UNIVERSIDAD DE  VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Diseño y fabricación de un sistema electrónico para la caracterización óptica de diodos emisores de luz

Autor:
D. Jorge Fernández Lucas

Tutores:
D. Jesús Arias Álvarez
D. Iván Santos Tejido

Valladolid, 1 de Junio de 2015

TÍTULO:	**Diseño y fabricación de un sistema electrónico para la caracterización óptica de diodos emisores de luz**
AUTOR:	**D. Jorge Fernández Lucas**
TUTOR:	**D. Jesús Arias Álvarez**
	D. Iván Santos Tejido
DEPARTAMENTO:	**Departamento de Electricidad y Electrónica**

TRIBUNAL

PRESIDENTE:	**D. Jesús Hernández Mangas**
VOCAL:	**D. Jesús Arias Álvarez**
SECRETARIO:	**D. Iván Santos Tejido**
SUPLENTE:	**D. Pedro López Martín**
SUPLENTE:	**D. José Vicente Antón**

FECHA:	**1 de Junio de 2015**
CALIFICACIÓN:	

RESUMEN DEL TRABAJO FIN DE GRADO

El propósito de este Trabajo Fin de Grado es el diseño y construcción de un sistema para automatizar la caracterización óptica de diodos emisores de luz. La realización se llevará a cabo mediante un monocromador controlado por una aplicación software que gestiona el movimiento del motor paso a paso, incluido en el monocromador, y un fotodiodo que mide la intensidad de luz que atraviesa el monocromador.

PALABRAS CLAVE

Espectro de emisión, LED, monocromador óptico, fotodiodo, calibración.

ABSTRACT

The aim of this project is the design and construction of a system to automate the optical characterization of Light Emission Diodes. The system will consist on a monochromator controlled by a software application which manages the movement of the stepping motor included in the monochromator, and a photodiode which measures the light intensity that goes through the monochromator.

KEYWORDS

Emission spectrum, LED, monochromator, photodiode, calibration.

AGRADECIMIENTOS

Me gustaría expresar mi agradecimiento a todas aquellas personas que han hecho posible la realización del presente trabajo, en especial a D. Iván Santos Tejido y a D. Jesús Arias Álvarez, por la ayuda y la dedicación que me han prestado durante la realización del mismo, así como sus consejos e indicaciones para tratar de llevar a buen término todo el trabajo realizado.

Asimismo, querría extender mi agradecimiento a todo el Departamento de Electricidad y Electrónica de la Universidad de Valladolid, gracias a ellos y a sus equipos hemos sido capaces, tanto de desarrollar el sistema, como de alcanzar los objetivos propuestos.

Además, me gustaría dar las gracias a mi familia y amigos, ya que han estado ahí siempre que lo he necesitado, dándome su constante apoyo y cariño.

Gracias a todos.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

1.1 MOTIVACIÓN

El objetivo principal del presente Trabajo Fin de Grado es diseñar y construir un sistema para automatizar la obtención del espectro de emisión de LEDs (*Ligth Emission Diodes*, diodos emisor de luz). Este sistema se utilizará para diseñar prácticas de la asignatura optativa Optoelectrónica del 4º curso del Grado en Ingeniería de Tecnologías Específicas de Telecomunicación, mención en Sistemas Electrónicos.

La realización del sistema se ha llevado a cabo en el Departamento de Electricidad y Electrónica de la Universidad de Valladolid, donde hemos podido aprovechar los equipos disponibles, entre ellos: monocromador, fresadora, osciloscopios, fuentes de alimentación, etc.

1.2 INTERÉS DE LOS LEDs

Los LEDs son dispositivos emisores de luz baratos, pequeños, compactos, y versátiles, lo que los hace muy interesantes para ciertas aplicaciones, por ejemplo:

- Fuente de iluminación, siendo menor su consumo y mayor su tiempo de vida frente a las lámparas incandescentes.
- En mandos a distancia y aplicaciones de control remoto se utilizan los LEDs infrarrojos.
- Como indicadores de estado, encendido/apagado.
- En dispositivos de señalización como pueden ser las señales de tráfico o paneles luminosos.
- En los inicios de la fibra óptica se utilizaban LEDs debido a su bajo coste, ahora se opta por los láseres, ya que tienen mejores prestaciones.
- Incluso se pretenden utilizar para iluminar una vivienda y, además, transmitir datos de forma que se cree que podrían reemplazar a las redes de área local (LAN) [1].

1.2.1 Principio de funcionamiento

Un LED es un diodo (una unión p-n) construido con materiales adecuados que emite luz cuando se polariza en directa (es decir, cuando $V_{PN} > 0V$). Al polarizar el LED en directa se favorece la recombinación de portadores banda a banda en la zona de carga espacial, es decir, la ocupación de un hueco de la banda de valencia por parte de un electrón que se encontraba en la banda de conducción, que da lugar a la emisión de fotones. La energía de los fotones emitidos es aproximadamente igual a la energía del GAP (diferencia de energía entre los bordes de las bandas de conducción y valencia del semiconductor) de la zona activa del diodo y, por tanto, la longitud de onda del fotón emitido será:

$$\lambda(\mu m) = \frac{1.24}{E_{gap}(eV)} \tag{1.1}$$

En la expresión anterior podemos observar que si se cambia la energía del GAP del material semiconductor de la zona activa, entonces se cambia la longitud de onda de emisión. Luego si queremos emitir en una cierta longitud de onda, tendremos que elegir el material adecuado de forma que el semiconductor sea directo y la energía de su GAP sea la deseada.

1.2.2 Materiales de interés en Optoelectrónica

Antes de presentar los materiales de interés en Optoelectrónica, describiremos cuáles son las longitudes de onda interesantes en las aplicaciones que usan LEDs. Se denomina espectro electromagnético a la distribución energética del conjunto de las ondas electromagnéticas. El espectro se divide en regiones de forma que su estudio resulta más sencillo. En la figura 1.1 se puede observar el entorno de la región visible. Dentro de las regiones de la figura 1.1, centraremos nuestro interés en las regiones visible e infrarroja, que son las regiones más relevantes para la Optoelectrónica:

- Región visible (390 – 760 nm), en la cual podemos encontrar los colores y especificar el rango de longitudes de onda de cada color:
    - Violeta (390 – 455 nm).
    - Azul (455 – 492 nm).
    - Verde (492 – 577 nm).
    - Amarrillo (577 – 597 nm).
    - Naranja (597 – 622 nm).
    - Rojo (622 – 760 nm).
- Región infrarroja (0.76 - 300µm), dentro de esta región nos centraremos en la banda infrarroja cercana (NIR: 0.76 – 2 µm), ya que en esta banda se encuentran las ventanas de comunicaciones ópticas:
    - Primera ventana (850 – 950 nm).
    - Segunda ventana: 1300 nm.
    - Tercera ventana: 1550 nm.

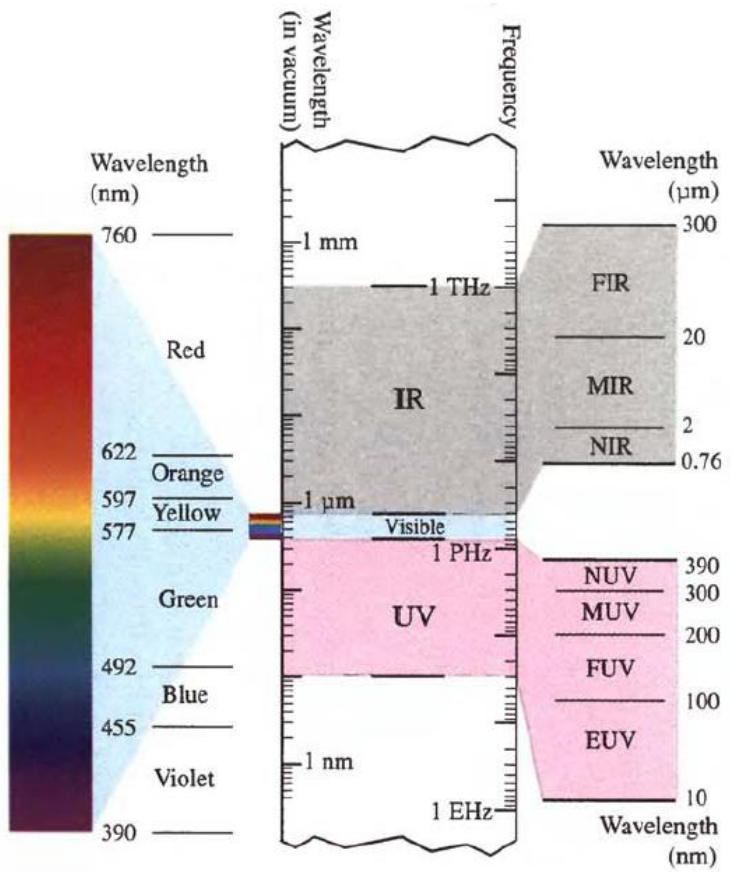


Figura 1.1: Frecuencias y longitudes de onda ópticas. La región infrarroja (IR) del espectro comprende las bandas infrarroja cercana (NIR), infrarroja media (MIR) e infrarroja lejana (FIR), mientras que la región ultravioleta (UV) comprende las bandas ultravioleta cercana (NUV), ultravioleta media (MUV), ultravioleta lejana (FUV) y ultravioleta extrema (EUV). La radiación en la banda EUV es conocida como rayos X blandos [2].

Una vez descritas las regiones de interés del espectro electromagnético, es necesario encontrar los materiales adecuados para emitir en esas longitudes de onda. Antes debemos distinguir entre dos tipos de semiconductores [2]:

- Semiconductores con GAP directo: Son los materiales semiconductores en los que el mínimo de energía de la banda de conducción y el máximo de energía de la banda de valencia coinciden en el mismo número de onda k (mismo momento). El **GaAs** es un semiconductor directo como puede apreciarse en la parte derecha de la figura 1.2.
- Semiconductores con GAP indirecto: Son los materiales semiconductores en los que no sucede lo explicado anteriormente. El **Si** es un semiconductor indirecto como puede verse en la parte izquierda de la figura 1.2.

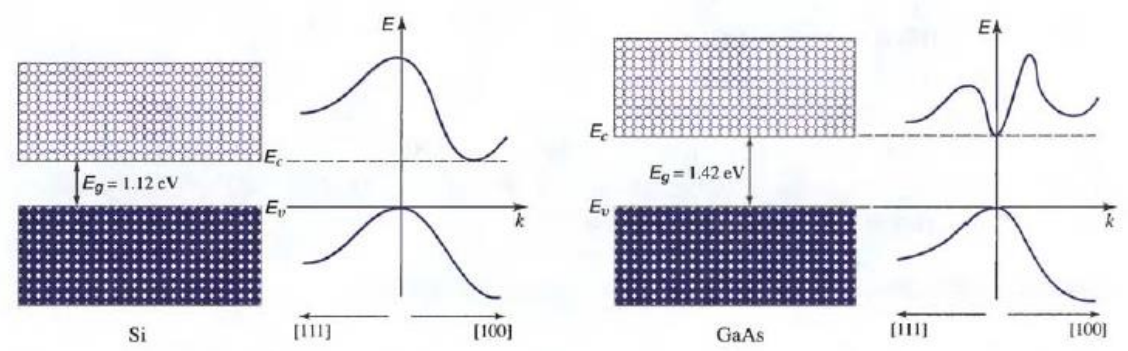


Figura 1.2: Sección transversal de la función E-k (energía frente a número de onda) para **Si** (izquierda) y **GaAs** (derecha) a lo largo de dos direcciones del cristal: [111] hacia la izquierda y [100] hacia la derecha [2].

La distinción es importante, ya que los semiconductores directos como el **GaAs** son emisores eficientes de fotones, mientras que los semiconductores indirectos como el Si no son capaces de emitir fotones eficientemente bajo circunstancias normales. Es por esta razón que, de entre todos los semiconductores existentes, los semiconductores directos son los más adecuados para la emisión de luz de forma eficiente y, por lo tanto, los que nos interesan para la zona activa de los LEDs.

Entonces necesitamos encontrar semiconductores directos con el GAP adecuado para emitir a la longitud de onda deseada. Para ello se realizan aleaciones de diferentes semiconductores binarios formando así aleaciones ternarias o cuaternarias. Dependiendo de cada aleación y de su composición en concreto, se conseguirá un valor determinado de GAP y, por lo tanto, una longitud de onda de emisión.

Procedamos a explicar los semiconductores elementales y compuestos binarios más utilizados en las denominadas familias semiconductoras usadas en Optoelectrónica [2]:

- Semiconductores elementales (IV): El silicio (**Si**) y el germanio (**Ge**) son los semiconductores elementales más importantes de la columna IV de la tabla periódica. Estos materiales no se utilizan a la hora de fabricar emisores de luz debido a su GAP indirecto, pero suelen utilizarse como fotodetectores. En la tabla 1.1 podemos observar la energía del GAP y la longitud de onda asociada siguiendo la ecuación 1.1 de estos materiales.
- Semiconductores binarios (III-V): Son compuestos formados combinando un elemento de la columna III con un elemento de la columna V. Estos compuestos binarios son semiconductores muy importantes en óptica. En la tabla 1.1 se listan estos 12 compuestos con sus características más significativas.

Estos semiconductores se pueden combinar entre sí dando lugar a las aleaciones ternarias o cuaternarias de las denominadas familias de semiconductores.

En la figura 1.3 se resumen las principales combinaciones de semiconductores III-V en función de sus características (parámetro de red, energía del GAP, y longitud de onda asociada). Los semiconductores elementales y binarios se representan con un punto. Los

semiconductores terciarios se ilustran con una línea que une sus dos compuestos binarios. Los semiconductores cuaternarios, en cambio, están representados por un área, ya que tienen un grado de libertad a mayores con respecto a los ternarios.

Compuesto	Tipo de GAP	Energía de la banda GAP E_g (eV)	Longitud de onda λ (nm)
Si	Indirecto	1.12	1110
Ge	Indirecto	0.66	1880
AlN	Directo	6.20	200
AlP	Indirecto	2.45	506
AlAs	Indirecto	2.16	574
AlSb	Indirecto	1.58	785
GaN	Directo	3.39	366
GaP	Indirecto	2.26	549
GaAs	Directo	1.42	873
GaSb	Directo	0.73	1700
InN	Directo	0.65	1910
InP	Directo	1.35	919
InAs	Directo	0.36	3440
InSb	Directo	0.17	7290

Tabla 1.1: Tabla con los semiconductores elementales y compuestos binarios, muestra el tipo de GAP, la energía del GAP y la longitud de onda correspondiente [2].

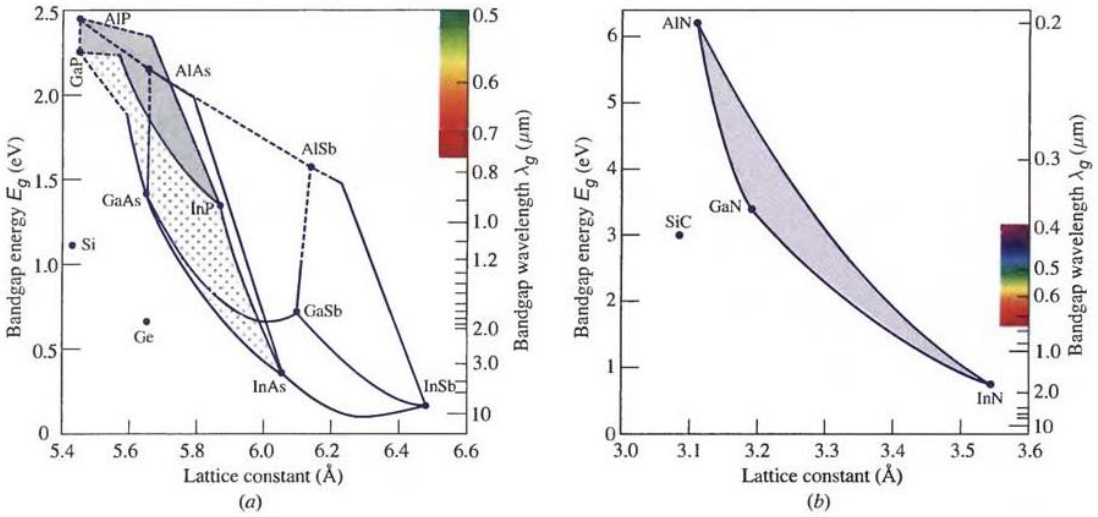


Figura 1.3: Energía de GAP, longitud de onda asociada (ecuación 1.1) y parámetro de red para Si, Ge, SiC y los 12 compuestos binarios III-V de la tabla 1.1. Las curvas de trazo continuo y punteado representan composiciones de GAP directo e indirecto, respectivamente [2].

Veamos ahora cuales de las diferentes combinaciones de materiales semiconductores mostradas en la figura 1.3 son adecuadas para emitir luz en las regiones interesantes del espectro electromagnético (visible e infrarroja cercana) [2]:

- **GaAs**: Es un semiconductor binario (III-V) de GAP directo que se usó para fabricar el primer diodo láser en 1962, emitiendo a 0.873 µm, es decir, en el infrarrojo cercano. Poco después, se crearon otros semiconductores binarios de GAP directo para emitir luz infrarroja: **GaSb** (1.70 µm) y **InP** (0.919 µm), emitiendo en el infrarrojo cercano, y **InAs** (3.44 µm) y **InSb** (7.29 µm), emitiendo en el infrarrojo medio.
- **GaAsP**: A medida que la fracción de fósforo aumenta (**GaAs**$_{1-x}$**P**$_x$) nos adentramos en la región visible del espectro, ofreciendo emisión en la región del color rojo, como podemos ver en la figura 1.3 (a). Aunque el GAP pasa a ser indirecto a partir del rojo, se puede conseguir emitir en el naranja, el amarillo y el verde dopando con nitrógeno (**GaAsP**:**N** y **GaP**:**N**).
- **InGaAsP**: El semiconductor cuaternario **In**$_{1-x}$**Ga**$_x$**As**$_{1-y}$**P**$_y$ es una aleación muy versátil y ampliamente utilizada en la región infrarroja cercana. Su longitud de onda de emisión puede ser seleccionada dentro de amplio un rango (0.549 µm (**GaP**) – 3.44 µm (**InAs**)) variando los ratios de mezclado x e y entre 0 y 1, ver área punteado de la figura 1.3 (a). Solo una parte de este rango disfruta del beneficio de un GAP directo, sin embargo, el **InGaAsP** se utiliza para fabricar LEDs para sistemas de comunicaciones en segunda ventana y para multitud de aplicaciones debido a su bajo coste.
- **AlGaAs**: El semiconductor ternario **Al**$_x$**Ga**$_{1-x}$**As** es un material de GAP directo en las regiones del rojo y del infrarrojo cercano del espectro. Una desventaja de este material es el tiempo de vida limitado del dispositivo asociado a la oxidación y corrosión del material con el tiempo cuando el contenido de aluminio es suficientemente alto.
- **AlInGaP**: El semiconductor cuaternario (**Al**$_x$**Ga**$_{1-x}$)$_y$ **In**$_{1-y}$**P** tiene un GAP directo sobre buena parte de las regiones infrarroja cercana y visible. Este es el material escogido para aplicaciones de alto brillo, tales como semáforos y señalización de tráfico en las regiones del rojo, naranja, ambar (naranja-amarillo) y amarillo.
- **InGaN**: El **In**$_x$**Ga**$_{1-x}$**N** es un semiconductor ternario de GAP directo con una longitud de onda de emisión que abarca desde 366 nm del **GaN** hasta 1.61 µm del **InN**. Sin embargo, es muy difícil crecerlo con alto contenido de **InN**, por lo que solamente se utiliza, para fabricar LEDs de alto brillo, en el rango de 366 nm a 580 nm, comprendiendo las regiones ultravioleta cercana, violeta, azul y verde del espectro (ver figura 1.3 (b)).

Aunque las siguientes combinaciones de materiales semiconductores emiten en la región del ultravioleta, la cual se encuentra fuera de nuestra zona de interés, debemos al menos mencionarlas ya que, aunque no se usan para emitir, sí que se usan dentro de los LEDs rodeando las zonas que emiten luz:

- **AlGaN**: **Al**$_x$**Ga**$_{1-x}$**N** es un semiconductor ternario basado en nitrógeno de GAP directo, pero su longitud de onda de emisión cae entre los 200 nm (**AlN**) y los 366 nm (**GaN**) (fuera de nuestra región de interés, ver figura 1.3 (b)), cubriendo las regiones media y cercana del ultravioleta, que van de los 200 nm hasta los

390 nm. Plantillas de **AlGaN**/**AlN**/zafiro sirven como sustratos transparentes para emisores ultravioletas basados en **AlGaN**. El uso de **GaN** como un sustrato es evitado a causa de su absorción a longitudes de onda inferiores a 366 nm.

- **AlInGaN**: El semiconductor cuaternario (**$Al_xIn_yGa_{1-x-y}$N** tiene una ventaja respecto de los compuestos ternarios **InGaN** y **AlGaN**, de los que ya hablamos anteriormente, y es que puede fabricarse eligiendo adecuadamente el valor de los ratios de mezclado x e y, de forma que tienda hacia el **GaN** (ver figura 1.3 (b)), incrementando de ese modo la eficiencia cuántica. Los LEDs de **AlInGaN** con esos ratios de mezclado tienen un rango de emisión que va desde los 366 nm (**GaN**) hasta los 250 nm (**AlInN**), este rango de emisión está fuera de nuestra región de interés, pero el **AlInGaN** también puede utilizarse como capa de contacto transparente.

1.2.3 Espectro de emisión del LED

El valor obtenido por la expresión 1.1 representa la longitud de onda máxima de emisión, pero no solo se emite a esa longitud de onda, tendremos un ancho espectral de emisión. En la figura 1.4 podemos apreciar el espectro de emisión de tres LEDs de diferentes colores.

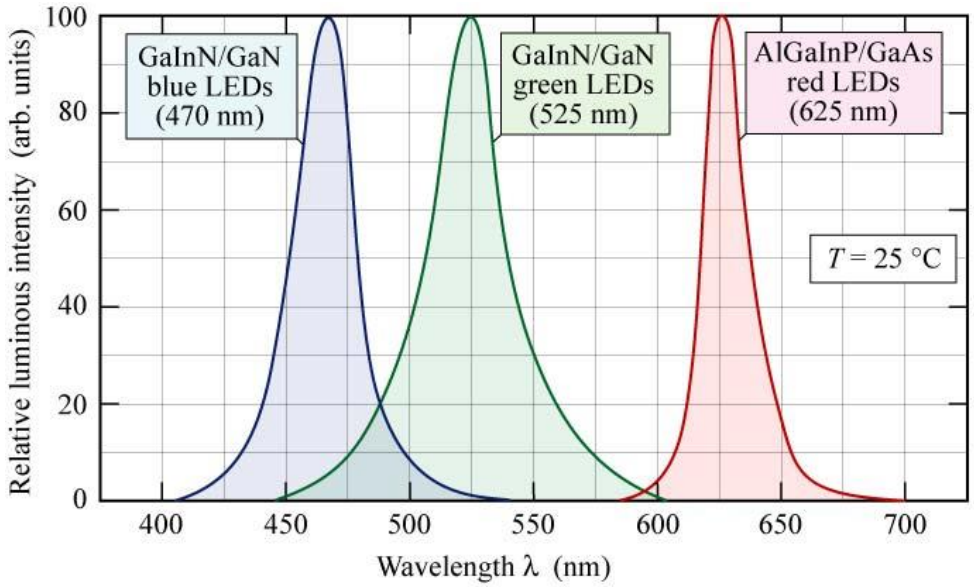


Figura 1.4: Espectro emitido por LEDs rojo, verde y azul [3].

El LED presenta un ancho espectral asociado a la temperatura de emisión que viene dado (en μm) por la siguiente expresión:

$$\Delta\lambda = 1.45\lambda^2 K_B T \tag{1.2}$$

El aumento de la temperatura produce el aumento del ancho espectral de emisión. Esto es debido a que aumenta la probabilidad de recombinación banda-centro, dando lugar a emisiones con otras longitudes de onda distintas. En ocasiones, la calidad estructural del

LED no es demasiado buena, lo que hace que la anchura espectral sea mayor que la debida al efecto de la temperatura.

1.3 DESCRIPCIÓN DEL SISTEMA Y OBJETIVOS

El objetivo principal es, como ya mencionamos anteriormente, diseñar y construir un sistema para automatizar la obtención del espectro de emisión de LEDs. Para eso, necesitamos los siguientes elementos [4]:

- Un dispositivo que controle la longitud de onda en la que nos encontramos, que será un monocromador. Nosotros contamos con el monocromador MonoSpec 10, que tiene un motor incorporado para mover su red de difracción y cambiar así la longitud de onda que lo atraviesa.
- Un dispositivo que nos mida la intensidad de la luz que atraviesa el monocromador, que será un fotodiodo (PD, *photodiode*). El fotodiodo tendremos que elegirlo en función de las longitudes de onda que nos interesan. En nuestro caso hemos elegido el fotodiodo TEMD5010X01 cuyo rango espectral va desde los 430 nm hasta los 1100 nm, que cubre el rango visible del espectro e incluso parte del infrarrojo cercano.
- Un dispositivo para interpretar los datos provenientes del fotodiodo, que será un microcontrolador. En nuestro sistema hemos utilizado el microcontrolador LPC2103, que cuenta con un ADC (*Analog-to-Digital Converter*), para poder digitalizar la señal procedente del fotodiodo. El microcontrolador podrá encargarse además de controlar el motor del monocromador y el LED (encenderlo y apagarlo).
- Por último, necesitamos un dispositivo para representar los datos obtenidos y obtener así la curva del espectro de emisión del LED. Además, nos interesaría poder crear una interfaz para que un usuario pueda controlar el sistema. El candidato idóneo para todo ello es un PC.

Teniendo en cuenta todos los dispositivos necesarios, el diseño esquemático del sistema desarrollado se muestra en la figura 1.5.

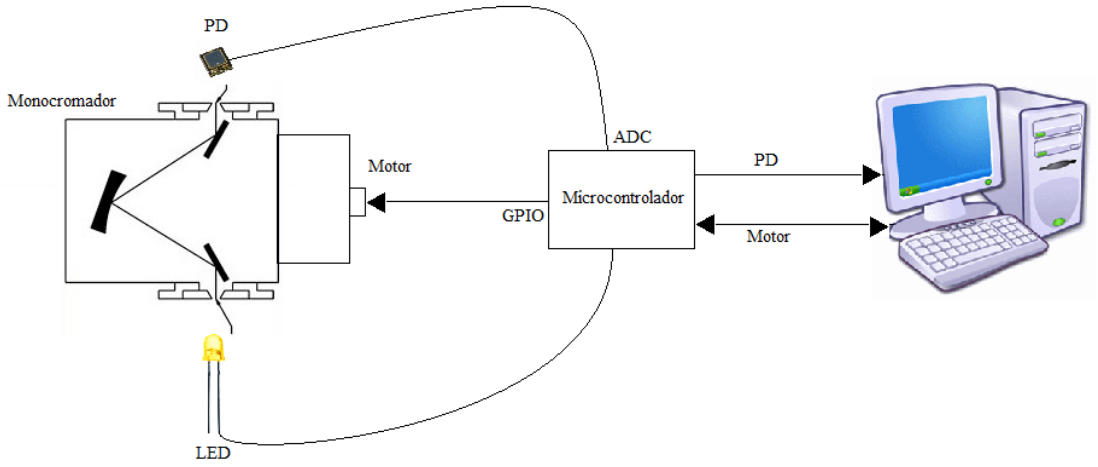


Figura 1.5: Representación esquemática de los componentes y conexiones del sistema diseñado [4].

Para poder controlar el sistema desde el PC necesitaremos que este se comunique con el microcontrolador, que hará de intermediario entre el PC y el resto de elementos. Entonces, las interconexiones del microcontrolador con el resto de dispositivos serán las siguientes:

- Se conecta al PC mediante un conector USB, que le otorga los 5 voltios de alimentación necesarios para el microcontrolador.
- Se conecta al motor a través de un conector de 15 pines. Los puertos GPIO del microcontrolador que van al motor se han conectado aprovechando un cable Ethernet modificado unido a un conector hembra de 15 pines.
- Se conecta al fotodiodo a través de un cable apantallado para evitar interferencias en las lecturas del fotodiodo.
- Por último, debería conectarse al LED, pero en su lugar, se conecta a la montura que sujetará el LED, de forma que cambiarlo sea lo más sencillo posible (esto nos interesa ya que no deseamos que nuestro sistema sea capaz de caracterizar un único LED).

Hasta aquí llegaría el diseño de la parte *hardware* del sistema (elementos e interconexiones que lo componen). El sistema final construido puede verse en la figura 1.6.

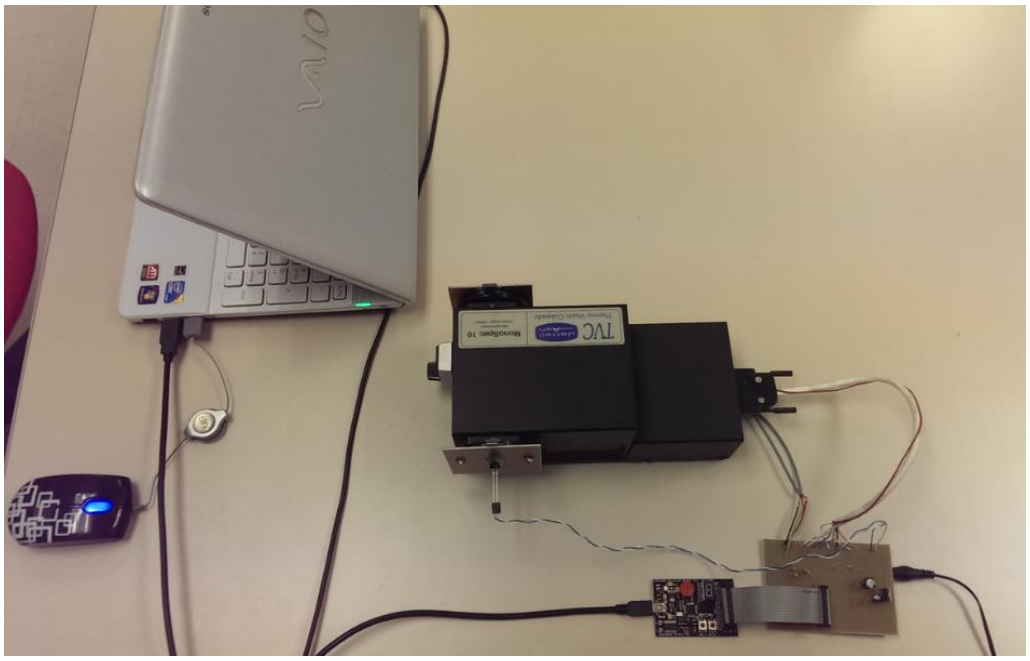


Figura 1.6: Sistema diseñado y construido durante las prácticas en empresa [4].

Durante las prácticas en empresa en el GIR de Electrónica de la Universidad de Valladolid se llevó a cabo el diseño de la parte *hardware* del sistema, que incluye todos los elementos mencionados anteriormente y las interconexiones entre ellos.

El objetivo principal del presente Trabajo Fin de Grado es el desarrollo del *software* de control del sistema, el cual llevó consigo la realización de las siguientes tareas:

- Programación del microcontrolador.
- Desarrollo de la interfaz para el control del sistema desde un PC.
- Comunicaciones entre el microcontrolador y el PC.
- Calibración de la respuesta en frecuencia del fotodiodo.

1.4 CONOCIMIENTOS PREVIOS

En este apartado comentamos los conocimientos adquiridos durante los estudios universitarios que han resultado de utilidad para el desarrollo de este Trabajo Fin de Grado (TFG).

Lo primero de todo, hablar sobre el estudio realizado sobre los componentes ópticos del sistema como son el monocromador, el *grating*, el LED y el fotodiodo. Hemos tenido que recopilar información sobre todos estos componentes, los cuales ya habían sido estudiados en las asignaturas de Sistemas de Comunicaciones Guiadas y Sistemas de Comunicaciones Ópticas, por lo que partíamos una base.

En cuanto a la programación de la aplicación en el PC y en el microcontrolador, las asignaturas como Programación, Ingeniería de Sistemas Software o Desarrollo de Aplicaciones Distribuidas nos han servido de gran ayuda, ya que en ellas aprendimos a programar en lenguaje C y C++.

En lo referente al microprocesador, ya habíamos estudiado este mismo modelo de microprocesador y habíamos programado sobre él en la asignatura de Desarrollo Práctico de Sistemas Electrónicos, lo que nos ha ayudado mucho en el desarrollo de este TFG. Además, habíamos estudiado el funcionamiento del ADC en Instrumentación de Equipos Electrónicos y en Sistemas de Comunicación, aunque en menor detalle. También habíamos estudiado acerca de temporizadores en Sistemas Electrónicos Basados en Microprocesador, pero ha sido ahora cuando hemos podido utilizarlos de manera práctica.

En definitiva, aunque muchas de las tareas realizadas ya las habíamos estudiado, ese estudio había sido solamente teórico y ha sido en este TFG donde hemos tenido la oportunidad de llevar a la práctica todo lo estudiado y ver que las cosas funcionan, aunque no tan idealmente como suponemos en la teoría.

1.5 ESTRUCTURA DE LA MEMORIA

La presente memoria se estructura en 7 capítulos en los que se expondrán las tareas realizadas así como los resultados obtenidos durante la realización del presente Trabajo Fin de Grado (TFG):

- En el capítulo actual se presenta el objetivo principal del sistema a diseñar, se muestra nuestro interés por los LEDs y se da una breve pincelada sobre su funcionamiento, además, se muestra el diseño y la fabricación realizados durante las prácticas en empresa.

- En el segundo capítulo explicamos, detalladamente, todos y cada uno de los dispositivos que componen el sistema: monocromador, LED, fotodiodo, PC microcontrolador y PCB.
- En el tercer capítulo hablaremos sobre las tareas que el sistema es capaz de realizar y sobre cómo desarrollamos el código necesario para que un usuario cualquiera pueda controlar todo el sistema desde un PC y, además, pueda ejecutar las diferentes tareas.
- En el cuarto capítulo comentaremos las diferentes funciones que el microcontrolador deberá realizar y cómo el PC le indica cuál de ellas realizar en cada momento. Por lo tanto, veremos detalladamente las comunicaciones entre el PC y el microcontrolador, así como el lenguaje de comandos ideado para este fin. Además, daremos todos los detalles importantes sobre el código grabado en la memoria flash del microcontrolador.
- En el quinto capítulo explicaremos por qué es necesaria la calibración y cómo la llevamos a cabo: cuantificar el fondo de escala de ruido, corregir la forma de la señal obtenida por el fotodiodo mediante la respuesta espectral del mismo y dar magnitud (unidades) a esa señal obtenida. Además, por último, comentaremos como incorporamos todo esto al código de la aplicación.
- En el sexto capítulo presentaremos varios ejemplos de medidas experimentales realizadas. Mostramos los espectros de emisión obtenidos para varios LEDs de diferentes colores y los comparamos con los ilustrados por sus hojas de especificaciones.
- En el séptimo capítulo recogeremos las conclusiones más importantes obtenidas de este Trabajo Fin de Grado, así como las posibles mejoras y el trabajo futuro para el sistema. Además, hemos añadido un apartado de opinión personal sobre lo aprendido durante la realización de este trabajo.
- La última sección recopila toda la bibliografía empleada para la realización del sistema y de la presente memoria, además de un apéndice que muestra el contenido del CD, en el que se presenta dicho trabajo, a modo de resumen.

CAPÍTULO 2. COMPONENTES DEL SISTEMA

2.1 INTRODUCCIÓN

En el presente capítulo describiremos en detalle los componentes de que consta el sistema desarrollado. Para ello, primero nombraremos estos componentes y posteriormente pasaremos a explicarlos más profundamente. El sistema consta de los siguientes elementos:

- Monocromador.
- Diodo LED.
- Fotodiodo.
- Microcontrolador.
- PC.
- Placa de circuito impreso (PCB).

2.2 MONOCROMADOR

Un monocromador es un dispositivo óptico que permite, por medio de un mecanismo, seleccionar y transmitir una estrecha banda de longitudes de onda, ya sean electromagnéticas o no, a partir de una fuente emisora que produzca una amplia gama de longitudes de onda.

Nosotros, en particular, contamos con el monocromador MonoSpec 10 del que podemos ver un esquema en la figura 2.1, además, el apéndice A nos explica dónde podemos encontrar las hojas de especificaciones del mismo. El monocromador consta de los siguientes elementos:

- Dos ranuras, una de entrada y otra de salida.
- *Grating* (rejilla óptica), utilizado para separar espacialmente los colores (o longitudes de onda) de la luz.
- Motor paso a paso, utilizado para mover el *grating* y seleccionar así la longitud de onda que deseamos enfocar hacia la ranura de salida.

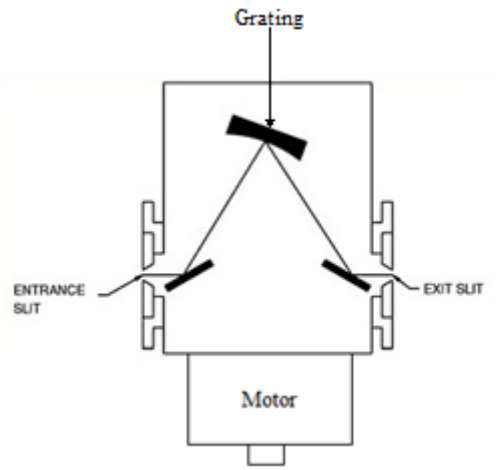


Figura 2.1: Esquema del monocromador MonoSpec10 [5].

En la figura 2.1 podemos apreciar como el rayo de luz entra por la ranura y es reflejado por un espejo hacia el *grating*, que se encargará de separar espacialmente sus distintas componentes espectrales. Una de esas componentes espectrales será reflejada por el segundo espejo y enfocada hacia la ranura de salida del monocromador. No nos quedaremos con una única longitud de onda a la salida, eso es imposible. En nuestro caso, el MonoSpec 10 nos ofrecería una resolución de 1 nm con un *grating* de 1200 g/mm, nosotros contamos con un *grating* de 600 g/mm por lo que cabe pensar que la resolución con la que contamos es de 2 nm.

2.2.1 *Grating*

Un *grating* es un componente óptico con un patrón regular, que divide (difracta) la luz en varios haces que viajan en diferentes direcciones [4]. Las direcciones de esos haces dependen del espaciado entre las ranuras (escalones) del *grating* y de la longitud de onda de la luz incidente, de modo que el *grating* actúa como un elemento dispersivo. Gracias a esto, los *gratings* se utilizan habitualmente en monocromadores (como es nuestro caso) y espectrómetros.

El *grating* con el que cuenta nuestro monocromador es de 600 g/mm (como ya mencionamos anteriormente), esto quiere decir que hay 600 ranuras (escalones) por milímetro. Si tuviésemos más escalones por milímetro, los haces de luz estarían más separados y nos sería más sencillo reducir el rango de longitudes de onda que tenemos a la salida del monocromador, es decir, obtendríamos mayor resolución.

En función de la orientación del *grating*, se deja pasar una u otra longitud de onda por el monocromador. Entonces, la manera de elegir qué longitud de onda queremos a la salida es seleccionar la posición del *grating*. El MonoSpec 10 cuenta con un motor paso a paso que se encarga del movimiento del *grating*, así como de controlar su posición, y que procedemos a explicar a continuación.

2.2.2 Motor paso a paso

El motor paso a paso, incorporado en el monocromador MonoSpec 10, es el que se encarga de controlar la posición del *grating* y así la longitud de onda que se deja pasar. El monocromador, además de tener un motor incorporado, tiene también incorporada una lógica que detecta los finales de carrera (es decir, cuando el espejo llega a sus posiciones extremas para prevenir superarlas). El motor y la lógica de los finales de carrera necesitan una alimentación de 12 y 5 V, respectivamente.

El motor paso a paso es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que significa que es capaz de avanzar una serie de grados (paso) dependiendo de sus entradas de control [4]. Para entender su funcionamiento fijémonos en la imagen de la figura 2.2. En la imagen pueden observarse los 15 pines de que consta el conector del monocromador. Los pines del 1 al 10 controlan la lógica de los finales de carrera, mientras que el resto de los pines son para el control del motor paso a paso, el cual podemos observar en más detalle a la derecha de la imagen.

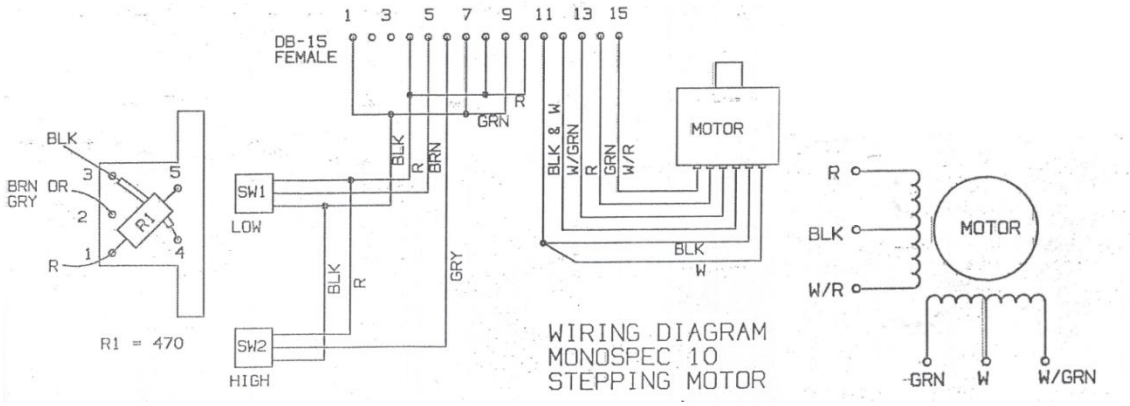


Figura 2.2: Esquema sobre el conector de 15 pines del monocromador. Imagen obtenida de la hoja de especificaciones del monocromador MonoSpec 10 [5].

Observamos que el motor paso a paso consta de una serie de bobinas, las cuales generarán un campo magnético dependiendo de la corriente que circule por ellas. La variación de la dirección de ese campo magnético producirá movimiento de seguimiento por parte del rotor de imán, el cual intentará alinearse con el campo magnético inducido por las bobinas.

Los pines 11, 12, 13, 14 y 15 controlan esas bobinas. El pin 11 corresponde a la toma central de las bobinas, mientras que los otros pines permiten mover el motor haciendo circular la corriente por las bobinas. En nuestro, diseño durante las prácticas en empresa, decidimos conectar el pin 11 a alimentación (12 V), entonces el resto de los pines son activos en baja, ya que no circula ninguna corriente cuando se encuentran conectados a la alimentación. Si hubiésemos conectado la toma central de las bobinas (pin 11) a tierra, entonces el resto de los pines habrían sido activos en alta.

Para controlar las bobinas del motor paso a paso, necesitaremos conectar los pines que controlan estas bobinas con los pines de salida GPIO del microcontrolador. Esta conexión no puede realizarse directamente ya que, como veremos más adelante, el microcontrolador no puede ofrecernos 12 V a la salida de sus pines, entonces no conseguiríamos que las bobinas dejasen de conducir. En este punto, se presenta la necesidad de diseñar un circuito para el acondicionamiento de las salidas GPIO del microcontrolador que controlan los pines del motor paso a paso.

