Generación de Ondas SPWM con Arduino para la Excitación de Inversores Trifásicos

Andriach, Juan Pablo; Diaz, Ariel Ivan; Mariani, Cristian; Silva Bustos, Matias; Iparraguirre, Javier
Universidad Tecnológica Nacional
Facualtad Regional Bahía Blanca
11 de Abril 461, Bahia Blanca, Argentina
{juan.p.andriach, arielivandiaz, mariani.cristian, masilvabustos, javierip}@ieee.org

***Resumen* - En este artículo se presenta una implementación simple de un generador de onda senoidal mediante PWM, para ser aplicado en el control de un inversor trifásico. Se utilizó la plataforma Arduino Uno como base del proyecto, la cual utiliza como núcleo el microcontrolador AtMega328P. Se realizaron simulaciones y se evaluaron los resultados experimentales a partir de las formas de onda obtenidas. Se analiza además las limitaciones de la implementación desarrollada.**

Palabras clave - arduino; pwm; inversor; generador onda senoidal;

I. INTRODUCCIÓN

El presente trabajo es el resultado de la inquietud de lograr un controlador digital de velocidad para motores de corriente alterna basado en hardware y software libre. Como principal requisito se planteó que el control del sistema de potencia debería ser sencillo y económico. El primer paso consta en el diseño de un inversor trifásico el cual nos permitiera generar las ondas senoidales para alimentar el motor de CA (corriente alterna), a partir del sistema de suministro de energía.

Es posible encontrar trabajos similares donde se muestra que el control puede ser analógico [1] o digital [2, 3]. Nuestro aporte se distingue debido a que el control se basa en Arduino. Esto permite obtener el mismo resultado con una plataforma abierta y de bajo costo. En la implementación digital propuesta en este artículo se genera directamente las ondas senoidales utilizando las señales de salida PWM que proporciona la placa Arduino.

En el caso de los inversores utilizados como variadores de velocidad para motores de inducción estos deben tener la capacidad de ajustar la frecuencia de acuerdo a la velocidad de salida deseada. Además se debe poder ajustar la tensión de salida de modo que se mantenga un flujo constante del entrehierro en la región del par de torsión constante.

En la siguiente sección se hace una introducción a la plataforma Arduino y las especificaciones técnicas del hardware utilizado. Luego se describen los fundamentos teóricos del algoritmo y la forma de implementarlo por software. En la siguiente IV se presentan los resultados de las simulaciones y de los ensayos de laboratorio, se analizan las formas de onda y las características espectrales de las señales. Por último se hace un análisis de la posible implementación del algoritmo para la excitación de inversores trifásicos.

II. PLATAFORMA ARDUINO

Arduino es una plataforma electrónica abierta que permite utilizar software y hardware libre para el desarrollo de proyectos. Posee un entorno de desarrollo flexible y fácil de usar. La placa utilizada en este trabajo es la Arduino Uno [4], basada en un microcontrolador Atmega 328. Alguna de sus características son las siguientes:

- MicrocontroladorAtmega 328P @ 16MHz.
- Voltaje de trabajo 5 V.
- Tensión de alimentación 7 V -12 V.
- Conexión USB.
- 32 kB de memoria flash (Programa).
- 2 kB SRAM
- 14 terminales de E/S digital (6 PWM de 8 bits)
- 6 (DIP) o 8 (SMD) terminales de entrada analógicos

El lenguaje de programación y las librerías que incorpora Arduino permiten el control de sus salidas destinadas para PWM. Para ello se invoca a la función *analogWrite(pin, dutyCycle)*, donde *pin* es una de las salidas PWM (pines 3, 5, 6, 9, 10 y 11) y *dutyCycle* es un valor entre 0 y 255. Si bien esta función ofrece una implementación simple para el PWM, no permite controlar la frecuencia del mismo.

III. IMPLEMENTACIÓN

En la PWM el ancho de cada pulso se modula según otra función, llamada moduladora. En este caso, la función moduladora es una función sinusoidal. El microcontrolador provee el hardware para la síntesis de PWM, pero es necesario configurarlo en tiempo de ejecución. Para esto se calcula la señal moduladora y luego se cargan estos valores en los registros que controlan el PWM. A continuación se presentan dos alternativas para generar la función moduladora y luego el algoritmo utilizando.

