otro lado, el Winston modificado presenta un amplio rango en sus valores de energía. Esta diferencia nos indica que la geometría del cono de winston modificado focaliza mejor los rayos. Por otro lado podemos ver un peak de energía cercano a 3.36 [w] para las 3 geometrías, esto representa a los rayos que en las 3 geometrías menos camino óptico recorren. Finalmente, la geometría de la barra muestra como claramente los rayos que llegan al detector, son aquellos que recorren un camino óptico mínimo. Es decir, aquellos rayos en los cuales el Ángulo de radiación inicial en la superficie era mínimo. De esta forma, y a pesar de que el cono de Winston modificado presenta una mayor variación en sus valores de energía, es la geometría que focaliza de mejor manera los rayos. Dando como resultado que se maximice la energía detectada.

### 4.1.5. Tiempo de llegada de los rayos en las geometrías

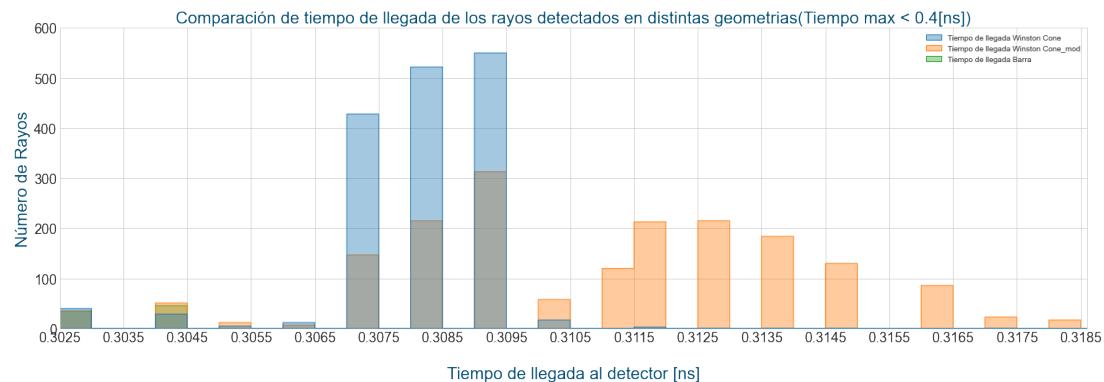
Otro factor importante a considerar es el tiempo de llegada de cada rayo al detector. A continuación, se mostrará un histograma que nos indica cómo varía la distribución del tiempo según la geometría utilizada.



**Figura 4.1.7:** Comparación de tiempo de llegada de los rayos detectados en distintas geometrías en un rango de 5[ns].

Podemos ver fácilmente como la geometría de la barra presenta los mayores tiempos de llegada, esto debido a que habrán rayos que recorren mucho camino óptico antes de llegar al detector.

A continuación podemos apreciar un histograma para tiempos de llegada menor a 0.4 nanosegundos. Esto debido a que en este rango se encuentran los rayos que aportan mayor energía.

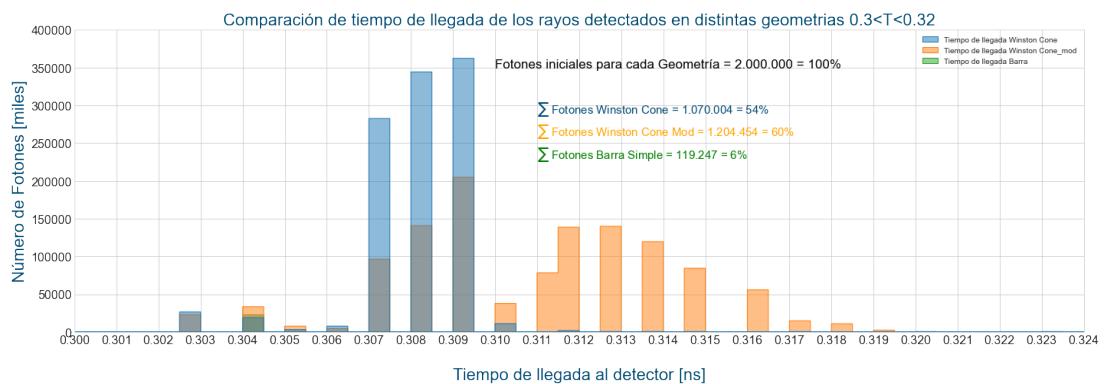


**Figura 4.1.8:** Comparación de tiempo de llegada de los rayos detectados en distintas geometrías para un tiempo menor a 0.4[ns]

#### 4.1.6. Número de fotones detectados en las simulaciones

Por otro lado podemos realizar una conversión de energía y rayos detectados a fotones detectados utilizando la ecuación 3.0.1 explicada en la sección anterior. Para ello asumimos que por cada rayo inicial hay 1000 fotones presentes. De esta forma, podemos obtener los tiempos de llegada para cada fotón detectado. Según la figura 4.1.5 podemos apreciar como gran parte de la información se encuentra en el rango de tiempos entre 0.3 y 0.32 [ns].

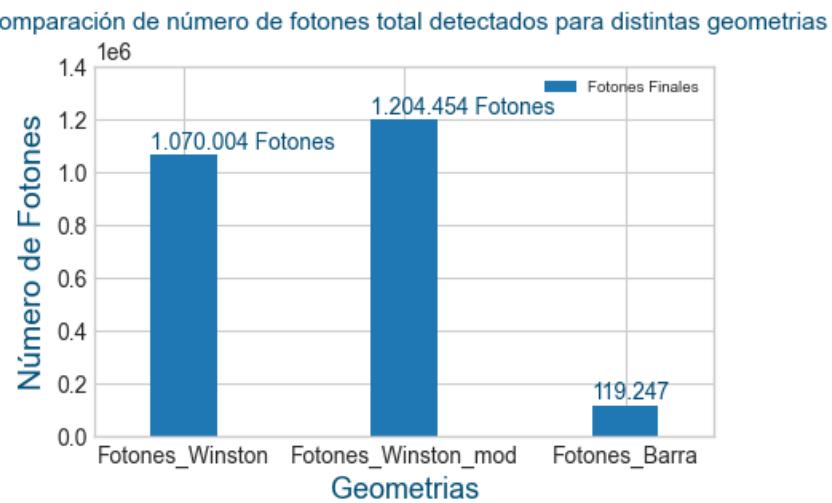
Por ello, el siguiente histograma nos muestra información en dicho rango. Además. Al haber realizado esta conversión a fotones, tenemos una cantidad de 2 millones de fotones iniciales por geometría. Por lo tanto podemos mostrar porcentualmente cuántos fotones están llegando al detector en cada geometría utilizada.



**Figura 4.1.9:** Comparación de tiempo de llegada de los fotones detectados y porcentaje de fotones detectados.

Tal como indica la figura 4.1.9, podemos apreciar que las diferencias de tiempo son tan pequeñas ( $\approx 10[ps]$ ) que no debe representar una diferencia cuantificable si se usa un PMT. Con el uso de un MPPC, estas diferencias serían apenas cuantificables.

Por último, utilizando los datos del gráfico anterior, podemos realizar una comparación del número total de fotones detectados en las geometrías de estudio. De esta forma podemos ver visualmente la diferencia entre ellas descrita anteriormente.



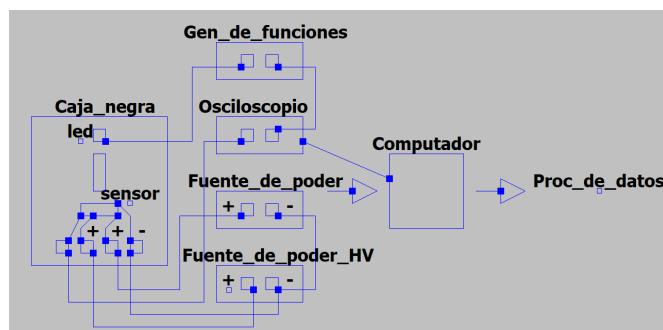
**Figura 4.1.10:** Comparación de número total de fotones detectados para distintas geometrías.

## Capítulo 5

# Comparación experimental de recolección de luz para las tres geometrias en estudio

A continuación y tal como se indicó en la introducción del documento, se presentan 3 montajes experimentales para la detección de fotones. Se utilizaron diferentes geometrías como guías de luz, construidos en 2 materiales. Además, se utilizaran 2 tipos de sensor en la detección de fotones, un PMT y MPPC.

A continuación se puede ver el diagrama utilizado para los 3 montajes mencionados. Donde lo único que variará será el tipo de sensor dentro de una caja negra. Esta caja se utiliza con el fin de que no entre luz al interior, debido a que afectaría a las mediciones. Además de dañar los sensores utilizados.



**Figura 5.0.1:** Diagrama de conexiones del setup utilizado en los montajes realizados.

### 5.0.1. Montaje 1: Detección de luz utilizando un PMT como sensor y resina transparente como material

Para este montaje se realizó la impresión 3D SLT de las 3 geometrías en resina fotocurable transparente. Las geometrías fueron montadas en una caja negra. En ella, una de las caras será iluminada por un led. Mientras que la otra irá junto al PMT. Ambos extremos serán acoplados con aceite de silicona, esto con el fin de minimizar el cambio de medio. Luego, el generador de funciones irá conectado al led para generar un pulso. Mientras que la salida del fotomultiplicador irá conectado al osciloscopio, de forma que se pueda replicar las condiciones de la simulación.

### 5.0.2. Preparación de las geometrías

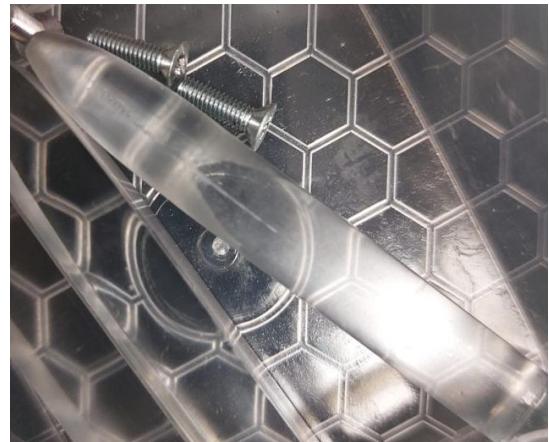
- Lijado y Pulido
  - 1. Materiales utilizados:
    - a) Lija al agua de 1000, 1500, 2000 granos
    - b) Alúmina para pulir de 5 y 3  $\mu\text{m}$
    - c) Paño para pulir
    - d) Agua Destilada
    - e) Gotero

Lijar las geometrías con la lija de menor a mayor grano hasta eliminar todos los desperfectos, para ello utilizar el agua destilada y el gotero. Luego, aplicar con el paño la Alúmina de 5 y 3  $\mu\text{m}$  respectivamente por varios minutos hasta lograr un resultado satisfactorio.

A continuación, se puede apreciar una de las geometrías antes del proceso de lijado y pulido.



(a) Cono de Winston sin tratamiento de lijado y pulido

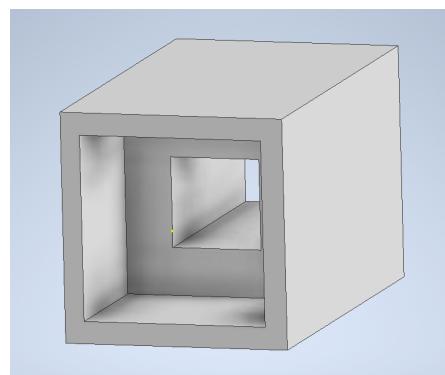


(b) Cono de Winston con tratamiento de lijado y pulido

**Figura 5.0.2:** Cono de Winston antes y después del tratamiento de lijado y pulido.

Con las geometrías listas, se procedió a montar todo dentro de una caja negra. Para ello se utilizaron 2 acoplos diseñados en software CAD. De tal forma que se pueda conectar a la geometría con el led en una cara y el PMT en la restante.

Se puede apreciar los 2 acoplos mencionados. Estos acoplos fueron diseñados con el fin de alinear de la mejor forma posible el área sensible del PMT con la guía de luz y esta a su vez con el led.



(a) Conector entre la Geometría y el led que iluminará la cara



(b) Conector led y PMT

**Figura 5.0.3:** Acoplos para las geometrías de estudio.

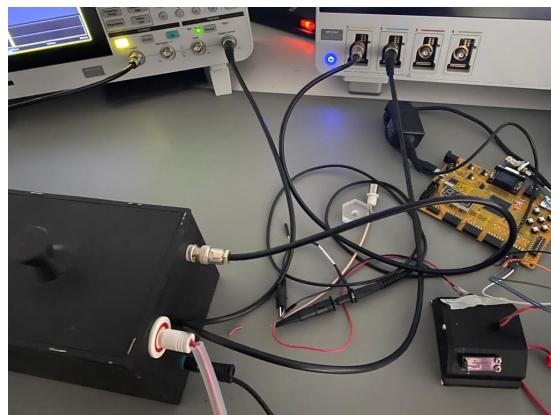
Finalmente se armo todo en la caja negra. Así, el montaje queda de la siguiente forma.



(a) Caja negra con PMT  
(b) Caja negra con todo  
montado montado

**Figura 5.0.4:** Montaje de la geometría con los acoplos requeridos para ser conectado al PMT y led.

La figura 5.0.5 muestra las conexiones electricas que consisten en iluminar el led con el generador de funciones, y la salida del PMT conectada a un osciloscopio.



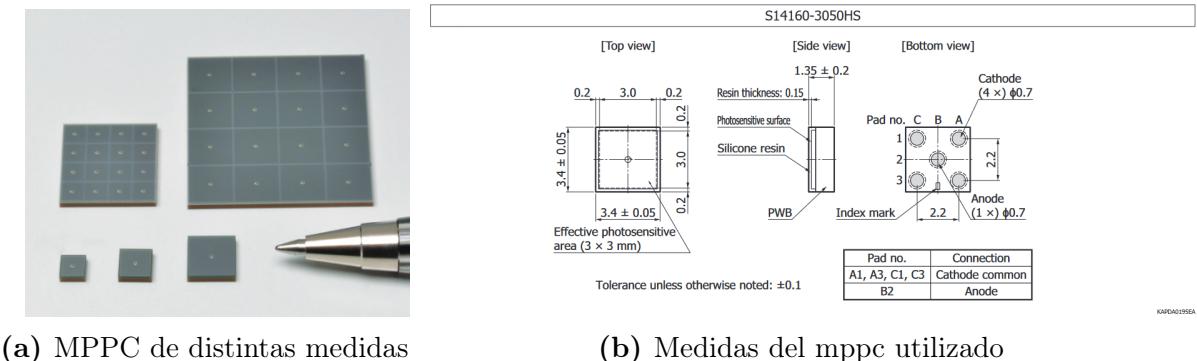
**Figura 5.0.5:** Caja negra conectada al generador de funciones y osciloscopio

### 5.0.3. Montaje 2: Detección de luz utilizando un MPPC S14160-3050HS y resina transparente como material

Utilizando las mismas Geometrías se reemplazó el montaje de la caja negra y PMT por el MPPC. Ahora la salida del MPPC fue conectada a una placa que convertirá la corriente de salida en voltaje. Para ello, se usaron 2 fuentes de alimentación, una encargada de suministrar 5V y otra configurada a 40 V (Voltaje necesario para que el MPPC entre en modo de operación).

Nuevamente, el generador de funciones irá conectado al led, mientras que el osciloscopio irá conectado a uno de los 2 canales de la placa. Por otro lado, al igual que el montaje anterior, se debe utilizar aceite de silicona en los extremos de las geometrías como acoplante.

A continuación, en la figura 5.0.6 se muestra el mppc utilizado junto a un diagrama de sus medidas<sup>12</sup>.



**Figura 5.0.6:** Detalles del MPPC utilizado.

A continuación, se presentan las caracteristicas entregadas por el fabricante para el MPPC utilizado<sup>12</sup>. De esta información los datos más relevantes para este trabajo son la tensión de ruptura, voltaje de operación y la ganancia del detector.

**► Electrical and optical characteristics (Typ. Ta=25 °C, Vover=2.7 V, unless otherwise noted)**

Parameter	Symbol	S14160/S14161 -3050HS-04, -08	S14160/S14161 -4050HS-06	S14160/S14161 -6050HS-04	unit	
Spectral response range	$\lambda$		270 to 900		nm	
Peak sensitivity wavelength	$\lambda_p$		450		nm	
Photon detection efficiency at $\lambda_p$ <sup>*3</sup>	PDE		50		%	
Breakdown voltage	V <sub>BR</sub>		38		V	
Recommended operating voltage <sup>*4</sup>	V <sub>op</sub>		V <sub>BR</sub> + 2.7		V	
V <sub>op</sub> variation between channels in one product <sup>*5</sup>	Typ. Max.	-	0.1 0.2		V	
Dark current	Typ. Max.	ID	0.6 1.8	1.1 3.3	2.5 7.5	$\mu$ A
Crosstalk probability	-		7		%	
Terminal capacitance	C <sub>t</sub>	500	900	2000	pF	
Gain	M		$2.5 \times 10^6$		-	
Temperature coefficient of recommended reverse voltage	$\Delta T_{Vop}$		34		mV/°C	

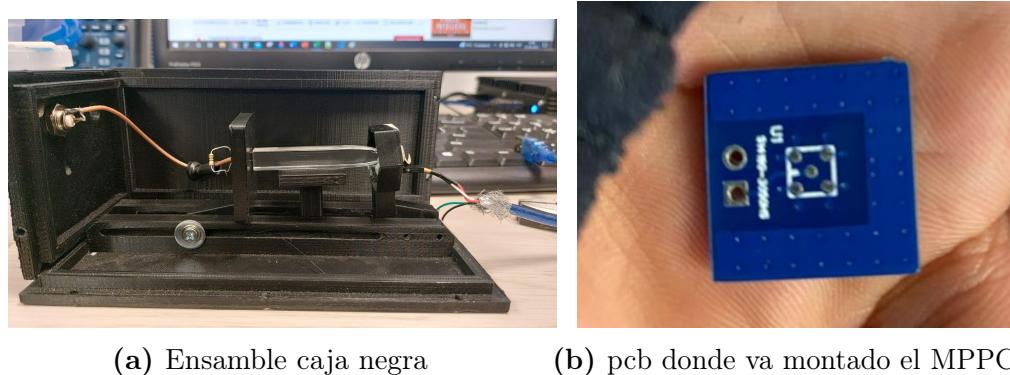
\*3: Photon detection efficiency does not include crosstalk and afterpulses.

\*4: Refer to the data attached for each product.

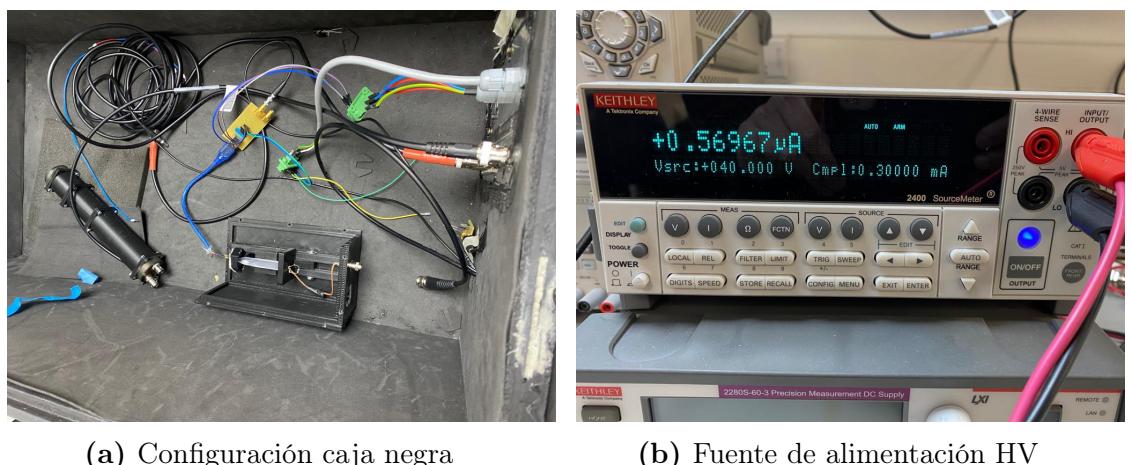
\*5: The parameter is for the S14161 series (multichannel type)

**Figura 5.0.7:** Información datasheet MPPC S14160-3050HS

A diferencia del montaje 1 donde se utilizó un PMT como sensor, el utilizar un MPPC en este montaje implica una nueva forma de montar la guía de luz para que quede alineada de forma correcta con respecto al led y el MPPC. En la figura 5.0.8(a) se puede apreciar el montaje general, fabricado con impresión 3D FDM en PLA.



