



UNIVERSIDAD AUTÓNOMA DE YUCATÁN
FACULTAD DE INGENIERÍA

**DESARROLLO DE HERRAMIENTAS
COMPUTACIONALES PARA EL DISEÑO Y
PRUEBA DE UN CONTROLADOR PARA UN
RECTIFICADOR MULTINIVEL
MONOFÁSICO HÍBRIDO**

TESIS

PRESENTADA POR:

RAMIRO JOSÉ ÁLVAREZ RAMÍREZ

**EN SU EXAMEN PROFESIONAL
EN OPCIÓN AL TÍTULO DE:
INGENIERO EN MECATRÓNICA**

MÉRIDA, YUCATÁN, MÉXICO,

2017



UADY

FACULTAD DE
INGENIERIA

Febrero 19 de 2016
SA11/0/073

C. Pasante de Ingeniero en Mecatrónica
Ramiro José Álvarez Ramírez
Presente

En relación con el escrito presentado por el **Dr. Manuel Israel Flota Bañuelos** en el que certifica haber revisado y aprobado la Tesis "**Desarrollo de herramientas computacionales para el diseño y prueba de un controlador para un rectificador multinivel monofásico híbrido**", que le fuera señalada para la presentación de su Examen Profesional, se le autoriza la impresión de los ejemplares necesarios.

Atentamente,
"Luz, Ciencia y Verdad"



Facultad de Ingeniería
SECRETARÍA ACADÉMICA

ING. JORGE ALEJANDRO TAPIA GONZÁLEZ
Secretario Académico

c.c.- Secretaría Administrativa
c.c.- Archivo
JATG/mgdb*

Aunque este trabajo hubiere servido para el Examen Profesional y hubiere sido aprobado por el sínodo, sólo el autor es responsable de las doctrinas emitidas en él.

Prefacio

Existe un problema a la hora de desarrollar productos de electrónica de potencia y especialmente rectificadores, que en el caso de este escrito también es una fuente de poder conmutada, porque puede llegar a ser costoso y peligroso al momento de probar diferentes algoritmos para el controlador encargado de la conmutación en caso de que se haya pasado por alto algún error.

Por lo tanto se plantea el desarrollo de una herramienta de software para computadora que permita probar el algoritmo del controlador tanto en simulación como en emulación sin necesidad del circuito objetivo y no las características eléctricas o térmicas de un circuito como la mayoría de los softwares comerciales. Dado que la mayoría de los microcontroladores pueden funcionar con alguna variación del lenguaje de programación C, la herramienta desarrollada es capaz de utilizar cualquier librería de funciones en C estándar que el usuario le cargue, con esto podrá ver en la pantalla de la computadora cómo reacciona su modelo matemático y variar los diferentes valores que pueda tener el mismo sin necesidad de recompilar.

Índice

Introducción.....	1
Antecedentes.....	2
Rectificadores monofásicos.....	2
Herramientas computacionales.....	6
Capítulo I. Modelo matemático del rectificador activo.....	13
Modelo inicial del rectificador.....	15
Modelo de la corriente	18
Consideración de diferentes resistencias	19
Controlador	21
Observador	25
Modelo completo del sistema	28
Conversión a MATLAB	29
Resultados.....	31
Capítulo II. Herramienta de diseño: Simulador del modelo del rectificador activo.....	36
Traducción de código de MATLAB a lenguaje C	37
Interfaz del simulador	39
Simulador.....	41
Resultados	46

Capítulo III. Herramienta de pruebas: Interfaz de control del emulador.....	50
Capítulo IV. Emulador	53
Resultados	58
Conclusiones.....	62
Referencias	63
Apéndices	67
A. Tabla de comandos del microcontrolador	67

Lista de figuras

Figura 1. Puente de diodos para conversión de corriente alterna a corriente directa.	3
Figura 2. Comportamiento de un rectificador por puente de diodos ante una perturbación en la señal de alimentación.....	4
Figura 3. Filtro de corriente alterna para reducción de la THD.	4
Figura 4. Circuito rectificador de tres niveles.	5
Figura 5. Interface de la herramienta de diseño para convertidores DC-DC commutados.	8
Figura 6. Interfaz de la herramienta de síntesis para moduladores Sigma-Delta.	10
Figura 7. Resultados de la herramienta de prototipado virtual para sistemas electrónicos de potencia: a) Grafica de voltaje y corriente con respecto a la temperatura en el tiempo, b) Comparación térmica de la placa base de la simulación (izquierda) y medición experimental (derecha).	12
Figura 8. Topología del rectificador analizado.....	13

Figura 9. Estados posibles en el circuito.	14
Figura 10. Sentidos de las corrientes en el circuito rectificador.....	16
Figura 11. Diagrama de bloques matemáticos.	29
Figura 12. Simulación en MATLAB Simulink basado en el modelo matemático del sistema.	31
Figura 13. Resumen de las salidas (V_T segunda gráfica, V_D tercera y I_S abajo) del rectificador basados en una alimentación de 127 VAC.....	32
Figura 14. Comparación de las tensiones totales de salida V_T , arriba la salida del sistema, en medio la tensión deseada y abajo el error de la tensión analizada.	33
Figura 15. Comparación de las diferencia de tensiones entre los capacitores V_D , arriba la salida del sistema, en medio el valor deseado y abajo el error analizado.....	34
Figura 16. Comparación de la corriente consumida, arriba la salida del sistema, en medio la forma de la corriente deseada y abajo el error analizado.....	35
Figura 17. Diagrama de flujo del DLL.	39
Figura 18. Captura de pantalla de la interfaz de PC	40
Figura 19. Archivo *.CSV con los nombres y valores de los parámetros.....	42
Figura 20. Diagrama de flujo del simulador en la interfaz de PC.	45
Figura 21. Resultados la tensión total de salida en el simulador.....	47
Figura 22. Resultados de la diferencia de la tensión en la simulación.	48
Figura 23. Resultados de la corriente consumida en el simulador.	49
Figura 24. Controles de la conexión y cambio de valores de la interfaz al emulador.....	51
Figura 25. Arreglo de los bytes por posición del protocolo de recepción.	52
Figura 26. Arreglo de los bytes por posición del protocolo de transmisión.	53
Figura 27. Fotografía del PSoC 4 Prototyping Kit.	54

Figura 28. Captura de la configuración del componente serial	55
Figura 29. Configuración del IC USB-Serial en la PCB del PSoC4 Pioneer Kit.....	56
Figura 30. Diagrama de flujo del emulador.....	57
Figura 31. Ejemplo de creación de paquete.....	58
Figura 32. Resultados de la tensión total de la emulación en el microcontrolador.	59
Figura 33. Resultados de la diferencia de tensiones entre capacitores en la emulación en el microcontrolador	60
Figura 34. Resultados de las corrientes en la emulación del microcontrolador.	61

Lista de tablas

Tabla 1. Relación de los estados del rectificador analizado.	15
Tabla 2. Resumen de las ecuaciones del modelo matemático.....	30
Tabla 3. Valores de prueba del algoritmo.....	31

Introducción

Una de las formas más comunes de convertir corriente alterna (AC) a corriente directa (DC) es a través de circuitos rectificadores de puente completo monofásico (ver Figura 1), estos son capaces de mantener una tensión de corriente directa con bajo rizo con la desventaja que provocan una alta distorsión a la fuente causando armónicos en la tensión y la corriente, además de tener un bajo factor de potencia y en caso de haber una caída de tensión en la alimentación, estos circuitos son incapaces de mantener el voltaje de salida deseado (Lin, Huang, & Yang, Control scheme of hybrid active filter for power quality improvement, 2002).

Por otra parte, se han desarrollado varias herramientas computacionales para agilizar el proceso de diseño de sistemas electrónicos de potencia pero dependen de librerías o programas de terceros para poder operar, además que estas herramientas tienden a enfocarse únicamente en el co-diseño de hardware-software del sistema y obviando las demás partes del desarrollo como las pruebas de rendimiento, depuración, optimización y caracterización del sistema (Teich, 2012).

Este trabajo tiene como objetivo presentar un nuevo enfoque en el proceso de diseño e implementación de un rectificador monofásico para superar los inconvenientes de los modelos de desarrollo existentes permitiendo facilitar y agilizar el proceso de diseño del sistema.

Antecedentes

Rectificadores monofásicos

Actualmente la mayoría de la maquinaria, instrumentos, herramientas, aparatos, etc. funcionan con energía eléctrica por lo que se ha vuelto un recurso indispensable tanto para la industria, el comercio y el hogar por lo tanto tener una buena calidad en la señal eléctrica se ha vuelto una preocupación para muchos países, debido a esto se han realizado diferentes estándares de los valores permitidos en ésta (Flota, 2009). La contaminación de la calidad de la energía eléctrica se ha debido, en gran parte, a la proliferación de cargas no lineales producidas por los aparatos electrónicos que día a día van aumentando en los diferentes ambientes de nuestra vida.

Debido a que los aparatos electrónicos funcionan en su mayoría con corriente directa es necesario rectificar la corriente alterna de la toma de corriente (127VAC, 60Hz) lo cual en condiciones normales y sin ningún tipo de control aplicado provocan ruido en la señal de entrada por la consunción de la carga no lineal. Esta contaminación en la calidad de la señal eléctrica se cuantifica como la distorsión armónica total (*THD*, por sus siglas en inglés), este término expresa la distorsión como un porcentaje de la frecuencia fundamental del voltaje y la corriente (Shmilovitz, 2005).

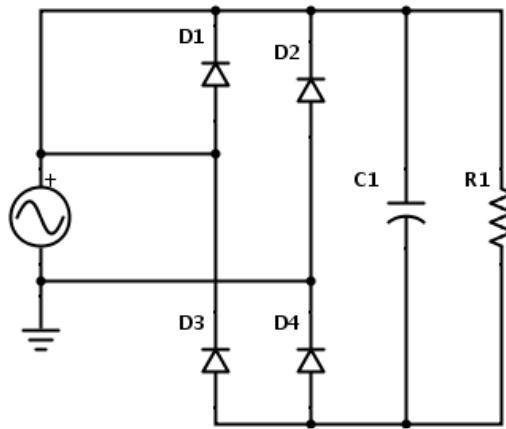


Figura 1. Puente de diodos para conversión de corriente alterna a corriente directa.

Usualmente cuando se quiere convertir la corriente alterna de la toma de corriente (127VAC, 60Hz) en corriente directa (180V sin ningún tipo de control aplicado), se utiliza la estructura clásica de un rectificador de puente completo el cual es alimentado por la conexión a corriente alterna al cual se le encuentra conectado a su salida una carga cualquiera (ver Figura 1); una desventaja de este tipo de circuito es que su consumo será altamente no lineal y dependerá únicamente de la carga conectada.

La Figura 2 muestra los problemas a la salida de un rectificador por puente de diodos cuando la tensión de entrada sufre una disminución.

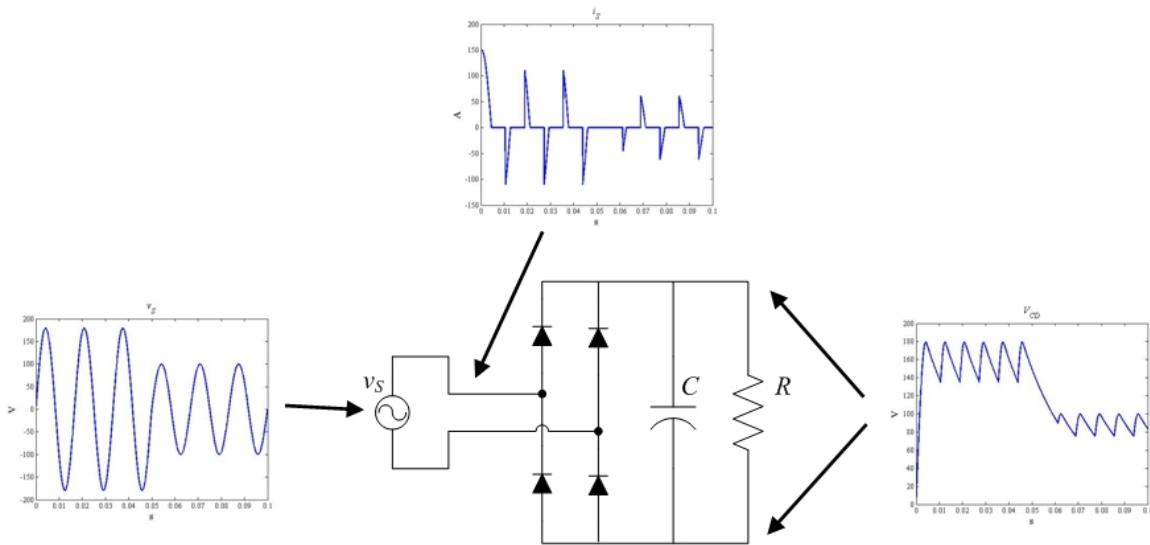


Figura 2. Comportamiento de un rectificador por puente de diodos ante una perturbación en la señal de alimentación.

Una forma común de disminuir la THD es poner un filtro de corriente alterna a la entrada del puente de diodos (ver Figura 3), esto ayudará la disminución de la THD pero seguirá dependiendo únicamente de la carga conectada (Rashid, 2001).

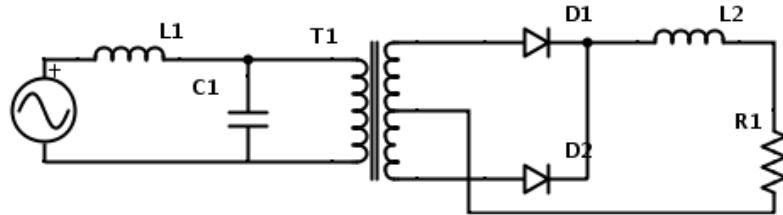


Figura 3. Filtro de corriente alterna para reducción de la THD.

Existen otras formas de disminuir la THD mediante el control activo de la rectificación de la corriente alterna, ya sea por SCRs o transistores, controlando de esta forma en que momento de la frecuencia se permite el paso de la corriente en cualquier sentido deseado ya sea para consumo de la carga como para la conversión a una sinusoidal quasi perfecta en el consumo de la corriente aun cuando la carga del circuito no la necesite. Una de las formas de hacer esto es mediante la comparación de una

sinusoidal de la misma frecuencia fundamental que la señal de alimentación en alterna contra una señal triangular de una frecuencia mucho mayor a la fundamental para crear una modulación por ancho de pulso (*PWM*, por sus siglas en inglés) el cual puede de ser de dos o tres niveles, los cuales mientras mayor cantidad de niveles tenga más difícil será el control pero mejor será la disminución de la THD (Lin & Lu, A New Control Scheme for Single-Phase PWM Multilevel Rectifier with Power-Factor correction, 1999).

También existen otras formas más complejas para la disminución de la THD, las cuales se enfocan en el control de otros elementos del circuito en vez de la activación selectiva del puente de diodos y aunque su control tiene una mayor dificultad producen mejores resultados, estos tipos de rectificación se hacen normalmente por control no lineal y observadores, por ejemplo véase la Figura 4 que muestra la estructura que será el objeto de estudio de este trabajo. Este tipo de control tiene ventajas ya que se puede ajustar a cargas variantes en el tiempo, por lo que la carga no se vuelve un factor primario en la forma de la corriente consumida sino que dependerá mayoritariamente del control realizado (Lin, Chen, & Tsay, Bi-Directional AC/DC converter base on neutral point clamped, 2001).

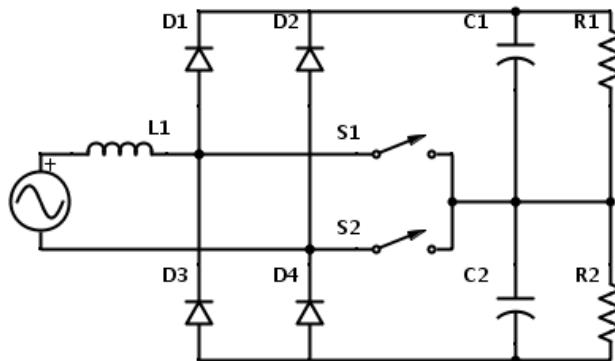


Figura 4. Circuito rectificador de tres niveles.

Actualmente los esquemas de control aplicados a rectificadores activos son los siguientes: Control proporcional con un disminución en la THD de 7.3% (Buja & Castellan, 2003), Control proporcional integral con una disminución del 2.7% (Hung, Lin, & Chen, 2002), Control PI y SVPWM con una disminución del 2.7% (Pires & Silva, 2005), Histéresis con una disminución del 2.6% (Huang, Lin, & Huang, 2003), Basado en pasividad los resultados indican que al incrementarse la carga deja de regular la tensión de CD pues ésta cae (Monopoli, Dell'Aquila, Liserre, & Rotondo, 2005), Predictivo con una disminución del 3.7% (Monopoli, Clare, Zanchetta, Gerry, & Wheeler, 2008).

Dentro del segmento de los circuitos controlados por observadores, se obtienen ventajas sustanciales a los anteriores debido a que tienen un mejor rendimiento a cargas variantes en el tiempo y estiman los valores de estas cargas durante el tiempo de ejecución (Hermann, 1997).

Herramientas computacionales

El co-diseño de hardware-software surgió como una nueva disciplina para diseñar circuitos integrados complejos a principios de la década de 1990. En ese entonces el diseño concurrente de hardware y software ya era una actividad de todos los días, al menos para las compañías de microprocesadores que tenían que decidir cuidadosamente como diseñar la interface entre el hardware y software del microprocesador. Debido a los avances tecnológicos predichos por Gordon Moore, las técnicas de co-diseño hardware-software se han vuelto hoy una necesidad para un exitoso diseño de un sistema electrónico y por lo tanto, son usados cada vez más por compañías que desarrollan productos de sistemas electrónicos embebidos (Teich, 2012).

Debido a la complejidad de los sistemas electrónicos analógicos, digitales y de potencia actuales, es un hecho conocido que los proyectos que se realizan requieren de un enfoque multidisciplinario, que

permitan a expertos y no expertos en las diferentes áreas facilitar y agilizar el proceso de diseño de estos, por lo cual se requiere el desarrollo de herramientas computacionales que permitan atacar los problemas de diseño en una forma más efectiva.

Ingenieros y científicos han utilizado ampliamente MATLAB en la industria, que es una herramienta de software con un lenguaje propio de alto nivel que permite agilizar el desarrollo matemático de un proyecto como son la manipulación de matrices, representación de datos, prueba de modelos de control, procesamiento de señales, finanzas computacionales, entre otros. Tradicionalmente se han desarrollado una amplia variedad de herramientas computacionales para usarse junto con las librerías de MATLAB o bien hechos dentro de este, debido a la familiaridad que tienen los ingenieros con su lenguaje de programación. Entre las aplicaciones basadas en librerías de MATLAB se encuentran la herramienta de diseño para fuentes de poder conmutadas utilizada para mejorar el aprendizaje en estudiantes de electrónica (Miaja, Lamar, & Pérez, 2001), este tipo de herramienta tiene como ventaja aumentar el entendimiento en el funcionamiento del sistema incluso antes de su construcción permitiendo así probar diferentes modelos de controladores y reduciendo el tiempo de diseño comparado con el modelo tradicional.

Se han desarrollado igualmente herramientas gráficas de diseño integrado para controladores de convertidores DC-DC conmutados para FPGAs (*Field Programmable Gate Array* por su siglas en inglés) usando simulación en MATLAB Simulink con generación de código Verilog HDL (Leng, Yang, & Tsai, 2008) permitiendo a través de este modelo de trabajo introducir únicamente los parámetros de diseño en el programa y así poder simular y exportar los resultados para su implementación como muestra la Figura 5.

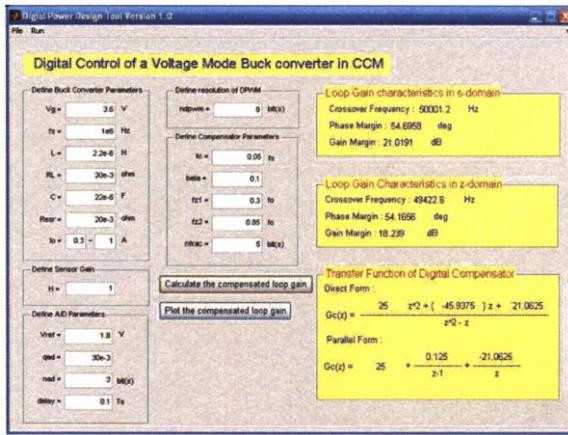


Figura 5. Interface de la herramienta de diseño para convertidores DC-DC conmutados.

