Trabajo de Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

Desarrollo de un analizador mono-canal digital mediante un equipo STEMLab 125-14 Red Pitaya

MEMORIA

Autor: Xavier Pradas Barro
Director: Alfredo de Blas del Hoyo

Convocatòria: Junio 2021

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resumen

El objetivo principal de este TFG es llevar a cabo la creación un analizador monocanal mediante el equipo *STEMlab 125-14 Red Pitaya*. Luego, se desea determinar si podría llegar a sustituir con garantías a un analizador monocanal clásico del laboratorio de Instrumentación Nuclear. Con ello, no sólo se pretende ahorrar a nivel de recursos económicos, sino también dar un paso adelante en cuanto a practicidad.

Durante el transcurso del proyecto, se muestrean distintos tipos de señal mediante *Red Pitaya* con la finalidad de comprender el funcionamiento de sus parámetros de adquisición. Paralelamente, se va elaborando un algoritmo de detección de máximos locales de señal (necesario para la creación del monocanal) haciendo uso del lenguaje de programación *Python*.

Al término del trabajo, el analizador monocanal se ha podido crear satisfactoriamente y proporciona resultados muy parecidos al de un analizador monocanal de laboratorio. Los conteos realizados usando distintas fuentes radiactivas así lo avalan.

Sin embargo, se concluye que la sustitución no se puede considerar ni mucho menos viable. El problema radica en la extrema lentitud de muestreo del equipo *Red Pitaya*. Trabajando vía control remoto, la adquisición de señal no es continua (se adquieren bloques de 16K muestras). Este hecho se intenta solucionar creando un bucle de adquisición, pero el equipo *Red Pitaya* no es capaz de concatenar dos iteraciones, requiere de un tiempo muerto entre muestreos de unos *300 ms*.

1. Introducción

1.1. Objectivos del proyecto

A continuación, se presentan los objetivos establecidos para este trabajo final de grado:

- Instalar un equipo *STEMLab 125-14 Red Pitay*a en el laboratorio docente de Instrumentación Nuclear.
    - Establecimiento del modo de funcionamiento más adecuado.
- Poder realizar muestreos satisfactorios de pulsos generados en detectores de radiación ionizante. Determinar los límites de aplicación para los pulsos a tratar.
    - Establecer las condiciones de muestreo más adecuadas para las señales generadas en un detector de radiación ionizante.
- Crear de un analizador monocanal (selector de pulsos) con el equipo STEMLab 125-14 Red Pitaya.
    - Implementar un sistema de conteo de los pulsos de entrada seleccionados.
- Efectuar una comparación del analizador monocanal creado con un analizador monocanal comercial de instrumentación nuclear.

1.2. Alcance del proyecto

Para poder cumplir los objetivos del proyecto, se debe pasar por una serie de etapas que pueden agruparse de la siguiente forma:

1. Familiarización con el material: teniendo en cuenta la inexperiencia del usuario, en la fase inicial del proyecto, es necesario un proceso de estudio del funcionamiento del equipo *Red Pitaya*. Complementariamente, también se empieza a experimentar con instrumentos del laboratorio que se usan durante todo el transcurso del proyecto (por ejemplo, el osciloscopio).

Aunque lo que se consigue en esta etapa no supone grandes avances en el trabajo, sin duda la experiencia adquirida ayuda a encarar las siguientes fases con las ideas mucho más claras.

2. Comprensión del código de adquisición de señal: *Red Pitaya* dispone de un seguido de comandos de adquisición de señal que es indispensable dominar para poder realizar las operaciones de muestreo adecuadamente.

3. Desarrollo del código: para la creación del analizador monocanal se hace uso del lenguaje de programación *Python*. Lógicamente, este código va evolucionando a medida que se producen avances en el proyecto.

4. Realización de los montajes pertinentes en el laboratorio de instrumentación nuclear: durante todo el proyecto se trabaja con distintos instrumentos del laboratorio. Es necesario entender el funcionamiento de cada uno de ellos, sobre todo de cara a la parte final del proyecto, momento en el que se realiza el montaje para comparar los dos monocanales.

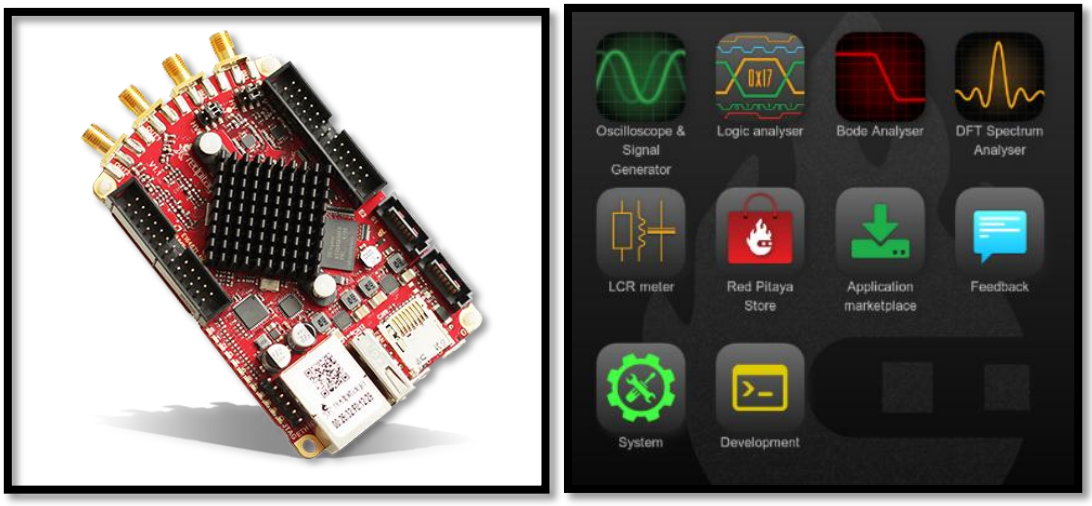
2. Conceptos previos

2.1. Equipo STEMlab Red Pitaya

El equipo *Red Pitaya* es un microcontrolador potente que incluye una matriz de puertas lógicas programables (*FPGA)*. Cuenta con un sistema operativo basado en *Linux* y una comunidad de usuarios que desarrollan y comparten sus aplicaciones. Entre sus características, hay dos factores que la diferencian de otros sistemas similares, como podría ser *Raspberry Pi*:

- *Dispone de un hardwar*e dedicado a la adquisición y generación de señales analógicas.
- Se trata de un dispositivo integrado que combina una computadora con procesador *ARM (Advanced RISC Machine*), una *FPGA* y una placa electrónica con convertidores de analógico a digital y vicerversa *(ADC* y *DAC*). *[1]*

Debido a esas ventajas, la *Red Pitaya* puede reemplazar algunos elementos de laboratorio. Los usuarios pueden acceder a la interfaz web de *Pitaya* conectándola a una red local *LAN*. Esta interfaz consta de diversas aplicaciones disponibles, tales como osciloscopio, generador de funciones, analizador de espectros... La *Figura 1* y la *Figura 2* muestran su aspecto físico y su interfaz web, respectivamente.



Figuras 1 y 2: *Red Pitaya y su interfaz web. [2][3]*

La *Red Pitaya* se puede controlar remotamente vía *Ethernet* utilizando distintos lenguajes de programación como *Matlab* o *Python* a través de la lista de comandos *SCPI (Standard*

Commands for Programmable Instruments). Para la realización de este TFG, se ha usado *Python*.

El servidor *SCPI* utiliza un conjunto de órdenes que son reconocidas por los instrumentos para permitir acciones específicas. Existen conjuntos de órdenes destinados a la adquisición de datos a partir de entradas analógicas rápidas, a la generación de señales o al control de otros periféricos de la plataforma Red Pitaya.

Para este proyecto, se utiliza mayoritariamente el bloque de comandos *SCPI* destinados a la adquisición de datos.

2.2. ¿Qué es un analizador monocanal (SCA)?

Un analizador monocanal o *Single Channel Analyzer (SCA)* produce un pulso lógico de salida solo si el valor máximo del pulso de entrada está dentro de un rango de tensiones establecido con dos límites predefinidos. La selección de la ventana de tensiones es llevada a cabo por el usuario según sus intereses.

Con el objetivo de que los pulsos de tensión de entrada del *SCA* sean fácilmente analizables, previamente se hace pasar la señal por un amplificador (a lo largo del proyecto se detalla más la función de este instrumento). La *Figura 3* muestra tres pulsos distintos que podrían llegar al analizador monocanal. La función de éste no es otra que discriminar esos pulsos según sus niveles de tensión:

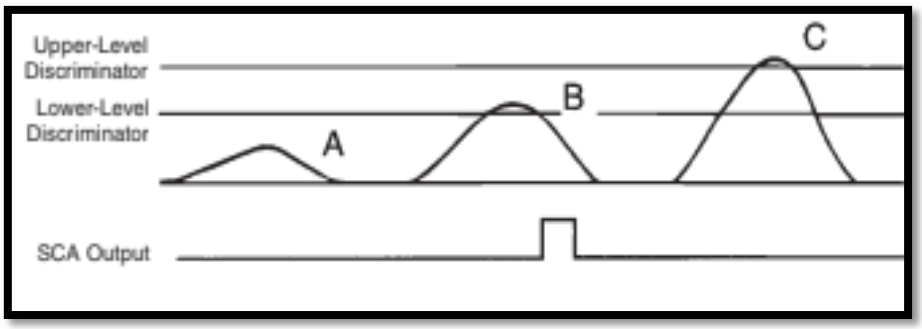


Figura 3: *Funcionamiento de un analizador monocanal* ***[4]***

- **Pulso A:** la tensión de su máximo es menor a la del límite inferior definido. El *SCA* no genera ningún pulso de salida.

- **Pulso B:** su máximo cae dentro de los límites establecidos. El *SCA* genera un pulso digital cada vez que analiza un pulso de este tipo. Complementariamente, se suele usar un contador para acumular el número de pulsos digitales generados.
- **Pulso C:** tiene una amplitud mayor a la del límite superior establecido. El *SCA* no genera ningún pulso de salida.

Una vez se conoce el concepto, el objetivo final es poder llegar a implementar un analizador mono-canal mediante el equipo Red Pitaya. Para ello, hay que seguir una serie de pasos que se dividen en tres grandes bloques de experimentación.

3. BLOQUE I: MUESTREO DE SEÑALES

Para llevar a cabo una operación de adquisición de señal con la *Red Pitaya*, el primer paso indispensable es realizar las conexiones pertinentes. Lógicamente, debe conectarse a la corriente. Además, es necesario conectarla vía *Ethernet* con el ordenador en el que se esté manejando el código de adquisición. Por último, con el cable *Ethernet* ya conectado, se debe activar el servidor *SCPI* en la interfaz web de *Red Pitaya (*secuencia de botones *Development-SCPI Server-Run). [5]*

En cuanto a las señales de entrada, siempre son pulsos de tensión. Las señales que se desean adquirir simplemente se deben conectar a la *Red Pitaya* a través de uno de sus *inputs* mediante *cables BNC-SMA*. En realidad, lo que hace *Red Pitaya* es un muestreo de la señal recibida. Básicamente, va adquiriendo valores de la señal continua que le llega. El tiempo que tarda entre muestra y muestra se denomina periodo de muestreo.

Otro aspecto a tener en cuenta para realizar una correcta adquisición es el establecimiento de la ganancia de entrada de la *Red Pitaya*. Se hace a través de los *jumpers* y sirve para definir el rango de tensiones de trabajo. Según la amplitud de la tensión de entrada, estos *jumpers* se colocan en posición '*Low Voltage: LV'* (para tensiones menores a *1V*) o en posición '*High Voltage: HV'* (amplitudes de *1 a 20 V*). *[6]*

En la *Figura 4* se pueden apreciar las conexiones comentadas. En este caso, la señal de entrada es de amplitud superior a *1 V* y, consecuentemente, los *jumpers* del *input1* están colocados en modo *HV*.

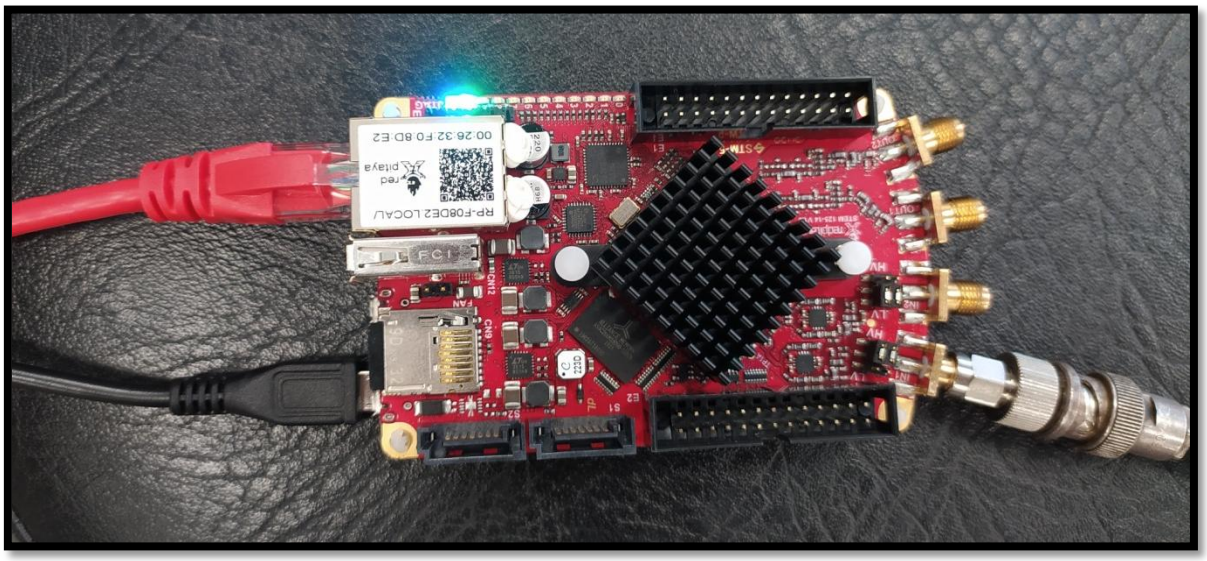


Figura 4: *Conexiones de Red Pitaya para un proceso de adquisición de señal. Fuente: Propia*

Cabe destacar que, aunque para este proyecto no se ha probado, podría resultar interesante usar los dos Inputs de la *Pitaya* a la vez (por ejemplo, para realizar una comparación de señales).

3.1. Comandos y parámetros de adquisición de señal

Para controlar la Red Pitaya de forma remota, existen los llamados comandos *SCPI* (*Standard Commands for Programmable Instruments*). [7]

Como ya se ha comentado anteriormente, la *Red Pitaya* puede ejercer funciones muy diversas pero, para este proyecto, principalmente se usan sus capacidadades de adquisición de señal. Consecuentemente, en los *Subapartados 3.1.1-3.1.5* se presentan los comandos y parámetros de adquisición más importantes.

3.1.1. Longitud de un muestreo

Antes que nada, es importante tener en cuenta una limitación que condiciona los procesos de adquisición de manera clara: Red Pitaya obtiene un total de 16.384 muestras de tensión por cada proceso de adquisición. Este parámetro es fijo, se trata de una restricción y, por lo tanto, no se puede modificar.

Sin embargo, si bien es cierto que la cantidad de muestras adquirida es fija, el tiempo que dura un periodo de adquisición puede ser modificado mediante el factor de diezmado, presentado a continuación.

3.1.2. Factor de diezmado

El diezmado (o *decimation*, en inglés) es un parámetro que modifica la frecuencia de muestreo o, lo que es lo mismo, el tiempo transcurrido entre muestra y muestra adquiridas (periodo de muestreo). Así pues, resulta muy útil, ya que permite adaptar el proceso de adquisición en función de la velocidad de la señal de entrada.

A continuación, en la *Tabla 1* se muestran todos los valores que puede tomar el factor de *diezmado*, así como su equivalente frecuencia de muestreo y el tiempo total que dura un bloque de adquisición en cada caso.

Diezmado	Frecuencia de muestreo (Kilo muestras / s)	Periodo de muestreo (ns)	Tiempo total de un proceso de muestreo (ms)
1	125.000	8	0,13
8	15.600	64,03	1,05
64	1953	512,24	8,39
1024	122,07	8.192	134,22
8192	15,26	65.536	1.074
65536	1,91	524.289	8.590

Tabla 1: *Parámetros de adquisición en función del factor de diezmado [8]*

Analizando la tabla se concluye que, para señales de entrada muy rápidas, es recomendable trabajar con diezmados pequeños, ya que, para no perder información acerca de la señal, se requiere de una frecuencia de muestreo elevada.

Si se trabaja con señales de entrada más bien lentas, en cambio, es necesario hacer uso un diezmado elevado, ya que trabajar con una alta frecuencia de muestreo no tendría ningún sentido, sería una pérdida de tiempo.

3.1.3. Tiempo muerto

El problema pricipal de muestrear con diezmados pequeños es que el tramo de señal que se adquiere es muy corto. Eso provoca que, si la señal no es periódica, no se pueden sacar conclusiones.

Existe una solución aparentemente sencilla que permite aumentar el tiempo total de muestreo: aplicar un bucle de adquisición, es decir, ir obteniendo bloques de 16.384 muestras continuamente hasta que el tiempo total de muestreo sea el deseado.

Aplicando este bucle, sin embargo, aparece otro problema: la Pitaya necesita un tiempo considerable para “recuperarse” entre muestreo y muestreo. Éste se conoce como

tiempo muerto, y se puede comprobar que su valor es de unos 0,3 segundos. Pongamos por caso que se desea muestrear una señal rápida durante 10 segundos. Ésta situación hipotética se analiza en la *Tabla 2*:

Diezmado	Cantidad de muestreos	Tiempo necesario
1	76.294	6 h 22 min
8	9.537	47 min 40 s
64	1.192	5 min 58 s

Tabla 2: *Tiempo total necesario para muestrear 10 segundos de señal. Fuente: Propia*

Se puede observar que, usando diezmados pequeños, se necesita una cantidad descomunal de tiempo para lograr muestrear apenas 10 segundos de señal. Por lo tanto, el tiempo muerto supone una grandísima limitación a la hora de muestrear señales rápidas.

3.1.4. Trigger

El *trigger* (o disparo) podría definirse como la condición que desencadena el inicio del proceso de adquisición de señal.

Existe la opción de transmitirle a la *Pitaya* la orden de empezar el muestreo inmediatamente (*Acquire Trigger Now*). Sin embargo, también es común usar algún tipo de "detonante" como, por ejemplo, esperar a que la señal de entrada adquiera un cierto nivel de tensión a partir del cual se ejecute el muestreo (*Acquire Trigger Level*). [7]

3.1.5. Control y lectura de señal

Existen tres comandos de control que tienen la función de gestionar los tempos de la adquisición:

- ***'ACQ: START'*****:** como su nombre indica, sirve para iniciar el proceso de adquisición.
- ***'ACQ: STOP':*** finaliza el proceso de adquisición de señal.
- ***'ACQ: RST':*** además de finalizar el proceso de adquisición de señal, reinicia todos los parámetros a sus valores por defecto.

En cuanto a los comandos de lectura, el más importante es ***'ACQ: SOUR<n>: GAIN***

<par>'. Tiene la función de indicar a la *Pitaya* de dónde tiene que leer la señal (<*n*>: *input 1* o *input 2*) y en qué modo leerlos, siempre concordante a la posición de los *jumpers* (<*par*>: *LV* o *HV*). *[7]*

3.2. Experimentación

Una vez explicados los comandos, parámetros y limitaciones de adquisición del equipo *Red Pitaya* más destacables, conviene comentar de forma cronológica el proceso que se sigue para llegar a cumplir uno de los principales objetivos del proyecto (Apartado *1.1*): *"Establecer las condiciones de muestreo más adecuadas para las señales generadas en un detector de radiación ionizante"*.