A. Consideraciones del Cálculo de la Función Coseno

En aplicaciones de tiempo real los algoritmos están acotados en tiempo o ciclos de trabajo del CPU, y en un microcontrolador pequeño, en espacio o memoria también. Por ello es necesario analizar el diseño del algoritmo de cálculo numérico para una función dada.

También hay que tener en consideración la representación numérica de los números fraccionarios, que pueden ser de coma fija y coma flotante. Para el primer caso, se representa con un entero, suponiendo que los bits menos significativos representan valores fraccionarios. Para el segundo, se lo representa con dos enteros, una mantisa, representado en coma fija, y un exponente entero. Esto multiplica la cantidad de operaciones necesarias.

Ya que los valores de la función están comprendidos entre 1 y -1, se prefiere la representación en coma fija: 7 bits de mantisa + 1 bit de signo. La mantisa se pondera con 2^{-1} para el MSB, con 2^{-7} al LSB y en forma correlativa los demás. Otra forma de expresar lo anterior es $2^{-7} \cdot V$, donde V es el valor representado por el entero. De esta manera se codifican racionales de -1 a 0,9921875 incluyendo 0. La elección de la resolución se funda en el ancho de palabra del temporizador, el cual es igual al del argumento de *analogWrite()*. La suficiencia de ésta se verificará experimentalmente.

Como se adoptan valores discretos en el argumento de coseno hay que establecer la resolución, o sea el menor valor representado. El criterio adoptado es que dos valores consecutivos del argumento resulten en valores consecutivos del condominio, así se aprovecha eficientemente la representación.

$$\cos(z + R) - \cos x = R_a \qquad (1)$$

donde $R_a = arcsen(R)$ para $x \approx \pi/2$ que es donde la función cambia más rápidamente.
Para valores pequeños de R resulta $R_a = R$. Por tanto, una resolución de 1/128 radianes para el argumento es suficiente.

En general se estudiaron dos formas de representación:
por serie truncada, con requerimientos en tiempo, y por tabla, requiriendo espacio.

- Serie truncada

  Partiendo de la expansión en serie de Taylor-McLaurin de la función coseno:

  $$\cos x = 1 - x + \ldots. \qquad (2)$$

  Se puede demostrar que el error cometido al truncar en el término n-ésimo es menor a ese término. Como se trabaja con valores discretos, un error menor a 0,5 de la resolución es más que suficiente.

- Por tabla

  En este caso, el "cálculo" se convierte en un acceso a memoria. La resolución angular está dada por 2π/N radianes. La resolución de amplitud es de $2/(2^W - 1)$, para una representación con una palabra de W bits. La tabla ocupa W· N bits.
  Se cumple:

  $$\omega \cdot t_d = R_a \qquad (3)$$

  donde t_d es el intervalo de tiempo entre muestras, y ω la pulsación que se desea obtener.

  Con una resolución fija, para variar la frecuencia o pulsación, basta modificar el tiempo de espera entre cada ciclo t_d.
  Sin embargo, para la realización de un PWM con frecuencia de portadora fija, (t_d constante, ω variable) esta técnica no es conveniente ya que exige una resolución variable con la frecuencia moduladora. Una posible solución es la interpolación de valores, exigiendo un cálculo adicional que tendrá que ser tenido en cuenta.

B. Algoritmo

Por cuestiones de simplicidad y rendimiento se optó por el método por tabla y se encaró el algoritmo en forma directa, siendo este:

```
loop()
{
        ++N;
        analogWrite(cos(N*R));
        delayMicroseconds(PERIODO/N);
}
```

Donde *N* es el número de elementos en una tabla de valores, R es la resolución angular, y *PERIODO* es el periodo de la señal moduladora (coseno) en microsegundos

Este algoritmo tiene la ventaja de ser sumamente simple y como primera aproximación es más que suficiente. Sin embargo adolece de ciertos inconvenientes:

- Impone un retardo en el lazo principal del programa, lo que hace que el agregado de otros procedimientos de escrutinio y cálculo se haga difícil, sino imposible.
- La salida analógica de *analogWrite()* no es tal, sino que se implementa por modulación PWM.
- No asegura el sincronismo correcto entre el inicio de un pulso de la salida PWM y la modificación del ciclo de trabajo, dando un comportamiento indefinido a la salida desde la llamada a *analogWrite()* hasta el inicio del siguiente ciclo PWM.
- Los niveles de salida de la señal senoidal resultan "escalonados" en vez de variar en forma continua, lo que agrega armónicas.

