Control de un motor brushless con aumento de la resolución del resolver mediante “oversampling”

Memoria

Autor: Gabriel Gross Igor
Director: Joan Bergas

Titulación: Ingeniería Industrial
Convocatoria: Septiembre 2004

Resumen

En este proyecto se ha diseñado y montado el control completo de un motor brushless senoidal con un DSP (Digital Signal Processor). En el DSP no se solo se ha programado el control propiamente dicho, sino también un algoritmo para adquirir la posición del motor con más resolución de la que nos permite el algoritmo tradicional (basado en el undersampling); esta técnica es el sobremuestreo (oversampling en inglés).

El control que se ha diseñado sirve como base para algoritmos superiores que también se pueden integrar en el DSP y por tanto pueden aprovechar la potencia de cálculo del mismo. Con esto se puede lograr algoritmos que realicen funciones relativamente complejas con motores, gracias a la gran flexibilidad de programación que ofrece el DSP.

Este DSP controla un puente ondulador trifásico de hasta 100A, con el que es posible controlar casi todo el rango de motores brushless comerciales. Por otro lado, con el mismo equipo y programa, también es posible controlar motores síncronos, siempre que lleven un resolver.

Para realizar este control se ha diseñado y construido una placa de control que integra el DSP y otros elementos auxiliares. Por otra parte, también se han construido y montado placas auxiliares para el funcionamiento de todo el conjunto (fuente de alimentación, filtro, sondas,…)

Índice

1 Glosario

- Oversampling o sobre-muestreo: técnica que consiste en muestrear una señal con más frecuencia de la que es necesario según el teorema del muestreo. En este documento se hablará indistintamente de sobre-muestreo o de su equivalente en inglés oversampling.

- Aliasing: solapamiento de las frecuencias que se produce si no se cumple el teorema del muestreo.

- DAC (o conversor D/A) y ADC (o conversor A/D): Conversor analógico a digital y conversor digital a analógico respectivamente

- Dithering: adición de ruido blanco a una señal para que esta se convierta en una señal constantemente cambiante.

- DSP (Digital Signal Processor): procesador digital de señales

- Interrupción: evento del DSP que redirecciona el programa que se ejecuta hacia una porción del código asociada al evento.

- PLL (Phase Loop Lock): generalmente se usa para designar un tipo de sistema, eléctrico o algoritmico, que se encarga de mantener constante la fase del ángulo de una señal de entrada oscilante.

- PWM (Pulse Width Modulation): modulación por ancho de pulsos.

- Resolver: aparato que sirve para obtener la posición angular absoluta en motores.

- SKiiP: ondulador trifásico de la marca Semikron que se ha utilizado en este proyecto.

- SNR (Signal to Noise Ratio): ratio entre la potencia de la del ruido y la potencia de la señal.

- SVPWM (Space Vector PWM): técnica para generar señales PWM mediante los 8 vectores de tensión sintetizables con un ondulador trifásico.

2 Introducción

2.1 Objetivos

El objetivo de este proyecto es el desarrollo de un control digital de bajo coste para un motor brushless (sin escobillas). Para el tratamiento digital de las señales, control del convertidor de potencia e implementación del algoritmo de control se usará un DSP (procesador digital de señales) de bajo coste de la marca Texas Instruments.

El DSP que se va a usar es un componente relativamente barato pero de gran potencia de cálculo, este va montado sobre una placa electrónica de dos capas y de bajo coste. Esa placa también se diseñará y construirá.

Por otro lado se pretende conseguir un aumento de resolución en la posición leída a través del resolver que incorpora el motor, esto se pretende conseguir mediante la técnica del oversampling (o sobre-muestreo). Esto se hace para suplir la poca resolución (y por consiguiente poca precisión) que tiene el conversor analógico digital del DSP y para solventar el ruido que tiene la señal. Aunque existen soluciones específicas para la conversión resolver-to-digital se quiere evitar el uso de estas soluciones por el elevado coste que suponen. Lo que se pretende es integrar un método parecido al que utilizan estas soluciones pero mediante software en el DSP, y tener con ello una solución compacta y barata que haga todas las funciones en un solo encapsulado.

2.2 Abasto

Este proyecto contempla los siguientes apartados:

- Diseño conceptual de todo el sistema.
- Simulación informática del sistema para verificar el funcionamiento del diseño, detectar parámetros necesarios en el sistema real y detectar posibles problemas.
- Diseño de elementos electrónicos: placa de control para el DSP, fuente de alimentación, filtro para el resolver y sondas de corriente.
- Montaje de todos los elementos electrónicos necesarios para el prototipo: tanto la placa de control como los elementos auxiliares.
- Montaje del prototipo completo

- Implementación y optimización del algoritmo de control de velocidad y control de par del motor, en el DSP.
- Comprobación de los resultados.
- Análisis económico del prototipo.

3 Conceptos preliminares

Para comprender los siguientes capítulos se va ha hacer una breve explicación del funcionamiento de los elementos más importantes que integra el sistema.

3.1 Motor brushless

El motor brushless es básicamente un motor síncrono trifásico que tiene un rotor con imanes permanentes. Los devanados del estator son alimentados con tensiones de manera que el imán permanente del rotor sigue los campos magnéticos creados por los devanados del estator. Según T.J.E Miller [17] hay dos tipos de motor brushless: el motor brushless trapezoidal y el motor brushless senoidal. Aunque el principio básico de funcionamiento es totalmente igual, la diferencia más destacable está en la forma de alimentar los devanados del estator. En el motor brushless senoidal cada una de las fases (devanados) se alimenta con pulsos rectangulares de tensión con un desfase entre cada una de las fases de 120º. En el motor brushless senoidal en cambio se alimenta con tensión alterna trifásica.

Esta diferencia es a nivel de uso, a nivel de construcción el motor brushless senoidal se diferencia del trapezoidal por lo siguiente:

- El flujo del entrehierro es senoidal y está generado por los imanes del rotor que tienen una forma especial.
- Los devanados tienen una distribución senoidal, mientras que en el trapezoidal los devanados están concentrados.

En este proyecto se usará un motor brushless senoidal que incorpora un resolver para la lectura de la posición y por tanto también la velocidad.

3.2 Resolver

El resolver es un sistema para obtener la posición absoluta del eje del rotor de un motor. Generalmente está acoplado directamente al motor. Se suele usar sobretodo en aplicaciones donde se requiere conocer la poción con cierta precisión. El funcionamiento del resolver es el de un transformador, se basa en el esquema de la Ilustración 3.1:

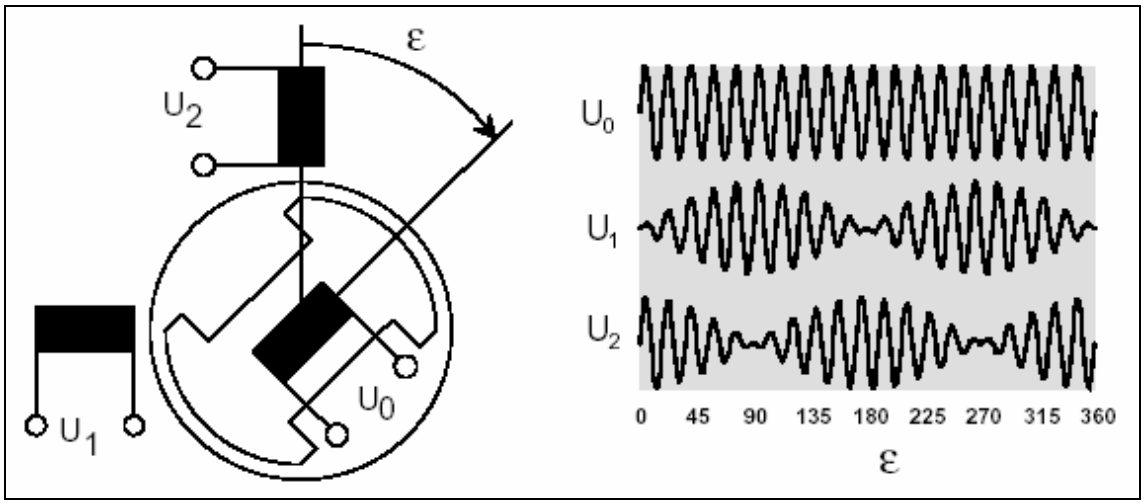

Ilustración 3.1: Esquema de un resolver y las señales de entrada y salida.

En el rotor hay una bobina (con N vueltas) que se alimentada con tensión alterna u_0 de frecuencia f_{ref}, que sirve como portadora. En el estator hay otras dos bobinas (normalmente de N/2 vueltas) desfasadas 90º una de la otra, en estas se inducen las tensiones u_1 y u_2, que son la señal portadora (u_0) moduladas con el seno y coseno respectivamente del ángulo mecánico (ε), como se ve en la Ilustración 3.1. Estas señales siguen las siguientes formulas:

$$\begin{aligned} u_0(t) &= \hat{u}_0 \cdot \sin(\omega_{ref} t) \\ u_1(t,\varepsilon) &= \hat{u}_0 \cdot \sin(\omega_{ref} t) \cdot 0.5 \cdot \sin(\varepsilon) \\ u_2(t,\varepsilon) &= \hat{u}_0 \cdot \sin(\omega_{ref} t) \cdot 0.5 \cdot \cos(\varepsilon) \end{aligned} \qquad \text{Ec. 3.1}$$

Con las señales u_1 y u_2 es posible obtener el ángulo mecánico del motor (ε) operando de la siguiente manera:

$$\begin{aligned} \arctan\left(\frac{u_1(n)}{u_2(n)}\right) &= \arctan\left(\frac{\hat{u}_0 \cdot \sin(\omega_{ref} t) \cdot 0.5 \cdot \sin(\varepsilon)}{\hat{u}_0 \cdot \sin(\omega_{ref} t) \cdot 0.5 \cdot \cos(\varepsilon)}\right) = \arctan\left(\frac{\sin(\varepsilon)}{\cos(\varepsilon)}\right) = \\ &= \arctan(\tan(\varepsilon)) = \varepsilon \end{aligned} \qquad \text{Ec. 3.2}$$

Como la función arcotangente no es biyectiva se tiene que operar de la siguiente manera para obtener el ángulo en los 4 cuadrantes:

$$\varepsilon(n) = \begin{cases} \arctan\left(\dfrac{u_1(n)}{u_2(n)}\right) & si\ u_2(n) \geq 0 \\ \pi + \arctan\left(\dfrac{u_1(n)}{u_2(n)}\right) & si\ u_2(n) < 0 \end{cases} \qquad \text{Ec. 3.3}$$

El espectro de las señales u_1 y u_2 es el de la Ilustración 3.2.

