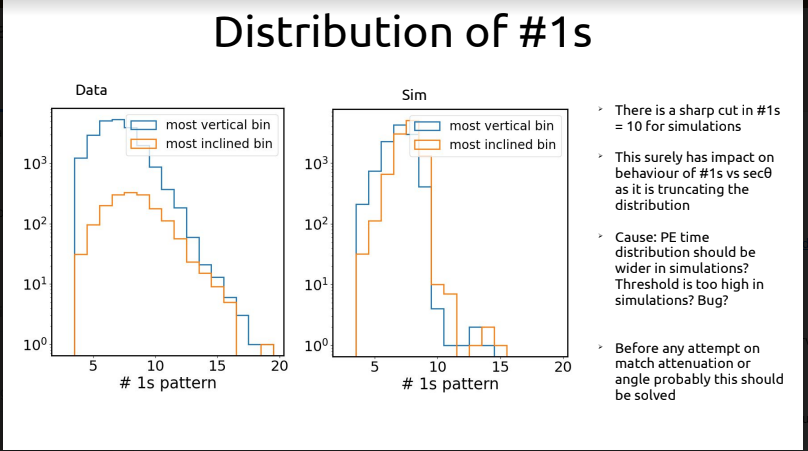
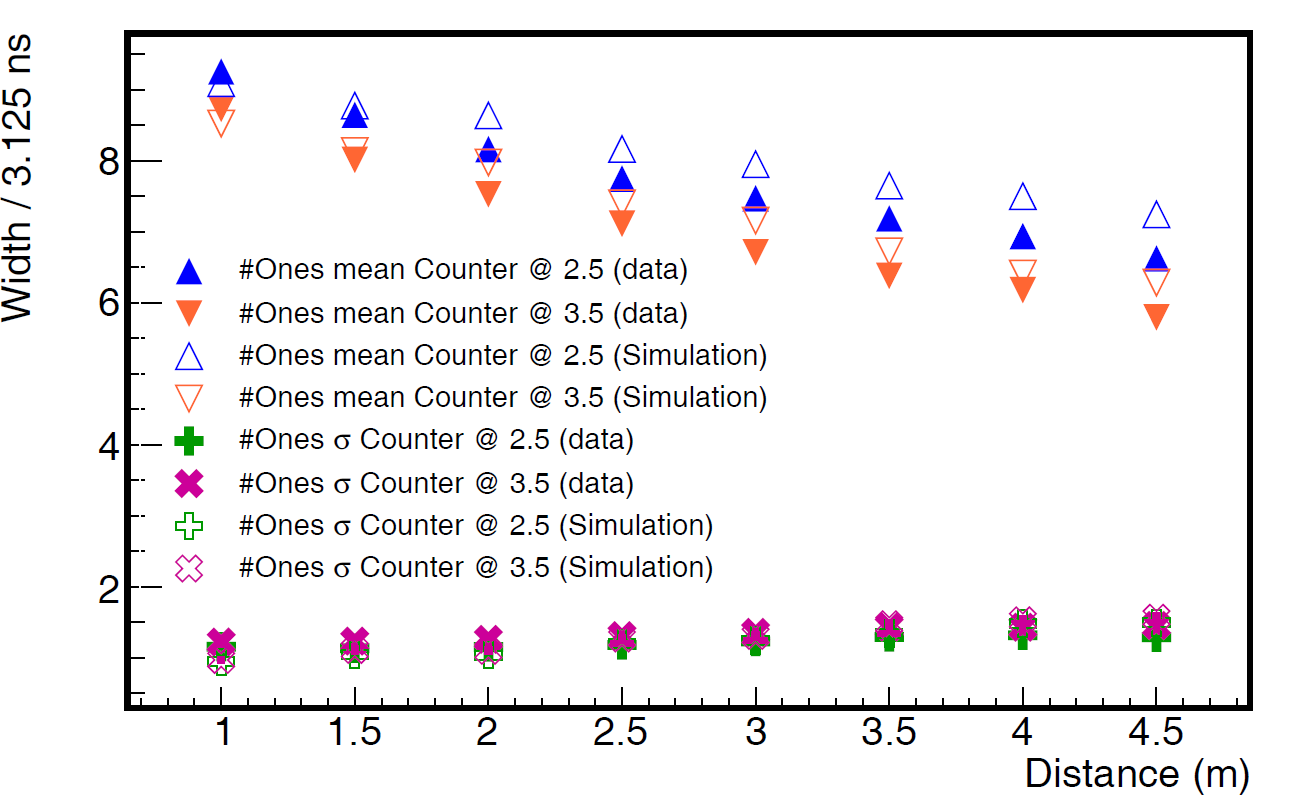
**Proyecto Amiga dentro del Observatorio Pierre Auger: problemas con las simulaciones con foco en la electrónica de las barras centelladoras**

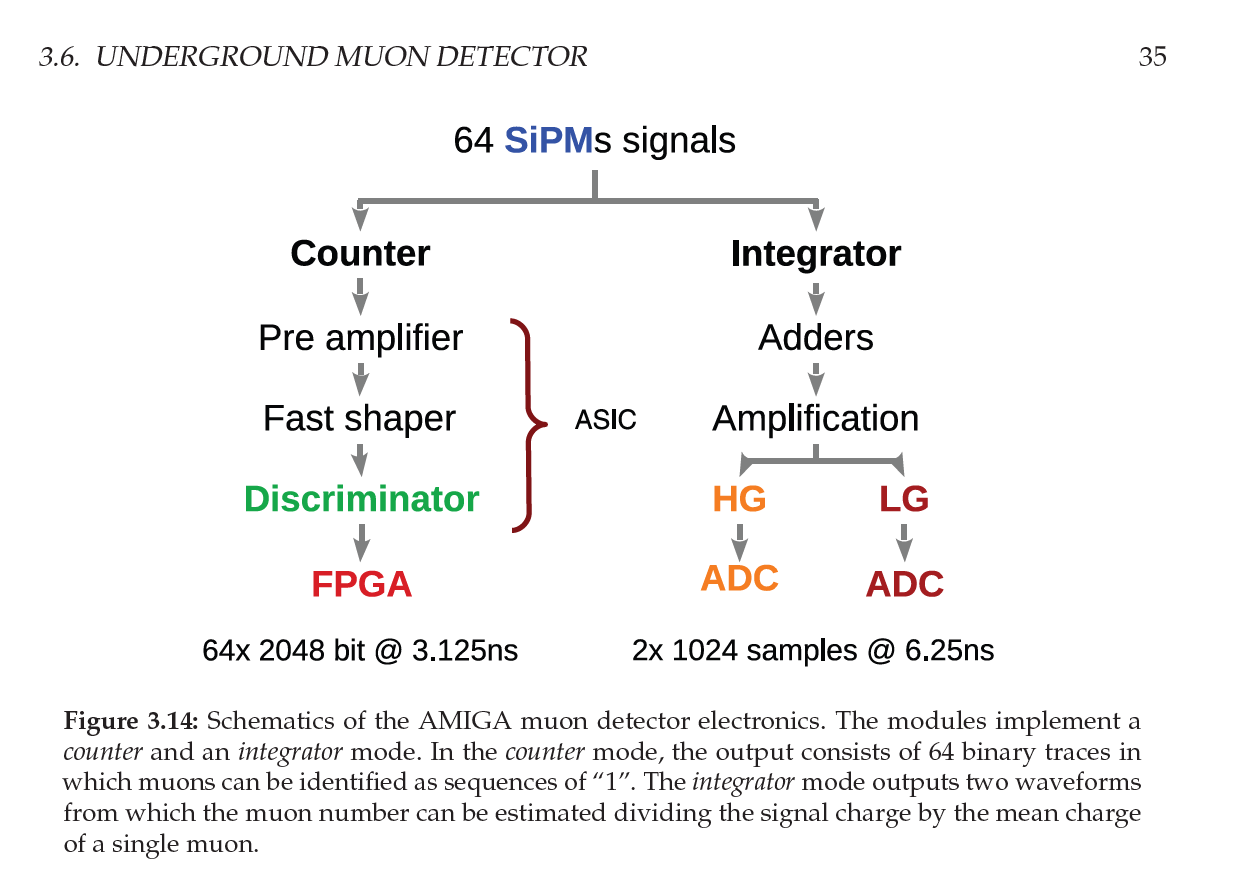
**Motivación:** hay una discrepancia significativa entre la cantidad de #1s entre la simulación y los datos medidos. Esto se puede ver en el plot de distribución de # 1s donde se comparan los datos medidos (ya sea de laboratorio o en el campo) con los datos simulados. En la simulaciones NO se pueden ver cadenas de más de quince (15) # 1s y en los datos medidos se puede ver que la cola del histograma de frecuencia de aparición de #1s tiene valores de más de 15.



