Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

Digitalización y reacondicionamiento de un sistema clínico de cromatografía

Autor: David Rodríguez Martínez

Tutor: Juan Manuel Miguel Jiménez

2023

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES

Trabajo Fin de Grado

Digitalización y reacondicionamiento de un sistema clínico de cromatografía

Autor: David Rodríguez Martínez

Tutor/es: Juan Manuel Miguel Jiménez

TRIBUNAL:

Presidente: José Antonio Jiménez Calvo

Vocal 1º: Óscar Esteban Martínez

Vocal 2º: Juan Manuel Miguel Jiménez

FECHA: 22/05/2023

INDICE

1. RESUMEN 1

2. ABSTRACT 2

3. AGRADECIMIENTOS 3

4. RESUMEN EXTENDIDO 4

4.1 FUNDAMENTO 4
4.2 PRINCIPIO DE MEDICIÓN 4
4.3 METODOLOGÍA DE LA MEDICIÓN 5

5. GLOSARIO DE ACRÓNIMOS Y ABREVIATURAS. 9

6. ESTUDIO TEÓRICO 10

6.1 HARDWARE 10
6.1.1 ADC-SAR 10
6.1.2 Tratamiento de la señal 13
6.1.3 Amplificador Operacional "Rail-to-Rail" 14
6.2 SOFTWARE 15
6.2.1 Entornos de Desarrollo 15
6.2.1.1 Microsoft Visual Studio 2022 15
6.2.1.2 Arduino IDE 15
6.2.2 Lenguajes de programación 15
6.2.2.1 C# 15

7. DEFINICIÓN DEL PROBLEMA Y OBJETIVOS 17

7.1 DEFINICIÓN DEL PROBLEMA 17
7.2 OBJETIVOS 17
7.3 DISPOSITIVOS HARDWARE 18
7.3.1 Arduino Nano 18
7.3.2 Amplificador operacional "rail-to-rail" LMV321 18
7.3.3 PC con Windows que soporte la versión 4.7.2 de .NET Framework 18
7.4 ELEMENTOS SOFTWARE 19
7.4.1 Entorno de desarrollo 19

8. TRANSFORMACIÓN DIGITAL 20

8.1 ANÁLISIS DE LA SITUACIÓN INICIAL Y NECESIDADES DEL USUARIO 20
8.1.1 Análisis del uso de la impresora matricial 20
8.1.2 Oportunidades de mejora con la digitalización de la impresora matricial 20
8.1.3 Análisis del automuestreador 21
8.1.4 Oportunidades del automuestreador 21
8.2 SELECCIÓN DEL HARDWARE Y SOFTWARE NECESARIOS 21
8.3 INTEGRACIÓN DEL ARDUINO Y DEL PROGRAMA EN C# EN EL ENTORNO .NET FRAMEWORK WINFORMS 21

9. TRABAJO REALIZADO 22

9.1 PRIMERA APROXIMACIÓN 22
9.2 SEGUNDA APROXIMACIÓN 25
9.2.1 Amplificador con ganancia fija 25
9.2.2 Seguidores de tensión 26
9.2.3 Referencia de tensión variable 26
9.2.4 Sumador 26
9.2.5 Versión 5 del programa C#. Interfaz gráfica 26
9.2.6 Versión 5 del programa C#. Flujo de funcionamiento 28

9.2.7 velocidad UART ... *29*
9.2.8 Error de linealidad del Arduino ... *30*
9.3 TERCERA APROXIMACIÓN ... 31
9.3.1 Circuito para interrupción ... *31*
9.3.2 Seguidores de tensión ... *33*
9.3.3 Referencia de tensión variable ... *33*
9.3.4 Filtro paso bajo ... *33*
9.3.5 Modificación en la interfaz ... *33*
9.3.6 Versión 14 del programa C#. Flujo de funcionamiento ... *36*
9.3.6.1 Mejora de la eficiencia del guardado de archivos ... 36
9.3.6.2 Mejoras de la capacidad de lectura de archivos ... 37
9.4 IMPLEMENTACIÓN Y FABRICACIÓN DEL PROTOTIPADO ... 43

10. CONCLUSIONES Y TRABAJO FUTURO. ... 45

11. LISTADO DE MATERIALES Y EQUIPO ... 46

11.1 DESARROLLO ... 46
11.2 EJECUCIÓN ... 46

12. PRESUPUESTO ... 49

12.1 COSTES DIRECTOS ... 49

13. ANEXO A ... 50

13.1 MANUAL DE USUARIO ... 50

DIGITALIZACIÓN Y REACONDICIONAMIENTO DE UN SISTEMA CLÍNICO DE CROMATOGRAFÍA ... 51

A.1. DESCRIPCIÓN DEL SISTEMA ... 54
a. Objeto ... *54*
A.2. INSTALACIÓN ... 54
a. Instalación Hardware ... *54*
b. Instalación Software ... *55*
A.3. DESCRIPCIÓN DEL SISTEMA ... 56
a. Pantalla 1 ... *56*
b. Pantalla 2 ... *57*
c. Mensajes de error ... *58*
d. Ayudas contextuales ... *59*
A.4. MODO DE FUNCIONAMIENTO ... 60
A.5. FAQ ... 61
A.6. MANTENIMIENTO ... 61
e. Cálculo de la no linealidad ... *61*
A.7. GLOSARIO ... 63
A.8. CONTACTO ... 63

14. ANEXO B. CÓDIGO DEL PROGRAMA. ... 64

14.1 FORM1.CS ... 64
14.2 FORM2.CS ... 70

15. REFERENCIAS ... 79

Índice de Figuras

Figura 1 Esquema HPLC [3] 5

Figura 2 Cromatograma de un estándar de calibración 5

Figura 3 Cromatograma de un control de orina con niveles patológicos 6

Figura 4 Cromatograma de un paciente 7

Figura 5 Modo de medición previo 8

Figura 6 Esquemático ADC tipo SAR [4] 10

Figura 7 Ejemplo de operación SAR con 4 bits[4] 11

Figura 8 Ejemplo de comparaciones sucesivas ADC [5] 11

Figura 9 Señal de cromatografía y la aplicación de una función de detección de picos sin tratamiento para el ruido blanco 13

Figura 10 Cambio cualitativo aplicando un filtro de media móvil 14

Figura 11 Ilustración Voltage swing 14

Figura 12 Sistema final 17

Figura 13 Esquema primera aproximación 22

Figura 14 Muestra sin tratar primera aproximación 23

Figura 15 Comparativa señal sin tratar y tratada con filtro de media móvil orden 800 24

Figura 16 Aplicación de un filtro de media móvil de un orden excesivo 24

Figura 17 Esquema segunda aproximación 25

Figura 18 Interfaz de programa Versión 5 27

Figura 19 Cuadro de diálogo de la opción de leer Versión 5 27

Figura 20 Ejemplo de representación de una muestra clínica con la opción leer 28

Figura 21 Diagrama de flujo versión 5 del programa C# 29

Figura 22 Tiempo entre transmisiones 29

Figura 23 Error medición Arduino 30

Figura 24 Esquema tercera aproximación 31

Figura 25 Circuito de la señal de salida del automuestreador[8] 32

Figura 26 Duración señal automuestreador 32

Figura 27 Interfaz de programa Versión 14 34
Figura 28 Cuadro de diálogo de la opción de leer. Versión 14 34
Figura 29 Ejemplo de representación de una muestra clínica con la opción leer. versión 14 35
Figura 30 Diagrama de flujo de la versión 14 del programa C# 36
Figura 31 Visualización detección de picos 37
Figura 32 Resta de la información no necesaria 38
Figura 33 Función con marcadores personalizados 39
Figura 34 Función con marcadores personalizados y los de la función de cálculo de picos 40
Figura 35 Ejemplo de página usando *PDF todo* 41
Figura 36 Ejemplo de una página usando *PDF siguiente* 42
Figura 37 Cara superior de la placa de prototipado 43
Figura 38 Cara inferior de la placa de prototipado 44
Figura 39 Caja estanca con taladro 44
Figura 40 Diagrama completo con cables a conectar marcados 54
Figura 41 Conexión circuito con módulo analógico 55
Figura 42 pantalla 1 56
Figura 43 Ejemplo de búsqueda de puerto COM 57
Figura 44 Pantalla 2 57
Figura 45 Error Arduino no conectado 58
Figura 46 Diagrama de flujo del programa completo 59
Figura 47 Modificación de los parámetros del cálculo de picos 60
Figura 48 Conexionado para mantenimiento 62

Índice de ecuaciones

Ecuación 1 6

Ecuación 2 6

Ecuación 3 11

Ecuación 4 12

Ecuación 5 23

Ecuación 6 32

Ecuación 7 33

Ecuación 8 38

Índice de tablas

Tabla 1 Tiempos de retención y concentración de distintas sustancias 7

Tabla 2 Equipo y Software necesario para desarrollo 46

Tabla 3 Equipo y Software necesario para ejecución 46

Tabla 4 Listado de componentes del circuito 48

Tabla 5 Costes directos 49

Tabla 6 Ejemplo de tabla de valores 61

1. Resumen

La incorporación de una máquina moderna para automatizar un proceso presenta una oportunidad para, a su vez, hacer una digitalización de todo el proceso. Este hecho repercutirá en una oportunidad de analizar los datos y extraer la información necesaria de una forma mucho más eficaz.

En este Trabajo de Fin de Grado se digitalizará y reacondicionará un cromatógrafo clínico. Para ello, se realizará el diseño y la fabricación de un circuito electrónico que lleve a cabo el proceso de conversión analógico-digital y un programa que reciba los datos y los procese.

2. Abstract

The addition of a modern machine to automate a process presents an opportunity to, in turn, digitalize the entire process. This will result in an opportunity to analyze the data and extract the necessary information in a much more efficient way.

In this Final Degree Project, a clinical chromatograph will be digitized and reconditioned. To do this, an electronic circuit will be designed and manufactured to carry out the analogue-digital conversion process and a program to receive and process the data.

3. Agradecimientos

En primer lugar, quiero agradecer a mis padres por su apoyo incondicional durante toda mi carrera universitaria. Sin su amor, aliento y sacrificio, nunca habría llegado hasta aquí. Su constante apoyo ha sido fundamental para mi éxito académico y personal.

También quiero agradecer a mis amigos y compañeros de la carrera por su amistad, su ayuda y su motivación en momentos difíciles. Sus conversaciones, bromas y experiencias compartidas han sido una fuente constante de inspiración para mí.

A mi novia Lara, le agradezco profundamente su amor, comprensión y paciencia durante todo este tiempo. Su cariño y su apoyo incondicional me han dado la fuerza y la motivación para seguir adelante, incluso en los momentos más difíciles.

Agradezco también a mi tutor Juan Manuel Miguel Jiménez por su sabiduría, guía y compromiso en el desarrollo de este proyecto. Muchas gracias por darme la oportunidad de desarrollar este proyecto, ha sido desafiante y sin duda me ha hecho mejorar y desarrollarme como profesional.

Finalmente, quiero agradecer a la Universidad de Alcalá por brindarme la oportunidad de formarme en una institución tan prestigiosa. La calidad de la educación que recibí y el ambiente de aprendizaje que encontré aquí han sido fundamentales para mi desarrollo académico y personal.

A todas estas personas, quiero decirles que su apoyo y su cariño han sido un regalo invaluable para mí. Nunca olvidaré sus contribuciones a mi vida y a mi carrera, y siempre estaré agradecido por todo lo que han hecho por mí.

4. Resumen extendido

4.1 Fundamento

Hoy en día todavía hay numerosos procesos que se realizan con máquinas puramente analógicas, este hecho es más visible en sectores dónde las condiciones económicas no permiten llevar a cabo una transformación digital o no había suficientes incentivos. Recientemente con los avances en inteligencia artificial y big data, la transformación digital debe realizarse allá donde sea posible. Además, la adquisición de una máquina nueva para la automatización del proceso y permitir así que los trabajadores del laboratorio puedan invertir su tiempo en otras cosas repercutirá en una mejora sustancial de la productividad del laboratorio.

El proceso en cuestión tiene como finalidad medir la cantidad de metanefrinas en una muestra. Esto es especialmente útil y es un requerimiento fundamental para llevar a cabo intervenciones quirúrgicas en zonas próximas a los riñones para pacientes con hipertensión. Para entender esta relación procederé a explicar el origen de los peligros que se quieren detectar.

Las glándulas suprarrenales se sitúan en la parte superior de los riñones y son las encargadas de producir distintas hormonas para regular diversas funciones. Este trabajo se centrará en un único grupo, las **catecolaminas**, este grupo de hormonas se encarga principalmente de regular la presión sanguínea, un miembro conocido de este grupo es la adrenalina. Las células dentro de la glándula suprarrenal encargadas de producir este tipo de hormonas son las células suprarrenales/cromafines, cuándo estas mutan y producen **tumores neuroendocrinos** que se acumulan cerca de vasos sanguíneos en forma de quistes. Esto aumentaría levemente los niveles de dichas hormonas de forma constante, cosa que se usará para detectar dichos tumores. Pero este no es el problema en una intervención quirúrgica, el principal problema es que, si se desconoce la existencia de dichos tumores y de forma inintencionada se dañan, éstos verterán una gran cantidad de catecolaminas en el torrente sanguíneo, hecho que producirá casi con total seguridad una parada cardiorrespiratoria en un paciente hipertenso.

4.2 Principio de medición

El principio de medición usado para obtener las concentraciones de los metabolitos de las catecolaminas es la **cromatografía** que permite la separación, identificación, cuantificación, cualificación y determinación de componentes químicos individuales en una mezcla. Para efectuar la separación se usará el método **HPLC** (High-Performance Liquid-Chromatography)[1].

Dicho método consiste en el uso de una fase móvil de base de sílice no polar que está constantemente circulando por el sistema cuya circulación está producida por una bomba de alta presión[2]. Entonces, se procede a inyectar una muestra previamente tratada y al cabo de un tiempo determinado por la polaridad de cada sustancia, pasará por la columna de absorción y posteriormente al detector. De esta forma dependiendo del orden de llegada de cada sustancia se podrá determinar el tipo de sustancia en la gráfica. En la Figura 1 se puede ver el esquema total.

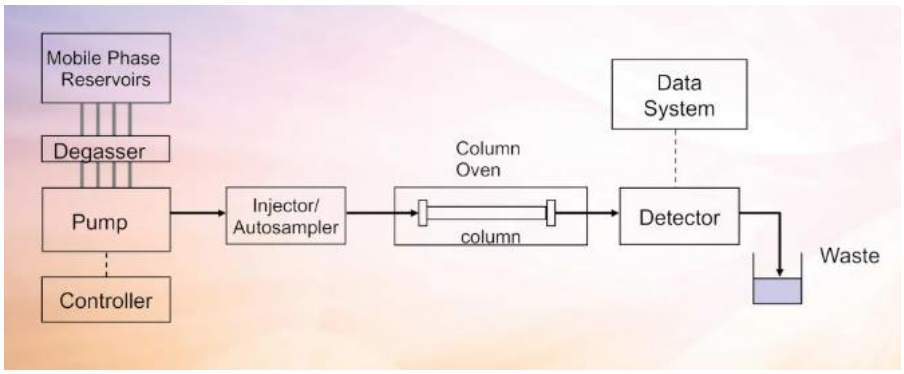

Figura 1 Esquema HPLC [3]

4.3 Metodología de la medición

Antes de introducir la muestra de un paciente en el cromatógrafo, se procederá a introducir un estándar de calibración como el representado en la Figura 2, el cual proporcionará información del estado de la fase móvil y los **tiempos de retención** de cada sustancia. Además, el estándar será usado para la obtención de la concentración de cada sustancia en cada muestra de cada paciente.

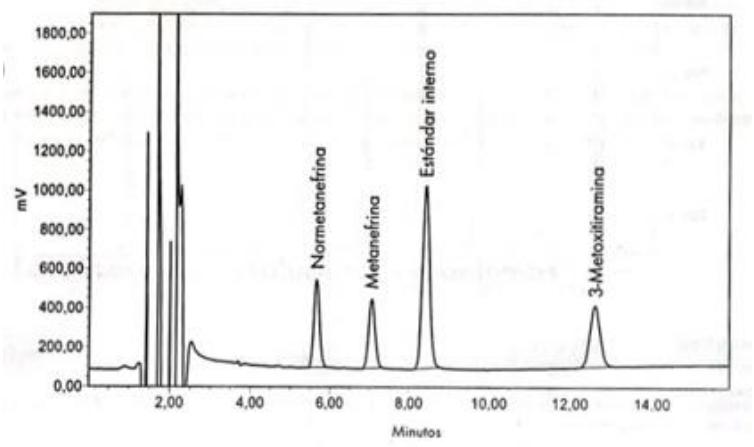

Figura 2 Cromatograma de un estándar de calibración

Como se puede ver en la Figura 2 en el propio estándar hay un **Estándar interno**, este estándar será una referencia para calcular los factores de calibración. Es fundamental no confundirlo con el **estándar de calibración,** al igual que todas las muestras contiene un estándar interno.

El procedimiento para el cálculo de la concentración es el siguiente:

1. Introducir estándar de calibración, y una vez imprimida la gráfica por la impresora del cromatógrafo, medir con una regla la altura de cada pico ($h_M, h_{NM}, h_{MX}, h_{Estándar\ interno}$). Posteriormente se dividirán las alturas de cada curva de cada metabolito entre la altura del Estándar interno y sobre una constante, representado en la Ecuación 1.

$$F_{M\ o\ NM\ o\ MX} = \frac{h_{M\ o\ NM\ o\ MX}}{h_{Estándar\ interno}} * \frac{1}{cte_{M\ o\ NM\ o\ MX}}$$

Ecuación 1

De la Ecuación 1 se obtienen los factores estandarizados que usarán para el cálculo de la concentración de cada sustancia F_M(Factor Metanefrina), F_{NM}(Factor Normetanefrina) y F_{MX}(Factor 3-Metoxitiramina).

2. Si los niveles del estándar son erráticos se procederá a introducir muestras de control con niveles altos como en la Figura 3 y/o un control con niveles bajos con tal de comprobar la calidad de la medición.

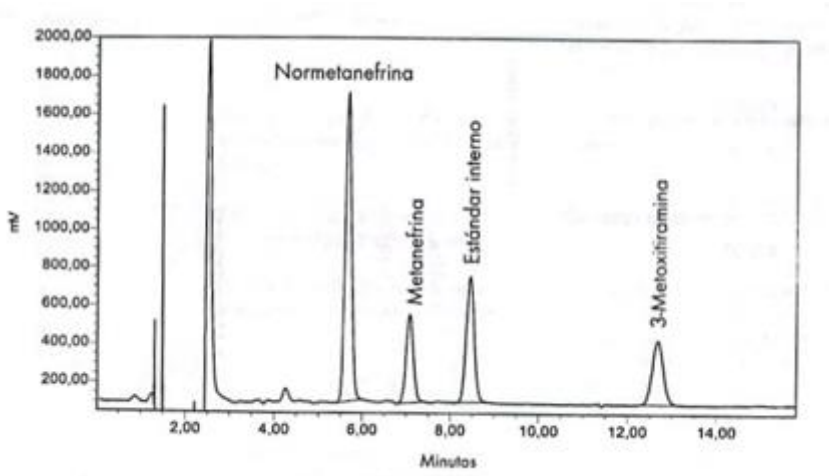

Figura 3 Cromatograma de un control de orina con niveles patológicos

3. Si los niveles del estándar y los de los controles son normales se procederá a introducir la muestra del paciente, que debido a distintas impurezas en la muestra y/o en la fase móvil tendrá una forma similar a la Figura 4. Para obtener la concentración de la muestra de cada paciente, medir con una regla la altura de cada pico, usar los factores de calibración previamente calculados con el estándar e introducirlos en la Ecuación 2 :

$$C_{M\ o\ NM\ o\ MX}\left[\frac{\mu g}{L}\right] = \frac{h_{M\ o\ NM\ o\ MX}}{h_{Estándar\ interno}} * \frac{1}{F_{M\ o\ NM\ o\ MX}} * \frac{1}{Volumen\ de\ la\ muestra[Litros]}$$

Ecuación 2

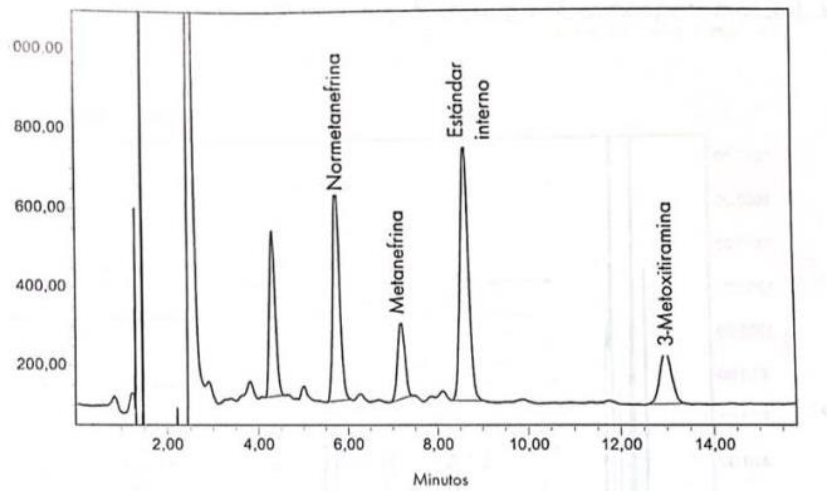

Figura 4 Cromatograma de un paciente

4. Comparación de la concentración obtenida y comprobar si están dentro del rango de normalidad. Si se calculan los niveles de concentración del estándar introducido se deberían obtener las concentraciones y tiempos de retención representados en la Tabla 1.

Tabla 1 Tiempos de retención y concentración de distintas sustancias

Analito	**Tiempos de retención[min]**	**Concentración** $\left[\frac{\mu g}{L}\right]$
Normetanefrina (NM)	5,5	208-312
Metanefrina(M)	7,0	109-164
Estándar Interno	8,4	1
3-Metoxitiramina (MX)	12,5	89-150

La medición hasta ahora se ha realizado siguiendo el esquema mostrado en la Figura 5. Este procedimiento, aparte de ser completamente analógico presenta las siguientes deficiencias:

- Es necesario inyectar manualmente cada muestra, haciendo prescindir al personal de laboratorio de su valioso tiempo.
- Impresión directa en papel provocando una nula digitalización y, por tanto, la imposibilidad de la formación de una base de datos digital que permita un tratamiento de la información útil, por no hablar del espacio que ocupan físicamente los resultados.
- Método de medición relativamente impreciso ya que hay que usar una regla y medir sobre el papel.

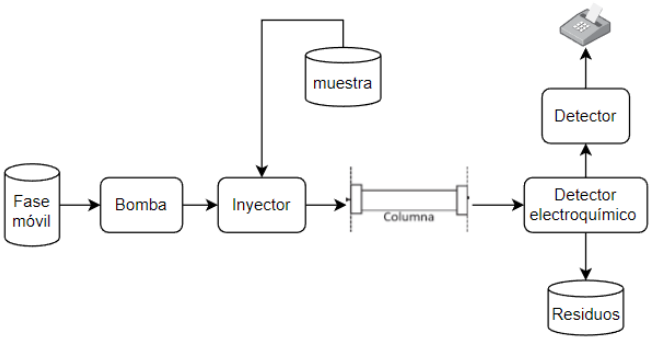

Figura 5 Modo de medición previo

En este Trabajo de Fin de Grado se añadirá un automuestreador, se sustituirá la impresora matricial que actúa a su vez de circuito que trata la señal y se realizará la transformación digital de todo el proceso.

5. Glosario de acrónimos y abreviaturas.

- **HPLC:** High-performance liquid chromatography
- **NM:** Normetanefrina
- **M**: Metanefrina.
- **MX:** 3-metoxitiramina
- **ADC:** Analog to Digital Converter.
- **DAC:** Digital to Analog Converter
- **SAR:** Successive-Approximation Register.
- **SPAN:** Es el valor de la resta aritmética entre los limites superior e inferior dentro del rango de medición de una variable.

6. Estudio teórico

6.1 Hardware

6.1.1 ADC-SAR

Los sistemas de conversión Analógico-Digital, de ahora en adelante se abreviarán como ADC (Analog to Digital Converter), han sido ampliamente estudiados. En este proyecto se usará el ADC integrado en el Arduino nano. Se trata de un ADC tipo SAR (Successive-Approximation Register), cuyo funcionamiento radica en comparaciones sucesivas de la señal a la entrada del ADC con la señal en un DAC interno controlada a su vez por el registro SAR. En la Figura 6 se puede observar el esquemático completo del ADC integrado en el Arduino nano.

