Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Eléctrica

Gestión de Energía Eléctrica con controladores en tiempo real

Autor: Víctor García Carrascal

Tutor: Moisés San Martín Ojeda
Departamento de Ingeniería Eléctrica

En Valladolid, a 10 de junio de 2016

RESUMEN

Con el dispositivo NI myRIO-1900 se ha desarrollado un programa con LabVIEW para el control y la gestión del consumo de una red eléctrica monofásica.

NI myRIO-1900 se conecta a la red a través de unos transductores para medir los valores instantáneos de la tensión y la intensidad. Con los valores anteriores se opera para sacar la forma de onda de la tensión y de la intensidad, el valor eficaz de la tensión y de la intensidad, frecuencia, factor de potencia, potencia activa, potencia reactiva, potencia aparente, THD de la tensión y de la intensidad y el Cos (φ). Los valores se presentan al usuario gráfica y numéricamente.

Los valores anteriores se guardan en la "nube" para su visionado desde cualquier parte del mundo y que sean guardados con seguridad
El programa está hecho de tal manera que se puede el consumo mediante dos botones que controlan dos contactores.

PALBRAS CLAVE

Control, Gestión, Red Eléctrica, LabVIEW, NI myRIO-1900.

ÍNDICE

1. Objeto

1.1. Antecedentes

Actualmente las empresas, las industrias o cualquier consumidor de energía eléctrica buscan optimizar costes para ser más competitivos en el mercado. Para realizar ahorros en los costes se puede actuar sobre un gran número de parámetros, entre ellos el consumo de energía eléctrica.
Los analizadores de redes miden una variedad de variables eléctricas, con el principal objetivo de obtener el control y la gestión de una instalación, máquina, industria, etcétera, permitiendo optimizar al máximo los costes energéticos.
Estos equipos están diseñados para ser instalados de forma sencilla en cualquier instalación y para que su uso sea totalmente adaptable a cualquier tipo de medida requerida. Disponen de una memoria interna donde se guardan todos los parámetros deseados.
Existe una gran variedad de analizadores los cuales exportan o muestran los parámetros eléctricos directa o indirectamente a través de display y transmiten por comunicaciones todas las magnitudes eléctricas medidas y/o calculadas.

1.2. Objetivos

Como objetivo del presente trabajo de fin de grado, dentro del Grado en Ingeniería Eléctrica impartido en la Universidad de Valladolid es el desarrollo de un sistema de control y seguimiento de la energía mediante el uso de Labview para registrar y monitorizar las variables eléctricas de una red monofásica para su posterior análisis además de controlar una serie de relés para la función que desee el usuario mediante salidas del módulo NI myRIO-1900. Además se podrá acceder al registro histórico de las variables tomadas desde cualquier ordenador que tenga acceso a internet ya que estas variables se guardan periódicamente en una página web y se podrán leer si el ordenador tiene el VI correspondiente.

1.3. Justificación

Este Trabajo de Fin de Grado se realiza con el motivo de la creación de un analizador de redes que además permite controlar salidas mediante LabVIEW y asentar los conocimientos eléctricos adquiridos durante el Grado para su aplicación en este trabajo.

2. Marco teórico

2.1. NI LabVIEW.

2.1.1. ¿Qué es NI LabVIEW?

LabVIEW es un entorno de programación gráfica usado por miles de ingenieros e investigadores para desarrollar sistemas sofisticados de medida, pruebas y control usando íconos gráficos e intuitivos y cables que parecen un diagrama de flujo. Ofrece una integración incomparable con miles de dispositivos de *hardware* y brinda cientos de bibliotecas integradas para análisis avanzado y visualización de datos, todo para crear instrumentación virtual. La plataforma LabVIEW es escalable a través de múltiples objetivos y sistemas operativos, desde su introducción en 1986 se ha vuelto un líder en la industria.

2.1.2. Introducción.

Durante más de 20 años, NI LabVIEW se ha sido utilizado por millones de ingenieros y científicos para desarrollar test sofisticados y aplicaciones de medida y control. Además de que LabVIEW provee de una variada gama de características y herramientas de asistentes e interfaces de usuario configurables, se diferencia por ser un lenguaje de programación gráfico de propósito general (conocido como G), con su compilador asociado, su enlazador, y herramientas de depuración.

2.1.3. Historia sobre la búsqueda de un lenguaje de programación.

Para entender mejor el valor añadido de la programación gráfica de LabVIEW, es útil remontarse al primer lenguaje de programación de alto nivel. En los albores de la edad moderna de la computación a mediados de los 50, un reducido grupo de IBM decidió crear una alternativa práctica a la programación de la enorme unidad central IBM 704 (un supercomputador en su época) en lenguaje ensamblador, el más moderno disponible en aquel entonces. El resultado fue FORTRAN, un lenguaje de programación más legible cuyo propósito era acelerar el proceso de desarrollo.
La comunidad ingenieril fue, en principio, escéptica de que este método pudiese superar los programas desarrollados a mano en ensamblador, pero pronto se demostró que los programas hechos con FORTRAN se ejecutaban casi tan eficientemente como aquellos escritos en ensamblador. Al mismo tiempo, FORTRAN redujo el número de sentencias necesarias en un programa en un factor 20, por lo que es considerado a menudo el primer leguaje de desarrollo de alto nivel. No sorprende que FORTRAN ganase rápidamente la aceptación de la comunidad científica.
Cincuenta años más tarde, hay todavía importantes lecciones en esta anécdota.
Primero, durante más de 50 años, los ingenieros han buscado formas más fáciles y rápidas de solucionar sus problemas de programación. Después, los lenguajes de programación elegidos para traducir sus tareas han tendido hacia niveles mayores de abstracción. Estas lecciones ayudan a explicar la inmensa popularidad y la extensa adopción de G desde su aparición en 1986; G representa un lenguaje de programación de extremadamente alto nivel cuyo propósito es aumentar la productividad de sus usuarios ejecutándose a casi la misma velocidad que los lenguajes de programación de niveles inferiores como FORTRAN, C y C++.

2.1.4. LabVIEW: Programación gráfica de flujo de datos.

LabVIEW es diferente de la mayoría de lenguajes de propósito general principalmente en dos vertientes. Primero, la programación G se desarrolla cableando iconos gráficos en un diagrama que compila directamente a código máquina de modo que los procesadores del ordenador pueden ejecutarlo sin preocuparse por obtener problemas de compatibilidad.
Aunque se representa gráficamente en lugar de texto, G contiene los mismos conceptos de programación que se pueden encontrar en la mayoría de los lenguajes tradicionales. Por ejemplo, G incluye todas las construcciones estándar tales como tipos de datos, bucles, eventos, variables, recursividad y programación orientada a objetos.

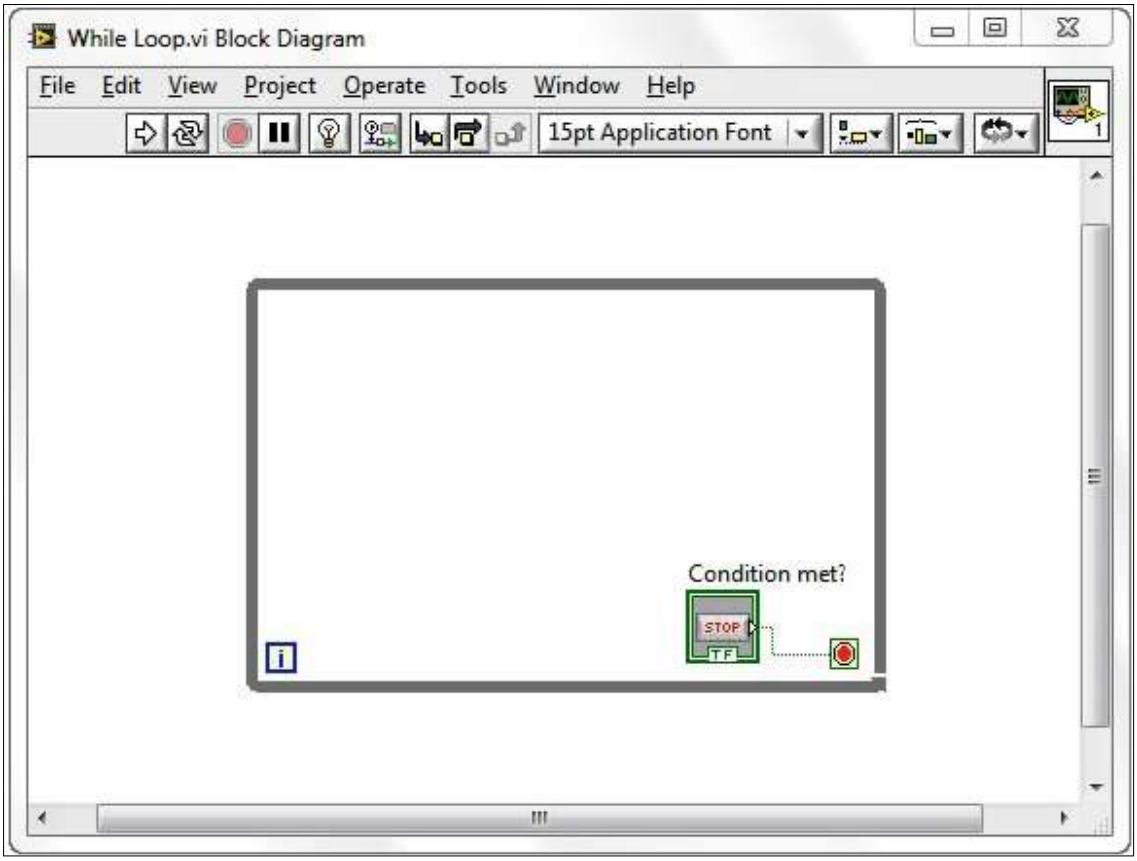


Figura 2.1. Un bucle *While* en G se representa por un lazo gráfico que se ejecuta hasta que se cumple una condición de parada.

La segunda diferencia principal es que el código G desarrollado en LabVIEW se ejecuta de acuerdo con las reglas del flujo de datos en lugar de la manera tradicional (en otros palabras, una serie secuencial de comandos para ser llevados a cabo) que se encuentra en la mayoría de los lenguajes de programación basados en texto como C y C++.
Los lenguajes de flujo de datos como G (también VEE de Agilent, Microsoft Visual y Apple Quartz Composer) promueven los datos como concepto principal detrás de cualquier programa. La ejecución de un datagrama es dirigida por el dato o dependiente del mismo. El flujo de datos entre los nodos del programa, líneas no secuenciales de texto determina el orden de ejecución.

Esta distinción puede ser menor a priori, pero el impacto es extraordinario ya que presenta rutas de datos entre partes del programa para ser el centro de atención del desarrollador. Los nodos en un programa de LabVIEW (en otras palabras, funciones y estructuras como bucles y subrutinas) tienen entradas, procesan datos y generan salidas. Una vez que todas las entradas de los nodos dados contienen un dato válido, el nodo ejecuta su lógica, produce datos de salida y pasa los datos al siguiente nodo en la secuencia del flujograma. Un nodo que recibe datos de otro, se puede ejecutar solo después de que el primero complete su ejecución.

2.1.5. Beneficios de la programación G.

- **Programación gráfica intuitiva.**

Como todo el mundo, los ingenieros y científicos aprenden observando y procesando imágenes sin necesidad de pensamiento consciente. Se denominan "pensadores visuales", ya que son especialmente adeptos a organizar información con procesamiento visual. En otras palabras, piensan mejor en imágenes. Esto se refuerza a menudo en facultades y universidades donde se anima a los estudiantes a modelar soluciones a problemas como diagramas de proceso. Sin embargo, la mayoría de los lenguajes de programación de propósito general requieren el empleo de cantidades ingentes de tiempo en aprender la sintaxis necesaria asociada con el lenguaje y mapear la estructura del mismo al problema a solventar. La programación gráfica con G provee de una experiencia más intuitiva.

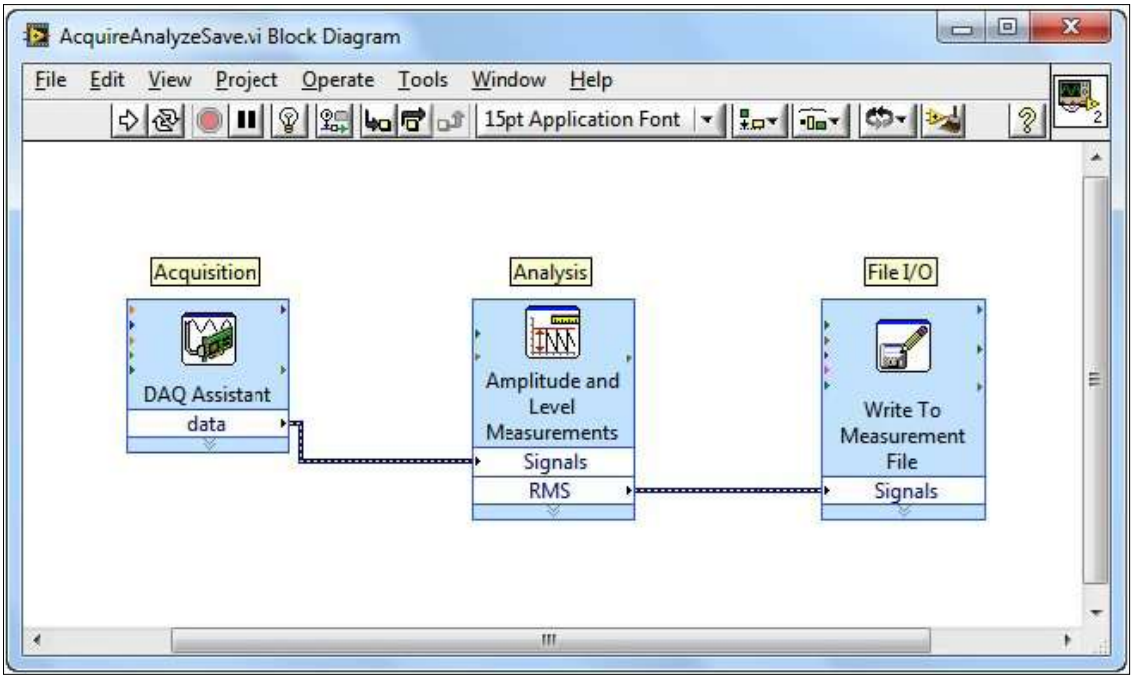


Figura 2.2. Los datos se originan en la función de adquisición y luego fluyen a las funciones de análisis y almacenamiento a través de los cables.

El código G es más sencillo de entender por ingenieros y científicos porque están familiarizados con la visualización y la modelización gráfica de procesos y tareas en términos de diagramas de bloque y flujogramas (que siguen también las reglas del flujo de datos).

Además, ya que los lenguajes de flujo de datos requieren basar la estructura del programa en el propio flujo uno se obliga a pensar en términos del problema que quiere solucionar.
Por ejemplo, y programa típico en G puede adquirir, en primer lugar, de varios canales de datos de temperatura, después pasarlos a una función de análisis y finalmente escribirlos a disco. En conjunto, el flujo de datos y los pasos involucrados en este programa son sencillos de comprender en el diagrama de LabVIEW.

- Herramientas de depuración interactiva

Puesto que el código G de LabVIEW es sencillo de comprender, las tareas más comunes de programación como el depurado, se vuelven más intuitivas también. Por ejemplo,
LabVIEW provee de herramientas de depuración únicas que se puede usar para observar el movimiento interactivo de los datos por los cables de un programa de LabVIEW y ver los valores que pasan de una función a otra (conocido en el entorno de LabVIEW como ejecución hightlighting).

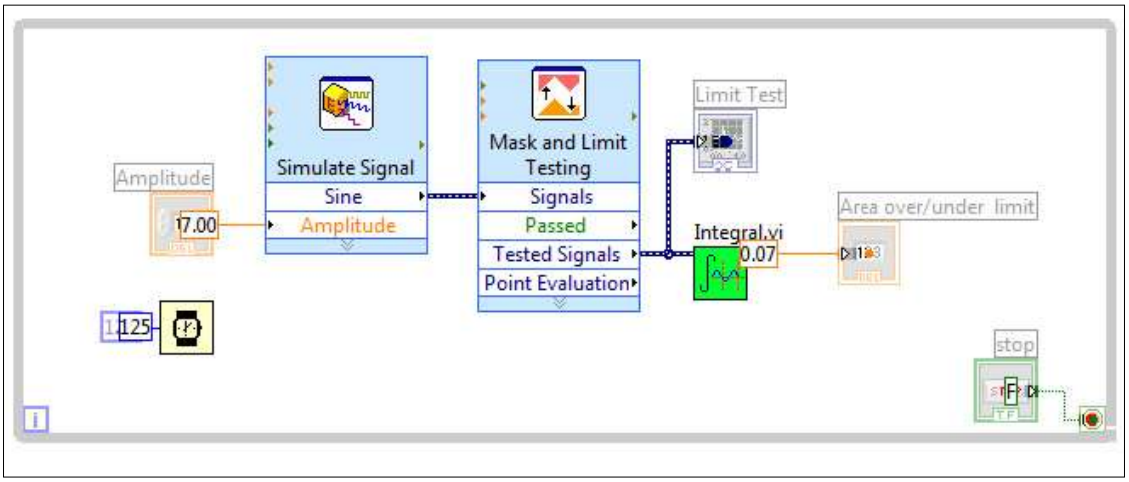


Figura 2.3. La ejecución Highlight provee de una forma intuitiva de entender el orden de ejecución del código G.

LabVIEW también ofrece herramientas de depuración para G comparables conaquellas que se encuentran en los lenguajes tradicionales. Estas herramienta, accesibles desdela barra de herramientas de un diagrama, incluyen sondas, puntos de parada y ejecución pasoa paso.

Figura 2.4. La barra de herramientas del diagrama de bloques ofrece acceso a las herramientas de depuración estándar como la ejecución paso a paso.

Con las herramientas de depuración de G, se puede sondar los datos en muchas partes del programa simultáneamente, pausar la ejecución, y ejecutar paso a paso una subrutina sin programación compleja. Aunque esto es posible en otros lenguajes de programación, es más fácil visualizar el estado del programa y la

relación entre las partes paralelas del código (que son comunes en G por su naturaleza gráfica).

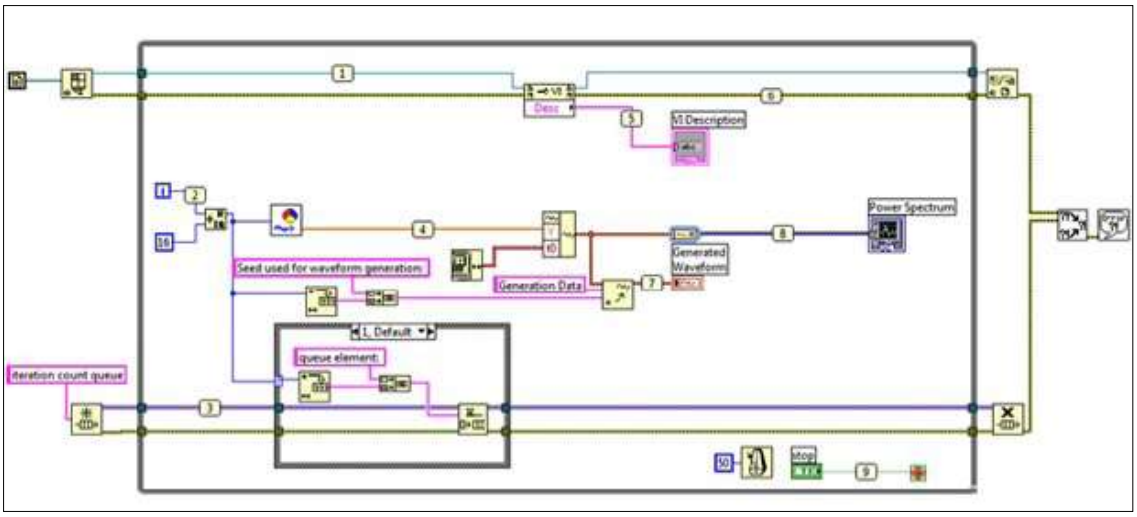


Figura 2.5. Las sondas son formas eficientes de ver los valores moviéndose por los cables de la aplicación, incluso para secciones paralelas del código.

Una de las más comunes herramientas de depuración usadas en LabVIEW se encuentra en el compilador. Mientras se está desarrollando un programa, el compilador continuamente busca errores y provee de información semántica y sintáctica de la aplicación.
Si existe un error, no se puede ejecutar el programa, sólo se ve un botón de ejecución roto en la barra de herramientas.

Figura 2.6. La flecha rota de ejecución provee de información inmediata de errores sintácticos en el código G.

Presionando el botón roto de ejecución, se abre una lista con los problemas que se han de arreglar. Una vez hecho, el compilador de LabVIEW transforma el programa en código máquina. Una vez compilado, el rendimiento de los programas de G es comparable al de aquellos basados en texto como C.

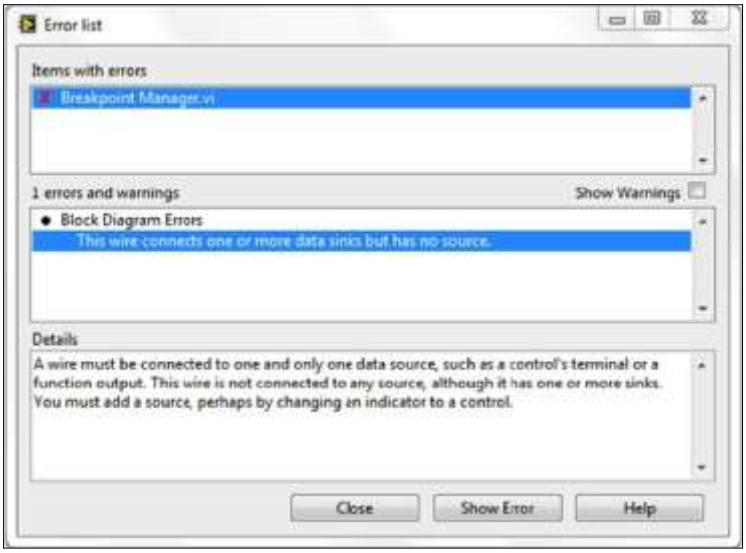


Figura 2.7. La lista de errores muestra una explicación detallada de cada error sintáctico en la jerarquía completa del código.

- Paralelismo y rendimiento automáticos

Los lenguajes de flujo de datos como LabVIEW permiten paralelizar automáticamente. A diferencia de los lenguajes secuenciales como C y C++, los programas gráficos contienen de forma inherente información sobre qué partes del código se pueden ejecutar en paralelo. Por ejemplo, un patrón común de diseño en G es el Productor/Consumidor, en el que dos bucles *while* se ejecutan independientemente: el primero es el responsable de la producción de datos y el segundo del procesamiento. En la ejecución en paralelo (posiblemente a frecuencias diferentes) los datos pasan entre los bucles usando colas, que son estructuras de datos estándar en los lenguajes de programación de propósito general.
El paralelismo es importante en la programación ya que desbloquea las ganancias de rendimiento relativas a los programas secuenciales debido a cambios recientes en el diseño de los procesadores. Durante más de 40 años, los fabricantes de chips incrementaron la velocidad del reloj del procesador para aumentar el rendimiento. Hoy, en cambio, el aumento de las velocidades de reloj para obtener mejoras en el rendimiento no es viable debido al consumo de energía y a las restricciones de disipación de calor. Como resultado, los fabricantes de chips han diseñado nuevas arquitecturas con múltiples núcleos de procesamiento en un único chip.

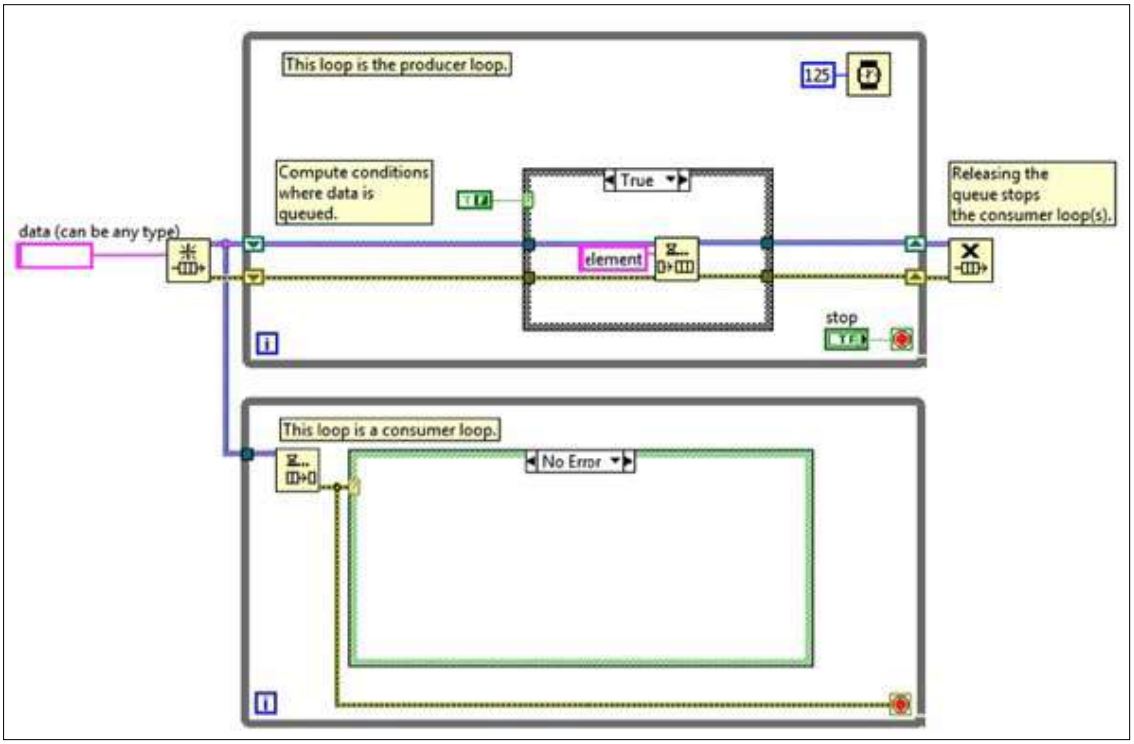


Figura 2.8. El patrón de diseño Productor/Consumidor de LabVIEW es a menudo usado para aumentar el rendimiento de las aplicaciones que requieren tareas paralelas.

Para sacar provecho al rendimiento disponible en procesadores multinúcleo, se ha de usar el multihilo en sus aplicaciones (en otras palabras, dividir las aplicaciones en secciones discretas que puedan ser ejecutadas de forma independiente). Si se emplea los tradicionales lenguajes basados en texto, debe crear y administrar hilos para implementar el paralelismo, un desafío de envergadura para programadores no expertos.

Por el contrario, la naturaleza paralela del código G hace a la multitarea y multihilo fáciles de implementar. El compilador trabaja continuamente para identificar secciones paralelas del código. Siempre que el código G tiene una rama en un cable o una secuencia paralela de nodos en un diagrama, el compilador intenta ejecutar el código en paralelo del conjunto de hilos administrados por LabVIEW. En términos de computación científica, esto se conoce como paralelismo implícito porque no se tiene que escribir el código con el propósito de la ejecución paralela, el lenguaje G se encarga de ello por su cuenta.

Más allá del multihilo en un sistema multinúcleo, G provee de mayor ejecución en paralelo extendiendo la programación gráfica a las FPGAs. Éstas son chips reprogramables de silicio de naturaleza paralela – con cada tarea de procesamiento independiente asignado a una sección del chip – pero sin estar limitadas por el número de núcleos disponibles. Como resultado, el rendimiento de una parte de la aplicación no se ve afectado al añadir más procesamiento.

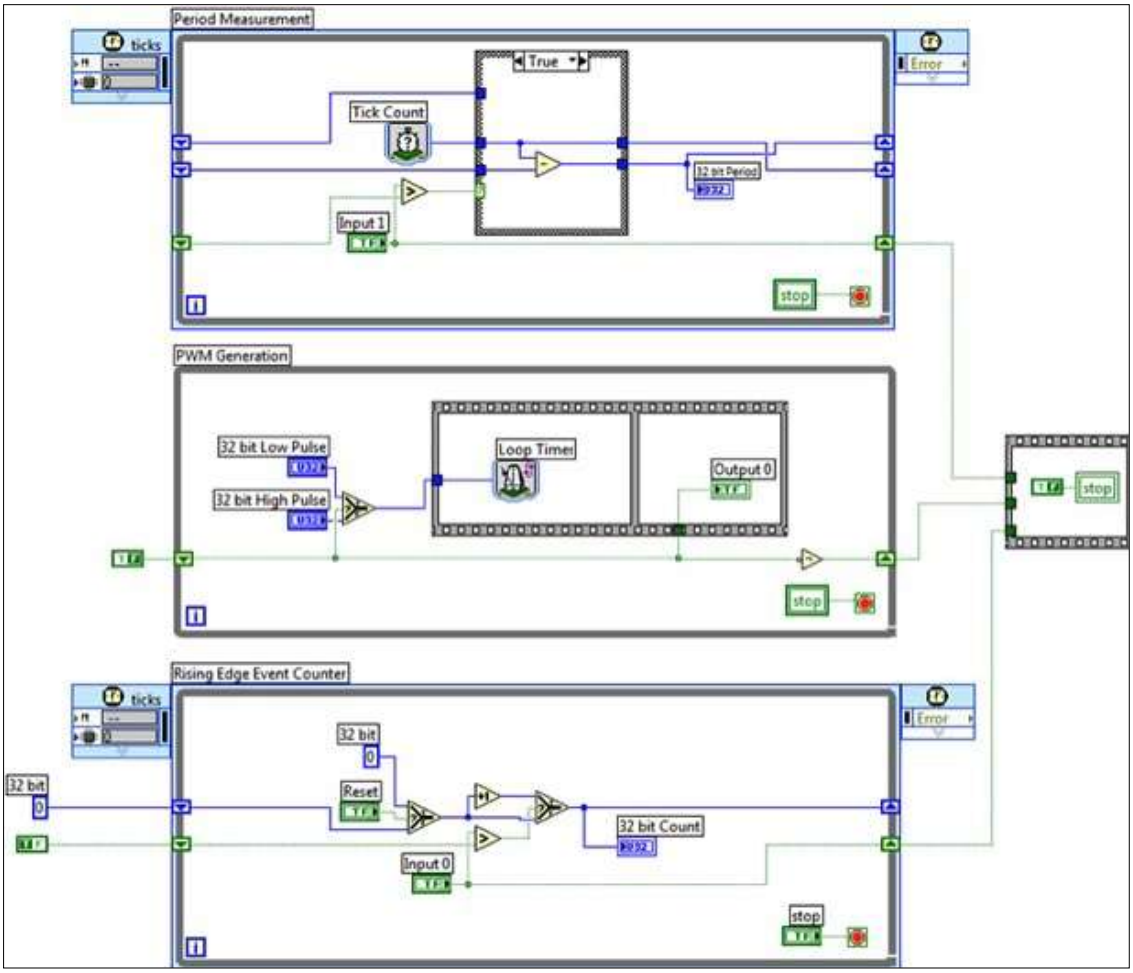


Figura 2.9. El código de LabVIEW FPGA con paralelismo genera caminos independientes en el chip de la FPGA.