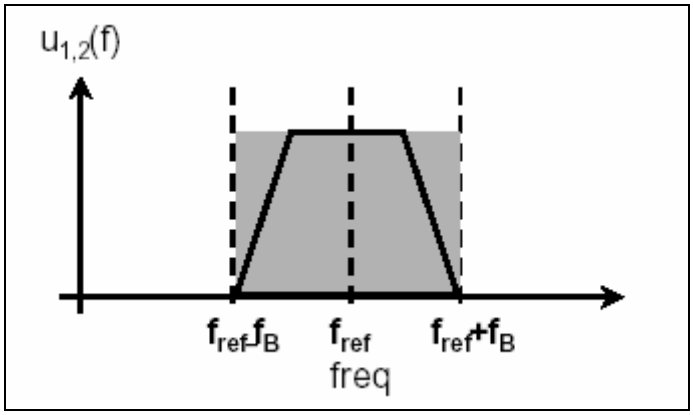

Ilustración 3.2: espectro frecuencial de las señales de salida del resolver.

Donde:

$$f_B = \frac{\omega_{mec,\max}}{2\pi} = \frac{1}{2\pi}\frac{d\varepsilon}{dt}\bigg|_{\max} \qquad \text{Ec. 3.4}$$

3.3 Convertidor resolver-to-digital mediante undersampling

Una manera simple y común para sacar el ángulo girado (ε) es usar la llamada técnica del undersampling, cuyo esquema de funcionamiento viene representada en la Ilustración 3.3. Se llama undersampling, por que se muestrea a una frecuencia menor que el mínimo que impone el teorema del muestreo de Shannon para recuperar completamente la señal.

En este método se digitalizan u_1 y u_2 a la frecuencia de referencia (f_{ref}), con ello se demodula la señal y por tanto se obtiene directamente el seno y el coseno del ángulo mecánico (ε). Con estas dos señales se calcula el arcotangente de su cociente, así se tendrá directamente el ángulo mecánico (ε) con N+1 bits de resolución teóricamente, siendo N el número de bits del ADC.

Para obtener la máxima precisión ambas señales se tienen que muestrear a la vez en su valor máximo (o cerca de el) sincronizado con la señal de referencia; es decir, en el instante t_n, según:

$$\omega_{ref} t_n = (4n+1)\frac{\pi}{2} \qquad n = 0,1,2,3,... \qquad \text{Ec. 3.5}$$

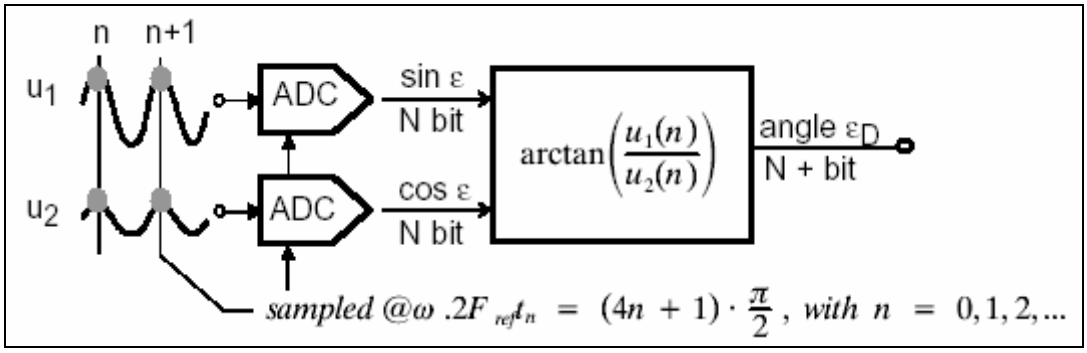

Ilustración 3.3: esquema de funcionamiento del resolver-to-digital mediante undersampling.

Para evitar el aliasing (el solapamiento de frecuencias al muestrearlas) hay que muestrear a una frecuencia mínima de $2f_B$, para ello se tendrá que poner un filtro analógico anti-aliasing que elimine las frecuencias fuera de la banda (f_{ref}-f_B, f_{ref}+f_B) antes de muestrearlas. Obviamente esto eliminará el offset de las señales antes de muestrearlas, cosa que podría reducir la resolución efectiva.

La ventaja de este método es su sencillez de aplicación, pero tiene diversos inconvenientes, estos son:

- Necesita un filtro anti-aliasing analógico pasa banda, que generalmente son complicados de diseñar e implementar.
- Para conseguir la máxima precisión se ha de muestrear siempre en el máximo de la señal, esto puede ser difícil, sobretodo si se tiene en cuenta los retrasos de los filtros que hay en el camino de la señal, las variaciones que pueden sufrir estos retrasos a medida que se calienten y/o envejecen los componentes.
- La implementación de la función división y arcotangente, son funciones que requieren un importante tiempo de cálculo cuando se implementan en un DSP.
- La resolución depende únicamente de la resolución del ADC.

3.4 Conversión AD con oversampling

Para hacerse una idea intuitiva, se puede decir que el oversampling (o sobremuestreo) se basa en coger más valores de los necesarios para luego promediarlos y obtener con ello un solo valor más exacto.

Realmente los métodos de oversampling para convertidores Analógicos a Digital se basan en digitalizar la señal analógica, muestreando a una frecuencia mayor que la requerida. Luego esta señal se filtra con un filtro digital pasa-bajos y finalmente se vuelve a reducir la frecuencia de muestreo a través de un diezmado. El diezmado consiste en coger uno de cada k muestras, reduciendo así la frecuencia de muestreo k veces, ya que no es necesario para la aplicación, adquirir tantas señales por unidad de tiempo.

Esta técnica tiene varias ventajas. La primera es la reducción de las especificaciones de la señal de entrada; por eso se puede prescindir del filtro analógico de entrada e implementarlo digitalmente. La segunda es que se consigue un aumento de resolución en la señal de salida.

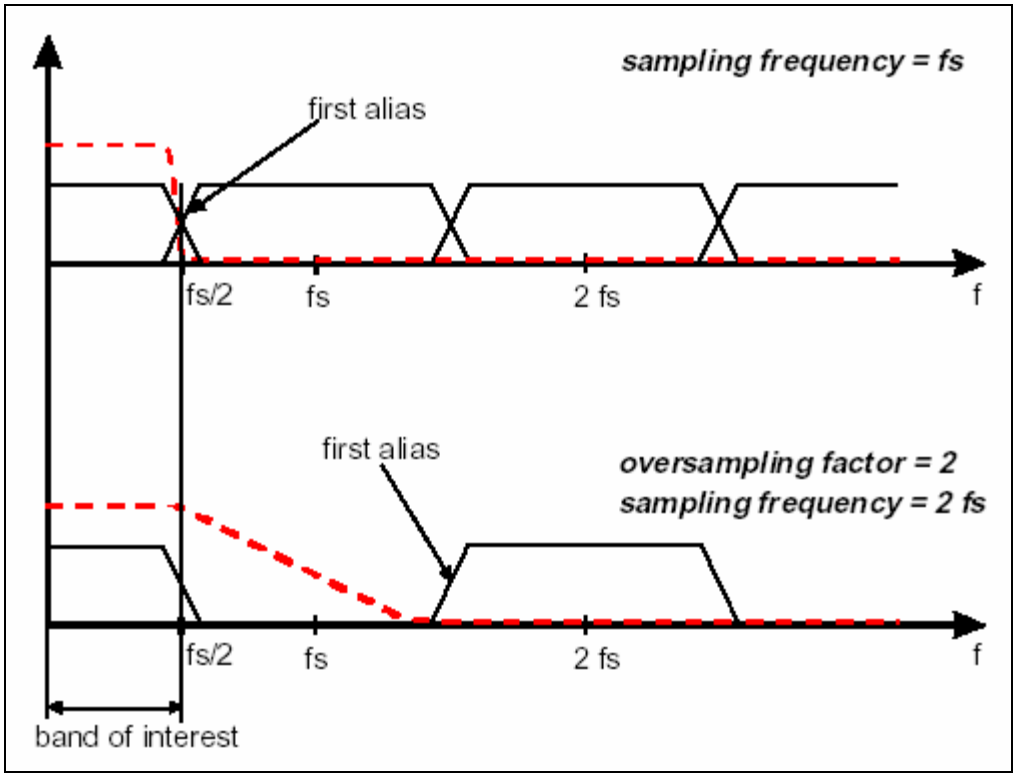

Ilustración 3.4: disminución de los requerimientos del filtro al aumentar la frecuencia de muestreo f_s.

Como vemos en la Ilustración 3.4 al muestrear a mayor frecuencia, el aliasing se produce a frecuencias mayores ($f_s/2$ ahora es mayor, siendo f_s la frecuencia de muestreo mínima según el teorema del muestreo), y por tanto los requerimientos del filtro (en rojo) son mucho menores (tienen una pendiente mucho menos abrupta). Por esto cabe la posibilidad no poner un filtro analógico e implementarlo digitalmente.

3.4.1 Principio teórico del oversampling

Al cuantificar la señal muestreada, se introduce un error, que se denomina ruido de cuantificación. Si se tiene una señal constantemente cambiante se puede modelar este error como ruido blanco que reparte su energía uniformemente a lo largo de todo la banda de frecuencias que va desde el cero hasta la mitad de la frecuencia de muestreo. Para señales no cambiantes o poco cambiantes, el ruido no es de este tipo. Para poder aplicar el oversampling, la señal de entrada del conversor AD (ADC) ha de ser una señal constantemente cambiante. Para conseguir esto se le añade una señal de error (dithering signal). Para que el sistema funcione correctamente esta señal ha de tener una amplitud de al menos el valor del bit menos significativo del ADC.

Según Texas Instruments [12], para aumentar la resolución y dependiendo de la señal de error que se añada, hay principalmente dos métodos:

- Aumento de resolución añadiendo ruido blanco
- Aumento de resolución añadiendo una señal triangular

A todo esto se ha de comentar, que cuando la señal varía muy rápido, más de un bit por periodo de muestreo, ninguno de los dos métodos tiene mucho sentido. Ya que, a medida que la señal va cambiando más rápidamente por cada periodo de lectura de los valores, cada vez va teniendo menos importancia el calcular el valor con más resolución, ya que se pasa de calcular un valor más exacto a calcular una media del valor que ha tenido en un periodo, esto se puede ver gráficamente en la Ilustración 3.5.