Pues bien, este diseño se llevó a cabo durante las prácticas en empresa y puede verse a continuación, en la figura 2.3. Los transistores MOSFET (utilizados como interruptores) son los PMV37EN de tipo N y los diodos son los PMEG4005.

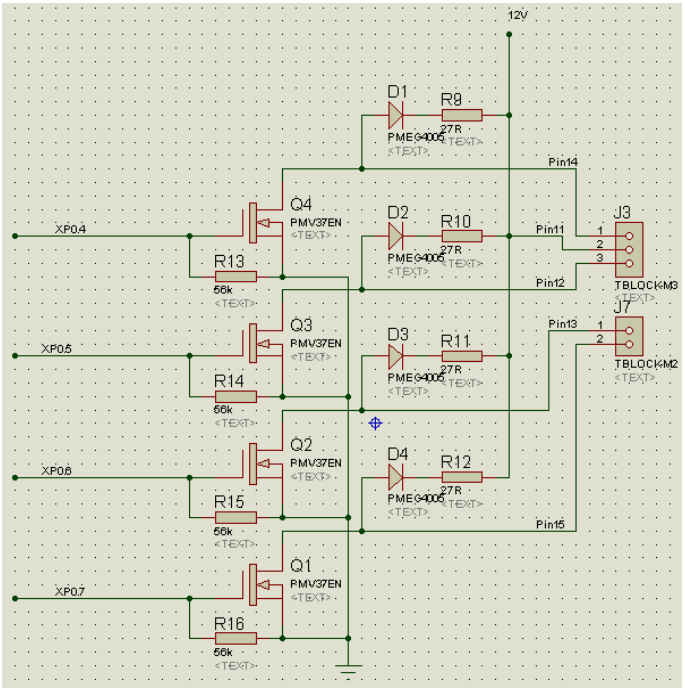


Figura 2.3: Circuito esquemático para el control del motor paso a paso mediante los pines GPIO del microcontrolador [4].

Cuando en el circuito de la figura 2.3 se activa, por ejemplo, la salida XP0.4 del microcontrolador (es decir, se pone en alta), el transistor Q4 pasará a conducir, y el pin 14 estará a una tensión próxima a cero ya que la resistencia del canal del MOSFET es despreciable. Si por el contrario desactivamos la salida XP0.4 (es decir, se pone en baja), el transistor Q4 pasará a corte, por lo que el pin 14 estará conectado a la alimentación de 12 V.

Luego cuando el microcontrolador activa una de sus salidas GPIO, el pin correspondiente del motor es puesto en baja, y viceversa. Entonces, a la hora de programar el microcontrolador (explicado en detalle en capítulo 4), podremos ver esos pines de salida GPIO que controlan el motor como si fuesen activos en alta.

2.3 LED

No olvidemos que el objetivo principal de nuestro sistema es la caracterización de LEDs. El LED a caracterizar se encuentra situado en una de las ranuras del monocromador. No utilizaremos ningún LED en concreto, sino que queremos que cambiarlo sea lo más sencillo posible.

Durante las prácticas en empresa se diseñó una placa PCB para acoplar el LED a la ranura del monocromador de la mejor manera posible, podemos observar el resultado de las dos caras de la placa en la figura 2.4. Apreciamos que la placa tiene dos agujeros laterales para sujetarla al monocromador mediante tornillos, y un tercer agujero central, con una montura (la montura se puede apreciar en la figura de la izquierda), para poder acoplar LEDs de 5 mm de diámetro [4].

Figura 2.4: PCB para acoplar el LED al monocromador [4].

Los LEDs (*Light-Emitting Diodes*), como su propio nombre indica, son diodos (estructuras p-n) en los que se produce la emisión de luz al ser polarizados en directa. Este efecto se debe a la luminiscencia por recombinación banda a banda. Si los electrones y huecos están en la misma región, pueden recombinarse, es decir, los electrones pueden pasar a "ocupar" los huecos "cayendo" desde un nivel energético superior a otro inferior más estable, emitiendo un fotón. La energía del fotón emitido ($E=h\upsilon$, donde h es la constante de Planck y υ es la frecuencia del fotón) será igual a la energía de la banda prohibida (E_g) del material semiconductor. Entonces, diferentes materiales emitirán fotones con distintas frecuencias y, por lo tanto, emitirán luz de distintos colores.

2.3.1 Características I – V

La característica I – V (corriente - voltaje) de un diodo de unión p-n viene dada por la ecuación:

$$I = I_s(e^{eV/kT} - 1) \tag{2.1}$$

Donde *Is* es la corriente de saturación inversa, *e* es la carga del electrón, *K* es la constante de Boltzman y *T* es la temperatura. A veces, el comportamiento de la unión p-n se aproxima por el de un interruptor ideal como se muestra en la figura 2.5. A partir del voltaje umbral (V_γ = *~0.7 V*, para el caso de un diodo de silicio) el diodo conduce. Entonces, nuestro interés se centrará sobre todo en la obtención de ese voltaje umbral.

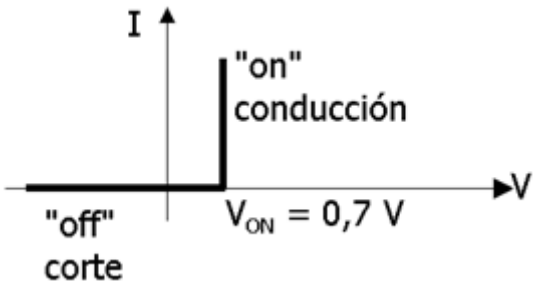


Figura 2.5: Característica corriente – voltaje ideal de una unión p-n de silicio.

Empíricamente, se obtiene que la tensión umbral (V_γ) se puede estimar como la energía del GAP (E_g) menos 0.3 – 0.4 V.

$$V_\gamma \approx \frac{E_g}{q} - [0.3 - 0.4]\ V \tag{2.2}$$

En la figura 2.6 podemos apreciar distintas características I – V para diodos hechos con diferentes materiales semiconductores, además de la energía del GAP de estos materiales. Si observamos el voltaje umbral experimental mostrado en la figura y lo comparamos con la energía del GAP de estos materiales, podemos comprobar que se cumple la estimación de la tensión de disparo de la ecuación (2.2).

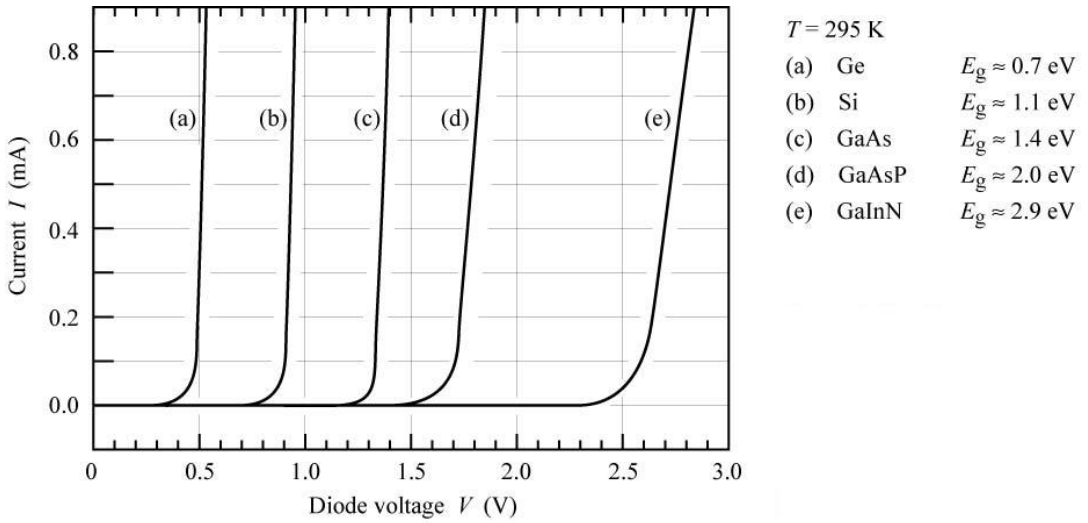


Figura 2.6: Característica corriente – voltaje de una unión p-n hecha con diferentes materiales semiconductores [3].

Entonces, si la energía del GAP aumenta, también se incrementará el voltaje umbral a partir del cual el LED comienza a emitir luz. Pues bien, en nuestro sistema la alimentación para el LED nos la otorga el microcontrolador, el cual como máximo puede darnos *3.3 V*, luego estamos limitados en cuanto a los LEDs que podemos utilizar, ya que solo podremos utilizar aquellos que tengan un voltaje umbral inferior a

3.3 V. Por ejemplo, no podremos utilizar LEDs azules (**AlGaInN**) o ultravioletas (**AlGaN**), ya que su voltaje umbral es superior al que es capaz de ofrecernos el microcontrolador. En la tabla 2.1 podemos ver la energía del GAP y el voltaje umbral para los materiales mostrados anteriormente en la figura 2.6 y así observar que, efectivamente, existe una diferencia entre ambos de 0.3 – 0.4 V.

Semiconductor	E_g (eV)	Vγ (V)
Ge	0.7	0.4
Si	1.1	0.8
GaAs	1.4	1.2
GaAsP	2.0	1.6
GaInN	2.9	2.5

Tabla 2.1: Tabla que contiene la energía del GAP y el voltaje umbral para algunos materiales. Puede observarse que están relacionados, cuando la energía del GAP aumenta, también lo hace el voltaje umbral.

2.3.1.1 Desviaciones de la característica I – V ideal

La ecuación de Shockley (2.1) da la característica teórica I – V de una unión p-n. Para describir las características medidas experimentalmente, esta ecuación suele escribirse de la siguiente manera:

$$I = I_S e^{eV/(n_{ideal}kT)} \tag{2.3}$$

Donde n_{ideal} es el factor de idealidad del diodo:

- n_{ideal} *= 1.0.* Caso ideal, la corriente del diodo está dominada por la difusión de portadores.
- n_{ideal} *= 2.0.* La corriente del diodo está dominada por la generación y recombinación de portadores en la ZCE (Zona de Carga Espacial).
- n_{ideal} *= 1.1 – 1.5*. Valores típicamente tomados para diodos reales.

Los diodos tienen, con frecuencia, resistencias parásitas. El efecto de las resistencias parásitas en serie y en paralelo puede verse en la figura 2.7.

Una resistencia en serie puede ser causada por una resistencia de contacto excesiva o por la resistencia de la región neutral. Una resistencia en paralelo puede ser causada por imperfecciones en la zona de la unión p-n o en la superficie.

La característica I – V del diodo dada por la ecuación de Shockley debe ser modificada para tener en cuenta estas resistencias parásitas. La característica I – V de un diodo de unión p-n polarizado en directa viene dada por:

$$I - \frac{V - IR_S}{R_P} = I_S e^{(V-IR_S)/(n_{ideal}kT)} \tag{2.4}$$

Donde R_S es la resistencia en serie y R_P es la resistencia en paralelo. Para $R_P \rightarrow \infty$ y $R_S \rightarrow 0$, esta ecuación se reduce a la ecuación de Shockley.

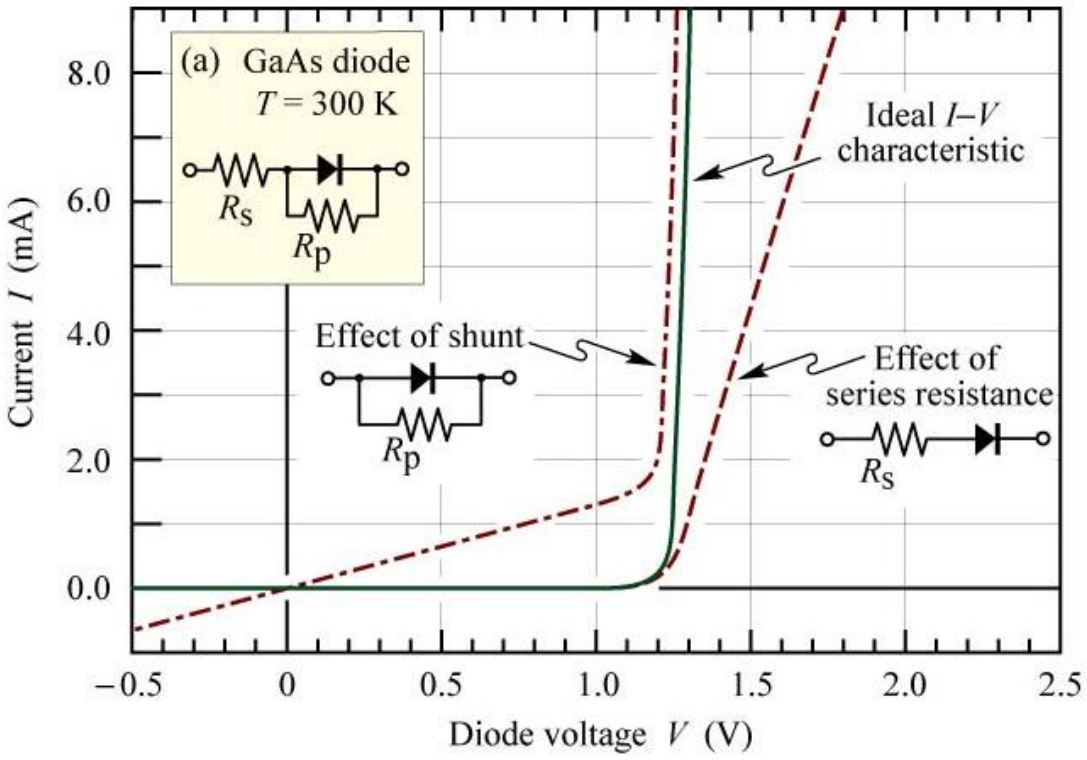


Figura 2.7: Efecto de las resistencias parásitas en serie y en paralelo sobre la característica corriente – voltaje de una unión p-n [3]. Muestra los circuitos utilizados para modelar el comportamiento real de los diodos.

2.3.2 Responsividad y eficiencia de conversión de potencia

La responsividad de un LED es la razón de potencia emitida (P_o) a corriente inyectada (I_F). Generalmente se expresa en unidades de *W/A*, y cuando la longitud de onda se expresa en micrómetros puede calcularse como:

$$R = \frac{P_o}{I_F} \tag{2.5}$$

Por último, la eficiencia de conversión de potencia total viene dada por:

$$\eta_{Total} = \frac{P_o}{P_E} = \frac{P_o}{I_F V_F} \tag{2.6}$$

Donde P_E es la potencia eléctrica aplicada.

2.4 FOTODIODO

Para poder caracterizar un LED mediante nuestro sistema, además del monocromador para seleccionar una longitud de onda, necesitaremos un dispositivo capaz de medir la potencia óptica emitida por el LED a esa longitud de onda. Pues bien, uno de los dispositivos capaces de realizar esta tarea es el fotodiodo, del cual procederemos a explicar el funcionamiento más adelante.

Por lo tanto, el fotodiodo tendrá que colocarse a la salida del monocromador, es decir, perfectamente alineado con la ranura de salida. Y la mejor forma de acoplarlo en esa posición es mediante una pequeña placa PCB, como ya hicimos para acoplar el LED en la ranura de entrada. El diseño de esta placa puede apreciarse en la figura 2.8, este diseño fue llevado a cabo durante las prácticas en empresa.

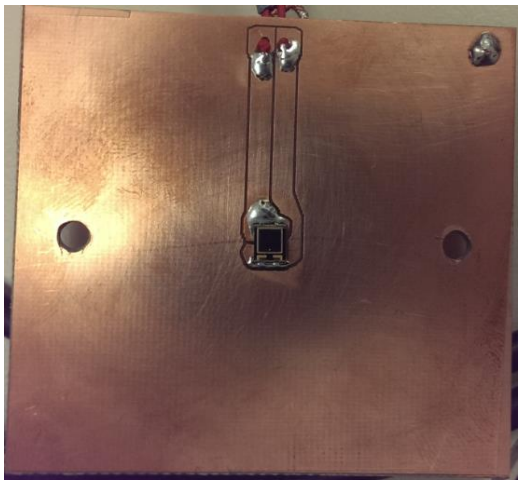


Figura 2.8: PCB fabricada para acoplar el fotodiodo a la ranura de salida del monocromador [4].

El objetivo primordial de esta placa, además de acoplar el fotodiodo, es aislarlo de la luz exterior lo máximo posible, para evitar así la intrusión de ruido en el sistema. Es por esto que hemos utilizado cables apantallados, para conectar el fotodiodo al microcontrolador, y una placa con capa metálica por los dos lados, mejorando así el apantallamiento. En la parte superior derecha de la figura puede observarse la toma de tierra del cable apantallado. También podemos ver los dos agujeros laterales para sujetarla al monocromador, igual que en la placa para acoplar el LED. Además, podemos apreciar que hemos utilizado un fotodiodo de montaje superficial, en concreto el fotodiodo TEMD5010X01, del cual hablaremos en más detalle posteriormente.

2.4.1 El fotodiodo p-n

Un fotodiodo es una unión p-n, expuesta a la luz a través de una abertura cristalina (a veces en forma de lente), cuya corriente inversa se ve incrementada cuando absorbe fotones, cuya energía $h\upsilon$ excede la energía de la banda prohibida (GAP) del material. Su diseño y construcción están orientados a lograr que su sensibilidad a la luz sea máxima. Los fotodiodos se comportan como células fotovoltaicas debido a su construcción, es decir, en ausencia de luz generan cierta tensión en sus bornes. Esta corriente presente en ausencia de luz recibe el nombre de corriente de oscuridad. Cuanto menor sea la corriente de oscuridad, mayor será la sensibilidad del fotodiodo, ya que podrá medir intensidades de luz que destaquen menos sobre el fondo de ruido [2 y 6].

Imaginemos que aplicamos un voltaje de polarización en inversa sobre la unión p-n (el fotodiodo), que se encuentra bajo iluminación, como muestra la figura 2.9. Los fotones son absorbidos en cualquier parte de la unión con un coeficiente de absorción α. Cuando se absorbe un fotón, se genera un par electrón-hueco. Pero es solo en las zonas donde

hay presente un campo eléctrico donde los portadores de carga pueden ser desplazados en una dirección concreta. Una unión p-n solo puede mantener un campo eléctrico en la zona de vaciamiento, por lo que es en esta región en la que nos interesa generar fotoportadores [2].

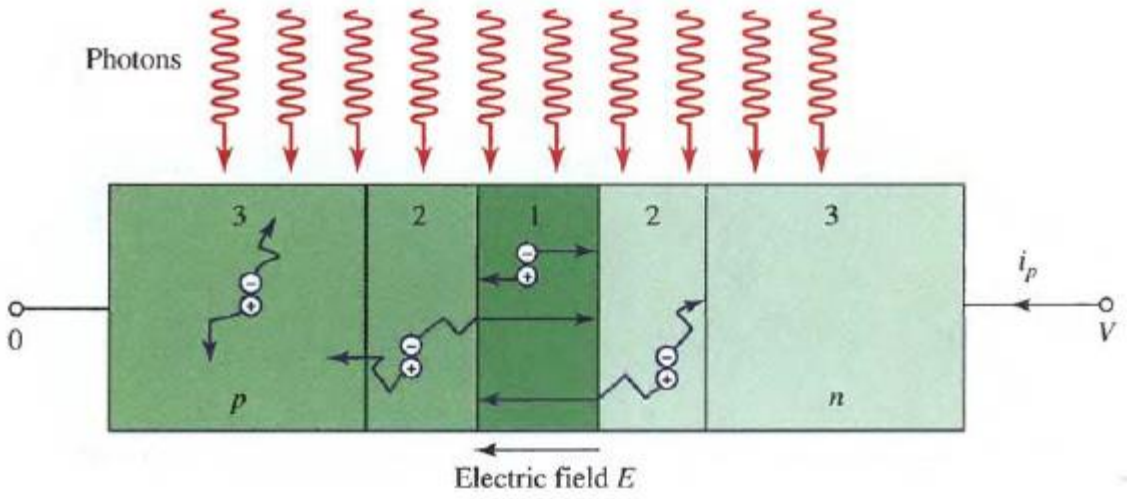


Figura 2.9: Fotones iluminando un fotodiodo detector p-n en inversa [2].

Sin embargo, hay tres posibles regiones donde pueden generarse los pares electrón-hueco:

- Electrones y huecos generados en la zona de vaciamiento (región 1 de la figura 2.9) rápidamente son atraídos en direcciones opuestas bajo la influencia del campo eléctrico, como resultado se genera una fotocorriente en inversa. Cada par de portadores contribuyen a la corriente exterior con una carga *e* (carga del electrón), ya que la recombinación no tiene lugar en la zona de vaciamiento.
- Electrones y huecos generados lejos de la zona de vaciamiento (región 3 de la figura 2.9) no pueden ser transportados debido a la ausencia de campo eléctrico. Los electrones y huecos generados en esta zona no contribuyen a la fotocorriente del dispositivo.
- Electrones y huecos generados fuera de la zona de vaciamiento, pero en sus inmediaciones (región 2 de la figura 2.9), tienen una cierta probabilidad de entrar en la zona de vaciamiento por difusión aleatoria. Un electrón generado en el lado p es atraído hacia el lado n, de forma que contribuye a la corriente exterior con una carga *e*. Lo mismo sucede con un hueco generado en el lado n.

2.4.2 Composición y respuesta espectral

Lo que define las propiedades de sensibilidad espectral de un fotodiodo es el material semiconductor que se emplea en la construcción. Los fotodiodos han sido fabricados utilizando muchos de los materiales semiconductores listados en la tabla 1.1 (del capítulo anterior), también utilizando compuestos binarios, ternarios y cuaternarios. En la tabla 2.2, mostrada a continuación, podemos ver algunos ejemplos de materiales

utilizados para la fabricación de fotodiodos, así como el rango espectral a que dan lugar [6]. El límite espectral superior tiene que ver con el GAP del material utilizado, al aumentar la longitud de onda de los fotones, la frecuencia se ve reducida, entonces la energía $h\upsilon$ del fotón pasa a ser inferior a la energía de la banda prohibida del material, por lo que no genera un par electrón-hueco. El límite espectral inferior, en cambio, es debido a que si la longitud de onda λ es muy pequeña, se da el fenómeno de absorción superficial, lo que da lugar a una rápida recombinación de los portadores generados de forma que no se genera fotocorriente [7].

Material	Rango (nm)
Silicio (Si)	190-1100
Germanio (Ge)	800-1700
Indio galio arsénico (InGaAs)	800-2600

Tabla 2.2: Tabla con algunos ejemplos de rango espectral de un fotodiodo dependiendo del material con el que ha sido fabricado [6].

También es posible la fabricación de fotodiodos para su uso en el campo de los infrarrojos medios (longitud de onda entre 2 y 20 μm), pero estos requieren refrigeración por nitrógeno líquido.

En la figura 2.10 podemos apreciar la respuesta espectral obtenida a partir de los materiales mencionados en la tabla 2.2. Démonos cuenta de que la elección del material de nuestro fotodiodo estará íntimamente relacionada con la región o regiones del espectro que nos interesan, por ejemplo:

- Elegiremos un fotodiodo basado en **silicio** si nuestras regiones de interés son la región visible y la infrarroja cercana (primera ventana de comunicaciones ópticas (850 – 950 nm)).
- Por el contrario, elegiremos un fotodiodo basado en **InGaAs** si nuestro interés se centra en las comunicaciones ópticas de segunda (1300 nm) y tercera ventana (1550 nm).

Recordemos que las regiones del espectro que a nosotros nos interesan son la visible (390 nm – 760 nm) y la infrarroja cercana (760 nm – 2000 nm), que son las regiones más importantes en el campo de la optoelectrónica. En función de esto, como ya comentamos previamente, deberemos escoger el tipo de fotodiodo que deseamos, es decir, qué material de fabricación nos interesa. Pues bien, parece sensato elegir un fotodiodo fabricado con silicio, ya que es el único (dentro de los mencionados anteriormente) capaz de trabajar en la región visible. Al elegir el silicio como material de fabricación, dado que la mayor sensibilidad se encuentra en torno a los 880 nm, podremos detectar especialmente bien los rayos infrarrojos que se encuentren en la primera ventana de comunicaciones ópticas. Por el contrario, los rayos procedentes de segunda y tercera ventana no seremos capaces de captarlos.

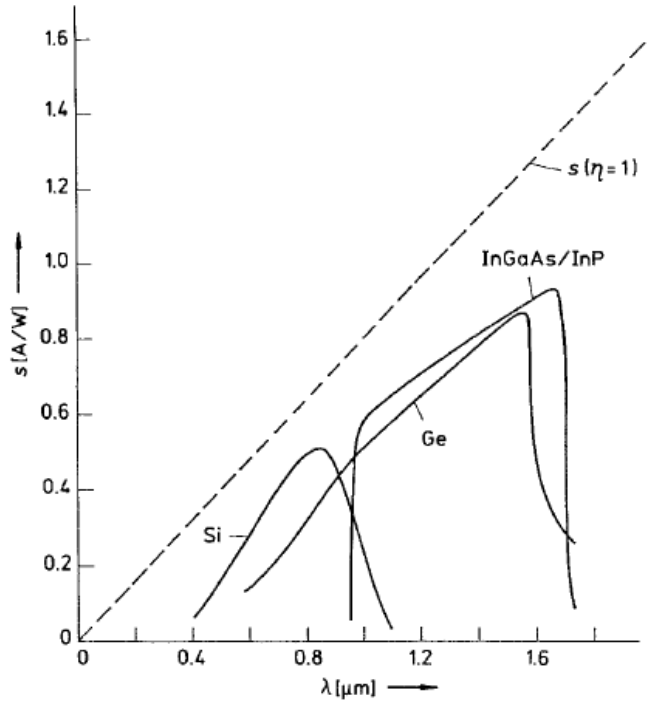


Figura 2.10: Respuesta espectral obtenida a partir de los materiales mencionados en la tabla 2.2.

2.4.3 Característica I-V

Como un dispositivo electrónico, el fotodiodo tiene una relación I-V que viene dada por:

$$I = I_s \left[exp\left(\frac{eV}{kT}\right) - 1 \right] - I_p \tag{2.7}$$

Esta es la relación I-V normal de una unión p-n (ecuación (2.1)) con una fotocorriente añadida $-I_p$ proporcional al flujo de fotones (y por lo tanto a la potencia óptica incidente), ver figura 2.11. La constante de proporcionalidad entre la fotocorriente I_p generada por el fotodiodo y la potencia óptica incidente, P_{opt}, se denomina responsividad (de la cual hablaremos en detalle en el apartado 2.4.4) [8]:

$$I_p = \Re P_{opt} \tag{2.8}$$

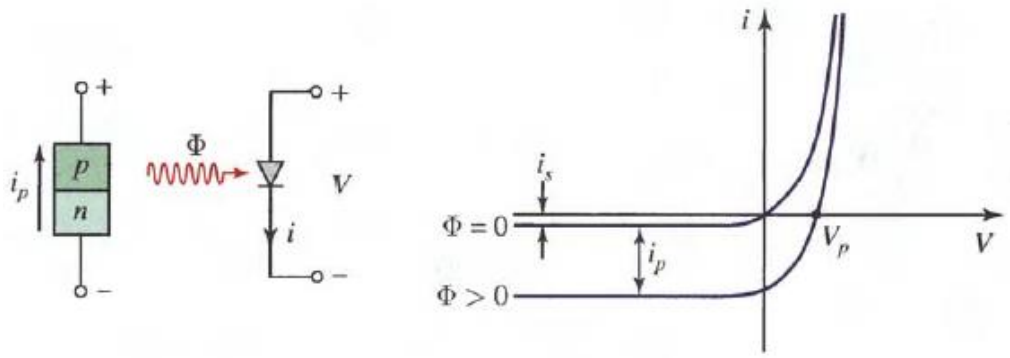


Figura 2.11: Fotodiodo genérico y su característica I-V [2].

El fotodiodo tiene tres modos de operación:

- Circuito abierto (figura 2.12 (a)): en este modo de operación, la luz genera pares electrón-hueco en la zona de vaciamiento. Los electrones liberados adicionalmente en el lado n de la capa se recombinan con los huecos del lado p, y viceversa. La red resultante es un incremento del campo eléctrico, el cual produce un fotovoltaje V_p a través del dispositivo que crece a medida que crece el flujo de fotones Φ. Este modo de operación es usado, por ejemplo, en paneles solares. La responsividad de un fotodiodo fotovoltaico es medida en *V/w* mejor que en *A/w*.
- Circuito cerrado ($V=0$) (figura 2.12 (b)): la corriente de cortocircuito es simplemente la fotocorriente i_p.
- Inversa (figura 2.13): un fotodiodo podría operar polarizado en inversa o modo de "fotoconducción". En la figura 2.13 podemos ver las condiciones de operación si añadimos una resistencia de carga en serie con el fotodiodo.

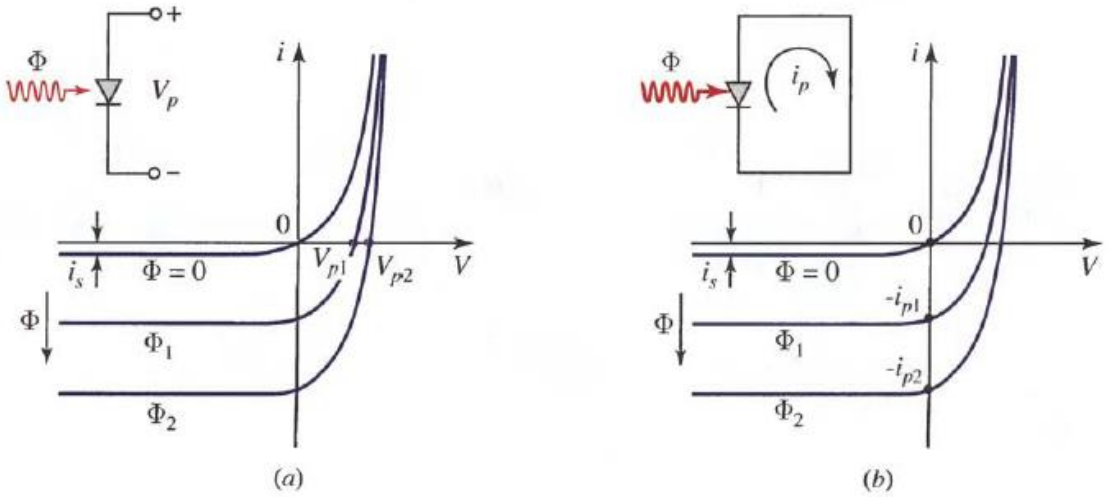


Figura 2.12: (a) Fotodiodo operando en circuito abierto. **(b)** Fotodiodo operando en cortocircuito [2].

Los fotodiodos operan normalmente polarizados fuertemente en inversa por las siguientes razones:

- Un fuerte voltaje de polarización inversa crea un gran campo eléctrico en la zona de la unión que incrementa la velocidad de desplazamiento de los portadores, reduciendo así el tiempo de tránsito.
- Un fuerte voltaje de polarización inversa incrementa el ancho de la zona de vaciamiento, reduciendo así la capacidad que se forma en la unión, reduciendo así el tiempo de carga (la constante de tiempo RC), lo que hace que el dispositivo tenga mejor respuesta temporal.
- El incremento de la zona de vaciamiento lleva a un área fotosensible mayor, haciendo más fácil la absorción de fotones.

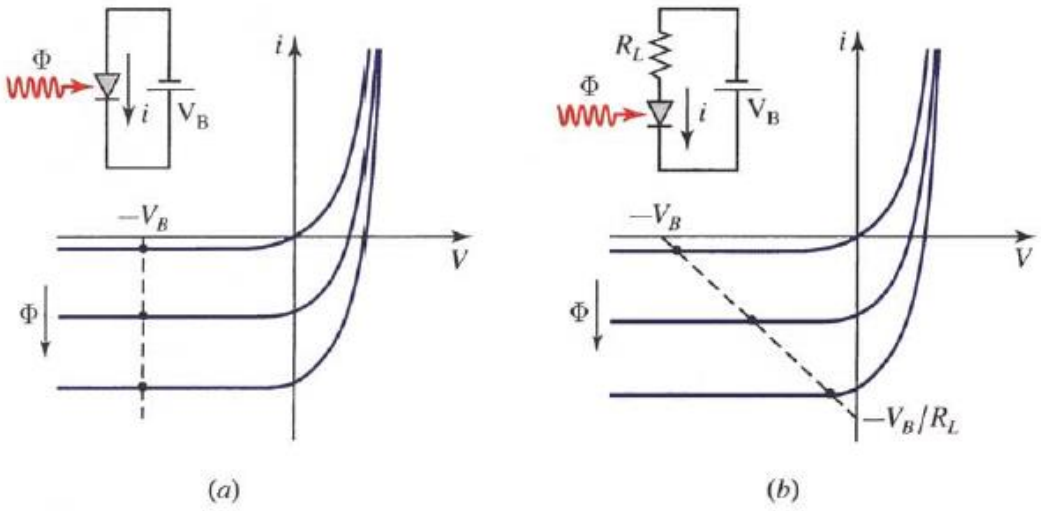


Figura 2.13: Fotodiodo polarizado en inversa (a) sin resistencia de carga, y (b) con resistencia de carga. El punto de operación tiende hacia la línea discontunua [2].

2.4.4 Eficiencia cuántica y responsividad

La eficiencia cuántica externa se define como la probabilidad de que un fotón incidente sobre el dispositivo genere un par electrón-hueco que contribuya a la corriente del fotodiodo [7 y 9]. Entonces, podemos escribir la eficiencia cuántica como:

$$\eta_{ext} = \frac{nº\ de\ pares\ e{-}h\ fotogenerados\ que\ contribuyen\ a\ la\ corriente}{nº\ de\ fotones\ incidentes} \tag{2.9}$$

No todos los fotones incidentes generan portadores que contribuyen a la fotocorriente, los efectos de reflexión en la superficie, transparencia del material a los fotones de energía inferior a la de la banda prohibida del mismo, la probabilidad de absorción cerca de la superficie del dispositivo y la rápida recombinación de portadores en este caso por la abundancia de defectos, hace que la eficiencia cuántica se reduzca. Si tenemos en cuenta estos factores, la eficiencia cuántica total vendrá dada por:

$$\eta_{ext} \approx (1 - R)[1 - \exp(-\alpha d)]\eta_i \tag{2.10}$$

Donde R es la reflexión que se produce en la superficie del dispositivo, α es coeficiente de absorción del material, d es la profundidad del fotodetector y η_i es la eficiencia cuántica interna.

Debemos notar que el coeficiente de absorción en la ecuación (2.10) depende de la longitud de onda, luego la eficiencia cuántica del dispositivo que utilicemos también dependerá de la longitud de onda. Por lo que nuevamente llegamos a la conclusión de que debemos elegir el material de fabricación de nuestro fotodiodo acorde al rango de longitudes de onda a detectar.

Pasemos ahora a hablar de la responsividad, como ya comentamos anteriormente, la fotocorriente proporcionada por el fotodiodo I_p es proporcional a la potencia óptica incidente y a la responsividad del fotodiodo, ver ecuación (2.8). Este parámetro, cuyas

unidades son *A/W*, representa la capacidad del fotodiodo de generar pares electrón-hueco por la incidencia de una señal óptica. Es un dato que suelen suministrar los fabricantes de dispositivos.

Si cada fotón incidente generase un par electrón-hueco, un flujo de fotones ϕ produciría el mismo flujo de electrones, con lo que tendríamos una fotocorriente igual a [9]:

$$I_p = e\phi \tag{2.11}$$

Por tanto, una potencia óptica incidente, $P_{opt} = \phi hv$, generaría una fotocorriente de la siguiente forma:

$$I_p = e\frac{P_{opt}}{hv} \tag{2.12}$$

Pero debemos tener en cuenta que no todos los fotones generan un par electrón-hueco, luego deberemos incluir la fracción de fotones útiles en la generación de esa corriente, y ese valor nos lo proporciona la eficiencia cuántica externa, por tanto:

$$I_p = e \cdot \eta_{ext} \cdot \frac{P_{opt}}{hv} = \Re P_{opt} \tag{2.13}$$

Llegados a este punto, podemos despejar la responsividad y obtenemos lo siguiente:

$$\Re(\lambda) = \frac{e \cdot \eta_{ext}(\lambda)}{hv}\left[\frac{A}{W}\right] = \eta_{ext} \cdot \frac{\lambda(\mu m)}{1.24} \tag{2.14}$$

$$\Re[dB] = 20\log(\Re[\tfrac{A}{W}]) \tag{2.15}$$

Según las diferentes relaciones que aparecen en la ecuación (2.14), la responsividad crece con λ hasta que se alcance el valor de la longitud de onda de corte λ_C, a partir de la cual la energía de los fotones es inferior a la del GAP y no son capaces de producir pares electrón-hueco, ya lo comentamos anteriormente. La responsividad del detector puede degradarse cuando la potencia óptica incidente es muy elevada. Se produce entonces la saturación del detector, es decir, se pierde la relación lineal (figura 2.14) entre la potencia óptica recibida y la corriente generada según la ecuación (2.13). En general, la responsividad dependerá también de factores como la temperatura y el ángulo de incidencia de la radiación sobre el detector (en la figura 2.16 se muestra la respuesta angular del fotodiodo utilizado en nuestro sistema).

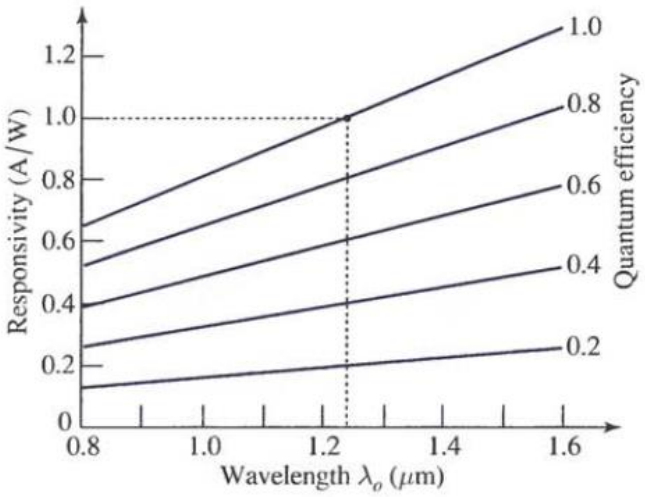


Figura 2.14: Responsividad y eficiencia cuántica externa en función de la longitud de onda [2].

2.4.5 En nuestro sistema

En nuestro sistema tenemos situado el fotodiodo en la ranura de salida del monocromador. La elección del fotodiodo adecuado a nuestro sistema se basa en las regiones del espectro que nos interesan (visible e infrarroja cercana). Como ya comentamos anteriormente, el material más adecuado para nuestros propósitos es el silicio, cuya zona sensible va desde los 400 nm hasta los 1100 nm, lo cual cubre todo el espectro visible y parte del infrarrojo (primera ventana de comunicaciones ópticas (850 – 950 nm)).

Finalmente, nos decidimos a utilizar el fotodiodo basado en silicio de montaje superficial TEMD5010X01, que es de alta sensibilidad y velocidad, incluyendo un área sensible de 7.5 mm^2 [10]. Elegimos este fotodiodo porque es sensible al rango del espectro que nos interesa (debido a su fabricación mediante silicio), en concreto el rango espectral del fotodiodo TEMD5010X01 va desde los 430 nm hasta los 1100 nm, y su sensibilidad máxima se encuentra en torno a los 900 nm (figura 2.15). Además tiene una buena respuesta angular (mostrada en la figura 2.16), aunque en nuestro caso la luz deberá incidir de forma perpendicular. Nuestro fotodiodo proporciona una fotocorriente cuando es iluminado, esta corriente varía entre unos pocos µA hasta 100 µA en función de la iluminación y tiene una corriente de oscuridad típicamente de 2 nA (aunque puede llegar hasta los 30 nA) [10].

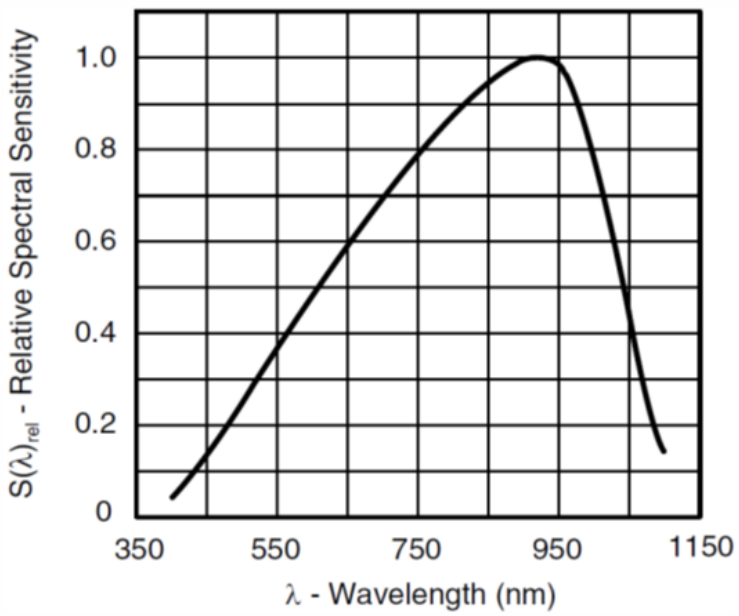


Figura 2.15: Sensibilidad espectral relativa (normalizada) en función de la longitud de onda del fotodiodo TEMD5010X01 [10]. Podemos observar que el rango espectral va desde los 430 nm hasta los 1100 nm y que su sensibilidad máxima se encuentra en torno a los 900 nm.

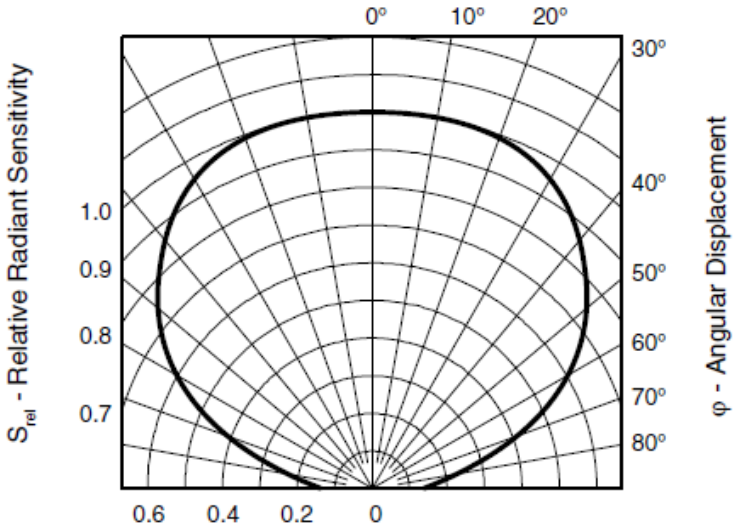


Figura 2.16: Respuesta angular del fotodiodo TEMD5010X01 incluido en nuestro sistema [10].

2.4.6 Acondicionamiento del fotodiodo

Como ya comentamos anteriormente, el fotodiodo proporciona una fotocorriente de unos pocos µA cuando es iluminado. Entonces, será necesario convertir esa corriente en voltaje, para que el ADC del microcontrolador pueda realizar su conversión. Además, al ser una corriente tan pequeña, necesitaremos amplificarla de manera que el voltaje obtenido cubra todo el rango de entrada al ADC (0 – 3.3V, como veremos más adelante) y poder aprovechar así toda su resolución.

Durante las prácticas en empresa, se diseñó y fabricó un circuito con 2 etapas. La primera etapa se encargaría de la conversión de corriente a tensión y la segunda de

incluir un circuito amplificador para la señal (ya convertida en tensión). La segunda etapa no solo amplificaba la señal, sino que además incluimos una forma de seleccionar la ganancia deseada dentro de ciertas ganancias preestablecidas. De esta manera podremos amoldarnos a la intensidad de la luz procedente del LED.