Históricamente, la programación en FPGA eran el campo de sólo expertos formados con un profundo conocimiento de los lenguajes de diseño *hardware*. Aumentan los ingenieros no expertos en FPGA que quieren usar el *hardware* personalizado de la FPGA para retinas únicas de temporización y disparo, control ultra rápido, protocolos digitales, procesamiento digital de la señal (DSP), RF y comunicaciones y muchas otras aplicaciones que requieren *hardware* de alta velocidad, fiabilidad, personalización y alto determinismo. G encaja perfectamente con la programación de las FPGAs ya que claramente representa el paralelismo el flujo de datos y está creciendo rápidamente en popularidad como herramienta para desarrolladores que buscan procesamiento paralelo y ejecución determinista.

- Combinando G con otros lenguajes

Aunque el código G provee de una representación excelente para el paralelismo y elimina el requisito de los desarrolladores de entender y administrar el uso de memoria, no es necesariamente ideal para todas las tareas. En particular, las fórmulas matemáticas y las ecuaciones pueden ser representadas sucintamente en texto. Por esa razón, se puede usar LabVIEW para combinar la programación gráfica con varias formas de programación en texto.

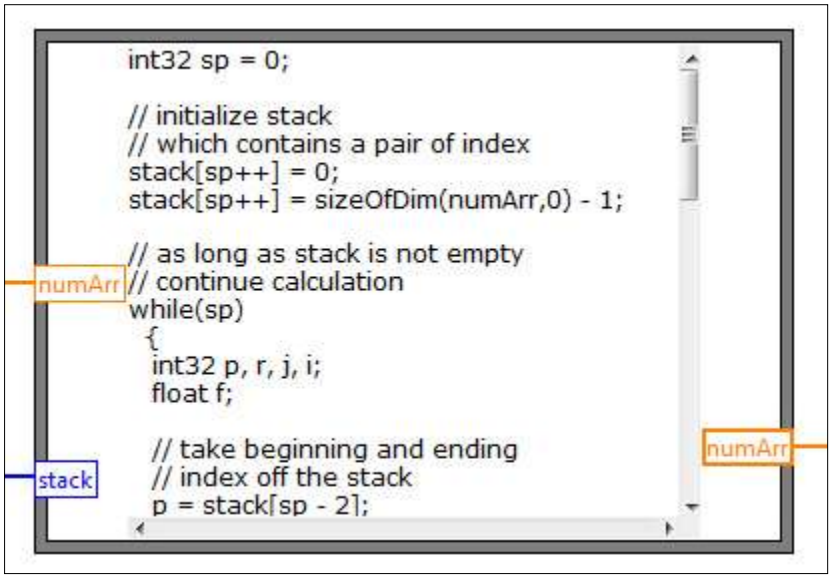


Figura 2.10. Formula Node usa sintaxis similar a C para representar expresiones matemáticas de una forma sucinta en formato de texto.

Trabajando con LabVIEW, se puede elegir un enfoque textual, gráfico o combinado. Por ejemplo, LabVIEW contiene el concepto de Formula Node que evalúa las formulas y expresiones matemáticas de un modo similar a C en el diagrama de bloques. Estas fórmulas matemáticas se pueden ejecutar codo con codo e integrarlas en el código gráfico de LabVIEW. Igualmente, el MathScript Node añade programación textual matemática en LabVIEW, generalmente compatible con la sintaxis de los archivos .m.

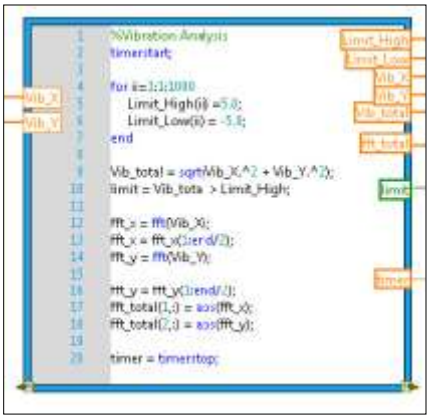


Figura 2.11. Con MathScript Node, usted puede crear o reutilizar archivos .m para el procesamiento de señales y el análisis de datos.

- Una manera mejor de solucionar problemas

LabVIEW y su lenguaje de programación gráfico de flujo de datos provee de unamejor manara de solucionar los problemas que las alternativas tradicionales de bajo nivel y laprueba está en su longevidad. Las claves diferenciadoras de la programación en G son elcódigo gráfico intuitivo que se crea y las reglas de movimiento de los datos que gobiernan laejecución que se combinan para ofrecer una experiencia de programación que expresa elpensamiento de los procesos de sus usuarios de forma más cercana que otros lenguajes. Apesar de que G es un lenguaje de alto nivel, se puede lograr rendimientos comparables a losde los lenguajes como C gracias al compilador de LabVIEW.

2.1.6. Sistemas operativos en tiempo real (SOTR).

Un sistema operativo de tiempo real es un sistema operativo que ha sido desarrollado para aplicaciones que interactúan activamente con un entorno con dinámica conocida en relación con sus entradas, salidas y restricciones temporales, para darle un correcto funcionamiento de acuerdo con los conceptos de predictibilidad, estabilidad, controlabilidad y alcanzabilidad. Como tal, se le exige corrección en sus respuestas bajo ciertas restricciones de tiempo. Si no las respeta, se dirá que el sistema ha fallado. Para garantizar el comportamiento correcto en el tiempo requerido se necesita que el sistema sea predecible.

Por lo regular Sistema Operativo de tiempo real suele tener la misma arquitectura que un Sistema Operativo convencional, pero su diferencia radica en que proporciona mayor prioridad a los elementos de control y procesamiento que son utilizados para ejecutar los procesos o tareas.

1. El SOTR debe ser multitarea y permisible.
2. Un SOTR debe poder asignar prioridades a las tareas.
3. El SOTR debe proporcionar medios de comunicación y sincronización entre tareas.
4. Un SOTR debe poder evitar el problema de inversión de prioridades.
5. El comportamiento temporal del SOTR debe ser conocido.

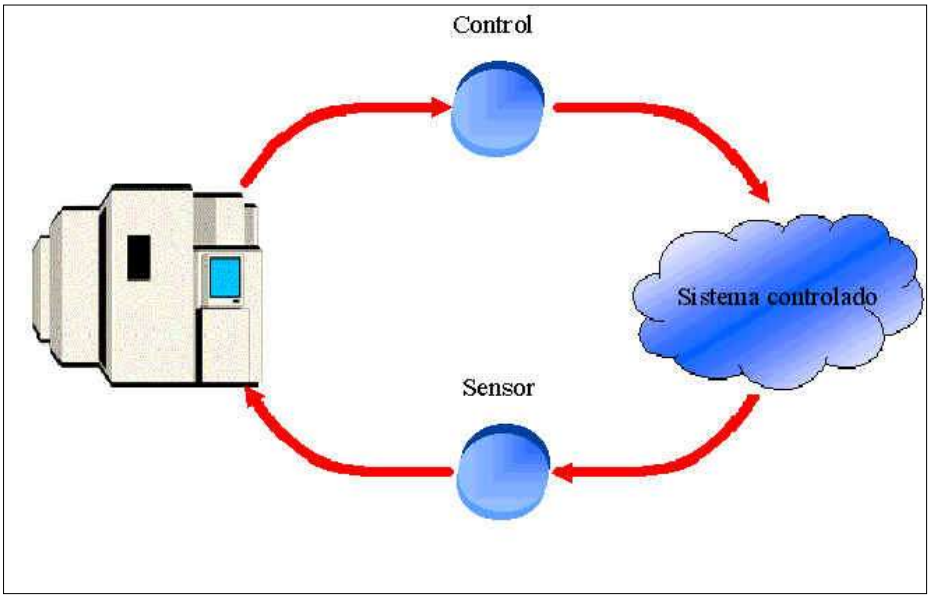


Figura 2.12. Esquema SOTR

- **Métodos de diseño:**

Cuando se elabora software de tiempo real se deben incorporar una alta calidad.
Al elaborar el software de tiempo real se presentan múltiples problemas como:

1. Representación de interrupciones y cambio de contexto.
2. Comunicación y sincronización de tareas.
3. Grandes variaciones en las tasas de datos.
4. Requisitos especiales para manejo de errores y recuperación de fallos.
5. Procesamiento asíncrono.

Para evitar muchos de los problemas que se presentan al elaborar software de tiempo real se han establecido algunos métodos como lo son:

1. Metodología de flujo de datos.
2. Metodología de estructura de datos.
3. Metodología orientada a los objetos.

- **Requisitos temporales:**

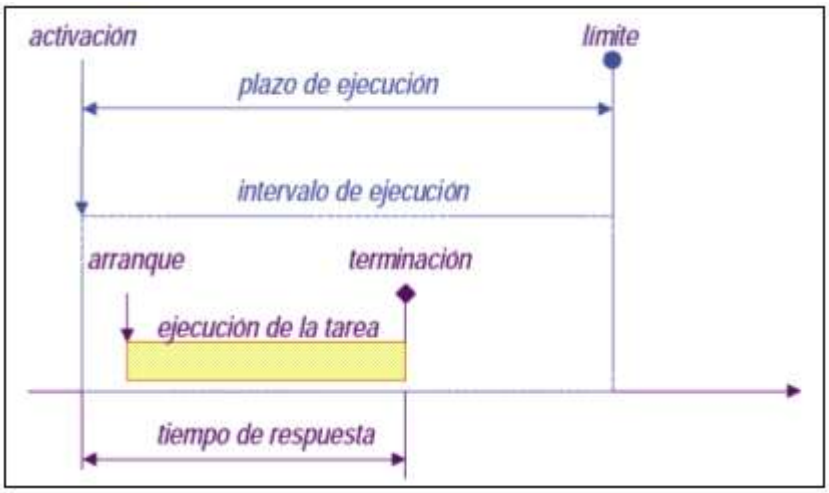


Figura 2.13. Requisitos temporales

Tiempo real estricto ***(hard real-time)***
– Todas las acciones deben ocurrir dentro del plazo especificado
Ejemplo: control de vuelo

Tiempo real flexible ***(soft real-time)***
– Se pueden perder plazos de vez en cuando
– El valor de la respuesta decrece con el tiempo
Ejemplo: adquisición de datos

Tiempo real firme ***(firm real-time)***
– Se pueden perder plazos ocasionalmente
– Una respuesta tardía no tiene valor
Ejemplo: sistemas multimedia

- **Estructura de un SOTR:**

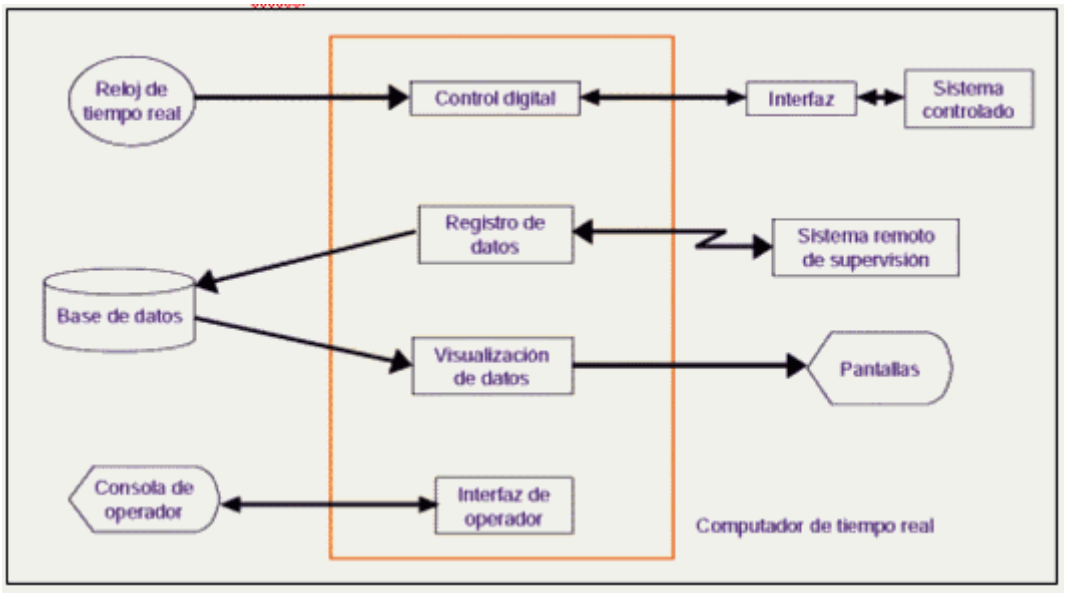


Figura 2.13. Diagrama flujo SOTR

Se caracterizan por presentar requisitos especiales en cinco áreas generales:

- Determinismo:
  Este término es una parte fundamental en estos sistemas, podría decirse que es una cualidad ya que es la capacidad de determinar con una alta probabilidad, cuanto es el tiempo que tarda una tarea en iniciar, es decir, que los STR necesitan que ciertas tareas se comiencen a ejecutar antes que otras.

- Sensibilidad:
  Este término se basa en el tiempo que tarda una tarea en ejecutarse. La sensibilidad se enfoca a 3 aspectos los cuales son:
  1. La cantidad de tiempo que tarda iniciar la ejecución de una interrupción
  2. La cantidad de tiempo que se necesita para realizar las tareas que pidió la interrupción.
  3. Los efectos de Interrupciones anidadas.

- Control del usuario:
  Todos los el usuario tienen un mejor control de todos los procesos que se ejecutan en el sistema esto es:
  1. Los procesos son capaces de especificar su prioridad
  2. Los procesos son capaces de especificar el manejo de memoria que requiere
  3. Los procesos especifican que derechos tiene sobre el sistema.

- Fiabilidad:
  1. En los STR la fiabilidad juega un papel muy importante, ya que el sistema no debe de presentar fallos, sino que más aun la calidad del servicio que ofrezca no debe de degradarse más allá de un límite especificado.

2. El sistema tiene que tener la capacidad de seguir funcionando aunque se presenten grandes catástrofes, o fallos mecánicos. Por lo general una degradación en el servicio en un STR lleva consecuencias catastróficas.

- Tolerancia a los fallos:
  Al hablar de tolerancia a los fallos nos estamos refiriendo a la capacidad de un sistema de conservar la máxima capacidad y los máximos datos posibles en caso de un problema grave que afecte a parte del sistema.
  Al referirnos a la tolerancia a los fallos estamos hablando también de la estabilidad ya que un sistema de tiempo real cuando le es imposible cumplir todos los plazos de ejecución de las tareas que tenía asignado en ese momento, el sistema cumple los plazos de las tareas más críticas y de mayor prioridad que hasta ese momento se estaban ejecutando.
  Entonces el sistema debe de fallar de manera que cuando se presente un problema en el sistema conserve gran parte de los datos y capacidades del sistema en la mayor medida posible.

- Características concretas:

2. Se presentan en entornos en donde deben ser aceptados y procesados una gran cantidad de sucesos, donde la mayoría de estos sucesos son externos al sistema computacional, con un tiempo de respuesta inmediato.
3. Pueden ser utilizados en muchos ámbitos entre los cuales están en control industrial, conmutación telefónica, control de vuelo, simulaciones en tiempo real., aplicaciones militares (entre otras).
4. Proporciona rápidos tiempos de respuesta.
5. Capacidad de procesar ráfagas de miles de interrupciones por segundo sin perder un solo suceso.
6. El proceso que tenga mayor prioridad expropia recursos.
7. La mayoría de los de procesos son estáticos.
8. La gestión de archivos se enfoca a velocidad de acceso que a la utilización eficiente del recurso.

- Son de tiempo compartido

2.1.6.1. Sistemas embebidos

Un sistema embebido es un sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real. Al contrario de lo que ocurre con los ordenadores de propósito general (como por ejemplo una computadora personal o PC) que están diseñados para cubrir un amplio rango de necesidades, los sistemas embebidos se diseñan para cubrir necesidades específicas. En un sistema embebido la mayoría de los componentes se encuentran incluidos en la placa basey muchas veces los dispositivos resultantes no tienen el aspecto de lo que se suele asociar a una computadora.

3. Controlador NI myRIO-1900

3.1 Descripción.

El controlador NI myRIO-1900 es un dispositivo de diseño embebido para estudiantes con tecnología de E/S reconfigurables (RIO) estándar en la industria, dispone de tres conectores de E/S, habilidades inalámbricas, un procesador ARM en tiempo real dual-core y un FPGA Xilinx personalizado. Por medio de sus componentes internos, acceso a software y una biblioteca con recursos y tutoriales, myRIO ofrece una herramienta accesible que ayuda a los estudiantes.

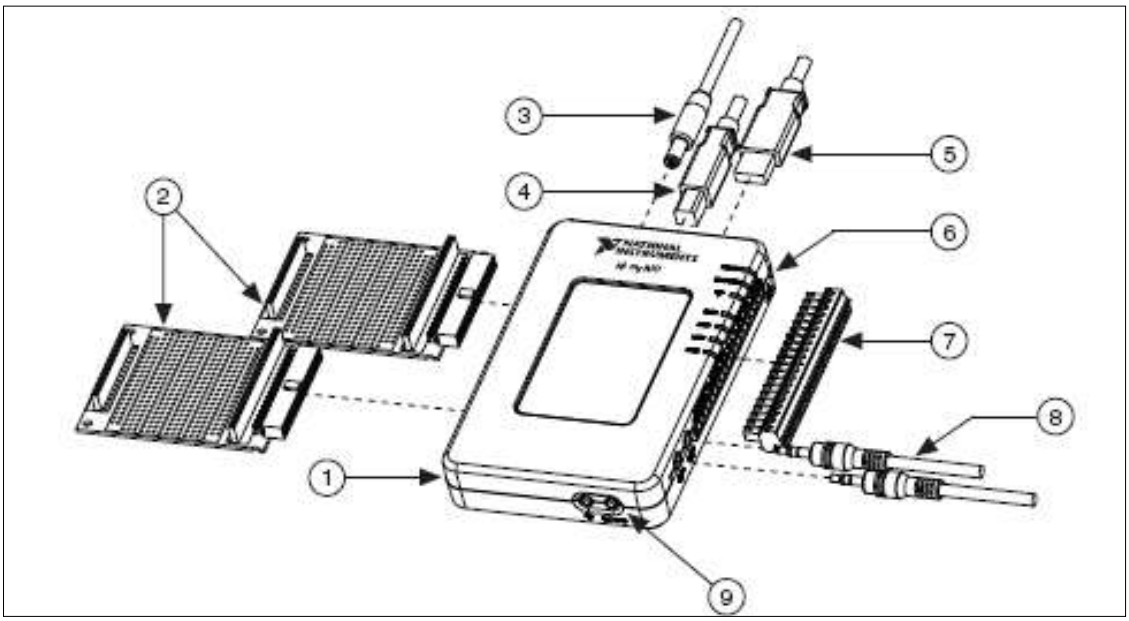


Figura 3.1. NI myRIO-1900.

1. NI myRIO-1900
2. Puertos de expansión
3. Cable de alimentación
4. Cable USB del dispositivo
5. Cable USB Host
6. LEDs
7. Mini puerto del sistema de terminales
8. Cables In/Out Audio
9. Botones

3.2 Descripción general del hardware.

El controlador NI myRIO-1900 proporciona entrada analógica (AI), salida analógicas (AO) entradas y salidas digitales (DIO), audio, alimentación de salida en el dispositivo embebido. La siguiente figura muestra la disposición y funciones de los componentes de NI myRIO-1900. Se conecta a un ordenador mediante USB o mediante red inalámbrica.

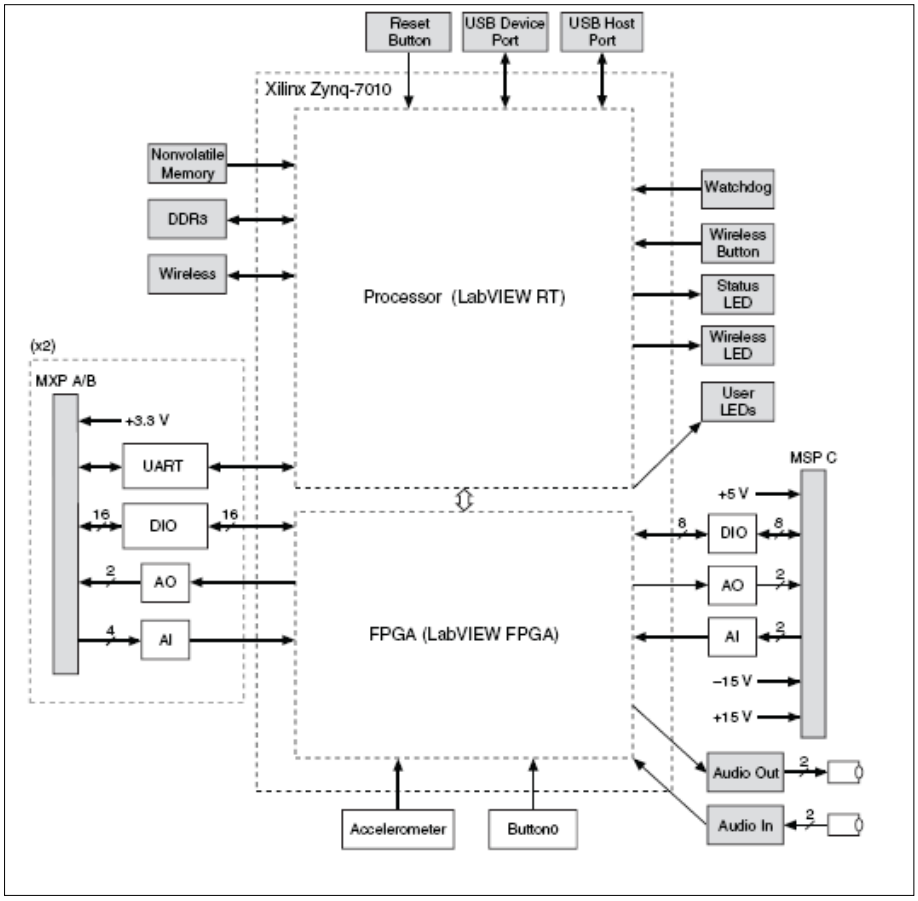


Figura 3.2. Diagrama de bloques del hardware de NI myRIO-1900.

3.3 Disposición de los pines.

En los puertos de expansión MXP A y B de NI-1900 myRIO los conectores llevan grupos idénticos de señales. Las señales se distinguen en el software por el nombre del conector, como en ConnectorA / ESD1 y ConnectOrb / ESD1.. La figura y la tabla muestran las señales de los conectores A y B en MXP. Algunos pines llevan funciones secundarias, así como funciones primarias.

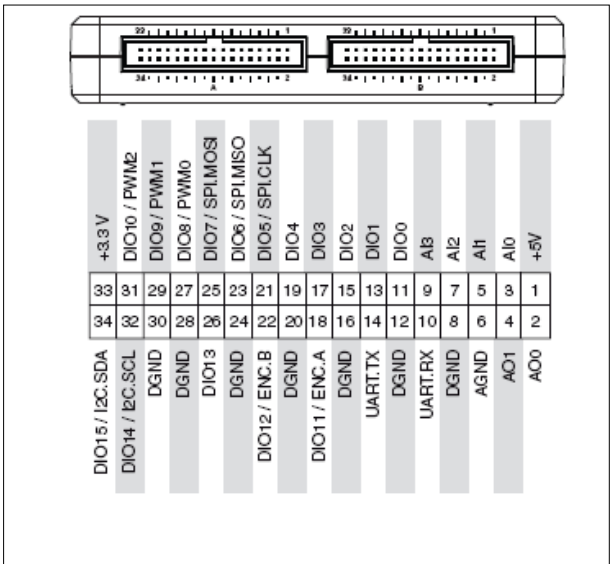


Figura 3.3. Las señales primarias / secundarias en los conectores A y B MXP.

Nombre de la señal	Referencia	Dirección	Descripción
+5V	DGND	Salida	+5V tensión de salida.
AI <0..3>	AGND	Entrada	Entradas de 0-5V.
AO <0..1>	AGND	Salida	Salidas de 0-5V.
AGND	N/A	N/A	Referencia para entradas y salidas analógicas.
+3.3V	DGND	Salida	+3.3V tensión de salida.
DIO <0..15>	DGND	Entrada o salida	Líneas digitales de propósito general con 3,3 V de salida, 3,3 V / entrada compatible con 5V.
UART.RX	DGND	Entrada	Entrada UART. Las líneas UART son eléctricamente idénticas a la DIO.
UART.TX	DGND	Salida	Salida UART. Las líneas UART son eléctricamente idénticas a la DIO.
DGND	N/A	N/A	De referencia para señales digitales, + 5V y 3,3 V.

Tabla 3.1. Descripciones de señales en los conectores A y B MXP.

La siguiente figura y la tabla muestran las señales en el mini puerto del sistema (MSP) conector C. Algunos pines llevan funciones secundarias, así como funciones primarias.

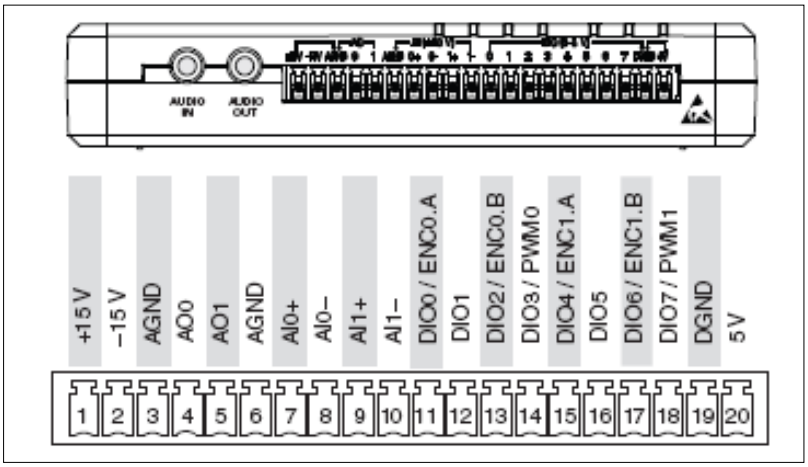


Figura 3.4. Las señales primario / secundario en conector C MSP.

Nombre de la señal	Referencia	Dirección	Descripción
+15V/-15V	AGND	Salida	+15 V/-15 V tensión de salida.
AI0+/AI0-; AI1+/AI1-	AGND	Entrada	± 10 V, diferenciales canales de entrada analógica.
AO <0..1>	AGND	Salida	± 10 V, diferenciales canales de salida analógica.
AGND	N/A	N/A	De referencia para señales digitales, +15V/-15V tensión de salida.
+5V	DGND	Salida	+5V tensión de salida.
DIO <0..7>	DGND	Entrada o salida	Líneas digitales de propósito general con 3,3 V de salida, 3,3V/entrada compatible con 5V.
DGND	N/A	N/A	De referencia para señales digitales, + 5V tensión de salida.

Tabla 3.2. Descripciones de señales en conector C MSP

Nombre de la señal	Referencia	Dirección	Descripción
Entrada audio	N/A	Entrada	La entrada derecha e izquierda de la salida de audio en el conector estéreo
Salida audio	N/A	Salida	Las salida derecha e izquierda de la salida de audio en el conector estéreo.

Tabla 3.3. Descripciones de señales en los conectores de audio.

3.4 Canales de entrada analógica

NI myRIO-1900 tiene canales de entrada analógica en los conectores A y B de los puertos de expansión (MXP), en el conector C del puerto Mini Sistema (MSP), y un conector de entrada de audio estéreo. Las entradas analógicas + se multiplexan a un convertidor único de analógico a digital (ADC) que las muestras de todos los canales. Los conectores A y B MXP tienen cuatro canales de una sola terminal de entrada
analógica por conector, AI0-EA3, que se puede utilizar para medir las señales V 0-5. El conector C MSP tiene dos canales de alta impedancia de entrada analógica diferencial, AI0 y EA1, que se pueden utilizar para medir señales de hasta ± 10 V. Las entradas de audio tienen ± 2,5 V rango de escala.

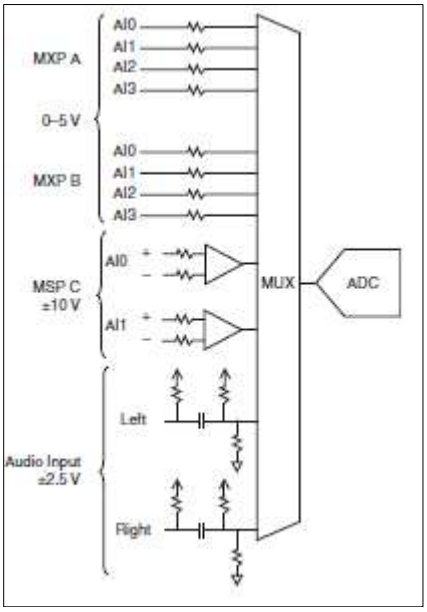


Figura 3.5 NI myRIO-1900. Circuito de entrada analógica.

3.5 Canales de salida analógica

NI myRIO-1900tiene canales de salida analógica en los conectores A y B de los puertos de expansión (MXP), en el conector C del puerto Mini Sistema (MSP), y un conector de entrada de audio estéreo. Cada canal de salida analógica tiene un convertidor dedicado-digital a analógico (DAC), para que todos puedan actualizarse simultáneamente. Los DAC para los canales de salida analógica son controlados por dos buses de comunicación en serie de la FPGA. Los conectores A y B MXP comparten un bus, y el conector C MSP y las salidas de audio comparten un segundo bús. Por lo tanto, la tasa máxima de actualización se especifica de manera agregada en la sección de salida analógica de las Especificaciones. Los conectores A y B MXP tienen dos canales de salida analógicos por conector, AO0 y SA1, que se puede utilizar para generar señales de 0-5 V. El conector C MSP tiene dos canales de salida analógicos, AO0 y SA1, que se pueden utilizar para generar señales de hasta ± 10 salidas de nivel de línea estéreo de salidas de audio.

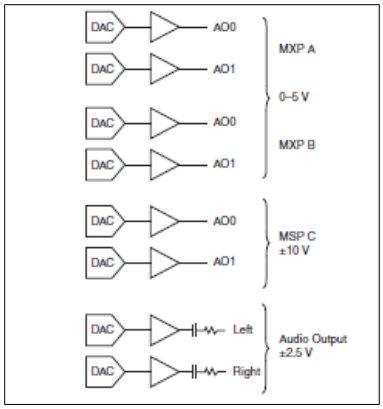


Figura 3.6 NI myRIO-1900. Circuito de salida analógica.

3.6 Acelerómetro

NI myRIO-1900 contiene un acelerómetro de tres ejes.

