6 PRUEBAS REALIZADAS AL MONTAJE DSP

6.1 Seguidor Onda Ref

Como primer objetivo, una vez realizado los análisis eléctricos del montaje, es el chequear el funcionamiento del DSP con el resto de los circuitos. Para ello se utilizará un programa cuya función es la reproducción en la salida del inversor de una señal introducida en la entrada del adaptador de señal (señal de referencia).

En esta fase vamos a utilizar el programa de control en su versión de control no monitorizado. Una de las diversas opciones que nos ofrece el Code Composer es la monitorización de la ejecución del programa para ver como se va ejecutando las instrucciones en tiempo real. Este procedimiento esta bien para ir comprendiendo el funcionamiento interno del DSP e ir tomando contacto el microprocesador. Pero esto presenta varios inconvenientes importantes que son importantes conocer. El principal es que este procedimiento ralentiza la ejecución de las instrucciones por lo que los valores que se pueden observar en el osciloscopio no son realistas, introduciendo diferentes errores que pueden inducir a una mala interpretación de los datos representados. Otro inconveniente es el procedimiento de arranque del monitor el cual no siempre se consigue satisfactoriamente, teniendo incluso que proceder al reseteo del ordenador para que el monitor se ejecute correctamente. Por lo tanto, se procederá a la ejecución de los diversos programas utilizados en este proyecto sin utilizar el monitor pues por medio del osciloscopio se puede monitorizar perfectamente la ejecución de los programas. El programa monitor sólo es realmente útil para observar datos que no son posibles obtener por el osciloscopio, pero teniendo en cuenta el efecto anterior.

6.2 Elementos utilizados para el funcionamiento

Aquí utilizamos, además del DSP, una serie de dispositivos que son esenciales para la realización del experimento. Estos dispositivos son:

- Generador de funciones (suministrará la señal de entrada al adaptador de señales, canal de referencia).
- Fuentes de tensión
  - Alimentación del circuito del lazo, 24V.
  - Alimentación del circuito del driver, 15V.
  - Alimentación del circuito del adaptador de señales, fuente simétrica de $\pm 15\mathrm{V}$.
- Osciloscopio Digital bicanal.
- PC donde ejecutamos el programa en el DSP por medio del Code Composer, como también ejecutamos el programa de captura de señales del osciloscopio, el WaveStar.

6.3 Comenzando: Procedimiento de arranque del DSP

Antes de ejecutar cualquier programa en DSP debemos seguir una serie de pasos para que no tener “efectos indeseados” durante el funcionamiento del montaje. Los pasos a seguir son los descritos a continuación:

Debemos asegurarnos primeramente que tipo de señales vamos a capturar por el osciloscopio, pues éste debe de estar con la tierra en flotante en algunas medidas (lecturas de señal en la resistencia shunt del lazo). En tal caso, para la alimentación del osciloscopio se procederá a la instalación de un transformador que realizará la tarea de aislamiento galvánico (puede estar así de forma indefinida) y tendremos especial cuidado con el cable del puerto serie que concepta el osciloscopio con el PC pues éste conecta a tierra el osciloscopio. Aquí si queremos capturar señales de la resistencia shunt del lazo deberemos desconectar el cable serie, proceder a la captura de la señal por medio del botón activar/parar del osciloscopio, desconectar la sonda de lectura y proceder al volcado de la señal por medio del programa WaveStar. Si no se sigue fielmente este procedimiento se cortocircuitará el inversor a través del osciloscopio y fundirá el fusible que conecta el inversor con el driver.

Antes de ejecutar el Code Composer deberemos conectar el DSP a la alimentación, pues si no está conectado nos dará un error en el momento de traspasar el programa al DSP. Si el procedimiento no se hace correctamente, el DSP no conectaría y tendríamos en el peor de los casos reiniciar el PC.

Una vez conectado el Dsp y haber ejecutado el Code Composer, el siguiente paso es cargar el proyecto que estamos realizando por medio de la carpeta **Proyect -> Open** de menú del programa. Paso después nos iríamos a la carpeta de menú **File -> Load Program** donde aparecerá una ventana donde buscaremos nuestro programa caso. Si este no existe tendríamos que crearlo por medio del comando del menú **Proyect -> Rebuild All.**

Una vez cargado el programa en el Dsp procedemos al encendido del los dispositivos de alimentación del sistema. Primeramente encenderemos el generador de señales y haremos una lectura de la tensión de pico, ¿por qué?; uno de los problemas que tenemos en las lecturas es sobrepasar la tensión de protección, marcada por los diodos, pasando de una señal senoidal a una señal cuadrada. Este tipo de señal hace aumentar la intensidad del lazo produciendo la rotura de la resistencia por sobrecalentamiento (esta solo puede disipar una potencia determinada, ¡cuidado!). Por tanto intentaremos no sobrepasar la tensión que haría actuar los diodos y produciría estos efectos indeseados. Tomaremos una tensión de 1V de pico a pico como nuestra tensión de referencia.

A continuación encenderemos la fuente de tensión que alimenta el adaptador de señales, comprobando que la tensión de referencia sigue siendo de un voltio pico a pico. Es normal que al alimentar este circuito tengamos que variar la tensión del generador para adecuarla a la señal de referencia que

queremos colocar. Posteriormente la fuente que alimenta el driver y por último la fuente que alimenta el lazo. Para el lazo tendremos la precaución de no ponerlo a 24 V, sino a menos, porque un error de la ejecución de las instrucciones del DSP podría producir un fallo eléctrico en el circuito y provocar la rotura de algún componente.

Después de conectar todas las alimentaciones, nos dirigimos a la barra del menú **Debug->Reset DSP** (Code Composer) con el objetivo de resetear todos los registros del DSP a su estado original. Una vez reseteado volvemos a la barra del menú **Debug -> Go main** para actualizar todos los registros según los parámetros introducidos en el mismo, dejando el programa preparado para la ejecución. Durante esta etapa podemos observar si los registros han sido actualizados o no.

Ejecución del programa por medio de Run, de la misma barra del menú, o por medio del icono correspondiente que está situado a la derecha.

6.4 Captura y análisis de las señales

Una vez ejecutado el programa podremos ver en el osciloscopio las diferentes formas de onda en función del punto en que hayamos conectado la sonda.

Las formas de onda que se analizará con el osciloscopio principalmente son:

- La señal de referencia, introducida por el generador de señal.
- La señal de salida que encontramos en el primer bloque de amplificadores operacionales en el circuito de adaptación de señal del lazo. Esta señal está desfasada 180 debido a las características del circuito de amplificación. La señal leída, de voltaje, en la resistencia es la utilizada para poder medir la intensidad que circula por el lazo. Esta está en fase con la intensidad.

Procedemos a la captura de las señales por medio del osciloscopio configurando adecuadamente nuestro programa.

Para la realización de las medidas hemos configurado el margen de error en 100, siendo este valor un parámetro del rizado. Esto es parte del programa.

Las ecuaciones que trataran los datos de captura son:

$$i_0 = (ADC\_result - 565) \cdot 12 \quad (1)$$

$$ref = (ADC1\_result - 555) \cdot 49 \quad (2)$$

$$err = (i_0 - ref) \quad (3)$$

donde (1) lo utilizamos para adaptar la señal del lazo al DSP, (2) es nuestra señal de referencia y (3) el resultado de la diferencia de (1) y (2), el margen de error obtenido.

Aunque en teoría debería de ser restados los datos del resultado de la conversión por 512, la línea de origen ó 0, hemos observado la existencia de un offset en las lecturas por lo que hay que realizar una calibración para obtener unas lecturas lo más acordes con la realidad. Esta calibración se ha realizado introduciendo una señal de referencia, viendo la salida del conversor por medio del circuito del lazo y buscando el valor más acorde para que la salida no aparezca el offset. En este offset interviene en cierta medida los diferentes circuitos que consta el conjunto de la placa de adaptación de señales. Los multiplicadores son para poder operar ondas de la misma amplitud. O sea que 49/12 es aproximadamente 4, es decir, la amplitud de la intensidad del lazo es 4 veces superior a la referencia.

El objetivo de esta fase de pruebas de la placa es la búsqueda de los límites de trabajo del conjunto de dispositivos montados. Como se observará a continuación, estos límites dependerán de diversos factores internos del programa que “corre” en el DSP y externos a él (circuitos). En el interno, dependiendo del programa, pueden aparecer nuevos parámetros que hagan aumentar los factores que influyan en las medidas. En el externo, en función de los dispositivos utilizados como también en cambios en las características de los mismos (cambio de resistencias, ganancias…), pueden también variar los factores que influyen en las señales de trabajo. Por tanto, este estudio solo se puede utilizar como aproximado, eso si, teniendo en cuenta que se trabajará con los factores más característicos tanto externos como internos.

A continuación se describe los parámetros más relevantes que se van a someter a test, para realizar un análisis de los efectos en las señal de salida, la cual alimenta al lazo.

- Externos
  - Tensión de entrada
  - Tensión de alimentación del lazo
- Internos
  - Periodo de muestreo
  - Error

Antes de realizar medidas hay que tomar unos valores de referencia para los cambios posteriores de los mismos y poder realizar comparaciones.

- Externos
  - La tensión de entrada al adaptador de señal (referencia) es de 1.03 V aproximadamente.
  - La tensión de alimentación al circuito inversor es de 24 V aproximadamente
- Internos
  - El periodo de muestreo es de 100 (200 Khz)
  - El error es de 100

Una vez definido los parámetros y referencias solo queda concretar que tipo de señales se van a muestrear. Pues como se trabaja con un osciloscopio bicanal (dos entradas) se tomaran dos muestras que son:

- El canal 1 corresponde a la señal de referencia, que es ofrecida por el generador de señal. Se conectará la pinza negativa a la masa del generador y del montaje y la pinza positiva a la entrada del adaptador de señal, referencia positiva de esta.
- El canal 2 es la señal leída de la salida del amplificador diferencial del circuito de adaptación de señales, la correspondiente al circuito de lectura de la señal del lazo. Esta lectura es la más importante pues es la señal que aparecerá en el lazo. Para este experimento el lazo (conductores) es sustituido por una impedancia de L=0.083H por simplificar del montaje.

A partir de ahora se expondrá las diferentes capturas realizadas con el osciloscopio. En la primera medida se toma una captura de la señal de referencia (canal 1 en rojo) y de la salida del lazo (canal 2 en azul). Esta captura es la referida a los parámetros con los valores de referencia antes expuestos.

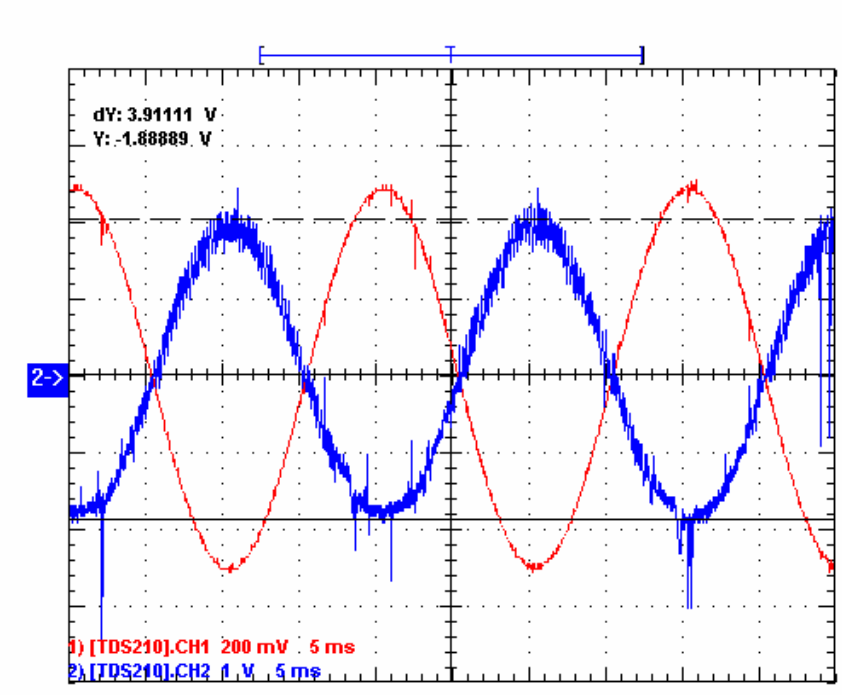

Figura 6.1. Onda 1

6.4.1 Parámetro muestreo

En este apartado se procede a variar el parámetro de muestreo, correspondiente al programa. Se estudiará la influencia de este valor en la señal de salida en el lazo. Por tanto el nuevo conjunto de valores son:

- la tensión de entrada es de 1.01 aprox.
- La tensión de alimentación de 24 V aproximadamente
- El periodo de muestreo es de **300 (66,7 Khz)**
- El error es de 100

Se mantiene los parámetros anteriores salvo el valor de muestreo que lo cambiamos el valor de 100 por 300, es decir, multiplicando el anterior por 3. Se ha comprobado que debido al ruido introducido en el osciloscopio producido por las conmutaciones, las lecturas en la tensión de referencia difieren unas décimas.

