UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA AUTÓNOMO PARA LA ADQUISICIÓN DE VARIABLES ANALÓGICAS BASADO EN CRIO 9012 Y CON COMUNICACIÓN ETHERNET

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO ELECTRÓNICO

(Á R E A: E L E C T R Ó N I C A)

P R E S E N T A:

FABIO ALEJANDRO RUIZ MOLINA

DIRECTOR DE TESIS: M. I. LAURO SANTIAGO CRUZ

MÉXICO, D. F. 2014

AGRADECIMIENTOS

*A mi maestro y tutor **Lauro Santiago**, por todo su tiempo, sus consejos y su confianza.*

*A mis amigos **Miguel Ibáñez, Héctor Valera, Héctor Cuchillo y Erick Preciado**, sin ustedes mi estancia en el laboratorio no hubiera sido divertida.*

*A **Enrique Gómez**, por todas las facilidades brindadas para el desarrollo del presente trabajo.*

*A **Ponciano Trinidad**, por su apoyo para llevar a cabo las pruebas del sistema desarrollado.*

*A **todos mis maestros** a lo largo de mi vida, sin ustedes no sería la persona que hoy soy.*

DEDICATORIAS

*A mis padres, **Oralia y José**, por darme la vida, por todo su amor y por su confianza. Sin su apoyo este logro seguiría siendo un sueño.*

*A mi **Roxan**, porque sin tu amor, comprensión y compañía este proyecto de vida no sería una realidad. Forever and never.*

*A mis hermanos **Lidia, Pepe y Paco**, por hacer de mi familia una familia especial.*

Contenido

Índice de Figuras vii
Índice de Tablas xi
Prólogo xii
Capítulo 1. Introducción 1
Capítulo 2. Generalidades del sistema 4
2.1. Sistemas embebidos 4
2.2. Sistemas de adquisición de señales (SAS) 5
2.3. Elementos y procesos en la adquisición de señales 6
2.3.1. Conversión analógica-digital 7
2.4. Microcontrolador 9
2.4.1. Arquitecturas von Neumann y Harvard 10
2.4.2. Arquitecturas CISC y RISC 11
2.4.3. Microcontrolador Freescale MPC5200 12
2.5. FPGA 14
2.5.1. Elementos lógicos 15
2.5.2. El arreglo y la interconexión 17
2.5.3. Lógica extendida 20
2.5.4. Configuración 22
2.5.5. FPGA Virtex-5 LX50 24
2.6. Bus serie universal 27
2.7. Almacenamiento masivo USB 28
2.8. Ethernet 31
2.9. Interconexión de componentes periféricos 38
2.10. Unidad de estado sólido 39
2.11. Reloj de tiempo real 41
2.12. La programación gráfica 42
Capítulo 3. Desarrollo del sistema 46
3.1. Requerimientos 46
3.2. El hardware del sistema 47
3.2.1. MCU embebido de tiempo real 48
3.2.2. Chasis reconfigurable basado en FPGA 54
3.2.3. Sistema de conversión analógico-digital 55
3.2.4. Computadora personal 68
3.3. El software del sistema 68
3.3.1. Software de la PC 69
3.3.2. Software del MCU del cRIO 9012 86
3.3.3. Software del FPGA en el chasis 9113 113
Capítulo 4. Pruebas del sistema 125
4.1. Pruebas con geófono 125
4.1.1. Configuración de la prueba 127
4.1.2. Prueba en marcha 134

4.1.3. Procesamiento con software de PC 139
4.2. Pruebas con generador de señales 149
Capítulo 5. Resultados y conclusiones 154
5.1. Resultados 154
5.2. Conclusiones 155
5.3. Comentarios finales 156
Apéndices 158
A. Ventanas del software del sistema 159
B. Código implementado 167
Bibliografía 178

ÍNDICE DE FIGURAS

Capítulo 2

Figura 2.1. Estructura general de un SAS para medida 5
Figura 2.2. Señal senoidal y su versión digital con resolución de 3 bits 8
Figura 2.3. Estructura general de un MCU 9
Figura 2.4. a) Arquitectura von Neumann, b) Arquitectura Harvard 11
Figura 2.5. Diagrama de bloquees simplificado del MPC5200 13
Figura 2.6. Una vista abstracta de un FPGA 15
Figura 2.7. Esquemático de una LUT de tres entradas 16
Figura 2.8. Bloque lógico LUT 17
Figura 2.9. La arquitectura island-style de un FPGA 17
Figura 2.10. Arquitectura island-style compleja 18
Figura 2.11. Bloque de conexión a detalle 18
Figura 2.12. Arquitectura del bloque de conmutación 19
Figura 2.13. Jerarquía de conexión para enlaces largos 19
Figura 2.14. Sumador completo de 4 bits 20
Figura 2.15. Flujo típico para lograr configurar un FPGA 23
Figura 2.16. Virtex-5 CLB y slice 25
Figura 2.17. LUT de seis entradas con doble salida 25
Figura 2.18. a) Bloque RAM y b) bloque DSP del Virtex-5 26
Figura 2.19. Conectores USB 28
Figura 2.20. Estructura general memoria USB basada en Flash 29
Figura 2.21. Estructura de la pila del protocolo de red para Ethernet 32
Figura 2.22. Estructura de un segmento TCP 34
Figura 2.23. Estructura general del datagrama IP 35
Figura 2.24. Estructura general de un Frame Ethernet 36
Figura 2.25. Panel frontal (izquierda) y diagrama de bloques (derecha) 43
Figura 2.26. Icono de un VI (izquierda) y su conector (derecha) 43

Capítulo 3

Figura 3.1. Esquema de alto nivel de CompactRIO 48
Figura 3.2. NI cRIO 9012 (MCU de tiempo real) 49
Figura 3.3. Puertos, indicadores y controles de NI cRIO 9012 50
Figura 3.4. Distribución de terminales en conector RJ-45 53
Figura 3.5. Chasis reconfigurable NI 9113 54
Figura 3.6. Ensamble de NI cRIO con chasis reconfigurable NI 9113 56

Figura 3.7. Módulo de entradas análogicas NI 9205 56
Figura 3.8. Circuito de entrada para un canal analógico del NI 9205 57
Figura 3.9. Terminales del módulo NI 9205 57
Figura 3.10. Medición diferencial 58
Figura 3.11. Voltaje en modo común 59
Figura 3.12. Medición diferencial de señal referenciada 61
Figura 3.13. Conexión al NI 9205 61
Figura 3.14. Medición diferencial de señales flotadas 63
Figura 3.15. Conexión al NI 9205 63
Figura 3.17. Medición en modo terminación simple de una señal referenciada 64
Figura 3.18. Medición usando el NI 9205 65
Figura 3.19. Conexión para medición NRSE 66
Figura 3.20. Gráfica del típico CMRR típico entre los canales A1+ y A1- 66
Figura 3.21. Software asociado al sistema cRIO 69
Figura 3.22. Diagrama de flujo de la ventana Menú 70
Figura 3.23. Primera parte del diagrama de flujo de la ventana Procesamiento 72
Figura 3.24. Diagrama de flujo del sub VI Lee configuración 73
Figura 3.25. Segunda parte del diagrama de flujo de la ventana Procesamiento 74
Figura 3.26. Tercera parte del diagrama de flujo de la ventana Procesamiento (1/2) 76
Figura 3.27. Tercera parte del diagrama de flujo de la ventana Procesamiento (2/2) 77
Figura 3.28. Cuarta parte del diagrama de flujo de la ventana Procesamiento 79
Figura 3.29. Diagrama de flujo del sub VI multiGráfico 80
Figura 3.30. Quinta parte del diagrama de flujo de la ventana Procesamiento 81
Figura 3.31. Primera parte del diagrama de flujo de la ventana FTP 84
Figura 3.32. Segunda parte del diagrama de flujo de la ventana FTP 85
Figura 3.33. Número de canales vs Número de muestras por segundo 89
Figura 3.34. Primera parte del diagrama de flujo de la página web Principal 93
Figura 3.35. Primera parte del diagrama de flujo del sub VI Configuración de canales 95
Figura 3.36. Diagrama de flujo del sub VI Cálculo de frecuencias 96
Figura 3.37. Segunda parte del diagrama de flujo del sub VI Configuración de canales 96
Figura 3.38. Tercera parte del diagrama de flujo del sub VI Configuración de canales 98
Figura 3.39. Primera parte del diagrama de flujo del sub VI Configura txt 99
Figura 3.40. Segunda parte del diagrama de flujo del sub VI Configura txt 99
Figura 3.41. Diagrama de flujo del sub VI Última configuración 100
Figura 3.42. Cuarta parte del diagrama de flujo del sub VI Configuración de canales 101
Figura 3.43. Diagrama de flujo sub VI Fecha-Hora 103
Figura 3.44. Diagrama de flujo sub VI Frecuencia-Ticks 104
Figura 3.45. Diagrama de flujo de sub VI Cantidad de archivos 104
Figura 3.46. Diagrama de flujo de sub VI Cálculos DMA 105
Figura 3.47. Segunda parte del diagrama de flujo de la página Principal 105
Figura 3.48. Tercera parte del diagrama de flujo de la página Principal 107
Figura 3.49. Primera parte del diagrama de flujo del sub VI Modo Continuo 108
Figura 3.50. Segunda parte del diagrama de flujo del sub VI Modo Continuo 110
Figura 3.51. Diagrama de flujo del sub VI Modo Disparo 111
Figura 3.52. Diagrama de flujo del sub VI Separa canales 112
Figura 3.53. Diagrama de flujo para disparo sobre umbral inferior 116
Figura 3.54. Diagrama de flujo para disparo debajo umbral inferior 117
Figura 3.55. Diagrama de flujo para disparo en intervalo 118

Figura 3.56. Diagrama de flujo de inicio y configuración del VI FPGA 119
Figura 3.57. Diagrama de flujo para adquisición del VI FPGA 121
Figura 3.58. Diagrama de flujo para la transmisión del VI FPGA 124

Capítulo 4

Figura 4.1. Conexiones para prueba con geófono 125
Figura 4.2. Salida vs Frecuencia GS-11D 126
Figura 4.3. Geófonos GS-11D 126
Figura 4.4. Página web Principal 128
Figura 4.5. Menú de control remoto 128
Figura 4.6. Ventana Fecha-Hora 129
Figura 4.7. Configuración de Hora 129
Figura 4.8. Configuración de Fecha 130
Figura 4.9. Prueba de Fecha y Hora 130
Figura 4.10. Ventana Configuración Canales 131
Figura 4.11. Apartado Parámetros de Canales 131
Figura 4.12. Apartado Configuración Almacenamiento 132
Figura 4.13. Generación de archivo de configuración 133
Figura 4.14. Carpeta y archivo de configuración en memoria 133
Figura 4.15. Contenido de archivo de configuración 134
Figura 4.16. Página principal en adquisición 134
Figura 4.17. Ventana Modo Software 135
Figura 4.18. Osciloscopio en monitoreo 136
Figura 4.19. Sistema en monitoreo 137
Figura 4.20. Adquisición en progreso 137
Figura 4.21. Adquisición realizada 138
Figura 4.22. Archivos de adquisición en memoria 138
Figura 4.23. Ejecutable de software para PC 139
Figura 4.24. Ventana Menú 140
Figura 4.25. Ventana FTP 140
Figura 4.26. Respuesta servidor FTP 141
Figura 4.27. Selección de la ubicación de archivo 142
Figura 4.28. Selección de archivo de configuración 142
Figura 4.29. Ventana Procesamiento 143
Figura 4.30. Datos de archivo a procesar 144
Figura 4.31. Mensaje por lectura 144
Figura 4.32. Información de canales seleccionados 144
Figura 4.33. Visualización de datos 146
Figura 4.34. Controles del despliegue gráfico 146
Figura 4.35. Conversión parcial o total del archivo 147
Figura 4.36. Archivo con codificación ASCII 148
Figura 4.37. Segundo archivo generado 148
Figura 4.38. Tercer archivo generado 149
Figura 4.39. Cuarto archivo generado 149
Figura 4.40. Interconexión para pruebas con generador 150
Figura 4.41. Archivos en PC de modo temporizado 151
Figura 4.42. Propiedades de los archivos 151
Figura 4.43. Primer archivo temporizado 152
Figura 4.44. Segundo archivo temporizado 153

Figura 4.45. Tercer archivo temporizado 153

Índice de tablas

Capítulo 2
Tabla 2.1. Tipos de cable soportados por Ethernet 38
Tabla 2.2. Equivalencia entre LabVIEW y leguajes convencionales 44
Capítulo 3
Tabla 3.1. Cableado para conexiones Ethernet 53
Tabla 3.2. Pares para medición diferencial 58
Tabla 3.3. Coeficientes de escalamiento 67

PRÓLOGO

Los campos de ingeniería, ciencia y tecnología actualmente son muy dinámicos, debido a los recientes avances en electrónica, computación y otras áreas tecnológicas. Estos avances han dado como resultado en un número de programas de computadora y sistemas electrónicos que resuelven tanto problemas tradicionales como actuales. Estos programas y sistemas, usando el incremento de las capacidades de la computación y la electrónica, son la clave para la revolución y éxito de instrumentación con comunicación vía web, desarrollando aplicaciones que pueden conectarse con hardware desde cualquier parte del planeta.

Actualmente existe una gran cantidad de software y de hardware, que en conjunto permiten desarrollar sistemas de gran alcance. Evidentemente hay software y hardware que permiten crear sistemas más complejos que otros. Este es el caso de Laboratory Virtual Instrumentation Engineering Workbench (LabVIEW) y Compact Reconfigurable Input Output (cRIO). El crecimiento exponencial de LabVIEW lleva a que National Instruments maneje la frase “LabVIEW en todas partes”. Esta frase se encuentra respaldada por el creciente potencial de los instrumentos virtuales desarrollados en LabVIEW y por su capacidad para ejecutarse en una gran variedad de hardware. Incluso existe la oportunidad de reconfigurar el hardware para que su funcionalidad tome caminos que sus diseñadores no habían anticipado.

El comunicar un arreglo de compuertas programables en campo (FPGA, por sus siglas en inglés), un microcontrolador (MCU, por sus siglas en inglés) y periféricos entrada/salida, resulta una tarea laboriosa pero que de lograrse representa un sistema muy poderoso y de amplia utilidad en varias áreas de la ingeniería que necesitan adquisición, almacenamiento y procesamiento de señales analógicas. Aún más provechoso es el hecho de poder comunicar el sistema anteriormente descrito con una computadora. Y por si fuera poco, sería muy útil el hecho de poder adquirir decenas de señales analógicas, a tasas de muestreo diferentes, con frecuencias de muestreo altas, con miles de datos por segundo y simultáneamente. Todo esto culminaría en un sistema que puede realizar actividades de acorde con las exigencias actuales de las diferentes áreas de la ingeniería.

El presente trabajo describe el proceso de diseño y desarrollo de un sistema de adquisición de variables analógicas, con comunicación Ethernet y con almacenamiento de datos vía bus universal en serie (USB, por sus siglas en inglés). El desarrollo del sistema está basado en la tecnología cRIO y desarrollado con el software LabVIEW. El sistema permitirá tener una herramienta importante para las distintas áreas del Instituto de Ingeniería de la Universidad Nacional Autónoma de México.

El presente trabajo está organizado en cinco capítulos, estos están desarrollados de la siguiente manera:

- Capítulo 1. Este capítulo está destinado para hacer la introducción del trabajo. Se presenta la motivación del mismo y una propuesta que permita dar solución al problema planteado.
- Capítulo 2. En este capítulo se abordan temas que ayudan a entender en general las implicaciones del sistema desarrollado. Se empieza abordando conceptos básicos relacionados con el sistema. Se continúa describiendo a grandes rasgos el hardware y el software que se ha usado. Prácticamente se hace una descripción de las arquitecturas de los subsistemas electrónicos y del modo de operar de LabVIEW. Inclusive se abordan temas de almacenamiento y comunicación de datos entre dispositivos.
- Capítulo 3. Aquí se presenta lo que se espera del sistema. Se realiza una descripción de la tecnología cRIO 9012, NI 9113, NI 9205, de LabVIEW y de los respectivos módulos de LabVIEW para controlar a cRIO 9012. Posteriormente se muestran las implicaciones de las mediciones con NI 9205. También se plantea el requerimiento para cada parte del software desarrollado y el funcionamiento del mismo, éste último basado en diagramas de flujo.
- Capítulo 4. Dentro de este capítulo se presentan pruebas que dan constancia del funcionamiento del sistema desarrollado. Se expone el proceso que se debe seguir para poner en operación al sistema.
- Capítulo 5. Este último capítulo está dedicado a los resultados y conclusiones. Se realiza una inspección sobre el cumplimiento del objetivo principal. Además se plantean las mejoras que en un futuro se le podrían realizar al sistema. Al final se resalta la importancia del trabajo realizado.

Posterior a los cinco capítulos, se encuentran los apéndices. En el primero de ellos se puede ver las ventanas del software desarrollado y con las cuales podrá interactuar el usuario. En el segundo apéndice se presentan algunas partes del código implementado. Al final se encuentra la bibliografía consultada.

Capítulo 1
Introducción

En este capítulo se presenta la motivación del trabajo realizado, el objetivo principal del mismo y una propuesta que permita cumplir con este objetivo.

LabVIEW es un ambiente de programación gráfico basado en el concepto de programación de flujo de datos. Este paradigma de la programación ha sido ampliamente usado para adquisición de datos, instrumentación y control. Hay tres componentes importantes involucrados en aplicaciones de prueba y medición: adquisición de datos, análisis de datos y visualización de datos. Las características de LabVIEW cubren estos componentes y además brindan un entorno de desarrollo fácil de usar.

cRIO de National Instruments es un sistema avanzado y embebido de control y adquisición de datos, diseñado para aplicaciones que requieren alto rendimiento y fiabilidad. Con la arquitectura abierta y embebida, tamaño pequeño, extrema robustez y flexibilidad del sistema es posible construir sistemas embebidos personalizados con un excelente desempeño.

cRIO combina un microcontrolador de tiempo real embebido, un FPGA de alto rendimiento y módulos de entrada/salida intercambiables. Cada módulo de entrada/salida se conecta directamente al FPGA, proporcionando personalización de bajo nivel para temporización y procesamiento de señales de entrada/salida. El FPGA es conectado al microcontrolador de tiempo real embebido vía un bus interconexión de componentes periféricos (PCI, por sus siglas en inglés) de alta velocidad. Esto representa una arquitectura de bajo costo (en comparación a otros sistemas dedicados), con acceso abierto a recursos de hardware de bajo nivel. LabVIEW contiene mecanismos integrados para transferencia de datos, para pasar datos desde los módulos de entrada/salida al FPGA y también desde el FPGA al microcontrolador embebido; para análisis en tiempo real, procesamiento posterior, registro de datos o comunicación a una computadora conectada en red.

Actualmente para cada área de la ingeniería se cuenta con sistemas de adquisición dedicados, que son capaces de resolver algunas de las necesidades actuales de cada área, pero difícilmente las compañías que se dedican a la construcción de adquisidores para las distintas áreas de la ingeniería, tienen en su catálogo de ventas un sistema estándar que sea capaz de asimilar señales analógicas

de diferente naturaleza, previamente acondicionadas, con las características mencionadas en el párrafo anterior.

Es propósito de esta tesis implementar herramientas de hardware (cRIO 9012, NI 9113 y NI 9205) y herramientas de software (LabVIEW) para desarrollar un sistema de adquisición de señales analógicas estándar, al cual sólo se le tenga que agregar una tarjeta de acondicionamiento según la naturaleza de la señal analógica a adquirir y así poder desarrollar un sistema embebido robusto, flexible y confiable, que pueda adquirir cualquier tipo de señal analógica previamente acondicionada.

El sistema contará con un subsistema de digitalización (NI 9205), el cual enviará datos a un FPGA (NI 9113), que a su vez será controlado por un microcontrolador (cRIO 9012). Este último deberá establecer comunicación con una computadora de dos maneras:

1. Puerto Ethernet y una aplicación desarrollada en LabVIEW, la cual se estará ejecutando en una computadora.
2. Puerto Ethernet y una página web desarrollada en LabVIEW, que se encontrará empotrada en el cRIO y disponible todo el tiempo.

Las dos opciones de comunicación listadas anteriormente establecerán comunicación con:

1. Un instrumento virtual (VI, por sus siglas en inglés) que se encontrará ejecutando en el MCU embebido en el cRIO 9012.
2. Un VI que se encontrará ejecutando en el FPGA embebido en el NI 9113.

Los instrumentos virtuales descritos en los dos puntos anteriores se encontrarán ejecutando de manera autónoma y perenne, mientras el cRIO se encuentre alimentado correctamente. En dado caso de que se suspenda el suministro de energía, al regresar ésta, el sistema será capaz de retomar los parámetros de configuración y continuar con la adquisición de las señales analógicas, lo anterior siempre y cuando el usuario reinicie el proceso. Con una alimentación continua el sistema se detiene hasta cumplir condiciones de paro impuestas por el usuario.

El microcontrolador también tendrá la capacidad de comunicarse con un dispositivo de estado sólido (SSD, por sus siglas en inglés) y una memoria flash con comunicación USB, ambas para almacenamiento de datos. En la memoria flash se almacenarán archivos binarios que contienen los datos derivados de un proceso de adquisición y que pueden ser leídos en el entorno LabVIEW. Con la finalidad de facilitar la lectura de la información, y que ésta pueda ser procesada con software convencional (Matlab, Word, Excel, Bloc de notas, etc.), existirá la opción de pasar los datos a código estándar estadounidense para el intercambio de información (ASCII, por sus siglas en inglés).

Se usará un geófono y un generador de señales para ejemplificar el proceso de adquisición de datos, desde la configuración del subsistema de digitalización (NI 9205)

pasando por el procesamiento de la señal con el FPGA y microcontrolador hasta el procesamiento de los datos en la computadora.

Cada canal tendrá asociados los siguientes parámetros:

1. Constante de amplificación.
2. Datos de 32 bits.
3. Frecuencia de muestreo.
4. Habilitado/Deshabilitado.
5. Nombre del canal.
6. Rango de voltaje: $\pm$10 V, $\pm$5 V, $\pm$1 V o $\pm$200 mV.
7. Modo de operación: diferencial, terminación simple no referenciada (NRSE, por sus siglas en inglés) o terminación simple referenciada (RSE, por sus siglas en inglés).
8. Unidades.
9. Formato de Hora: tiempo universal coordinado (UTC, por sus siglas en inglés), local o verano.

El almacenamiento se dará bajo las siguientes condiciones:

1. El usuario lo inicia y detiene.
2. En un intervalo de tiempo.
3. Cuando la señal en medición alcance un nivel o se encuentre en un intervalo definido por el usuario.
4. Continuamente, hasta agotarse el espacio de almacenamiento.

Procesamiento de datos:

1. Despliegue de la información en gráficas.
2. Despliegue de la información en tablas.
3. Información simultánea de un grupo de canales definidos por el usuario.
4. Información almacenada por canal en formato ASCII, con capacidad para ser leída por un software comercial.
5. Acceso aleatorio de los datos, para hacer uso eficiente de la memoria de acceso aleatorio (RAM, por sus siglas en inglés) de la computadora donde esté corriendo el software de procesamiento.

En el capítulo siguiente se abordarán las generalidades que permitirán profundizar en el entendimiento de lo desarrollado en la presente tesis.

CAPÍTULO 2
GENERALIDADES

En este capítulo se aborda de manera concisa los conceptos que contextualizan y ayudan a entender en plenitud el contenido de la presente tesis, la descripción de los puertos, de las arquitecturas internas del cRIO y por último de las herramientas de software necesarias para la programación del mismo.

2.1. Sistemas embebidos

Un sistema embebido es un sistema basado en un microprocesador, que ha sido construido para controlar una función o rango de funciones y no está diseñado para ser programado por el usuario final, muy diferente al caso de una computadora personal (PC, por sus siglas en inglés)[1]. Un usuario puede hacer elecciones sobre funcionalidad pero no puede cambiar la funcionalidad del sistema agregando o remplazando software. Un sistema embebido está diseñado para implementar una tarea en particular aunque con elecciones y diferentes opciones.

La aparición de microprocesadores embebidos fue mucho antes que la PC, los microprocesadores embebidos son mucho más usados en la vida cotidiana que cualquier otro circuito electrónico que se haya fabricado. Un carro puede tener más de 50 microprocesadores que controlan funciones y adquieren señales de los distintos elementos que conforman al sistema (bolsas de aire, motor, frenos, seguros, etc.). Con el advenimiento de la era digital se están remplazando muchas de las tecnologías analógicas. En el mundo de consumo, el dominio de los sistemas embebidos es cada vez mayor.

Los sistemas embebidos pueden ser actualizados para mejorar su funcionalidad. La clave está en la capacidad para agregar funciones nuevas, no depende de cambiar el hardware, puede ser realizado por un simple cambio en el software. Si el sistema está conectado a una vía de comunicación como un teléfono o una PC en red, la actualización puede ser realizada remotamente sin tener que estar un ingeniero o técnico en el mismo lugar que el dispositivo.

[1] Heath S., Ebedded Systems Design, segunda edición, Newnes, Oxford, 2003, 2-3.

2.2. Sistema de adquisición de señales (SAS)

Un sistema es un conjunto de elementos o partes organizadas para realizar una función determinada. Un sistema electrónico es aquel cuyos elementos o partes realizan sus funciones respectivas por medios electrónicos.

En un SAS para medida, las entradas son señales generadas por diferentes fenómenos físicos, y las salidas son normalmente señales visuales, acústicas o eléctricas. Se considera que una señal es una función de variables independientes como el tiempo, la distancia, la posición, la temperatura y la presión. Los SAS suelen realizar una única tarea, aunque con grados de complejidad muy diversos. En la figura 2.1 se puede apreciar la estructura general de un SAS.

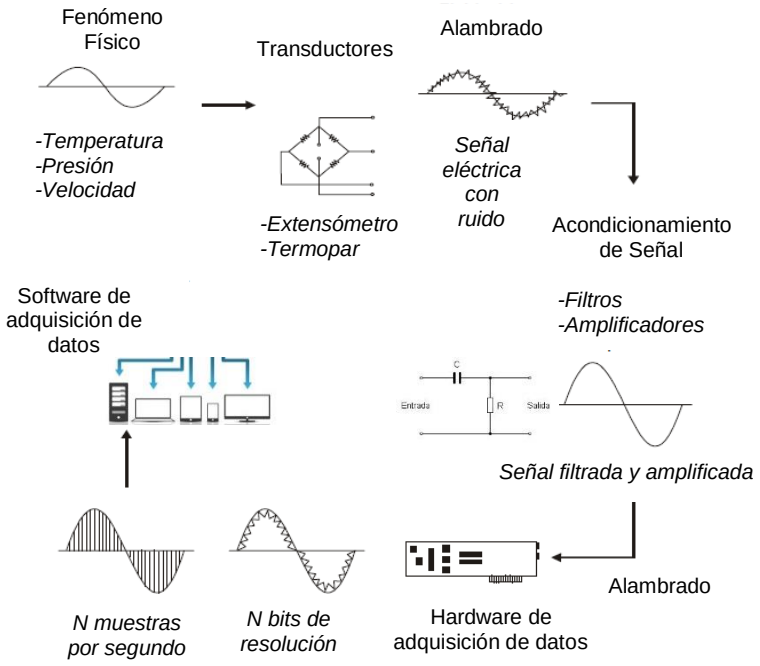


Figura 2.1. Estructura general de un SAS para medida.

Las dimensiones físicas y la complejidad de un SAS, evaluada por el número de elementos con una función diferente y de importancia, pueden ser muy dispares. El diseño de un SAS implica considerar las partes que lo integran y sus tareas respectivas. La identificación de cada parte se hace desde una perspectiva determinada que puede ser a nivel transistor, de componente, de circuito o de subsistema. En un nivel superior están los diseñadores de sistemas de tratamientos de datos, comunicación, medida o control, quienes emplean elementos en forma de equipos o tarjetas de circuito impreso para realizar su trabajo, que suele implicar la interconexión de elementos distantes[2].

[2] Pallás Areny R., Adquisición y Distribución de Señales, primera edición, Marcombo, Barcelona, 1993, 2-3.

2.3. Elementos y procesos en la adquisición de señales

En un sistema de medida hay que adquirir la información (en forma de señales analógicas o digitales), procesarla y presentarla. A veces, además, hay que registrarla. En un sistema de control hay que comparar el resultado de la medida con los objetivos establecidos, y actuar en consecuencia sobre el sistema físico o proceso para modificar el parámetro deseado. Ello requiere la generación de tensiones analógicas y digitales, y el control de la potencia a aplicar. Para obtener un funcionamiento correcto hay que contar y temporizar simultáneamente varios eventos independientes.

En el transcurso del presente trabajo, cuando se hable de señal analógica, se estará considerando una señal en tiempo continuo con amplitud continua. Por otro lado, cuando se hable de una señal digital, se estará considerando una señal en tiempo discreto, con amplitudes que toman valores discretos y que pueden ser representados por medio de un número finito de dígitos.

El transductor es el primer elemento de un SAS. Éste es el que mide la magnitud de interés. Ésta puede ser mecánica, térmica, eléctrica, magnética, óptica o química. Salvo en el caso de que las magnitudes a medir sean todas eléctricas, el elemento de medida es un transductor que convierte energía de una forma física en otra forma distinta, en nuestro caso en energía eléctrica. Un transductor se denomina también un sensor por la capacidad que ofrece de percibir, tras el procesamiento oportuno, fenómenos que de otra forma serían inaccesibles a nuestros sentidos[3].

Después del transductor, puede existir la conexión física de éstos hacia el hardware de acondicionamiento de señal y/o al hardware de adquisición de datos. En la figura 2.1 a esta parte se le llamó alambrado. Cuando el acondicionador de señal y/o adquisidor de datos está lejos del *host*, el alambrado da una conexión física entre estos elementos de hardware y el *host* (PC). Si esta conexión física es una interfaz de comunicación Recommended Standard 232 (RS-232), Recommended Standard 485 (RS-485), Registered Jack 45 (RJ-45), etc., el alambrado está regularmente referido a cables de comunicación.

Las señales eléctricas generadas por los transductores a menudo necesitan ser convertidas a una forma aceptable para el hardware de adquisición de datos. Normalmente se usa un convertidor analógico-digital (ADC, por sus siglas en inglés), el cual convierte la información de la señal al formato digital requerido. Además, muchos transductores requieren alguna forma de excitación o terminación puente para su operación correcta y precisa.

Usualmente, antes de una señal analógica ingrese al ADC, se busca acondicionar la señal. El acondicionamiento de señal implica una o varias tareas. Por ejemplo: filtrado, amplificación, linealización, aislamiento y excitación.

[3] Park J., Steve M., Practical Data Acquisition for Instrumentation and Control Systems, Newnes, Perth, 2003, 17.

El hardware de adquisición y/o control de datos puede ser definido como aquel componente que desempeña las siguientes funciones:

- Procesamiento y conversión de señales analógicas a formato digital. Suele emplearse un ADC.
- Procesamiento de señales digitales. Éstas suelen contener información de un sistema o proceso, en lugar de un fenómeno físico.
- Procesamiento y conversión de señales digitales a formato analógico. Suele utilizarse un convertidor digital-analógico (DAC, por sus siglas en inglés).
- Generación de señales digitales para control.
- Generación de señales analógicas para control.

En la parte final de un SAS, figura 2.1, se encuentra el software de adquisición de datos. Este puede ser hospedado en diferentes arquitecturas. Algunos de uso común son la PC, la Laptop y los sistemas embebidos. El software de adquisición de datos normalmente corre bajo un sistema operativo, éste puede ser multitareas como Windows, multiproceso como VxWorks y en general cualquiera que soporte el programa realizado. El programa puede ser un panel interactivo desplegado en pantalla, un programa dedicado al control de entradas/salidas, un recolector de datos, un manejador de aplicaciones, o una combinación de todos estos.

Hay tres opciones disponibles para programar cualquier hardware de adquisición de datos:

- Programar directamente los registros del hardware de adquisición de datos.
- Utilizar un software de bajo nivel, usualmente viene con el hardware. La idea es poder desarrollar una aplicación para tareas específicas.
- Utilizar software *off-the-shelf*, éste puede ser una aplicación que venga con el mismo hardware y que permita llevar a cabo tareas requeridas para una aplicación particular. Alternativamente se pude usar software de un tercero, tal es el caso de LabVIEW. Éste otorga una interfaz gráfica para programar las tareas requeridas de un elemento de hardware particular, como también brinda herramientas para analizar y desplegar los datos adquiridos.

2.3.1. Conversión analógica-digital

Cuando se habla de conversión analógica-digital, prácticamente se está haciendo referencia a realizar de forma periódica medidas de la amplitud de una señal analógica y traducirlas a un lenguaje numérico. El proceso de conversión consta a grandes rasgos de tres etapas:

1. Muestreo: consiste en tomar muestras periódicas de la amplitud de una onda. La velocidad con que se toman las muestras, es lo que se conoce como frecuencia de muestreo y se encuentra regida por el teorema de Nyquist. Este teorema indica que la frecuencia de muestreo (fs) será el doble de la frecuencia máxima (fm) de la señal a muestrear. Se puede pensar en una señal que tiene un ancho de banda que va de los 20 Hz hasta los 22.5 kHz, su frecuencia máxima sería

de 22.5 kHz, entonces, su frecuencia de muestreo está dada por la siguiente ecuación:

$$fs = 2 \times fm = 2 \times (22{,}500\ Hz) = 45{,}000\ Hz \qquad (2.1)$$

2. Cuantificación: consiste en convertir una sucesión de muestras de amplitud continua en una sucesión de valores discretos. Estos valores han sido previamente establecidos en base a un código. En esta etapa se mide el nivel de tensión de cada una de las muestras obtenidas en la etapa anterior, y se les atribuye un valor finito (discreto) de amplitud. Este valor se selecciona por aproximación dentro de dos niveles previamente fijados. Los valores prestablecidos para ajustar la cuantificación, se eligen en función de la propia resolución que utilice el código seleccionado durante la etapa de codificación. La señal ha quedado representada por valores finitos. La diferencia debida a la aproximación realizada, da origen a lo que se conoce como error de cuantificación.
3. Codificación: consiste en pasar de los valores (finitos) originados en la etapa de cuantificación, al sistema binario, mediante códigos prestablecidos. En la figura 2.2 se puede apreciar una señal senoidal y el resultado de haberla digitalizado con un ADC de 3 bits.

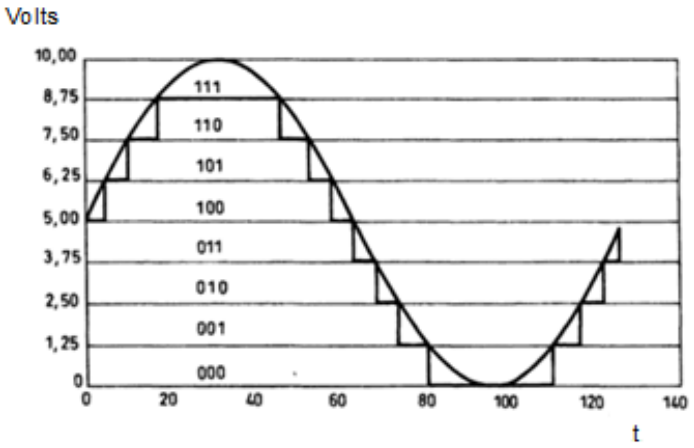


Figura 2.2. Señal senoidal y su versión digital con resolución de 3 bits.

El parámetro de resolución puede ser definido mediante el número de bits con los que cuenta un ADC. Este parámetro determina la precisión con la que se reproduce la señal original. En la figura 2.2, únicamente se tener 3 bits de resolución, por lo que sólo existen 8 valores posibles. Si se tuviera un mayor número de bits, se podría decir que se tiene un convertidor de mayor resolución. Entre más valores se tengan para asignar, una señal puede ser digitalizada con mayor calidad. Evidentemente el número de bits no es el único factor involucrado en la calidad de la digitalización, existen otros factores importantes, tal es el caso del rango de medición.

2.4. Microcontrolador

Los microcontroladores pueden ser considerados como un circuito integrado que contiene microprocesador, memoria y periféricos. La figura 2.3 muestra el diagrama de bloques general de un microcontrolador. En esta figura aparecen otros elementos adicionales a los tres básicos. Estos elementos son: temporizadores, control de interrupciones, perro guardián, buses de direcciones, bus de datos y bus de control. A pesar de que la estructura mostrada involucra más elementos, sigue siendo bastante general.

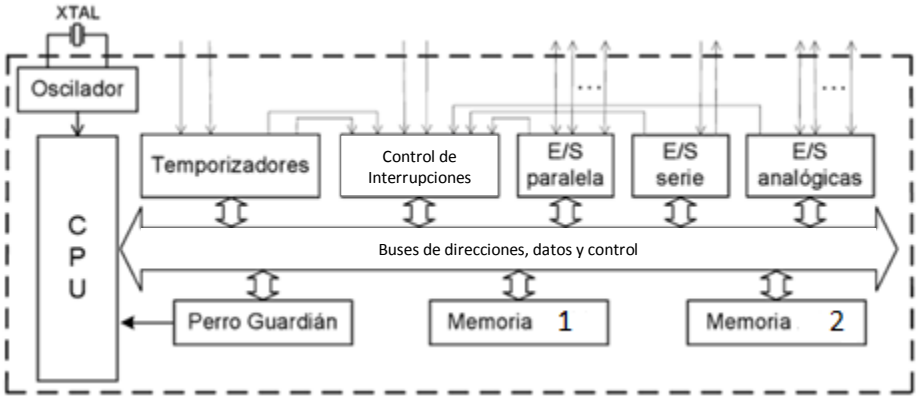


Figura 2.3. Estructura general de un MCU.

Un microprocesador puede entenderse como una unidad central de procesamiento (CPU, por sus siglas en inglés), formada por una unidad aritmética y lógica (ALU, por sus siglas en inglés), una unidad de control de registros y buses que permitan la transferencia de información.

Los microcontroladores disponen de un oscilador que genera los pulsos que sincronizan todas las operaciones internas. El oscilador puede ser del tipo resistor-capacitor (RC), aunque generalmente se prefiere que esté controlado por un cristal de cuarzo (XTAL), debido a la gran estabilidad de frecuencia de éste. La velocidad de ejecución de las instrucciones del programa está en relación directa con la frecuencia del oscilador del microcontrolador.

La CPU es el cerebro del microcontrolador. Esta unidad trae las instrucciones del programa, una a una, desde la memoria donde están almacenadas, las interpreta (decodifica) y hace que se ejecuten. En la CPU se incluyen los circuitos de la ALU para realizar operaciones aritméticas y lógicas elementales con los datos binarios. La CPU dispone de diferentes registros, algunos de propósito general y otros para propósitos específicos. Entre estos últimos están el registro de instrucción (IR, por sus siglas en inglés), el acumulador (ACC, por accumulator), el registro de estado (SR, por sus siglas en inglés), el contador de programa (PC, por sus siglas en inglés), el registro de direcciones de datos (DAR, por sus siglas en inglés) y el puntero de la pila (SP, por sus siglas en inglés).

- El *IR* almacena la instrucción que está siendo ejecutada por la CPU. El IR es inevitable para el programador.
- El *ACC* es el registro asociado a las operaciones aritméticas y lógicas que se pueden realizar en la ALU. En cualquier operación, uno de los datos debe estar en el ACC y el resultado se obtiene en el ACC. La función del ACC puede ser desempeñada por otros elementos dependiendo del microcontrolador, por ejemplo en el PIC estaríamos hablando del registro de trabajo (WR, pos sus siglas en inglés).
- El *SR* agrupa los bits indicadores de las características del resultado de las operaciones aritméticas y lógicas realizadas en la ALU. Entre otros indicadores están el signo del resultado (positivo o negativo), si el resultado es cero, si hay acarreo o préstamo, el tipo de paridad (par o impar) del resultado, etc.
- El *PC* es el registro de la CPU donde se almacenan direcciones de instrucciones. Cada vez que la CPU busca una instrucción en la memoria, el PC contiene la dirección de la instrucción que será ejecutada a continuación. Las instrucciones de transferencia de control modifican el valor del PC.
- El *DAR* almacena direcciones de datos situados en la memoria. Este registro es indispensable para el direccionamiento indirecto de datos en la memoria. El DAR toma diferentes nombres según el microcontrolador.
- El *SP* es el registro que almacena direcciones de datos en la pila.
- La *memoria* del microcontrolador es el lugar donde son almacenadas las instrucciones del programa y los datos que manipula. En un microcontrolador pueden haber varios tipos de memoria: memoria de sólo lectura (ROM, por sus siglas en inglés), RAM, memoria de sólo lectura programable borrable (EPROM, por sus siglas en inglés), memoria de sólo lectura programable borrable eléctricamente (EEPROM, por sus siglas en inglés), FLASH, memoria dinámica de acceso aleatorio síncrona (SDRAM, por sus siglas en inglés), memoria estática de acceso aleatorio (SRAM, por sus siglas en inglés), etc.
- Las entradas y salidas son particularmente importantes en los microcontroladores, pues a través de ellas el microcontrolador interacciona con el exterior. Forman parte de estas los puertos paralelo y serie, los temporizadores y la gestión de interrupciones. El microcontrolador puede incluir también entradas y salidas asociadas a conversión analógica-digital o digital-analógica. Los puertos paralelos se organizan en grupos de varias líneas de entradas y salidas digitales (8 regularmente). Normalmente es posible manipular individualmente las líneas de los puertos paralelos. Los puertos en serie pueden ser de varios tipos, según la norma de comunicación que implementen: RS-232, inter circuitos integrados (I^2C, por sus siglas en inglés), USB, Ethernet, etc.
- Existen otros recursos de gran relevancia que garantizan un funcionamiento seguro del microcontrolador, como el denominado perro guardián.

2.4.1. Arquitecturas von Neumann y Harvard

En la memoria de un microcontrolador se almacenan instrucciones y datos. Las instrucciones deben pasar secuencialmente a la CPU para su decodificación y ejecución, en tanto que algunos datos en memoria son leídos por la CPU y otros son

escritos en la memoria desde la CPU. La organización de la memoria y su comunicación con la CPU son dos aspectos que influyen en el nivel de prestaciones del microcontrolador.

Las arquitecturas von Neumann y Harvard son modelos generales del hardware de los microcontroladores que representan dos soluciones diferentes al problema de la conexión de la CPU con la memoria y la organización de la memoria como almacén de instrucciones y datos.

La arquitectura von Neumann toma el nombre del matemático John von Neumman que propuso la idea de un ordenador con el programa almacenado. Esta arquitectura utiliza una memoria única para instrucciones y datos, figura 2.4. Esto significa que con un mismo bus de direcciones se localizan (direccionan) instrucciones y datos y que por un sólo bus de datos transitan tanto instrucciones como datos. La misma señal de control que emite el CPU para leer un dato sirve para leer una instrucción.

El término Harvard se debe al nombre del lugar donde Howard Aiken diseñó los primeros ordenadores que utilizan memorias separadas para instrucciones y datos, una concepción diferente al ordenador de programa almacenado. Esta arquitectura utiliza memorias separadas para instrucciones y datos, figura 2.4. En este caso la memoria de programa (almacena instrucciones) tiene su bus de direcciones, su propio bus de datos y su bus de control. Por otra parte, la memoria de datos tiene sus propios buses de direcciones, datos y control, independientes de los buses de la memoria de programa. Esta arquitectura tiene la ventaja de una mayor velocidad de ejecución de los programas (más líneas de conexión).

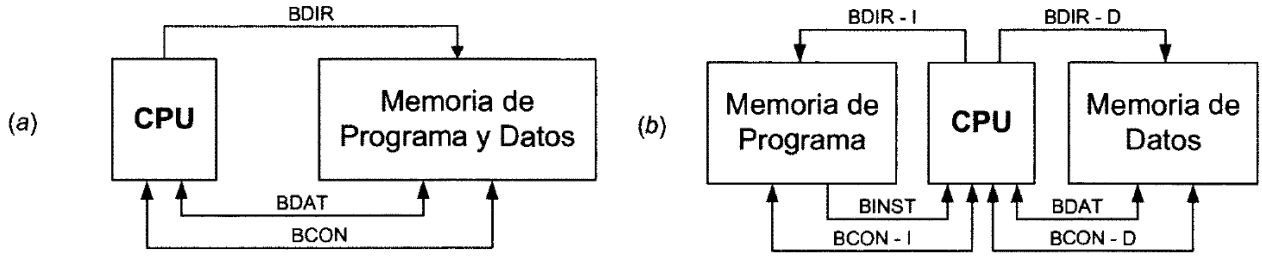


Figura 2.4. a) Arquitectura von Neumann, b) Arquitectura Harvard.

2.4.2. Arquitecturas CISC y RISC

Con base en el repertorio de instrucciones de un microcontrolador es posible realizar una clasificación de éstos. Al primer grupo se le conoce como computadores con un conjunto de instrucciones complejas (CISC, por sus siglas en inglés). Al segundo grupo se le conoce como computadores con un conjunto de instrucciones reducidas (RISC, por sus siglas en inglés). El tipo y cantidad de instrucciones repercute directamente sobre la arquitectura de la CPU.

En la arquitectura RISC, la CPU dispone de un repertorio corto de instrucciones sencillas. Cada instrucción puede realizar una operación muy simple, como mover un dato entre la CPU y la memoria, pero a alta velocidad. Se puede lograr que todas las instrucciones tengan la misma longitud. Hay pocos modos de direccionamiento de los datos y son aplicables a todas las celdas de la memoria de datos. La complejidad de la CPU disminuye, de modo que es fácil aumentar la frecuencia del oscilador de la CPU y con ello la velocidad de las instrucciones. Desde los ochentas, esta ha sido la tendencia predominante en el diseño de microcontroladores.

2.4.3. Microcontrolador Freescale MPC5200

Con un alto nivel de integración y muy rentable el microcontrolador MPC5200 es adecuado para usar en redes, medios de comunicación, control industrial y aplicaciones automotrices. Éste ofrece 760 millones de instrucciones por segundo (MIPS, por sus siglas en inglés), una unidad de punto flotante (FPU, por sus siglas en inglés), unidad de manejo de memoria (MMU, por sus siglas en inglés) para cambiar de tarea rápidamente, está repleto de entradas/salidas (56) y opera con sólo 1 watt. El MPC5200 sirve como compuerta de enlace multimedia en red con procesamiento intensivo, almacenamiento de acceso por red, conexión a internet, receptor de televisión (decodificador), consola de audio para autos, automatización industrial, análisis y detección de imágenes, e instrumentos médicos y electrónicos. Una sólida elección de sistema operativo en tiempo real (RTOS, por sus siglas en inglés) y un paquete de software base (BSP, por sus siglas en inglés) para compatibilidad con el RTOS otorgan a los usuarios un conjunto completo y flexible de soluciones. El diagrama de bloques simplificado del MPC5200 se presenta en la figura 2.5.

La FPU, también conocida como coprocesador matemático, es un componente de la unidad central de procesamiento especializado en el cálculo de operaciones en punto flotante. Las operaciones básicas que toda FPU puede realizar son la suma y multiplicación usuales, algunos sistemas más complejos son capaces también de realizar cálculos trigonométricos o exponenciales.

La MMU es un dispositivo de hardware formado por un grupo de circuitos integrados, responsable del manejo de los accesos a la memoria por parte del CPU. Entre sus funciones se encuentran la traducción de las direcciones lógicas (virtuales) a direcciones físicas (reales), la protección de la memoria, el control de *caché* y conmutación de banco, que es una técnica para aumentar la cantidad de memoria utilizable más allá de la cantidad directamente *direccionable* por el procesador, sin tener que ampliar el bus de direcciones.

Bloques del MPC5200 de interés

Los controladores de memoria Double Data Rate (DDR) son utilizados para controlar DDR SDRAM, donde la información es transferida en el flanco de subida y el de bajada del reloj de la memoria del sistema. Estos controladores de memoria permiten duplicar la cantidad de datos a transferir sin aumentar la frecuencia de reloj o el tamaño del bus que va a la celda de memoria.

El BestComm es un controlador inteligente de acceso directo a memoria (DMA, por sus siglas en inglés). Desde la liberación del reset, el BestComm está inactivo. Tiene su propio circuito de control que ejecuta su propio conjunto de instrucciones. Cuando el sistema arranca, el programa de carga descarga el código de programa del BestComm en la SRAM interna del BestComm. Luego, el programa de carga inicializa todos los registros para la adecuada configuración del BestComm.

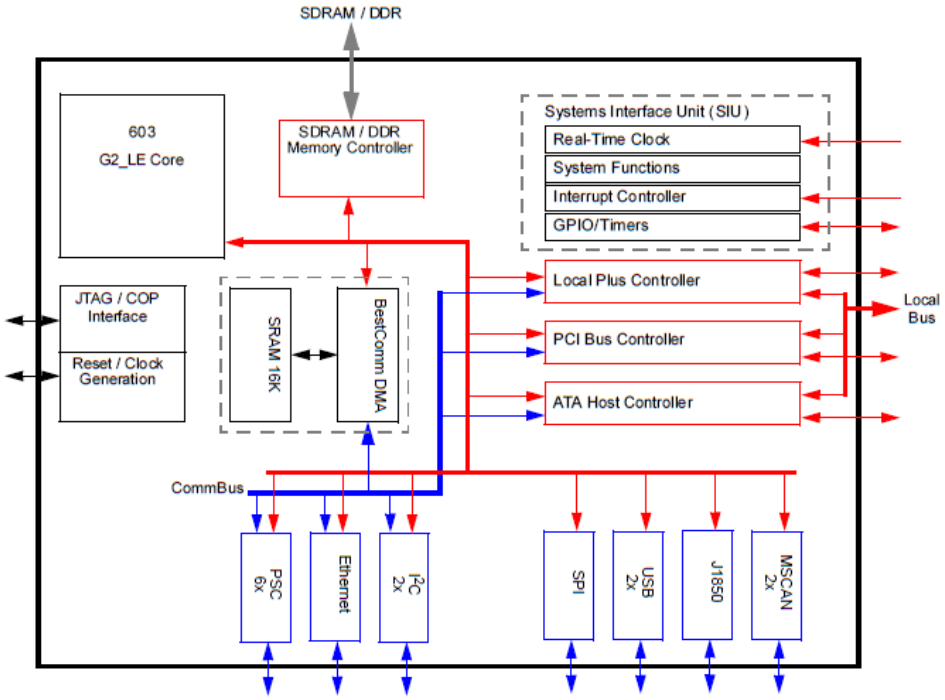


Figura 2.5. Diagrama de bloques simplificado del MPC5200[4]

El BestComm puede transferir información hacia o desde la memoria SDRAM y del bus LocalPlus con la mayoría de los módulos periféricos, tales como controladores de comunicación serie programables (PSC, por sus siglas en inglés), I^2C, y Ethernet.

En el sistema cRIO la forma de comunicación entre el MPC5200 y el FPGA es a través de un bus PCI. El MPC5200 cuenta con un módulo dedicado para efectuar comunicación mediante PCI. Este módulo se llama PCI Bus Controller. Al ser un módulo dedicado realiza un conjunto de tareas por sí sólo, con lo cual el microcontrolador dedica menos recursos para la comunicación. PCI Bus Controller permite trabajar a 33 y 66 MHz. Maneja 32 bits para el bus de direcciones/datos. A pesar de que está diseñado para la versión 2.2 de PCI, el módulo puede trabajar con otras versiones.

De manera similar al PCI Bus Controller, existe otro módulo dedicado y que es de gran utilidad para el sistema del cual es objeto esta tesis. Se está hablando del módulo Ethernet. Como su nombre lo indica éste está diseñado para realizar tareas que

[4] Freescale Semiconductor, Datasheet MPC5200, fourth revision, 2005, 5.

tengan que ver con la comunicación Ethernet. El módulo puede trabajar con tres estándares diferentes IEEE 802.3 a 100 megabits por segundo (Mbps, pos sus siglas en inglés) con interfaz independiente, IEEE 802.3 10 Mbps con interfaz independiente y 10 Mbps con interfaz de 7 alambres.

Otro módulo dedicado de interés es el USB. Éste únicamente está diseñado para funcionar como *host*. Fue fabricado para la versión 1.1 de USB pero puede trabajar con versiones superiores; se encuentra limitado a la velocidad de transferencia de la versión 1.1. Este módulo puede trabajar con dos puertos simultáneamente, pero la arquitectura de cRIO sólo hace uso de un puerto.

Por último se tiene a la unidad de interfaz de sistemas (SIU, por sus siglas en inglés). Esta unidad cuenta con conexiones de propósito específico y general, éstas permite comunicar al MCP5200 con el exterior. Es de especial interés las conexiones que permiten interactuar con el reloj de tiempo real interno. Este último permite prescindir de una señal externa, necesaria para temporizar los programas de la presente tesis.

2.5. FPGA

En el mundo de los ordenadores y la electrónica, se ha usado dos formas diferentes de llevar a cabo cómputo: hardware y software. El cómputo por software brinda la flexibilidad para modificar aplicaciones y desempeñar un gran número de diferentes tareas, pero en desempeño, área eficiente de silicio y uso de energía es superado por los circuitos integrados de aplicación específica (ASIC, por sus siglas en inglés).

Los FPGAs son dispositivos realmente revolucionarios que mezclan los beneficios de ambas formas de hacer cómputo, hardware y software. Sin embargo, fusionando los beneficios de ambos se tiene un precio que pagar. Los FPGAs brindan casi todos los beneficios de ambos pero no todos. Comparado con los microprocesadores, los FPGAs son normalmente más rápidos y eficientes en energía, pero crear programas eficientes para los FPGAs es más complejo. Típicamente los FPGAs son útiles sólo para aplicaciones que procesan grandes flujos de datos, tales como procesamiento de señales, creación de redes, etc. Comparados con los ASICs, los FPGAs ocupan de 5 a 25 veces más área, tienen más retrasos y su desempeño es menor. Sin embargo un diseño de ASIC puede tomar de meses a años y mucho dinero, mientras que un diseño de FPGA puede tomar días y menor cantidad de dinero.

La figura 2.6 muestra de manera general la estructura interna de un FPGA, el cual está compuesto de bloques lógicos en una estructura de *ruteo* general. Los bloques lógicos contienen elementos de procesamiento para llevar a cabo lógica *combinacional* simple, como también flip-flops para implementar lógica secuencial. Porque las unidades lógicas son a menudo simples memorias, cualquier combinación *booleana* función de tal vez seis o cinco entradas puede ser implementada en cada bloque lógico. La estructura de *ruteo* general permite alambrar arbitrariamente, así los elementos lógicos pueden ser conectados de la manera deseada. Con las unidades

lógicas fijas, las cuales son fabricadas en silicio, los FPGAs son capaces de implementar sistemas completos en un único dispositivo programable.

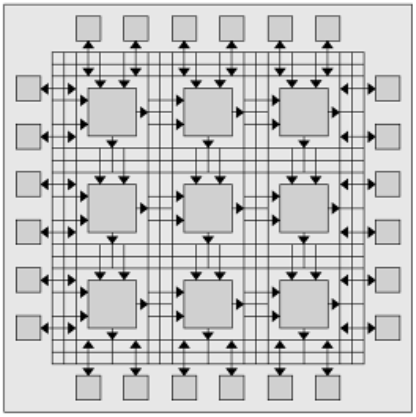


Figura 2.6. Una vista abstracta de un FPGA[5].

En términos muy generales, hay solamente dos tipos de recursos en un FPGA: lógicos y de interconexión. Los recursos lógicos son los que permiten realizar cosas como aritmética (1+1 = 2) y funciones lógicas (if [ready] x=1 else x=0). Los recursos de interconexión son los que permiten llevar los datos desde un nodo de cómputo a otro, por ejemplo, llevar el resultado de un cálculo de un nodo a otro.

Debido a que el FPGA combina la flexibilidad del software con el desempeño del hardware, un diseñador FPGA debe pensar diferente a los diseñadores que usan otros dispositivos. Los desarrolladores de software generalmente escriben programas secuenciales que explotan la habilidad del microcontrolador, para pasar rápidamente a través de una serie de instrucciones. En contraste, un diseñador FPGA de alto nivel requiere pensar sobre paralelismo espacial, esto es, usar simultáneamente múltiples recursos esparcidos a lo largo y ancho del circuito integrado para llevar a cabo un gran número de operaciones.

2.5.1. Elementos lógicos

Cualquier cómputo puede ser representado como una ecuación *booleana*. A su vez, cualquier ecuación booleana puede ser expresada como una tabla de verdad. Desde estos principios, podemos construir estructuras complejas que puedan realizar aritmética, tales como sumadores o multiplicadores, también como estructuras de toma de decisiones que pueden evaluar sentencias condicionantes. Combinando esto, es posible describir elaborados algoritmos con el simple hecho de usar tablas de verdad.

De las observaciones anteriores es posible ver la tabla de verdad como el corazón del cómputo en un FPGA. Un elemento de hardware que puede fácilmente implementar una tabla de verdad es la Look Up Table (LUT). Desde una perspectiva de

[5] Hauck, S., DeHon A., Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation, first edition, Morgan Kaufmann, United State, 2008, xxvi.

implementación de un circuito, una LUT puede ser formada por un multiplexor n:1 (n a uno) y una memoria de n bits. La figura 2.7 muestra una LUT típica de n entradas que puede ser encontrada en los FPGAs actuales. La imagen de la izquierda es la generalidad, la imagen de la derecha corresponde a una operación XOR. Casi todos los FPGAs comerciales tienen establecido como bloque de construcción básico la LUT.

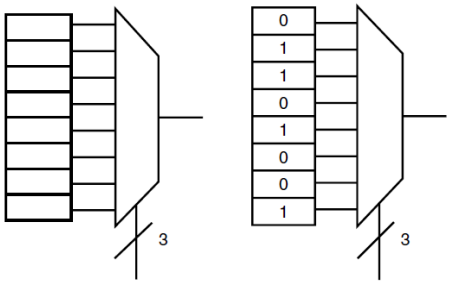


Figura 2.7. Esquemático de una LUT de tres entradas.

La LUT puede computar cualquier función de N entradas por simple programación de la misma, con una tabla de verdad correspondiente a la función que deseamos implementar.

Estudios empíricos actuales han demostrado que la estructura 4-LUT (LUT de cuatro entradas) hace el mejor equilibrio entre área de silicio y retraso en tiempo para un amplio rango de circuitos de prueba. Por supuesto, el cómputo basado en FPGA evoluciona y obliga a actualizar los diseños todo el tiempo. Xilinx ha liberado el FPGA Virtex-5 basado en SRAM con arquitectura 6-LUT.

La pregunta del número de LUTs por bloque lógico también ha sido investigada, evidencia empírica sugiere que grupos de varias 4-LUT dentro de un bloque lógico puede mejorar el área de silicio y el retraso en tiempo.

Con solamente LUTs, no hay manera de que un FPGA pueda mantener cualquier sentido de estado, entonces, es imposible implementar cualquier lógica secuencial o de estados. Para remediar esta situación, se agrega un elemento que permita el almacenamiento de un bit en el bloque lógico básico, este elemento tiene forma de un flip-flop tipo D. Ahora el bloque lógico debe verse algo parecido a la figura 2.8. La salida *multiplexada* selecciona un resultado de la función implementada por la LUT o del bit almacenado en el *flip-flop* tipo D.

En la figura 2.8 es fácil de identificar los puntos programables, esto incluye el contenido de la 4-LUT, el selector de señal para la salida del multiplexor y el estado inicial del *flip-flop* tipo D. La mayoría de los FPGAs comerciales usan bits SRAM conectados a puntos de configuración para configurar el FPGA. Escribiendo un valor para cada uno de los bits de configuración se prepara al FPGA para llevar a cabo la función solicitada.

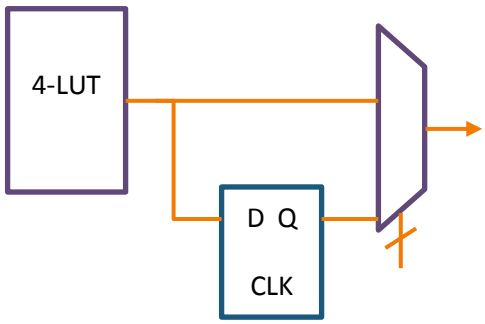


Figura 2.8. Bloque lógico LUT.

2.5.2. El arreglo y la interconexión

Se ha definido lo que normalmente se conoce como bloque lógico o bloque de función en un FPGA. Ahora es necesario explicar cómo se distribuyen y conectan los bloques lógicos para formar la estructura de un FPGA.

Los FPGAs actuales implementan lo que normalmente se llama arquitectura island-style. Como se muestra en la figura 2.9, este diseño tiene bloques lógicos distribuidos en un arreglo de dos dimensiones e interconectados de alguna manera. Los bloques lógicos forman las islas y flotan en un océano de interconexiones.

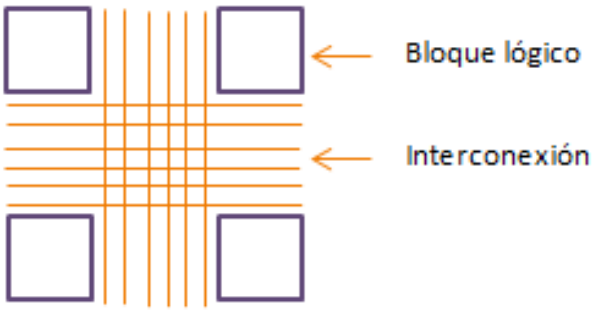


Figura 2.9. La arquitectura island-style de un FPGA.

Cómputos muy extensos son divididos en piezas de dimensión 4-LUT y mapeados en los bloques lógicos físicos en el arreglo. Las interconexiones son configuradas para *rutear* señales entre bloques lógicos de una manera adecuada. Con suficientes bloques lógicos y conexiones eficientes es posible llevar a cabo cualquier cómputo deseado en un FPGA.

Estructura de interconexión

La estructura de interconexión más usual es la nearest-neighbor, ésta sólo necesita conexiones en cuatro direcciones: norte, sur, este y oeste. Esto permite que cada bloque lógico se pueda comunicar directamente con cada uno de sus vecinos inmediatos (ver figura 2.9). Aunque está estructura ofrece una solución de conexión, sufre de muchos retrasos y problemas de conectividad (superposición de señales). Es

necesario implementar algo para que exista la posibilidad de saltar bloques lógicos, la figura 2.10 nos muestra una estructura más robusta que mejora la conectividad en el FPGA. Con el bloque de conexión (CB, por sus siglas en inglés) y el bloque de conmutación (SB, por sus siglas en inglés), la arquitectura island-style soportar estructuras de conexión más complejas.

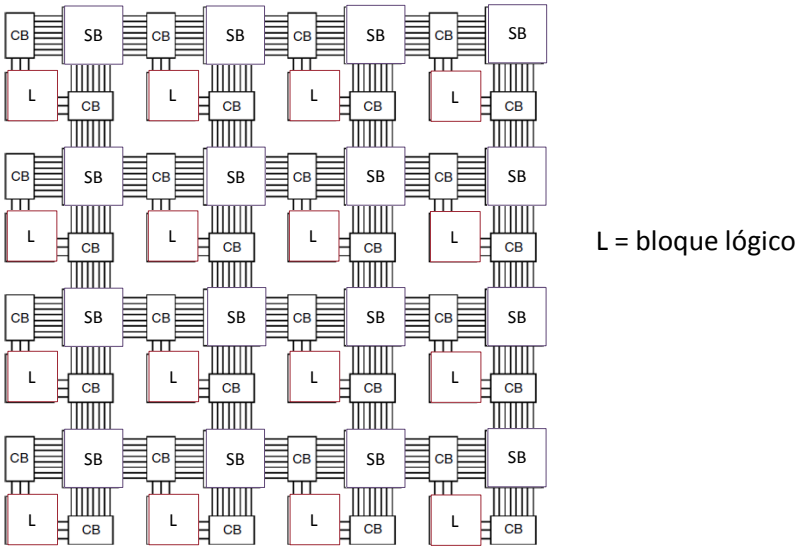


Figura 2.10. Arquitectura island-style compleja.

Segmentado

En la figura 2.10 se ha incorporado a CB y a SB, esta estructura es más genérica. El bloque lógico que se muestra en la misma figura, accesa a sus recursos de comunicación cercanos a través del CB, el cual conecta las entradas y salidas del bloque lógico para enlazar recursos a través del SB (es programable) o con multiplexores. El CB mostrado en la figura 2.11 con mayor precisión, permite asignar arbitrariamente pistas horizontales y verticales a las salidas y entras del bloque lógico, incrementando la flexibilidad del *ruteo*. El SB mostrado en la figura 2.12, se encuentra en el punto donde las pistas de conexión horizontales y verticales convergen, en el caso más general, este es una simple matriz de interruptores programables que permiten a una señal conectarse de una pista a otra.

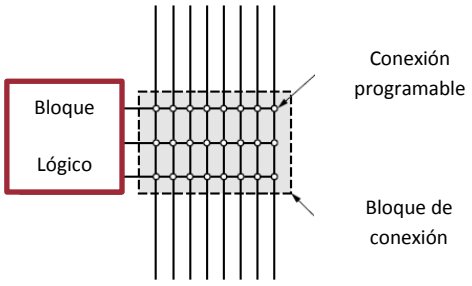


Figura 2.11. Bloque de conexión a detalle.

Es importante observar que seguirán existiendo retardos, sin embargo el hecho de que exista el CB y el SB permite separar la interconexión de la lógica, de tal forma que el *ruteo* sea realizado sin consumir recursos de los bloques lógicos.

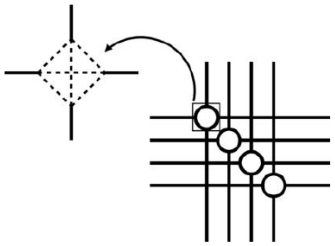


Figura 2.12. Arquitectura del bloque de conmutación.

Jerarquía

Un enfoque ligeramente diferente para reducir los retrasos de largas conexiones es usar un enfoque jerárquico, ilustrado en la figura 2.13. En el nivel más bajo de la jerarquía, existen matrices de 2x2 donde los elementos son bloques lógicos que están agrupados. Dentro de este bloque, local, la conexión nearest-neighbor es todo lo que está disponible. A su vez, un grupo de 2x2 de estas matrices de 2x2 conforman un grupo de 16 bloques lógicos. En este nivel de la jerarquía, conexiones largas en la frontera de los más pequeños (grupos de 2x2), conectan cada grupo de cuatro bloques lógicos a otro grupo en una jerarquía superior. Esto se repite en niveles más altos de la jerarquía, con grupos más grandes y conexiones más largas.

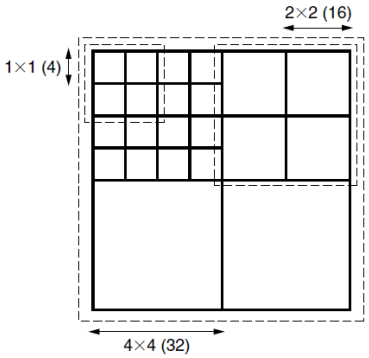


Figura 2.13. Jerarquía de conexión para enlaces largos.

El patrón de interconexión descrito considera que un buen diseño (bien mapeado) de circuito tiene la mayoría de sus conexiones locales, y solamente un número limitado de conexiones que necesitan viajar largas distancias. Esta arquitectura de interconexión hace uso eficiente del área de silicio mientras preserva algunos enlaces largos para minimizar el retraso de las señales que necesitan cruzar largas distancias.

Programación

Como en los bloques lógicos en un FPGA comercial típico, cada punto de conmutación en la estructura interconectada es programable. Dentro de los bloques de conexión, multiplexores programables seleccionan que ruta deben tomar las terminales de entradas y salidas; en los bloques de conmutación, la unión entre rutas verticales y horizontales es conmutada a través de un interruptor programable; finalmente, el cambio entre pistas de diferentes niveles de jerarquía es llevado también a través de interruptores programables.

Pata todos los puntos programables, como en los bloques lógicos, los FPGAs modernos usan bits SRAM para mantener los valores de configuración definidos por el usuario.

2.5.3 Lógica extendida

Los FPGAs modernos incorporan elementos lógicos básicos que se han ido incrementando. La idea de esto es aumentar el desempeño de operaciones comunes como son el cálculo aritmético y el almacenamiento de datos.

Cadena de acarreo rápido

Una operación básica que se necesita en el FPGA es la suma. Obviamente esta operación puede ser implementada con el uso de 4-LUT. La problemática de esta operación no viene dada por su complejidad para programarla, está dada por la variación de la señal de acarreo que va de los bits de menor orden a los bits de mayor orden. Como se muestra en la figura 2.14, esta señal se origina de las señales primarias de menor orden, va a través de los bloques lógicos, sale a la interconexión, entra en el bloque lógico adyacente, y así sucesivamente. Los retrasos son acumulados en cada punto de conmutación a lo largo del camino.

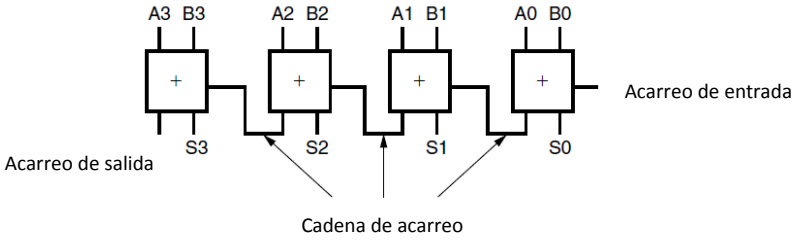


Figura 2.14. Sumador completo de 4 bits.

Una manera de incrementar la velocidad es generando un acceso directo a la cadena de acarreo entre bloques lógicos adyacentes. Se puede complementar esto creando una ruta conmutada dedicada y que mínimamente conecte la salida del bloque lógico que genera la señal de acarreo al bloque lógico adyacente de mayor orden. Esta cadena de acarreo no va a tener que ser interconectada en la red de interconexión general. Se incrementa dramáticamente la velocidad de la adición pero se generan restricciones en la disposición espacial de la adición de múltiples bits.

Cabe resaltar que el éxito de esta optimización recae en la habilidad de la herramienta de configuración para identificar sumas en la descripción de circuito del diseñador y luego usar la lógica dedicada.

Multiplicadores

Al igual que la suma, la multiplicación es muy requerida en la mayoría de algoritmos. Muchas implementaciones (diseños probados) están disponibles si se quiere usar los bloques lógicos con los que cuenta el FPGA para construir multiplicadores. Mientras que sin duda se puede implementar una multiplicación, solamente se puede hacer con un gran retraso en el tiempo o con una cantidad grande de bloques lógicos, lo cual dependerá de la eficiencia de la implementación. En esencia, los bloques lógicos no son muy eficientes para desarrollar multiplicaciones.

En lugar de hacer las multiplicaciones en bloque lógicos, existe la opción de tener multiplicadores por hardware, que se comuniquen con la estructura general del FPGA. Según los diseñadores VLSI (Very Large Scale Integration) la implementación de multiplicadores hechos de transistores usando la técnica que sea, permite usar menos área de silicio, se consigue mayor velocidad y se consume menor potencia que usando LUTs.

Por supuesto, como en el caso de las cadenas de acarreo rápido, los multiplicadores imponen consideraciones de diseño y restricciones físicas, pero se agrega una opción más para cómputo en la paleta de operaciones. Después de lo anterior sólo queda un buen diseño y excelentes herramientas para hacer eficiente el diseño.