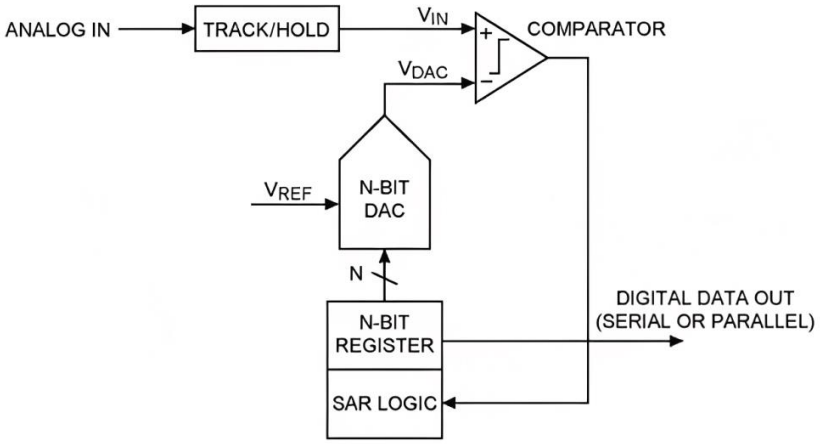

Figura 6 Esquemático ADC tipo SAR [4]

En el Arduino nano se pueden seleccionar 2 voltajes de referencia internos:5 voltios y 1,1 voltios. Además, si se requieren otros voltajes de referencia, pueden ser introducirlos por el pin Vref. Para entender mejor el funcionamiento de los ADC-SAR, en la Figura 7 se puede observar como para un ADC de 4 bits se parte siempre de Vref/2 debido a que los bits del DAC comienzan todos a 0 menos el más significativo. Posteriormente se efectúa la comparación y si la tensión a la entrada del ADC es superior a la fijada por el DAC entonces el bit permanece a 1, en caso contrario se cambia a 0. Este proceso se repite sucesivamente con el siguiente bit más significativo.

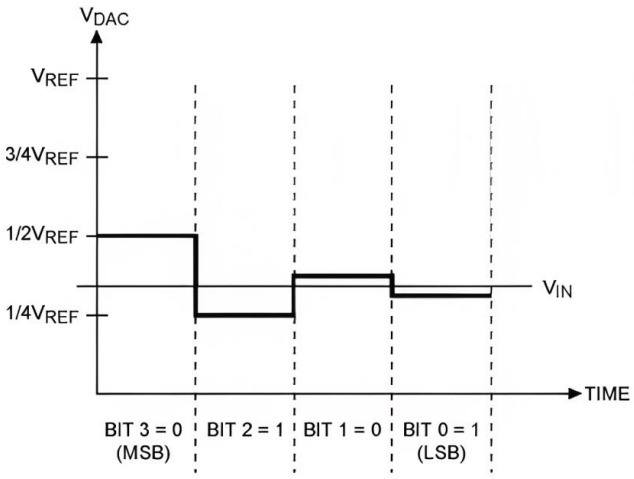

Figura 7 Ejemplo de operación SAR con 4 bits[4]

El ADC del Arduino nano tiene 10 bits de resolución y el ciclo de conversión es de 13 pulsos de reloj. Desglosando el uso de ciclo de reloj se sabe que el primer pulso[5] separa la entrada al ADC y mantiene el valor de voltaje. Los siguientes 11 pulsos son propios del SAR para recorrer los 10 bits del DAC, la razón por la que son 11 pulsos y no 10 es porque el último es usado para aplicar correcciones con el comparador. Y el último pulso almacena el dato leído.

Operation of 10bit successive approximation ADC

nbits	10
Vin	3.597
Vref	5.000

Input Vl	Test data value N	Clock Cycle	change	DAC Vt	Compare k	Test value binary
3.597	512	1	512	2.5000	0	100000 0000
3.597	768	2	256	3.7500	1	110000 0000
3.597	640	3	128	3.1250	0	101000 0000
3.597	704	4	64	3.4375	0	101100 0000
3.597	736	5	32	3.5938	0	101110 0000
3.597	752	6	16	3.6719	1	101111 0000
3.597	744	7	8	3.6328	1	101110 1000
3.597	740	8	4	3.6133	1	101110 0100
3.597	738	9	2	3.6035	1	101110 0010
3.597	737	10	1	3.5986	1	101110 0001
RESULT	736	11		3.5938		101110 0000

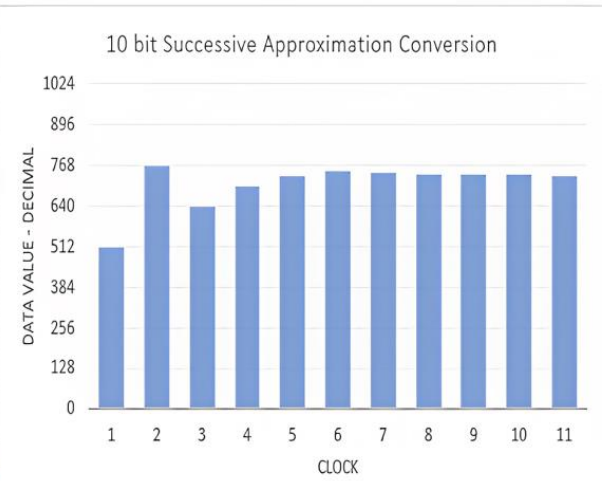

Figura 8 Ejemplo de comparaciones sucesivas ADC [5]

La resolución del ADC dependerá del voltaje de referencia. Un menor voltaje de referencia proporcionará una mejor resolución, pero será un factor limitante en la tensión máxima que será capaz de leer. En la Ecuación 3 se representa la resolución usando un voltaje de referencia de 5 Voltios mientras que en la Ecuación 4 se puede observar la resolución con un voltaje de referencia de 1,1 Voltios.

$$Resolución|_{Vref=5V} = \frac{5V}{1024} = 0{,}00488\,V$$

Ecuación 3

$$Resolución|_{Vref=1,1V} = \frac{1,1V}{1024} = 0,00107\,V$$

Ecuación 4

6.1.2 Tratamiento de la señal

Las señales eléctricas en sistemas de cromatografías están sujetas a distintas clases de ruidos e interferencias que dificultan la identificación de cada pico y, por tanto, el cálculo de su área. Entre los problemas a la hora de tratar la señal se puede encontrar:

- Ruido blanco, asociado prácticamente a cualquier medio de transmisión de señales electromagnéticas. De no ser tratado, puede hacer imposible la implementación de la función de detección de picos. Como se puede ver en la Figura 9, se generan una gran cantidad de falsos positivos, haciendo prácticamente imposible la lectura correcta de la muestra.

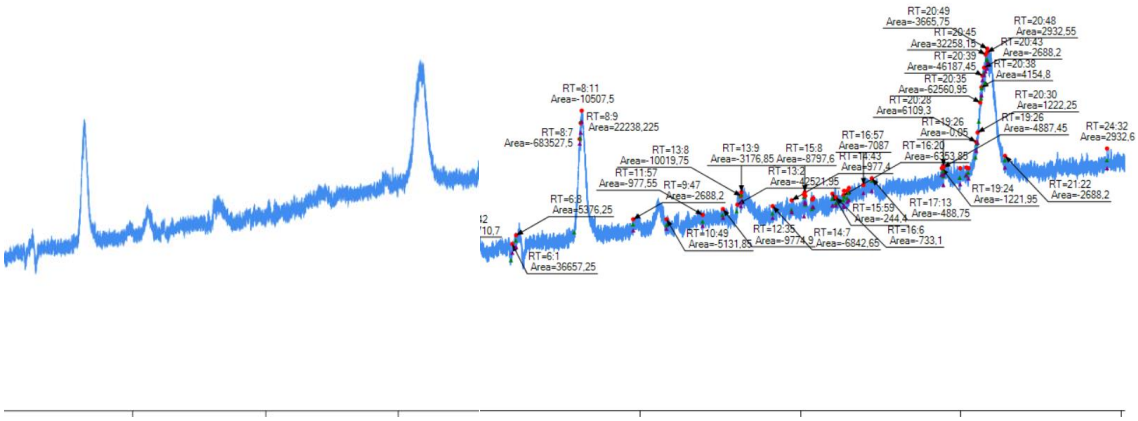

Figura 9 Señal de cromatografía y la aplicación de una función de detección de picos sin tratamiento para el ruido blanco

Teniendo en cuenta las características del ruido blanco, se llega a la conclusión de que, si se suman todas sus componentes, éstas se anulan por lo que aplicando un filtro de media móvil[6] se obtendrá una mejor relación señal ruido.

Un filtro de media móvil de orden N consiste en la suma de las N-1 anteriores muestras con la muestra actual, dividido entre N. La determinación de N es proporcional al periodo de muestreo de la señal y su valor debe ajustarse de forma empírica. Un valor muy alto provocará el aplanamiento y desplazamiento de los picos mientras que un valor muy pequeño no repercutirá en una mejora de la relación señal ruido. En la Figura 10 se puede apreciar los resultados aplicando un filtro mediador de orden 200.

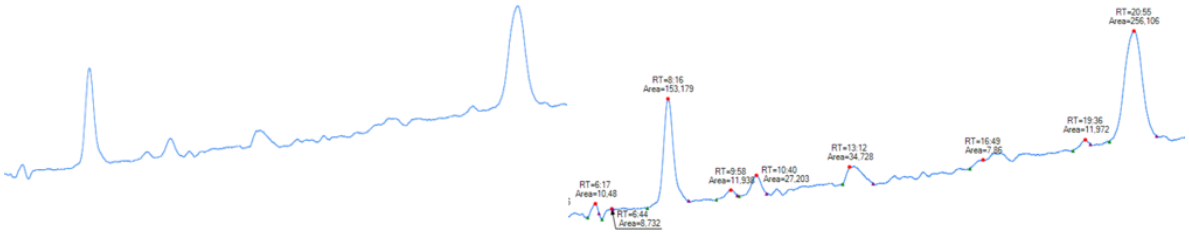

Figura 10 Cambio cualitativo aplicando un filtro de media móvil

- Impurezas químicas de la analítica, son picos pequeños que no corresponden a impurezas en la muestra química que se quiere medir. Para paliar su efecto en la función de detección de picos se podrán modificar distintos parámetros.
- Línea base con pendiente. La corrección de la línea base requeriría añadir complejidad al circuito de acondicionamiento y por técnicas en el software se pueden paliar sus efectos.

6.1.3 Amplificador Operacional "*Rail-to-Rail*"

Los amplificadores operacionales de propósito general requieren de una alimentación simétrica para tener unas especificaciones viables. Como amplificador genérico se usará de ejemplo el **LM358**, un popular amplificador operacional de propósito general. En su hoja de características se puede encontrar un "***Voltage output swing from rail*"** de *1,4V* al "***Positive rail (V+)***" y 0,75V al "***Negative rail (V–)***" a 1mA de salida, esto quiere decir que alimentando con 4V el máximo voltaje que se obtiene a su salida será de 2,6V, lo cual es un factor limitante a tener en cuenta. En la Figura 11 se puede observar la variable *Voltage Swing* gráficamente sobre un amplificador operacional como seguidor de tensión.

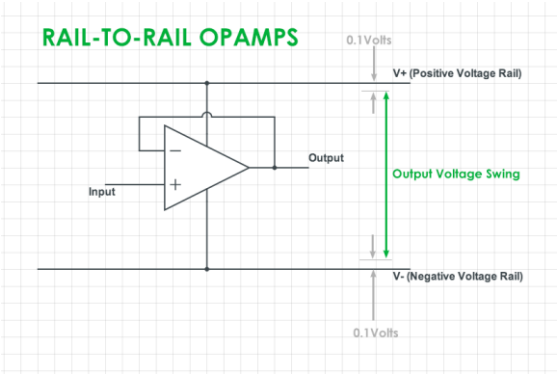

Figura 11 Ilustración Voltage swing

Por tanto, para aplicaciones con una tensión de alimentación relativamente baja y necesitando que la tensión de salida del amplificador pueda alcanzar valores próximos a la tensión de alimentación, lo ideal es usar un amplificador operacional rail-to-rail [7].

6.2 Software

Respecto al Software, se usarán 2 plataformas de desarrollo distintas y 2 lenguajes de programación distintos. La exclusión del uso de Matlab de este proyecto se debe a su elevado coste y a que se busca minimizar los costes.

6.2.1 Entornos de Desarrollo

6.2.1.1 Microsoft Visual Studio 2022

Es un entorno de desarrollo gratuito compatible con múltiples lenguajes de programación, como C++, C# y Python. Además, la interfaz es intuitiva y permite añadir extensiones que ayuden en el desarrollo del código.

6.2.1.2 Arduino IDE

Comparado con Microsoft Visual Studio 2022, es un entorno de desarrollo muy sencillo. No posee algunas funcionalidades básicas, como detectar errores en el código sin que haga falta compilar o una herramienta de depuración. Una vez compilado el código sin errores, éste podrá ser cargado de forma sencilla a la tarjeta vía conexión USB, pero será complejo comprobar el correcto funcionamiento del sistema sin depuración.

6.2.2 Lenguajes de programación

6.2.2.1 C#

C# es un lenguaje de alto nivel orientado a objetos cuyo propósito en su creación fue hacer un lenguaje simple de implementar. El coste de permitir programar de forma más sencilla y rápida tiene un precio: la velocidad de ejecución y el tamaño de los archivos.

Una cualidad que hace que la programación con C# sea más rápida es su recolector de basura o en inglés *garbage collector*. Este término se usa para referirse a un gestor automático de memoria. Este gestor se encargará automáticamente de reservar y liberar memoria dinámica y, por ende, de gestionar también los punteros, que son la principal fuente de error en distintos programas.

Como el sistema operativo final es Windows, se procederá a usar el *Framework* .NET y el conjunto de librerías Winforms.

6.2.2.1.1 .NET Framework

.NET Framework es un marco de trabajo desarrollado por Microsoft para el desarrollo de aplicaciones orientadas al sistema operativo Windows. Proporciona librerías y herramientas para ayudar a desarrollar las correspondientes aplicaciones.

6.2.2.1.2 Windows Forms

Windows Forms, es una única librería de clases gráficas de código abierto incluida en .NET Framework. Su objetivo es proporcionar métodos para desarrollar interfaces gráficas para una aplicación determinada.

7. Definición del problema y objetivos

7.1 Definición del problema

El problema al modernizar una parte o proceso en el que intervienen varias máquinas radica en la compatibilidad entre ellas. En muchos casos, si se sustituye una máquina por otra más moderna, esta no es compatible con las otras máquinas existentes en el proceso. Esto se debe a que las máquinas eléctricas modernas a menudo tienen diferentes características técnicas, como voltajes, corrientes, frecuencias y modos de control, lo que puede hacer que no funcionen correctamente en conjunto con las máquinas más antiguas. Además, también puede haber problemas de comunicación entre las máquinas si utilizan diferentes protocolos y tecnologías de red.

Para resolver este problema, se requiere una planificación cuidadosa y un enfoque sistemático para la modernización, que incluya una evaluación exhaustiva de las características técnicas y la compatibilidad de las máquinas eléctricas existentes y nuevas, así como la implementación de medidas para asegurar la comunicación y la interoperabilidad entre ellas. Esto puede incluir la instalación de dispositivos de adaptación o la actualización de los sistemas de control y monitoreo existentes para garantizar una integración efectiva de las máquinas eléctricas modernas en el proceso existente.

En este caso, además de la incorporación de una nueva máquina y de su correspondiente adaptación al resto, se efectuará una transformación digital de los resultados obtenidos. El objetivo principal del proyecto es la transformación digital de la salida del sistema y la implementación del automuestreador. Es decir, pasar del sistema ilustrado en la Figura 5 al ilustrado en la Figura 12.

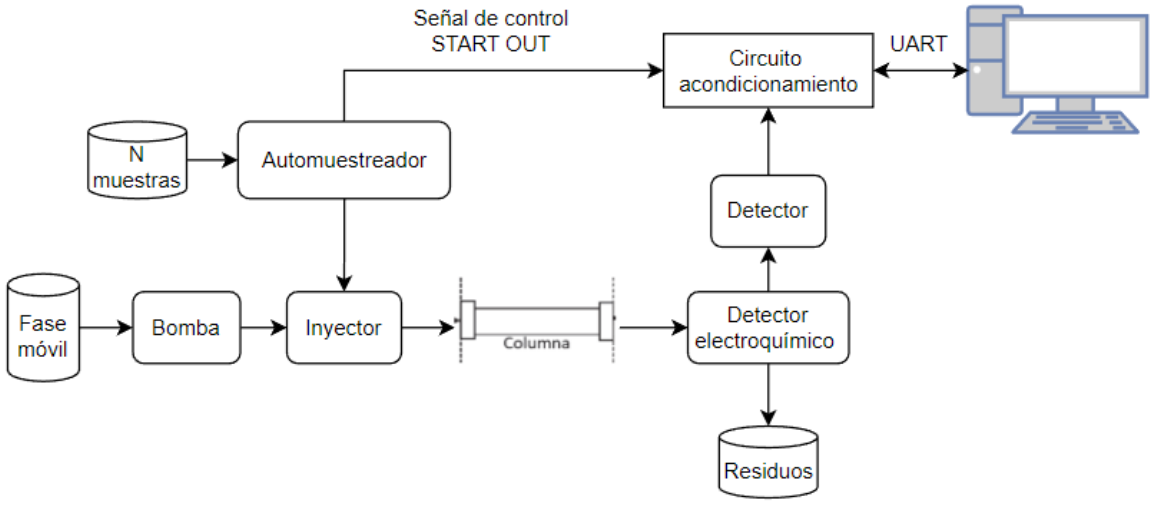

Figura 12 Sistema final

7.2 Objetivos

Los objetivos del proyecto son distintos para cada parte del mismo:

- Análisis de la señal y su entorno:
  - Analizar las señales a recibir, tanto la química como las de control del automuestreador.
- Sustitución de la impresora matricial:

  - Etapa de acondicionamiento de la señal, además de su tratamiento para obtener información.
  - Transmisión de la información obtenida de la señal a un ordenador.
  - Almacenamiento de la señal de forma ordenada e intuitiva que permita la creación de una base de datos para futuros estudios. Además, debe ser capaz de ver en tiempo real el estado del proceso.
- Incorporación de la máquina de automatización:
  - Interpretar las señales de control emitidas por la automuestreadora y enviar la información extraída al ordenador para que sea debidamente procesada.

7.3 Dispositivos Hardware

7.3.1 Arduino Nano

Arduino Nano es una placa de microcontrolador de código abierto que está diseñada para proyectos electrónicos que requieren una pequeña placa de bajo costo y de tamaño compacto. A continuación, se describe la especificación técnica de Arduino Nano:

- Microcontrolador: ATmega328P
- Voltaje de funcionamiento: 5V
- Voltaje de entrada (recomendado): 7-12V
- Voltaje de entrada (límite): 6-20V
- Pines digitales de entrada/salida: 14 (de los cuales 6 proporcionan PWM)
- Pines de entrada analógica: 8
- Corriente de DC por pin de E/S: 40 mA
- Corriente de DC para pin de 3.3V: 50 mA
- Memoria flash: 32 KB (ATmega328P) de los cuales 2 KB se utilizan para el bootloader
- SRAM: 2 KB (ATmega328P)
- EEPROM: 1 KB (ATmega328P)
- Velocidad de reloj: 16 MHz
- Interfaz USB: CH340G
- Dimensiones: 18 mm x 45 mm

La placa también incluye una serie de componentes adicionales, como un regulador de voltaje, un cristal oscilador, un botón de reinicio y una serie de conectores para alimentación, programación y conexión a otros dispositivos electrónicos. Además, la placa es compatible con el software y las librerías de Arduino, lo que facilita su programación para una amplia variedad de proyectos electrónicos.

7.3.2 Amplificador operacional “rail-to-rail” LMV321

El motivo de selección del este amplificador operacional es su capacidad de proporcionar voltajes a su salida próximos a la tensión de alimentación, permitiendo de esta manera que se pueda alimentar a todo el sistema desde el Arduino, usando el voltaje del puerto USB.

7.3.3 PC con Windows que soporte la versión 4.7.2 de .NET Framework

La versión de Windows mínima que soporta dicha versión es Windows 7. El resto de las librerías estarán en la carpeta del programa y será necesario instalar los drivers correspondientes. Además, no será necesario instalar software extra.

7.4 Elementos Software

7.4.1 Entorno de desarrollo

La opción idónea para una aplicación de Windows es Microsoft Visual Studio, una herramienta de desarrollo integrada (IDE) que ofrece a los desarrolladores una amplia gama de características para la creación de aplicaciones para Windows con C#. Visual Studio es un IDE de primera clase que facilita el desarrollo de aplicaciones para Windows al proporcionar un conjunto completo de herramientas de programación y depuración. Los desarrolladores pueden utilizar Visual Studio para escribir, depurar y desplegar sus aplicaciones de forma más eficiente.

Microsoft Visual Studio es una herramienta altamente efectiva y popular para los desarrolladores que desean crear aplicaciones para Windows en C#. Ofrece una amplia gama de características útiles, una integración fácil con otras herramientas de Microsoft, y es muy fácil de usar.

Además, para la programación en Arduino se usará su entorno de desarrollo Arduino IDE, ya que tiene un lenguaje intuitivo y el uso de este entorno es simple.

La información sobre las características de la señal es prácticamente nula, por tanto, la primera etapa de desarrollo consistirá en el muestreo de la señal y en su estudio. Como la función a realizar por el microcontrolador parece ser de adquisición de una señal analógica, la gestión de una señal de control de interrupción y la transmisión de los datos obtenidos, se optará por un Arduino nano.

Respecto al programa que reciba los datos en el ordenador, sin información sobre la plataforma de destino se opta por una extensión de Excel que permite recibir datos por el puerto COM del ordenador. De esta forma se podrá medir con el Arduino y almacenar directamente las variables enviadas desde el Arduino en Excel o en Matlab.

A priori, la planificación puede parecer sencilla: usar el conversor Analógico-Digital del Arduino además de preparar una única interrupción, enviar los datos por puerto serie, recibirlos en el ordenador con la extensión del Excel y calcular el área de cada curva. No hace falta mencionar que desde luego el desarrollo no ha sido tan simple y pese al sinfín de problemas encontrados han sido todos resueltos.

8. Transformación digital

8.1 Análisis de la situación inicial y necesidades del usuario

El análisis de la situación inicial es un aspecto fundamental para el éxito de cualquier proyecto de transformación digital, especialmente en un entorno de laboratorio médico donde se requiere alta precisión y rapidez en la toma de decisiones. En este apartado se pretende identificar la situación inicial en la que se encuentra el laboratorio médico en relación con la impresora matricial obsoleta, y la oportunidad de automatizar y digitalizar el proceso completo.

Para ello, se llevará a cabo un análisis exhaustivo de la situación actual del laboratorio médico, con el objetivo de identificar los principales problemas y limitaciones asociados al proceso en cuestión, así como de las necesidades y requisitos específicos que se esperan.

En primer lugar, se describirá detalladamente la situación actual del laboratorio médico, incluyendo la descripción de los procesos de impresión y la herramienta que actualmente se utiliza para ello. Se identificarán los problemas y limitaciones asociados a la impresora matricial obsoleta, tales como la baja velocidad de impresión, la baja calidad de la impresión, el alto costo de mantenimiento y la dificultad para encontrar repuestos.

Posteriormente, se identificarán las necesidades y requisitos específicos que el laboratorio médico espera de la nueva máquina automuestreadora. Estos requisitos pueden incluir aspectos como la velocidad de inyección, la calidad de la muestra, el costo de mantenimiento, la facilidad de uso, la integración con otros sistemas del laboratorio, entre otros.

Para ello, se llevarán a cabo entrevistas, reuniones y pruebas con los usuarios finales del laboratorio médico, con el objetivo de obtener una comprensión detallada de sus necesidades y requisitos específicos en cuanto a la impresión. Además, se analizarán las regulaciones y normativas que deben cumplir las máquinas que se utilicen en el laboratorio médico.

8.1.1 Análisis del uso de la impresora matricial

En una primera impresión, la impresora matricial imprimía el cromatograma y de este se extraía la información por un método poco moderno. Consiste en medir con una regla la altura de los picos, obtener unas relaciones y calcular la concentración. Más tarde, conforme se fueron haciendo pruebas, se descubriría que la impresora matricial no sólo lee la información del módulo analógico, sino que además la modifica. Esto se pudo entender gracias a un parámetro dentro del mismo aparato que regula la ganancia de la señal. A priori se pensaba que extraía la información de la señal y aplicaba la atenuación a la información, pero en una prueba midiendo con el dispositivo final en paralelo se pudo observar esta característica.

Además, la impresora matricial requiere de vigilar de forma constante el suministro de papel y, de ocupar espacio de almacenamiento para su posterior archivo. Otro problema que puede dar es el SPAN, no es excesivamente grande y obliga a tener que repetir algunas muestras y estar ajustando el parámetro de atenuación constantemente para que la señal esté dentro de los límites de medida.

8.1.2 Oportunidades de mejora con la digitalización de la impresora matricial

La digitalización de este módulo no sólo excluye el uso del papel, sino que además permite un análisis de datos digital permitiendo prescindir del uso de una regla que además era un método poco preciso. El programa además permitirá un cálculo computacional rápido y más exacto.

