industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Control escalar de velocidad de una máquina asíncrona utilizando una tarjeta sbRIO-9606

TRABAJO FIN DE GRADO

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR: **Pablo Sánchez Sánchez**
DIRECTOR: **José Antonio Villarejo Mañas**

Universidad Politécnica de Cartagena

Cartagena, Septiembre de 2015

Agradecimientos

Agradecer a la Universidad Politécnica de Cartagena en concreto a la Escuela Técnica Superior de Ingeniería Industrial, así como a su equipo docente por su dedicación y esfuerzo en mi formación como ingeniero.

A mi compañero, Jose David Trapero Díaz, por su ayuda desinteresada en todo lo que me ha hecho falta.

A mi director de este proyecto, José Antonio Villarejo Mañas, por su ayuda, tiempo y dedicación a la hora de llevar a cabo este proyecto.

A mi padre, Pedro Sánchez Rodríguez, por la revisión de este trabajo en cuanto a contenido y forma.

Y por último y no menos importante, a mi familia y mi pareja por haberme apoyado en cada instante y haber tenido paciencia en los momentos más delicados.

ÍNDICE GENERAL

MEMORIA

Capítulo 1

1. Introducción

1.1. Fases de desarrollo

El proyecto se ha desarrollado en cuatro etapas que se describen a continuación:

- En la primera etapa se han estudiado los manuales de uso del programa de entorno gráfico LabVIEW, y se han visualizado los vídeos tutoriales que explican el funcionamiento del mismo.

- En la segunda etapa se han definido las características básicas del programa de control. Para ello se han concretado qué objetos se iban a usar a nivel de hardware (Single-Board RIO, mezzanine card 9606, motor asíncrono,…) como también las librerías a utilizar que posteriormente serán implementadas. En esta etapa también se han realizado una serie de ejemplos de código para comprobar la funcionalidad del hardware, así como de sus puertos de entradas y salidas. Estos ejemplos junto con lo estudiado en la primera etapa nos ayudarán a usar el programa de LabVIEW a un nivel más avanzado.

- En la tercera etapa se ha desarrollado el programa de control. Teniendo en cuenta las distintas características que definen el controlador, se ha diseñado un código de programa que sea capaz de controlar la velocidad del motor en función de la amplitud y frecuencia de la tensión de alimentación introducida por el usuario. Esta etapa estará constituida por varias sub-etapas, ya que a la hora de desarrollar el programa de control se irá construyendo pequeños programas (subprogramas) que la integración de estos conformarán el programa final de control.

- En la cuarta y última etapa, se han realizado una serie de ensayos y verificaciones para validar el código implementado en la etapa anterior así como evidenciar cómo el sistema en su conjunto es capaz de funcionar cuáles son sus condiciones operativas.

1.2.Esquema de la memoria

La memoria de este proyecto se divide en siete partes, sin contar los anexos. Estos capítulos están ordenados de tal forma que se facilite la tarea al lector.

- En el capítulo dos se explica cuáles son los objetivos principales y los objetivos secundarios que se intentan conseguir en este proyecto.

- En el capítulo tres se definen los contenidos teóricos básicos para, posteriormente, entender cómo se ha llevado a cabo el proyecto y el porqué. En este capítulo se describe el entorno de programación gráfica LabVIEW, así como sus librerías a implementar, el funcionamiento de los motores asíncronos, la forma de operar de los inversores trifásicos como *driver* de potencia del motor, y la arquitectura de la Single-Board RIO incluyendo los diferentes componentes que la estructuran.

- En el capítulo cuatro se comenta el contenido principal del proyecto. En él, se explica cómo se han ido desarrollando todas las partes que constituyen el proyecto. Primero se hace el estudio de los manuales de uso del entorno de programación gráfica LabVIEW y visualización de vídeos tutoriales, explicando el funcionamiento del entorno de programación gráfica. Segundo se comenta una serie de ejercicios de ejemplo realizados para comprobar el funcionamiento y comunicaciones con la FPGA[1]. Tercero se desarrolla el programa de control a través del entorno de programación gráfica LabVIEW, destinado a comprobar que es posible hacer funcionar el controlador a través de la elaboración del banco de pruebas. Y cuarta, y última se diseñará una placa de conexiones para poder facilitar el montaje de la instalación.

- En el capítulo cinco se exponen los resultados obtenidos en el banco de pruebas, definiendo los cuatro ensayos aplicados al controlador (tanto sin carga como con carga) y comentando los resultados obtenidos de los propios ensayos.

[1] FPGA: Field Programmable Gate Array

- En el capítulo seis se explica con detalle el coste económico del: hardware, software y mano de obra. Cabe destacar que este es un proyecto docente por lo que los costes serán muy bajos.

- En el capítulo siete y último se explican las conclusiones de este proyecto, a las que se han llegado a través de los resultados de los ensayos de validación. Además se proponen algunas propuestas para mejorar la capacidad del sistema global, de tal manera que en los proyectos futuros se trabaje de una manera más eficiente.

Capítulo 2

2. Objetivos

Este Trabajo Final de Grado consiste en la profundización de conocimientos en materia relacionada con la electrónica de potencia y máquinas eléctricas.

La tecnología a la que se hace referencia está relacionada con el control de actuadores para motores asíncronos, aplicando el sistema de hardware de la Single-Board Rio y programación LabVIEW de National Instrument.

El objetivo final de este Trabajo Final de Grado es el control de la velocidad de giro de un motor asíncrono trifásico mediante la regulación de un par de variables eléctricas, como son la frecuencia y la amplitud de la tensión de alimentación del motor. Esto se llevará a cabo a través de la programación en LabVIEW sobre una sbRIO 9606 y su expansión. Se realizará un programa que funcionando en tiempo real sea capaz de controlar la velocidad de un motor asíncrono trifásico.

Dividiremos este proyecto en cuatro partes. De estas cuatro partes, las tres últimas nos definirán los requisitos del proyecto, los cuales se explican a continuación:

1. Aprendizaje y desarrollo de un nivel alto de programación del lenguaje G, que se necesita para elaborar cualquier tipo de programa a través del entorno de programación gráfica llamado LabVIEW, proporcionado por National Instruments. El curso de aprendizaje consiste en dominar los manuales de usuario suministrados por la propia empresa National Instruments y la visualización de vídeo tutoriales facilitados por la propia web de National Instrument.

2. Desarrollo de un programa específico para el control de motores asíncronos. Este programa a su vez se divide en dos subjetivos. El primero será en realizar un programa

que controle la velocidad de un motor asíncrono atendiendo a la creación de un PWM[2] que compare las señales senoidales de control con una señal triangular. El segundo objetivo será elaborar el mismo programa utilizando la tecnología del SVPWM[3] que se explica posteriormente en el marco teórico.

3. Montaje del circuito de potencia a través de una placa de conexiones que se diseñará "ad hoc".

4. Por último, se desarrollan los ensayos de validación del diseño-. La aplicación se realiza sobre un motor asíncrono en un banco de pruebas y se le somete a distintas condiciones para calcular factores de trabajo.

El sistema deberá de realizar las siguientes tareas:

1. Programación de rampa de arranque en función de los parámetros de la máquina y la carga.
2. Configuración de rampa de aceleración en función de los parámetros de la máquina y la carga.
3. Selección de sentido de giro.
4. Programación de secuencia de encendido y parada.
5. Protecciones y control: Sobrecorriente y temperatura.

[2] PWM: Pulse-With Modulation

[3] SVPWM: Space Vector Pulse With Modulation

Capítulo 3

3. Marco teórico

3.1.National Instrument y su entorno de programación LabVIEW

3.1.1. Introducción

National Instruments, o NI, es una empresa americana con sede en Austin, Texas. Fue fundada en 1976 y desde entonces se dedica a producir equipos automatizados de prueba y programas de instrumentación virtual. Las aplicaciones más comunes incluyen la adquisición de datos, el control de instrumentos y la visión artificial

3.1.2. Instrumentos de medida

La creciente complejidad de los dispositivos y convergencia de la tecnología están impulsando a los sistemas de pruebas para ser más flexibles, mientras que las presiones de los costes están exigiendo mayor vida útil del sistema. Una arquitectura modular y definida por software es la única manera de cumplir con estos requisitos. La instrumentación modular usa componentes compartidos, buses de alta velocidad y software abierto definido por usuario para cubrir las necesidades de los equipos de pruebas automatizadas (ATE) de hoy en día y a futuro.

3.1.3. Software

El software de National Instrument ha posibilitado tanto a técnicos como investigadores a aumentar la producción y reducir los costes. El producto estrella de National Instrument es Labview™, esta herramienta es una de las más utilizadas en la rama industrial, esto lo demuestra en sus distintas nominaciones como “Test Product of the Year” por los lectores de Test & Measurement World en 1993 y 1999.

3.1.4. Instrumentación

A finales de la década de los 90s, National Instrument desarrolló su primer producto, GPIB[4], una tarjeta de interfaz que permitía interconectar instrumentos tradicionales con un ordenador. A partir de este momento la compañía ha continuado ampliando y mejorando su gama de productos hasta el desarrollo de instrumentos de altas prestaciones basados en PC[5] o modulares.

3.1.5. Adquisición de datos

Aprovechando el procesador de los ordenadores personales, el hardware de National Instrument para la adquisición de datos y el acondicionamiento de señal permitía adquirir y analizar datos físicos tales como la presión, la temperatura y la vibración. La interacción entre el software y el hardware de National Instrument para la adquisición de datos generando los mencionados "instrumentos virtuales" que sustituyen a los tradicionales ofreciendo unas soluciones más flexibles y unos costes más reducidos.

3.1.6. Adquisición de imágenes y control de ejes

En 1997 National Instrument aumentó su línea de productos de adquisición de datos, incorporando nuevos productos para la adquisición de imágenes y control de ejes. En este caso, el hardware y el software de National Instruments facilitan el desarrollo de soluciones integradas para todos los clientes que operan en esos sectores.

3.1.7. Redes industriales.

Ya sea para comunicarse con dispositivos como instrumentos de proceso, controladores lógicos programables (PLC[6]s), sensores inteligentes y controladores de un solo ciclo o para realizar control de instrumentos desde su PC, National Instruments ofrece una variedad de

[4] GPIB: General Purpose Instrumentation Bus

[5] PC: Personal Computer

[6] PLC: Programmable Logic Controller

herramientas de hardware y software confiables y fáciles de usar para satisfacer las necesidades de comunicación.

3.1.8. El punto fuerte

El ordenador personal ha enriquecido nuestra forma de trabajar. Desde el ámbito profesional al académico, el PC garantiza un acceso rápido a cualquier información en cualquier parte del mundo y permite utilizar instrumentos *hardware* y *software* para aumentar la productividad.

El objetivo de National Instruments es ofrecer a sus clientes los instrumentos *hardware* y *software* que necesitan para satisfacer sus exigencias de medida y automatización.

Ya sea con ordenadores portátiles estándar para controlar laboratorios en un entorno hostil o con PC's modulares especiales para la monitorización de materiales de gran importancia y valor, National Instruments ofrece la solución de medida y control más adecuada.

Por todo ello se puede afirmar que el PC constituye la base de la estrategia de control.

Todos los productos National Instruments, de un modo u otro, aprovechan las mejoras continuas en términos de velocidad y prestaciones de la industria informática, lo que se traduce, desde el punto de vista del cliente, en mejores resultados con un coste inferior.

3.1.9. Páginas web desarrolladas por National Instrument

- **Página Web National Instruments (ni.com)**

Esta es la página web principal de National Instruments. Desde aquí se puede acceder a una información más detallada de sus productos tanto de hardware como de software. Desde esta página web también podemos acceder a su tienda online en el que podemos filtrar los productos según las necesidades facilitando información sobre sus complementos tanto a nivel de hardware como del software requerido y cursos para una mayor optimización de los recursos adquiridos.

- **NI Developer Zone (zone.ni.com)**

Esta es la página del soporte técnico. Aquí se pueden encontrar los siguientes manuales de los productos, documentos de soporte, controladores y actualizaciones, tutoriales, códigos de ejemplos y foros de debate. Esta página ofrece soporte a los productos de National Instruments tanto a nivel de hardware como de software.

- **MyNI (my.ni.com)**

Esta página facilita la comunicación con National Instruments. Aquí se puede crear un perfil para poder comentar en el foro y administrar las preferencias específicas de cada utilizador.

Desde esta web se puede acceder a los productos donde se puede hacer descargas de software como activar licencias o ampliar las mismas.

Se puede acceder al soporte técnico tanto de software, como de controlador de hardware y del propio hardware.

Los usuarios de MyNI pueden además investigar dentro de diversas áreas del sitio web y visualizar, de acuerdo con el área geográfica donde residan, el nombre y dirección de la sede de National Instruments más cercana y el calendario de eventos programados para su zona.

3.1.10. Single-Board RIO

3.1.10.1. Introducción

Los nuevos productos NI Single-Board RIO amplían la familia de opciones de implantación de NI RIO al hardware embebido a nivel de tarjeta y de bajo costo. Una vez más, utilizando la arquitectura estándar de RIO NI y LabVIEW, se pueden crear rápidamente prototipos de sistemas embebidos de forma modular y flexible con CompactRIO y descargarlos rápidamente sobre el nuevo hardware embebido a nivel de tarjeta y de bajo costo NI Single-Board RIO. Puesto que es posible reutilizar el mismo código de LabVIEW desde la creación del prototipo hasta la implantación, se puede acortar el tiempo de lanzamiento e incrementar la fiabilidad del

dispositivo embebido y de la máquina. Los nuevos productos NI Single-Board RIO ofrecen las siguientes características:

- Adquisición y control embebido en una sola tarjeta.
- Programación gráfica mediante LabVIEW y herramientas de drivers para la conexión de programas y aplicaciones con el fin de acelerar el desarrollo.
- Procesador incorporado de tiempo real para un funcionamiento fiable e independiente y el procesamiento de señal.
- Chip incorporado de FPGA para la personalización del procesamiento y de la sincronización de las E/S[7].
- E/S analógicas y digitales incorporadas.
- Sistemas de bajo coste para el diseño de sistemas embebidos a nivel de tarjeta.

Cada dispositivo NI Single-Board RIO integra un procesador en tiempo real embebido, una FPGA de alto rendimiento y E/S analógicas y digitales en una sola tarjeta. Al igual que el resto del hardware RIO de NI, las E/S se conectan directamente a la FPGA, proporcionando una personalización a bajo nivel de la temporización y del procesamiento de las señales de las E/S. La FPGA está conectada al procesador embebido de tiempo real a través de un bus PCI[8] de alta velocidad. LabVIEW contiene mecanismos de transferencia de datos para transmitir los datos de las E/S a la FPGA y de la FPGA al procesador embebido para el análisis en tiempo real, el post-procesado, el registro de datos o la comunicación con el ordenador host conectado en red.

[7] E/S: Entradas / Salidas

[8] PCI: Peripheral Component Interconnect

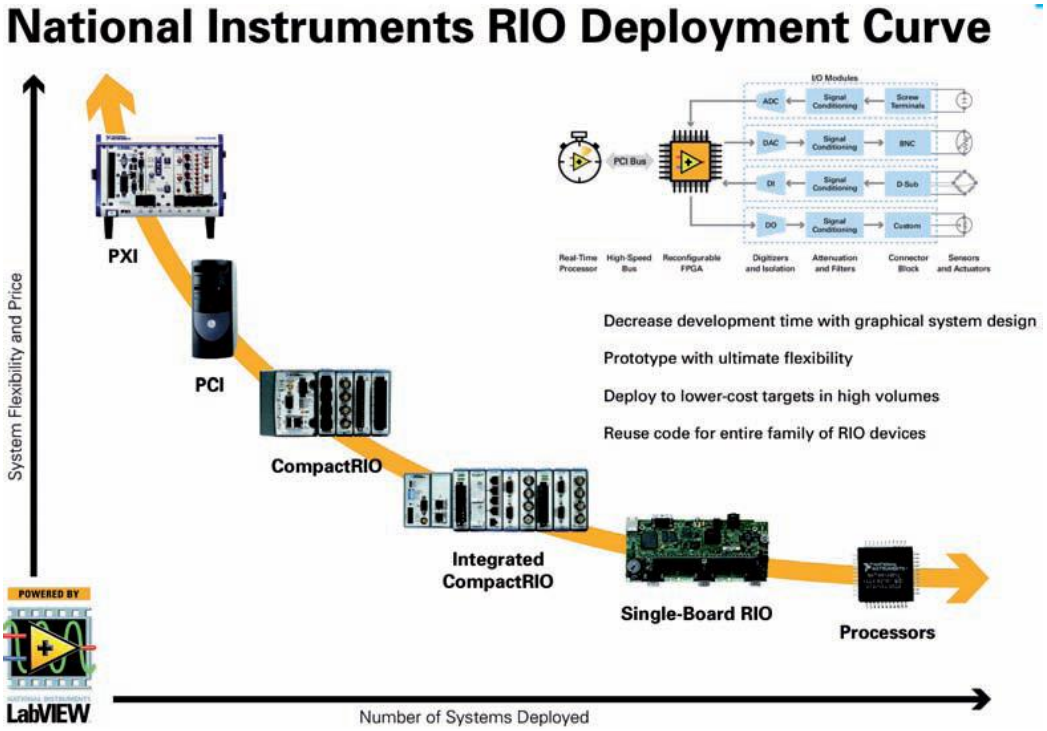


Fig. 3.1. Curva de despliegue de NI RIO: Desde la creación rápida de prototipos a la implantación de bajo costo.

3.1.10.2. Definición

La plataforma NI Single-Board RIO combina dispositivos embebidos de despliegue que tienen un procesador en tiempo real, arreglo de compuertas programable en campo (FPGA) y E/S analógica y digital en una sola tarjeta programada con el software NI LabVIEW. Se puede incrementar la E/S analógica y digital integrada usando módulos de la Serie C. Los dispositivos NI Single-Board RIO están diseñados para aplicaciones OEM[9] embebidas de control y adquisición y de alto volumen que requieren alto rendimiento y fiabilidad

3.1.10.3. Sistema base

La opción más pequeña para NI Single-Board RIO combina el procesador en tiempo real del más alto rendimiento con un FPGA Xilinx Spartan-6 y periféricos integrados como USB[10],

[9] OEM: Original Equipment Manufacturer

[10] USB: Universal Serial Bus

RS232[11], CAN[12] y Ethernet. Además de los periféricos, el sistema incluye 96 líneas de E/S digital de FPGA a las cuales se tiene acceso a través del conector de Tarjeta RIO Mezzanine, el cual es un conector de alta densidad y alto ancho de banda que permite acceso directo al FPGA y procesador

Fig. 3.2. Single-Board RIO NI9606.

3.1.10.4. Expansión

La NI 9683 es una tarjeta de E/S analógica y digitales múltiples para cualquier dispositivo NI Single-Board RIO. Se puede conectar todas las entradas y salidas a las tarjetas controladoras NI Single-Board RIO a través del conector para Tarjetas RIO Mezzanine (RMC[13]).

La NI 9683 ofrece conexiones para 16 canales de entrada analógica simultáneos con referencia a tierra aislada; ocho canales de entrada analógica escaneada; ocho canales de salida analógica, todos con protección para sobre voltaje de ±30 V; 28 canales de entrada digital tipo sourcing muestreados simultáneamente; 14 canales push-pull de salida digital de medio puente;

[11] RS232: Recommended Standard 232

[12] CAN: Controller Area Network

[13] RMC: Rack Mount Chassis

24 canales de salida digital tipo sinking; cuatro canales de salida digital para control de relés y 32 canales de E/S digital LVTTL[14].

Con la versatilidad de los canales de E/S de la NI 9683, se puede usar esta tarjeta en una variedad de aplicaciones industriales desde comunicación con dispositivos industriales como solenoides, actuadores y relés, hasta electrónica de potencia y control de motores

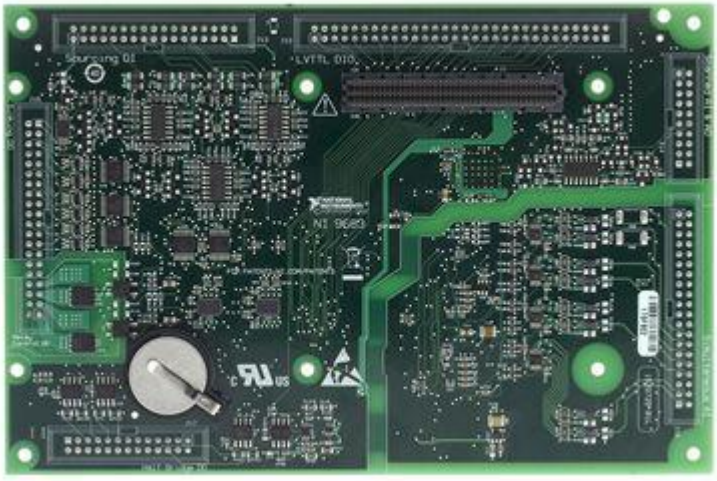


Fig. 3.3. RIO mezzanine NI9683.

[14] LVTTL: Low Voltage Transistor Transistor Logic

3.1.11. LabVIEW

3.1.11.1. ¿Qué es LabVIEW?

LabVIEW (acrónimo de Laboratory Virtual Instrumentation Engineering Workbench) es una plataforma y entorno de desarrollo para diseñar sistemas, con un lenguaje de programación visual gráfico. Recomendado para sistemas hardware y software de pruebas, control y diseño, simulado o real y embebido, pues acelera la productividad. El lenguaje que usa se llama lenguaje G, donde la G simboliza que es lenguaje Gráfico.

3.1.11.2. Introducción

Durante más de 20 años, NI LabVIEW se ha 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.

3.1.11.3. Historia sobre la búsqueda de un lenguaje de programación de más alto nivel.

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 años 50, un reducido grupo de ingenieros 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++.

3.1.11.4. LabVIEW: Programación gráfica

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. 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, ciclos, eventos, variables, recursividad y programación orientada a objetos.

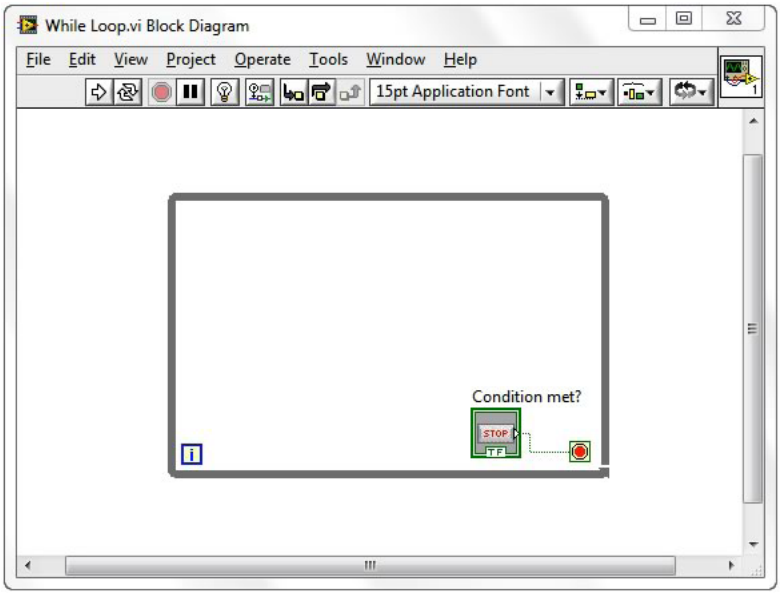


Fig. 3.4. Un bucle While en G se representa por un lazo gráfico que se ejecuta hasta que se cumpla una condición de parada.

El segundo diferenciador principal es que el código G desarrollado en LabVIEW se ejecuta de acuerdo con las reglas del flujo de datos en lugar del acercamiento más tradicional (en otros términos, una serie secuencial de comandos para ser llevados a cabo) que se encuentran 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 ciclos 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 flujo de datos. Un nodo que recibe datos de otro, se puede ejecutar solo después de que el primero complete su ejecución. [10]

3.1.11.5. 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.

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 diagramas de flujo (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. Por ejemplo, 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.

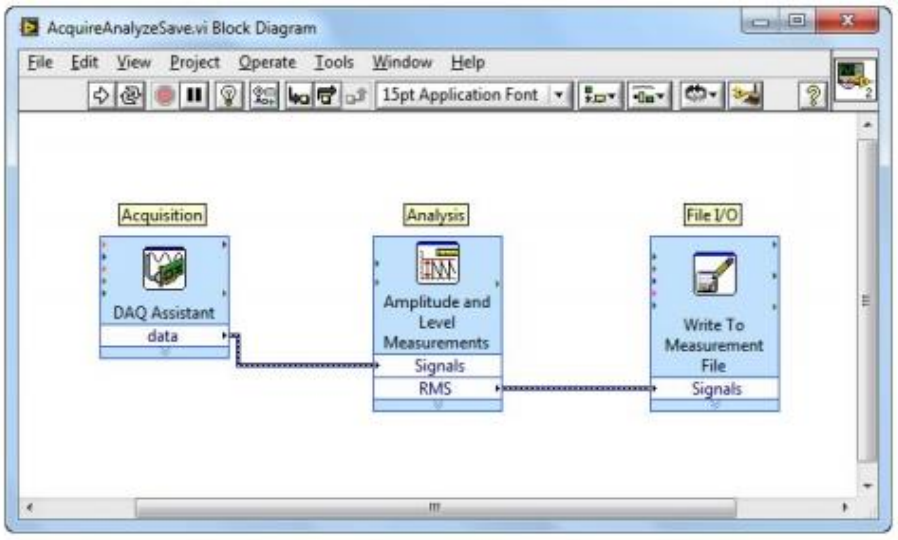


Fig. 3.5. Los datos se originan en la función de adquisición y luego fluyen a las funciones de análisis y almacenamiento a través de los cables.

- **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 pueden 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).

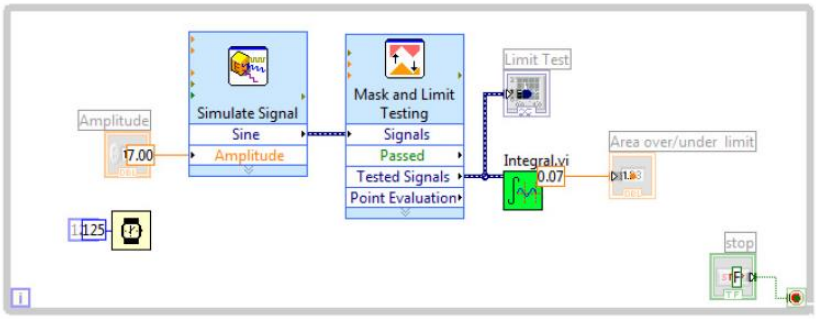


Fig. 3.6. La ejecución Highlight provee de una forma intuitiva de entender el orden de la ejecución del código G.

LabVIEW también ofrece herramientas de depuración para G comparables con aquellas que se encuentran en los lenguajes tradicionales. Estas herramienta, son accesibles desde la barra de herramientas de un diagrama, incluyen sondas, puntos de parada y ejecución paso a paso.

Fig. 3.7. La barra de herramientas del diagrama de bloques ofrece acceso a las herramientas de depuración estándar como la ejecución pasa 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).

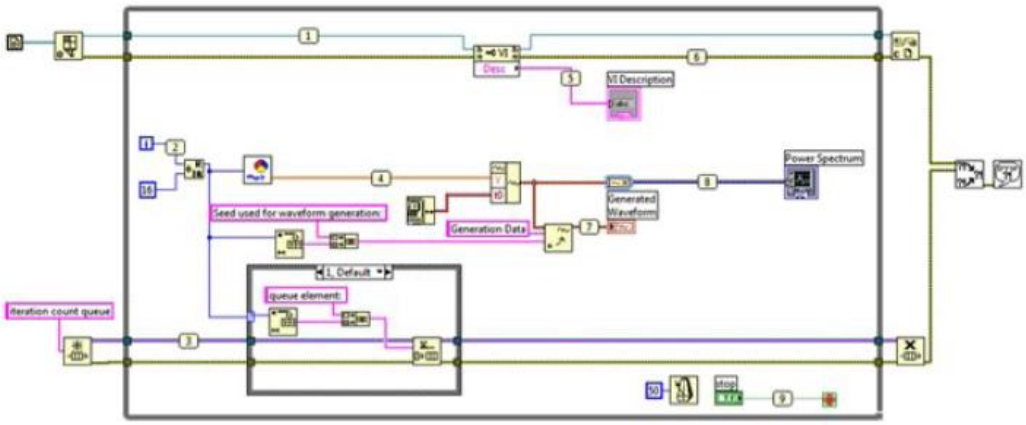


Fig. 3.8. Las sondas don formas eficientes de ver los valores moviéndose por los cables de la aplicación, incluso para secciones paralelas al código.

Una de las herramientas más comunes 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 ve un botón de ejecución roto en la barra de herramientas.

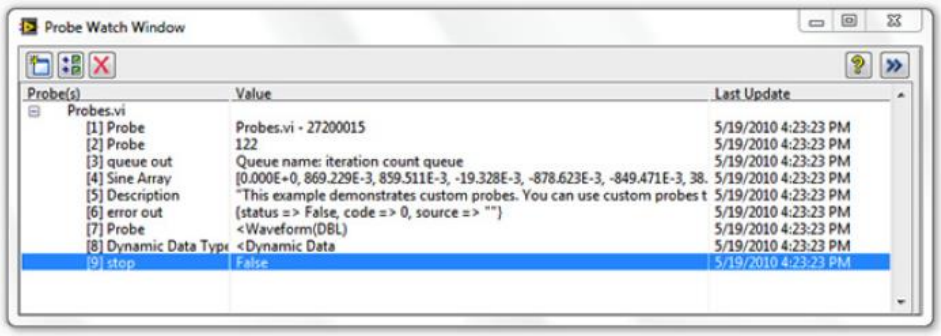


Fig. 3.9. Vista de los valores de las sondas en la ventana de visualización de sondas, que muestra los valores de las sondas de la aplicación completa (incluyendo subrutinas).

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 su programa en código máquina. Una vez compilado, el rendimiento de los programas de G es comparable al de aquellos basados en texto como C.

Fig. 3.10. 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 su programa en código de máquina. Una vez compilado, el rendimiento de los programas de G es comparable al de aquellos basados en texto como C.

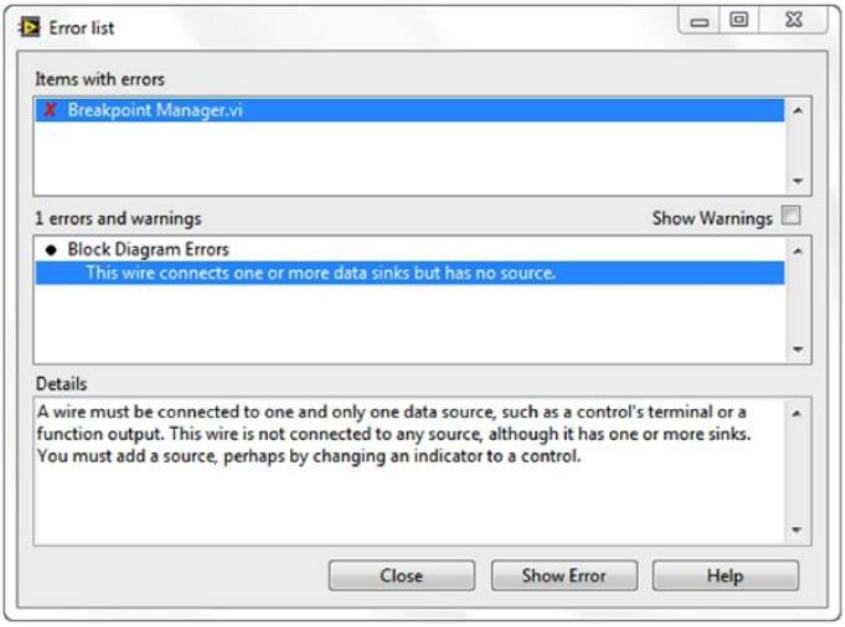


Fig. 3.11.La lista de errores muestra una explicación detallada de cada error sintáctico en la jerarquía completa del código.

