Universidad Zaragoza
1542

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Modulador digital configurable para identificación de etapas de potencia

Digital configurable modulator for power systems identification

Autor

Javier Inchaurralde Rodríguez-Maimón

Director

Luis Ángel Barragán

ESCUELA DE INGENIERÍA Y ARQUITECTURA

Diciembre de 2016

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Javier Inchaurralde Rodríguez-Maimón,

con nº de DNI 73025660-Q en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Modulador digital configurable para la identificación de etapas de potencia.

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 18 de Noviembre de 2016

Fdo: Javier

MODULADOR DIGITAL CONFIGURABLE PARA IDENTIFICACIÓN DE ETAPAS DE POTENCIA

RESUMEN

En este trabajo se ha implementado un modulador configurable para la identificación de etapas de potencia. El modulador se ha diseñado para que la señal PWM introducida en el sistema permitiera su identificación mediante pequeñas variaciones en el ciclo de trabajo en un rango de frecuencias de interés próximas a la de cruce.

La etapa escogida para implementar los métodos de identificación empleados ha sido un convertidor Buck o etapa reductora. El modulador ha sido implementado en una FPGA modelo SPARTAN 3E 1200 integrando en la arquitectura diseñada la generación de la señal PWM y la recepción de la información del valor de los parámetros característicos de las perturbaciones introducidas en la señal de modulación.

Para valorar la eficacia y utilidad de los métodos a realizar experimentalmente se ha llevado a cabo previamente su simulación mediante software. El software empleado ha sido Scilab, de libre licencia. Por otra parte, la información recibida por la FPGA ha sido transmitida desde un PC haciendo uso de una interfaz gráfica para configurar los datos correspondientes a los parámetros de la señal de modulación y enviarlos mediante comunicación serie. La interfaz, que permitía visualizar en tiempo real los valores configurados ha sido programada mediante Processing, un software libre basado en Java.

El modulador se ha diseñado con capacidad de crear dos tipos de perturbaciones distintas en el ciclo de trabajo de la señal PWM con el fin de implementar posteriormente dos métodos de identificación independientes. Dichos métodos han sido previamente comparados mediante simulación y una vez obtenidos y procesados los resultados experimentales resultantes de la implementación con una etapa Buck.

ÍNDICE

CAPÍTULO 1.

INTRODUCCIÓN.

1.1 ESTADO DE LA TÉCNICA

La identificación de sistemas dinámicos e invariables se fundamenta en aplicar métodos estadísticos y matemáticos a las salidas y entradas introducidas en un sistema. Como resultado, se pueden obtener modelos parámetricos o no parámetricos.

Los modelos paramétricos, entre los que se incluye el modelado mediante leyes físicas, son aquellos en los que se conoce o presupone un número limitado de parámetros mediante los cuales se puede describir el sistema [1].

En el caso del modelado físico, para el cuál no se requiere una identificación del sistema, los diferentes términos de las ecuaciones que permiten modelarlo son los que determinan cuáles y cuántos son los parámetros que rigen el comportamiento de un sistema.

En otra ocasiones este tipo de modelos se utilizan para analizar un sistema del que no se tiene ningún conocimiento previo aproximándolo a un sistema lineal o no lineal discretizado para muestrear un conjunto finito de valores en la entrada y la salida del mismo. El inconveniente en la implementación de este tipo de métodos se encuentra en la elección del orden del sistema para realizar la aproximación, el cual depende del número de parámetros del mismo [2].

La forma general de representar estos modelos es formulando la salida como la suma de la entrada del sistema con un término de valor distribuido aleatoriamente en el tiempo de media nula, cada uno de ellos multiplicados por un filtro de orden finito:

$$y(t) = G(q^{-1})u(t) + H(q^{-1})e(t) \qquad (1)$$

El procedimiento de identificación consiste en tratar de identificar y optimizar los valores de los coeficientes de cada filtro a partir de la estimación del error entre las salidas esperadas del sistema y las reales, aplicando métodos de regresión lineal.

Los modelos no paramétricos, al contrario que los paramétricos, son más sencillos de obtener ya que no requieren ninguna información previa del sistema y por tanto del número de parámetros. Las técnicas de identificación correspondientes a este tipo de modelos permiten por si solas caracterizar el sistema con precisión suficiente para numerosas aplicaciones con un número no finito de parámetros. Dentro de este grupo de modelos se incluye el análisis transitorio de la respuesta del sistema a entradas específicas como son el impulso, escalón y rampa unitarios; y el análisis de la respuesta en frecuencia. Normalmente, interesa conocer una respuesta del sistema a frecuencias cercanas a las de corte del mismo con el fin de implementar los métodos de control más utilizados habitualmente en la mayoría de aplicaciones como son el controlador proporcional-integral.

Las técnicas de identificación no paramétricas basadas en la respuesta en frecuencia más

habitualmente empleadas y que son desarrolladas en este trabajo son el método de la correlación cruzada [3] y el análisis de Fourier [4].

El método de la correlación cruzada tiene como objetivo hallar de forma indirecta la respuesta al impulso del sistema, al ser esta imposible de generar de forma física, a partir de la correlación entre la respuesta al ruido blanco y éste mismo para, una vez normalizado el resultado obtenido en función de la varianza del ruido blanco introducido, aplicar la técnica de la Transformada Discreta de Fourier y obtener una representación de la respuesta del sistema a distintas frecuencias. El rango de frecuencias de análisis del método viene dado para una única entrada en función del número de muestras y la frecuencia de muestreo utilizada.

En las técnicas que emplean únicamente el análisis de Fourier el grado de exactitud obtenido en la caracterización depende del número distinto de frecuencias de las entradas introducidas en el sistema, las cuáles suelen ser senoides o sumas de ellas. La caracterización es inmediata para cada frecuencia por la propia definición de sistema invariante y lineal, ya que la respuesta del sistema $G(\alpha)$ a una senoide de magnitud A y fase β es otra senoide de la misma frecuencia con amplitud $G \cdot A$ y fase $\alpha + \beta$.

1.2 MOTIVACIÓN DEL TRABAJO

El trabajo realizado se sitúa en el marco de la identificación de sistemas, el cuál es un campo activo de la electrónica y de la ingeniería de sistemas. El motivo fundamental que me ha llevado a realizar este trabajo es mi creciente interés por la electrónica y en particular por la electrónica digital enfocada a analizar, crear, y/o implementar algoritmos que sirvan tanto para mejorar los métodos actuales en el control de sistemas como en la identificación mediante soluciones prácticas e innovadoras.

Por otra parte, el hecho de poder realizar un trabajo que me permitiera llevar a cabo una aplicación real que requiriera de un desarrollo experimental completo en el laboratorio me ha permitido observar y aprender algunas de las pautas necesarias para poder llevar a cabo más adelante de forma más eficaz trabajos similares que involucren una fase previa de simulación en software.

1.3 OBJETIVOS DEL TRABAJO

El objetivo de este trabajo ha sido el de encontrar una solución rápida, precisa y con los mínimos recursos posibles al problema de la identificación en etapas de potencia. A pesar de que por regla general, el modelo físico de la etapa que se está empleando es conocido, es posible que haya imprecisiones en la caracterización de su funcionamiento en condiciones y momentos distintos. Esto, es debido por una parte, a la previa suposición al modelar el sistema de que éste es lineal e invariable en el tiempo y por otra, a la omisión en el modelado de elementos o fenómenos que tienen influencia real en el comportamiento del mismo. Así pues, en la mayoría de aplicaciones de potencia, los elementos no responden con la misma eficacia que en un principio debido al desgaste temporal provocado por su uso continuado. Del mismo modo, al querer simplificar el comportamiento de un sistema para obtener un primer modelo del mismo lo que se hace es

idealizar comportamientos de los elementos electrónicos despreciando fenómenos que pueden tener una influencia relevante.

No obstante, estos modelos basado en un conocimiento previo de las leyes físicas que rigen el comportamiento de los elementos que integran la etapa en régimen permanente proporcionan una primera aproximación fiable en las condiciones esperadas de funcionamiento que además sirve para establecer las pautas a la hora de usar métodos alternativos de identificación experimentales sobre el sistema físico real estudiado. Estos últimos proporcionan un modelo del mismo sin entrar a analizar en detalle el motivo físico de las diferencias con respecto al comportamiento previsto en un principio, mediante únicamente un análisis de la respuesta ante diferentes entradas.

El objetivo principal de este trabajo ha sido el diseño e implementación de un modulador de forma digital mediante una FPGA del ciclo de trabajo de los transistores de la etapa escogida, la cuál ha sido un convertidor reductor o Buck, con el fin de obtener de dos formas alternativas el modelo del sistema analizado. Se han empleado dos métodos distintos basados en las referencias [3] y [4] respectivamente: el método de la correlación cruzada y el método de la DFT. El método de la correlación cruzada ha sido implementado de forma digital tal siguiendo la metodología descrita en [3], mientras que para el método de la DFT se ha partido de un análisis del método llevado a cabo con un modulador implementado de forma analógica [4] para reproducir con la máxima exactitud posible la señal de modulación de forma digital.

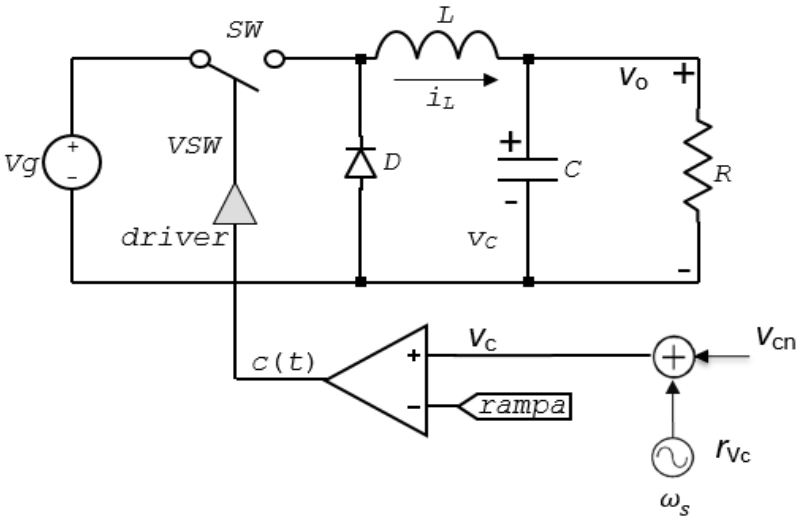

Figura 1. Implementación del método de la DFT de forma analógica.

En el método analógico una señal sinusoidal de baja amplitud es generada por una fuente de tensión AC con un cierto offset que actúa como el ciclo medio de trabajo. La señal es llevada a un amplificador operacional que actúa comparando esta señal con otra que forma una rampa de amplitud igual al valor de tensión de offset de la señal con la que se compara entre el ciclo de trabajo medio y de periodo el de conmutación del transistor.

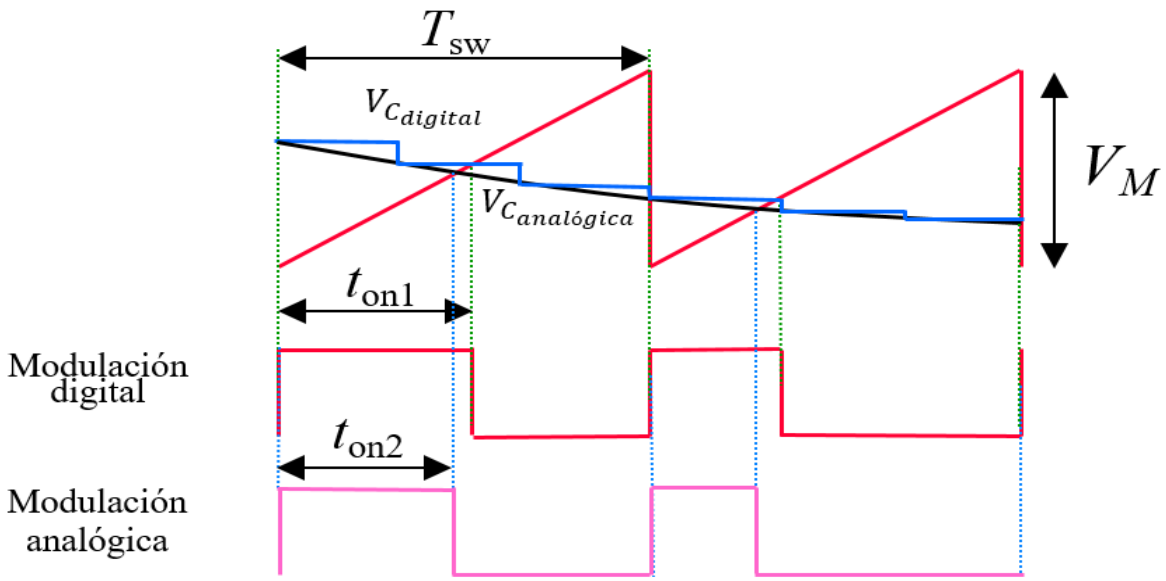

Figura 2. Comparación de las señales de modulación generadas de forma analógica y digital con una perturbación senoidal.

Al ser generada de forma digital, por el contrario, el valor que toma la señal de modulación PWM contiene un cierto error al ser generada la onda senoidal mediante valores discretos, por lo que es conveniente tomar el mayor número posible de éstos para cada periodo de la perturbación.

De este modo, partiendo de un modelo simple obtenido de forma teórica mediante las consideraciones antes mencionadas siendo además conocido el valor de los parámetros que caracterizaban los elementos que integraban la etapa al modulador se le ha otorgado la capacidad de hacer variar el ciclo de trabajo del convertidor de dos formas distintas para así aplicar los dos métodos de identificación analizados en este trabajo a los valores de la salida y de la entrada. Por otra parte, debido al carácter independiente de los métodos empleados con respecto a la etapa analizada, el modulador podría haberse empleado con similar eficacia en otro tipo de sistemas lineales e invariantes en el tiempo modulados por señales PWM conmutando a la misma frecuencia para obtener un modelo no paramétrico de los mismos.

1.4. ESTRUCTURA DE LA MEMORIA

La memoria de este trabajo se ha dividido en siete capítulos o bloques:

- INTRODUCCIÓN: En él se ha analizado el estado actual de los métodos aplicados a la identificación de sistemas así como los objetivos y la motivación a nivel académico y técnico de la realización del trabajo.

- MODELADO DEL CONVERTIDOR: Se ha descrito el funcionamiento del convertidor en modo contInuo así como el modelado en pequeña señal del mismo. Asimismo se presenta el software utilizado (Scilab) para realizar el modelado y las simulaciones como alternativa frente a otros de licencia restringida.

- IDENTIFICACIÓN DEL CONVERTIDOR: Se han resumido los fundamentos de los dos métodos utilizados para realizar la identificación así como las pautas para ser llevados a cabo y se ha realizado una comparativa de los resultados obtenidos mediante simulación al llevar a cabo el proceso de identificación.

- IMPLEMENTACIÓN: Se ha descrito el proceso completo que tiene lugar desde la transmisión de los datos correspondientes a los parámetros de la modulación desde el PC hasta la obtención del modelo del sistema entrando en detalle en el funcionamiento de la interfaz desarrollada para permitir la comunicación con la FPGA, el protocolo de comunicación serie empleado, los parámetros enviados y el programa VHDL para recibir los datos y generar la señal PWM añadiendo pequeñas perturbaciones en el ciclo de trabajo.

- VERIFICACIÓN EXPERIMENTAL: Se ha descrito el procedimiento llevado a cabo en el laboratorio para la realización de las medidas así como los resultados finales obtenidos tras ser procesados mediante software.

- CONCLUSIONES Y TRABAJOS FUTUROS: Por último, se han expuesto las conclusiones extraídas a raíz del trabajo realizado y se presenta un avance de futuras ideas a implementar en posteriores trabajos.

CAPÍTULO 2.

MODELADO DEL CONVERTIDOR.

2.1 CONVERTIDOR REDUCTOR BUCK

La función del convertidor Buck es la de reducir una tensión contInua de entrada para obtener a la salida un valor de tensión contInua proporcional al ciclo de trabajo empleado. La regulación del ciclo de trabajo se hace mediante el uso de un transistor que se activa un determinado intervalo de tiempo cada periodo de conmutación. La aplicación principal de esta etapa es la de control de velocidad de motores de tensión continua CC y en fuentes de energía CC.

De este modo la relación entre el voltaje de entrada y de salida es:

$$Vin = \frac{(Vin \cdot t_{on} + 0 \cdot t_{off})}{T_{conm}} = DVout \qquad (2)$$

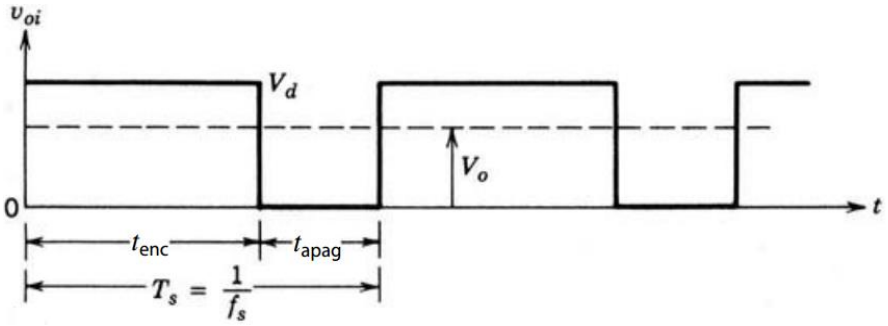

Figura 1. Voltaje de entrada y medio de salida en un convertidor Buck [5].

Al fluctuar el valor del voltaje en la salida entre el valor de la entrada y 0 en cada ciclo de conmutación es necesario implementar un filtro paso bajo formado por una bobina y un condensador con el fin de eliminar los armónicos adicionales de la señal de modulación y de esta manera obtener únicamente un valor continuo correspondiente al valor medio de tensión en la salida. La frecuencia de corte del filtro debe ser de orden de magnitud menor que la frecuencia de conmutación para poder realizar el filtrado.

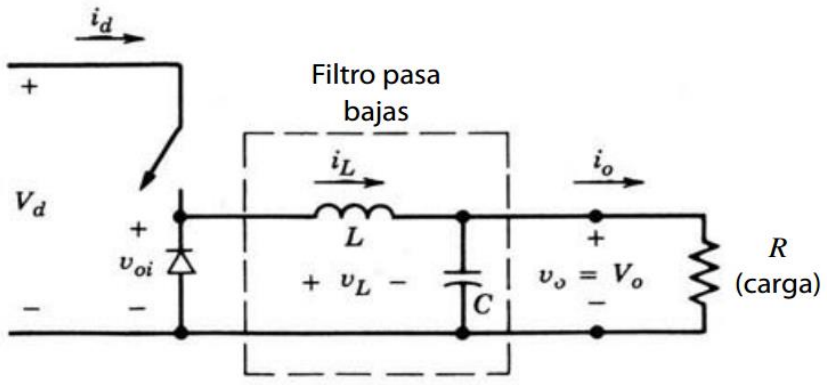

Figura 2. Representación del esquema del convertidor y del filtro paso bajo [5].

Durante el tiempo en el que interruptor está encendido, el diodo se encuentra en corte y la corriente fluye desde la fuente hacia la bobina, que es cargada en corriente, y hacia el condensador y la carga. Al estar el interruptor apagado, la bobina se descarga en corriente y es recirculada por el diodo.

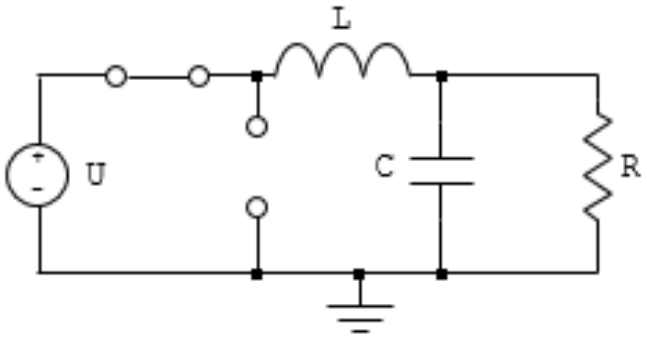

Figura 3 .Representación del circuito equivalente del Buck con el transistor en modo ON .

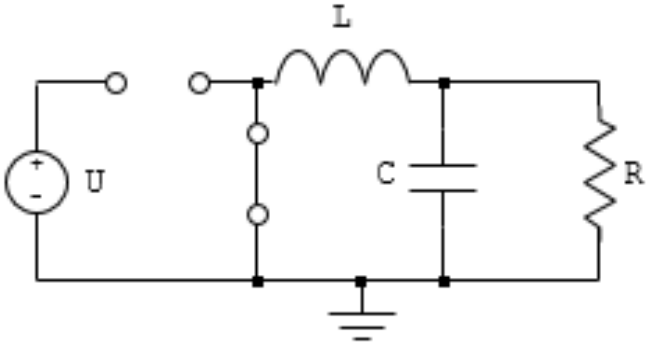

Figura 4 .Representación del circuito equivalente del Buck con el transistor en modo OFF .

En régimen permanente debe cumplirse para cualquier elemento del circuito que la intensidad y la tensión sean periódicos a la frecuencia de conmutación. En el caso de cualquier inductor se cumple que la integral de la tensión a lo largo de un periodo es nula, lo mismo ocurre con el voltaje promedio.

Esto se puede observar en el comportamiento de la corriente de la bobina en el modo de conducción continua del Buck, en el que ésta nunca alcanza un valor nulo y aumenta o disminuye de forma lineal en torno a un valor medio según el transistor que regula los tiempos de conmutación este encendido o apagado, ya que la tensión durante los tiempos en alto o en bajo tiene un valor constante positivo o negativo respectivamente

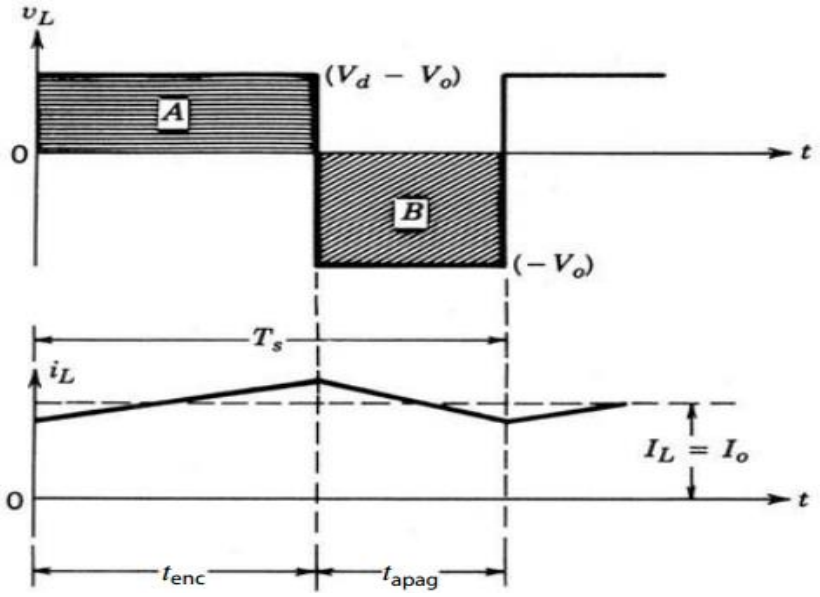

Figura 5. Tensión y corriente de la bobina en modo continuo del Buck [5].

De este modo en el tiempo en el cuál el transistor está conduciendo la caída de tensión en la bobina es igual a la tensión de la fuente menos el voltaje en la carga mientras que cuando este está en corte y el diodo conduce la caída de en la bobina se reduce la tensión en la carga cambiando el signo.

Por otra parte, partiendo de la premisa de que la integral de la tensión a lo largo de un ciclo de conmutación es nula, se llega de nuevo a la expresión que relaciona el voltaje de entrada con el de salida mediante una relación de áreas:

$$(V_d - V_o)t_{on} = V_o(T - t_{off}) \quad (3)$$

$$V_o = DV_d \quad (4)$$

2.2 MODELADO DE PEQUEÑA SEÑAL

Tras un análisis del funcionamiento del convertidor se ha realizado un modelado en pequeña señal para mediante la transformada de Laplace poder traducir el funcionamiento en modo circuital a bloques que utilicen expresiones en el dominio s lo para poder reproducir más fácilmente mediante software la eficacia de los métodos de identificación estudiados en este trabajo. Para ello se han determinado las ecuaciones en régimen temporal que rigen su comportamiento físico para posteriormente introducir pequeñas perturbaciones en torno a un punto de equilibrio para por último, tras eliminar los términos no lineales y de segundo orden obtener la función de transferencia tensión de salida-ciclo de trabajo .Las variables perturbadas han sido la tensión de salida, la corriente por la bobina y el ciclo de trabajo.

A continuación, se ha obtenido a la expresión correspondiente al modo de funcionamiento en ON del transistor y con el diodo en corte:

$$V_o\left(\frac{R_o+R_L+Ls-R_oV_i}{R_o}\right)+\frac{V_o}{1+sCR_c}Cs(R_L+Ls)=0 \tag{5}$$

De modo similar, en el modo de funcionamiento con el diodo conduciendo y el transistor apagado el sistema puede ser modelado de la misma forma que en el modo anterior pero teniendo en cuenta ahora que el término correspondiente a la tensión de entrada es nulo:

$$V_o\left(\frac{R_o+R_L+Ls}{R_o}\right)+\frac{V_o}{1+sCR_c}Cs(R_L+Ls)=0 \tag{6}$$

Posteriormente sumando (5) y (6) y multiplicando cada ecuación por la fracción de tiempo en alto y en bajo respectivamente y perturbando las variables corriente de la bobina, tensión de salida y ciclo de trabajo [6] se ha obtenido la función de transferencia tensión de salida-ciclo de trabajo para pequeñas variaciones:

$$G(s)=\frac{\widehat{V_o}}{\hat{d}}=\frac{V_i\frac{R_O}{R_L+R_O}(s\cdot R_C\cdot C+1)}{LC\left(\frac{R_C+R_O}{R_L+R_O}\right)s^2+\left(R_CC+\left(\frac{R_L}{R_O}\right)C+\left(\frac{L}{R_L+R_O}\right)\right)s+1} \tag{7}$$

La cual representa a un sistema de segundo orden con un cero y dos polos conjugados.

El sistema ha sido representado mediante un diagrama de bloques para realizar las simulaciones en Scilab en presencia de perturbaciones en el ciclo de trabajo.

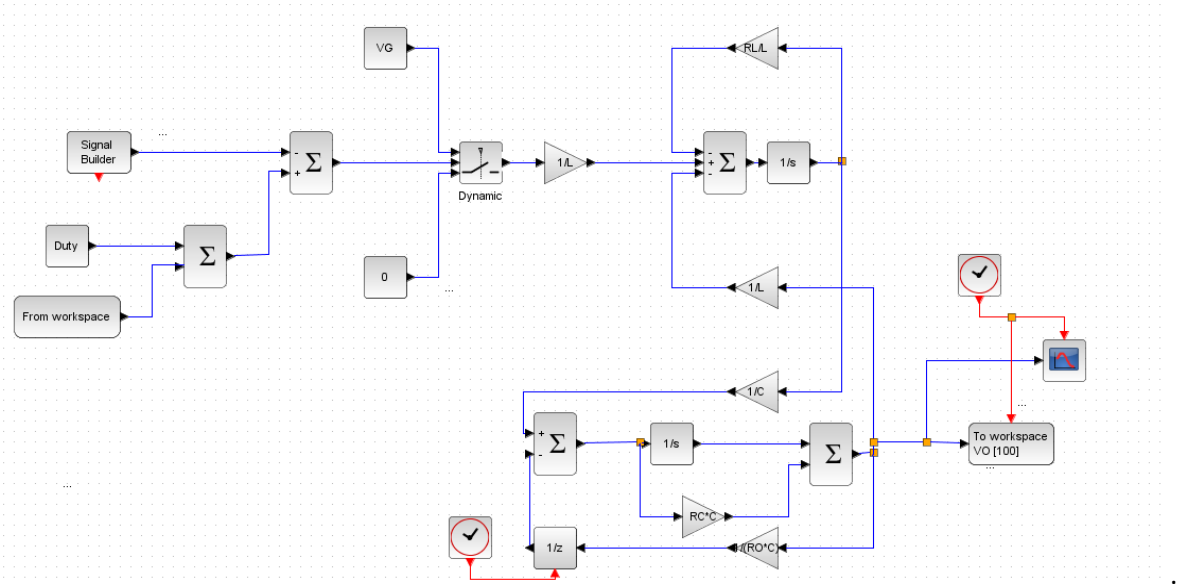

Figura 6 .Diagrama de bloques empleado para realizar las simulaciones.

En el diagrama se observa como el bloque de la esquina superior izquierda es el responsable de generar la onda periódica que tiene una forma característica de rampa a la que se resta el valor constante del ciclo de trabajo (Duty) en presencia de perturbaciones generadas mediante un vector de valores y tiempos creado desde el "workspace. La señal resultante es llevado a un comparador que genera la señal de modulación alternando valores entre la tensión de entrada (Vg) y 0.

A continuación el valor de la señal de modulación se multiplica por la ganancia 1/L y le es sustraído en el bloque sumatorio el valor de la tensión de salida Vo multiplicado por 1/L y a su vez la corriente por la bobina I_L multiplicada por la ganancia R_L/L. La corriente I_L es obtenida a la salida del bloque una vez integrada su salida satisfaciendo de este modo la expresión de la dinámica de la corriente en la bobina:

$$\frac{dI_L}{dt} = \frac{V_g}{L} - \frac{I_L R_L}{L} - \frac{V_o}{L} \tag{8}$$

Por otra parte, la corriente I_L es dividida entre la capacitancia del condensador C y a continuación le es sustraída la tensión de salida Vo multiplicada por la ganancia $1/(R_o C)$, satisfaciendo la expresión de la dinámica de la tensión en el condensador:

$$\frac{dV_c}{dt} = \frac{I_L}{C} - \frac{V_o}{R_o C} \tag{9}$$

Por último, la derivada de la tensión del condensador es por un lado integrada y por otro multiplicada por la ganancia R_cC obteniendo la tensión de salida V_o una vez sumadas ambas variables.

$$\frac{dV_c}{dt} + \frac{V_c}{R_cC} = \frac{V_o}{R_cC} \tag{10}$$

La tensión de salida es almacenada en un registro para ser procesada posteriormente.