En el ramo militar también existen herramientas gráficas de diseño como la desarrollada por el programa de defensa de Corea del Sur junto con la universidad nacional de Seúl para sistemas electrónicos de potencia para vehículos de combate eléctricos híbridos (Choi, Lee, Cho, & Yoon, 2010) que tiene por objetivo crear un sistema de potencia híbrido óptimo y una estrategia de control de manejo óptimo incluyendo la reducción total del consumo de combustible y la optimización del desempeño de movimiento con respecto al costo, masa y espacio volumétrico requerido para el sistema de potencia del vehículo en cuestión, el cual puede ser 4x4, 6x6, 8x8 y de pista.

Sin embargo este tipo modelo de los trabajos mencionados anteriormente tiene la desventaja que la portabilidad del programa se ve reducida, limitando la cantidad de dispositivos que puedan ejecutarlas debido a los requisitos previos de las herramientas, y perdiendo la versatilidad debido a las restricciones del lenguaje de MATLAB.

Por estas razones también se han desarrollado otro tipo de herramientas basadas en lenguajes de computadora más generales, las cuales pueden ser exclusivas, gratuitas, libres o de pago. Entre las herramientas de co-diseño hardware-software creadas por empresas para sus propias plataformas

podemos encontrar PSoC Designer, PSoC Creator desarrollados por Cypress Semiconductor, este último programa es también utilizado para crear el software del proyecto de este escrito, los cuales sirven para crear el software y depurar los productos de la plataforma de esta empresa. Xilinx y Altera han creado también sus entornos de trabajo para sus plataformas de FPGAs los cuales permiten diseñar las arquitecturas, el firmware de sus circuitos integrados y plataformas de desarrollo así como la simulación, pero carecen de entornos de comunicación que permitan probar el rendimiento real del sistema a desarrollar y permiten, hasta un cierto grado, la optimización de estos. Normalmente estas compañías crean sus propios entornos de desarrollo integrado (*Integrated Development Environment, IDE* por sus siglas en inglés) para sus plataformas impidiendo así su utilización con otras soluciones industriales dificultando y ralentizando el proceso de desarrollo y prueba para la preparación del lanzamiento del producto electrónico. Por otra parte es común que empresas más pequeñas o profesionistas independientes generen sus propios programas personalizados que se adapten a una necesidad específica de sus proyectos, pero esto resulta en una pérdida de tiempo al no contar con una herramienta que centralice la mayoría o todas las necesidades del proyecto electrónico a realizar, lo que genera que se tenga que estar alternando entre programas, IDE y/o kits de desarrollo de software (*Software Development Kit, SDK*) durante el diseño y prueba del sistema electrónico.

Entre las herramientas de diseño con lenguajes generales para sistemas electrónicos podemos encontrar aplicaciones web aceleradas por unidades de procesamiento gráfico de propósito general (*General Purpose Graphics Processing Unit, GPGPU*) para la síntesis de moduladores Sigma-Delta (Brückner, Zorn, & Anders, 2014), este enfoque toma la ventaja que provee una GPGPU de alto rendimiento que pueden llegar a tener cientos o incluso miles de núcleos programables con una capacidad alta de paralelización permitiendo reducir significativamente el tiempo de simulación a

unos pocos minutos en comparación a simulaciones hechas en procesadores (*Central Processing Unit*, CPU) únicamente los cuales pueden tardar desde decenas de minutos hasta horas según la cantidad de parámetros que se prueben en simulación. Esta herramienta tiene la particularidad que puede ser usada por cualquier persona con acceso a internet pues se encuentra para el uso libre a través de la página web www.sigma-delta.de en la Figura 6 se puede ver la interfaz gráfica de esta herramienta.

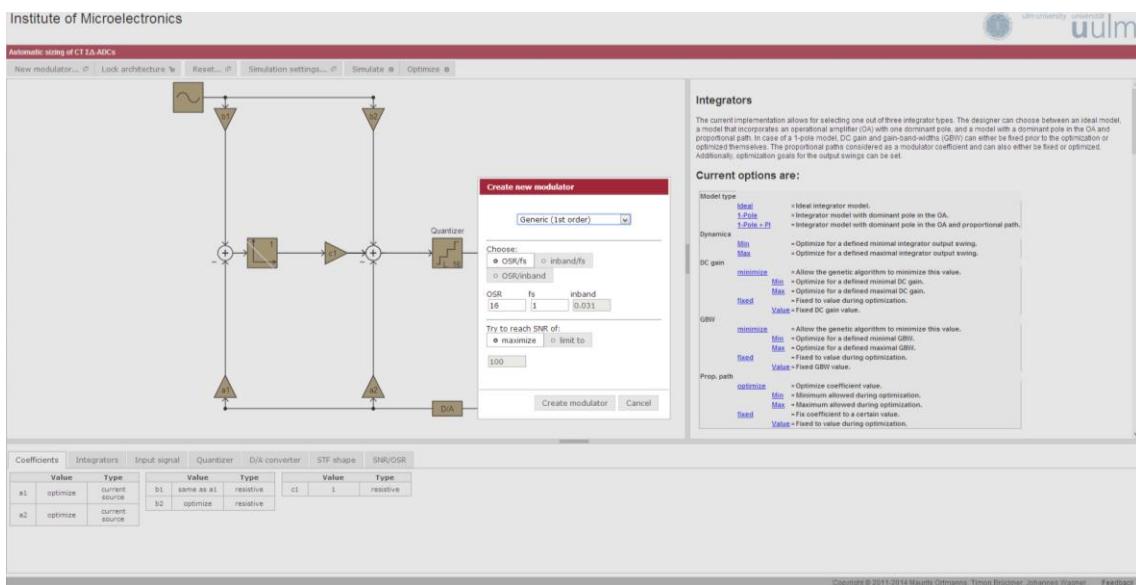


Figura 6. Interfaz de la herramienta de síntesis para moduladores Sigma-Delta.

Otra herramienta es Quick Fil que sirve para diseñar para filtros de microondas el cual fue desarrollado en C# .NET para propósitos educativos, esta aplicación no requiere ningún paquete de software adicional proporcionándole una gran portabilidad. Soporta tres principales diseños de filtros: Pasa bajas, Pasa altas y Pasa banda, una vez realizado el diseño devuelve la distribución de los elementos del circuito en placa, permite la exportación a formato JPEG de la distribución de la placa y genera

reportes con las especificaciones completas del proyecto una vez realizado (Radhakrishnan & Ravikumar, 2013).

También se han desarrollado herramientas de diseño más completas para prototipado virtual de sistemas electrónicos de potencia que permiten la evaluación de estos sin la necesidad de construir prototipos físicos (Evans, Castellazzi, & Johnson, 2014). Debido a que los sistemas electrónicos de potencia requieren técnicas multidisciplinarias de modelado para predecir su comportamiento lo suficiente para validar el desempeño del diseño potencial, este debe tener en cuenta el diseño de la geometría de la placa base y el circuito de aplicación. Dentro del diseño geométrico se deben considerar dos dominios físicos en el que el modelo geométrico debe operar: el electromagnético y el térmico. Esta herramienta requiere de un modelo matemático ideal del circuito de aplicación que incluya los dispositivos semiconductores a usar para poder simular el comportamiento del sistema a través del tiempo para los valores eléctricos y su desempeño térmico en estado estable. En la Figura 7 se puede ver los resultados de usar esta herramienta.

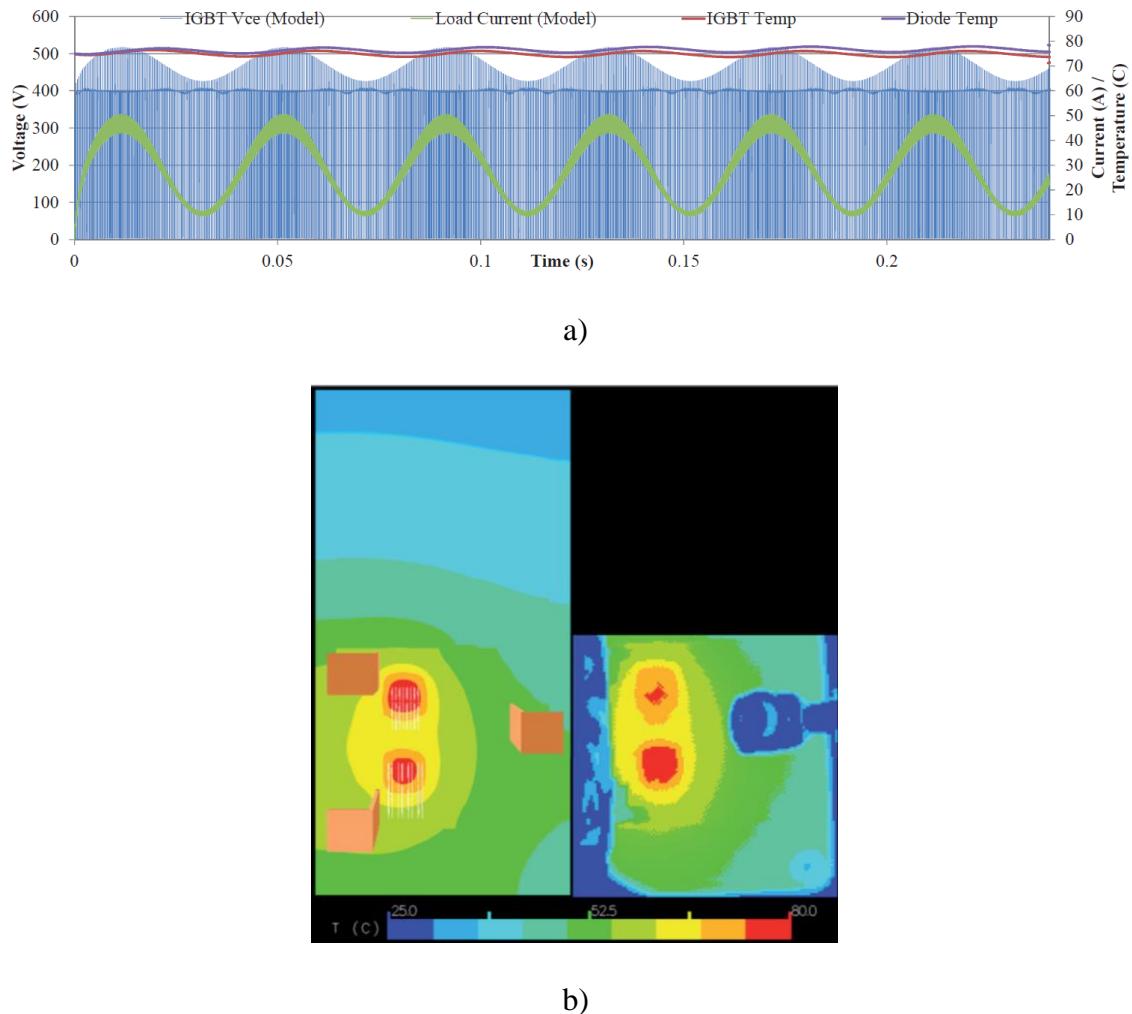


Figura 7. Resultados de la herramienta de prototipado virtual para sistemas electrónicos de potencia: a) Grafica de voltaje y corriente con respecto a la temperatura en el tiempo, b) Comparación térmica de la placa base de la simulación (izquierda) y medición experimental (derecha).

Capítulo I. Modelo matemático del rectificador activo

La topología elegida es la del rectificador monofásico híbrido como se muestra en la Figura 8, el cual tiene tres niveles de voltaje que son la tensión de la suma de los capacitores, la tensión de un capacitor (debe tenerse en cuenta que ambos capacitores tienen el mismo voltaje) y la diferencia de potencial entre capacitores. La salida es definida por la conmutación de los interruptores S_1 y S_2 que son de frecuencia variable y es dependiente únicamente de los valores de salida definidos por el usuario.

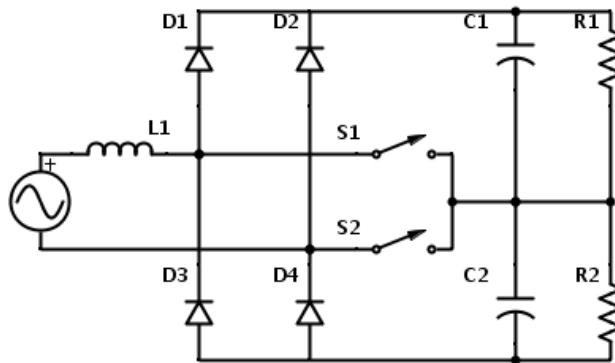


Figura 8. Topología del rectificador analizado.

Esta configuración es un elevador de voltaje que toma como entrada la salida monofásica, la utilizada comúnmente en el sector doméstico, de 127 VAC a 60Hz y la convierte en corriente directa de hasta 1000V a 1A, capaz de tener una salida de 1kW bajo cualquier valor de tensión de salida deseado, otra característica deseada a través del control es que mantenga un THD bajo en la corriente consumida.

Este circuito contiene 8 diferentes estados como muestra la Figura 9, todos estos estados pueden estar activos consecutivamente sin ningún orden específico pues su activación dependerá del control que se desarrollara más adelante. Los primeros cuatro estados suponen un flujo de corriente positiva mientras que los cuatro siguientes consideran flujos de corriente negativa.

Por ejemplo la primera sección de la Figura 9, el interruptor S_1 está cerrado mientras que S_2 se encuentra abierto provocando que se cargue el capacitor C_2 ; en la segunda sección, que también considera corriente positiva, vemos que al abrir el interruptor S_1 y mantener el S_2 cerrado, se cargara el capacitor C_1 .

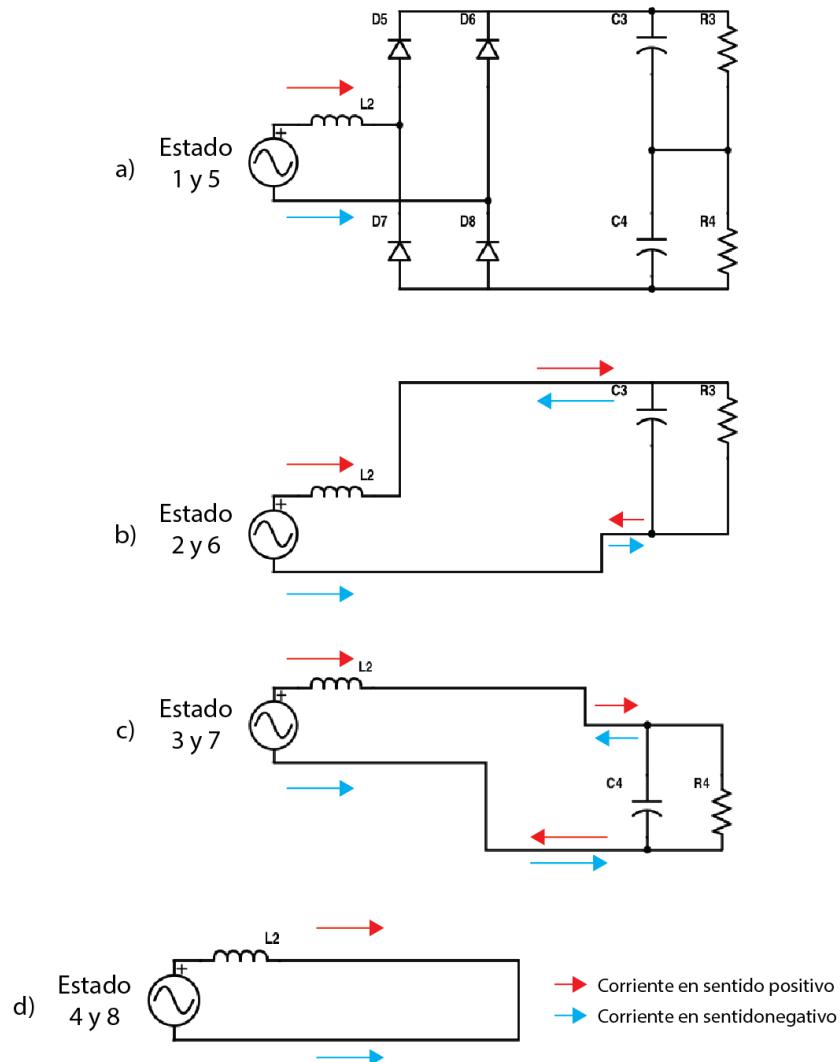


Figura 9. Estados posibles en el circuito.

En la Tabla 1 se pueden observar todas las diferentes interacciones entre las tensiones y las corrientes del circuito provocadas por el cambio de estado de los interruptores y el sentido de la corriente I_s ,

siendo V_{12} la diferencia de potencial entre las terminales de los interruptores (debe tenerse en cuenta que debido a que se está modelando el sistema se consideran interruptores ideales).

S₁	S₂	I_s	V₁₂	I₁	I₂
0	0	+	V _{C1} +V _{C2}	I _s	I _s
0	1		V _{C1}	I _s	0
1	0		V _{C2}	0	I _s
1	1		0	0	0
0	0		-(V _{C1} +V _{C2})	-I _s	-I _s
0	1		-V _{C2}	0	-I _s
1	0		-V _{C1}	-I _s	0
1	1		0	0	0

Tabla 1. Relación de los estados del rectificador analizado.

Modelo inicial del rectificador

Partiendo de la información de la Tabla 1 sabemos que V_{12} es una función de los estados interruptores, el signo de la corriente y las tensiones de los capacitores. Quedando la ecuación de la siguiente manera, la cual cumple con todos los estados de la tabla mencionada.

$$V_{12}(U_1, U_2, \text{Sign}(I_s), V_{C1}, V_{C2}) = \frac{\text{Sign}(I_s) + 1}{2} (V_{C1}(1 - S_1) + V_{C2}(1 - S_2)) - \frac{\text{Sign}(I_s) - 1}{2} (V_{C1}(U_2 - 1) + V_{C2}(U_1 - 1))$$

La tensión de la fuente V_s es la suma de las diferencias de potenciales del inductor L y V_{12} , se deprecia la caída de potencial de la resistencia R_s pues es insignificante comparada a las resistencias de carga R_1 y R_2 .

$$V_s = L \frac{di_L}{dt} + V_{12} \rightarrow L \frac{di_L}{dt} = -V_{12} + V_s$$

Ahora establecemos las ecuaciones de las corrientes según la Figura 10. Sentidos de las corrientes en el circuito rectificador.

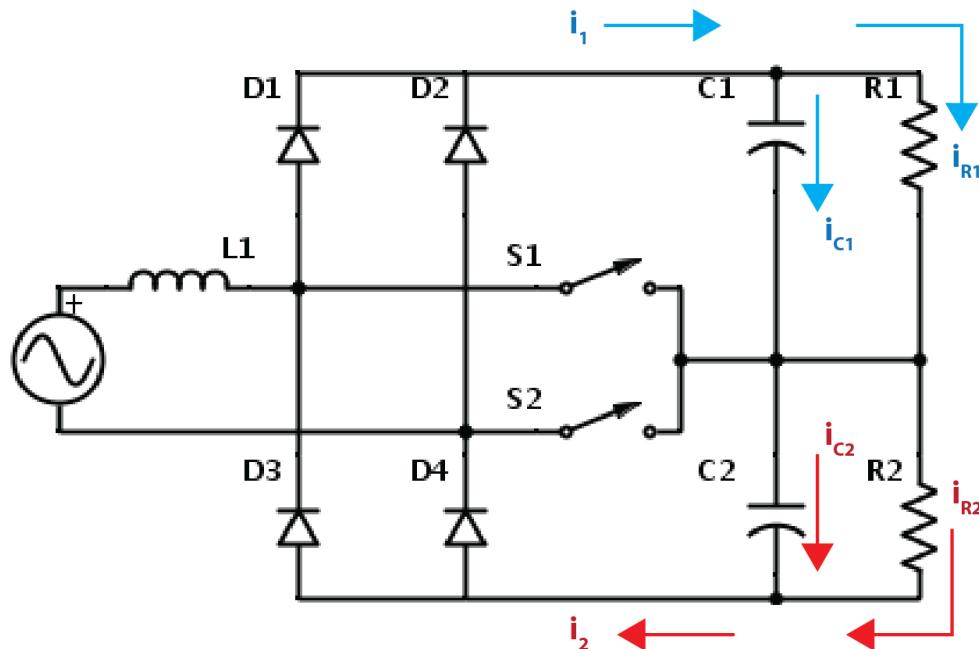


Figura 10. Sentidos de las corrientes en el circuito rectificador.

$$i_1 = i_{c1} + i_{R1}$$

$$i_{R1} = \frac{V_{c1}}{R_1}$$

$$i_2 = i_{c2} + i_{R2}$$

$$i_{R2} = \frac{V_{c2}}{R_2}$$

En base a las anteriores ecuaciones definimos las corrientes i_1 e i_2 , para que reflejen los valores de la Tabla 1.

$$i_1(i_s, \text{Sign}(i_s), S_1, S_2) = i_s \left(\left(\frac{\text{Sign}(i_s) + 1}{2} \right) (1 - S_1) + \left(\frac{\text{Sign}(i_s) - 1}{2} \right) (1 - S_2) \right)$$

$$i_2(i_s, \text{Sign}(i_s), S_1, S_2) = i_s \left(\left(\frac{\text{Sign}(i_s) + 1}{2} \right) (1 - S_2) + \left(\frac{\text{Sign}(i_s) - 1}{2} \right) (1 - S_1) \right)$$

Declaramos las nuevas variables

$$V_T = V_{C1} + V_{C2}$$

$$V_D = V_{C1} - V_{C2}$$

Siendo V_T el voltaje de salida deseado y V_D la diferencia de potencial deseado entre los capacitores.