- **Paralelismo y rendimientos 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 ciclos 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 ciclos usando colas, que son estructuras de datos estándar en los lenguajes de programación de propósito general.

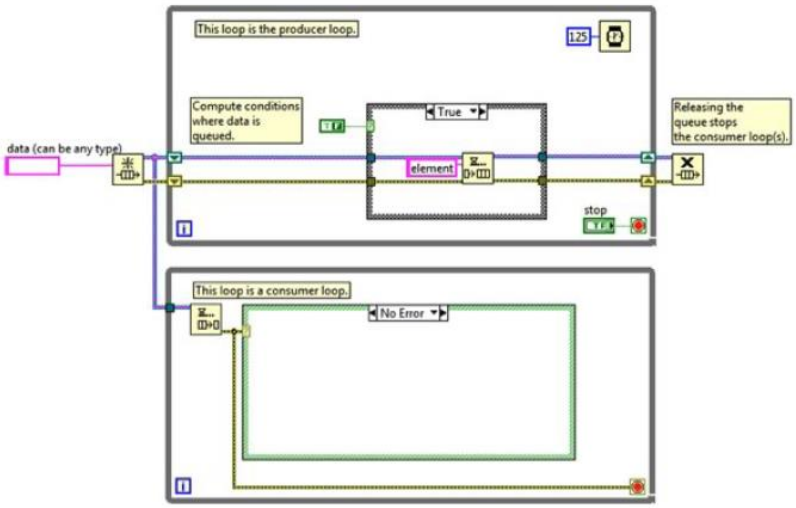
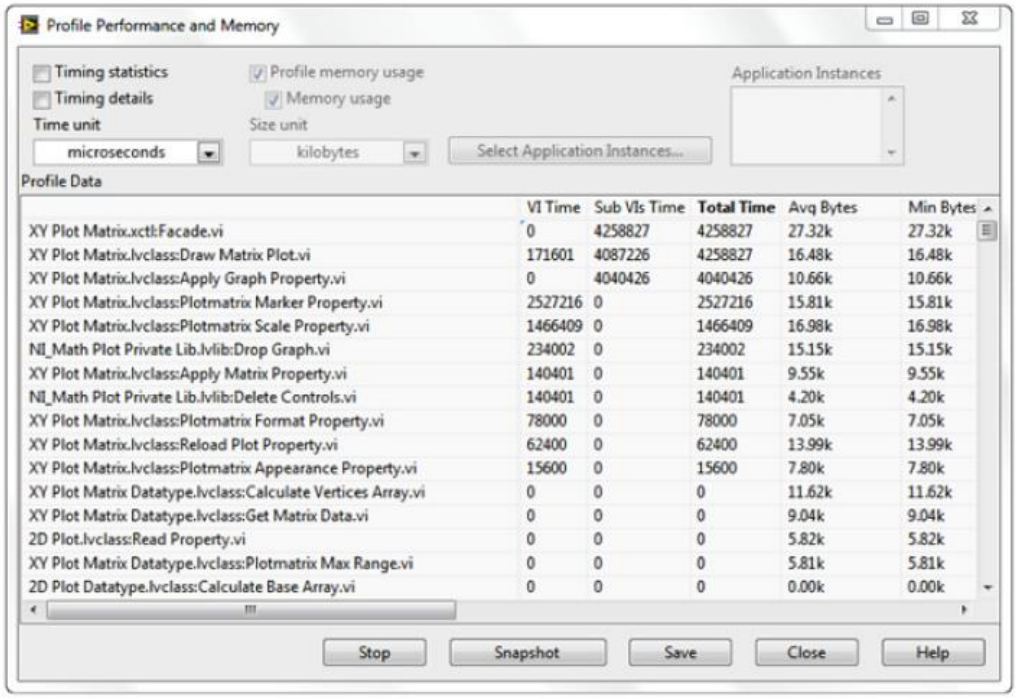


Fig. 3.12. El patrón de diseño Productor/Consumidor de LabVIEW es a menudo usado para aumentar el rendimiento de las aplicaciones que requieren tareas paralelas.

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. Actualmente, 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. 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 ex profeso 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 chip 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. Históricamente, la programación en FPGA pertenecía sólo al campo de los 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 y el flujo de datos y está creciendo rápidamente en popularidad como herramienta para desarrolladores que buscan procesamiento paralelo y ejecución determinista.



	VI Time	Sub VIs Time	Total Time	Avg Bytes	Min Bytes
XY Plot Matrix.xctl:Facade.vi	0	4258827	4258827	27.32k	27.32k
XY Plot Matrix.lvclass:Draw Matrix Plot.vi	171601	4087226	4258827	16.48k	16.48k
XY Plot Matrix.lvclass:Apply Graph Property.vi	0	4040426	4040426	10.66k	10.66k
XY Plot Matrix.lvclass:Plotmatrix Marker Property.vi	2527216	0	2527216	15.81k	15.81k
XY Plot Matrix.lvclass:Plotmatrix Scale Property.vi	1466409	0	1466409	16.98k	16.98k
NI_Math Plot Private Lib.lvlib:Drop Graph.vi	234002	0	234002	15.15k	15.15k
XY Plot Matrix.lvclass:Apply Matrix Property.vi	140401	0	140401	9.55k	9.55k
NI_Math Plot Private Lib.lvlib:Delete Controls.vi	140401	0	140401	4.20k	4.20k
XY Plot Matrix.lvclass:Plotmatrix Format Property.vi	78000	0	78000	7.05k	7.05k
XY Plot Matrix.lvclass:Reload Plot Property.vi	62400	0	62400	13.99k	13.99k
XY Plot Matrix.lvclass:Plotmatrix Appearance Property.vi	15600	0	15600	7.80k	7.80k
XY Plot Matrix Datatype.lvclass:Calculate Vertices Array.vi	0	0	0	11.62k	11.62k
XY Plot Matrix Datatype.lvclass:Get Matrix Data.vi	0	0	0	9.04k	9.04k
2D Plot.lvclass:Read Property.vi	0	0	0	5.82k	5.82k
XY Plot Matrix Datatype.lvclass:Plotmatrix Max Range.vi	0	0	0	5.81k	5.81k
2D Plot Datatype.lvclass:Calculate Base Array.vi	0	0	0	0.00k	0.00k

Fig. 3.13. La administración de memoria en LabVIEW es opcional, pero los usuarios avanzados pueden esbozar el uso de memoria para identificar las secciones de la aplicación a optimizar

Cuando el código G muestra un comportamiento inusual o no esperado que no puede resolver usando las herramientas de depuración citadas con anterioridad, se puede usar herramientas más avanzadas con el LabVIEW Desktop Execution Trace Toolkit. Este toolkit está diseñado para usuarios avanzados que quieren realizar análisis dinámico de código para lo siguiente: Detectar fugas de memoria, aislar la fuente de un evento específico o comportamiento

no deseado, monitorizar la aplicación por zonas para mejorar el rendimiento, identificar la última llamada antes de un error, y asegurar que la ejecución de una aplicación es la misma en dispositivos diferentes.

- **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 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. Trabajando con LabVIEW, se puede elegir un enfoque textual, gráfico o combinado. Por ejemplo, LabVIEW contiene el concepto de "Formula Node" que evalúa las fórmulas 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.

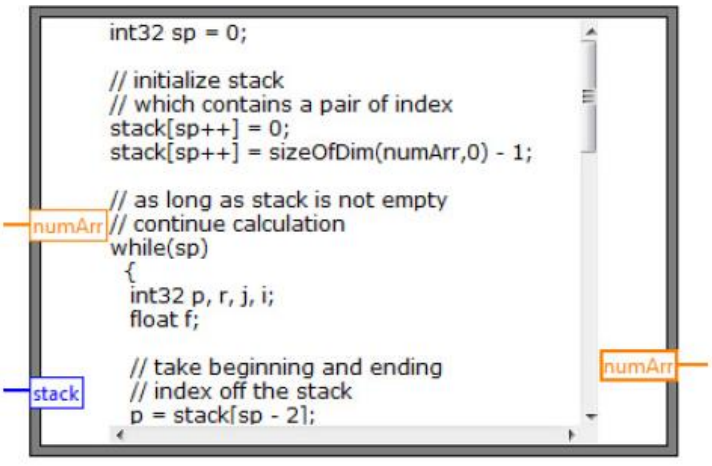


Fig. 3.14. Formula Node usa sintaxis similar a C para representar expresiones matemáticas de una forma sucinta en formato de texto.

Igualmente, el MathScript Node añade programación textual matemática en LabVIEW, generalmente compatible con la sintaxis de los archivos .m.

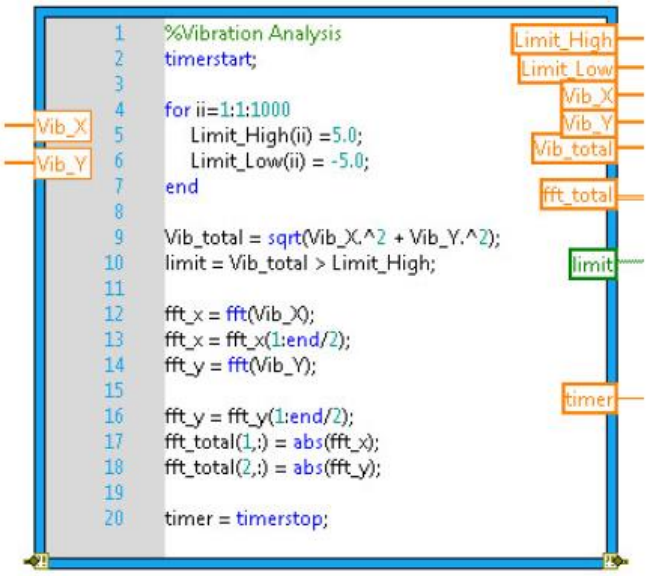


Fig. 3.15. Con MathScript Node, se puede crear o reutilizar archivos .m para el procesamiento de señales y el análisis de datos

- **Una manera mejor de solucionar problemas.**

LabVIEW y su lenguaje de programación gráfico de flujo de datos provee de una mejor manera de solucionar los problemas que las alternativas tradicionales de bajo nivel y la prueba están en su longevidad. Las claves diferenciadoras de la programación en G son el código gráfico intuitivo que se crea y las reglas de movimiento de los datos que gobiernan la ejecución que se combinan para ofrecer una experiencia de programación que expresa el pensamiento de los procesos de sus usuarios de forma más cercana que otros lenguajes. A pesar de que G es un lenguaje de alto nivel, se puede lograr rendimientos comparables a los de los lenguajes como C gracias al compilador de LabVIEW. [10]

3.2.Motores eléctricos

3.2.1. Introducción

El motor eléctrico es un dispositivo que transforma la energía eléctrica en energía mecánica por medio de la acción de los campos magnéticos generados en sus bobinas. Son máquinas eléctricas rotatorias compuestas por un estator y un rotor.

Los motores eléctricos se dividen en dos grandes grupos: motores de corriente continua y motores de corriente alterna. Los motores de corriente continua fueron los primeros en utilizarse en las aplicaciones en las que era un requisito el control de la velocidad de giro del motor. Los motores de corriente continua son constructivamente más complejos que los de alterna debido al uso de colectores de delgas, escobillas, etc. Esto hace que sean menos robustos y fiables que los de corriente alterna, además de más caros en cuanto a construcción y mantenimiento. El motor de corriente alterna asíncrono y trifásico representa la solución industrial más solicitada. Con la aparición de los inversores estáticos quedó solucionado el problema de control de velocidad de los motores de alterna.

Algunos de los motores eléctricos son reversibles, ya que pueden transformar energía mecánica en energía eléctrica funcionando como generadores.

Existen diferentes tipos de motores eléctricos y cada tipo tiene distintos componentes cuya estructura determina la interacción de los flujos eléctricos y magnéticos que originan la fuerza o par de torsión del motor.

El principio fundamental que describe cómo es que se origina una fuerza por la interacción de una carga eléctrica puntual q en campos eléctricos y magnéticos es la Ley de Lorentz

$$F = q(E + v \times B)$$

Donde:

- “q” es la carga eléctrica puntual
- “E” es el campo eléctrico

- "v" es la velocidad de la partícula
- "B" es la densidad de campo magnético

En el caso de un campo puramente eléctrico la expresión se reduce a:

$$F = q \cdot E$$

La fuerza en este caso está determinada solamente por la carga q y por el campo eléctrico E. Es la fuerza de coulomb que actúa a lo largo del conductor originando el flujo eléctrico.

En el caso de un campo puramente magnético:

$$F = q(v \times B)$$

La fuerza está determinada por la carga, la densidad del campo magnético B y la velocidad de la carga v. esta fuerza es perpendicular al campo magnético y a la dirección de la velocidad de la carga. Normalmente hay muchísimas cargas de movimiento por lo que conviene reescribir la expresión en términos de densidad de carga ρ y se obtiene entonces densidad de fuerza Fv (fuerza por unidad de volumen):

$$F = \rho(E + v \times B)$$

Al producto ρV se le conoce como densidad de corriente $\mathbf{J}$ (amperes por metro cuadrado):

$$\mathbf{J} = \rho v$$

Entonces la expresión resultante describe la fuerza producida por la interacción de la corriente con un campo magnético:

$$Fv = J \times B$$

Esto es un principio básico que explica cómo se originan las fuerzas en sistemas electromagnéticos como los motores eléctricos. Sin embargo, la completa descripción para cada tipo de motor eléctrico depende de sus componentes y de su construcción. [1]

En resumen y conceptualmente las dos ecuaciones que determinan el control de velocidad son las siguientes:

$$n = K_1 V$$

$$T = K_2 \frac{V}{f}$$

Siendo:

n la velocidad, V la tensión de alimentación, f la frecuencia y K_1 y K_2 constantes de proporcionalidad y T el par.

Variando la tensión se variará la velocidad, pero además para que el par permanezca constante y el par motor pueda vencer al par resistente será necesario variar la tensión y la frecuencia de la tensión en la misma proporción.

3.2.2. Tipos de motores eléctricos

Atendiendo al tipo de corriente que se utiliza para su alimentación podemos clasificar los motores eléctricos.

3.2.2.1. Motores de corriente continua

El motor de corriente continua (denominado también motor de corriente directa, motor CC o motor DC). Una máquina de corriente continua se compone principalmente de dos partes. El estator da soporte mecánico al aparato y contiene los devanados principales de la máquina, conocidos también con el nombre de polos, que pueden ser de imanes permanentes o devanados de cobre sobre un núcleo de hierro. El rotor en generalmente de forma cilíndrica, también devanado y con núcleo, alimentado con corriente directa mediante escobillas fijas. [2]

3.2.2.2. Motores de corriente alterna

3.2.2.2.1. Motores síncronos

Los motores síncronos son un tipo de motor de corriente alterna en el que la rotación del eje está sincronizada con la frecuencia de la corriente de alimentación; el periodo de rotación es exactamente igual a un número entero de ciclos de CA. Su velocidad de giro es constante y depende de la frecuencia de la tensión de la red eléctrica a la que esté conectado y por el número de pares de polos del motor, siendo conocida esa velocidad como “velocidad de sincronismo”.

Este tipo de motor contiene electromagnetos en el estator del motor que crean un campo magnético que rotan en el tiempo a esta velocidad de sincronismo. [8]

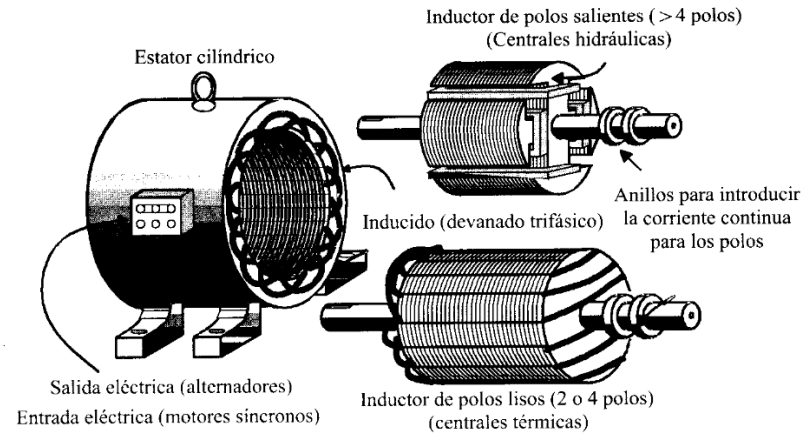


Fig. 3.16. Tipos de máquinas síncronas.

3.2.2.2.2. Motores asíncronos

Los motores asíncronos o de inducción son aquellos motores eléctricos en los que el rotor nunca llega a girar en la misma frecuencia con la que lo hace el campo magnético del estator. Cuanto mayor es el par motor mayor es esta diferencia de frecuencias.
Están constituidos por un devanado inductor, situado en el estator, por el cual se introduce una corriente alterna, este devanado puede ser trifásico o monofásico, en el caso de motores de más de 1 HP normalmente es trifásico.

El devanado inducido está ubicado en el rotor, este puede ser del tipo devanado (monofásico o trifásico, de acuerdo al estator) o jaula de ardilla. En este el campo giratorio del estator induce FEMS y al estar en cortocircuito (jaula de ardilla) o cerrado por medio de un reóstato de arranque (rotor devanado o con anillos) aparecen corrientes en el rotor que al reaccionar con el campo giratorio del estator producen el giro del rotor a una velocidad cercana y menor a la del campo giratorio del estator. [8]

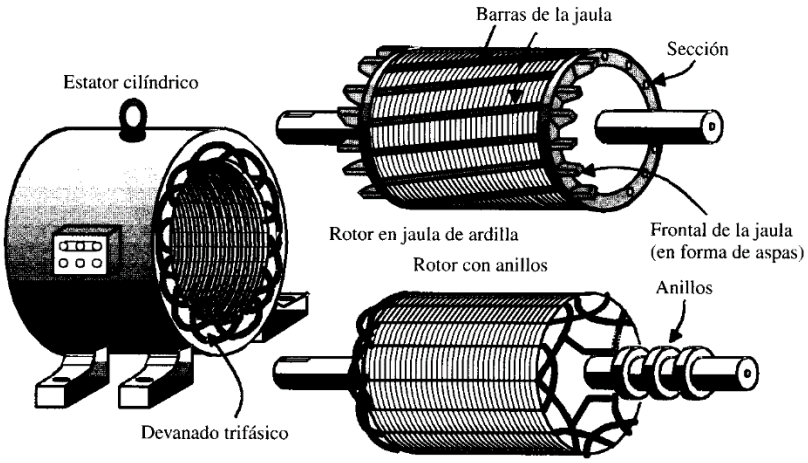


Fig. 3.17.Tipos de máquinas asíncronas.

3.2.3. Motores asíncronos

3.2.3.1. Introducción

Los motores asíncronos o de inducción son un tipo de motor de corriente alterna en el que la corriente eléctrica del rotor necesaria para producir torsión es inducida por inducción electromagnética del campo magnético de la bobina del estator. Por lo tanto, un motor de inducción no requiere una conmutación mecánica aparte de su misma excitación o para todo o parte de la energía transferida del estator al rotor, como en los universales, DC y motores grandes síncronos. El primer prototipo de motor eléctrico capaz de funcionar con corriente alterna fue desarrollado y construido por el ingeniero Nikola Tesla y presentado en el *American Institute of Electrical Engineers* (en español, *Instituto Americano de Ingenieros Eléctricos*, actualmente IEEE) en 1888.

El motor asíncrono trifásico está formado por un rotor, que puede ser de dos tipos: a) de jaula de ardilla; b) bobinado, y un estator, en el que se encuentran las bobinas inductoras. Estas bobinas son trifásicas y están desfasadas entre sí 120º en el espacio. Según el *Teorema de Ferraris*, cuando por estas bobinas circula un sistema de corrientes trifásicas equilibradas, cuyo desfase en el tiempo es también de 120º, se induce un campo magnético giratorio que envuelve al rotor. Este campo magnético variable va a inducir una tensión en el rotor según la Ley de

inducción de Faraday: la diferencia entre el motor a inducción y el motor universal es que en el motor a inducción el devanado del rotor no está conectado al circuito de excitación del motor sino que está eléctricamente aislado. Tiene barras de conducción en todo su largo, incrustadas en ranuras a distancias uniformes alrededor de la periferia. Las barras están conectadas con anillos (en cortocircuito como dicen los electricistas) a cada extremidad del rotor. Están soldadas a las extremidades de las barras. Este ensamblado se parece a las pequeñas jaulas rotativas para ejercitar a mascotas como hámsters y por eso a veces se llama "jaula de ardillas", y los motores de inducción se llaman motores de jaula de ardilla.

$$\mathcal{E} = -N\frac{d\Phi}{dt}$$

Entonces se da el *efecto Laplace* (o efecto motor): todo conductor por el que circula una corriente eléctrica, inmerso en un campo magnético experimenta una fuerza que lo tiende a poner en movimiento. Simultáneamente se da el *efecto Faraday* (o efecto generador): en todo conductor que se mueva en el seno de un campo magnético se induce una tensión.

El campo magnético giratorio, a velocidad de sincronismo, creado por el bobinado del estator, corta los conductores del rotor, por lo que se genera una fuerza electromotriz de inducción.

La acción mutua del campo giratorio y las corrientes existentes en los conductores del rotor, originan una fuerza electrodinámica sobre dichos conductores del rotor, las cuales hacen girar el rotor del motor.

La diferencia entre las velocidades del rotor y el campo magnético se denomina deslizamiento o resbalamiento.

La velocidad de rotación del campo magnético o *velocidad de sincronismo* está dada por:

$$n_{sinc} = \frac{60 f_e}{p}$$

Donde f_e es la frecuencia del sistema, en Hz, y p es el número de pares de polos en la máquina. Estando así la velocidad dada en revoluciones por minuto (rpm).

Lo que produce el voltaje inducido en la barra del rotor es el movimiento relativo del rotor en comparación con el campo magnético del estator, esto se puede observar en la siguiente ecuación:

$$\mathcal{E}_{ind} = (\vec{v} \times \vec{B}).\ell$$

Donde:

- $\vec{v}$: Velocidad de la barra *en relación con el campo magnético*
- $\vec{B}$: Vector de densidad de flujo magnético
- ℓ: Longitud del conductor en el campo magnético
- $\times$: Representa la operación "producto vectorial"

3.2.3.2. Topología

El motor de jaula de ardilla consta de un rotor constituido por una serie de conductores metálicos (normalmente de aluminio) dispuestos paralelamente unos a otros, y cortocircuitados en sus extremos por unos anillos metálicos, esto es lo que forma la llamada jaula de ardilla por su similitud gráfica con una jaula de ardilla. Esta 'jaula' se rellena de material, normalmente chapa apilada. De esta manera, se consigue un sistema n-fásico de conductores (siendo n el número de conductores, comúnmente 3) situado en el interior del campo magnético giratorio creado por el estator, con lo cual se tiene un sistema físico muy eficaz, simple, y muy robusto (básicamente, no requiere mantenimiento al carecer de escobillas).

El motor de rotor bobinado tiene un rotor constituido, en vez de por una jaula, por una serie de conductores bobinados sobre él en una serie de ranuras situadas sobre su superficie. De esta forma se tiene un bobinado en el interior del campo magnético del estator, del mismo número de polos (ha de ser construido con mucho cuidado), y en movimiento. Este rotor es mucho más complicado de fabricar y mantener que el de jaula de ardilla, pero permite el acceso al mismo desde el exterior a través de unos anillos que son los que cortocircuitan los bobinados.

Esto tiene ventajas, normalmente es como la posibilidad de utilizar un reóstato de arranque que permite modificar la velocidad y el par de arranque, así como el reducir la corriente de arranque.

En cualquiera de los dos casos, el campo magnético giratorio producido por las bobinas inductoras del estator genera unas corrientes inducidas en el rotor, que son las que producen el movimiento.

3.2.3.3. Principios de funcionamiento

El motor asincrónico funciona según el principio de inducción mutua de Faraday. Al aplicar corriente alterna trifásica a las bobinas inductoras, se produce un campo magnético giratorio, conocido como campo rotante, cuya frecuencia será igual a la de la corriente alterna con la que se alimenta al motor. Este campo al girar alrededor del rotor en estado de reposo, inducirá corrientes en el mismo, que producirán a su vez un campo magnético que seguirá el movimiento del campo estatórico, produciendo una cupla o par motor que hace que el rotor gire (principio de inducción mutua). No obstante, como la inducción en el rotor sólo se produce si hay una diferencia en las velocidades relativas del campo estatórico y el rotórico, la velocidad del rotor nunca alcanza a la del campo rotante. De lo contrario, si ambas velocidades fuesen iguales, no habría inducción y el rotor no produciría cupla. A esta diferencia de velocidad se la denomina "deslizamiento" y se mide en términos porcentuales, por lo que ésta es la razón por la cual a los motores de inducción se los denomina asincrónicos, ya que la velocidad rotórica difiere levemente de la del campo rotante. El deslizamiento difiere con la carga mecánica aplicada al rotor, siendo máximo con la máxima carga aplicada al mismo. Sin embargo, a pesar de esto, el motor varía poco su velocidad, pero el par motor o cupla aumenta (y con ello la intensidad de corriente consumida) por lo que se puede deducir que son motores de velocidad constante.

Eléctricamente hablando, se puede definir al motor asincrónico como un transformador eléctrico cuyos bobinados del estator representan el primario, y los devanados del rotor equivalen al secundario de un transformador en cortocircuito.

En el momento del arranque, producto del estado de reposo del rotor, la velocidad relativa entre campo estatórico y rotórico es muy elevada. Por lo tanto, la corriente inducida en el rotor es muy alta y el flujo de rotor (que se opone siempre al del estator) es máximo. Como consecuencia, la impedancia del estator es muy baja y la corriente absorbida de la red es muy alta, pudiendo llegar a valores de hasta 7 veces la intensidad nominal. Este valor no hace ningún daño al motor ya que es transitorio, y el fuerte par de arranque hace que el rotor gire enseguida, pero causa bajones de tensión abruptos y momentáneos que se manifiestan sobre todo como parpadeo en las lámparas lo cual es molesto, y puede producir daños en equipos electrónicos sensibles. Los motores de inducción están todos preparados para soportar esta corriente de arranque, pero repetidos y muy frecuentes arranques sin períodos de descanso pueden elevar progresivamente la temperatura del estator y comprometer la vida útil de los devanados del mismo hasta originar fallas por derretimiento del aislamiento. Por eso se utilizan en potencias medianas y grandes, dispositivos electrónicos de "arranque suave", que minimizan la corriente de arranque del motor.

Al ganar velocidad el rotor, la corriente del mismo disminuye, el flujo rotórico también, y con ello la impedancia de los devanados del estator, recordemos que es un fenómeno de inducción mutua. La situación es la misma que la de conectar un transformador con el secundario en corto a la red de CA y luego con una resistencia variable intercalada ir aumentando progresivamente la resistencia de carga hasta llegar a la intensidad nominal del secundario. Por ende, lo que sucede en el circuito estatórico es un reflejo de lo que sucede en el circuito rotórico.

3.2.3.4. Regulación de la velocidad

De las fórmulas de la velocidad de sincronismo y del deslizamiento se deduce que:

$$n = n_1(1 - s) = \frac{60 \cdot f_1}{p}(1 - s)$$

Esto indica que se puede regular la velocidad de un motor asíncrono modificando su número de polos, la frecuencia del estator o el deslizamiento.

En este trabajo se controlará la velocidad mediante la variación de la frecuencia.

La regulación por variación de la frecuencia consiste en variar la frecuencia f_1 de las corrientes del estator con lo que se modifica la velocidad de sincronismo n_1 de la máquina. Para ello se alimenta el estator a través de un variador de frecuencias.

Este sistema permite variar la velocidad de forma continua entre un amplio margen de velocidades. Para frecuencias f_1 por debajo de la asignada interesa variar la tensión V_1 del estator en función de la frecuencia de forma que el flujo por polo Φ_M sea el mismo para todas las frecuencias. De esta manera se consigue que para las frecuencias el par que suministra la máquina a la corriente asignada sea el mismo (el par asignado) y que también a todas las frecuencias el par máximo sea el mismo. Para frecuencias f_1 por encima de la asignada no se puede mantener el flujo por polo Φ_M constante porque entonces la f.e.m. E_1 sería mayor que condiciones asignadas lo que conllevaría que la tensión en el estator fuera superior a la asignada. Por lo tanto, para frecuencias por encima de la asignada se mantiene el valor eficaz de las tensiones del estator igual al asignado. [8]

3.3.Drivers de potencia

3.3.1. Topología del inversor

Los convertidores CC-CA son conocidos por inversores. Ellos pueden tener salida variable en voltaje y frecuencia.

La topología del inversor se define por el modo de conexión de los dispositivos de conmutación que conforman el puente. Cada rama del puente consta de dos dispositivos de conmutación. Una o dos ramas –puente H- son empleados para puentes monofásicos o bifásicos. Tres ramas son empleadas para puentes trifásico siendo este el de mayor uso, aunque pueden emplearse mayor número de ramas para aumentar la capacidad de conducción. [7]

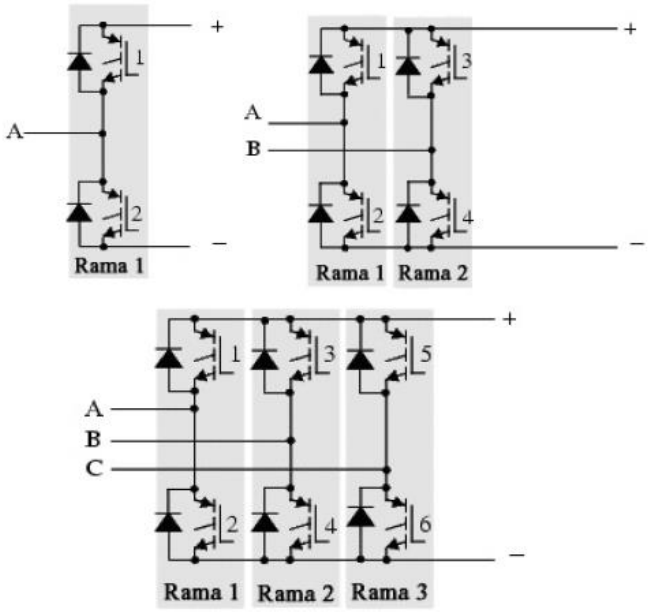


Fig. 3.18. Topología de Inversores de una, dos y tres ramas.

La configuración más simple es el inversor monofásico, de medio puente, el cual requiere una fuente CC de tres conductores, sin embargo los inversores trifásicos se utilizan normalmente en aplicaciones de alta potencia. Tres inversores monofásicos de medio puente (o puente completo) pueden conectarse para formar un inversor trifásico, siempre que sus señales de puerta estén desfasadas 120°. Las configuraciones de seis dispositivos de conmutación, son los más

empleados, pudiéndose aplicar señales de control: conducción a 120° o conducción a 180°, es decir, permitiendo la conducción por dos o tres dispositivos de conmutación respectivamente.

