

Trabajo Fin de Grado
Ingeniería de las Tecnologías de
Telecomunicación

Control y automatización de procesos microfluídicos
de síntesis de radiofármacos usando LabVIEW

Autor: Guillermo Gómez Chávez

Tutor: Antonio Luque Estepa

Dep. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017



Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

**Control y automatización de procesos
microfluídicos de síntesis de radiofármacos usando
LabVIEW**

Autor:

Guillermo Gómez Chávez

Tutor:

Antonio Luque Estepa

Profesor titular

Dep. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Trabajo Fin de Grado: Control y automatización de procesos microfluídicos de síntesis de radiofármacos
usando LabVIEW

Autor: Guillermo Gómez Chávez

Tutor: Antonio Luque Estepa

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Este proyecto me ha permitido incorporar no solo los conocimientos adquiridos en el ámbito educativo, sino también usar las vivencias y aportes externos que han moldeado mi paso por esta Universidad para llegar al resultado final. Todo ello me ha permitido darle mayor sentido al esfuerzo llevado a cabo y sentir así que devuelvo algo productivo de esta etapa.

Agradezco particularmente haber podido disfrutar de la colección más diversa posible de amigos ya que creo que han enriquecido positivamente mi experiencia. Por supuesto también a mi familia más cercana y a los que lo son por derecho propio.

Por último, me gustaría hacer una mención especial al Departamento de Ingeniería Electrónica de la ETSI pues demuestran día a día que fomentar la colaboración basada en la amistad y la complicidad es tan importante como la teoría que debe adquirirse. En este sentido debo agradecer también la colaboración continua de Blas Salvador Domínguez pues ha permitido conocer de primera mano las necesidades particulares de las pruebas a realizar, dando sentido desde el primer día al proyecto y a su finalidad.

La medicina se ha convertido en uno de los grandes impulsores de la investigación y el desarrollo tecnológico. Dentro del ámbito sanitario, el tratamiento y detección del cáncer supone uno de los mayores retos a los que se enfrenta la sociedad, constituyendo una línea de investigación activa y multidisciplinar en la cual la ingeniería supone un pilar de avance continuo.

Englobado dentro del proyecto MICRORAD, fruto de acuerdos entre empresas y universidades, este TFG ha desarrollado una interfaz de usuario fácil de usar que permite controlar, tanto de manera directa como por medio de secuencias automáticas, un prototipo para la síntesis de radiofármacos para la generación de marcadores PET. Así mismo, se ha desarrollado una interfaz más, destinada a la gestión de ficheros de texto en los que almacenar, consultar y modificar secuencias de hasta N estados con las cuales poder automatizar los experimentos.

En ambos casos, las interfaces han sido desarrolladas en la plataforma LabVIEW desde la cual se interactúa tanto la placa Arduino del prototipo como los controladores de caudal adquiridos a la empresa Fluigent.

Abstract

Medicine has become a major driver of researches and technological development. Within the medical field, the treatment and detection of cancer is one of the most important challenges facing society, constituting a line of active and multidisciplinary research in which engineering is a pillar of continuous advancement.

Encompassed within the MICRORAD project, which is result of agreements between companies and universities, this FYP has developed a user-friendly user interface for the control, both directly and through automatic sequences, of a prototype for the synthesis of radiopharmaceuticals for the generation of PET markers. In addition to it, it has also been developed another interface meant to manage text files in which sequences of up to N states are stored, consulted and modified to use them to automatize experiments.

For both cases, interfaces have been developed using LabVIEW platform from which interaction is achieved between the Arduino-based prototype and the Flow controllers adquired to the company Fluigent.

Agradecimientos	ix
Resumen	xi
Abstract	xiii
Índice	xiv
Índice de Tablas	xvi
Índice de Figuras	xviii
Notación	xxi
1 Introducción	1
1.1. Marco de trabajo	1
1.2 Motivación del proyecto	1
1.3 Objetivos	2
1.4 Estructura del documento	3
2 Estado del arte	4
2.1. Sistema Hardware	4
2.1.1. MFCS -EZ	6
2.1.1. FLOW RATE PLATFORM	7
2.3 Sistema Software	8
2.2.1 Introducción a LabVIEW™	9
3 MICRORAD HARDWARE	10
3.1. Estado del sistema	10
3.2. Componentes del sistema	12
3.3. Descripción del sistema	14
4 MICRORAD SOFTWARE	16
4.1. Estado del sistema	16
4.2. Integración de Fluigent en LabVIEW: FRCM.	17
4.3. Definición del modo automático	20
4.4. Planteamiento global del sistema	24
4.4. Estructura de ficheros y decisiones de diseño	34
4.4.1 Front Panel.vi	35
4.4.2 ModoAutomatico.vi	35
4.4.3 Funciones	36
4.5. Pruebas parciales	38
5 Pruebas y resultados	40
5.1 Secuencia de un estado con CCDuración sin Fluigent	40
5.2 Secuencia de dos estados con CCDuración y CCTemperatura sin Fluigent	40
5.3 Secuencia de tres estados con Fluigent, CCVolumen, CCDuración y CCTemperatura	41

<i>5.4 Resultados de usar Prueba5_1.txt</i>	41
<i>5.5 Resultados de usar Prueba5_2.txt</i>	43
<i>5.6 Resultados de usar Prueba5_3.txt</i>	44
6 Conclusiones y desarrollo futuro	46
Anexo A: Manual de instalación	48
Anexo B: Manual de usuario: Preparación del entorno.	50
Anexo C: Manual de usuario: Realización de una prueba.	53
Anexo D: Código fuente	
Referencias	75

ÍNDICE DE TABLAS

Tabla 2-1. Comparativa de placas Arduino compatibles con LabVIEW	5
Tabla 2-2. Especificaciones del MFCS-EZ	6
Tabla 3-1. Lista de componentes necesarios	12
Tabla 4-1. Resumen de controles e indicadores del FRONT PANEL	25
Tabla 4-2. Resumen de controles e indicadores del ModoAutomatico.vi	30
Tabla B-1. Resmuen de conexiones por bolsa de componentes	52
Tabla C-1. Protocolo de inicialización del programa	53

ÍNDICE DE FIGURAS

Figura 2-1. Prototipo Hardware para MicroRad	4
Figura 2-2. Comparativa del comportamiento de MFCS con bombas peristálticas	5
Figura 2-3. Experimento con MFCS y el software MAESFLO	6
Figura 2-4. Representación de los componentes del Flow Rate Platform	7
Figura 2-5. Experimento usando tanto MFCS-EZ como el FRP	7
Figura 3-1. Pinout de la placa Arduino Mega 2560 utilizado	11
Figura 3-2. Esquema de conexiones del experimento completo	14
Figura 4-1. Front Panel de la versión software de partida	16
Figura 4-2. Modificación del VI Check For Pin Out Of Range.vi del sistema para el correcto funcionamiento del programa.	17
Figura 4-3. Selección de la FRP y coeficientes de realimentación	18
Figura 4-4. Ejemplo de calibración del FRCM a través de su wizard, expresado en % del caudal máximo posible	18
Figura 4-5. Block Diagram de Fluigent FRCM SDK Example.vi	19
Figura 4-6. Respuesta del FRCM integrado en LabVIEW™	20
Figura 4-7. Respuesta estable del FRCM en el tiempo	20
Figura 4-8. Generación de un fichero mediante la función Prompt User	22
Figura 4-9. Interfaz obtenida mediante la función Prompt User	22
Figura 4-10. Estructura de un fichero .txt con una secuencia	23
Figura 4-11. Simplificación de una estructura básica para leer un estado de una secuencia	23
Figura 4-12. Global.vi	24
Figura 4-13. Interfaz de usuario del FRONT PANEL principal del programa.	25
Figura 4-14. Front Panel de ModoAutomatico.vi para la gestión de secuencias automáticas	31
Figura 4-15. Estructura de ficheros	34
Figura 4-16. Estructura del Block Diagram de FRONT PANEL	35

Figura 4-17. Estructura del Block Diagram de ModoAutomatico.vi	36
Figura 4-18. Ejemplo de la generación de una función en LabVIEW™	36
Figura 4-19. Uso de funciones en LabVIEW™	38
Figura 4-20. Colección de pruebas parciales sobre ModoAutomatico.vi	39
Figura 4-21. Barra de herramientas para la depuración de programas en LabVIEW™	39
Figura 5-1. Estado 1 de la secuencia correspondiente a Prueba5_1.txt	40
Figura 5-2. Estados 1 y 2 de la secuencia correspondiente a Prueba5_2.txt	41
Figura 5-3. Secuencia completa correspondiente a Prueba5_3.txt	42
Figura 5-4. Resultados de Prueba5_1.txt	43
Figura 5-5. Resultados del estado 1 de la secuencia Prueba5_2.txt	43
Figura 5-6. Resultados del estado 2 de la secuencia Prueba5_2.txt	44
Figura 5-7. Resultados del estado 1 de la secuencia Prueba5_3.txt	44
Figura 5-8. Resultado del estado 2 de la secuencia Prueba5_3.txt	45
Figura 5-9. Resultado del estado 3 de la secuencia Prueba5_3.txt	45
Figura A-1. Paquetes que se deben instalar en LabVIEW para ejecutar el programa	47
Figura A-2. Verificación de la conexión entre VIPM y LabVIEW	47
Figura A-3 Error al utilizar los paquetes de Fluigent si no hay enlace entre VIPM y LabVIEW	48
Figura A-4. Ajustes para usar los pines analógicos de la Arduino Mega 2560	48
Figura B-1. Esquema de conexiones del sistema	50
Figura B-2. Fotografía de las conexiones reales entre los componentes	52

Notación

<i>Referencia</i>	Nombre de fichero, tabla o figura
SW	Aplicación software
HW	Hardware

1 INTRODUCCIÓN

1.1 Marco de trabajo

En el año 2012 el Departamento de Ingeniería Electrónica de la Universidad de Sevilla presentó a la Junta de Andalucía el proyecto denominado “*Microlab-en-chip para producción de radiofármacos para diagnóstico PET MICRORAD*” durante la convocatoria del mencionado año para los Proyectos de Investigación de Excelencia, siendo esta concedida poco después. Dicho proyecto está siendo desarrollado conjuntamente por el Centro Nacional de Aceleradores (CNA) y el Departamento de Ingeniería Electrónica, Grupo de Microsistemas (GM-US) ambos de la Universidad de Sevilla (US), y el Grupo de Ingeniería Electrónica (GEE-UIB) del Departamento de Física de la Universidad de las Islas Baleares (UIB). [1]

El objetivo final de MICRORAD es el desarrollo de tecnologías para la creación de dispositivos reactor-on-chip miniaturizados dedicados a la generación altamente eficiente de radiofármacos a partir de radioisótopos para aplicaciones de tomografía por emisión de positrones (PET), una de las técnicas más eficientes para el diagnóstico de patologías tumorales, para el control de calidad de los radiofármacos producidos, así como el desarrollo de protocolos de producción de radiofármacos en dichos sistemas. [1]

1.2 Motivación del proyecto

MICRORAD se constituye para contribuir al desarrollo de varios sectores clave en Andalucía, ayudando al paso a una economía basada en el conocimiento, así como para impulsar la transferencia de tecnología desde universidades y centros de investigación a industrias que puedan explotar comercialmente los resultados, reduciendo sus costes de producción y fomentando el desarrollo económico y la creación de empleo.

La tomografía por emisión de positrones (conocida como PET, según el acrónimo en inglés) es una de las técnicas diagnósticas no invasivas más novedosa introducida dentro del ámbito de la medicina nuclear la cual puede detectar la aparición temprana de una enfermedad antes de que sea evidente con otros exámenes por imagen. Esta técnica emplea trazadores marcados con emisores de positrones, y con ella se pueden obtener medidas cuantitativas in vivo de diversos procesos fisiológicos y bioquímicos, como por ejemplo flujo sanguíneo, metabolismo glucídico y de ácidos grasos, distribución de receptores o farmacocinética de los radiofármacos en los tejidos.

Pese a que España ya cuenta con más de 75 instalaciones PET, su utilización está siendo frenada por el alto precio de la prueba y la congestión del sistema sanitario [3]. En este sentido las mejoras en la producción utilizando tecnología de microfluidos respecto a los sistemas macroscópicos ya existentes consistirán principalmente en la reducción del tiempo de producción, el aumento de la eficiencia, la reducción del coste de los dispositivos de producción y el control de calidad integrado.

Actualmente MICRORAD consta de dos líneas de trabajo paralelas, una de investigación y otra de desarrollo de la interfaz de control de las que ambas están activas y siguen arrojando nuevos resultados. El presente Trabajo Fin de Grado (TFG) surge como continuación del esfuerzo que llevaron a cabo Iván Carrillo y Blas Salvador Domínguez sobre esta última línea de trabajo en las instalaciones de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla entre los meses de noviembre de 2014 y junio de 2015. El fruto de la mencionada colaboración fue la consecución de un prototipo de pruebas basado en la plataforma Arduino Mega 2560 y el inicio del desarrollo de un programa Software (SW) que permitiese el control de dicho prototipo mediante el entorno de programación LabVIEW. Pese a que el prototipo Hardware (HW) se completó satisfactoriamente,

esta última fase no se llegó a finalizar y sólo existía una primera aproximación que permitía una comunicación básica entre el Arduino y los componentes del prototipo. En consecuencia, se hace notable la necesidad de completar e incorporar nuevas utilidades al prototipo HW, así como de diseñar una interfaz SW que permitiese actuar sobre él de manera controlada.

1.3 Objetivos

La tarea principal de este TFG se centra en generar una interfaz de fácil uso que permita definir programas secuenciales que se puedan guardar y modificar para su posterior uso. Esta interfaz facilitará la elaboración de secuencias automatizadas a partir de las características introducidas por el usuario.

Así pues, se deberá generar un fichero con el cual sea posible crear, visualizar y modificar los valores de cada elemento del sistema para cada uno de los hasta N estados de una secuencia. Dicho sistema se compone de:

- 1) 6 solenoides destinados a gestionar las válvulas de acceso al microreactor.
- 2) Un sensor de temperatura que se comunicará vía I2C con el Arduino. La temperatura debe representarse en C°.
- 3) Un calentador regulable mediante una salida PWM cuya actuación debe estar controlada mediante un PID de valores configurables.
- 4) Una de las vías de entrada controladas por los solenoides se podrá conectar al sistema de control de presión de Fluigent.

La nueva interfaz debe ser capaz de actuar sobre los módulos MFCS-EZ y FLOW RATE PLATFORM (FRP) de Fluigent adquiridos por el Departamento de Ingeniería Electrónica para el control del volumen introducido al micro-reactor y cuyo funcionamiento se detalla en la sección 2.1. Así pues, queda como tarea de este TFG elegir la mejor manera de incluir dicho HW y estudiar los modos de comunicación entre ellos y con el prototipo.

El carácter vivo de MICRORAD hace que, aunque la modificación del HW no vaya a ser objeto de este TFG, sí que sea necesario tener en cuenta una serie de mejoras que se prevén necesarias en un futuro próximo a raíz de los resultados que se están obteniendo en la línea de investigación.

- Posiblemente será necesario incluir hasta un total de 10 solenoides
- Es probable que en un futuro cercano se desee incorporar un relé destinado a controlar una bomba de vacío o ventilador.

Cada estado de la secuencia deberá diferenciar entre la asignación (Set Point o SP) de valores a los elementos del sistema y la condición de cambio (CC) del estado al siguiente.

- 1) Asignación: La actuación sobre cada solenoide
- 2) Asignación: La temperatura deseada en C°
- 3) Asignación: El estado de la bomba de vacío
- 4) Asignación: Caudal que se desea introducir, expresado en microlitros por minuto.
- 5) Condición de cambio: El volumen que se desea introducir mediante la entrada conectada a Fluigent, expresado en microlitros.
- 6) Condición de cambio: Temperatura de transición que sirva para indicar que la temperatura deseada ha sido alcanzada.
- 7) Condición de cambio: Duración del estado expresado en segundos.

1.4 Estructura del documento

Este documento consta de varios bloques temáticos estructurados de manera muy similar. Dadas las características del proyecto, resulta conveniente separar las explicaciones del trabajo realizado sobre los componentes Hardware (HW) del realizado sobre la interfaz Software (SW).

El primer bloque temático está formado por las secciones 1 y 2. A lo largo de dichos capítulos se presenta tanto el contexto en el que se inicia el TFG como las necesidades que se pretenden satisfacer y se da una introducción teórica de los instrumentos y herramientas que se han empleado.

El siguiente bloque temático explica las modificaciones realizadas sobre el HW y el SW mediante las secciones 3 y 4 respectivamente. Aquí se encuentra la información necesaria para entender qué es lo que hace cada fichero del proyecto LabVIEW recogido en el Anexo D junto con las pruebas parciales llevadas a cabo.

Finalmente, el último bloque temático presenta las pruebas finales realizadas sobre el sistema completo para demostrar que el resultado es el esperado y probar la versatilidad del tipo de secuencias que se pueden obtener. De esta manera, la sección 5, junto con el Anexo A, *Manual de usuario: Preparación del entorno*, y el Anexo B, *Manual de usuario: Realización de una prueba*, permiten al lector adquirir los conocimientos necesarios para poder trabajar de manera autónoma con los componentes de este TFG.

2 ESTADO DEL ARTE

La microfluidica es hoy en día una de las áreas de investigación con mayor potencial y cuyas aplicaciones siguen aumentando a mayor ritmo. Se trata de un campo multidisciplinar que comprende partes de la Física, la Química, la Ingeniería y la Biotecnología. La aplicación en la que se centra este TFG combina todas las áreas mencionadas para conseguir un dispositivo que acerque las salas PET a los hospitales y a los pacientes, así como para acelerar la investigación. Para ello, el dispositivo se apoya en el uso de los siguientes componentes:

- Un Microcontrolador.
- Una placa de trabajo que concentre un microreactor y los sensores y actuadores descritos en la sección 1.3.
- Un sistema de inyección para los fluidos.
- Un entorno software para el control.

2.1 Sistema Hardware

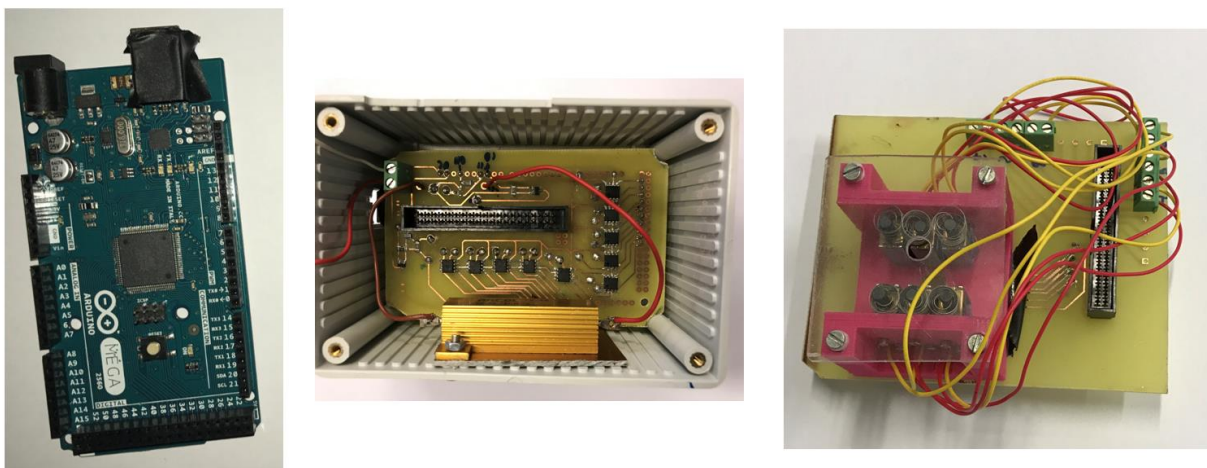


Figura 2-1. Prototipo Hardware para MicroRad

Al partir este TFG de un trabajo previo, una de las primeras tareas debe ser siempre la validación del diseño y la búsqueda de alternativas en caso de que las hubiese. En este sentido cabe preguntarse si la elección del desarrollo del prototipo de la *Figura 2-1* sobre la plataforma Arduino™ es la más idónea.

Dadas las características de los ensayos que van a realizarse sobre este prototipo, no existen indicios de necesidades fuera del alcance del Arduino por lo que su menor coste, mayor facilidad de despliegue y existencia de proyectos públicos y librerías sobre otras plataformas como Raspberry™ hace de él una elección más que razonable. Dentro de la familia Arduino existe una amplia gama de placas en el mercado de las que caben destacar la Arduino UNO, la Arduino Duemilanove y la Arduino Mega 2560 debido a que son las únicas que pueden integrarse en el entorno gráfico LabVIEW™ en cualquiera de sus versiones. Finalmente, la *Tabla 2-1* demuestra que la Arduino UNO dispone de un número bastante limitado de Inputs/Outputs y que dado que no se ha observado falta de almacenamiento ni lentitud en el sistema, reemplazar la Arduino Mega 2560 actual por la Duemilanove no conllevaría ninguna ventaja adicional.

Name	Processor	Operating/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/6	1	2	32	Regular	1
Mega 2560	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	4	8	256	Regular	4
Due	ATSAM3X8E	3.3 V / 7-12 V	84 MHz	12/2	54/12	-	96	512	2 Micro	4

Tabla 2-1. Comparativa de placas Arduino compatibles con LabVIEW

Con respecto al sistema de inyección de fluidos parece evidente la necesidad de un componente externo que pueda recibir ordenes y permita así que todo el sistema funcione de manera autónoma y controlada. Para ello existe una gran variedad de opciones limitadas principalmente por el presupuesto disponible. Así pues, podría parecer buena idea adquirir únicamente las válvulas necesarias para ahorrar del presupuesto, pero ello obligaría a desarrollar todo el sistema de control desde cero y desviaría el esfuerzo de las partes realmente importantes del proyecto.

Fluigent es una empresa líder mundial en el suministro de soluciones para la manipulación de microfluidos; desde controladores para la generación de gotas hasta plataformas de organ-on-chip. En el año 2006 sacaron al mercado de microfluidos unas bombas de control de flujo impulsadas por presión las cuales presentan numerosas ventajas con respecto a las jeringas y bombas peristálticas convencionales. [4]

Las bombas peristálticas constan de una tubería flexible, entre 3 y 25 mm de diámetro, que al ser comprimida sucesivamente por unas ruedas que giran continuamente, obligan a circular al líquido en la dirección del giro. El efecto resultante es similar al del movimiento peristáltico del aparato digestivo animal, del cual recibe su nombre. No ofrecen posibilidades de fugas, al no existir partes rígidas fijas y móviles en contacto, aunque presentan problemas por vida limitada del material elástico de la conducción. Suelen suministrar caudales reducidos, por lo que se emplean frecuentemente a escala de laboratorio. [5]

La tecnología de Fluigent por el contrario permite controlar el flujo de fluidos mediante regulación de la presión aplicada. El principio se basa en la presurización neumática de depósitos (tanto depósitos externos como cartuchos desechables o pozos integrados) que contienen los líquidos a inyectar en los sistemas microfluídicos.

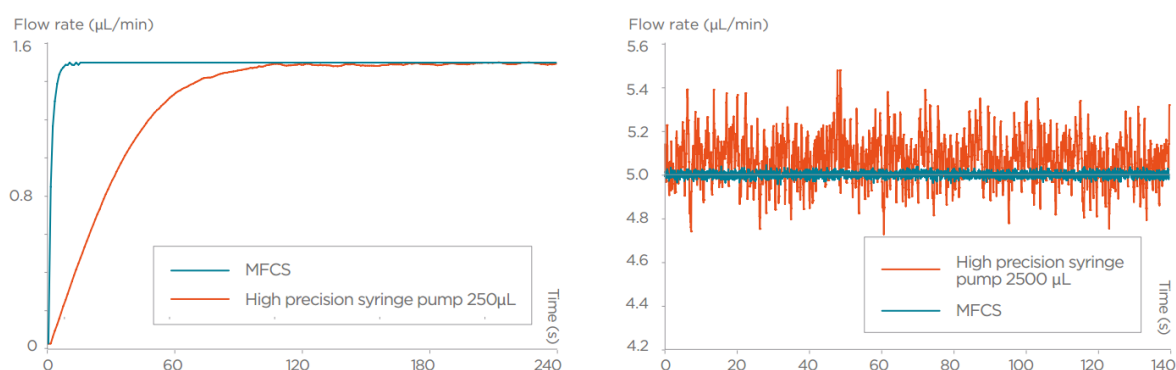


Figura 2-2. Comparativa del comportamiento de MFCS con bombas peristálticas

En este caso, la regulación de la fuente de presión proviene de la combinación del camino neumático original con un algoritmo de regulación muy rápido patentado por la empresa. Los principales beneficios de esta tecnología se enumeran a continuación.

1. Las presiones de salida son controladas gracias a un software dedicado, MAESFLO™.
2. Los actuadores de presión proporcionan de forma inmediata y automática las presiones solicitadas con una estabilidad muy alta gracias a un circuito de realimentación y al software FRCM, Flow Rate Control Module.
3. La conexión de las salidas de presión a los depósitos herméticos proporciona un control preciso y suave

del flujo de la muestra usada en el dispositivo.

4. Su uso es compatible con cualquier aplicación microfluídica.

Fluigent ha desarrollado la tecnología patentada FASTAB™ que incluye un algoritmo avanzado de control de realimentación sin elementos mecánicos involucrados. Estas características permiten un flujo sin pulso, así como una mayor capacidad de respuesta.

2.1.1 MFCS™-EZ y MAESFLO™

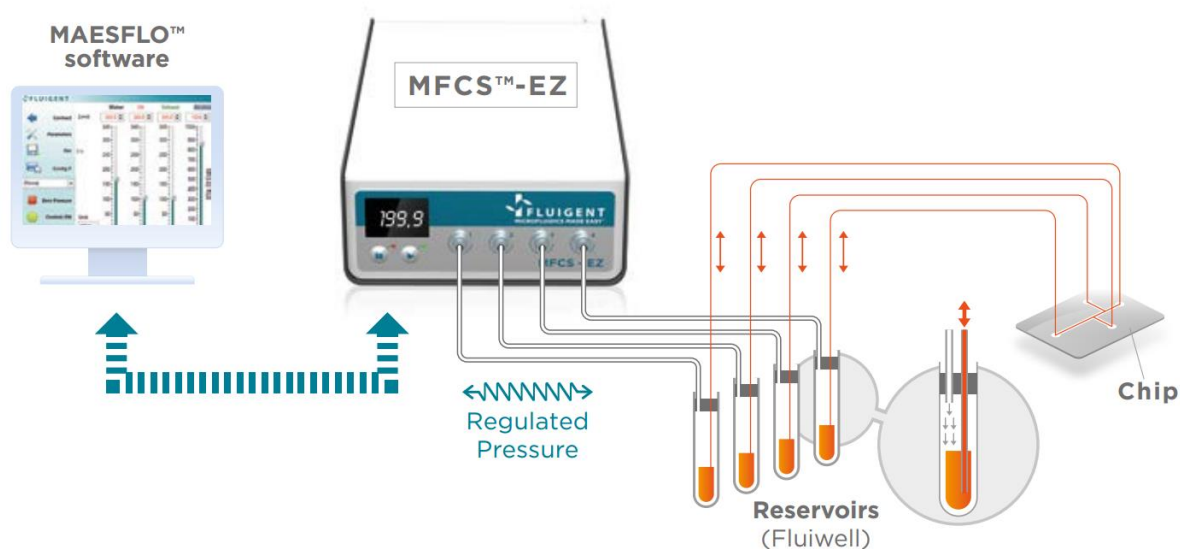


Figura 2-3. Experimento con MFCS y el software MAESFLO

El MFCS-EZ, Microfluidic Flow Control System, es un controlador de flujo por presión para todo tipo de aplicaciones micro y nano fluídicas desarrollado por la empresa Fluigent. En presencia de una fuente de presión externa, este dispositivo, con la ayuda del SW dedicado MAESFLO™, es capaz de proporcionar hasta cuatro salidas reguladas independientes con valores de presión configurables que se usarán para impulsar el fluido del depósito correspondiente a través del circuito tal y como muestra la *Figura 2-3*. [6]

	MFCS™-EZ
Pressure sensor resolution	0.03 % full scale
Pressure stability	< 0.1% CV (on measured values)
Response time	Down to 40 ms (depending on user PC operating system and configuration)
Settling time	Down to 100 ms (output volume dependent)
Pressurizing gas	Non corrosive or explosive gas (pressurized air recommended or N ₂ , AR, CO ₂)
Size	16 x 23 x 6.5 cm ³ (6.3 x 9 x inch ³)
Weight	2.0 kg (4.4 lbs)

Tabla 2-2. Especificaciones del MFCS-EZ

La figura superior se ha obtenido de los manuales de la empresa y muestra las especificaciones más relevantes del MFCS-EZ a tener en cuenta al decidir su conveniencia para el proyecto desarrollado. Resaltan los tiempos de establecimiento y de respuesta, pero cabe destacar que se mencione la dependencia del sistema operativo a

usar, de lo que habrá que comprobar cómo de relevante llega a ser.