Se hace un cambio de variable para que englobe el estado de los capacitores C_1 y C_2 , y al mismo tiempo deje las subsecuentes ecuaciones de tal manera que sean más sencillas.

$$u_1 = 1 - S_1 \quad V_{C1} = \frac{V_T + V_D}{2}$$

$$u_2 = 1 - S_2 \quad V_{C2} = \frac{V_T + V_D}{2}$$

Analizando el voltaje V_{12} haciendo las sustituciones con base a las variables establecidas anteriormente.

$$V_{12} = \frac{\text{Sign}(I_s) + 1}{2} \left(u_1 \frac{V_T + V_D}{2} + u_2 \frac{V_T - V_D}{2} \right) - \frac{\text{Sign}(I_s) - 1}{2} \left(u_2 \frac{V_T + V_D}{2} + u_1 \frac{V_T - V_D}{2} \right)$$

$$\begin{aligned} V_{12} &= \frac{1}{4} (\text{Sign}(I_s)(u_1(V_T + V_D) + u_2(V_T - V_D)) + (u_1(V_T + V_D) - u_2(V_T - V_D)) \\ &\quad + \text{Sign}(I_s)(u_1(V_T - V_D) + u_2(V_T + V_D)) + (-u_1(V_T - V_D) - u_2(V_T + V_D))) \end{aligned}$$

$$V_{12} = \frac{1}{4} (\text{Sign}(i_s)(u_1(2V_T) + u_2(2V_T)) + (u_1(2V_D) - u_2(2V_D)))$$

$$V_{12} = \frac{1}{2} (\text{Sign}(i_s)(u_1 + u_2)V_T + (u_1 - u_2)V_D)$$

Definiendo nuevas variables para simplificar la ecuación

$$\mu_1 = u_1 + u_2$$

$$\mu_2 = u_1 - u_2$$

Sustituyendo en la ecuación de la tensión V_{12} nos queda

$$V_{12} = \frac{1}{2} (Sign(i_s) \mu_1 V_T + \mu_2 V_D)$$

Modelo de la corriente

Para modelar completamente el circuito necesitamos establecer las ecuaciones de los capacitores y de la corriente consumida, que se muestra a continuación

$$L \frac{di_L}{dt} = -\frac{1}{2} Sign(i_s) \mu_1 V_T - \frac{1}{2} \mu_2 V_D + V_s$$

Establecemos que las corrientes a través de los capacitores son

$$C \frac{dV_T}{dt} = C \frac{dV_{C1}}{dt} + C \frac{dV_{C2}}{dt}$$

Considerando que las corrientes de cada uno de los capacitores son las siguientes

$$C_1 \frac{dV_{C1}}{dt} = i_1 - i_{R1} \quad C_2 \frac{dV_{C2}}{dt} = i_2 - i_{R2}$$

Como se ve en la Figura 8, ambos capacitores tienen el mismo valor de capacitancia y por el momento también se considerará valores de resistencias.

$$C = C_1 = C_2$$

$$R = R_1 = R_2$$

A continuación se deducen las ecuaciones de corriente que pasan por los capacitores C_1 y C_2

$$C \frac{dV_{C1}}{dt} = \frac{i_s}{2} ((Sign(i_s) + 1) u_1 + (Sign(i_s) - 1) u_2) - \frac{V_{C1}}{R}$$

$$C \frac{dV_{C1}}{dt} = \frac{i_s}{2} (\text{Sign}(i_s)(u_1 + u_2) + u_1 - u_2) - \frac{V_T + V_D}{2R}$$

$$C \frac{dV_{C1}}{dt} = \frac{i_s}{2} (\text{Sign}(i_s)\mu_1 + \mu_2) - \frac{V_T + V_D}{2R}$$

$$C \frac{dV_{C2}}{dt} = \frac{i_s}{2} ((\text{Sign}(i_s) + 1)u_2 + (\text{Sign}(i_s) - 1)u_1) - \frac{V_{C2}}{R}$$

$$C \frac{dV_{C2}}{dt} = \frac{i_s}{2} (\text{Sign}(i_s)(u_2 + u_1) + u_2 - u_1) - \frac{V_T - V_D}{2R}$$

$$C \frac{dV_{C2}}{dt} = \frac{i_s}{2} (\text{Sign}(i_s)\mu_1 - \mu_2) - \frac{V_T - V_D}{2R}$$

Ahora unificamos ambas ecuaciones para establecer la corriente que pasa por ambos capacitores

$$C \frac{dV_T}{dt} = \frac{i_s}{2} (\text{Sign}(i_s)\mu_1 + \mu_2) - \frac{V_T + V_D}{2R} + \frac{i_s}{2} (\text{Sign}(i_s)\mu_1 - \mu_2) - \frac{V_T - V_D}{2R}$$

$$C \frac{dV_T}{dt} = i_s \text{Sign}(i_s)\mu_1 - \frac{V_T}{R}$$

Y la diferencia de corriente entre los capacitores C_1 y C_2

$$C \frac{dV_D}{dt} = C \frac{dV_{C1}}{dt} - C \frac{dV_{C2}}{dt} = i_s \mu_2 - \frac{V_D}{R}$$

Consideración de diferentes resistencias

Debido que en la práctica las resistencias pueden no ser idénticas ya sea por el error de manufactura o por necesidades del diseño, se deben considerar cargas diferentes en las ecuaciones anteriores.

Regresando a las corrientes de los capacitores y considerando lo mencionado, tenemos que

$$C \frac{dV_{C1}}{dt} = -\frac{V_{C1}}{R_1} + \frac{1}{2} i_s ((\text{Sign}(i_s) + 1)u_1 + (\text{Sign}(i_s) - 1)u_2)$$

$$C \frac{dV_{C2}}{dt} = -\frac{V_{C2}}{R_2} + \frac{1}{2} i_s ((Sign(i_s) + 1)u_2 + (Sign(i_s) - 1)u_1)$$

Quedando así la corriente que pasa por ambos capacitores ($C \frac{dV_T}{dt}$) y la diferencia de corriente entre ellos ($C \frac{dV_D}{dt}$) de la siguiente forma

$$C \frac{dV_T}{dt} = -\frac{V_{C1}}{R_1} - \frac{V_{C2}}{R_2} + i_s Sign(i_s) \mu_1$$

$$C \frac{dV_D}{dt} = -\frac{V_{C1}}{R_1} + \frac{V_{C2}}{R_2} + i_s \mu_2 \dots (1)$$

Para simplificar las ecuaciones por venir, se harán unos cambios de variables tomando en cuenta los primeros dos términos de la penúltima ecuación

$$-\frac{V_{C1}}{R_1} - \frac{V_{C2}}{R_2} = -\frac{V_T + V_D}{2R_1} - \frac{V_T - V_D}{2R_2} = -V_T \left(\frac{1}{2R_1} + \frac{1}{2R_2} \right) - V_D \left(\frac{1}{2R_1} - \frac{1}{2R_2} \right)$$

Declarando nuevas variables que relacionan las cargas del circuito

$$G_1 = \left(\frac{1}{2R_1} + \frac{1}{2R_2} \right) \quad G_2 = \left(\frac{1}{2R_1} - \frac{1}{2R_2} \right)$$

Por lo tanto

$$-V_T \left(\frac{1}{2R_1} + \frac{1}{2R_2} \right) - V_D \left(\frac{1}{2R_1} - \frac{1}{2R_2} \right) = -V_T G_1 - V_D G_2$$

Ahora aplicando lo mismo a la ecuación (1):

$$-\frac{V_{C1}}{R_1} + \frac{V_{C2}}{R_2} = -\frac{V_T + V_D}{2R_1} + \frac{V_T - V_D}{2R_2}$$

$$= -V_T \left(\frac{1}{2R_1} - \frac{1}{2R_2} \right) - V_D \left(\frac{1}{2R_1} + \frac{1}{2R_2} \right) = -V_T G_2 - V_D G_1$$

Sustituyendo en las ecuaciones originales

$$C \frac{dV_T}{dt} = -V_T G_1 - V_D G_2 + i_s \text{Sign}(i_s) \mu_1$$

$$C \frac{dV_D}{dt} = -V_T G_2 - V_D G_1 + i_s \mu_2 \dots (2)$$

Controlador

El controlador del sistema propuesto tiene como objetivo alcanzar el voltaje deseado en el rectificador a través de los interruptores del mismo. Para lograr esto se hace usos de control por linealización, alta ganancia y proporcional-integral (PI). De manera general tendrá como entradas la tensión de entrada del rectificador, la tensión total actual de los capacitores, la diferencia de tensión entre capacitores, la corriente de salida del circuito, la tensión total deseada de los capacitores, la diferencia de tensión deseada entre capacitores y las G's del sistema (las cuales fueron explicadas anteriormente en este capítulo). Partiendo del modelo de la corriente y considerando que la corriente consumida $i_s = i_L$

$$L \frac{di_s}{dt} = -\frac{1}{2} \text{Sign}(i_s) V_T \mu_1 - \frac{1}{2} V_D \mu_2 + V_s$$

Definiendo el valor de la corriente deseada de salida como i_s^* t y siendo \tilde{i}_s el error de la corriente de salida tenemos que

$$i_s^* = L \frac{di_s^*}{dt} \quad \tilde{i}_s = i_s - i_s^*$$

Ahora se multiplica ambos lados de la ecuación por L y derivando en función del tiempo, con esta ecuación ahora podemos sustituir en la ecuación inicial de esta sección para obtener la ecuación de la corriente deseada

$$L \frac{d\tilde{i}_s}{dt} = L \frac{di_s}{dt} - L \frac{di_s^*}{dt}$$

$$L \frac{d\tilde{i}_s}{dt} = -\frac{1}{2} Sign(i_s) V_T \mu_1 - \frac{1}{2} V_D \mu_2 + V_S - L \frac{di_s^*}{dt} \dots (2)$$

Creamos una señal de control μ_1 que nos permita eliminar variables, siendo k una constante de proporcionalidad definida por el usuario que establecerá la velocidad del cambio del error de la corriente, el cual siempre debe disminuir tendiendo hacia cero. Se sustituye en la ecuación (2)

$$\mu_1 = \frac{2}{V_T} Sign(i_s) (V_S + k \tilde{i}_s)$$

$$L \frac{d\tilde{i}_s}{dt} = -Sign(i_s)^2 V_S - Sign(i_s)^2 k \tilde{i}_s - \frac{1}{2} V_D \mu_2 + V_S - L \frac{di_s^*}{dt}$$

$$\text{Si tenemos en cuenta que } Sign(i_s) = \begin{cases} 1, & i_s > 0 \\ 0, & i_s = 0 \\ -1, & i_s < 0 \end{cases}$$

Entonces podemos deducir que $Sign(i_s)^2$ será prácticamente 1 en la mayoría de las ocasiones, con lo cual podemos simplificar la ecuación de la corriente deseada.

$$L \frac{d\tilde{i}_s}{dt} = -Sign(i_s)^2 k \tilde{i}_s - V_S (Sign(i_s)^2 - 1) - \frac{1}{2} V_D \mu_2 - L \frac{di_s^*}{dt}$$

$$L \frac{d\tilde{i}_s}{dt} = -Sign(i_s)^2 k \tilde{i}_s - \frac{1}{2} V_D \mu_2 - L \frac{di_s^*}{dt}$$

Debemos recordar que siempre se busca que la diferencia de potenciales entre capacitores V_D sea aproximadamente igual a cero por lo que se vuelve despreciable si $k \gg 0$. Siendo $L \frac{di_s^*}{dt}$ la derivada de una sinusoidal, no debe perjudicar al error debido a que su promedio es cero. Ahora se sustituye μ_1 en la ecuación del modelo del voltaje de la suma de tensiones en los capacitores.

$$C \frac{dV_T}{dt} = -G_1 V_T - G_2 V_D + i_s Sign(i_s) \mu_1 = -G_1 V_T - G_2 V_D + i_s Sign(i_s) \left(\frac{2}{V_T} Sign(i_s) (V_s + k \tilde{i}_s) \right)$$

Recordando que $i_s = \tilde{i}_s + i_s^*$

$$C \frac{dV_T}{dt} = -G_1 V_T - G_2 V_D + (\tilde{i}_s + i_s^*) Sign(i_s) \left(\frac{2}{V_T} Sign(i_s) (V_s + k \tilde{i}_s) \right)$$

$$C \frac{dV_T}{dt} = -G_1 V_T - G_2 V_D + \frac{2 Sign(i_s)^2}{V_T} (k \tilde{i}_s^2 + k \tilde{i}_s i_s^* + \tilde{i}_s V_s + i_s^* V_s)$$

Considerando que \tilde{i}_s tiende a cero por ser el error de la corriente

$$k \tilde{i}_s^2 \rightarrow 0 \quad k \tilde{i}_s i_s^* \rightarrow 0 \quad \tilde{i}_s V_s \rightarrow 0$$

$$C \frac{dV_T}{dt} = -G_1 V_T - G_2 V_D + \frac{2 Sign(i_s)^2}{V_T} i_s^* V_s \dots (R1)$$

Siendo \tilde{V}_T el error del voltaje y V_T^* el voltaje total de salida deseado tenemos que $\tilde{V}_T = V_T - V_T^*$, se multiplican ambos lados de la ecuación y se deriva en función del tiempo, debido a que el voltaje deseado es una constante, su derivada es cero

$$C \frac{d\tilde{V}_T}{dt} = C \frac{dV_T}{dt} - C \frac{dV_T^*}{dt} \rightarrow C \frac{d\tilde{V}_T}{dt} = C \frac{dV_T}{dt}$$

$$C \frac{d\tilde{V}_T}{dt} = -G_1 V_T - G_2 V_D + \frac{2 \operatorname{Sign}(i_s)^2}{V_T} i_s^* V_s \dots (3)$$

Si el voltaje pico de la fuente V_{sp} tenemos que la corriente deseada es

$$i_s^* = \frac{-V_s V_T}{2 V_{sp}^2} (-G_1 V_T - G_2 V_D + k_{P1} \tilde{V}_T + K_{I1} \eta)$$

Siendo $\frac{d\eta}{dt} = \tilde{V}_T$ y sustituyendo i_s^* en la ecuación (3). Debe tenerse en cuenta que los términos $k_{P1} \tilde{V}_T$ y $K_{I1} \eta$ son parte del control PI, que funciona a través de retroalimentación de los parámetros calculando la desviación entre el valor medido y el valor deseado.

$$C \frac{d\tilde{V}_T}{dt} = -G_1 V_T - G_2 V_D + \operatorname{Sign}(i_s)^2 \frac{V_s^2}{V_{sp}^2} (-G_1 V_T - G_2 V_D + k_{P1} \tilde{V}_T + K_{I1} \eta)$$

$$C \frac{d\tilde{V}_T}{dt} = -G_1 V_T \left(1 - \operatorname{Sign}(i_s)^2 \frac{V_s^2}{V_{sp}^2}\right) - G_2 V_D \left(1 - \operatorname{Sign}(i_s)^2 \frac{V_s^2}{V_{sp}^2}\right) - \operatorname{Sign}(i_s)^2 \frac{V_s^2}{V_{sp}^2} (k_{P1} \tilde{V}_T + K_{I1} \eta)$$

Ahora hacemos el control a partir de la ecuación (2), definimos \tilde{V}_D el error de la diferencia de potencial entre los capacitores de salida

$$\tilde{V}_D = V_D - V_D^*$$

$$C \frac{dV_D}{dt} = -V_T G_2 - V_D G_1 + \tilde{i}_s \mu_2 + i_s^* \mu_2$$

$$C \frac{d\tilde{V}_D}{dt} = C \frac{dV_D}{dt} - C \frac{dV_D^*}{dt}$$

Debido a que la diferencia de potenciales entre capacitores tiende a cero tenemos que $\frac{dV_D^*}{dt} \rightarrow 0$ y $\tilde{i}_s \mu_2 \rightarrow 0$. Para normalizar creamos una nueva señal de control

$$\mu_2 = \frac{-i_s^*}{1 + i_s^{*2}} (-V_T G_2 - V_D G_1 + k_{P2} \tilde{V}_D + K_{I2} \varepsilon)$$

Quedando así

$$C \frac{d\tilde{V}_D}{dt} = -G_2 V_T \left(1 - \frac{i_s^{*2}}{1 + i_s^{*2}} \right) - G_1 V_D \left(1 - \frac{i_s^{*2}}{1 + i_s^{*2}} \right) - \frac{i_s^{*2}}{1 + i_s^{*2}} (k_{P2} \tilde{V}_D + K_{I2} \varepsilon)$$

Observador

Para hacer las estimaciones de las resistencias en tiempo de ejecución, ya sea durante simulación, emulación o aplicación; se usara un observador por inmersión e invariancia, normalmente utilizado para para sistemas generales no lineales de orden reducido. La noción de inmersión tiene una larga tradición en la teoría de control, su idea básica es proyectar el sistema en consideración dentro de un sistema con propiedades pre-especificadas, por ejemplo el problema clásico de inmersión de un sistema genérico no lineal hacia un sistema lineal y controlable por medio de retroalimentación de estados dinámicos o estáticos. La inmersión ha sido utilizada en la teoría de regularización no lineal para derivar las condiciones necesarias y suficientes para una regularización robusta, por ejemplo, este tipo de regularización es alcanzable siempre que el sistema pueda ser inmerso dentro de un sistema lineal y observable.