La primera etapa, la cual puede observarse en la figura 2.17, consta de un amplificador MPC6282, dos condensadores y dos resistencias. Los condensadores C1 y C2 se incluyen para evitar oscilaciones de la señal de salida.

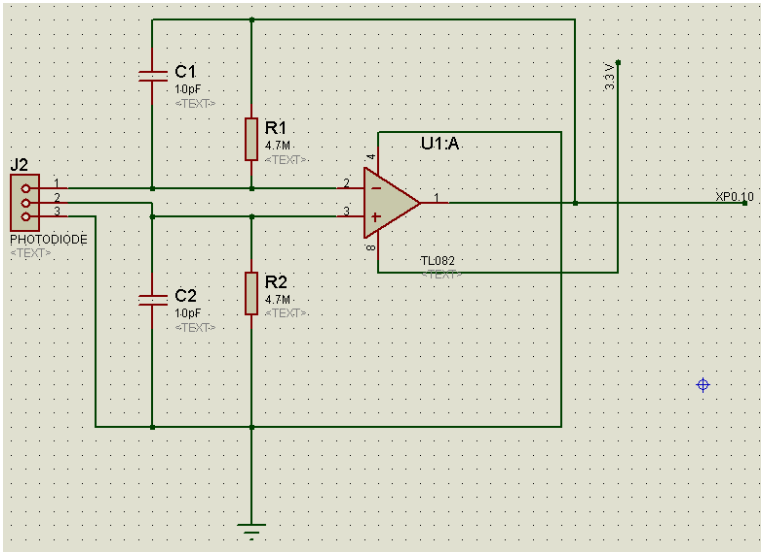


Figura 2.17: Primera etapa del acondicionamiento del fotodiodo, circuito conversor corriente-tensión.

La segunda etapa de ganancia (figura 2.18) fue construida con un amplificador operacional en configuración no inversora (también el MPC6282), y un multiplexor triple de 2 a 1 CD4053B que va a permitir seleccionar la ganancia del lazo de realimentación. Como se puede apreciar, los multiplexores se han conectado en árbol para obtener un multiplexor de 4 a 1 y así variar la ganancia del bucle de realimentación entre 4 opciones posibles.

Se eligieron los siguientes factores de ganancia para esta etapa amplificadora: x1, x4, x16 y x64. Para conseguir estas ganancias, es necesario que las resistencias del circuito de la figura 2.18 tengan los siguientes valores [4]:

$$R_T = R_1 + R_2 + R_3 + R_4 = 10K\Omega\ ;\ G_0 = 1\ ;\ G_1 = \frac{R_1 + R_2 + R_3 + R_4}{R_1 + R_2 + R_3}$$

$$G_2 = \frac{R_1 + R_2 + R_3 + R_4}{R_1 + R_2}\ ;\ G_3 = \frac{R_1 + R_2 + R_3 + R_4}{R_1}$$

$$R_1 = \frac{R_T}{G_3} = 156\ \Omega\ ;\ R_2 = \frac{R_T}{G_2} - R_1 = 469\ \Omega$$

$$R_3 = \frac{R_T}{G_1} - R_1 - R_2 = 1875\ \Omega\ ;\ R_4 = R_T - R_1 - R_2 - R_3 = 7500\ \Omega$$

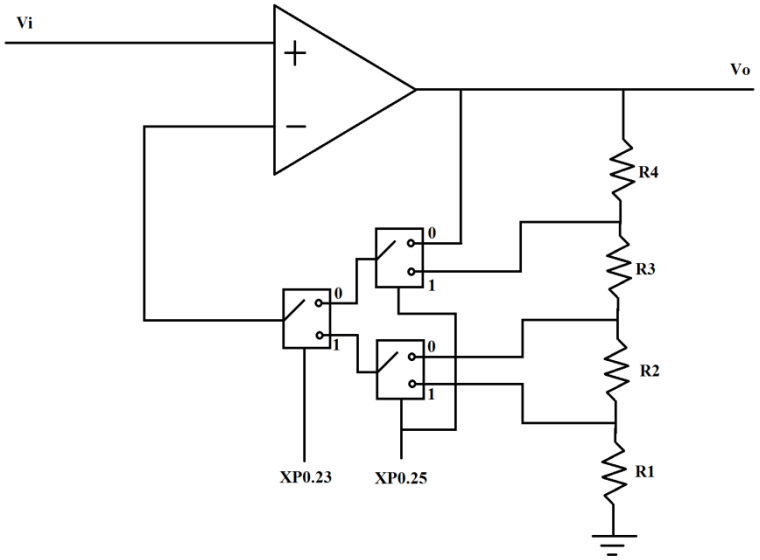


Figura 2.18: Segunda etapa del acondicionamiento de la señal del fotodiodo, circuito selector de ganancia. `XP0.23` y `XP0.25` son los pines del microcontrolador que controlan los multiplexores seleccionando así la ganancia deseada en cada momento. Los valores para R1, R2, R3 y R4 fueron calculados para obtener unas ganancias preestablecidas [4].

No disponemos de resistencias cuyos valores sean exactamente los calculados teóricamente, por lo que deberemos conformarnos con valores similares. Debido a esto, no obtendremos exactamente los valores de ganancia comentados anteriormente sino que serán unos valores aproximados. En la tabla 2.3 mostramos las resistencias utilizadas y las ganancias finalmente obtenidas.

Resistencias utilizadas (Ω)	Ganancias obtenidas
R1 = 150	G0 = 1
R2 = 470	G1 = 4.10
R3 = 1800	G2 = 16
R4 = 15000‖15000 = 7500	G3 = 66.13

Tabla 2.3: Contiene las resistencias utilizadas para el circuito selector de ganancia (segunda etapa del acondicionamiento de la señal del fotodiodo) y las ganancias finalmente obtenidas.

Como podemos observar en la tabla, hemos podido utilizar resistencias de valores muy similares a los calculados. En el caso de R4, al no tener ninguna resistencia de un valor aproximado, hemos utilizado 2 resistencias del doble del valor en paralelo, con lo cual obtenemos el valor de resistencia deseado. Debido a la utilización de resistencias muy similares a las calculadas anteriormente, los factores de ganancia obtenidos finalmente son del orden de los deseados.

Resumiendo, el usuario será capaz de seleccionar la ganancia adecuada a cada situación, dependiendo de la intensidad de la corriente procedente del fotodiodo, para aprovechar la resolución que le ofrece el convertidor A/D.

2.5 PC

Necesitaremos una forma sencilla para que el usuario pueda interaccionar con el sistema y pueda controlar todos los parámetros en cuanto a medidas se refiere, es decir, longitudes de onda inicial y final entre las cuales recoger la intensidad con la que emite el LED, la ganancia para esas medidas, donde almacenar y representar el espectro de emisión del LED, etc.

El candidato perfecto para realizar esta tarea es, sin lugar a dudas, un PC. Entonces, necesitaremos crear una interfaz mediante la cual el usuario sea capaz de controlar todos estos aspectos del sistema.

Como interfaz hemos utilizado la consola de línea de comandos del sistema operativo Linux Ubuntu 12.04 LTS, en la cual lanzamos un ejecutable proveniente de un programa escrito en el lenguaje de programación C/C++, del cual hablaremos en detalle en el capítulo 3. En la figura 2.19 podemos observar la consola de Linux una vez ha sido lanzado el ejecutable del programa.

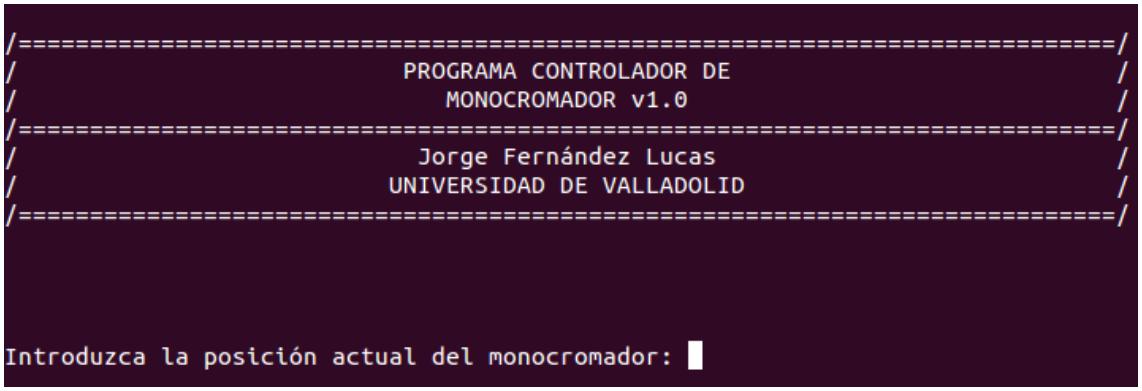


Figura 2.19: Aspecto de la consola de línea de comandos de Linux al lanzar el fichero ejecutable del programa.

Para escribir el programa hemos utilizado el entorno de desarrollo integrado (IDE), de código abierto y multiplataforma, Eclipse. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C/C++ o aplicaciones Java. El IDE Eclipse posee herramientas y funciones muy útiles para nuestro trabajo, recogidas además en una interfaz (figura 2.20) que lo hace muy cómodo de usar. El compilador utilizado por Eclipse es el compilador gcc/g++, el cual se encarga de la compilación del código C/C++ y la obtención de los ficheros ejecutables.

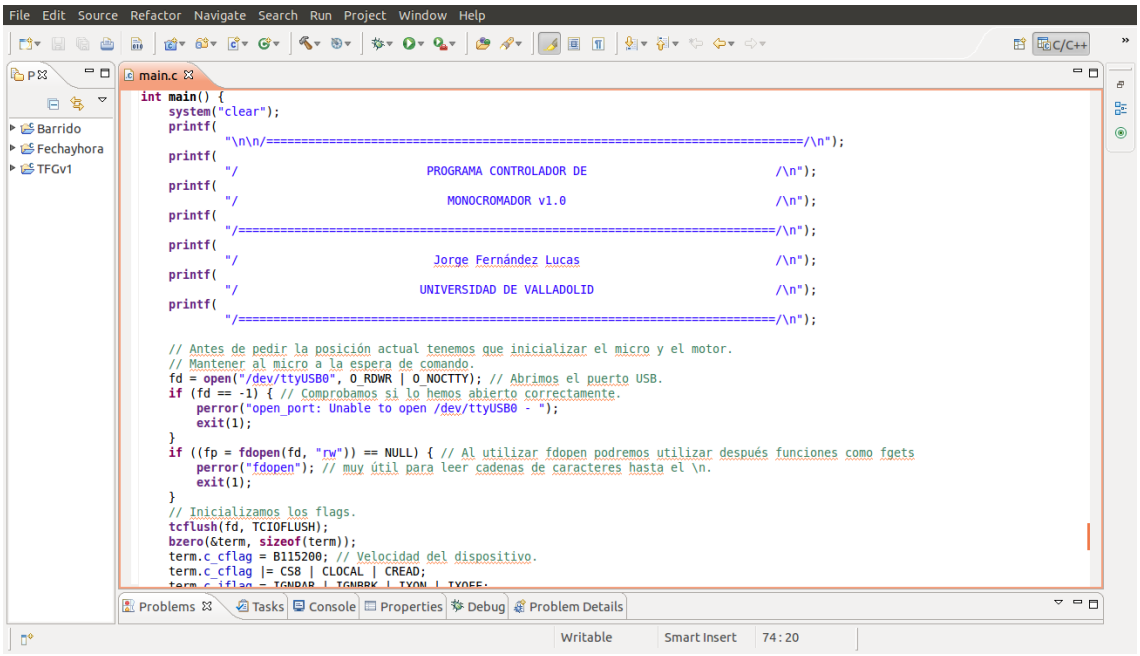


Figura 2.20: Interfaz del entorno de desarrollo integrado Eclipse, en la figura podemos apreciar una parte del programa que desarrollamos gracias a esta herramienta.

Además de la interfaz de usuario, necesitaremos el PC para realizar el programa que introduciremos en el microcontrolador, explicado en mayor detalle en el capítulo 4. En este caso, para escribir el programa no hemos utilizado la herramienta Eclipse, sino que nos ha bastado con utilizar un editor de textos corriente, en nuestro caso hemos utilizado el editor gedit, por defecto incluido en el sistema operativo Linux Ubuntu 12.04 LTS. Para ejecutar y grabar el programa en el microcontrolador hemos utilizado una serie de ficheros proporcionados por Jesús Arias Álvarez, que pueden encontrarse en su página web (http://www.ele.uva.es/~jesus/hardware_empotrado/, en concreto hemos utilizado las carpetas `bootloader` y `lpc2103simple`). Gracias a estos ficheros hemos podido ejecutar una serie de comandos de los cuales, los más útiles han resultado los mostrados a continuación:

- `make`: Gracias a esta orden pudimos ejecutar el programa en el microcontrolador y comprobar su funcionamiento, sin grabarlo y, por lo tanto, sin perjudicar la memoria flash que puede ser reescrita un número finito de veces.
- `makeburn`: Esta orden, en cambio, grababa el programa en la memoria flash del microcontrolador, por lo que la utilizamos una vez que el funcionamiento del programa era el correcto y ejecutaba las órdenes deseadas.

También necesitaremos que el PC se comunique con el microcontrolador, que será quien haga de intermediario entre el PC y el resto de elementos. El PC enviará una orden al microcontrolador, este último deberá interpretarla, realizarla y avisar al PC de que la orden ha sido realizada correctamente (o de forma errónea), entonces el PC podrá continuar con la siguiente orden. Por otro lado, el microcontrolador, cuando tenga que realizar una medida, tendrá que responder al PC con el resultado de la medida, esta

respuesta servirá como señal de orden realizada con éxito. Para la comunicación entre PC y microcontrolador hemos utilizado el código ASCII, es decir, el PC envía una cadena de caracteres en código ASCII por el puerto USB, y el microcontrolador interpreta esa cadena y ejecuta la orden correspondiente (hablaremos de ellas en el capítulo 4).

2.6 MICROCONTROLADOR

Un microcontrolador es un circuito integrado, en cuyo interior posee toda la arquitectura de un PC, esto es CPU, memorias RAM, EEPROM, y circuitos de entrada y salida.

Un microcontrolador de fábrica, no realiza ninguna tarea, este debe ser programado para que realice desde un simple parpadeo de un led hasta un sofisticado control de un robot. Un microcontrolador es capaz de realizar la tarea de muchos circuitos lógicos como compuertas AND, OR, NOT, NAND, conversores A/D, D/A, temporizadores, decodificadores, etc., simplificando todo el diseño a una placa de reducido tamaño y pocos elementos [11].

En nuestro sistema, el microcontrolador se encargará del control del motor paso a paso y, por lo tanto, de controlar la posición de la rejilla y así la longitud de onda que se deja pasar a través del monocromador. Además, también se encarga de encender y apagar el LED, así como de interpretar la señal analógica procedente del fotodiodo y seleccionar una ganancia para la misma [4].

2.6.1 El microcontrolador LPC2103

En nuestro caso, contamos con el microcontrolador LPC2103, que es un microcontrolador basado en un núcleo ARM7 que posee un cristal de cuarzo que permite una frecuencia máxima de funcionamiento de 14.7456 MHz; viene montado sobre una *Educational Board* (ver figura 2.21) utilizada ampliamente en entornos educativos por su fácil curva de aprendizaje y que a su vez viene con varios elementos, mencionamos algunos de ellos aunque no los vamos a utilizar:

- Botón de *reset*.
- Botón `P0.14` activo en baja.
- LED de 7 segmentos.
- Entrada analógica vía potenciómetro.
- Otros.

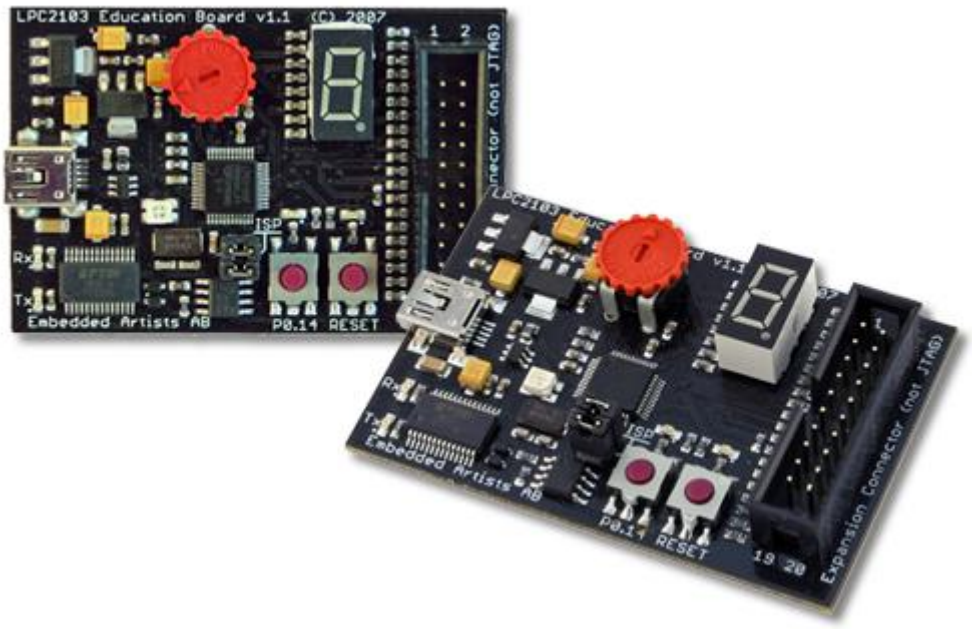


Figura 2.21: Imagen del microcontrolador LPC2103 usado en nuestro sistema. Observamos que aparece montado sobre una *Educational Board* en la que además se pueden apreciar los distintos elementos comentados anteriormente.

2.6.1.1 Principales características

De todas las características del microcontrolador LPC2103, las más interesantes para nosotros son aquellas de las que nos hemos servido para llevar a cabo nuestros propósitos. Procedemos a explicarlas a continuación [12]:

- Convertidor analógico digital (ADC) de 10 bits provisto con 8 canales de entrada, nosotros necesitaremos uno de estos canales de entrada para digitalizar la señal analógica proveniente del fotodiodo. El convertidor A/D es capaz de conseguir un tiempo de conversión mínimo de 2.44 µs para cada canal y, además, posee registros para almacenar el resultado dedicados a cada canal. Su rango de medidas se encuentra entre 0 V y 3.3 V, por lo que procuraremos que la señal procedente de nuestro fotodiodo aproveche al máximo dicho rango mediante el selector de ganancia.
- Cuatro temporizadores, dos de 32 bits y dos de 16 bits. Nosotros utilizaremos el Timer1, que es un temporizador de 32 bits, con 4 canales de captura (`CAP1[3..0]`), 4 registros de cuenta (`MR[3..0]`) y 4 salidas externas (`MAT1[3..0]`). Utilizaremos uno de los registros de cuenta (`MR0`) para que cuando el temporizador se iguale a ese valor, se reinicie y además se produzca un flanco en una de las salidas externas (`MAT1.0`), que la utilizaremos para controlar los tiempos de conversión del convertidor A/D.
- Múltiples interfaces en serie, incluyendo dos UARTs (*Universal Asynchronous Receiver/Transmitter*). Necesitaremos uno de estos puertos para realizar nuestra comunicación con el PC, en nuestro caso utilizaremos el UART0, se trata de un receptor/transmisor de 16 *bytes* basados en la disciplina de cola FIFO (*First In*

First Out). En el capítulo 4 veremos cómo afecta la disciplina FIFO a la hora de leer las órdenes proporcionadas por el PC.

- 32 pines de entrada/salida de propósito general (GPIO/FGPIO) tolerantes a 5 V, los utilizaremos para controlar cada dispositivo presente en el sistema (motor y lógica de finales de carrera del monocromador, LED, fotodiodo y selector de ganancia). Pueden ser configuradas como FGPIO (*Fast General Porpouse I/O*) o GPIO, en nuestro caso utilizaremos las FGPIO por su mayor rapidez y sus características más avanzadas. Aunque el LPC2103 nos ofrece 32 GPIO (P0.0-P0.31), no todas ellas son accesibles ya que utilizamos este microcontrolador montado sobre una *Educational Board*. Podemos observar el conector de la *Educational Board* en la figura 2.22, consta de 20 pines: del 1 al 17 son las GPIO/FGPIO accesibles, el pin 18 no se utiliza, el 19 proporciona una salida de 3.3 V y, por último, el 20 es una toma de tierra.

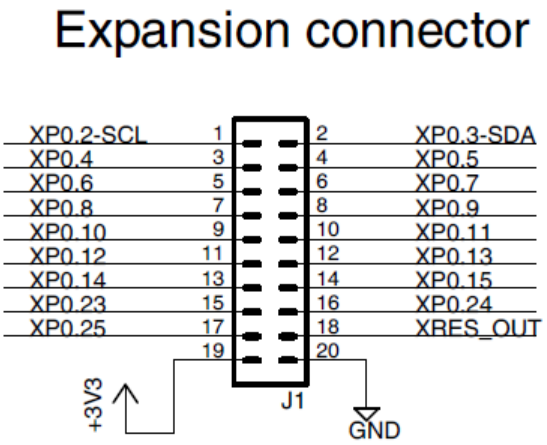


Figura 2.22: Conector de la *Educational Board* sobre la que viene montado nuestro microcontrolador.

2.7 PCB

Ante la necesidad de interconectar todos los dispositivos descritos anteriormente, llegamos a la conclusión de que la forma más cómoda sería realizar las conexiones mediante una PCB. El diseño de esta placa se realizó durante las prácticas en empresa, puede observarse el resultado en la figura 2.23, además, en el apéndice A se explica dónde pueden encontrarse los diseños realizados para la placa mediante las herramientas ARES e ISIS. A continuación mostramos las especificaciones que cumple la placa, además de interconectar todos los dispositivos:

- Acoplar el microcontrolador, conector DIL20.
- Circuito de control sobre el LED, que básicamente es encenderlo y apagarlo.
- Circuitos de control del motor paso a paso y de lectura de finales de carrera.
- Acoplar fuente de alimentación externa de 12V.
- Circuito regulador de tensión, para obtener 5 voltios, necesarios para alimentar la lógica de los finales de carrera, a partir de los 12V con los que alimentamos al motor. El microcontrolador también necesita 5 voltios de alimentación pero los obtiene del PC mediante una conexión USB.

- Circuito convertidor corriente-tensión con amplificación seleccionable para la señal procedente del fotodiodo.
- Circuito selector de ganancia para el fotodiodo con distintas ganancias seleccionables por medio del microcontrolador.

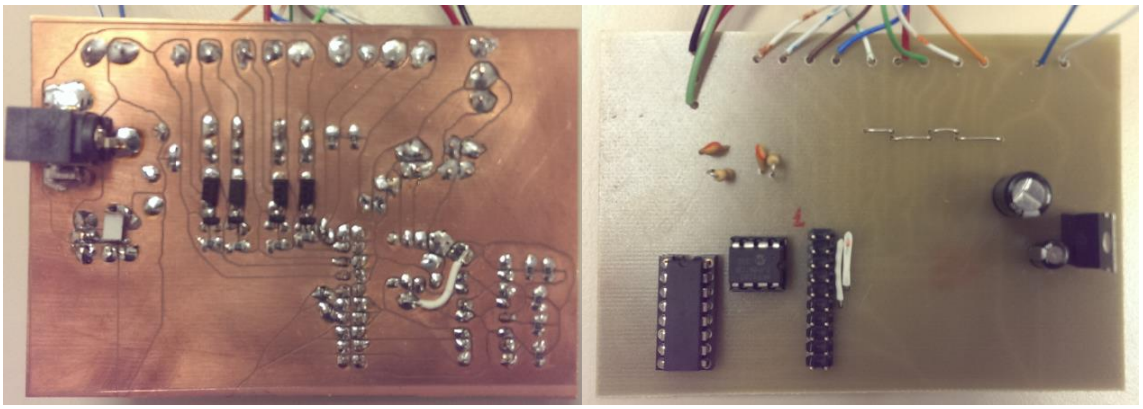


Figura 2.23: Placa PCB fabricada utilizando la fresadora disponible en la ETSIT. A la izquierda se muestra la cara *Top Copper* con los componentes superficiales y un cable de puenteo usado para llevar la alimentación de 3.3 V al multiplexor. A la derecha se muestra la cara *Bottom Copper* con los componentes de inserción y los cables de puenteo. El microcontrolador se encuentra desconectado del conector DIL20 para poder apreciar todos los elementos de la placa correctamente [4].

CAPÍTULO 3. DESARROLLO INTERFAZ DEL PC

3.1 INTRODUCCIÓN

En este capítulo comentaremos la aplicación desarrollada desde el punto de vista del PC. Primero veremos lo que el usuario es capaz de hacer, mostrando para ello diversos ejemplos y los diferentes menús de la interfaz, y posteriormente pasaremos a explicar cómo llevamos a cabo el desarrollo del código necesario para conseguir los diferentes objetivos de la aplicación.

Ante la necesidad de que el usuario fuese capaz de controlar el sistema de una forma amigable, se desarrolló una interfaz de usuario. Como ya comentamos en el capítulo 2, como interfaz utilizamos la consola de línea de comandos del sistema operativo Linux Ubuntu 12.04 LTS, en la cual lanzamos un ejecutable proveniente de un programa escrito en el lenguaje de programación C/C++. El desarrollo de dicho programa lo llevamos a cabo mediante el entorno de desarrollo integrado (IDE), de código abierto y multiplataforma, Eclipse, del cual ya hablamos previamente.

Como ya hemos dicho, la interfaz que utilizamos es la consola de línea de comandos de Linux, pues bien, nos pareció que la forma más sencilla e intuitiva de manejar esta interfaz para el usuario sería por medio de menús (por los cuales el usuario podrá ir "navegando"). Para aclararlo mostramos un ejemplo a continuación, en la figura 3.1, se trata del menú principal de la aplicación.

En el siguiente apartado mostraremos el resto de menús a los que el usuario puede acceder y, además, explicaremos las distintas acciones que este puede realizar. Al explicar dichas tareas, observaremos que están íntimamente relacionadas con las diferentes funciones de que consta el código, en el apéndice A se explica dónde puede encontrarse dicho código. Asimismo explicaremos los aspectos más importantes de estas funciones más adelante.

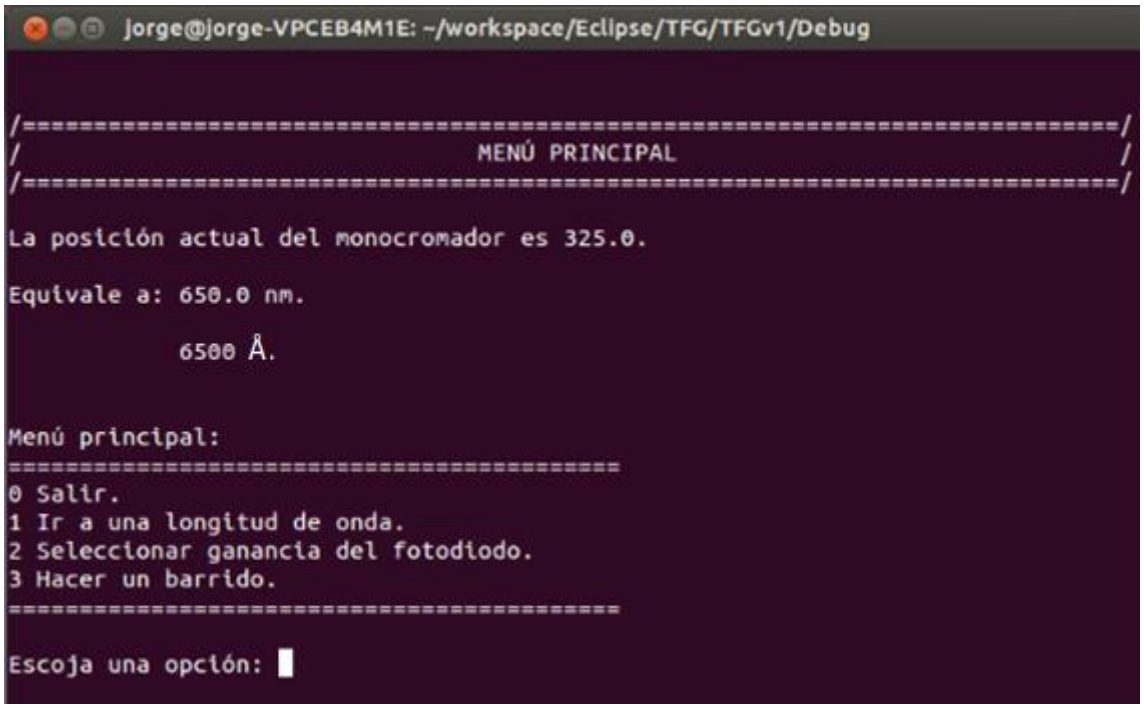


Figura 3.1: Menú principal de la aplicación desarrollada, en este entorno deberá desenvolverse el usuario.

3.2 TAREAS REALIZABLES POR EL USUARIO

La idea es que el usuario sea capaz de realizar las siguientes tareas con la ayuda del sistema:

- Salir de la aplicación.
- Ir a una longitud de onda.
- Seleccionar una ganancia para la señal del fotodiodo.
- Hacer un barrido en longitudes de onda.

Para que el usuario pueda seleccionar una de las tareas mencionadas, se le proporciona el menú principal (mostrado en la figura 3.1). Veamos como el usuario interacciona con la interfaz para llevar a cabo cada una de las acciones disponibles.

3.2.1 Salir

Si el usuario selecciona la opción “Salir de la aplicación” en el menú principal, el programa terminará, devolviéndole así el control a la consola de comandos. En la siguiente figura podemos apreciar el aspecto de la interfaz al seleccionar esta opción.

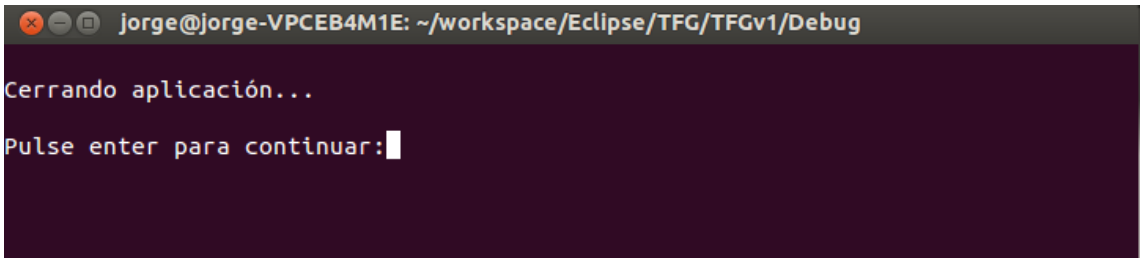


Figura 3.2: Muestra el aspecto de la consola de comandos cuando seleccionamos la opción salir de la aplicación.

3.2.2 Ir a una longitud de onda

El usuario seleccionará esta opción si desea obtener la medida de la intensidad del LED en una longitud de onda en concreto. Notar que el usuario no podrá seleccionar la ganancia aplicada a dicha medida, si el usuario desea realizar esta medida con alguna ganancia en particular deberá preseleccionarla con ayuda de la opción "Seleccionar ganancia del fotodiodo" (explicada en el apartado 3.2.3). Si el usuario no hubiese seleccionado ninguna ganancia, por defecto no se amplifica la señal, es decir, la ganancia seleccionada es x1.

Al seleccionar la opción "Ir a una longitud de onda" en el menú principal, el usuario pasará a ver el siguiente menú (figura3.3), en el cuál se le pide que introduzca la posición en Å (ångströms) a la que desea desplazar el monocromador.

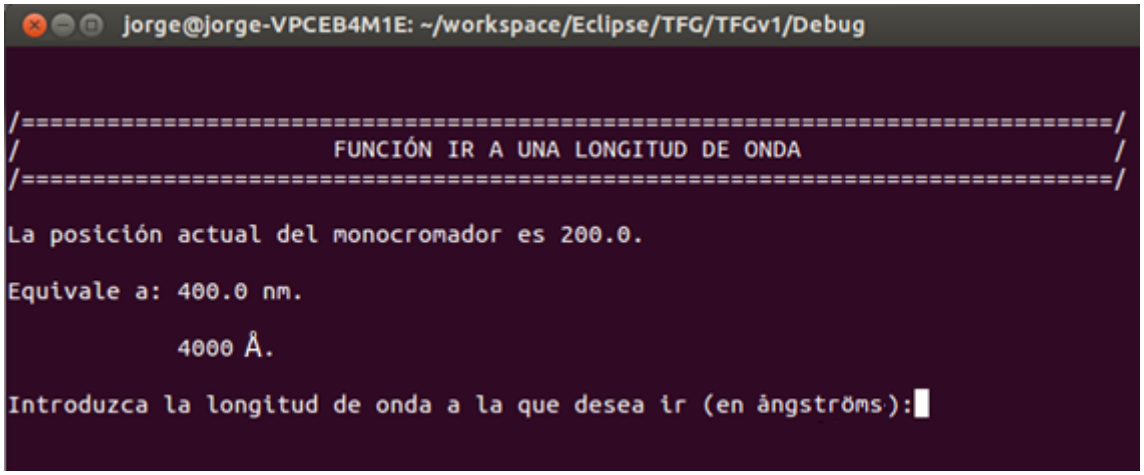


Figura 3.3: Aspecto de la consola de comandos cuando el usuario selecciona la opción "Ir a una longitud de onda". El usuario deberá introducir la longitud de onda en **Å** a la que desea ir.

Una vez seleccionada la longitud de onda en cuestión, el PC le enviará la orden correspondiente al microcontrolador para que este coloque el monocromador en la posición deseada. Al encontrarse el monocromador en la posición deseada, el PC deberá indicarle al microcontrolador que encienda el LED, que realice la medida, es decir, interprete la señal recibida del fotodiodo, y, por último, deberá indicarle que apague el LED. Cuando el PC reciba la medida realizada, este se la mostrará por pantalla al usuario, podemos ver el resultado de esta acción en la figura 3.4.

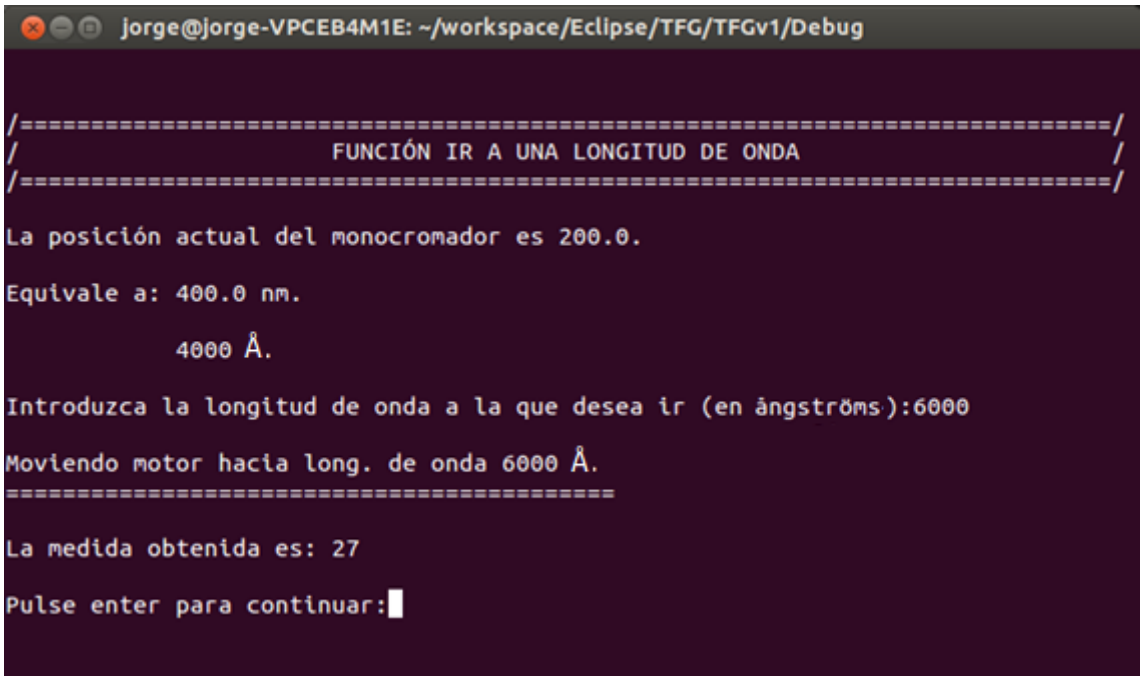


Figura 3.4: Aspecto de la consola de comandos una vez que la acción "Ir a una longitud de onda" ha sido realizada, podemos observar el resultado de la medida realizada en la longitud de onda en concreto.

Cuando se finaliza la acción, la aplicación muestra de nuevo el menú principal esperando a que el usuario indique la siguiente tarea que desea realizar.

3.2.3 Seleccionar una ganancia para la señal del fotodiodo

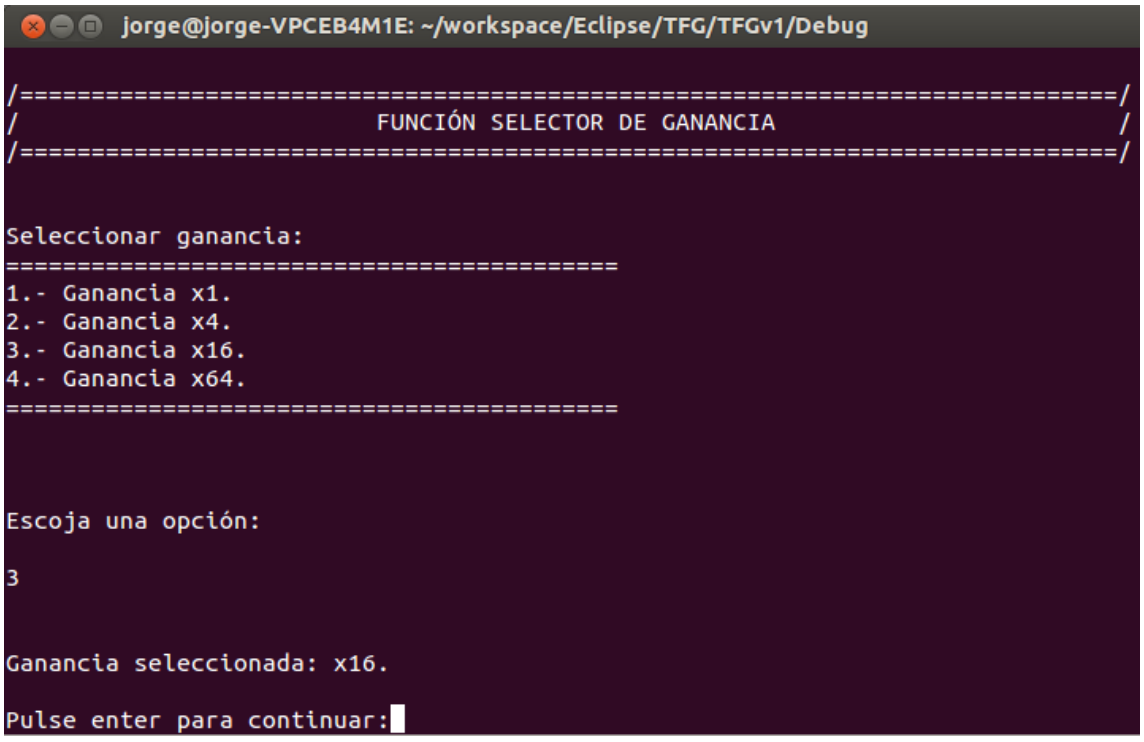


Figura 3.5: Aspecto de la interfaz cuando el usuario selecciona la opción "Seleccionar ganancia del fotodiodo". Después de escoger la ganancia deseada, la aplicación confirma que, efectivamente, dicha ganancia ha sido seleccionada correctamente.

Si el usuario desea seleccionar una ganancia para la señal procedente del fotodiodo, deberá decantarse por esta opción. Al seleccionar dicha opción, el usuario accederá al menú mostrado en la figura 3.5, donde se le pide que seleccione una de las cuatro ganancias disponibles. Una vez que el usuario ha elegido su ganancia, el PC deberá confirmar por pantalla la ganancia seleccionada y, posteriormente, se mostrará de nuevo el menú principal a la espera de la siguiente acción a realizar.

3.2.4 Hacer un barrido

Esta es la tarea más potente y completa que el usuario puede realizar, se trata de un barrido en frecuencia, es decir, realizamos la representación de la intensidad emitida por el LED frente a la longitud de onda, lo que es lo mismo que la obtención del espectro de emisión del LED. Esta representación se realizará a partir de un fichero de datos que se guardará en el proceso, más adelante comentaremos qué datos contiene este fichero. Pasemos a explicar los pasos que ha de seguir el usuario para realizar un barrido de forma satisfactoria, para ello, nos apoyaremos en un barrido que hemos realizado a modo de ejemplo.

Primeramente, la aplicación le pedirá al usuario el rango del barrido, es decir, las posiciones inicial y final del mismo. En la figura 3.6 podemos observar que, en el barrido de ejemplo, el usuario desea hacer un barrido desde los 5600 hasta los 6500 Å.

Figura 3.6: La aplicación le pide al usuario que introduzca las posiciones inicial y final del barrido que desea realizar. Además, la aplicación nos indica la posición en la que se encuentra actualmente el monocromador.

Una vez introducidas las posiciones inicial y final del barrido, el siguiente paso será indicarle a la aplicación la resolución con la que queremos realizar dicho barrido. La aplicación nos dará a elegir entre 4 resoluciones disponibles, el usuario deberá elegir la que más le convenga en cada momento. Pongamos un ejemplo para ver por qué nos interesa tener varias resoluciones posibles.

El usuario podría desear caracterizar un LED del cuál desconoce el rango de emisión. Entonces, lo recomendable sería realizar primero un barrido de gran rango y de forma rápida para ver en qué longitudes de ondas está emitiendo, para ello al usuario le interesará usar poca resolución, es decir, un incremento entre medidas grande, 20 Å por ejemplo. Una vez que el usuario sabe el rango de emisión del LED, podría interesarle centrarse en ese rango y realizar un barrido con mayor resolución, es decir, un incremento entre medidas pequeño. Este incremento podría ser de 5 Å, como en el barrido del ejemplo (ver figura 3.7), o incluso de 2.5 Å.