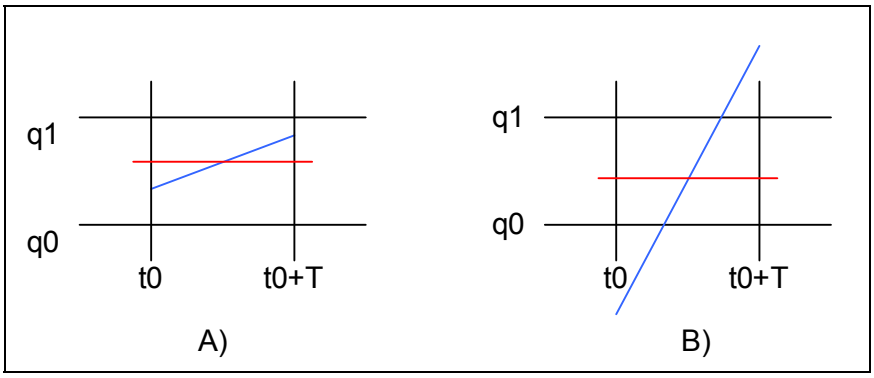

Ilustración 3.5: A) señal que varía más despacio de lo que se lee, B) señal que varía más rápido de lo que se lee.

3.4.2 Aumento de resolución añadiendo ruido blanco

Como se ha dicho antes, el ruido de cuantificación distribuye su potencia uniformemente en todo el espectro desde cero hasta la mitad de la frecuencia de muestreo. Su potencia es independiente de la frecuencia de muestreo. Así cuando se usan frecuencias de muestreo superiores (k veces), el ruido se reparte uniformemente sobre un rango mayor de frecuencia, resultando en una densidad de potencia efectiva menor en la banda de interés, ver Ilustración 3.6.

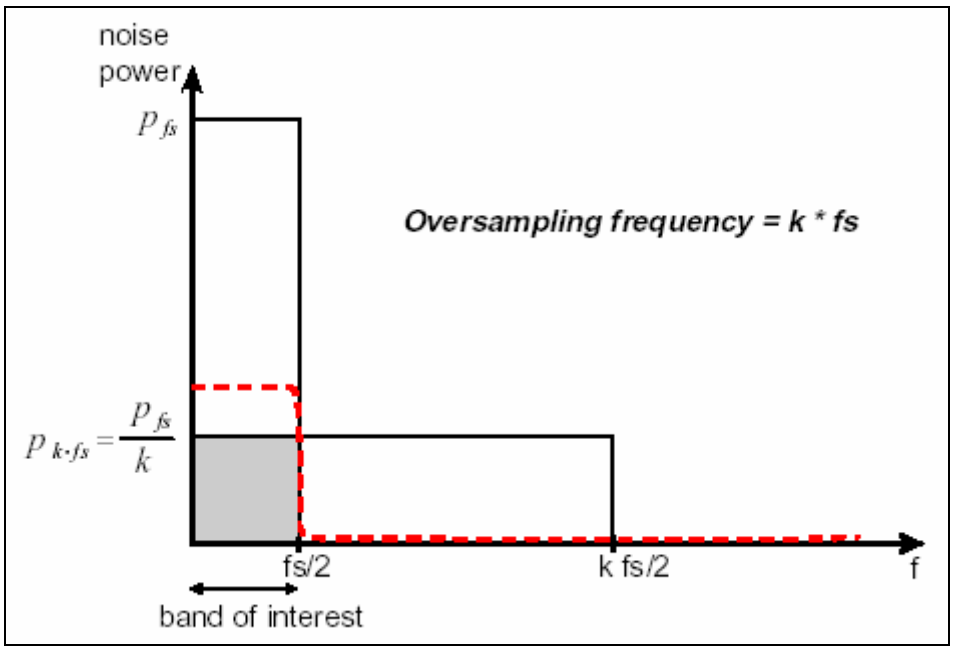

Ilustración 3.6: repartición de la energía del ruido a lo largo del espectro de frecuencias, cuando se muestrea a f_s y cuando se muestrea a $k \cdot f_s$.

El filtro pasabajos quitará las frecuencias superiores a fs/2, dejando un error mucho menor del que se tendría. Finalmente se reduce la frecuencia de muestreo k veces por medio del diezmado. Según Texas Instruments [12], con esto conseguimos una mejora en el SNR (ratio entre la potencia del ruido y la potencia de la señal) de:

$$SNR_{máx}\left[dB\right] \approx 6.02 \cdot N + 1.76 + 10 \cdot \log_{10}\left(k\right) \qquad \text{Ec. 3.6}$$

Donde N es el número de bits del ADC y k es el factor de sobremuestreo.

Según el factor de sobre-muestreo que se utilice se consigue las mejoras indicadas en la Tabla 3.1.

Factor de oversampling	Mejora del SNR [dB]	Resolución extra [bits]
2	3	0.5
4	6	1.0
8	9	1.5
16	12	2.0
32	15	2.5
64	18	3.0
128	21	3.5
256	24	4.0
512	27	4.5
1024	30	5.0
2048	33	5.5
4096	36	6.0

Tabla 3.1: mejora en la resolución al utilizar el oversampling añadiendo ruido blanco.

Como se ve, se consigue una mejora de 3dB (0.5 bit) en la resolución por cada vez que duplicamos el factor de sobre-muestreo.

Para generar el ruido se requiere tiempo de cálculo adicional, aunque también puede servir el ruido que introduce el propio ADC, con lo que se ahorra este tiempo de cálculo.

Este método no pone limitaciones en la forma de onda y se puede usar en muchas aplicaciones, sobretodo cuando es posible usar factores de sobre-muestreo grandes.

3.4.3 Aumento de resolución añadiendo una señal triangular

Con esta técnica se añade una señal triangular que idealmente varía entre n-0.5 y n+0.5 veces el valor del bit menos significativo del ADC, siendo n un valor entero.

Cuando la señal de entrada esta entre dos bits q_0 y q_1, al añadir esta señal triangular, el ADC convertirá esta señal o bien a q_0 o bien a q_1, en vez de convertirlo todo el rato al mismo. Así la relación del número de veces que se convierta a uno u a otro representa la posición relativa que tiene entre los dos bits.

Así en la Ilustración 3.7 se muestra un ejemplo con un factor de sobre-muestreo de 16 y una señal de entrada que cuya posición relativa entre los bits q_0 y q_1 es 0.6. Si se leyese la señal sin añadir nada, se leería siempre el valor q_1.

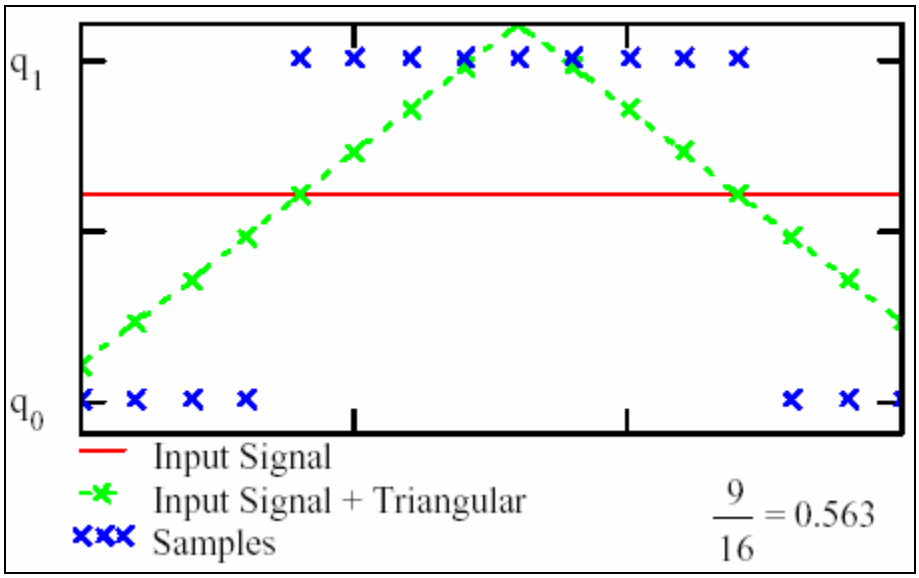

Ilustración 3.7: ejemplo de oversampling añadiendo una señal triangular

Sin embargo cuando añadimos la señal; 9 de las 16 veces se aproxima más a q_1 y 6 veces a q_0, por lo que si se promedia y diezma con un factor de 16, se tendrá una posición relativa entre estos dos bits de 0.563, más precisa que si no se aplica oversampling.

Así según Texas Instruments [12], el SNR es de:

$$SNR_{máx}\left[dB\right] = 6.02 \cdot N + 1.76 + 20 \cdot \log_{10}\left(\frac{k}{2}\right) \qquad \text{Ec. 3.7}$$

Lo que nos lleva a las mejoras de la Tabla 3.2.

Factor de oversampling	Mejora del SNR [dB]	Resolución extra [bits]
2		0
4	6	1
8	12	2
16	18	3
32	24	4
64	30	5
128	36	6
256	42	7
512	48	8
1024	54	9
2048	60	10
4096	66	11

Tabla 3.2: mejora en la resolución al utilizar el oversampling añadiendo una señal triangular.

Como vemos hay una mejora de 6dB (1bit) por cada vez que duplicamos la frecuencia de muestreo. La mejora es superior que la del primer método, pero sin embargo, requiere que la señal de entrada no esté correlacionada con la señal de triangular. Así se ha de generar la señal triangular suficientemente precisa, lo cual puede ser relativamente complicado.

4 Aplicación del oversampling a la conversión resolver-to-digital

4.1 Descripción del método recomendado por Texas Instruments

En la conversión analógico-digital de las señales del resolver, se usa la primera de las dos técnicas de oversampling descritas, es decir se añade ruido blanco. Este ruido en realidad no se añade, ya que se confía en el ruido de la señal analógica introducida por todo tipo de perturbaciones externas. Esta elección se hace por las características de la señal de entrada y por la facilidad del método

El esquema de funcionamiento del método recomendado por Martin Staebler [7] es el de la Ilustración 4.1.