RAM

Otra área de mejora que se viene buscando desde hace tiempo es el almacenamiento de datos en un solo circuito integrado. Las arquitecturas actuales de FPGAs cuentan con cantidades generosas de RAM en circuito integrado, que puede ser accedida desde la arquitectura general del FPGA.

Las celdas SRAM son extremadamente pequeñas y al estar físicamente distribuidas en el FPGA, pueden ser muy útiles para muchos algoritmos. Agrupando muchas SRAM en bancos de memoria, los diseñadores pueden implementar grandes ROMs para cálculos con LUTs y operaciones con coeficientes constantes a velocidades altas, además de grandes RAMs para *buffers* y colas. Lo anterior saca ventaja de una buena estrategia de sincronización y la velocidad ganada a raíz de no tener que comunicarse con una memoria externa. Actualmente los FPGAs cuentan con RAM dedicada que van desde los kilobits hasta los megabits.

Boques de procesadores

En un sentido general, los FPGA son bastante eficientes al implementar varios niveles de pipeline, manejando tamaños de palabra no estandarizados, y

proporcionando información y procesos en paralelo. La inclusión de bloques de procesadores reconoce el hecho de que el flujo de algunos algoritmos es procedural y contiene un alto grado de ramificación, lo que no permite la aceleración de los mismos usando FPGAs.

En los FPGAs de gama alta es posible encontrar bloques enteros de procesadores. Estos bloques son capaces de trabajar por encima de los 300 MHz, incluir FPU, correr sistemas operativos embebidos completos y algunos incluso pueden reprogramar la estructura del FPGA que se encuentra a su alrededor.

La idea general de los elementos considerados como lógica extendida es la de poder sostener funcionalidad critica, que no puede ser llevada a cabo eficientemente en la estructura base de un FPGA.

2.5.4. Configuración

Unas de las características que definen al FPGA es su capacidad para actuar como un “hardware en blanco” para el usuario final. Cada elemento configurable en el FPGA requiere 1 bit de almacenamiento para mantener la configuración definida por el usuario. Para un FPGA basado en LUT, estas localidades programables generalmente incluyen el contenido del bloque lógico y la conectividad de la estructura de *ruteo*. La configuración del FPGA es llevada a cabo a través de programar los bits de almacenamiento conectados a estas localidades programables, de acuerdo a la definición del usuario. Para las LUTs, esto se traduce en llenarlas con unos y ceros. Para la estructura de *ruteo*, la programación habilita y deshabilita interruptores a lo largo de las pistas.

La configuración puede ser pensada como un archivo binario plano, el cual contiene un mapa, bit por bit, de los bits programables en el FPGA. Este archivo es normalmente llamado bitstream *y* es generado por las herramientas específicas del vendedor. Una vez generado el bitstream un diseño de hardware ha finalizado. A pesar de que la forma exacta del bitstream no ha sido publicada, entre más grande sea el FPGA el bitstream es más grande.

Debido a que la configuración de un FPGA meramente involucra almacenamiento de valores en localidades de memoria, similarmente a compilar y luego cargar un programa en un microcontrolador, la creación de un circuito basado en FPGA es el proceso “simple” de crear un bitstream para cargarlo dentro del dispositivo. En la figura 2.15 se esboza el proceso para llegar al bitstream.

Aunque hay herramientas para generar el bitstream desde lenguajes de alto nivel (software), esquemáticos, y otros formatos, los diseñadores FPGA normalmente empiezan con una aplicación escrita en lenguaje de descripción de hardware (HDL, por sus siglas en inglés) tales como Verilog o VHDL. Este diseño abstracto es optimizado para ajustarlo a la lógica disponible en el FPGA, a través de una serie de pasos: ‘Síntesis de la lógica’ convierte el código con construcciones y comportamientos lógicos de alto nivel en compuertas lógicas, seguido por ‘Tecnología de mapeo’, que se

encarga se separar las compuertas en grupos que se ajusten mejor a los recursos lógicos del FPGA. Después, *'*Colocación*'* asigna los grupos lógicos para especificar los bloques lógicos y '*Ruteo*' determina los recursos de interconexión que van a llevar (guiar) las señales del usuario. Finalmente, 'Generación de Bitstream' crea un archivo binario que configura todos los puntos programables del FPGA para configurar los bloques lógicos y conectar los recursos apropiadamente.

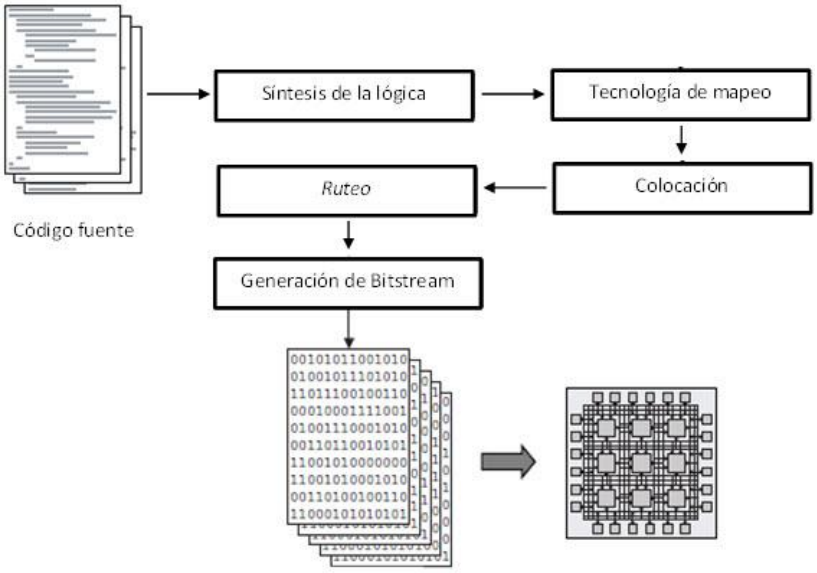


Figura 2.15. Flujo típico para lograr configura un FPGA.

Después que un diseño ha sido compilado, se pude programar el FPGA para que desarrolle una función específica, descargando el bitstream en el dispositivo. Normalmente un microcontrolador o un microprocesador descargan el bitstream al dispositivo, o una EEPROM programada con el bitstream es conectada al puerto de configuración del FPGA.

Actualmente hay muchos métodos conocidos para almacenar un solo bit de información binaria. Los métodos más populares se abordan a continuación.

SRAM

El método más ampliamente usado para almacenamiento de información de configuración en FPGAs comerciales es SRAM. Este método se ha vuelto popular debido a que provee velocidad e infinita reconfiguración en una tecnología bien conocida.

Los inconvenientes de SRAM vienen dados por los consumos de potencia y por la volatilidad de los datos. Comparada con otras tecnologías, las celdas SRAM son grandes (6 a 12 transistores) y disipa una considerable potencia. Otro inconveniente importante es que la SRAM no puede mantener su contenido sin energía, lo que significa que al prender el FPGA éste no está configurado y debe ser programado

usando lógica y almacenamiento que no pertenecen al circuito integrado. Eso puede ser complementado con una memoria de almacenamiento no volátil para sostener la configuración y un microcontrolador para llevar a cabo el procedimiento de programación.

Memoria Flash

Aunque es menos popular que SRAM, muchas familias de dispositivos usan memoria Flash para mantener la información de la configuración. Las principales diferencias con SRAM son que la memoria Flash es no volátil y que solamente pueden ser escritas un número finito de veces. Esta tecnología no necesita almacenamiento extra o hardware para programar en el arranque. En esencia, un FPGA basado en Flash puede estar listo inmediatamente.

Una celda de memoria Flash puede estar hecha con menos transistores en comparación con una celda SRAM. Este diseño puede tener menos consumo de potencia debido a que hay menos transistores operando.

Las memorias Flash tienen un número de ciclos limitados de escritura y a menudo tiene velocidades de escritura más lentas que SRAM. El número de escrituras puede variar según la tecnología, pero típicamente es de cientos de miles a millones. Adicionalmente, la mayoría de técnicas de escritura requieren voltajes más grandes comparados con otros circuitos y se requiere de hardware adicional para preparar la escritura en Flash.

Antifusible

Un tercer enfoque para alcanzar la programación es la tecnología *antifusible*. Ésta se basa en enlaces hechos de metal que tienen un comportamiento contrario aún fusible. El enlace *antifusible* normalmente está abierto (no conectado). Un proceso de programación involucra altas corrientes o un láser derrite el enlace para formar una conexión eléctrica, en esencia se crea un alambre o un atajo entre las terminales del *antifusible.* Esta tecnología tiene muchas ventajas pero una clara desventaja, no es reprogramable. Una vez que el enlace se funde, este ha sufrido una transformación que no es posible revertir. FPGAs basados en esta tecnología son generalmente considerados one-time programmable (OTP).

El enlace puede ser muy pequeño, comparado con la celda multitransistor que representa una SRAM, no se requiere de ningún transistor. Lo anterior resulta en retrasos muy pequeños, cero consumo de potencia y no son susceptibles a alta radiación de partículas que inducen errores. Todo esto hace a la tecnología candidata para aplicaciones militares y espaciales.

2.5.5. FPGA Virtex-5 LX50

La figura 2.16 muestra un bloque lógico Virtex-5, el cual está referido como un bloque lógico configurable (CLB, por sus siglas en inglés), el cual comprende dos

conjuntos de LUTs y flip-flops que comparten conexiones, con la idea de utilizar una cadena de acarreo rápido. A cada conjunto descrito se le conoce como slice. Además de las slices en la figura 2.16, se pude apreciar una matriz de conmutación. Como se muestra, cada slice contiene cuatro LUTs de seis entradas (6-LUTs) y cuatro flip-flops. Las LUTs en un Virtex-5 son verdaderas LUTs que han sido construidas con LUTs más pequeñas y pueden combinarse con la ayuda de multiplexores.

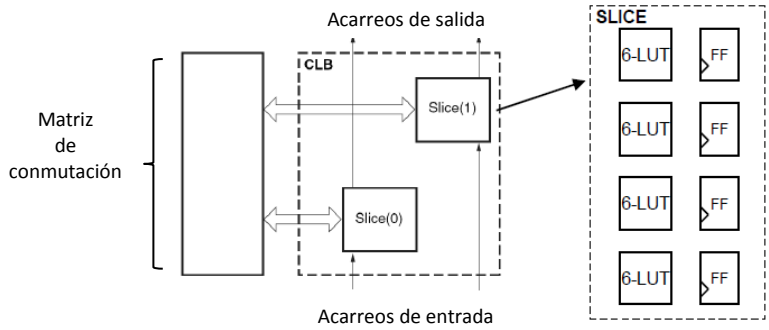


Figura 2.16. Virtex-5 CLB y slice.

Tradicionalmente, una LUT tiene una sola salida, sin embargo, las LUTs en Virtex-5 tienen dos salidas. Las LUTs pueden implementar una sola función lógica con un máximo de 6 entradas, la señal de salida aparecerá en una de las dos salidas. Alternativamente, la LUT puede implementar dos funciones que juntas pueden usar hasta 5 entradas. En este caso, ambas salidas son usadas, la entrada sobrante en la LUT debe estar ligada a lógica negada. Notar en la figura 2.17 que cuando las dos salidas de la LUT están en uso, solamente una de ellas puede ser registrada.

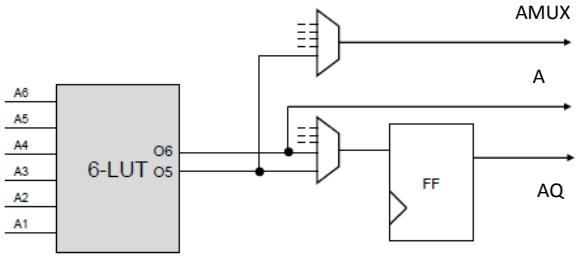


Figura 2.17. LUT de seis entradas con doble salida.

Una observación clave es que una de la dos salidas (O6) maneja una salida directa del slice, definida con la letra A en la figura; mientras que la otra (O5) debe pasar a través de un multiplexor adicional antes de alcanzar la salida de la slice, AMUX. Esto hace a O5 más lenta que O6. Consecuentemente, para altos desempeños, O5 no debe ser usada en pistas *combinacionales* de tiempo crítico.

Virtex-5 está basado en una tecnología de 64 nm, lo que lo hace una alternativa programable al uso de tecnología ASIC. Cuenta con 560 terminales de entrada/salida de las que puede disponer el usuario, con ello supera el promedio de entradas y salidas de un MCU comercial.

Bloques RAM y DSP

La figura 2.18 (a) muestra un bloque RAM del Virtex-5, el cual puede almacenar hasta 36 kbits de información. Un bloque de RAM puede operar como una simple RAM de 36 kbits, o como dos RAMs independientes de 18 kbits. Las RAMs tienen una relación de aspecto reconfigurable, una RAM puede ser configurada como 16 k x 1, 8 k x 2, 2 k x 9, o 1 k x 18. Como se muestra en la figura 2.18, una RAM de 36 kb puede ser vista como si tuviera dos ranuras dentro de ella. La escritura y la lectura del bloque de RAM es una operación asíncrona; cada bloque RAM tiene puestos independientes de rescritura y lectura. Múltiples bloques RAM son organizados en columnas en el Virtex-5 y dos bloques RAM adyacentes pueden combinarse para implementar memorias más grandes.

La figura 2.18 (b) representa el bloque de un procesador de señales digitales (DSP, por sus siglas en inglés) de un Virtex-5 llamado DSP48E. En general, el DSP48E comprende un multiplicador de 25 x 18, el cual recibe las entradas A y B. El resultado de la multiplicación se introduce en una ALU que puede implementar adición, substracción y funciones lógicas variadas. El resultado de las operaciones realizadas es entregado a través de la salida P. La operación de multiplicar dos números y el producto sumarlo al acumulador (MAC), dominante en los circuitos DSP, puede ser realizada en un DSP48E. Múltiples DSP48E pueden ser enlazados para formar funciones más complejas a través de los puertos PCIN , PCOUT, ACOUNT, BCOUNT, ACIN, BCIN mostrados en la figura 2.18. Los registros configurables (para pipeline) y el *ruteo* de la circuitería están presentes en varios lugares del bloque.

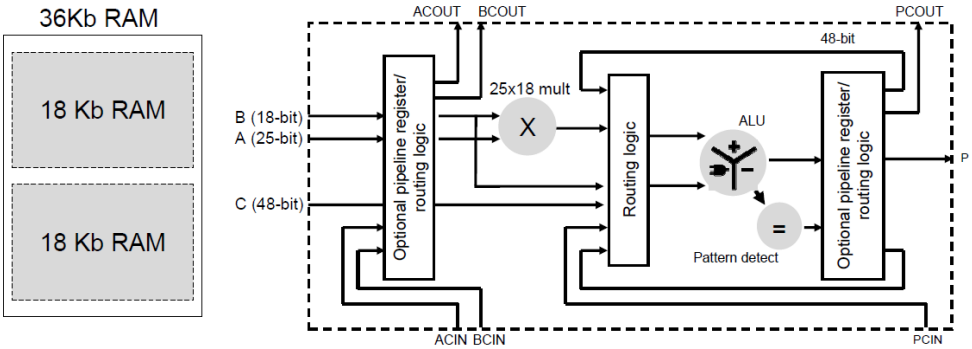


Figura 2.18. (a) Bloque RAM y b) bloque DSP del Virtex-5[6].

La distribución del Virtex-5 contiene columnas de bloques RAM y columnas de bloques DSP48E, integrados en la estructura regular de los bloques lógicos y la interconexión.

[6] XILINX, Virtex-5 FPGA XtremeDSP Design Considerations, fifth revision, 2009, 14.

2.6. Bus serie universal

El bus serie universal es mejor conocido como USB (por sus siglas en inglés), éste es un estándar industrial desarrollado en los años 90', que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre ordenadores y periféricos. El desarrollo de este estándar está respaldado por un conjunto de empresas como Intel, IBM y Microsoft. Actualmente el grupo de desarrollo está conformado por más de 685 compañías. El campo de aplicación del USB se extiende actualmente a cualquier dispositivo electrónico. Se han diseñado variaciones para uso exclusivamente industrial y militar.

Algunos dispositivos requieren una potencia mínima, así que se pueden conectar varios sin necesitar fuentes de alimentación adicionales. Existen concentradores con fuente de alimentación que pueden proporcionarle corriente eléctrica a otros dispositivos sin quitarle corriente al resto de la conexión.

En el caso de los discos duros, sólo una selecta minoría implementa directamente la interfaz USB como conexión nativa, siendo los discos externos mayoritariamente Integrated Drive Electronics (IDE) o serial Advanced Technology (ATA) con un adaptador en su interior.

Velocidad de transmisión

Existen cuatro tipos USB, si se clasifican respecto a su velocidad de transferencia de datos:

Baja velocidad (1.0). Su tasa de transferencia puede llegar a 1.5 Mbits/s. Utlizado en su mayor parte por dispositivos de interfaz humana como teclados, los ratones las cámaras web, etc.

Velocidad completa (1.1). Tasa de transferencia de hasta 12 Mbits/s. Esta fue la más rápida antes de la especificación USB 2.0, y muchos dispositivos fabricados en la actualidad trabajan a esa velocidad. Estos dispositivos dividen el ancho de banda de la conexión USB entre ellos, basados en un algoritmo de impedancias Last In First Out (LIFO).

Alta velocidad (2.0). Tasa de transferencia de hasta 480 Mbits/s pero por lo general de hasta 125 Mbits/s. El cable USB 2.0 dispone de cuatro líneas, un par para datos y otro par de alimentación.

Súper alta velocidad (3.0). Tiene una tasa de transferencia de hasta 4.8 Gbits/s. La velocidad del bus es diez veces más rápida que la del USB 2.0, debido a que han incluido 5 contactos adicionales, y es compatible con los estándares anteriores.

Los niveles de transmisión de la señal para las versiones 1.0 y 1.1 varían de 0 a 0,3 [V] para nivel bajo y de 2.8 a 3.6 [V] para nivel alto; en el caso de las versiones 2.0

y 3.0, cuando se trabaja a altas velocidades, se puede tener una variación de ± 400 [mV].

Con respecto al suministro de energía el USB 3.0 puede entregar 1 [A] a 5 [V] por puerto, lo cual nos permite trabajar con 5 [W], en contraste con los otros casos que solo pueden entregar 500 [mA] a 5 [V] lo que da como resultado 2.5 [W].

USB *en configuración servidor o dispositivo*

El USB *On The Go* (OTG) permite a un puerto actuar como servidor (*host*) o como dispositivo. Esta facilidad está específicamente diseñada para dispositivos como los teléfonos inteligentes o tabletas. Si éstos se conectan a una PC fungirán como un simple dispositivo. Si por el contrario éstos se conectan a una memoria USB, disco USB, teclado o ratón, los teléfonos inteligentes o tabletas funcionarán como servidores. El USB hace uso de dos conectores pequeños, el mini A y el mini B, el propósito es tener conectores universales. En la figura 2.19 se muestran los tipos de conectores USB más usados (comerciales) en la actualidad.

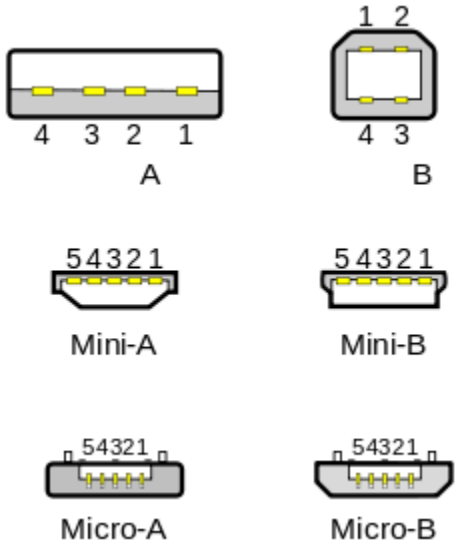


Figura 2.19. Conectores USB.

2.7. Almacenamiento masivo USB

USB implementa conexiones a dispositivos de almacenamiento usando un grupo de estándares llamado USB *mass storage device class* (MSC). Este grupo de estándares se diseñó inicialmente para memorias ópticas y magnéticas, pero ahora sirve también para soportar una amplia variedad de dispositivos, particularmente memorias USB.

USB MSC es un conjunto de protocolos de comunicación definido por la USB *Implementers Forum* que funciona sobre USB, un estándar que proporciona una

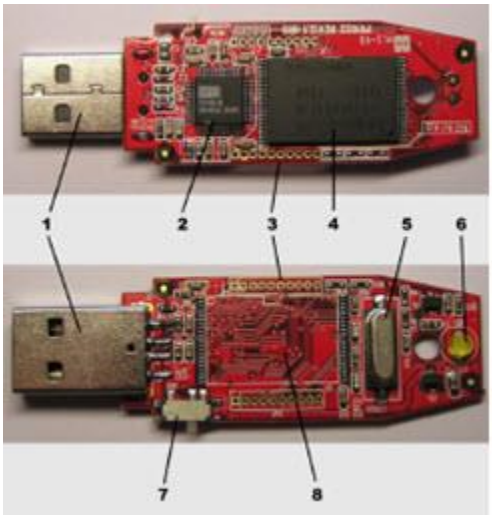
interfaz de comunicación para una variedad de dispositivos de almacenamiento. A veces también es llamado USB Mass Storage (UMS).

Memorias USB basadas en Flash

La memoria Flash se encuentra basada en semiconductores, es no volátil y re-escribible como se mencionó previamente. Debido a su alta velocidad, relativa durabilidad y bajo consumo de energía, la memoria Flash resulta ideal para muchos usos. Su pequeño tamaño, ligereza y versatilidad permite escogerlas como dispositivos portátiles.

Este tipo de memorias están fabricadas con compuertas lógicas NOR y NAND para almacenar los ceros y unos correspondientes. Comercialmente existen una variedad muy grande de fabricantes, dimensiones físicas y capacidades de almacenamiento.

Cuando las memorias Flash se pueden comunicar vía USB se dice que tenemos una memoria USB basada en Flash. En la figura 2.20 se puede apreciar un esquema general de este tipo de dispositivos.



1. Conector USB
2. Dispositivo de control de almacenamiento masivo USB
3. Puntos de prueba
4. Circuito de memoria Flash
5. Oscilador de cristal
6. LED
7. Interruptor de seguridad contra escritura
8. Espacio disponible para un segundo circuito de memoria Flash

Figura 2.20. Estructura general memoria USB basada en Flash.

Los sistemas de archivo de disco no fueron ideados para memorias Flash. Temas específicos del Flash como el número limitado de escrituras por célula pueden suceder en sistemas de archivos con escritura frecuente, haciendo que los fabricantes reserven el 5% de la capacidad del medio para técnicas que prolonguen la vida de los dispositivos de almacenamiento (wear levelling).

La especificación de UMS no exige un sistema de archivos particular. Proporciona una interfaz simple para leer y escribir sectores de datos usando el conjunto de comandos transparentes Small Computer System Interface (SCSI). Los

sistemas operativos pueden tratar el dispositivo USB como un disco duro, y pueden formatearlo con el sistema de archivos que se desee.

Debido a su relativa simplicidad, el sistema de archivos más usado en sistemas embebidos actuales es el File Allocation Table 32 (FAT32) de Microsoft, con soporte opcional para nombres largos de archivos. Los discos duros de gran tamaño basados en USB pueden venir formateados con New Technology File System (NTFS), el cual está menos soportado fuera de Microsoft Windows. Hierarchical File System (HFS+), second extended file system (ext2), Unix File System (UFS) y Reliance son otros ejemplos de sistemas de archivos.

Sistema de archivos FAT

FAT es un sistema de archivos y pertenece a una familia estándar de la industria. FAT es una herencia de otros sistemas de archivos pero más simple y robusto. Este ofrece buen desempeño incluso en aplicaciones ligeras, pero no puede entregar el mismo desempeño, confiabilidad y *escalabilidad* como algunos sistemas modernos. Sin embargo, éste es soportado por prácticamente todos los sistemas operativos existentes para ordenadores personales, y por lo tanto es un formato muy adecuado para el intercambio de datos entre ordenadores y dispositivos de casi cualquier tipo y generación, desde los 80’ hasta la actualidad.

Hoy en día, los sistemas de archivo FAT se encuentran comúnmente en memorias de estado sólido, tarjetas de memoria flash, y en muchos dispositivos portátiles y embebidos.

El nombre del sistema de archivos se origina a raíz de que el elemento más importante para su funcionamiento es una tabla de índice, la FAT, estáticamente alojada en el momento del formato. La tabla contiene las entradas para cada *cluster,* una zona contigua de almacenamiento en disco. Cada entrada contiene el número del siguiente *cluster* en el archivo, además de una marca indicando el final de archivo, espacio en disco no utilizado y áreas reservadas especiales del disco. El directorio raíz del disco contiene el número del primer *cluster* de cada archivo en el directorio; el sistema operativo puede después atravesar la tabla FAT, buscando el número de *cluster* de cada parte sucesiva del archivo en disco como una cadena de *clusters* hasta que el final del archivo es alcanzado. De la misma manera, los sub directorios son implementados como archivos especiales que contienen las entradas del directorio de sus respectivos archivos.

FAT 32

Con el fin de superar el tamaño de volumen que podían manejar sus predecesores, y al mismo tiempo permitir al sistema operativo DOS trabajar con procesadores modernos, Microsoft diseñó una nueva versión del sistema de archivos FAT, FAT 32, el cual puede manejar una mayor cantidad de *clusters*. Los valores de los *clusters* son representados por números de 32 bits, de los cuales 28 bits son usados para sostener el número de *cluster*. Los cuatro bits restantes han sido reservados para

futuras mejoras. Ya que un volumen con FAT 32 puede contener muchos más *clusters* que los que se podía con sus predecesores, es posible reducir significativamente el tamaño de los *clusters* y así limitar el espacio desperdiciado del volumen. El tamaño del volumen máximo que puede manejar FAT 32 es de 2 TB. El tamaño de un *cluster* puede ir de los 512 bytes hasta los 4,096 bytes. FAT 32 permite el uso de nombres largos para un archivo (255 caracteres). El tamaño de un archivo no puede exceder los 4 GB.

2.8. Ethernet

Con una conexión de red, un dispositivo puede alcanzar más allá de su interfaz local para enviar y recibir información de cualquier tipo, sobre distancias cortas y largas, vía cables o a través del aire. Dispositivos de diferentes tipos pueden ser comunicados usando protocolos de red que sean soportados por todos los dispositivos involucrados. En una red de sistemas embebidos, cada sistema puede comunicarse con otro sistema en la red, compartiendo información y respondiendo a solicitudes cada que sea necesario. Las computadoras incorporadas en la red pueden monitorear y controlar la operación de los sistemas embebidos.

Muchas redes locales siguen el estándar para redes popularmente conocido como Ethernet. Las redes Ethernet son flexibles y capaces de realizar muchas actividades de basto interés en cuestiones de redes. Muchos productos diseñados para usarse en redes tienen soporte para basarse en Ethernet.

La red más pequeña une solo dos dispositivos. Por ejemplo, un *data logger* puede conectarse a una computadora remota que recibe y despliega la información almacenada. En el otro extremo, Internet es la red más grande. Con una conexión de Internet, las computadoras pueden acceder a recursos en Internet y además poner disponibles sus recursos a otra computadora en Internet.

Todas las redes deben tener componentes físicos que habiliten a los dispositivos en la red para intercambiar información. Los dispositivos deben estar de acuerdo sobre cómo se va a compartir la vía de comunicación que los conecta, de tal manera que se pueda asegurar que los datos transmitidos llegan a su destino.

Todas las redes incluyen los siguientes componentes físicos:

- Dos o más dispositivos que necesitan comunicarse uno con otro. Para el caso de la presente tesis tenemos una PC y un cRIO 9012.
- Una interfaz física definida, para asegurar que la salida del dispositivo que transmite es compatible con la entrada del dispositivo que recibe. Para la presente tesis el estándar Ethernet especifica la interfaz.
- Cables o receptores-transmisores inalámbricos para conectar los dispositivos. Las redes Ethernet tiene varias opciones de cables.

Los dispositivos en la red deben estar de acuerdo en los siguientes puntos para compartir en la red:

- Reglas para decidir cuándo un dispositivo puede transmitir en la red. El estándar Ethernet contiene reglas para especificar cuando un dispositivo puede transmitir.
- Una manera de identificar el destino previsto de una transmisión. Todas las comunicaciones en una red Ethernet incluyen una dirección de hardware que identifica la interfaz Ethernet o el receptor previsto. Algunas comunicaciones adicionalmente usan protocolos de Internet que contienen información sobre el direccionamiento, como una dirección que identifica el transmisor y receptor en Internet, además es posible identificar un puerto de acceso, o el proceso que recibe el dispositivo destino.
- Un formato definido para la información enviada a través de la red, así un dispositivo puede entender y usar la información que recibe de la red. En redes Ethernet, todos los datos viajan en estructuras llamadas *frames*. Cada *frame* incluye campos para datos, direccionamiento, y otra información que pueda ayudar a los datos a alcanzar su destino sin errores.

La pila del protocolo de red

Se puede pensar que existe un conjunto de módulos aislados en donde un módulo no tiene que preocuparse de la tarea del otro, sólo debe ocuparse en desarrollar su tarea y depositar o recibir datos si es necesario. Estos módulos en conjunto permiten llevar a cabo las tareas que requiera la comunicación en un dispositivo. Los módulos pueden estar apilados en capas, uno sobre otro. La pila del protocolo de red de cualquier dispositivo contempla los módulos involucrados en la creación de redes. La figura 2.21 muestra una estructura general de la pila del protocolo de red que conecta a una red Ethernet y soporta protocolos de Internet comunes.

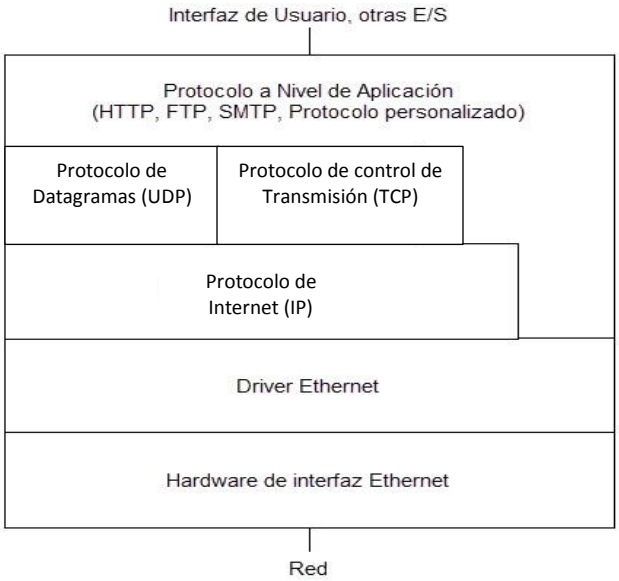


Figura 2.21. Estructura de la pila del protocolo de red para Ethernet.

Al fondo de la pila se encuentra la interfaz de hardware que conecta al cable de red. Al principio de la pila hay un módulo o módulos que otorgan información para

enviar en la red y de cómo usar la información recibida de la red. Ejemplos de éstos son: protocolo de transferencia de hipertexto (HTTP, por sus siglas en inglés), protocolo de transferencia de archivos (FTP, por sus siglas en inglés) y el protocolo para la transferencia simple de correo electrónico (SMTP, por sus siglas en inglés). En medio hay uno o varios módulos involucrados con direccionamiento, verificación de error, estado del suministro, uso de información y control.

El número de capas que debe atravesar un mensaje puede variar. Para algunos mensajes que viajan solamente dentro de una red local, la capa de aplicación puede comunicarse directamente con el *driver* Ethernet. Los mensajes que viajan en Internet deben usar el protocolo de internet (IP, por sus siglas en inglés), estos mensajes pueden usar protocolo de datagramas (UDP, por sus siglas en inglés) o protocolo de control de transmisión (TCP, por sus siglas en inglés) para agregar capacidades de verificación de errores o control de flujo.

La aplicación: Proporcionar y usar datos de la red

La aplicación proporciona datos para enviar a la red y usa los datos recibidos de la misma. Una aplicación normalmente tiene una interfaz de usuario que permite al usuario pedir información de un dispositivo en la red o mandar información en la red. En un sistema embebido, la interfaz de usuario puede habilitar configuraciones básicas y funciones de monitoreo, mientras que el sistema lleva a cabo sus comunicaciones de red sin intervención del usuario.

La información que puede enviar y recibir la aplicación puede ser cualquier cosa, un simple byte, una línea de texto, una petición a una página Web, el contenido de una página web, un archivo que contiene texto, una imagen, información binaria, código de programa, etc.

La información que envía una aplicación sigue un protocolo, un conjunto de reglas que permiten a la aplicación del dispositivo receptor entender que debe hacer con la información recibida. Una aplicación puede usar HTTP para solicitar y enviar páginas web, FTP para transferir archivos, o el SMTP para mensajes de correo electrónico. Las aplicaciones pueden también enviar y recibir datos usando protocolos de aplicación personalizados.

En un sistema embebido la aplicación puede ser un módulo que periódicamente lee y almacena lecturas provenientes de un sensor o de estados de una señal externa. Un sistema embebido puede funcionar como un servidor web que recibe y responde a peticiones de páginas web, las cuales permiten que los usuarios puedan proporcionar entradas de datos o ver información en tiempo real.

Una capa de aplicación puede soportar múltiples procesos o tareas. Por ejemplo, un único sistema puede hospedar una página web y también proporciona un servidor FTP que permita descargar archivos. Los números de puerto pueden identificar procesos específicos en el dispositivo destino.

Control de errores, flujo y puertos

Una comunicación a menudo incluye información adicional para ayudar a obtener los datos al destino de una manera eficiente y sin errores. Un módulo que soporta TCP puede agregar información para usar control de errores, control de flujo e identificar un proceso de la capa de aplicación en los dispositivos fuente y destino.

El control de errores ayuda a detectar cuando la información recibida no coincide con la que fue enviada. El control de flujo ayuda al transmisor a determinar cuando el receptor está listo para más información. Un valor que identifique un puerto del nivel de aplicación o un proceso, puede ayudar en el *ruteo* de la información recibida para ser procesada correctamente en la capa de aplicación.

En el envío de datos usando TCP, la capa de aplicación pasa la información a ser enviada y los valores que identifican la fuente de la información y destino a una capa TCP. La capa TCP crea un segmento TCP que consiste en un encabezado seguido por la información de la aplicación, como se muestra en la figura 2.22. El encabezado es una estructura definida con campos que contienen información usada en el control de errores, control de flujo y *ruteo* del mensaje al puerto correcto en el destino.

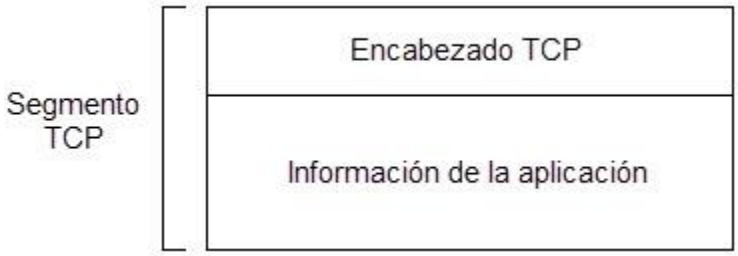


Figura 2.22. Estructura de un segmento TCP.

La capa TCP no cambia el mensaje a ser enviado. Este solamente coloca el mensaje en la porción de la información del segmento TCP. El segmento TCP encapsula la información recibida por la capa de aplicación, se puede decir que funciona como un contenedor. La capa TCP pasa luego el segmento a la capa IP para que sea transmitida en la red.

En algunas redes, la comunicación puede saltar la capa TCP. Por ejemplo, una red local de un sistema embebido no necesita de control de flujo o control de errores adicional de lo que el *frame* de Ethernet puede entregar. En estos casos, una aplicación puede comunicarse directamente con otra capa más abajo en la pila del protocolo de red, tal como la capa IP o *driver* Ethernet.

La capa IP

La capa IP puede ayudar a la información a llegar a su destino, inclusive si los dispositivos fuente y destino están en diferentes redes locales. Como el nombre lo

sugiere, el IP permite a los dispositivos en internet comunicarse uno con otro. Debido a que el IP está íntimamente relacionado con TCP, las redes locales que usan TCP también usan IP. El término TCP/IP hace referencia a la comunicación que usa TCP e IP.

En las redes Ethernet, una única dirección de hardware identifica cada interfaz en la red. Las direcciones IP son más flexibles debido a que no especifican un tipo de red. Un mensaje que usa IP puede viajar a través de diferentes tipos de redes, incluyendo Ethernet, anillo y redes inalámbricas.

En el envío de un mensaje, la capa TCP pasa el segmento TCP y las direcciones de la fuente y el destino a la capa IP. La capa IP encapsula el segmento TCP en un *datagrama*, el cual consiste en un encabezado seguido de una porción de información que puede contener un segmento TCP como se muestra en la figura 2.23. El encabezado tiene campos para las direcciones IP de la fuente y el destino, control de error del encabezado, *ruteo* y un valor que identifica el protocolo, tal como TCP o UDP, usado por la porción de información.

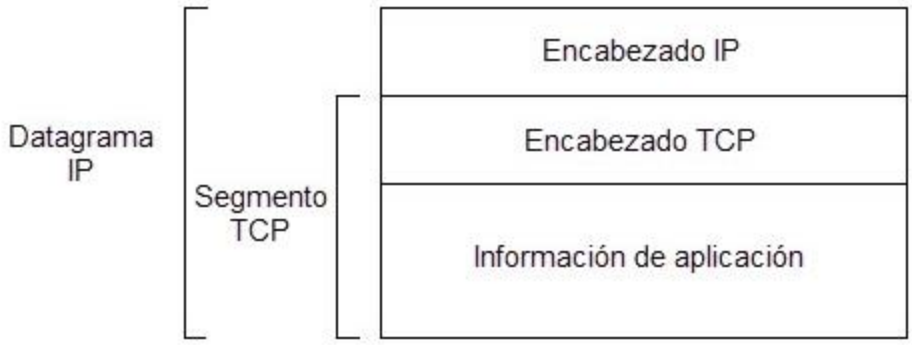


Figura 2.23. Estructura general del *datagrama* IP.

En el encabezado IP, las direcciones IP de la fuente y el destino identifican el dispositivo transmisor y el dispositivo receptor. Cada dispositivo en una red que usa dirección IP debe tener una dirección que es única dentro de la red o redes. Las redes locales pueden usar direcciones de tres bloques, reservadas exclusivamente para este tipo de redes.

Normalmente se usan tres protocolos en conjunto con IP para asignar y aprender direcciones IP, tales protocolos son: DHCP (Dynamic Host Configuration Protocol), DNS (Domain Name System) y ARP (Address Resolution Protocol).

Una comunicación en red local que no usa TCP puede no requerir IP. En lugar de eso, la capa de aplicación puede comunicar directamente con una capa más debajo de la pila como el *driver* Ethernet.

El *driver* y controlador Ethernet

En una red Ethernet, la interfaz a la red es un controlador Ethernet en circuito integrado y su *driver*. El *driver* Ethernet contiene código de programa que maneja comunicación entre el controlador en un circuito integrado y el nivel más alto en la pila del protocolo de red. Para enviar un *datagrama* IP en una red Ethernet, la capa IP pasa el *datagrama* al *driver* del controlador Ethernet. El driver le da instrucciones al controlador Ethernet para transmitir un *frame* Ethernet que contiene el *datagrama*, precedido por un encabezado que contiene direccionamiento y control de errores para la información, como se muestra en la figura 2.24.

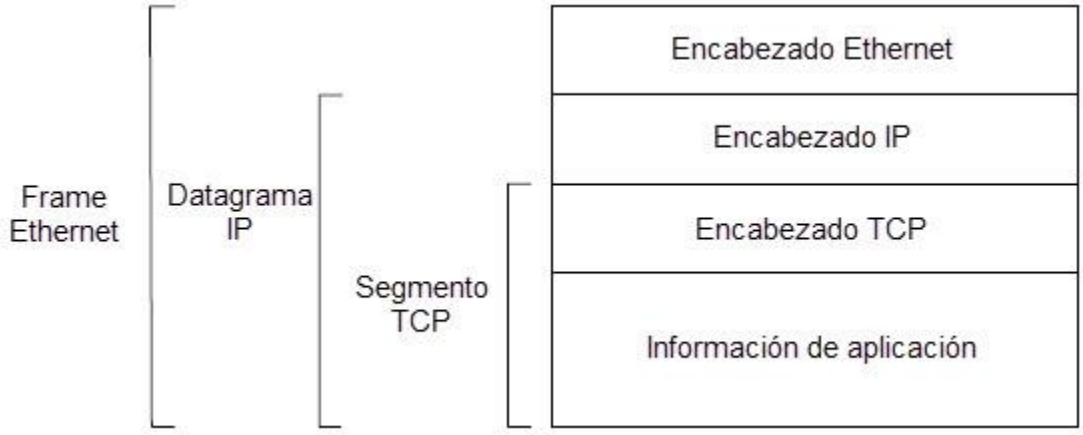


Figura 2.24. Estructura general de un *frame* Ethernet.

En la recepción de un *datagrama* IP en una red, el controlador Ethernet verifica para ver si la dirección del destino coincide con la dirección que puede aceptar el controlador previamente configurado, esta dirección puede ser del hardware de la interfaz, de una dirección de difusión (nodos) o de multidifusión (redes). Si hay una coincidencia, el controlador verifica los errores y el driver pasa el *datagrama* o una indicación de error a la capa IP.

Clientes y servidores

En algunas redes los dispositivos pueden mandar mensajes a otros dispositivos en la red en cualquier momento. Por ejemplo, un dispositivo que lleva a cabo monitoreo puede mandar una notificación de alarma a un dispositivo maestro tan pronto como la condición de alarma se presenta. El dispositivo no tiene que esperar a que el dispositivo maestro le pida la información.

En contraste muchas otras comunicaciones de red están entre un dispositivo cliente, el cual solicita recursos, y un dispositivo servidor, el cual otorga los recursos solicitados. Un recurso puede ser una página web, un archivo, o cualquier información que el servidor hace disponible. Para correr múltiples procesos, un solo dispositivo puede funcionar como cliente y servidor.

El servidor y el cliente deben estar de acuerdo en el protocolo para solicitar y mandar recursos. Para tares comunes, existen protocolos estandarizados tales como HTTP, FTP, SMTP, etc. Los dispositivos típicamente envían la petición y responden en las porciones de información de un segmento TCP.

Cuando se usa un navegador para ver páginas web, el navegador está funcionando como un cliente, solicitando páginas del dispositivo servidor que almacena las páginas web. Para funcionar como un servidor que proporciona páginas web, un dispositivo debe correr un software de servidor. Los sistemas embebidos pueden funcionar como servidores con la adición de un código de programa que decodifique y responda a peticiones recibidas.

Otro uso para los términos de cliente y servidor se refiere a dispositivos que han establecido una conexión TCP, la cual permite a los dispositivos intercambiar información usando TCP. El cliente es el dispositivo que inicia la conexión, mientras que el servidor es la computadora que acepta la petición para conectarse. Una vez establecida la conexión, ambos dispositivos pueden enviar mensajes uno al otro, a través de un protocolo de alto nivel se puede limitar lo que cada dispositivo puede hacer.

Facilidad de uso y versatilidad

Ethernet puede transferir cualquier tipo de información, desde mensajes cortos hasta archivos largos. Una comunicación Ethernet puede tomar ventajas de la existencia de protocolos de alto nivel tales como TCP e IP, o puede usar un protocolo para una aplicación específica. Ethernet no necesita dispositivos rápidos, con el agrado de un controlador Ethernet, incluso un microcontrolador de 8 bits, es posible realizar comunicación en una red Ethernet.

Con Ethernet, mucho del trabajo ha sido hecho. Todos los dispositivos en la red siguen las especificaciones del estándar Ethernet para interconexión, manejo del tráfico de la red e intercambio de información. No es necesario diseñar la interfaz de hardware o inventar las “reglas del juego”. Ethernet es lo suficientemente flexible para permitir opciones. Por ejemplo, una red puede usar pares trenzados, fibra óptica o cable coaxial, en la tabla 2.1 se muestran características de cada tipo de cable. Los requerimientos para cada tipo de cable y velocidad están especificados, de esta manera, todo lo que se necesita es decidir el tipo de cable que encaja mejor en la aplicación.

Una conexión full dúplex es aquella en la que se puede implementar una comunicación bidireccional, enviando y recibiendo información de manera simultánea. En contraste en una conexión half dúplex, sólo se permite que la información fluya en una u otra dirección, pero no en las dos al mismo tiempo, con este tipo de conexión, cada extremo de la conexión transmite uno después del otro.

Tipo de Cable	Par trenzado	Fibra óptica	Cable coaxial
Máxima tasa de transferencia (Mb/s)	1,000	10,000	10
Máxima longitud por segmento (m)	100	2,000 (half dúplex)	500 (grueso)
		5,000 (full dúplex)	185 (delgado)
Costo	Bajo	Alto	Moderado
Inmunidad al ruido	Buena	Excelente	Bueno
Facilidad de instalación	Excelente	Buena con cables prefabricados	Regular

Tabla 2.1. Tipos de cable soportados por Ethernet.

2.9. Interconexión de componentes periféricos

El *bus* PCI puede estar sujeto a la conexión de un gran número de dispositivos que requieren accesos rápidos entre ellos y/o a la memoria del sistema; estos dispositivos pueden ser accedidos por un procesador a frecuencias muy altas, son capaces de trabajar a frecuencias cercanas a la máxima frecuencia del *bus* del procesador. Todas las lecturas y escrituras en el *bus* PCI pueden ser llevadas a cabo como transferencias repetitivas, no es necesario esperar por una entrada de otro dispositivo o por un proceso interno.

Como se ha mencionado en el párrafo anterior, el bus PCI pude trabajar como un bus de procesador, es decir, puede operar parecido a un bus que está dentro de un procesador. Para que trabaje como un bus de procesador es necesario que sus procedimientos estén estandarizados, de tal manera que pueda ser usado con los diferentes procesadores en el mercado.

Cuando algún dispositivo es conectado al bus, el procesador le asigna un conjunto de sus direcciones disponibles para que pueda comunicarse con él. El Bus PCI otorga la ventaja de enchufar y usar (PnP, por sus siglas en inglés). Al conectar un dispositivo en el bus, el procesador es capaz de reconocerlo rápidamente y establecer una comunicación. Para hacer una conexión PnP correctamente, es necesario tener dispositivos PnP, sistema operativo PnP y software PnP.

Algunos sistemas de buses como Arquitectura Estándar de la Industria (ISA, por sus siglas en inglés) y Arquitectura Estándar Industrial Extendida (EISA, por sus siglas en inglés) fueron buenos mientras los sistemas basados en procesadores no tenían más que una aplicación corriendo a la vez. Actualmente los sistemas pueden correr más de una aplicación a la vez, además ahora se requiere de un bus que pueda manejar varios dispositivos. El bus PCI puede cumplir con estos requisitos, gracias a que trabaja con tasas de reloj más rápidas que los antiguos buses y además puede ser inhabilitado, lo que permite a otros dispositivos tener la oportunidad de interactuar con el procesador, esto es enviar y recibir información.

Los tamaños de las tarjetas son una ventaja adicional del bus PCI. La mayoría de los circuitos impresos (PCBs, por sus siglas en inglés) incorporan componentes de montaje superficial, esto permite crear tarjetas PCI mucho más pequeñas y con mayores funciones que las que podrían otorgar otras tarjetas como las EISA.

Los dispositivos PCI son capaces de usar interrupciones (IRQ, por sus siglas en inglés) y Acceso Directo a Memoria (DMA, por sus siglas en inglés) de una manera diferente a otros sistemas de buses. El bus PCI tiene sus propias interrupciones internas llamadas INT#. Lo anterior es importante para el manejo adecuado de dispositivos PnP, al estar varios dispositivos funcionando al mismo tiempo sin interferir uno con otro se hace necesario el uso de recursos como IRQ y DMA. A veces también existe lo que se conoce como bus maestro PCI, el cual puede administrar los tiempos para el acceso al bus de cada uno de los dispositivos. El comando utilizado para la administración de tiempos se conoce como comando temporizador de latencia (suma de retrasos).

Puede existir un dispositivo externo que haga las veces de maestro, éste puede interrumpir el funcionamiento de otros dispositivos conectados al bus PCI en medida de sus necesidades, para ello debe hacer uso de las interrupciones del bus. El bus PCI es realmente un sistema de muchos buses con dispositivos que están alojando sus recursos en cada uno de ellos.

La especificación PCI cubre el tamaño físico del bus (tamaño y espacio de los bordes de las tarjetas y los contactos eléctricos), características eléctricas, temporización del bus y protocolos. Las especificaciones están a cargo del Grupo de Interés Especial PCI (PCI-SIG, por sus siglas en inglés).

2.10. Unidad de estado sólido

Un SSD, es un dispositivo de almacenamiento de datos que puede usar memoria no volátil, como la memoria Flash, o memoria volátil como la SDRAM, para almacenar datos. En general su tiempo de búsqueda es pequeño y sus procesos acumulan pocos retrasos.

Actualmente la mayoría de los SSDs utilizan memoria flash basada en compuertas NAND, éstas son capaces de retener los datos sin alimentación. Para aplicaciones que requieren acceso rápido, aunque se tenga que sacrificar la permanencia de los datos al no tener energía, los SSDs pueden ser construidos a partir de RAM. Para estos últimos SSDs, se podrían emplear fuentes de alimentación independientes y así evitar la pérdida de datos.

Debido a que cRIO tiene en su estructura un SSD con memoria Flash basada en compuertas NAND, nos enfocaremos en esta arquitectura. Este tipo de SSD se compone principalmente por:

- Controladora: Es un procesador que debe administrar, gestionar y unir los módulos de memoria NAND. Ejecuta un software preinstalado por el fabricante y es el factor que define la velocidad del SSD.
- *Caché*: Es un dispositivo de memoria RAM. Mientras el SSD está en funcionamiento, en esta parte se encuentra el directorio de la colocación de bloques e información para aminorar el desgaste del SSD (Wear Leveling).
- Condensador: Este elemento es necesario para mantener la integridad de los datos del *Caché.* Si la alimentación se ha detenido inesperadamente, este dispositivo debe dar el tiempo suficiente para que los datos de Caché viajen a la memoria no volátil.

Sistema de archivos Reliance Nitro

Varios sistemas de archivos han realizado grandes esfuerzos para trabajar y gestionar archivos, esto ha incrementado la vida útil de los SSDs. Lo anterior es de vital importancia, se debe recordar que la principal desventaja de los SSDs es su limitado número de ciclos escritura/lectura.

El sistema de archivos elegido por National Instruments para gestionar sus SSDs es Reliance Nitro, el desarrollo de éste se encuentra a cargo de la empresa Datalight. Reliance Nitro está orientado a dispositivos embebidos, dónde la perdida de energía es muy frecuente y la protección de los datos es de vital importancia. La confianza sobre la protección de datos, ante eventos inesperados, es la cualidad más importante de este sistema de archivos.

Reliance Nitro puede administrar diferentes tecnologías, por ejemplo: memorias flash, RAM, discos duros, dispositivos de almacenamiento USB y Securie Digital (SD). Además, este sistema de archivos es capaz de trabajar con casi todos los sistemas operativos de 32 bits comerciales.

La velocidad para acceder a un directorio o archivo es mucho mayor que otros sistemas, en acceso es mucho más rápido que FAT. Su velocidad no compromete la integridad de los datos.

La estructura del sistema de archivos Reliance Nitro está basada en una arquitectura de árbol, lo que facilita la búsqueda de información a altas velocidades. El tamaño del volumen máximo es de 32 TB, la única restricción que este sistema de archivos le pone al tamaño de archivo, viene dada por el espacio libre en el dispositivo. Lo anterior quiere decir que un archivo puede tener el tamaño de la unidad de almacenamiento, algo que FAT no puede lograr para unidades de almacenamiento con más de 4 GB de espacio.

Para la gestión de datos, Reliance Nitro usa prácticamente dos árboles:

- Árbol de directorios. El propósito del este árbol es asociar un nombre con un único número de archivo dentro de un directorio principal. Esta asociación es llevada a cabo mediante una estructura de árbol.

- Árbol de asignación. El propósito de este árbol es asociar bloques asignados con un archivo, así como la información sobre el archivo mismo.

Reliance Nitro comprende dos estados, el estado de trabajo y el estado comprometido. Los datos originales (estado comprometido) se conservan hasta que los nuevos datos (estado de trabajo) se escriben y se realiza una nueva operación. Durante la operación en el estado de trabajo, todas las modificaciones del sistema de archivos, incluyendo cambios en los directorios, archivos e información de los archivos, se almacenan en un área del medio que no está en uso y no contiene datos en estado comprometido. La posibilidad de que los datos se corrompan es eliminada, debido a que los datos de un nuevo proceso se escriben en una parte no utilizada por el medio. Escribir en una parte del dispositivo que no es usada, permite al estado anterior del sistema de archivos no ser tocado. Los datos del proceso anterior están siempre disponibles.

2.11. Reloj de tiempo real

Es de vital importancia para un sistema de adquisición de señales contar con una referencia de tiempo, ésta permite ubicar en el dominio del tiempo a los fenómenos que han ocurrido durante una adquisición de datos. Para el desarrollo de esta tesis la referencia de tiempo está a cargo de un reloj de tiempo real.

Los Relojes en Tiempo Real (RTC, por sus siglas en inglés) son dispositivos que contienen un oscilador y que generalmente necesitan un cristal para trabajar con la precisión requerida. Este cristal es un dispositivo piezoeléctrico, esto quiere decir que el cristal es capaz de polarizarse eléctricamente cuando se le aplica una fuerza para deformarlos y viceversa. Por lo dicho anteriormente el cristal es capaz de oscilar a una frecuencia precisa y conocida.

El RTC se encuentra embebido dentro del sistema de medición, permite agregar estampas de tiempo a los datos grabados en el dispositivo de almacenamiento. La exactitud de la estampa de tiempo va a depender del cristal con el que cuenta el RTC. La exactitud se puede medir en unidades Partes por Millón (PPM). Cristales con menor PPM cuestan mucho más que los que cuentan con un alto PPM.

Cuando un cristal cuenta con una exactitud de 20 PPM, lo cual se interpreta como $\pm$ 20, tendremos en el tiempo una variación de $\pm$ 20 segundos después de 11 días, 13 horas, 46 minutos y 40 segundos. Lo anterior equivale a decir que después de un millón de segundos tendremos una variación de $\pm$ 20 segundos. Para este ejemplo hemos supuesto que el cristal opera a una frecuencia de 1 Hz.

Es importante resaltar que las variaciones de temperatura son de vital importancia en el comportamiento del cristal, un incremento en la temperatura puede producir una mayor cantidad de inexactitud en la medición del tiempo.

2.12. La programación gráfica

Actualmente el lenguaje gráfico de mayor uso a nivel mundial es LabVIEW. Sus competidores cercanos tales como Visual Designer de Burr Brown o Visual Engineering Environment de Hewlet Packard, no han logrado superar la cantidad de usuarios con los que cuenta LabVIEW de National Instruments.

LabVIEW es un ambiente de programación en el que es posible crear programas usando una notación gráfica, es decir, conectando nodos funcionales a través de alambres mediante los cuales la información fluye. En este sentido, LabVIEW es diferente a los lenguajes de programación como C, C++ o Java, en los que debemos programar con texto. El ambiente de desarrollo LabVIEW puede trabajar en computadoras que cuenten con sistema operativo Windows, MAC OS X o Linux. En LabVIEW es posible crear programas que corran en las plataformas mencionadas, además de una gran variedad de plataformas embebidas, incluyendo FPGAs, DSPs y MCUs.

Tomar mediciones, analizar datos y presentar resultados al usuario son las tres actividades en las que está basado LabVIEW. Este ambiente de programación ofrece más flexibilidad que los instrumentos de laboratorio estándar, debido a que está basado en software. Con la computadora, hardware con capacidad de conexión y LabVIEW, se puede crear el instrumento virtual necesario. En cualquier momento es posible modificar el instrumento virtual.

LabVIEW cuenta con una extensa cantidad de archivos de biblioteca o funciones y subrutinas que ayudan con la mayoría de las tareas programables, sin alboroto de apuntadores, asignación de memoria y otras complicaciones de la programación que se encuentran en los lenguajes de programación convencionales. LabVIEW cuenta con archivos de biblioteca de código para aplicaciones específicas como: adquisición de datos (DAQ, por sus siglas en inglés), bus de interfaz de propósito general (GPIB, por sus siglas en inglés), control de instrumentos por puerto serie, análisis de datos, presentación de datos, almacenamiento de información, comunicación vía internet, etc.

La forma en la que trabaja LabVIEW

Un programa en LabVIEW consiste en uno o más instrumentos virtuales (VIs, por sus siglas en inglés). Los instrumentos virtuales recibieron su nombre debido a que en apariencia y operación suele imitar a instrumentos físicos reales. Sin embargo, detrás de su apariencia, los VIs son análogos a los programas principales, funciones y subrutinas de lenguajes de programación populares como C o Visual Basic. En el transcurso de esta tesis nos referiremos a un programa de LabVIEW como VI. Éste último tiene tres partes principales: un panel frontal, un diagrama de bloques y un icono.

El panel frontal es la interface de usuario interactiva de un VI, su nombre es debido a que simula el panel frontal de un instrumento físico. En la figura 2.25 (área izquierda) se muestra un panel frontal, este puede contener botones, perillas, gráficas, muchos otros controles (entradas de los usuarios) e indicadores (salidas del programa).

Se puede introducir los datos usando el mouse y teclado, posteriormente se pueden ver los resultados del programa en la ventana.

El diagrama de bloques, figura 2.25 (área derecha), es la fuente de código de un VI, construido en lenguaje de programación gráfico de LabVIEW (lenguaje G). El diagrama de bloques es el programa ejecutable real. Los componentes de un diagrama de bloques son VIs de bajo nivel, basados en funciones, constantes y estructuras de control de ejecución. Se dibujan alambres para conectar los objetos apropiados y definir el flujo de datos entre ellos. Los objetos en el panel frontal tiene sus correspondientes terminales en el diagrama de bloques, de esa manera puede pasar información desde el usuario hacia el programa y de regreso al usuario.

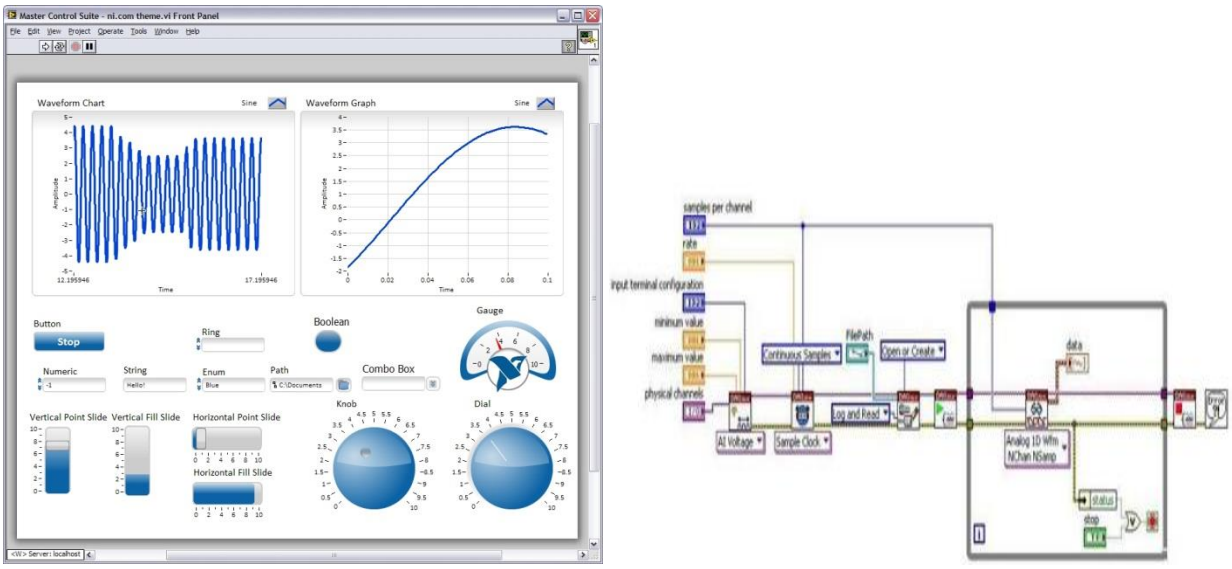


Figura 2.25. Panel frontal (izquierda) y diagrama de bloques (derecha).

Para poder usar un VI como una subrutina en el diagrama de bloques de otro VI, éste debe tener un icono con un conector como se muestra en la figura 2.26. Un VI que es usado dentro de otro VI se llama sub VI y es análogo a una subrutina. El icono es una representación gráfica de un VI y es usado como un objeto en el diagrama de bloques de otro VI. El conector de un VI es el mecanismo usado para alambrar datos en un VI desde otro diagrama de bloques cuando el VI es usado como sub VI. Muy parecido a los parámetros de una subrutina, el conector define las entras y salidas de un VI.

Figura 2.26. Icono de un VI (izquierda) y su conector (derecha).

Los instrumentos virtuales son jerárquicos y modulares. Se pueden usar como programas de alto nivel o subprogramas. Con esta arquitectura, LabVIEW promueve el

concepto de programación modular. Primero, se divide una aplicación en una serie de sub tareas simples. Luego, se construye un VI para llevar a cabo cada una de las sub tareas y finalmente se combinan estos VIs en un diagrama de bloques de alto nivel para realizar las tarea principal.

La programación modular es un *plus* debido a que permite ejecutar cada sub VI por sí mismo, lo cual facilita el *debugging*. Además, muchos sub VIs de bajo nivel a menudo llevan a cabo tareas comunes para diferentes aplicaciones, esto permite que puedan ser usados independientemente por cada aplicación.

En la tabla 2.2 es posible apreciar una comparación o relación existente entre LabVIEW y los lenguajes de programación convencionales.

LabVIEW	Lenguajes Convencionales
VI	Programa
Función	Función o método
Sub VI	Sub rutina, sub programa, objeto
Panel frontal	Interfaz de usuario
Diagrama de bloques	Código de programa
G	C, C++, Java, Pascal, BASIC, etc.

Tabla 2.2. Equivalencia entre LabVIEW y lenguajes convencionales.

LabVIEW en plataformas diferentes a una PC

Hay módulos especiales que incrementan el campo de acción de LabVIEW. Algunos de estos módulos son: LabVIEW Real-Time, LabVIEW FPGA, LabVIEW PDA y LabVIEW Embedded, los cuales permiten correr VIs de LabVIEW en otros sistemas diferentes a una PC.

El módulo LabVIEW Real-Time es una combinación de hardware y software que permite tomar porciones de código desarrollado en LabVIEW, para descargarlo y ejecutarlo en cualquier microcontrolador usado en las arquitecturas cRIO y con sistema operativo de tiempo real. Esto significa que puedes garantizar que ciertas partes del programa en LabVIEW van a correr con precisión, incluso si la interfaz de usuario con la máquina *host* colapsa debido un paro inesperado de la máquina o a un error en la comunicación.

El módulo LabVIEW FPGA permite descargar y ejecutar VIs en un FPGA. Este módulo permite aprovechar la capacidad de ejecución en paralelo inherente a la programación basada en flujo de datos. La idea es hacer compatible la programación en LabVIEW con los FPGAs que cuentan también con la capacidad de procesar en paralelo.

El módulo LabVIEW Embedded permite compilar los VIs y ejecutarlos en cualquier microcontrolador o microprocesador de 32 bits, lo anterior es resultado de integrar LabVIEW con cadenas de herramientas de terceros. Podemos mencionar a la colección de compiladores GNU, eCos y Wind River Tornado/VxWorks.

En este capítulo se han abordado los conceptos básicos que permiten entender el contenido de la presente tesis. En el siguiente capítulo se presenta todo lo relacionado al desarrollo del sistema de adquisición de variables analógicas.

Capítulo 3
Desarrollo del sistema

En este capítulo se hace una descripción sustancial de la forma en que se implementó el sistema planteado en el capítulo 1. Se empieza abordando los requerimientos que debe cumplir el sistema, de tal forma que tengamos presente el objetivo que se quiere alcanzar. Una vez visto los objetivos, se detalla la parte del hardware implementado y su configuración. Por último, se explica la parte del sistema en donde se tiene mayor injerencia, el software.

3.1. Requerimientos

La ciencia experimental y la ingeniería en todo el mundo se ha diversificado de una forma inimaginable. Esta diversificación ha producido un sin número de áreas de desarrollo, cada una de ellas tiene su propio objeto de estudio, y aunque en apariencia esos objetos de estudio comparten poco o nada, en todas las áreas existe un elemento en común, el procesamiento de señales analógicas.

Sin entrar en detalle, a decir de los expertos, vivimos en un mundo analógico. Esto da lugar a una creciente necesidad de digitalizar señales analógicas. Es por ello que esta tesis tiene como objetivo contar con un sistema de adquisición de señales analógicas estándar, al cual sólo se le tenga que agregar un sistema de acondicionamiento, no siempre necesario, según la naturaleza de la señal analógica, y así poder desarrollar un sistema robusto, flexible y confiable, que pueda adquirir cualquier tipo de señal analógica previamente acondicionada.

El contar con un sistema de adquisición de señales estándar, involucra desde una perspectiva muy general, las necesidades descritas a continuación:

Hardware:

- *Puertos y dispositivos de entrada/salida*. Éstos permiten la comunicación del sistema de adquisición de datos con el entorno donde se ubica y otros sistemas electrónicos.

- *Elementos de acondicionamiento.* Éstos deben ayudar a los dispositivos entrada/salida a interactuar de manera correcta con los digitalizadores.
- *Digitalizadores.* Su tarea debe ser modificar la presentación de las señales analógicas, a un formato que pueda ser reconocido por el dispositivo electrónico de control.
- *Dispositivos de aislamiento*. Ya que las características eléctricas de las señales externas podrían dañar la electrónica o inducir errores en la medición, el aislamiento es de gran importancia.
- *Dispositivos de almacenamiento de datos*. Cuando se quiere generar un historial de actividades, para posteriormente efectuar un análisis, se debe contar con un sistema de almacenamiento de datos.
- *Controlador*. Se necesita que alguien administre los recursos del sistema, bajo un conjunto de condiciones impuestas por las necesidades del usuario.

Software:

- Una interfaz de configuración que pueda ser manipulada casi por cualquier tipo de usuario y que contenga:
  - Configuración de la frecuencia de muestreo.
  - Hora y fecha de referencia.
  - Identificadores para los archivos.
  - Duración de pruebas.
- Diferentes modos de almacenamiento de la información:
  - De manera continua.
  - Cuando una señal cumpla una condición impuesta.
  - El usuario pide el inicio y decide cuando parar.
  - Durante un intervalo de tiempo establecido.
- Capacidad para manejar archivos en formato ASCII o binario.
- Despliegue de los datos contenidos en archivos mediante gráficos y tablas.
- Acceso aleatorio a los archivos para uso eficiente de memoria.
- Visualización en tiempo real de la información proveniente del ADC.

3.2. El hardware del sistema

En el Instituto de Ingeniería de la UNAM, en la Coordinación de Instrumentación, el uso de la plataforma cRIO de National Instruments empieza a ser de gran importancia. Lo anterior debido a que desde hace algunos años, la creciente demanda de sistemas de adquisición en las diferentes áreas del Instituto de Ingeniería, ha obligado al departamento a lograr sistemas de adquisición que demandan versatilidad y que cuentan con una gran cantidad de requerimientos.

¿Qué es CompactRIO?

cRIO es un sistema reconfigurable y embebido de control y adquisición de datos. Su arquitectura robusta incluye módulos de E/S, un chasis que cuenta con un MCU de tiempo real y un chasis reconfigurable basado en FPGA. cRIO puede ser programado

con las herramientas que brinda el entorno de desarrollo LabVIEW, además se presta para ser usado en una gran cantidad de aplicaciones embebidas que requieran control y monitoreo (caso de la presente tesis).

De una manera muy general, en la figura 3.1 se puede apreciar un esquema del sistema CompactRIO. En el extremo izquierdo se tienen los puertos Ethernet, USB y RS-232. A través del puerto Ethernet se realiza el enlace con una PC. Este enlace permite la transferencia de información entre el sistema cRIO y la PC. El puerto Ethernet también permite programar el hardware que incorpora cRIO. El puerto USB permite realizar la conexión con una memoria flash. En esta memoria se almacena toda la información de la adquisición de datos. En la presente tesis no se hizo uso del puerto RS-232; puede usarse para comunicación RS-232. A la derecha de los puertos mencionados se puede ver al MCU de tiempo real. Éste es el encargado de administrar a los tres puertos ya mencionados. El MCU también tiene la tarea de comunicarse con el SSD, el cual almacena la ruta del último archivo de configuración creado. A la derecha del MCU, se puede ver una conexión a través de PCI entre el MCU y el FPGA. Por último, se puede ver que el FPGA es el encargado de interactuar con los módulos de entrada/salida. Existen módulos de diferentes arquitecturas y que pueden realizar tareas diferentes. Inclusive National Instruments puede fabricar módulos a la medida.

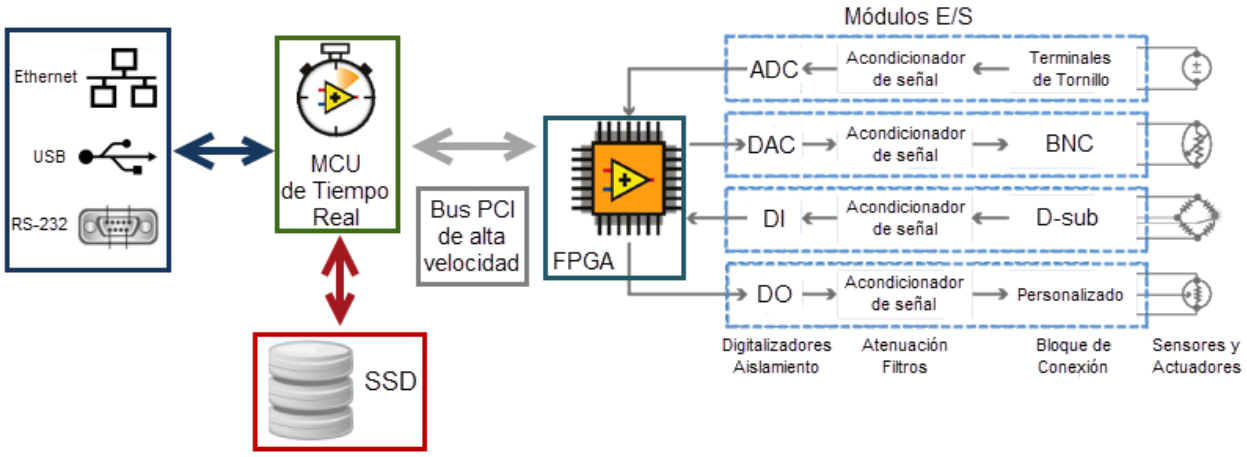


Figura 3.1. Esquema de alto nivel de CompactRIO.

La familia cRIO cuenta con una amplia variedad de opciones, contemplan diferentes tipos de MCUs y FPGAs. Es posible encontrar equipos para casi cualquier tipo de aplicación.