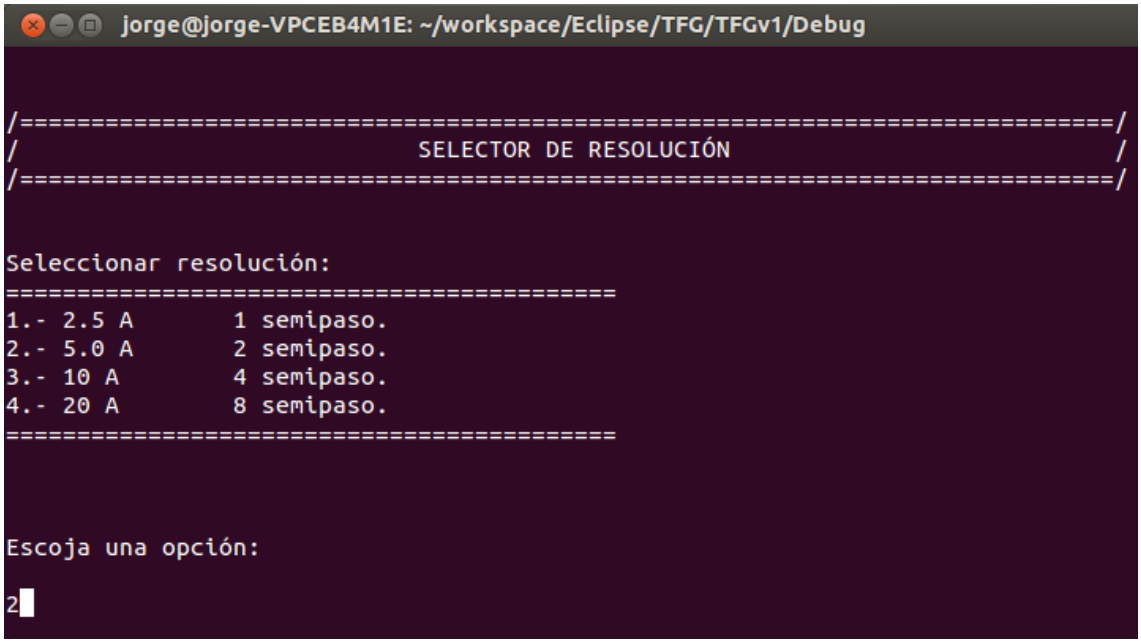


Figura 3.7: La aplicación le indica al usuario las diferentes resoluciones seleccionables y le pide a este que escoja una. Como muestra la aplicación, cada incremento entre medidas está directamente relacionado con un número entero de semi-pasos del motor del monocromador. Estos números de semi-pasos van desde un octavo de vuelta del motor (1 semi-paso) a una vuelta completa (8 semi-pasos).

A continuación, el usuario deberá seleccionar la ganancia con la que desea realizar el barrido, para ello la aplicación le muestra el mismo menú que en la opción "Seleccionar ganancia del fotodiodo". Hemos incluido este paso para que el usuario pueda escoger todos los aspectos del barrido fácilmente, sin tener que volver a la opción "Seleccionar ganancia del fotodiodo" del menú principal para escoger la ganancia deseada. En la figura 3.8 puede apreciarse la ganancia seleccionada para el barrido de ejemplo.

Por último, el usuario tendrá que elegir el nombre con el cual desea guardar el fichero de datos. La aplicación le ofrece al usuario un nombre por defecto que contiene la fecha, las posiciones inicial y final, la resolución y la ganancia del barrido. Si el usuario no desea guardar el fichero con ese nombre puede escribir el nombre que desee, en el barrido de ejemplo, el fichero se almacenará con el nombre "Prueba", ver figura 3.9.

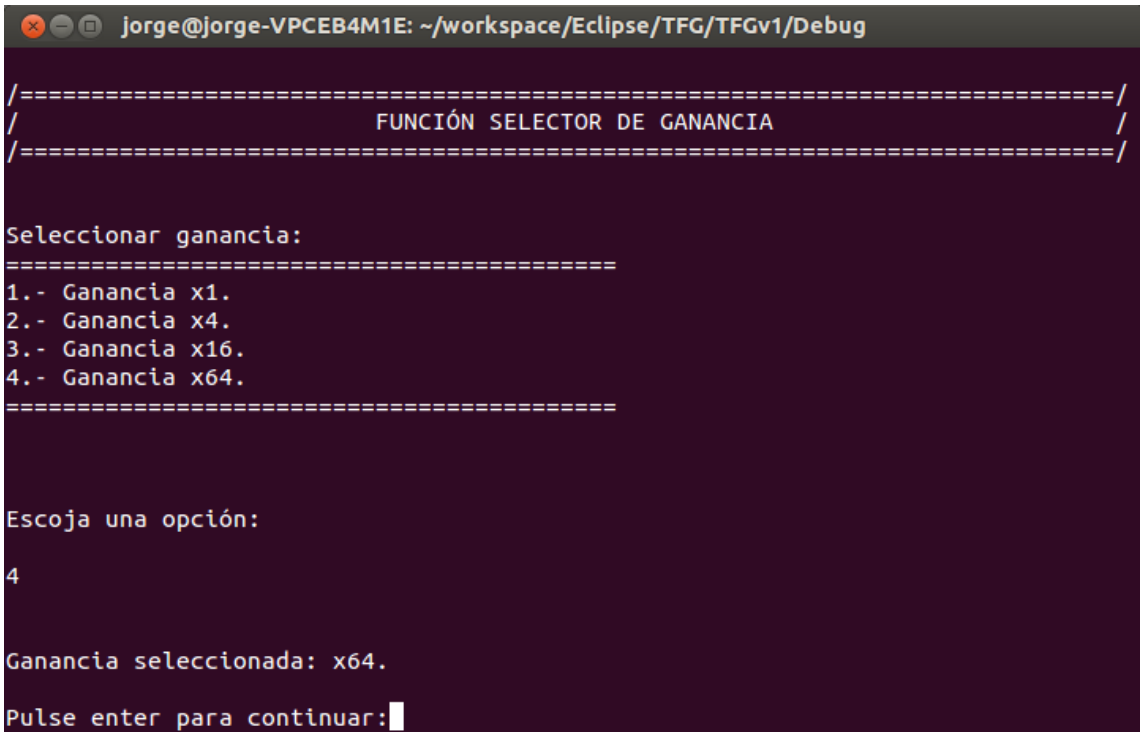


Figura 3.8: La aplicación le indica al usuario las diferentes ganancias seleccionables y le pide a este que escoja una. Después de escoger la ganancia deseada, la aplicación confirma que dicha ganancia ha sido seleccionada correctamente.

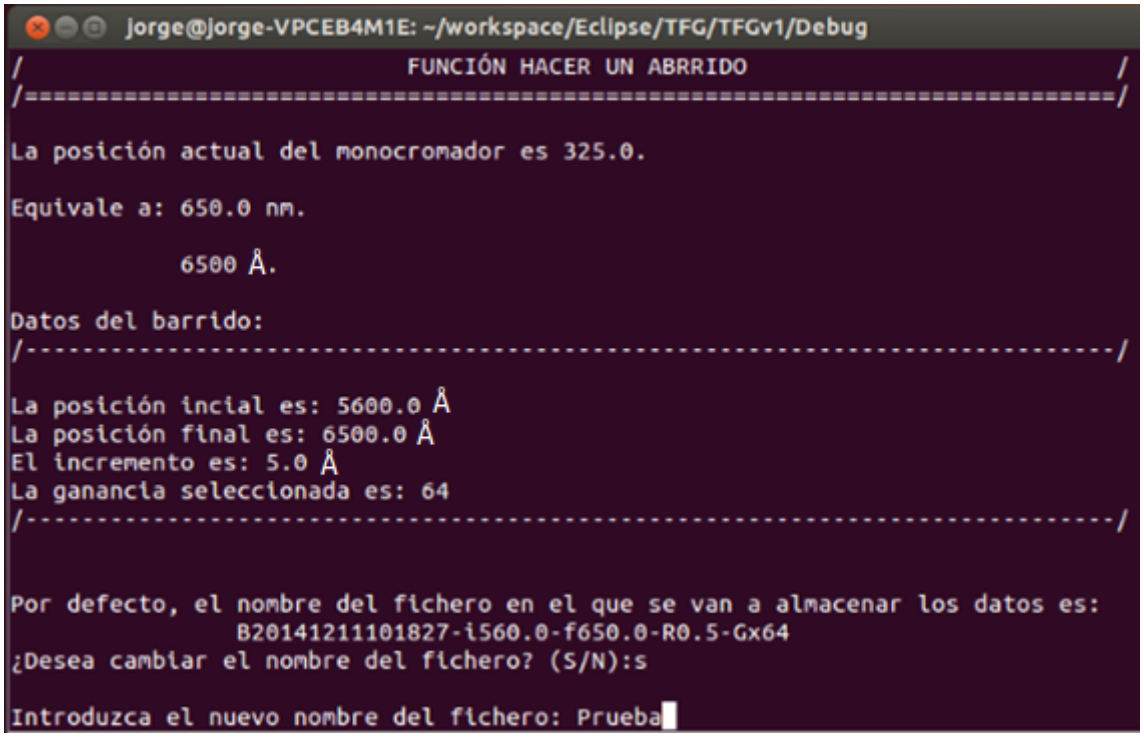


Figura 3.9: Aspecto de la interfaz en el último paso antes de comenzar el barrido. Apreciamos el resumen de las características del barrido que se va a realizar y el nombre por defecto con el que se guardarán los datos si así lo deseamos, en el ejemplo de la figura el usuario decidió cambiar el nombre por defecto por el de “Prueba”.

Antes de escoger el nombre, la aplicación nos muestra un pequeño resumen de las características del barrido que vamos a realizar: posiciones inicial y final, resolución y ganancia seleccionada. En la figura 3.9 podemos apreciar las características del barrido de ejemplo.

Cuando el barrido comienza, la consola de comandos comenzará a mostrarnos dos columnas con información sobre el barrido que estamos realizando, para que podamos comprobar si todo va según lo previsto o hay algún error. La primera columna contiene la posición del monocromador en Å y la segunda la medida obtenida en esa posición en concreto. En la figura 3.10 podemos apreciar el aspecto de la consola de comandos mientras el barrido de ejemplo está realizándose.

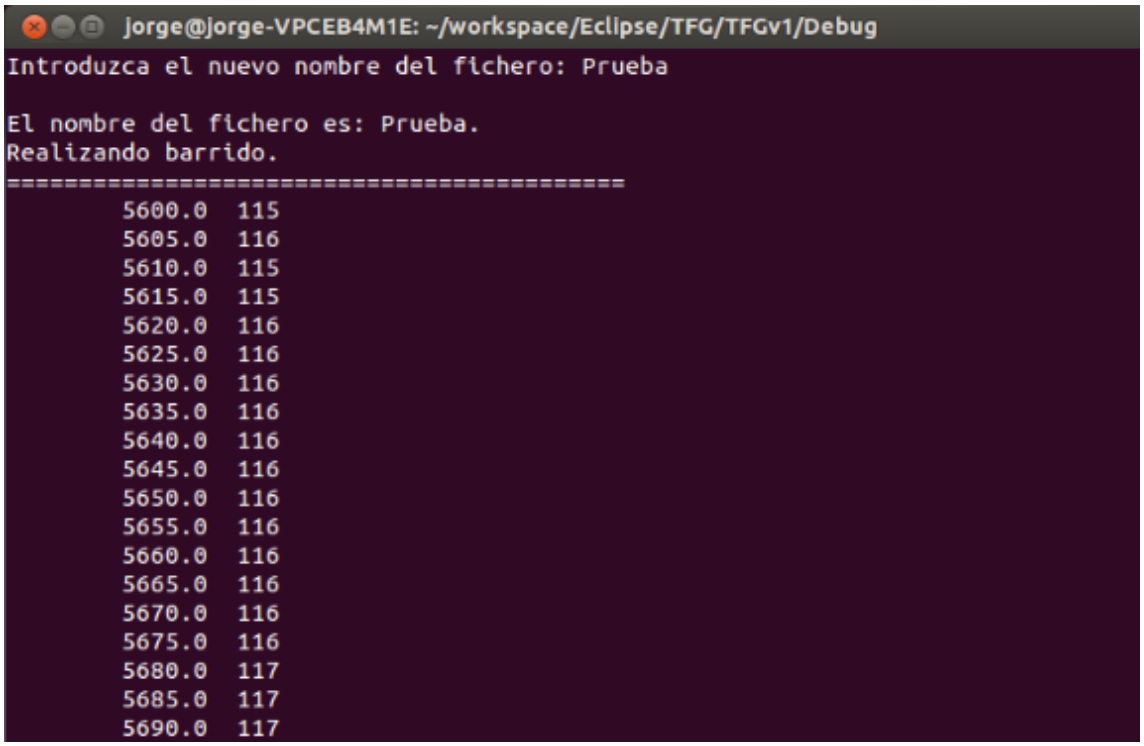


Figura 3.10: Este es el aspecto de la consola de comandos mientras se está realizando un barrido. La aplicación nos muestra dos columnas de datos: la primera es la posición del monocromador en longitud de onda (en Å) y la segunda es la medida obtenida para la señal de LED en esa longitud de onda.

Los datos mostrados por la consola de comandos durante el barrido se almacenan en el fichero de datos. Además de estos datos, en el fichero son almacenados otra serie de datos en forma de columna. A continuación explicaremos cada una de las columnas almacenadas en el fichero "Prueba", el contenido de este fichero puede apreciarse en la figura 3.11:

- 1ª columna: Esta columna contiene la posición del monocromador en cuanto a longitud de onda, está mostrada en Å.
- 2ª columna: En esta columna tenemos almacenadas las medidas obtenidas menos el nivel de ruido (**Ycorr**). El nivel de ruido es el valor que nos da el convertidor A/D sin ninguna fuente de luz, este nivel es debido a la corriente de oscuridad del fotodiodo, la cual podría ser amplificada incluso x64 en algunas ocasiones.

Restamos este nivel a la señal para ganar así espacio en la representación de la misma, ya que el ruido no nos interesa (en el capítulo 5 veremos esto en mayor detalle).

- 3ª columna: Esta columna contiene la medida obtenida sin realizar ningún cambio sobre ella, es la mostrada por la aplicación en la consola de comandos durante la realización del barrido (**Y**).
- 4ª columna: En esta columna almacenamos la medida corregida (calibrada) teniendo en cuenta la respuesta espectral del fotodiodo (**F**) de la figura 2.17, menos el nivel de ruido (**Xcorr = Ycorr/F**). Veremos cómo se realizó la calibración de la señal en detalle en el capítulo 5.
- 5ª columna: En esta otra columna almacenamos la señal calibrada (teniendo en cuenta la respuesta espectral del fotodiodo) sin restarle el nivel de ruido (**X = Y/F**).
- 6ª columna: Esta última columna contiene la respuesta espectral del fotodiodo (**F**), es decir, la función de calibración que relaciona las columnas 2 y 3 con las columnas 4 y 5, respectivamente. En el capítulo 5 veremos cómo ajustamos la respuesta del fotodiodo a una función para obtener los valores de la 6ª columna.

Una vez tenemos almacenado el fichero correctamente, el siguiente paso de la aplicación es representar los resultados y mostrárselos al usuario, para ello, el programa de la aplicación está enlazado con la herramienta gnuplot, que realiza esta representación de forma automática. La representación se realizará de la intensidad del LED frente a la longitud de onda, es decir, la aplicación nos estará mostrando la emisión espectral del LED, lo cual era nuestro objetivo. Luego, para la representación, solamente necesitaremos dos columnas de las seis de que consta el fichero de datos, estas columnas serán la cuarta y la primera. La cuarta columna contiene la intensidad del LED calibrada por la respuesta espectral del fotodiodo menos el nivel de ruido (**Xcorr**), la cual será representada en nuestro eje Y, y la primera columna contiene la longitud de onda correspondiente, la cual será nuestro eje X. Comentar que a la hora de representar los datos hemos normalizado la amplitud de la señal entre 0 y 1, por eso los valores del eje Y de la gráfica no se corresponden con los del fichero de datos. En cuanto al eje X, en la gráfica se muestra la longitud de onda en nanómetros (nm) mientras que en el fichero de datos está en Å, para realizar este cambio de unidades únicamente tenemos que dividir los Å entre 10 para obtener los nanómetros. Puede verse la representación de nuestro barrido de ejemplo en la figura 3.12.

Llegados a este punto, la aplicación ya habría cumplido con su objetivo final pero, además de mostrarle al usuario la representación de la emisión espectral del LED, la aplicación guarda esta representación en un fichero con el mismo nombre que el fichero de datos pero cuya extensión es `.eps`, que es un formato de archivo gráfico.

Tras la finalización del barrido, la aplicación vuelve a mostrarle al usuario el menú principal (figura 3.1) y se mantiene a la espera de recibir la siguiente tarea a realizar.

Prueba

1	5600.0	0	115	0.0000	295.2210	0.3895
2	5605.0	1	116	2.5597	296.9207	0.3907
3	5610.0	0	115	0.0000	293.5063	0.3918
4	5615.0	0	115	0.0000	292.6566	0.3930
5	5620.0	1	116	2.5375	294.3494	0.3941
6	5625.0	1	116	2.5302	293.5023	0.3952
7	5630.0	1	116	2.5229	292.6603	0.3964
8	5635.0	1	116	2.5157	291.8232	0.3975
9	5640.0	1	116	2.5085	290.9910	0.3986
10	5645.0	1	116	2.5014	290.1637	0.3998
11	5650.0	1	116	2.4943	289.3412	0.4009
12	5655.0	1	116	2.4873	288.5234	0.4020
13	5660.0	1	116	2.4803	287.7104	0.4032
14	5665.0	1	116	2.4733	286.9021	0.4043
15	5670.0	1	116	2.4664	286.0985	0.4055
16	5675.0	1	116	2.4595	285.2995	0.4066
17	5680.0	2	117	4.9053	286.9577	0.4077
18	5685.0	2	117	4.8916	286.1610	0.4089
19	5690.0	2	117	4.8781	285.3688	0.4100
20	5695.0	2	117	4.8646	284.5812	0.4111
21	5700.0	2	117	4.8512	283.7981	0.4123
22	5705.0	2	117	4.8379	283.0193	0.4134
23	5710.0	3	118	7.2371	284.6574	0.4145
24	5715.0	3	118	7.2173	283.8808	0.4157
25	5720.0	3	118	7.1977	283.1086	0.4168

Figura 3.11: Aspecto del fichero de datos guardado para el barrido de ejemplo, en este caso se trata de un LED de color amarillo. La columna de la izquierda solamente es el número de línea, no forma parte del contenido del fichero. El contenido del resto de columnas fue explicado anteriormente.

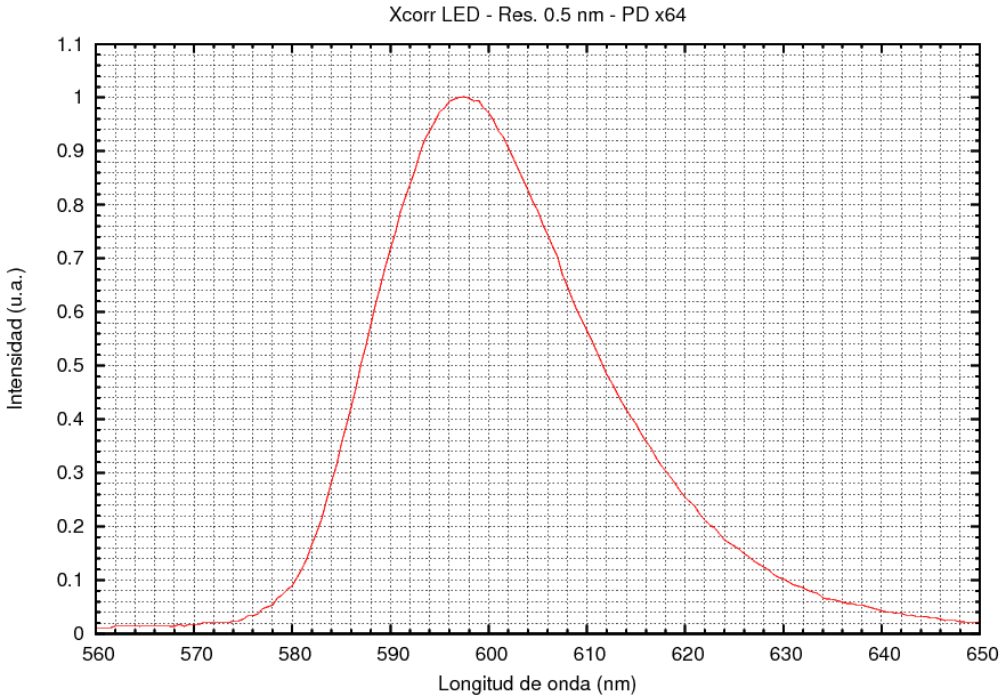


Figura 3.12: Muestra la representación del barrido de ejemplo. En este caso se trata de un LED amarillo y vemos que el máximo se encuentra en torno a 597 nm, los cuales corresponden al rango del espectro de dicho color. Pueden observarse la resolución y la ganancia escogidas para el barrido en cuestión.

3.3 FUNCIONES DEL PROGRAMA

Para que el usuario sea capaz de realizar todas las tareas mencionadas en el apartado anterior, tuvimos que desarrollar un programa en lenguaje C/C++ mediante el entorno de desarrollo integrado (IDE) Eclipse, como ya comentamos anteriormente. En el presente apartado comentaremos los aspectos más relevantes sobre el código desarrollado (el apéndice A explica dónde podemos encontrar dicho código). Para seguir un orden más o menos claro y no liar el asunto, imaginemos que arrancamos la aplicación desde cero y comentemos las diferentes funciones del programa que se irán ejecutando. Antes de meternos de lleno a comentar las diferentes funciones, vamos a mostrar un diagrama de flujo con el que entenderemos la ejecución del programa:

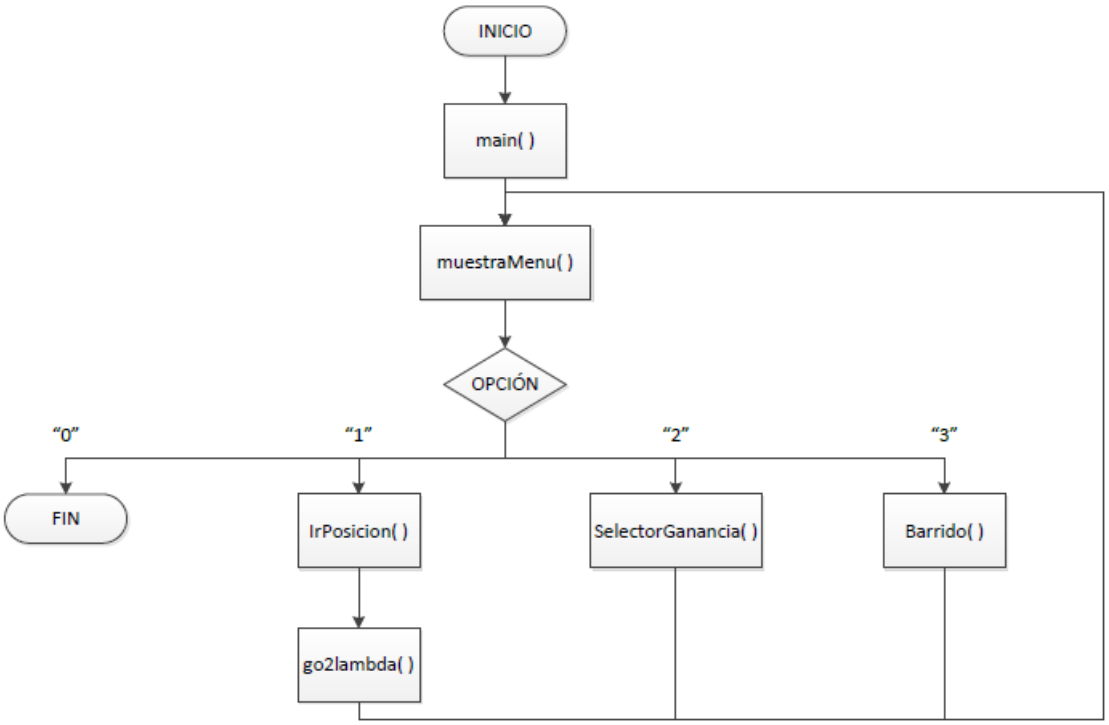


Figura 3.13: Diagrama de flujo que muestra la ejecución del programa del PC.

Al iniciar el programa, la primera función en ejecutarse será la función principal `main()`, esta se encargará del inicio de los *flags* del microcontrolador, etc. (lo explicaremos más adelante). Posteriormente se ejecutará la función `muestraMenu()`, la cual le dará al usuario a elegir entre diversas opciones o tareas a realizar. A continuación, dependiendo de la opción elegida por el usuario, el programa ejecutará la función correspondiente.

Pasemos ahora a explicar cada una de las funciones que aparecen en el diagrama más detenidamente. Comenzaremos por la función `main()`, que es la primera en ejecutarse, y así seguiremos el orden de ejecución del programa.

3.3.1 Función principal

Se trata de la función `main()`, esta es la función principal y es la primera en ejecutarse. Lo primero que deberá realizar dicha función será abrir el puerto USB para el control del microcontrolador y la inicialización de los *flags* del mismo. Debido a la dificultad de dicha tarea, tuvimos que recurrir a la ayuda de Jesús Arias para realizarla.

Una vez inicializado el microcontrolador, lo primero que hace este es dar una primera vuelta al motor paso a paso del monocromador para controlar la posición exacta del mismo. Esta vuelta se realiza comprobando, claro está, los finales de carrera del motor y, en caso de llegar al final de carrera, se advertirá al PC, en el capítulo 4 veremos esto en mayor detalle. Entonces, en la función `main()` comprobaremos que la primera vuelta de control sobre el motor se haya realizado correctamente, es decir, que no se haya alcanzado ningún final de carrera.

Posteriormente, se pasará a pedirle al usuario que introduzca la posición actual del monocromador, es decir, el valor actual mostrado en el *display* del mismo. Llegamos a la conclusión de que esta era la mejor manera de controlar la posición del monocromador. En función del valor introducido por el usuario realizaremos todo lo demás, luego, si el usuario introduce un valor erróneo, arrastraremos ese error durante el resto de la ejecución de la aplicación.

Una vez que el usuario ha introducido el valor actual mostrado en el *display* del monocromador, tendremos que calcular el equivalente de esa posición en nm y en Å. Para calcular el valor equivalente en nm basta con multiplicar por 2 el valor mostrado en el *display*, esto se debe a que contamos con un *grating* de 600 g/mm montado en nuestro monocromador. Entonces, si nos fijamos en las hojas de especificaciones del monocromador (página 15), en el apéndice A se explica dónde pueden encontrarse, veremos que teniendo un *grating* de 600 g/mm, tendremos que multiplicar el valor del *display* por 2 para obtener la posición del monocromador en nm. Cuando ya tenemos calculado el valor equivalente en nm, pasarlo a Å no es más que multiplicar por 10.

Al final de la función `main()` pasaremos a llamar a la función que le mostrará el menú principal al usuario, es decir, la función `muestraMenu()`, la cual explicamos a continuación.

3.3.2 Muestra menú

Esta función es la encargada de mostrarle al usuario el menú principal (figura 3.1), en el que vimos que se mostraban las posibles tareas a realizar, para que el usuario pueda escoger la que desee. Una vez que el usuario haya escogido la tarea deseada, la función `muestraMenu()` se encargará de llamar a la función encargada de realizar dicha tarea.

Normalmente, la función correspondiente a la tarea en cuestión, le devolverá el control a la función `muestraMenu()` al finalizar dicha tarea. Esto no sucede cuando la acción seleccionada es "Salir de la aplicación".

3.3.3 Ir a longitud de onda

Si el usuario selecciona la opción "Ir a una longitud de onda" en el menú principal, la función `muestraMenu()` llamará a la función `IrPosicion()`. Esta función le muestra al usuario la posición actual y le pregunta por la nueva posición a la que desea desplazar el monocromador (figura 3.3). Una vez introducida la nueva posición, se procederá a llamar a la función auxiliar `go2lambda(posNueva)` pasándole como parámetro la nueva posición a la que se desea desplazar el monocromador, explicaremos el funcionamiento de esta función en el apartado siguiente, en el apartado 3.3.4.

Cuando la función auxiliar `go2lambda()` nos devuelve el control, el monocromador ya se encontrará situado en la posición deseada y procederemos a encender el LED, realizar la medida, interpretarla y apagar el LED.

Por último, la función muestra por pantalla el resultado de la medida obtenida y le pide al usuario que pulse una tecla para continuar (figura 3.4), devolviendo así el control a la función `muestraMenu()` que volverá a mostrar el menú principal de tareas realizables.

Para realizar las tareas que estamos comentando, como encender o apagar el LED, mover el monocromador a la posición deseada, etc., deberemos conseguir que el PC le indique correctamente al microcontrolador la tarea a realizar, ya que es el microcontrolador quien se encargará de gobernar el resto de elementos del sistema (como el LED, el motor del monocromador, el fotodiodo, etc.). Entonces, necesitaremos que el PC sea capaz de comunicarle al microcontrolador una serie de órdenes, que las traduciremos en una serie de comandos. Para que el PC le comunique al microcontrolador el comando deseado, este utiliza la función `write()` de la librería `unistd.h`. Hablaremos de estos comandos en el siguiente capítulo cuando expliquemos las comunicaciones entre el PC y el microcontrolador.

3.3.4 Función auxiliar

Como ya hemos comentado en el apartado anterior, esta función es llamada `go2lambda(posNueva)`. A esta función la llamaremos desde diversas funciones para ir a una longitud de onda en concreto, la cual recibirá como parámetro, además la recibirá en Å. Lo primero que hará esta función será comparar la posición actual con la posición nueva restándolas, el resultado de la resta serán los Å que deberemos avanzar o retroceder.

Una vez calculado el número de Å a avanzar o retroceder, tendremos que calcular el número de semi-pasos a realizar. Para calcular los semi-pasos hemos definido una función que los calcula, `lambda2semipasos()`. La función multiplica el número de Å a avanzar o retroceder por 8 y lo divide entre 20, ya que 8 semi-pasos equivalen a 20 Å, es decir, la función realiza una simple regla de tres.

Para saber si debemos avanzar o retroceder lo que hacemos es comprobar el signo de la resta realizada de manera que, si este es negativo significará que la posición nueva es menor que la posición actual y por lo tanto tendremos que retroceder, y, si por el contrario, el signo es positivo significará que deberemos avanzar. Entonces le indicaremos al microcontrolador que avance o retroceda el número de semi-pasos correspondiente. Por último, la función deberá comprobar si la acción fue realizada correctamente o si se alcanzó alguno de los finales de carrera, para ello tendrá que leer la respuesta entregada por el microcontrolador, para ello, el PC utilizará la función `fgets()` de la librería `stdio.h`.

3.3.5 Selector de ganancia

Cuando el usuario selecciona esta opción, la función `muestraMenu()` pasará el control del programa a la función `SelectorGanacia()`. Dicha función le mostrará al usuario las diferentes ganancias disponibles y le pedirá que elija una. Una vez seleccionada la ganancia deseada se comprobará que la opción elegida es válida, si no lo es, se le pedirá que vuelva a seleccionar una de las ganancias mostrándole de nuevo las ganancias disponibles. Cuando la opción elegida por el usuario sea correcta, el PC deberá indicarle al microcontrolador mediante un comando (ya veremos cuál) que seleccione la ganancia en cuestión y, además, la aplicación le indicará al usuario que efectivamente se ha seleccionado dicha ganancia.

3.3.6 Hacer barrido

Como ya comentamos anteriormente, la tarea más potente y completa que el usuario puede realizar es un barrido. Al realizar un barrido, el usuario obtendrá un fichero con los datos del barrido y otro con la representación del mismo en formato `.eps`, que es un formato de archivo gráfico. Anteriormente, utilizamos un barrido de ejemplo para explicar los pasos que debía seguir el usuario para realizar un barrido de forma satisfactoria. Pues bien, ahora pasaremos a explicar cómo la función `Barrido()`, que es la encargada de realizar dicha tarea, va guiando al usuario por los diferentes menús vistos en el apartado 3.2.4.

Primeramente, la función `Barrido()` le pide al usuario que introduzca el rango del barrido. Para ello, muestra un menú (figura 3.6) en el que aparece la posición actual del monocromador, para que el usuario pueda utilizarla como referencia, y le pide las posiciones inicial y final de dicho barrido.

Con el rango del barrido ya seleccionado, pasamos a pedirle al usuario que introduzca la resolución con la que desea realizar el barrido. Para ello, mostramos un nuevo menú con las posibles resoluciones y le pedimos al usuario que elija una (figura 3.7). Comprobaremos que, efectivamente, la resolución elegida se encuentra entre las seleccionables, en caso de no ser así, le indicaremos al usuario que la opción elegida no es válida y le pediremos que elija de nuevo una de las opciones. La máxima resolución posible será para un incremento entre medidas de 2.5 Å (0.25 nm), esto se debe a que el motor paso a paso del monocromador no nos permite realizar un incremento inferior a un semi-paso, que se corresponde con los 2.5 Å, luego tenemos una limitación por *hardware*. Para la mínima resolución posible, simplemente elegimos un incremento entre medidas de 20 Å porque nos pareció que era el adecuado, es el equivalente a una vuelta completa del motor paso a paso.

A continuación, el usuario deberá seleccionar una ganancia entre las distintas opciones. Para ello, la función `Barrido()` lo que hará será llamar a la función `SelectorGanancia()`, ya explicada anteriormente, que le pedirá al usuario que elija una ganancia entre las cuatro posibles. Cuando esa función nos devuelva el control, ya tendremos nuestra ganancia seleccionada. Llegados a este punto, ya tendríamos seleccionados todos los aspectos de nuestro barrido, los cuales la función `Barrido()` nos resume en un nuevo menú (figura 3.9).

Además de resumir las características del barrido que se va a realizar, la función le preguntará al usuario si desea que el fichero de datos se guarde con el nombre por defecto o si desea cambiarlo. En caso de que desee cambiarlo, le pedirá que introduzca el nuevo nombre. Si bien, desea guardarlo con el nombre por defecto, saber que el nombre por defecto contiene las características del barrido (posiciones inicial y final, resolución y ganancia), así como la fecha del mismo, incluimos la fecha para que no se solape con barridos anteriores, ya que dos barridos podrían tener las mismas características. Para obtener la fecha utilizamos la función `localtime()` incluida en la librería `time`, que es una de las librerías estándar del lenguaje C.

Ahora que conocemos el nombre del fichero, podemos abrirlo en modo escritura para ir almacenando en él los datos del barrido a medida que este se va realizando. Llega el momento de pasar a realizar el barrido, para ello, encendemos el LED con el comando correspondiente (del cual hablaremos en el siguiente capítulo) y después entraremos en un bucle. Este bucle se encargará de las siguientes tareas:

- Recorrerá todo el rango del barrido con el incremento seleccionado. Para ello, irá realizando llamadas sucesivas a la función auxiliar `go2lambda(posNueva)` pasándole como parámetro la nueva posición a la que se desea desplazar el monocromador.
- Tomará e interpretará las medidas realizadas.
- Calibrará esas medidas mediante una función de calibración calculada, esto tuvimos que hacerlo ya que la respuesta del fotodiodo no es la misma para todas

las frecuencias, es decir, no tiene la misma sensibilidad en todo su rango espectral. En el capítulo 5 explicaremos cómo llevamos a cabo la calibración

- Almacenará en el fichero las medidas calibradas y sin calibrar, junto con la longitud de onda correspondiente a cada medida. Ya explicamos previamente, en el apartado 3.2.4, el contenido del fichero de datos.
- Mostrará las medidas por pantalla a medida que se van obteniendo, para que el usuario pueda comprobar si el barrido se está realizando correctamente.
- Además, almacenará los máximos de las medidas calibradas y sin calibrar para poder utilizarlos posteriormente a la hora de representar el espectro. Gracias a los máximos podremos representar las medidas normalizadas en el eje y.

Una vez que hemos terminado de realizar el barrido, abandonamos el bucle, apagamos el LED y cerramos el fichero de datos.

Ahora, solo nos queda representar y mostrar al usuario las medidas obtenidas (figura 3.12). Para ello utilizamos la herramienta Gnuplot, que es una herramienta muy útil a la hora de generar gráficas de funciones y datos. Debido a que el alumno no estaba muy familiarizado con la herramienta, se ha necesitado de la ayuda de Iván Santos.

Por último, la función le pide al usuario que pulse una tecla para continuar, devolviendo así el control a la función `muestraMenu()` que volverá a mostrar el menú principal de tareas realizables.

3.3.7 Salir

En realidad, cuando el usuario selecciona la opción “Salir”, la función muestraMenu() no llama a ninguna función que salga de la aplicación como en el resto de los casos, sino que es ella misma la que se encarga de finalizar el programa. Para ello, devuelve el control a la función `main()` que realiza un “`return 0;`” finalizando así la ejecución del programa devolviendo el control a la consola de comandos de Linux.

3.4 CONCLUSIÓN

Ahora que ya hemos comentado el código del PC (de la aplicación), podemos pasar a comentar el código grabado en el microcontrolador. En realidad, la programación de la aplicación se llevó a cabo después de la del microcontrolador. Esto tiene su lógica, ya que sin el programa del microcontrolador, no sabríamos si todo esto que hemos estado explicando funcionaría. Pero a la hora de explicarlo, nos pareció que la forma más sencilla era comentar primero las tareas que el sistema puede realizar, después pasar a la explicación de las funciones que ejecutan cada tarea y, por último, comentar cómo el PC le indica al microcontrolador la tarea que debe realizar y cómo este procede a su realización. Estas dos últimas partes las veremos en el siguiente capítulo, en el cuál explicaremos los aspectos más importantes del programa grabado en el microcontrolador así como los comandos utilizados para la comunicación entre este y el PC.

CAPÍTULO 4. PROGRAMACIÓN MICROCONTROLADOR

4.1 INTRODUCCIÓN

Como ya mencionamos en capítulos anteriores, el microcontrolador debe realizar las siguientes funciones:

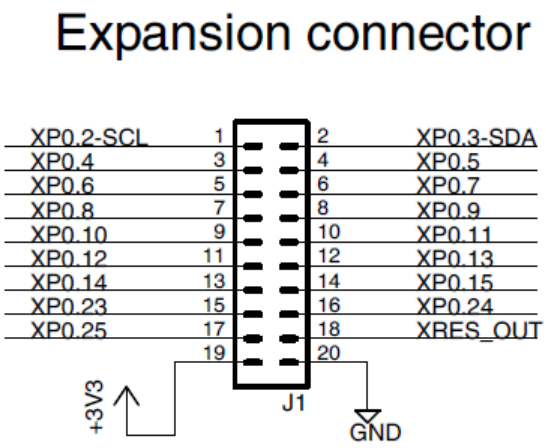
- Mover el motor del monocromador, básicamente avanzar y retroceder.
- Encender y apagar el LED.
- Medir e interpretar la señal procedente del fotodiodo.
- Seleccionar una ganancia para dicha señal.

Para que el microcontrolador pueda encargarse de las tareas mencionadas necesitaremos:

- Asignar uno o varios pines GPIO del microcontrolador a cada función.
- Programar el microcontrolador para que este realice las diferentes tareas.
- El PC deberá ser capaz de comunicarse con el microcontrolador para indicarle la tarea a realizar.

La asignación de pines se llevó a cabo durante las prácticas en empresa, teniendo en cuenta que necesitábamos una entrada al ADC (`XP0.10`), para la señal del fotodiodo, y un temporizador (`XP0.12`), para controlar los tiempos de conversión del ADC. En la figura 4.1, junto con la tabla 4.1, mostramos la información correspondiente a la disposición de los pines del microcontrolador LPC2103.

Una vez mostrada la asignación de los pines del microcontrolador a cada función, solo nos resta comentar los diferentes comandos utilizados para la comunicación entre el PC y el microcontrolador, además de los aspectos más importantes del programa grabado en la memoria flash del microcontrolador. Estos comentarios los realizaremos a lo largo del presente capítulo.



Pin	Función
XP0.4	Motor X+
XP0.5	Motor X-
XP0.6	Motor Y+
XP0.7	Motor Y-
XP0.8	LED
XP0.10	Fotodiodo
XP0.13	Final Carrera Alto
XP0.15	Final Carrera Bajo
XP0.23	Selector Ganancia
XP0.25	Selector Ganancia

Figura 4.1 y Tabla 4.1: Conector de expansión de la placa *Educational Board* LPC2103 junto con las funciones que realiza cada uno de los pines de la misma [4].

4.2 COMUNICACIÓN ENTRE PC Y MICROCONTROLADOR

A lo largo del capítulo anterior, comentamos en varias ocasiones que el PC iba a necesitar enviarle órdenes al microcontrolador, para que este realizase la tarea adecuada en cada momento. La idea es que el microcontrolador se mantenga a la espera de que el PC le envíe una orden y, cuando la reciba, deberá ejecutarla. Las órdenes están directamente relacionadas con las tareas que el microcontrolador debe realizar, comentadas en el apartado anterior. Así pues, a continuación mostramos los diferentes comandos que hemos ideado para que el PC pueda comunicarse con el microcontrolador:

- **Ax** (Avanzar x semi-pasos): Con esta orden el PC le indica al microcontrolador que quiere que el motor del monocromador avance y la cantidad de semi-pasos a avanzar (la x representa el número de semi-pasos).
- **Rx** (Retroceder x semi-pasos): Similar a la orden anterior, la diferencia es que en este caso queremos que el motor del monocromador retroceda (la x nuevamente representa el número de semi-pasos).
- **E** (Encender LED): El PC le indica al microcontrolador que encienda el LED.
- **O** (Apagar (OFF) LED): El PC le indica al microcontrolador que apague el LED. Utilizamos la O de *off* porque la A de apagar ya estaba siendo utilizada en la orden de avanzar.
- **M** (Medir): Con esta orden el PC le está indicando al microcontrolador que realice una medida de la señal del fotodiodo. El microcontrolador devolverá el resultado de la medida realizada.
- **Gx** (Ganancia x): El PC le indica al microcontrolador la ganancia que desea aplicarle a la señal procedente del fotodiodo (en este caso, x representa la ganancia). Tenemos 4 ganancias seleccionables, es por esto que la x solo puede tomar valores entre 1 y 4:
    - x=1: La ganancia seleccionada es 1.
    - x=2: La ganancia seleccionada es 4.
    - x=3: La ganancia seleccionada es 16.

- x=4: La ganancia seleccionada es 64.

Ya comentamos en el apartado 2.4.6 cómo conseguíamos estas ganancias seleccionables.

Una vez realizada la tarea en cuestión, el microcontrolador deberá indicárselo al PC, en nuestro caso, respondemos con la señal de tarea terminada, OK. Además, en el caso de que la orden fuese realizar una medida, el microcontrolador deberá responderle al PC con el resultado de la medida realizada. En la figura 4.2 puede apreciarse un ejemplo del funcionamiento del programa del microcontrolador. En este caso estamos ejecutando directamente el programa del microcontrolador, no lo hacemos por medio del programa del PC, de esta manera somos nosotros mismos quienes estamos representado el papel del PC y quienes estamos enviándole las órdenes al microcontrolador. Las órdenes y las respuestas son enviadas en código ASCII por el puerto serie USB, para facilitar la comunicación entre ambos dispositivos (PC y microcontrolador).

