Universidad Zaragoza
1542

instituto de investigación en ingeniería de Aragón
Universidad de Zaragoza

Escuela de Ingeniería y Arquitectura
Universidad Zaragoza

Trabajo Fin de Grado

Medidas de reflectancia en nieve mediante detección síncrona con un espectrómetro óptico

Reflectance measurement of snow using an optical spectrometer-based lock-in amplifier

Autor/es

Sara Pérez Tortajada

Director/es

Iñigo Salinas Áriz

Escuela de ingeniería y arquitectura

Julio, 2020

“Nadie sabe lo que puede hacer hasta que lo intenta”
- Publilius Syrus

Gracias a Iñigo por haberlo hecho posible,

también a Rocío y Rafa por su colaboración.

RESUMEN DEL PROYECTO FIN DE CARRERA

"MEDIDAS DE REFLECTANCIA EN NIEVE MEDIANTE DETECCIÓN SÍNCRONA CON UN ESPECTRÓMETRO ÓPTICO"

Realizado por: Sara Pérez Tortajada **Dirigido por:** Iñigo Salinas Áriz

El Grupo de Tecnologías Fotónicas (GTF) de la Universidad de Zaragoza, junto con AEMET y la CHE está llevando a cabo diversos trabajos sobre nuevas tecnologías para la medida del equivalente en agua y otros parámetros de la nieve.

El desarrollo de este proyecto está centrado en la medida de la reflectancia de la nieve. Es necesario realizar su medida no solo espectralmente sino también de la reflexión difusa, y el problema principal es la luz ambiente, por lo que habitualmente se realiza por la noche o a oscuras. Este problema puede resolverse usando detección síncrona (*lock-in*). Sin embargo, no se conoce si la realización de detección síncrona con un espectrómetro óptico será posible.

Por consiguiente, se deberá evaluar dicha posibilidad y, a partir de las conclusiones extraídas, diseñar un primer prototipo y el software de medida correspondiente.

Índice de contenidos

Índice de figuras

Índice de tablas

Apéndices

1 Introducción y objetivos

1.1. Introducción

La capa de nieve tiene al menos dos papeles principales en el calentamiento global. El primero es su alta reflectividad a la radiación solar, que disminuye la cantidad absorbida y evita una subida excesiva de la temperatura, el segundo es como buen indicador de frialdad.

Las medidas en la nieve son muy importantes, por ejemplo, para el cambio climático, la previsión de avalanchas y la regulación de caudales. Existen diferentes tipos, debido a que no es suficiente con la más simple que es la altura, sino que es necesario conocer otros datos como la densidad o el equivalente en agua (SWE), el cual describe la cantidad de agua líquida almacenada en el paquete de nieve, y es fundamental en trabajos como el pronóstico de caudales de ríos, prevención de inundaciones, el control del nivel de agua de los embalses de las centrales eléctricas, la planificación para el riego de cultivos y como variable de entrada y control para muchos fines de investigación ambiental, incluido el estudio sobre el cambio climático.

Otra medida importante es la de la reflectancia, en la cual se va a centrar el proyecto. La reflectancia o albedo se define como la relación entre la radiación reflejada en todo el hemisferio y la radiación incidente. La nieve tiene el albedo más alto de la superficie terrestre, y se puede utilizar para obtener propiedades ópticas y microfísicas de la nieve cercana a la superficie, tales como determinar el tamaño del grano de nieve (la reflectancia disminuye a medida que aumenta el tamaño del grano), el coeficiente de absorción de impurezas, etc.

Para obtener esa información necesitamos conocer no solamente la reflectancia global o promedio sino también su dependencia espectral, es decir, cómo refleja la nieve en cada longitud de onda, lo que nos permitirá analizar el color, que depende de la suciedad y otros factores.

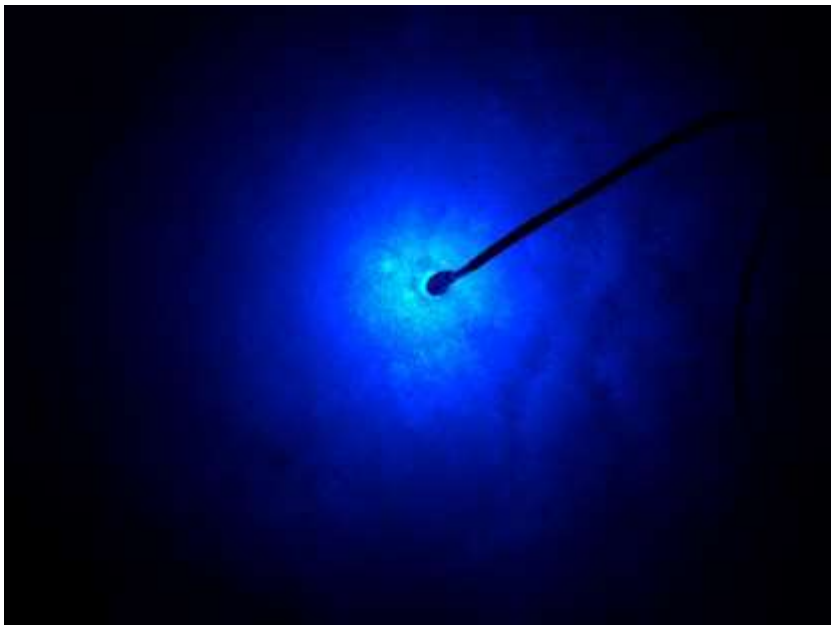


Figura 1. Reflexión difusa de la luz en la nieve

Para medir bien esa reflectancia es necesario hacerlo en todas las direcciones (no sólo la reflectancia especular sino también la difusa) porque, como se ve en la imagen previa, la cantidad de luz que se refleja en direcciones distintas a la predicha por la ley de Snell (que daría la reflexión especular) es muy grande.

La manera de realizar esa medida de reflectancia es utilizando un fotogoniómetro, es decir, un sistema con forma de círculo o semicírculo para medir en función del ángulo y que nos permitirá obtener la función de distribución de reflectancia bidireccional (BRDF) espectral.

La BRDF es una función de cuatro variables reales que define cómo la luz se refleja en una superficie opaca. Se emplea tanto en la óptica de la luz del mundo real, en algoritmos de gráficos computacionales, como en algoritmos de visión computarizada. La función toma una dirección de la luz entrante, ω_i y dirección saliente, ω_r (en un sistema de coordenadas donde la normal a la superficie, n, es el eje z), y devuelve la relación de radiancia reflejada que sale a lo largo de ω_r con la irradiancia incidente en la superficie desde la dirección ω_i. Cada dirección está parametrizada por el ángulo azimutal Φ y el ángulo zenital θ.

$$f_x(\omega_i, \omega_r) = \frac{dL_r(\omega_r)}{dE_i(\omega_i)} = \frac{dL_r(\omega_r)}{L_i(\omega_i) * \cos\theta_i \, d\omega_i}$$

Donde L es radiancia, o potencia por unidad de ángulo sólido en la dirección de un rayo por unidad de área proyectada perpendicular al rayo, E es la irradiancia, o potencia por unidad de superficie, y θ_i es el ángulo entre ω_i y la superficie normal, n. El índice i indica luz incidente, mientras que el índice r indica luz reflejada.

La BRDF tiene como unidades sr-1, con estereorradianes (sr) como unidad de ángulo sólido.

Uno de los principales problemas de esta medida es la luz ambiente, por lo que para evitarla se realiza habitualmente por la noche o a oscuras. Este problema, sin embargo, puede solucionarse utilizando una fuente de luz modulada y detección síncrona (*lock-in*).

El objetivo final del proyecto global es construir un equipo para medir en varios ángulos el espectro de la reflectancia de una superficie, para así obtener su BRDF espectral en tiempo real. Este trabajo se centrará en el desarrollo de la **medida espectral en tiempo real y con detección síncrona**, quedando para trabajos futuros el desarrollo del equipo completo de medida de BRDF, con un sistema mecánico que permita realizar la medida de reflectancia para distintos ángulos.

Figura 2. Esquema fotogoniómetro

El equipo desarrollado en este proyecto iría incluido en la caja, la cual se movería alrededor del arco.

1.2. Antecedentes

El carbono puede provocar retroalimentaciones de albedo y acelerar el deshielo, su aumento debido a la revolución industrial ha llevado al reconocimiento de que el forzamiento radiativo (RF), es decir, la diferencia entre la insolación (luz solar) absorbida por la Tierra y la energía irradiada de vuelta al espacio, por partículas absorbentes de luz (LAP) ha contribuido a una reducción en la criósfera global, término que describe las partes de la superficie terrestre donde el agua se encuentra en estado sólido.

La nieve, un tipo de cubierta terrestre espacialmente extensa que se concentra en las altas latitudes y/o ambientes de alta elevación, ejerce fuerza en el equilibrio energético global principalmente a través de su alto albedo. Los registros de los satélites y de observaciones terrestres a largo plazo muestran que la masa y la extensión de la capa de nieve están disminuyendo, y esto es en gran parte atribuido a un clima más cálido.

Sin embargo, la duración de la nieve sufre mayor impacto por el oscurecimiento de la superficie por LAP, lo cual aumenta el crecimiento del grano y eso reduce aún más el albedo de la nieve. En consecuencia, el oscurecimiento de la nieve por partículas absorbentes de luz tiene implicaciones climáticas a corto y largo plazo [1].

Por otro lado, el albedo juega un papel importante en el clima, al regular la cantidad de radiación de onda corta reflejada o absorbida y posteriormente reradiada como radiación de onda larga por una superficie. Los cambios de albedo en la nieve están relacionados con la dispersión de la luz, que es causada por cambios en la densidad de empaquetamiento de partículas y materiales extraños como polvo en la nieve. Estos factores afectan la velocidad de fusión y pueden ser un indicador de nieve o equivalente en agua [2].

Para comprender la correlación entre la reflectancia y las propiedades físicas de la nieve necesitamos conocer la correlación entre la reflectividad y su textura y estructura, además de en algunas ocasiones también sus propiedades físicas, las cuales han sido investigadas numerosas veces. Se realizaron dos experimentos en los que hubo una disminución significativa en la reflectancia espectral de la nieve debido a la metamorfosis, ellos mismos confirmaron que los tamaños de grano más pequeños tienen reflectancias más grandes, y también se encontró que el metamorfismo se

observó solo en la capa más alta, por lo que la reflectancia de la nieve se ve fuertemente afectada por la textura de la capa superior [3].

Estos hechos hacen conveniente disponer de un equipo portátil y fácil de utilizar en campo que permita medir la reflectancia de la nieve en distintas situaciones y condiciones. Para ello se dispone principalmente de dos herramientas que habrá que estudiar y hacer funcionar de manera conjunta en este proyecto: el amplificador *lock-in* y el espectrómetro compacto.

1.2.1. El amplificador *lock-in*

Un circuito de detección síncrona o amplificador *lock-in* se caracteriza esencialmente por medir un voltaje alterno de entrada y generar una salida continua proporcional a la amplitud de la señal alterna que está siendo medida. Se trata de un amplificador porque generalmente el nivel de salida DC es mayor que el nivel de entrada AC, y es síncrono porque captura y mide la señal a una determinada frecuencia, ignorando el resto de las componentes espectrales [4].

Consiste básicamente en un demodulador que detecta la señal modulada a una frecuencia de referencia que debe ser suministrada al *lock-in* junto con la señal a medir. La señal de interés puede ser modulada externa o internamente. El amplificador *lock-in* actúa por tanto como un filtro paso banda estrecho, que selecciona la señal cuya frecuencia es la de referencia eliminando el resto de las componentes. De esta forma mejora considerablemente la relación señal a ruido, y por ello su aplicación esencial es la detección y medida de señales de pequeña amplitud inmersas en ruido.

En el caso del reflectómetro presentado en este proyecto, el amplificador permitirá medir la reflectancia de la nieve en condiciones de fuerte luz ambiente, sin necesidad de hacer la medida de noche o en una caja cerrada, como ocurre con otros equipos.

1.2.2. El espectrómetro compacto

Los avances tecnológicos de los últimos años han permitido diseñar equipos para realizar medidas espectrales con un tamaño y consumo muy inferior a los de los tradicionales equipos de laboratorio.

Como equipo de laboratorio podemos destacar el espectrofotómetro UV / Vis LAMBDA 850+, que cumple con los estándares de la industria para un rendimiento, flexibilidad y conveniencia ultra altos. Su rango de operación es para longitudes de onda entre 175 y 900 nm. Sin embargo, posee unas grandes dimensiones (102 x 30 cm) y un peso elevado (77 Kg), por lo que no es portátil.

Figura 3. Espectrofotómetro UV / Vis LAMBDA 850+

Entre los equipos compactos, se pueden citar:

Figura 4. Espectrómetro Compact CCD CCS175

El espectrómetro CCD CCS175, del cual existen 3 modelos con diferentes rangos de longitudes de onda: 350 – 700 nm, 500 – 1000 nm, 200 – 1000 nm. Sus dimensiones son 122 x 79 x 29.5 mm.

Figura 5. Mini espectrómetro serie MS C10988MA-01

La serie MS son cabezales de espectrómetro del tamaño de un pulgar (27.6 x 13 x 16.8 mm), desarrollados para su instalación en equipos de medición móviles. Su rango de respuesta espectral es de 340 a 750 nm.

Figura 6. Espectrómetros Broadcom Qmini AFBR-S20M2WV

Este espectrómetro en miniatura viene con un exhaustivo procesamiento y evaluación para aplicaciones móviles y soluciones integradas. Además, cubre un amplio rango de longitudes de onda desde 225 a 1000 nm. Es el que vamos a utilizar en nuestro proyecto.

1.3. Motivación

A partir de las experiencias previas y lo visto en la bibliografía, se considera interesante tener un equipo para medir la BRDF espectral de la nieve.

Para cumplir los requisitos que nos planteamos, de poder medir con luz ambiente, es imprescindible conseguir hacer detección síncrona con el espectrómetro, es decir, de todos los pixeles del espectro que vayamos a usar. Sin embargo, esto es algo que nunca antes ha sido probado, por lo menos en el Grupo de Tecnologías Fotónicas (GTF) de la Universidad de Zaragoza y no tenemos constancia de que haya sido experimentado por otras universidades o grupos de investigación. Por lo tanto, se trata de una solución que no se sabía si iba a funcionar, ya que la velocidad de adquisición del espectrómetro está muy limitada, entonces suponía un gran riesgo para el proyecto.

Este desarrollo tiene como marco el trabajo que está realizando el Grupo de Tecnologías Fotónicas de la Universidad de Zaragoza, junto con AEMET y la CHE sobre nuevas tecnologías para la medida del equivalente en agua y otros parámetros de la nieve.

Se están llevando a cabo pruebas de diversos sistemas en la instalación experimental de AEMET en la estación de Formigal. Por ejemplo, se ha instalado un sistema para medida del equivalente en agua basado en un radar SFCW (Stepped-Frequency Continuous-wave), ya operativo y cuya fiabilidad se ha verificado tras comparaciones con el telenivómetro N014 de la CHE y con medidas experimentales con los modelos electromagnéticos de la estructura multicapa del manto nivoso. Este sistema se muestra en las siguientes fotografías:

Figura 7. Instalación experimental en la estación de Formigal

También está comenzando el desarrollo de un equipo para la medida del tamaño y la velocidad de las gotas de lluvia a partir del efecto Doppler, utilizando un pluviómetro radar de emisión vertical a 24 GHz. Este estudio pretende obtener información básica del espectro Doppler, su relación con el tipo de precipitación y su intensidad.

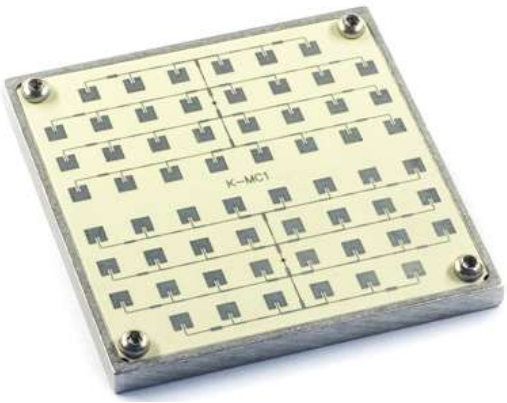


Figura 8. Radar para pluviómetro Doppler

1.4. Objetivos

El objetivo principal de este trabajo fin de carrera es evaluar la posibilidad de construir un sistema para medida en campo de la reflectancia espectral de la nieve en condiciones de fuerte luz ambiente, utilizando para ello detección síncrona con un espectrómetro compacto.

En primer lugar, se evaluará la posibilidad de realizar detección síncrona espectral utilizando un espectrómetro compacto. Será necesario establecer los límites y capacidades del sistema.

Posteriormente, a partir de las conclusiones del estudio previo, se realizará el diseño de un prototipo de medida de reflectancia utilizando detección síncrona espectral y se desarrollará el software para utilizar dicho prototipo.

También se diseñará una placa de circuito impreso para modular correctamente la fuente de luz del prototipo, o sea de un LED.

1.5. Estructura de la memoria

La estructura de la memoria está organizada en 7 capítulos, siendo el primero Introducción y objetivos, el cual acabamos de realizar.

En el segundo capítulo se presenta una breve descripción del estado del arte. A continuación, en el tercero, Materiales y métodos, se explican más en profundidad los procedimientos utilizados.

En el capítulo 4, Desarrollo e implementación, se expone el primer prototipo y seguidamente, en el capítulo 5, se muestran los resultados.

El sexto contiene un presupuesto aproximado de un equipo portable con el que se podría realizar medidas en nieve.

Por último, en el capítulo 7, se incluyen las conclusiones obtenidas, así como posibles líneas de trabajos futuro.

Al final de la memoria se incorporan una serie de anexos que completan el trabajo realizado.

2 Estado del arte

Anteriormente ya hemos nombrado la importancia del albedo de la nieve y como influía el tamaño del grano en la reflectancia. Hay diferentes trabajos que tratan sobre esto.

Entre ellos, existe alguno más teórico [5], en el que se propone una secuencia de ecuaciones analíticas que se pueden usar para recuperar el tamaño de los granos y el coeficiente de absorción de los contaminantes de la reflectancia de la nieve o las mediciones de albedo de la nieve en regiones visibles e infrarrojas cercanas al espectro electromagnético de la nieve. Esta teoría es validada utilizando mediciones de reflectancia espectral y albedo de nieve limpia y contaminada en varios lugares. Se plantea una técnica para derivar el albedo de la nieve a partir de mediciones de reflectancia en una geometría de observación fija. Las mediciones espectrales de la nieve se realizaron utilizando un espectrómetro de campo, este instrumento presenta un rango espectral de 350 – 2500 nm. Los datos conseguidos se recogieron bajo condiciones de poca claridad al mediodía.

Por otro lado, tenemos varios trabajos que han hecho frente a este problema de manera más práctica, como los que vemos a continuación.