**Figura 5.0.8:** Ensamble de la geometría de estudio con el led y MPPC- Montaje 2.



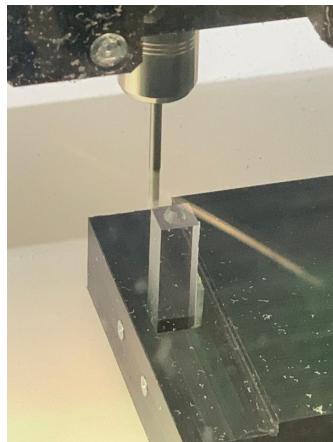
**Figura 5.0.9:** Configuración general del montaje y fuente de alimentación de alto voltaje utilizada.

#### 5.0.4. Montaje 3: Detección de luz utilizando un MPPC S14160-3050HS y acrílico como material

Este experimento es analógico al anterior. La única diferencia fue el cambio de material en las 3 geometrías. En esta ocasión, se utilizó PMMA(Acrílico) para construirlas. El realizar este cambio de material implica diversas dificultades para la construcción de las geometrías. Por lo mismo, se utilizó una fresadora CNC para realizar los cortes y dar la forma adecuada.



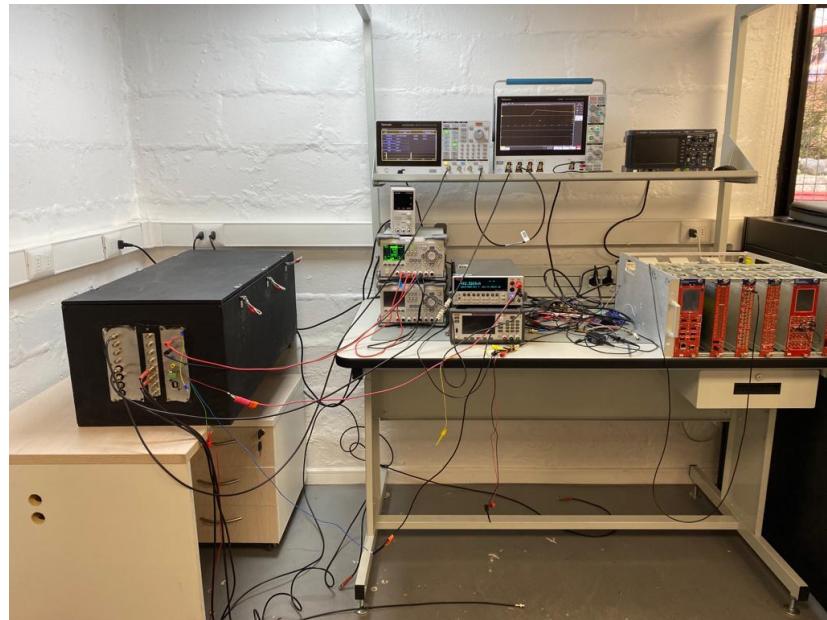
(a) Barra de Acrílico recien cortada



(b) Fresadora CNC dando forma a la barra

**Figura 5.0.10:** Proceso de fabricación de la geometría utilizando una fresadora CNC.

Con las Geometrías listas, se procede nuevamente a lijar y pulir. Esta vez, se utilizó lija de 400 y 600 granos. Además de alúmina de 5 y 1  $\mu\text{m}$ . Para hacer más fácil el trabajo se utilizó una máquina orbital para pulir las superficies.



**Figura 5.0.11:** Montaje General

Como consideración el montaje 3 será el más utilizado al momento de realizar los análisis correspondientes. Ya que este nuevo material(Acrílico) es más homogéneo y sus propiedades ópticas isotrópicas, lo que le permite obtener resultados más confiables.

# Capítulo 6

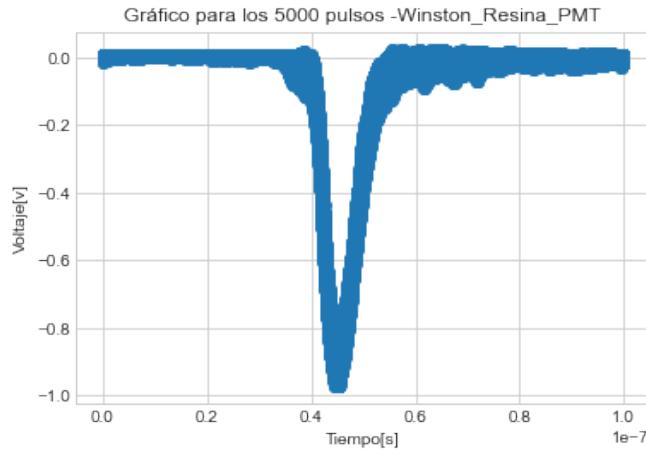
## Resultados experimentales

Para cada caracterización se tomaron 5000 pulsos en el osciloscopio, que representan 5000 experimentos individuales. En el caso del montaje 1 y 2 los 5000 pulsos se conforman de 1250 datos de voltaje en función del tiempo y se uso un Sampling Rate de 12.5[GS/s]. En el montaje 3 los 5000 pulsos se conforman de 3125 datos de voltaje en función del tiempo y se uso un Sampling Rate de 3.125[GS/s].

## 6.1. Montaje 1

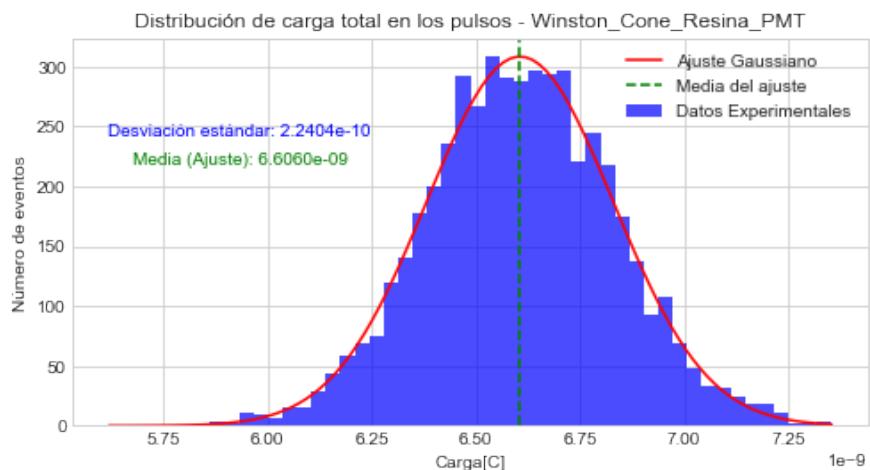
### 6.1.1. Cono de Winston Resina-PMT:

En la figura 6.1.1 se puede apreciar la gráfica para los 5000 pulsos utilizados.



**Figura 6.1.1:** Gráfico para los 5000 pulsos - Winston - Resina - Montaje 1.

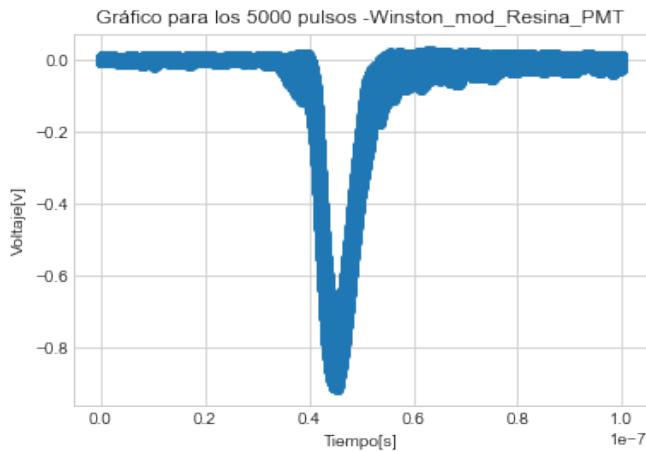
A continuación se grafico la distribución de cargas para la geometría del cono de winston. Para obtener esta grafica se integraron los valores correspondientes de voltaje en función del tiempo para cada pulso. De esta forma se puede construir un histograma con las cargas de cada pulso. Luego se procedio a hacer un ajuste gaussiano al histograma, de esta forma podemos obtener el sigma y media presente en este fit.



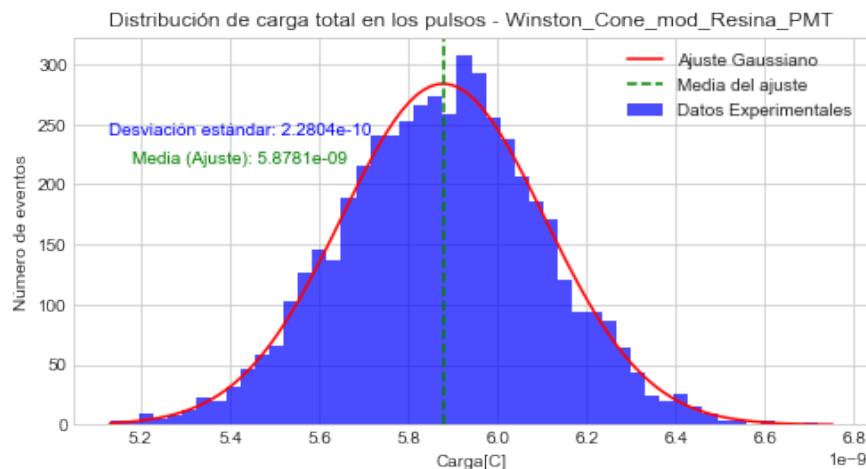
**Figura 6.1.2:** Distribución de cargas para el cono de Winston - Montaje 1.

### 6.1.2. Cono de Winston modificado Resina-PMT:

Los siguientes gráficos se obtuvieron haciendo lo mismo explicado anteriormente.

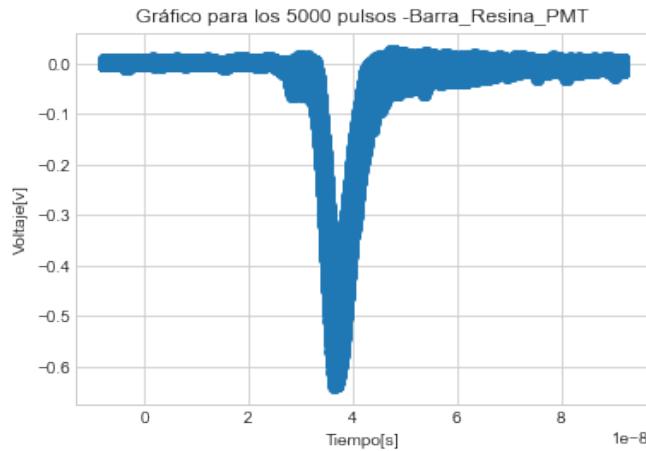


**Figura 6.1.3:** Gráfico para los 5000 pulsos - Winston modificado - Resina - Montaje 1.

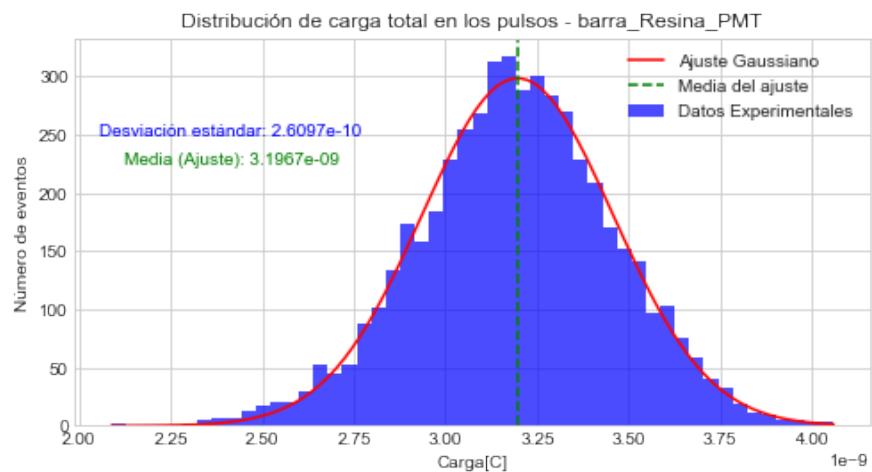


**Figura 6.1.4:** Distribución de cargas para el cono de Winston modificado - Resina - Montaje 1.

### 6.1.3. Barra simple Resina-PMT:



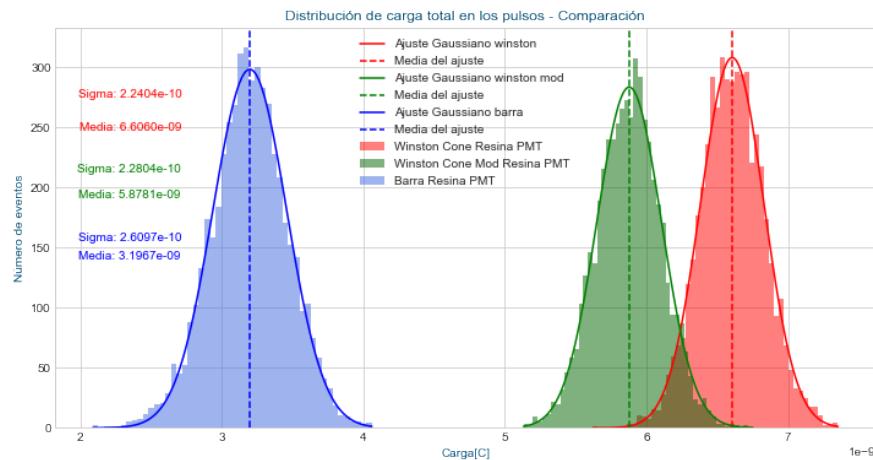
**Figura 6.1.5:** Gráfico para los 5000 pulsos - Barra - Resina - Montaje 1.



**Figura 6.1.6:** Distribución de cargas para la barra - Resina - Montaje 1.

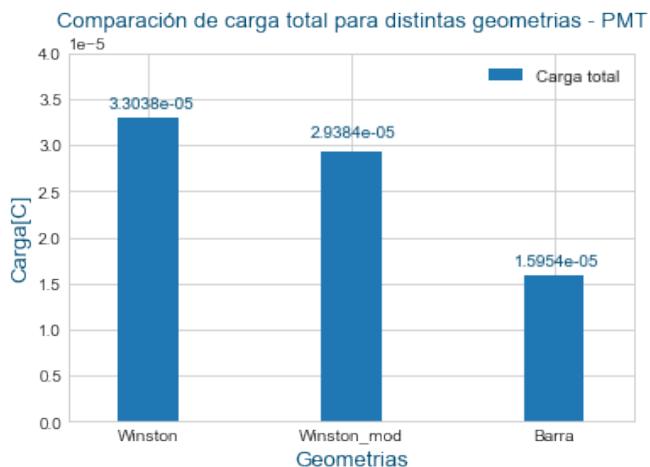
### 6.1.4. Comparación en la distribución de carga en las geometrías

A continuación podemos apreciar una comparación en la distribución de carga para las 3 geometrías.



**Figura 6.1.7:** Distribución de carga en las 3 geometrías - Resina - Montaje 1.

Por último, podemos realizar una comparación en la carga total de las 3 geometrías utilizadas.

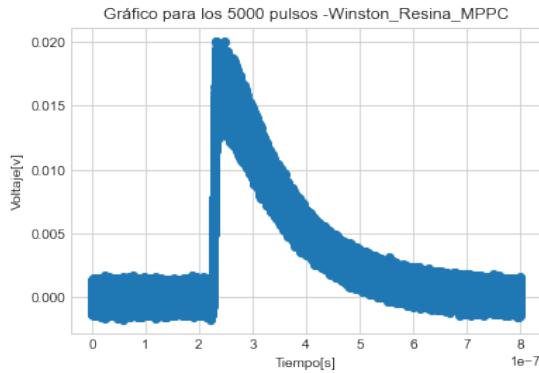


**Figura 6.1.8:** Comparación de carga total en las 3 geometrías - Resina - Montaje 1.

## 6.2. Montaje 2

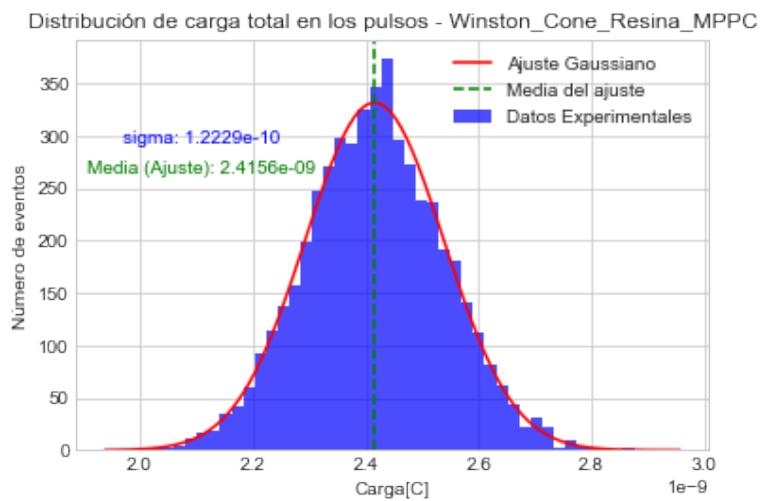
### 6.2.1. Cono de Winston Resina-MPPC:

En la figura 6.2.1 se puede apreciar la gráfica para los 5000 pulsos utilizados.



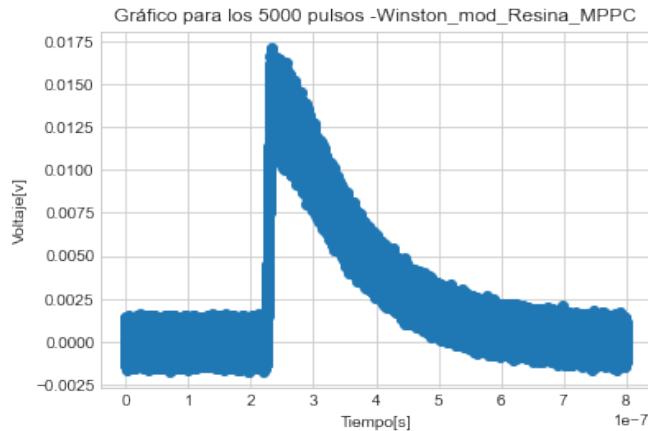
**Figura 6.2.1:** Gráfico para los 5000 pulsos - Winston - Resina - Montaje 2.

A continuación se grafico la distribución de cargas para la geometría del cono de winston. Para obtener esta grafica se integraron los valores correspondientes de voltaje en funcion del tiempo para cada pulso. De esta forma se puede construir un histograma con las cargas de cada pulso. Luego se procedio a hacer un ajuste gaussiano al histograma, de esta forma podemos obtener el sigma y media presente en este fit.

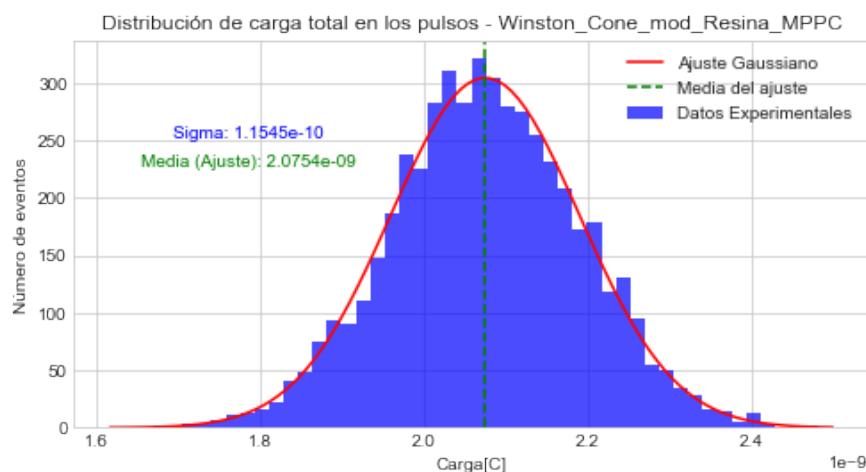


**Figura 6.2.2:** Distribución de carga cono de Winston - Resina - Montaje 2.

### 6.2.2. Cono de Winston modificado-MPPC:



**Figura 6.2.3:** Gráfico para los 5000 pulsos - Winston modificado - Resina - Montaje 2.



**Figura 6.2.4:** Distribución de carga Winston cone modificado - Resina - Montaje 2

### 6.2.3. Barra simple Resina-MPPC:

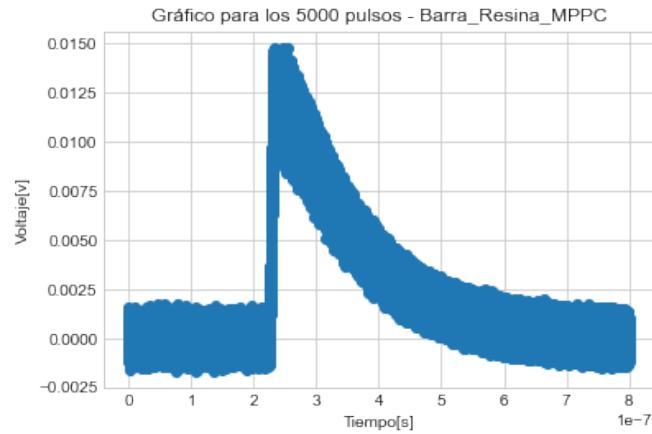


Figura 6.2.5: Gráfico para los 5000 pulsos - Barra - Resina - Montaje 2.