Para el sistema tratado en este escrito partimos que la salida Z del observador es el error de la estimación de las cargas, el vector que relaciona las cargas del rectificador G , las estimaciones de las mismas \hat{G} y β que es un término de corrección a definir.

$$Z = G - \hat{G} + \beta \dots (4)$$

Donde $G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}$ y $\hat{G} = \begin{bmatrix} \hat{G}_1 \\ \hat{G}_2 \end{bmatrix}$. Si se tiene en cuenta que $Z \rightarrow 0$ entonces las cargas del rectificar deben ser $G = \hat{G} - \beta$. Recordando las ecuaciones de corrientes del circuito

$$C \frac{dV_T}{dt} = -V_T G_1 - V_D G_2 + i_s \text{Sign}(i_s) \mu_1$$

$$C \frac{dV_D}{dt} = -V_T G_2 - V_D G_1 - i_s \text{Sign}(i_s) \mu_1$$

Ahora para hacer más cómoda la ecuación lo ponemos en forma matricial, para agrupar ambas en una sola ecuación debemos incluir los estados de los interruptores que están en función de las corrientes del sistema así como una matriz de las tensiones del mismo.

$$F(i_s) = \begin{bmatrix} i_s \text{Sign}(i_s) \mu_1 \\ i_s \mu_2 \end{bmatrix} \quad G_V = \begin{bmatrix} V_T & V_D \\ V_D & V_T \end{bmatrix}$$

De manera que nos quede una ecuación lineal de la corriente del sistema

$$C dV = F(i_s) - G_V G$$

Derivando la ecuación (4) respecto al tiempo

$$\dot{Z} = \dot{G} - \dot{\hat{G}} \rightarrow \dot{Z} = -\dot{\hat{G}} + \frac{d\beta}{dV} \dot{V}$$

Debido a que G es una constante $\dot{G} \rightarrow 0$.

$$\dot{Z} = -\dot{\hat{G}} + \frac{d\beta}{dV} \frac{1}{C} (F(i_s) - G_V G) = -\dot{\hat{G}} + \frac{d\beta}{dV} \frac{1}{C} (F(i_s) - G_V (Z + \hat{G} - \beta))$$

Reacomodando la ecuación para que quede de la forma $\dot{x} = kx$

$$\dot{Z} = -\frac{d\beta}{dV} \frac{1}{C} G_V Z - \dot{\hat{G}} + \frac{d\beta}{dV} \frac{1}{C} (F(i_s) - G_V (\hat{G} - \beta))$$

Para que quede de la forma requerida $-\dot{\hat{G}} + \frac{d\beta}{dV} \frac{1}{C} (F(i_s) - G_V (\hat{G} - \beta)) = 0$.

$$\dot{\hat{G}} = \frac{d\beta}{dV} \frac{1}{C} (F(i_s) - G_V (\hat{G} - \beta))$$

Quedando así

$$\dot{Z} = -\frac{d\beta}{dV} \frac{1}{C} G_V Z$$

Debe ser positivo para que todo quede negativo, se propone una $\frac{d\beta}{dV}$

$$\frac{d\beta}{dV} = C \Gamma G_V$$

$$\Gamma = \begin{bmatrix} \Gamma_1 & 0 \\ 0 & \Gamma_2 \end{bmatrix}$$

$$\dot{Z} = - \begin{bmatrix} \Gamma_1 & 0 \\ 0 & \Gamma_2 \end{bmatrix} \begin{bmatrix} V_T^2 + V_D^2 & 2V_T V_D \\ 2V_T V_D & V_T^2 + V_D^2 \end{bmatrix} Z$$

$$\frac{d\beta}{dV} = C \begin{bmatrix} \Gamma_1 & 0 \\ 0 & \Gamma_2 \end{bmatrix} \begin{bmatrix} V_T & V_D \\ V_D & V_T \end{bmatrix}$$

Si sabemos que $\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$, $x = V = \begin{bmatrix} V_T \\ V_D \end{bmatrix}$

$$\frac{d\beta}{dx} = \begin{bmatrix} \frac{d\beta_1}{dx_1} & \frac{d\beta_1}{dx_2} \\ \frac{d\beta_2}{dx_1} & \frac{d\beta_2}{dx_2} \end{bmatrix} \rightarrow \beta = C \begin{bmatrix} \Gamma_1 & 0 \\ 0 & \Gamma_2 \end{bmatrix} \begin{bmatrix} \frac{1}{2}(V_T^2 + V_D^2) \\ V_T V_D \end{bmatrix}$$

Quedando la salida del observador de la siguiente forma:

$$\dot{\hat{G}} = \frac{d\beta}{dV} \frac{1}{C} \left(F(i_s) - G_V(\hat{G} - \beta) \right)$$

Modelo completo del sistema

El modelo completo del sistema se puede observar en la Figura 11. Diagrama de bloques matemáticos., las entradas del sistema son la tensión de alimentación, la tensión total de los capacitores, la diferencia de potencial entre capacitores además de los valores de los componentes del circuito rectificador. El controlador tendrá como salidas las S's descritas anteriormente que son los estados de los interruptores que sirven como alimentación para el rectificador y el observador. El observador está a cargo de realizar las estimaciones de las G's del sistema que servirán como entrada al controlador para que este pueda tomar las medidas adecuadas y a través de la conmutación de los estados de los interruptores obtener los valores de tensión deseados.

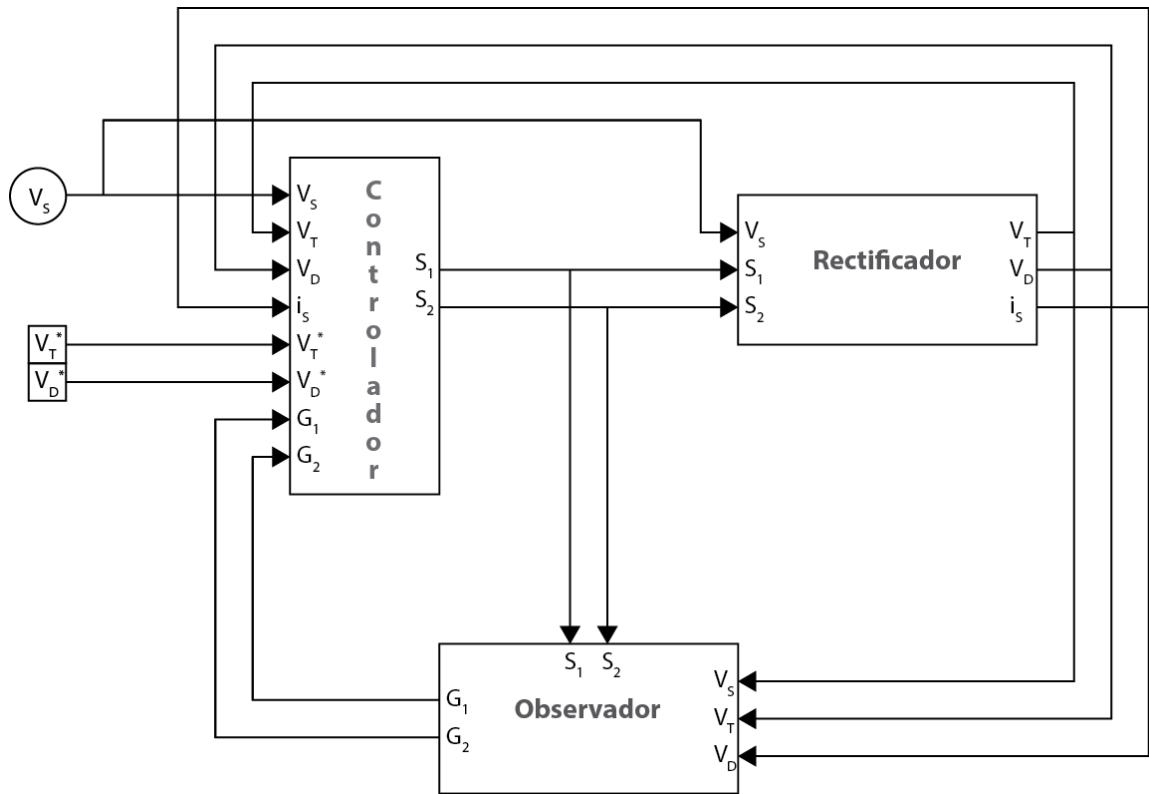


Figura 11. Diagrama de bloques matemáticos.

Conversión a MATLAB

En la Tabla 2 se puede observar un resumen de las ecuaciones necesarias del modelo del rectificador, el controlador y el observador para el algoritmo en el lenguaje de MATLAB.

Rectificador	
Descripción	Ecuación
Corriente de los capacitores	$C \frac{dV_T}{dt} = -V_T G_1 - V_D G_2 + i_s \text{Sign}(i_s) \mu_1$
Diferencia de corrientes entre los capacitores	$C \frac{dV_D}{dt} = -V_T G_2 - V_D G_1 + i_s \mu_2$
Corriente de salida	$L \frac{di_L}{dt} = -\frac{1}{2} \text{Sign}(i_s) \mu_1 V_T - \frac{1}{2} \mu_2 V_D + V_s$
Controlador	
Corriente de salida deseada	$i_s^* = \frac{-V_s V_T}{2 V_{sp}^2} (-G_1 V_T - G_2 V_D + k_{P1} \tilde{V}_T + K_{I1} \eta)$

Error de la tensión de salida	$\tilde{V}_T = V_T - V_T^*$
Error de la corriente de salida	$\tilde{i}_s = i_s - i_s^*$
Error de la tensión de la diferencia de tensión	$\tilde{V}_D = V_D - V_D^*$
Interruptor 1	$\mu_1 = \frac{2}{V_T} \text{Sign}(i_s)(V_s + k \tilde{i}_s)$
Interruptor 2	$\mu_2 = \frac{-i_s^*}{1 + i_s^{*2}} (-V_T G_2 - V_D G_1 + k_{P2} \tilde{V}_D + K_{I2} \varepsilon)$
Observador	
Derivada del valor estimado de las resistencias	$\dot{\hat{G}} = \frac{d\beta}{dV} \frac{1}{C} (F(i_s) - G_V(\hat{G} - \beta))$
Error del valor estimado de las resistencias	$\beta = C \begin{bmatrix} \Gamma_1 & 0 \\ 0 & \Gamma_2 \end{bmatrix} \begin{bmatrix} \frac{1}{2}(V_T^2 + V_D^2) \\ V_T V_D \end{bmatrix}$

Tabla 2. Resumen de las ecuaciones del modelo matemático.

Ahora con que se tienen las traducciones de las ecuaciones más importantes, se puede completar las partes intermedias del algoritmo necesarias para que funcione adecuadamente. Con esto se crearon tres componentes principales, como se ve en la Figura 12. Simulación en MATLAB Simulink basado en el modelo matemático del sistema., los cuales contienen el rectificador, controlador y observador como se pueden ver en los bloques mayores del centro. También se tuvieron que hacer bloques personalizados para la integración. Este algoritmo toma sus valores deseados y valores de inicio del bloque mayor a la izquierda.

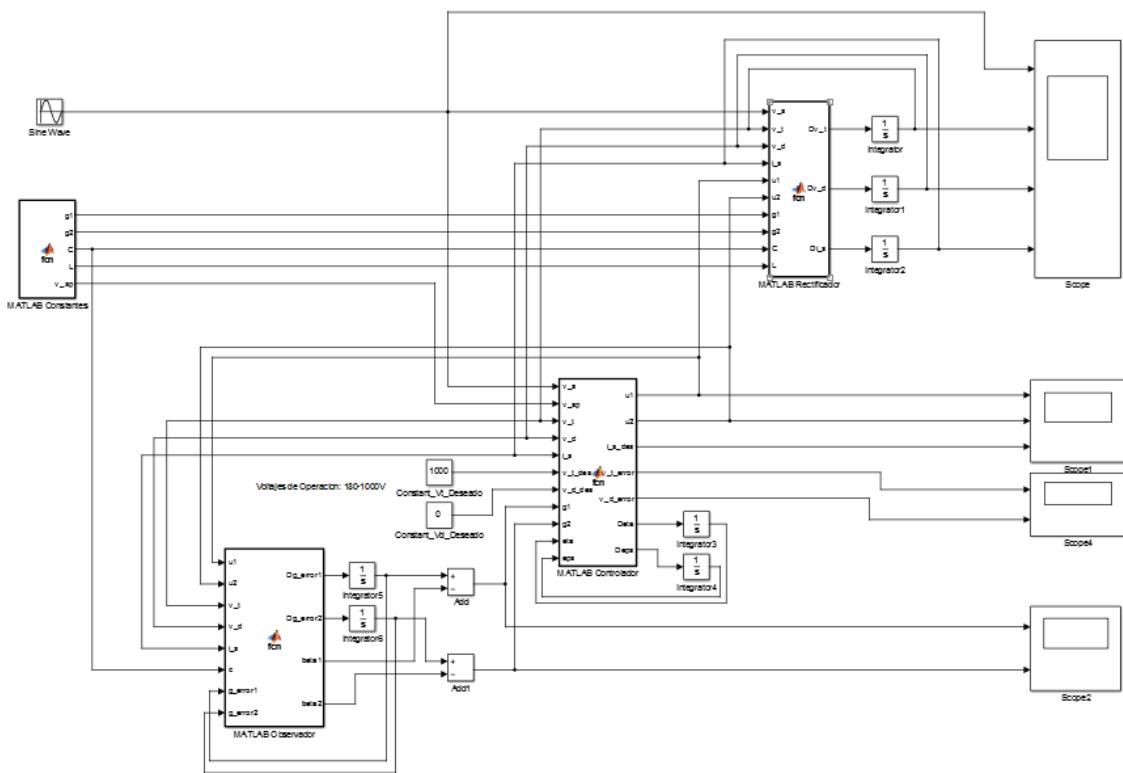


Figura 12. Simulación en MATLAB Simulink basado en el modelo matemático del sistema.

Resultados

Para los efectos de comprobación se utilizará la siguiente configuración que se muestra en la Tabla 2, existen algunos detalles adicionales para los componentes integradores que no se tratarán pues estos variaran de modelo a modelo y según el algoritmo utilizado.

Descripción	Valor
Resistencia R_1	100Ω
Resistencia R_2	100Ω
Voltaje pico V_{SP}	180V
Capacitancia C	$470\mu F$
Inductancia L	10mH
Voltaje total deseado V_T	500V
Diferencia de voltaje deseado V_D	0V

Tabla 3. Valores de prueba del algoritmo.

En Figura 13 se puede observar, la alimentación del circuito que en este caso es la alimentación monofásica 127VAC 60Hz; las salidas que son la suma de las tensiones de los capacitores que se aproxima a 500VDC, la diferencia de potencial entre capacitores que tiende a 0VDC y la corriente consumida que conforme se alcanza la tensión de salida deseada tiende a ser una sinusoidal cuasi perfecta. Es importante mencionar la forma de la corriente de entrada pues esto define la distorsión armónica total, la cual se desea que tienda a cero. La Figura 14, Figura 15 y Figura 16 muestran las comparaciones de los valores obtenidos y los deseados de la tensión total de salida, la diferencia de total entre capacitores y la corriente consumida, respectivamente. Como se puede observar el modelo del sistema propuesto es correcto pues los valores obtenidos se aproximan a los deseados por lo que se puede hacer su portación a código C.

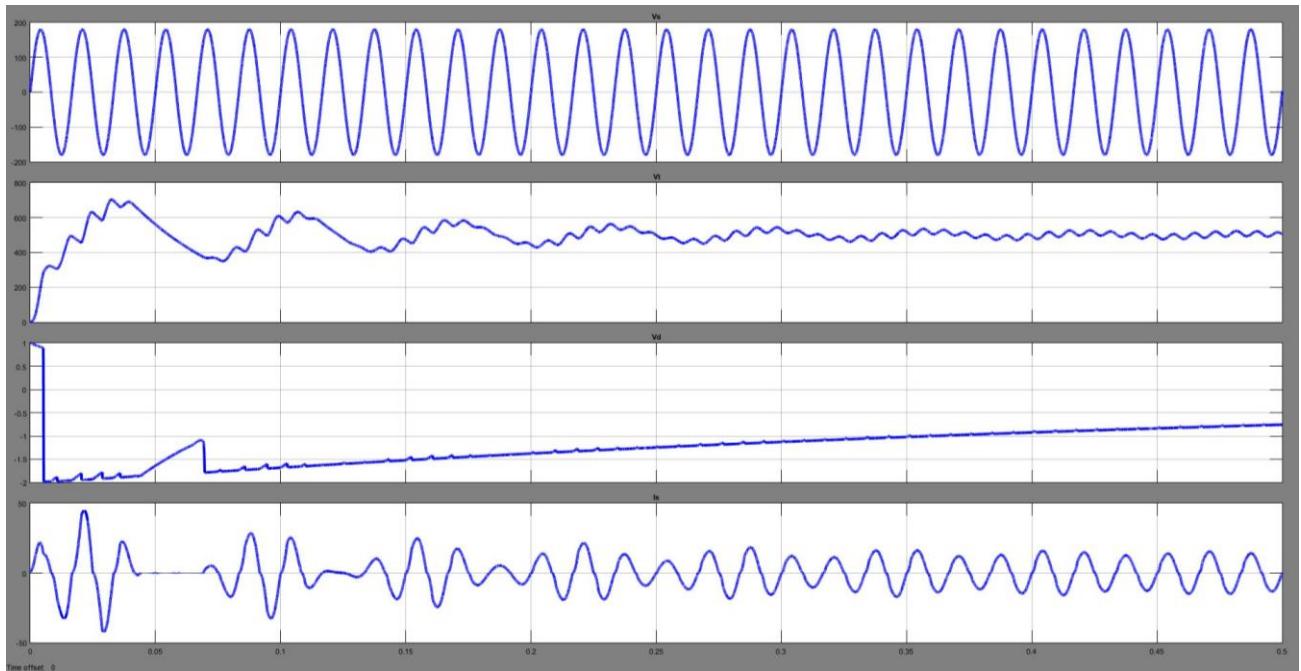


Figura 13. Resumen de las salidas (V_T segunda gráfica, V_D tercera y I_S abajo) del rectificador basados en una alimentación de 127 VAC.

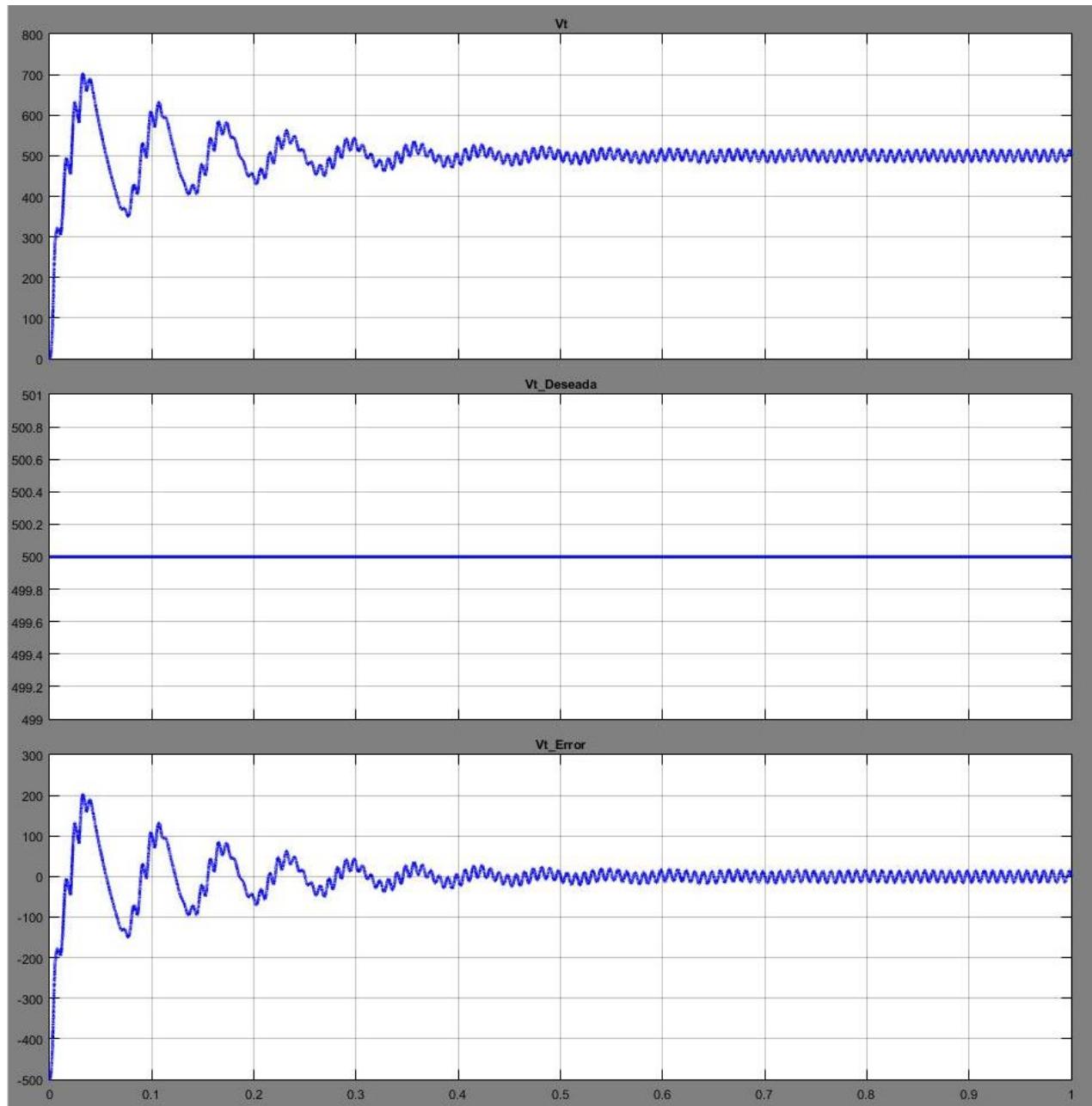


Figura 14. Comparación de las tensiones totales de salida VT, arriba la salida del sistema, en medio la tensión deseada y abajo el error de la tensión analizada.