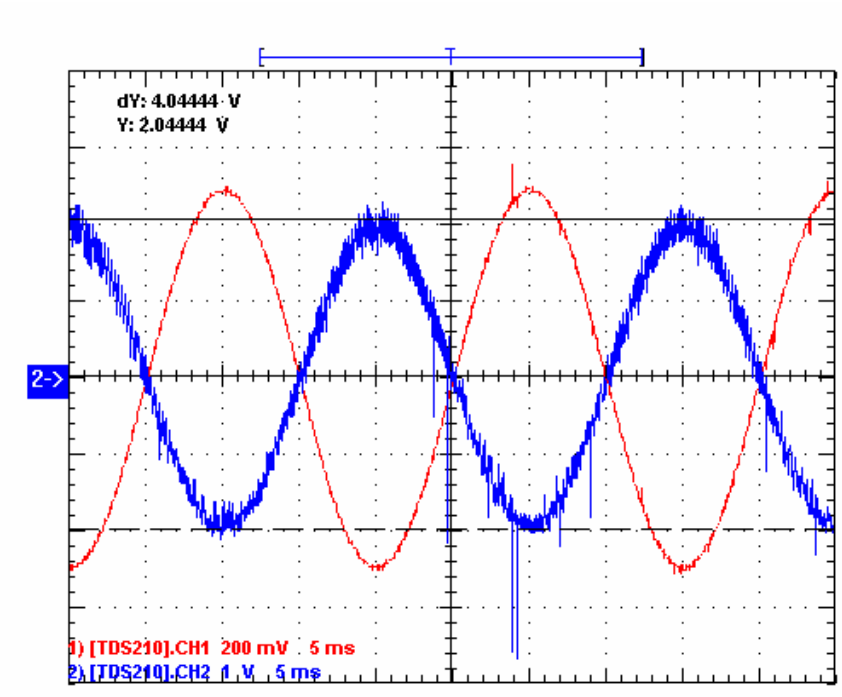

Figura 6.2. Onda 2

Según el programa, el periodo de muestreo es una función que depende de la frecuencia de trabajo del microprocesador DSP (su inverso), de los parámetros de configuración de los registros del programa (de valor 1 en este caso) y de una variable que es llamada PERIOD que será el parámetro que modificaremos. La relación es:

$$\text{Tiempo de muestreo} = 50\,\text{ns} \cdot 1 \cdot \text{PERIOD}$$

Para este caso en que PERIOD=300 se tiene:

$$\text{Tiempo de muestreo} = 50\,\text{ns} \cdot 1 \cdot 300 = 0{,}000015\,\text{s} \rightarrow 66{,}7\,\text{kHz}$$

A esta frecuencia de muestreo no se aprecia una disminución en la calidad de la señal.

Realizando un nuevo cambio se tiene un nuevo conjunto de valores:

- la tensión de entrada es de 1.01 aprox.
- La tensión de alimentación de 24 V aproximadamente
- El periodo de muestreo es de **500 (40 Khz)**
- El error es de 100

Se sigue manteniendo los parámetros anteriores salvo el valor de muestreo que lo cambiamos el valor de 300 por 500, o también si lo comparamos con el valor de referencia de 100 a 500.

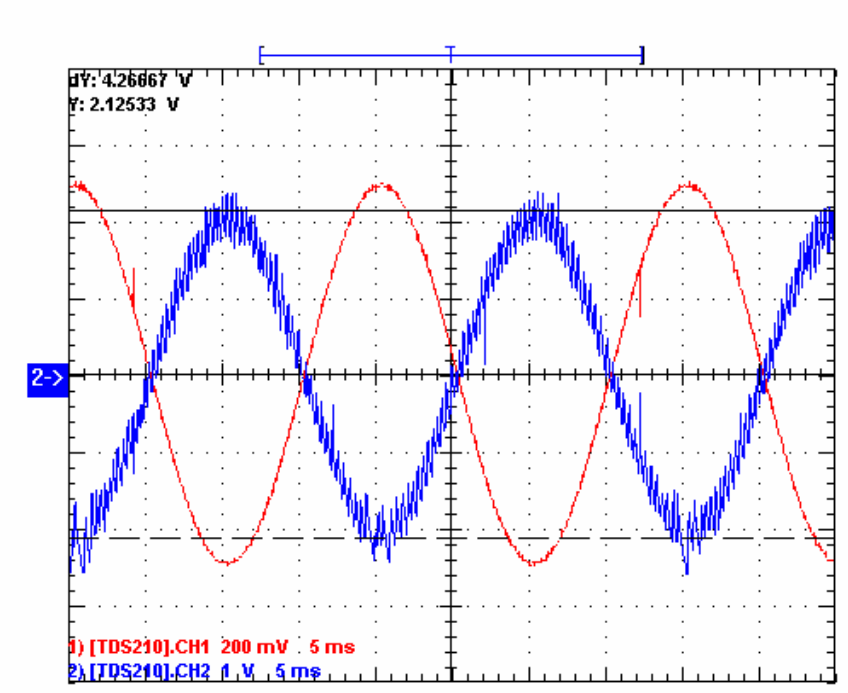

Figura 6.3. Onda 3

A diferencia de la onda 2, en la onda 3 existe una pequeña variación en la calidad de la señal de salida. Al disminuir la frecuencia de muestreo, los tiempos de los disparos de los transistores aumenta por lo que también aumenta el error respecto a la señal de referencia. Al final hay menos conmutaciones y esto disminuye la calidad de la forma de onda. Ahora bien, el haber disminuido la calidad no significa que ésta no sea aceptable si no que al disminuir esa frecuencia de muestreo el microprocesador puede realizar otras tareas también importantes. Además a menor número disparos menores pérdidas en las conmutaciones y la temperatura en los transistores disminuye.

Como referencia en las medidas, una lectura de la intensidad que circula por el lazo es muy intuitiva para observar como se comporta la forma de onda al tener una observación directa limpia (al no pasar por la etapa diferencial). En la figura onda 3.1 se muestra esta lectura.

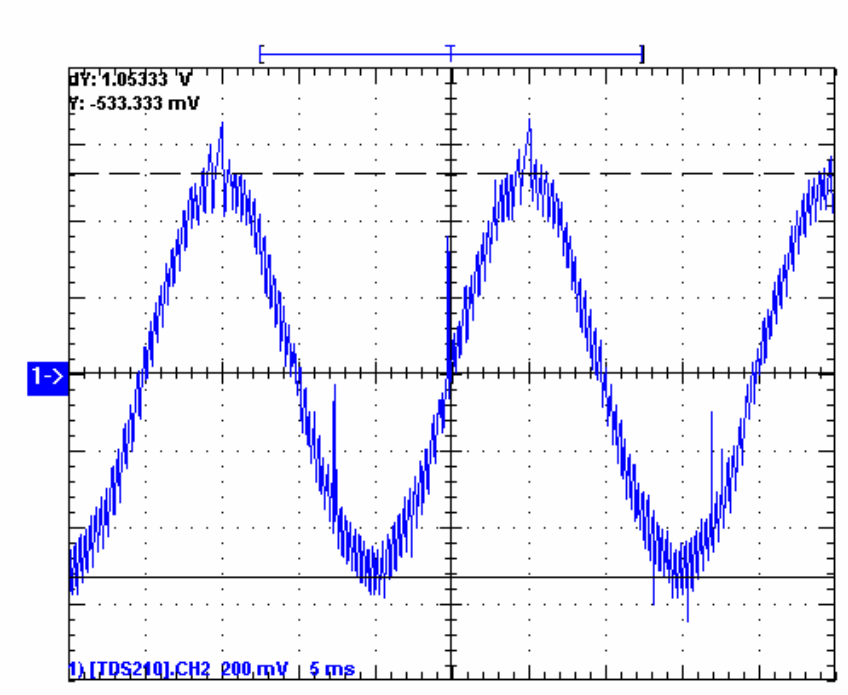

Figura 6.4. Onda 3.1

La onda 3.2 la lectura del canal 2 (el número que aparece a la izquierda, 1, indica que solo hay una sola forma de onda, no que sea el canal 1) está tomada directamente del lazo, la resistencia shunt del lazo. Para esta operación se tomaron las precauciones adecuadas para aislar completamente el osciloscopio de tierra, desconectando la sonda del canal uno que esta conectada a la masa del circuito. Se aprecia que la salida del amplificador diferencial es un múltiplo de 4.

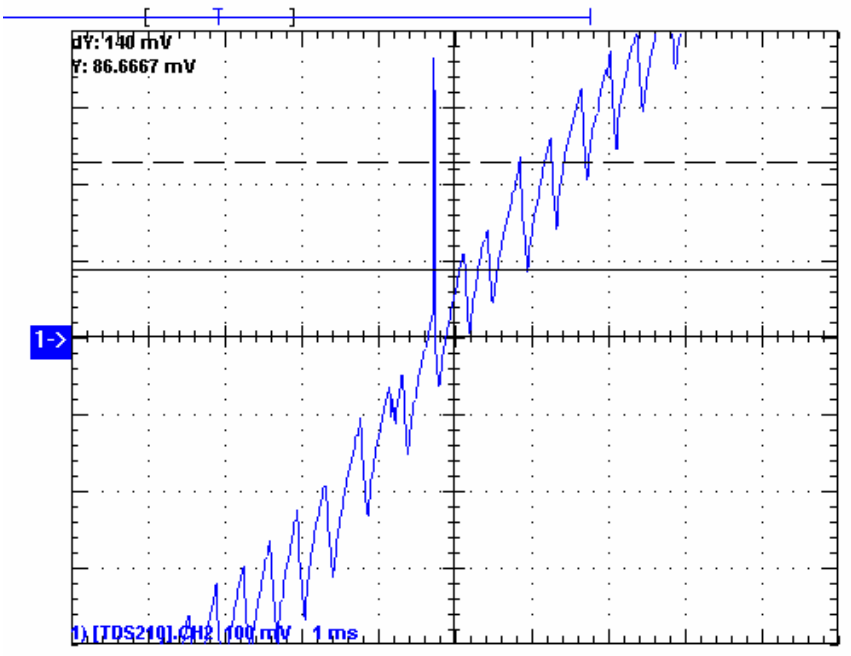

Figura 6.5. Onda 3.2

Ésta es una buena captura para observar de una forma muy nítida las conmutaciones de los transistores (dientes de sierra). En la figura onda 3.2, la tensión en la que se mueve el error es de 140mV de pico a pico, por lo que la tensión de error (o banda de conmutación) estará aproximadamente en 70mV. En la figura aparecen ciertas irregularidades que pueden deberse al sistema de muestreo del osciloscopio.

Con estas capturas hemos visto la relación entre el periodo de muestreo y como se relaciona con los disparos de los transistores de potencia de nuestro inversor. A la hora del diseño tendremos este dato en cuenta pues es importante definir un buen periodo de muestreo conjugando además que necesitamos unos tiempos de ejecución para nuestro programa para que estos tiempos no entorpezca su ejecución.

Se procede a realizar una última variación en el parámetro PERIOD. Se elige un valor muy superior para observar como varia la onda a una variación superior a un valor aceptable. Los valores utilizados son:

- la tensión de entrada es de 1.01 aprox.
- La tensión de alimentación de 24 V aproximadamente
- El periodo de muestreo es de **1000 (20 Khz)**
- El error es de 100

Aquí cambiaremos ahora a 1000 el periodo de muestreo y se mantiene el resto de valores como están.