Es muy importante destacar la utilidad del osciloscopio, que es primordial tanto en la primera fase del trabajo como en las posteriores. Coloquialmente, se podría decir que actúa como guía: lo que se ve en el osciloscopio tiene que coincidir con la señal adquirida por la Red Pitaya.

Así pues, durante todo el proyecto, con el objetivo de asegurar la calidad del muestreo, toda señal muestreada se grafica también con el osciloscopio. Principalmente, se ha usado el modelo *Tektronix TBS 1152B* del laboratorio de Instrumentación Nuclear.

3.2.1. Generador de funciones

En esta primera fase del proyecto, únicamente se busca la familiarización del usuario con el equipo Red Pitaya. En un inicio, se trabaja con un generador de funciones del laboratorio, conectado directamente con uno de los inputs de la *Red Pitaya* a través de un cable *BNC* y un adaptador *BNC-SMA*.

Contando con la ayuda de un pequeño código *Python* de ejemplo que hay en la página web, se llevan a cabo numerosas pruebas con las distintas señales que puede crear el generador (triangular, cuadrada, seno...). *[9]*

El código está hecho de tal forma que las 16.384 muestras de voltaje obtenidas para cada proceso de muestreo se guardan en una variable de tipo lista. A continuación, en la *Figura 5*, se presenta el mencionado código "de fábrica" junto con un breve comentario de cada línea:

```
"Importación de las librerías necesarias"
import sys
import redpitaya_scpi as scpi

"Conexión del código con la Red Pitaya a través de su dirección IP"
rp_s = scpi.scpi('169.254.106.150')

"Inicialización del proceso de adquisición"
rp_s.tx_txt('ACQ:START')

"Adquisición inmediata"
rp_s.tx_txt('ACQ:TRIG NOW')

"Definición del factor de diezmado"
rp_s.tx_txt('ACQ:DEC 1')

"Bucle del que se sale sólo cuando el trigger está en estado'TD'"
while 1:
    rp_s.tx_txt('ACQ:TRIG:STAT?')
    if rp_s.rx_txt() == 'TD':
        break

"Lectura de los 16.384 datos adquiridos"
rp_s.tx_txt('ACQ:SOUR1:DATA?')

"Almacenamiento de esos datos en una lista de números decimales"
buff_string = rp_s.rx_txt()
buff_string = buff_string.strip('{}\n\r').replace("  ", "").split(',')
buff = list(map(float, buff_string))
```

Figura 5: *Código de adquisición de señal proporcionado por Red Pitaya. Fuente: Propia*

Las muestras almacenadas en la lista (en este caso, llamada *buff*) son fácilmente graficables si se hace uso de la librería *matplotlib.pyplot* de *Python*. Mediante el método *plot*, se crea un simple gráfico de puntos donde el *eje X* representa el número de muestra y el *eje Y* el valor de tensión de la misma. *[10]*

Se añaden las siguientes líneas visibles en la *Figura 6* al código anterior (no sin antes haber importado la librería *matplotlib.pyplot*) con la intención de graficar los datos adquiridos:

```
"Graficar los valores de la lista buff. Eje X = nº muestras, Eje Y = tensión [V]"
plt.plot(buff)

"Título del eje Y"
plt.ylabel('Voltatge')

"Añadir la cuadrícula al gráfico"
plt.grid()

"Mostrar el gráfico
plt.show()
```

Figura 6: *Código útil para graficar la señal adquirida. Fuente: Propia*

Una vez se tiene la posibilidad de graficar los datos obtenidos, básicamente se trata de ir jugando con todos los parámetros de adquisición y de graficar cada vez las muestras

adquiridas para ir entendiendo poco a poco cómo funciona todo.

Por ejemplo, con el generador de funciones se crea una **señal senoidal** de amplitud pico a pico $\boldsymbol{A_{pp} = 2\ V}$ y periodo $\boldsymbol{T = 131{,}072\ \mu s}$. En cuanto a los comandos de adquisición, se usa el modo ***Trigger Now*** y un ***diezmado de 1***. Teniendo en cuenta que el *diezmado 1* implica un tiempo total de muestreo de *131,072 µs* (ver *Tabla 1*), éste coincide con el periodo de la señal generada. Consecuentemente, se espera adquirir 16.384 muestras que conformen exactamente 1 periodo de un seno. En la *Figura 7* se observa que el resultado es satisfactorio.

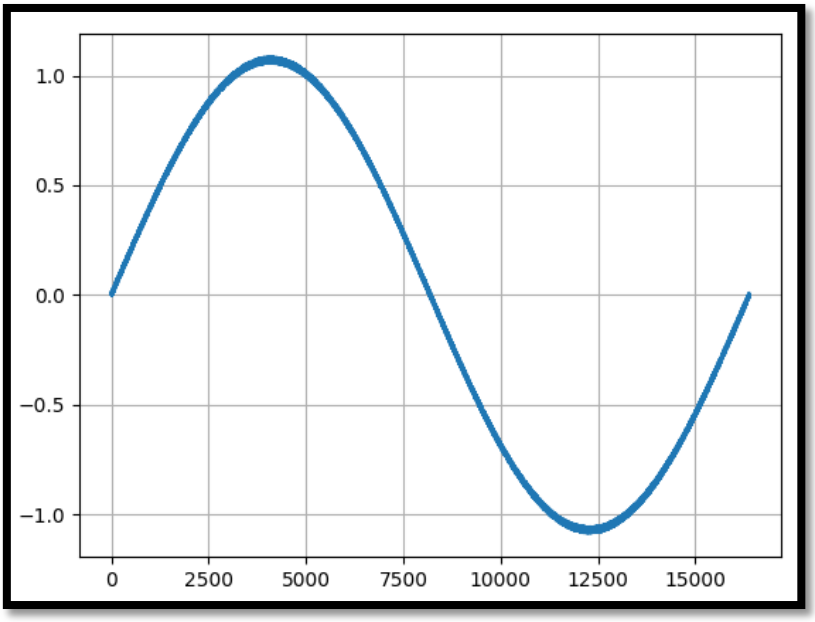


Figura 7: *Adquisición de un seno con diezmado 1. Font: Pròpia. Fuente: Propia*

Si se repite la misma operación, cambiando únicamente el *diezmado de valor 1 a 8*, el tiempo total de muestreo resulta ser 8 veces mayor (*1,049 ms*). Así pues, en esta situación, lo esperado es que las más de 16K muestras conformen ahora 8 periodos de una señal senoidal. En la siguiente *Figura 8* se puede apreciar que se cumplen las previsiones:

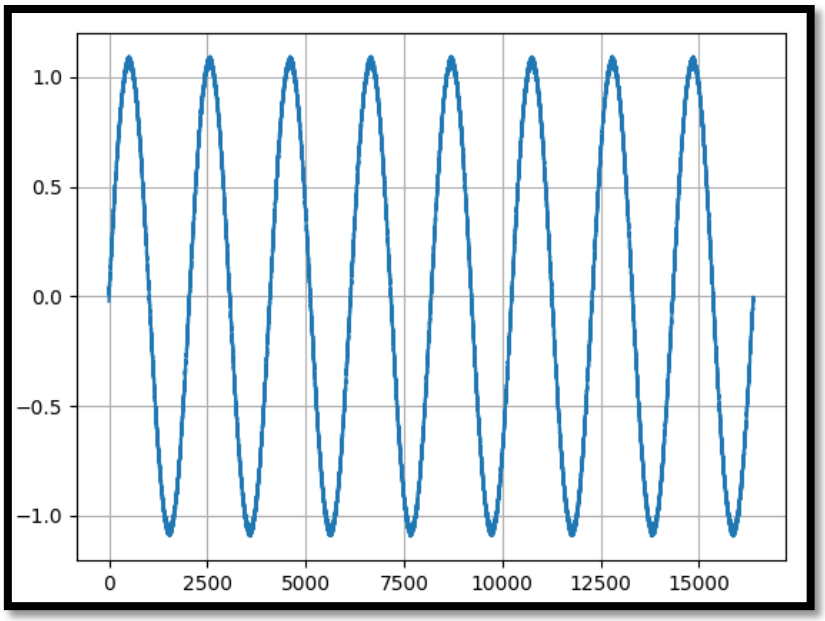


Figura 8: *Muestreo de un seno con diezmado 8. Fuente: Propia*

Este tipo de sencillas pruebas son útiles para detectar pequeños problemas que se pueden manifestar. En las dos figuras anteriores se puede ver claramente uno de ellos: las tensiones de la señal adquirida no concuerdan exactamente con las de la señal definida.

Por alguna razón, la *Pitaya* amplía ligeramente esa señal de entrada. Se podría decir que multiplica cada una de las muestras por una constante (aplica una ganancia > 1). La *Figura 9* es una ampliación de la *Figura 7*. En ella, se aprecia que la amplitud de la señal adquirida (*1,07 V*) es aproximadamente un 7% mayor a lo definido (*1 V*):

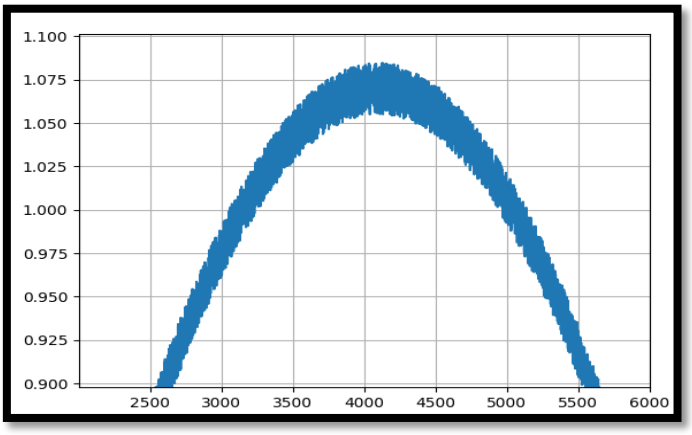


Figura 9: *Muestreo ampliado de un seno con diezmado 1. Fuente: Propia*

Muestreando otro tipo de señales (cuadrada, triangular) se observa que el problema

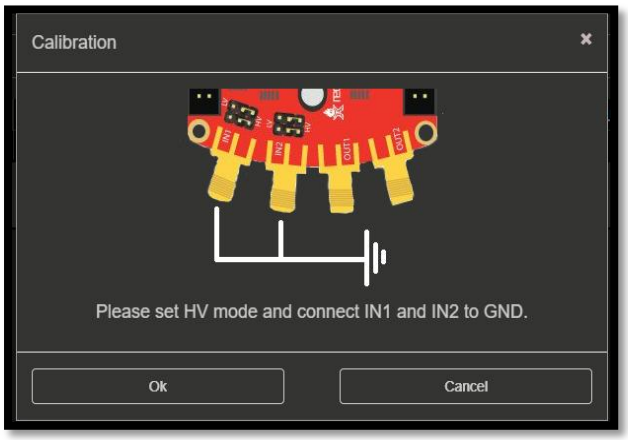
mencionado se reproduce para todas ellas. Después de una búsqueda por la web de *Red Pitaya*, se encuentra un apartado que habla sobre cómo calibrarla. Se intuye, pues, que no viene calibrada de fábrica y por eso comete ese error a la hora de adquirir señales.

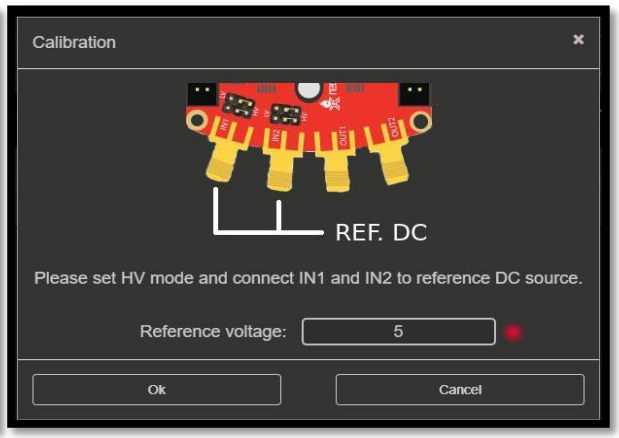
En el próximo subapartado se explica brevemente el proceso de calibrado.

3.2.1.1. Calibrado

El proceso de calibrado es bastante sencillo. Se realiza desde la interfaz de *Red Pitaya*, en el apartado '*System-Calibration'*. Se usa la opción del modo automático, recomendada para usuarios no expertos. De esta forma, la propia aplicación va guiando paso a paso al usuario. Básicamente, hay dos parámetros a calibrar:

- El offset: se define como el nivel de continua que se suma a una señal alterna. Para calibrarlo, *Red Pitaya* indica que se conecten los inputs a una conexión a tierra (*GND*), tal como se puede observar en la *Figura 10.*
- La ganancia: es una magnitud que expresa la relación entre la amplitud de una señal de salida respecto a la señal de entrada. Para calibrarla, se necesita conectar los inputs de la *Red Pitaya* a una fuente de continua de *5 V*, tal como se puede ver en la *Figura 11. [11]*





Figuras 10 y 11: *Calibrado del offset y de la ganancia de la Red Pitaya. Fuente: Interfaz de Red Pitaya.*

Una vez completado el proceso de calibrado, se repite la adquisición de señal senoidal del ejemplo anterior. A continuación, en la *Figura 12*, se puede ver que el muestreo ahora sí resulta satisfactorio: obviando el ruido, tema que se trata más adelante (*Apartado 4.2*), la señal generada tiene igual amplitud que la adquirida por la Pitaya.

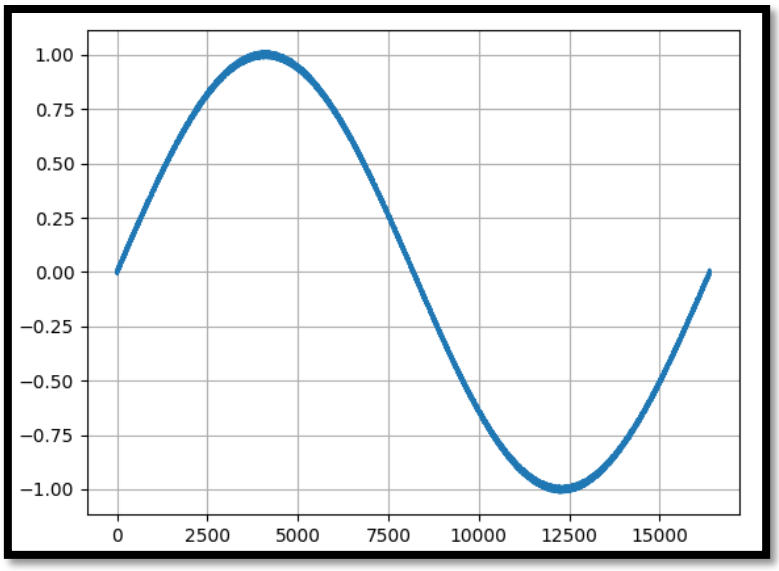


Figura 12: *Adquisición de un seno con diezmado 1 (Red Pitaya calibrada). Fuente: Propia*

3.2.2. Generador de funciones Red Pitaya

Una alternativa bastante útil al generador de funciones es hacer uso del equipo *Red Pitaya* como generador de señal y como receptor de la misma de forma simultánea.

La vertiente generadora de *Red Pitaya* permite crear cualquier señal típica de un generador de funciones. Por lo tanto, no hace falta desplazarse hasta el laboratorio cada vez que se quiere hacer alguna prueba.

Es necesario echar un vistazo a los comandos *SCPI* de generación para poder utilizar la *Red Pitaya* como generador. No se considera indispensable comentarlos detalladamente, ya que no son importantes para este proyecto. *[7]*

Para aplicar este método de trabajo, simplemente se necesita un cable *SMA-SMA*, el cual conecta uno de los *outputs* de la Pitaya (señal generada) con uno de sus propios *inputs* (señal adquirida). En la *Figura 13* se muestra dicha conexión.

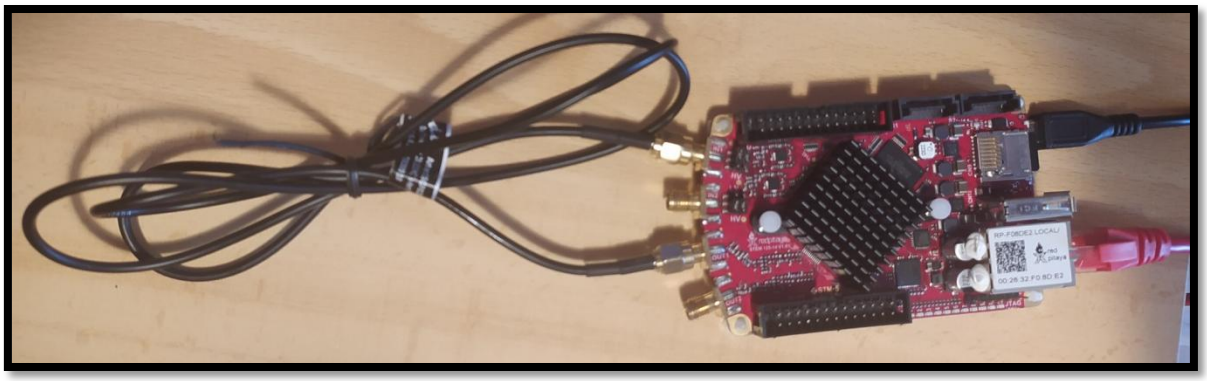


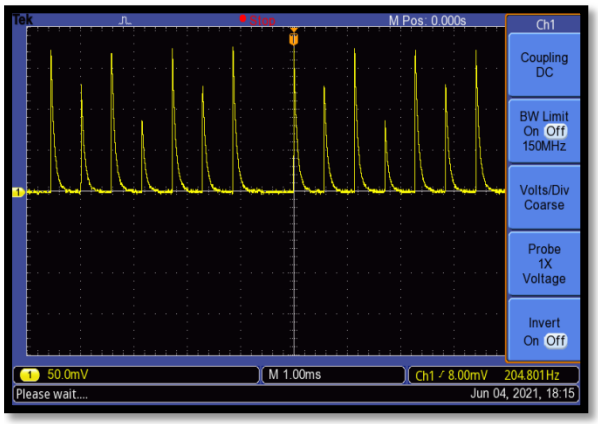
Figura 13: *Red Pitaya funcionando como generador y receptor simultáneamente.*

Fuente: Propia

3.2.3. Generador de tren de pulsos

Una vez ya se comprenden todos los parámetros y comandos de adquisición de señal de la *Red Pitaya* y se es capaz de muestrear adecuadamente distintas señales del generador de funciones, el siguiente paso es intentar repetir el proceso con señales más parecidas a las que se analizaran con el *SCA*.

Para ello, inicialmente se utiliza un generador de tren de pulsos (*Figura 14*) fabricado en la propia *ETSEIB*. La señal que genera este aparato consiste en un conjunto de 8 pulsos de distintas amplitudes que se repiten periódicamente. En la *Figura 15* se muestran dos periodos de esa señal vista en el osciloscopio (el octavo pulso es muy pequeño y no se aprecia bien).