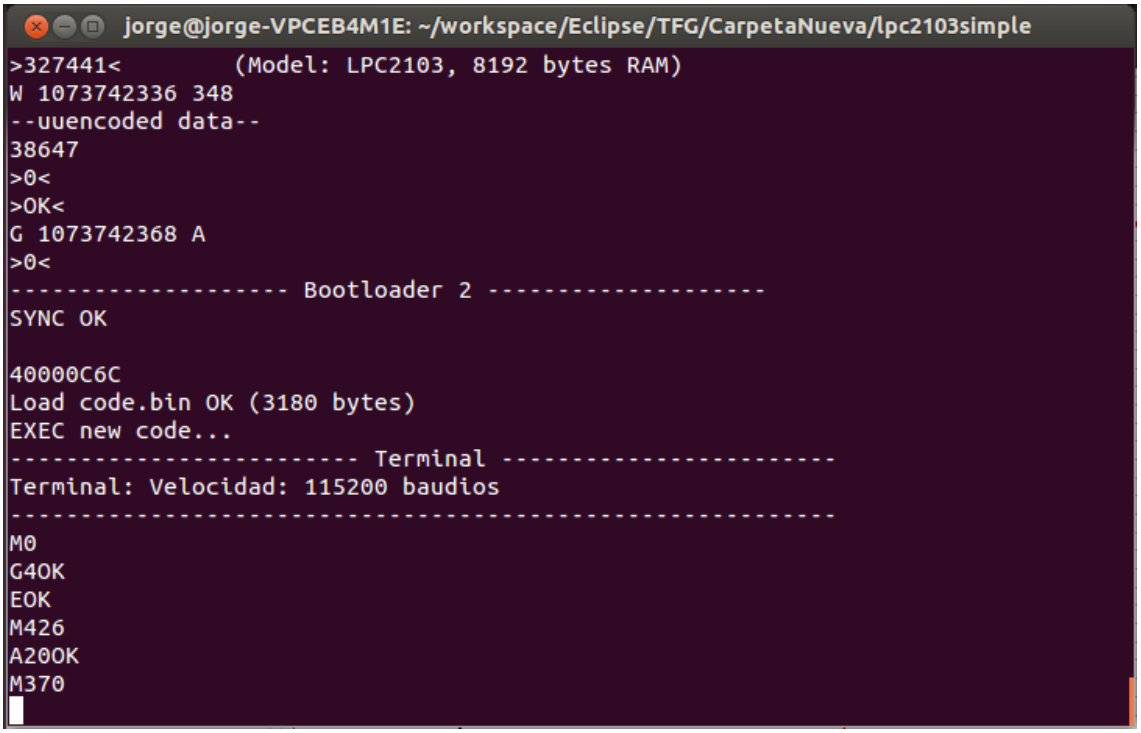


Figura 4.2: Captura que muestra el funcionamiento del programa del microcontrolador, en este caso es el usuario, en lugar del PC, quien le indica las órdenes a realizar. Cabe resaltar que el usuario final de la aplicación no va a ver este intercambio de órdenes y respuestas entre PC y microcontrolador.

Ya comentamos en el capítulo anterior que para llevar a cabo la comunicación por parte del PC utilizábamos las funciones `write()` y `fgets()`, para escribir y leer por el puerto USB, respectivamente. Pues bien, para llevar a cabo la comunicación por parte del microcontrolador, utilizamos otras dos funciones, las cuales han sido proporcionadas e implementadas por Jesús Arias:

- `_gets()`: Esta función la utilizamos para la lectura de las órdenes procedentes del PC (en el caso de la figura era el usuario quien las enviaba, pero el

funcionamiento es el mismo para el caso del PC). Explicaremos su funcionamiento detalladamente en el apartado 4.3.1.3.

- `_puts()`: En cambio, esta otra función la utilizamos para realizar la escritura en pantalla o, en el caso de comunicación con el PC, para realizar la escritura a través del puerto serie USB.

Pueden verse ejemplos de la utilización de ambas funciones en código grabado en la memoria flash del microcontrolador, el apéndice A indica donde encontrar dicho código.

4.3 COMENTARIOS ACERCA DEL CÓDIGO

La programación la llevamos a cabo con ayuda de un PC, en este caso no utilizamos el entorno de desarrollo Eclipse como cuando programamos la aplicación del PC, en este caso utilizamos un simple editor de textos, el editor gedit, por defecto incluido en el sistema operativo Linux Ubuntu 12.04 LTS. Las pruebas correspondientes en el microcontrolador pudieron ser realizadas gracias a unas herramientas proporcionadas por Jesús Arias, que nos permitieron utilizar los comandos `make` y `makeburn`, de los que ya hablamos anteriormente en el apartado 2.5.

Antes de pasar a explicar detalles acerca del código en sí, veamos un diagrama de flujo del mismo para entender mejor su funcionamiento.

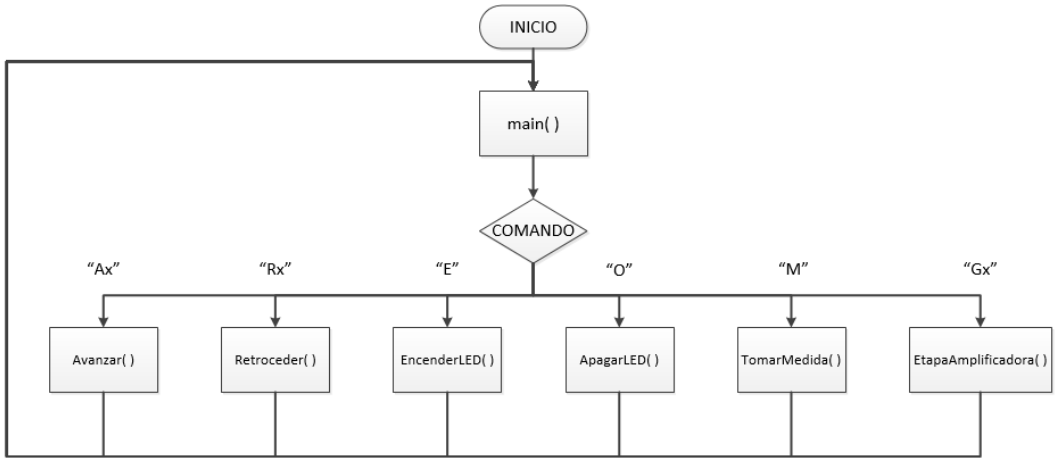


Figura 4.3: Diagrama de flujo del programa grabado en la memoria flash del microcontrolador.

En el diagrama de flujo podemos apreciar las diferentes funciones de que consta el programa. Observamos que consta de una función principal `main()` y de varias funciones que se corresponden perfectamente con cada uno de los comandos explicados en el apartado anterior, pues bien, el programa se encargará de ejecutar unas funciones u otras en función del comando introducido.

A continuación, explicaremos los aspectos más importantes del código de cada una de las diferentes funciones. Se recomienda observar el código mientras se leen los

siguientes apartados para su mejor comprensión, el apéndice A indica dónde encontrar dicho código. Empezaremos por la función principal, la función `main()`.

4.3.1 Función principal

Como ya hemos comentado previamente, se trata de la función `main()`, esta va a ser la primera función en ejecutarse y, además, consta de 3 partes bien diferenciadas:

4.3.1.1 Configuración de pines

Al iniciar el microcontrolador, deberemos configurar sus pines, para ello deberemos utilizar una serie de registros de que consta el mismo. A continuación vamos a mencionar algunas de las configuraciones que hemos necesitado realizar.

La primera de las configuraciones será indicarle al microcontrolador que deseamos que utilice las FGPIO, entradas/salidas de propósito general rápidas. Para ello utilizamos el registro SCS (*System Control and Status flags register*).

El siguiente paso será configurar algunos de los pines del microcontrolador para que no actúen como entradas/salidas de propósito general. En nuestro caso, hemos configurado el registro PINSEL0 (*Pin function Select register*) para que los pines `XP0.0`, `XP0.1`, `XP0.10` y `XP0.11` funcionen como UART de salida, UART de entrada, entrada al convertidor A/D y temporizador, respectivamente.

A continuación deberemos indicarle al microcontrolador cuáles de los pines FGPIO van a operar como salidas, para ello utilizamos el registro FIODIR (*Fast GPIO Port Direction control register*). En nuestro caso, los pines que operan como salida son los que se encargan de controlar el motor (`XP0.4`, `XP0.5`, `XP0.6` y `XP0.7`), de controlar el LED (`XP0.8`) y de controlar el selector de ganancia (`XP0.23` y `XP0.25`).

También deberemos configurar el temporizador, indicándole los parámetros adecuados, para obtener la señal de reloj deseada, en nuestro caso hemos configurado el temporizador para obtener una señal de reloj cuyo periodo es de 1 ms. Recordemos que vamos a utilizar esta señal de reloj para controlar los tiempos de conversión del ADC, luego este realizará una conversión cada milisegundo.

En el código aparecen, en forma de comentarios, las explicaciones de por qué le damos a cada registro ese valor en concreto, si se quiere profundizar en la configuración de los pines se recomienda echarle un vistazo.

4.3.1.2 Vuelta inicial del motor

Recordemos que el motor paso a paso consta de una serie de bobinas, las cuales generarán un campo magnético dependiendo de la corriente que circule por ellas. La variación de la dirección de ese campo magnético producirá movimiento de seguimiento por parte del rotor de imán, el cual intentará alinearse con el campo magnético inducido por las bobinas.

Cuando inicializamos el sistema, no sabemos en qué punto (o paso) se encuentra el rotor de imán. Al no conocer la posición del rotor, no sabemos cuál de las bobinas (pines del microcontrolador) activar para que el motor realice el siguiente paso. Es por esto que realizamos una vuelta completa inicial, de esta forma nos aseguramos de colocar el rotor en una posición conocida, conociendo así la bobina (o el pin) a activar para realizar el siguiente paso.

Con esta vuelta inicial terminamos con la parte de inicialización del sistema. La forma de implementar esta primera vuelta es la misma que la de las funciones `Avanzar()` y `Retroceder()`, lo veremos en mayor detalle más adelante.

4.3.1.3 Bucle infinito a la espera de comando

Como ya comentamos anteriormente, el microcontrolador debe permanecer a la espera de recibir un comando procedente del PC. Cuando reciba el comando deberá interpretarlo, ejecutarlo y permanecer a la espera de un nuevo comando. Para conseguirlo necesitaremos de un bucle infinito en el que primero se lea el comando introducido y después se interprete dicho comando para llamar a la función encargada de realizar la acción deseada.

Entonces, el primer paso es leer el comando. Para ello utilizamos, como ya comentamos anteriormente, una de las funciones implementadas por Jesús Arias, se trata de la función `_gets()`. Procedemos a explicar el funcionamiento de esta función a continuación:

```
k = _gets(pw, w);
```

`k` es una variable entera que va a almacenar el número de caracteres que posee el comando introducido, `w` es otra variable entera en la que se almacena el comando y, por último, `pw` es un puntero que apunta a la dirección de memoria de `w`.

La única complicación para leer el comando es entender como la variable `w` almacena el comando introducido. Comentar que todo comando debe terminar con un salto de línea y un retorno de carro, es decir, con `\n`. Pues bien, para comprender como se almacena el comando en la variable `w` lo mejor es poner un ejemplo:

Supongamos que el comando introducido es el siguiente: **`A230\n`**

Entonces, la variable `w` tendrá almacenado lo siguiente: **`032A`**

Este almacenamiento se debe a que el UART utiliza la disciplina de cola FIFO, como ya comentamos en el capítulo 2 cuando describíamos las características del microcontrolador. Cada carácter del comando tiene una longitud de 8 bits, entonces la forma de leer el comando que hemos implementado es coger cada carácter por separado empezando por el de la izquierda, para ello tendremos que utilizar la variable `k`. Una vez que tenemos el carácter en cuestión deberemos decodificarlo, recordemos que viene en código ASCII. Después, iremos almacenando los caracteres numéricos (es decir,

todos menos el último) en una variable llamada `num`, que almacenará el factor de ganancia o el número de pasos a avanzar o retroceder. Deberemos tener en cuenta que cuando lleguemos al último carácter, el de los 8 bits menos significativos, se tratará de la letra del comando, la cual almacenamos en otra variable distinta a la anterior, en este caso utilizamos la variable `comand`.

Una vez que hayamos leído el comando deberemos pasar a la interpretación del mismo. La interpretación se puede realizar con un simple `switch` de la variable `comand`, solo hay que tener en cuenta nuevamente la codificación ASCII. Una vez interpretado el comando, solo habrá que llamar a la función relacionada con el mismo, además, en el caso de las funciones `Avanzar()`, `Retroceder()` y `EtapaAmplificadora()`, deberemos pasarle a la función la variable `num`, ya que contendrá en número de pasos a avanzar o la ganancia a seleccionar.

Cuando la función a realizar se finalice y devuelva el control a la función `main()` esta deberá avisar al PC de este hecho devolviendo la señal `OK`, excepto en el caso de la función `TomarMedida()` que devuelve el resultado de la medida realizada.

4.3.2 Avanzar y Retroceder

En realidad son dos funciones distintas pero podemos explicarlas en un solo apartado debido a su similitud. Expliquemos una de ellas, por ejemplo, la función `Avanzar(int semiPasos)`, ya que, una vez entendamos el funcionamiento de esta, el de la otra será evidente.

En primer lugar necesitamos saber el número de semi-pasos que se desea avanzar (o retroceder), recordemos que este número fue introducido en el comando, `A200\n` (o `R500\n`), y que después de interpretarlo lo introdujimos en una variable llamada `num`. Pues bien, necesitaremos pasarle esa variable a nuestra función `Avanzar(int semiPasos)`(o `Retroceder(int semiPasos)`), como comentamos en el apartado anterior.

Una vez conocemos el número de semi-pasos a avanzar (o retroceder), solo tendremos que avanzar los semi-pasos de uno en uno hasta llegar al total, es decir, tendremos que realizar un bucle, explicaremos cómo realizamos estos semi-pasos en el siguiente subapartado. Antes de dar el semi-paso, tendremos que tener en cuenta que podríamos tener el *grating* del monocromador en uno de sus límites, es decir, tendremos que comprobar los finales de carrera (final de carrera por arriba en la función `Avanzar(int semiPasos)` y final de carreara por abajo en la función `Retroceder(int semiPasos)`). Necesitaremos recordar que estos finales de carrera son activos en baja a la hora de programar.

En el caso de que se hayan podido realizar todos los semi-pasos, le devolvemos el control del programa a la función `main()`, la cual enviará la señal `OK` al PC. En el

caso de que hayamos llegado al final de carrera (ya sea por arriba o por abajo), le enviaremos el correspondiente mensaje de aviso al PC (`fcA` o `fcB`) y le devolveremos el control a la función `main()`, que enviará el mensaje `OK` de todas formas, pero el PC ya estará informado de lo que está sucediendo y actuará en consecuencia advirtiendo al usuario.

Hasta aquí llegaría el funcionamiento de estas 2 funciones, pasemos ahora a explicar cómo realizamos los semi-pasos, es decir, cómo controlamos los pines del motor para hacer que conduzcan las bobinas deseadas y en la dirección adecuada.

4.3.2.1 Control sobre motor paso a paso

Antes de pasar a los pines GPIO del microcontrolador, recordemos cómo los pines 11, 12, 13, 14 y 15 del motor paso a paso controlan las bobinas (ya se habló sobre esto en el apartado 2.2.2). El pin 11 corresponde a la toma central de las bobinas, mientras que los otros pines permiten mover el motor haciendo circular la corriente por las bobinas. Recordemos que en nuestro diseño durante las prácticas en empresa decidimos conectar el pin 11 del monocromador a alimentación (12 V), lo que hacía que el resto de los pines para el control del motor fuesen activos en baja, ya que no circula ninguna corriente cuando se encuentran conectados a la alimentación. Pero, como vimos en el apartado 2.2.2, durante las prácticas en empresa se diseñó un circuito para el acondicionamiento de las salidas GPIO del microcontrolador que controlaban los pines del motor paso a paso, gracias al cual, a la hora de programar podíamos ver las salidas GPIO como si fuesen activas en alta.

Entonces, la tabla de orden de fases del motor quedaría como se muestra en la figura 4.4, en este caso concreto el motor tendrá un paso angular de 90º y un semi-paso de 45º (al excitarse más de una bobina). En la figura pueden apreciarse los pines que deben activarse para orientar al motor en una determinada dirección, además, también se aprecia la secuencia a seguir para hacer que el motor gire.

Orientación	Y-	Y+	X-	X+
Pines Motor	15	13	12	14
Pines micro	XP0.7	XP0.6	XP0.5	XP0.4
	0	0	0	1
	1	0	0	1
	1	0	0	0
	1	0	1	0
	0	0	1	0
	0	1	1	0
	0	1	0	0
	0	1	0	1

Avanzar

Retroceder

Núm. en decimal
1
9
8
10
2
6
4
5

Figura 4.4: Tabla de orden de fases del motor [4].

Por consiguiente, el algoritmo que hemos implementado en el microcontrolador para controlar el motor, básicamente lo que hace es activar los pines según le indica la tabla de orden de fases de la figura 4.4. Utilizamos los números en decimal para activar los

pines del microcontrolador de una forma más sencilla, simplificando así el código en gran medida.

4.3.3 Encender y Apagar LED

Estas dos funciones (`EncenderLED()` y `ApagarLED()`) podemos explicarlas en un solo apartado debido a su similitud. La única diferencia entre ambas será que tendremos que poner el pin del microcontrolador correspondiente en alta para encender el LED y en baja para apagarlo. Recordemos que el pin del microcontrolador destinado a controlar el LED es el `XP0.9`, esto puede verse en la introducción del presente capítulo, en la figura 4.1 y la tabla 4.1.

Durante las prácticas en empresa diseñamos el circuito para controlar el LED, aunque en realidad no hubo que realizar un gran diseño, simplemente tuvimos que incluir una resistencia en serie con el LED. .

4.3.4 Tomar medida

La función `TomarMedida()` es la encargada de leer la señal procedente del fotodiodo. Como ya comentamos anteriormente, la señal del fotodiodo es analógica y necesitamos pasarla al dominio digital, para ello, utilizamos el convertidor A/D presente en el microcontrolador LPC2103. Tenemos 8 posibles entradas al convertidor, en nuestro caso utilizamos la `AD0.3`, a la cual tenemos acceso desde el pin `XP0.10` del microcontrolador. Para tomar este pin como entrada al ADC y que el convertidor funcione de la forma deseada necesitamos de una configuración previa, la cual ya fue comentada anteriormente (en el apartado 4.3.1.1, en la función `main()`).

Para leer el resultado de una conversión A/D utilizaremos el registro AD0GDR (*A/D Global Data Register*). La forma de obtener dicho resultado se basará en los dos pasos que mostramos a continuación:

- Comprobar el bit DONE del registro ADGDR: Cuando este bit valga 1 significará que el resultado está disponible y pasaremos al siguiente paso.
- Leer el resultado de la conversión: Este se encontrará almacenado en el campo RESULT del registro ADGDR, una vez leído el resultado lo almacenaremos en una variable.

Comprobando el funcionamiento del sistema nos dimos cuenta de que no siempre obteníamos el mismo resultado, para un LED y una longitud de onda concretos. Esto era debido a que la señal del fotodiodo fluctuaba bastante a causa del ruido, entonces, decidimos que lo mejor era hacer varias medidas y, posteriormente, quedarnos con el valor medio de todas ellas. Con esta solución ya obteníamos los resultados deseados, pero tenía un inconveniente, y este era el tiempo que tardábamos en dar el resultado, que se había visto ampliado notablemente. Teníamos que buscar un compromiso entre obtener un buen resultado de y emplear para ello un tiempo razonable, esto lo

realizamos mediante el método heurístico "ensayo y error". Al final, llegamos a la conclusión de que el mínimo número de medidas que obtenían un buen resultado eran 75. Recordemos que el convertidor A/D lo habíamos configurado previamente para realizar una conversión cada milisegundo, entonces, en realizar una medida tardaremos 75 milisegundos. En el capítulo 6 comentaremos cómo podríamos reducir este tiempo en una posible versión futura del sistema.

4.3.5 Etapa Amplificadora

En el apartado 2.4.6 comentamos cómo acondicionamos la señal procedente del fotodiodo para ser convertida al dominio digital posteriormente por el ADC. Como recordaremos, el acondicionamiento constaba de una etapa amplificadora en la cual podíamos seleccionar una ganancia entre cuatro posibles opciones. Pues bien, los encargados de seleccionar la ganancia deseada serán dos de los pines del microcontrolador (`XP0.23` y `XP0.25`, el más y el menos significativo, respectivamente). En la tabla 4.2 mostramos la orden del PC, así como los valores de las salidas GPIO del microcontrolador y las ganancias seleccionadas en cada caso.

Comando	XP0.23	XP0.25	Ganancia
G1	0	0	x1
G2	0	1	x4
G3	1	0	x16
G4	1	1	x64

Tabla 4.2: Tabla de selección de ganancias [4]. La tabla contiene la orden del PC, el valor que deberán tomar los pines `XP0.23` y `XP0.25` del microcontrolador y la ganancia obtenida.

Entonces, la programación del microcontrolador, para seleccionar la ganancia deseada, simplemente será activar las salidas GPIO (`XP0.23` y `XP0.25`) según la tabla 4.2, dependiendo del comando introducido por el PC. Recordemos que al llamar a la función `EtapaAmplificadora(int opcion)`, le pasamos una variable entera que se corresponde con el número introducido por el PC en el comando, que solo puede tomar valores entre 1 y 4 (G[1..4]). Luego esta función tendrá que comprobar el valor de esa variable utilizando, por ejemplo, `switch(opcion)` y dependiendo del valor de la variable seleccionará el valor de los pines `XP0.23` y `XP0.25` del microcontrolador siguiendo la tabla 4.2. Una vez los pines tienen asignados sus valores correspondientes le podemos devolver el control a la función `main()` para que le envíe al PC la señal de `OK` y pase a esperar el siguiente comando.

4.4 CONCLUSIÓN

Hasta aquí llegaría la realización del sistema tanto *hardware* como *software*, solamente faltaría realizar la calibración del mismo. En el presente TFG, hemos comentado sobre todo la parte *software* ya que la parte *hardware* fue realizada durante las prácticas en empresa.

A grandes rasgos, a lo largo de este capítulo hemos visto como conseguimos que el PC y el microcontrolador pudiesen comunicarse. Además, comentamos los aspectos más reseñables del código grabado en la memoria flash del microcontrolador.

Como comentamos en el capítulo anterior, el código grabado en el microcontrolador se desarrolló antes que el código de la aplicación del PC. Antes de pasar a desarrollar el código de la aplicación, tuvimos que realizar diversas pruebas sobre el programa grabado en el microcontrolador para comprobar su funcionamiento. Para ello necesitábamos ejecutar el programa en el microcontrolador, pero, gracias al comando `make` proporcionado por Jesús Arias, no necesitamos grabarlo en la memoria flash, que recordemos que puede ser reescrita un número finito de veces.

Cuando el programa del microcontrolador funcionó correctamente, sí que procedimos a grabarlo en la memoria flash del mismo, para ello utilizamos otro de los comandos proporcionados por Jesús, en este caso se trata del comando `makeburn`. Ya hablamos del funcionamiento de estos comandos en el apartado 2.5 del capítulo 2.

Una vez que el programa del microcontrolador estuvo grabado en la memoria flash y su funcionamiento fue el correcto, se pasó a desarrollar el código de la aplicación, el cuál fue comentado en el capítulo anterior. Y cuando el código de la aplicación pasó a estar lo suficientemente avanzado, pasamos a incluir las comunicaciones entre el PC y el microcontrolador, debido a su complejidad necesitamos de la ayuda de Jesús. Posteriormente, simplemente hubo que pulir algunos errores tanto del código de la aplicación como del microcontrolador.

En el siguiente capítulo explicaremos el concepto de calibración, por qué era necesaria en nuestro sistema, cómo realizamos su incorporación al código y las diferentes dificultades que hemos tenido que solucionar durante la misma.

CAPÍTULO 5. CALIBRACIÓN DEL FOTODIODO

5.1 INTRODUCCIÓN

Recordemos que el objetivo final del sistema es caracterizar un LED, es decir, representar su espectro de emisión de la forma más fiel posible. El problema es que la señal emitida por el LED se va a ver afectada por diferentes factores:

- Para empezar, antes de que la señal pueda ser captada por el fotodiodo, esta se ve afectada por el ruido, producido, por ejemplo, por la luz ambiente, este efecto produce una variación aleatoria en la señal.
- Un factor relacionado con el anterior es el nivel de ruido, el cual definimos como el valor que nos da el convertidor A/D sin ninguna fuente de luz. Este nivel es debido a la corriente de oscuridad del fotodiodo, la cual puede ser amplificada incluso por un factor de 64 en algunas ocasiones.
- Otro factor importante es que la respuesta espectral del fotodiodo depende de la longitud de onda, es decir, el fotodiodo no es igual de sensible a todas las longitudes de onda, por lo que la representación que realizaremos del espectro de emisión del LED aparecerá modificada, pareciendo que el LED emitió con mayor intensidad en las regiones que el fotodiodo es más sensible, y viceversa.
- El último factor sería que no conocemos la magnitud de la intensidad emitida por el LED, el convertidor A/D del microcontrolador solamente nos ofrece un valor numérico entre 0 y 1023.

Entonces, nuestro objetivo a lo largo del presente capítulo será calibrar el sistema, es decir, corregir todos estos factores mencionados anteriormente.

Comenzaremos por reducir el ruido sobrepuesto a la señal emitida por el LED, para reducirlo lo que hacemos es promediar ese nivel de ruido, es decir, realizamos numerosas medidas de la señal y tomamos el valor medio de todas ellas como resultado final. Se comprobó empíricamente que el número mínimo de medidas que había que realizar para obtener buenos resultados eran 75.

A continuación cuantificaremos el fondo de escala de ruido, esta cuantificación deberemos realizarla, claro está, para cada una de las ganancias seleccionables, ya que el ruido se ve también amplificado por dichas ganancias. De esta forma, podremos eliminar de las representaciones el nivel de ruido, ya que este no nos interesa. Aclarar

que no eliminamos el ruido de la señal, sino que reducimos el nivel de ruido a cero en la representación, ganando así espacio para el espectro de emisión del LED propiamente dicho.

El siguiente paso será calibrar la deformación producida por el fotodiodo sobre la señal. Para entender mejor el concepto, fijémonos en la figura 5.1. Como podemos observar, la señal emitida por el LED es llamada **X**, cuando esta señal pasa a través del monocromador, lo que hace este es quedarse con la intensidad emitida en una "única" longitud de onda, **X(λ)**. Entonces, cuando queramos representar el espectro de emisión, deberemos barrer las longitudes de onda deseadas con ayuda del monocromador. Una vez que la señal ha superado el monocromador, le toca el turno al fotodiodo. Como hemos comentado anteriormente, este tiene una respuesta espectral **F(λ)**, que depende de la longitud de onda y modifica la señal **X(λ)**, obteniéndose así la señal final **Y(λ)**, que es la que le entregamos al convertidor A/D incluido en el microcontrolador y, por lo tanto, la señal que representamos.

$$\mathbf{Y}(\boldsymbol{\lambda}) = \mathbf{F}(\boldsymbol{\lambda}) \cdot \mathbf{X}(\boldsymbol{\lambda}) \tag{5.1}$$

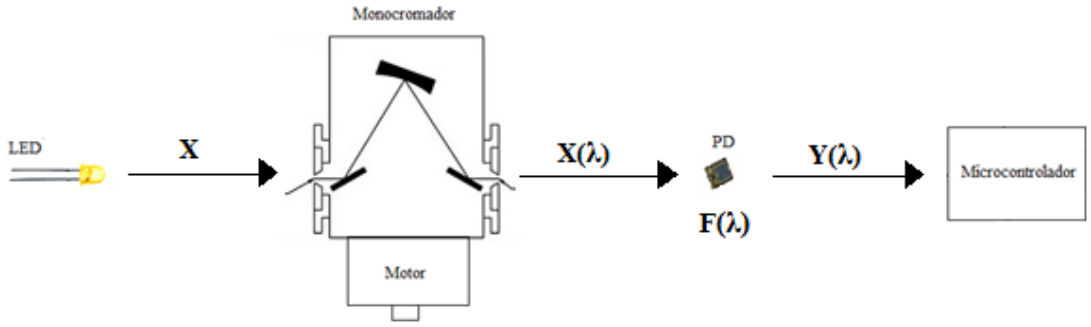


Figura 5.1: Representación esquemática de lo que sucede en el sistema para entender mejor el concepto de calibración.

Pues bien, la señal que deseamos representar es **X**, que es la señal emitida por el LED, y además, deseamos representarla frente a la longitud de onda (por esta razón utilizamos el monocromador), por lo que nuestro interés se centrará en representar **X(λ)**. Para obtener **X(λ)**, deberemos despejar la ecuación anterior.

$$\mathbf{X}(\boldsymbol{\lambda}) = \mathbf{Y}(\boldsymbol{\lambda})/\mathbf{F}(\boldsymbol{\lambda}) \tag{5.2}$$

Entonces, debido a que la señal **Y(λ)** ya la conocemos puesto que es la que nos ofrece el fotodiodo, para conocer la señal **X(λ)** nuestro objetivo se reduce a obtener la respuesta espectral del fotodiodo **F(λ)**.

Por último, lo ideal sería proporcionarle una magnitud, es decir, darle unidades a la intensidad emitida por el LED. Hasta el momento solamente obteníamos valores entre 0 y 1023, los cuales no significaban nada, simplemente eran los valores que nos entregaba el convertidor A/D. Como veremos más adelante, debido a una serie de problemas, no conseguimos llevar a cabo este objetivo, por lo que tuvimos que conformarnos con

normalizar la señal entre 0 y 1 a la hora de representarla, hablaremos de ello en detalle en el apartado 5.4.

A lo largo del presente capítulo procederemos a explicar las tareas llevadas a cabo para conseguir tales objetivos. Como veremos a continuación, tuvimos bastantes problemas y, aunque algunos los llegamos a solucionar ingeniosamente, otros no los conseguimos llevar a buen puerto.

5.2 CUANTIFICAR EL FONDO DE ESCALA DE RUIDO

Como comentábamos anteriormente, el primero de los ajustes que realizamos para una mejor representación de la señal fue (aparte de tomar varias muestras para cada medida) cuantificar el fondo de escala de ruido para cada una de las ganancias. De esta manera, podremos representar las señales a partir del nivel de ruido ya que este no nos interesa.

Para cuantificar los fondos de escala de ruido para las diferentes ganancias seleccionables (x1, x4, x16, x64) realizamos una serie de barridos sin ninguna fuente de luz. Estos barridos fueron realizados para un amplio rango espectral, desde los 430 hasta los 1100 nanómetros. Tras su realización y con ayuda del fichero `Ceros.m` de Matlab (el apéndice A indica dónde encontrarlo), obtuvimos los siguientes resultados:

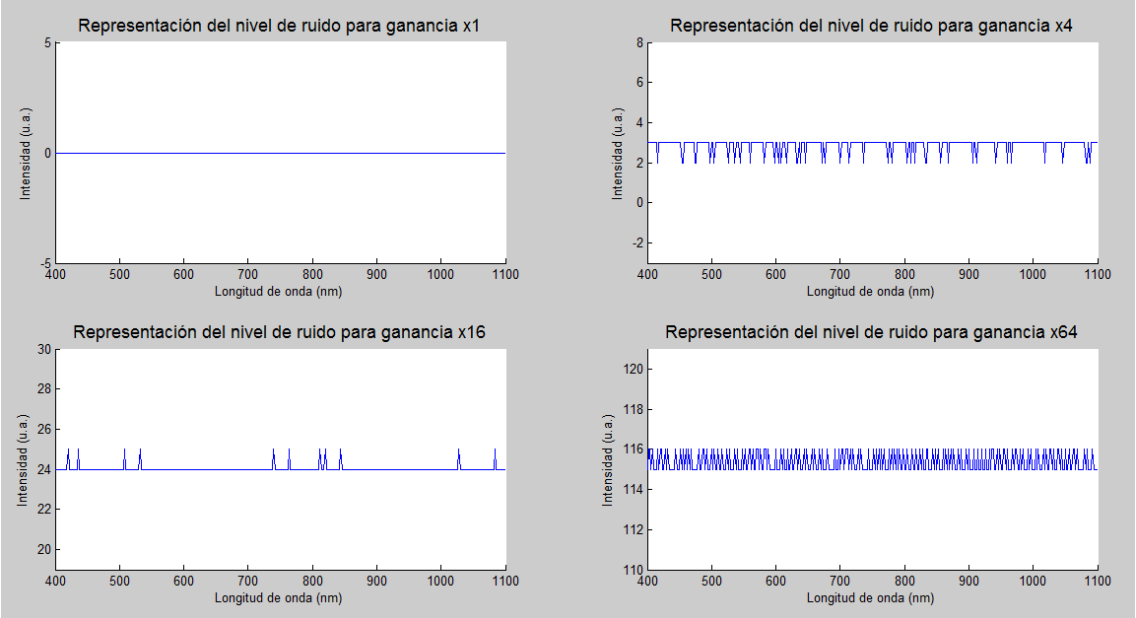


Figura 5.2: Representación del nivel de ruido para las diferentes ganancias seleccionables.

Ganancia	x1	x4	x16	x64
Media del ruido	0	2.87	24.03	115.37
Varianza del ruido	0	0.11	0.03	0.23

Tabla 5.1: Tabla que contiene las medias y varianzas de los fondos de escala de ruido para cada una de las ganancias aplicables a la señal emitida por el fotodiodo.

Como podemos observar en la figura 5.2 y en la tabla 5.1, las varianzas del nivel de ruido son muy pequeñas, esto se debe a que cuando realizamos una medida, en realidad estamos haciendo un promedio de 75 medidas (como comentábamos antes), por lo que

estamos promediando el nivel de ruido. Una vez obtenido los valores medios de los niveles de ruido, los añadimos al código de la aplicación, para ello, lo que hacemos es restarle a la señal **Y(λ)** la media del ruido redondeada a la baja, de esta forma no obtendremos valores negativos y conseguiremos bajar el nivel de *offset* y así tener mayor espacio para la representación de la señal del LED, que es lo que nos interesa. Al restarle el nivel de ruido a la señal **Y(λ)**, obtenemos una señal que llamamos **Ycorr(λ)**. Posteriormente, utilizaremos la ecuación 5.2 para calcular **X(λ)** a partir de la señal **Y(λ)** y, además, calcularemos **Xcorr(λ)** a partir de la señal **Ycorr(λ)**. De esta forma obtenemos las columnas del fichero de datos del que hablábamos en el capítulo 3 (apartado 3.2.4), que utilizaremos posteriormente para la representación de la señal.

5.3 RESPUESTA ESPECTRAL DEL FOTODIODO

Aunque la curva de respuesta espectral del fotodiodo ya fue mostrada en el capítulo 2, volvamos a mostrarla a continuación para evitarnos tener que retroceder hasta aquel apartado para visualizarla.

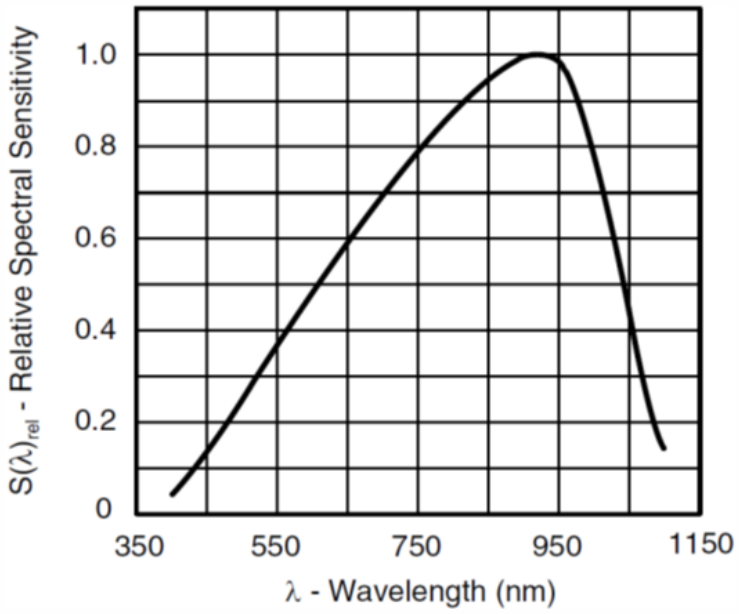


Figura 5.3: Respuesta espectral normalizada frente a longitud de onda del fotodiodo TEMD5010X01 [10]. Podemos observar que el rango espectral va desde los 430 nm hasta los 1100 nm y que su sensibilidad máxima se encuentra en torno a los 900 nm.

Esta curva de respuesta espectral ha sido obtenida de las hojas de especificaciones del fotodiodo [10]. Pues bien, el primer paso para conseguir una función analítica que se ajuste lo mejor posible a **F(λ)** será digitalizar esta respuesta espectral.

5.3.1 Digitalización de la respuesta espectral

Para llevar a cabo la digitalización de la respuesta espectral del fotodiodo utilizamos una aplicación online, la aplicación WebPlotDigitizer [13]. Esta aplicación te permite digitalizar cualquier curva en tres sencillos pasos:

- Primero deberás seleccionar una imagen de la curva a digitalizar, en nuestro caso utilizaremos la imagen mostrada anteriormente.
- Posteriormente tendrás que indicarle a la aplicación cuales son los puntos máximos y mínimos de los ejes X e Y, además, deberás indicarle el valor de cada uno para que esta pueda tomarlos como puntos de referencia.
- Por último, deberás recorrer la curva con el ratón marcando sobre la misma los puntos que consideres necesarios. A mayor número de puntos, mejores resultados, esta es la labor más tediosa, a continuación puede verse un ejemplo de la misma.

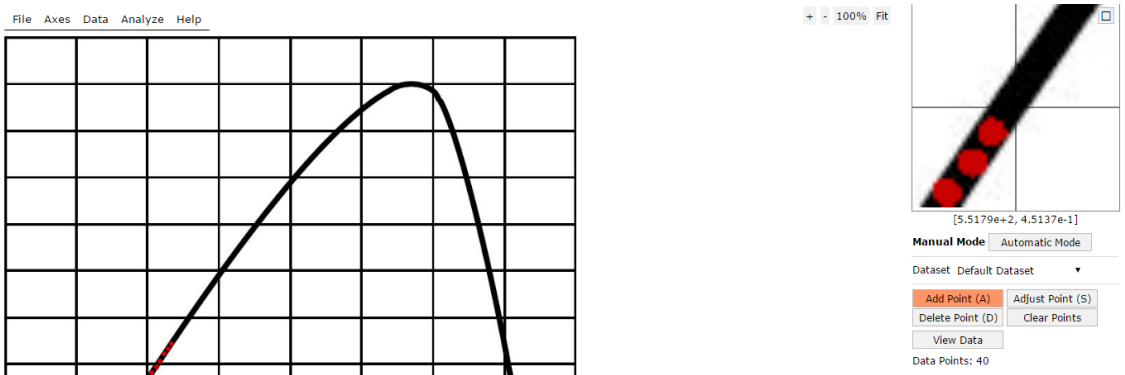


Figura 5.4: Ejemplo de utilización de la herramienta WebPlotDigitizer. Como podemos observar, hemos cargado la imagen de la respuesta espectral del fotodiodo y estamos recorriendo la curva marcando los puntos para digitalizarla.

Una vez finalizada la digitalización, la aplicación te permitirá descargar los resultados obtenidos. Estos resultados básicamente serán las coordenadas de los puntos marcados anteriormente con el ratón.

En nuestro caso, realizamos los pasos descritos anteriormente varias veces para obtener una abundante colección de puntos y obtener así mejores resultados.

5.3.2 Ajuste a función analítica

Ahora que ya tenemos digitalizada nuestra función (la curva de respuesta espectral), pasemos a ajustar la colección de puntos obtenidos a una función analítica. Para ello, durante la realización del presente TFG se utilizó nuevamente una aplicación online, en este caso se trataba de la aplicación ZunZun [14]. Lamentablemente, esta aplicación ya no se encuentra disponible, por lo que vamos a pasar a explicar cómo se realizaría este ajuste mediante otra aplicación, la aplicación LAB Fit [15]. A pesar de mostrar la realización del ajuste con otra aplicación, al final mostraremos los resultados obtenidos mediante ZunZun y continuaremos a partir de ellos, ya que fue la herramienta que se usó durante la realización del TFG.

Pasemos entonces a explicar cómo sería el procedimiento mediante la herramienta LAB Fit. En primer lugar deberemos descargar e instalar esta herramienta, ya que no es de uso online como ocurría con ZunZun. En la figura 5.5 podemos observar el aspecto de la aplicación LAB Fit.

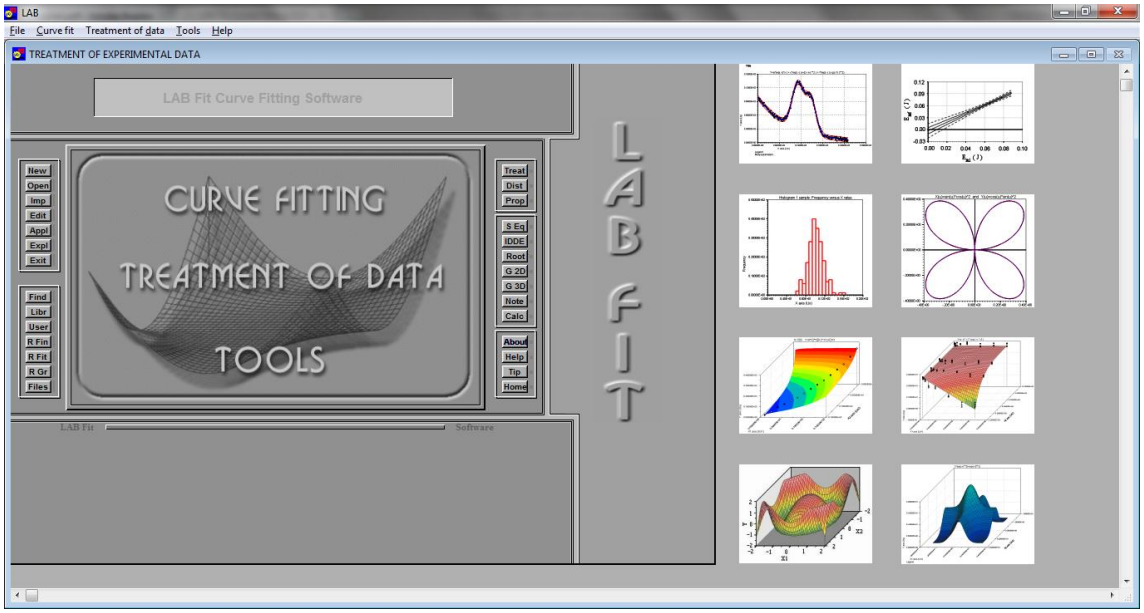


Figura 5.5: Aspecto de la ventana principal de la aplicación LAB Fit. Nuestro área de trabajo se sitúa a la izquierda de la pantalla mientras que, en la parte derecha, la aplicación nos muestra algunos ejemplos de su utilización.

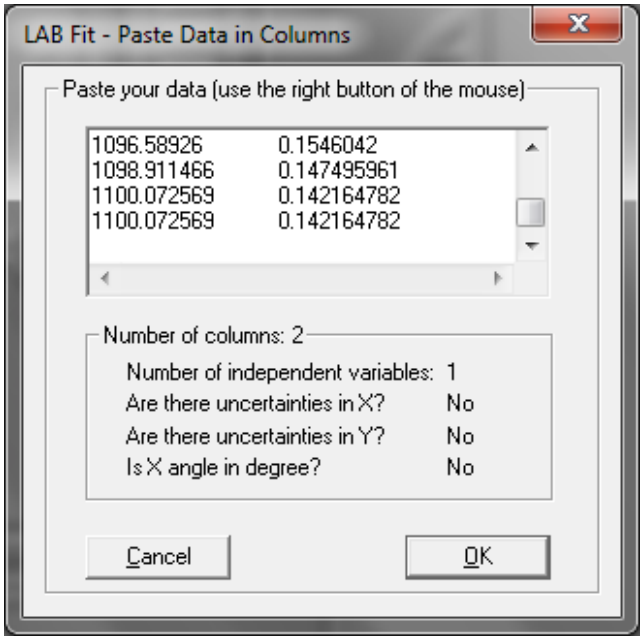


Figura 5.6: Ventana utilizada para la introducción de los datos que se desean ajustar mediante una función analítica.

