Capítulo 4

SOFTWARE INVOLUCRADO

4. SOFTWARE INVOLUCRADO

4.1. INTRODUCCIÓN

En este capítulo se pretende hacer una descripción del software que se ha empleado para el buen desarrollo del presente proyecto. Se comentará el sistema operativo. También se realizará una breve descripción del entorno de programación (Labview).

4.2. SISTEMA OPERATIVO

En el presente apartado se discutirá cuando es necesario o no un sistema en tiempo real, para terminar definiendo el sistema operativo usado en este proyecto así como sus razones principales.

4.2.1. DISCUSIÓN SOBRE LA NECESIDAD O NO DE UN SISTEMA OPERATIVO EN TIEMPO REAL

Una solución de tiempo real es requerida para aplicaciones que deben ejecutar determinísticamente una tarea en software sin interrupciones o interferencias de otras tareas no-críticas. Las aplicaciones que se ejecutan determinísticamente muchas veces realizan una tarea crítica en iteraciones y

estas iteraciones consumen siempre la misma cantidad de tiempo del procesador. De esta manera las aplicaciones determinísticas son apreciadas no tanto por su velocidad, sino por su confiabilidad a responder con consistencia a entradas y generar salidas con muy poco "parpadeo" o jitter. (Jitter es el término que se usa para identificar el promedio de las diferencias en tiempo que le toma a un ciclo ejecutarse).

Un ejemplo común de una aplicación determinística es un lazo de control de tiempo crítico. Un lazo de control de tiempo crítico adquiere información de un sistema físico y responde a esa información con una salida de tiempo muy precisa. Por ejemplo, en la el hangar de ensamblaje de piezas de avión, donde cientos de piezas se ensamblan al día. A medida que dos piezas son unidas desde sus dos puntas, un torque es requerido para girar las piezas, embobinen y queden interconectadas. Suponga que la máquina que conecta las piezas usa un lazo de control que responde a los incrementos en la resistencia entre las piezas aplicando más torque. Ahora suponga que cuando se llega a un nivel crítico de torque, el lazo de control responde terminando su tarea. Bajo estas condiciones, el lazo debe ejecutarse determinísticamente porque un retraso en el software puede causar daños severos a las piezas y al equipo. La gráfica 4.1 que se muestra a continuación ilustra torque contra tiempo en LabVIEW, note el abrupto cambio de nivel el cual corresponde a la conexión completa entre las dos piezas.

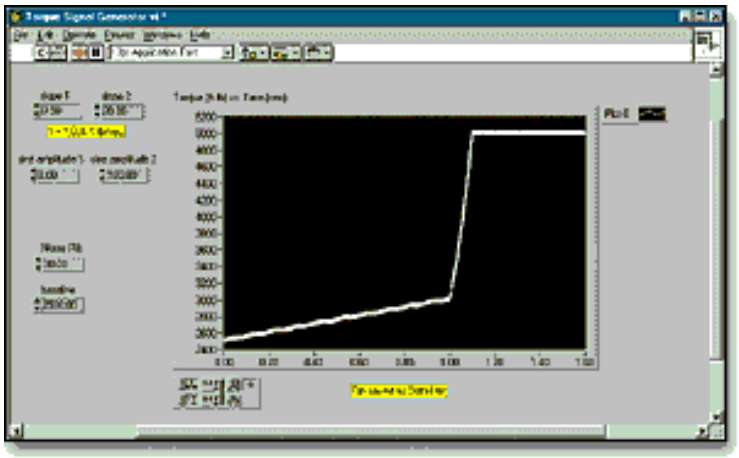

Figura 4.1. *Ejemplo Labview. Torque vs Tiempo. Necesidad de un sistema en tiempo real.*

Debe de usarse un sistema operativo de tiempo real para garantizar que la tarea crítica se ejecuta de manera precisa en el horario establecido. El sistema operativo de Windows esta diseñado para ofrecer o compartir tiempo del procesador a tareas que corren en el fondo. Como en este caso el usuario tiene muy poco control sobre cómo el procesador comparte su tiempo, Windows no es considerado un sistema operativo de tiempo real.

De esta manera, tareas no críticas (como un protector de pantalla o software de protección de virus) pueden interferir con un proceso de tiempo crítico en LabVIEW, como una aplicación de adquisición de datos o un lazo de control. Pensemos en el siguiente ejemplo. Si se le pide a Windows que abra un Web browser (como Netscape o Internet Explorer), el procesador suspende temporalmente la ejecución de LabVIEW para empezar a ejecutar las instrucciones requeridas para abrir Internet Explorer.

Desde el punto de vista del usuario, esto tal vez no represente un cambio visible en el panel frontal de LabVIEW, pero si varias instancias de Internet Explorer ocurren, el usuario eventualmente notaría un retraso en la interfaz al usuario.

Al considerar soluciones de prueba y medición para una aplicación, debe de ser capaz de distinguir entre las aplicaciones que requieren procesamiento en tiempo real y aquellas que no lo necesitan. Algunos sistemas de adquisición de datos no requieren procesamiento en tiempo real porque no involucran un proceso de tiempo crítico en software. Considere una aplicación en donde se debe adquirir datos, procesar y almacenar a disco continuamente a una razón de 1 Mmuestra/s. La aplicación no responde con una salida, simplemente colecciona datos muy rápido con un tiempo muy preciso. National Instruments cuenta con las tarjetas de multifunction de la serie E (MIO) las cuales pueden realizar esta tarea sin necesidad de tecnología de tiempo real. Esta aplicación solamente necesita relojes en hardware precisos para asegurar que las muestras se digitalizan en el intervalo de tiempo especificado. Una vez digitalizada la señal, la tarjeta MIO almacena los valores en un buffer FIFO en

hardware el cual se empieza a llenar. Una vez que el buffer FIFO llega a su nivel crítico, los datos son transferidos en bloques a un buffer en RAM hasta que LabVIEW los lee del buffer en RAM. La razón a la cual se leen los datos del buffer en RAM dependen de qué tan rápido LabVIEW puede "leer" los datos en un ciclo, el cual a su vez es gobernado por la velocidad del procesador y otros factores como: Escritura/Lectura de archivos, Procesamiento de señales, Tareas corriendo en el fondo (como protector de pantalla, Web browser, software anti-virus, etc.), Actividad en la interfaz al usuario, Otras comunicaciones de hardware (como GPIB, serial RS-232, etc.)

A medida que el programador asigna más tareas al lazo, las iteraciones del lazo requieren de más tiempo para completar. De la misma manera, a medida que el usuario interactúa con otros programas, genera eventos en la interfaz de usuario y realiza otras entradas/salidas de hardware la velocidad de ejecución del lazo se volverá más lenta.

Esto se debe a que el sistema operativo debe asignar y compartir el tiempo del procesador entre un mayor número de instrucciones. Ahora, asuma que existe un mínimo de actividad en el bus de datos y solamente hay algunas otras aplicaciones corriendo además de LabVIEW. Bajo estas condiciones, la mayoría de las computadoras modernas pueden fácilmente leer del buffer en RAM a una razón suficiente para evitar que los datos sean sobre escritos o perdidos, esto es asumiendo que la tarjeta MIO esta adquiriendo a su velocidad máxima. Además, típicamente cada vez que los datos son leídos de la RAM el procesador cuenta con tiempo extra para realizar otras tareas como almacenar datos a disco y procesamiento de señales. A medida que las condiciones varían, el usuario empíricamente determina las limitaciones de la computadora y ajusta los parámetros adecuadamente. El punto a recalcar es que aún con las interrupciones de software de otros procesos la adquisición de datos o proceso en hardware consistentemente adquiere a una razón de 200 KS/s. La razón de esto es que la tarjeta MIO cuenta con temporizadores/contadores dedicados que regulan las conversiones análogas-digitales (A/D) así que no hay posibilidad de que otro proceso en software afecte esta velocidad. Pero como

mencionamos, el lazo en LabVIEW que lee los datos del buffer en RAM es susceptible a dichas interrupciones.

El usuario debe asegurarse que otras aplicaciones no tomen todo el valioso tiempo de procesador que LabVIEW requiere para coleccionar los datos que están en RAM. La lección que se aprende en este apartado es que el lazo que lee en LabVIEW no se ejecuta determinísticamente pero los datos son adquiridos determinísticamente por el hardware. De hecho, es muy improbable que dos operaciones de lectura tomen el mismo número de ciclos de reloj ya que el sistema operativo asigna remandas de tiempo a otros procesos que están sucediendo al mismo tiempo. Y mientras el procesador pueda regresar al lazo de lectura frecuentemente para leer los datos en RAM, la razón de adquisición de datos en hardware no pasará al tiempo promedio de lectura en software.