3.7 Conversión de valores de datos sin procesar a voltaje

Puede utilizar las siguientes ecuaciones para convertir valores de datos sin procesar a voltios:

$$V = \text{Valor de datos sin procesar} * \text{Peso LSB}$$
$$\text{Peso LSB} = \div \text{rango nominal } 2^{ADC\ Resolución}$$

donde el valor de los datos sin procesar es el valor devuelto por el nodo de la FPGA de E / S.
El peso LSB es el valor en voltios del incremento entre los valores de los datos.
El rango nominal es el valor absoluto en voltios del rango nominal total, de pico a pico del canal.
La resolución ADC es la resolución de la ADC en bits. (ADC Resolución = 12)

• Para los canales AI y AO en los conectores MXP:
Peso LSB = 5 V ÷ 212 = 1.221 mV
*Lectura máxima = 4095 * 1.221 mV = 4.999 V*

• Para los canales AI y AO en los conectores del MSP:
LSB Peso = 20 V ÷ 212 = 4.883 mV
*Lectura positiva máxima = 2.047 * 4.883 mV = 9.995 V*
*Lectura negativa máxima = -2048 mV * 4.883 = -10.000 V*

• Para audio de entrada / salida:
Peso LSB = 5 V ÷ 212 = 1.221 mV
*Lectura positiva máxima = 2.047 * 1.221 mV = 2.499 V*
*Lectura negativa máxima = -2048 mV * 1.221 = -2,500 V*

• Para el acelerómetro:
LSB = Peso 16 g ÷ 212 = 3.906 mg
*Lectura positiva máxima = 2.047 * 3.906 mg = 7,996 g*
*Lectura negativa máxima = -2,048 * 3,906 mg = -8,000 g*

3.8 Líneas DIO

NI myRIO-1900 tiene 3,3 V de propósito general en líneas DIO en los conectores MXP y MSP. Los conectores A y B MXP tienen 16 líneas DIO por conector. En los conectores MXP, cada línea DIO de 0 a 13 tiene una resistencia de pull-up de 40 kW a 3,3 V, y las líneas DIO 14 y 15 tienen resistencias pull-up de 2,2 kW a 3,3 V. MSP conector C tiene ocho líneas DIO. Cada línea MSP DIO tiene una resistencia de 40 kW desplegable a tierra. DGND es la referencia para todas las líneas DIO. Puede programar todas las líneas individualmente como entradas o salidas. Las funciones digitales secundarias incluyen Serial Peripheral Interface Bus (SPI), I2C, la modulación por ancho de pulso (PWM), y la entrada del codificador en cuadratura.

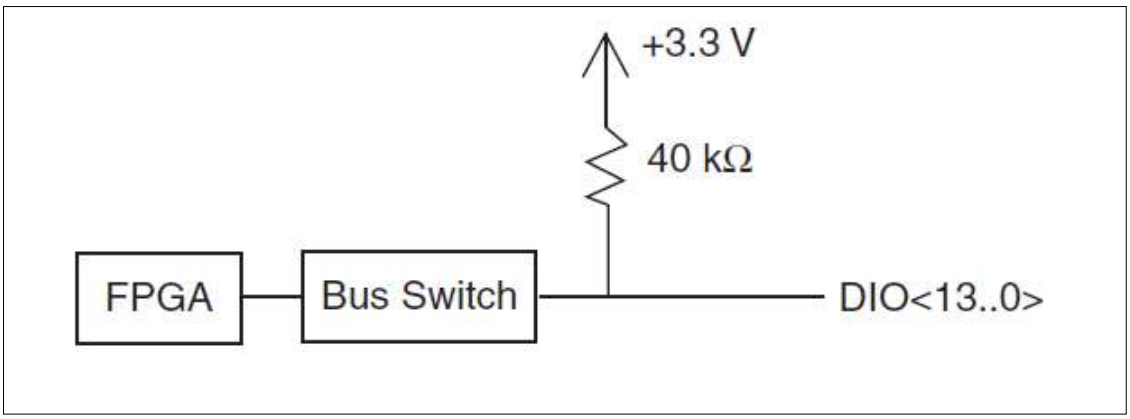


Figura 3.7 Líneas DIO <13..0> en conector A o B MXP

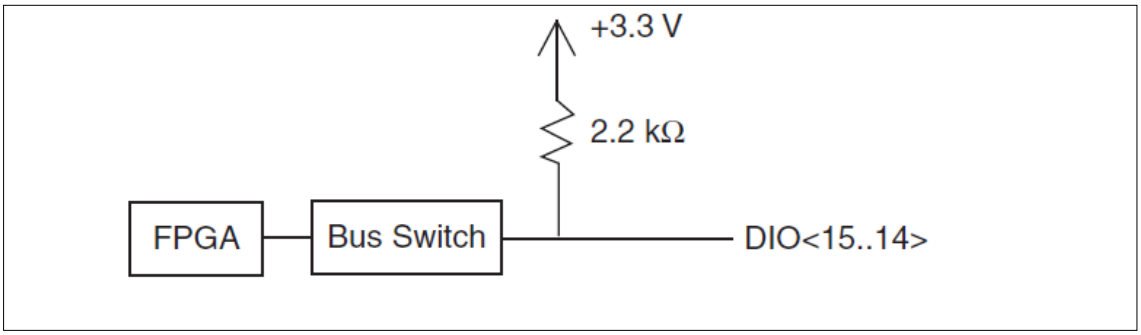


Figura 3.8 Líneas DIO <15..14> en conector A o B MXP

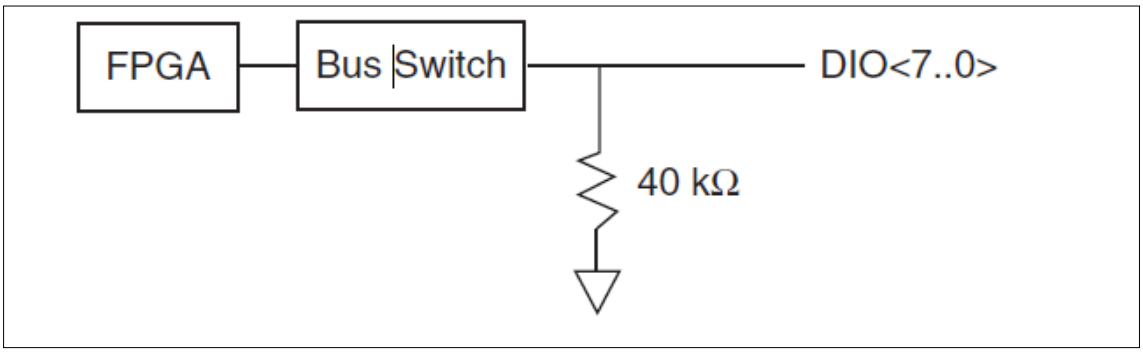


Figura 3.9 Líneas DIO <7..0> en conector C MSP

3.9 Líneas UART

NI myRIO-1900 tiene una línea de entrada UART y una línea de salida de transmisión UART en cada conector MXP. Las líneas UART son eléctricamente idénticas a las líneas DIO de 0 a 13 en los conectores MXP. Como en esas líneas, UART.RX y UART.TX tienen 40 kW de resistencias pull-up a 3,3 V.

3.10 Usando el botón Reset

Presionando y soltando el botón Reset reinicia el procesador y la FPGA. Si mantiene pulsado el botón de reinicio durante 5 segundos y luego lo suelta se reinicia el procesador y la FPGA y obliga a NI myRIO-1900 en modo seguro.
En modo seguro, NI myRIO-1900 lanza sólo los servicios necesarios para la actualización de la configuración e instalación de software. Cuando NI myRIO-1900 está en modo seguro, puede comunicarse con él mediante el uso de las líneas de UART conector MXP R. Necesita los siguientes elementos para comunicarse con el dispositivo Myrio sobre UART:

• (Por ejemplo, número de pieza de USB a TTL UART cable convertidor de serie TTL-232RG-VSW3V3-WE de FTD Chip)

• Programa de terminal de puerto serie configurado con los siguientes ajustes:
- 115.200 bits por segundo
- Ocho bits de datos
- Sin paridad
- Un bit de parada
- Sin control de flujo

3.11 Entendiendo la luz de los LED's

- LED de encendido

El LED de encendido se iluminará, y NI myRIO-1900 está encendido. Este LED indica que la fuente de alimentación conectada al dispositivo es adecuado.

- LED de estado

El LED de estado está apagado durante el funcionamiento normal. NI myRIO-1900 ejecuta una prueba de encendido (POST) cuando se enciende la alimentación del dispositivo. Durante la POST, el LED de encendido y LED de estado se encienden. Cuando el LED de estado se apaga, la prueba de encendido se ha completado. NI myRIO-1900 indica las condiciones de error específicas mediante el parpadeo del LED de estado de un cierto número de veces que cada pocos segundos, como se muestra en la Tabla 3.4:

Cantidad de parpadeos cada pocos segundos	Indicación
2	El dispositivo ha detectado un error en su software. esto por lo general se produce cuando se interrumpe un

	intento de actualizar el software. Vuelva a instalar software en el dispositivo.
3	El dispositivo está en modo seguro.
4	El dispositivo se queda sin memoria. Revisar su RT VI y comprobar el uso de memoria. Modificar el VI como sea necesario para resolver el problema de uso de memoria.
Continuamente parpadeando	El dispositivo ha detectado un error irrecuperable.

Tabla 3.4. Indicaciones LED de estado

- LED's 0-3

Puede utilizar LEDs 0-3 para ayudar a depurar la aplicación o recuperar fácilmente el estado de la aplicación. La lógica TRUE se convierte en un LED encenddoy una lógica falsa convierte un LED apagado.

3.12 Uso del puerto USB Host

El puerto host USB NI myRIO-1900 es compatible con las cámaras web que se ajustan a la clase de dispositivo de vídeo USB (UVC) de protocolo, así como cámaras de visión artificial que cumplen con el USB3 estándar de visión y son compatibles con versiones anteriores de USB 2.0. NI myRIO-1900 puerto host USB NI también es compatible con las cámaras Basler as USB3.
El puerto host USB NI también es compatible con unidades flash USB y adaptadores de USB a IDE formateados con sistemas de archivos FAT16 y FAT32. LabVIEW por lo general los mapas de dispositivos USB al / T, A / V, / W o / X de accionamiento, empezando por el / U si está disponible.

3.13 Dimensiones de NI myRIO-1900

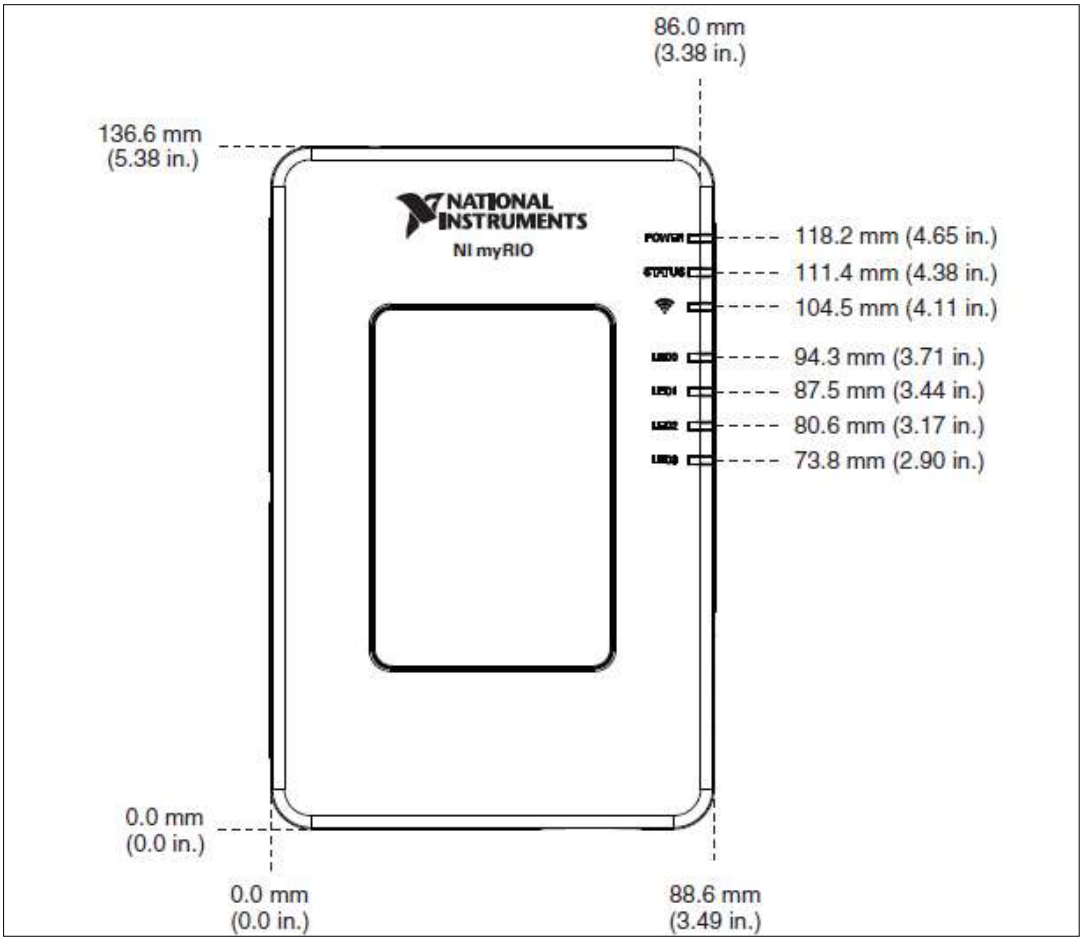


Figura 3.10 NI myRIO-1900 Frente

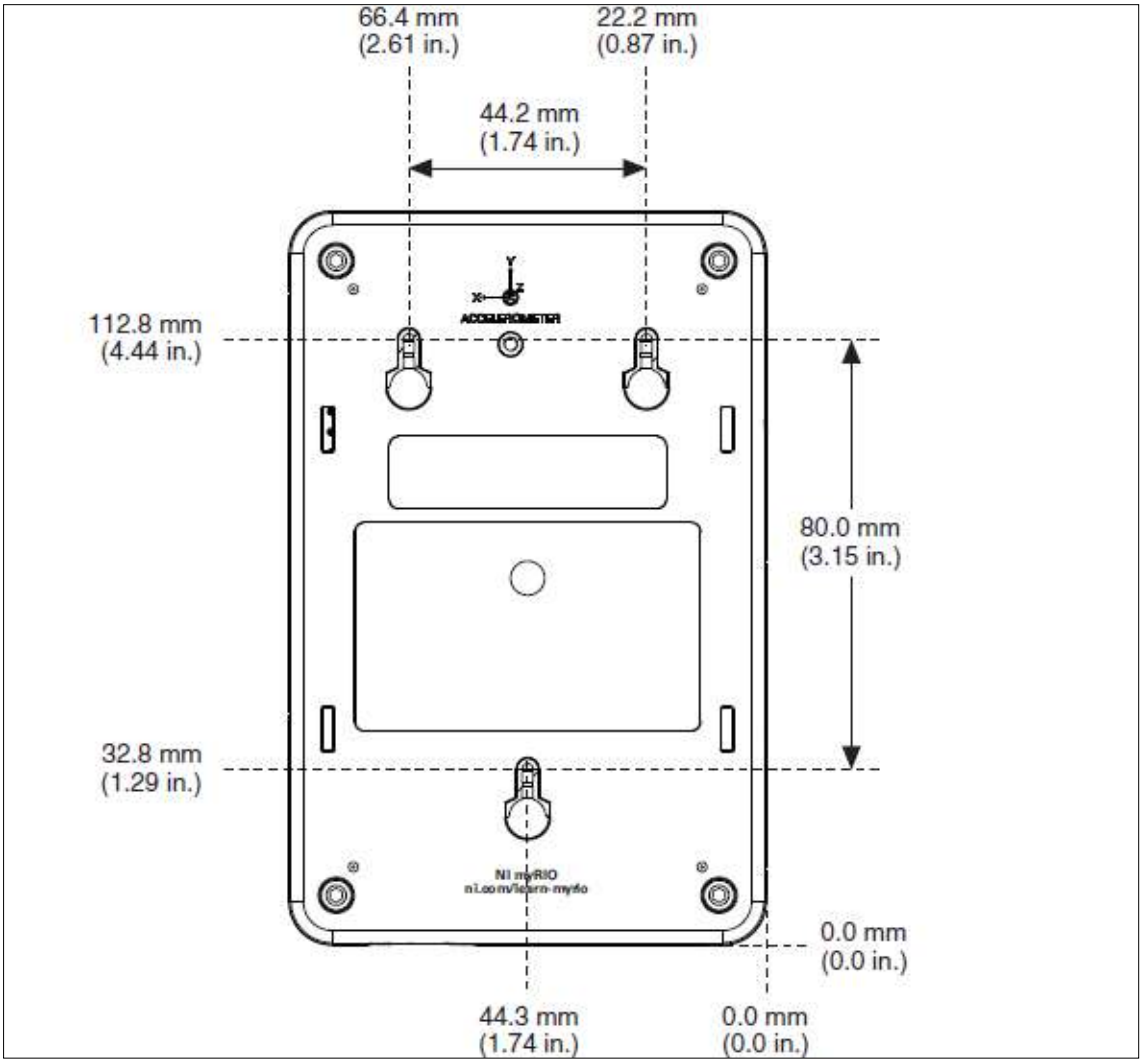


Figura 3.11 NI myRIO-1900 Parte trasera

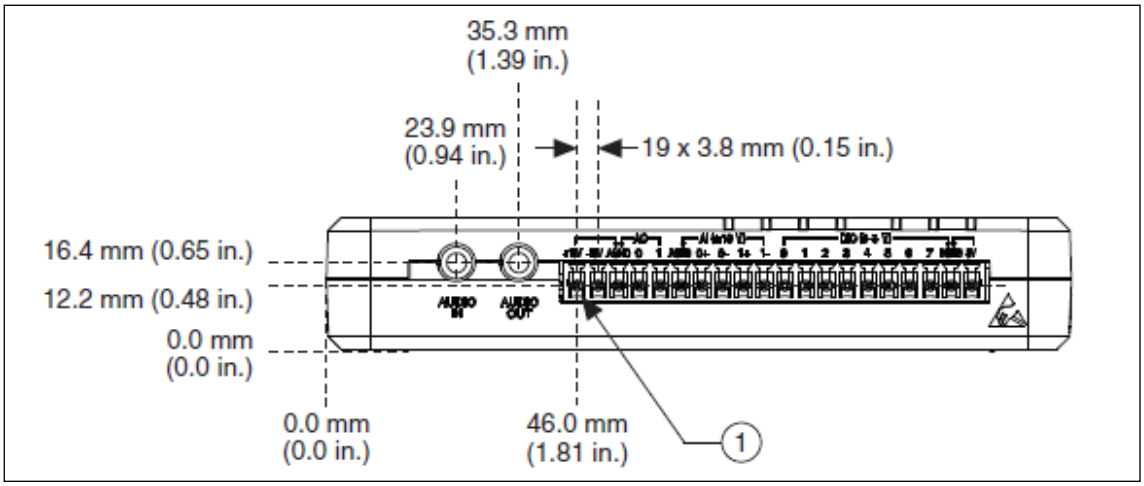


Figura 3.12 NI myRIO-1900 Lado MSP

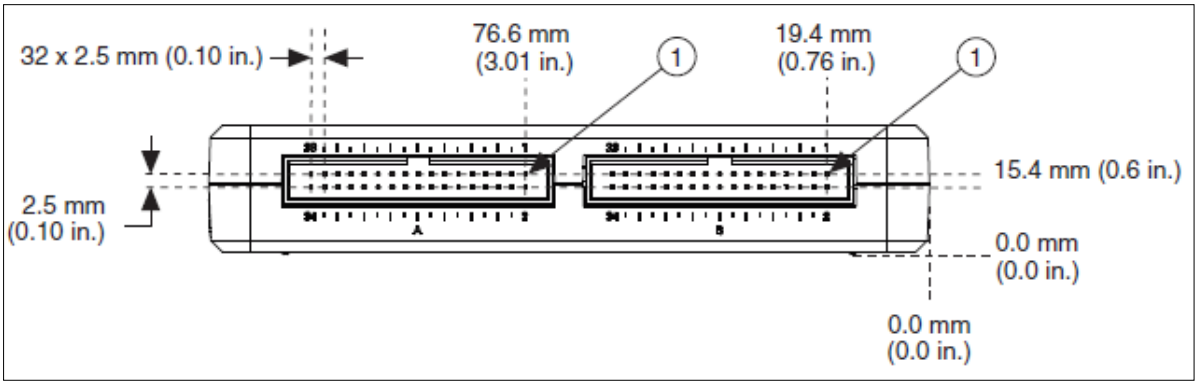


Figura 3.13 NI myRIO-1900 Lado MXP

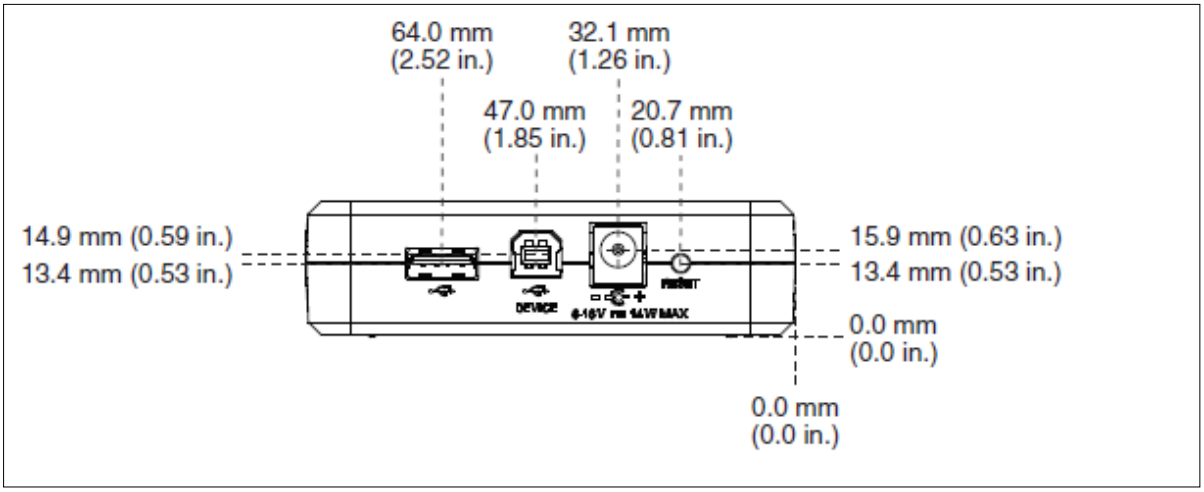


Figura 3.14 NI myRIO-1900 I/O

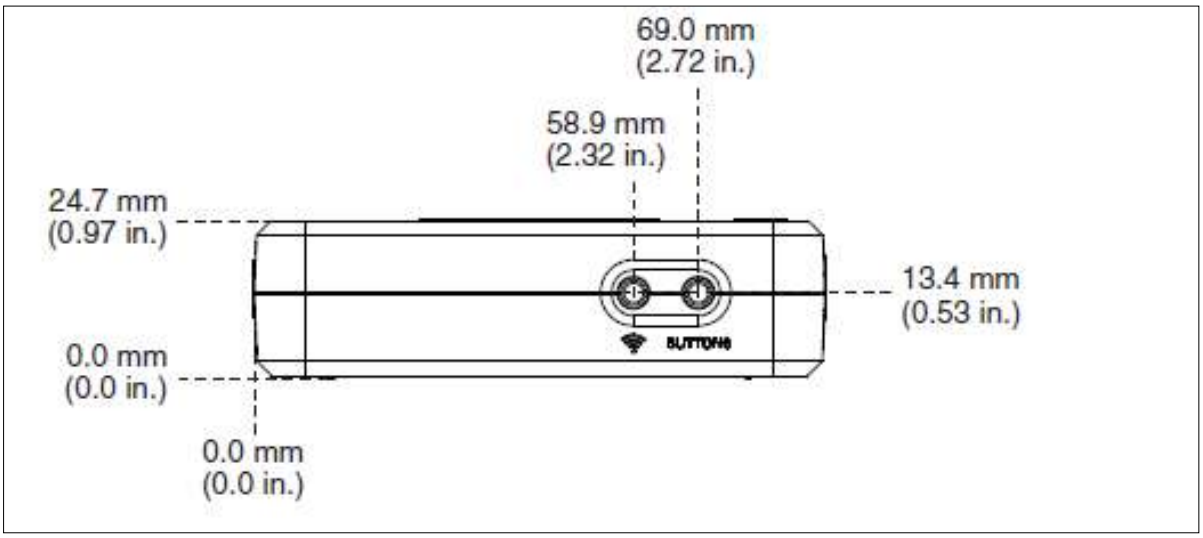


Figura 3.15 NI myRIO-1900 Usuario

3.14 Especificaciones

Procesador

Tipo de procesador.....Xilinx Z-7010
Velocidad del procesador.....667 MHz
Núcleos del procesador.....2

Memoria

Memoria no volátil.....256 MB
Memoria DDR3.....512 MB
Frecuencia de reloj DDR3.....533 MHz
Anchura del bus de datos DDR3.....16 bits

FPGA

Tipo de FPGA.....Xilinx Z-7010

Características inalámbricas

Modo de radio.....IEEE 802.11 b,g,n
Banda de frecuencia.....ISM 2.4 GHz
Ancho de banda.....20 MHz
Canales..... USA 1-11, International 1-13
TX power..... +10 dBm max (10 mW)
Rango en el exterior..... Hasta 150 m
Directividad de la antena.....Omnidireccional
Seguridad.....WPA, WPA2, WPA2-Enterprise

Puertos USB

Puerto host USB..... USB 2.0 Hi-Speed
Puerto dispositivo USB..... USB 2.0 Hi-Speed

Entrada analógica

Tasa de muestreo..... 500 kS/s
Resolución..... 12 bits
Protección contra sobretensiones..... ±16 V

Conectores MXP

- Configuración..... 4 canales por conector
- Impedancia de entrada
    1. >500 kΩ a 500 kS/s
    2. 1 MΩ Potencia encendido
    3. 4.7 kΩ Potencia apagado
- Fuente de impedancia recomendada..... 3 kΩ o menor
- Rango nominal ..... 0 V a +5 V
- Precisión absoluta..... ±50 mV
- Ancho de banda..... >300 kHz

Conectores MSP

- Configuración.....Dos canales diferenciales
- Impedancia de entrada
    1. Hasta 100 nA fuga de potencia de encendido
    2. 4.7 kΩ Potencia apagado

- Rango nominal ..... ±10 V
- Voltaje de trabajo..... ±10 V of AGND
- Precisión absoluta..... ±200 m
- Ancho de banda..... 20 kHz mínimo, >50 kHz normal
-

Entrada de audio

- Configuración..... Una entrada estéreo
- Impedancia de entrada..... 10 kΩ en DC
- Rango nominal..... ±2.5 V
- Ancho de banda..... 2 Hz a >20 kHz

Salida analógica

Tasas de actualización máxima

- Todos los canales AO en los conectores MXP.....345 kS/s
- Todos los canales AO en los conectores MSP y en los canales de salida de audio.....345 kS/s

Resolución.....12 bits
Protección contra sobrecarga..... .....±16 V
Voltaje de encendido.....0 V Después de la inicialización de la FPGA

Conector MXP

- Configuración.....Dos canales por conector
- Precisión absoluta.....50 Mv
- Corriente.....3 mA

Conector MSP

- Configuración .....Dos canales por conector
- Rango .....±10 V
- Precisión absoluta.....±200 mV
- Corriente .....2 mA

Salida de audio

- Configuración .....Una salida de audio
- Impedancia de salida .....100 Ω en serie con 22 µF
- Ancho de banda
    1. 70 Hz a >50 kHz dentro de carga de 32 Ω;
    2. 2 Hz a >50 kHz dentro de carga de alta impedancia

Digital I/O

Número de líneas:

- Conectores MXP..... 2 puertos de 16 líneas DIO (un puerto por conector); una línea de UART.RX y uno UART.TX por conector
- Conector MSP .....1 puerto de 8 líneas DIO

Control de dirección..... Cada línea DIO es individualmente programable como entrada o salida
Nivel lógico.....5 V entrada; 3.3 V salida
Niveles lógicos de entrada

- Entrada de baja tensión, VIL ..... 0 V min; 0.8 V max

- Entrada de alta tensión, VIH ..... 2.0 V min; 5.25 V max

Niveles lógicos de salida

- Salida de alta tensión, VOH sourcing 4 mA ..... 2.4 V min; 3.465 V max
- Salida de baja tensión, VOL sinking 4 mA ..... 0 V min; 0.4 V max

Ancho de pulso mínimo..... 20 ns

Frecuencias máximas para funciones digitales secundarias

- SPI .....4 MHz
- PWM..... 100 kHz
- Cuandratura de entrada del encoder .....100 kHz
- I^2C..... 400 kHz

Líneas UART

- Velocidad de transmisión máxima..... 230,400 bps
- Bits de datos..... 5, 6, 7, 8
- Stop bits..... 1, 2
- Paridad..... Impar, par, marca, Espacio
- Flow control.....XON/XOFF

Acelerómetro

Número de ejes..... 3
Rango..... ±8 g
Resolución..... 12 bits
Frecuencia de muestreo..... 800 S/s
Ruido..... 3.9 mgrms típico a 25 °C

Potencia de salida

+5 V potencia de salida

- Voltaje de salida ..... 4.75 V a 5.25 V
- Corriente máxima por cada conector... ..... 100 mA

+3.3 V potencia de salida

- Voltaje de salida..... 3.0 V a 3.6 V
- Corriente máxima por cada conector..... 150 mA

+15 potencia de salida

- Voltaje de salida.....+15 V a +16 V
- Corriente máxima .....32 mA (16 mA durante el encendido)

-15 V potencia de salida

- Voltaje de salida.....-15 V a -16 V
- Corriente máxima.....32 mA (16 mA durante el encendido)

Potencia máxima combinada de +15V y -15V de potencia de salida.....500 mW

Requerimientos de potencia

Requiere una fuente de alimentación conectada al conector de alimentación.
Rango de voltaje de la fuente de alimentación..... 6-16 VDC
Consumo máximo de energía..... 14 W
Consumo de energía en reposo..... 2,6 W

Medioambiental

Temperatura ambiente cerca del dispositivo (IEC 60068-2-1, IEC 600682-2).....0 a 40 ° C
Temperatura de almacenamiento (IEC 60068-2-1, IEC 600682-2).....- 20 a 70 ° C
Humedad de funcionamiento (IEC 60068-2-56)..... 10 a 90% de

humedad relativa, sin condensación
Humedad de almacenamiento (IEC 60068-2-56) ..... 10 a 90% de humedad relativa, sin condensación
Altitud máxima ..... 2.000 m
Grado de contaminación (IEC 60664) ..... 2 uso en interiores.

Características físicas

Peso.....193 g

4. Conexiones y elementos que interactúan con NI myRIO-1900

La corriente y la tensión no se conectan directamente a las entradas de NI myRIO-1900 sino que se conectan a un transductor.