Algunas aplicaciones de inversores requieren de medios de control sobre el voltaje de salida. En la mayoría de esas aplicaciones el control es usualmente requerido a fin de proveer un ajuste continuo (*stepless*) del voltaje de salida. Los métodos de control pueden ser agrupados en tres grandes categorías:

- Control de voltaje suministrado al inversor.
- Control del voltaje entregado por el inversor.
- Control del voltaje dentro del inversor.

Hay un número de bien conocidos métodos de controlar el voltaje CC suministrado al inversor o del voltaje CA entregado por el inversor. Esto incluye el uso de Chopper CC, amplificadores magnéticos, reguladores de inducción, rectificadores de fase controlado y transistores series o reguladores *shunt*. La principal desventaja de estos métodos es que la potencia entregada por el inversor es manejada dos veces, una vez por el control de voltaje CC o CA y otra por el inversor. Esto generalmente involucra más equipos que los que serían necesario que si el control de voltaje es hecho dentro del inversor. El control de la salida del inversor puede ser logrado por la incorporación de controles de relación de tiempo dentro del circuito del inversor.

Un método de controlar el voltaje dentro del inversor involucra el uso de las técnicas de modulación de ancho de pulso (PWM). Con esta técnica el voltaje de salida del inversor es controlado por la variación de la duración de los pulsos de voltaje de salida.
Es importante considerar que los interruptores precisan de un tiempo mínimo, tanto en la apertura para anular la corriente, como para el cierre para su establecimiento. Por tanto se debe prestar atención al instante de cierre de un interruptor durante un tiempo de bloque necesario del interruptor complementario a la misma rama. Esta corriente de descarga circulará por los diodos dispuestos en paralelo con cada interruptor. Una vez que la corriente sea nula, se permitirá el cierre del interruptor complementario. Este tiempo de espera se denomina generalmente tiempo muerto y debe de ser respetado y tenido en cuenta durante el diseño. [18]

Es por eso que las señales de control de los dos interruptores de un mismo brazo deben ser complementarias para el inversor de conducción a 180º, a fin de no cortocircuitar la fuente de continua de alimentación, o para el caso de conducción a 120º, que las señales de control nunca estén activas al mismo tiempo. También es importante usar protecciones en los apagados y encendidos de los interruptores, de tal forma que se pueda absorber la energía procedente de los elementos reactivos del circuito durante el proceso de conmutación controlando parámetros tales como la evolución de la tensión o corriente en el interruptor, o bien limitando los valores máximos de tensión que ha de soportar.

3.3.2. Inversor trifásico

El objetivo de un inversor trifásico es generar energía eléctrica de corriente alterna a partir de una fuente de energía de corriente continua, con magnitudes y frecuencias deseadas.
Se constituye principalmente por dispositivos electrónicos de potencia, que trabajan como interruptores operando en corte y saturación con una secuencia apropiada para obtener tres tensiones de salida simétricas y balanceadas. El controlador es otro componente fundamental en la constitución del convertidor, es el que genera las señales de encendido y apagado de los dispositivos semiconductores y garantiza su buen comportamiento. Cualquier tipo de inversor (monofásico o trifásico) utilizan dispositivos con activación y desactivación controlada (es decir BJT[15], MOSFET[16], IGBT[17], MCT[18], SIT[19], GTO[20]) o tiristores de conmutación forzada, según la aplicación.

[15] BJT: Bipolar Junction Transistor

[16] MOSFET: Metal-oxide-semiconductor Field-effect transistor

[17] IGBT: Insulated Gate Bipolar Transistor

[18] MCT: MOS Controlled Thyristor

[19] SIT:Static Induction Transistor

[20] GTO: Gate Turn-Off Thyristor

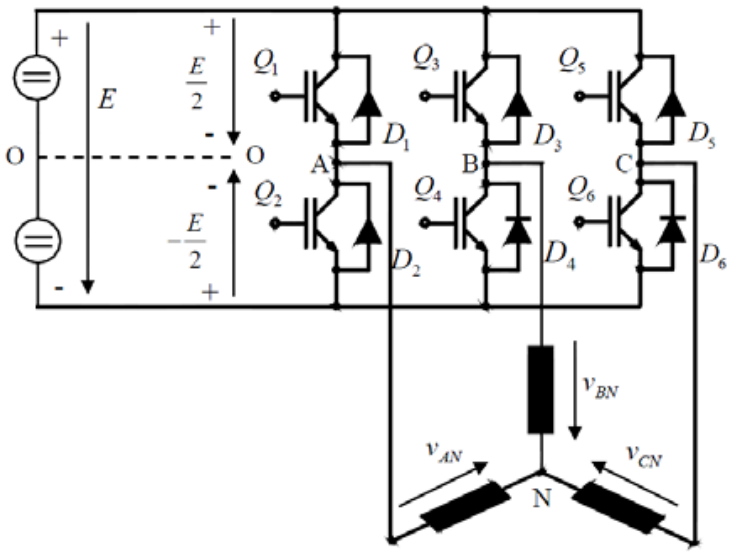


Fig. 3.19. Inversor trifásico con dos niveles de tensión.

La Fig. 3.19 muestra el caso de un puente con tres ramas, o inversor trifásico en puente completo, el cual se compone de seis dispositivos de conmutación (6 transistores IGBTs) designados Q1 a Q6 y seis diodos de libre circulación (D1 a D6) dispuesto en conexión inversa con los interruptores, empleados para conducir la corriente reactiva de retorno a la fuente de tensión E. Estos diodos aseguran por un lado la continuidad de la corriente en la carga inductiva y por otro lado la reversibilidad de la potencia al poder inyectar corriente desde la carga a la batería de continua. Cada brazo del inversor está formado por dos interruptores o dispositivos de conmutación en paralelo con sus diodos de libre circulación, estando la salida a cada fase del motor situada en el punto medio del brazo. Estos inversores se dividen según su forma de operar en: conducción a 180º de cada elemento, con lo cual habrá 3 elementos en conducción al mismo tiempo y conducción a 120º, con 2 elementos por vez. Además pueden alimentar los dos tipos característicos de cargas trifásicas simétricas: conexión delta y estrella [16].

3.3.2.1. Conducción a 180º

Cada transistor conducirá durante 180º. Tres transistores se mantienen activos durante cada instante del tiempo. Cuando el transistor Q1 está activado, la fase "a" se conecta con la terminal positiva del voltaje de entrada. Cuando se activa el transistor Q4, la fase "a" se lleva a la terminal negativa de la fuente DC. En cada ciclo existen seis modos de operación, cuya duración es de 60 grados. Los transistores se numeran según su secuencia de excitación por ejemplo (612,

123, 234, 345, 456, 561, 612). Las señales de excitación mostradas en la Fig. 3.20 están desplazadas 60 grados unas de otras, para obtener voltajes trifásicos balanceados. [9]

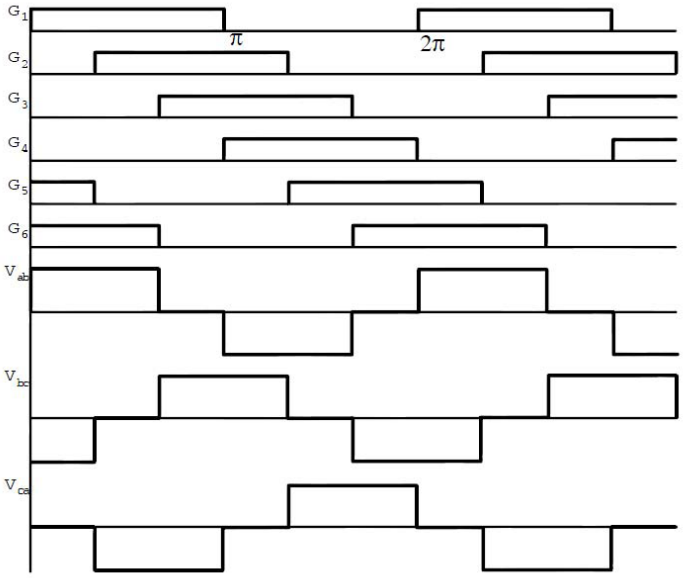


Fig. 3.20. Secuencia de la señales de excitación de los transistores a 180º.

La limitación de la simultaneidad de la conducción de los dos interruptores de un mismo brazo, implica que solamente existen ocho configuraciones posibles de salida del inversor.

Las tensiones se obtienen en función del estado de los diferentes interruptores, como se muestra en la siguiente figura, en la que se puede apreciar que en el primer estado y en el último las tensiones de alimentación del motor son nulas, por lo que a veces estos estados se denominan "estados de libre circulación".

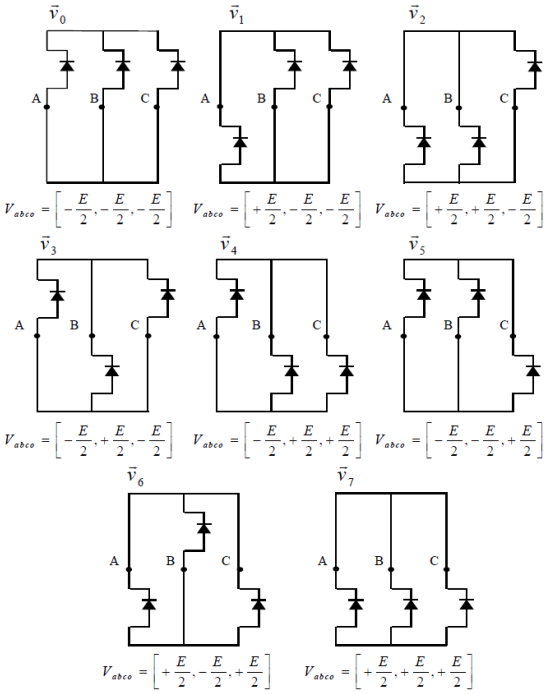


Fig. 3.21. Distintas configuraciones del inversor en función del estado de los interruptores, para una conducción de 180º.

.

3.3.2.2. Conducción a 120º

En este tipo de control, cada transistor conduce durante 120º. En cualquier instante del tiempo, solo conducen dos transistores, dándose la posibilidad de que los dos transistores de una misma rama estén abiertos. Las señales de excitación se muestran en la Fig. 3.22. La secuencia de conducción de los transistores es 61, 12, 23, 34, 45, 56, 61,… [7].

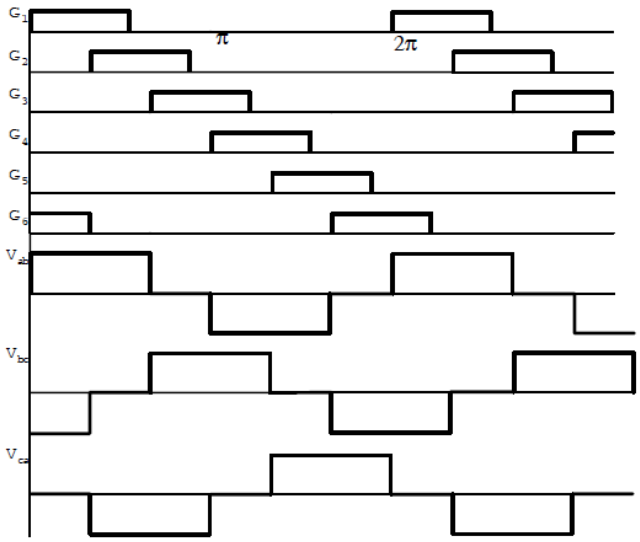


Fig. 3.22. Secuencia de la señales de excitación de los transistores a 120º.

La limitación de la simultaneidad de la conducción de los dos interruptores de un mismo brazo es una propiedad que se repite de nuevo en el inversor trifásico con conducción a 120º, pero la diferencia entre estos dos tipos de inversores radica en la posibilidad de tener dos interruptores de una misma rama en abierto. Esta nueva característica implica que existen más de ocho combinaciones posibles de salida del inversor. Las tensiones calculadas corresponden a los valores respecto a la referencia "o", situada en el punto medio ficticio de la alimentación continua del inversor (Vao, Vbo, Vco). También se muestra las tensiones entre dos fases de salida (Vab, Vbc, Vca).

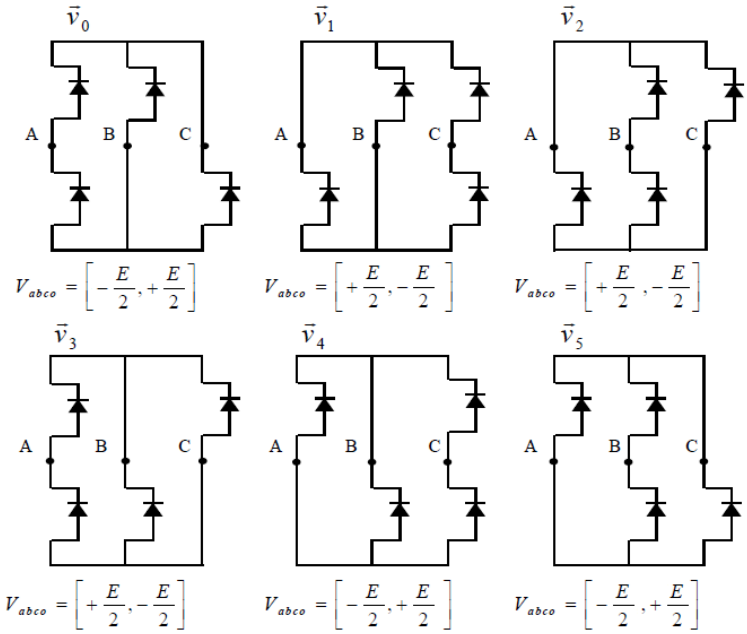


Fig. 3.23. Distintas configuraciones del inversor en función del estado de los interruptores, para una conducción de 120º.

Para la aplicación de inversores trifásicos en motores debe utilizarse la misma señal PWM para controlar los seis interruptores, aunque esta señal es producida y generada por el propio programa dentro del controlador. Esta señal PWM varía la tensión de salida y su frecuencia, pero para activar los interruptores en la secuencia correcta se diseña una señal a partir de una puerta lógica AND de dos entradas, que se forma a partir de la señal PWM. Por tanto, se obtiene seis señales distintas que indican qué interruptores deben estar abiertos y cuales cerrados.

3.3.3. Tecnología PWM- Modulación por ancho de pulso

La tecnología PWM es una tecnología robusta. Esta técnica de control se basa en emplear una modulación múltiple (varios pulsos de disparo en cada medio ciclo de voltaje de salida), en la que el ancho de cada pulso varía en proporción con la amplitud de una onda senoidal evaluada en el centro del mismo pulso. [17]

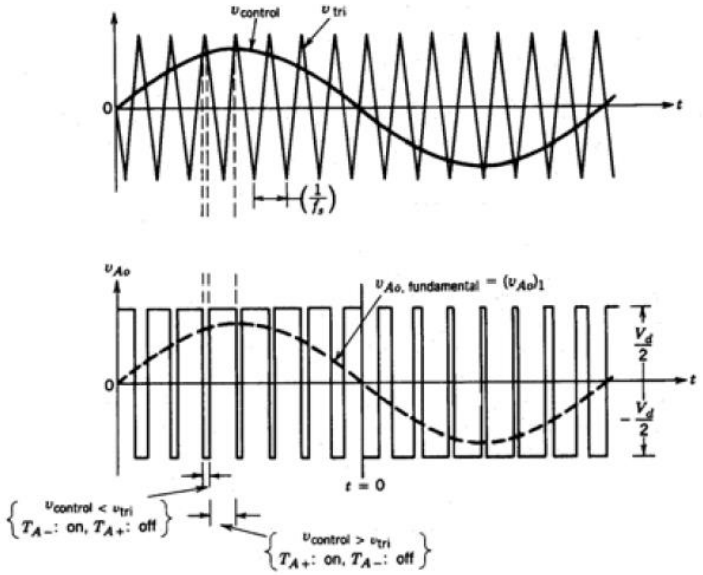


Fig. 3.24. Esquema de modulación de ancho de pulso.

La modulación se logra empleando una señal de control senoidal "Vcontrol" a la frecuencia de salida deseada "f1", que es comparada con una onda portadora triangular para generar las señales de disparo. La frecuencia de la forma de onda triangular establece la frecuencia de conmutación del inversor "fs", y esta se mantiene constante. La relación de modulación de amplitud es:

$$m_a = V_{control}/V_{triangular}$$

Donde:

- "$V_{control}$" es la amplitud pico de la señal de control.
- "$V_{triangular}$" es la amplitud pico de la onda portadora triangular.
- "m_a" es la relación de modulación de amplitud.

La relación de modulación de frecuencia es:

$$m_f = f_s/f_1$$

Donde:

- "m_f" es la relación de modulación en frecuencia.
- "f_s" es la frecuencia de la onda portadora triangular.

- "f_1" es la frecuencia de la señal de control.

En el caso de un puente completo de dos ramas se cumple: cuando Vcontrol > Vtrip, TA+ y TB- se mantienen operando y V0 = Vd. Cuando Vcontrol < Vtrip, TA- y TB+ se mantienen operando y V0 = -Vd. El voltaje de salida fluctúa entre +Vd y –Vd.

El voltaje de salida del inversor contiene armónicos, cuyo espectro armónico se muestra en la Fig. 3.25.

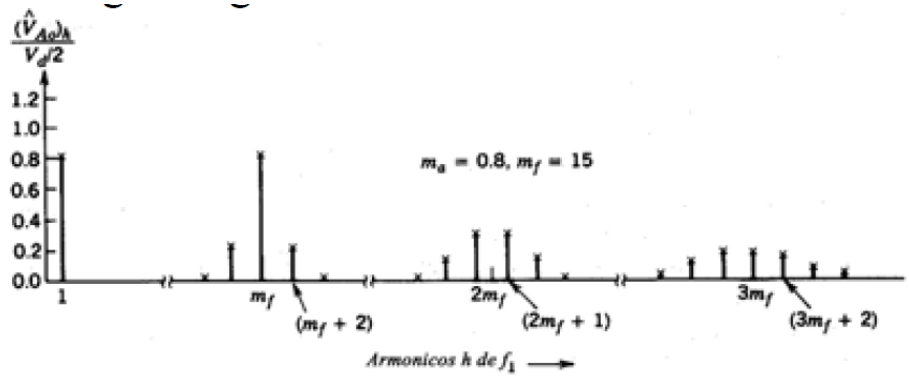


Fig. 3.25. Contenido de armónicos del esquema PWM.

Para la relación de modulación de amplitud menor a uno (m_a<1), se opera en el rango lineal del inversor, y el voltaje pico de la componente de frecuencia fundamental de voltaje de salida preserva una relación lineal entre el índice de modulación de amplitud y la tensión Vd.

La relación lineal mientras m_a sea menor que la unidad.

$$V_{A0} = m_a \cdot V_d$$

Donde:

- "V_{A0}" es la amplitud de pico de la señal de salida.
- "m_a" es la relación de modulación de amplitud.
- "V_d" es la tensión de alimentación del circuito inversor.

El PWM empuja los armónicos de la onda de voltaje de salida al rango de las altas frecuencias, alrededor de la frecuencia de conmutación fs y sus múltiplos como mf, 2mf, 3mf y más. Mf debe ser un número entero impar tal que la forma de onda de voltaje de salida solo contenga armónicas impares. De esta forma, es más fácil de eliminar los armónicas filtrando las

altas frecuencias. De ahí que sea deseable usar una frecuencia de conmutación lo más alta posible, aunque esto tiene la desventaja que las pérdidas de conmutación aumentan proporcionalmente. El valor seleccionado que delimita entre grande y bajo el valor de mf es 21.

Para pequeños valores del índice de modulación de frecuencia (mf < 21) la forma de onda de la señal triangular y la señal de control deben estar sincronizadas. Esto es denominado PWM sincronizado, donde mf no es un entero, y se producen subarmónicos de la frecuencia fundamental, los cuales no son deseables. Cuando mf se hace grande, los subarmónicos debido al PWM asincrónico son pequeños. De ahí que PWM asincrónico se emplee más.

Para PWM's con m_a <1 la amplitud del voltaje fundamental varía linealmente con ma. Pero en contraposición, la magnitud de la componente de frecuencia fundamental es menor. Cuando m_a tiene un valor mayor que uno, la amplitud también se incrementa lo cual produce una sobremodulación. Por tanto, la forma de onda de salida contiene muchos más armónicos en el lado de la banda comparado con el rango lineal. En la Fig. 3.26 se muestra la amplitud normalizada del pico de la componente de frecuencia fundamental V_{A0}/V_d como función de la relación del índice de modulación de amplitud.

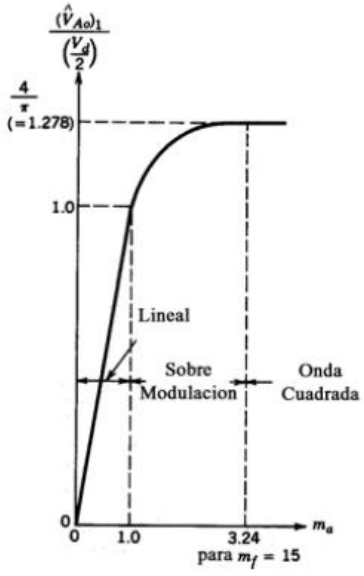


Fig. 3.26. Zonas de operación del esquema PWM.

Los armónicos que son dominantes en el rango lineal pueden no ser dominantes durante la sobremodulación. La amplitud de la componente fundamentalmente no varía linealmente con ma. En aplicaciones de potencia, la región de sobremodulación debe ser evitada para minimizar

la distorsión en el voltaje de salida. Cuando m_a es suficientemente grande (m_a>>1) la forma de onda del voltaje de salida se degenera desde una PWM a una forma de onda cuadrada. Y en este caso, la magnitud del voltaje fundamental máximo de salida dependerá de la tensión de alimentación Vd,

$$V_{A0_{max}} = 4 \cdot V_d/\pi = 1.278 \cdot V_d$$

Donde:

- "$V_{A0_{max}}$" es la tensión fundamental de salida máxima.
- "V_d" es la tensión de alimentación del circuito de potencia.

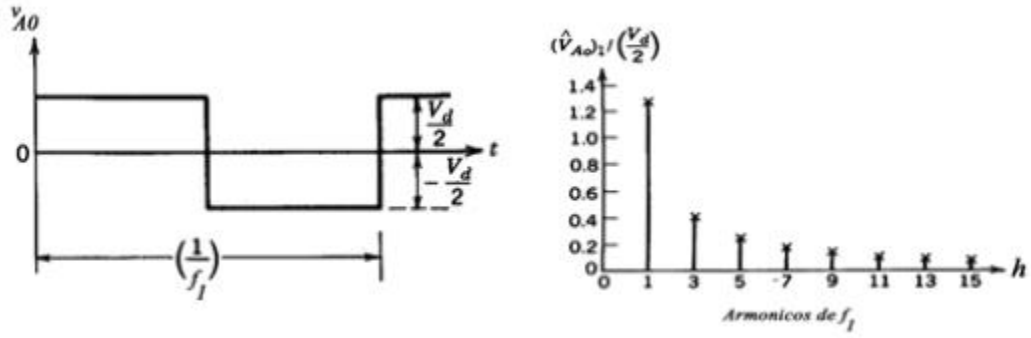


Fig. 3.27. Contenido de armónicos del esquema PWM, para operación en onda cuadrada.

Los inversores trifásicos ofrecen una significativa ventaja. Si mf es elegido tal que cumple que es impar y múltiplo de tres (por ejemplo 3, 9, 15, 21, 29…) se logra que la forma de onda de voltaje de salida sea más senoidal, incluso los armónicos estarán ausentes en el voltaje de fase. Además, es deseable empujar mf al valor más alto posible. Cuando mf es alto, los armónicos estarán a muy altas frecuencias, de tal forma que la onda de salida apenas se verá afectada por estos armónicos, obteniendo una mejora en las pérdidas de conmutación.

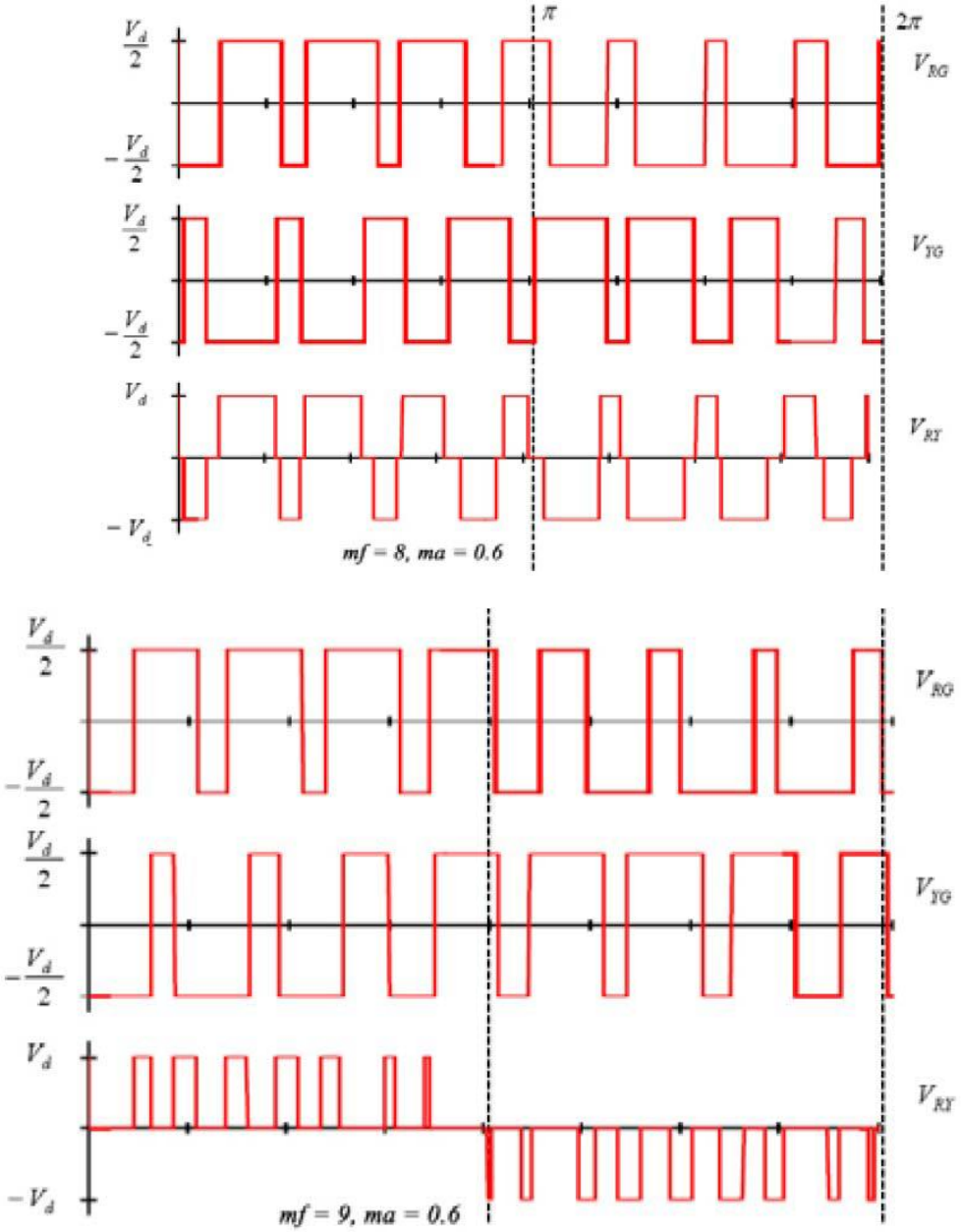


Fig. 3.28. Forma de onda de voltaje en Inversor trifásico con esquema PWM.

3.3.3.1. Aplicación

En la actualidad existen muchos circuitos integrados en los que se implementa la modulación PWM, además de otros muy particulares para lograr circuitos funcionales que puedan controlar fuentes conmutadas, controles de motores, controles de elementos termoeléctricos, choppers para sensores en ambientes ruidosos y algunas otras aplicaciones.

3.3.3.1.1. En los motores

La modulación por ancho de pulsos es una técnica utilizada para regular la velocidad de giro de los motores eléctricos de inducción o asíncronos. Mantiene el par motor constante y no supone un desaprovechamiento de la energía eléctrica. Se utiliza tanto en corriente continua como en alterna, como su nombre lo indica, al controlar: un momento alto (encendido o alimentado) y un momento bajo (apagado o desconectado), controlado normalmente por relés (baja frecuencia) o MOSFET o tiristores (alta frecuencia).

Otros sistemas para regular la velocidad modifican la tensión eléctrica, con lo que disminuye el par motor; o interponen una resistencia eléctrica, con lo que se pierde energía en forma de calor en esta resistencia.

Otra forma de regular el giro del motor es variando el tiempo entre pulsos de duración constante, lo que se denomina modulación por frecuencia de pulsos.

En los motores de corriente alterna también se puede utilizar la variación de frecuencia.

La modulación por ancho de pulsos también se usa para controlar servomotores, los cuales modifican su posición de acuerdo al ancho del pulso enviado cada un cierto período que depende de cada servo motor.

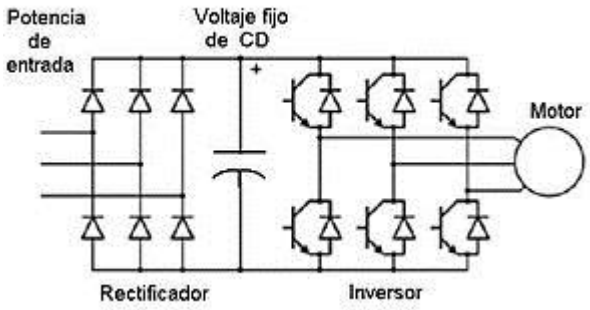


Fig. 3.29. Diagrama de ejemplo de la utilización de la modulación de ancho de pulsos en un variador de frecuencia.

3.4. Implementación en software

3.4.1. Generador de señales

Para generar un generador de señales trifásicas partimos de 3 señales desfasadas 120°entre ellas.

$$Va = V \cdot \cos(\omega t)$$

$$Vb = V \cdot \cos\left(\omega t - \frac{2\pi}{3}\right)$$

$$Vc = V \cdot \cos\left(\omega t + \frac{2\pi}{3}\right)$$

Para simplificar el análisis de un circuito con 3 fases se emplea la transformada de Clarke o también conocida como transformada alpha-beta ($\alpha\beta\gamma$) que simplifica este en un sistema de 2 fases.

$$\overrightarrow{V_{\alpha\beta}}(t) = K \cdot \left(Va(t) \cdot 1 + Vb(t) \cdot e^{j\frac{2\pi}{3}} + Vc(t) \cdot e^{-j\left(\frac{2\pi}{3}\right)}\right)$$

Hacemos el siguiente cambio de variable:

$$\alpha = e^{j\frac{2\pi}{3}}$$

$$\alpha^2 = e^{-j\frac{2\pi}{3}}$$

Quedándonos:

$$\overrightarrow{V_{\alpha\beta}}(t) = K \cdot (Va(t) \cdot 1 + Vb(t) \cdot \alpha + Vc(t) \cdot \alpha^2)$$

$$\overrightarrow{V_{\alpha\beta}}(t) = V_\alpha + j \cdot V_\beta$$

$$V_\alpha = K \cdot \left(Va(t) + Vb(t) \cdot \cos\left(\frac{2\pi}{3}\right) + Vc(t) \cdot \cos\left(\frac{2\pi}{3}\right)\right)$$

$$V_\beta = K \cdot \left(Va(t) \cdot 0 + Vb(t) \cdot \sin\left(\frac{2\pi}{3}\right) - Vc(t) \cdot \sin\left(\frac{2\pi}{3}\right)\right)$$

A la hora de construir la matriz se utiliza un término nulo $V_0 = \left[\frac{1}{\sqrt{2}} \quad \frac{1}{\sqrt{2}} \quad \frac{1}{\sqrt{2}}\right]$ para que la matriz sea cuadrática.

$$\begin{bmatrix} V_\alpha \\ V_\beta \\ V_0 \end{bmatrix} = K \cdot \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} Va(t) \\ Vb(t) \\ Vc(t) \end{bmatrix}$$

Los valores de K más normales son 1, $\frac{2}{3}$ y $\sqrt{\frac{2}{3}}$.