Los parámetros del Buck que han utilizado tanto en simulación como al realizar las medidas experimentales han sido los siguientes: L = 68 µH, C = 220 µF, resistencia serie equivalente (ESR) del condensador R_C = 80 mΩ, resistencia DC de la bobina R_L = 98 mΩ , resistencia de carga R_o = 2.5 Ω ,tensión de entrada Vi= 5 V y una frecuencia de conmutación fs = 100 kHz.

La frecuencia de cruce del sistema se sitúa en f= 3.18 kHz , el margen de fase en 34,025° y la frecuencia de corte del filtro viene dada por la relación:

$$f_c = \frac{1}{2\pi\sqrt{LC}} = 1.301\ kHz \tag{11}$$

El diagrama de Bode simulado en Scilab se presenta a continuación:

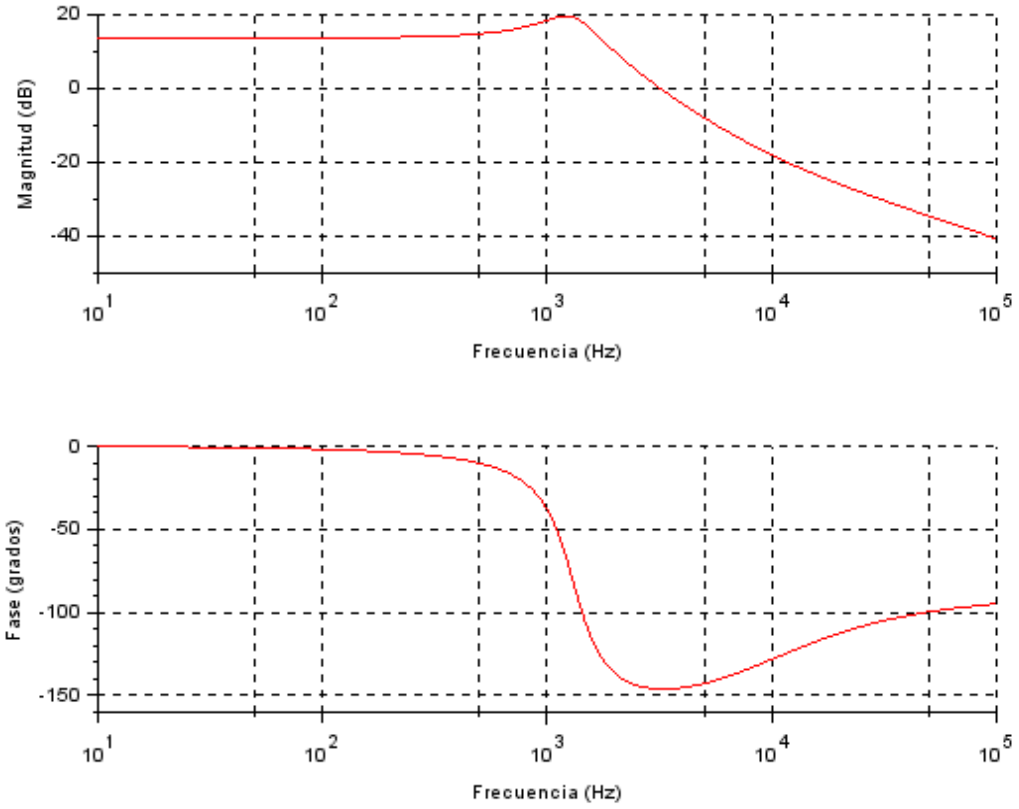

Figura 7 .Diagrama de Bode del sistema analizado.

En él se observa la caída de fase y ganancia a la frecuencia de corte del filtro de -40dB/década y -150° y la posterior recuperación debido al cero del condensador.

2.3 SOFTWARE LIBRE SCILAB

Para realizar las simulaciones se ha empleado el software de licencia libre Scilab. Scilab, creado en 1990 por investigadores pertenecientes al INRIA (Institut National de Recherche en Informatique et Automatique) y al ENPC (École Nationale des Ponts et Chaussées) [7] opera en Windows, Linux y Mac OS X. Incorpora funcionalidades similares a las de Matlab integrando una consola en la cual se pueden visualizar las variables empleadas y los resultados obtenidos de las operaciones que se quieren realizar, un entorno similar a Simulink llamado xCos mediante el cuál realizar simulaciones en tiempo real con una librería de bloques predeterminados y la posibilidad de escribir código en notas para después ser compilado.

La principal diferencia de Scilab con respecto a Matlab es la cantidad de toolboxes que este último incorpora así como su constante optimización en el rendimiento gracias a la financiación por la venta de su licencia de uso profesional que permite a MathWorks mantener el software en constante actualización. Por el contario las ventajas que presenta Scilab son su uso gratuito y la posibilidad de poder ser utilizado con un propósito académico en la enseñanza incorporando las mismas funcionalidades que Matlab con un rendimiento similar para simular modelos de complejidad media. Esto permite al usuario por una parte recrear el comportamiento de fenómenos físico-matemáticos con un propósito de aprendizaje de la materia en sí, como familiarizarse con el entorno y la metodología de trabajo de Matlab con miras a su posterior uso para proyectos que requieran de grandes cálculos tanto en el entorno empresarial como en investigación.

Se puede observar como en la interfaz principal se muestran el tamaño y valor de las variables utilizadas, un historial de comandos, la pantalla de operaciones y los archivos existentes en directorio en el cuál se está trabajando:

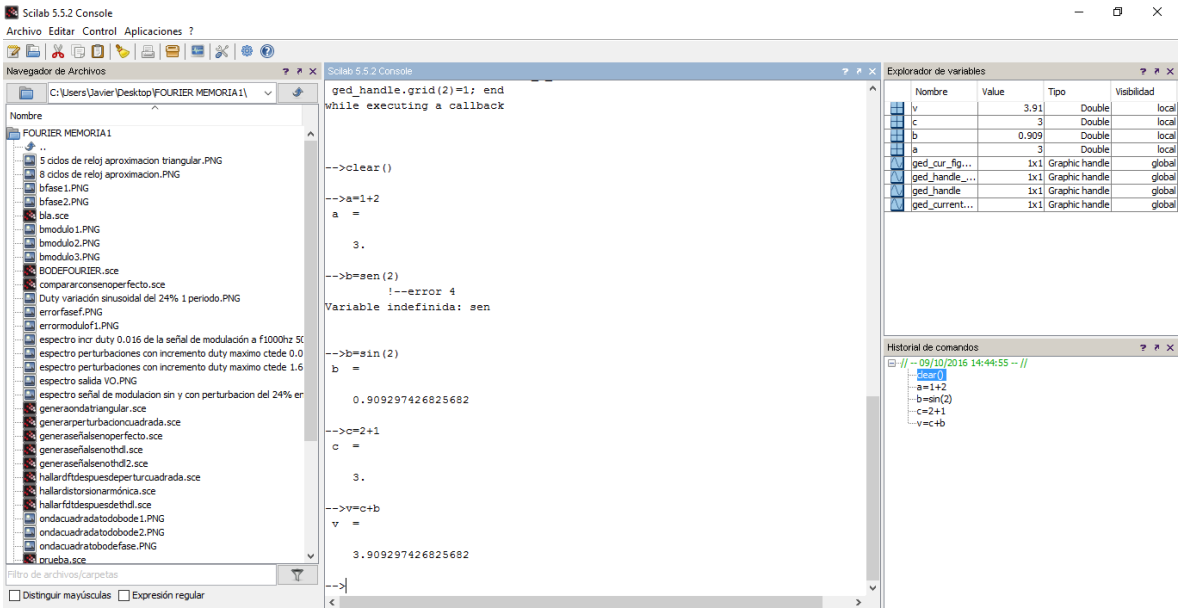

Figura 8 .Interfaz de Scilab.

Con respecto a las posibilidades que ofrece Scilab a la hora de simular una etapa de potencia se encuentran las librerías propias que describen un modelo en el dominio de Laplace y que han sido empleadas en este trabajo y además una librería programada en el lenguaje Modélica [8] para simular la etapa a nivel circuital. Para poder ejecutarla es necesario disponer de un compilador adicional de lenguaje C o Fortran, el cuál puede ser adquirido fácilmente descargando la versión 2012 de Visual Basic Studio como se indica en [9]. Dicha librería incorpora la siguiente paleta de dispositivos electrónicos:

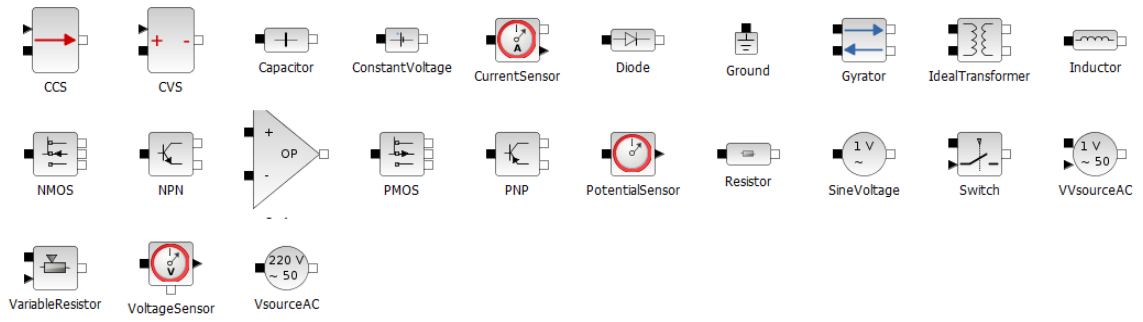

Figura 9. Paleta de bloques para crear modelos circuitales.

De este modo, la etapa empleada en este trabajo ha sido simulada alternativamente basándose en las leyes de Kirchoff:

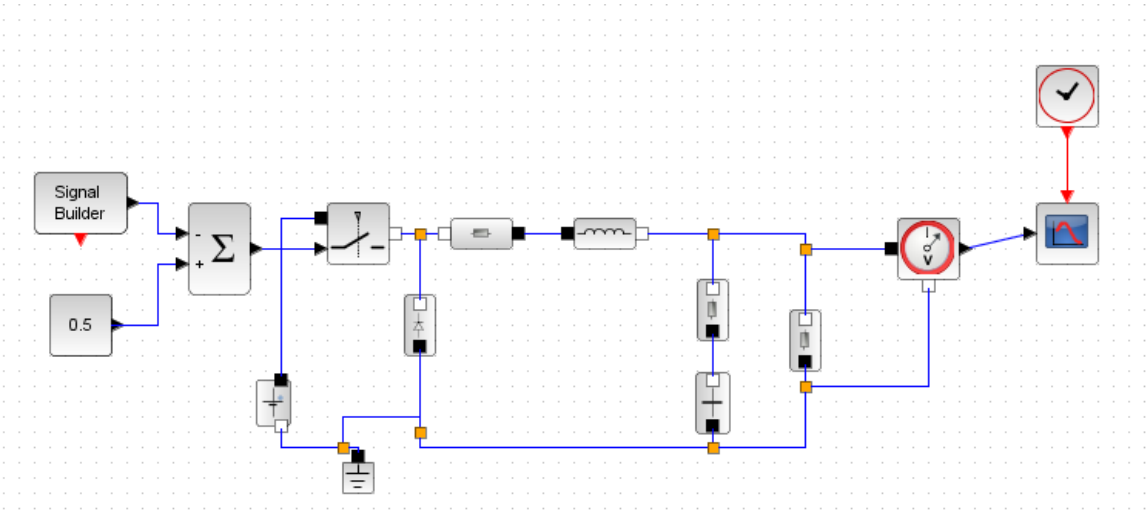

Figura 10. Etapa reductora modelada con bloques electrónicos.

Con unos resultados similares a los obtenidos con el diseño en el dominio 's':

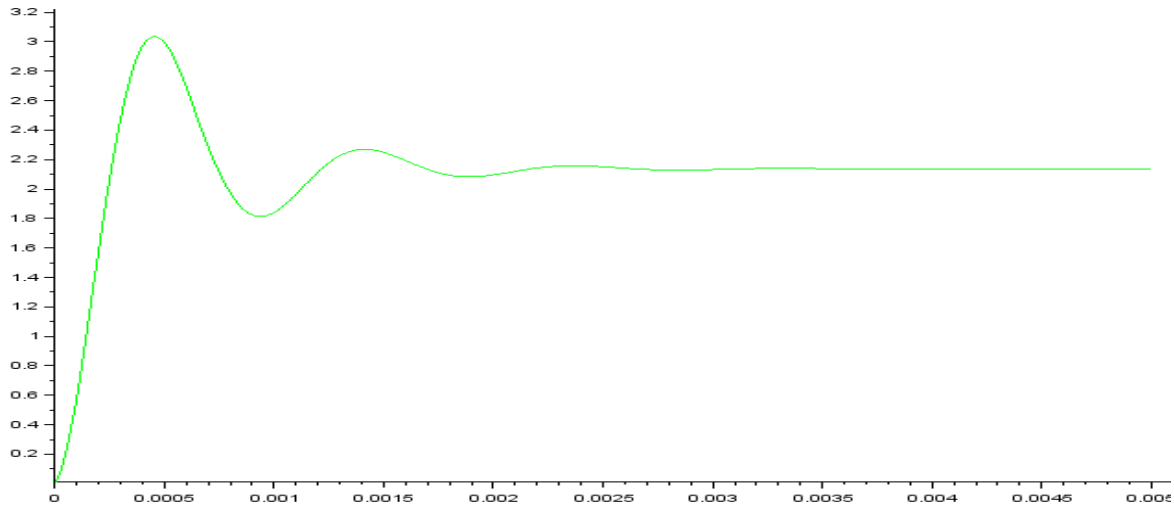

Figura 11. Tensión de salida de la etapa Buck.

Además mediante Modélica es posible crear un código que relacione las variables de tensión e intensidad del mismo por lo que podría simularse la etapa a nivel macro mediante fuentes dependientes que representaran el comportamiento de diodo y transistor [10].

CAPÍTULO 3.

IDENTIFICACIÓN DEL CONVERTIDOR.

3. 1 MÉTODO DE LA DFT

El primer método de identificación analizado se basa en inyectar en el ciclo de trabajo del convertidor una perturbación formada por una onda sinusoidal o una suma de sinusoides de baja amplitud a frecuencias próximas a las de la frecuencia de cruce del sistema en bucle abierto, cuándo la ganancia de este es de 0 dB, con el fin de obtener un modelo del sistema con el objetivo de poder diseñar un controlador para el sistema en bucle cerrado.

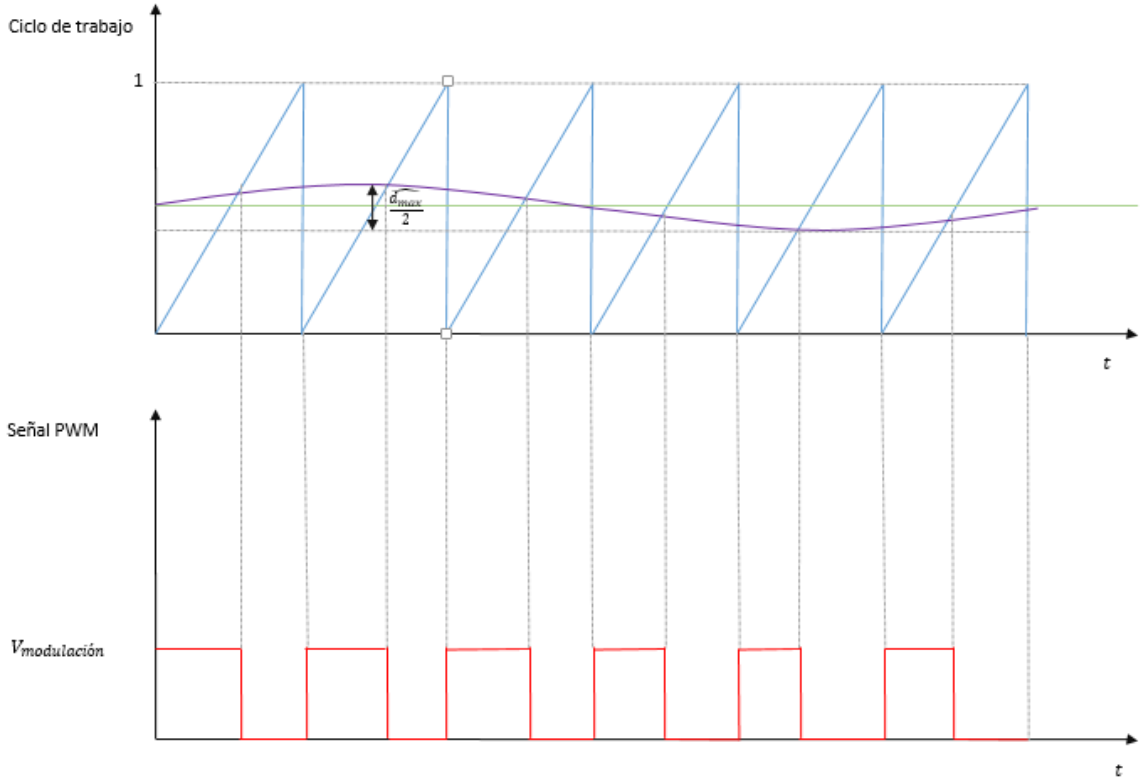

Figura 12. Variación sinusoidal de pequeña señal en el ciclo de trabajo.

El valor de la función de transferencia en módulo y fase a la frecuencia de la perturbación introducida se puede obtener directamente aplicando la DFT a los valores de la entrada y la salida del sistema a lo largo de un número finito de periodos de la perturbación.

El método se fundamenta en el funcionamiento en régimen permanente del Buck, que al actuar como un filtro paso bajo, elimina los armónicos a frecuencias de orden de magnitud superior a las de corte del filtro, como son las de la señal de modulación en condiciones no perturbadas.

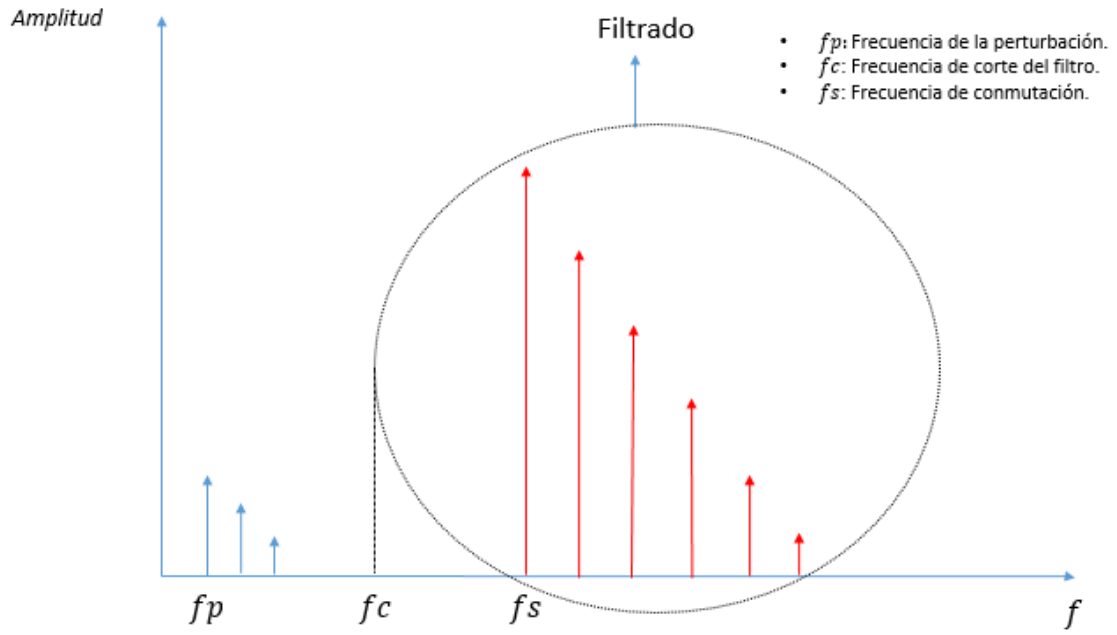

Figura 13. Armónicos de la señal de modulación PWM aplicando una perturbación sinusoidal

Los armónicos de la perturbación son añadidos a la señal de modulación, lo que permite que la salida del sistema esté formada, además de por una componente continua resultante del filtrado de los armónicos a altas frecuencias, de componentes sinusoidales que permiten identificar la respuesta del sistema para cada frecuencia de las perturbaciones inyectadas.

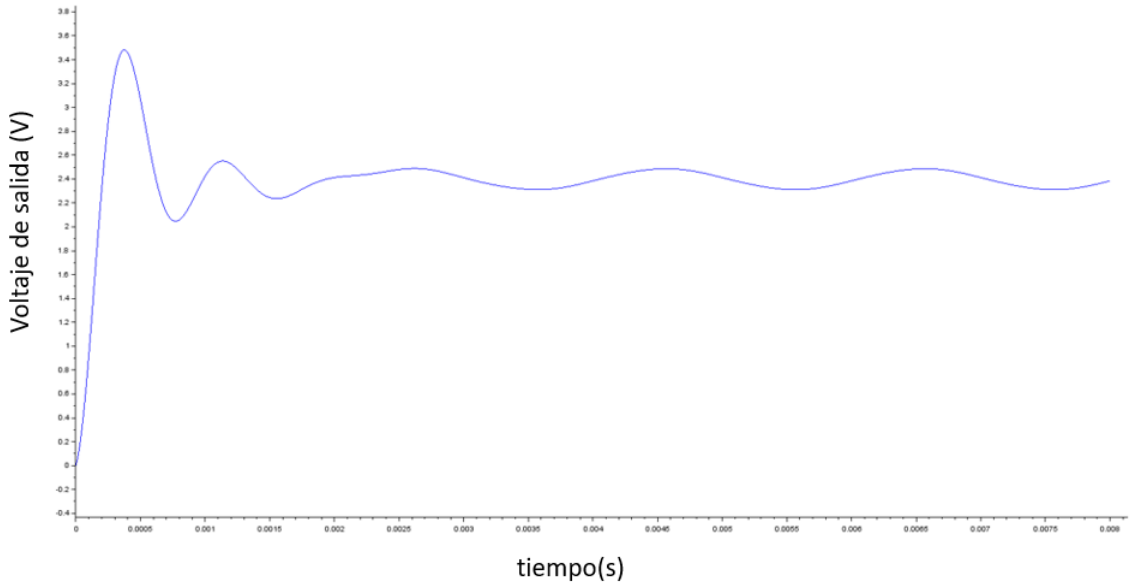

Figura 14. Tensión de salida del sistema habiendo inyectado una perturbación sinusoidal en la entrada.

La perturbación, para ser generada de forma digital mediante FPGA, requiere discretizar la señal que se quiere reproducir a partir de un rango limitado de valores determinado por el número de ciclos de reloj correspondientes a incrementos pequeños del ciclo de trabajo. De este modo, un incremento del 1% en el valor del ciclo de trabajo se corresponde a 5 ciclos de reloj de tiempo añadido en el que la señal de modulación está en alto, siendo 500 el número total de ciclos en un periodo de conmutación para las frecuencias de conmutación del sistema y de reloj de la FPGA empleadas de 100 Khz y 50 Mhz.

En un principio, y con el objetivo de simplificar la obtención de un modelo del sistema en un amplio rango de frecuencias en torno a la de cruce se ha considerado la opción de introducir una señal a baja frecuencia formada por un número infinito de sinusoides tal como una onda cuadrada cuadrada o triangular y aplicar la DFT a las frecuencias correspondientes al armónico fundamental de la señal y sus múltiplos tanto en la entrada como en la salida del sistema.

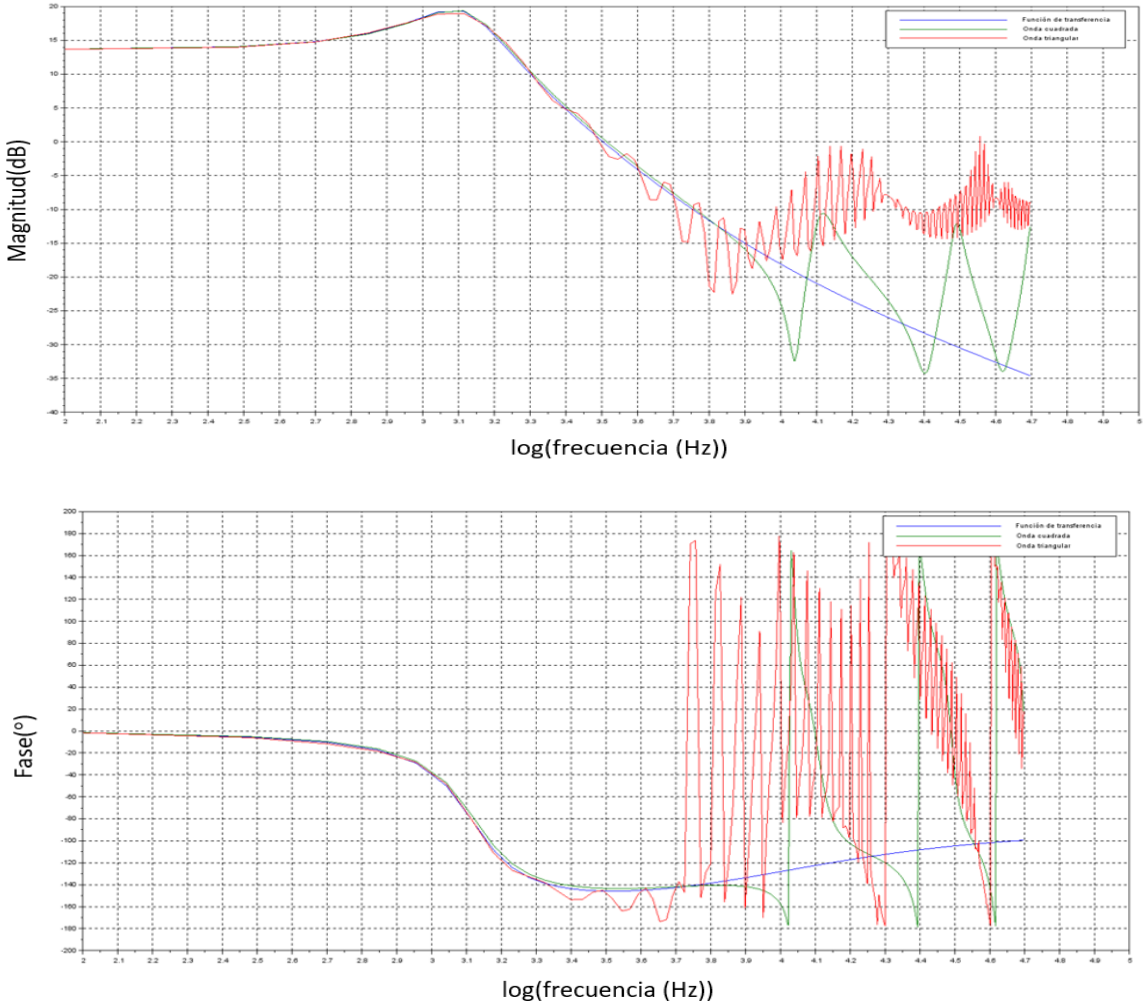

Figura 15. Diagrama de Bode obtenido con una única señal cuadrada y triangular.

El problema de este método residía en que, al tener la onda introducida una amplitud muy reducida y decaer la amplitud de sus armónicos de forma proporcional al inverso de la frecuencia

en la onda cuadrada, y de forma proporcional al inverso del cuadrado de la frecuencia en la onda triangular, a valores elevados de frecuencia el error en la identificación aumentaba drásticamente, lo cual ocurría de forma más pronunciada en el caso de la onda triangular.

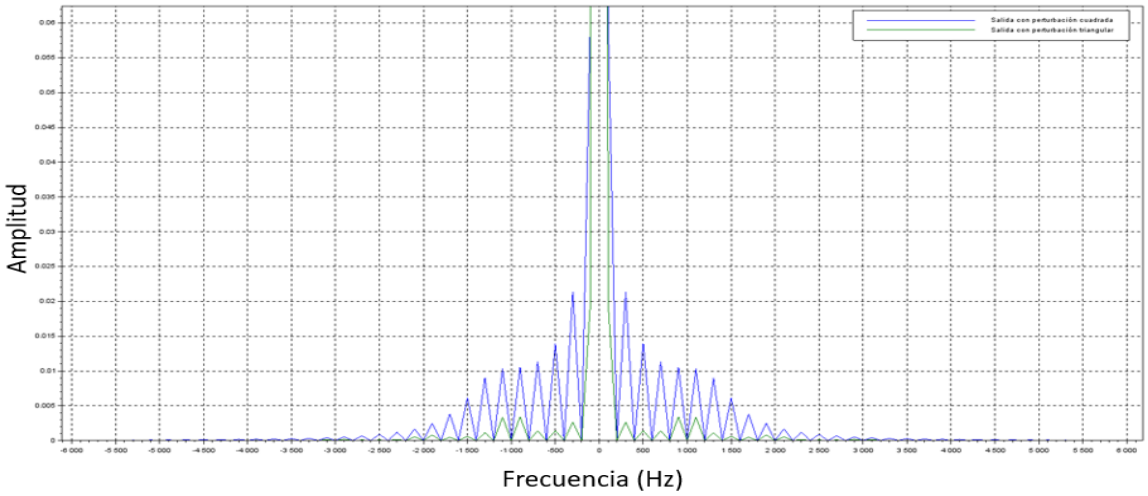

Figura 16. Amplitud de los armónicos en la salida para una onda triangular y cuadrada.

La alternativa escogida ha consistido en introducir para cada frecuencia a la cual se quería obtener el valor de la función de transferencia una onda sinusoidal o cuadrada (analizando únicamente su armónico fundamental). De esta forma se evitaba que la atenuación de los armónicos comprometa la identificación.

En el caso de la onda sinusoidal, esta ha podido ser generada con miras a ser implementada en un futuro de forma digital mediante valores discretos partiendo de un número limitado de ciclos de reloj equivalentes a la amplitud de la onda continua tomada como referencia para redondear los valores de esta al número entero de ciclos de reloj más próximo.