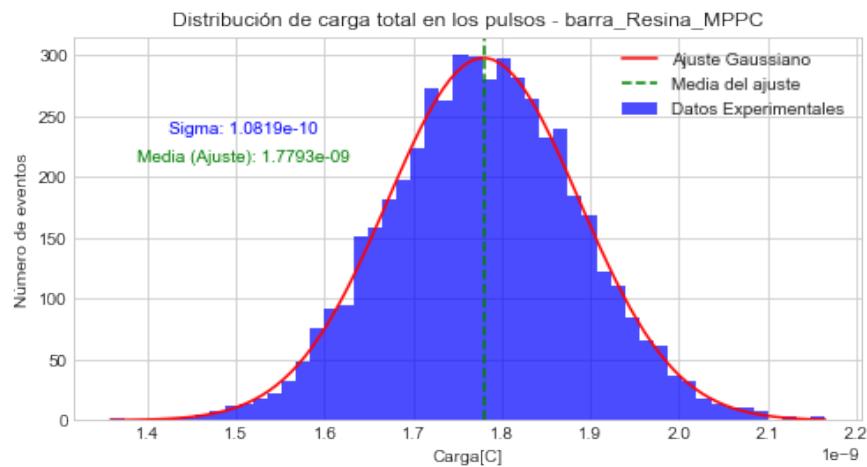
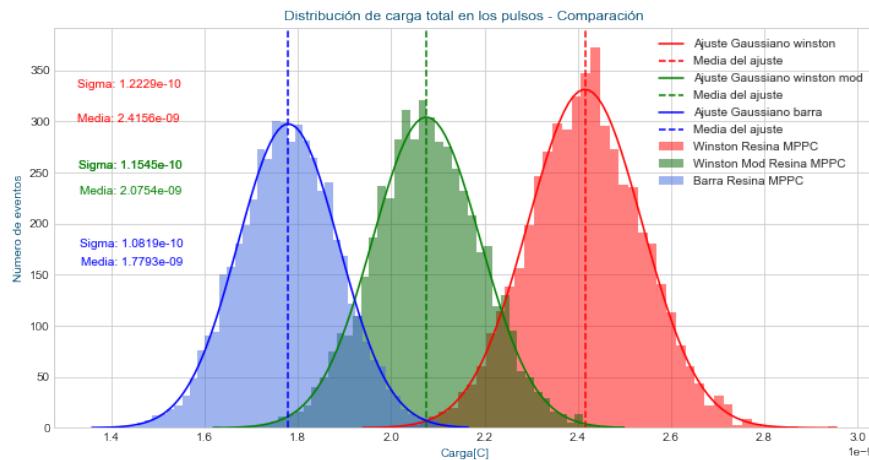


Figura 6.2.6: Distribución de carga Barra simple - Resina - Montaje 2.

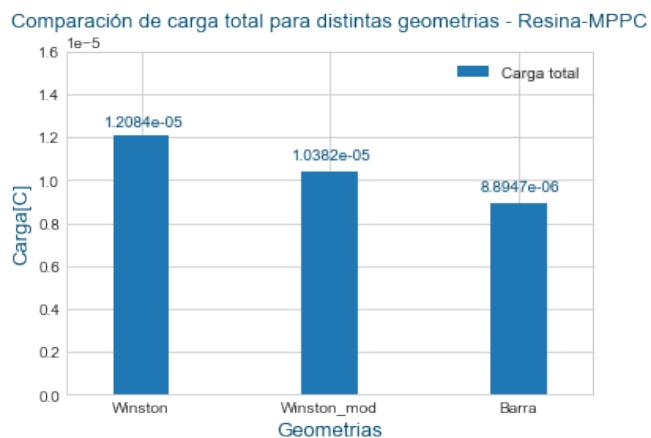
### 6.2.4. Comparación en la distribución de carga en las geometrías

A continuación podemos apreciar una comparación en la distribución de carga para las 3 geometrías.



**Figura 6.2.7:** Comparación distribución de carga en las 3 geometrías - Resina - Montaje 2.

Por último, podemos realizar una comparación en la carga total de las 3 geometrías utilizadas. Para lograr esto basta con sumar la contribución de cargas para cada fit.



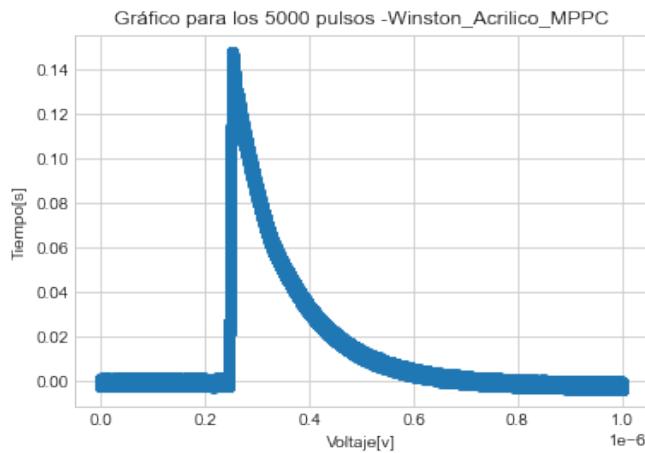
**Figura 6.2.8:** Comparación de carga total en las 3 geometrías - Resina - Montaje 2

## 6.3. Montaje 3

Tal como se indicó anteriormente este montaje será el más analizado con respecto a los anteriores. Aquí hay un total de 5000 pulsos donde cada pulso se conforma de 3125 datos de voltaje en función del tiempo.

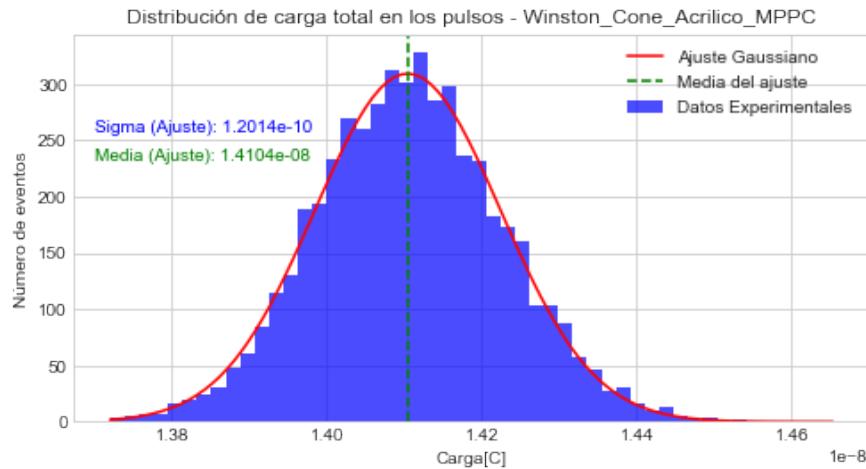
### 6.3.1. Cono de Winston Acrílico-MPPC:

En la figura 6.3.1 se puede apreciar la grafica para los 5000 pulsos utilizados.



**Figura 6.3.1:** Gráfico para los 5000 pulsos - Winston - Acrílico - Montaje 3.

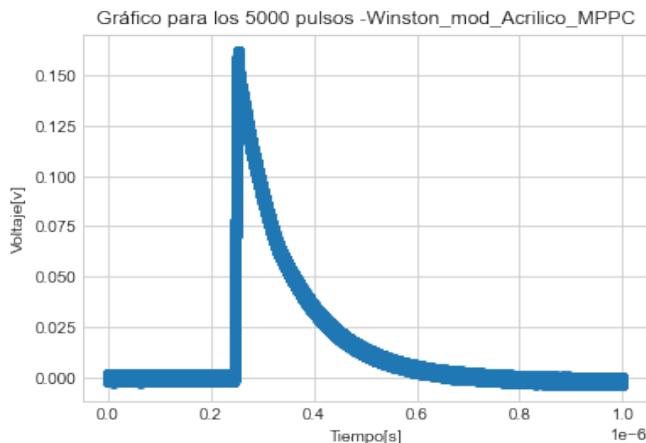
A continuación se grafico la distribución de cargas para la geometría del cono de winston. Para obtener esta grafica se integraron los valores correspondientes de voltaje en función del tiempo para cada pulso. De esta forma se puede construir un histograma con las cargas de cada pulso. Luego se procedió a hacer un ajuste gaussiano al histograma, de estas forma podemos obtener el sigma y media presente en este fit.



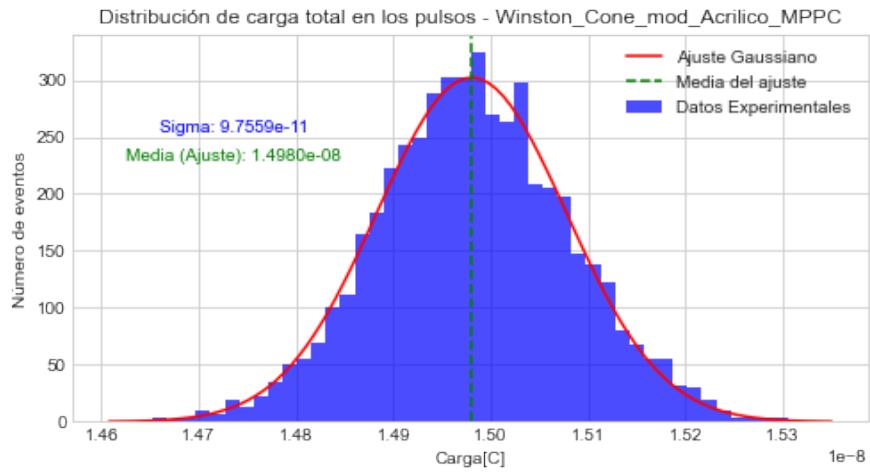
**Figura 6.3.2:** Distribución de carga Winston cone - Acrílico - Montaje 3.

### 6.3.2. Cono de Winston modificado Acrílico-MPPC:

De igual forma a lo señalado para el cono de Winston, aquí se obtuvieron las gráficas de igual manera.



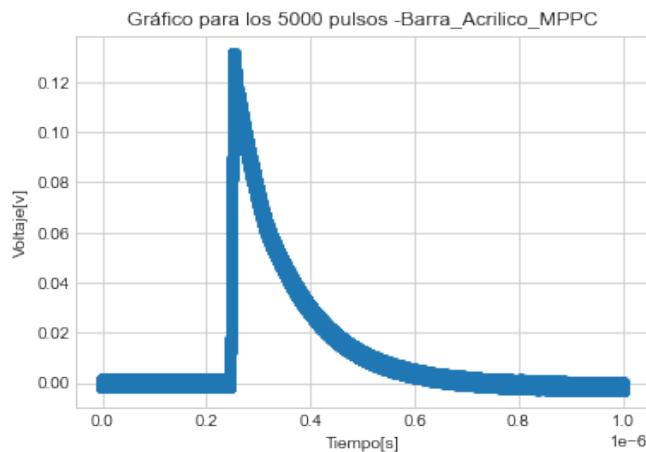
**Figura 6.3.3:** Gráfico para los 5000 pulsos - Winston modificado - Acrílico - Montaje 3.



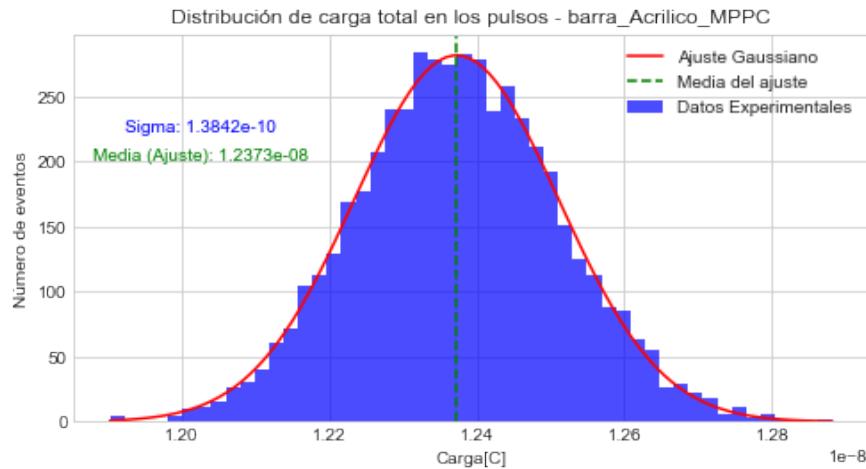
**Figura 6.3.4:** Distribución de carga Winston cone modificado - Acrílico - Montaje 3.

### 6.3.3. Barra simple Acrílico-MPPC:

De igual forma a lo señalada para el cono de winston, aquí se obtuvieron las graficas de igual manera.



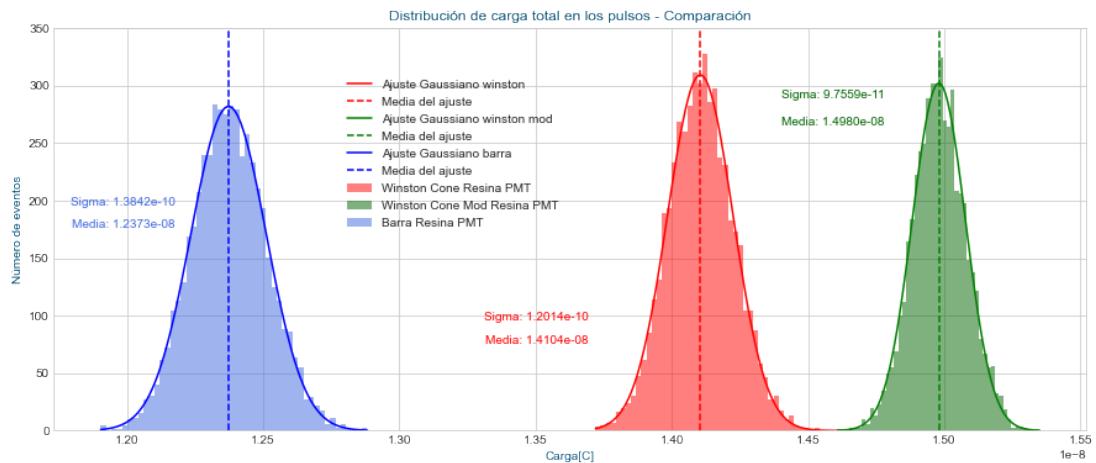
**Figura 6.3.5:** Gráfico para los 5000 pulsos - Barra - Acrílico - Montaje 3.



**Figura 6.3.6:** Distribución de carga Barra simple - Acrílico - Montaje 3.

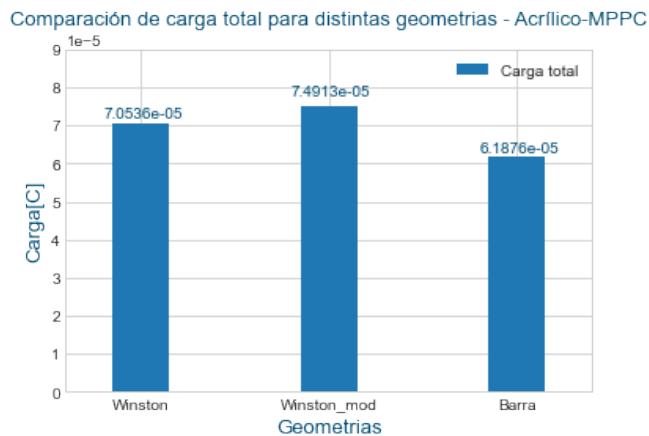
#### 6.3.4. Comparación en la distribución de carga en las geometrías

A continuación, en la figura 6.3.7 se muestran los ajustes de los histogramas mencionados en un solo plot, de tal forma que se puedan apreciar las diferencias de carga asociadas a cada geometría.



**Figura 6.3.7:** Comparación distribución de carga en las 3 geometrías - Acrílico - Montaje 3.

Por último, podemos realizar una comparación en la carga total de las 3 geometrías utilizadas. Para lograr esto basta con sumar la contribución de cargas para cada geometría de estudio.

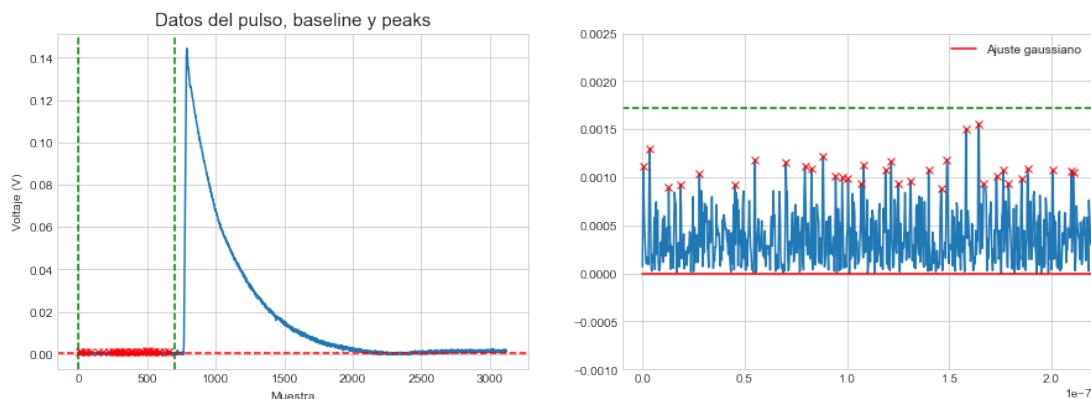


**Figura 6.3.8:** Comparación de carga total en las 3 geometrías - Acrílico - Montaje 3.

#### 466.4. Análisis de la baseline del pulso para obtener la carga de un fotoelectrón

## 6.4. Análisis de la baseline del pulso para obtener la carga de un fotoelectrón

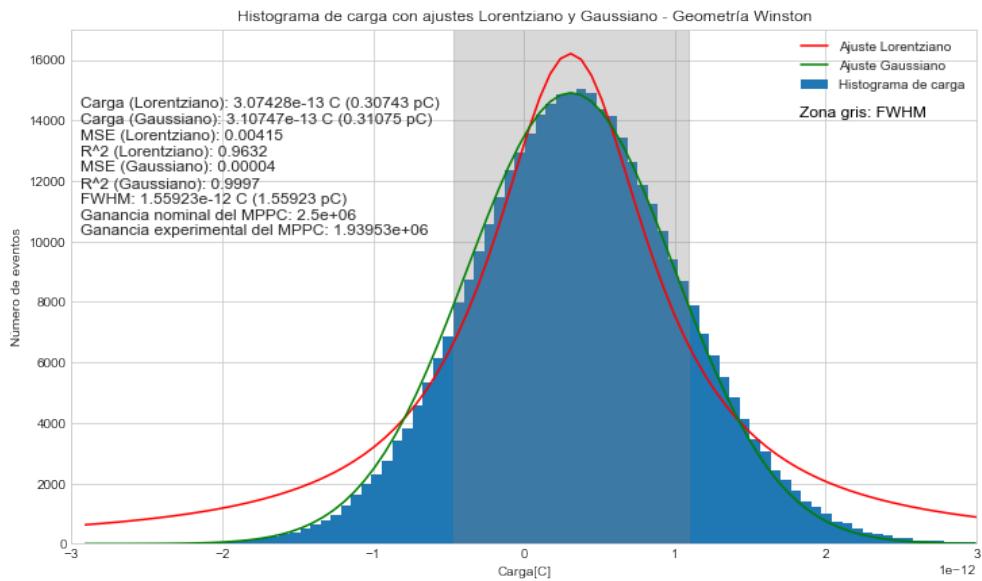
En vista de lo obtenido en la sección anterior podemos determinar cuál geometría es más eficiente a la hora de focalizar la luz. El problema de esto, es que al trabajar con valores de carga no podemos realizar una comparación del todo adecuada con respecto a la sección de simulaciones que esta expresada en fotones. Es por eso, que en esta sección se mostrará cómo determinar la carga generada de un foton detectado, de esta forma podremos convertir la carga de cada experimento a número de fotones. En primera instancia para realizar esto lo ideal sería realizar una calibración previa al detector(MPPC). Una alternativa es estimar dicha cantidad haciendo un análisis a la *baseline* de los pulsos, dicha región la denominaremos como ROI(*Region of Interest*). Primero se calcula la media y desviación de los datos en el ROI. Ahora procedemos a buscar los peaks más grandes dentro de esta región, los cuales se encontrarán entre sigma y 6 veces sigma. Una vez que fueron detectados los peaks en la ROI se procede a integrar los valores de voltaje que los componen, para así poder obtener la carga asociada a cada una. Luego se construye un histograma de cargas para los peaks detectados, ahora se hace un ajuste gaussiano y a la función lorentziana, esto con el fin de saber cual de las dos se ajusta mejor a los datos. Finalmente la carga de un fotoelectrón estará asociada al segundo parámetro de ambas funciones, esto debido a que dicho parámetro corresponde a la media del ajuste aplicado. A continuación, se presentarán los resultados obtenidos.