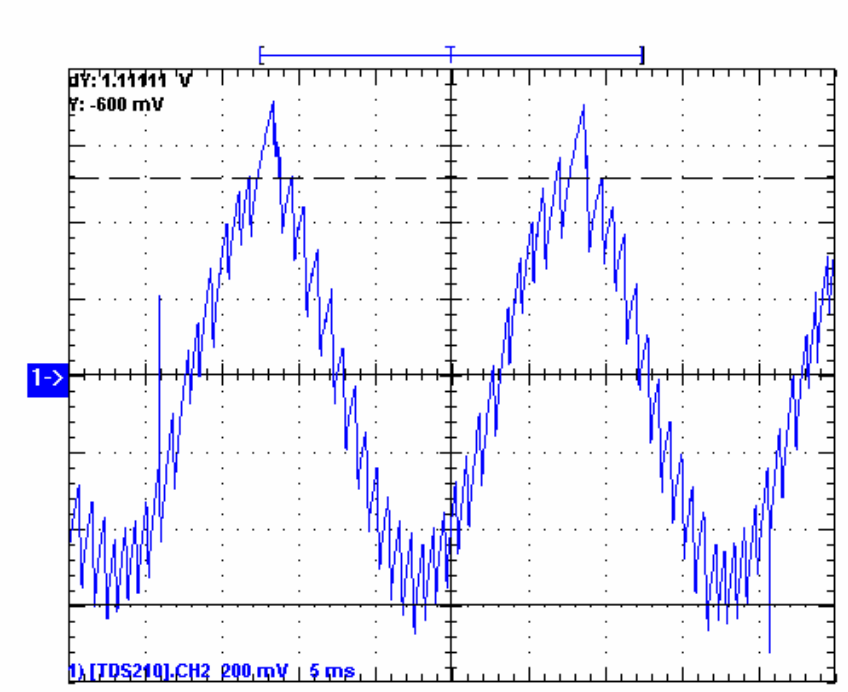

Figura 6.6. Onda 4

Aquí se puede ver con mayor precisión como la disminución del periodo de muestreo hace también disminuir las conmutaciones de los transistores del

inversor y por tanto aumenta la amplitud del rizado. Realizando un zoom sobre la onda se mide esa amplitud y se podrá leer ese valor para comparar con la medida de la figura onda 3.2

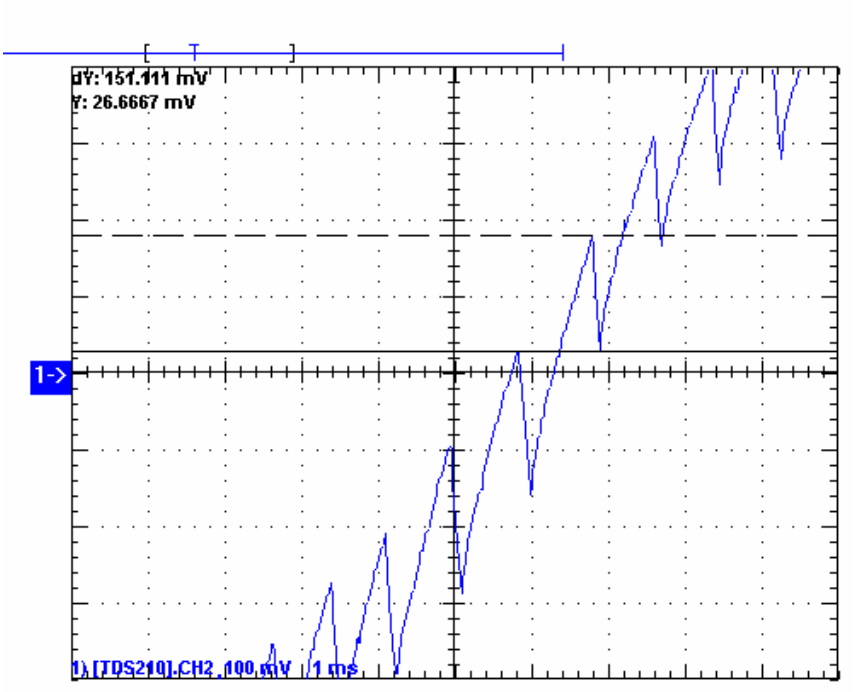

Figura 6.7. Onda 4.1

Obsérvese que respecto a la figura onda 3.2 la amplitud del rizado a aumentado ligeramente, pero la frecuencia con que conmuta a disminuido considerablemente pasando de 3 periodos en la figura onda 3.2 a un periodo en la figura 6.7 en un ms. La referencia de las medidas siempre sobre la resistencia shunt.

Una vez realizado este test, se elige el valor de 500 (40 Khz) para el parámetro del periodo de muestreo por ser un valor que cumple aceptablemente las características que necesitamos y además tiene un número de conmutaciones aceptables. Estas características pueden variar al cambiar los otros parámetros.

6.4.2 Parámetro error

Se variará ahora el parámetro de error. Se define el error a la diferencia entre la señal de referencia y la muestreada. Esta diferencia crea, en la señal de salida, una señal con dientes de sierra. Se procede a un continuo muestreo y comparación. Si supera el error definido conmuta los transistores, cambiando la pendiente de la señal de salida.

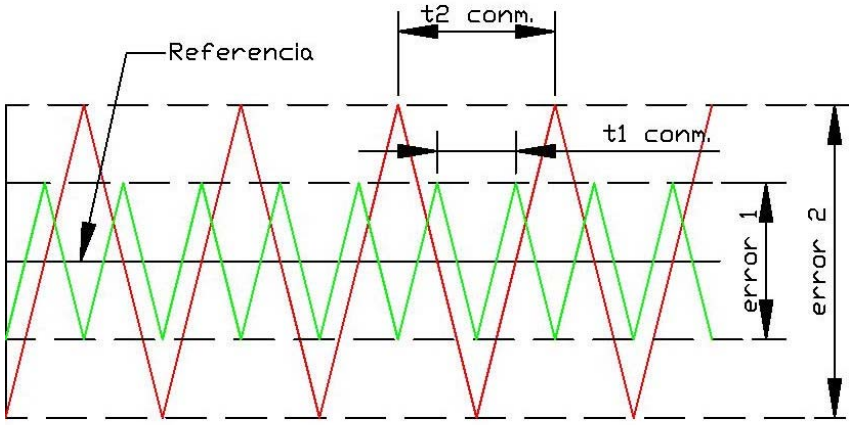

Figura 6.8. Conmutación de los transistores

El objetivo es ver sus diferentes formas de ondas y buscar la más adecuada en función de este error. A primera vista parece que ha menor error más precisión en la salida. Esto puede tener serias complicaciones, pues a menor error mayor número de conmutaciones y por tanto mayores perdidas por conmutación y calentamientos de los transistores. A bajas potencias apenas se notará, las perdidas, pero a medida que la intensidad vaya aumentando por el lazo este efecto se hará más evidente.

Los parámetros para este ensayo son:

- La tensión de entrada es de 1.01 aprox.
- La tensión de alimentación de 24 V aproximadamente.
- El periodo de muestreo es de 500.
- El error es de **50**.

En la figura que se expone a continuación no es más que una ampliación de la figura de la onda 5, en él se ha querido detallar con mayor precisión la forma de onda en función de las conmutaciones. Analizando los datos que nos aporta la forma de onda, a una escala de 1ms en el eje X y a 500mV en el eje Y.

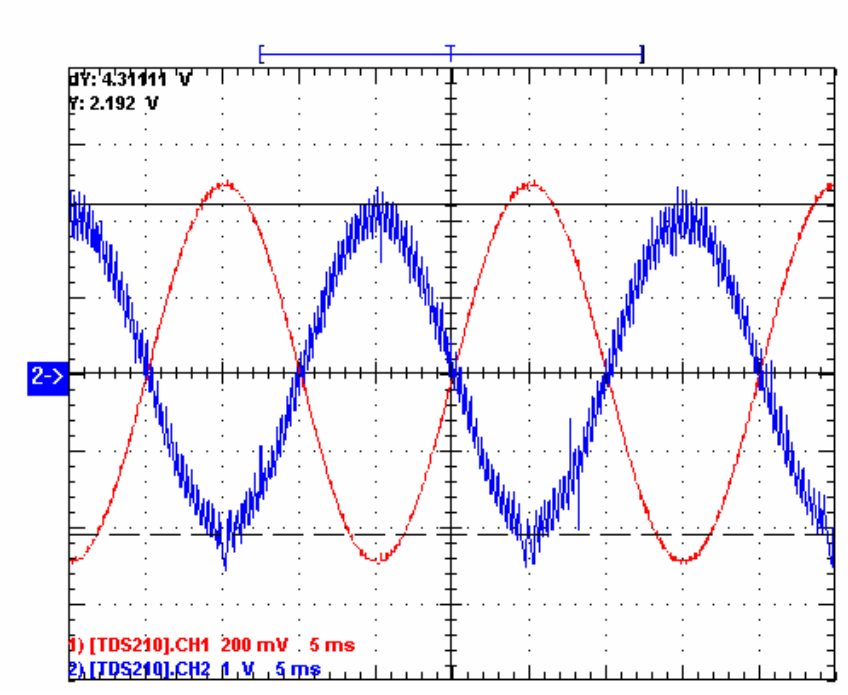

Figura 6.9. Onda 5

La lectura, amplificada por 4, es de 377mV por lo que sin amplificar es del orden de 94mV, que es la salida aproximada que se tendría en la resistencia shunt. Por tanto, según los datos ahora obtenidos, un margen de error menor da como resultado, claramente esperado, un mayor número de conmutaciones de los transistores del semipuente, al estrecharse la banda de error permitida. Esto significa una mejor definición de la señal de salida.

La principal dificultad en este tipo de medida, como se aprecia en las diferentes capturas, es la precisión de la señal, pues no es del todo uniforme presentando lecturas diferentes en función de la zona escogida en la onda. En la figura 3.1, escogiendo otro punto de lectura nos podemos encontrar lecturas del orden de 470mV por lo que ya presenta un error considerable respecto a la lectura anterior, invalidando de cierta manera el método. Sin embargo, un análisis visual, al menos, despeja ciertas dudas al ver si es homogénea la forma de onda representada y si es coherente las lecturas.

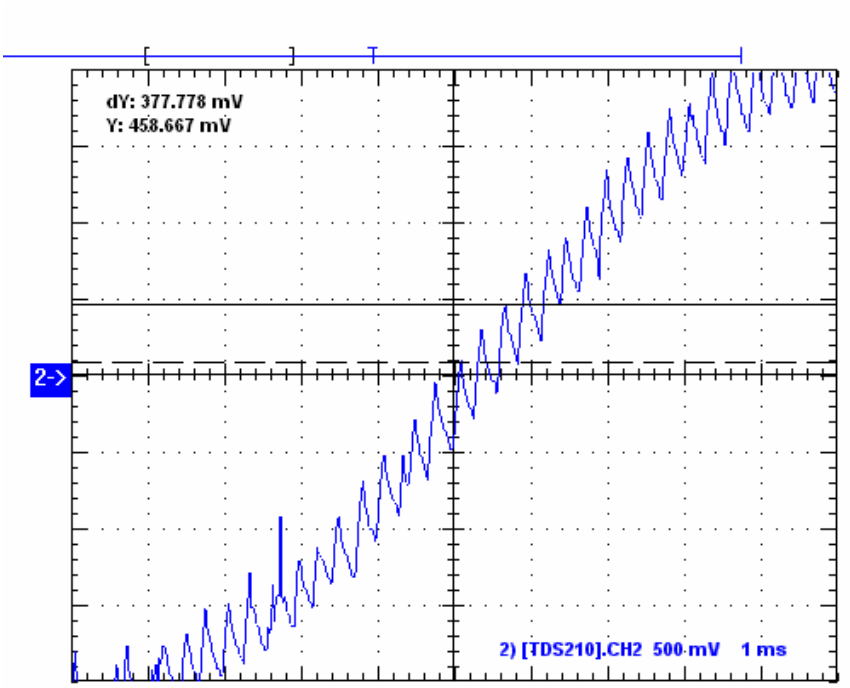

Figura 6.10. Onda 5.1

Si antes se ha experimentado con un error de 50, que es la mitad de 100, ahora se va a estudiar un error de 200, que es el doble de 100. En principio esta banda de error introduciría a que la onda presentará una mayor deformación al tener que oscilar en un rango mayor, sin embargo, con un error mayor disminuiría las conmutaciones. Viéndose gráficamente se puede sacar las conclusiones de una manera más intuitiva.