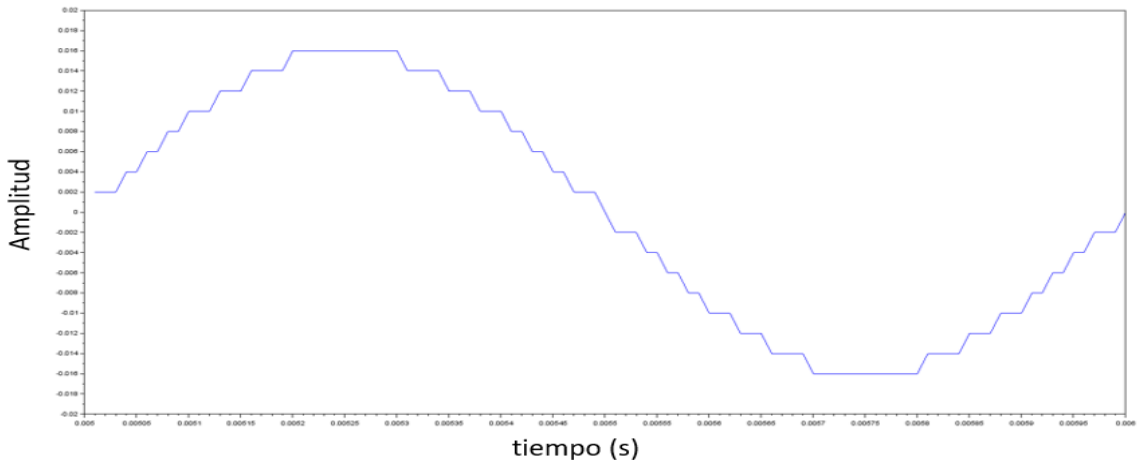

Figura 17. Onda sinusoidal discretizada mediante 8 valores enteros en amplitud.

El grado de aproximación de la perturbación a una onda senoidal mediante valores enteros está relacionada con la distorsión armónica causada por los armónicos múltiplos del fundamental. La distorsión armónica representa la potencia total de la onda presente en los armónicos que no son el fundamental y viene dada por la expresión:

$$THD(\%) = \frac{\sqrt{{V_{rms}}^2 - {V_1}^2}}{{V_1}^2} \qquad (12)$$

Donde V_{rms} es el valor eficaz de la señal, V_1 el valor eficaz del primer armónico y THD la distorsión armónica total.

De este modo, a mayor número de ciclos de reloj la aproximación mejoraba y por tanto el valor de estos armónicos adicionales se veía reducido. Asimismo, el valor en módulo del armónico fundamental de la onda se aproximaba más al de una onda sinusoidal de la misma amplitud así como al armónico fundamental de una onda cuadrada de amplitud $\frac{\pi}{4}$ la de la senoide de referencia.

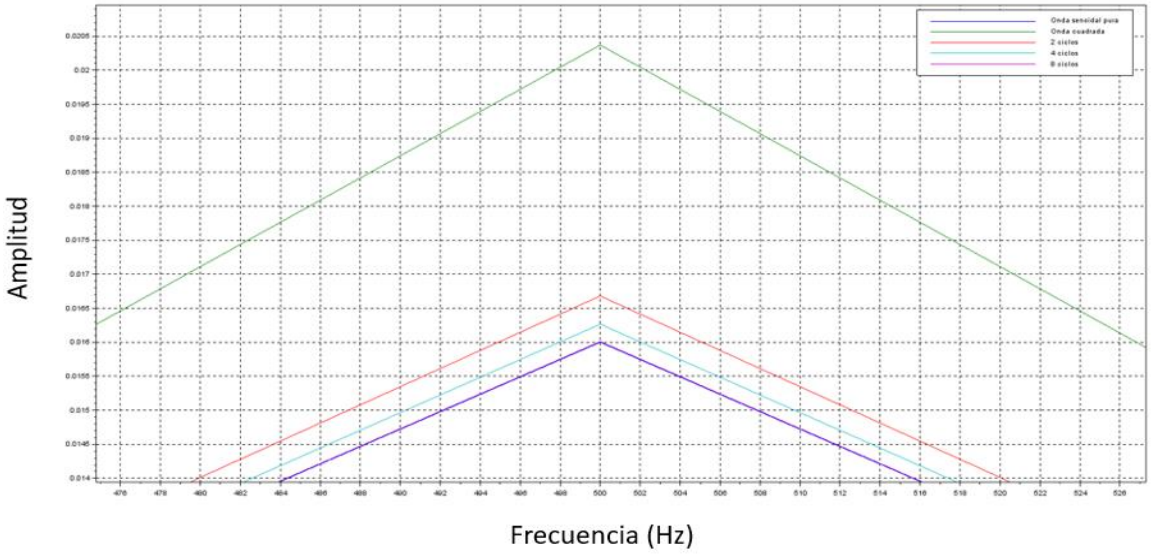

Figura 18. Amplitud de las distintas ondas generadas mediante aproximación con distintos números de ciclos de reloj.

De este modo, el número de ciclos de reloj máximos empleados para generarla señal se ha escogido estableciendo un compromiso entre la exactitud en la aproximación a una senoide pura y el hecho de que un número de ciclos elevado provocaría una amplitud superior a la que es necesaria para obtener un modelo en pequeña señal. Se ha establecido que ésta no debía superar el 2%.

Para elegir el número de ciclos se ha calculado por tanto el porcentaje de error del primer armónico de la perturbación respecto del de una sinusoide pura para 2, 4 y 8 ciclos de reloj en amplitud a una frecuencia de la perturbación de 500 Hz. A su vez se ha calculado la distorsión

armónica para cada uno de los tres casos.

Ciclos de reloj /Armónico	1º	3º	5º	7º	THD (%)	Error (%)
2	$4.170 \cdot 10^{-3}$	$3.525 \cdot 10^{-4}$	$5.895 \cdot 10^{-4}$	$8.807 \cdot 10^{-4}$	17.446	4.255
4	$8.133 \cdot 10^{-3}$	$1.886 \cdot 10^{-4}$	$2.05 \cdot 10^{-5}$	$2.026 \cdot 10^{-4}$	9.304	1.1663
8	0.016	$1.912 \cdot 10^{-4}$	$3.08 \cdot 10^{-5}$	$1.27 \cdot 10^{-5}$	4.955	0.06

Tabla 1. Magnitud de los primeros armónicos, distorsión armónica y error en amplitud para distintos números de ciclos de reloj.

De este modo, se ha determinado que la mejor aproximación a la onda sinusoidal era la generada con 8 ciclos de reloj, fluctuando de este modo para un ciclo de trabajo medio de 0.5 el valor instantáneo de éste entre 0.484 y 0.516 o expresado en ciclos de reloj tras los cuáles la señal PWM pasaba de alto a bajo 242 y 258 respectivamente.

Para crear la perturbación se ha redondeado al entero más próximo el valor de un número de muestras tomadas a partir de la señal sinusoidal de amplitud 8 (equivalente al número de ciclos de reloj escogidos) para un cuarto de periodo.

Este valor ha sido escogido por dos razones con miras a la implementación en la FPGA:

En primer lugar, al ser generados experimentalmente los valores de la perturbación sinusoidal mediante una tabla de valores resultantes de muestrear la señal continua para un cuarto de periodo debía cumplirse que:

$$f_{perturbación} = \frac{1}{(N \cdot t_{clk} \cdot 4 \cdot n)} \tag{13}$$

, siendo N el número de valores de la tabla y n el número de ciclos de reloj de periodo t_{clk} tras los cuáles se tomaba el siguiente valor de la tabla ambos números enteros.

Por otra parte, teniendo en cuenta que el periodo de la perturbación introducida debía ser múltiplo del de conmutación con el fin de que las diferentes variaciones introducidas en el ciclo de trabajo en cada ciclo de conmutación fueran idénticas a lo largo de un número finito de periodos de la perturbación analizados se debía cumplir la siguiente relación:

$$p \cdot f_{perturbación} = f_{conmutación} \tag{14}$$

, siendo p asimismo un número entero.

De este modo dado que el tiempo de reloj de la FPGA es de 20 ns y con el fin de poder generar frecuencias de la perturbación suficientemente altas se ha determinado la cifra de 50 valores de la tabla como adecuada para realizar la aproximación.

La frecuencia de muestreo escogida ha sido la de conmutación, ya que a frecuencias de muestreo superiores la mínima resolución en frecuencia para un mismo número de muestras tomadas disminuía y además aumentaba el valor del límite superior del rango analizado de frecuencias, lo cual no era deseable ya que la identificación a frecuencias cercanas a la de conmutación no era efectiva debido al filtrado de sus respectivos armónicos ni interesaba para el análisis. Frecuencias de muestreo inferiores mejoraban por otra parte la resolución pero disminuían el rango de análisis.

De este modo, el número N de muestras tomadas durante cada periodo de la perturbación se ha determinado por la relación $N = \frac{f_s}{f_{per}}$ con la mínima resolución en frecuencia igual a $\Delta f = \frac{f_s}{N} = f_{per}$, correspondiente al primer armónico a analizar tanto en la entrada (la perturbación) como en la salida del sistema y f_s y f_{per} la frecuencia de muestreo y de la perturbación, respectivamente.

Debido a que en la práctica no es posible generar mediante la FPGA directamente mediante diferentes valores de tensión la perturbación para ser medida y registrada, el procedimiento para hallar el valor del armónico fundamental en la entrada ha consistido en generar una onda cuadrada sincronizada con la perturbación y escalar su valor en amplitud para hallar la amplitud del primer armónico.

3.2 MÉTODO DE LA CORRELACIÓN CRUZADA

El siguiente método estudiado consiste en introducir una perturbación binaria pseudoaleatoria (PRBS) en el sistema que simule el comportamiento de una entrada compuesta por ruido blanco de desviación típica igual a la magnitud de la perturbación introducida. La perturbación produce un solo valor fijo de incremento en el ciclo de trabajo con signo variable haciendo que éste varíe en torno al valor constante que tomaría en ausencia de la perturbación en cada ciclo de conmutación.

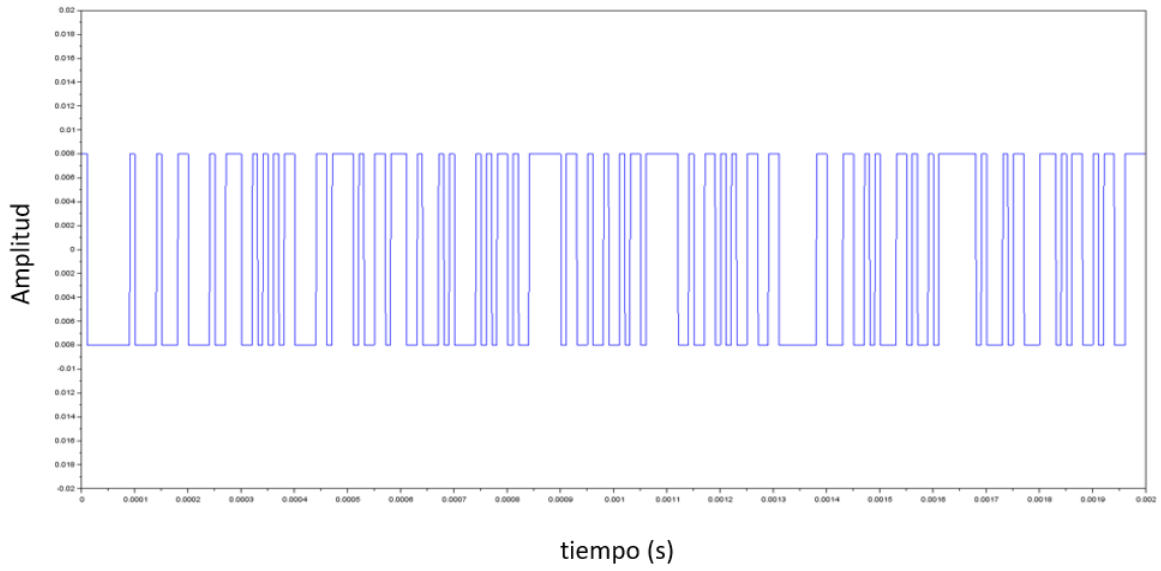

Figura 19. Perturbación pseudoaleatoria introducida en el ciclo de trabajo.

La perturbación se puede descomponer en una suma finita de senoides de la misma amplitud y fase distribuida aleatoriamente a frecuencias comprendidas entre $f = \frac{f_{sw}}{N}$ y $f = \frac{f_{sw}}{2}$, con N el número de elementos o valores que componen la secuencia y que se repiten en cada periodo de ésta y $f_{sw} = 100\ Khz$ la frecuencia de conmutación, que es igual a la que estos elementos se alternan para que las variaciones introducidas en el ciclo de trabajo en cada periodo de conmutación se produzcan en el mismo momento.

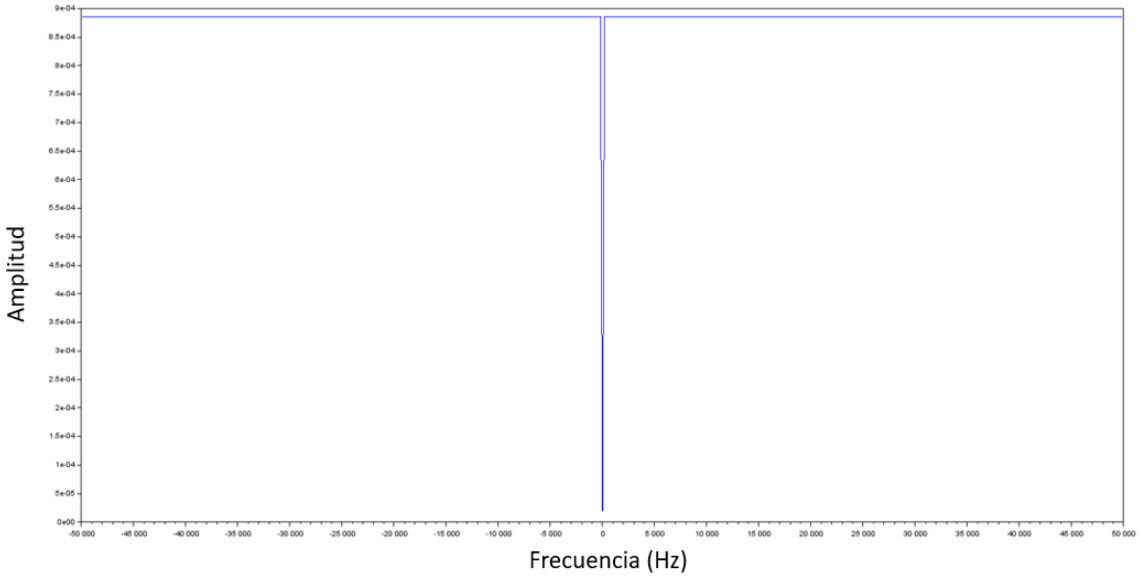

Figura 20. Espectro de un periodo de la secuencia pseudoaleatoria.

La función de transferencia $G(s)$, dado que los armónicos no ven disminuido su valor en amplitud a lo largo del rango de frecuencias analizadas puede obtenerse con gran precisión directamente

aplicando la DFT a los valores $u(kT)$ de la perturbación y de la salida $y(kT)$ del sistema. Para ello, la frecuencia de muestreo escogida debe ser igual a la de conmutación para caracterizar correctamente la perturbación obteniéndose la respuesta en frecuencia del sistema como:

$$G(\omega j) = \frac{y(\omega j)}{u(\omega j)} \tag{15}$$

Los valores de la entrada, al contrario que en el método anterior que involucraba una perturbación senoidal, pueden ser tomados directamente de la perturbación ya que es viable generar ésta digitalmente dado que únicamente alterna un par de valores distintos correspondientes a dos tensiones distintas de salida de la FPGA : un valor no nulo de tensión para un incremento positivo y un valor nulo de para un incremento negativo del ciclo de trabajo respectivamente. Al tener la tensión de salida del modulador un valor fijo, ésta sólo aporta información acerca de cuándo tiene lugar un incremento positivo o negativo pero no da información sobre su magnitud, siendo ésta tenida en cuenta a la hora de aplicar el algoritmo sobre los resultados experimentales.

La secuencia es generada por un registro de bits retroalimentado cuya longitud determina el número de elementos de la secuencia generada, los cuales se repiten en el mismo orden en cada periodo de la misma. Este número de elementos viene dado por la relación $N = 2^L - 1$, siendo L el número de bits del registro. El bit de salida, cuyo valor determina el signo del incremento introducido en el ciclo de trabajo en cada periodo de conmutación , se encuentra en un extremo del registro. Éste es comparado con uno o varios de los demás bits del registro mediante puertas lógicas XOR para determinar el valor del bit futuro de entrada del mismo, el cuál se encuentra en el extremo opuesto al de salida, en el periodo de conmutación siguiente. El resto de bits al acabar el ciclo son desplazados una posición hacia el extremo de la salida, siendo el bit correspondiente a ésta sustituido por su adyacente. Los valores iniciales pueden ser escogidos al azar mientras no todos los bits sean '0', permaneciendo el orden de los valores de la perturbación inalterado.

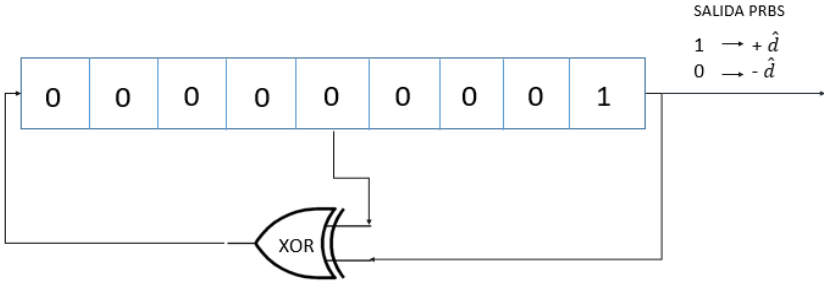

Figura 21. Registro generador de 9 bits de la secuencia aleatoria.

Las posiciones de los bits que deben ser comparados en cada ciclo se mantienen constantes a lo largo de todos los elementos de la secuencia y corresponden a los elementos no nulos del

polinomio primitivo cuyo orden es igual al número de bits del registro.

Los polinomios primitivos son aquellos cuyo máximo divisor de sus coeficientes es '1'. Son bastante empleados a la hora de implementar secuencias pseudoaleatorias. En general, el interés en este tipo de polinomios se centra en los trinomios primitivos, los cuáles solamente incorporan 3 términos no nulos. De este modo, el número de puertas lógicas para generar la secuencia se ve reducido.

Orden	**Polinomio primitivo**
7	$x^7 + x^3 + 1$
8	$x^8 + x^4 + x^3 + x^2 + 1$
9	$x^9 + x^4 + 1$
10	$x^{10} + x^3 + 1$
11	$x^{11} + x^2 + 1$

Tabla 2. Polinomios primitivos de orden 7 al 11 .

Si bien es cierto que el método de la aplicación de la DFT a los valores de la entrada y la salida del convertidor proporciona un modelo bastante aproximado del sistema, la exactitud alcanzada es mayor al aplicar éste a la respuesta impulsional unitaria temporal del convertidor obtenida a partir del método de la correlación cruzada aplicando éste a los valores de la entrada y la salida del sistema a lo largo de un número entero de periodos de la secuencia pseudoaleatoria y obteniendo el vector respuesta al impulso unitario por medio de la relación entre ésta y la autocorrelación de los valores de la perturbación.

La salida del sistema en tiempo discreto y puede ser obtenida como la convolución de la señal de entrada u con su respuesta al impulso unitario h más un término v que incluye todo tipo de ruido ajeno a la perturbación introducida:

$$y(n) = \sum_{n=1}^{\infty} h(k)u(n-k) + v(n) \qquad (16)$$

A su vez, la correlación entre la entrada y la salida del sistema puede ser obtenida mediante la expresión:

$$R_{uy}(m) = \sum_{n=1}^{\infty} u(n)y(n+m) \qquad (17)$$

o alternativamente como la convolución de la respuesta al impulso unitario con la autocorrelación de la señal de entrada más la correlación entre ésta y el ruido v que es introducido de forma adicional al sistema:

$$R_{uy}(m) = \sum_{n=1}^{\infty} h(n)R_{uu}(m-n) + R_{uv}(m) \tag{18}$$

Este último término puede ser despreciado si se tiene en cuenta que la perturbación introducida tiene un comportamiento similar al ruido blanco. Así pues, dado que en la práctica el número de valores registrados es finito, las funciones correlación cruzada y autocorrelación pueden ser aproximadas y posteriormente normalizadas en función del número de muestras tomadas como:

$$R_{uv}(m) = \frac{1}{N}\sum_{n=1}^{N-1} h(n)y(m+n) \tag{19}$$

$$R_{uu}(m) = \frac{1}{N}\sum_{n=1}^{N-1} u(n)u(m+n) \tag{20}$$

con $m = 0,1,\dots,N-1$ y N número total de muestras que debe ser múltiplo del número de elementos de la secuencia pseudoaleatoria.

De este modo la función autocorrelación de la entrada toma los valores

$$R_{uu}(m) = \begin{cases} e^2, para\ \ m = p \cdot M\ \ con\ \ p = 0,1,2..y\ M = 2^L - 1 \\ \dfrac{-e^2}{N}, \qquad para\ el\ resto\ de\ valores \end{cases} \tag{21}$$

Donde M es el número total de elementos de la secuencia y p es el número de periodos.

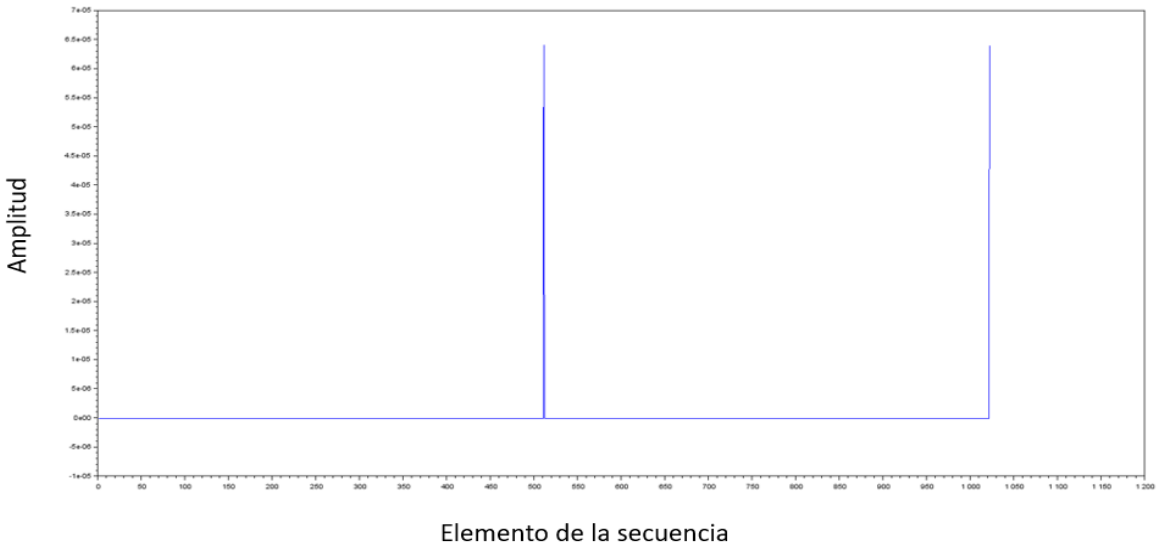

Figura 22. Función de autocorrelación para dos periodos de la secuencia PRBS con un registro de 9 bits.

En consecuencia ,dado que el número de muestras tomado en un periodo es bastante elevado al aumentar de forma exponencial con el tamaño del registro los términos del vector autocorrelación cuando $m \neq p \cdot M$ pueden ser despreciados quedando (19) de la forma

$$R_{uy}(m) \cong h(m) \sum_{i=0}^{p-1} R_{uu}(M \cdot i) = h(m) \cdot pR_{uu}(0) \qquad (22)$$

y por tanto el vector respuesta al impulso unitario puede ser obtenido como

$$h(m) = \frac{R_{uy}(m)}{pR_{uu}(0)} \qquad (23)$$

La función de transferencia puede ser obtenida de esta forma con más precisión aplicando directamente aplicando la DFT al vector resultante obtenido de la respuesta impulsional en el dominio del tiempo, siendo la resolución mínima mejorada a mayor longitud del registro y por tanto la precisión en la identificación mayor.

$$F\big(h(m)\big) \rightarrow G(\omega j)$$

3.3 RESULTADOS DE LA SIMULACIÓN

Se ha calculado en Scilab la función de transferencia para 19 frecuencias submúltiplos de la frecuencia de conmutación con el método de la DFT comprendidas entre 100 y 16666 Hz, al ser las discrepancias entre los valores medidos y los reales en la identificación para ambos métodos válidas para un valor de frecuencia menor que fs/5 [11]. El rango de frecuencias se ha situado en torno a la frecuencia de cruce del sistema con el fin de reconstruir el diagrama de Bode que caracteriza su respuesta frecuencial.

Para ello se han introducido en el ciclo de trabajo del convertidor de forma independiente una perturbación cuadrada y tres senoidales aproximadas por distinto número de ciclos de reloj en amplitud. El valor medio del Duty, sobre el cuál se ha inyectado la perturbación se ha establecido en 0.5. Los valores de la entrada se han tomado a partir de una onda cuadrada sincronizada con la perturbación.

Los valores obtenidos para la identificación de la magnitud en dB de la respuesta del sistema se muestran a continuación

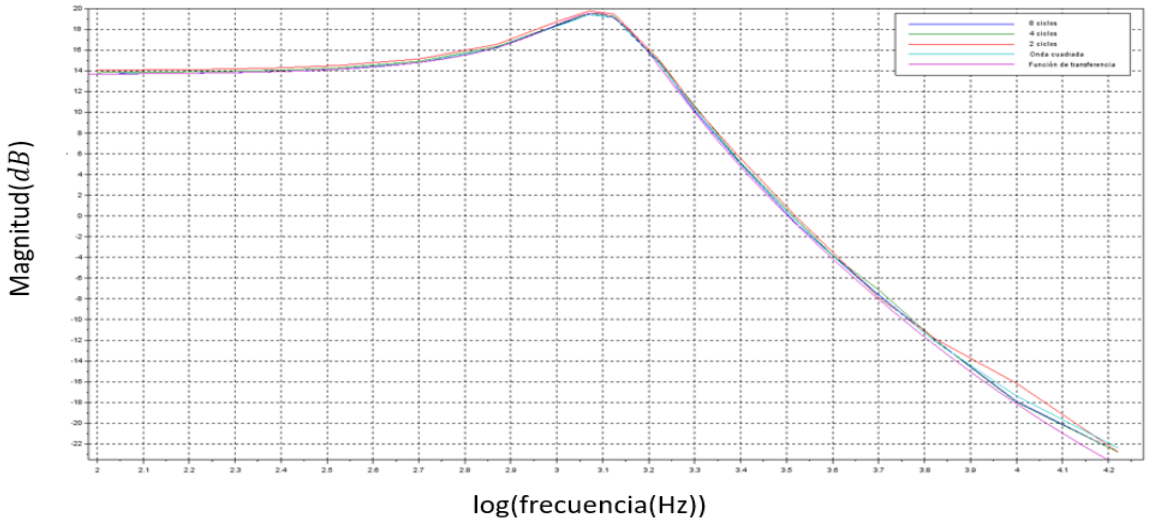

Figura 23 .Magnitud de la respuesta en frecuencia con el método de la DFT.

Los valores de la fase a distintas frecuencias han sido los siguientes:

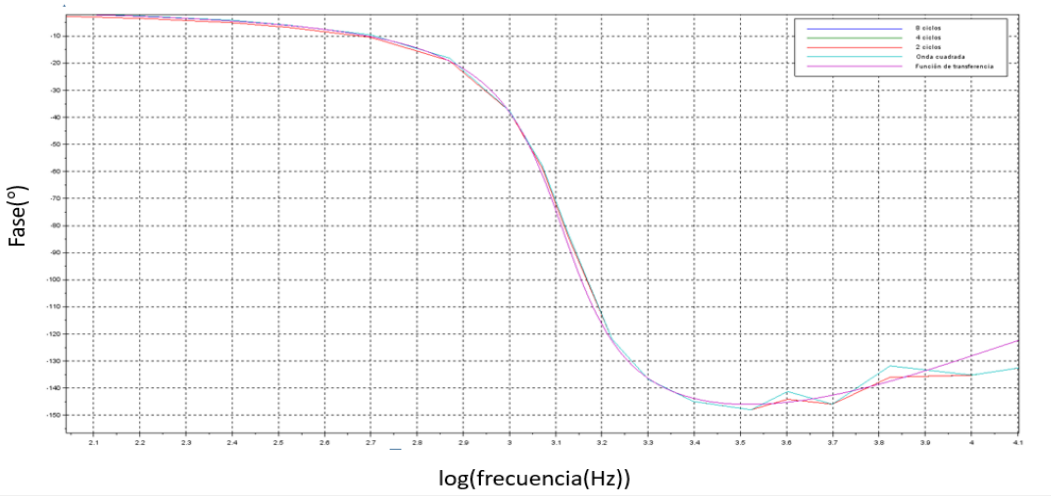

Figura 24. Fase de la respuesta en frecuencia con el método de la DFT.