2.1.2 FLOW RATE PLATFORM

Este módulo requiere de dos componentes para poder funcionar, el FLOW UNIT (mostrado en la parte izquierda de la *Figura 2-4*) y el FLOWBOARD (mostrado en la parte derecha de la *Figura 2-4*).

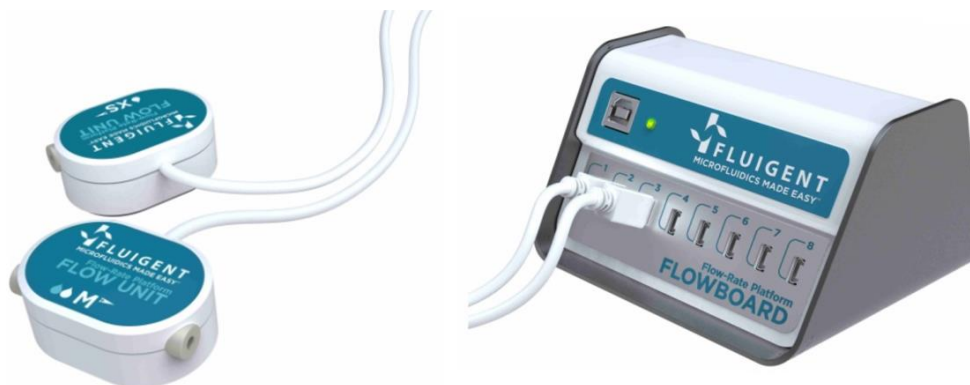


Figura 2-4. Representación de los componentes del Flow Rate Platform

El uso de ambos componentes proporciona una solución para medir y/o controlar caudales en cualquier aplicación de microfluídica, permitiendo comprobar en todo momento el caudal y el volumen de líquidos que fluyen a través de un experimento como el de la *Figura 2-5*. En este punto ya se puede observar cómo se realiza la regulación del caudal suministrado únicamente con componentes de Fluigent y el software de la empresa, por lo que solo quedaría comprobar cómo se obtiene la medida.

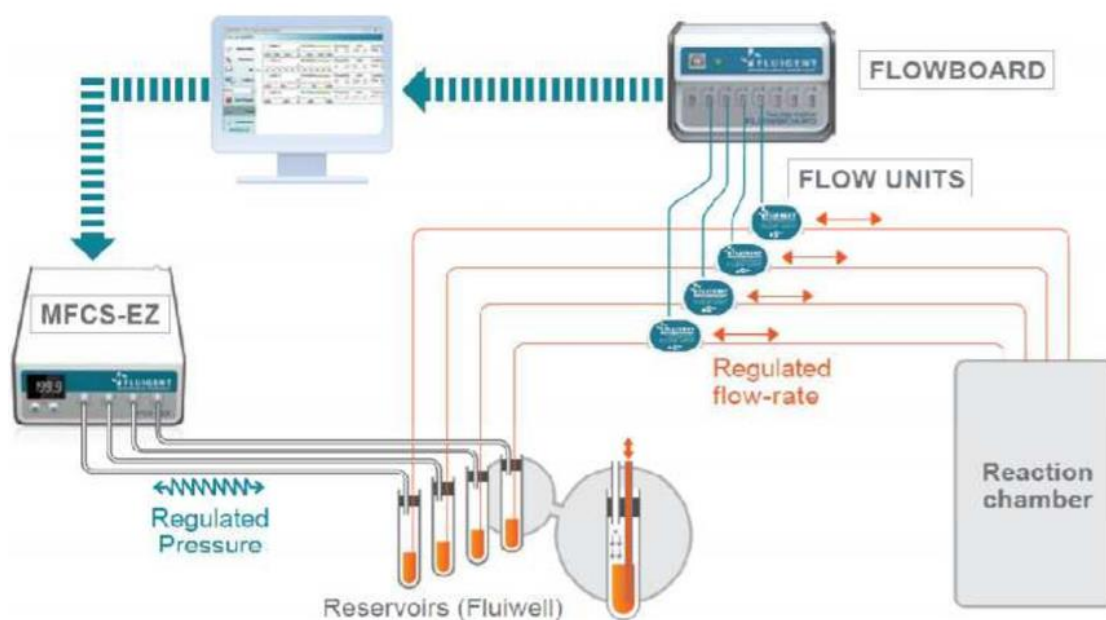


Figura 2-5. Experimento usando tanto MFCS-EZ como el FRP

El FRP calcula el caudal suministrado en base a la propagación del calor a través del flujo inyectado. La unidad se compone de dos partes:

- Un microcalentador que proporciona una cantidad mínima de calor ($<1\text{ }^{\circ}\text{C}$) al medio monitorizado (no tiene efecto en las células).

- 2 sensores de temperatura, situados a ambos lados del calentador,

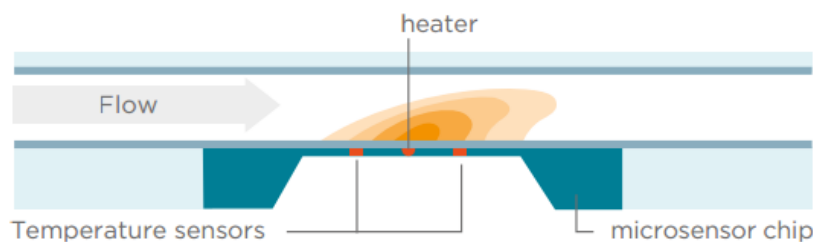


Figura 2-6. Obtención de medida del caudal por el FRP

Este tipo de medidas se está haciendo cada vez más común por la calidad de los resultados y el mínimo efecto que tiene sobre las muestras que recorren el conductor, sin embargo, se debe tener en cuenta el rango de temperatura de funcionamiento de esta unidad para proporcionar datos fiables el cual es de 10°C a 50°C. Lo que también observamos en la descripción del sistema anterior es que el FRP es un módulo únicamente de lectura lo que nos indica que el SW de control no está integrado en los módulos, sino que se realiza a través de los programas específicos de Fluigent.

FLOW UNITS	XS	S	M	L	XL	
Over pressure resistance between the FLOW UNIT sides (bar)	200		100	12	5	
Wetted materials :						
Internal Sensor Capillary Material	Quartz Glass (Fused silica)		Borosilicate Glass 3.3 (Duran®)			
Fitting material	100% PEEK™ (polyetheretherketone)					
Additional sealing material	None		Teflon®		ETFE (Tefzel®)	
Total internal volume	1 µL	1.5 µL	5.1 µL	< 30µL	< 90 µL	
Internal Sensor Capillary, Inner Diameter	25 µm	150 µm	430 µm	1 mm	1.8 mm	
Size	80 x 35 x 22 mm					Length x width x height
Length of the cable	1.5 m					
Weight	97 g					
FLOWBOARD						
Input	5V 100 mA					
Size	114 x 102 x 70 mm					Length x width x height
Weight	478 g					

Figura 2-7. Características y limitaciones del FRP

La figura superior concentra las especificaciones más relevantes de cada uno de los modelos existentes de Flow Units que hay que tener en cuenta. Teniendo en cuenta que el módulo adquirido por la Universidad de Sevilla es de tipo M podemos concluir que el uso de todos estos componentes va a tener únicamente impacto en el tamaño del sistema final por sus elevadas dimensiones pero que su uso es compatible con las aplicaciones que se van a desarrollar según a asegurado el equipo de investigación de MICRORAD.

2.2 Sistema Software

Al contrario que en el caso anterior, en el que el prototipo HW ya estaba fabricado, la interfaz software de LabVIEW solo era parcialmente funcional por lo que migrar el control del sistema a otro entorno de programación no hubiese supuesto una pérdida de tiempo ni recursos significativa. Sin embargo, la elección de

LabVIEW presenta grandes ventajas respecto a otros programas como Matlab.

Matlab™ y LabVIEW™ fueron creados bajo contextos distintos, en el primer caso lo primordial era conseguir una interfaz que permitiese realizar cálculos matemáticos complejos de manera sencilla mientras que en el segundo caso el objetivo era poder crear instrumentos virtuales para la adquisición de datos. Con el paso del tiempo, las evoluciones de ambas plataformas han convergido en sistemas con bastantes similitudes y que permiten interactuar con prácticamente cualquier HW externo.

En el contexto del proyecto, si bien la plataforma Arduino también se puede integrar en Matlab y, aunque no es trivial, se puede desarrollar una interfaz en Simulink que lleve a cabo las mismas funciones que se van a desarrollar en LabVIEW, este último presenta una interfaz gráfica mucho más intuitiva y fácil de modificar por cualquier usuario, justificando así su elección para un proyecto multidisciplinar como el de MICRORAD.

2.2.1 Introducción a LabVIEW™

LabVIEW™ es el entorno de programación gráfico de la compañía National Instruments diseñado para ingenieros y científicos y orientado al desarrollo de aplicaciones de pruebas, adquisición de datos de instrumentos, procesado, análisis de datos y control remoto de instrumentos. La naturaleza intuitiva de la programación gráfica de LabVIEW lo hace fácil de aprender y usar para todo tipo de aplicaciones, además posee una extensa gama de paquetes con los que integrar utilidades de otros entornos de programación como Arduino y Matlab. [10]



Figura 2-8. Pantalla de inicio de LabVIEW™

El Diseño gráfico de sistemas es un enfoque moderno que permite simplificar y acelerar drásticamente el desarrollo de diseños, la generación de prototipos y así desplegar sistemas embebidos rápidamente. El uso de un sistema de programación gráfica como LabVIEW tiene muchas ventajas [11] que facilitarán en gran medida el diseño de un programa.

- Flexibilidad del lenguaje de programación sin la complejidad de los entornos de desarrollo tradicionales.
- Tiempo de desarrollo más bajo que con cualquier otro lenguaje de programación.
- No es necesaria la instalación de compiladores o entornos de programación, LabVIEW viene preparado para comenzar a programar inmediatamente después de su instalación.
- Posibilidad de incorporar código escrito en otro lenguaje de programación.
- Trabajar con diferentes plataformas de hardware desde un mismo software, existiendo una gran compatibilidad con los equipos y dispositivos de cada marca.
- Funcionalidad Completa.
- Capacidades de E/S Integradas.

A los programas diseñados en LabVIEW se les llama VIs que son las siglas de Virtual Instruments, es decir, instrumentos virtuales. Cada VI está estructurado en dos partes que aparecen como dos ventanas separadas en las que se desarrolla una parte concreta de la funcionalidad del instrumento:

1. El Front Panel (Panel frontal) simula la interfaz de usuario. Es la carcasa exterior donde se encuentran todos los botones, leds y pantallas de información que, en el lenguaje de LabVIEW se engloban en dos grupos; controles e indicadores.
2. El Block Diagram (Diagrama de bloques) concentra todas las conexiones y circuitería interna que gestiona el funcionamiento de los elementos representados en el Front Panel.

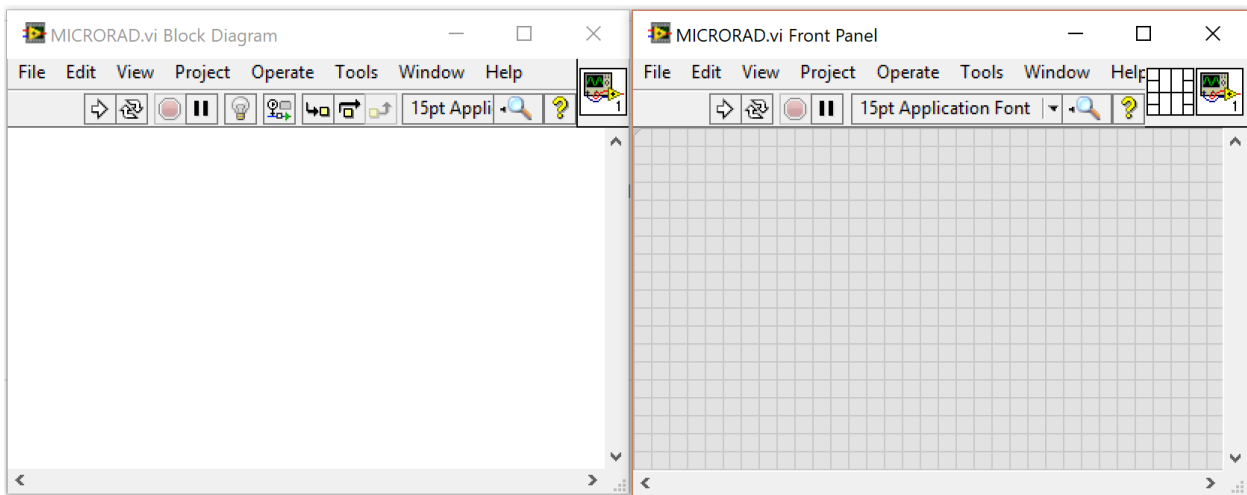


Figura 2-9. Ejemplo de Block Diagram (izquierda) y de Front Panel (derecha) LabVIEW

Dentro del Block Diagram la organización estándar es por bloques de los cuales se ejecutan un elemento de cada uno de izquierda a derecha y de arriba abajo hasta completar totalmente todos los bloques y actualizan los datos en el mismo orden. De esta manera se consigue una ejecución prácticamente paralela, pero hay que tener en cuenta que este orden puede afectar al resultado final si no se tiene cuidado o no se usan estructuras que aseguren un comportamiento estable del sistema.

3 MICRORAD HARDWARE

3.1 Estado del sistema

Uno de los problemas principales que se encontraron al comienzo de este proyecto fue que no había quedado recojido suficiente documentación sobre las distintas versiones del HW ni del SW que se habían desarrollado. En particular no se disponía de información sobre el PinOut del Arduino, es decir, no se sabía de qué se encargaba cada pin de la placa. Sin embargo, tras una primera inspección del código y del HW fue fácil descubrir que la disposición de los solenoides en la placa consigue facilitar el rutado y la localización e identificación de los componentes pero había provocado que únicamente se estuviese utilizando una fila de pines digitales de las dos disponibles en la Mega 2560 y que obligaba así a utilizar los pines analógicos como digitales. Esto último ha tenido gran relevancia en la parte SW y se detalla en la sección siguiente. Por último, queda mencionar que esta falta de información y los problemas encontrados durante las primeras pruebas hizo imprescindible realizar una serie de tests de continuidad sobre el prototipo para poder descartar los componentes cómo fuente de error y centrar así el esfuerzo en la parte SW. Efectivamente todos los componentes demostraron estar en buenas condiciones y respondían al Pinout del Arduino Mega 2560 descrito en la figura siguiente. La descripción más detallada del uso de cada pin se detalla en el apartado 3.3.

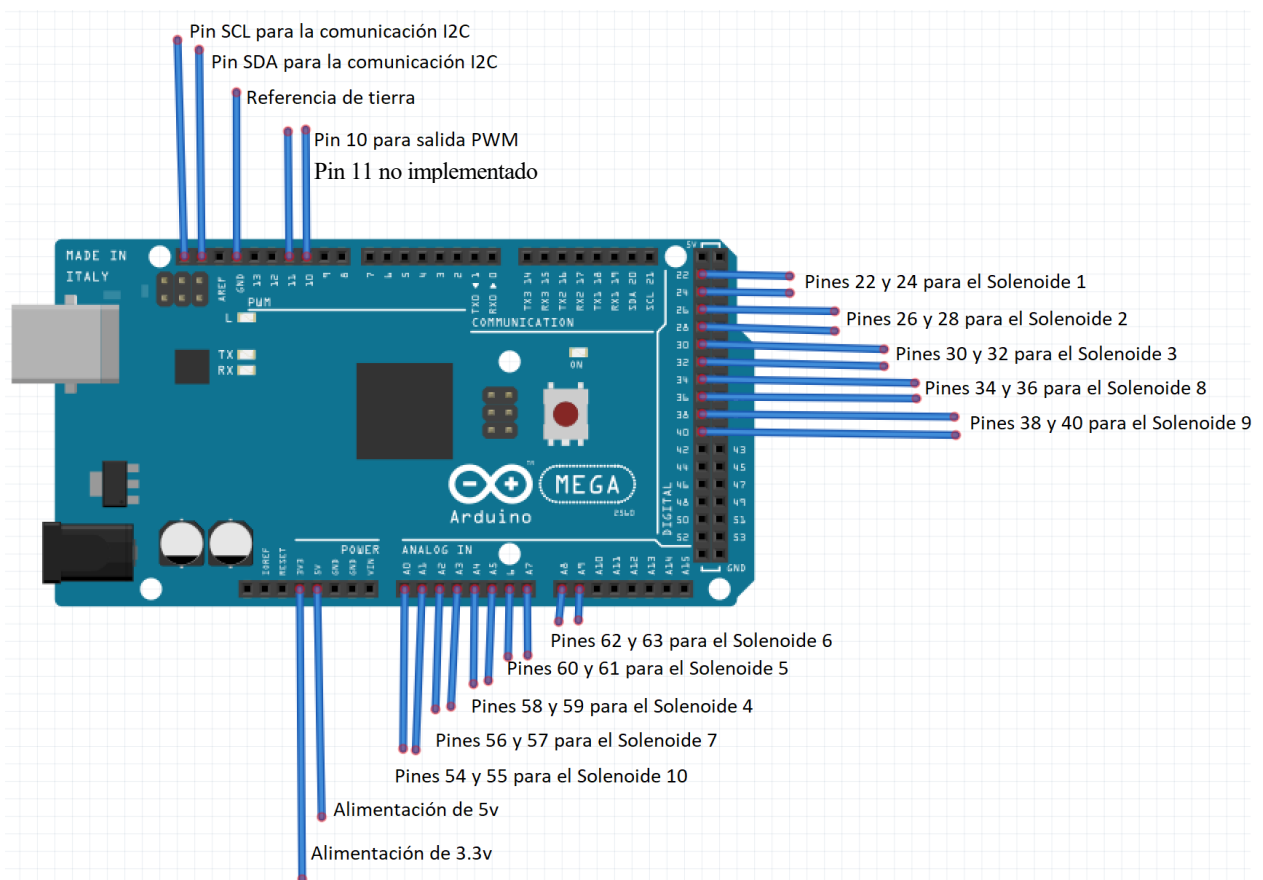






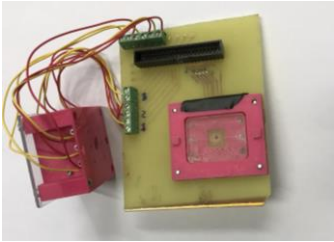






Figura 3-1. Pinout de la placa Arduino Mega 2560 utilizado

3.2 Componentes del sistema

A continuación, se presentan los componentes necesarios para poder realizar el montaje del experimento junto con una descripción de su funcionalidad principal. Será en el siguiente apartado cuándo se mostrará un primer acercamiento a las conexiones finales.

Nº	Componente	Uso	Identificación
1	Ordenador con todo el Software instalado. Consultar Anexo A.	Representación de datos y control del sistema completo	
2	HUB de al menos 4 USB con alimentación propia	Permite conectar todos los módulos a un único ordenador	
3	Fuente de alimentación	Proporciona 12 voltios y hasta 2 amperios a la placa 1	
4	Pendrive para FRCM	Contiene el Software FASTAB™ para cerrar el bucle de control de caudal usado por el FRCM.	
5	Arduino Mega 2560	Control de las platas 1 y 2 y envío de datos al ordenador.	

6	Placa 1	Conexión directa con el Arduino, contiene la electrónica destinada al control de los solenoides, así como la destinada al calentador.	
7	Placa 2	Contiene el reactor-on-chip, los 6 solenoides encargados de abrir y cerrar las válvulas, el calentador y el sensor de temperatura.	
8	Bus paralelo	Conecta las placas 1 y 2.	
9	MFCS-EZ	Controlador de presión. Debe conectársele una fuente de presión que no supere 1 bar y será capaz de proporcionar de manera estable por la salida habilitada hasta aproximadamente el 60% del valor de la presión suministrada.	
10	FLOWBOARD	Procesador de medidas de caudal, puede gestionar hasta 8 Flow units distintos	
11	FLOW UNIT	Sensor de Caudal talla M, capaz de proporcionar medidas fiables entre 0 y 80 ul/min cuando trabaja en el rango de temperatura [10°C, 50°C].	





12	Bolsa 1	Conexiones entre el MFCS-EZ y la bolsa 2	
13	Bolsa 2	Depositos para inyección de fluidos	
14	Bolsa 3	Conexiones entre la bolsa 2 y el Flow Unit	
15	Adaptador del conector de presión del MFCS-EZ.	Es el tubo a través del que se deberá suministrar una entrada de presión al MFCS-EZ.	

Tabla 3-1. Lista de componentes necesarios

3.3 Descripción del sistema

De los dos apartados anteriores queda por concretar qué elementos constituyen a las placas 1 y 2 y cómo funcionan ya qué, aunque su elección no ha sido fruto de este TFG, era necesario averiguar algunas especificaciones mínimas para poder sustituir el componente en caso de mal funcionamiento o similar.

La comunicación I2C (Inter-Integrated Circuit) se destina a la lectura del sensor de temperatura el cual se encuentra situado debajo de la estructura de solenoides de la placa 2. El sensor en cuestión es el TMP102 empaquetado en un SOT-563 y, entre las características más importantes, ofrece una resolución de 0.0625°C gracias a su CAD integrado de 12 bits, una precisión de $\pm 0.5^{\circ}\text{C}$ y un rango de funcionamiento $[-40^{\circ}\text{C}, +125^{\circ}\text{C}]$, todo ello con un tamaño inferior al de un SOT-23. Además, el hecho de que utilice I2C permitirá utilizar nuevos sensores y actuadores utilizando estos mismos pines y requiriendo muy poca modificación del SW.

El pin 10 del Arduino se utiliza para graduar el calentador mediante una salida PWM. Este calentador es en realidad una espira resistiva situada en la parte inferior de la estructura de solenoides en la placa 2 y que rodea al sensor de temperatura. Pese a que no se está utilizando actualmente, existe parte del rutado necesario para utilizar el pin 11 como otra salida PWM.

La apertura o cierre de cada válvula se basa en impulsos electromagnéticos de un solenoide (un electroimán) que trabaja junto a un muelle diseñado para devolver a la válvula a su posición neutral cuando el solenoide se desactiva. Los solenoides son muy útiles para realizar acciones a distancia sobre válvulas de control de gas y fluidos ya que funcionan muy bien en entornos peligrosos por altas temperaturas o por trabajar con productos químicos peligrosos. La estructura de solenoides de la placa 2 consta de 6 solenoides tipo TDS-K04E los cuales precisan de 2 pines cada uno para controlar su estado. Esto se realiza mediante un Puente H de la familia BD6220

configurado de manera que los pines de un solenoide tienen que tener valores opuestos y dependiendo de cual sea el pin que tenga valor lógico alto (TRUE) tendrá un estado u otro. Todos los pares de pines funcionan de forma análoga, si el pin cuya numeración es menor que la de su pareja tiene valor lógico alto entonces el solenoide estará cerrando la válvula y si ocurre al contrario la estará dejando abierta. Es decir, para el solenoide 1, si el pin 22=TRUE y el pin 24=FALSE entonces el solenoide estará cerrado. Cualquier otra combinación no es posible gracias al SW desarrollado.

La figura siguiente muestra cómo van a estar conectados los componentes de la tabla anterior en el experimento realizado, aunque el procedimiento paso a paso con las conexiones de los elementos reales se recoge en el Anexo B, *Manual de usuario: preparación del entorno*. Dicho experimento se ha ensamblado siguiendo los manuales de Fluigent y, antes de realizar alguna modificación sustancial, siempre se recomienda consultarlos de nuevo para asegurar la integridad de todos los components. Finalmente, la información necesaria para arrancar el programa de manera segura se encuentra recogida en el Anexo C, *Manual de usuario: Realización de una prueba*.

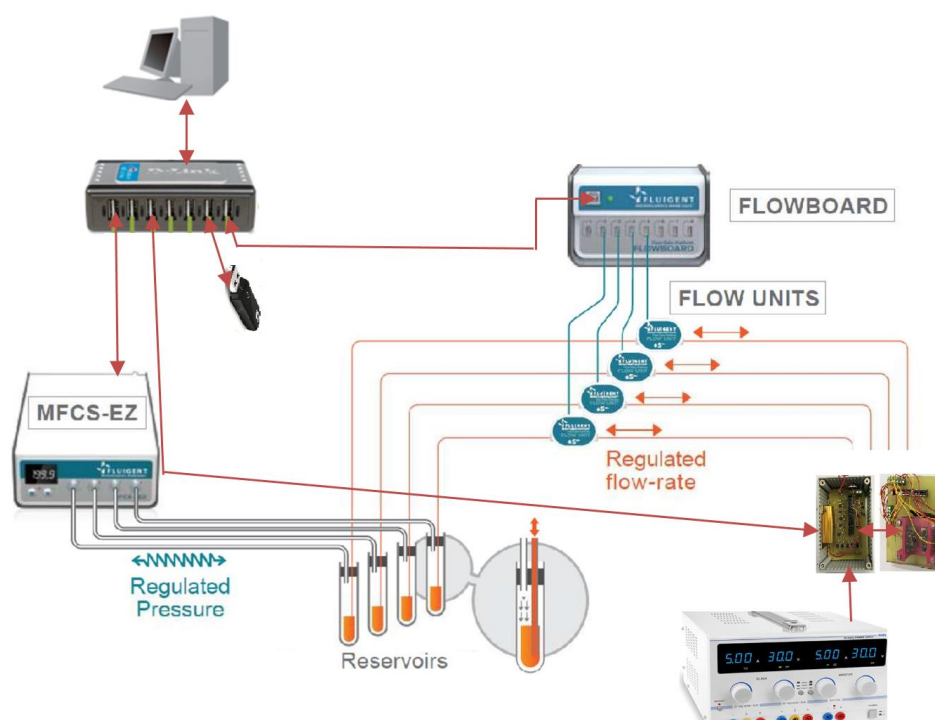


Figura 3-2. Esquema de conexiones del experimento completo

4 MICRORAD SOFTWARE

4.1 Estado del sistema

Al inicio de las pruebas sobre el HW no se disponía de una interfaz única que permitiese comprobar el funcionamiento completo del HW ni existía información sobre las conexiones con el Arduino. Como respuesta a esta situación se decidió llevar a cabo pruebas con todas las versiones SW encontradas para así tomar aquella que produjese los resultados más favorables como punto de partida.

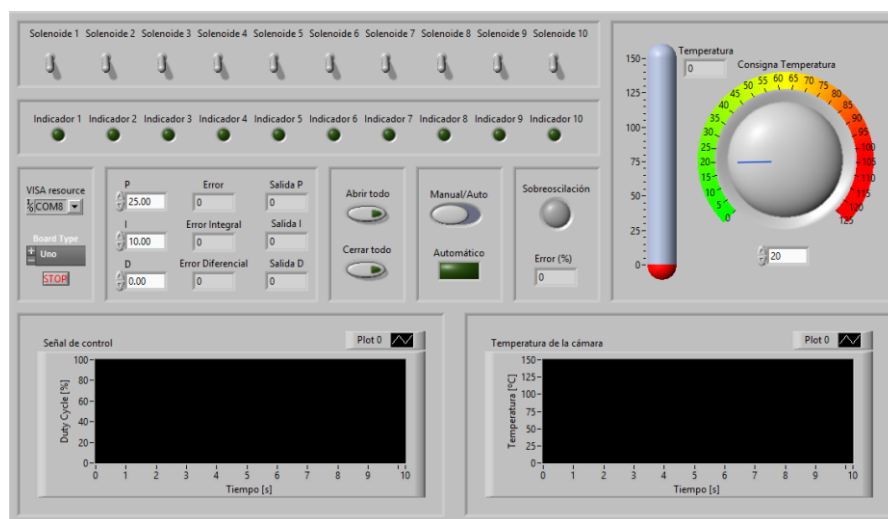


Figura 4-1. Front Panel de la versión software de partida

Al ejecutar el programa cuyo Front Panel se muestra en la *Figura 4-1* se pudo concluir que, en el mejor de los casos, el sistema de partida presentaría las siguientes dificultades iniciales:

- Solo permitía actuar sobre 3 de los 6 solenoides del sistema
- No actuaba sobre el calentador, pero sí que media la temperatura.
- La comunicación con el Arduino era intermitente y provocaba que se cerrase el programa de manera repentina.
- El selector Manual/Modo automático presentaba un comportamiento estático en el modo automático y no devolvía el control manual de manera estable.