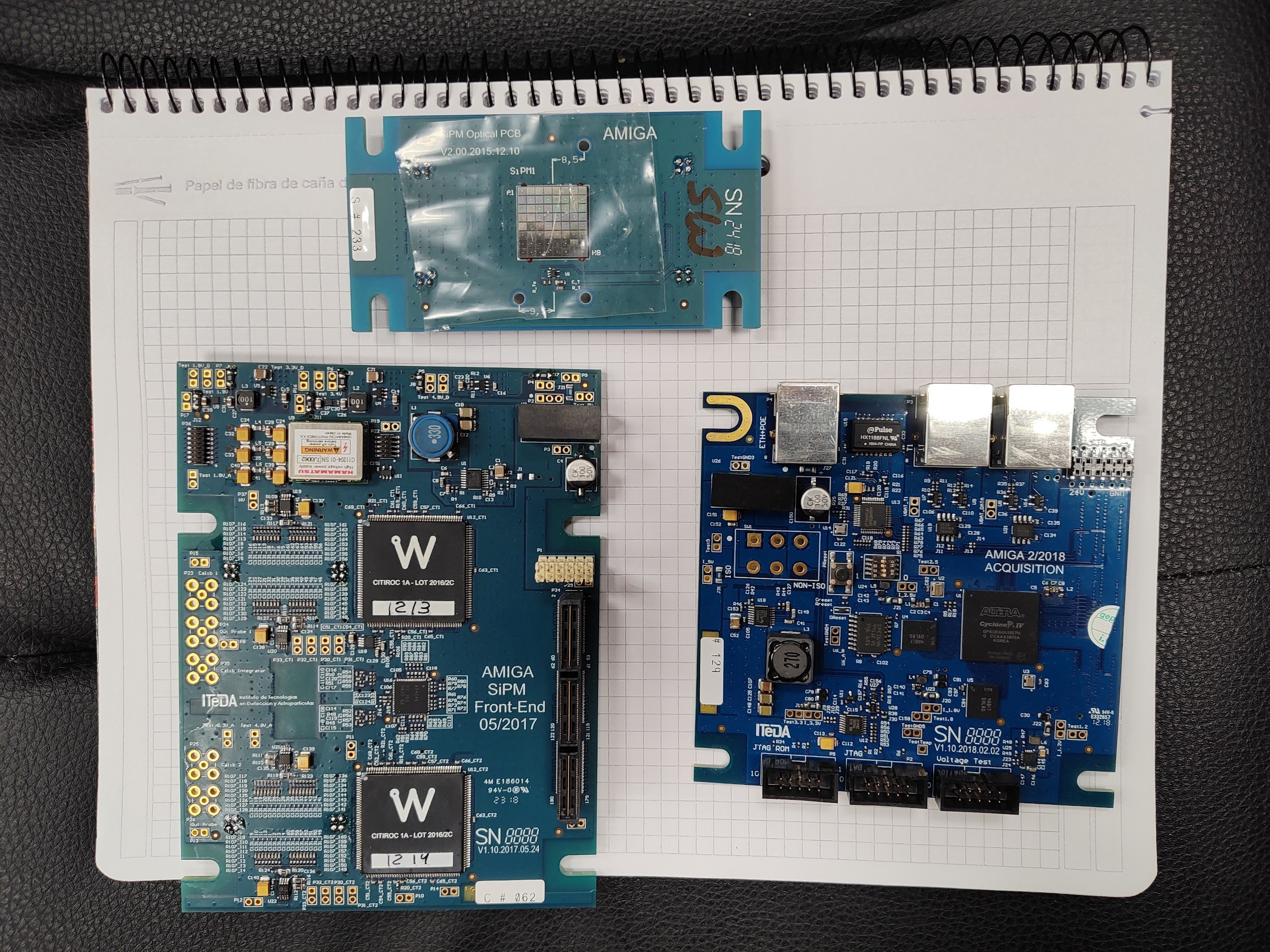
Es importante destacar que este problema apareció hace relativamente poco dado que los valores medios y los 𝝈 de las distribuciones de #1s son muy similares entre las simulaciones y los datos medidos.