El error en la caracterización en tanto por ciento en la magnitud en el rango de frecuencias analizado ha sido el siguiente:

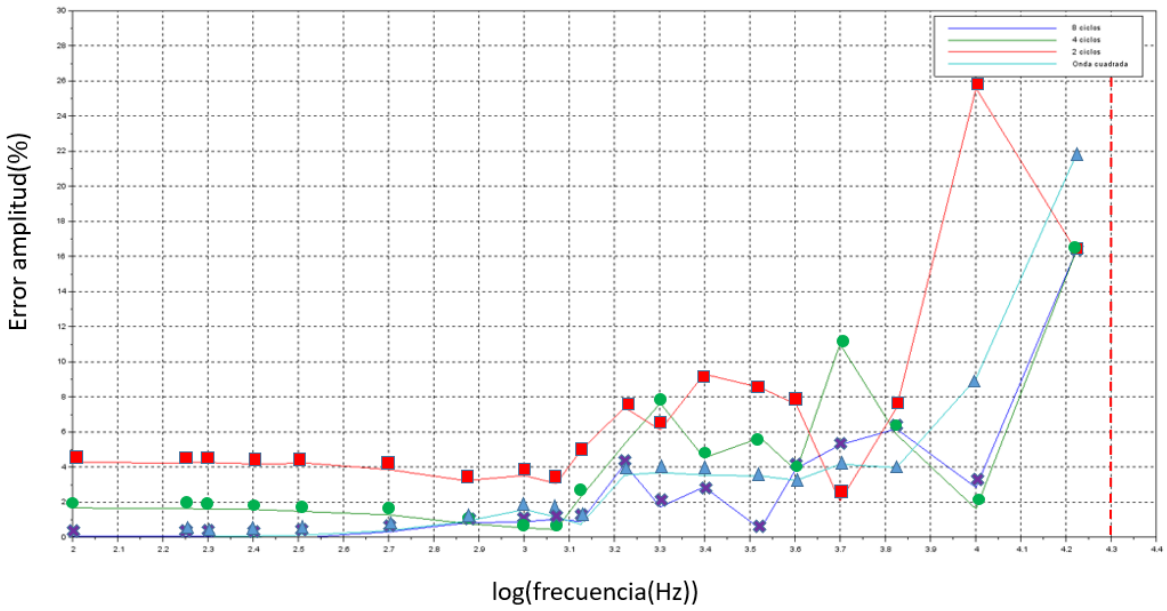

Figura 25. Error en la magnitud de la respuesta en frecuencia respecto a la función de transferencia ideal con el método de la DFT.

El error obtenido para la fase en tanto por ciento se muestra a continuación:

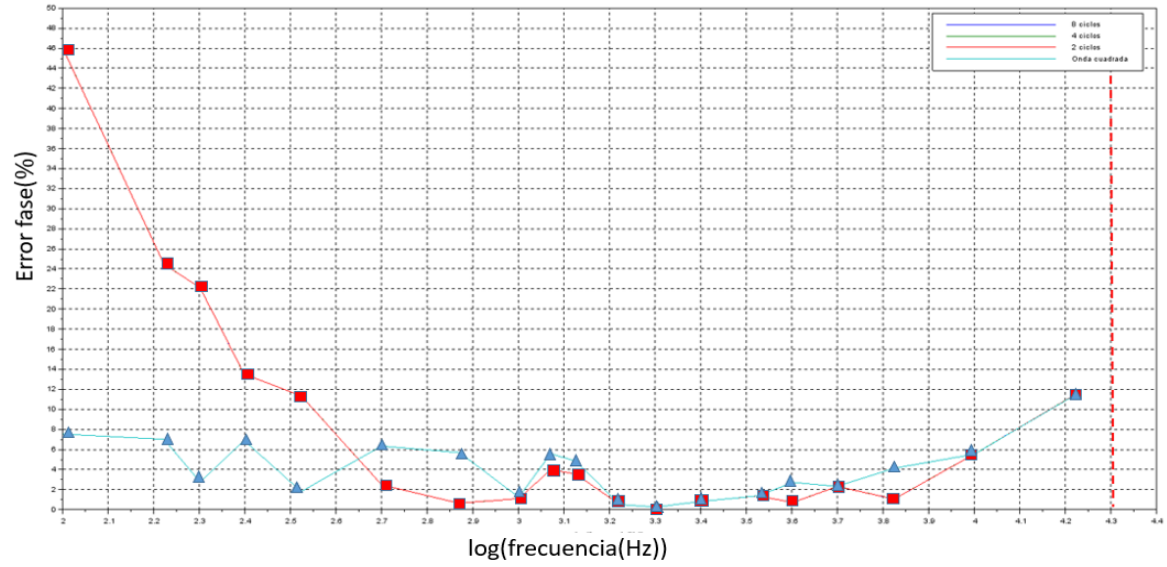

Figura 26. Error en la fase de la respuesta en frecuencia respecto a la función de transferencia ideal con el método de la DFT.

De este modo se ha determinado que las opciones con menos error eran la onda sinusoidal de 8 ciclos y la onda cuadrada.

Por otra parte se ha realizado la simulación aplicando el método de la correlación cruzada generando las secuencias que componen la perturbación con registros de 7, 8, 9 ,10 y 11 bits para un ciclo de trabajo medio de 0.5 y con un valor fijo de incremento y decremento del mismo del 0.8%. Los valores de la entrada han sido tomados directamente de la propia perturbación.

Los resultados en la caracterización de la magnitud del sistema en dB se muestran a continuación:

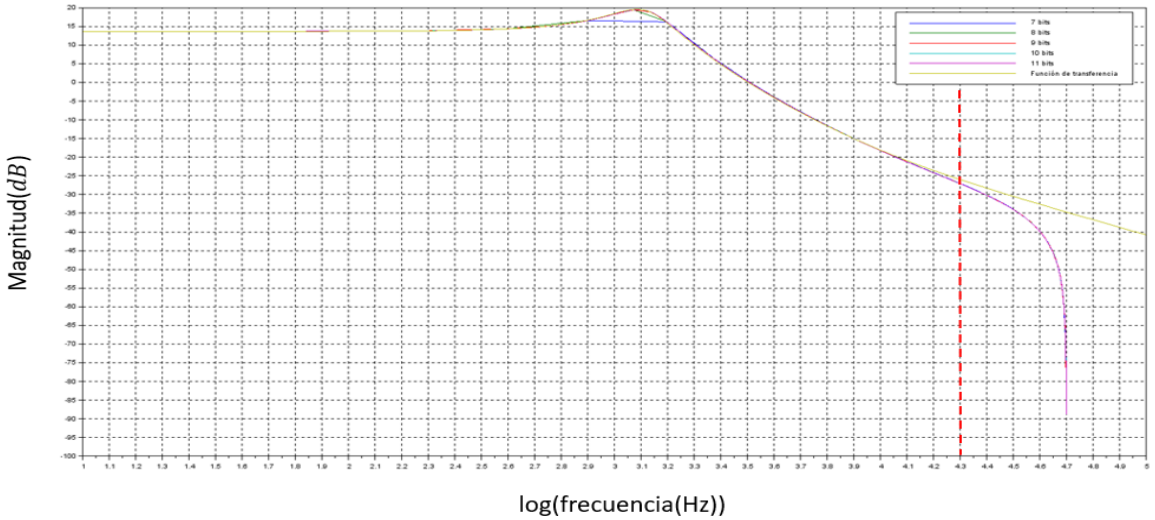

Figura 27. Magnitud de la respuesta en frecuencia con el método de la correlación cruzada.

Los valores de la fase a distintas frecuencias han sido los siguientes:

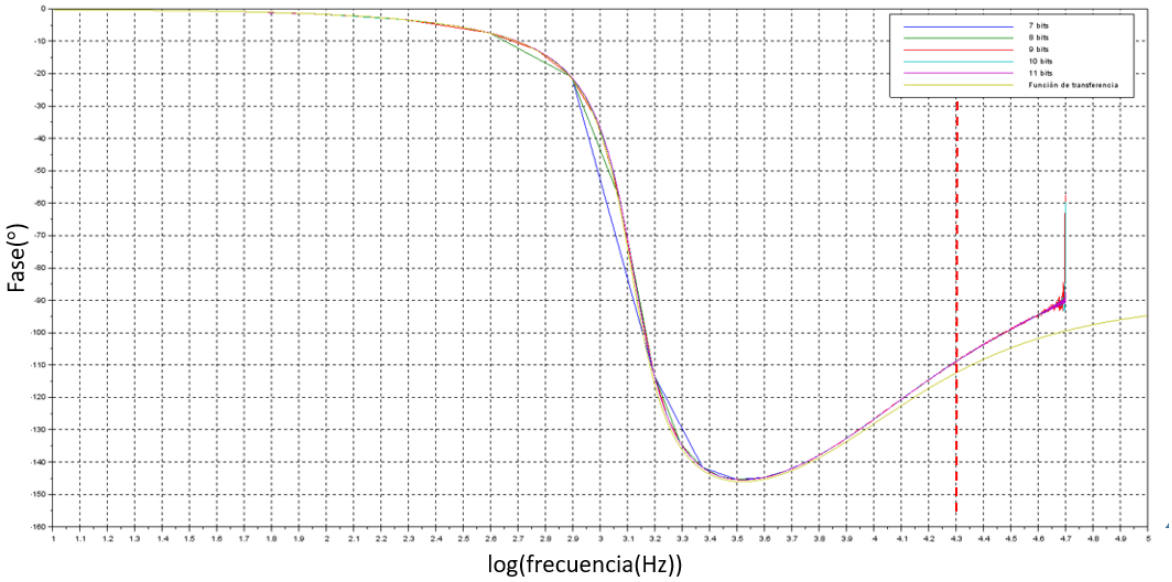

Figura 28. Fase de la respuesta en frecuencia con el método de la correlación cruzada.

El error en tanto por ciento en la caracterización del módulo ha sido el mostrado a continuación:

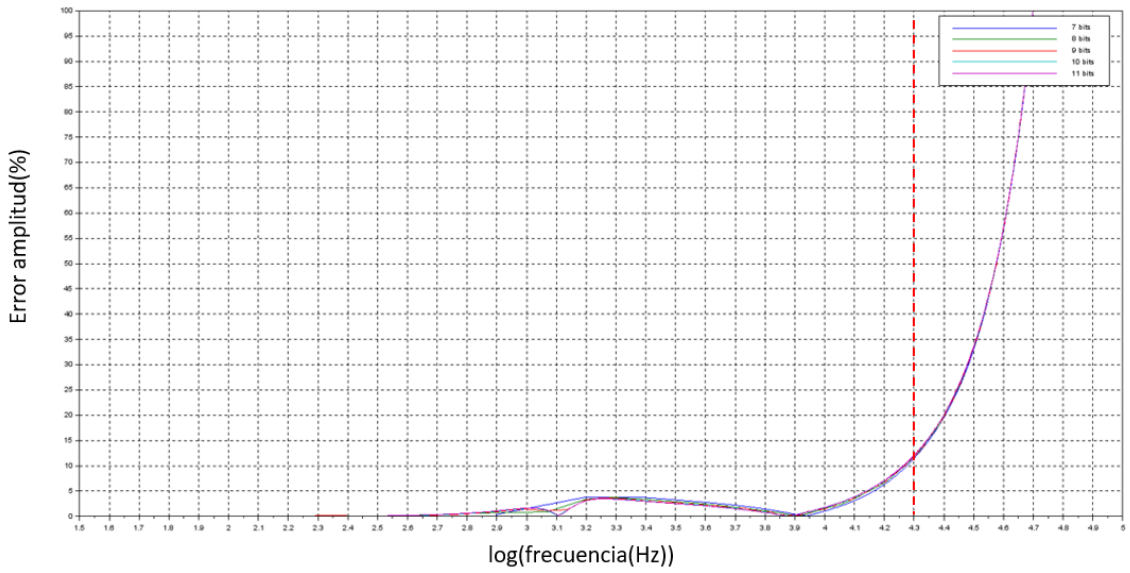

Figura 29. Error en la magnitud de la respuesta en frecuencia respecto a la función de transferencia ideal con el método de la DFT.

Por su parte el error en tanto por ciento para la fase ha sido el siguiente:

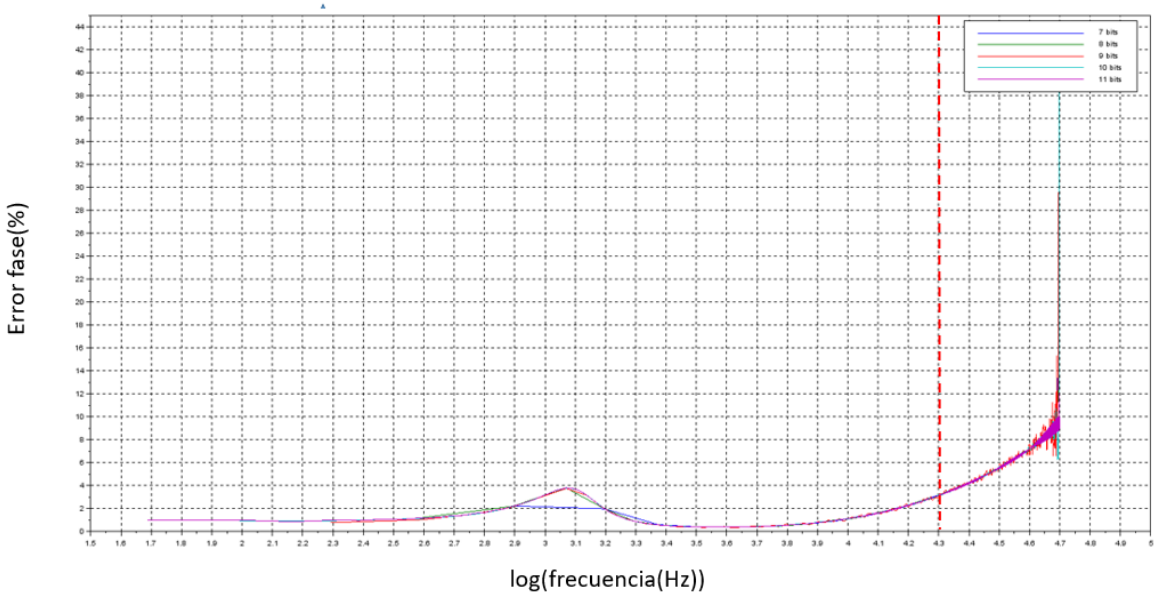

Figura 30. Error en la fase de la respuesta en frecuencia respecto a la función de transferencia ideal con el método de la DFT.

Se ha determinado que el error a frecuencias próximas a las de corte era menor a mayor longitud del registro generador de la secuencia pseudoaleatoria. El rango de frecuencias evaluado ha sido el comprendido entre $\frac{f_S}{N}$ y $\frac{f_S}{5}$ siendo N el número de elementos de la secuencia para cada

longitud de registro, f_s la frecuencia de muestreo empleada y $\Delta f = \frac{f_s}{N}$ la resolución mínima.

La inyección de la perturbación se ha realizado en ambos métodos en el instante del inicio de la modulación con el fin de que el sistema alcanzara una respuesta estacionaria lo más rápido posible . Los valores de la salida del sistema se han tomado una vez alcanzado el régimen permanente. La identificación se ha realizado a lo largo de un periodo. La frecuencia de muestreo se ha establecido en $fs = 100\ Khz$, equivalente a la de conmutación.

CAPÍTULO 4.

IMPLEMENTACIÓN.

4.1 ESQUEMA DEL PROCESO

El proceso comprendido desde la elección de los valores de los parámetros de la modulación hasta el procesamiento de la entrada y la salida del sistema para obtener un modelo del mismo se ha descrito mediante el siguiente diagrama de bloques:

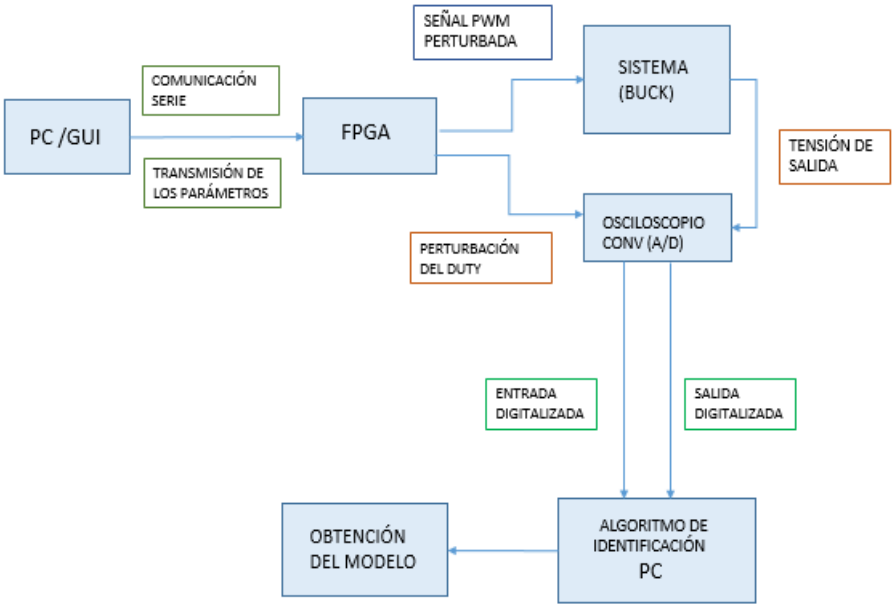

Figura 31. Esquema del proceso de obtención del modelo experimentalmente.

En primer lugar desde una interfaz en el PC se escogen los valores de los parámetros que describen la perturbación que se desea generar y mediante un puerto serie conectado a la FPGA éstos son enviados en cadenas de uno o 2 bytes empleando el protocolo RS-232.Éstos son posteriormente procesados en la FPGA mediante una arquitectura definida en VHDL configurada para recibir los datos al mismo tiempo que éstos son enviados por el puerto serie del PC para generar la señal de salida PWM con la variación introducida en el ciclo de trabajo. La perturbación, así como la salida del sistema son registradas y digitalizadas con un osciloscopio con el fin de ser procesadas mediante software aplicando los algoritmos de identificación descritos en el capítulo anterior para un modelo no paramétrico del sistema.

4.2 INTERFAZ GRÁFICA/GUI:

Para configurar los parámetros de la modulación y enviar la orden de inicio y parada desde el PC a la FPGA se ha elaborado una interfaz gráfica en el lenguaje de programación y entorno de

desarrollo integrado de código abierto Processing, basado en Java [12]. Processing fue creado en el año 2001 por Casey Reas y Ben Fry y está distribuido con licencia GNU GPL con el fin de ser empleado en diversos campos como son electrónica, el arte visual y el diseño gráfico en el contexto de aprendizaje de un lenguaje de programación con la ayuda de referencias visuales.

La interfaz elaborada incorpora para cada parámetro a modular un botón de aumentar y disminuir su magnitud, con lo que éstos pueden ser configurados manualmente haciendo click con el ratón y visualizados en todo momento mediante una serie de pantallas que muestran su valor. Por otra parte, contiene en la parte inferior dos botones, el de START, que tras ser pulsado envía la información en código binario de todos los parámetros que han sido previamente elegidos y seguidamente la orden de inicio de la modulación, y el de STOP, que envía la orden de finalizar el proceso de modulación a la FPGA. A diferencia de una entrada RESET, la orden de STOP enviada desde el PC permite conservar la información de los parámetros en la FPGA.

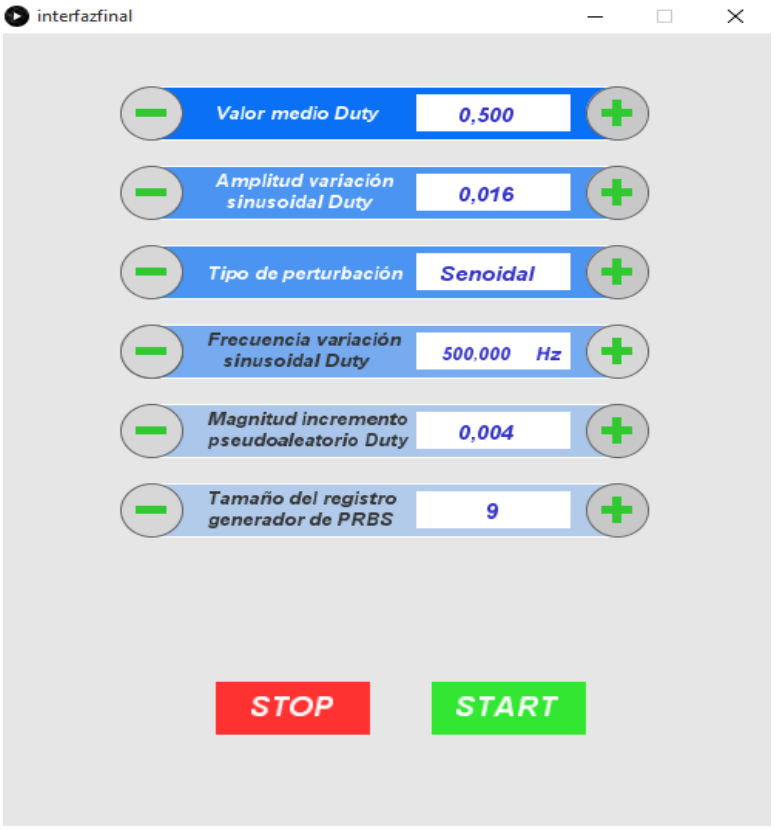

Figura 32. Interfaz empleada para configurar los parámetros de la modulación.

La información de cada parámetro viene contenida en dos bytes. La información viene codificada en complemento a 2. De este modo en ambos bytes sólo se tienen en cuenta los 7 bits restantes enviados. En el caso del primer byte, los 4 bits más significativos de estos 7 restantes determinan a qué parámetro corresponde el valor numérico que se está enviando. Este valor viene determinado por los 3 y 7 bits menos significativos del primer y segundo byte respectivamente.

Para el caso de los parámetros cuyo valor máximo puede ser configurado con solamente 7 bits, el primer byte solo contiene la información de cuál es el parámetro a configurar. Por otra parte, en el caso de las órdenes de inicio y parada de la modulación tan solo se envía un byte al no ser necesario configurar ningún parámetro y toda la información necesaria para identificar la orden se encuentra en los 4 bits más significativos sin tener en cuenta el bit de signo.

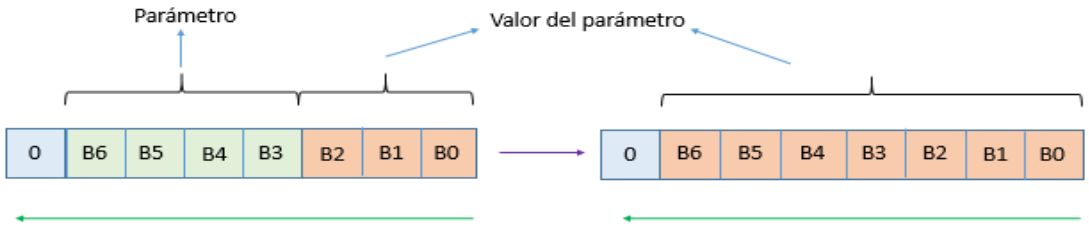

Figura 33. Modo de codificación de los parámetros para ser enviados en código binario.

El protocolo mediante el cual se envía la información desde el PC a la FPGA es el protocolo de comunicación serie RS-232. Éste determina que la información enviada va precedida por un bit de START o de inicio de envío que es identificado al tomar la señal de entrada el valor '0'.A continuación son enviados los 8 bits que contienen la información relevante. A continuación se envía un bit de paridad que puede ser par o impar con el fin de comprobar que el número de bits en alto '1' en la información enviada anteriormente es correcto. Si no lo es, la información recibida es desechada. Por último, se envía uno o varios bits de STOP de valor '1' con el cuál finaliza el proceso de envío.

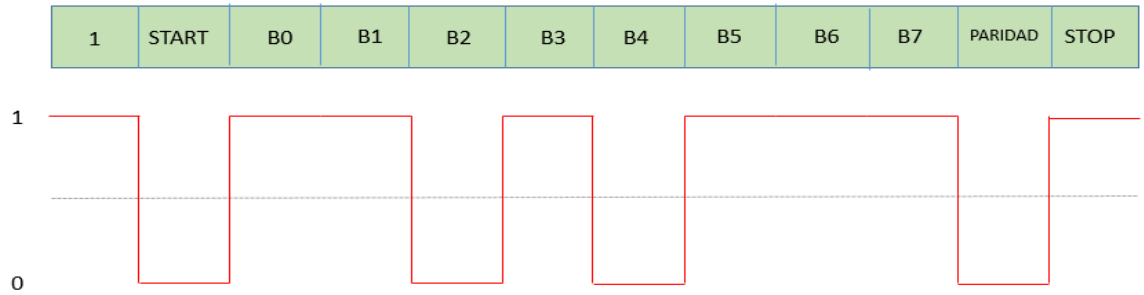

Figura 34. Trama de bits enviada mediante el protocolo RS-232.

La velocidad de transmisión de los bits desde el PC a la FPGA se ha establecido en 9600 baudios (bits/s). En estado de no envío de información, el valor envíado permanece en '1'. En el protocolo implementado se ha escogido paridad par y un único bit de STOP.

Los parámetros a configurar y los 4 bits que permiten identificarlos son:

1. Valor medio del ciclo de trabajo: 0110
2. Perturbación cuadrada o sinusoidal con 8 ciclos de reloj: 1100
3. Factor multiplicador de la amplitud de la perturbación sinusoidal:1000
4. Frecuencia de la variación sinusoidal del ciclo de trabajo: 1010
5. Incremento en el ciclo de trabajo debido a la perturbación pseudoaleatoria : 1110
6. Tamaño del registro que genera la secuencia pseudoaleatoria: 0111

Por otra parte, los bits que definen la orden de inicio y parada de la modulación son las siguientes:

- Inicio de la modulación: 0010
- Parada: 0100

A continuación se realiza una explicación más detallada de cada uno de estos parámetros y como su valor es codificado a código binario para ser enviado a la FPGA.

- **VALOR MEDIO DEL CICLO DE TRABAJO:** Es el valor medio alrededor del cuál el ciclo de trabajo fluctúa en cada periodo en presencia de perturbaciones y es enviado como número de ciclos de reloj tras los cuáles la señal de modulación cambia de estar en alto a estar en bajo en cada periodo. De este modo, el valor en binario enviado, teniendo en cuenta que la frecuencia de conmutación (100 Khz) corresponde a 500 ciclos de reloj de la FPGA al ser su frecuencia interna 50MHz es el valor del ciclo de trabajo entre 0 y 1 multiplicado por 500. Se requieren dos bytes para enviar la información numérica correspondiente a este parámetro.

- **PERTURBACIÓN SINUSOIDAL O CUADRADA**: Con este parámetro se determina si la perturbación introducida es cuadrada o sinusoidal definida por 1 y 8 ciclos de reloj en amplitud respectivamente y codificada en binario por esos mismos valores, los cuáles se envían en el segundo byte.

- **FACTOR MULTIPLICADOR DE LA PERTURBACIÓN CUADRADA/ SINUSOIDAL**: Por otra parte, se envía un factor de escala o multiplicador mediante el cual es posible aumentar la amplitud de la perturbación sinusoidal o cuadrada. En la interfaz es posible visualizar ésta amplitud total, resultante de la multiplicación de este factor por el número mínimo de ciclos de reloj que definen la perturbación. Éste valor de escala viene codificado en sólo un byte al no ser deseable un factor multiplicador muy grande siendo que se pretende obtener un modelo en pequeña señal.

- **FRECUENCIA DE LA PERTURBACIÓN CUADRADA/SINUSOIDAL:** Para configurar este parámetro se envía un valor binario que indica el número de ciclos de reloj menos uno tras los cuáles se modifica el valor de la perturbación. Ésta toma 50 valores en cada mitad de un semiperiodo .De este modo, la relación entre la frecuencia de la perturbación y el valor numérico enviado viene

dado por la relación:

$$f_{per} = \frac{1}{(50 \cdot t_{clk} \cdot 4 \cdot (1 + Valor\ enviado))} \quad (24)$$

La información del parámetro viene codificada en dos bytes ya que para frecuencias ligeramente inferiores a las de corte el valor enviado supera la capacidad de un solo byte.

- **MAGNITUD DE LA PERTURBACIÓN PSEUDOALEATORIA:** Este parámetro corresponde al valor del incremento del Duty debido a la perturbación PRBS, el cual viene expresado en binario en número de ciclos de reloj de incremento o decremento en cada periodo de conmutación y su valor se configura solo con el segundo byte al trabajar en pequeña señal.

- **TAMAÑO DEL REGISTRO GENERADOR DE LA SECUENCIA PRBS:** Con este parámetro es posible escoger el tamaño del registro que genera la secuencia pseudoaletoria entre 9,10 y 11 bits. Para leer el valor que configura el tamaño del registro, el cuál es ese mismo codificado en binario, de nuevo el programa en la FPGA sólo necesita recoger los datos del segundo byte.

4.3 IMPLEMENTACIÓN DIGITAL EN FPGA.

4.3.1 MODULADOR DIGITAL

A continuación se describe la arquitectura desarrollada mediante VHDL tanto para la recepción de la información transmitida mediante comunicación serie con el PC como para generar la señal de modulación. Los bloques mostrados representan procesos que operan de forma simultánea desde que se recibe el primer bit hasta que comienza la modulación.

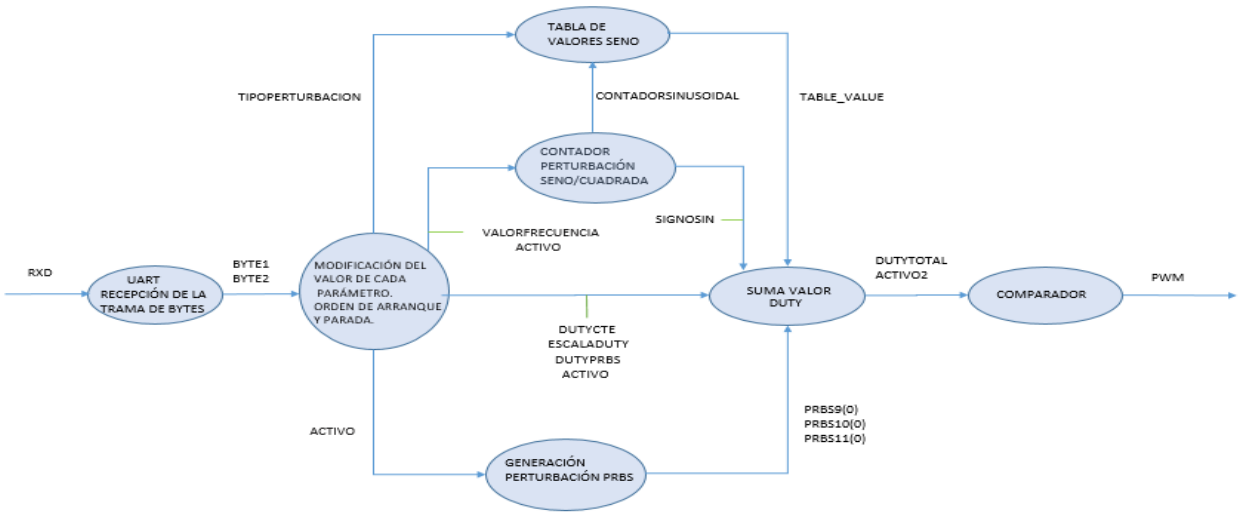

Figura 35. Esquema de la arquitectura VHDL del modulador.