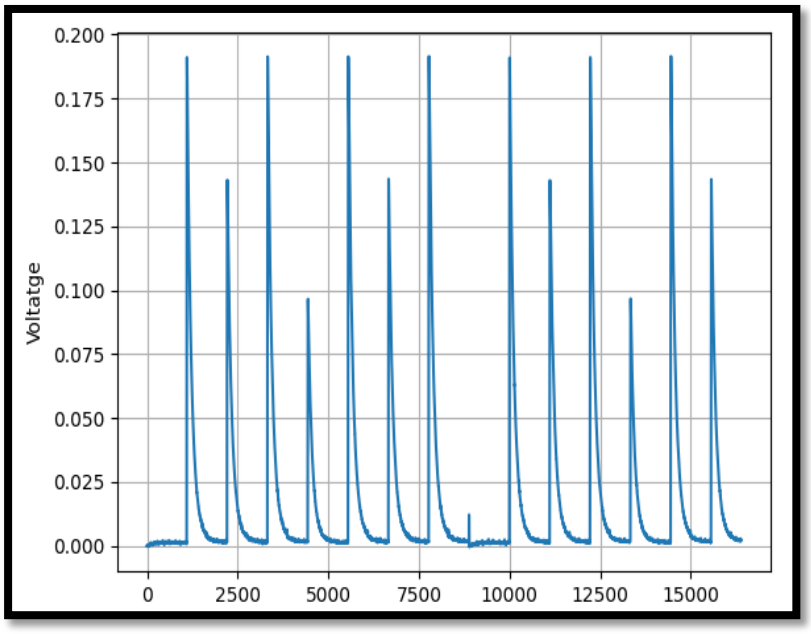
Figuras 14 y 15: *Generador de pulsos y señal que genera vista en el osciloscopio.*

Fuente: Propia

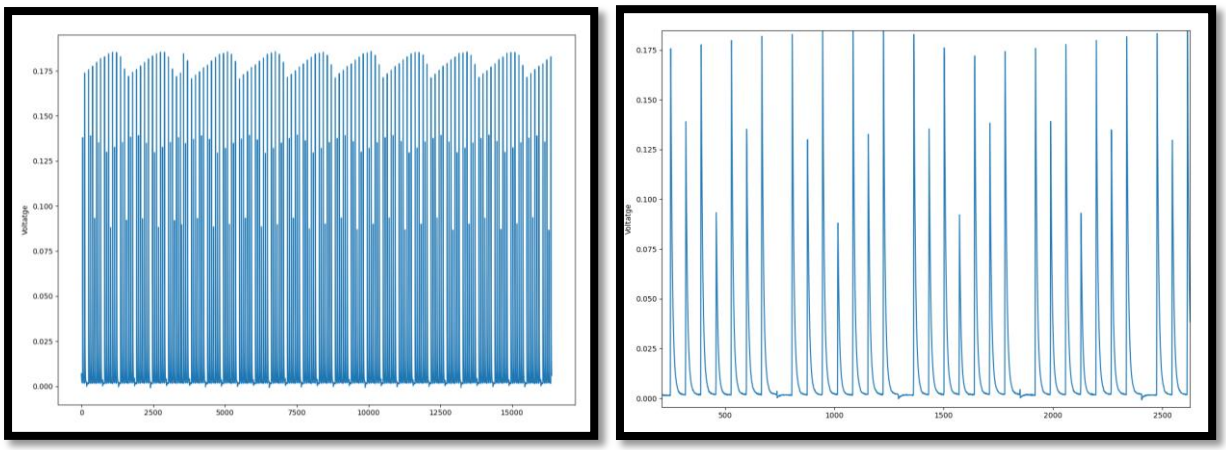
En cuanto a la amplitud de los pulsos de esta señal, simplemente comentar que es inferior a *1 V* en todos ellos. De hecho, el pulso más alto tiene una amplitud de unos *175 mV*. Por lo tanto, a la hora de muestrear con *Red Pitaya*, se colocan los *jumpers* en posición *LV*.

Por otra parte, se observa que el periodo de la señal es de unos *T = 5 ms*. Así pues, si se usan los siguientes diezmados:

- ***Diezmado 1:*** teniendo en cuenta la *Tabla 1*, la frecuencia de muestreo sería demasiado elevada para muestrear esta señal con sólo 16.384 muestras. Únicamente se adquiriría un 2,6% de la señal de todo un periodo. No tiene sentido usar este diezmado.
- ***Diezmado 8***: a esta frecuencia de muestreo, se espera adquirir aproximadamente 1/5 parte de la señal de todo un periodo, es decir, *1 ms* de señal. Por lo tanto, este diezmado también es insuficiente para muestrear un periodo entero.
- ***Diezmado 64:*** en este caso, el proceso de muestreo dura unos *8 ms*, así que en principio la Pitaya debe ser capaz de muestrear aproximadamente 1,6 periodos de señal. La *Figura 16* demuestra que, efectivamente, es así.
- ***Diezmado 1024***: se adquieren unos 27 periodos de señal. Tal como se puede ver en la *Figura 17*, la señal queda demasiado compactada como para poder analizarla bien. La *Figura 18* enseña ese mismo muestreo ampliado entre las muestras 0 y 2500. Parece que las alturas de los pulsos no acaban de concordar entre periodo y periodo. Se concluye que la frecuencia de muestreo es demasiado baja (con este diezmado, la escasa cantidad de muestras que conforman los pulsos provoca imprecisiones en el proceso de adquisición).



Figuras 16: *Señal del generador de tren de pulsos. Diezmado 64. Fuente: Propia*



Figuras 17 y 18: *Señal del generador de tren de pulsos original y ampliada. Diezmado 1024. Eje x: nº muestra / Eje y: tensión [V]. Fuente: Propia*

Para terminar con el generador de tren de pulsos, resulta interesante comparar un pulso de señal muestreado con un pulso visto en el osciloscopio. Para ello, se sigue el siguiente proceso:

1. Se extraen mediante un *USB* los datos de una captura del osciloscopio (automáticamente se genera un archivo *.csv*).
2. Paralelamente, se realiza un muestreo con Red Pitaya y se guardan las tensiones de las muestras obtenidas en un archivo *.csv*.
3. Se abre una hoja de Excel y se extraen los datos de ambos archivos *.csv*.
4. Se transforma el eje X de los datos de Pitaya a escala temporal.
5. Se selecciona la sección de señal que se desea graficar (en este caso, se elige el pulso de mayor amplitud).
6. Se superponen los datos de *Red Pitaya* y los del osciloscopio en una misma gráfica.

Los resultados son muy parecidos, pero los obtenidos con la *Pitaya* son más precisos. Esto es debido a que la frecuencia de muestreo definida (en este caso se usa *diezmado 8*) es más elevada que la del osciloscopio. La superposición de ambas señales se muestra en la *Figura 19*:

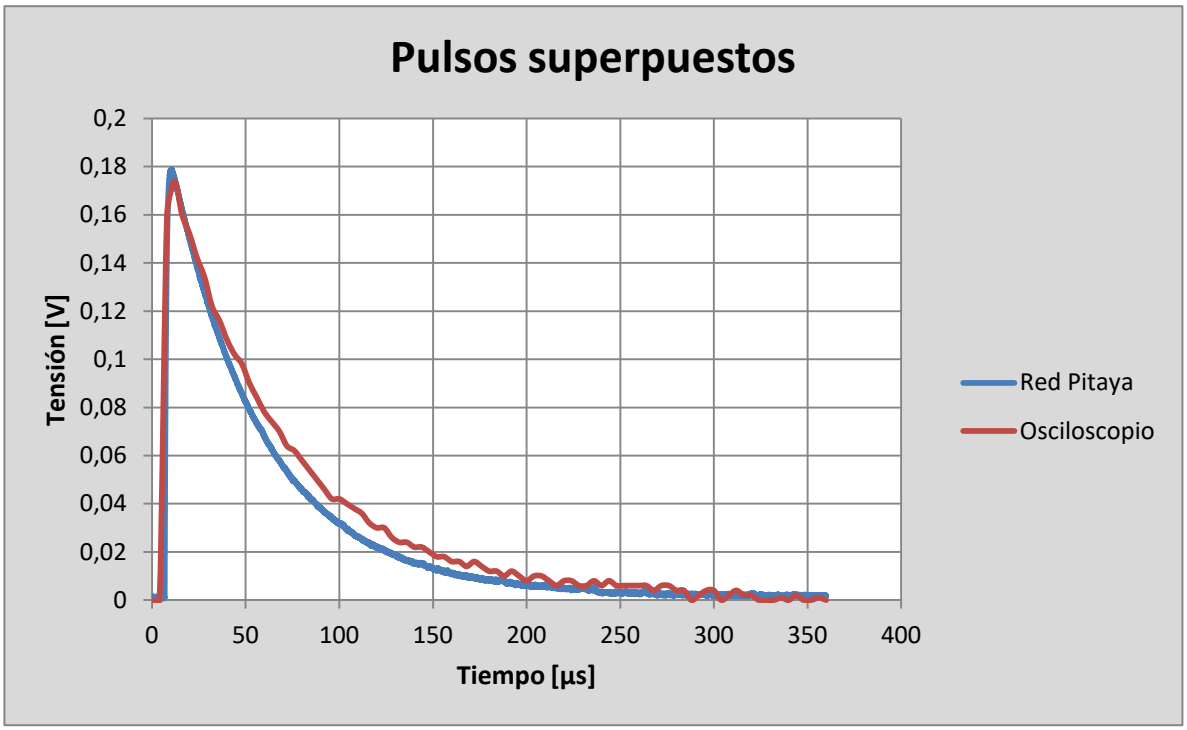


Figura 19: *Superposición del pulso más alto del generador (Pitaya vs Osciloscopio).*

Fuente: Propia

3.2.4. Muestreo de señales amplificadas

En esta fase del proyecto, llega el momento de intentar muestrear señales generadas por isótopos radiactivos. Los radioisótopos tienen un núcleo atómico inestable (por el balance entre neutrones y protones) y emiten partículas con una determinada energía cinética cuando se desintegran. La energía se libera, principalmente, en forma de rayos alfa (núcleos de helio), beta (electrones o positrones) o gamma (energía electromagnética). *[12]*

La idea es detectar esa energía emitida por la fuente de radiación y, a través de una serie de componentes, acabar transformándola en una señal de tensión que pueda ser muestreada con la *Pitaya*. Para ello, se trabaja con el siguiente montaje (*Figura 20*):

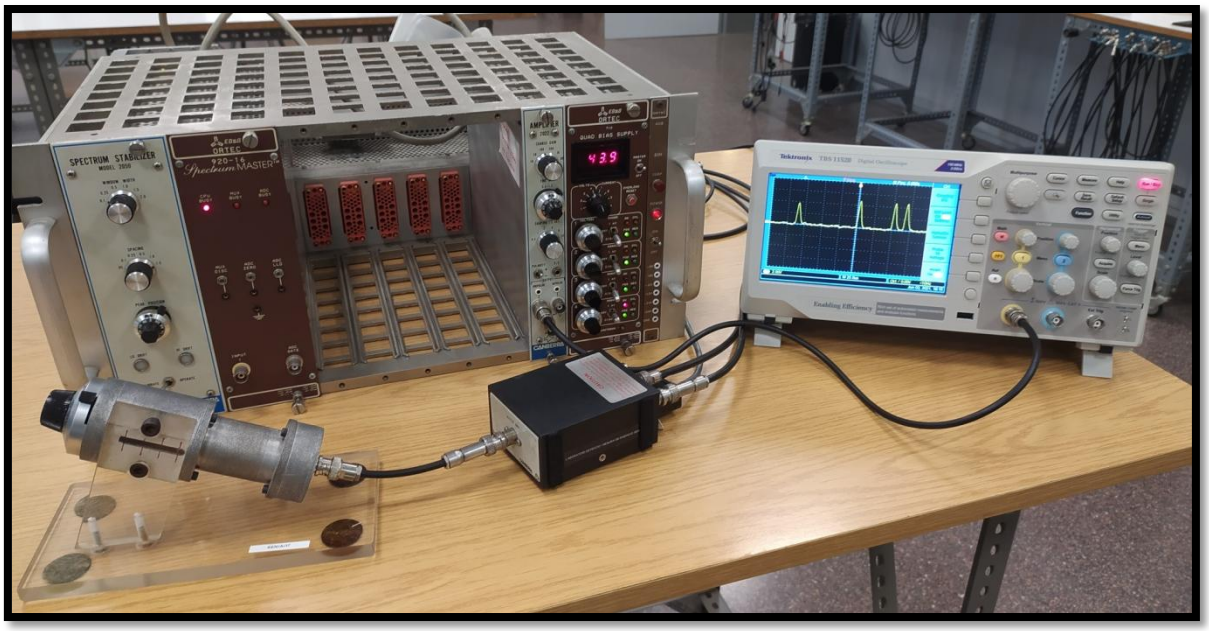


Figura 20: *Montaje para muestrear pulsos amplificados. Fuente: Propia*

Los componentes que configuran este montaje son los siguientes:

a. **Fuente de *Americio-241*:** En el *Apartado 5.3.2* se explican los detalles acerca de esta fuente y se presenta su espectro radiactivo.
   De momento, lo importante es comentar que la desintegración de los núcleos es estocástica. Por lo tanto, no hay manera de prever el tiempo que transcurre entre la emisión de una partícula y la siguiente.

b. **Detector de Barrera de Superficie de Silicio (*SSB*):** es el encargado de captar la radiación emitida por la fuente. Concretamente, este tipo de detector únicamente distingue las partículas *alfa* y *beta* de la fuente, generando una carga (hueco de electrón) a su salida que se convierte en una corriente merced al campo eléctrico que se establece en su región de empobrecimiento. En este caso, El ^{241}Am no es un emisor *beta*, con lo cual todas las cargas generadas son debidas a partículas *alfa*. [13]

c. **Preamplificador:** es un circuito de acondicionamiento de la señal antes de su posterior tratamiento o conformado. Concretamente, el preamplificador de la marca *Canberra* está diseñado para un rendimiento óptimo con *detectores de Barrera de Superficie de Silicio (SSB)*. Funciona como convertidor de carga a voltaje, es decir,

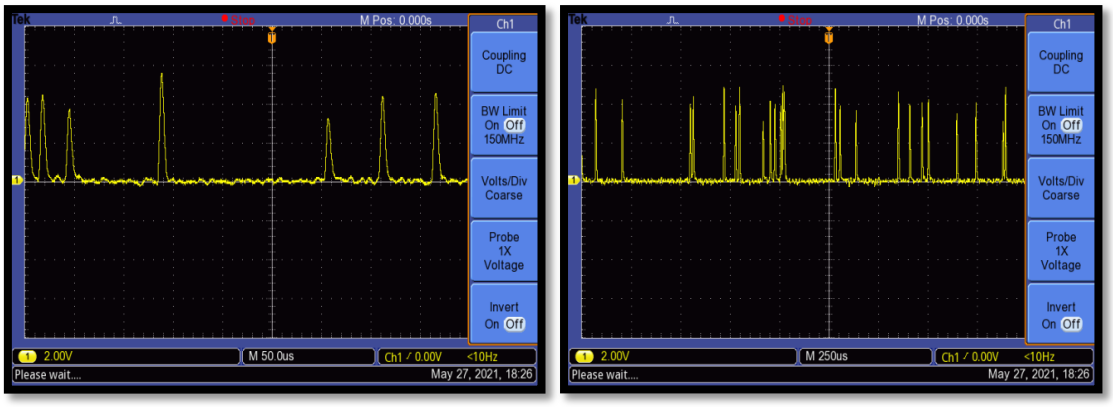
la salida proporciona una tensión directamente proporcional a la carga recogida (a razón de unos *0,45 V* por *pC*). [14]

d. Amplificador *de pulsos*: se usa para hacer más clara la señal de salida del preamplificador. Es muy útil para solucionar el *pile-up* (apilamiento) de la señal, a la vez que la convierte en pulsos *cuasi-Gaussianos* y, por lo tanto, con mayor relación señal-ruido.

e. Osciloscopio: su función es graficar la señal que proviene del amplificador. Si se establecen las escalas de tiempo (*eje X*) y de tensión (*eje Y*) adecuadas se puede observar de forma clara que todos esos pulsos *cuasi-Gaussianos* tienen amplitud similar.

Este hecho es lógico, ya que todos corresponden al mismo tipo de radiación (*alfa*) y, por lo tanto, a un nivel de energía y de tensión determinado. Además, se detecta que la anchura de la señal es bastante pequeña (o lo que es lo mismo, los pulsos son rápidos), de unos 5 µs.

A continuación, en las *Figuras 21 y 22*, se presenta la señal descrita mediante un par de capturas del osciloscopio:



Figuras 21 y 22: *Visualización de la señal de salida del amplificador vía osciloscopio. Fuente: Propia*

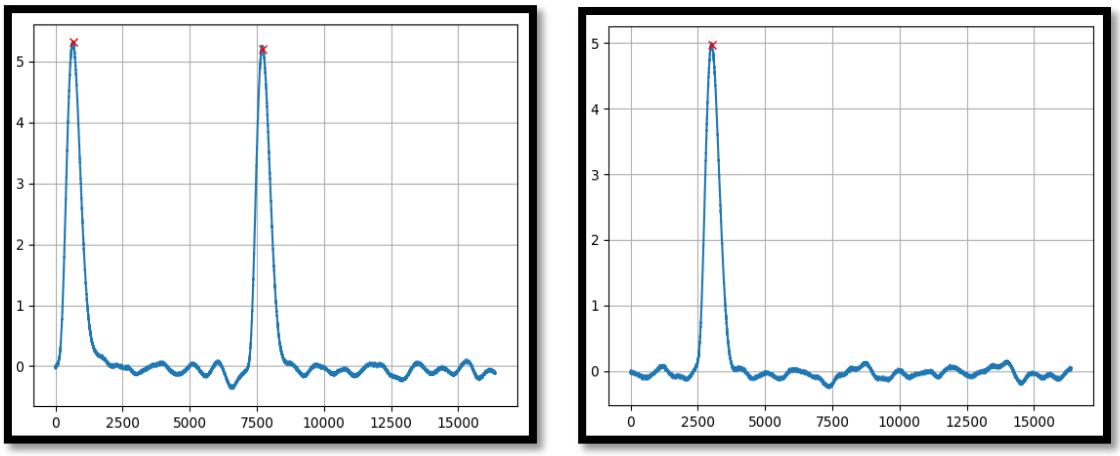
Para muestrear este tipo de señal con el equipo Red Pitaya hay que tener en cuenta dos factores:

- No es posible usar factores de diezmado superiores a 64. Con un factor de 1024, el tiempo de adquisición entre muestra y muestra sería de alrededor de *8 µs* (*Tabla*

1), con lo cual no sería posible captar los pulsos (que tienen una anchura de unos *5 μs*).

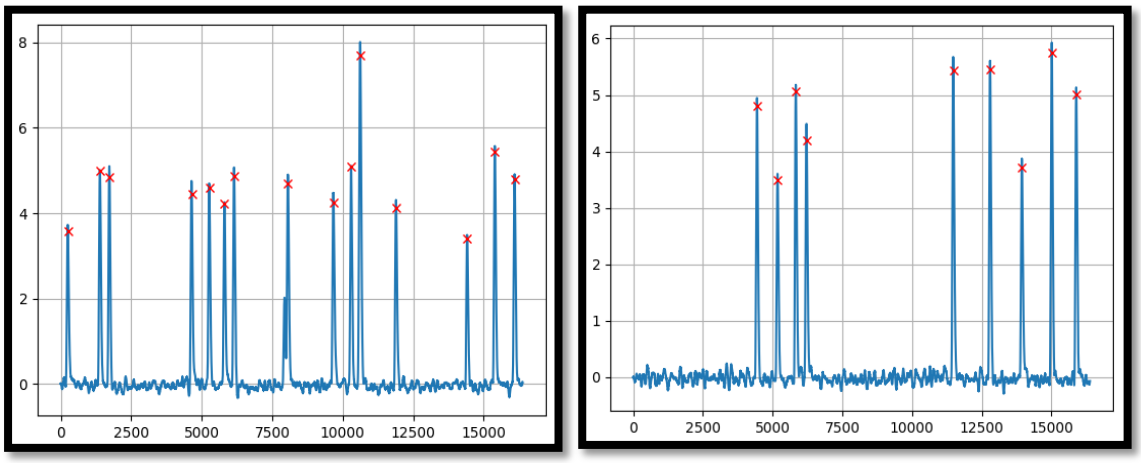
- Nunca se van a obtener dos muestreos iguales. Esto es debido a la ya comentada aleatoria emisión de partículas *alfa* por parte de la fuente.