Recordando lo visto sobre NI my RIO-1900:

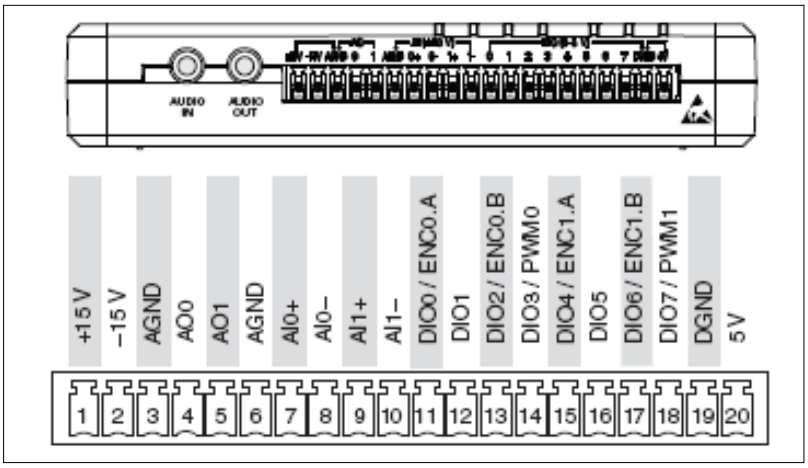


Figura 4.1 Las señales primario / secundario en conector C MSP.

En las entradas AI de NI myRIO-1900 hay que introducir una tensión de ±0…10V tal y como se ha visto en el apartado 3 de la memoria.

- La tensión irá a la entrada AI0+/AI0-.
- La intensidad irá a la entrada AI1+/AI1-.

- La salida digital del Contactor 2 será DIO0.
- La salida digital del Contactor 1 será DIO1.

El transductor que se usará tiene el siguiente esquema:

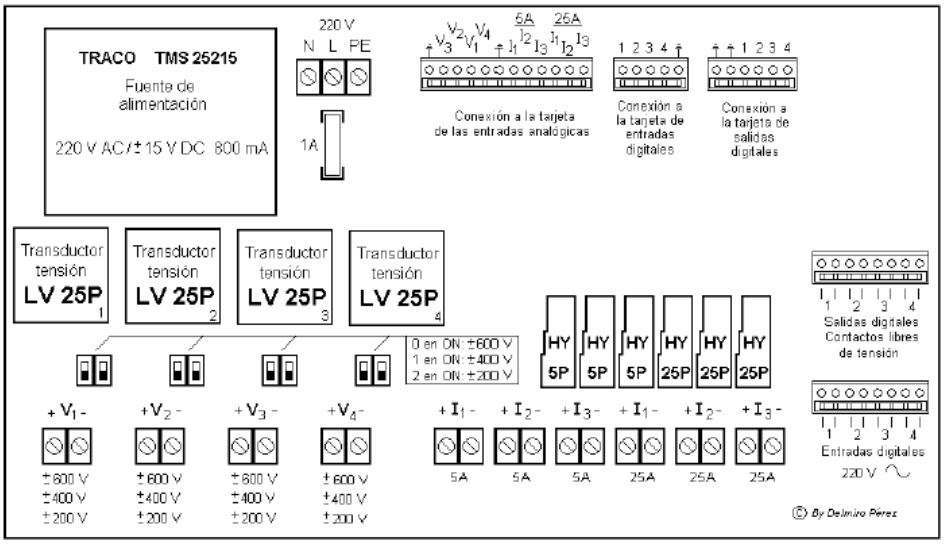


Figura 4.2 Esquema transductor

- **Alimentación:**

A la conexión N y L irá un neutro y una fase respectivamente para alimentar a los transductores.

- **Medida de la corriente:**

Se conectará el cable de la red a la entrada + I_1 – que va un transductor de corriente LEM HY 5P cuyas características son las siguientes:

Características del transductor de corriente LEM HY 5P y LEM HY 25P	
Corriente nominal del primario	5 y 25 A
Tensión de salida analógica instantánea	$\pm 4V$
Precisión total a +25°C	$\pm 1\%$ de I_1
Tensión de alimentación	$\pm 15V(\pm 5\%)$
Aislamiento	2.5 kV_{rms}/50Hz/1 min
Linealidad	<0.1% de I_1
Tiempo de respuesta	<3µs
DI/DT	>50A/µs
Temperatura de operación	-10°C a +80°C
Temperatura de almacenaje	-25°C a +85°C
Consumo de corriente	10 mA
Resistencia interna secundaria	100 Ω
Peso	14g
Relación de transformación	1 : 1000

LEM HY 5P corresponde a una Corriente nominal del primario de 5A y una tensión de salida analógica instantánea de $\pm 4V$, por lo que aunque la entrada de NI myRIO-1900 sea de 0…10V solo se llegará hasta 4 (Se perderá precisión pero la función será igual de válida).

Si 5A corresponden a 4V, la relación de corriente será $\frac{5}{4}$ = 1,25. Por lo tanto en el código la entrada de intensidad se multiplicará por 1,25.

La salida del transductor que se encuentra en la parte que pone "Conexión a la tarjeta de las entradas analógicas" será I_1 e irá a la entrada AI1+ de NI myRIO-1900. El cable de masa del transductor irá a AI1- de NI myRIO-1900.

- **Medida de la tensión:**

Se conectará el cable de la red a la entrada + V_1 – que va un transductor de tensión LEM LV 25 P cuyas características son las siguientes:

Características del transductor de tensión LEM LV 25P	
Corriente nominal del primario	10 mA
Corriente de salida analógica	25 mA
Relación de transformación	2500 : 1000
Precisión total a +25ºC	$\pm 0.6\%$ de I_1
Tensión de alimentación	$\pm 15V (\pm 5\%)$
Aislamiento	2.5 kV_{rms}/50Hz/1 min
Linealidad	<0.2%
Tiempo de respuesta	<40µs para $R_1 = 25k\Omega$
Temperatura de operación	0ºC a +70ºC
Temperatura de almacenaje	-25ºC a +85ºC

En la placa de circuito impreso , para cambiar la escala de entrada de tensiones existen dos pequeños contactos (swichtes), configurados de la siguiente manera:
- Si los dos contactos están a OFF, la escala es de *600 V*.
- Si está conectado (a ON) uno de los contactos, la escala es de *400 V*.
- Si están conectados (a ON) los dos contactos, la escala es de *200 V*.

En nuestro caso los dos contactos están en OFF (Escala 600V).

Se tiene el siguiente circuito:

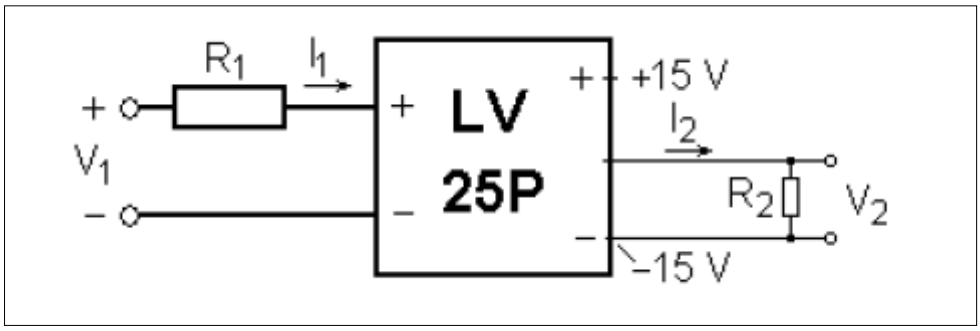


Figura 4.3 Circuito transductor tensión

Como la salida analógica es una corriente máxima de 25 mA y se quiere obtener una salida analógica máxima de 5 V se colocará una resistencia en paralelo:

$$R_2 = \frac{V_2}{I_2}$$

$$R_2 = \frac{5\ V}{0{,}025\ A} = 200\ \Omega$$

La salida del transductor que se encuentra en la parte que pone “Conexión a la tarjeta de las entradas analógicas” será V_1 e irá a la entrada AI0+ de NI myRIO-1900. El cable de masa del transductor irá a AI0- de NI myRIO-1900.

- **Salidas digitales:**

Las salidas digitales pasan por unos optoacopladores que conectan los relés de unos contactos libres de potencial.

5. Desarrollo

5.1. Primera fase: Aprendizaje de Labview

En esta primera fase de desarrollo del proyecto, es necesario adquirir conocimiento acerca de las herramientas que se usarán más adelante. Para ello, se han llevado a cabo dos cursos de aprendizaje de National Instruments sobre el programa LabVIEW. Cada curso consta de un manual teórico y un manual práctico que se describen a continuación.

- **Cursos de LabVIEW - manual teórico y práctico.**

Estos manuales tienen como objetivo enseñar conceptos, técnicas, características, VI y funciones de programación de LabVIEW que puede utilizarse para crear aplicaciones de prueba y medida, adquisición de datos, control de instrumentos, registro de datos, análisis de medidas y generación de informes. Estos manuales presuponen que el usuario está familiarizado con Windows y que tiene experiencia en escribir algoritmos en forma de diagrama de flujos o diagramas funcionales (y que se ha realizado el curso LabVIEW Core 1 para el caso del curso de LabVIEW Core 2) Los manuales de ejercicios y del curso están divididos en lecciones, descritas de este modo:

En el manual teórico, cada lección consta de lo siguiente:
· Una introducción que describe el objetivo de la lección y lo que se aprenderá.
· Una descripción de los temas de la lección.
· Un cuestionario de resumen que prueba los conocimientos enseñados en la lección.

En el manual de ejercicios cada lección consta de un conjunto de ejercicios para reforzar los conocimientos adquiridos de cada tema y de secciones de ejercicios opcionales y de retos o un conjunto de ejercicios adicionales para realizar si el tiempo lo permite.

- **LabVIEW Core 1. Temario y objetivos.**

El primer curso de National Instruments contiene las siguientes lecciones [9]:
· Lección 1 – configuración del hardware.
· Lección 2 – explorando LabVIEW.
· Lección 3 – resolución de problemas y depuración de VI's.
· Lección 4 – implementación de un VI.
· Lección 5 – relacionar datos.
· Lección 6 – gestión de recursos.
· Lección 7 – desarrollo de aplicaciones modulares.
· Lección 8 – técnicas y modelos de diseño comunes.
· Lección 9 – uso de variables.

Al final de este curso, el usuario debería estar preparado para hacer las siguientes tareas:

· Comprender los panes frontales, los diagramas funcionales, los iconos y los paneles conectores.
· Usar las estructuras de programación y los tipos de datos existentes en LabVIEW.
· Usar varias técnicas de edición y depuración.
· Crear y guardar VI's para poder utilizarlos como subVI's.
· Mostrar y registrar datos.
· Crear aplicaciones que utilicen dispositivos DAQ.
· Crear aplicaciones que usen instrumentos de puerto serie y GPIB.

- LabVIEW Core 2. Temario y objetivos.

El segundo curso de National Instruments contiene las siguientes lecciones [10]:
· Lección 1 – introducción.
· Lección 2 – técnicas de sincronización.
· Lección 3 – programación de eventos.
· Lección 4 – gestión de errores.
· Lección 5 – control de la interface de usuario.
· Lección 6 – técnicas de E/S de fichero.
· Lección 7 – mejora de un VI existente.
· Lección 8 – creación y distribución de aplicaciones.

Al final de este curso, el usuario debería saber hacer las siguientes tareas:

· Aplicar patrones de diseño comunes que utilicen notificaciones, colas y eventos.
· Usar la programación de eventos de forma eficaz.
· Controlar programáticamente objetos de la interfaz de usuario.
· Evaluar formatos de E/S de archivos binarios y utilizarlos en aplicaciones.
· Modificar código existente para mejorar la usabilidad.
· Preparar, crear y desplegar aplicaciones independientes.

También mencionar el proceso de autoaprendizaje únicamente experimentando con pequeños VI's sobre la marcha.

5.2. Segunda fase: Estructura y funcionamiento del programa

5.2.1. Especificaciones

El programa tiene que cumplir una seria de características:

· Leer los valores instantáneos de la tensión y la intensidad

· Operar con ellos para sacar la forma de onda de la tensión y de la intensidad, el valor eficaz de la tensión y de la intensidad, la frecuencia, el factor de potencia, la potencia activa, la potencia reactiva, la potencia aparente, la THD de la tensión y de la intensidad y el Cos (φ).

· Guardar los valores anteriores en la "nube" para su visionado desde cualquier parte del mundo y que sean guardados con seguridad.

· Presentar los anteriores valores al usuario de una manera sencilla y gráfica.

· Que el usuario pueda interactuar con el programa fácilmente para elegir el periodo concreto en el que quiere ver los valores.

· Que el usuario pueda interactuar con el programa fácilmente pudiendo manejar una serie de botones que activen unas salidas, en este caso en concreto serían dos contactores.

5.2.2. Estructura

La estructura es la siguiente:

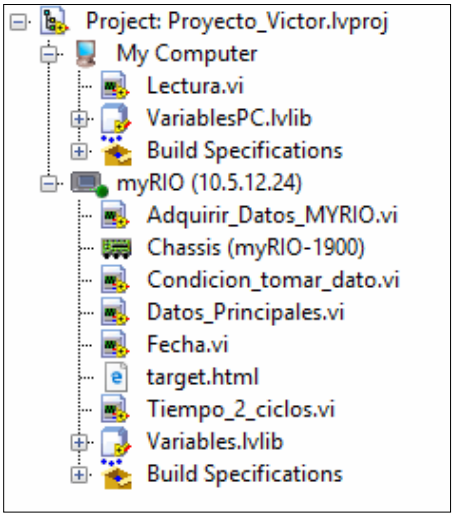


Figura 5.1 Estructura del programa

El proyecto se divide en dos programas principales:

Programa del sistema embebido (NI myRIO-1900)
Programa del ordenador

5.2.3. Programa del sistema embebido (NI myRIO-1900)

El programa principal donde se ejecutan las llamadas al resto de VI's de NI myRIO-1900es "Adquirir_Datos_MYRIO· y se está ejecutando continuamente mientras esté el sistema encendido, éste se encarga de la adquisición de los valores instantáneos de la tensión y de la intensidad que operando con ellos obtendrá:

· Forma de onda de la tensión y de la intensidad.
· Valor eficaz de la tensión y de la intensidad.
· Frecuencia.
· Factor de potencia.
· Potencia activa
· Potencia reactiva
· Potencia aparente
· THD de la tensión y de la intensidad
· Cos (φ)

Estos datos, tomados en tiempo real, son enviados a la red para su lectura mediante variables globales.

La tensión y la intensidad eficaz, las potencias, las potencias-hora consumidas, el factor de potencia y los THD se almacenan en archivos separados por días. El dato se toma cada 10 segundos o cada vez que se produce un cambio considerable respecto al dato anterior.

Almacenamiento en la "nube":

Una gran ventaja con la que cuenta este proyecto es el almacenamiento de los datos en la web lo que significa que se podrán consultar desde cualquier parte del mundo y la seguridad de que estos datos no serán perdidos.

Desde NI myRIO-1900 cada día el archivo de datos se subirá a www.box.com donde hay una cuenta abierta para este proyecto. Si la subida del archivo ha sido satisfactoria se borrará de NI myRIO-1900 quedando almacenada en la "nube". En caso de que haya un error de subida o NI myRIO-1900 no esté conectado a la red el archivo se subirá al siguiente día junto con el archivo de ese mismo día.

5.2.4. Programa del ordenador

Desde el programa "Lectura" del ordenador se podrán visualizar los datos en tiempo real y el histórico de datos tomados:

El histórico de datos se podrá consultar desde cualquier ordenador con Internet ya que NI myRIO-1900 los sube diariamente a www.box.com tal y como se acaba de explicar. Estos datos los importa el ordenador y opera con ellos.

Los datos en tiempo real se podrán consultar siempre que el ordenador del usuario esté conectado a la misma red que NI myRIO-1900

El programa "Lectura" también dispone de botones para controlar dos salidas.

El histórico de datos se presentará de la siguiente manera:

Figura 5.2 Tensión(Blanco) / Intensidad (Rojo) / FP (Verde)

Donde se pueden ver tanto las gráficas de los datos tomados y a qué hora fueron tomados como el consumo de electricidad en potencias-hora.

La manera de visualizar los datos y controlar las salidas se explicará más adelante tal y como viene en el índice.

Los datos en tiempo real se mostrarán de la siguiente manera:

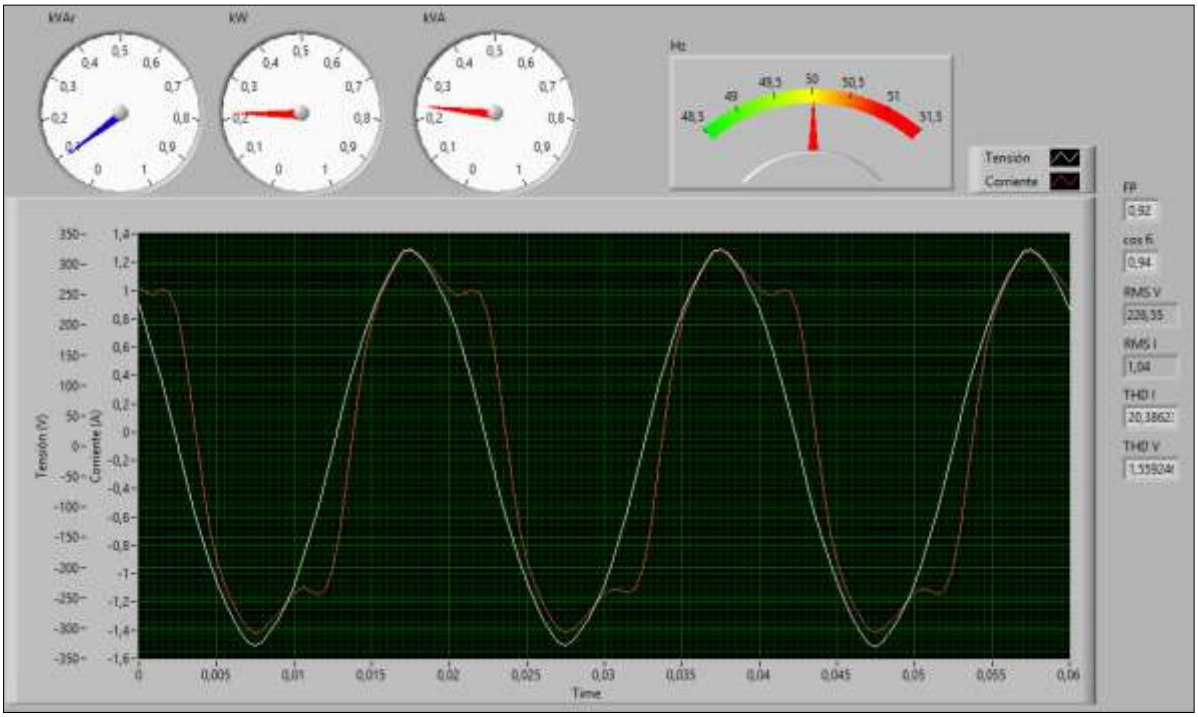


Figura 5.3 Interfaz usuario

5.3. Funcionamiento del programa del sistema embebido (NI myRIO-1900)

Las imágenes de cada parte del código se encuentran en el Anexo 1.

VI: "Adquirir_Datos_MYRIO"

- **Bucle 1**

Es un bucle "while" que estará ejecutándose siempre tal y como se explica más adelante, no es un bucle determinista, para ejecutar una iteración tiene que acabar la anterior por lo que no se sobrecargará la CPU, no es necesario que sea determinista ya que únicamente se trata de una gestión de una red eléctrica sin ninguna función crítica.

En la entrada C/AI0 toma el valor instantáneo de la tensión y en la entrada C/AI1 toma el valor instantáneo de la intensidad. Se emplea un bucle "for" que toma un dato de cada variable eléctrica cada 500 microsegundos durante 300 veces (300 datos) con el que se obtiene un array de con el que se operará posteriormente para obtener los valores eléctricos. Estos dos array serán variables globales que se usarán para trasladar los datos instantáneos al programa del ordenador para poder visualizarlos y para operar en el otro bucle de este mismo programa.

Existen dos botones, uno que activa una variable global booleana que activa la salida C/DIO0 y el LED0 (A la variable global se le ha llamado "Contactor 2" ya que está salida activará un contactor) y otro que activa otra variable global booleana que activa la salida C/DIO1 y el LED1 (A la variable global se le ha llamado "Contactor 1" ya que está salida activará un contactor). En el caso de este proyecto la salida DIO0 activa un contactor con carga resistiva y la salida DIO1 activa un contactor con carga inductiva). Se emplea una variable global para poder ser activado desde el programa del ordenador.
Solo se cambia el valor de las variables booleanas anteriores si el estado del botón es diferente del anterior. Al iniciar el programa del ordenador que lo manejará los dos botones estarán en OFF pero si actualmente el programa de NI myRIO-1900 está ejecutándose y se encuentra con las dos variables activadas éstas no cambiarán ya que no he pulsado ningún botón. Para pasar estas variables a OFF sería necesario pulsar el botón a ON, como tiene el mismo valor que tenía antes seguirá activada, entonces habría que volver a pulsar para pasar el botón y la variable a OFF. Esto es necesario para que no haya cambios imprevistos en la instalación a gestionar al ejecutar el programa.

Además hay un botón STOP que activará una variable local para parar el programa de adquisición de datos, este programa no se puede parar desde el ordenador y siempre estará ejecutándose siempre que NI myRIO-1900 tenga alimentación eléctrica.
Por necesidades constructivas el bucle espera 1300 ms adicionales a lo que tarda ya entre iteración e iteración para no sobrecargar la CPU.

- **Bucle 2**

Es un bucle “while” que estará ejecutándose siempre tal y como se explica más adelante, no es un bucle determinista, para ejecutar una iteración tiene que acabar la anterior por lo que no se sobrecargará la CPU, no es necesario que sea determinista ya que únicamente se trata de una gestión de una red eléctrica sin ninguna función crítica.

En este bucle es donde se hacen todas las operaciones con los datos tomados. En primer lugar se ejecuta el VI “Datos_principales” del cual se obtiene el valor eficaz de la tensión (V), intensidad (A), potencia activa (kW), potencia reactiva(kVAr), potencia aparente (kVA), el factor de potencia (FP), el coseno(φ), la frecuencia y la gráfica con la forma de onda de la tensión y la intensidad. Todos estos datos se meten en una estructura “case” que se activa cada 10 segundos o cuando el VI “Condicion_tomar_dato” manda un valor booleano TRUE, este VI estará unido con un “OR” a la condición de los 10 segundos y se activará siempre que haya un cambio significativo en los datos tomados, se explicará con detalle más adelante.

Para que se active cada 10 segundos se obtiene la fecha en segundos y se divide entre 10, cuando el resto es 0 significa que han pasado 10 segundos por lo que cuando el resto sea igual que 0 devuelve un valor booleano TRUE que activa el “case”, en caso de que haya algún error o que el programa se quede bloqueado y pasa por ejemplo de 9 a 11 segundos sin pasar por 10, si el resto actual es menor al anterior también devuelve un valor booleano TRUE (Robustez) que se une al anterior con un “OR” para activar el “case” cuando se cumpla una de las dos condiciones.

En la estructura “case” además se genera un valor de fecha y hora que corresponderá a los datos que se tomen en ese momento. Se inicializa un array por cada tipo de dato, cada array tiene un tamaño de 10 valores, los datos “Fecha en la que se toma el dato”, “kW”, “kVAr”, “kVA”, “V”, “A”, “THD (V)”, “THD (A)” y “FP” se van almacenando por orden de muestreo en sus respectivos array hasta llegar a 10, esto significa que tenemos 9 arrays (8 de datos y 1 de su fecha), los primeros datos de tensión, intensidad, potencias… se almacenarán en su respectivo array en la posición 1, cuando se vuelva activar la estructura “case” (cada 10 segundos si no se activa por un cambio brusco en los datos) cada valor tomado ocupará la posición 2 correspondiente en su array, así sucesivamente hasta llegar a 10.
Una vez se tomen 10 datos se activa otra estructura “case” que almacena esos array en un fichero, si ya hay datos en ese fichero se solaparan a ellos ordenadamente.
El siguiente muestreo de datos volverá a almacenarse en los mismos arrays en los que se almacenaron antes pero en la posición 1 sustituyendo a los datos anteriores y de esta manera sucesivamente se volverá a repetir el ciclo explicado.
Esto se hace por varios motivos, si únicamente se almacenaran los datos en el array en cuanto haya un fallo o la memoria se llene se perderán todos los datos. Sin embargo si siempre abrimos y cerramos un fichero solo para guardar un dato acabaremos disminuyendo la vida útil del dispositivo por lo que una solución intermedia será almacenar datos en un array y una vez haya almacenados 10 se guardará en un fichero solapándose a los datos tomados anteriormente, así si hay un fallo perderemos solo los datos que se encuentren en el array (10 o menos datos), que corresponderían como máximo a un minuto y medio de toma de datos

(Los datos del fichero no se borrarían ya que se encuentra guardado en el disco duro).

Sin embargo para almacenar los kVArh, kVAh y kWh se hace de una manera diferente, consiste en tomar el dato que corresponda, kW, kVA o kVAr, se multiplica por el tiempo que ha tardado entre la toma de dos datos sucesivos, lo da el VI "Tiempo_2_ciclos", este VI da el tiempo en segundos, por lo que dividiendo entre 3600 se obtiene el correspondiente kVArh, kVAh o kWh. Respetando los ciclos de guardado como los datos anteriores, estos tipos de datos se sumarán a sí mismos 10 veces en memoria volátil, una vez completados los 10 ciclos se sumarán a las potencias-hora correspondientes que ya hubiera almacenadas en el disco duro, el valor de la memoria volátil se pondrá a cero y se volverá a repetir el ciclo sucesivamente, almacenando así de manera robusta el total de potencias-hora consumidas. Esto se hace para coordinar el guardado de estos datos con los anteriores y por el mismo motivo que los datos anteriores.
Si el sen(fi) es mayor que 0 se guardarán como kVArh inductivos, si es menor que 0 se guardarán como kVArh capacitivos.
El guardado de los datos anteriores en disco duro se hace de la siguiente manera:

- Todos los datos mencionados entran en un "case".
- Se envía una dirección "Path" al "case" con la siguiente estructura "aaaammdd.txt" correspondiendo "aaaa" al año, "mm" al mes y "dd" al día, ".txt" será el formato de guardado de los datos. Esto significa que cuando cambie el día cambiará la dirección.
- Se comprobará si existe un fichero con ese nombre, si no existe se creará uno por lo que cada día tendrá su fichero correspondiente, se escribirán los datos obtenidos en binario con la función "Write to Binary File Function", en ese fichero agrupando todos los arrays en un cluster, y se cerrará el archivo.
- Sin embargo si existe un fichero con ese nombre es más complejo:
    1. Se lee el cluster binario de ese fichero del que se obtendrán los 10 arrays (Recordando: 9 arrays de datos y 1 array de la fecha correspondiente cada dato).
    2. Con un "build array" por cada tipo de dato se añaden el array de datos nuevos al array de datos que ya estaban guardados. Con las potencias-hora únicamente se sumarán las potencias-hora acumuladas con el dato de potencia-hora nuevo (que será la suma de 10 ciclos para coordinar el guardado de estos datos con los anteriores tal y como se ha explicado anteriormente).
    3. Se escriben los nuevos arrays de datos en el archivo y se cierra.

- Las potencias-hora se presentarán numéricamente, el valor será el mismo que se haya guardado en ese instante.
- Cada array de datos se juntará con un array de fecha con un "bundle", de esta manera el 1º dato de cada array corresponderá a la 1º fecha del array de fecha, el 2º a la 2º y así sucesivamente, teniendo así el momento exacto en el que se tomó cada conjunto de datos.

- El “bundle” de la tensión, intensidad y factor de potencia con sus fechas se unirán en un array y se presentará en una gráfica que mostrará los tres datos simultanéamente.
- El “bundle” de la potencia activa, potencia reactiva y potencia aparente con sus fechas se unirán en un array y se presentará en una gráfica que mostrará los tres datos simultanéamente.
- El “bundle” de la THD de la tensión y el THD de la intensidad con sus fechas se unirán en un array y se presentará en una gráfica que mostrará los dos datos simultanéamente.
- Los datos que se leyeron anteriormente del cluster de datos guardados en el fichero se envían mediante una variable global “Variant” (a la que se ha llamado “TIME”) a la red que posteriormente se usarán por el VI del ordenador para poder ver los datos del día del que se están recogiendo datos.

Por último este bucle emplea una variable local llamada “Recoger datos” que se activa cada vez que el nombre del fichero es diferente, por lo que se activará una vez al final de cada día. Este bucle tiene una espera de 2500 ms para no sobrecargar la CPU.

- Bucle 3

Tendrá una estructura “case” (solo se activará al final de cada día cuando se active la variable local “Recoger datos”) que activará la subida de los ficheros a la web de la siguiente manera:

1. Se obtienen los nombres de los archivos con datos que hay en el dispositivo (si no ha habido ningún problema en la subida solo debería haber uno).
2. Entran los nombres de los archivos y la dirección de su localización en un array en formato “string” (Cada posición del array será la dirección de un archivo) en una estructura “Flat Sequence” que se encargará de que unas operaciones se realicen después de otras (el guardado antes que el borrado del archivo).

Guardado del archivo:

1. Los datos mencionados entran en una estrucuta “While Loop”
2. Se recorren los nombres de los archivos de manera que la iteración uno extraiga el nombre del archivo en formato “string” de la posición uno del array con los nombres de los archivos. La iteración 2 extraerá la posición 2 y así sucesivamente.
3. El nombre que corresponda en la iteración se unirá con la dirección de la carpeta con los archivos mediante la función “Concatenate Strings” añadiendo un “/” entre medias.
4. El “string” anterior pasará a formato “path” (Dirección de archivo) que pasará a una estructura “case” donde se guardará el archivo con esa dirección.