Aplicamos la transformada inversa obteniendo la inversa de Clarke.

$$\begin{bmatrix} Va(t) \\ Vb(t) \\ Vc(t) \end{bmatrix} = Q \cdot \begin{bmatrix} 1 & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} V_\alpha \\ V_\beta \\ V_0 \end{bmatrix}$$

Haciendo el producto entre la transformada de Clarke y la transformada inversa de Clarke obtenemos la matriz identidad.

$$KQ \cdot \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \end{bmatrix} = KQ \cdot \begin{bmatrix} \frac{3}{2} & 0 & 0 \\ 0 & \frac{3}{2} & 0 \\ 0 & 0 & \frac{3}{2} \end{bmatrix}$$

De donde podemos despejar Q de la siguiente manera:

$$K \cdot Q = \frac{2}{3} \rightarrow K = \sqrt{\frac{2}{3}}$$

$$\sqrt{\frac{2}{3}} \cdot Q = \frac{2}{3} \rightarrow Q = \sqrt{\frac{2}{3}}$$

Este es el resultado más normal que se da.

De donde representamos las funciones trigonométricas en funciones exponenciales:

$$\begin{cases} e^{j\theta} = \cos\theta + j\sin\theta \\ e^{-j\theta} = \cos\theta - j\sin\theta \end{cases} \rightarrow \frac{e^{j\theta} + e^{-j\theta}}{2} = \cos\theta$$

$$V_a(t) = \hat{V} \cdot \frac{e^{j\omega t} + e^{-j\omega t}}{2}$$

$$V_b(t) = \hat{V} \cdot \frac{e^{j\left(\omega t - \frac{2\pi}{3}\right)} + e^{-j\left(\omega t - \frac{2\pi}{3}\right)}}{2}$$

$$V_b(t) = \hat{V} \cdot \frac{e^{j\left(\omega t + \frac{2\pi}{3}\right)} + e^{-j\left(\omega t + \frac{2\pi}{3}\right)}}{2}$$

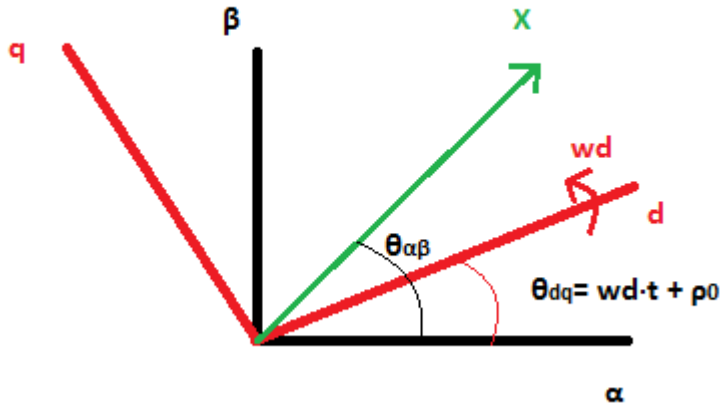
Una vez hecho esto aplicaremos la transformada de Clark:

$$\overrightarrow{V_{\alpha\beta}} = (V_a(t) \cdot 1 + V_b(t) \cdot \alpha + V_c(t) \cdot \alpha^2)$$

$$\overrightarrow{V_{\alpha\beta}} = K \cdot \frac{\hat{V}}{2} \cdot [e^{j\omega}(1 + 1 + +1) \quad e^{-j\omega}(1 + \alpha^2 + \alpha)]$$

Donde:

$$\overrightarrow{V_{\alpha\beta}} = K \cdot \frac{3}{2} \cdot \hat{V} \cdot e^{j\omega t} \rightarrow \begin{cases} V_\alpha = \frac{3}{2} \cdot K \cdot \hat{V} \cdot \cos \omega t \\ V_\beta = \frac{3}{2} \cdot K \cdot \hat{V} \cdot \sin \omega t \end{cases}$$



$$\overrightarrow{X_{\alpha\beta}} = |X| \cdot e^{j\theta_{\alpha\beta}}$$

$$\overrightarrow{X_{dq}} = |X| \cdot e^{j(\theta_{\alpha\beta} - \theta_{dq})}$$

Entonces tenemos que:

$$\overrightarrow{X_{dq}} = |X| \cdot e^{j\theta_{\alpha\beta}} \cdot e^{j(-\theta_{dq})} = \overrightarrow{X_{\alpha\beta}} \cdot e^{j(-\theta_{dq})}$$

$$\begin{cases} \overrightarrow{X_{dq}} = \overrightarrow{X_{\alpha\beta}} \cdot e^{-j\theta_{dq}} \\ \overrightarrow{X_{\alpha\beta}} = \overrightarrow{X_{dq}} \cdot e^{j\theta_{dq}} \end{cases}$$

Con estas ecuaciones obtenemos la matriz de Park:

$$\begin{bmatrix} X_d \\ X_q \\ X_0 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_\alpha \\ X_\beta \\ Z \end{bmatrix}$$

Donde $X_0 = 0$

$$X = X_d + jX_q \rightarrow \begin{cases} X_d = X_\alpha \cos(\theta_{dq}) + X_\beta \sin(\theta_{dq}) \\ X_q = X_\beta \cos(\theta_{dq}) - X_\alpha \sin(\theta_{dq}) \end{cases}$$

De donde podemos sacar la matriz inversa:

$$\begin{bmatrix} X_\alpha \\ X_\beta \\ Z \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_d \\ X_q \\ X_0 \end{bmatrix}$$

Si elijo $\omega_d = \omega$

$$\overrightarrow{X_{dq}} = |X| \cdot e^{j\omega t} \cdot e^{-j\omega t} = |X| = \frac{3}{2} \cdot k \cdot \hat{V}$$

En la Fig. 3.30 se muestra el diagrama del proceso que hemos seguido para obtener un generador de señales trifásicas.

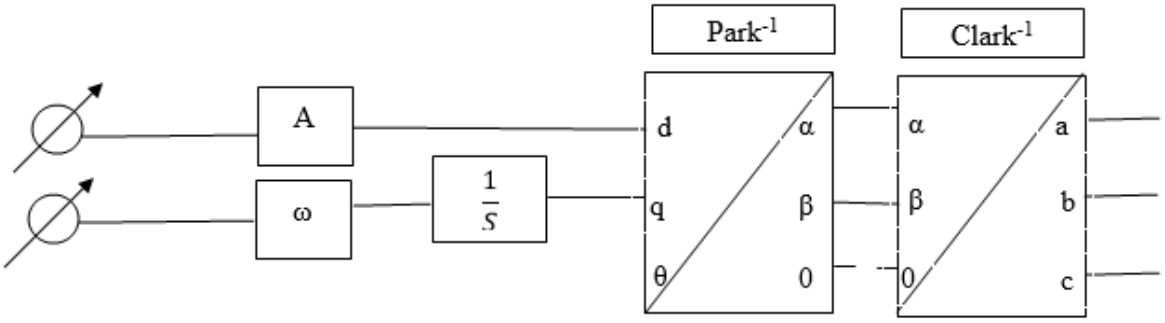


Fig. 3.30. Diagrama de como generar una señal trifásica.

3.4.2. SVPWM

3.4.2.1. Introducción

La modulación por ancho de pulso en el espacio vectorial es una técnica de control muy utilizada en el control de inversores trifásicos. Esta técnica es muy apropiada para implementación en medios digitales, ya que consiste de un arreglo de transistores que funcionan como conmutadores, los cuales son accionados por señales digitales. Existen varios algoritmos para utilizar el SVPWM y modular al inversor. Unos de los algoritmos más populares es el conocido como el de 7 segmentos (u 8 segmentos), el cual se presentará en este trabajo, ya que es un algoritmo que permite un buen desempeño del inversor. [4]

3.4.2.2. Desarrollo

El vector espacial de referencia. En la modulación PWM basada en vectores espaciales se explota la interacción entre las tres fases y en lugar de usar un modulador para cada fase, se procesa un único modulador para el vector espacial de voltaje del conjunto trifásico.

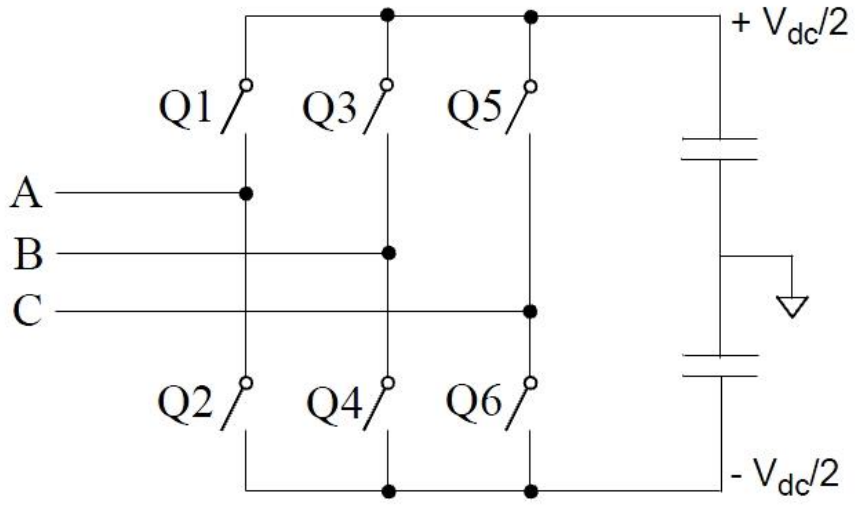


Fig. 3.31. Inversor modulado.

Al aplicar la transformación de Clarke sobre el conjunto trifásico de señales moduladoras de fase, se obtiene el vector espacial de referencia. Cuando las señales moduladoras conforman

un sistema balanceado de señales senoidales, en estado estacionario se caracteriza por poseer amplitud constante y rotar en el plano αβ trazando una trayectoria circular.

La velocidad de rotación y la amplitud del vector de referencia están determinadas por la frecuencia angular w_my la amplitud A de las señales moduladoras, respectivamente, por lo tanto, el vector de referencia puede ser definido mediante la siguiente expresión:

$$\overrightarrow{V^*} = Ae^{jw_m t} = M\frac{V_{DC}}{2}e^{jw_m t}$$

Representación vectorial de los estados del inversor. En este apartado se expondrá la representación de los estados del inversor en el plano αβ. Un vector espacial de voltajes puede ser representado en función de sus voltajes instantáneos de fase. En el caso del inversor, los voltajes de fase en la carga pueden ser escritos así:

$$V_{AN} = V_{A0} - V_{N0}$$
$$V_{BN} = V_{B0} - V_{N0}$$
$$V_{CN} = V_{C0} - V_{N0}$$

Por lo tanto, el vector espacial de voltajes de fase del estator puede ser representado en términos de los voltajes en el inversor:

$$V_{Svn} = V_{A_n} + V_{B_n}e^{j\frac{2\pi}{3}} + V_{C_n}e^{j\frac{4\pi}{3}}$$

Al reemplazar el valor de los estados del inversor se obtienen los valores instantáneos del vector espacial que puede generar el inversor.

$$V_{Svn} = (V_{A_0} - V_{N_0}) + (V_{B_0} - V_{N_0})e^{j\frac{2\pi}{3}} + (V_{C_0} - V_{N_0})e^{j\frac{4\pi}{3}}$$

Que simplificando se quedaría como:

$$V_{Svn} = V_{A_0} + V_{B_0}e^{j\frac{2\pi}{3}} + V_{C_0}e^{j\frac{4\pi}{3}}$$

Estos se resumen en la Tabla 3.1., donde a cada estado j, se ha asociado un vector.

A	B	C	V_{A_0}	V_{B_0}	V_{C_0}	$\overrightarrow{V_{SV}}$
0	0	0	0	0	0	0
0	0	1	0	0	V_{DC}	$V_{DC}e^{j\frac{4\pi}{3}}$
0	1	0	0	V_{DC}	0	$V_{DC}e^{j\frac{2\pi}{3}}$
0	1	1	0	V_{DC}	V_{DC}	$V_{DC}e^{j\pi}$
1	0	0	V_{DC}	0	0	V_{DC}
1	0	1	V_{DC}	0	V_{DC}	$V_{DC}e^{j\frac{5\pi}{3}}$
1	1	0	V_{DC}	V_{DC}	0	$V_{DC}e^{j\frac{\pi}{3}}$
1	1	1	V_{DC}	V_{DC}	V_{DC}	0

Tabla 3.1.Estados del vector espacial.

Este conjunto de vectores se clasifica en dos grupos. Los vectores $\overrightarrow{v_0}$y $\overrightarrow{v_7}$ corresponden al grupo de vectores nulos o vectores cero y los vectores $\overrightarrow{v_1}$a $\overrightarrow{v_6}$ corresponden al grupo de vectores activos o vectores básicos.

Los vectores activos dividen el plano αβ en seis sectores formando los ejes de un hexágono. Este hexágono es conocido como el hexágono del inversor.

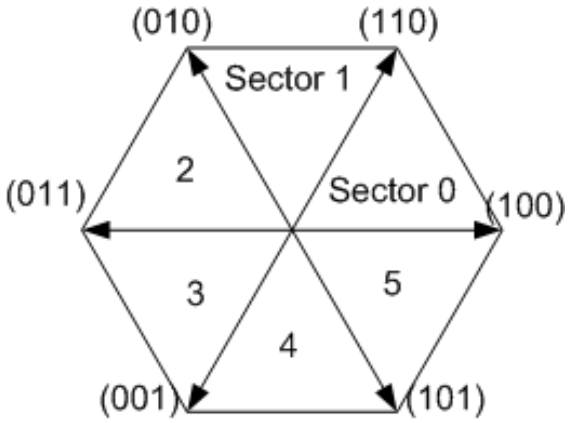


Fig. 3.32. Hexágono del inversor.

Una vez conocido en el sextante en el que vamos a trabajar y conocidas sus proyecciones se podrán obtener los ciclos de trabajo de los transistores donde:

$$V_{\alpha\beta} = V_{xxx} \cdot d_1 + V_{xxx} \cdot d_2 + \frac{V_{000}}{V_{111}} \cdot \frac{d_3}{2}$$

El vector nulo que se elegirá será el que menos conmutaciones tenga.

Conociendo las proyecciones del vector $V_{\alpha\beta}$ sobre los ejes ejes Z_{1X} y Z_{1y} podremos determinar si está en el primero o en el cuarto sextante.

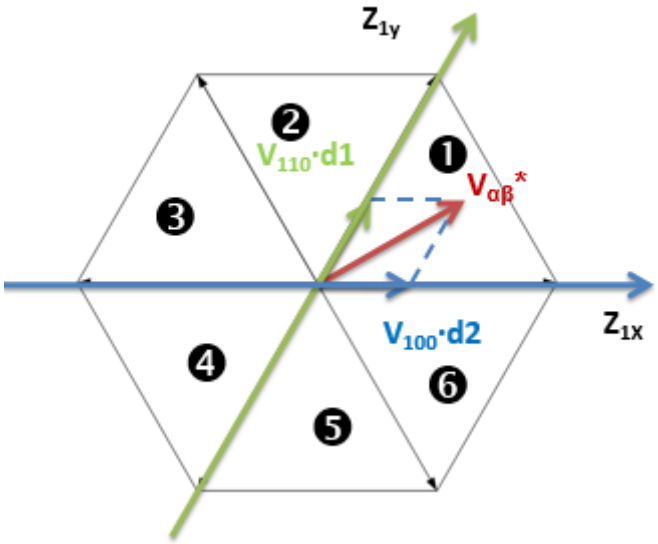


Fig. 3.33, Proyecciones en el primero o cuarto sextante.

Conociendo las proyecciones del vector Vαβ sobre los ejes Z_{2X} y Z_{2y} podremos determinar si está en el segundo o en el quinto sextante.

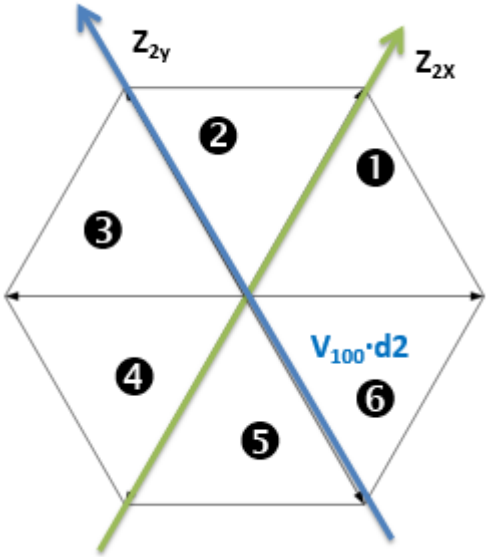


Fig. 3.34. Proyecciones en el segundo y quinto sextante.

Conociendo las proyecciones del vector Vαβ sobre los ejes Z_{3X} y Z_{3y} podremos determinar si está en tercero o en el sexto sextante.

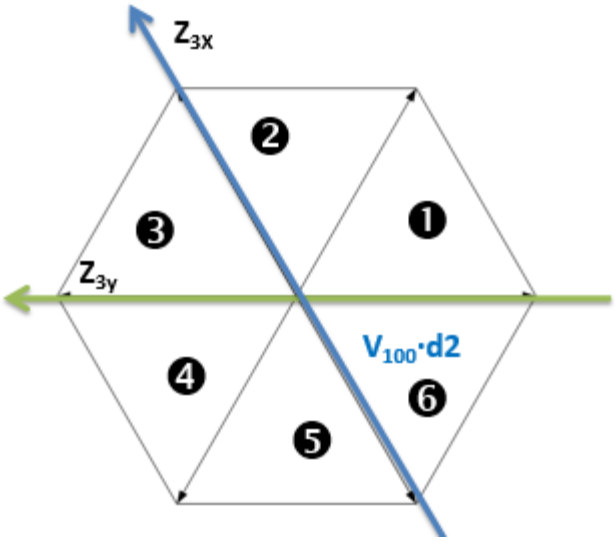


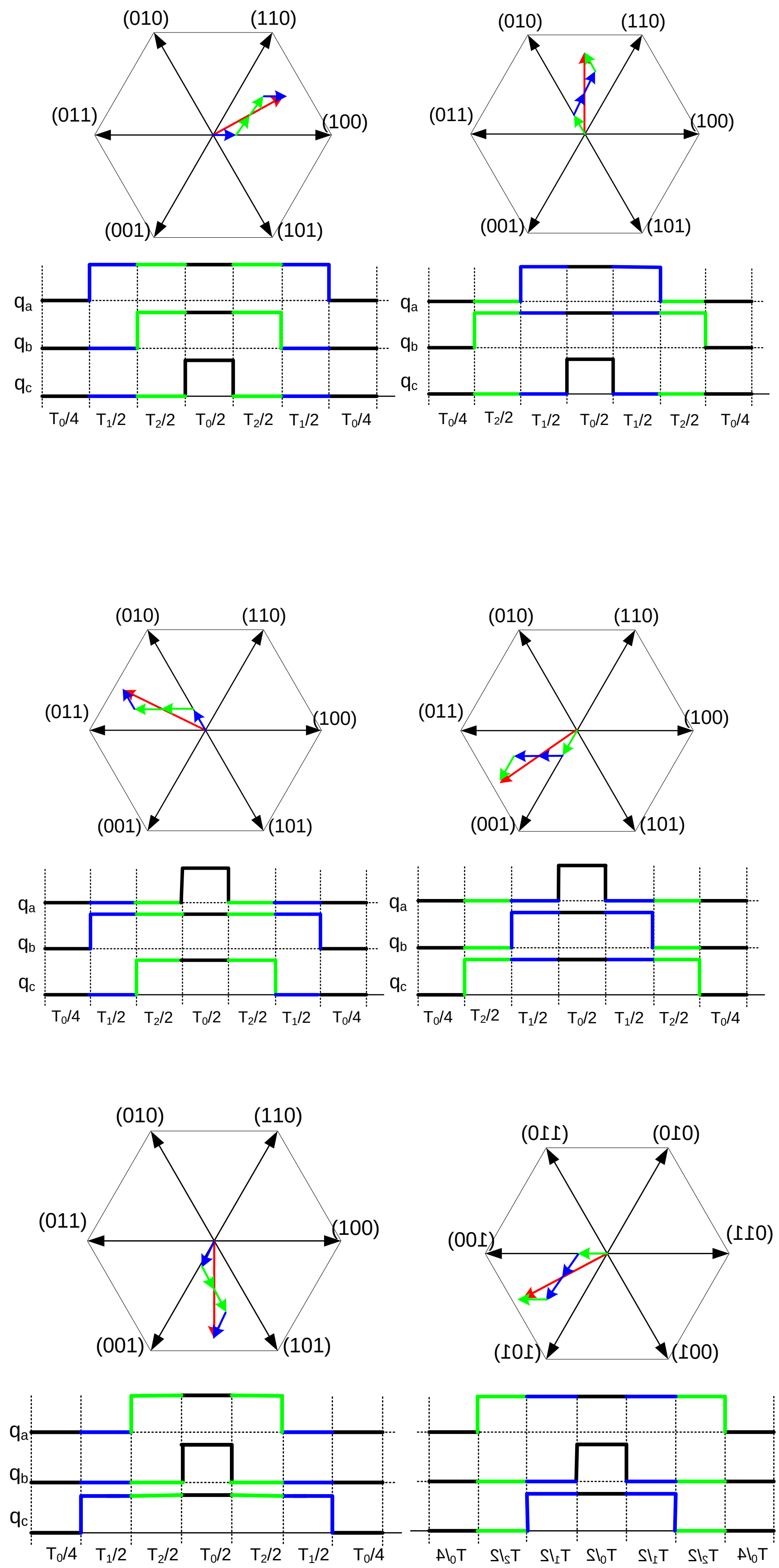
Fig. 3.35. Proyecciones en el tercero y el sexto sextante.

De la anterior ecuación se despejará d_1 y d_2 y se podrá obtener el ciclo de trabajo de d_3 que será:

$$d_3 = 1 - d_1 - d_2$$

La secuencia de activación de transistores se llevará de tal forma que solo cambie un polo de potencia a la vez, de esta forma minimizaremos las conmutaciones. Esto llevará a que la secuencia en los sectores pares sea diferente a los impares.

La obtención de los ciclos útiles de los canales PWM. Una vez establecidos los tiempos de activación de los vectores y la secuencia de conmutación, se deben traducir estos resultados a ciclos útiles de los canales PWM que alimentarán las compuertas del inversor. Teniendo el patrón de conmutación resulta sencillo identificar estos ciclos útiles.


(010)
(110)
(011)
(100)
(001)
(101)
qa
qb
qc
T0/4 T1/2 T2/2 T0/2 T2/2 T1/2 T0/4
T0/4 T2/2 T1/2 T0/2 T1/2 T2/2 T0/4

Una vez obtenidos los ciclos útiles de los canales PWM se pueden deducir las señales trifásicas de control.

3.4.2.3. Sobremodulación

Se produce sobremodulación cuando la amplitud es más grande que la circunferencia circunscrita en el hexágono del inversor.

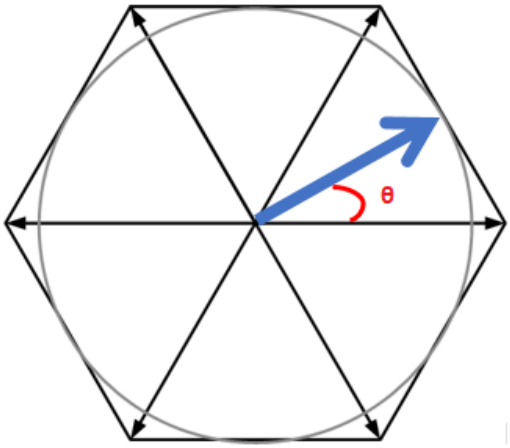


Fig. 3.36. Si se supera la circunferencia circunscrita se produce sobremodulación.

$$|V_{\alpha\beta}|_{max} = V_{DC} \cdot \sqrt{\frac{2}{3}} \cdot \cos\theta$$

Capítulo 4

4. Proyecto

4.1.Primera fase: Aprendizaje de LabVIEW

4.1.1. Introducción

En esta se estudió el manual de LabVIEW FPGA[21] y fueron visualizados los 14 vídeos tutoriales sobre como programar con Labview en FPGA. En el nombre de los vídeos se adjunta el hipervínculo del citado vídeo.

4.1.2. Contenido de los Vídeo Tutoriales

1. Writing Your First LabVIEW FPGA Program

En este vídeo se aprende como usar el software de diseño gráfico LabVIEW para programar una FPGA.

Al finalizar este vídeo de 6:32 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Utilizar estructuras gráficas y nodos de E/S para construir circuitos digitales personalizados.
- Compilar el diagrama de bloques para ejecutar código de LabVIEW en el hardware.

2. Implementing Counters in LabVIEW FPGA

Las FPGAs son excelentes para la implementación de contadores, y LabVIEW le permite implementarlos gráficamente.

[21] FPGA: Field Programmable Gate Array

Al finalizar este vídeo de 6:32 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Crear un contador de eventos simples en LabVIEW FPGA para contar flancos ascendentes digitales
- Mostrar el valor del contado de registro en los LED de un módulo NI CompactRIO

3. Using Analog Inputs and Outputs in LabVIEW FPGA

Se puede utilizar los nodos de E/S en LabVIEW FPGA para generar señales analógicas y tomar medidas analógicas.

Al finalizar este vídeo de 5:51 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Generar un voltaje analógico utilizando un módulo de salida analógica CompactRIO.
- Medir la tensión de la salida usando un módulo de entrada analógica CompactRIO.

4. Using Graphical Loop Structures in LabVIEW FPGA

A diferencia de las CPU[22], el hardware de una FPGA permite ejecutarse en paralelo y LabVIEW FPGA tiene una estructura de bucle gráfico que permite ejecutar simultáneamente diferentes partes de su diagrama de bloques.

Al finalizar este vídeo de 6:14 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Utilizar múltiples estructuras “While Loop” LabVIEW FPGA para crear circuitos independientes.
- Utilizar las funciones “Loop Timer” para especificar la rapidez a la que se deben de ejecutar los diferentes bucles.

[22] CPU: Central Processing Unit

5. Measuring Loop Timing in LabVIEW FPGA

LabVIEW FPGA permite ejecutar diagramas de bloques gráficos en hardware, que pueden ejecutarse en el orden de microsegundos y nanosegundos.

Al finalizar este vídeo de 4:43 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Saber cómo funciona la sincronización de bajo nivel cuando el código de LabVIEW se está ejecutando en la FPGA
- Utilizar las funciones de recuento de paso en LabVIEW FPGA para controlar la velocidad de ejecución del bucle.

6. Single-Cycle Timed Loops in LabVIEW FPGA

En este vídeo se aprende acerca de "Single-Cycle Timed Loop", una estructura especial de LabVIEW FPGA que le permite optimizar su diseño de FPGA para ambos tamaños y velocidad.

Al finalizar este vídeo de 4:34 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Ejecutar la lógica FPGA en un "Single-Cycle Timed Loop" dentro de un solo "tick".
- Lograr 25 tasas de nanosegundos al reloj de la compilación por defecto de 40 MHz.

7. Debouncing Digital Signals in LabVIEW FPGA

Interruptores y relés mecánicos a menudo suelen rebotar al cambiar de estado, se puede usar LabVIEW FPGA para implementar el rebote de circuitos y filtrar los bordes digitales no deseados.

Al finalizar este vídeo de 9:10 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Aplicar gráficamente un filtro de rebote digital en un sencillo contador de eventos.

- Programar una mínima cantidad fija de tiempo para identificar transiciones válidas.

8. Using Feedback Nodes in LabVIEW FPGA

Además de utilizar registros de desplazamiento, también se puede utilizar "Feedback Nodes" en LabVIEW FPGA para pasar los datos entre diferentes bucles.

Al finalizar este vídeo de 4:16 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Aprender como "Feedback Nodes" pueden ser útiles para hacer diagramas de bloques de Labview más fáciles de leer.
- Utilizar "Feedback Nodes" dentro de subVIs para crear funciones modulares que puedan contener datos entre iteraciones de bucle.

9. Generating Signals in LabVIEW FPGA

Se puede generar dinámicamente formas de onda utilizando LabVIEW FPGA y salidas analógicas con los nodos de E/S.

Al finalizar este vídeo de 8:48 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Generar continuamente valores sobre la FPGA utilizando las funciones de "Sine Wave Generator".
- Generar una señal de tensión sinusoidal de salida y leerlo de nuevo con un canal de entrada analógica.

10. Using Graphical Case Structures in LabVIEW FPGA

Las "Case Structures" se utilizan para activar selectivamente partes de su diseño de FPGA y son buenas para la implementación de hardware personalizado, triggers y máquinas de estado.

Al finalizar este vídeo de 3:27 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Crear gráficamente condiciones temporizadas con LabVIEW para "case structures".
- Utilizar una señal de entrada digital para activar y desactivar un canal de entrada analógica en el hardware

11. Finite Sampling Using For Loops in LabVIEW FPGA

"For Loops" son estructuras útiles cuando ya se conoce el número de iteraciones que se desea ejecutar el código, y trabajar bien para un número finito de muestras de voltaje analógicas.

Al finalizar este vídeo de 4:12 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Utilizar "For Loops" en LabVIEW para predeterminar el número de muestras analógicas a tomar.
- Combinar "For Loop" con "While Loop" para crear una aplicación de adquisición de datos reactivable.

12. Implementing Simple Event Triggers in LabVIEW FPGA

Una transición digital de falso a verdadero (flanco ascendente) o de verdadero a falso (Flanco descendente) podría ser solo el evento de disparo que su aplicación necesite

Al finalizar este vídeo de 5:59 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Monitorear continuamente una línea de entrada digital usando "Single-Cycle Timed loop".
- Usar un flanco ascendente para activar el bucle de adquisición de datos.

13. Custom Analog Triggering in LabVIEW FPGA

En LabVIEW FPGA se puede configurar el tipo exacto de la condición de disparo que se necesita, en función del valor de los canales de entrada analógicos.

Al finalizar este vídeo de 7:16 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Monitorear continuamente un canal de entrada analógica y hacer uso solo de las muestras que estén por encima de un determinado umbral definido por el usuario.
- Implementar un disparador "OR" para especificar múltiples condiciones de disparo dentro del hardware de la FPGA.

14. Configuring Independent Analog Channels in LabVIEW FPGA

La mayoría de los dispositivos de adquisición de datos están diseñados para compartir relojes de muestreo y disparos, pero con LabVIEW FPGA se puede aplicar diferentes sistemas de sincronización en el hardware y lograr una verdadera operación independiente.

Al finalizar este vídeo de 7:42 minutos, el usuario deberá estar preparado para realizar las siguientes tareas:

- Usar una estructura paralela de bucle para controlar los canales de entrada y salida analógica independientemente.
- Configurar diferentes relojes de muestreo y las condiciones de alarma sin afectar a otros canales de E/S.