**Figura 6.4.1:** Pulso y peaks detectados en la ROI

### 6.4.1. Ganancia experimental - Geometría cono de Winston

A continuación, se pueden ver los 2 ajustes realizados al histograma de cargas de los peaks detectados en la ROI. Para este gráfico se utilizaron los datos obtenidos por la geometría del cono de Winston. Se hizo un plot con los parámetros obtenidos para ambos ajustes. Considerar que un MSE cercano a cero y un  $r^2$  cercano a 1 son los casos ideales(indicativos que representan un mejor ajuste). Es por este motivo que la carga más adecuada para un fotoelectrón corresponde a la entregada al realizar un ajuste gaussiano.



**Figura 6.4.2:** Histograma de cargas con ajuste, valor de carga, mse,  $r^2$  - Geometría Winston

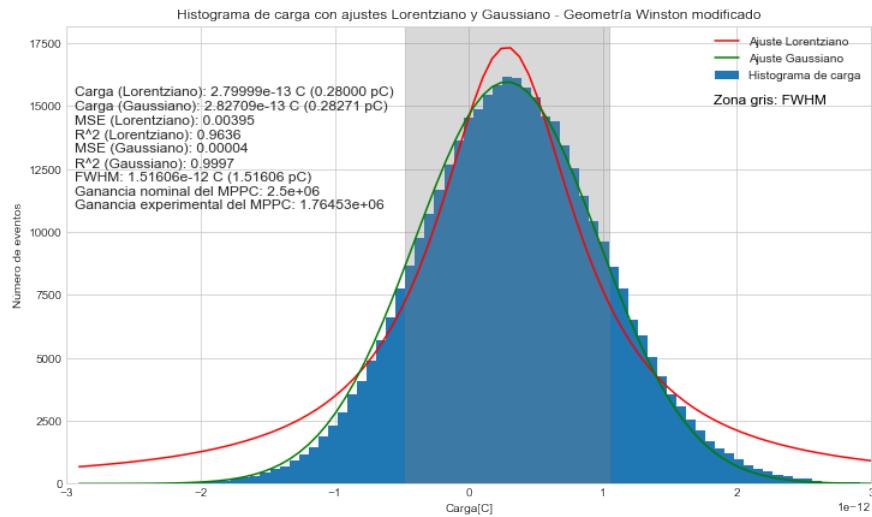
Donde la ganancia experimental fue calculada con la siguiente fórmula

$$Gain_{exp} = \frac{\text{Carga detectada de un foton}}{\text{Carga electron}} = \frac{0,31075[pC]}{1,602176634e - 19} \quad (6.4.1)$$

A continuación se realizó el mismo procedimiento pero utilizando los datos obtenidos por las geometrías del cono de winston modificado y la barra respectivamente. En este contexto es crucial obtener ganancias relativamente similares en los tres casos. Caso contrario sería indicativo de un problema en alguno de los pasos anteriormente mencionados.

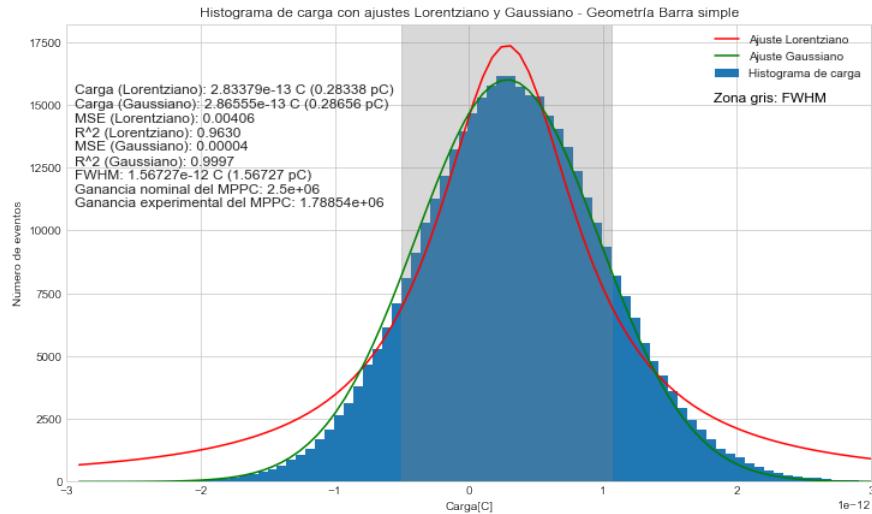
#### 486.4. Análisis de la baseline del pulso para obtener la carga de un fotoelectrón

##### 6.4.2. Ganancia experimental - Geometría cono de Winston mod



**Figura 6.4.3:** Histograma de cargas con ajuste, valor de carga, mse,  $r^2$  - Geometría Winston modificado

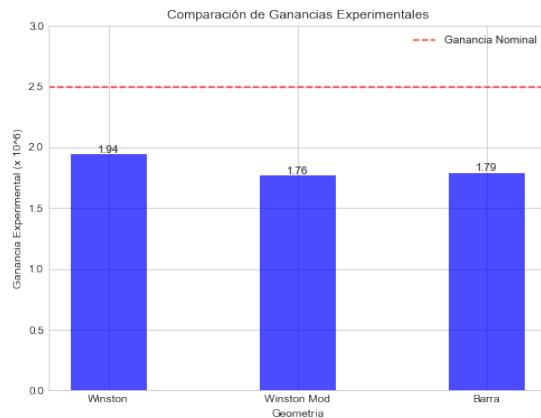
##### 6.4.3. Ganancia experimental - Geometría cono de barra simple



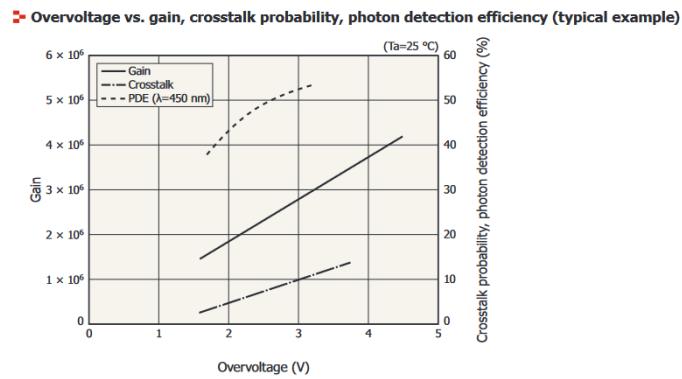
**Figura 6.4.4:** Histograma de cargas con ajuste, valor de carga, mse,  $r^2$  - Geometría barra

#### 6.4.4. Comparación ganancia experimental vs nominal

Tal como se muestra en la Figura 6.4.5, se obtuvieron tres ganancias experimentales muy similares entre sí. La línea segmentada representa la ganancia nominal indicada por el fabricante<sup>12</sup>. Es evidente que estas ganancias son altamente consistentes con la ganancia nominal. Es importante señalar que los datos se tomaron a una temperatura aproximada de 20°C en el laboratorio y con un overvoltage de 2 V. En la Figura 6.4.6, se presenta una gráfica de overvoltage vs gain del MPPC utilizado<sup>12</sup>. Esta gráfica proporciona un valor esperado para la ganancia en función del overvoltage. Como se mencionó anteriormente, bajo las condiciones específicas mencionadas, se esperaría una ganancia inferior a  $2^6$ , valor que concuerda notablemente con los resultados experimentales obtenidos.



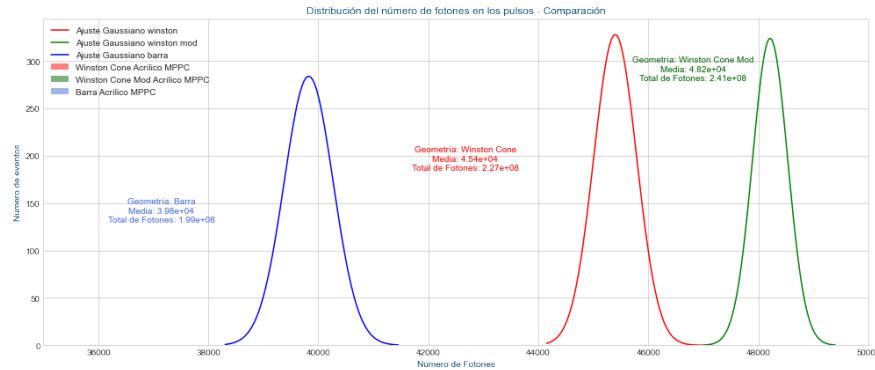
**Figura 6.4.5:** Gráfico de barras para comparar las ganancias obtenidas vs la ganancia nominal



**Figura 6.4.6:** Gráfico que muestra la relación entre ganancia y overvoltage a una temperatura ambiente de 25°C

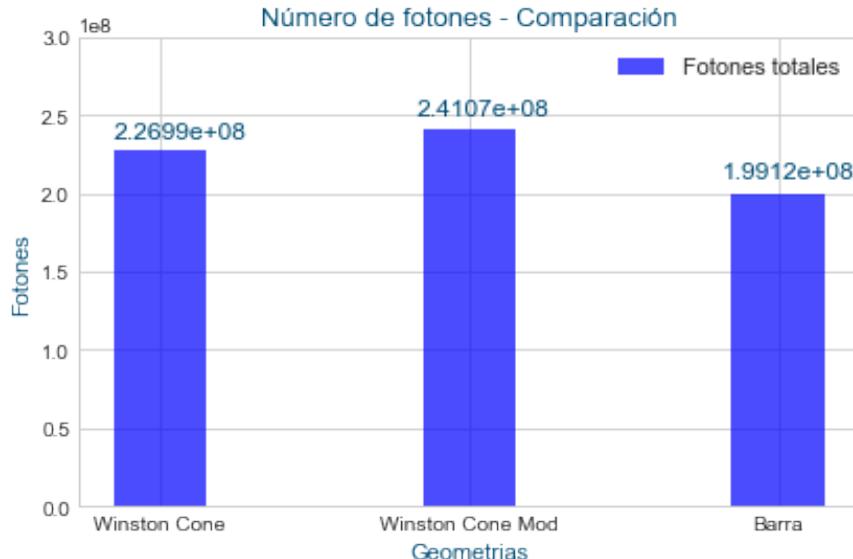
#### 506.4. Análisis de la baseline del pulso para obtener la carga de un fotoelectrón

Aplicando lo mencionado anteriormente y utilizando la carga de un fotoelectrón dado por el ajuste gaussiano de la geometría del cono Winston, se convirtió la carga de cada geometría a número de fotones, de esta manera la siguiente figura nos muestra la distribución de fotones en cada una de las geometrías. Además, se muestra la media asociada al ajuste de cada caso.



**Figura 6.4.7:** Gráfico que muestra la comparación de fotones en cada una de las geometrías

A continuación, se puede observar un gráfico de barras que representa de manera más sencilla la cantidad de fotones totales en cada geometría de estudio.



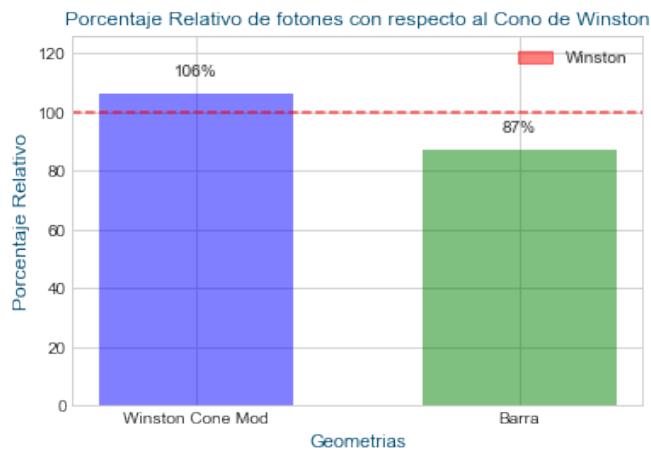
**Figura 6.4.8:** Gráfico que muestra la cantidad total de fotones en cada geometría

#### 6.4. Análisis de la baseline del pulso para obtener la carga de un fotoelectrón51

Por último podemos calcular el porcentaje relativo de fotones en la geometría del cono de Winston modificado y la barra con respecto a la geometría del cono de Winston. Para ello se utilizó la siguiente ecuación.

$$Winston\ mod = \frac{fotones\ totales\ winston\ mod}{fotones\ totales\ winston} \times 100 = \quad (106\%) \quad (6.4.2)$$

$$Barra = \frac{fotones\ totales\ barra\ mod}{fotones\ totales\ winston} \times 100 = \quad (87\%) \quad (6.4.3)$$



**Figura 6.4.9:** Gráfico que muestra el porcentaje relativo de fotones de ambas geometrías con respecto a la geometría del cono de Winston

# Capítulo 7

## Conclusion

En la sección de simulaciones se obtuvo que la geometría que mejor focaliza los fotones es la del cono de Winston modificado, obteniendo así la cantidad de 1.205.454 fotones detectados de un total de 2 millones iniciales(60 %). Esta cantidad fue muy superior a lo obtenido por la barra simple(119.247 fotones) y levemente mejor que en el cono de Winston(1.070.004 fotones). Por otro lado, la geometría que mejor tiempo de detección presenta es el cono de Winston, donde gran parte de la detección se encuentra en el rango de 0.307 - 0.309 [ns]. La segunda con mejor tiempo de detección fue el cono de Winston modificado, en el rango de 0.310 - 0.319 [ns]. Por último, la geometría de la barra presenta los peores tiempos de detección, tal como indica la figura 4.1.7 su rango temporal de detección se encuentra entre 0 y 5 [ns]. Finalmente, debido a que la diferencia temporal en la detección es tan pequeña(en el cono de Winston y el cono de Winston modificado) no es del todo relevante en la elección de una geometría a utilizar.

En la sección experimental se obtuvo que la geometría que mejor focaliza los fotones es la del cono de Winston modificado, obteniendo así la cantidad de 2.41e+08 fotones detectados vs los 2.26e+08 fotones del cono de Winston y 1.9912e+08 fotones de la barra simple. A modo de aclaración y tal como se indicó al final del capítulo 5, Las conclusiones de la sección experimental se basaron en los resultados obtenidos en el montaje 3.

Por otro lado, dada la Figura 4.1.9 de la sección de simulaciones y la Figura 6.4.9 de la sección de resultados experimentales, obtuvimos los porcentajes de mejora en la focalización de fotones para las geometrías del cono de Winston modificado y la barra, en relación con la geometría del cono de Winston. Tanto en las simulaciones como en el montaje experimental, se observó una mejora del 6 % en la geometría del cono de Winston modificado en comparación con el cono de Winston en la detección de fotones. Es notable que esta mejora porcentual obtenida a partir de las simulaciones concuerda de manera consistente con los resultados experimentales. Esta consistencia en los porcentajes puede atribuirse a una precisa técnica utilizada para determinar la carga de un fotoelectrón, así como a un ajuste adecuado (modelo gaussiano) al histograma de carga de los picos detectados en la Región de Interés (ROI).

En última instancia, podemos concluir que la geometría del cono de Winston modificado es la que logra una mejor focalización de los fotones, superando al cono de Winston en un 6 %, que inicialmente se consideraba la mejor opción. En relación a la discrepancia entre el porcentaje obtenido en la simulación y en el experimento para la barra, es posible que esta diferencia se deba a una variedad de factores que merecen un análisis más detallado.

## Referencias

- [1] Merz, J. (2015). Photon detection efficiency measurement of Winston cones for the FAMOUS telescope (Tesis de licenciatura). RWTH Aachen University.
- [2] OJEDA, N. E. C. (2017). Estudio de polarización vectorial. Chapter 1.
- [3] Grossman, E. N., Friedman, O. D., Nelson, A. O. (2013). Non-imaging Winston cone concentrators for submillimeter-wave, overmoded waveguide. *IEEE Transactions on Terahertz Science and Technology*, 4(1), 65-74.
- [4] Hénault, F., Petrucci, P. O., Jocou, L., Khélifi, B., Manigot, P., Hormigos, S., ... Punch, M. (2013, September). Design of light concentrators for Cherenkov telescope observatories. In *Nonimaging Optics: Efficient Design for Illumination and Solar Concentration X* (Vol. 8834, pp. 13-24). SPIE.
- [5] GONZALEZ, V. M. A. G. (2012). Conos concentradores de radiación para astronomía en longitudes de onda milimétricas.
- [6] Nelson, A. O., Grossman, E. N. (2014, May). Advanced designs for non-imaging submillimeter-wave Winston cone concentrators. In *Terahertz Physics, Devices, and Systems VIII: Advanced Applications in Industry and Defense* (Vol. 9102, pp. 191-201). SPIE.
- [7] Berry, H. G., Gabrielse, G., Livingston, A. E. (1977). Measurement of the Stokes parameters of light. *Applied optics*, 16(12), 3200-3205.
- [8] Singh, S. (2002). Refractive index measurement and its applications. *Physica Scripta*, 65(2), 167.
- [9] Antoni, T., Apel, W. D., Badea, A. F., Bekk, K., Bercuci, A., Blümer, H., ... KASCADE Collaboration. (2003). Measurements of attenuation and absorption lengths with the KASCADE experiment. *Astroparticle Physics*, 19(6), 703-714.
- [10] M. Bodmera, N. Phana, M. Golda, D. Loombaa, J.A.J. Matthews and K. Rielage (2014). Measurement of Optical Attenuation in Acrylic Light Guides for a Dark Matter Detector. University of New Mexico, Department of Physics and Astronomy, Albuquerque, NM 87131, USA.
- [11] Morocho, M. A. R., Hernández, C. V. (2012). Caracterización óptica de

Diodos emisores de Luz mediante su espectros de emisión y patrones de radiación. *Scientia et technica*, 2(51), 66-70.

- [12] [https://www.hamamatsu.com/content/dam/hamamatsu-photonics/sites/documents/99\\_SALES\\_LIBRARY/ssd/s14160\\_s14161\\_series\\_kapd1064e.pdf](https://www.hamamatsu.com/content/dam/hamamatsu-photonics/sites/documents/99_SALES_LIBRARY/ssd/s14160_s14161_series_kapd1064e.pdf)
- [13] Matsuoka, K., Okubo, R., Adachi, Y. (2023). Demonstration of a 25-picosecond single-photon time resolution with gaseous photomultiplication. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1053, 168378.
- [14] Qi, J., Hood, N., Kopec, A., Ma, Y., Xu, H., Zhong, M., Ni, K. (2023). Low Energy Electronic Recoils and Single Electron Detection with a Liquid Xenon Proportional Scintillation Counter. arXiv preprint arXiv:2301.12296.
- [15] Chan, H. W., Prodanović, V., Theulings, A. M. M. G., Tao, S., Smedley, J., Hagen, C. W., ... Graaf, H. (2023). The construction and characterization of MgO transmission dynodes. *Journal of Instrumentation*, 18(06), P06028.
- [16] Ieki, K., Iwamoto, T., Kobayashi, S., Mori, T., Ogawa, S., Onda, R., ... Toyoda, K. (2023). Study on degradation of VUV-sensitivity of MPPC for liquid xenon scintillation detector by radiation damage in MEG II experiment. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1053, 168365.
- [17] de Souza, H. V. (2021). ARAPUCA, light trapping device for the DUNE experiment= ARAPUCA, dispositivo de coleta de luz para o experimento DUNE (Doctoral dissertation, [sn]).
- [18] Milind Diwan. (2021). Photomultiplier tube charge and time measurement basics. BNL physics department summer lectures.
- [19] Souza, H. V., Segreto, E., Machado, A. A., Sarmento, R. R., Bazetto, M. C. Q., Paulucci, L., ... Ayala-Torres, M. A. (2021). Liquid argon characterization of the X-ARAPUCA with alpha particles, gamma rays and cosmic muons. *Journal of instrumentation*, 16(11), P11002.