5. A la estructura “Case” entrarán además los datos de la cuenta donde subimos los archivos (Dirección, cuenta y contraseña).
6. La condición que activa las funciones del “Case anterior” es que el día que se esté guardando no sea el actual, esto es porque al guardar el archivo a la vez que se están introduciendo datos en ese mismo archivo provoca un guardado corrupto, cosa que ocurre al guardarse los datos que tal y como se ha explicado se produce cuando se cambia de fecha, es decir todos los días a las 00:00, al no ser un guardado instantáneo los datos de el día que empieza se estarán tomando a la vez que se estará guardando ese miso archivo cuando termine de guardar los anteriores archivos (Los datos del día actual se ven en “tiempo real” desde el ordenador mediante variables globales).
7. Cuando se detecte que el “string” con el nombre está vacío significa que ya no hay archivo por lo que saldrá del bucle.
8. Si no se detectan problemas en la subida de los datos se envía un valor booleano “TRUE” al siguiente nivel de la estructura “Flat Sequence”, si no hay problema envía un “FALSE”

Borrado del archivo:

1. Las siguientes funciones solo se activarán si no hay problema en el guardado de los datos. Es decir si entra un valor booleano “FALSE”.
2. Al igual que en el guardado entran los nombres de los archivos y la dirección de su localización en un array en formato “string”.
3. Los datos mencionados entran en una estrucuta “While Loop”
4. Se recorren los nombres de los archivos de manera que la iteración uno extraiga el nombre del archivo en formato “string” de la posición uno del array con los nombres de los archivos. La iteración 2 extraerá la posición 2 y así sucesivamente.
5. El nombre que corresponda en la iteración se unirá con la dirección de la carpeta con los archivos mediante la función “Concatenate Strings” añadiendo un “/” entre medias.
6. El “string” anterior pasará a formato “path” (Dirección de archivo) que pasará a una estructura “case” donde se guardará el archivo con esa dirección.
7. La condición que activa las funciones del “Case anterior” es que el día que se esté guardando no sea el actual, por el mismo motivo que en el guardado. A continuación se borrará el nombre del archivo que

corresponda con la dirección “path” que ha entrado a la estructura “Case”.

8. Cuando se detecte que el “string” con el nombre está vacío significa que ya no hay archivo por lo que saldrá del bucle.

3. El último nivel de la estructura “Flat Sequence” Será cambiar el valor de la variable local “Recoger datos” a FALSE” que es la que activa el guardado y borrado de datos que se acaba de explicar, por lo que para volver a realizar estas funciones tendrá que volver a cambiar a “TRUE” que será cuando cambie el día.

VI: “Datos_Principales”

En el VI “Adquirir_Datos_MYRIO” se usaba este VI del que se obtenían los datos. A continuación se explica la manera de obtener estos datos:

1. En primer lugar mediante la variable global “TensiónR”, que es un array con 300 valores instantáneos, se obtiene la forma de onda de la tensión mediante la función “Build Waveform” introduciendo como intervalo entre datos 0,0005 segundos que corresponden a 500 microsegundos, que es el mismo intervalo que se puso entre la toma de un dato y otro en el VI “Adquirir_Datos_MYRIO”.
2. Con otra función “Build Waveform” a la que se le introduce la forma de onda anterior, se le introducen los 300 valores instantáneos de la intensidad que se colocarán su posición respecto a la forma de onda anterior.
3. Con la función “Build Array” se unen las dos formas de onda anteriores y se representan en una misma gráfica, esta gráfica será una variable global que se enviará al VI “Adquirir_Datos_MYRIO” del dispositivo.

4. De la forma de onda de la tensión se obtiene:
   - El valor eficaz de la tensión mediante la función “Basic Averaged DC-RMS VI”. Su valor será una variable global que se enviará al VI “Adquirir_Datos_MYRIO” del dispositivo.
   - La Tasa de Distorsión Armónica (THD) de la tensión mediante la función “Harmonic Distortion Analyzer VI”, se multiplicará por 100 para dar su valor en porcentaje. Su valor será una variable global que se enviará al VI “Adquirir_Datos_MYRIO” del dispositivo.
   - La frecuencia de la onda mediante la función “Extract Single Tone Information VI”. Su valor será una variable global que se enviará al VI “Adquirir_Datos_MYRIO” del dispositivo. La fase de la onda de la tensión también se obtendrá mediante esta función.

5. De la forma de onda de la intensidad se obtiene:
   - El valor eficaz de la intensidad mediante la función “Basic Averaged DC-RMS VI”. Su valor será una variable global que se enviará al VI “Adquirir_Datos_MYRIO” del dispositivo.
   - La Tasa de Distorsión Armónica (THD) de la intensidad mediante la función “Harmonic Distortion Analyzer VI”, se multiplicará por 100 para dar su valor en porcentaje. Su valor será una variable global que se enviará al VI “Adquirir_Datos_MYRIO” del dispositivo.

- La fase de la onda de la intensidad mediante la función “Extract Single Tone Information VI”.

Para obtener la diferencia de fase se resta la fase de la intensidad a la de la tensión, para pasarlo a radianes se multiplica por 2, por π y se divide entre 360. Con este valor se sacará el cos(φ) mediante la función “Cosine Function” que su valor será una variable global que se enviará al VI “Adquirir_Datos_MYRIO” del dispositivo.
También se obtendrá el sin(φ) mediante la función “Sine Function” que se usará para saber si los kVAr son capacitivos o inductivos. Si es mayor que 0 serán inductivos, si es menor que 0 serán capacitivos.

6. Multiplicando el valor eficaz de la tensión y el de la intensidad dividiéndolo posteriormente entre 1000 se obtienen los kVA (Potencia aparente). Su valor será una variable global que se enviará al VI “Adquirir_Datos_MYRIO” del dispositivo.

7. Multiplicando el array de los valores instantáneos de la tensión y los de la intensidad se obtienen los valores instantáneos de la potencia activa. Con la función “Build Waveform” se saca su forma de onda y con la función “Basic Averaged DC-RMS VI” se obtiene su valor medio que será el valor eficaz de la potencia activa (W) que dividiendo entre 1000 se obtendrán los kW (su valor será una variable global que se enviará al VI “Adquirir_Datos_MYRIO” del dispositivo).

8. Dado que Potencia aparente = $\sqrt{(Potencia\ activa)^2 + (Potencia\ reactiva)^2}$ o dicho de otra manera S = $\sqrt{P^2 + Q^2}$ entonces como ya tenemos S y P, con la operación Q= $\sqrt{S^2 - P^2}$ se obtendrán los kVAr (Potencia reactiva). Con una estructura “Case”, si son kVAr capacitivos (sin(φ)<0) los kVAr se representarán con un valor negativo, si son kVAr inductivos (sin(φ)>0) los kVAr se representarán con un valor positivo. El valor de los kVAr será una variable global que se enviará al VI “Adquirir_Datos_MYRIO” del dispositivo.

9. El factor de potencia se calculará dividiendo la potencia activa entre la potencia aparente. Su valor será una variable global que se enviará al VI “Adquirir_Datos_MYRIO” del dispositivo.

VI “Condicion_tomar_dato”

En el VI “Adquirir_Datos_MYRIO” se explicaba que los datos se tomaban cada 10 segundos o cuando el VI Condicion_tomar_dato mandaba un valor booleano “TRUE”. A continuación se explica cuando envía dicho valor:

- Con los valores de tensión eficaz se hace la media de los 10 últimos valores, se envía un valor “TRUE” cuando:
    1. El valor actual de tensión es menor que la media multiplicada por 0,8.
    2. El valor actual de tensión es mayor que la media multiplicada por 1,25.
- Con los valores de intensidad eficaz se hace la media de los 10 últimos valores, se envía un valor “TRUE” cuando:
    1. El valor actual de intensidad es menor que la media multiplicada por 0,2.
    2. El valor actual de intensidad es mayor que la media multiplicada por 2.

Esto se hace por si los datos que se están tomando son muy parecidos no saturar la CPU del dispositivo tomando valores con gran frecuencia, siendo suficiente tomarlos cada 10 segundos tal y como se explicó anteriormente, sin embargo cuando haya alguna variación con esta condición no pasará desapercibida. Es un método de optimización.
Para tomar un dato por variación de tensión será más sensible que si varía la intensidad ya que la intensidad varía más fácilmente al introducir o quitar cargas, sin embargo si varía la tensión será por huecos, sobretensiones, etc.

VI “Fecha”

Es el VI que utiliza el VI “Adquirir_Datos_MYRIO” que da el nombre del archivo según la fecha. Funciona de la siguiente manera:

1. Se obtiene el valor de la fecha con la función “Get Date/Time In Seconds Function”.
2. Mediante la función “Format Date/Time String Function” se obtiene un string de la fecha actual con el formato YYYY MM DD, esto se hace introduciendo en la función en el apartado “time format string” según el manual el siguiente código “%Y %m %d”
3. Se extrae el año, el mes y los días por separado con la función “String Subset Function”, el año empezaría en la posición 0 del string de la fecha y ocupa 4 posiciones, el mes empezaría en la posición 5 y ocupa 2 posiciones, el día empezaría en la posición 8 y ocupa 2 posiciones. La posición se introduce en “offset” de la función y las posiciones que ocupa en “length”.
4. Con la función “Concatenate Strings Function” se juntan los 3 strings anteriores y un “.txt” de manera que quede un string final con el formato “YYYYMMDD.txt” que mediante la función “String To Path Function” pasará de ser un string a ser una dirección.
5. Mediante la función “Build Path Function” insertando la dirección anterior en la entrada “name or relative path” y el directorio por defecto (Obtenido con la función “Default Data Directory VI) en la entrada “base path”. La salida de esta función será la dirección final del archivo. Su valor será una variable global que se enviará al VI “Adquirir_Datos_MYRIO” del dispositivo.

VI "Tiempo_2_ciclos"

Este VI se utiliza en el VI "Adquirir_Datos_MYRIO" y se utiliza para saber cuánto tiempo transcurre desde que se toma un dato de potencia a otro para obtener tal y como se explicó las potencias-hora. Este VI consiste en:

1. Con la función "Format Date/Time String Function" insertando en la entrada "time stamp" la fecha obtenida con la función "Get Date/Time In Seconds Function" y en la entrada "time format string" escribiendo el código "%S%4u" obteniendo los segundos en formato "string" con 4 dígitos de precisión.
2. Con la función "Fract/Exp String To Number Function" se pasan los segundos de formato a "string" a formato "float".
3. Se resta el valor de segundos actual al anterior y se obtiene una variable global float que se enviará al VI "Adquirir_Datos_MYRIO" del dispositivo.
4. Para resolver el momento en el que el valor anterior está acabando el minuto pero el valor actual ha vuelto a empezar por 0 (empezado un nuevo minuto), si el valor anterior es mayor que el actual se le suman 59 segundos al valor actual antes de restar el actual al anterior para que la diferencia de segundos sea la correcta.

5.4. Funcionamiento del programa del ordenador.

VI "Lectura"

- Bucle 1

Tal y como se explicó anteriormente, a este VI llegan las 2 variables globales con los arrays de los valores instantáneos de la intensidad y de la tensión. Se obtienen exactamente los mismos datos que en el VI "Datos_Principales", y se muestran de la siguiente manera:

· *Valor eficaz Intensidad:* Valor numérico
· *Valor eficaz Tensión:* Valor numérico
· *Potencia aparente:* Indicador tipo "manómetro"
· *Potencia activa:* Indicador tipo "manómetro"
· *Potencia reactiva:* Indicador tipo "manómetro" Se mostrará negativa cuando sea capacitiva y positiva cuando sea inductiva.
· *Factor de potencia:* Valor numérico
· *Cos(φ):* Valor numérico
· *Factor de Potencia:* Valor numérico
· *Frecuencia:* Indicador
· *THD Intensidad:* Valor numérico
· *THD Tensión:* Valor numérico
· *Forma de onda de la tensión e intensidad:* Juntas en una misma gráfica con un eje y para cada tipo de valores.

Todos estos datos se verán en "tiempo real" contando con el retraso entre el enviado de las variables globales y su cálculo. Se llegó a la conclusión de que era más eficaz enviar únicamente las variables globales de los valores instantáneos de

la tensión e intensidad y calcular el resto de valores que mandar las variables globales con el contenido de todos los valores.

Desde el VI “Adquirir_Datos_MYRIO” se envía una variable global tipo “Variant” llamada “TIME” en la que se encuentran los arrays con los datos históricos del día actual. El funcionamiento es el siguiente:

1. La variable “TIME” se conecta a la entrada “variant” de la función “Variant To Data Function”.
2. A la entrada “type” de la función “Variant To Data Function” se conecta el tipo de datos que se van a extraer, es decir un “cluster” con 8 “arrays” de datos tipo “double” (V, A, FP, Kw, Kva, Kvar, THD (V) y THD (A)), 3 datos tipo “double” (kVArh, kVAh, kWh) y un “array” de fechas que se corresponderán con la fecha exacta en la que cada dato fue tomado.
3. En la salida “ data” de la función “Variant To Data Function” saldrá una salida con los datos que se corresponderán al tipo que se introdujo en la entrada “type” que se extraerán con la función “Unbundle”
4. Se introduce en una función “Bundle” el “array” de fechas y el “array” de un tipo de dato (Primero la fecha (eje X) y segundo los datos (eje Y)). Esto se hace con todos los datos menos con las potencias-hora.
5. La salida “Bundle” de los datos tensión, intensidad y factor de potencia se unen en un “Build Array”. La salida “Bundle” de los datos potencia activa, reactiva y aparente se unen en un “Build Array”. La salida “Bundle” de los datos THD de la intensidad y el THD de la tensión se unen en un “Build Array”.
6. Cada salida de los “Build Array” se unen a gráficas múltiples. La gráfica de la tensión, intensidad y factor de potencia tendrá un eje Y para cada tipo de dato. La gráfica de las potencias únicamente dispondrá de un eje Y para las 3 potencias. La gráfica de los THD tendrá un único eje Y para las dos tasas de distorsión armónica.
7. Las potencias-hora como son un valor acumulativo se muestra cada una numéricamente.

Existen dos botones, uno que activa una variable global booleana que activa la salida C/DIO0 y el LED0 (A la variable global se le ha llamado “Contactor 2” ya que está salida activará un contactor) y otro que activa otra variable global booleana que activa la salida C/DIO1 y el LED1 (A la variable global se le ha llamado “Contactor 1” ya que está salida activará un contactor). Estas salidas se activan en el VI “Adquirir_Datos_MYRIO” ya que son enviadas mediante una variable global .En el caso de este proyecto la salida DIO0 activa un contactor con carga resistiva y la salida DIO1 activa un contactor con carga inductiva). Se emplea una variable global para poder ser activado desde el programa del ordenador.

Solo se cambia el valor de las variables booleanas anteriores si el estado del botón es diferente del anterior. Al iniciar el programa del ordenador que lo manejará los dos botones estarán en OFF pero si actualmente el programa de NI myRIO-1900 está ejecutándose y se encuentra con las dos variables activadas éstas no cambiarán ya que no he pulsado ningún botón. Para pasar estas variables a OFF sería necesario pulsar el botón a ON, como tiene el mismo valor que tenía antes seguirá activada, entonces habría que volver a pulsar para pasar el botón y la

variable a OFF. Esto es necesario para que no haya cambios imprevistos en la instalación a gestionar al ejecutar el programa.

Existe una variable booleana global llamada “stop” que al pasar su valor a “TRUE” para el bucle. Esta variable booleana global se activará mediante un botón que estará en el bucle 2. Esta variable es puesta en “FALSE” siempre que se inicie el programa, es decir, en la iteración 0, ya que después de haberla activado para parar los 2 bucles del programa habrá quedado en “TRUE”.

Como método de evitar sobrecargas en el CPU, entre iteración e iteración habrá como mínimo 1500 ms de intervalo. Esto se hace mediante la función “Wait (ms)”.

- Bucle 2

En primer lugar como método de evitar sobrecargas en el CPU, entre iteración e iteración habrá como mínimo 500 ms de intervalo. El stop se activa de igual manera que el bucle anterior.
Habrá una estructura “Flat Sequence”:

1. El que el primer paso será, como los bucles no van sincronizados, poner en “False” la variable global “stop”, así se asegura el reinicio de esta variable.
2. El segundo nivel se encargará de cargar los datos del ordenador:
    1. Se inicializa un “string” que escribe “Cargando datos…” y se activa un Led para indicar que el programa está bajando los datos de la página web al ordenador.
    2. Si no hay una carpeta llamada Archivos de datos se crea y si la hay se extraen los nombres de los archivos de datos en un “array de strings” en la salida “filenames” de la función “List Folder” a la que como dirección “path” de esta función se le habrá introducido la dirección completa de la carpeta “Archivos de datos”.
    3. Los nombres de los archivos en el ordenador entran en una estructura case que se activará en la iteración 0, es decir para actualizar el programa habría que reiniciarlo.
    4. Dentro de este case se introducen los datos de la página web (Dirección, cuenta y contraseña) en la función “Open Session VI” para iniciar sesión, posteriormente se introduce la dirección de los archivos de la página en la función “Directory Listing VI” del que se obtiene un “array de strings” con todos los nombres de los archivos.
    5. A su vez, dentro de este case habrá un bucle for que se ejecuta tantas veces como numero de “strings” hay en el “array de strings” de la página web. Este tamaño se obtiene con la función “Array Size”.
    6. Dentro de este bucle “For” se lee el nombre del archivo (como este bucle se ejecuta tantas veces como nombres de archivos hay, se irán leyendo todos los nombres de los archivos) en la página web, posteriormente se busca en el ordenador si existe ese nombre dentro del ordenador, si existe no se hace nada, si existe se guarda en el ordenador mediante la función “Simple Ge VI”.

En este punto ya están guardados todos los archivos con datos en el ordenador.

3. El siguiente paso de la estructura "Flat Sequence" será poner en "FALSE" el Led booleano y también la propiedad de visible del "string" Cargando datos…". Así se indicará que se ha acabado de cargar los datos. También hay una estructura case que tiene las siguientes funciones:

    1. Si no hay ningún archivo mostrar un "display" al usuario que ponga "No hay fechas disponibles" mediante la función "Configure display Message to User". Si el usuario pulsa el botón "OK" si aunque no haya archivos NI myRIO-1900 está funcionando podrá ver los datos en tiempo real y lo datos acumulados del día actual.
    2. En caso de que haya archivos se crea una función Menú con la que se creará un Menú en la parte superior en la que se elegirá la fecha y el periodo a visualizar. Se podrá visualizar un año completo, un mes completo o un día completo y por supuesto el que elija el usuario. Por defecto se mostrarán los datos del último día. Para detalles consultar el código adjunto. En resumen se recorren con bucles "While" los nombres de las fechas, en primer lugar el año, después el primer mes, luego los días de ese mes, después el segundo mes y así sucesivamente. De esta manera solo se pueden seleccionar las fechas en las en las que haya lectura de datos.
       Este menú se mostrará de tal manera que el primer menú sean los años, en el submenú de cada año se elegirá el mes a visualizar o si se lee el año completo y en el submenú del mes dará la opción de elegir un día o el mes completo a visualizar.

4. Es el último nivel de la estructura "Flat Sequence". En primer lugar hay una estructura "Case" que es la que se encarga de dar el nombre del archivo a leer.
    - Si es la primera iteración el valor del "Case" será "TRUE", entonces entrarán los nombres de los ficheros que tienen el formato AAAAMMDD, este formato no es casual, está hecho así ya que la fecha siempre será ascendente.
      El "array de strings" se pasa a formato "array de enteros" con la función "Decimal String To Number Function", de este "array" se saca el número más alto (que será la última fecha tomada) con la función "Array Max & Min Function", una vez sacado este número se vuelve a pasar a "string" con la función "Number To Decimal String Function".
      Se extraen los "strings" correspondientes al año, mes y día del "string" con la última fecha tomada y se llevan junto a otro "string" ".txt" al "case" donde se leen los archivos que se explicará a continuación.
    - Si no es la primera iteración se realizará el mismo proceso anterior pero con la fecha elegida en el menú que se explicó antes. El "string" con la fecha saldrá de una estructura "Event", también dentro de la estructura "Event" se encuentra el botón de "Stop" que activará la

variable global “STOP” que para los dos bucles después de que finalicen todas las operaciones de la iteración. La estructura “Event” es importante porque el bucle se quedará en espera hasta que cambiemos la fecha en el menú o pulsemos el botón “Stop”, es una manera de optimizar el uso de la CPU. Cuando se pulsa el botón “Stop” al esperar que acaben todas las operaciones de la iteración se evita que se produzcan errores.

El día, el mes y el año seleccionado de una de las dos maneras anteriores junto con la extensión “.txt” entran a usa estructura “Case” que se activa siempre y cuando haya archivos en el ordenador.
Dentro de esta estructura “Case” hay otra a la que entran los mismos datos, solo se activará siempre que la fecha seleccionada sea diferente a la anterior, esto evita que el programa opere con los datos en caso de que se seleccione la misma fecha, por lo que el resultado será el mismo pero más eficiente.
Dentro de este último “Case” hay otra estructura “Case”. Las condiciones son si se ha elegido día, mes o año.

Si se elige representar un día:

- Se juntan el día, mes y año con el “.txt” y se obtiene el nombre del archivo a abrir, se junta con el directorio donde se encuentra el “string” final y se pasa a dirección “path”. Este proceso es semejante a los descritos en casos anteriores.
- Con la dirección del archivo que queremos representar se abre mediante la función “Open/Create/Replace File Function”
- Como el archivo aunque esté guardado en “.txt” está en binario se abrirá con la función “Read from Binary File Function”
- A la entrada “ data type” de la función “Read from Binary File Function” se conecta el tipo de datos que se van a extraer, es decir un “cluster” con 8 “arrays” de datos tipo “double” (V, A, FP, Kw, Kva, Kvar, THD (V) y THD (A)), 3 datos tipo “double” (kVArh, kVAh, kWh) y un “array” de fechas que se corresponderán con la fecha exacta en la que cada dato fue tomado.
- En la salida “ data” de la función “Read from Binary File Function” saldrá una salida con los datos que se corresponderán al tipo que se introdujo en la entrada “data type” que se extraerán con la función “Unbundle”

Si se elige representar un mes:

- El proceso es el mismo que el anterior solo que se añaden todos los datos de todos los días de ese mes.
- Para sumar los datos se recorren los días del mes:
    1. Como punto de partida ya esta elegido el mes y el año por lo que solo queda recorrer los días que haya de ese mes.
    2. Para recorrer el bucle se utilizará como día el número correspondiente a cada iteración. Este número se pasará a “string” y se añadirá al año y mes correspondiente mediante la función “Concatenate Strings” (Si el día es menor que 10 se

le añade un 0 delante después de pasarlo a string para seguir cumpliendo el formato).

3. Como al "Case" entra el "array de strings" con los nombres de los ficheros con la función "Search 1D Array" se busca a ver si existe un fichero con ese nombre.
4. Si no existe (la función "Search 1D Array" da un valor de -1) y entra a un case en el que los valores serán los mismos que los de antes (se conservan mediante la función "Feedback Node") y se pasará a la siguiente iteración.
5. Si existe un archivo con ese nombre se leerá de la misma manera descrita que en el bucle 1. Si es el primer archivo de ese mes únicamente se leerá, si ya es un archivo posterior se sumará al anterior, los históricos de datos mediante la función "Build Array" (Los datos anteriores que se han conservado mediante la función "Feedback Node"primero y los actuales después para respetar el orden. Las potencias-hora se suman con una simple suma del dato anterior al actual.

Si se elige representar un año:

- El proceso es semejante que el anterior solo que en vez de recorrerse todos los días del mes se recorren los días de cada mes y todos los meses. Se comprueba si existe el archivo y si existe se suma al anterior.

Una vez se tengan los datos a representar tanto si son de un día, de un mes o de un año:

- Se introduce en una función "Bundle" el "array" de fechas y el "array" de un tipo de dato (Primero la fecha (eje X) y segundo los datos (eje Y)). Esto se hace con todos los datos menos con las potencias-hora.
- La salida "Bundle" de los datos tensión, intensidad y factor de potencia se unen en un "Build Array". La salida "Bundle" de los datos potencia activa, reactiva y aparente se unen en un "Build Array". La salida "Bundle" de los datos THD de la intensidad y el THD de la tensión se unen en un "Build Array".
- Cada salida de los "Build Array" se une a gráficas múltiples. La gráfica de la tensión, intensidad y factor de potencia tendrá un eje Y para cada tipo de dato. La gráfica de las potencias únicamente dispondrá de un eje Y para las 3 potencias. La gráfica de los THD tendrá un único eje Y para las dos tasas de distorsión armónica.
- Las potencias-hora como son un valor acumulativo se muestra cada una numéricamente.

En la gráfica de la tensión, intensidad y factor de potencia se representa la tensión mediante puntos blancos, la intensidad mediante puntos rojos y el factor de potencia mediante puntos verdes.
En la gráfica de la potencia se representa la potencia activa mediante puntos blancos, la potencia aparente mediante puntos rojos y la potencia reactiva mediante puntos verdes. (Se representa en KW, KVA Y KVAR).

En la gráfica de las THD se representa la tasa de distorsión armónica de la intensidad en blanco y la tasa de distorsión armónica de la tensión en rojo.

Las fechas vienen representadas con el formato HH:MM:SS DD/MM/AAAA.

5.5. Creación de la aplicación en tiempo real en NI myRIO-1900

Dentro de "myRIO" LabVIEW permite mediante la herramienta "Build Especifications" crear un una aplicación en tiempo real" haciendo botón derecho en la herramienta seleccionando "New"→ "Real-Time Application" tal y como muestra la siguiente imagen:

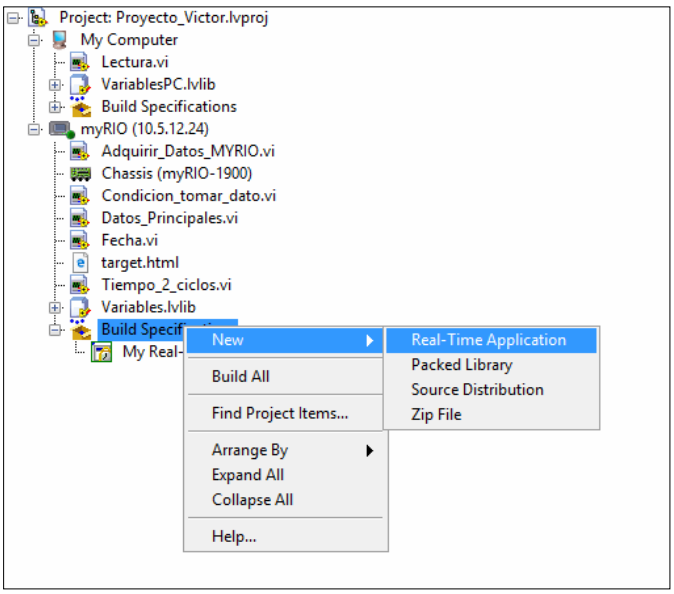


Figura 5.4. Creación aplicación en tiempo real

Después de configurar algunos parámetros sencillos tal y como la carpeta de destino se seleccionan los VI's correspondientes dentro de la aplicación en tiempo real y se presiona el botón "Build":

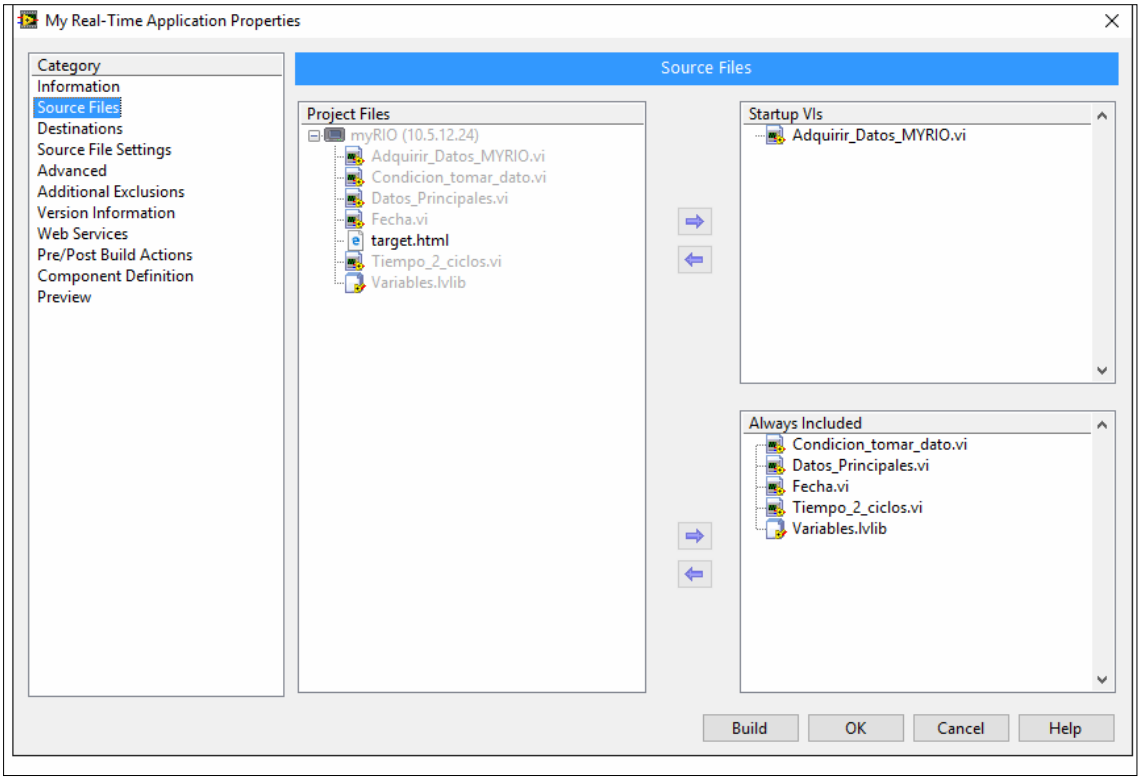


Figura 5.5. Configuración aplicación en tiempo real

Una vez creada se hace botón derecho en la aplicación y se selecciona “Deploy”:

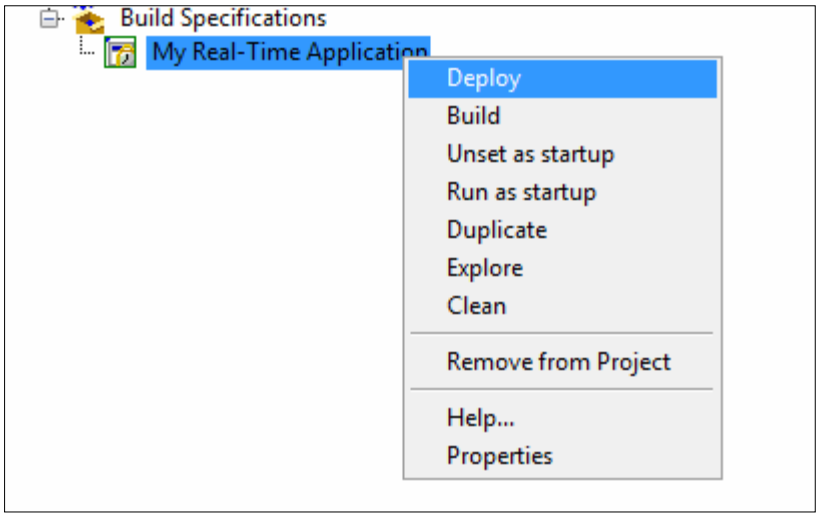


Figura 5.6. Uso aplicación tiempo real

Esto introducirá el programa dentro de NI myRIO-1900 como una aplicación en tiempo real y se ejecutará siempre que el dispositivo tenga alimentación.