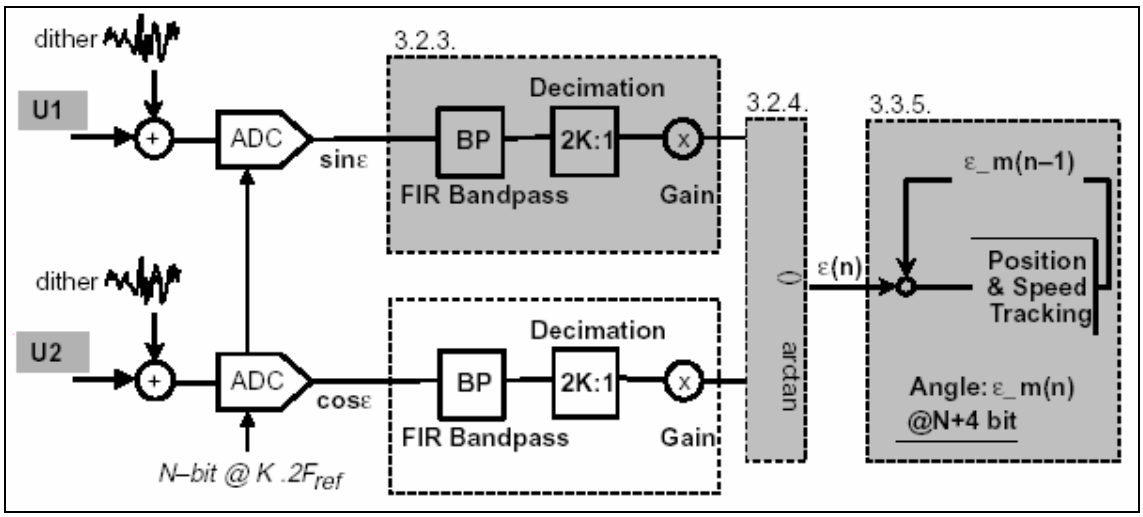

Ilustración 4.1: esquema del método de oversampling propuesto por Texas Instruments para la conversión resolver-to-digital.

En el primer paso, a ambas señales de entrada se le añade ruido. En el paso siguiente se muestrea las señales con N bits a una frecuencia de $2k \cdot f_{ref}$ (factor de sobre-muestreo de 2k).

Después se las pasa por un filtro FIR simétrico pasa-banda que deja pasar las frecuencia en la banda que va desde f_{ref}-$f_{ref}/2$ hasta f_{ref}+$f_{ref}/2$. Este filtro actúa como un filtro anti-alias digital con tal de que se cumpla el criterio de Nyquist al hacer el diezmado, ya que el diezmado se puede considerar como un nuevo muestreo, a una tasa de muestreo de f_{ref} muestras por segundo en nuestro caso.

La mejora en resolución se consigue en esta etapa del proceso gracias a que este filtro actúa como promediador. Como se indica en la Tabla 3.1 se puede conseguir hasta 0.5bits de mejora cada vez que se dobla la frecuencia de muestreo.

Por otro lado, a causa del filtro FIR se introduce un retardo de grupo constante y no deseado, que se tendrá que subsanar más adelante. Con un filtro adecuado se consigue que el retardo de grupo sea exactamente de un periodo de la frecuencia de referencia ($1/f_{ref}$). Un filtro FIR de M+1 coeficientes (orden M) tiene un retardo de M/2 muestras. Si escogemos un filtro de orden 4k, es decir de (4k+1) coeficientes, tenemos un retardo de (4k/2) muestras de la frecuencia ($2k \cdot f_{ref}$), esto significa un retardo de $(4k/2) \cdot 1/(2k \cdot f_{ref}) = (1/f_{ref}) = T_{ref}$, lo cual es exactamente un periodo de la frecuencia de referencia.

Seguidamente el diezmado toma una muestra cada 2k muestras, reduciendo la frecuencia a f_{ref} de nuevo. Así se reduce el espectro a la banda que va de 0 a $f_{ref}/2$.

En este punto las señales se pueden dividir y aplicando el arcotangente obtener el ángulo mecánico como se ha visto en la Ec. 3.2.

4.1.1 Problemas del método general

Este sistema tiene diversos problemas:

- El problema principal de esta técnica viene dada por el hecho de que la señal analógica que introducimos al resolver se produce digitalmente con el DSP y por ello tiene que ser filtrada y también amplificada a valores adecuados para el resolver (unos 10V RMS en nuestro caso). Por otro lado las señales de salida del resolver tendrán que volver a pasar por unos amplificadores operacionales para adaptar su amplitud a valores admitidos por el DSP (0-3.3V). Todos estos procesos de filtrado y tratamiento de la señal producen un retardo importante en la señal, que arrastrará hasta el momento de hacer el diezmado. Como se ha comentado anteriormente este diezmado se comporta como otro muestreo, y como la señal a muestrear es senoidal, sigue teniendo importancia que se haga el muestreo de la señal en su punto máximo para conseguir la máxima precisión.

- Como ya se ha indicado antes, al filtrar con el filtro FIR obtenemos un retardo de grupo. Este se tendrá que corregir para permitir el control del motor en tiempo real. El sistema propuesto por Martin Staebler también propone una corrección de este retardo.

- La función división y arcotangente son funciones que requieren mucho tiempo cálculo, por lo que es preferible evitarlas.

4.2 Método de oversampling mejorado

Para corregir todos estos problemas que se producen en la etapa inicial (la del sobre-muestreo) y además integrar un método que obtiene la velocidad, se ha ideado un nuevo sistema que se basa en el esquema de la Ilustración 4.2. A partir de ahora por comodidad se designará la frecuencia de referencia (o excitación) con f_e y su pulsación con ω_e, y la frecuencia y pulsación mecánica con f_m y ω_m respectivamente.

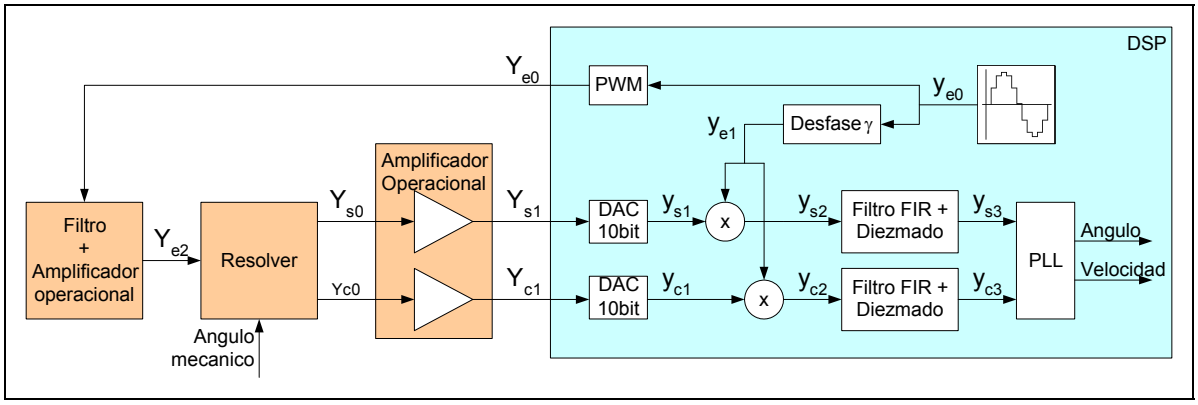

Ilustración 4.2: Esquema del método de oversampling mejorado para la conversión resolver-to-digital.

Como se ve en la Ilustración 4.2, con el DSP se produce digitalmente los valores de la señal de excitación y_e a una frecuencia de $(2k \cdot f_e)$. Para poder utilizar esta señal digital en el resolver se tendrá que convertir analógica. Así se sintetiza y_{e0} en una señal PWM (Y_{e0}), que tendrá que pasar por un filtro analógico. Este filtro elimina las componentes de alta frecuencia, con tal de tener una señal lo más senoidal posible en la entrada del resolver. La conmutación de la señal PWM se hará a una frecuencia de $k \cdot f_e$, es decir a la mitad de la frecuencia con que se genera y_e, esto se debe a que esta disminución en la frecuencia de conmutación no empeora la señal una vez filtrada, pero simplifica considerablemente el algoritmo. La señal Y_{e0} también se tendrá que amplificar a valores de tensión adecuados para el resolver.

A la salida del resolver se tendrá la señal Y_{s0} e Y_{c0} , que se hacen pasar por unos amplificadores operacionales para adaptar la señal a valores adecuados para ADC del DSP, con ello se obtiene las señales Y_{s1} e Y_{c1}. A estas señales se les tendría que añadir ruido blanco antes de muestrearlas, pero el ruido que hay en las señales se puede suponer

aproximadamente como ruido blanco, por lo tanto ya es suficiente para que funcione el oversampling. El muestreo se hará a una frecuencia de ($2k \cdot f_e$).

Una vez digitalizadas las señales con el ADC (y_{s1}, y_{c1}), se multiplican ambas por la señal senoidal de excitación del resolver con un desfase γ (y_{e1}). Es por esto que la señal de excitación se produce a la frecuencia $2k \cdot f_e$. Con esta multiplicación se consigue que la banda de frecuencias de la señal se desdoble como muestra la Ilustración 4.3.

En el último paso se filtra y diezma por un factor de 2k, con lo que la frecuencia a la salida es de f_e nuevamente. Estos dos procesos se hacen en un solo paso para ahorrar cálculos, ya que el diezmado consiste en coger una de cada 2k muestras generadas por el filtro, las restantes no hacen falta ni calcularlas. Así una vez filtradas y diezmadas las señales y_{s3} e y_{c3} corresponden al seno y coseno respectivamente del ángulo mecánico girado por el motor.

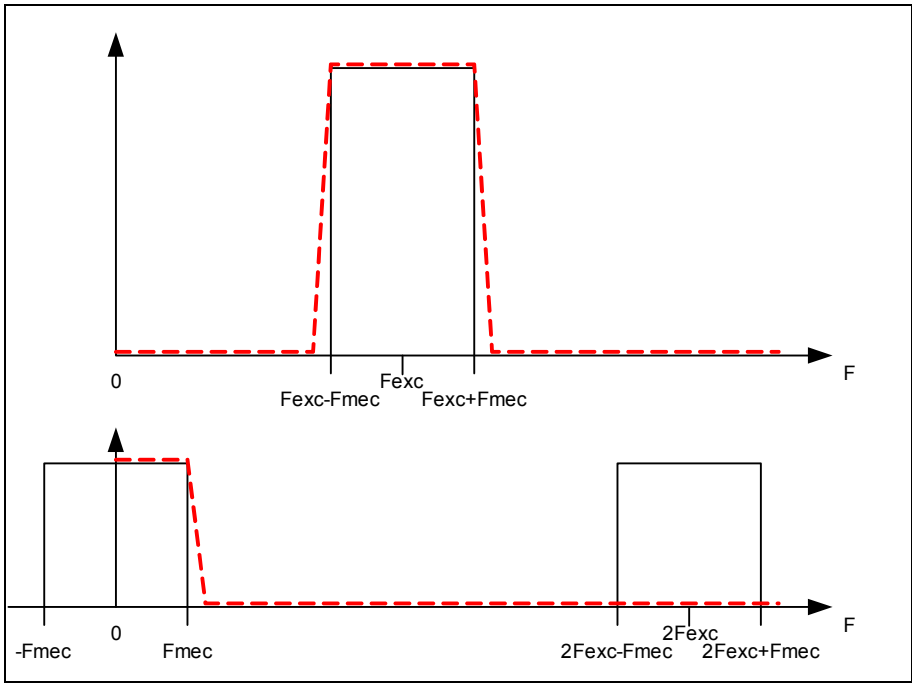

Ilustración 4.3: desdoblamiento de de la banda de frecuencias al multiplicar por y_{e1}.