Una vez comprobado que los componentes de las placas se encontraban en buenas condiciones se hizo patente que los errores anteriores debían estar localizados en algún punto del código. Con la ayuda del Pinout se pudieron corregir varios fallos en la numeración de los pines del Arduino y con ello todos los problemas de comunicación. No obstante, la solución debía implicar algún paso adicional ya que, aunque los solenoides 4, 5 y 6 ya eran accesibles cualquier acción sobre ellos provocaba la terminación del programa y de LabVIEW por completo. Por último se descubrió que LabVIEW realiza una comprobación sobre la numeración de los pines que Arduino no requiere necesariamente y es que, según explica National Instruments en su página oficial, utilizar los pines analógicos como digitales implica necesariamente modificar un VI del sistema para indicar al paquete de funciones de Arduino la nueva numeración de los pines y que todos van a ser utilizados como pines digitales de propósito general. Dicho VI se encuentra siempre en *C:\Program Files\National Instruments\LabVIEW 2013\vi.lib\LabVIEW Interface for Arduino \Utility\Check For Pin Out Of Range.vi* y se ha demostrado que para solucionar los fallos de comunicación con el Arduino de manera definitiva es suficiente con modificar el valor de *Num Pins* tal y cómo se muestra a continuación.

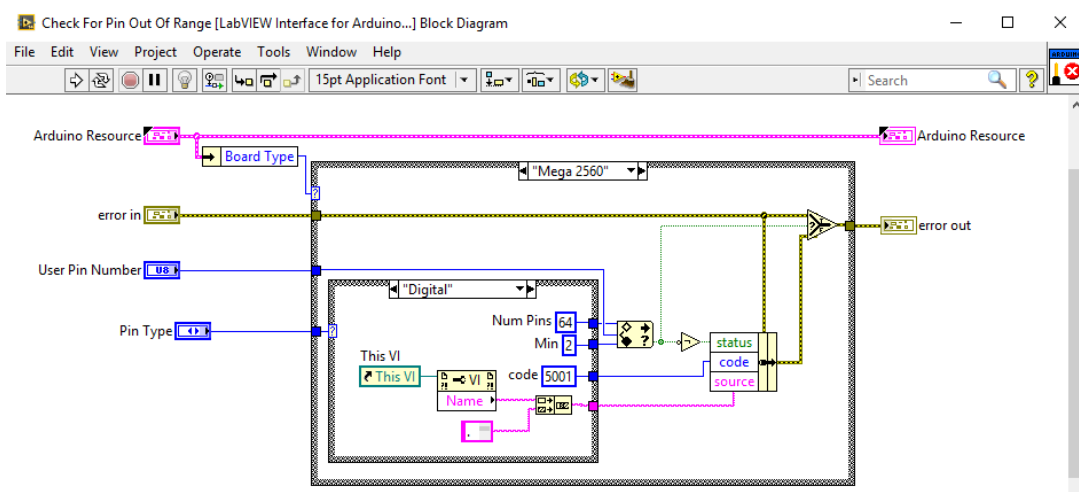


Figura 4-2. Modificación del VI Check For Pin Out Of Range.vi del sistema para el correcto funcionamiento del programa.

4.2 Integración de Fluigent en LabVIEW: FRCM

Estudiando las maneras de comunicar LabVIEW con Fluigent se han realizado demos con todos los bloques que proporcionan los paquetes de Fluigent para el entorno, divididos en dos grupos según interactúen con el MFCS-EZ o con el FRP. Así se ha comprobado que los requerimientos sobre el sistema son bastante elevados y las funciones existentes muy limitadas, de hecho, los ejemplos proporcionados de cada módulo demuestran que solo existe una manera de obtener los datos en cada caso y hay poco margen de modificaciones.

Esta falta de posibilidades la suple Fluigent con el FRCM, Flow Rate Control Module, pues se trata de un SW de la empresa que recibe Set Points de caudal y controla que se alcance asumiendo el control de ambos módulos. Es decir, al usar el FRCM perdemos la posibilidad de mandar ordenes directas al MFCS-EZ, pero una vez solicitamos un caudal determinado, es el SW de Fluigent a través del Pendrive, mostrado en la *Tabla 3-1*, el que se encarga de realizar todo el control y además lo hace en segundo plano. En contrapartida, el FRCM tiene también muy pocas posibilidades y por ejemplo no cuenta con ninguna función que muestre la evolución del caudal hasta haberlo alcanzado, sino que se limita a avisar de si este se alcanza o no. Por suerte, al ser el FRP un módulo de lectura exclusivamente, si que es posible utilizarlo a la vez que el FRCM sin entrar en conflicto con él y así tener la información que necesitamos.

La utilización de esta herramienta requiere de un trabajo previo y externo a LabVIEW en el que se calibran los dispositivos y se deciden las características del control. Dicho proceso se realiza partiendo del SW MAESFLO una vez se tenga conectado el Pendrive de FRCM y se haya seleccionado la plataforma FRP a través de la opción *Parameters*. Es también en este último apartado dónde se puede modificar el coeficiente de realimentación que usará después FRCM sobre el MFCS-EZ. Este coeficiente puede tomar valores entre 0 y 10 pero se recomienda mantener el valor por defecto de 5 ya que cualquier valor superior a este acelera la respuesta, pero puede llevar a inestabilidad y valores inferiores mejoran la estabilidad, pero realentizan la respuesta. [10]

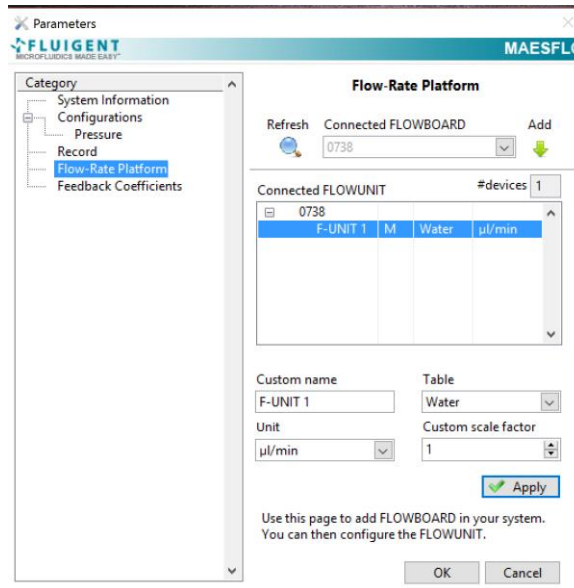


Figura 4-3. Selección de la FRP y coeficientes de realimentación

El wizard que aparece al pulsar sobre *Display* >> *Flow-rate Control Module* permite seleccionar el límite superior e inferior de presión que se podrá suministrar a nuestro sistema para asegurar el nivel de caudal solicitado. Después de varias pruebas se ha determinado que el intervalo 0 a 30 mbar permite obtener cualquier presión dentro del rango que ofrece el Flow Unit, de 0 a 80 $\mu\text{l}/\text{min}$, y asegura la no saturación del mismo. Todo valor superior a 35 mbar satura el sensor rápidamente y a por debajo de los 25 mbar el intervalo de presiones accesibles se reduce notoriamente.

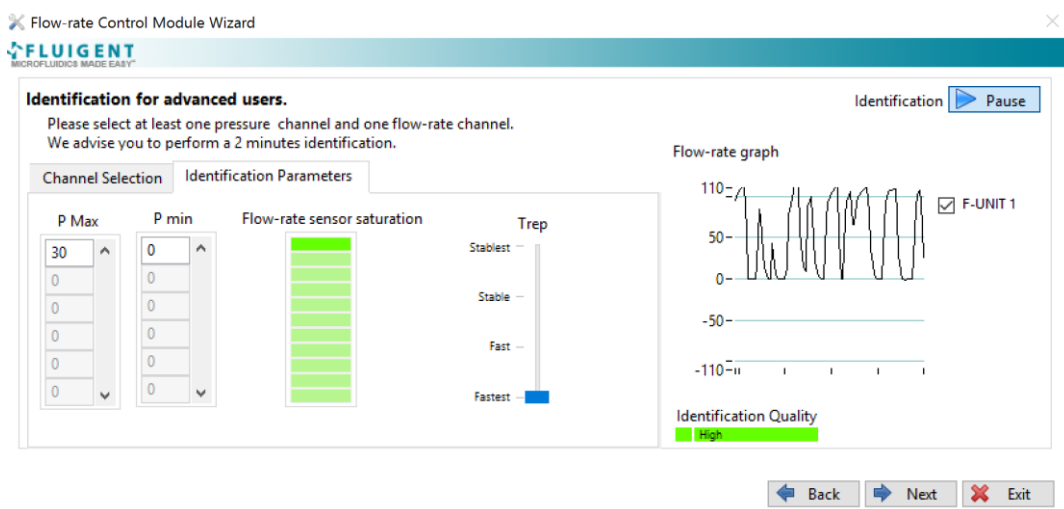


Figura 4-4. Ejemplo de calibración del FRCM a través de su wizard, expresado en % del caudal máximo posible

Una vez terminada la calibración se actualizará el indicador de la calidad de la identificación para decidir si volver a comenzar o guardar los resultados. En adición a estas comprobaciones se ha creído conveniente comprobar que utilizando LabVIEW se van a obtener los resultados esperados. Para ello se ha empleado un ejemplo básico del uso del FRCM que se ha elaborado siguiendo los manuales de Fluigent con la supervisión via email del equipo comercial de la empresa ya que no había sido proporcionado en origen. El resultado de esta colaboración fue el programa Fluigent FRCM SDK Example.vi cuyo código se muestra a continuación junto con los resultados obtenidos con su uso.

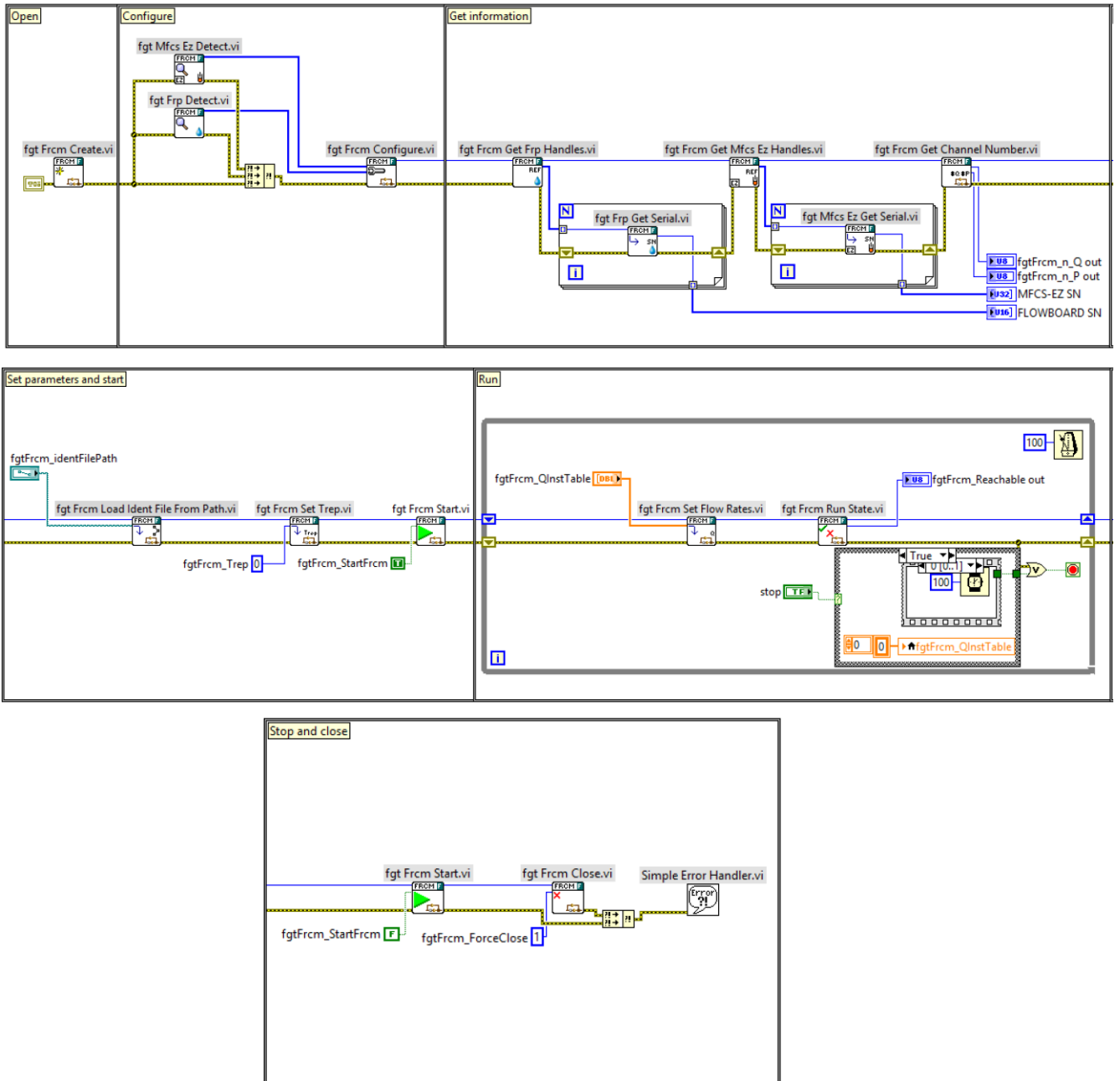


Figura 4-5. Block Diagram de Fluigent FRCM SDK Example.vi

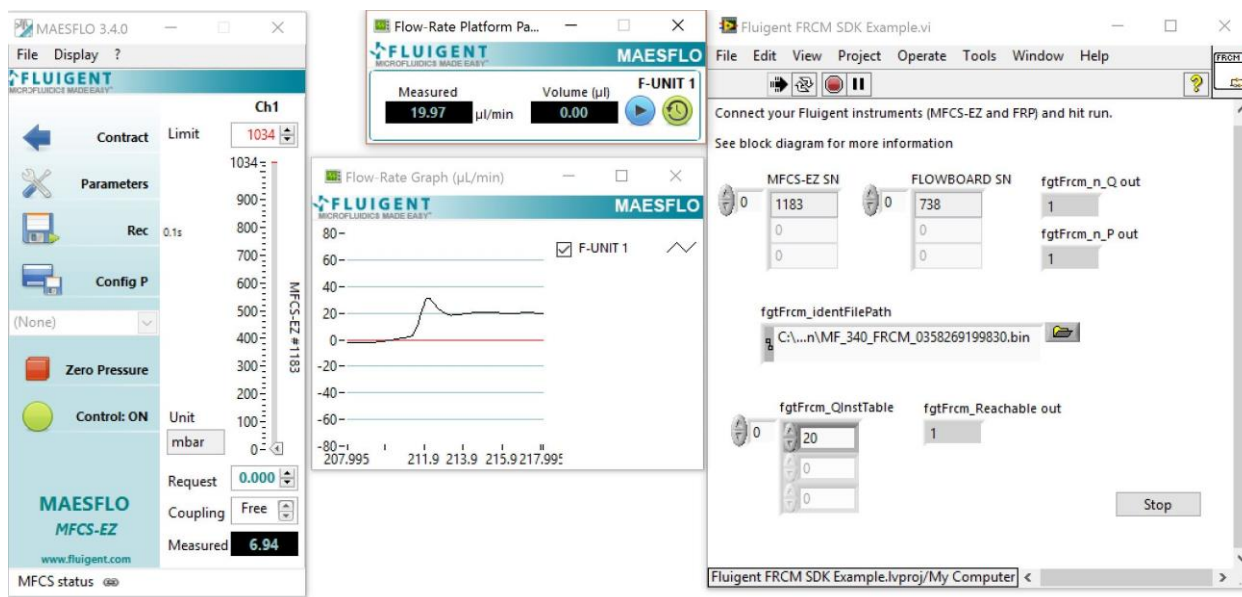


Figura 4-6. Respuesta del FRCM integrado en LabVIEW™

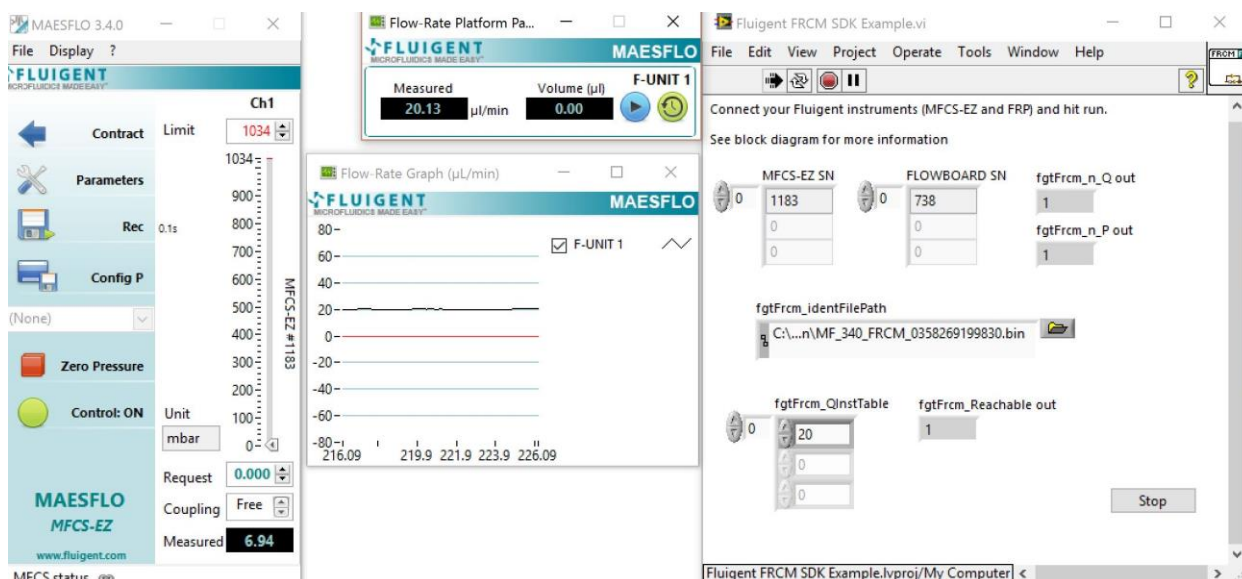


Figura 4-7. Respuesta estable del FRCM en el tiempo

Pese a que esta configuración se demostró capaz de responder positivamente a todas las pruebas realizadas al comienzo del proyecto, ha sido necesario repetirla al llegar el verano pues las condiciones de temperatura en el lugar de trabajo habían cambiado significativamente y con ello los requerimientos de la medida del caudal del Flow Unit ya que esta se fundamenta en la detección de gradientes de temperatura en el fluido. Así pues, se recomienda realizar calibraciones periódicas sobre el mismo siguiendo las indicaciones mencionadas aquí y en los manuales de Fluigent.

4.3 Definición del modo automático

Con objeto de generar la secuencia de estados para el funcionamiento automático del sistema, se han investigado varias líneas de programación de las cuales se ha evaluado tanto la facilidad de aprendizaje cómo la compatibilidad con LabVIEW:

1. LabVIEW™: En este entorno de programación existen tres tipos de archivos con los que se puede

trabajar; archivos de texto .txt, archivos de mediciones y archivos Excel.

- a. Write/read to measurement file: escribe datos en un archivo de medición basado en texto (.lvm), un archivo binario con cabeceras (.tdm) o un archivo binario medición sin cabeceras (.tdms). En LabVIEW 2013 y posteriores, también ofrece la opción de escribir en archivos de Microsoft Excel (.xlsx).
 - b. Write/read to spreadsheet file: está orientado a la gestión de bases de datos de pequeño y mediano tamaño mediante archivos Excel. Aunque la escritura es sencilla para procesos que vuelcan gran cantidad de datos, no está pensado para leer ni modificar ciertas secciones del fichero, pudiendo llegar a hacer dicho proceso muy complejo.
 - c. Write/read to a text file: para este entorno de programación un fichero de texto se lee como una única cadena o string y se escribe como una sola o como combinación de ellas. Dichos strings tienen la característica de que cada elemento contiene todo aquello que haya entre dos saltos de carro. Es decir, se realiza un acceso indirecto y limitado al fichero a través de un único string que contiene todas las líneas del fichero ordenadas en posiciones sucesivas. Además de esto, la colección de funciones orientadas a string es muy amplia y se puede realizar prácticamente cualquier cosa.
2. Programación en Python: Un fichero es, para Python, un objeto que se crea con una operación de apertura, y sobre el que se pueden ejecutar funciones de lectura, escritura y cierre. Refleja un fichero en el disco duro, y dependiendo del modo de apertura, podremos hacer unas cosas u otras, incluyendo la pérdida total de la información del mismo. Python nos permite además generar un entorno gráfico, pero presenta una dificultad media para la aplicación que se pretende conseguir. [8]
 3. Programación en Visual Basic: Visual Basic está diseñado para crear de manera productiva aplicaciones con seguridad de tipos orientadas a objetos. Visual Basic permite establecer como destino tanto dispositivos móviles, web como Windows. La facilidad del lenguaje permite crear aplicaciones para Windows en muy poco tiempo, en otras palabras, permite un desarrollo eficaz con menor inversión tanto en tiempo como en dinero. En contrapartida no existe forma alguna de exportar el código a otras plataformas diferentes a Windows [9].

Ya que todas las opciones presentan pros y contras se decidió realizar pruebas de gestión de ficheros de texto con cada herramienta y así elegir la más intuitiva y cómoda. Bajo estos criterios se ha elegido utilizar el propio entorno LabVIEW debido a su conveniencia durante el tiempo de ejecución y poco efecto sobre el mismo, carácter gráfico versátil y facilidad de generar un fichero con la estructura más parecida a los requerimientos de lectura de las funciones existentes.

Una vez decidido LabVIEW y un fichero de texto plano como base sobre la que trabajar se comenzó a investigar las diferentes opciones existentes para gestionar dicho fichero. En este respecto hay que tener en cuenta que la manera en la que este entorno gestiona los ficheros de texto es copiando el contenido completo del mismo en un string cuyos elementos son cada línea del fichero. Una de las mayores ventajas de esto es que no consume apenas recursos del programa ya que no necesita mantener abierto el fichero durante la ejecución completa y una vez se ha trasladado el contenido a una variable tipo string existen multitud de funciones con las que modificarla de manera muy personalizada y rápida.

En una primera aproximación se barajó la posibilidad de usar la función de librería *Prompt User* dentro de la paleta *Dialog and User Interface*, la cual permite definir una lista de elementos a completar por el usuario. En el caso práctico mostrado a continuación al pulsar sobre la opción *Nueva secuencia* se abrían una serie de formularios como los de la *Figura 4-8* que se repetían y guardaban en función de los otros dos controles del programa.

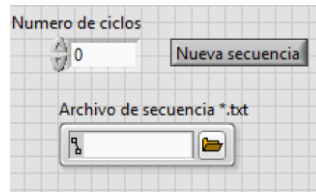


Figura 4-8. Generación de un fichero mediante la función Prompt User

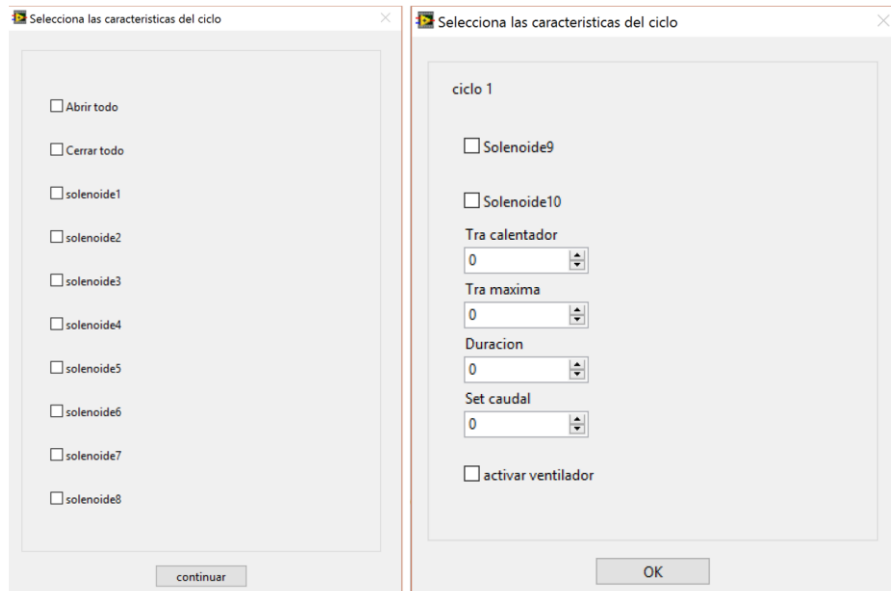


Figura 4-9. Interfaz obtenida mediante la función Prompt User

Es cierto que esta es una opción muy rápida de implementar y en la que es muy fácil incluir nuevas funcionalidades. Sin embargo, las deficiencias de esta opción se hicieron patentes rápidamente al darnos cuenta de que cualquier modificación sobre un elemento de un estado requería reescribir completamente la secuencia desde cero. Además, mostrar las características de un estado específico no resulta posible usando la misma interfaz de formularios, así como tampoco lo es modificar ni guardar dicho estado sin alterar el resto de la secuencia.

A raíz de la problemática anterior se hace notoria la necesidad de una interfaz gráfica que permita visualizar y modificar un estado cualquiera de la secuencia en cualquier momento. Así pues, lo más lógico será recrear ordenadamente los actuadores del Front Panel final en una interfaz no conectada al Arduino.

Siguiendo estos pasos se ha decidido que el fichero de una secuencia tenga la estructura de la *Figura 4-10*, de la que se desprende que cada estado de una secuencia ocupará 18 líneas consecutivas del fichero partiendo de la línea 1 pues la 0 se reserva para mantener el valor actualizado del número de estados existentes en la secuencia. Los elementos del fichero se corresponden a las especificaciones presentadas durante los objetivos del proyecto y van a permitir crear estados con una variedad de casuísticas de salto considerable. Por ejemplo, un estado podrá actuar sobre los elementos del sistema hasta la consecución de; alcanzar una temperatura determinada por CCTemperatura dentro del reactor-on-chip, detectar que ha pasado un volumen determinado por CCVolumen por el Flow Unit, finalizar una temporización marcada por CCDuración o incluso la primera que ocurra de una combinación de las anteriores. Además, se explicará más adelante que se ha decidido introducir un control en la interfaz principal del programa que permita finalizar un estado cuando el usuario decida actuando con la mayor prioridad posible sobre el resto.

Al final de esta sección se encuentran las distintas pruebas que se han realizado para comprobar que se generan estas secuencias correctamente y es en la sección 5 cuando se presentarán 3 secuencias tipo que se probarán sobre el sistema real. Sin embargo, es cierto que la generación de secuencias con estructuras que no sean lineales no es inmediata y la generación de bucles complejos solo puede hacerse mediante la ejecución repetida de una secuencia base por parte del usuario o copiando uno a uno los estados.

0: Número de estados de la secuencia
1: Abrir todo
2: Cerrar todo
3: Solenoide 1
4: Solenoide 2
.....
12: Solenoide 10
13: SPTemperatura
14: SPCaudal
15: SPVentilador
16: CCTemperatura
17: CCDuración
18: CCVolumen
19: Abrir todo
.....