- **LECTURA DE LOS DATOS**

En primer lugar el proceso destinado a la lectura de los bytes o byte enviado desde el PC (en el caso de la orden de START o STOP solo se necesita uno), comprueba que la paridad es correcta y que el parámetro enviado es uno de los 6 posibles .A continuación almacena los valores de los bytes en los registros BYTE1 y BYTE2.

- **ALMACENAMIENTO EN REGISTROS**

Tras la modificación de los registros BYTE1 y BYTE2 se activa el proceso que almacena el valor de cada uno de los parámetros uno de los 6 registros correspondientes:

NOMBRE DEL REGISTRO	PARÁMETRO QUE CONFIGURA
1. DUTYCTE	Valor medio del ciclo de trabajo.
2. TIPOPERTURBACION	Perturbación sinusoidal o cuadrada.
3. ESCALADUTY	Factor multiplicador de la perturbación.
4. VALORFRECUENCIA	Frecuencia de la variación cuadrada/senoidal.
5. DUTYPRBS	Magnitud de la variación pseudoaleatoria.
6. TAMANOREGISTRO	Tamaño del registro generador de PRBS.

El último byte enviado, correspondiente a la orden de START pone a '1' la señal ACTIVO que permite el comienzo de la generación de las perturbaciones senoidal/cuadrada y PRBS. En el caso de enviarse la orden de STOP esta señal se pone a '0', los valores de las perturbaciones son reseteados y la salida PWM se pone asimismo a '0'.

- **GENERADOR DE LA PERTURBACIÓN SENOIDAL/CUADRADA**

La variación senoidal en el ciclo de trabajo es generada mediante una tabla de 50 valores resultantes de muestrear la mitad de un semiperiodo de una onda senoidal perfecta de amplitud 8 y redondear al entero más próximo. El incremento instantáneo en el ciclo de trabajo o valor de la tabla (TABLE_VALUE) viene determinado por el valor en ese momento de un contador dependiente de la señal VALORFRECUENCIA, que le marca cada cuantos ciclos de reloj el contador debe incrementar en uno su valor en función de la frecuencia escogida para la perturbación. Una vez que el contador llega a 50, éste comienza a descender para que la tabla de valores genere la segunda mitad de cada semiperiodo de la perturbación. En el caso de la onda cuadrada, la tabla de valores recoge un único valor constante correspondiente a 1 ciclo de reloj de incremento en el ciclo de trabajo.

A su vez, una vez que el contador llega a 0 éste comienza a ascender de nuevo pero ahora la señal SIGNOSIN ve alterado su valor para quede registrado el signo negativo de estos valores

correspondientes al segundo semiperiodo de la onda.

- **GENERADOR DE LA SECUENCIA PSEUDOALEATORIA**

Al mismo tiempo, el contador en el proceso que genera la secuencia PRBS se activa para, tras 500 ciclos de reloj, permitir la retroalimentación del último bit del registro (el de salida) y de los bits definidos por los polinomios primitivos en registros de tres tamaños distintos PRBS9, PRBS11 y PRBS12 cada 500 ciclos de reloj con el fin de variar el valor del bit de salida en cada periodo de conmutación y así generar la perturbación pseudoaleatoria .

- **SUMATORIO DE LOS VALORES DEL CICLO DE TRABAJO**

El valor final del ciclo de trabajo (DUTYTOTAL) es obtenido como resultado de sumar al valor medio de éste (DUTYCTE) los incrementos correspondientes a la perturbación cuadrada/senoidal y pseudoaleatoria. Éste se obtiene mediante la relación:

$$Dutytotal = Dutycte \mp Escaladuty * Tablevalue \mp DutyPrbs$$

Donde el signo correspondiente al término $Escaladuty * Tablevalue$ viene determinado por el valor de la señal SIGNOSIN y el signo del término $DutyPrbs$, depende del valor en cada periodo de conmutación del bit de salida del registro utilizado para generar la secuencia, el cuál puede ser PRBS9(0), PRBS11(0) o PRBS12(0) dependiendo del valor de la señal TAMANOREGISTRO.

De este modo se ha establecido que para la elección correcta de un modo de funcionamiento que genere un solo tipo de perturbación y así poder aplicar los algoritmos de identificación correspondientes a uno de los dos métodos estudiados el valor de una las señales configuradas por la interfaz gráfica ESCALADUTY y DUTYPRBS debe elegirse como nula en función del método que se esté utilizando.

- **COMPARADOR Y GENERACIÓN DE LA SEÑAL DE MODULACIÓN**

Por último, se establece un comparador con el que una vez que un contador que se incrementa en uno en cada ciclo de reloj y cuenta de 0 a 499 supere el valor determinado por la señal DUTYTOTAL la salida PWM toma valor ‘0’ y caso contrario permanece en ‘1’.

4.3.2 BLOQUE DE COMUNICACIÓN SERIE

Para la recepción de los bits de entrada se ha generado un primer proceso formado por una máquina de estados siguiendo la secuencia correspondiente al protocolo RS-232.

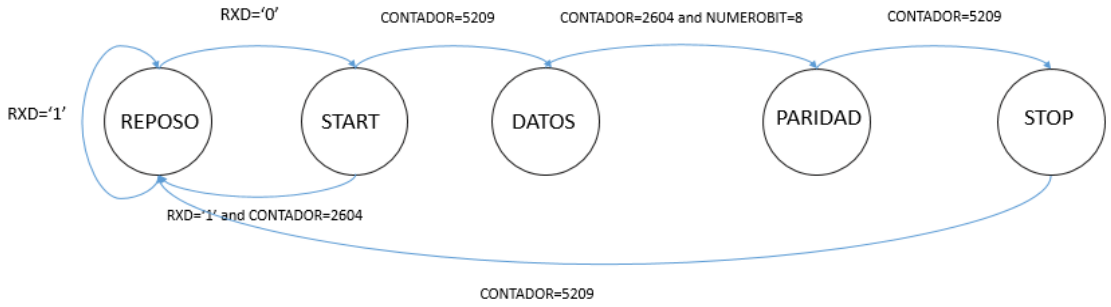

Figura 36. Máquina de estados diseñada para la recepción de los bits.

Partiendo del estado REPOSO, la máquina pasa a START cuando se recibe un '0' en la entrada RXD. En el estado START se lee a mitad del bit enviado su valor para comprobar que es '0': si la entrada ha tomado valor '1' se vuelve al estado REPOSO. En el caso de que el valor de la entrada continúe en '0', una vez que el contador de ciclos de reloj llega a 5209, correspondientes a una frecuencia de 9598,77 baudios se pasa al estado DATOS , en el que se leen los 8 bits correspondientes al byte enviado para posteriormente leer el bit de paridad en el estado PARIDAD y por último en el estado STOP almacenar en los registros BYTE1 y /o BYTE2 la información recibida si la información enviada es correcta y es completa.

La lectura de los valores del byte recibido, la cuál tiene lugar en el estado DATOS se muestra en el siguiente diagrama:

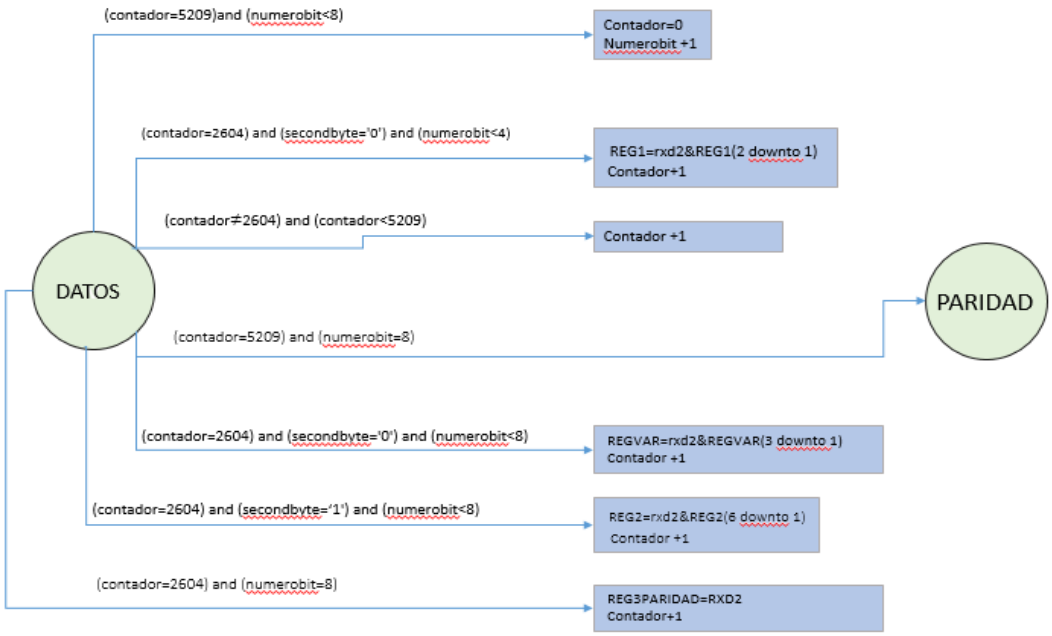

Figura 37. Descripción del estado DATOS.

Donde se muestra que la lectura de los bits tiene lugar a mitad de byte y estos son almacenados en los registros correspondientes al valor de los parámetros (REG1 y REG2) y la clase de parámetro (REGVAR) según el byte a leer sea el primero o el segundo de la trama enviada.

Una vez que se ha completado el tiempo de lectura del byte se pasa al estado de PARIDAD, el cuál se muestra más en detalle a continuación:

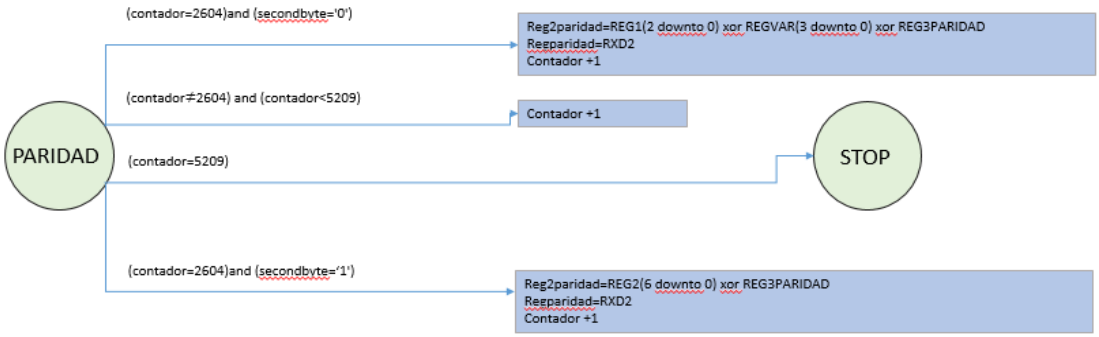

Figura 38. Descripción del estado PARIDAD.

En él se lee a mitad de bit el valor correspondiente a la paridad, la cuál se ha configurado como par. A su vez se guarda en un registro la paridad correspondiente a los valores leídos en el byte en el estado DATOS operando mediante puertas XOR.

Por último, se pasa al estado STOP:

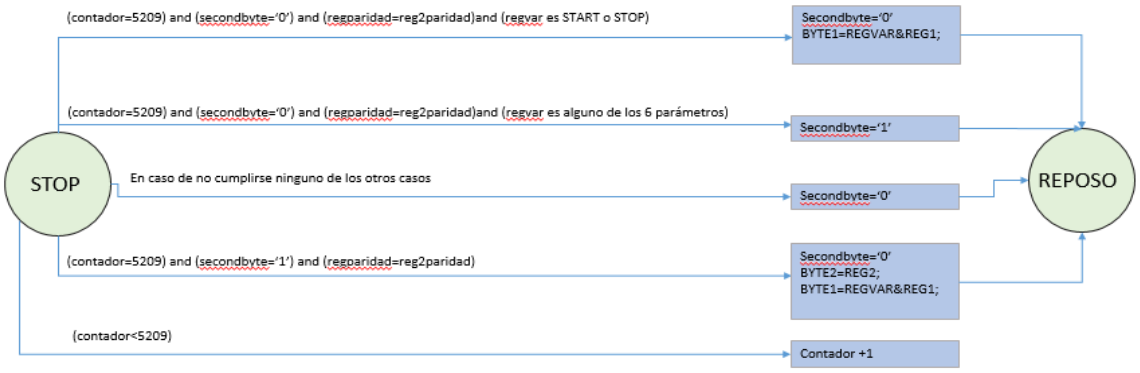

Figura 39. Descripción del estado STOP.

En éste se almacena la orden de START o STOP en el registro BYTE1 y en el caso de recibirse un primer byte correspondiente a la cofiguración de un parámetro la señal SECONDBYTE toma el valor '1' para una vez recibido el segundo byte correspondiente al valor del parámetro, modificarse los registros BYTE1 y BYTE2.En cualquier caso esto solamente es posible si la paridad es correcta. Una vez finalizado el tiempo correspondiente al bit de stop se vuelve de nuevo al

estado REPOSO.

4.4 RESULTADOS DE LA SIMULACIÓN

Se ha simulado en un entorno TestBench utilizando el programa ISE Design Suite 14.7 la transmisión de datos a la FPGA y la generación de las señales internas de la arquitectura involucradas en el proceso de generar la señal externa de modulación.

Para ello se ha simulado la transmisión de la información de los 6 parámetros en tramas de 2 bytes y de la orden de START a la frecuencia de comunicación de 9598,77 baudios correspondiente a 5209 ciclos de reloj por cada valor de bit asignado en la señal de entrada de lectura RXD.

Los valores de la entrada RXD así como los diferentes estados que toma el proceso de recepción de bits y las señales relacionadas durante la transmisión de los primeros 4 bytes se muestran en la siguiente imagen:

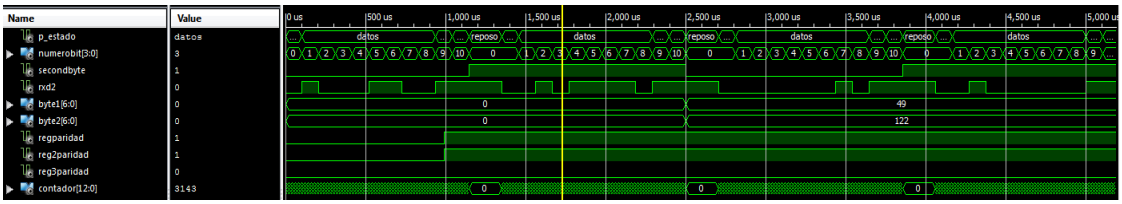

Figura 40. Recepción de los 4 bytes correspondientes a los dos primeros parámetros enviados.

Al acabar la lectura de los dos primeros bytes los registros BYTE1 y BYTE2 toman los valores enviados si la comunicación se ha realizado correctamente. La señal secondbyte se pone a '1' si se detecta que se ha enviado el primer byte correspondiente cualquiera de los 6 parámetros.

Para cada byte enviado la máquina de estados toma todos los valores posibles hasta llegar al estado REPOSO.

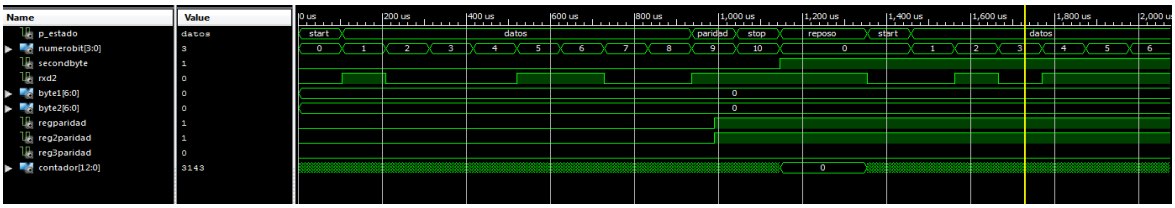

Figura 41. Simulación de la máquina de estados.

Para verificar la correcta generación de la perturbación senoidal en el ciclo de trabajo y de la señal cuadrada sincronizada con ésta empleada para poder aplicar la DFT en la entrada se ha simulado la transmisión de la información correspondiente a distintas frecuencias y factores de escala:

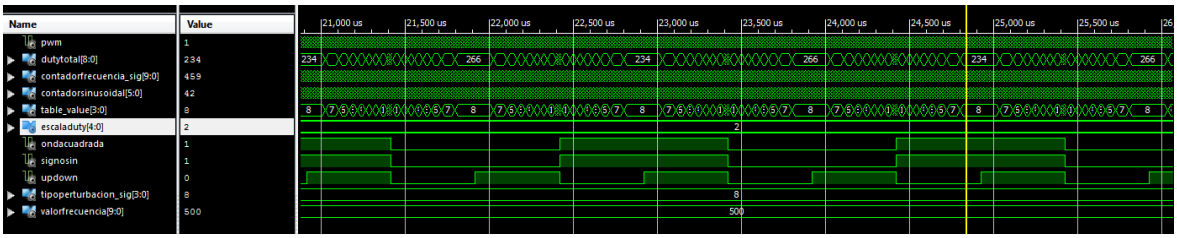

Figura 42. Simulación de dos periodos de la perturbación senoidal de frecuencia 500Hz en el ciclo de trabajo.

Los valores que toma el ciclo de trabajo así como los valores de la tabla que son utilizados para generar la perturbación y el contador que hace que estos se alternen se muestran en detalle en la siguiente imagen**:**

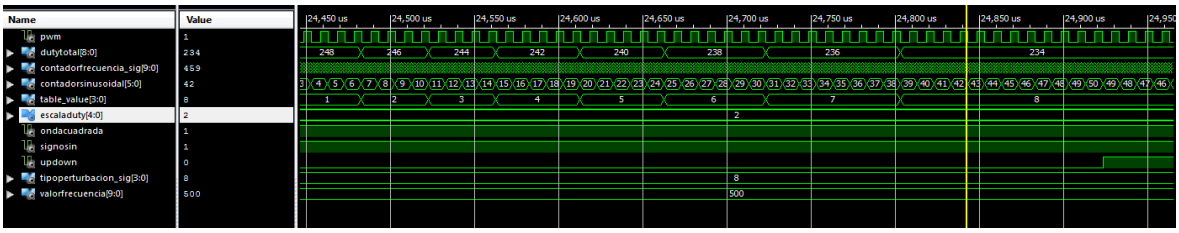

Figura 43. Simulación de un cuarto de periodo de la perturbación senoidal.

Se ha comprobado asimismo la generación de la perturbación pseudoaleatoria los tres tamaños de registro empleados:

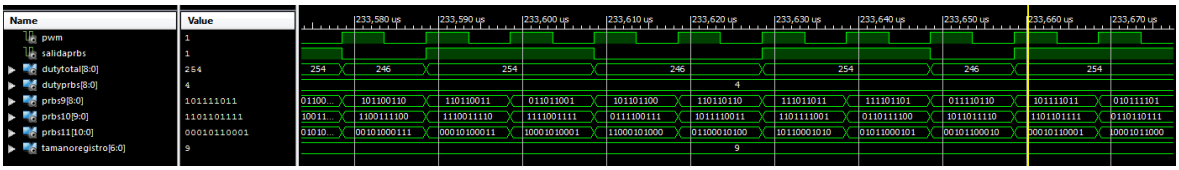

Figura 44. Simulación de la perturbación pseudoaleatoria en el ciclo de trabajo.

Un periodo completo de la secuencia PRBS generada con un registro de 9 bits se muestra en la siguiente imagen**:**

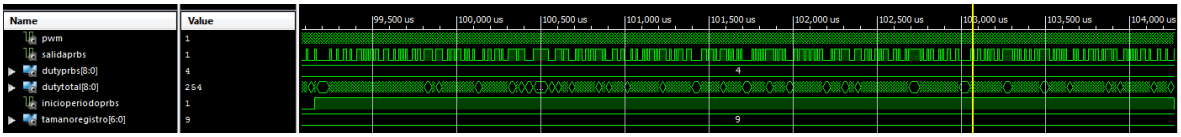

Figura 45. Periodo completo de la secuencia pseudoaleatoria.

Por último se ha simulado la orden de STOP para verificar la desactivación de las señales correspondientes a la perturbación y a la salida PWM.

CAPÍTULO 5.

VERIFICACIÓN EXPERIMENTAL.

Para la implementación de la modulación mediante la tarjeta SPARTAN 3E 1200 se ha utilizado un convertidor Buck con valores idénticos en los componentes que lo integran que los escogidos al realizar las simulaciones mediante software. La FPGA [13], se ha conectado mediante el puerto DB9 a un puerto USB configurado como puerto serie en el PC desde el que se han enviado los datos mediante el protocolo RS-232. La FPGA se ha acoplado al Buck empleando tres salidas diferentes G1, G2 y G3 para configurar los transistores del convertidor escogido.

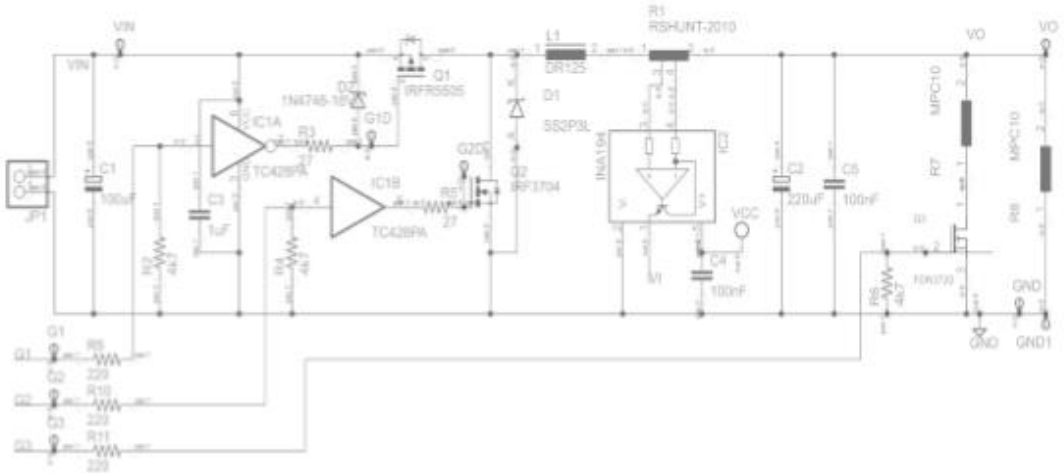

Figura 46. Esquema electrónico de la etapa Buck empleada para las pruebas experimentales[13].

La primera salida, G1, es la señal PWM que activa el transistor MOSFET cuando ésta se encuentra en alto regulando el ciclo de trabajo del convertidor. La salida G2 por su parte, se ha configurado con un valor constante de '0' para que el segundo transistor en paralelo con el diodo permanezca en corte y permita el funcionamiento del primero. Por último, la salida G3 se ha configurado con un valor constante de '1' para activar la resistencia R7 y así obtener una resistencia total de 2.5 ohmios equivalente a la utilizada en la simulación.

Para la obtención de los valores de la perturbación y digitalización de los mismos mediante un osciloscopio se ha generado en el caso del método de la perturbación senoidal una onda cuadrada desfasada 180º con la perturbación que se ha configurado mediante un pin de salida adicional. Por su parte la salida correspondiente perturbación PRBS se ha configurado mediante otro pin de salida distinto.

Por último se ha generado una salida cuyo valor conmutaba en cada ciclo de conmutación al alcanzar 400 ciclos de reloj con el fin de eliminar cualquier error provocado por la desviación el valor teórico del tiempo de oscilación del reloj de la FPGA. Los flancos tanto en alto como en bajo de dicha señal han sido la referencia para seleccionar los valores de entre los muestreados en los

otros dos canales a los que se les ha aplicado los algoritmos de la DFT y la correlación cruzada a lo largo de un periodo de la perturbación.

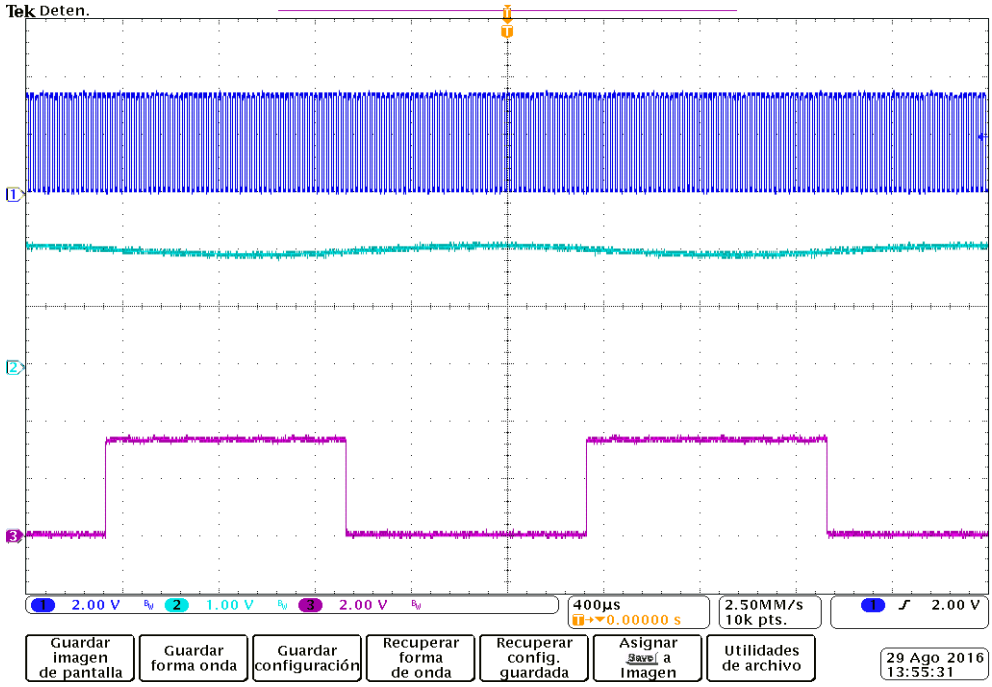

Figura 47. Representación de las tres señales con una perturbación senoidal.

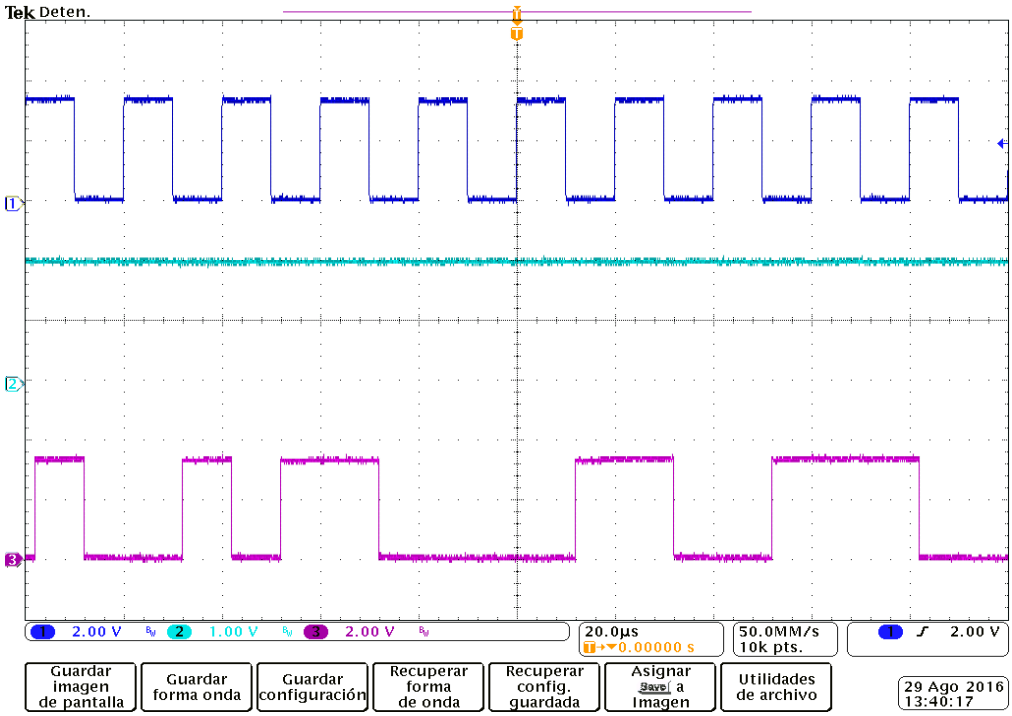

Figura 48. Representación de las tres señales con una perturbación pseudoaleatoria.

Para la obtención del modelo mediante perturbaciones senoidales se han empleado 8 frecuencias distintas submúltiplos del periodo de conmutación con el fin de obtener el diagrama de Bode del sistema en torno a la frecuencia de corte del mismo. Debido al desfase de π radianes de la entrada se ha tomado el conjugado del valor calculado. Los resultados obtenidos para el módulo y la fase se muestran a continuación.