La explicación de todo esto es la siguiente: A la salida del resolver las señales tendrán las componentes frecuenciales dentro de la banda $f_e \pm f_m$, siendo f_m la frecuencia mecánica del motor. Si ahora se multiplica estas señales por la señal de excitación (digital) se tendrá unas nuevas componentes frecuenciales en $0 \pm f_m$ y en $2f_e \pm f_m$. Se puede ver el desarrollo matemático para la señal seno en la Ec. 4.1 (para el coseno el proceso es idéntico).

$$\begin{aligned}
y_{e0} &= \sin(\omega_e t) \\
y_{e1} &= \sin(\omega_e t - \gamma) \\
Y_{e2} &= \sin(\omega_e t - \alpha) \\
Y_{s0} &= \sin(\omega_e t - \alpha)\sin(\omega_m t) \\
y_{s1} &= \sin(\omega_e t - \alpha - \beta)\sin(\omega_m t - \beta) \\
y_{s2} &= y_{s1} \cdot y_{e1} = \sin(\omega_e t - \alpha - \beta)\sin(\omega_m t - \beta)\sin(\omega_e t - \gamma)
\end{aligned} \qquad \text{Ec. 4.1}$$

Siendo α el retraso que hay desde el DSP hasta la entrada del resolver, y β el que hay desde el resolver hasta la entrada del DSP. Desarrollando esta fórmula con el programa matemático MAPLE se obtiene:

$$\begin{aligned}
y_{s2} = &\frac{1}{4}\sin\left((2\omega_e - \omega_m)t - \alpha - \gamma\right) - \frac{1}{4}\sin\left((2\omega_e + \omega_m)t - \alpha - 2\beta - \gamma\right) + \\
&+\frac{1}{4}\sin(\omega_m t + \alpha - \gamma) + \frac{1}{4}\sin(\omega_m t - \alpha - 2\beta + \gamma)
\end{aligned} \qquad \text{Ec. 4.2}$$

En la Ilustración 4.3 se puede ver las bandas de frecuencias que se forman. Si ahora se eliminan las componentes de alta frecuencia (las componentes en la banda $2f_e \pm f_m$) con un filtro, se obtiene y_{s3}:

$$y_{s3} = \frac{1}{4}\left[\sin(\omega_m t + \alpha - \gamma) + \sin(\omega_m t - \alpha - 2\beta + \gamma)\right] \qquad \text{Ec. 4.3}$$

Se demuestra que esta señal es una señal senoidal con un desfase A (función de de α y β) y una amplitud sin(B) (donde B es función de α, β, γ) según la Ec. 4.4. Así existe un valor de $\gamma_{óptima}$ que maximiza la amplitud de la señal senoidal de manera que esta tenga amplitud máxima de 1/2. Este valor de γ se encuentra numéricamente en función de los retrasos α y β medidos.

$$\begin{aligned}
y_{s3} &= \frac{1}{2}\sin(\omega_m t + \mathrm{A})\sin(\mathrm{B}) = \frac{1}{2}\sin(\varphi_m(t) + \mathrm{A})\sin(\mathrm{B}) \\
y_{s3}(\gamma_{óptima}) &= \frac{1}{2}\sin(\varphi_m(t) + \mathrm{A})
\end{aligned} \qquad \text{Ec. 4.4}$$

La ventaja de este método reside en el hecho de que ahora la señal ya no varía a la frecuencia f_e, sino solamente a la frecuencia mecánica f_m. Así ahora ya no es importante el instante en que se diezma. En cambio el valor de γ que se retrasa y_{e1}, sí que depende del retraso de las señales, pero esto es más fácil de integrar en el algoritmo del DSP. Por otro

lado se podría pensar que conviene que la amplitud sea máxima en el momento de diezmar y que por tanto el factor 1/2 por el que está multiplicado las señales produzca una perdida de resolución, pero esto no es así, ya que todos estos cálculos se realizan digitalmente, y con precisiones mayores a las de las señales.

El filtro utilizado para esto es un filtro pasa-bajos en vez de pasa-banda. Como en el sistema recomendado, el orden del filtro se pone de acuerdo con el factor de oversampling para producir un desfase exacto de un periodo de la frecuencia de excitación.

4.3 PLL para cálculo del ángulo y de la velocidad

El sistema que se propone para solucionar el retardo de grupo producido por el filtro se basa en una PLL (Phase Loop Lock). El término PLL se usa generalmente para designar un tipo de sistema, eléctrico o algoritmico, que se encarga de mantener constante la fase del ángulo de una señal de entrada oscilante. Así su aplicación más usual es en sistemas donde haya que generar una señal de reloj con una frecuencia constante. Pero también se puede utilizar con muchos otros objetivos.

En este proyecto se utilizará para obtener el ángulo, así, en vez de aplicar la función arcotangente, se aplica el sistema que se explica a continuación.

A la entrada de la PLL se tendrá las señales y_{s3} e y_{c3} con varios retardos: el más importante en principio es el retardo de un periodo $T_e=1/f_e$ producido por el filtro FIR digital. Por otra parte se tiene los retardos producidos por los diversos elementos analógicos.

$$\begin{aligned} y_{s3} &= \sin\left(\varphi_m(t)+\Delta\right) = y_Q \\ y_{c3} &= \cos\left(\varphi_m(t)+\Delta\right) = y_D \end{aligned} \qquad \text{Ec. 4.5}$$

Si estos valores se representan en coordenadas D-Q representan un vector de modulo constante que gira a velocidad ω_m alrededor del origen de coordenadas D-Q. Siendo:

$$\omega_m = \frac{d\varphi_m}{dt} \qquad \text{Ec. 4.6}$$

Si ahora se representa este vector en unas nuevas coordenadas d-q giradas un ángulo φ'_m respecto a las coordenadas D-Q, se tendrá su expresión en estas nuevas coordenadas según:

$$
\begin{aligned}
y_D &= y_d \cos(\varphi'_m) - y_q \sin(\varphi'_m) \\
y_Q &= y_d \sin(\varphi'_m) + y_q \cos(\varphi'_m) \\
\begin{bmatrix} y_D \\ y_Q \end{bmatrix} &= \begin{bmatrix} \cos(\varphi'_m) & -\sin(\varphi'_m) \\ \sin(\varphi'_m) & \cos(\varphi'_m) \end{bmatrix} \begin{bmatrix} y_d \\ y_q \end{bmatrix} \\
\begin{bmatrix} y_d \\ y_q \end{bmatrix} &= \begin{bmatrix} \cos(\varphi'_m) & \sin(\varphi'_m) \\ -\sin(\varphi'_m) & \cos(\varphi'_m) \end{bmatrix} \begin{bmatrix} y_D \\ y_Q \end{bmatrix} \\
y_d &= y_D \cos(\varphi'_m) + y_Q \sin(\varphi'_m) = -\cos(\varphi_m)\sin(\varphi'_m) + \sin(\varphi_m)\cos(\varphi'_m) \\
y_q &= -y_D \sin(\varphi'_m) + y_Q \cos(\varphi'_m) = -\cos(\varphi_m)\sin(\varphi'_m) + \sin(\varphi_m)\cos(\varphi'_m)
\end{aligned}
\qquad \text{Ec. 4.7}
$$

Si ahora variando φ'_m se minimiza y_q, se consigue que la nueva base (giratoria) esté en fase con el ángulo mecánico, y por tanto se cumplirá: $\varphi'_m=\varphi_m$. Es decir que el valor de φ'_m es igual al ángulo mecánico del motor. Todo esto se puede ver en la Ilustración 4.4.

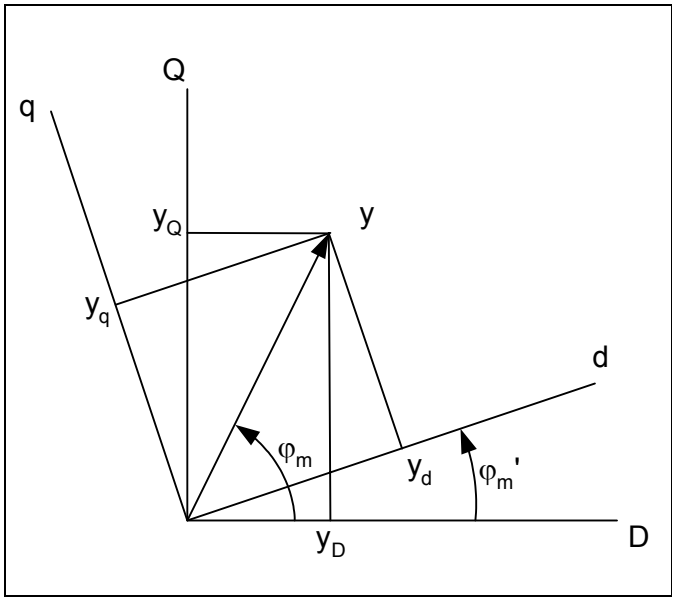

Ilustración 4.4: cambio de las coordenadas D-Q a las coordenadas giratorias d-q.

Por tanto lo que se pretende es minimizar y_q. Para ello se introduce un controlador PID que se ajustará para conseguir la velocidad, esta se integrará seguidamente para conseguir la posición, tal y como se observa en la Ilustración 4.5.

A la realimentación con φ_m' se le aplica un retardo de un ciclo. Así, cuando se aplique el giro de ángulo φ'_m a y_D e y_Q (recordemos que tienen un retardo de un ciclo) se obtendrá el error (retardado). Con esto se obliga a que el error pasado (retardado) se anule en el momento actual, con lo que se consigue que la señal teóricamente se adelante un ciclo. Esto solo ocurrirá de manera perfecta cuando se esté en estado estacionario con velocidad constante.

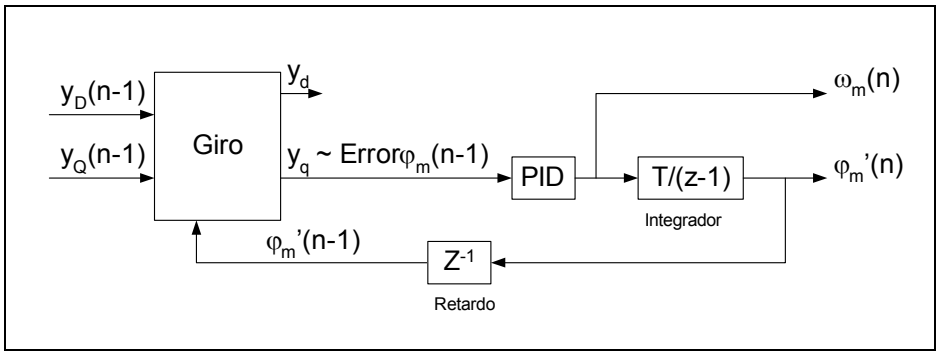

Ilustración 4.5: esquema de bloques de la PLL.