# Apéndice A

## A0.1. Código Simulaciones

```
import pandas as pd
from scipy import stats
import numpy as np
import scipy
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import statsmodels.api as sm
import statsmodels.formula.api as smf
import seaborn as sns
#from scipy import random
from scipy.stats import norm
import warnings
from itertools import repeat
warnings.filterwarnings("ignore", category=DeprecationWarning)
plt.style.use('seaborn-whitegrid') # Gráficos estilo seaborn

# Cono de Winston
df1 = pd.read_csv("winston_VF.csv", sep = ',') # sep = ',' sirve
para separar en 2 columnas el excel leido.

lista_columnas1=[]
for i in df1.iloc[:,1:]:
    if df1[i][5000]!=df1[i][4999]:
        lista_columnas1.append(i)

for i in range(374):
    df1[lista_columnas1[i]]=np.NAN
    i=i+1
```

```
df1.iloc[[5000]]  
  
lista_1=df1.iloc[[5000]].mean()  
  
df_1 = pd.DataFrame()  
  
df_1['List1']=lista_1  
  
df_1= df_1.drop(df_1.index[[0]])  
  
df_1['List1'].sum()  
  
Winston=5355.023442220399  
  
# Cono de Winston_mod  
  
df2 = pd.read_csv("winston_mod_VF.csv", sep = ',') # sep = ','  
          sirve para separar en 2 columnas el excel leido.  
  
Lista_null=df2.columns[df2.isna().any()].tolist()  
  
df2[Lista_null]=np.NAN  
  
df2.iloc[[319]]  
  
lista_2=df2.iloc[[319]].mean()  
  
df_2 = pd.DataFrame()  
  
df_2['List2']=lista_2  
  
df_2= df_2.drop(df_2.index[[0]])  
  
df_2['List2'].sum()  
  
Winston_mod=6022.270808689949  
  
## Barra  
  
df3 = pd.read_csv("barra_VF.csv", sep = ',') # sep = ',' sirve  
          para separar en 2 columnas el excel leido.
```

```
lista_columnas3=[]
for i in df3.iloc[:,1:]:
    if df3[i][5000]!=df3[i][4999]:
        lista_columnas3.append(i)

for i in range(1417):
    df3[lista_columnas3[i]]=np.NAN
    i=i+1

df3.iloc[[5000]]

lista_3=df3.iloc[[5000]].mean()

df_3 = pd.DataFrame()

df_3['List3']=lista_3

df_3= df_3.drop(df_3.index[[0]])

df_3['List3'].sum()

Barra=596.23591625573

# Comparacion energ a promedio en el detector de las 3
geometrias

list=(Winston,Winston_mod,Barra)

lista_name=['Winston','Winston_mod','Barra']

df = pd.DataFrame()

df['Geometr a']=lista_name

df['Energ a Promedio']=list

facecolor = '#EAEAEA'
color_bars = '#3475D0'
txt_color1 = '#252525'
txt_color2 = '#004C74'
# Establecer el tama o de la figura
```

```

plt.rcParams["figure.figsize"] = (6, 4)
df.plot(x='Geometria', y='Energia Promedio', kind="bar", width
=0.3, align='center', edgecolor='black', label='Energia total
[w]')
plt.tick_params(axis="x", rotation=0, labelsize=10)
plt.xlabel('Geometrias', fontsize=10, horizontalalignment='center
', position=(0.5,-0.5), c=txt_color2) # Titulo eje x
plt.ylabel('Energia[w]', fontsize=10, c=txt_color2)
plt.ylim(0,7000)
plt.title('Comparacion de energia total recibida en el rea
sensible para distintas geometrias', fontsize=10) # Titulo de
la grafica
plt.legend(prop={'size': 10})
plt.text(-0.14,5500, '5355,0234 [w]', fontsize = 10, c=txt_color2
)
plt.text(0.84,6200, '6022,2708 [w]', fontsize = 10, c=txt_color2)
plt.text(1.84,700, '596,2359 [w]', fontsize = 10, c=txt_color2)
# Ajustar el espacio entre las barras
plt.tight_layout()
plt.show()

df_=pd.DataFrame()

df_['Winston_Cone']=df_1
df_['Winston_Cone_mod']=df_2
df_['Barra_Simple']=df_3


data = pd.DataFrame()

data['Winston_Cone']=lista_1
data['Winston_Cone_mod']=lista_2
data['Barra_Simple']=lista_3

data=data.drop(data.index[[0]])

data = data.fillna(0)

color_bars = '#3475D0'
txt_color1 = '#252525'
txt_color2 = '#004C74'

```

```
plt.figure(figsize = (8,4))
sns.histplot(data,bins=100,shrink=1, color=('r','a','p'),alpha
=0.6,element='step', edgecolor='black', linewidth=0.5)
plt.xlabel('\nEnerg a [W]', c=txt_color2, fontsize=10)
plt.ylabel('N mero de Rayos', c=txt_color2, fontsize=10)
plt.title('Distribuci n de energ a total de rayos para
distintas geometrias',c=txt_color2, fontsize=12)
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 10)

# Crear una lista de etiquetas para la leyenda
legend_labels = ["Barra", "Winston_Cone_mod", "Winston_Cone"]
plt.legend(legend_labels, loc='upper center')

plt.show()

facecolor = '#EAEAEA'
color_bars = '#3475D0'
txt_color1 = '#252525'
txt_color2 = '#004C74'
plt.figure(figsize = (20,7))
sns.histplot(data,bins=1500, color=('r','a','p'),alpha=0.6,
edgecolor='black', linewidth=0.5)
#plt.xlim(3.48,3.62)
plt.xlim(3.225,3.375)
plt.ylim(0,350)
plt.xlabel('\nEnerg a en el detector [W]', c=txt_color2,
fontsize=21)
plt.ylabel('N mero de Rayos', c=txt_color2, fontsize=21)
plt.title('Distribuci n de energ a de los rayos detectados en
distintas geometrias(Energ a min > 3[W])',fontsize=20,loc=
'center', c=txt_color2)
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
#plt.legend(df_,prop={'size': 15})

facecolor = '#EAEAEA'
color_bars = '#3475D0'
txt_color1 = '#252525'
txt_color2 = '#004C74'
plt.figure(figsize = (10,5))
```

```
sns.histplot(data=df_1, x='List1', bins=40, alpha=0.6, edgecolor='black', linewidth=0.5)
plt.ylim(0,100)
#plt.xlim(3.2,3.37)
plt.xlabel('\nEnerg a en el detector [W]', c=txt_color2, fontsize=10)
plt.ylabel('N mero de Rayos', c=txt_color2, fontsize=10)
plt.title('Distribuci n de los rayos detectados en el Cono de Winston', fontsize=12, loc='center', c=txt_color2)
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 10)

facecolor = '#EAEAEA'
color_bars = '#3475D0'
txt_color1 = '#252525'
txt_color2 = '#004C74'

plt.figure(figsize = (20,7))
sns.histplot(data=df_1, x='List1', bins=1500, alpha=0.6, edgecolor='black', linewidth=0.5)
# plt.ylim(0,600)
plt.xlim(3.29,3.37)
plt.xlabel('\nEnerg a en el detector [W]', c=txt_color2, fontsize=18)
plt.ylabel('N mero de Rayos', c=txt_color2, fontsize=18)
plt.title('Distribuci n de energ a de los rayos detectados en el Cono de Winston(Energ a min > 3[W])', fontsize=20, loc='center', c=txt_color2)
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)

facecolor = '#EAEAEA'
color_bars = '#3475D0'
txt_color1 = '#252525'
txt_color2 = '#004C74'

plt.figure(figsize = (20,7))
sns.histplot(data=df_2, x='List2', bins=50, alpha=0.6, edgecolor='black', linewidth=0.5)
# plt.ylim(0,600)
# plt.xlim(3.2,3.37)
plt.xlabel('\nEnerg a en el detector [W]', c=txt_color2, fontsize=18)
plt.ylabel('N mero de Rayos', c=txt_color2, fontsize=18)
```

```
plt.title('Distribucion de energia de los rayos detectados en  
el Cono de Winston modificado', fontsize=20, loc='center', c=  
txt_color2)  
plt.xticks(fontsize = 18)  
plt.yticks(fontsize = 18)  
  
facecolor = '#EAEAEA'  
color_bars = '#3475D0'  
txt_color1 = '#252525'  
txt_color2 = '#004C74'  
plt.figure(figsize = (20,7))  
sns.histplot(data=df_3, x='List3', bins=50, alpha=0.6, edgecolor='  
black', linewidth=0.5)  
#plt.ylim(0,600)  
#plt.xlim(3.2,3.37)  
plt.xlabel('\nEnergia en el detector [W]', c=txt_color2,  
fontsize=18)  
plt.ylabel('Número de Rayos', c=txt_color2, fontsize=18)  
plt.title('Distribucion de energia de los rayos detectados en  
la Barra', fontsize=20, loc='center', c=txt_color2)  
plt.xticks(fontsize = 18)  
plt.yticks(fontsize = 18)  
  
## Trabajando con los tiempos de llegada al detector  
  
df1_t=pd.DataFrame()  
  
df1_t=df1  
  
df1_t=df1_t.dropna(axis=1)  
  
p1 = df1_t.nunique()  
print(p1)  
  
df2_t=pd.DataFrame()  
  
df2_t=df2  
  
df2_t=df2_t.dropna(axis=1)
```

```
p2 = df2_t.nunique()
print(p2)

df3_t=pd.DataFrame()

df3_t=df3

df3_t=df3_t.dropna(axis=1)

p3 = df3_t.nunique()
print(p3)

lista_ind_1=[]
for i in range(1,1627):
    val_indice1=p1.values[i]-1
    lista_ind_1.append(df1['Tiempo (ns)'][val_indice1])
    i=i+1

lista_ener_1=[]
for j in range(1,1628):
    for i in range(1,1627):
        val_indice1=p1.values[i]-1
        lista_ener_1.append(df1['Rayo '+ ' '+repr(j)][val_indice1])
        i=i+1
    j=j+1

lista_ind_1 = (lista_ind_1 + 2000 * [0])[:2000]

lista_ind_2=[]
for i in range(1,1831):
    val_indice2=p2.values[i]-1
    lista_ind_2.append(df2['Tiempo (ns)'][val_indice2])
    i=i+1

lista_ind_2 = (lista_ind_2 + 2000 * [0])[:2000]

lista_ind_3=[]
for i in range(1,584):
    val_indice=p3.values[i]-1
    lista_ind_3.append(df3['Tiempo (ns)'][val_indice])
    i=i+1
```

```
lista_ind_3 = (lista_ind_3+ 2000 * [0])[:2000]

df_t = pd.DataFrame()

df_t['Tiempo de llegada Winston Cone']=lista_ind_1
df_t['Tiempo de llegada Winston Cone_mod']=lista_ind_2
df_t['Tiempo de llegada Barra']=lista_ind_3

df_t= df_t.drop(df_t.index[[0]])

plt.figure(figsize = (20,7))
sns.histplot(df_t,bins=100000,binwidth=0.2,kde=False,element='
    step',alpha=0.5,legend=True)
plt.xticks(np.arange(0.2,4.8,0.2))
plt.xlim(0.2,4.8)
plt.xlabel('\nTiempo de llegada al detector [ns]', c=txt_color2,
    fontsize=22)
plt.ylabel('N mero de Rayos', c=txt_color2, fontsize=22)
plt.title('Comparaci n de tiempo de llegada de los rayos
    detectados en distintas geometrias',fontsize=22,loc='center',
    c=txt_color2)
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
#plt.legend(df_t,prop={'size': 20})
plt.tight_layout()

plt.figure(figsize = (20,7))
sns.histplot(df_t,bins=1000000,binwidth=0.0005,kde=False,element='
    step',alpha=0.4,legend=True)
plt.xlim(0.3025,0.3186)
plt.ylim(0,600)
plt.xticks(np.arange(0.3025,0.3186,0.001))
plt.xlabel('\nTiempo de llegada al detector [ns]', c=txt_color2,
    fontsize=22)
plt.ylabel('N mero de Rayos', c=txt_color2, fontsize=22)
plt.title('Comparaci n de tiempo de llegada de los rayos
    detectados en distintas geometrias(Tiempo max < 0.4[ns])',
    fontsize=22,loc='center', c=txt_color2)
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
#plt.legend(df_t,prop={'size': 20})
plt.tight_layout()
```

```
## Tiempo de llegada vs numero de fotones

df_f1 = pd.DataFrame()
df_f2 = pd.DataFrame()
df_f3 = pd.DataFrame()

dft_f = pd.DataFrame()

Lista_frec=df_t['Tiempo de llegada Winston Cone'].value_counts()
Lista_frec2=df_t['Tiempo de llegada Winston Cone_mod'].
    value_counts()
Lista_frec3=df_t['Tiempo de llegada Barra'].value_counts()

Lista_frec=[]
Lista_frec2=[]
Lista_frec3=[]

df_frec=pd.DataFrame()
df_frec2=pd.DataFrame()
df_frec3=pd.DataFrame()

df_ener_1=pd.DataFrame()

df_frec2['Tiempo de llegada Winston Cone_mod']=Lista_frec2.index
df_frec2['Frecuencia']=Lista_frec2.values

df_frec3['Tiempo de llegada Barra']=Lista_frec3.index
df_frec3['Frecuencia']=Lista_frec3.values

df_frec['Tiempo de llegada Winston Cone']=Lista_frec.index
df_frec['Frecuencia']=Lista_frec.values

lista_ener_1=[]

for j in range(28):
    for i in df1_t['Tiempo (ns)']:
        if i == df_frec['Tiempo de llegada Winston Cone'][j]:
            lista_ener_1.append(df1_t['Rayo 1998'][df1_t['Tiempo
(ns)']==i].values)
        else:
            i=i+1
```

```
j=j+1

lista_ener_2=[]

for j in range(18):
    for i in df2_t['Tiempo (ns)']:
        if i == df_frec2['Tiempo de llegada Winston Cone_mod'][j]:
            lista_ener_2.append(df2_t['Rayo 2000'][df2_t['Tiempo (ns)']==i].values)
        else:
            i=i+1
    j=j+1

df_frec['Energias']=lista_ener_1
df_frec2['Energias']=lista_ener_2
df_frec3['Energias']=lista_ener_3

df_frec['Energias']=df_frec['Energias'].astype(float)
df_frec2['Energias']=df_frec2['Energias'].astype(float)
df_frec3['Energias']=df_frec3['Energias'].astype(float)

df_p3=pd.DataFrame()

df_p3['rayos']=p3.index
df_p3['frecuencia']=p3.values

df_p3.sort_values(by='frecuencia',inplace=True)

lista_ener_3=[]

for j in range(129):
    for i in df3_t['Tiempo (ns)']:
        if i == df_frec3['Tiempo de llegada Barra'][j]:
            lista_ener_3.append(df3_t['Rayo 1959'][df3_t['Tiempo (ns)']==i].values)
        else:
            i=i+1
    j=j+1

### Conversion a fotones
```

```

df_frec['fotones']=(df_frec['Energias']*1000*df_frec['Frecuencia']
])/5

df_frec2['fotones']=(df_frec2['Energias']*1000*df_frec2['
Frecuencia'])/5

df_frec3['fotones']=(df_frec3['Energias']*1000*df_frec3['
Frecuencia'])/5

df_tiempos=pd.DataFrame()
df_tiempo_winston=pd.DataFrame()
df_tiempo_winston_mod=pd.DataFrame()
df_tiempo_barra=pd.DataFrame()

df_tiempo_winston['Tiempo de llegada Winston Cone']=np.repeat(
    df_frec['Tiempo de llegada Winston Cone'],df_frec['fotones'])
df_tiempo_winston_mod['Tiempo de llegada Winston Cone_mod']=np.
repeat(df_frec2['Tiempo de llegada Winston Cone_mod'],df_frec2
['fotones'])
df_tiempo_barra['Tiempo de llegada Barra']=np.repeat(df_frec3['
Tiempo de llegada Barra'],df_frec3['fotones'])

df_tiempo_winston=df_tiempo_winston.reset_index(drop=True)
df_tiempo_winston_mod=df_tiempo_winston_mod.reset_index(drop=True
)
df_tiempo_barra=df_tiempo_barra.reset_index(drop=True)

df_tiempos['Tiempo de llegada Winston Cone']=df_tiempo_winston['
Tiempo de llegada Winston Cone']
df_tiempos['Tiempo de llegada Winston Cone_mod']=
df_tiempo_winston_mod['Tiempo de llegada Winston Cone_mod']
df_tiempos['Tiempo de llegada Barra']=df_tiempo_barra['Tiempo de
llegada Barra']

df_tiempos=df_tiempos.reset_index(drop=True)

plt.figure(figsize = (20,7))
sns.histplot(df_tiempos,bins=100000,binwidth=0.0005,kde=False,
element='step',alpha=0.5,legend=True)
plt.xticks(np.arange(0.3,0.324,0.001))
plt.xlim(0.3,0.324)

```

```
plt.ylim(0,400000)
plt.xlabel('\nTiempo de llegada al detector [ns]', c=txt_color2,
           fontsize=22)
plt.ylabel('Número de Fotones [miles]', c=txt_color2, fontsize
           =22)
plt.title('Comparación de tiempo de llegada de los rayos
           detectados en distintas geometrías 0.3<T<0.32', fontsize=22, loc
           ='center', c=txt_color2)
plt.xticks(fontsize = 16)
plt.yticks(fontsize = 16)
# plt.legend(df_t, prop={'size': 20})
plt.tight_layout()
plt.text(0.310,350000,'Fotones iniciales para cada Geometría =
2.000.000 = 100%', fontsize = 17, c='black')
plt.text(0.311,290000,'$\sum$ Fotones Winston Cone = 1.070.004 =
54%', fontsize = 15, c=txt_color2)
plt.text(0.311,260000, '$\sum$ Fotones Winston Cone Mod =
1.204.454 = 60%', fontsize = 15, c='orange')
plt.text(0.311,230000,'$\sum$ Fotones Barra Simple = 119.247 = 6%
', fontsize = 15, c='green')

## Energía como fotones

lista_valor1=[]
for i in range(2000):
    valor1=(data['Winston_Cone'][i]*1000)/5
    lista_valor1.append(valor1)
    i=i+i

lista_valor2=[]
for i in range(2000):
    valor2=(data['Winston_Cone_mod'][i]*1000)/5
    lista_valor2.append(valor2)
    i=i+i

lista_valor3=[]
for i in range(2000):
    valor3=(data['Barra_Simple'][i]*1000)/5
    lista_valor3.append(valor3)
    i=i+i

df_fotones = pd.DataFrame()
```

```
rayos=[]

for k in range(1,2001):
    rayos.append('Rayo' + repr(k))

df_fotones['Rayos']=rayos
df_fotones['Número_de_fotones_finales_Winston_Cone']=
    lista_valori
df_fotones['Número_de_fotones_finales_Winston_Cone_mod']=
    lista_valor2
df_fotones['Número_de_fotones_finales_Barra_Simple']=
    lista_valor3

fotones1=df_fotones['Número_de_fotones_finales_Winston_Cone'].sum()

df_fotones['Número_de_fotones_finales_Winston_Cone'].sum()

fotones2=df_fotones['Número_de_fotones_finales_Winston_Cone_mod'].sum()

df_fotones['Número_de_fotones_finales_Winston_Cone_mod'].sum()

fotones3=df_fotones['Número_de_fotones_finales_Barra_Simple'].sum()

df_fotones['Número_de_fotones_finales_Barra_Simple'].sum()

df_f_mean = pd.DataFrame()

list_m=(fotones1,fotones2,fotones3)

lista_name_m=['Fotones_Winston','Fotones_Winston_mod','
    Fotones_Barra']

df_f_mean['Geometría']=lista_name_m
df_f_mean['Fotones_Finales']=list_m

plt.rcParams["figure.figsize"] = (5,3)
df_f_mean.plot(x='Geometría', y='Fotones_Finales', kind="bar",
    width=0.3)
```

```
plt.tick_params(axis="x", rotation=0, labelsize=10)

plt.xlabel('Geometrias', fontsize=13, horizontalalignment='center',
           position=(0.5,-0.5), c=txt_color2) # Titulo eje x
plt.ylabel('Número de Fotones', fontsize=14, c=txt_color2) #
    Titulo eje x
plt.ylim(0,1.4e6)
facecolor = '#EAEAEA'
color_bars = '#3475D0'
txt_color1 = '#252525'
txt_color2 = '#004C74'
plt.title('Comparación de número de fotones total detectados
para distintas geometrias', fontsize=11, c=txt_color2) #
    Titulo de la grafica
plt.legend(prop={'size': 7})
plt.text(-0.15,1.09e6, '1.070.004 Fotones', fontsize = 10, c=
    txt_color2)
plt.text(0.85,1.22e6, '1.204.454 Fotones', fontsize = 10, c=
    txt_color2)
plt.text(1.85,0.14e6, '119.247', fontsize = 10, c=txt_color2)
```