IV. Simulación

Analizando el circuito esquemático [4] de la placa Arduino podemos observar que está compuesta básicamente por dos partes, la etapa de programación (interfaz USB) y el procesador. La parte de interés es la sección del procesador, con su respectivo oscilador de frecuencia, la cual se simuló con Proteus [5]. Este entorno de desarrollo posee en sus librerías el procesador AtMega328P, lo que facilita el trabajo de simulación considerablemente. Desde el IDE de Arduino, podemos obtener el archivo hexadecimal de la compilación, el cual se puede cargar directamente al procesador en la simulación de Proteus.

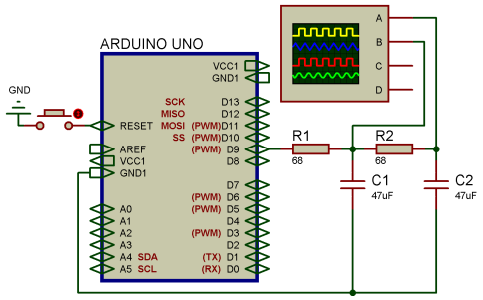

Fig.: 1 Captura del esquemático en Proteus

En la Figura 1 se observa el esquemático de la simulación de Arduino con un doble filtro RC sobre el cual se analizó la señal de salida del PWM por medio del osciloscopio en la Figura 2. En esta última figura se observa la señal filtrada por el doble filtro RC (amarillo) y la señal correspondiente a la salida del primer filtro RC (azul).

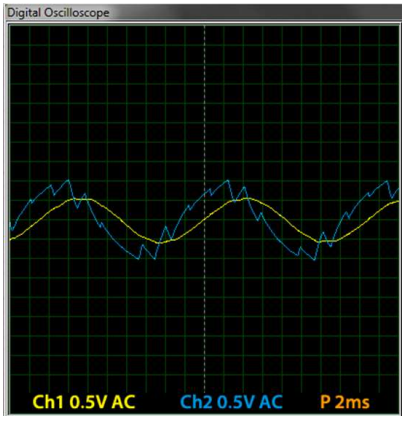

Fig.2: Captura del osciloscopio de la simulación, usando doble filtro RC y un solo filtro RC.

A partir de la figura se concluye que al pasar la señal por un segundo filtro RC, se reducen los picos montados sobre la misma, dando lugar a una onda senoidal de mejores características. Esta prueba se realizó con el fin poder verificar el correcto funcionamiento de la función que genera la señal en Arduino, así como su correcta implementación. Continuando con la simulación, se procedió a evaluar el comportamiento de las tres salidas PWM funcionando simultáneamente, dando como resultado la representación de una tensión trifásica en la que cada onda está 120º desfasada de las demás.

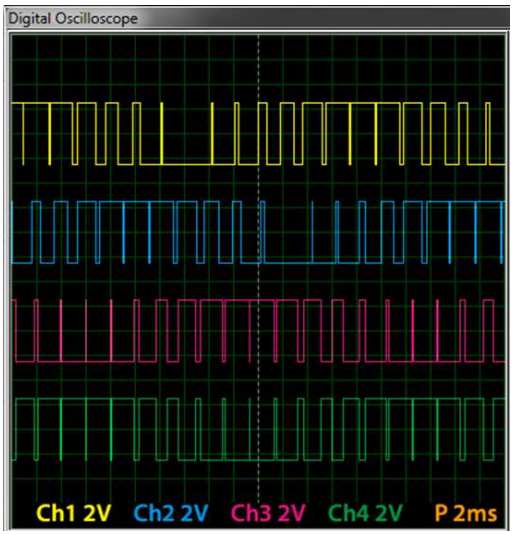

Fig.3: Captura del osciloscopio de la simulación de las 3 salidas PWM y en el canal 4 la señal componente del canal 3.

V. Ensayos y Resultados

A. *Formas de Onda*

Una vez efectuada la simulación se procedió a realizar los primeros ensayos de laboratorio con el algoritmo para generar las ondas seno. En la Figura 4 se observa la forma de onda de

la señal tomada de una de las salidas PWM de la placa Arduino. En la Figura 5 se puede apreciar la misma señal pero luego de ser filtrada por un doble filtro RC. De esta manera se pudo comprobar que efectivamente la modulación del PWM corresponde con la de una onda senoidal, cuya frecuencia puede ser variada según lo mencionado en la sección anterior.