4.4 Requerimientos del DSP

4.4.1 Elección del DSP a utilizar

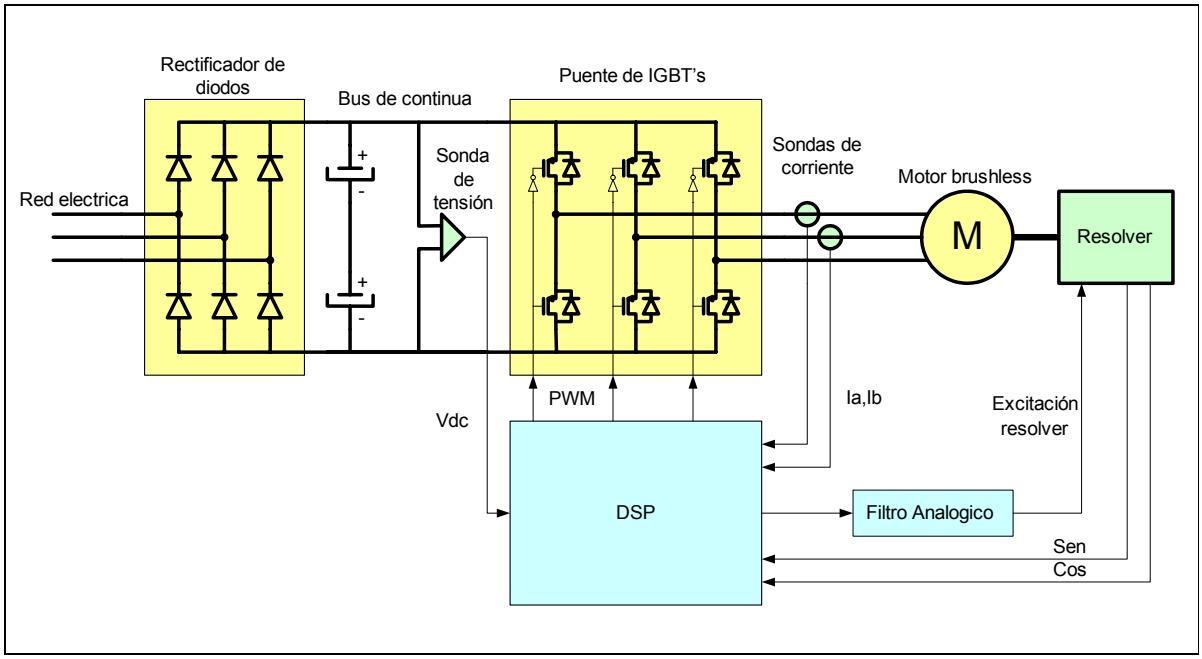

Ilustración 4.6: esquema del sistema completo.

Como se puede ver en la Ilustración 4.6, el DSP necesitará las siguientes entradas y salidas:

- 2 entradas analógicas para las señales procedentes del resolver.
- 1 Salida Compare para excitar el resolver con una señal PWM.
- 3 salidas PWM para el control de los IGBT's que ondulan la tensión que se aplicará al motor.

- 2 entradas analógicas para sensores de corrientes que consume el motor,.
- 1 entrada analógica para un sensor de tensión del bus de continua. Esta tensión es necesaria para la seguridad del sistema y para independizar el control de corriente de la tensión del bus. Así podremos actuar debidamente si la tensión de bus sube o baja.

La marca Texas Instruments dispone de una familia de DSP de coma fija, de bajo consumo, bajo coste y altas prestaciones con ciertos periféricos especialmente diseñadas para el control de motores, convertidores de potencia y accionamientos, todo ello integrado en un único chip, se trata de la familia 240x. La nueva gama de esta familia, la 240xA, ofrece 40 MIPS (millones de instrucciones por segundo) y una mayor integración de los periféricos. Los diversos tipos de DSP de la familia se diferencian principalmente por:

- Cantidad y tipo de memoria
- Número de periféricos
- Número de de entradas/salidas digitales
- Número de entradas analógicas
- Número de patillas del encapsulado

A mayor número de entradas y salidas aumenta el número de pines del encapsulado, lo que dificulta tanto el diseño de la placa para de control, como el montaje de la misma, y sobretodo, el soldado del DSP sobre la placa. Por todo esto, y por el aumento de precio, conviene elegir un modelo que se ajuste lo máximo posible a las necesidades. El tipo de memoria para el almacenamiento del programa es memoria Flash, ya que se trata de un prototipo.

Así se ha escogido el 2403A (concretamente el TMS320LF2403A) como la solución adecuada. Las principales características son se pueden ver en el anejo E.2.

4.4.2 Elección de la mejora en la resolución que se quiere conseguir

El tiempo de cálculo requerido por el DSP para hacer el oversampling tiene una relación exponencial con la mejora en resolución que se quiera conseguir. La frecuencia con la que se tiene que adquirir y guardar muestras de la señal seno y coseno (f_{ovs}), viene en función de la mejora de resolución en bits que se quiera conseguir (b) según la formula:

$$f_{ovs} = 2 \cdot \left(2^{\frac{b}{0.5}} \right) f_s \qquad \text{Ec. 4.8}$$

Para hacer la elección de la mejora en resolución deseada se tiene que tener en cuenta el tiempo de cálculo máximo del que se dispone para el oversampling, ya que el DSP estará haciendo otras actividades a la vez. Entre estas actividades está el control de posición, el control de velocidad, el control de corriente, el algoritmo del SVPWM, un sistema de monitorización de variables, un sistema seguridad del sistema y un algoritmo de control global.

Lo primero que se ha hecho es elegir la frecuencia de conmutación de los IGBT y la frecuencia con la que actúa el control de corriente y el algoritmo SVPWM para variar el ancho de los pulsos. Cuanto más bajo es la frecuencia de conmutación menores son las perdidas por conmutación. Por otro lado, cuanto más baja es la frecuencia de conmutación, más rizado tendrá la corriente de salida del convertidor para una inductancia dada, y además, las frecuencias bajas son audibles para el oído humano. Por eso se ha tomado una frecuencia intermedia de 9000Hz. Esta frecuencia es relativamente poco audible por el oído humano. Para obtener un control de corriente preciso, el control se realiza también a 9000Hz.

El control de la velocidad o de la posición se hace a la mitad de frecuencia (4500Hz), ya que si también se hiciera a 9000Hz no daría tiempo a que el motor adquiera estas velocidades y posiciones. Por lo tanto se necesitarán lecturas de posición y velocidad (adquiridas con oversampling y diezmadas) a una frecuencia de 4500Hz. Para el control de la corriente también hace falta la posición, por lo tanto haría falta una lectura de de posición a 9000Hz, pero en realidad se hace una estimación con la posición y velocidad anterior.

Haciendo una previsión cualitativa de estas y otras posibles necesidades de cálculo adicionales, teniendo en cuenta las simulaciones hechas y contando con la experiencia de varios profesores del departamento de electrotecnia, se ha optado por una mejora de 2bits, es decir un factor de oversampling de k=16, lo cual supone tomar muestras a una frecuencia de 2·16·4500=144000Hz=144kHz. Este valor es teórico y válido si se supone que la señal es senoidal perfecta y se tiene velocidad mecánica prácticamente nula. En realidad se introducen errores sobretodo por los siguientes factores:

- La velocidad no es nula.

- Los componentes analógicos como los filtros, amplificadores analógicos y el resolver no son perfectos por lo tanto introducen ruido, retrasan la señal y la deforman.

- El ADC introduce ruidos al muestrear.

- Los filtros FIR digitales son filtros finitos, por tanto no filtran perfectamente, e introducen retrasos.

- En general todas los cálculos digitales pueden introducir errores, pero al trabajar con 16 bits de resolución, estos cálculos no introducen ningún error que podamos notar en señales que tienen una calidad de 10 bits o 12 bits, se tendría que hacer un número muy elevado de operaciones seguidas para que afecte a la calidad de 10 bits o 12 bits en caso de una mejora extrema.

- Dentro de la PLL también hay elementos, como el PID y el integrador que pueden introducir errores, dependiendo de los parámetros y métodos utilizados.

Los ADC's del DSP son de 10 bits, con esto se consiguen 1024 posiciones diferentes, en condiciones teóricas ideales. Aunque en realidad nunca se llega a estos valores por culpa de los diversos ruidos e inexactitudes. Para medir la resolución real efectiva del ADC se ha hecho un pequeño programa que simplemente adquiere los valores a una frecuencia de 144kHz. Como valor de entrada se ha puesto una tensión constante nula y se han muestreado 1000 valores:

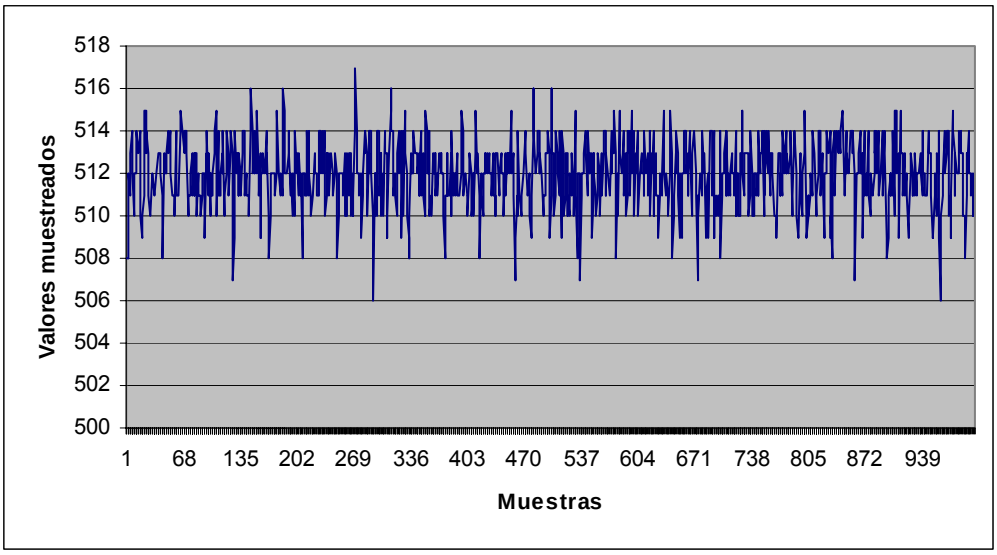

Ilustración 4.7: Valores muestreados al poner una tensión constante nula a la entrada del DAC sobre fondo de escala de 10bits (1024 posiciones).