Para realizar el ajuste, lo primero que deberemos hacer será introducir nuestra colección de datos en la aplicación, para ello, hacemos clic en el botón `New` y pegamos todos los puntos que obtuvimos de la digitalización previa de la respuesta espectral (figura 5.6).

Posteriormente guardaremos nuestros datos en un fichero `.txt` y podremos verlos representados en la ventana principal de la aplicación (figura 5.7).

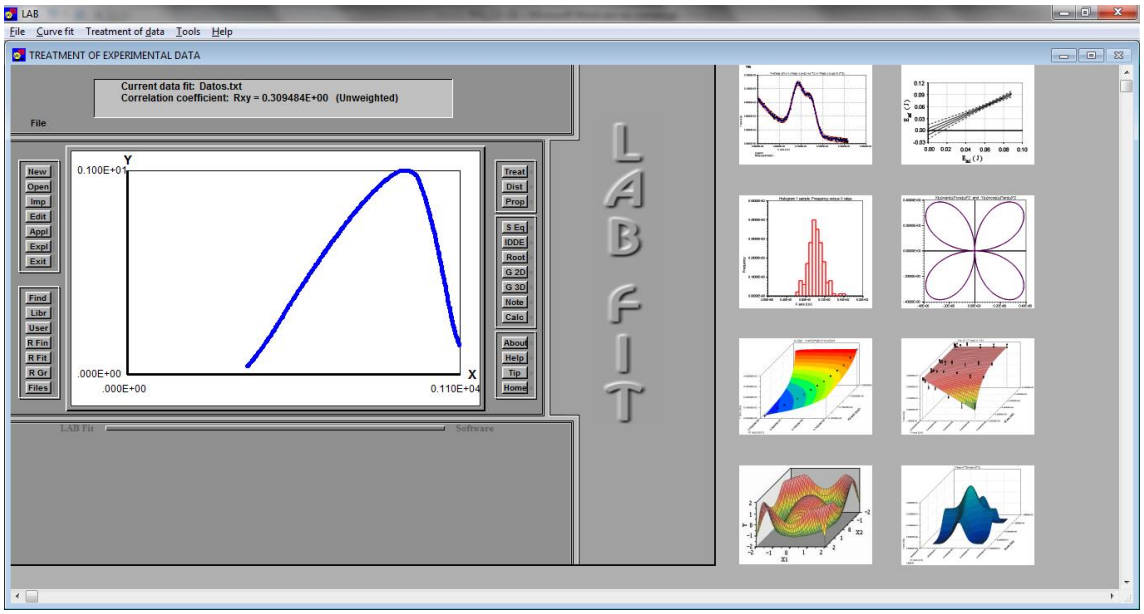


Figura 5.7: Ahora la ventana principal de la aplicación nos muestra la representación de la colección de datos introducida.

Después de haber introducido los datos en la aplicación, podremos pasar al ajuste propiamente dicho. Lo primero será buscar qué tipo de ecuación se ajusta mejor a nuestros datos, para ello hacemos clic en el botón `Find`, y la aplicación se encargará de encontrar el tipo de ecuación más análoga a nuestra curva (figura 5.8),

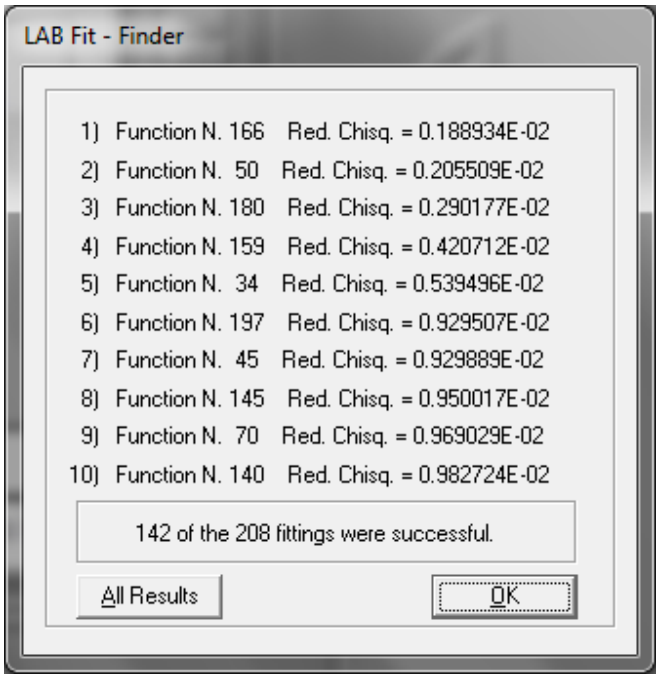


Figura 5.8: Muestra las funciones más afines a la colección de datos introducidos en la aplicación.

Ahora que conocemos la función que mejor se ajusta a nuestra curva, deberemos seleccionarla para realizar dicho ajuste, para ellos hacemos clic en el botón `Libr` y aparecerá la siguiente ventana (figura 5.9), en la cual podremos seleccionar la función deseada.

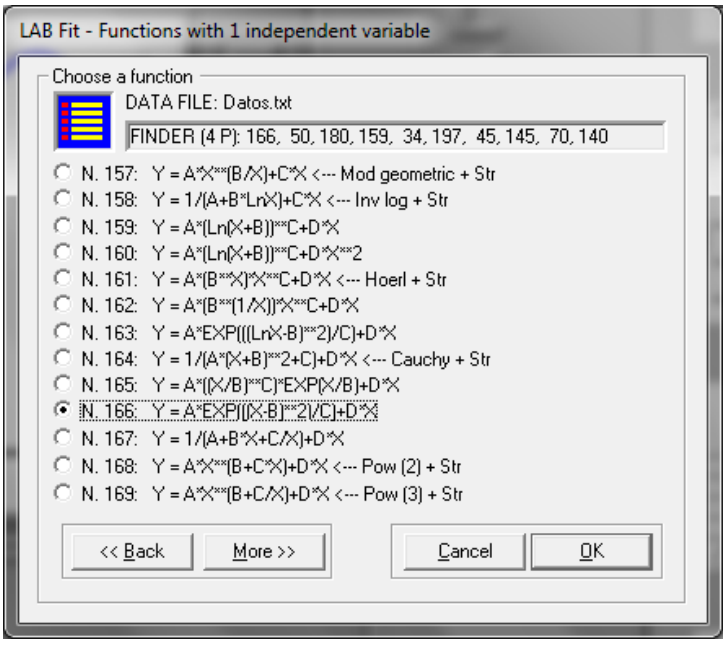


Figura 5.9: Gracias a esta ventana podremos seleccionar la función mediante la cual deseamos ajustar los datos. En nuestro caso elegimos la ecuación número 166 puesto que era la que nos sugería la aplicación.

Después seleccionaremos las condiciones iniciales para cada uno de los cuatro parámetros (en nuestro caso hemos dejado los valores por defecto) y estaremos en disposición de observar los resultados obtenidos (figuras 5.10 y 5.11).

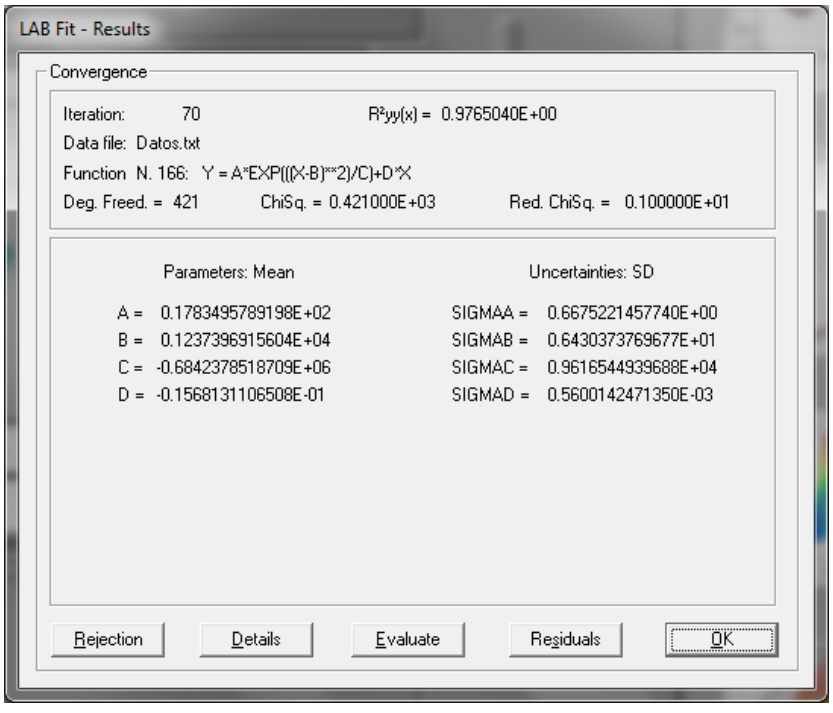


Figura 5.10: Muestra los resultados del ajuste realizado mediante la aplicación LAB Fit.

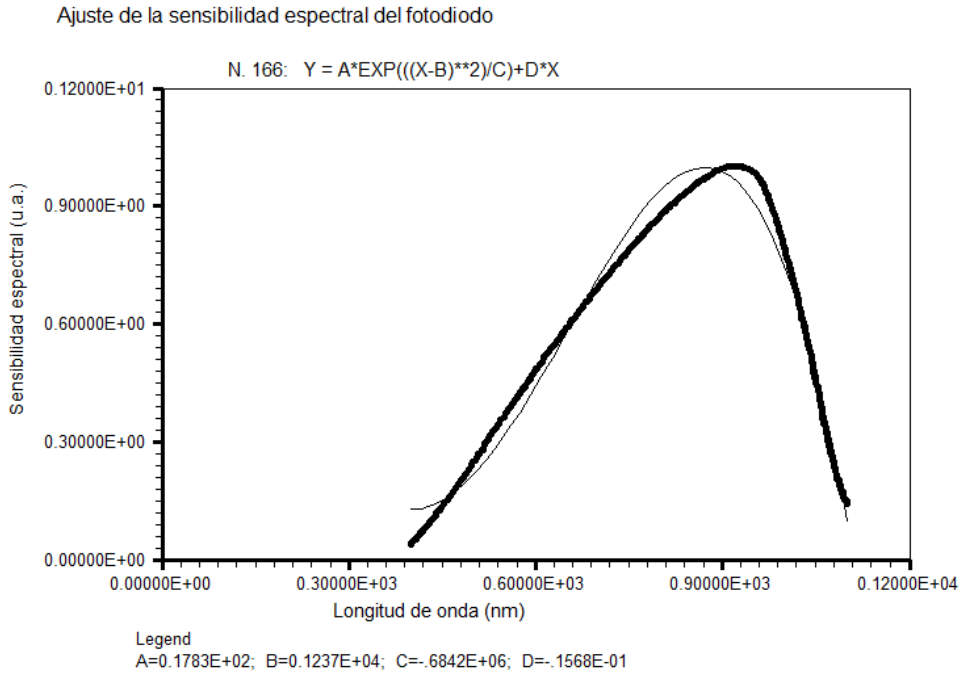


Figura 5.11: Representación de los datos (línea gruesa) junto con la función de ajuste (línea fina) obtenida por la herramienta LAB Fit.

Como podemos observar en las figuras anteriores, los resultados del ajuste no son demasiado buenos, obtuvimos mejor resultados gracias a la aplicación ZunZun de la que hablábamos anteriormente. Quizás sea porque esta aplicación nos ofrecía la posibilidad de utilizar un mayor número de coeficientes para el ajuste, pasemos a mostrar los resultados obtenidos mediante ZunZun.

Gracias a esta aplicación, obtuvimos varias funciones analíticas candidatas a ajustar nuestra curva de respuesta espectral digitalizada. De todas estas funciones, las 3 que mejor se adaptan al comportamiento de la respuesta espectral de nuestro fotodiodo pueden observarse, junto con sus coeficientes y sus errores cuadráticos absolutos, en las figuras 5.12, 5.13 y 5.14.

```
Coefficients


y = a1 / (1.0 + exp(-(x-b1)/c1)) + a2 / (1.0 + exp(-(x-b2)/c2)) + a3 / (1.0 + exp(-(x-b3)/c3))

Fitting target of lowest sum of squared absolute error = 9.0868181987774372E-03

a1 = -2.2396638131085447E+00
b1 =  5.2767116064641004E+02
c1 = -2.4488947209637138E+02
a2 =  1.4355786182558115E+00
b2 =  1.0417397878121276E+03
c2 = -4.2597769222516320E+01
a3 =  3.5953126123811206E-02
b3 =  9.1149076966477628E+02
c3 =  1.4320874233947341E+01
```

Figura 5.12: Función **F2** junto con sus coeficientes y su error cuadrático absoluto, todo ello proporcionado por la aplicación online ZunZun.

```
Coefficients

y = a * (d * (1/(1+((x-b)/c)^f )) + (1-d) * exp(-0.5 * ((x-b)/c)^g )) + Offset

Fitting target of lowest sum of squared absolute error = 9.2847761996496941E-03

a =  3.5860707392432687E+00
b =  6.7281389379524569E+01
c =  9.7768146621145388E+02
d =  4.0199829918532520E-01
f =  2.3478298894790441E+01
g = -1.5903507664965841E+00
Offset = -1.5368507271534160E+00
```

Figura 5.13: Función **F3** junto con sus coeficientes y su error cuadrático absoluto, todo ello proporcionado por la aplicación online ZunZun.

```
Coefficients

y = a*exp(-b*x) + c*exp(-(x-d)^2  / f^2 ) + g*exp(-(x-h)^2  / i^2 ) + j*exp(-(x-k)^2  / l^2 ) + Offs
et

Fitting target of lowest sum of squared absolute error = 1.4856601780781816E-02

a = -1.7903641692617505E+03
b =  1.0096783384310190E-01
c =  5.1302696091214595E-01
d =  9.7748196775399106E+02
f = -1.0904347713554418E+02
g = -2.5654609993016003E+01
h =  7.1509258456888597E+02
i =  2.1519599787435874E+02
j =  2.6570867548538679E+01
k =  7.1690236106604561E+02
l =  2.1818939867743504E+02
Offset = -1.8890389513785263E-01
```

Figura 5.14: Función **F5** junto con sus coeficientes y su error cuadrático absoluto, todo ello proporcionado por la aplicación online ZunZun.

Para quedarnos con la mejor candidata, procedimos a representar las 3 funciones junto con la colección de puntos obtenida de la digitalización, esta representación puede verse en la figura 5.15.

Como podemos observar, la función que mejor sigue el trazado de la colección de puntos es, a nuestro juicio, la función **F2**. Además de ser la función que mejor sigue el trazado de puntos, sobre todo en la zona del máximo que es la zona más delicada, también es la función de menor error cuadrático absoluto, como puede comprobarse en las figuras anteriores. Es por esto que, de aquí en adelante, entenderemos que esta función es la curva de respuesta espectral del fotodiodo y la utilizaremos para calibrar las modificaciones realizadas por este sobre la señal procedente del LED.

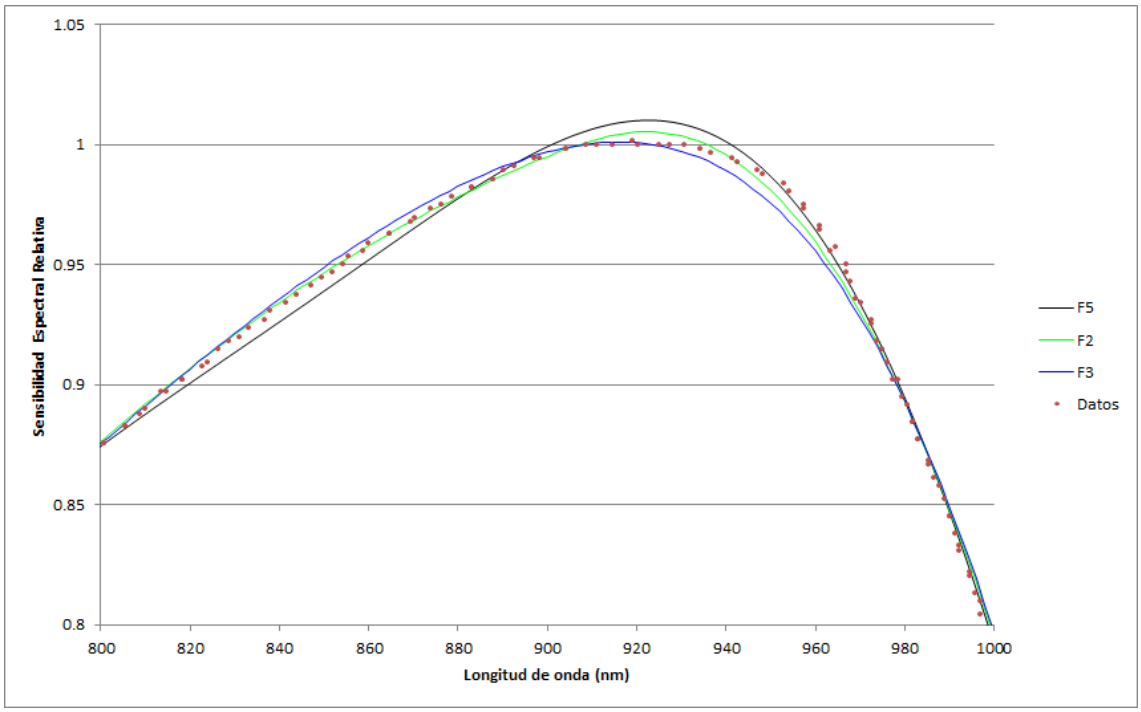


Figura 5.15: Representación de las funciones analíticas obtenidas mediante la aplicación ZunZun junto con la colección de puntos obtenidos de digitalizar la curva de respuesta espectral del fotodiodo.

5.3.3 Incorporación al código

Para incorporar la función **F2** al código, podemos aprovechar el código fuente que nos proporciona la aplicación ZunZun. Esta aplicación nos proporciona el código necesario para la obtención de la función en varios lenguajes, entre ellos: C++, C, Java, Matlab, etc.

En nuestro caso, no utilizamos el código proporcionado por la aplicación, sino que la implementamos nosotros mismos gracias a la ecuación y los coeficientes que esta nos proporcionó, ver figura 5.12.

Una vez obtenida la función, estaríamos en condiciones de implementar la ecuación (5.2) para obtener finalmente las señales calibradas **X(λ)** y **Xcorr(λ)**, que son las señales que nos interesan. Hasta este punto, habríamos conseguido tanto eliminar el ruido de la señal obtenida del fotodiodo como calibrar (corregir) la deformación producida por el mismo.

5.3.4 Ejemplo de calibración

Como comentábamos anteriormente, hasta el momento hemos conseguido eliminar tanto el nivel de ruido como calibrar la deformación producida por la respuesta espectral del fotodiodo. Pues bien, lo lógico ahora sería mostrar un ejemplo, para ello, volvamos al barrido de ejemplo que utilizamos para explicar el funcionamiento del sistema a lo largo del capítulo 3. Recordemos que teníamos guardados los datos del barrido en un

fichero llamado `Prueba.txt`, y que este contiene tanto el espectro de emisión del LED calibrando la deformación producida por el fotodiodo como el espectro de emisión sin calibrar. Entonces, gracias al contenido del fichero, estamos en condiciones de comparar ambos espectros, calibrado y sin calibrar. Para realizar esta comparación, hemos utilizado la herramienta Matlab y el fichero `CorreccionF2.m`, nuevamente el apéndice A explica dónde encontrarlo. Mediante este *script* cargamos y representamos los datos del fichero `Prueba.txt`. A continuación mostramos las representaciones obtenidas para el espectro de emisión del LED sin el nivel de ruido y sin calibrar la deformación producida por el fotodiodo (**Ycorr(λ)**, en color azul), y para dicho espectro incluyendo la calibración de la deformación del fotodiodo (**Xcorr(λ)**, en color rojo).

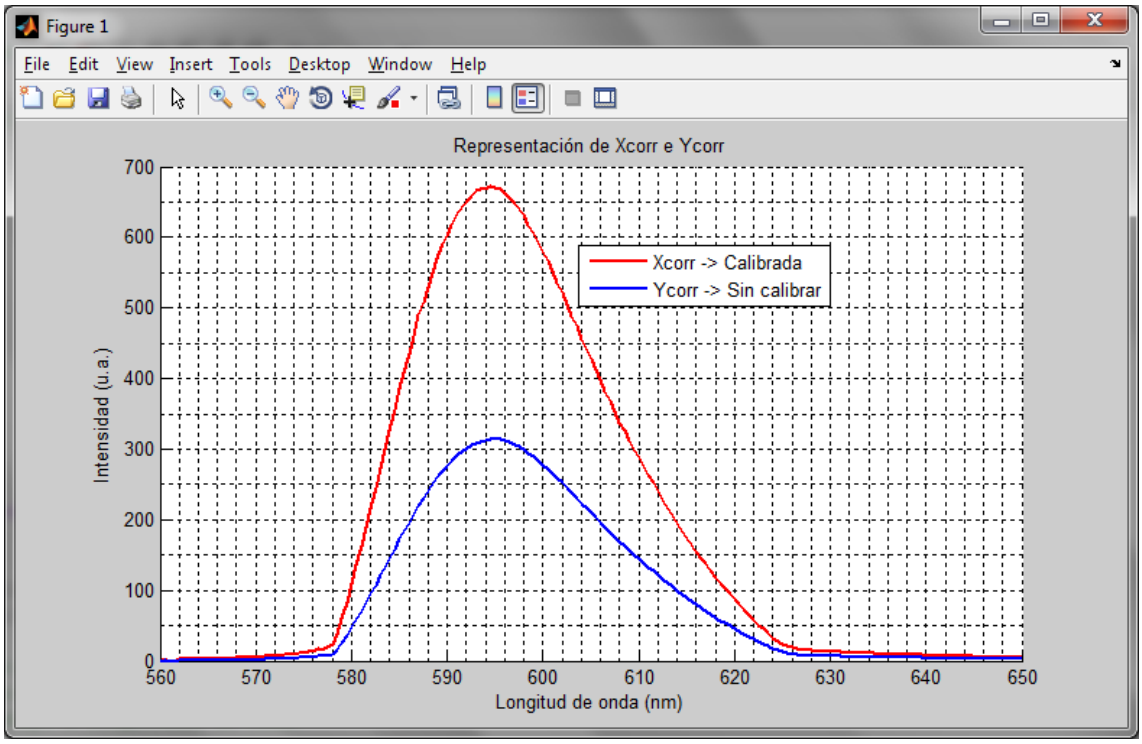


Figura 5.16: Ejemplo para un barrido del cual hemos representado las columnas **Ycorr(λ)** y **Xcorr(λ)** del fichero de datos. La columna **Ycorr(λ)** contiene el espectro de emisión del LED menos el nivel de ruido sin calibrar la deformación producida por el fotodiodo, en cambio, la columna **Xcorr(λ)** sí que incluye la calibración de esa deformación.

Para poder comparar correctamente los espectros de emisión calibrado y sin calibrar del LED, debemos tener presente la respuesta espectral del fotodiodo mostrada en la figura 5.3. Si nos fijamos en dicha respuesta espectral, observaremos que, en el rango de emisión del LED (575 – 625 nm), se encuentra en continuo crecimiento. Entonces, cabría esperar que el espectro de emisión del LED sin calibrar (**Ycorr**) apareciese deformado de tal manera que, pareciese que emite con mayor intensidad a medida que se incrementa la frecuencia. Si observamos la figura 5.16, parece que, efectivamente, se cumple esta premisa.

Después de calibrar el espectro de emisión (**Xcorr**), este debería entonces poseer mayor intensidad a longitudes de onda menores y menor a longitudes de onda mayores, comparándolo con el espectro sin calibrar (**Ycorr**). Si nos fijamos nuevamente en la

figura 5.16, observamos que también se cumple dicha premisa. De hecho, al calibrar la señal, parece que tiende más a la forma gaussiana (a la forma normal), no llega a recuperar dicha forma porque el espectro de emisión del LED no es perfectamente gaussiano.

Además, el espectro de emisión calibrado muestra una intensidad mayor debido a que la respuesta espectral del fotodiodo está normalizada entre 0 y 1. Luego, para poder comparar mejor ambos espectros de emisión, pasemos a mostrarlos normalizados a sus máximos en amplitud, ver figura 5.17.

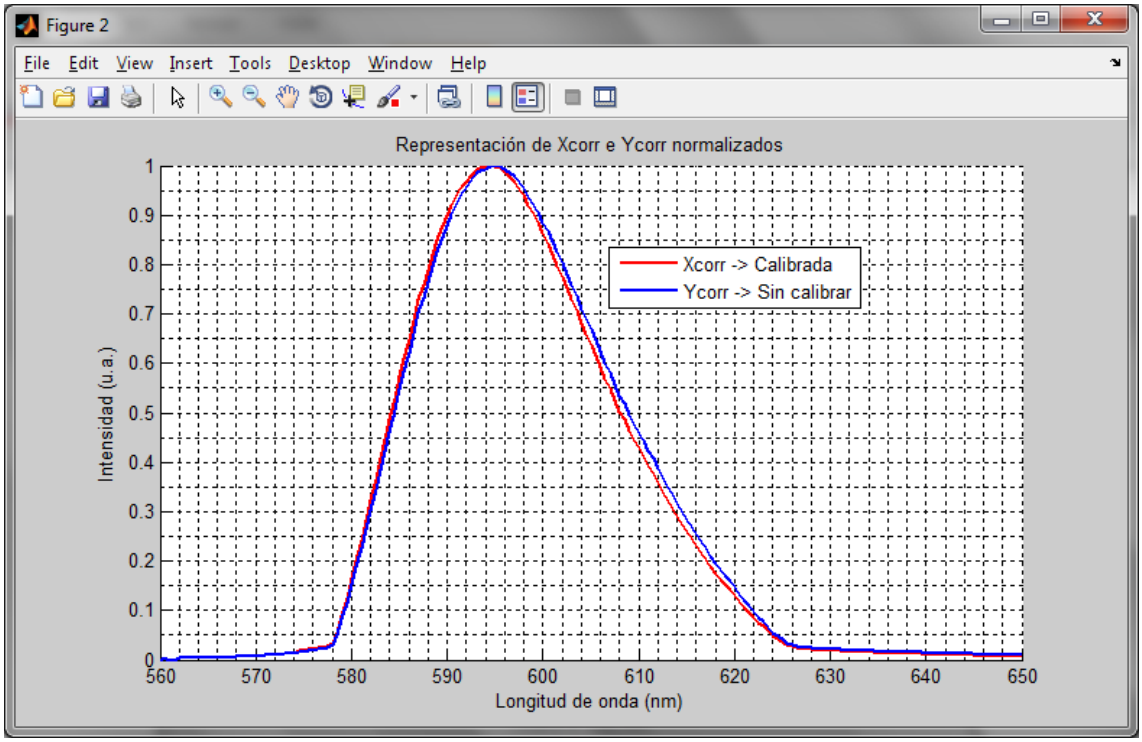


Figura 5.17: Misma representación que la anterior, pero esta vez normalizamos las columnas **Ycorr(λ)** y **Xcorr(λ)** del fichero de datos a su máximo. La columna **Ycorr(λ)** contiene el espectro de emisión del LED menos el nivel de ruido sin calibrar la deformación producida por el fotodiodo, en cambio, la columna **Xcorr(λ)** sí que incluye la calibración de esa deformación.

Al observar las señales al mismo nivel de intensidad (figura 5.17) podemos apreciar que, efectivamente se cumple que la señal sin calibrar (**Ycorr**) parece emitir con mayor intensidad a medida que aumenta la frecuencia, esto demuestra el efecto de la respuesta espectral del fotodiodo. Además, observando dicha figura también observamos que los cambios producidos por la calibración son mínimos, luego parece ser que la deformación producida por el fotodiodo no es muy severa.

A continuación, representamos la diferencia entre ambos espectros, con y sin calibración (**Xcorr - Ycorr**), en la figura 5.18.

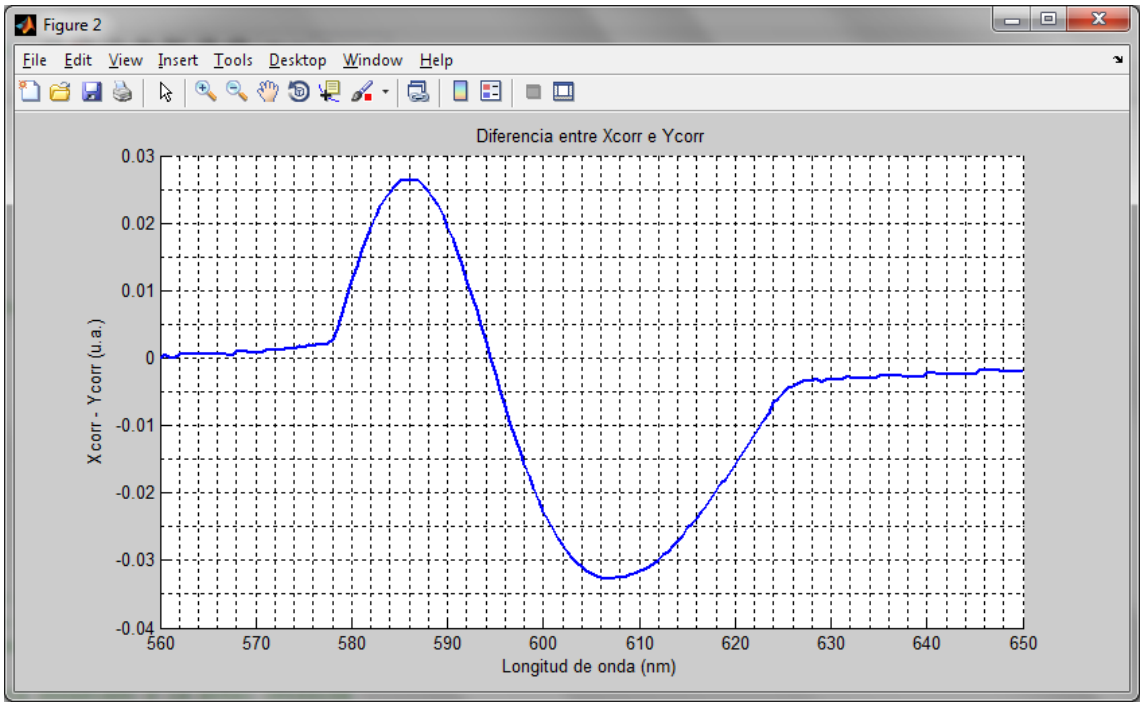


Figura 5.18: Representación de la diferencia entre las 2 señales mostradas en la figura 5.17. Como era de esperar, la señal **Xcorr(λ)** es mayor que **Ycorr(λ)** a longitudes de onda menores y menor a longitudes de onda mayores.

Podemos observar que la señal calibrada (**Xcorr**) está por encima a longitudes de onda más bajas y pasa a estar por debajo a longitudes de onda mayores, como era de esperar ya que comentábamos que el fotodiodo deformaba la señal haciendo que pareciese emitir con mayor intensidad a longitudes de onda mayores. En el capítulo 6 mostraremos algunos ejemplos de espectros de emisión obtenidos para diferentes LEDs mediante nuestro sistema, compararemos dichos espectros con los mostrados por las hojas de especificaciones de dichos LEDs, entonces podremos ver si la calibración de la forma de la señal ha sido lo suficientemente buena.

Continuemos con nuestra calibración, ahora le toca el turno a la amplitud de la señal, lo ideal sería conseguir darle unidades a dicha amplitud, para ello pasemos al siguiente apartado.

5.4 AMPLITUD DE LA SEÑAL

Una vez calibrada la forma de la señal obtenida por el fotodiodo, procederemos a calibrar la amplitud de la señal, es decir, nuestro objetivo será dar magnitud (unidades) a la intensidad emitida por el LED y captada por nuestro fotodiodo. Para ello, vamos a utilizar dos lámparas, una incandescente y otra halógena, procederemos a obtener los espectros de radiación de ambas lámparas (gracias a nuestro sistema), los cuales compararemos con el espectro de radiación del cuerpo negro. Es decir, vamos a comparar los filamentos de las lámparas con un emisor ideal, con un cuerpo negro. Los espectros de radiación obtenidos de las lámparas y el del cuerpo negro podrían diferir en una constante, pero deberían tener la misma forma. Pues bien, nuestro objetivo será

calcular esa constante (la llamaremos ***M***) para corregir la amplitud y proporcionar así unidades a la señal obtenida.

$$\boldsymbol{PD} \cdot \boldsymbol{M} = \boldsymbol{W} \tag{5.3}$$

Donde ***PD*** es la señal proporcionada por el fotodiodo al medir el espectro de emisión de cualquiera de las lámparas y ***W*** es el espectro de radiación del cuerpo negro. Como comentábamos anteriormente, ***M*** es la constante por la cual ambos espectros pueden diferir. Antes de pasar a comparar el espectro de radiación del cuerpo negro con el de las lámparas, expliquemos qué es el cuerpo negro y cuál es su espectro de radiación.

5.4.1 Espectro de radiación del cuerpo negro

La radiación producida por cuerpos opacos fue ampliamente estudiada a finales del siglo XIX, dando como resultado las siguientes leyes empíricas. Primeramente se descubrió que la potencia óptica radiada (vía radiación electromagnética) por un cuerpo era proporcional a su área y a la cuarta potencia de su temperatura absoluta [16 y 17].

$$\mathrm{W} = \varepsilon\sigma\mathrm{A}\mathrm{T}^4 \tag{5.4}$$

Donde W es potencia radiada en vatios, ε es la emisividad (la cual toma valores entre 0 y 1, siendo 1 para un emisor ideal, el cual se conoce como cuerpo negro), σ es la constante de Stefan-Boltzman, A es el área de la superficie del cuerpo en m^2 y T es la temperatura del cuerpo en grados Kelvin.

En segundo lugar, se observó que la distribución espectral de la energía emitida a una temperatura dada tenía un máximo definido, y que ese máximo se desplazaba a longitudes de onda menores a medida que la temperatura se incrementaba, como ilustra la figura 5.19. Este desplazamiento es dado por la Ley del Desplazamiento de Wien.

$$\lambda_m T = constante \tag{5.5}$$

Donde λ_m es la longitud de onda a la cual la potencia radiada es máxima para una temperatura T dada.

El espectro de radiación del cuerpo negro se describe por la Ley de Radiación de Planck [16, 17 y 18].

$$W(\lambda, T) = \frac{2\pi hc^2}{\lambda^5} \frac{1}{e^{hc/\lambda kT} - 1} \tag{5.6}$$

Donde T es la temperatura (en grados Kelvin), h es la constante de Planck, c es la velocidad de la luz y k es la constante de Boltzman. $W(\lambda,T)$ es conocida como la excitación espectral, la cual es un flujo de potencia por unidad de área y longitud de onda, por lo que sus unidades en el SI son W/m^3. La figura 5.19 muestra la variación de $W(\lambda,T)$ con la longitud de onda a diferentes temperaturas.

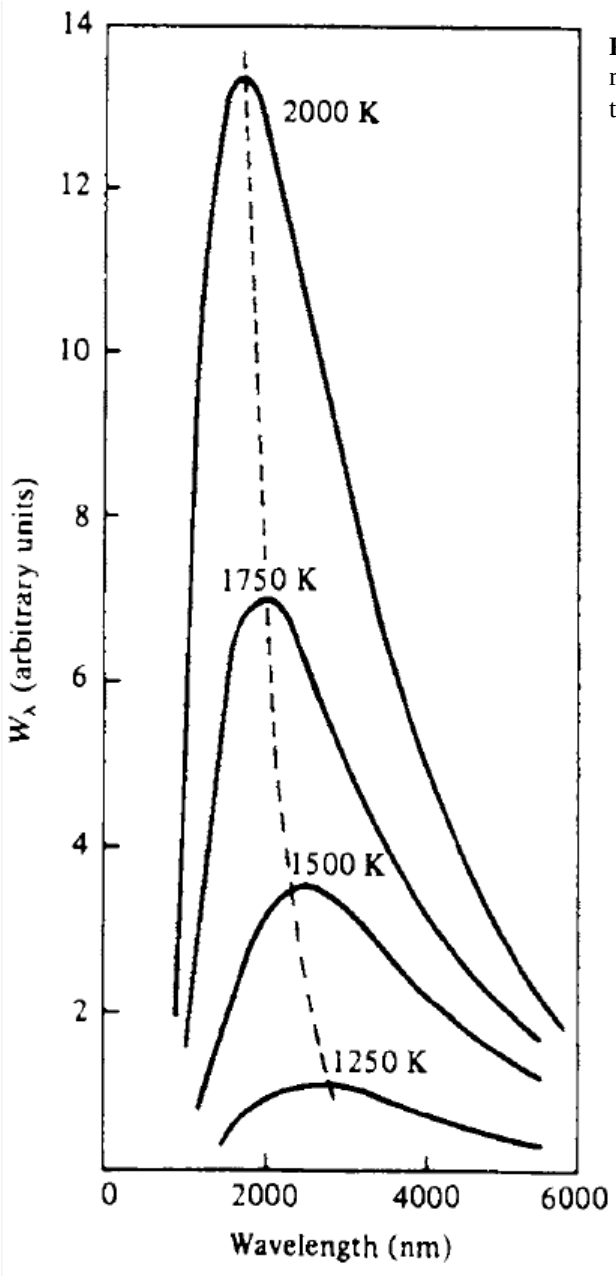


Figura 5.19: Distribución espectral de la radiación de un cuerpo negro a varias temperaturas [17].

Ahora que ya conocemos qué es el cuerpo negro y cuál es su espectro de radiación *W(λ,T)*, ya estamos en condiciones de compararlo con los espectros de emisión de las lámparas.

5.4.2 Comparación de los espectros

Como ya comentamos anteriormente, la tarea más potente y completa que el usuario es capaz de realizar mediante el sistema es un barrido en longitud de onda, es decir, obtener la representación de la intensidad emitida por el LED frente a la longitud de onda, lo que es lo mismo que el espectro de emisión del LED. En este caso, no utilizaremos ningún LED, sino que utilizaremos las lámparas halógena e incandescente, como comentamos en apartados anteriores.

Se realizaron una serie de barridos sucesivos aumentando con cada barrido el voltaje y la corriente a los que era sometida la lámpara, aumentando también a su vez la intensidad de luz emitida, así como la temperatura de la misma, por la que después nos interesaremos. Posteriormente, calibramos cada uno de estos barridos, con la ayuda de la ecuación (5.2) y de nuestra función analítica **F2,** y los comparamos con el espectro de radiación del cuerpo negro, **W(λ,T)**.

Previamente se comentó que las gráficas obtenidas de los barridos tras su calibración y la gráfica de radiación del cuerpo negro podrían diferir en una constante, pero deberían tener la misma forma. Es decir, los espectros de radiación de las lámparas y del cuerpo negro deben tener la misma forma. Para comparar ambos espectros, deberemos conocer los dos parámetros de los que depende el espectro de radiación del cuerpo negro:

- La longitud de onda (**λ**), la cual conoceremos gracias al monocromador y la utilizaremos como variable independiente a la hora de representar los espectros de radiación.
- La temperatura (**T**) a la que se encuentra el filamento de la lámpara en cada barrido, la cual tendremos que calcular. No es posible medir la temperatura del filamento mientras se está realizando el barrido, por ello, deberemos ingeniárnoslas para calcularla.

5.4.2.1 Cálculo de la relación R(T)/R(273)

Partiremos de que la temperatura del filamento es constante para cada barrido y, como hemos comentado anteriormente, el valor de la misma no es posible medirlo cuando se está realizando el barrido. Pues bien, si nos fijamos en las tablas de tungsteno incluidas en [19] (mostradas también a continuación, tabla 5.2), podemos apreciar que si conociésemos la relación entre la impedancia ofrecida por la lámpara a la temperatura **T** y la impedancia a 273 K (**R(T)/R(273)**), podríamos conocer la temperatura **T**, que es la temperatura a la cual se realizó el barrido. Aclarar que, aunque utilizamos dos lámparas diferentes, una halógena y otra incandescente, podemos usar las mismas tablas para ambas lámparas debido a que el filamento de ambas es de tungsteno (la diferencia entre ellas está en el gas que lo rodea).

Entonces, necesitaríamos conocer la relación **R(T)/R(273)** para conocer la temperatura **T**. Pues bien, **R(T)** lo podemos conocer midiendo el voltaje y la corriente en la lámpara cuando se realiza el barrido (**R(T) = V(T)/I(T)**), el problema está en que no podemos calcular **R(273)**. Para solventar el problema vamos a incluir la impedancia a la temperatura ambiente **R(300)**, entonces tendríamos que se cumple la siguiente ecuación:

$$\frac{\boldsymbol{R(T)}}{\boldsymbol{R(273)}} = \frac{\boldsymbol{R(T)}}{\boldsymbol{R(300)}} \cdot \frac{\boldsymbol{R(300)}}{\boldsymbol{R(273)}} \qquad (5.7)$$

Z = R(T)/R(273)	T	Z = R(T)/R(273)	T
1	273.16	5.6	1103
1.2	314	5.8	1136
1.4	355	6	1168
1.6	394	6.2	1200
1.8	434	6.4	1232
2	472	6.6	1264
2.2	511	6.8	1296
2.4	548	7	1328
2.6	586	7.2	1360
2.8	623	7.4	1391
3	659	7.6	1422
3.2	695	7.8	1454
3.4	731	8	1485
3.6	766	8.2	1516
3.8	801	8.4	1547
4	835	8.6	1578
4.2	870	8.8	1609
4.4	904	9.0	1640
4.6	938	9.2	1671
4.8	971	9.4	1702
5	1004	9.6	1733
5.2	1037	9.8	1763
5.4	1070	10	1794

Tabla 5.2: Tabla de Tungsteno para el rango de temperaturas 273 – 1800 K. Muestra la relación entre la impedancia ofrecida por el filamento a una temperatura **T** y la impedancia ofrecida a 273 K, de forma que si conocemos esa relación podemos conocer la temperatura **T** y viceversa [19].

La relación **R(300)/R(273)** la podemos calcular fijándonos en las tablas de tungsteno y realizando la siguiente operación:

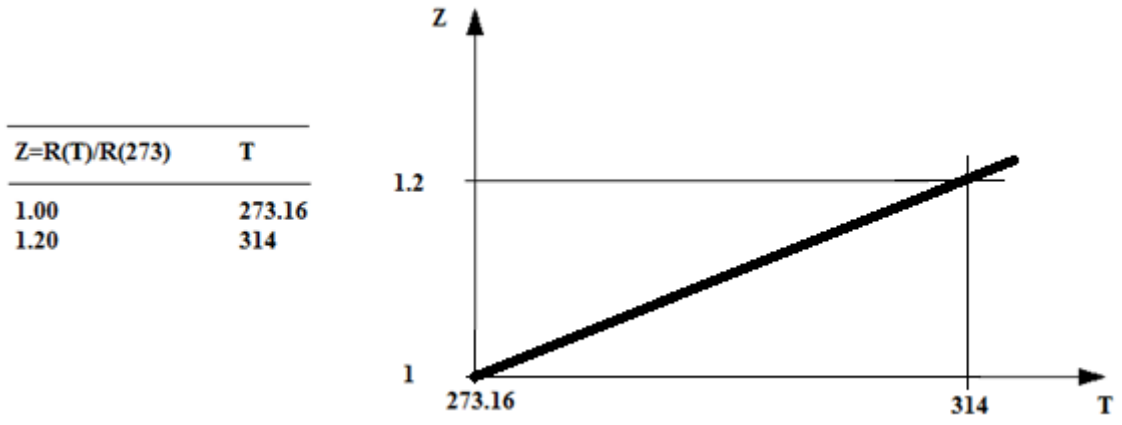


Figura 5.20: Representación de la tabla de tungsteno para facilitar el cálculo de la relación **R(300)/R(273)**.

$$\frac{\boldsymbol{R(300)}}{\boldsymbol{R(273)}} = pte \cdot (300 - 273.16) + 1 =$$

$$= \frac{(1.2-1)}{(314-273.16)} \cdot (300 - 273.16) + 1 = 1.1314 \qquad (5.8)$$

Llegados a este punto, conocemos **R(T)** y la relación **R(300)/R(273)**, entonces, lo único que nos queda por conocer es **R(300)**. Para conocer la resistencia ofrecida por la lámpara a temperatura ambiente utilizamos diversos métodos:

- Método 1: Multímetro.