La presencia de un buffer en RAM elimina el riesgo de pérdida de datos cuando el tiempo del procesador se comparte con otras aplicaciones. Esta técnica es conocida como búferes acquisition o adquisición con buffer y es muy común en la adquisición de datos. Esta discusión ilustra cuando un sistema no necesita incorporar tecnología de tiempo real. Hoy en día muchos sistemas de adquisición de datos operan de manera similar al ejemplo expuesto porque una adquisición controlada por un reloj en hardware resulta datos que están espaciados de manera precisa y los cuales pueden ser procesados posteriormente. Pero cuando la aplicación "como el ejemplo del torque en las piezas" adquiere señales las cuales deben de ser procesadas inmediatamente y deben incorporar una señal de respuesta rápida y muy precisa, entonces es una aplicación de tiempo real y típicamente requiere de hardware y software especial de tiempo real. Este no es el caso de la cámara de hipoxia, donde el tiempo no es un factor crucial.

4.2.2. WINDOWS 2000 PROFESSIONAL

El sistema operativo instalado en el ordenador donde se encuentra la tarjeta de adquisición de datos para la cámara de hipoxia crónica es la versión profesional de Windows 2000 y en este apartado se pretende exponer las razones que justifican la elección de este sistema operativo, una descripción detallada del mismo se sale del alcance de este proyecto.

Hay dos motivos fundamentales que han llevado a la elección de este sistema:

Por un lado la seguridad ya que dispone de un sistema de seguridad que permite la protección de datos del usuario tanto de forma local como a través de una red de área local, redes telefónicas o Internet. Dispone de un sistema de control de acceso de los usuarios al sistema, pudiendo existir varios tipos de usuarios que usen el mismo sistema cada uno con distintos privilegios. Esto lo hace ideal para el caso de que como ampliación del proyecto, cámara se pueda acceder a través de internet, pudiendo protegerlo de un mal uso.

Por último cabe destacar su versatilidad ya que combina la potencia y seguridad de su predecesor Windows NT con la facilidad de uso del Windows 98.

4.3. Herramientas de desarrollo

Las herramientas con las que se ha trabajado para la elaboración de la aplicación de control y monitorización de la cámara de hipoxia ha sido principalmente con Labview. Por su importancia en el presente proyecto, se ofrece a continuación una breve descripción de estos entornos de programación de National Instruments.

4.3.1. LABVIEW

A diferencia de los lenguajes de propósito general (Visual C, Visual Basic, etc...) , Labview se encuentra totalmente orientado al control de instrumentación, siendo perfecto para el desempeño del presente proyecto.

4.3.1.1. DEFINICIÓN DE LABVIEW

LabVIEW quiere decir *Laboratory Virtual Instrument Engineering Workbench* en inglés, y es una herramienta de desarrollo basado en un lenguaje de programación gráfica que usa iconos en vez de líneas de texto para crear aplicaciones.

Es un entorno de desarrollo especialmente diseñado para crear aplicaciones de medida, automatización y control, con funcionalidades específicas de adquisición de datos, control de instrumentos, análisis de medidas y representación de datos.

LabVIEW está completamente integrado para comunicaciones con el hardware, tales como: GPIB, VXI, RS232, RS485 y tarjetas de adquisición de datos y también tiene múltiples librerías para estándares de software como: redes TCP/IP y Active X.

En contraste con los lenguajes de programación basados en texto, donde las instrucciones determinan la ejecución del programa, LabVIEW usa una programación de flujo de datos, en la que el flujo de datos determina la ejecución de la aplicación.

La programación tradicional basada en texto en un diseño *top-down*, en el que se debe escribir el código que se ejecuta línea por línea. En vez de una ejecución secuencial LabVIEW se basa en el concepto de flujo de datos.

Los archivos básicos creados con LabVIEW se denominan Instrumentos Virtuales o VI (siglas en inglés de *Virtual Instruments*), porque su aspecto y

funcionamiento imitan a instrumentos reales como los ya descritos en el capítulo dos de este documento.

Un VI contiene los siguientes tres elementos:

Panel Frontal: que sirve como interfaz de usuario de la VI.

Diagrama de bloques: que contiene el código fuente gráfico que define la funcionalidad del instrumento virtual.

Icono y panel de conectores: que identifica al VI y sirve para interconectar VIs.

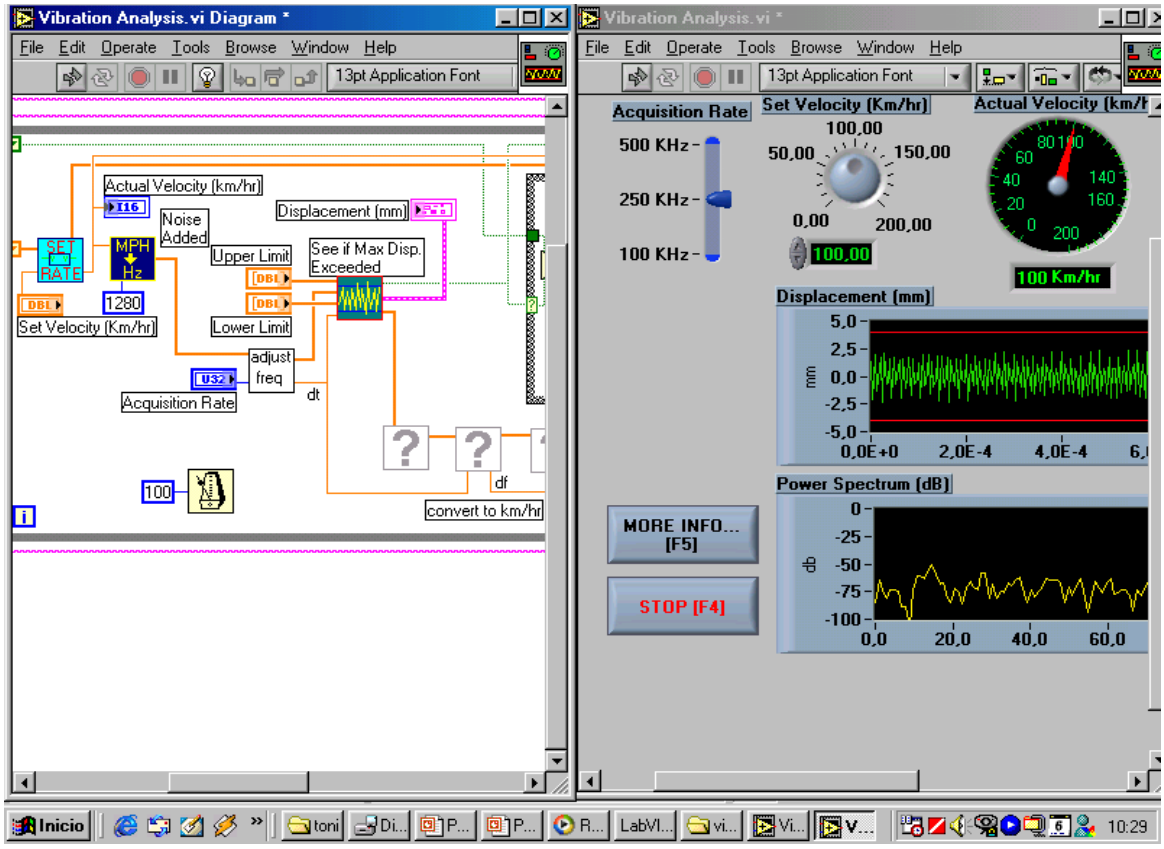

Figura 4.2. *Panel de control y diagrama de bloques*

En LabVIEW se diseña la interfaz de usuario usando un conjunto de herramientas de diseño y objetos, y es lo que se conoce como panel frontal. Se le añade el código usando representaciones gráficas de funciones para controlar los objetos del panel frontal. El diagrama de bloques es el que

contiene ese código fuente gráfico que en algunos aspectos se parece a un diagrama de flujo.

En el panel frontal se diseña con controles e indicadores, que son las variables que maneja el instrumento virtual. Los controles pueden ser botones, interruptores, *knobs*, diales y otros dispositivos de entrada mientras que los indicadores son LEDs, gráficos y otros displays. Los controles simulan dispositivos de entrada y suministran los datos al diagrama de bloques del VI, en cambio los indicadores simulan dispositivos de salida y muestran los datos adquiridos o generados por el diagrama de bloques. Todos los VI usan funciones que manipulan la entrada de la interfaz de usuario u otras fuentes y muestra esa información o la mueve a otros ficheros u otros ordenadores.