Se puede ver que por un lado hay un error estacionario, ya que la media no es exactamente 512 sino 511.975, y por otro lado hay un error de dispersión, que se puede considerar aleatorio. Si se supone una distribución normal, el error de dispersión queda representado por la desviación estándar que vale 1.597. Esta desviación estándar, viene a dar una idea del valor medio del error de dispersión.

Se puede ver que el error permanente (512-511.975=0.025) es mucho menor que el error de dispersión (1.597), por tanto se puede despreciar. Así, si se tiene en cuenta solo el error de dispersión, se obtendrá de media 1024/(2·1.597)=320.6 posiciones diferentes, lo cual representa una resolución de aproximadamente 8.3 bits (ver Ec. 4.9), en vez de los 10 que se tendrían que obtener idealmente. En realidad, si se toma un rango que esté entre el valor medio ± la desviación estándar, solo se tendrá el 68% de las mediciones, pero dicha desviación estándar representa el valor medio del error.

$$\begin{aligned} 2^b &= 320.6 \\ b \cdot \ln 2 &= \ln 320.6 \\ b &= \frac{\ln 320.6}{\ln 2} = 8.3 bits \end{aligned} \qquad \text{Ec. 4.9}$$

Por otro lado se sabe a través de otros compañeros del departamento de electrotecnia que aplicando la técnica del undersampling para obtener el Angulo girado, se obtiene una resolución efectiva del Angulo que cualitativamente está alrededor de los 7 a 8 bits, que corrobora de forma aproximada lo observado. Con esto se plasma el hecho de que la propia técnica del undersampling no es muy precisa, sobretodo por el hecho de que solo se tiene un valor, y este puede estar bastante lejos del valor real, en cambio con el oversampling siempre se tiene un valor promediado y por lo tanto más exacto. Es por esto que es conveniente aplicar una técnica promediadora como el oversampling.

Si hablamos del ángulo girado por el motor, con 10bits se obtendría una resolución de 360º/1024=0.3516º=21’56’’. Según el fabricante del motor, el resolver tiene un error eléctrico máximo de ±10’, con lo que el valor real siempre estará dentro de una banda de 20’ de ancho. Por ello no tiene mucho sentido mejorar la resolución mucho más allá de los 20’.

Con todo lo anterior se confirma que una mejora en la resolución de 2 bits es adecuada para este caso.

5 Simulaciones

5.1 Descripción

Para tener una aproximación del funcionamiento de la técnica del oversampling y de la mejora que produce se han hecho diversas simulaciones con el programa de cálculo numérico MATLAB 6.5 R13 y el paquete de simulación SIMULINK.

El esquema general del sistema que se ha usado para simular es el de la Ilustración 5.1. En azul están los elementos que funcionan en tiempo continuo, es decir aquellos que son analógicos, y en naranja, aquellos que suceden en tiempo discreto, es decir aquellos que se implementan digitalmente. En blanco están los elementos auxiliares que sirven para observar y comparar resultados dentro de la propia simulación, estos elementos son aquellos que no intervienen en el sistema real.

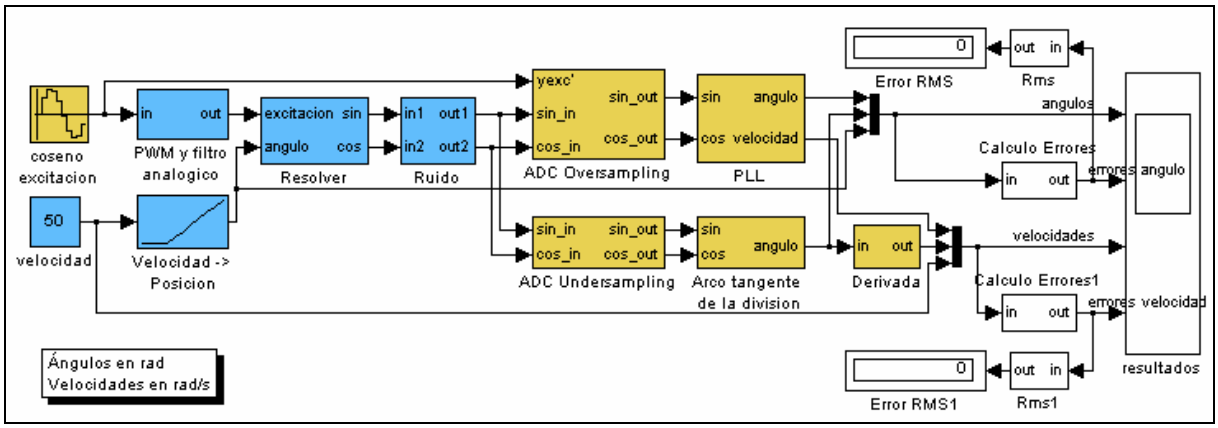

Ilustración 5.1: esquema general de la simulacion usada.

Se puede ver que se ha simulado por un lado el sistema propuesto: la adquisición de las señales seno y coseno con oversampling, y el cálculo del ángulo y la velocidad con la PLL (Para una explicación del sistema ver los capítulos 4.2 y 4.3). Por otro lado se ha simulado la adquisición de las señales seno y coseno con undersampling, y el cálculo del ángulo mediante el arcotangente de la división y la velocidad derivando la posición (Para una explicación del funcionamiento de este sistema ver el capítulo 3.3). También se tiene los errores del ángulo calculado con ambos sistemas respecto al ángulo real para poder comparar.

En el primer bloque se genera un seno en tiempo discreto, a alta frecuencia (2·16·4500Hz=144000Hz). En el bloque “PWM y filtro analógico” se simula el paso de digital a analógico que sufre la señal de excitación del resolver. Contiene el filtro analógico que se

pondrá para hacer la señal de excitación lo más senoidal posible. Este filtro es el que introducirá el mayor retraso, pero para simular otros posibles retrasos que pueda haber en el camino de la señal hasta retornar al DSP se permite introducir un retraso adicional.

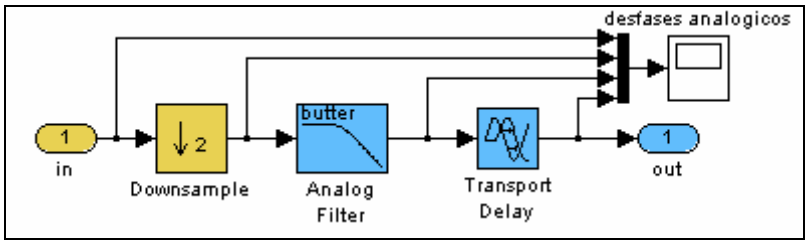

Ilustración 5.2: detalle del bloque "PWM y filtro analógico".

El bloque "Resolver" no requiere explicaciones adicionales, para más información ver el capítulo 3.2.

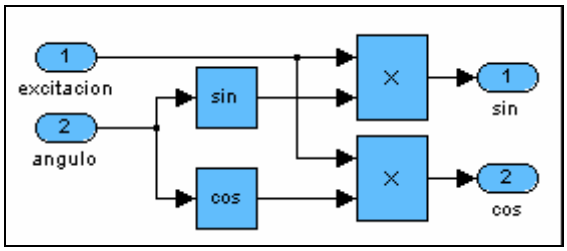

Ilustración 5.3: detalle del bloque "Resolver".

En la realidad toda la parte analógica introduce diversos ruidos, para tenerlos en cuenta se introducen en la simulación mediante el bloque "Ruido". Este ruido se supone blanco, y para que coincida lo máximo con la realidad se ajusta hasta que su desviación estándar coincida con la desviación estándar medida en la prueba del capitulo 4.4.2. Así, si sobre 512 la desviación estándar es de 1.597, sobre 1 (semiamplitud de la senoide simulada) ha de ser 0.00312 aproximadamente. Los ruidos que se añaden a ambas señales tienen las mismas características pero son diferentes; utilizan semillas diferentes para generar los valores aleatorios.

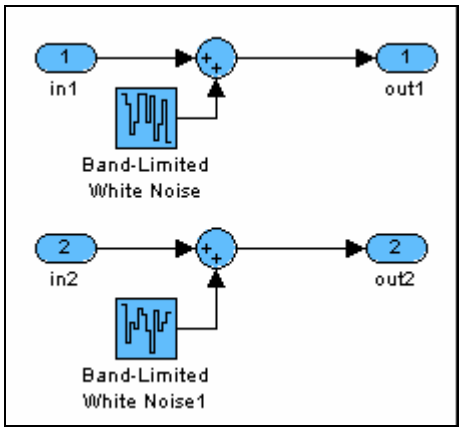

Ilustración 5.4: detalle del bloque "Ruido".

El bloque que simula el oversampling, integra tanto los elementos que simulan el ADC (conversor analógico a digital) como los filtros y diezmadores del oversampling. El ADC está formado por el mantenedor de orden cero y el cuantizador de 10 bits. Después los valores obtenidos con el ADC se multiplican por la señal de excitación debidamente retrasada. Lo siguiente en la cadena, son unos bloques del MATLAB que integra en uno solo, un filtro FIR y el diezmado. Con esto se obtiene la señal demodulada, es decir, los valores correspondientes al seno y al coseno del ángulo mecánico (a una frecuencia f_{exc}).

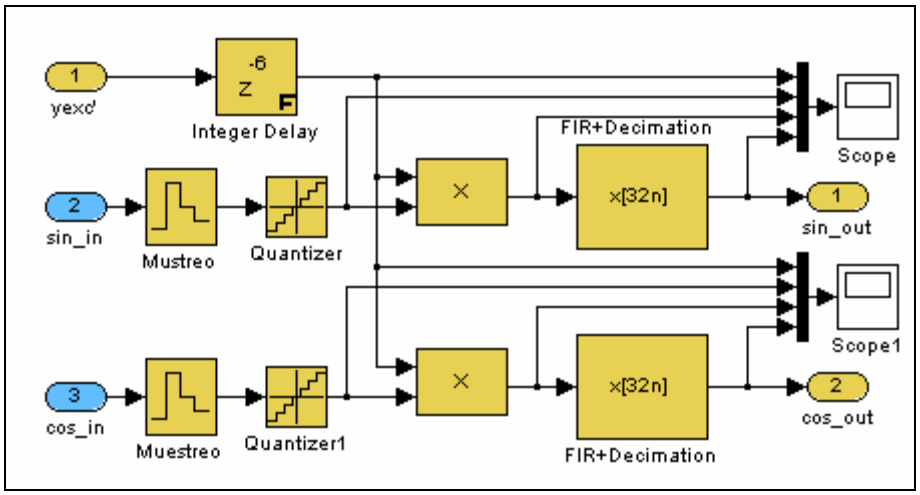

Ilustración 5.5: detalle del bloque “ADC Oversampling”.