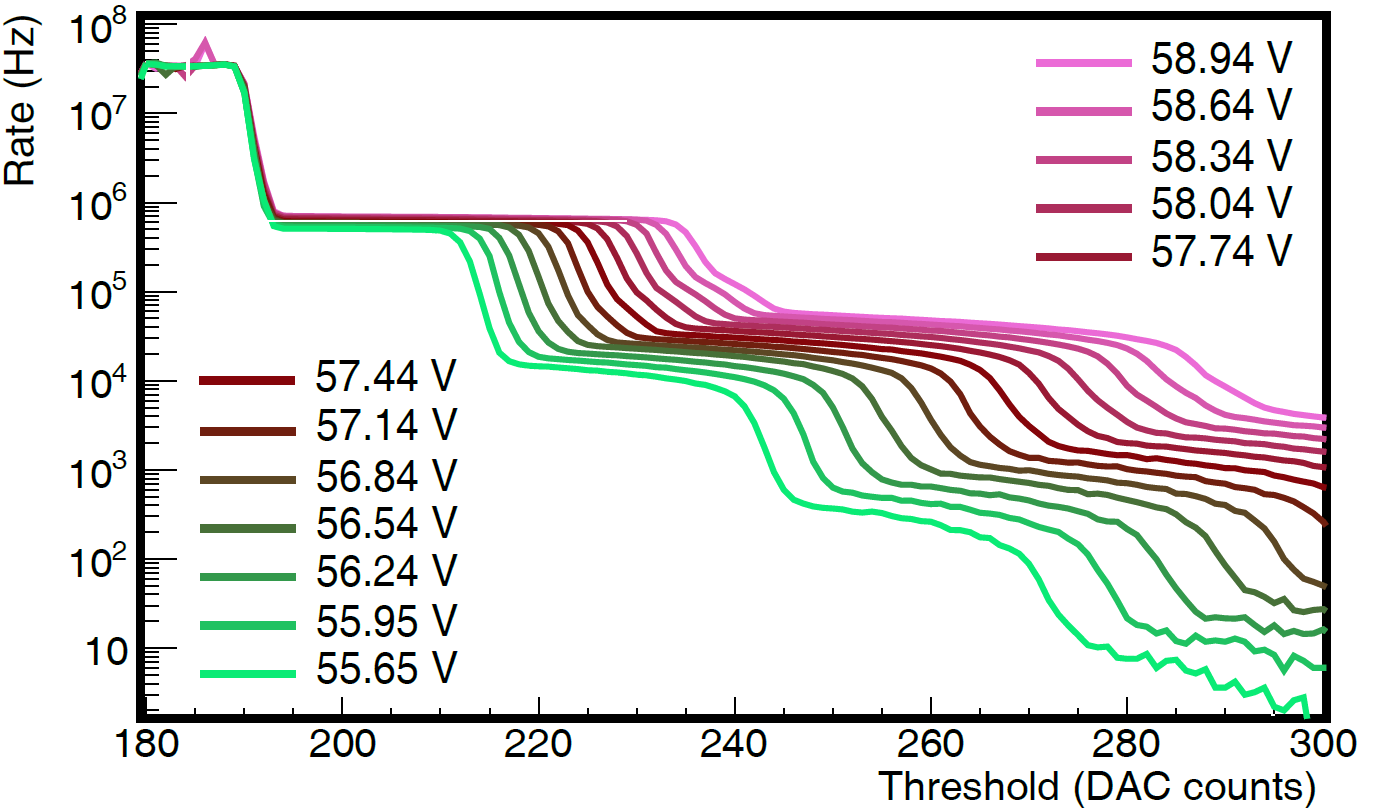
La primera línea de trabajo era lograr simular la electrónica que recibe las señales de luz que llegan de las fibras ópticas adosadas a las barras centelladores. Esta electrónica es la que se utiliza para medir los muones de los UHECR. Esta electrónica es la que procesa las señales de fotones producidas por los muones al pasar por la barra centelladora (una barra con dopantes) y las convierte en algo que se pueda medir y analizar. La señal de fotones generada por los muones es muy pequeña (de entre 20 y 40 fotones de 𝛌~500nm) con lo cual se lo hace pasar por un integrado que se llama Citiroc 1A que es a lo que yo me refiero cuando digo electrónica. Se pueden hacer dos tipos de mediciones con este integrado, una binaria y otra integral, pero yo voy a trabajar mayoritariamente con la binaria.



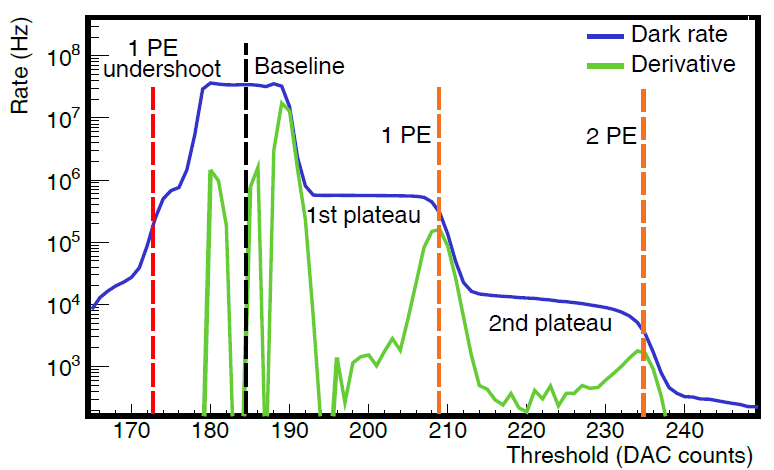
Aquí tengo una imágen de todo lo que se usa en el experimento y se puede ver toda la electrónica: en la placa de arriba se ve el array de SiPMs que reciben la señal de fotones generada por los muones y la convierten en un voltaje (corriente de electrones); luego a la izquierda se ve el integrado Citiroc 1A que realiza todo el mejoramiento y procesamiento de la señal; y finalmente en la derecha se ve el FPGA que produce los #1s de los que hablaba en la motivación y que son el objeto de estudio de este trabajo.



Es importante entender cómo es que funcionan los SiPM’s que se utilizan para transformar fotones en electrones (la luz generada puede ser por el decaimiento de los dopantes de la barra centelladora cuando llega un muón, pero también pueden recibir fotones por otras razones). Un Fotomultiplicador de Silicio (SiPM) responde de manera uniforme y genera mesetas para diferentes voltajes porque opera en modo Geiger. En este modo, cualquier fotón absorbido desencadena una avalancha, produciendo un pulso de salida estándar, en gran medida independiente del voltaje exacto, siempre que supere el umbral de ruptura. Lo que permite este comportamiento de los SiPMs es que cada una de estas mesetas (que se corresponden a un conjunto de valores de voltaje) se corresponden a número discreto y definido de fotones detectados (llamados fotoequivalentes o PE) y podemos usar esto para el análisis.