Por ejemplo, se puede dar el caso de captar 2 pulsos de señal en un determinado muestreo y 1 solo pulso en otro (aunque se usen exactamente las mismas condiciones de adquisición). A continuación se muestran dos imágenes (*Figuras 23 y 24*) para ilustrar esta situación. Se usa *diezmado 1*:



Figuras 23 y 24: *Muestreos de señal generada por partículas alfa (diezmado 1). Fuente: Propia*

Con *diezmado 8*, lógicamente aparecen más pulsos ya que el tiempo de muestreo es 8 veces mayor. Debajo, se presentan dos muestreos en estas condiciones. Mientras que en uno de ellos aparecen 15 pulsos (*Figura 25*), en el otro se obtienen 9 pulsos, casualmente separados en dos bloques (*Figura 26*). Esta variedad concuerda con la aleatoriedad de las desintegraciones.



Figuras 25 y 26: *Muestreos de señal generada por partículas alfa (diezmado 8). Fuente: Propia*

Una vez se es capaz de muestrear satisfactoriamente este tipo de pulsos *cuasi-Gaussianos* que provienen del amplificador, ya se puede pasar al siguiente gran bloque del proyecto: la creación del analizador monocanal con el equipo *Red Pitaya*.

4. BLOQUE II: ANALIZADOR MONOCANAL (SCA)

A nivel conceptual, ya se ha explicado lo que es un *SCA* y sus funciones (*Apartado 2.2*). El objetivo, pues, no es más que lograr crear un código con el lenguaje de programación *Python* que consiga emular la función de un analizador monocanal comercial.

4.1. Algoritmo propio

Cabe recordar que el pequeño código de ejemplo de adquisición de señal disponible en la página web de *Red Pitaya* guarda las 16.384 muestras de tensión obtenidas durante el muestreo en una variable de tipo lista.

Teniendo esto en cuenta, inicialmente, se piensa en crear un algoritmo que sea capaz de encontrar los máximos locales de una lista. Una vez encontrados, la idea es analizar cada máximo uno por uno y ver si su tensión cae dentro de una ventana predeterminada. En caso afirmativo, se suma una unidad a un contador. El código inicial queda de la siguiente forma (*Figura 27*):

```
"Inicialización del contador del SCA"
count = 0

"Valor del límite inferior del SCA"
lim_inf = ...

"Valor del límite superior del SCA"
lim_sup = ...

"Recorrido posición a posición de la lista buff"
for i in range(len(buff)-1):

    "Si el elemento de la posición 'i' se encuentra dentro de los límites del SCA..."
    if lim_inf < buff[i] < lim_sup:

        "Si, además, ese elemento es mayor al anterior y al posterior al mismo tiempo (es decir, es un máximo local)..."
        if buff[i-1] < buff[i] > buff[i+1]:

            "Se actualiza el contador del SCA ya que se trata de un máximo local dentro de la ventana"
            count = count + 1

    "Muestra el valor final del contador del SCA"
    print(count)
```

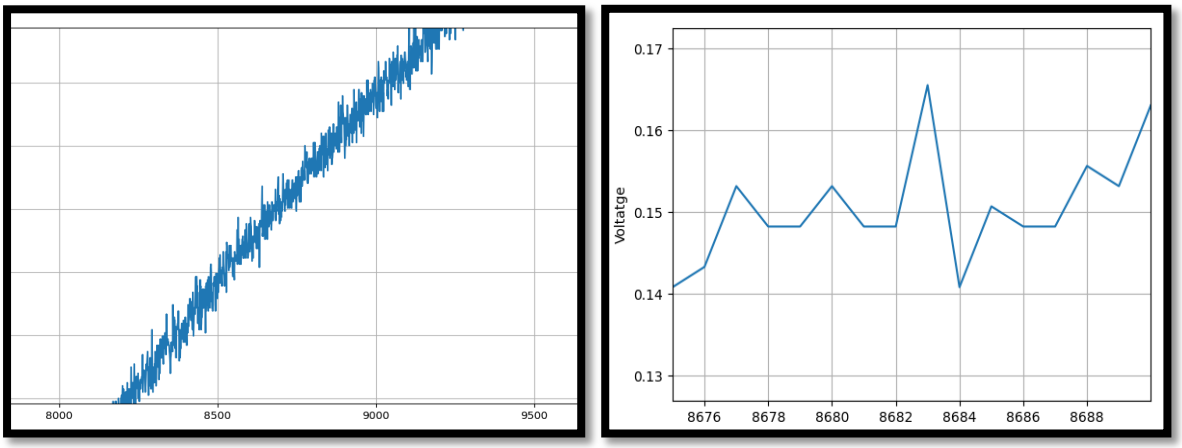
Figura 27: *Código inicial para contar máximos de una lista. Fuente: Propia*

Durante las primeras pruebas con este código ya se detecta un gran problema: cuenta muchos más máximos de los que realmente hay en la señal muestreada.

El problema radica en el segundo condicional: la línea de código '**if buff [i – 1] < buff [i] > buff [i + 1]:**' únicamente compara la tensión del elemento actual de la lista (**buff [i]**) con la de sus vecinos (**buff [i – 1]** y **buff [i + 1]**). Esta condición no resulta suficiente

para determinar la presencia de un máximo local, ya que hay que tener en cuenta el ruido de la señal.

En las *Figuras 28 y 29* se puede ver claramente el efecto del ruido. Si bien es cierto que la tendencia de la señal en ese conjunto de muestras es de crecimiento, si se analiza con una escala de tiempo menor (muestra a muestra), se puede apreciar que no se trata de un crecimiento continuo, sino que se producen pequeñas fluctuaciones.



Figuras 28 y 29: *Señal con ruido y Señal con ruido ampliada. Fuente: Propia*

Por ejemplo, si se utiliza el algoritmo anterior para un conjunto de muestras como el de la *Figura 29*, éste considera que muchas de las muestras (nº 8678, nº 8680, nº 8683 entre otras), corresponden a máximos locales por el simple hecho de que el valor de su tensión es superior a la de sus muestras vecinas.

En realidad, es obvio que ninguna de esas muestras es un pico de tensión. Tal como se puede ver en la *Figura 28*, la señal sigue con su tendencia de crecimiento hasta más allá de la muestra nº 9000.

Por todo lo dicho, de momento se descarta que el algoritmo creado por si solo sea suficiente para la implementación del *SCA*. En los próximos apartados se buscan alternativas.

4.2. Suavizado de señal

Con el objetivo de eliminar completamente el ruido, se piensa en aplicar un método de suavizado de la señal. Si se consiguiera este propósito, la señal dejaría de fluctuar entre

muestras y, en consecuencia, se podría reutilizar el algoritmo creado en el apartado anterior obteniendo resultados satisfactorios.

Para llevar a cabo el suavizado, se realiza una búsqueda por internet. Los que se consideran más implementables para el proyecto son los siguientes: *[15]*

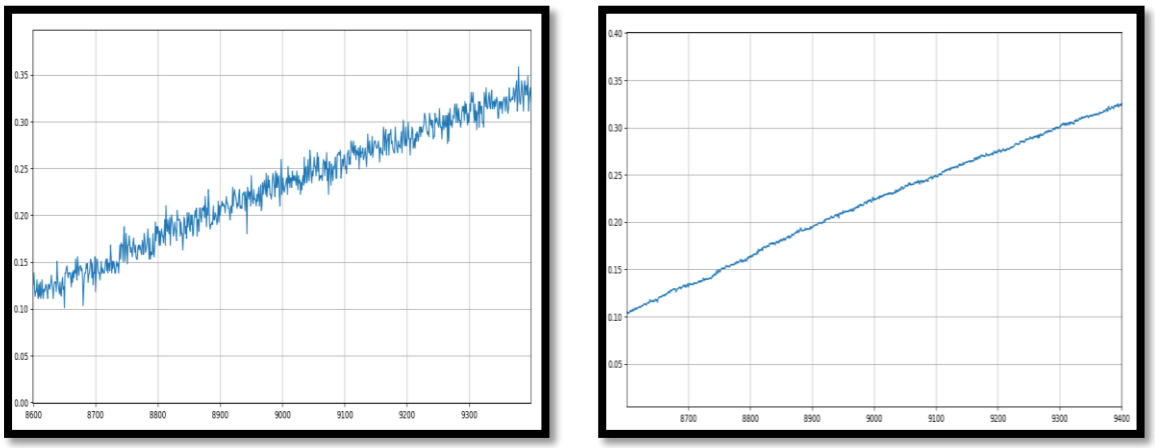
- Filtro de media móvil.
- Filtro de Savitzky-Golay.

4.2.1. Filtro de media móvil

Consiste en sustituir el valor de cada muestra de señal por la media de ella misma y un número *x* de muestras anteriores, siendo *x* un valor a determinar por el usuario. *[16]*

Ejemplo

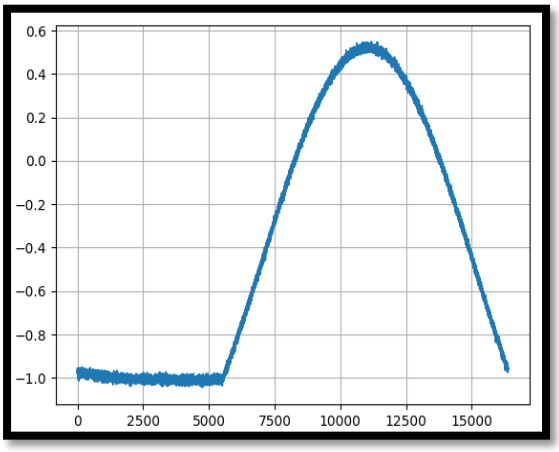
Para una señal cualquiera (*Figura 30*) se aplica un filtro de media móvil de $x = 8$ y el ruido se ve claramente reducido (*Figura 31*).

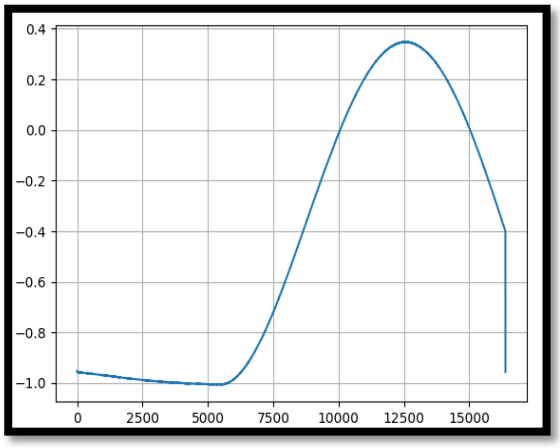


***Figuras 30 y 31**: Señal original y señal suavizada con x = 8. Fuente: Propia*

Si se sigue incrementando poco a poco el número de muestras, el ruido continúa menguando. Sin embargo, para un conjunto de muestras demasiado elevado, aparece otro problema: se puede llegar a manipular totalmente la señal.

En las *Figuras 32* y *33* se puede observar esta situación para una señal ruidosa cualquiera, con $x = 60$. La aplicación del filtro reduce el ruido totalmente, pero por otro lado tanto la forma como la amplitud de la señal se ven modificadas.





Figuras 32 y 33: *Señal original y Señal suavizada con x=60. Fuente: Propia*

En resumen, se pueden sacar las siguientes conclusiones respecto al filtro de media móvil:

Ventajas

- No es un filtro complejo a nivel conceptual.
- Se puede aplicar en pocas e intuitivas líneas de código.

Inconvenientes

- Puede modificar la señal y, con ello, la altura real de los picos.
- El valor de X debe ser redefinido cada vez que se cambian las condiciones de muestreo, lo cual no es muy práctico.

Los inconvenientes se consideran demasiado importantes. Consecuentemente, se descarta la aplicación de este filtro para el *SCA*.

4.2.2. Filtro de Savitzky-Golay

A nivel conceptual, este filtro es un poco más complejo que el anterior. Consiste en reemplazar cada valor de la lista por uno nuevo obtenido mediante un ajuste polinomial a *2n + 1* muestras vecinas (incluida la muestra a suavizar), siendo *n* igual o mayor que el orden del polinomio. Esta descripción se puede visualizar en la *Figura 34*. *[17]*

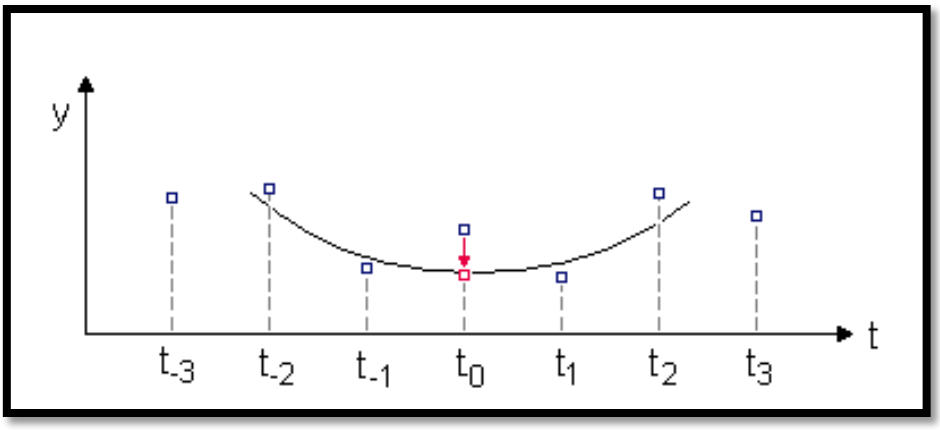


Figura 34: *Ajuste polinomial de grado 2 de la muestra t_0 en función de las muestras t_{-3}-t_3.* *[18]*

A priori, la creación del algoritmo debería resultar más costosa que en el caso anterior. Por suerte, buscando por la web *scipy.org*, muy útil para encontrar distintos algoritmos ya existentes en lenguaje *Python*, se encuentra que el filtro de *Savitzky-Golay* ya está hecho (concretamente, la función se llama ***sicpy.signal.savgol_filter***). Así pues, no es necesario crearlo por cuenta propia. *[19]*

Para implementarlo de forma adecuada, es necesario descargar la librería *scipy.signal* de *Python*. Luego, se debe incorporar la función ***savfol_filter*** al código de adquisición de señal presentado en el *Apartado 4.1* (*Figura 27*) para aplicar el filtro a la lista *buff*.

Los parámetros que conforman el filtro de Savitzky-Golay son los siguientes:

- *x*: simplemente se trata de la lista de valores que se quiere filtrar.
- *window_length*: es el número de muestras que se desea tomar a la hora de realizar el ajuste polinomial. Se define como *2n + 1*, siendo *n* el número de muestras vecinas a cada lado. Por ejemplo, si se desea recalcular el valor de una muestra teniendo en cuenta sus 3 muestras vecinas (*n = 3*), se debe definir *window_length* = 7.
- *Polyorder:* es el orden del polinomio utilizado para ajustar las muestras. Lógicamente, su valor debe ser menor que *window_length*. En principio, se usa siempre *polyorder = 2*.

Una vez explicados los conceptos, se procede a probar el filtro para una señal ruidosa como la del apartado anterior.

Se usan 3 *window_length* distintas (*7, 25 y 101*) y se aprecia que cuantas más muestras

se cojen mejor es el ajuste polinomial. Obviamente esto se cumple hasta un cierto límite, no tendría ningún sentido definir una *window_length* de tamaño desorbitado. El resultado se puede ver en la *Figura 35*:

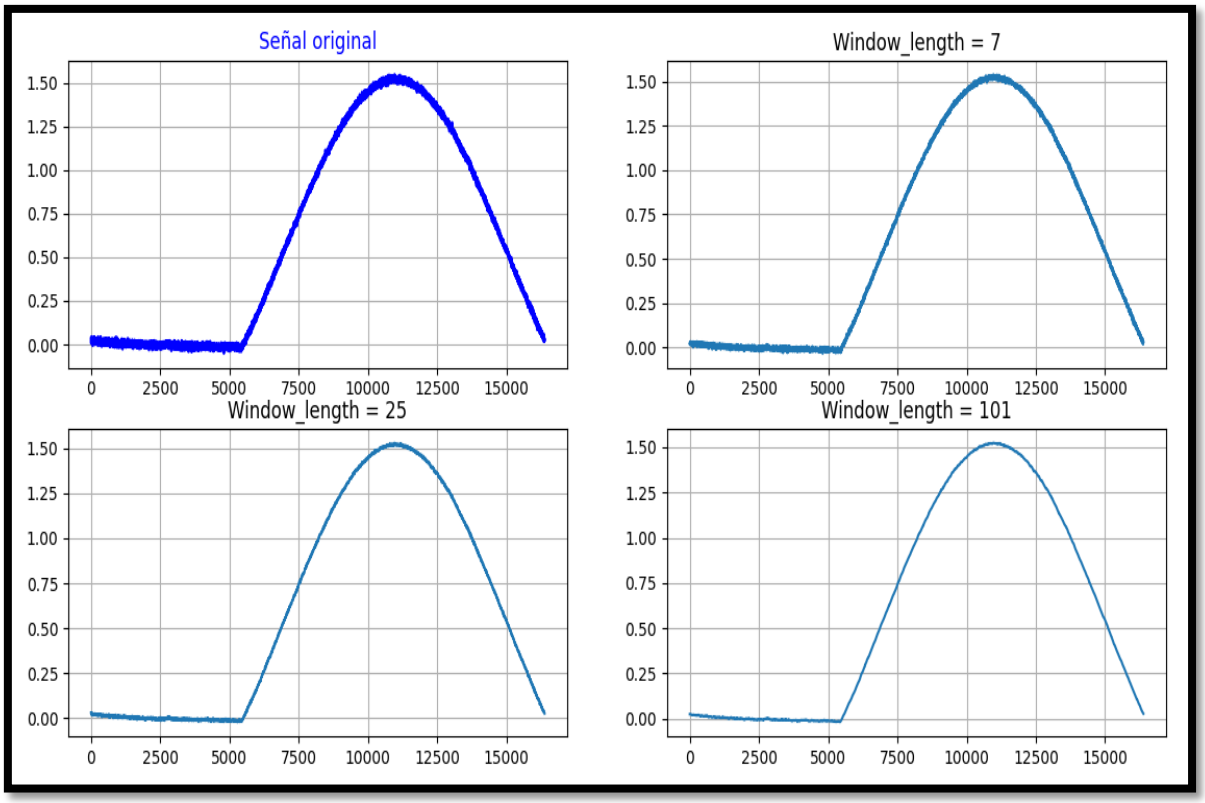


Figura 35: *Filtro Savitzki-Golay: visualización del ruido según el parámetro window_length. Fuente: Propia*

En resumen, los pros y contras más evidentes del filtro *Savitzky-Golay* son los siguientes:

Ventajas

- Ya existe una función en lenguaje *Python* para aplicar este filtro (no es necesario crear el algoritmo).
- El resultado del suavizado es muy satisfactorio ya que la señal original apenas se ve modificada por el ajuste polinomial.

Inconvenientes

- El parámetro *window_length* óptimo no es fácil de definir. Debe ser modificado a base de "prueba y error" hasta que el resultado es el deseado.
- No se acaba de adaptar perfectamente a la forma de ciertos picos.

Se puede afirmar sin ningún tipo de duda que el filtro de *Savitzky-Golay* es más

adecuado que el de media móvil. Sin embargo, no es fácil erradicar por completo el ruido de la señal y, por lo tanto, aunque se aplicara el filtro, el algoritmo de detección de máximos creado todavía detectaría algunos máximos locales "falsos".