La estratigrafía del tamaño del grano es crítica para la mecánica, propiedades termodinámicas y radiativas de la capa de nieve, además de para el desarrollo de técnicas de inversión directa para la estimación del equivalente de agua en nieve a partir de teledetección de microondas y para estimar el albedo y la fracción de área cubierta de nieve.

Las mediciones tradicionales usando una lente manual no son muy precisas, ya que carecen de repetibilidad. Sin embargo, se demuestra una mejora significativa en el modelado del albedo espectral y flujo neto de onda corta con mediciones de espectroscopía de contacto. La técnica empleada combina una sonda de contacto de dispositivos analíticos espectrales (ASD) modificada con un espectrorradiómetro de campo (ASD FieldSpec FR), cuyo rango de longitud de onda es de 350 – 2500 nm. La sonda de contacto dispone de una fuente de luz estable integrada con el soporte para el cable óptico del espectrorradiómetro, la cual está alineada a 23° con el cuerpo de la sonda y con ello evitamos los problemas de iluminación variable desde la topografía de la superficie, por lo que puede implementarse en cualquier momento del día y permite una rápida adquisición [6].

La determinación del área de superficie específica de nieve a partir de mediciones de reflectancia en el dominio infrarrojo cercano representa una técnica prometedora para adquirir rápida y cuantitativamente perfiles estratigráficos de la nieve. Los resultados muestran claramente que la relación depende de la forma del grano, los cúbicos reflejan alrededor del 40% más que los esféricos. Hasta ahora, todas las técnicas requieren de mucho tiempo, además de que las muestras deben ser transportadas a un laboratorio, lo que limita el número de muestras que pueden tomarse.

La base teórica para medir la superficie específica área (SSA), es decir, el área de superficie total de la interfaz aire / hielo por unidad de masa, utilizando la reflexión del infrarrojo cercano, ya sea capturado por fotografía, un espectrómetro u otros instrumentos, nace de la idea de representar la nieve mediante una colección de esferas con la misma relación volumen / superficie (V/A) que los granos de nieve, con ello se pretendía acelerar los cálculos de albedo de nieve. Sin embargo, la equivalencia V/A no es estrictamente válida cuando se consideran diferentes formas de grano. Para tratar este problema, se desarrolla un modelo óptico que simula en detalle la trayectoria de la luz desde la fuente, a través de la muestra de nieve y hacía el detector, este enfoque se conoce como trazado de rayos. Se calcula para diferentes formas de grano y se concluye que, para un determinado

albedo, SSA puede variar en un rango de ± 20% [7].

3 Materiales y métodos

3.1. Amplificador *lock-in*

En esta sección describiremos las ecuaciones que describen el funcionamiento de un amplificador *lock-in*, para comprender cómo trabaja y porque discrimina las señales cuya frecuencia se halla en un estrecho entorno de la frecuencia de referencia.

Asimismo, se trata de uno de los elementos clave del equipo, que como hemos mencionado en la introducción nos permitirá medir en condiciones de luz ambiente.

Existen de dos tipos, de una fase y de doble fase. En el primer caso (ver Figura 9), el amplificador *lock-in* recibe una señal de entrada de una frecuencia f_s, y precisa una señal de referencia de esta misma frecuencia, se multiplican ambas señales pudiéndose expresar el producto como una serie de armónicos. La contribución en DC es proporcional a la señal de interés, dicha componente puede extraerse mediante un filtro paso bajo (LPF). La salida del sistema es, por lo tanto, una señal continua proporcional a la amplitud de la señal de interés.

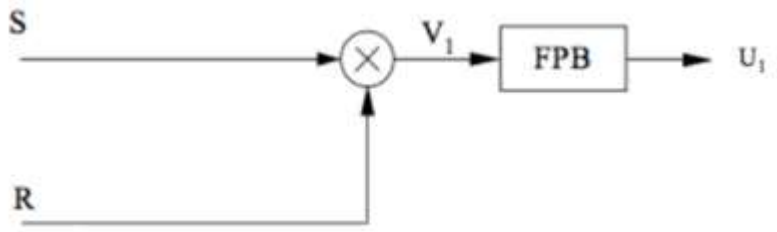


Figura 9. Esquema del algoritmo lock-in de una fase

Nos centraremos en el amplificador *lock-in* de fase doble, como el de la Figura 10, donde R es una señal de referencia, cuya frecuencia principal es f_r y S es la señal de interés con frecuencia f_s, amplitud V_s y desfase θ_s.

$$S = V_s * \sin(2\pi f_s t + \theta_s)$$

De la señal de referencia R generamos dos señales de referencia internas I y Q, con la misma frecuencia y fase, desfasadas $\pi/2$ radianes una respecto a la otra:

$$I = V_{iq} * \sin(2\pi f_r t + \theta_r)$$

$$Q = V_{iq} * \sin(2\pi f_r t + \theta_r + \pi/2)$$

Multiplicando la señal de referencia I por la de interés obtenemos la señal V_1, y al multiplicar la señal en cuadratura Q por la de interés se obtiene V_2.

$$V_1 = S * I = V_s * V_{iq} * \sin(\omega_s t + \theta_s) * \sin(\omega_r t + \theta_r)$$

$$V_2 = S * Q = V_s * V_{iq} * \sin(\omega_s t + \theta_s) * \sin(\omega_r t + \theta_r + \pi/2)$$

Aplicando la siguiente relación trigonométrica:

$$\sin\alpha * \sin\beta = \frac{1}{2} * [\cos(\alpha-\beta) - \cos(\alpha+\beta)]$$

Las ecuaciones anteriores se convierten en:

$$V_1 = \frac{V_s * V_{iq}}{2} * \{\cos[(\omega_s - \omega_r) * t + (\theta_s - \theta_r)] - \cos[(\omega_s + \omega_r) * t + (\theta_s + \theta_r)]\}$$

$$V_2 = \frac{V_s * V_{iq}}{2} * \{\cos[(\omega_s - \omega_r) * t + (\theta_s - \theta_r - \pi/2)] - \cos[(\omega_s + \omega_r) * t + (\theta_s + \theta_r + \pi/2)]\}$$

Como se observa aparecen dos componentes sinusoidales cuyas frecuencias son la suma y la resta de las frecuencias de la señal de entrada y de la de referencia, respectivamente.

De esta forma, si las frecuencias de la señal de entrada f_s y la señal de referencia f_r son iguales, al hacer el producto aparecerán componentes de continua proporcionales a la amplitud de la señal de entrada, la amplitud de las señales de referencia internas y también al desfase de la señal de entrada con estas últimas:

$$V_1 = \frac{V_s * V_{iq}}{2} * \{\cos(\theta_s - \theta_r) - \cos[2 * \omega_r * t + (\theta_s + \theta_r)]\}$$

$$V_2 = \frac{V_s * V_{iq}}{2} * \{\cos(\theta_s - \theta_r - \pi/2) - \cos[2 * \omega_r * t + (\theta_s + \theta_r + \pi/2)]\}$$

Para poder separar estas componentes, ambos productos se filtran con filtros paso bajo muy estrechos que únicamente permitirán pasar la componente continua, de manera que a la salida sólo se obtienen las componentes de la señal de interés, eliminando cualquier aportación de ruido o señales cuya frecuencia sea distinta a la de referencia. Las señales proporcionadas por ambos filtros corresponden a las componentes en fase y en cuadratura de la señal de interés, U_1 y U_2:

$$U_1 = \frac{V_s * V_{iq}}{2} * \cos(\theta_s - \theta_r)$$

$$U_2 = \frac{V_s * V_{iq}}{2} * \cos(\theta_s - \theta_r - \pi/2)$$

De donde se obtendrá la magnitud de la señal medida a la frecuencia de referencia, M:

$$M = \sqrt{{U_1}^2 + {U_2}^2} = \frac{V_s * V_{iq}}{2} * \sqrt{[\cos(\theta_s - \theta_r)]^2 + [\sin(\theta_s - \theta_r)]^2} = \frac{V_s * V_{iq}}{2}$$

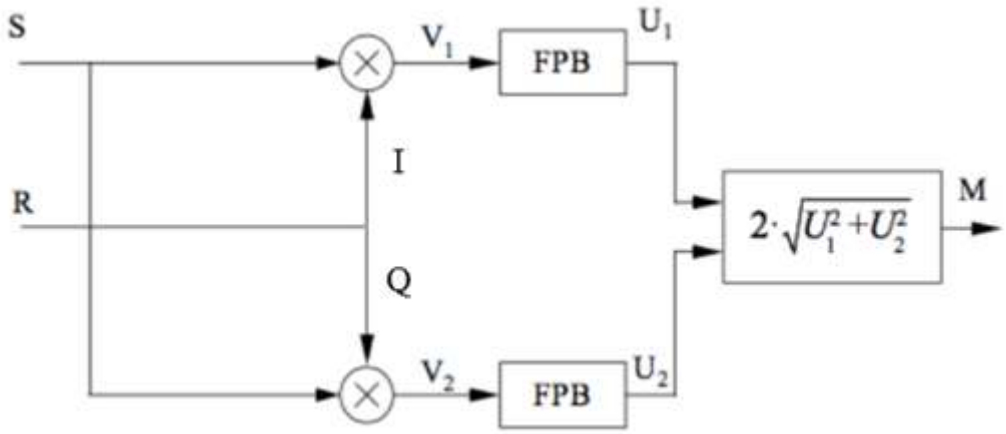


Figura 10. Esquema del algoritmo lock-in de doble fase

Resumiendo, el amplificador *lock-in* se comporta como un filtro paso banda centrado en la frecuencia de referencia f_r y con un ancho de banda equivalente al filtro paso bajo utilizado para filtrar los productos. Por lo tanto, el valor de la señal a ruido a la salida del amplificador *lock-in* mejorará notablemente respecto a la de la entrada del mismo, dependiendo de la anchura del filtro.

Por tanto, cuanto más estrecho sea el filtro mayor será la SNR a la salida, pero también implicará un mayor tiempo de estabilización de la medida, por lo que existe un compromiso entre ambos valores.

Esto nos permite evitar el efecto de la luz ambiente, debido a que solo vamos a detectar la señal que está modulada a la frecuencia elegida. La luz ambiente es básicamente continua, es decir, frecuencia cero, la cual será distinta de la frecuencia de modulación.

Sin embargo, podría existir un problema cuando dispongamos de demasiada luz ambiente, en cuyo caso el detector se saturaría y el *lock-in* no sería capaz de filtrar.

3.2 Espectrómetro compacto

Un espectrómetro es un dispositivo utilizado para analizar la composición espectral de la luz. Su funcionamiento consiste en dispersar la luz blanca de una fuente de luz a partir de un prisma o rejilla de difracción en una serie de longitudes de onda discretas, también conocidas como espectro cromático.

La cámara interior del espectrómetro contiene un conjunto de componentes ópticos que forman un banco óptico. Un haz de luz es guiado hacia el banco óptico mediante una ranura muy estrecha, la cual se utiliza para controlar la luz que ingresa en él. Posteriormente, el haz rebota en varios componentes hasta alcanzar el fotodetector en forma de espectro cromático.

Existen diferentes configuraciones de bancos ópticos, una de las más usadas es la “Czerny-Turner” (ver Figura 11), en la cual se basa nuestro espectrómetro.

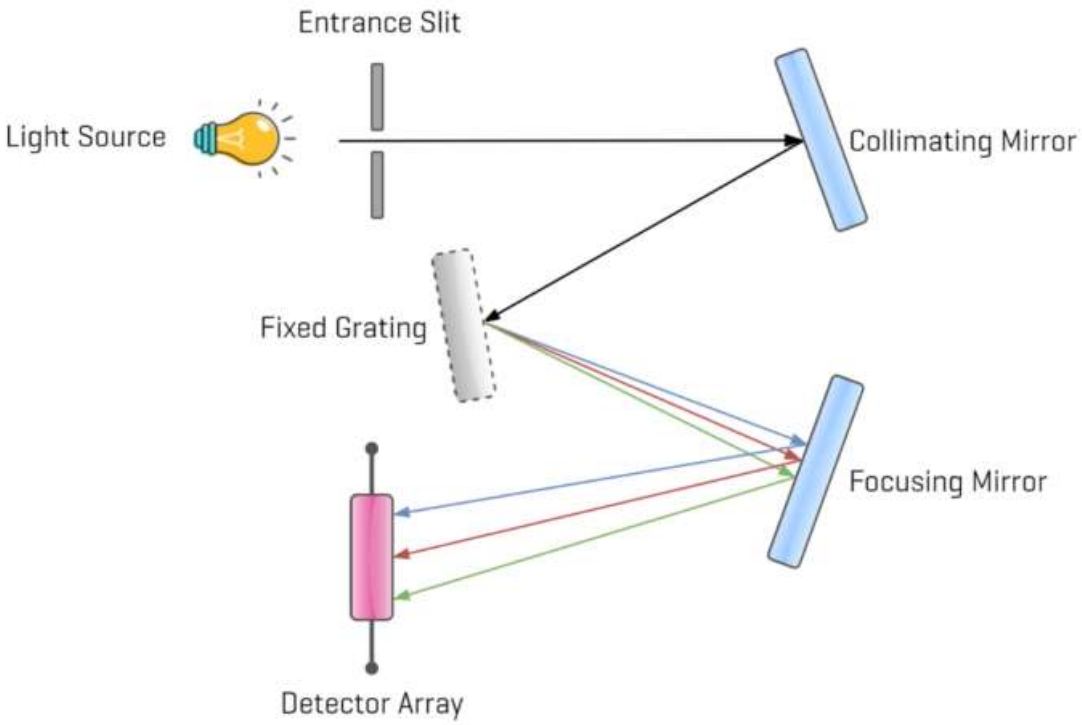


Figura 11. Diagrama de bloques de una configuración típica de un espectrómetro "Czerny-Turner"

Como se aprecia en el diagrama, dicha configuración consta de dos espejos cóncavos y una rejilla de difracción plana. El primer espejo es colocado de manera que colima la luz emitida desde la ranura de entrada y dirige el haz hacia la rejilla de difracción, que separa la luz en sus componentes cromáticos. Seguidamente, el segundo espejo se utiliza para enfocar la luz dispersa hacia los detectores.

4 Desarrollo e implementación

4.1. Requisitos del diseño

Lo más significativo de nuestro diseño es que debe ser portable, para transportarlo con facilidad y poder realizar medidas en la nieve. Teniendo en cuenta eso, nos disponemos a buscar un ordenador compacto, un espectrómetro pequeño, además de una forma sencilla de alimentación.

En el caso del PC, elegimos el LattePanda Alpha 800s, que además incluye un microprocesador Arduino totalmente funcional.

Como espectrómetro usaremos el Broadcom Qmini AFBR-S20M2WV con rango de 225 a 1000nm, el cual ya ha sido probado y usado en proyectos anteriores, por lo que su buen funcionamiento esta corroborado.

Figura 12. LattePanda Alpha 800s y espectrómetros Broadcom Qmini AFBR-S20M2WV, con rango de 225 a 1000nm

Sus hojas de características están adjuntas en el Anexo A.

En cuanto a la forma de alimentación, decidimos utilizar tanto para el circuito de la placa como el ordenador una fuente externa. Una de las soluciones posibles es la siguiente:

Figura 13. Power bank Powergorilla

Consta de 2 salidas, y una tensión que puede variar desde los 5 a los 24 voltios, por lo que nos sirve para alimentar ambos dispositivos, ya que para el circuito simplemente necesitamos 5 V, y para el Lattepanda 12 V.

4.2. Primer prototipo

El primer prototipo se controla mediante el PC compacto y el Arduino.

El prototipo también dispone del espectrómetro, es decir, un instrumento óptico usado para medir las propiedades de la luz sobre una porción específica del espectro electromagnético. En nuestro caso lo utilizaremos para medir la reflectancia.

Para esta parte no será necesario el uso de la fuente de alimentación externa, ya que como fuente de luz tendremos un LED alimentado con el Arduino y el PC lo enchufamos directamente a la corriente eléctrica.

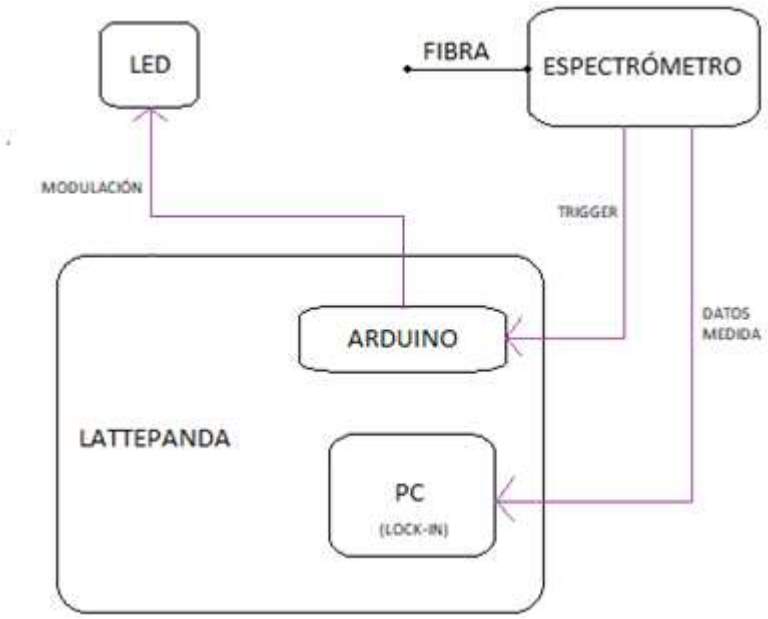


Figura 14. Diagrama de bloques primer prototipo

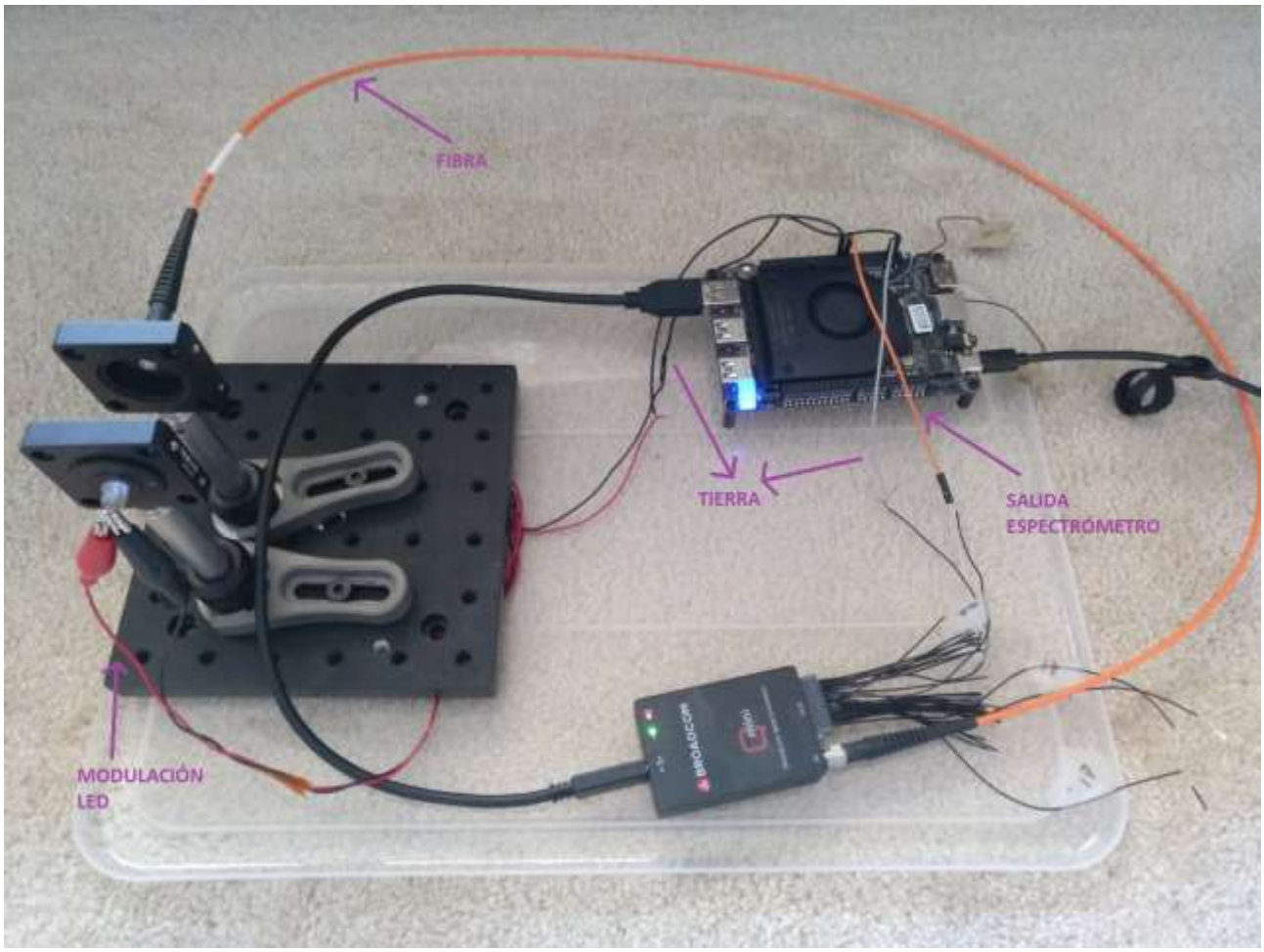


Figura 15. Primer prototipo de laboratorio

Para poder realizar detección síncrona necesitaremos modular la fuente de luz (en este caso un LED) y conocer la frecuencia y fase de esa señal de modulación para poder usarla como referencia.