Los parámetros para el ensayo son:

- la tensión de entrada es de 1.01 aprox.
- La tensión de alimentación de 24 V aproximadamente.
- El periodo de muestreo es de 500.
- El error es de **200**.

Comparando la figura onda 5 con la figura de la onda 6 se aprecia la mejor definición de la señal de la onda 5, mayor número de conmutaciones, que la onda 6, menor número de ellas. Eso no deja, por tanto, de ser la onda 6 una buena señal, pues sigue de una manera fiel su referencia, eso si, menos precisa.

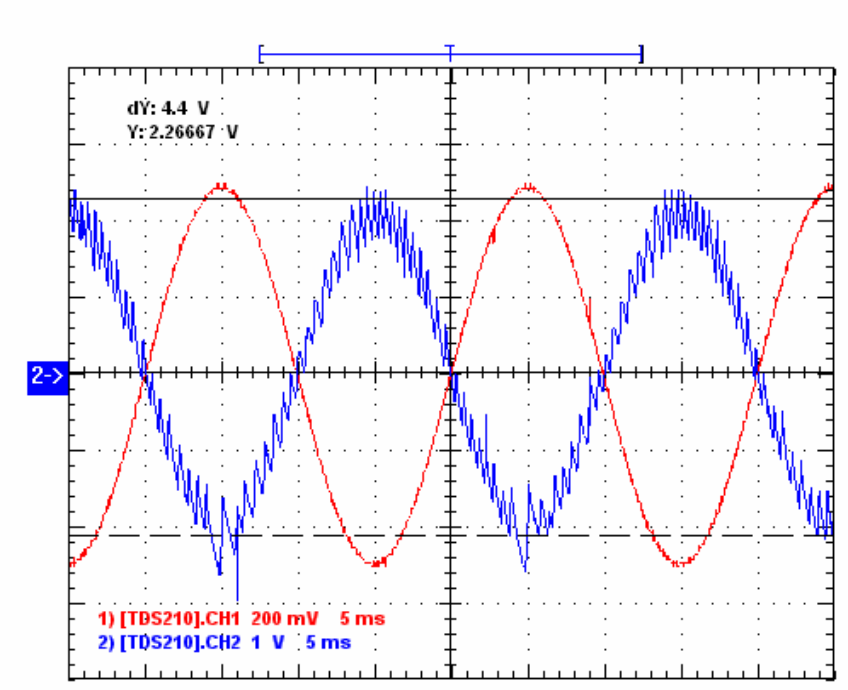

Figura 6.11. Onda 6

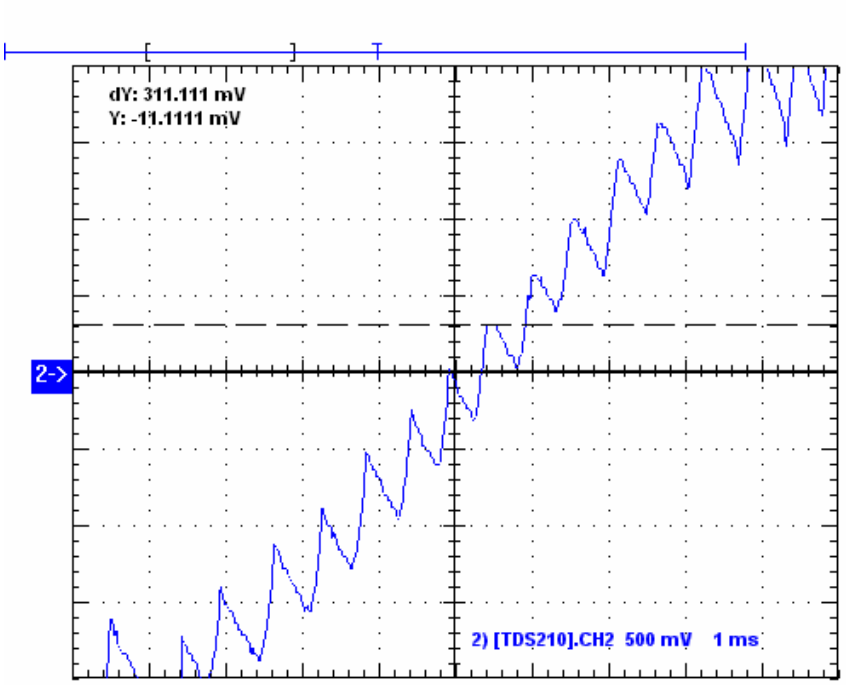

Figura 6.12. Onda 6.1

Aquí se puede observar, con mayor precisión y comparando con la onda 5.1, la disminución de las conmutaciones y el aumento de la banda de error, donde la lectura aquí obtenida es de 311mV, muy superior a las otras lecturas, tanto para un error de 100 como para un error de 50. Por tanto se demuestra una estrecha relación entre el parámetro de error, la banda de error en la señal y el número de conmutaciones obtenidas. Con estos datos hay que jugar para elegir aquella que mejor se adecue a la forma más deseada.

Variamos ahora de 200 a 400 el error para evidenciar como un incremento muy significativo en este parámetro introduce una forma de onda más deformada.

Los parámetros para el ensayo son:

- la tensión de entrada es de 1.01 aprox.
- La tensión de alimentación de 24 V aproximadamente.
- El periodo de muestreo es de 500.
- El error es de **400**.

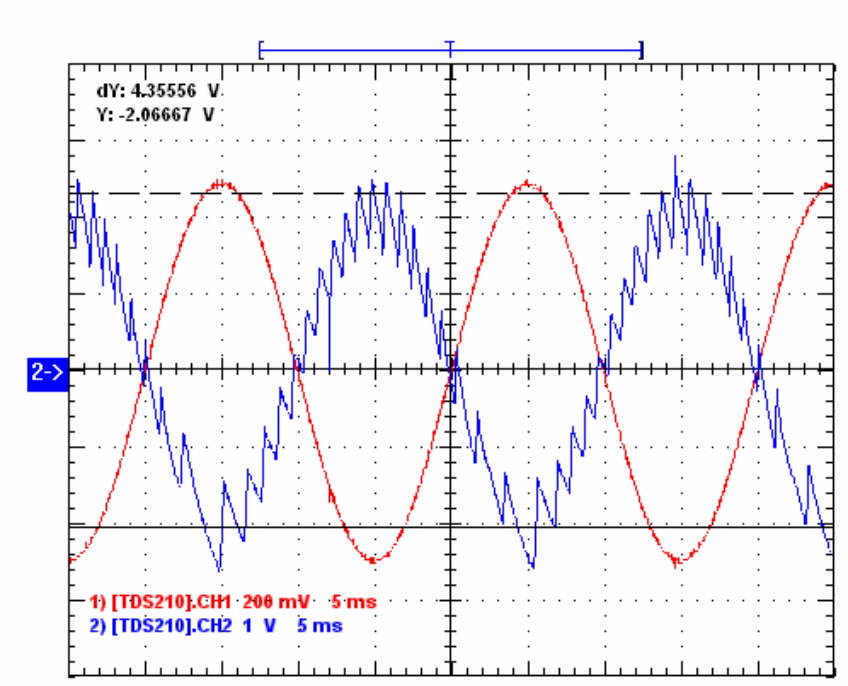

Figura 6.13. Onda 7

Es evidente, con la lectura que se observa, que ha disminuido de una manera muy significativa las conmutaciones y ha aumentado la banda del rizado.

Con esto se da por terminado las diversas capturas relacionadas con los parámetros del programa, que son independientes a las señales del exterior. Ahora procederemos a manipular dos de los parámetros externos al programa como son la tensión de alimentación y la tensión de la señal de referencia. Este parámetro, la tensión de la señal de referencia, será el más importante de todos.

Dejaremos como valores estables el periodo de muestreo de 500 y el error del seguimiento en 100.

6.4.3 Parámetro voltaje

En estas capturas procederemos a la variación de la tensión de alimentación desde 24 voltios, que va a ser nuestra referencia, hasta una tensión admisible para la generación de una señal aceptable por el inversor. Mantendremos la tensión de 1 V_{pp} para nuestro estudio, aunque en posteriores análisis se podría variar la tensión de alimentación y el voltaje de la señal de referencia.

Los parámetros para el ensayo son:

- la tensión de entrada es de 1.01 aprox.
- La tensión de alimentación de **20 V** aproximadamente.
- El periodo de muestreo es de 500.
- El error es de 100.

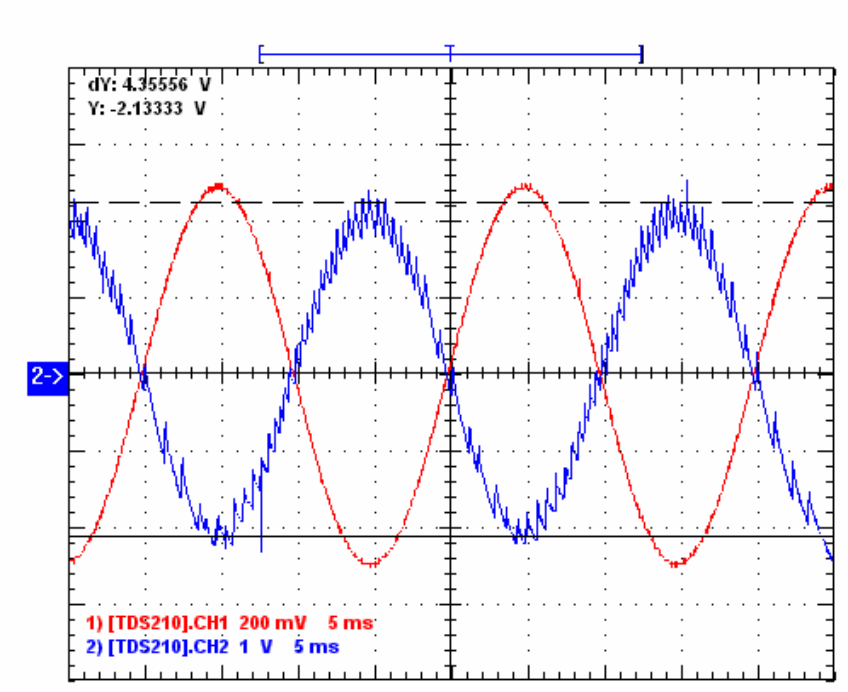

Figura 6.14. Onda 8

En esta figura se ve como en estas tensiones ya se aprecia que el número de conmutaciones disminuye de una forma significativa (comparada con la forma de referencia, onda 3). Además si analizamos las dos pendientes del seno, la positiva y la negativa, vemos que hay un número diferente de conmutaciones entre ambas, siendo superior en la pendiente positiva que en la negativa. Se reducirá la tensión y veremos si este efecto se acentúa o no.

Los parámetros para la siguiente prueba son:

- la tensión de entrada es de 1.01 aprox.
- La tensión de alimentación de **15 V** aproximadamente.
- El periodo de muestreo es de 500.

- El error es de 100.