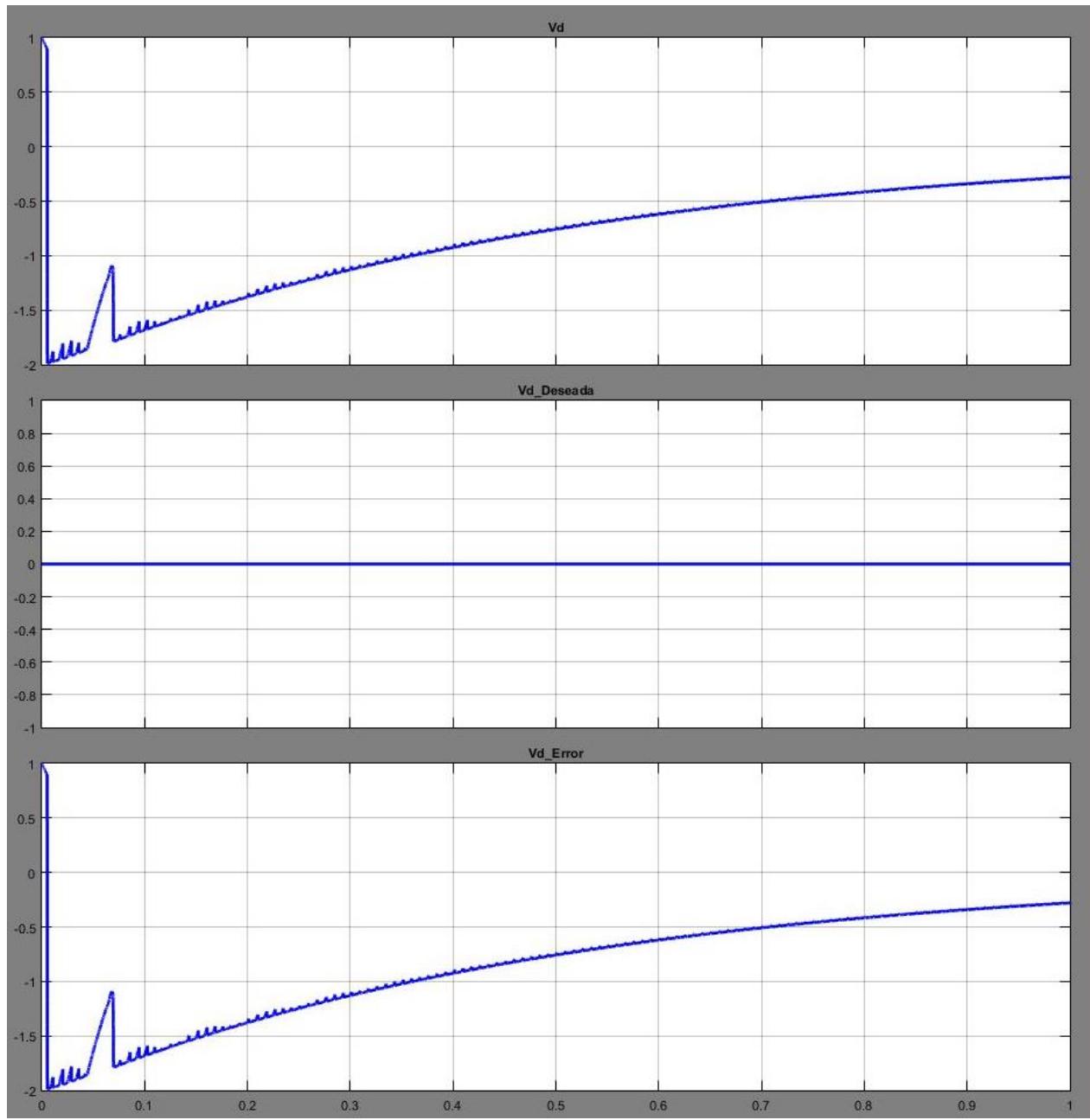


Figura 15. Comparación de las diferencia de tensiones entre los capacitores VD, arriba la salida del sistema, en medio el valor deseado y abajo el error analizado.

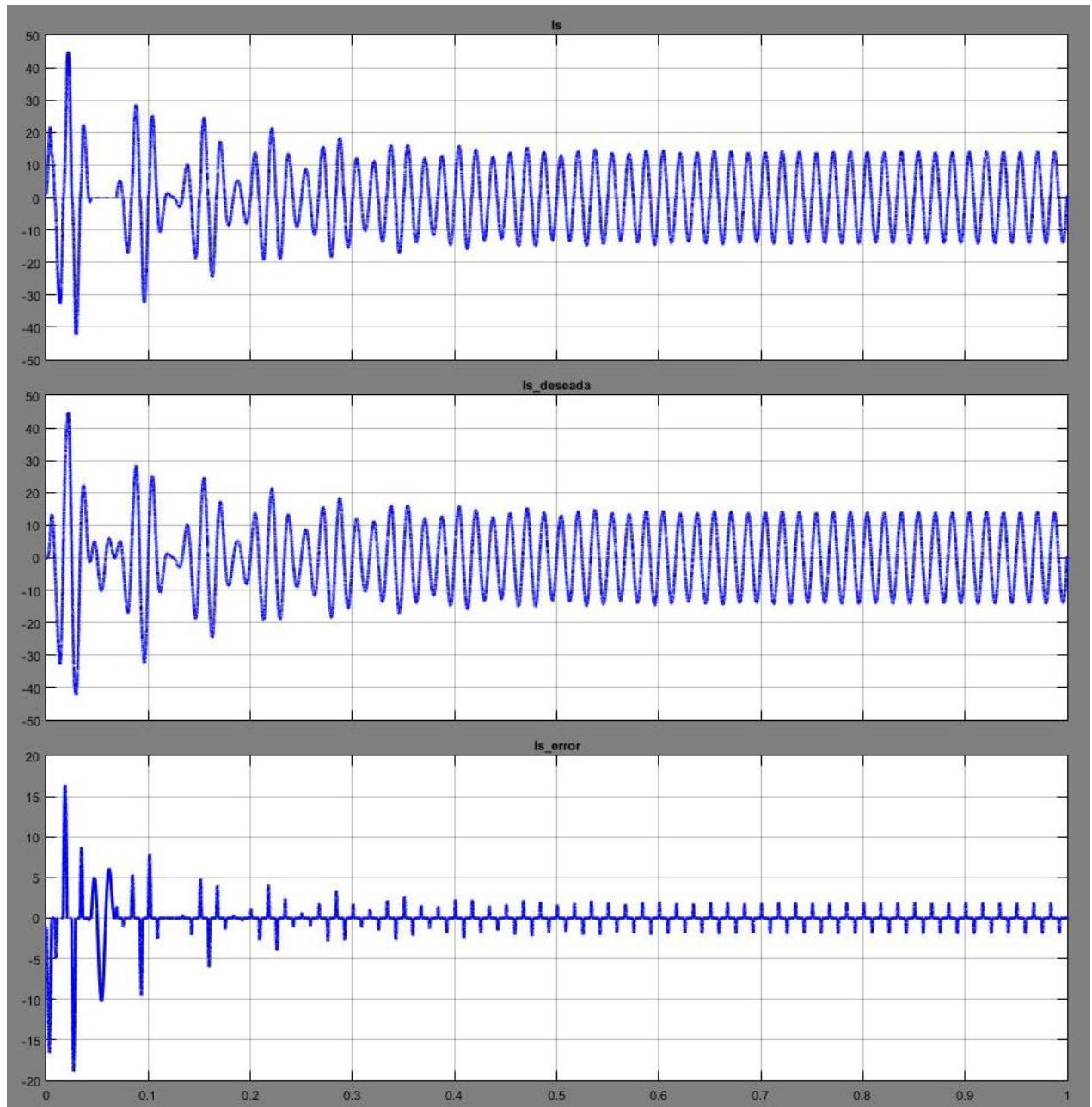


Figura 16. Comparación de la corriente consumida, arriba la salida del sistema, en medio la forma de la corriente deseada y abajo el error analizado.

Capítulo II. Herramienta de diseño: Simulador del modelo del rectificador activo.

Debido a que el código de MATLAB no se puede utilizar directamente en el microcontrolador dadas las diferencias entre los lenguajes de estos, es necesario implementar un paso intermedio que permita utilizar el mismo código de lenguaje C tanto en la PC como en el microcontrolador y así agilizar el proceso de desarrollo y depuración, es por eso que se decidió crear un simulador y el ambiente adecuado de este para ser probado.

Este simulador, como muchos otros, tiene como objetivo reducir los tiempos y por lo tanto los costos de desarrollo, ya que sirven como herramientas que permiten probar diferentes configuraciones de hardware (en caso de los simuladores de circuitos electrónicos) o diferentes versiones de código, como por ejemplo los simuladores de lenguaje VHDL para FPGAs que permiten ver las señales de salida de cada componente y del proyecto. Al igual que los simuladores de VHDL, el simulador tratado aquí nos permite ver las señales de salida y las intermedias o de control; esto permitirá analizar el comportamiento de una versión de código en específico para buscar posibles errores o para verificar su estado de funcionamiento y poder avanzar a la siguiente etapa de producción. Este tipo metodología de trabajo es importante en fuentes de poder, ya que si la circuitería fue diseñada pobremente puede resultar riesgoso al momento de realizar las pruebas en laboratorio, y especialmente si el control es meramente por software y no por hardware como las fuentes de poder comutadas actuales, que podrían hacer que los componentes funcionen incorrectamente o de una manera no deseada, este tipo de comportamiento conlleva riesgos de explosión, incendio o cortos circuitos, según sea el caso.

Traducción de código de MATLAB a lenguaje C

Debido a que el código MATLAB, en específico las variables que funcionan como matrices, el código no es directamente utilizable en C para su uso en PC o en un microcontrolador ya que MATLAB tiene embebido el código para la matemática de matrices mientras que esta funcionalidad no está soportada por defecto en C, por lo que en la traducción de esta funcionalidad se debe añadir lo que provoca grandes cambios entre ambos códigos.

Además de las diferencias en el lenguaje el código de C debe ser escrito en tal forma que solo exista una función a la que tenga que llamar el simulador y que esta pueda ser llamada enésimas veces. Esta función deberá contener las llamadas a los modelos del rectificador, controlador y observador. Para crear el DLL se utilizó el entorno de desarrollo Visual Studio 2013, que es una herramienta gratuita para el desarrollo de software.

El método que *Rectifier* es una representación en código C del modelo matemático del rectificador a utilizar, por lo tanto este toma como parámetros las variables necesarias para funcionar como son la tensión monofásica, los diferencia de potencial total y entre capacitores deseado, así como los valores de capacitancia, inductancia y las variables asociadas a las cargas; en base al valor instantáneo de estos devolverá como valores de salida las derivadas de la suma de las tensiones de los capacitores, el diferencial de tensiones entre capacitores y la corriente consumida.

El método *Controller* es la encargada de aplicar los diferentes tipos de controladores explicados en el capítulo anterior y devolver los valores de los errores de la corriente consumida, de la diferencia de tensión entre capacitores y la diferencia de potencial de la suma de los capacitores, el cual es nuestra salida principal, para poder ser utilizados por retroalimentación en el siguiente ciclo del algoritmo.

Cabe mencionar que este método también establece los valores que tendrán los interruptores S1 y S2 los cuales establecen el estado abierto o cerrado de estos.

La última pieza importante del algoritmo es el método *Observer*, que se encarga de hacer las estimaciones de las cargas en el rectificador. Aunque únicamente para propósitos de simulación este no parezca tener relevancia, es de notarse que la indispensabilidad de su prueba con los demás métodos del algoritmo para asegurar que funcione tal y como se desea una vez que este se exporte al microcontrolador.

Estas funciones se unifican bajo el método *SteppedSimulation* que será el que exportara el DLL para poder interactuar con la interfaz del simulador en la PC, además de realizar las llamadas a las funciones mencionadas también realiza las integraciones de las variables correspondientes para poder realizar la retroalimentación. En la Figura 17 se puede observar las interacciones de los diferentes métodos en el diagrama de flujo.

Cabe mencionar que todas las funciones utilizan parámetros de paso por referencia, esto es así para facilitar la visualización de estas en la interface del simulador en la PC. Claro esto conlleva algunas desventajas a nivel programático que se explicaran en las siguientes secciones.

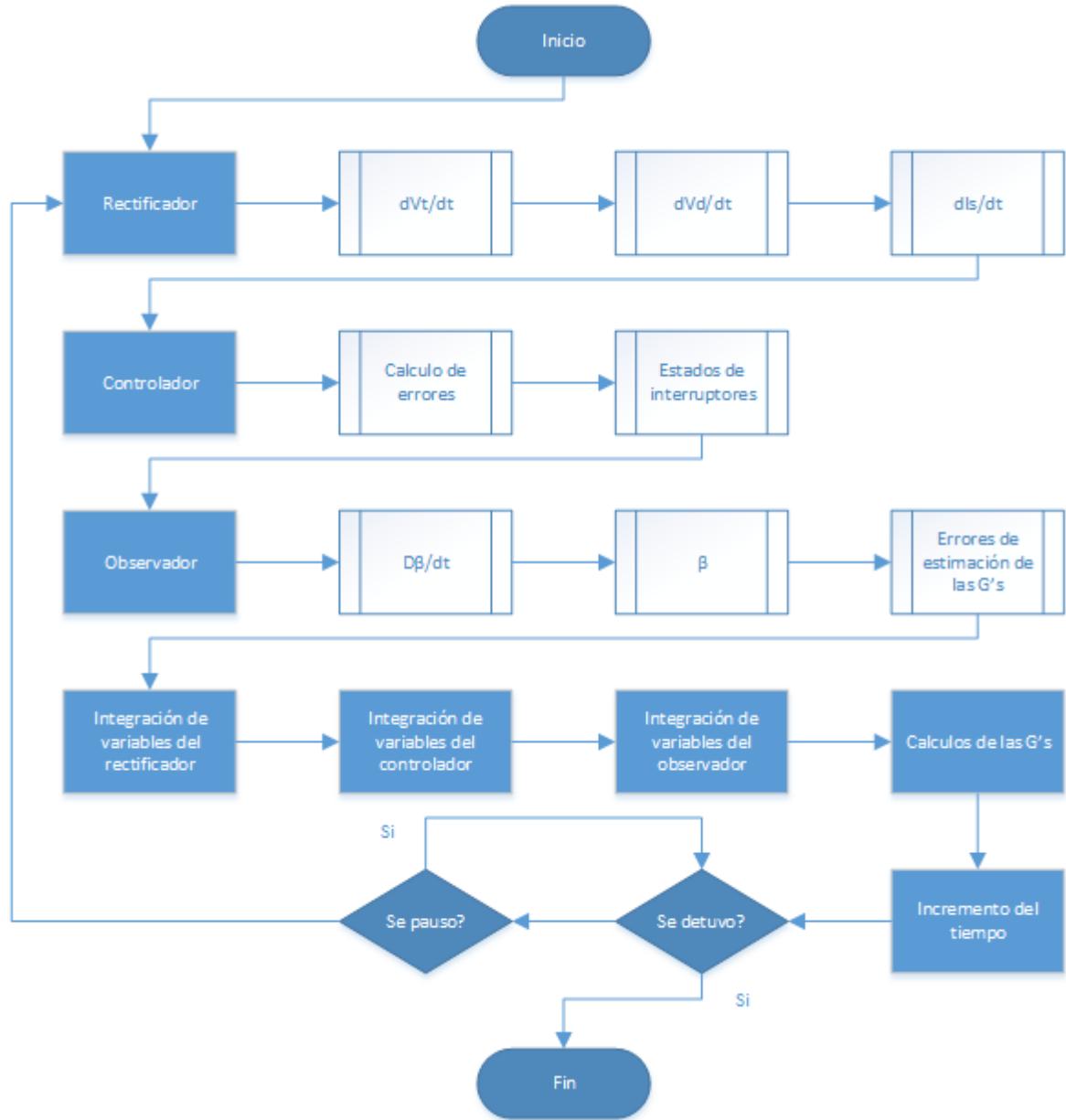


Figura 17. Diagrama de flujo del DLL.

Interfaz del simulador

Esta interfaz tiene el objetivo de facilitar el uso de diferentes algoritmos destinados para usar en Microcontroladores programados en estándar C y acelerar el proceso de desarrollo de nuevas aplicaciones.

La interfaz de simulación en la PC (al igual que la interfaz de emulación para el microcontrolador, de la cual se hablará en el capítulo 3) fue programada en el lenguaje C# (versión 5.0) con el .NET Framework 4.5.1 a través del modelo de trabajo Windows que será utilizado para crear las diferentes interfaces de usuario para la utilización de la aplicación. Se divide en cinco principales secciones, en la parte superior podemos ver la barra de menú en la cual podemos crear, guardar o abrir proyectos así como ver información adicional acerca del programa. A la izquierda del programa podemos ver la gráfica que se crea con los datos recibidos del DLL. En el centro podemos ver las pestañas de *Simulador* e *Interfaz a emulador* (por el momento nos enfocaremos únicamente a la simulación) esta permite iniciar, pausar o detener la simulación y también monitorear cualquiera de las señales que se usen como parámetro en la función del DLL, además de poder cambiarles el valor. En la parte derecha del programa se observa los controles comunes, utilizados tanto en la simulación como la emulación, aquí se controla la forma en la que se muestra las señales en gráfica de manera parecida, aunque simplificada, a un osciloscopio.

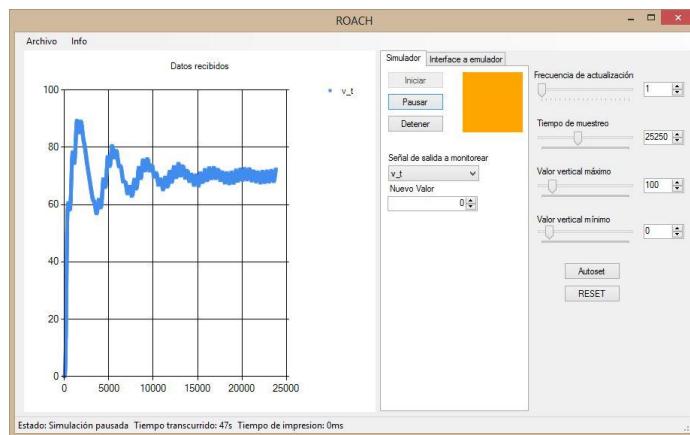


Figura 18. Captura de pantalla de la interfaz de PC

Simulador

El inicio de un proyecto de simulación en el programa de la PC empieza con dos archivos: uno contendrá los nombres y valores iniciales de los parámetros utilizados en el método exportado, y el propio DLL en C. Aparte de los archivos es necesario que es el nombre de la función del DLL y el nombre del proyecto, con esta información se puede crear un nuevo proyecto en el programa. Al guardar el proyecto se creara un XML en la dirección especificada con el cual se podrá cargar el proyecto completamente y continuar en la etapa que se quedó del desarrollo.

El archivo con los parámetros se puede crear en cualquier programa de hojas de cálculo y guardarlo como *.CSV, la Figura 19 muestra la ubicación de las columnas en la hoja, en la primera columna están los nombres de las variables y la siguiente contiene el valor inicial (o fijo, según sea el caso), cabe mencionar que el orden en que se ingresen los valores debe coincidir con el utilizado en el método del DLL y será el mismo orden que se mostrara en el *combobox* que permite escoger la señal a graficar en la interfaz.

A	B
1 eval_time	0
2 timeStep	1.67E-05
3 vt_des	70
4 vd_des	0
5 v_s	0
6 vs_amplit	34
7 vs_freque	377
8 r1	100
9 r2	100
10 C	4.70E-04
11 L	1.00E-02

Figura 19. Archivo *.CSV con los nombres y valores de los parámetros.

Una vez creado (o cargado el proyecto) se inicia la simulación en el botón respectivo, el proceso empieza creando e inicializando los hilos de comunicación con el DLL, impresión de la gráfica e impresión de texto que se ejecutarán en segundo plano. Incluyendo los hilos estándar que se crean en el *Windows Forms* inicial el programa contiene siete hilos, pero para propósitos de simplicidad solo se explicarán cuatro de ellos, tres de ellos fueron nombrados anteriormente solo faltando el hilo original del propio *Windows Forms*.

Debido a la arquitectura del programa primero se empieza con el hilo de comunicación hacia el DLL, pero antes de entrar en este tema debe comprenderse el sistema de carga del DLL en tiempo de ejecución. Todo inicia con la llamada a la función de la clase *LinkerToUnmanagedLibrary* encargada de la lógica de llamadas a través de los tres delegados correspondientes que permiten el paso por referencia de valores de punto flotante de 25, 50 o hasta 75 parámetros a una función no administrada, esta es aquella que se encuentra fuera del dominio del CLR (*Common Language Runtime* por sus siglas en inglés) donde reside C# pero no así el DLL en C. Con el archivo *.CSV que se utilizó para

crear el proyecto, el programa automáticamente habrá leído los nombres de las variables, y desplegado en el *combobox* de la interfaz, los valores de inicio que se pasó previamente al inicializar la clase serán guardados en un arreglo cuyos elementos serán utilizados como parámetros para el delegado correspondiente, el cual fue elegido mediante la cuenta de la filas en el archivo mencionado, este arreglo mantendrá siempre los valores actualizados de la simulación y serán accesibles para su impresión en la gráfica a través del *combobox* mencionado.