El Arduino tiene un SPAN variable que es muy superior al de la impresora matricial que además es variable y lo ajusta automáticamente para una mayor resolución.

8.1.3 Análisis del automuestreador

El automuestrador permite la automatización del proceso. Hasta el momento, el modo de operación era pinchar las muestras de una en una. Esto obligaba al doctor a tener que estar cada 15-20 minutos retirando el papel, calculando la concentración e inyectando una nueva muestra.

Pese a que la impresora matricial podría trabajar con el automuestreador. Esta combinación daría muchos errores. En primer lugar, la impresora matricial no tiene forma de saber cuándo se inyecta una nueva muestra por lo que sólo imprimiría indefinidamente mientras se van introduciendo muestras, impidiendo así un seguimiento del tiempo de aparición de cada sustancia, lo cual es fundamental en el análisis del cromatograma. En segundo lugar, la línea base, aumenta de forma constante. Teniendo en cuenta que el SPAN de la impresora matricial es muy reducido, sin un ajuste constante, lo cual destruye la virtud de la automatización, acabaría por saturar y perder la información de la muestra.

8.1.4 Oportunidades del automuestreador

Con el Arduino además de leer la señal del módulo analógico se puede configurar una entrada de interrupción. De esta forma se puede notificar al programa cuando se inyecte una nueva muestra y reiniciar así la información por la pantalla además de guardar la muestra anterior en la base de datos.

8.2 Selección del hardware y software necesarios

Respecto al Hardware, un Arduino nano tiene la rapidez suficiente para leer el cromatograma teniendo en cuenta que éste suele tardar entre 15 y 25 minutos. Respecto al ordenador, vale cualquiera que permita el uso de mínimo un Windows 7.

En la selección de Software, se prioriza interfaz sobre velocidad. Por lo que una API en C# usando WinForms es perfectamente válida.

8.3 Integración del Arduino y del programa en C# en el entorno .NET Framework WinForms

El Arduino se comunicará con el programa del ordenador a través del USB usando el protocolo UART. La velocidad de transmisión es de 38400 baudios. La velocidad es la adecuada para los datos que se están enviando con fines de depuración. Está limitada superior e inferiormente.

- Superiormente: En el programa, cuando llega un nuevo dato por UART. Se crea un hilo, por tanto, en un ordenador con poca potencia de procesamiento pueden perderse hilos o incluso saturar la memoria RAM en caso de que se creen más hilos que los que se destruyan.
- Inferiormente: El periodo de muestreo se puede variar. Por ejemplo, el usado son 50ms, que es más que suficiente para la señal medida.

9. Trabajo realizado

9.1 Primera aproximación

Los datos proporcionados por el jefe del laboratorio era que la señal iba de 0 a 5 Voltios. Ante tan poca información se optó por montar un circuito lo más seguro posible para no interferir en el dispositivo. Por tanto, se añadió un amplificador operacional en configuración de seguidor de tensión con el fin de separar el ADC del Arduino de la salida del módulo analógico. Además, al no saber la impedancia de salida del módulo analógico o la impedancia de entrada de la impresora matricial, se corre el riesgo de provocar un malfuncionamiento y gracias al amplificador operacional se consigue la capacidad de leer datos en paralelo al funcionamiento normal del equipo.

En la Figura 13 se muestra la configuración y el conexionado del circuito de adquisición. La aplicación usada, al no saber todavía el sistema operativo de destino, es una extensión de Excel que puede comunicarse con el Arduino y almacenar los datos en páginas de Excel.

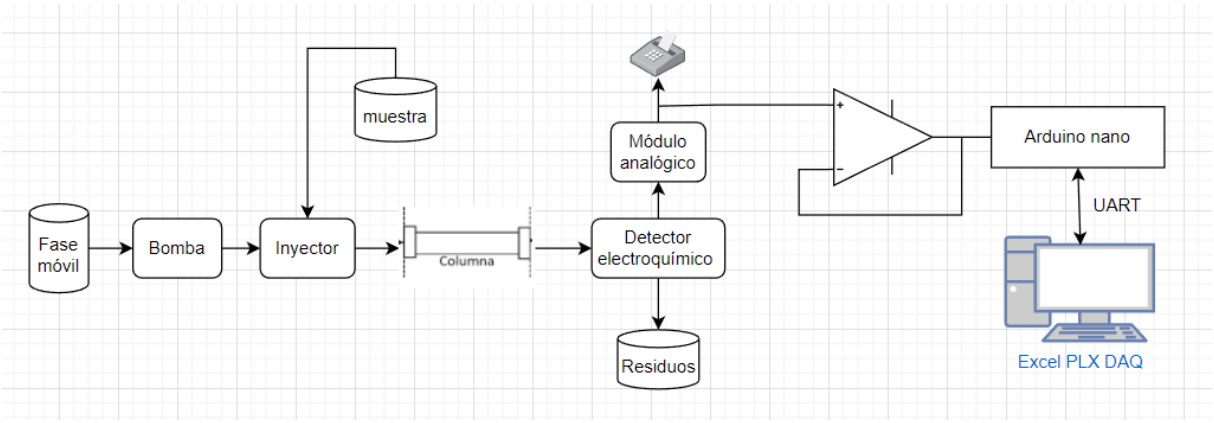

Figura 13 Esquema primera aproximación

El desempeño de esta primera aproximación fue aceptable, pese al elevado nivel de ruido de la señal. Permitió obtener información suficiente de la misma para su estudio y análisis. En la Figura 14 se puede observar una elevada cantidad de ruido y apenas 3 picos. A priori no parece posible extraer ningún tipo de información de la señal, pero más adelante se verá que con el uso de filtros se podrá obtener información de la señal.

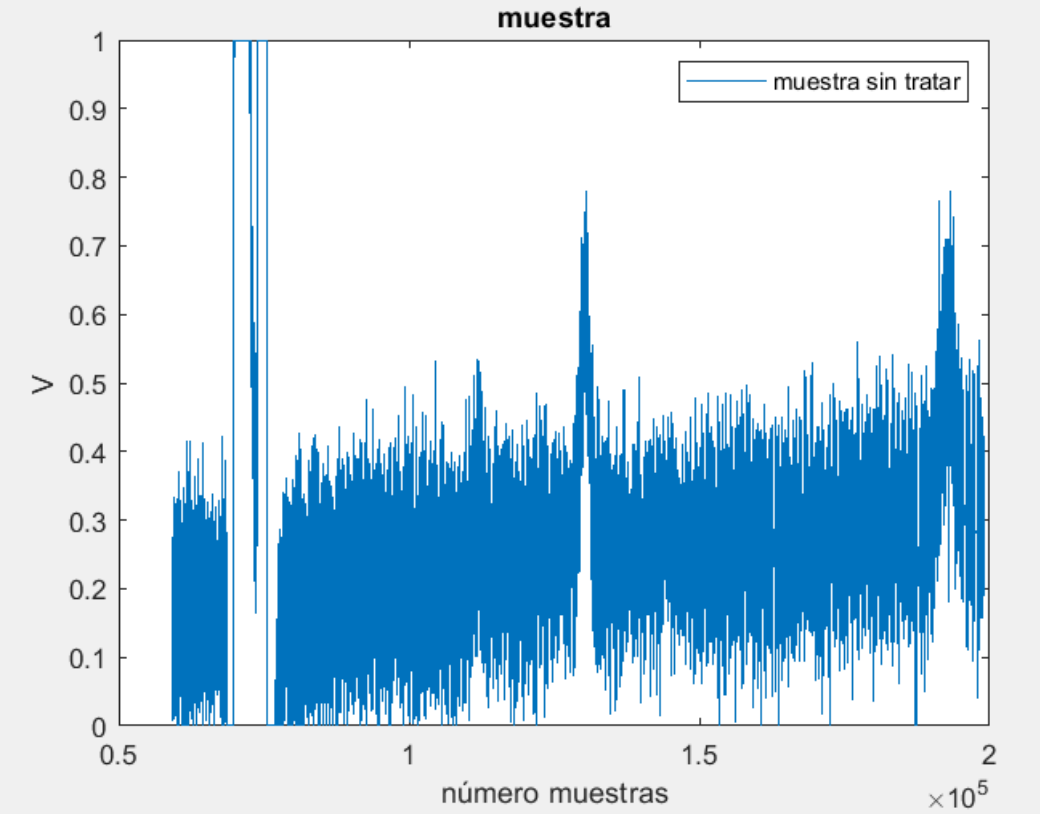

Figura 14 Muestra sin tratar primera aproximación

El filtro de media aritmética móvil es una buena opción para tratar este tipo de señales ya que permite reducir la variabilidad y eliminar los valores extremos, suavizando la señal y haciendo que sea más fácil de interpretar. Además, este filtro es fácil de implementar y puede ser muy eficiente en términos de computación. En la Ecuación 5 se muestra la operación aritmética que ejecuta el filtro. Siendo cada parámetro:

- y[n]: muestra n filtrada.
- N: orden del filtro.

$$y[n] = \frac{1}{N} * \sum_{k=0}^{N-1} y[n-k]$$

Ecuación 5

Es importante tener en cuenta que el filtro de media aritmética móvil puede reducir la resolución temporal de la señal, ya que cada valor filtrado se calcula como una media de un intervalo de tiempo. Esto puede ser un problema si se necesita una alta resolución temporal, pero en muchos casos la reducción de la resolución temporal es un precio razonable a pagar por una señal más limpia y fácil de interpretar. Como se puede observar en la Figura 15, aplicando un filtro de media móvil de orden 800 se reduce considerablemente el nivel de ruido y permite la observación de picos más pequeños. El orden del filtro dependerá del nivel de ruido y la frecuencia de muestreo.

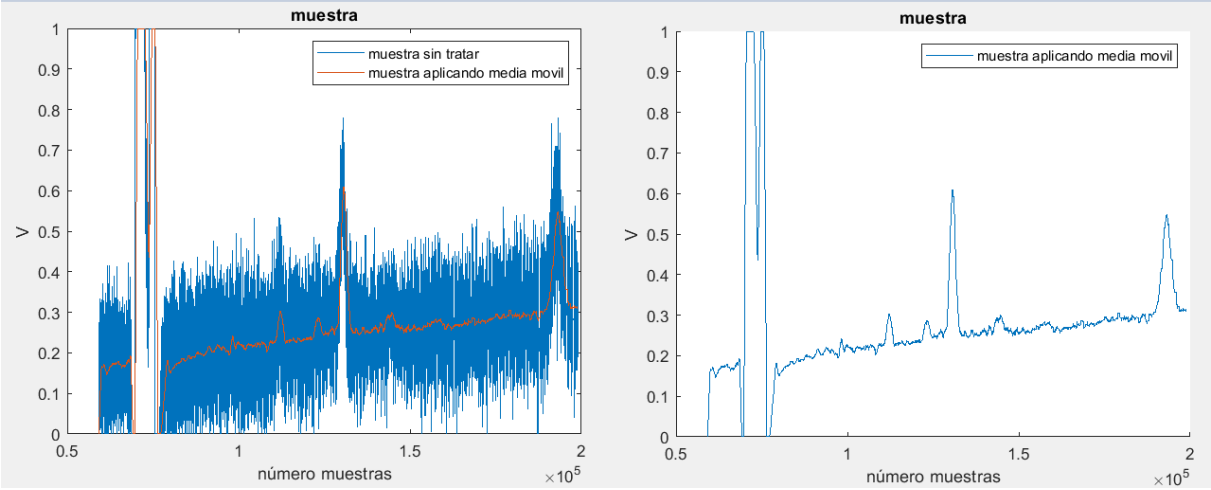

Figura 15 Comparativa señal sin tratar y tratada con filtro de media móvil orden 800

El problema del filtro de media móvil es el aplanamiento de los picos. Un orden demasiado elevado del filtro reducirá la altura y aumentará el ancho del pico llegando a eliminar los picos más pequeños. En la Figura 16 se puede observar el efecto de usar un filtro de un orden muy elevado.

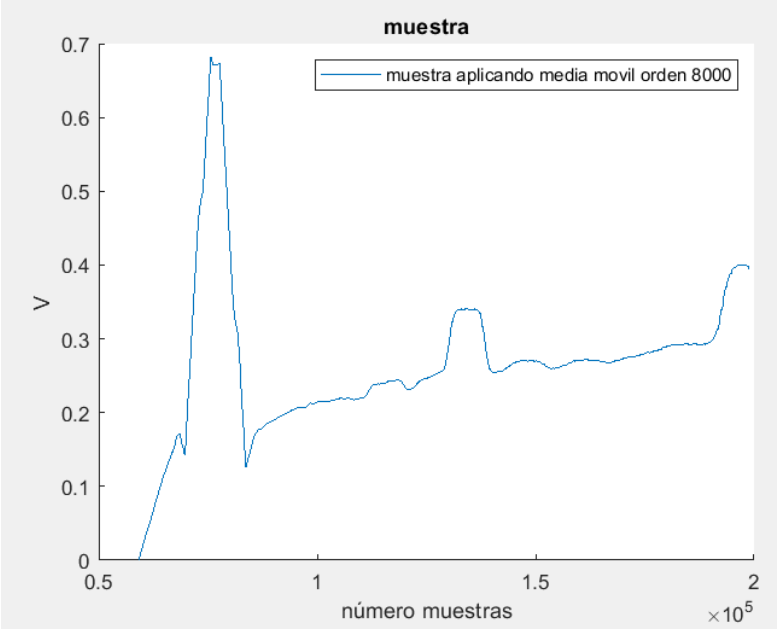

Figura 16 Aplicación de un filtro de media móvil de un orden excesivo

La implementación con Matlab se ha hecho con la función filter() de la forma:

```
filter(ones(1, 8000)*(1/8000) ,1 , muestra_3(:,2));
```

El filtro se implementa mediante la función "filter" de MATLAB, que se utiliza para aplicar un filtro digital a una señal de entrada. La función "ones(1, 8000)*(1/8000)" se utiliza para crear un vector de coeficientes de filtro que representan una media móvil de orden 8000. El valor 1/8000 se utiliza para normalizar los coeficientes de filtro y asegurar que la suma de los coeficientes sea igual a 1, lo que produce una media aritmética móvil.

La función "filter" toma tres argumentos: los coeficientes de filtro, b, en este caso representados por "ones(1, 8000)*(1/8000)"; los coeficientes del filtro de realimentación, a, que en este caso es igual a 1 ya que no se aplica realimentación en este filtro; y la señal de entrada, que se representa por "muestra_3(:,2)".

9.2 Segunda aproximación

En esta aproximación aparte de proporcionar una solución a todos los problemas encontrados. Se añadió al circuito base componentes que permitiesen la modificación de la señal recibida para poder efectuar pruebas y tener flexibilidad a la hora de medir la señal o solucionar cualquier problema. En la Figura 17 se puede ver el esquema completo del bloque de "circuito de acondicionamiento". Se procederá a explicar cada etapa y su motivo.

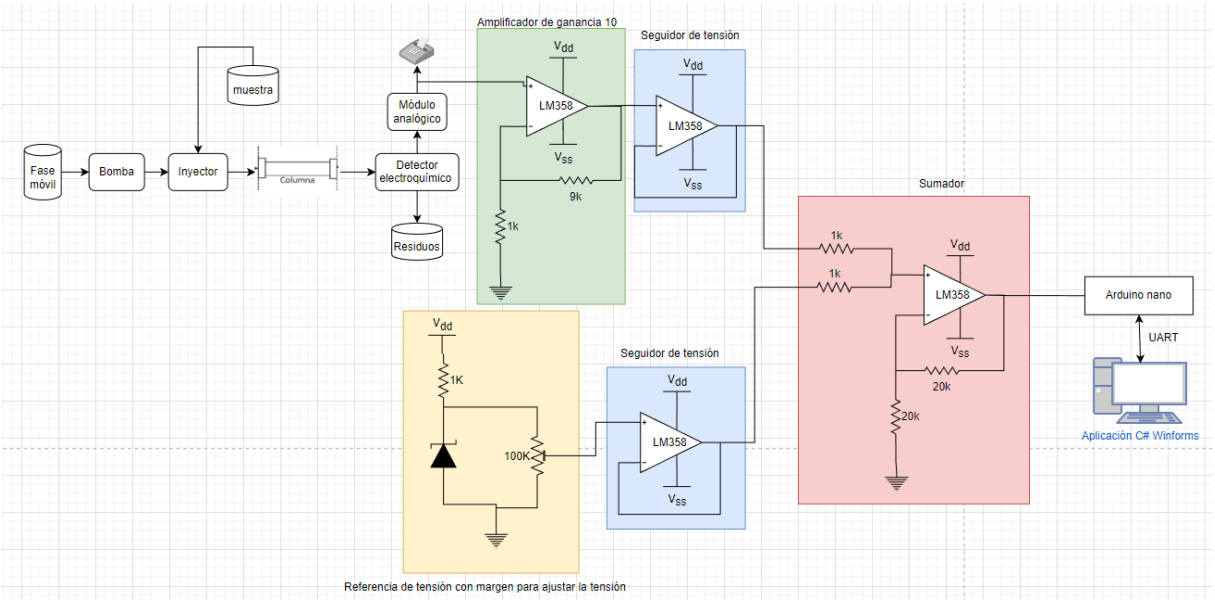

Figura 17 Esquema segunda aproximación

9.2.1 Amplificador con ganancia fija

Con tal de aumentar la resolución de la señal y ajustarla dentro de los valores de SPAN del Arduino Nano, se aplica una ganancia que permite aprovechar mejor el ADC integrado.

9.2.2 Seguidores de tensión

El propósito fundamental de los seguidores es no modificar la ganancia de voltaje y proporcionar suficiente corriente de salida, evitando así sobrecargar la salida de un amplificador, debido al efecto de la baja impedancia de entrada de la siguiente etapa.

9.2.3 Referencia de tensión variable

El rango operativo del Arduino es estrictamente de 0 a 5 Voltios por lo que los valores negativos son interpretados como 0 Voltios. Para solucionar este problema se optó por añadir un voltaje de Offset que se sumaría a la señal.

9.2.4 Sumador

Su propósito es sumar al cromatograma una señal de offset para evitar así la pérdida de información. Además, esta etapa permite calibrar y realizar pruebas en el propio laboratorio para solucionar imprevistos. La ganancia de 2 añade simpleza a la hora de comprobar que la salida es la suma de las dos tensiones de entrada.

9.2.5 Versión 5 del programa C#. Interfaz gráfica

La extensión de Excel que recibía los datos del Arduino Nano resultó ser poco fiable y con capacidad muy limitada. Por lo que era necesario otro método, para desarrollar la aplicación pertinente es necesario saber el sistema operativo de destino. Teniendo en cuenta que los doctores no estaban familiarizados con sistemas Linux y con el fin de proporcionar un interfaz lo más sencillo posible, se seleccionó Windows como sistema operativo final. Entonces en base a esta selección se optó por desarrollar una aplicación en C# usando Winforms, se optó por este lenguaje de programación por su sencillez ya que el objetivo de la aplicación era ser lo más sencilla posible puesto que se tenía en cuenta que el grueso del trabajo fuese la parte Hardware.

La primera interfaz funcional con la opción de leer documentos habilitada fue la versión 5. En esta etapa de diseño había una prioridad funcional con vistas a la depuración, lo cual se ve reflejado en la primera interfaz funcional del programa en la Figura 18 de la cual se procederá a explicar cada elemento numerado a continuación.

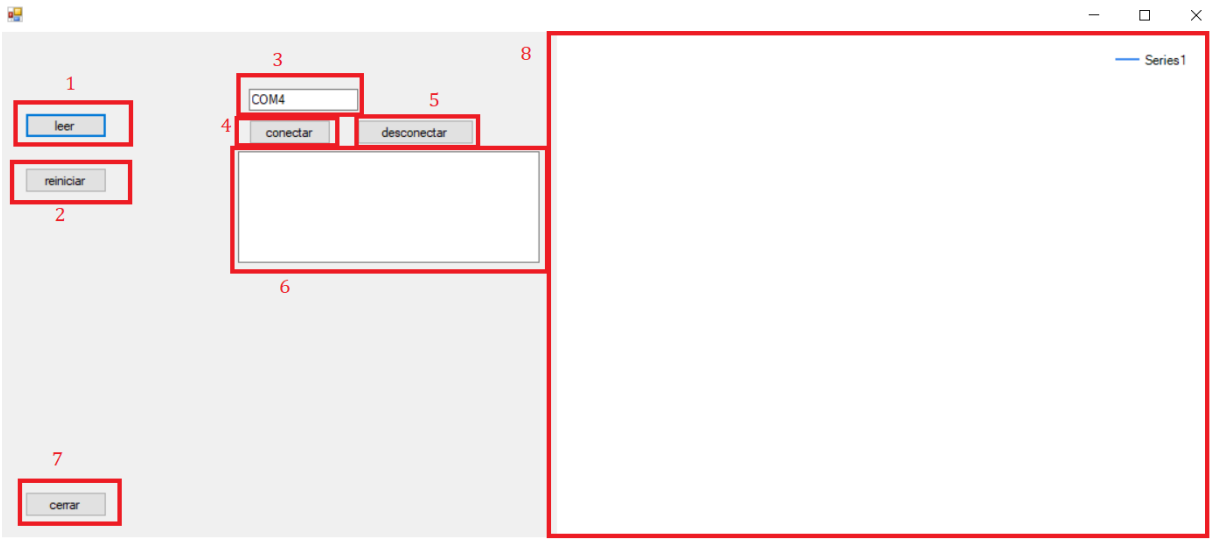

Figura 18 Interfaz de programa Versión 5

Botón leer: permite la lectura de una muestra previamente guardada y la representa en una gráfica en un proceso aparte, de forma que pueda hacerse de forma paralela a la lectura de datos. Como se muestra en la Figura 19, se abre una nueva pestaña y se deberá seleccionar el fichero correspondiente a la muestra que se quiere leer.

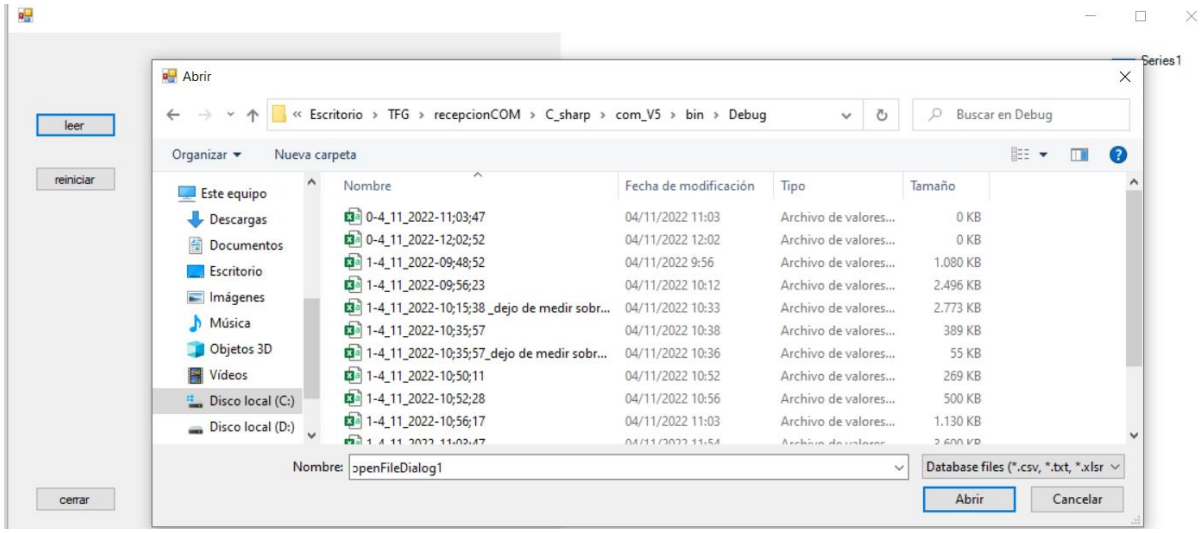

Figura 19 Cuadro de diálogo de la opción de leer Versión 5

Una vez seleccionado el fichero, representará la gráfica correspondiente como la vista en la Figura 20. En el eje x se representará el número de cada muestra tomada por el Arduino y en el eje y, el valor de voltaje multiplicado en el Software por una ganancia. Como en esta etapa de desarrollo todavía no prima el cálculo y análisis del área, el valor de la ganancia fija por software no tiene importancia.