En el diagrama de bloques se encuentra el código fuente. Este código se implementa de una forma gráfica, a veces puede ser similar a un diagrama de flujo. Para ello se cuenta con una serie de librerías que aporta la opción de funtion palette, la cual aporta funciones de muy variadas características como aritméticas, de adquisición de datos, filtrado, lógicas, de almacenamiento de datos...., lo cual ayuda enormemente en el desarrollo de cualquier nueva aplicación.

4.3.1.2. CONCEPTO DE INSTRUMENTACIÓN VIRTUAL

Muchas veces la realización de una medida requiere la intervención de varios equipos e instrumentos electrónicos; unos generan estímulos sobre el dispositivo que se pretende medir, y otros recogen la respuesta a estos estímulos. Este conjunto de instrumentos que hace posible la realización de la medida, recibe el nombre de sistema de instrumentación IS (Instrumentation System). Todo sistema de instrumentación consta de unos equipos, un sistema de interconexión de éstos y un controlador inteligente que gestiona el funcionamiento de todo el sistema y da las órdenes para que una medida se realice correctamente.

La utilización manual de instrumentos para realizar medidas es prácticamente un hecho aislado, sólo en los procesos de investigación y desarrollo de nuevos prototipos, o en entornos docentes es una práctica habitual. A nivel industrial las medidas para el control de un determinado proceso, las pruebas funcionales sobre un equipo o el control de calidad de la producción se realizan de manera automática. La automatización de las medidas requiere que los equipos gocen de un cierto grado de inteligencia para que puedan ser gobernados por un controlador que se comunica con los instrumentos a través de un BUS de instrumentación (GPIB, VXI, RS232...).

La compañía National Instruments ha definido un instrumento virtual (IV) de la siguiente manera:

“Un instrumento que no es real, se ejecuta en una computadora y tienes sus funciones definidas por software.” (National Instruments, 2001).

El concepto de instrumentación virtual nace a partir del uso del computador personal (PC) como "instrumento" de medición de tales señales como: temperatura, presión, caudal, etc. Es decir, el PC comienza a ser utilizado para realizar mediciones de fenómenos físicos representados en señales de corriente (Ej. 4-20mA) y/o voltaje (Ej. (0-5Vdc).

Sin embargo, el concepto de "instrumentación virtual" va más allá de la simple medición de corriente o voltaje, sino que también involucra el procesamiento, análisis, almacenamiento, distribución y despliegue de los datos e información relacionados con la medición de una o varias señales específicas. Es decir, el instrumento virtual no se conforma con la adquisición de la señal, sino que también involucra la interfaz hombre-máquina, las funciones de análisis y procesamiento de señales, las rutinas de almacenamiento de datos y la comunicación con otros equipos.

Un ejemplo; el osciloscopio tradicional tiene una funcionalidad ya predefinida desde la fábrica donde lo diseñan, producen y ensamblan. Es decir,

la funcionalidad de este tipo de instrumento es definida por el fabricante del equipo, y no por el usuario mismo. El término "virtual" nace precisamente a partir del hecho de que cuando se utiliza el PC como "instrumento" es el usuario mismo quién, a través del software, define su funcionalidad y "apariencia" y por ello se dice que "virtualizamos" el instrumento, ya que su funcionalidad puede ser definida una y otra vez por el usuario y no por el fabricante. Incluso por medio del instrumento virtual se pueden realizar funciones que no están disponibles mediante el uso manual del instrumento real.

El instrumento virtual es definido entonces como una capa de software y hardware que se le agrega a un PC en tal forma que permite a los usuarios interactuar con la computadora como si estuviesen utilizando su propio instrumento electrónico "hecho a la medida".

¿Cómo construir un instrumento virtual? Para construir un instrumento virtual, sólo se requiere un PC, una tarjeta de adquisición de datos con acondicionamiento de señales (PCMCIA, ISA, XT, PCI, etc.) y el software apropiado, los tres elementos clave en la conformación de un instrumento virtual, teniendo un chasis de acondicionamiento de señales como elemento opcional. El "acondicionamiento de señales" es opcional, porque dependiendo de cada señal y/o aplicación, se puede o no requerir amplificación, atenuación, filtraje, aislamiento, etc. de cada señal. Si la señal está en el rango de los +/- 5Vdc y no se requiere de aislamiento o filtraje, la misma puede ser conectada directamente la tarjeta de adquisición de datos.

En el instrumento virtual, el software es la clave del sistema, a diferencia del instrumento tradicional, donde la clave es el hardware. Con el sistema indicado anteriormente, se podría construir un osciloscopio "personalizado", con la interfaz gráfica que uno desee, agregándole inclusive más funcionalidad. Sin embargo, este mismo sistema puede también ser utilizado en la medición de temperatura, o en el control de arranque/parada de una bomba centrífuga. Es allí donde radica uno de los principales beneficios del instrumento virtual, su

flexibilidad. Este instrumento virtual no sólo permite visualizar la onda, sino que a la vez permite graficar su espectro de potencia en forma simultánea. Sería complicado poder hacer algo así con un instrumento convencional. Para finalizar, la siguiente tabla (ver Tabla 4.1) indica algunas de las principales diferencias entre el instrumento convencional o tradicional, y el instrumento virtual.

Instrumento Tradicional	Instrumento Virtual
Definido por el fabricante	Definido por el usuario
Funcionalidad específica, con conectividad limitada.	Funcionalidad ilimitada, orientado a aplicaciones, conectividad amplia.
Hardware es la clave.	Software es la clave
Alto costo/función	Bajo costo/función, variedad de funciones, reusable.
Arquitectura "cerrada"	Arquitectura "abierta".
Lenta incorporación de nuevas tecnología.	Rápida incorporación de nuevas tecnologías, gracias a la plataforma PC.

Instrumento Tradicional	Instrumento Virtual
Bajas economías de escala, alto costo de mantenimiento.	Altas economías de escala, bajos costos de mantenimiento.

Tabla 4.1. *Diferencias instrumento virtual vs tradicional*

La flexibilidad, el bajo costo de mantenimiento, la reusabilidad, la personalización de cada instrumento, la rápida incorporación de nuevas tecnologías, el bajo costo por función, el bajo costo por canal, etc... son algunos de los beneficios que ofrece la instrumentación virtual.

La instrumentación virtual puede también ser implementada en equipos móviles (laptops), equipos distribuidos en campo (RS-485), equipos a distancia (conectados vía radio, ethernet, etc.), o equipos industriales (NEMA 4X, etc.). Existe una tarjeta de adquisición de datos para casi cualquier bus o canal de comunicación en PCs (ISA, PCI, USB, serial RS-232/485, paralelo EPP, PCMCIA, CompactPCI, etc.).

Por ejemplo, las técnicas utilizadas normalmente para evaluar las características de medición de un multímetro digital (DMM o Digital Multi-Meter) pueden ser utilizadas para evaluar las características de medición de un instrumento virtual (VMM o Virtual Multi-Meter). Entre dichas características se encuentran las siguientes:

	DMM	**VMM con tarjeta especializada**	**VMM con tarjeta de propósito general**
Hardware utilizado	HP 34401 A DMM	DAQCard 4050	PCI-MIO-16XE-10
No. de Canales	1	1	16 (Diferencial)

	DMM	**VMM con tarjeta especializada**	**VMM con tarjeta de propósito general**
Conversión AC	True RMS	True RMS	True RMS (por software)
Resolución (convertidor de 16-bits)	61/2 - 41/2 dígitos	51/2 dígitos	41/2 dígitos
Rango de entrada (ACV)	100 mV - 750 V	20 mV - 250 V	100mV - 250 V (con acondicionamiento SCXI)
Sensibilidad (ACV)	0.1 uV	0.1 uV	1.5 uV
Rango de Entrada (DCV)	100 mV - 1000 V	20 mV - 250 V	100 mV - 250 V
Sensibilidad (DCV)	0.1 uV	0.1 uV	1.5 uV
NMRR	60 dB	80 dB	variable (80-120 dB)
CMRR	70 dB (AC), 140 dB (DC)	90 dB (AC), 30 dB (DC)	variable (80-120 dB)
Velocidad de medición (lecturas/seg.)	5-1 K lecturas/seg	10, 50 , 60 K lecturas/seg	100 K lecturas/seg

Tabla 4.2. *Ejemplo Multímetro Digital Tradicional vs Multímetro Virtual*

Un instrumento virtual puede realizar las tres funciones básicas de un instrumento convencional: adquisición, análisis y presentación de datos. Sin

embargo, el instrumento virtual permite personalizar el instrumento, y agregarle mucha más funcionalidad sin incurrir en costos adicionales. El instrumento virtual se apalanca en la flexibilidad y poder del PC, y mediante el software que lo acompaña, el nivel de adaptabilidad y personalización del instrumento virtual es casi ilimitado.