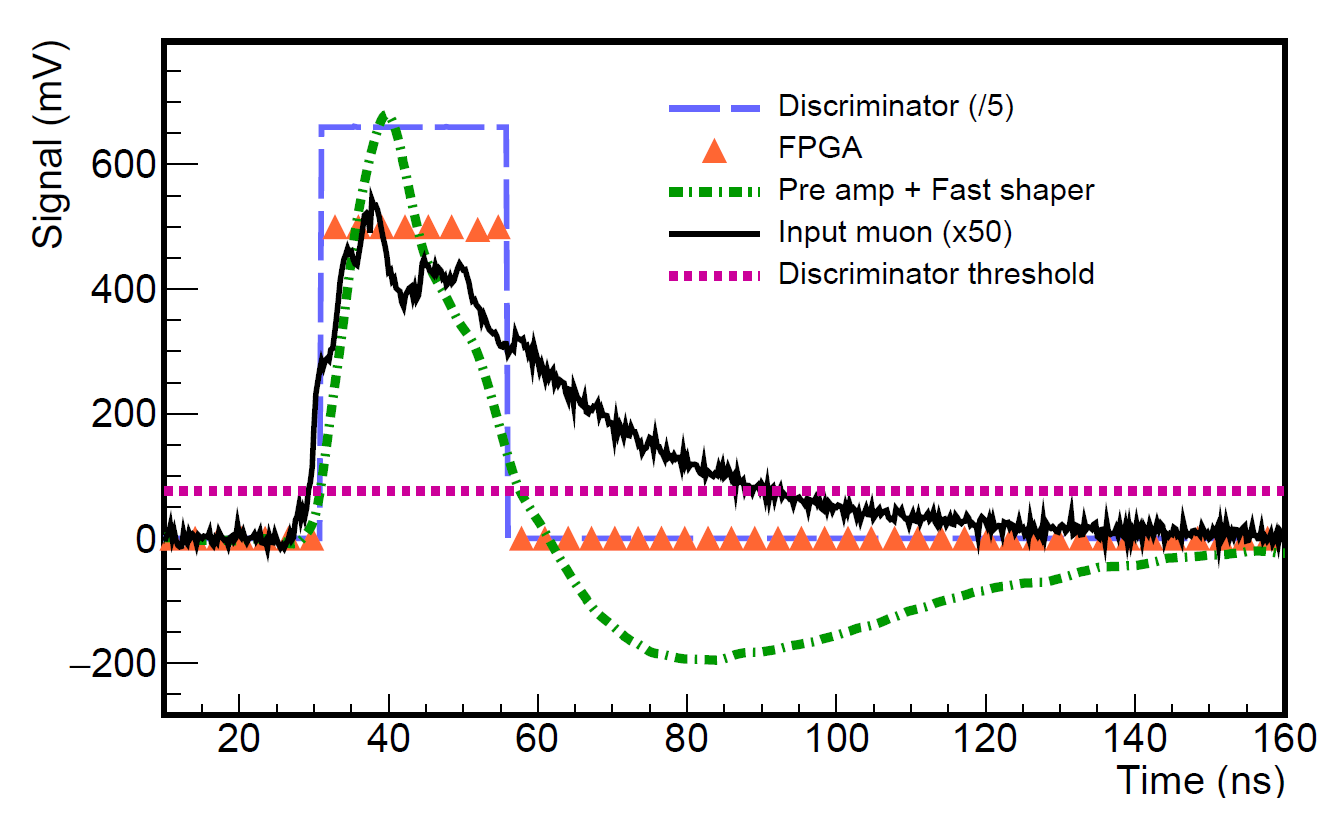
Lo que se ve es que cada meseta (conjunto de valores de voltaje) representa la entrada de 1, 2 y 3 fotones (a esto es lo que llamo 1, 2 y 3 PhotoEquivalent, PE) respectivamente. Dado que los SiPM’s tienen este espectro de PE bien definido es posible calibrarlo usando el *dark-rate pulses* (fotones generados por agitación térmica). Es por esto que estoy en proceso de armar la curva de *drak rate*, dado que es necesario para la simulación de la electrónica, específicamente del discriminador. El umbral de discriminacion para medir los #1s se colocó en 2.5PE (de esta manera se logra eliminar la mayoría de las señales generadas por los fotones de agitación térmica porque la frecuencia con la que se generan más de 2 fotones simultáneos por agitación térmica es baja). En la tesis de 2019 de Botti se hace una medición de todo esto y se encuentra la siguiente curva:



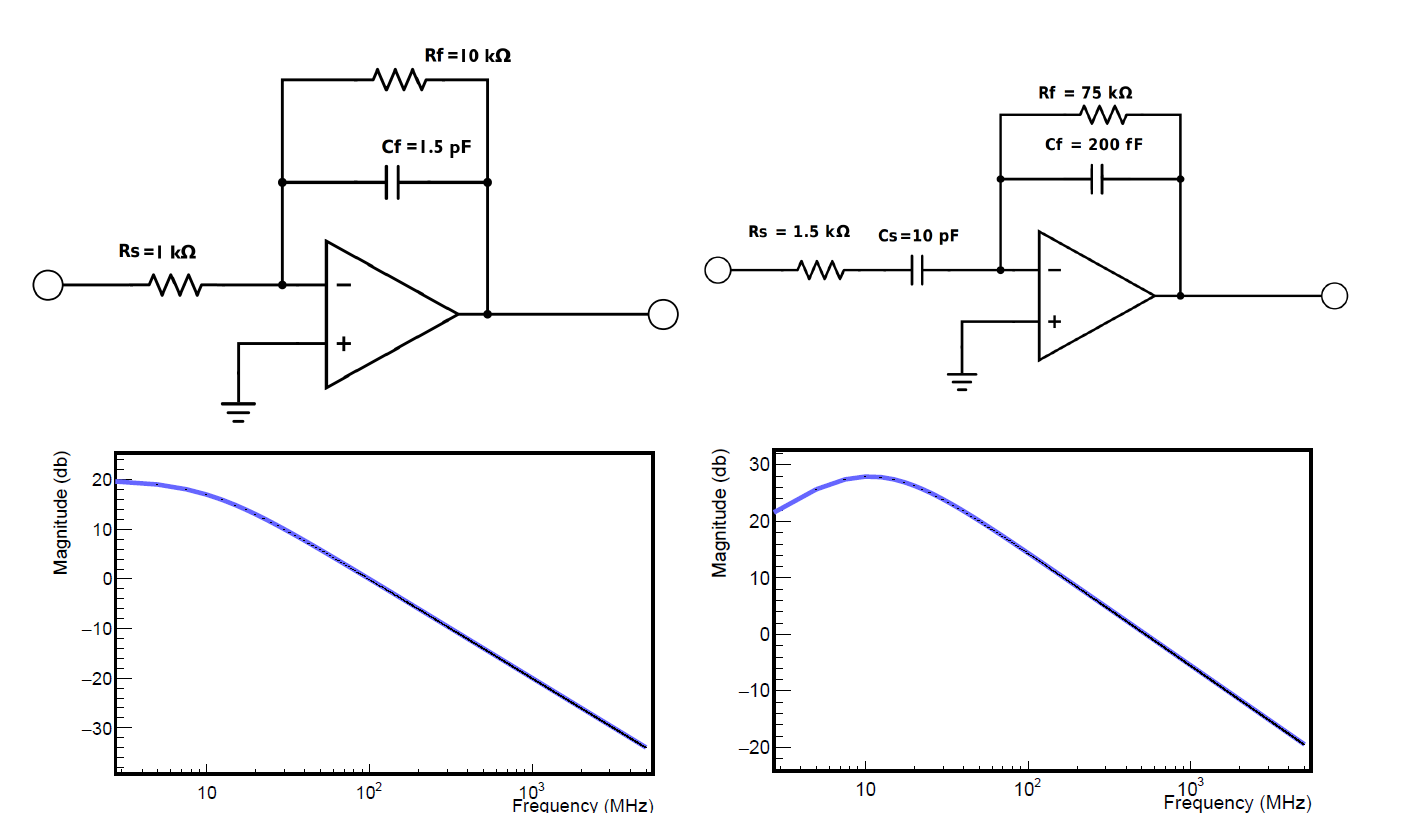
Es importante aclarar que para simular los fotones de agitación térmica, el código que los genera no debe presentar offset’s temporales (aclaro esto por algo que se explica más adelante), sino que son esencialmente la señal de 1, 2, … , n-fotones simultáneos. Estoy terminando de hacer esto utilizando los valores de frecuencia relativa que aparecen en la tesis de Botti y los valores de voltaje encontrados por mi en las simulaciones utilizando la electrónica.