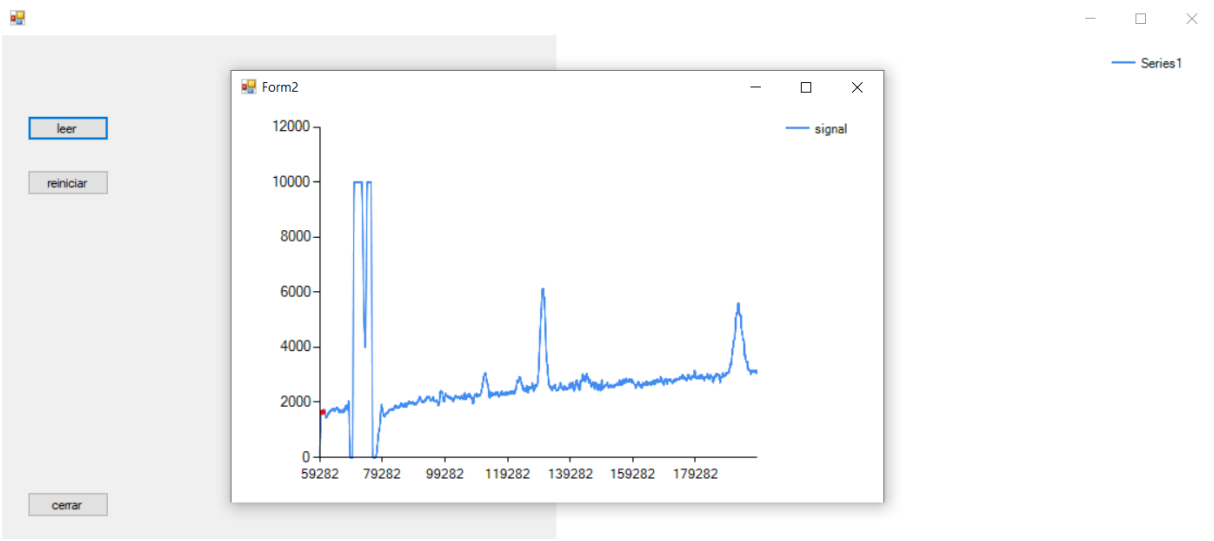

Figura 20 Ejemplo de representación de una muestra clínica con la opción leer

Botón reiniciar: el botón reiniciar envía al Arduino por el puerto UART un mensaje indicando que se reinicie la cuenta del número de muestras. Además, borra los datos que se estuvieran representando y empieza a guardar nuevos.

Cuadro de texto para seleccionar puerto: en este cuadro de texto se introducirá el puerto COM del ordenador al que está conectado el Arduino vía USB.

Botón Conectar: establecerá la conexión con el Arduino conectado al puerto serie seleccionado y si no hay inconvenientes, inmediatamente comenzará la transmisión de datos y la representación gráfica.

Botón Desconectar: Cierra el puerto serie y pone fin a la comunicación por el puerto serie.

Cuadro de texto de datos recibidos: en este cuadro de texto se representarán las cadenas de caracteres recibidas por el Arduino a tiempo real y su fin es principalmente con vistas a la depuración del sistema.

Botón Cerrar: cierra la comunicación por el puerto serie, elimina todos los hilos creados por el programa y finaliza el programa.

Gráfica de datos recibidos a tiempo real: representa a tiempo real los datos recibidos en el mismo formato que el botón leer visto en la Figura 20.

9.2.6 Versión 5 del programa C#. Flujo de funcionamiento

En la Figura 21 se muestra el diagrama de flujo del programa. El principal inconveniente de esta versión del programa reside en la creación de ficheros y el protocolo que se sigue a la hora de añadir nueva información a cada fichero. El origen del problema reside en la forma en que se reciben datos del puerto serie por parte de la función de la librería `System.IO.Ports`. Consiste en que cada vez que se recibe una cadena interpretada como válida, se crea un hilo que ejecuta la función asociada a dicho evento. En dicha función, se almacena el dato en una lista en memoria dinámica y se añade a un fichero ya creado, esta necesidad de acceder al mismo fichero ocasiona problemas por lo que es necesario un control de

acceso al fichero por semáforos. Además, hacer una operación de E/S y escribir en el disco datos tan pequeños como máximo un string de 20 caracteres es extremadamente ineficiente y puede reducir considerablemente la vida útil en discos SSD.

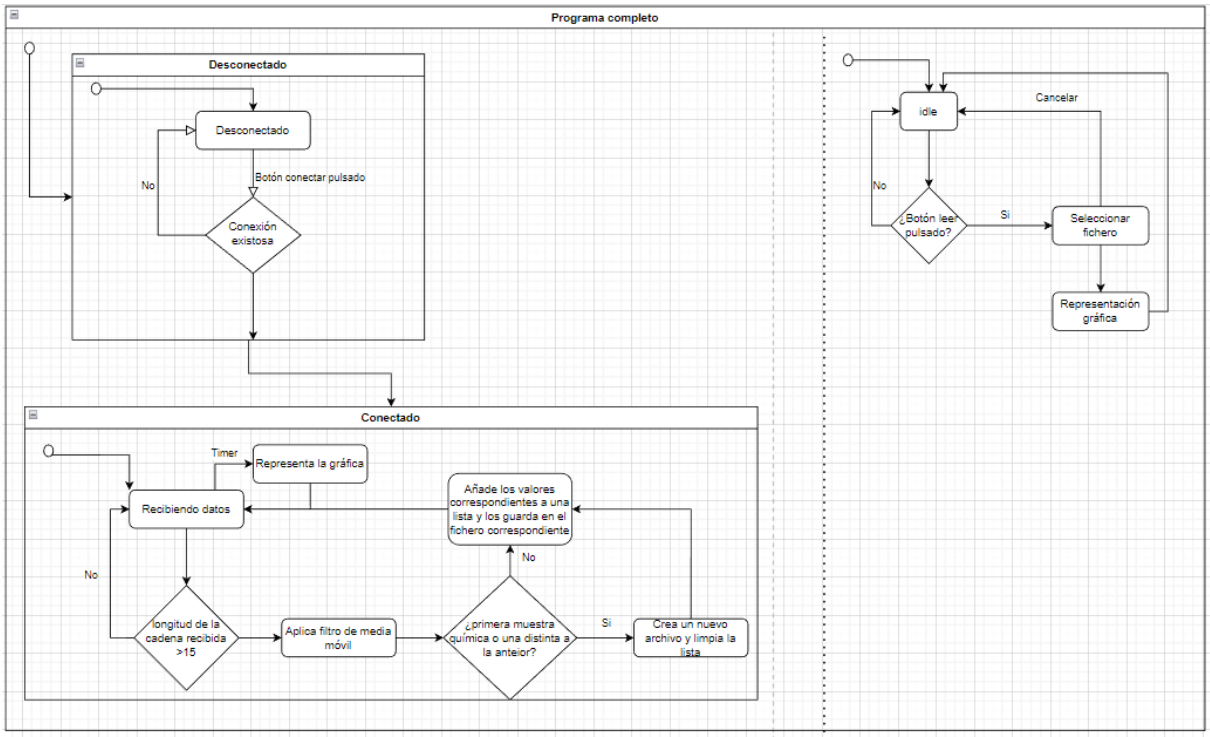

Figura 21 Diagrama de flujo versión 5 del programa C#

9.2.7 velocidad UART

La velocidad seleccionada para el puerto UART es de 38400 baudios. Teniendo en cuenta que el periodo de muestreo es de 10ms, tal y como se puede ver en la Figura 22, se podría ajustar la velocidad de transmisión a 19.200 baudios, pero dejaría muy poco margen al Arduino para ejecutar otras funciones y un aumento del volumen de datos repercutiría en errores de transmisión.

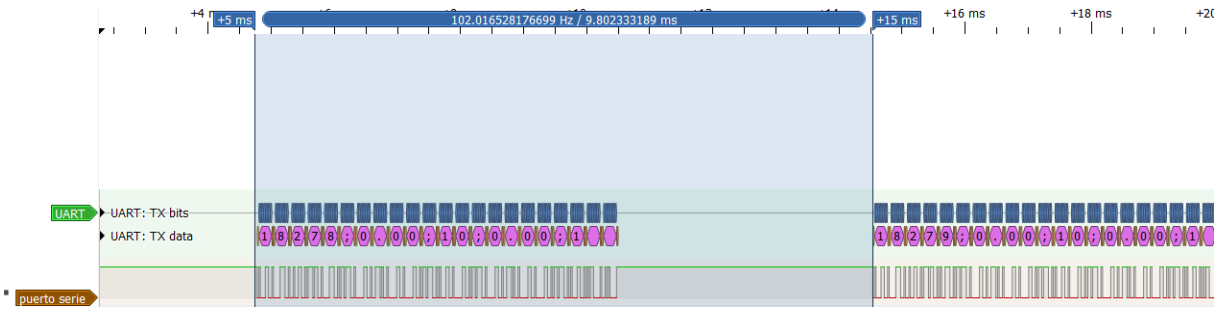

Figura 22 Tiempo entre transmisiones

9.2.8 Error de linealidad del Arduino

A continuación, se estudiará la linealidad del Arduino. Para ello, se aplicará una tensión conocida a la entrada del ADC y se comprobará con la que ha medido y esto se representará en la Figura 23. Para el experimento se hará la prueba con un Arduino con mucho uso y otro nuevo, se les nombrará *usado* y *nuevo*. La línea *ideal* sería el comportamiento ideal que correspondería a una relación 1:1 cumpliéndose Vread/Vin = 1.

Es importante tener en cuenta que la referencia de tensión no permanece constante en el Arduino puesto que se ha configurado para que sea variable. Esto se ha hecho porque en caso de configurarla a 1,1V el valor máximo sería 1,1V imponiendo un límite superior muy bajo. Entonces, cada vez que lee un valor con el ADC, si ve que es superior a 1,1V, cambiará la referencia a Vcc que sería idealmente 5V.

En la Figura 23 se puede observar la deriva que tiene el Arduino con mucho uso *viejo* mientras que el Arduino *nuevo* mantiene una cierta linealidad, sobre todo en valores previos a 1,1V dejando ver que su modo óptimo de funcionamiento sería de 0 a 1,1V. En futuras mejoras este análisis del error de linealidad se tendrá en cuenta.

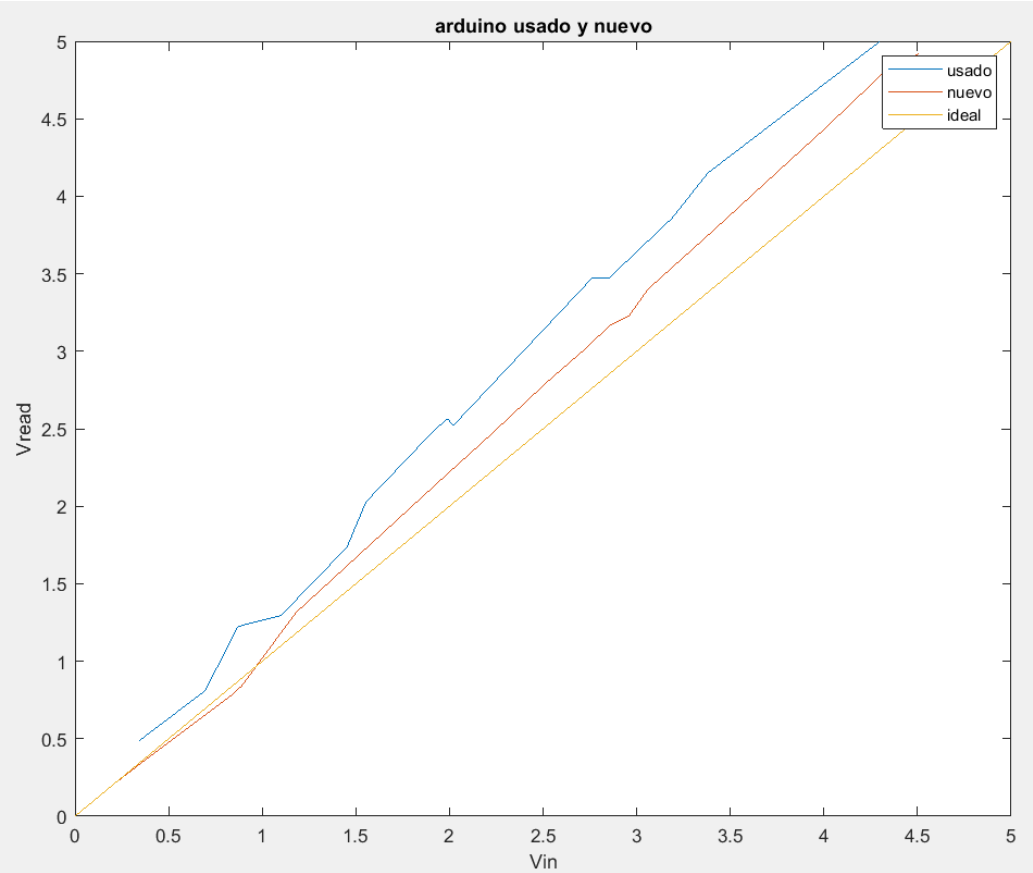

Figura 23 Error medición Arduino

9.3 Tercera aproximación

Con el fin de no prolongar excesivamente el documento, se tomará la tercera aproximación como la final, ignorando algunos pequeños cambios que hubo. En esta aproximación tal y como se puede ver en la Figura 24, se simplificó la parte Hardware para hacer un dispositivo más compacto y fiable. Además, al usar el LMV321 se evitó la necesidad de añadir una fuente de voltaje simétrica, permitiendo alimentar al circuito con el propio Arduino.

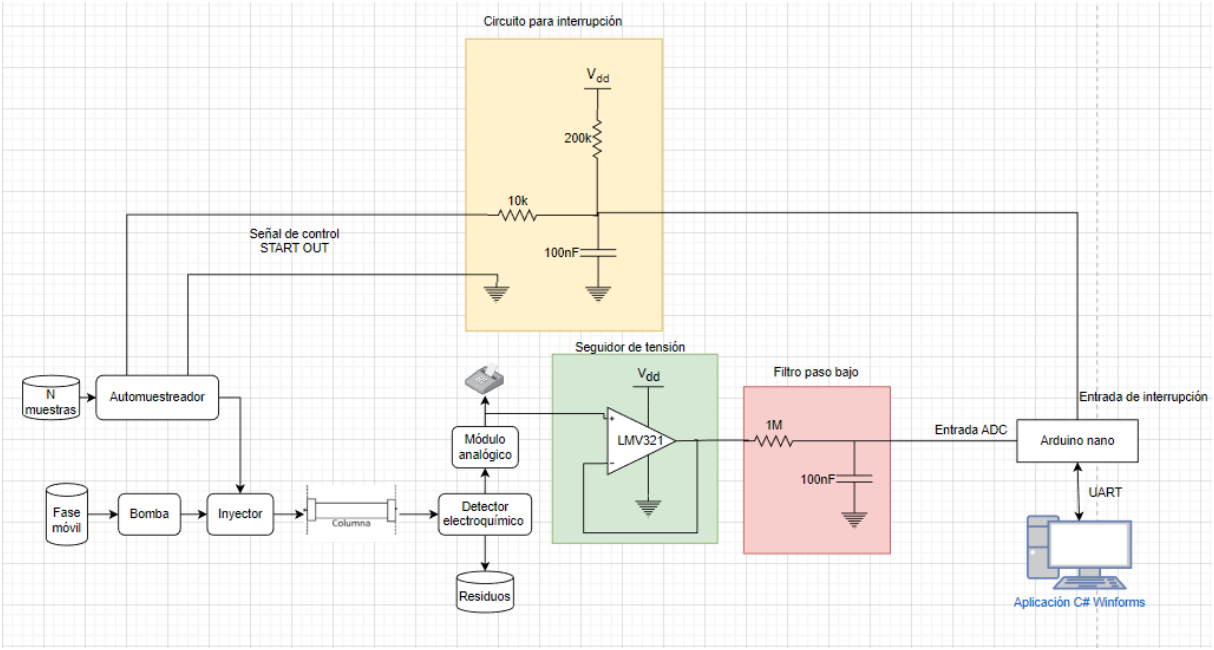

Figura 24 Esquema tercera aproximación

A continuación, se procederán a explicar los elementos de la Figura 24.

9.3.1 Circuito para interrupción

Para diseñar esta etapa, es fundamental hacerlo en base a las características del automuestreador y los posibles errores del Arduino. En la hoja del fabricante, se indica cómo se señaliza el comienzo de cada muestra química, como se puede ver en la Figura 25, se debe alimentar el circuito cumpliendo las especificaciones mostradas. Para el cumplimiento de las especificaciones se usará el divisor de tensión resistivo (ver Figura 24) cuya función principal es la limitación de la corriente, siendo su efecto sobre la tensión secundario, mientras que el condensador es necesario para eliminar los rebotes del interruptor electromecánico del cromatógrafo, evitando así falsas interrupciones al Arduino.

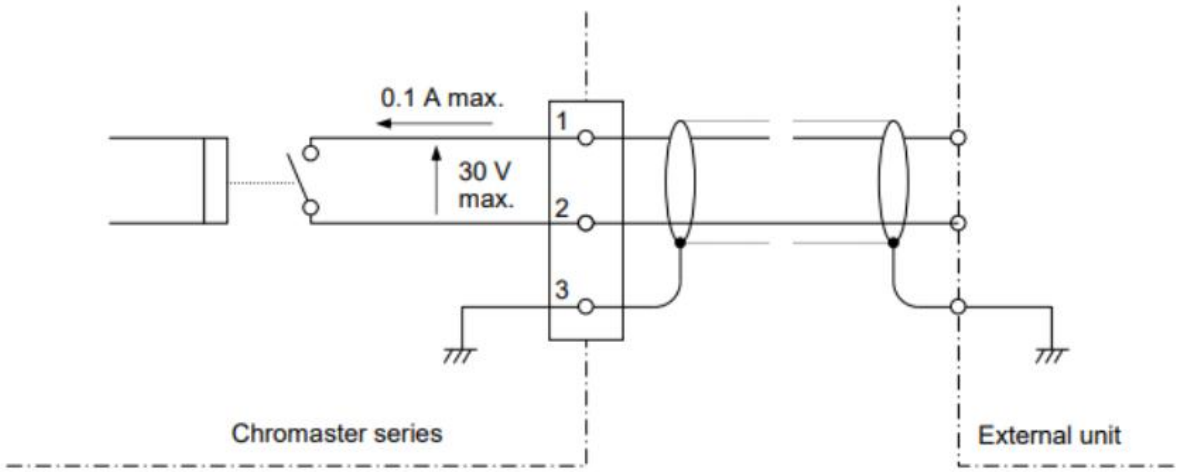

Figura 25 Circuito de la señal de salida del automuestreador[8]

El diseño del divisor de tensión debe proporcionar un voltaje cuando se cierre el circuito lo suficientemente bajo para activar la interrupción por flanco de bajada en el Arduino. En Arduino Nano (y en la mayoría de los microcontroladores en general), el voltaje mínimo para ser considerado como LOW (bajo) en una entrada digital es típicamente de 0.3 * VCC, donde VCC es la tensión de alimentación del microcontrolador.

Por lo tanto, en un Arduino Nano alimentado con 5V, un voltaje de entrada digital de menos de aproximadamente 1.5V se considerará como LOW. En este caso, teniendo resistencias de 200kΩ y 20 kΩ, el voltaje en la entrada de interrupción del Arduino cuando se cierra la entrada de interrupción será el indicado en Ecuación 6.

$$V_{entrada\ de\ interrupción\ Arduino} = \frac{10\text{k}\Omega}{10\text{k}\Omega + 200\text{k}\Omega} VCC = 0{,}048 * VCC$$

Ecuación 6

Usando un analizador lógico, se puede comprobar que la duración de la señal del automuestreador que indica la inyección de una nueva muestra es de aproximadamente 1s, visto en la Figura 26

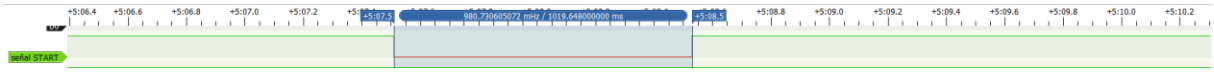

Figura 26 Duración señal automuestreador

9.3.2 Seguidores de tensión

El propósito de este amplificador operacional es proporcionar una entrada de alta impedancia para evitar cualquier interferencia en los otros dispositivos conectados. Como novedad se puede observar en Figura 24 que el amplificador operacional que se usó en la segunda aproximación se ha cambiado, eliminando la necesidad de añadir un voltaje simétrico, pudiendo usar al Arduino como fuente de tensión.

9.3.3 Referencia de tensión variable

La referencia de tensión fue eliminada puesto que consumía en exceso para su polarización y además la pérdida de información ocasionada por la ausencia de la referencia de tensión es despreciable.

9.3.4 Filtro paso bajo

Debido a que la señal medida experimenta variaciones muy lentas, se optó por implementar un filtro paso bajo de una frecuencia lo más pequeña posible con los componentes en stock. De esta forma se eliminará prácticamente todo el ruido de alta, media y baja frecuencia.

La frecuencia del filtro vendrá definida por la Ecuación 7.

$$F_{corte} = \frac{1}{2\pi * C * R} = \frac{1}{2\pi * 1M\Omega * 100\text{nF}} = 1.591\ Hz$$

Ecuación 7

9.3.5 Modificación en la interfaz

La modificación más visible es el nuevo botón “next”. La mayoría de los cambios son de eficiencia de código y considerable mejora de la opción “leer”. En la Figura 27 *Interfaz de programa Versión* 14 se muestra la interfaz y se procederá a continuación a explicar un resumen de las nuevas funcionalidades de la función leer y del nuevo botón “next”.

Figura 27 Interfaz de programa Versión 14

Botón leer: esta funcionalidad ha experimentado multitud de cambios y se han añadido muchas características. La primera es la creación de carpetas para una mejora en la gestión de los archivos como la vista en la Figura 28.

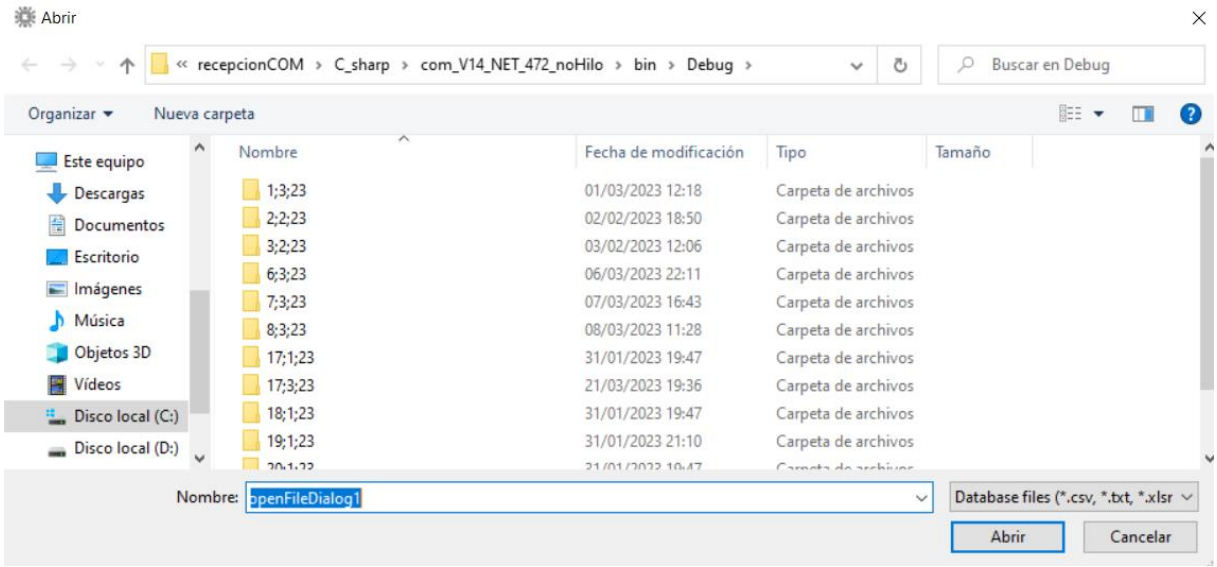

Figura 28 Cuadro de diálogo de la opción de leer. Versión 14

En la representación de la gráfica se procederá a explicar los pasos que sigue el programa:

1. Abre el fichero, lo lee y aplica el filtro de media móvil.
2. Aplica una función de detección de picos personalizada que se explicará más adelante.
3. Los puntos guardados en memoria dinámica por la función de detección de picos son usados por la función de cálculo de área y se asigna un área a cada curva.
4. Los marcadores son representados sobre la gráfica con el área de cada curva.

En la Figura 29 se puede ver el tratamiento completo de la señal y el cálculo del área debajo de cada curva. Se ha realizado una considerable mejora gráfica respecto a la Figura 20, además del cambio de la nomenclatura del eje x de número de muestras a minutos.