4.3.1.3. ENTORNO DE TRABAJO DE LABVIEW

Los elementos de trabajo básicos del entorno *LabVIEW* son los menús de la ventana de aplicación, la paleta de herramientas, la paleta de controles y la paleta de funciones.

Menús de Ventanas: En los menús ventanas es análogo al de las aplicaciones de Windows con la incorporación de nuevas funcionalidades específicas de LabVIEW. Es donde están recogidas las funcionalidades típicas como operaciones de fichero (nuevo, guardar, propiedades, etc.), operaciones de edición (cortar, pegar, etc.), temas específicos de operación, herramientas, buscador, aspectos de ventana y de ayuda.

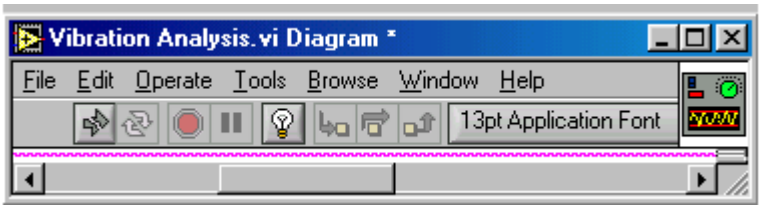

Figura 4.3. *Menús de ventana*

Paleta de Herramientas: Es accesible tanto desde el panel frontal como desde el diagrama de bloques. Una herramienta de dicha paleta es un modo de operación especial del cursor del ratón. Al seleccionar una herramienta el icono del cursor cambia su aspecto al de la herramienta seleccionada. Las Herramientas se usan para modificar los objetos del panel frontal y del

diagrama de bloques. En la figura 3.2 se puede ver el aspecto de la *Tool Palette*.

Paleta de Controles: Tan sólo es accesible desde el panel frontal, y contiene los controles e indicadores que se usan para diseñar la interfaz de usuario. En la figura 3.2 se puede ver esta paleta de controles.

Paleta de Funciones: Es accesible sólo desde el diagrama de bloques y contiene los objetos utilizados para programar las VIs, tales como funciones aritméticas, lógicas, de cadenas de caracteres, de manejo de ficheros, bucles, adquisición de datos, etc. En la figura 4.4 se puede ver el aspecto de la paleta de funciones.

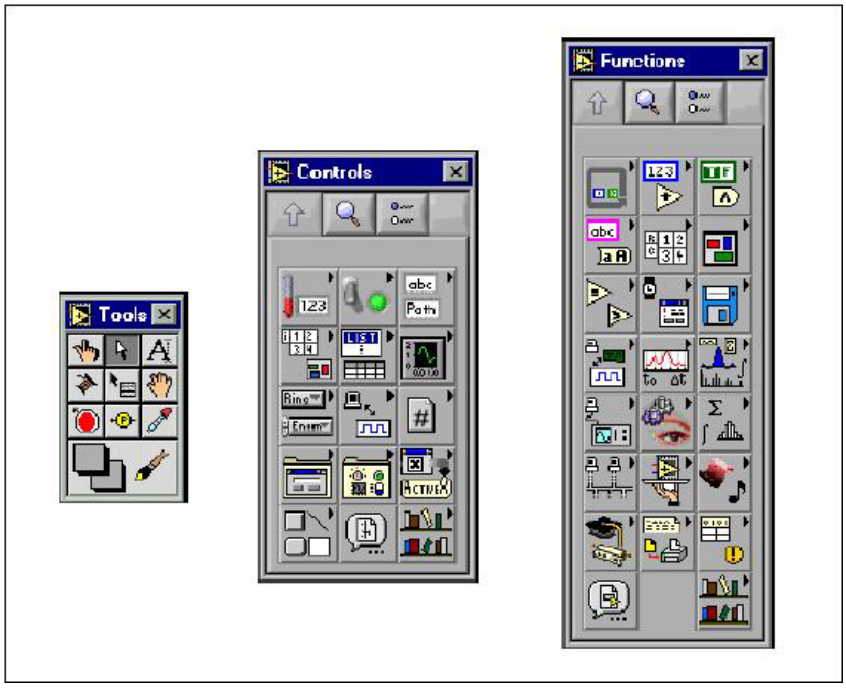

Figura 4.4. *De Izquierda a Derecha: Paletas de Herramientas, Controles y Funciones*

4.3.1.4. PRINCIPALES MOTIVOS DE LA ELECCIÓN DE LABVIEW

Los sistemas automáticos y de medida tradicionales consisten en un conjunto de instrumentos caros y cerrados diseñados para una tarea

específica. Normalmente en estos sistemas es necesario diseñar todo desde abajo a arriba, y programar todo el software de control del sistema que es totalmente dependiente del hardware. En los sistemas tradicionales el hardware define el sistema.

Al usar instrumentos basados en computadores cambia todo, ya que se pueden sistemas automáticos y de medida con la ventaja de usar una tecnología de PC, flexible y de bajo costo.

El concepto de instrumentación virtual permite al usuario diseñar su propia solución mediante un software de desarrollo integrado en el ordenador para una amplia variedad de hardware de medida. Con una solución basada en LabVIEW es posible la adquisición de datos de varios dispositivos hardware, definir una aplicación para analizar o tomar decisiones según los datos adquiridos y entonces representar los datos de varias formas como interfaces gráficas, páginas web, bases de datos, archivos, etc.

LabVIEW proporciona la flexibilidad de un lenguaje de programación potente pero sin la complejidad de los entornos de programación tradicionales ya que usa un entorno de programación gráfico e intuitivo.

Tiene una buena integración con los instrumentos y dispositivos de medida, que permite configurar rápidamente cualquier dispositivo de medida y usarlos virtualmente, desde tarjetas de adquisición de datos a controladores de movimiento, sistemas adquisición de imágenes o PLCs.

Además se puede comunicar con otras aplicaciones y compartir datos a través de ActiveX, la Web, DLLs, SQL, librerías compartidas, TCP/IP, XML, OPC, etc. Esto permite crear aplicaciones flexibles y abiertas que pueden comunicarse con otras aplicaciones.

4.4. FUNCIONES IMPLEMENTEDAS

A continuación vamos a explicar el funcionamiento de las funciones implementadas para la realización de este proyecto.

Todo el proceso que se realiza mediante el software se puede dividir en cuatro grupos; adquisición, tratamiento, monitorización y almacenamiento de datos. Por ello para la explicación del software involucrado haremos esa distinción, lo que nos ayudará a comprender mejor el funcionamiento.

4.4.1. ADQUISICIÓN DE DATOS

La adquisición de datos analógicos en LabVIEW se basa en una serie de funciones disponibles en la librería *anlogin.llb,* que permiten la captura de información a través de los canales de entrada analógica de la tarjeta DAQ con diferentes opciones.