4.2.Segunda fase: Ejemplos de programación

4.2.1. Introducción

El programa está compuesto por varios códigos distintos. Estos códigos tienen una jerarquía que le da una clara importancia dependiendo del nivel en el que están. En la Fig. 4.1, se muestra como están distribuidos los diferentes códigos dentro de la Single-Board RIO.

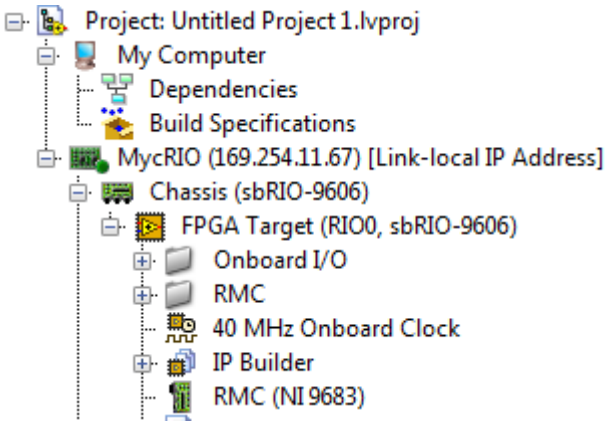


Fig. 4.1. Jerarquía del programa de control

Como se observa hay dos niveles de programación. El primero se ejecuta dentro del ordenador (My Computer), aunque puede compilarse dentro del controlador. Esto es debido a que la velocidad de comunicación entre la Single-Board RIO y el ordenador es suficientemente más grande que la velocidad con la que se ejecutan las instrucciones dentro del código, de esta forma la compilación es más rápida. El segundo se compila dentro de la FPGA debido a que este código desarrolla funciones que deberán ejecutarse lo más rápido posible para obtener el mayor control posible.

Las comunicaciones entre el ordenador se realizan a través de un cable Ethernet y mediante una dirección de IP[23] (169.254.11.67)

[23] IP: Internet Protocol

4.2.2. Ejemplos de programación

A continuación, se detallará los ejemplos de programación que hemos llevado a cabo para comprobar el funcionamiento del hardware y la verificación de los conocimientos adquiridos a la hora de programación.

4.2.2.1. Led[24]

El primer programa a realizar consistió en introducir un interruptor dentro de un bucle que se ejecutará constantemente.

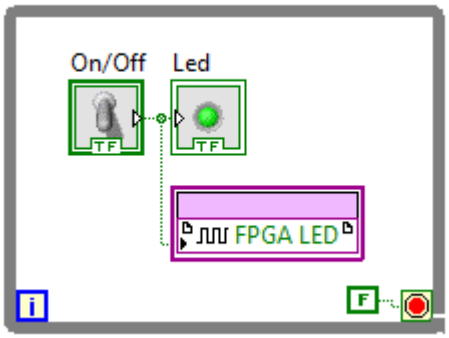


Fig. 4.2. Diagrama de bloques LED.vi

El interruptor se conectó al Led de la FPGA y se le colocó un símbolo booleano para visualizar el encendido y apagado en el ordenador.

Cada vez que se subía el interruptor tanto el led de la FPGA como el símbolo para la monitorización, se encendían. Cuando este se bajaba, ambos led se apagaban.

Con este primer programa conocido como "Hola mundo" en la Single-Board RIO, se pudo comprobar el buen funcionamiento entre la FPGA y el ordenador.

[24] Led: Light Emitting Diode

4.2.2.2. Comparador

El segundo programa que hemos realizado es una variación del anterior, pero en este no hemos utilizado un interruptor para encender un led.

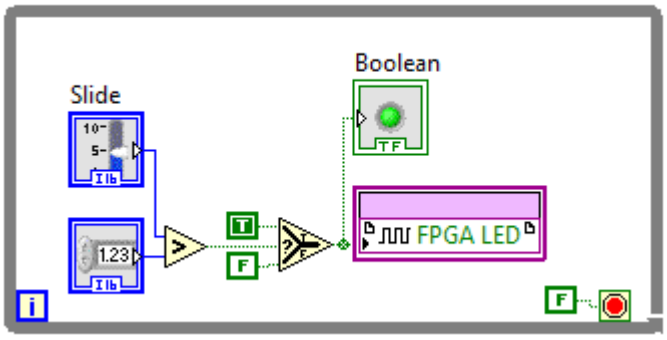


Fig. 4.3. Diagrama de bloques comparador.vi

Este programa consistirá en introducir dos datos enteros de 16 bit con signo que irán a un comparador donde si A>B que encenderá el Led y si A>B se apagará el Led.

4.2.2.3. Termómetro

En este programa se utiliza la temperatura de la FPGA que nos la dan en grados Farenhait y el programa consistirá en la utilización de datos en punto fijo para conseguir la temperatura en grados Celsius.

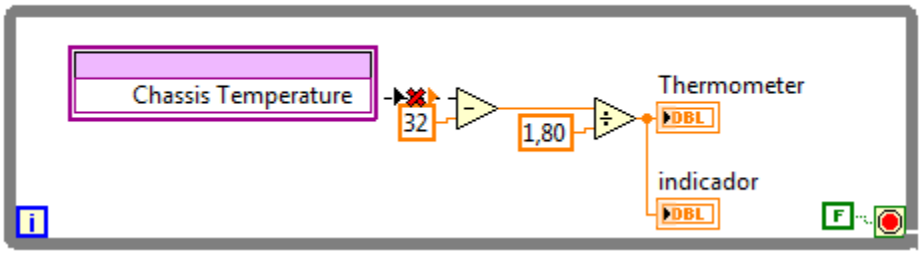


Fig. 4.4. Programa de conversión de ºF a ºC de la sbRIO.

4.2.2.4. Pulsos

Este programa consistirá en la implementación de un contador en el que cada vez que se apague o baje un interruptor para apagar el Led de la FPGA este se incrementará. El contador se visualizará a través de un indicador que mostrará los datos enteros de 8 bits sin signo.

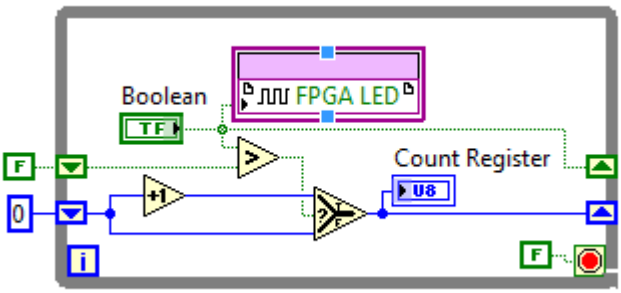


Fig. 4.5. Diagrama de bloques pulsos.vi

A continuación se mostrará en la Fig. 4.6 como sería la interfaz que visualizaría el usuario para poder controlar este programa.

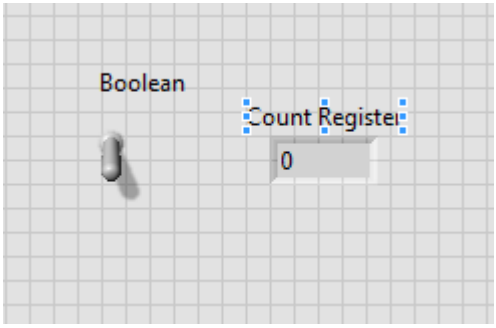


Fig. 4.6. Interfaz visual pulsos.vi

4.2.2.5. Comparador utilización I/O

Este es el último programa que se realizará como ejemplo antes de pasar a la siguiente fase. Este consistirá en implementar un comparador como el anterior. La diferencia estará en que se utilizarán las entradas y salidas analógicas. Para ello utilizaremos los bloques A0/AO0 y Scanned/AI0 que encontraremos dentro de la RMC[25] (NI-9683).

[25] RMC: RIO Mezzanine Card

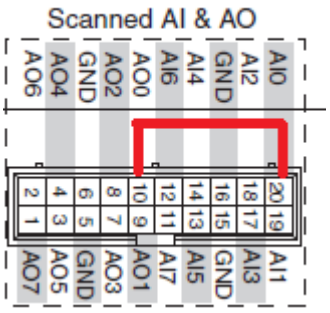


Fig. 4.7. Conexiones de la Single-Board RIO.

A través de un "slide" se introducirán un valor de tipo entero de 16 bits con signo. Este dato lo convertiremos a un dato de punto fijo, de 16 bits de ancho de palabra y de 16 bits de parte entera, ya que las entradas y salidas analógicas de la Single-Board RIO trabajan con este tipo de dato.

Los datos se enviarán por el pin de salida AO/AO0 y se recibirán por el pin de entrada Scanned/AI0. Estos datos se compararán con un valor que nosotros introduciremos por "y" que será un dato de punto fijo de 16 bits de ancho de palabra y 16 bits de parte entera. Cuando el dato que recibamos por Scanned/AI0 sea más grande que el dato de "y" se encenderá tanto el Led de la FPGA como un indicador luminoso en la parte de monitorización.

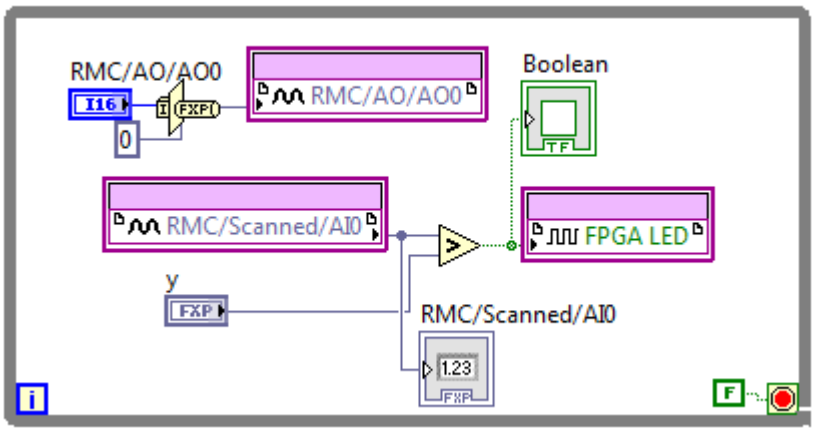


Fig. 4.8. Diagrama de bloques comparador ES.vi

4.3. Tercera fase: construcción del controlador de velocidad de un motor asíncrono

4.3.1. Introducción

En esta fase se va a explicar cómo se ha construido el programa de control. Este se ha elaborado tras el ensamblaje de varios programas, que los habremos programado y probado previamente.

4.3.2. Generado trifásico de señales desarrollado en lenguaje de programación G

4.3.2.1. Introducción

Para comenzar a elaborar el generador trifásico de señales hemos programado un diagrama de bloques utilizando el "Formula Node", que son unos nodos que le puedes meter todas las fórmulas matemáticas y estas realizan las operaciones.

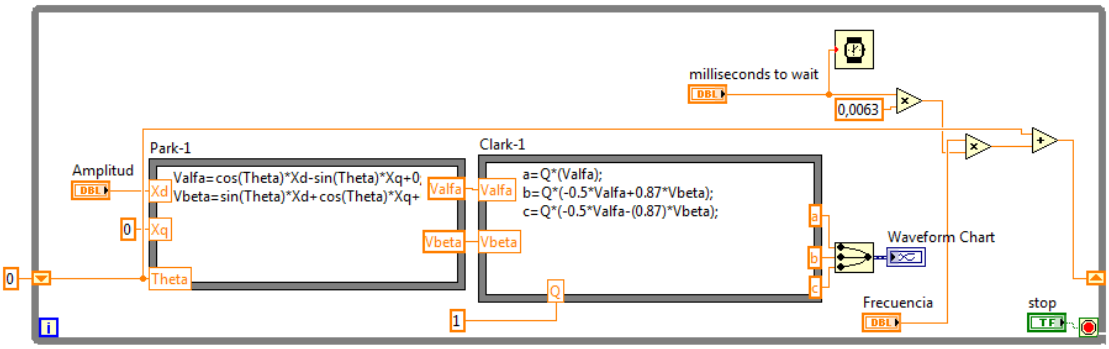


Fig. 4.9. Diagrama de bloques utilizando Formula Node.

Esta es una buena forma de comprobar que la implementación matemática para elaborar el generador de señales trifásico es correcta.

4.3.2.2. Código .vi

Este programa se ejecuta dentro de la FPGA y consiste en introducir dos parámetros como son la amplitud y la frecuencia, y a través de estos parámetros y lo explicado anteriormente en el capítulo 3, el marco teórico, se consigue generar tres señales desfasadas 120 grados, entre ellas. Estas señales se juntan en una sola señal para mandarlas a través de un bloque FIFO[26] al programa de monitorización.

En este programa se han utilizado matemáticas de alto rendimiento para las FPGA y así optimizar los recursos de esta. En la Tabla 4.1 se comentará las variables implementadas en el programa y en la Fig. 4.10 se mostrará el código implementado.

Nombre	Tipo	Descripción
Amplitud	FXP	Es la amplitud con la que se quiere generar las señales.
Freq (Hz)	FXP	Es la frecuencia a la que se quiere generar las señales.
stop	BOOL	Controla la finalización del bucle.
Q	FXP	Es una constante.
sqrt(3)/2	FXP	Es una constante.
0	I16	Es una constante.

Tabla 4.1. Descripción de las variables del sistema.

[26] FIFO: First In First Out

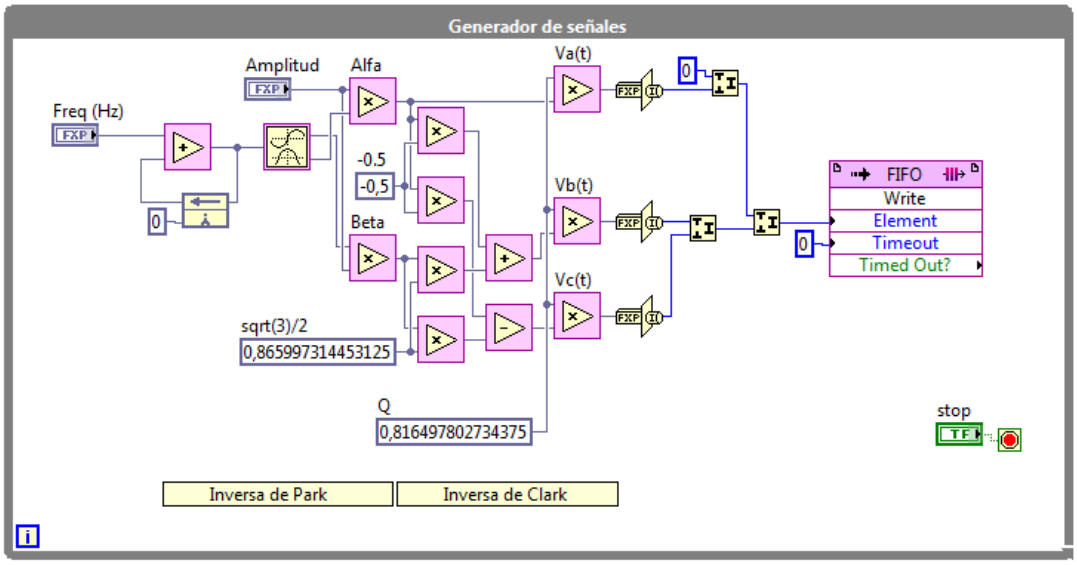


Fig. 4.10. Diagrama de bloques código.vi

4.3.2.3. Código monitorización .vi

Este programa se ejecuta en el ordenador y es el que se comunica con la FPGA. A través de sus bloques de lectura y escritura se pueden enviar los datos con los que queremos ejecutar el programa código.vi así como leer los datos que se generan en este mismo programa. Los datos se recibirán a través del bloque FIFO que serán introducidos en el código.vi y que se definirán posteriormente.

Nombre	Tipo	Descripción
Plot1	BOOL	Enciende y apaga la visualización de las señal Va(t)
Plot2	BOOL	Enciende y apaga la visualización de las señal Vb(t)
Plot 3	BOOL	Enciende y apaga la visualización de las señal

		Vc(t)
Plot11	U32	Caja framed color box de color enmarcada con la que podemos cambiar los colores de la señal Va(t)
Plot22	U32	Caja framed color box de color enmarcada con la que podemos cambiar los colores de la señal Va(t)
Plot3	U32	Caja framed color box de color enmarcada con la que podemos cambiar los colores de la señal Va(t)
Freq (hz)	FXP	Es la frecuencia a la que se quiere generar las señales.
Amplitud	FXP	Es la amplitud con la que se quiere generar las señales.

Tabla 4.2. Descripción de las variables del sistema.

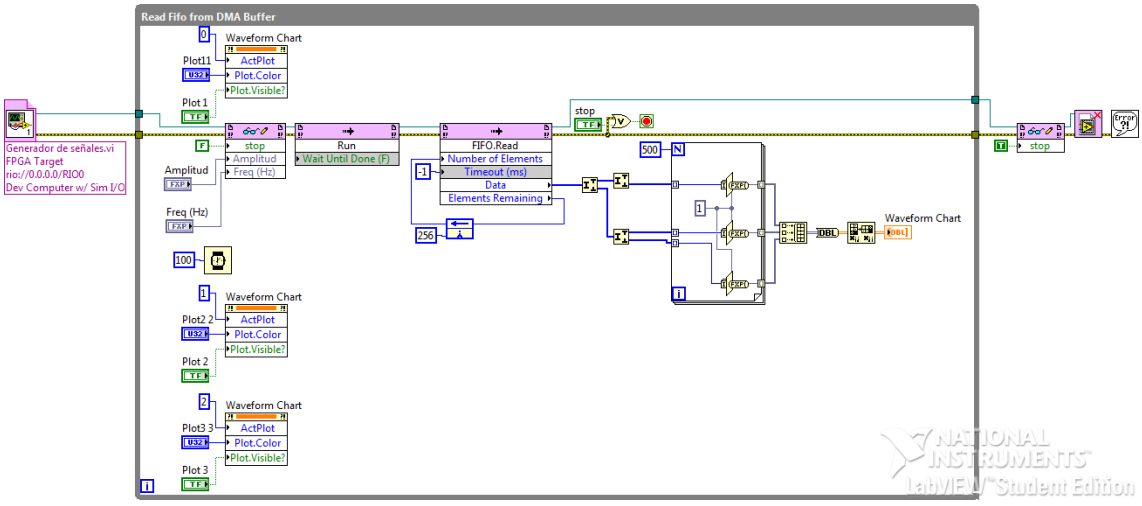


Fig. 4.11. Diagrama de bloques monitorización.vi

4.3.2.4. Bloque FIFO

El módulo FIFO mejora la transferencia de datos en las aplicaciones de adquisición de datos de alta velocidad búfer en dispositivos CompactRIO o R Series. La FPGA FIFO se puede configurar para dar acceso directo a memoria (DMA[27]). Esto ofrece mejoras de rendimiento significativas sobre el uso de la FIFO local y la lectura de los indicadores de uso de la FPGA Interface Host VIs para transferir datos de la FPGA.

Las transferencias de DMA se llevan a cabo mediante una arquitectura FIFO. La FIFO se compone de 2 partes que se comporta como una. La primera parte está en el dispositivo FPGA. Esta utiliza un bloque de memoria RAM[28] en la FPGA. La segunda parte de la DMA FIFO está en la maquina host. Esta parte utiliza la memoria de la maquina host. La DMA transfiere automáticamente los datos de la memoria RAM de la FPGA a la memoria de la maquina host.

A continuación, se muestran el cuadro de propiedades de la FIFO. En este caso se envían datos de tipo sin signo de 64 bit (U64).

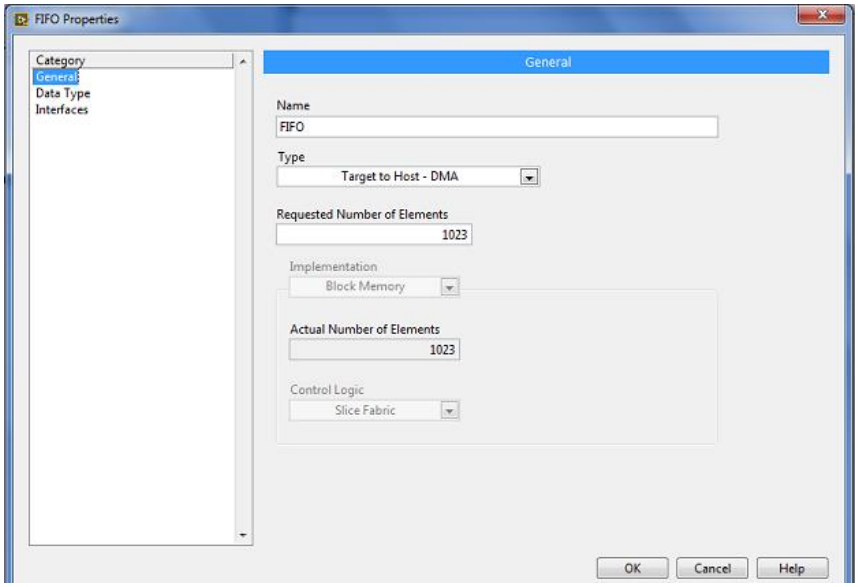


Fig. 4.12. Propiedades FIFO.

[27] DMA: Direct Memory Access

[28] RAM: Random Access Memory

4.3.3. Generador de señal triangular desarrollado en lenguaje de programación G

4.3.3.1. Codigo.vi

Este programa consistirá en generar una señal triangular con una frecuencia de 1Khz. Para esto se utilizará un bucle "While Loop timed" en el que el tiempo vendrá por el reloj interno de la FPGA que es de 40MHz. Para generar una señal triangular de 1KHz que será nuestra frecuencia mínima se necesitará que la señal se genere en 40.000 muestras, ya que si cada muestra se ejecuta con una frecuencia de 40MHz las 40.000 se ejecutará en 1KHz (40M /40k = 1K) y así conseguiremos la frecuencia de nuestra señal.

Para implementar este código se partirá de un valor inicial (-10.000) y de una constante de tipo booleana (TRUE) en cada iteración de bucle se irán incrementando hasta llegar a los 10.000.

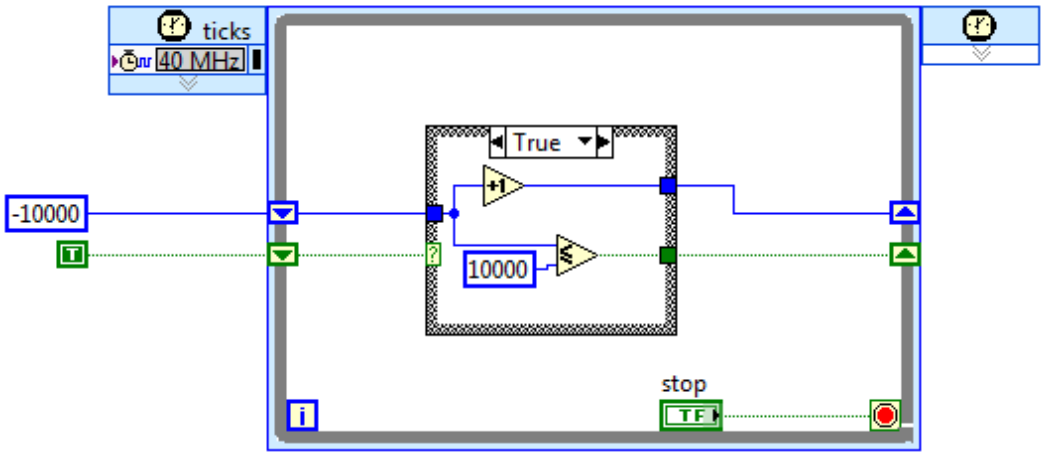


Fig. 4.13. Diagrama de bloques código.vi

Una vez llegado a los 10.000 se cambiará la variable de tipo booleana (FALSE) y se irá decrementando hasta llegar a los -10.000 donde se volverá a cambiar la variable de tipo booleana (TRUE).

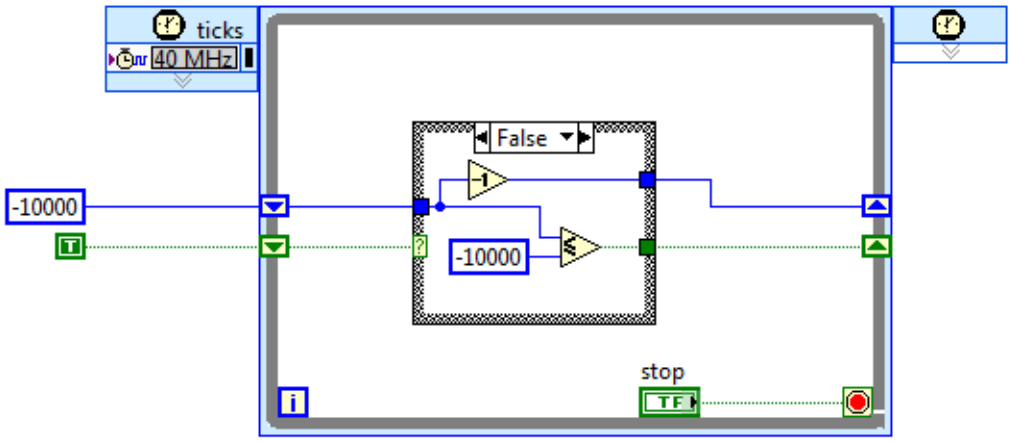


Fig. 4.14. Diagrama de bloques codigo.vi

Esta señal triangular será la que posteriormente compararemos con la salida del generador de señales trifásica para la obtención del PWM.

4.3.4. Generador de rampas de arranque y parada desarrollado en lenguaje de programación G.

4.3.4.1. Introducción

Para hacer un programa que genere una rampa de arranque y de parada, crearemos un VI que implemente las funciones necesarias. Este primer VI será de prueba y no se compilará en la FPGA.

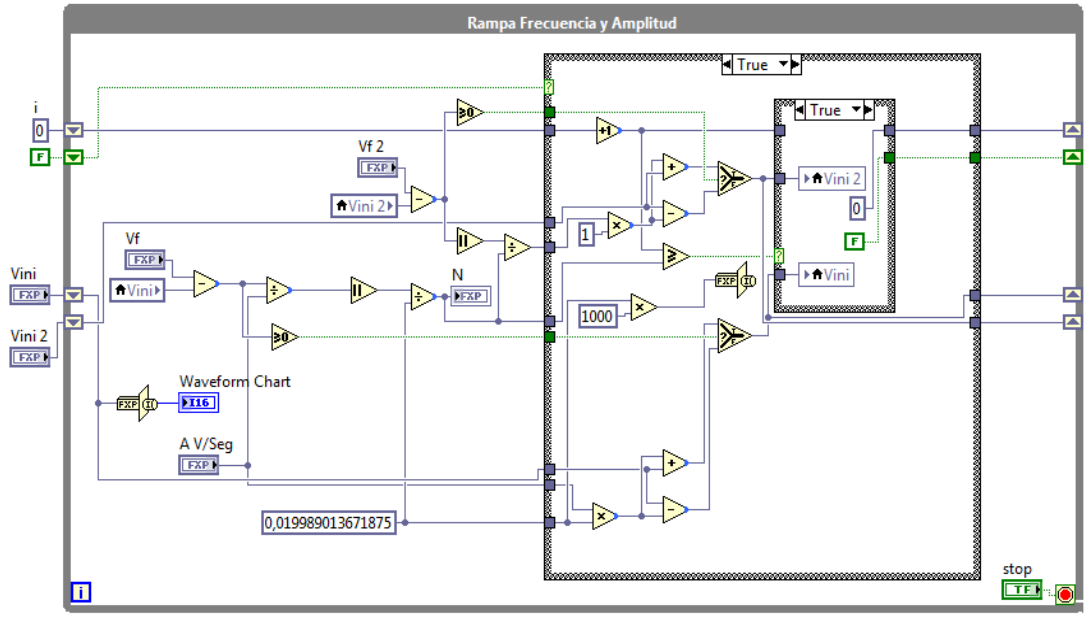


Fig. 4.15. Diagrama de bloques codigo.vi

4.3.4.2. Código.vi

Este programa será el que se ejecutará en la FPGA. Este se ha implementado con matemáticas de alto rendimiento para optimizar los recursos de la FPGA.

Consiste en introducir dos valores finales (Vf, Vf 2), ya que los primeros valores iniciales que tendremos serán ambas variables a 0 y después esta variable irá actualizándose al valor final que haya generado la rampa. Otra de las variables a introducir será “A V/Seg” que esta marcará

la velocidad con la que se generará la rampa. Una vez introducido estos datos se harán una serie de operaciones para calcular cuanta distancia hay entre la variable inicial y la final y se calculará el incremento con el que se generará la rampa y el número de iteraciones en las que se sumará dicho incremento. Una vez hecho esto el programa esperará hasta que el usuario pulse el botón "Ok". Una vez hecho esto se empezará a generar la rampa con el incremento calculado anteriormente y con una frecuencia de 50Hz.

Nombre	Tipo	Descripción
Vini	FXP	Variable de control con el que podremos configurar los valores iniciales de la rampa.
Vini 2	FXP	Variable de control con el que podremos configurar los valores iniciales de la rampa.
Vf	FXP	Variable de control con el que podremos configurar los valores finales de la rampa.
Vf 2	FXP	Variable de control con el que podremos configurar los valores finales de la rampa.
A V/Seg	FXP	Variable de control con la que configuraremos la velocidad con la que se genere la rampa.
Freq (Hz)	FXP	Indiciador que muestra cómo se genera la rampa.

Amplitud	FXP	Indiciador que muestra cómo se genera la rampa.
OK	BOOL	Elemento de control con el que controlaremos el arranque de la rampa.

Tabla 4.3. Descripción de las variables del sistema.

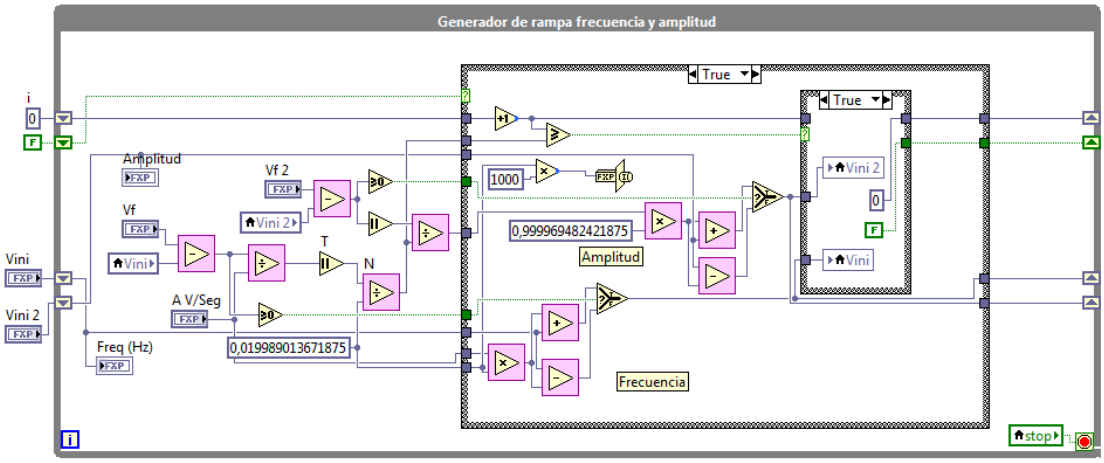


Fig. 4.16. Diagrama de bloques codigo.vi

4.3.5. Programa de control de la velocidad de un motor asíncrono a través de un PWM desarrollado en lenguaje de programación G.