4.3. Función find_peaks_cwt

Pese a que los filtros analizados cumplen con el propósito de reducir el ruido de la señal (sobre todo el de Savitzky-Golay), no acaban de solucionar el problema del conteo de máximos. Finalmente, se decide aparcar la idea de suavizar la señal para poder hacer uso del algoritmo creado en el *Apartado 4.1*.

Después de un proceso de búsqueda de alternativas, se encuentra una opción que puede resultar muy interesante en la ya mencionada web *scipy.org*: la función ***find_peaks_cwt***, la cual forma parte de la librería *scipy.signal*. ***[20]***

Precisamente, el propósito de esta función coincide con el asunto que está resultando complicado resolver: encontrar los máximos locales de una lista dada. Concretamente, en **[20]** definen la función ***find_peaks_cwt*** como *"ideal para encontrar picos agudos entre datos ruidosos"*. Parece ser que concuerda perfectamente con lo que se busca.

Sin embargo, su aplicación no es tan sencilla. Los dos parámetros más importantes a proporcionar a la función son los siguientes:

- *Vector*: no es más que la lista de la cual queremos sacar los máximos locales.
- *Width*: realmente, es un parámetro complejo de definir adecuadamente, ya que no está explicado de forma detallada en la *web*.
  Se podría decir que el parámetro *width* es la anchura esperada (en número de muestras) de los pulsos a una altura concreta. En este proyecto no se aplica la fórmula de manera estricta en ningún caso (ver Apartado *4.3.1*). *[21]*

La función ***find_peaks_cwt*** aplica un suavizado del parámetro *Vector* para cada anchura del parámetro *width* y, a continuación, busca si hay máximos locales. Otro aspecto a comentar acerca de esta función es la forma que tiene de comunicar al usuario los máximos encontrados: en lugar de proporcionar directamente los valores de esos máximos, lo que hace es crear una lista en la cual **almacena todas las posiciones de los máximos** que encuentra en el parámetro *Vector* (en este proyecto, la lista llamada *buff*). [20]

Por ejemplo, si se pasa la lista de números enteros [1,4,0,2,1] como parámetro *Vector,* la respuesta que se obtiene es otra lista con el siguiente contenido: [1,3], ya que los máximos locales de la lista se encuentran en las posiciones 1 y 3 de la misma (en lenguaje *Python*, el primer elemento de una lista corresponde a la posición 0).

Una vez hechas estas aclaraciones, la función ***find_peaks_cwt*** se incorpora al código, sustituyendo al condicional del algoritmo propio, el cual no proporcionaba los resultados deseados. Se hace alguna que otra modificación más para adaptar el algoritmo a la nueva función y el resultado final es el siguiente (*Figuras 36 y 37*):

```
"Importación de la librería matplotlib para poder graficar"
import matplotlib.pyplot as plt

"Importación de la función find_peaks_cwt"
import scypi.pyplot.find_peaks_cwt as peaks

"Definición de la anchura de los pulsos"
width = [...]

"Las posiciones de los máximos locales se guardan en una variable tipo lista en formato string"
ind_max = peaks(buff, width)

"Creación de 3 listas vacías"
x_max = []
y_max = []
p_max = []

"Inicialización del contador del SCA"
count = 0

"Recorrido índice a índice de la lista 'ind_max'"
for j in range(len(ind_max)):

    "Transformación de cada elemento de la lista 'ind_max' de string a entero"
    ind_max[j] = int(ind_max[j])

    "Si la tensión se encuentra dentro de los límites del SCA..."
    if a <= buff[ind_max[j]] <= b:
```

Figura 36: *Código SCA (Parte I). Fuente: Propia*

```
    "Si la tensión se encuentra dentro de los límites del SCA..."
    if a <= buff[ind_max[j]] <= b:

        "Se actualiza el contador ya que se trata de un máximo local dentro de la ventana"
        count = count + 1

        "Se añade la posición del máximo a la lista x_max"
        x_max.append(ind_max[j])

        "Se añade la tensión del máximo a la lista y_max"
        y_max.append(buff[ind_max[j]])

        "Se añaden tanto la posición como la tensión del máximo a la lista p_max"
        p_max.append((ind_max[j],buff[ind_max[j]]))

"Se muestran los valores finales (tras el recorrido) de 'count' y de 'p_max'"
print(p_max)
print(count)

"Se grafican las 16.384 muestras de señal de la lista buff"
plt.plot(buff)
plt.ylabel('Voltatge')
plt.grid()

"Se marcan los máximos con cruces de color rojo"
plt.plot(x_max, y_max, "xr")

plt.show()
```

Figura 37: *Código SCA (Parte II). Fuente: Propia*

Para determinar la eficacia de la función ***find_peaks_cwt***, se hace uso de una de las señales ya muestreada anteriormente: la correspondiente al *Apartado 3.2.4*.

Con el objetivo de captar unos cuantos máximos, se decide realizar un muestreo con *diezmado 8* (con *diezmado 1* lo normal es que se vean, con suerte, un par de pulsos). De momento, la ventana de tensión que se establece es muy grande (*lim_inf = 1 V y lim_sup = 10 V*), ya que lo que interesa en estos momentos es simplemente ver si el algoritmo es capaz de encontrar todos los picos de señal, tengan la altura que tengan.

La *Figura 38* muestra que las cruces rojas no acaban de marcar la posición de los máximos de forma precisa. Esto es debido a una mala definición del parámetro *width*, no es difícil de corregir y se explica detalladamente cómo hacerlo en próximamente (*subapartado 4.3.1)*. Aún así, el resultado es satisfactorio en cuanto a que el algoritmo detecta los 9 máximos que se pueden apreciar visualmente:

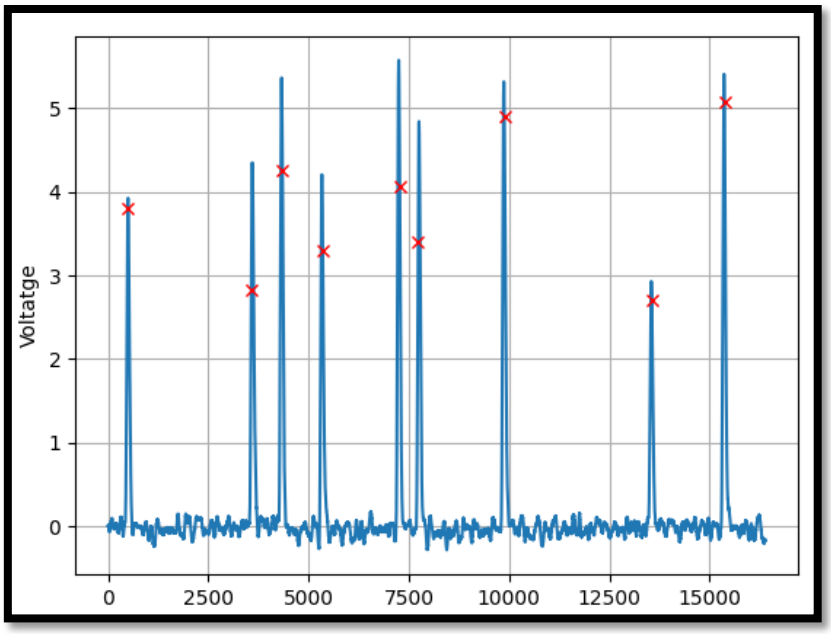


Figura 38: *Comprobación del resultado de la función find_peaks_cwt. Fuente: Propia*

Vistos los resultados, se concluye que la función ***find_peaks_cwt*** es la alternativa más adecuada para implementar el analizador mono-canal. Por lo tanto, el código mostrado en la página anterior (*Figuras 36 y 37*) ya está cerca de ser el definitivo, aunque todavía falta un detalle muy importante: la implementación del bucle de adquisición.

4.3.1. Definición del parámetro width

Hay que tener en cuenta que, cuando se cambia el diezmado, también es necesario modificar el parámetro *width* de la función ***find_peaks_cwt*** si se desean obtener los máximos de forma precisa. Esto es debido al hecho de que la anchura de los pulsos (en número de muestras) disminuye a medida que aumenta el diezmado.

Es cierto que hay un procedimiento determinado para encontrar el parámetro *width* ideal, pero es un poco confuso y laborioso. Tras experimentar lo suficiente, se concluye que es más práctico aplicar el método de “prueba y error”. *[22]*

Tras unas cuantas pruebas, se da por satisfactorio un parámetro *width = [200]* en caso de usar *diezmado 1*. Aplicando proporcionalidad, se sospecha que los valores de *width* presentados en la *Tabla 3* pueden ser válidos para diezmados superiores:

Diezmado	1	8	64
width	[200]	[25]	[3]

Tabla 3: *Parámetro width en función del factor de diezmado. Fuente: Propia*

En apartados posteriores, se usan exactamente los valores de la tabla y el resultado es satisfactorio.

4.4. Bucle de adquisición de señal

Lo que se ha conseguido hasta ahora está bien, pero no sirve de nada si no se puede ampliar el tiempo de muestreo. ¿Qué conclusiones se pueden sacar contando picos de señal durante *130 µs* (*diezmado 1*), *1 ms* (*diezmado 8*) o, como máximo, *8 ms* (*diezmado 64*)? Pocas o ninguna.

Para ampliar el tiempo de muestreo a través del código, la única solución posible es aplicar un bucle (*for).* La idea es que, una vez la *Red Pitaya* adquiera las 16.384 muestras de la primera iteración, en lugar de quedarse ahí, vuelva a adquirir otras 16.384 muestras automáticamente, y luego otro bloque de muestras... y así succesivamente durante el tiempo que se desee.

Para este propósito, se debe reestructurar un poco el código. Se piensa en dos alternativas:

- **Primero muestrear y luego contar los máximos:** para aplicar esta opción, se deberían ir añadiendo los bloques de muestras a la lista *buff* al final de cada iteración. Cuando finalizase la adquisición de datos, la lista *buff* pasaría a ser analizada por el código con el objetivo de encontrar todos los máximos.
- **Muestrear y contar los máximos para cada iteración:** en este caso, el algoritmo para encontrar los máximos estaría metido dentro del *for*. La lista *buff* se reiniciaría al final de cada iteración, no sin antes ser analizada.

La segunda alternativa, aunque parece más laboriosa, acaba siendo la escogida. La razón principal es que, si se aplica la primera opción para un periodo de muestreo largo (muchas iteraciones), probablemente llegue un momento en el cual la lista *buff* ya no sea capaz de llenarse más. Además, si se lleva a cabo la segunda alternativa, se pueden

monitorizar los resultados iteración a iteración.

Finalmente, el código completo del analizador monocanal definitivo queda de la siguiente forma (*Figuras 39 y 40*, únicamente se comentan las líneas nuevas):

```
from scipy.signal import find_peaks_cwt as peaks
from time import sleep
import redpitaya_scpi as scpi
import matplotlib.pyplot as plt

ip = '169.254.106.150'
rp_s = scpi.scpi(ip)

rp_s.tx_txt('ACQ:RST')
rp_s.tx_txt('ACQ:SOUR1:GAIN LV')
rp_s.tx_txt('ACQ:TRIG:LEV 0')
rp_s.tx_txt('ACQ:TRIG:DLY 8192')
rp_s.tx_txt('ACQ:DEC 64')

lim_inf = 0.22
lim_sup =  0.45
width = [3]

count = 0

"Implementación del bucle de adquisición. En este caso, 120 iteraciones definidas"
for i in range(120):
    "Al inicio de cada iteración, empieza un nuevo proceso de adquisición"
    rp_s.tx_txt('ACQ:START')
    rp_s.tx_txt('ACQ:TRIG CH1_PE')

    while 1:
        rp_s.tx_txt('ACQ:TRIG:STAT?')
        if rp_s.rx_txt() == 'TD':
            break

    rp_s.tx_txt('ACQ:SOUR1:DATA?')

    buff_string = rp_s.rx_txt()
    buff_string = buff_string.strip('{}\n\r').replace("  ", "").split(',')
    buff = list(map(float, buff_string))
```

Figura 39: *Código SCA definitivo (I). Fuente: Propia*

```
    buff_string = rp_s.rx_txt()
    buff_string = buff_string.strip('{}\n\r').replace("  ", "").split(',')
    buff = list(map(float, buff_string))

    ind_max = peaks(buff, width) # funcion lenta

    x_max = []
    y_max = []
    p_max = []

    for j in range(len(ind_max)):
        ind_max[j] = int(ind_max[j])
        if lim_inf <= buff[ind_max[j]] <= lim_sup:
            count = count + 1
            x_max.append(ind_max[j])
            y_max.append(buff[ind_max[j]])
            p_max.append((ind_max[j],buff[ind_max[j]]))

    "Monitorización del conteo y valor de los máximos iteración a iteración"
    print(i)
    print(count)
    print(p_max)

plt.plot(buff)
plt.plot(x_max, y_max, "xr")
plt.ylabel('Voltatge')
plt.grid()
plt.show()
```

Figura 40: *Código SCA definitivo (II). Fuente: Propia*

Para asegurar que la aplicación del código definitivo es exitosa, se define un periodo total

de muestreo de 10 iteraciones (163.840 muestras totales) de la señal del *Apartado 3.2.4* para factores de *diezmado 1 y 8*. El resultado es el siguiente (*Figuras 41 y 42*):

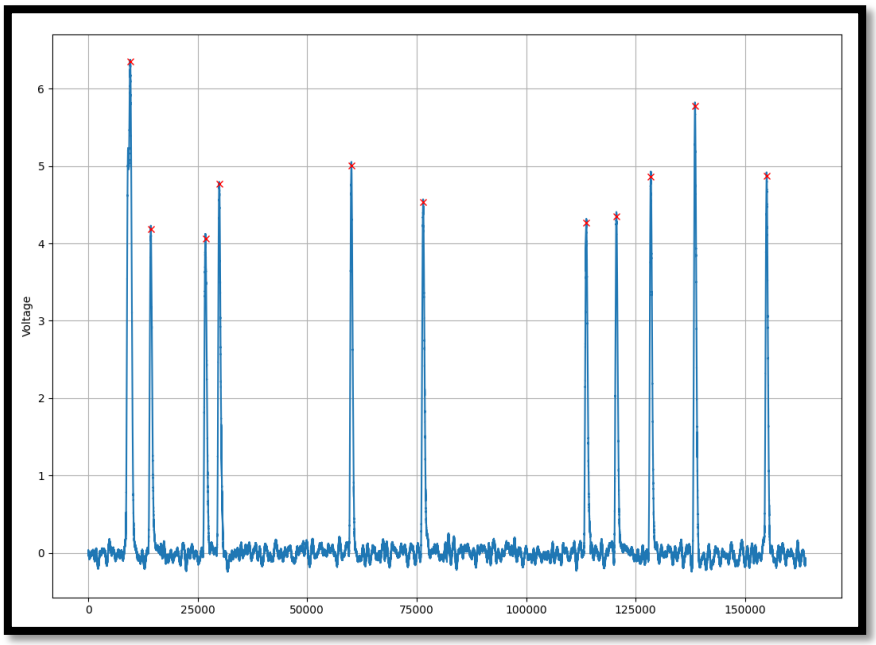


Figura 41: *Bloque de 163.840 muestras adquiridas (10 iteraciones) con diezmado 1.*

Fuente: Propia

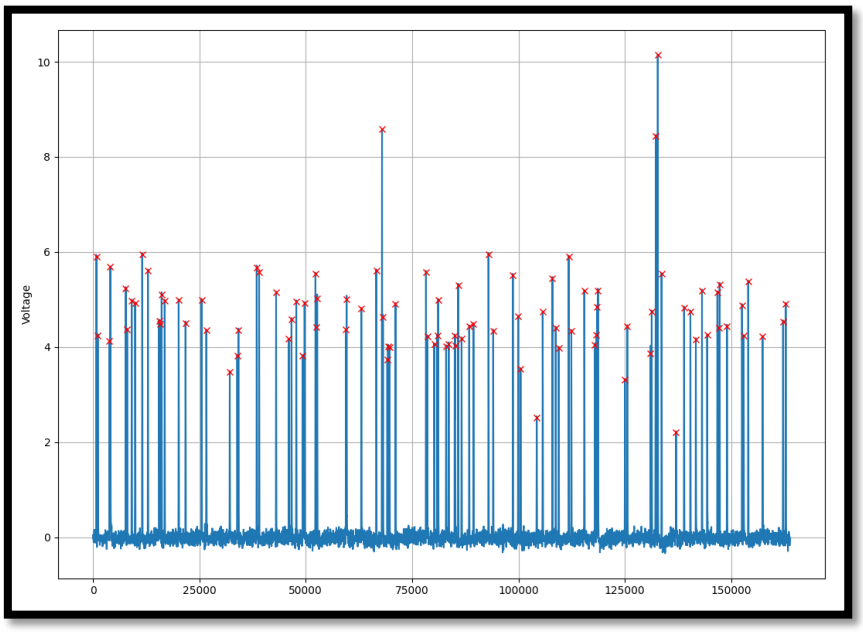


Figura 42: *Bloque de 163.840 muestras adquiridas (10 iteraciones) con diezmado 8.*

Fuente: Propia

Los resultados son, sin duda, muy buenos: la señal parece bien muestreada y los máximos marcados de manera precisa en ambos casos (ya se usan los parámetros *width* de la *Tabla 3*).

Como punto negativo, cabe recordar que la implementación del bucle tiene el gran inconveniente del tiempo muerto (ya explicado en el *Apartado 3.1.3*). Además, se puede comprobar que la incorporación de la función externa ***find_peaks_cwt*** todavía ralentiza más el proceso (lo hace entre 2 y 3 veces más lento). Ante eso no se puede hacer nada, así que si, por el propósito que sea, se necesita analizar una señal durante un periodo de tiempo muy largo (horas), el uso de la *Red Pitaya* no es una opción viable.

En este momento se da por finalizada la creación del analizador mono-canal (*SCA*) mediante el equipo *Red Pitaya*. En el *Bloque III* se pone a prueba su validez, comparándolo con un *SCA* comercial de la marca *Ortec* a través de distintas señales.

5. BLOQUE III: Comparación del *SCA* creado con un *SCA* de la marca *Ortec*

5.1. Componentes y montaje

Para testear el analizador mono-canal creado con la Red Pitaya, lo mejor que se puede hacer es comparar sus resultados con los de un SCA comercial. Para ello, en éste bloque se hace uso de un montaje compuesto de los siguientes componentes (presentados en el orden correspondiente al de su utilización):

a. Fuentes radiactivas: son las encargadas de emitir los distintos tipos de radiación captados por el detector. Concretamente, se repite el experimento para tres fuentes distintas: *Cesio-137, Cobalto-60 y Americio-241* (*Figura 42*). Más adelante se aporta más información sobre ellas y se muestran sus espectros.