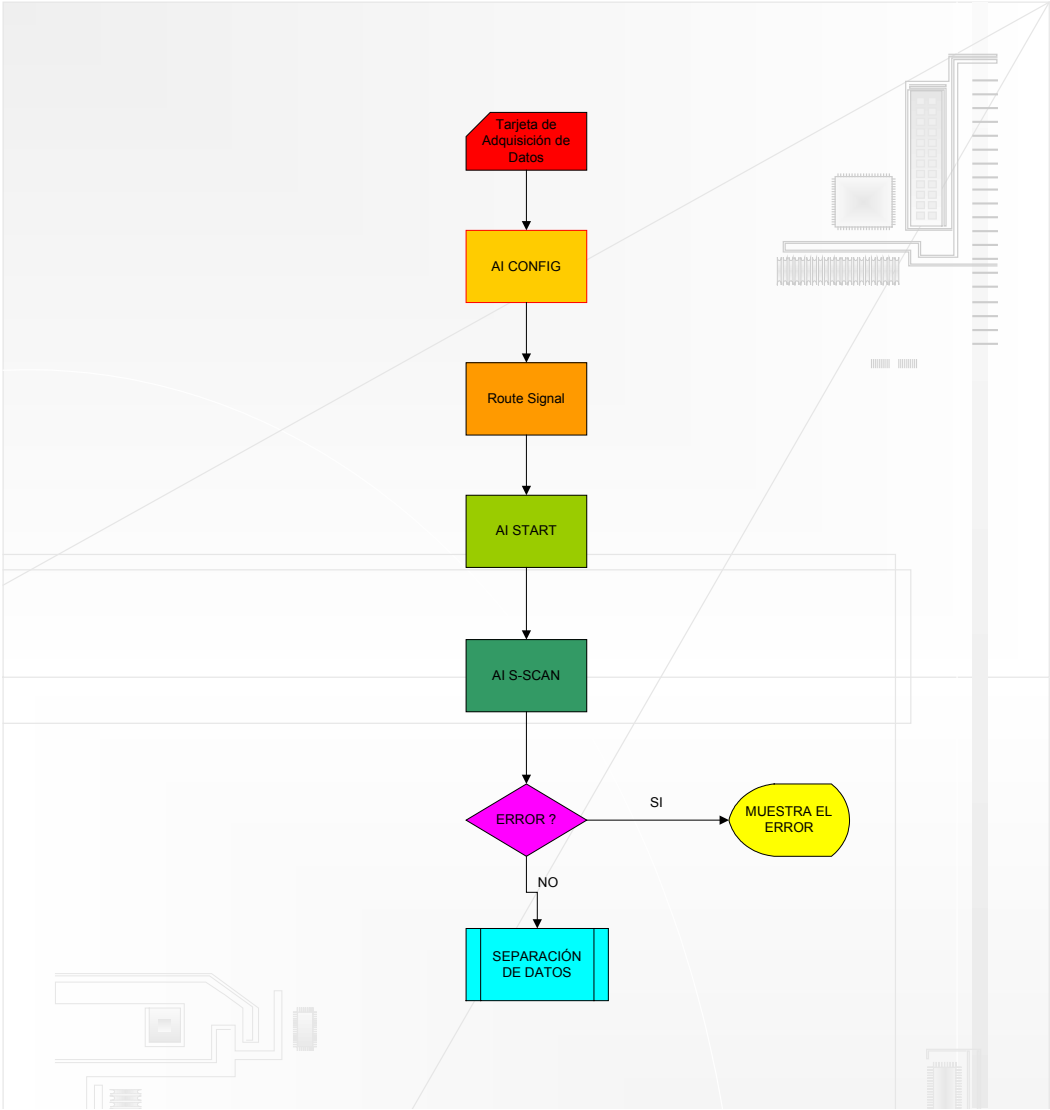

Figura 4.5. *Diagrama de flujo para la adquisición de datos*

En concreto, para la Captura de Datos utilizamos una serie de funciones:

AI CONFIG: esta función permite seleccionar los canales desde donde van a proceder los datos de las entradas analógicas. Este VI permite configurar el hardware y el buffer de datos.

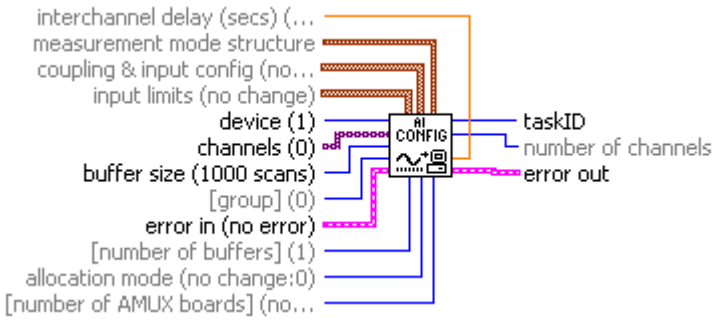

Figura 4.6. *Función AI CONFIG*

A continuación indicamos los parámetros y los valores que utilizamos en nuestro caso:

- Device: este parámetro indica el dispositivo de National Instruments desde el cual se adquiere los datos, que en nuestro caso es PCI-DAQ 6024-E. Para referirnos a este dispositivo le hemos asignado el número 1.
- Channels: este parámetro crea una variable en el panel frontal desde la cual podemos seleccionar los canales desde los que recibir los datos.
- Se pueden ir almacenando en la memoria un número de datos durante la adquisición de estos, es con este parámetro con el que se establece este número de datos almacenados.
- Task, este parámetro hace identifica la tarea que se está realizando, esta etiqueta, será usada por el resto de funciones de adquisición de datos y funciones de error.
- Error, identifica el tipo de error que pueda ocurrir durante el proceso realizado por las funciones de adquisición de datos, este error será mostrado al llegar a la función Error.

ROUTE SIGNAL; se usa este VI para encaminar una señal especificada como entrada o salida.

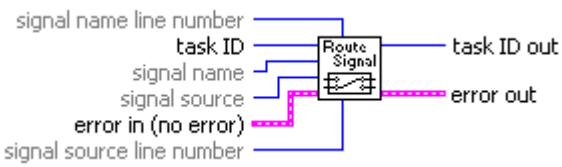

Figura 4.7. *Función Route Signal*

AI START; se usa este VI para empezar a almacenar datos. Con esta función podemos establecer el periodo de muestreo, el número de muestras a adquirir, las condiciones del trigger.

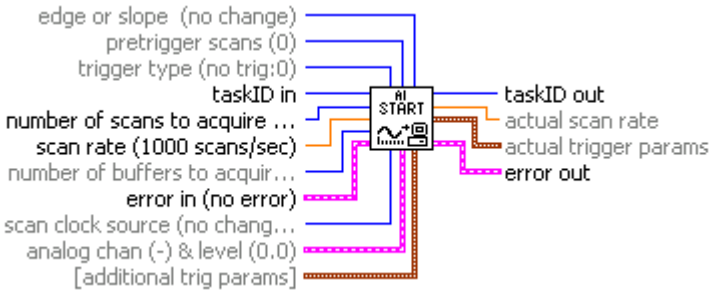

Figura 4.8. *Función AI START*

Los parámetros que utilizamos son:

- TaskID in se utiliza para tener identificada la tarea en la que se está cada momento. Esta unido a taskID out de la función Route Signal.
- Error out indica si ha ocurrido algún error durante el desarrollo de esta función, va unido a error in de la función AI S-SCAN.
- Scan rate, es el número de muestras a adquirir. Equivale al tiempo de muestreo por canal. En nuestro caso tiene un valor de 10 muestras/seg.
- Number of scans to acquire, es el número total de muestras que LabVIEW adquiere antes de completar al adquisición. Una muestra es un punto por canal. En nuestro caso tiene un valor de 0, que significa la adquisición continua.

AI-S SCAN, devuelve una muestra adquirida desde los canales de la tarjeta de adquisición de datos, si los datos no son almacenados o una muestra de los datos que habían sido almacenadas.

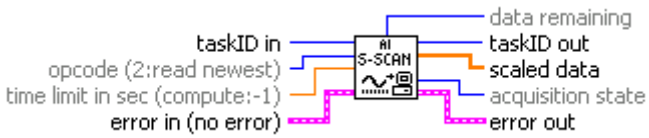

Figura 4.9. *Función AI S-SCAN*

En esta función los parámetros utilizados son los siguientes:

- TaskID in se utiliza para tener identificada la tarea en la que se está cada momento. Esta unido a taskID out de la función AI-START. TaskID out está unido a TaskID in de AI CLEAR.
- Error out indica si ha ocurrido algún error durante el desarrollo de esta función, va unido a error in de la función AI CLEAR. También error in está unido a error out de AI-START.
- Opcode, especifica el tipo de datos que son recuperados por el VI, en nuestro caso es *read oldest data.*
- Scaled data, contiene los datos adquiridos, se crea una especie de matriz en donde cada fila contiene los datos de un canal distinto, de esta forma, se pueden separar estos datos y trabajar con los datos de cada canal con los criterios que se establezcan para cada uno.

AI CLEAR, limpia todas la tareas asociadas a la tarea identificada con TaskID in.

Figura 4.10. *Función AI CLEAR*

Los parámetros utilizados son:

- TaskID in se utiliza para tener identificada la tarea en la que se está cada momento. Esta unido a taskID out de la función ERROR.
- Error out indica si ha ocurrido algún error durante el desarrollo de esta función, va unido a error in de la función AI CLEAR. También error in está unido a error out de AI S-SCAN.

ERROR, esta función indica si ha ocurrido un error en algún momento. Cuando ocurre algún error da un pequeña descripción del mismo. También existe la opción de un diálogo indicando el error que ha aparecido y preguntando cual es el siguiente paso.

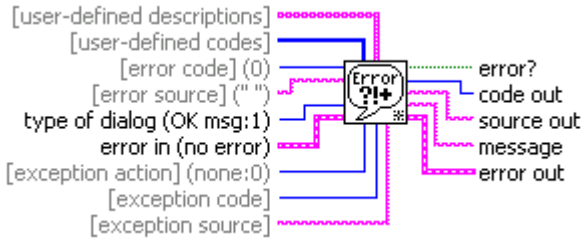

Figura 4.11. *Función ERROR*

El único parámetro que utilizamos es error in, en este parámetro se indica si ha ocurrido algún error durante el desarrollo del proceso y qué tipo de error.