Figura 4-10. Estructura de un fichero .txt con una secuencia

De entre las funciones orientadas a string encontramos *Scan From String* que, junto a *Build Array*, suponen el pilar principal para leer y modificar una secuencia. Esta primera permite leer el string desde un offset inicial, sin embargo, este solo puede expresarse en nº de caracteres por lo que acceder a una única línea del fichero no puede hacerse en un solo paso siguiendo este proceso. Otra manera de entender la misma función es darse cuenta de que, si se expanden las salidas que proporciona la función hasta tener 18 y de alguna manera conseguimos quitar el primer elemento del fichero que almacena el numero de estados en la secuencia, la función estará devolviendo de forma ordenada los elementos de un estado en cada salida de la función. Es decir, que para obtener la línea 12 correspondiente al valor del solenoide 10 del primer estado bastaría con realizar un bloque como el de la *Figura 4-11*.

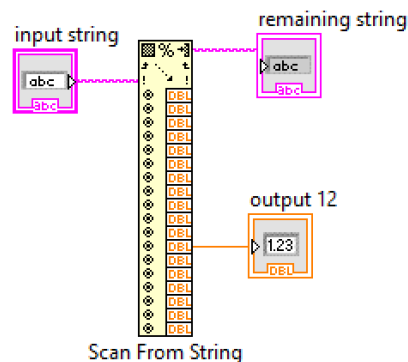


Figura 4-11. Simplificación de una estructura básica para leer un estado de una secuencia

Después de varias versiones y de comprobar todas las posibilidades se ha llegado a la conclusión de que el protocolo más efectivo para tratar un string que contenga la secuencia completa se basa en la utilización de bucles For que, teniendo en cuenta el número de estados de la secuencia, vaya recorriendo el string en tramos de 18 elementos de manera que al final de cada iteración se actualiza el valor de input string con remaining string para poder comenzar con el elemento adecuado en cada iteración.

No obstante, es necesario afinar un poco más este enfoque para poder llegar a cualquier estado de la secuencia. La función final encargada de dicho proceso se puede consultar en el Anexo D entre el código de *LeeEstado.vi*.

4.4 Planteamiento global del sistema

Las características de este proyecto hacen necesario diferenciar al menos dos Front Panels distintos, uno que tenga la interfaz de usuario principal y otro destinado a la gestión de las secuencias del modo automático a los que, de ahora en adelante, llamaremos *FRONT PANEL* y *ModoAutomatico.vi* respectivamente. Esta situación ha hecho necesario establecer un orden sobre qué controles e indicadores son los que van a tener efecto sobre el Arduino y Fluigent y qué otros van a servir como refuerzo visual del usuario. En consecuencia, se ha decidido que la comunicación con el Arduino se haga exclusivamente mediante variables globales guardadas en *Global.vi* y que únicamente estén conectadas al *FRONT PANEL*.

A continuación, se van a presentar los front panels de cada VI mencionado arriba junto con una tabla en la que se describirá la funcionalidad de cada elemento que aparece. En el caso de *Global.vi* no será necesaria una tabla pues se encuentran ordenadas por funcionalidad y aquellas que lo requieren están referenciadas en las tablas correspondientes. Es decir, en la *Tabla 4-1* aquellos controles o indicadores con efecto sobre el Arduino o Fluigent tienen una variable global asociada y en la *Tabla 4-2* se muestra una columna similar de variables globales asociadas, pero en este caso se utilizan para la comunicación entre las distintas funciones del proyecto destinadas a la gestión del fichero de secuencias.

De *Global.vi* también puede entenderse que las variables globales de la parte izquierda corresponden a aquellas que se utilizan en el *FRONT PANEL* principal del programa mientras que aquellas situadas a la derecha son las utilizadas en el resto de VIs secundarios, tanto *ModoAutomático.vi* como cualquier función necesaria.

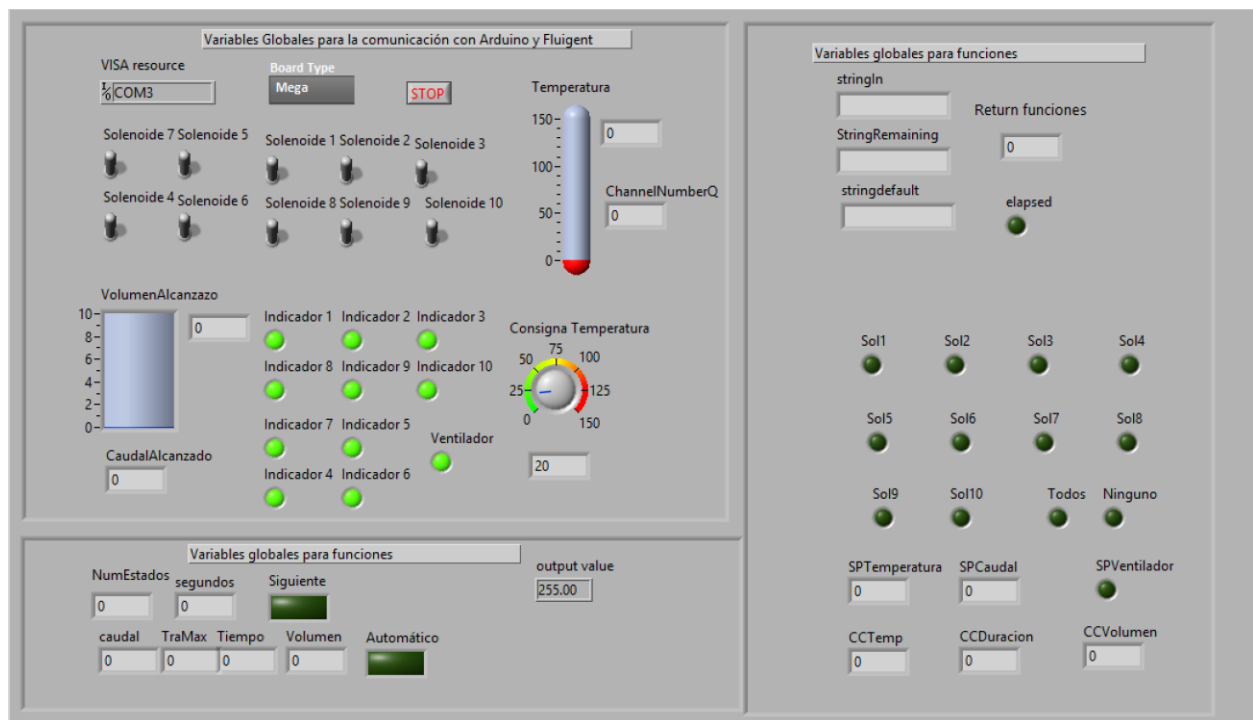


Figura 4-12. Global.vi

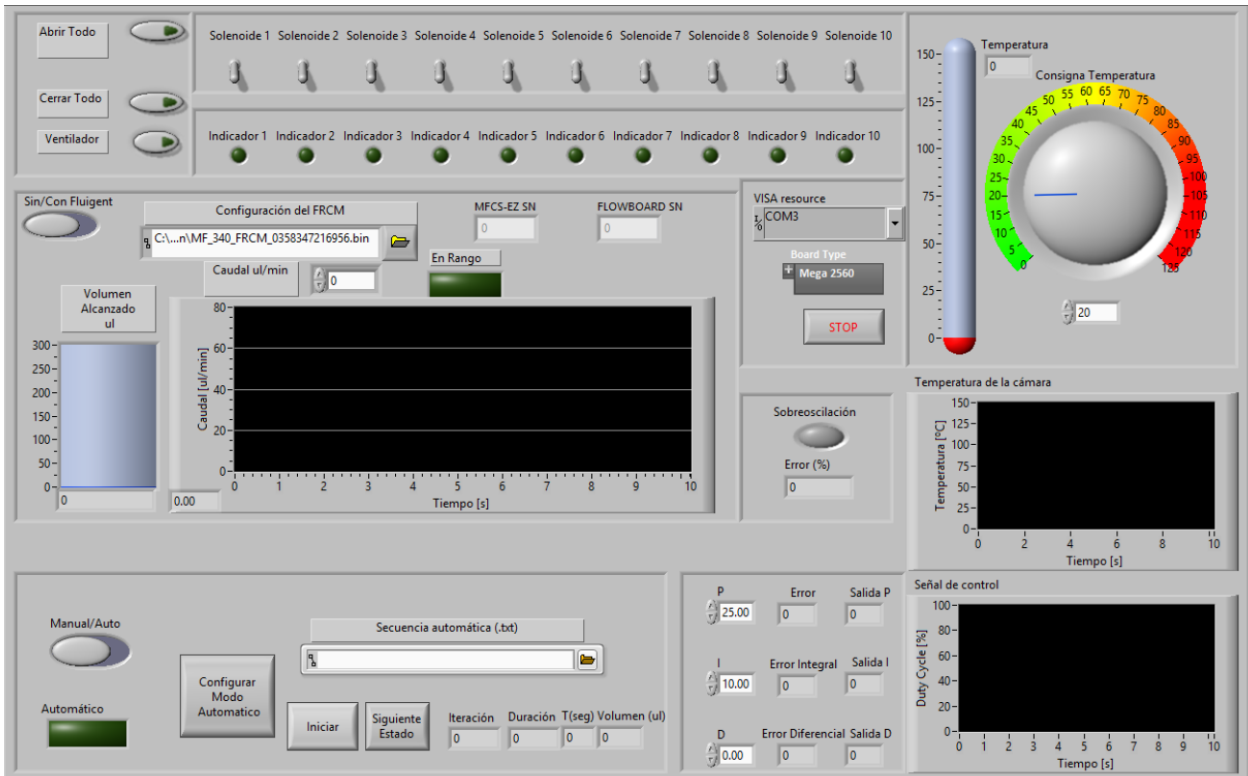





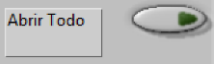
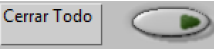
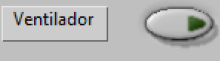
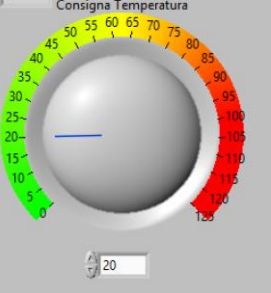
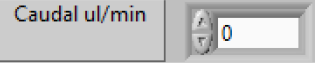
Figura 4-13. Interfaz de usuario del FRONT PANEL principal del programa.

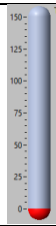
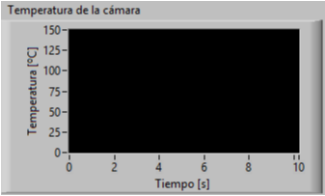
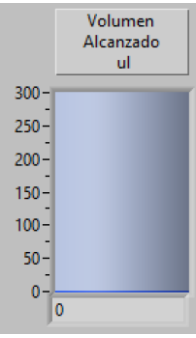
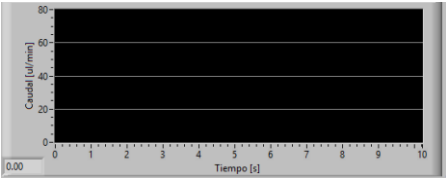

Aquí se puede comprobar que se ha conservado todo lo que había en el programa de origen, pero dejando espacio para las nuevas funcionalidades. Durante el modo manual se puede interactuar con todos los elementos del interfaz excepto con el cuadro inferior izquierdo del que únicamente estará habilitado el control *Configurar Modo Automático*. Además, la estructura interna del programa ha permitido poder trabajar, incluso durante el modo automático, sin ninguno de los componentes de Fluigent, lo cual permite agilizar ciertas tareas de investigación. Para lo que habrá que hacer es mantener el control *Sin/Con Fluigent* en la posición que aparece en la *Figura 4-13*.

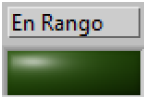
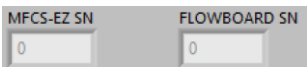
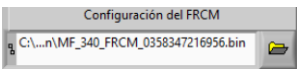
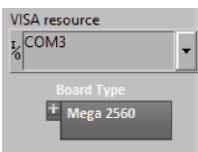
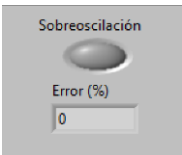
Aunque todos los controles e indicadores se definen abajo, cabe destacar la existencia del control *Siguiente Estado* en el *FRONT PANEL*. Esto se debe a que se ha querido simplificar al máximo la interfaz de *ModoAutomático.vi*, pero a la vez fomentar la mayor versatilidad posible. Es por ello por lo que se ha creído conveniente incluir, además de las condiciones de cambio configurables en cada estado, un botón que permita al usuario saltar al siguiente estado de una secuencia en cualquier momento.

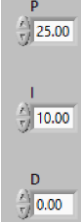
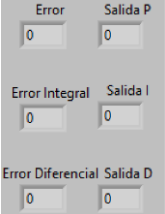
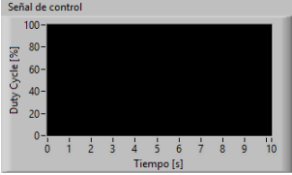
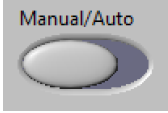
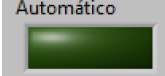
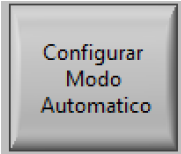
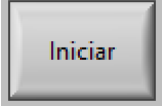
Alguna de las utilidades ya encontradas a esto ha sido al usar ese botón en lugar de *CCDuración* en los casos en los que no se sabía cuánto tiempo mantener una temperatura determinada y también para forzar la salida de una secuencia sin cerrar bruscamente el programa.

Nombre	Descripción	In/Out	Front Panel		Variable asociada	Global
			Lee	Modific		
Stop	 Botón de Stop para para la ejecución del programa completo. Cierra de	In	Si	No	Stop	

	manera ordenada todos los bloques antes de terminar el programa.				
Run	 Botón de inicio para arrancar el programa.	In	Si	No	
Solenoides [10] Indicador[10]	 <p>Solenoid= False-->Cerrado Solenoid= True-->Abierto</p> <p>Gestionan el acceso a los solenoides de la placa durante el modo manual y muestran las ordenes enviadas a los solenoides durante el modo automático.</p>	In	Si	Si	Solenoid [10] Indicador [10]
Abrir todo	 <p>Aplica True a todos los solenoides</p>	In	Si	Si	
Cerrar todo	 <p>Aplica False a todos los solenoides.</p>	In	Si	Si	
Ventilador	 <p>Este control accionará el extractor de vacío cuando se instale</p>	In	Si	Si	Ventilador
Consigna Temperatura	 <p>Envía un Set Point de temperatura al calentador, se expresa en °C.</p>	In	Si	Si	Consigna Temperatura
fgtFRCM_QInst aTable	 <p>Entrada de la función fgtFRCM_set Flow Rates que envía la orden de alcanzar un caudal determinado al FRCM, se expresa en ul/min.</p> <p>Durante el modo automático requiere cambiar de tipo la variable global</p>	In	Si	Si	Caudal

	asociada para ser compatible con las funciones de FRCM.				
Temperatura	 <p>Representa la temperatura real leída en el reactor-on-chip en un instante.</p>	Out	No	Si	Temperatura
Temperatura de la cámara	 <p>Muestra la evolución de la temperatura de la cámara en el tiempo.</p>	Out	No	Si	
Volumen Alcanzado	 <p>Representa el volumen absoluto que ha pasado por el Flow Unit hasta dicho instante, expresado en ul.</p> <p>Solamente se lee durante el modo automático, cuando existe una condición de cambio de estado por volumen alcanzado para poder obtener el volumen a alcanzar con respecto al que ya se tenía.</p>	Out	Si	Si	Volumen alcanzado
Caudal Alcanzado	 <p>Representa el caudal que está midiendo el FRCM expresado en ul/min.</p> <p>Esta medida se realiza en intervalos de 100ms lo cual nos permite obtener el volumen correspondiente</p>	Out	No	Si	Caudal Alcanzado
Sin/Con Fluigent	 <p>Control que permite ejecutar el programa sin la parte del experimento correspondiente a Fluigent.</p> <p>Se puede iniciar el uso de Fluigent en cualquier momento, pero para iniciarlo</p>	In	Si	Si	

	de nuevo después de haberlo parado se recomienda parar la ejecución completa con el botón de Stop ya que los bloques de Fluigent pueden no haber vuelto a sus valores por defecto correctamente.				
En Rango	 <p>Indicador asociado a la salida de la función fgtFRCM_RunState que muestra si el valor de Caudal va a alcanzarse (>0) o no puede asegurarse (=0) pero cambiando el tipo de dato para hacerlo más visual.</p>	In	Si	Si	
MFCS-EZ SN y FLOWBOARD SN	 <p>Indicadores con los números de serie de los módulos de Fluigent encontrados.</p> <p>Ha sido necesario establecer la condición de no continuar el programa hasta localizar los módulos para evitar cierres repentinos del programa.</p>	In	Si	Si	
fgtFrcm_identFilePath	 <p>Fichero de configuración del FRCM para la calibración de los módulos de Fluigent.</p>	In	Si	No	
VISA resource y Board Type	 <p>Controles dedicados a la detección y elección de la placa Arduino.</p>	In	Si	No	VISA resource y Board Type
Sobreoscilación y Error(%)	 <p>Indicador de sobreoscilación y Error cometido entre la temperatura alcanzada y la consigna de temperatura.</p>	Out	No	Si	

P, D, I	 <p>Coeficientes del PID para la gestión de la temperatura mediante PWM.</p>	In	Si	No	
Error, Error Integral, Error Diferencial, Salida P, Salida I, Salida D.	 <p>Indicadores obtenidos durante la ejecución del PID</p>	Out	No	Si	
Señal de control	 <p>Muestra la evolución del duty cycle del PWM tras la actuación de PID.</p>	Out	No	Si	
Manual/Auto	 <p>Selector de modo manual o modo automático que impide el uso del botón iniciar si no se ha colocado en la posición Auto primero.</p> <p>Se utiliza junto con las condiciones de cambio, el botón de siguiente estado y el botón de stop para terminar el modo automático en cualquier momento.</p>	In	Si	Si	Automático
Automático	 <p>Indicador de que el modo automático está funcionando</p>	Out	No	Si	
Configurar Modo Automático	 <p>Hace que el Front Panel de ModoAutomatico.vi aparezca en primer plano.</p>	In	Si	No	
Iniciar	 <p>Botón que inicia la secuencia cargada en el Abrir *.txt</p>	In	Si	No	

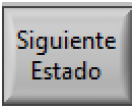
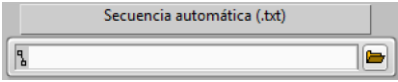
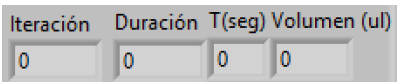
<p>Siguiente estado</p>	 <p>Permite al usuario saltar al siguiente estado de la secuencia aunque no se haya cumplido ninguna de las condiciones de cambio.</p> <p>Se utiliza junto con las condiciones de cambio, el botón de Manual/Auto y el botón de stop para terminar el modo automático en cualquier momento.</p>	In	Si	Si	Siguiente
<p>Abrir *.txt</p>	 <p>Es el selector de los archivos de secuencias creados con el ModoAutomatico.vi</p>	In	Si	No	
<p>Iteración, Duración, T(seg) y Volumen (ul)</p>	 <p>Información sobre el estado que se está ejecutando en ese momento:</p> <p>Iteración muestra el orden, de 1 a N.</p> <p>Duración muestra los segundos que puede durar un estado cómo máximo.</p> <p>T(seg) muestra el tiempo en segundos que ha pasado desde que se ha iniciado un estado que tenia Duración>0.</p> <p>Volumen muestra la suma, en el instante inicial, de Volumen Alcanzado con la condición de cambio de volumen de dicho estado.</p>	Out	Si	No	<p>Iteración→NumEstados</p> <p>Duración→Tiempo</p> <p>T(seg)→Segundos</p> <p>Volumen(ul)→Volumen</p>

Tabla 4-1. Resumen de controles e indicadores del FRONT PANEL

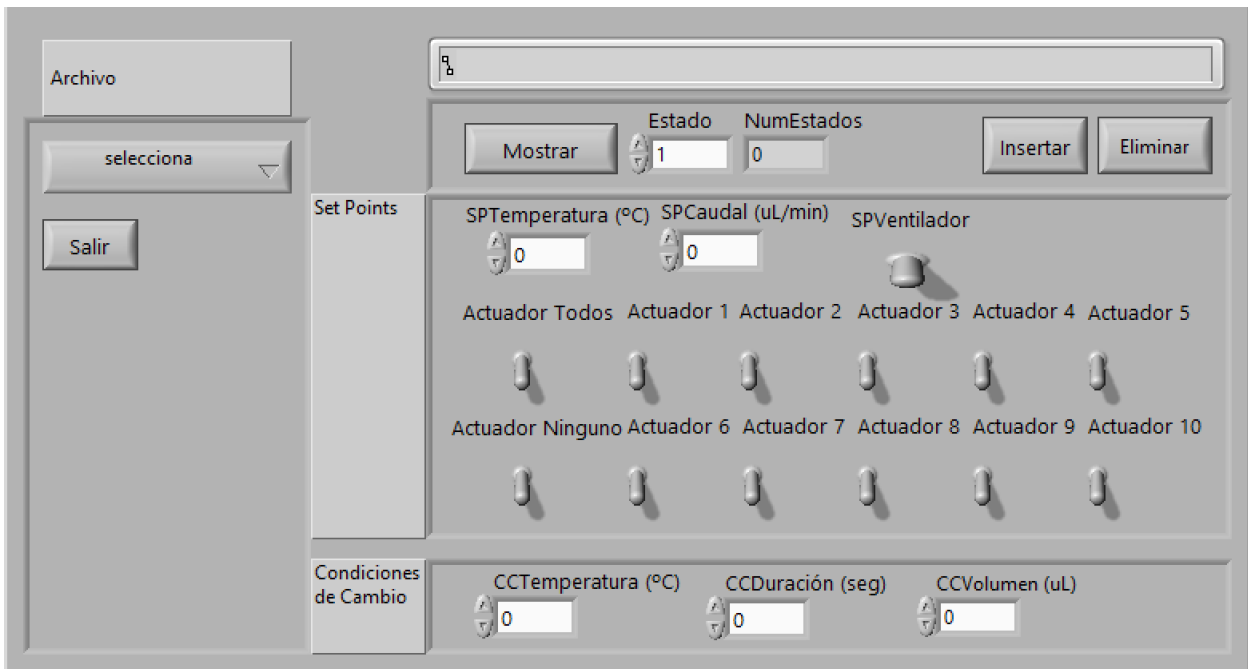
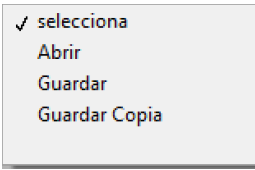
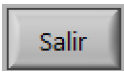
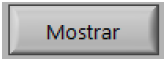
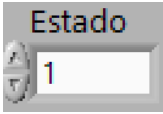
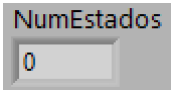
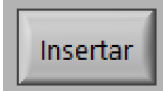
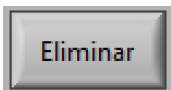

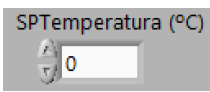


Figura 4-14. Front Panel de ModoAutomatico.vi para la gestión de secuencias automáticas

Nombre	Descripción	In/Out	Modo Automatico.vi		Variable Global Asociada
			Lee	Escribe	
Selecciona	 <p>Actúa como la pestaña de “archivo” de un editor de texto.</p>	In	No	Si	
Salir	 <p>Una vez aparezca el Front Panel de ModoAutomático.vi este es el botón que nos permitirá volver al programa principal</p>	In	No	Si	
Mostrar		In	No	Si	

	<p>Junto con el control “Estado” permite visualizar un estado concreto de la secuencia</p>				
Estado	 <p>Selecciona la posición del estado sobre el que se quiere actuar</p>	In	No	Si	
Numestados	 <p>Muestra el número de estados que tiene el fichero seleccionado en cada momento.</p> <p>Tanto para crear como para eliminar un estado es necesario actualizar su valor y volver a guardarlo en el fichero.</p>	Out	Si	Si	Numestados.
Insertar	 <p>Permite incorporar un nuevo estado en la posición seleccionada por el control Estado. En caso de insertar un estado en una posición intermedia de la secuencia los siguientes estados serán desplazados a a la derecha. Actualiza el valor de NumEstados</p>	In	No	Si	
Eliminar	 <p>Elimina de la secuencia el estado seleccionado por el control Estado.</p> <p>En caso de eliminar un estado en una posición intermedia de la secuencia los siguientes estados serán desplazados a a la izquierda. Actualiza el valor de NumEstados</p>	In	No	Si	
Vinculo	 <p>Contiene la ruta hasta el fichero de secuencia sobre el que se está trabajando</p>	In	No	Si	
SPTemperatura	 <p>Valor en °C que se mandará a Temperatura durante la ejecución del estado correspondiente.</p>	In/Out	Si	Si	SPTemperatura

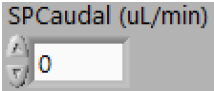
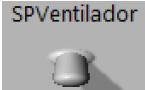
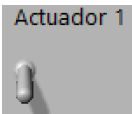
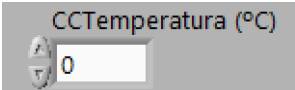
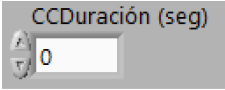
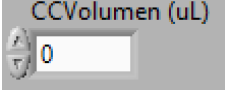
SPCaudal	 <p>SPCaudal (uL/min)</p> <p>Valor en ul/min que se le mandará a Caudal durante la ejecución del estado correspondiente.</p>	In/Out	Si	Si	SPCaudal
SPVentilador	 <p>SPVentilador</p> <p>Estado del extractor que se asignará a Ventilador durante la ejecución del estado correspondiente.</p>	In/out	Si	Si	SPVentilador
Actuador [10] ActuadorTodos ActuadorNinguno	 <p>Actuador 1</p> <p>Posición del solenoide correspondiente que se asignará a Ventilador durante la ejecución del estado correspondiente.</p>	In/Out	Si	Si	Sol [10] Todos Ninguno
CCTemperatura	 <p>CCTemperatura (°C)</p> <p>Temperatura en °C a la que se pasará al siguiente estado. Este se enviará a la función CondicionesCambio.vi a través de la variable Tramax.</p>	In/Out	Si	Si	CCTemperatura
CCDuración	 <p>CCDuración (seg)</p> <p>Duración en segundos para cambiar de estado. Se enviará a CondicionesCambio.vi a través de Tiempo.</p>	In/Out	Si	Si	CCDuración
CCVolumen	 <p>CCVolumen (uL)</p> <p>Volumen en ul que deben haber fluido por el Flow Unit para cambiar de estado. Se enviará a CondicionesCambio.vi a través de Volumen.</p>	In/Out	Si	Si	CCVolumen

Tabla 4-2. Resumen de controles e indicadores del ModoAutomatico.vi

4.5 Estructura de ficheros y decisiones de diseño

Apartados anteriores ya han introducido la existencia de varios VI, instrumentos virtuales, en el programa. Estos surgen no solo de la necesidad de separar *FRONT PANEL* y *ModoAutomatico.vi*, sino también como un intento de combatir la tendencia que presenta el uso de LabVIEW a propiciar una programación desordenada en un único Block Diagram. El modo de funcionamiento de LabVIEW no hace trivial la descomposición de programas complejos en bloques pequeños, sin embargo, un conocimiento un poco más avanzado del entorno permite convertir un fichero VI bien en un Front Panel que interactue con el usuario o bien en una función completamente personalizada que pueda ser utilizada en cualquier otro VI del proyecto.