4.3.5.1. Codigo.vi

Este programa es el resultado final de la suma de los tres anteriores descritos anteriormente. Los tres programas se ejecutan paralelamente. En estos programas se han creado variables locales para que haya una interacción entre las 3 etapas. Aunque ya se ha explicado las 3 etapas por separado ahora se van a explicar la interacción entre las 3.

Para comenzar el usuario tendrá que introducir los siguientes valores finales de la frecuencia y de la amplitud (Vf y Vf2). Una vez introducidos estos valores y pulsado el botón de

"OK" se empezará a generar una rampa en estas variables. Estos datos que se van generando se podrán visualizar en los indicadores de "Amplitud" y "Freq (Hz)" que estos se enviarán como variable local al generador de señales.

Cuando las variables locales del generador de señales de "amplitud" y "Freq (Hz)" comienzan a recibir datos, se empieza a generar 3 señales trifásicas, las cuales se multiplicarán a su salida por una variable local "x" que serán comentadas más adelante, obteniéndose los siguientes indicadores "Va(t), Vb(t) y Vc(t)".

Las variable "Va(t), Vb(t) y Vc(t)" son recibidas como variables locales en el generador de señal triangular. En este bucle While Loop se genera la variable "X" que se multiplica como variable local en el generado de señales. Esta variable representa la amplitud de la señal triangular. Las variables locales que contienen las señales obtenidas por el generador de señales se comparan con la señal triangular obteniendo el PWM que lo enviaremos por los pines de salida Half-Bridge de la FPGA.

Cabe mencionar que todos los bucles se detienen con el mismo botón de control que se ubica en el bucle de generador de señales y se extiende a los demás bucles por medio de variables locales.

Parte	Nombre	Tipo	Descripción
Generador de señales	F. Señal triangular	U32	Variable de control en la que podemos introducir la frecuencia de la señal triangular.
	Va(t)	I16	Indicador para visualizar la señal Va(t)

	Vb(t)	I16	Indicador para visualizar la señal Vb(t).
	Vc(t)	I16	Indicador para visualizar la señal Vc(t).
	Stop	BOOL	Controla la parada de ejecución de los bucles While Loop.
Generador de la señal triangular	Señal Triangular	I16	Variable de control de la amplitud de la señal triangular.
	Triangle wave	I16	Indicador de la señal triangular.
	X	I16	Indicador de la amplitud de la señal triangular.
Generador de rampa frecuencia y amplitud	Vini	FXP	Variable de control inicial de la frecuencia.
	A V/Seg	FXP	Variable de control con la que configuraremos la velocidad con la que se genere la rampa.
	Vini 2	FXP	Variable de control inicial de la amplitud.
	Vf	FXP	Variable de control final de la frecuencia.

	Vf2	FXP	Variable de control final de la amplitud.
	Amplitud	FXP	Indicador de la amplitud.
	Freq (Hz)	FXP	Indicador de la frecuencia.
	Ok	BOOL	Pulsador de arranque de la rampa.

Tabla 4.4. Variables del sistema.

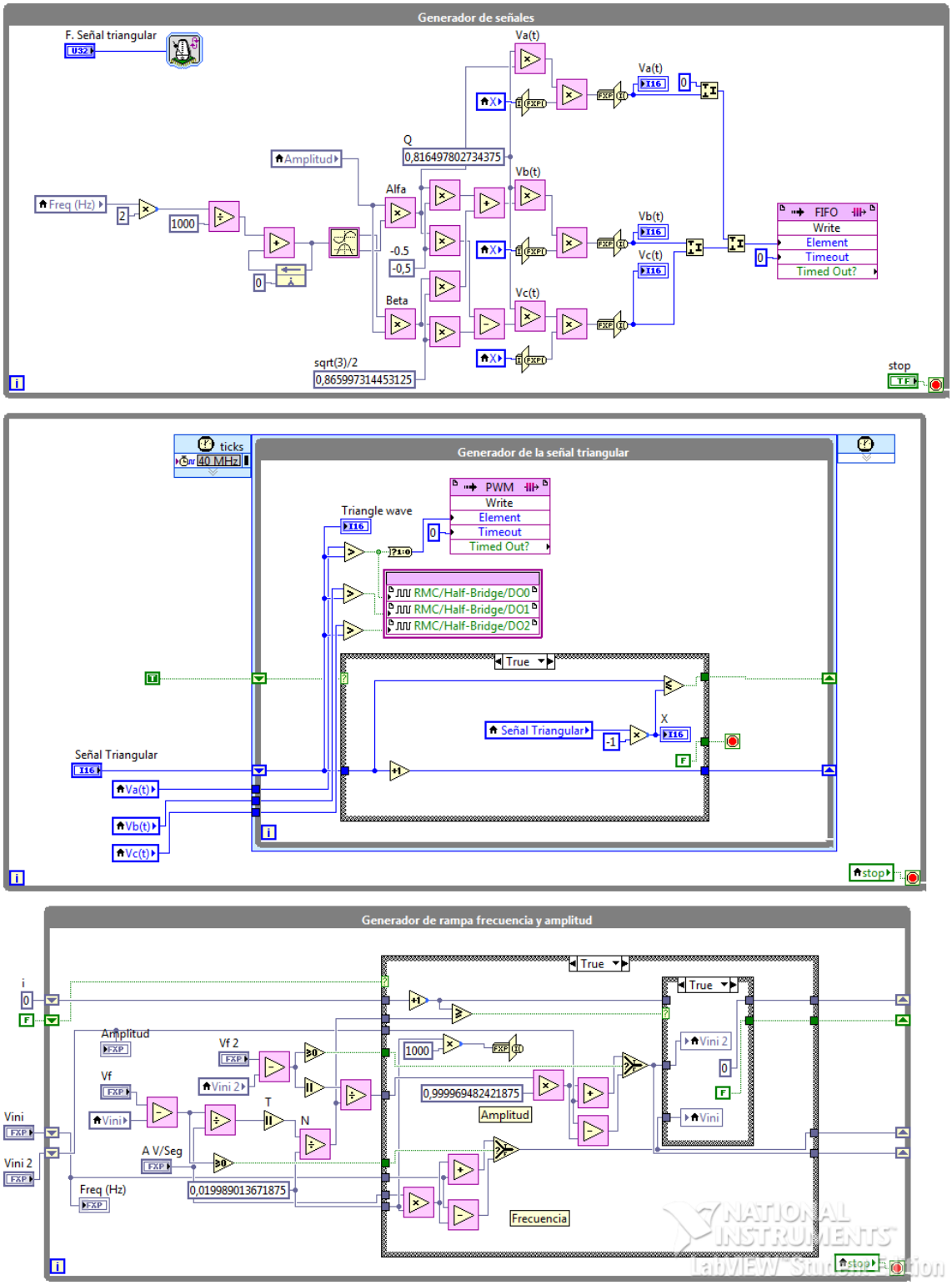


Fig. 4.17. Diagrama de bloques codigo.vi

4.3.5.2. Código de monitoriacion.vi

Como ya hemos visto anteriormente, este programa se compilará en ordenador, en el que leeremos y escribiremos las variables que se quieren ejecutar o visualizar en "código.vi"

Nombre	Tipo	Descripción
Amplitud	FXP	Variable de control final de la amplitud.
Freq (Hz)	FXP	Variable de control final de la frecuencia.
OK	BOOL	Pulsador de arranque de la rampa.
A V/Seg	FXP	Variable de control con la que configuraremos la velocidad con la que se genere la rampa.
Freq. Señal Triangular	U32	Variable de control en la que podemos introducir la frecuencia de la señal triangular.
Va(t)	U32	Caja framed color box de color enmarcada con la que podemos cambiar los colores de la señal Va(t)
Va(t)	BOOL	Interruptor para la visualización de la señal Va(t)
Vb(t)	U32	Caja framed color box de color enmarcada con la que podemos cambiar los colores de la señal Vb(t)

Vb(t)	BOOL	Interruptor para la visualización de la señal Vb(t)
Vc(t)	U32	Caja framed color box de color enmarcada con la que podemos cambiar los colores de la señal Vc(t)
Vc(t)	BOOL	Interruptor para la visualización de la señal Vc(t)

Tabla 4.5. Variables del sistema

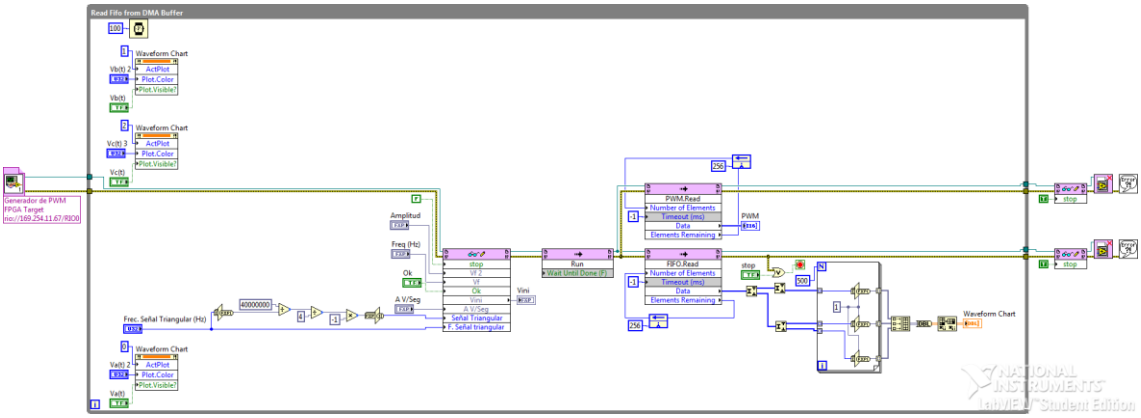


Fig. 4.18. Diagrama de bloques monitorizacion.vi

4.3.6. Generador de señales trifásicas a partir del Space Vector Modulation desarrollado en lenguaje de programación gráfica G.

4.3.6.1. Código.vi

En este programa se ha implementado un generador de señales trifásicas a través de la tecnología del vector espacial mediante las matemáticas que se han explicado en el marco teórico, consiguiendo automatizar este sistema. Este código se ejecutará dentro de la FPGA.

El programa consistirá en la introducción por parte del usuario de unas variables (frecuencia, amplitud, la tensión de la señal en continua, y la tensión de pico de la triangular) y a través de estas junto con las matemáticas de alto rendimiento, generar tres señales de control (Vca, Vcb y Vcc).

Nombre	Tipo	Descripción
Frecuencia	FXP	Variable de control de la frecuencia.
Amplitud	FXP	Variable de control de la amplitud.
VDC	FXP	Variable de control de la tensión en continua.
Vpt	FXP	Variable de control del voltaje de pico de la señal triangular.
Vca	FXP	Indicador de la señal de control Vca
Vcb	FXP	Indicador de la señal de control Vca
Vcc	FXP	Indicador de la señal de control Vca

Tabla 4.6. Variables del sistema

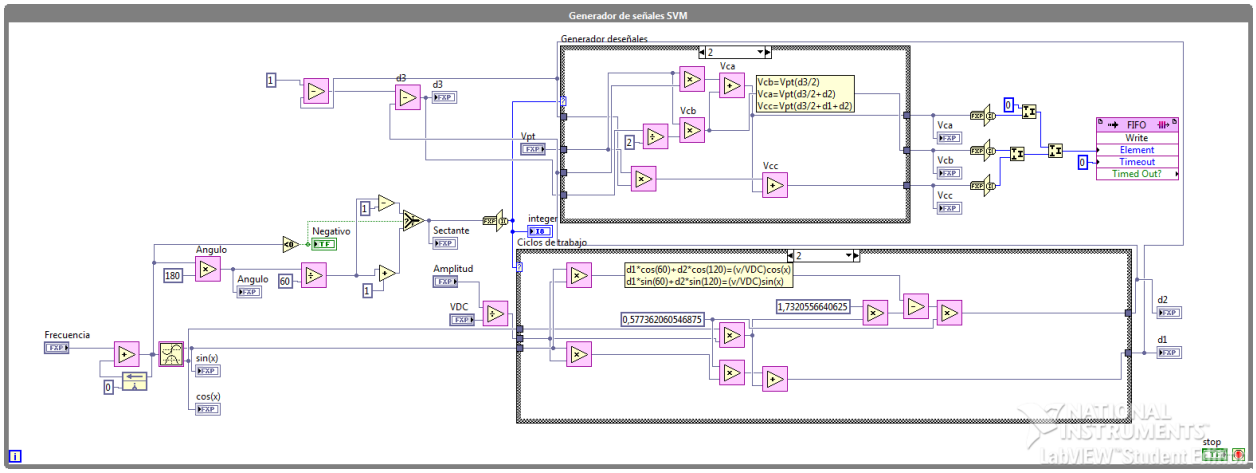


Fig. 4.19. Diagrama de bloques codigo.vi

4.3.6.2. Código de monitorización. Vi

Este programa se compilará en el ordenador. Este consistirá en leer y en escribir sobre las variables pudiéndose llevar un control en tiempo real de las variables implementadas en el programa que se ha compilado en la FPGA.

Las variables que se controlarán se indicaran en la Tabla 4.7.

Nombre	Tipo	Descripción
Frecuencia	FXP	Variable de control de la frecuencia.
Amplitud	FXP	Variable de control de la amplitud.
Vpt	FXP	Variable de control del voltaje de pico de la señal triangular.
VDC	FXP	Variable de control de la tensión en continua.

Sectante	FXP	Indicador que muestra en que sextante se encuentra el vector espacial.
Angulo	FXP	Indicador que muestra el ángulo que forma el vector espacial.
Negativo	BOOL	Indicador que muestra si el ángulo que forma el vector espacial es negativo.

Tabla 4.7. Variables del sistema.

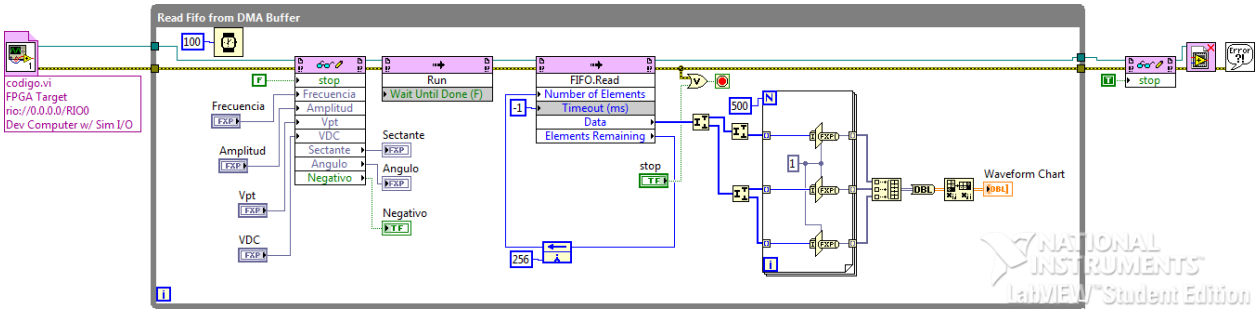


Fig. 4.20. Diagrama de bloques monitorizacion.vi

4.3.7. Programa de control desarrollado en lenguaje de programación G con la librerías de la FPGA.

4.3.7.1. Código.vi

A partir de las librerías que se disponen en LabVIEW como es la de "motion" se ha utilizado uno de sus bloques para generar una señal trifásica a partir de un vector espacial. Este programa consistirá en lo mismo que en el Generador de señales trifásicas a partir del Space Vector Modulation desarrollado en lenguaje de programación gráfica G., salvo que aquí se han utilizado las librerías que vienen de serie en LabVIEW. Como se puede apreciar, este código es la versión reducida del que se ha implementado anteriormente.

Nombre	Tipo	Descripción
Amplitud	FXP	Variable de control de la amplitud.
Freq (Hz)	FXP	Variable de control de la frecuencia.

Tabla 4.8. Variables del sistema

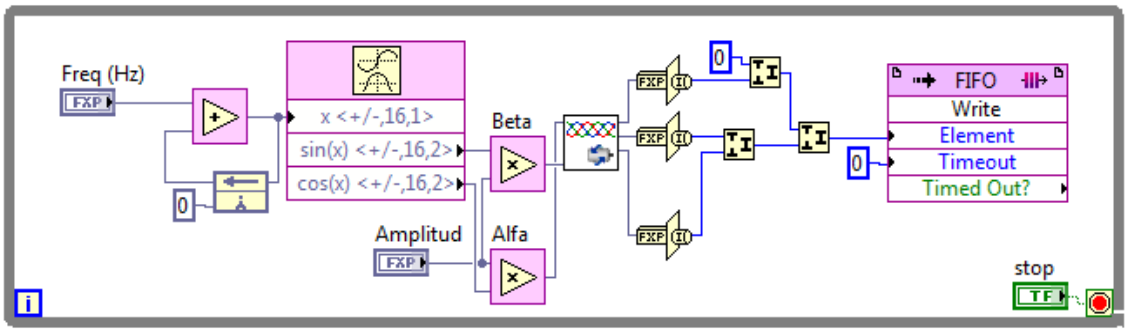


Fig. 4.21. Diagrama de bloques codigo.vi

4.3.8. Generador de SVPWM desarrollado en lenguaje de programación gráfica G.

4.3.8.1. Código

Este programa es fruto de la unificación de los programas: generar señales trifásicas por medio de la tecnología del Space Vector Modulation, el generador de la señal triangular y del generador de las rampas de amplitud y frecuencia.

Este consistirá en ajustar una tensión de continua para el generador (VDC) y en introducir una frecuencia, una amplitud y una frecuencia de la señal triangular. La frecuencia de la señal triangular nos delimitará los valores de pico de esta.

Una vez introducidos estos valores el programa no comenzará a funcionar hasta que el usuario no pulse el botón de "Ok", una vez hecho esto se activará el generador de rampa de amplitud y frecuencia que se generarán a una frecuencia de 50Hz, los datos que se generan serán

mandados al generador de señales trifásicas. Es ahí, en el generador de señales trifásicas, por medio de la tecnología del Space Vector Modulation donde se generan tres señales de control (Vca, Vcb y Vcc) que serán enviadas al generador de la señal triangular. Cabe destacar que este bucle se generará a la misma frecuencia que se genere la señal triangular.

Cuando los datos llegan a la señal triangular estos se comparan con esta, mandando la señal originada por la comparación por tres de las salidas de los pines Half Bridge.

A continuación, mostraremos la Tabla 4.9 con las diferentes variables que intervienen en nuestro sistema.

Parte	Nombre	Tipo	Descripción
Generador de señales SVM	F. Señal triangular	U32	Variable de control en la que podemos introducir la frecuencia de la señal triangular.
	VDC	FXP	Variable de control de la tensión continua.
	Vca	I16	Indicador para visualizar la señal de control Vca
	Vcb	I16	Indicador para visualizar la señal la señal de control Vcb.
	Vcc	I16	Indicador para visualizar la señal de control Vcc(t).
	Stop	BOOL	Controla la parada de ejecución de los

			bucles While Loop.
Generador de la señal triangular	Señal Triangular	I16	Variable de control de la amplitud de la señal triangular.
	Triangle wave	I16	Indicador de la señal triangular.
Generador de rampa frecuencia y amplitud	Vini	FXP	Variable de control inicial de la frecuencia.
	A V/Seg	FXP	Variable de control con la que configuraremos la velocidad con la que se genere la rampa.
	Vini 2	FXP	Variable de control inicial de la amplitud.
	Vf	FXP	Variable de control final de la frecuencia.
	Vf2	FXP	Variable de control final de la amplitud.
	Amplitud	FXP	Indicador de la amplitud.
	Freq (Hz)	FXP	Indicador de la frecuencia.
	Ok	BOOL	Pulsador de arranque de la rampa.

Tabla 4.9. Variables del sistema.

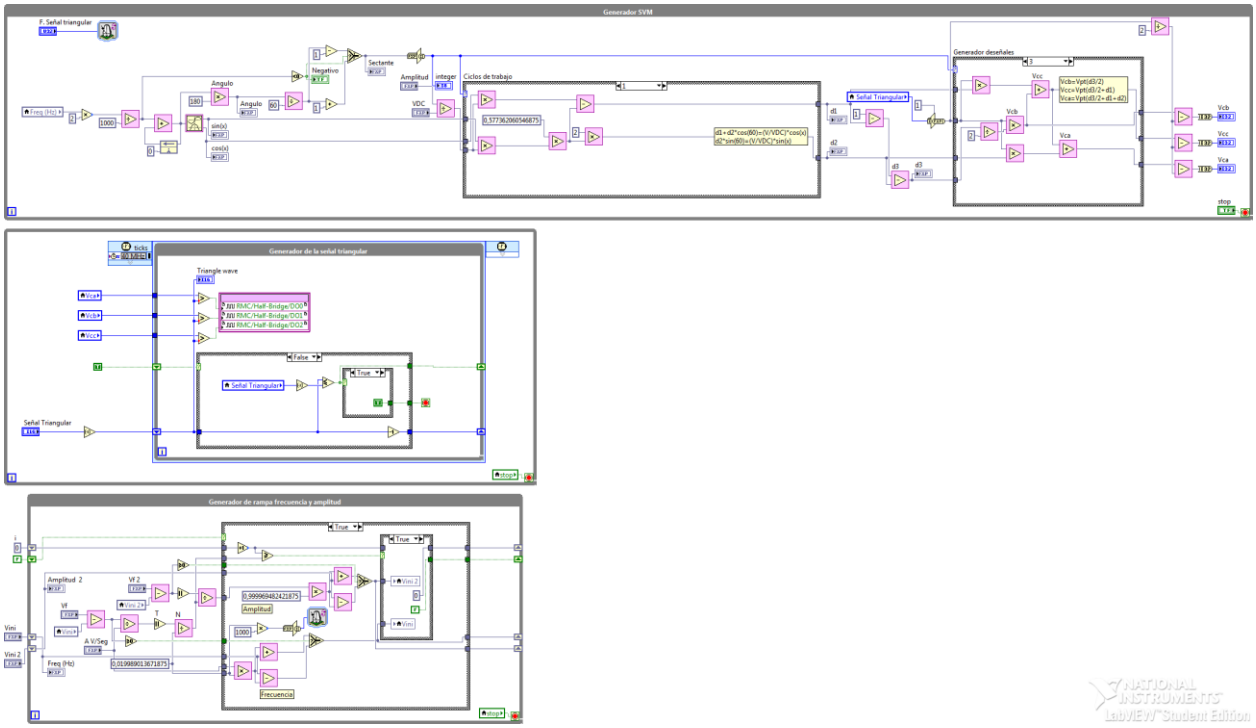


Fig. 4.22. Diagrama de bloques codigo.vi

4.3.8.2. Monitorización

Este programa servirá para controlar desde el ordenador las variables anteriormente implementadas, estas se analizan en la Tabla 4.10.

Nombre	Tipo	Descripción
Frecuencia	FXP	Variable de control de la frecuencia.
Amplitud	FXP	Variable de control de la amplitud.
VDC	FXP	Variable de control de la tensión en continua.
Frec. Señal Triangular (Hz)	U32	Variable de control en la que podemos introducir la frecuencia de la señal triangular.
A V/Seg	FXP	Variable de control con la que configuraremos la velocidad con la que se genere la rampa.
Vca	I32	Indicador para visualizar la señal de control Vca
Vcb	I32	Indicador para visualizar la señal de control Vcb
Vcc	I32	Indicador para visualizar la señal de control Vcc
Ok	BOOL	Pulsador de arranque de la rampa.

Tabla 4.10. Variables del sistema.

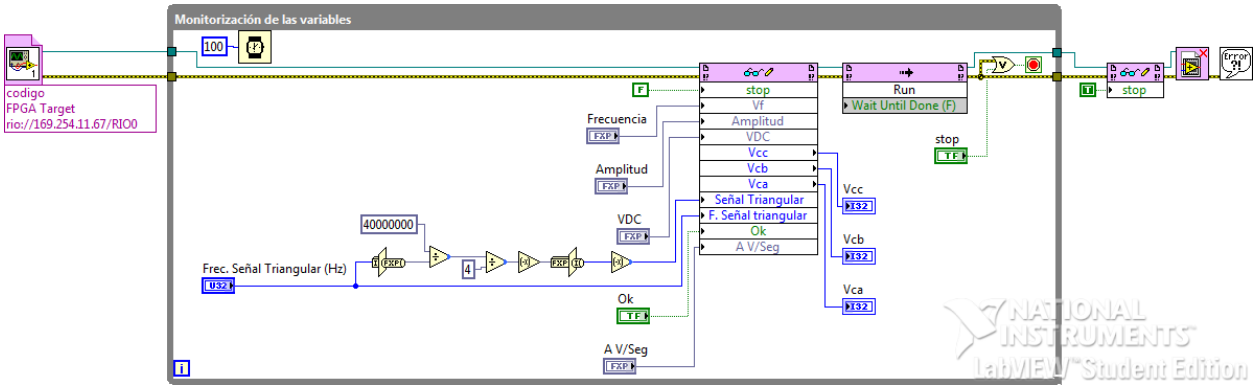


Fig. 4.23. Diagrama de bloques monitorizacion.vi

4.4. Cuarta fase: Diseño de placa de conexiones

A continuación se diseñará una placa de conexiones que sirva de nexo de unión entre la Single-Board RIO y los aparatos de potencia de los que dispone el laboratorio de potencia de la Universidad Politécnica de Cartagena, esta será la mostrada en la Fig. 4.24.

Una vez diseñada la placa de conexiones se explicará que se conecta en cada puerto de la misma. El bloque "Half-Bridge" se conectará con su homólogo en la Single-Board RIO, los pines del 4 al 10 y del 15 al 25 irán conectados a tierra (GND), los pines 20, 24 y 26 irán conectados a los puertos de salida X3-1, X3-2 y X3-1, consecutivamente. Por estos puertos se obtendrán los disparos del PWM. Finalmente, se conectará el pin 2 a VCC.

Por último, se han colocado dos regletas de salida de 2 salidas cada una. La primera (X1-1 y X1-2) servirá para conectar la placa a una fuente de alimentación en el que se conectará X1-1 al positivo y X1-2 al negativo. La segunda regleta servirá para alimentar la placa del PWM y se conectará de la misma forma el X2-1 a VCC y X2-2 al GND.

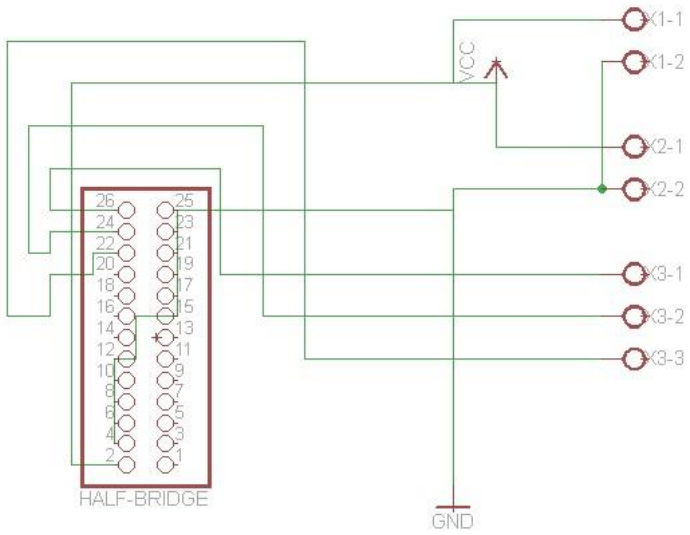


Fig. 4.24. Esquemático de la placa de conexiones.

Para finalizar se presentará un diagrama de cómo quedaría todo el sistema conectado que se mostrará en la Fig. 4.25.

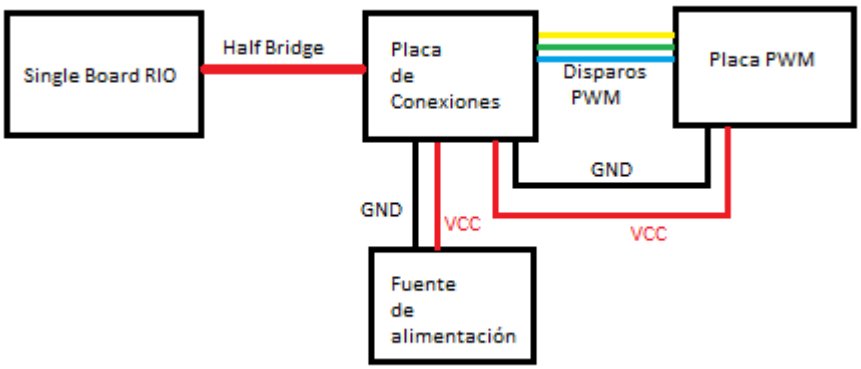


Fig. 4.25. Esquema de conexiones de nuestro sistema.

Capítulo 5

5. Experimentación y resultados

En este capítulo expondremos los resultados obtenidos a través de los diferentes experimentos realizados.

5.1. Generar una señal trifásica

En este experimento se ha conseguido generar una señal trifásica a través del código que se ha implementado en los distintos programas que han sido explicados anteriormente. Dado que con todos los programas que se han implementado, se han obtenido los mismos resultados, sólo se comenta uno de ellos.

A continuación se mostrarán el valor que se le ha dado a las distintas variables en la Tabla 5.1 y se mostrará en la Fig. 5.1 el resultado obtenido.

Variables	Valores
Amplitud	1
Freq (Hz)	1

Tabla 5.1. Variables prefijadas para el desarrollo del experimento.

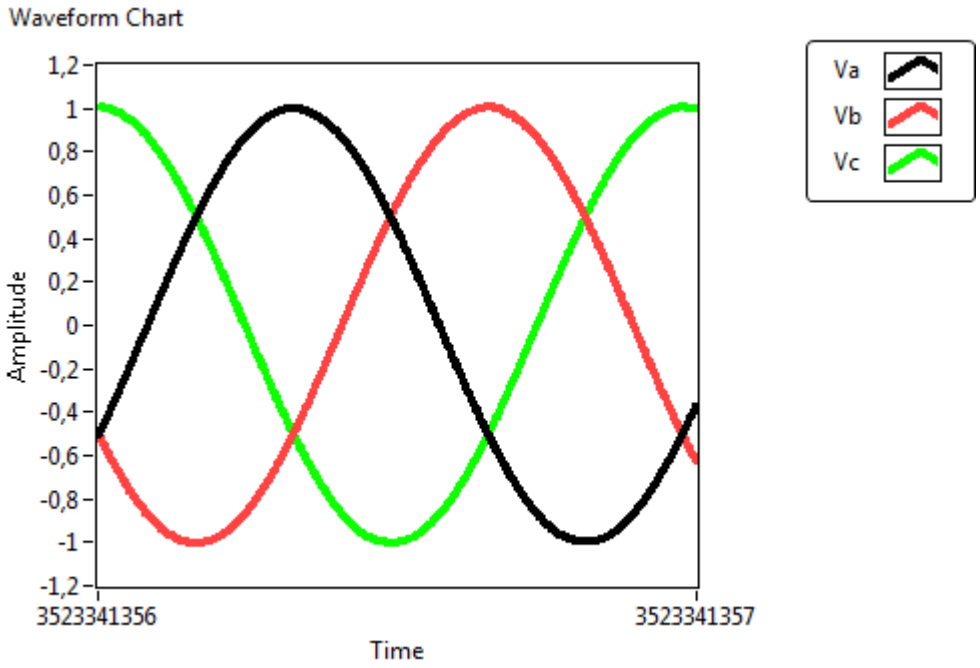


Fig. 5.1. Señal trifásica mostrada por el waveform chart.