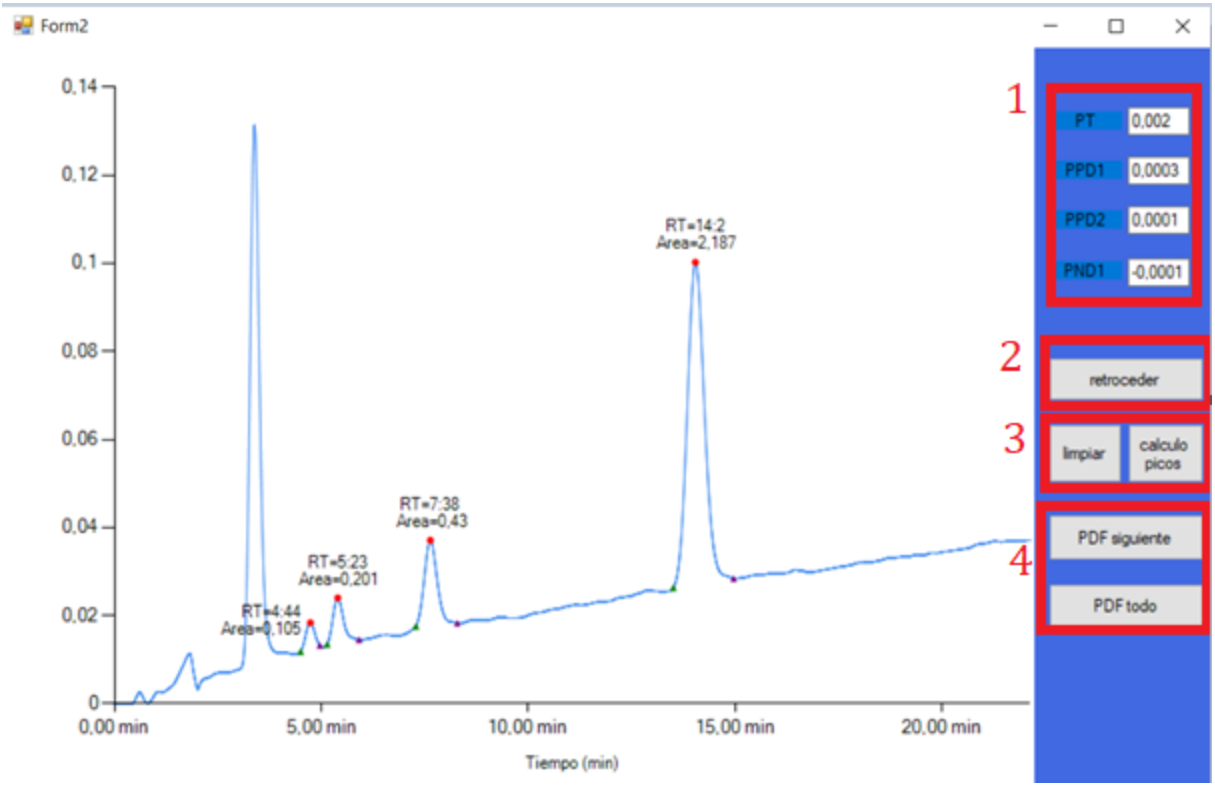

Figura 29 Ejemplo de representación de una muestra clínica con la opción leer. versión 14

La funcionalidad de los botones de la Figura 29 será explicada a continuación:

1. Parámetros de la función de detección de picos: permite variar dichos parámetros para ajustar la función de detección a picos más pequeños o más grandes.
2. Botón retroceder: permite el borrado de puntos con el cálculo de picos personalizado.
3. Botones limpiar y cálculo de picos: el botón limpiar elimina todos los marcadores puestos por la función de calcular picos. Mientras que el botón "**cálculo de picos**" aplica nuevamente la función de cálculo de picos usando los parámetros de la parte superior.
4. Botones PDF siguiente y PDF todo:
    a. PDF siguiente: inicia un nuevo documento pdf y abre la primera muestra química de la carpeta en la que está el directorio de trabajo. Una vez establecidos los marcadores personalizados habrá que volver a presionar el botón para pasar a la siguiente muestra hasta que llega a la última muestra y genera el documento pdf con marcadores personalizados.
    b. PDF todo: genera un pdf con todas las muestras dentro del directorio de trabajo con los marcadores de la función de cálculo de picos.

Botón next: envía al Arduino un comando para que reinicie el número de la muestra y aumente el número de la muestra química.

9.3.6 Versión 14 del programa C#. Flujo de funcionamiento

En la Figura 30 se muestra el diagrama de flujo del programa. En esta versión se han corregido errores y se han optimizado muchas funciones, además del gran añadido de funcionalidades y herramientas. Debido al gran volumen de actualizaciones se procederá a explicar las funciones y los cambios más destacados.

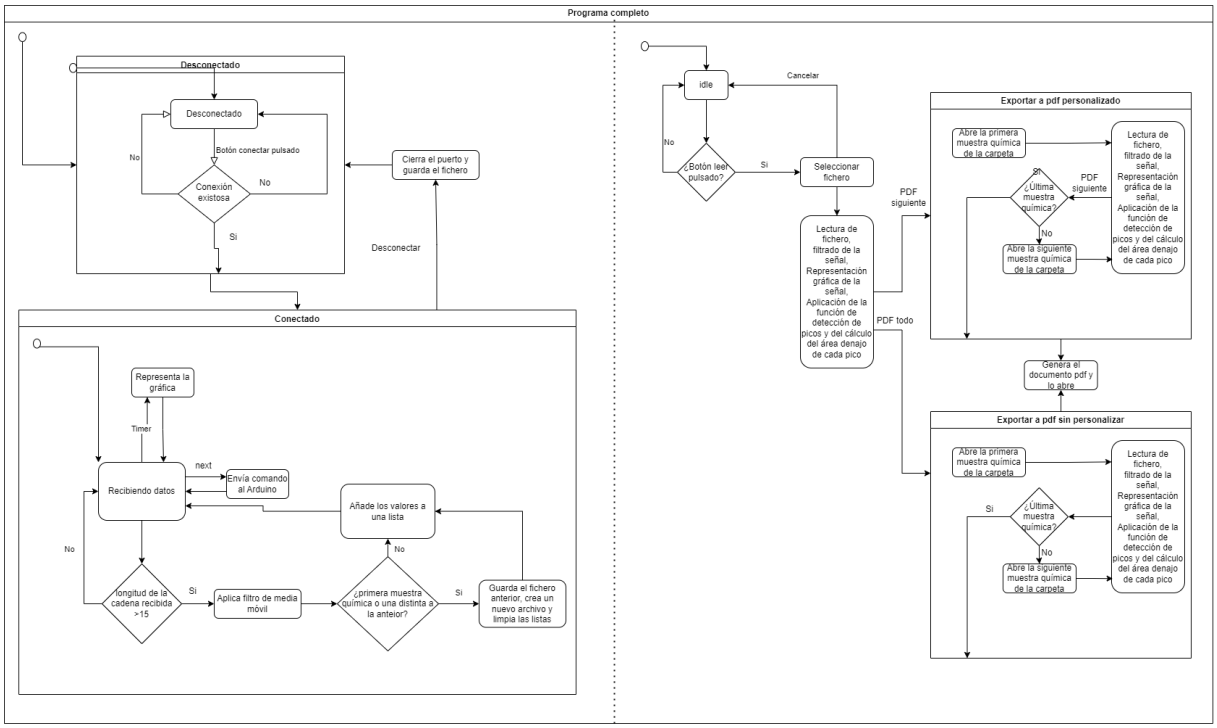

Figura 30 Diagrama de flujo de la versión 14 del programa C#

9.3.6.1 Mejora de la eficiencia del guardado de archivos

En anteriores versiones, cuando se recibía un dato, éste se añadía al fichero inmediatamente. Esto era extremadamente ineficiente porque requería de abrir el fichero, añadir 20 bytes y cerrarlo. En discos duros SSD que están sustituyendo a los HDD, no es recomendable hacer operaciones de escritura, sobre todo teniendo en cuenta que el tamaño mínimo de escritura suele ser de 4K bytes.

Por tanto, se optó por guardar los datos en memoria dinámica y cuando se cambiase la muestra química o se desconectase del puerto serie, se escribiese el archivo en el disco duro. De esta forma se reduce considerablemente el número de escrituras en el disco duro.

Para un tiempo de muestreo de 50ms, suponiendo que la muestra química suele tardar 20 minutos en completarse, nos da un total de 24.000 muestras. Sabiendo que cada muestra ocupa 20 bytes, se obtiene

un volumen de datos a escribir de 469KB, suponiendo un tamaño de escritura de 4KB se obtiene un total de 118 ciclos de escritura. Usando la versión anterior, el guardado del archivo descrito requeriría un total de 24.000 ciclos de escritura. Por tanto, se obtiene una mejora del 203390%, por supuesto, esta mejora viene al precio de una peor fiabilidad ya que cualquier fallo del programa o del equipo en mitad de la muestra química provocará que ésta no se guarde y, por tanto, se pierda.

9.3.6.2 Mejoras de la capacidad de lectura de archivos

Esta funcionalidad es la que más cambios ha sufrido, debido a que la interpretación de los datos y un correcto modo de exportación es desde un punto de vista de un trabajador del laboratorio, lo más importante.

9.3.6.2.1 Implementación de la función de detección de picos y cálculo de área

La función de detección de picos usa 4 parámetros fundamentales vistos en la Figura 29. Su función y su significado se explicará a continuación de la Figura 31.

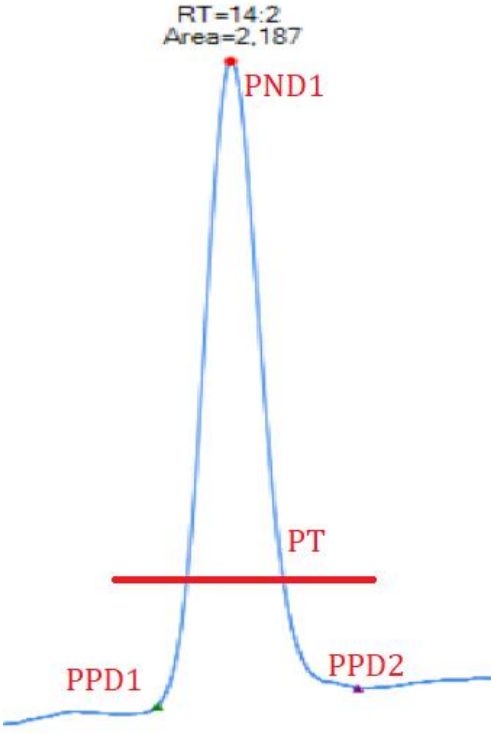

Figura 31 Visualización detección de picos

- PT (Peak Threesold): Valor mínimo de la altura de pico, si se detecta un pico cuya altura máxima esté por debajo de este valor, será descartado.
- PPD1 (Peak Positive Derivative 1): Un valor por encima del estipulado indicará el inicio de un posible pico. Será marcado por un triángulo verde.
- PPD2 (Peak Positive Derivative 2): Un valor por encima del estipulado indicará el fin del pico completo. Será marcado con un triángulo morado.

- PND1 (Peak Negative Derivative 1): Un valor por debajo del estipulado indicará el comienzo del descenso del pico, indicando así el valor máximo del pico. Será marcado con un punto rojo.

La función de detección de pico detectará los picos correspondientes y los almacenará en variables globales. Posteriormente se calculará el área bajo la curva de cada pico y se representará en el pico, indicando además el RT (Retention Time) en un **formato de "minutos : segundos".**

El cálculo del área se hará mediante la regla del trapecio, este método se usa para aproximar el área bajo una curva. Se divide el área en varios trapecios y se suma el área de cada uno para obtener una aproximación del área total. La fórmula para calcular el área de un trapecio es (base mayor + base menor) x altura / 2. Para aproximar el área bajo la curva, se aplica la fórmula del trapecio a cada trapecio y se suma el resultado. La precisión de la aproximación mejora a medida que se utilizan más trapecios.

Para saber la concentración únicamente, se requiere el área bajo la curva, tomando como referencia la línea base. Como esta línea base está constantemente en crecimiento, hay que restar un área parecida a un trapecio como se ve en la figura Figura 32.

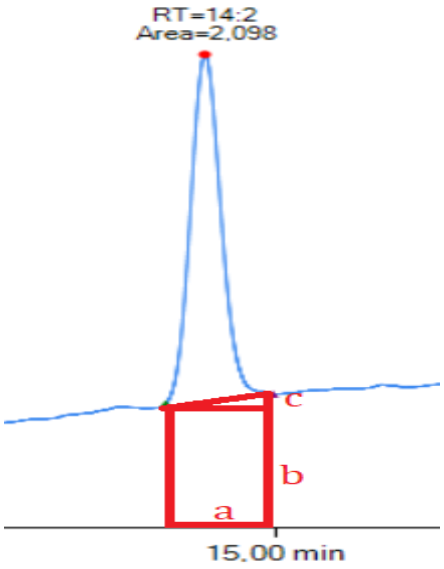

Figura 32 Resta de la información no necesaria

La ecuación del área quedará como la Ecuación 8.

$$A_{curva} = A_{regla\ del\ trapecio} - a * (b + \frac{c}{2})$$

Ecuación 8

9.3.6.2.2 Implementación de la personalización de gráficas

Puesto con la función de detección de picos puede no detectar algún pico, es necesario añadir una función para que el doctor pueda seleccionar el pico de interés. Esta función consiste en clicar 2 puntos y se calculará el área bajo la curva entre esos dos puntos, restando por supuesto el área de offset. En caso de querer eliminar un punto se podrá usar el botón *retroceder*.

En la Figura 33 se pueden ver los marcadores personalizados y cómo en el primer punto que sería el de la izquierda representa el tiempo en ese punto y como en el segundo punto (a la derecha) se representa el área ya calculada con el tiempo de retención (RT). En la Figura 34, por su parte, se puede ver la comparación entre el área calculada por la función de cálculo de picos y el área calculada de forma personalizada. Se puede observar la similitud o igualdad entre ambos valores, mientras que para la zona recta con pendiente sin curvas se obtiene un área de 0,004 que es despreciable.

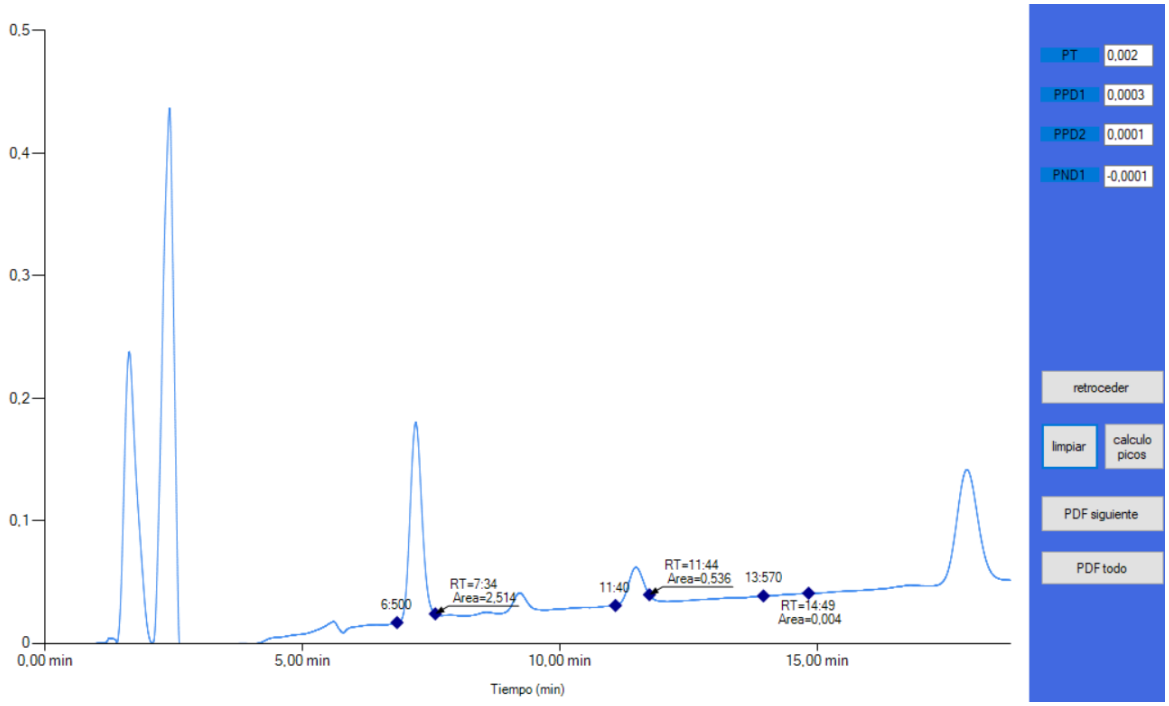

Figura 33 Función con marcadores personalizados

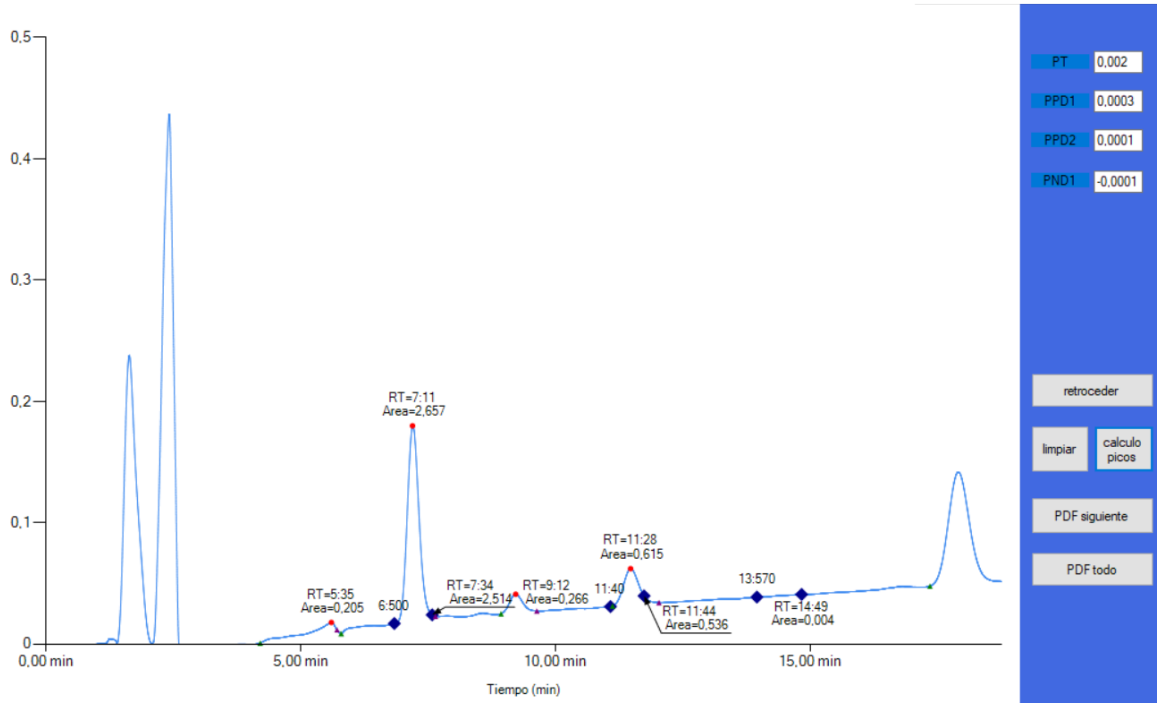

Figura 34 Función con marcadores personalizados y los de la función de cálculo de picos

9.3.6.2.3 Exportación a pdf

Con el fin de poder imprimir u obtener un archivo más universal como es el formato pdf, se han añadido dos opciones para exportar a pdf, mediante los botones “PDF siguiente” y “PDF todo”. “PDF todo” exporta todas las muestras de la carpeta a un pdf sin ningún tipo de personalización, mientras que “PDF siguiente” proporciona al usuario la capacidad de editar cada muestra y marcar únicamente los picos que interesan.

En la Figura 35 se puede ver el resultado al usar el botón “*PDF todo*”, en cada página del PDF imprime 2 gráficas que van numeradas según su orden en la carpeta. En el documento estarán todas las muestras químicas de la carpeta del día correspondiente.

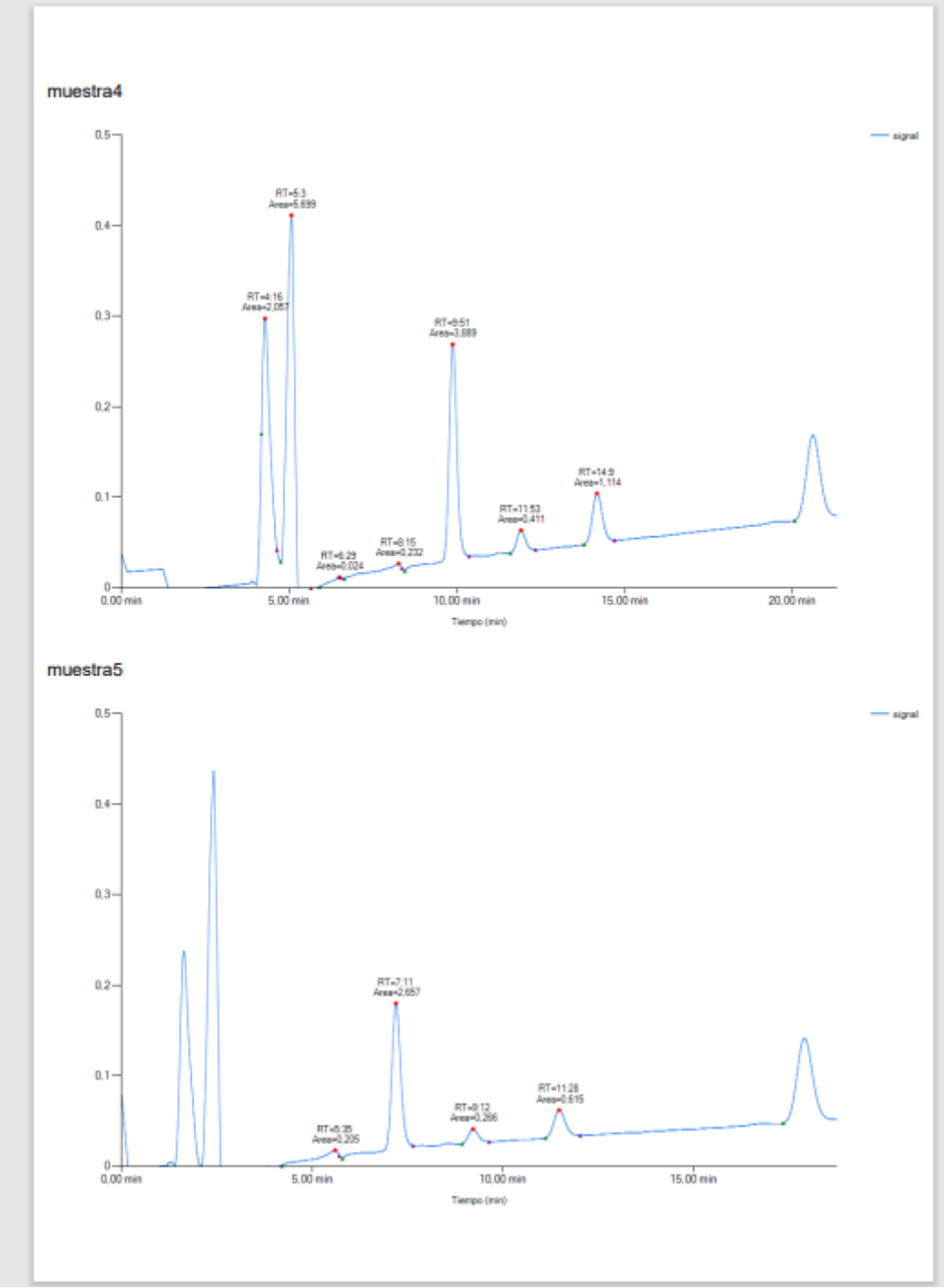

Figura 35 Ejemplo de página usando *PDF todo*

En la Figura 36 se pueden observar las mismas gráficas que en la Figura 35 pero con marcadores personalizados.