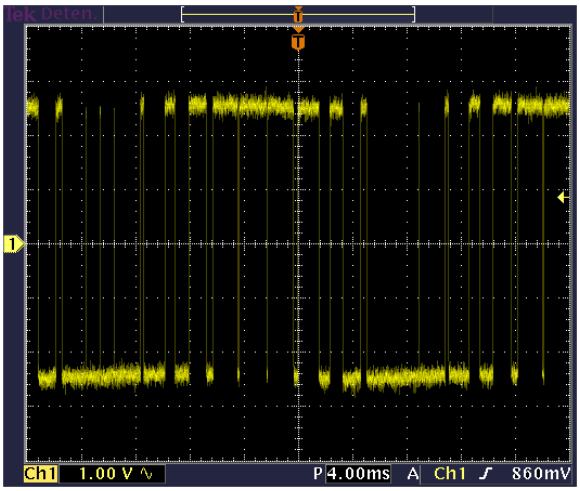

Fig.4: Captura del osciloscopio de la señal SPWM.

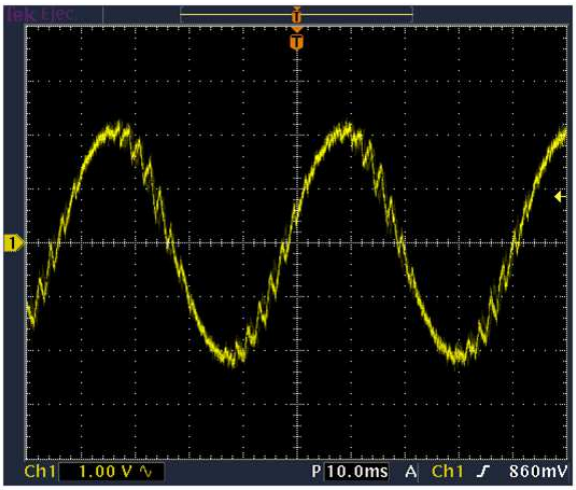

Fig.5: Captura del osciloscopio de la señal SPWM filtrada.

B. Analisis de Espectro

A continuación se hicieron ensayos usando un analizador de espectro por medio de una placa adquisidora con interfaz en LABVIEW [6]. A partir de este análisis se tomaron tres capturas, en la cuales se puede apreciar que todas poseen una componente de frecuencia en común ubicada en 490 Hz, la cual corresponde con la frecuencia de trabajo del PWM de Arduino.

La Figura 6 muestra el PWM trabajando a 50Hz, que corresponde a la frecuencia estándar de la mayoría de las conexiones eléctricas de baja potencia. Por medio de la interfaz de LABVIEW se obtuvo que el valor medio de la componente fundamental es de 2,15 Volts, el de la componente de 490Hz es de 1.3 Volts, mientras que el valor medio de la mayor componente armónica corresponde a 0,7 Volts.

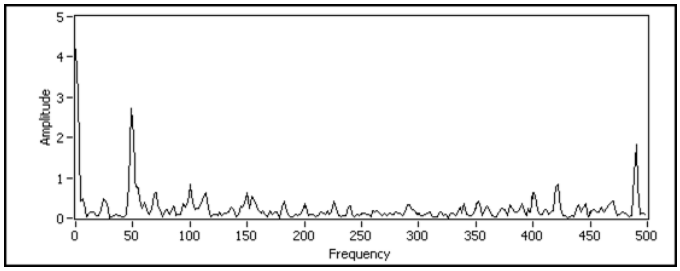

Fig. 6: Análisis de espectro de la señal generada a 50 Hz.

Al disminuir la frecuencia de la señal generada se encontró que a valores muy bajos de la misma, la componente fundamental comienza a mezclarse con la componente de continua que es parte del PWM de Arduino. Se consideró 15Hz como un valor mínimo en el cual la componente fundamental está lo suficientemente separada y bien definida con respecto a la componente de continua. Esto se puede observar en la Figura 7.

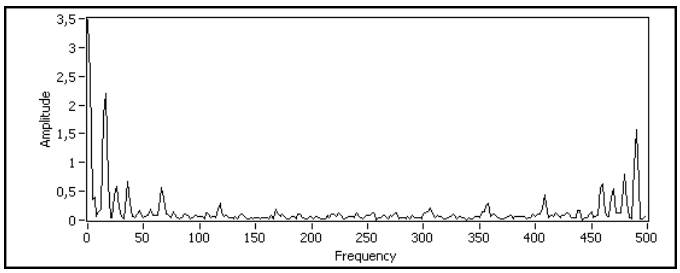

Fig.7: Límite inferior en 15 Hz.