3.2.1. MCU embebido de tiempo real

El cRIO 9012, el cual se muestra en la figura 3.2, cuenta con el MCU industrial y de tiempo real Freescale MPC5200, a este dispositivo se le ha dedicado una parte del capítulo 2. Al MPC5200 se le ha agregado una memoria DRAM de 64 MB, la cual funciona como su memoria principal, además está configurado para alcanzar una

frecuencia de procesamiento de 400 MHz. Es importante agregar que este MCU puede ejecutar aplicaciones de tiempo real, lo cual implica determinismo y confiabilidad.

Figura 3.2. NI cRIO 9012 (MCU de tiempo real).

Al ser determinístico, el cRIO 9012 tiene la capacidad de definir, con una alta probabilidad, la cantidad de tiempo que se toma una tarea en iniciarse. Lo anterior tiene una relación muy fuerte con las interrupciones, ya que el determinismo se puede medir en relación al tiempo de respuesta que tiene el sistema ante una interrupción.

El MCU embebido cRIO 9012 está diseñado para ambientes hostiles, exactitud y bajo consumo de potencia. Cuenta con dos entradas para alimentación, éstas pueden recibir tensiones de corriente directa que no superen los 35 Volts. Para iniciar el cRIO 9012 es necesario suministrar mínimamente 9 Volts, pero ya en operación se puede disminuir la tensión hasta 6 Volts, esto lo hace un buen prospecto para trabajar por un largo periodo de tiempo con baterías o energía solar. La alimentación entrega energía al cRIO 9012, al NI 9113 y a los módulos entrada/salida que pudieran llegar a conectarse. El consumo del cRIO 9012 es de 6 W, pero el de todo el sistema puede llegar a ser de 20 W. cRIO 9012 puede operar a una temperatura ambiente que se encuentre entre -40 y 70 °C, cuando se exceden los 50 °C en el ambiente, el fabricante recomienda usar disipadores de calor.

El cRIO 9012 necesita el sistema operativo VxWorks y el *driver* NI RIO para su correcto funcionamiento bajo el entorno de programación LabVIEW. Para lograr su programación se necesita el núcleo de LabVIEW, el módulo LabVIEW FPGA y el módulo LabVIEW Real-Time.

El MCU de tiempo real se puede conectar a un *chasis* reconfigurable basado en FPGA de cuatro u ocho espacios. El circuito FPGA de este *chasis* controla cada módulo de entrada/salida del sistema y transfiere la información al MCU a través de un bus PCI local, usando funciones basadas en comunicación.

Con el puerto de comunicación serie y el puerto Ethernet (10/100 Mbits/s) es posible establecer comunicación usando los protocolos: comunicación serie, TCP/IP,

UDP y Modbus/TCP. El cRIO 9012 también cuenta con características que permiten trabajan con HTTP y FTP. Para almacenamiento adicional, el cRIO 9012 incorpora un puerto USB *host,* al cual se le pueden conectar dispositivos de almacenamiento de datos basados en USB (memorias Flash y discos duros). Además, cRIO 9012 cuenta con un sistema de archivos tolerante a fallas (Reliance) que brinda confianza para el almacenamiento de datos en su SSD con capacidad de 128 MB.

En la figura 3.3 se muestran los puertos, indicadores y controles con los que cuenta el cRIO 9012.

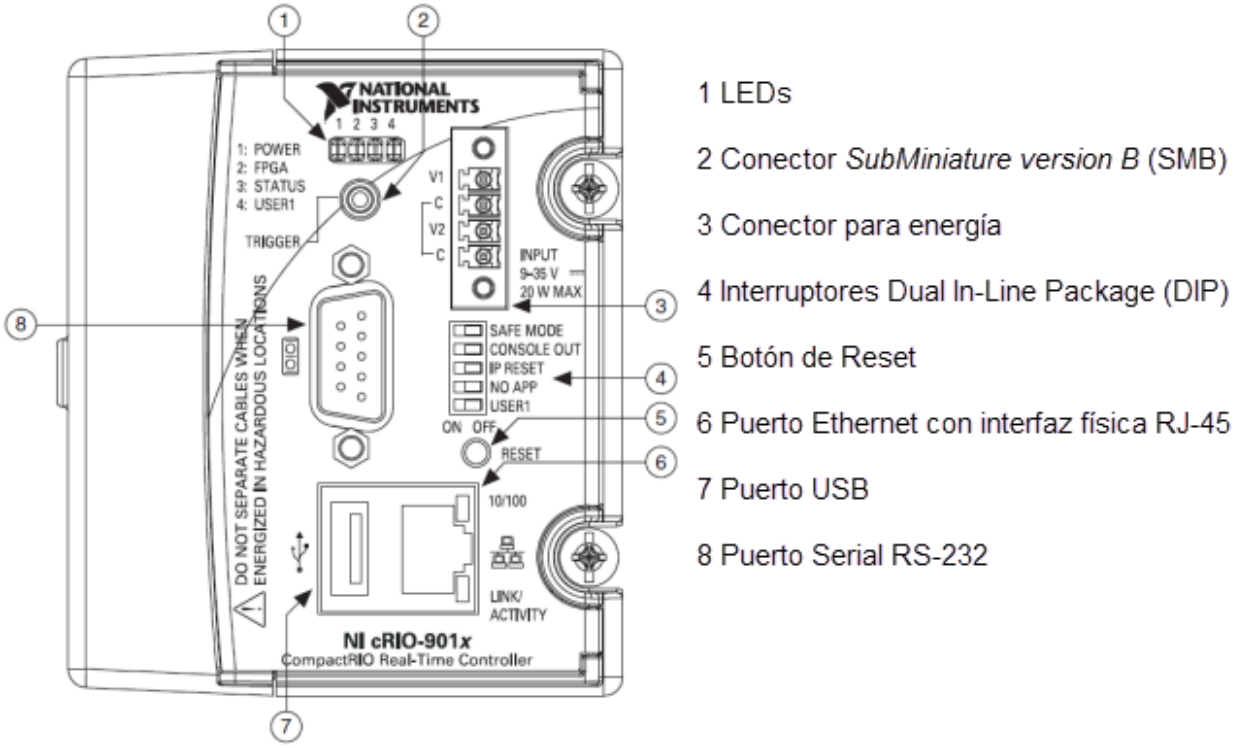


Figura 3.3. Puertos, indicadores y controles del NI cRIO 9012.

En la figura 3.3 es posible ver que existen cuatro LEDs. El primer LED, denominado POWER, es el encargado de indicar si el controlador está energizado o no, también nos indica si la fuente que hemos conectado al cRIO es adecuada y que la energía se está distribuyendo en todo el sistema. Al decir sistema nos referimos a todos los elementos del cRIO 9012, al chasis reconfigurable y a los módulos entrada/salida. El segundo LED, denominado FPGA, ayuda a depurar la aplicación. Es una manera fácil de indicar el estado de un proceso dentro del VI que esté corriendo en el FPGA. El control de este LED se logra haciendo uso del módulo LabVIEW FPGA y del driver NI RIO. El tercer LED se llama STATUS (estado). Durante una operación en la que no haya ocurrido ningún tipo de error relacionado al software y hardware del cRIO, siempre debe permanecer apagado. En caso de que existe algún problema, existe la posibilidad de que el LED lo indique mediante cuatro modos de operación descritos a continuación:

- El LED parpadea cada segundo. Esto quiere decir que el cRIO no está configurado. Es necesario hacer uso de la aplicación Measurement & Automatitaion Explorer (MAX), la cual forma parte de las herramientas que ofrece la distribución básica de LabVIEW.
- El LED parpardea 2 veces por segundo. El cRIO ha detectado un error en el software. Esto usualmente ocurre cuando un intento de actualización de software es interrumpido. Es necesario reinstalar el software en el cRIO.
- El LED parpadea 3 veces por segundo. El controlador se encuentra en modo seguro debido a que la opción SafeMode del interruptor DIP se encuentra en la posición ON.
- El LED parpadea 4 veces por segundo. El software del cRIO ha colapsado dos veces sin reiniciar o sin prender y apagar entre colapsos. Esto normalmente ocurre cuando el cRIO desbordado su memoria.
- El LED se encuentra prácticamente prendido todo el tiempo. Erro no identificado y es necesario contactar a National Instruments.

Para poder mandar las indicaciones antes descritas, el cRIO hace un auto examen cada vez que es energizado, de tal forma que hay que esperar unos segundos para poder saber si el sistema se encuentra trabajando de manera adecuada.

El conector SMB que se aprecia en la figura 3.3, permite conectar dispositivos digitales al cRIO. Por ejemplo, si se conecta el pulso por segundo que da como salida un Sistema de Posicionamiento Global (GPS, por sus siglas en inglés), se puede corregir la variación del reloj del sistema. Internamente cRIO 9012 cuenta con un reloj de tiempo real, este servirá como medio para sincronizar el reloj del sistema, lo que nos permite prescindir del conector SMB.

Como se había mencionado en este mismo capítulo, el cRIO necesita una fuente de energía externa. El cRIO filtra y regula la energía suministrada y entrega energía al MCU, al FPGA y a los módulos de entrada/salida. Es necesario conectar la terminal C y la terminal V1 o la terminal V2, figura 3.3. Adicionalmente, es posible tener dos fuentes de energía conectadas. En este caso, se usan las terminales C, V1 y V2. En las terminales V1 y V2 se deben conectar los conductores positivos y en las terminales C los conductores negativos de las fuentes. El cRIO toma la fuente de energía con la mayor tensión, además tiene circuitería para la protección de tensión inversa.

Es importante saber que las terminales C están internamente conectas una con otra y además están conectadas al *chasis*. Si se usan dos fuentes de alimentación, hay que asegurarse que comparten una tierra común.

En la figura 3.3 se alcanza a apreciar un control con 5 interruptores tipo DIP. El interruptor rotulado con el nombre SAFE MODE, es el que determina si el motor

embebido de LabVIEW Real-Time se ejecuta cuando el MCU inicia. Si el interruptor está en la posición OFF, el motor de LabVIEW Real-Time se ejecuta. Para una operación normal (todos los recursos de software disponibles), es necesario mantener este interruptor en OFF. Cuando el interruptor está en posición ON, el cRIO ejecuta sólo los servicios esenciales para actualizar su configuración e instalación de software, el motor de LabVIEW Real-Time no corre.

El interruptor llamado CONSOLE OUT, permite realizar conexión con una computadora mediante el puerto de comunicación serie, para ello es necesario contar con software que se encargue de administrar la conexión en la PC y además que el interruptor éste en la posición de ON. Durante una operación normal es necesario dejar este interruptor en la posición OFF.

El interruptor IP RESET, en conjunto con el botón RESET, permiten poner la dirección IP del cRIO en 0.0.0.0. Para que lo anterior suceda, es necesario poner el interruptor en ON y reiniciar el cRIO. Para establecer una dirección IP, es necesario hacer uso del programa MAX o implementar una rutina por software que se encargue de ello.

Es posible establecer una aplicación que se ejecute siempre que inicie el cRIO, para ello es necesario realizar una serie de pasos que se describen en páginas posteriores de la presente tesis. Por lo pronto, nos interesa saber que es posible tener una aplicación de arranque, esto debido a que el interruptor NO APP en su posición ON, permite evitar que cualquier aplicación configurada como aplicación de arranque pueda ser ejecutada. Entonces, para que una aplicación pueda trabajar cada que se energiza el cRIO, es necesario tener en la posición OFF al interruptor NO APP.

El programador del cRIO puede definir la función que va a desempeñar el interruptor USER1. Para poder definir su función, es necesario leer su estado dentro de la aplicación que se encuentre ejecutando en el MCU, además usar las herramientas que brinda el módulo LabVIEW Real Time.

Como se puede intuir por los comentarios anteriores y por el nombre, el botón Reset permite reiniciar el sistema, lo que equivale a apagar y prender el cRIO.

Conexión del cRIO 9012 a una red o PC

La conexión del cRIO a una red o una PC se realiza mediante el puerto Ethernet con interfaz física RJ-45, la ubicación de este puerto se puede apreciar en la figura 3.3. Se puede usar un cable de categoría 5, un tipo de cable de par trenzado cuya categoría es uno de los grados de cableado estandarizado, permite transferir 100 Mb por segundo a una frecuencia máxima de 100 MHz.

En la tabla 3.1 se muestran los dos tipos de cable que podemos implementar: cableado normal, conector 1 y conector 2 normal; cableado cruzado, conector 1 y conector 2 cruzado. La numeración de las terminales se aprecia en la figura 3.4.

Pin	Conector 1	Conector 2 Normal	Conector 2 Cruzado
1	Blanco/Naranja	Blanco/Naranja	Blanco/Verde
2	Naranja	Naranja	Verde
3	Blanco/Verde	Blanco/Verde	Blanco/Naranja
4	Azul	Azul	Azul
5	Blanco/Azul	Blanco/Azul	Blanco/Azul
6	Verde	Verde	Naranja
7	Blanco/Café	Blanco/Café	Blanco/Café
8	Café	Café	Café

Tabla 3.1. Cableado para conexiones Ethernet.

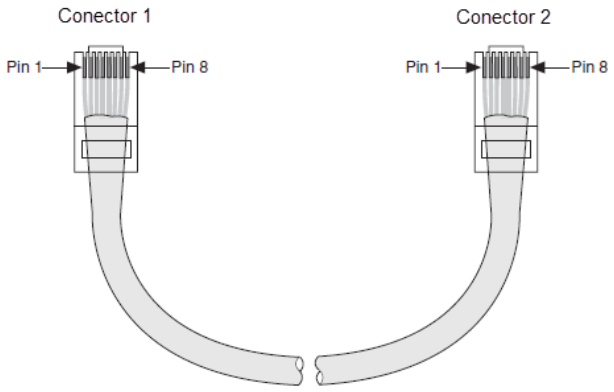


Figura 3.4. Distribución de terminales en conector RJ-45.

Para el caso de la presente tesis, se hizo una conexión directa entre PC y cRIO 9012 usando un cable cruzado blindado. La PC se comunica con el cRIO haciendo uso de una conexión estándar Ethernet. Para nuestro caso que tenemos una conexión directa, no existen restricciones para asignar las direcciones IP, pero para hacer eficiente la comunicación, las direcciones IP se asignan como si pertenecieran a la misma sub red. El cRIO tiene la dirección 169.254.88.15 y la PC 169.254.88.14.

cRIO pude trabajar con direcciones IP estáticas o dinámicas. Para el desarrollo de la presente tesis se optó por una IP estática, se asigna la primera vez que se configura el cRIO desde la aplicación Measurement & Automation Explorer (MAX) y jamás cambia. La aplicación MAX viene en cualquier versión de LabVIEW.

El cRIO 9012 soporta dispositivos de almacenamiento USB, ejemplo de ellos son las memorias USB Flash, discos externos con conectividad USB o dispositivos con conexión IDE pero que cuentan con un adaptador USB. Todos estos dispositivos deben

estar formateados con un sistema de archivos FAT 32, lo anterior es para que tanto el cRIO como las computadoras con WINDOWS, MAC OS y LINUX (SOs más usados) puedan procesar la información. Se pude conectar los diferentes dispositivos USB al cRIO mientras éste está encendido o en su defecto antes de encenderlo. Para acceder al dispositivo conectado, en LabVIEW se debe usar la ruta u:\directoriox. El puerto USB se puede apreciar en la figura 3.3.

El puerto RS-232 permite la comunicación serie, este tipo de comunicación no fue explotada en el presente trabajo así que no se profundizará en el tema.

3.2.2. Chasis reconfigurable basado en FPGA

El *chasis* reconfigurable, mostrado en la figura 3.5, es el corazón del sistema, en él se encuentra el núcleo reconfigurable que administra las entradas y salidas. Cuando se habla de núcleo, se hace referencia al FPGA que tiene en su interior, este tiene conexiones individuales a cada módulo de entrada y salida. El FPGA cuenta con funciones para escritura/lectura que permiten, entre otras cosas, mandar información a los módulos entrada/salida o recibir información de los mismos. Debido a que no hay un *bus* de comunicación compartido entre el FPGA y los módulos entrada/salida, se pueden sincronizar con precisión operaciones de entrada/salida en cada módulo con 25 ns de resolución. El FPGA puede realizar procesamiento de señales en formato entero o punto fijo, tomar de decisiones y pasar señales de un módulo a otro.

Figura 3.5. Chasis reconfigurable NI 9113.

No debemos preocuparnos mucho por el hecho de que el FPGA sólo opere datos en formato entero o punto fijo, el MCU es capaz de convertir los datos que vengan del FPGA (en formato entero o punto fijo) a otros formatos.

El rango de temperatura ambiente, para que el *chasis* 9113 logre trabajar adecuadamente, es el mismo que el cRIO 9012, va de -40 a 70 °C. Al *chasis* 9113 se le debe proporciona dos líneas de alimentación 5 V y 3.3 V, la primera línea permite un consumo máximo de 500 mW y para el segundo caso puede llegar a ser 2,800 mW. En

total se tiene un consumo máximo de 3,300 mW. Como ya se había comentado, la potencia para que el *chasis 9113* pueda operar está a cargo del cRIO 9012.

El FPGA está conectado al MCU de tiempo real a través de un bus local PCI. El MCU puede pedir información de cualquier control o indicador presente en el VI que corre en el FPGA, para ello se puede hacer uso de funciones escritura/lectura o del conjunto de herramientas *scan interface*, ambos conjuntos de herramientas son proporcionadas por el módulo LabVIEW FPGA.

El FPGA es capaz de generar peticiones de interrupción (IRQs, por sus siglas en inglés) para sincronizar la ejecución del software en el MCU con el FPGA. Para propósitos de esta tesis, usaremos otro método de sincronización, éste será la comunicación DMA y sus respectivas funciones de escritura/lectura.

En la figura 3.5 se puede apreciar que contamos con cuatro ranuras, cada una identificada con su número y con su propio conector. Cada conector de los que se aprecian en la figura, es capaz de funcionar como interfaz física entre un módulo E/S y el FPGA. En la misma figura vemos que solo existen 4 conectores, entonces, con el chasis 9113 sólo es posible trabajar 4 módulos E/S.

El FPGA que tiene incorporado el chasis 9113 es el Virtex-5 LX50 de Xilinx, a este dispositivo ya le fue dedicado una sección en el capítulo 2. Esta arquitectura de FPGA permite implementar gran cantidad de lógica por *slice*, brinda la oportunidad de que LabVIEW FPGA tome ventaja de su estructura y mejore la utilización de recursos en hardware. Lo anterior permite llevar a cabo una gran cantidad de operaciones en un sólo ciclo de reloj.

La base de tiempo para el FPGA es generada adentro del chasis 9113 y se encuentra trabajando a 40 MHz, en el peor de los casos, la base de tiempo trabaja con una exactitud de 100 ppm y con *jitter* de 250 ps.

En la figura 3.6 se puede apreciar el proceso de ensamble entre el cRIO 9012 y el chasis reconfigurable 9113.

3.2.3. Sistema de conversión analógico-digital

Para fines de esta tesis, de los 4 conectores para módulos E/S sólo se usará el conector 1. En este se conectará un módulo de entradas analógicas NI 9205, el cual es prácticamente un módulo de conversión analógico digital. En la figura 3.7 podemos ver el NI 9205 con sus dos tipos de conector, para esta tesis se usará el conector D-subminiature 37 (DC-37).

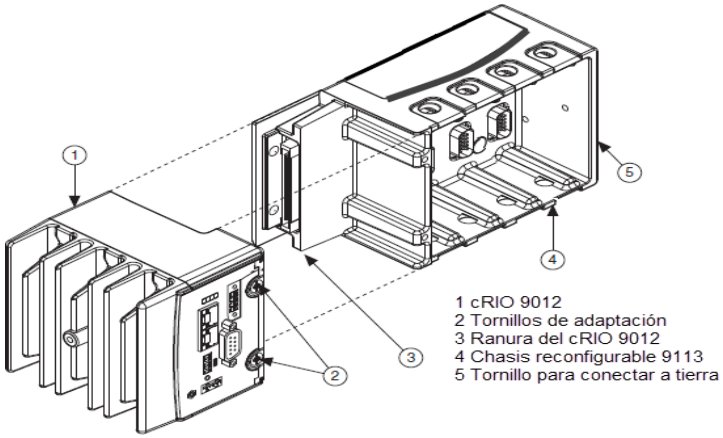


Figura 3.6. Ensamble de NI cRIO 9012 con chasis reconfigurable NI 9113.

Figura 3.7. Módulo de entradas analógicas NI 9205.

El 9205 es capaz de recibir 32 señales unipolares o 16 señales diferenciales, todas estas señales son analógicas. Cuenta con un convertidor analógico digital de 16 bits de resolución y con una máxima tasa de muestreo de 222 kS/s. En el esquema interno del 9205 mostrado en la figura 3.8 se puede apreciar los bloques que lo conforman.

La figura 3.8 nos indica que todos los canales analógicos se encuentran conectados a un multiplexor, esto permite procesar varias señales analógicas con sólo un amplificador de ganancia programable (PGIA, por sus siglas en inglés). La señal que se presente en las terminales del PGIA puede ser amplificada, la ganancia que se quiere proporcionar se establece mediante software, en esta tesis no amplificaremos las señales. Una vez que el PGIA ha procesado la señal, esta es recibida por el ADC y el circuito de disparo interno (Trigger en inglés). El circuito de disparo interno debe ser programado por software, puede responder a varias situaciones que llegue a presentar

la señal, su principal limitante viene dada por las funciones de software para leer su estado, éstas en general no responden a frecuencias superiores a los 30 KHz. El ADC procesa la señal analógica y posteriormente transmite la información resultante al FPGA que se encuentra en el *chasis* 9113.

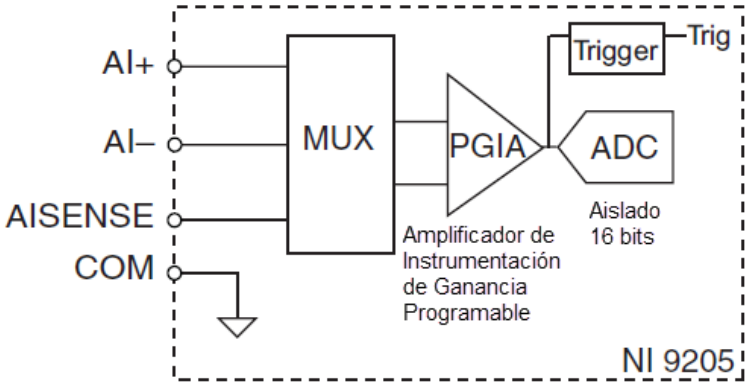


Figura 3.8. Circuito de entrada para un canal analógico del NI 9205.

La distribución de los canales (AIx) en el módulo 9205 se presenta en la figura 3.9. En la imagen se pude ver que, adicional a los 32 canales analógicos, se tiene una entrada digital (PFIO), una salida digital (DO0), dos terminales llamadas COM y una llamada AISENSE. La terminal COM es la conexión a la referencia del sistema de medición. La terminal AISENSE permite tener una conexión directa a la terminal inversora del PGIA. En los siguientes párrafos se profundiza al respecto.

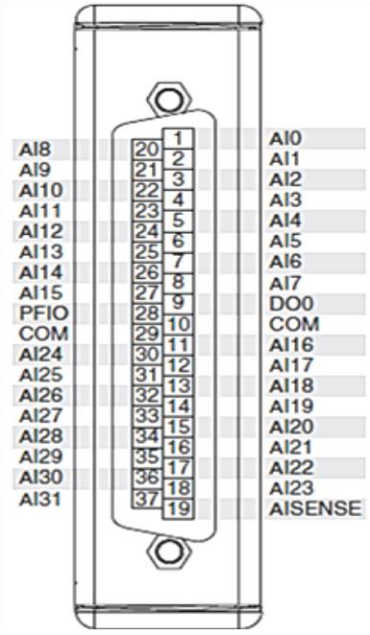


Figura 3.9. Terminales del módulo NI 9205.

Debido a que los canales digitales no son utilizados en la presente tesis, no se hará una descripción de ellos.

Medición en modo diferencial

Para cualquier medición diferencial, el número de canales disponibles en el NI 9205 se reduce a la mitad. En general este tipo de medición es la que otorga la mayor exactitud y menor cantidad de ruido. En la tabla 3.2 se muestra los pares AI que son válidos para medición diferencial.

Canal	Señal+	Señal-	**Canal**	Señal+	Señal-
0	AI0	AI8	**16**	AI16	AI24
1	AI1	AI9	**17**	AI17	AI25
2	AI2	AI10	**18**	AI18	AI26
3	AI3	AI11	**19**	AI19	AI27
4	AI4	AI12	**20**	AI20	AI28
5	AI5	AI13	**21**	AI21	AI29
6	AI6	AI14	**22**	AI22	AI30
7	AI7	AI15	**23**	AI23	AI31

Tabla 3.2. Pares para medición diferencial.

Como se puede apreciar en la figura 3.8, el sistema de medición cuenta con un PGIA. Al ser un amplificador de instrumentación, podemos asegurar que tenemos un amplificador diferencial con alta relación de rechazo al modo común (CMRR, por sus siglas en inglés). El que sea un amplificador de ganancia programable, nos indica que es capaz de modificar su ganancia mediante software. Lo anterior permite contar con un amplificador diferencial con características mejoradas, lo que se convierte en una herramienta de gran utilidad para los fines que persigue la presente tesis.

En general, un sistema de medición diferencial, como el mostrado en la figura 3.10, no tiene ninguna de sus entradas conectadas a una referencia fija, como tierra física o tierra del sistema.

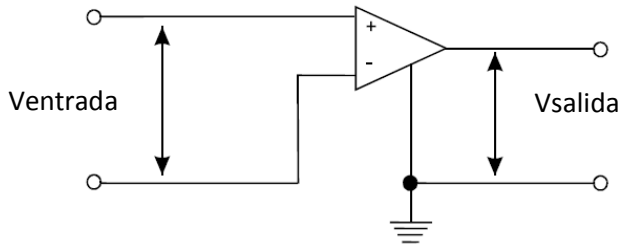


Figura. 3.10. Medición diferencial.

La medición diferencial es muy útil porque, el ruido que se induce de igual manera en cada una de las líneas que transportan la señal, aparece como un voltaje en modo común en la entrada del amplificador y este puede ser rechazado en gran medida por las características de los amplificadores diferenciales.

Voltaje en modo común

Idealmente un sistema de medición diferencial mide sólo la diferencia de potencial entre su terminal positiva y su terminal negativa. Cuando se mide una señal usando entradas diferenciales, hay un voltaje que está presente en ambas líneas de entrada y que se mide respecto a la referencia del sistema de medición, a este voltaje se le conoce como voltaje en modo común. Éste se muestra en la figura 3.11.

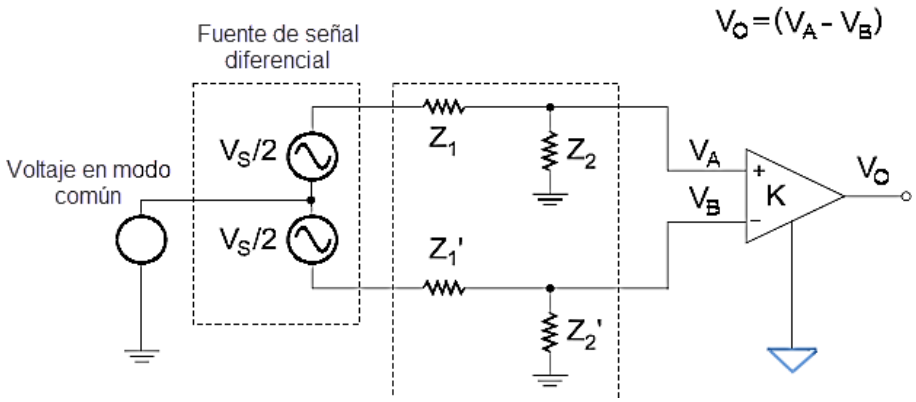


Figura 3.11. Voltaje en modo común.

El voltaje en modo común V_{mc} puede ser calculado con la siguiente ecuación[7]:

$$V_{mc} = \frac{V_A + V_B}{2} \tag{3.1}$$

donde:

V_A = el voltaje con respecto a la referencia del amplificador de instrumentación, que se encuentra en la terminal no inversora del sistema de medición

V_B = el voltaje con respecto a la referencia del amplificador de instrumentación, que se encuentra en la terminal inversora del sistema de medición

Relación de Rechazo al Modo Común

Idealmente, un amplificador diferencial rechazaría completamente cualquier voltaje en modo común presente en sus terminales y sólo amplificaría la diferencia de potencial entre ellas. Sin embargo, en realidad estos amplificadores no logran rechazar

[7] Park J., Steve M., Practical Data Acquisition for Instrumentation and Control Systems, Newnes, Perth, 2003, 51.

del todo al voltaje en modo común. La CMRR mide la capacidad que tiene un amplificador de entrada diferencial para rechazar señales que son comunes a las dos señales de entrada.

La CMRR está definida como la relación que existe entre la señal en modo común presente en la entrada del amplificador y la señal producida por este voltaje en la salida del amplificador, tal y como lo muestra la siguiente ecuación[8]:

$$CMRR = 20 \log_{10} \frac{V_{mc}}{V_{out}} \tag{3.2}$$

La CMRR, normalmente expresada en decibeles (dB), puede ser usada para calcular el error del voltaje de salida, el cual ocurriría debido a un voltaje en modo común que aparece en la entrada. Un alto CMRR mejora el rechazo de señales en modo común y brinda salidas más exactas.

Los sistemas de medición diferencial tienen otra limitación, existe un voltaje de entrada en modo común máximo y otro mínimo permisible en cada entrada, ambos medidos con respecto a la referencia del sistema. Aplicar un voltaje en modo común fuera del rango permitido puede generar un error en la lectura y hasta un daño al circuito.

Cuando se va a realizar una medición en modo diferencial con el NI 9205, se requiere dos terminales AI y una terminal COM, la forma de conectar COM dependerá de si la señal a medir se encuentra o no referenciada.

Fuentes de señal

En el presente trabajo se considerarán que existen dos categorías principales de fuentes de señal: referenciadas a tierra y las no referenciadas a tierra (también llamadas flotadas). Las fuentes de señal referenciadas a tierra son aquellas en las que las señales de salida están referencias a un sistema de tierra. Un sistema de tierra puede ser en general una estructura conductora. Por otro lado, una fuente de señal flotada es aquella en la que la señal no está referenciada a un sistema de tierra. Un ejemplo de una fuente de señal flotada son las baterías.

Medición diferencial de señales no flotadas (con referencia)

Una señal referenciada se mide de mejor manera con un sistema de medición diferencial, tal y como se muestra en la figura 3.12. En esta configuración, cualquier diferencia de potencial (∆Vg) entre la referencia de la fuente de la señal y la referencia

[8] Ibídem.

del sistema de medición, aparece como un voltaje en modo común para el sistema de medición. El voltaje diferencial medido se define como[9]:

$$V_m = (V_S + \Delta V_g) - \Delta V_g = V_S \tag{3.3}$$

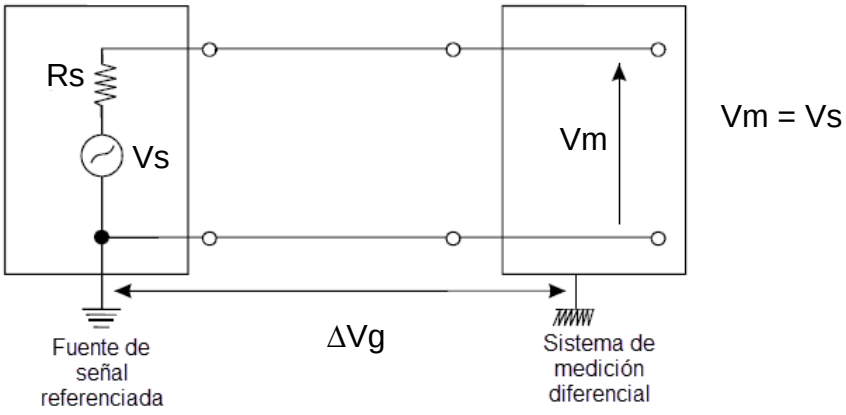


Figura 3.12. Medición diferencial de señal referenciada.

En la ecuación 3.3 se han despreciado la caída de tensión que podría generar Rs y otros efectos debidos a la interconexión.

Para una medición en modo diferencial de una señal referenciada, la conexión del NI 9205 debe verse como se muestra en la figura 3.13.

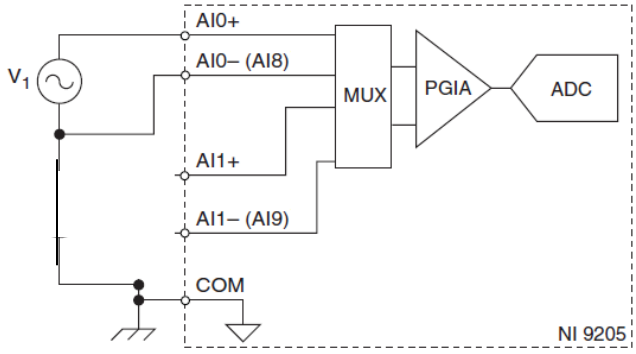


Figura 3.13. Conexión al NI 9205.

La figura 3.13 muestra la forma de hacer una conexión entre una fuente de señal referenciada y las conexiones diferenciales disponibles en el NI 9205. En la figura se ha usado el par diferencial 1, éste y el resto de pares diferenciales se pueden identificar en la tabla 3.2.

[9] Park J., Steve M., Practical Data Acquisition for Instrumentation and Control Systems, Newnes, Perth, 2003, 52.

Medición diferencial de señales flotadas (sin referencia)

Cuando se usa un sistema de medición diferencial para medir el voltaje de una señal flotada, se debe tener cuidado en asegurarse de que el nivel de voltaje en modo común, de la señal con respecto a la referencia del sistema de medición, no excede el límite de voltaje en modo común a la entrada del sistema de medición. Además, si no existe un camino de retorno a la tierra del sistema de medición para las corrientes de polarización de entrada del amplificador de instrumentación, y se consideran las cargas de las capacitancias parásitas, se pude tener que el nivel de voltaje de la señal a medir flote más allá del rango válido a la entrada del sistema de medición. Este fenómeno se acentúa cuando la impedancia de salida de la fuente que genera la señal es alta. El grado en el que el voltaje de la señal a medir va a flotar, depende de la magnitud de las corrientes de polarización de entrada y del balance del sistema.

Un sistema balanceado sigue el siguiente criterio[10]:

- Las impedancias de entrada a tierra de cada terminal del amplificador de instrumentación son iguales.
- Las impedancias a tierra de cada cable de la señal son iguales.
- Las impedancias a tierra de cada terminal de la fuente que genera la seña son iguales.

Un sistema balanceado también permite alcanzar inmunidad al ruido, ya que los voltajes de ruido inducidos que aparecen en los alambres que llevan la señal son iguales y deberían ser cancelados por el amplificador diferencial.

Existen mecanismos para lograr que el esquema de interconexión mostrado en la figura 3.14 conforme un sistema balanceado. Uno de esos mecanismos tiene que ver con el uso de dos resistencias. Estos resistores son llamados en la literatura resistores de polarización. Los resistores de polarización se conectan entre cada terminal de entrada y la referencia del sistema de medición. Éstos proporcionan, a las corrientes de polarización presentes en las entradas del amplificador de instrumentación, un camino de retorno a la referencia de medición.

Cuando la señal está acoplada en corriente directa (DC, por sus siglas en inglés), porque contiene componentes de corriente directa y corriente alterna (AC, por sus siglas en inglés), y además la fuente de la señal tiene baja impedancia a la salida, solamente un resistor de polarización necesita ser conectado entre la entrada negativa y la referencia. Si la impedancia de salida de la fuente es relativamente alta comparada con la impedancia de entrada del amplificador de instrumentación, el uso de una sola

[10] Park J., Steve M., Practical Data Acquisition for Instrumentation and Control Systems, Newnes, Perth, 2003, 54-55.

resistencia puede llevarnos a resultados erróneos, hay que usar las dos resistencias. Por otro lado, cuando las señales no tienen componente de DC (acopladas en AC), ambos resistores son requeridos[11].

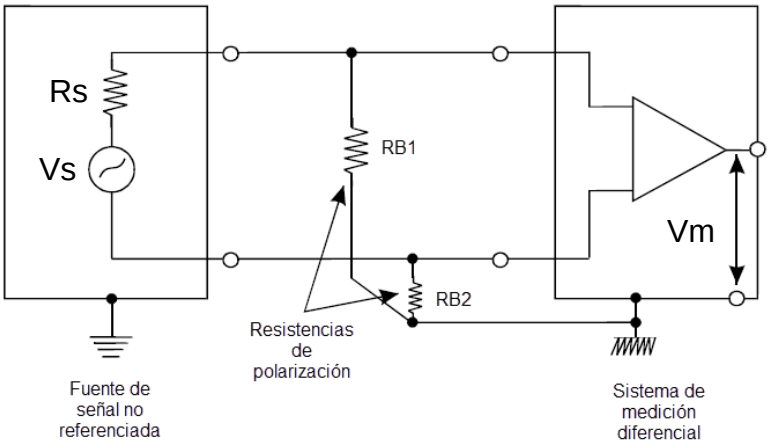


Figura 3.14. Medición diferencial de señales flotadas.

Los resistores de polarización deben ser lo suficientemente grandes para permitir que la señal flote respecto a la referencia del sistema de medición y para no cargar a la fuente de la señal (deben ser más grandes que la impedancia de salida de la fuente), pero lo suficientemente pequeñas para mantener el voltaje en cada terminal de entrada dentro del rango de voltaje en modo común que admite el sistema de medición. Los resistores de polarización suelen tener valores entre 10 kΩ y 100 kΩ[12].

Para hacer una medición en modo diferencial de una señal no referenciada con el NI 9205, se debe tener una configuración como la que se muestra en la figura 3.15.

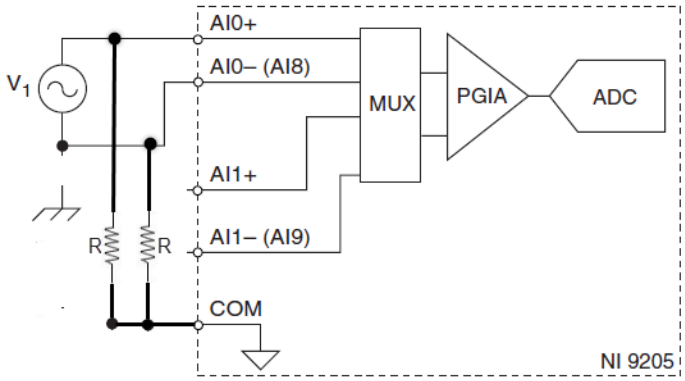


Figura 3.15. Conexión al NI 9205.

[11] Ibídem.
[12] Ibídem.

Medición en modo terminación simple

Cuando la fuente de señal sólo usa un alambre para transportar la información que representa a la señal, se dice que existe una señal de terminación simple. Este modo de transmitir señales es muy común actualmente. Tiene la ventaja de ser más económico que el mecanismo diferencial, pero tiene mayor susceptibilidad al ruido. Para realizar una medición de una señal de terminación simple, suele emplearse una medición en modo terminación simple. Debido a que la fuente de señal de terminación simple tiene un alambre para la señal y otro para la referencia, la medición en modo terminación simple debe contar con dos conexiones también.

El NI 9205 proporciona dos opciones para medición en modo terminación simple. Si la referenciada de la fuente de la señal se conecta directamente a la referencia del sistema de medición, entonces, se tiene una medición en modo terminación simple referenciada (RSE, por sus siglas en inglés), por otro lado, si se va a medir una señal referenciada pero la referencia de la fuente de la señal se conecta directamente a la terminal inversora del PGIA, entonces, se tiene una medición en modo terminación simple no referenciada (NRSE, por sus siglas en inglés).

Medición en modo terminación simple de señales referenciadas usando COM

Cuando un sistema de medición referenciado se usa para medir señales referenciadas, tal y como se muestra en la figura 3.17, pueden ocurrir problemas de medición. En esta configuración, cualquier diferencia de potencial (∆Vg) entre la referencia de la fuente de la señal y la referencia del sistema de medición se suma al voltaje que representa a la señal, lo que convierte a ∆Vg en parte de la medición. El voltaje medido se representa con la siguiente ecuación:

$$V_m = V_S + \Delta V_g \tag{3.4}$$

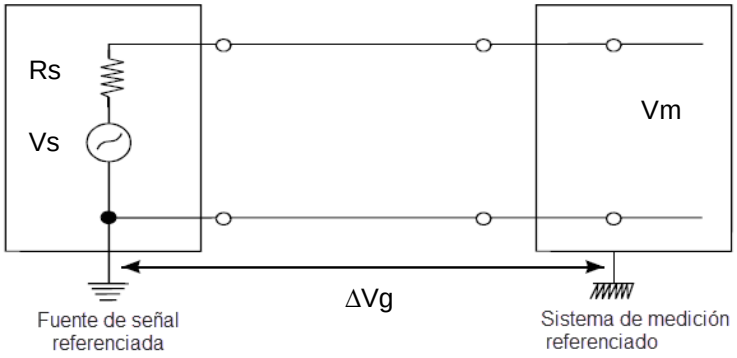


Figura 3.17. Medición en modo terminación simple de una señal referenciada.

En la ecuación 3.4 se han despreciado la caída de tensión que podría generar Rs y otros efectos debidos a la interconexión.

Si los niveles de voltaje de la señal son altos, en comparación con la diferencia de potencial entre las referencias (ΔVg), y los cables entre la fuente y el sistema de medición tienen baja impedancia, entonces, las inexactitudes en la medición son aceptables y este tipo de medición se puede implementar.

Al trabajar con el sistema cRIO y en especial con el NI 9205, existen casos en los que este tipo de medición es muy útil y económica, por ejemplo: cuando los 32 canales comparten la misma referencia o cuando se necesita hacer la medición de 32 señales y no se quiere recurrir a otro módulo NI 9205.

En la figura 3.18 se puede apreciar la forma de realizar la conexión entre una señal y el módulo NI 9205, si es que se desea hacer una medición en modo terminación simple de una señal referenciada, haciendo uso del NI 9205. En la figura, la referencia de la señal ha sido conectada a la referencia del sistema de adquisición mediante la terminal COM.

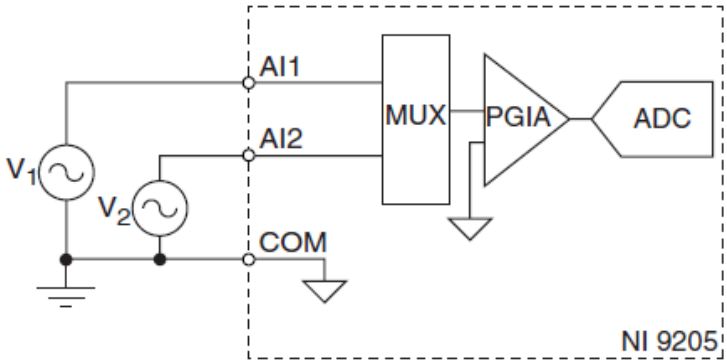


Figura 3.18. Medición usando el NI 9205.

En una medición RSE, como las que se muestran en la figura 3.18, los niveles de voltaje en las terminales AI1 y AI2 se miden con respecto a la referencia interna del NI 9205. En caso de que se dejara sin conectar COM, las señales flotarían fuera del rango de trabajo del NI 9205. No hay forma de asegurar que la señal de entrada está dentro de los ±10 V con respecto a COM.

Medición en modo terminación simple de señales referenciadas usando AISENSE

Las mediciones NRSE se diferencian de las mediciones RSE en una primera instancia por el uso de la terminal AISENSE del módulo NI 9205. Otra diferencia marcada es que con el uso de AISENSE existe una disminución de ruido (incluyendo ΔVg) en las mediciones. Al tener una conexión a la terminal inversora del PGIA, permite hacer una medición parecida a la diferencial. La similitud entre mediciones RSE y NRSE es que ambas permiten medir los 32 canales casi simultáneamente.

En la figura 3.19 se puede apreciar la forma de conectar las terminales del NI 9205 para realizar mediciones NRSE.

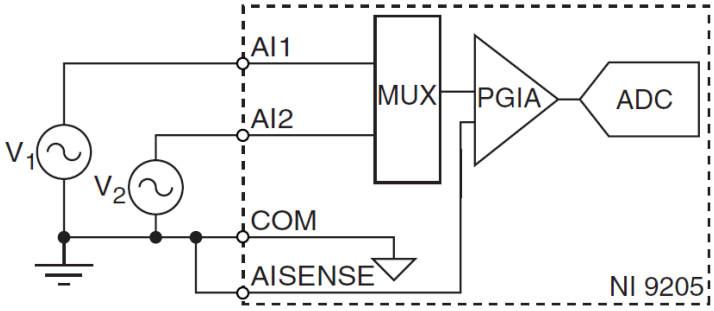


Figura 3.19. Conexión para medición NRSE.

En una medición NRSE, como la que se muestra en la figura 3.19, los niveles de voltaje en las terminales AI1 y AI2 se miden con respecto a AISENSE.

Si queremos realizar una medición en modo terminación simple con el NI 9205, se debe usar una terminal AIx y la terminal COM o AISENSE, donde estas últimas son la referencia.

Otras características de operación

La CMRR del módulo NI 9205 cuando un par de canales se encuentren operando en modo diferencial se puede apreciar en la figura 3.20.

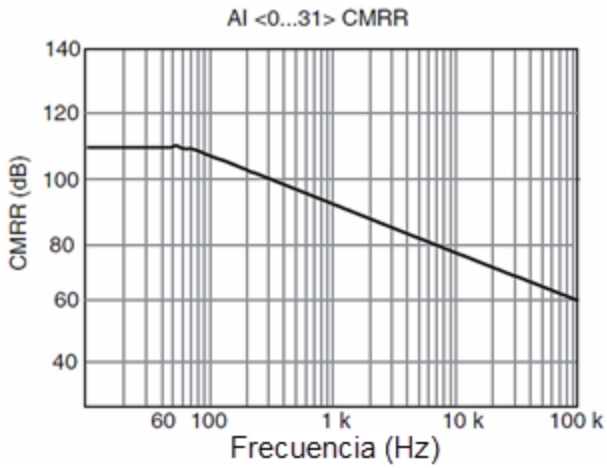


Figura 3.20. Gráfica del CMRR típico entre los canales A1+ y A1-.

El módulo 9205 cuenta con inmunidad al ruido y protección contra transitorios de 1,000 Volts eficaces. Soporta operar a una temperatura ambiente que se encuentre entre los -40 y 70°C, las mismas condiciones de temperatura que rigen la correcta operación del cRIO 9012 y el chasis 9113.

Para cada canal del NI 9205, se tiene un sistema de protección que le permite a cada canal soportar voltajes fuera del rango de operación normal, esto con la finalidad de prevenir su daño ante una mala conexión. Por cada canal, el NI 9205 puede soportar en su entrada hasta 42.4 Volts pico o ±60 Volts de corriente directa, ambos referenciados a tierra física. Si el COM no se encuentra a tierra física, el voltaje máximo en corriente directa es de ±30 V. La corriente de polarización de entrada es de ±100 pA.

Entre cada canal (AIx) y el COM, al estar energizado el 9205, existe una impedancia de entrada mayor a 10 GΩ en paralelo con 100 pF, y al estar apagado, existe una impedancia de 4.7 kΩ como mínimo. Todos los canales analógicos del NI 9205 se encuentran acoplados en DC.

La máxima potencia que puede llegar a consumir el NI 9205, y que es proporcionada por el cRIO 9012, es de 625 mW.

Cada canal del NI 9205 tiene rangos de entrada programables, esto quiere decir que mediante software podemos fijar el rango de tensiones que permite pasar el 9205, estos rangos pueden ser ±200 mV, ±1 V, ±5 V, y ±10 V.

Para el NI 9205 existe la posibilidad de programar el tiempo de conversión. La precisión que se puede obtener en las mediciones dependerá en gran medida del tiempo de conversión establecido, el valor mínimo para éste es de 4.5 µs. Cuando usamos estos 4.5 µs, se puede obtener una precisión de ± 120 ppm por paso a escala completa o visto de otra forma ± 8 bits menos significativos (LSB, por sus siglas en inglés). El fabricante también proporciona el dato para 8 µs, este es de ± 30 ppm por paso a escala completa o ± 2 LSB[13].

En la tabla 3.3 se puede apreciar los coeficientes de escalamiento típicos que da el fabricante, según el rango de entrada establecido. Se observa que el rango que otorga la mayor precisión en las mediciones es el de ± 200 mV, esto era de esperarse debido a que el rango de medición ha disminuido, y al igual que en los casos anteriores, se tienen 65,536 (2^{16}) valores para representar los voltajes.

Voltaje Nominal [V]	Coeficiente de escalamiento típico (µV/LSB)
±10	328
±5	164.2
±1	32.8
±0.2	6.57

Tabla 3.3. Coeficientes de escalamiento[14].

[13] National Instruments, NI 9205 operating instructions and specifications, 2008, 20-21.
[14] Ibídem.

Por lo escrito en párrafos anteriores, es claro que el NI 9205 cumple con una gran parte de los requisitos de hardware mencionados al inicio de capítulo.

El chasis reconfigurable basado en FPGA, el MCU de tiempo real y los módulos entrada/salida se combinan para crear un sistema embebido completo y autónomo.

3.2.4. Computadora personal

Para configurar el sistema cRIO, es necesario establecer una comunicación entre el puerto Ethernet del cRIO 9012 y una PC. Como se ha descrito en párrafos anteriores, para comunicar al cRIO y la PC, se hará uso de un cable categoría 5 con conectores RJ-45.

El sistema cRIO tiene un accionar autónomo, no necesita estar conectado siempre a la PC; la conexión se hace necesaria cuando se quiere configurar el funcionamiento del sistema, se quiere interactuar con el software que gobierna al sistema o se desea transferir información entre las unidades de almacenamiento del cRIO y de la PC.

La comunicación mediante FTP y HTTP (páginas web), permite incorporar a casi cualquier PC comercial, indistintamente de su sistema operativo. Como se verá más adelante, el software de procesamiento de datos que reside en la PC ha sido diseñado para versiones de sistemas operativos Windows XP, 7 y 8, excluyendo el uso de otros sistemas operativos que no sean estos.

Considerando lo anterior y las herramientas disponibles, para la presente tesis se ha hecho uso de una computadora portátil marca DELL, modelo INSPIRON 14R. Esta computadora cuenta con: tercera generación del MCU Intel Core i3-2310M a 2.10 GHz con 2 MB Caché, 3 GB en dos canales SDRAM DDR3 a 1,600 MHz, disco duro de 320 GB a 7,200 Revoluciones Por Minuto (RPM) con interfaz Serial Advanced Technology Attachtment (SATA), adaptador de red integrado 10/100 Mbps y sistema operativo de 64 bits Windows 7 Home Premium Service Pack 1.

3.3. El software del sistema

El software para desarrollar el sistema, que es propósito de la presente tesis, es LabVIEW. Por lo escrito en el capítulo 2, se sabe que aparte del núcleo de LabVIEW usaremos los módulos LabVIEW FPGA y LabVIEW Real-Time. Estas herramientas, en conjunto, permiten contar con lo necesario para explotar al máximo los recursos de nuestro sistema cRIO; además, tienen la peculiaridad de otorgar un nivel de abstracción sin precedentes en la programación de MCUs y FPGAs. Este hecho es de suma importancia, debido a que nos permite incrementar la complejidad de las aplicaciones de software sin tener que preocuparte en demasía por la estructura del hardware, aunque nunca se puede perder de vista las limitantes del mismo. Tener conocimiento del hardware siempre resultará importante, nos puede ayudar a explotar mejor sus virtudes y tratar de que sus desventajas pasen desapercibidas.

Con la intención de ilustrar de mejor manera la herramienta de software asociada a cada parte del hardware, del cual se va a hacer uso, en la figura 3.21 se muestra que el núcleo de LabVIEW se encarga de administrar los recursos correspondientes a la PC, el módulo LabVIEW Real-Time es el encargado de administrar al cRIO 9012 (MCU) y por último, sumamente importante, el módulo LabVIEW FPGA se encarga de administrar al chasis NI 9113 (FPGA).

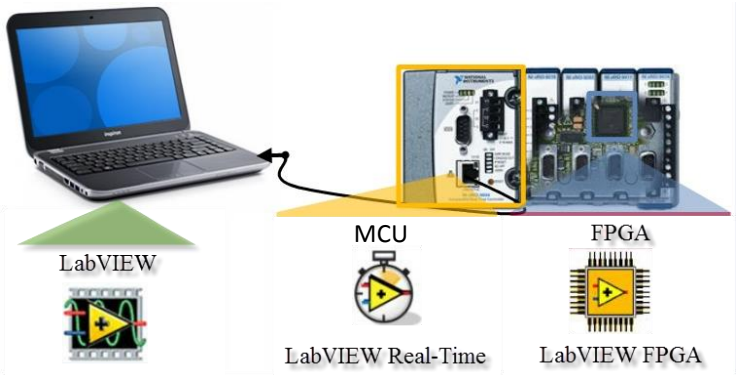


Figura 3.21. Software asociado al sistema cRIO.

3.3.1. Software de la PC

Para esta parte del software principalmente se consideran tres ventanas: la ventana *Menú*, la ventana *Procesamiento* y la ventana *FTP*. Cada una de estas ventanas será descrita considerando los requerimientos y su operación basada en diagramas de flujo.

Ventana Menú

Requerimiento

Se necesita una interfaz que permita acceder a los dos procesos principales que se realizan en la PC, la transferencia de archivos FTP y el procesamiento de la información derivada de la adquisición de datos. Se quiere que además de poder elegir entre los dos procesos mencionados, el usuario tenga la opción de cerrar la interfaz en cualquier momento. Adicionalmente, la interfaz debe de permitir visualizar si algún proceso ha sido elegido. Por último, la interfaz debe ser intuitiva para facilitar el uso de la misma.

Descripción de la operación

Con la finalidad de mostrar la operación de la ventana *Menú*, en la figura 3.22 se presenta el diagrama de flujo asociado a dicha ventana. Al empezar a ejecutarse la ventana *Menú,* el algoritmo pregunta por el control *Salir*. Mientras el usuario no haya presionado el control salir, la interfaz no se detiene. Posteriormente, cuando el algoritmo se ha dado cuenta de que el usuario no ha escogido salir, borra el contenido del indicador *Activa*. Se da paso a verificar si el usuario ha realizado alguna petición. En

dado caso de que el usuario haya seleccionado el control *Procesamiento*, se modifica el valor del indicador *Activa* y se manda a llamar a la ventana *Procesamiento*. De manera similar, cuando el usuario selecciona el control *FTP,* el algoritmo modifica el valor del indicador *Activa* y se manda a llamar a la ventana *FTP*. Finalmente se pregunta por el estado del control Salir, cuando éste ha sido presionado el algoritmo finaliza y la ventana *Menú* se cierra.

Según el diagrama de flujo los controles se leen en paralelo. Siempre que sea posible LabVIEW procesará en paralelo por defecto. Existe la posibilidad de forzar el procesamiento secuencial, para ello es necesario usar una *Flat sequence structure* o *Stacked sequence structure*.

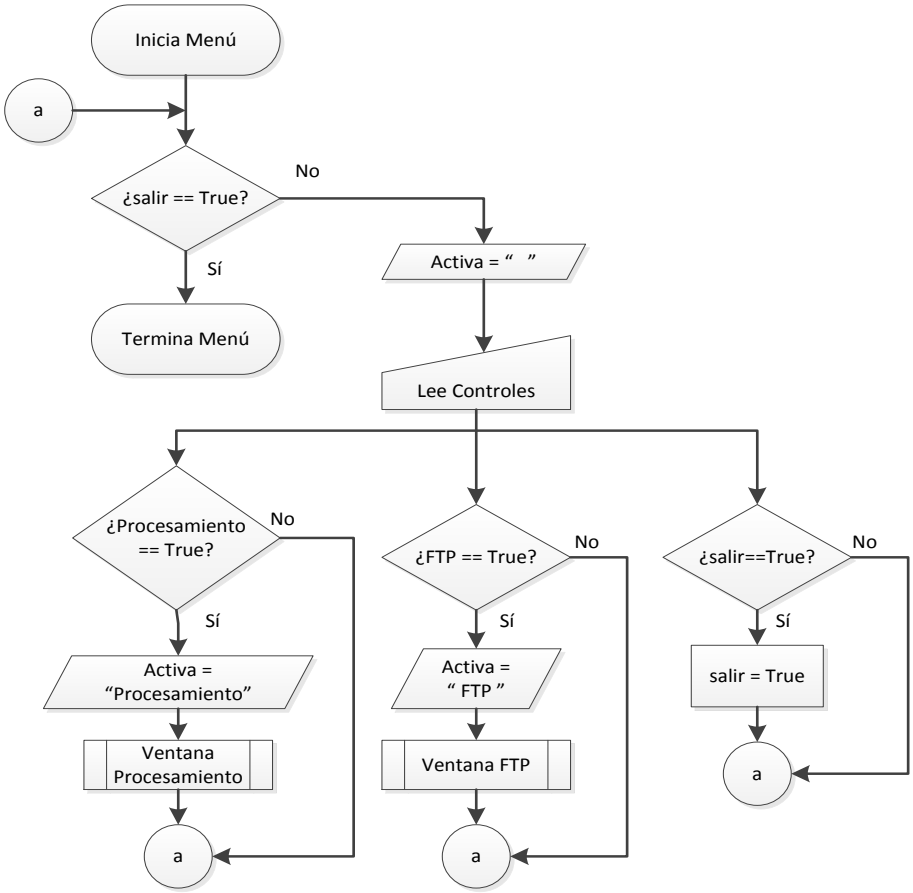


Figura 3.22. Diagrama de flujo de la ventana Menú.

Ventana Procesamiento

Requerimiento

Se quiere realizar el procesamiento de la información derivada de la adquisición de datos y que se encuentra almacenada en archivos binarios. El procesamiento debe girar en torno a los siguientes puntos:

- La operación de la interfaz debe ser intuitiva. Esto implica la existencia de pocos indicadores y controles. Además, para evitar que el usuario establezca parámetros incorrectos, la aplicación debe inhabilitar los controles cuando sea necesario y desplegar mensajes que indiquen al usuario si puede o no realizar una acción.
- La información referente a la configuración de la adquisición de datos debe ser desplegada y estar disponible mientras la venta esté en operación.
- Se debe poder desplegar en forma de gráfica y de tabla los datos de la adquisición. Estas dos formas de despliegue deben ser simultáneas.
- Se debe permitir que los gráficos generados sean manipulados. Esto implica que se pueda realizar: *zoom out* y *zoom in* de un área del gráfico definida por el usuario, modificar el tamaño del trazo, modificar el color del trazo, fijar auto escala, modificar la representación de los valores en ambos ejes y habilitación/inhabilitación de los trazos.
- Ya que la cantidad de datos a procesar puede saturar la memoria RAM de la PC, se necesita que el acceso a los archivos sea aleatorio.
- La aplicación debe poder desplegar la información de hasta cuatro canales simultáneamente.
- Debido a que la información se encuentra contenida en archivos binarios, se requiere que esta información pueda ser codificada en ASCII.

Descripción de la operación

El diagrama de flujo para describir la operación de la ventana procesamiento es extenso, es por ello que el diagrama se presenta en cinco partes. A sí mismo, cuando el diagrama contenga sub VIs, éstos serán desarrollados en un apartado diferente.

En la figura 3.23 se muestra el diagrama de flujo asociado a la primera parte de la ventana *Procesamiento*. Al iniciar la ventana se empiezan a ejecutar dos ciclos *while* en paralelo. El ciclo *while1*, se encarga de estar monitoreando el valor de tres controles: *Visualización, Configuración y Parar*. Debido a que la ventana cuenta con dos secciones principales, la parte de configuración y la parte de despliegue de datos de adquisición (visualización), los dos primeros controles son utilizados para ir de una sección a otra. El tercer control, *Parar*, permite interrumpir la ejecución de la ventana *Procesamiento*. El control *Parar* permite detener la operación de los dos ciclos *while*. Los tres controles antes mencionados se encuentran disponibles mientras la ventana se encuentre activa. Por otro lado, el ciclo *while2* es más extenso. Éste se encarga de ejecutar el resto del código correspondiente a la ventana. La primera acción que lleva a cabo el ciclo *while2* es fijar los valores por default de indicadores y controles. Éste continúa abriendo un cuadro de diálogo, mediante el cual el usuario puede elegir el archivo de configuración correspondiente. Se procede a mandar a llamar al sub VI *Lee*

Configuración. Cuando este VI termina su ejecución, comienza el accionar de un ciclo *for*. Este ciclo sirve para generar un arreglo de una dimensión, éste tendrá un tamaño igual al número de canales configurados. Paralelamente al ciclo *for,* se verifica si el modo de almacenamiento es disparo, si éste es el caso, se muestran los indicadores relacionados a este modo. Por último, en este diagrama de flujo, se ve que se deben modificar algunos controles e indicadores con información de la configuración.

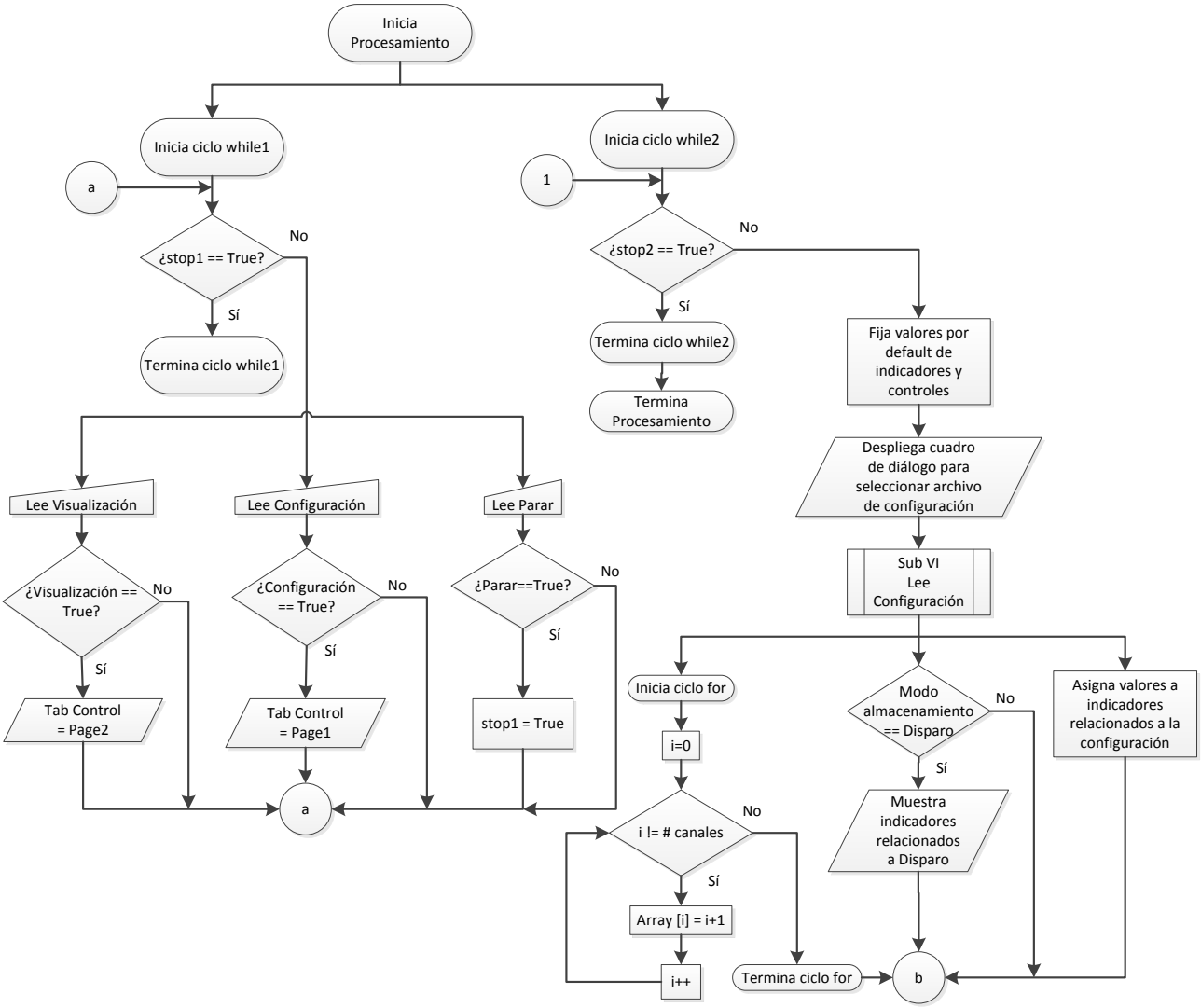


Figura 3.23. Primera parte del diagrama de flujo de la ventana Procesamiento.

En el diagrama de flujo de la figura 3.23 se manda a llamar al sub VI *Lee Configuración*. El diagrama de flujo asociado a este sub VI se presenta en la figura 3.24. Lo primero que hace el sub VI es leer la ruta del archivo de configuración, ésta ha sido previamente establecida por el usuario. Una vez que se ha identificado la ubicación del archivo, se procede a ejecutar la función *Open/Create/Replace File*. Esta función permite abrir una referencia al archivo. Creada la referencia al archivo, se ejecuta la

función *Read from Text File*. Esta función permite cargar el contenido del archivo en memoria. Posteriormente se cierra la referencia al archivo. En este punto ya se tiene disponible la información para ser procesada. Para manipular los datos leídos, se ejecuta la función *Spreadsheet String to Array*. Esta función permite depositar los datos en un arreglo de dos dimensiones. Se recorre el arreglo y a cada dato de éste se le da el formato adecuado. Ya que los datos han sido formateados, éstos son depositados en los controles e indicadores correspondientes. El sub VI ha terminado su ejecución.

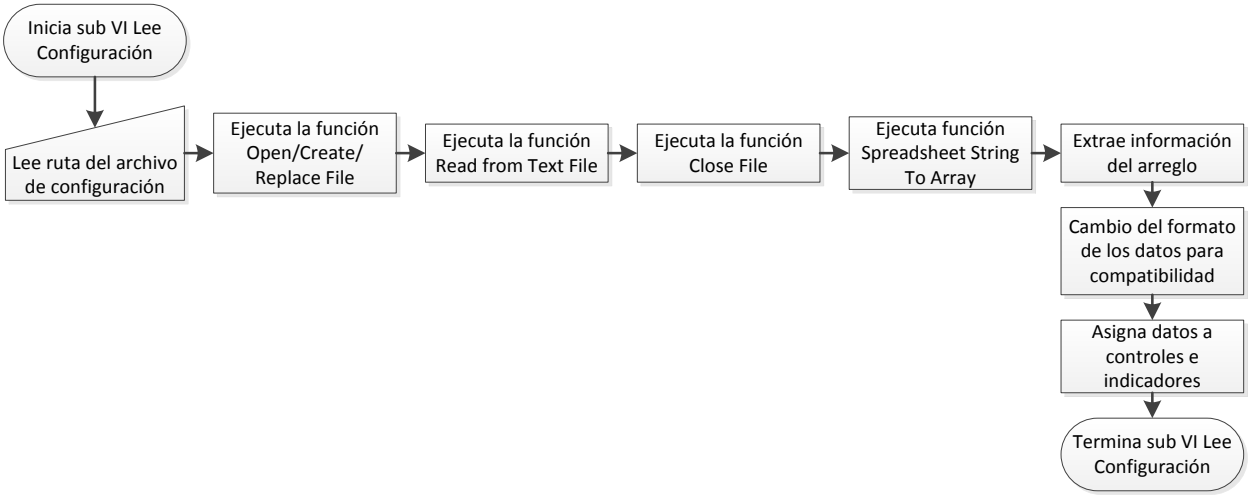


Figura 3.24. Diagrama de flujo del sub VI Lee configuración.

Ya que se ha descrito el sub VI *Lee Configuración* y se terminó con la descripción de la primera parte del diagrama de flujo de la ventana *Procesamiento*, se da paso a describir la segunda parte de la ventana mencionada.

En la figura 3.25, se presenta la segunda parte del diagrama de flujo correspondiente a la ventana *Procesamiento*. Esta parte inicia monitoreando si el usuario ha elegido realizar alguna acción. Las acciones que puede elegir son tres: leer archivo completo, leer archivo por partes o generar archivo con codificación ASCII. Mientras alguna acción no haya sido elegida, la aplicación seguirá en espera de que el usuario elija una. Cuando el usuario ya ha elegido una acción a realizar, se procede a reconocer la carpeta en donde se encuentra el archivo de configuración. Se debe recordar que la ubicación de esta carpeta ya ha sido establecida, figura 3.23. A continuación se despliega un cuadro de diálogo, éste permitirá elegir el archivo que contiene los datos de la adquisición. El cuadro de diálogo ya mostrará la carpeta que contiene a los archivos que se han generado en la adquisición de datos. Lo anterior se debe a que estos archivos se encuentran en la misma carpeta del archivo de configuración. Una vez seleccionado el archivo a procesar, la ruta de éste se pasa a una función encargada de abrir una referencia al archivo. Seguido de esto, se obtiene el tamaño del archivo en unidades de bytes. Posteriormente, como éste es un archivo del

tipo binario, en lugar de usar la función antes descrita para leer archivos de texto, se usa una función para leer archivos binarios, *Read from Binary File*. No se pone en memoria todo el contenido del archivo, sólo se extraen los primeros 64 bits del mismo. Sucedido lo anterior se cierra la referencia al archivo, tal y como se muestra en la figura 3.25. Ya se tienen los primeros 64 bits, estos bits contiene la información correspondiente a la hora y fecha en la que se inició la adquisición de datos almacenados en el archivo. Además de la información antes mencionada, se tiene la frecuencia de muestreo, el número de canales y el tamaño del archivo. Con estos datos se calcula la cantidad de segundos que tiene el archivo de adquisición. La segunda parte del diagrama de flujo asociado a la ventana *Procesamiento*, ha terminado.