Para conseguir esto, el ordenador es quien señala al espectrómetro cuando debe medir, controlado por un tiempo de integración que muestra cuando debe empezar; posteriormente el PC se queda esperando hasta que acaba, y es entonces cuando manda otra vez la orden de medida. Este proceso tiene la suficiente regularidad para mantener un periodo constante, como puede verse en la imagen del osciloscopio (ver Figura 16).

Lo que se muestra en la parte de abajo es la señal que genera el espectrómetro durante el tiempo de medida, a una frecuencia de 40 Hz, y en la parte de arriba tenemos la modulación del LED, a una frecuencia de 3.33 Hz.

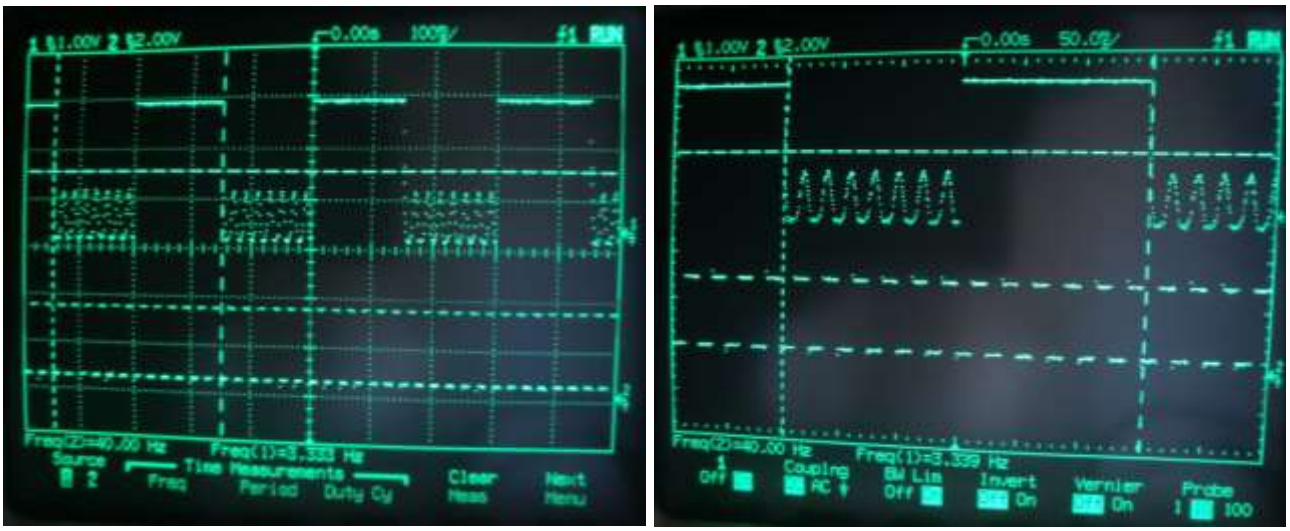


Figura 16. Capturas del osciloscopio

En las siguientes capturas, podemos apreciar que la señal de la modulación del LED, en la parte que esta abajo, es un poco ruidosa, creemos que es debido a que no tiene la suficiente corriente.

Esa señal del espectrómetro se conecta a una entrada digital del Arduino para poder generar una señal de modulación para el LED sincronizada con la adquisición y, por tanto, con el contador del ordenador. El periodo de modulación será un múltiplo del de adquisición o muestreo, según un factor que se fija en el programa de Arduino.

Cada medida del espectrómetro devuelve un vector de 2500 valores, correspondientes a las 2500 longitudes de onda del espectro que se miden en cada adquisición. Cada una de esas longitudes de onda será sometida a su correspondiente procesado *lock-in* aunque, como no es necesaria una resolución tan alta, podemos quedarnos con menos medidas (suele ser suficiente con 100, que da una resolución de 8 nanómetros).

En cuanto al filtro, estamos utilizando un Butterworth de orden 4 y con una frecuencia de corte de 0.53 Hz.

Si seleccionamos uno de los procesados que se están realizando simultáneamente, podemos verificar el correcto funcionamiento del *lock-in* (ver Figura 17). Se observa cómo, tras las oscilaciones iniciales debidas a la convergencia del filtro, el módulo del *lock-in* alcanza un valor estable correspondiente a la amplitud de la señal modulada para esa longitud de onda.

La medida es mucho más estable cuando la selección del número de pixeles se realiza con promediado.

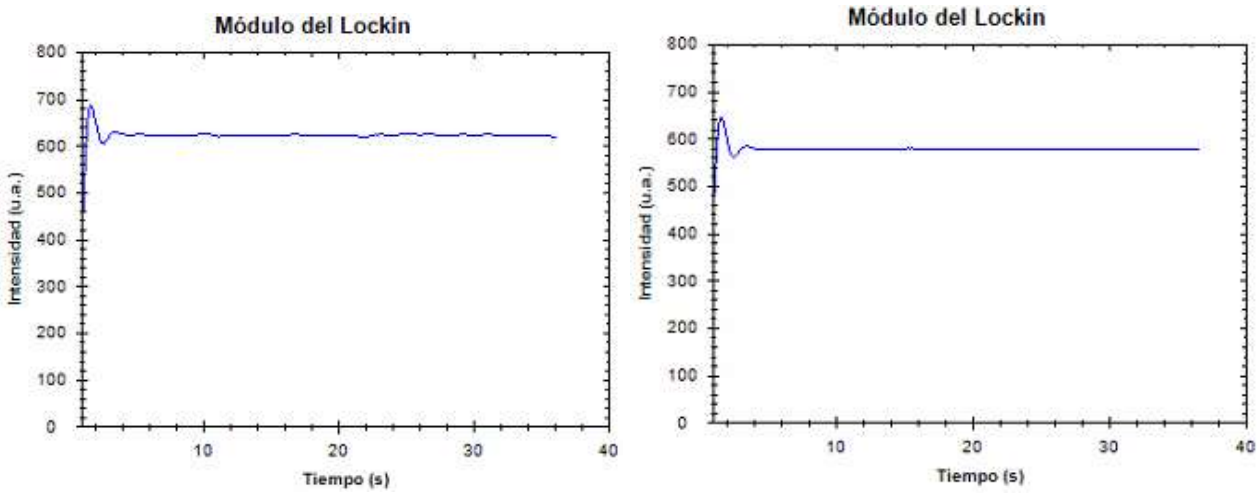


Figura 17. Módulo del lock-in sin y con promediado

En la Figura 17 se muestra el *lock-in* 17, que corresponde con el máximo del espectro del LED blanco (460 nm aproximadamente).

Después, modificamos el programa para obtener el espectro con varios puntos, consiguiendo así una de las primeras pruebas (ver Figura 18), se trata del espectro de un LED rojo, el cual se mantiene muy estable. Esta medida fue tomada con 100 puntos de los 2500 que pixeles que contiene el espectrómetro.

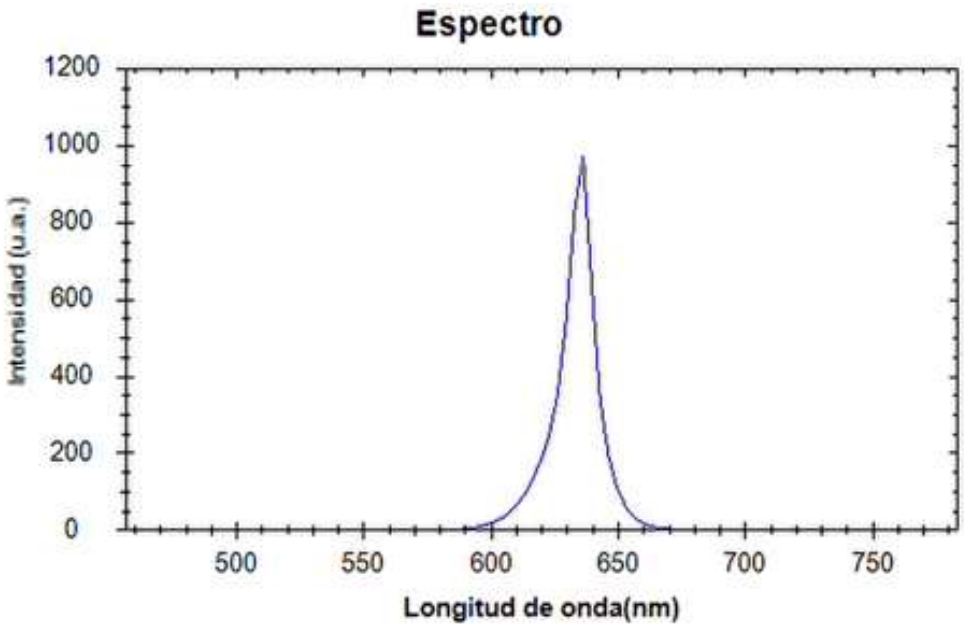


Figura 18. Espectro de un LED rojo, con 100 puntos

A continuación, obtenemos el espectro, cambiando a un LED blanco (ver Figura 19). Hay que tener en cuenta que los LED blancos son fabricados con un LED azul y con un fósforo que emite en amarillo, por eso en el espectro podemos ver el pico grande que se corresponde al azul, y seguidamente una pequeña montaña que se corresponde con el amarillo. Tenemos menor cantidad de luz que, en el caso anterior, pero está distribuida en todo el espectro visible.

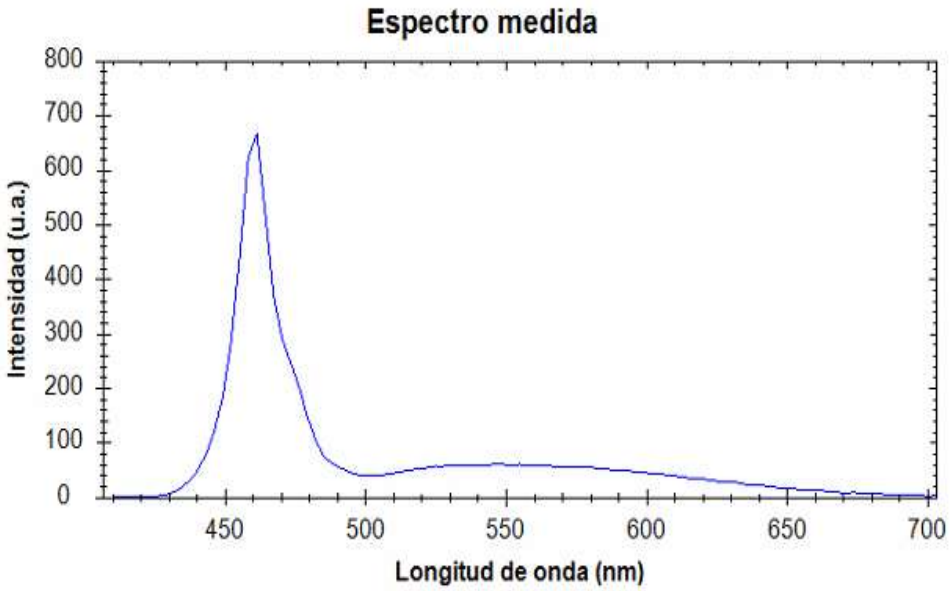


Figura 19. Espectro LED blanco

Debido a que el LED es alimentado directamente con el Arduino no disponemos una cantidad de luz adecuada para poder realizar las primeras pruebas en reflexión, por lo que medimos en transmisión, posteriormente solucionaremos este problema alimentando bien el LED en una placa.

Además, hemos corroborado que, al introducir luz externa, tanto en el caso de exponerlo a la luz del sol como de iluminarlo con luz artificial, por ejemplo, con el LED blanco del móvil, la medida no se ve afectada, lo cual era de esperar, ya que se trata de la ventaja que nos ofrece el *lock-in*.

Se verifica que hemos sido capaces de hacer el procesado lock-in con el espectrómetro.

4.3. Placa de modulación de LED

Como ya hemos mencionado, en el primer prototipo estábamos alimentando el LED directamente con el Arduino, con lo cual conseguíamos poca luz. Por ello, procedemos a diseñar una placa a través del programa EAGLE, que nos va a permitir independizar la modulación y la alimentación del LED.

A continuación, se muestra (Figura 20) el esquemático en el cual definimos el circuito. Este circuito ha sido diseñado por el GTF y ha sido probado y utilizado en otros proyectos para diferentes aplicaciones. Está basado en dos etapas de transistores bipolares (BJT) en cascada, que constituyen una configuración que funciona como interruptor. Su control reside en la señal digital MOD1, generada por el Arduino y utilizada para modular el LED, de tal manera que se iluminará cuando la señal digital introducida se encuentre a nivel alto, según la frecuencia de modulación establecida.

Asimismo, la colocación de los BJT nos asegura una corriente de 1.4 voltios en el punto Q1B, de tal manera que la corriente que circula por el LED es controlada por la resistencia R3. Además, existe una conversión de voltaje a intensidad a través de la resistencia R2.

Por otro lado, el circuito requiere de una alimentación de 5 voltios de tensión, filtrada a través de un condensador de desacoplo C1, por seguridad.

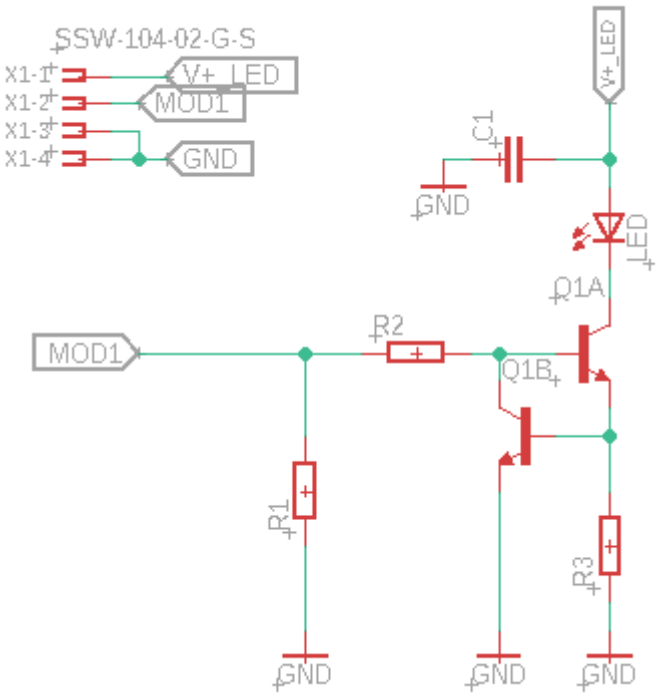


Figura 20. Circuito para modular y alimentar el LED

A partir de ahí, se crea la propia placa para su fabricación, considerando que encaje en los soportes habituales para lentes de una pulgada de diámetro, se realiza el diseño de forma circular, cumpliendo con las medidas adecuadas y colocando el LED en el centro, lo que conlleva posicionar los demás componentes alrededor, quedando tras haber definido las pistas de la siguiente forma:

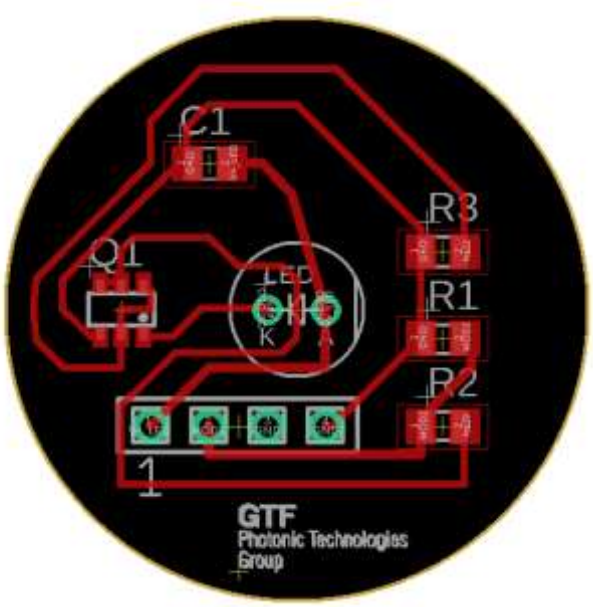


Figura 21. Placa de modulación de LED

4.4. Programación y aplicación

En los apartados anteriores hemos explicado los distintos aspectos de hardware, en este caso se aborda el desarrollo del software realizado en cuanto a la programación del microprocesador Arduino y la aplicación de Visual Studio mediante lenguaje C#.

4.4.1. Programación Arduino

El Arduino integrado en el LattePanda se encarga de generar la señal de modulación.