Figura 43: *Fuentes de ^{241}Am, ^{60}Co y ^{137}Cs, de izquierda a derecha. Fuente: Propia*

b. Fuente de alta tensión *Canberra 3196D*: alimenta al detector. Este modelo está particularmente bien adaptado para el uso con detectores de alta resolución. El voltaje de la salida es continuamente ajustable de *±30 V* a *±6000 V*. [23]

Figura 44: *Fuente de alta tensión Canberra 3196D. [23]*

c. Detector de centelleo de *NaI (Tl)* de *3x3 pulgadas*: tiene la función de captar toda la radiación que emite la fuente. Luego, transmite lo detectado en forma de carga hacia el preamplificador. Concretamente, el centelleador *NaI (Tl)* tiene una relación precio-resolución muy buena y permite determinaciones de energía suficientemente precisas para este proyecto. Una desventaja de algunos cristales inorgánicos como el *NaI* es su higroscopicidad, una propiedad que requiere que se alojen en un recipiente hermético para protegerlos de la humedad. [24]

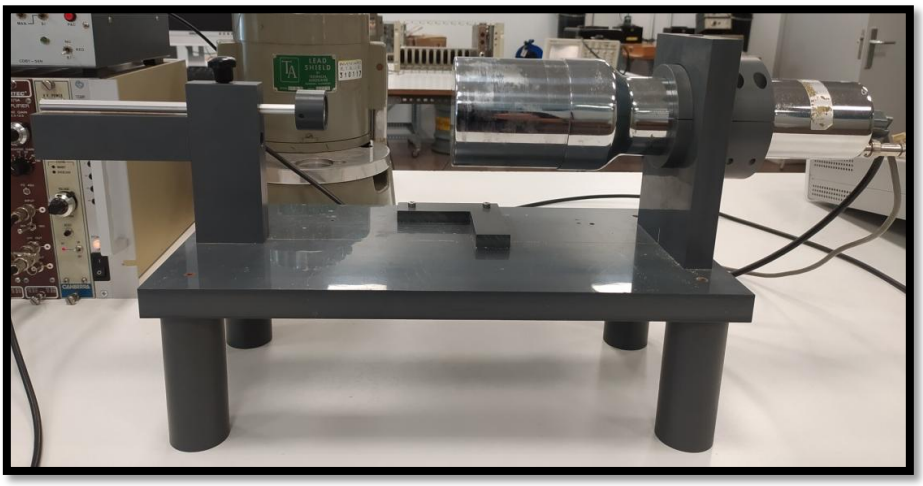


Figura 45: *Detector de centelleo de NaI (Tl) de 3x3 pulgadas. Fuente: Propia*

d. Preamplificador *Canberra 2007P*: tal como se comenta en el *Apartado 3.2.4*, funciona como convertidor de carga a voltaje, es decir, la salida proporciona una tensión directamente proporcional a la carga recogida.

Figura 46: Preamplificador Canberra 2007P. [25]

e. Amplificador *Ortec 550A*: tiene la misma función que en el *Apartado 3.2.4*. Se usa para hacer más clara la señal de salida del preamplificador. Elimina el apilamiento convirtiendo la señal en pulsos *cuasi-Gaussianos* y, por lo tanto, más sencillos de analizar. Se puede modificar la ganancia del amplificador en función del nivel de tensiones con el que se desee trabajar. En la figura 47, se observa que recibe una señal (provinente del preamplificador, aunque no se vea en la imagen) mediante el *input* y luego se pueden apreciar dos salidas a través del *output*: una de ellas va a parar al osciloscopio y la otra a la *Red Pitaya*.

Figura 47: Amplificador Ortec 550ª. *Fuente: Propia*

f. *A*nalizador multi-canal (*MCA*) *Ortec*: se trata de una tarjeta *Trump-PCI* de la marca *Ortec* que se conecta al ordenador a través de una de sus ranuras libres. Es el encargado de generar el espectro de cada una de las fuentes. Para controlarlo y visualizar los espectros, se hace uso del emulador *Maestro*.

Todos los tipos de radiación captados por el detector tienen un nivel de energía determinado. La espectrometría funciona de la siguiente forma:

- El *eje X* se divide en 1024 particiones de tensión, la primera de las cuales corresponde a *0 V* y la nº 1024 a *10 V*. La tensión es proporcional a la energía (más a la derecha implica más energía).
- El eje Y representa el número de cuentas. Si se deja trabajar al multi-canal durante el tiempo suficiente (cuestión de segundos), se observa que hay ciertos niveles de tensión en los que las cuentas se van acumulando. Estas zonas de apilamiento corresponden a un tipo de radiación determinado.

Por ejemplo, tal como se detalla en el *Apartado 5.3.1*, para la espectrometría del *Cesio-137*, se puede observar un mayor número de cuentas en la ventana de *3,1 V a 3,81 V* (*Figura 48*). Todas las cuentas que se apilan en esa zona son rayos *gamma* y corresponden a un nivel de energía de *661 KeV*.

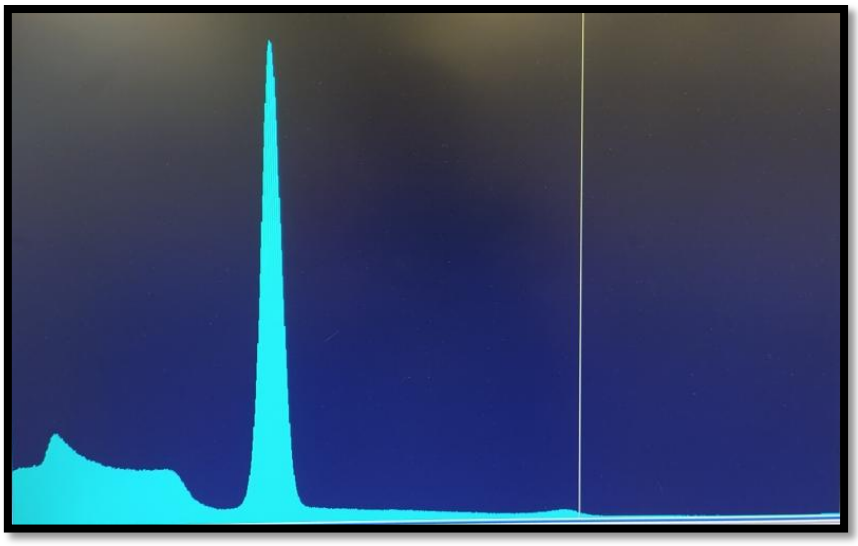


Figura 48: *Espectro del Cesio-137 creado con el Software Maestro. Fuente: Propia*

En este proyecto, el objetivo del *MCA* es sacar las ventanas de tensión (límites superior e inferior) correspondientes a ***rayos gamma*** para cada una de las fuentes. Estos límites se usan a continuación en el *SCA*.

g. Analizador mono-canal (*SCA*) *Ortec 550A:* en él, se establecen los límites de tensión sacados del MCA. Es primordial definirlos con exactitud, ya que de ellos depende el resultado que proporcionará el contador.

Figura 49: *Analizador mono-canal (SCA) Ortec 550A. Fuente: Propia*

h. Contador: se dedica a contar durante un tiempo definible el número de partículas captadas por el detector que tienen un valor de tensión que cae dentro de la ventana sacada de la espectrometría (y definida en el *SCA)*.

Figura 50: *Contador. Fuente: Propia*

i. **Osciloscopio *Tektronix TBS 1152B*:** ayuda a visualizar la señal de salida del componente que se desee. Es útil para detectar problemas en el montaje, ya que si no muestra lo esperado, es indicativo de que alguna conexión o parámetro se ha definido de forma errónea.

   Para este proyecto, es especialmente útil conectarlo en paralelo junto con la *Red Pitaya* a la salida del amplificador. De esta forma, se pueden comparar ambas señales (la muestreada con la *Pitaya* y la del osciloscopio).

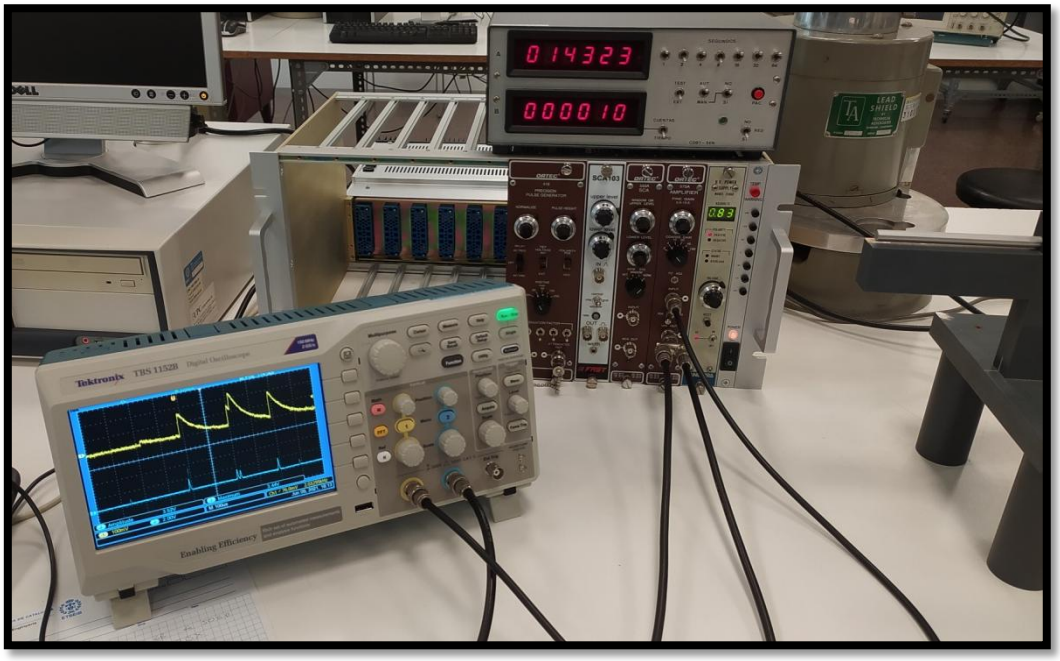


Figura 51: *Visualización de la señal antes (canal amarillo) y después de pasar por el amplificador (canal azul). Fuente: Propia*

Finalmente, se adjunta un diagrama de bloques que muestra todas las conexiones a realizar entre los componentes presentados para completar el montaje de forma adecuada (*Figura 52*):

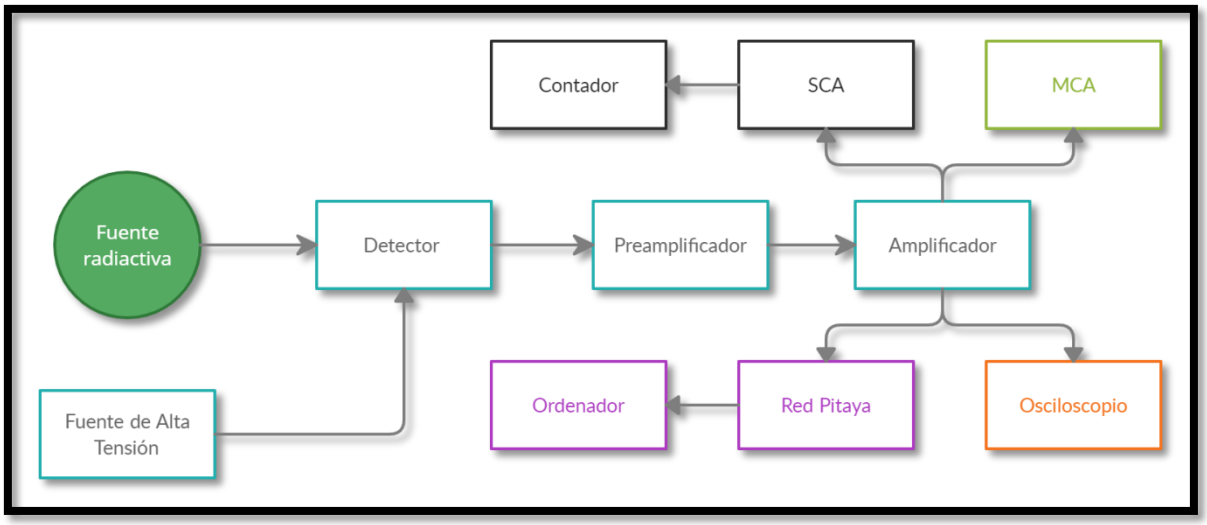


Figura 52: *Conexiones entre los distintos componentes del montaje. Fuente: Propia*

5.2. Sistema de comparación

El proceso a seguir para llevar a cabo una correcta comparación entre ambos SCA se expone paso a paso en la siguiente *Tabla 4*:

PASO	SCA ORTEC	SCA RED PITAYA
1	Realizar el montaje correcto y todas las conexiones pertinentes.	
2	Escoger la fuente y colocarla cerca del detector.	
3	Obtener la espectrometría de la fuente mediante el *MCA*.	
4	Detectar y anotar cuáles son los límites superior e inferior de tensión correspondientes a radiación de tipo *gamma*.	
5	Definir los límites en el SCA *Ortec*.	Definir los límites (parámetros *lim_inf* y *lim_sup*) en el código.

6	Inicializar el contador (los conteos son de 10 segundos).	Decidir el factor de diezmado con el que se va a muestrear la señal (se usan *diezmados 8 y 64*)
7	Anotar el número total de cuentas y repetir el proceso varias veces (se hace 5 veces en este caso) para ganar fiabilidad.	Determinar un tiempo total de muestreo lógico según el diezmado escogido (se muestrearán *100 ms* para *diezmado 8* y *1 segundo* para *diezmado 64*).
8	Hacer y anotar la media de los 5 conteos.	Determinar el número de iteraciones necesarias para poder muestrear durante los tiempos definidos (95 y 120 iteraciones respectivamente).
9	Comparar los datos con los del otro *SCA*, teniendo en cuenta la diferencia de tiempo de muestreo.	
10	Volver al paso 2, escoger otra fuente y repetir los pasos 3-9.	
11	Sacar conclusiones en base a los resultados obtenidos.	

Tabla 4: *Procedimiento paso a paso para la correcta comparación de ambos SCAs.*
Fuente: Propia

5.3. Comparativa

Cuando el montaje ya está realizado a la perfección, es el momento de escoger la fuente. Tal como se ha comentado anteriormente, la idea es repetir el proceso para 3 fuentes distintas: *Cesio-137, Cobalto-60 y Americio-241*.

5.3.1. Fuente de Cesio-137

El *Cesio-137* es un isótopo radiactivo obtenido artificialmente con un periodo de semidesintegración de 32 años. Fundamentalmente, tiene aplicaciones industriales y médicas. Dentro del ámbito médico, se utiliza para la irradiación de tumores ginecológicos (se aplica a las pacientes para emitir una irradiación localizada del tumor

sin dañar órganos vitales cercanos). [26]

Mediante el Software Maestro del analizador multi-canal, se obtiene el siguiente espectro (*Figura 53*).

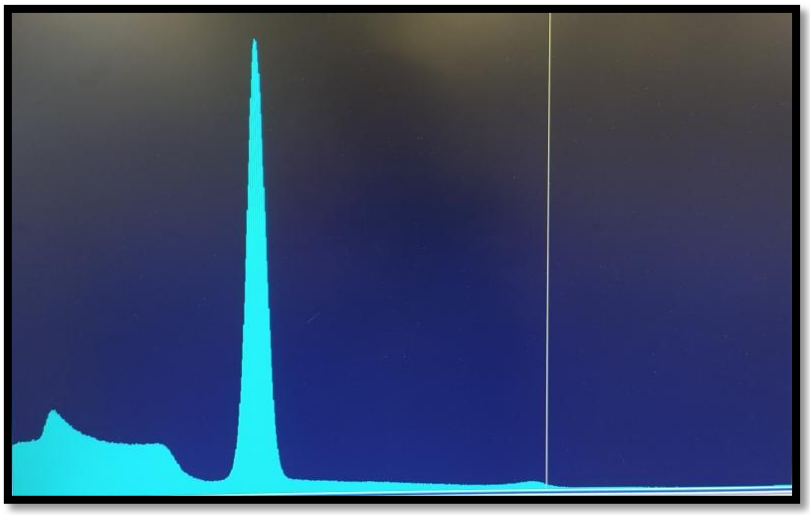


Figura 53: *Espectrometría del ^{137}Cs*

Los rayos *gamma* de este isótopo radiactivo tienen un nivel de energía de unos *661 KeV* y corresponden a la alta y estrecha montaña que se puede apreciar en la parte central-izquierda de la imagen. Con la ayuda del programa, se percibe que este conjunto de partículas se apilan entre las marcas nº 317 y nº 390 del eje X (recordar que en total hay 1024 marcas, siendo la última de ellas equivalente a *10 V*).

Haciendo una simple regla de tres, se sacan los límites de tensión equivalentes a las marcas:

- Límite inferior: marca nº 317 = **3,1 V.**
- Límite superior: marca nº 390 = **3,81 V.**

Seguidamente, se introducen los valores de tensión tanto en el *SCA* comercial como en la *Red Pitaya*. Teniendo en cuenta el rango de tensiones, se trabaja con los jumpers de la *Pitaya* en modo *HV*.

A continuación, ya se puede proceder a realizar los conteos. Se deciden realizar 5 conteos de *10 segundos* para el SCA *Ortec*. En vista de los resultados (número de cuentas muy elevado) y con la finalidad de ahorrar tiempo, se decide que no es necesario muestrear tales cantidades de tiempo con la Red Pitaya.

Por practicidad, se descarta usar un factor de *diezmado 1* (se considera que para muestrear un mínimo de tiempo aceptable se tardaría demasiado). Finalmente, se deciden establecer dos métodos de conteo con el equipo Red Pitaya:

- **95 iteraciones con factor de *diezmado 8*:** de esta forma, se consigue muestrear la señal de salida del amplificador durante 100 ms. Por lo tanto, se espera un total de cuentas unas 100 veces menor al del *SCA Ortec*. El proceso se repite 5 veces.
- **120 iteraciones *con factor de diezmado 64:*** en este caso, se consigue muestrear la señal durante *1 segundo*. Por lo tanto, se espera un total de cuentas unas 10 veces menor al del SCA comercial. El proceso también se repite 5 veces.

Análisis estadístico

El resultado de los conteos resulta ser el siguiente (*Tabla 5*):

Conteo	SCA Ortec (10 s)	SCA Pitaya (0,1 s)	SCA Pitaya (1 s)
1	121.399	1.173	12.346
2	121.075	1.220	12.458
3	120.788	1.216	12.001
4	120.636	1.185	12.096
5	121.320	1.209	12.154
Media	121.043	1.200	12.211

Tabla 5: *Conteo de partículas gamma del ^{137}Cs para los 3 métodos utilizados. Fuente: Propia*

Los resultados son bastante parecidos para los 3 métodos de conteo. Las pequeñas diferencias son debidas a que se han hecho pocas mediciones (sólo 5 en cada caso). Para acabar de afinar, se deberían hacer más, sobretodo en el caso del Diezmado 8, ya que sólo se está muestreando durante 100 ms.

La varianza es una medida de dispersión que representa la variabilidad de una serie de datos respecto a su media. Viene dada por la siguiente expresión (*Ecuación 1*) :

$$\sigma^2(\bar{n}) = \frac{1}{m_G^2 t_G^2}\sum_{i=1}^{m_G} G_i + \frac{1}{m_B^2 t_B^2}\sum_{i=1}^{m_B} B_i$$

Ecuación 1: *Fórmula de la varianza.* [27]

- Los términos B_i hacen referencia a las cuentas debidas a la radiación del entorno. Pueden ser considerados 0 ya que todos los conteos son realizados bajo las mismas circunstancias (mismo entorno). Así pues, todo el segundo término de la suma se puede eliminar.
- Los términos G_i hacen referencia a las cuentas totales obtenidas en cada conteo *i*.
- m_G es el número de conteos
- t_G es el tiempo que dura el conteo.

Una vez se conocen estos parámetros estadísticos, se calcula el intervalo de confianza, que se puede definir como la ventana dentro de la cual se estima que estará cierto valor desconocido con un determinado nivel de confianza. La fórmula es la siguiente:

$$\boldsymbol{I_c = [\bar{G} - k \cdot \sigma, \bar{G} + k \cdot \sigma]}$$

Equación 2: *Intervalo de confianza.* [27]

- $\bar{G}$ es la media de los 5 conteos.
- k es un parámetro que depende del error que definido. Se escoge una probabilidad del 99%, lo que equivale a k = 2,58.