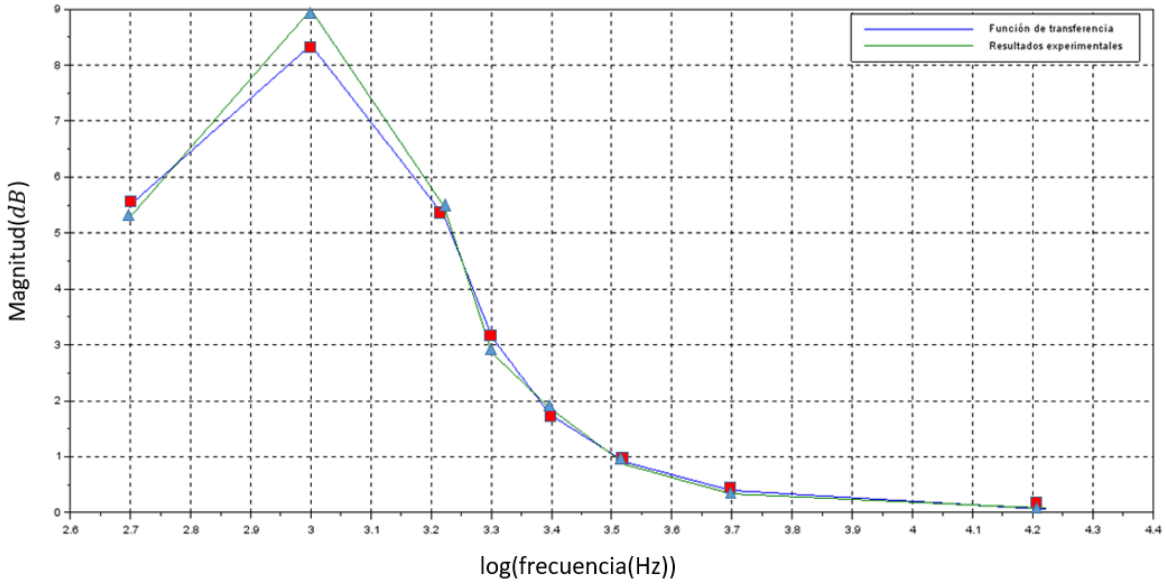

Figura 49. Resultados experimentales para el módulo con el método de la DFT.

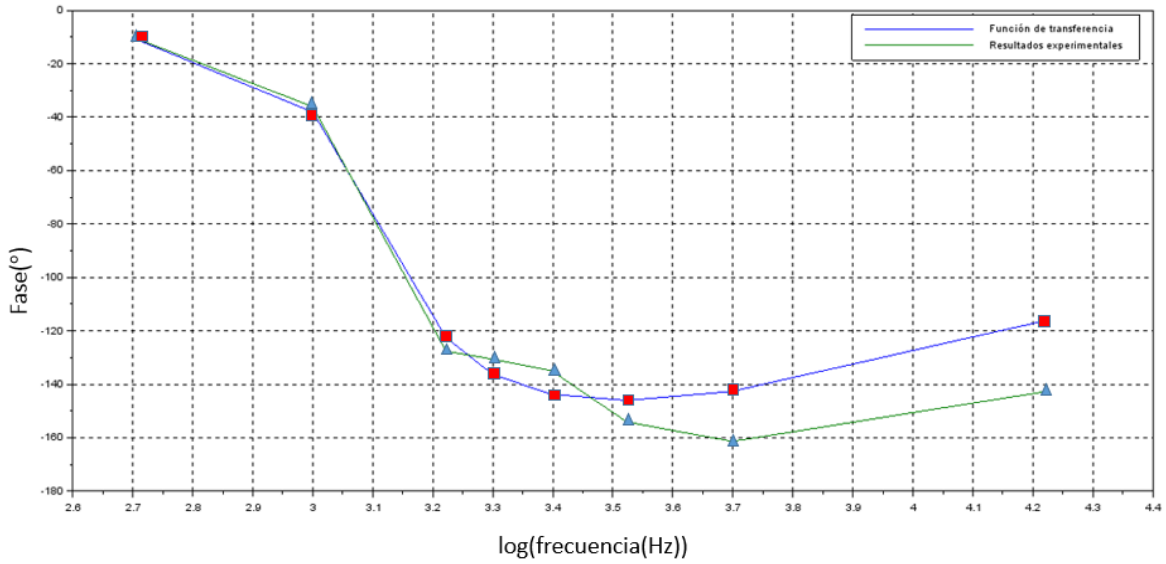

Figura 50. Resultados experimentales para la fase con el método de la DFT.

Los errores cometidos en la caracterización para las frecuencias evaluadas se muestran en las siguientes imágenes:

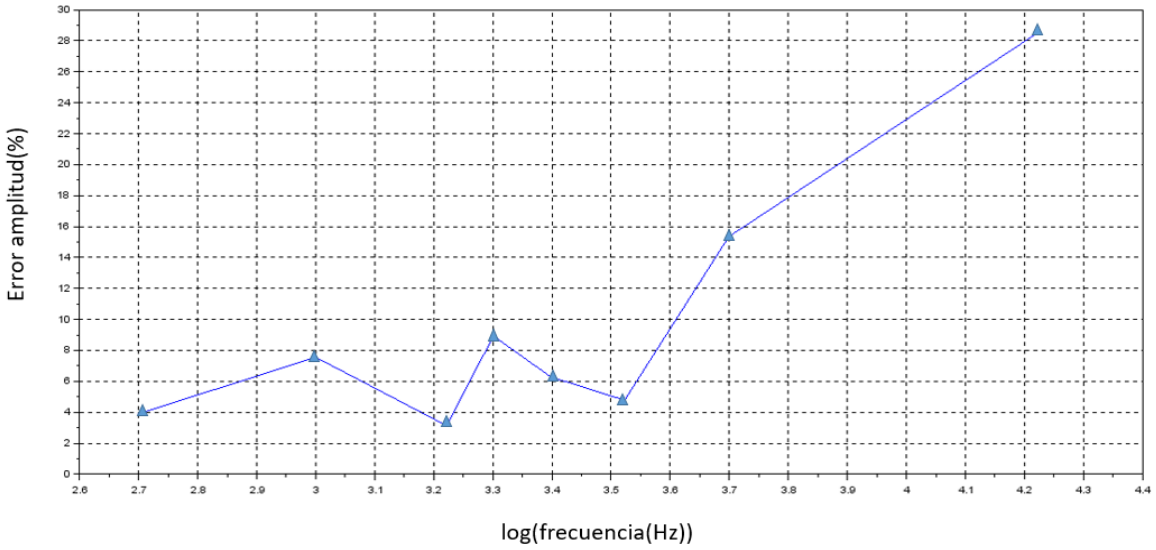

Figura 51. Error experimental en la caracterización del módulo con el método de la DFT.

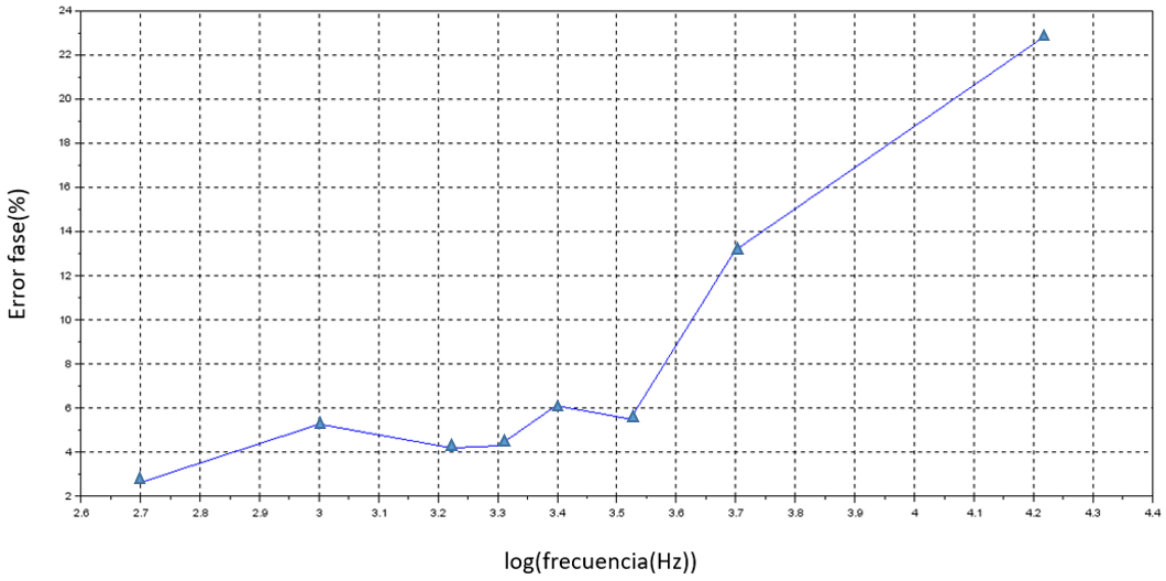

Figura 52. Error experimental en la caracterización de la fase con el método de la DFT.

Por su parte, mediante el método de la correlación cruzada para 9,10 y 11 bits con un incremento

del 0.8% en el ciclo de trabajo se han obtenido los siguientes resultados para el módulo :

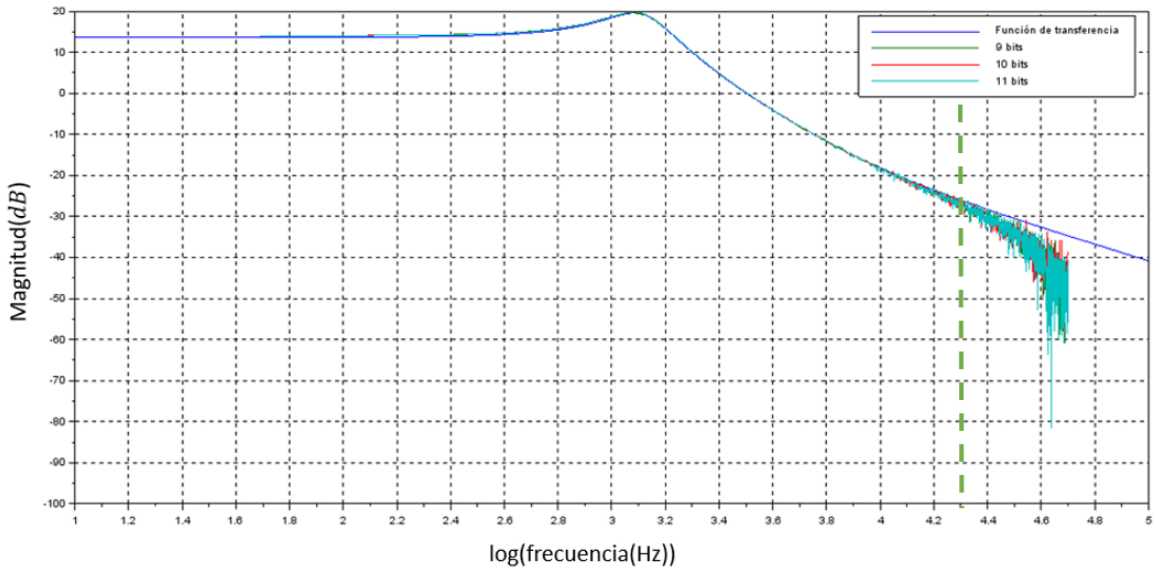

Figura 53. Resultados experimentales para el módulo con el método de la correlación cruzada.

Y para la fase, respectivamente:

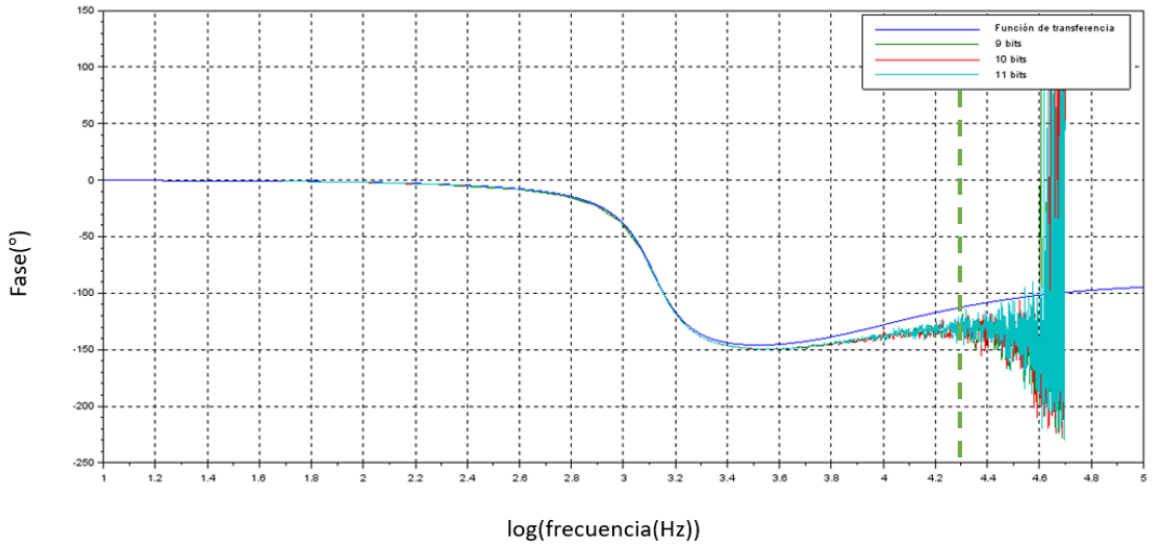

Figura 54. Resultados experimentales para la fase con el método de la correlación cruzada.

Con los siguientes errores respecto a la función de transferencia teórica:

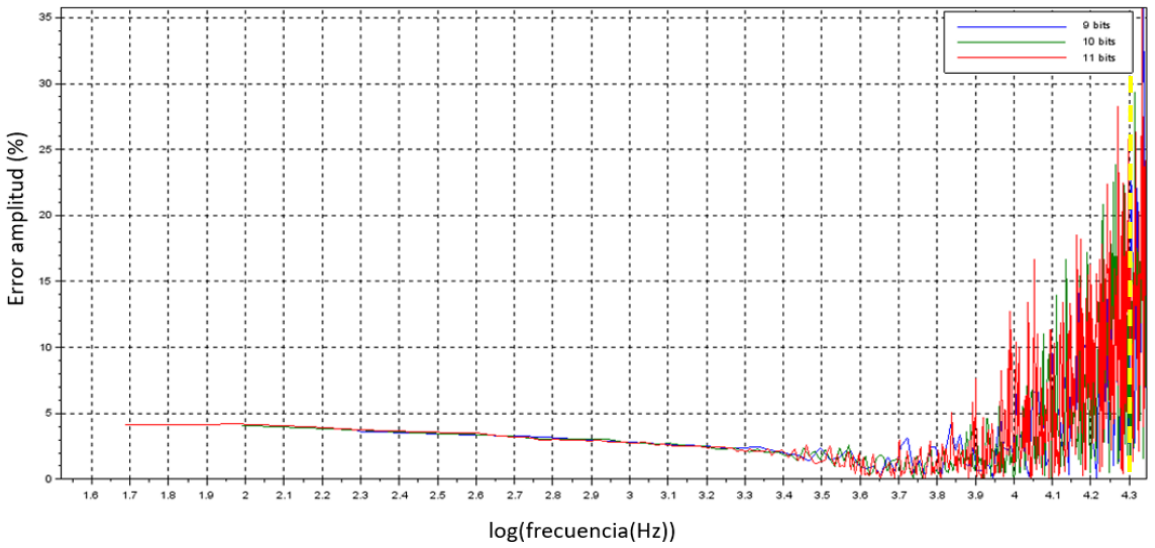

Figura 55. Error experimental en la caracterización del módulo con el método de la correlación cruzada.

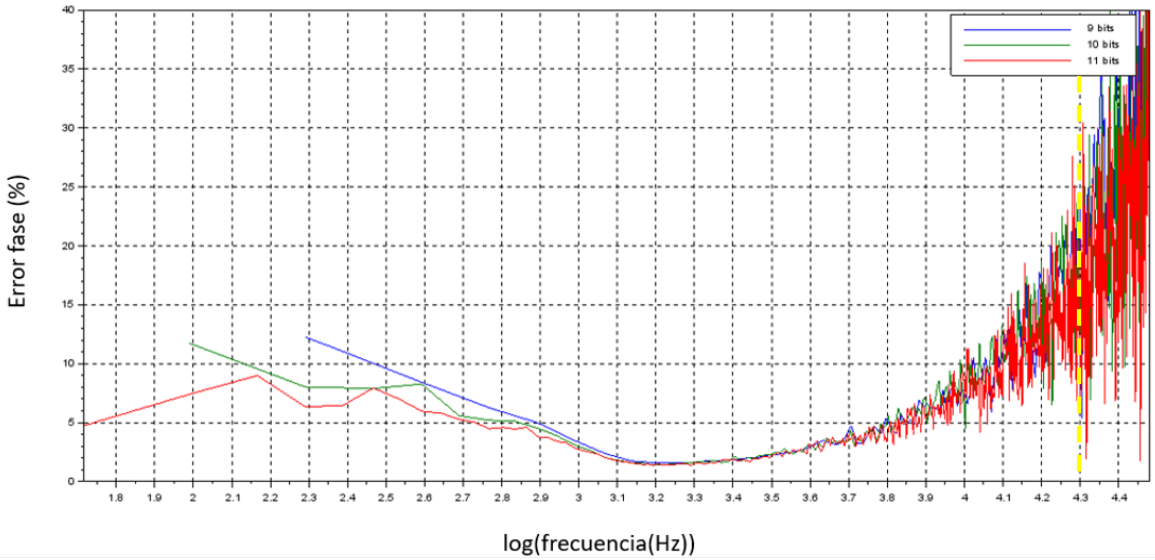

Figura 56. Error experimental en la caracterización de la fase con el método de la correlación cruzada.

CAPÍTULO 6.

CONCLUSIONES Y TRABAJOS FUTUROS.

6.1. CONCLUSIONES

Tras la simulación y posterior comprobación experimental de los dos métodos se ha determinado que ambos métodos presentan una gran aproximación con la ventaja de que en el caso de la generación de la secuencia PRBS solamente es necesario procesar los resultados obtenidos una vez para obtener un modelo del sistema en un rango elevado de frecuencias, siendo asimismo la complejidad a la hora de ser implementado de forma digital menor que con el método de la generación de la perturbación senoidal .

6.2. LÍNEAS DE TRABAJO FUTURAS

Con respecto a futuros trabajos relacionados con el realizado se incluye la aplicación de métodos alternativos de identificación como el método DPL (Dual Phase Lock-in Algorithm) [4] .

Otra línea de trabajo alternativa sería la implementación de un controlador del sistema en tiempo real realizando la DFT a la frecuencia de cruce deseada y ajustando automáticamente las constantes del controlador en función de la magnitud y fase obtenidas (autotuning).

7. BIBLIOGRAFÍA

[1] L. Ljung. *"System Identification. Theory for the user*". Prentice Hall. 1987.

[2] T. Söderström y P. Stoica. *"System Identification".* Prentice Hall. 1989.

[3] B. Miao, R. Zane, and D. Maksimovic, "*System identification of power converters with digital control through cross-correlation methods*", IEEE Transactions on Power Electronics, vol. 20, no. 5, pp. 1093-1099, 2005.

[4] L.A. Barragán, J.I. Artigas, O. Lucía, D. Navarro, I. Urriza, and O. Jiménez, *"Frequency response measurement of DC-DC converters using a lock-in algorithm*", in IEEE Applied Power Electronics Conference and Exposition (APEC), pp. 1218-1223, 2012.

[5] N. Mohan, T. Undeland, and W. Robbins, "*Power Electronics: Converters, Applications and Design".* John Wiley & Sons. 1989.

[6] R. W. Erickson, "*Fundamentals of Power Electronics"*, 2 ed., Colorado: Kluwer academic publishers. 2001.

[7] C. Medrano, JM Valiente, I.Plaza y P.Ramos, "*Evaluación de herramientas de software libre para cálculo numérico*".Congreso TAEE Tecnologías Aplicadas a la Enseñanza de la Electrónica. Madrid, 2008.

[8] www.scicos.org

[9] https://wiki.scilab.org

[10] Torrey D.A., Selamogullari U.S.:"*A Behavioral Model for DC-DC Converter using Modelica* ". 2nd International Modelica Conference, Proceedings,pp.167-172.

[11] M. Hagen and V. Yousefzadeh, "*Applying digital technology to PWM control-loop designs*," . Power Supply Design Seminar, 2009.

[12] I.Buioli , J. Pérez." *Processing. Un lenguaje al alcance de todos"*, 2013.

[13] J. Iguaz Navarro. "*Control en FPGA de un convertidor reductor en modo corriente de pico* ". Trabajo Fin de Máster en Ingenieria Electrónica. Universidad de Zaragoza, 2013.

ANEXO A: MÓDULO VHDL

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:        15:25:17 06/06/2016
-- Design Name:
-- Module Name:        Modulador- Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;


entity Modulador is
    Port ( clk : in    STD_LOGIC;
           rst : in    STD_LOGIC;
           rxd : in    STD_LOGIC;
           pwm: out    STD_LOGIC;
                       salidatransistor2:out STD_LOGIC;
                       salidatransistor3:out STD_LOGIC;
                       ondacuadrada:out STD_LOGIC;
                       salidaPRBS:out STD_LOGIC;
                       salidadebug:out STD_LOGIC;
                       referpwm:out STD_LOGIC;
                       inicioperiodoPRBS:out STD_LOGIC
                       );
end Modulador ;

architecture Behavioral of Modulador is


signal numerobit,numerobit_sig:unsigned(3 downto 0);
type estado is (REPOSO,START,DATOS,STOP,PARIDAD);
```

```
signal p_estado,n_estado: estado;
signal REGVAR,REGVAR_sig:std_logic_vector(3 downto 0);
signal REG1,REG1_sig: std_logic_vector(2 downto 0);
signal REG2,REG2_sig,BYTE1,BYTE1_sig,BYTE2,BYTE2_sig: std_logic_vector(6 downto 0);
signal secondbyte,secondbyte_sig:std_logic;
signal contador,contador_sig:unsigned(12 downto 0);
signal
regparidad_sig,regparidad,reg2paridad_sig,reg2paridad,reg3paridad,reg3paridad_sig:std_logic;
signal table_value: unsigned(3 downto 0);
signal contadorsinusoidal_sig,contadorsinusoidal: unsigned(5 downto 0);
signal updown,updown_sig :std_logic;
signal contadorfrecuencia,contadorfrecuencia_sig : unsigned(9 downto 0);
signal valorfrecuencia,valorfrecuencia_sig:unsigned(9 downto 0);
signal dutycte,dutycte_sig: unsigned(8 downto 0);
signal dutytotal,dutytotal_sig:unsigned(8 downto 0);
signal dutyprbs,dutyprbs_sig:unsigned(8 downto 0);
signal escaladuty,escaladuty_sig:unsigned(4 downto 0);
signal tipoperturbacion,tipoperturbacion_sig:unsigned(3 downto 0);
signal contadorpwm,contadorpwm_sig:unsigned(8 downto 0);
signal señalpwm,señalpwm_sig:std_logic;
signal signosin,signosin_sig:std_logic;
signal PRBS9,PRBS9_sig:std_logic_vector(8 downto 0);
signal PRBS10,PRBS10_sig:std_logic_vector(9 downto 0);
signal PRBS11,PRBS11_sig:std_logic_vector(10 downto 0);
signal tamanoregistro,tamanoregistro_sig:unsigned(6 downto 0);
signal contadorprbs,contadorprbs_sig:unsigned(8 downto 0);
signal activo,activo_sig,activo2,activo2_sig:std_logic;
signal inicsecPRBS,inicsecPRBS_sig:std_logic;
signal rxd1,rxd2:std_logic;
signal refpwm,refpwm_sig:std_logic;


begin
salidadebug<=rxd2;
pwm<=señalpwm;
ondacuadrada<=signosin;
salidaPRBS<=PRBS9(0)    when tamanoregistro=9 else
                        PRBS10(0) when tamanoregistro=10 else
                        PRBS11(0);
inicioperiodoPRBS<=inicsecPRBS;
salidatransistor2<='0';
salidatransistor3<='1';
referpwm<=refpwm;



process(CLK,RST)
```

```
begin
   if RST='1' then
     p_estado<=REPOSO;
          REGVAR<="0000";
          REG1<="000";
          REG2<="0000000";
          BYTE1<=(OTHERS=>'0');
          BYTE2<=(OTHERS=>'0');
          numerobit<="0000";
          contador<=(others=>'0');
          secondbyte<='0';
          regparidad<='0';
          reg2paridad<='0';
          reg3paridad<='0';
          contadorsinusoidal<=(others=>'0');
         updown<='0';
         contadorfrecuencia<=(others=>'0');
         valorfrecuencia<=(others=>'0');
         dutycte<=(others=>'0');
         dutytotal<=(others=>'0');
         dutyprbs<=(others=>'0');
         escaladuty<=(others=>'0');
         tipoperturbacion<=(others=>'0');
         contadorpwm<=(others=>'0');
         señalpwm<='0';
         signosin<='0';
         PRBS9<="000000001";
         PRBS10<="0000000001";
         PRBS11<="00000000001";
         tamanoregistro<="0001001";
         contadorprbs<=(others=>'0');
         activo<='0';
         activo2<='0';
         inicsecPRBS<='0';
         rxd2<='0';
         rxd1<='0';
         refpwm<='0';




   elsif (CLK'EVENT and CLK='1') then
     p_estado<=n_estado;
          REGVAR<=REGVAR_sig;
          REG1<=REG1_sig;
          REG2<=REG2_sig;
          BYTE1<=BYTE1_sig;
```

```
            BYTE2<=BYTE2_sig;
            numerobit<=numerobit_sig;
            contador<=contador_sig;
            secondbyte<=secondbyte_sig;
           regparidad<=regparidad_sig;
           reg2paridad<=reg2paridad_sig;
           reg3paridad<=reg3paridad_sig;
           contadorsinusoidal<=contadorsinusoidal_sig;
           updown<=updown_sig;
           contadorfrecuencia<=contadorfrecuencia_sig;
           valorfrecuencia<=valorfrecuencia_sig;
           dutycte<=dutycte_sig;
           dutytotal<=dutytotal_sig;
           dutyprbs<=dutyprbs_sig;
           escaladuty<=escaladuty_sig;
           tipoperturbacion<=tipoperturbacion_sig;
           contadorpwm<=contadorpwm_sig;
           señalpwm<=señalpwm_sig;
           signosin<=signosin_sig;
           PRBS9<=PRBS9_sig;
           PRBS10<=PRBS10_sig;
           PRBS11<=PRBS11_sig;
           tamanoregistro<=tamanoregistro_sig;
           contadorprbs<=contadorprbs_sig;
           activo<=activo_sig;
           activo2<=activo2_sig;
           inicsecPRBS<=inicsecPRBS_sig;
           rxd2<=rxd1;
           rxd1<=rxd;
           refpwm<=refpwm_sig;
    end if;


  end process;

process(BYTE1,BYTE2,ACTIVO,DUTYCTE,ESCALADUTY,TIPOPERTURBACION,VALORFRECUENCIA,DU
TYPRBS,activo2,TAMANOREGISTRO)
BEGIN

if BYTE1(6 downto 3)="0010" then
      ACTIVO_sig<='1';
      DUTYCTE_SIG<=DUTYCTE;
      ESCALADUTY_SIG<=ESCALADUTY;
      TIPOPERTURBACION_SIG<=TIPOPERTURBACION;
      VALORFRECUENCIA_SIG<=VALORFRECUENCIA;
      DUTYPRBS_SIG<=DUTYPRBS;
      TAMANOREGISTRO_SIG<=TAMANOREGISTRO;
elsif BYTE1(6 downto 3)="0100" then
```

```
        ACTIVO_sig<='0';
        DUTYCTE_SIG<=DUTYCTE;
        ESCALADUTY_SIG<=ESCALADUTY;
        TIPOPERTURBACION_SIG<=TIPOPERTURBACION;
        VALORFRECUENCIA_SIG<=VALORFRECUENCIA;
        DUTYPRBS_SIG<=DUTYPRBS;
        TAMANOREGISTRO_SIG<=TAMANOREGISTRO;
    elsif BYTE1(6 downto 3)= "0110" then
        ACTIVO_sig<=ACTIVO;
        DUTYCTE_sig<=UNSIGNED(BYTE1(1 downto 0)& BYTE2(6 downto 0));
        ESCALADUTY_SIG<=ESCALADUTY;
        TIPOPERTURBACION_SIG<=TIPOPERTURBACION;
        VALORFRECUENCIA_SIG<=VALORFRECUENCIA;
        DUTYPRBS_SIG<=DUTYPRBS;
        TAMANOREGISTRO_SIG<=TAMANOREGISTRO;
    elsif BYTE1(6 downto 3)="1000" then
        ACTIVO_SIG<=ACTIVO;
        DUTYCTE_SIG<=DUTYCTE;
        ESCALADUTY_sig<=UNSIGNED(BYTE2(4 downto 0));
        TIPOPERTURBACION_SIG<=TIPOPERTURBACION;
        VALORFRECUENCIA_SIG<=VALORFRECUENCIA;
        DUTYPRBS_SIG<=DUTYPRBS;
        TAMANOREGISTRO_SIG<=TAMANOREGISTRO;
    elsif BYTE1(6 downto 3)="1100" then
        ACTIVO_SIG<=ACTIVO;
        DUTYCTE_SIG<=DUTYCTE;
        ESCALADUTY_sig<=ESCALADUTY;
        TIPOPERTURBACION_SIG<=UNSIGNED(BYTE2(3 downto 0));
        VALORFRECUENCIA_SIG<=VALORFRECUENCIA;
        DUTYPRBS_SIG<=DUTYPRBS;
        TAMANOREGISTRO_SIG<=TAMANOREGISTRO;
    elsif BYTE1(6 downto 3)="1010" then
        ACTIVO_SIG<=ACTIVO;
        DUTYCTE_SIG<=DUTYCTE;
        ESCALADUTY_sig<=ESCALADUTY;
        TIPOPERTURBACION_SIG<=TIPOPERTURBACION;
        VALORFRECUENCIA_sig<=UNSIGNED(BYTE1(2 DOWNTO 0)& byte2(6 DOWNTO 0));
        DUTYPRBS_SIG<=DUTYPRBS;
        TAMANOREGISTRO_SIG<=TAMANOREGISTRO;
    elsif BYTE1(6 DOWNTO 3)="1110" THEN
        ACTIVO_SIG<=ACTIVO;
        DUTYCTE_SIG<=DUTYCTE;
        ESCALADUTY_sig<=ESCALADUTY;
        TIPOPERTURBACION_SIG<=TIPOPERTURBACION;
        VALORFRECUENCIA_sig<=VALORFRECUENCIA;
        DUTYPRBS_sig<=UNSIGNED(BYTE1(1 DOWNTO 0) & BYTE2(6 DOWNTO 0));
        TAMANOREGISTRO_SIG<=TAMANOREGISTRO;
    elsif BYTE1(6 DOWNTO 3)="0111" THEN
```