Para ello, consta de una entrada digital conectada al espectrómetro, en la cual está esperando que ocurra una interrupción, en nuestro caso de trata de un cambio de estado, debido a que empleamos el modo CHANGE.

```
attachInterrupt( digitalPinToInterrupt(2), ISR_muestreo, CHANGE);
```

Cuando esto sucede, entra a la subrutina de interrupción, en ella hemos implementado un contador, con el que obtenemos el número de medidas que van llegando, una vez alcanzado un valor determinado, se produce un cambio de estado en la señal de salida, de manera que se genera la señal modulación con un periodo mayor, por lo que será más lenta.

```
void ISR_muestreo()
{
     contador++;
   if (contador==factor)
     {
          contador=0;
          modulacion=!modulacion;
          if (modulacion)
               digitalWrite(1, HIGH);
          else
               digitalWrite(1, LOW);
     }
}
```

Donde el factor puede modificarse a través del monitor serie, inicialmente vale 12, quedando de la siguiente forma:

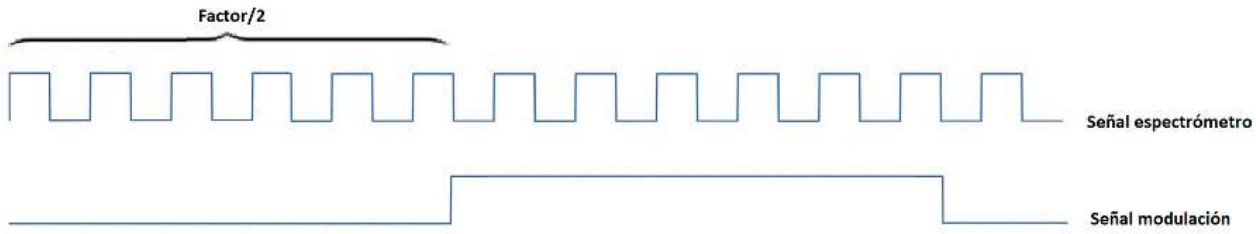


Figura 22. Señales salida del espectrómetro y salida del Arduino (modulación))

Con lo que conseguimos una sincronización entre la modulación y el PC, de forma que el ordenador es el que va mandando la adquisición, y como el Arduino sincroniza la modulación con la adquisición, al final la modulación esta sincronizada con el ordenador.

Por lo tanto, la modulación depende del PC, de cuando éste mide, y con el Arduino simplemente aumentamos el periodo en el factor indicado, ya que, si la modulación fuera directamente de la salida del espectrómetro, tendríamos simplemente una muestra por periodo, de modo que cada vez que el ordenador midiera, en la modulación se produciría un cambio.

El código completo queda descrito en el Anexo B.

4.4.2. Aplicación Visual Studio

El programa comprendido en el LattePanda es el encargado del control del equipo y los parámetros de configuración, además del manejo de datos, lo que permite su adquisición, guardado y representación.

El código se puede consultar en el Anexo C.

Estructura

El programa cuenta con una estructura de clases y objetos jerárquica, con el objetivo de poder diferenciar cada parte y comprenderlo de forma sencilla.

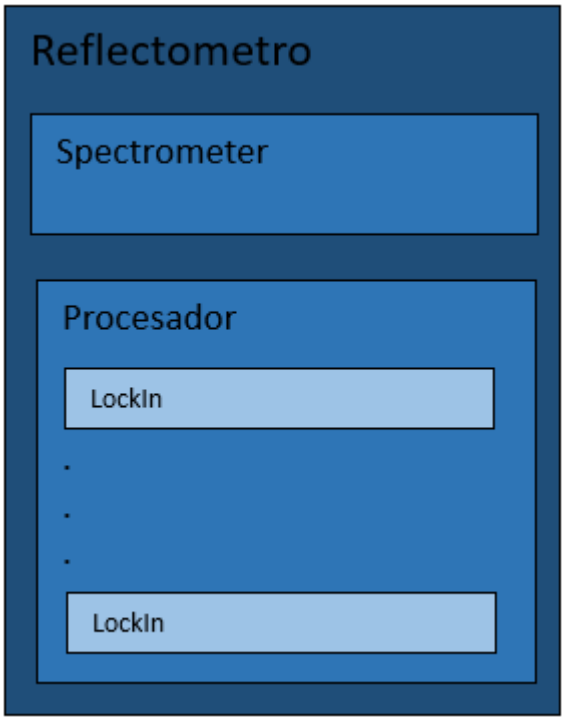


Figura 23. Estructura de clases de la aplicación

La Figura 23 recoge el conjunto de clases del proyecto junto con su esquema de encapsulado.

La clase fundamental del programa es *Reflectometro*, que representa el equipo completo, dentro de ella disponemos de un objeto de tipo *Spectrometer*, cuya clase viene ya implementada del sistema de desarrollo comercial del espectrómetro, también tenemos otro objeto de tipo *Procesador*, clase que proviene de proyectos previos, pero ha sido modificada para poder usarla con un espectrómetro. Este último es el encargado de realizar el procesado *lock-in* y permite obtener los

datos del espectrómetro. A su vez el procesador puede contener N *lock-in*, que serán el número de pixeles que realmente vamos a procesar.

Además, para poder controlar *Reflectometro* creamos una clase más grande llamada *Principal* que contiene un objeto de tipo *Reflectometro* y es la que se va a comunicar con él, pidiéndole datos, permitiendo representar las medidas realizadas y guardarlas.

Interfaz gráfica

La interfaz gráfica principal permite visualizar el espectro de la medida realizada en tiempo real, además una vez que sea estable y decidamos calibrar, nos mostrará también el coeficiente de reflexión o transmisión.

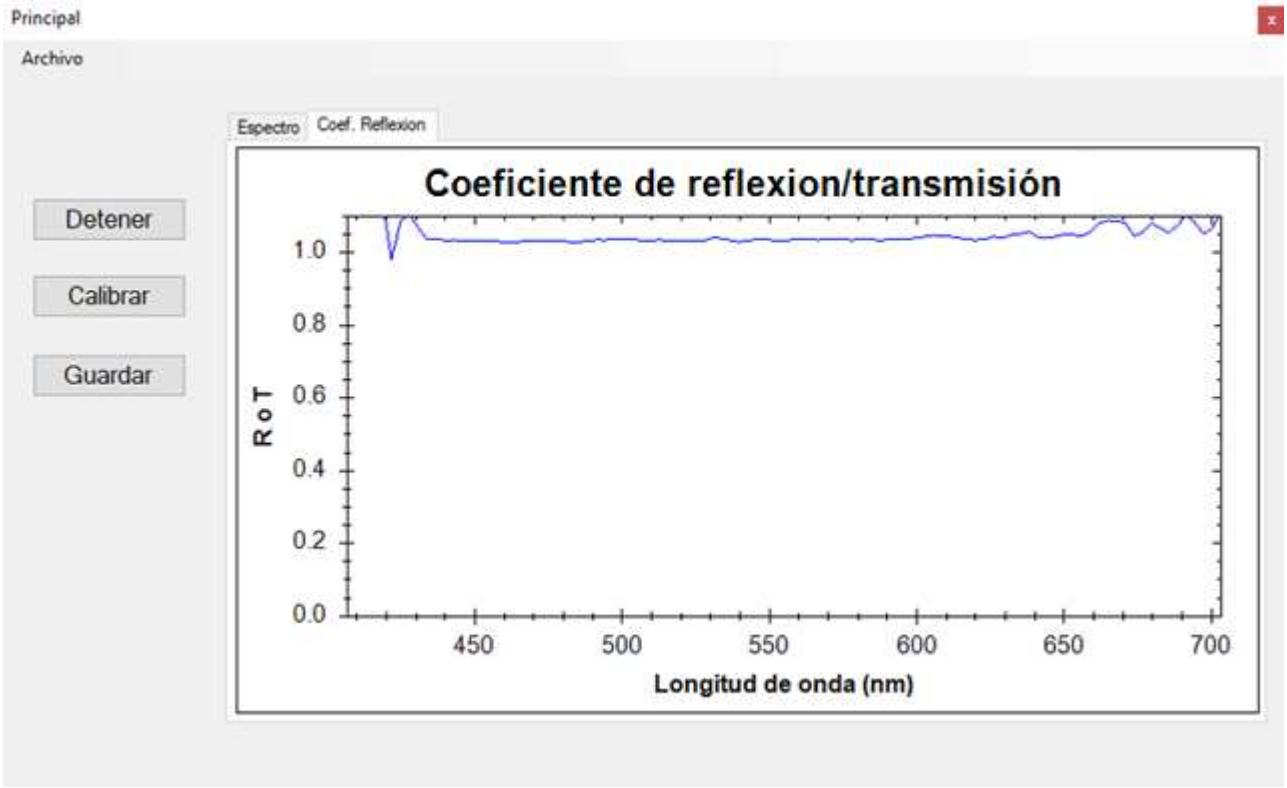


Figura 24. Interfaz principal

Podemos ver que presenta tres botones, el primero “Medir/Detener”, para iniciar y parar la medida, el segundo “Calibrar” nos permite hacer la medida de patrón, a partir de la cual conseguimos el valor del coeficiente, y por último “Guardar”.

Asimismo, dispone de un panel de control, situado en “Archivo”, en el encontramos opciones más avanzadas (ver Figura 25). Entre ellas podemos destacar la posibilidad de seleccionar los valores inicial y final de longitud de onda, la frecuencia de muestreo, el filtro, indicar el valor de pixeles del espectrómetro a utilizar, ya que dispone de 2500, pero con 100 sería suficiente como ya habíamos mencionado anteriormente.

Sobre todo, debemos resaltar la forma en la que se puede realizar el diezmado de los pixeles elegidos, puede ser sin o con promediado, con esta última opción conseguiremos reducir el ruido.

La interfaz gráfica de panel de control también nos permite ver el espectro, pero además la medida temporal de un *lock-in*, teniendo los medios para poder seleccionar cuál.

Igualmente muestra, a diferencia de la principal, valores numéricos, lo cual nos permite corroborar el funcionamiento de manera objetiva.

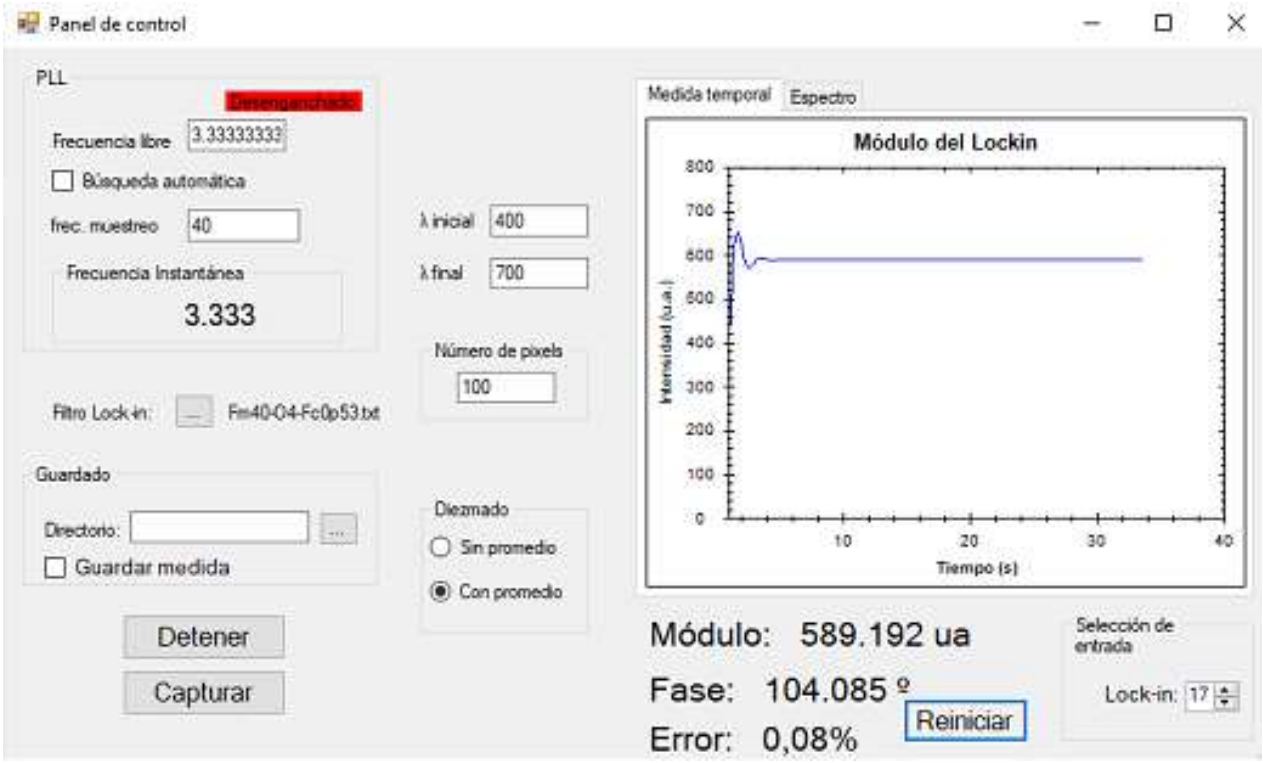


Figura 25. Interfaz panel de control

5 Resultados

Uno de los primeros y más importantes resultados que conseguimos es verificar la posibilidad de hacer *lock-in* con el espectrómetro y que éste funciona correctamente, ya que ésta era una de las principales dudas que tuvimos al inicio del proyecto.

Tras experimentar con el programa, logramos establecer los límites de nuestro sistema, física y electrónicamente es posible un rango desde 20 Hz a 40 KHz en la frecuencia de muestreo. Sin embargo, los valores más altos (a partir de 500 Hz) no son prácticos, porque en esos casos el tiempo de integración del espectrómetro es mínimo y apenas tendremos señal.

Como ya hemos nombrado, las primeras medidas son realizadas en transmisión por simplicidad:

$$T\ (en\ tanto\ por\ 1) = \frac{medida}{medida\ de\ referencia} * T_{patrón}$$

Donde la medida de referencia se toma sin poner ninguna muestra, por lo tanto, como en medio tendremos aire, el valor de transmisión del patrón será 1.

Antes que nada, comparamos la medida de transmisión utilizando y sin utilizar *lock-in*, las cuales se realizan con nuestro programa explicado anteriormente y con el provisto con el espectrómetro (*Waves*) respectivamente.

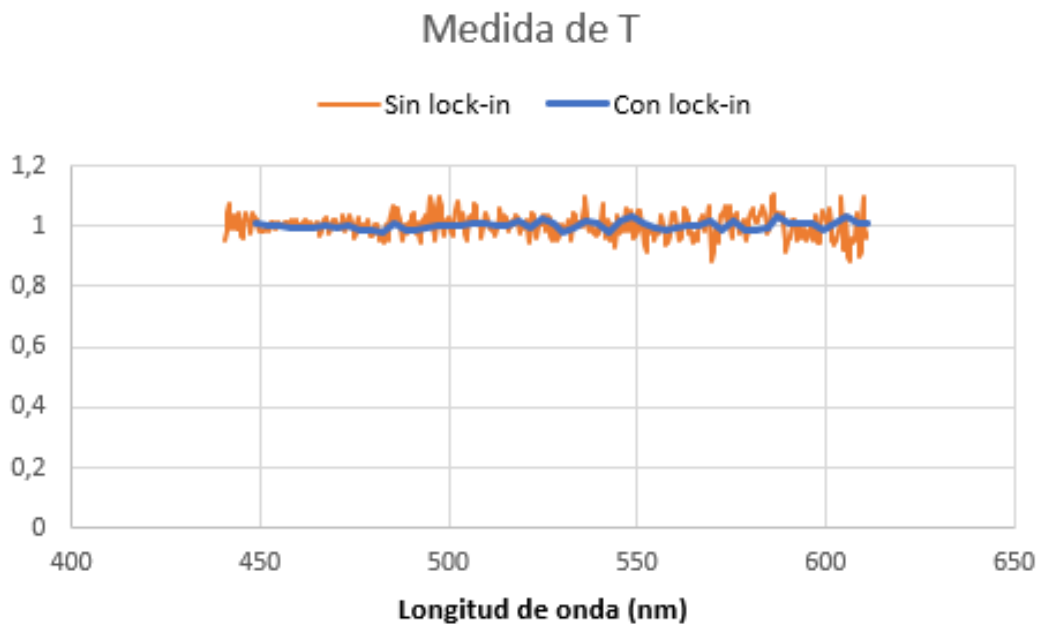


Figura 26. Comparación coeficiente de transmisión utilizando y sin utilizar lock-in

La mejora que se obtiene con este procesado frente a la adquisición directa es significativa (ver Figura 26): por ejemplo, se ha calculado la desviación estándar en el rango de 440 nm a 610 nm de longitud de onda, que pasada a porcentaje nos indica un cambio de 3.54% a 1.32%.

A continuación, contrastamos la representación del coeficiente de transmisión sin y con promediado:

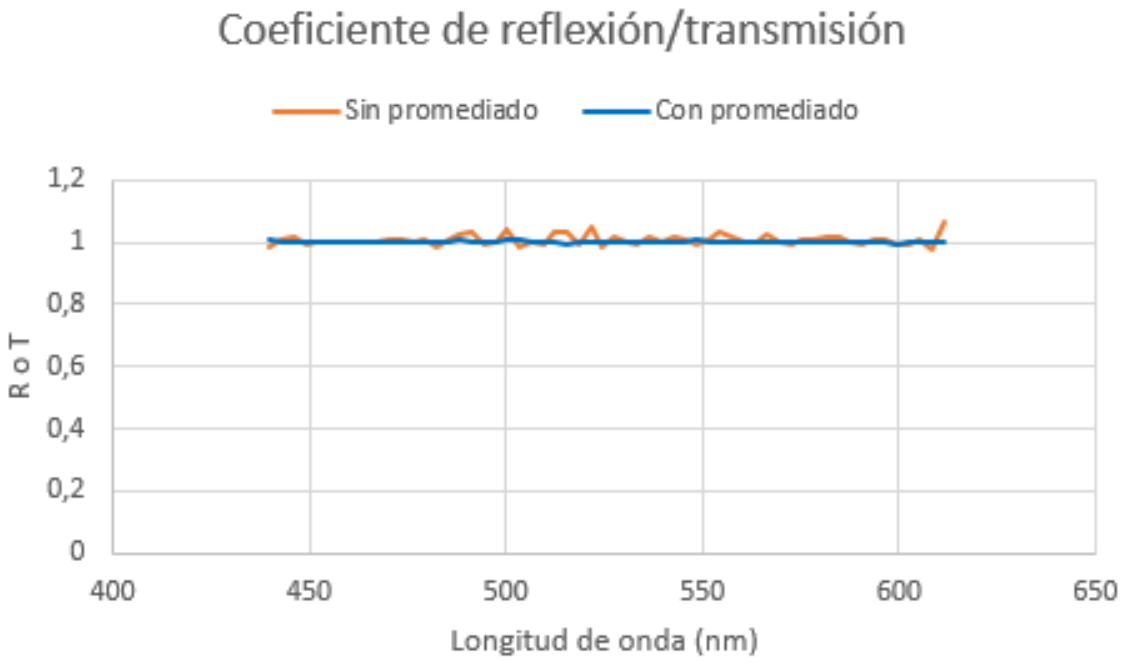


Figura 27. Coeficiente sin y con promediado respectivamente

Como podemos observar, se logra una mejor desviación estándar con el promedio, alcanzando un progreso en el error del 1.65% al 0.27%.