La estructura de ficheros completa queda representada en la *Figura 4-15* y consta de dos Front Panels principales, *FRONT PANEL.vi* y *ModoAutomatico.vi*, un fichero para la definición de las variables globales y una colección de funciones:

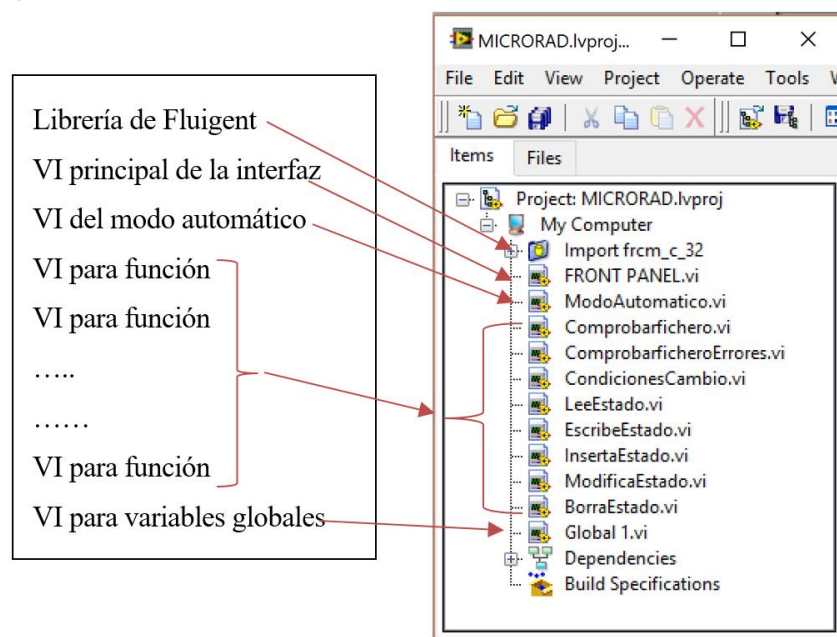


Figura 4-15. Estructura de ficheros

El presente proyecto utiliza una estructura común para cada bloque dentro de los Block Diagrams. Esta consta de una estructura tipo Case para iniciarlo y una estructura secuencial paralela o apilada con bucles While o For para definir su funcionalidad.

4.5.1 FRONT PANEL.vi

Su Block Diagram consta de 6 bloques encargados de tareas específicas como se detalla a continuación. La funcionalidad completa de cada bloque queda comentada dentro del código que se puede consultar en el *Anexo D*.

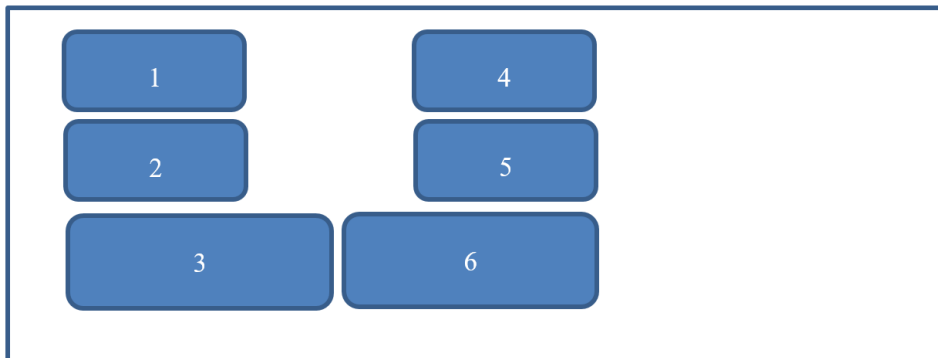


Figura 4-16. Estructura del Block Diagram de FRONT PANEL

- 1 y 2: se encargan de refrescar los valores de los indicadores del Front Panel principal. La única diferencia entre uno y otro es que en 2 se incluye un tiempo de 10ms en los que el valor se mantiene constante destinado a hacer más cómoda la visualización de gráficas tal y cómo aconseja NI.
- 3: gestiona la comunicación con el FRCM y con FRP por medio de dos líneas de ejecución paralelas.
- 4: se encarga de la comunicación con el Arduino. Dado que la asignación de pines del Arduino solo se debe realizar una vez, las funciones dedicadas a tal tarea se encuentran directamente fuera de este bloque y conectadas a él.
- 5: es el bloque que gestiona el PID para llevar la temperatura leída por el sensor hasta la consigna de temperatura seleccionada por el usuario.
- 6: tiene dos modos de funcionamiento independientes que son iniciados en función del valor de Manual/Automático. En este caso no se hace uso de la función leeEstado.vi para poder realizar cambios sobre las variables locales directamente lo cual ha sido relevante por ejemplo para poder mostrar el progreso del tiempo de ejecución de un estado de manera cómoda para el usuario.

4.5.2 ModoAutomatico.vi

Su Front Panel se abre al pulsar en *Configurar Modo Automático* y está aislado de la comunicación con el prototipo. Dicho Front Panel, *Figura 4-14*, ofrece una visión bastante clara de cómo estará estructurado el Block Diagram ya que los 4 botones principales que iniciarán un proceso son naturalmente *Mostrar*, *Selecciona*, *Insertar* y *Eliminar* respectivamente.

Cada uno de estos bloques se ejecuta de manera similar y están estructurados para que sea fácilmente localizable qué secciones hay y de que se encarga cada una. Los bloques están ordenados de la siguiente forma

- 1) Comprobar que el fichero es correcto → ComprobarFichero.vi
- 2) Imprimir errores si los hubiese o ejecutar la función principal de cada bloque.
- 3) Actualizar el fichero y refrescar el Front Panel.

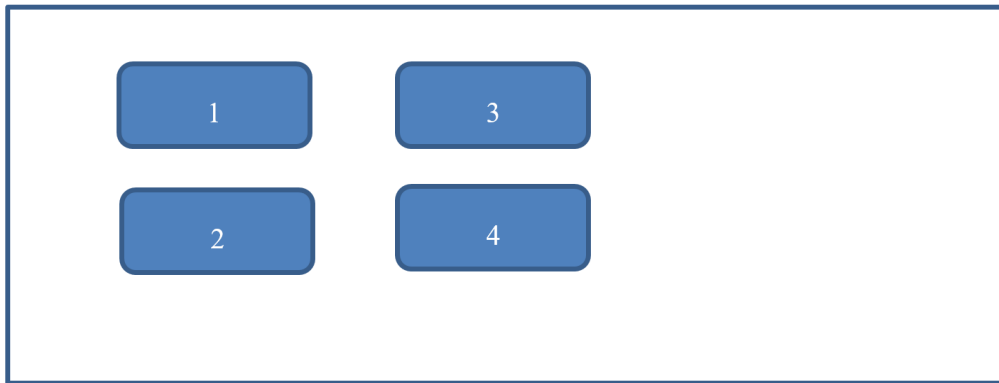


Figura 4-17. Estructura del Block Diagram de ModoAutomatico.vi

- 1: corresponde a *Mostrar* y su funcionalidad principal se encuentra en *LeeEstado.vi*.
- 2: aglutina las opciones del desplegable que se abre al pulsar en *Selecciona* ya que su funcionamiento interno es muy parecido con la salvedad de la manera en la que se debe abrir el fichero para cada caso. De esta manera todos se gestionan desde una única función en *ModificaEstado.vi*.
- 3: este bloque corresponde a *Insertar* y ha supuesto incluir una casuística más extensa que en el resto de los casos. Tal y como queda demostrado en la sección 4.6 el programa detecta todos los casos que pudiesen generar un comportamiento desconocido e imprime un mensaje de error. En este caso, lo más relevante es el valor del control *Estado*, enviado a *InsertaEstado.vi* en el momento en el que se pulsa *Insertar*, pues, aunque se espera que el usuario quiera elaborar la secuencia de manera ordenada, también se debe tener en cuenta la necesidad de incluir nuevos estados intermedios que no se hubiesen creído importantes o repetir estados mas adelante de la secuencia etc
- 4: explica cómo funciona *Eliminar* para borrar un estado de la secuencia. Dicho estado puede encontrarse en cualquier posición de la secuencia por lo que, complementariamente a cómo ocurre en el caso de *Insertar*, el estado eliminado implicará el desplazamiento del resto de la secuencia.

4.5.3 Funciones

Además de una breve descripción de cada función resulta imprescindible saber cómo crear o modificar las entradas y salidas es por ello por lo que también se va a presentar un resumen de tal proceso.

- *ComprobarFichero.vi*: Al inicio de cada bloque de *ModoAutomatico.vi*. Recibe la ruta al fichero y el valor del control *Estado* y devuelve un string con la secuencia completa, un entero con el número de estados del fichero y un entero para usarse como bandera de errores.
 - Si el fichero es correcto -->0
 - Si está vacío el fichero-->1
 - Si el path no está bien-->2
 - Si se quiere acceder a un estado fuera de rango-->3
- *ComprobarFicheroErrores.vi*: siempre se encuentra a continuación de *ComprobarFichero.vi* si la bandera es distinta de cero. Todos los mensajes disponibles se encuentran en la sección 4.6.
- *CondicionesCambio.vi*: Se inicia automáticamente al comienzo de cada nuevo estado y devuelve true cuando la primera condición de cambio se ha cumplido.
- *LeeEstado.vi*: Realiza la función principal de *Mostrar*, recibe la posición del estado que se quiere leer [1, *NumEstados*] y lo devuelve con las variables globales correspondientes.

- *EscribeEstado.vi*: Se utiliza en *Insertar*, *Guardar* y *Guardar Copia* para obtener un string de un único estado con los valores que aparezcan en el Front Panel de *ModoAutomatico.vi*.
- *InsertaEstado.vi*: Recibe la ruta al fichero, el número de estados que tiene y la posición en la que se desea insertar el nuevo estado y devuelve la ruta del fichero actualizado. Basicamente, recorre el string de la secuencia e inserta el estado creado con *EscribeEstado.vi* en la posición adecuada.
- *ModificaEstado.vi*: Igual que *InsertaEstado.vi* pero en esta ocasión se asegura de reemplazar el estado que estaba en la posición marcada por el control *Estado* por aquello que esté en el front panel de *ModoAutomatico.vi*.
- *BorraEstado.vi*: Parecido a los anteriores, pero al localizar el estado marcado por el control *Estado* continúa recorriendo el string sin haberlo guardado en el fichero actualizado.

Los Front Panels de los VI para funciones necesitan definir las entradas y las salidas, controles e indicadores respectivamente, como se muestra a continuación. Para asociar los controles a los pines de entrada de la función y los indicadores a los pines de salida es necesario redefinir los terminales con el botón derecho sobre la figura de la parte superior derecha de uno de los Front panels usando *Add terminal* y *Remove terminal*. En origen tiene



el aspecto y debe llegar a conseguirse algo parecido el caso siguiente.

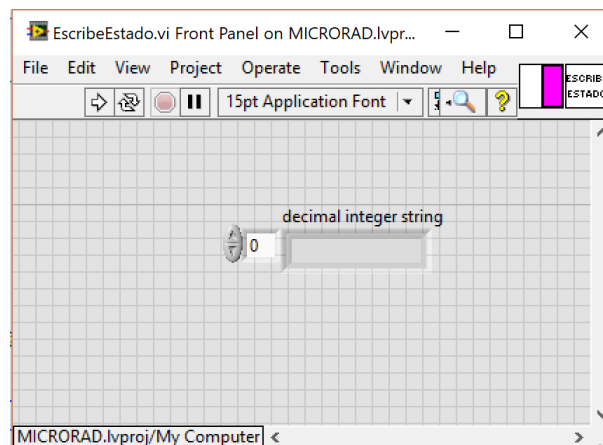


Figura 4-18. Ejemplo de la generación de una función en LabVIEW™

Finalmente, para utilizar el VI como una función en otro fichero solo será necesaria una estructura como la de la *Figura 4-19* que corresponde al uso de *EscribeEstado.vi* en el interior de *InsertaEstado.vi* cuando *Estado* iguala a *N* el cual recibe valor de *NumEstados*.

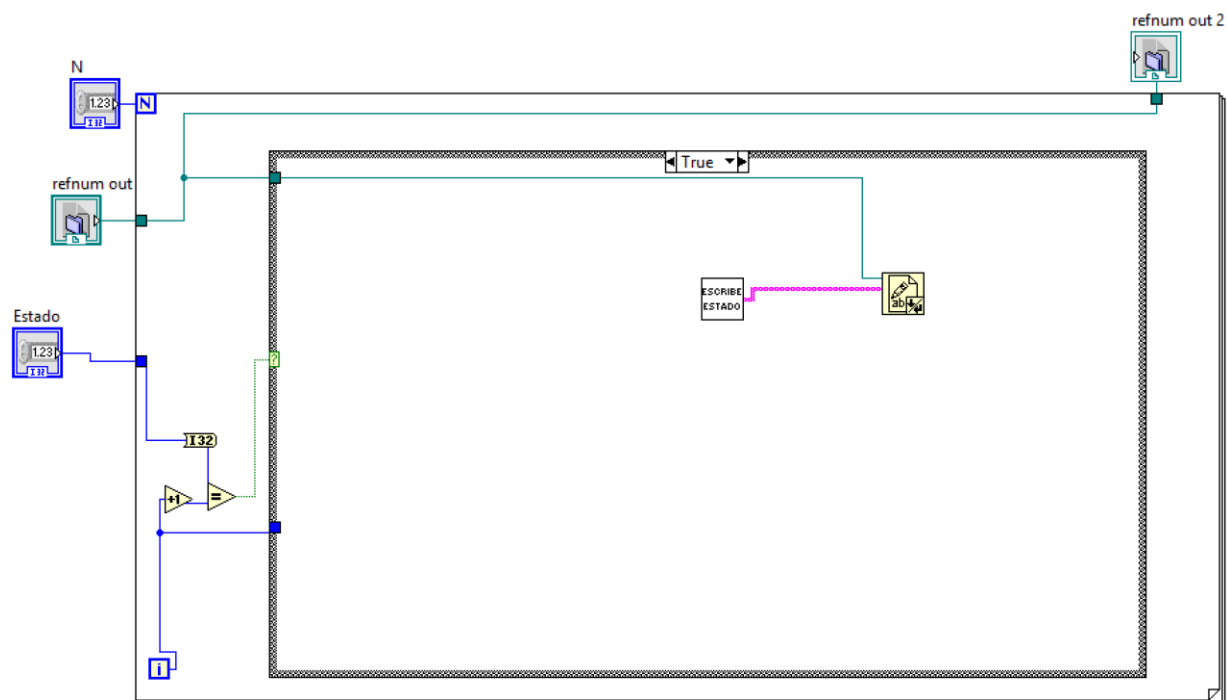
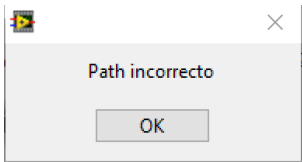
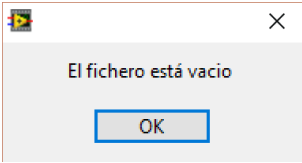
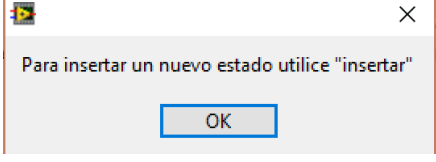


Figura 4-19. Uso de funciones en LabVIEW™

4.6 Pruebas parciales

El funcionamiento del fichero *ModoAutomatico.vi* ha sido comprobado con numerosas pruebas en las que se ha comprobado tanto que las funcionalidades son las esperadas como que cualquier casuística incorrecta genera los mensajes de error correspondientes.

Descripción de la prueba	Mensaje generado
Pulsar en las opciones <i>Mostrar</i> , <i>Insertar</i> , <i>Eliminar</i> , <i>Guardar</i> o <i>Guardar Copia</i> , sin haber abierto un fichero	
Pulsar en las opciones <i>Mostrar</i> o <i>Eliminar</i> , con un fichero vacío	
Pulsar en las opciones <i>Guardar</i> o <i>Guardar copia</i> con un fichero vacío	

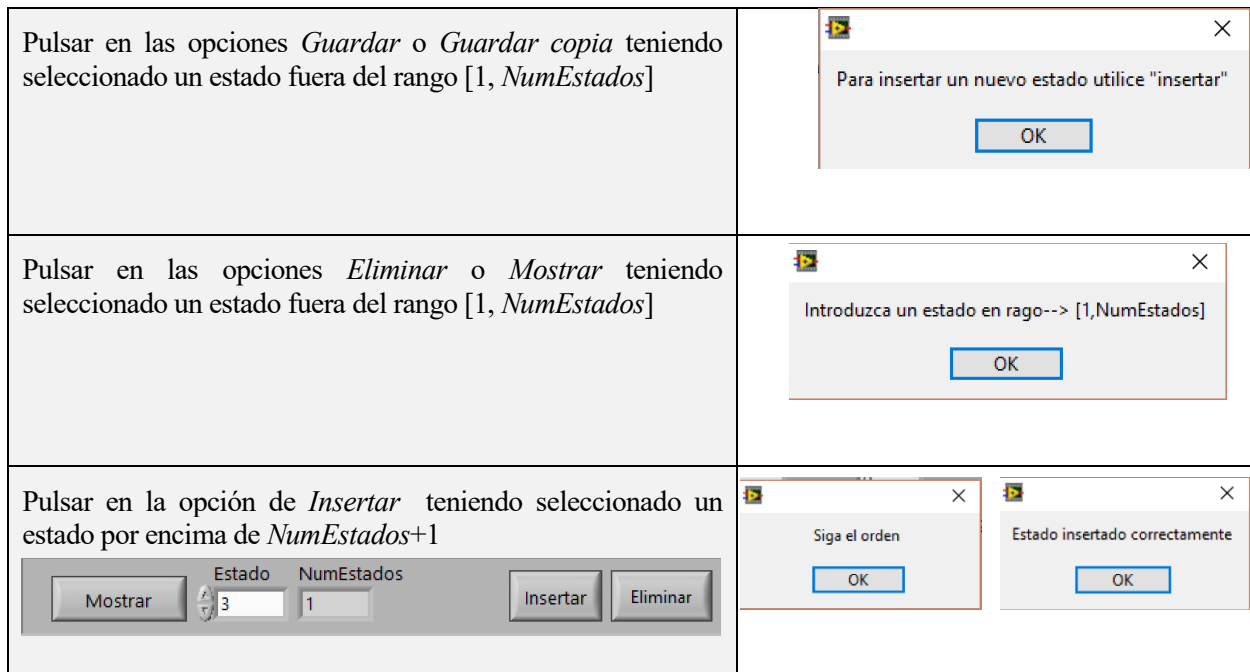


Figura 4-20. Colección de pruebas parciales sobre *ModoAutomatico.vi*

Con carácter previo a estas pruebas también se ha realizado la depuración [12] mediante ejecución paso a paso y con puntos de parada mediante las opciones correspondientes de la barra de herramientas principal del entorno LabVIEW definidas a continuación.

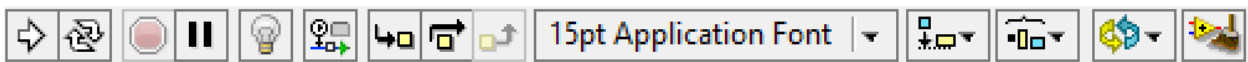




Figura 4-21. Barra de herramientas para la depuración de programas en LabVIEW™

- 1) Botón *Run*: LabVIEW compila el VI e inicia su ejecución. Puede ejecutar un VI si el botón *Run* aparece como una flecha en blanco, como se muestra la *Figura 4-21*. En caso de aparecer roto  indica que el programa contiene errores tanto por cableado como por incoherencia de tipos en conexiones. Al hacer click en el icono aparece la ventana *Error list* mostrando todos los errores y advertencias.
- 2) El botón *Run Continuously* permite ejecutar el VI hasta que el usuario detenga la ejecución. También puede hacer clic en el botón otra vez para deshabilitar la ejecución continua.
- 3) El botón *Abort Execution*. Detiene el VI inmediatamente.
- 4) El botón *Highlight Execution* muestra la ejecución del diagrama de bloques de manera animada.

Usar la ejecución resaltada junto con la herramienta *single-stepping* ( +



) permite ver cómo los valores de los datos se mueven de nodo a nodo a través del VI de manera muy clara. Haciendo click en el botón otra vez deshabilitamos la ejecución animada.

5 PRUEBAS FINALES

En esta sección se van a presentar una serie de casos prácticos sobre las posibilidades del programa completo desarrollado. En los primeros 3 apartados se emplea *ModoAutomatico.vi* para crear secuencias de 1, 2 y 3 estados respectivamente y los 3 apartados posteriores muestran el comportamiento de FRONT PANEL al usar las secuencias anteriores.

5.1 Secuencia de un estado con CCDuración sin Fluigent

Se trata del ejemplo más sencillo, un ejemplo podría basarse en mantener todos los solenoides abiertos durante 10 segundos, para ello bastará con replicar el siguiente escenario. Al ver el contenido del fichero del fichero generado en la parte derecha de la *Figura 5-1* comprobamos que la estructura se corresponde al diseño que se buscaba.

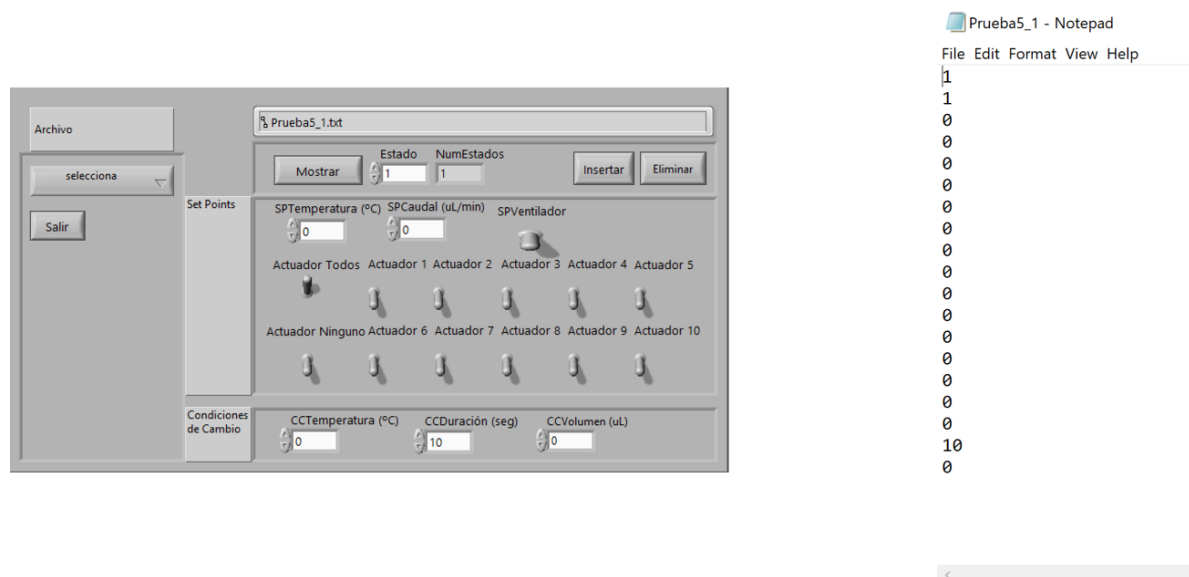


Figura 5-1. Estado 1 de la secuencia correspondiente a Prueba5_1.txt

5.2 Secuencia de dos estado con CCDuración y CCTemperatura sin Fluigent

En este caso supondremos que en el primer estado se desea mantener abierto el solenoide 1 durante 10 segundos y a continuación se desea cerrarlo hasta que se alcancen los 40°C en la cámara.

5.4 Resultados de usar Prueba5_1.txt.

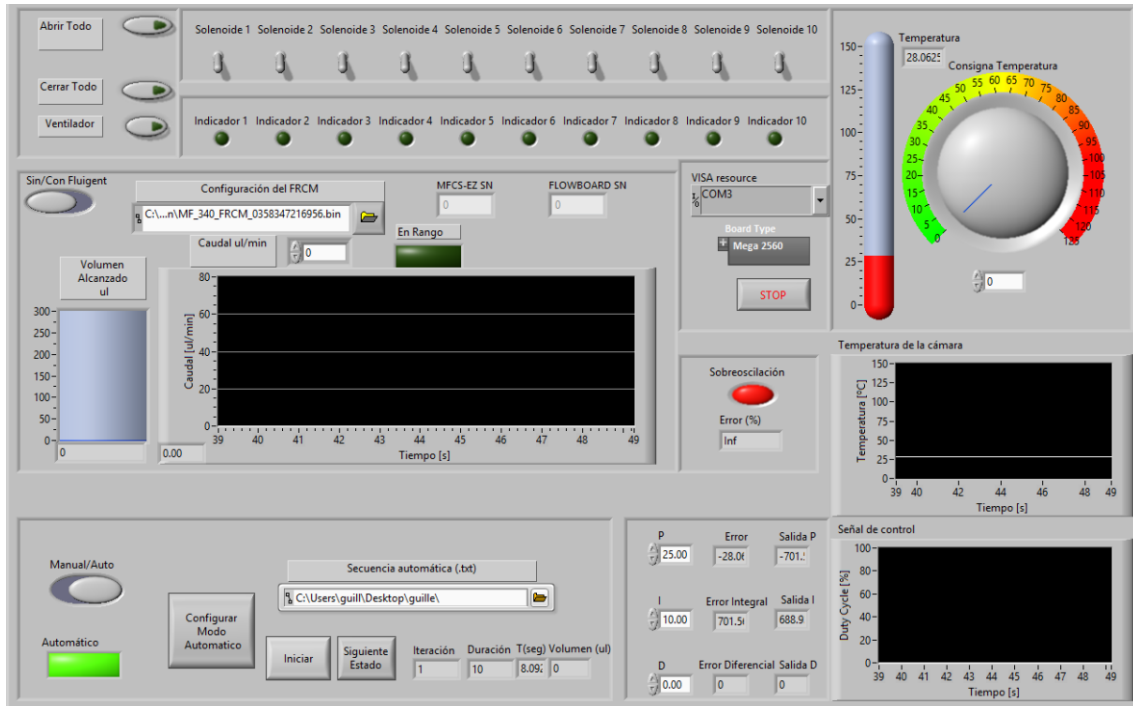


Figura 5-4. Resultados de Prueba5_1.txt

Aquí podemos ver cómo todos los solenoides están abiertos gracias a los indicadores correspondientes y además el indicador inferior central de $T(seg)$ muestra el tiempo transcurrido antes de realizar la captura de pantalla.

5.5 Resultados de usar Prueba5_2.txt.

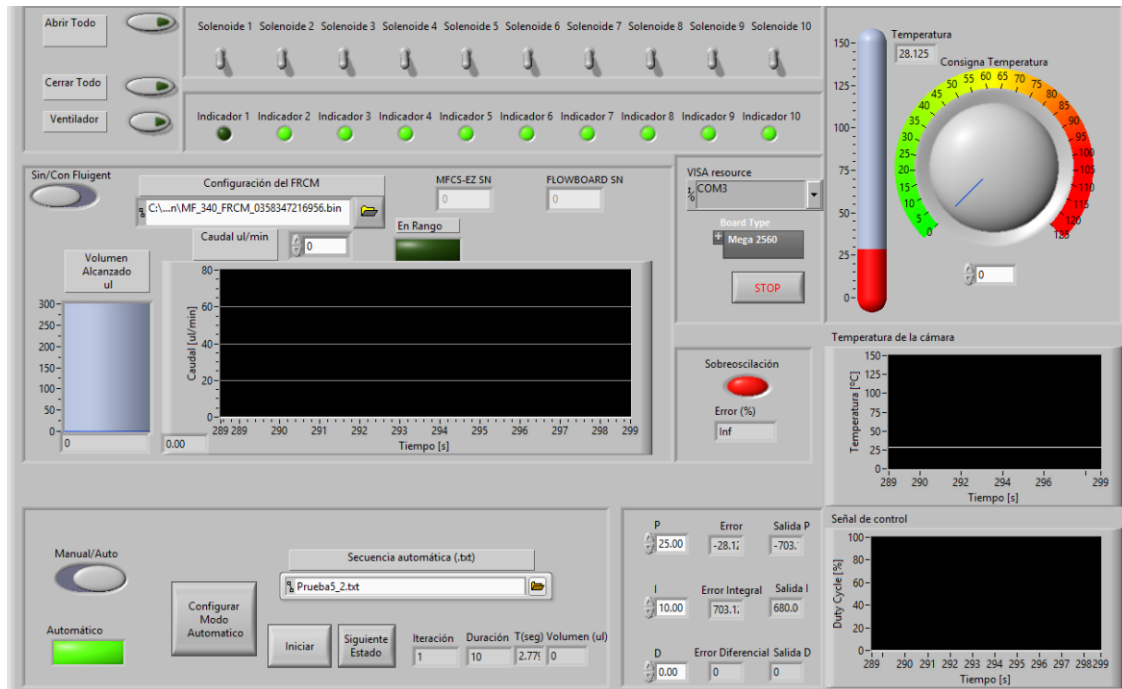


Figura 5-5. Resultados del estado 1 de la secuencia Prueba5_2.txt

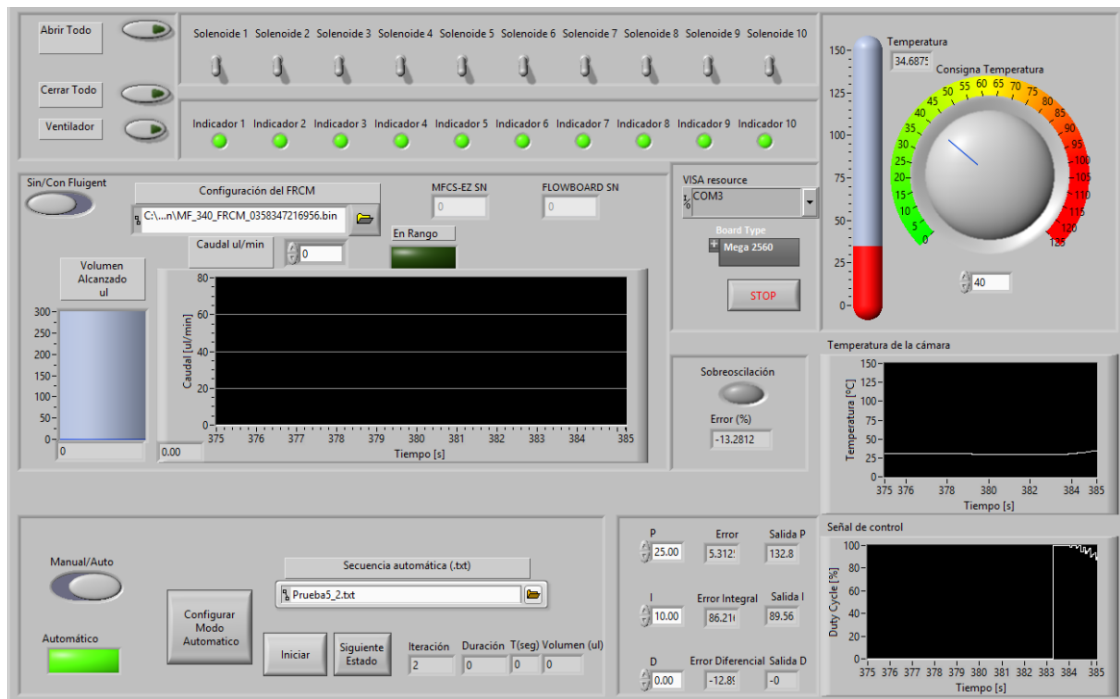


Figura 5-6. Resultados del estado 2 de la secuencia Prueba5_2.txt

En esta ocasión debemos contextualizar la prueba ya que, aunque se observa que el segundo estado ha aumentado la temperatura medida en la cámara de 28°C a más de 34°C, lo cierto es que se alcanzan los 40 °C antes de 2 segundos y no da lugar a apreciar la curva de actuación.