Una vez seleccionado el delegado correspondiente se hace la llamada la función de la clase *UnmanagedLibrary*, responsable de la carga dinámica del DLL en C, para a su vez llamar al método no administrado exportado por el usuario. Este proceso empieza obteniendo el puntero a la función deseada, téngase en cuenta que el DLL ya fue cargado en memoria durante el tiempo de ejecución automáticamente en la creación o carga del proyecto, a través de un método contenido en kernel32.dll de Windows (esta biblioteca de enlace dinámico se encuentra instalada por defecto en todas las versiones del sistema operativo) encargado del manejo de memoria, operaciones de entrada/salida, creación de hilos y procesos, y sincronización de funciones; se verifica que el puntero no sea nulo, algo común en código adaptativo, de no serlo se convierte el puntero a un delegado apropiado a la cantidad de parámetros establecidos anteriormente para que sea utilizable por C# y el programa, y se devuelve la llamada para que pueda ser utilizado como cualquier método nativo.

Regresando al hilo de comunicación con el DLL, se llama al delegado que contiene la función exportada y se devuelven a través de los parámetros (tómese en cuenta que todos estos son por paso por referencia) y se guarda el valor deseado, seleccionado en el *combobox* correspondiente, en una fila concurrente. La ventaja de usar una fila concurrente es que permite acceder a ella de manera

segura desde varios hilos al mismo tiempo permitiendo que cuando se haya guardado exitosamente un valor pueda leerse desde otro hilo para su impresión en la gráfica.

El hilo de impresión es relativamente sencillo comparado al hilo anterior, una vez que se detecta un dato en la fila concurrente se llama al método *PrintConcurrent* que toma como parámetros el componente de la gráfica, la propia fila y el rango máximo a imprimir establecido en los controles comunes de la interfaz. Inicia invocando al componente de la gráfica en su hilo original (hilo del *Windows Forms*) una vez ahí se detecta si la cantidad de puntos impresos supera el rango establecido, de ser así se borra los puntos de la gráfica, sino se saca el dato de la fila y se imprime en la gráfica con el método de impresión de puntos rápidos establecidos por defecto en el componente mencionado.

El tercer hilo es el encargado de la impresión de texto en la barra de estado en la parte inferior de la ventana y únicamente escribe el estado actual del programa, tiempo transcurrido de la simulación así como el tiempo en que tarda en imprimirse una pantalla completa de la gráfica.

Por último el hilo del *Windows Forms* se encarga de manejar todo aquello con que el usuario interactúa o puede observar, como las acciones de los controles, menús y ventanas de proyecto, carga o guardado.

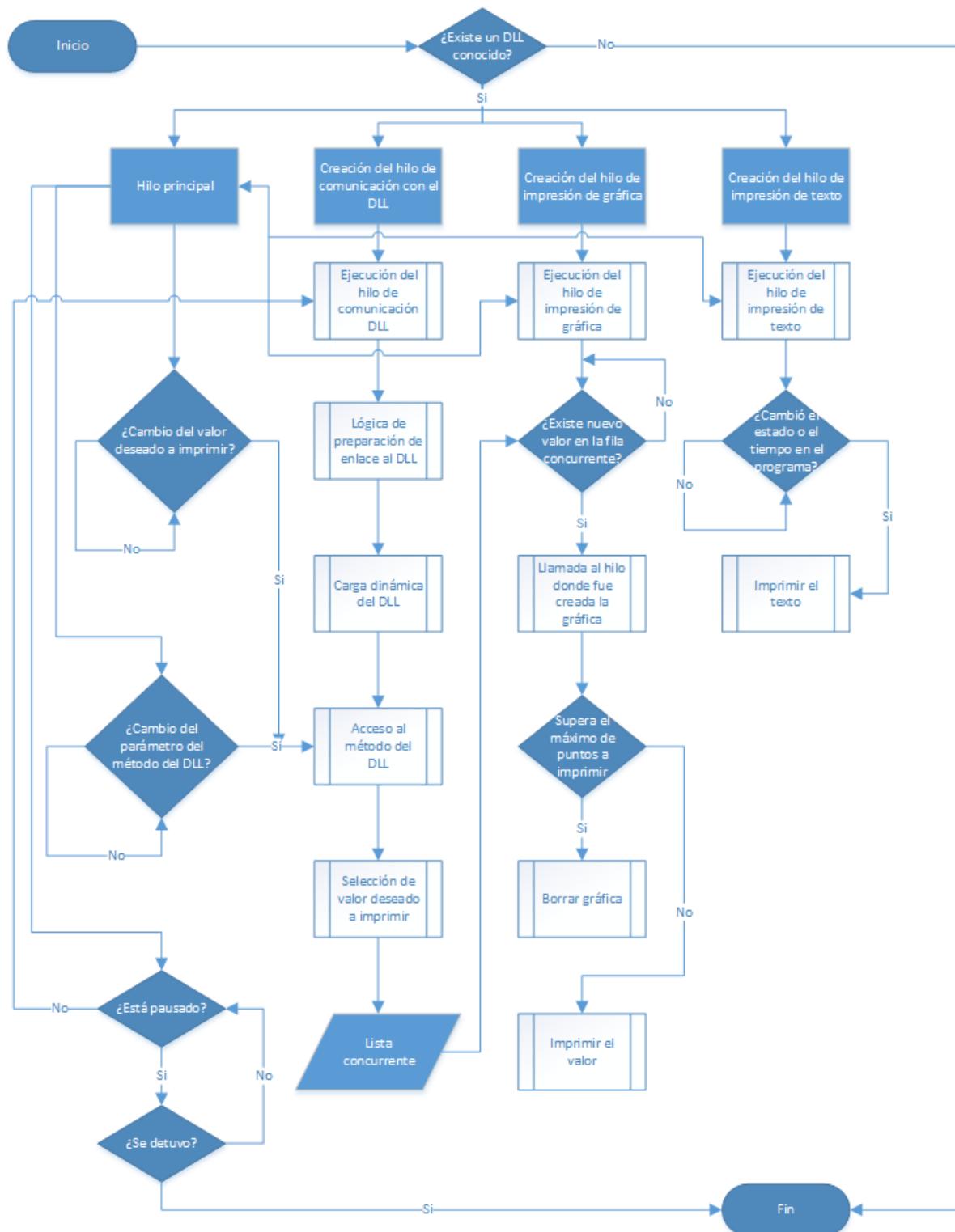


Figura 20. Diagrama de flujo del simulador en la interfaz de PC.

Resultados

En la Figura 21, Figura 22 y Figura 23 los resultados obtenidos en la simulación con la implementación de algoritmo en el DLL, se puede observar prácticamente los mismos resultados que los obtenidos al final del capítulo anterior con la única diferencia siendo las gráficas que involucran a la diferencia de tensión entre los capacitores V_D que en esta ocasión tuvo un tiempo de respuesta inmediato al valor deseado. Como el resultado obtenido de la portación del código de MATLAB a lenguaje C fue satisfactorio, se puede proceder con confianza a su implementación en el microcontrolador deseado.

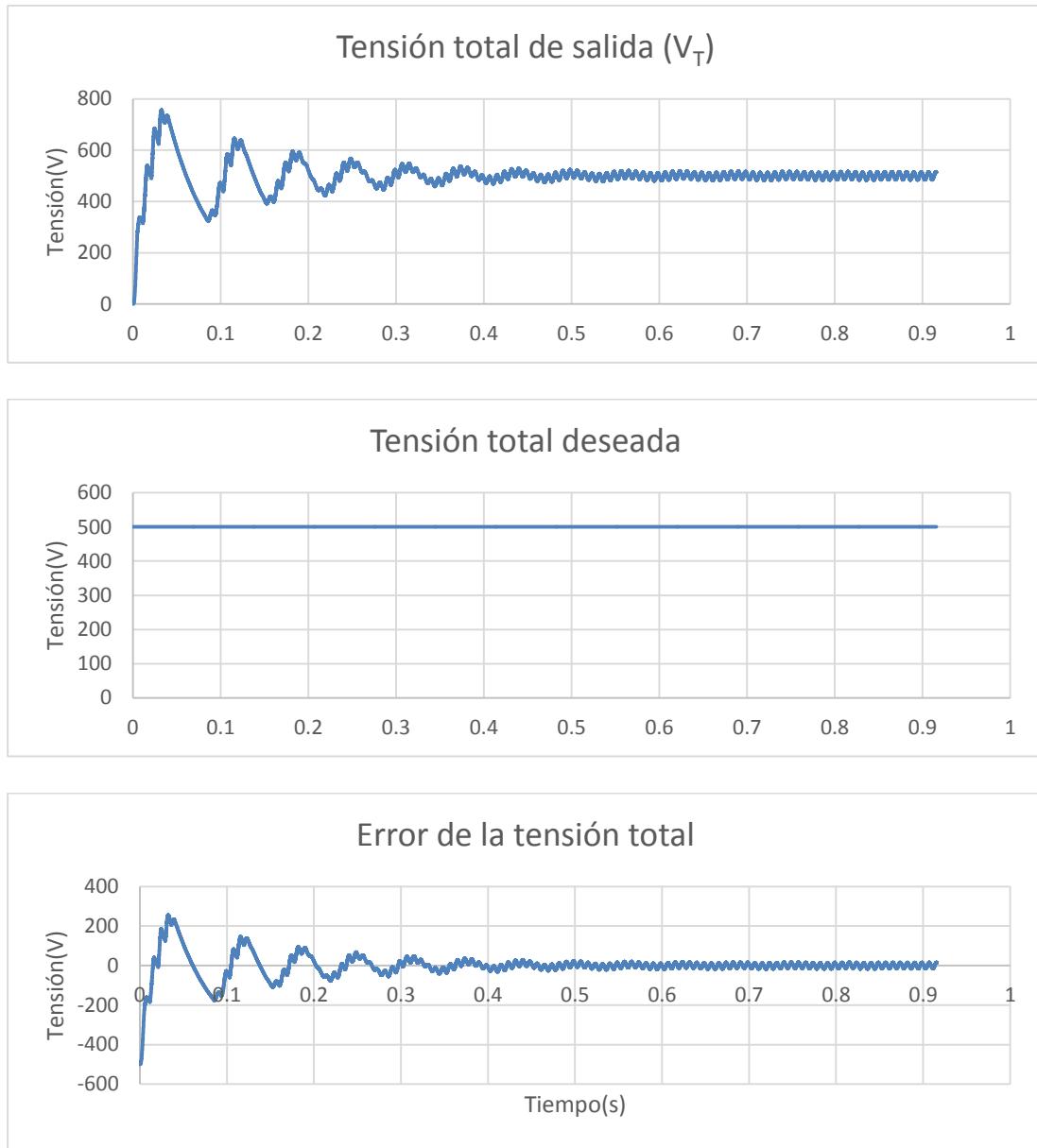


Figura 21. Resultados la tensión total de salida en el simulador.

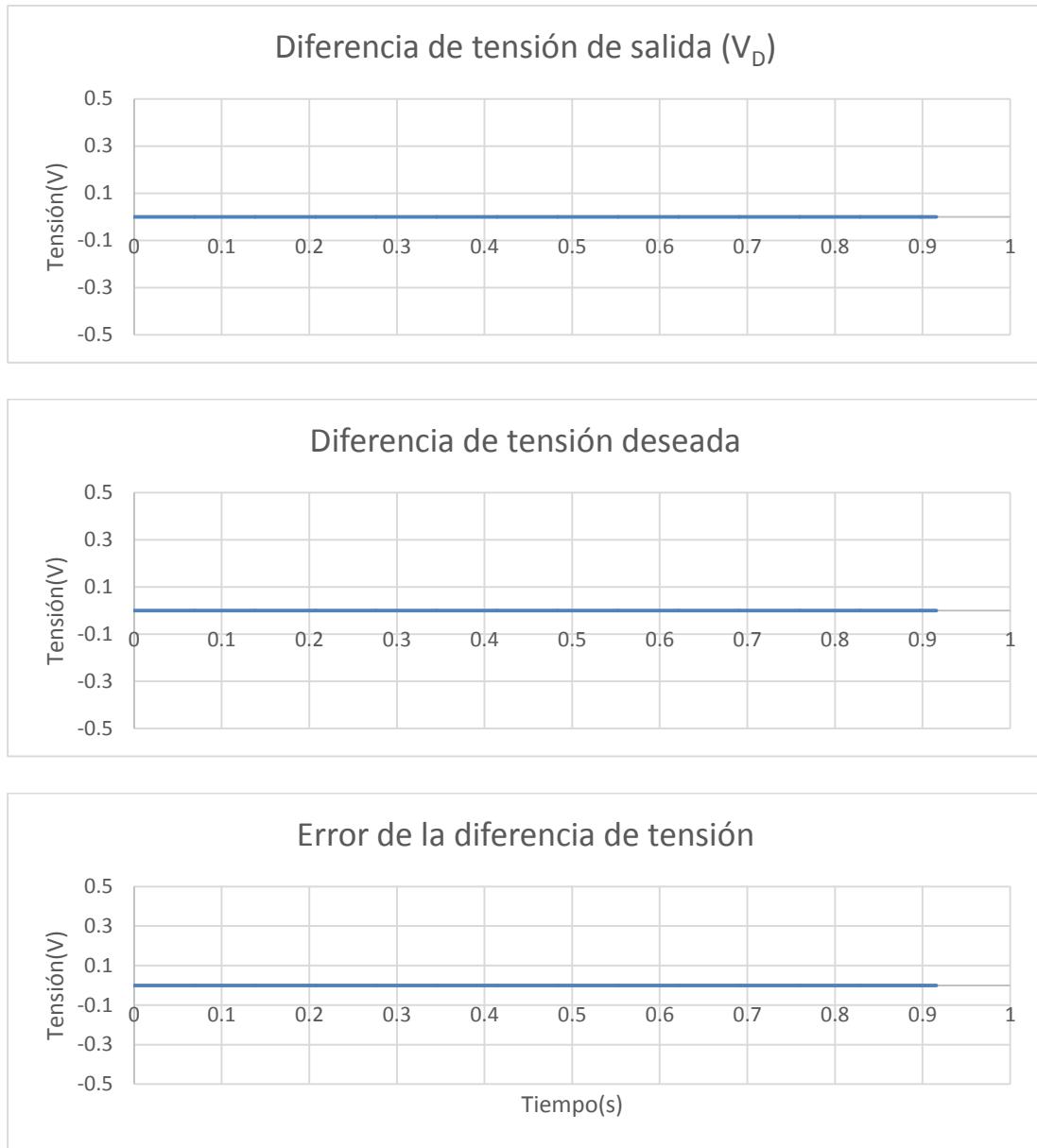


Figura 22. Resultados de la diferencia de la tensión en la simulación.

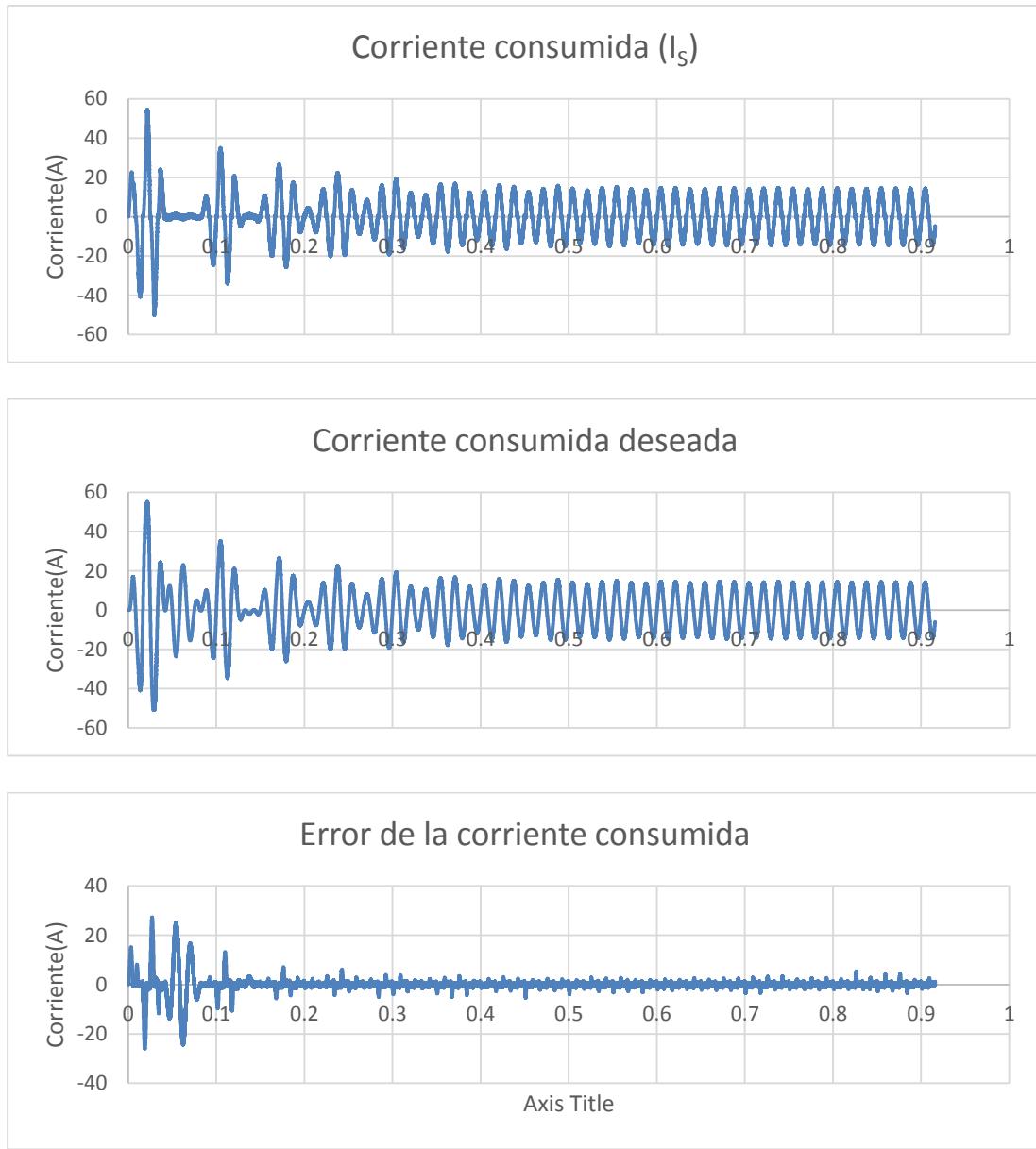


Figura 23. Resultados de la corriente consumida en el simulador.

Capítulo III. Herramienta de pruebas: Interfaz de control del emulador

Para poder manipular el emulador en tiempo de ejecución es necesario alguna interfaz externa la cual permita modificar los valores deseados ya sea de conexión o de configuración de la emulación. La interfaz creada se comunica con el microcontrolador a través de una conexión serial tipo UART (*Universal Asynchronous Receiver-Transmitter* por sus siglas en inglés) pero a diferencia de otros Microcontroladores que utilizan una conexión RS-232 el *PSoC 4 Prototyping kit* utilizado contiene un circuito integrado encargado de hacer la comunicación a través de un cable USB, reduciendo el costo para el usuario y facilitando uso.

La interfaz permite modificar todos los valores de conexión, como son el baud rate, handshake, paridad, etcétera, como se puede ver en la Figura 24. Además de esa funcionalidad también puede hacerse cambios en tiempo de ejecución de alguno de los valores en la emulación, o modos de funcionamiento como soporta el código del emulador de este escrito, según se haya implementado en el controlador.

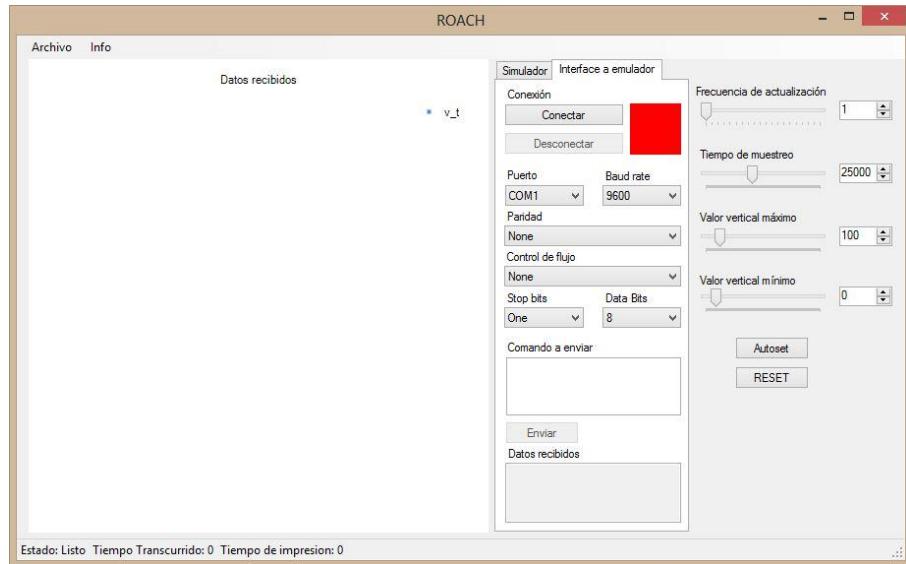


Figura 24. Controles de la conexión y cambio de valores de la interfaz al emulador.