**Listing A.1:** Simulaciones

## A0.2. Código Montaje 1

```

import pandas as pd
import numpy as np
import scipy
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf
import seaborn as sns
from scipy import random
import warnings
from itertools import repeat
warnings.filterwarnings("ignore", category=DeprecationWarning)
plt.style.use('seaborn-whitegrid') # Gr ficos estilo seaborn
plt.rcParams["figure.figsize"] = (12, 4) # Tama o gr ficos
plt.rcParams["figure.dpi"] = 100 # resoluci n gr ficos

df_winston = pd.read_csv("winston_1khz_5.7v_PMT.csv", sep = ',')
# sep = ',' sirve para separar en 2 columnas el excel leido.

df_winston_mod = pd.read_csv("winston_mod_1khz_5.7v_0ns_delay_PMT
.csv", sep = ',') # sep = ',' sirve para separar en 2
columnas el excel leido.

df_barra = pd.read_csv("rec_1khz_5.7v_PMT.csv", sep = ',') # sep
= ',' sirve para separar en 2 columnas el excel leido.

df_winston['Time'] -= df_winston['Time'][0]
df_winston_mod['Time'] -= df_winston_mod['Time'][0]
df_barra['Time'] -= df_barra['Time'][0]

plt.scatter(df_winston['Time'],df_winston['Voltaje'])
plt.xlabel("Voltaje")
plt.ylabel("Tiempo")
plt.title("Gr fico para los 5000 pulsos -Winston_Resina_PMT")
plt.show()

plt.scatter(df_winston_mod['Time'],df_winston_mod['Voltaje'])
plt.xlabel("Voltaje")
plt.ylabel("Tiempo")

```

```
plt.title("Gráfico para los 5000 pulsos -Winston_mod_Resina_PMT")
)
plt.show()

plt.scatter(df_barra['Time'],df_barra['Voltaje'])
plt.xlabel("Voltaje")
plt.ylabel("Tiempo")
plt.title("Gráfico para los 5000 pulsos -Barra_Resina_PMT")
plt.show()

## Integración de datos para los 5000 pulsos

pulsos=[]
for i in range(1,5001):
    pulsos.append(i)

### Winston

import scipy.integrate

d = 1250
lista2 = []

num_pulses = len(df_winston) // d

for i in range(num_pulses):
    datosV = df_winston['Voltaje'][i*d : (i+1)*d]
    datosT = df_winston['Time'][i*d : (i+1)*d]
    valor = scipy.integrate.trapz(datosV, datosT)
    lista2.append(valor)

### Winston mod

import scipy.integrate

d = 1250
lista2_mod = []

num_pulses = len(df_winston_mod) // d

for i in range(num_pulses):
    datosV = df_winston_mod['Voltaje'][i*d : (i+1)*d]
```

```

    datosT = df_winston_mod['Time'][i*d : (i+1)*d]
    valor = scipy.integrate.trapz(datosV, datosT)
    lista2_mod.append(valor)

    ### Barra

import scipy.integrate

d = 1250
lista2_barra = []

num_pulses = len(df_barra) // d

for i in range(num_pulses):
    datosV = df_barra['Voltaje'][i*d : (i+1)*d]
    datosT = df_barra['Time'][i*d : (i+1)*d]
    valor = scipy.integrate.trapz(datosV, datosT)
    lista2_barra.append(valor)

df1_winston=pd.DataFrame()

df1_winston_mod=pd.DataFrame()

df1_barra=pd.DataFrame()

df1_winston['Pulsos']=pulsos
df1_winston['integral V']=lista2
df1_winston_mod['Pulsos']=pulsos
df1_winston_mod['integral V']=lista2_mod
df1_barra['Pulsos']=pulsos
df1_barra['integral V']=lista2_barra

df1_winston.rename(columns={'integral V': 'Carga'}, inplace=True)
df1_winston_mod.rename(columns={'integral V': 'Carga'}, inplace=True)
df1_barra.rename(columns={'integral V': 'Carga'}, inplace=True)

df1_winston['Carga']=df1_winston['Carga']*-1
df1_winston_mod['Carga']=df1_winston_mod['Carga']*-1
df1_barra['Carga']=df1_barra['Carga']*-1

carga_total_winston=df1_winston['Carga'].sum()

```

```
carga_total_winston_mod=df1_winston_mod['Carga'].sum()

carga_total_barra=df1_barra['Carga'].sum()

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
plt.figure(figsize=(8, 4))
# Crear el histograma de la distribución de carga total
hist, bin_edges = np.histogram(df1_winston['Carga'], bins=50,
                                density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
plt.bar(bin_centers, hist, width=np.diff(bin_edges), color='blue',
        alpha=0.7, label='Datos Experimentales')

# Definir la función Gaussiana para el ajuste
def gaussian(x, amplitude, mean, stddev):
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))

# Estimar los valores iniciales para el ajuste
amplitude_guess = hist.max()
mean_guess = df1_winston['Carga'].mean()
stddev_guess = df1_winston['Carga'].std()

# Ajustar la función Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[amplitude_guess, mean_guess, stddev_guess])
x = np.linspace(df1_winston['Carga'].min(), df1_winston['Carga'].max(), 100)
plt.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano')

text_x = popt[1] - 3* popt[2]
text_y = 0.8 * hist.max()
plt.text(text_x, text_y, f'Desviación estándar: {popt[2]:.4e}', color='b', ha='center', va='center')

plt.xlabel('Carga (pC)')
```

```
plt.ylabel('Frecuencia')
plt.title('Distribuci n de carga total en los pulsos -
Winston_Cone_Resina_PMT')
plt.legend()
plt.grid(True)
plt.show()

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
plt.figure(figsize=(8, 4))
# Crear el histograma de la distribuci n de carga total
hist, bin_edges = np.histogram(df1_winston_mod['Carga'], bins=50,
density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
plt.bar(bin_centers, hist, width=np.diff(bin_edges), color='blue'
, alpha=0.7, label='Datos Experimentales')

# Definir la funci n Gaussiana para el ajuste
def gaussian(x, amplitude, mean, stddev):
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))

# Estimar los valores iniciales para el ajuste
amplitude_guess = hist.max()
mean_guess = df1_winston_mod['Carga'].mean()
stddev_guess = df1_winston_mod['Carga'].std()

# Ajustar la funci n Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[amplitude_guess, mean_guess, stddev_guess])
x = np.linspace(df1_winston_mod['Carga'].min(), df1_winston_mod['Carga'].max(), 100)
plt.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano')

text_x = popt[1] - 2* popt[2]
text_y = 0.8 * hist.max()
plt.text(text_x, text_y, f'Desviaci n est ndar: {popt[2]:.4e}',
color='b', ha='center', va='center')

plt.xlabel('Carga (pC)')
```

```
plt.ylabel('Frecuencia')
plt.title('Distribuci n de carga total en los pulsos -
Winston_Cone_mod_Resina_PMT')
plt.legend()
plt.grid(True)
plt.show()

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
plt.figure(figsize=(8, 4))
# Crear el histograma de la distribuci n de carga total
hist, bin_edges = np.histogram(df1_barra['Carga'], bins=50,
density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
plt.bar(bin_centers, hist, width=np.diff(bin_edges), color='blue'
, alpha=0.7, label='Datos Experimentales')

# Definir la funci n Gaussiana para el ajuste
def gaussian(x, amplitude, mean, stddev):
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))

# Estimar los valores iniciales para el ajuste
amplitude_guess = hist.max()
mean_guess = df1_barra['Carga'].mean()
stddev_guess = df1_barra['Carga'].std()

# Ajustar la funci n Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[
    amplitude_guess, mean_guess, stddev_guess])
x = np.linspace(df1_barra['Carga'].min(), df1_barra['Carga'].max()
(), 100)
plt.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano')

text_x = popt[1] - 3* popt[2]
text_y = 0.8 * hist.max()
plt.text(text_x, text_y, f'Desviaci n est ndar: {popt[2]:.4e}',
color='b', ha='center', va='center')

plt.xlabel('Carga (pC)')
plt.ylabel('Frecuencia')
```

```
plt.title('Distribuci n de carga total en los pulsos -  
barra_Resina_PMT')  
plt.legend()  
plt.grid(True)  
plt.show()  
  
import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
from scipy.optimize import curve_fit  
  
txt_color2 = '#004C74'  
  
# Definir la funci n Gaussiana para el ajuste  
def gaussian(x, amplitude, mean, stddev):  
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))  
  
fig, ax = plt.subplots(figsize=(10, 6))  
  
# Datos del primer gr fico  
hist, bin_edges = np.histogram(df1_winston['Carga'], bins=50,  
                               density=True)  
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2  
ax.bar(bin_centers, hist, width=np.diff(bin_edges), color='red',  
       alpha=0.5, label='Winston Cone Resina PMT')  
  
# Ajustar la funci n Gaussiana a los datos  
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[hist.max(),  
                                                    df1_winston['Carga'].mean(), df1_winston['Carga'].std()])  
x = np.linspace(df1_winston['Carga'].min(), df1_winston['Carga'].  
                 max(), 100)  
ax.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano  
winston')  
text_x = popt[1] - 19 * popt[2]  
text_y = 0.7 * hist.max()  
ax.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='r', ha='center', va='center')  
  
# Datos del segundo gr fico  
hist, bin_edges = np.histogram(df1_winston_mod['Carga'], bins=50,  
                               density=True)
```

```
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
ax.bar(bin_centers, hist, width=np.diff(bin_edges), color='darkgreen', alpha=0.5, label='Winston Cone Mod Resina PMT')

# Ajustar la funci n Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[hist.max(),
    df1_winston_mod['Carga'].mean(), df1_winston_mod['Carga'].std()])
x = np.linspace(df1_winston_mod['Carga'].min(), df1_winston_mod['Carga'].max(), 100)
ax.plot(x, gaussian(x, *popt), 'g-', label='Ajuste Gaussiano
winston mod')
text_x = popt[1] - 15.5* popt[2]
text_y = 0.6 * hist.max()
ax.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='g', ha='center', va='center')

# Datos del tercer gr fico
hist, bin_edges = np.histogram(df1_barra['Carga'], bins=50,
    density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
ax.bar(bin_centers, hist, width=np.diff(bin_edges), color='royalblue', alpha=0.5, label='Barra Resina PMT')

# Ajustar la funci n Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[hist.max(),
    df1_barra['Carga'].mean(), df1_barra['Carga'].std()])
x = np.linspace(df1_barra['Carga'].min(), df1_barra['Carga'].max(), 100)
ax.plot(x, gaussian(x, *popt), 'b-', label='Ajuste Gaussiano
barra')
text_x = popt[1] - 3.25* popt[2]
text_y = 0.63* hist.max()
ax.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='blue', ha='center', va='center')

# Configurar la leyenda
ax.set_xlabel('Carga', c=txt_color2)
ax.set_ylabel('Frecuencia', c=txt_color2)
ax.set_title('Distribuci n de carga total en los pulsos -
Comparaci n', c=txt_color2)
ax.legend()
```

```

ax.grid(True)
plt.show()

### Comparacion de carga promedio en las 3 geometrias

df_comp = pd.DataFrame()

Winston=carga_total_winston
Winston_mod=carga_total_winston_mod
Barra=carga_total_barra

list=(Winston,Winston_mod,Barra)

lista_name=['Winston','Winston_mod','Barra']

df_comp['Geometria']=lista_name
df_comp['Carga Promedio']=list

df_comp.plot(x='Geometria', y='Carga Promedio', kind="bar",width=0.3)
plt.tick_params(axis="x", rotation=0, labelsize=10)
plt.rcParams["figure.figsize"] = (6, 4)
plt.xlabel('Geometrias', fontsize=13, horizontalalignment='center',
           position=(0.5,-0.5), c=txt_color2) # Titulo eje x
plt.ylabel('Carga', fontsize=13,c=txt_color2)
plt.title('Comparacion de carga total para distintas geometrias - PMT', fontsize=13, c=txt_color2) # Titulo de la grafica
plt.legend(prop={'size': 10})
plt.legend(labels=['Carga total'], prop={'size': 10}) # Renombrar la leyenda
plt.ylim(0,4e-5)
plt.text(1.8,1.7e-05,"{:.4e}".format(carga_total_barra), fontsize=10,c=txt_color2)
plt.text(-0.2,3.4e-05,"{:.4e}".format(carga_total_winston),
         fontsize=10,c=txt_color2)
plt.text( 0.8,3.1e-05,"{:.4e}".format(carga_total_winston_mod),
         fontsize=10,c=txt_color2)

```

**Listing A.2:** Montaje 1

### A0.3. Código Montaje 2

```
import pandas as pd
import numpy as np
import scipy
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf
import seaborn as sns
from scipy import random
import warnings
from itertools import repeat
warnings.filterwarnings("ignore", category=DeprecationWarning)
plt.style.use('seaborn-whitegrid') # Gráficos estilo seaborn
plt.rcParams["figure.figsize"] = (12, 4) # Tamaño de los gráficos
plt.rcParams["figure.dpi"] = 100 # Resolución de los gráficos

df_winston = pd.read_csv("2khz_3.5v_winston_MPPC.csv", sep = ',')
    # sep = ',' sirve para separar en 2 columnas el excel leido.

df_winston_mod = pd.read_csv("2khz_3.5v_out_winston_mod_MPPC.csv"
    , sep = ',') # sep = ',' sirve para separar en 2 columnas el
    # excel leido.

df_barra = pd.read_csv("2khz_3.5v_cuadrado_MPPC.csv", sep = ',')
    # sep = ',' sirve para separar en 2 columnas el excel leido.

df_winston['Time'] -= df_winston['Time'][0]
df_winston_mod['Time'] -= df_winston_mod['Time'][0]
df_barra['Time'] -= df_barra['Time'][0]

plt.scatter(df_winston['Time'],df_winston['Voltaje'])
plt.xlabel("Voltaje")
plt.ylabel("Tiempo")
plt.title("Gráfico para los 5000 pulsos -Winston_Resina_MPPC")
plt.show()

plt.scatter(df_winston_mod['Time'],df_winston_mod['Voltaje'])
plt.xlabel("Voltaje")
plt.ylabel("Tiempo")
```

```
plt.title("Gráfico para los 5000 pulsos -Winston_mod_Resina_MPPC")
plt.show()

plt.scatter(df_barra['Time'],df_barra['Voltaje'])
plt.xlabel("Voltaje")
plt.ylabel("Tiempo")
plt.title("Gráfico para los 5000 pulsos - Barra_Resina_MPPC")
plt.show()

## Integración de datos para los 5000 pulsos

pulsos=[]
for i in range(1,5001):
    pulsos.append(i)

### Winston

import scipy.integrate

d = 1250
lista2 = []

num_pulses = len(df_winston) // d

for i in range(num_pulses):
    datosV = df_winston['Voltaje'][i*d : (i+1)*d]
    datosT = df_winston['Time'][i*d : (i+1)*d]
    valor = scipy.integrate.trapz(datosV, datosT)
    lista2.append(valor)

### Winston mod

import scipy.integrate

d = 1250
lista2_mod = []

num_pulses = len(df_winston_mod) // d

for i in range(num_pulses):
    datosV = df_winston_mod['Voltaje'][i*d : (i+1)*d]
    datosT = df_winston_mod['Time'][i*d : (i+1)*d]
```

```
valor = scipy.integrate.trapz(datosV, datosT)
lista2_mod.append(valor)

### Barra

import scipy.integrate

d = 1250
lista2_barra = []

num_pulses = len(df_barra) // d

for i in range(num_pulses):
    datosV = df_barra['Voltaje'][i*d : (i+1)*d]
    datosT = df_barra['Time'][i*d : (i+1)*d]
    valor = scipy.integrate.trapz(datosV, datosT)
    lista2_barra.append(valor)

df1_winston=pd.DataFrame()

df1_winston_mod=pd.DataFrame()

df1_barra=pd.DataFrame()

df1_winston['Pulsos']=pulsos
df1_winston['integral V']=lista2
df1_winston_mod['Pulsos']=pulsos
df1_winston_mod['integral V']=lista2_mod
df1_barra['Pulsos']=pulsos
df1_barra['integral V']=lista2_barra

df1_winston.rename(columns={'integral V': 'Carga'}, inplace=True)
df1_winston_mod.rename(columns={'integral V': 'Carga'}, inplace=True)
df1_barra.rename(columns={'integral V': 'Carga'}, inplace=True)

carga_total_winston=df1_winston['Carga'].sum()

carga_total_winston_mod=df1_winston_mod['Carga'].sum()

carga_total_barra=df1_barra['Carga'].sum()
```

```
## distribucion winston

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit

# Crear el histograma de la distribuci n de carga total
hist, bin_edges = np.histogram(df1_winston['Carga'], bins=50,
                               density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
plt.bar(bin_centers, hist, width=np.diff(bin_edges), color='blue'
        , alpha=0.7, label='Datos Experimentales')

# Definir la funci n Gaussiana para el ajuste
def gaussian(x, amplitude, mean, stddev):
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))

# Estimar los valores iniciales para el ajuste
amplitude_guess = hist.max()
mean_guess = df1_winston['Carga'].mean()
stddev_guess = df1_winston['Carga'].std()

# Ajustar la funci n Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[amplitude_guess, mean_guess, stddev_guess])
x = np.linspace(df1_winston['Carga'].min(), df1_winston['Carga'].max(), 100)
plt.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano')

text_x = popt[1] - 2.5* popt[2]
text_y = 0.8 * hist.max()
plt.text(text_x, text_y, f'sigma: {popt[2]:.4e}', color='b', ha='center', va='center')

plt.xlabel('Carga')
plt.ylabel('Frecuencia')
plt.title('Distribuci n de carga total en los pulsos -
Winston_Cone_Resina_MPPC')
plt.legend()
plt.grid(True)
```

```
plt.show()

## distribucion carga winston mod

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit

# Crear el histograma de la distribucion de carga total
hist, bin_edges = np.histogram(df1_winston_mod['Carga'], bins=50,
                               density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
plt.bar(bin_centers, hist, width=np.diff(bin_edges), color='blue'
        , alpha=0.7, label='Datos Experimentales')

# Definir la funcion Gaussiana para el ajuste
def gaussian(x, amplitude, mean, stddev):
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))

# Estimar los valores iniciales para el ajuste
amplitude_guess = hist.max()
mean_guess = df1_winston_mod['Carga'].mean()
stddev_guess = df1_winston_mod['Carga'].std()

# Ajustar la funcion Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[amplitude_guess, mean_guess, stddev_guess])
x = np.linspace(df1_winston_mod['Carga'].min(), df1_winston_mod['Carga'].max(), 100)
plt.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano')

text_x = popt[1] - 2.5* popt[2]
text_y = 0.8 * hist.max()
plt.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='b', ha='center', va='center')

plt.xlabel('Carga')
plt.ylabel('Frecuencia')
plt.title('Distribucion de carga total en los pulsos -
Winston_Cone_mod_Resina_MPPC')
plt.legend()
```

```
plt.grid(True)
plt.show()

## Distribucion barra

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit

# Crear el histograma de la distribucion de carga total
hist, bin_edges = np.histogram(df1_barra['Carga'], bins=50,
                               density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
plt.bar(bin_centers, hist, width=np.diff(bin_edges), color='blue',
        alpha=0.7, label='Datos Experimentales')

# Definir la funcion Gaussiana para el ajuste
def gaussian(x, amplitude, mean, stddev):
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))

# Estimar los valores iniciales para el ajuste
amplitude_guess = hist.max()
mean_guess = df1_barra['Carga'].mean()
stddev_guess = df1_barra['Carga'].std()

# Ajustar la funcion Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[amplitude_guess, mean_guess, stddev_guess])
x = np.linspace(df1_barra['Carga'].min(), df1_barra['Carga'].max(), 100)
plt.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano')

text_x = popt[1] - 2.5* popt[2]
text_y = 0.8 * hist.max()
plt.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='b', ha='center', va='center')

plt.xlabel('Carga')
plt.ylabel('Frecuencia')
plt.title('Distribucion de carga total en los pulsos -\nbarra_Resina_MPPC')
```

```
plt.legend()
plt.grid(True)
plt.show()

## comparacion distribucion de carga de las 3 geometras

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit

txt_color2 = '#004C74'

# Definir la funcion Gaussiana para el ajuste
def gaussian(x, amplitude, mean, stddev):
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))

fig, ax = plt.subplots(figsize=(10, 6))

# Datos del primer grafico
hist, bin_edges = np.histogram(df1_winston['Carga'], bins=50,
                               density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
ax.bar(bin_centers, hist, width=np.diff(bin_edges), color='red',
       alpha=0.5, label='Winston Resina MPPC')

# Ajustar la funcion Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[hist.max(),
                                                       df1_winston['Carga'].mean(),
                                                       df1_winston['Carga'].std()])
x = np.linspace(df1_winston['Carga'].min(), df1_winston['Carga'].max(), 100)
ax.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano
winston')
text_x = popt[1] - 8* popt[2]
text_y = 0.68 * hist.max()
ax.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='red', ha=
'center', va='center')

# Datos del segundo grafico
hist, bin_edges = np.histogram(df1_winston_mod['Carga'], bins=50,
                               density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
```

```

ax.bar(bin_centers, hist, width=np.diff(bin_edges), color='darkgreen', alpha=0.5, label='Winston Mod Resina MPPC')

# Ajustar la funci n Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[hist.max(), df1_winston_mod['Carga'].mean(), df1_winston_mod['Carga'].std()])
x = np.linspace(df1_winston_mod['Carga'].min(), df1_winston_mod['Carga'].max(), 100)
ax.plot(x, gaussian(x, *popt), 'g-', label='Ajuste Gaussiano winston mod')
text_x = popt[1] - 5.5* popt[2]
text_y = 0.6 * hist.max()
ax.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='g', ha='center', va='center')

# Datos del tercer gr fico
hist, bin_edges = np.histogram(df1_barra['Carga'], bins=50, density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
ax.bar(bin_centers, hist, width=np.diff(bin_edges), color='royalblue', alpha=0.5, label='Barra Resina MPPC')

# Ajustar la funci n Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[hist.max(), df1_barra['Carga'].mean(), df1_barra['Carga'].std()])
x = np.linspace(df1_barra['Carga'].min(), df1_barra['Carga'].max(), 100)
ax.plot(x, gaussian(x, *popt), 'b-', label='Ajuste Gaussiano barra')
text_x = popt[1] - 3.1* popt[2]
text_y = 0.5 * hist.max()
ax.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='royalblue', ha='center', va='center')

# Configurar la leyenda y etiquetas de ejes
ax.set_xlabel('Carga', c=txt_color2)
ax.set_ylabel('Frecuencia', c=txt_color2)
ax.set_title('Distribuci n de carga total en los pulsos - Comparaci n', c=txt_color2)
ax.legend()
ax.grid(True)