A continuación, se muestran una serie de medidas realizadas para verificar el funcionamiento de nuestro sistema.

En primer lugar, introducimos unas gafas de ver normales, que tienen aproximadamente una transmisión del 0.9, lo cual se confirma en la siguiente imagen, obteniendo valores en torno al 92%.

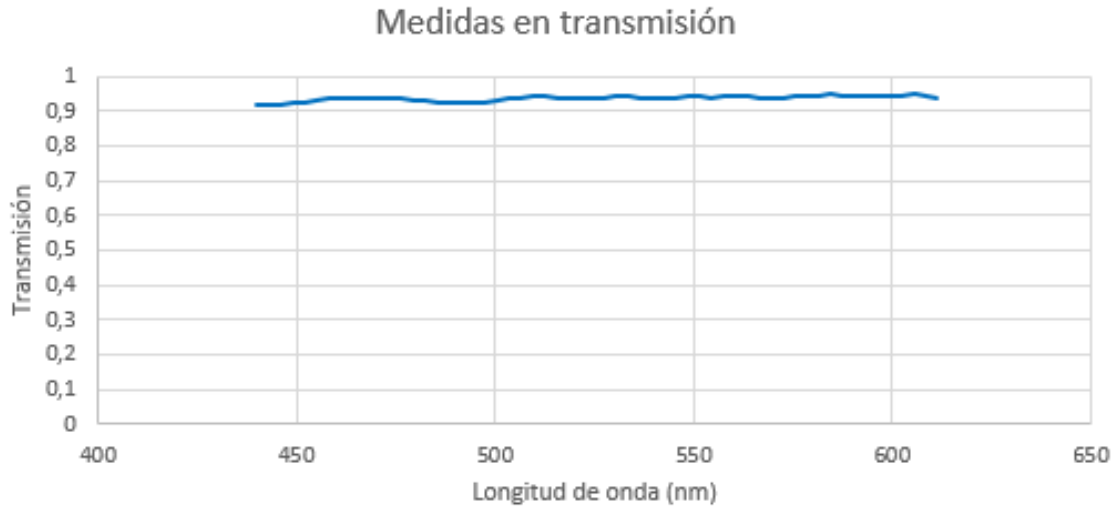


Figura 28. Medidas en transmisión con gafas normales

A causa de que la gráfica anterior es plana, podemos demostrar que estamos midiendo correctamente, ya que no visualizamos la forma de la curva del LED (ver Figura 17), debido a que

estamos dividiendo por la referencia, entonces solo depende de la muestra que hayamos puesto, en este caso se trataba de unas gafas trasparentes, las cuales dejan pasar prácticamente toda la luz, y cuya atenuación suele ser por lo que refleja el vidrio (8%).

En lugar del coeficiente de transmisión se puede mostrar también la atenuación en dB que producirían las gafas:

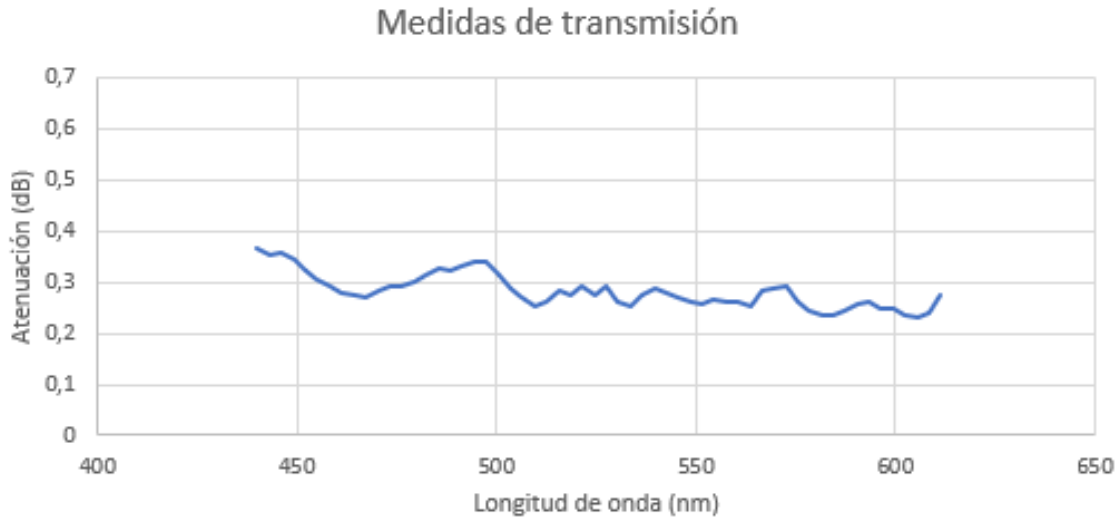


Figura 29. Atenuación de las gafas normales en dB

En segundo lugar, cambiamos las gafas de ver normales por unas de sol y repetimos las mediciones:

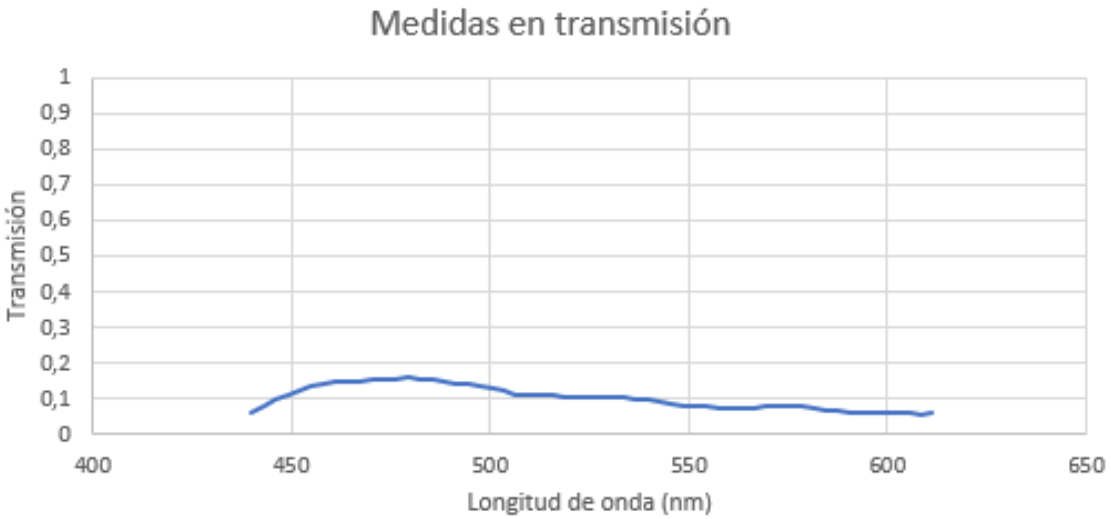


Figura 30. Medidas en transmisión con gafas de sol

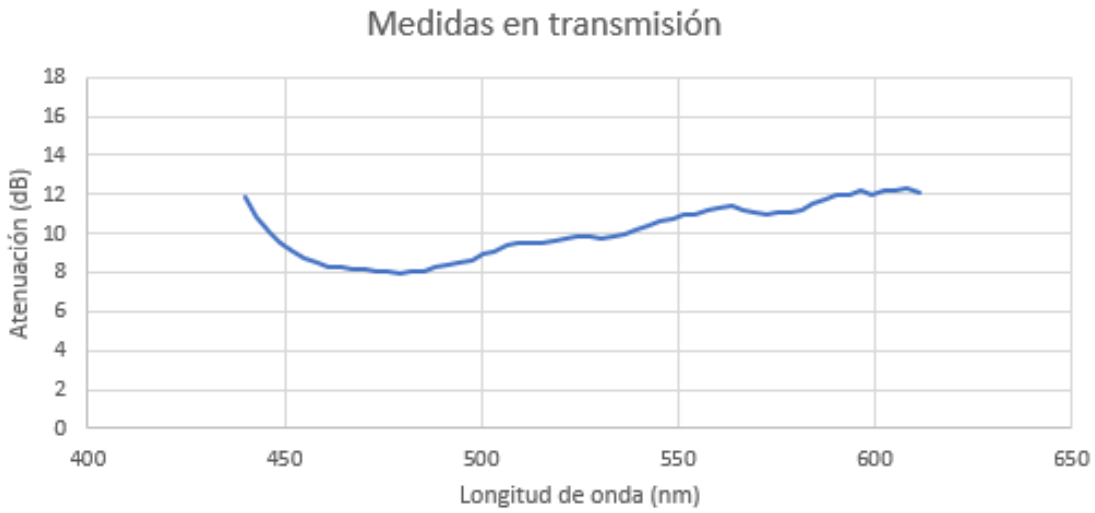


Figura 31. Atenuación de las gafas de sol en dB

Como era de esperar, en este caso disminuye bastante la transmisión, entonces la atenuación aumenta. También debemos remarcar que la curva de atenuación es bastante plana, ya que ambas gafas atenúan aproximadamente la misma luz en todos los colores.

Posteriormente, se muestra el espectro de ambas medidas (ver Figura 32).

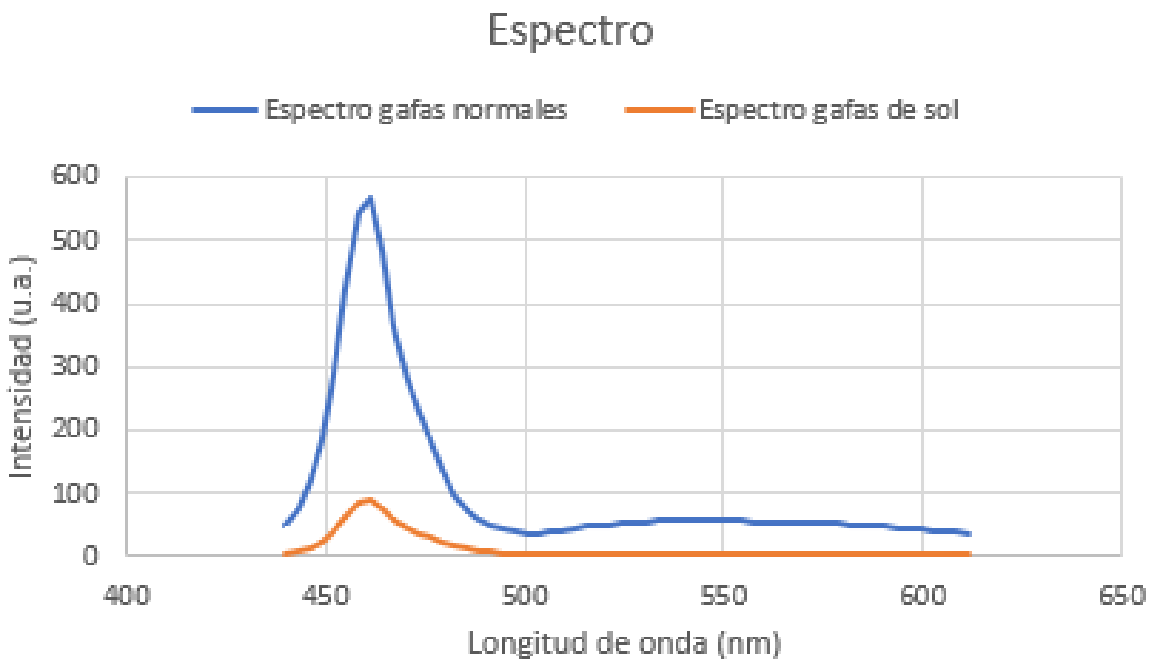


Figura 32. Espectro de la medida realizada con gafas normales y de sol

Continuamos realizando medidas en transmisión, pero esta vez vamos a utilizar dos filtros, como los que se presentan en la sucesiva fotografía:

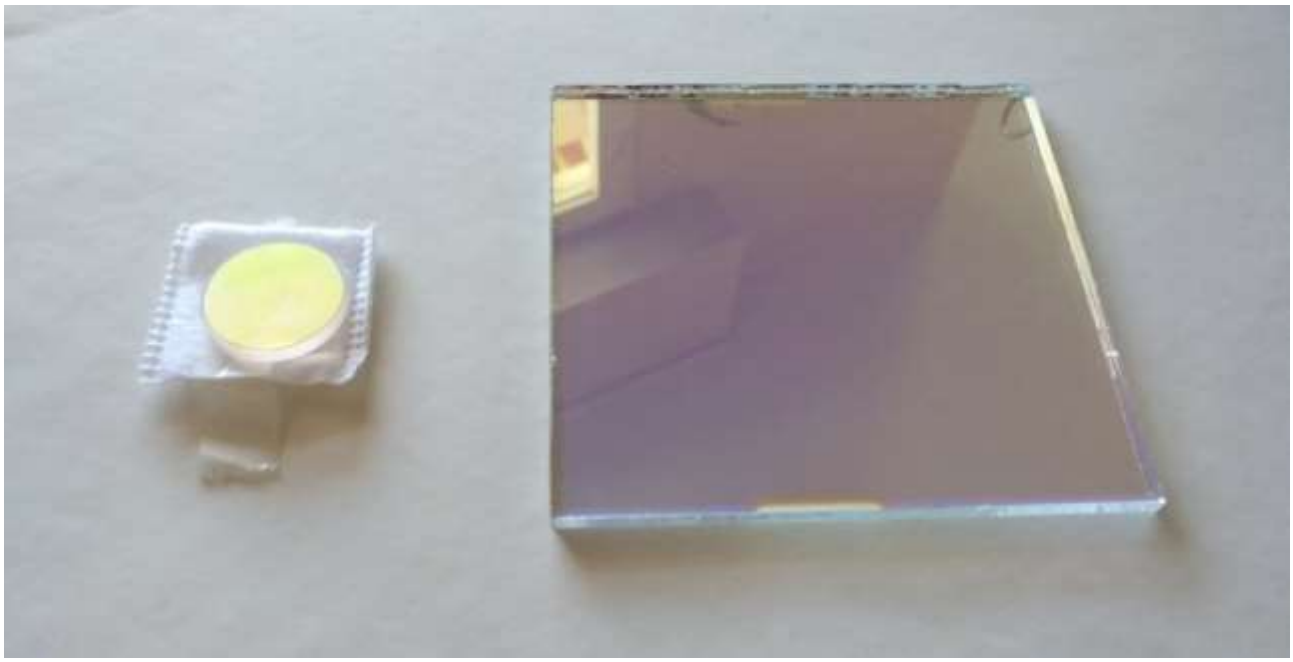


Figura 33. Filtro paso bajo comercial y filtro paso bajo diseñado en prácticas de máster

Primero usaremos el de la derecha, que es un filtro cuadrado diseñado en prácticas de una asignatura de Máster, que actúa como filtro paso bajo sencillo de 500 nm.

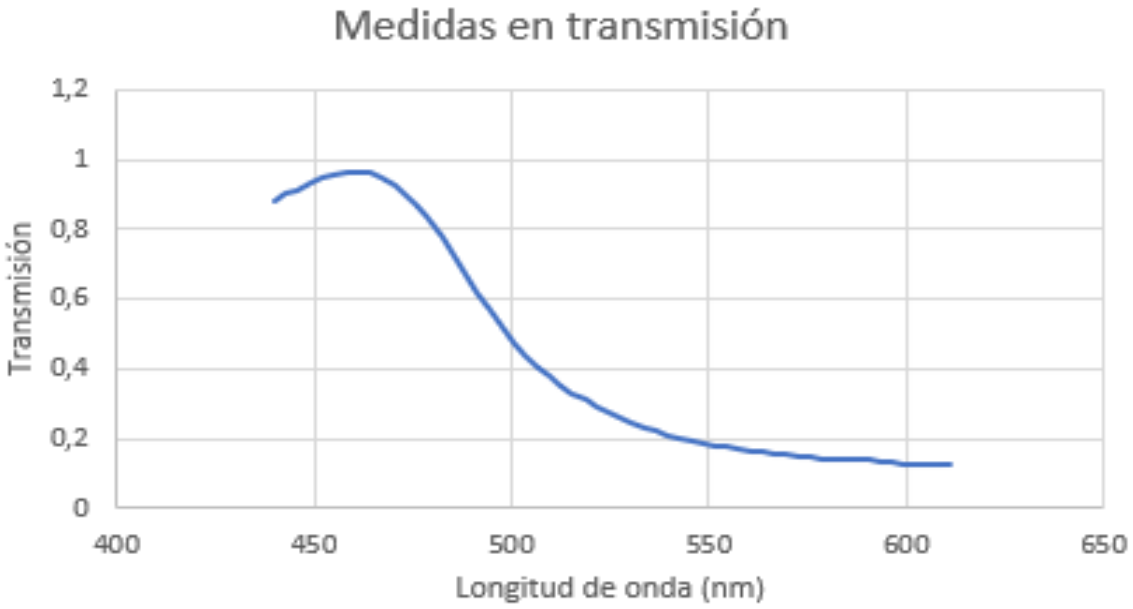


Figura 34. Medidas en transmisión con vidrio (filtro paso bajo sencillo)

Podemos observar que transmite sobre todo en azul, en otras palabras, refleja en amarillo (ver Figura 33). De igual manera que en las medidas anteriores, podemos mostrar la atenuación en dB del filtro:

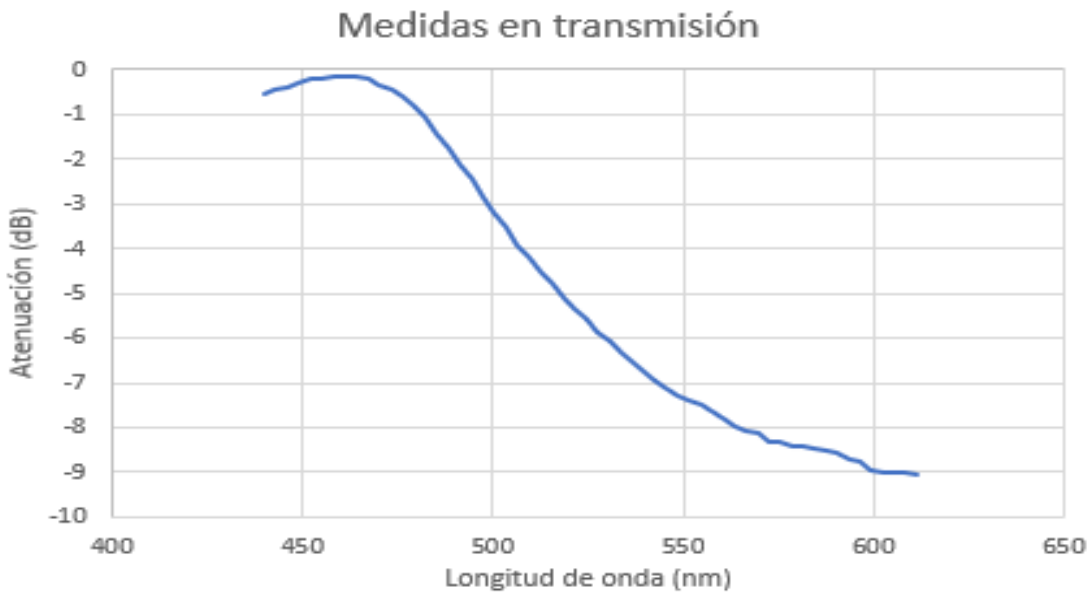


Figura 35. Atenuación del vidrio (filtro paso bajo sencillo) en dB

En segundo lugar, también utilizamos un filtro paso bajo, pero en este caso comercial, en concreto el 64-602 de Edmund Optics, con longitud de onda de corte de 525 nm, se trata del que esta sitiado a la izquierda en la Figura 33 y con el que volvemos a rehacer las medidas:

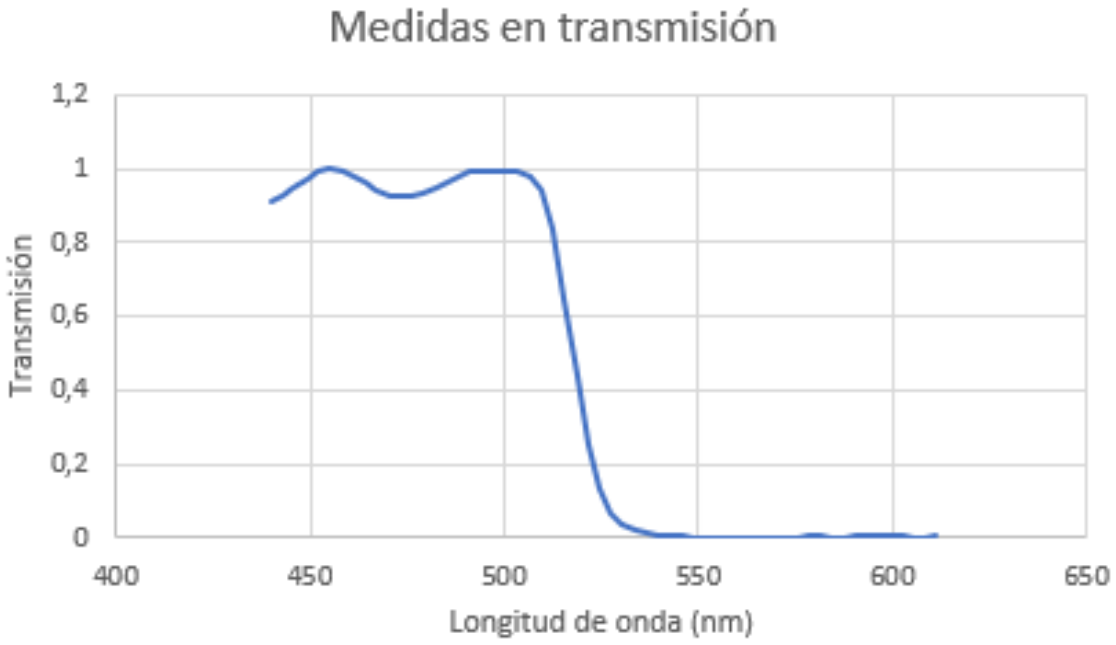


Figura 36. Medidas en transmisión del filtro paso bajo comercial

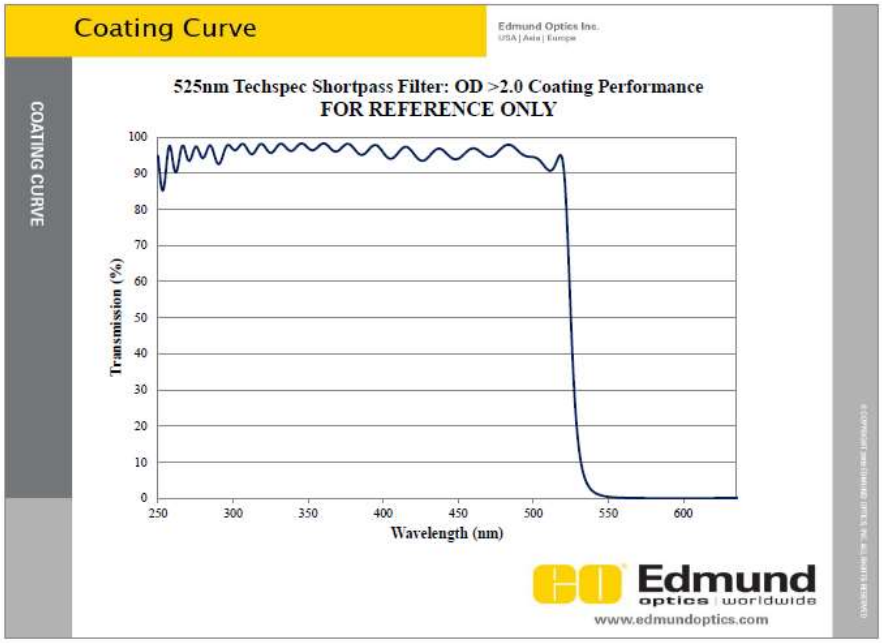


Figura 37. Transmisión espectral del filtro 64-602

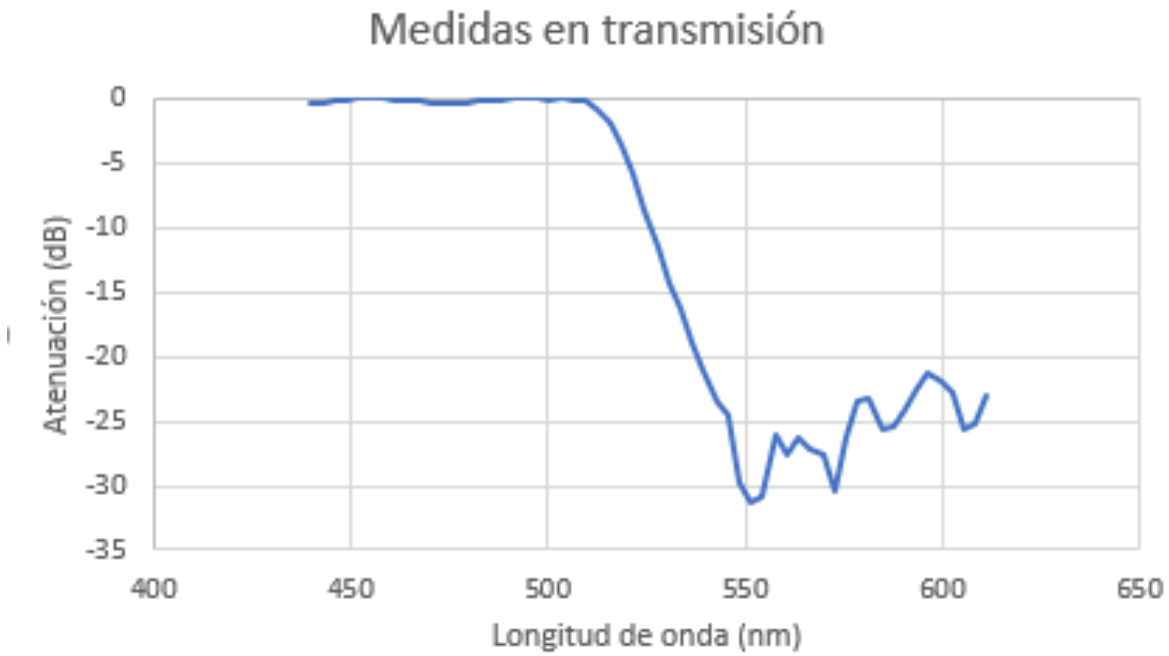


Figura 38. Atenuación del filtro paso bajo comercial en dB

Se comprueba que se trata un filtro más ideal, es decir, más parecido a un escalón. La medida sigue la curva proporcionada por el fabricante (ver Figura 37).

Por último, empleamos un filtro paso banda de 500 nm, concretamente el FB500-40 de Thorlabs:

Figura 39. Filtro paso banda

Presentamos conjuntamente la medida realizada en transmisión junto con las especificaciones del filtro:

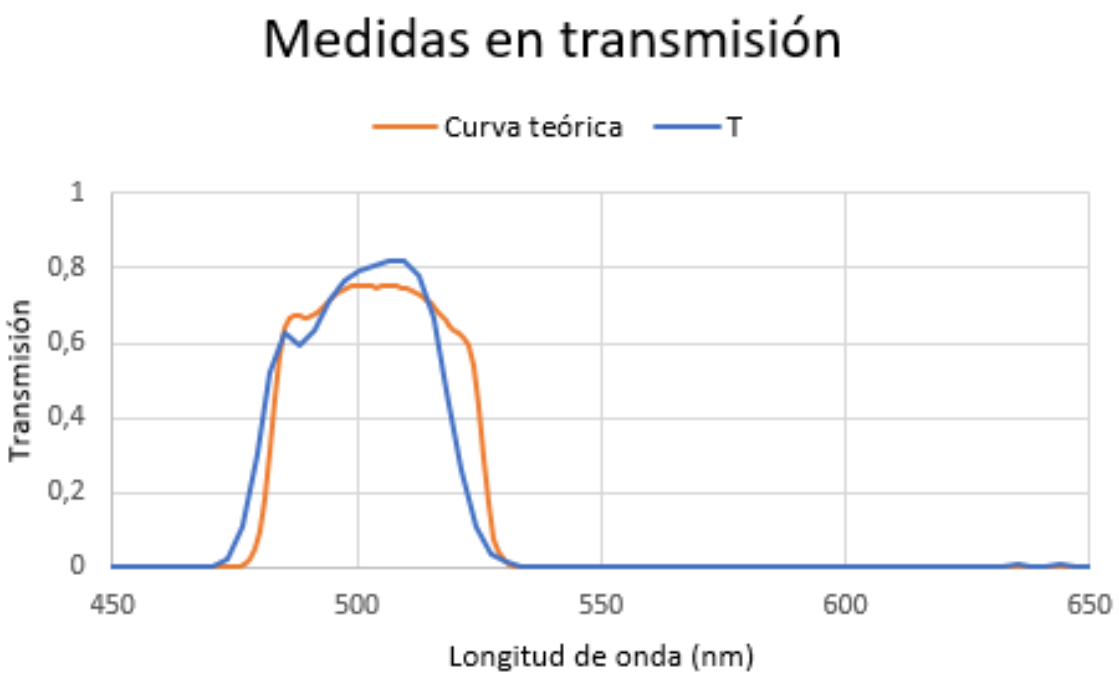


Figura 40. Medidas en transmisión y curva del fabricante del filtro paso banda

La forma del filtro medido es similar a lo esperado teóricamente.

6 Presupuesto

Incorporamos un presupuesto aproximado de un primer modelo que sería portable, para poder realizar medidas en nieve, muchos de los elementos coinciden con los que hemos usado en el prototipo de laboratorio.

Producto	Modelo	Unidades	Precio (€) / Unidad	Total
Ordenador y Arduino	LattePanda Alpha 800s	1	337,082	337,082
Espectrómetro	Broadcom Qmini AFBR-S20M2WV	1	1287,94	1287,94
Placa y componentes	*Diseño propio*	1	200	200
Fuente de alimentación	Power bank Powergorilla	1	235,95	235,95
Fibra	Solarization-Resistant Patch Cable	1	104,26	104,26
Carcasa	Carcasa de fibra de vidrio	1	97,74	97,74
			TOTAL (€)	2262,972

Tabla 1. Presupuesto primer modelo portable

El precio final se encuentra en torno a los 2000 €, por lo que se trata de un coste bastante asequible para un equipo de medida tan específico.

7 Conclusiones y líneas futuras

7.1. Conclusiones

Se ha demostrado la viabilidad de realizar detección *lock-in* con un espectrómetro óptico para poder realizar medidas de reflectancia en nieve.

También se ha construido un prototipo de laboratorio que puede evolucionar a un sistema de campo, con un coste aproximado de 2000 €.

Los resultados experimentales de las medidas espectrales de transmisión muestran un error cerca del 1%, aproximadamente un tercio del inicial. La implementación del promediado también nos proporciona una mejora significativa.

Se han definido los límites y capacidades del sistema en un rango de 20 Hz a 40 KHz de frecuencia de muestreo.

Además, se ha creado un software en Visual Studio 2019 y en Arduino para proporcionar el control del equipo y la visualización de los datos, entre ellos el espectro, el módulo de un *lock-in* y el coeficiente de reflexión o transmisión. Con él se ha comprobado que el funcionamiento global del sistema de detección síncrona mediante un espectrómetro óptico es correcto.

Adicionalmente, se ha llevado a cabo el diseño de una placa a través del programa EAGLE que será utilizada para la alimentación el LED, ya que actualmente se está realizando a través del Arduino.

7.2. Líneas futuras

En primer lugar, tiene que realizarse la fabricación de la placa descrita anteriormente para poder alimentar LEDs más potentes y realizar mejor las mediciones.

Además, se debe continuar el montaje del prototipo de medida de reflectancia utilizando detección síncrona espectral, el cual deberá hacerse más robusto y evaluar su funcionamiento con medidas de reflectancia y en nieve.

Tras ello, será necesario crear un sistema completo para medir en todos los distintos ángulos (fotogoniómetro).

Finalmente, se seguirá con el desarrollo y diseño de un equipo final. Después habrá que estudiar la viabilidad de un diseño o equipo comercial, para su posterior fabricación.

Una vez conseguido, sería interesante industrializarlo, para poder fabricarlo en serie, lo que conlleva componentes más baratos y adecuados, aumentar la resistencia del fotogoniómetro, etc.

Cabe destacar que la posibilidad de poder hacer detección *lock-in* con el espectrómetro puede aplicarse en otros proyectos diferentes.

8 Bibliografía

[1] *S. M. K. Skiles, M. Flanner, J. M. Cook, M. Dumont, and T. H. Painter, "Radiative forcing by light-absorbing particles in snow", Nat. Clim. Chang., vol. 8, no. 11, pp. 964–971, 2018.*

[2] *PANalytical Company: Snow & ice.*

[3] *T. Nakamura, O. Abe, T. Hasegawa, R. Tamura, T. Ohata, "Spectral reflectance of snow with a known grain-size distribution in successive metamorphism", Cold Reg. Sci. Technol. 32, 13-26, 2001.*

[4] *R. Alonso, F. Villuendas, J. Borja, L. A. Barragán, and I. Salinas, "Low-cost, digital lock-in module with external reference for coating glass transmission/reflection spectrophotometer," Meas. Sci. Technol., vol. 14, no. 5, 2003.*

[5] *A. Kokhanovsky, M. Lamare, B. Di Mauro, G. Picard, L. Arnaud, M. Dumont, F. Tuzet, C. Brockmann, J. E. Box, "On the reflectance spectroscopy of snow", The Cryosphere, 12, 2371–2382, 2018*

[6] *T. H. Painter, N. P. Molotch, M. Cassidy, M. Flanner, K. Sterren, "Instruments and Methods – Contact spectroscopy for determination of stratigraphy of snow optical grain size", Journal of Glaciology, Vol. 53, No. 180, 2017*

[7] *G. Picard, L. Arnaud, F. Domine, M. Fily, "Determining snow specific surface area from near-infrared reflectance measurements: Numerical study of the influence of grain shape", Cold Regions Science and Technology, 56, 10-17, 2009*

Anexo A. Hoja de características

Espectrómetro

BROADCOM

Data Sheet

Qmini AFBR-S20M2XX

Miniature Spectrometer with Onboard Processing for Mobile Applications and Industrial Integration

Key Features

- Spectral resolution from 0.5 nm
- Miniature size
- Customizable wavelength range, sensitivity, and resolution
- Powerful onboard processing and evaluation

Applications

- Color measurement
- Chemical analysis
- Quality control
- System integration
- Counterfeit detection
- Environmental analysis
- Biomedical applications
- Light analysis
- Process control and monitoring

Overview

Within an amazingly small design, the Qmini delivers technical specifications that are unprecedented at this size. Its compact design enables tight integration in applications where space is limited, like mobile analysis devices. The Qmini includes a powerful electronics board that enables:

- Full processing of spectra in the device (offset, nonlinearity, dark spectrum, and spectral sensitivity)
- Averaging and smoothing
- Binning and buffering of spectra

The Qmini also features a replaceable entrance slit, reduced stray light, and lower power consumption.

Part Number	Product Configuration	Wavelength Range	Spectral Resolution
AFBR-S20M2UV	Qmini UV	220 nm to 400 nm	0.5 nm
AFBR-S20M2VI	Qmini VIS	370 nm to 750 nm	0.7 nm
AFBR-S20M2NI	Qmini NIR	730 nm to 1080 nm	0.7 nm
AFBR-S20M2WU	Qmini Wide UV Sensitivity optimized at 300 nm	225 nm to 1000 nm	1.5 nm
AFBR-S20M2WV	Qmini Wide VIS Sensitivity optimized at 500 nm	225 nm to 1000 nm	1.5 nm
AFBR-S20M2VN	Qmini VIS/NIR	480 nm to 1100 nm	1.5 nm

Specifications	
Focal length	50 mm
Grating	300 or 600 lines/mm
Entrance slit	20 µm (changeable)
Dynamic range	1300:1
Numerical aperture	0.1
Stray light	<0.1 %
Exposure time range	3 µs to 600s
Detector	2500-pixel linear CCD sensor
A/D converter	16-bit
Calibration	Wavelength, sensitivity, nonlinearity, and multiple dark spectra stored in device
Internal memory	32 MB (>3000 spectra)
Transfer speed to PC	USB 2.0 High-Speed
Optical interface	SMA connector
Digital interfaces	USB 2.0 with Type-C connector, SPI, UART
Dimensions	64.0 mm × 42.0 mm × 14.5 mm
Weight	60g
Operating temperature	-15°C to 60°C (non-condensing)
Storage temperature	-25°C to 70°C
Power consumption	5V DC, up to 130 mA
PC operating system	Windows 10, 8, 7, Vista, XP

Broadcom Spectroscopy

Data Sheet

Application Software

Every Qmini spectrometer includes Waves user software developed for general-purpose spectroscopy applications. Waves includes sophisticated algorithms for data acquisition and evaluation, which provides the following features through a clear and straightforward user interface.

- Take and display series of spectra
- Automatic exposure control with dark spectrum interpolation
- Import most ASCII-based file formats
- Export as ASCII table to almost any numerical analysis software
- Comprehensive tools for displaying and analyzing spectra
- Strip charts for comparing characteristic values between multiple spectra including peak follower in real time
- Graph printing and export to PDF
- Dynamic peak finder (no need to set a threshold level)
- Dark spectrum interpolation
- Transmission, absorption, and reflection measurements
- Colorimetry

Waves is very easy to use and very intuitive. Various spectrum evaluation options are available with minimal effort and only a few mouse clicks. For example, to zoom in, adjust the zoom slider. To move around, adjust the scrollbar. To change the x-axis unit, select the corresponding button. Values such as peaks or colorimetry are instantly calculated as soon as a spectrum is taken. Waves is available as a free download from our website.