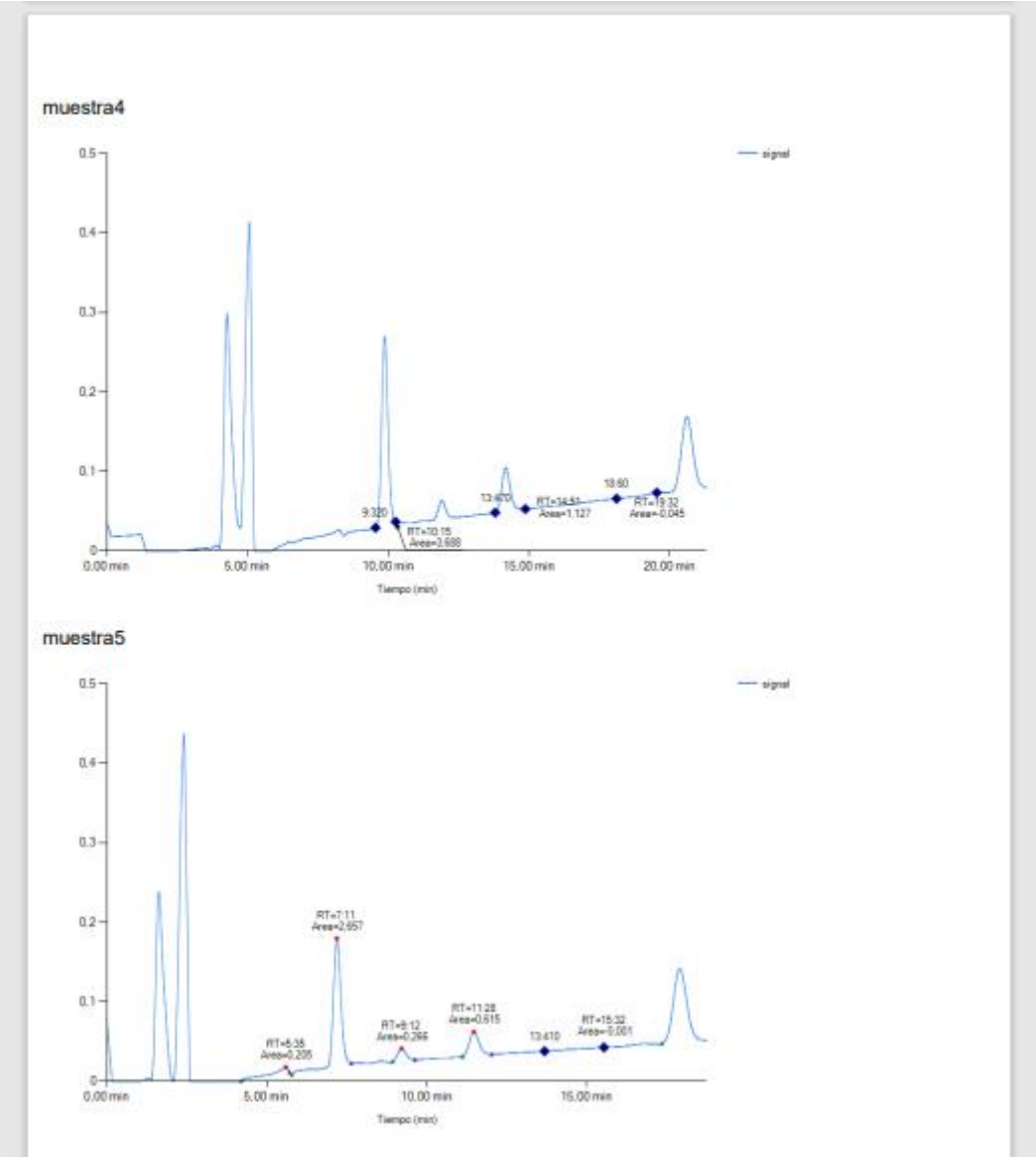

Figura 36 Ejemplo de una página usando *PDF siguiente*

9.4 Implementación y fabricación del prototipado

A falta de presupuesto para la fabricación de la PCB, se optó por fabricar el circuito en una placa de prototipado y como contenedor, una caja estanca. La soldadura es parecida a la de una placa de circuito, pero imposibilita poner componentes SMD, por lo que el LMV321 necesita un adaptador a THT.

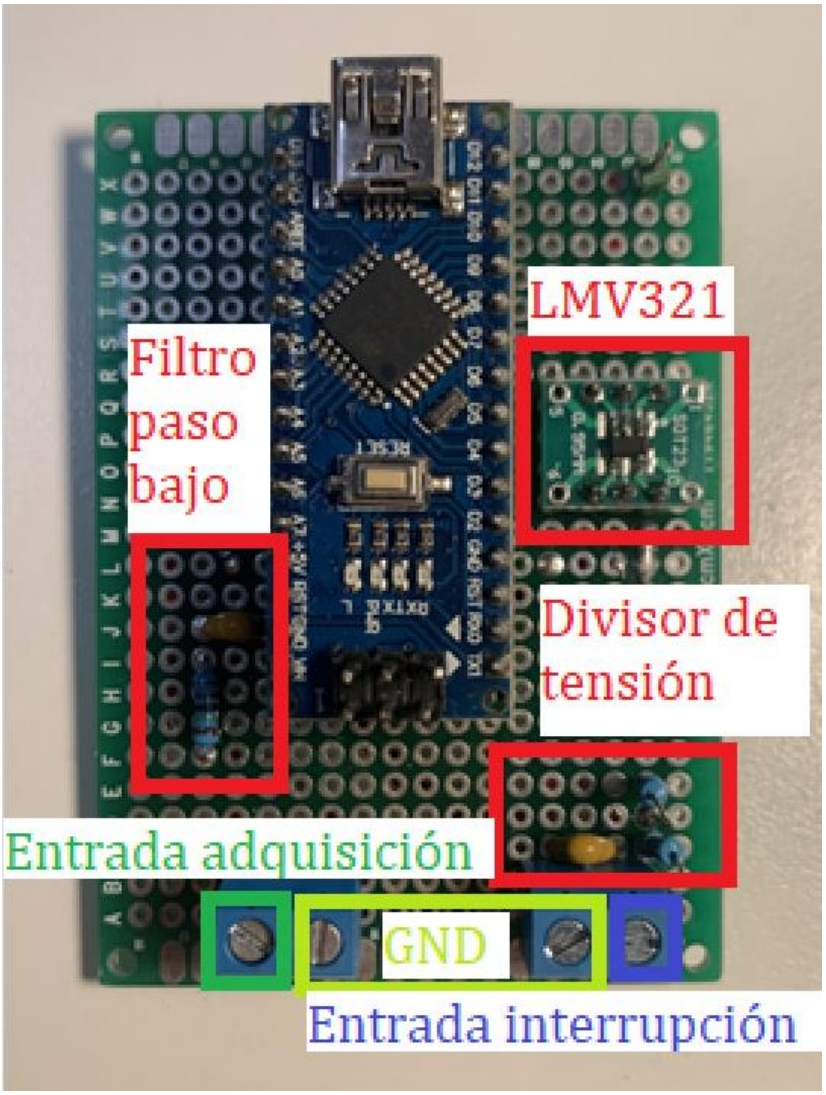

Figura 37 Cara superior de la placa de prototipado

El principal problema en estas placas de prototipado es el conexionado entre componentes. A falta de pistas, comúnmente se suele hacer emulando las pistas con estaño, pero se optó por usar cables para distancias largas y evitar un uso excesivo de soldadura.

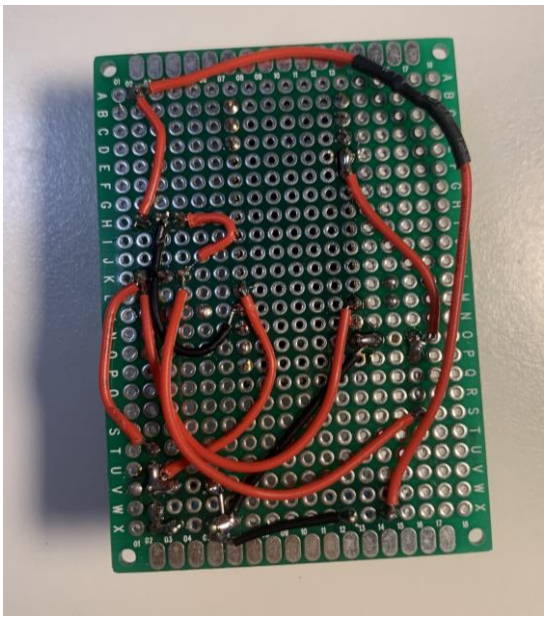

Figura 38 Cara inferior de la placa de prototipado

Respecto al contenedor, la caja estanca, fue necesario hacer 4 taladros para la sujeción de la placa de prototipado.

Figura 39 Caja estanca con taladro

10. Conclusiones y trabajo futuro.

Lo que a priori parecía un trabajo fundamentalmente de Hardware de adquisición y tratamiento de una señal se ha acabado convirtiendo en un trabajo más desarrollado en la parte Software e informática. La aplicación en un principio sólo tenía la función de recolectar los datos, guardarlos y representarlos, la causa principal de su evolución fue la constante adaptación a las necesidades que presentaba el personal del laboratorio.

Uno de los principales problemas, fue que para probar el funcionamiento y tomar una muestra, era necesario ir en horario de trabajo y comprobar en paralelo las muestras reales de pacientes cuyo resultado era crucial para su salud, por lo que cualquier interferencia eléctrica era completamente inadmisible.

Otro de los problemas fue la experiencia nula en lenguajes de programación orientados a objetos y en interfaces gráficas. Esto, combinado con la necesidad de controlar hilos en el recibimiento de datos por USB, supuso un gran problema inicial que, conforme se fue adquiriendo práctica, se eliminó y en problemas posteriores el proceso de corrección sería mucho más rápido.

El techo que se observó cuando se intentaba hacer el cálculo de la concentración fue la imposibilidad de automatizar esa tarea con un porcentaje de éxito apreciable. Esto requeriría que la función detección de picos supiese exactamente los 4 picos que interesan y qué sustancia es cada uno. Además, habría que tener en cuenta si al momento se está pinchando una muestra química de estándar, control 1, control 2 o de paciente.

Una de las mejores aportaciones de este trabajo es que, al confeccionar una base de datos, en un futuro con suficientes muestras se podrá entrenar un modelo que detecte los 4 picos de interés y permita hacer el cálculo de la concentración con tanta precisión como un humano. Este hecho que permitirá automatizar el cálculo de la concentración y alcanzar un nivel de automatización superior.

11. Listado de materiales y equipo

El principal enfoque de este proyecto ha sido hacer una aplicación lo más sencilla de entender para el personal de laboratorio y lo menos costosa posible. Es necesario notar que para la ejecución de todo el proyecto no es necesario todo el Software o el Hardware utilizado en su desarrollo.

11.1 Desarrollo

Es importante mencionar que para el desarrollo no ha sido necesario Matlab, sólo se ha usado en exclusiva para la presentación de algunos datos en este documento.

Tabla 2 Equipo y Software necesario para desarrollo

Tipo	Recurso	Unidades	Coste
Software	Microsoft Visual Studio	1	0
	Arduino IDE	1	0
	Windows 10 Home	1	130€
	PulseView	1	0
Hardware	Ordenador gama media-alta	1	500€
	Analizador Lógico 24MHz	1	6,73€
	Multímetro Digital	1	20€
	Arduino Nano	1	1,27€
Total			658€

11.2 Ejecución

Para la ejecución del programa hacen falta menos recursos

Tabla 3 Equipo y Software necesario para ejecución

Tipo	Recurso	Unidades	Coste
Software	Windows 7 o superior	1	50€

Hardware	Ordenador gama baja	1	100€
	Arduino Nano	1	1,27€
	Circuito	1	7,62€
Total			158,89€

Tabla 4 Listado de componentes del circuito

Componente	Unidades	Coste
Condensadores cerámicos 100nF	2	0.40€
Resistencias 1/4W 1M, 200k,10k	3	0.12€
LMV321	1	0.11€
Placa de prototipado	1	0.20€
Caja estanca	1	1,79€
Montaje y fabricación	2	15,38€
Total		33,26€

12. Presupuesto

12.1 Costes directos

A continuación, se reflejan los costos totales de desarrollo y fabricación, todos los precios incluyen el IVA del 21%. Se añadirá un 30% en concepto de ganancias.

Tabla 5 Costes directos

Tipo	Subtipo	Unidades	Coste por unidad	Coste total para la empresa	Coste para el cliente
Materiales	Equipos	1	658€	658€	855,4€
	Producto	1	34,02€	34,02€	44,23€
Costes humanos Ingeniero electrónico	Horas	120	15,38€	1.845,6€	2.399,28€
	Movilidad	16	7€	112€	145,6€
Mantenimiento	Análisis	1	38,47€	38,47€	50€
	Transporte	2	7€	14€	18,2€
Total				2.702,09€	3.512,72€

A partir de la Tabla 5 se puede calcular que el coste de producción del producto es tan sólo el 1,26% del coste total, el bajo precio de producción se puede aprovechar para obtener más beneficios acondicionando la aplicación para otros usos.

La estructura del proyecto es un sistema que toma muestras de voltaje, las envía a un ordenador, las representa y las analiza. Esta estructura se puede adaptar rápidamente a muchas otras aplicaciones manteniendo un coste de fabricación muy bajo, lo que supone una ventaja competitiva clara. Además, cabe la posibilidad de diseñar una PCB y pedir su fabricación en una empresa externa lo cual reduciría considerablemente los costes de producción y mejoraría las prestaciones.

13. Anexo A

13.1 Manual de Usuario

Digitalización y reacondicionamiento de un sistema clínico de cromatografía

Versión:

Fecha: 25/04/2023

0100

Queda prohibido cualquier tipo de explotación y, en particular, la reproducción, distribución, comunicación pública y/o transformación, total o parcial, por cualquier medio, de este documento sin el previo consentimiento expreso y por escrito de David Rodríguez Martínez

HOJA DE CONTROL

Organismo	Universidad de Alcalá		
Proyecto	Digitalización y Reacondicionamiento de un sistema clínico de cromatografía		
Entregable	Digitalización y reacondicionamiento de un sistema clínico de cromatografía		
Autor	David Rodríguez Martínez		
Versión/Edición	1.7	**Fecha Versión**	16/04/2023
Aprobado por	David Rodríguez Martínez	**Fecha Aprobación**	22/05/2023
		N.º Total de Páginas	13

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
0100	Versión inicial	David Rodríguez Martínez	16/04/2023

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos
David Rodríguez Martínez

INDICE

A.1. DESCRIPCIÓN DEL SISTEMA 54

A. OBJETO 54

A.2. INSTALACIÓN 54

A. INSTALACIÓN HARDWARE 54
B. INSTALACIÓN SOFTWARE 55

A.3. DESCRIPCIÓN DEL SISTEMA 56

A. PANTALLA 1 56
B. PANTALLA 2 57
C. MENSAJES DE ERROR 58
D. AYUDAS CONTEXTUALES 59

A.4. MODO DE FUNCIONAMIENTO 60

A.5. FAQ 61

A.6. MANTENIMIENTO 61

E. CÁLCULO DE LA NO LINEALIDAD 61

A.7. GLOSARIO 63

A.8. CONTACTO 63

A.1. DESCRIPCIÓN DEL SISTEMA

a. Objeto

Este manual ha sido diseñado para proporcionarle toda la información necesaria para utilizar correctamente el producto o servicio y garantizar una experiencia satisfactoria para el usuario.

Este manual de usuario ha sido creado para guiarlo a través de todas las características y funciones del producto o servicio, y ayudarlo a solucionar cualquier problema que pueda surgir durante su uso. Su finalidad es proporcionar instrucciones claras y detalladas, así como gráficos e imágenes para mejorar la comprensión.

Antes de utilizar el producto o servicio, se recomienda encarecidamente que lea detenidamente este manual para familiarizarse con todas las funciones y características que ofrece. El manual también incluye información sobre la seguridad y las precauciones a tomar durante el uso del producto o servicio.

Si tiene alguna pregunta o problema durante el uso del producto o servicio, este manual de usuario es la primera fuente de información y solución de problemas. Si no encuentra la respuesta que está buscando, no dude en ponerse en contacto con nuestro servicio de atención al cliente para obtener ayuda adicional.

A.2. Instalación

a. Instalación Hardware

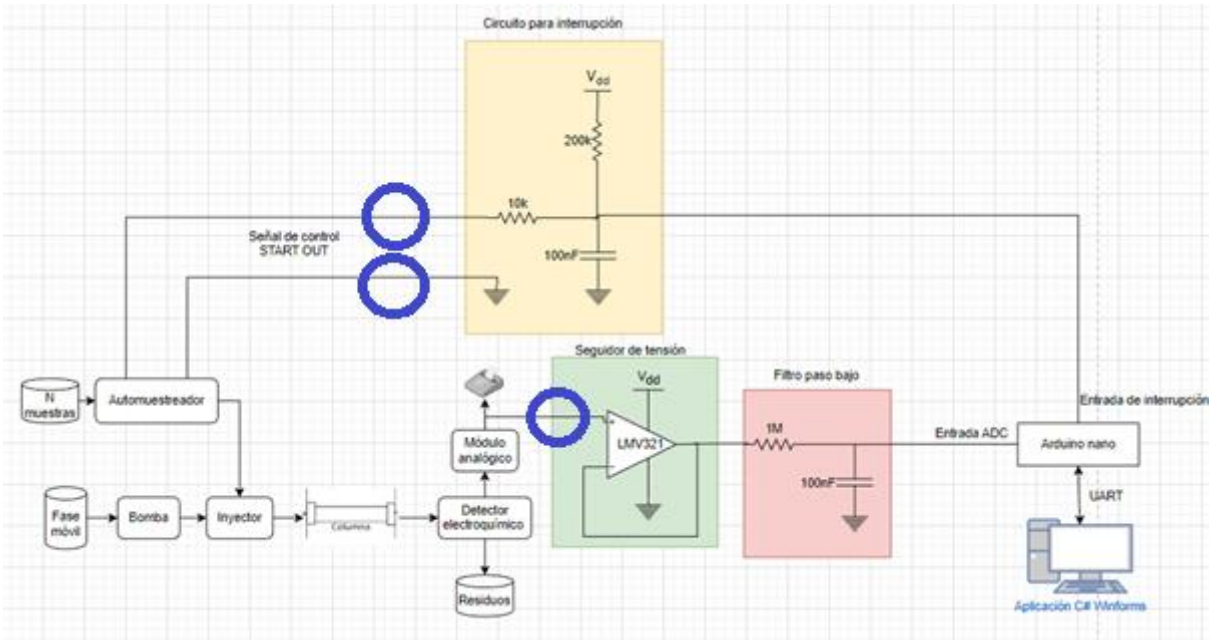

Figura 40 Diagrama completo con cables a conectar marcados

En la Figura 40 se muestra el diagrama completo y con las circunferencias azules los cables correspondientes que son necesarios conectar correctamente. En la Figura 41 se especifica de una forma más visual el modo de conexionado con el módulo analógico. Respecto al conexionado con el automuestreador, la polaridad de los cables es indiferente por las características de la señal.

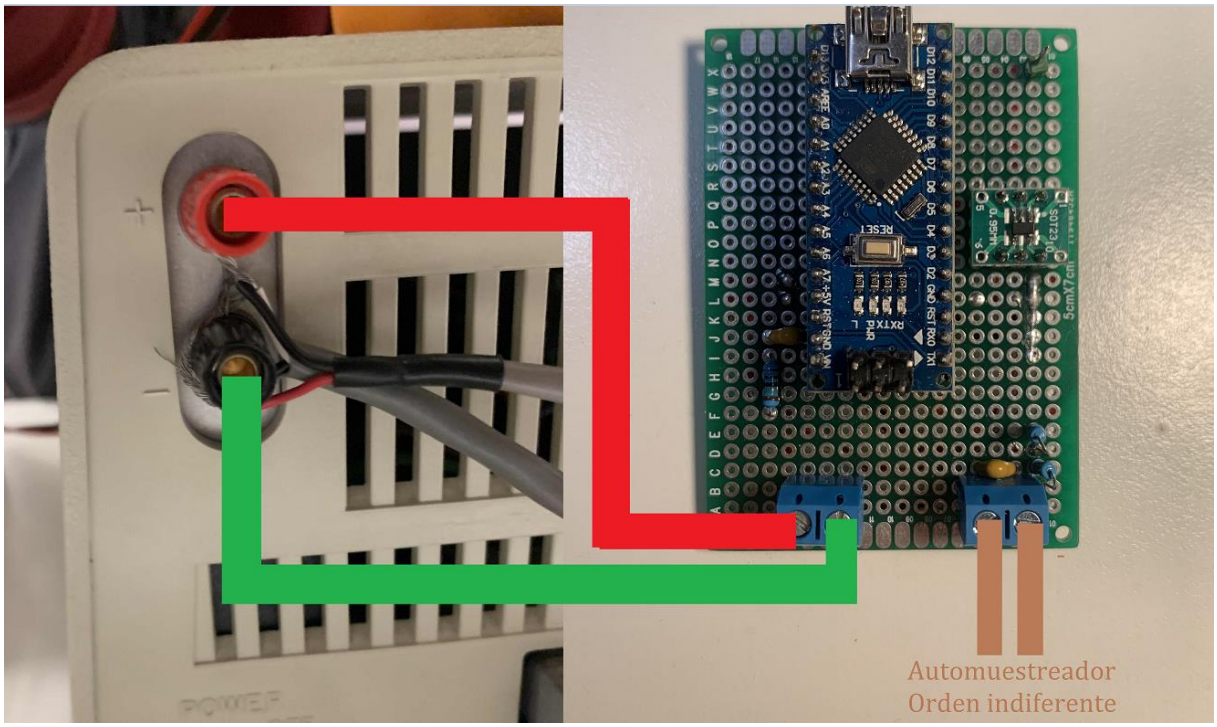

Figura 41 Conexión circuito con módulo analógico

b. Instalación Software

En este apartado se deben instalar una serie de drivers y paquetes de distribución para el correcto funcionamiento:

- Driver CH340: estarán dentro de la carpeta del programa. En caso de pérdida, se deberá instalar un controlador válido para el CH340.
- Paquete de distribución Microsoft .NET Framework 4.7.2: el instalador de este paquete también se incluye en la carpeta del programa. En caso de extravío se puede encontrar fácilmente si se tiene conexión a internet.

A.3. DESCRIPCIÓN DEL SISTEMA

a. Pantalla 1

Figura 42 pantalla 1

Puerto COM	En este cuadro de texto deberá ponerse el puerto COM en el que está conectado el Arduino. Cada puerto USB del ordenador será un puerto COM distinto. Para ver en qué puerto está conectado, hay que ir a "Administrador de dispositivos", buscar "Puertos (COM y LPT)" y el nombre "USB-SERIAL CH340 (COM <nº de puerto>)" tal y como se muestra en la Figura 43
conectar	Si está conectado el Arduino, al presionar este botón se conectará por el puerto serie y comenzará el registro y la representación de datos.
desconectar	Permitirá desconectarse del Arduino, interrumpiendo así el registro de datos.
reiniciar	Estando el Arduino conectado, borra todos los datos registrados en la presente muestra química y comienza la representación y el guardado desde cero.
next	Guarda los datos de la muestra química actual y pasa a la siguiente.
Datos recibidos	Mostrará la cadena de caracteres recibida del Arduino en tiempo real.
leer	Abre un cuadro de diálogo para seleccionar el archivo correspondiente y abre la pantalla 2.
cerrar	Cierra la pantalla 1

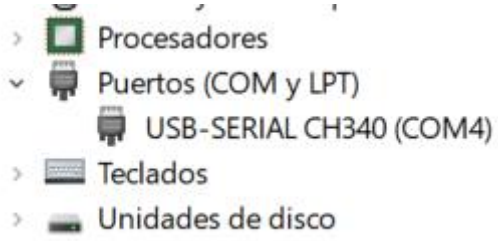

Figura 43 Ejemplo de búsqueda de puerto COM

b. Pantalla 2

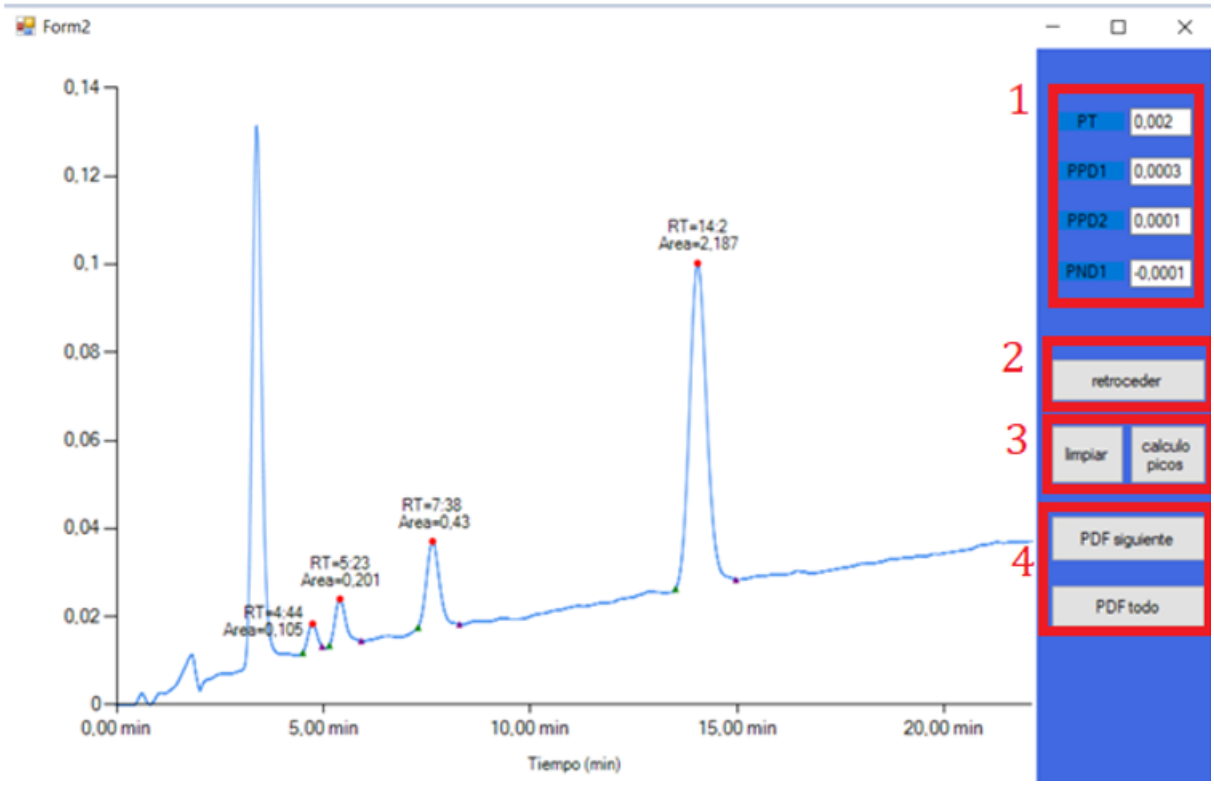

Figura 44 Pantalla 2

1	PT	Umbral de altura de pico
	PPD1	Primera derivada positiva (triángulo verde)
	PPD2	Segunda derivada positiva (triángulo morado)