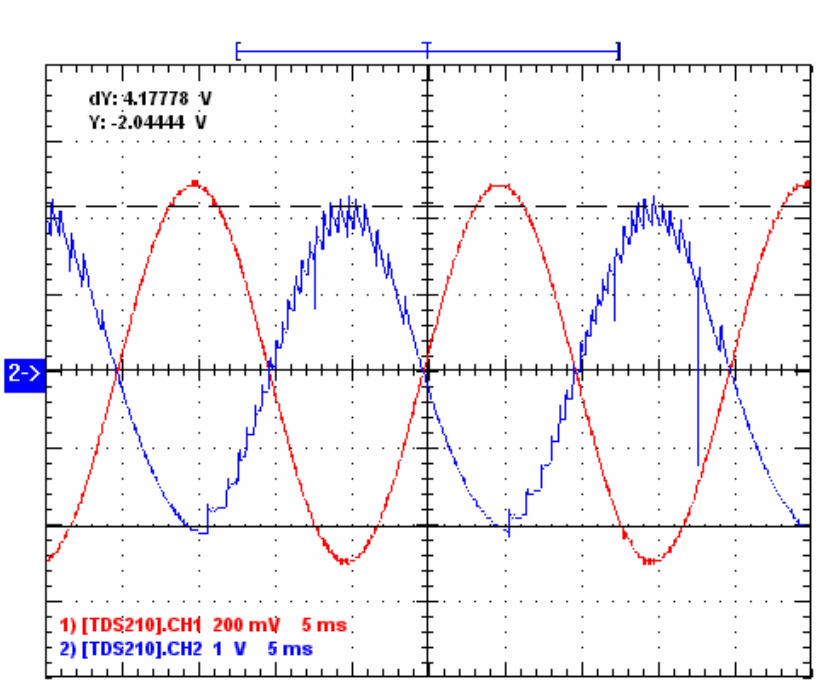

Figura 6.15. Onda 9

Con la bajada de tensión se acentúa el efecto anterior. Además el número de conmutaciones disminuye en ambas pendientes. Sin embargo la diferencia de potencial en la resistencia se mantiene casi constante, por lo que deduce que la intensidad que circula por el lazo la mantiene a pesar de la caída de tensión en la fuente, teniendo en la salida una señal fiable, por lo que deduce que el lazo puede ser alimentado por diferentes tensiones siempre que pueda aportar la intensidad que necesite el lazo.

Aquí vamos a buscar ahora la tensión mínima de trabajo, el cual el lazo debe de ser alimentado, para que la intensidad que circule por él sea la predeterminada por el DSP.

Los parámetros para la siguiente prueba son:

- la tensión de entrada es de 1.01 aprox.
- La tensión de alimentación de **¿? V** aproximadamente.
- El periodo de muestreo es de 500.
- El error es de 100.

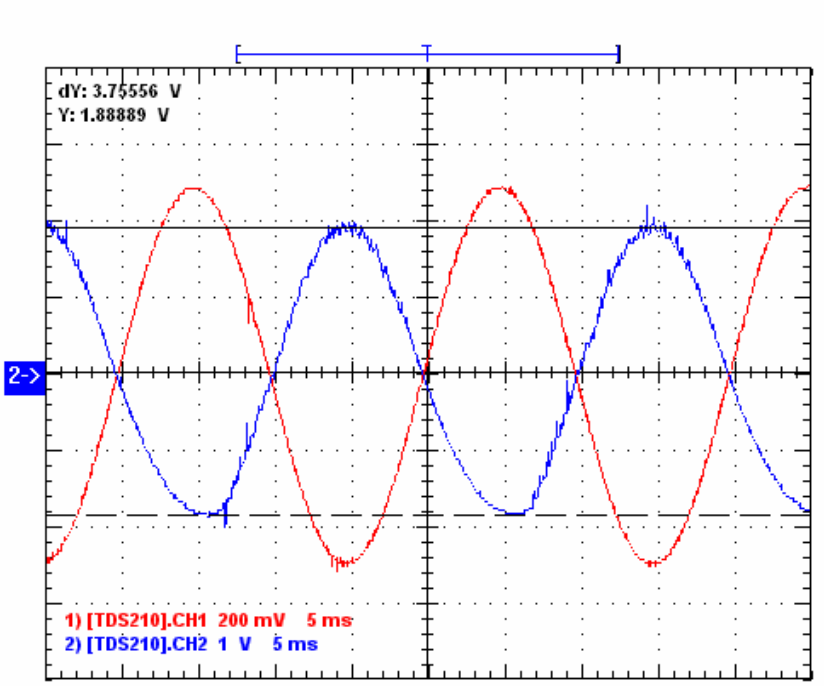

Figura 6.16. Onda 10

Para la tensión de la figura onda 10, la tensión que alimenta el lazo dada por la fuente es de 3.2 voltios. A partir de ese valor comienza a producirse las conmutaciones de los transistores para regular la formación de la onda senoidal y mantener la intensidad por el lazo. También se aprecia una pequeña bajada de tensión por la dificultad de mantener la intensidad en el lazo debido a la bajada de tensión.

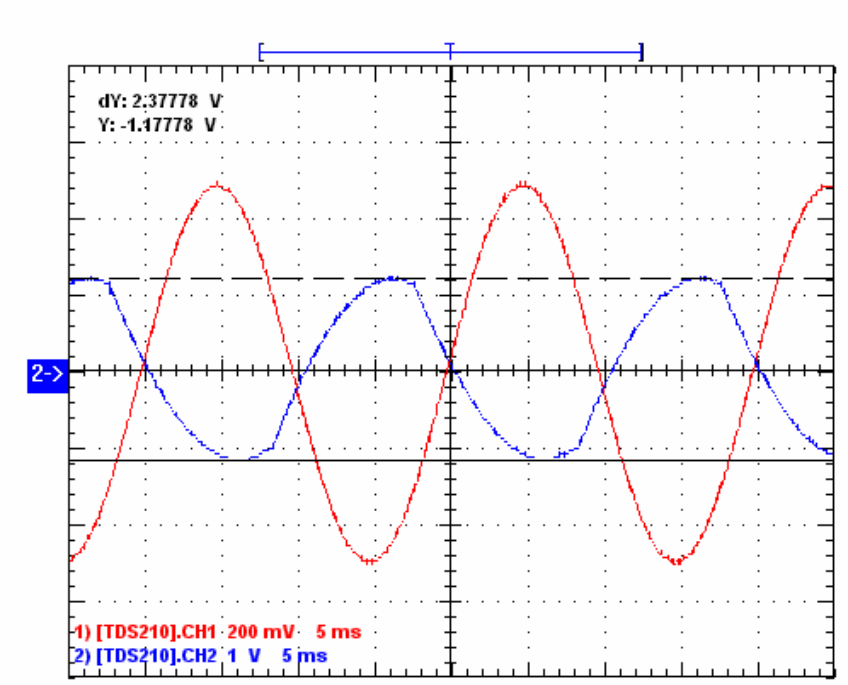

Figura 6.17. Onda 10.1

Bajando la tensión de alimentación se podrá apreciar como desaparecen las conmutaciones, viéndose además las curvas de carga y descarga de los condensadores

6.4.4 Parámetro referencia

A partir de aquí se mantiene la tensión de alimentación en 24 voltios y se procede a la variación de la tensión de referencia, en nuestro caso la parte más importante de nuestro estudio.

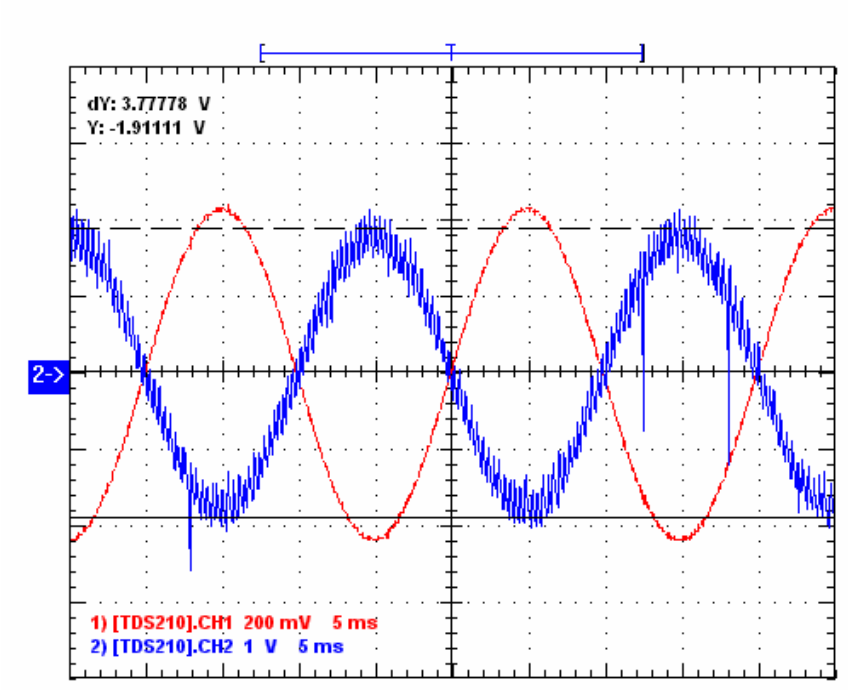

Figura 6.18. Onda 11

Los parámetros para la siguiente prueba son:

- la tensión de entrada es de **0.9 V_{pp}** aprox.
- La tensión de alimentación de 24 V aproximadamente.
- El periodo de muestreo es de 500.
- El error es de 100.

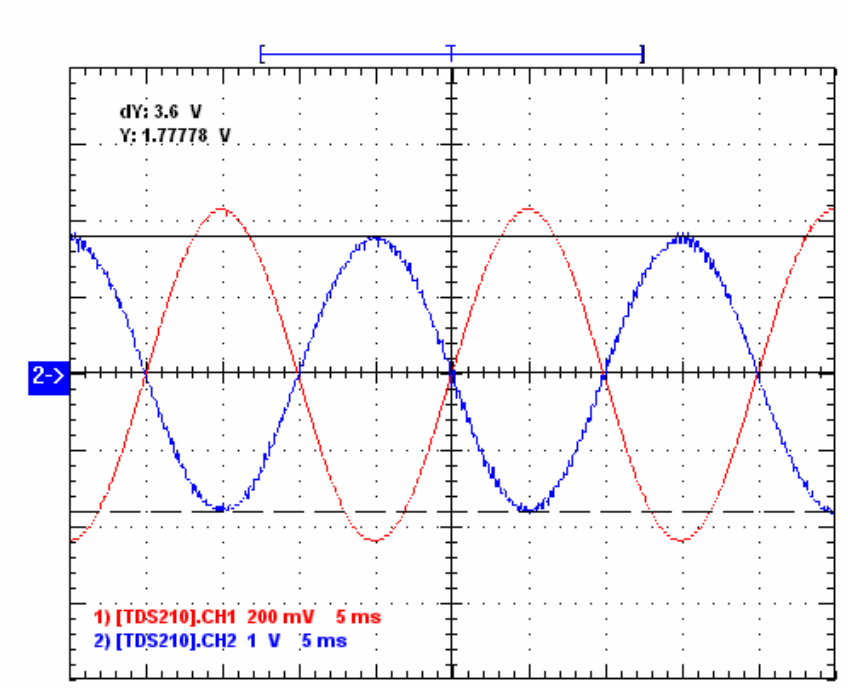

Figura 6.19. Onda 11.1

En la onda 11.1 se ha procedido a utilizar una función del osciloscopio como es el promediado de la señal. Este consiste en tomar diferentes señales, unas 16 en nuestro caso, y hacer una media entre ellas, obteniendo una señal más nítida y eliminando las conmutaciones. Esto nos servirá para observar mejor el rango de tensiones en que se mueve la señal obtenida del lazo. Si aplicamos el multiplicador por 4, la lectura de la señal de salida es de 3.6, o sea 4x0.9 voltios de la entrada.

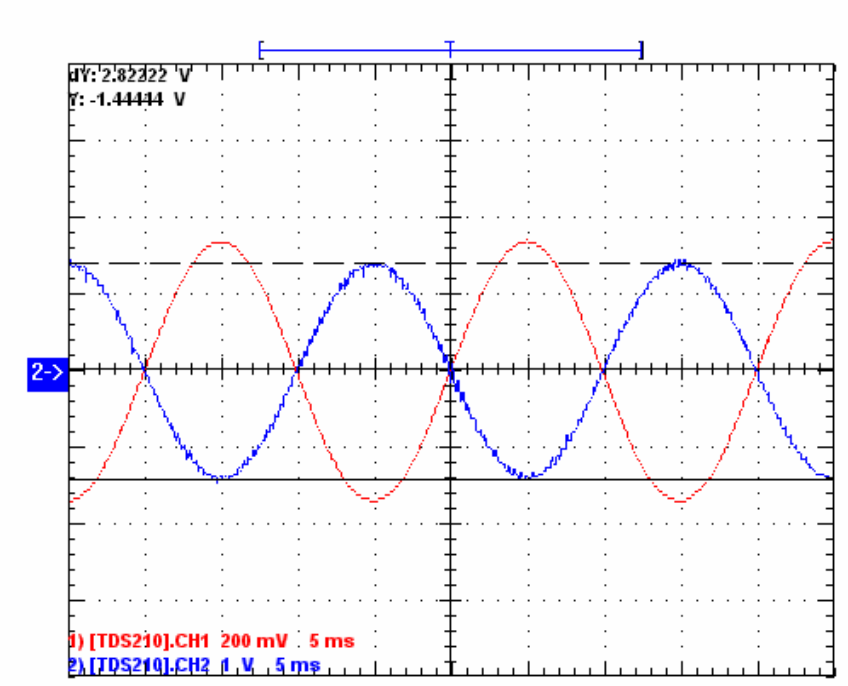

Figura 6.20. Onda 12

Los parámetros para la siguiente prueba son:

- la tensión de entrada es de **0.7 V_{pp}** aprox.
- La tensión de alimentación de 24 V aproximadamente.
- El periodo de muestreo es de 500.
- El error es de 100.