La capacidad de hacer cambio alguno o varios parámetros de la simulación se hicieron para aumentar la flexibilidad de la herramienta, específicamente si se detectan diferencias de rendimiento en la ejecución del código entre la PC y el microcontrolador; o simplemente para algún cambio de estado o función que el usuario desee.

El protocolo de recepción de datos desde el microcontrolador utiliza un paquete de 5 bytes, los primeros dos bytes son los delimitadores de inicio y los últimos tres corresponden al valor del dato enviado. En la Figura 25 se puede observar un ejemplo así como la forma en que se ensambla el paquete; este protocolo fue creado con la intención de mantener una comunicación de una sola vía no segura para mantener la velocidad de los datos y no demandar demasiado al microcontrolador en el proceso de envío de datos para que sus recursos se utilicen en la emulación del sistema o bien, el control del circuito físico.

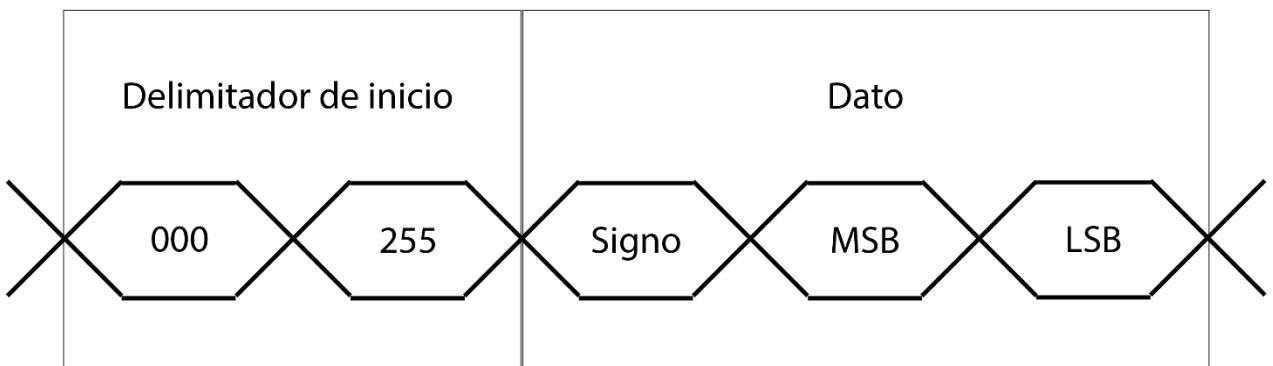


Figura 25. Arreglo de los bytes por posición del protocolo de recepción.

Para la interfaz del emulador solo se sustituye un hilo de los cuatro mencionados en el capítulo anterior, pues en este no se utilizara el hilo de comunicación con el DLL sino uno de comunicación UART, este comienza leyendo el buffer del componente serial UART, luego se utiliza como filtro frontal la búsqueda de los bytes delimitadores de inicio, si los dos primeros bytes recibidos coinciden con los valores predefinidos de estos se procede a resemblar el valor original enviado por el microcontrolador y se guarda en la fila concurrente para su impresión en la gráfica.

El cambio de valores se realiza con un protocolo diferente al utilizado en la recepción de datos del programa desde el microcontrolador, el paquete está formado por 8 bytes los cuales el usuario debe proporcionar en el formato que muestra la Figura 26, el primer byte es el delimitador de inicio, el segundo es el comando a ejecutar, los bytes 3 al 7 son el valor a enviar y el ultimo es la suma de verificación. Cuando el paquete es recibido y ratificado por el microcontrolador a través de la suma de verificación, el paquete es convertido al valor flotante descrito y asignado a la variable deseada en el mismo.

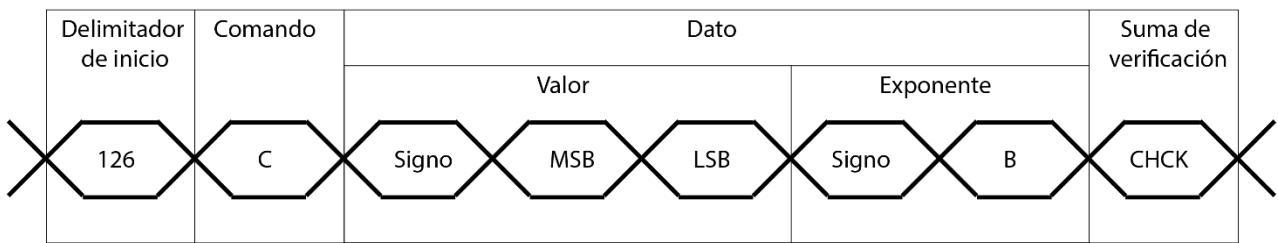


Figura 26. Arreglo de los bytes por posición del protocolo de transmisión.

La configuración de la comunicación serial UART utilizada en este particular emulador fue una tasa de baudios de 9600bps que es la tasa más rápida en la que se puede asegurar en 100% de transmisión y recepción de los datos, a 8 bits de datos, sin paridad y un bit de parada. Esta configuración se mantuvo sencilla para efectos de simplicidad pues no era necesario agregarle complejidad en este aspecto para el alcance de este proyecto. Cabe mencionar que la interfaz del emulador permite cambiar cualquiera de estos valores para adaptarse al proyecto a realizar.

Capítulo IV. Emulador

La plataforma elegida para el emulador fue un kit de prototipado *PSoC 4 CY8CKIT-049 Prototyping Kit* de Cypress que se muestra en la Figura 27, de ahora en adelante *PSoC 4 Prototyping Kit*, que utiliza un microcontrolador de la familia PSoC 4, este kit tiene un precio de \$4.00 USD lo cual lo hace una alternativa de bajo costo a muestras del fabricante o como método de aprendizaje/desarrollo para estudiantes en nuestra comunidad. El *PSoC 4 Prototyping Kit* contiene un circuito integrado USB serial que provee a los usuarios conexión directa a la PC que soporta las configuraciones USB-UART, USB-GPIO, USB-I2C y USB-SPI. La serie del PSoC 4 contiene un procesador ARM de 32 bits Cortex-M0 a 48 MHZ, 32kB de almacenamiento en Flash y 4KB de SRAM, lo cual lo hace una solución ideal de bajo costo para cálculos intensivos para un sistema embebido, además de sus

especificaciones técnicas la disposición de sus pines lo hace extremadamente cómodo para trabajar en tablero de prototipado, también conocidos popularmente como protoboards o breadboards, o bien para ser añadido a una placa personalizada a manera de tarjeta de expansión o shield.



Figura 27. Fotografía del PSoC 4 Prototyping Kit.

Para programar el PSoC 4 se utilizó el entorno de diseño integrado (IDE por sus siglas en inglés) PSoC Creator 3.2 SP1 que es un plataforma de desarrollo gratuita que se puede descargar directamente de la página de Cypress Semiconductor. Esta software nos permite utilizar los módulos predeterminados tanto digitales como analógicos así como la configuración de pines del SoC (System On Chip por sus siglas en inglés), manejar el código a ejecutar y una ventaja que tiene el PSoC 4 sobre otros Microcontroladores es que si se escoge el *kit CY8CKIT-042 PSoC 4 Pioneer Kit*(de ahora en adelante *Pioneer Kit*), con un costo de \$25 USD, sobre el *Prototyping Kit* mencionado anteriormente(cabe mencionar que ambos tienen el mismo microcontrolador) es que contiene un IC de la familia PSoC 5 que además de ser utilizado para programar el PSoC 4 también sirve a manera de depurador directamente desde el programa sin necesidad de agregar costos en otros instrumentos. En este escrito se utilizó ambos plataformas de prototipado, primero se utilizó el *Pioneer Kit* para agilizar el proceso de desarrollo y luego se utilizó el *PSoC 4 Prototyping Kit* para reducir costos, espacio y aumentar la simplicidad de uso en una PCB (*Printed Circuit Board*) personalizada.

El emulador contiene un componente digital que es un *Serial Communication Block* (SCB) en modo UART con la misma configuración a la de la interface de PC como se muestra en la Figura 28. Además del propio componente de comunicación también se utilizó un componente para manejar la interrupción de recepción de datos cuando el *FIFO* (First In First Out) no está vacío, esta interrupción se encarga de manejar los bytes recibidos para que puedan ser utilizados con el firmware creado para el emulador. Esta es la parte principal que permite el cambio de valores en tiempo de ejecución desde la interfaz de PC para hacer ajustes finos en el control o para cambiar la salida deseada.

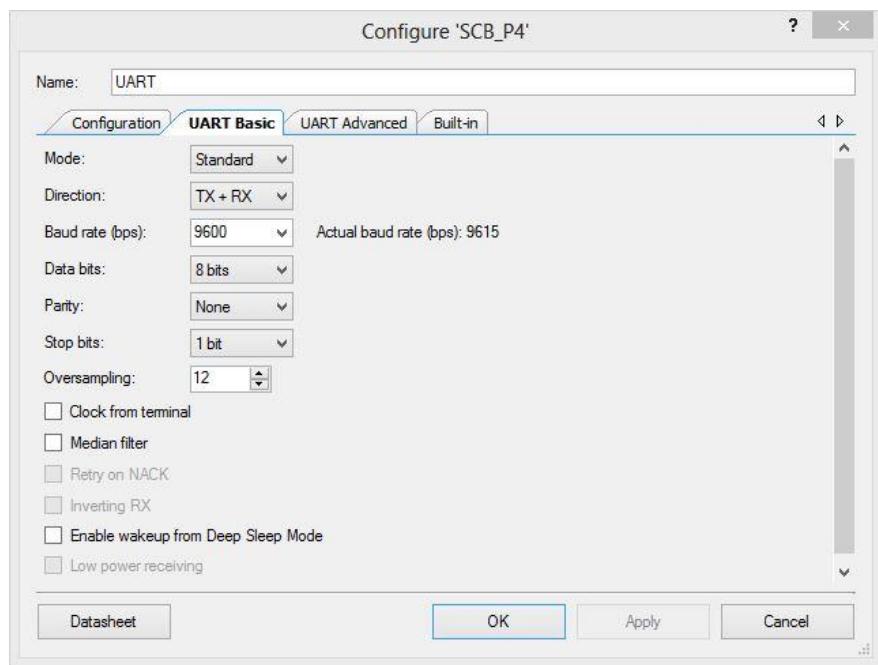


Figura 28. Captura de la configuración del componente serial.

Además del PSoC Creator 3.2 también fue necesario utilizar el programa USB Serial Configuration Utility, disponible por descarga gratuita, necesario para configurar el IC USB-Serial con los mismos valores de los parámetros, como muestra la Figura 29, en el bloque SCB UART dentro del PSoC 4 Prototyping Kit y de esta manera lograr una comunicación exitosa con el programa de PC.

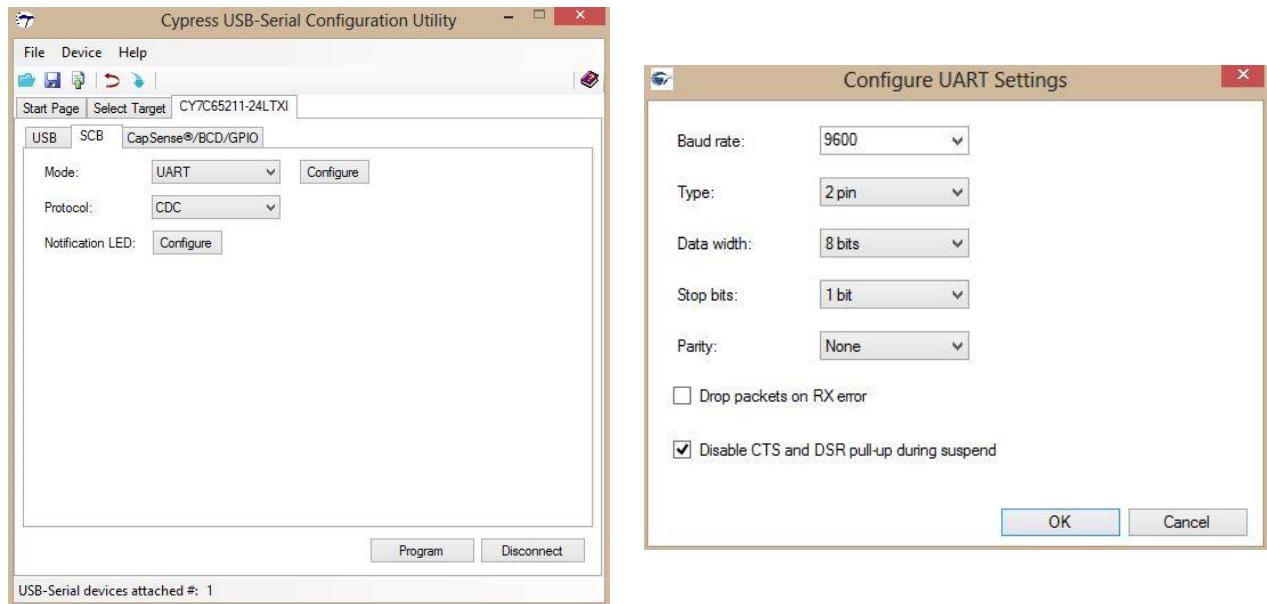


Figura 29. Configuración del IC USB-Serial en la PCB del PSoC4 Pioneer Kit.

Ya que las funciones fueron probadas en el simulador, estas pueden ser importadas directamente al PSoC. Los métodos que se importan son la integración, la función signo, el rectificador, el controlador y observador; para simplificar el proceso también se agregara una función que encapsule todo el proceso de la emulación.

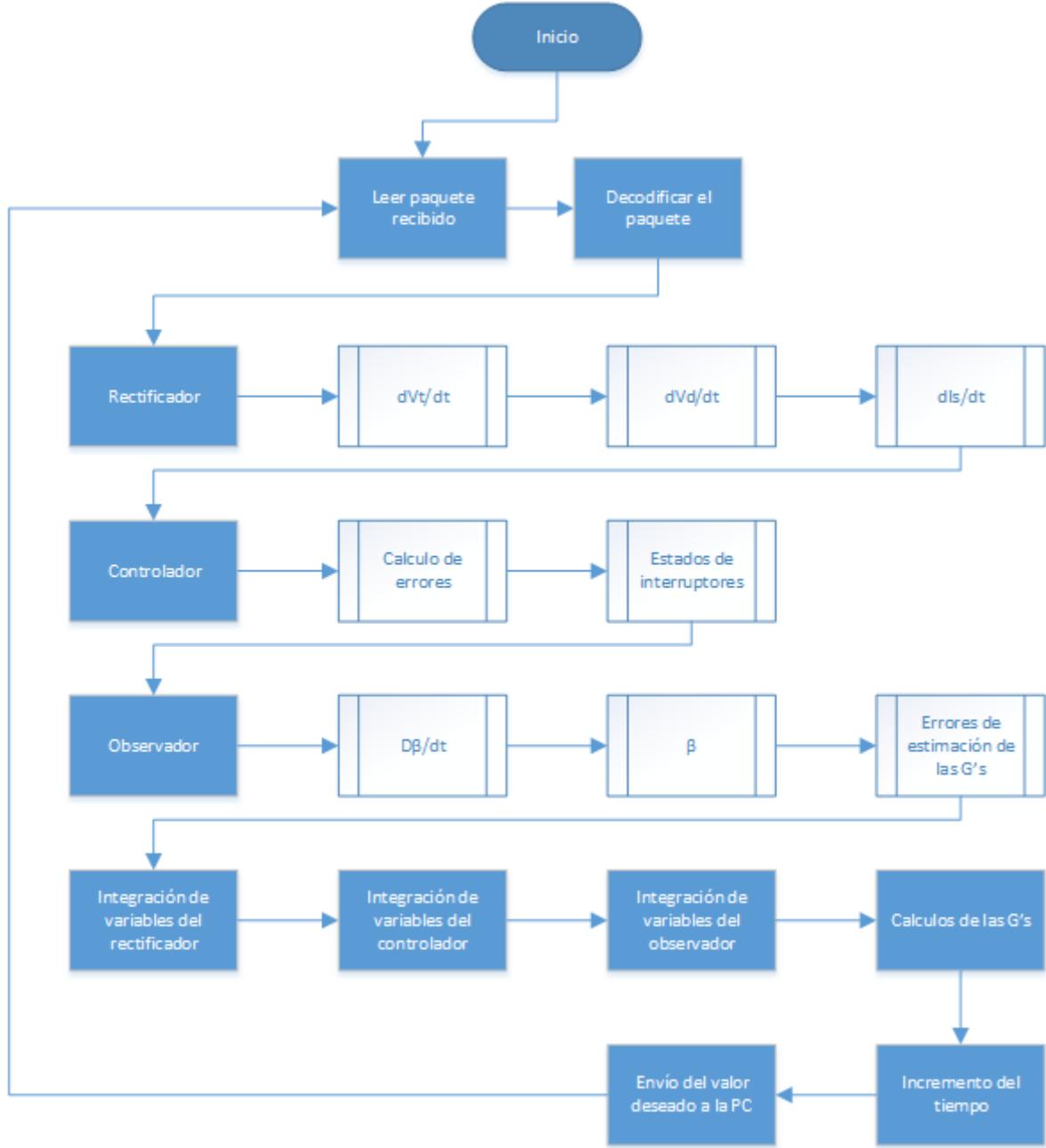


Figura 30. Diagrama de flujo del emulador.

Para cambiar los valores de los parámetros del rectificador, controlador u observador en el emulador es necesario utilizar el protocolo de 8 bytes descrito en el capítulo anterior, por ejemplo, si se requiere cambiar la tensión de salida deseada a 700V se debe de empezar con el delimitador de inicio, luego el

comando que es {0} (El apéndice A muestra la tabla de comandos utilizados en el emulador), el signo positivo que es {1}, el valor puesto en dos bytes {2, 188}, el exponente con signo y valor {1, 1} y la suma de verificación {62}, el paquete descrito se puede ver en la Figura 31. Para calcular la suma de verificación se deben de sumar todos los bytes, no incluyendo el delimitador de inicio, y manteniendo únicamente los 8 bits menos significativos del resultado y a continuación restárselos a 255.

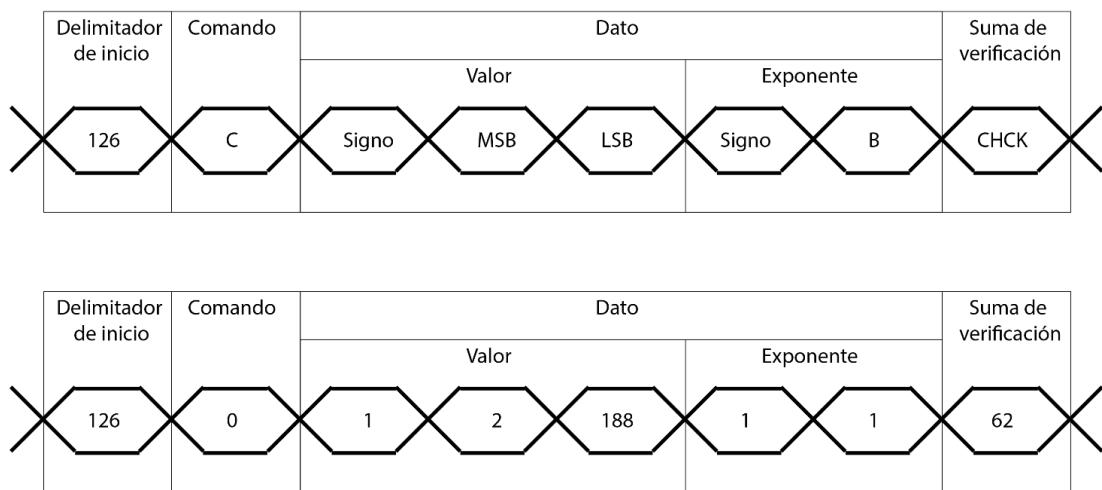


Figura 31. Ejemplo de creación de paquete.