5.6. Creación del ejecutable del programa del ordenador

Dentro de “My Computer” LabVIEW permite mediante la herramienta “Build Especifications” crear un ejecutable “.exe” haciendo botón derecho en la herramienta seleccionando “New”→ “Application (EXE)” tal y como muestra la siguiente imagen:

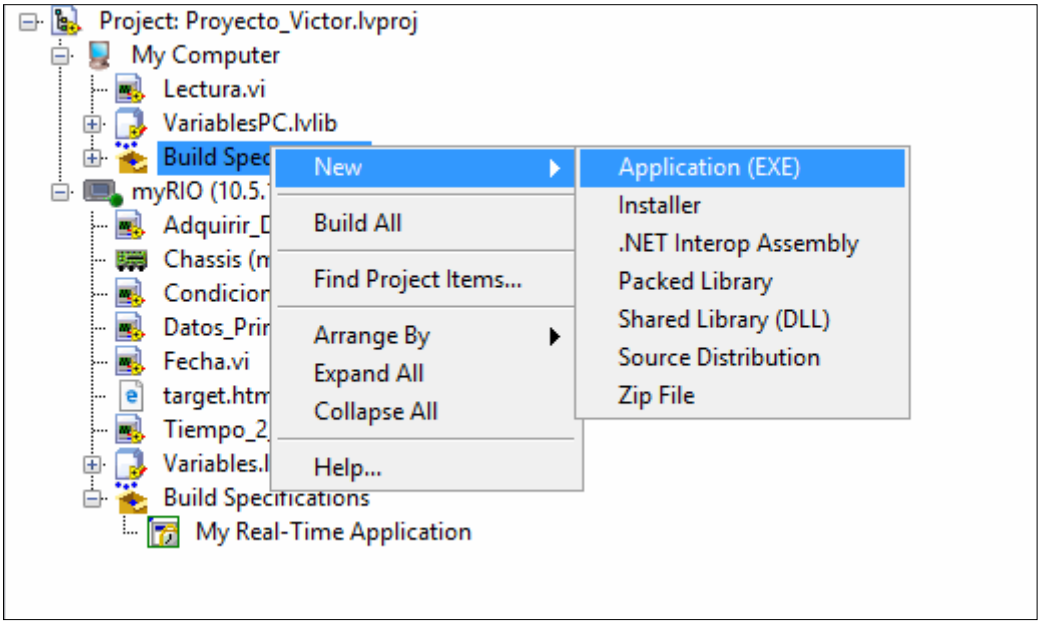


Figura 5.7. Creación ejecutable

Después de configurar algunos parámetros sencillos tal y como la carpeta de destino se seleccionan los VI's correspondientes dentro del ejecutable y se presiona el botón "Build":

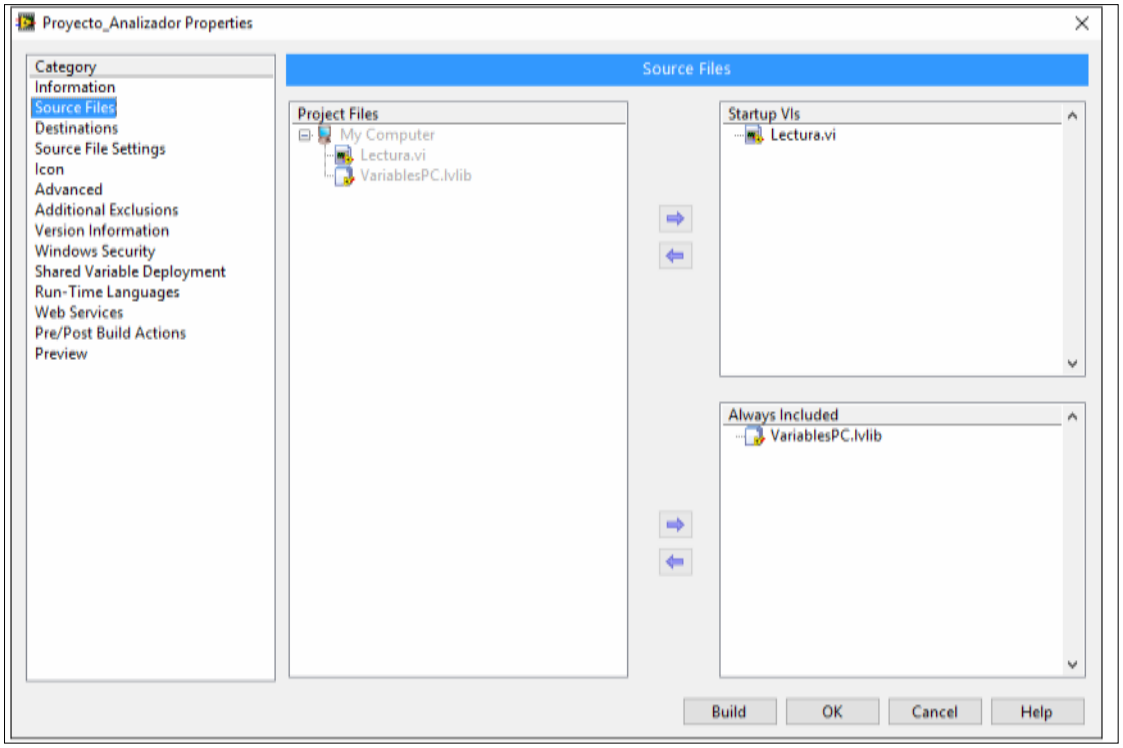


Figura 5.8. Configuración ejecutable

Una vez se presione el botón "Build" se creará el ejecutable que en este caso se ha llamado "Proyecto_Analizador"

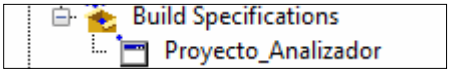


Figura 5.8. Ejecutable

6. Uso del programa por el usuario

Para abrir el programa se ha creado un ejecutable (“.exe”) llamado analizador tal y como se ve en la siguiente imagen:

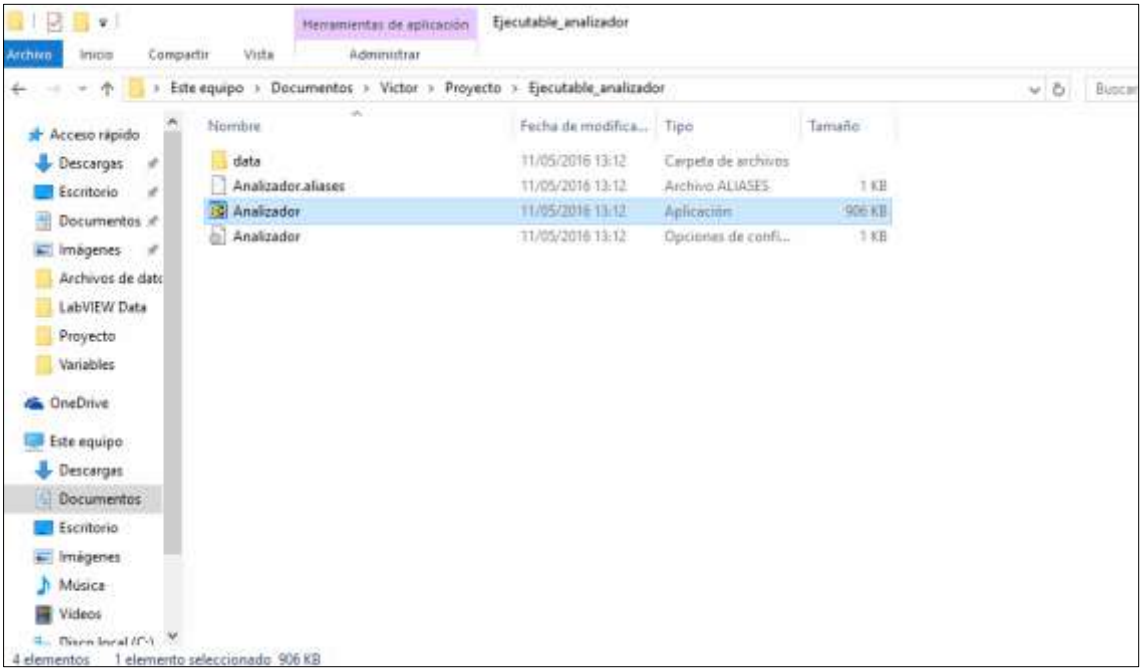


Figura 6.1 Carpeta del programa

Si estamos conectados a la misma red que el dispositivo se ejecutará sin problemas. Sin embargo si no estamos conectados a la misma red saldrá el siguiente mensaje:

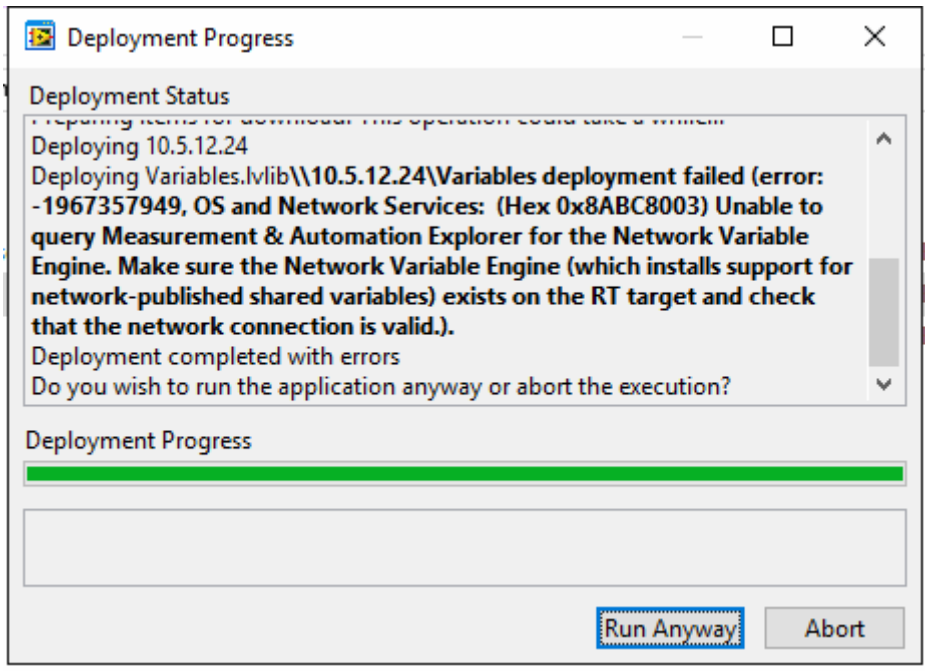


Figura 6.2 Carga del programa

Se refiere a que algunas variables globales no se pueden usar ya que no estamos conectados a la misma red, pulsando el botón “Run Anyway” se iniciará el programa sin problemas.

Una vez iniciado se mostrará siempre los datos del último día tomado:

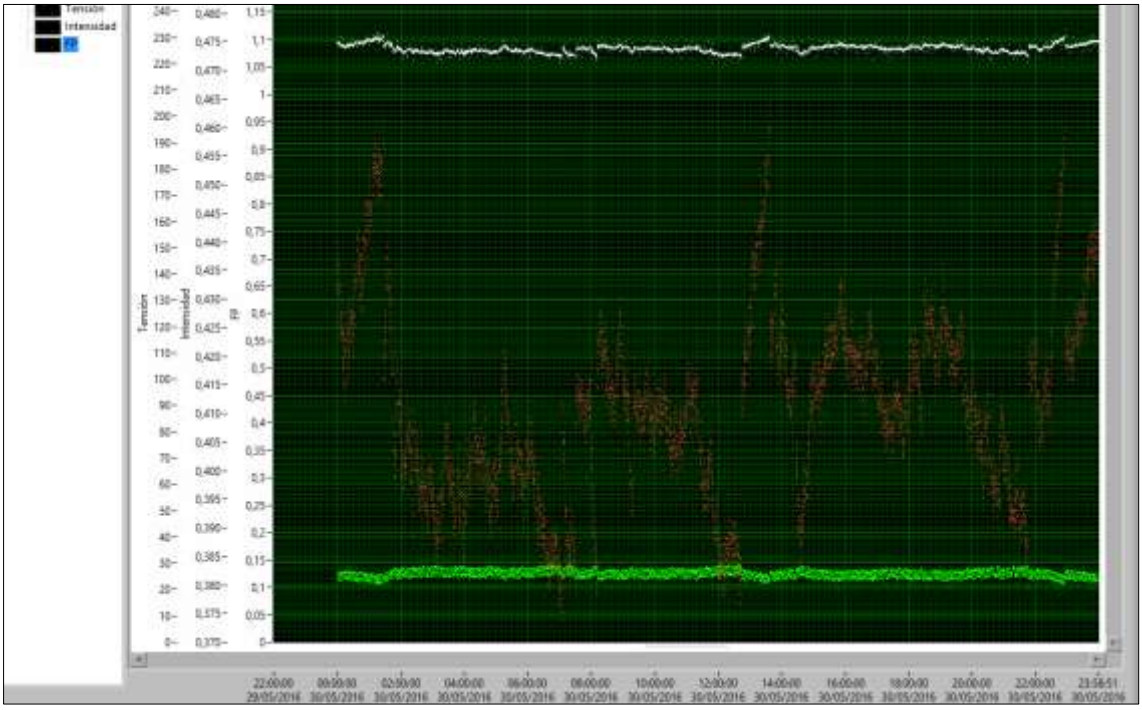


Figura 6.3 Grafica datos históricos

Vemos que los datos tomados son de 24h. Como no ha habido variación de cargas la intensidad se ha mantenido prácticamente igual durante todo el día, por lo que se muestra con gran precisión.
En la parte superior hay un menú llamado “Fecha” donde se elegirá la fecha de los datos a mostrar en la pestaña “Fechas anteriores”.

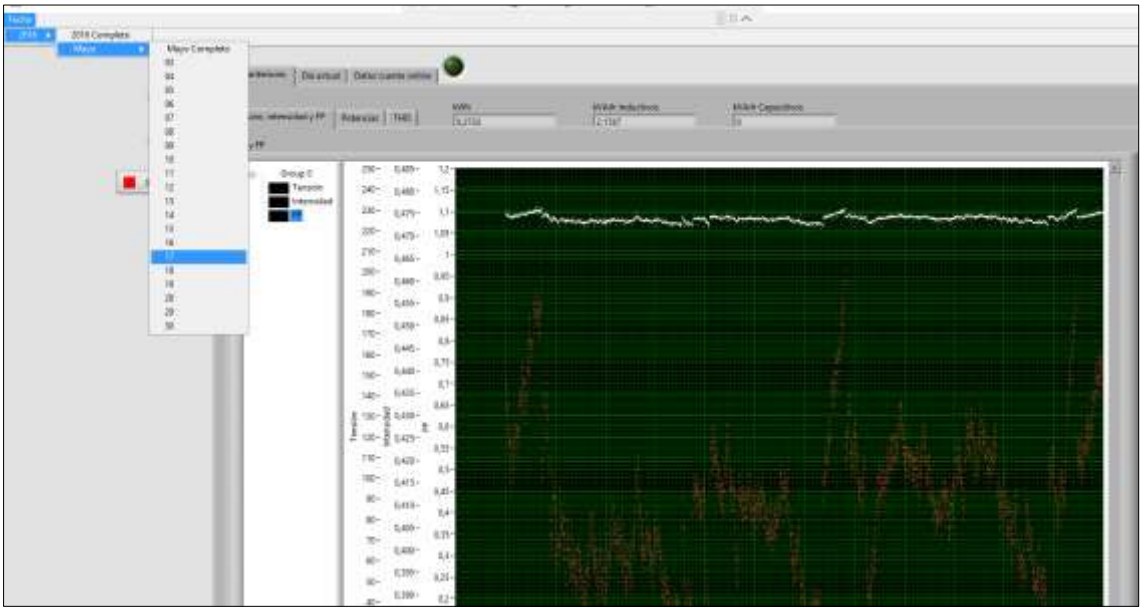


Figura 6.4 Selección fecha

Se puede elegir un año completo, un mes completo o un día concreto. En el caso de elegir un mes o un año completo el programa tardará unos segundos en hacer el cálculo ya que junta los datos de los archivos diarios.

Existen 3 pestañas superiores:

- **Fechas anteriores:**

Se visualizan los kWh, kVArh inductivos y kVArh Capacitivos del día que hayamos elegido en el menú. Aparte tendrá 3 subpesatañas donde se elegirán los datos que queremos ver ("Tensión, intensidad y FP", "Potencias" y "THD"):

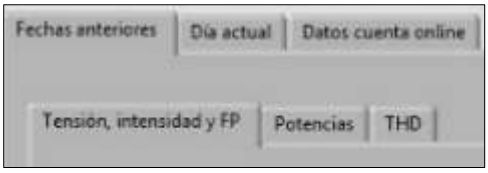


Figura 6.5 Selección menú datos históricos

- **Día actual:**

Existen dos subpestañas, "Valores Instantáneos" y "Datos":

Figura 6.6 Selección menú datos día actual

Si se elige la pestaña "Valores Instantáneos" tal y como indica su propio nombre se verán los valores instantáneos tal y como muestra la siguiente imagen:

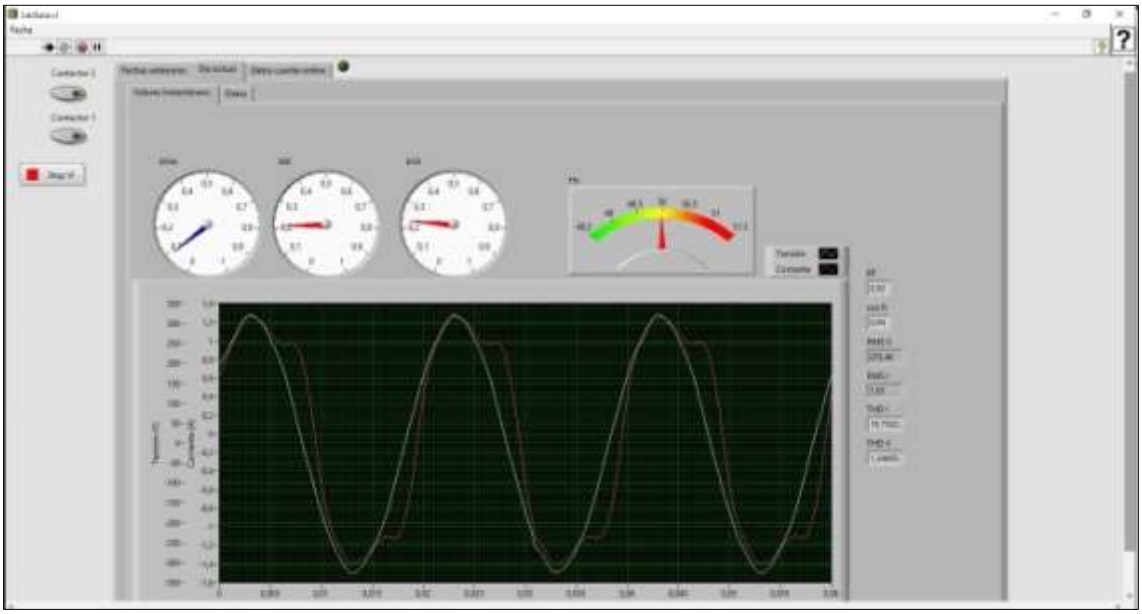


Figura 6.7 Grafica tiempo real

Donde se muestran en indicadores la potencia reactiva, la potencia activa, la potencia aparente y la frecuencia.

Numéricamente se muestra el FP, el cos(φ), el valor eficaz de la tensión y de la intensidad y la THD de la tensión y la Intensidad.
Gráficamente la forma de onda de la tensión (En blanco) y la corriente (En rojo).

Si se elige la pestaña “Datos” se visualizan los kWh, kVArh inductivos y kVArh Capacitivos del día actual. Aparte tendrá 3 subpesatañas donde se elegirán los datos que queremos ver (Tensión, intensidad y factor de potencia; Potencias o THD) de la misma manera que los datos históricos.

Si estamos conectados a la misma red que el dispositivo se verán los datos del día actual. Si no se está conectado a la misma red no aparecen datos tal y como muestra la siguiente imagen:

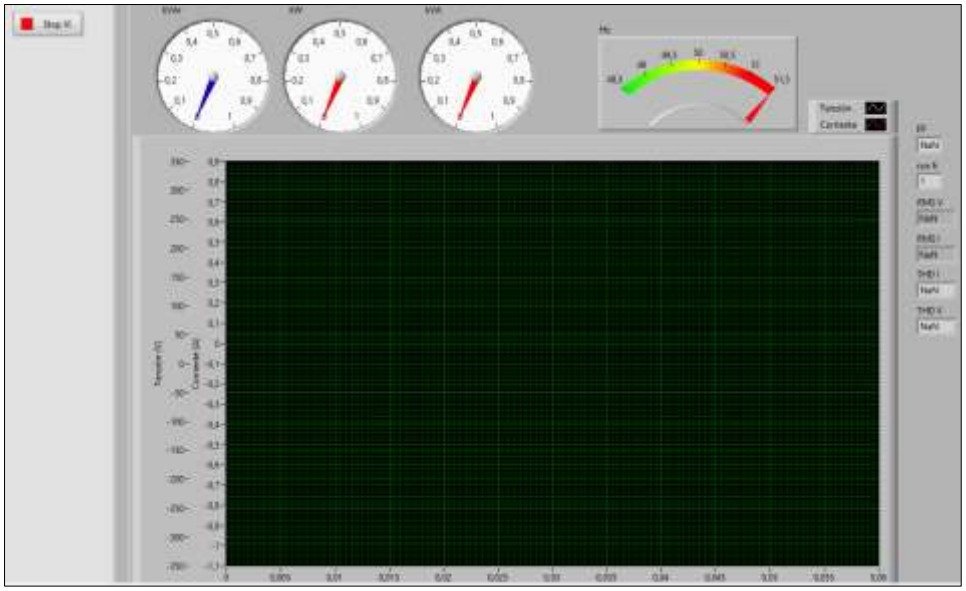


Figura 6.8 Grafica vacía

- **Datos cuenta online:**

Es donde se introducirán los datos de la cuenta, contraseña y dirección de los archivos de la página web que se esté usando.

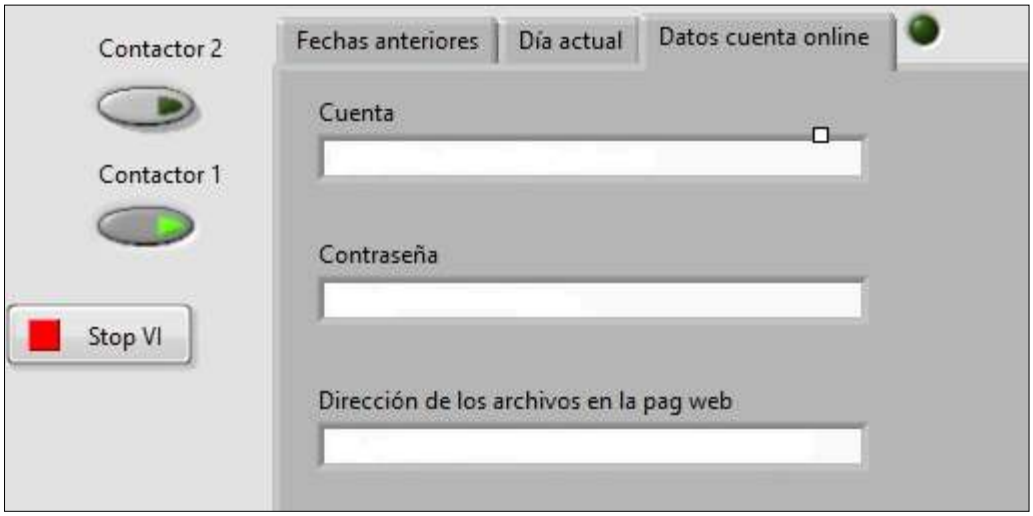


Figura 6.9 Datos cuenta online

La pantalla inicial se muestra de la siguiente manera:

Figura 6.10 Pantalla inicial

Ya se ha explicado cómo usar el menú “Fecha”. El manejo es muy sencillo:

· Cuando se inicie el programa se encenderá el LED verde y aparecerá una frase que indica “Cargando datos…” tal que así:

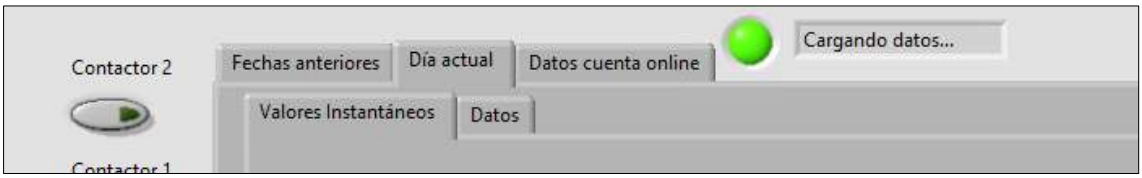


Figura 6.10 Pantalla carga

Hasta que no se quite la frase y se apague el LED es conveniente no tocar el programa.

· El botón “Stop VI” parará el programa cuando haya acabado de realizar todas las operaciones de la iteración, de esta manera se evita un parado forzoso.

· Dos botones llamados “Contactor 1” y “Contactor 2”:

7.6 “Contactor 1”: Activa un contactor con carga resistiva.

7.7 “Contactor 2”: Activa un contactor con carga inductiva.

Estas salidas las elegirá el usuario, pero en este ejemplo serán las explicadas.

Estos dos botones funcionan si se cambia el valor respecto al anterior, es decir, solo se cambian si el estado del botón es diferente del anterior. Al iniciar el programa del ordenador que lo manejará los dos botones estarán en OFF pero si actualmente el programa de NI myRIO-1900 está ejecutándose y se encuentra con las dos variables activadas éstas no cambiarán ya que no he pulsado ningún botón.

Para pasar estas variables a OFF sería necesario pulsar el botón a ON, como tiene el mismo valor que tenía antes seguirá activada, entonces habría que volver a pulsar para pasar el botón y la variable a OFF. Esto es necesario para que no haya cambios imprevistos en la instalación a gestionar al ejecutar el programa.

7. MANUAL DE CONFIGURACIÓN DE HARDWARE Y SOFTWARE

7.1 Configuración WIFI sobre MyRIO.

Para poder conectar nuestro dispositivo en la red, vamos a configurar inicialmente el Wifi de este, para poder estar en la red. Lo podemos llevar a cabo desde dos aplicaciones.

- NI Web-based Configuration & Monitoring
- Measurement & Automation Explorer (MAX)

Nosotros vamos a hacer desde MAX:

- Seleccionamos el dispositivo y con botón derecho seleccionamos Configuración web:

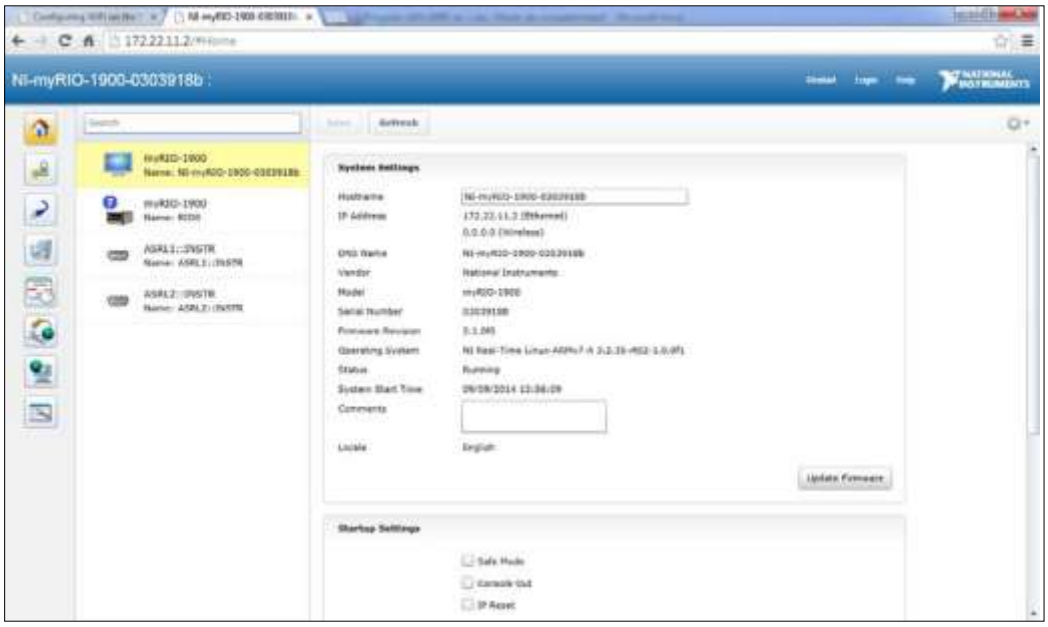


Figura 7.1. Esquema configuración WEB De MyRIO

- Nos vamos a configuración de red pulsando  y configuramos las características de configure WiFi settings in the Wireless Adapter wlan0.
- Seleccionamos Select Connect to wireless network in Wireless Mode.
- In Country, introducimos el país donde está localizado el equipo. Debemos seleccionar antes el país que el canal de red wireless.
- Seleccionamos la red de la que dispongamos, posiblemente sea necesario una contraseña para acceder.
- Seleccionamos DHCP or Link Local in Configure IPv4 Address, y especificamos que NI myRIO adquiera una dirección IP automáticamente.
- Guardamos la configuración WIFI y conectamos NI myRIO a la red wireless que seleccionamos.

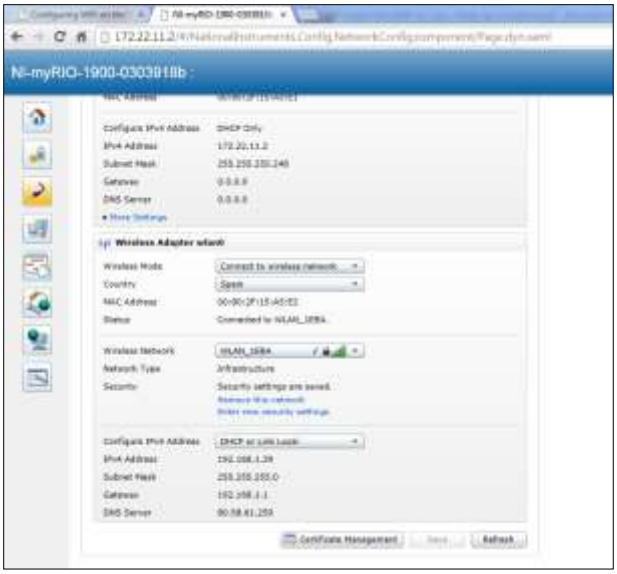


Figura 7.2 Esquema Configuración de las redes de conexión del MyRIO.

7.2 Configuración día y hora en MyRIO.

En la misma configuración web podemos hacerlo, debemos guardar los cambios.