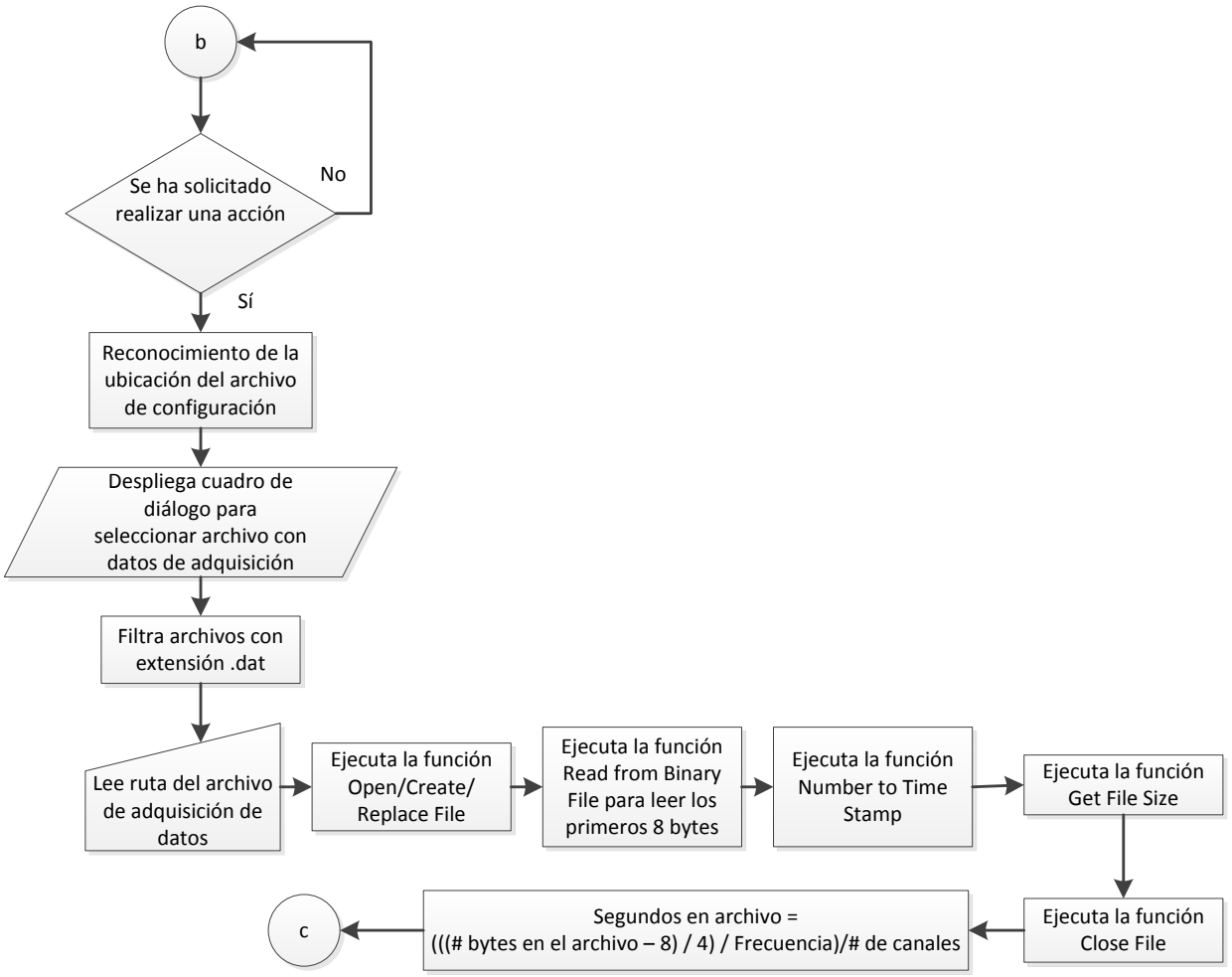


Figura 3.25. Segunda parte del diagrama de flujo de la ventana Procesamiento.

La tercera parte del código es bastante extensa, su objetivo es permitir que mediante cuadros de diálogo, controles e indicadores el usuario pueda establecer si quiere ver una porción del archivo, quiere verlo todo o quiere crear un archivo de texto codificado en ASCII.

Si el usuario quiere ver una porción del archivo, tiene la opción de seleccionar hasta cuatro canales diferentes y fijar la cantidad de segundos que desea procesar; evidentemente, como ya se había comentado, la lógica del programa bloquea el procesamiento en caso de que el usuario quiera establecer un parámetro sin sentido, por ejemplo un valor en segundos que excede el número de segundos contenidos en el archivo.

Cuando el usuario quiere procesar el archivo completo, debe tener en mente la cantidad de datos que representan los cuatro canales que se van a desplegar. Esto debido a que los datos que se quieran procesar tienen que ser cargados en memoria RAM. Entonces, si la memoria RAM de la PC que esté utilizando el usuario no tiene la capacidad suficiente para contener la cantidad de datos solicitada, el software mandará una mensaje de error relacionado a la memoria, lo que producirá que el funcionamiento de la PC no sea óptimo e inclusive se tenga que reiniciar la misma. Aunque el usuario sólo quiera procesar un canal, el sistema por *default* asigna el canal 1 en los otros tres espacios para seleccionar canal.

En caso de que el usuario quiera generar un archivo de texto con codificación ASCII, podrá hacerlo bajo dos situaciones, de hecho son situaciones similares a las planteadas para el despliegue, la principal diferencia radica en que para el despliegue se pueden escoger hasta cuatro canales y en la generación del archivo de texto sólo se puede escoger un canal. Las precauciones al elegir una u otra deben ser las mismas que se han mencionado en los párrafos anteriores, se debe contemplar el número de datos que se van a procesar en relación a la disponibilidad de memoria RAM de la PC.

La tercera parte del diagrama de flujo de la ventana *Procesamiento* se presenta en las figuras 3.26 y 3.27. En la primera de estas figuras se distingue el conector *c*, este conector es el enlace entre la segunda y la tercera parte del diagrama de flujo. Al iniciar esta parte del diagrama se pregunta por el control *Parar*. Si se ha presionado el control *Parar* se da paso a desplegar un cuadro de diálogo. En este cuadro de diálogo el usuario puede elegir entre tres opciones: permanecer en la ventana, iniciar un nuevo proceso dentro de la ventana o abandonar la ventana. En los casos dos y tres se retorna al inicio del ciclo *while2* a través del conector *1*. Para el primer caso mencionado se da un salto a otra parte del diagrama a través del conector *d*. Si el control *Parar* no ha sido presionado también se da un salto a través del conector *d*. Este conector lleva a leer el control *Elija una opción*. Mediante este control el usuario puede elegir si quiere leer todo el archivo, una porción del mismo o en su defecto, codificar la información en ASCII. Si el usuario decide leer todo el archivo, entonces, se procede a habilitar los controles necesarios para la selección de canales. Después de esto se abrirá un cuadro de diálogo, éste permitirá seleccionar el archivo de datos que se desea procesar. Se

preguntará al usuario si desea aplicar la constante de amplificación a los datos. Esta constante fue establecida en la configuración de la adquisición.

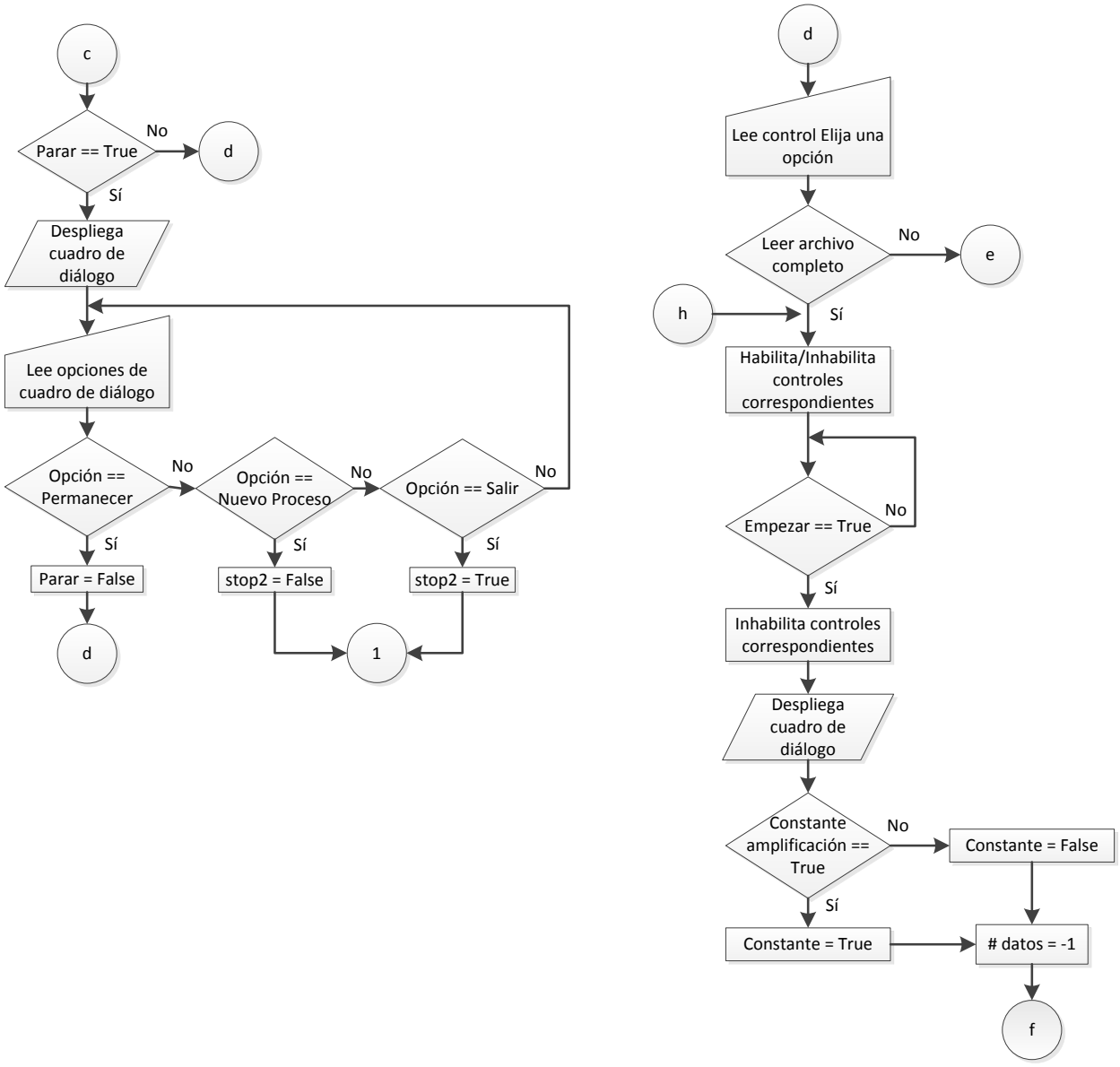


Figura 3.26. Tercera parte del diagrama de flujo de la ventana Procesamiento (1 de 2).

En la figura 3.27 se presenta el caso en el que el usuario elije que quiere leer el archivo por partes, entonces, se procede de manera similar al caso anterior. La diferencia radica en que se establecen dos parámetros adicionales, el segundo en que se quiere empezar a leer y la cantidad de segundos que se quiere visualizar. A continuación el sistema se asegura de que los parámetros establecidos sean válidos. La tercera acción disponible en el control *Elija una opción*, la que permite codificar la

información en ASCII, tiene un proceso similar al descrito para las otras dos opciones. Por lo anterior esta acción no requiere de mayor explicación.

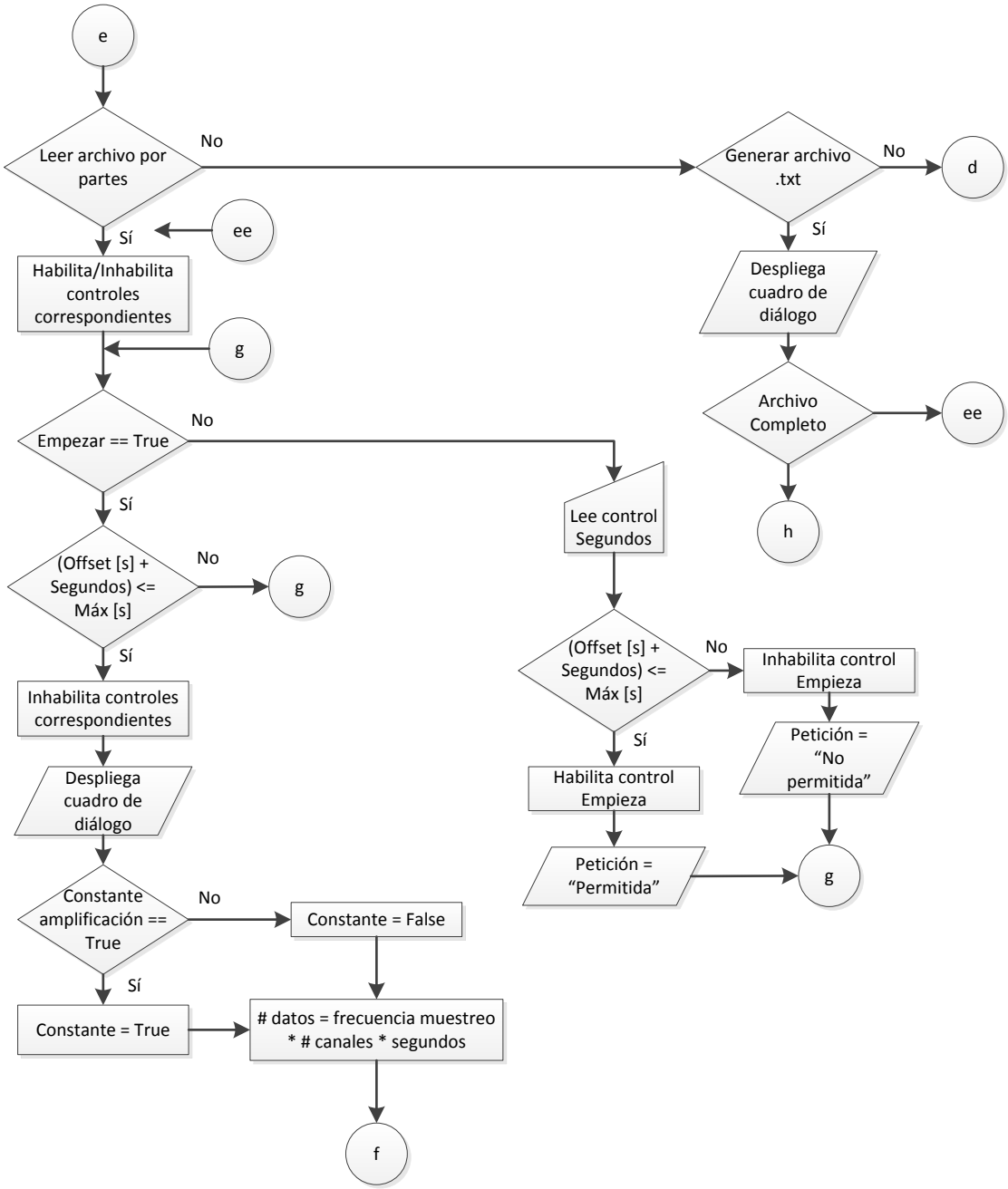


Figura 3.27. Tercera parte del diagrama de flujo de la ventana Procesamiento (2 de 2).

En los diagramas de las figuras 3.26 y 3.27 aparecen conectores con la letra f, éstos indican el enlace a la cuarta parte del código que a continuación se describe.

La cuarta parte del diagrama de flujo, figura 3.28, empieza calculando el número de bytes que se deben procesar, cada dato es de 32 bits o lo que es equivalente a 4 bytes, entonces:

$$\#\ \text{bytes a leer} = 4 * Fm * \#\ \text{Canales en archivo} * \#\ \text{Seg} + 8 \quad (3.5)$$

donde *Fm* se refiere a frecuencia de muestreo; # *Canales en archivo,* se refiere a la cantidad de canales que se adquirieron; y # *Seg,* son los segundos que desea procesar el usuario.

Posteriormente, en caso de ser necesario, se calcular el *offset* (desplazamiento en el archivo) que desea el usuario, para ello es necesario considerar que la información de hora y fecha se encuentran al principio de todos los archivos de datos y consume 8 bytes, entonces:

$$\text{Offset [bytes]} = 8 + \text{Offset [s]} * Fm * \#\ \text{Canales en archivo} * 4 \quad (3.6)$$

donde Offset [s] es el control donde el usuario establece a partir de que segundo quiere empezar a leer el archivo.

Una vez realizado los cálculos anteriores, se verifica la configuración del FPGA que se usó para procesar los datos. Las configuraciones del FPGA disponibles serán tema en párrafos posteriores, cuando se describa el código que se encarga de administrar los recursos del mismo. Existen 18 configuraciones distintas; para saber cuál es la que se ha usado, se usa la información de configuración. Ya que se ha identificado la configuración del FPGA, se obtiene la cantidad de muestras que existen de cada canal analógico. Por el diseño del sistema, todos los canales deben tener la misma cantidad de muestras. Después de esto, se verifica si el número de canales a procesar es uno o cuatro. Sucedido lo anterior, inicia el ciclo denominado despliegue de datos. Éste se ejecutará 1 o cuatro veces, esto dependerá de si el usuario quiere generar un archivo de texto de la información (binaria) que está contenida en el archivo en procesamiento o sólo visualizar la información contenida en el mismo. La tarea que debe realizar el ciclo es, a grandes rasgos, la clasificación de la información. La información que se ha leído del archivo de configuración y del archivo que contiene las amplitudes está mezclada, esto quiere decir que es necesario separar la información y hacer que ésta misma quede asociada de manera correcta con el canal analógico correspondiente. Ya que la información ha sido clasificada, ésta debe ser vaciada en los indicadores correspondientes.

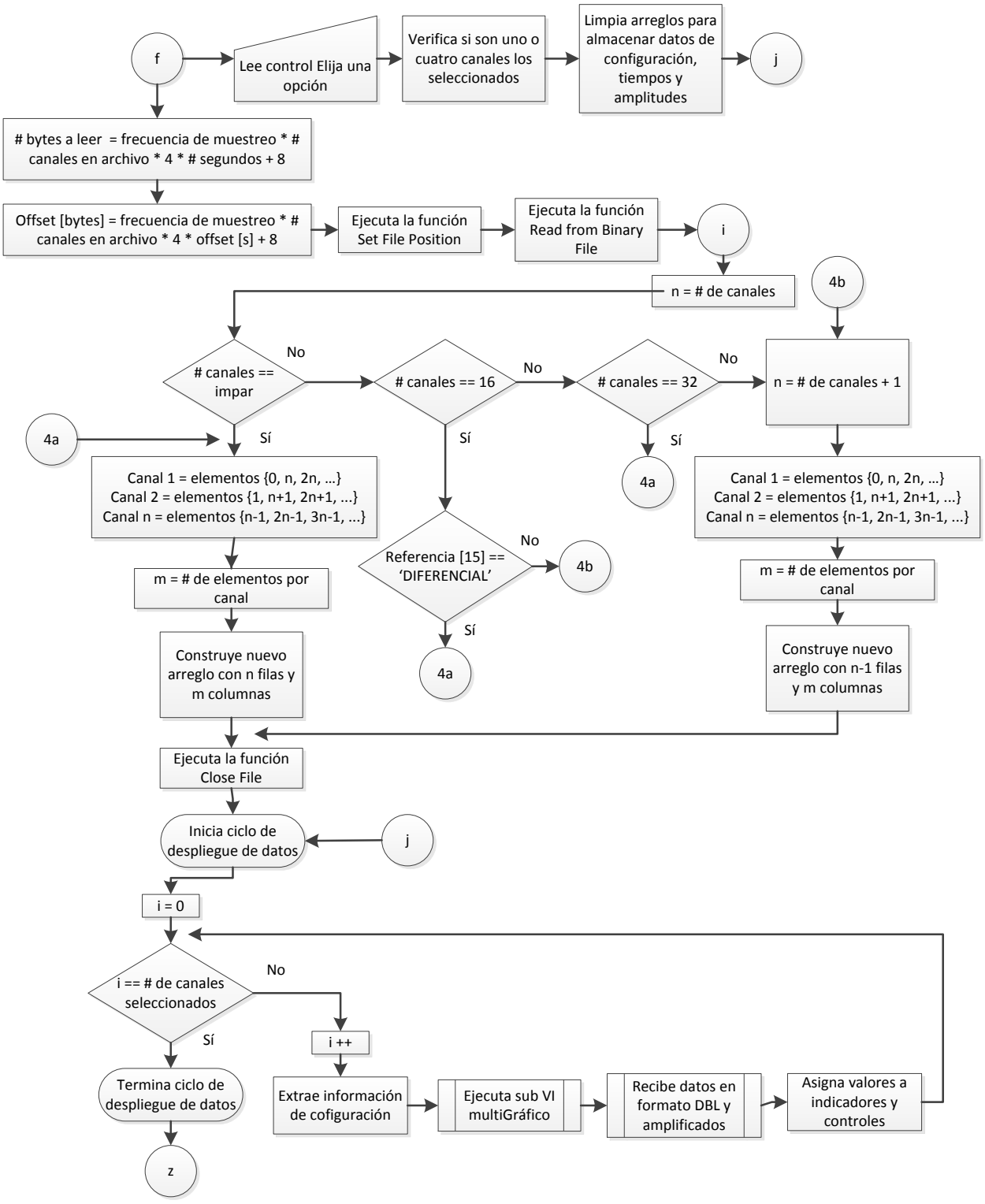


Figura 3.28. Cuarta parte del diagrama de flujo de la ventana Procesamiento.

Una vez descrito el diagrama de flujo de la figura 3.28, se da paso a explicar la operación del sub VI *multiGráfico,* figura 3.29. Éste recibe tres entradas y genera dos

salidas. Las entradas le permiten al sub VI obtener un arreglo de datos, saber si el usuario ha elegido aplicar las constantes de amplificación y en dado caso de que esto último sea afirmativo, saber cuál es el valor de dichas constantes. El arreglo de datos contiene la información de las amplitudes relacionadas a las señales digitalizadas a través del NI 9205. A través de sus salidas, el sub VI entrega al ciclo *while* un nuevo arreglo de datos en formato *double precision float* (DBL, por las siglas que maneja LabVIEW) y amplificados, esto último en dado caso de haber sido solicitado por el usuario.

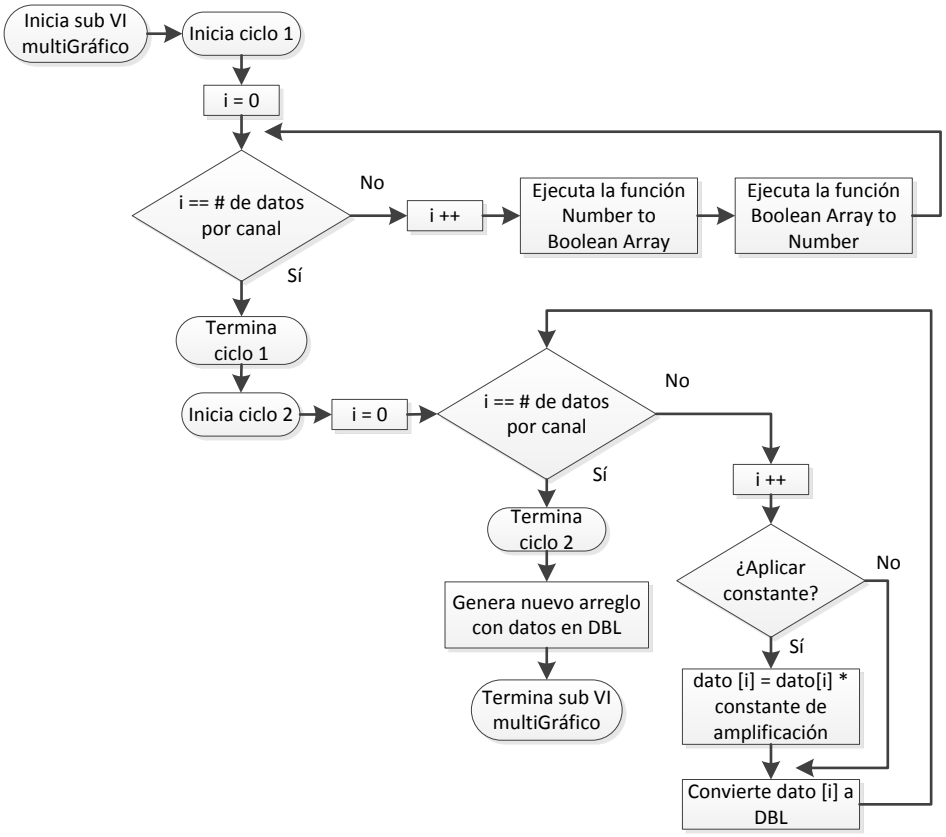


Figura 3.29. Diagrama de flujo del sub VI multiGráfico.

En la figura 3.29 es de resaltar dos funciones esenciales para el accionar de este sub VI. En primer lugar tenemos una función que es capaz de realizar la conversión de un número entero, o en representación punto fijo, a un arreglo *booleano* con 8, 16, 32 o 64 elementos, esta función es la que recibe en una primera estancia el arreglo de datos con las amplitudes. En segundo lugar hay una función que toma el arreglo *booleano* de la función descrita previamente y lo convierte a un número en representación punto fijo. Esta última función necesita ser configurada, se le indica que se va a requerir una longitud de palabra de 26 bits, de los cuales se usará 1 bit para signo, 4 bits para la parte entera y 21 bits para la parte fraccionaria. La configuración está fundamentada

por el hecho de que el convertidor del NI 9205 entrega valores que van del -10.3 al +10.3 (se está considerando la tolerancia del convertidor) y una resolución en el mejor de los casos de 6.57 µV (ver tabla 3.3).

Descrito el sub VI anterior, se da paso a la descripción de la quinta y última parte del diagrama de flujo de la ventana Procesamiento, figura 3.30. En la cuarta parte del diagrama de flujo, figura 3.28, se ve el conector z, éste es el enlace con la quinta parte.

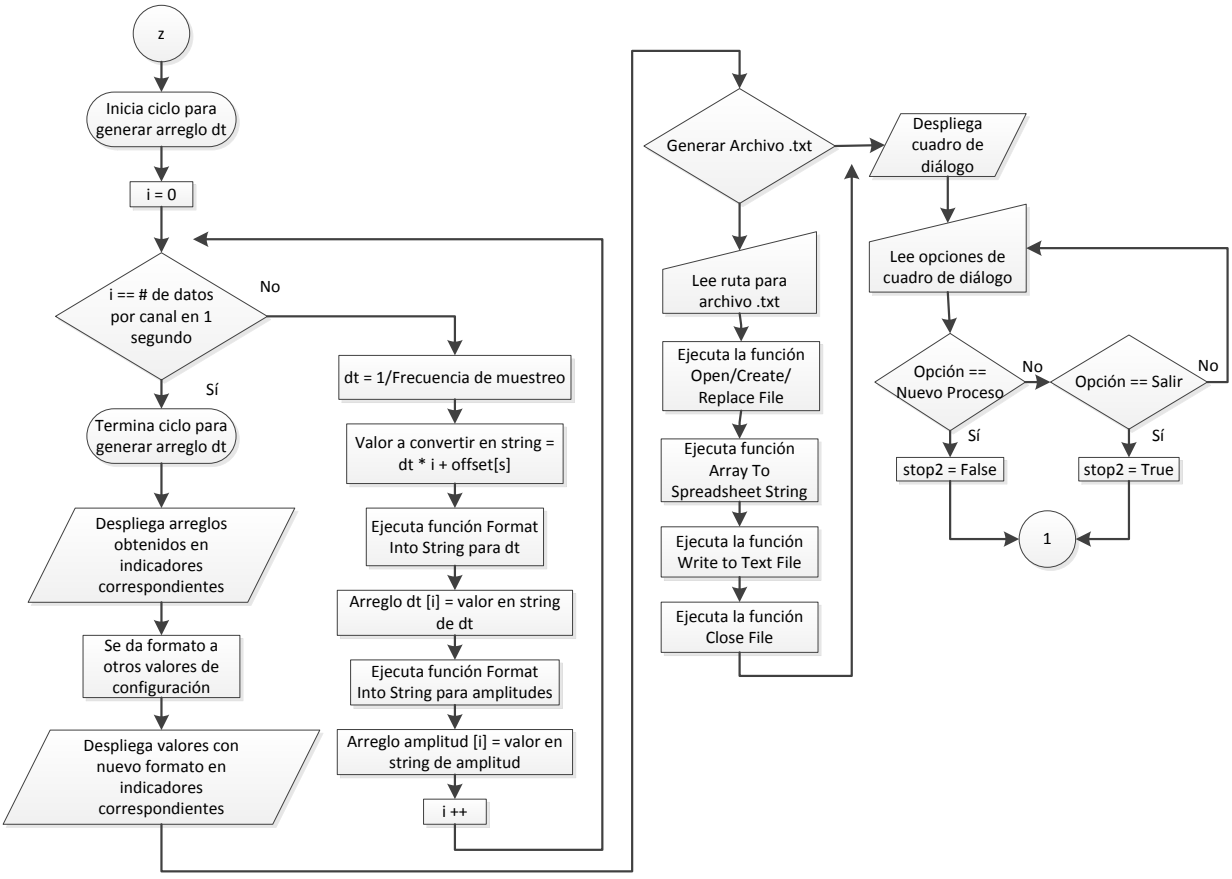


Figura 3.30. Quinta parte del diagrama de flujo de la ventana Procesamiento.

La primera tarea a realizar en esta parte del diagrama de flujo, es generar un arreglo con todos los dt que se obtienen a partir del dato de la frecuencia de muestreo, de tal forma que se pueda asociar a cada muestra un indicador de tiempo relativo. Una vez hecho lo anterior, podremos saber el momento justo en el que cada muestra fue tomada. Debido a que es necesario preparar la memoria para el tipo de dato que se va a manejar, y además tratando de hacer uso eficiente de la memoria, se hace uso de la función *Format Into String.* Esta función, además de permitir lo dicho anteriormente, es la encargada de convertir un dato numérico (entero, decimal, etc.) a texto codificado en ASCII, lo cual es de suma importancia, ya que de ser necesario, permite dejar casi lista la información para ser almacenada en un archivo de texto.

Debido a que la máxima frecuencia de muestreo que se puede alcanzar en el sistema es de 125,000 Hz, y que la máxima cantidad de segundos que puede contener un archivo es de 600 segundos, se necesita tres enteros y seis decimales (%3.6f) para poder representar todos los dt que puedan llegar a existir en este sistema. Estos datos deben ser proporcionados a la función *Format Into String*.

Ya que se tiene el tiempo en formato texto, entonces, la siguiente tarea a realizar debe ser la transformación de las amplitudes a un formato de texto, para ello se lleva a cabo un proceso muy similar al realizado para transformar los datos del tiempo, la única diferencia es que para las amplitudes se necesitan dos enteros y nueve decimales (%2.9f). Una vez concluido este proceso, se verifica si el usuario ha elegido Generar un archivo de texto, de no ser así, se vacía toda la información que hasta el momento se ha convertido a formato texto en un indicador y que le hemos llamado *Tabla de configuración*. Si el usuario ha pedido Generar un archivo de texto, entonces, además de todas las tareas anteriores, se le da el formato de texto a toda la información de configuración que contendrá el nuevo archivo. Después, tal y como se alcanza a ver en la figura 3.30, toda la información se agrupa en un arreglo y se manda a la función *Array to SpreadSheet String*. En paralelo al proceso anterior, se abre una referencia al archivo en donde se quiere depositar la información, ya que se ha abierto la referencia y la función *Array to SpreadSheet String* ha organizado la información, se manda a escribir al archivo toda la información con la función *Write to Text File* y posteriormente se cierra la referencia al archivo.

Hasta aquí la descripción correspondiente a la ventana Procesamiento, daremos paso a la descripción de la ventana FTP.

Ventana FTP

Es la segunda opción que ofrece la ventana Menú. En conjunto con la ventana *Procesamiento*, la ventana *FTP* conforma una herramienta poderosa que cumple con los requerimientos que se establecieron para el software de la PC.

Requerimiento

Se quiere contar con una interfaz que permita realizar la transferencia de los archivos, que se encuentran alojados en el SSD y en la memoria USB del cRIO, hacia la PC. La transferencia se busca que esté basada en FTP.

Con la finalidad de que la interfaz sea sencilla, se asignan por default algunos parámetros de configuración relacionados a la transferencia FTP, esto permite reducir la cantidad de datos que debe proporcionar el usuario para poder llevar a cabo la comunicación.

El usuario puede decidir si quiere visualizar el contenido del dispositivo de almacenamiento o si quiere transferir un archivo. Una vez elegida la acción a realizar, deberá introducir la dirección IP del cRIO y la ruta de la ubicación del archivo o del directorio. Ya que se ha realizado lo anterior, el usuario mediante un control indica que está listo para llevar a cabo la tarea.

Si el usuario ha indicado que quiere conocer el contenido del dispositivo de almacenamiento, éste contenido se despliega a través de un indicador, el cual además de los nombres de archivo o carpeta, entrega una descripción de su fecha de creación y su tamaño en bytes. Para saber si la comunicación se ha establecido de manera correcta, se tiene un indicador que a través de mensajes de texto despliega el estado de la comunicación, con ello se puede saber si la comunicación fue realizada con éxito.

Una vez realizada una actividad se pregunta al usuario si quiere realizar otro proceso o abandonar la ventana. En caso de que el usuario no haya realizado ninguna actividad y quiera interrumpir la ejecución de la ventana, existe un control que permite interrumpir la ejecución y abandonar la ventana.

Descripción de la operación

A través del diagrama de flujo presentado en dos partes, figura 3.31 y figura 3.32, podemos apreciar el proceso de operación de la ventana *FTP*. En la figura 3.31 se puede ver que esta ventana empieza configurando los controles e indicadores con los que va a trabajar el usuario, esto con la finalidad de evitar la mayor cantidad de errores posibles y que la interfaz sea intuitiva. Posterior a ello, mientras el usuario no seleccione una de las dos acciones que se pueden llevar a cabo, se pregunta constantemente sobre la acción que dese realizar, una vez que el usuario ha elegido la actividad a realizar, se espera una confirmación de que el control *Empezar* ha sido seleccionado.

Si el usuario selecciona que quiere ver el contenido de la memoria USB o la SSD del cRIO, se lee la dirección IP establecida. Luego, a través de una serie de VIs proporcionados por el módulo de comunicación FTP de LabVIEW, se busca establecer una comunicación FTP entre la PC y el cRIO. Al terminar la ejecución de cada VI involucrado en este proceso, se verifica el funcionamiento de éstos para determinar si se ha generado algún tipo de error que impida la comunicación. Realizada la comunicación de manera correcta, se lee la ruta que el usuario quiere revisar. Sino hay algún error, la aplicación concluye la tarea solicitada, despliega la información que existe en la ruta especificada y manda un mensaje para conocer si todo ha marchado de manera correcta. Por el otro lado, si lo que el usuario quiere hacer es transferir un archivo desde el cRIO hasta la PC, debe proporcionar la misma información del caso anterior, ésta es la dirección IP del cRIO y la ruta del archivo, pero para este caso, en el

control *Ruta remota* además de establecer la ubicación del archivo, debe poner el nombre y la extensión del mismo.

Por default se genera una carpeta llamada *dataADC* y que se encuentra en la unidad C de la PC. Esta carpeta se propone como la carpeta por default para almacenar los archivos que se transfieran, el usuario puede optar por otra carpeta haciendo uso de un cuadro de diálogo, que además de permitir seleccionar la ubicación del archivo en transferencia, permite darle un nombre al mismo.

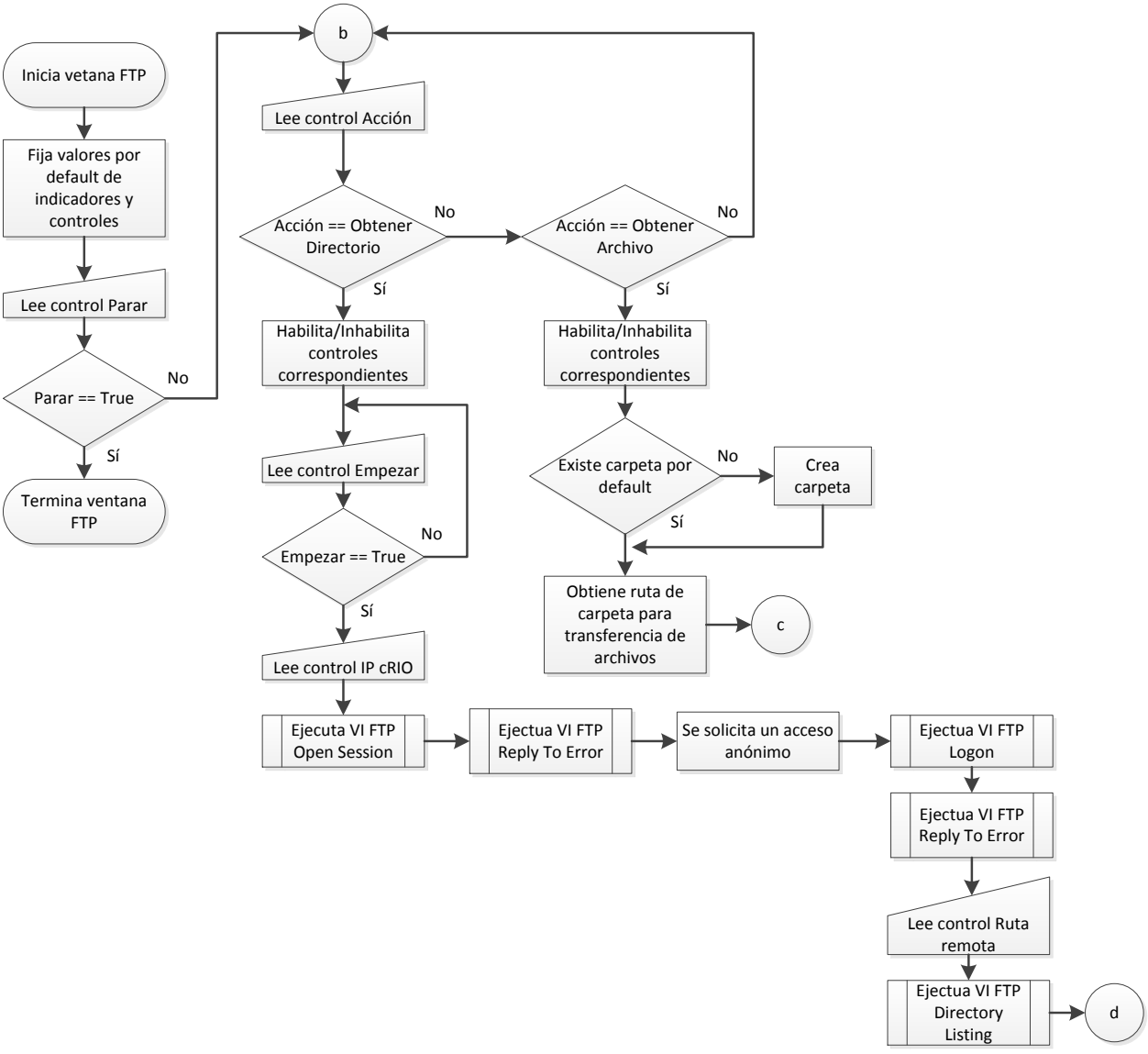


Figura 3.31. Primera parte del diagrama de flujo de la ventana FTP.

En la figura 3.32 se muestra que para comenzar la transferencia de un archivo, igual que en el caso anterior, la aplicación espera a que el usuario seleccione el control *Empezar*. Una vez seleccionado este control, se solicita un acceso anónimo (sin

password ni nombre de usuario); posteriormente, se solicita una transferencia en modo binaria y pasiva. El modo binario se debe a que este modo permite la transferencia de archivos con contenido arbitrario, en otras palabras, archivos que pueden contener texto, imágenes o una combinación de éstos. El modo pasivo se usa debido a que permite evitar problemas con el *firewall* de la PC, lo anterior debido a que todas las peticiones de conexión se hacen desde la PC (cliente), con esto el *firewall* no tiene que filtrar ninguna conexión que pudiera iniciar el cRIO (servidor).

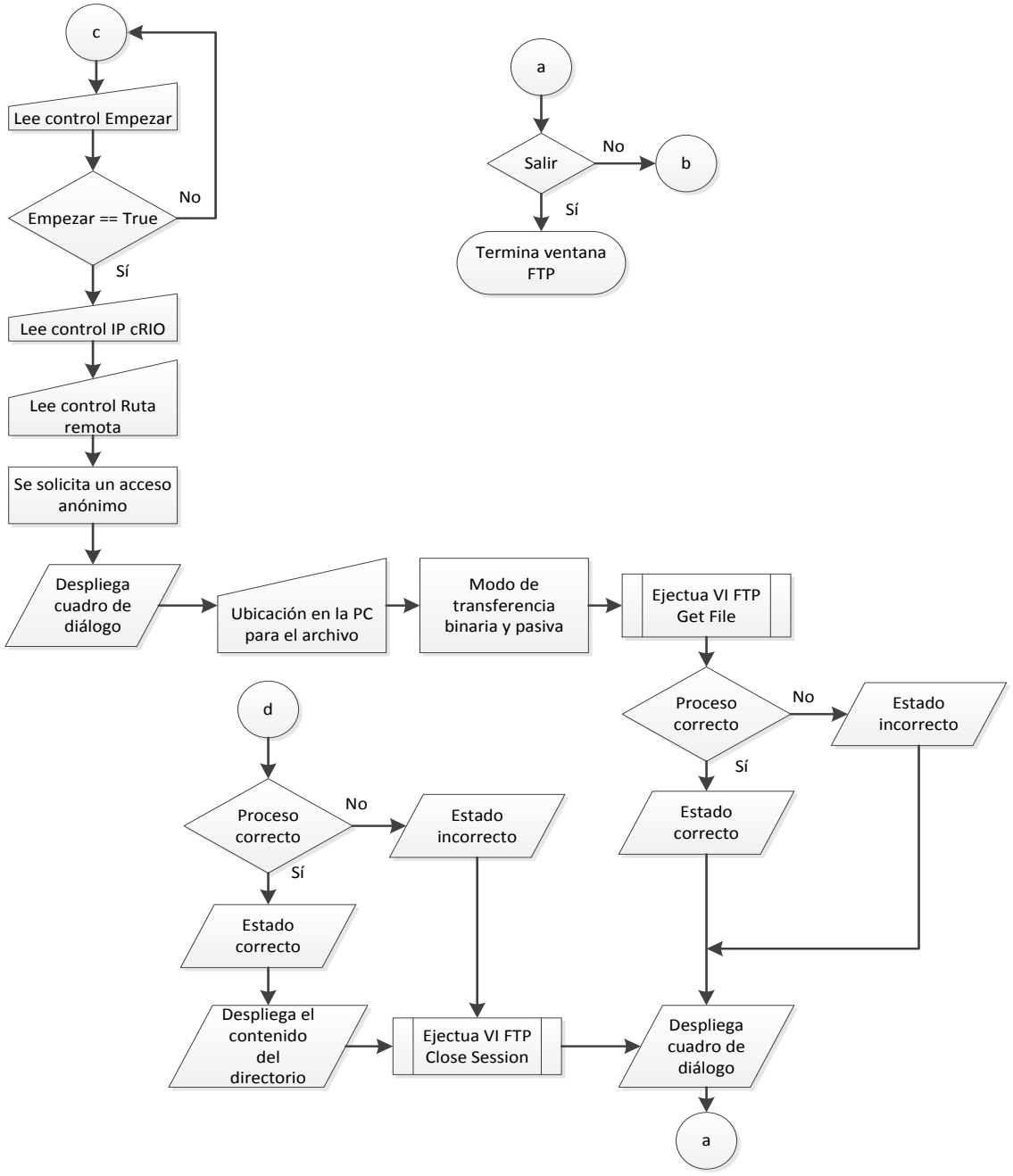


Figura 3.32. Segunda parte del diagrama de flujo de la ventana FTP.

3.3.2. Software del MCU del cRIO 9012

Hasta el momento se ha explicado el diseño del software correspondiente a la PC, el siguiente bloque de software corresponde al que se va a ejecutar en el MCU de tiempo real del cRIO 9012, entonces, daremos paso a la descripción del software del MCU mencionado.

Requerimiento

Se quiere una interfaz que facilite la configuración y la puesta en marcha de una adquisición de variables analógicas. Además de lo anterior, se quiere que la interfaz pueda ser utilizada sin restricción del sistema operativo.

Para lograr que la interfaz pueda ser utilizada en cualquier sistema operativo, se utilizarán páginas web, éstas serán desarrolladas con la ayuda de la herramienta *Web Plublishing Tool*, la cual pertenece al entorno de desarrollo de LabVIEW. A grandes rasgos, lo que debe hacer la herramienta mencionada es traducir el lenguaje gráfico de LabVIEW a código HTML (Hyper Text Markup Language), este último es la herramienta más usada para el desarrollo de páginas web y puede ser ejecutado en cualquier navegador comercial actual. Las páginas web contarán con varias características que permitan su facilidad de uso y al igual que el código fuente del sistema, estarán almacenadas en el SSD.

La página web que permita el acceso a la interfaz llevará el nombre de *Principal*. Cualquier operación que se quiera realizar deberá ser emitida desde esta página, inclusive la acción de parar el sistema. La página *Principal* deberá estar disponible siempre, de tal forma que una vez que el cRIO esté energizado el usuario podrá acceder a ella.

Al comenzar la ejecución de la interfaz, el usuario debe poder establecer si lo que necesita es configurar los parámetros de una adquisición, ajustar la hora base del cRIO o iniciar una adquisición de variables analógicas, tomando la última configuración guardada en el USB (el SSD del cRIO guarda la dirección de la última configuración).

En el caso de que el usuario elija *Adquisición*, debe verificar que ha tomado esa decisión y que quiere proseguir con la misma. A continuación el usuario debe elegir el canal que desea visualizar. El usuario puede monitorear un canal a la vez, esto quiere decir, que al iniciar un proceso de adquisición, se estará desplegando la información que contenga 1 de los 32 canales que pueden ser configurados. Por cuestiones de la integridad de la información, el monitoreo del canal estará activo únicamente mientras no haya registro de información, en los lapsos en los que se esté almacenando información en la memoria USB, no habrá forma de ver lo que sucede con los canales

en procesamiento. Posterior a la elección del canal, el usuario debe confirmar que está listo para continuar.

La configuración con la que se va a llevar a cabo la adquisición será desplegada en la página web llamada *Principal*, de esa manera el usuario podrá conocer lo que contiene el archivo de configuración y estar seguro de que es el archivo que quiere usar. Una vez sucedido lo anterior, por default el navegador debe abrir una página nueva donde se indique el modo de adquisición que ha sido elegido por el usuario, el canal que se encuentra en monitoreo, la cantidad de muestras por segundo que se están adquiriendo, la cantidad de segundos adquiridos, si el módulo NI 9205 ha generado un error, el estado del *buffer* de comunicación entre el FPGA y el MCU, datos sobre el último archivo generado y un control de paro.

Dependiendo del modo de almacenamiento, la ventana que se despliegue al iniciar la adquisición tendrá algunas variantes. Por ejemplo, en el caso del modo disparo, aparecen dos indicadores que mostrarán el límite de voltaje (sobre umbral superior, debajo umbral inferior o en intervalo) establecido por el usuario para iniciar almacenamiento, para este mismo caso, un control que permita iniciar el almacenamiento pierde sentido, para el resto de los modos sí existe un control que permite iniciar el almacenamiento. Para el modo temporizado, una vez iniciado el almacenamiento, el control de paro se deshabilita debido a que ya se ha impuesto una condición de tiempo que debe detener el almacenamiento. Todos los modos, a excepción del modo continuo, cuentan con una gráfica en donde se despliega la señal que está siendo procesada a través del canal elegido por el usuario.

Toda la información adquirida de la conversión analógica digital, debe ser almacenada en la memoria USB. No hay posibilidad de que el sistema guarde la información si no se ha insertado una memoria USB. Si la memoria no se encuentra, el *buffer* de comunicación entre el FPGA y el MCU llegará a saturación y entonces la adquisición de información colapsará. Sin embargo, la aplicación debe ser capaz de permitir monitorear cualquier canal sin tener una memoria USB insertada.

Para poder detener un proceso de adquisición, es necesario indicarlo primero en la página web llamada *Principal* y posteriormente indicarlo en la ventana que esté activa según el modo de adquisición. No es posible detener un proceso haciendo uso exclusivamente del control de paro que tienen las ventanas de adquisición; aunque éste sea presionado, el sistema siempre debe preguntar si en la página web *Principal* ya se ha dado esa indicación. Lo anterior permite que los procesos de adquisición no sean detenidos al cerrar las páginas web, esto quiere decir que aunque se cierren la interfaz (navegador web), la aplicación sigue corriendo sin mayor problema), entonces, la aplicación debe tener un comportamiento *stand-alone*.

En el caso de que en la página web *Principal* el usuario elija que quiere realizar una *configuración* de adquisición, será necesario abrir una nueva ventana en donde éste pueda efectuar la configuración necesaria. Esta ventana debe poder identificarse con el nombre de *Configuración de Canales* y debe ir modificando su aspecto a medida que el usuario va estableciendo los parámetros de configuración. Inicialmente el usuario debe poder introducir un nombre de estación (caracteres alfanuméricos), la cantidad de canales que quiere adquirir y el tipo de medición que se va a realizar en cada canal (Diferencial, NRSE, RSE). El software debe limitar la cantidad de canales por adquirir, en relación al tipo de medición que se quiera realizar en cada canal.

La cantidad de canales en modo diferencial es lo primero que se debe elegir, éstos pueden ser separados hasta 8, o de lo contrario 16. Estas cantidades se deben a dos razones, las restricciones del mapeo de funciones en el FPGA y a que cada medición diferencial necesita dos canales. Si se elige al menos un canal en modo diferencial, no se podrá elegir ningún canal en NRSE o RSE.

La cantidad de canales en modo NRSE es lo que se elige inmediatamente después de la cantidad de diferenciales. Para que se puedan elegir canales NRSE no debe haber diferenciales, a manera de ejemplo se puede suponer que se han elegido 8 diferenciales, esto equivale a tener 16 canales ocupados y a pesar de que sobran 16 canales, no se podrá elegir ningún NRSE. Aunque los NRSE no se pueden combinar con los diferenciales, si se pueden procesar simultáneamente con los canales RSE.

La cantidad de canales en modo RSE es lo último en elegirse, el número de éstos estará limitado tanto por los NRSE escogidos como por el hecho de que no haya ningún diferencial, por ejemplo, si se eligen 16 NRSE, se podrá seleccionar 16 RSE.

No es posible seleccionar los canales arbitrariamente, esto quiere decir que la asignación se hace secuencial, por ejemplo, si el usuario selecciona un canal diferencial, los canales AI0 y AI8 del NI 9205 se reservan en automático, si se selecciona un canal NRSE y dos RSE, el sistema configura al AI0 como NRSE y a los canales AI1 y AI2 como RSE.

Una vez que se han asignado la cantidad de canales que se van a procesar en la adquisición, el usuario podrá establecer los parámetros de los mismos. Los parámetros que el usuario puede establecer son: la frecuencia de muestreo, el nombre del canal (cuenta con uno por default), unidades de las mediciones, la constante de amplificación y el rango. El modo de medición con el que trabaja cada canal también es un parámetro que ya ha sido asignado en la etapa anterior.

La frecuencia de muestreo tiene que ser un sub múltiplo de 40 MHz, la cual es la frecuencia de la señal de reloj con la que trabaja el FPGA. Adicional a esta restricción,

se tiene que la relación entre número de muestras que toma el convertidor y el número de canales no es lineal, eso quiere decir que si un solo canal se puede muestrear a 125 KHz, dos canales no se pueden muestrear a 125 KHz / 2. Ante este hecho y a manera de encontrar una relación entre el número de muestras y el número de canales, se realizaron una gran cantidad de pruebas en laboratorio. Los resultados de estas pruebas son presentados de manera gráfica en la figura 3.33.

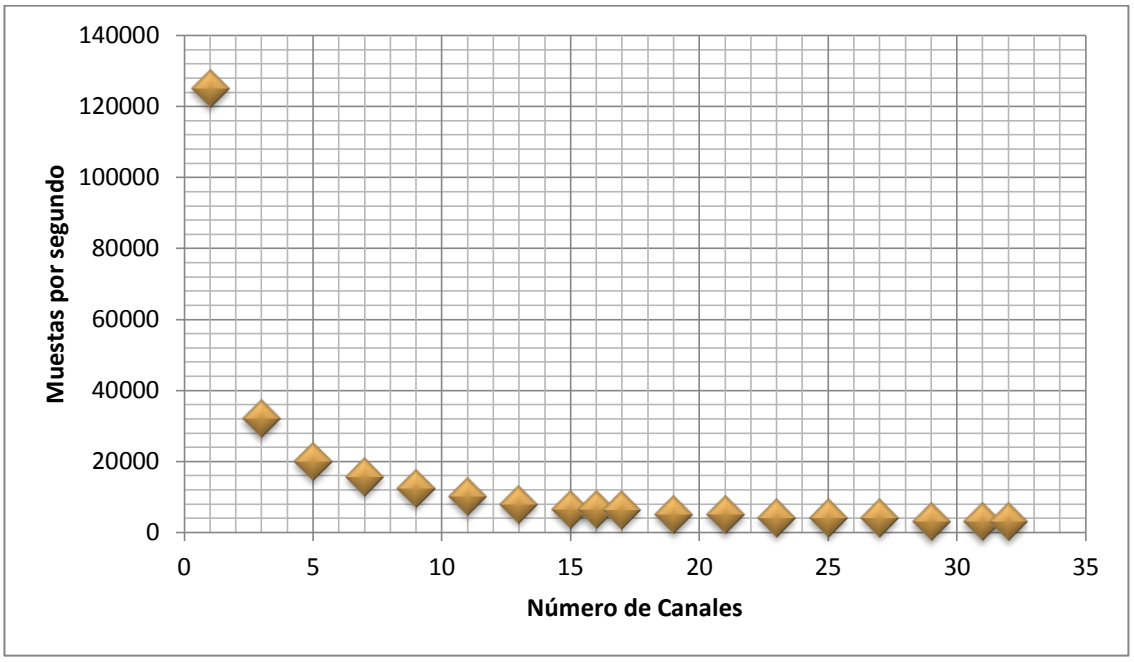


Figura 3.33. Número de canales vs Número de muestras por segundo.

La función presentada en la figura 3.33 asemeja una función exponencial, este comportamiento se puede atribuir al hecho de que las funciones que se usan en el FPGA, en especial las que se usan para leer los datos, tienen comportamientos no lineales. No obstante, también puede haber comportamientos no lineales en el hardware del NI 9205.

El nombre del canal puede contener cualquier carácter alfanumérico. La constante de amplificación debe ser entera y puede tener un valor máximo de 100. En el rango se pueden establecer cuatro opciones: +/- 200 mV, +/- 1 V, +/- 5 V, +/- 10 V, donde el valor por default es de +/- 10 V. Como ya se había mencionado, el modo de medición en el que pueden estar configurados los canales son tres.

Cada vez que el usuario haya terminado de configurar un canal, debe indicar mediante un control que desea guardar la configuración de ese canal, en automático el sistema le permitirá configurar el canal que sigue. A manera de referencia siempre existirá un indicador en el cual se despliegue la cantidad de canales que el usuario debe configurar.

Establecidos los parámetros de configuración para los canales, se requiere que el usuario pueda configurar el modo de almacenamiento. El usuario puede elegir entre cuatro modos distintos de almacenamiento: *continuo, disparo, software y temporizado*.

El modo de almacenamiento *continuo* debe ser escogido cuando el objetivo es almacenar constantemente los datos resultados de la prueba, hasta que el espacio libre del dispositivo se agote. En este modo, se puede establecer el intervalo de tiempo que durará cada archivo, este tiempo puede ir de 1 hasta 600 segundos. Entre el cierre de un archivo y la apertura de uno nuevo, puede pasar entre uno y dos segundos.

El modo de almacenamiento *disparo* permite almacenar la información únicamente cuando la señal en monitoreo cumpla una condición establecida por el usuario, mientras la condición no se cumpla, el sistema permite ver lo que sucede en uno de los 32 canales sin almacenar información. Los canales que pueden desencadenar el proceso de almacenamiento (los que pueden realizar el disparo), son únicamente los canales AI0, AI1 y AI2 (figura 3.9), si los tres están habilitados, con que uno cumpla la condición de disparo, es más que suficiente para iniciar el almacenamiento. Es importante hacer notar que el modo de medición en el que fueron configurados los tres canales, no interfiere con su funcionamiento en modo disparo.

El modo de almacenamiento disparo permite tres configuraciones. El primer modo se llama *sobre umbral superior,* cuando el sistema opera en este modo, la señal debe encontrarse por encima de un nivel de voltaje especificado para que se inicie el registro de datos, es decir, si el voltaje seleccionado como *umbral superior* es 5 y la señal se encuentra en 5.1 Volts, el sistema debe empezar a almacenar la información. Bajo la misma configuración, si el valor de la señal es de 4.9 Volts el sistema no debe almacenar, en lugar de eso debe mostrar la señal que está siendo muestreada en el canal seleccionado para visualización. El segundo modo se llama *debajo umbral inferior*, si el sistema se encuentra operando en este modo, se debe iniciar el almacenamiento cuando la señal se encuentre por debajo de un nivel de voltaje especificado, tomando el ejemplo anterior, si el voltaje como *umbral inferior* es 5 y la señal se encuentra en 5.1 Volts, el sistema no debe almacenar información, sin embargo, cuando el valor de la señal es de 4.9 V el sistema debe iniciar el registro de datos. El tercer modo de operación se llama *en intervalo,* como su nombre lo indica, para que el registro de datos se inicie, la señales deben estar dentro de un intervalo definido a través de un *umbral superior* y un *umbral inferior*. Si por ejemplo una señal llega a tener un valor de 4.3 Volts, y adicionalmente se han fijado los valores de 4 Volts para el umbral inferior y 4.5 Volts para el umbral superior, entonces, el sistema debe iniciar el registro de datos.

Independientemente de las condiciones de umbral definidas, el sistema tiene una protección de tiempo para evitar que un archivo pueda ser erróneamente creado, esta

protección es una cantidad de segundos fijada por el usuario, la cual puede ir de 1 hasta 600. En caso de que ocurra una condición de disparo y ésta se mantenga hasta el punto de superar el tiempo de protección, se forzará el cierre del archivo en proceso de llenado y se abrirá uno nuevo.

El modo de almacenamiento *software* ofrece la posibilidad de que el usuario inicie y pare la adquisición en cualquier instante. La operación de este modo depende de un control de inicio y otro de paro. No se debe perder de vista que si en dado caso el usuario dejará pasar 600 segundos de adquisición, aunque no se genere la acción de paro, el sistema automáticamente parará la adquisición y prepara al sistema para iniciar una vez más.

El último modo de almacenamiento es el *temporizado,* este modo otorga la opción de fijar un número exacto de tiempo durante el cual se desea realizar un registro. Este tiempo debe ser fijado en segundos y puede recibir los valores de 1 hasta 600.

Con base en pruebas de laboratorio, se encontró que la velocidad para escribir en la memoria USB se vería afectada por el espacio libre con el que contaba la misma. Cuando el dispositivo se encontraba al 80% de su capacidad, no se podía escribir a la memoria con la misma velocidad que se alcanzaba cuando existía mayor cantidad de espacio libre. Lo anterior suena lógico si se hace una analogía entre el mecanismo de escritura y una bodega. Cuando en una bodega hay pocas cajas pero todas diferentes entre ellas, es sencillo localizar a cualquiera de ellas en un tiempo relativamente corto, pero si el número de cajas empieza a incrementar y se sigue usando el mismo mecanismo de búsqueda, el tiempo para localizar una caja se incrementa considerablemente, hasta un punto en el que el tiempo de búsqueda no puede predecirse. En el caso de la presente tesis, el tiempo de búsqueda y escritura se ve limitado por tres factores: el MCU, las funciones de escritura y el sistema de archivos.

Por lo descrito en el párrafo anterior, el software cuenta con un algoritmo que verifica que la capacidad de la memoria USB se encuentra por debajo del 80%, en dado caso que se exceda ese porcentaje el sistema detendrá la adquisición. Además de ello, en la ventana principal de la interfaz web, se debe desplegar la cantidad de archivos que se pueden crear bajo una configuración definida.

Otro factor a considerar y que ha parecido en párrafos anteriores, es el del límite de tiempo para un archivo. Para poder asegurar el correcto almacenamiento de los datos, 600 segundos es la cantidad de tiempo que puede contener como máximo un archivo. En diversas pruebas de laboratorio se pudo observar que entre más grandes eran los archivos, mayor probabilidad había de que después de un cierto número de archivos creados, hubiera uno que no se generara de manera correcta. Por todo esto y

para facilidad del usuario, se ha establecido 600 segundos como el tiempo máximo que puede contener un archivo de adquisición.

Una vez que se ha elegido el modo de almacenamiento y que se han establecido los parámetros que requiere el modo elegido, el usuario debe elegir el formato de tiempo que desea manejar. Son tres formatos que pueden ser elegidos: Hora UTC, Hora Local y Hora Local Verano. La Hora UTC está referida a la hora del meridiano de Greenwich, la Hora Local le resta seis horas a la Hora UTC y la Hora Local Verano le resta cinco horas a la Hora UTC.

Ya que se ha realizado todo lo anterior, el usuario debe indicar que desea guardar la configuración. Si la configuración se almacenó en la USB de manera exitosa, entonces, la ventana desplegará un mensaje que indique que todo marcha bien, de lo contrario el mensaje deberá indicar que hubo un error en el almacenamiento. Una vez que el usuario haya leído el mensaje, debe indicarle al sistema que lo ha hecho a través de un control. Presionado el control, la ventana deberá cerrarse, siendo la página web principal la única que esté en activo.

En el caso de que en la página web principal el usuario elija que desea *ajustar fecha y hora*, se abrirá una nueva ventana que permitirá verificar la fecha y hora con la que está trabajando el sistema, además de la realización de ajustes a estos parámetros. La hora debe ser establecida en UTC y en un formato de 24 horas. Para facilidad del usuario cada componente de la hora y fecha tendrán su propio control. El usuario debe indicar que quiere modificar la hora o la fecha mediante un control. En caso de que el usuario quiera abandonar la ventana actual, podrá indicarlo a través de otro control. Una vez realiza la solicitud de paro, la ventana se cierra y lo único activo es la página web *Principal.*

Descripción de la operación

Principal

Tal y como se aprecia en la figura 3.34, al cargarse la página web *Principal* se modifican los valores de los indicadores y controles para empezar de manera adecuada un nueva tarea. Se espera a que se escoja una acción a realizar, según ésta se puede iniciar una configuración de adquisición, una configuración de hora y fecha o empezar con una adquisición de datos.

Cuando se elige realizar una adquisición, nodo a, se debe identificar el último archivo de configuración almacenado en la memoria USB, la ruta de este archivo se encuentra en el SSD. Ahora que se tiene la ruta del último archivo de configuración, se manda esta información a un sub VI que se encarga de leer los datos de configuración. Antes de depositar esta información en los controles e indicadores correspondientes, se

procesa los datos referentes al rango y modo de medición. Se tiene el valor de la frecuencia de muestreo con la que se quiere procesar; debido a que el que hará la petición será el FPGA se tiene que realizar una transformación de frecuencia a *Ticks.* Paralelamente a esta conversión se obtienen la cantidad de archivos que pueden ser generados, para esto se considera el espacio libre del dispositivo USB y la máxima cantidad de bytes que puede llegar a tener un archivo. Se prosigue a realizar los cálculos necesarios para configurar el *buffer* DMA y a refrescar los indicadores y controles que reflejen que todo el proceso anterior se ha llevado a cabo.

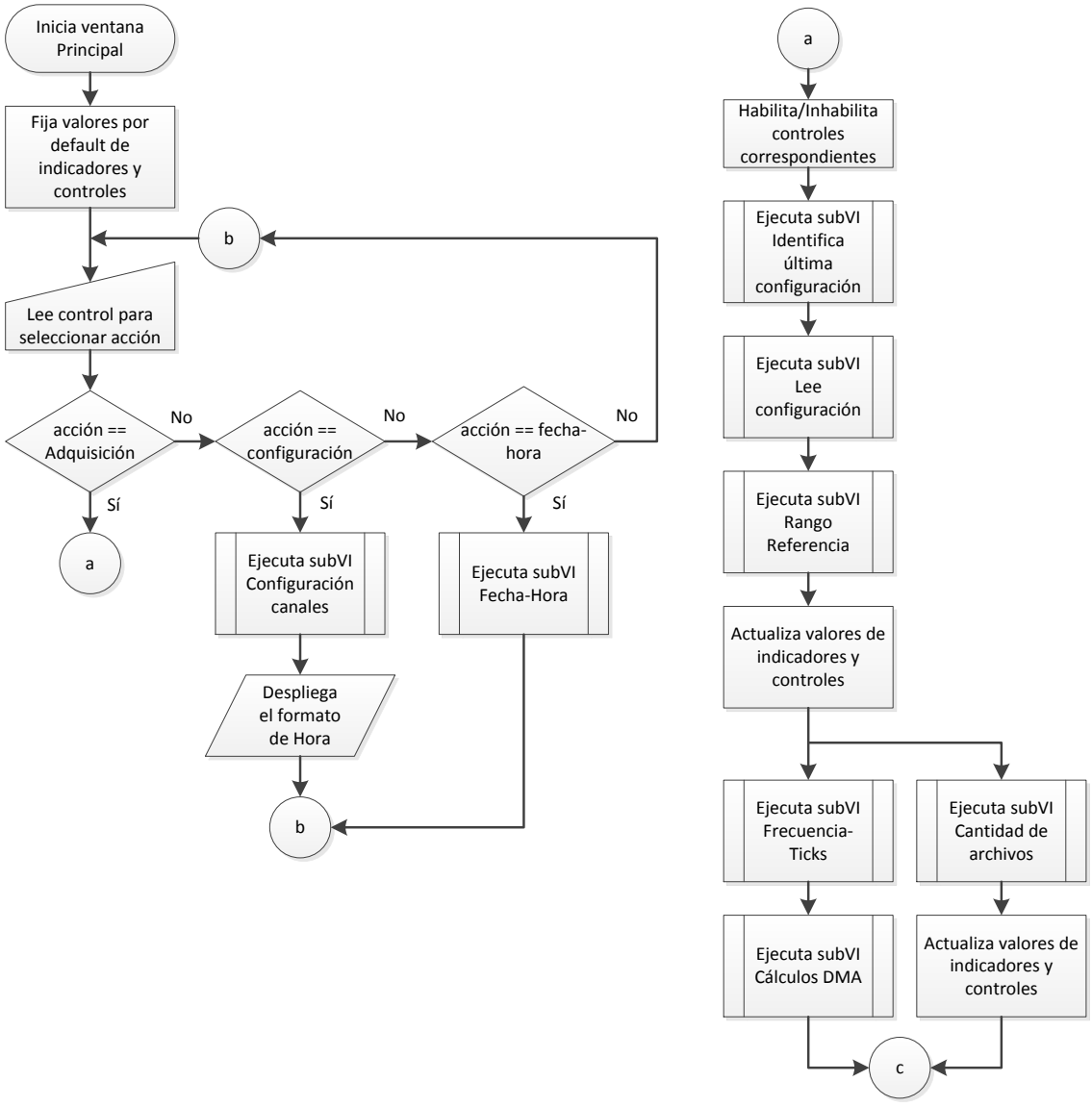


Figura 3.34. Primera parte del diagrama de flujo de la página web Principal.

Se ha explicado a grandes rasgos el diagrama de flujo de la figura 3.34, ahora se dará paso a explicar todos los sub VIs contenidos en la figura.

Primera parte del sub VI Configuración canales

El sub VI *Configuración canales* es descrito mediante el diagrama de flujo que se muestra en la figura 3.35. A grandes rasgos, este algoritmo lleva paso a paso al usuario para que pueda configurar los canales analógicos de manera adecuada. Primero se fijan los valores por default de todos los controles e indicadores, posteriormente se restringen la cantidad de canales diferenciales y de terminación simple que pueden ser seleccionados. Una vez realizado lo anterior, el usuario puede empezar a seleccionar la cantidad de canales diferenciales que desea usar, si se solicita al menos uno en este modo, el algoritmo inhabilita la selección de canales en modo RSE y NRSE. Si no se escogió ningún canal diferencial, entonces, se pueden elegir 32 canales en modo NRSE. Según la cantidad de NRSE elegidos, será la cantidad de canales que puedan ser elegidos en modo RSE. Cabe aclarar que los canales en modo RSE se pueden combinar con los NRSE. Para pasar a la siguiente sección de configuración, se lee el nombre de directorio que el usuario haya establecido, este directorio será el que va a contener los archivos tanto de configuración como de adquisición, más adelante se verá que a este nombre se le agrega texto que está relacionado con la hora y fecha de creación.

Ya en la segunda sección de configuración, nodo CCa, se hace una modificación de los indicadores y controles de la ventana, inmediatamente después se verifica la cantidad de canales que han sido elegidos. Este dato entra como parámetro al sub VI *Cálculo de frecuencias.* Una vez ejecutado este sub VI, se despliega en la ventana actual el arreglo de frecuencias permitidas. Posteriormente se espera a que el usuario elija una frecuencia, ésta será la frecuencia de muestreo para todos los canales habilitados. Inicia un nuevo ciclo, while1*,* el cual será ejecutado hasta alcanzar a configurar todos los canales elegidos para adquirir. Los conectores CCc, CCd y CCe enlazan a diagramas de flujo que son tema de párrafos posteriores. Por lo tanto se da por terminada la descripción de la figura 3.35, dando paso a la descripción del sub VI *Cálculo de frecuencias*.

Sub VI Cálculo de frecuencias

La tarea del sub VI *Cálculo de frecuencias,* figura 3.36*,* se lleva a cabo a través de un ciclo *while*, un contador y comparaciones sucesivas. Este sub VI permite generar un arreglo que contiene todas las frecuencias de muestreo que pueden ser usadas para la adquisición de datos.

Segunda parte del sub VI Configuración de canales

Cuando se acabó la descripción de la figura 3.35, se mencionó al conector CCc. Este conector permite remitirnos al diagrama de la figura 3.37. Este diagrama

corresponde al código que tiene que ser ejecutado dentro del ciclo *while1.* En cada iteración de este ciclo el sistema indica el modo de medición seleccionado, después indica el número de canales a configurar, despliega los valores por default de indicadores y controles que sirven de guía al usuario, muestra la frecuencia de muestreo establecida para todos los canales, lee el nombre del canal, la constante de amplificación, las unidades y el rango. Al final de cada iteración todos los datos introducidos por el usuario se agrupan en un *cluster*. Éste último se pasa a un arreglo de 2 dimensiones y para finalizar el ciclo se espera que el usuario seleccione el control *Guardar.*

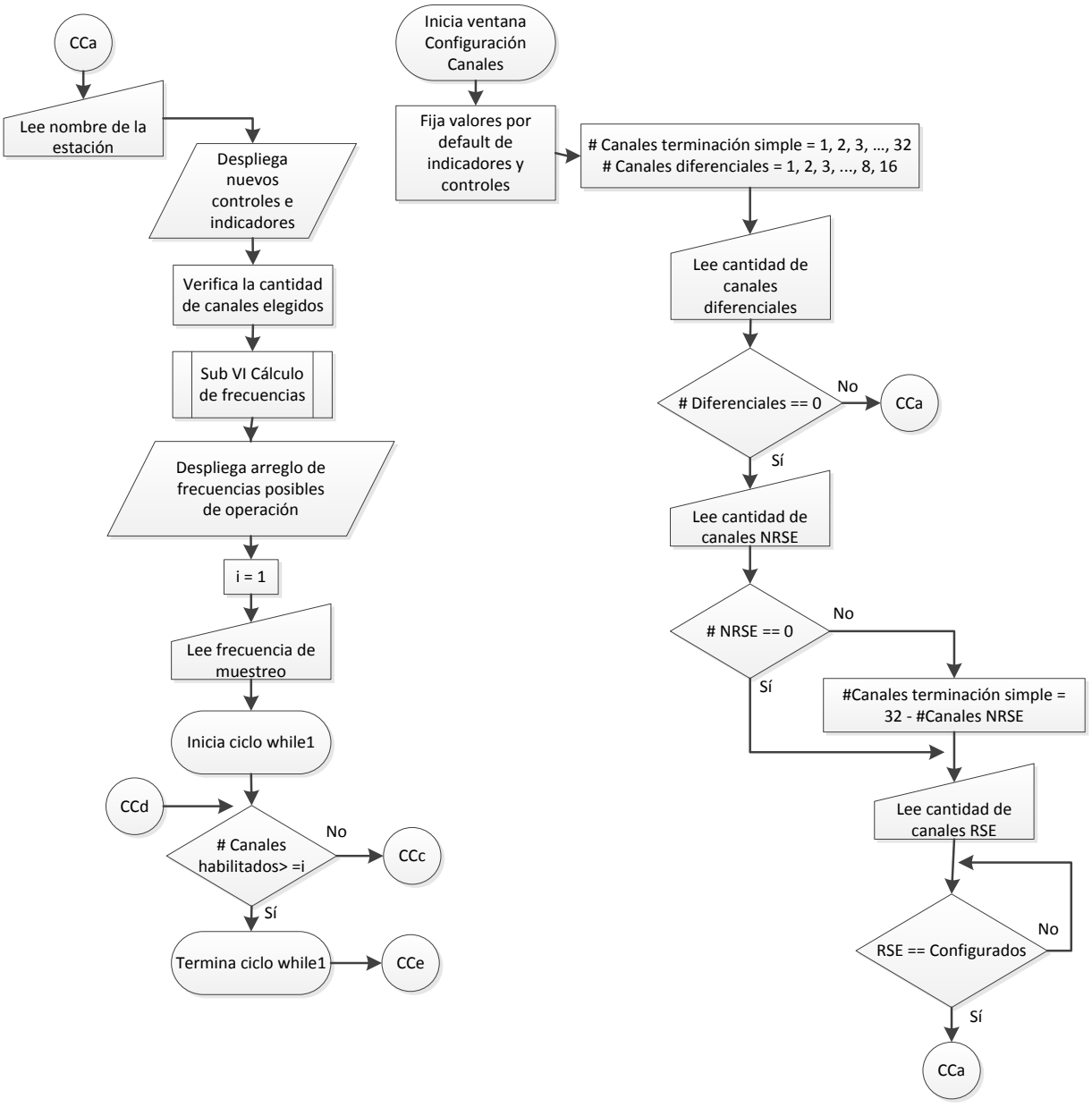


Figura 3.35. Primera parte del diagrama de flujo del sub VI *Configuración de canales*.