5.6 Resultados de usar Prueba5_3.txt.

En este último ejemplo vamos a poder comprobar el funcionamiento de todos los bloques en cada estado.

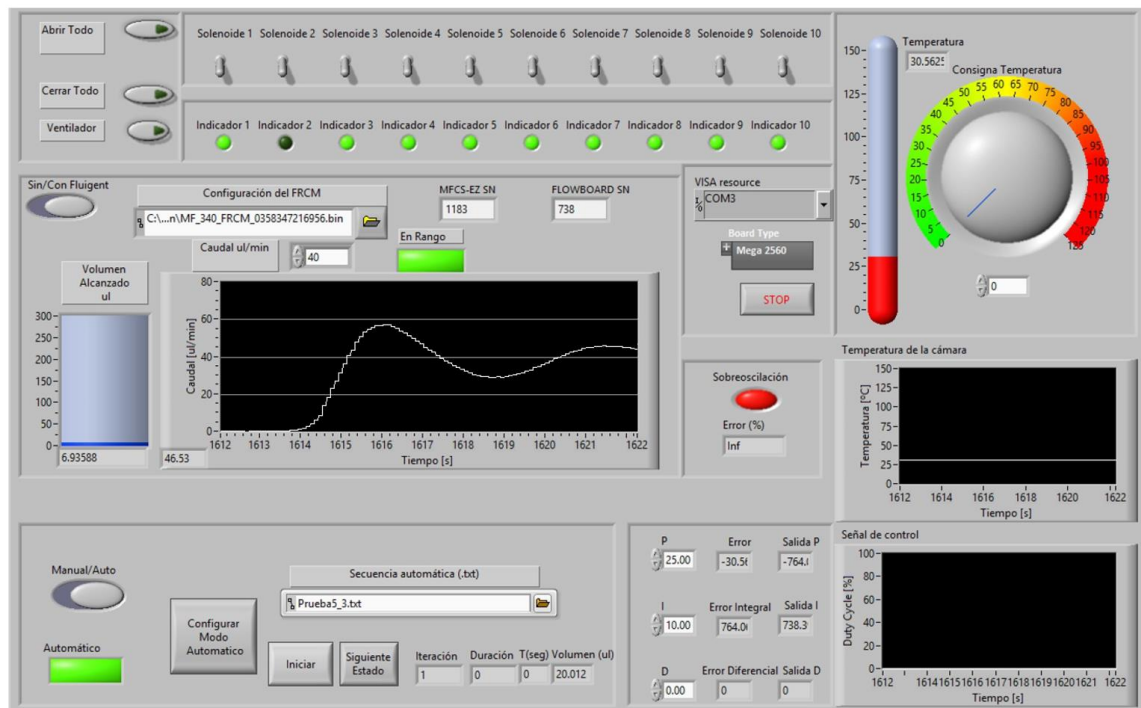


Figura 5-7. Resultados del estado 1 de la secuencia Prueba5_3.txt

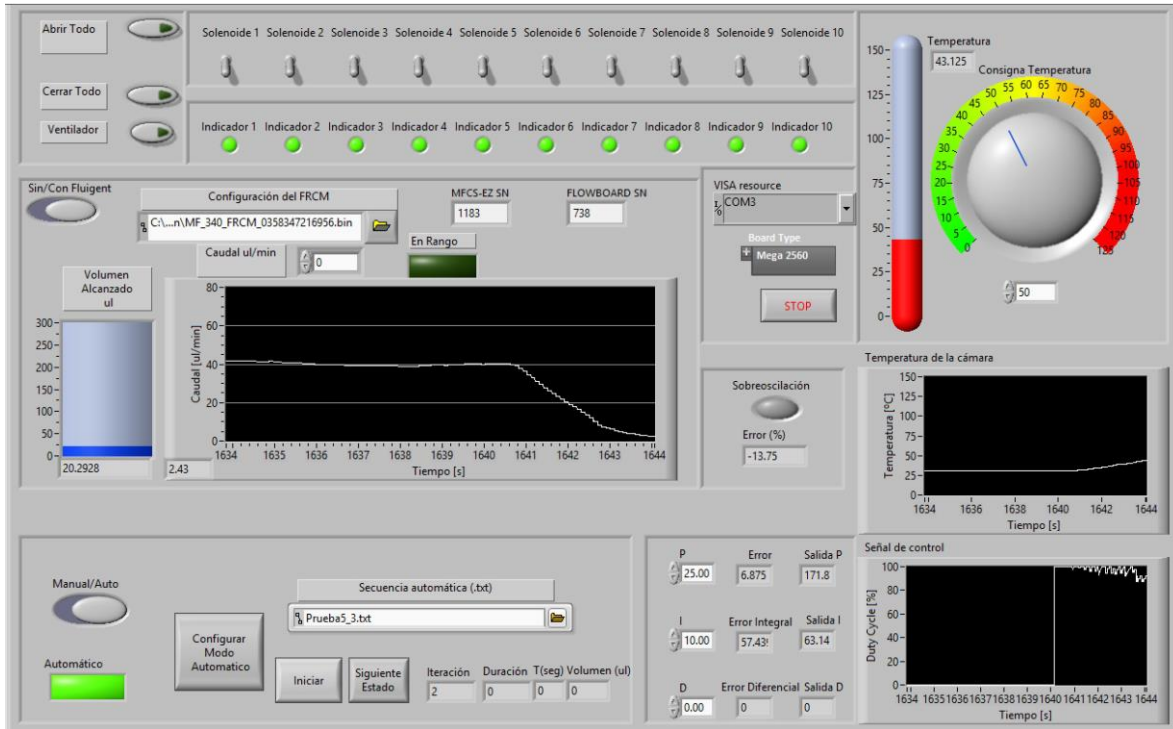


Figura 5-8. Resultado del estado 2 de la secuencia Prueba5_3.txt

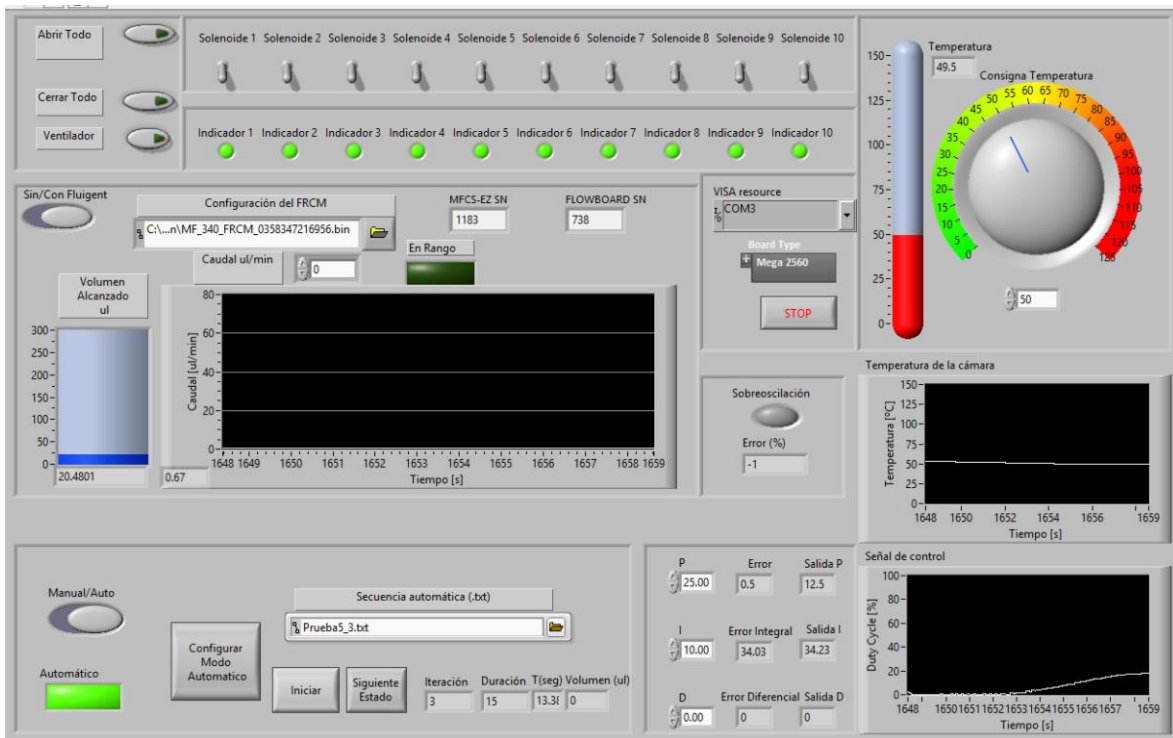


Figura 5-9. Resultado del estado 3 de la secuencia Prueba5_3.txt

Es con estas últimas tres figuras dónde se observa que todos los bloques interactúan entre ellos como se esperaba y que tenemos disponible toda la información que necesitábamos. Particularmente es la *Figura 5-8* demuestra que en las transiciones tanto el FRCM como el PID del calentador actúan rápidamente para alcanzar las nuevas demandas. También se observa que el control del caudal introduce una cierta oscilación al principio pero se estabiliza rápidamente y además se tiene en cuenta a la hora de calcular el volumen alcanzado.

6 CONCLUSIONES Y DESARROLLO FUTURO

De manera global podemos garantizar que todos los objetivos marcados al comienzo de este documento han sido alcanzados y además se han añadido varias funcionalidades adicionales para un desarrollo de aplicaciones más fluido.

Incluso habiendo partido sin conocimiento previo sobre el entorno LabVIEW, este ha demostrado tener una curva de aprendizaje muy rápida. Así pues, continuar trabajando sobre esta plataforma permitiría incluir nuevas fases al proyecto de manera casi inmediata como por ejemplo si se quisieran crear bases de datos con los resultados de las distintas pruebas llevadas a cabo.

La primera etapa de la experiencia trabajando con LabVIEW ha sido muy positiva ya que solucionar errores relacionados con Arduino cuenta con una extensa gama de foros y webs en las que se pueden encontrar soluciones rápidamente. En la segunda etapa por el contrario, integrar Fluigent en LabVIEW no consta de foros ni trabajos de código libre disponibles lo que, sumado a la complejidad de las funciones, ha acabado suponiendo un reto importante. Pese a esto último la empresa ha prestado atención rápida y completa e incluso se ha ofrecido a enviar paquetes actualizados disponibles para la versión de LabVIEW 2013 que es con la que se ha desarrollado este TFG.

Con respecto a la interfaz de usuario se ha trabajado para que sea lo más intuitiva posible y para que los mensajes de error sean capaces de orientar al usuario y permitirle saber qué está haciendo en cada momento. En contrapartida, la instalación del SW necesario requiere de varios pasos previos a la utilización de la aplicación.

Cómo se ha mencionado a lo largo de este documento, MICRORAD es un proyecto vivo que sigue adaptando su sistema a los resultados que obtiene durante la investigación. Es por ello por lo que ya se han planteado algunas modificaciones al HW para una nueva versión del prototipo en la que, por ejemplo, se dispongan de los 10 solenoides capaces de gestionar el SW existente ahora, se modifique la manera en la que se calienta el microreactor llevándolo a la parte superior de la cámara en lugar de la inferior o se incluya un conducto sobre el que permitir hacer el vacío dentro de la cámara.

Previendo futuras aplicaciones, se ha dejado preparado el SW para el uso de más de un Flow Unit con muy poca programación requerida y además se ha desarrollado un documento informando de qué pasos llevar a cabo si se desea añadir una nueva utilidad que requiera el uso de un nuevo pin del Arduino.

La continuación natural de este proyecto debería centrarse en aumentar las funcionalidades de ModoAutomático.vi. Por ejemplo sería interesante que la interfaz fuese capaz de introducir bucles de estados, modificar fácilmente la numeración de los estados en la secuencia o introducir mayor número de condiciones de cambio.

Por otra parte, las dificultades encontradas durante la integración de Fluigent en LabVIEW sugieren que el esfuerzo podría orientarse a trasladar dicha gestión a Matlab y después utilizar alguno de los las funciones existentes para invocar Matlab desde LabVIEW. Para ello habría que sustituir el bloque 3 de FRONT PANEL por una de las 3 posibilidades existentes; un ActiveX, un DLL o mediante Matlab Node. La primera de ellas funciona como un contenedor que enlaza programas y que es gestionado por Microsoft Technologies. De esta manera se puede insertar un proyecto completo de Matlab, mediante el comando comtool, que se ejecuta como una única función dentro del VI. La siguiente técnica sigue el mismo razonamiento anterior pero integra el proyecto como una librería compartida (dll) usando los recursos de LabVIEW. Por último, también se puede utilizar una función de LabVIEW que permite insertar un archivo de Matlab .m directamente en el VI.

Acorde con la información proporcionada por NI, el uso de ActiveX se ha demostrado más rápido que el resto de utilidades, aunque es la más compleja y no tiene soporte ni en Linux ni en Mac.

ANEXO A. MANUAL DE INSTALACION

Dada la naturaleza del sistema, el Front Panel principal hace uso de la paleta Arduino y de la paleta de Fluigent, cuyos procesos de instalación se describen en este anexo.

Para poder instalar estas paletas es necesario tener instalado primero el programa VI Package Manager desde el que se pueden localizar los archivos de extensión vip que permitan obtener cómo mínimo lo siguiente.

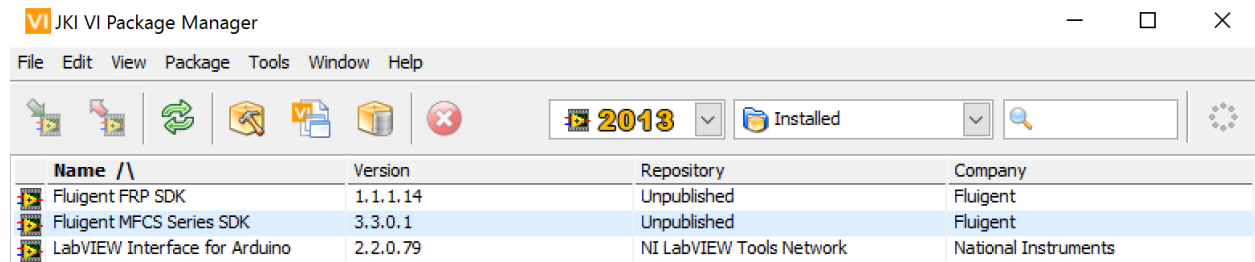


Figura A-1. Paquetes que se deben instalar en LabVIEW para ejecutar el programa

El uso de FRCM no tiene relevancia aquí ya que se encuentra importado directamente al proyecto en lugar de a la aplicación para permitir la modificación de sus archivos.

En primer lugar, es necesario comprobar que el enlace entre ambos programas funciona correctamente. Para ello lo más efectivo a probado ser el seleccionar la versión de LabVIEW que estamos utilizando (incluso si solamente existe una versión ha resultado imprescindible realizar este paso) en el VIPM. Para ello se debe pulsar sobre verificar dentro de la sección Tools/Options/LabVIEW cuya ventana corresponde con la *Figura A-2*.

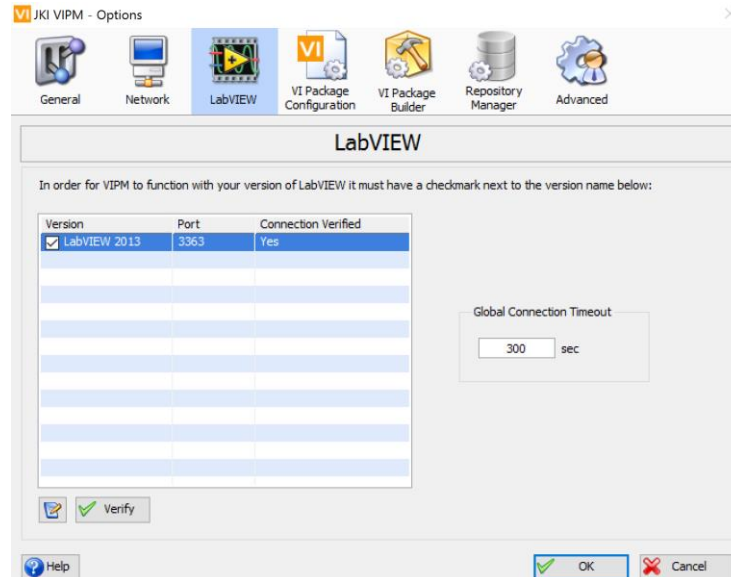


Figura A-2. Verificación de la conexión entre VIPM y LabVIEW

En caso de no tener éxito se recomienda reiniciar el ordenador y volver a probar. Si el resultado sigue siendo negativo o se obtienen mensajes como el de la *Figura A-2* también resulta efectivo desinstalar VI Package Manager e instalar la última versión disponible en la página oficial de National Instrument.

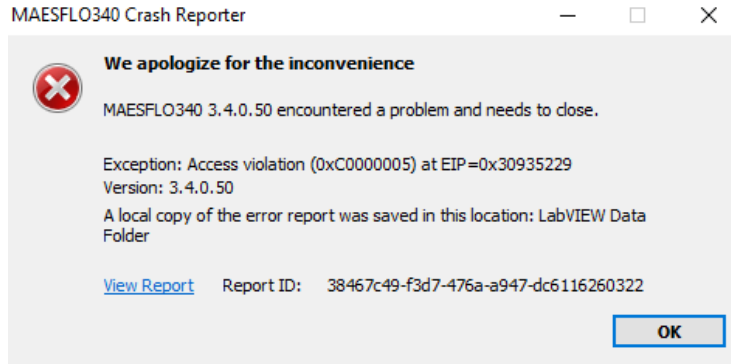


Figura A-3 Error al utilizar los paquetes de Fluigent si no hay enlace entre VIPM y LabVIEW

Una vez hayamos comprobado que VI Package Manager es capaz de lanzar LabVIEW correctamente debemos buscar mediante File/OpenPackage(s) los archivos:

- national_instruments_lib_LabVIEW_interface_for_arduino-1.2.0.21.vip ó superior que se debe haber descargado de la página oficial de National Instrument.
- C:\fluigent\Fluigent SDK\Fluigent MFCS Series SDK\MFCS Series Toolkit for LabVIEW\fluigentmfcs_series-3.3.0.1.vip
- C:\fluigent\Fluigent SDK\Fluigent FRP SDK\FRP Toolkit for LabVIEW\fluigentfrp-1.1.1.14.vip

El siguiente paso será modificar el VI del sistema que comprueba la numeración de los pines de Arduino tal y como se explicó en la sección 4.1.

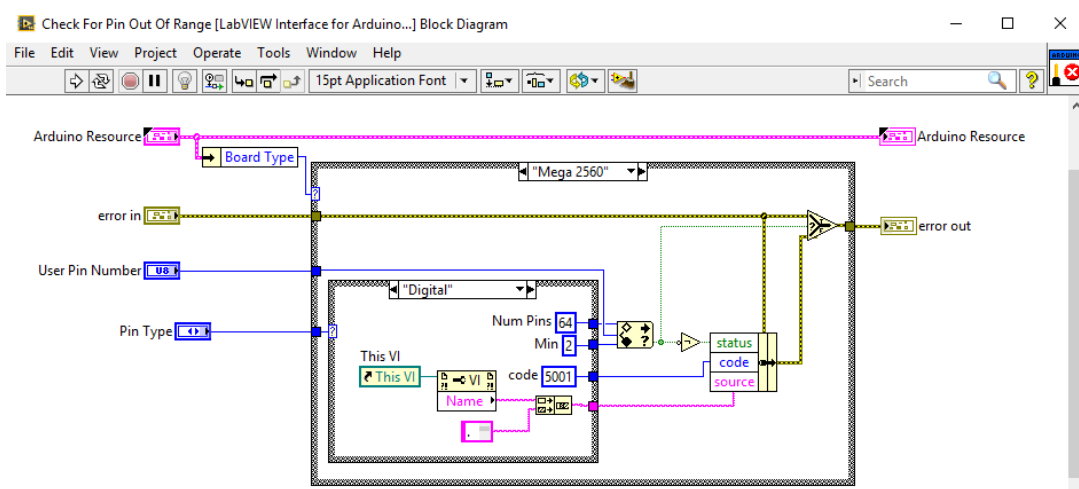


Figura A-4. Ajustes para usar los pines analógicos de la Arduino Mega 2560

Es también importante instalar los programas oficiales de Fluigent para el FRP y para el MFCS-EZ localizados cada uno en las carpetas Flow Rate Platform y MAESFLO respectivamente.

Por último, también es necesario preparar el Arduino para poder comunicarse con LabVIEW. Para ello habrá que cargar en la placa a través de IDE de Arduino descargable en la página web oficial de la compañía, el siguiente programa de librería que facilita National Instruments desde su página oficial:

- C:\Program Files\National Instruments\LabVIEW 2013\vi.lib\LabVIEW Interface for Arduino\Firmware\LIFA_Base\LIFA_Base.ino

ANEXO B. MANUAL DE USUARIO: PREPARACIÓN DEL ENTORNO

En este documento se va a presentar de nuevo el diagrama de conexiones entre los elementos del sistema y se se mostrarán los componentes ensamblados entre sí ordenados según corresponde al punto 1 o al 2 y finalmente de manera completa.

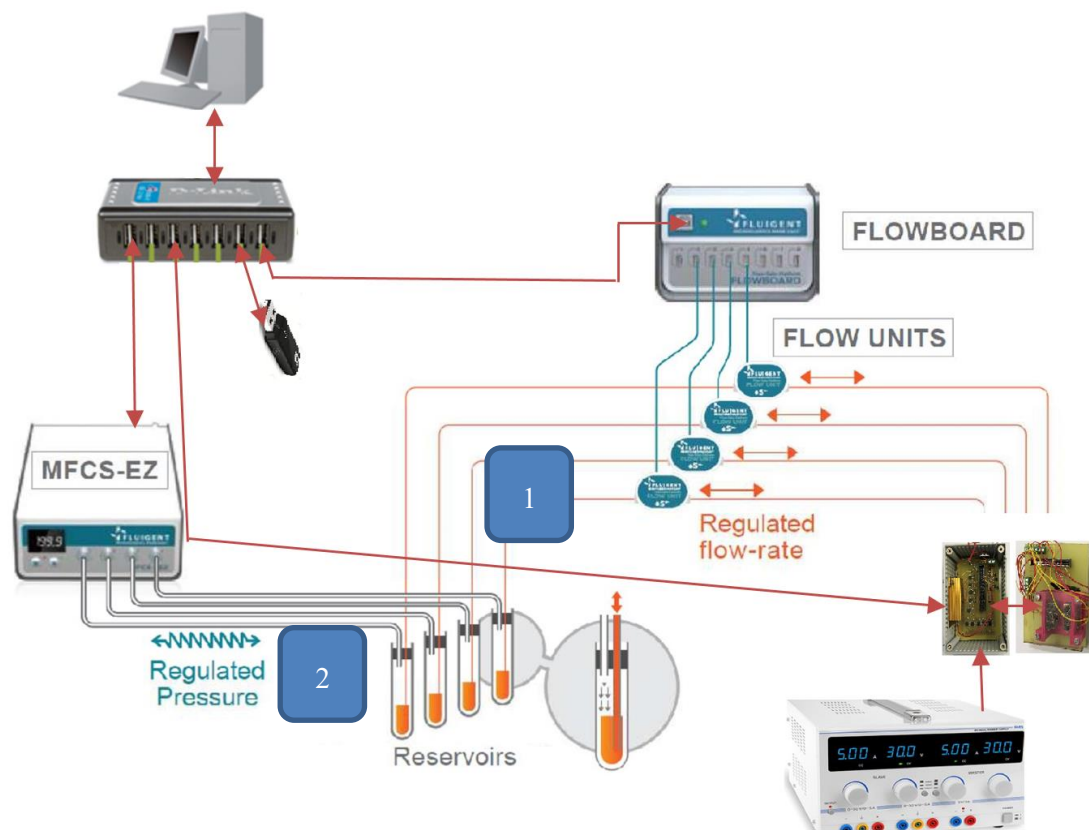

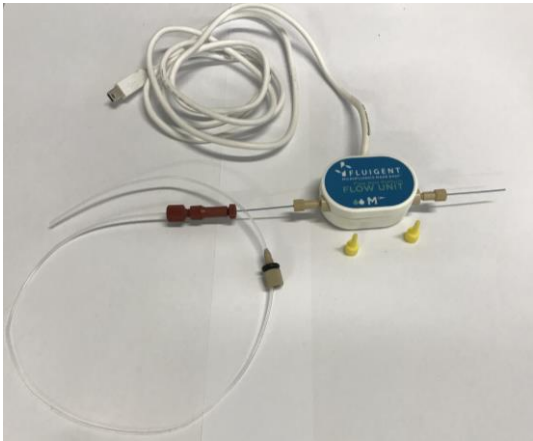



Figura B-1. Esquema de conexiones del sistema

Nombre del componente	Descripción	Identificación
Bolsa 3	Depósitos de líquido y soporte del mismo. Correspondiente al punto 1 de la <i>Figura B-1</i> .	
Bola 2	Adaptadores de tubos entre el Flow Unit y el depósito de la Bolsa 3. Correspondiente al punto 1 de la <i>Figura B-1</i> .	
Bolsa 1	Adaptadores de tubos para la salida de presión del MFCS-EZ. Correspondiente al punto 2 de la <i>Figura B-2</i> .	