```

```
plt.show()

### Comparacion de carga promedio en las 3 geometrias

df_comp = pd.DataFrame()

Winston=carga_total_winston
Winston_mod=carga_total_winston_mod
Barra=carga_total_barra

list=(Winston,Winston_mod,Barra)

lista_name=['Winston','Winston_mod','Barra']

df_comp['Geometria']=lista_name
df_comp['Carga Promedio']=list

## comparacion de carga total

df_comp.plot(x='Geometria', y='Carga Promedio', kind="bar", width=0.3)
plt.tick_params(axis="x", rotation=0, labelsize=10)
plt.rcParams["figure.figsize"] = (6, 4)
plt.xlabel('Geometrias', fontsize=13, horizontalalignment='center', position=(0.5,-0.5), c=txt_color2) # Titulo eje x
plt.ylabel('Carga', fontsize=13,c=txt_color2)
plt.title('Comparacion de carga total para distintas geometrias - Resina-MPPC', fontsize=13, c=txt_color2) # Titulo de la grafica
plt.legend(prop={'size': 10})
plt.legend(labels=['Carga total'], prop={'size': 10}) #
Renombrar la leyenda
plt.ylim(0,1.6e-5)
plt.text(1.8,0.95e-05,"{:.4e}".format(carga_total_barra),
        fontsize=10,c=txt_color2)
plt.text(-0.2,1.25e-05,"{:.4e}".format(carga_total_winston),
        fontsize=10,c=txt_color2)
plt.text( 0.8,1.1e-05,"{:.4e}".format(carga_total_winston_mod),
        fontsize=10,c=txt_color2)
```

Listing A.3: Montaje 2

#### A0.4. Código Montaje 3

```
import pandas as pd
import numpy as np
import scipy
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf
import seaborn as sns
from scipy import random
import warnings
from itertools import repeat
warnings.filterwarnings("ignore", category=DeprecationWarning)
plt.style.use('seaborn-whitegrid') # Gráficos estilo seaborn
plt.rcParams["figure.figsize"] = (12, 4) # Tamaño de los gráficos
plt.rcParams["figure.dpi"] = 100 # Resolución de los gráficos

df_winston = pd.read_csv("winston_acrilico_MPPC_000.csv", sep = ',')
# sep = ',' sirve para separar en 2 columnas el excel leido.

df_winston_mod = pd.read_csv("winston_mod_acrilico_MPPC_000.csv",
sep = ',') # sep = ',' sirve para separar en 2 columnas el excel leido.

df_barra = pd.read_csv("barra_acrilico_MPPC_000.csv", sep = ',')
# sep = ',' sirve para separar en 2 columnas el excel leido.

import pandas as pd

pulse_length = 3125

num_pulses = len(df_winston) // pulse_length

for i in range(num_pulses):

    start_index = i * pulse_length
    end_index = start_index + 740
```

```
pulse_data = df_winston['Voltaje'][start_index:end_index].
values
baseline_mean = pulse_data.mean()

df_winston['Voltaje'][start_index:start_index + pulse_length]
-= baseline_mean

print(df_winston.head(5))

import pandas as pd

pulse_length = 3125

num_pulses = len(df_winston_mod) // pulse_length

for i in range(num_pulses):
    start_index = i * pulse_length
    end_index = start_index + 740
    pulse_data = df_winston_mod['Voltaje'][start_index:end_index].
    values
    baseline_mean = pulse_data.mean()
    df_winston_mod['Voltaje'][start_index:start_index +
    pulse_length] -= baseline_mean
print(df_winston_mod.head(5))

import pandas as pd

pulse_length = 3125
num_pulses = len(df_barra) // pulse_length

for i in range(num_pulses):
    start_index = i * pulse_length
    end_index = start_index + 740
    pulse_data = df_barra['Voltaje'][start_index:end_index].
    values

    # Calcular el promedio de la linea base (primeros 740
    valores)
    baseline_mean = pulse_data.mean()
    df_barra['Voltaje'][start_index:start_index + pulse_length]
    -= baseline_mean
print(df_barra.head(5))
```

```
df_winston['Time'] -= df_winston['Time'][0]
df_winston_mod['Time'] -= df_winston_mod['Time'][0]
df_barra['Time'] -= df_barra['Time'][0]

plt.scatter(df_winston['Time'],df_winston['Voltaje'])
plt.xlabel("Tiempo")
plt.ylabel("Voltaje")

plt.title("Gráfico para los 5000 pulsos -Winston_Acrilico_MPPC")
plt.show()

plt.scatter(df_winston_mod['Time'],df_winston_mod['Voltaje'])
plt.xlabel("Voltaje")
plt.ylabel("Tiempo")
plt.title("Gráfico para los 5000 pulsos -
Winston_mod_Acrilico_MPPC")
plt.show()

plt.scatter(df_barra['Time'],df_barra['Voltaje'])
plt.xlabel("Voltaje")
plt.ylabel("Tiempo")
plt.title("Gráfico para los 5000 pulsos -Barra_Acrilico_MPPC")
plt.show()

## Integracion de datos para los 5000 pulsos

pulsos=[]
for i in range(1,5001):
    pulsos.append(i)

#Winston

import scipy.integrate

d = 3125
lista2 = []

num_pulses = len(df_winston) // d

for i in range(num_pulses):
```

```
datosV = df_winston['Voltaje'][i*d : (i+1)*d]
datosT = df_winston['Time'][i*d : (i+1)*d]
valor = scipy.integrate.trapz(datosV, datosT)
lista2.append(valor)

## winston mod

import scipy.integrate

d = 3125
lista2_mod = []

num_pulses = len(df_winston_mod) // d

for i in range(num_pulses):
    datosV = df_winston_mod['Voltaje'][i*d : (i+1)*d]
    datosT = df_winston_mod['Time'][i*d : (i+1)*d]
    valor = scipy.integrate.trapz(datosV, datosT)
    lista2_mod.append(valor)

## Barra

import scipy.integrate

d = 3125
lista2_barra = []

num_pulses = len(df_barra) // d

for i in range(num_pulses):
    datosV = df_barra['Voltaje'][i*d : (i+1)*d]
    datosT = df_barra['Time'][i*d : (i+1)*d]
    valor = scipy.integrate.trapz(datosV, datosT)
    lista2_barra.append(valor)

df1_winston=pd.DataFrame()

df1_winston_mod=pd.DataFrame()

df1_barra=pd.DataFrame()

df1_winston['Pulsos']=pulsos
```

```

df1_winston['integral V']=lista2
df1_winston_mod['Pulsos']=pulsos
df1_winston_mod['integral V']=lista2_mod
df1_barra['Pulsos']=pulsos
df1_barra['integral V']=lista2_barra

df1_winston.rename(columns={'integral V': 'Carga'}, inplace=True)
df1_winston_mod.rename(columns={'integral V': 'Carga'}, inplace=
    True)
df1_barra.rename(columns={'integral V': 'Carga'}, inplace=True)

carga_total_winston=df1_winston['Carga'].sum()

carga_total_winston_mod=df1_winston_mod['Carga'].sum()

carga_total_barra=df1_barra['Carga'].sum()

## distribucion de carga winston

plt.figure(figsize=(8, 4))
# Crear el histograma de la distribucion de carga total
hist, bin_edges = np.histogram(df1_winston['Carga'], bins=50,
    density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
plt.bar(bin_centers, hist, width=np.diff(bin_edges), color='blue'
    , alpha=0.7, label='Datos Experimentales')

# Definir la funcion Gaussiana para el ajuste
def gaussian(x, amplitude, mean, stddev):
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))

# Estimar los valores iniciales para el ajuste
amplitude_guess = hist.max()
mean_guess = df1_winston['Carga'].mean()
stddev_guess = df1_winston['Carga'].std()

# Ajustar la funcion Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[

    amplitude_guess, mean_guess, stddev_guess])
x = np.linspace(df1_winston['Carga'].min(), df1_winston['Carga'].
    max(), 100)

```

```
plt.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano')

text_x = popt[1] - 2.2* popt[2]
text_y = 0.8 * hist.max()
plt.text(text_x, text_y, f'Sigma (Ajuste): {popt[2]:.4e}', color=
'b', ha='center', va='center')

plt.text(text_x, text_y * 0.9, f'Media (Ajuste): {popt[1]:.4e}', color='g', ha='center', va='center')

plt.xlabel('Carga')
plt.ylabel('Frecuencia')
plt.title('Distribuci n de carga total en los pulsos -
Winston_Cone_Acrilico_MPPC')
plt.legend()
plt.grid(True)
plt.show()

## distribucion de carga winston modificado

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
plt.figure(figsize=(8, 4))

# Crear el histograma de la distribuci n de carga total
hist, bin_edges = np.histogram(df1_winston_mod['Carga'], bins=50,
density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
plt.bar(bin_centers, hist, width=np.diff(bin_edges), color='blue',
alpha=0.7, label='Datos Experimentales')

# Definir la funci n Gaussiana para el ajuste
def gaussian(x, amplitude, mean, stddev):
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))

# Estimar los valores iniciales para el ajuste
amplitude_guess = hist.max()
mean_guess = df1_winston_mod['Carga'].mean()
```

```
stddev_guess = df1_winston_mod['Carga'].std()

# Ajustar la funci n Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[amplitude_guess, mean_guess, stddev_guess])
x = np.linspace(df1_winston_mod['Carga'].min(), df1_winston_mod['Carga'].max(), 100)
plt.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano')

text_x = popt[1] - 2.5* popt[2]
text_y = 0.8 * hist.max()
plt.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='b', ha='center', va='center')

# Agregar el valor de la media del ajuste al gr fico como texto
# en notaci n cient fica
plt.text(text_x, text_y * 0.9, f'Media (Ajuste): {popt[1]:.4e}', color='g', ha='center', va='center')

plt.xlabel('Carga')
plt.ylabel('Frecuencia')
plt.title('Distribuci n de carga total en los pulsos - Winston_Cone_mod_Acrilico_MPPC')
plt.legend()
plt.grid(True)
plt.show()

## distribucion de carga barra

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
plt.figure(figsize=(8, 4))

# Crear el histograma de la distribuci n de carga total
hist, bin_edges = np.histogram(df1_barra['Carga'], bins=50, density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
plt.bar(bin_centers, hist, width=np.diff(bin_edges), color='blue', alpha=0.7, label='Datos Experimentales')
```

```
# Definir la función Gaussiana para el ajuste
def gaussian(x, amplitude, mean, stddev):
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))

# Estimar los valores iniciales para el ajuste
amplitude_guess = hist.max()
mean_guess = df1_barra['Carga'].mean()
stddev_guess = df1_barra['Carga'].std()

# Ajustar la función Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[
    amplitude_guess, mean_guess, stddev_guess])
x = np.linspace(df1_barra['Carga'].min(), df1_barra['Carga'].max(),
                 100)
plt.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano')

text_x = popt[1] - 2.5* popt[2]
text_y = 0.8 * hist.max()
plt.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='b', ha='center',
         va='center')

# Agregar el valor de la media del ajuste al gráfico como texto
# en notación científica
plt.text(text_x, text_y * 0.9, f'Media (Ajuste): {popt[1]:.4e}',
         color='g', ha='center', va='center')

plt.xlabel('Carga')
plt.ylabel('Frecuencia')
plt.title('Distribución de carga total en los pulsos -'
          'barra_Acrílico_MPPC')
plt.legend()
plt.grid(True)
plt.show()

## comparacion distribucion de carga

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit

txt_color2 = '#004C74'
```

```

# Definir la función Gaussiana para el ajuste
def gaussian(x, amplitude, mean, stddev):
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))

fig, ax = plt.subplots(figsize=(12, 6))

# Datos del primer gráfico
hist, bin_edges = np.histogram(df1_winston['Carga'], bins=50,
                               density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
ax.bar(bin_centers, hist, width=np.diff(bin_edges), color='red',
       alpha=0.5, label='Winston Cone Resina PMT')

# Ajustar la función Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[hist.max(),
                                                       df1_winston['Carga'].mean(), df1_winston['Carga'].std()])
x = np.linspace(df1_winston['Carga'].min(), df1_winston['Carga'].max(), 100)
ax.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano
winston')
text_x = popt[1] - 5 * popt[2]
text_y = 0.75 * hist.max()
ax.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='red', ha=
'center', va='center')
ax.text(text_x, text_y * 0.9, f'Media: {popt[1]:.4e}', color='red
', ha='center', va='center')

# Datos del segundo gráfico
hist, bin_edges = np.histogram(df1_winston_mod['Carga'], bins=50,
                               density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
ax.bar(bin_centers, hist, width=np.diff(bin_edges), color='
darkgreen', alpha=0.5, label='Winston Cone Mod Resina PMT')

# Ajustar la función Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[hist.max(),
                                                       df1_winston_mod['Carga'].mean(), df1_winston_mod['Carga'].std
()])
x = np.linspace(df1_winston_mod['Carga'].min(), df1_winston_mod['
Carga'].max(), 100)

```

```
ax.plot(x, gaussian(x, *popt), 'g-', label='Ajuste Gaussiano
winston mod')
text_x = popt[1] - 4* popt[2]
text_y = 0.9* hist.max()
ax.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='darkgreen
', ha='center', va='center')
ax.text(text_x, text_y * 0.9, f'Media: {popt[1]:.4e}', color='
darkgreen', ha='center', va='center')

# Datos del tercer gráfico
hist, bin_edges = np.histogram(df1_barra['Carga'], bins=50,
density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
ax.bar(bin_centers, hist, width=np.diff(bin_edges), color='
royalblue', alpha=0.5, label='Barra Resina PMT')

# Ajustar la función Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[hist.max(),
df1_barra['Carga'].mean(), df1_barra['Carga'].std()])
x = np.linspace(df1_barra['Carga'].min(), df1_barra['Carga'].max
(), 100)
ax.plot(x, gaussian(x, *popt), 'b-', label='Ajuste Gaussiana
barra')
text_x = popt[1] - 2.8 * popt[2]
text_y = 0.7* hist.max()
ax.text(text_x, text_y, f'Sigma: {popt[2]:.4e}', color='royalblue
', ha='center', va='center')
ax.text(text_x, text_y * 0.9, f'Media: {popt[1]:.4e}', color='
royalblue', ha='center', va='center')

# Configurar la leyenda y etiquetas de ejes
ax.set_xlabel('Carga', c=txt_color2)
ax.set_ylabel('Frecuencia', c=txt_color2)
ax.set_title('Distribución de carga total en los pulsos -
Comparación', c=txt_color2)
ax.legend()
ax.grid(True)
plt.show()

## distribucion de numero de fotones

import pandas as pd
```

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit

txt_color2 = '#004C74'

# Valor de la carga de un fotoelectrón en Coulombs
charge_per_photoelectron = 3.10747e-13

# Definir la función Gaussiana para el ajuste
def gaussian(x, amplitude, mean, stddev):
    return amplitude * np.exp(-(x - mean)**2 / (2 * stddev**2))

fig, ax = plt.subplots(figsize=(10, 6))

# Datos del primer gráfico
hist, bin_edges = np.histogram(df1_winston['Carga'], bins=50,
                               density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2

# Convertir carga a número de fotones
num_photons = df1_winston['Carga'] / charge_per_photoelectron

# Ajustar la función Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[hist.max(),
    num_photons.mean(), num_photons.std()])
x = np.linspace(num_photons.min(), num_photons.max(), 100)
ax.bar(bin_centers, hist, width=np.diff(bin_edges), color='red',
    alpha=0.5, label='Winston Cone Resina PMT')
ax.plot(x, gaussian(x, *popt), 'r-', label='Ajuste Gaussiano
winston')
text_x = popt[1] - 7 * popt[2]
text_y = 0.75 * hist.max()
ax.text(text_x, text_y, f'Geometría: Winston Cone\nMedia: {popt
[1]:.2e}\nTotal de Fotones: {num_photons.sum():.2e}', color='
red', ha='center', va='center')

# Datos del segundo gráfico
hist, bin_edges = np.histogram(df1_winston_mod['Carga'], bins=50,
                               density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
```

```
# Convertir carga a n mero de fotones
num_photons_mod = df1_winston_mod['Carga'] /
    charge_per_photoelectron

# Ajustar la funci n Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[hist.max(),
    num_photons_mod.mean(), num_photons_mod.std()])
x = np.linspace(num_photons_mod.min(), num_photons_mod.max(),
    100)
ax.bar(bin_centers, hist, width=np.diff(bin_edges), color='darkgreen',
    alpha=0.5, label='Winston Cone Mod Resina PMT')
ax.plot(x, gaussian(x, *popt), 'g-', label='Ajuste Gaussiano
    winston mod')
text_x = popt[1] - 5.5* popt[2]
text_y = 0.9* hist.max()
ax.text(text_x, text_y, f'Geometr a: Winston Cone Mod\nMedia: {popt[1]:.2e}\nTotal de Fotones: {num_photons_mod.sum():.2e}', color='darkgreen', ha='center', va='center')

# Datos del tercer gr fico
hist, bin_edges = np.histogram(df1_barra['Carga'], bins=50,
    density=True)
bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2

# Convertir carga a n mero de fotones
num_photons_bar = df1_barra['Carga'] / charge_per_photoelectron

# Ajustar la funci n Gaussiana a los datos
popt, _ = curve_fit(gaussian, bin_centers, hist, p0=[hist.max(),
    num_photons_bar.mean(), num_photons_bar.std()])
x = np.linspace(num_photons_bar.min(), num_photons_bar.max(),
    100)
ax.bar(bin_centers, hist, width=np.diff(bin_edges), color='royalblue',
    alpha=0.5, label='Barra Resina PMT')
ax.plot(x, gaussian(x, *popt), 'b-', label='Ajuste Gaussiano
    barra')
text_x = popt[1] - 6* popt[2]
text_y = 0.5 * hist.max()
ax.text(text_x, text_y, f'Geometr a: Barra\nMedia: {popt[1]:.2e}
    }\nTotal de Fotones: {num_photons_bar.sum():.2e}', color='royalblue', ha='center', va='center')
```

```

# Configurar las leyendas
ax.set_title('Distribuci n del n mero de fotones en los pulsos
- Comparaci n', c=txt_color2)
ax.set_xlabel('N mero de Fotones', c=txt_color2)
ax.set_ylabel('Frecuencia', c=txt_color2)
ax.grid(True)
ax.legend()
ax.set_xlim(35000, 50000)
plt.tight_layout()
plt.show()

relative_percentage_winston_mod = (num_photons_mod.sum() /
    num_photons.sum()) * 100
relative_percentage_barra = (num_photons_bar.sum() / num_photons.
    sum()) * 100

print(f"Porcentaje relativo para Winston Cone Mod Resina PMT: {
    relative_percentage_winston_mod:.2f}%)")
print(f"Porcentaje relativo para Barra Resina PMT: {
    relative_percentage_barra:.2f}%)"