En la figura onda 12, promediada a 16, se observa claramente, como en la figura onda 11.1, como coincide la señal de salida con la tensión de entrada, aplicando únicamente el multiplicador por 4. Con estas observaciones se puede decir que sigue de manera fiel la tensión de referencia.

Se continúa variando la tensión buscando alguna anomalía que diferencia las lecturas de las actuales

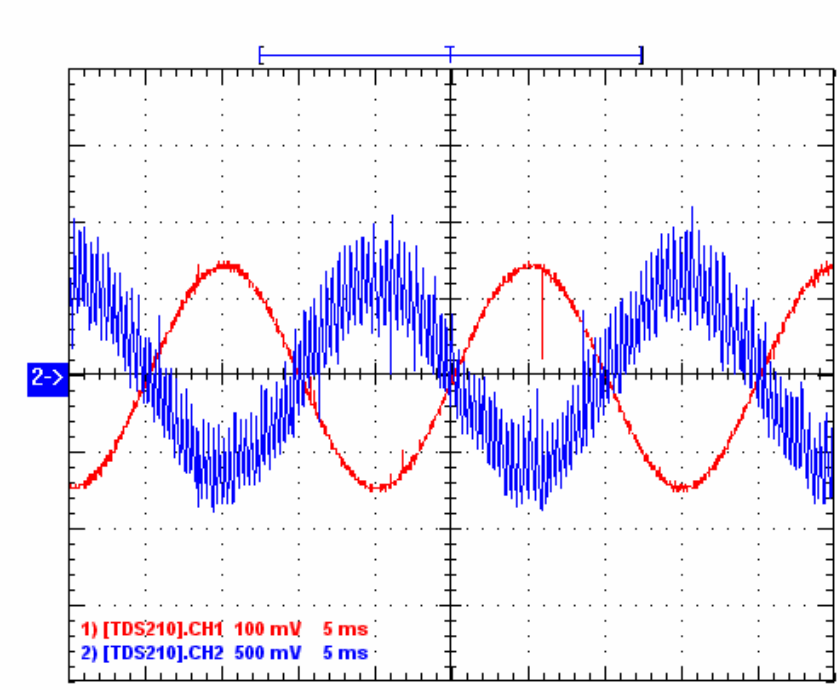

Figura 6.21. Onda 13

Los parámetros para la siguiente prueba son:

- la tensión de entrada es de **0.3 V_{pp}** aprox.
- La tensión de alimentación de 24 V aproximadamente.
- El periodo de muestreo es de 500.
- El error es de 100.

Aquí si se aprecia con mucha claridad el rizado de la señal. La disminución de la tensión de entrada produce un aumento del rizado de la señal de salida. Si se analiza más profundamente, al ser el error cte y por tanto la banda donde debe moverse el rizado, al disminuir la tensión y mantenerse esa banda, la proporción de tensión de rizado y tensión de entrada aumenta por lo cual este error, en proporción al tamaño, es mayor.

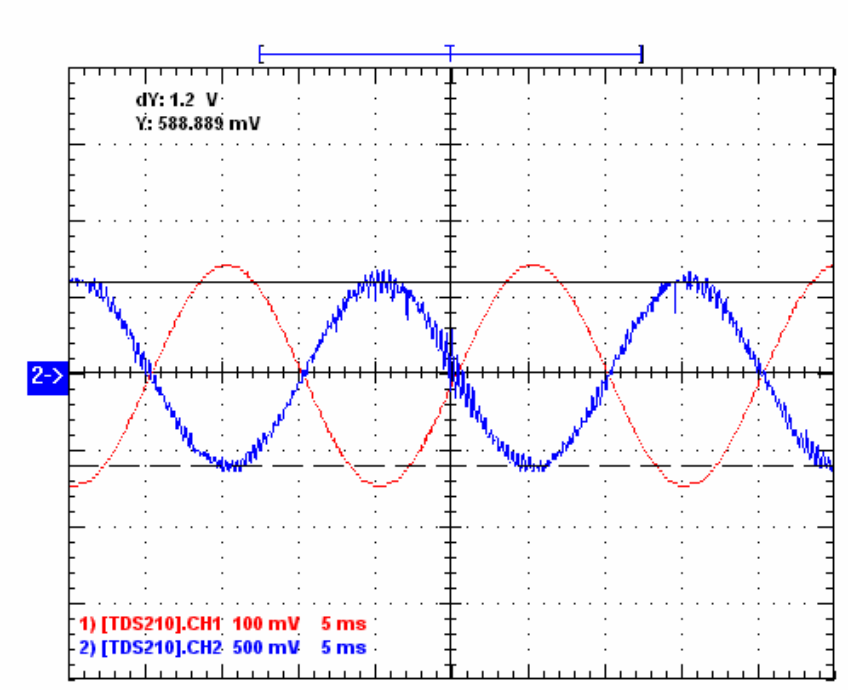

Figura 6.22. Onda 13.1

Aun así, el hecho que el rizado aumente, no tiene porque aumentar el error, pues al ser promediado la señal, aproximadamente se cumple la proporcionalidad entre señal de entrada y señal de salida.

Buscamos ahora la tensión mínima a partir del cual se comienza hacer evidente la señal

Los parámetros para la siguiente prueba son:

- la tensión de entrada es de **¿? V_{pp}** aprox.
- La tensión de alimentación de 24 V aproximadamente.
- El periodo de muestreo es de 500.
- El error es de 100.

La tensión para la cual se ha obtenido la representación de la figura onda 14 es de 200mV. Por debajo de esta tensión la señal deja de ser senoidal e incluso desaparece, por lo que dejarían de conmutar los transistores al no llegar una tensión lo suficientemente alta para excitarlos y producir la saturación del canal. La señal de la figura onda 14 está promediada en 16 lecturas.

Se ha procedido a realizar diversos cambios en el programa para poder mejorar los resultados en las conmutaciones de los transistores sin obtener resultados apreciables. Por tanto para estos voltajes es difícil encontrar parámetros en el programa que mejoren los resultados.

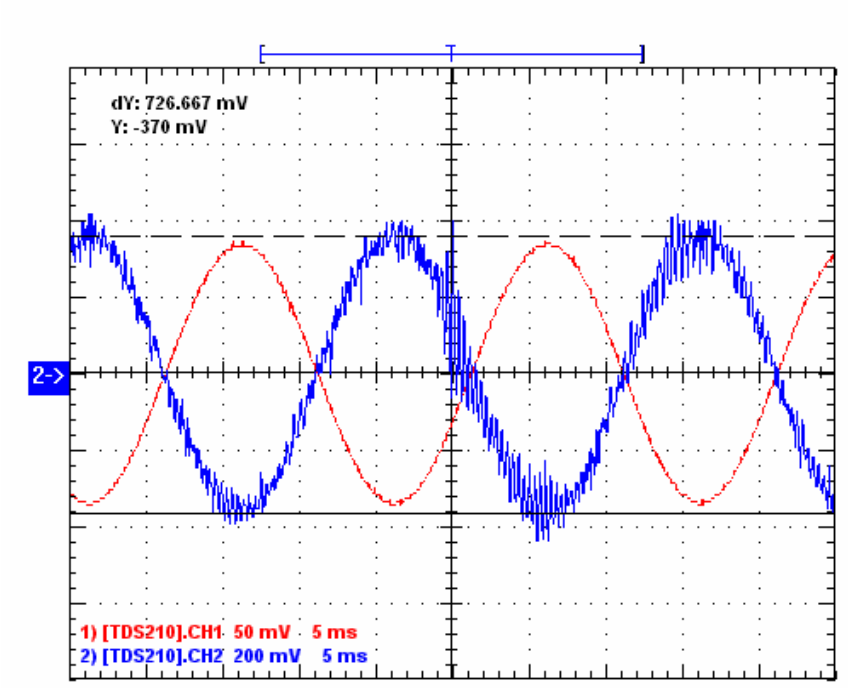

Figura 6.23. Onda 14

Buscamos ahora la tensión superior en el que la onda cambia de forma sustancial, eso si, por la parte alta, es decir, la tensión mayor que se puede introducir al circuito del DSP sin que llegue a saltar las protecciones

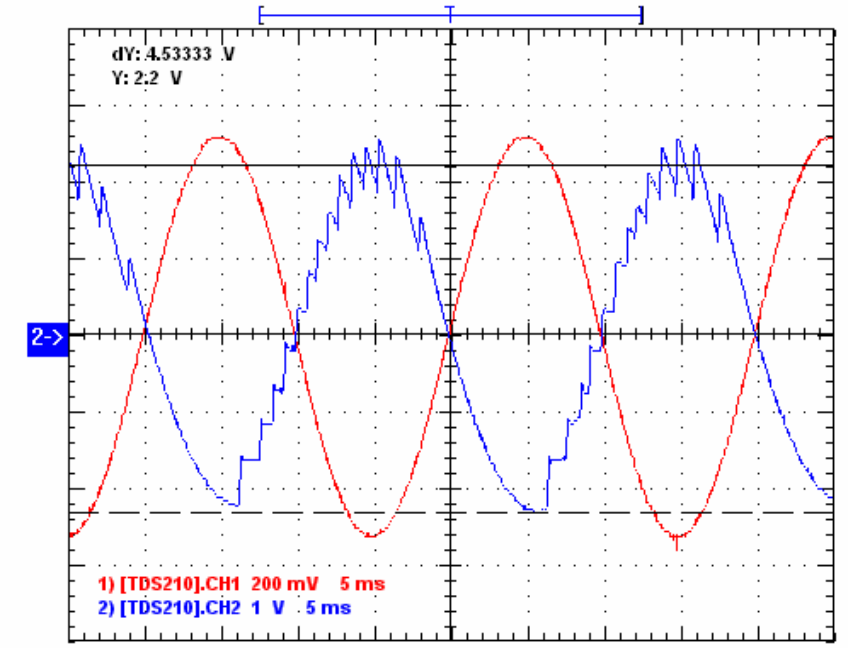

Figura 6.24. Onda 15

Los parámetros para la siguiente prueba son:

- la tensión de entrada es de **¿? V_{pp}** aprox.
- La tensión de alimentación de 24 V aproximadamente.

- El periodo de muestreo es de 500.
- El error es de 100.

A partir de 1.09 voltios, véase figura onda 15, algo superior, el sistema se hace inestable, viéndose que las conmutaciones disminuyen de una forma muy apreciable. En este punto, un aumento de la tensión hace que actúen las protecciones de la placa de adquisición de datos, ello conlleva que la señal pase de senoidal a cuadrada, muy peligrosa si se sabe que hay una inductancia conectada y por tanto produciría un aumento de la intensidad, perjudicial para el montaje (para eso está la protección que se instala en la línea del lazo).

6.4.5 Conclusiones

Por tanto, en este estudio sacamos las siguientes conclusiones

Externas:

Se comprueba que el dispositivo puede funcionar en una banda de tensiones, estando cerca de 24 su óptimo en nuestro experimento.

La tensión de la señal de referencia, la que ofrece el conjunto de amplificación debe de estar comprendida entre 200 mV y 1 V, procurando no sobrepasar la tensión máxima por motivos de seguridad del circuito. Aquí, aumenta el porcentaje de error al disminuir la tensión y disminuye a medida que aumenta la tensión.

Internas:

Escogido unos parámetros externos, los internos es buscar aquellos que, realizando el mínimo esfuerzo de cálculo por el DSP, el sistema trabaje con cierta soltura sin comprometer el funcionamiento del programa debido a la ruptura del mismo por las interrupciones programadas.