Software Library

A software development kit (SDK) is also included to control the spectrometer and take spectra from your own software. It consists of a Windows DLL library for the .NET framework, documentation, and sample code. The SDK can be used with any programming language that can use .NET DLLs, including C#, Visual Basic .NET, C++, Delphi, LabVIEW, Matlab, and Mathematica.

Communication Protocol

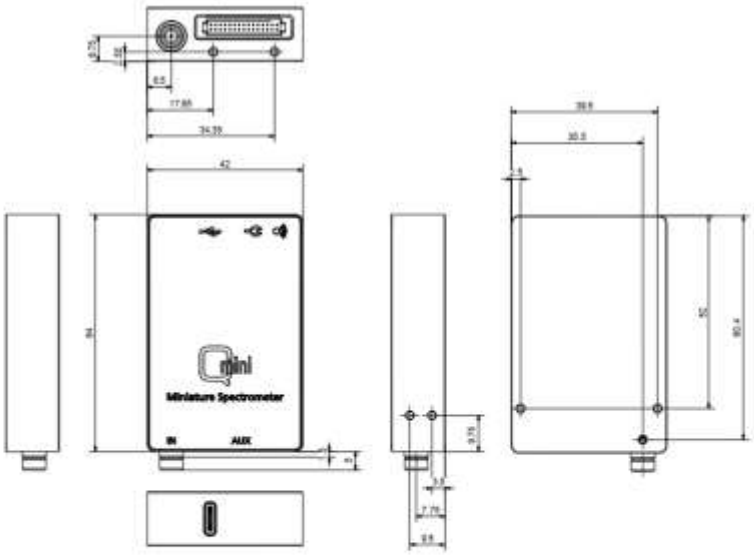
The spectrometer can also be directly controlled from an embedded microcontroller or other operating systems using the device communication protocol. Just like our application software, the protocol is designed to be both powerful and easy to use for software developers.

I/O Port

The Qmini includes a new auxiliary connector for analog and digital I/O, communication interfaces and power supply (if USB is not used). The eight digital channels can be configured as trigger input, shutter or flash lamp control, process control, or general purpose I/O pins.

The Qmini supports three trigger modes: software trigger, interval trigger, and external trigger. It can be set to trigger on the start or the end of the exposure period.

Qmini Schematic Drawing



BROADCOM
connecting everything

For more product information: broadcom.com

AFBR-S20M2XX-DS101 July 9, 2018

LattePanda

CARACTERISTICAS

- Especificaciones sólidas y diseño extremo
- Soporte de sistema operativo dual: Windows / Linux
- Redes definitivas en la nube y las cosas
- Hasta 42 interfaces extensibles brindan jugadas ricas
- Diseño ultrafino, minimizando la ocupación del espacio.
- Reconstruya cómo codificar de manera eficiente con Multible OS

ESPECIFICACIÓN

- CPU: Intel 8th m3--8100y
- Núcleo: 1.1-3.4 GHz de doble núcleo, cuatro hilos
- Benchmark (PassMark): hasta 4128, doble poder de cómputo en comparación con los mismos productos de rango de precios en el mercado
- Gráficos: Intel HD Graphics 615, 300-900MHz
- **RAM: 8G LPDDR3**
- Memoria externa:
  - 1x M.2 M Key, PCIe 4x, compatible con SSD NVMe y SSD SATA
  - 1x M.2 E Key, PCIe 2x, compatible con USB2.0, UART, PCM
- Conectividad: WIFI 802.11 AC, 2.4G y 5G
- Doble banda Bluetooth 4.2
- Gigabyte Ethernet
- Puertos USB: 3x USB 3.0 Tipo A / 1x USB Tipo C, compatible con PD, DP, USB 3.0
- Pantalla: Salida HDMI / Soporte DP tipo C / pantallas táctiles eDP extensibles
- Coprocesador : Arduino Leonardo
- GPIO y otras características: 2x 50p GPIO incluyendo I2C, I2S, USB, RS232, UART, RTC, Administración de energía, botón de encendido extensible, todo lo que necesita
- Soporte del sistema operativo: Windows 10 Pro / Linux Ubuntu
- Dimensión: 115 mm * 78 mm * 14 mm

LISTA DE EMBARQUE

- LattePanda Alpha 800s x1
- Regalo gratis - Ventilador de enfriamiento activo x1
- Regalo gratis - Adaptador de corriente PD 45w x1

Anexo B. Código fuente Arduino

```
#include <TimerOne.h>

int factor=12;
int contador=0;
bool modulacion=false;

void ISR_muestreo()
{
 contador++;
  if (contador==12)
  {
    contador=0;
    modulacion=!modulacion;
    if (modulacion)
      digitalWrite(1, HIGH);
    else
      digitalWrite(1, LOW);
  }
}

void setup() {
 pinMode (2, INPUT_PULLUP);
 attachInterrupt( digitalPinToInterrupt(2), ISR_muestreo, CHANGE);
}

void readSerialInput() // Función invocada cuando se recibe algo por el Serial
{
  String str = Serial.readStringUntil('\n');
  str = str.substring(0, str.length() - 1);//Para quitar el /r con Visual Studio
  factor = str.toInt();
  Serial.println(factor);
}

void loop() {
  if (Serial.available()) //comprueba el puerto serie
    readSerialInput();
}
```

Anexo C. Código fuente Visual Studio

Principal.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
using ZedGraph;

namespace Nieve
{
    public partial class Principal : Form
    {
        Reflectometro miReflectometro;
        double[] medidaPatron;
        double[] medidaEspectro;
        double[] medidaCoeficiente;

        public Principal()
        {
            InitializeComponent();
            miReflectometro = new Reflectometro();
            miReflectometro.UnaMedida += new
Reflectometro.MedidaHandler(miReflectometro_UnaMedida); //EVENTO
        }

        private void miReflectometro_UnaMedida(object sender, double[] medida)
        {
            medidaEspectro = new double[medida.Length];
            miReflectometro.Medida.CopyTo(medidaEspectro, 0);
            representaGrafica(zedGraphControlEspectro, medidaEspectro);//representar medida
en la gráfica

            if (medidaPatron != null) //hay calibración
            {
                medidaCoeficiente = new double[medida.Length];
                for (int i = 0; i < miReflectometro.NumLockin; i++)
                    medidaCoeficiente[i] = medidaEspectro[i] / medidaPatron[i] * 1; //porque
de momento el valor del patrón es 1 (aire)
                representaGrafica(zedGraphControlCoeficiente, medidaCoeficiente);
            }
        }

        #region Inicialización y cerrado de la ventana

        private void Principal_Load(object sender, EventArgs e)
        {
            IniciarGraficas();
        }

        private void Principal_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (buttonMedir.Text != "Medir")
                miReflectometro.PararMedida();
        }
```

```
    #endregion

    #region Graficas

        private void IniciarGraficas()
        {
            //GRÁFICA ESPECTRO
            GraphPane grafica = zedGraphControlEspectro.GraphPane;
            PointPairList listaEspectro;
            LineItem curva;
            listaEspectro = new PointPairList(miReflectometro.LambdasGrafica,
miReflectometro.LambdasGrafica);
            curva = grafica.AddCurve("", listaEspectro, Color.Blue, SymbolType.None);
            grafica.Title.FontSpec.Size = 20;
            grafica.Title.Text = "Espectro medida";
            grafica.XAxis.Title.FontSpec.Size = 16;
            grafica.XAxis.Title.Text = "Longitud de onda (nm)";
            grafica.YAxis.Title.FontSpec.Size = 16;
            grafica.YAxis.Title.Text = "Intensidad (u.a.)";
            grafica.XAxis.Scale.FontSpec.Size = 16;
            grafica.XAxis.Scale.Min = miReflectometro.LambdasGrafica[0];
            grafica.XAxis.Scale.Max =
miReflectometro.LambdasGrafica[miReflectometro.LambdasGrafica.Length-1];
            grafica.YAxis.Scale.FontSpec.Size = 16;
            grafica.YAxis.Scale.Min = 0;
            grafica.YAxis.Scale.Max = 800;
            grafica.YAxis.Scale.MaxAuto = true;
            zedGraphControlEspectro.AxisChange();
            zedGraphControlEspectro.Invalidate();

            //GRÁFICA COEFICIENTE REFLEXIÓN
            grafica = zedGraphControlCoeficiente.GraphPane;
            PointPairList listaCoeficiente;
            listaCoeficiente = new PointPairList(miReflectometro.LambdasGrafica,
miReflectometro.LambdasGrafica);
            curva = grafica.AddCurve("", listaCoeficiente, Color.Blue, SymbolType.None);
            grafica.Title.FontSpec.Size = 20;
            grafica.Title.Text = "Coeficiente de reflexion/transmisión";
            grafica.XAxis.Title.FontSpec.Size = 16;
            grafica.XAxis.Title.Text = "Longitud de onda (nm)";
            grafica.YAxis.Title.FontSpec.Size = 16;
            grafica.YAxis.Title.Text = "R o T";
            grafica.XAxis.Scale.FontSpec.Size = 16;
            grafica.XAxis.Scale.Min = miReflectometro.LambdasGrafica[0];
            grafica.XAxis.Scale.Max =
miReflectometro.LambdasGrafica[miReflectometro.LambdasGrafica.Length - 1];
            grafica.YAxis.Scale.FontSpec.Size = 16;
            grafica.YAxis.Scale.Min = 0;
            grafica.YAxis.Scale.Max = 1.1;
            zedGraphControlCoeficiente.AxisChange();
            zedGraphControlCoeficiente.Invalidate();
        }

        private void representaGrafica(ZedGraphControl grafica, double[] medida)
        {
            ZedGraph.GraphPane panel = grafica.GraphPane;
            if (panel.CurveList.Count <= 0)
                return;

            LineItem curve = panel.CurveList[0] as LineItem;
```

```
            if (curve == null)
                return;
            curve.Points = new PointPairList(miReflectometro.LambdasGrafica, medida);
            panel.XAxis.Scale.Min = miReflectometro.LambdasGrafica[0];
            panel.XAxis.Scale.Max =
miReflectometro.LambdasGrafica[miReflectometro.LambdasGrafica.Length - 1];
            grafica.AxisChange();
            grafica.Invalidate();
        }

        #endregion

        private void buttonMedir_Click(object sender, EventArgs e)
        {
            if (buttonMedir.Text == "Medir")
            {
                miReflectometro.Medir();
                buttonMedir.Text = "Detener";
            }
            else
            {
                miReflectometro.ParарMedida();
                buttonMedir.Text = "Medir";
            }
        }

        private void buttonCalibrar_Click(object sender, EventArgs e)
        {
            if (buttonMedir.Text == "Medir")
            {
                MessageBox.Show("El sistema debe estar midiendo para poder tomar la
calibración");
            }
            else
            {
                medidaPatron = new double[miReflectometro.Medida.Length];
                miReflectometro.Medida.CopyTo(medidaPatron, 0);
            }
        }

        private void zedGraphControlEspectro_Load(object sender, EventArgs e)
        {

        }

        private void zedGraphControlCoeficiente_Load(object sender, EventArgs e)
        {

        }

        private void buttonCapturar_Click(object sender, EventArgs e)
        {
            //EXAMINAR DIRECTORIO
            SaveFileDialog dialogoFichero = new SaveFileDialog();
            dialogoFichero.Filter = "Archivo .dat (.dat)|*.dat";
            dialogoFichero.RestoreDirectory = true;
            dialogoFichero.CheckPathExists = true;
            dialogoFichero.AddExtension = true;
            StreamWriter escritorEspectro;
            double[] datos = new double[miReflectometro.LambdasGrafica.Length];
```

```
            for (int i = 0; i < miReflectometro.LambdasGrafica.Length; i++)
                datos[i]=miReflectometro.LambdasGrafica[i];

            if (dialogoFichero.ShowDialog() == DialogResult.OK)
            {
                escritorEspectro = new StreamWriter(dialogoFichero.FileName);
                escritorEspectro.WriteLine("Longitud de onda (nm)" + "\t" + "Medida" + "\t"
+ "Calibración" + "\t" + "R");

                for (int i = 0; i < miReflectometro.LambdasGrafica.Length; i++)
                {
                    escritorEspectro.WriteLine(datos[i] + "\t" + medidaEspectro[i] + "\t" +
medidaPatron[i] + "\t" + medidaCoeficiente[i]);
                }
                escritorEspectro.Close();
            }
        }

        private void panelDeControlToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (buttonMedir.Text == "Detener")
            {
                miReflectometro.ParарMedida();
                buttonMedir.Text = "Medir";
            }
            miReflectometro.ShowDialog();
        }