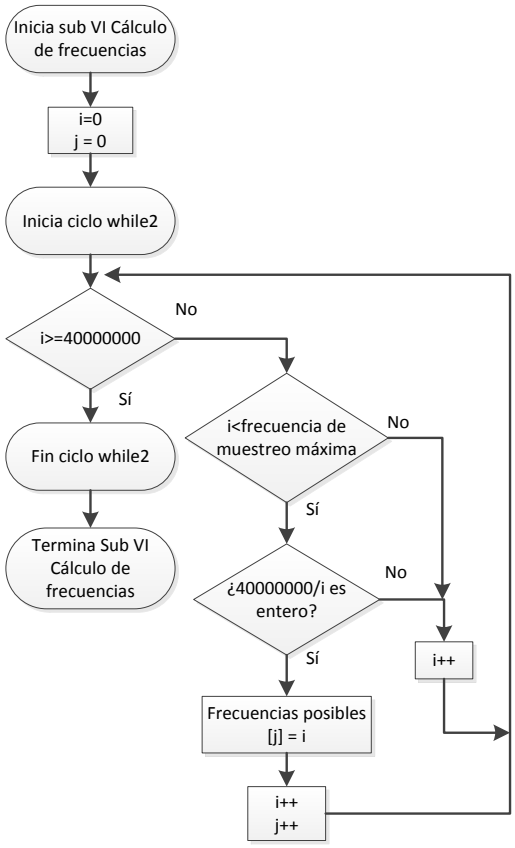


Figura 3.36. Diagrama de flujo del sub VI *Cálculo de frecuencias*.

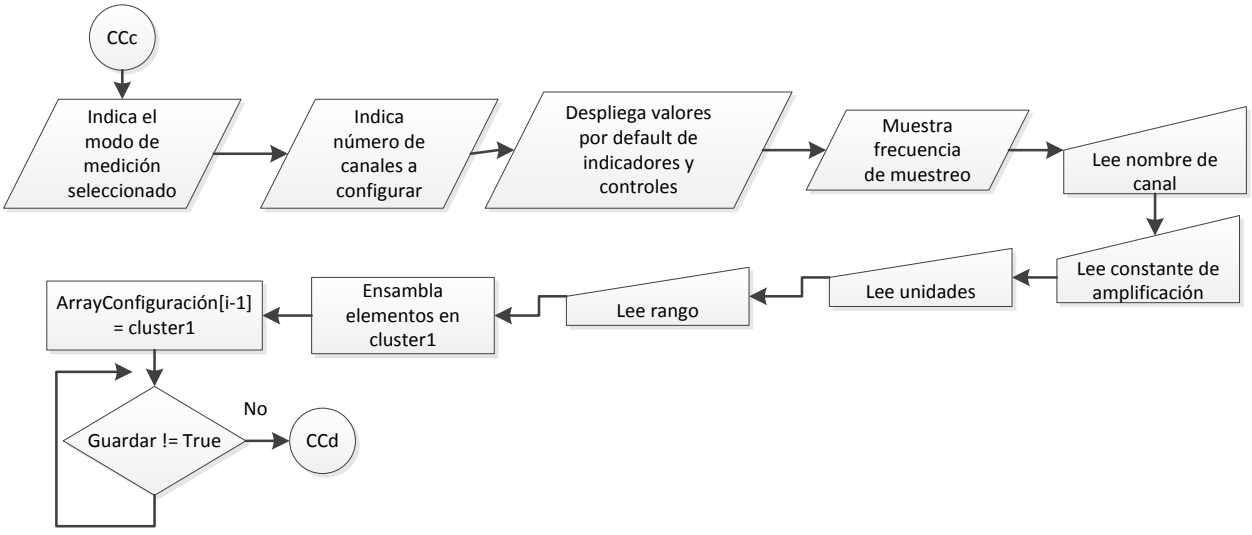


Figura 3.37. Segunda parte del diagrama de flujo del sub VI *Configuración de canales*.

Tercera parte del diagrama de flujo del sub VI Configuración de canales

Al finalizar la descripción de la figura 3.35 se mencionó al conector CCe. Este conector es un enlace con el diagrama de flujo mostrado en la figura 3.38. Este

diagrama muestra que se lleva a cabo una modificación de la apariencia de la ventana. Prácticamente se despliegan indicadores y controles que le permitirán al usuario configurar su modo de almacenamiento y los parámetros correspondiente al modo elegido. En todos los modos se le permite al usuario fijar un tiempo en segundos, éste es el tiempo de adquisición por archivo; en el caso del modo disparo, el tiempo leído será la duración máxima que puede tener un archivo en caso de que se presente un fenómeno no controlado. Además del tiempo, en todos los modos se puede elegir el formato de tiempo que se quiere utilizar, éstos ya fueron comentados en la parte de requerimiento. En caso de que el usuario haya elegido el modo disparo, se le pide (adicional a lo anterior) que elija el modo de disparo que desea realizar y dependiendo de ello se solicita uno o dos valores de umbral.

En este punto ya se tiene la información de la configuración, se prosigue a ejecuta un sub VI para dar el formato necesario a la información leída de la interfaz y poder plasmarla en un archivo de texto codificada en ASCII.

Sub VI Configura txt

El sub VI *Configura txt*, figura 3.39, empieza preparando un arreglo que contendrá la información agrupada de todos los canales ya configurados por el usuario, continúa a través de un ciclo decodificando la información proporcionada y redistribuyendo la misma. El ciclo *while* se detiene hasta que ha leído la información de todos los canales configurados.

Realizada la lectura de todos los canales configurados, se necesita generar nuevos elementos de texto que permitan identificar a que se refiere cada dato a plasmar en el archivo y buscar la manera de anexarlos al arreglo de datos obtenido previamente. Para lograr lo anterior, se hace uso de varias funciones especializadas de *LabVIEW* que permiten manipular arreglos. Se prosigue a hacer la unión de información. Seguido de esto, a través de la función *Get date/time in seconds,* se obtiene la información de la fecha y la hora actual. Se da paso a modificar esta información a un formato alfanumérico, y ya con el formato adecuado, ésta se usa para asignar el nombre de la carpeta que contendrá los archivos de adquisición. Posteriormente, a este nombre se le agregará información, si es que el usuario proporcionó un identificador en la primera parte de la configuración de adquisición.

Posterior a tener la información unida, se prosigue a ejecutar un conjunto de funciones que permiten crear el archivo de texto que contendrá toda la información de la configuración realizada y que es fundamental para poder llevar a cabo una adquisición, figura 3.40. Por último, antes de finalizar el sub VI *Configura txt,* se manda a llamar otro sub VI que se encarga de crear un archivo. La misión de este archivo es

almacenar la dirección del último archivo de configuración, lo que permitirá, al iniciar una adquisición, conocer con exactitud la localización de la configuración.

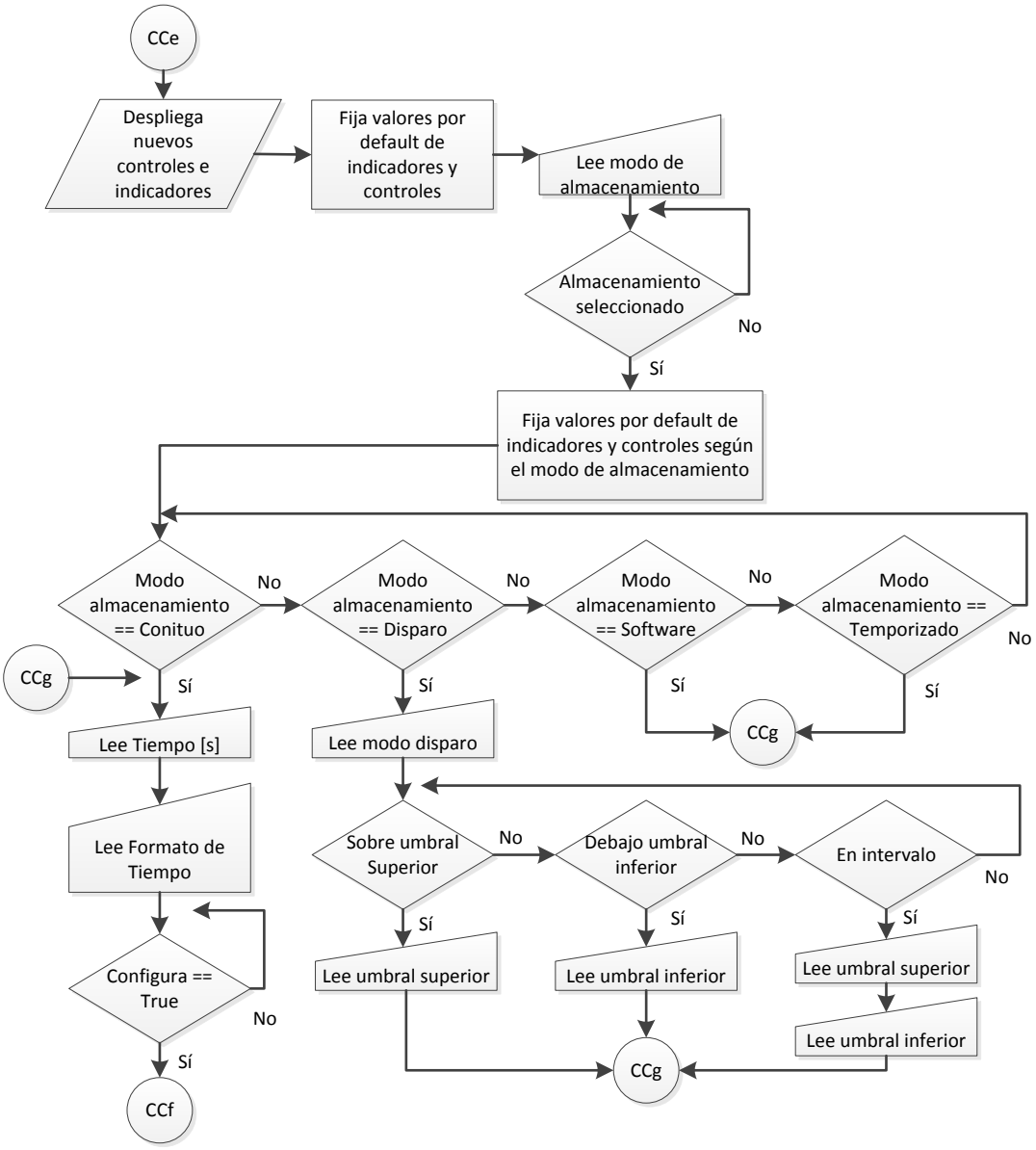


Figura 3.38. Tercera parte del diagrama de flujo del sub VI *Configuración de canales*.

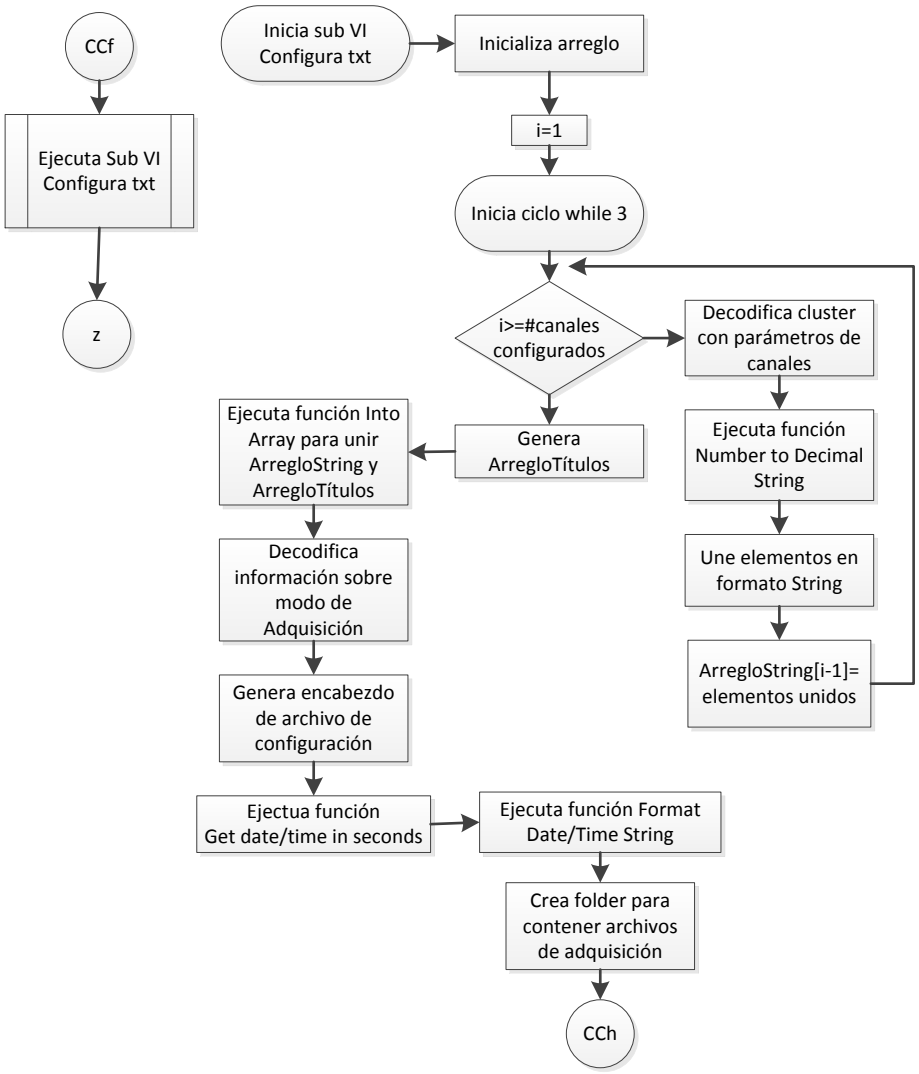


Figura 3.39. Primera parte del diagrama de flujo del sub VI *Configura txt*.

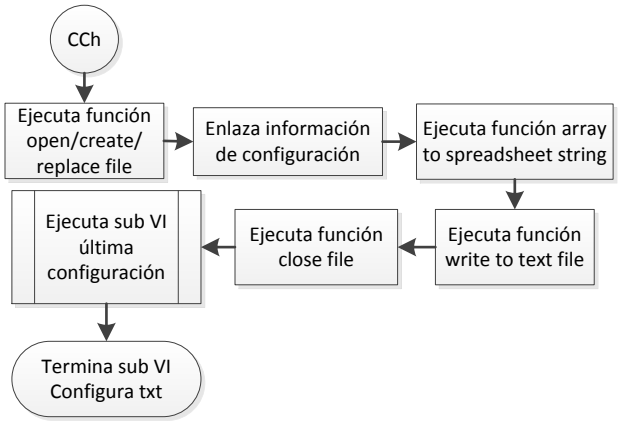


Figura 3.40. Segunda parte del diagrama de flujo del sub VI *Configura txt*.

Sub VI Última configuración

Para lograr crear el archivo de la última configuración, figura 3.41, primero se crea una ruta que apunte al SSD del cRIO 9012, debido a que éste es inmovible, aseguramos que la información de la dirección del último archivo creado esté segura. Hecho lo anterior, se manda a llamar un conjunto de funciones que permiten generar de manera exitosa el archivo de texto a hospedar en el SSD y se extrae la ruta del archivo de configuración creado para poder guardarla.

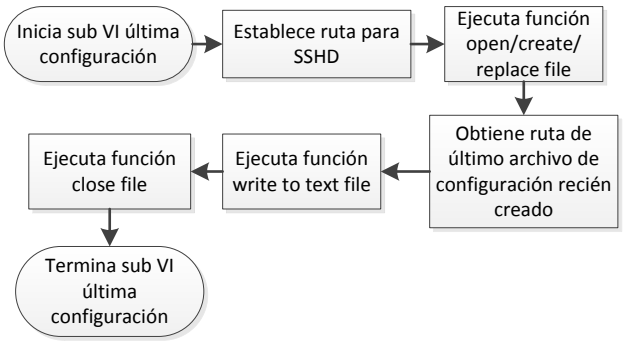


Figura 3.41. Diagrama de flujo del sub VI *Última configuración*.

Cuarta parte del sub VI Configuración Canales

Ya descrito el sub VI *Configura txt* y su sub VI *Última configuración*, retomamos el diagrama de flujo del sub VI *Configuración Canales,* figura 3.42. Nos encontramos en la parte final de este sub VI, se cambia por última vez la apariencia de la ventana, después se indica al usuario si el archivo de configuración se ha creado sin problemas, esto incluye que se haya almacenado de manera correcta en la memoria USB. Si sucedió algún error en el proceso de configuración se verá reflejado en esta instancia; será a través de un indicador que se le mostrará al usuario que todo ha salido bien o todo lo contrario. Si todo ha salido bien, se despliega el mensaje “Archivo de configuración creado”, de lo contrario se despliega “Se ha producido un error”. Para asegurar que el usuario ha leído el mensaje y está listo para finalizar la configuración actual, el algoritmo espera a que el usuario presione un control de verificación antes de pasar a la última acción. Cuando todo marcha bien, la última acción consiste en terminar la ejecución de la ventana *Configuración Canales*. Si ha existido un error, se debe mandar a llamar una función del módulo *Real Time LabVIEW,* la cual nos ayuda a reiniciar el cRIO 9012 y por ende el software. Antes de reiniciar el sistema se cierra la ventana actual.

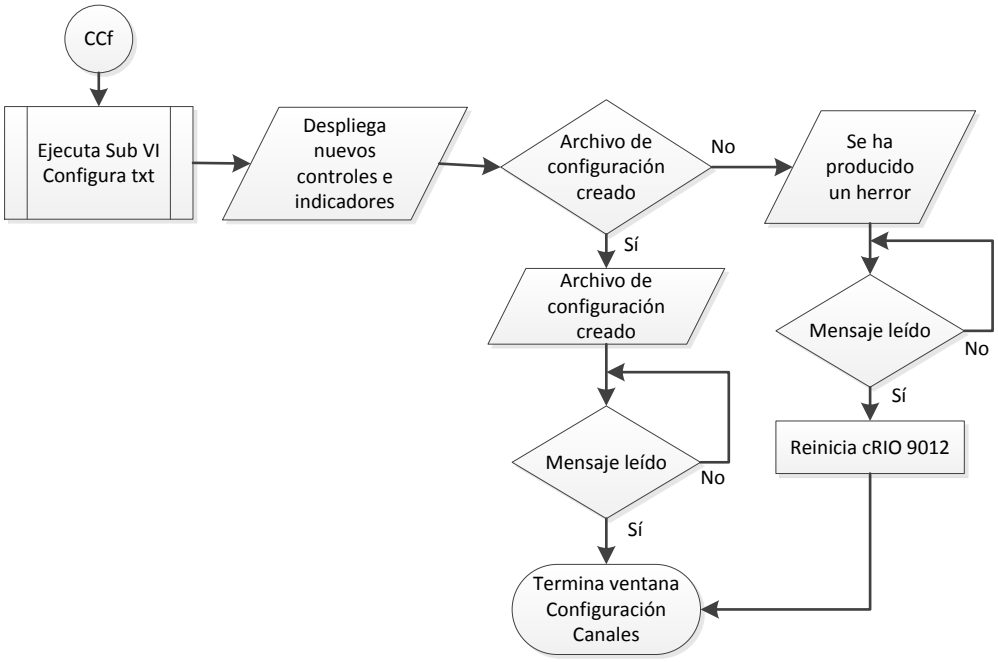


Figura 3.42. Cuarta parte del diagrama de flujo del sub VI *Configuración de Canales*.

Una vez que se ha descrito el sub VI *Configuración Canales*, se procede a hacer lo mismo con el sub VI *Fecha-Hora*.

Sub VI Fecha-Hora

Este sub VI se pone en operación cuando en la ventana web *Principal* se solicita configurar fecha y hora. Es a través de esta rutina que el usuario puede configurar el reloj de tiempo real. De hecho, éste es el encargado de suministrar la base de tiempo en todo momento. La información para configurar el reloj debe ser proporcionada en formato UTC.

Al iniciar la ejecución del sub VI, figura 3.43, se fijan los valores por default de indicadores y controles. Posteriormente, se verifica si el usuario ha seleccionado abandonar la aplicación. Si lo anterior es afirmativo, el sub VI termina su ejecución y la única ventana activa es la página web principal. Si no se ha seleccionado abandonar la aplicación, entonces, se prosigue a monitorear el control *Acción*. Mediante este control, el sistema identifica si se va a realizar una configuración o se quiere probar la configuración actual. Dependiendo de la acción seleccionada, se despliegan los indicadores y controles necesarios.

Cuando se ha seleccionado la opción configuración, lo primero a realizar es el monitoreo del control *Configurar*. Si este control tiene un valor afirmativo, se aplica la modificación pertinente y el sub VI retorna a su punto inicial. Al leer un valor negativo en el control *Configurar,* se prosigue a monitorear un nuevo control. A través de este

control el usuario indica si quiere establecer la fecha o quiere establecer la hora. Si se quiere establecer la fecha, se despliegan tres nuevos controles. Uno de estos controles permitirá establecer el día, otro el mes y el último el año. Cuando se selecciona configuración de hora, se despliega un control para fijar la hora otro para los minutos y un último para los segundos. Tanto para configurar la hora como para la fecha, se construye un *cluster* que agrupa los datos. Una vez que los datos han sido agrupados, son entregados al VI *Set Date and Time.* Esta última función debe actualizar el reloj de tiempo real.

Cuando se selecciona la opción probar, en lugar de la opción configuración, figura 3.43, se despliega un solo indicador. En éste se debe mostrar la fecha y la hora con las que está operando el cRIO. La actualización del indicador se realiza una vez por segundo. La obtención de los parámetros de fecha y hora se hace con la ayuda de la función *Get Date/Time in seconds*.

Sub VI Frecuencia-Ticks

Este sub VI está encargado de hacer la conversión de frecuencia a número de *Ticks,* figura 3.44*.* La señal de reloj por default del FPGA se encuentra operando a 40 MHz, por consiguiente, para obtener la cantidad de *Ticks,* se dividen los 40 MHz entre la frecuencia de muestreo.

Sub VI Cantidad de archivos

Este sub VI ha sido desarrollado con la finalidad de calcular la cantidad de archivos que pueden ser creados. Cuando el usuario introduzca una memoria flash en el puerto USB, el cRIO le asignará por default la ruta u:\. Posterior a tener identificado el dispositivo USB, se ejecuta la función *Get Volume info.* Esta función viene incorporada en el núcleo de LabVIEW y permite obtener datos sobre la unidad de almacenamiento. Ya que se ha ejecutado la función, se extraen los parámetros del tamaño de la unidad y de su espacio libre, ambos están dados en bytes. Se calcula el espacio libre real, figura 3.45, este dato representa el espacio de almacenamiento libre (se considera como máximo el 80% de la capacidad de la unidad). Simultáneamente se realiza el cálculo de bytes por archivo. Para que éste se pueda llevar acabo, se necesita saber la información del número de canales, el tiempo máximo que puede contener un archivo y la frecuencia de muestreo. Adicionalmente se sabe que cada dato debe ocupar 4 bytes. Con toda la información recabada y calculada, se facilita el encontrar el número de archivos que pueden ser creados, figura 3.45.

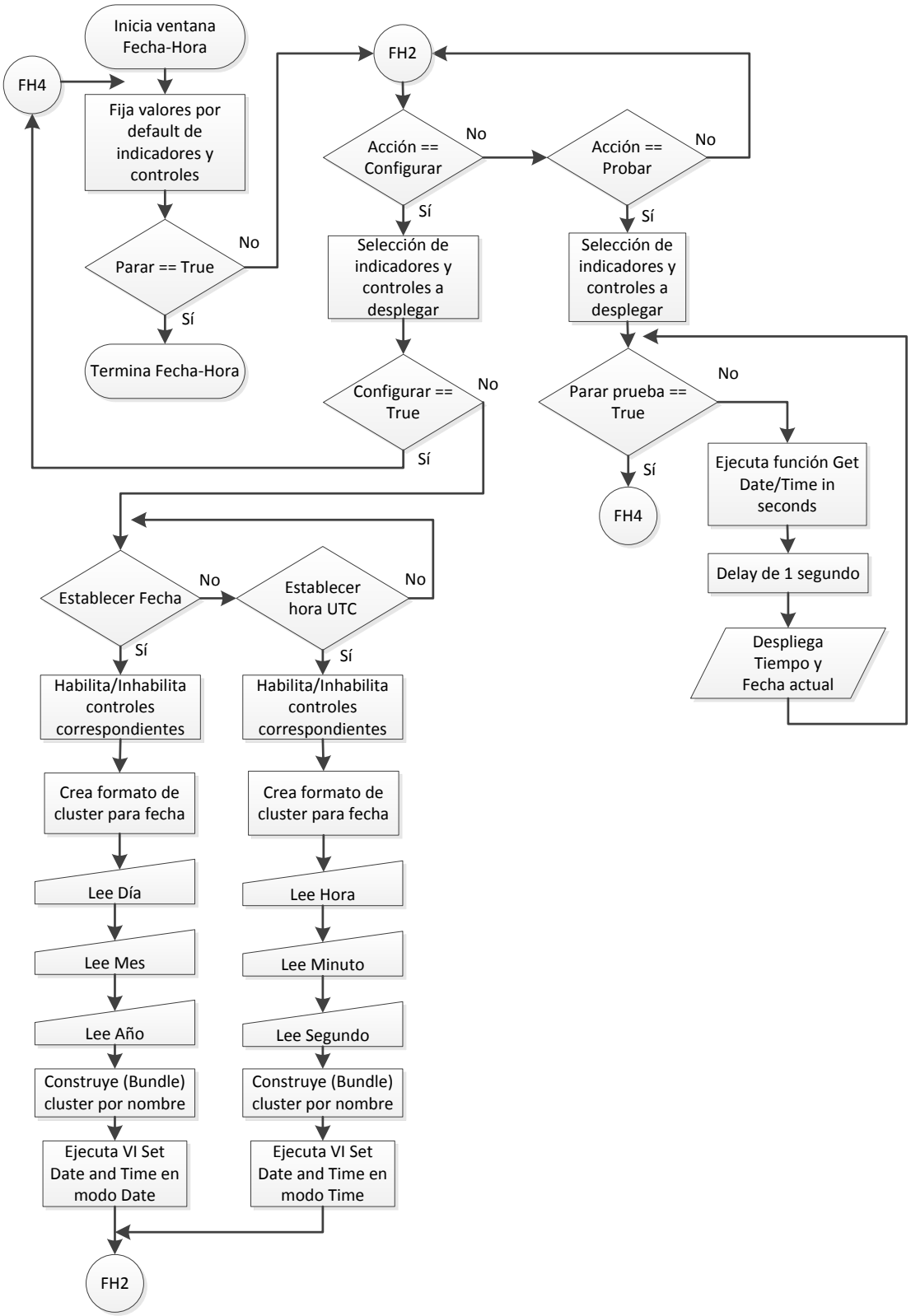


Figura 3.43. Diagrama de flujo sub VI Fecha-Hora.

Figura 3.44. Diagrama de flujo sub VI Frecuencia-Ticks.

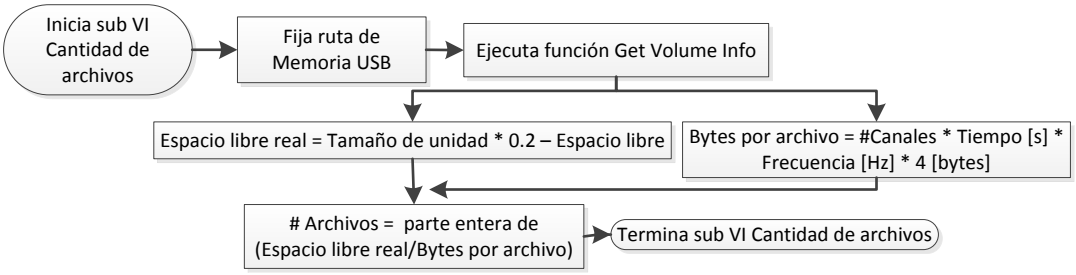


Figura 3.45. Diagrama de flujo de sub VI Cantidad de archivos.

Sub VI Cálculos DMA

Este sub VI tiene la tarea de generar algunos de los parámetros que se necesitan para configurar el *buffer* DMA. Entre otras cosas, el *buffer* necesita conocer el tiempo para adquirir el bloque de datos, el tamaño del bloque datos y el espacio del que dispone para trabajar, figura 3.46. Para conocer el tiempo en milisegundos que debe tardar el MCU en obtener los datos, se multiplica el periodo de muestreo por mil y el resultado de esto se multiplica por el número de muestras. Al tiempo anterior se le denomina tiempo para adquirir bloque. Para conocer la cantidad de datos que se van a leer en cada segundo, se multiplica el número de muestras por el número de canales. A este resultado se le llama tamaño del bloque. Para calcular la cantidad de memoria necesaria para contener los datos en cada ciclo, se multiplica el tamaño del bloque de datos por seis. Al dato obtenido se le conoce como tamaño del *buffer* del host. Es importante resaltar que el *buffer* DMA tiene que ser configurado tanto del lado del MCU como del lado del FPGA. Mediante el sub VI Cálculos DMA sólo se ha generado la configuración necesaria para la parte del MCU.

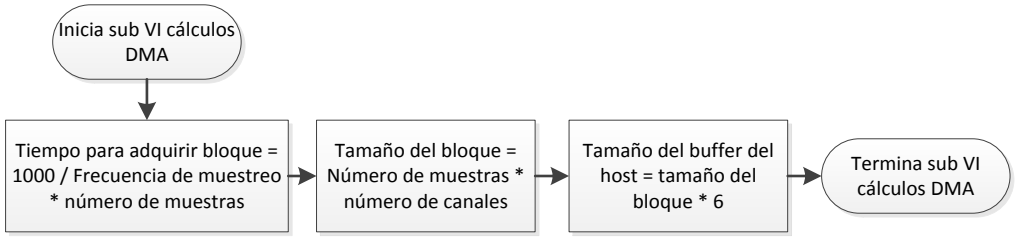


Figura 3.46. Diagrama de flujo de sub VI Cálculos DMA.

Segunda parte de la página Principal

Al iniciar la descripción de los diagramas de flujo, se describió la primera parte del diagrama de flujo asociado a la ventana *Principal*. Se hizo un paréntesis para describir los sub VIs *Configuración canales*, *Fecha-Hora, Frecuencia-Ticks, Cantidad de archivos y Cálculos DMA*. Ahora retomaremos la descripción del diagrama de flujo asociado a *Principal* y que ha empezado a ser descrito en la figura 3.34.

Ya que se han realizado los cálculos para el DMA, figura 3.34, se procede a generar el primer archivo binario que contendrá los datos de la adquisición, figura 3.47. En cuanto se haya generado el archivo (con extensión .dat), el sistema iniciará a monitorear el control *Empieza*. Al tener este control un valor verdadero, se prosigue a verificar la cantidad de archivos permitidos. Si esta cantidad fuera menor que uno, quiere decir que la unidad no tiene espacio ni para un archivo y no se permite iniciar una adquisición. Si existe espacio suficiente para contener al menos un archivo, se prosigue a ejecutar la función *FPGA Open VI Reference*. Esta función permite abrir una referencia al FPGA, lo que inicia formalmente la comunicación entre el MCU y el FPGA. A la función se le dan tres parámetros, éstos son: la ruta del *bitfile* (se genera al compilar el VI que administra los recursos del FPGA), la dirección IP del cRIO 9012 y la indicación de que debe reconfigurar el FPGA en ese instante. Realizado lo anterior, se establecen los valores por default del VI que corre en el FPGA. Para finalizar está parte del algoritmo, se pone en marcha el VI del FPGA y se inicializan los arreglos que se van a utilizar en la siguiente parte del algoritmo.

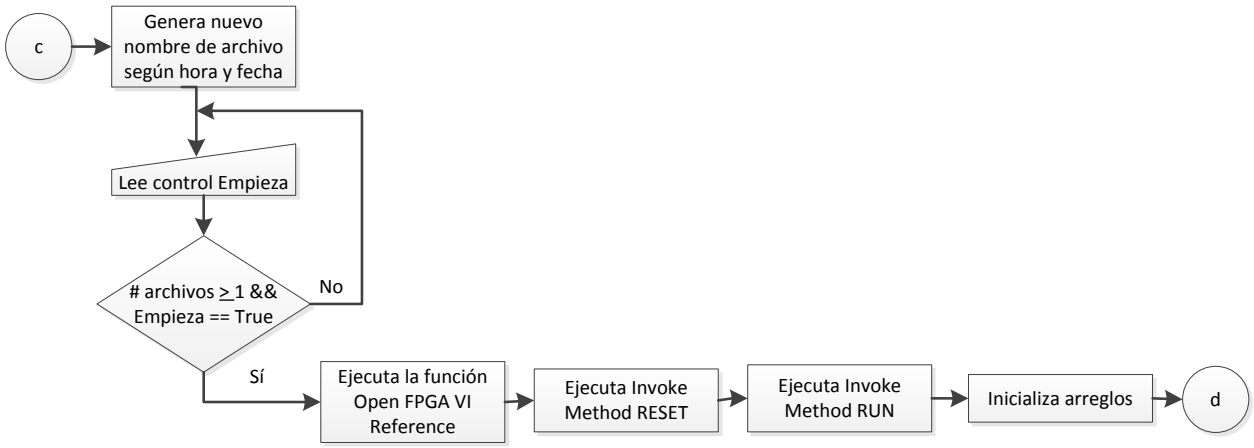


Figura 3.47. Segunda parte del diagrama de flujo de la página Principal.

Tercera parte de la página Principal

Inicializados los arreglos, se inicia un ciclo *while* que se estará ejecutando hasta que mediante el control *Parar*, el usuario indique que ya no quiere adquirir, figura 3.48.

Al iniciar el ciclo, se establece el tamaño del *buffer* DMA. En éste estarán conteniendo todos los datos que viajan del FPGA al MCU y su estructura es First Input First Output (FIFO). El parámetro que fija su tamaño, ya fue calculado con el sub VI *Cálculos DMA.* Por consiguiente, se le indica al buffer que debe comenzar a operar. Con la ayuda de la función Control Read/Write, se transfiere la información contenida en los controles e indicadores del VI actual al VI del FPGA. No se transmite toda la información, sólo la que necesita el VI del FPGA.

El VI del FPGA ya se encuentra en acción y ha recibido toda la información que necesitaba del MCU. A continuación se debe verificar el modo de adquisición que ha elegido usuario, figura 3.48. Para cada modo de adquisición existe un sub VI. En los párrafos posteriores se abordará cada uno de éstos.

Sub VI Modo Continuo

Este modo de adquisición permite adquirir datos hasta agotar el espacio efectivo de la memoria. Su primera tarea es reconocer la información suministrada por el usuario en la configuración, figura 3.49. Esta información consiste en la duración de cada archivo y en el número de canales a procesar. A continuación se indican las muestras por segundo que se van a estar adquiriendo. Posteriormente se da paso a establecer valores iniciales para las siguientes variables: *cuenta de segundos adquiridos, número de elementos a leer y encabezado*.

Una vez leída la configuración para la adquisición e inicializadas las variables, se inicia un ciclo *while, conector MC*. Dentro de éste, lo primero a realizar es la verificación del estado del *buffer* DMA. Para verificarlo se leen tres indicadores pertenecientes al VI del FPGA. El primer indicador se llama *buffer underflow*. Éste nos indica si el buffer se está quedando vacío. Es decir, la velocidad con la que el MCU está leyendo al *buffer*, está muy por encima de la velocidad con la que el FPGA escribe en el *buffer*. El segundo indicador se llama *buffer overflow*. Éste se encarga de avisar que el FPGA está sobre escribiendo el *buffer*. El tercer indicador, *error E/S*, indica si ha existido algún problema con el NI 9205. Basta con que se presente una de las situaciones mencionadas para que el ciclo *while* termine, figura 3.49. Existen otras situaciones que se podrían presentar y obligar a terminar el ciclo *while.* La primera de éstas es que la cantidad de segundos haya sido alcanzada. Es decir, el tiempo por archivo se ha cumplido. La segunda situación se presenta cuando el usuario elije parar la adquisición. Lo anterior se realiza mediante el control *Parar*. La tercera situación se presenta cuando la función para leer *buffer* ha tenido algún problema, cualquiera que éste sea. Como última condición, se puede tener que la función de escritura binaria mande un error.

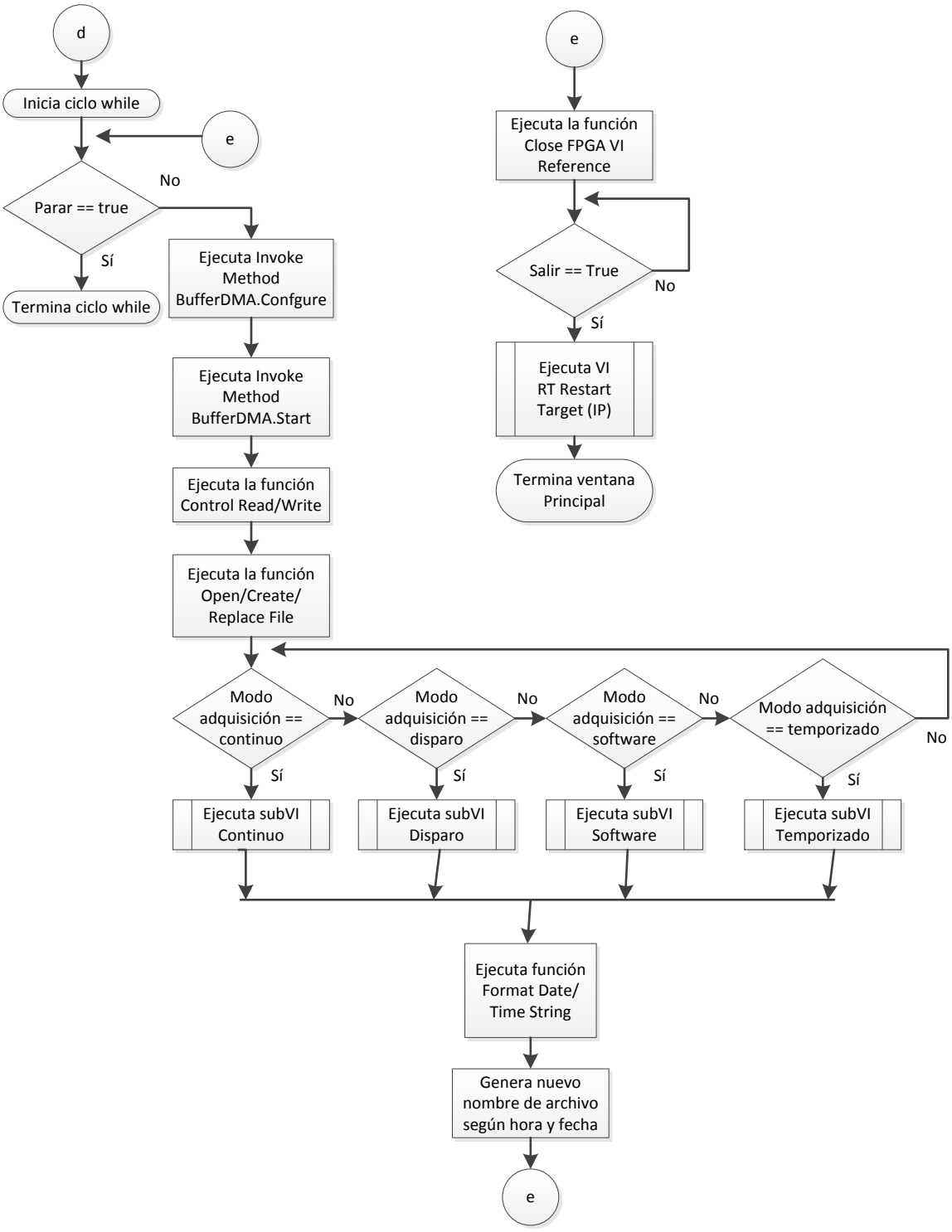


Figura 3.48. Tercera parte del diagrama de flujo de la página Principal.

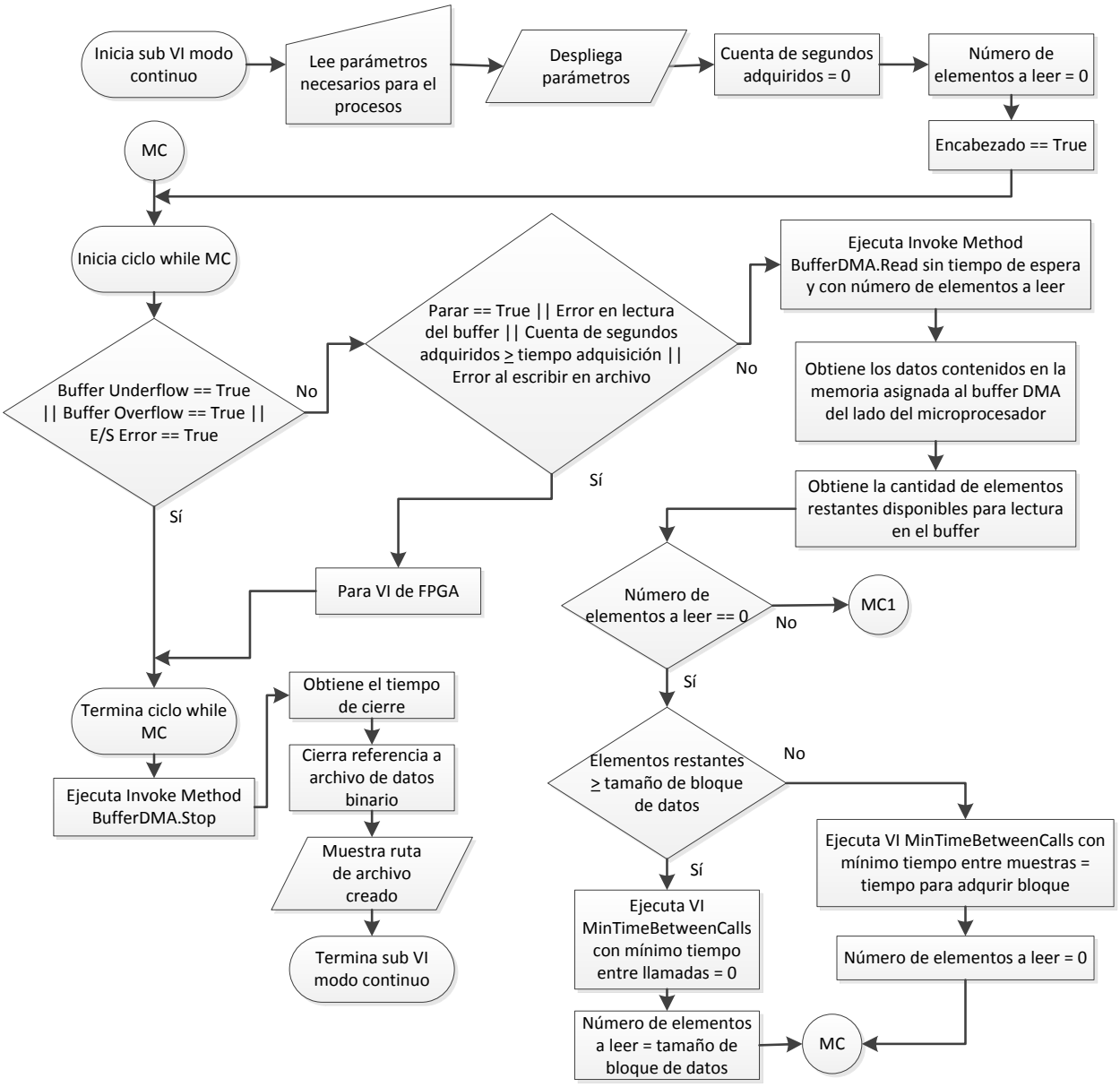


Figura 3.49. Primera parte del diagrama de flujo del sub VI Modo Continuo.

Descritas todas las condiciones bajo las cuales, el ciclo *while* se detiene, se da paso a explicar lo que sucede en cada iteración. Se inicia invocando al método *BufferDMA.Read*, al cual se le proporciona dos parámetros: la cantidad de elementos a leer (calculado con anterioridad) y el tiempo de espera. Éste último se fija en cero, ya que se quiere que la lectura sea inmediata. Una vez ejecutado el método *BufferDMA.Read*, se obtiene dos parámetros de esté. El primero es la cantidad de datos contenidos en la memoria que se ha asignado al *buffer* (sólo se considera la parte asignada del lado del MCU). Efectivamente, este parámetro es un arreglo con enteros de 32 bits sin signo, son los datos que se han pedido leer. El segundo parámetro indica

la cantidad de elementos restantes en el buffer (elementos restantes). Tal y como se observa en la figura 3.49, el segundo parámetro nos ayuda a decir cuando hay que leer. Cuando ya se han leído todos los datos en el buffer, el número de elementos a leer es cero. Si esto ocurre, entonces, se pasa a preguntar si el número de elementos restantes es mayor o igual al número de datos solicitados (tamaño del bloque de datos). Si es mayor o igual, se ejecuta el VI *MinTimeBetweenCalls* con mínimo tiempo entre llamadas igual a 0*.* Además, se modifica el valor de *número de elementos a leer*, siendo ahora igual al *tamaño de bloque de datos*. En contraste, cuando el valor de elementos restantes es menor al *tamaño del bloque de datos*, sucede lo siguiente: el VI *MinTimeBetweenCalls* se ejecuta pero con el valor del tiempo necesario para adquirir un bloque (calculado previamente), adicionalmente, el *número de elementos a leer* toma el valor de cero.

MinTimeBetweenCalls es un VI que forma parte del módulo *Real Time LabVIEW*. Su función, a grande rasgos, es temporizar la lectura de datos. Profundizando un poco, asegura que entre ciclos pase un tiempo mínimo. Este tiempo es el que se necesita para leer un bloque de datos. Este es el único parámetro que recibe y se llama mínimo tiempo entre llamadas. El dato debe ser proporcionado en milisegundos.

Para ver lo que pasa cuando el *número de elementos a leer* no es igual a cero, nos remitimos a la figura 3.50. Al entrar en esta opción, se establece el *número de elementos a leer* igual a cero. Posteriormente, se pregunta por la variable *Grabar*. Si la variable tiene un valor afirmativo, se prosigue a preguntar por la variable *Encabezado*. Cuando esta variable tiene un valor afirmativo, se obtiene el tiempo actual, se le da el formato adecuado y se almacena la información en el archivo binario de adquisición. Almacenado el tiempo, se cambia el valor de la variable a falso. Cuando la variable *Grabar* es positiva pero la variable *Encabezado* es negativa, se escribe el bloque de datos en el archivo binario. Después, se incrementa el contador de segundos adquiridos. El conector MC indica que el ciclo ha terminado, ya que éste es un enlace al inicio del ciclo *while* mostrado en la figura 3.49.

Cuando el ciclo *while* finaliza, se prosigue a parar la ejecución del *buffer* DMA. Inmediatamente después, se obtiene el tiempo de cierre. Lo anterior permite conocer el instante en el que el archivo dejo de almacenar. Por último, se cierra la referencia al archivo de datos y se muestra su ruta física en la ventana asociada al sub VI. Ha terminado el sub VI Modo continuo.

Sub VI Modo Disparo

Debido a la similitud de este sub VI con el sub VI *Modo Continuo*, sólo se explicará en que se diferencia uno del otro. La primera gran diferencia entre *Disparo* y *Continua*, se debe a que *Disparo* despliega una parte del bloque de datos cuando la

variable *Grabar* tiene el valor de falso. El despliegue se realiza a través de una gráfica de número de muestra vs amplitud de la señal. La información a desplegar corresponde a la información de un sólo canal. Este canal a monitorear ha sido elegido en la página web *Principal.* Por lo anterior, antes de entrar al ciclo *while,* es necesario leer el número de canal que ha seleccionado el usuario. La segunda diferencia radica en la existencia de la variable *Primera vez*. Esta variable puede producir que el ciclo *while* pare su ejecución. Su valor inicia por *default* en falso. Cuando se almacena por primera vez un bloque de datos en el archivo binario, el valor de la variable *Primera vez* cambia a verdadero. La tercera diferencia tiene que ver con el monitoreo de un tercer parámetro. Este parámetro es el modo de disparo elegido por el usuario.

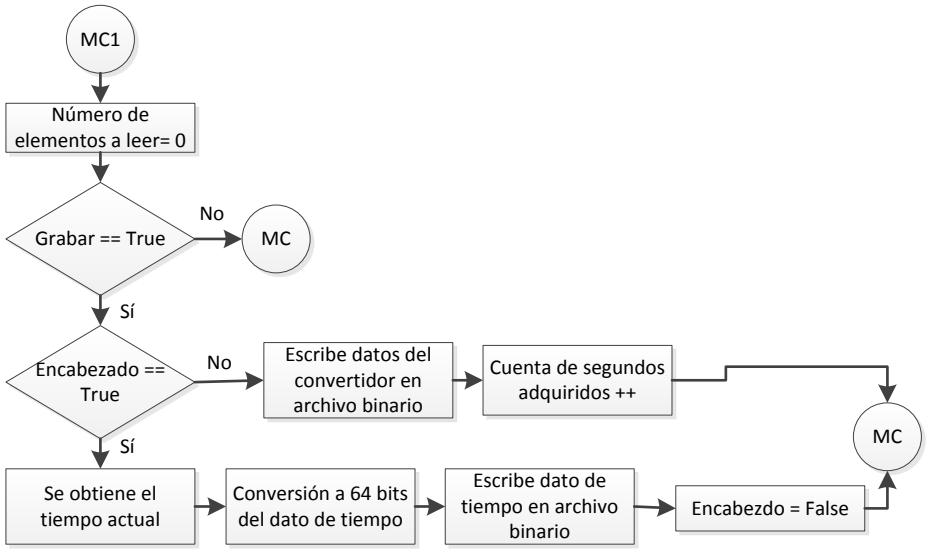


Figura 3.50. Segunda parte del diagrama de flujo del sub VI Modo Continuo.

Expuesto los cambios entre *Modo Disparo* y *Modo Continua*, nos centraremos en dos procesos: el inicio del almacenamiento y el despliegue de datos en forma de gráfica. Estás dos situaciones se ven reflejadas en la figura 3.51. De hecho, se omite parte del diagrama, esto debido a la similitud con el mostrado en la figura 3.49 y 3.50.

Cuando se ha verificado que el *número de elementos a leer* es igual a cero, se pasa al monitoreo de tres variables. La primera variable debe indicar cuando la señal, o las señales, en monitoreo, toman un valor sobre el umbral superior. La segunda variable indica si la señal, o las señales, en monitoreo, tienen un valor por debajo del umbral inferior. La última variable indica si la señal o señales en monitoreo se encuentran por encima del umbral inferior y por debajo del umbral superior. Estas tres variables son actualizadas por el FPGA. Si alguna de ellas contiene un valor verdadero, el algoritmo prosigue a verificar el tipo de disparo que ha seleccionado el usuario. Si el tipo de disparo seleccionado coincide con el caso del disparo, entonces, se prosigue a cambiar el valor de la variable *Grabar* a verdadero.

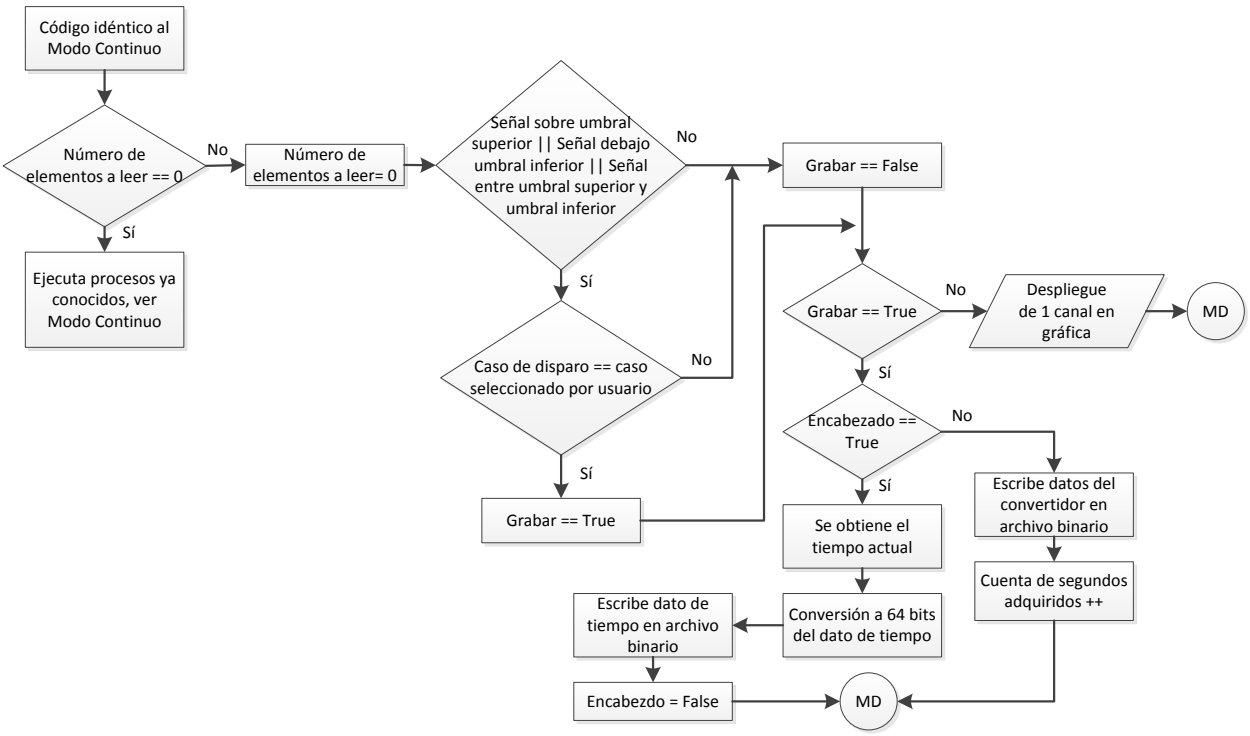


Figura 3.51. Diagrama de flujo del sub VI Modo Disparo.

Para el caso de *Modo Continua*, cuando *Grabar* tenía el valor de falso, había un retorno al inicio del ciclo *while*. Para el caso de *Disparo*, si *Grabar* tiene el valor de falso, se prosigue a graficar los datos del canal en monitoreo. Con la finalidad de que lo anterior pueda llevase a cabo, es necesario un proceso de división y conversión de la información, figura 3.52. El sub VI que se encarga de llevar a cabo el proceso mencionado se llama *Separa Canales*.

El sub VI *Separa Canales* empieza leyendo la cantidad de canales que se están adquiriendo, el número de canal que se quiere desplegar en gráfica y evidentemente, el boque de datos. Según la información recopilada, se identifica la forma en que se debe segmentar el bloque. Únicamente se quiere tener la información correspondiente al canal en monitoreo. Por segundo se va a desplegar una cantidad de datos igual a la frecuencia de muestreo. Estos datos forman un arreglo, para recorrerlo hacemos uso de un ciclo *while*. Al iniciar cada ciclo se pregunta por la cantidad de iteraciones. Cuando el número de iteraciones es igual o mayor al número de muestras por segundo, el ciclo termina y con ello la ejecución del sub VI. En el caso contrario, se lee un dato del arreglo, se convierte a formato binario, se incrementa el contador de iteraciones y se realiza un último cambio de formato al dato. Al terminar el proceso, el dato se encuentra en un formato de punto fijo.

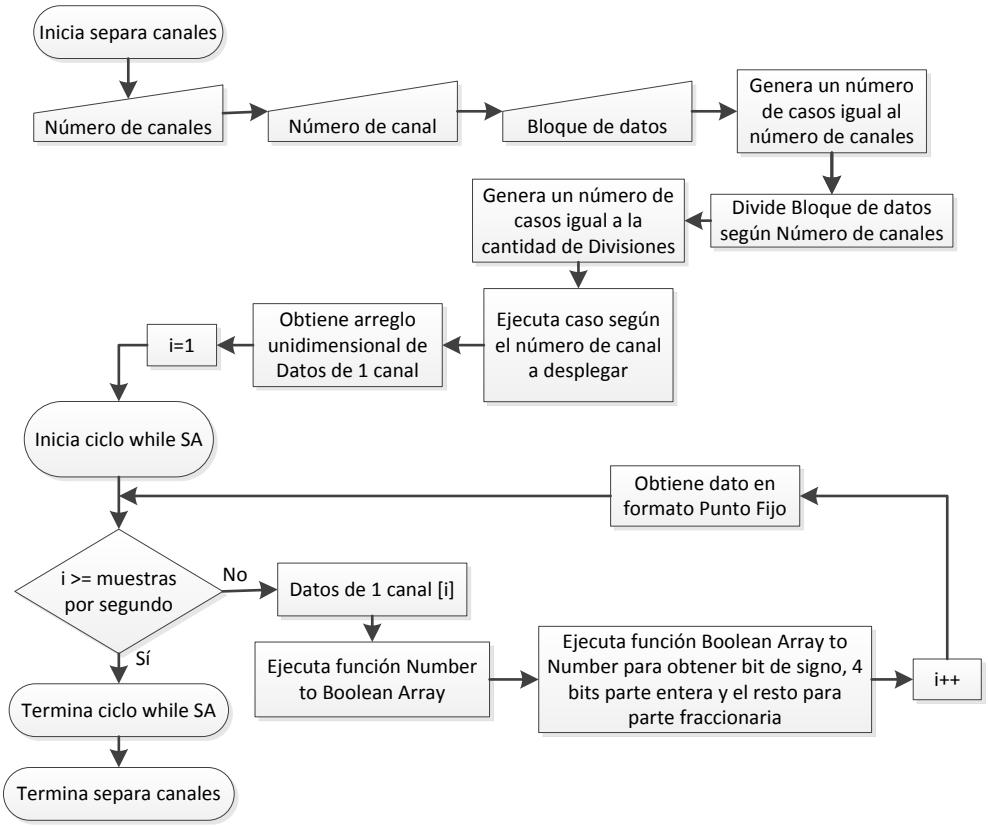


Figura 3.52. Diagrama de flujo del sub VI Separa canales.

Sub VI Modo Software

La ejecución de este sub VI se encuentra muy relacionada con el sub VI *Modo Disparo*. Evidentemente, esta relación implica que la ejecución del sub VI *Modo Software,* también está relacionada con la ejecución del sub VI *Modo Continua*. Una vez más, aprovechando la similitud de este sub VI con los anteriores, únicamente se abordará las diferencias entre éstos.

El sub VI *Modo Software* empezará mostrando un canal de manera gráfica. Iniciará a almacenar hasta que el control *Grabar* sea presionado. Mientras el control Grabar no vuelva a ser presionado, el sistema continuará almacenando información. Esto significa la principal diferencia con los sub VIs antes mencionados.

Aunque el *Modo Software* está diseñado para que el usuario inicie y pare el proceso de almacenamiento, el contador de segundos adquiridos sigue operando. Es decir, si el usuario deja pasar 600 segundos de adquisición y no para la adquisición, la condición de tiempo parará la adquisición.

Sub VI Modo Temporizado

Como es de esperar, este sub VI tiene mucha semejanza con los anteriores. Este sub VI permite definir un intervalo de tiempo para la adquisición. La selección del tiempo ya fue hecha en la etapa de configuración. Cuando inicia la operación del sub VI, por default se encuentra la variable Guardar en falso y por lo tanto, se realiza el despliegue de un canal en la gráfica. Para iniciar el almacenamiento, es necesario cambiar a verdadero el valor del control *Grabar*. Una vez que el tiempo establecido se ha cumplido, se para la adquisición y se prepara para iniciar una nueva.

Se ha terminado la descripción de los modos de almacenamiento, por lo tanto, se continúa con la descripción de la parte final de la figura 3.48. En el punto que nos encontramos, ya se ha llevado a cabo un proceso de adquisición. El mecanismo de almacenamiento que se llevó a cabo, forzosamente tuvo que estar controlado por alguno de los sub VIs descritos en los párrafos anteriores.

Al salir de algunos de los modos de almacenamiento, se obtiene el tiempo en el que esto ocurrió y posteriormente se genera un nuevo nombre de archivo. El nombre del archivo contendrá información sobre la hora actual. Se manda a llamar la función *Close FPGA VI Reference*. Esta función debe cerrar la comunicación entre el FPGA y el MCU. A continuación, se pregunta por el estado del control *Salir*. Si este control tiene un valor verdadero, se prosigue a ejecutar el VI *RT Restart Target (IP).* Este VI forma parte del módulo *LabVIEW Real Time* y permite reiniciar al cRIO. Esta función necesita como parámetro la dirección IP del cRIO. Ejecutada la función anterior, se ha terminado con la ejecución de la ventana *Principal*. Si el control *Salir* tiene el valor de falso, el sistema espera a que el valor sea verdadero. Cuando se termina la ejecución de la ventana *Principal*, el navegador indica que no encuentra el VI a ejecutar.

3.3.3. Software del FPGA en el chasis 9113

Requerimiento

Se quiere contar con una aplicación que administre al NI 9205 y que mande la información obtenida al MCU. La aplicación debe poder controlar todos los recursos del NI 9205. Además de controlar los recursos, debe permitir explotar al máximo el NI 9205. Lo anterior tiene que ver con: trabajar con velocidades variables de transmisión de datos entre el NI 9205 y el FPGA, y poder leer los 32 canales o un subconjunto de ellos.

La administración involucra el control y monitoreo de todos los elementos lógicos y físicos involucrados en la conversión analógico-digital, temporización para la lectura de canales y temporización para la transferencia de información. Para logar lo anterior, se hace necesario que el software del FPGA se comunique constantemente con el software del MCU.

La velocidad de intercambio de información FPGA-MCU, forzosamente involucra a la frecuencia de muestreo y a la capacidad del *buffer* DMA. Se necesita que el software permita operar bajo distintas frecuencias de muestreo y diferentes cantidades de información.

La aplicación debe contemplar la selección del número de canales a procesar. Lo anterior implica que el usuario puede elegir adquirir un solo canal, adquirir 8 o adquirir 32. Un razonamiento rápido permite pensar en una cantidad grande de posibilidades (subconjuntos). Las configuraciones permitidas se reducen a 18. Esta cantidad es resultado de las restricciones inherentes a la lógica implementada y al compromiso de ésta con los recursos de hardware. Sin embargo, esas 18 configuraciones contemplan los siguientes aspectos interesantes: uso del máximo número de canales disponibles (32 en modo RSE o NRSE), uso de la máxima frecuencia de muestreo posible (1 canal) y 16 canales en modo diferencial.

Por lo redactado en el párrafo anterior, se infiere que el software debe encargarse de la configuración de los canales analógicos. Ya se explicó que los parámetros de configuración son proporcionados por el software del MCU, pero es el software del FPGA el que debe culminar la acción de configuración.

El FPGA debe temporizar la lectura de los canales y posteriormente la transmisión de los datos obtenidos. Además, se debe de encargar de monitorear el valor de los primeros tres canales. Posterior a este monitoreo, el software debe indicar al MCU si la señal se encuentra en alguno de los tres estados de disparo. La identificación del estado la debe hacer en un ciclo de reloj. Se deben usar funciones de *LabVIEW FPGA* para leer los canales, esto incorpora limitantes considerables. Cada configuración de lectura diferente necesita una función diferente. Es decir, la función de lectura debe ser configurada de una forma para leer un canal y de otra forma para leer ocho canales. Cada configuración diferente reduce sustancialmente la capacidad del FPGA.

Existen limitantes inherentes a la transferencia de información entre el FPGA y el MCU. Estas limitantes se deben a las funciones que se utilizan para leer y escribir al *buffer* DMA. Las funciones además de permitir el intercambio de información, sincronizan al FPGA con el MCU. A pesar de que la transferencia por DMA no es la única opción, si es la más rápida y la que permite el manejo de una mayor cantidad de datos.

Cuando se utiliza comunicación DMA, el FPGA puede usar la memoria RAM del MCU como si fuera propia. Esto ofrece un mejor desempeño en comparación con otras técnicas de comunicación, por ejemplo, la que se utiliza para leer y escribir los indicadores y controles que comparten el FPGA y el MCU. La transferencia por DMA se

complementa usando *buffers* con arquitectura FIFO. El *buffer* FIFO está compuesto por dos partes que se comportan como una. Una parte de este *buffer* FIFO se encuentra en el FPGA y usa bloques de RAM del mismo FPGA. La segunda parte del *buffer* FIFO se encuentra en el MCU y usa memoria RAM del mismo MCU. A través de funciones de escritura/lectura otorgadas por LabVIEW FPGA, se hace posible la transferencia de información de un dispositivo a otro.

Se desea explotar al máximo el módulo *LabVIEW FPGA* y sus funciones robustas. Es por ello que se requiere fundamentar el diseño en la programación gráfica. Sin embargo, debe hacerse notar que el ambiente de desarrollo de *LabVIEW* soporta descripción de hardware y programación gráfica, inclusive simultáneamente.

Descripción de la operación

Toda la lógica que controla al FPGA estará resguardada en un VI que llamaremos VI del FPGA. Como es de esperar este VI es extenso, por ello lo dividiremos en 7 pequeñas secciones. Las primeras tres servirán para explicar los algoritmos que necesita el modo disparo. Después daremos paso a explicar la configuración del NI 9205, cuarta sección. La quinta sección abordará la temporización de la adquisición y transmisión. Posteriormente dedicaremos la sexta sección para la lectura del NI 9205. Por último habrá una sección para la transferencia de datos hacia el MCU por DMA, séptima sección.

Los tres diagramas que a continuación se presentan, se relacionan a los tres modos de disparo (éstos han sido plenamente explicados en el apartado 3.3.2). Se presentará uno para cada modo, y se debe tener en mente que se ejecutan de manera simultánea. Los tres procesos se ejecutan en un ciclo de reloj. Ya se ha abordado el paralelismo que ofrece un FPGA.

Disparo sobre umbral superior

Mientras el VI del FPGA esté activo, la rutina Disparo sobre umbral superior se debe encontrar operando. Para lograr lo anterior se hace uso de un ciclo *while*. Este ciclo únicamente parará cuando el VI del FPGA también lo haga, figura 3.53. Una vez que el ciclo *while* empieza, se configura la frecuencia de operación de un *Timed Loop*. Esta estructura es parecida a un ciclo *while*, pero tiene la capacidad de recibir varios parámetros de operación. Para el presente trabajo sólo se usa uno de tantos parámetros, la frecuencia de operación. Ésta se fija a 40 MHz, la frecuencia a la que estará trabajando el FPGA. Antes de entrar al *Timed Loop*, se fija el valor de falso a la variable llamada *valor final*.

Cuando la estructura *Timed Loop* ha comenzado, rápidamente trata de leer tres indicadores, figura 3.53. Estos tres indicadores pertenecen a los tres primeros canales

analógicos del NI 9205. Estos valores son actualizados en otra parte del VI, ésta será abordada en páginas posteriores. Una vez leídos los valores, se hace una comparación entre éstos y el valor de umbral superior. Si alguno de los tres indicadores no tiene un valor mayor al del umbral superior, entonces, se almacena el resultado y se vuelve a checar. Por otro lado, cuando el alguno de los indicadores tiene un valor mayor que el de umbral superior, se prosigue a lo siguiente: se almacena el resultado de la comparación y se realiza una operación lógica XOR con el valor actual y el valor anterior de la comparación. Si el resultado de la XOR es falso, la estructura *Timed Loop* vuelve a empezar su accionar. Si el resultado de la XOR es verdadero, la estructura *Timed Loop* termina y el indicador *disparo sobre umbral superior* se actualiza. El valor de este indicador dependerá del valor de la variable *valor final*. Ha terminado una iteración del ciclo *while*.

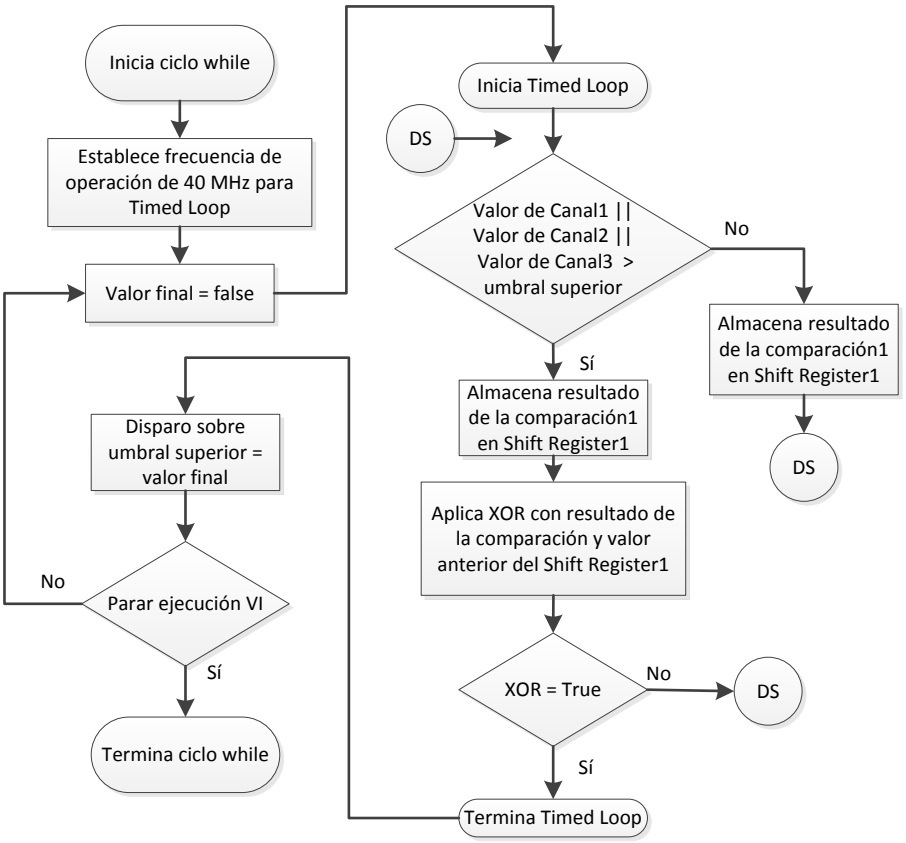


Figura 3.53. Diagrama de flujo para disparo sobre umbral superior.