Escogidas estas nos quedamos con 500 (40 Khz) como periodo de muestreo y 100 como error de seguimiento de la señal de referencia. Estas pueden variar a lo largo del experimento en función del volumen de cálculo a que se someta al DSP y las exigencias mínimas requeridas para un correcto funcionamiento del sistema.

6.5 El programa Seguidor Onda Ref

En la carpeta de *Seguidor Onda Ref* se encuentra el fichero, con extensión **c,** el cual será cargado en el *code composer* para su ejecución. Por simplificar el proceso de rehacer los ficheros **.c** se ha optado por mantener los nombres de los ficheros y solo cambiar los nombres de los directorios en los cuales están alojados. Esto puede introducir errores de interpretación si no están convenientes ordenados, pero facilita en gran manera la realización de cambios en el fichero y poderse guardar en otro directorio manteniendo la estructura original.

En el proyecto “Diseño de un Sistema de Control para un Motor de Continua basado en DSP” de Juan Carlos del Pino, se detalla con profundidad la estructuras de registros que se utilizarán en sucesivo, por lo que es recomendable leer el capítulo 4 para introducirse en el campo de los registros del DSP en cuestión. Por tanto, no se va a entrar en profundidad en la cuestión de registros y solo se va a comentar el funcionamiento del programa.

En el Anexo, en el apartado 3 se encuentra el programa en su totalidad. Se analiza a continuación el programa obviando la inicialización de los registros.

El programa consiste en realizar una copia de la señal de entrada y pasarla al lazo. Para tenemos que:

- Definir de los parámetros de los registros
- Inicializar los registros
- Programar para capturar la señal y transmitirla al inversor para su reconstrucción.

Primeramente se procederá a la definición de los parámetros de los bits de registro. Para ello se utiliza la directiva “#define” para el compilador sustituya el valor de texto por el valor numérico. Para saber a que registro nos referimos, estos están definidos en el fichero regis243.h que será necesario incluir en el programa. Para ello se utiliza “#include” seguido del nombre del fichero entrecomillado.

Una vez llegado hasta a este punto, comienza propiamente a desarrollarse el programa. En este apartado se procede a definir las variables que se utilizaran el en programa como son:

- ADC0_result, ADC1_result: son dos variables donde se guardaran los valores de la conversión de la señal de intensidad para ACD0 y para la referencia para ADC1.
- PERIOD: Se utiliza para definir el periodo de tiempo entre una conversión y la siguiente.
- ref, io: se utilizan para almacenar los valores del cálculo de obtener la señal de referencia, para ref, y la señal de intensidad, para io, a partir de los datos adquiridos de la conversión.

- err: es el error entre la señal de intensidad y la referencia. Se obtiene realizando la diferencia entre ambos.
- y: se utiliza como bandera para mandar la señal correspondiente al inversor y así producir la rampa que oscilará en el canal que marca el error.

```
#define PERIOD 500 /* 500--->40Khz  T1 PERIOD = 50ns * 1 * 500 =
0.000025s (40Khz)*/

unsigned int ADC0_result,ADC1_result;


signed int ref,io;
signed int err;
unsigned int y;
```

Se define la función de interrupción `interrupt void ADC_ISR(void)`y la función para atrapar interrupciones `void c_dummy1(void).`

```
interrupt void ADC_ISR(void);

void c_dummy1(void);
extern _out_wsgr(); /*declaración de función externa para
            configurar WSGR */


void c_dummy1(void)
{
      while(1);         /*función para atrapar interrupciones
                  espúreas */
}
```

Esta función es peculiar porque es propia de microcontroladores en C. Producida una interrupción en el DSP y después de analizar su prioridad, el programa deja de ejecutar el programa principal y ejecuta la función asociada a la interrupción, para ello hay que asegurarse que corresponde a esa interrupción y no a otra. Para ello se realizará siguiente pregunta `if((PIVR-0x0004)==0).`Si se cumple, se ejecuta la interrupción, si no buscaría la siguiente función de interrupción. Aquí interesa que se pierda el menor tiempo posible, para que el programa principal siga funcionando. Una vez comprobada la interrupción se procede a pasar los valores de la conversión de los registros ADCFIFO a las variables correspondientes definidas anteriormente.

Hay que tener en cuenta que las variables son de 16 bits y los registros de conversión son de 10 bits (resolución de 1024 puntos), por lo que se desplazan estos 10 bits a los bits más significativos de las variables ADC0_result y ACD1_result. La instrucción `ADCTRL1 |= 0x0100` inicializa de nuevo la conversión.

```
interrupt void ADC_ISR(void) /*rutina de interrupción*/
{
```

```
 if((PIVR-0x0004)==0)   /*Verifica la interrupción ( 4 =
                 ADC ) */
      {
      ADCTRL1 |= 0x0100;
      ADC0_result=ADCFIFO2>>6;          /* Intensidad*/
 /*   asm (" LACL 7036h");     /* Vacío las FIFO */
 /*   asm (" LACL 7036h");  */
      ADC1_result=ADCFIFO1>>6;          /* Referencia */

 /*   asm (" LACL 7038h"); */
 /*   asm (" LACL 7038h");  */


      }
}
```

Aquí comienza el programa principal. Como es habitual en C, este comienza con la función main. El programa principal está compuesto de dos partes bien diferenciadas:

1. Esta parte consiste en cargar en los registros correspondientes los valores, definidos anteriormente arriba, para que DSP sea perfectamente operativo para el uso que se le quiere dar. Es fundamental tener especial cuidado en esta parte, pues un error en un registro puede producir errores inesperados e incluso el mal funcionamiento del programa. Esta parte es fundamental e imprescindible si queremos hacer operativo el programa.
2. Una vez cargados todos los registros y actualizados está el núcleo del programa, el cual es la parte que nos interesa, que son las instrucciones que van a manipular los datos para realizar los cálculos correspondientes y manipular los puertos necesarios para realizar los objetivos deseados.

```
void main(void)
{
...
```

Es Importante que se inicialice al principio los registros antes de la puesta en marcha las lecturas y el inversor. A partir de aquí se puede considerar, junto con las rutinas de interrupción, el programa que va a manipular los datos para obtener una señal senoidal a partir de la señal de referencia.

La la rutina del programa se introduce dentro de un bucle infinito, `while(1),` esto nos asegura la ejecución repetida de las instrucciones del bucle, con la salvedad que será interrumpido por la ejecución de la rutina de interrupción para actualizar los valores de ADC0_result y ADC1_result. Dentro del bucle aparecen 2 grupos bien diferenciados.

1. Las operaciones del cálculo para obtener el valor de la intensidad en el lazo (realmente lo que se obtiene es la tensión al ser la intensidad un valor proporcional a la tensión aplicando la ecuación V=R*I) y la tensión

de la señal de referencia. Una vez obtenidas se compara evaluando una diferencia entre ellas.

2. Una vez obtenido el error entre la intensidad del lazo (valor en tensión) y la señal de referencia (también en tensión) se procede a la comparación del error con un valor tabulado el cual cumpliendo la condición:

    a. Si el error supera el valor positivo respecto al tabulado, *err>100*, es decir de una forma aproximada, el valor de salida es mayor que la señal de entrada o referencia (rampa positiva) se procede a mandar al driver que controla los transistores del inversor un 0, *XF=0*. Esto hace que se genere una rampa negativa para contrarrestar el incremento de la señal y proceder a disminuir progresivamente esa diferencia.
    b. Si el error es inferior a un valor negativo respecto al tabulado, *err<-100*, es decir de una forma aproximada, el valor de salida es menor que la señal de entrada o referencia (rampa negativa) se procede a mandar al driver que controla los transistores del inversor un 1, *XF=1*, a través del puerto. Esto hace que se genere una rampa positiva para contrarrestar la disminución de la señal de salida y proceder a disminuir esa diferencia entre la entrada y la salida.

```
 while(1){

       io= (ADC0_result-565)*12; /*lectura I lazo */
        ref= (ADC1_result-555)*49; /*lectura I ref */
        err= (io - ref); /*calcular error*/

        if(err>100)
        {
          asm ("  clrc XF"); /*XF es 0*/
                        /*PDDATDIR= 0xFF00;*/
        }  /*fin if */

        else if(err<-100)
            {
            asm ("   setc XF");     /*XF es 1*/
                        /*PDDATDIR= 0xFFFF;*/
            }   /*fin else if*/
      }  /*fin bucle while(1)*/
} /*fin main() */
```

6.6 OndaDSP

El siguiente programa que ahora se comenta procede de una modificación del programa *Seguidor Onda Ref,* al cual se han realizado unas modificaciones para que esté sigua la señal de referencia creada por el programa por medio de funciones matemáticas.

De cierta manera, esta nueva evolución introduce nuevas características no contemplada en programa anterior *Seguidor Onda Ref.* Estas son:

- Se añade un nuevo temporizador al programa, T2, al cual lleva asociado también la introducción de sus registros asociados a el, como las instrucciones necesarias para la puesta a punto del mismo.
- La introducción de una nueva rutina de interrupción. Como función principal de ésta es el control de tiempos para la formación de la señal de referencia. Además hay que prestar cierta atención a priorizar las interrupciones, por la existencia de varias en el programa.
- Debido que el DSP debe generar una señal senoidal interna, se ha optado por la incorporación al programa de una librería matemática para la utilización de la función seno para generar la señal de referencia.

Comenzando a analizar el programa, primeramente aparece la inclusión de las librerías necesarias para que el compilador pueda enlazar las funciones que se necesitan. Aquí se ha incluido el fichero **math.h** donde están almacenadas las funciones matemáticas que se utilizaran. Investigando en los manuales y en la ayuda que incorpora el programa, esta librería está diseñada para trabajar en coma flotante, pues las variables con las que trabaja son del tipo *float.* Esto es un grave inconveniente pues el DSP trabaja en coma fija. Al no poder trabajar en coma flotante, al compilar el programa, el *Code Composer* transforma las operaciones en coma flotante en coma fija. Estas nuevas rutinas introducidas generan unos tiempos de ejecución demasiados altos, inadmisibles para el correcto funcionamiento para el programa, por lo que es necesario realizar las modificaciones oportunas en el mismo para minimizar dichos efectos.

Por tanto debido a los tiempos de ejecución, aproximadamente de 7000 ciclos de reloj (si la frecuencia de reloj es de 20 MHz, en la ejecución de las instrucciones tarda unos $3.5*10^{-4}$ segundos). Es un tiempo el cual hace imposible el seguimiento de la señal. A continuación se va a exponer una parte de las instrucciones del programa que se encuentra en el directorio *OndaDSP* puesto son, a mi juicio, las que muestran las mayores diferencias respecto al programa *Seguidor Onda Ref.*

```
/* definimos las variables para el seno */
signed int refin;
float pi=3.141596;
float seno;

unsigned int ADC0_result,ADC1_result;


signed int io;
signed int err;
unsigned int y;
/*calculo de tiempos de ejecución de los programas*/
unsigned int t2;


interrupt void ADC_ISR(void);
interrupt void ONDA_SEN(void);
void c_dummy1(void);
extern _out_wsgr();
```

```
void c_dummy1(void)
{
      while(1);         /*bucle para capturar interrupciones espureas*/
}


interrupt void ADC_ISR(void) /*rutina de interrupción*/
{

 if((PIVR-0x0004)==0)   /*Verifica la interrupción ( 4 =
                 ADC ) */
      {

      ADC0_result=ADCFIFO2>>6;          /* Intensidad*/
 /*   asm (" LACL 7036h");     /* Vacío las FIFO */
 /*   asm (" LACL 7036h");  */
      ADC1_result=ADCFIFO1>>6;          /* Referencia */

 /*   asm (" LACL 7038h"); */
 /*   asm (" LACL 7038h");  */


      t2=T2CNT;
      seno=sin(2*pi*T2CNT/PERIOD2);
                 /*calcula el dato en radianes 0 a 2pi */
      refin=seno*4000;
      /* valor a escala del conversor */
      err=io-refin;
      /*calcula el error*/
      ADCTRL1 |= 0x0100; /*activar conversor AD */

      }
   }

interrupt void ONDA_SEN(void)
   {
    if((PIVR-0x002B)==0) /*verifica interrupción Compara Timer2 */
          EVIFRB=(T2PINT);     /*habilita la interrupción*/
}
```

Éste es el núcleo que varia principalmente respecto al programa anteriormente visto. El mayor problema se da en las instrucciones para generar la señal seno, pues se pretende que se hagan en tiempo real y el DSP no tiene la suficiente potencia para hacerlo, por lo que se debe buscar alternativas que, introduciendo complejidad en el programa, lo haga lo más funcional posible. Bien, lo más novedoso son las siguientes instrucciones:

```
t2=T1CNT;
seno=sin(2*pi*T2CNT/PERIOD2);
refin=seno*4000;
err=io-refin;
```

Con *t2* se pretende almacenar el valor del registro que actualmente tiene y que estará comprendido entre 0 y PERIOD2 donde 0 corresponde 0 grados y PERIOD2 a 360 grados. A medida que el temporizador avanza T2CNT se va actualizando y de esa forma se crea la señal seno en el tiempo. Por tanto, por medio de

$$ángulo = 2 \cdot \pi \cdot {T2CNT}/{PERIOD2}$$

se obtiene la fase de la señal senoidal. Aplicando posteriormente la función **sin()** integrada en la librería math.h que incorpora el *Code Composer.* He aquí donde, al operar en flotante y del DSP en coma fija, los tiempos de ejecución se disparan. En la operación para obtener *refin* es algo menor a 1000 ciclos de reloj, que todavía es muy alto puesto lo normal es estar comprendido entre 2 y 10 ciclos una operación de este tipo. Esto se debe a que la operación de multiplicación de seno por 4000 (factor para poder comparar con la salida) se sigue realizando en flotante.