A continuación, se presentan los resultados obtenidos para cada uno de los métodos de conteo (*Tabla 6*):

Método	SCA Ortec (10 s)	SCA Pitaya (DEC 8)	SCA Pitaya (DEC 64)
σ^2 [cps^2]*	242	24.012	2.442
σ [cps]	15,56	154,96	49,42
I_C	[12.064, 12.144]	[11.606 , 12.406]	[12.083 , 12.338]

Tabla 6: *Parámetros estadísticos del ^{137}Cs para los 3 métodos de conteo utilizados.*

**cps:* *cuentas por segundo. Fuente: Propia*

Lógicamente, cuanto mayor es el tiempo de muestreo, más fiables son los resultados. El *Intervalo de confianza* va quedando reducido a medida que crece el tiempo de muestreo. Sin embargo, en este caso, cabe remarcar que con la Pitaya muestreando con *diezmado 8 (0,1 s)* se obtiene un intervalo de confianza más acorde que con *diezmado 64 (1 segundo)*.

Los dos dos primeros conteos con diezmado 64, casualmente, resultan muy elevados, hecho que provoca un incremento erróneo en la media (y, consecuentemente, del *Ic*).

Visualización del proceso

A continuación, se puede observar un proceso de muestreo realizado por la Pitaya con diezmado 8 (*Figura 54*). Como ya se ha explicado en apartados anteriores, los picos de señal que quedan dentro de los límites establecidos se remarcan con crucecitas de color rojo y son contados por el SCA.

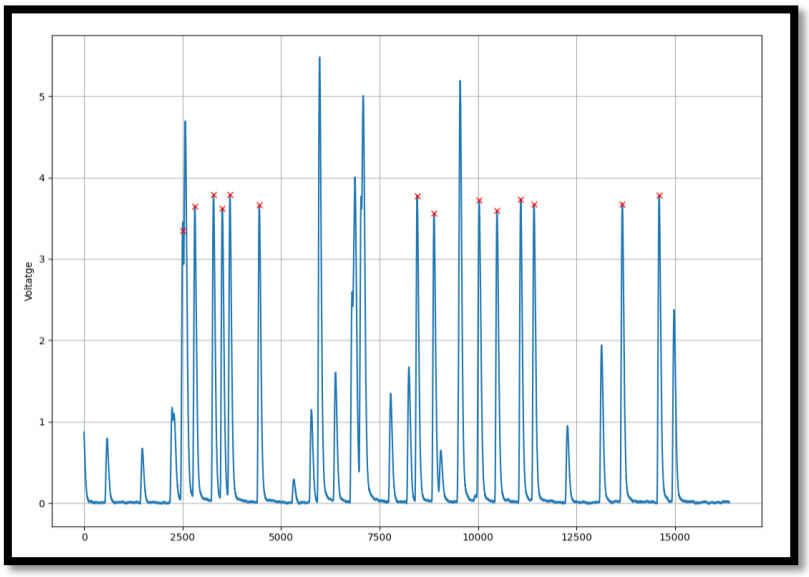


Figura 54: *Funcionamiento del SCA de Red Pitaya con ^{137}Cs y diezmado 8. Fuente: Propia*

Usando diezmado 64 se observa que la señal queda mucho más compactada, pero, de todas formas, el SCA funciona satisfactoriamente (*Figura 55*):

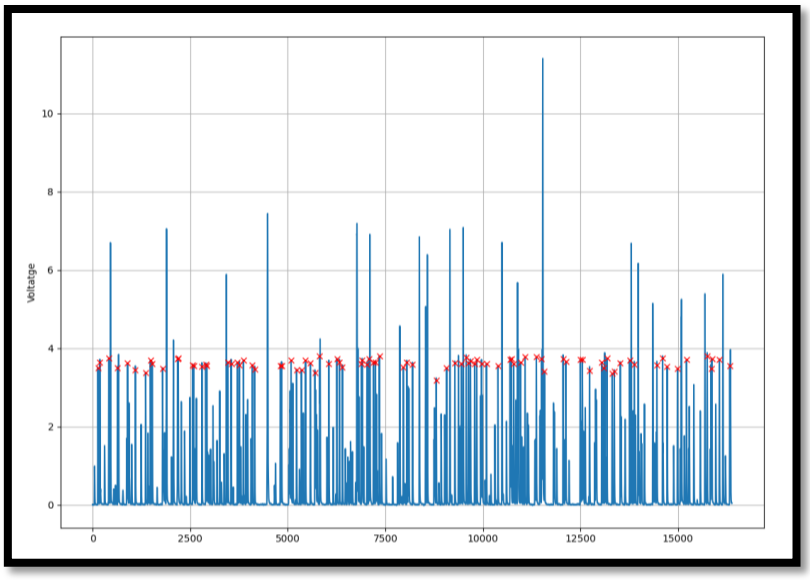


Figura 55: *Funcionamiento del SCA de Red Pitaya con ^{137}Cs y diezmado 64. Fuente: Propia*

5.3.2. Fuente de Americio-241

Como todos los isótopos del americio, el ^{241}Am es radiactivo. Tiene un periodo de semidesintegración de 432,2 años. Sus aplicaciones abarcan distintos campos. Algunos detectores de humo contienen una pequeña muestra (cerca de *0,2 mg*) de ^{241}Am como fuente de radiación ionizante. Este mismo isótopo puede ser utilizado como fuente portátil de rayos gamma para su uso en radiografías. *[28]*

Mediante el Software Maestro del analizador multi-canal, se saca la siguiente espectrometría (*Figura 56*).

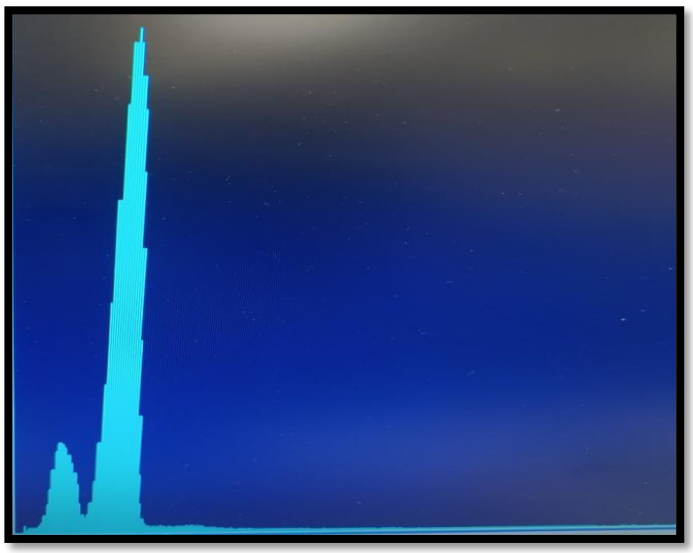


Figura 56: *Espectro del* ^{241}Am. *Fuente: Propia*

Lor rayos *gamma* de este isótopo radiactivo tienen un nivel de energía de unos *59 KeV* y corresponden al alto y estrecho apilamiento de conjunto de cuentas que se puede apreciar en la parte izquierda de la imagen. Con la ayuda del programa, se percibe que este conjunto de partículas se acumulan entre las marcas nº 23 y nº 46 del eje X.

No está de más comentar que no se modifica la ganancia de ningún componente del montaje al substituir las fuentes. Así pues, si una energía de *661 KeV* corresponde más o menos a la marca nº 354 del *eje X* del espectro (punto medio de la ventana del *Cesio-137*), se puede intuir que los *59 KeV* se encuentran alrededor de la marca nº 32 del mismo. Efectivamente, esa marca forma parte del intervalo encontrado, así que se puede asegurar que el procedimiento seguido es correcto.

Igual que en el caso anterior, se sacan los límites de tensión equivalentes a las marcas:
Límite inferior: marca nº 23 = ***0,22 V*****.**
Límite superior: marca nº 46 = ***0,45 V*****.**

De manera análoga al apartado anterior, se introducen los valores de tensión tanto en el *SCA* comercial como en la *Red Pitaya*. En este caso, teniendo en cuenta el rango de tensiones, se trabaja con los jumpers de la *Pitaya* en modo *LV*.

El método de conteo, tanto para la *Red Pitaya* como para el *SCA Ortec* se implementa de forma idéntica que en el caso del *Cesio-137*.

Análisis estadístico

Los conteos resultantes se pueden visualizar en la *Tabla 7*:

Conteo	SCA Ortec (10 s)	SCA Pitaya (0,1 s)	SCA Pitaya (1 s)
1	26.111	245	2.537
2	26.029	273	2.673
3	25.964	271	2.621
4	26.001	251	2.568
5	25.853	268	2.629

Media	25.992	262	2.611

Tabla 7: *Conteo de partículas gamma del* ^{241}Am *para los 3 métodos utilizados. Fuente: Propia*

En este caso, los tres métodos de conteo también proporcionan resultados similares.

Se cree que ha habido bastante suerte en el caso del *diezmado 8*, ya que han salido 2 conteos inferiores a lo esperado (245 y 251), pero éstos se han visto compensados por 3 conteos superiores (273, 271 y 268). Al final, la media resulta satisfactoria, pero realmente se ve necesario hacer bastantes más conteos para obtener valores más fiables.

En la *Tabla 8* se presentan los parámetros estadísticos obtenidos para cada uno de los métodos de conteo:

Método	**SCA Ortec (10 s)**	**SCA Pitaya (DEC 8)**	**SCA Pitaya (DEC 64)**
σ^2 [cps^2]	52	5.232	521
σ [cps]	7,21	72,33	22,83
I_C	[2.581, 2.618]	[2.429 , 2.803]	[2.547 , 2.664]

Tabla 8: *Parámetros estadísticos del* ^{241}*Am para los 3 métodos de conteo utilizados. Fuente: Propia*

Los resultados son acordes a lo esperado. El intervalo de confianza de más anchura corresponde al tiempo de muestreo más corto. A medida que se aumenta el tiempo de muestreo, los límites de ese intervalo van reduciendo su distancia paulatinamente.

Visualización del proceso

A continuación, se puede observar un proceso de muestreo realizado por la *Pitaya* con *diezmado 8 (Figura 57)*. Mientras los 4 picos que caen dentro de la ventana son contabilizados, el pulso de menor amplitud (que corresponde a radiación *alfa*) es descartado por el *SCA*.

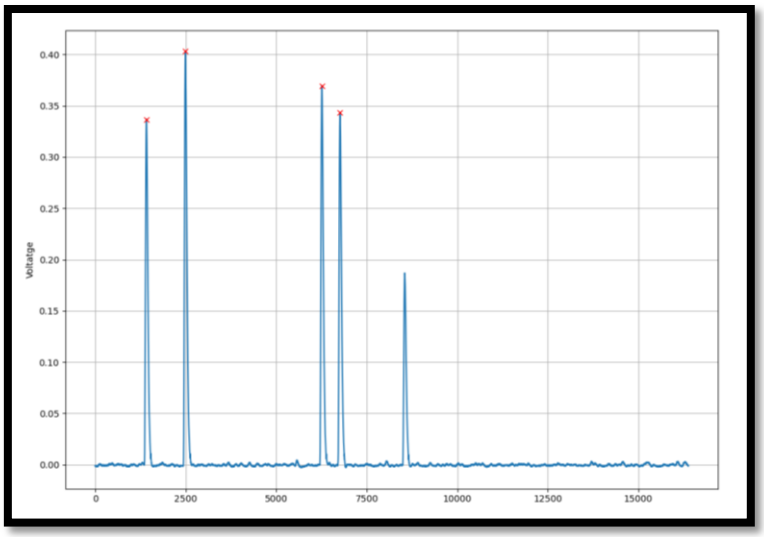


Figura 57: *Funcionamiento del SCA de Red Pitaya con* ^{241}Am *y diezmado 8. Fuente: Propia*

Usando *diezmado 64* se observa que el SCA también funciona satisfactoriamente (*Figura 58*):

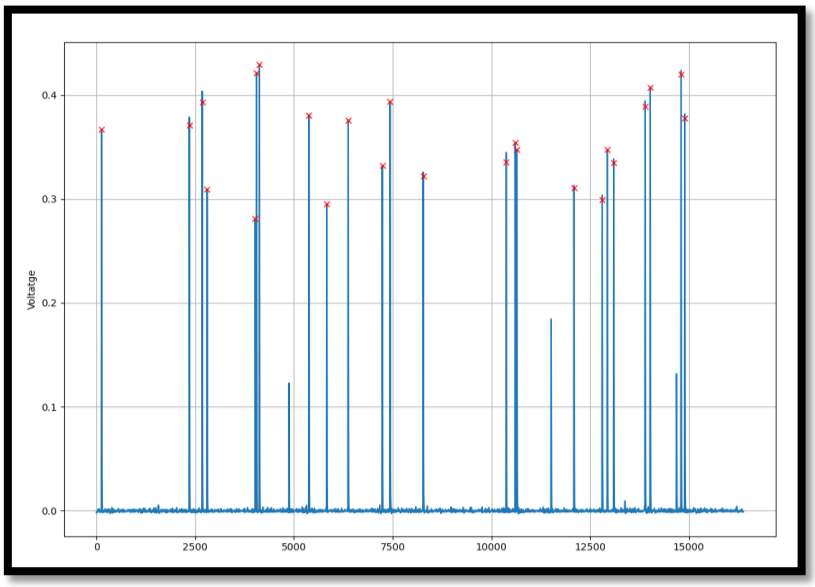


Figura 58: *Funcionamiento del SCA de Red Pitaya con* ^{241}Am *y diezmado 64. Fuente: Propia*

5.3.3. Fuente de *Cobalto-60*

El ^{60}Co es un isótopo radiactivo sintético del cobalto, con un periodo de semidesintegración de 5,27 años. La mayoría de las instalaciones de procesamiento de radiaciones industriales y médicas utilizan *Cobalto-60*. Por ejemplo, hoy en día resulta muy útil en procesos de esterilización de equipos médicos, entre otras aplicaciones. [29]

Mediante el *Software Maestro* del *MCA*, se obtiene el siguiente espectro (*Figura 59*):

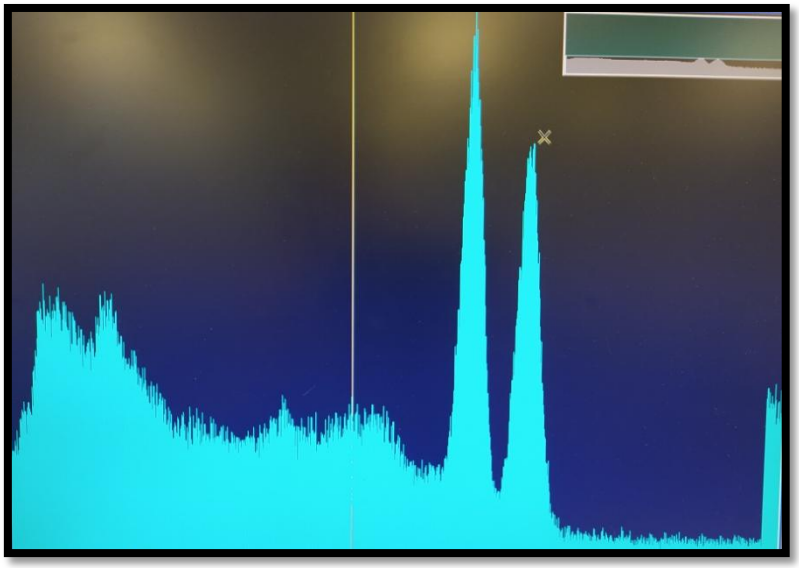


Figura 59: *Espectrometría del ^{60}Co. Fuente: Propia*

Los rayos gamma de este isótopo radiactivo pueden tener dos niveles de energía distintos, aunque relativamente cercanos. El primer grupo de gammas posee un nivel de energía de unos *1,1 MeV* y la energía del segundo grupo es de alrededor de *1,3 MeV*. Estos niveles de energía corresponden a los dos picos prácticamente consecutivos de partículas que se ven en la parte central-derecha del espectro.

Las ventanas de tensiones de los dos tipos de gamma son los siguientes:

- Gammas de *1,1 MeV*:
  - Límite inferior: marca nº 560 = ***5,45 V*.**
  - Límite superior: marca nº 635 = ***6,2 V.***
- Gammas de 1,3 MeV:
  - Límite inferior: nº 635 = ***6,2 V.***
  - Límite superior: nº 712 = ***6,95 V*.**

Teniendo en cuenta los valores de los límites, se trabaja con los jumpers de la Pitaya en

modo *HV*.

Se aplica el mismo procedimiento que para las fuentes anteriores, con una pequeña diferencia: en este caso, se descarta usar el *diezmado 8* ya que las cuentas obtenidas durante *10 segundos* son de alrededor de 2.000. Con 95 iteraciones a *diezmado 8 (100 ms*), se esperaría obtener únicamente alrededor de 20 cuentas y los resultados no serían concluyentes con dicha cantidad. Sólo se hace uso del *diezmado 64*.

Análisis estadístico

Los conteos referentes a las partículas gamma de *1,1 MeV* quedan así (*Tabla 9*):

Conteo	SCA Ortec (10 s)	SCA Pitaya (0,1 s)
1	2.331	245
2	2.393	233
3	2.464	223
4	2.481	220
5	2.425	242
Media	2.419	233

Tabla 9: *Conteo de partículas gamma (1,1 MeV) del ^{60}Co para los 2 métodos utilizados. Fuente: Propia*

Parece ser que, aunque no son muy distantes, los resultados de la *Red Pitaya* no son del todo satisfactorios. Eso es debido a que 2 de los conteos son un poco inferiores a lo esperado (220 y 223), reduciendo bastante la media. Sería conveniente muestrear más tiempo para obtener resultados más concluyentes.

En la *Tabla 10* se presentan los parámetros estadísticos obtenidos para ambos métodos de conteo:

Método	SCA Ortec (10 s)	SCA Pitaya (DEC 8)
σ^2 [cps^2]	5	47
σ [cps]	2,2	6,82
I_C	[236 , 247]	[215 , 250]

Tabla 10: *Parámetros estadísticos del ^{60}Co (1,1 MeV) para los 2 métodos de conteo utilizados. Fuente: Propia*

Por otro lado, los conteos referentes a las partículas gamma de *1,3 MeV* resultan ser los siguientes (*Tabla 11*):

Conteo	SCA Ortec (10 s)	SCA Pitaya (0,1 s)
1	2.212	202
2	2.168	228
3	2.206	220
4	2.247	216
5	2.128	218
Media	2.192	217

Tabla 11: *Conteo de partículas gamma (1,3 MeV) del ^{60}Co para los 2 métodos utilizados. Fuente: Propia*

Para las gammas de *1,3 MeV*, el número de cuentas es ligeramente inferior al de las gammas de *1,1 MeV*. Esto concuerda con lo visto en el espectro (la acumulación de cuentas es un poco inferior).

En este caso, los resultados con la Pitaya son muy acordes a lo esperado. Sin embargo, esto no signfica que se haya establecido la ventana de tensión de forma más precisa que en el caso anterior. Simplemente, se ha tenido más fortuna en este bloque de conteos.

La *Tabla 12* muestra los parámetros estadísticos obtenidos para ambos métodos de conteo:

Método	SCA Ortec (10 s)	SCA Pitaya (DEC 8)
σ^2 [cps^2]	4	43
σ [cps]	2,09	6,58
I_C	[214 , 225]	[200 , 234]

Tabla 12: *Parámetros estadísticos del ^{60}Co (1,3 MeV) para los 2 métodos de conteo utilizados. Fuente: Propia*

Visualización del proceso

Por último, comentar una modificación bastante interesante que se realiza en el código con la intención de optimizar el análisis del *Cobalto-60* mediante *Red Pitaya*. Básicamente, se decide añadir una nueva ventana de tensiones al código, a la vez que se implementa un nuevo contador. De esta forma, se puede realizar el conteo de los dos tipos de rayos *gamma* a la vez (contar los dos tipos de máximos en un mismo periodo de muestreo), no es necesario repetir el proceso en dos ocasiones. Coloquialmente, se podría decir que se crea un bicanal. Para ver cómo queda modificado el código, ver el *Anexo*.