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

carga_total_winston = df1_winston['Carga'].sum()
carga_total_winston_mod = df1_winston_mod['Carga'].sum()
carga_total_barra = df1_barra['Carga'].sum()

charge_per_photoelectron = 3.10747e-13
fig, ax = plt.subplots(figsize=(6, 4))
geometrias = ['Winston Cone', 'Winston Cone Mod', 'Barra']
cargas_totales = [carga_total_winston/charge_per_photoelectron,
    carga_total_winston_mod/charge_per_photoelectron,
    carga_total_barra/charge_per_photoelectron]

ax.bar(geometrias, cargas_totales, color=['b', 'b', 'blue'],
    alpha=0.7, width=0.3)
ax.set_ylabel('Fotones', fontsize=11,c=txt_color2)
ax.set_ylim(0, 3e8)
ax.set_title('N mero de fotones - Comparaci n', fontsize=13,c=

```

```
txt_color2)

plt.xlabel('Geometrias', fontsize=11, horizontalalignment='center'
', position=(0.5,-0.5), c=txt_color2)
plt.legend(labels=['Fotones totales'], prop={'size': 11})

plt.text(1.82,2.1e08,"{:.4e}".format(carga_total_barra/
charge_per_photoelectron), fontsize=12,c=txt_color2)
plt.text(-0.15,2.35e08,"{:.4e}".format(carga_total_winston/
charge_per_photoelectron), fontsize=12,c=txt_color2)
plt.text( 0.83,2.5e08,"{:.4e}".format(carga_total_winston_mod/
charge_per_photoelectron), fontsize=12,c=txt_color2)
plt.tight_layout()
plt.show()

import matplotlib.pyplot as plt

# Porcentajes relativos
relative_percentage_winston_mod = 106
relative_percentage_barra = 87

geometries = ['Winston Cone Mod', 'Barra']

# Valores de los porcentajes
percentages = [relative_percentage_winston_mod,
relative_percentage_barra]
fig, ax = plt.subplots()
bar_positions = [1, 1.5]
bars = ax.bar(bar_positions, percentages, color=['blue', 'green'],
width=0.3, alpha=0.5)
ax.set_xticks(bar_positions)
ax.set_xticklabels(geometries)

winston_porcentaje=100

# Agregar etiquetas
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 5, f'{yval}%',
', ha='center', va='bottom', fontsize=10)

# Configurar etiquetas
ax.set_ylabel('Porcentaje Relativo', fontsize=12,c=txt_color2)
```

```
ax.set_title('Porcentaje Relativo de fotones con respecto al Cono  
de Winston', fontsize=12,c=txt_color2)  
ax.set_ylim(0, max(percentages) + 20)  
plt.xlabel('Geometrias', fontsize=11, horizontalalignment='center'  
' , position=(0.5,-0.5), c=txt_color2)  
legend = plt.legend(labels=['Winston'], prop={'size': 10})  
legend.legendHandles[0].set_color('red')  
  
plt.axhline(y=winston_porcentaje, linestyle='--', color='r',  
alpha=0.7)  
plt.show()  
  
## distribucion de carga de peaks detectados en la ROI - WINSTON  
  
import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd  
from scipy.optimize import curve_fit  
from scipy.signal import find_peaks  
from sklearn.metrics import mean_squared_error, r2_score  
  
# Funcion lorentziana  
def lorentzian(x, a, x0, gamma):  
    return a * (gamma**2) / ((x - x0)**2 + gamma**2)  
  
# Funcion gaussiana  
def gaussian(x, a, x0, sigma):  
    return a * np.exp(-(x - x0)**2 / (2 * sigma**2))  
  
pulse_length = 3125  
num_pulses = len(df_winston) // pulse_length  
charges_lo = []  
  
for i in range(num_pulses):  
    start_index = i * pulse_length  
    end_index = start_index + 700  
    pulse_data = df_winston['Voltaje'][start_index:end_index].  
    values  
    baseline = np.mean(pulse_data)  
    sigma = np.std(pulse_data)  
    threshold = 6 * sigma
```

```
peak_indices, _ = find_peaks(pulse_data, height=(sigma,
threshold))

for peak in peak_indices:
    peak_start = max(0, peak - 10)
    peak_end = min(len(pulse_data) - 1, peak + 10)
    peak_time = df_winston['Time'][start_index + peak_start:
start_index + peak_end + 1].values
    peak_voltage = pulse_data[peak_start:peak_end + 1]
    charge = np.trapz(peak_voltage - baseline, x=peak_time)
    charges_lo.append(charge)

hist, bin_edges = np.histogram(charges_lo, bins=100)

popt_lorentzian, _ = curve_fit(lorentzian, bin_edges[:-1], hist,
p0=[max(hist), np.mean(charges_lo), np.std(charges_lo)])

popt_gaussian, _ = curve_fit(gaussian, bin_edges[:-1], hist, p0=[
max(hist), np.mean(charges_lo), np.std(charges_lo)])

charge_per_photoelectron_lorentzian = popt_lorentzian[1]

charge_per_photoelectron_gaussian = popt_gaussian[1]

# Normalizar
hist_normalized = hist / max(hist)

# Ajustar una función lorentziana
popt_lorentzian, _ = curve_fit(lorentzian, bin_edges[:-1],
hist_normalized, p0=[max(hist_normalized), np.mean(charges_lo),
np.std(charges_lo)])

# Ajustar una función gaussiana
popt_gaussian, _ = curve_fit(gaussian, bin_edges[:-1],
hist_normalized, p0=[max(hist_normalized), np.mean(charges_lo),
np.std(charges_lo)])

# Calcular las curvas de ajuste utilizando los datos no
normalizados
y_fit_lorentzian = lorentzian(x_fit, *popt_lorentzian) * max(hist
)
y_fit_gaussian = gaussian(x_fit, *popt_gaussian) * max(hist)
```

```
# Calcular el mse
y_fit_lorentzian_normalized = lorentzian(bin_edges[:-1], *
    popt_lorentzian)
mse_lorentzian = mean_squared_error(hist_normalized,
    y_fit_lorentzian_normalized)

# Calcular el mse
y_fit_gaussian_normalized = gaussian(bin_edges[:-1], *
    popt_gaussian)
mse_gaussian = mean_squared_error(hist_normalized,
    y_fit_gaussian_normalized)

# Calcular el coeficiente
r2_lorentzian = r2_score(hist_normalized,
    y_fit_lorentzian_normalized)

# Calcular el coeficiente
r2_gaussian = r2_score(hist_normalized, y_fit_gaussian_normalized
    )

# Ganancia del MPPC
mppc_gain = 2.5e6

# Calcular la carga de un fotoelectr n dividida por la carga
# elemental del electr n
charge_divided_by_elementary = charge_per_photoclectron_gaussian
    / (1.602176634e-19)
ganancia_winston=charge_divided_by_elementary*1e-6

# Calcular la FWHM
half_max = max(hist) / 2.0
indexes = np.where(hist >= half_max)[0]
fwhm_start = bin_edges[indexes[0]]
fwhm_end = bin_edges[indexes[-1]]
fwhm = fwhm_end - fwhm_start

# Visualizaci n
plt.figure(figsize=(12, 7))
plt.bar(bin_edges[:-1], hist, width=np.diff(bin_edges), align='
    edge', label='Histograma de carga')
plt.plot(x_fit, y_fit_lorentzian, 'r-', label='Ajuste Lorentziano')
```

```
')

plt.plot(x_fit, y_fit_gaussian, 'g-', label='Ajuste Gaussiano')

plt.axvspan(fwhm_start, fwhm_end, color='gray', alpha=0.3)
plt.xlim(-3e-12, 3e-12)
plt.xlabel('Carga (pC)')
plt.ylabel('Frecuencia')
plt.title('Histograma de carga con ajustes Lorentziano y
           Gaussiano - Geometria Winston')
plt.text(0.805, 0.83, 'Zona gris: FWHM', transform=plt.gca() .
         transAxes, fontsize=12, color='black')
plt.legend()

text = f"""Carga (Lorentziano): {
    charge_per_photon_lorentzian:.5e} C ({{
    charge_per_photon_lorentzian * 1e12:.5f} pC)}
Carga (Gaussiano): {charge_per_photon_gaussian:.5e} C ({{
    charge_per_photon_gaussian * 1e12:.5f} pC)}
MSE (Lorentziano): {mse_lorentzian:.5f}
R^2 (Lorentziano): {r2_lorentzian:.4f}
MSE (Gaussiano): {mse_gaussian:.5f}
R^2 (Gaussiano): {r2_gaussian:.4f}
FWHM: {fwhm:.5e} C ({fwhm * 1e12:.5f} pC)
Ganancia nominal del MPPC: {mppc_gain:.1e}
Ganancia experimental del MPPC: {charge_divided_by_elementary*1e
-6:.5f}e+06"""

plt.text(0.01, 0.6, text, transform=plt.gca().transAxes, fontsize
        =12)
plt.show()

## distribucion de carga de peaks detectados en la ROI - WINSTON
MOD

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit
from scipy.signal import find_peaks
from sklearn.metrics import mean_squared_error, r2_score

# Funcion lorentziana
def lorentzian(x, a, x0, gamma):
```

```

    return a * (gamma**2) / ((x - x0)**2 + gamma**2)

# Función gaussiana
def gaussian(x, a, x0, sigma):
    return a * np.exp(-(x - x0)**2 / (2 * sigma**2))

pulse_length = 3125
num_pulses = len(df_winston_mod) // pulse_length
charges_lo = []

for i in range(num_pulses):
    start_index = i * pulse_length
    end_index = start_index + 700
    pulse_data = df_winston_mod['Voltaje'][start_index:end_index].values
    baseline = np.mean(pulse_data)
    sigma = np.std(pulse_data)
    threshold = 6 * sigma
    peak_indices, _ = find_peaks(pulse_data, height=(sigma,
    threshold))

    for peak in peak_indices:
        peak_start = max(0, peak - 10)
        peak_end = min(len(pulse_data) - 1, peak + 10)
        peak_time = df_winston_mod['Time'][start_index +
peak_start:start_index + peak_end + 1].values
        peak_voltage = pulse_data[peak_start:peak_end + 1]
        charge = np.trapz(peak_voltage - baseline, x=peak_time)
        charges_lo.append(charge)

hist, bin_edges = np.histogram(charges_lo, bins=100)

popt_lorentzian, _ = curve_fit(lorentzian, bin_edges[:-1], hist,
p0=[max(hist), np.mean(charges_lo), np.std(charges_lo)])

popt_gaussian, _ = curve_fit(gaussian, bin_edges[:-1], hist, p0=[

max(hist), np.mean(charges_lo), np.std(charges_lo)])

charge_per_photoelectron_lorentzian = popt_lorentzian[1]

charge_per_photoelectron_gaussian = popt_gaussian[1]

# Normalizar

```

```
hist_normalized = hist / max(hist)

popt_lorentzian, _ = curve_fit(lorentzian, bin_edges[:-1],
    hist_normalized, p0=[max(hist_normalized), np.mean(charges_lo)
    , np.std(charges_lo)])

popt_gaussian, _ = curve_fit(gaussian, bin_edges[:-1],
    hist_normalized, p0=[max(hist_normalized), np.mean(charges_lo)
    , np.std(charges_lo)])

y_fit_lorentzian = lorentzian(x_fit, *popt_lorentzian) * max(hist
    )
y_fit_gaussian = gaussian(x_fit, *popt_gaussian) * max(hist)

y_fit_lorentzian_normalized = lorentzian(bin_edges[:-1], *
    popt_lorentzian)
mse_lorentzian = mean_squared_error(hist_normalized,
    y_fit_lorentzian_normalized)

y_fit_gaussian_normalized = gaussian(bin_edges[:-1], *
    popt_gaussian)
mse_gaussian = mean_squared_error(hist_normalized,
    y_fit_gaussian_normalized)

r2_lorentzian = r2_score(hist_normalized,
    y_fit_lorentzian_normalized)

r2_gaussian = r2_score(hist_normalized, y_fit_gaussian_normalized
    )

# Ganancia del MPPC
mppc_gain = 2.5e6

charge_divided_by_elementary = charge_per_photoclectron_gaussian
    / (1.602176634e-19)
ganancia_winston_mod=charge_divided_by_elementary*1e-6

# Calcular la FWHM
half_max = max(hist) / 2.0
indexes = np.where(hist >= half_max)[0]
fwhm_start = bin_edges[indexes[0]]
```

```

fwhm_end = bin_edges[indexes[-1]]
fwhm = fwhm_end - fwhm_start

plt.figure(figsize=(12, 7))
plt.bar(bin_edges[:-1], hist, width=np.diff(bin_edges), align='
    edge', label='Histograma de carga')
plt.plot(x_fit, y_fit_lorentzian, 'r-', label='Ajuste Lorentziano
    ')
plt.plot(x_fit, y_fit_gaussian, 'g-', label='Ajuste Gaussiano')

plt.axvspan(fwhm_start, fwhm_end, color='gray', alpha=0.3)
plt.xlim(-3e-12, 3e-12)
plt.xlabel('Carga (pC)')
plt.ylabel('Frecuencia')
plt.title('Histograma de carga con ajustes Lorentziano y
    Gaussiano - Geometria Winston modificado')
plt.text(0.805, 0.83, 'Zona gris: FWHM', transform=plt.gca().
    transAxes, fontsize=12, color='black')
plt.legend()

text = """Carga (Lorentziano): {
    charge_per_photoelectron_lorentzian:.5e} C ({
        charge_per_photoelectron_lorentzian * 1e12:.5f} pC)
Carga (Gaussiano): {charge_per_photoelectron_gaussian:.5e} C ({
        charge_per_photoelectron_gaussian * 1e12:.5f} pC)
MSE (Lorentziano): {mse_lorentzian:.5f}
R^2 (Lorentziano): {r2_lorentzian:.4f}
MSE (Gaussiano): {mse_gaussian:.5f}
R^2 (Gaussiano): {r2_gaussian:.4f}
FWHM: {fwhm:.5e} C ({fwhm * 1e12:.5f} pC)
Ganancia nominal del MPPC: {mppc_gain:.1e}
Ganancia experimental del MPPC: {charge_divided_by_elementary*1e
    -6:.5f}e+06"""
plt.text(0.01, 0.6, text, transform=plt.gca().transAxes, fontsize
    =12)
plt.show()

## distribucion de carga de peaks detectados en la ROI - BARRA

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

```

```
from scipy.optimize import curve_fit
from scipy.signal import find_peaks
from sklearn.metrics import mean_squared_error, r2_score

# Función lorentziana
def lorentzian(x, a, x0, gamma):
    return a * (gamma**2) / ((x - x0)**2 + gamma**2)

# Función gaussiana
def gaussian(x, a, x0, sigma):
    return a * np.exp(-(x - x0)**2 / (2 * sigma**2))

pulse_length = 3125
num_pulses = len(df_barra) // pulse_length
charges_lo = []

for i in range(num_pulses):
    start_index = i * pulse_length
    end_index = start_index + 700
    pulse_data = df_barra['Voltaje'][start_index:end_index].values
    baseline = np.mean(pulse_data)
    sigma = np.std(pulse_data)
    threshold = 6 * sigma
    peak_indices, _ = find_peaks(pulse_data, height=(sigma,
threshold))

    for peak in peak_indices:

        peak_start = max(0, peak - 10)
        peak_end = min(len(pulse_data) - 1, peak + 10)
        peak_time = df_barra['Time'][start_index + peak_start:
start_index + peak_end + 1].values
        peak_voltage = pulse_data[peak_start:peak_end + 1]
        charge = np.trapz(peak_voltage - baseline, x=peak_time)
        charges_lo.append(charge)

hist, bin_edges = np.histogram(charges_lo, bins=100)

popt_lorentzian, _ = curve_fit(lorentzian, bin_edges[:-1], hist,
p0=[max(hist), np.mean(charges_lo), np.std(charges_lo)])
```

```
popt_gaussian, _ = curve_fit(gaussian, bin_edges[:-1], hist, p0=[  
    max(hist), np.mean(charges_lo), np.std(charges_lo)])  
  
charge_per_photoelectron_lorentzian = popt_lorentzian[1]  
  
charge_per_photoelectron_gaussian = popt_gaussian[1]  
  
hist_normalized = hist / max(hist)  
  
popt_lorentzian, _ = curve_fit(lorentzian, bin_edges[:-1],  
    hist_normalized, p0=[max(hist_normalized), np.mean(charges_lo),  
    np.std(charges_lo)])  
  
popt_gaussian, _ = curve_fit(gaussian, bin_edges[:-1],  
    hist_normalized, p0=[max(hist_normalized), np.mean(charges_lo),  
    np.std(charges_lo)])  
  
y_fit_lorentzian = lorentzian(x_fit, *popt_lorentzian) * max(hist)  
)  
y_fit_gaussian = gaussian(x_fit, *popt_gaussian) * max(hist)  
  
y_fit_lorentzian_normalized = lorentzian(bin_edges[:-1], *  
    popt_lorentzian)  
mse_lorentzian = mean_squared_error(hist_normalized,  
    y_fit_lorentzian_normalized)  
  
y_fit_gaussian_normalized = gaussian(bin_edges[:-1], *  
    popt_gaussian)  
mse_gaussian = mean_squared_error(hist_normalized,  
    y_fit_gaussian_normalized)  
  
r2_lorentzian = r2_score(hist_normalized,  
    y_fit_lorentzian_normalized)  
  
r2_gaussian = r2_score(hist_normalized, y_fit_gaussian_normalized  
)  
  
# Ganancia del MPPC  
mppc_gain = 2.5e6  
  
charge_divided_by_elementary = charge_per_photoelectron_gaussian  
/ (1.602176634e-19)
```

---

```

ganancia_barra=charge_divided_by_elementary*1e-6

# Calcular la FWHM
half_max = max(hist) / 2.0
indexes = np.where(hist >= half_max)[0]
fwhm_start = bin_edges[indexes[0]]
fwhm_end = bin_edges[indexes[-1]]
fwhm = fwhm_end - fwhm_start

plt.figure(figsize=(12, 7))
plt.bar(bin_edges[:-1], hist, width=np.diff(bin_edges), align='edge', label='Histograma de carga')
plt.plot(x_fit, y_fit_lorentzian, 'r-', label='Ajuste Lorentziano')
plt.plot(x_fit, y_fit_gaussian, 'g-', label='Ajuste Gaussiano')

plt.axvspan(fwhm_start, fwhm_end, color='gray', alpha=0.3)
plt.xlim(-3e-12, 3e-12)
plt.xlabel('Carga (pC)')
plt.ylabel('Frecuencia')
plt.title('Histograma de carga con ajustes Lorentziano y
Gaussiano - Geometria Barra simple')
plt.text(0.805, 0.83, 'Zona gris: FWHM', transform=plt.gca().transAxes, fontsize=12, color='black')
plt.legend()

text = f"""
Carga (Lorentziano): {
    charge_per_photoclectron_lorentzian:.5e} C ({
    charge_per_photoclectron_lorentzian * 1e12:.5f} pC)
Carga (Gaussiano): {charge_per_photoclectron_gaussian:.5e} C ({
    charge_per_photoclectron_gaussian * 1e12:.5f} pC)
MSE (Lorentziano): {mse_lorentzian:.5f}
R^2 (Lorentziano): {r2_lorentzian:.4f}
MSE (Gaussiano): {mse_gaussian:.5f}
R^2 (Gaussiano): {r2_gaussian:.4f}
FWHM: {fwhm:.5e} C ({fwhm * 1e12:.5f} pC)
Ganancia nominal del MPPC: {mppc_gain:.1e}
Ganancia experimental del MPPC: {charge_divided_by_elementary*1e
-6:.5f}e+06"""

plt.text(0.01, 0.6, text, transform=plt.gca().transAxes, fontsize=12)

```

```
plt.show()

ganancias_experimentales = [ganancia_winston,
    ganancia_winston_mod, ganancia_barra]
geometrias = ['Winston', 'Winston Mod', 'Barra']
# Ganancia nominal
ganancia_nominal = 2.5

plt.figure(figsize=(8, 6))
bars = plt.bar(geometrias, ganancias_experimentales, width=0.4,
    color='blue', alpha=0.7)
plt.xlabel('Geometria')
plt.ylabel('Ganancia Experimental (x 10^6)')
plt.title('Comparacion de Ganancias Experimentales')
plt.ylim(0, 3)
plt.ticklabel_format(axis='y', style='sci', scilimits=(0,0))
plt.gca().yaxis.get_offset_text().set_visible(False)
plt.axhline(y=ganancia_nominal, linestyle='--', color='red',
    alpha=0.7, label='Ganancia Nominal')
plt.legend()

for bar, ganancia in zip(bars, ganancias_experimentales):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(),
        f'{ganancia:.3}', ha='center', va='bottom', fontsize=10)

plt.show()
```

**Listing A.4:** Montaje 3

**Figura A0.1:** Universidad Andrés Bello