	PND1	Derivada negativa (punto rojo)
2	retroceder	Elimina el último marcador puesto manualmente. Sólo funciona con los marcadores puestos manualmente.
3	limpiar	Elimina todos los marcadores puestos por la función de detección de picos.
	cálculo de picos	Detecta todos los picos usando los parámetros del campo 1 y calcula el área de cada uno.
4	PDF siguiente	Si es la primera vez que se pulsa, abre la primera muestra de la carpeta sin guardar los cambios.
		Si no es la primera vez que se pulsa, guarda los cambios y abre la siguiente muestra de la carpeta. Y, si está en la última muestra y se pulsa, cierra la ventana y abre el documento pdf generado.
	PDF todo	Detecta los picos y calcula el área de todas las muestras de la carpeta, acto seguido, representa el documento pdf con todas las gráficas de las muestras.

c. Mensajes de error

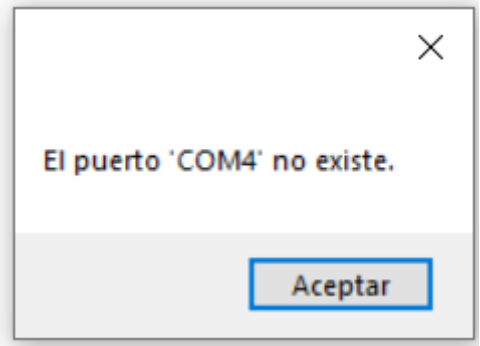

Figura 45 Error Arduino no conectado

d. Ayudas contextuales

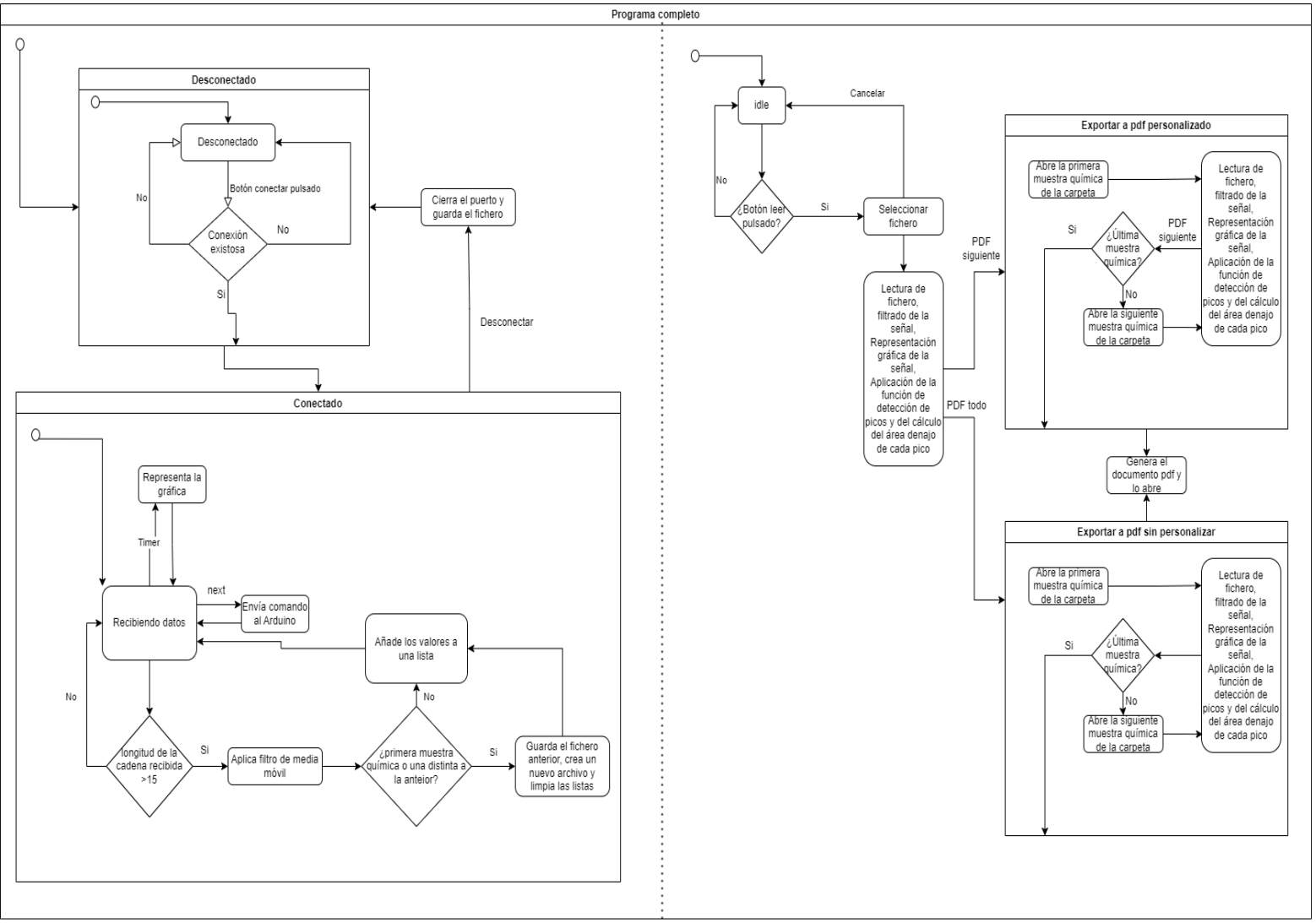

Figura 46 Diagrama de flujo del programa completo

A.4. Modo de funcionamiento

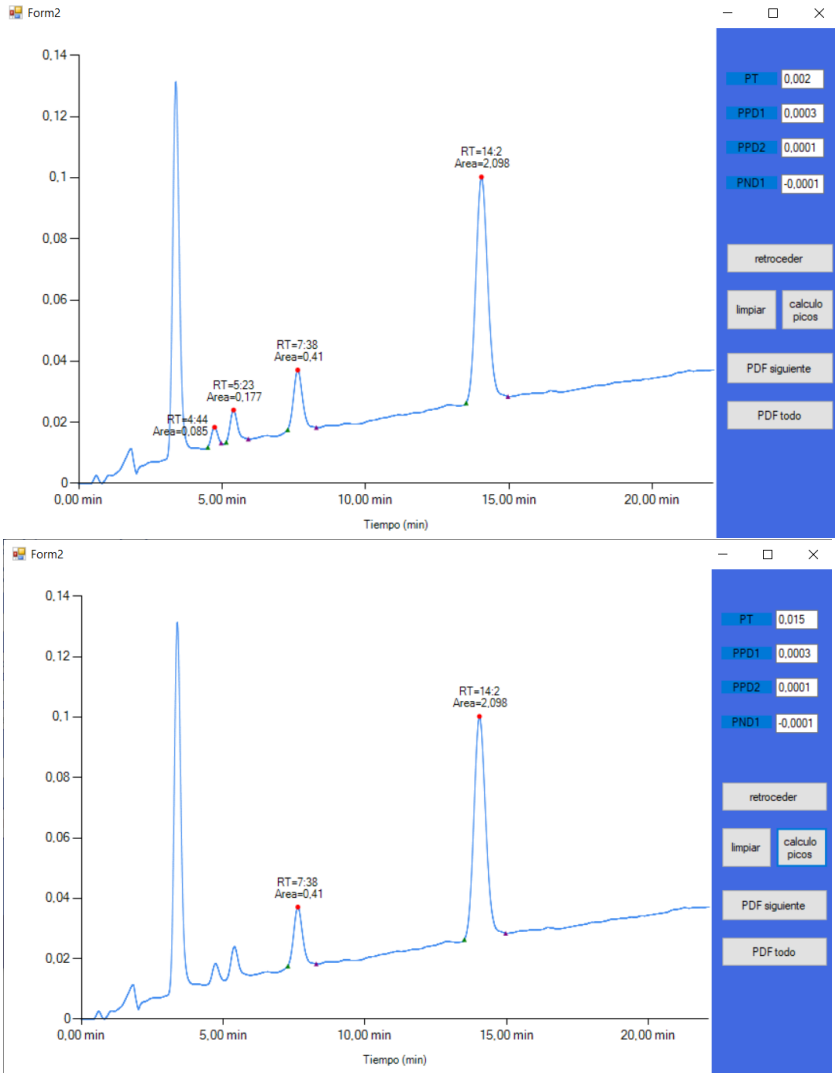

Figura 47 Modificación de los parámetros del cálculo de picos

Como se puede ver en la Figura 47 variando los parámetros de la función de detección de picos se pueden cambiar los picos que se detectan.

Es importante recordar que, si se quiere una representación de marcadores personalizados, se debe pulsar "PDF siguiente" para que se empiecen a guardar los marcadores personalizados y de la función de detección de picos.

A.5. FAQ

¿Se puede medir los picos en la pantalla 1?

No, el único fin de esta gráfica es poder ver en tiempo real el resultado de la muestra química. Para poder calcular los picos y el área se deberá esperar a que el automuestreador emita una señal de inicio de la siguiente muestra, se pulse el botón **desconectar** o el botón **next**

A.6. MANTENIMIENTO

Para realizar el mantenimiento, es necesario el siguiente equipo:

- Ordenador con un S.O Windows 7 o superior
- Voltímetro calibrado.
- Fuente de tensión regulable de rango 0V a 5V.

e. Cálculo de la no linealidad

Para comprobar la linealidad del sistema en la adquisición de voltaje, se deberá introducir en la entrada correspondiente un voltaje conocido y leer el resultado transmitido por el Arduino a la aplicación en el ordenador. Se anotarán los resultados en una tabla como la Tabla 6 y se procederá a su representación gráfica siendo el eje x “valor introducido” y el eje y “voltaje leído”

Tabla 6 Ejemplo de tabla de valores

Voltaje introducido	Voltaje leído

Entonces se procederá a hacer el cálculo de la “No Linealidad”. Para ello se aproximará una recta por mínimos cuadrados y se aplicará la siguiente ecuación:

$$Nl(\%) = \frac{|y_{max} - (m * x_{max} + n)|}{x_{max} - x_{min}} * 100\%$$

y_{max}	Valor leído máximo
x_{min}	Valor introducido mínimo
x_{max}	Valor introducido máximo
m	Pendiente de la recta ajustada
n	Valor de la intersección de la recta ajustada con el eje y

Se tomarán datos de 0 a 5 Voltios, pero la no linealidad sólo se realizará con valores de 0 a 3V y de 0 a 1V. Un valor de no linealidad superior al 50% será considerado como avería y el Arduino Nano deberá ser reemplazado.

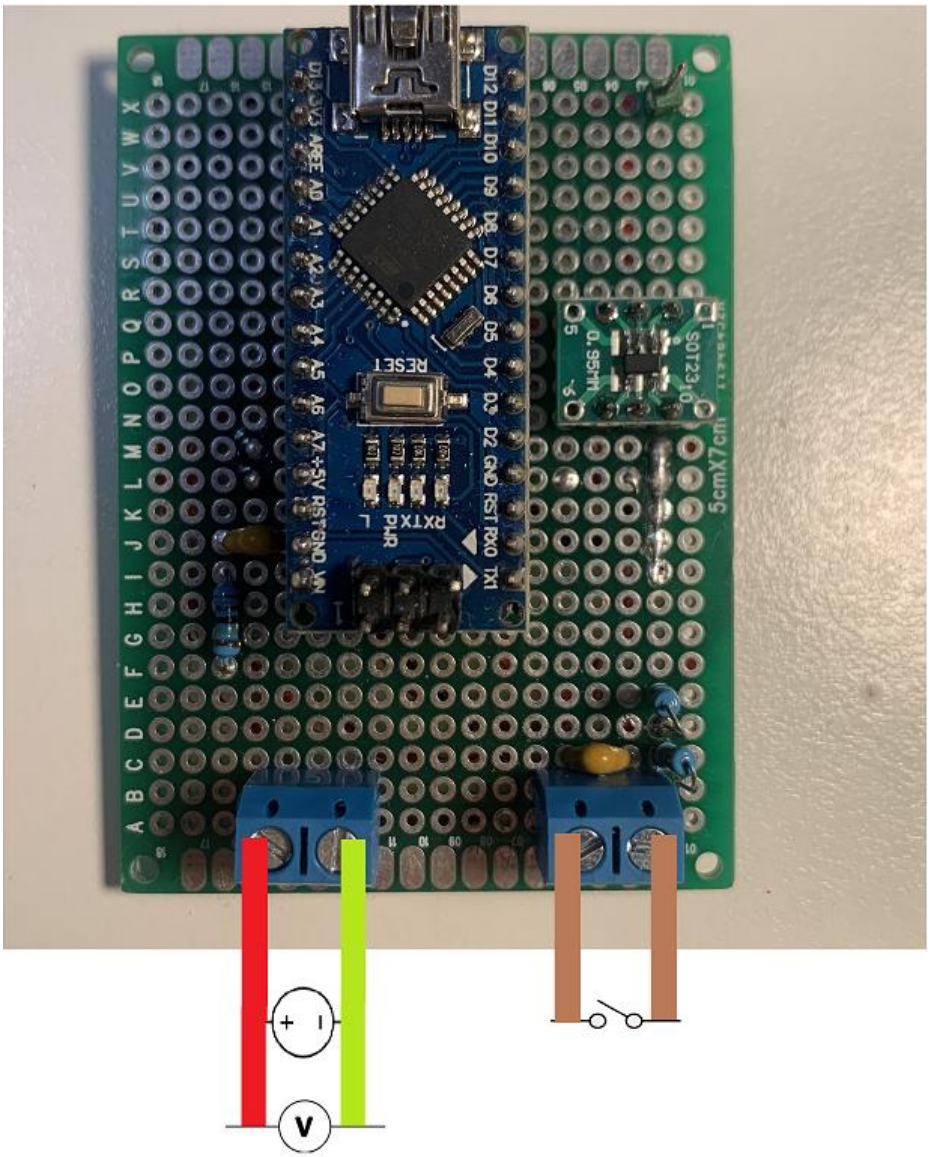

Figura 48 Conexionado para mantenimiento

A.7. GLOSARIO

Término	Descripción
PDF	PDF (Portable Document Format) es un formato de archivo diseñado para presentar y compartir documentos de manera electrónica de manera fiable y segura
COM	El puerto COM (Communication Port) es una interfaz de comunicación serial utilizada en los sistemas informáticos para la conexión de dispositivos externos como módems, impresoras, cámaras, GPS, entre otros. Los puertos COM se utilizan para transmitir y recibir datos de manera serial, es decir, un bit a la vez, a través de un cable que conecta el dispositivo externo con el ordenador.
USB	USB (Universal Serial Bus) es una interfaz de comunicación estándar utilizada en los sistemas informáticos para la conexión de dispositivos externos como ratones, teclados, unidades de almacenamiento, cámaras, smartphones y otros dispositivos electrónicos.

A.8. Contacto

Email: david.romart98@gmail.com

14. Anexo B. Código del programa.

14.1 Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.IO;
using System.IO.Ports;
using System.Threading;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;

namespace com
{
    public partial class btnread : Form
    {
        private TextWriter archivo_w1;
        private TextWriter archivo_w2;
        private TextReader archivo_r;

        public int Ts = 50;
        public const int orden_filtro = 2;
        private double[] filtro = new double[orden_filtro];
        public int i_arry = 0;
        public int i = 0;

        public int n_documento = 0;
        public UInt64 n_muestra = 0; //muestra química
        public UInt64 n_muestra_prev = 0;
        public int reading = 0;
        public int stop = 0;

        private delegate void serialCalback(string val);

        public string descriptor_r;
        public string descriptor_w;
        public string nombre, dir_trabajo;
        private static Mutex lstMutex = new Mutex();
        private static Mutex wrMutex = new Mutex();
        private static Mutex obMutex = new Mutex();
        private static Mutex flMutex = new Mutex();
        private static ReaderWriterLock fl_lock = new ReaderWriterLock();
        private string incomSting;

        private List<ulong> listA = new List<ulong>();
        private List<double> listB = new List<double>();

        public btnread()
        {
            InitializeComponent();
        }

        public void guardar()
        {
            using (StreamWriter archivo_w1 = new StreamWriter(nombre))
            {
                for (int i = 0; i < listA.Count; i++)
                {
                    archivo_w1.Write(listA[i]);
                    archivo_w1.Write(";");
```

```
                    archivo_w1.Write(listB[i]);
                    archivo_w1.Write(";");
                    archivo_w1.Write(Ts);
                    archivo_w1.Write("\n");
                }
            }
        }

        private void button5_Click(object sender, EventArgs e)  //leer
        {
            Form2 newForm = new Form2();
            newForm.ShowDialog();
        }

        private void button6_Click(object sender, EventArgs e)  //reiniciar
        {
            listA.Clear();
            listB.Clear();
            serialPort1.Write("1\n");
        }

        private void button6_Click_1(object sender, EventArgs e)    //conectar
        {
            listA.Clear();
            listB.Clear();
            timer1.Enabled = true;
            stop = 0;
            serialPort1.DataReceived -= serialPort1_DataReceived_1;//De no ser por esto habría 2
eventos lanzados
            serialPort1.DataReceived += serialPort1_DataReceived_1;
            if (!serialPort1.IsOpen)
            {
                try
                {
                    serialPort1.PortName = textBox1.Text.ToString();
                    serialPort1.BaudRate = 38400;
                    reading = 1;
                    serialPort1.Open();
                    serialPort1.Write("1\n");               //reiniciamos los parámetros del
arduino
                    while (!(serialPort1.BytesToRead == 0 && serialPort1.BytesToWrite == 0))
                    {
                        serialPort1.DiscardInBuffer();
                        serialPort1.DiscardOutBuffer();
                    }
                    serialPort1.DiscardInBuffer();
                    serialPort1.DiscardOutBuffer();

                    btnconnect.Enabled = false;
                    btndisconnect.Enabled = true;
                    btninit.Enabled = true;
                    button6.Enabled = true; //Botón next
                    //serialPort1.Write("1\n");
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        }

        private void setText(string val)
        {
            if (this.textBox2.InvokeRequired)
            {
                serialCalback scb = new serialCalback(setText);
                this.Invoke(scb, new object[] { val });
            }
```

```
        else
        {
            textBox2.Text = val;
        }
    }

    private void btnclose_Click(object sender, EventArgs e)
    {
        if(serialPort1.IsOpen) guardar();
        System.Windows.Forms.Application.ExitThread();
        this.Close();
    }

    public delegate void InvokeDelegate();

    private void button6_Click_2(object sender, EventArgs e) //desconectar
    {
        timer1.Enabled = false;
        guardar();

        Console.WriteLine("CERRAR");
        stop = 1;
        serialPort1.DataReceived -= serialPort1_DataReceived_1;
        while (!(serialPort1.BytesToRead == 0 && serialPort1.BytesToWrite == 0))
        {
            serialPort1.DiscardInBuffer();
            serialPort1.DiscardOutBuffer();
        }
        BeginInvoke(new InvokeDelegate(InvokeMethod));
    }

    public void InvokeMethod()
    {
        try
        {
            lock (serialPort1)
            {
                if (serialPort1.IsOpen)
                {
                    if (reading == 0)
                    {
                        serialPort1.Close();
                    }
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        btnconnect.Enabled = true;
        btndisconnect.Enabled = false;
    }

    private void textBox2_TextChanged(object sender, EventArgs e)
    {
    }

    private void serialPort1_DataReceived_1(object sender, SerialDataReceivedEventArgs e)
    {
        int a = 0;
        double s_filtrada = 0;
        DateTime d = DateTime.Today;

        // Se utiliza un bloqueo para garantizar la exclusión mutua
        // en el acceso al puerto serial.
        lock (serialPort1)
        {
```

```
                try
                {
                    // Se indica que se está leyendo del puerto serial.
                    reading = 1;

                    // Se lee el mensaje recibido del puerto serial.
                    incomSting = serialPort1.ReadLine();

                    // Se elimina el carácter de retorno de carro del mensaje.
                    incomSting = incomSting.Replace("\r", "");

                    // Se separan los valores del mensaje por el caracter ';'.
                    var values = incomSting.Split(';');
                    int len = incomSting.Length + 1;

                    // Se realiza el filtrado del valor recibido.
                    if (len > 15)
                    {
                        filtro[i] = Convert.ToDouble(values[1]) / 10000.0;
                        i++;

                        // Si se ha llegado al final del filtro, se vuelve a comenzar desde el
principio.
                        if (i == orden_filtro) i = 0;

                        // Se calcula el promedio del filtro.
                        s_filtrada = 0;
                        for (a = 0; a < orden_filtro; a++)
                        {
                            s_filtrada += filtro[a];
                        }
                        s_filtrada = s_filtrada / orden_filtro;

                        // Se espera a obtener el mutex para acceder a las listas de valores.
                        obMutex.WaitOne();
                        try
                        {
                            // Se convierten y añaden los valores a las listas de valores.
                            Ts = Int32.Parse(values[2]);
                            listA.Add(UInt64.Parse(values[0]) / 1);
                            listB.Add(s_filtrada / 100.0);
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show(ex.Message);
                        }

                        // Se libera el mutex para permitir a otros threads acceder a las listas.
                        obMutex.ReleaseMutex();

                        // Se espera a obtener el mutex para acceder al archivo de registro.
                        wrMutex.WaitOne();

                        try { n_muestra = UInt64.Parse(values[4]); }
                        catch (Exception ex) { MessageBox.Show(ex.Message); }

                        // Si el directorio de trabajo es nulo, se crea una carpeta para el día
actual.
                        if (dir_trabajo == null) //1º documento
                        {
                            d = DateTime.Today;
                            dir_trabajo = Directory.GetCurrentDirectory() + "\\" +
DateTime.Now.ToString("yyyy;MM;dd");
                            Directory.CreateDirectory(dir_trabajo);  //creamos una carpeta para
cada dia
                            nombre = dir_trabajo + "\\" + n_muestra + "-" + d.Day.ToString() +
"_" + d.Month.ToString() + "_" + d.Year.ToString() + "-" + DateTime.Now.ToString("HH;mm;ss tt") +
".csv";
```

```
                            archivo_w1 = TextWriter.Synchronized(new StreamWriter(nombre));
                            archivo_w1.Close();
                        }
                        // Si la muestra actual es diferente a la muestra previa, se crea un
nuevo archivo de registro.
                        else if (n_muestra != n_muestra_prev)    //nuevo documento
                        {
                            guardar();
                            nombre = dir_trabajo + "\\" + n_muestra + "-" + d.Day.ToString() +
"_" + d.Month.ToString() + "_" + d.Year.ToString() + "-" + DateTime.Now.ToString("HH;mm;ss tt") +
".csv";
                            archivo_w1 = TextWriter.Synchronized(new StreamWriter(nombre));
                            archivo_w1.Close();

                            listA.Clear();
                            listB.Clear();

                            // Envía el comando 5 indicando al Arduino que se ha aumentado la
muestra química
                            serialPort1.Write("5\n");
                            listA.Clear();
                            listB.Clear();
                            d = DateTime.Today;
                            nombre = dir_trabajo + "\\" + n_muestra + "-" + d.Day.ToString() +
"_" + d.Month.ToString() + "_" + d.Year.ToString() + "-" + DateTime.Now.ToString("HH;mm;ss tt") +
".csv";
                            obMutex.WaitOne();      //seccion critica para el fichero
                            archivo_w1 = TextWriter.Synchronized(new StreamWriter(nombre));
                            archivo_w1.Close();
                            n_muestra_prev = n_muestra;

                            // se libera el mutex para permitir que otros hilos accedan a la
sección crítica
                            obMutex.ReleaseMutex();
                        }
                        //liberacion sección crítica
                        wrMutex.ReleaseMutex();
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
            reading = 0;
            setText(incomSting);
        }

        // La siguiente función se utiliza para escribir en el archivo de guardado.
        private void R_W_files(object sender)
        {
            Console.WriteLine("RECIBIENDO");