Disparo debajo umbral inferior

A pesar de que esta rutina es muy parecida a la anterior, figura 3.54, se ha decidido poner su diagrama de flujo correspondiente. La primera diferencia entre este diagrama y el anterior, se encuentra en el inicio de la estructura *Timed Loop*. Es decir,

en el comparador de magnitudes. Si el valor de alguno de los tres canales está por encima del umbral inferior, se almacena el resultado de la comparación y retorna a preguntar otra vez por los valores de los canales. En el caso contrario, cuando alguno de los valores es inferior al umbral inferior se prosigue a lo siguiente: se almacena el resultado de la comparación y se actualiza el valor de la variable *valor final* a verdadero. En este punto se realiza una operación lógica XOR con el valor de comparación actual y el valor pasado. Se prosigue de manera idéntica a la rutina anterior (disparo sobre umbral superior). El indicador en este caso se denomina *disparo debajo umbral inferior.*

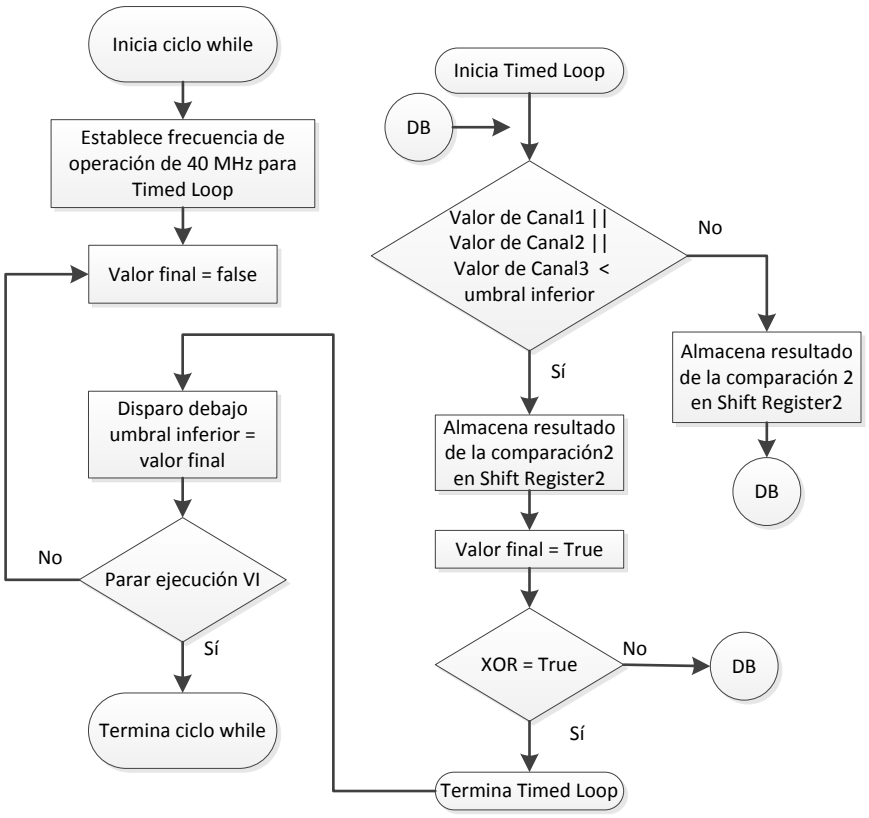


Figura 3.54. Diagrama de flujo para disparo debajo umbral inferior.

Disparo en intervalo

Este modo de disparo involucra los dos anteriores. Únicamente haremos hincapié en las diferencias con los otros dos. Este modo es evidentemente más extenso, figura 3.55. Para poder determinar el estado del indicador *disparo en intervalo,* esta rutina necesita pasar forzosamente por dos comparaciones de magnitud. *Disparo en intervalo* debe indicar si la condición de “en intervalo” se ha cumplido. Cuando éste tiene un valor verdadero, alguna de las tres señales tiene un valor dentro del intervalo establecido. Por el contrario, cuando este indicador tiene un valor de falso, ninguna de las tres señales tiene su magnitud en el intervalo. No se debe perder de vista que para

este caso, se requiere estar por encima del umbral inferior y no por debajo. Además, se requiere estar por debajo del umbral superior y no por encima.

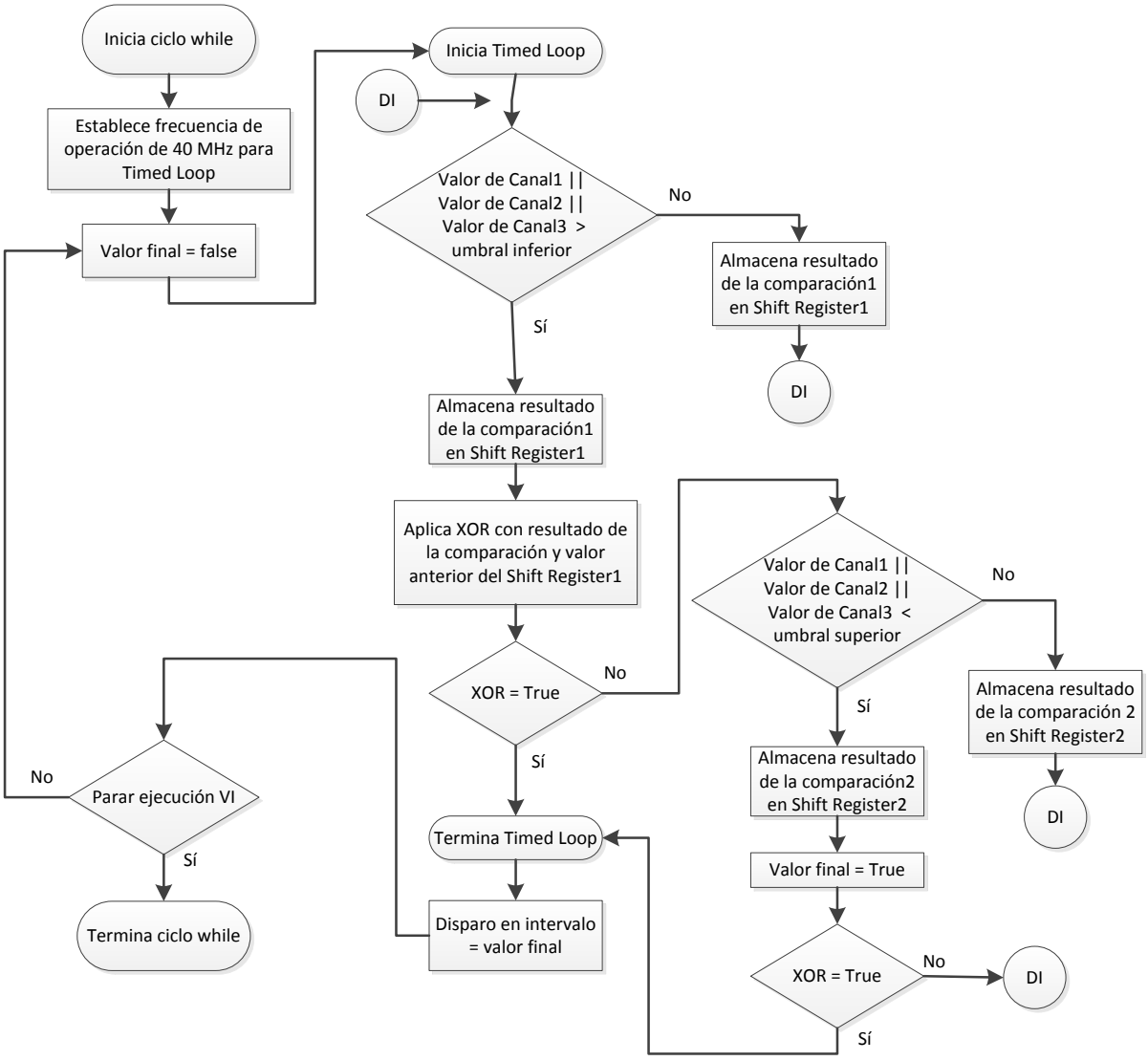


Figura 3.55. Diagrama de flujo para disparo en intervalo.

Inicio y configuración

Paralelamente a los tres algoritmos anteriores debe correr el “motor de tiempo”, figura 3.56. El motor de tiempo es un ciclo *while* que contiene desde el proceso de configuración hasta la transmisión de los datos. Al iniciar el ciclo lo primero a realizar es preguntar por el control *Empezar*. Mientras el control *Empezar* no tenga un valor de verdadero, motor de tiempo no puede avanzar. Una vez que *Empezar* tiene un valor de verdadero, se da paso a establecer las condiciones iniciales de todos los indicadores y controles. Posteriormente, el FPGA recibe del MCU el número de canales configurados

por el usuario. Ya que el FPGA identificada la cantidad de canales, éste prosigue a solicitar el arreglo de *clusters* que contiene la configuración de los canales. A continuación se da paso a recorrer todo el arreglo para ir extrayendo la información de los canales. Para esto se utiliza un ciclo llamado ciclo de configuración, figura 3.56. El ciclo tendrá un número de iteraciones igual al número de canales configurados. Para cada canal se quiere obtener el rango de voltaje y el modo de medición (ver sección 3.3.2). Ya que se han obtenido los dos parámetros anteriores, se ejecuta el método *FPGA I/O* en dos modos por cada canal. Un modo, Set Voltage Range, permite indicarle al NI 9205 el rango de voltaje que se quiere medir. El otro modo, *Set Terminal,* permite indicarle al NI 9205 si el modo va a ser diferencial, NRSE o RSE.

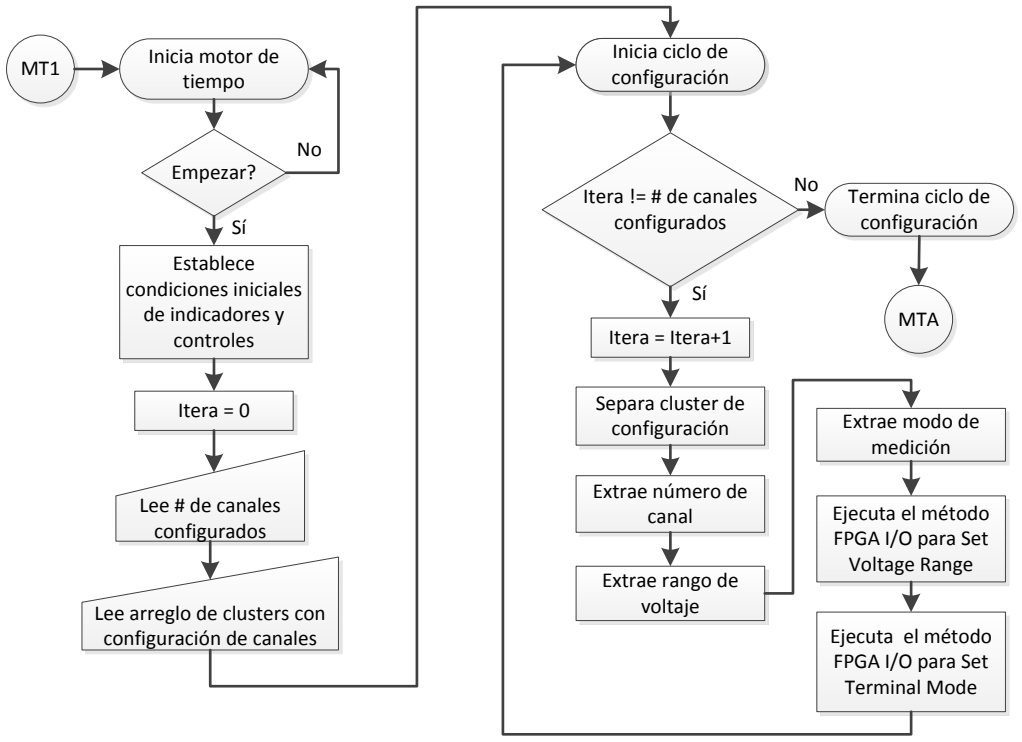


Figura 3.56. Diagrama de flujo de inicio y configuración del VI FPGA.

Temporización de lectura y transmisión

Ya que el FPGA ha configurado al NI 9205, se procede a iniciar el ciclo de adquisición. Este ciclo prácticamente se puede abordar en tres partes: temporización de la adquisición, lectura del NI 9205 y transmisión de datos al MCU.

El proceso de temporización prácticamente se efectúa con el VI *Monitor Loop Rate* y con el VI *Loop Timer*. *Loop Timer* es un VI que permite que un ciclo sea ejecutado en un intervalo específico de tiempo. Cuando una parte del código en el ciclo no alcanza a ejecutarse, *Loop Timer* establece una nueva referencia de tiempo para las

siguientes iteraciones. La primera vez que *Loop Timer* se ejecuta, guarda el tiempo actual. En la segunda ejecución de *Loop Timer*, éste suma la duración solicitada del ciclo al tiempo inicial y espera a que se cumpla el intervalo. El intervalo de tiempo se especifica en *Ticks* (ciclos de reloj) y es una de las tres entradas que necesita el VI. Este dato ya ha sido generado en el MCU con el sub VI *Frecuencia-Ticks*. La segunda entrada del VI es la cantidad de bits del contador interno, se ha establecido de 32 bits. La tercera entrada es para decidir las unidades de la cuenta. Las unidades establecidas son *Ticks*. *Loop Timer* cuenta con una sola salida llamada *Tick Count*, figura 3.57. En esta salida se ve reflejada la cantidad de *Ticks* solicitados o en su defecto, la cantidad de Ticks que *Loop Timer* calcula se necesitan para ejecutar toda la lógica del ciclo.

Cuando *Loop Timer* incrementa el tiempo de operación del ciclo, se debe a que la lógica necesita más tiempo por todos los retrasos inherentes. Evidentemente, si el tiempo de operación se incrementa aleatoriamente, estamos incurriendo en un error. El ciclo debe ejecutarse en el tiempo calculado por el MCU. Para saber que el tiempo solicitado ha sido modificado por *Loop Timer*, se hace uso del VI *Monitor Loop Rate*. Este VI calcula el número de *Ticks* que han pasado desde la última vez que fue llamado y lo comparará con la cuenta deseada. La cuenta deseada es proporcionada por *Loop Timer* mediante *Tick Count* (salida del VI). Si más *Ticks* de los deseados han pasado, el VI genera un indicador *booleano* con valor verdadero. Cuando no se supera el número de *Ticks* deseados, el indicador toma el valor de falso. El indicador mencionado es el llamado *Buffer Underflow*, abordado en el software para el MCU. Una vez que este indicador tomar el valor verdadero, la adquisición se detiene y se necesita reiniciar toda la parte de adquisición de datos, figura 3.57. Cuando se reinicia el ciclo de adquisición, es necesario reiniciar el estado de *Buffer Underflow.* Para lograr esto se debe mandar un valor verdadero a la entra llamada *Reset* del VI *Monitor Loop Rate*. Para el presente trabajo, se están usando las dos entradas que permite *Monitor Loop Rate*, pero sólo una salida del mismo. Las entradas son el Reset y la cuenta de Ticks requerida (Count). La salida es la que modifica al indicador *Buffer Underflow*.

Lectura

Para poder realizar la lectura de los canales analógicos, se necesita usar la función *FPGA I/O,* figura 3.57. Mediante esta función es posible tener acceso a todas las entradas y salidas de los módulos conectados al NI 9113. Esta función está diseñada para ser ejecutada sólo en el FPGA. A la función se le tiene que indicar la cantidad de entradas o salidas que se quiere leer o escribir. Una vez que la función se esté ejecutando, el número de entradas/salidas no puede ser modificado. Lo anterior implica que todas las variantes que se quieran tener del *FPGA I/O*, deben ser establecidas previa generación del *bitfile*.

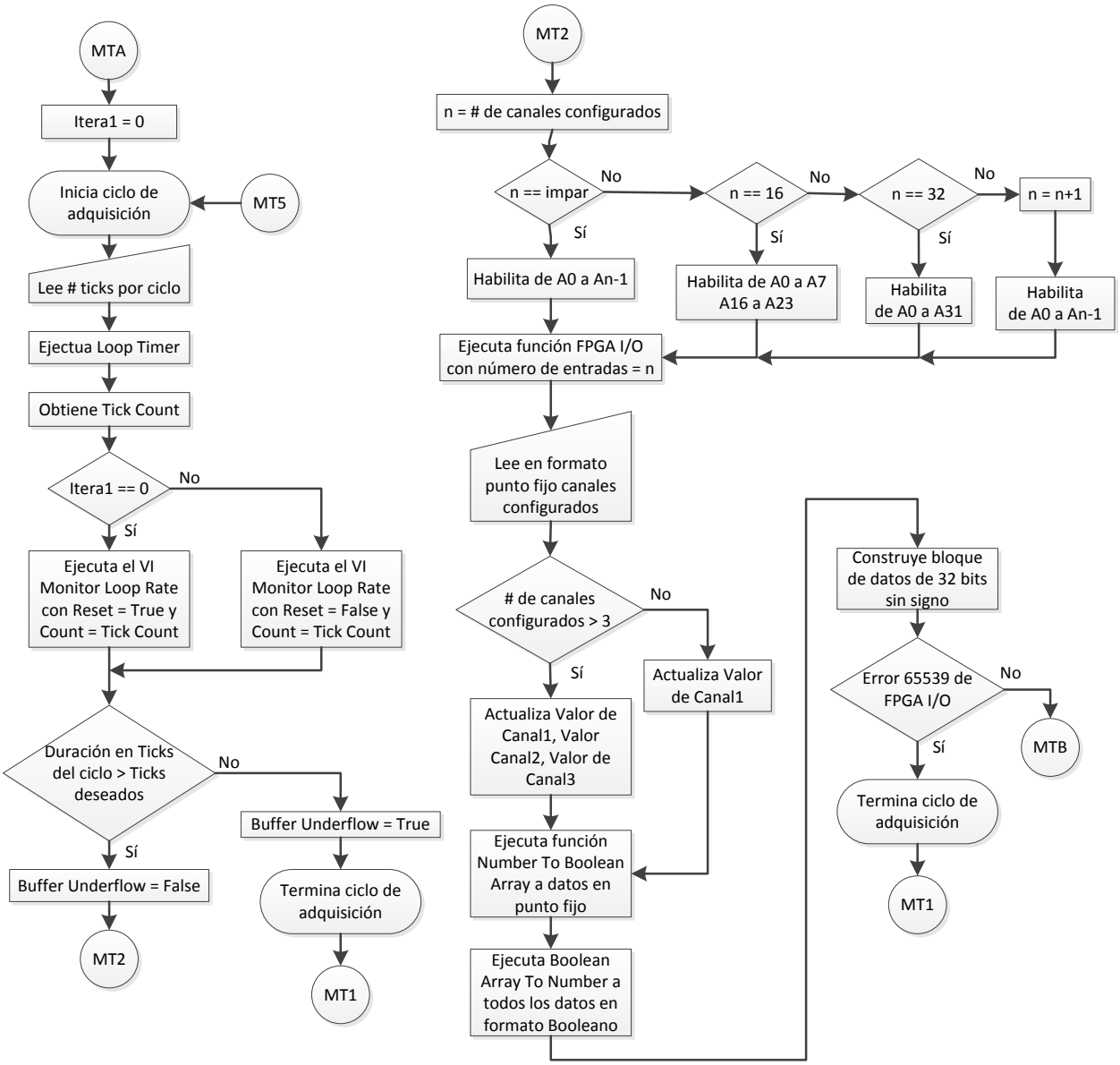


Figura 3.57. Diagrama de flujo para adquisición del VI FPGA.

En toda la figura 3.57, *n* indica la cantidad de canales configurados. Si el número de canales es impar, se habilitan *n* entradas a través de la función *FPGA I/O*. Si el número de canales es par, entonces se consideran *n+1* canales habilitados. Lo anterior significa que cuando el MCU quiera leer 4 canales, el FPGA leerá 5. A pesar de que el FPGA está mandando bloques de información con 5 canales, el MCU sólo almacena lo perteneciente a 4 canales. Es decir, el MCU sólo almacena lo que desde un principio solicitó. Por ejemplo, para el caso en el que los canales se configuran en modo terminación simple, si iban a ser 32 configuraciones de la función *FPGA I/O*, ahora serían 16. Por ejemplo, la configuración para leer 5 y 4 canales es la misma en el FPGA. Esto trae como consecuencia un ahorro en recursos del FPGA.

La frecuencia de muestreo está limitada por la cantidad de entradas que se solicitan a través de la función *FPGA I/O*. De hecho, en el apartado 3.3.2 se presentó una gráfica de la relación existente entre número de canales y la frecuencia de muestreo (figura 3.33). Por ejemplo, cuando se tiene 3 canales, la frecuencia de muestreo máxima es 32,000 Hz, cuando se tiene 5 canales es 20,000 Hz. Con base en esto, y lo dicho en el párrafo anterior, se puede afirmar que la máxima frecuencia de muestreo para 4 canales es 20,000 Hz y para 2 canales es 32,000 Hz. Existen dos casos que se salen de la lógica anterior, cuando se adquieren 16 canales en modo diferencial y 32 en modo terminación simple, figura 3.57. En el primer caso, como se aprecia en la tabla 3.2, existe un salto entre el canal diferencial 7 y el canal diferencial 16, es decir, no están contiguos. Esto genera que deba existir una configuración especial para lograr medir 16 canales en modo diferencial. Para el segundo caso, cuando se quieren leer 32 canales en modo terminación simple, también se necesita una configuración adicional. Esto se debe a que la configuración para leer 31 canales se comparte con la de 30 canales, entonces, la configuración para 32 se queda sólita.

Resulta importante aclarar que en el caso de las mediciones en modo diferencial, lo que se lee es el nodo positivo de la conexión. En la parte de configuración perteneciente al VI del FPGA, cuando un canal se configura como diferencial, se le está indicando al NI 9205 que se reconfigure internamente y que mande la medición diferencial en el nodo positivo de la conexión, tabla 3.2.

Ya que se ha habilitado el número de nodos necesarios en la función *FPGA I/O*, se prosigue a mandar los datos de los tres primeros canales a los ciclos de disparo. En el caso de que sólo se haya habilitado un canal, entonces, sólo se actualiza el indicador perteneciente al primer canal. El formato de los datos se encuentra en punto fijo, usa una palabra de 26 bits, de los cuales 4 bits son para la parte entera, 1 bit para el signo y 21 bits para la parte fraccionaria. Después de haber mandado la información a los ciclos de disparo, se ejecuta la función *Number to Boolean Array.* Esta función se ejecuta un número de veces igual a los canales configurados, pero la ejecución es en paralelo. La tarea de esta función es convertir el dato en formato punto fijo a un arreglo *booleano*. El tamaño del arreglo *booleano* será de 32 elementos. Estos 32 elementos representan 32 bits. La función *Number to Boolean Array* identifica cuantos elementos son necesarios para representar el dato en punto fijo. Esta puede asignar: 8, 16, 32 o 64 elementos (bits). El arreglo *booleano* resultante está codificado en complemento a dos. Una vez que se tienen todos los datos como arreglos *booleanos*, se manda a llamar a la función *Boolean Array to Number*. Esta función interpreta a los arreglos booleanos como la representación binaria de un número. Cuando el número es signado, la función sobreentiende que se trata de una representación en complemento a dos. La función *Boolean Array to Number* ha sido configurado para que convierta el arreglo *booleano* a un entero sin signo de 32 bits. La conversión a 32 bits de todos los datos es necesaria

por dos razones. La primera razón es que todos los datos deben tener el mismo formato para que puedan ser transmitidos por DMA. La segunda razón tiene que ver con el hardware para la trasmisión por DMA. El bus por el que viajan los datos es de 32 bits, entonces, si los datos son de 32 bits se hace eficiente la transferencia de datos por DMA.

En este punto de algoritmo, ya se cuentan con los datos codificados en el formato necesario para ser transmitidos al MCU. Se prosigue a agruparlos en un solo bloque. Este bloque es prácticamente un arreglo con un dato por canal habilitado. Para saber que efectivamente se ha adquirido una muestra por canal habilitado, se pregunta a la función *FPGA I/O* si se generó el error 65539. El error 65539 se presenta cuando la función *FPGA I/O* detecta que ha perdido una o más muestras del convertidor (NI 9205). Monitoreando este error, también es posible asegurar que el ciclo de adquisición se ejecuta a una velocidad permitida por el NI 9205. Cuando el error 65539 se presenta, se detiene la adquisición y se retorna a preguntar por el control *Empezar,* figura 3.57.

Transmisión

El proceso de lectura ha concluido y ahora se necesita enviar al MCU el bloque de datos obtenido. Para lograr la transmisión de datos a través de DMA, se hace uso del método *Write FIF0.* A este método hay que especificarle el *buffer* FIFO al que se quiere escribir. La función sólo puede escribir un elemento a la vez. Es por lo anterior que se hace necesario el uso de un ciclo *while*. Éste permitirá recorrer el bloque de datos elemento por elemento, figura 3.58. Otro parámetro indispensable es el tiempo de espera. Debido a que queremos que la transferencia sea lo más rápida posible, se establece un tiempo de espero igual a cero. El método *Write FIFO* es capaz de indicar si no hay espacio disponible en el *buffer* al que se está escribiendo. El indicador que absorbe esta indicación se llama *Buffer Overflow.* Como se había mencionado con anterioridad, este indicador es tipo *booleano.* Cuando el indicador tiene un valor verdadero, la adquisición y la transmisión deben parar ya que el *buffer* se está sobrescribiendo. De lo contrario, la velocidad con la que el FPGA escribe al *buffer* es coherente con la velocidad a la que el MCU está leyéndolo. Si todo marcha sin problemas, el ciclo de transmisión debe parar cuando se han acabado de mandar todos los datos. Después de terminado el ciclo de transferencia, una iteración del ciclo de adquisición de datos está completa. El MCU ya cuenta con una muestra de cada canal configurado y está en espera del resto de los datos.

Este capítulo ha mostrado las implicaciones y el desarrollo del sistema del cual es objeto esta tesis. Para mostrar la puesta en marcha de lo presentado en diagramas de flujo, se prosigue al capítulo 4.

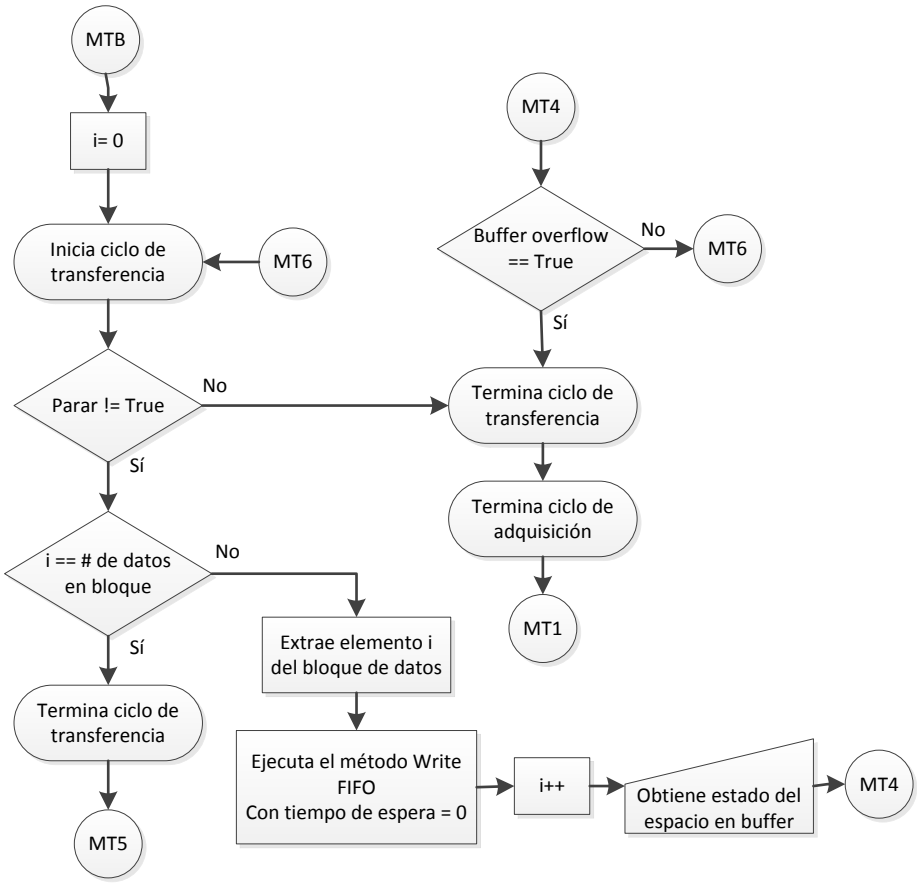


Figura 3.58. Diagrama de flujo para transmisión del VI FPGA.

Capítulo 4
Pruebas del sistema

En este capítulo se muestra el resultado de dos pruebas realizadas al sistema. La primera prueba tiene como fuente de señal a un geófono. En la segunda prueba la fuente es un generador de señales. Para el caso del geófono se aborda desde la configuración de la adquisición y se culmina con el procesamiento de los datos en la PC. En el caso del generador de señales, únicamente se muestra el procesamiento de los datos en la PC. No se muestran pruebas de todos los posibles modos de operación, debido a que eso extendería en demasía el escrito. No obstante, las pruebas aquí presentadas muestran de manera general el desempeño del sistema desarrollado.

4.1. Pruebas con geófono

Las pruebas con el geófono fueron montadas tal y como se muestra en la figura 4.1. Del extremo izquierdo al derecho de la figura tenemos a un generador de señales analógico, posteriormente tenemos un amplificador de potencia, seguido de éste está un geófono montado sobre un excitador dinámico, a continuación se presenta una bifurcación, un enlace va hacia el cRIO y el otro al osciloscopio y por último se tiene que el cRIO está conectado a una PC.

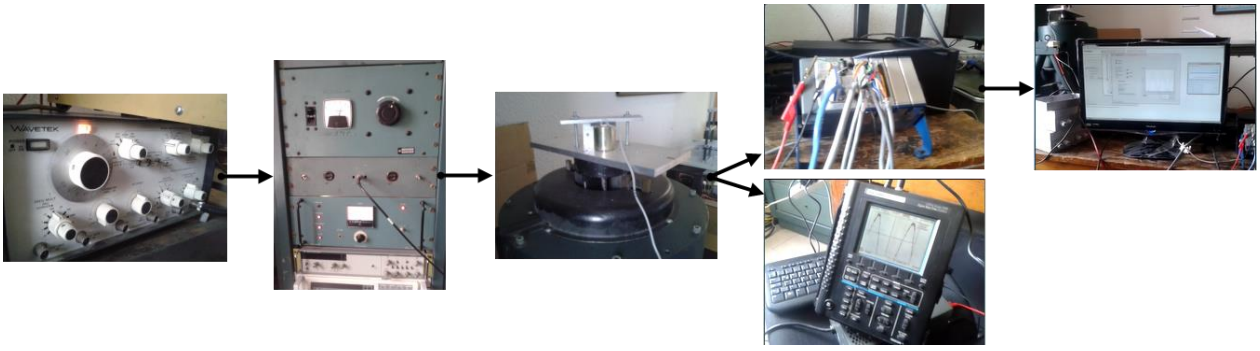


Figura 4.1. Conexiones para prueba con geófono.

El generador de señales fija la frecuencia y la forma de onda de la señal de entrada. El amplificador genera la potencia necesaria para que la señal de entrada pueda ser entregada al excitador dinámico. El eje de éste último se encuentra acoplado a una estructura que fija el geófono al eje. La señal analógica que genera el geófono es mandada tanto al cRIO como al osciloscopio. Para poder ver lo que está llegando al cRIO, éste se encuentra conectado a una PC.

El geófono a utilizar es el GS-11D y en la figura 4.2 se puede apreciar su curva característica de salida. Este es un sensor de tipo electromagnético que tiene la forma de un cilindro metálico, figura 4.3. A grandes rasgos tiene en su interior un imán y una bobina sostenida por un resorte. Este sensor entrega una medición proporcional a la velocidad del movimiento de la superficie que lo sostiene.

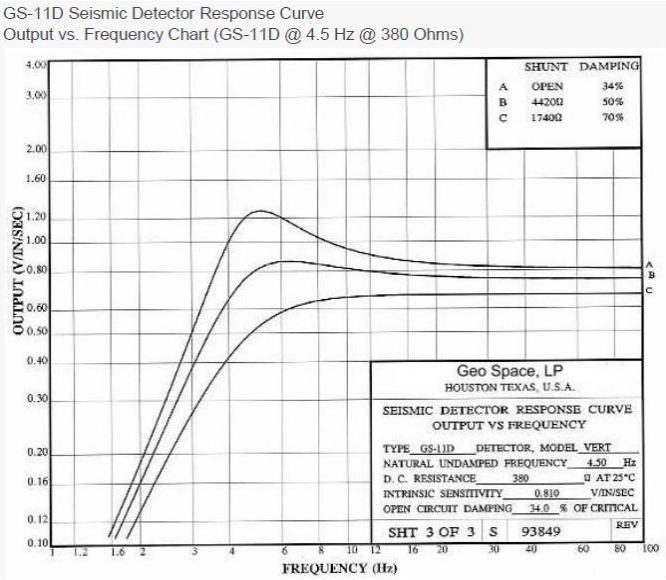


Figura 4.2. Salida vs Frecuencia GS-11D[15].

Figura 4.3. Geófonos GS-11D.

[15] Geospace Technologies, Geophones GS-11D, 2012, 2.

El excitador dinámico es un equipo que se usa en la Coordinación de instrumentación del Instituto de Ingeniería (II) de la UNAM. Este equipo permite simular una vibración en dirección horizontal o vertical. La dirección de la vibración queda determinada por la posición del eje del excitador dinámico, para esta prueba la posición es vertical, figura 4.1.

Debido a que la forma de onda que produce el generador de señales es conocida, no se tendrá problemas en concluir si la señal que se está adquiriendo con el cRIO es correcta o incorrecta. Además, el hecho de tener conectado el osciloscopio brindará mayor confianza en los resultados obtenidos.

4.1.1. Configuración de la prueba

La configuración de la prueba inicia generando el archivo de configuración. Para generar este archivo es necesario acceder a la página web *Principal*, figura 4.4. Para acceder a esta página se necesita teclear la siguiente ruta en el navegador: 169.254.88.15:8000/llamada.html. La ruta contiene la dirección IP del cRIO, el número del puerto de acceso y el nombre del archivo html. Una vez que se despliega la página, se prosigue a dar clic derecho sobre cualquier parte del recuadro gris. Inmediatamente después nos aparecerá un menú en el que se debe seleccionar *Request Control of VI, figura 4.5*. Después de seleccionada esta opción ya es posible tener control sobre el VI. Se prosigue con el programa al seleccionar, en el apartado *Opciones,* la acción a realizar. Se elige la opción denominada *FechaHora*. Esta opción desplegará una nueva ventana con el título *Fecha-Hora [Remote Panel] – Controller,* figura 4.5*.* A través del control *Acción* es posible seleccionar dos opciones diferentes, *Configurar* y *Probar*. La primera opción, como su nombre lo dice, permite hacer la configuración del reloj del sistema. Este reloj es el que permite fijar todas las referencias de tiempo y fecha en el software. Se puede poner como ejemplo el encabezado de los archivos de adquisición. Por otro lado, la opción Probar sirve para verificar la fecha y la hora con la que el reloj del sistema está trabajando.

Una vez configurada la fecha y la hora, se realizará una prueba para verificar que el reloj ha tomado los cambios solicitados. Al seleccionar la opción de Configurar en el control *Acción*, figura 4.6, la ventana se modifica y toma la apariencia de la figura 4.7. Mediante el control 1 se selecciona si se quiere establecer la hora o la fecha. Primero se configurará la hora, para ello en el control 1 se selecciona la opción *Establecer Hora UTC*. Posteriormente, con la ayuda de los controles del 2 al 4, se establece las 20 horas con 28 minutos con 0 segundos. Realizado lo anterior se prosigue a dar clic en el control 5 (*Configurar*). El sistema ha modificado la hora y se prosigue a configurar la fecha. Es necesario volver a elegir en el control *Acción* la opción de *Configurar*. En el control 1 ahora se indica que se quiere establecer la fecha, figura 4.8. A través de los nuevos controles 2, 3 y 4, se fija la fecha con mes 7, día 22 y año 2013. Una vez más se usa el control 5 para establecer la fecha.

Debido a que la configuración ya se ha realizado, se da paso a verificar que la misma haya tenido el efecto solicitado. En el control Acción, figura 4.6, se selecciona la opción de Probar. Automáticamente la ventana genera un nuevo indicador, éste

contiene la información de la fecha y la hora con la que el sistema está trabajando, figura 4.9. Para terminar la prueba se selecciona el control 1 de la figura en descripción. Como se desea abandonar la ventana *Fecha-Hora,* se da clic sobre el control *Parar*, figura 4.9. El control *Parar* siempre está presente en la ventana Fecha-Hora.

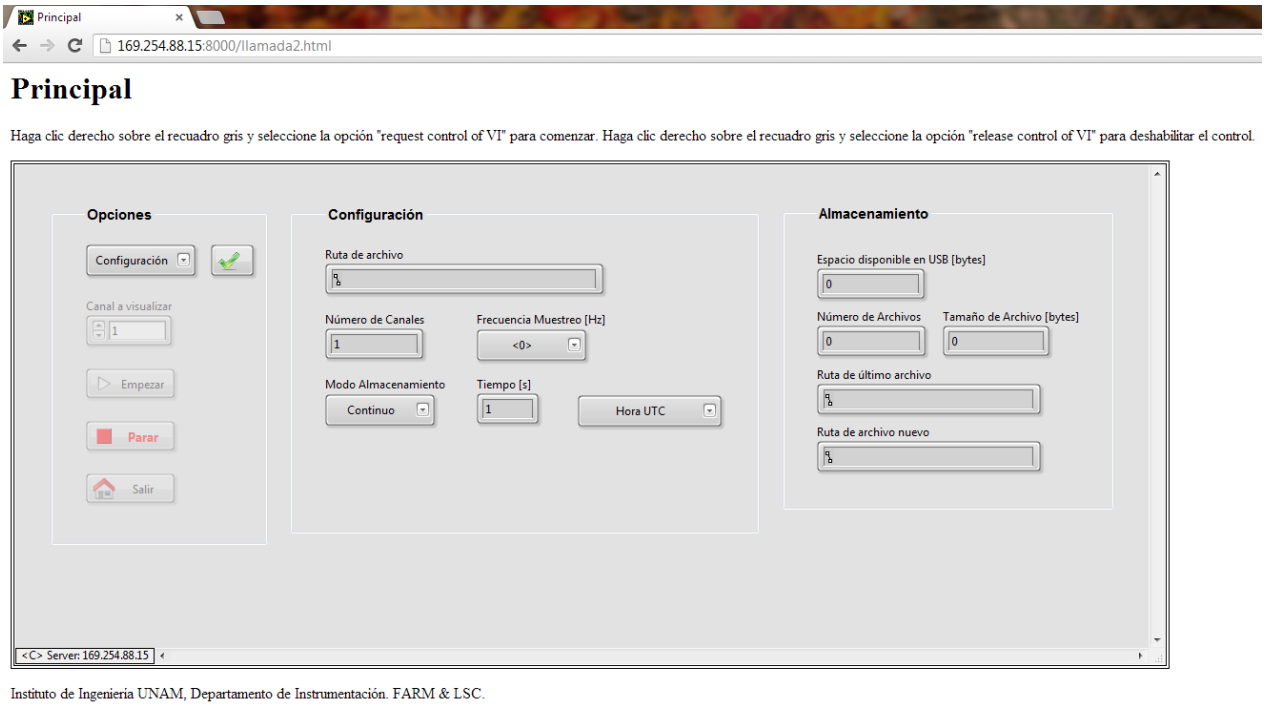


Figura 4.4. Página web Principal.

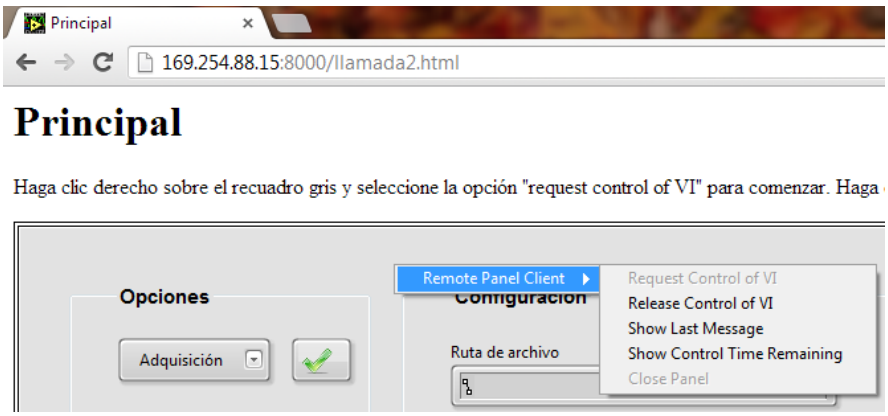


Figura 4.5 Menú de control remoto.

Es importante resaltar que la configuración de la fecha y la hora no es indispensable para realizar una adquisición. A pesar de ello, se aconseja llevar a cabo la configuración a menudo. Un tiempo corto entre una y otra configuración de fecha y hora, mejorará la exactitud del encabezado de tiempo en los archivos.

Después de haber configurado la fecha y hora, se prosigue a llevar a cabo la configuración de una adquisición: modo de medición de los canales, cantidad de canales, mecanismo de almacenamiento, tiempos de almacenamiento, etc. Para ello, en la página web *Principal*, en el apartado *Opciones*, se selecciona la acción de *Configuración*, figura 4.4.

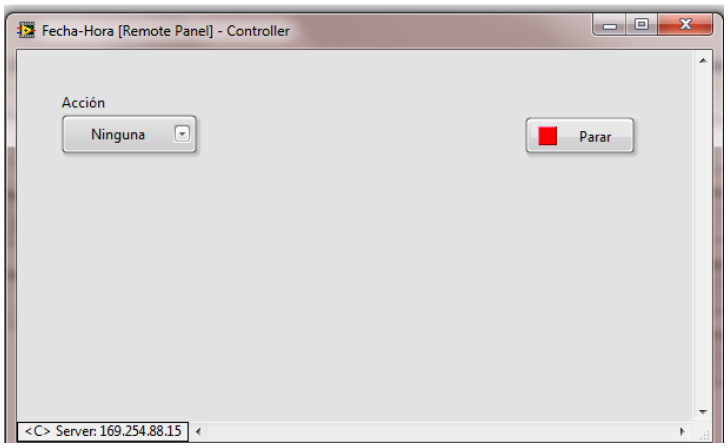


Figura 4.6. Ventana Fecha-Hora.

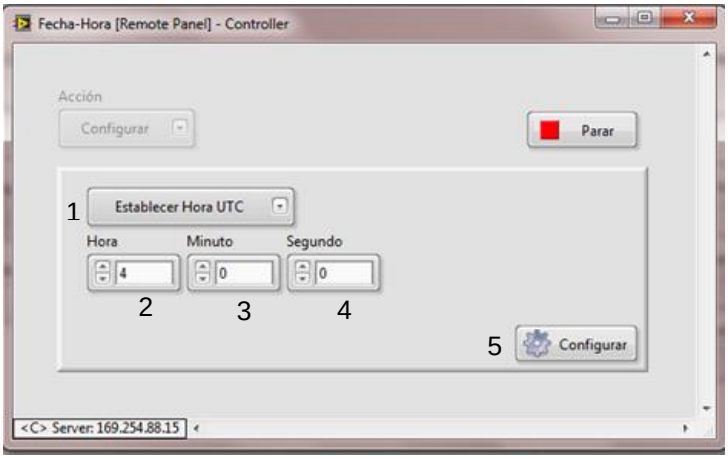


Figura 4.7. Configuración de Hora.

Para empezar a introducir los parámetros necesarios para la adquisición de datos, el sistema despliega una nueva ventana, figura 4.10. Esta ventana lleva el título de *Configuración Canales [Remote Panel] – Controller.* En el control *Nombre de Estación* se introducirá geófono. En el apartado *Tipo de Canal*, en el control *Diferenciales*, se elige la opción de tres *Diferenciales*. Es necesario indicar al sistema que la selección ya ha sido hecha, entonces, se da clic en el control 1. Cabe recordar que una vez que se han elegido canales en modo diferencial, no es posible elegir algún otro modo (NRSE, RSE).

Puesto que se ya se ha seleccionado el control 1 de la figura 4.10, el sistema modifica la apariencia de la ventana y muestra un nuevo apartado llamado *Parámetros*

Canales, figura 4.11. Se pude empezar a operar los controles hasta que los indicadores, *Frecuencias listas* y *Habilitado* se iluminan. El sistema indica la cantidad de canales que deben ser configurados, en este caso 3. El orden de selección de los parámetros no es muy relevante. Se fija la frecuencia de muestreo a 200 Hz, se deja el nombre por default que el sistema genera para los canales, se establece las unidades de Volts, la constante de amplificación se deja en 1 (sin amplificación) y el rango de operación se fija en +/- 1V. Cada que se quiera guardar la configuración de un canal, es necesario seleccionar el control *Guarda*.

Figura 4.8. Configuración de Fecha.

Figura 4.9. Prueba de Fecha y Hora.

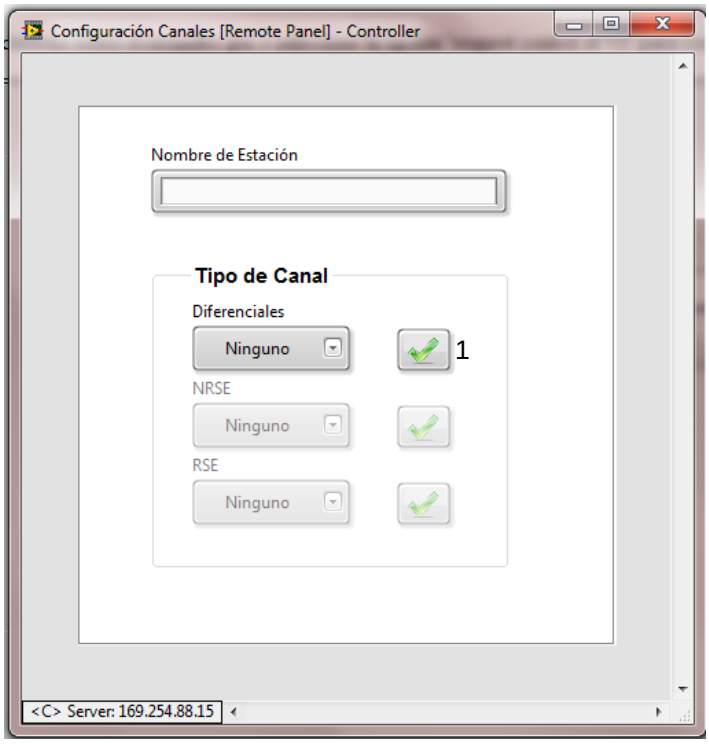


Figura 4.10. Ventana Configuración Canales.

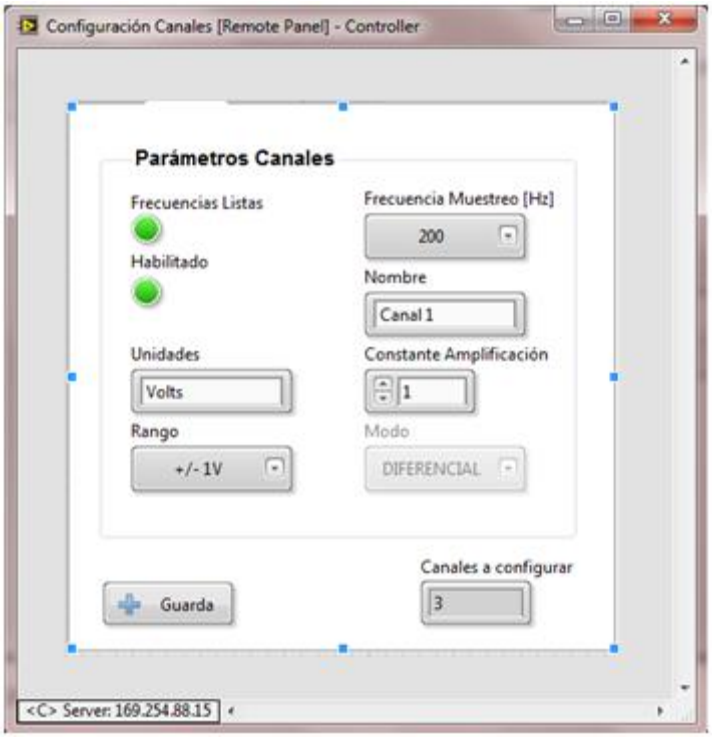


Figura 4.11. Apartado Parámetros de Canales.

En este punto de la configuración ya se ha fijado la forma de operación de los canales analógicos del NI 9205. Es turno de especificar el modo de almacenamiento que se desea. Configurado el último canal, el sistema modifica una vez más la ventana y ahora se ve como en la figura 4.12. Lo que se puede apreciar es el apartado *Configuración Almacenamiento* y sus respectivos controles. Se procede a elegir en el control *Modo Almacenamiento* la opción de *Software*. Se debe indicar a través del control 1, figura 4.12, que el modo de almacenamiento ha sido elegido. Mientras no se dé clic sobre el control 1, no se habilitarán el resto de los controles. Debido a que los controles *Modo Disparo*, *Umbral Superior* y *Umbral Inferior* son para modo Disparo, deben permanecer inhabilitados. De manera similar el control *Tiempo* debe mantenerse deshabilitado. Recordar que en modo *Software* el usuario debe iniciar y detener la adquisición. Es decir, no es necesario temporizar la adquisición. Por otro lado, el sistema fija un límite de 600 segundos para cualquier archivo de adquisición. Se continúa seleccionando el formato de hora que se desea usar, se elige *Hora Local (verano)*. A manera de confirmar que se ha terminado la configuración, se debe dar clic en el control *Configura*.

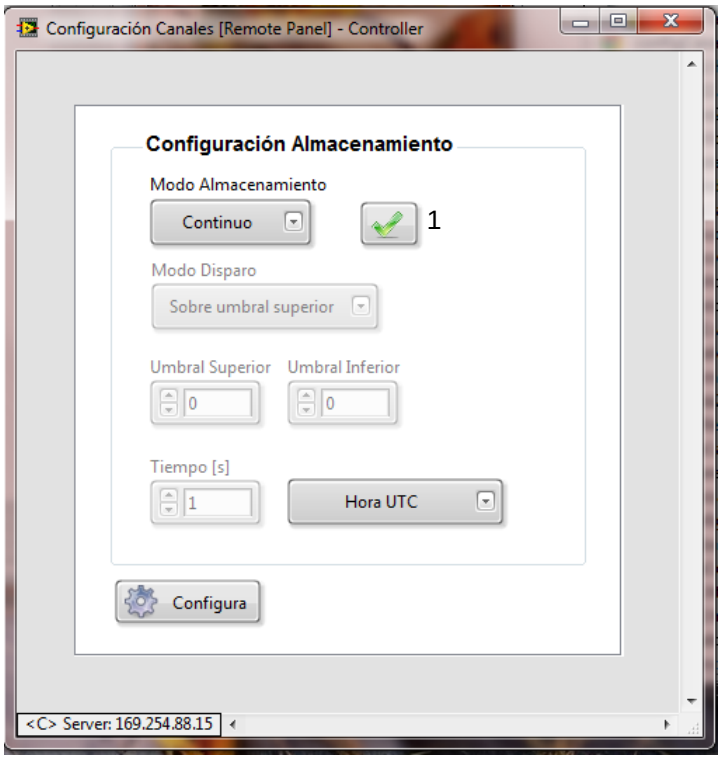


Figura 4.12. Apartado Configuración Almacenamiento.

Una vez que se ha terminado de introducir los parámetros de configuración, necesarios para llevar a cabo una adquisición de datos, el sistema procederá a tratar de generar el archivo de configuración, la generación del archivo dependerá de la correcta operación del software y de que la memoria USB esté insertada de manera correcta. Si todo marcha bien el sistema desplegará un mensaje como el de la figura 4.13. Para terminar la sesión de configuración se da clic en el único control existente. Realizado lo

anterior, el archivo de configuración se encuentra en la memoria USB y listo para ser leído por el MCU.

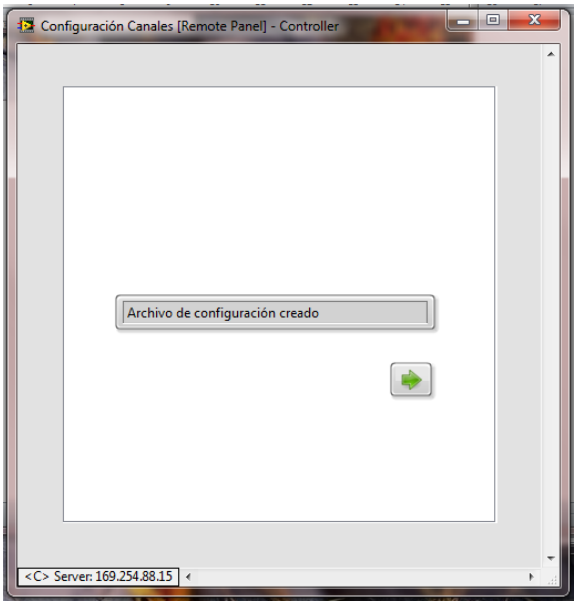


Figura 4.13. Generación de archivo de configuración.

En la figura 4.14 se puede observar la carpeta y el archivo creados. La imagen muestra cómo se verían desde el explorar de Windows Seven. Se debe prestar atención en el nombre que tiene la carpeta contenedora del archivo de configuración. Está formado por el nombre de estación, la fecha y la hora de la creación de la carpeta.

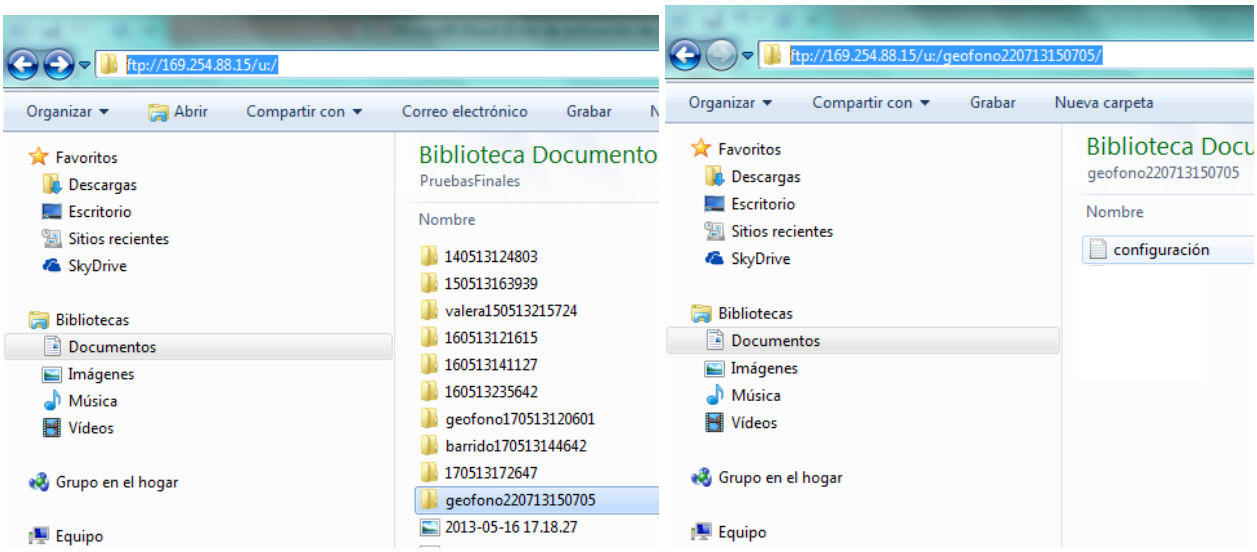


Figura 4.14. Carpeta y archivo de configuración en memoria.

El archivo de configuración tiene texto codificado en ASCII, por lo que prácticamente cualquier procesador de texto puede leerlo. En la Figura 4.15 se ve el

archivo de configuración generado. Este archivo ha sido abierto con el programa Excel 2010 de Microsoft.

	A	B	C	D	E	F	G	H
1	FormatoHora:	Local verano						
2	ModoDisparo:							
3	Tiempo [s]:	600						
4	Umbral inferior:							
5	Umbral superior:							
6	ModoAlmacenamiento:	Software						
7	Canales:	3						
8	Frecuencia de muestreo	Longitud de palabra	Constante de amplificación	Nombre de canal	Unidades	Habilitado	Rango de Voltaje	Modo
9	200	32 bits	1	Canal 1	Volts	TRUE	+/- 1V	DIFERENCIAL
10	200	32 bits	1	Canal 2	Volts	TRUE	+/- 1V	DIFERENCIAL
11	200	32 bits	1	Canal 3	Volts	TRUE	+/- 1V	DIFERENCIAL
12								

Figura 4.15. Contenido de archivo de configuración.

4.1.2. Prueba en marcha

La configuración está realizada y ahora se procede con la adquisición de datos. Para poder comenzar adquirir datos, es necesario remitirse a la página web *Principal*, figura 4.4. En el apartado *Opciones* se elige la acción de *Adquisición*. Posterior a confirmar la tarea a realizar, en el apartado *Configuración* se despliega toda la información del archivo de configuración que se acaba de crear. Paralelamente, en el apartado de *Almacenamiento* se despliega la siguiente información: el espacio útil que queda en la memoria, el número de archivos que se pueden generar con la configuración actual, el tamaño de un archivo y las rutas de los archivos último y actual. Una vez sucedido lo anterior se habilita el control *Canal a Visualizar*, figura 4.4. En este control es posible seleccionar el canal que se desea observar en tiempo real. Evidentemente, el número de canal tiene que corresponder con uno de los tres canales que han sido configurados. Se opta por seleccionar el canal 1. Elegido el canal de monitoreo, sólo queda dar clic en el control *Empezar*. En la Figura 4.16 se puede ver la apariencia que toma la página web *Principal*.

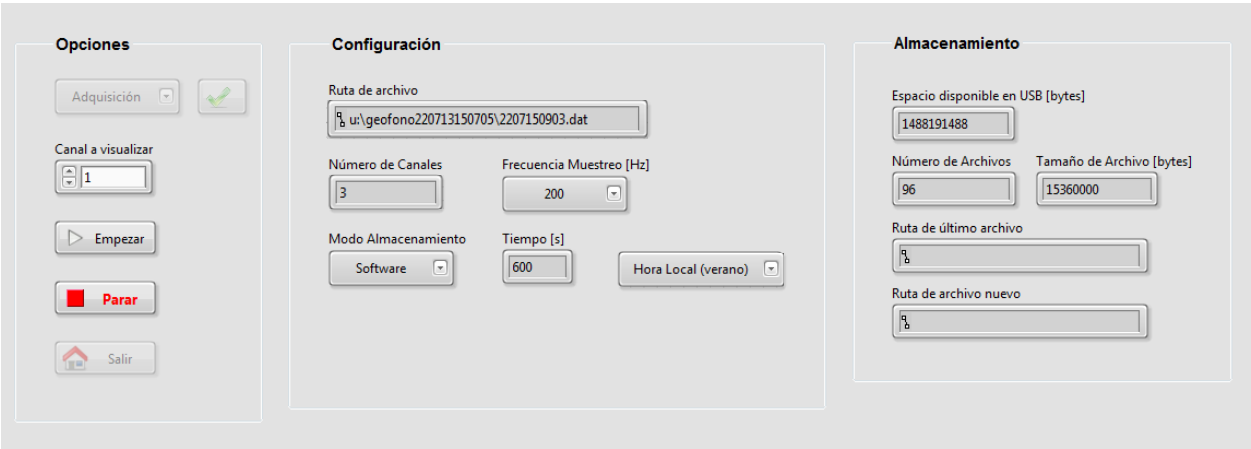


Figura 4.16. Página principal en adquisición.

Al abrir la ventana de adquisición en modo software (página web), figura 4.17, el sistema ya se encontrará adquiriendo datos y desplegándolos a través de una gráfica. Paralelamente, el sistema bloqueará el control *Empezar* de la página web Principal. Debido a que el modo de adquisición es software, el sistema permanecerá desplegando el canal 1 hasta que se dé clic en el control *Grabar*. Al no haber todavía ningún archivo creado se ven algunos indicadores vacíos. Es posible apreciar que se están adquiriendo 600 muestras por segundo y que el canal 1 es el que se está desplegando. La gráfica cuenta con varias herramientas sumamente intuitivas. A través de estas herramientas es posible manipular la gráfica de muchas maneras. Ejemplo de estas herramientas son: *zoom in*, *zoom out*, amplificación por área, auto escala (vertical, horizontal o ambas), escala fija (vertical, horizontal o ambas), número de líneas de la cuadricula, color de líneas, ancho del trazo, *offset*, precisión de las referencias (amplitud), entre otras.

Cabe comentar que la ventana de la figura 4.17 tiene mucho parecido con las ventanas correspondientes a los demás modos de adquisición. Los controles e indicadores prácticamente son los mismos en todos los casos.

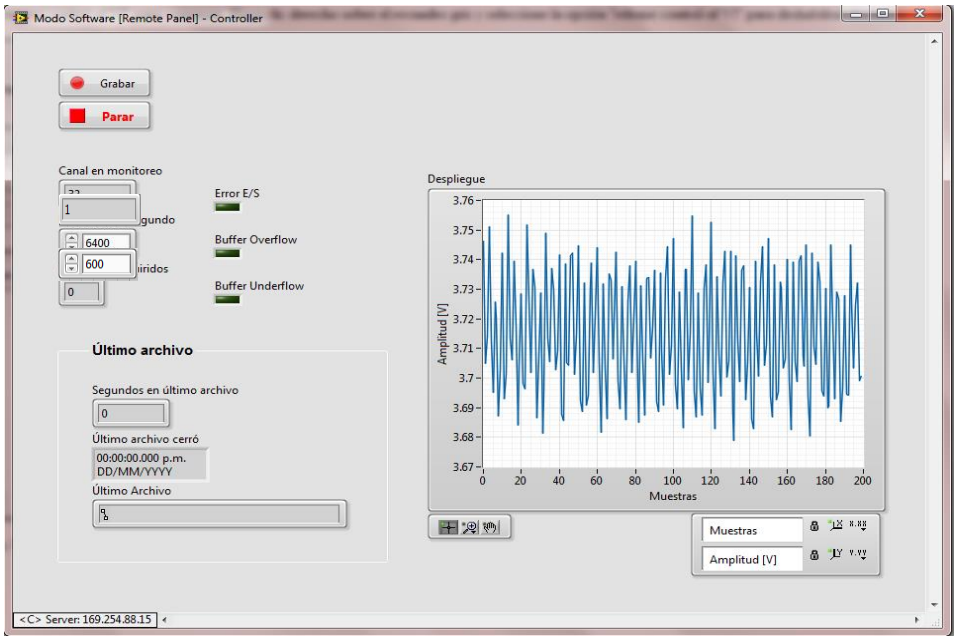


Figura 4.17. Ventana Modo Software.

Sistema y Osciloscopio

Para tener una referencia de la operación del sistema, un osciloscopio y el sistema han sido puestos a medir la señal de salida del geófono. El generador que fija la forma de onda con la que se mueve el excitador dinámico, ha sido configurado para entregar una forma de onda sinusoidal. En la figura 4.18 se presenta la foto que corresponde al monitoreo de un osciloscopio Tektronix, modelo THS3000. Para este caso se tiene una configuración de 100 mV por división y 50 ms por división. Se observa que la señal tiene un período de 350 ms o lo que es igual a 2.8571 Hz. En lo

referente a la amplitud, se puede ver que en el osciloscopio se complica la medición. Se aprecia una señal un tanto ruidosa. Por otro lado, en la figura 4.19 se presenta la foto que corresponde a la gráfica de la ventana de modo Software. En este caso se tiene que una cuadricula completa equivale a 1 segundo. En cada segundo se despliegan 200 muestras, entonces, una muestra (eje horizontal) corresponde a 5 ms. Haciendo una inspección se puede ver que entre picos máximos existen 70 muestras, lo que equivale a tener 350 ms. El período calculado de la imagen del osciloscopio coincide con el período calculado en la imagen derivada del modo Software del sistema. Al igual que para la obtención del periodo, resulta sencillo identificar la amplitud de la señal, máximo en 0.8 V y mínimo en 0.75 V. Un apunte interesante con respecto a la medición de la amplitud es que las crestas de la señal del sistema son mucho más limpias que las del osciloscopio.

Almacenamiento de los datos

Se van a generar cuatro archivos binarios diferentes. El primero en generarse será el de menor cantidad de información (menor tiempo). Los dos siguientes contendrán una cantidad de información idéntica; la cantidad de información será más grande que la del primero. El cuarto y último archivo será el de mayor cantidad de información. La idea de generar archivos con tiempos diferentes, permitirá mostrar la capacidad del sistema para trabar con archivos de diferentes tamaños. Se obtendrá un archivo con 30 segundos de adquisición de datos, dos archivos con 60 segundos y uno de 600 segundos. La adquisición de datos no es temporizada, es modo software, no obstante, en la ventana de adquisición de datos se despliega la cantidad de segundos adquiridos, entonces, será este indicador el que permitirá tener certidumbre sobre la duración de las pruebas.

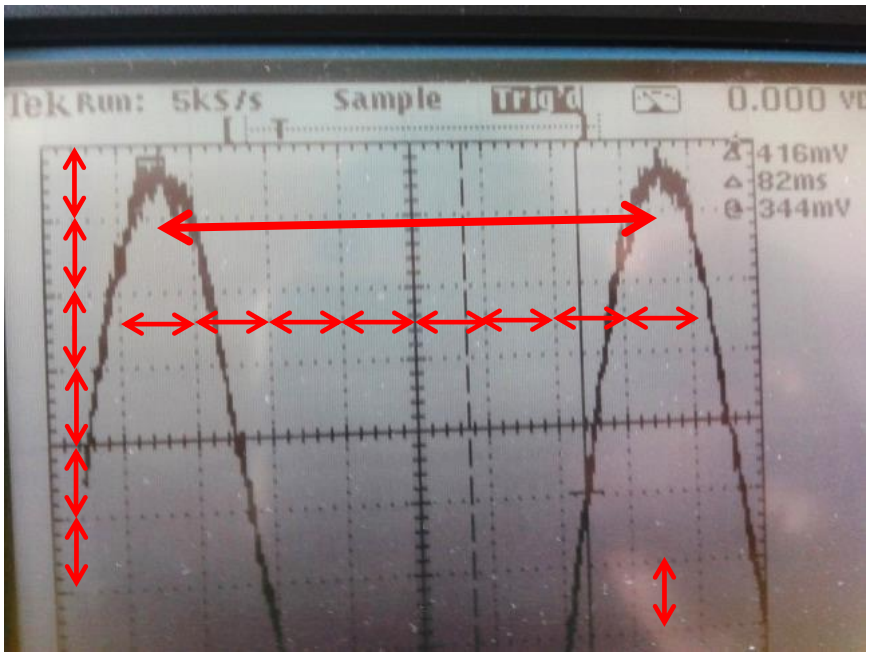


Figura 4.18. Osciloscopio en monitoreo.

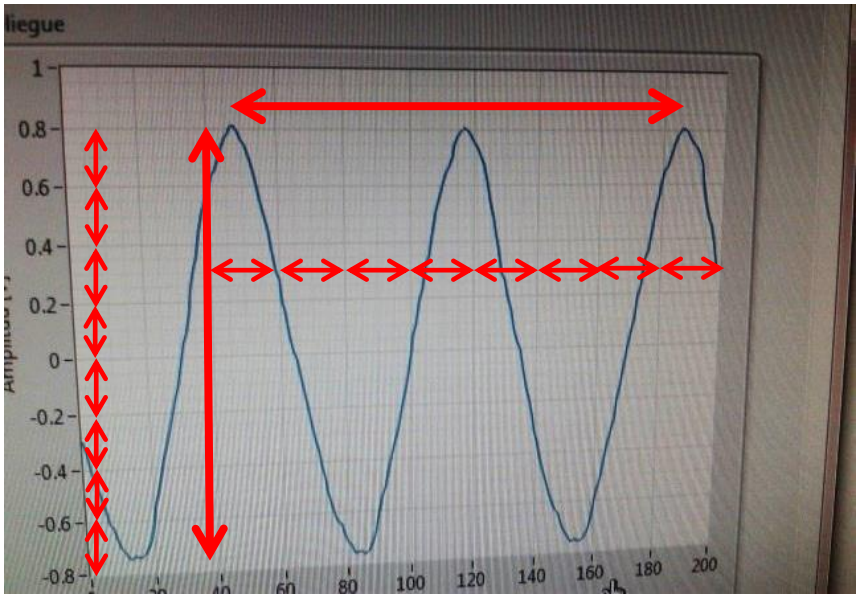


Figura 4.19. Sistema en monitoreo.

Al estar adquiriendo en modo software, el usuario es quien inicia y detiene la adquisición. El inicio se hace mediante el control *Grabar* y el paro mediante el control *Parar*, figura 4.17. Se procede a presionar el control *Grabar*, automáticamente el sistema deshabilita este control, detiene el despliegue del canal 1 e inicia el contador de segundos adquiridos, figura 4.20. Una adquisición ha comenzado.

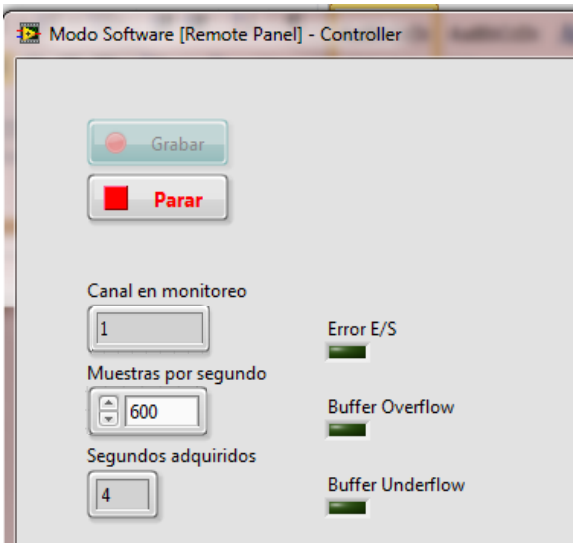


Figura 4.20. Adquisición en progreso.