Mediante una serie de ensayos, se comprobó que al aumentar la frecuencia de la señal, se llega a un valor límite, dado que por más que se disminuya el periodo de la señal que se genera por software, la placa Arduino no responde incrementando la frecuencia. Se llegó como a límite a un valor de 165Hz, en donde la amplitud de la componente fundamental es considerablemente menor que en las señales de menor frecuencia anteriormente ensayadas. Además la amplitud de los armónicos se ve incrementada a medida que la frecuencia se acerca al valor límite. Este caso se puede observar con detalle en la Figura 8.

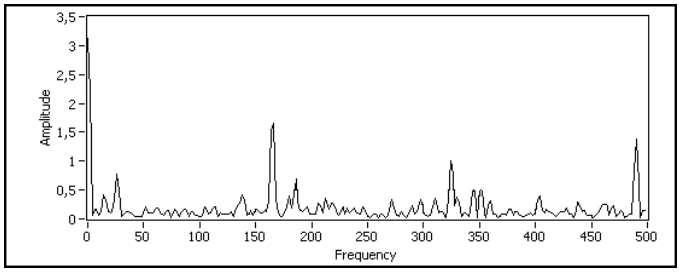

Fig.8: Límite Superior 165 Hz.

En la Figura 9 se representan los valores medidos de frecuencia ensayada versus los valores de frecuencia teóricos que se le pidieron a la placa Arduino por software. Descartamos los valores menores a 15Hz por la falta de definición y para valores mayores a 150Hz, la respuesta deja de ser lineal, por lo que el Arduino deja de responder de la forma esperada.

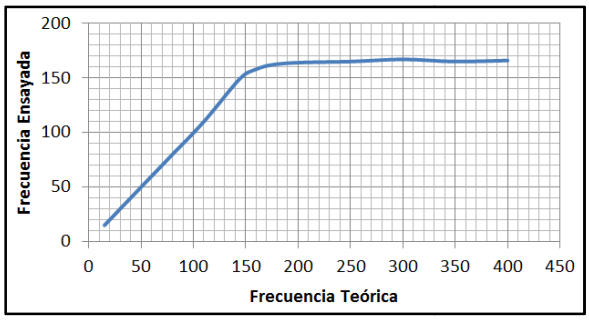

Fig. 9: Gráfico de frecuencia ensayada versus teórica según las pruebas realizadas.

VI. Aplicaciones y Trabajos Futuros

Considerando que los resultados obtenidos en los ensayos son más que aceptables, se puede pensar en que la implementación desarrollada es totalmente apta para ser aplicada en la excitación de inversores trifásicos y en el control de velocidad de motores de CA. Es por ello que se procedió a ensayar el algoritmo, pero esta vez aplicado a tres canales PWM del Arduino para generar una tensión de salida trifásica.

La Figura 10 tomada del osciloscopio muestra la forma de onda SPWM de las tres fases. La frecuencia de cada una de estas es controlada y variada de forma simultáneamente por software. Se puede observar que cada una de las tres señales se encuentra desfasada 120º de las demás.

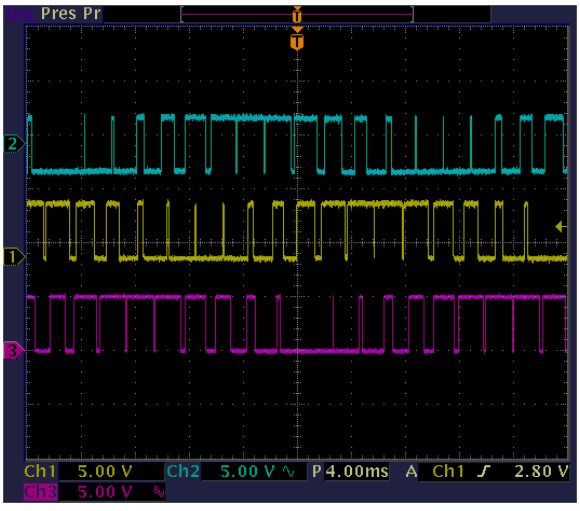

Fig.10: Señales SPWM desfasadas 120º tomadas del Arduino.