Para entrar un poco más en detalle de esto, hay que entender que no es tan simple como buscar el voltaje máximo del pico de la señal de n-fotones (n-PhotoEquivalent) a la salida de la electrónica (aclaró que al final de la electrónica porque hay amplificación y deformación de la señal comparado con la entrada). Esto se debe a que el discriminador solo genera una señal cuadrada cuando el voltaje es superior el threshold de 2.5PE seteado, entonces la forma de medir la cantidad de unos asociada a esta señal es con un FPGA que recibe dicha señal cuadrada y, con una frecuencia de muestreo de (1/3.125ns), produce un 1 por cada cachito de 3.125ns de señal positiva (no se si se entiende bien).

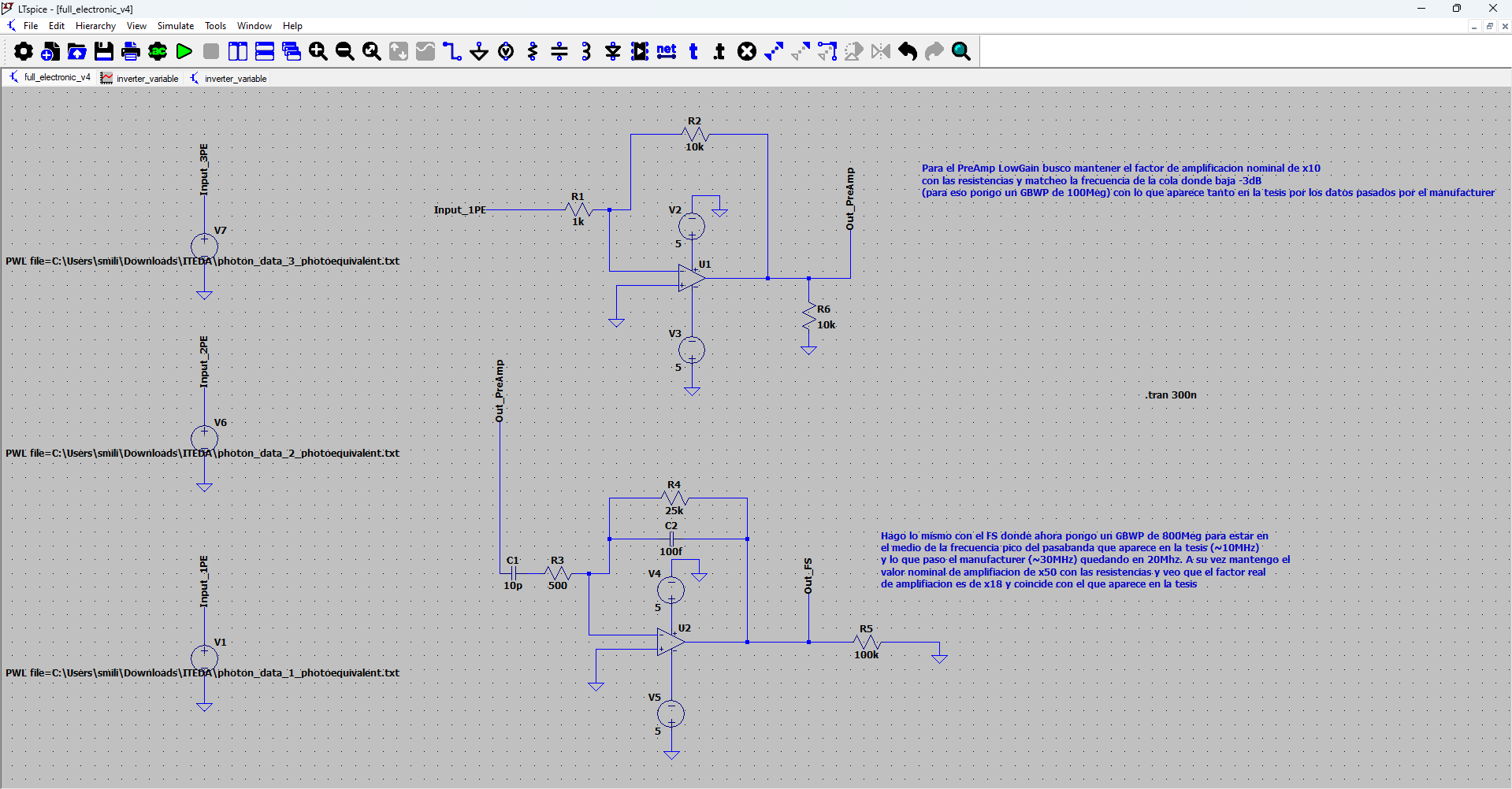
Volviendo a la electrónica, no se utilizan todos los elementos del integrado Citiroc 1A. Del mismo se usa el PreAmplificador de baja ganancia y el FastShaper, para luego pasar esta señal por un discriminador y hacer una señal cuadrada con ciertas condiciones que es lo que representa el tamaño de la señal de fotones analizando su distribución temporal (esto son las cadenas de #1s). El PreAmp no deforma considerablemente la señal y principalmente funciona como un amplificador inversor. Por otro lado el FastShaper es un componente que si deforma considerablemente la señal dado que es muy sensible a la subida de la señal que le llega pero la cola la deforma generando lo que se llama un *undershoot*.



Para todo esto me apoye en gran medida en una tesis de doctorado de 2019 de Ana Botti. Para eso tuve que investigar y entender cómo funcionan los amplificadores a nivel de la electrónica y aprender a usar el programa LTSpice para la simulación de dicha electrónica. Acá entran los conceptos de ancho de banda, ganancia nominal y GBWP. De esta manera logre simular tanto el PreAmplificador y el FastShaper (el discriminador también, pero es trivial y se puede hacer también con un script de Python). Cabe destacar varios detalles que son relevantes en esto. No tenemos todos los parámetros para los cuales hacer la simulación y dado que la electrónica es un integrado muy complejo, lo máximo que se puede hacer es simular su respuesta como circuitos RC y amplificadores (es decir que dado a su complejidad no es trivial medir todo directamente). Sin embargo, hay dos parámetros que son muy importantes y que si conocemos. En primer lugar tenemos las respuestas en frecuencia (es decir el Diagrama de Bode) de tanto el PreAmp y el FS y esto lo tenemos tanto de la tesis como de parte directa de los fabricantes del integrado. Con lo cual esto fue uno de los parámetros que se buscaron representar y mantener el las simulaciones de la electrónica que hice. Por otro lado, el otro parámetro que se busco mantener y representar correctamente fue el factor de amplificación nominal (x10 para el PreAmp y x50 para el FS).