```
    ACTIVO_SIG<=ACTIVO;
    DUTYCTE_SIG<=DUTYCTE;
    ESCALADUTY_sig<=ESCALADUTY;
    TIPOPERTURBACION_SIG<=TIPOPERTURBACION;
    VALORFRECUENCIA_sig<=VALORFRECUENCIA;
    DUTYPRBS_sig<=DUTYPRBS;
    TAMANOREGISTRO_SIG<=UNSIGNED(BYTE2(6 DOWNTO 0));
elsif BYTE1(6 DOWNTO 3)="0001" THEN
    ACTIVO_SIG<=ACTIVO;
    DUTYCTE_SIG<=DUTYCTE;
    ESCALADUTY_sig<=ESCALADUTY;
    TIPOPERTURBACION_SIG<=TIPOPERTURBACION;
    VALORFRECUENCIA_sig<=VALORFRECUENCIA;
    DUTYPRBS_sig<=DUTYPRBS;
    TAMANOREGISTRO_SIG<=TAMANOREGISTRO;
else
    ACTIVO_SIG<=ACTIVO;
    DUTYCTE_SIG<=DUTYCTE;
    ESCALADUTY_sig<=ESCALADUTY;
    TIPOPERTURBACION_SIG<=TIPOPERTURBACION;
    VALORFRECUENCIA_sig<=VALORFRECUENCIA;
    DUTYPRBS_sig<=DUTYPRBS;
    TAMANOREGISTRO_SIG<=TAMANOREGISTRO;
END IF;
END PROCESS;
```

```
process(signosin,PRBS9,prbs10,prbs11,tamanoregistro,dutycte,dutyprbs,table_value,activo,dutyto
tal,activo2,escaladuty)
  begin
  IF activo='1' and signosin='0' and PRBS9(0)='1' and tamanoregistro=9 THEN
  dutytotal_sig<=(table_value*escaladuty)+dutycte+dutyprbs;
  activo2_sig<='1';
  ELSIF ACTIVO='1' AND signosin='1' and PRBS9(0)='1' and tamanoregistro=9 THEN
  DUTYTOTAL_sig<=dutycte-table_value*escaladuty+dutyprbs;
  activo2_sig<='1';
  ELSIF      ACTIVO='1' AND signosin='0' and PRBS9(0)='0' and tamanoregistro=9 THEN
  DUTYTOTAL_sig<=dutycte+table_value*escaladuty-dutyprbs;
  activo2_sig<='1';
  ELSIF activo='1' and signosin='1' and PRBS9(0)='0' and tamanoregistro=9 THEN
  DUTYTOTAL_sig<= dutycte-table_value*escaladuty-dutyprbs;
  activo2_sig<='1';
  ELSIF activo='1' and signosin='0' and PRBS10(0)='1' and tamanoregistro=10 THEN
```

```
 dutytotal_sig<=(table_value*escaladuty)+dutycte+dutyprbs;
 activo2_sig<='1';
 ELSIF ACTIVO='1' AND signosin='1' and PRBS10(0)='1' and tamanoregistro=10 THEN
 DUTYTOTAL_sig<=dutycte-table_value*escaladuty+dutyprbs;
 activo2_sig<='1';
 ELSIF      ACTIVO='1' AND signosin='0' and PRBS10(0)='0' and tamanoregistro=10 THEN
 DUTYTOTAL_sig<=dutycte+table_value*escaladuty-dutyprbs;
 activo2_sig<='1';
 ELSIF activo='1' and signosin='1' and PRBS10(0)='0' and tamanoregistro=10 THEN
 DUTYTOTAL_sig<= dutycte-table_value*escaladuty-dutyprbs;
 activo2_sig<='1';
 ELSIF activo='1' and signosin='0' and PRBS11(0)='1' and tamanoregistro=11 THEN
 dutytotal_sig<=(table_value*escaladuty)+dutycte+dutyprbs;
 activo2_sig<='1';
 ELSIF ACTIVO='1' AND signosin='1' and PRBS11(0)='1' and tamanoregistro=11    THEN
 DUTYTOTAL_sig<=dutycte-table_value*escaladuty+dutyprbs;
 activo2_sig<='1';
 ELSIF      ACTIVO='1' AND signosin='0' and PRBS11(0)='0' and tamanoregistro= 11 THEN
 DUTYTOTAL_sig<=dutycte+table_value*escaladuty-dutyprbs;
 activo2_sig<='1';
 ELSIF activo='1' and signosin='1' and PRBS11(0)='0' and tamanoregistro=11 THEN
 DUTYTOTAL_sig<= dutycte-table_value*escaladuty-dutyprbs;
 activo2_sig<='1';
 else
 dutytotal_sig<=dutytotal;
 activo2_sig<='0';
 END IF;
 end process;


process(contadorpwm,dutytotal,ACTIVO,activo2,señalpwm,refpwm)
BEGIN
if contadorpwm=499 AND activo='1' AND ACTIVO2='1'    then
     señalpwm_sig<='1';
     contadorpwm_sig<=(others=>'0');
     refpwm_sig<=refpwm;
elsif (contadorpwm<dutytotal) and (contadorpwm=400) and (ACTIVO='1') AND ACTIVO2='1' then
     señalpwm_sig<='1';
     contadorpwm_sig<=contadorpwm+1;
     refpwm_sig<=not(refpwm);
elsif (contadorpwm>=dutytotal) and (contadorpwm=400) and (ACTIVO='1') AND ACTIVO2='1' then
     señalpwm_sig<='0';
     contadorpwm_sig<=contadorpwm+1;
     refpwm_sig<=not(refpwm);
elsif (contadorpwm<dutytotal) and (ACTIVO='1') AND ACTIVO2='1' then
     señalpwm_sig<='1';
     contadorpwm_sig<=contadorpwm+1;
     refpwm_sig<=refpwm;
```

```
elsif (contadorpwm>=dutytotal) and (ACTIVO='1') AND ACTIVO2='1' then
    señalpwm_sig<='0';
    contadorpwm_sig<=contadorpwm+1;
    refpwm_sig<=refpwm;
elsE
    señalpwm_sig<='0';
    contadorpwm_sig<=(others=>'0');
    refpwm_sig<='0';
end if;
end process;



process(contadorsinusoidal,contadorfrecuencia,valorfrecuencia,updown,signosin,ACTIVO)
BEGIN
 if contadorsinusoidal<50 and updown='0' and contadorfrecuencia=valorfrecuencia and
ACTIVO='1' then
      contadorsinusoidal_sig<=contadorsinusoidal+1;
      contadorfrecuencia_sig<=(others=>'0');
      updown_sig<=updown;
      signosin_sig<=signosin;
 elsif contadorsinusoidal=50 and updown='0' and contadorfrecuencia=valorfrecuencia AND
ACTIVO='1' then
      contadorsinusoidal_sig<=contadorsinusoidal-1;
      updown_sig<='1';
      contadorfrecuencia_sig<=(others=>'0');
      signosin_sig<=signosin;
 elsif contadorsinusoidal<50 and contadorsinusoidal>0 and updown='1' and
contadorfrecuencia=valorfrecuencia AND ACTIVO='1' then
      contadorsinusoidal_sig<=contadorsinusoidal-1;
      contadorfrecuencia_sig<=(others=>'0');
      updown_sig<=updown;
      signosin_sig<=signosin;
 elsif contadorsinusoidal=0 and updown='1' and contadorfrecuencia=valorfrecuencia and
signosin='0' AND ACTIVO='1' then
      contadorsinusoidal_sig<=contadorsinusoidal+1;
      updown_sig<='0';
      contadorfrecuencia_sig<=(others=>'0');
      signosin_sig<='1';
 elsif contadorsinusoidal=0 and updown='1' and contadorfrecuencia=valorfrecuencia and
signosin='1' AND ACTIVO='1'   then
      contadorsinusoidal_sig<=contadorsinusoidal+1;
      updown_sig<='0';
      contadorfrecuencia_sig<=(others=>'0');
      signosin_sig<='0';
 elsif contadorfrecuencia<valorfrecuencia AND ACTIVO='1'   then
      contadorsinusoidal_sig<=contadorsinusoidal;
      contadorfrecuencia_sig<=contadorfrecuencia+1;
```

```
      updown_sig<=updown;
      signosin_sig<=signosin;
  else
      contadorsinusoidal_sig<=(others=>'0');
      contadorfrecuencia_sig<=(others=>'0');
      updown_sig<='0';
      signosin_sig<='0';
  end if;
  end process;


process(contadorsinusoidal,tipoperturbacion,table_value)
    begin
      if tipoperturbacion=1 then
                table_value<="0001";
      elsif contadorsinusoidal="000000" and tipoperturbacion=8 then
            table_value <= "0000";
      elsif contadorsinusoidal="000001" and tipoperturbacion=8 then
            table_value <= "0000";
           elsif contadorsinusoidal="000010" and tipoperturbacion=8 then
            table_value <= "0001";
      elsif contadorsinusoidal="000011" and tipoperturbacion=8 then
            table_value <= "0001";
           elsif contadorsinusoidal="000100" and tipoperturbacion=8 then
            table_value <= "0001";
      elsif contadorsinusoidal="000101" and tipoperturbacion=8 then
            table_value <= "0001";
      elsif contadorsinusoidal="000110" and tipoperturbacion=8 then
            table_value <= "0001";
           elsif contadorsinusoidal="000111" and tipoperturbacion=8 then
            table_value <= "0010";
           elsif contadorsinusoidal="001000" and tipoperturbacion=8 then
            table_value <= "0010";
           elsif contadorsinusoidal="001001" and tipoperturbacion=8 then
            table_value <= "0010";
           elsif contadorsinusoidal="001010" and tipoperturbacion=8 then
            table_value <= "0010";
           elsif contadorsinusoidal="001011" and tipoperturbacion=8 then
            table_value <= "0011";
           elsif contadorsinusoidal="001100" and tipoperturbacion=8 then
            table_value <= "0011";
           elsif contadorsinusoidal="001101" and tipoperturbacion=8 then
            table_value <= "0011";
           elsif contadorsinusoidal="001110" and tipoperturbacion=8 then
            table_value <= "0011";
           elsif contadorsinusoidal="001111" and tipoperturbacion=8 then
            table_value <= "0100";
           elsif contadorsinusoidal="010000" and tipoperturbacion=8 then
```

```
        table_value <= "0100";
      elsif contadorsinusoidal="010001" and tipoperturbacion=8 then
        table_value <= "0100";
      elsif contadorsinusoidal="010010" and tipoperturbacion=8 then
        table_value <= "0100";
elsif contadorsinusoidal="010011" and tipoperturbacion=8 then
        table_value <= "0100";
      elsif contadorsinusoidal="010100" and tipoperturbacion=8 then
        table_value <= "0101";
      elsif contadorsinusoidal="010101" and tipoperturbacion=8 then
        table_value <= "0101";
      elsif contadorsinusoidal="010110" and tipoperturbacion=8 then
        table_value <= "0101";
      elsif contadorsinusoidal="010111" and tipoperturbacion=8 then
        table_value <= "0101";
      elsif contadorsinusoidal="011000" and tipoperturbacion=8 then
        table_value <= "0101";
elsif contadorsinusoidal="011001" and tipoperturbacion=8 then
        table_value <= "0110";
      elsif contadorsinusoidal="011010" and tipoperturbacion=8 then
        table_value <= "0110";
      elsif contadorsinusoidal="011011" and tipoperturbacion=8 then
        table_value <= "0110";
      elsif contadorsinusoidal="011100" and tipoperturbacion=8 then
        table_value <= "0110";
      elsif contadorsinusoidal="011101" and tipoperturbacion=8 then
        table_value <= "0110";
      elsif contadorsinusoidal="011110" and tipoperturbacion=8 then
        table_value <= "0110";
      elsif contadorsinusoidal="011111" and tipoperturbacion=8 then
        table_value <= "0111";
      elsif contadorsinusoidal="100000" and tipoperturbacion=8 then
        table_value <= "0111";
      elsif contadorsinusoidal="100001" and tipoperturbacion=8 then
        table_value <= "0111";
      elsif contadorsinusoidal="100010" and tipoperturbacion=8 then
        table_value <= "0111";
      elsif contadorsinusoidal="100011" and tipoperturbacion=8 then
        table_value <= "0111";
elsif contadorsinusoidal="100100" and tipoperturbacion=8 then
        table_value <= "0111";
      elsif contadorsinusoidal="100101" and tipoperturbacion=8 then
        table_value <= "0111";
      elsif contadorsinusoidal="100110" and tipoperturbacion=8 then
        table_value <= "0111";
      elsif contadorsinusoidal="100111" and tipoperturbacion=8 then
        table_value <= "1000";
      elsif contadorsinusoidal="101000" and tipoperturbacion=8 then
```

```
            table_value <= "1000";
          elsif contadorsinusoidal="101001" and tipoperturbacion=8 then
            table_value <= "1000";
          elsif contadorsinusoidal="101010" and tipoperturbacion=8 then
            table_value <= "1000";
          elsif contadorsinusoidal="101011" and tipoperturbacion=8 then
            table_value <= "1000";
          elsif contadorsinusoidal="101100" and tipoperturbacion=8 then
            table_value <= "1000";
          elsif contadorsinusoidal="101101" and tipoperturbacion=8 then
            table_value <= "1000";
          elsif contadorsinusoidal="101110" and tipoperturbacion=8 then
            table_value <= "1000";
      elsif contadorsinusoidal="101111" and tipoperturbacion=8 then
            table_value <= "1000";
          elsif contadorsinusoidal="110000" and tipoperturbacion=8 then
            table_value <= "1000";
          elsif contadorsinusoidal="110001" and tipoperturbacion=8 then
            table_value <= "1000";
          elsif contadorsinusoidal="110010" and tipoperturbacion=8 then
            table_value <= "1000";
          else
                    table_value<="0000";
       end if;

end PROCESS;



process(contadorprbs,PRBS9,PRBS10,PRBS11,TAMANOREGISTRO,ACTIVO,inicsecPRBS)
begin
if contadorprbs=0 AND ACTIVO='1' AND TAMANOREGISTRO=9 AND PRBS9="000000001" THEN
   contadorprbs_sig<=contadorprbs+1;
   PRBS9_sig<=PRBS9;
   PRBS10_SIG<=PRBS10;
   PRBS11_SIG<=PRBS11;
   inicsecPRBS_sig<=not(inicsecPRBS);
elsif contadorprbs=0 AND ACTIVO='1' AND TAMANOREGISTRO=10 AND PRBS10="0000000001"
THEN
   contadorprbs_sig<=contadorprbs+1;
   PRBS9_sig<=PRBS9;
   PRBS10_SIG<=PRBS10;
   PRBS11_SIG<=PRBS11;
inicsecPRBS_sig<=not(inicsecPRBS);
elsif contadorprbs=0 AND ACTIVO='1' AND TAMANOREGISTRO=11 AND PRBS11="00000000001"
THEN
   contadorprbs_sig<=contadorprbs+1;
```

```
   PRBS9_sig<=PRBS9;
   PRBS10_SIG<=PRBS10;
   PRBS11_SIG<=PRBS11;
   inicsecPRBS_sig<=not(inicsecPRBS);
elsif contadorprbs<499 and ACTIVO='1' THEN
   contadorprbs_sig<=contadorprbs+1;
   PRBS9_sig<=prbs9;
   PRBS10_SIG<= prbs10;
   PRBS11_SIG<= prbs11;
   inicsecPRBS_sig<=inicsecPRBS;
elsif contadorprbs=499 AND ACTIVO='1' THEN
   contadorprbs_sig<=(others=>'0');
   PRBS9_sig<=(PRBS9(0)XOR PRBS9(5))&(PRBS9(8 downto 1));
   PRBS10_SIG<=(PRBS10(0)XOR PRBS10(3))&(PRBS10(9 DOWNTO 1));
   PRBS11_SIG<=(PRBS11(0)XOR PRBS11(2))&(PRBS11(10 DOWNTO 1));
   inicsecPRBS_sig<=inicsecPRBS;
else
   contadorprbs_sig<=(others=>'0');
   PRBS9_sig<="000000001";
   PRBS10_SIG<= "0000000001";
   PRBS11_SIG<="00000000001";
   inicsecPRBS_sig<='0';

end if;
end process;



process(P_ESTADO,REGVAR,REG1,REG2,CONTADOR,NUMEROBIT,SECONDBYTE,REGPARIDAD,REG
2PARIDAD,REG3PARIDAD,BYTE1,BYTE2,rxd2)
begin
case p_estado is
   when REPOSO=>
     if (RXD2='1')then
           n_estado<=REPOSO;
         else
           n_estado<=START;
         end if;
           contador_sig<=(others=>'0');
           numerobit_sig<=(OTHERS=>'0');
           REGVAR_sig<=REGVAR;
           REG1_sig<=REG1;
           REG2_SIG<=REG2;
           secondbyte_sig<=secondbyte;
           regparidad_sig<=regparidad;
           reg2paridad_sig<=reg2paridad;
           reg3paridad_sig<=reg3paridad;
```

```
            BYTE1_sig<=BYTE1;
            BYTE2_sig<=BYTE2;
when START=>
   if (contador<2604)or((contador>2604)and (contador<5209))then
            contador_sig<=contador+1;
     numerobit_sig<=numerobit;
            n_estado<=START;
            REGVAR_sig<=REGVAR;
            REG1_sig<=REG1;
            REG2_SIG<=REG2;
   elsif (contador=2604) and (rxd2='0')then
     contador_sig <=contador+1;
     numerobit_sig<=numerobit;
            n_estado<=START;
            REGVAR_sig<=REGVAR;
            REG1_sig<=REG1;
            REG2_sig<=REG2;
          elsif (contador=2604) and (rxd2='1')then
     contador_sig <=contador;
     numerobit_sig<=numerobit;
            n_estado<=REPOSO;
            REGVAR_sig<=REGVAR;
            REG1_sig<=REG1;
            REG2_sig<=REG2;
          else
     contador_sig <=(others=>'0');
     numerobit_sig<=numerobit+1;
            n_estado<=DATOS;
            REGVAR_sig<=REGVAR;
            REG1_sig<=REG1;
            REG2_sig<=REG2;
          end if;
            secondbyte_sig<=secondbyte;
            regparidad_sig<=regparidad;
            reg2paridad_sig<=reg2paridad;
            reg3paridad_sig<=reg3paridad;
            BYTE1_sig<=BYTE1;
            BYTe2_sig<=BYTE2;
 when DATOS=>
   if (contador<2604)or ((contador>2604) and (contador<5209))    then
            n_estado<=datos;
            Contador_sig<=Contador+1;
            REGVAR_sig<=REGVAR;
            REG2_sig<=REG2;
            REG1_sig<=REG1;
            numerobit_sig<=numerobit;
            regparidad_sig<=regparidad;
            reg2paridad_sig<=reg2paridad;
```

```
        reg3paridad_sig<=reg3paridad;
      elsif (contador=2604) and (secondbyte='0') and (numerobit<4) then
        n_estado<=datos;
        contador_sig<=contador+1;
        REGVAR_sig<=REGVAR;
        REG1_sig<=rxd2&REG1(2 downto 1);
        REG2_sig<=reg2;
        numerobit_sig<=numerobit;
        regparidad_sig<=regparidad;
        reg2paridad_sig<=reg2paridad;
        reg3paridad_sig<=reg3paridad;
      elsif (contador=2604) and (secondbyte='0') and (numerobit<8) then
        n_estado<=datos;
        contador_sig<=contador+1;
        REG1_sig<=REG1;
        REGVAR_sig<=rxd2&regvar(3 downto 1);
        REG2_SIG<=REG2;
        numerobit_SIG<=NUMEROBIT;
        regparidad_sig<=regparidad;
        reg2paridad_sig<=reg2paridad;
        reg3paridad_sig<=reg3paridad;
      elsif (contador=2604) and (secondbyte='1') and (numerobit<8) then
        n_estado<=datos;
        contador_SIG<=contador+1;
        REG2_sig<=rxd2&REG2(6 DOWNTO 1);
        REGVAR_sig<=REGVAR;
        REG1_SIG<=REG1;
        numerobit_sig<=numerobit;
        regparidad_sig<=regparidad;
        reg2paridad_sig<=reg2paridad;
        reg3paridad_sig<=reg3paridad;
      elsif (contador=5209) and (numerobit<8) then
  n_estado<=datos;
        contador_SIG<=(others=>'0');
        REG2_sig<=REG2;
        REGVAR_sig<=REGVAR;
        REG1_SIG<=REG1;
        numerobit_sig<=numerobit+1;
        regparidad_sig<=regparidad;
        reg2paridad_sig<=reg2paridad;
        reg3paridad_sig<=reg3paridad;
elsif (contador=2604) and (numerobit=8) then
        n_estado<=datos;
        contador_SIG<=contador+1;
        REG2_sig<=REG2;
        REGVAR_sig<=REGVAR;
        REG1_SIG<=REG1;
        numerobit_sig<=numerobit;
```

```
            regparidad_sig<=regparidad;
            reg2paridad_sig<=reg2paridad;
        reg3paridad_sig<=rxd2;
           else
            n_estado<=paridad;
            contador_SIG<=(others=>'0');
            REG2_sig<=REG2;
            REGVAR_sig<=REGVAR;
            REG1_SIG<=REG1;
            numerobit_sig<=numerobit+1;
            regparidad_sig<=regparidad;
            reg2paridad_sig<=reg2paridad;
            reg3paridad_sig<=reg3paridad;
           end if;
            secondbyte_sig<=secondbyte;
            BYTE1_sig<=BYTE1;
            BYTE2_sig<=BYTE2;
    when PARIDAD=>
        if (contador<2604)or((contador>2604)and(contador<5209)) then
            n_estado<=paridad;
            contador_sig<=contador+1;
            numerobit_sig<=numerobit;
            regparidad_sig<=regparidad;
            reg2paridad_sig<=reg2paridad;
            reg3paridad_sig<=reg3paridad;
            elsif (contador=2604)and (secondbyte='0') then
            n_estado<=paridad;
            contador_sig<=contador+1;
            numerobit_sig<=numerobit;
            regparidad_sig<=rxd2;
            reg2paridad_sig<=(((REG1(0)xor REG1(1))xor(REG1(2) xor REGVAR(0)))xor
((REGVAR(1)xor REGVAR(2))xor (REGVAR(3)xor REG3PARIDAD)));
            reg3paridad_sig<=reg3paridad;
            elsif (contador=2604)and (secondbyte='1') then
            n_estado<=paridad;
            contador_sig<=contador+1;
            numerobit_sig<=numerobit;
            regparidad_sig<=rxd2;
            reg2paridad_sig<=(((REG2(0)xor REG2(1))xor(REG2(2) xor REG2(3)))xor ((REG2(4)xor
REG2(5))xor (REG2(6)xor REG3PARIDAD)));
            reg3paridad_sig<=reg3paridad;
            else
            n_estado<=stop;
            contador_sig<=(others=>'0');
            numerobit_sig<=numerobit+1;
            regparidad_sig<=regparidad;
            reg2paridad_sig<=reg2paridad;
            reg3paridad_sig<=reg3paridad;
```

```
            end if;
            REG1_sig<=REG1;
            REG2_sig<=REG2;
            REGVAR_sig<=REGVAR;
            secondbyte_sig<=secondbyte;
            BYTE1_sig<=BYTE1;
            BYTe2_sig<=BYTe2;
          when STOP=>
          if (contador<5209) then
            n_estado<=STOP;
            contador_sig<=contador+1;
            secondbyte_sig<=secondbyte;
            BYTE2_sig<=byte2;
            BYTE1_sig<=byte1;
          elsif (Contador=5209) and (secondbyte='0') AND (REGPARIDAD=REG2PARIDAD)
AND((regvar="0010")or(regvar="0100"))then
            n_estado<=reposo;
            secondbyte_sig<='0';
            contador_sig<=contador;
            BYTE2_sig<=byte2;
            BYTE1_sig<=REGVAR&REG1;
          elsif (Contador=5209) and (secondbyte='0') AND (REGPARIDAD=REG2PARIDAD) and
((regvar="0110")or(regvar="1000")or (regvar="1010") or (regvar="1110")or (regvar="0111")or
(regvar="0001")or(regvar="1100")) then
            n_estado<=reposo;
            secondbyte_sig<='1';
            contador_sig<=contador;
            BYTE2_sig<=byte2;
            BYTE1_sig<=byte1;
          elsif (Contador=5209) and (secondbyte='1') AND (regparidad=REG2PARIDAD) then
            n_estado<=reposo;
       secondbyte_sig<= '0';
            Contador_sig<=contador;
            BYTE2_sig<=REG2;
            BYTE1_sig<=REGVAR&REG1;
          else
            n_estado<=reposo;
            secondbyte_SIG<='0';
            contador_sig<=contador;
            BYTE2_sig<=BYTE2;
            BYTE1_sig<=BYTE1;
          end if;
            numerobit_sig<=numerobit;
            REG1_sig<=REG1;
            REG2_sig<=REG2;
            REGVAR_sig<=REGVAR;
            regparidad_sig<=regparidad;
            reg2paridad_sig<=reg2paridad;
```

```
            reg3paridad_sig<=reg3paridad;
          END CASE;

end process;

end Behavioral;
```

ANEXO B : CÓDIGO EN PROCESSING DE LA INTERFAZ GRÁFICA

```
import processing.serial.*

    PFont font; // Declare the variable

    float segundobytedutycte2;

    byte    primerbytedutycte,segundobytedutycte;

    byte    primerbytedutyseno,segundobytedutyseno;

    byte    primerbytetiponda,segundobytetiponda;

    float segundobytefrecseno2;

    byte    primerbytefrecseno,segundobytefrecseno;

    byte    primerbytedutyprbs,segundobytedutyprbs;

    byte    primerbytesizeprbs,segundobytesizeprbs;



    float valordutyctereg= 250;

    float valordutysenoreg = 1;

    int valortipondareg= 1;

    float valorfrecsenoreg = 499;

    float valordutyprbsreg=2;

    int valorsizeprbsreg=9;


    float valordutyctereal=valordutyctereg/500;

    float valordutysenoreal=valordutysenoreg/500;

    int valortipondareal= 1;
```

```
    float valorfrecsenoreal=1/((valorfrecsenoreg+1)*50*4*20*0.000000001);
    float valordutyprbsreal=valordutyprbsreg/500;
    int valorsizeprbsreal=9;

void setup() {
    size (500, 600);
}
void draw() {
    background(230);
    stroke(255);
    fill(10,113,247);
    rect(250,60,300,40);

    //Botón duty medio
    font = loadFont("Arial-BoldItalicMT-15.vlw"); //Carga la fuente
    textFont(font);
    textSize(13);
    fill(255);
    text("Valor medio Duty",140, 65);

      //Botón Duty senoidal
        fill(10,113,247,180);
        rect(250,120,300, 40);
        font = loadFont("Arial-BoldItalicMT-15.vlw");
        textFont(font);
        textSize(13);
```

```
  fill(255);
  text("sinusoidal Duty", 147, 132);
  textFont(font);
  textSize(13);
  fill(255);
  text("Amplitud variación", 140, 116);


//Botón tipo perturbación
  fill(10,113,247,180);
  rect(250,180,300, 40);
  font = loadFont("Arial-BoldItalicMT-15.vlw");
  textFont(font);
  textSize(13);
  fill(255);
  text("Tipo de perturbación ", 135, 186);


  //Botón frecuencia sinusoidal
  fill(10, 113, 247,130);
  rect(250,240, 300, 40);
  textFont(font);
  textSize(13);
  fill(50);
  text("Frecuencia variación",135, 236);
  textFont(font);
```

```
  textSize(13);
  fill(50);
  text("sinusoidal Duty", 145, 252);


//Botón duty PRBS
  fill(10,113,247,70);
  rect(250,300,300, 40);
  textFont(font);
  textSize(13);
  fill(50);
  text("Magnitud incremento", 135, 296);
  textFont(font);
  textSize(13);
  fill(50);
  text("pseudoaleatorio Duty",135, 312);

  fill(10,113,247,60);
  rect(250,360,300,40);
  textFont(font);
  textSize(13);
  fill(50);
  text("Tamaño del registro",135,356);
  text("generador de PRBS",135,372);
```

```
  fill(200);
stroke(100);
ellipse(400,60,40,40);

fill(50,200,50);
noStroke();
rectMode(CENTER);
rect(400,60,20,6);
rect(400,60,6,20);
fill(200);
stroke(100);
ellipse(400,120,40,40);
fill(50,200,50);
noStroke();
rect(400,120,20,6);
rect(400,120,6,20);
fill(200);
stroke(100);
ellipse(400,180,40,40);
fill(50,200,50);
noStroke();
rect(400,180,20,6);
rect(400,180,6,20);
```

```
fill(215);
stroke(100);
ellipse(400,240,40,40);
fill(50,200,50);
noStroke();
rect( 400,240,20,6);
rect(400,240,6,20);

//Botón aumentar
fill(200);
stroke(100);
ellipse(400,300,40,40);
fill(50,200,50);
noStroke();
rect(400,300,20,6);
rect(400,300,6,20);
fill(200);
stroke(100);
ellipse(400,360,40,40);
fill(50,200,50);
noStroke();
rect(400,360,20,6);
rect(400,360,6,20);
```

```
fill(215);
stroke(100);
ellipse(98,240,40,40);
fill(50,200,50);
noStroke();
rect( 98,240,20,6);

fill(215);
stroke(100);
ellipse(98,180,40,40);
fill(50,200,50);
noStroke();
rect( 98,180,20,6);

fill(215);
stroke(100);
ellipse(98,120,40,40);
fill(50,200,50);
noStroke();
rect( 98,120,20,6);

fill(215);
stroke(100);
```

```
ellipse(98,60,40,40);
fill(50,200,50);
noStroke();
rect( 98,60,20,6);

fill(215);
stroke(100);
ellipse(98,240,40,40);
fill(50,200,50);
noStroke();
rect( 98,240,20,6);

fill(215);
stroke(100);
ellipse(98,300,40,40);
fill(50,200,50);
noStroke();
rect( 98,300,20,6);

fill(215);
stroke(100);
ellipse(98,360,40,40);
fill(50,200,50);
noStroke();
rect( 98,360,20,6);
```