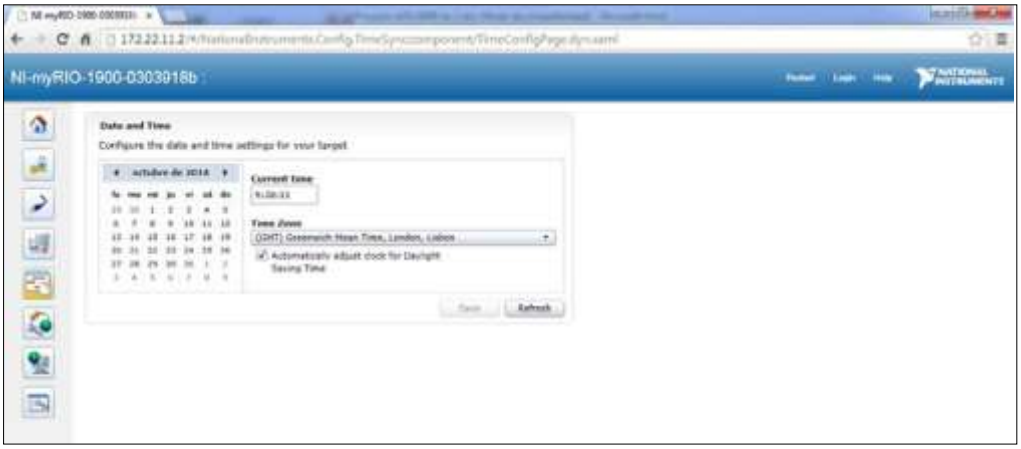


Figura 7.3 Esquema Configuración Horario del MyRIO.

7.3. Configuración IP

Haciendo botón derecho y “Propiedades” se elige la dirección IP del dispositivo:

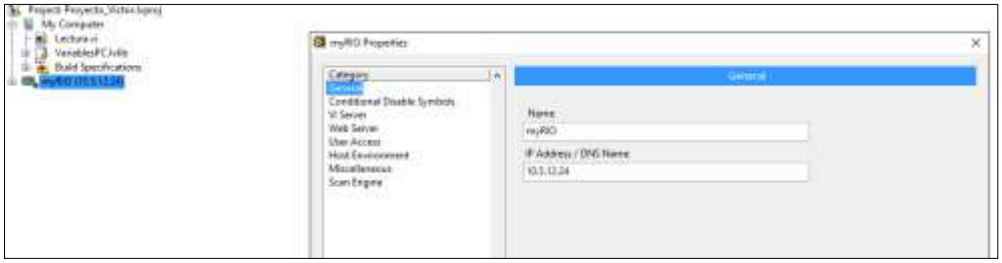


Figura 7.4 Configuración IP

8. Experimentación y resultado

Para comprobar que la programación es correcta se han llevado a cabo varías comprobaciones:

· Conectar una carga resistiva

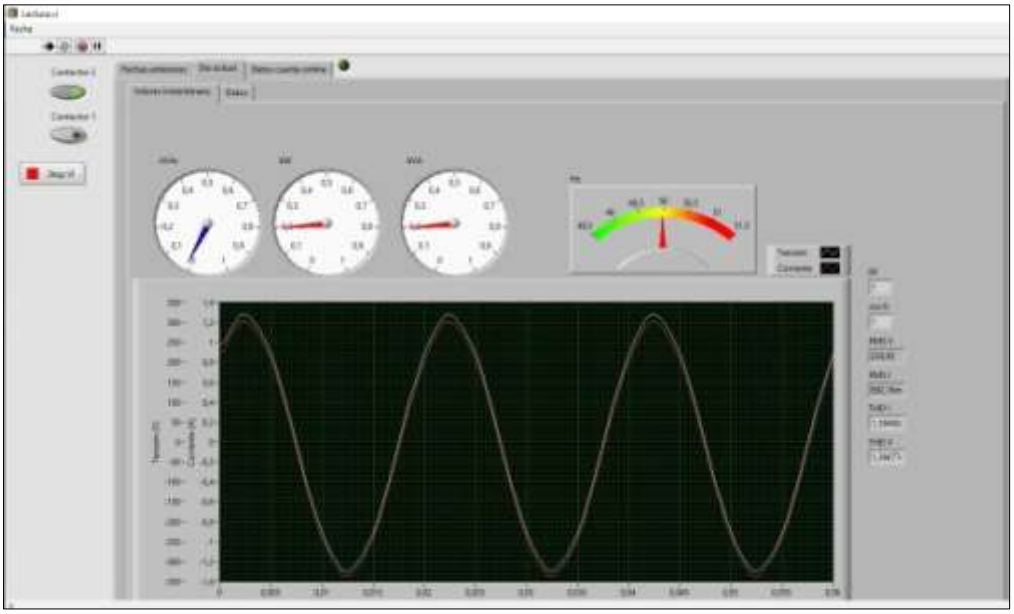


Figura 7.1 Gráfica 1

El valor eficaz de la tensión es 224,82 por lo que es una adquisición correcta. Las THD (%) son muy bajas y la frecuencia se mantiene en 50 Hz. Las formas de onda están en fase, los kVAr son prácticamente 0 y los 0,2 KW coinciden con los 0,2 KVAR. El factor de potencia y el cos(φ) son 1.

· Conectar una carga inductiva

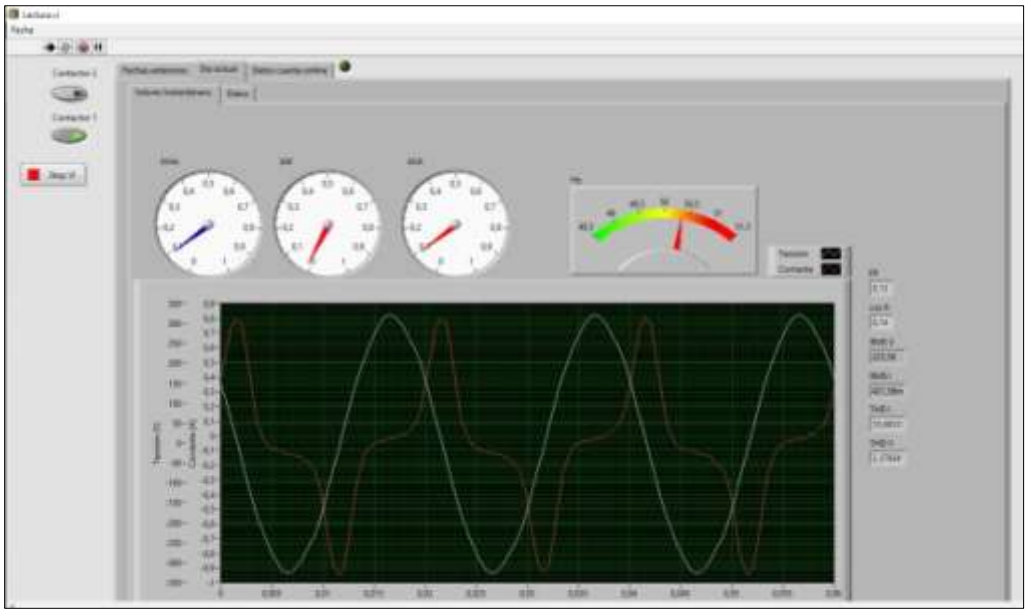


Figura 7.2 Gráfica 2

La intensidad (Rojo) está desfasada de la tensión (blanco) 90º, los KW son prácticamente 0 y los 0,1 KW coinciden con los 0,1 KVAR. El factor de potencia y el cos(φ) son 0,13 y 0,14 respectivamente.

La THD (Tasa de distorsión armónica) de la tensión apenas ha subido pero la de la intensidad ha subido a 11%, algo normal al conectar una carga completamente inductiva.

· Conectar carga inductiva + carga resistiva

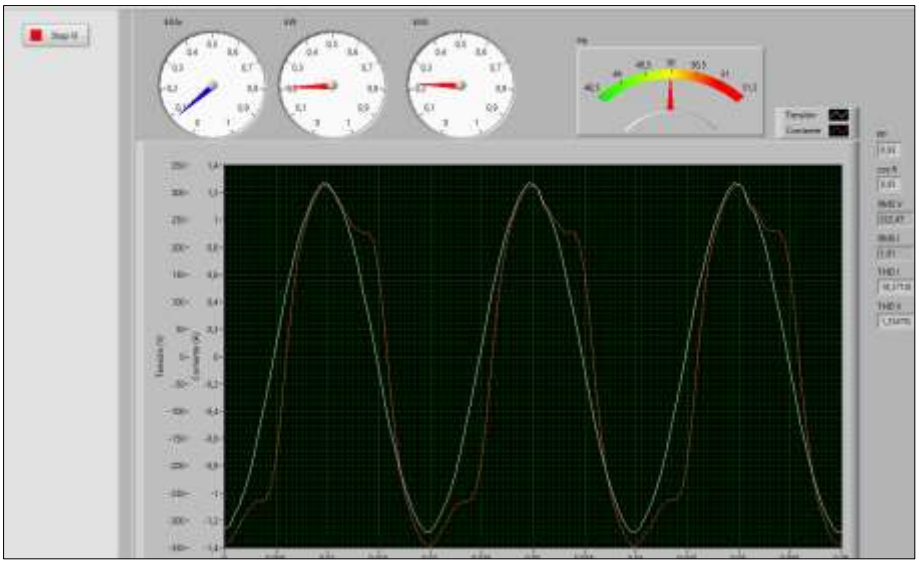


Figura 7.3 Gráfica 3

Vemos que los KVAR se corresponden a 0,1 al igual al conectar solo la carga inductiva, los KW son 0,2 al igual que al conectar solo la carga resistiva y la potencia aparente ronda los 0,22 KVA que se debería corresponder mediante la cuenta:

$$S = \sqrt{P^2 + Q^2} = \sqrt{0{,}2^2 + 0{,}1^2} = 0{,}2236\ldots \text{ KVA}$$

Por lo que el cálculo es correcto.

El resto de valores son normales teniendo en cuenta las cargas conectadas.

· Ejemplo toma de datos de un día cualquiera:

Gráfica Tensión, Intensidad y Factor de Potencia

Como estos datos abarcan un amplio espectro aparece una barra dentro de la gráfica para representar los datos, aquí se mostrarán mediante dos imágenes:

Figura 7.4 Gráfica 4. Blanco (Tensión), rojo (Intensidad) y verde (Factor de potencia)

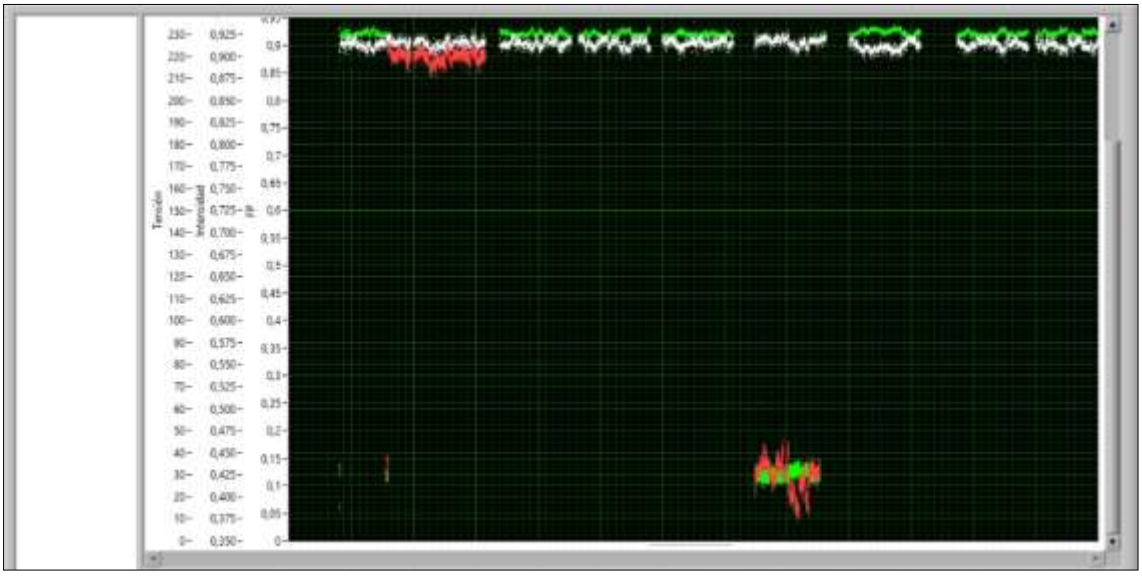


Figura 7.5 Gráfica 5. Blanco (Tensión), rojo (Intensidad) y verde (Factor de potencia)

Por lo visto en anteriores apartados, se deduce que en el primer tramo estarán conectadas las cargas inductiva y resistiva ya que el factor de potencia rona el 0,92 y se ve que es cuando más intensidad se consume con las cargas de las que disponemos en este proyecto.

Al desconectar la carga inductiva baja la intensidad y el factor de potencia sube a 1 como es lógico.

Si solo se conecta la carga inductiva tal y como se ve en la gráfica inferior el factor de potencia será muy bajo y la intensidad también bajará ya que la carga inductiva es bastante más pequeña que la resistiva.

- **Ejemplo de toma de datos de un día cualquiera sin variación de carga:**

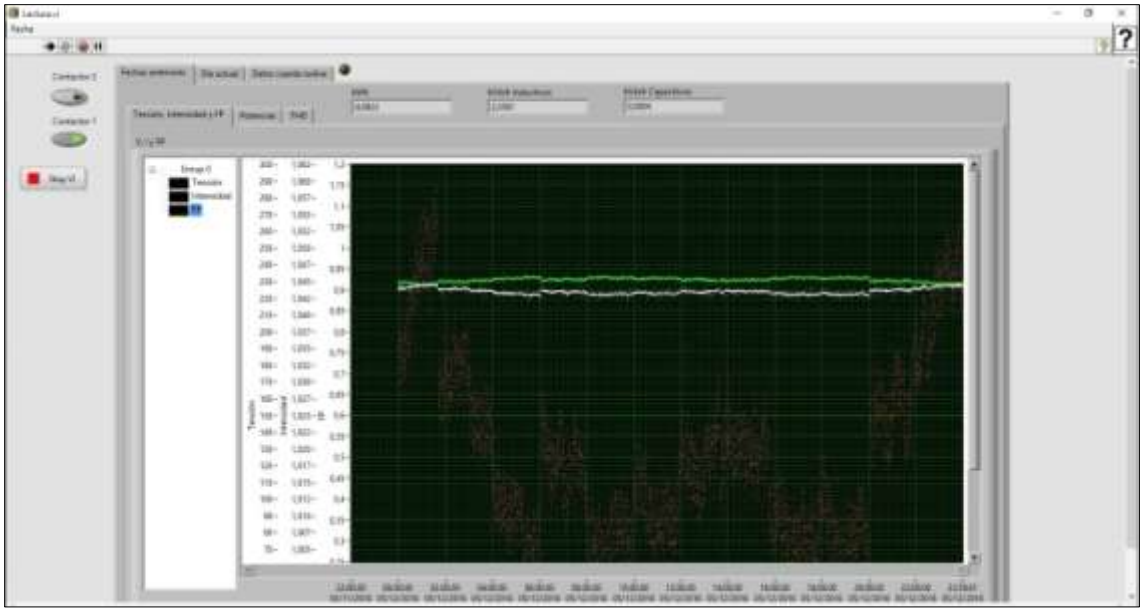


Figura 7.6 Gráfica 6. Blanco (Tensión), rojo (Intensidad) y verde (Factor de potencia)

Corresponde al día 12 de mayo, en las fechas de la gráfica se muestra mediante un formato mm/dd/aaaa porque los datos fueron tomados cuando la programación de la fecha estaba en ese formato, el actual sería dd/mm/aaaaa.
En este caso se ha conectado permanentemente una carga inductiva y capacitiva, los datos son normales, lo único destacable es el aumento de la precisión en el tomado de datos de la intensidad al no variar en exceso los datos.

- Ejemplo de toma de datos con desconexión de NI myRIO-1900

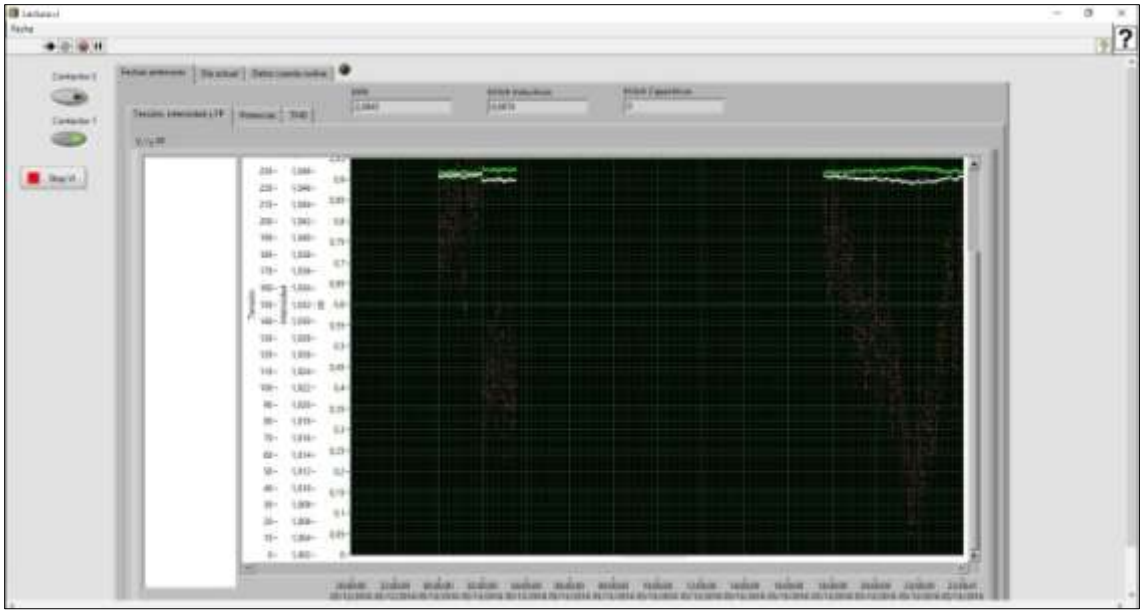


Figura 7.7 Gráfica 7

Al igual que antes, el formato de fecha está a la inversa. Se aprecia que a las 04:00 por la razón que esa (Desconexión de la corriente del laboratorio, error de NI

myRIO-1900…) se produjo una interrupción en la toma de datos. Como los datos se almacenan en fichero en cuanto el error se subsana sobre las 18:00 se vuelven a tomar datos y se almacenan en el mismo fichero hasta las 00:00

- **Todas las gráficas de los datos de un día aleatorio (se muestran dos imágenes por gráfica por la variación de los datos):**

Gráfica Tensión (Blanco), Intensidad (Rojo) y Factor de Potencia (Verde)

Figura 7.8 Gráfica 8 Parte superior de la gráfica

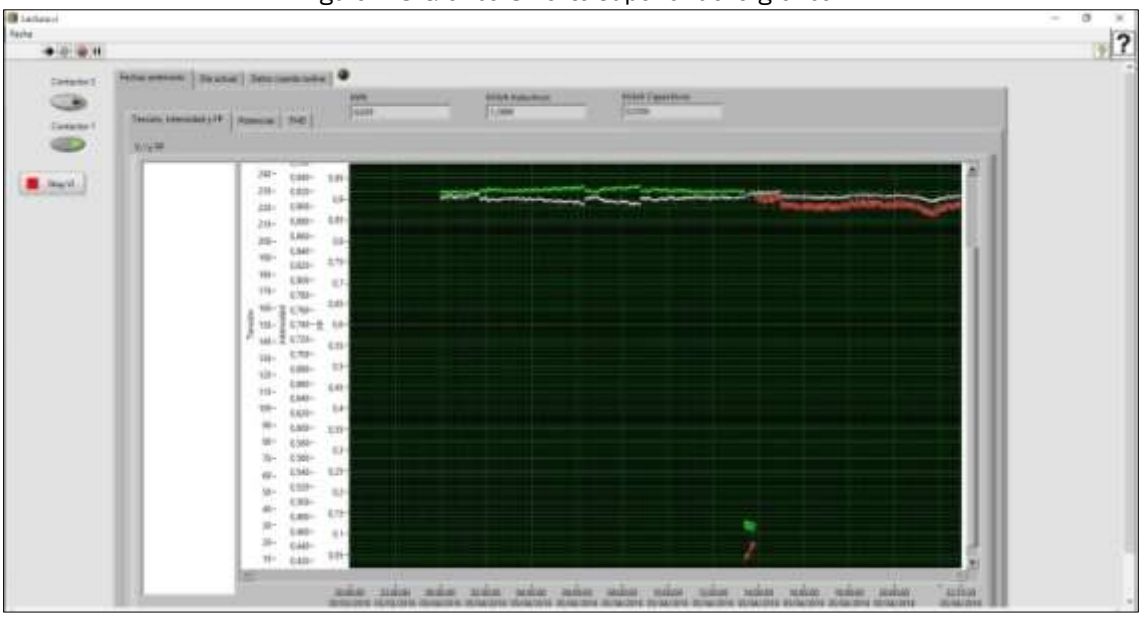


Figura 7.9 Gráfica 8 Parte inferior de la gráfica

Los datos son normales y semejantes a los explicados en apartados anteriores. Primero hay conectada una carga resistiva e inductiva y después una puramente resistiva con un pequeño tramo entremedias de carga puramente inductiva.

· Potencia aparente (Rojo), potencia activa (Blanco) y potencia reactiva (verde)

Figura 7.10 Gráfica 9. Parte superior de la gráfica

En primer lugar al haber conectada una carga resistiva e inductiva como es normal la potencia aparente será la más grande (sobre los 0,22 KVAR), por debajo la potencia activa (0,20 KW) y por último sobre los 0,10 KVAR que se corresponden a los datos que se mencionaron anteriormente por lo que el registro es correcto.
Se observa que al desconectar la carga inductiva y solo tener conectada una carga resistiva los KVA y los KW tienen el mismo valor y los KVAR se reducen a 0.

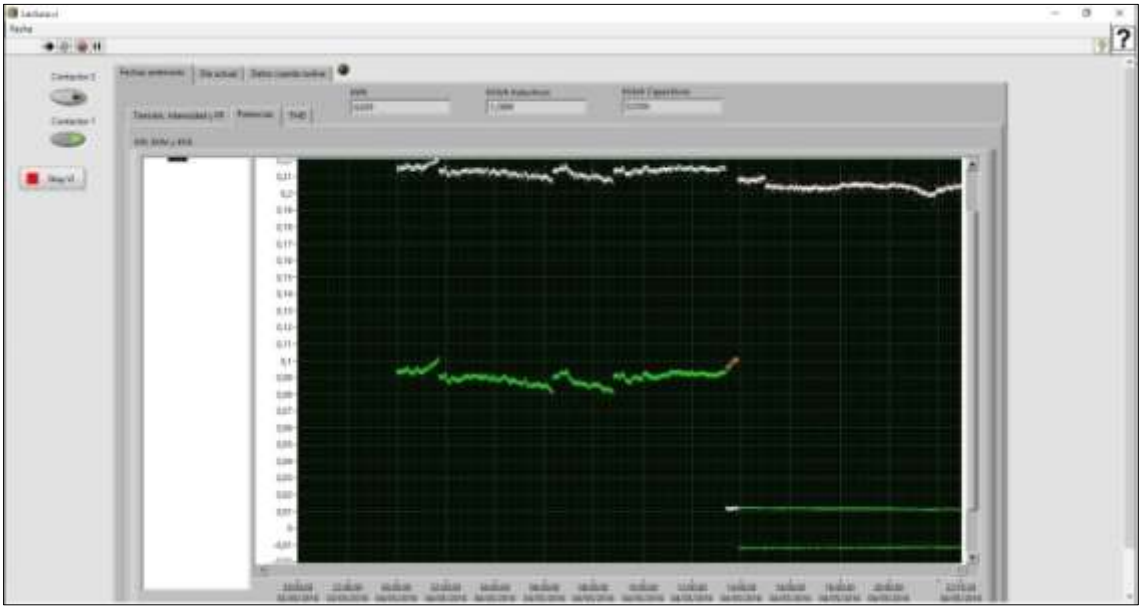


Figura 7.11 Gráfica 9. Parte inferior de la gráfica

Vemos que los KVAR en vez de ser un 0 preciso ronda entre +0,01 y -0,01, es un problema normal ya que los elementos encargados de la adquisición de datos no son absolutamente perfectos y hay una pequeña imprecisión.

· THD de la tensión y de la intensidad

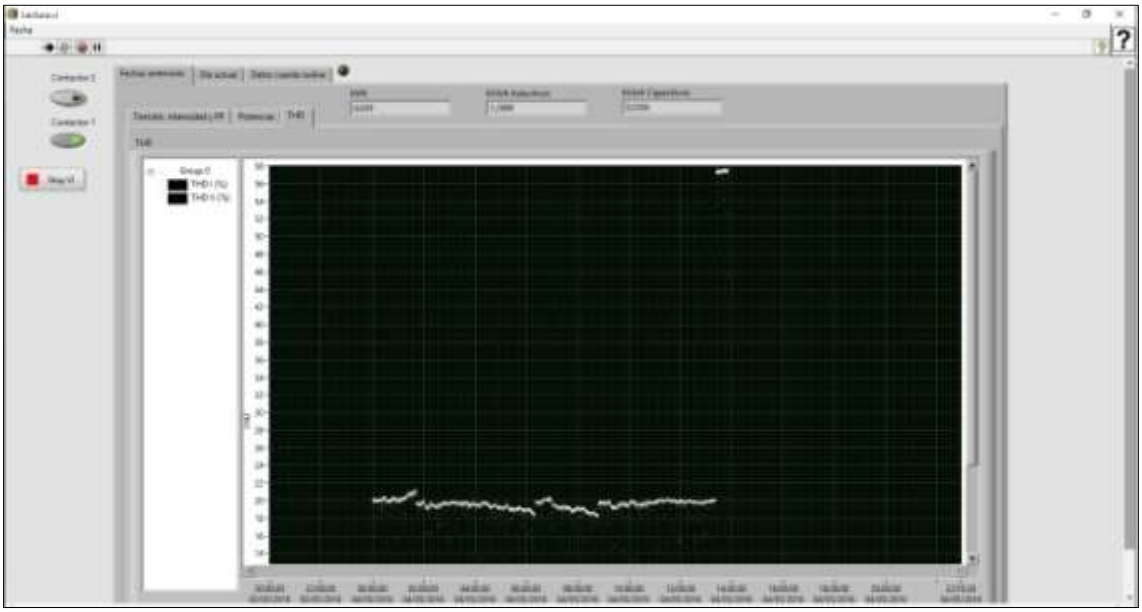


Figura 7.12 Gráfica 10. Parte superior de la gráfica

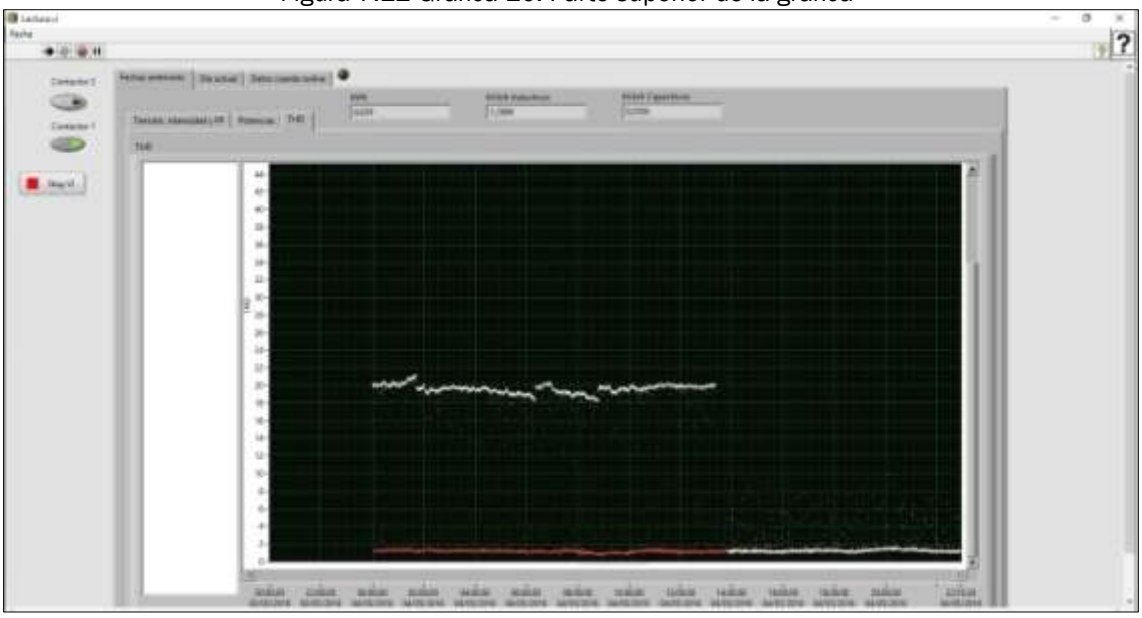


Figura 7.13 Gráfica 10. Parte inferior de la gráfica

Cuando hay conectadas una carga resistiva e inductiva se aprecia que hay en torno al 20% de THD en la intensidad.
En el pequeño tramo aproximadamente a la mitad de la gráfica cuando solo hay carga inductiva la THD de la intensidad sube al 60%.
Al final cuando hay carga puramente resistiva la THD de la intensidad estña en torno al 1%.
La tensión se mantiene durante todo el periodo en torno al 1%:

Mediante el estudio de todas estás gráficas se comprueba que el análisis de los datos ha sido correcto.

· Para comprobar que funciona el recopilado de datos de varios días en una gráfica se selecciona mostrar los datos del año completo.