Vale la pena aclarar que los valores de los capacitores y resistencia que aparecen en los circuitos del datasheet y la tesis dependen del factor de ganancia con lo cual esa es una de las razones por las que los valores que yo puse en las simulaciones no son idénticos a los que aparecen. Sin embargo, las partes del FS y el funcionamiento de PreAmp como un amplificador operacional sí están bien descriptas por lo cual se mantuvo su forma en las simulaciones.



Así fue como logré simular la electrónica usada en el experimento manteniendo los parámetros que se que deben quedar fijos, fijos (estos son los modelos v4 que están en el Drive). Esto ya aparenta ser una descripción correcta de la electrónica que se usa en el campo y como punto de partida para resolver la problemática que motivó el presente trabajo de laboratorio, me permite avanzar con el análisis. Para esto, hice un programa de Python que simula la llegada de fotones generados por muones por la fibra óptica al SiPM y que luego se usa como input por el Citiroc 1A. Hay dos métodos de hacer esto: por un lado hay varios papers que leí sobre como simular la llegada de fotones usando el mismo LTSpice (cosa que hice en un principio) pero también se pueden modelar los fotones de input con una función algebráica que está en la tesis con los parámetros ajustados y fue lo que yo use para todo el análisis subsiguiente. Para esto me armé un programa de Python que me permitió hacer las simulaciones de los fotones bajo ciertas condiciones.

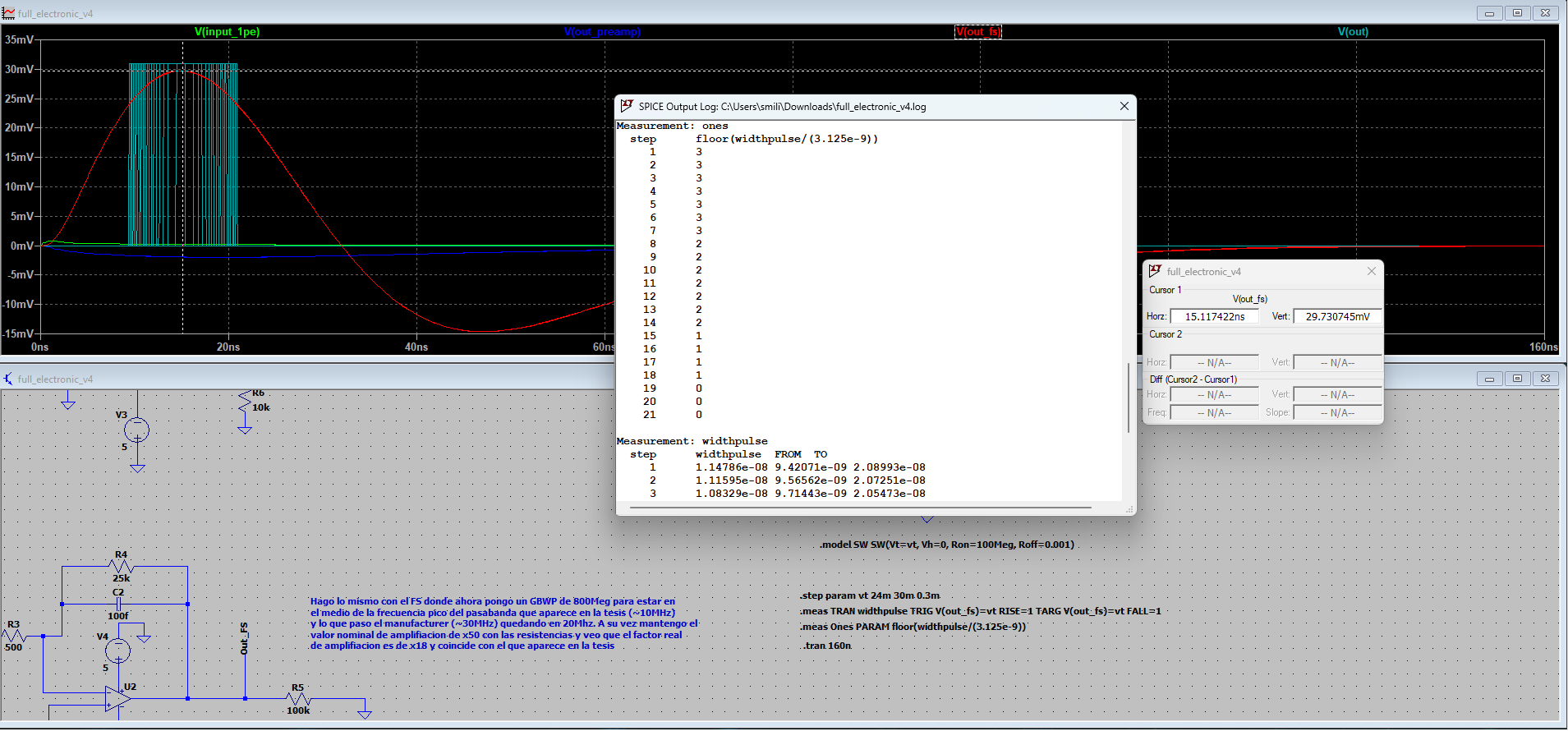
Acá vale la pena hacer un mini sidestep y explicar qué es lo que sucede cuando un muón llega a una barra centelladora. Lo que sucede es que hay una distribución temporal en la que los fotones se generan en la barra y llegan al SiPM debido a varias razones: las trazas de los muones no son perfectamente perpendiculares a la barra con lo cual hay muones que recorren más porción de la barra y los fotones se generan en distintos lugares (en 3ns el muón recorrió 1mts y puede volver a excitar a un dopante con lo cual esa diferencia posicional es relevante dado los tiempos característicos del experimento); que los procesos de decaimiento de los dopantes de la red de la barra centelladora son probabilísticos con lo cual no suceden todos al mismo tiempo; entre otras cosas. Es por esto que uno de los parámetros más relevantes que tiene la simulación de los fotones es el offset temporal que tiene cada uno, que lo modele sampleandolo de una distribución exponencial decreciente (dado que ese es el tipo de distribución de probabilidad que tienen los offsets temporales como lo indica la tesis de Botti). Sin embargo, no está hecha la medición de cual es el factor de rate de la exponencial decreciente y no es algo que pueda ser fácilmente medido (este factor debería implicar las variaciones de la trazas y los procesos de decaimiento de los dopantes). Tampoco queda claro cuál es ese valor que se usó con las simulaciones previamente hechas y por eso es que me tomé el trabajo de hacerlas de cero nuevamente. Con lo cual quiero encontrar el parámetro justo que me permite reproducir los histogramas de la distribución de #1s medidos en el campo. Este será el tema PRINCIPAL de lo que voy a terminar de investigar para ver si esta es la causa de las diferencias en los histogramas y ver si modificando esto se puede llegar a hacer aparecer la cola del histograma en las simulaciones.