4.4.2. TRATAMIENTO DE DATOS

El tratamiento de los datos dependerá del tipo de datos de que se trate, para la humedad y temperatura sólo se realizará una monitorización y almacenamiento de ellos. Por el contrario con los datos referentes a la concentración de oxígeno existente en el interior de la cámara se realizará, a demás de la monitorización y almacenamiento de ellos se utilizarán para la realización de un control PID con el que podremos controlar en cada momento la mezcla de gases.

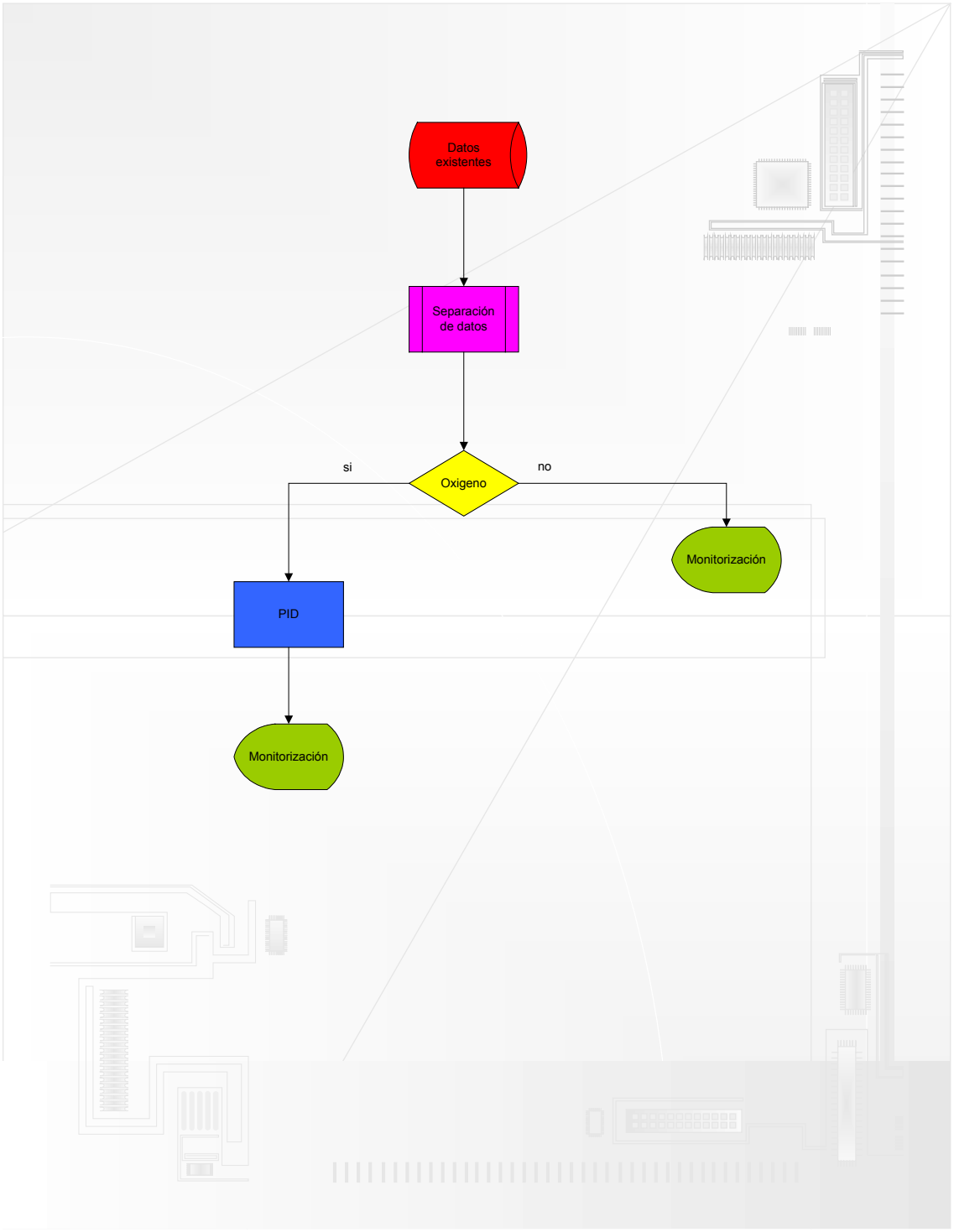

Figura 4.12. *Diagrama de flujo para el tratamiento de datos*

Los datos proceden de la función AI S-SCAN como comentamos anteriormente para poder separar los datos referentes a humedad, temperatura y oxígeno, es necesario separarlos, ya que vienen dados como en una matriz,

para ello utilizamos un elemento en el que indicamos el índice de la matriz que queremos extraer teniendo de esta manera los distintos datos.

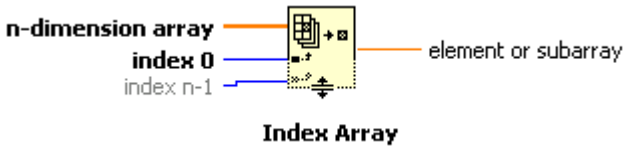

Figura 4.13. *Función Index Array*

Los datos vienen datos en voltaje, para poder monitorizarlos con las unidades que deseamos es necesario convertirlos según las siguientes ecuaciones:

Para humedad:

$$\% = ((1000 * X) – 1079) / 25{,}68$$

Para temperatura:

$$ºC = 10 * X$$

Para el oxígeno:

$$\% = (20{,}9 * x) / 4.93$$

Interfaz para la monitorización:

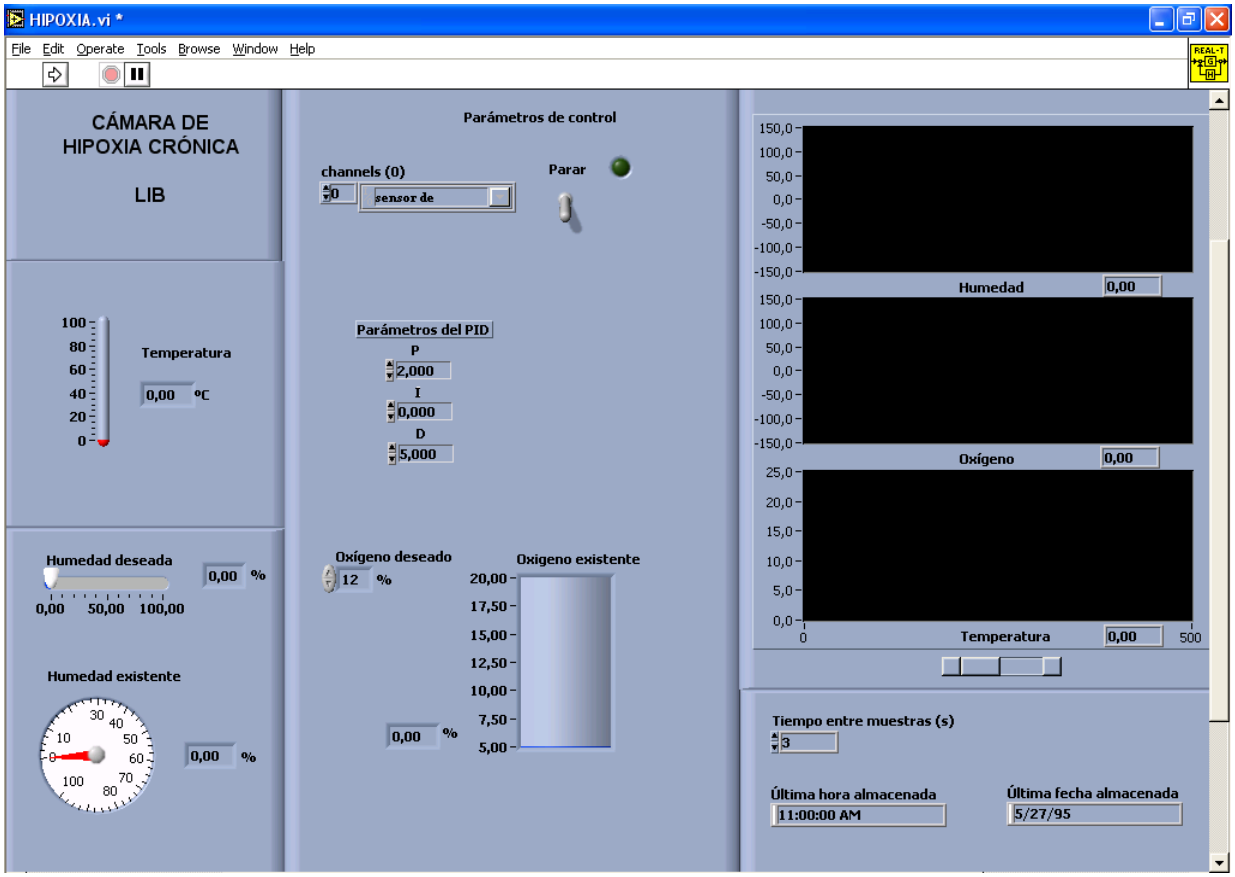

Figura 4.14. *Interfaz para la monitorización de las señales*

En el caso de los datos procedentes del sensor de oxígeno no sólo se usan para su monitorización y almacenado, sino que son utilizados para realizar el control PID con el que regular la mezcla de gases en el interior de la cámara.