Finalmente, se muestran dos procesos de muestreo realizados por la *Pitaya* con *diezmado 64 (Figuras 60 y 61)*. Las gammas de energía *1,1 MeV* son remarcadas con cruces rojas y las de *1,3 MeV* con cruces azules. Viendo los resultados, se concluye que el bicanal funciona perfectamente.

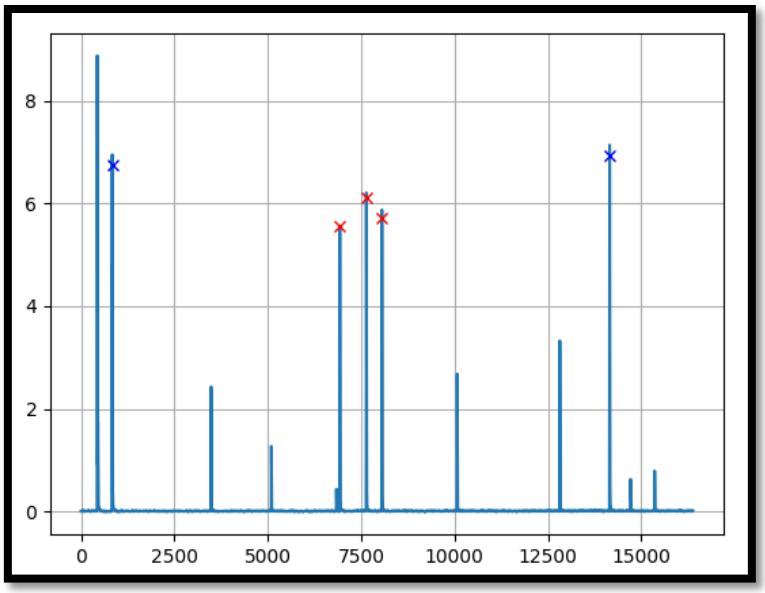


Figura 60: *Funcionamiento del bicanal de Red Pitaya con ^{60}Co y diezmado 64 (I).*

Fuente: Propia

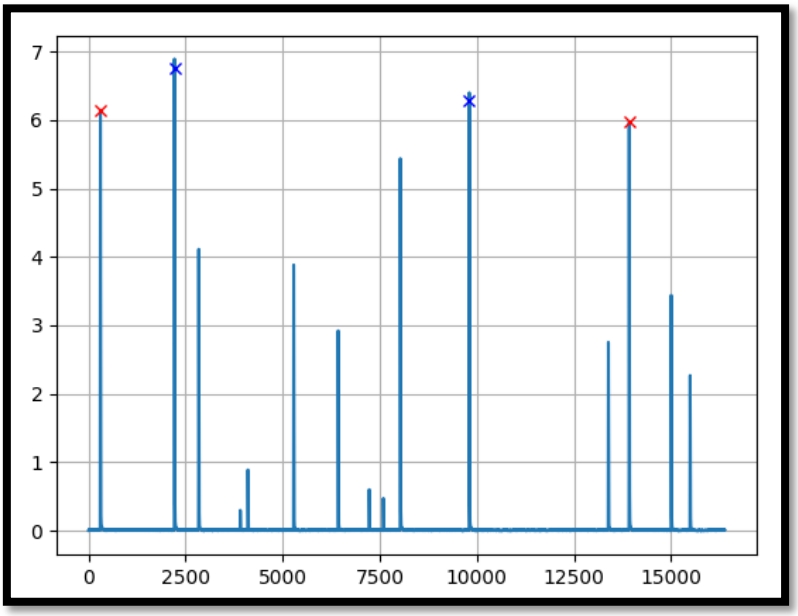


Figura 61: *Funcionamiento del bicanal de Red Pitaya con ^{60}Co y diezmado 64 (II).*

Fuente: Propia

En este momento, la comparación de los dos analizadores monocanal (*SCA Ortec* y *SCA Red Pitaya*) llega a su fin. Con ello, concluye la parte práctica de este TFG.

6. PRESUPUESTO E IMPACTO AMBIENTAL

Para confeccionar el apartado económico del proyecto se parte de la base de que se trabaja en el laboratorio de Instrumentación Nuclear. Se considera, pues, que ya se dispone de la istrumentación necesaria para realizar las diferentes tareas.

Por lo tanto, no se incluyen en el presupuesto elementos tales como el ordenador del laboratorio, el osciloscopio, los detectores de radiación ionizante, los cables BNC, y un largo *etcétera*. Únicamente se tienen en cuenta el precio del equipo *STEMlab 125-14 Red Pitaya* y sus accesorios y los sueldos del usuario que desarrola el proyecto y de su supervisor.

El equipo *STEMlab 125-14 Red Pitaya* tiene un precio de alrededor de 350€. Lógicamente, tanto el cable *Ethernet* como el cable de alimentación, ambos indispensables para el uso del equipo, vienen incluidos en el pack inicial. En contrapartida, el resto de accesorios deben ser comprados a parte (cable *SMA-SMA*, adaptador *SMA-BNC*).

En cuanto al usuario desarrolador del proyecto, se tiene en cuenta que trabaja en el laboratorio de Instrumentación Nuclear una media de 8 horas a la semana (normalmente repartidas en dos jornadas). Además, también realiza una parte importante del proyecto desde casa, unas 12 horas semanales de media.

El supervisor del proyecto, que en este caso es el director del TFG, también contribuye y dedica parte de su tiempo al trabajo. Entre reuniones y montajes en el laboratorio se puede considerar que su dedicación al proyecto es de unas 5 horas semanales.

En resumen, el coste estimado del proyecto se presenta en la *Tabla 13*:

Equipos y accesorios	**Precio (€)**	**Total (€)**	
Red Pitaya	350	365,36	
Cable SMA-SMA	12,55		
Adaptador BNC-SMA	2,81		
Usuarios	**Tiempo trabajado (h)**	**Coste unitario (€/h)**	**Total (€)**
Desarrollador del proyecto	240	15	3.600
Supervisor del proyecto	60	25	1.500
Ordenador personal	240	0,021	5,10
		Total (€)	**5.470,46€**

Tabla 13: *Recopilación de los costes del proyecto. Fuente: Propia*

En cuanto al impacto ambiental del proyecto, se puede decir que es mínimo. El equipo *STEMlab 125-14 Red Pitaya* no emite ni genera residuos de ningún tipo. En referencia a su vida útil, se puede decir que es bastante incierta. De hecho, como está conformada por distintos componentes, es muy probable que algunos de ellos dejen de funcionar antes que otros. En tal caso, no hay problema aparente, ya que el *hardware* de la *Pitaya* es fácilmente accesible (permite sustituir cualquier elemento rápidamente).

Los componentes electónicos de la *Red Pitaya* que dejan de funcionar pasan por un proceso de reciclaje, el cual incluye las fases de recogida, transporte para su almacenaje y posterior tratamiento. El proceso se da por finalizado una vez se obtienen las materias primas a partir de las cuales se fabricarán nuevos productos para su puesta en el mercado, es decir, cuando se produce su reintroducción en el ciclo de vida.

El impacto ambiental de este TFG básicamente se limita a la cantidad de energía

eléctrica consumida por el ordenador personal. Teniendo en cuenta que durante las 240 horas de trabajo del desarrollador el uso del ordenador ha sido indispensable, se puede estimar que el consumo total de energía eléctrica ronda los 48 KWh.

7. CONCLUSIONES

En general, el rendimiento del equipo *Red Pitaya* funcionando por control remoto (*SCPI*) dista mucho de ser óptimo. Todos los problemas radican en el infame tiempo muerto entre muestreos, el cual ralentiza muchísimo la adquisición de señales rápidas. Además, la incorporación del algoritmo detector de máximos al código también reduce la velocidad de los procesos de muestreo. Se puede concluir que, cuanta más cantidad (tiempo) de señal se necesita muestrear, menos útil resulta la *Pitaya*.

Aparentemente, existe la posibilidad de lidiar con el tema del tiempo muerto programando de forma interna con la *Red Pitaya*. Se encuentra disponible una lista de funciones integradas (*APIs*) con las que se pueden crear algoritmos en *lenguaje C*, los cuales son directamente ejecutables en la placa *Red Pitaya*. Según se ha podido leer, trabajando de este modo la adquisición de señal puede realizarse de manera cuasi-continua. No se ha explorado esta opción, pero sería interesante hacerlo de cara a una mejora y ampliación del proyecto.

Particularmente, para este proyecto no se ha necesitado muestrear cantidades de señal que imposibiliten el uso de *Red Pitaya*. Los pulsos analizados (provinentes del amplificador) son muy numerosos, hecho que facilita un análisis adecuado de la señal aunque se muestree durante poco tiempo. Por lo tanto, se puede decir que se han cumplido los objetivos principales planteados en el apartado *1.1*:

- Se ha conseguido realizar procesos de muestreo de señal precisos usando distintas frecuencias de muestreo con el equipo *Red Pitaya*.
- La creación del *SCA* también ha resultado satisfactoria, pues los resultados obtenidos en el apartado de comparación con un *SCA Ortec* (*Bloque III*) así lo corroboran.

Este trabajo se ha dado por finalizado con la creación y posterior validación del código del *SCA* creado con *Red Pitaya*. De cara a darle continuidad al proyecto, el siguiente paso podría ser diseñar la aplicación del *SCA* y poder incorporarla a la interfaz web de *Red Pitaya*. De esta forma, se podría trabajar de forma mucho más cómoda, sin la necesidad de ir modificando todos los parámetros desde el editor de código.

Como comentario final, decir que este proyecto ha resultado muy útil para aprender conceptos de distintos campos de ingeniería. Se han tocado desde nociones de programación hasta temas de instrumentación nuclear, pasando por operaciones de procesamiento de señal.

8. AGRADECIMIENTOS

- A **Alfredo de Blas**, tutor del trabajo: de entrada, por hacer un hueco para dirigir mi *TFG* cuando ya tenía unos cuantos alumnos asignados. Me ha gustado su forma de dirigirlo, dando siempre las pautas necesarias para ir avanzando, pero a la vez no fijando plazos estrictos para la realización de las tareas. Le agradezco la fluida comunicación que hemos tenido a través del correo electrónico, hecho que ha permitido organizar nuestros encuentros muy fácilmente. Siempre que he tenido dudas me las ha resuelto de forma inmediata. Finalmente, le doy las gracias por realizar y explicarme los montajes con los que he trabajado en el laboratorio.

- A **Roger Garcia Sánchez**, miembro del departamento de Ingeniería Nuclear: por acondicionar una zona del laboratorio para que pudiera trabajar allí cómodamente. Por echar una mano en todo lo que le he pedido, tanto a nivel de resolver dudas conceptuales como a nivel de material (cualquier cosa que he necesitado, Roger me lo ha traído rápidamente). Finalmente, le doy las gracias por estar pendiente de la evolución del trabajo y por aportar ideas junto con Alfredo.

- A **Mauricio Imbachi**, estudiante de doctorado que trabajaba también con la *Red Pitaya:* por darme valiosa información sobre *Red Pitaya*, basándose en su propia experiencia. Por seguir de cerca mis evoluciones (él solía estar en el mismo laboratorio ocupándose de sus tareas) y aportar sus opiniones y consejos. Gracias también por proporcionarme algunos enlaces de interés que han resultado muy útiles para la realización del trabajo.

- A **Marc Pradas**, mi hermano: por tener la idea de mezclar en un mismo código los comandos de generación de señal y los de adquisición para poder realizar muestreos de señales simples desde casa (a través del cable *SMA-SMA*).

- Al **departamento de Ingeniería Nuclear** en general: por permitirme usar sus instalaciones y su material, *Red Pitaya* incluida.

ANEXO: Código definitivo Cobalto-60 (bicanal)

```
from scipy.signal import find_peaks_cwt as peaks
from time import sleep
import redpitaya_scpi as scpi
import matplotlib.pyplot as plt

ip = '169.254.106.150'
rp_s = scpi.scpi(ip)

rp_s.tx_txt('ACQ:RST')
rp_s.tx_txt('ACQ:SOUR1:GAIN LV')
rp_s.tx_txt('ACQ:TRIG:LEV 0')
rp_s.tx_txt('ACQ:TRIG:DLY 8192')
rp_s.tx_txt('ACQ:DEC 64')

"Ventana 1,1 MeV"
lim_inf1 = 5.45
lim_sup1 =  6.2

"Ventana 1,3 MeV"
lim_inf2 = 6.201
lim_sup2 =  6.95

width = [3]

"Conteo 1,1 MeV"
count1 = 0

"Conteo 1,3 MeV"
count2 = 0

for i in range(120):
    rp_s.tx_txt('ACQ:START')
    rp_s.tx_txt('ACQ:TRIG CH1_PE')

    while 1:
        rp_s.tx_txt('ACQ:TRIG:STAT?')
        if rp_s.rx_txt() == 'TD':
            break

    rp_s.tx_txt('ACQ:SOUR1:DATA?')
```

Figura 63: *Código bicanal definitivo (I). Fuente: Propia*

```
    rp_s.tx_txt('ACQ:SOUR1:DATA?')

    buff_string = rp_s.rx_txt()
    buff_string = buff_string.strip('{}\n\r').replace("  ", "").split(',')
    buff = list(map(float, buff_string))

    ind_max = peaks(buff, width) # funcion lenta

    x_max1 = []
    y_max1 = []
    p_max1 = []

    x_max2 = []
    y_max2 = []
    p_max2 = []

    for j in range(len(ind_max)):
        ind_max[j] = int(ind_max[j])
        if lim_inf1 <= buff[ind_max[j]] <= lim_sup1:
            count1 = count1 + 1
            x_max.append(ind_max[j])
            y_max.append(buff[ind_max[j]])
            p_max.append((ind_max[j],buff[ind_max[j]]))

        elif lim_inf2 <= buff[ind_max[j]] <= lim_sup2:
            count2 = count2 + 1
            x_max2.append(ind_max[j])
            y_max2.append(buff[ind_max[j]])
            p_max2.append((ind_max[j],buff[ind_max[j]]))

    "Monitorización del conteo y valor de los máximos iteración a iteración"
    print(i)
    print(count1,count2)
    print(p_max1,p_max2)

plt.plot(buff)
plt.plot(x_max1, y_max1, "xr")
plt.plot(x_max2, y_max2, "xb")
plt.ylabel('Voltatge')
plt.grid()
plt.show()
```

Figura 64: *Código bicanal definitivo (II). Fuente: Propia*

Bibliografía

[1] DESCRIPCIÓN RED PITAYA: *https://marceluda.github.io/rp_lock-in_pid/TheApp/RedPitaya_board/*

[2] ASPECTO RED PITATA: *https://www.mouser.es/new/red-pitaya/red-pitaya-starter-kit/*

[3] PRESENTACIÓN DE LA INTERFAZ DE RED PITAYA: *https://redpitaya.readthedocs.io/en/latest/quickStart/first.html*

[4] SINGLE CHANNEL PULSE HEIGHT ANALYZERS (ORTEC): *http://122.physics.ucdavis.edu/course/cosmology/sites/default/files/files/Gamma%20Spec/PulseHeightAnalysis.pdf*

[5] CONEXIONES NECESARIAS DEL EQUIPO RED PITAYA: *https://redpitaya.readthedocs.io/en/latest/quickStart/needs.html*

[6] POSICIÓN DE LOS JUMPERS SEGÚN EL RANGO DE TENSIONES DE LA SEÑAL DE ENTRADA: *https://redpitaya.readthedocs.io/en/latest/developerGuide/125-14/fastIO.html?highlight=JUMPERS#analog-inputs*

[7] LISTA DE COMANDOS SCPI PARA CONTROLAR RED PITAYA VÍA CONTROL REMOTO: *https://redpitaya.readthedocs.io/en/latest/appsFeatures/remoteControl/SCPI_commands.html?highlight=SCPI%20COMANDS*

[8] FACTORES DE DIEZMADO CON LOS QUE PUEDE TRABAJAR RED PITAYA: *https://redpitaya.readthedocs.io/en/latest/appsFeatures/examples/acqRF-samp-and-dec.html?highlight=decimation#sampling-rate-and-decimations*

[9] CÓDIGO DE EJEMPLO DE ADQUISICIÓN CREADO POR RED PITAYA: *https://redpitaya.readthedocs.io/en/latest/appsFeatures/examples/acqRF-exm2.html*

[10] LIBRERÍA MATPLOTLIB.PYPLOT DE PYTHON: *https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html*

[11] PROCESO DE CALIBRADO DE RED PITAYA: *https://content.redpitaya.com/blog/is-your-red-pitaya-properly-calibrated*

[12] DESCRIPCIÓN DE LOS RADIOISÓTOPOS: *https://www.foronuclear.org/descubre-la-energia-nuclear/preguntas-y-respuestas/sobre-fisica-nuclear/que-son-los-radioisotopos/*

[13] DETECTOR DE BARRERA DE SUPERFICIE DE SILICIO:
https://indico.ific.uv.es/event/2645/contributions/3318/attachments/2641/2936/DetectSemiconductor2013.pdf

[14] PREAMPLIFICADOR DE LA MARCA CANBERRA:
https://www.directindustry.es/prod/canberra-industries/product-23661-555313.html

[15] (2009) COMPARISON OF PUBLIC PEAK DETECTION ALGORITHMS FOR MALDI MASS SPECTROMETRY DATA ANALYSIS:
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2631518/

[16] FILTRO DE MEDIA MÓVIL MATLAB:
https://es.mathworks.com/help/signal/examples/signal-smoothing.html

[17] DEFINICIÓN FILTRO SAVITZKY GOLAY:
http://www.statistics4u.info/fundstat_eng/cc_filter_savgolay.html

[18] FOTOGRAFÍA EJEMPLO FILTRO SAVITZKY GOLAY PARA N = 3:
http://www.statistics4u.info/fundstat_eng/cc_filter_savgolay.html

[19] FUNCIÓN SAVGOL_FILTER DE LA LIBRERÍA SCIPY.SIGNAL:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.savgol_filter.html

[20] FUNCIÓN FIND_PEAKS_CWT DE LA LIBRERIA SCIPY.SIGNAL:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks_cwt.html#scipy.signal.find_peaks_cwt

[21] FORO DE DISCUSIÓN DEL FUNCIONAMIENTO DE FIND_PEAKS_CWT:
https://stackoverflow.com/questions/31016267/peak-detection-in-python-how-does-the-scipy-signal-find-peaks-cwt-function-work

[22] PROCESO PARA EL CÁLCULO DEL PARÁMETRO WIDTH:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.peak_widths.html

[23] FUENTE DE ALTA TENSIÓN CANBERRA 3196D:
https://www.directindustry.es/prod/canberra-industries/product-23661-555310.html

[24] INFORMACIÓN SOBRE EL DETECTOR DE CENTELLEO DE NaI (Tl):
https://www.radiation-dosimetry.org/es/que-es-el-contador-de-centelleo-nai-tl-definicion/

[25] FOTOGRAFÍA DEL PREAMPLIFICADOR CANBERRA 2007P:
https://www.mirion.com/products/2007-2007p-photomultiplier-tube-base-preamplifier

[26] INFORMACIÓN SOBRE EL CESIO-137:
https://www.ld-didactic.de/software/524221es/Content/Appendix/Cs137.htm

[27] (2020) T04. Estadística aplicada a la detección y medida de la radiación. PDF

[28] INFORMACIÓN SOBRE EL AMERICIO-241: *https://iquimicas.com/americio-sirve-elemento-quimico-am/*

[29] INFORMACIÓN Y APLICACIONES DEL COBALTO-60: *https://www.lavanguardia.com/vida/20131204/54395168960/el-cobalto-60-util-para-tratamientos-medicos-pero-muy-contaminante.html*