Tal y como se aprecia en la figura 4.20, los indicadores *Error E/S, Buffer Overflow y Buffer Underflow*, deben mantenerse sin brillar mientras todo el proceso de adquisición esté marchando de manera correcta. Para la generación del primer archivo mencionado, a los 30 segundos de adquisición de datos, con la ayuda del control *Parar,*

se detiene el almacenamiento de datos. La ventana de captura en modo software reiniciará, el control *Grabar* volverá a estar habilitado, el indicador de *segundos adquiridos* se pondrá en cero y el apartado *Último archivo* desplegará información sobre el último archivo creado. Esta información consiste en el número de segundos que se almacenó en el archivo anterior, la hora en la que se terminó de almacenar en el archivo y la ubicación del archivo en la memoria, figura 4.21.

Para la generación de los otros tres archivos se procede bajo el mismo mecanismo mencionado. Los dos archivos siguientes son de 60 segundos y el último es de 600 segundos. Al tener los cuatro archivos guardados en la memoria USB, el contenido de la carpeta creada tiene que verse como en la figura 4.22. Esta carpeta ya contenía el archivo de configuración, ahora se han agregado los cuatro archivos de adquisición de datos.

Figura 4.21. Adquisición realizada.

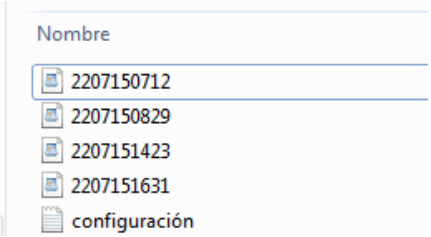


Figura 4.22. Archivos de adquisición en memoria.

El control *Parar* indica al sistema que el almacenamiento ha terminado, pero no le dice que deje de adquirir. Es decir, después de presionado el control *Parar* se deja de almacenar pero se sigue adquiriendo información para ser desplegada. Si lo que desea

el usuario es cerrar la sesión de adquisición, primero, en la página web *Principal*, figura 4.16, debe seleccionar el control *Parar*. Ya que este control haya sido seleccionado, en la ventana de Modo software, figura 4.21, debe presionarse el control *Parar*. Con esto la sesión de adquisición se cierra y la única ventana activa es la *Principal*. Es necesario, si se quiere realizar otra tarea con el sistema, que en esta misma ventana, figura 4.16, se presione el control *Salir*. Seguido de haber presionado *Salir*, el navegador desplegará un mensaje en el que indique que no encuentra el VI a desplegar. El mensaje dirá *Remote panel connection is closed*. Aunque el usuario trate de refrescar el navegador para volver cargar el VI, éste no se cargará hasta que el cRIO se haya reiniciado de manera adecuada. Por seguridad, cada que una sesión de adquisición ha terminado, el sistema (cRIO) se reinicia.

4.1.3. Procesamiento con software de PC

A manera de ilustrar el funcionamiento de la transferencia de archivos por FTP, se supondrá que el usuario quiere transferir el archivo por Ethernet. Evidentemente, el usuario puede extraer la memoria del cRIO 9012 y conectarla a una PC para ver su contenido. Una vez que se tiene acceso al archivo, los procesos de lectura y conversión son invariables.

Transporte de archivos por FTP

Cuando se requiere hacer la transferencia FTP es necesario usar el software para la PC. Para contar con este software es necesario hacer la instalación pertinente. Para los fines que persigue este capítulo, se considera que la instalación ya ha sido realizada. El acceso al programa es como el acceso a cualquier otro programa comercial en Windows, se accede mediante un ejecutable. El ejecutable se llama PC5 y tiene un icono como el de la figura 4.23.

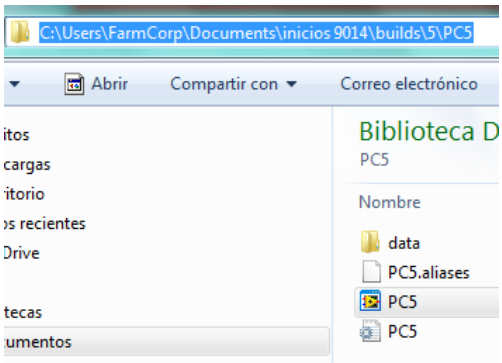


Figura 4.23. Ejecutable de software para PC.

Se accede a la aplicación haciendo doble clic sobre el icono PC5. Cuando la aplicación inicia, se despliega una ventana llamada Menú, figura 4.24. En la ventana se pueden apreciar el indicador *Activa*, el control *Procesamiento*, el control *FTP* y el control *Salir*. *Activa* se encarga de indicar si existe algún proceso en ejecución o en su defecto si ninguno está operando. *Procesamiento* permite abrir la ventana para procesar la

información. *FTP* despliega una nueva ventana que permitirá transferir los archivos. Por último, *Salir* permite cerrar la ventana Menú y terminar la ejecución del software.

Se comienza el proceso para transferir el archivo. En la ventana Menú se da clic en el control *FTP*. Inmediatamente se despliega la ventana FTP, figura 4.25. Se hace la suposición de que no se conoce con exactitud la organización de la memoria USB, entonces, se explora primero el directorio raíz. Para lograr lo anterior, en el control *Acción* se selecciona *Obtener directorio*. Se da paso a introducir la dirección IP del cRIO y la ruta del directorio raíz. La primera es 169.254.88.15 y se introduce en el control *IP cRIO*, la segunda es u:\ y se introduce en el control *Ruta remota*. Proporcionados los parámetros se da clic en el control *Empezar*. Si todo ha salido de manera correcta, el servidor FTP contestará con la información de la ruta especificada e indicará que todo ha salido bien, figura 4.26. La respuesta del servidor (cRIO) se reflejará a través de los indicadores *Contenido de la ruta* y *Respuesta*.

Figura 4.24. Ventana Menú.

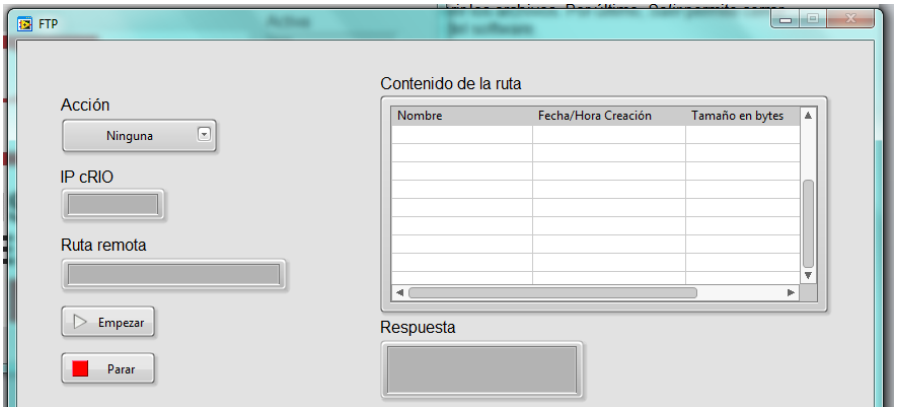


Figura 4.25. Ventana FTP.

Como ya se sabía, en la raíz principal de la memoria USB se encuentra la carpeta geofono220713150705. En esta carpeta se encuentra contenida toda la

información de la adquisición realizada. Se repite el proceso para verificar el contenido de una ruta, pero ahora para ver los archivos de la carpeta de interés. Con fines de lograr lo anterior, en *Ruta remota*, figura 4.25, se fija la ruta u:\ geofono220713150705\. La respuesta del servidor FTP es obvia. Se procede entonces a transferir el archivo de interés.

Para lograr transferir un archivo, en el control *Acción*, figura 4.25, se selecciona la opción de *Obtener Archivo.* Los parámetros a establecer son los mismos antes mencionados. Para este caso, la ruta debe ser u:\ geofono220713150705\nombre.dat. Es decir, se debe agregar el nombre y la extensión del archivo que se desea transferir. Ya que se ha dado clic al control *Empezar*, el software despliega un cuadro de diálogo para elegir la ubicación a donde el archivo se va a transferir, figura 4.27. Por default el software crea una carpeta para transferencias, la ruta de esta carpeta es c:\dataADC\. Esta ruta es la que aparece por default en el cuadro de diálogo. Se puede optar por cualquier nombre para el archivo, pero el software propone uno por default. El nombre por default es el mismo que tiene actualmente (2207150712), figura 4.27. Elegida la ruta de interés y el nuevo nombre, el software realiza la transferencia solicitada.

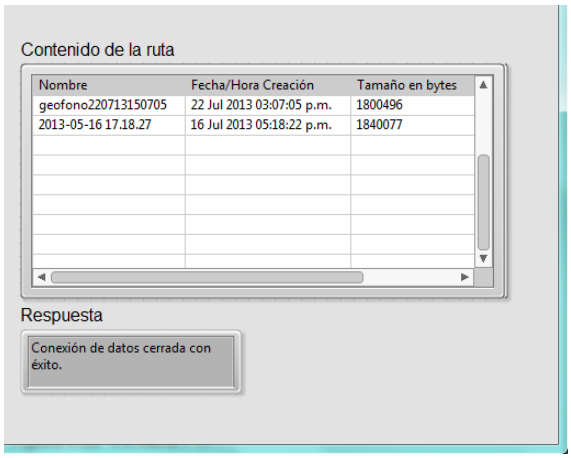


Figura 4.26. Respuesta servidor FTP.

Contando al archivo de configuración, se deben transferir cinco archivos. Debido a lo anterior, se repite el proceso de transferencia cuatro veces más. Después del quinto archivo transferido, en el cuadro de diálogo que aparece al final de cada proceso, se indica al sistema que se quiere abandonar la ventana FTP. La única ventana actualmente abierta será la ventana Menú.

Cabe hacer notar que la transferencia FTP puede tener detalles adicionales a la operación descrita. Por ejemplo, si la PC con la que se está trabajado tiene Firewall activado, este podría impedir que la comunicación se lleve a cabo de manera adecuada. Es muy importante que la PC esté configurada para permitir FTP sin problemas.

Lectura de archivos

Ya se tiene todos los archivos necesarios en el disco duro de la PC. Es turno de dar lectura a los archivos binarios transferidos. Para poder leer los archivos, se procede a hacer clic en el control *Procesamiento* de la ventana Menú, figura 4.24. Una nueva ventana tiene que ser desplegada. Inmediatamente después del despliegue de la ventana, se abre un cuadro diálogo que solicita la elección del archivo de configuración, figura 4.28. Se procede a ubicar la carpeta en donde se localizan los archivos transferidos. El cuadro de diálogo automáticamente filtra los archivos por extensión. Únicamente se pueden elegir los archivos con extensión .txt, con lo que la búsqueda se facilita de manera importante.

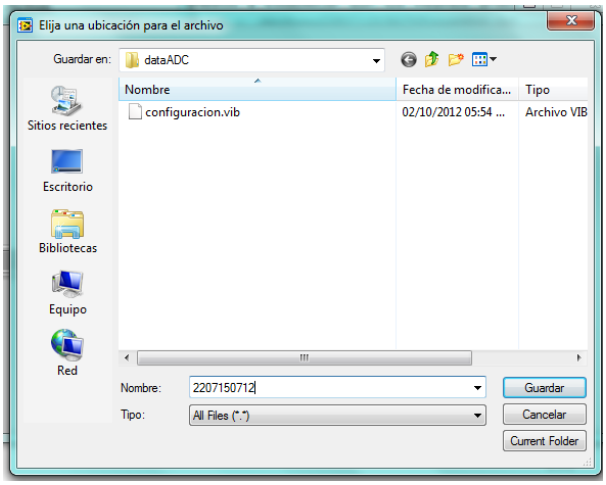


Figura 4.27. Selección de la ubicación de archivo.

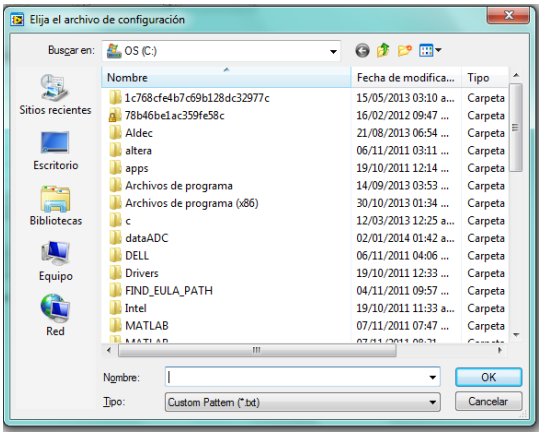


Figura 4.28. Selección de archivo de configuración.

Cuando el archivo de configuración es elegido, el software actualiza cuatro indicadores de la ventana Procesamiento, figura 4.29. Estos cuatro indicadores son el *# de Canales*, *Máx [s]*, *Constante* y *Almacenamiento*. El primer indicador muestra que 3 canales fueron configurados. El segundo indicador informa que a lo mucho existen 600

segundos en cada archivo. El tercer indicador muestra que no existe constante de amplificación. El último indicador informa que el modo de almacenamiento fue Software.

A través del control 1, figura 4.29, se le indica al programa que se quiere leer por partes un archivo. El programa despliega un nuevo cuadro de diálogo, en éste se pide que se elija el archivo a procesar. El programa identifica la ruta del archivo de configuración y por default se posiciona en esa ruta. Además, el cuadro de diálogo aplica un filtro a los archivos. En el cuadro de diálogo únicamente se pueden ver los archivos con extensión .dat. Se procesará los archivos en orden cronológico, entonces, se elige el archivo que se creó primero. Al elegir el archivo deseado, el programa hace un cálculo del número de segundos que realmente contiene el archivo y modifica el indicador *Máx [s]*. Adicionalmente, se habilita el control *Offset [s]*, el control *Segundos, el* control *Empezar* y el control *Selecciona 4 canales*. También, se muestra si la cantidad de datos a leer es permitida, la cantidad de datos solicitada, la ruta del archivo que se está procesando y el tamaño del mismo. En la figura 4.30 se muestra la nueva apariencia de la ventana Procesamiento.

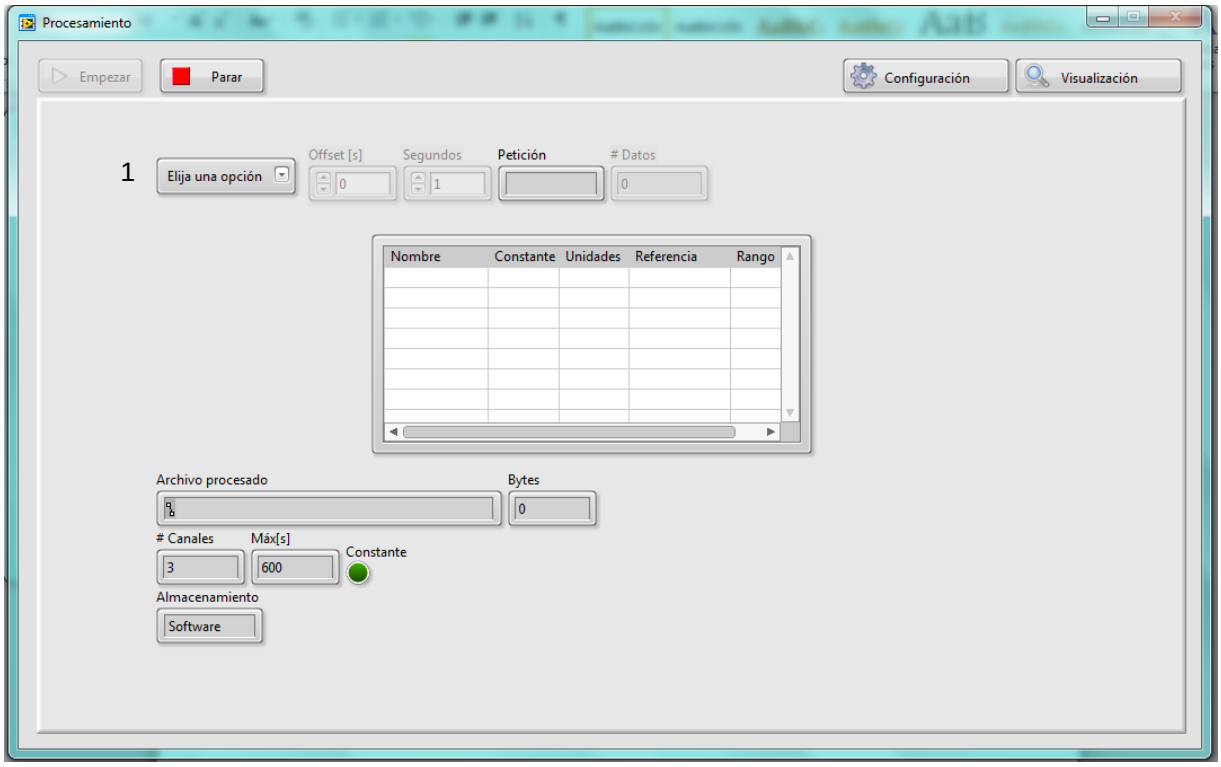


Figura 4.29. Ventana Procesamiento.

Los controles *Offset [s]* y *Segundos* se han habilitado porque sólo se quiere leer una parte del archivo. Cada que estos dos controles se modifiquen, los indicadores *Petición* y *# Datos* se modificarán. El primer indicador mostrará si la cantidad de datos que se quiere leer es posible o no. El segundo indicador desplegará la cantidad de datos que se están tratando de leer. Por ejemplo, en la figura 4.30, el control *Segundos* tiene el dato de 1, esto implica leer 600 datos y es una petición permitida.

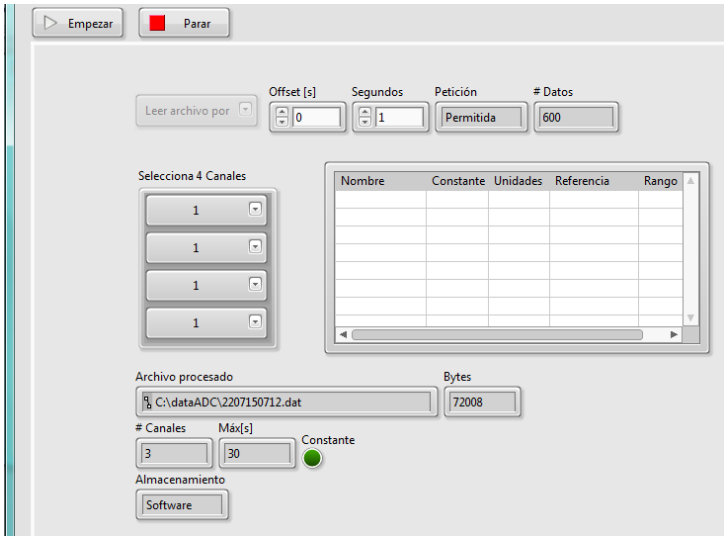


Figura 4.30. Datos de archivo a procesar.

Se quiere leer del segundo 10 al 20, para ello en el control *Offset [s]* se introduce el valor de 10 y en el control *Segundos* también el valor de 10. Además se busca leer los tres canales almacenados, entonces, en el control *Selecciona 4 Canales*, seleccionamos canal 1, canal 2, y canal 3. El control permite seleccionar cuatro canales, se deja la cuarta opción como canal 1. Realizado lo anterior se prosigue dar clic en el control *Empezar*. El programa desplegará el mensaje de la figura 4.31. Para poder ver los resultados de la lectura, se selecciona la opción de *Permanecer*. El mensaje desparecerá y en la ventana Procesamiento aparecerán cinco datos de los canales solicitados. Estos datos son: el nombre del canal, la constante de amplificación, las unidades, el modo de medición y el rango de medición, figura 4.32.

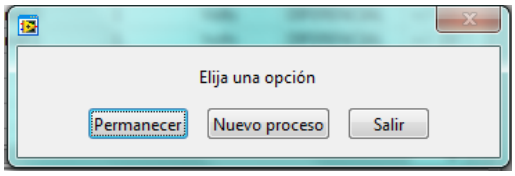
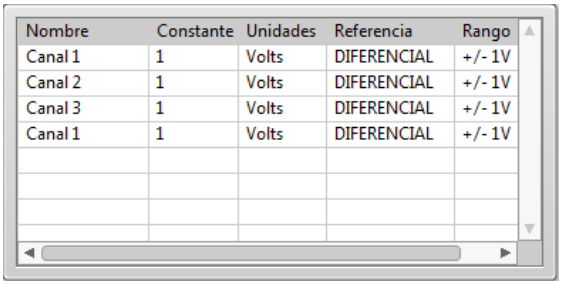


Figura 4.31. Mensaje pos lectura.



Nombre	Constante	Unidades	Referencia	Rango
Canal 1	1	Volts	DIFERENCIAL	+/- 1V
Canal 2	1	Volts	DIFERENCIAL	+/- 1V
Canal 3	1	Volts	DIFERENCIAL	+/- 1V
Canal 1	1	Volts	DIFERENCIAL	+/- 1V

Figura 4.32. Información de canales seleccionados.

Para ver la información que se ha cargado en memoria, en la ventana Procesamiento seleccionamos el control *Visualización*. Esta acción cambiará la apariencia de la ventana y se tendrá que ver como en la figura 4.33. A continuación se describen los elementos presentes en la figura mencionada.

- *t0:* es un indicador que despliega el tiempo y la fecha en la que se inició el almacenamiento en el archivo que se está procesando. Este tiempo es la referencia para los tiempos que se despliegan en el indicador en forma de tabla.
- *Frecuencia Muestreo [Hz]*: es un indicador que presenta la frecuencia a la que fueron muestreadas las señales. El valor está dado en Hertz.
- *dt*: es un indicador que despliega el incremento de tiempo entre muestras consecutivas.
- *Gráfica*: es un indicador que muestra el gráfico correspondiente a las señales muestreadas. Este indicador tiene tres controles asociados que se muestran con mayor detalle en la figura 4.34.
- El control llamado *A* en la figura 4.34, asocia un color de gráfica a cada canal que haya sido seleccionado para ser desplegado, además permite entre otras cosas: habilitar e inhabilitar el despliegue de un canal, elegir el color, el estilo y el ancho de la línea y copiar los datos a Microsoft Excel o al Clipboard de Windows.
- El control llamado *B* en la figura 4.34, permite entre otras cosas: que la gráfica tenga un ajuste automático en el rango de valores de ambos ejes (amplitud y tiempo), que el usuario fije el rango de valores en ambos ejes, que la escala en los ejes sea visible o invisible, el color de la cuadricula, el formato y la precisión de los datos.
- El control llamado *C* en la figura 4.34, permite entre otras cosas: hacer uso del ratón para desplazarse a lo largo y ancho de la gráfica, modificar el tamaño de la gráfica, seleccionar sólo una parte de la misma y restablecer el tamaño por default de la gráfica para deshacer los cambios realizados.
- *Tabla*: es un indicador en donde a todas las amplitudes (en Volts) se les asocia un tiempo relativo (en segundos). Debido a lo anterior todas las muestras quedan referenciadas en el tiempo. La información que se despliega en la gráfica es la misma que se despliega en la tabla.

En caso de que se quiera volver a ver la tabla donde se muestra la configuración de los canales, se debe presionar el control *Configuración*, figura 4.33. Si lo que se desea es abandonar la ventana Procesamiento, es necesario dar clic en el control *Parar*, figura 4.33.

Es importante hacer notar que la información almacenada, figura 4.33, es coherente con la señal que se encontraba generando el geófono, ya que la señal inyectada con el excitador dinámico es una señal senoidal.

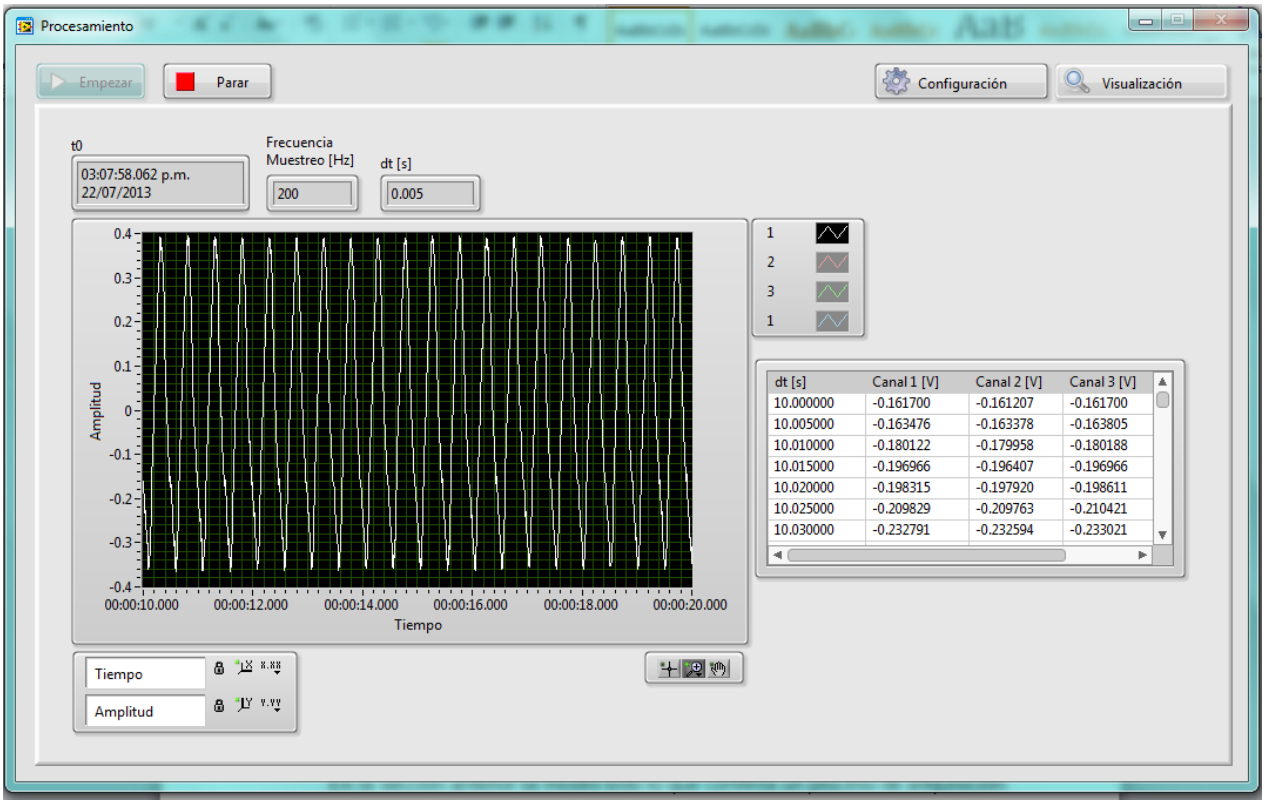


Figura 4.33. Visualización de datos.

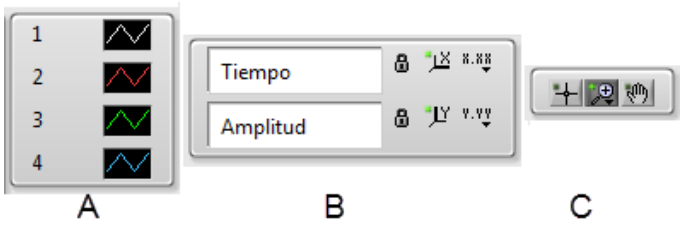


Figura 4.34. Controles del despliegue gráfico.

Conversión de binario a ASCII

Los archivos derivados de la adquisición contienen información codificada en binario. El proceso de visualización, mostrado anteriormente, permite manipular los datos en binario, pero se hace forzoso el uso del programa desarrollado en LabVIEW. Por otro lado, aunque la información puede ser exportada desde la gráfica a Excel 2010, no se pude asegura que esto se pueda hacer para cualquier versión de Excel. Inclusive, la exportación dependerá de la versión del sistema operativo. Para evitar dependencias de software, se mostrará el proceso para generar un archivo con datos codificados en ASCII.

Previamente elegido el archivo de configuración, en el control 1 de la ventana Procesamiento, se selecciona la opción de *Generar archivo .txt*, figura 4.29. Al

seleccionar esta opción se habilita un cuadro de diálogo. En este cuadro de diálogo hay que elegir el archivo que se desea procesar. Elegido el archivo que se desea procesar, un nuevo cuadro de diálogo es desplegado, figura 4.35. En este cuadro de diálogo se puede optar por convertir el archivo completo o sólo una fracción del mismo. Se selecciona la opción para convertir sólo una fracción del archivo. Cabe aclarar que ambas opciones de conversión sólo aplican sobre un canal. Es decir, se convierte toda la información referente a un canal o se convierte una fracción de la información referente a un canal.

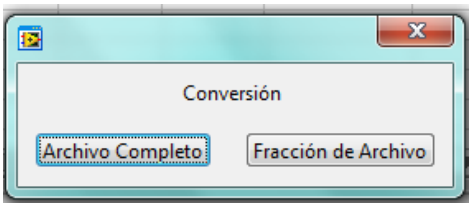


Figura 4.35. Conversión parcial o total del archivo.

Ya que se ha seleccionado la opción de conversión, el programa habilita los mismos controles e indicadores como en el caso de la lectura de un archivo. La única diferencia radica en que el control para seleccionar los canales a procesar, únicamente permite seleccionar un canal. Se opta por convertir una fracción de la información del canal 1. Tal y como se hizo para lectura, una vez seleccionado el canal, se da paso a establecer el *offset* y la cantidad de segundos que se quiere convertir. Se fija un *offset* nulo y el tiempo de un segundo. Posteriormente, se da clic en el control *Empezar.* Un nuevo cuadro de diálogo aparecerá y preguntará por la ubicación y el nombre que se le quiere dar al archivo a crear. Se elige el nombre de conversión.txt. Estrictamente sólo se necesita el nombre del archivo, no es necesario asignar una extensión. Esto queda a criterio del usuario. Cuando la conversión haya terminado, el programa mandará el mensaje de la figura 4.31.

A manera de verifica el contenido del archivo recién creado, se procede a abrir el mismo con la ayuda del software WordPad. Este software viene preinstalado en el sistema operativo Windows Seven (cualquier versión). En la figura 4.36 se puede apreciar la lectura realizada al archivo recién creado. En este archivo es posible identificar a grandes rasgos dos partes: la información general del archivo y los datos de la señal digitalizada.

Con respecto a la información general del archivo se tiene: la ruta en donde se encuentra el archivo procesado, la fecha en la que la adquisición fue realizada, la hora en la que inicio a almacenar el archivo, la frecuencia de muestreo usada, el tiempo que existe entre dos muestras consecutivas, nombre del canal, constante de amplificación, las unidades, el modo de operación y el rango de medición.

En relación a los dato de la señal, se tiene la muestra adquirida y un tiempo asociado. Se debe recordar que el tiempo es relativo y su referencia es el t0 que aparece en la tercera línea. Tanto el formato del tiempo como el formato de la amplitud, están programados para considerar todos los casos que se puedan presentar. Por

ejemplo, la frecuencia máxima de muestreo es 125,000 Hz, esto implica un incremento de tiempo de 0.000008 segundos. Otro caso podría ser cuando se tiene un rango de operación de +/- 200 mV, la resolución sería de 6.57 uV (según fabricante).

En este punto ya se ha realizado un proceso completo de adquisición. Se ha ido mostrando la manera de proceder de principio a fin, pero se quiere dejar en claro que el orden seguido no es riguroso. No obstante, queda por analizar lo que tienen los otros tres archivos generados. Para éstos, sólo se mostrará la gráfica correspondiente a los tres canales.

```
C:\dataADC\2207150712.dat
Fecha:     22-07-13
t0:   03:07:58.062
Frecuencia:      200   [Hz]
dt:   0.005000   [s]
Nombre     Constante  Unidades   Modo            Rango
Canal 1          1    Volts      DIFERENCIAL     +/- 1V

tiempo [s] amplitud [V]
0.000000   0.080160141
0.005000   0.064139366
0.010000   0.046243191
0.015000   0.013773441
0.020000   0.002555370
0.025000   -0.007149220
0.030000   -0.027150631
0.035000   -0.038861752
0.040000   -0.056988239
0.045000   -0.068403721
0.050000   -0.082286358
0.055000   -0.091036797
0.060000   -0.105972290
0.065000   -0.128901482
0.070000   -0.137882710
0.075000   -0.151502132
0.080000   -0.163969994
0.085000   -0.161206722
0.090000   -0.165187359
```

Figura 4.36. Archivo con codificación ASCII.

En la figura 4.37 se puede ver con 3 colores distintos, las gráficas correspondientes a toda la información contenida en el segundo archivo generado. Debido a la cantidad de datos, no se alcanza apreciar más que el máximo y el mínimo de la señal. Algo importante a rescatar, es el hecho de que los tres canales tienen la misma gráfica, tal y como se esperaba.

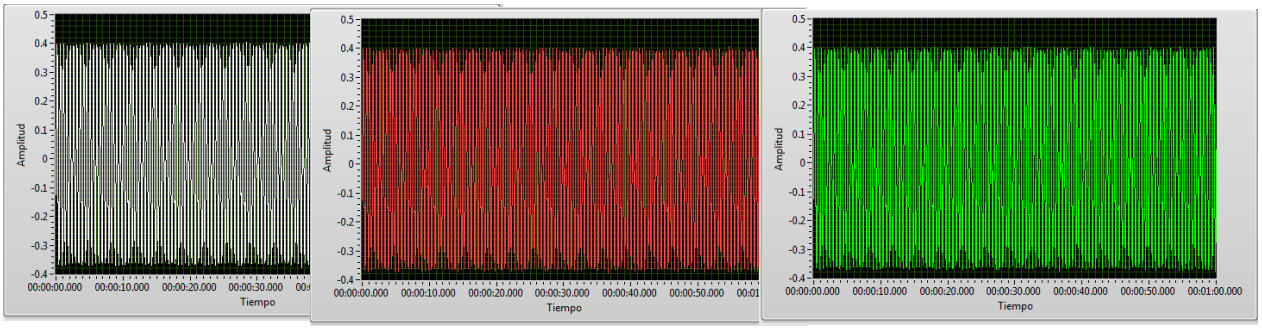


Figura 4.37. Segundo archivo generado.

En la figura 4.38 se presentan también los tres canales adquiridos, pero en lugar de mostrar todo el archivo, esta vez sólo se muestran los últimos cinco segundos de adquisición. Estas gráficas no están tan saturadas de datos como las anteriores, se alcanza a ver la forma sinusoidal que reproduce el geófono. Además, se alcanza a ver con detalle el máximo y el mínimo de la señal.

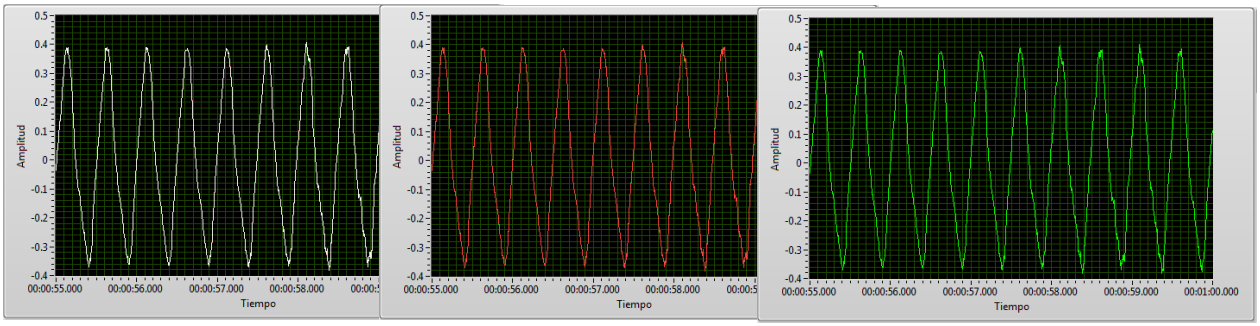


Figura 4.38. Tercer archivo generado.

Para el último archivo, se obtiene la gráfica correspondiente a un segundo de operación, figura 4.39. Estas gráficas permitirán encontrar la frecuencia de la señal sin problemas. Es importante resaltar de que con las diferentes gráficas presentadas, es posible apreciar el potencial del acceso aleatorio a los archivos. En el caso de archivos grandes (200 MB o más), el acceso aleatorio tiene mayor utilidad, ya que permite reducir el uso de memoria RAM de la PC.

4.2. Pruebas con generador de señales

En la sección anterior se mostró todo lo que conlleva un proceso de adquisición de datos. Es decir, se mostró desde la configuración hasta el procesamiento de los archivos binarios. Para la prueba con generador de señales se omitirá toda la parte de configuración y la descripción de la transferencia de archivos vía FTP, únicamente se abordará la interconexión de la prueba y el procesamiento de la información adquirida.

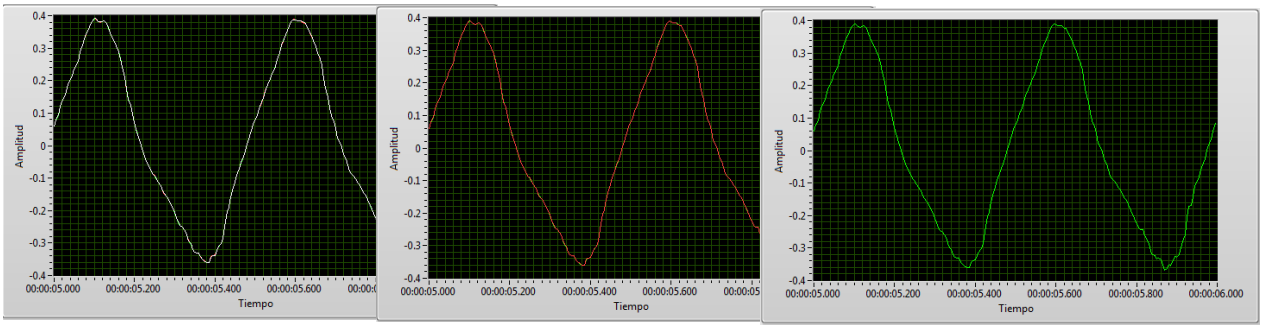


Figura 4.39. Cuarto archivo generado.

En la figura 4.40 se aprecia el equipo y la interconexión usada en la prueba. De izquierda a derecha se observa un generador de señales Tektronix modelo AFG 3022B, un osciloscopio Tektronix modelo TDS 1001C-EDU, la imagen almacenada de lo que

recibe el osciloscopio y una vez más el cRIO conectado a una PC. El generador de señales tiene que entregar una señal sinusoidal de 4 Vpp a una frecuencia de 100 Hz. Esta señal la estarán recibiendo simultáneamente el sistema y el osciloscopio. El osciloscopio, al igual que la prueba anterior, servirá de testigo.

La configuración para llevar a cabo esta prueba consiste en lo siguiente:

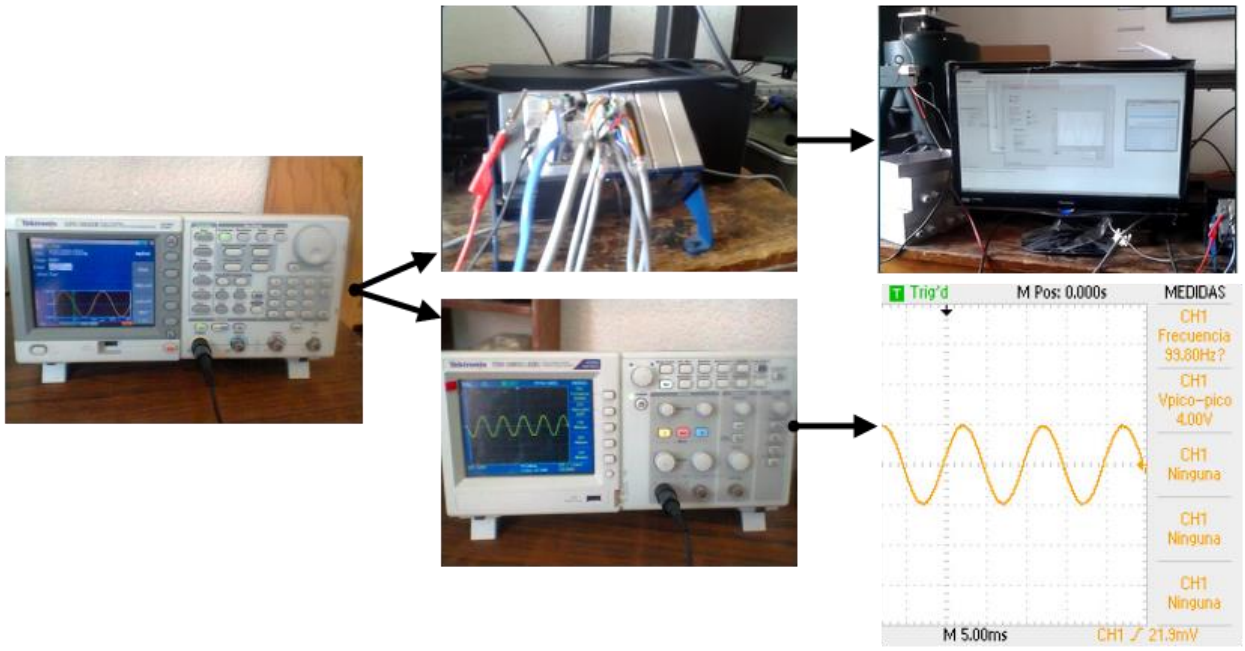


Figura 4.40. Interconexión para pruebas con generador.

- El modo de almacenamiento es temporizado. Se ha fijado un tiempo de 180 segundos para cada archivo de datos.
- El número de canales es igual a 3.
- La frecuencia de muestreo es igual a 2,000 Hz.
- Los nombres de los canales son canal 1, canal 2 y canal 3.
- Las unidades son volts.
- El rango de medición es de +/- 5 V.
- El modo de operación del canal 1 es NRSE, el del canal 2 es NRSE y el del canal 3 es RSE.

Se generaron tres archivos, figura 4.41. Todos los archivos deben tener el mismo tamaño. Teóricamente se puede obtener el tamaño de un archivo; se sabe que cada dato es equivalente a 32 bits, lo que es igual a 4 bytes. Al tener una frecuencia de muestreo de 2,000 Hz y 3 canales, se tienen 6,000 datos por segundo. Además, se sabe que son 180 segundos contenidos en el archivo. Lo último a considerar es el encabezado de tiempo, este ocupa 64 bits, lo que equivale a 8 bytes. Luego entonces se utiliza la siguiente ecuación para encontrar el tamaño en bytes de un archivo:

$$4 \text{ bytes} \times 6{,}000 \text{ mps} \times 180 \text{ segundos} + 8 \text{ bytes} = 4{,}320{,}008 \text{ bytes} \quad (4.1)$$

Para comprobar la validez de la ecuación, en la figura 4.42 se pueden ver las propiedades de los tres archivos. Las propiedades fueron obtenidas en un sistema operativo Windows Seven Home Premium. El campo de interés es tamaño, encerrado con un ovalo en la figura. Tamaño en disco se desprecia, ya que éste tiene que ver con el sistema de archivos y no estrictamente con la cantidad de información contenida.

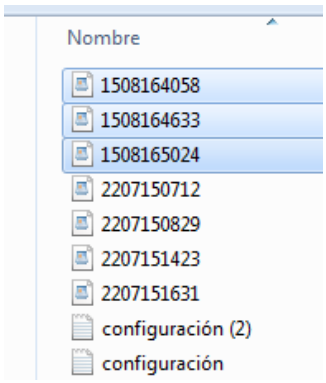


Figura 4.41. Archivos en PC de modo temporizado.

Realizada la inspección del tamaño de los archivos, se prosigue a ver que la información contenida en ellos sea la que el generador de señales estaba entregando. La señal que se aplicó siempre fue la misma, por lo tanto se debe obtener gráficas similares en los tres archivos.

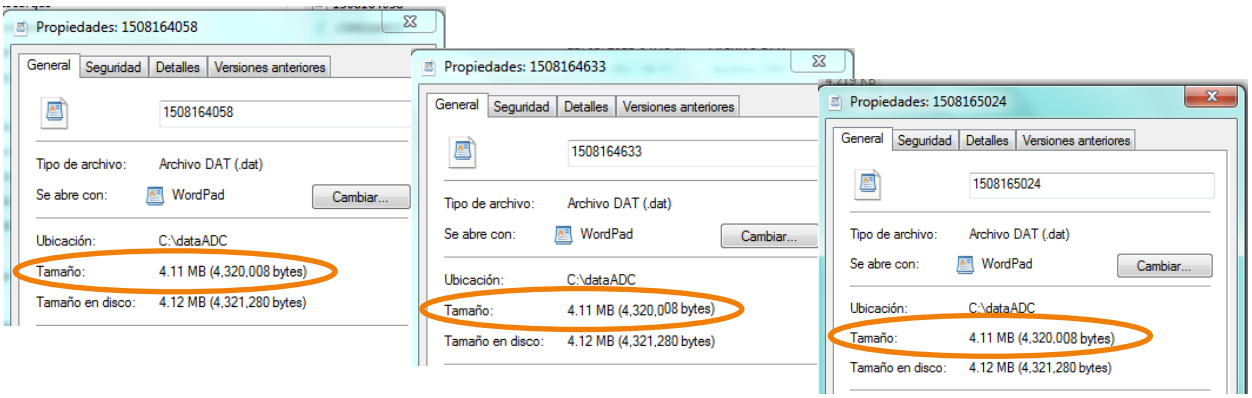


Figura 4.42. Propiedades de los archivos.

Las gráficas correspondientes al primer archivo se presentan en la figura 4.43. Se ha solicitado ver desde el segundo 10 hasta el segundo 14. A pesar de que los tres canales están habilitados para despliegue, únicamente se puede ver la gráfica del canal 1 (blanca). Sin embargo, observando la tabla de datos, se puede apreciar que el valor de los tres canales se modifica en el tercer decimal. Esto se puede atribuir al hecho de que los canales han sido configurados en modos distintos y además a que las muestras no son tomadas en paralelo (ADC multiplexado). Debido a que la señal que se aplicó

nunca varió, se observa que los máximos y mínimos siempre son los mismos (+2 y -2). Por la alta cantidad de datos desplegados en la figura 4.43, en lugar de ver una señal sinusoidal se alcanza a ver un bloque en forma rectangular.

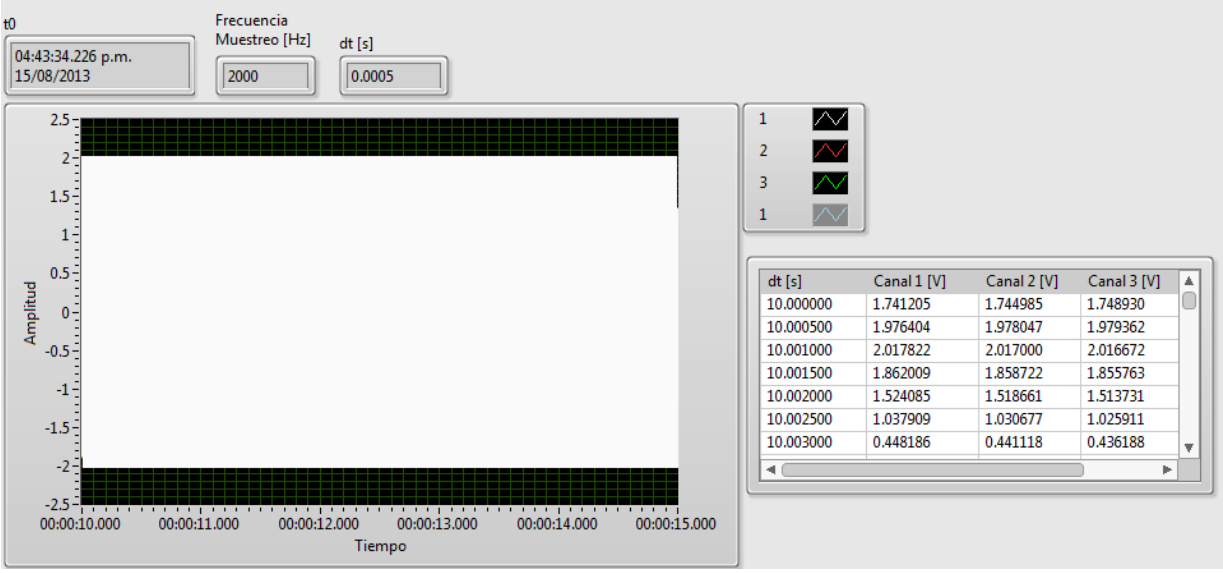


Figura 4.43. Primer archivo temporizado.

En el caso del segundo archivo, se ha escogido desplegar un período de tiempo más corto, a manera de tener menos datos y mayor certeza de la forma de onda almacenada. En la figura 4.44 se puede ver la gráfica y la tabla correspondiente a la información en el segundo archivo. Una vez más los tres canales están habilitados, pero debido a que contienen la misma información, únicamente se distingue el canal 1. Se ha pedido realizar el despliegue de un segundo de información, pero como se tiene una señal de 100 Hz, adicionalmente se ha realizado un *zoom in* con las herramientas que acompañan a la gráfica. La lectura presenta información a partir del segundo 150. En la gráfica únicamente se muestra 0.150 segundos de información.

En el caso del tercero y último archivo, se busca lograr ver únicamente dos ciclos de la señal adquirida, esto requerirá de leer un segundo de información y posteriormente volver a realizar un *zoom in*. Para este caso, sólo se ha habilitado el contenido del canal 2 (rojo), figura 4.45. Se ha configurado el despliegue para mostrar a partir del segundo 179.

La descripción realizada en este capítulo muestra, a grandes rasgos, el funcionamiento del sistema. En el capítulo siguiente se abordarán los resultados y conclusiones referentes al presente trabajo.

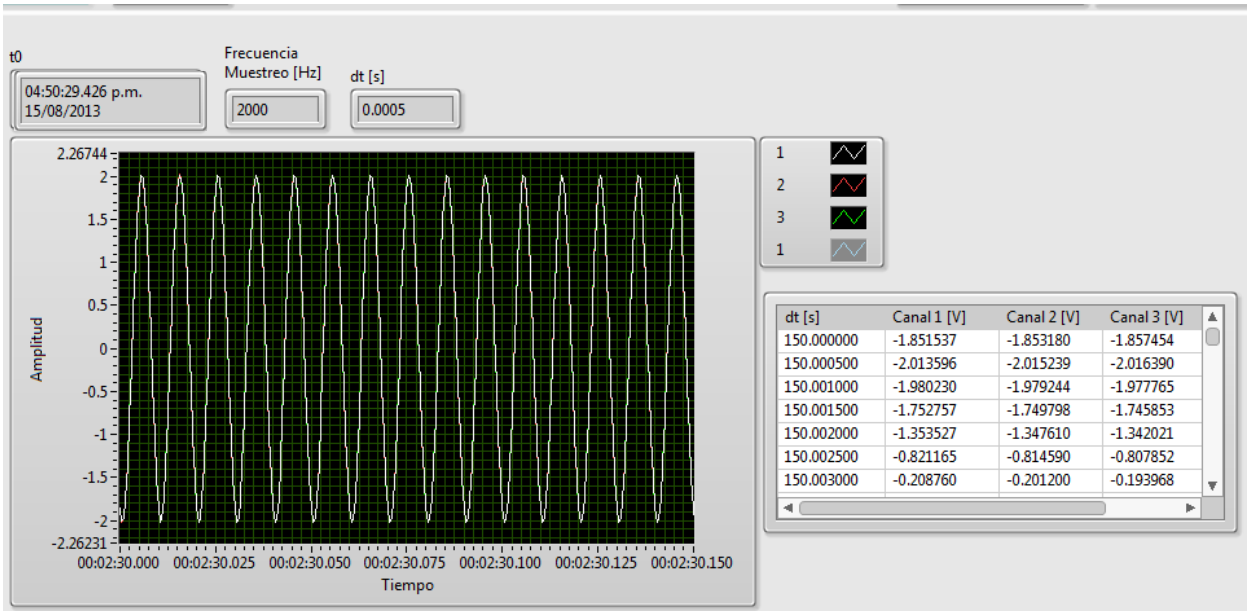


Figura 4.44. Segundo archivo temporizado.

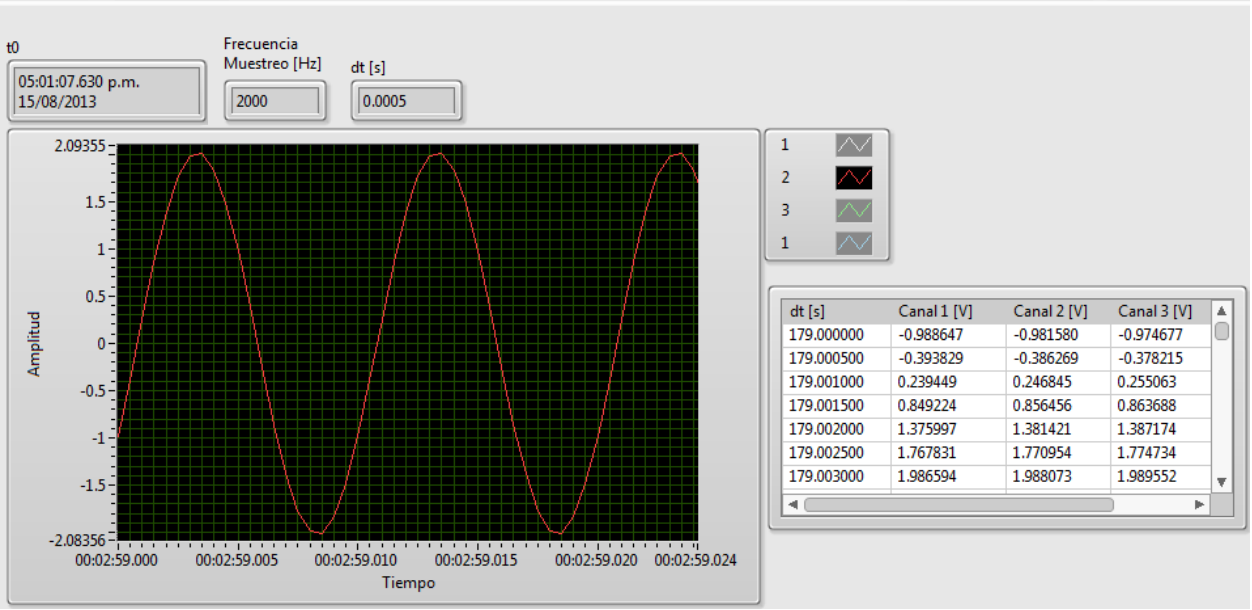


Figura 4.45. Tercer archivo temporizado.

Capítulo 5
Resultados y Conclusiones

En el presente capítulo se abordarán los objetivos alcanzados a través del sistema desarrollado. También se darán conclusiones sobre el trabajo realizado. Por último, partiendo del hecho que todo es perfectible, se plantearán expectativas a corto plazo y algunas modificaciones que podrían realizarse al sistema.

5.1. Resultados

Se logró realizar un sistema de adquisición de variables analógicas versátil y de gran alcance. El sistema involucra la siguiente lista de resultados:

- Software completamente intuitivo. Al existir pocos indicadores y controles, el usuario puede asimilar rápidamente las interfaces.
- Software autónomo, empotrado y confiable para un cRIO 9012. Este software está divido en dos parte. Existe una parte encargada de administrar al FPGA y otra que administra al MCU.
- Programación de un FPGA sin necesidad de usar lenguaje de descripción de hardware.
- Comunicación entre un MCU y un FPGA.
- Software para la adquisición de variables analógicas, independiente del sistema operativo. Éste está basado en HTML, con lo que únicamente se necesita un navegador web.
- Capacidad para trabajar con 13 frecuencias de muestreo diferentes. La frecuencia más alta de muestreo es cuando se adquiere un sólo canal y es de 125,000 Hz. La frecuencia más baja con la que se puede trabajar es de 3,200 Hz y es con 28, 29, 30, 31 y 32 canales activos.
- Se tiene una tasa de transferencia máxima de datos de 0.4768 MB por segundo entre el FPGA y el MCU.
- El sistema puede generar archivos con un tamaño de 286.10229 MB.

- Se puede almacenar hasta 25.6 GB de información. Este dato corresponde al 80% de una memoria USB de 32 GB. No se realizaron pruebas con dispositivos de mayor capacidad.
- Puede trabajar con los canales configurados en tres modos. Modo diferencial, modo NRSE y modo RSE. Los últimos dos modos pueden trabajar simultáneamente. Es decir, si hay ocho canales en adquisición, cuatro pueden estar trabajando en NRSE y otros cuatro en RSE.
- Cada canal cuenta con sus propios identificadores. Tiene un número asociado y un nombre.
- Se pueden seleccionar cuatro rangos de medición de voltajes: $\pm$10 V, $\pm$5 V, $\pm$1 V o $\pm$200 mV. Los rangos pueden trabajar simultáneamente, es decir, si hay siete canales en adquisición, tres pueden estar con un rango de $\pm$10 V, dos con $\pm$200 mV y el resto con $\pm$5 V.
- Los archivos en la memoria USB se encuentran agrupados por carpetas. Las carpetas tiene un identificador de tiempo y fecha. Adicionalmente el usuario puede proporcionar un identificador.
- Se pueden elegir entre tres formatos de tiempo: horario de verano local, horario local o UTC. El horario de verano local equivaldría a restar 5 horas a la hora del meridiano de Greenwich. El horario local equivaldría a restar 6 horas a la hora del mismo meridiano. En el caso del UTC se estaría usando el tiempo de Greenwich.
- Todas las amplitudes adquiridas pueden ser amplificadas en pos procesamiento. Es posible tener una amplificación hasta por un factor de 100.
- El sistema permite realizar la transferencia de archivos vía FTP. Esto permite que no sea necesario tener acceso físico a la memoria USB.
- Se puede visualizar el contenido de la memoria USB a distancia.
- Software para visualización y conversión de datos binarios a ASCII. Éste está diseñado para PC; únicamente corre sobre Windows XP, Vista y Seven (cualquier versión de éstos). Los archivos binarios permiten alcanzar tasas altas de escritura, mientras que la información en ASCII, permite tener compatibilidad con softwares comerciales.

5.2. Conclusiones

Se logró desarrollar un sistema de adquisición de señales analógicas estándar. Este sistema constituye un sistema embebido robusto, flexible y confiable. El sistema es capaz de adquirir cualquier tipo de señal analógica previamente acondicionada. Las bases para la construcción del sistema son: a nivel hardware cRIO 9012 y a nivel software LabVIEW.

Con base en lo anterior, se puede decir que se construyó un adquisidor de señales analógicas para distintas áreas de la ingeniería. La lógica desarrollada para la adquisición permite tener una diversidad de aplicaciones. Los problemas para adquirir señales analógicas se resumen a acondicionamiento de señal.

Se logró utilizar programación gráfica para programar una PC, un MCU y un FPGA. La forma en la que se ha programado el FPGA, sin necesidad de usar descripción de hardware, propone una nueva forma de desarrollar sistemas basados en FPGA. A pesar de que para programar microcontroladores existen lenguajes de alto nivel, el uso de LabVIEW para el mismo fin propone un nuevo paradigma de desarrollo. La transferencia de información vía PCI, Ethernet y FTP, haciendo uso de herramientas de LabVIEW, permiten desarrollar sistemas con mayor rapidez y con mayor alcance.

Esta tesis sin duda ayudará a expandir el uso de cRIO en el Instituto de Ingeniería de la UNAM. Proporciona fundamentos sólidos para futuros desarrollos basados en cRIO. De hecho, se tiene un desarrollo modular, lo cual permite identificar a plenitud todas las partes del software y hardware involucradas. Lo anterior implica que se pueden generar variantes del sistema desarrollado de una manera sencilla.

Desde un punto de vista académico, sin duda alguna, el sistema desarrollado puede fungir como base para reafirmar lo aprendido en distintos temas. Ejemplo de éstos son: conversión analógica-digital, programación gráfica para PC, programación de FPGAS con lenguaje gráfico, programación de microcontroladores con lenguaje gráfico, comunicación entre dispositivos electrónicos, comunicación FTP, comunicación Ethernet, sistemas embebidos, aplicaciones web, archivos binarios, manejo de memoria, compatibilidad de LabVIEW con otras aplicaciones, entre otros.

5.3. Comentarios finales

Equipos como cRIO tienen más de cinco años en el mercado, sin embargo, el uso de este tipo de tecnología es casi nulo en el Instituto de Ingeniería de la UNAM. Muchas veces la barrera principal es el conocimiento para utilizar la tecnología, con el presente trabajo esta barrera se ha empezado a romper.

Se sabe que prácticamente cualquier sistema es perfectible, por lo que existen mejoras que se pueden implementar al presente trabajo. Algunas mejoras que podrían ser incorporadas al sistema son:

- Longitud de palabra variable. Hasta el momento, cuando se transfirieren los datos desde el FPGA hacia el MCU, todos los datos se convierten a 32 bits, sin excepción. Podría desarrollarse un algoritmo adaptable, éste tendría que identificar el rango de medición seleccionado por el usuario y con base en eso, generar la longitud de palabra. Lo anterior permitirá reducir considerablemente el espacio en disco que se necesita para cada dato.
- Incrementar el número de configuraciones posibles de adquisición. Se vio que el principal inconveniente de esto es la función que manda a leer el convertidor. Se podría intentar plantear un algoritmo de lectura, en donde se incremente la eficiencia de la función que manda a leer el convertidor.
- En el modo continuo existe un tiempo muerto entre el cierre y la apertura de un nuevo archivo. Se podría trabajar en un algoritmo que tenga dos archivos abiertos simultáneamente (actual y posterior), cuando uno se quiera cerrar, se

cambiaría la referencia al otro archivo y mientras se escribe en este último se cierra la referencia al otro.

- En el programa para el pos-procesamiento de la información binaria, se podría implementar una opción que permita convertir un conjunto de canales a ASCII en lugar de sólo un canal.
- De manera similar al punto anterior, se podría permitir que en lugar de 4 canales, el usuario pudiera elegir 5 o más canales.
- Para el programa de procesamiento, actualmente se tienen que seleccionar siempre cuatro canales, se tendría que mejorar el algoritmo para seleccionar únicamente los que se quieren.
- Hacer pruebas para acceder a las páginas web desde una computadora en una red externa. Las pruebas únicamente fueron hechas en una red local. Un problema inmediato con esto sería la asignación de la IP.
- Al igual que el acceso a las páginas web, se podrían hacer pruebas en una red externa para probar las transferencias FTP.
- Podría implementarse un algoritmo que permitiera escribir en un archivo y monitorear un canal al mismo tiempo. Actualmente para la integridad de los datos, cuando éstos se almacenan, no se hace un despliegue de los mismos.
- En lugar de monitorear un canal, se podría implementar el despliegue de varios canales simultáneamente. Un detalle a considerar sería el ancho de banda que puede utilizar Ethernet con cable cruzado.

Se espera que el Instituto de Ingeniería de la UNAM, sobre todo la Coordinación de Instrumentación, presten mucha atención al conocimiento que la presente tesis ha generado. A pesar de que se necesita recurso humano preparado, se sabe de antemano que en la población universitaria hay potencial para continuar con este proyecto. El software y el hardware que soportan el desarrollo de la presente tesis se encuentra disponible en la Coordinación. El siguiente paso importante radica en identificar proyectos, en los cuales se pueda implementar lo hasta ahora desarrollado y de ser necesario, hacer las adecuaciones pertinentes.

Con el acelerado crecimiento de la tecnología, actualmente es posible crear sistemas tan sofisticados como el que en esta tesis se presenta. Las diferentes áreas de la ingeniería necesitan aprovechar que pueden contar con una herramienta de este tipo. De hecho, para estar en la frontera de la investigación, es importante estar en la frontera de la tecnología.

APÉNDICES

A. Ventanas del software del sistema

B. Código implementado

APÉNDICE A

Ventanas del software del sistema

En el presente apéndice se encuentran todas las ventanas del sistema a través de las cuales el usuario puede interactuar con el cRIO. Las ventanas se irán presentando con base en el orden que sus diagramas de flujo tuvieron en el capítulo 3.

Ventanas relacionadas al Software de la PC

Figura A.1. Menú.

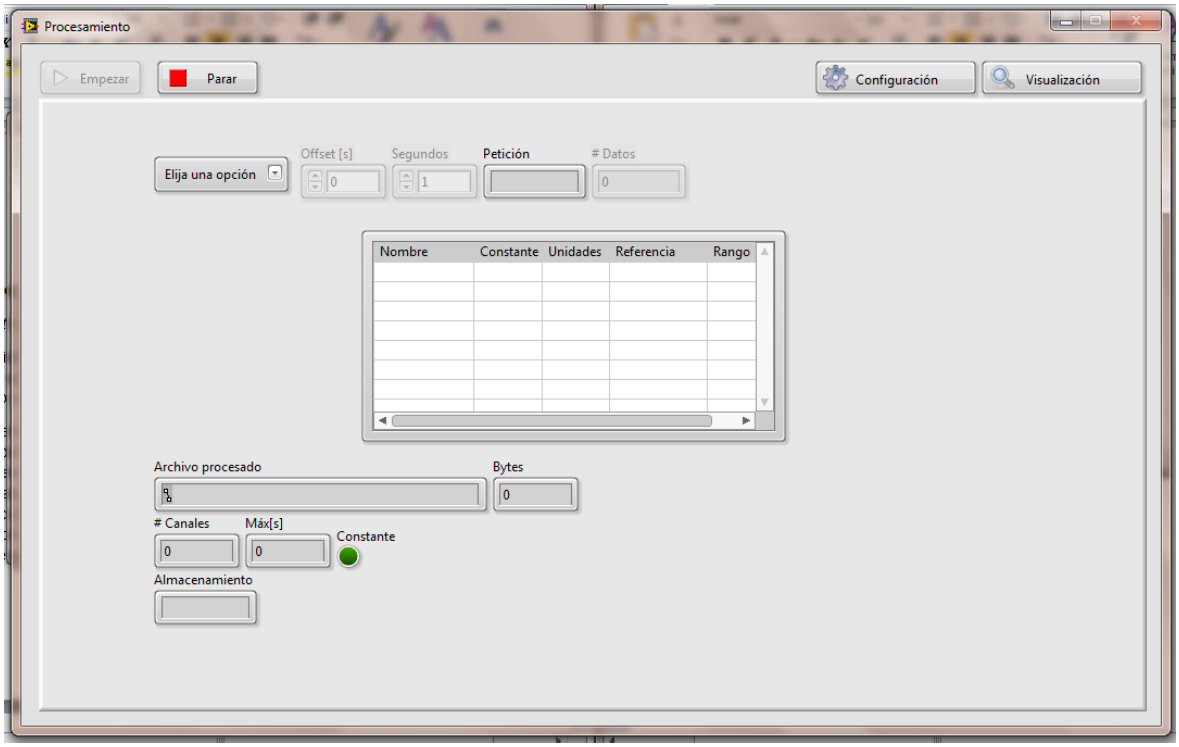


Figura A.2. Procesamiento, sección Configuración.

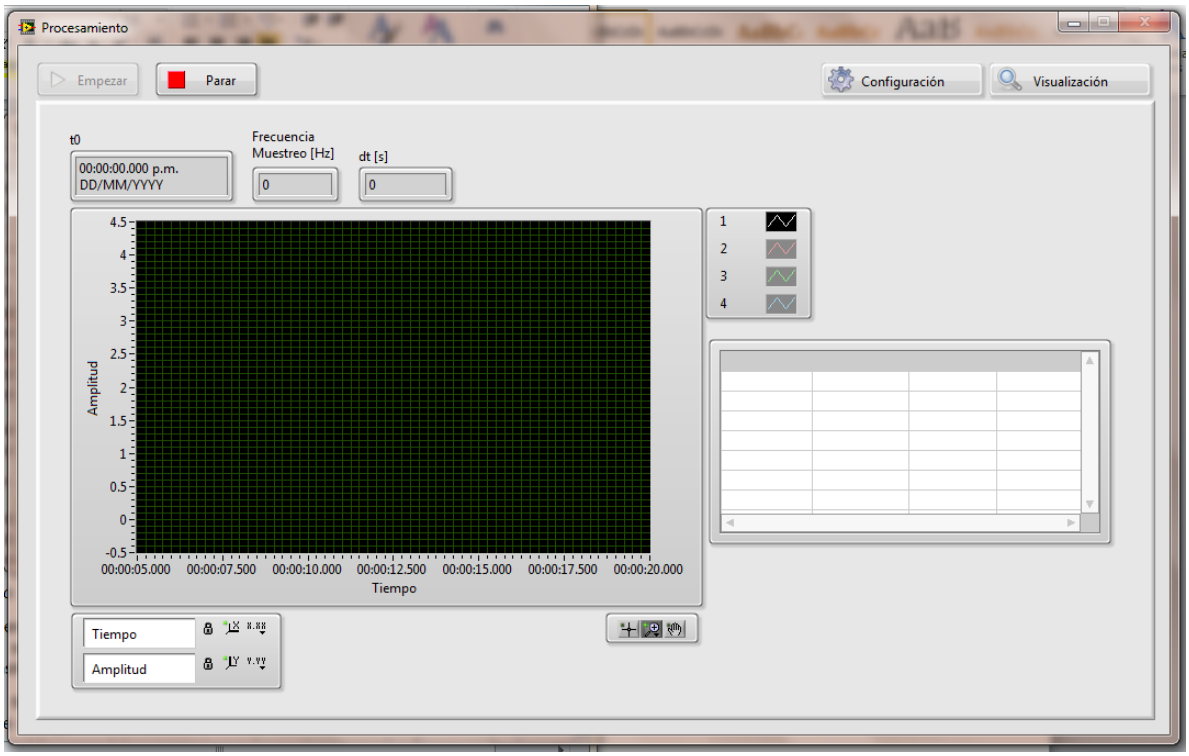


Figura A.3. Procesamiento, sección Visualización.

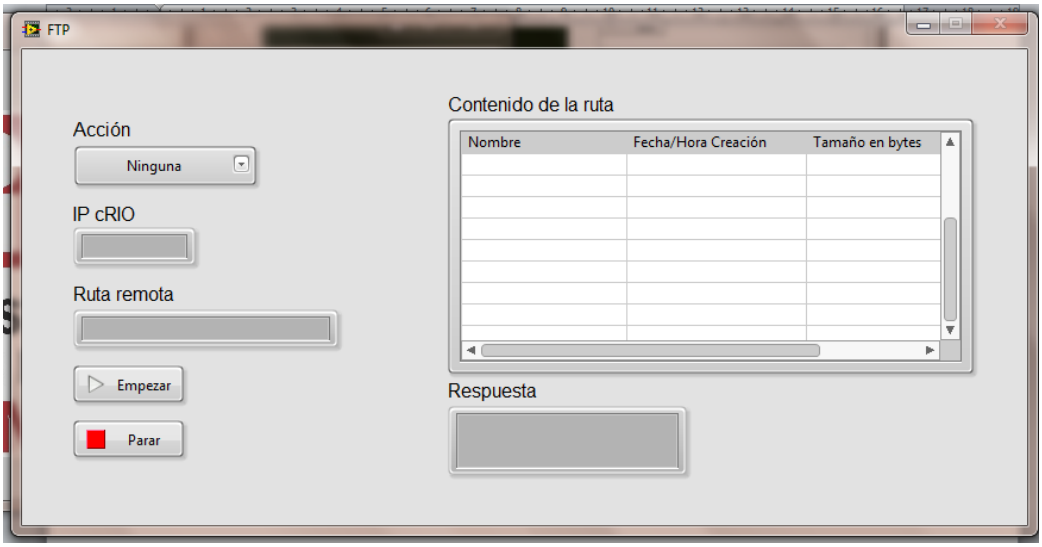


Figura A.4. FTP.