Antes de explicar como hemos realizado este control PID mediante software, vamos a dar una pequeña explicación de en qué consiste un control PID.

En el diagrama de bloques el control PID aparece como un SubVI, tal como muestra la Figura 4.15.

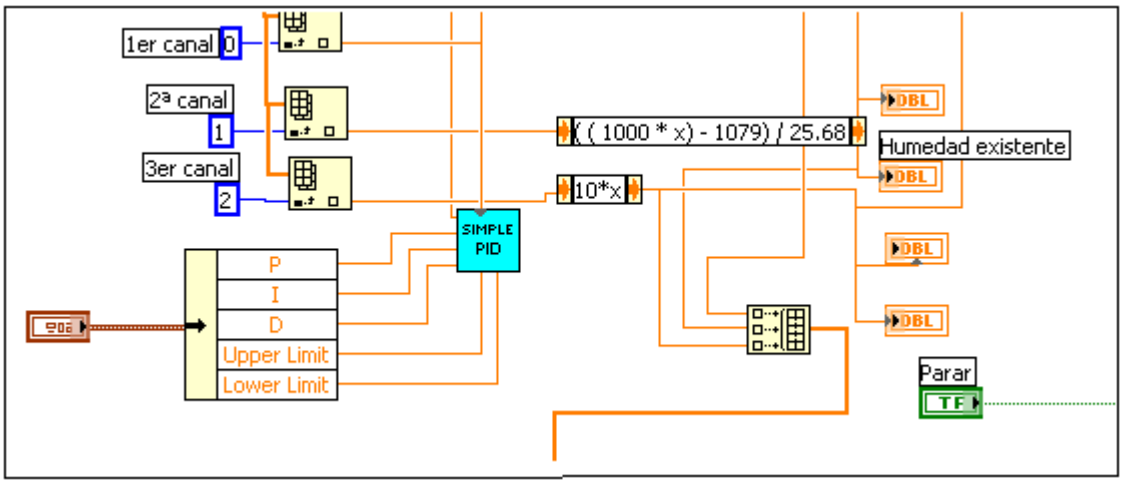

Figura 4.15. *Simple PID*

El código de este control aparece en la siguiente figura.

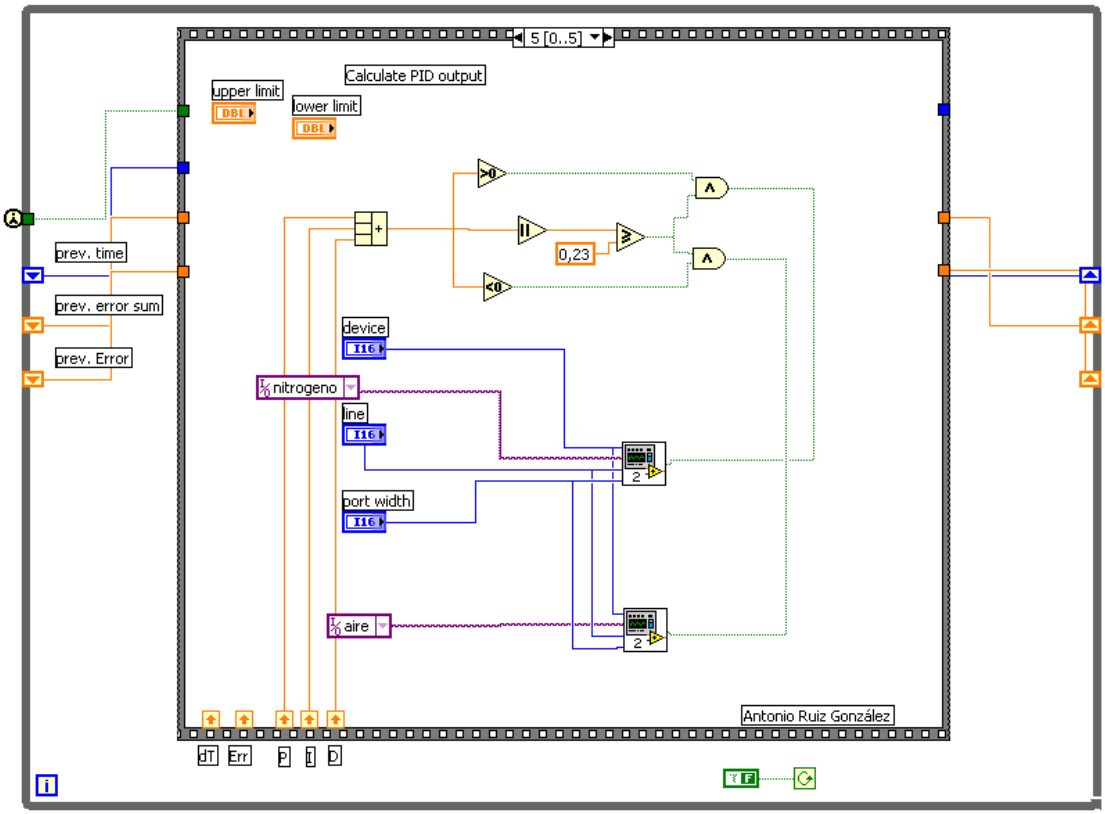

Figura 4.16. *Diagrama de bloques del control PID*

Esto es un simple PID el cual se usa para controlar el sistema. La idea de un PID es controlar el sistema por comparación entre el punto que hemos establecido y el valor real de la variable a controlar, dando una salida a los

distintos dispositivos que llevan a cabo una acción externa con el objetivo de variar esa variable real a la que hemos hecho referencia.

Esa variable a controlar es la concentración de oxígeno, la cual ya ha sido adquirida gracias a las funciones anteriormente estudiadas.

P es la componente proporcional. Gracias a esta componente dará una salida proporcional al error producido entre el valor existente y la referencia establecida. Su código es el de la Figura 4.17.

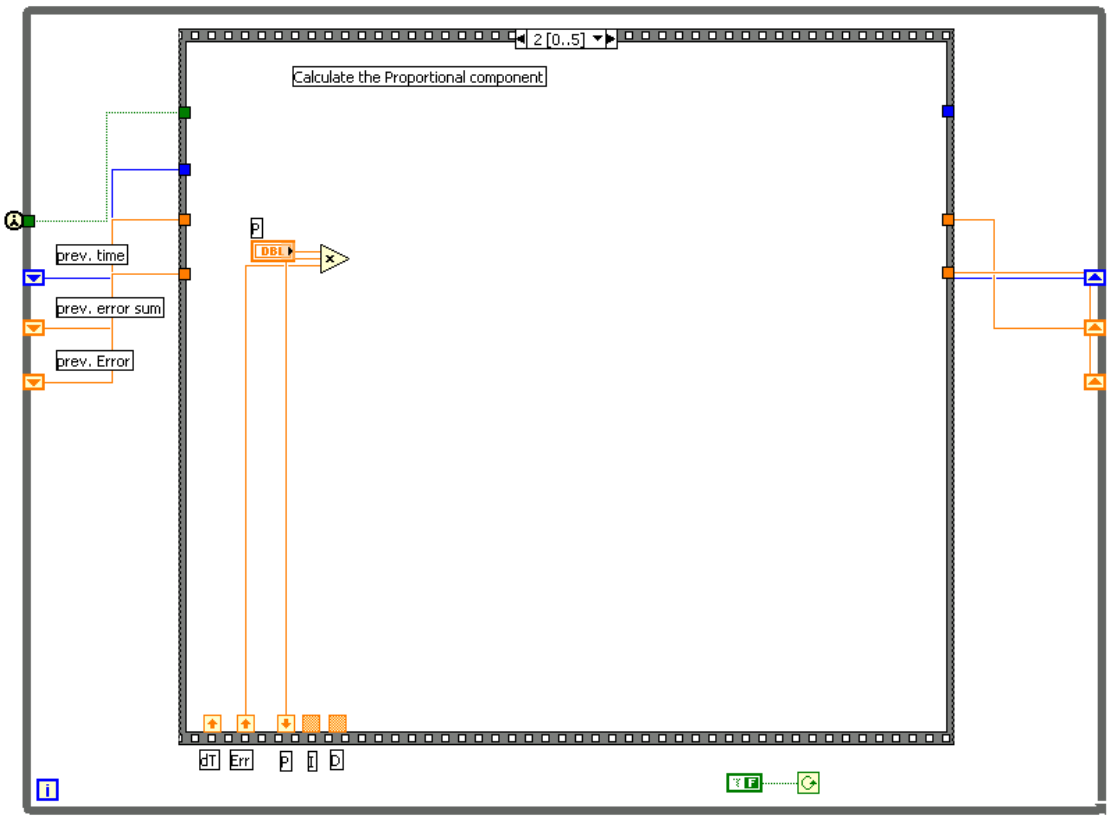

Figura 4.17. *Código para la componente proporcional*

I es la componente integral. Gracias a esta componente el control PID tendrá en cuenta que es lo que ha ocurrido en el pasado. Su código es el de la Figura 4.18.