Resultados

En la Figura 32 se puede observar que el tiempo de asentamiento es más rápido en comparación a los resultados obtenidos en MATLAB y el DLL en la PC, los picos visibles en la gráfica se deben a errores cometidos en la recepción o transmisión de los datos, esto fue un compromiso que se hizo durante el desarrollo del software. En la Figura 33 se observa el mismo comportamiento que en el DLL. Por último la Figura 34, correspondiente a la comparación de la corriente consumida muestra serias discrepancias con todos los resultados obtenidos anteriormente, nótese como la gráfica se encuentra desfasada con respecto al cero, este se debe a un error de transmisión en la lógica de transmisión o

selección de señales a transmitir en el microcontrolador, pues la tensión de salida deseada si se cumple y si existiese un problema, como es aparente, con la forma de la corriente el resultado obtenido en tensión sería completamente diferente o no sería estable.

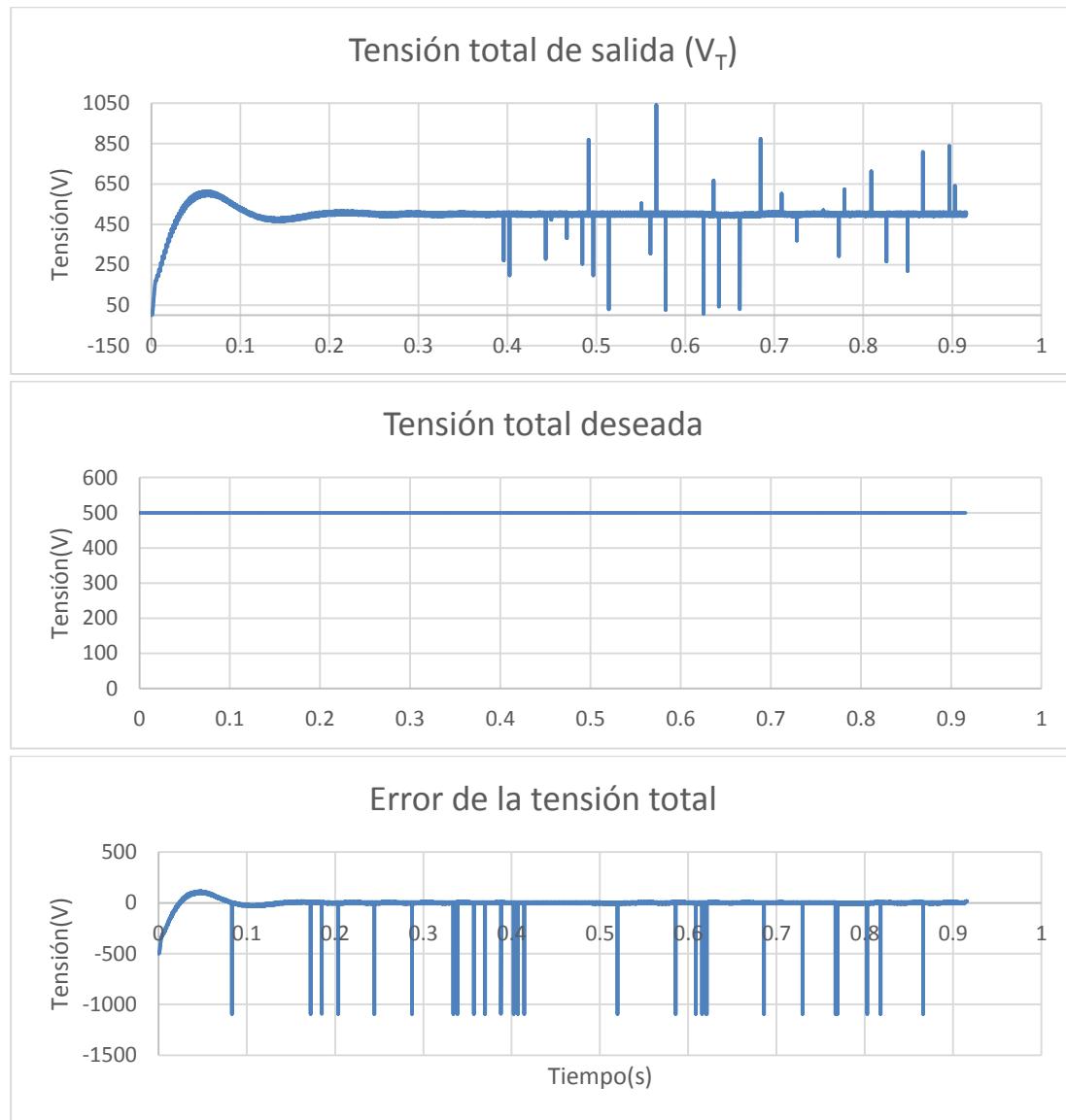


Figura 32. Resultados de la tensión total de la emulación en el microcontrolador.

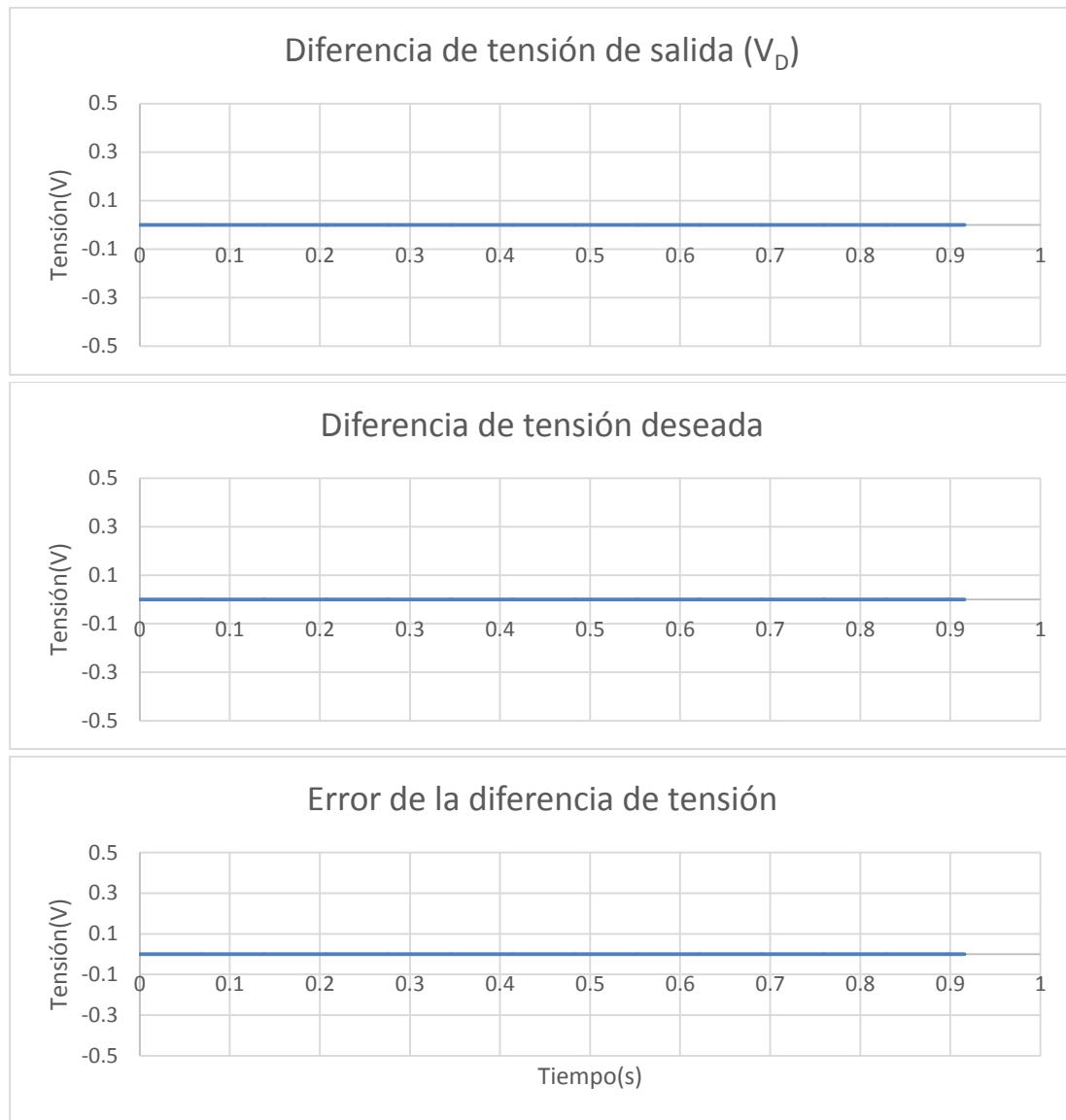


Figura 33. Resultados de la diferencia de tensiones entre capacitores en la emulación en el microcontrolador.

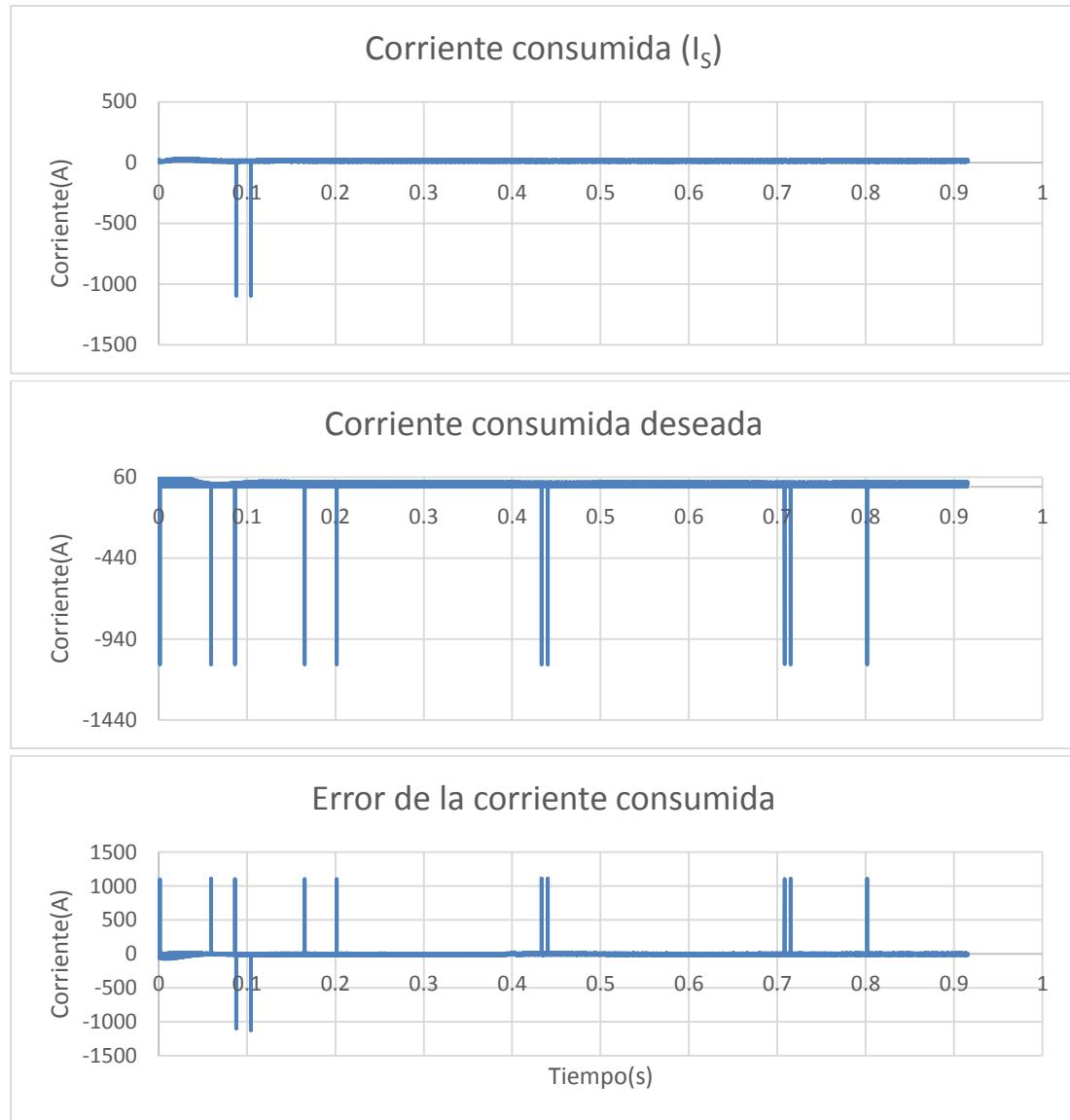


Figura 34. Resultados de las corrientes en la emulación del microcontrolador.

Conclusiones

A partir de los resultados obtenidos en esta tesis podemos concluir lo siguiente:

- Se diseñó una topología de rectificador con el respectivo control que sea capaz de producir y mantener la tensión de salida deseada, así como la diferencia de tensión entre los capacitores.
- Se modelo un observador que permite estimar el valor de las cargas del circuito en tiempo de ejecución, haciendo así más versátil el sistema como un todo.
- El modelo matemático del rectificar realizado permitió desarrollar un control y observador, los cuales pudieron ser analizados a través de simulación en diferentes plataformas para ratificar su funcionalidad propuesta.
- Se creó una herramienta capaz de cargar cualquier tipo de DLL en el lenguaje de programación C en tiempo de ejecución, siempre y cuando cumpla las condiciones establecidas, la cual permite el estudio y caracterización del algoritmo propuesto para su exportación hacia un microcontrolador que sea programable en estándar C.
- El software desarrollado es capaz de ratificar la funcionalidad propuesta del algoritmo a través de la comunicación serial UART con el microcontrolador.
- La herramienta de PC creada es capaz de hacer ajustes finos en el algoritmo, ya sea alojado en el DLL o en el microcontrolador.

Referencias

1. Brückner, T., Zorn, C., & Anders, J. (2014). A GPU-Accelerated Web-Based Synthesis Tool for CT Sigma-Delta Modulators. *IEEE Transactions on Circuits and Systems*, 1429-1441.
2. Buja, G., & Castellan, S. (2003). Development of an active power filter for medium-voltage diode rectifiers with DC-link capacitor. *European Conference on Power Electronics and Applications*, 1-9.
3. Choi, D. G., Lee, S. J., Cho, B., & Yoon, Y. G. (2010). Development of Design Tool for Hybrid Power Systems of Hybrid Electric Military Combat Vehicles. *Vehicle Power and Propulsion Conference* (pp. 1-5). Lille: IEEE.
4. Evans, P. L., Castellazzi, A., & Johnson, C. M. (2014). A multidisciplinary virtual prototyping design tool for power electronics. *International Conference on Integrated Power Systems* (pp. 1-7). Germany, Nuremberg: VDE.
5. Flota, M. (2009). *Análisis de propiedades, control y observación de un rectificador activo monofásico*. Tesis de doctorado, Facultad de Ingeniería. Universidad Autónoma de San Luis Potosí. México, San Luis Potosí.
6. Hart, D. (2001). *Electrónica de potencia, 1ra Edición*. España, Madrid: Prentice Hall.
7. Hermann, R. (1997). Nonlinear Controllability and Observability. *IEEE Transactions on automatic control*, 728-740.

8. Huang, S. J., Lin, B. R., & Huang, C. H. (2003). Implementation of a switching mode power supply with ZVT power factor correction. *International Conference on Power Electronics and Drive Systems*, 1402-1406.
9. Hung, T. L., Lin, B. R., & Chen, D. J. (2002). Half-bridge neutral point diode clamped rectifier for power factor correction. *IEEE Transactions on Aerospace and Electronic Systems*, 1287-1924.
10. Leng, C. W., Yang, C. H., & Tsai, C. H. (2008). An Integrated GUI Design Tool for Digitally Controlled Switching DC-DC Converter. *International Conference on Communications, Circuits and Systems* (pp. 1324-1327). Fujian: IEEE.
11. Lin, B. R., & Lu, H. H. (1999). A New Control Scheme for Single-Phase PWM Multilevel Rectifier with Power-Factor correction. *IEEE transactions on industrial electronics*, 820-828.
12. Lin, B. R., Chen, D. J., & Tsay, H. R. (2001). Bi-Directional AC/DC converter base on neutral point clamped. *IEEE transactions on industrial electronics*, 619-624.
13. Lin, B. R., Huang, C. H., & Yang, B. R. (2002). Control scheme of hybrid active filter for power quality improvement. *IEEE International Conference on Industrial Tecnology* (pp. 317-322 vol. 1). IEE.
14. Manjing, X. (2003). *Digital control for power factor correction*. Master's Thesis, Virginia Polytechnic Institute and State University. USA, Virginia.

15. Miaja, P. F., Lamar, D. G., & Pérez, M. A. (2001). A Switching-Mode Power Supply Design Tool to Improve Learning in a Power Electronics Course. *IEEE Transactions on Education*, 104-113.
16. Monopoli, V. G., Clare, J. C., Zanchetta, P., Gerry, D. B., & Wheeler, P. W. (2008). Predictive current control for multilevel active rectifiers with reduced switching frequency. *IEEE Transactions on Industrial Electronics*, 163-172.
17. Monopoli, V. G., Dell'Aquila, A., Liserre, M., & Rotondo, P. (2005). An energy-based control for an n-H-bridges multilevel active rectifier. *IEEE Transactions on Industrial Electronics*, 670-678.
18. Pires, V. F., & Silva, J. F. (2005). Three-phase single-stage four-switch PFC buck-boost-type rectifier. *IEEE Transactions on Industrial Electronics*, 52.
19. Radhakrishnan, V., & Ravikumar, S. (2013). Quick Fil: A Microwave filter design tool using C#.Net for teaching purposes. *International Conference on Computer Modelling and Simulation* (pp. 380-384). Cambridge: IEEE.
20. Rashid, M. (2001). *Power electronics handbook*. USA, California: Academic Press.
21. Shmilovitz, D. (2005). On the definition of total harmonic distortion and its effect on measurement interpretation. *IEEE Transaction on Power Delivery*, 526-528.
22. Teich, J. (2012). Hardware/Software Codesign: The Past, the Present, and Predicting the Future. *Proceedings of the IEEE*, 1411-1430.

23. Yinqing, Z. (1998). *Single phase power factor correction circuit with wide output voltage range*. Master's Thesis, Virginia Polytechnic Institute and State University. USA, Virginia.

Apéndices

A. Tabla de comandos del microcontrolador

Modo	Variable	Mascara del Comando	Comando	Valor Original
CAMBIO	Vt_des	CMD_VT DES	0	500
CAMBIO	Vd_des	CMD_VD DES	1	0
CAMBIO	R1	CMD_R1	2	100
CAMBIO	R2	CMD_R2	3	100
CAMBIO	C	CMD_C	4	4.70E-04
CAMBIO	L	CMD_L	5	1.00E-02
CAMBIO	k	CMD_K	6	100000
CAMBIO	kp1	CMD_KP1	7	0.01
CAMBIO	kp2	CMD_KP2	8	250
CAMBIO	ki1	CMD_KI1	9	10
CAMBIO	ki2	CMD_KI2	10	500
CAMBIO	g1	CMD_G1	11	
CAMBIO	g2	CMD_G2	12	
CAMBIO	gamma1	CMD_GAMMA1	13	1.00E-03
CAMBIO	gamma2	CMD_GAMMA2	14	1.00E-03
MODO	Emulador	CMD_EMULATOR_MODE	15	
MODO	Controlador	CMD_CONTROLLER_MODE	16	
MODO	Repetir pedido	CMD_REPEAT_REQUEST	17	
VER	Vt_des	SEE_VT DES	18	
VER	Vd_des	SEE_VD DES	19	
VER	R1	SEE_R1	20	
VER	R2	SEE_R2	21	
VER	C	SEE_C	22	
VER	L	SEE_L	23	
VER	k	SEE_K	24	
VER	kp1	SEE_KP1	25	
VER	kp2	SEE_KP2	26	
VER	ki1	SEE_KI1	27	
VER	ki2	SEE_KI2	28	
VER	g1	SEE_G1	29	
VER	g2	SEE_G2	30	

VER	gamma1	SEE_GAMMA1	31	
VER	gamma2	SEE_GAMMA2	32	
VER	v_t	SEE_V_T	33	POR DEFECTO
VER	v_d	SEE_V_D	34	
VER	i_s	SEE_I_S	35	
VER	u1	SEE_U1	36	
VER	u2	SEE_U2	37	
VER	vt_error	SEE_VT_ERROR	38	
VER	vd_error	SEE_VD_ERROR	39	
VER	is_error	SEE_IS_ERROR	40	