Ventanas relacionadas al Software del MCU

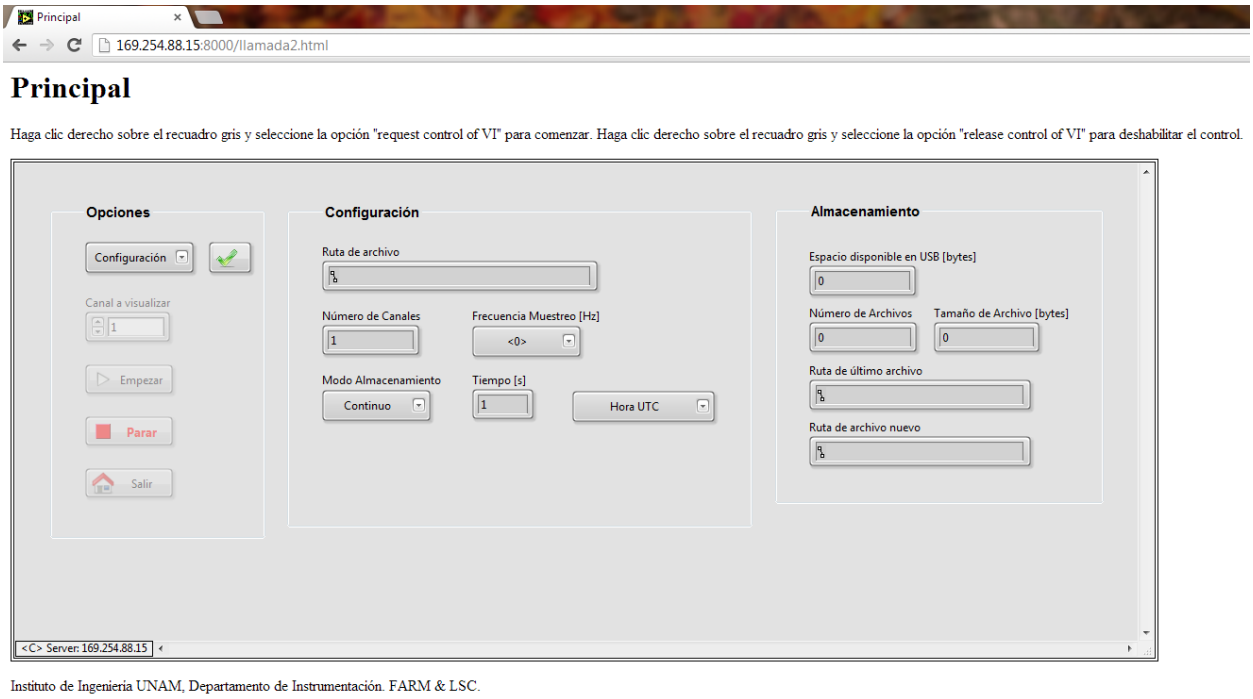


Figura A.5. Principal, página web principal.

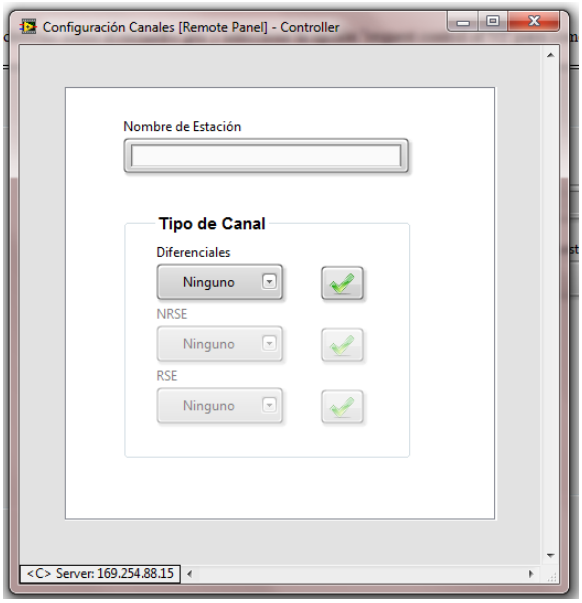


Figura A.6. Configuración canales, modo de operación para los canales.

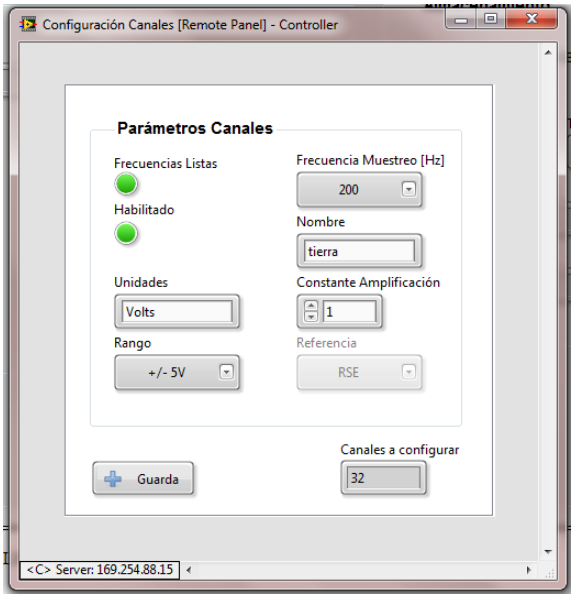


Figura A.7. Configuración canales, parámetros de operación para los canales.

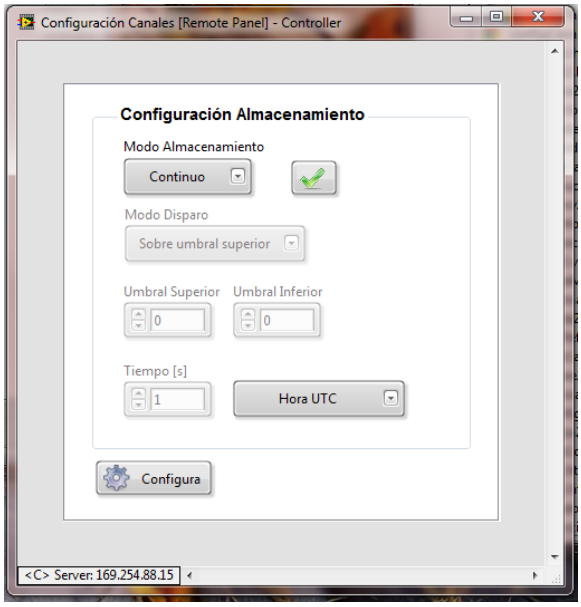


Figura A.8. Configuración canales, parámetros del modo de almacenamiento de datos.

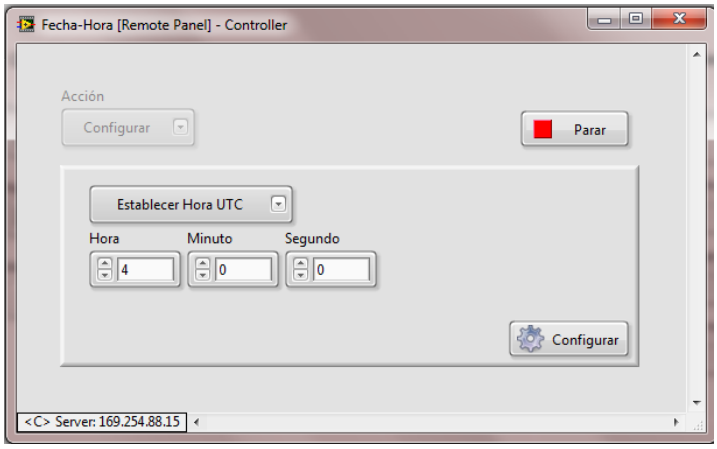


Figura A.9. Fecha-Hora, configuración de hora.

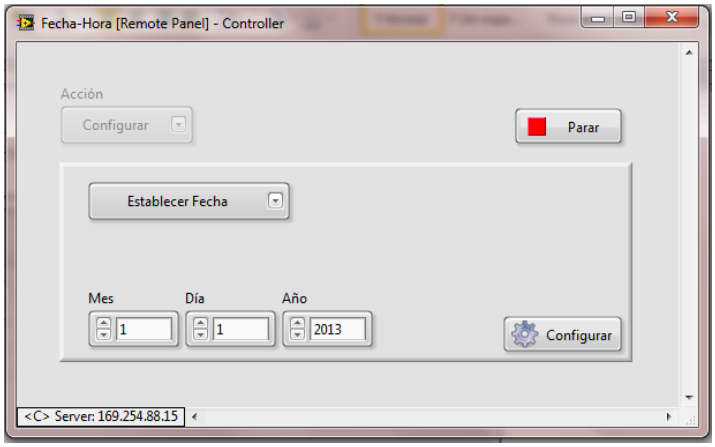


Figura A.10. Fecha-Hora, configuración de fecha.

Figura A.11. Modo Continuo.

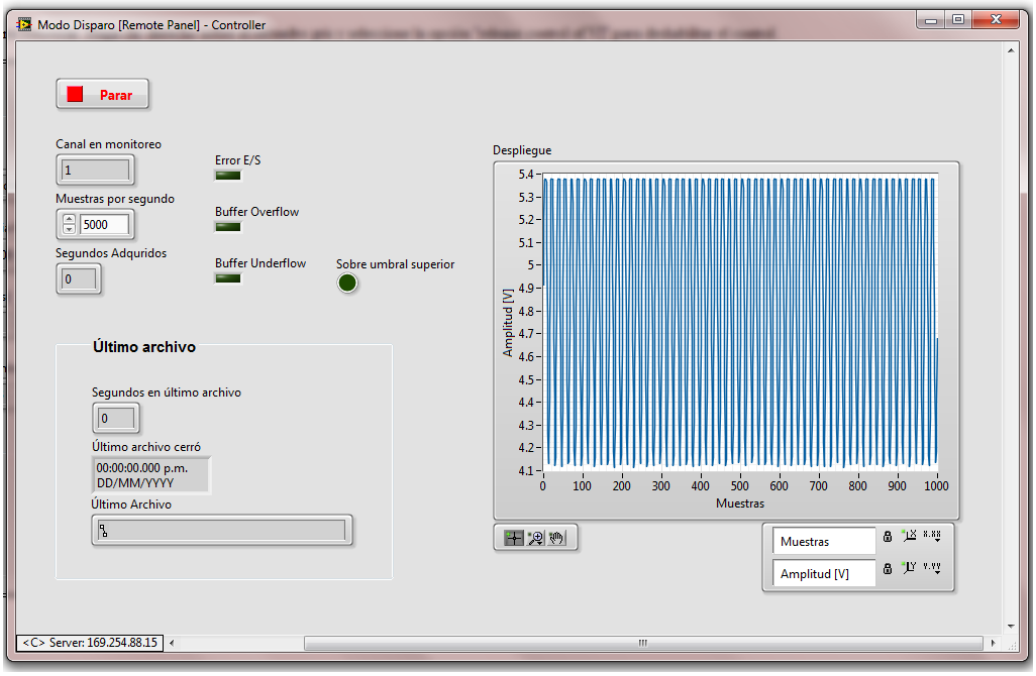


Figura A.12. Modo Disparo.

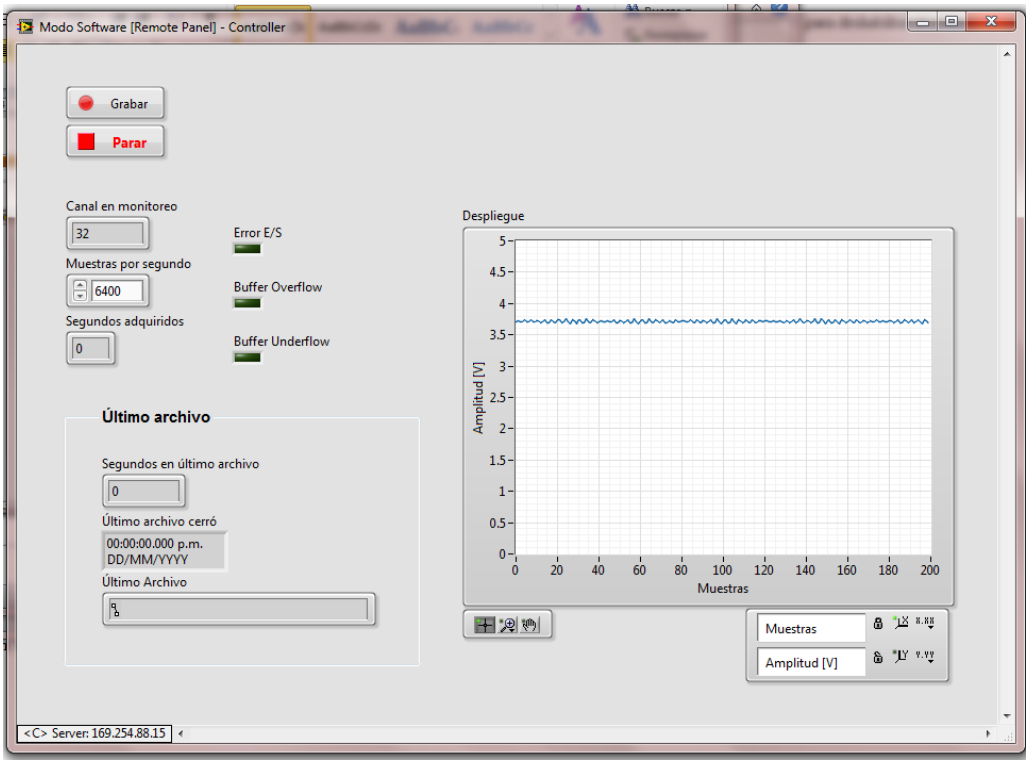


Figura A.13. Modo Software.

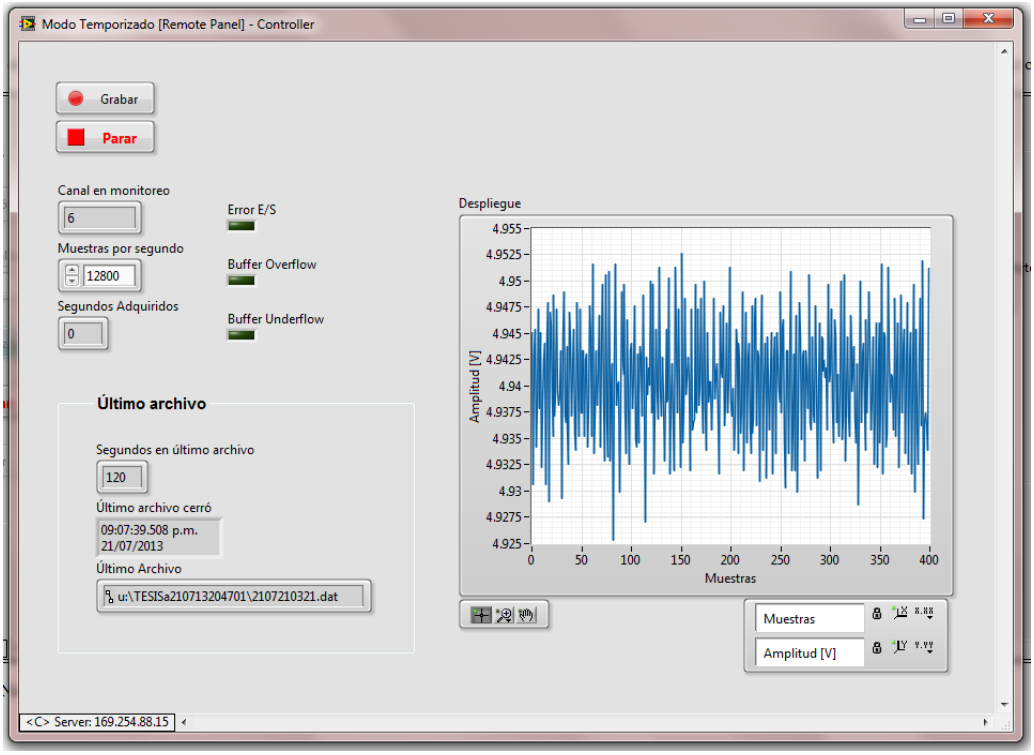


Figura A.14. Modo Temporizado.

APÉNDICE B

Código implementado

En este apéndice se presenta parte del código implementado en LabVIEW y que está encargado de administrar al sistema de adquisición de datos. Con la finalidad de facilitar la asociación de los diagramas de flujo del capítulo 3 con sus respectivos diagramas de bloques (código), los nombres de ambos mantienen una relación muy estrecha.

Código relacionado al Software de la PC

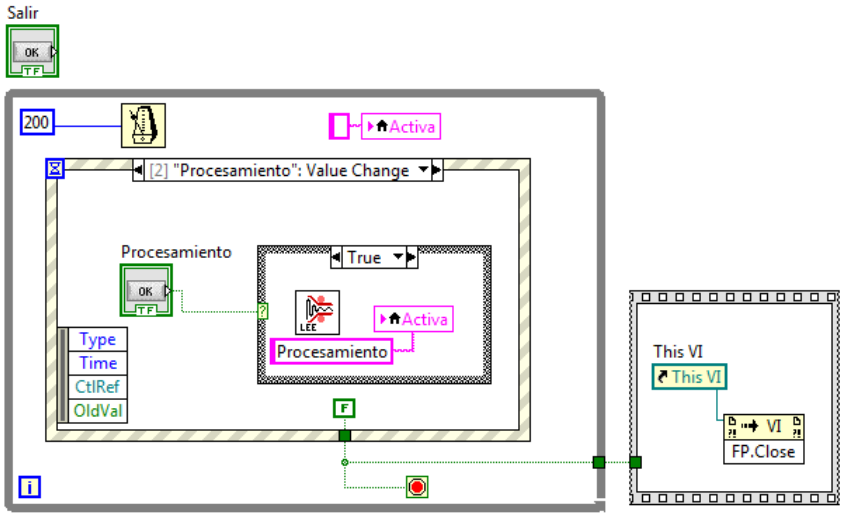


Figura B.1. Menú, llamado al sub VI de ventana Procesamiento.

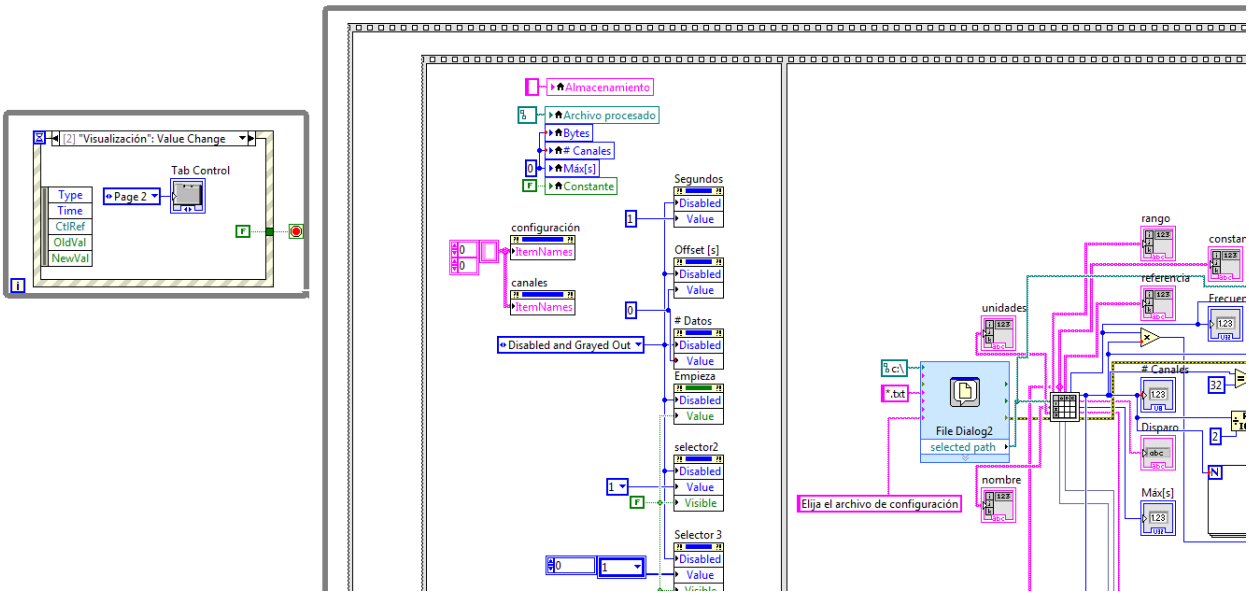


Figura B.2. Procesamiento, configurando controles e indicadores.

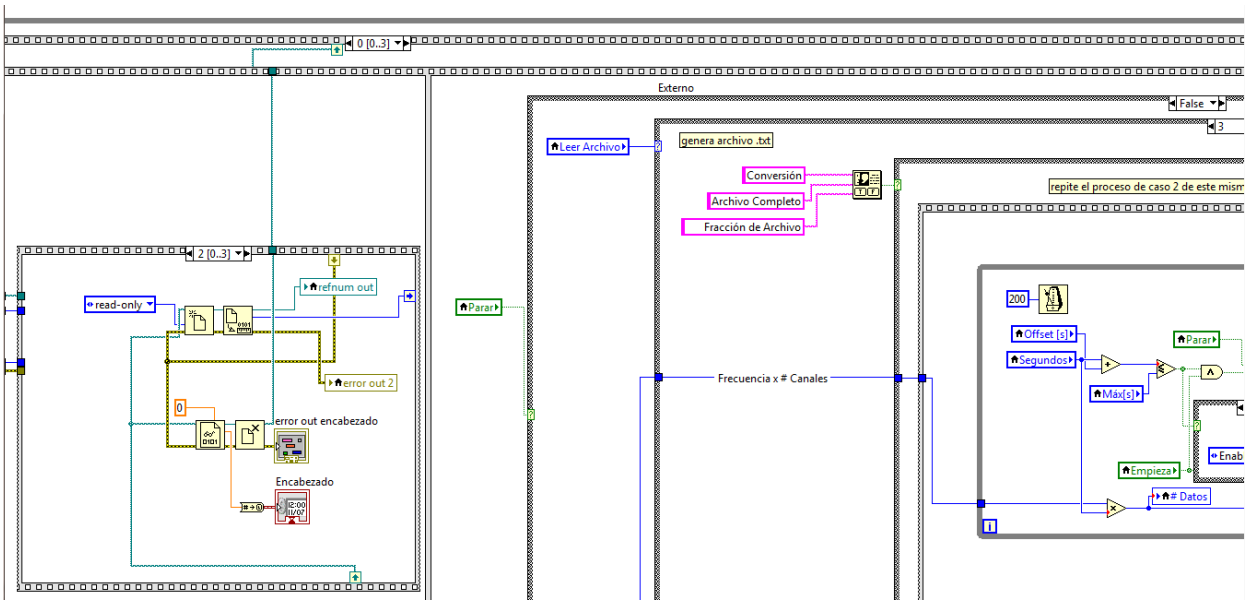


Figura B.3. Procesamiento, lectura de encabezado de archivo.

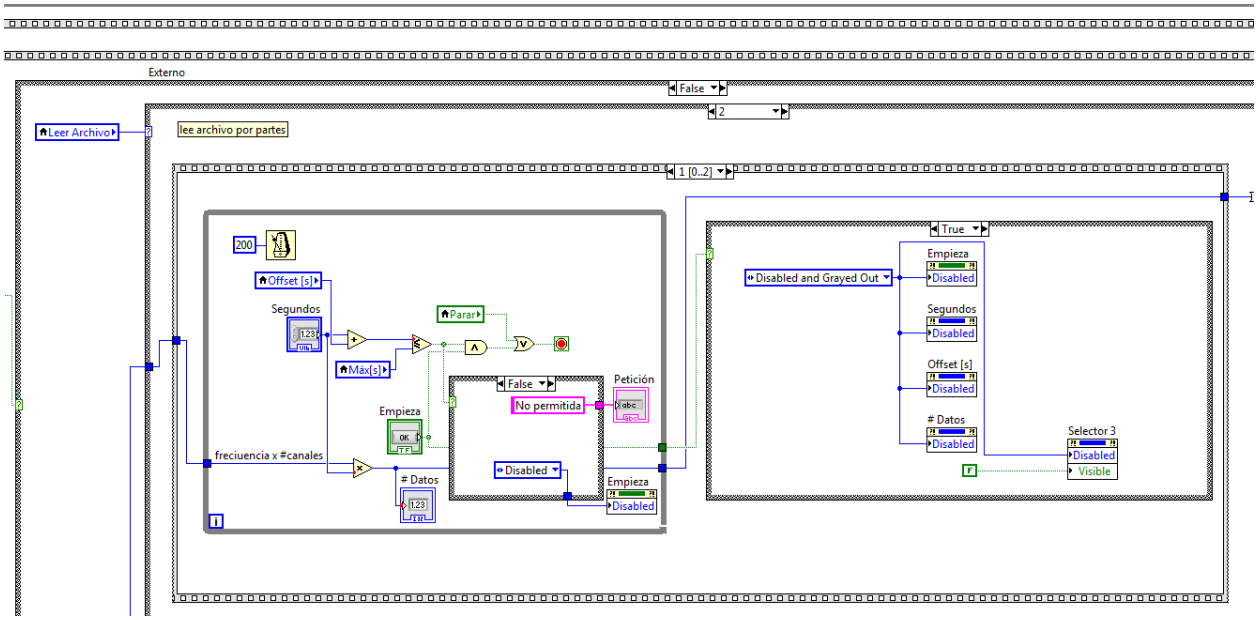


Figura B.4. Procesamiento, lectura de archivo por partes.

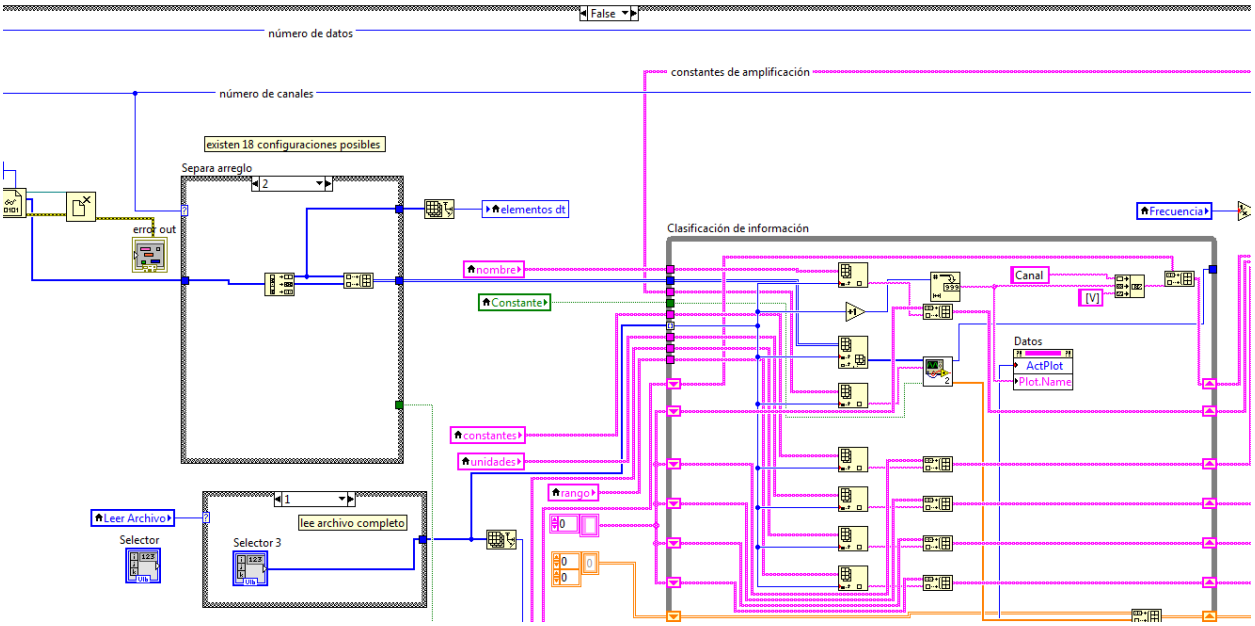


Figura B.5. Procesamiento, asignación de valores a indicadores.

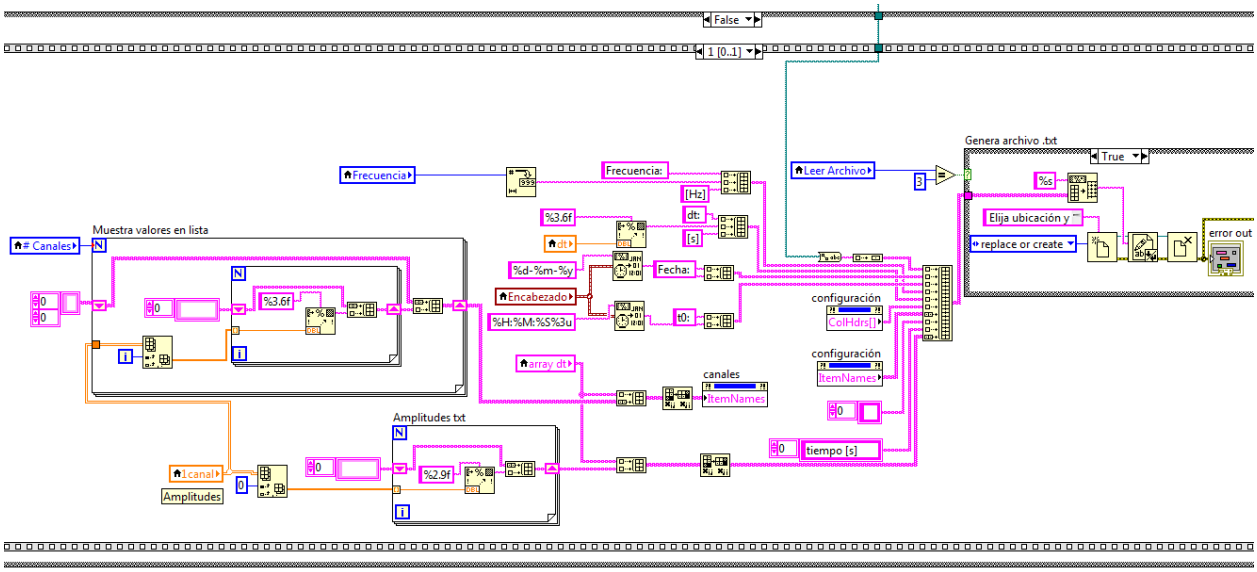


Figura B.6. Procesamiento, generación de archivo ASCII.

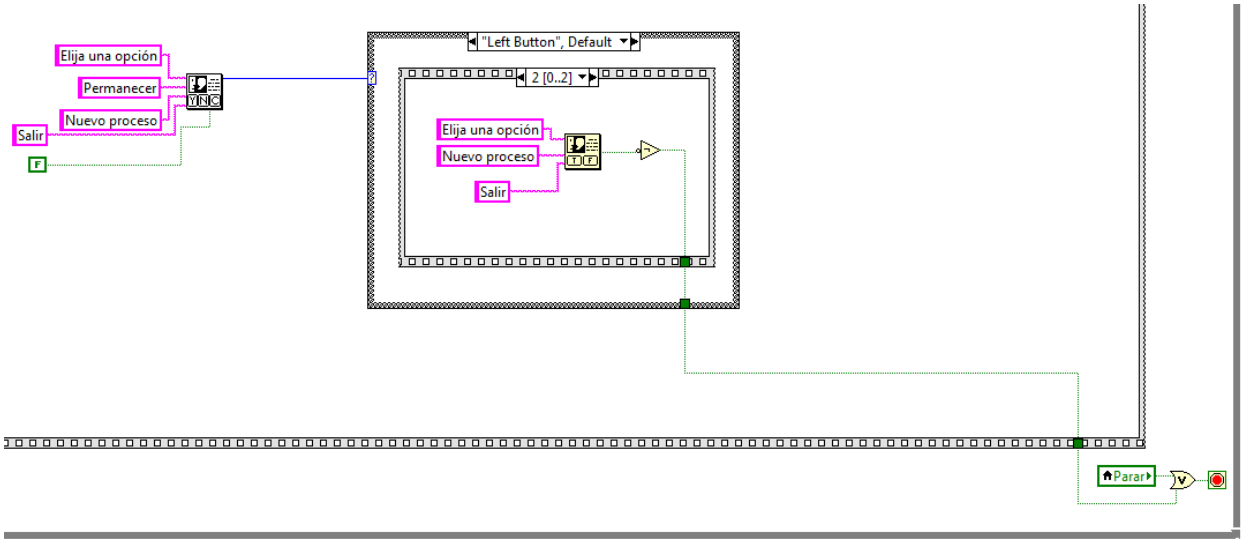


Figura B.7. Procesamiento, condición para detener ejecución.

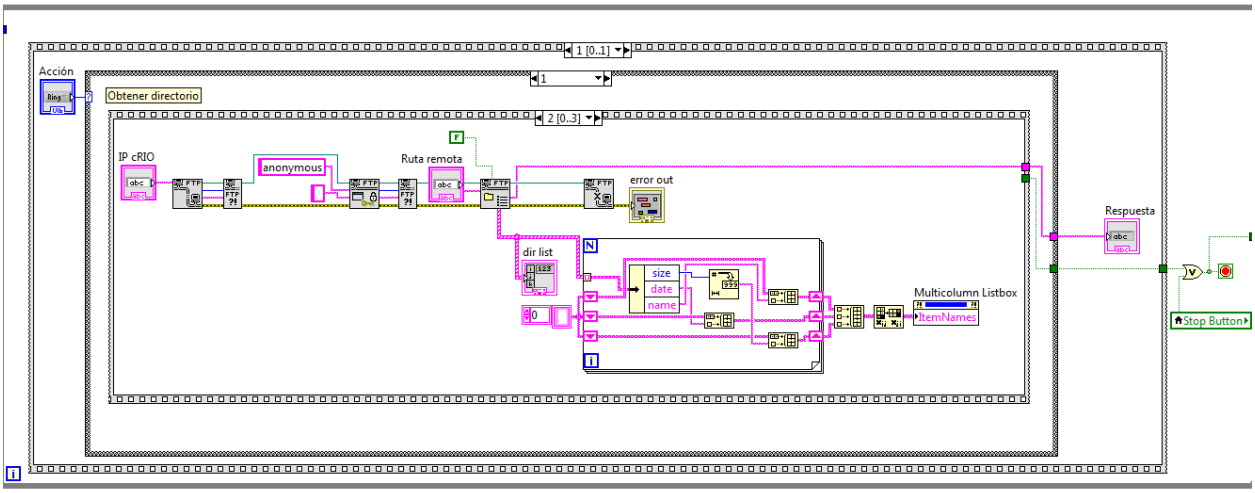


Figura B.8. FTP, obtención del directorio remoto.

Código relacionado al MCU

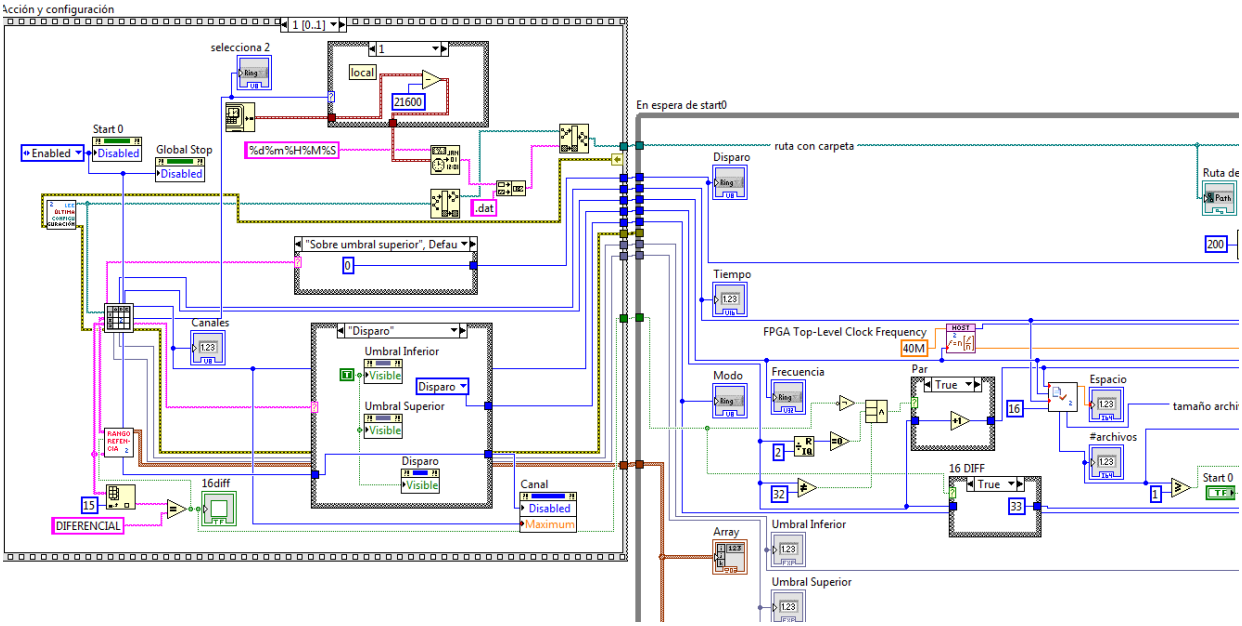


Figura B.9. Principal, inicio y selección de acción a realizar.

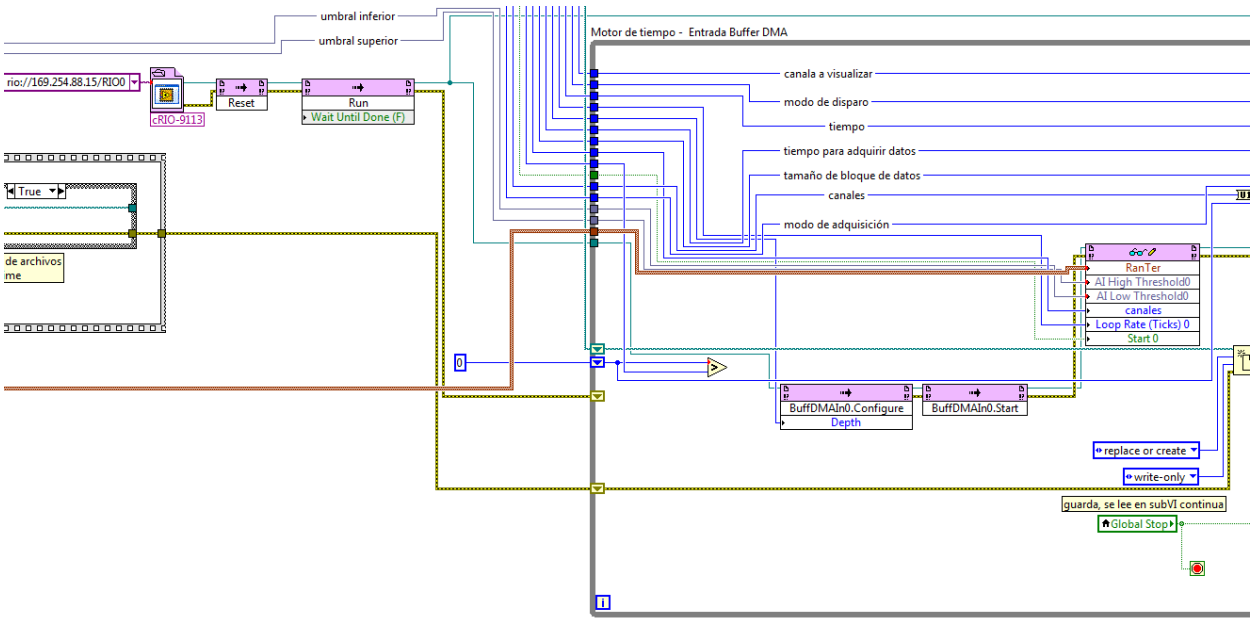


Figura B.10. Principal, inicio de comunicación con el FPGA.

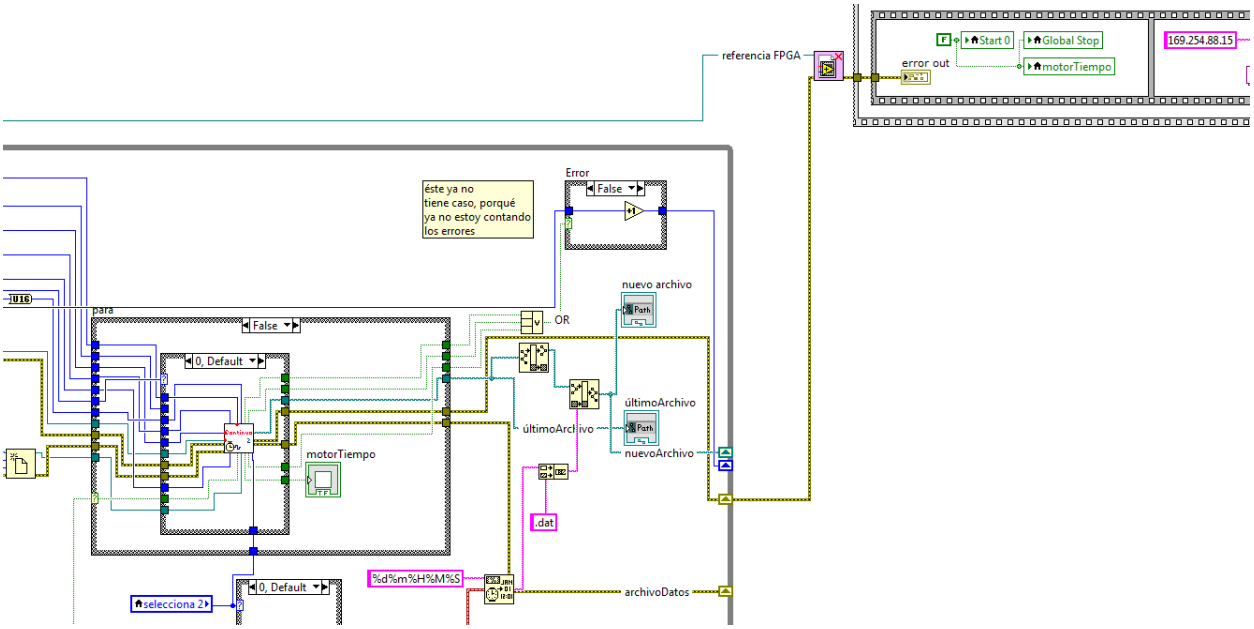


Figura B.11. Principal, llamada a sub VI para adquisición de datos.

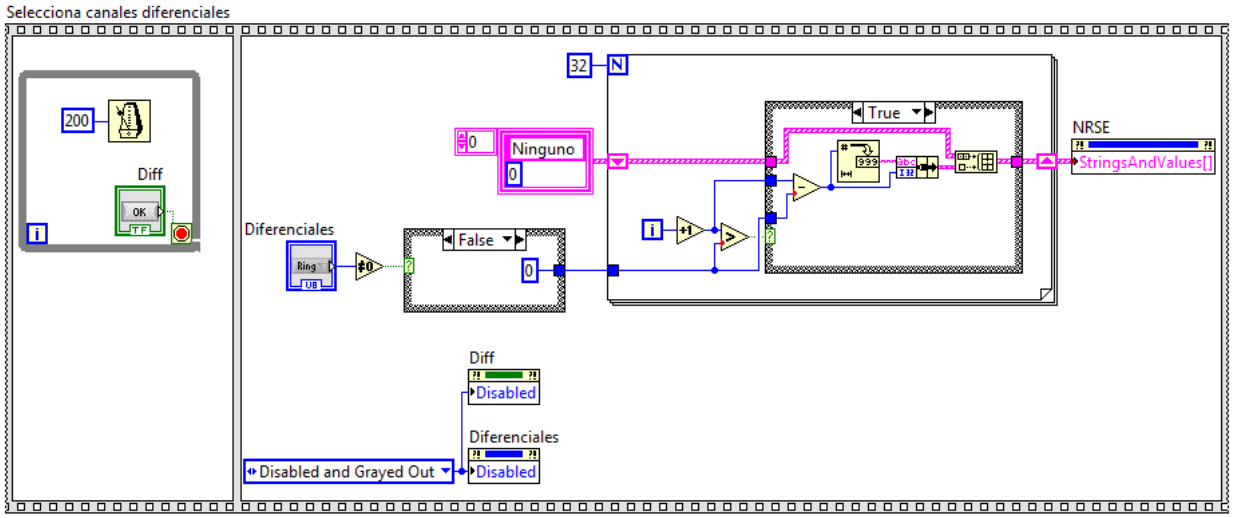


Figura B.12. Configuración Canales, selección de diferenciales.

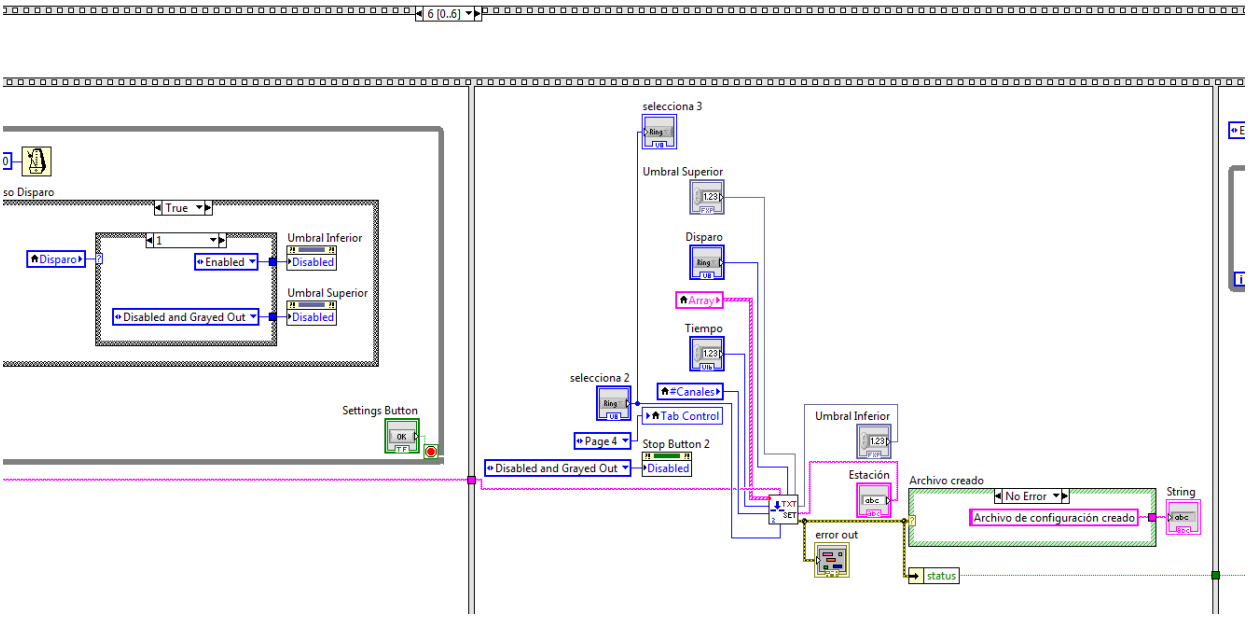


Figura B.13. Configuración Canales, almacenando configuración.

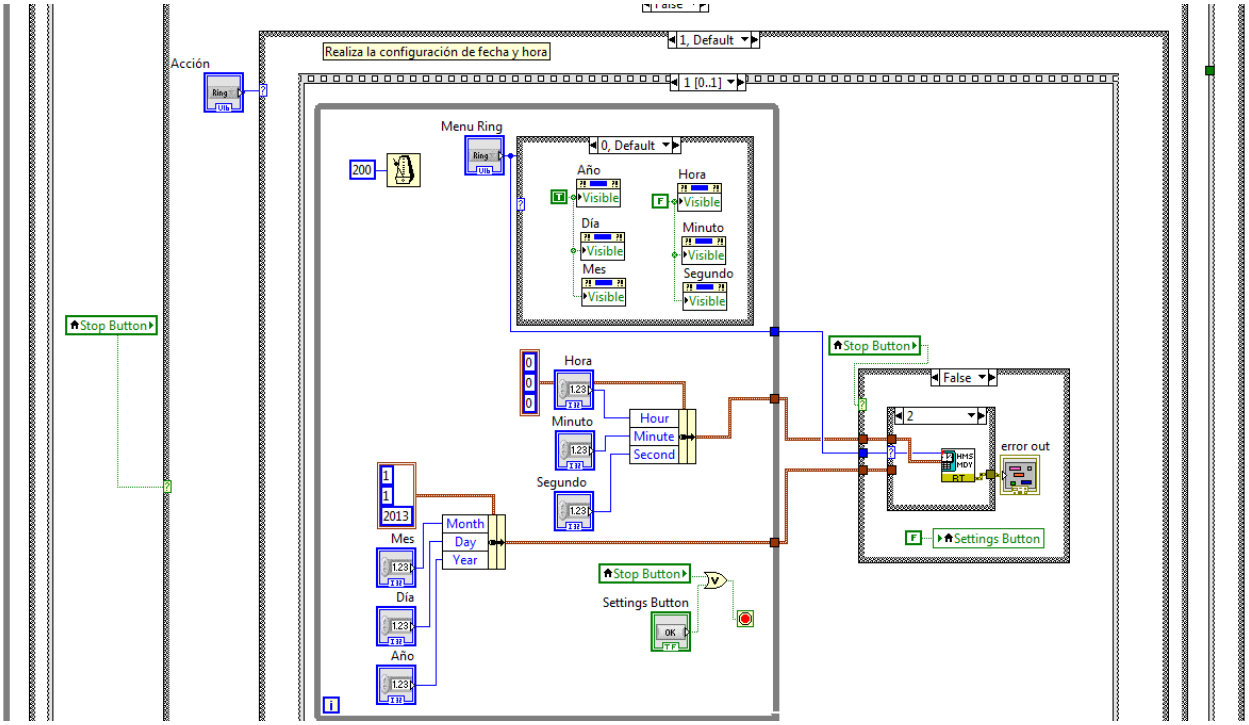


Figura B.14. Fecha-Hora, modificación de fecha y hora.

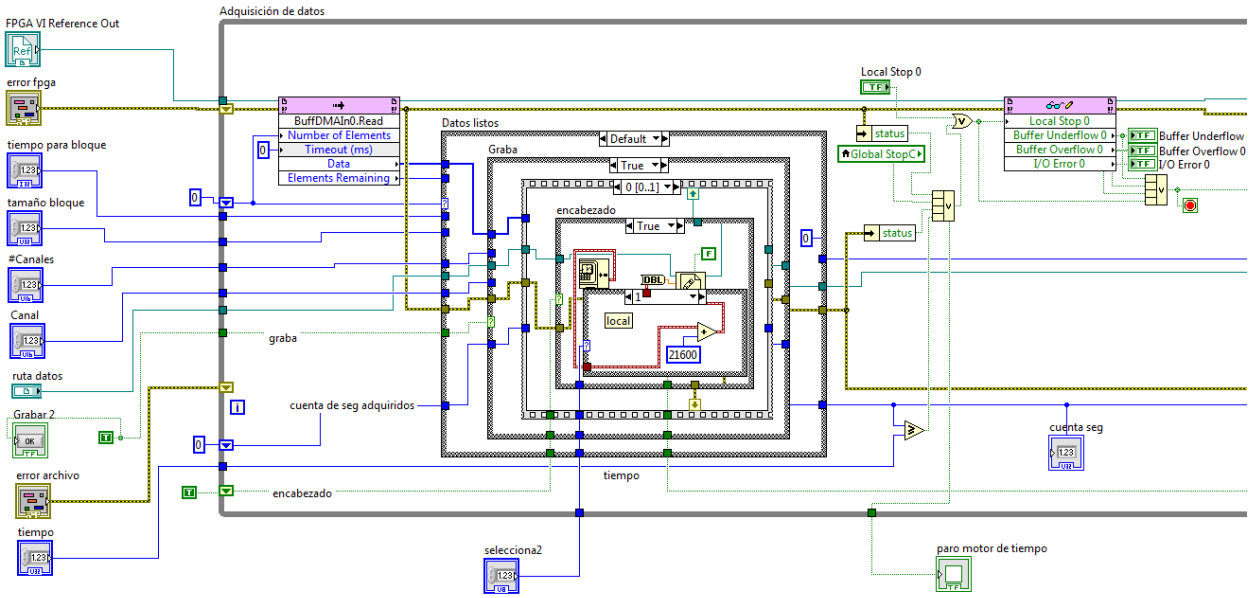


Figura B.15. Modo Continuo, lectura del *buffer* DMA y almacenamiento.

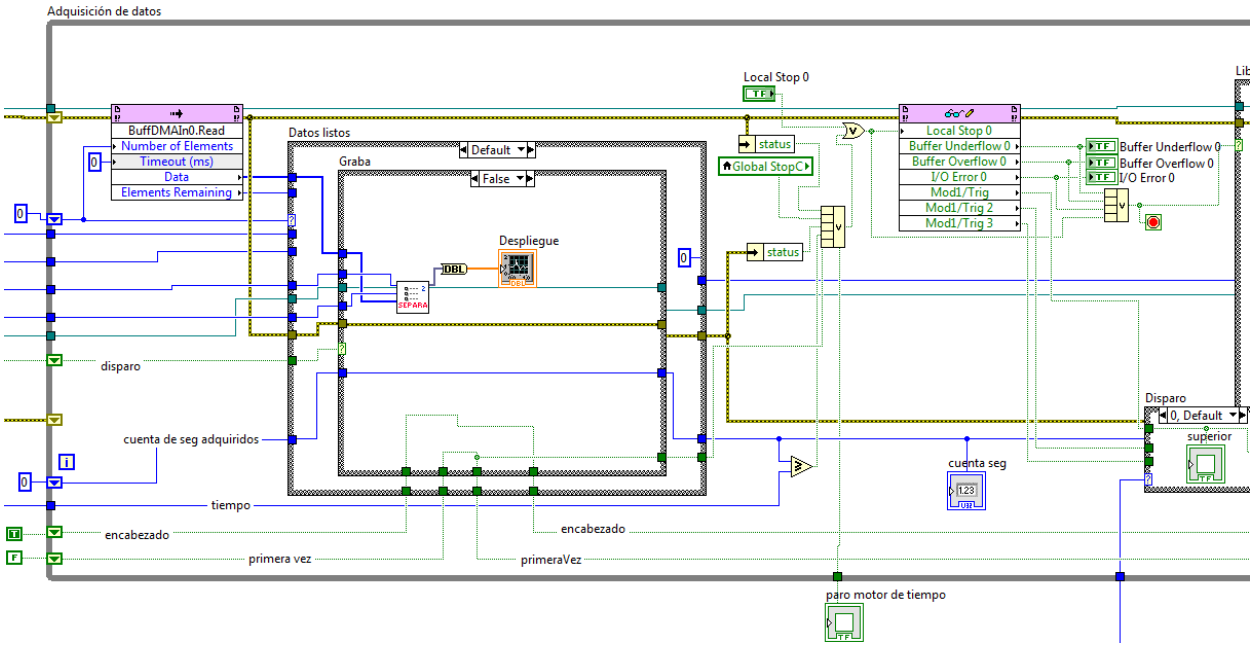


Figura B.16. Modo Disparo, despliegue de datos y monitoreo para disparo.

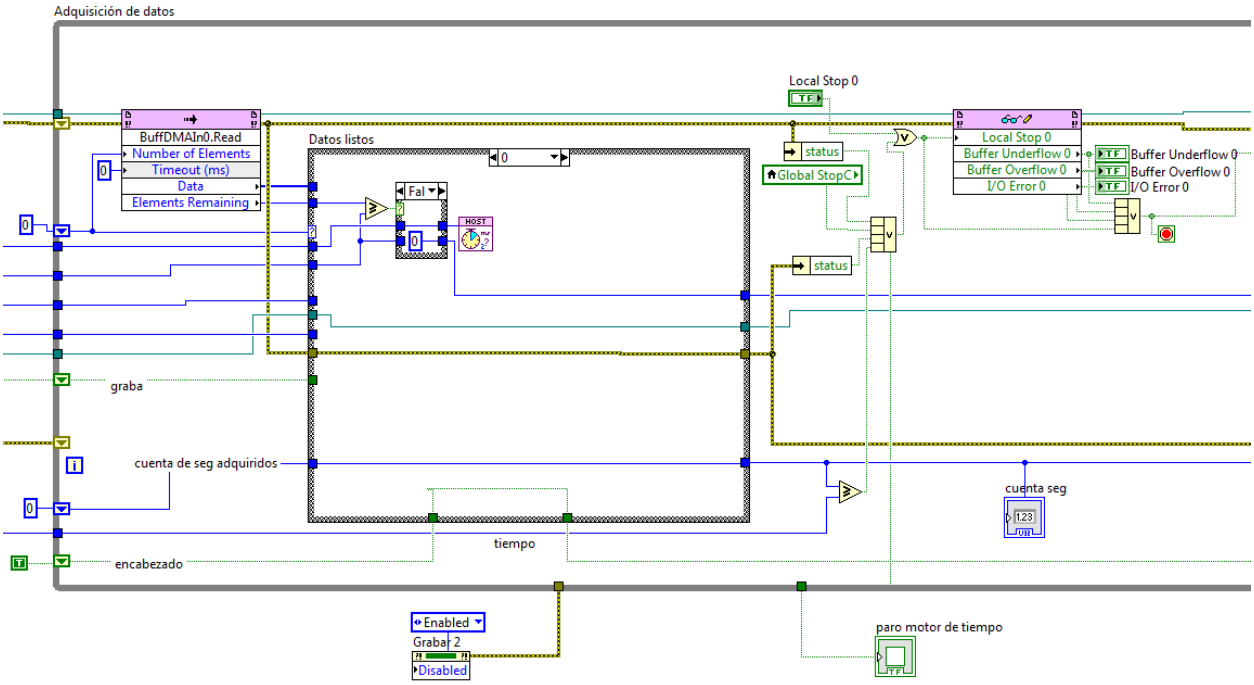


Figura B.17. Modo Software, verificación de datos disponibles en buffer.

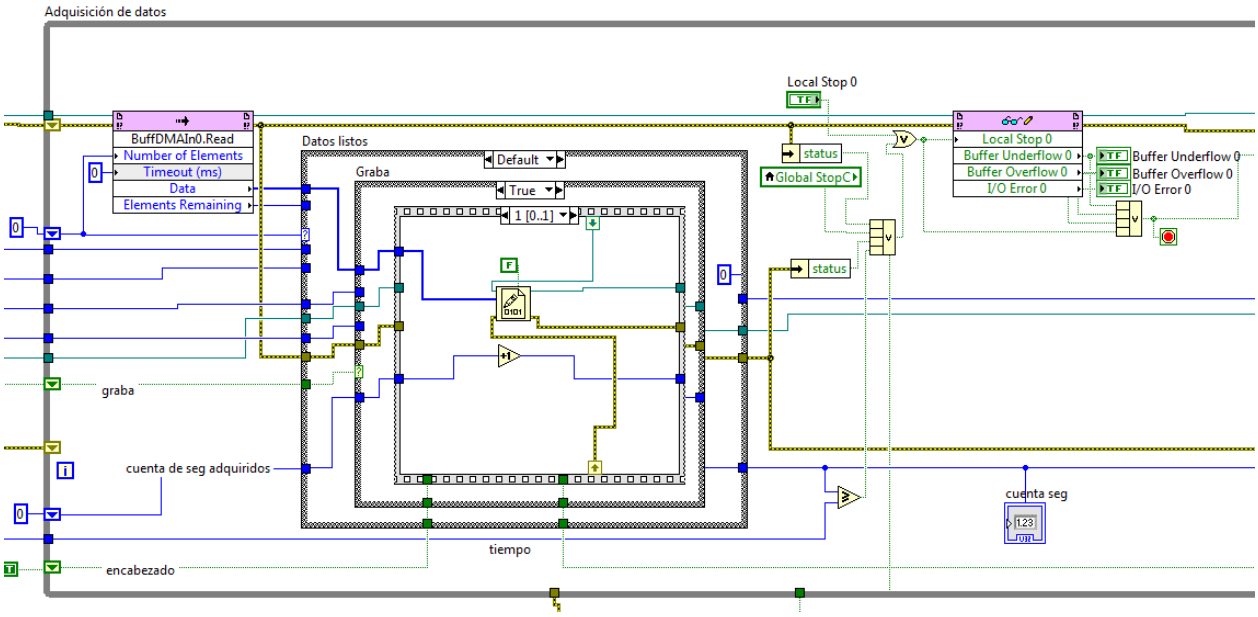


Figura B.18. Modo Temporizado, registro de segundos almacenados.

Código relacionado al FPGA

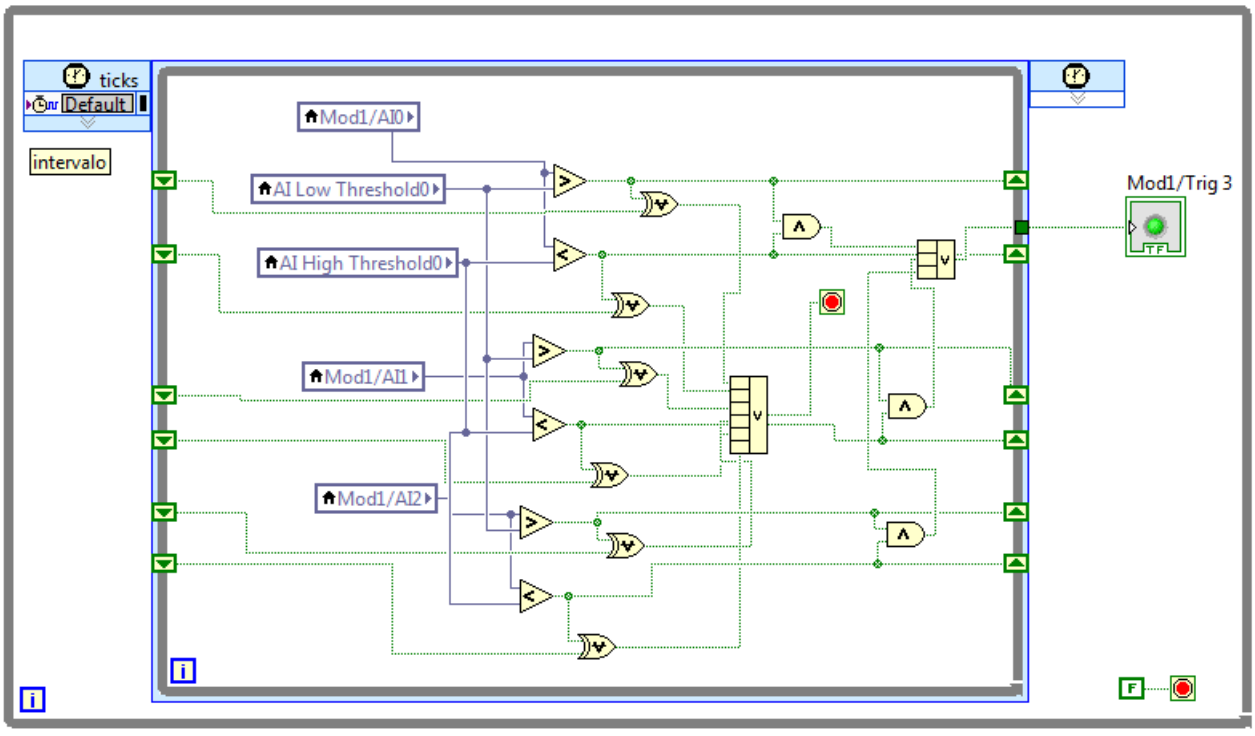


Figura B.19. VI del FPGA, disparo modo intervalo.

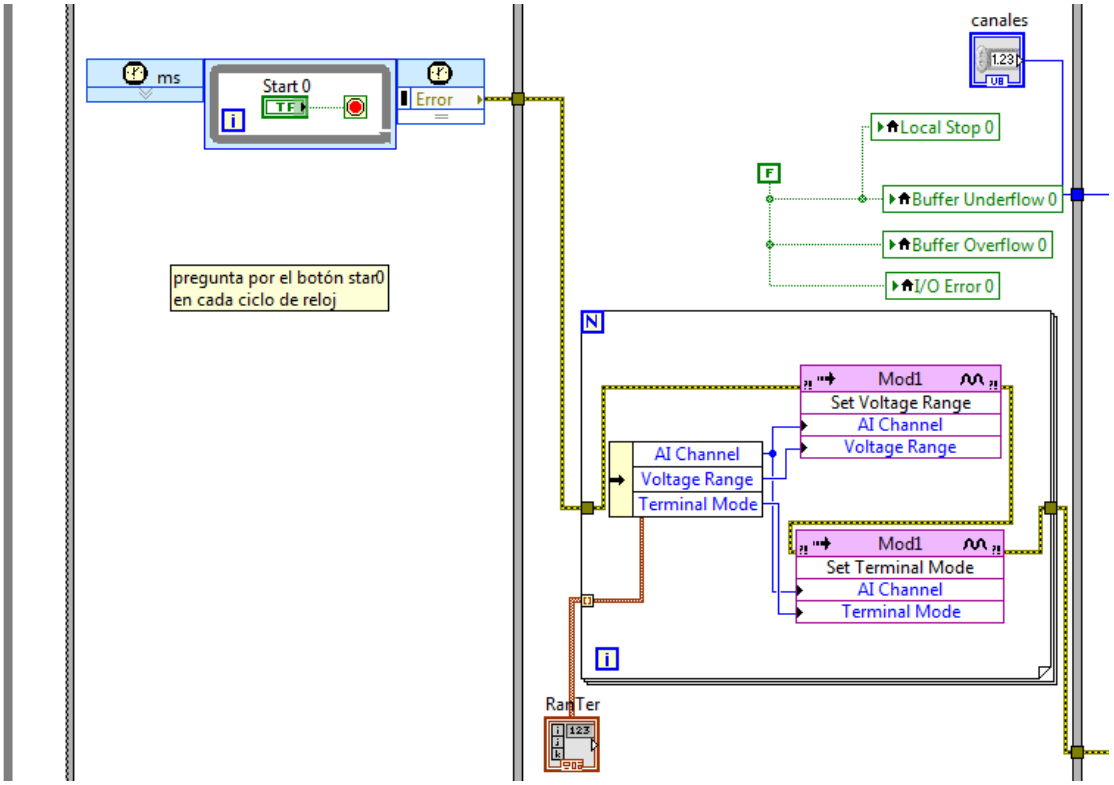


Figura B.20. VI del FPGA, lectura de instrucciones del MCU.

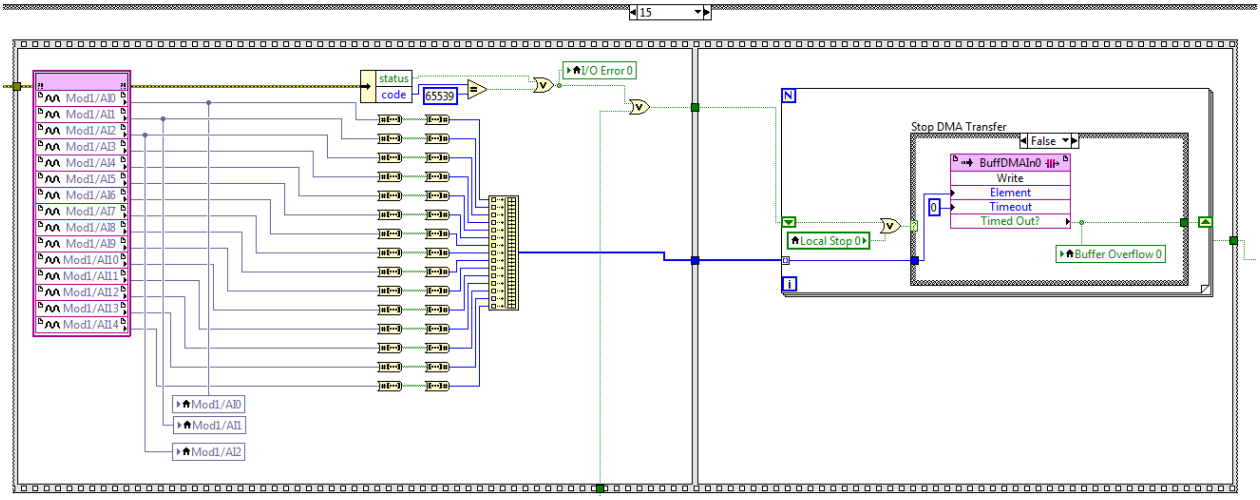


Figura B.21. VI del FPGA, petición de datos al NI 9205 y envío al MCU.

Bibliografía

Libros

1. Axelson J., Embedded Ethernet and Internet complete: designing and programming small devices for networking, primera edición, Lakeview Research, Madison, 2003.

2. Fairweather I., Brumfield A., LabVIEW: A developer's guide to Real World Integration, primera edición, CRC Press, Estados Unidos, 2012.

3. Hauck S., Dehon A., Reconfigurable Computing: The theory and practice of FPGA-based computing, primera edición, Elsevier, Estados Unidos, 2008.

4. Heath S., Embedded Systems Design, segunda edición, Newnes, Oxford, 2003.

5. Larsen R. W., LabVIEW for Egineers, primera edición, Pearson Education, Estados Unidos, 2011.

6. Mitra S. K., Digital Signal Processing: a computer based approach, segunda edición, McGraw-Hill, 2002.

7. Pallás Areny R., Adquisición y Distribución de Señales, primera edición, Marcombo, Barcelona, 1993.

8. Park J., Steve M., Practical Data Acquisition for Instrumentation and Control Systems, primera edición, Newnes, Perth, Inglaterra, 2003.

9. Shanley T., Anderson D., PCI System Architecture, cuarta edición, MindShare, Estados Unidos, 1999.

10. Travis J., Kring J., LabVIEW for Everyone: graphical programming made easy and fun, tercera edición, Prentice Hall, Estados Unidos, 2006.

11. Valdés Pérez F. E., Pallás Areny R., Microntroladores: fundamentos y aplicaciones con PIC, primera edición, Marcomobo, Barcelona, 2007.

Hojas de especificaciones

1. Freescale Semiconductor, Data Sheet MPC5200, Revisión 4, 2005.

2. Geospace Technologies, Geophones GS-11D, 2012.

3. National Instruments, Installation Instructions: CompactRIO Reconfigurable Embedded Chassis, 2009.

4. National Instruments, Operating instructions and specifications: CompactRIO NI cRIO-9012/9014, 2010.

5. National Instruments, Operating Instructions and specifications: NI 9205, 2008.

6. Xilinx, Virtex-5 Family Overview, Revisión 5, 2009.

7. Xilinx, Virtex-5 FPGA User Guide, Revisión 5.4, 2012.

8. Xilinx, Virtex-5 FPGA XtremeDSP Design Considerations, fifth revision, 2009.

Artículos

1. Gómez Rosas E. R., Mendoza García M. A., Santiago Cruz L., Ruiz Molina F. A., Desarrollo de un registrador de variables dinámicas y estáticas; útil en la instrumentación de edificios, puentes y vías elevadas, Memorias del congreso, Primer Congreso Iberoamericano de Instrumentación y Ciencias Aplicadas, 2013.

2. Microsoft, Volume and file size limits of FAT file systems, www.microsoft.com/technet/prodtechnol/winxppro/reskit/c13621675.mspx, 2004.

3. National Instruments, Field Wiring and Noise Considerations for Analog Signals, www.ni.com/white-paper/3344/en, 2011.

4. RhinoSoft, FTP Transfer Modes: ASCII vs Binary, www.rhinosoft.com/newsletter/NewsL2008-03-18.asp, 2008.

5. USB Implementers Forum, Universal Serial Bus Mass Storage Class Specification Overview, 2003.