            // Seccion critica para el fichero
            flMutex.WaitOne();
            try
            {
                fl_lock.AcquireWriterLock(1);

                archivo_w1.WriteLine(incomSting);
            }
            finally
            {
                fl_lock.ReleaseWriterLock();
            }

            // Libera la sección crítica
            flMutex.ReleaseMutex();
```

```
        }


        private Series mySeries = new Series("signal");

        private void timer1_Tick(object sender, EventArgs e)    //timer para representar la
gráfica
        {
            mySeries.Points.Clear();
            chart3.Series.Clear();
            // Como los valores de la lista listA están en nº de muestras es necesario hacer la
conversión
            // Para que se muestren en unidades de tiempo
            for (int j = 0; j < listA.Count; j++)
            {
                double xValue = (j * 50) / (60.0 * 1000.0);  // Valor en segundos
                double yValue = listB[j];                    // Valor en Voltios
                mySeries.Points.AddXY(xValue, yValue);
            }
            chart3.Series.Add(mySeries);

            chart3.ChartAreas[0].AxisX.MajorGrid.Enabled = false;
            chart3.ChartAreas[0].AxisX.MinorGrid.Enabled = false;
            chart3.ChartAreas[0].AxisY.MajorGrid.Enabled = false;
            chart3.ChartAreas[0].AxisY.MinorGrid.Enabled = false;

            // Obtener el número total de muestras
            int totalSamples = mySeries.Points.Count;

            // Calcular el valor máximo en minutos para el eje X
            double maxXValueInMinutes = totalSamples * 50.0 / 1000.0 / 60.0;

            // Definir el formato del label para mostrar minutos
            string labelFormat = "0.00' min'";

            // Asignar el nuevo label al eje X
            chart3.ChartAreas[0].AxisX.Title = "Tiempo (min)";
            chart3.ChartAreas[0].AxisX.LabelStyle.Format = labelFormat;
            chart3.Series["signal"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
        }

        public bool OptimizeOfLocalFormsOnly(System.Windows.Forms.Control chartControlForm)
//usado para agilizar los charts
        {
            if (!System.Windows.Forms.SystemInformation.TerminalServerSession)
            {
                SetUpDoubleBuffer(chartControlForm);
                return true;
            }
            return false;
        }

        private void button6_Click_4(object sender, EventArgs e)
        {
            // Envía un comando al Arduino para que aumente la muestra química
            serialPort1.Write("2\n");
        }

        public static void SetUpDoubleBuffer(System.Windows.Forms.Control chartControlForm)
//usado para agilizar los charts
        {
            System.Reflection.PropertyInfo formProp =
            typeof(System.Windows.Forms.Control).GetProperty("DoubleBuffered",
System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Instance);
            formProp.SetValue(chartControlForm, true, null);
        }
    }
```

```
}
```

14.2 Form2.cs

```
using iTextSharp.text;
using iTextSharp.text.pdf;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;

namespace com
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        public double Ts = 50;//poner 50
        public const int orden_filtro = 200;

        public double peak_threesold = 0.002;
        public double peak_pos1_der = 0.0003;
        public double peak_pos2_der = 0.0001;
        public double peak_neg_der = -0.0001;

        public Int32 x0 = 0, x1 = 0, x2 = 0;
        public Int16 contador = 0;
        public Int32[] markers = new Int32[50];
        public Int16 marker_i = 0;
        private List<double> listA = new List<double>();
        private List<double> listB = new List<double>();
        private double[] filtro = new double[orden_filtro];
        public string descriptor_r;
        public string descriptor_w;
        public string nombre, dir_trabajo;
        public string path_estandar;
        public Int16 n_pdf = 0;

        private Series series = new Series("signal");

        public Int32 n_peaks = 0;
        public int init_peak = 0;
        public double peak_init_value = 0.0;
        private List<int> x0_peak = new List<int>();        //inicio de pico
        private List<int> x1_peak = new List<int>();        //pos pico
        private List<int> x2_peak = new List<int>();        //fin pico
        private List<double> pos_peaks = new List<double>();

        // Esta función se utiliza para encontrar y registrar picos
        private void peak_finder()
        {
            int fin = 0;
            int fin_2 = 0;
```

```
            for (int i = 5000; i < listA.Count - 100; i++)// el 5000 inicial es porque antes de
ese tiempo se ignoran
            {
                //Se comprueba la derivada positiva con el valor PPD1
                if ((listB[i + 20] - listB[i]) >= peak_pos1_der) //derivada
                {
                    peak_init_value = listB[i + 1];
                    // Si supera el umbral se empieza a buscar el pico más alto
                    for (int j = i; fin != 1 && j < listA.Count - 100 && (j - i) < 50000; j++)
                    {
                        // Si la derivada negativa es menor que PND1, indicará que se ha llegado
al pico.
                        if ((listB[j + 20] - listB[j]) <= peak_neg_der)
                        {
                            // Si el valor del pico es mayor que el umbral, se guarda el valor
del pico y se marca en el gráfico
                            if ((listB[j] - peak_init_value) >= peak_threesold) //umbral de pico
                            {
                                pos_peaks.Add(j);
                                series.Points[i].MarkerStyle = MarkerStyle.Triangle;
                                series.Points[i].MarkerSize = 5;
                                series.Points[i].MarkerColor = Color.Green;
                                x0_peak.Add(i);
                                x1_peak.Add(j);
                                i = j;
                                // Se busca el final del pico con la derivada positiva PPD2
aprovechándo la característica de la línea base
                                for (int k = j; fin_2 != 1 && k < listA.Count - orden_filtro &&
(k - j) < 50000; k++)
                                {
                                    if ((listB[k + orden_filtro] - listB[k]) >= peak_pos2_der)
//fin del pico
                                    {
                                        series.Points[k].MarkerStyle = MarkerStyle.Triangle;
                                        series.Points[k].MarkerSize = 5;
                                        series.Points[k].MarkerColor = Color.Purple;
                                        fin_2 = 1;
                                        x2_peak.Add(k);
                                    }
                                }
                                fin_2 = 0;
                            }
                            fin = 1;
                        }
                    }
                    fin = 0;
                }
            }
        }
        // Esta función se utiliza para calcular el área de un pico
        public double area_calc(int n_x0, int n_x2)
        {
            double area = 0.0;
            int aux = 0;
            area = 0;
            // Cálculo del área por la regla del trapecio
            if (n_x0 > n_x2)
            {
                aux = n_x0;
                n_x0=n_x2;
                n_x2= aux;
            }
            for (int j = n_x0; j < n_x2 && j + 1 < listB.Count; j++)
            {
                area += (Ts * 1e-3) * ((listB[j + 1] - listB[j]) / 2 + listB[j]);
            }
            // Resta del área del offset
```

```
            area -= (Ts * 1e-3) * (n_x2 - n_x0) * (listB[n_x0] + (listB[n_x2] - listB[n_x0]) /
2); //le resto el área del offset

            return area;
        }
        // Esta función se utiliza para el cálculo del área de cada pico y su representación
        public double area_plot()
        {
            double area = 0.0;

            // Cálculo del área por la regla de todos los picos
            for (int i = 0; i < x2_peak.Count; i++)
            {
                area = 0;

                // Cálculo del área de cada pico
                for (int j = x0_peak[i]; j < x2_peak[i] && j + 1 < listB.Count; j++)
                {
                    area += (Ts * 1e-3) * ((listB[j + 1] - listB[j]) / 2 + listB[j]);
                }
                area -= (Ts * 1e-3) * (x2_peak[i] - x0_peak[i]) * (listB[x0_peak[i]] +
(listB[x2_peak[i]] - listB[x0_peak[i]]) / 2); //le resto el área del offset

                // Se marca el pico más alto y se representa el valor del área
                series.Points[x1_peak[i]].MarkerStyle = MarkerStyle.Circle;
                series.Points[x1_peak[i]].MarkerSize = 5;
                series.Points[x1_peak[i]].MarkerColor = Color.Red;
                string min = Math.Truncate((listA[x1_peak[i]] * Ts * 1e-3 / 60 - listA[0] * Ts *
1e-3 / 60)).ToString();
                string seg = Math.Truncate(((listA[x1_peak[i]] * Ts * 1e-3 / 60 - listA[0] * Ts *
1e-3 / 60) - Math.Truncate(listA[x1_peak[i]] * Ts * 1e-3 / 60 - listA[0] * Ts * 1e-3 / 60)) *
60).ToString();
                series.Points[x1_peak[i]].Label = "RT=" + min + ":" + seg
                    + "\n Area=" + Math.Round(area, 3).ToString(); ;
            }
            return area;
        }

        // Esta función lee un archivo y lo utiliza para generar una gráfica
        private void leer_y_plot(string FileName)
        {
            double s_filtrada = 0; // Variable para almacenar la señal filtrada
            int i = 0, a = 0; // Variables auxiliares para el filtrado
            using (var reader = new StreamReader(FileName)) // Abrir el archivo para leer
            {
                while (!reader.EndOfStream) // Leer el archivo mientras no se alcance el final
                {
                    var line = reader.ReadLine(); // Leer una línea del archivo
                    var values = line.Split(';'); // Separar los valores de la línea
                                                  // Si no es la última línea y el primer valor
no está vacío
                    if (!reader.EndOfStream && values[0] != "")
                    {
                        Ts = Convert.ToInt32(values[2]); // Obtener la tasa de muestreo
                        filtro[i] = Convert.ToDouble(values[1]); // Añadir el valor al filtro
                        i++;
                        if (i == orden_filtro) i = 0; // Reinciar el filtro cuando se llegue al
final
                        s_filtrada = 0;
                        for (a = 0; a < orden_filtro; a++)
                        {
                            s_filtrada += filtro[a]; // Calcular la suma de los valores del
filtro
                        }
                        s_filtrada = s_filtrada / (orden_filtro); // Calcular el valor promedio
de los valores del filtro
```

```
                        listA.Add(Convert.ToDouble(values[0])); // Añadir el valor de la muestra
a la lista
                        listB.Add(s_filtrada); // Añadir el valor de la señal filtrada a la lista
                    }
                }

                series.Points.Clear(); // Limpiar los puntos de la serie
                chart1.Series.Clear(); // Limpiar las series del gráfico
                for (int j = 0; j < listA.Count; j++) // Recorrer las muestras
                {
                    double xValue = (j * 50) / (60.0 * 1000.0);  // Calcular el valor en segundos
para el eje X
                    double yValue = listB[j];           // Obtener el valor de la muestra
                    series.Points.AddXY(xValue, yValue); // Añadir los valores al gráfico
                }
                chart1.Series.Add(series); // Añadir la serie al gráfico

                // Obtener el número total de muestras
                int totalSamples = series.Points.Count;

                // Calcular el valor máximo en minutos para el eje X
                double maxXValueInMinutes = totalSamples * 50.0 / 1000.0 / 60.0;

                // Definir el formato del label para mostrar en minutos en vez de nº muestras
                string labelFormat = "0.00' min'";

                // Asignar el nuevo label al eje X
                chart1.ChartAreas[0].AxisX.Title = "Tiempo (min)";
                chart1.ChartAreas[0].AxisX.LabelStyle.Format = labelFormat;

                chart1.ChartAreas[0].AxisX.MajorGrid.Enabled = false;
                chart1.ChartAreas[0].AxisX.MinorGrid.Enabled = false;
                chart1.ChartAreas[0].AxisY.MajorGrid.Enabled = false;
                chart1.ChartAreas[0].AxisY.MinorGrid.Enabled = false;

                chart1.Series["signal"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line; // Establecer el tipo de
gráfico de la serie
                peak_finder();
                area_plot();
            }
        }

        // Esta función se ejecuta al cargar el formulario
        private void Form2_Load(object sender, EventArgs e)
        {
            // Asigna los valores de los parámetros a los campos de texto correspondientes
            textBox1.Text = peak_threesold.ToString();
            textBox2.Text = peak_pos1_der.ToString();
            textBox3.Text = peak_pos2_der.ToString();
            textBox4.Text = peak_neg_der.ToString();

            // Establece el directorio actual como directorio inicial del cuadro de diálogo de
apertura de archivo
            openFileDialog1.InitialDirectory = Directory.GetCurrentDirectory();
            // Establece las extensiones de archivo permitidas para el cuadro de diálogo
            openFileDialog1.Filter = "Database files (*.csv, *.txt, *.xlsm)|*.xlsm;*.csv;*.txt";
            openFileDialog1.FilterIndex = 0;

            // Si se selecciona un archivo en el cuadro de diálogo correctamente
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                // Asigna la ruta del archivo seleccionado a la variable descriptor_r
                descriptor_r = openFileDialog1.FileName;
                // Asigna el directorio de trabajo a la ruta del archivo seleccionado
                dir_trabajo = openFileDialog1.InitialDirectory;
                // Restaura el directorio actual después de cerrar el cuadro de diálogo de
apertura de archivo
```

```
                openFileDialog1.RestoreDirectory = true;

                // Lee el archivo seleccionado y grafica los datos
                leer_y_plot(descriptor_r);
            }
            else
            {
                // Si se cancela la selección de archivo, se cierra el formulario
                this.Close();
            }
        }

        private ToolTip tooltip = new ToolTip();
        private Point? clickPosition = null;

        private void chart1_MouseMove(object sender, MouseEventArgs e)
        {
            if (clickPosition.HasValue && e.Location != clickPosition)
            {
                tooltip.RemoveAll();
                clickPosition = null;
            }
        }

        private void chart1_MouseClick(object sender, MouseEventArgs e)
        {
            // Guardamos la posición del clic del usuario
            var pos = e.Location;
            clickPosition = pos;
            double area = 0;
            // Comprobamos si el usuario ha hecho clic en el área de trazado del gráfico
            var results = chart1.HitTest(pos.X, pos.Y, false, ChartElementType.PlottingArea);

            // Obtenemos el valor inicial de x del gráfico (suponemos que está almacenado en
listA en la posición 1)
            x0 = Convert.ToInt32(listA[1]);

            // Recorremos los resultados obtenidos del método HitTest
            foreach (var result in results)
            {
                // Si el usuario ha hecho clic en el área de trazado...
                if (result.ChartElementType == ChartElementType.PlottingArea)
                {
                    // Obtenemos los valores de x e y del punto donde se hizo clic
                    Int32 xVal = (int)(60 * result.ChartArea.AxisX.PixelPositionToValue(pos.X) /
((double)Ts / 1000));
                    Int32 yVal = (int)result.ChartArea.AxisY.PixelPositionToValue(pos.Y);

                    // Modificamos el estilo y color del marcador correspondiente al punto donde
se hizo clic
                    series.Points[xVal - x0].MarkerStyle = MarkerStyle.Diamond;
                    series.Points[xVal - x0].MarkerSize = 12;
                    series.Points[xVal - x0].MarkerColor = Color.DarkBlue;

                    // Convertimos el valor de x del punto donde se hizo clic a minutos y lo
utilizamos como etiqueta para el punto
                    string min = Math.Truncate((Convert.ToDouble(xVal) * Ts * 1e-3 / 60 -
Convert.ToDouble(x0) * Ts * 1e-3 / 60)).ToString();
                    string seg = Math.Truncate(((Convert.ToDouble(xVal) * Ts * 1e-3 / 60 -
Convert.ToDouble(x0) * Ts * 1e-3 / 60) - Math.Truncate(Convert.ToDouble(xVal) * Ts * 1e-3 / 60 -
Convert.ToDouble(x0) * Ts * 1e-3 / 60)) * 60).ToString();
                    series.Points[xVal - x0].Label = min + ":" + seg + yVal.ToString();

                    // Contador para alternar entre el primer y segundo clic del usuario
                    Console.WriteLine(contador);
                    if (contador == 2) contador = 0;
```

```
                    // Si es el primer clic, almacenamos el valor de x en la posición actual del
array markers
                    if (contador == 0)
                    {
                        markers[marker_i] = xVal;
                        marker_i++;
                    }

                    // Si es el segundo clic, calculamos el área entre los dos valores de x y
actualizamos la etiqueta del punto correspondiente
                    if (contador == 1)
                    {
                        markers[marker_i] = xVal;
                        area = area_calc(markers[marker_i - 1], markers[marker_i]);
                        min = Math.Truncate((Convert.ToDouble(xVal) * Ts * 1e-3 / 60 -
Convert.ToDouble(x0) * Ts * 1e-3 / 60)).ToString();
                        seg = Math.Truncate(((Convert.ToDouble(xVal) * Ts * 1e-3 / 60 -
Convert.ToDouble(x0) * Ts * 1e-3 / 60) - Math.Truncate(Convert.ToDouble(xVal) * Ts * 1e-3 / 60 -
Convert.ToDouble(x0) * Ts * 1e-3 / 60)) * 60).ToString();
                        series.Points[markers[marker_i++] - x0].Label = "RT=" + min + ":" + seg +
"\n Area=" + Math.Round(area, 3).ToString();
                    }

                    contador++;
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)//clear   markers
        {
            // Itera entre todos los marcadores ya que para eliminarlos es necesario poner un
marcador vacío en su posición
            for (int i = 0; i < x0_peak.Count; i++)
            {
                series.Points[x0_peak[i]].MarkerSize = 0;
                series.Points[x0_peak[i]].Label = "";
            }
            for (int i = 0; i < x1_peak.Count; i++)
            {
                series.Points[x1_peak[i]].MarkerSize = 0;
                series.Points[x1_peak[i]].Label = "";
            }
            for (int i = 0; i < x2_peak.Count; i++)
            {
                series.Points[x2_peak[i]].MarkerSize = 0;
                series.Points[x2_peak[i]].Label = "";
            }
            x1_peak.Clear();
            x2_peak.Clear();
            x0_peak.Clear();
        }

        private void button2_Click(object sender, EventArgs e)// Botón calculo de picos
        {
            peak_finder();
            area_plot();
        }

        private void button3_Click(object sender, EventArgs e) // Boton retroceder
        {
            // Si hay marcadores, eliminar el último marcador y actualizar el contador
            if (marker_i > 0)
            {
                series.Points[markers[marker_i - 1] - x0].MarkerSize = 0;
                series.Points[markers[marker_i - 1] - x0].Label = "";
                marker_i--;
            }
            if (contador == 2) contador--;
```

```
            else if (contador == 1) contador--;
        }

        private string[] fileEntr;
        private Document doc_1 = new Document(iTextSharp.text.PageSize.A4, 10, 10, 42, 35);
        private bool ajuste = false;

        private void button4_Click(object sender, EventArgs e)  // Botón PDF siguiente
        {
            // Si aún no se ha realizado el ajuste, leer y graficar el primer archivo CSV y crear
un nuevo documento PDF
            if (ajuste == false)
            {
                var files = Directory.EnumerateFiles(Path.GetDirectoryName(descriptor_r), "*.*",
SearchOption.AllDirectories)
                    .Where(s => s.EndsWith(".csv"));
                fileEntr = files.ToArray();

                // Limpiar los datos del gráfico y leer los datos del primer archivo CSV
                chart1.Series.Clear();
                series.Points.Clear();
                x0_peak.Clear();
                x1_peak.Clear();
                x2_peak.Clear();
                pos_peaks.Clear();
                listA.Clear();
                listB.Clear();
                leer_y_plot(fileEntr[0]);

                // Crear un nuevo archivo PDF y abrirlo para escritura
                ajuste = true;
                PdfWriter wri = PdfWriter.GetInstance(doc_1, new
FileStream(Path.GetDirectoryName(descriptor_r) + "\\test.pdf", FileMode.Create));
                doc_1.Open();
                return;
            }

            // Si se han procesado todos los archivos CSV, guardar el último gráfico en el PDF y
cerrar el documento
            if (n_pdf == fileEntr.Length - 1)
            {
                var chartimage = new MemoryStream();
                chart1.SaveImage(chartimage, ChartImageFormat.Png);
                iTextSharp.text.Image Chart_image =
iTextSharp.text.Image.GetInstance(chartimage.GetBuffer());
                Chart_image.ScalePercent(55);
                Paragraph paragraph = new Paragraph("muestra" + (n_pdf).ToString());
                doc_1.Add(paragraph);
                doc_1.Add(Chart_image);
                doc_1.Close();

                // Abrir el archivo PDF en la aplicación predeterminada y cerrar la aplicación
actual
                Process.Start(Path.GetDirectoryName(descriptor_r) + "/test.pdf");
                this.Close();
            }
            else
            {
                // Si todavía hay archivos CSV por procesar, agregar la siguiente página del PDF
con el siguiente gráfico
                añadir_pdf(fileEntr[n_pdf + 1], fileEntr.Length, doc_1);
            }
        }

        private UInt16 presc = 0;

        private bool añadir_pdf(string fileName, int count, Document doc)
        {
```

```
            // Crear un MemoryStream para almacenar la imagen del gráfico
            var chartimage = new MemoryStream();

            // Guardar la imagen del gráfico en el MemoryStream
            chart1.SaveImage(chartimage, ChartImageFormat.Png);

            // Crear una instancia de iTextSharp Image a partir del MemoryStream para añadir la
imagen al documento
            iTextSharp.text.Image Chart_image =
iTextSharp.text.Image.GetInstance(chartimage.GetBuffer());

            // Escalar la imagen al 55%
            Chart_image.ScalePercent(55);

            // Crear un nuevo párrafo con un texto que incluye el número de muestra actual
            Paragraph paragraph = new Paragraph("muestra" + (n_pdf).ToString());

            // Agregar el párrafo y la imagen del gráfico al documento
            doc.Add(paragraph);
            doc.Add(Chart_image);

            // Incrementar el contador de presc para controlar el número de muestras por página
            presc++;

            // Si el número de muestras por página ha llegado a 2, agregar una nueva página al
documento
            if (presc == 2)
            {
                doc.NewPage();
                presc = 0;
            }

            // Limpiar los datos del gráfico y leer los datos del archivo CSV actual para el
siguiente gráfico
            chart1.Series.Clear();
            series.Points.Clear();
            x0_peak.Clear();
            x1_peak.Clear();
            x2_peak.Clear();
            pos_peaks.Clear();
            listA.Clear();
            listB.Clear();
            leer_y_plot(fileName);

            // Incrementar el contador de muestra química para el siguiente gráfico
            n_pdf++;

            // Si el contador de muestra ha llegado a count, devolver false para detener la
creación de más gráficos y PDFs
            if (n_pdf == count)
                return false;
            else
                return true;
        }

        private void button5_Click(object sender, EventArgs e)  //todoPDF
        {
            // Obtener todos los archivos CSV en el directorio del archivo descriptor_r y en
todos sus subdirectorios
            var files = Directory.EnumerateFiles(Path.GetDirectoryName(descriptor_r), "*.*",
SearchOption.AllDirectories)
                        .Where(s => s.EndsWith(".csv"));

            // Convertir el array de array de archivos en array de cadenas
            string[] fileEntries = files.ToArray();

            // Iterar a través de los archivos y hacer clic en el botón "PDF siguiente" para cada
uno
```

```
            for (Int16 k = 0; k < fileEntries.Length + 1; k++)
                button4_Click(sender, e);
        }

        private void button6_Click(object sender, EventArgs e)//Boton PDF next
        {
            button4_Click(sender, e);
        }

        private void textBox1_TextChanged_1(object sender, EventArgs e)
        {
            peak_threesold = Convert.ToDouble(textBox1.Text);
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
            peak_pos1_der = Convert.ToDouble(textBox2.Text);
        }

        private void textBox3_TextChanged(object sender, EventArgs e)
        {
            peak_pos2_der = Convert.ToDouble(textBox3.Text);
        }

        private void textBox4_TextChanged(object sender, EventArgs e)
        {
            peak_neg_der = Convert.ToDouble(textBox4.Text);
        }

    }
}
```

15. Referencias

[1] R. Malviya, V. Bansal, O. Pal, y P. Sharma, «High performance liquid chromatography: A short review», *Journal of Global Pharma Technology*, vol. 2, pp. 22-26, jun. 2010.

[2] «HITACHI, "INSTRUCTION MANUAL FOR MODEL L-6200 INTELLIGENT PUMP," Tokyo.».

[3] M. W. Dong, *HPLC and UHPLC for Practicing Scientists*. John Wiley & Sons, 2019.

[4] «Maxim > App Notes > A/D and D/A CONVERSION/SAMPLING CIRCUITS Understanding SAR ADCs», 2001.

[5] John Errington's, «How the Arduino ADC measures an input voltage».

[6] S. W. Smith, *Digital Signal Processing: A Practical Guide for Engineers and Scientists*, vol. 17, n.º 2. 2003.

[7] J. H. Huijsing y D. Linebarger, «Low-voltage operational amplifier with rail-to-rail input and output ranges», *IEEE J Solid-State Circuits*, vol. 20, n.º 6, pp. 1144-1150, 1985, doi: 10.1109/JSSC.1985.1052452.

[8] «CHROMASTER 5260 AUTOSAMPLER INSTRUCTOR MANUAL».

Universidad de Alcalá

Escuela Politécnica Superior

ESCUELA POLITECNICA SUPERIOR