Bolsa 2 y 3	Conexión entre los componentes de las bolsas 2 y 3.	
-------------	---	--

Tabla B-0-1. Resmuen de conexiones por bolsa de componentes

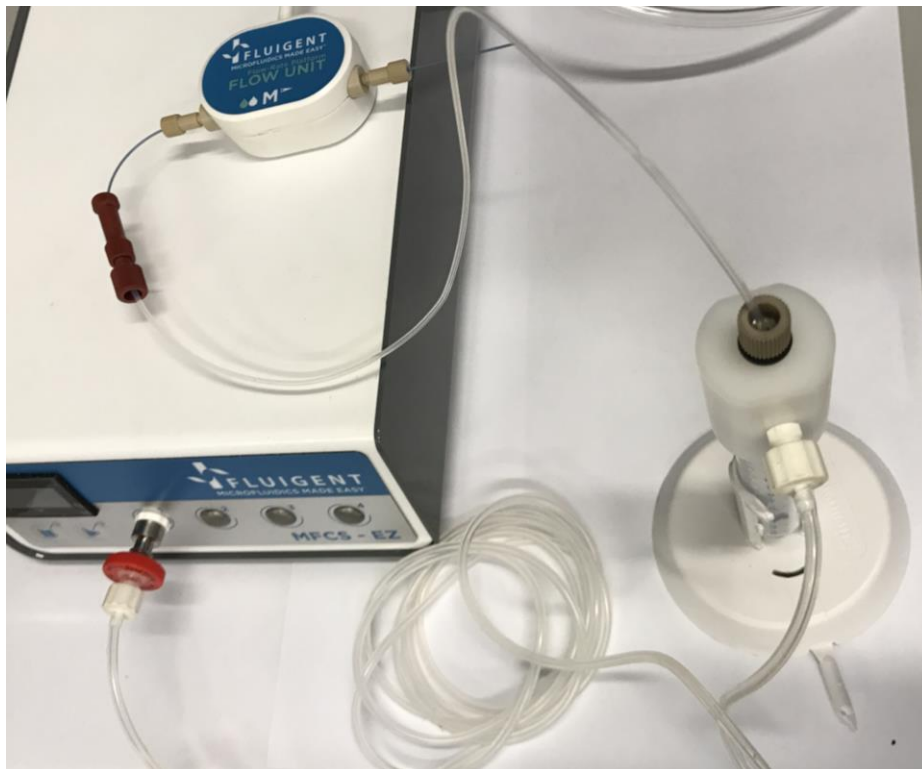
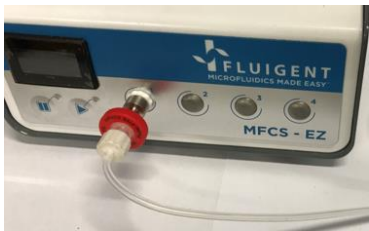
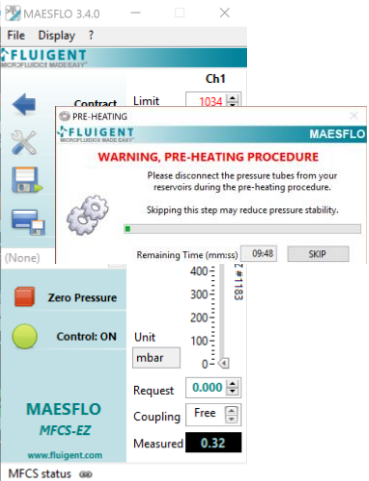
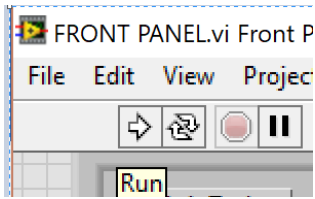


Figura B-2. Fotografía de las conexiones reales entre los componentes

ANEXO C. MANUAL DE USUARIO: REALIZACION DE UNA PRUEBA

La siguiente tabla debe entenderse como un protocolo de encendido y apagado del programa que tiene que seguirse secuencialmente para poder asegurar el comportamiento estable y la integridad de todos los componentes.

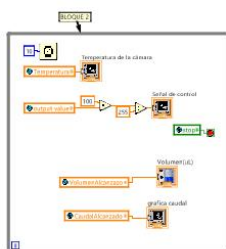
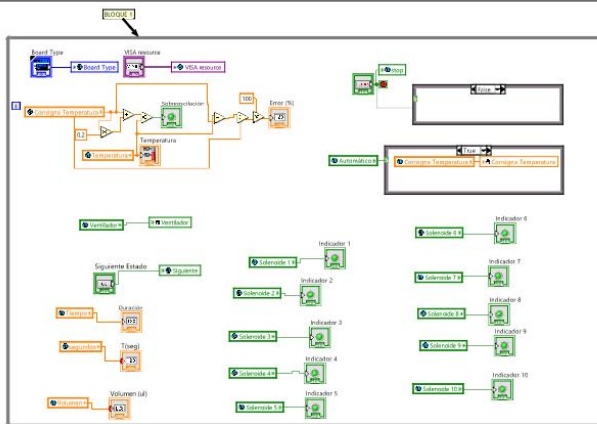
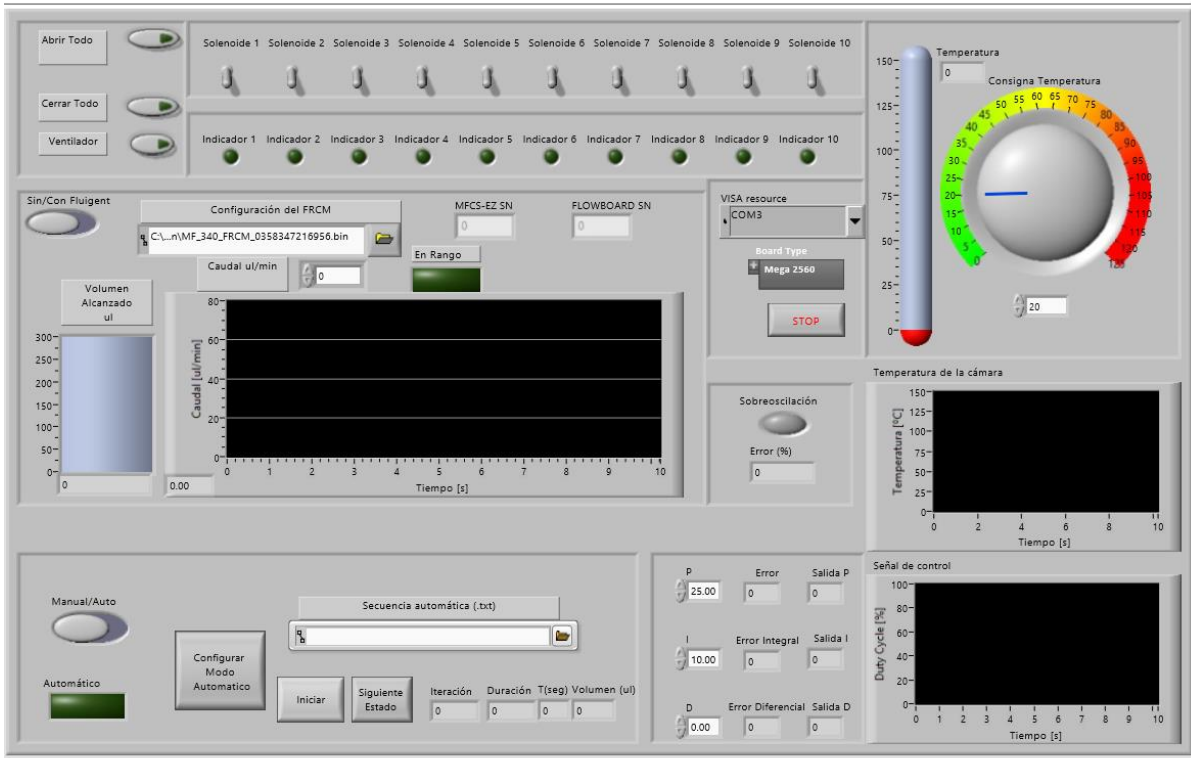
Orden	Descripción	Identificación
1	Desconectar el puerto de salida de presión del MFCS-EZ	
2	Iniciar el programa MAESFLO 3.4.0 o superior y esperar a que aparezca la ventana de la derecha	
3	Presionar el botón de play (▶) del MFCS-EZ y esperar a que terminen los 10 minutos de precalentamiento.	
4	Volver a conectar la salida de presión del MFCS-EZ	
5	Abrir el proyecto MICRORAD.lvproj	
6	Abrir FRONT PANEL.vi	
7	Iniciar el programa pulsando en RUN	
8	Para parar, si se estaba usando Fluigent, lo primero debe ser cerrar la comunicación con dichos módulos colocando	

	Sin/Confluente en su posición izquierda y después pulsar en el Control STOP del FRONT PANEL para finalizar el programa.	
9	Volver a 7 si se desea arrancar el sistema de nuevo.	

Tabla C-1. Protocolo de inicialización del programa

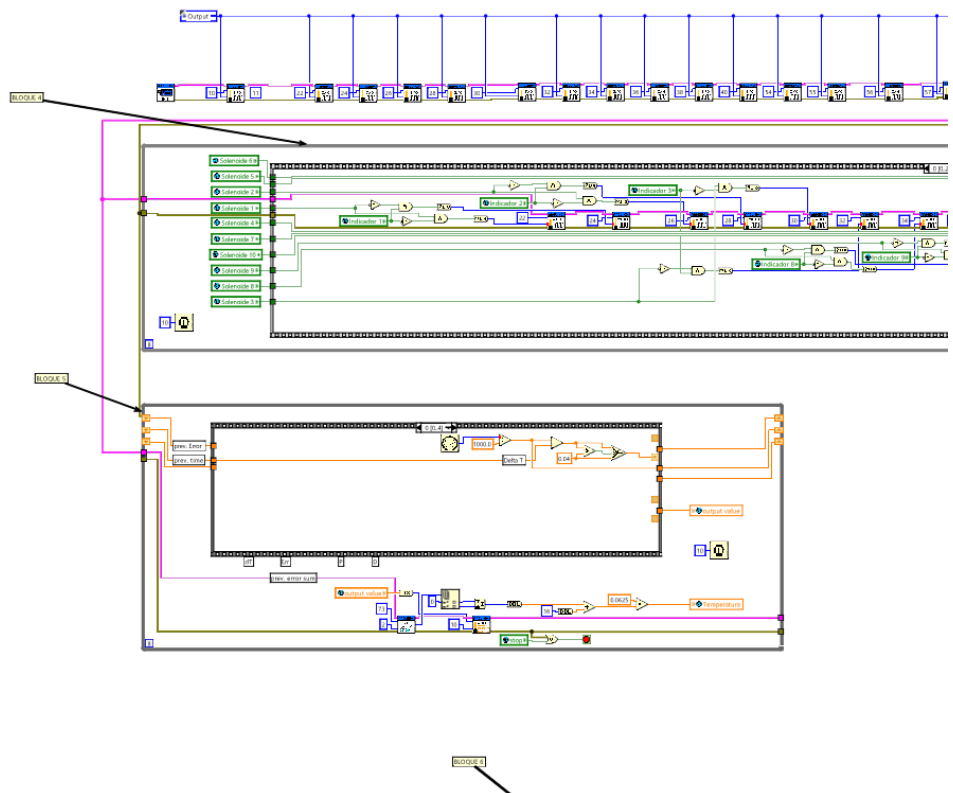
ANEXO D. CODIGO FUENTE

FRONT PANEL.vi
C:\TFG\Microrad\FRONT PANEL.vi
Last modified on 06-Sep-17 at 12:35 PM
Printed on 06-Sep-17 at 2:19 PM

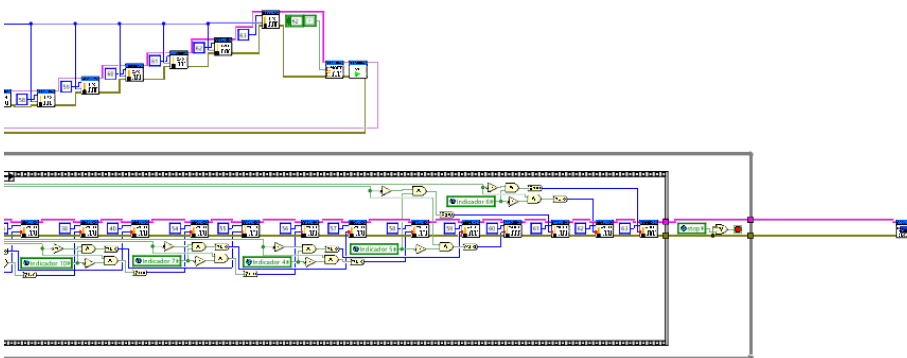




FRONT PANEL.vi
C:\TFG\Microrad\FRONT PANEL.vi
Last modified on 06-Sep-17 at 12:35 PM
Printed on 06-Sep-17 at 2:20 PM



FRONT PANEL.vi
C:\TFG\Microrad\FRONT PANEL.vi
Last modified on 06-Sep-17 at 12:35 PM
Printed on 06-Sep-17 at 2:20 PM



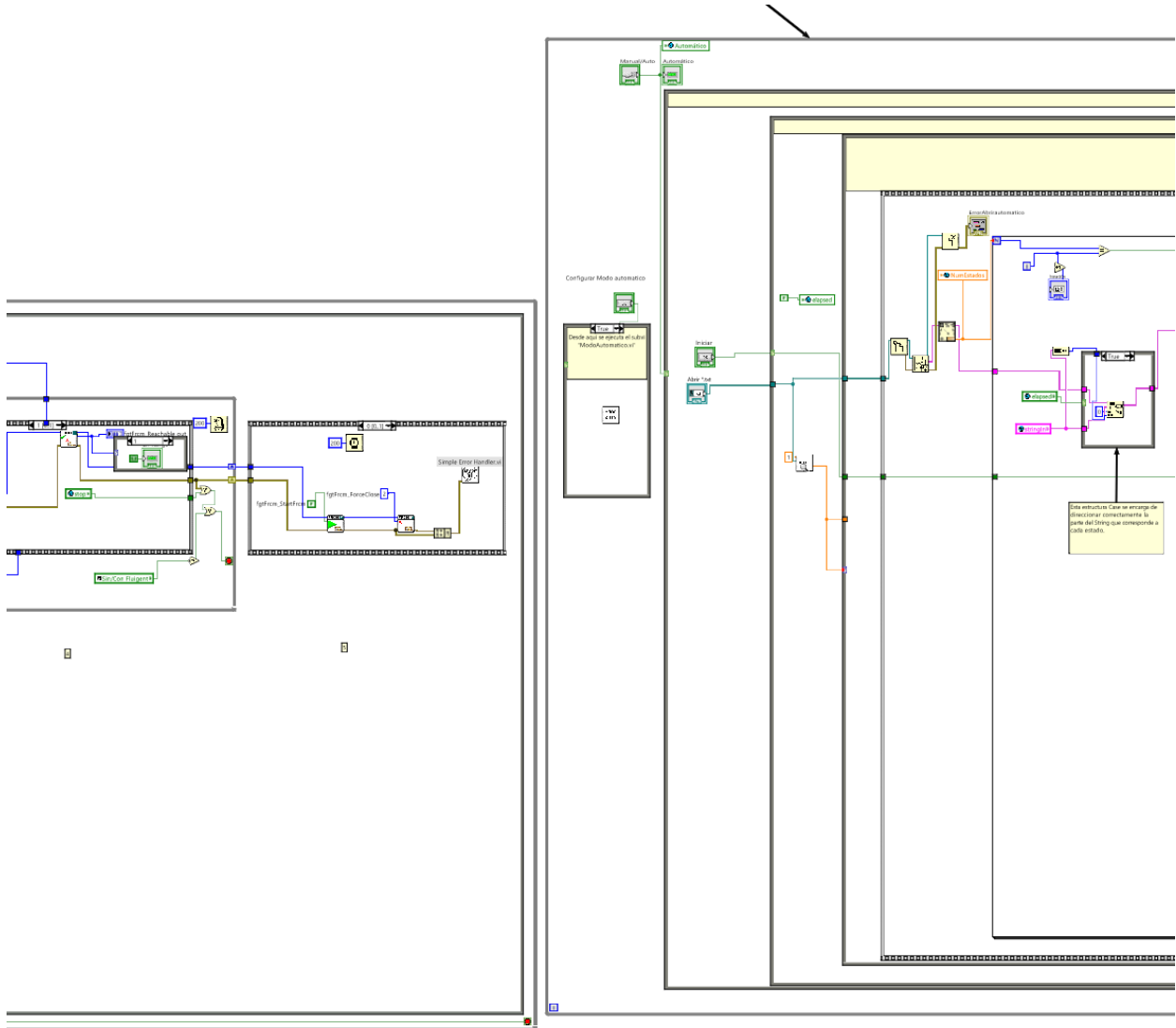


FRONT PANEL.vi

C:\TFG\Microrad\FRONT PANEL.vi

Last modified on 06-Sep-17 at 12:35 PM

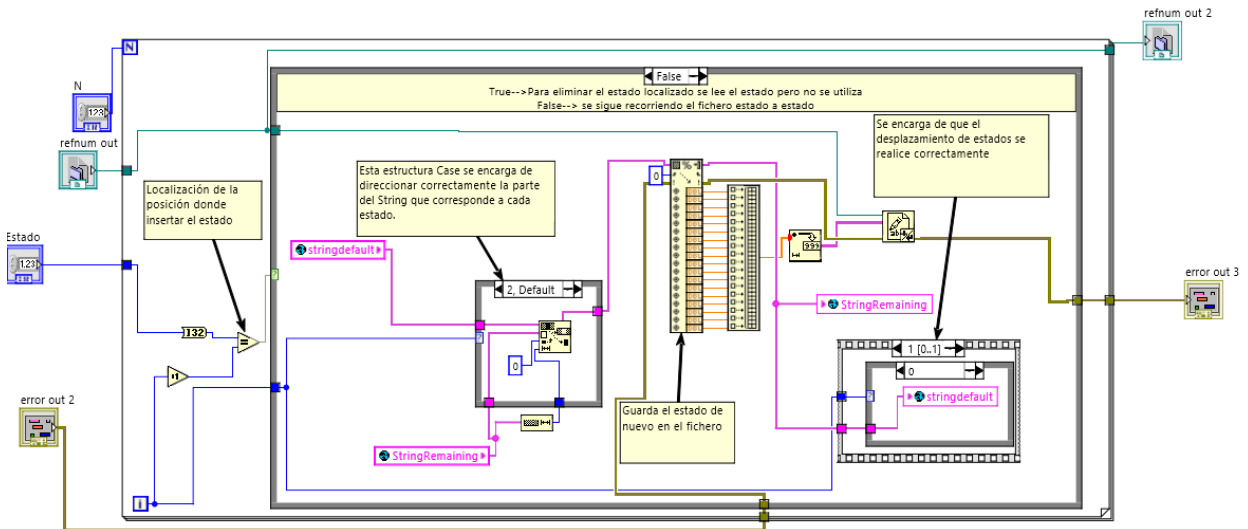
Printed on 06-Sep-17 at 2:20 PM



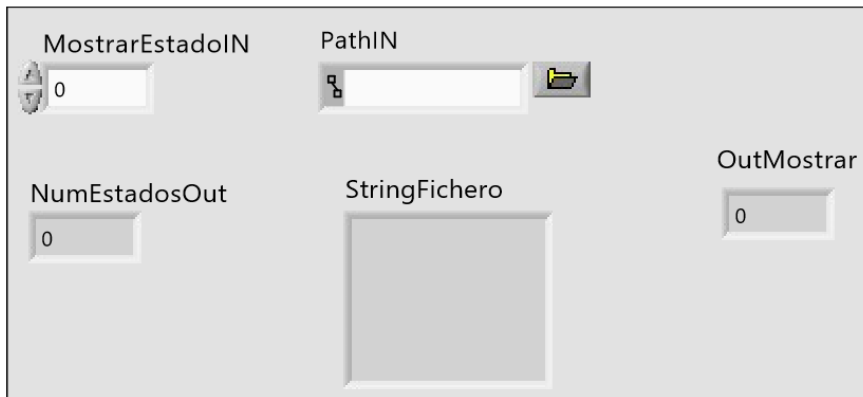
BorraEstado.vi
C:\TFG\Microrad\BorraEstado.vi
Last modified on 06-Sep-17 at 2:17 PM
Printed on 06-Sep-17 at 2:20 PM



refnum out
N
Estado
refnum out 2



Comprobarfichero.vi
 C:\TFG\Microrad\Comprobarfichero.vi
 Last modified on 06-Sep-17 at 12:37 PM
 Printed on 06-Sep-17 at 2:20 PM



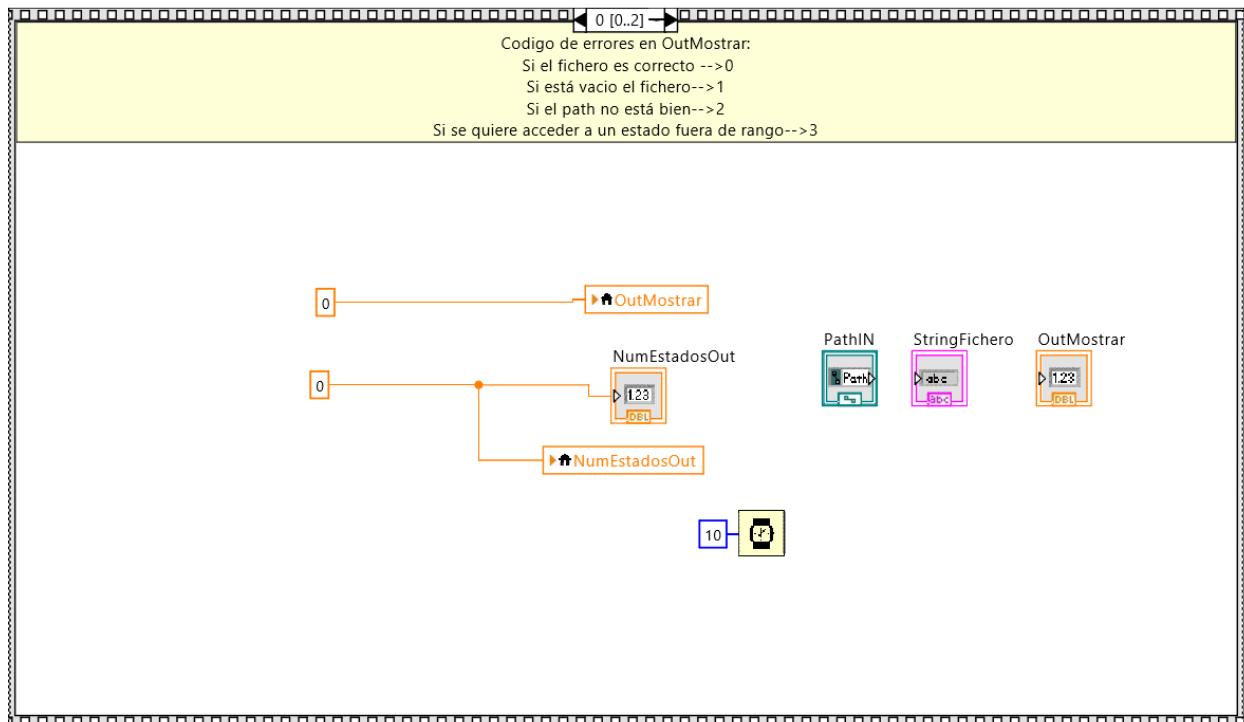
MostrarEstadoIN

PathIN

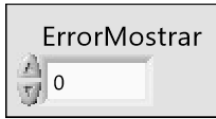
NumEstadosOut

StringFichero

OutMostrar



ComprobarficheroErrores.vi
C:\TFG\Microrad\ComprobarficheroErrores.vi
Last modified on 06-Sep-17 at 2:14 PM
Printed on 06-Sep-17 at 2:20 PM



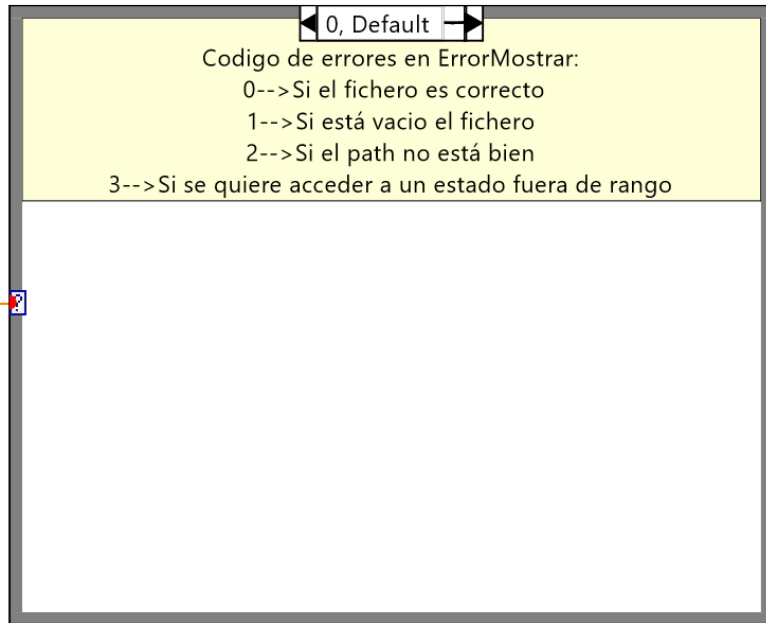
ErrorMostrar

ErrorMostrar

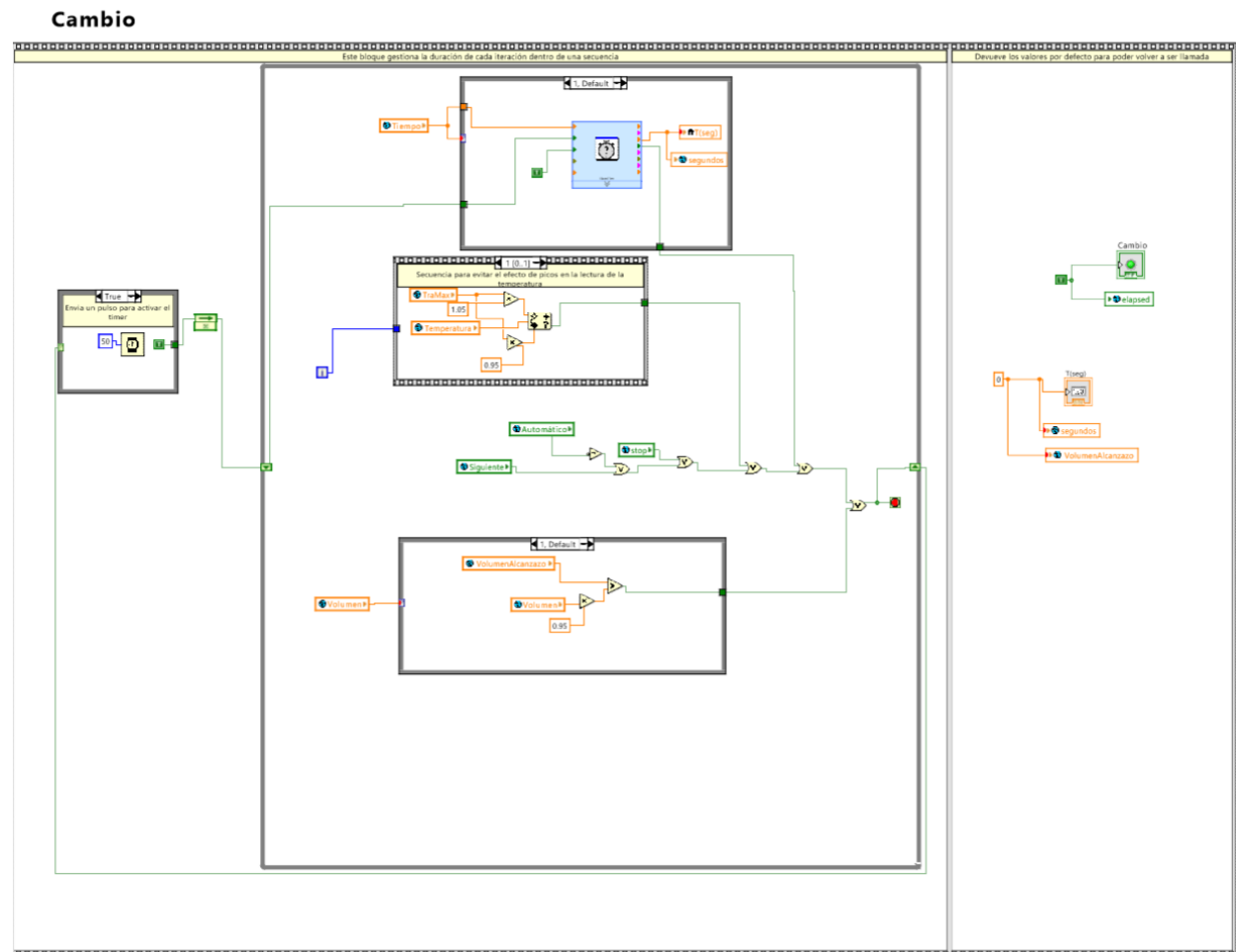


◀ 0, Default ▶

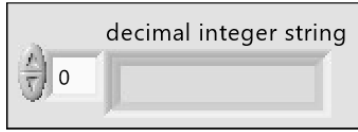
Código de errores en ErrorMostrar:
0-->Si el fichero es correcto
1-->Si está vacío el fichero
2-->Si el path no está bien
3-->Si se quiere acceder a un estado fuera de rango