5.2. Generar una rampa de arranque

5.2.1. Primer experimento.

Este experimento consistirá en probar el código implementado para generar una rampa de arranque tanto para la amplitud como para la frecuencia. La amplitud se introduce en un rango de [0,1], ya que esta vendrá determinada por la amplitud de la señal triangular en el PWM y se configura así desde un principio.

En la Fig. 5.2 se representa el mando de control con los datos que se han utilizado en el experimento. Obteniéndose como resultado final la Fig. 5.3.

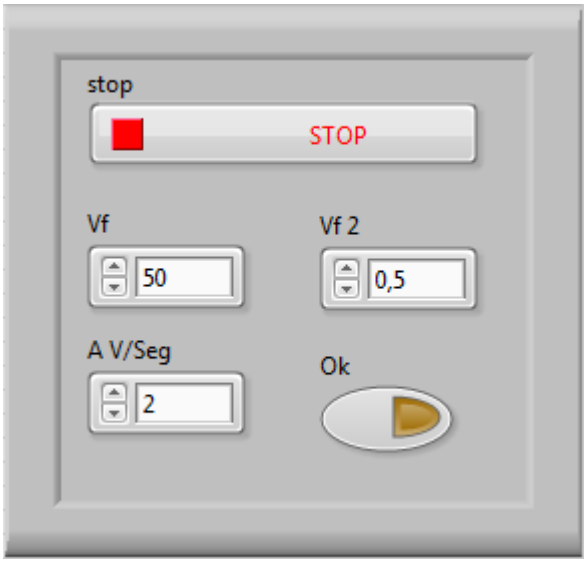


Fig. 5.2. Cuadro de control de las variables.

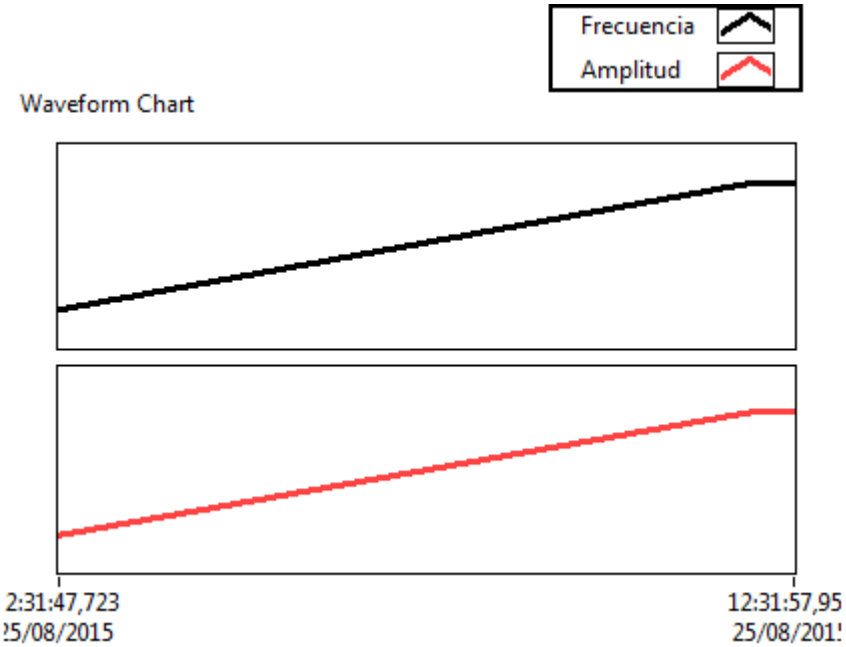


Fig. 5.3. Rampa generada mostrada por el waveform chart.

5.2.2. Segundo experimento.

En este experimento se visualizará la rampa de arranque del programa Generador de SVPWM desarrollado en lenguaje de programación gráfica G. y utilizando los datos que se muestran en la Tabla 5.4 se obtienen los resultados de la Fig. 5.4.

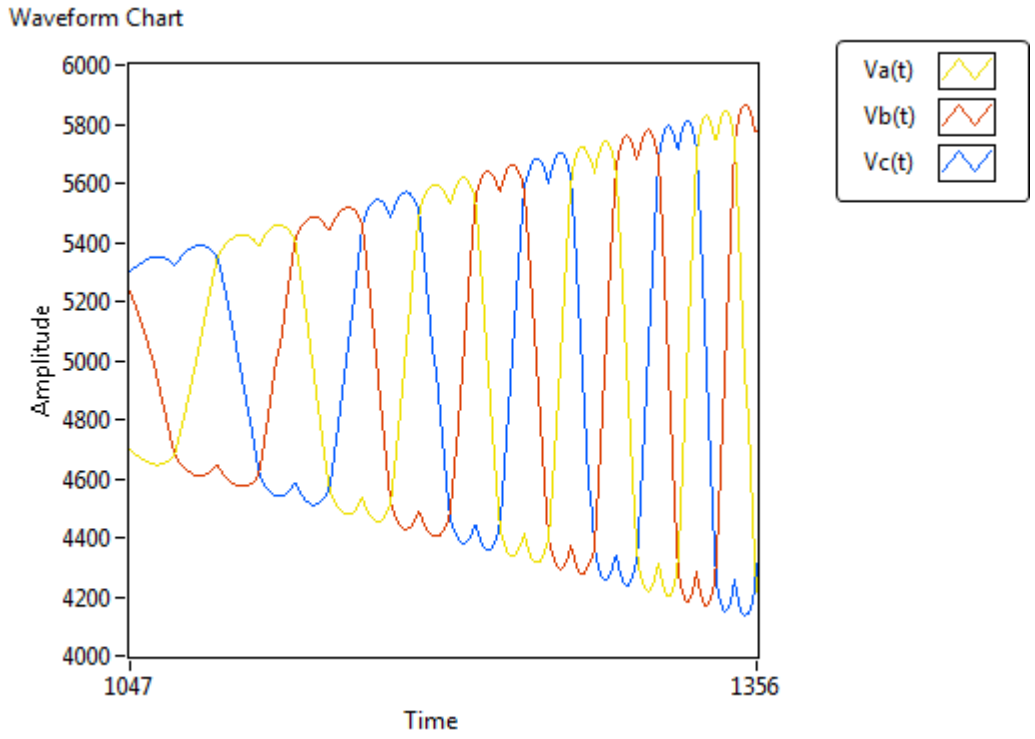


Fig. 5.4. Rampa de arranque en un generador trifásico.

5.3. Generar una señal triangular

En esta prueba se visualizará una señal triangular con una amplitud de 20.000. Utilizando el reloj de la FPGA que es de 40MHz, se obtiene que la frecuencia de la señal triangular será de 1KHz. No se visualiza bien por la alta frecuencia a la que va ejecutando cada iteración, ya que es del orden de los 40MHz.

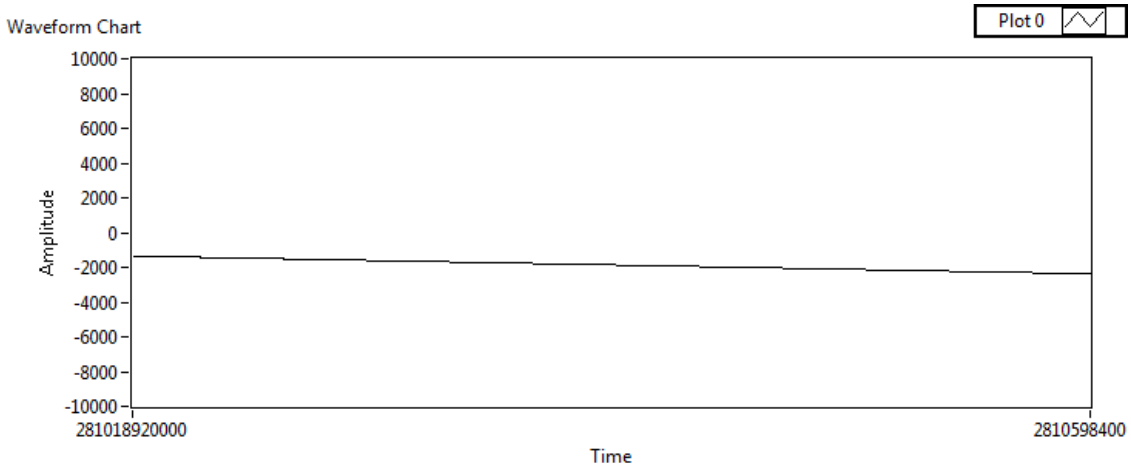


Fig. 5.5. Señal triangular mostrada por el waveform chart.

5.4. Generar un PWM por la comparación de una señal senoidal y una triangular

5.4.1. Primer experimento

En este experimento se ha simulado el Programa de control de la velocidad de un motor asíncrono a través de un PWM desarrollado en lenguaje de programación G. y se ha utilizado un osciloscopio para poder visualizar una de las señales. Este experimento no fue válido, debido a que se encontraron unos errores en la precisión de las operaciones, aunque se pudo visualizar claramente la resolución a la que podía llegar nuestro ancho de pulso a una frecuencia de 1KHz como se muestra en la Fig. 5.6. Cabe destacar que este primer programa en el que se hizo la simulación, la frecuencia de la señal triangular era constante y el usuario no podía modificarla.

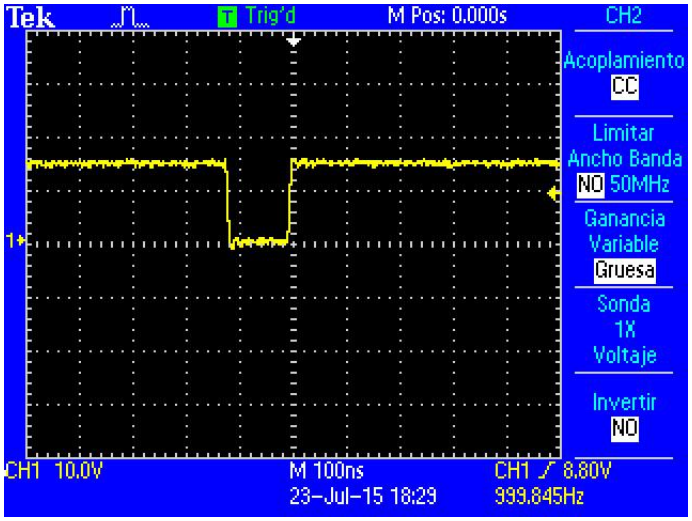


Fig. 5.6. Ancho de pulso observado a 100ns.

5.4.2. Segundo experimento

En este segundo experimento se corrigieron los problemas de resolución que se habían producido en el Primer experimento a la hora de generar la modulación por ancho de pulso. En la Fig. 5.7 se puede apreciar correctamente la señal de PWM generada a una frecuencia de 1KHz.

En la Tabla 5.2 se muestran los valores prefijados para las distintas variables que forman el programa de control.

Variables	Valores
Amplitud	0,5
F. Señal triangular (Hz)	1000
Freq (Hz)	50
A V/Seg	1

Tabla 5.2. Variables prefijadas para el desarrollo del experimento.

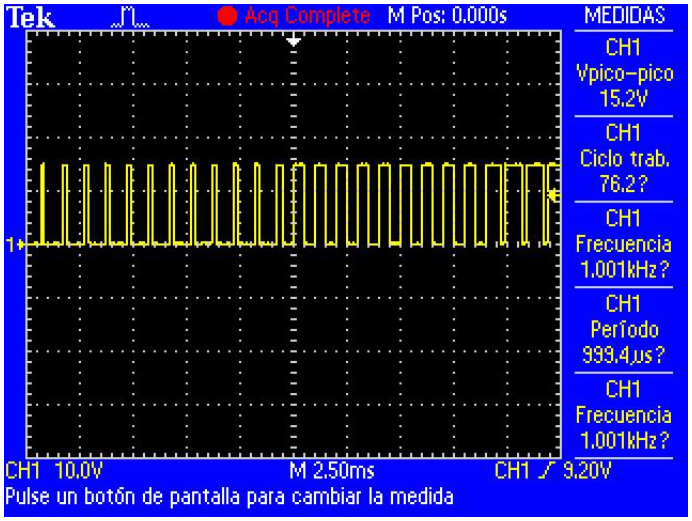


Fig. 5.7. PWM generado.

5.4.3. Tercer experimento (Inversor modulado)

Este experimento consistió en usar el PWM, generado en el Segundo experimento, para disparar los transistores de un inversor monofásico en puente completo modulado como el de la Fig. 5.8 generando una señal senoidal como la de la Fig. 5.9. Con este programa se verifica que se puede generar una señal senoidal con la frecuencia y la amplitud deseada.

Para este experimento se utilizarán los valores de las distintas variables que se muestran en la Tabla 5.2.

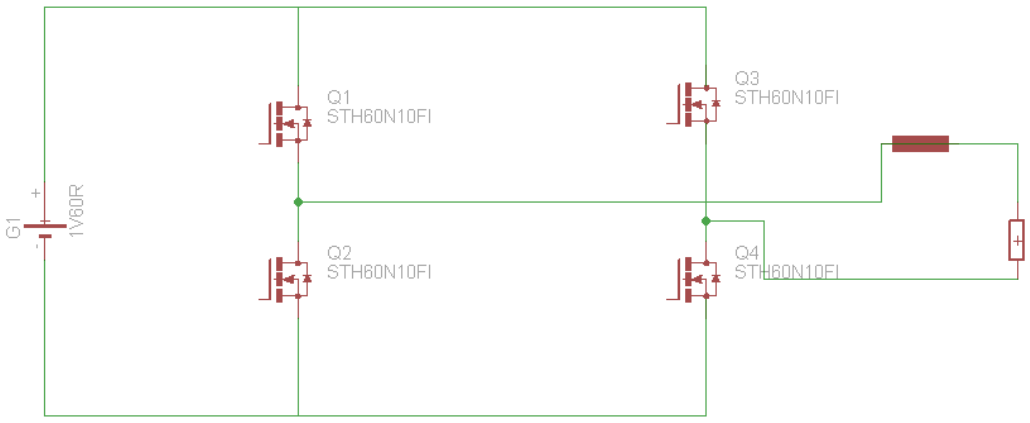


Fig. 5.8. Inversor modulado monofásico en puente completo

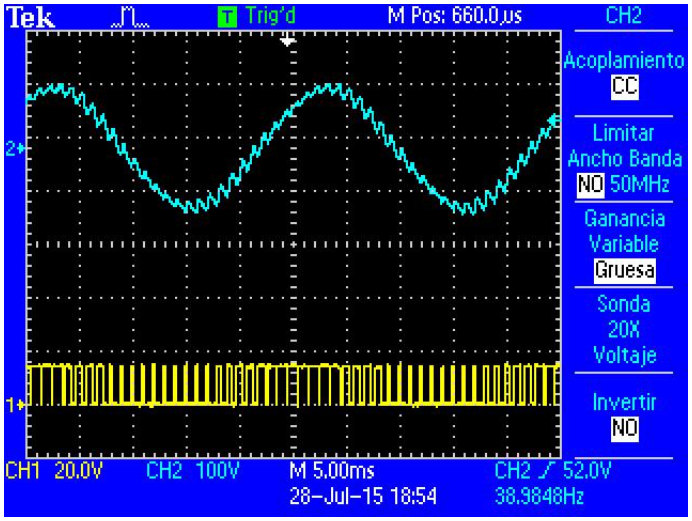


Fig. 5.9. Onda senoidal generada a la salida del inversor modulado.

Para este experimento se han utilizado los equipos del Laboratorio de electrónica de potencia utilizando: cubo de potencia (Inversor modulado), PCB[29] PWM, fuente de alimentación, sonda de tensión, y otros elementos auxiliares.

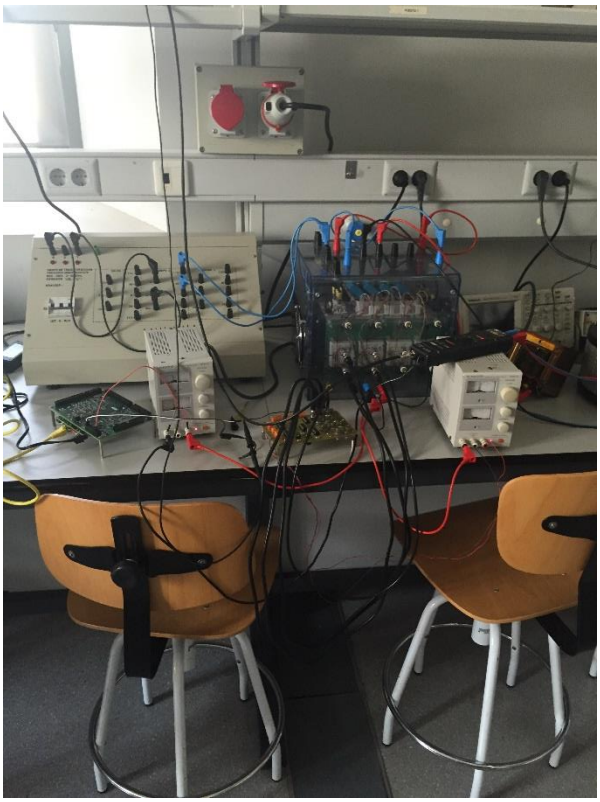


Fig. 5.10. Montaje de los equipos utilizados en el laboratorio de potencia.

[29] PCB: Printed Circuit Board

5.5. Generar una señal trifásica con el Space Vector Modulation

Este experimento es parecido al realizado anteriormente en el que se generaba una señal trifásica mediante el generador de señales. La diferencia radicará en que la señal trifásica generada ahora será realizada mediante una técnica diferente como es la del vector espacial modulado. Con dicho experimento se ha validado que se obtienen los mismos resultados, tanto por el código que se ha implementado como el que ha sido implementado usando las librerías de LabVIEW. Por los que a continuación se comentan los datos obtenidos de un solo experimento para evitar redundancia.

Este experimento se llevará a cabo con los datos de la Tabla 5.3.

Variable	Valores
Amplitud	0,599945
Freq(Hz)	1

Tabla 5.3. Variables prefijadas para el desarrollo del experimento

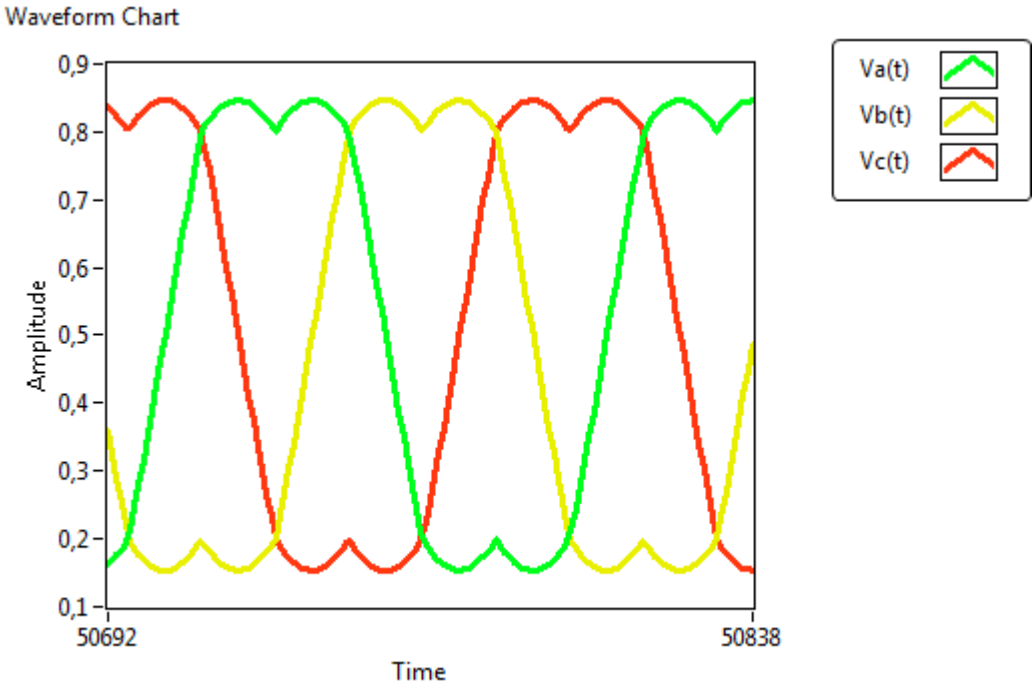


Fig. 5.11.Señal trifásica mostrada por el waveform chart.

5.6. Generar un PWM a través del SVPWM

.

5.6.1. Primer experimento.

Este experimento consistirá en ejecutar el Generador de SVPWM desarrollado en lenguaje de programación gráfica G. el cual ha sido explicado su funcionamiento en el capítulo 4.

Los valores de las variables que se han utilizado para desarrollar este experimento se representarán en la Tabla 5.4.

Variable	Valores
Amplitud	50
VDC	100
Frecuencia de la señal triangular	1 KHz
Frecuencia	50Hz
Señal de la triangular	10000

Tabla 5.4. Variables prefijadas para el desarrollo del experimento

La salida digital AO0 del Half-Bridge se conectará al canal uno del osciloscopio pudiéndose visualizar el PWM generado a partir de las variables de la Tabla 5.4 obteniéndose la Fig. 5.12.

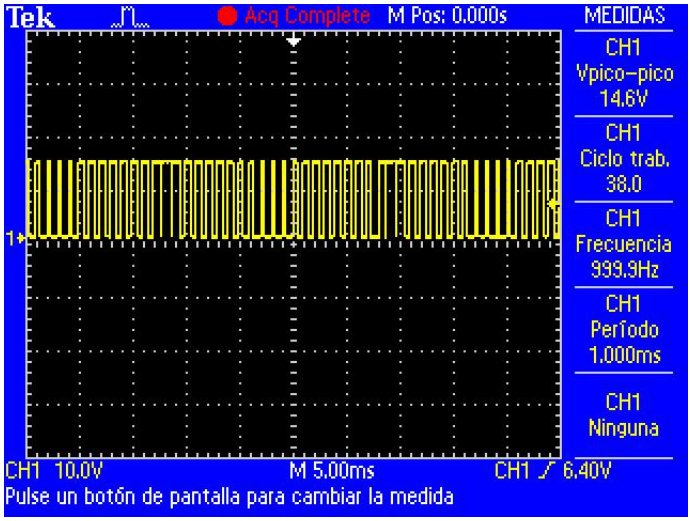


Fig. 5.12. PWM generado a partir del programa SVPWM

5.6.2. Segundo experimento (Inversor modulado).

Este se basará en utilizar la señal de PWM generada en el Primer experimento. para disparar los transistores de un inversor modulado monofásico de medio puente como el de la Fig. 5.13. En este circuito se medirá la intensidad, con una sonda de corriente, que circula por la resistencia pudiéndose observar la señal senoidal de la Fig. 5.14 por el canal 2 del osciloscopio a la vez que se observa por el canal 1 el PWM que se le mandan a los transistores del inversor modulado monofásico de medio puente.

Los datos que han sido utilizados a la hora de realizar el experimento son los de la Tabla 5.4.

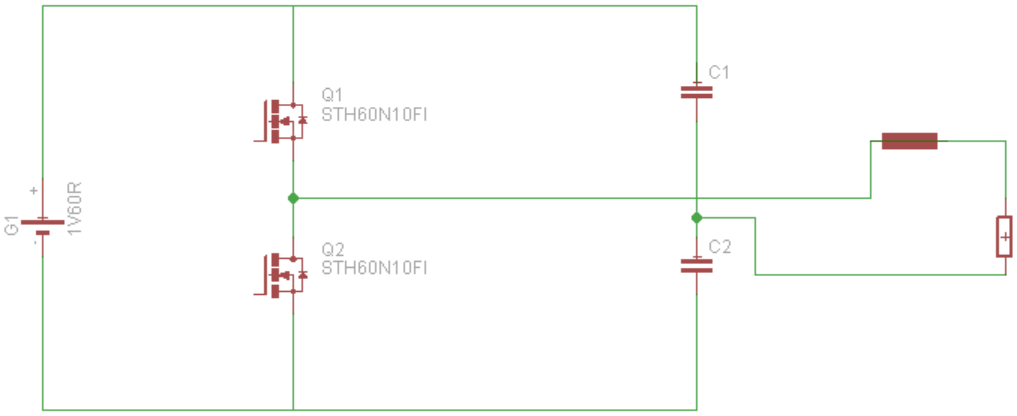


Fig. 5.13. Inversor modulado monofásico de medio puente.

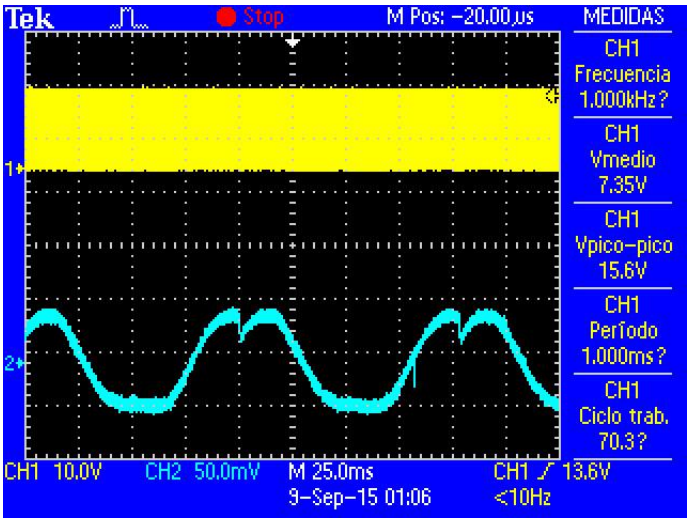


Fig. 5.14. Señal senoidal obtenida a través del SVPWM.

Capítulo 6

6. Presupuesto

	Concepto	Unidades	Precio unidad (€)	Precio Total (€)
Hardware	Single-Board RIO	1	921	921
	Mezzanine card NI9683			
Software	Software LabVIEW	1	0	0
Personal	Mano de obra	30	10	300
PCB	W237-102	2	0,95	1,9
	W237-102	1	0,95	0,95
	PINHD-2X13	1	3,59	3,59
	placa	1	10,46	10,46
	cable plano	1	4,95	4,95
	Total			1242,85

Capítulo 7

7. Conclusiones y trabajos futuros

A lo largo del desarrollo de este Trabajo Final de Grado se han podido sacar varias conclusiones:

- La utilización de flujogramas a la hora de implementar un programa es vital. Con su uso se mejoran los tiempos de desarrollo. En el momento de programar hay que tener las ideas muy claras de que se quiere hacer, de cuáles son nuestros datos de entradas y de cuáles serán nuestros datos de salida.
- Cuando se trabaja con datos de punto fijo hay que tener muy claro cuál será el ancho de palabra y cuántos bits tendrá la parte entera. Cuando se asigna un número de bits a la parte entera y otro a la parte decimal, hay que tener muy presente cuando se pueden producir desbordamientos.
- El gran potencial de los productos de National Instrumets. A nivel de hardware, el controlador de automatización programable Single-Board RIO ofrece un sistema reconfigurable de control y adquisición de bajo coste diseñado para aplicaciones que requieren alto rendimiento y fiabilidad. Y a nivel de software, el lenguaje G de programación utilizado en el entorno de programación gráfica LabVIEW, el cual ofrece una integración incomparable con miles de dispositivos de hardware y brinda cientos de bibliotecas integradas para análisis avanzado y visualización de datos, para crear instrumentación virtual.
- La incorporación de variables indicadores distribuidas tanto en el programa de código.vi y de monitorización.vi. Estas variables permiten visualizar en tiempo real las señales a través de una interfaz gráfica, de esta manera se consigue un mayor control de sistema implementado y una mayor ayuda debido a que va corrigiendo los errores que han ido surgiendo en la realización de dicho trabajo.
- Los resultados obtenidos en las pruebas realizadas han sido satisfactorios en todos los aspectos, pudiéndose comprobar de manera práctica las buenas propiedades de la

tecnología Single-Board RIO y LabVIEW, de tal forma que se pueden controlar todos los aspectos del entorno desde un único dispositivo sin tener que añadir electrónica física adicional. Este efecto se nota en la tecnología embebida usada que permite trabajar de una manera robusta y rápida, como se puede observar en el capítulo 5.

Este trabajo se podría completar añadiendo:

- Un regulador de PID[30] para llevar un control del sistema en lazo cerrado utilizando algunos sistemas de control. Esta implementación puede ayudar al controlador en la toma de decisiones haciéndolo más preciso y eficiente.
- Se pueden incorporar métodos de seguridad ante posibles errores fatales.
- Se puede modificar la programación para que solo se tenga que programar una única señal PWM y desfasarla a través de la electrónica física, en vez de las tres señales diseñadas. Este cambio podría compactar y simplificar el programa.
- La utilización de sensores Hall o de un encoder para medir la velocidad de la máquina asíncrona y conseguir una mayor precisión en el control.

[30] PID: Proporcional Integral Diferencial

Capítulo 8

8. Bibliografía

Libros

[1] Fitzgerald, A. K. (2003). *Electric Machinery* (Sexta ed.). International Edition: Mc.Graw Hill.

[2] Gottlieb, I. (1994). *Electric Motors & Control Techniques* (Segunda ed.). TAB Books.

[3] López Mesa Diana Jimena, C. M. (s.f.). *MODULACIÓN PWM APLICADA A INVERSORES TRIFÁSICOS DENTRO DEL ESQUEMA DE ACCIONAMIENTOS ELÉCTRICOS AC.*

[4] Rivera, J., Ortega, S., & Raygoza, J. J. (2009). IMPLEMENTACIÓN EN HARDWARE DE UN SVPWM EN UN SOFT-CORE NIOS II PARTE II:ALGORITMO DEL SVPWM. *e-Gnosis*, 1-8.

[5] Mohan, N. (2011). *Power Electronics.*

[6] National Instruments. (2003). *LabVIEW User Manual.*

[7] Hart, D. W. (1997). *Electrónica de Potencia* (Primera ed.). Prentice Hall.

[8] Mora, J. F. (2008). *Máquinas Eléctricas* (Sexta ed.). Mc Graw Hill.

[9] Rashid, M. H. (2004). *Electronica de potencia. Circuitos, dispositivos y aplicaciones* (Tercera ed.). Pearson Education.

Proyectos fin de carrera

[10] García Haro, Juan Miguel: '*Desarrollo de un controlador para motores DC brushless basado en CompactRIO y LabVIEW de National Instruments para el estudio de nuevos algoritmos de control*'. Proyecto fin de carrera, Universidad Carlos III de Madrid, 2011.