06/06

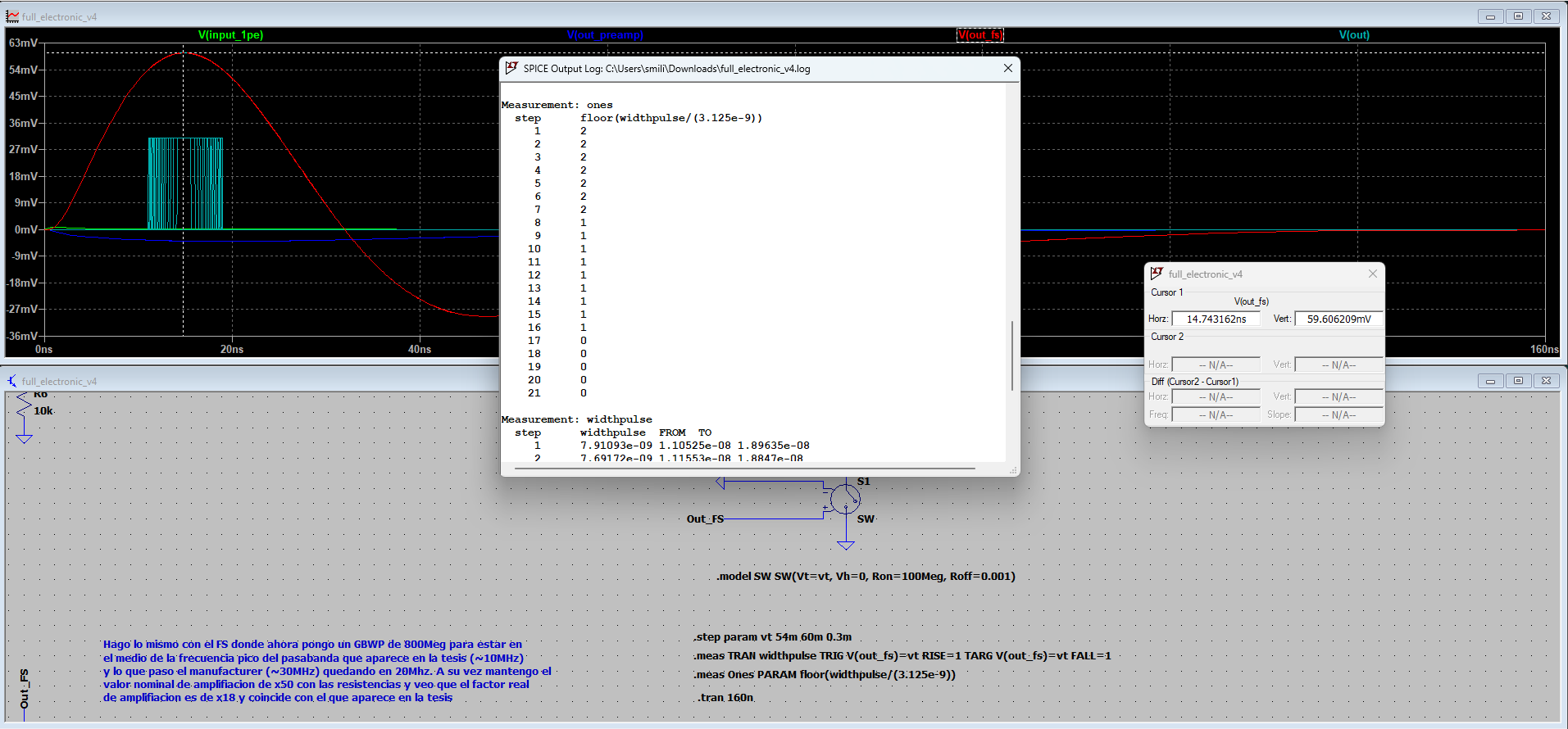
La idea del trabajo de hoy es ver si hay alguna variabilidad cuando simulo los pulsos de 1-2-3PE frente al valor medio de los mismos. Es importante verlo porque los pulsos se generan sampleando los parámetros de la función con gaussianas (creo que no debería ser mucho y van a dar parecidos). Esto es importante para calcular los thresholds.

Ya hice los códigos para hacer los n-fotoequivalentes medios y los paso por el full\_electronic\_v4 en el LTSpice (tengo que correr el circuito y ver que pasa en los logs → a partir de ver cuando tengo menos de un 1, digo que ese es el threshold de 1PE).

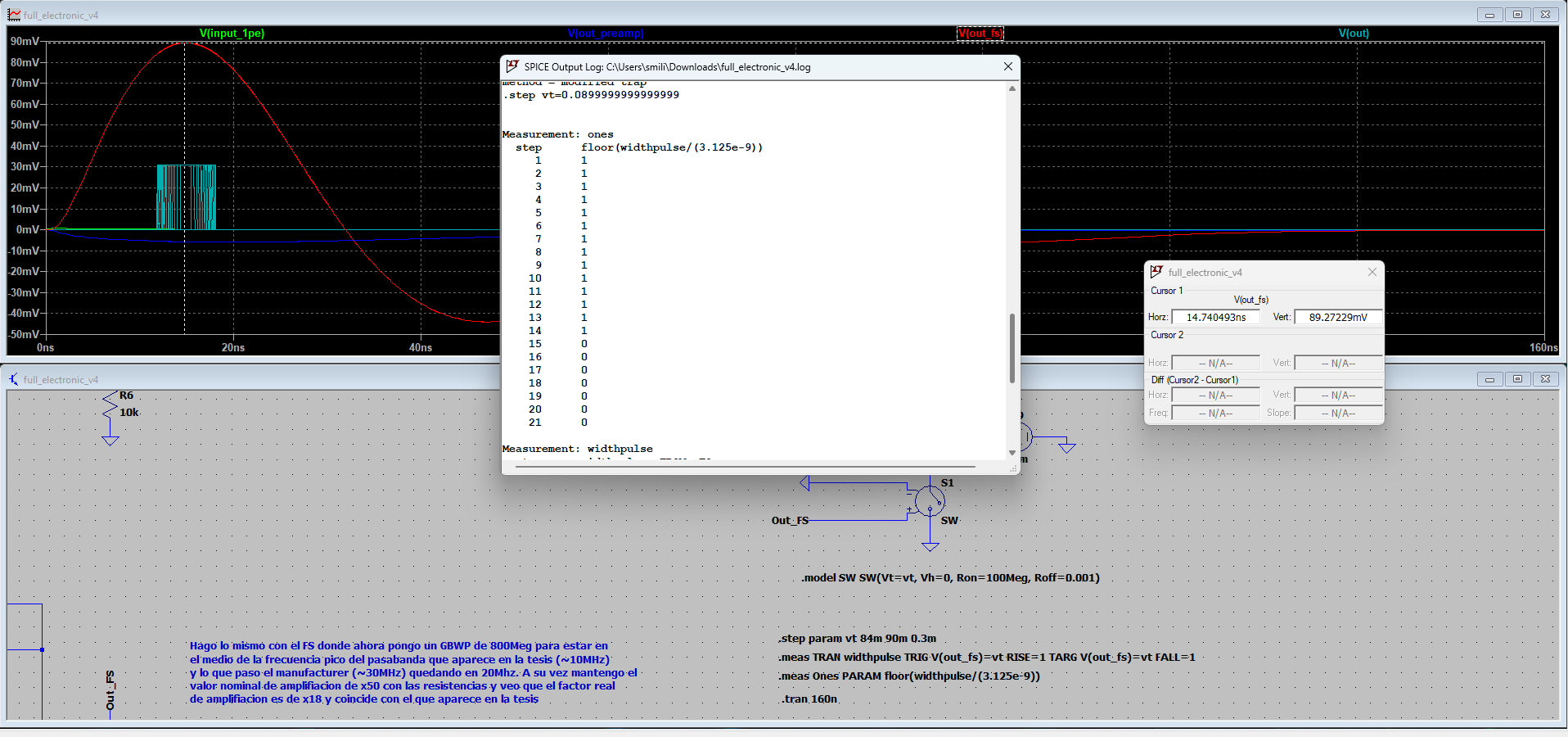
1. Para 1PE: el threshold está en 29.1mV (hay una leve diferencia con el máximo que está alrededor de 29.75mV) -> debería ver si es necesario hacer los pasos más pequeños para tener mejor precisión.