```
//huecos
textSize(15);
fill(255);
rect(320,60,100,28);
rect(320,120,100,28);
rect(320,180,100,28);
rect(320,240,100,28);
rect(320,300,100,28);
rect(320,360,100,28);
fill(50,50,200);
text(valordutyctereal,293,67);
text(valordutysenoreal,293,127);
if (valortipondareal<2){
text("Cuadrada",285,187);}
if (valortipondareal>7){
text("Senoidal",285,187);}
textSize(13);
text(valorfrecsenoreal,283,247);
text("Hz",348,247);
textSize(15);
text(valordutyprbsreal,293,307);
textSize(16);
text(valorsizeprbsreal,315,367);
```

```
  //Start


  fill(50,230,50);
  rect(330,510,100,40);
  textSize(20);
  fill(255);
  text("START",295,516);


  fill(255,50,50);
  rect(190,510,100,40);
  textSize(20);
  fill(255);
  text("STOP",162,516);







}
void mousePressed(){


if (((mouseX>380)&&(mouseX<420))&&((mouseY>40)&&(mouseY<80))){
    if (valordutyctereg<500){
```

```
      valordutyctereg=valordutyctereg+25;
      valordutyctereal=valordutyctereg/500;
        }
    }
if (((mouseX>78)&&(mouseX<118))&&((mouseY>40)&&(mouseY<80))){
      if (valordutyctereg>0){
    valordutyctereg=valordutyctereg-25;
    valordutyctereal=valordutyctereg/500;
    }
  }
if (((mouseX>380)&&(mouseX<420))&&((mouseY>100)&&(mouseY<140))){
    if    (valordutysenoreg<10){
    valordutysenoreg=valordutysenoreg+1;
    valordutysenoreal=valortipondareg*valordutysenoreg/500;
         }
  }
if (((mouseX>78)&&(mouseX<118))&&((mouseY>100)&&(mouseY<140))){
      if (valordutysenoreg>0){
        valordutysenoreg=valordutysenoreg-1;
        valordutysenoreal=valortipondareg*valordutysenoreg/500;
        }
  }
  if (((mouseX>380)&&(mouseX<420))&&((mouseY>160)&&(mouseY<200))){
    if    (valortipondareg<2){
    valortipondareg=8;
    valortipondareal=valortipondareg;
```

```
    valordutysenoreal=valortipondareg*valordutysenoreg/500;
        }
  }
if (((mouseX>78)&&(mouseX<118))&&((mouseY>160)&&(mouseY<200))){
     if (valortipondareg>7){              //si es 8 ciclos pasa a cuadrada
       valortipondareg=1;
       valortipondareal=valortipondareg;
       valordutysenoreal=valortipondareg*valordutysenoreg/500;
       }
  }

if (((mouseX>380)&&(mouseX<420))&&((mouseY>220)&&(mouseY<260))){
    if (valorfrecsenoreg>498){
    valorfrecsenoreg= 249;
    valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
    }
    else if ((valorfrecsenoreg>248)&&(valorfrecsenoreg<250)){
    valorfrecsenoreg= 149;
    valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
    }

    else if ((valorfrecsenoreg>148)&&(valorfrecsenoreg<150)){
    valorfrecsenoreg= 124;
    valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
    }
    else if ((valorfrecsenoreg>123)&&(valorfrecsenoreg<125)){
```

```
valorfrecsenoreg= 99;
valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
}
else if ((valorfrecsenoreg>98)&&(valorfrecsenoreg<100)){
valorfrecsenoreg= 74;
valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
}

else if ((valorfrecsenoreg>73)&&(valorfrecsenoreg<75)){
valorfrecsenoreg= 49;
valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
}

else if ((valorfrecsenoreg>48)&&(valorfrecsenoreg<50)){
valorfrecsenoreg= 24;
valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
}

else if ((valorfrecsenoreg>23)&&(valorfrecsenoreg<25)){
valorfrecsenoreg= 14;
valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
}}

if (((mouseX>78)&&(mouseX<118))&&((mouseY>220)&&(mouseY<260))){
if (valorfrecsenoreg<15){
  valorfrecsenoreg = 24;
```

```
  valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
}

else if ((valorfrecsenoreg>23)&&(valorfrecsenoreg<25)){
  valorfrecsenoreg = 49;
  valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
}

else if ((valorfrecsenoreg>48)&&(valorfrecsenoreg<50)){
  valorfrecsenoreg = 74;
  valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
}

else if ((valorfrecsenoreg>73)&&(valorfrecsenoreg<75)){
  valorfrecsenoreg = 99;
  valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
}

else if ((valorfrecsenoreg>98)&&(valorfrecsenoreg<100)){
  valorfrecsenoreg = 124;
  valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
}

else if ((valorfrecsenoreg>123)&&(valorfrecsenoreg<125)){
  valorfrecsenoreg = 149;
  valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
```

```
    }

    else if ((valorfrecsenoreg>148)&&(valorfrecsenoreg<150)){
      valorfrecsenoreg = 249;
      valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
    }

    else if ((valorfrecsenoreg>248)&&(valorfrecsenoreg<250)){
      valorfrecsenoreg = 499;
      valorfrecsenoreal=1/((50*4*20*0.000000001)*(valorfrecsenoreg+1));
    }}

  if (((mouseX>380)&&(mouseX<420))&&((mouseY>280)&&(mouseY<320))){
      if (valordutyprbsreg<40){
    valordutyprbsreg=valordutyprbsreg+1;
    valordutyprbsreal=valordutyprbsreg/500;
      }
  }
if (((mouseX>78)&&(mouseX<118))&&((mouseY>280)&&(mouseY<320))){
    if (valordutyprbsreg>0){
      valordutyprbsreg=valordutyprbsreg-1;
      valordutyprbsreal=valordutyprbsreg/500;    }
  }
    if (((mouseX>380)&&(mouseX<420))&&((mouseY>340)&&(mouseY<380))){
      if(valorsizeprbsreg<11){
          valorsizeprbsreg=valorsizeprbsreg+1;
```

```
        valorsizeprbsreal=valorsizeprbsreg;}
      }

  if (((mouseX>78)&&(mouseX<118))&&((mouseY>340)&&(mouseY<380))){
    if (valorsizeprbsreg>9){
      valorsizeprbsreg=valorsizeprbsreg-1;
      valorsizeprbsreal=valorsizeprbsreg;    }
  }

if (((mouseX>280)&&(mouseX<380))&&((mouseY>490)&&(mouseY<530))){
    if (valordutyctereg>127){
        if(valordutyctereg<256){
          primerbytedutycte=49;
          segundobytedutycte2=valordutyctereg-128;
          segundobytedutycte=byte(segundobytedutycte2);

                }

        if((valordutyctereg>255)&&(valordutyctereg<384)){
            primerbytedutycte=50;
            segundobytedutycte2=valordutyctereg-256;
            segundobytedutycte=byte(segundobytedutycte2);


        if(valordutyctereg>383){
          primerbytedutycte=51;
```

```
  segundobytedutycte2=valordutyctereg-384;
  segundobytedutycte=byte(segundobytedutycte2);

 }}
 if(valordutyctereg<128){
    primerbytedutycte=48;
    segundobytedutycte=byte(valordutyctereg);
 }

 primerbytedutyseno=64;
 segundobytedutyseno=byte(valordutysenoreg);

 primerbytetiponda=96;
 segundobytetiponda=byte(valortipondareg);

if (valorfrecsenoreg>127){
 if(valorfrecsenoreg<256){
  primerbytefrecseno=81;
  segundobytefrecseno2=valorfrecsenoreg-128;
  segundobytefrecseno=byte(segundobytefrecseno2);

        }

 if((valorfrecsenoreg>255)&&(valorfrecsenoreg<384)){
    primerbytefrecseno=82;
    segundobytefrecseno2=valorfrecsenoreg-256;
```

```
      segundobytefrecseno=byte(segundobytefrecseno2);




   }

   if(valorfrecsenoreg>383){
     primerbytefrecseno=83;
     segundobytefrecseno2=valorfrecsenoreg-384;
     segundobytefrecseno=byte(segundobytefrecseno2);

   }}
   if(valorfrecsenoreg<128){
      primerbytefrecseno=80;
      segundobytefrecseno=byte(valorfrecsenoreg);
   }



primerbytedutyprbs=112;
segundobytedutyprbs=byte(valordutyprbsreg);

primerbytesizeprbs=56;
segundobytesizeprbs=byte(valorsizeprbsreg);
```

```
Serial myPort;

myPort = new Serial(this,Serial.list()[1],9600,'E',8,2.0);


myPort.write(primerbytedutycte);


myPort.write(segundobytedutycte);


myPort.write(primerbytedutyseno);


myPort.write(segundobytedutyseno);


myPort.write(primerbytetiponda);


myPort.write(segundobytetiponda);


myPort.write(primerbytefrecseno);


myPort.write(segundobytefrecseno);


myPort.write(primerbytedutyprbs);


myPort.write(segundobytedutyprbs);


myPort.write(primerbytesizeprbs);
```

```
        myPort.write(segundobytesizeprbs);




        myPort.write(16);


        myPort.stop();
        }
        if (((mouseX>140)&&(mouseX<240))&&((mouseY>490)&&(mouseY<530))){
        Serial myPort;
        myPort = new Serial(this,Serial.list()[1],9600,'E',8,2.0);
        myPort.write(32);


        myPort.stop();
        }



}
```

ANEXO C: CÓDIGO EN SCILAB PARA EL PROCESADO EXPERIMENTAL

DFT

```
dat=readxls('C:\Users\Javier\Desktop\experimentales medidas\medidas exper fourier xls\10000.xls')
typeof(dat)
dato=dat(1) //get the first sheet
sen400=dato(:,1);
VO=dato(:,2);
cuad=dato(:,3);
j=1;
u=0;
pi=3.14159265359;
fcon=10^(5);
fper=10000;
N=fcon/fper;
fs=50*10^(6);
muestperiodoconm=fs/fcon;
for i=1:(N*muestperiodoconm-1)
      if (((u>1.77) & (sen400(i)<1.77)) | ((u<1.77) & (sen400(i)>1.77))) then
            VO2(j)=VO(i);
            cuad2(j)=cuad(i);
            j=j+1;
        end
        u=sen400(i);
end

for i=1:(N/2)
  cuad3(N/2+i)=cuad2(i);
  cuad3(i)=cuad2(i+N/2);
end

Y=fft(VO2);
X=fft(cuad3*0.016*pi/(3.333*2));
fdt2=Y(2)/X(2);
```

```
i=sqrt(-1);

G(1)= 5.2162348 - 1.619577*i
G(2)= 4.1509006 - 4.1161898*i;
G(3)=- 0.8037796 - 4.1472311*i
G(4)=-1.6626323 - 2.3220557*i;
G(5)=-1.0243681 - 2.0150269*i;
G(6)=-0.4988558 - 0.2536855*i;
```

```
G(7)=-1.1469187 - 0.3425567*i;

G(8)= -0.8413712 - 0.6413915*i;

f(1)=500;
f(2)=1000;
f(3)=10000/6;
f(4)=2000;
f(5)=2500;
f(6)=10000/3;
f(7)=5000;

f(8)=100000/6;

pi=3.14159265359;

VG=5;
L=68E-06;
C=220E-06;
RL=0.098;
RC=0.08;
R0=2.5;
IL0=0;
n1=RC*C*VG;
n0=VG;
d2=L*C*(1+RC/R0);
d1=L/R0+(RL*R0 + RC*R0+ RC*RL)*C/R0;
d0=1+ RL/R0;

for i=1:8
mod(i)=abs(G(i));
mag(i)=20*log10(mod(i));
fas(i)=(180/pi)*atan(imag(G(i)),real(G(i)));
frec(i)=log10(f(i));
x(i)=2*pi*f(i)*sqrt(-1);
num(i)=n1*x(i)+n0;
den(i)=d2*x(i)^2+d1*x(i)+d0;
Gp(i)=num(i)/den(i);
modr(i)=abs(Gp(i));
magr(i)=20*log10(modr(i))
fasr(i)=(180/pi)*atan(imag(Gp(i)),real(Gp(i)));
end

plot(frec,fas,frec,fasr)
```

PRBS

```
M = read_csv('tek0018ALL',16,0);
sen400=M(:,2);
VO=M(:,3);
PRBS=M(:,4);

numerobits=9;
fs=(2^(11-numerobits))*10^(6);
fconm=10^(5);
m=fs/fconm;
N=2^(numerobits)-1;
j=1;
i=1;
u=sen400(1);
while(j<(N))
      if (((u>1.77) & (sen400(i)<1.77)) | ((u<1.77) & (sen400(i)>1.77))) then
            u9(j)=PRBS(i);
            y9(j)=VO(i);
            j=j+1;
        end
        u=sen400(i);
        i=i+1;
end
```

```
pi=3.14159265359;
N9=511;
N10=1023;
N11=2047;
res9=10^5/N9;
res10=10^5/N10;
res11=10^5/N11;

for i=1:N9
      y9(N9+i)=y9(i);
      u9(N9+i)=u9(i);
end
for i=1:N10
      y10(N10+i)=y10(i);
      u10(N10+i)=u10(i);
end
for i=1:N11
      y11(N11+i)=y11(i);
      u11(N11+i)=u11(i);
end

for i=1:N9
   Ruy9(i)=0;
   for j=1:(N9)
        Ruy9(i)=Ruy9(i)+u9(j)*y9(j+i-1);
```

```
    end
    Ruy9(i)=Ruy9(i)/N9;
end

for i=1:N10
    Ruy10(i)=0;
    for j=1:(N10)
        Ruy10(i)=Ruy10(i)+u10(j)*y10(j+i-1);
    end
    Ruy10(i)=Ruy10(i)/N10;
end

for i=1:N11
    Ruy11(i)=0;
    for j=1:(N11)
        Ruy11(i)=Ruy11(i)+u11(j)*y11(j+i-1);
    end
    Ruy11(i)=Ruy11(i)/N11;
end

Ruy9=Ruy9/((0.008)^2);
Ruy10=Ruy10/((0.008)^2);
Ruy11=Ruy11/((0.008)^2);
Y9=(fft(Ruy9));
Y10=(fft(Ruy10));
Y11=(fft(Ruy11));

pi=3.14159265359;
VG=5;
L=68E-06;
C=220E-06;
RL=0.098;
RC=0.08;
R0=2.5;
IL0=0;
n1=RC*C*VG;
n0=VG;
d2=L*C*(1+RC/R0);
d1=L/R0+(RL*R0 + RC*R0+ RC*RL)*C/R0;
d0=1+ RL/R0;
N=10000;
res=10^5/N;
for i=1:N
f(i)=res*i;
f2(i)=log10(f(i));
x(i)=2*pi*f(i)*sqrt(-1);
num(i)=n1*x(i)+n0;
den(i)=d2*x(i)^2+d1*x(i)+d0;
```

```
G(i)=num(i)/den(i);
modreal(i)=20*log10(abs(G(i)));
fasereal(i)=(180/pi)*(atan(imag(G(i)),real(G(i))));
end

for i=2:((511+1)/2)
modY9(i)=20*log10(abs(Y9(i)));
fasY9(i)=(180/pi)*atan(imag(Y9(i)),real(Y9(i)));
frec9(i)=log10(res9*(i-1));
end

for i=2:((1023+1)/2)
modY10(i)=20*log10(abs(Y10(i)));
fasY10(i)=(180/pi)*atan(imag(Y10(i)),real(Y10(i)));
frec10(i)=log10(res10*(i-1));
end

for i=2:((2047+1)/2)
modY11(i)=20*log10(abs(Y11(i)));
fasY11(i)=(180/pi)*atan(imag(Y11(i)),real(Y11(i)));
frec11(i)=log10(res11*(i-1));
end

for i=1:(((511+1)/2)-1)
mod2Y9(i)=modY9(i+1);
fas2Y9(i)=fasY9(i+1);
frec29(i)= frec9(i+1);
end

for i=1:(((1023+1)/2)-1)
mod2Y10(i)=modY10(i+1);
fas2Y10(i)=fasY10(i+1);
frec210(i)= frec10(i+1);
end

for i=1:(((2047+1)/2)-1)
mod2Y11(i)=modY11(i+1);
fas2Y11(i)=fasY11(i+1);
frec211(i)= frec11(i+1);
end

plot(frec29,fas2Y9,frec210,fas2Y10,frec211,fas2Y11,f2,fasereal)
```

ANEXO D: CÓDIGO EN SCILAB PARA LA SIMULACIONES

DFT

```
VO.time=0;
VO.values=0;
senocreado.time=0;
senocreado.values=0;
ondacuadrada.time=0;
ondacuadrada.values=0;
ondacuadrada2.time=0;
ondacuadrada2.values=0;

pi=3.14159265359;
frecuencia=1000;
tentre200valores=1/(50*4*frecuencia);
incrminduty=1/500;//incremento mínimo.
numerodeunidades=8;// probamos 1,2 ,4 y 8 (sobre 500)
numerodeperiodos=1;//se modifica : 1,5 y 10.CUIDADADO
tiempopermanente=5*10^-3;
numeroperiodosperm=round(tiempopermanente*frecuencia);

for i=1:(numeroperiodosperm+numerodeperiodos)
for j=1:2
for    k=1:2
for l=1:50
if (j==1) and (k==1)
senocreado.time((i-1)*200+(l))=(tentre200valores)*((i-1)*200+(l-1));
senocreado.values((i-1)*200+(l))=incrminduty*round(numerodeunidades*(sin((l-1)*pi/(2*50))));
end
if (j==1) and (k==2)
senocreado.time((i-1)*200+(l)+50)= (tentre200valores)*((i-1)*200+(l-1)+50);
senocreado.values((i-1)*200+(l)+50)=incrminduty*round(numerodeunidades*(sin((49+l)*pi/(2*50)
)));
end
if (j==2) and (k==1)
senocreado.time((i-1)*200+(l)+100)=(tentre200valores)*((i-1)*200+(l-1)+100);
senocreado.values((i-1)*200+(l)+100)=-incrminduty*round(numerodeunidades*(sin((l-1)*pi/(2*50
))));
end
if (j==2) and (k==2)
senocreado.time((i-1)*200+(l)+150)=(tentre200valores)*((i-1)*200+(l-1)+150);
senocreado.values((i-1)*200+(l)+150)=-incrminduty*round(numerodeunidades*(sin((49+l)*pi/(2*5
0))));
end
end
end
```

```
end
end
  s2=senocreado.values;
for i=1:(numeroperiodosperm+numerodeperiodos)
for j=1:2        // semiperiodo positivo o negativo
for l=1:100      // 100 valores en un semiperiodo
if (j==1)
ondacuadrada.time((i-1)*200+(l))=(tentre200valores)*((i-1)*200+(l-1));
ondacuadrada.values((i-1)*200+(l))=incrminduty*numerodeunidades;
end
if (j==2)
ondacuadrada.time((i-1)*200+(l)+100)= (tentre200valores)*((i-1)*200+(l-1)+100);
ondacuadrada.values((i-1)*200+(l)+100)=-incrminduty*numerodeunidades;
end
end
end
end
--------------------------------------------------------------------------------
i=sqrt(-1);
pi=3.14159265359;

VG=5;
L=68E-06;
C=220E-06;
RL=0.098;
RC=0.08;
R0=2.5;
IL0=0;
n1=RC*C*VG;
n0=VG;
d2=L*C*(1+RC/R0);
d1=L/R0+(RL*R0 + RC*R0+ RC*RL)*C/R0;
d0=1+ RL/R0;

for i=1:19
mod8(i)=abs(G8(i));
mag8(i)=20*log10(mod8(i));
fas8(i)=(180/pi)*atan(imag(G8(i)),real(G8(i)));
mod4(i)=abs(G4(i));
mag4(i)=20*log10(mod4(i));
fas4(i)=(180/pi)*atan(imag(G4(i)),real(G4(i)));
mod2(i)=abs(G2(i));
mag2(i)=20*log10(mod2(i));
fas2(i)=(180/pi)*atan(imag(G2(i)),real(G2(i)));
mod(i)=abs(G(i));
mag(i)=20*log10(mod(i));
fas(i)=(180/pi)*atan(imag(G(i)),real(G(i)));
frec(i)=log10(f(i));
```

```
end

res=2;
for i=1:10000
f2(i)=res*(i);
frec2(i)=log10(f2(i));
x(i)=2*pi*f2(i)*sqrt(-1);
num(i)=n1*x(i)+n0;
den(i)=d2*x(i)^2+d1*x(i)+d0;
Gp(i)=num(i)/den(i);
modr(i)=abs(Gp(i));
magr(i)=20*log10(modr(i));
fasr(i)=(180/pi)*atan(imag(Gp(i)),real(Gp(i)));
end

for i=1:19

x2(i)=2*pi*f(i)*sqrt(-1);
num2(i)=n1*x2(i)+n0;
den2(i)=d2*x2(i)^2+d1*x2(i)+d0;
Gp2(i)=num2(i)/den2(i);
modrr(i)=abs(Gp2(i));
magrr(i)=20*log10(modrr(i))
fasrr(i)=(180/pi)*atan(imag(Gp2(i)),real(Gp2(i)));
emod8(i)=abs((mod8(i)-modrr(i))*100/modrr(i));
efas8(i)=abs((fas8(i)-fasrr(i))*100/fasrr(i));
emod4(i)=abs((mod4(i)-modrr(i))*100/modrr(i));
efas4(i)=abs((fas4(i)-fasrr(i))*100/fasrr(i));
emod2(i)=abs((mod2(i)-modrr(i))*100/modrr(i));
efas2(i)=abs((fas2(i)-fasrr(i))*100/fasrr(i));
emod(i)=abs((mod(i)-modrr(i))*100/modrr(i));
efas(i)=abs((fas(i)-fasrr(i))*100/fasrr(i));
end

plot(frec,efas8,frec,efas4,frec,efas2,frec,efas);
```

```
Y=abs(fft(perturcreada.values));
Nperiodos=1;
N=1000;
THD2=0;
for i=1:(N/2);
if i<>(Nperiodos+1);
THD2=Y(i)^2+THD2;
end
end
THD=sqrt(THD2)/Y(Nperiodos+1);
```

PRBS

```
clear();
PRBS.values=0;
y=[0,0,0,0,0,0,0,0,1];
N=511;
for i=1:N
      if y(9)==1
            salida=0.01;
      else
            salida=-0.01;
      end
      PRBS9(i)=salida;
      if y(5)==y(9)
         entrada=0;
      else
         entrada=1;
      end
           for j=1:8
             u(j)=y(j);
           end
           for j=1:8
             y(j+1)=u(j);
           end
      y(1)= entrada;
 end
```

```
pi=3.14159265359;
N1=127;
N2=255;
N3=511;
N4=1023;
N5=2047;
res7=10^5/N1;
res8=10^5/N2;
res9=10^5/N3;
res10=10^5/N4;
res11=10^5/N5;
for i=1:N2
y(i)=VO.values(i+3*N2);
u(i)=PRBS.values(i+3*N2);
end

for i=1:N2
      y(N2+i)=y(i);
      u(N2+i)=u(i);
```

```
end
muestra1=0;
for i=1:N2
   Ruu(i)=0;
   muestra1(i)=i;
   for j=1:(N2)
        Ruu(i)=Ruu(i)+u(j)*u(j+i);
   end
   Ruu(i)=Ruu(i)/N2;
end
RuuPRBS=Ruu;

for i=1:N2
   Ruy(i)=0;
   for j=1:N2
        Ruy(i)=Ruy(i)+u(j)*y(j+i-1);
   end
   Ruy(i)=Ruy(i)/N2;
end
RuyPRBS=Ruy;

Ruy8=Ruy/Ruu(N2);
Y8=(fft(Ruy8));

pi=3.14159265359;
VG=5;
L=68E-06;
C=220E-06;
RL=0.098;
RC=0.08;
R0=2.5;
IL0=0;
n1=RC*C*VG;
n0=VG;
d2=L*C*(1+RC/R0);
d1=L/R0+(RL*R0 + RC*R0+ RC*RL)*C/R0;
d0=1+ RL/R0;
N=10000;
res=10^5/N;
for i=1:N
f(i)=res*i;
f2(i)=log10(f(i));
x(i)=2*pi*f(i)*sqrt(-1);
num(i)=n1*x(i)+n0;
den(i)=d2*x(i)^2+d1*x(i)+d0;
G(i)=num(i)/den(i);
modreal(i)=20*log10(abs(G(i)));
fasereal(i)=(180/pi)*(atan(imag(G(i)),real(G(i))));
```

```
end

for i=2:((127+1)/2)
modY7(i)=20*log10(abs(Y7(i)));
fasY7(i)=(180/pi)*atan(imag(Y7(i)),real(Y7(i)));
frec7(i)=log10(res7*(i-1));
end

for i=2:((255+1)/2)
modY8(i)=20*log10(abs(Y8(i)));
fasY8(i)=(180/pi)*atan(imag(Y8(i)),real(Y8(i)));
frec8(i)=log10(res8*(i-1));
end

for i=2:((511+1)/2)
modY9(i)=20*log10(abs(Y9(i)));
fasY9(i)=(180/pi)*atan(imag(Y9(i)),real(Y9(i)));
frec9(i)=log10(res9*(i-1));
end

for i=2:((1023+1)/2)
modY10(i)=20*log10(abs(Y10(i)));
fasY10(i)=(180/pi)*atan(imag(Y10(i)),real(Y10(i)));
frec10(i)=log10(res10*(i-1));
end

for i=2:((2047+1)/2)
modY11(i)=20*log10(abs(Y11(i)));
fasY11(i)=(180/pi)*atan(imag(Y11(i)),real(Y11(i)));
frec11(i)=log10(res11*(i-1));
end

for i=1:(((127+1)/2)-1)
mod2Y7(i)=modY7(i+1);
fas2Y7(i)=fasY7(i+1);
frec27(i)= frec7(i+1);
end

for i=1:(((255+1)/2)-1)
mod2Y8(i)=modY8(i+1);
fas2Y8(i)=fasY8(i+1);
frec28(i)= frec8(i+1);
end

for i=1:(((511+1)/2)-1)
mod2Y9(i)=modY9(i+1);
fas2Y9(i)=fasY9(i+1);
frec29(i)= frec9(i+1);
```

```
end

for i=1:(((1023+1)/2)-1)
mod2Y10(i)=modY10(i+1);
fas2Y10(i)=fasY10(i+1);
frec210(i)= frec10(i+1);
end

for i=1:(((2047+1)/2)-1)
mod2Y11(i)=modY11(i+1);
fas2Y11(i)=fasY11(i+1);
frec211(i)= frec11(i+1);
end

plot(frec27,fas2Y7,frec28,fas2Y8,frec29,fas2Y9,frec210,fas2Y10,frec211,fas2Y11,f2,fasereal)
```

```
N7=127;
N8=255;
N9=511;
N10=1023;
N11=2047;
res7=fs/N7;
res8=fs/N8;
res9=fs/N9;
res10=fs/N10;
res11=fs/N11;

for i=1:(((127+1)/2)-1)
x7(i)=2*pi*res7*i*sqrt(-1);
num7(i)=n1*x7(i)+n0;
den7(i)=d2*x7(i)^2+d1*x7(i)+d0;
G7(i)=num7(i)/den7(i);
modreal7(i)=abs(G7(i));
fasereal7(i)=(180/pi)*(atan(imag(G7(i)),real(G7(i))));
emod7(i)=abs((10^(mod2Y7(i)/20)-modreal7(i))*100/modreal7(i));
efas7(i)=abs((fas2Y7(i)-fasereal7(i))*100/fasereal7(i));
end

for i=1:(((255+1)/2)-1)
x8(i)=2*pi*res8*i*sqrt(-1);
num8(i)=n1*x8(i)+n0;
den8(i)=d2*x8(i)^2+d1*x8(i)+d0;
G8(i)=num8(i)/den8(i);
modreal8(i)=abs(G8(i));
fasereal8(i)=(180/pi)*(atan(imag(G8(i)),real(G8(i))));
emod8(i)=abs((10^(mod2Y8(i)/20)-modreal8(i))*100/modreal8(i));
```

```
efas8(i)=abs((fas2Y8(i)-fasereal8(i))*100/fasereal8(i));
end

for i=1:(((511+1)/2)-1)
x9(i)=2*pi*res9*i*sqrt(-1);
num9(i)=n1*x9(i)+n0;
den9(i)=d2*x9(i)^2+d1*x9(i)+d0;
G9(i)=num9(i)/den9(i);
modreal9(i)=abs(G9(i));
fasereal9(i)=(180/pi)*(atan(imag(G9(i)),real(G9(i))));
emod9(i)=abs((10^(mod2Y9(i)/20)-modreal9(i))*100/modreal9(i));
efas9(i)=abs((fas2Y9(i)-fasereal9(i))*100/fasereal9(i));
end

for i=1:(((1023+1)/2)-1)
x10(i)=2*pi*res10*i*sqrt(-1);
num10(i)=n1*x10(i)+n0;
den10(i)=d2*x10(i)^2+d1*x10(i)+d0;
G10(i)=num10(i)/den10(i);
modreal10(i)=abs(G10(i));
fasereal10(i)=(180/pi)*(atan(imag(G10(i)),real(G10(i))));
emod10(i)=abs((10^(mod2Y10(i)/20)-modreal10(i))*100/modreal10(i));
efas10(i)=abs((fas2Y10(i)-fasereal10(i))*100/fasereal10(i));
end

for i=1:(((2047+1)/2)-1)
x11(i)=2*pi*res11*i*sqrt(-1);
num11(i)=n1*x11(i)+n0;
den11(i)=d2*x11(i)^2+d1*x11(i)+d0;
G11(i)=num11(i)/den11(i);
modreal11(i)=abs(G11(i));
fasereal11(i)=(180/pi)*(atan(imag(G11(i)),real(G11(i))));
emod11(i)=abs((10^(mod2Y11(i)/20)-modreal11(i))*100/modreal11(i));
efas11(i)=abs((fas2Y11(i)-fasereal11(i))*100/fasereal11(i));
end

plot(frec27,emod7,frec28,emod8,frec29,emod9,frec210,emod10,frec211,emod11)


--------------------------------------------------------------------------------
```