El primer método fue el más intuitivo, medimos la resistencia de la lámpara con un multímetro. Esta medida no fue muy buena ya que no nos daba un resultado fijo, variaba mucho. Fue por esto que pasamos al segundo método.

- Método 2: Circuito RC.

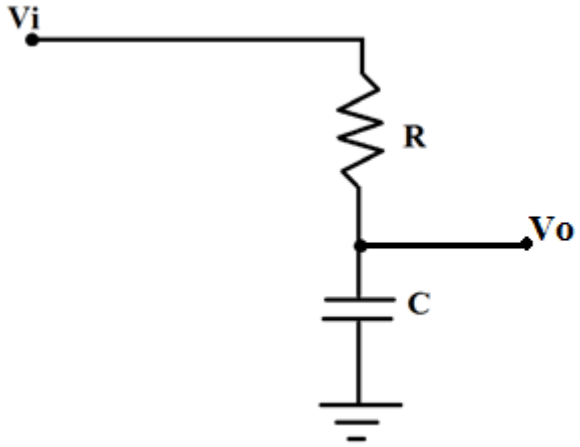


Figura 5.21: Esquema del circuito RC utilizado para medir la resistencia R de la lámpara.

Decidimos montar un circuito RC (figura 5.21) de manera que, cargando y descargando el condensador podríamos calcular la resistencia, ya que, la carga es proporcional a $1 - e^{\frac{-t}{RC}}$, la descarga es proporcional a $e^{\frac{-t}{RC}}$ y el valor del condensador era conocido. Para capturar la señal a la salida (V_o) de este circuito utilizamos un osciloscopio, pero el osciloscopio no era capaz de obtener una imagen estable de la señal por ser un condensador muy grande y una frecuencia muy baja. Para conseguir capturar una imagen de un periodo de carga o de descarga completo, recurrimos a grabar unos videos de la pantalla del osciloscopio, de los cuales sacamos las imágenes que buscábamos.

Posteriormente, digitalizamos las imágenes obtenidas con ayuda de la herramienta online WebPlotDigitizer (que ya utilizamos para digitalizar la respuesta del fotodiodo de las hojas de especificaciones) y representamos el logaritmo de estas medidas (el logaritmo de una función exponencial nos da una función lineal). Después aproximamos la gráfica por una función lineal cuya pendiente coincide con 1/RC, el valor del condensador era conocido por lo que obtuvimos el valor de la resistencia que era nuestro objetivo. Los resultados tampoco fueron buenos, obtuvimos un valor de 66 Ω para la lámpara halógena, lo cual era demasiado elevado, por lo que buscamos otra

alternativa para encontrar el valor de las resistencias de ambas lámparas, eso nos llevó al tercer método.

- Método 3: Circuito con R en serie.

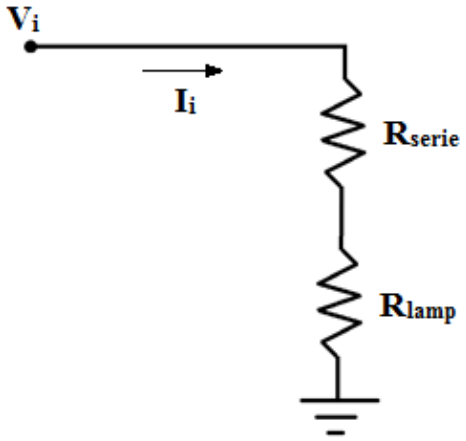


Figura 5.22: Esquema del circuito definitivo utilizado para medir la resistencia de la lámpara, se trata de un circuito con una resistencia en serie.

La tercera y última alternativa fue poner la lámpara en serie con resistencias de diferentes valores y aplicarles un voltaje. Gracias al voltaje medido en bornes de la resistencia y al valor de la misma, obtenemos la corriente que es la misma que pasa por la lámpara. Obtenida la corriente y gracias a la tensión en bornes de la lámpara, podemos calcular la resistencia de la lámpara. Los resultados obtenidos (tablas 5.3 y 5.4) fueron mejores que los anteriores, tomaremos la media de estos resultados como valor de la resistencia de la lámpara a temperatura ambiente (R(300), mostramos los resultados obtenidos en la tabla 5.5) y, entonces, ya estaremos en condiciones de obtener la temperatura del filamento para cada barrido.

Resistencia en serie (Ω)	Voltaje caído en R (V)	Corriente (mA)	Voltaje caído en lámpara (mV)	Resistencia de la lámpara (Ω)
56.5	0.986	17.4513274	9.3	0.53291075
68.7	0.988	14.3813683	7.6	0.52846154
82.5	0.994	12.0484848	6.3	0.52288732
99.4	0.995	10.0100604	5.3	0.52946734
123.9	0.996	8.03874092	4.2	0.52246988
148.9	0.998	6.70248489	3.5	0.52219439
182	0.999	5.48901099	2.9	0.52832833
220.3	0.999	4.53472537	2.3	0.5071972
261.5	1	3.82409178	2	0.523
328	1.001	3.05182927	1.6	0.52427572
338.8	1.001	2.95454545	1.5	0.50769231
396	1.002	2.53030303	1.3	0.51377246
461	1.002	2.17353579	1	0.46007984
557	1	1.79533214	0.9	0.5013
984	1.003	1.01930894	0.5	0.49052841

Tabla 5.3: Tabla con los resultados para la resistencia de la lámpara halógena a temperatura ambiente (**R(300)**) obtenidos para un circuito en serie con resistencias de diferentes valores.

Resistencia en serie (Ω)	Voltaje caído en R (V)	Corriente (mA)	Voltaje caído en lámpara (mV)	Resistencia de la lámpara (Ω)
680	0.957	1.40735294	24.4	17.3375131
811	0.953	1.17509248	20.3	17.2752361
985	0.957	0.9715736	16.8	17.2915361
1183	0.96	0.8114962	13.9	17.1288542
1479	0.962	0.65043949	11.1	17.0653846
1779	0.964	0.54187746	9.3	17.1625519
2170	0.966	0.44516129	7.6	17.0724638
2650	0.969	0.36566038	6.3	17.2291022
3240	0.969	0.29907407	5.2	17.3869969
3960	0.971	0.24520202	4.3	17.5365602
4650	0.972	0.20903226	3.6	17.2222222
5460	0.974	0.17838828	3.1	17.3778234
6800	0.976	0.14352941	2.5	17.4180328

Tabla 5.4: Tabla con los resultados para la resistencia de la lámpara incandescente a temperatura ambiente (**R(300)**) obtenidos para un circuito en serie con resistencias de diferentes valores.

Lámpara Halógena		Lámpara Incandescente	
Resistencia media	0.514	Resistencia media	17.26
Desviación típica	0.019	Desviación típica	0.14

Tabla 5.5: Tabla que contiene la media de la resistencia obtenida para ambas lámparas junto con su desviación típica. Estos valores medios serán los que utilicemos como las resistencias de las lámparas a temperatura ambiente (**R(300)**) para calcular la temperatura a la que se encontraban dichas lámparas durante la realización de los diferentes barridos.

5.4.2.2 Cálculo de la temperatura del filamento

Una vez tenemos calculado **R(300)** ya conocemos todos los datos de la ecuación (5.7) necesarios para calcular **R(T)/R(273)**. Pues bien, con este dato y gracias a las tablas de tungsteno [19], por fin podemos obtener la temperatura del filamento. Comentar que cada barrido fue realizado a una temperatura diferente por lo que tendremos que calcular esta para cada uno de ellos, este cálculo lo realizamos de la siguiente forma:

- Primeramente calculamos **R(T)**, este dato puede ser calculado mediante la ley de Ohm gracias a que cuando se realizaron los barridos medimos con un multímetro la corriente y el voltaje a los que era sometida la lámpara, ver tablas 5.6 y 5.7.
- En segundo lugar, deberemos tener en cuenta que **R(300)** y la relación **R(300)/R(273)** son iguales para todos los barridos ya que las temperaturas son las mismas.
- Con los datos mencionados en los dos puntos anteriores podemos calcular la relación **R(T)/R(273)** para cada barrido gracias a la ecuación (5.7). Estos resultados pueden apreciarse en las tablas 5.6 y 5.7.

Llegados a este punto estaríamos en disposición de utilizar las tablas de tungsteno [19] para obtener la temperatura. Para obtener este dato realizamos los siguientes pasos:

- Primero hemos representado en una hoja Excel los datos proporcionados por las tablas de tungsteno (o tabla 5.2) [19], hemos representado la temperatura **T** frente a la relación **R(T)/R(273)**, puede apreciarse en la figura 5.23.
- Una vez representados estos datos, los hemos ajustado por medio de un polinomio de grado 6 gracias a una herramienta incluida en Excel.

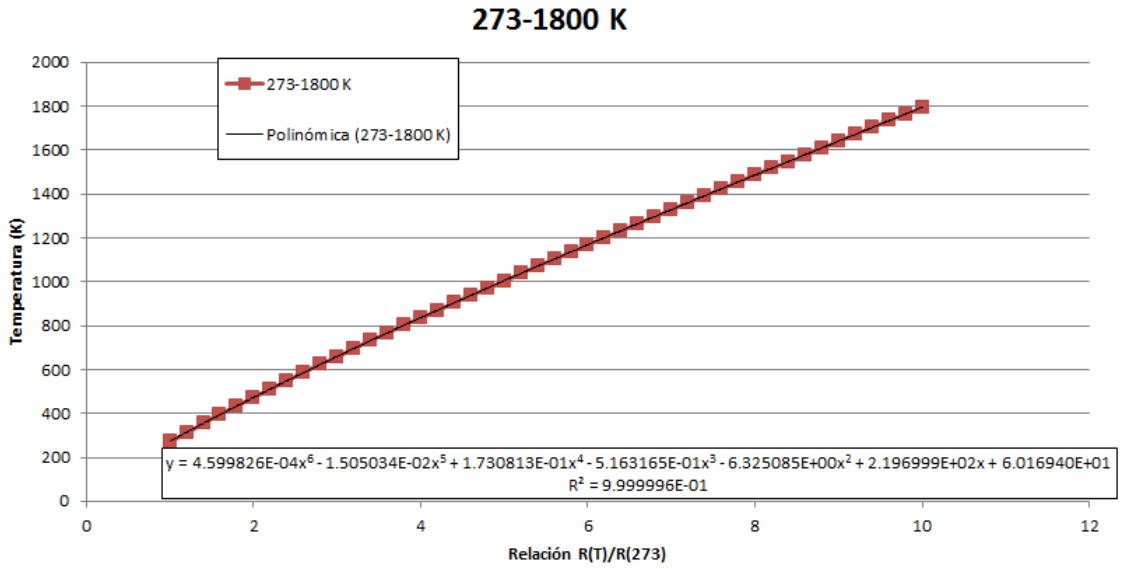


Figura 5.23: Representación de las tablas de Tungsteno [19] (temperatura **T** frente a impedancia **Z=R(T)/R(273)**) junto con el polinomio de grado 6 que ajusta la curva. Mostramos únicamente la representación para el rango de temperaturas 273 – 1800 K porque es el rango en el que vamos a trabajar.

- Ahora que ya tenemos una ecuación para los datos de la tabla 5.2 [19], solo tenemos que sustituir el valor de **R(T)/R(273)** de cada barrido en la ecuación y obtendremos así la temperatura **T** a la que se realizó cada barrido, que es lo que estábamos buscando. En las siguientes tablas podemos apreciar, a modo de resumen, los resultados obtenidos para ambas lámparas.

Barrido	Voltaje (mV)	Corriente (mA)	R lámpara (Ohms)	Relación R(T)/R(273)	Temperatura filamento (K)
1	1.264	0.63	2.00634921	4.340300386	893.680028
2	1.475	0.67	2.20149254	4.762450564	964.781652
3	1.769	0.71	2.4915493	5.389925311	1068.62774
4	2.063	0.75	2.75066667	5.950469418	1159.87994
5	2.453	0.8	3.06625	6.633165361	1269.47535
6	2.836	0.85	3.33647059	7.217728865	1362.20775
7	3.245	0.9	3.60555556	7.799835701	1453.68545
8	3.665	0.95	3.85789474	8.345716668	1538.78731
9	4.05	1	4.05	8.761294647	1603.19778
10	4.53	1.05	4.31428571	9.333019342	1691.41279
11	4.97	1.1	4.51818182	9.774104241	1759.31506

Tabla 5.6: Tabla que contiene las temperaturas obtenidas para cada uno de los barridos de la lámpara halógena. Además del resultado final obtenido para la temperatura, muestra los resultados de los pasos intermedios explicados anteriormente.

Barrido	Voltaje (mV)	Corriente (mA)	R lámpara (Ohms)	Relación R(T)/R(273)	Temperatura filamento (K)
1	1.657	22.5	73.6444444	4.74451907	961.783385
2	2.125	26	81.7307692	5.265477881	1048.1869
3	2.71	30	90.3333333	5.819695239	1138.7028
4	3.521	35	100.6	6.481121856	1245.19682
5	4.36	40	109	7.022289089	1331.30707
6	5.33	45	118.444444	7.630744311	1427.19501
7	6.35	50	127	8.181933159	1513.31726
8	7.45	55	135.454545	8.726614464	1597.83351
9	8.63	60	143.833333	9.266415113	1681.1529
10	9.91	65	152.461538	9.822284385	1766.73251
11	11.25	70	160.714286	10.35396491	1848.744
12	12.68	75	169.066667	10.8920643	1932.46552
13	14.19	80	177.375	11.42732594	2017.28791
14	14.93	82.5	180.969697	11.6589131	2054.74261

Tabla 5.7: Tabla que contiene las temperaturas obtenidas para cada uno de los barridos de la lámpara incandescente. Además del resultado final obtenido para la temperatura, muestra los resultados de los pasos intermedios explicados anteriormente.

Como podemos observar, la temperatura obtenida se incrementa para cada barrido, esto tiene sentido ya que el voltaje y la corriente a los que las lámparas se veían sometidas también fueron incrementándose. Ahora que ya conocemos la temperatura **T** a la que se realizó cada barrido, podemos continuar con la comparación entre estos y el espectro de radiación del cuerpo negro **W(λ,T)** y proceder así al cálculo de la constante **M** de la que hablábamos anteriormente.

5.4.2.3 Cálculo de la constante M

Para comparar los espectros de las lámparas con el espectro de radiación del cuerpo negro **W(λ,T)**, deberemos representar el espectro de una de las lámparas, obtenido para un barrido en concreto, junto con el espectro de radiación del cuerpo negro a la temperatura a la que se realizó el barrido, esto mismo deberemos repetirlo para cada uno de los barridos y para ambas lámparas. Como ya comentamos a lo largo del presente capítulo, las curvas deberían diferir únicamente en una constante, la constante **M**, la cual, además, debería ser la misma para todos y cada uno de los barridos, y para ambas lámparas.

Entonces, nuestro objetivo será calcular esa constante, para ello, ajustaremos cada uno de los barridos por separado calculando así la constante para cada uno de ellos y, posteriormente, realizaremos el promedio de todas ellas para obtener la constante **M** buscada. Para esta tarea recurrimos a la herramienta matemática Matlab, al igual que en otras ocasiones a lo largo del presente TFG.

5.4.2.4 Ajuste con Matlab mediante mínimos cuadrados

Para realizar el ajuste mediante el método de mínimos cuadrados programamos un *script* de Matlab llamado `Matriz.m`, el apéndice A explica dónde encontrarlo. Lo primero que hicimos fue formar la matriz **PD**, que almacena los datos de los barridos corregidos obtenidos por el fotodiodo, cada una de sus columnas contiene un barrido

diferente. Posteriormente, formamos una nueva matriz llamada **W**, esta matriz almacena los espectros de radiación del cuerpo negro para cada uno de los barridos en cada una de sus columnas. Aclarar, que cuando decimos que el espectro de radiación de cuerpo negro es para un barrido, nos referimos a que los datos fueron calculados para la temperatura a la que se realizó el barrido en cuestión. Por último, formamos el vector vacío **cte**, en el que posteriormente almacenaremos las constantes calculadas para cada barrido, las cuales, recordemos que deberían ser muy similares.

No olvidemos que lo que queremos es ajustar los datos corregidos medidos por el fotodiodo al espectro de radiación del cuerpo negro a esa temperatura. El ajuste lo realizaremos mediante una constante, ya que sabemos que se cumple la siguiente ecuación para cada barrido, que es equivalente a la ecuación 5.3, que mostrábamos al comienzo de este apartado.

$$\boldsymbol{PD}(:,\boldsymbol{i}) \cdot \boldsymbol{cte}(\boldsymbol{i}) = \boldsymbol{W}(:,\boldsymbol{i}) \tag{5.9}$$

Notar que **PD** y **W** son dos matrices conocidas (contienen los datos de los barridos y del espectro de radiación del cuerpo negro a la temperatura de cada uno de los barridos, respectivamente) y **cte** es el vector columna que contiene los parámetros a estimar, es decir, las constantes **M** de todos los barridos. Notar además que, en la ecuación (5.9) nos estamos quedando con las columnas i-ésimas de **PD** y **W**, y con el valor i-ésimo del vector **cte**, es decir, estamos quedándonos con un barrido en concreto, el barrido i-ésimo. Entonces, estaremos en disposición de calcular la constante **M** buscada mediante el método de mínimos cuadrados que dice que:

$$\boldsymbol{cte}(\boldsymbol{i}) = (\boldsymbol{PD}(:,\boldsymbol{i})' \cdot \boldsymbol{PD}(:,\boldsymbol{i}))^{-1} \cdot \boldsymbol{PD}(:,\boldsymbol{i})' \cdot \boldsymbol{W}(:,\boldsymbol{i}) \tag{5.10}$$

Donde $\boldsymbol{PD}(:,\boldsymbol{i})'$ es el vector traspuesto y conjugado de $\boldsymbol{PD}(:,\boldsymbol{i})$, como son números reales, coincide con el vector traspuesto.

Además, podríamos ir más lejos y calcular la constante **M** para cada uno de los barridos de una sola vez gracias a que el método de mínimos cuadrados es un método muy potente. Se cumpliría la siguiente relación:

$$\boldsymbol{PD} \cdot \boldsymbol{Cte} = \boldsymbol{W} \tag{5.11}$$

Notar que esta vez **PD** y **W** son matrices conocidas, que contienen los datos de todos los barridos y del espectro de radiación del cuerpo negro a la temperatura de cada barrido, y que **Cte** también es otra matriz en cuya diagonal principal se almacenan los parámetros a estimar (las constantes **M**) y el resto de elementos son iguales a cero.

Entonces, el método de mínimos cuadrados seguiría cumpliéndose, es decir, se seguiría cumpliendo la siguiente relación:

$$\boldsymbol{Cte} = (\boldsymbol{PD}' \cdot \boldsymbol{PD})^{-1} \cdot \boldsymbol{PD}' \cdot \boldsymbol{W} \tag{5.12}$$

Además, obtener el vector **cte** del que hablábamos previamente en Matlab sería muy sencillo:

$$\boldsymbol{cte = diag(Cte)} \tag{5.13}$$

Así que podríamos calcular todo el vector **cte** en una sola línea del *script* pero, en cada barrido, nuestro interés se centra en un rango diferente de longitudes de onda, ya que si el fotodiodo apenas recibe luz o, por el contrario, se ve saturado, deberemos desecharlo. Es por esto que separamos por barridos a la hora de calcular las constantes **M** del vector **cte**. Todas las constantes almacenadas en este vector (mostradas en la tabla 5.8) deberían ser iguales o, en su defecto, muy similares.

Lámpara Halógena		**Lámpara Incandescente**	
Barrido	**Constante M ($x10^7$)**	**Barrido**	**Constante M ($x10^8$)**
1	3.6752	1	S/N
2	1.7379	2	1.8648
3	1.9350	3	1.7106
4	2.2386	4	2.0436
5	2.8022	5	2.0684
6	3.5168	6	2.0932
7	4.0309	7	2.4726
8	4.5421	8	2.7159
9	4.7139	9	3.0900
10	S/N	10	3.2824
11	S/N	11	3.4468
		12	3.6379
		13	3.8508
		14	3.8987
Media	**3.24**	**Media**	**2.78**
Desviación típica	**1.11**	**Desviación típica**	**0.79**

Tabla 5.8: Tabla que muestra los resultados de las constantes obtenidas para los barridos realizados con ambas lámparas, así como la media de esa constante y la desviación típica para cada lámpara.

Una vez calculado el vector **cte** para cada lámpara, obtuvimos la media de todos sus elementos para obtener la constante buscada común a todos los barridos, la constante **M**, procedimos al cálculo de esta constante para cada lámpara por separado a pesar de que debería ser la misma para ambas. Como podemos observar en la tabla 5.8, la constante **M** obtenida para la lámpara halógena es del orden de 10 veces menor que la obtenida para la incandescente. Además si nos fijamos en las desviaciones típicas obtenidas para ambas lámparas, observamos que son del orden de 10^7, son enormes, lo cual indica que las constantes obtenidas para cada barrido, incluso para una misma lámpara, también difieren mucho.

A pesar de ser conscientes de los malos resultados obtenidos, pasamos a comprobarlos, es decir, a representar los datos de los barridos corregidos y multiplicados por la constante promedio **M**, junto con el espectro de radiación del cuerpo negro a la temperatura de cada barrido para comprobar si se ajustaban realmente tan mal como esperábamos. Los resultados pueden verse en las figuras 5.24 y 5.25, las líneas de menor grosor representan los datos obtenidos de cada barrido (ya corregidos por la

función **F2** y multiplicados por la constante promedio **M** obtenida para cada lámpara) y las líneas rojas representan los espectros de radiación del cuerpo negro correspondientes a cada barrido. Entonces, cada una de las líneas de menor grosor debería ajustarse lo más posible a su línea roja correspondiente y, cómo podemos observar, esto no ocurre, las curvas se ajustan realmente mal.

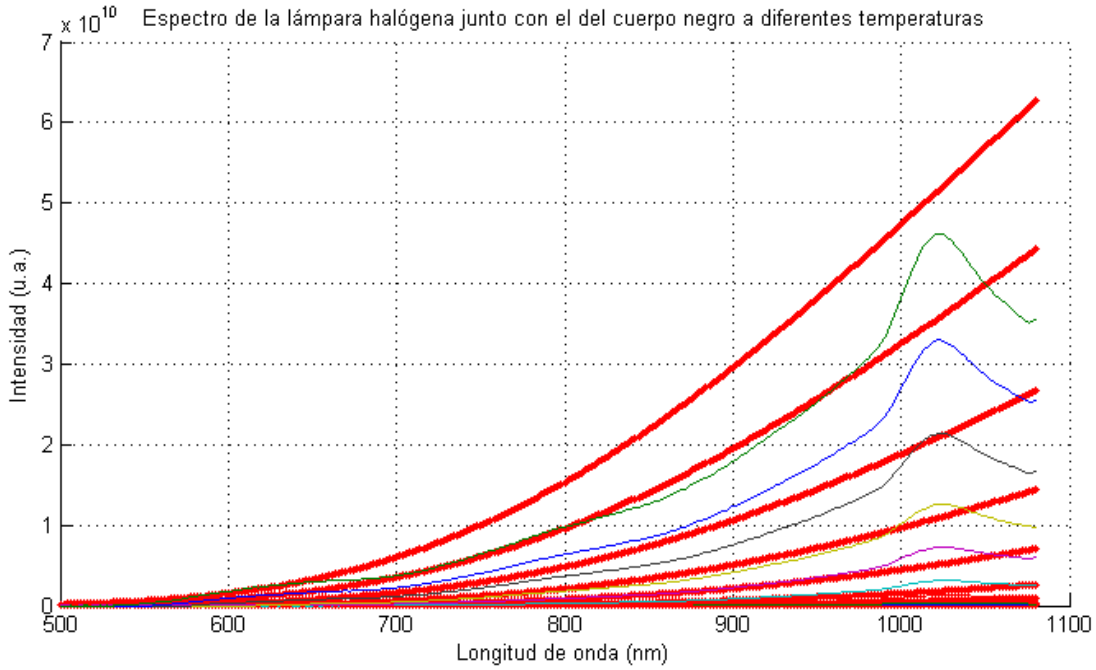
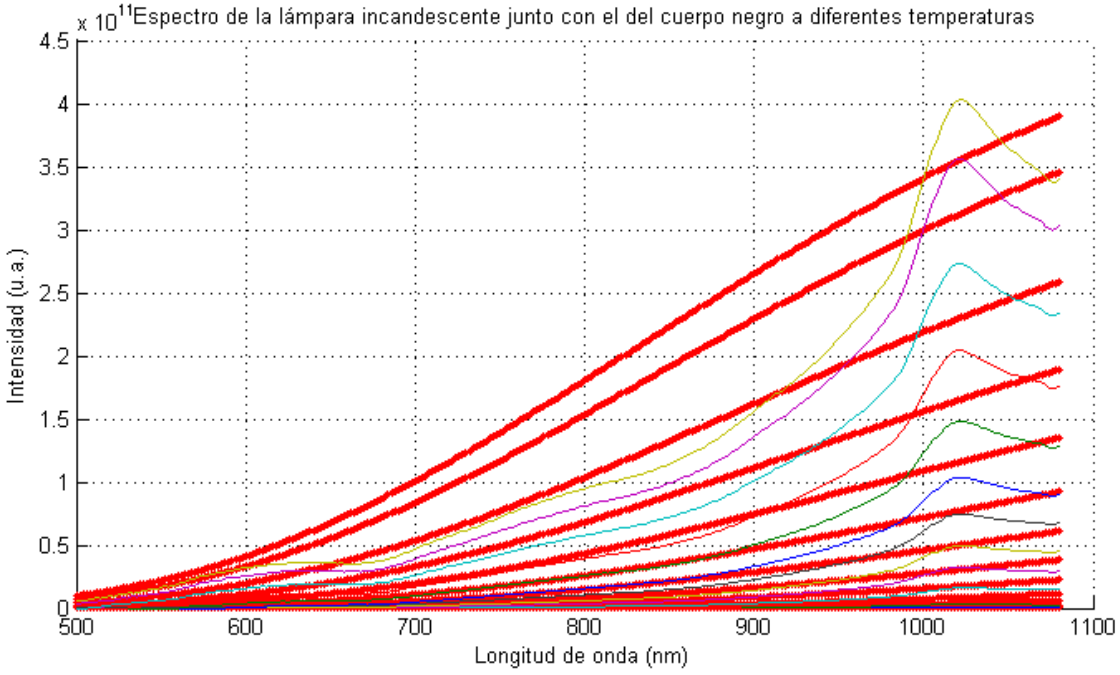


Figura 5.24: Representación en Matlab de los espectros de emisión de la lámpara halógena a diferentes temperaturas corregidos y multiplicados por la constante promedio **M** obtenida, junto con el espectro de radiación del cuerpo negro correspondiente a la temperatura de cada uno de los barridos.



.

Figura 5.25: Representación en Matlab de los espectros de emisión de la lámpara incandescente a diferentes temperaturas corregidos y multiplicados por la constante promedio **M** obtenida, junto con el espectro de radiación del cuerpo negro correspondiente a la temperatura de cada uno de los barridos.

Debido a los malos resultados obtenidos, decidimos utilizar otro método para ajustar los espectros de las lámparas al del cuerpo negro. Se trata de la herramienta Curve Fitting Tool que nos proporciona Matlab.

5.4.2.5 Ajuste con Matlab mediante Curve Fitting Tool

Al utilizar este nuevo método, además de dar libertad al parámetro **M**, también daremos libertad a otro parámetro, la temperatura **T**. Entonces, no utilizaremos las temperaturas que obtuvimos previamente para cada barrido (apartado 5.4.2.2), sino que Matlab se encargará de buscar la que mejor se ajuste a cada caso.

A continuación, vamos a explicar el funcionamiento de la herramienta, para ello utilizaremos únicamente una de las lámparas, la halógena, ya que para la incandescente sería exactamente lo mismo, solo cambiarían los resultados obtenidos, los cuales mostraremos al final para ambas lámparas.

En la figura 5.26 se muestra el aspecto de la herramienta en cuestión, pasemos pues a indicar cómo utilizar la herramienta para nuestros propósitos:

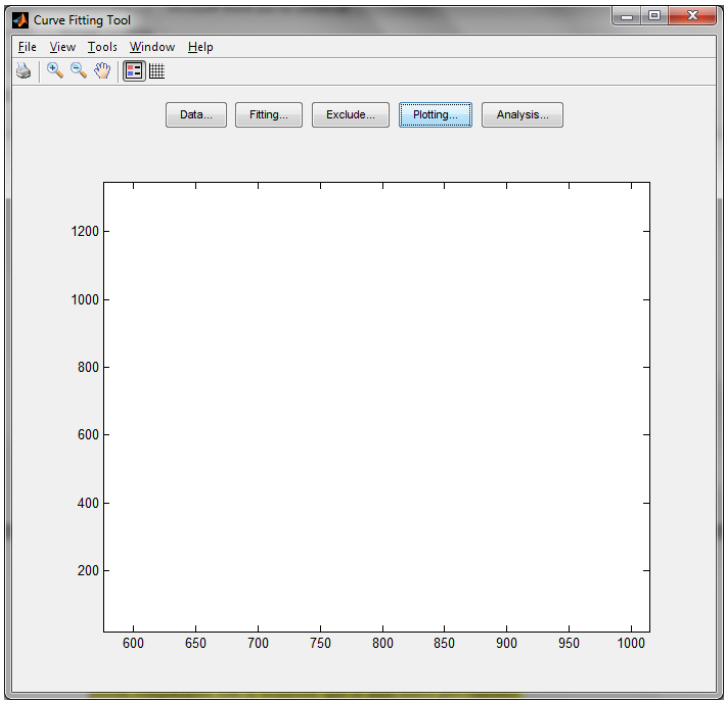


Figura 5.26: Aspecto de la herramienta Curve Fitting Tool de Matlab.

- Primeramente cargamos los datos del primer barrido (**Y1**). Para ello hacemos clic en el botón `Data…` y seleccionamos el vector del eje X y el vector del eje Y (ambos, claro está, deberán tener la misma longitud). Comentar que ya teníamos seleccionado el rango de interés de este barrido en concreto antes de cargar los datos. En la figura 5.27 puede apreciarse el resultado de seleccionar los datos del primer barrido, aparecen tan pocos puntos representados porque el rango de interés de este barrido es muy pequeño, la lámpara emitía con poca intensidad.

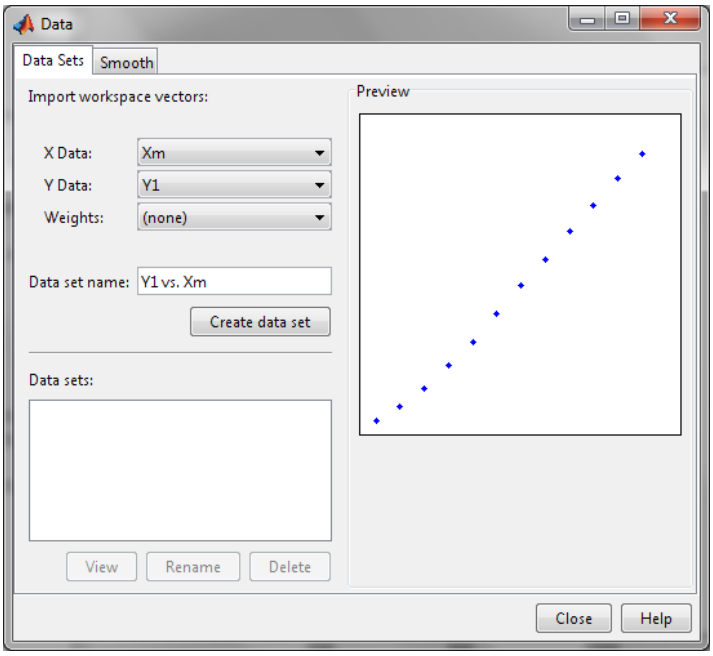


Figura 5.27: Aspecto de la ventana `Data` de la herramienta Curve Fitting Tool. Por medio de esta ventana se seleccionan los datos que se desean ajustar.

- A continuación deberemos indicarle a Matlab la ecuación que obedece el espectro de radiación del cuerpo negro (5.6), de la cual solo desconocemos la temperatura **T**, y además multiplicaremos esa ecuación por una constante, la constante **M**. Para ello hacemos clic en el botón `Fitting...` y aparecerá la siguiente ventana.

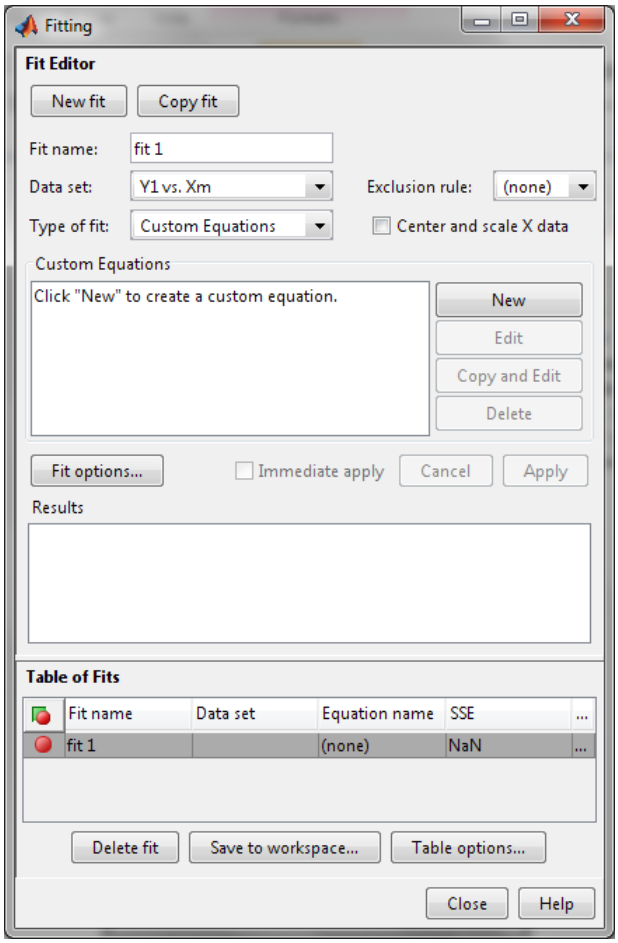


Figura 5.28: Aspecto de la ventana `Fitting` de la herramienta Curve Fitting Tool. Por medio de esta ventana se selecciona el tipo de ajuste que se desea realizar y los datos a los cuales deseamos ajustarnos.

En esta ventana deberemos seleccionar los datos que deseamos ajustar (**Y1**) y el tipo de ajuste a realizar, que en nuestro caso será `Custom Equations`. A continuación, deberemos crear una nueva ecuación, que será el espectro de radiación del cuerpo negro, mostrado en la ecuación (5.6). Esta ecuación será implementada dependiendo únicamente de los parámetros a estimar (**M** y **T**) y de la longitud de onda (llamada x en este caso), puede apreciarse a continuación, en la ecuación (5.14). Los valores constantes mostrados en la ecuación fueron calculados mediante Matlab y coinciden con los valores mostrados en [18]. En la figura 5.29 podemos apreciar la ecuación implementada en la herramienta de ajuste de Matlab.

$$\boldsymbol{W} = \boldsymbol{M} * \frac{3.743991056792121 \cdot 10^{-16}}{x^5 \cdot exp(\ 0.014403072438696 / \ (x * \boldsymbol{T})\ ) - 1} \tag{5.14}$$

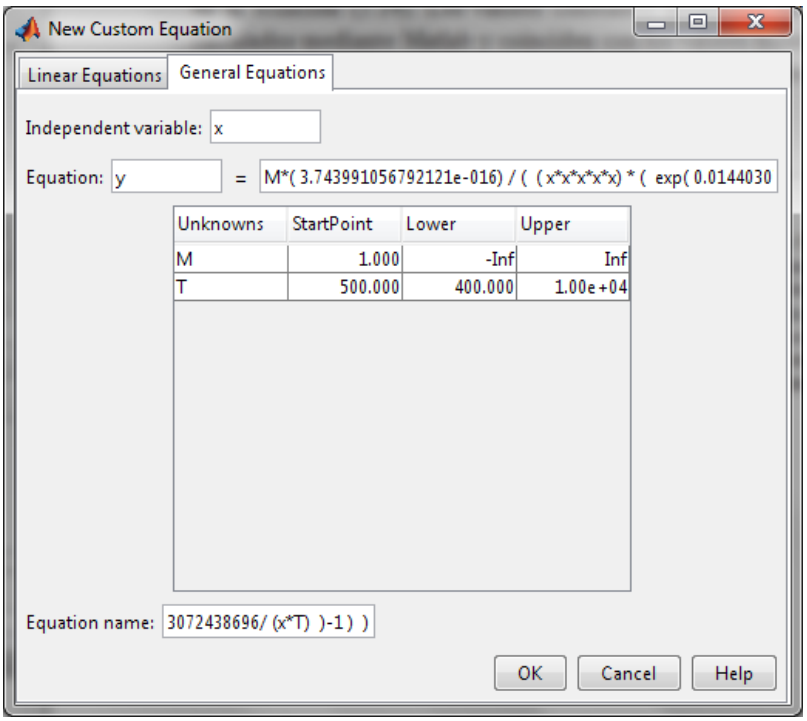


Figura 5.29: Implementación de la ecuación del espectro de radiación del cuerpo negro multiplicado por la constante **M**. Podemos observar los dos parámetros desconocidos que van a ser estimados.

- Cuando hayamos cargado los datos de todos los barridos y hayamos creado nuevos ajustes para cada uno de ellos, la ventana `Fitting` tendrá el aspecto de la figura 5.30. Ahora, en la ventana podremos seleccionar el ajuste deseado y observar los resultados obtenidos, los errores de ajuste cometidos, etc. Lo que a nosotros nos interesará será el valor de los dos parámetros a estimar **M** y **T**, los cuales pueden apreciarse para el primer barrido en la figura 5.30. Posteriormente recogeremos los resultados obtenidos para estos parámetros y los mostraremos todos juntos en una tabla a modo de resumen.

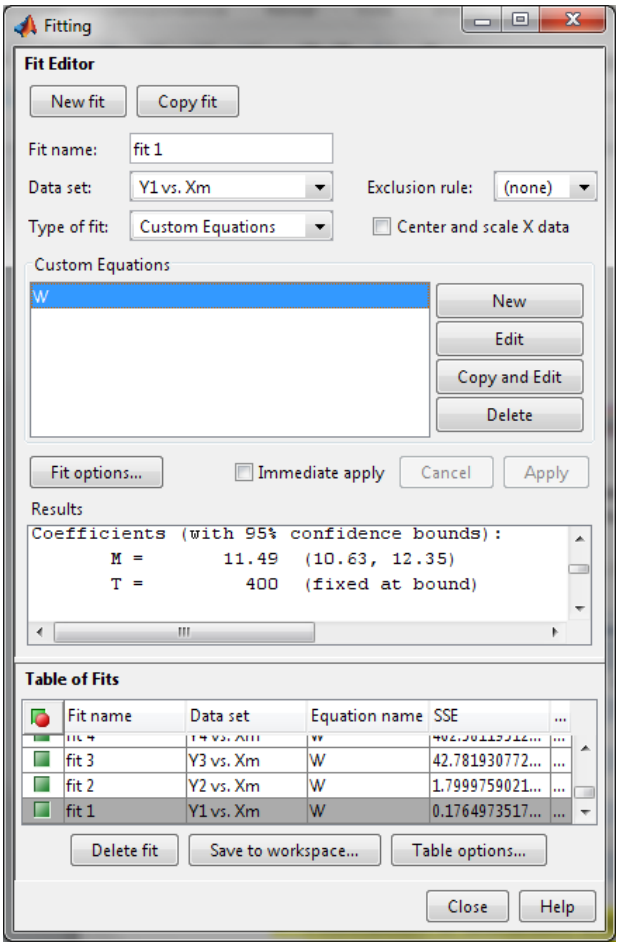


Figura 5.30: Aspecto de la ventana `Fitting` una vez cargados todos los datos de todos los barridos y creados los ajustes a cada uno de ellos.

- Volvamos a la ventana principal de la herramienta para observar el resultado de los ajustes obtenidos (figura 5.31). Podemos apreciar que los datos se ajustan bastante bien, pero recordemos que los ajustes se han realizado por separado y cada barrido tiene una constante **M** distinta.

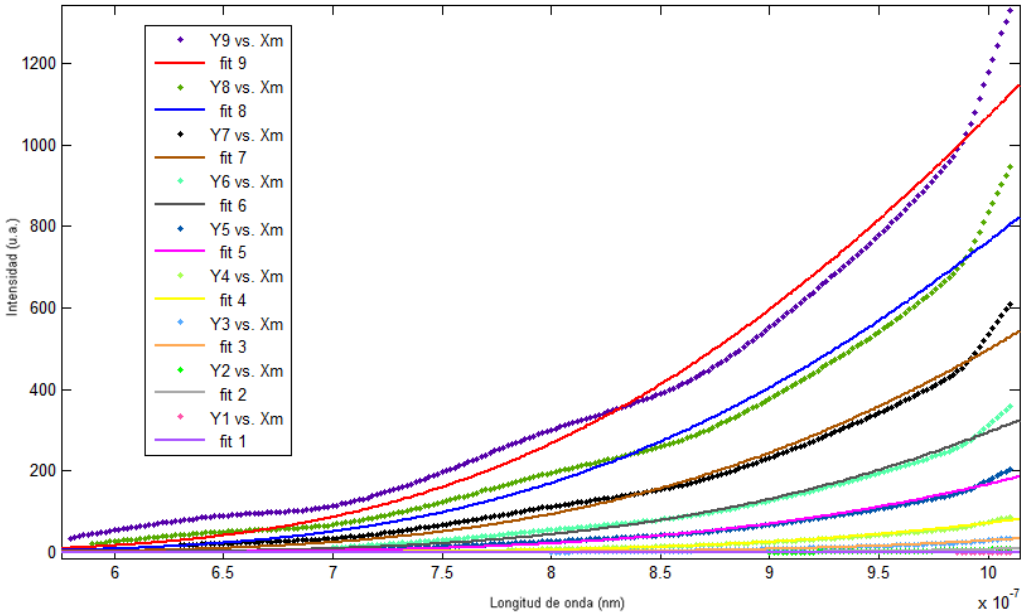


Figura 5.31: Aspecto de la ventana principal de la herramienta Curve Fitting Tool una vez que se han añadido todos los barridos de la lámpara halógena y se han realizado los ajustes al espectro de radiación del cuerpo negro.

A continuación, mostramos sendas tablas (tablas 5.9 y 5.10) a modo de resumen de los resultados obtenidos para cada uno de los barridos de ambas lámparas. Además, en la tabla 5.11 puede apreciarse el valor medio de la constante **M** y su desviación típica para cada lámpara por separado.