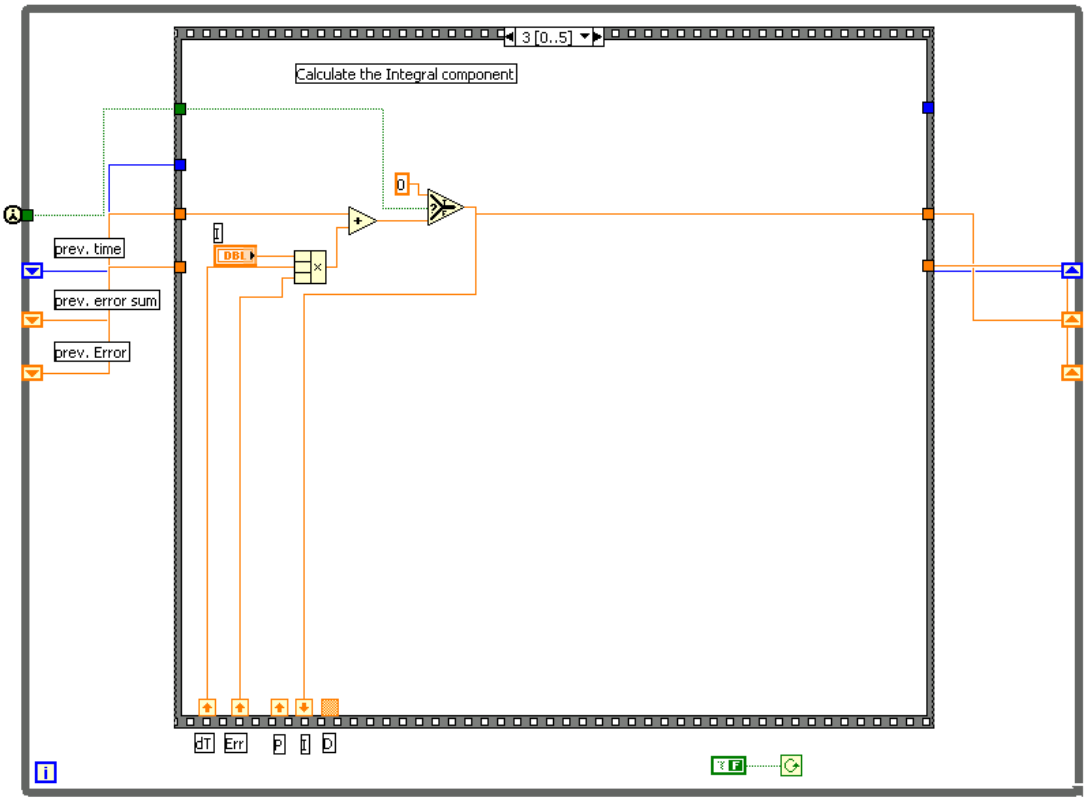

Figura 4.18. *Código para la componente integral*

D es la componente derivativa. Gracias a esta componente el control PID tendrá en cuenta que es lo que ha ocurrirá en el futuro. Su código es el de la Figura 4.19.

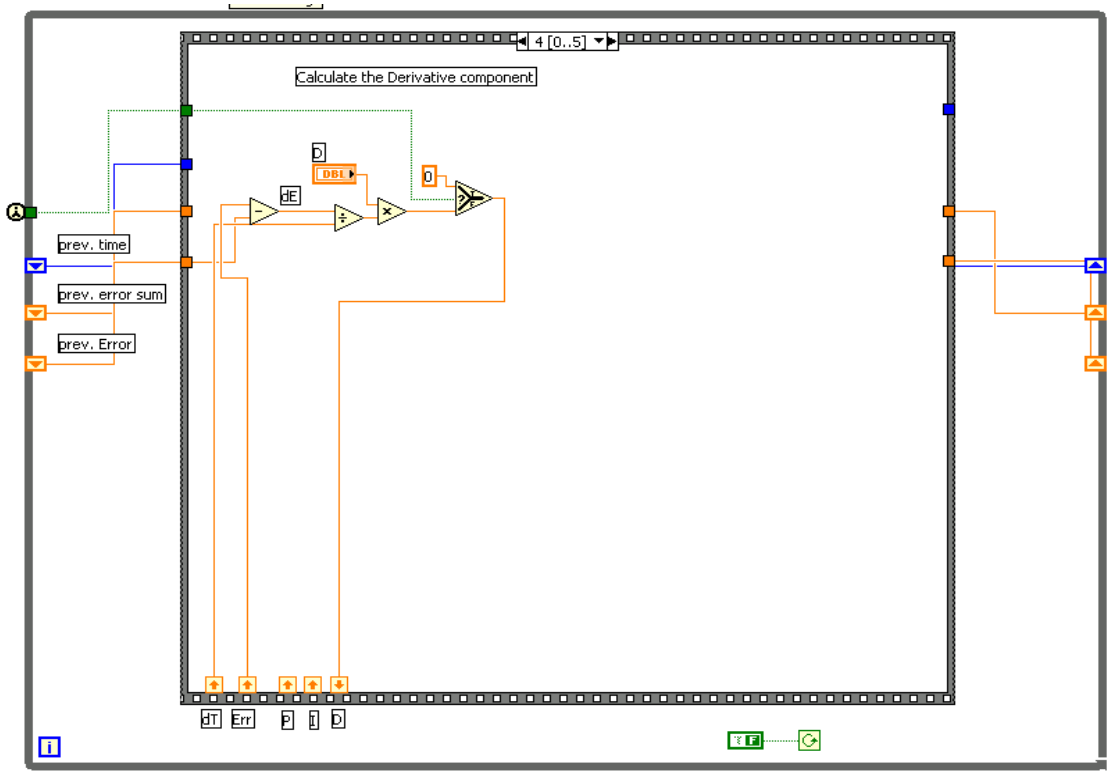

Figura 4.19. *Código para la componente derivativa*

4.4.3. Almacenamiento de datos

Los datos deseados como el oxígeno existente dentro de la cámara, la humedad y la temperatura, tras ser recogidos por los sensores son almacenados en un archivo de Excel, el cual ha sido creado al inicio de la ejecución del programa o bien remplazando los datos existentes en un archivo ya creado. Esto se consigue mediante la función *Write Charácter To File.vi.*

En este archivo a demás de los datos procedentes de los sensores también es almacenado la fecha y hora en que ha sido capturados. Esto se consigue mediante la función *CREATE DATA STRING.*

También se podrá elegir el tiempo entre las muestras a almacenar. Esto se consigue mediante la función *MULTI DELAY.*

A continuación se muestra el código que permite todo esto.

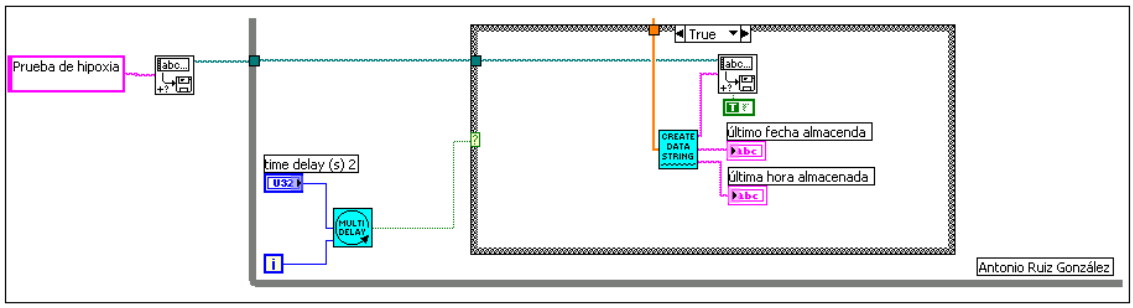

Figura 4.20. *Código para el almacenamiento de datos*