Páginas o documentos electrónicos en la red

[11] National Instrument. (12 de Febrero de 2015). *NI.* Obtenido de www.ni.com/white-paper/14556/es/pdf

[12] National Instruments. (s.f.). *OEM OPERATING INSTRUCTIONS AND SPECIFICATIONS NI sbRIO-9605/9606.* Manual. Obtenido de http://www.ni.com/pdf/manuals/373378a.pdf

[13] National Instruments. (s.f.). *USER GUIDE AND SPECIFICATIONS NI 9683.* Manual. Obtenido de http://www.ni.com/pdf/manuals/375960b.pdf

[14] National Instruments. (s.f.). *Aprenda LabVIEW*. Obtenido de http://www.ni.com/academic/students/learn-labview/esa/

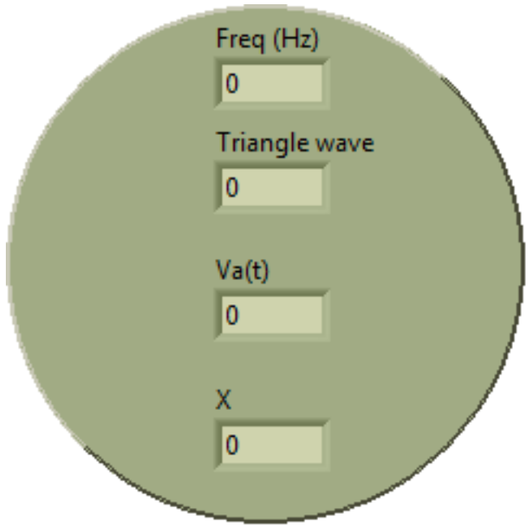
[15] National Instruments. (s.f.). *OEM OPERATING INSTRUCTIONS AND SPECIFICATIONS NI sbRIO-9605/9606*. Manual. Obtenido de http://www.ni.com/pdf/manuals/373378a.pdf

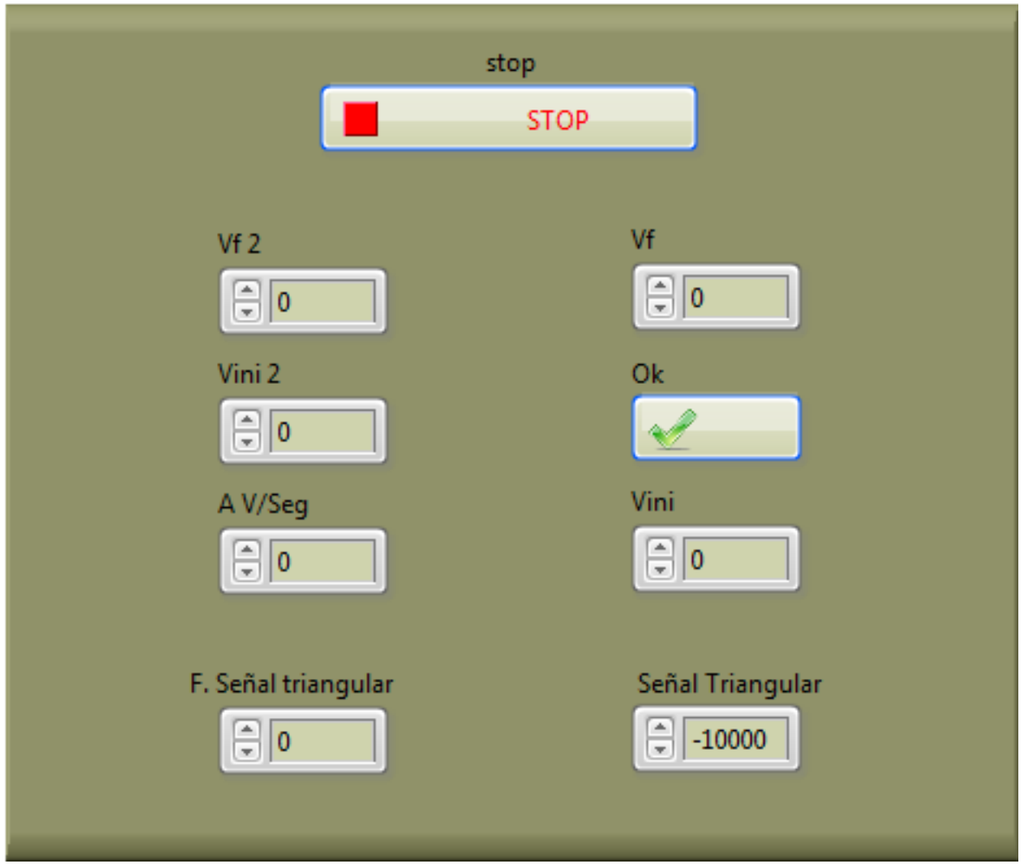
[16] (s.f.). Obtenido de http://www.ingenia-cat.com/reference/learn/TEC.PAP.0422232450.pdf

[17] (s.f.). Obtenido de http://www.uma.es/investigadores/grupos/electronicapotencia/index.php?option=com_content&view=article&id=81&Itemid=100

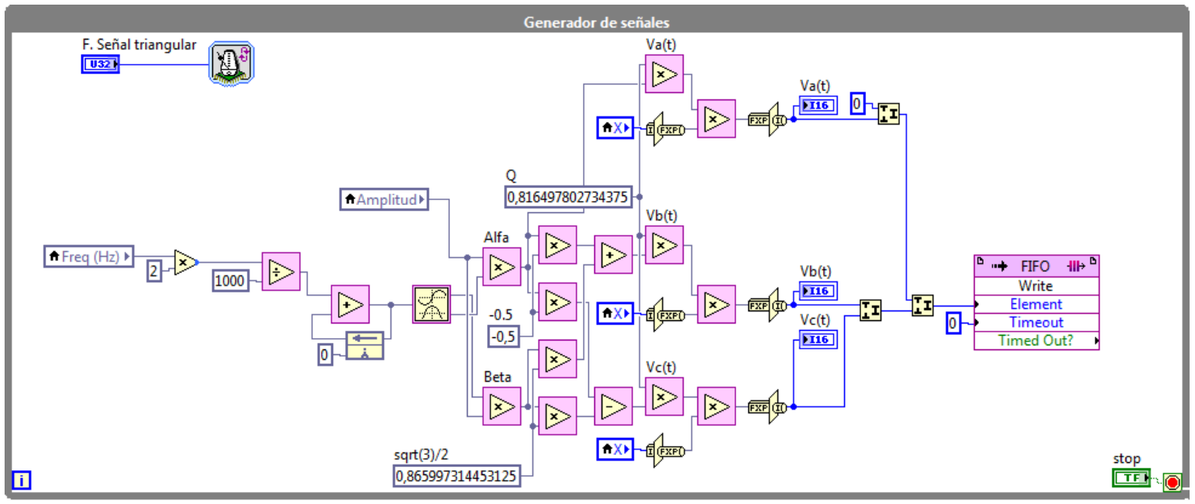
[18] (s.f.). Obtenido de http://www.uma.es/investigadores/grupos/electronica_potencia/index.php?option=com_content&view=article&id=82&Itemid=101

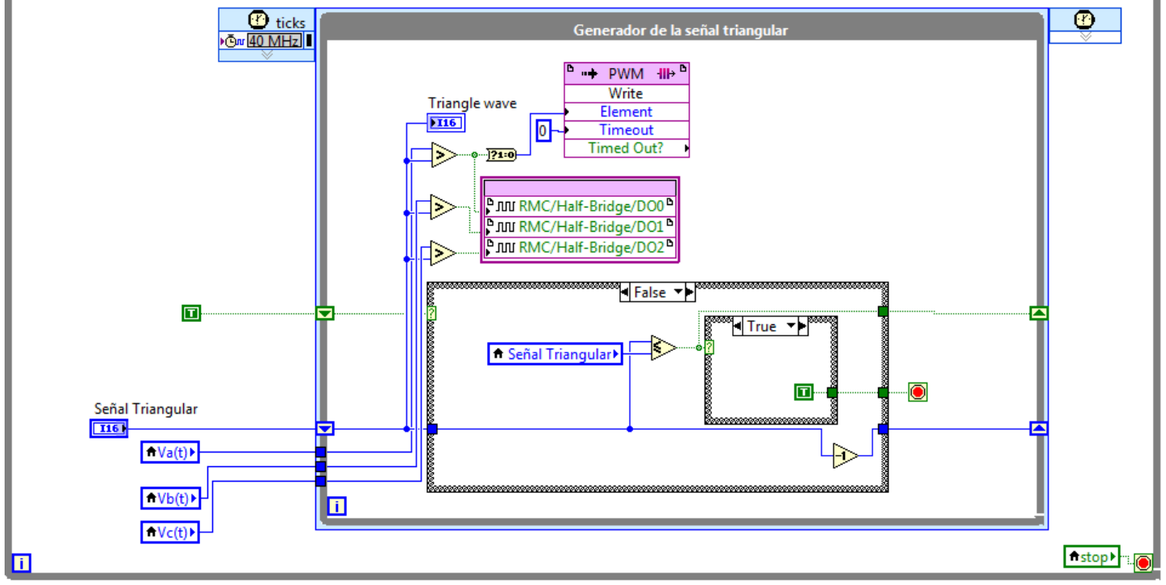
ANEXOS

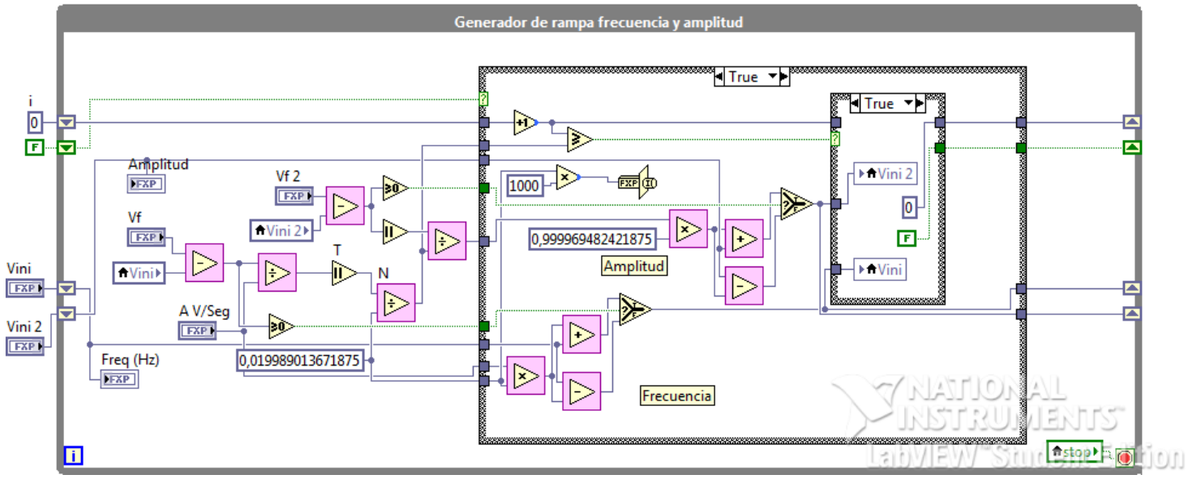




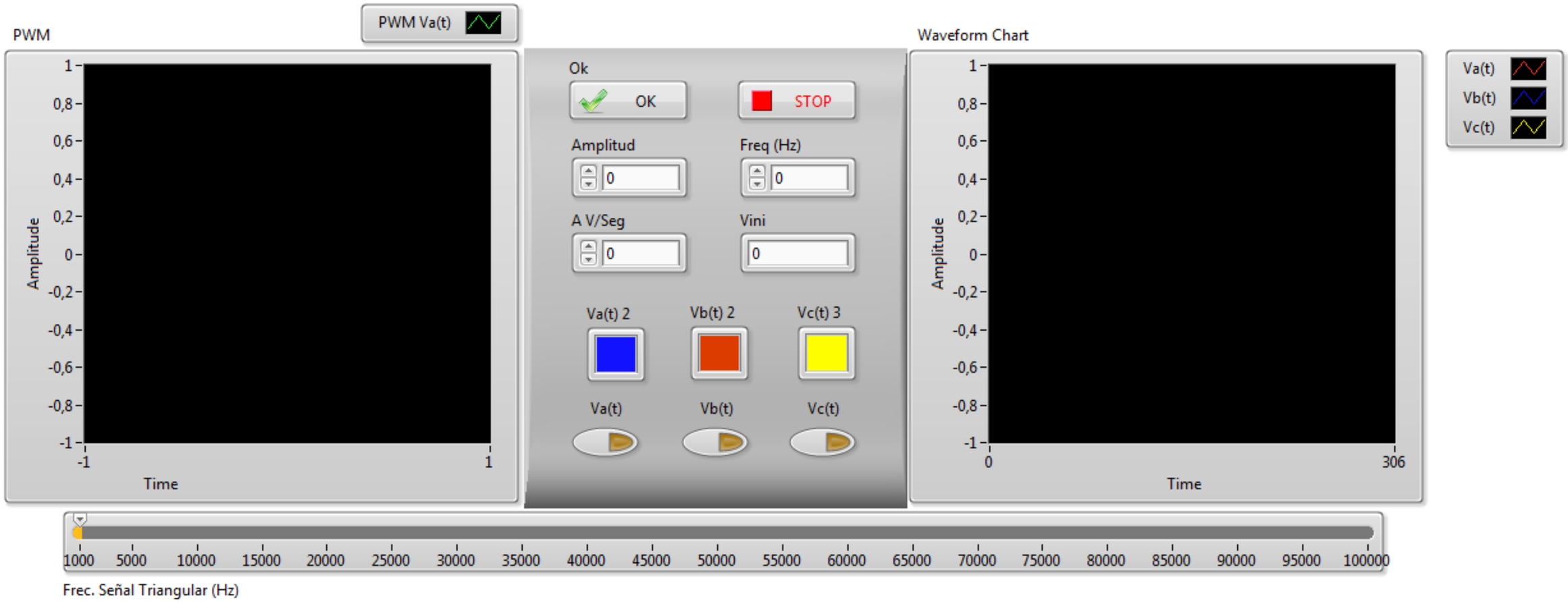
UNIVERSIDAD POLITÉCNICA DE CARTAGENA		
Proyecto:	Control escalar de velocidad de una máquina asíncrona utilizando una tarjeta sbRIO-9696	
Plano:	PWM. Interfaz visual del programa codigo.vi	Plano nº: 1
Autor:	Pablo Sánchez Sánchez	Fecha: 28/08/2015



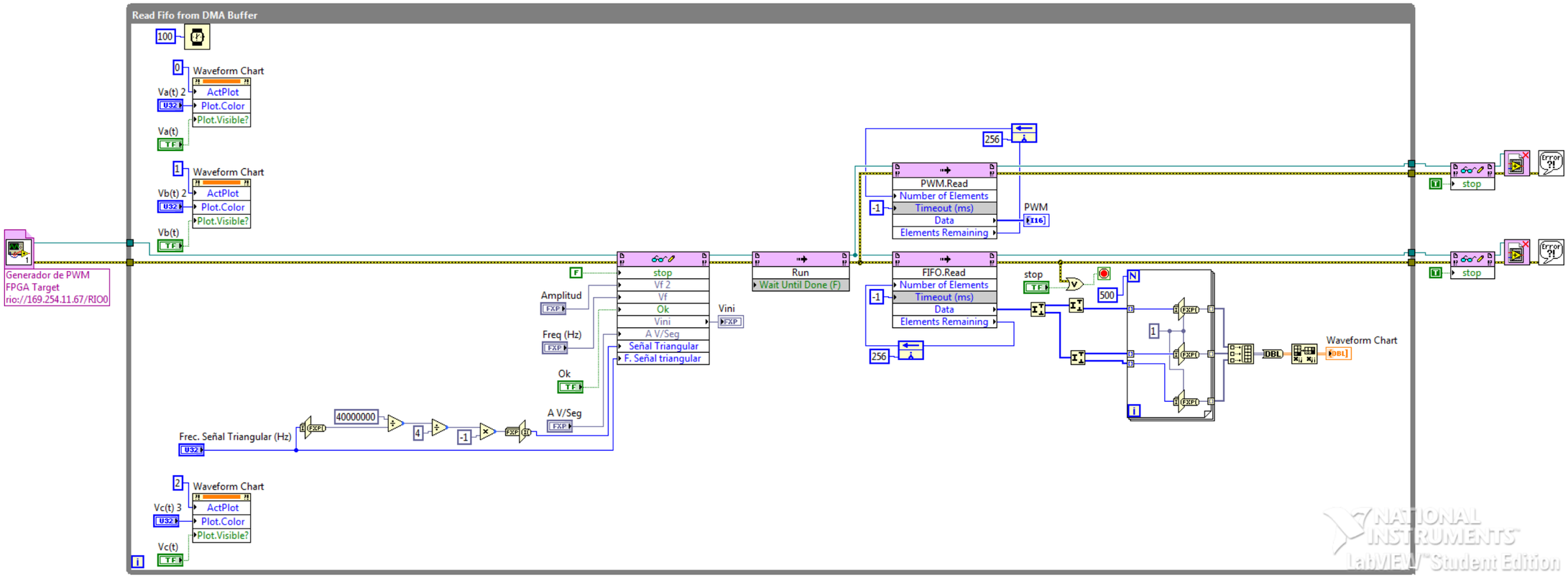




UNIVERSIDAD POLITÉCNICA DE CARTAGENA		
Proyecto:	Control escalar de velocidad de una máquina asíncrona utilizando una tarjeta sbRIO-9696	
Plano:	PWM. Diagrama de bloques del programa codigo.vi	Plano nº: 2
Autor:	Pablo Sánchez Sánchez	Fecha: 28/08/2015

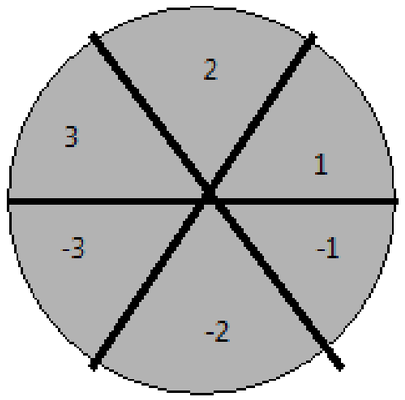


UNIVERSIDAD POLITÉCNICA DE CARTAGENA		
Proyecto:	Control escalar de velocidad de una máquina asíncrona utilizando una tarjeta sbRIO-9696	
Plano:	PWM. Interfaz visual del programa monitorizacion.vi	Plano nº: 3
Autor:	Pablo Sánchez Sánchez	Fecha: 28/08/2015



UNIVERSIDAD POLITÉCNICA DE CARTAGENA		
Proyecto:	Control escalar de velocidad de una máquina asíncrona utilizando una tarjeta sbRIO-9696	
Plano:	PWM. Diagrama de bloques del programa monitorizacion.vi	Plano nº: 4
Autor:	Pablo Sánchez Sánchez	Fecha: 28/08/2015

Sectantes



Vca 0 | Vcb 0 | Vcc 0

Amplitud 2 0 | Triangle wave 0 | Freq (Hz) 0

Sectante 0

d2 0 | d3 0

d1 0 | Angulo 0

stop

STOP

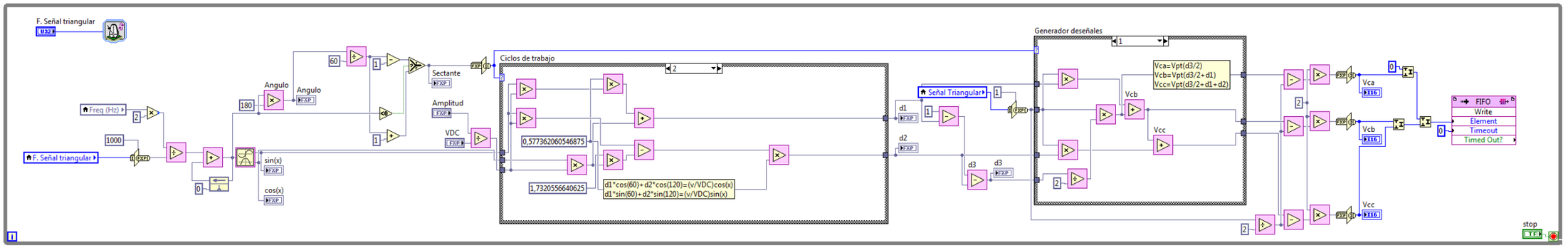
Amplitud 0 | Vf 0

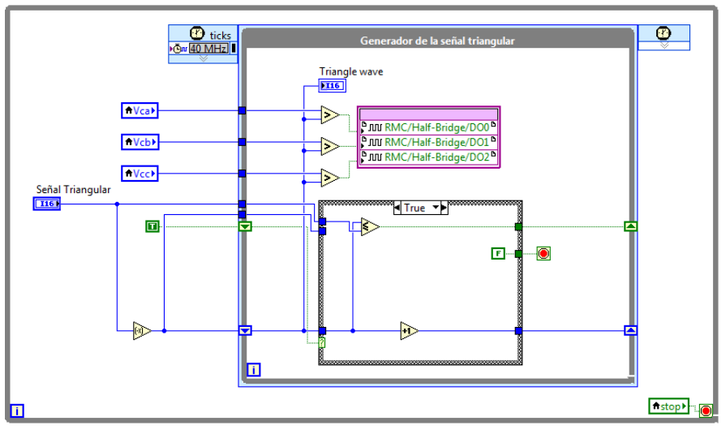
VDC 0 | Ok

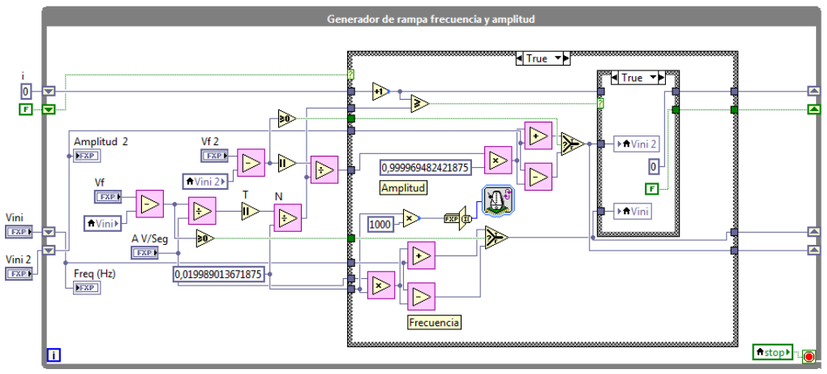
A V/Seg 0 | Vini 0

F. Señal triangular 0 | Señal Triangular -10000

UNIVERSIDAD POLITÉCNICA DE CARTAGENA		
Proyecto:	Control escalar de velocidad de una máquina asíncrona utilizando una tarjeta sbRIO-9696	
Plano:	SVPWM. Interfaz visual del programa codigo.vi	Plano nº: 5
Autor:	Pablo Sánchez Sánchez	Fecha: 28/08/2015

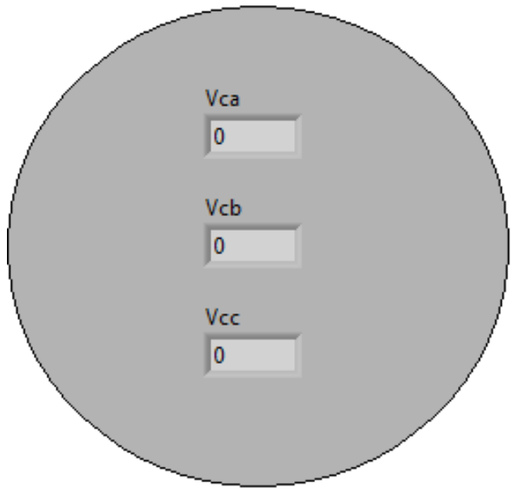


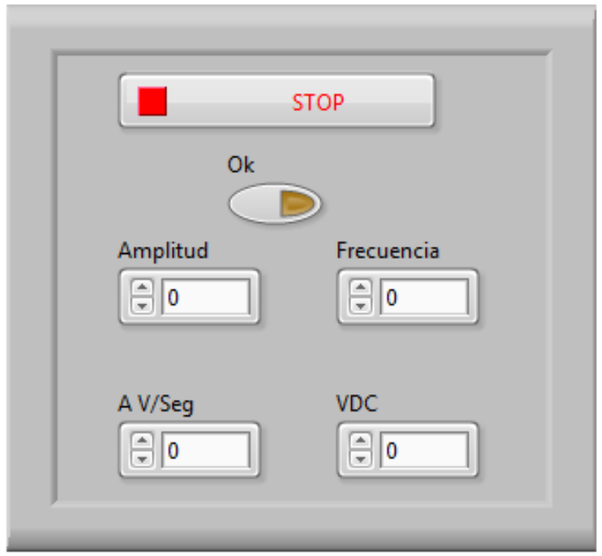


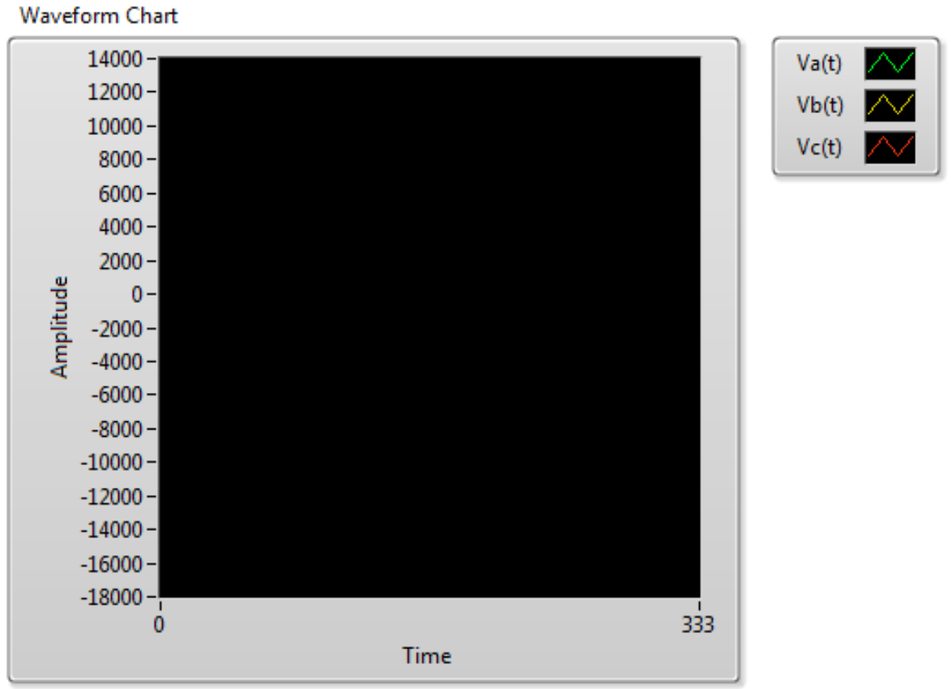


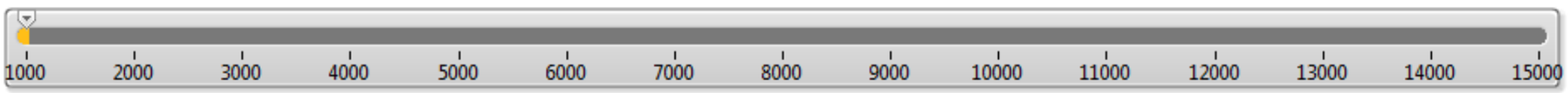
NATIONAL INSTRUMENTS
LabVIEW Student Edition

UNIVERSIDAD POLITÉCNICA DE CARTAGENA		
Proyecto:	Control escalar de velocidad de una máquina asíncrona utilizando una tarjeta sbRIO-9696	
Plano:	SVPWM. Diagrama de bloques del programa codigo.vi	Plano nº: 6
Autor: Pablo Sánchez Sánchez		Fecha: 28/08/2015



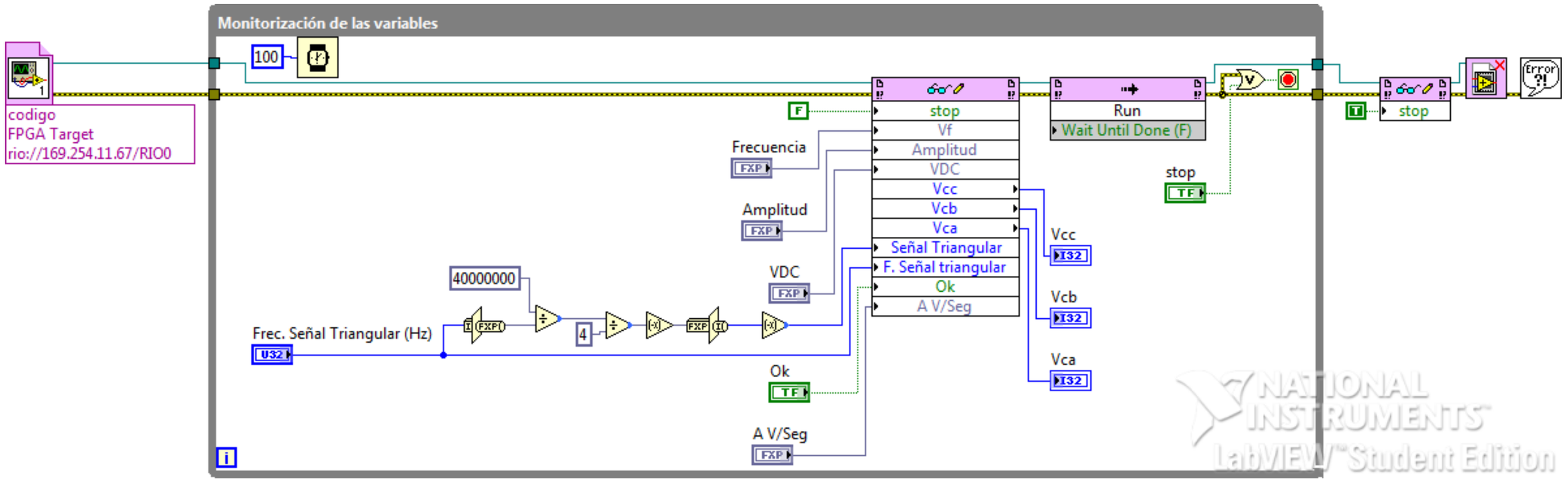




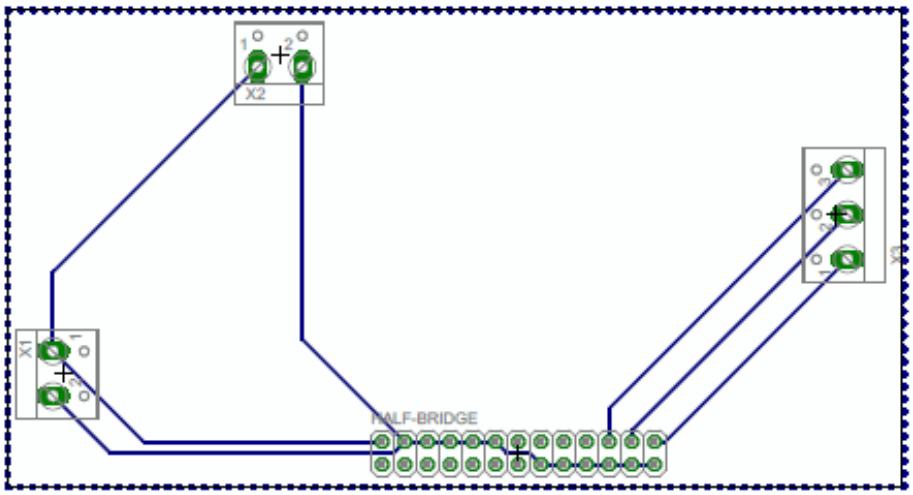


Frec. Señal Triangular (Hz)

UNIVERSIDAD POLITÉCNICA DE CARTAGENA		
Proyecto:	Control escalar de velocidad de una máquina asíncrona utilizando una tarjeta sbRIO-9696	
Plano:	SVPWM. Interfaz visual del programa monitorizacion.vi	Plano nº: 7
Autor:	Pablo Sánchez Sánchez	Fecha: 28/08/2015



UNIVERSIDAD POLITÉCNICA DE CARTAGENA		
Proyecto:	Control escalar de velocidad de una máquina asíncrona utilizando una tarjeta sbRIO-9696	
Plano:	SVPWM. Diagrama de bloques del programa monitorizacion.vi	Plano nº: 8
Autor: Pablo Sánchez Sánchez		Fecha: 28/08/2015



UNIVERSIDAD POLITECNICA DE CARTAGENA	Proyecto: Control escalar de velocidad de una máquina asincrona utilizando una sbRIO-9696	Plano nº: 9
Fecha: 28/08/2015	Plano: Diseño electrónico de la placa de conexiones - capa bottom	