1. Para 2PE: el threshold está en 58.5mV (hay una leve diferencia con el máximo que está alrededor de 59.6mV)



1. Para 3PE: el threshold está en 87.9mV (hay una leve diferencia con el máximo que está alrededor de 89.2mV)



Arme un script para recuperar la curva de dark-rate con las frecuencias relativas que aparecen en la tesis de Botti. Para esto, uso la función tanh invertida para recuperar la forma que tienen las transiciones de los fotoequivalentes (y la desplazo sobre el eje y hacia arriba para que terminen en 0 restando 1 y las desplazo en el eje x hacia donde esta el threshold del fotoequivalente respectivo).

Llego al resultado que estoy buscando: el threshold usado en las mediciones de 2.5PE es de 73.2mV!! → este código es el que se llama dark\_rate

Ahora voy a simular la llegada de 100 muones, pasarlos por mí electrónica y ver la cantidad de 1’s para hacer el histograma de frecuencias de unos. Lo que hago para simular los muones, primeramente, es armar el gráfico de cantidad de fotoequivalentes según su posición en la barra centelladora. Eso me da, sin embargo, el valor medio de la cantidad de PE, y como yo quiero tomar en cuenta la variabilidad de la generación de fotoequivalentes, voy usar este valor medio como mi parámetro de Poisson y samplear el valor real simulado de la cantidad de fotoequivalentes de allí. Así es como ya tengo hecha la relación entre la llegada de mis 100 muones y la cantidad de fotoequivalentes, con lo cual puedo hacer pasar esta lista de 100 cantidades de fotones por mi función generadora de señales codeada anteriormente. Es importante destacar que esto debería repetirse para muchos valores de λ (factor de decaimiento de la función del offset temporal), pero esto lo voy a hacer en el futuro, primero quiero poder recuperar el histograma de frecuencia de la simulación que aparece en la primera página del trabajo.

Ya tengo el código que hace los distintos pulsos haciendo todo lo que está antes (codigo = muon\_sim\_with\_distance\_and\_poissondist) → sólo falta entonces pasar todo esto por la electrónica y ver la cantidad de 1’s de cada uno y hacer el histograma. Falta para esto ver si hay una forma de hacer un loop en LTSpice sobre los archivos o si puedo hacerlo mirando el código directo del LTSpice. La otra idea es enganchar todos los pulsos de muones en un mismo archivo separándolos temporalmente para que el undershoot del FS no me joda, pero esto es para estudiar si todo falla.

07/06

Hoy estuvimos discutiendo y viendo el código grande y posta de las simulaciones del Auger. Conclusiones de hoy:

* La función de la que se samplea el offset temporal tiene que ser una doble exponencial decreciente (con los tiempos característicos de la emisión de fotones de la barra centelladora y de la fibra óptica). Estos tiempos característicos están en el mega código y son 3.5ns (fibra) y 3.7ns (barra centelladora). La idea es hacer la simulación con estos valores con el objetivo de ver si puedo recuperar los valores del histograma de las simulaciones que fueron la motivación de este trabajo.

Luego la idea después es ver si se puede observar si la función de la cual se calcula el offset temporal, se puede hacer (es más, se debería hacer así) poniendo un parámetro variable multiplicativo del estilo de (la idea es que la forma en la que lo haga ahora no tenga los factores multiplicativos y sea plain). Esto es importante porque sino podría pasar que el me deforme el principio del histograma de distribución de unos y la parte del principio de las simulaciones comparado con los datos ya da bien.

La otra cosa que es súper relevante y que probablemente tenga sea una de las posibles razones del error es que va a haber que variar los valores de alguno de los *tau*.

Entonces es importante modificar la función offset.

* No hay una forma de hacer un loop for sobre archivos en el LTSpice y además es problemático el tema de que aunque hubiera me sobreescribe el archivo .log y eso hace que no pueda ver los resultados por separado. La idea entonces va a ser hacer lo que discutimos ayer de agarrar todos las señales, ponerlas en un mismo archivo poniendo suficiente espacio entre ellas (Fede dice de poner 1 microseg, pero para mi con 500 ns ya tamos, lo tengo que pensar y ver en el código). La idea es hacer esto por separado a los archivos individuales, es decir, generar los pulsos solos, guardar esos .txt y después agarrar y unir todos los archivos en uno nuevo llamado la unión.
* La idea para empezar a probar, no es usar el archivo que genera los muones usando la curva de cantidad de muones en función de la distancia, sino que a manopla con el código original, generar tiras de 1 a 50 muones. Sin embargo como esto esta con la función que randomente genera el offset temporal de cada uno, la idea es repetir esto muchas veces (quiero aclarar que si hacemos esto muchas veces, ~50 como decía Fede, volvemos al mismo drama de que el LTSpice no tiene for loops).
* Es importante algo que vi hoy que tiene que ver con que el PreAmp: está alargando mucho la señal de input de muones (o fotones) y esto está haciendo que incluso para pocos PE tengo cadenas de muchos 1’s (del orden de seis 1’s cuando solo mando 15 fotones). Esto se debe a que el GBWP es muy chico (actualmente está en 100Meg) → es importante que no estuvo puesto al azar el GBWP sino que está fijado para que la respuesta en frecuencia sea similar a la que nos dio el manufacturer (~10MHz). Hice la prueba y si pongo GBWP = 500Meg, se arregla este problema del alargamiento de la respuesta de la señal post PreAmp comparado con el pulso original. No hice la prueba para ver qué pasa con las cadenas de 1’s pero esto debería arreglar el problema. Estoy convencido de esto porque el FastShaper no me parece que esté alargando mucho temporalmente la señal que le llega del PreAmp.

13/06

Hoy estuve todo el dia trabajando en modificar el código con el nuevo offset (estuvo difícil y termine decantandome con usar dos samples de exponenciales decrecientes con los tiempos característicos correctos) y buscando la forma de generar el archivo que concatena todas las señales para poder pasarlo por el LTSpice.

Falta discutir como hacer para eliminar la cosa random del muestreo del offset. Podemos ver que pasa con los histogramas de frecuencia pasando una sola tira de datos si