CondicionesCambio.vi
C:\TFG\Microrad\CondicionesCambio.vi
Last modified on 06-Sep-17 at 2:16 PM
Printed on 06-Sep-17 at 2:20 PM

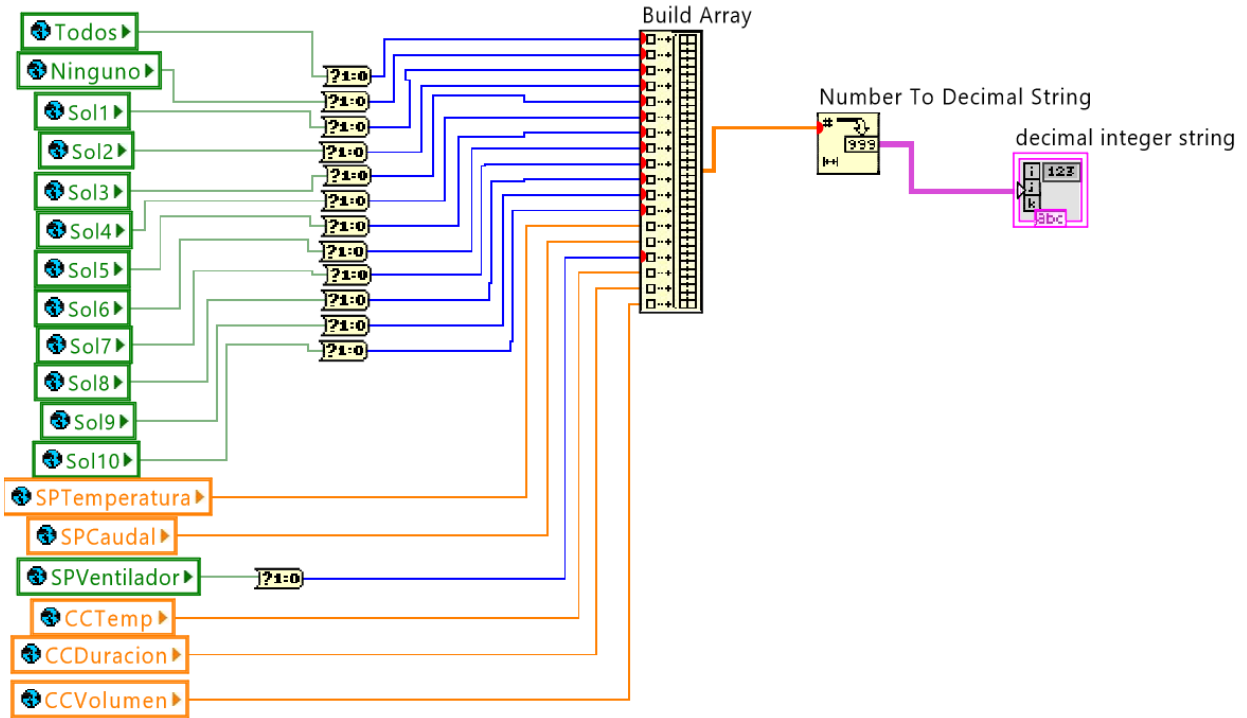


EscribeEstado.vi
C:\TFG\Microrad\EscribeEstado.vi
Last modified on 24-Aug-17 at 7:24 PM
Printed on 06-Sep-17 at 2:20 PM



decimal integer string

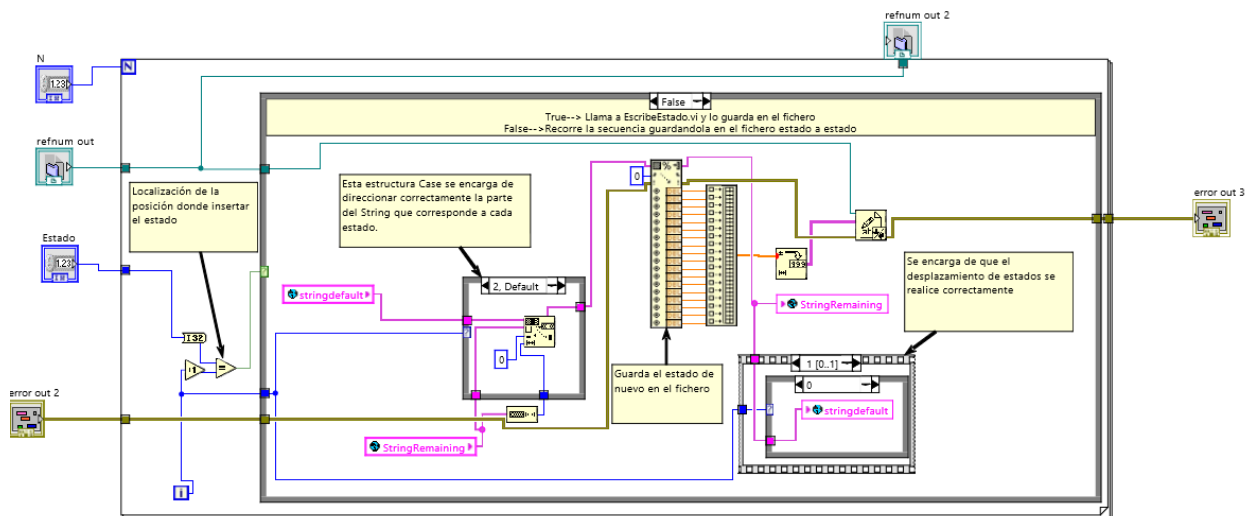
CCVolumen



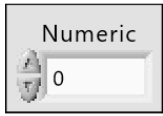
InsertaEstado.vi
 C:\TFG\Microrad\InsertaEstado.vi
 Last modified on 30-Aug-17 at 3:10 PM
 Printed on 06-Sep-17 at 2:20 PM



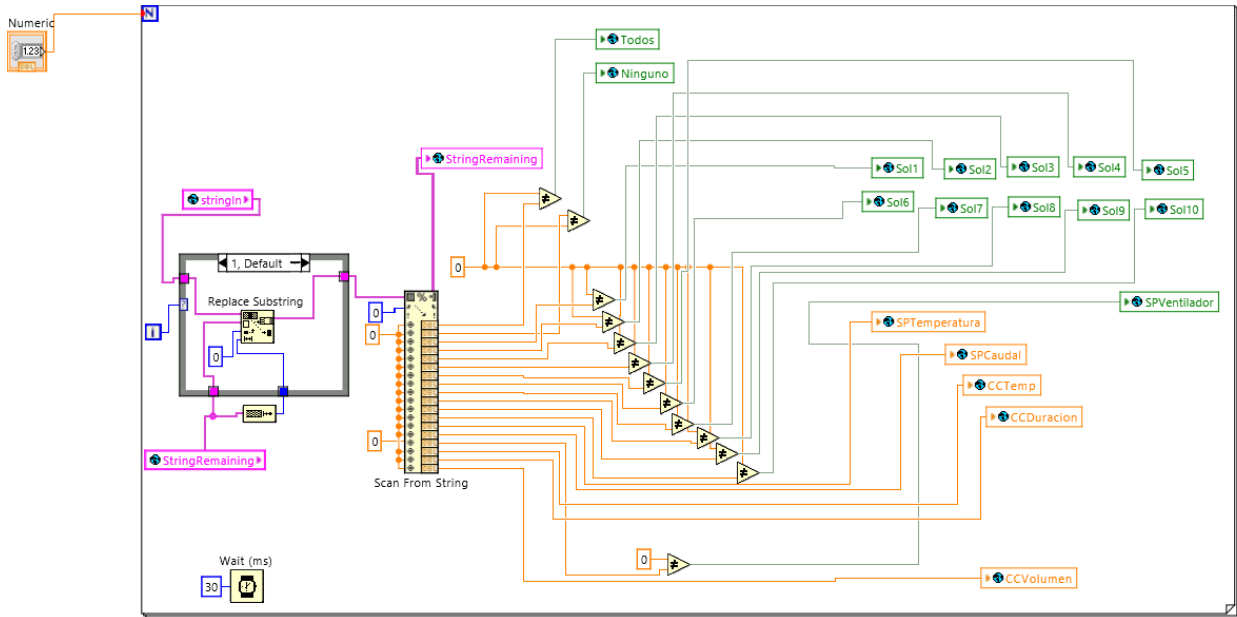
refnum out
Estado
N
refnum out 2



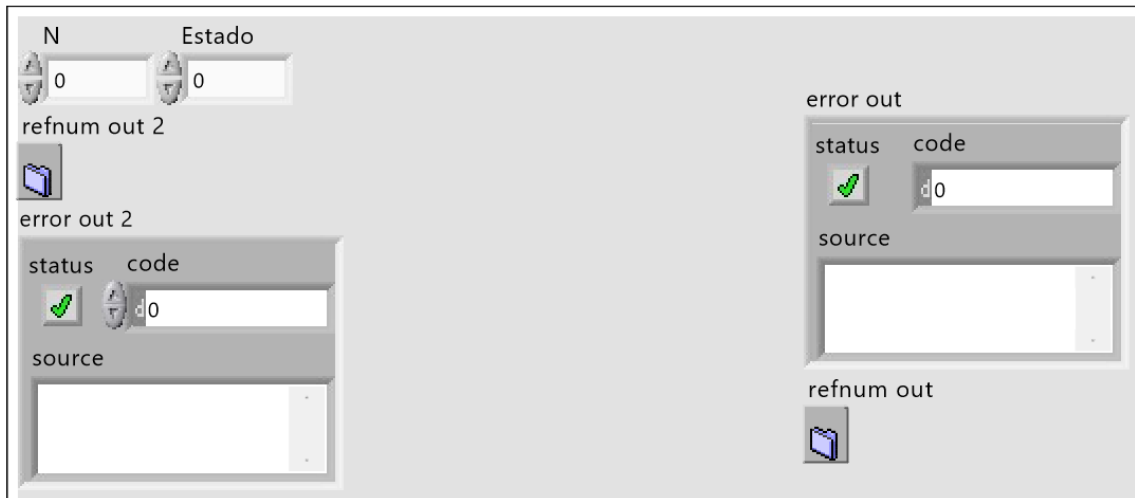
LeeEstado.vi
C:\TFG\Microrad\LeeEstado.vi
Last modified on 30-Aug-17 at 1:25 PM
Printed on 06-Sep-17 at 2:20 PM



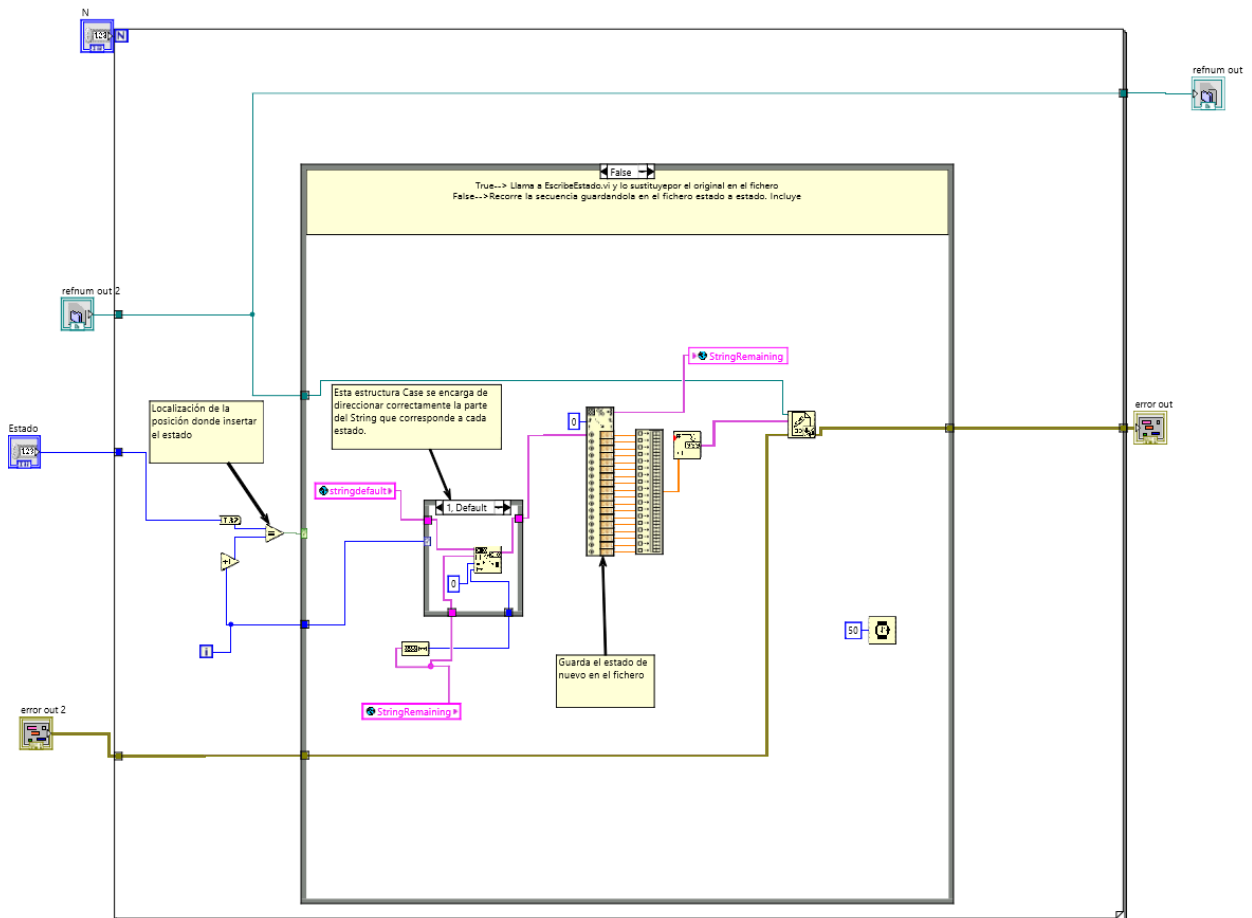
Numeric



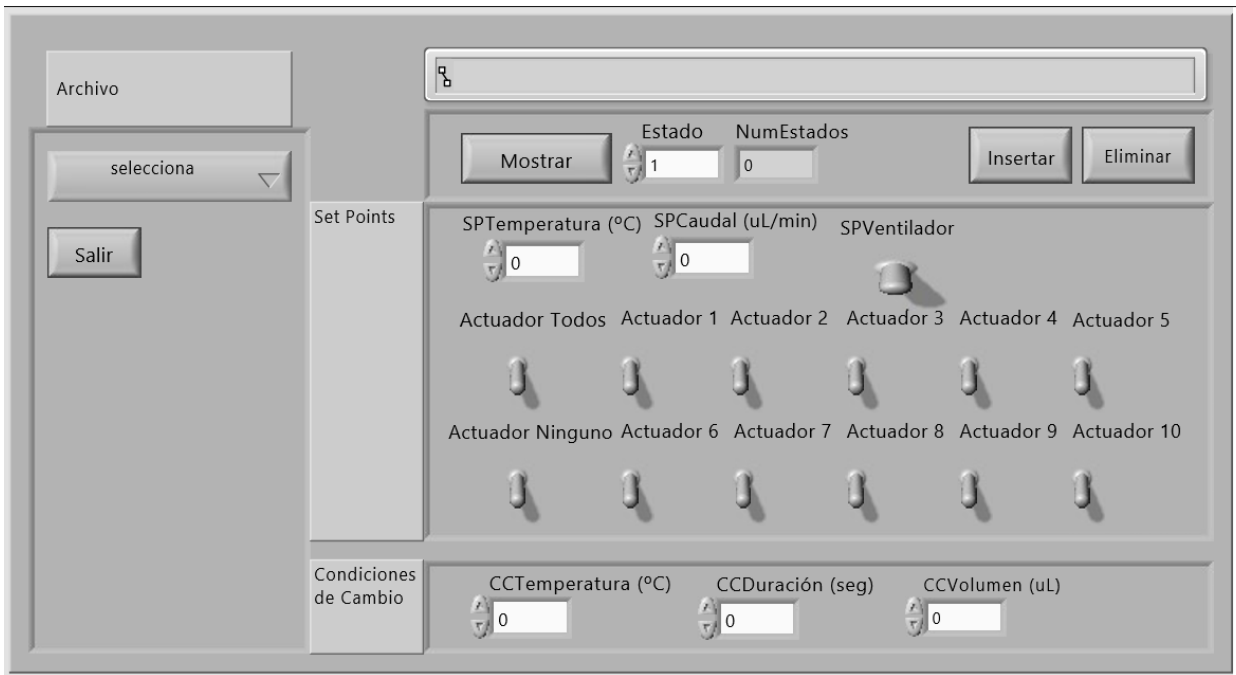
ModificaEstado.vi
 C:\TFG\Microrad\ModificaEstado.vi
 Last modified on 30-Aug-17 at 2:51 PM
 Printed on 06-Sep-17 at 2:21 PM



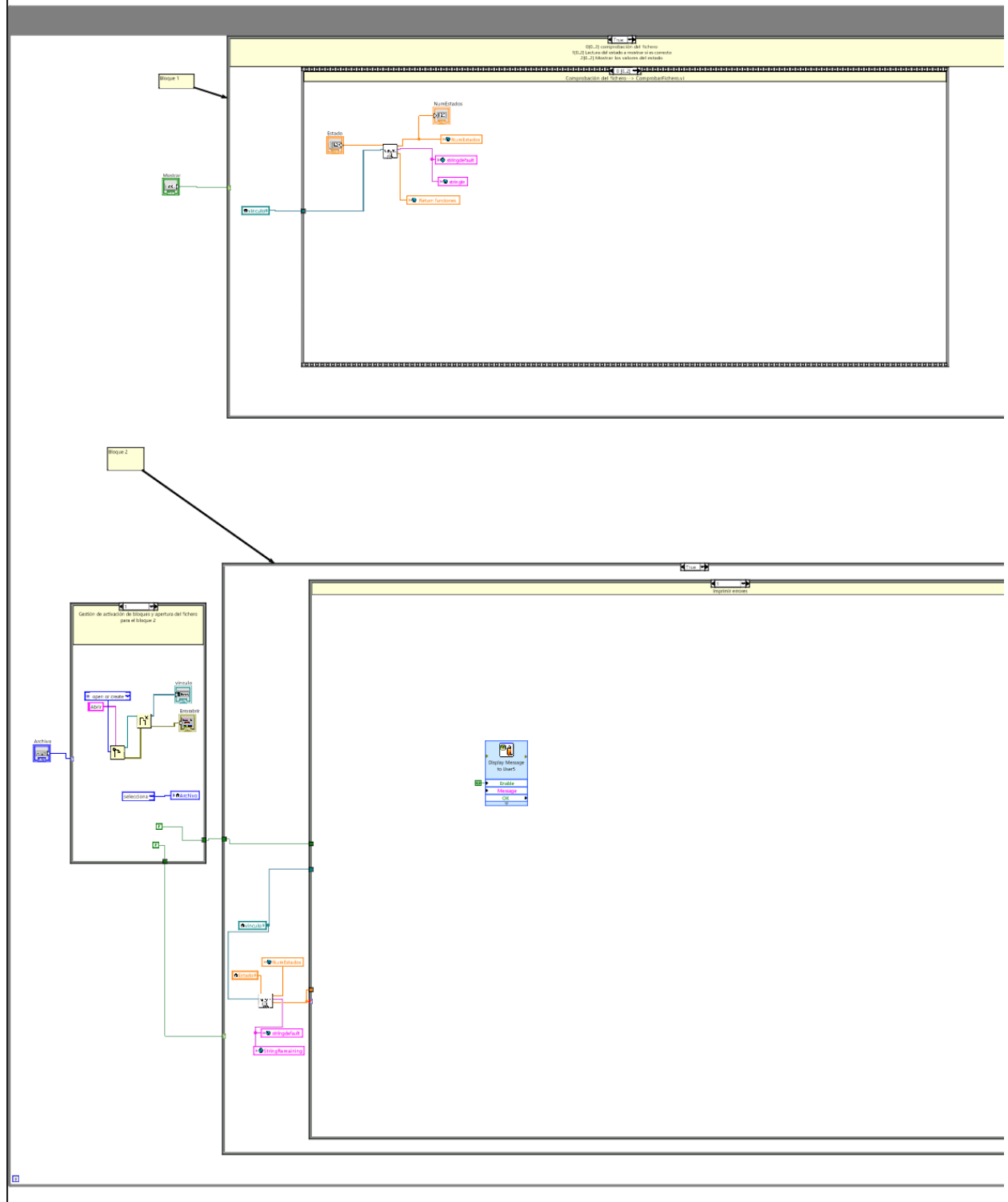
N
refnum out 2
Estado
refnum out



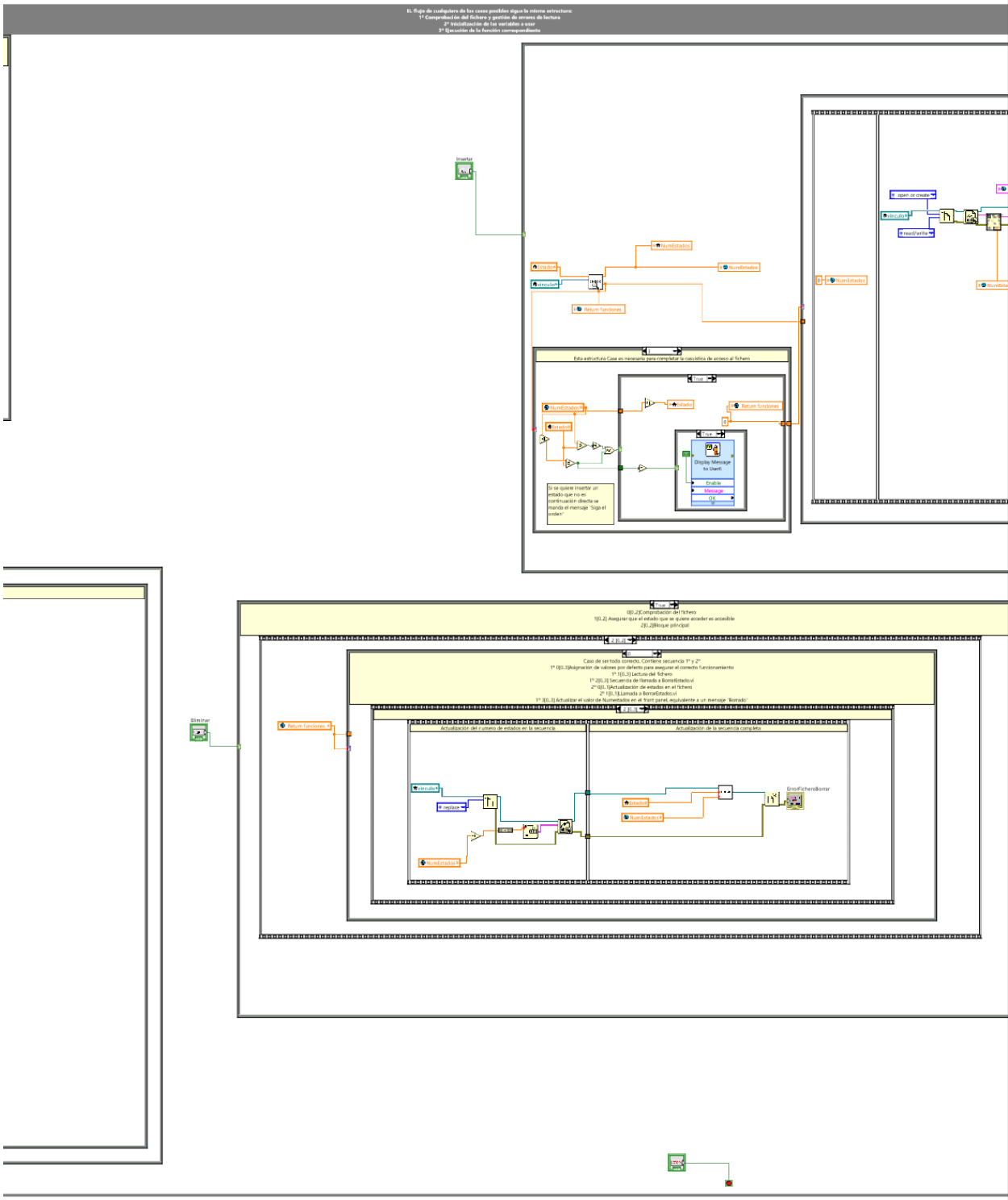
ModoAutomatico.vi
C:\TFG\Microrad\ModoAutomatico.vi
Last modified on 30-Aug-17 at 2:53 PM
Printed on 06-Sep-17 at 2:21 PM



ModoAutomatico.vi
C:\TFG\Microrad\ModoAutomatico.vi
Last modified on 30-Aug-17 at 2:53 PM
Printed on 06-Sep-17 at 2:21 PM



ModoAutomatico.vi
C:\TFG\Microrad\ModoAutomatico.vi
Last modified on 30-Aug-17 at 2:53 PM
Printed on 06-Sep-17 at 2:21 PM



Global 1.vi
C:\TFG\Microrad\Global 1.vi
Last modified on 25-Aug-17 at 6:51 PM
Printed on 06-Sep-17 at 2:20 PM

Variables Globales para la comunicación con Arduino y Fluiqent

VISA resource: COM3

Board Type: Mega

STOP

Temperatura: 0

Solenoides: Solenoide 1-10

ChannelNumberQ: 1

VolumenAlcanzado: 0

Consigna Temperatura: 0

Ventilador: 0

CaudalAlcanzado: 0

Indicadores: Indicador 1-10

Variables globales para funciones

stringIn

StringRemaining: 0

stringdefault

Return funciones

Sol1-Sol10, Todos, Ninguno

SPTemperatura: 0

SPCaudal: 0

SPVentilador: 0

CCTemp: 0

CCDuracion: 0

CCVolumen: 0

Variables globales para funciones

NumEstados: 3

segundos: 0

Siguiete

output value: 0.00

caudal: 0

TraMax: 0

Tiempo: 0

Volumen: 0

Automático

REFERENCIAS

- [1] Memoria científico-técnica para proyectos de investigación de excelencia. Convocatoria 2012. Microlab-on-chip para producción de radiofármacos para diagnóstico PET.
- [2] Radiological Society of North America, Inc. (RSNA). www.radiologyinfo.org
- [3] Mayka Sánchez. Madrid 18 FEB 2003. El País.
- [4] flyer_Software. SOFTWARE SOLUTIONS. Fluigent Smart microfluidics
- [5] MECÁNICA DE FLUIDOS Tema2. Impulsión de fluidos. I. Martín, R. Salcedo, R. Font. Universidad de Alicante → https://rua.ua.es/dspace/bitstream/10045/20299/4/tema2_impulsion.pdf
- [6] flyer_Microfluidic-Flow-Control-System-EZ. MFCSTM -EZ Microfluidic Flow Control System. Fluigent Smart microfluidics
- [7] flyer_Flow-Rate-Platform. FLOW-RATE PLATFORM. Fluigent Smart microfluidics
- [8] Ficheros de texto en Python. 2015/10/01 pitando.net
- [9] Cómo: Escribir texto en archivos en Visual Basic. <https://docs.microsoft.com/es-es/dotnet/visual-basic/developing-apps/programming/drives-directories-files/how-to-write-text-to-files>
- [10] LabVIEW™: applications and solutions / Rahman Jamal, Herbert Pichlik
- [11] Plataforma para la caracterización experimental de dispositivos de RF. TFG de Ignacio Pérez Lupiáñez, 2016
- [12] National Instruments: <http://www.ni.com/getting-started/LabVIEW™™-basics/esa/debug>
- [13] 354_MAN_001_A User Manual MAESFLO V340
- [14] Flow-Rate Platform LabVIEW API
- [15] 001_MAN_006_A Flow-Rate Platform User Manual
- [16] 001_MAN_006_A_Script Module v1.6 User Manual
- [17] 350_MAN_002_A FRP SFP User Manual