Barrido	Valor temperatura T	Valor constante M
1	400	11.49
2	692.3	2.321e-005
3	908.6	6.061e-007
4	1038	2.057e-007
5	1152	1.205e-007
6	1196	1.336e-007
7	1297	8.885e-008
8	1379	6.991e-008
9	1440	6.296e-008

Tabla 5.9: Tabla que muestra los resultados obtenidos de los dos parámetros a estimar para cada uno de los barridos de la lámpara halógena mediante Curve Fitting Tool.

Barrido	Valor temperatura T	Valor constante M
1	S/N	S/N
2	402.4	22.83
3	646.3	0.000108
4	807.8	3.103e-006
5	931.8	5.966e-007
6	1070	1.614e-007
7	1070	2.513e-007
8	1180	1.064e-007
9	1214	1.058e-007
10	1296	7.084e-008
11	1378	4.975e-008
12	1458	3.725e-008
13	1508	3.468e-008
14	1518	3.703e-008

Tabla 5.10: Tabla que muestra los resultados obtenidos de los dos parámetros a estimar para cada uno de los barridos de la lámpara incandescente mediante Curve Fitting Tool.

Lámpara Halógena		**Lámpara Incandescente**	
Valor promedio	1.28	Valor promedio	1.76
Desviación típica	3.83	Desviación típica	6.33

Tabla 5.11: Tabla que muestra los resultados promedio obtenidos del parámetro **M**, así como la desviación típica para ambas lámparas mediante la herramienta Curve Fitting Tool.

Llegados a este punto, solo nos quedará comprobar los resultados proporcionados por Matlab. Pues bien, si nos fijamos en las temperaturas proporcionadas por Curve Fitting Tool y las comparamos con las obtenidas mediante las tablas de Tungsteno (mostradas en las tablas 5.6 y 5.7), podemos observar que no tienen relación aparente, difieren en unos 200 K para la lámpara halógena y en unos 470 K para la lámpara incandescente. También podemos apreciar que las constantes **M** obtenidas para cada barrido no son

nada similares entre sí para ninguna de las 2 lámparas, difieren incluso más que para el método de mínimos cuadrados que ya desechamos en el apartado anterior.

Si representamos los resultados obtenidos para **M** y **T** mediante la herramienta Curve Fitting Tool (figura 5.32), observamos que ambas lámparas (halógena e incandescente) siguen el mismo patrón, existe una correlación entre ambos parámetros (**M** y **T**). Esto indica que, efectivamente, ambas lámparas poseen el mismo comportamiento y que hay algún efecto que hemos ignorado a la hora de calibrar. En un trabajo futuro deberíamos indagar sobre estos posibles efectos que no estamos teniendo en cuenta.

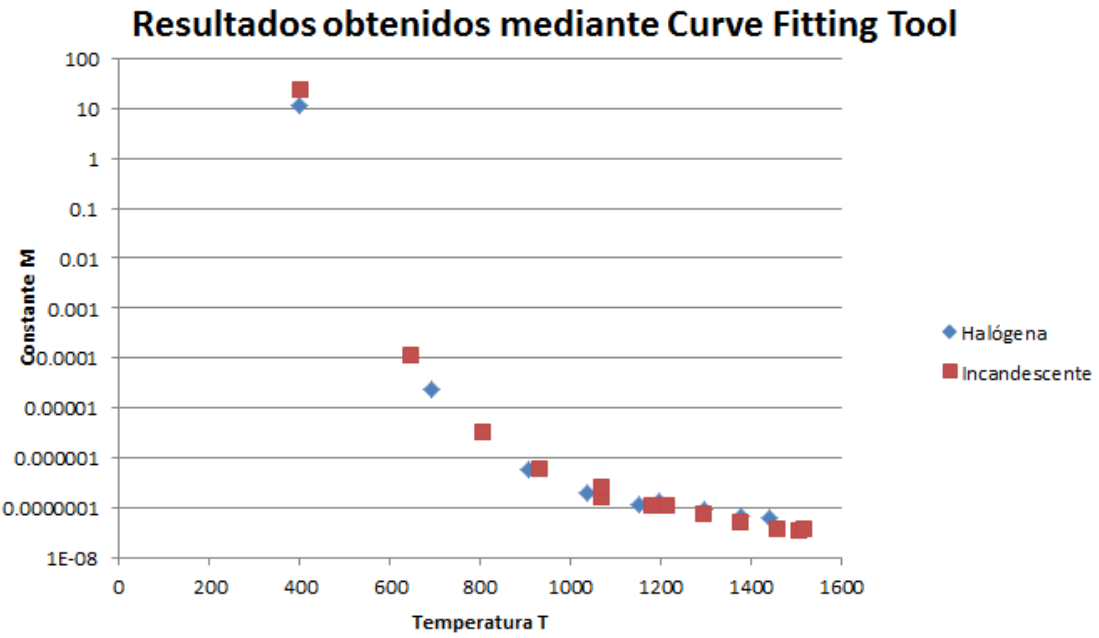


Figura 5.32: Representación de los resultados obtenidos para la constante **M** y la temperatura **T** mediante la herramienta Curve Fitting Tool de Matlab.

Por todo esto no pudimos añadir la constante **M** al código de la aplicación y, por lo tanto, no pudimos darle magnitud (unidades) a la amplitud de las señales obtenidas por el fotodiodo. Tuvimos que conformarnos con la corrección de la forma proporcionada por la función **F2**.

5.5 CONCLUSIÓN

Como conclusión, decir que en cuanto a la cuantificación del fondo de escala de ruido hemos obtenido unos buenos resultados y hemos podido aplicarlos al código y a las señales obtenidas mediante el fotodiodo, evitándonos así, el tener que mostrar los niveles de ruido a la hora de representar las señales.

También hemos obtenido una buena función de ajuste a la respuesta espectral del fotodiodo. Aunque al incluir esta función en el código, hemos observado que la señal calibrada por esta respuesta espectral apenas difiere de la señal sin calibrar (ver figuras 5.16, 5.17 y 5.18). Esto podría deberse a que la respuesta espectral del fotodiodo produzca una deformación muy pequeña sobre las señales captadas, entonces la

corrección también debe ser ligera. Tendremos que comparar algunos ejemplos de espectros de emisión, obtenidos para diferentes LEDs mediante nuestro sistema, con los espectros mostrados por las hojas de especificaciones de dichos LEDs, entonces podremos comprobar si la calibración de la forma de la señal ha sido lo suficientemente buena. Esto lo haremos a lo largo del próximo capítulo, como ya comentamos previamente.

En cuanto a los resultados obtenidos para la constante **M**, comentar que no han sido tan buenos como esperábamos y que no hemos podido añadir esta constante al código para darle unidades a las medidas realizadas por el sistema. Esto podría deberse a que se cometiese algún error a la hora de alinear la lámpara con la ranura del monocromador o a que los vidrios de las lámparas absorbiesen parte de la luz emitida por las mismas a ciertas frecuencias, entonces el espectro de radiación ya no se asemejaría perfectamente al del cuerpo negro. Además, recordemos que al representar los resultados obtenidos para **M** y **T** (figura 5.32), estos parecen seguir el mismo comportamiento para ambas lámparas, lo que indica que hay algún efecto que se nos está escapando a la hora de calibrar. En un trabajo futuro deberíamos seguir esta línea de trabajo para descubrir cuáles son estos efectos y poderlos tener en cuenta durante la calibración del sistema.

CAPÍTULO 6. MEDIDAS EXPERIMENTALES

6.1 INTRODUCCIÓN

Ahora que ya tenemos nuestro sistema programado y calibrado, podemos pasar a realizar una serie de medidas para comprobar el funcionamiento del mismo. A lo largo del presente capítulo mostraremos diversos ejemplos de medidas realizadas mediante nuestro sistema, obtendremos los espectros de emisión de diferentes LEDs y los compararemos con los espectros mostrados por sus hojas de especificaciones.

6.2 EJEMPLOS DE MEDIDAS REALIZADAS PARA DIFERENTES LEDS DE INSERCIÓN DE 5 mm

A continuación mostramos algunos ejemplos de funcionamiento de nuestro sistema, es decir, mostraremos algunos espectros de emisión obtenidos para diferentes LEDs. En este caso, utilizamos varios LEDs de diferentes colores, todos ellos empaquetados en el popular $T1^{3/4}$, un empaquetado de 5 mm de diámetro para el cuál se diseñó explícitamente un soporte para facilitar su acople a la ranura de entrada del monocromador. Concretamente utilizaremos los siguientes modelos de LED, HLMP-39xx, HLMP-38xx y HLMP-37xx, los cuales se corresponden respectivamente a los colores verde, amarillo y rojo, el apéndice A explica dónde encontrar las hojas de especificaciones de dichos LEDs. Aclarar que el LED de color amarillo utilizado a continuación es distinto del utilizado como ejemplo en varias ocasiones a lo largo del presente TFG.

Pues bien, pasemos entonces a la representación de los espectros de emisión de estos LEDs obtenidos mediante nuestro sistema y a su posterior comparación con los espectros mostrados por las hojas de especificaciones de los mismos. En la figura 6.1 podemos observar los espectros obtenidos mediante nuestro sistema para los tres LEDs en cuestión, representamos la columna **Xcorr** del fichero de datos. Notar además que hemos representado cada LED con su color correspondiente para facilitar la visualización de la imagen. Para realizar esta representación hemos empleado los ficheros de datos que genera nuestra aplicación junto con un *script* de Matlab llamado `LEDs.m`, el apéndice A explica dónde pueden encontrarse tanto los ficheros como el *script*. Aclarar que hemos representado los tres espectros en una misma imagen debido a

que así aparecen en las hojas de especificaciones (figura 6.2), de esta manera facilitamos la comparación.

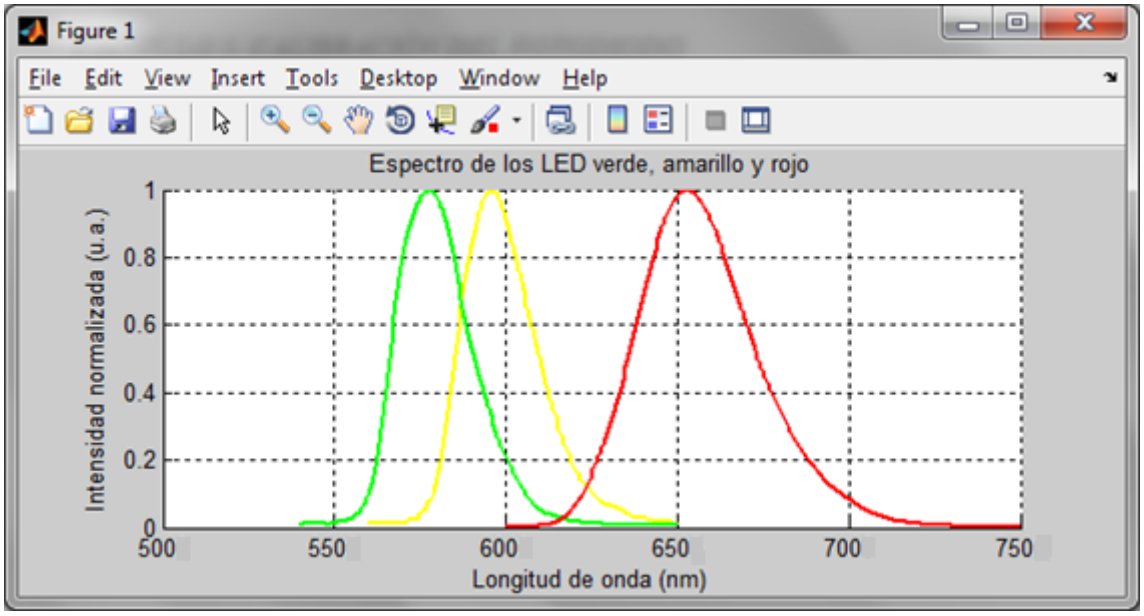


Figura 6.1: Espectros de emisión de tres LEDs de diferentes colores (verde, amarillo y rojo) obtenidos mediante nuestro sistema y representados con ayuda de Matlab.

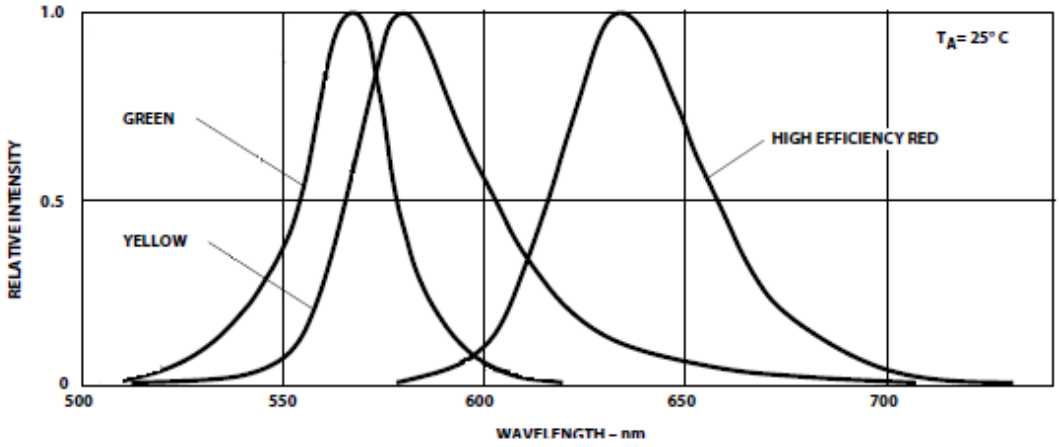


Figura 6.2: Espectros de emisión para los mismos LEDs que los mostrados en la imagen anterior pero, en este caso, estos espectros están sacados de las hojas de especificaciones de dichos LEDs.

Aparentemente, si comparamos las figuras anteriores, los máximos aparecen desplazados ligeramente a la derecha en los resultados obtenidos mediante nuestro sistema, respecto de los espectros mostrados por las hojas de especificaciones. Procedamos a comparar los resultados para verificarlo, en la siguiente tabla mostramos los máximos de emisión obtenidos para cada uno de los LEDs mediante nuestro sistema y los comparamos con los máximos según las hojas de especificaciones.

LED	Sistema (nm)	Hoja especif. (nm)	Error (nm)
Verde	577.5	565	12.5
Amarillo	595.5	583	12.5
Rojo	652	635	17

Tabla 6.1: Tabla que contiene los máximos de emisión de los LEDs obtenidos mediante nuestro sistema y los compara con los ofrecidos por las hojas de especificaciones.

Según la tabla 6.1 parece que, efectivamente, tenemos un ligero error de desplazamiento hacia la derecha, de hecho, si recordamos los rangos del espectro para cada color dentro de la región visible, observamos que el LED de color verde no está emitiendo ni tan siquiera en su región (verde (492 – 577 nm)), está desplazado hacia su derecha metiéndose en la región del color amarillo. Esto sugiere que hay un pequeño error de calibración en el monocromador, la solución a este problema podría ser corregir este desplazamiento mediante código, pero antes tendríamos que realizar una serie de pruebas para acotar bien ese error.

En cuanto a la forma de los espectros, los obtenidos mediante nuestro sistema parecen más abruptos que los mostrados por las hojas de especificaciones, esto ocurre para los LEDs de color amarillo y verde, no ocurre así para el rojo. Tendríamos que realizar una mejor comparación entre dichos espectros, ya que a simple vista no podemos afirmar que la calibración de la respuesta espectral del fotodiodo sea buena o mala.

6.3 EJEMPLO DE APLICACIÓN PRÁCTICA: PINZAS DE PULSIOXÍMETRO

A parte de estos ejemplos mostrados para LEDs de distintos colores, a continuación mostramos otro ejemplo de aplicación práctica de nuestro sistema. Y es que Jesús Arias estaba realizando un proyecto, por el cuál podía saber el nivel de saturación de oxígeno en sangre gracias a unas pinzas de pulsioxímetro (figura 6.3), las cuales poseen dos LEDs incorporados que deberemos caracterizar.

Figura 6.3: Pinzas de pulsioxímetro utilizadas por Jesús.

El interés de Jesús se centraba más en saber en qué longitud de onda se encontraba el máximo de emisión, y no tanto en saber con qué intensidad emitía el LED, esto hizo que nuestro sistema fuese el ideal, ya que, como comentamos anteriormente, no llegamos a calibrar el sistema en cuanto a amplitud se refiere, no llegamos a darle unidades a las señales obtenidas mediante el fotodiodo. Este hecho hizo que Jesús pensase en nuestro sistema para ayudarle a caracterizar los LEDs que utilizaba en su proyecto, y comprobar si realmente poseían el espectro de emisión que sus hojas de especificaciones indicaban.

A continuación, mostramos los espectros obtenidos gracias al sistema para los LEDs de tres pinzas de Nellcor (figura 6.4). Comentar que para dicha representación se utilizó el *scrit* de Matlab `LEDsJesus.m`, que representa la columna **Xcorr** del fichero de datos y utiliza una nomenclatura, gracias a la cual podemos distinguir entre los diferentes LEDs de cada pinza. Además, añadir que, según las hojas de especificaciones, los LEDs incluidos en las pinzas poseen sus máximos de emisión en los 660 nm (color rojo) y en los 905 nm (infrarrojo). Lamentablemente, el fabricante no ofrece ninguna imagen del espectro de emisión de dichos LEDs, por lo que no podemos mostrarla a continuación para compararla con los resultados obtenidos mediante el sistema.

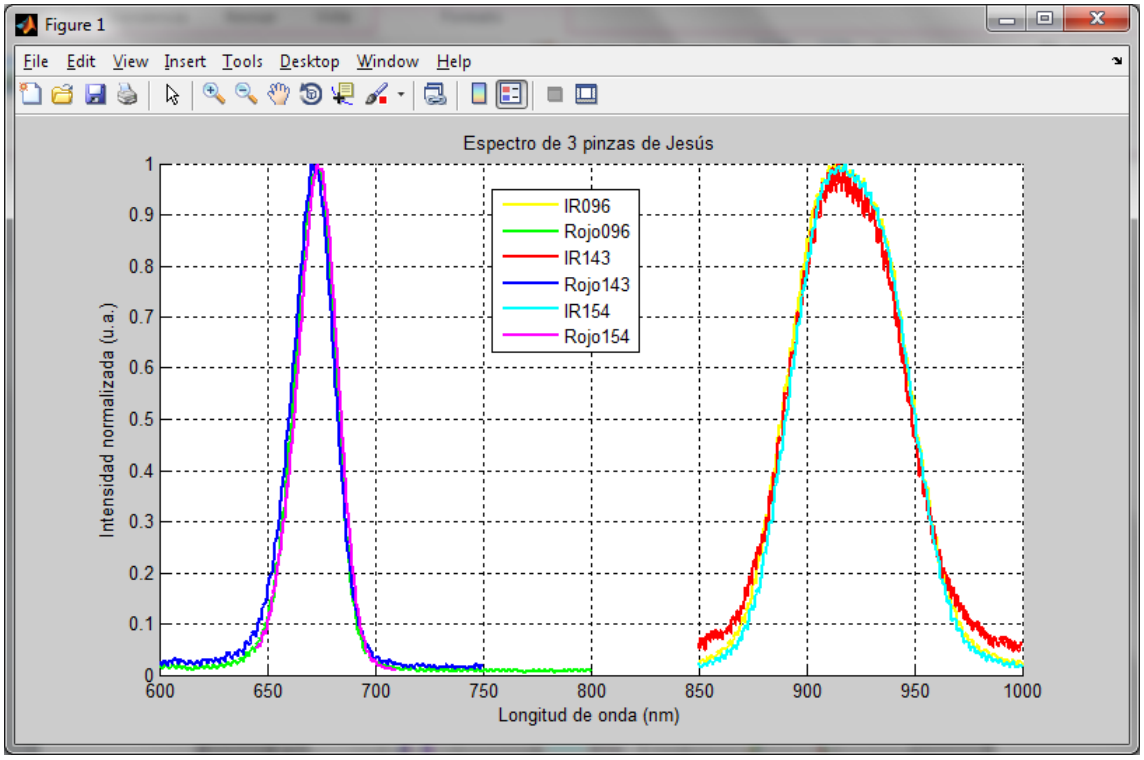


Figura 6.4: Espectros de emisión obtenidos para los LEDs de tres pinzas Nellcor, cada pinza posee un LED que emite en la región del color rojo y otro que emite en el infrarrojo.

Para poder obtener los espectros de emisión de los LEDs de las pinzas, tuvimos que acoplar estas pinzas a la ranura del monocromador de una forma un tanto peculiar, se realizó un soporte de manera rudimentaria (no podemos mostrar el soporte debido a que no se realizó ninguna foto del mismo). Es por esto que, como podemos apreciar en las representaciones de los espectros obtenidas, se introdujo bastante ruido en el sistema. Este inconveniente podría haberse reducido aumentando el número de medidas que el

fotodiodo realiza para obtener cada muestra, veríamos que las curvas del espectro obtenidas se suavizarían. No realizamos este cambio sobre el programa grabado en el microcontrolador debido a que, como comentamos anteriormente, nuestro interés se centraba sobre todo en la longitud de onda y no tanto en el valor exacto de la intensidad emitida por el LED, además este cambio habría ralentizado bastante los barridos realizados.

Como ya comentamos anteriormente, el fabricante de las pinzas no nos proporciona ningún espectro de emisión en sus hojas de especificaciones para compararlo con los obtenidos mediante nuestro sistema. Lo que el fabricante nos proporciona son las longitudes de onda de emisión máxima, por lo que a continuación pasaremos a compararlas con las obtenidas mediante el sistema, y comprobar si continuamos teniendo la desviación en longitud de onda mostrada en el apartado anterior. Para ello, mostramos una tabla con las longitudes de onda de emisión máxima obtenidas mediante nuestro sistema junto con las mostradas por las hojas de especificaciones y las comparamos entre sí.

LED	Sistema (nm)	Hoja especif. (nm)	Error (nm)
IR096	913.25	905	8.25
Rojo096	673.50	660	13.50
IR143	914.75	905	9.75
Rojo143	673.00	660	13.00
IR154	918.00	905	13.00
Rojo154	673.50	660	13.50

Tabla 6.2: Tabla que contiene las longitudes de onda de emisión máxima de las pinzas obtenidos mediante nuestro sistema y las compara con las ofrecidas por las hojas de especificaciones.

Como podemos apreciar en la tabla 6.2, parece que, efectivamente, seguimos teniendo un ligero error de desplazamiento en cuanto a longitud de onda, al igual que en los ejemplos del apartado anterior, esto indica un pequeño error de calibración del monocromador. En un posible trabajo futuro habría que realizar diferentes pruebas para acotar bien este error y, posteriormente, corregirlo mediante código, lo cual resultaría bastante sencillo.

6.4 CONCLUSIÓN

Mediante las diversas medidas realizadas a modo de ejemplo, hemos comprobado que existe un ligero error de calibración del monocromador. Como ya hemos comentado, este error debería ser acotado en un posible trabajo futuro y, una vez acotado, la corrección del mismo mediante código sería bastante sencilla. Además, comentar que al inicio de la realización del TFG, se utilizó otro monocromador MonoSpec 10 que presentaba un error de calibración considerablemente mayor que el actual, fue por esta razón que decidimos cambiarlo.

CAPÍTULO 7. CONCLUSIONES Y TRABAJO FUTURO

7.1 INTRODUCCIÓN

En el desarrollo del presente TFG teníamos el objetivo de desarrollar un sistema capaz de obtener el espectro de emisión de un LED de manera automatizada. Ya partíamos de una base, y es que ya habíamos diseñado y fabricado la parte *hardware* del sistema previamente, durante las prácticas en empresa. Pero para este TFG teníamos pendiente de desarrollar, a nuestro juicio, la parte más compleja del sistema, la parte *software* del mismo. Las partes más importantes desarrolladas durante este TFG han sido la programación, tanto del microcontrolador como de la aplicación del PC, y la calibración, con la cual ya hemos visto que tuvimos una serie de complicaciones.

En las siguientes páginas reflexionaremos tanto sobre el trabajo realizado, como sobre las principales conclusiones que se pueden extraer del mismo, además, propondremos posibles líneas de trabajo futuro, las cuales se centrarán, sobre todo, en nuevas funcionalidades aplicables al sistema desarrollado y en mejoras en cuanto a calibración se refiere.

7.2 CONCLUSIONES

A lo largo de este apartado comentaremos las conclusiones extraídas del trabajo realizado, tanto de la programación del sistema como de la calibración del mismo.

7.2.1 Programación del sistema

Durante la programación del sistema hemos adquirido mayor soltura en el manejo de sistemas operativos basados en GNU/Linux, ya que tanto la programación del microcontrolador como el desarrollo de la aplicación para el PC se llevaron a cabo en dichos entornos. La idea de utilizar este tipo de sistemas operativos no siempre estuvo clara, de hecho, al principio se comenzó programando el microcontrolador en un entorno Windows, pero finalmente nos decantamos por Linux debido a que ha sido el sistema operativo más utilizado durante los estudios del alumno.

El lenguaje de programación que elegimos, tanto para desarrollar la aplicación del PC como para programar el microcontrolador, fue el lenguaje C. Lo elegimos por la misma razón que elegimos un entorno Linux, porque ha sido el lenguaje de programación

predominante durante la carrera. Gracias a esta elección, hemos adquirido nuevas habilidades en dicho lenguaje, como pueden ser, por ejemplo, las comunicaciones con un microcontrolador mediante un puerto serie USB. Además, hemos desarrollado un método de comunicación entre ambos, el cual se basa en una serie de comandos (órdenes) que el PC le envía al microcontrolador y este los ejecuta y contesta al PC con la señal `OK` cuando las tareas que involucra dicho comando han sido realizadas correctamente. Esto ya fue explicado en el capítulo 4, donde puede verse la lista de comandos que el microcontrolador es capaz de reconocer y ejecutar.

La idea principal, a la hora de programar la aplicación para el PC, era desarrollar una interfaz por medio de la cual el usuario fuese capaz de controlar el sistema, pero debido a la falta de tiempo, decidimos que la consola de comandos debería ser suficiente. Esto nos lleva directamente a una de las posibles mejoras sobre las que hablaremos en el apartado de trabajo futuro, el desarrollo de una interfaz de usuario lo más amigable y sencilla posible.

Durante la programación del microcontrolador hemos aprendido a servirnos de buena parte de las funcionalidades del LPC2103, que es un microcontrolador utilizado mayoritariamente en entornos educativos, lo cual no quita que sea una herramienta verdaderamente potente. Las funcionalidades más destacables que hemos aprendido a utilizar del microcontrolador podrían ser los temporizadores y el convertidor A/D, además, los conocimientos adquiridos sobre estos aspectos se pueden extrapolar a la programación de otros microcontroladores fácilmente.

En líneas generales, la programación del sistema fue un éxito ya que todas las tareas que nos propusimos que el sistema fuese capaz de realizar, este las llevó a cabo sin apenas complicaciones. Incluso añadimos alguna tarea adicional sobre la marcha, por ejemplo, durante las prácticas en empresa nos planteamos añadir el selector de ganancia con 4 posibles ganancias seleccionables, lo cual fue un éxito y funcionó a la perfección.

7.2.2 Calibración del sistema

La calibración del sistema realizada durante el presente TFG básicamente se dividió en tres apartados:

- El primero de ellos fue cuantificar el fondo de escala de ruido para cada una de las ganancias seleccionables. Esta cuantificación fue realmente un éxito y conseguimos eliminar el nivel de ruido de las representaciones de las señales obtenidas mediante el fotodiodo. De esta manera conseguimos bajar el nivel de offset ganando así espacio para la representación del espectro de emisión del LED en cuestión, que es donde se centra nuestro interés.
- En segundo lugar, realizamos la calibración de la respuesta espectral del fotodiodo. Ya comentamos en el capítulo 5 (ver figura 5.1) que esta calibración es necesaria debido a que el fotodiodo no es igual de sensible a todas las longitudes de onda, por lo que la representación que realizaremos del espectro

de emisión del LED aparecerá modificada. Para realizar esta calibración tuvimos que ajustar la respuesta espectral del fotodiodo a una función analítica (función **F2**) y posteriormente utilizarla (siguiendo la ecuación (5.2)) para corregir la deformación producida por el fotodiodo. Obtuvimos una buena función de ajuste a la respuesta espectral, pero al comparar los espectros obtenidos para diferentes LEDs mediante nuestro sistema con los espectros mostrados por sus hojas de especificaciones, observamos que la forma de dichos espectros no era exactamente la misma (ver figuras 6.1 y 6.2). Esto puede deberse a que la curva de respuesta del fotodiodo proporcionada por las hojas de especificaciones sea una respuesta promedio, y puede que no se asemeje demasiado a la de nuestro fotodiodo en particular. Para solventar este problema podríamos buscar un fotodiodo más preciso, lo cual nos lleva directamente a una de las posibles mejoras de las que hablaremos a lo largo del siguiente apartado, trabajo futuro.

- En tercer y último lugar, después de corregir la forma de las señales captadas por el fotodiodo, quisimos darles magnitud (unidades). Para ello, utilizamos 2 lámparas (una incandescente y otra halógena) y comparamos sus espectros de emisión con los del cuerpo negro, es decir, comparamos los filamentos de las lámparas con un emisor ideal, con un cuerpo negro. Los espectros de emisión de las lámparas y el del cuerpo negro deberían tener la misma forma, solo podrían diferir en una constante (**M**), la cual habríamos utilizado para dar magnitud a las señales captadas por el fotodiodo pero, como ya vimos, el cálculo de la constante se complicó y esto no fue posible. Los resultados que obtuvimos para esta constante no fueron lo suficientemente buenos como para añadirlos al código y, por lo tanto, tampoco pudimos darle unidades a las medidas realizadas por el sistema, tuvimos que conformarnos con representar las señales normalizadas en amplitud. Los malos resultados obtenidos podrían deberse a que se cometiese algún error a la hora de alinear las lámparas con la ranura del monocromador o a que los vidrios de las lámparas absorbiesen parte de la luz emitida por las mismas a ciertas frecuencias, entonces el espectro de radiación ya no se asemejaría perfectamente al del cuerpo negro. Además, recordemos que al representar los resultados obtenidos para **T** y **M** (figura 5.32), estos parecen seguir el mismo comportamiento para ambas lámparas, lo que indica que hay algún efecto que se nos está escapando a la hora de calibrar. En un trabajo futuro deberíamos seguir esta línea de trabajo para descubrir cuáles son estos efectos y poderlos tener en cuenta durante la calibración del sistema.

Por todo esto concluimos que la calibración del fotodiodo no fue tan buena como esperábamos. A pesar de ello, hemos comprendido el concepto de calibración y por qué es ésta tan necesaria. Incluso hemos aprendido cómo llevarla a cabo de manera teórica para un fotodiodo (que podría extenderse a cualquier otro tipo de sensor), aunque al final no hemos obtenido los resultados deseados al aplicarla en la práctica. Esto nos lleva a pensar que seguramente haya diferentes efectos que no hemos tenido en cuenta a la hora de calibrar, y a la vez nos sirve para darnos cuenta de que, en la práctica, no todo

es tan ideal como en la teoría podría parecer. Además, pensando en el trabajo futuro, mejorar la calibración realizada sería una de las líneas de trabajo más claras a seguir.

7.3 TRABAJO FUTURO

En cuanto al trabajo futuro se pueden distinguir dos claras vertientes: la primera sería la incorporación de nuevas funcionalidades o mejoras al sistema y la segunda estaría enfocada hacia la mejora de la calibración.

7.3.1 Nuevas funcionalidades para el sistema

A continuación, mostramos una serie de nuevas funcionalidades o mejoras sobre las que se podría trabajar. A lo largo de este apartado mostramos algunas posibles mejoras para nuestro sistema, pero esto no quiere decir que sean las únicas, de hecho, estas posibles mejoras podrían llegar a ser infinitas si dejamos correr nuestra imaginación:

- La primera de ellas, a nuestro juicio la más importante, sería el desarrollo de una interfaz que le permitiese al usuario controlar todo el sistema de manera simple e intuitiva. En la figura 7.1 mostramos un ejemplo de lo que podría ser la ventana principal de la aplicación, posteriormente, dependiendo de la tarea que el usuario desee realizar, se deberían mostrar diferentes submenús.

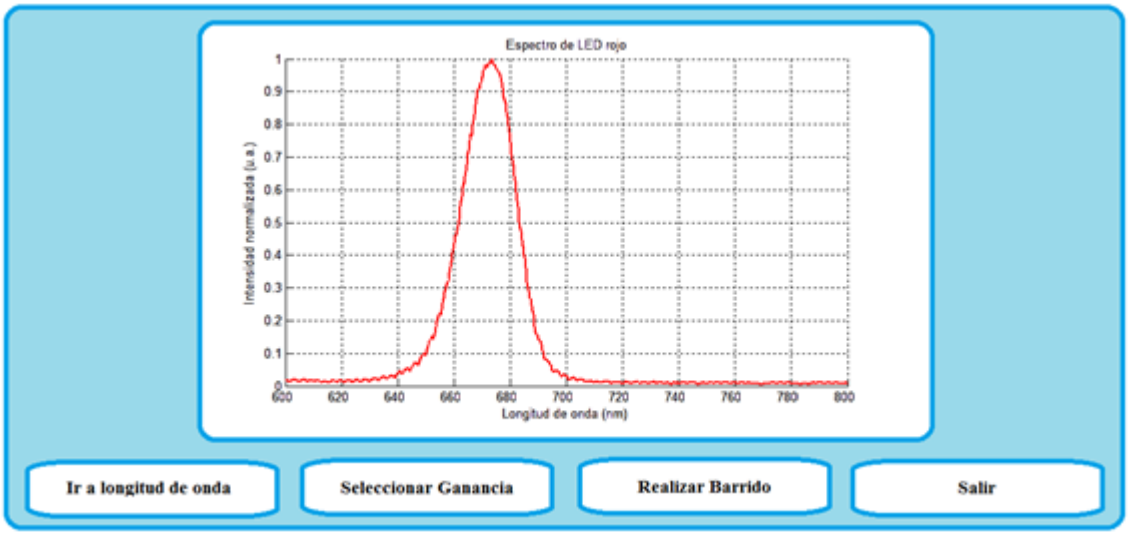


Figura 7.1: Posible interfaz de usuario a desarrollar en un trabajo futuro.

- Otra posible funcionalidad, relacionada con la mencionada anteriormente, sería que la nueva interfaz te permitiera representar más de una señal a la vez y poder incluir marcadores, para poder así comparar dichas señales más fácilmente.
- Recordemos que al LED le proporcionamos una alimentación de 3.3 V por lo que estamos limitados en cuanto a los LEDs que podemos utilizar, ya que solamente podremos utilizar los LEDs cuyo voltaje umbral sea inferior a los 3.3 V. Pues bien, eso deja fuera, por ejemplo, a los LEDs de color azul y de frecuencias mayores, cuyo voltaje umbral es mayor, lo cual nos lleva directamente a otra de las posibles mejoras: alimentar al LED con un voltaje

mayor para poder utilizar así, todo tipo de LEDs. Actualmente, alimentamos al LED directamente desde el microcontrolador, el cual como máximo puede otorgarnos esos 3.3 V, entonces, la nueva forma de alimentar al LED sería usar, por ejemplo, los 5 voltios con los que alimentamos la lógica de los finales de carrera y un MOSFET que funcionase a modo de interruptor controlado por uno de los pines del microcontrolador. De esta manera alimentaríamos al LED con un voltaje mayor y no estaríamos tan limitados en cuanto a los LEDs que podemos utilizar.

- Otra posible mejora sería que al iniciar el sistema el usuario no tuviese que indicarle a la aplicación la posición inicial del monocromador. Esto podría realizarse moviendo el monocromador a la misma posición cada vez que la ejecución de la aplicación finalice, aunque sería un poco engorroso. Otra posible solución sería dotar a la aplicación de una zona de memoria donde pudiese almacenar la posición en la que se encuentra en cada momento, pero no podríamos asegurar que, en el próximo arranque de la aplicación, ese espacio de memoria no se haya sobrescrito.
- En el sistema actual, tenemos desarrollada una PCB para acoplar LEDs de 0.5 mm de diámetro, los cuales se acoplan realmente bien. Una posible mejora sería que se pudiesen acoplan otros LEDs de diferentes tamaños y que, además, este acople pueda realizarse de la manera más sencilla posible. Por ejemplo, diseñando una nueva PCB con un soporte que pueda amoldarse a diferentes tamaños y formas.
- Otra nueva funcionalidad podría ser indicarle a la aplicación los parámetros para la realización de varios barridos (las longitudes de onda inicial y final, la resolución, la ganancia y el nombre de los ficheros en los que deseamos guardar los datos de dichos barridos), y que el sistema los vaya realizando uno tras otro. Así podríamos dejarlo trabajando y pasar a realizar otras tareas en vez de tener que realizar todos los barridos de uno en uno, esperando a que uno termine para poder empezar con el siguiente. Por ejemplo, podríamos dejar al sistema realizando barridos toda la noche y así tenerlos listos a la mañana siguiente, incluso se colaría menos ruido procedente de la luz ambiente.

7.3.2 Mejoras para la calibración

Como ya hemos comentado en varias ocasiones, la calibración de nuestro sistema no salió tan bien como esperábamos, es por esto que una clara línea de actuación futura sería mejorar este aspecto del sistema. A continuación mostramos algunas ideas para mejorar la calibración:

- Buscar un fotodiodo más preciso. Como comentábamos anteriormente, una de las posibles razones por las que la calibración de la respuesta espectral del fotodiodo no tuvo éxito, pudo ser que la repuesta de nuestro fotodiodo no se ajustase demasiado bien a la proporcionada por el fabricante en las hojas de especificaciones. Entonces, podríamos buscar un mejor candidato y ver si

obtenemos mejores resultados de calibración, tanto respecto a la forma de la señal como a la amplitud de la misma, y conseguir darle así magnitud (unidades) a la amplitud a las señales obtenidas.

- Indagar sobre los posibles efectos que estamos ignorando a la hora de calibrar la magnitud de las señales captadas por el fotodiodo. Ya vimos anteriormente que al representar los resultados (**T** y **M**) obtenidos durante la calibración, estos parecían seguir un mismo patrón para ambas lámparas, lo cual nos sugería la existencia de algún efecto que no estábamos teniendo en cuenta a la hora de calibrar. Por cuestiones de tiempo no pudimos seguir trabajando sobre estos resultados, es por esto que una clara línea de trabajo sería continuar por este camino hasta encontrar los posibles efectos ignorados, y tenerlos en cuenta a la hora de calibrar la magnitud de las señales obtenidas mediante el sistema.
- Calibrar el monocromador. Como vimos en el capítulo anterior, los diferentes ejemplos de medidas realizadas sugerían la existencia de un pequeño error de calibración (de unos 13 nm) en cuanto a la posición del monocromador. La solución a este problema podría ser corregir ese pequeño desplazamiento mediante código, pero antes tendríamos que realizar una serie de pruebas para acotar bien ese error.

7.4 OPINIÓN PERSONAL

Como conclusión final, diremos que el sistema cumple con las características requeridas en cuanto a hardware y software se refiere. En cambio, en cuanto a la calibración, esta se podría mejorar y, pensando en un posible trabajo futuro, esa sería una clara línea de actuación (además, claro está, del desarrollo de una interfaz de usuario simple e intuitiva).

Aunque muchas de las tareas realizadas ya las habíamos estudiado, ese estudio había sido solamente teórico y ha sido durante la realización de este TFG donde hemos tenido la oportunidad de llevar a la práctica todo lo aprendido y ver que las cosas funcionan, aunque no tan idealmente como suponemos en la teoría.

También hemos aprendido a valernos por nosotros mismos a la hora de buscar soluciones a los problemas con los que nos encontramos, y a cooperar con otros compañeros que nos otorgan una visión diferente, para lograr la resolución de los mismos. Además, nunca había realizado un trabajo de tales dimensiones, lo que pienso que me puede servir de gran ayuda para un futuro próximo en el que tenga que realizar proyectos de mayor envergadura y, seguramente, colaborando con decenas de personas.

Como opinión final, solo queda añadir que resulta muy satisfactorio darse cuenta de que, a partir de una idea, hemos podido crear un sistema propio y funcional, el cual nos demuestra que todo lo estudiado sirve para algo y tiene aplicación en el mundo real.

REFERENCIAS

[1] http://noticiasdelaciencia.com/not/9950/ademas-de-iluminar-las-bombillas-led-serviran-para-transmitir-datos/

[2] B. Saleh y M. C. Teich, Fundamentals of Photonics (2nd ed), Wiley, 2007.

[3] E. Fred Schubert, Light-Emitting Diodes (1ª ed), Cambridge, 2003.

[4] Jorge Fernández Lucas, Memoria de prácticas externas, Universidad de Valladolid, Julio 2014.

[5] Hojas de especificaciones del monocromador MonoSpec 10.

[6] http://www.bolanosdj.com.ar/TEORIA/SENSORESOPTICOS.PDF

[7] Agrawal G. P. Fiber – Optic Communication Systems (3ed), Wiley – Interscience, EUA, 2002.

[8] http://nemesis.tel.uva.es/images/tCO/contenidos/tema2/tema2_5_2.htm

[9] http://www.info-ab.uclm.es/labelec/solar/otros/infrarrojos/fotodetectores.htm

[10] Hojas de especificaciones del fotodiodo TEMD5010X01.

[11] Carlos A. Reyes, Microcontroladores PIC Programación en Basic (3ra ed), 2008.

[12] Hojas especificaciones microcontrolador LPC2103.

[13] http://arohatgi.info/WebPlotDigitizer/

[14] http://zunzun.com/

[15] http://zeus.df.ufcg.edu.br/labfit/

[16] Dan MacIsaac, Gary Kanner, y Graydon Anderson, Basic Physics of the Incandescent Lamp (Lightbulb), 1999.

[17] John Wilson y John Hawkes, Optoelectronics. An introduction (3ª ed), Prentice-Hall, 1998.

[18] David A. DeWolf, Electro optics handbook, Burle Industries, 1992.

[19] C J Workowski, Temperature measurements in field emission microscopy. Part I: A tungsten resistance temperature scale, 1976.

APÉNDICE. CONTENIDO DEL CD

En este apéndice indicaremos el contenido del CD en el cuál se presenta este Trabajo Fin de Grado. A lo largo del trabajo, hemos comentado en numerosas ocasiones que el apéndice A nos explica cómo acceder a los diversos programas y diseños realizados durante el desarrollo del mismo. Pues bien, todos estos contenidos son accesibles por medio del CD en el cuál se incluye la presente memoria y el contenido de dicho CD es el siguiente:

- Código de la aplicación para PC.
- Código del programa grabado en la memoria flash del microcontrolador.
- Hojas de especificaciones de todos los elementos que componen el sistema:
    - Amplificador Operacional MCP6282.
    - Diodos son los PMEG4005.
    - Fotodiodo TEMD5010X01.
    - Fuente de alimentación DA18-XXXEU-M
    - LED HLMP.
    - Microcontrolador LPC2103.
    - Monocromador MonoSpec10.
    - Multiplexor triple de 2 a 1 CD4053B.
    - Transistores Mosfet PMV37EN de tipo N.
- Diseño esquemático realizado para la PCB mediante la herramienta Proteus ISIS.
- *Layout* de la placa PCB realizado mediante la herramienta Proteus ARES.
- Código de los diferentes *scripts* de Matlab utilizados:
    - Ceros.m
    - CorreccionF2.m
    - Matriz.m
    - LEDs.m
    - LEDsJesus.m