Los datos se empezaron a tomar sobre el día 3 de mayo, aproximadamente el día 19 se desconectó NI myRIO-1900 y el día 30 de mayo se volvió a conectar. La toma de datos transcurrió sin problemas:

Gráfica Tensión, Intensidad y Factor de potencia

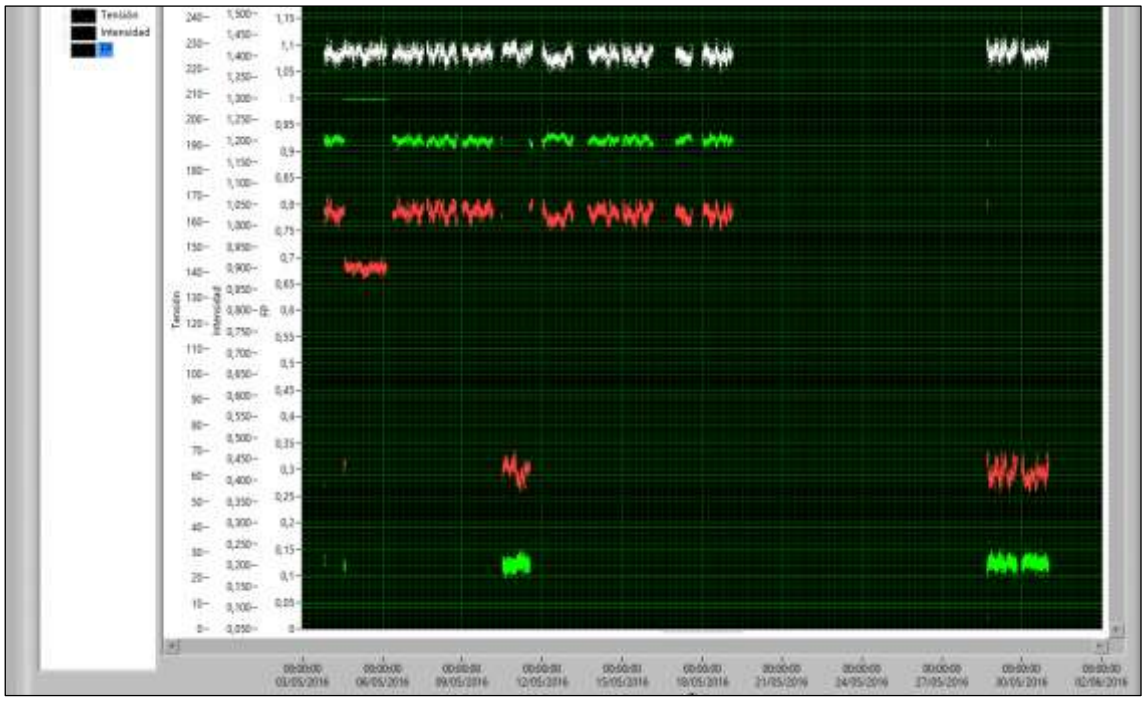


Figura 7.14 Gráfica 11. V, I y FP

Gráfica potencia activa, potencia aparente y potencia reactiva:

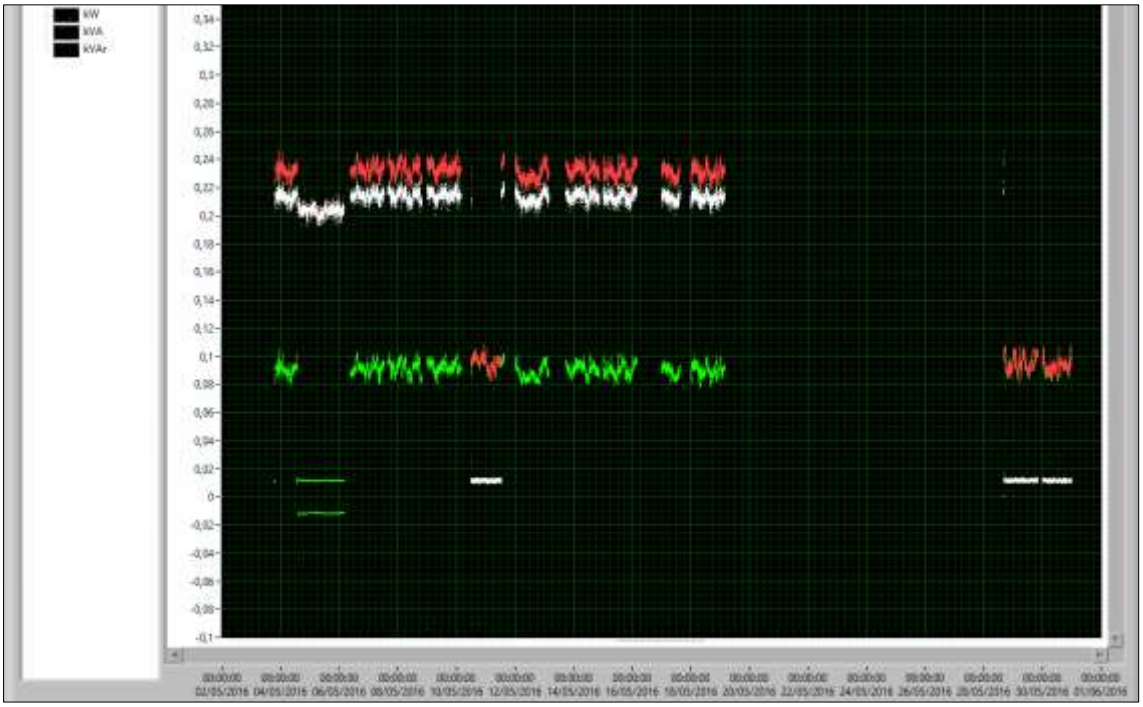


Figura 7.15 Gráfica 15. KW, KVA y KVAR

Gráfica THD de la tensión y la intensidad

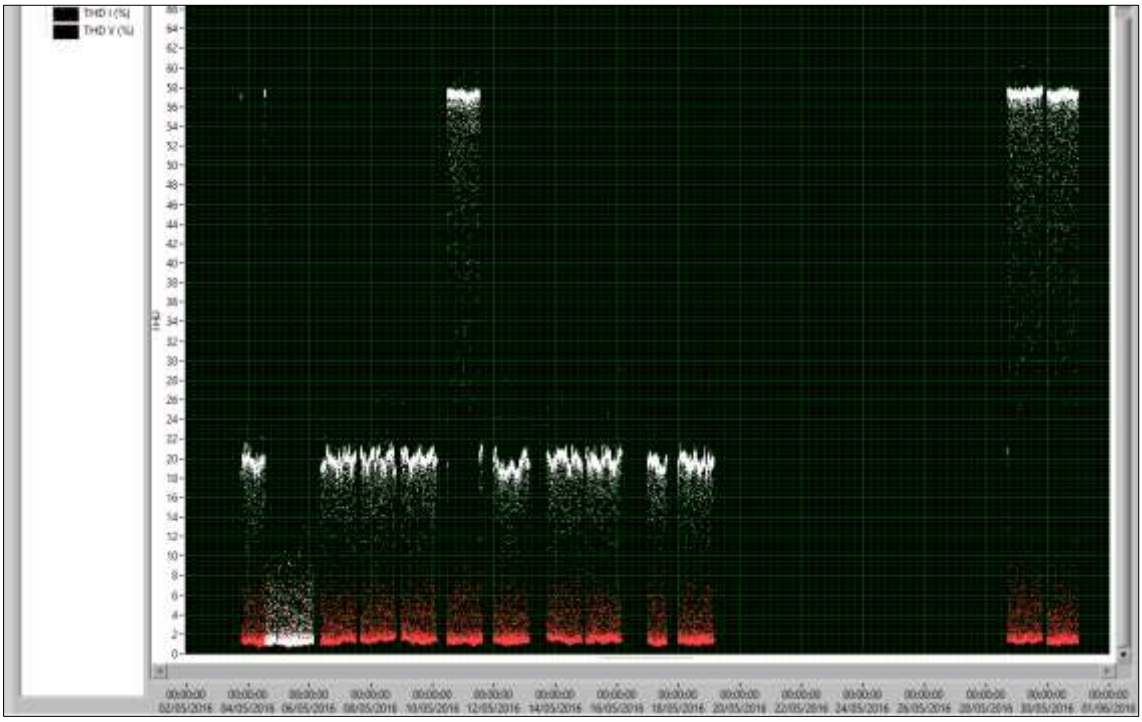


Figura 7.16 Gráfica 16. THD

9. Presupuesto

Dispositivo NI-myRIO	925,62€
Herramienta LabVIEW básica	822,3€
Transductores	100€
Montaje transductores	50€
Mano de obra 1h	50€
Materiales (Cables, desgaste herramientas…)	15€
Desplazamiento a cualquier lugar de España	300€
Total	2262,93€
Total + IVA	**2738,15€**

10. Conclusiones

Con el análisis de los datos del punto 7 se concluye que los resultados obtenidos son lógicos, tanto la toma de datos como su procesamiento y muestra al usuario son correctos. Se han cumplido los objetivos de la creación de la herramienta para la gestión y control del consumo, permitiendo el almacenamiento de los datos en la "nube", el control de varios contactores y la visualización los consumos en tiempo real.

Los datos se muestran al usuario de manera sencilla gráfica y numéricamente, con esto se pueden ver los consumos para optimizar la factura eléctrica visualizando consumos innecesarios a determinadas horas, permite conocer consumos anómalos de los electrodomésticos y visualizando el consumo diario se podría ver cuál es la tarifa más acorde al consumo de nuestra red, contratando la que tenga periodos baratos a las horas de nuestro máximo consumo. Al poder ver los consumos en tiempo real es posible detectar si se están usando más aparatos de los que son necesarios y poder apagarlos, algo que repercutirá en la factura eléctrica.
En definitiva el objetivo sería ahorrar dinero en la factura eléctrica tras el análisis de los consumos.

Poder realizar esta función con únicamente una tarjeta NI myRIO-1900 supone una gran ventaja debido a su gran versatilidad. En actualizaciones futuras del software se podría incluso analizar varios circuitos a la vez o una red trifásica a la vez que también se le pueden añadir funciones complementarias ya que el programa está hecho de tal manera que consume pocos recursos.

Como es normal en programación el programa no está al 100% depurado y su explotación implicaría una mejora continua del programa con actualizaciones sucesivas.

11. Bibliografía

[1]Cursos de Labview: http://spain.ni.com/formacion. [Consulta: 20 a 30 de junio de 2015].

[2]Guía de usuario de NI myRIO-1900. [Consulta: 15 de julio de 2015].

[3]Libro: Eficiencia en el uso de la energía eléctrica. Autores: Josep Balcells, Jordi, Autonell, Vicente Barra, Joan Brossa, Francesc Fornieles, Bernat Garcia, Joan Ros. Editorial marcombo. CIRCUTOR. [Consulta: 25 de septiembre de 2015].

[4]Autor: Juan Miguel García Haro: "Desarrollo de un controlador para motoresDC brushless basado en CompactRIO yLabVIEW de National Instruments para el estudio de nuevos algoritmos de control". Proyecto Fin de Carrera, Universidad Juan Carlos III de Madrid. [Consulta: 20 de mayo de 2016].

[5] https://es.wikipedia.org/wiki/Sistema_operativo_de_tiempo_real. [Consulta: 21 de mayo de 2016].

[6]https://es.wikipedia.org/wiki/Tiempo_real. [Consulta: 21 de mayo de 2016].

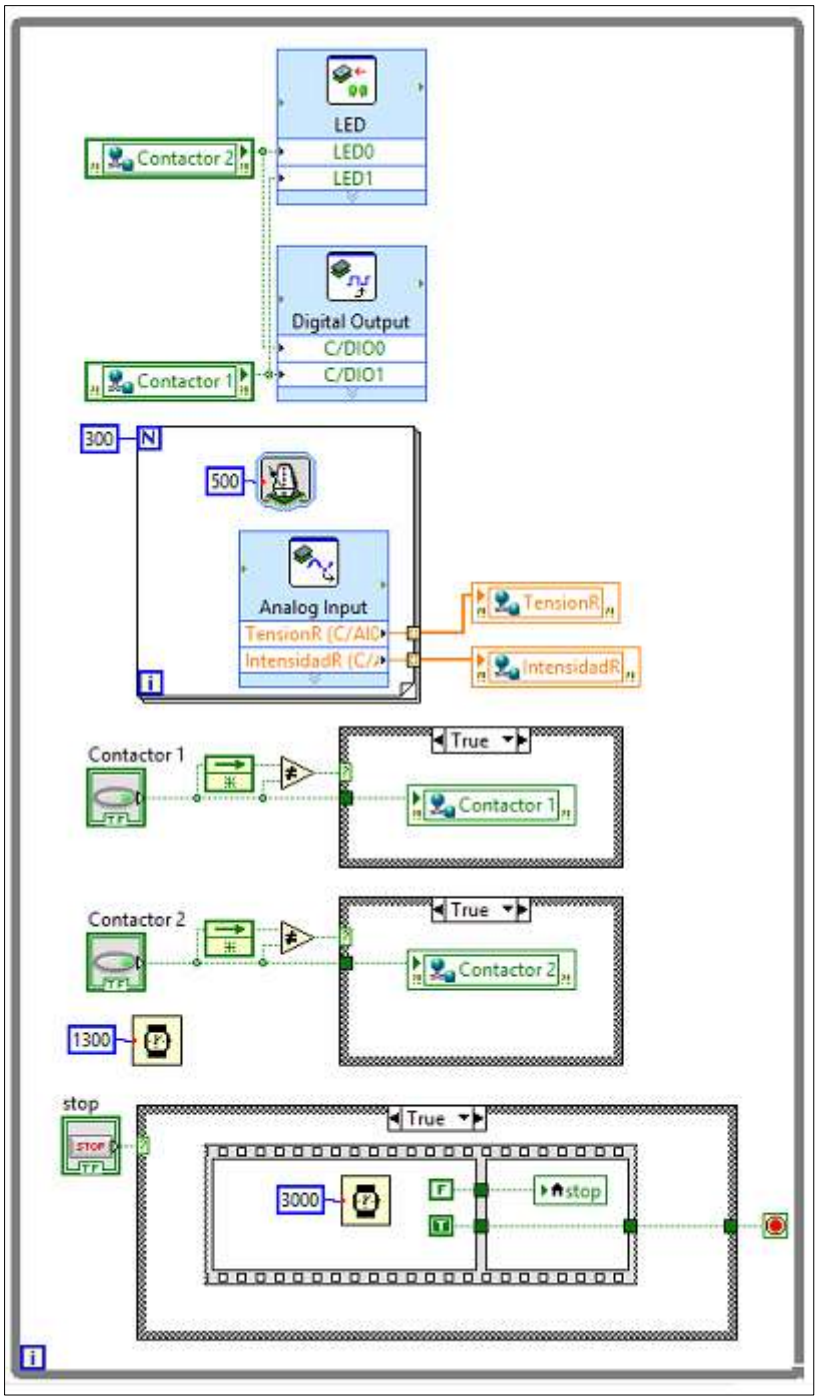
[7]http://www.monografias.com/trabajos37/sistemas-tiempo real/sistemas-tiempo-real2.shtml. [Consulta: 21 de mayo de 2016].

[8]Autor: Antonio José Fernández Agüera: "Aplicación de controladores comerciales para almacenamiento y monitorización remota de variables eléctricas.". Trabajo de Fin de Máster, UNIVERSIDAD POLITÉCNICA DE CARTAGENA. [Consulta: 23 de mayo de 2016].

[9]http://www.monografias.com/trabajos82/analizador-redes/analizador-redes.shtml [Consulta: 23 de mayo de 2016].

ANEXO 1

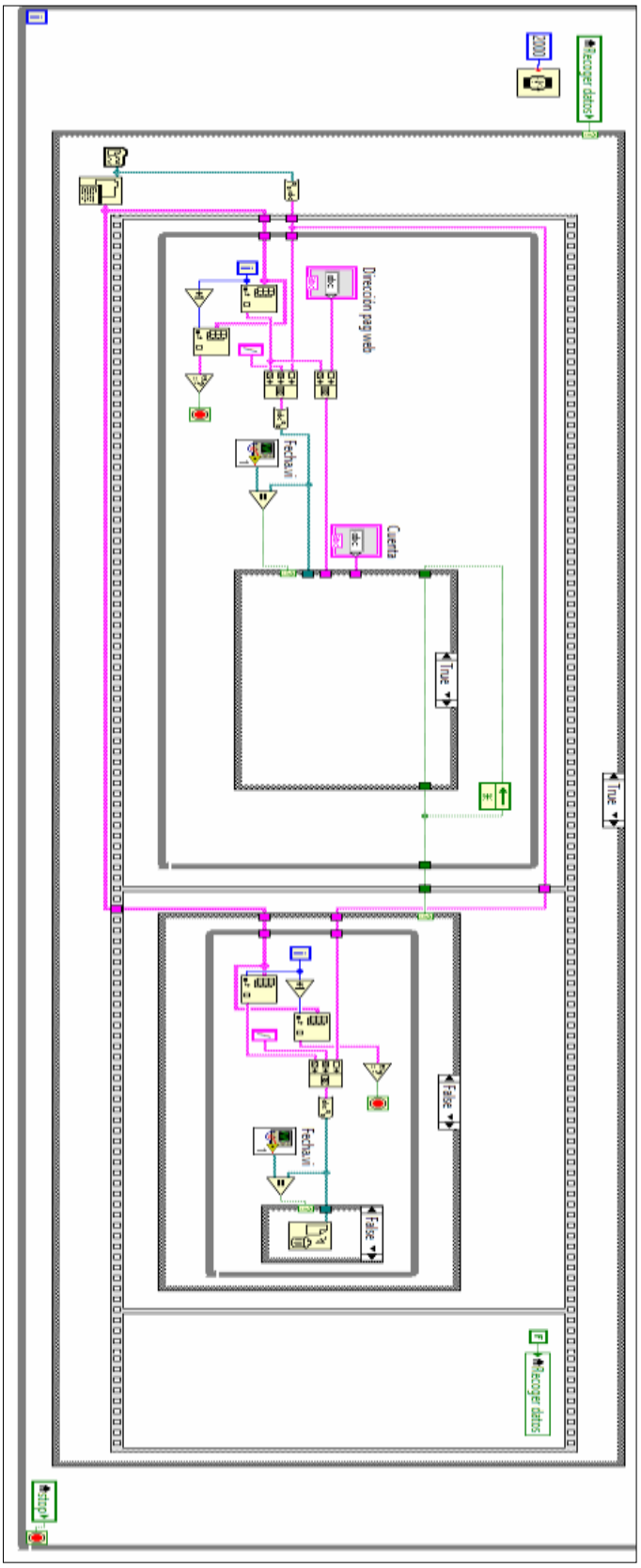
VI: “Adquirir_Datos_MYRIO”. Bucle 1



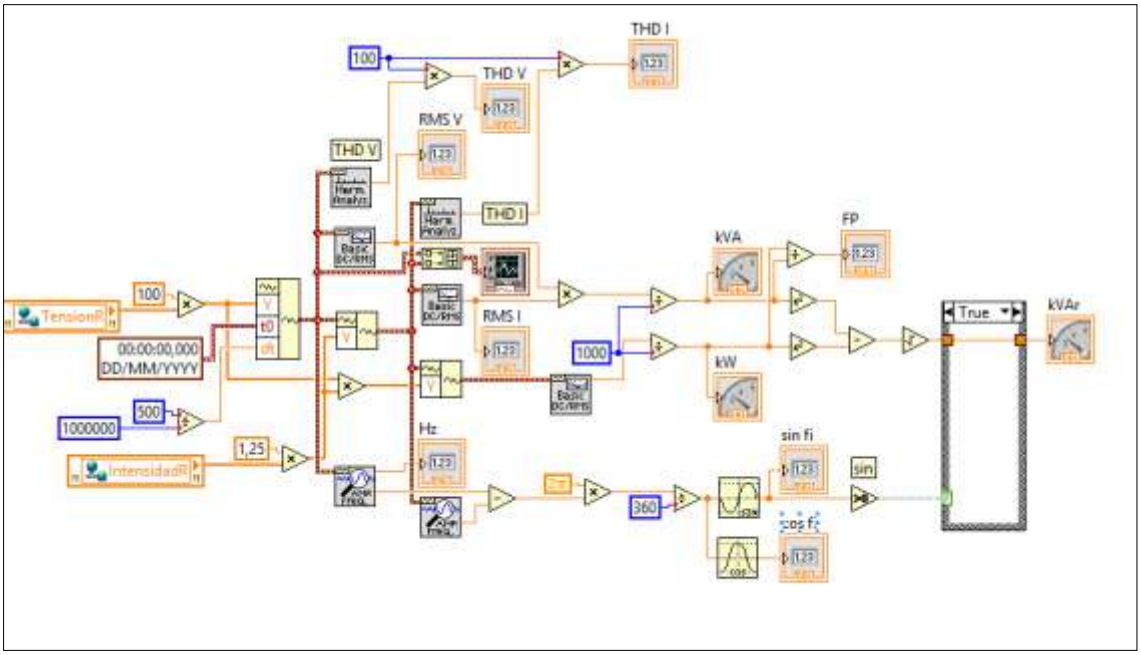
VI: “Adquirir_Datos_MYRIO”. Bucle 2

Debido a su tamaño, consultar código para visualizar esta parte del código.

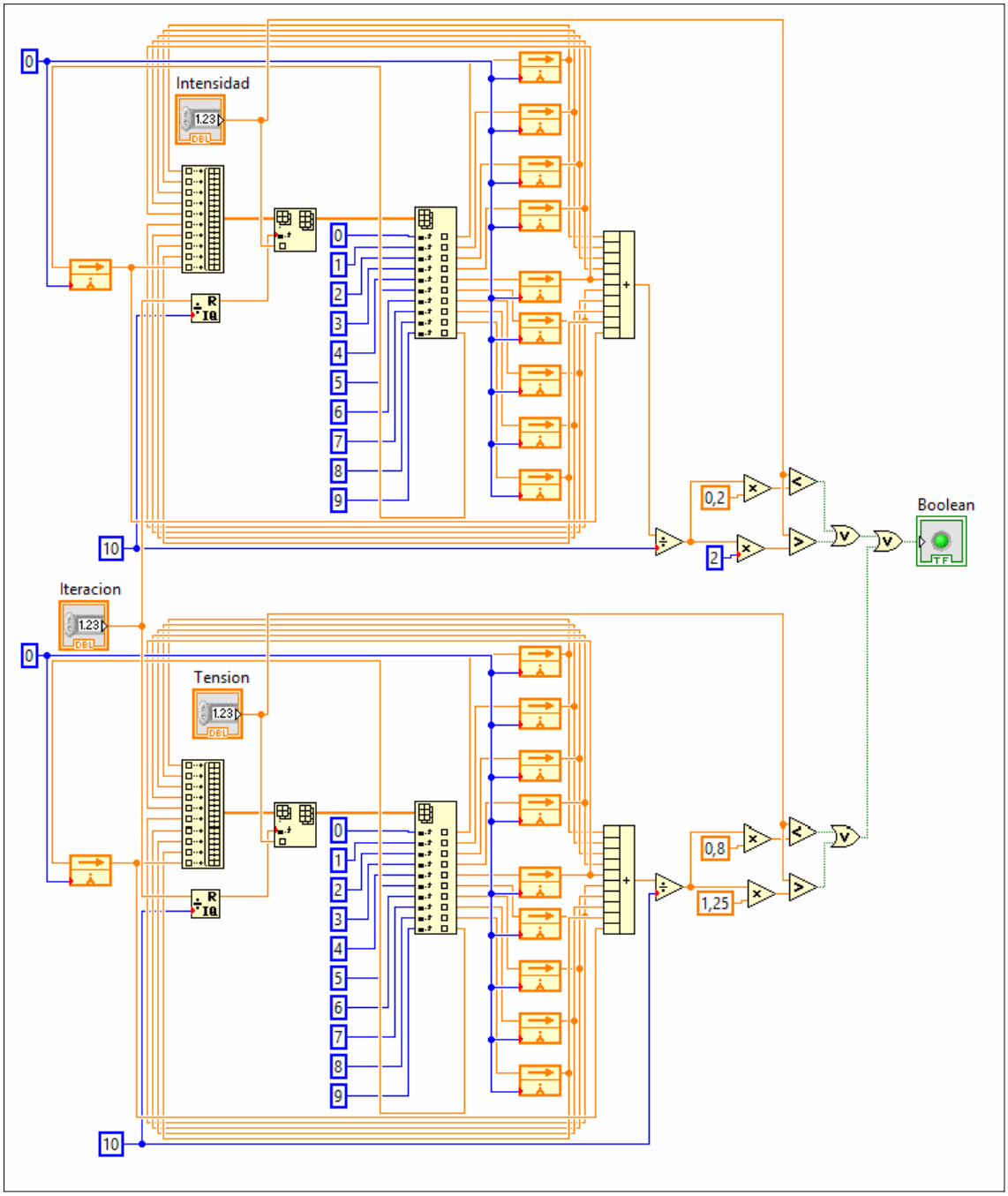
VI: “Adquirir_Datos_MYRIO”. Bucle 3



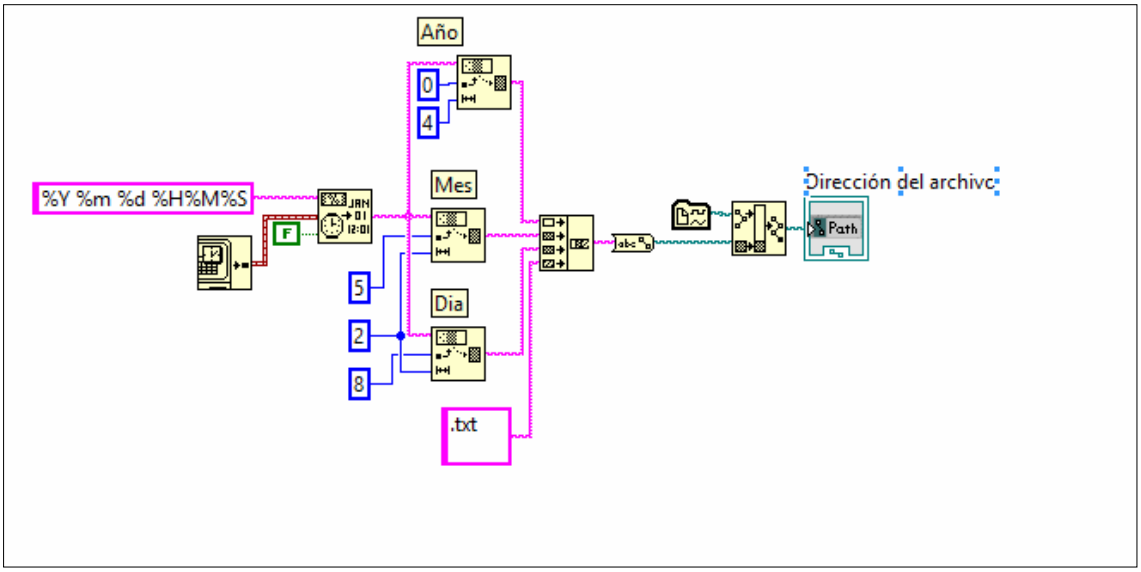
VI: “Datos_Principales”



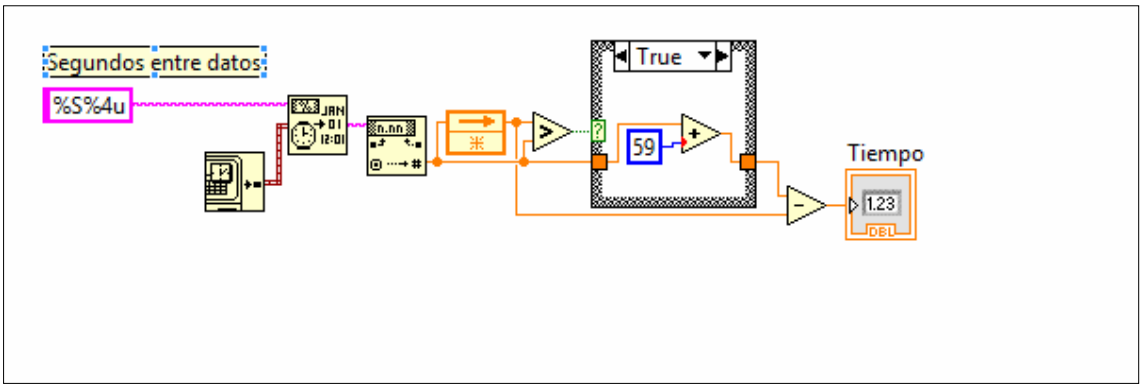
VI “Condicion_tomar_dato”



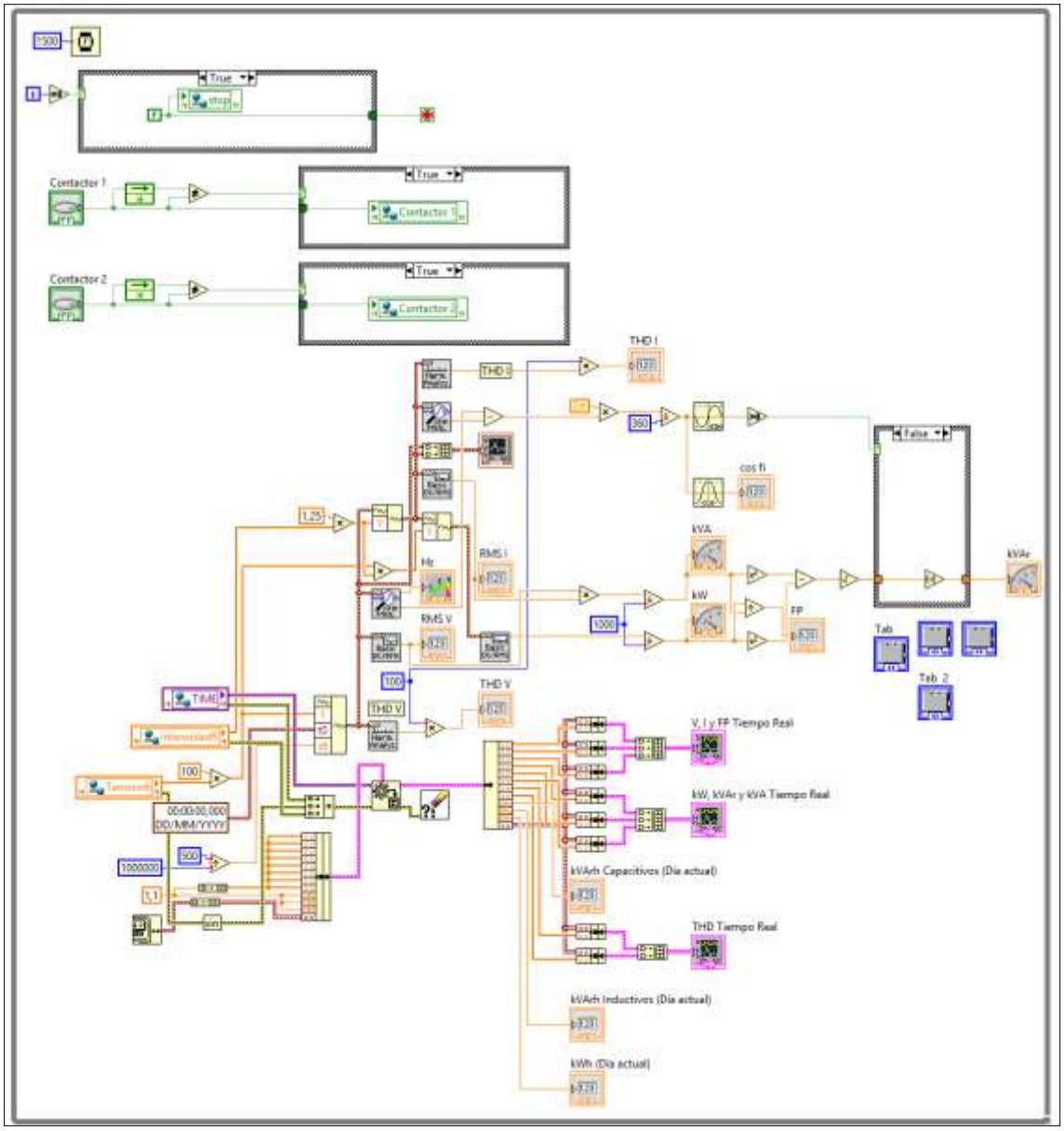
VI “Fecha”



VI “Tiempo_2_ciclos”



VI “Lectura”. Bucle 1



VI “Lectura”. Bucle 2

Debido a su tamaño, consultar código para visualizar esta parte del código.