        private void salirToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

Reflectometro.cs

```
using LockinDAQ;
using RgbDriverKit;
using System;
using System.Drawing;
using System.Globalization;
using System.IO;
using System.Windows.Forms;
using ZedGraph;

namespace Nieve
{
    public partial class Reflectometro : Form
    {

        Procesador miProcesador;
        Procesador.Configuracion configuracion;
        NumberFormatInfo nfi;

        //Variables gráfica
        double ejeTiempo = 0;
        const int DATOSAREPRESENTAR = 400;
        int lockinSel = 0;

        //Variables Excel
        bool guardar;
        string rutaExcel;
        StreamWriter escritorExcel;

        //Variables lock-in
        double[] ruido = new double[500];
        int contRuido = 0;
        bool inicioRuido = true;
        Spectrometer miEspectrometro;
        int numLockin;
        double[] lambdas;
        double[] lambdasGrafica;

        //Variables de configuracion
        bool tarjetaEsNI = false; //dice si es NI o LJ
        bool pll = false;
        bool diferencial = false;
        double min = 0;
        double max = 0.1;

        public int NumLockin
        {
            get
            {
                return numLockin;
            }

        }
        public double[] LambdasGrafica
        {
```

```
            get
            {
                return lambdasGrafica;
            }
        }

        public double[] Medida
        {
            get
            {
                return miProcesador.Modulos;
            }
        }

        public delegate void MedidaHandler(object sender, double[] medida);
        public event MedidaHandler UnaMedida; //evento de medida de un LED realizada

        protected virtual void OnUnaMedida(double[] medida)
        {
            if (UnaMedida != null)
                UnaMedida(this, medida);
        }

        public Reflectometro()
        {
            InitializeComponent();
            nfi = new NumberFormatInfo();
            nfi.NumberDecimalSeparator = ".";
            IniciarEspectrometro();
            IniciarConfiguracion();
            IniciarGraficas();
        }

        #region Inicialización y cerrado de la ventana

        private void IniciarEspectrometro()
        {
            if (miEspectrometro != null) miEspectrometro.Close();

            Spectrometer[] devices = Qseries.SearchDevices();

            if (devices.Length == 0) devices = RgbSpectrometer.SearchDevices();
            if (devices.Length == 0) devices = Qstick.SearchDevices();
            if (devices.Length == 0)
            {
                MessageBox.Show("No spectrometer found.", "Cannot initialize spectrometer",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            miEspectrometro = devices[0];

            try
            {
                miEspectrometro.Open();
                lambdas = miEspectrometro.GetWavelengths();
                MessageBox.Show("Device    name:    "    +    miEspectrometro.ModelName    +
Environment.NewLine
                    + "Manufacturer: " + miEspectrometro.Manufacturer + Environment.NewLine
                    + "Serial number: " + miEspectrometro.SerialNo + Environment.NewLine
```

```
                    + "Number of pixels: " + miEspectrometro.PixelCount,
                    "Spectrometer found and initialized", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                miEspectrometro.UseExternalTrigger = false; //USAMOS TRIGGER INTERNO
                miEspectrometro.ExternalTriggerSource = 2;
                miEspectrometro.TriggerOption =
SpectrometerTriggerOptions.FreeRunningTriggerStart;
            }
            catch (Exception ex)
            {
                miEspectrometro = null;
                MessageBox.Show(ex.Message, "Cannot initialize spectrometer",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        #endregion

        #region Graficas

        private void IniciarGraficas()
        {
            //GRÁFICA TIEMPO
            GraphPane grafica = zedGraphControlTiempo.GraphPane;
            RollingPointPairList lista;
            LineItem curva;
            lista = new RollingPointPairList(DATOSAREPRESENTAR);
            curva = grafica.AddCurve("", lista, Color.Blue, SymbolType.None);
            grafica.Title.FontSpec.Size = 20;
            grafica.Title.Text = "Módulo del Lockin";
            grafica.XAxis.Title.FontSpec.Size = 16;
            grafica.XAxis.Title.Text = "Tiempo (s)";
            grafica.YAxis.Title.FontSpec.Size = 16;
            grafica.YAxis.Title.Text = "Intensidad (u.a.)";
            grafica.XAxis.Scale.FontSpec.Size = 16;
            grafica.XAxis.Scale.Min = 1;
            grafica.XAxis.Scale.Max = 40;
            grafica.YAxis.Scale.FontSpec.Size = 16;
            grafica.YAxis.Scale.Min = 0;
            grafica.YAxis.Scale.MaxAuto = true;
            zedGraphControlTiempo.AxisChange();
            zedGraphControlTiempo.Invalidate();

            //GRÁFICA ESPECTRO
            grafica = zedGraphControlEspectro.GraphPane;
            PointPairList listaEspectro;
            listaEspectro = new PointPairList(lambdasGrafica, lambdasGrafica);
            curva = grafica.AddCurve("", listaEspectro, Color.Blue, SymbolType.None);
            grafica.Title.FontSpec.Size = 20;
            grafica.Title.Text = "Espectro";
            grafica.XAxis.Title.FontSpec.Size = 16;
            grafica.XAxis.Title.Text = "Longitud de onda (nm)";
            grafica.YAxis.Title.FontSpec.Size = 16;
            grafica.YAxis.Title.Text = "Intensidad (u.a.)";
            grafica.XAxis.Scale.FontSpec.Size = 16;
            grafica.XAxis.Scale.Min = lambdasGrafica[0];
            grafica.XAxis.Scale.Max = lambdasGrafica[lambdasGrafica.Length - 1];
            grafica.YAxis.Scale.FontSpec.Size = 16;
            grafica.YAxis.Scale.Min = 0;
            grafica.YAxis.Scale.Max = 800;
```

```
        grafica.YAxis.Scale.MaxAuto = true;
        zedGraphControlEspectro.AxisChange();
        zedGraphControlEspectro.Invalidate();
    }

    private void representaGraficaTiempo(double salidaLockin, double tiempo)
    {
        if (zedGraphControlTiempo.GraphPane.CurveList.Count <= 0)
            return;

        LineItem curve = zedGraphControlTiempo.GraphPane.CurveList[0] as LineItem;
        if (curve == null)
            return;

        RollingPointPairList lista = curve.Points as RollingPointPairList;
        if (lista == null)
            return;
        lista.Add(tiempo, salidaLockin);

        ZedGraph.Scale escalaX = zedGraphControlTiempo.GraphPane.XAxis.Scale;
        if (tiempo > escalaX.Max - escalaX.MinorStep)
        {
            escalaX.Max = tiempo + escalaX.MinorStep;
            escalaX.Min = escalaX.Max - 40;
        }

        if (tabControlGraficas.SelectedIndex == 0)
        {
            zedGraphControlTiempo.AxisChange();
            zedGraphControlTiempo.Invalidate();
        }
    }

    private void representaGraficaEspectro(double[] medidaEspectro)
    {
        ZedGraph.GraphPane grafica = zedGraphControlEspectro.GraphPane;
        if (grafica.CurveList.Count <= 0)
            return;

        LineItem curve = grafica.CurveList[0] as LineItem;
        if (curve == null)
            return;
        curve.Points = new PointPairList(lambdasGrafica, medidaEspectro);
        grafica.XAxis.Scale.Min = lambdasGrafica[0];
        grafica.XAxis.Scale.Max = lambdasGrafica[lambdasGrafica.Length - 1];

        if (tabControlGraficas.SelectedIndex == 1)
        {
            zedGraphControlEspectro.AxisChange();
            zedGraphControlEspectro.Invalidate();
        }
    }

    #endregion

    public void IniciarConfiguracion()
    {
        int lambdaInicial = 400;
        int lambdaFinal = 700;
        int frecMuestreo = 40;
```

```
            numLockin = 100;
            lambdasGrafica = new double[numLockin];
            int salida = 0;//primer reloj
            double umbralEntrada = 0.5;
            int[] entradas = new int[numLockin];
            string[] rutasFiltros = new string[numLockin];

            for (int i = 0; i < numLockin; i++)
            {
                rutasFiltros[i] = "Fm40-O4-Fc0p53.txt"; //lock-in
            }

            configuracion = new Procesador.Configuracion(frecMuestreo, pll, tarjetaEsNI,
diferencial, min, max, lambdaInicial, lambdaFinal, false);

            configuracion.confSeñal.salida = salida;
            configuracion.confSeñal.amplitud = 1;
            configuracion.confSeñal.continua = 0;
            configuracion.confSeñal.entradas = entradas;
            configuracion.confSeñal.rutasFiltros = rutasFiltros;
            configuracion.confSeñal.umbralPLL = umbralEntrada;
            configuracion.confSeñal.numLockines = numLockin;

            miProcesador = new Procesador(miEspectrometro, configuracion);

            textBoxFrecMuestreo.Text = configuracion.frecMuestreo.ToString();
            textBoxfrecLibre.Text = configuracion.confSeñal.frecuencia.ToString(nfi);
            textBoxlambdaInicial.Text = configuracion.lambdaIni.ToString();
            textBoxlambdaFinal.Text = configuracion.lambdaFin.ToString();
            textBoxNumlockin.Text = configuracion.confSeñal.numLockines.ToString();
            string texto = configuracion.confSeñal.rutasFiltros[lockinSel];
            labelFiltroLockin.Text = texto.Substring(texto.LastIndexOf("\\") + 1);
            labelEnganchado.BackColor = Color.Red;
            labelEnganchado.Text = "Desenganchado";
            numericUpDownLockin.Maximum = configuracion.confSeñal.numLockines - 1;
            numericUpDownLockin.Value = lockinSel;
        }
        public void Reconfigurar()   //configuracion.frecMuestreo, configuracion.lambdaIni,
configuracion.lambdaFin, configuracion.confSeñal.numLockines
        {
            double frecLibre = configuracion.frecMuestreo / 12.0;
            int[] entradas = new int[numLockin];
            string[] rutasFiltros = new string[numLockin];
            int pixelInicial = (int)((configuracion.lambdaIni - lambdas[0]) /
(lambdas[lambdas.Length - 1] - lambdas[0]) * (lambdas.Length - 1));
            int pixelFinal = (int)((configuracion.lambdaFin - lambdas[0]) /
(lambdas[lambdas.Length - 1] - lambdas[0]) * (lambdas.Length - 1));
            int deltaPixel = (pixelFinal - pixelInicial) / (numLockin - 1);

            for (int i = 0; i < numLockin; i++)
            {
                entradas[i] = deltaPixel * i + pixelInicial;//pixel
                lambdasGrafica[i] = lambdas[entradas[i]];
                rutasFiltros[i] = "Fm40-O4-Fc0p53.txt"; //lock-in
            }

            configuracion.confSeñal.frecuencia = frecLibre;
            configuracion.confSeñal.numLockines = numLockin;
            configuracion.confSeñal.entradas = entradas;
            miProcesador.Configurar(configuracion);
```

```
            textBoxFrecMuestreo.Text = configuracion.frecMuestreo.ToString();
            textBoxfrecLibre.Text = configuracion.confSeñal.frecuencia.ToString(nfi);
            textBoxlambdaInicial.Text = configuracion.lambdaIni.ToString();
            textBoxlambdaFinal.Text = configuracion.lambdaFin.ToString();
            textBoxNumlockin.Text = configuracion.confSeñal.numLockines.ToString();
            string texto = configuracion.confSeñal.rutasFiltros[lockinSel];
            labelFiltroLockin.Text = texto.Substring(texto.LastIndexOf("\\") + 1);
            labelEnganchado.BackColor = Color.Red;
            labelEnganchado.Text = "Desenganchado";
            numericUpDownLockin.Maximum = configuracion.confSeñal.numLockines - 1;
            numericUpDownLockin.Value = lockinSel;
        }
        private bool LeerParametros()
        {
            try
            {
                double periodoProv = 1.0 / Convert.ToDouble(textBoxFrecMuestreo.Text, nfi);
                double divisor = Math.Round(periodoProv / 5e-8, 0);
                configuracion.frecMuestreo    =    Convert.ToDouble(textBoxFrecMuestreo.Text,
nfi);//(1/(5e-8*divisor));
                configuracion.confSeñal.frecuencia = Convert.ToDouble(textBoxfrecLibre.Text,
nfi);
                configuracion.lambdaIni = Convert.ToInt32(textBoxlambdaInicial.Text);
                configuracion.lambdaFin = Convert.ToInt32(textBoxlambdaFinal.Text);
                numLockin = Convert.ToInt32(textBoxNumlockin.Text);
                configuracion.promedios = radioButtonConPromedio.Checked;
                Reconfigurar();
                guardar = checkGuardar.Checked;
            }
            catch
            {
                MessageBox.Show("Algún  parámetro  es  incorrecto,  se  vuelve  a  cargar  la
configuración");
                return false;
            }
            return true;
        }

private void PrepararGuardar()
        {
            escritorExcel = new StreamWriter(rutaExcel); //Creamos el fichero de excel
            escritorExcel.WriteLine("fMuestreo:" + configuracion.frecMuestreo.ToString());
            escritorExcel.WriteLine("fLibre:"                                              +
configuracion.confSeñal.frecuencia.ToString());
            escritorExcel.WriteLine("Ganancia:"                                            +
configuracion.confSeñal.gananciaPLL.ToString());
            escritorExcel.WriteLine();
            escritorExcel.WriteLine("Módulo(mVrms)\tFase (º)\tFrecuencia (Hz)\tEnganche");
        }

        private void buttonExaminarFiltroLockin_Click(object sender, EventArgs e)
        {
            OpenFileDialog dialogoFichero = new OpenFileDialog();
            dialogoFichero.Filter = "Archivo de texto (.txt)|*.txt";
            dialogoFichero.Multiselect = false;
            dialogoFichero.RestoreDirectory = true;

            if (dialogoFichero.ShowDialog() == DialogResult.OK)
            {
```

```
                configuracion.confSeñal.rutasFiltros[lockinSel] = dialogoFichero.FileName;
                labelFiltroLockin.Text                                                    =
dialogoFichero.FileName.Substring(dialogoFichero.FileName.LastIndexOf("\\") + 1);
            }
        }


        private void numericUpDownLockin_ValueChanged(object sender, EventArgs e)
        {
            lockinSel = (int)numericUpDownLockin.Value;
        }


        private void buttonExaminarGuardar_Click(object sender, EventArgs e)
        {
            SaveFileDialog dialogoFichero = new SaveFileDialog();
            dialogoFichero.Filter = "Archivo .dat (.dat)|*.dat";
            dialogoFichero.RestoreDirectory = true;
            dialogoFichero.CheckPathExists = true;
            dialogoFichero.AddExtension = true;
            if (dialogoFichero.ShowDialog() == DialogResult.OK)
            {
                rutaExcel = dialogoFichero.FileName;
                textBoxRutaGuardar.Text                                                   =
dialogoFichero.FileName.Substring(dialogoFichero.FileName.LastIndexOf("\\") + 1);
            }
        }

        private void buttonMedir_Click(object sender, EventArgs e)
        {
            if (!miProcesador.Midiendo)
            {
                Medir();
            }
            else
            {
                ParарMedida();
            }
        }

        public void Medir()
        {
            if (!LeerParametros())
                return;
            if (guardar)
                PrepararGuardar();
            labelEnganchado.BackColor = Color.Red;
            labelEnganchado.Text = "Desenganchado";
            timerMedida.Enabled = true;
            buttonMedir.Text = "Detener";
            miProcesador.IniciaAdquisicion(checkBusqueda.Checked);
        }

        public void ParарMedida()
        {
            buttonMedir.Enabled = false;
            timerMedida.Enabled = false;
            miProcesador.ParарAdquisicion();
            buttonMedir.Text = "Medir";
            labelEnganchado.Text = "Desenganchado";
```

```
        labelEnganchado.BackColor = Color.Red;
        if (guardar)
            escritorExcel.Close();
        buttonMedir.Enabled = true;
    }

    private void button1_Click(object sender, EventArgs e) //CAPTURAR
    {
        double[] londa = new double[lambdasGrafica.Length];
        double[] medida = new double[lambdasGrafica.Length];
        lambdasGrafica.CopyTo(londa, 0);
        miProcesador.Modulos.CopyTo(medida, 0);

        StreamWriter escritorEspectro;
        SaveFileDialog dialogoFichero = new SaveFileDialog();
        dialogoFichero.Filter = "Archivo .dat (.dat)|*.dat";
        dialogoFichero.RestoreDirectory = true;
        dialogoFichero.CheckPathExists = true;
        dialogoFichero.AddExtension = true;
        if (dialogoFichero.ShowDialog() == DialogResult.OK)
        {
            escritorEspectro = new StreamWriter(dialogoFichero.FileName);
            escritorEspectro.WriteLine("Longitud de onda (nm)" + "\t" + "Medida");

            for (int i = 0; i < lambdasGrafica.Length; i++)
            {
                escritorEspectro.WriteLine(londa[i] + "\t" + medida[i]);
            }
            escritorEspectro.Close();
        }
    }

    private void timerMedida_Tick(object sender, EventArgs e)
    {
        ejeTiempo += timerMedida.Interval / 1000.0;
        double medida = miProcesador.Modulo(lockinSel);
        labelResultado.Text = medida.ToString("0.000" + " u.a.", nfi);
        if (this.Visible)
        {
            representaGraficaTiempo(medida, ejeTiempo);
            representaGraficaEspectro(miProcesador.Modulos);
        }
        labelFase.Text = miProcesador.Fase(lockinSel).ToString("0.000", nfi) + " º";
        labelFrecuencia.Text = miProcesador.Frecuencia().ToString("0.000", nfi);
        if ((guardar) && (escritorExcel.BaseStream != null))
            escritorExcel.WriteLine(ejeTiempo + "\t" + medida.ToString("0.0000"));

        ruido[contRuido] = medida;
        contRuido++;

        if (contRuido == ruido.Length)
        {
            contRuido = 0;
            inicioRuido = false;
        }

        if (inicioRuido)
        {
            labelRuido.Text = Desviacion(ruido, contRuido).ToString("0.00") + "%";
            buttonReset.BackColor = SystemColors.Control;
```

```
            }
            else
            {
                labelRuido.Text = Desviacion(ruido).ToString("0.00") + "%";
                buttonReset.BackColor = SystemColors.ControlDark;
            }

            OnUnaMedida(miProcesador.Modulos); //produce un evento de medida
        }

        private double Desviacion(double[] vector, int longitud)
        {
            double promedio = 0;
            double desv = 0;
            for (int i = 0; i < longitud; i++)
                promedio = promedio + vector[i];
            promedio = promedio / longitud;
            for (int i = 0; i < longitud; i++)
                desv = desv + (vector[i] - promedio) * (vector[i] - promedio);
            desv = Math.Sqrt(desv / (longitud - 1));
            desv = 100 * desv / promedio;
            return desv;
        }

        private double Desviacion(double[] vector)
        {
            return Desviacion(vector, vector.Length);
        }

        private void buttonReset_Click(object sender, EventArgs e)
        {
            contRuido = 0;
            inicioRuido = true;
        }

        private void radioButtonSinPromedio_CheckedChanged(object sender, EventArgs e)
        {
            if (((RadioButton)sender).Checked)
                miProcesador.Promediado = false;

        }

        private void radioButtonConPromedio_CheckedChanged(object sender, EventArgs e)
        {
            if (((RadioButton)sender).Checked)
                miProcesador.Promediado = true;
        }

        private void Reflectometro_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (e.CloseReason==CloseReason.UserClosing)
            {
                this.Hide();
                if (miProcesador.Midiendo)
                    PararMedida();
                e.Cancel = true;
            }
        }
    }
}
```

Procesador.cs

```
using System;
using System.Threading;
using System.Globalization;
using System.Windows.Forms;
using RgbDriverKit;
using System.Linq;

namespace LockinDAQ
{
    public class Procesador //Este objeto realiza un procesado lock-in usando los canales
indicados de la tarjeta y los filtros
    {
        const double FREC_MAX = 20000;
        bool promediado = false;
        bool midiendo;
        public bool Promediado
        {
            get
            {
                return promediado;
            }
            set
            {
                promediado=value;
            }
        }
        public struct Configuracion
        {
            public Señal.ConfSeñal confSeñal;
            public double frecMuestreo;
            public bool PLL;//si hay o no PLL
            public bool NI;//si es NI o LabJack
            public bool diferencial;//medida diferencial o a tierra
            public double vMin;//máximos y mínimos de la adquisición, para escalar (sólo en
NI)
            public double vMax;
            public double factorModulacion; //si la frecuencia que se proporciona al
modulador no es la frecuencia de la señal (frecuenciaMod=frecuenciaSeñal*factorModulacion)
            public int lambdaIni;
            public int lambdaFin;
            public bool promedios;

            public Configuracion(double vFrecMuestreo, bool vPLL, bool vNI, bool
vDiferencial, double vVMin, double vVMax, int vlambdaIni, int vlambdaFin, bool vPromedios,
double vFactorModulacion=1)
            {
                confSeñal = new Señal.ConfSeñal();
                frecMuestreo = vFrecMuestreo;
                PLL = vPLL;
                NI = vNI;
                diferencial = vDiferencial;
                vMin = vVMin;
                vMax = vVMax;
                lambdaIni = vlambdaIni;
```

```
            lambdaFin = vlambdaFin;
            promedios=vPromedios;
            factorModulacion = vFactorModulacion;
        }
    }

    Configuracion miConf;
    Thread procesoAdquisicion;
    Señal miSeñal;
    NumberFormatInfo nfi;
    Spectrometer miEspectrometro;
    int[] entradasTotal; //Medidas recibidas
    bool buscar;

    public double Modulo(int numLockin)
    {
        if (numLockin< miSeñal.miConf.numLockines)
            return miSeñal.modulos[numLockin];
        else
            return -1;
    }

    public double[] Modulos
    {
        get
        {
            return miSeñal.modulos;
        }
    }

    public double Fase(int numLockin)
    {
        if (numLockin < miSeñal.miConf.numLockines)
            return miSeñal.fases[numLockin];
        else
            return -1;
    }

    public double Frecuencia()
    {
        return miSeñal.miConf.frecuencia;
    }

    public bool Midiendo
    {
        get { return midiendo; }
    }

    public Procesador(Spectrometer vEspectrometro, Configuracion vConf)
    {
        midiendo = false;
        Inicializar(vConf);
        nfi = new NumberFormatInfo();
        nfi.NumberDecimalSeparator = ".";
        miEspectrometro = vEspectrometro;
    }

    public void Inicializar(Configuracion vConf)
    {
        miConf = vConf;
```

```
            miConf.confSeñal.frecMuestreo = miConf.frecMuestreo;
            miConf.confSeñal.PLL = miConf.PLL;
            miSeñal = new Señal(miConf.confSeñal);
            entradasTotal = miSeñal.miConf.entradas;  //asignar las entradas a los datos del
espectrómetro
        }

        public void Configurar(Configuracion vConf)
        {
            miConf = vConf;
            double divisor = Math.Round(48e6 / miConf.frecMuestreo, 0);
            miConf.frecMuestreo = 48e6 / divisor;
            miConf.confSeñal.frecMuestreo = miConf.frecMuestreo;
            miSeñal.Configurar(miConf.confSeñal);
            entradasTotal = miSeñal.miConf.entradas;
        }

        public void CambiarFiltro(string filtroNuevo, int numSeñal, int numLockin)
        {
            miConf.confSeñal.rutasFiltros[numLockin] = filtroNuevo;
            miSeñal.Configurar(miConf.confSeñal);
        }

        public void IniciaAdquisicion(bool vBuscar)
        {
            buscar = vBuscar;
            procesoAdquisicion = new Thread(new ThreadStart(Adquirir));
            procesoAdquisicion.Name = "Proceso Adquisicion";
            procesoAdquisicion.Priority = ThreadPriority.Highest;
            procesoAdquisicion.Start();
        }

        private void Adquirir()
        {
            double[] datosSeñal;
            double[] datosSuma;
            int factorDiezmado;
            midiendo = true;
            float[] intensities;
            datosSeñal = new double[miSeñal.columnasDatos.Length]; //se asigna a cada señal
sus columnas de datos
            miEspectrometro.ExposureTime = (float)(1.0/(2 * miConf.confSeñal.frecMuestreo));

            try
            {
                while (midiendo)
                {
                    Application.DoEvents();
                    miEspectrometro.StartExposure();
                    while(miEspectrometro.Status > SpectrometerStatus.Idle)
                    {

                    }

                    intensities = miEspectrometro.GetSpectrum();

                    if (promediado)
                    {
                        factorDiezmado = 2500 / miSeñal.miConf.numLockines;
```

```
                        datosSuma = new double[factorDiezmado];
                        for (int j = 0; j < miSeñal.columnasDatos.Length; j++)
                        {
                            Array.Copy(intensities, (entradasTotal[j] - factorDiezmado / 2),
datosSuma, 0, factorDiezmado);
                            datosSeñal[j] = datosSuma.Sum()/factorDiezmado;
                        }
                        miSeñal.Procesar(datosSeñal, buscar);
                    }
                    else
                    {
                        for (int j = 0; j < miSeñal.columnasDatos.Length; j++)
                            datosSeñal[j] = (double)intensities[entradasTotal[j]];
                        miSeñal.Procesar(datosSeñal, buscar);
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Cannot take spectrum",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
    }

        public void PararAdquisicion()
        {
            midiendo = false;
        }
    }
}
```