Debido a que la estructura en la que se plantea hace imposible su ejecución en unos tiempos aceptables, se da por invalidado el camino a seguir para realización del programa compensador del lazo. ¿Eso quiere decir que hay que tomar otro camino?, la respuesta es no. Visto donde están los inconvenientes, se buscará una solución de compromiso teniendo en cuenta este inconveniente. En este caso no es posible el cálculo en tiempo real para crear la señal de referencia, al menos, con el DSP de actualmente se utiliza en este estudio, el 243 de la familia TMS de Texas Instruments.

La existencia de otra rutina de interrupción, *interrupt void ONDA_SEN(void),* tiene como objetivo definir el fin del periodo de la señal senoidal para resetear T2CNT y comenzar de nuevo.

6.7 DSPseno

Partiendo del análisis del programa *Seguidor Onda Ref* y *OndaDSP* se llega a éste que no es más que el embrión de partida para el futuro programa de control del lazo. Aquí, una vez solucionado el problema anterior, se pretende crear una señal, introducirle un desfase y a continuación hacer que la salida "siga" la referencia creada. El objetivo es ver como actúa el conjunto de los circuitos, incluido el DSP, al ser creado la referencia internamente y ser seguida, que es lo que ocurría en el programa ControlNoMot.

Las características que resaltan fundamentalmente para hacerlo operativo son:

- La existencia de dos temporizadores, T1 para controlar los tiempos de captura del DSP y T2, para controlar la formación de la señal de referencia. Esto conlleva también la introducción de sus registros asociados, como las instrucciones necesarias para la puesta a punto del mismo.

- La utilización de rutinas de interrupción que, asociadas a los temporizadores, controlan la entrada de datos y la ejecución del núcleo del programa.
- La creación de una tabla de valores al inicio de ejecución para no tener que depender de los tiempos de ejecución de la función seno. Más tarde se explicará su construcción y funcionamiento.

En este programa se puede estructurar en los diferentes bloques que a continuación se describe:

- Inclusión de las librerías necesarias para el correcto funcionamiento de las diferentes funciones.
- Definición de los parámetros de los registros y de las variables del entorno del programa principal y funciones.
- Definición de las funciones y construcción de las mismas. Aquí se incluyen las dos rutinas de interrupción utilizadas.
- Inicialización de los registros en el programa principal.
- Programa que genera la señal de salida en función de los datos capturados por el DSP y analizados por las rutinas de interrupción.

Antes de todo, y para que reconozca las funciones matemáticas, se incluye la librería **math** y así estará operativo la función **sin()**. A continuación se define los valores que deben de tener los registros en el momento de la inicialización del programa. Estos registros, en función de la operativa del programa, pueden variar y adoptar otros valores.

Definidos los registros comienza propiamente a desarrollarse el programa. En este apartado se procede a definir las variables que se utilizaran el en programa como son:

- ADC0_result: es donde se almacena los valores de la conversión de la señal de intensidad para ACD0. Aquí no es necesario el ACD1.
- PERIOD: se utiliza para definir el periodo de tiempo entre una conversión y la siguiente.
- PERIOD2: se utiliza para definir el periodo de tiempo de una onda completa senoidal.
- ref, io: se utilizan para almacenar los valores del cálculo de obtener la señal de referencia, para *ref*, y la señal de intensidad, para *io*, a partir de los datos adquiridos.
- err: es el error entre la señal de intensidad y la referencia. Se obtiene realizando la diferencia entre ambos.
- y, n, tref: variables que se utilizan para almacenar datos auxiliares en los cálculos.
- seno[360], senodesf[360]: son dos tablas de valores donde se almacenan los datos obtenidos a partir de la función seno y el seno desfasado.
- desf, DESFASE: el desfase que se le quiere dar a la señal respecto a la calculada en seno[360].

- refin, refret: utilizadas como variables auxiliares para obtener seno[360] y senodesf[360].
- pi: el númeroπ.

```
#define PERIOD 500 /* 500--->40Khz     T1 PERIOD = 50ns * 1 * 640 = 0.000032s
(31Khz)*/

/* Periodo de formación de la onda senoidal */
#define PERIOD2 3125 /* 50ns*128*3125=.002s o sea 50Hz. Si 360/3125 = 0.1125º
*/

#define DESFASE 40  /*valor en grados del desfase que se le quiere introducir
a la señal*/

/* definimos las variables para el seno */

float pi=3.14159;

signed int seno[360],senodesf[360]; /*tablas del seno y seno desfasado*/

signed int refin,refret;  /*señales de referencia internas y referencia
internas desfasada*/

unsigned int tref; /*tiempo de referencia*/

unsigned int desf=DESFASE; /*desfase*/

unsigned int ADC0_result;


signed int ref,io,err;

unsigned int y,n;  /*parametros de interacion*/
```

Se define la función de interrupción interrupt void ADC_ISR(void), interrupt void ONDA_SEN(void)y la función para atrapar interrupciones void c_dummy1(void).

```
interrupt void ADC_ISR(void);
interrupt void ONDA_SEN(void);

void c_dummy1(void);
extern _out_wsgr(); /*declaración de función externa para
       configurar WSGR */


void c_dummy1(void)
{
      while(1);          /*función para atrapar interrupciones
             espúreas */
}
```

Es ésta la primera de las rutinas de interrupción. Su función principal es la de capturar datos externos a través del ADC con el objetivo de realizar un seguimiento de las señales. Una vez verificado la interrupción (PIVR debe de ser igual a 0x0004) y se activa la rutina, primeramente se lee el valor que

actualmente tiene T2CNT que es donde almacena la base de tiempos de la señal senoidal de referencia, programada para ser una señal de 50 Hz. Esta base está definida según la ecuación `50ns*128*3125=.002s` donde 3125 son las divisiones el timer2, así cada vez que se actualice este temporizador, habrá avanzado la señal 0.1125 grados, por lo que será este la precisión de nuestra señal. A continuación se obtiene los datos de la señal del lazo y seguidamente se calcula la tensión de dicha señal.

```
interrupt void ADC_ISR(void) /*rutina de interrupción*/
{

 if((PIVR-0x0004)==0)     /*Verifica la interrupción ( 4 =
                    ADC )  */
      {
   /*lo primero que hacemos es leer el tiempo de la onda que generamos*/
      tref=T2CNT; /*valor entre 0 y 3125*/
      ADC0_result=ADCFIFO2>>6;          /* Intensidad*/
 /*   asm (" LACL 7036h");     /* Vacío las FIFO */
 /*   asm (" LACL 7036h");  */

      io= (ADC0_result-565)*12;

      ADCTRL1 |= 0x0100;   /*activar conversor*/
      }
}
```

Esta es la parte novedosa del programa, la rutina para controlar la base de tiempos de la señal de referencia. Una vez introducida en el registro T2PR el dato PERIOD2, y se actualiza T2CNT, ya está operativa está interrupción. El objetivo final es controlar que al llegar T2CNT al número cargado en T2PR, este se resetee y comience de nuevo. Esto se hace con la instrucción `EVIFRB=T2PINT.` Con ello cada PERIOD2, conseguimos que esta rutina se active cada 50 Hz, que es la frecuencia con la que se quiere construir la referencia.

```
interrupt void ONDA_SEN(void)   /* Para construir la señal      senoidal*/
{
      if((PIVR-0x002B)==0) /*verifica interrupción Compara Timer2 */
          EVIFRB=T2PINT;     /*habilita la interrupción escribiendo un 1*/
}
```

Una vez llegados a este punto y definidas las rutinas de interrupción comienza la parte principal de programa. En ella se define los siguientes bloques.

- Esta parte consiste en cargas en los registros correspondientes los valores definidos anteriormente arriba para que DSP sea perfectamente operativo para el uso que se le quiere dar. Se hará mención sobre los registros de los temporizadores más adelante.
- Una vez cargados todos los registros y actualizados, comienza el núcleo del programa donde aparecerán las instrucciones que construye la tabla de datos del seno y la que hace seguir esa tabla, la señal de salida.

He aquí la parte novedosa que resuelve el problema planteado en el programa *OndaDSP*. En esta parte lo que se intenta resolver y dar solución es a la cuestión de los tiempos de ejecución de la función seno. Construyendo una tabla en el tiempo que se inicia cuando arranca el programa, hace que nos ahorremos una cantidad de tiempo importante cada vez que se necesite conocer el valor del seno, ya que solo hay que consultar a la tabla y no calcularlo. De esta manera el valor es aproximado pero con un error mínimo para el rango en que se trabaja, 1 grado. Esto tiene un ligero inconveniente, es el tiempo que se necesita para calcular la tabla para del seno y del seno desfasado, que ronda algo superior a 0.02 segundos. Al realizarse durante el arranque, y por tanto no necesario durante la etapa de control, hace que el programa vaya más suelto.

El conjunto se divide en dos bloques, el primero compuesto por la formula en el que se calcula de los 360 valores solo 180, el semieje positivo, a lo que a continuación se obtiene el semieje negativo con solo multiplicar por –1 los valores antes obtenidos. Una vez calculado la tabla del seno, solo queda introducir el desfase que se desea para la onda según el procedimiento que se adjunta.

```
/*********************************************************/
 /*Programa generación tabla del seno y desfase*/


 for(n=0;n<180;n++)
  {
       seno[n]=1023*sin(pi*n/180);        /*el valor de lo almacenado va desde
0 hasta n-1*/
     seno[180+n]=-seno[n];              /*1023, factor de escala*/
       }

  for(n=0;n<360;n++)    /*creación onda desfasada*/
  {
       if(n+desf<360)
             {
             senodesf[n+desf]=seno[n];     /*ángulo comprendido entre desf y
360*/
             }
       else
       {
             senodesf[(n+desf)-360]=seno[n];    /*comprendido entre 0 y
desf*/
             }
  }

 /*********************************************************/
```

Aquí, en el bucle, se encuentra la parte que, como anteriores programas, controla las conmutaciones de inversor activando uno u otro transistor. Ahora bien, para el cálculo de la señal senoidal para su seguimiento por la señal de salida, aquí se ha incluido las operaciones que, en función del tiempo, obtiene

el valor del seno aplicando unos coeficientes para amplificar la señal. Con esta señal, solo queda comparar la señal de salida con la referencia y obtener el error que será el parámetro con lo que trabajará el bucle para actuar sobre el inversor.

```
 while(1){

        n=(tref*360)/PERIOD2; /*valor comprendido entre 0 y 359 grados*/

        ref=5*seno[n]; /*amplificamos la señal por 5*/
        err=io-ref;  /*error de seguimiento*/



        if(err>100)
        {
           asm ("clrc XF"); /*XF es 0*/
                         /*PDDATDIR= 0xFF00;*/
        }

        else if(err<-100)
            {
           asm ("setc XF");     /*XF es 1*/
                                /*PDDATDIR= 0xFFFF;*/
            }
      }
}
```