Para la implementación del inversor trifásico, cada una de las fases debe ser invertida para poder excitar cada par de interruptores complementarios [7]. La inversión se realizó por hardware, es decir, utilizando tres compuertas AND 7408. La Figura 11 muestra la salida de uno de los canales PWM (amarillo) y su correspondiente complemento (verde).

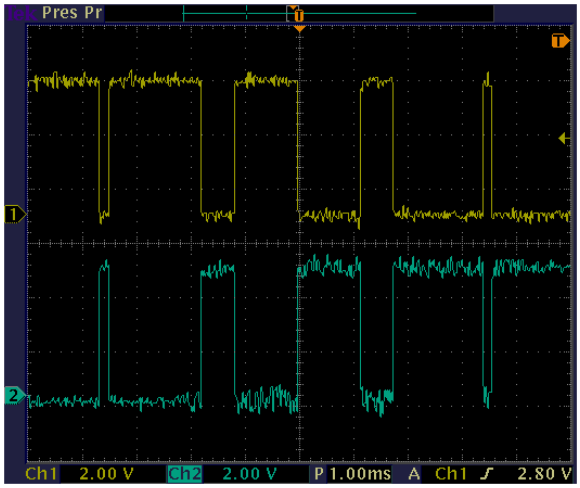

Fig. 11: Señales SPWM complementarias.

Analizando las señales en superposición, se aprecia que el tiempo muerto en cada conmutación es prácticamente nulo. Esto es gracias a que el tiempo de propagación de la compuerta es de unos pocos nanosegundos.

VII. Conclusiones

La implementación desarrollada en este trabajo permite la generación de ondas senoidales para la excitación de inversores trifásicos de una manera sencilla y utilizando una plataforma electrónica abierta, de hardware y software libre, a un bajo costo. Los ensayos de laboratorio arrojaron muy buenos resultados, trabajando dentro de los límites de frecuencia establecidos en las secciones anteriores. De estas condiciones es posible definir un intervalo de trabajo para la generación de una onda senoidal lo suficientemente limpia, variando la frecuencia de la componente fundamental entre 15Hz y 150Hz.

Si bien el algoritmo fue pensado para el control de un variador de velocidad de motores de AC, tiene múltiples aplicaciones en el manejo de accionamientos de potencia.

Recurriendo a esta implementación como base, es posible aumentar la complejidad del algoritmo para poder controlar de manera efectiva la amplitud de la tensión de salida, y de esta manera lograr que la relación v/f se mantenga constante, condición necesaria para el manejo de motores de inducción.

REFERENCIAS

[1] Ian F. Crowley, Ho Fong Leung, “PWM Techniques: A Pure Sine Wave Inverter”, Worcester Polytechnic Institute Major Qualifying Project, 2011.

[2] Aganza T. Alejandro, Pérez R. Javier y Beristain J. José Antoni, “Inversor trifásico SPWM para el control de velocidad de un motor de inducción implementado en el microcontrolador PIC18F2431”, RIEE&C, Revista de Ingeniería Eléctrica, Electrónica Y Computación, VOL. 2 NO. 1, 2006.

[3] Gamboa Benítez Silvana del Pilar, Quelal Analuisa Paulo Alexis, Rivera Argoti Pablo, “Diseño y Construcción de un Variador de Velocidad con el Microcontrolador 80C196MC”, Escuela Politécnica Nacional, JIEE, Vol. 19, 2005.

[4] Arduino Uno Rev3 Schematic [acceso 16 de Junio de 2013] <**http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf**>

[5] Proteus Desing Suite Version 8 [acceso 16 de Junio de 2013]<**http://www.labcenter.com/index.cfm**>

[6] Software de Desarrollo de Sistemas NI LabVIEW [acceso 19 de Junio de 2013] <**http://www.ni.com/labview/esa/**>

[7] Ned Mohan, Tore M. Undeland, William P. Robbins, “Electrónica de Potencia, Convertidores, aplicaciones y diseño”, Tercera Edición.

[8] Gregorio Moctezuma Jiménez, Gabriel G. Luna Mejía y Daniel U. Campos-Delgado, “Diseño e Implementación de un Variador de Velocidad para Motor CA”.

[9] Thida Win, Hnin Nandar Maung, “Analysis of Variable Frecuency Three Phase Induction Motor Drive”, World Academy of Science, Engineering and Technology 18, 2008.

[10] A. Maamoun, M. Ahmed, “Microprocessor Control System for PWM IGBT Inverter Feeding Three-Phase Induction Motor”