Para hallar tanto el ángulo mecánico de este seno y coseno, como para hallar la velocidad se aplica la PLL. En esta se aplica el sistema en lazo cerrado de la Ilustración 5.6. El funcionamiento está comentado en capítulo 4.3.

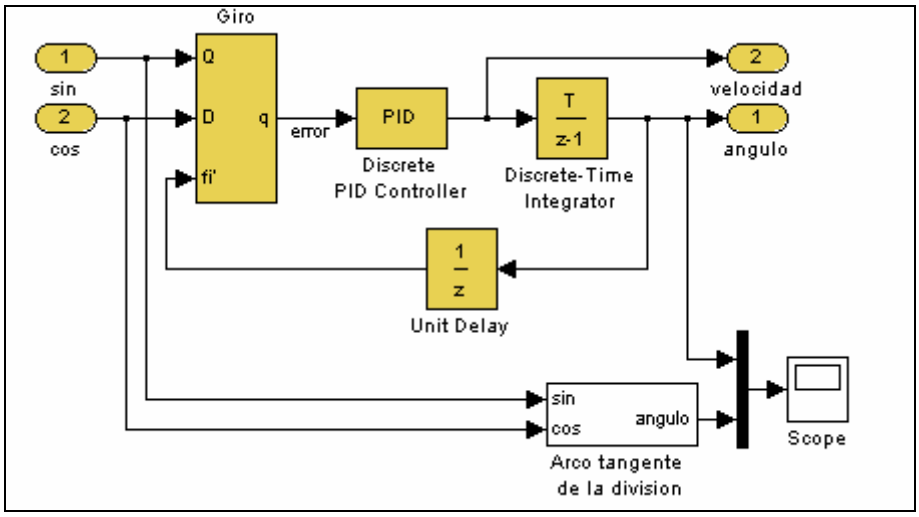

Ilustración 5.6: detalle del bloque "PLL".

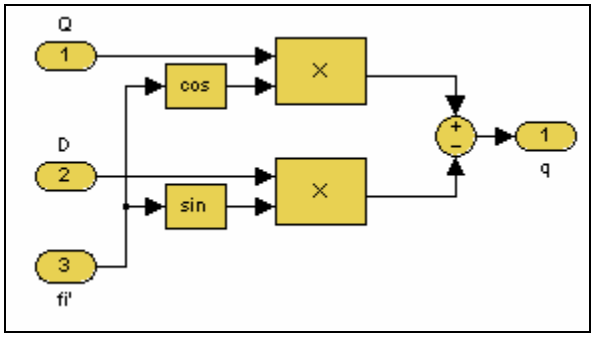

Ilustración 5.7: detalle del bloque "Giro".

Por otro lado se simula el método del undersampling. Este consta básicamente de la parte correspondiente al ADC: muestreador y cuantizador. Tiene la peculiaridad de que el muestreo se tiene que hacer con un cierto desfase para muestrear en el máximo de la señal portadora. Igual que en el "ADC Oversampling", a la salida se tiene la señal demodulada: los valores correspondientes al seno y al coseno del ángulo mecánico respectivamente a una frecuencia de muestras de f_{exc}. Pero esta vez se aplica la división de ambos valores y el arcotangente de este resultado para hallar el ángulo mecánico (en vez de utilizar la PLL).

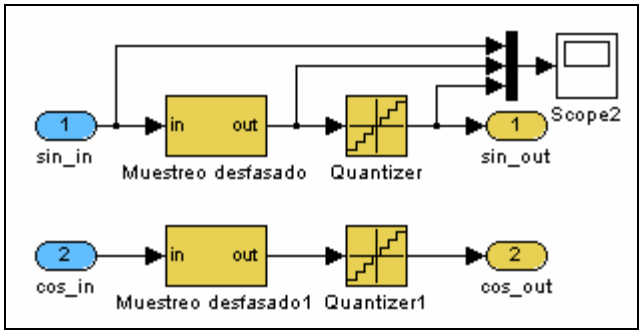

Ilustración 5.8: detalle del bloque "ADC Undersampling".

Con este método se tiene que implementar un derivador aparte para hallar la velocidad, cosa que se ahorra con la PLL.

5.2 Resultados

Con todo esto hay que decir que los elementos que podemos variar para aumentar la mejora de resolución son los siguientes:

- El tipo y parámetros del filtro FIR. Después de diversas pruebas se ha elegido un filtro FIR pasabajos creado con una ventana tipo Chebyshev con una frecuencia de corte de 200Hz. Esta frecuencia de corte tiene cierto margen para permitir el paso a la frecuencia máxima que se van a producir. Esta frecuencia máxima es de 166Hz que corresponde a un motor girando a 10000rpm. No es normal encontrar motores eléctricos en el mercado, que permitan superar este límite.

- Por otro lado se pueden variar los parámetros del controlador PID que hay en la PLL.

Con la mejora introducida por el proceso de oversampling teóricamente y en condiciones ideales se puede conseguir una mejora de medio bit por cada vez que doblamos la frecuencia de sobre-muestreo.

Con el sistema completo montado y ajustado se han hecho simulaciones de 0.02s para diversas velocidades constantes, desde velocidad 0 hasta 1000rad/s (9549.3rpm). Así se ha comparado el sistema con Undersampling y arcotangente, el sistema con Oversampling y PLL y el sistema con Oversampling y arcotangente. Este último se ha hecho para ver la importancia que tiene la PLL frente al Oversampling en la mejora de la resolución del ángulo girado por el motor.

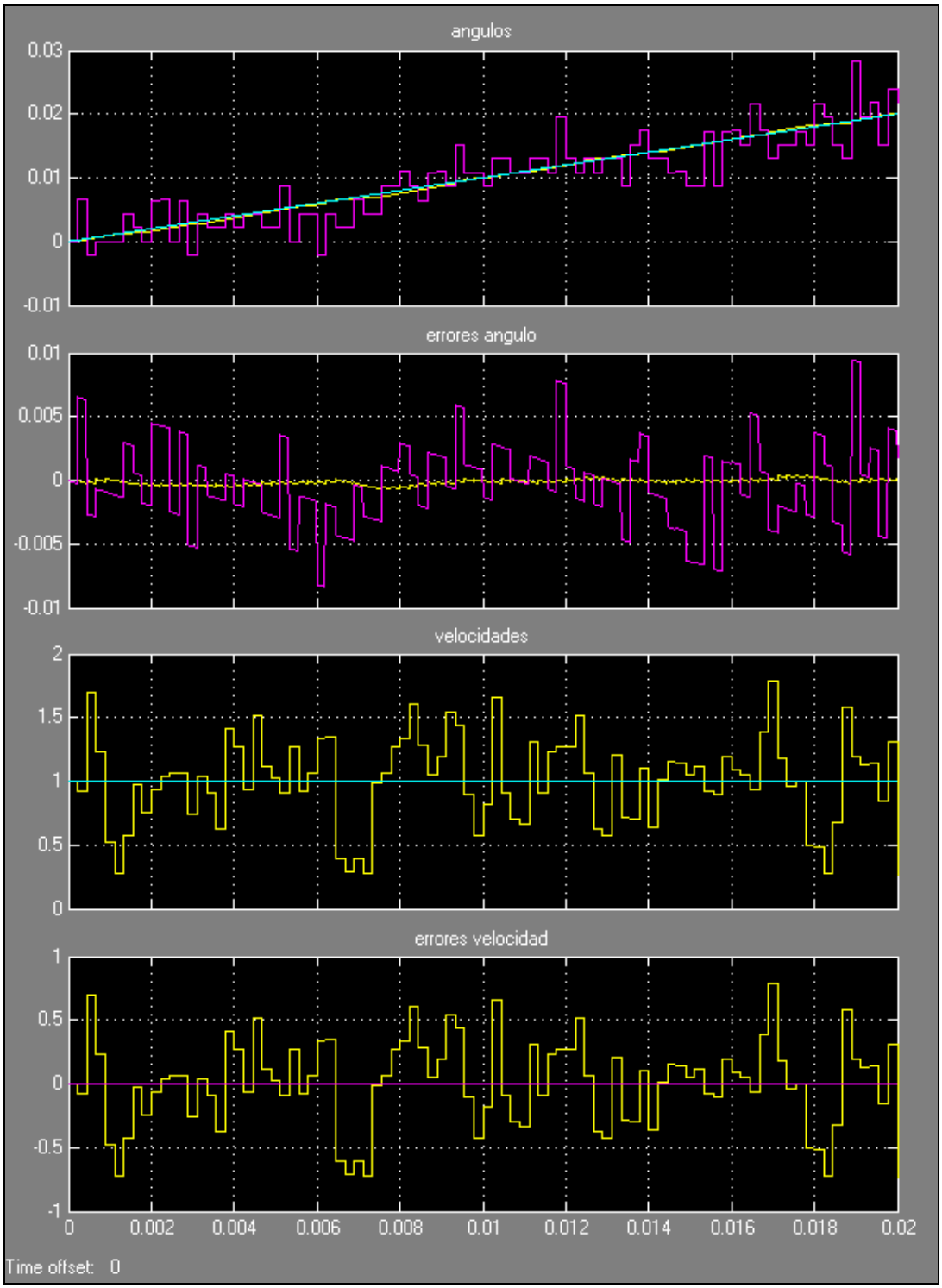

Ilustración 5.9: ejemplo de una grafica de resultados para una simulación con velocidad mecánica constante de 1rad/s. Las curvas en amarillo son del sistema Oversampling + PLL, las lilas para el sistema Undersampling + arcotangente, y las azules para la señal real.

Para evaluar correctamente los resultados se ha de diferenciar entre precisión y resolución cuando se hace una medida:

- Precisión: es la capacidad de tener el error más pequeño respecto al valor real. Para medir la precisión se ha tomado el valor efectivo de este error a lo largo del tiempo. Cuanto mayor sea el valor efectivo del error, menor es la precisión.

- Resolución: es la capacidad de diferenciar el máximo número de valores diferentes sin confundirlos, o lo que es lo mismo, la capacidad de diferenciar porciones lo más pequeñas posibles sin confundirlas. Este concepto está íntimamente ligado con el concepto de dispersión de la señal, por eso, para medirlo se ha tomado la desviación estándar de la señal a lo largo del tiempo. Así, cuanto mayor sea la desviación, menor será la resolución de la medida.

5.2.1 Precisión

La precisión del ángulo (valor efectivo, del error respecto al valor real) se puede ver en la Ilustración 5.10. Se observa que el error en los tres sistemas analizados aumenta con la velocidad. La máxima precisión la tiene el sistema con Oversampling y PLL.

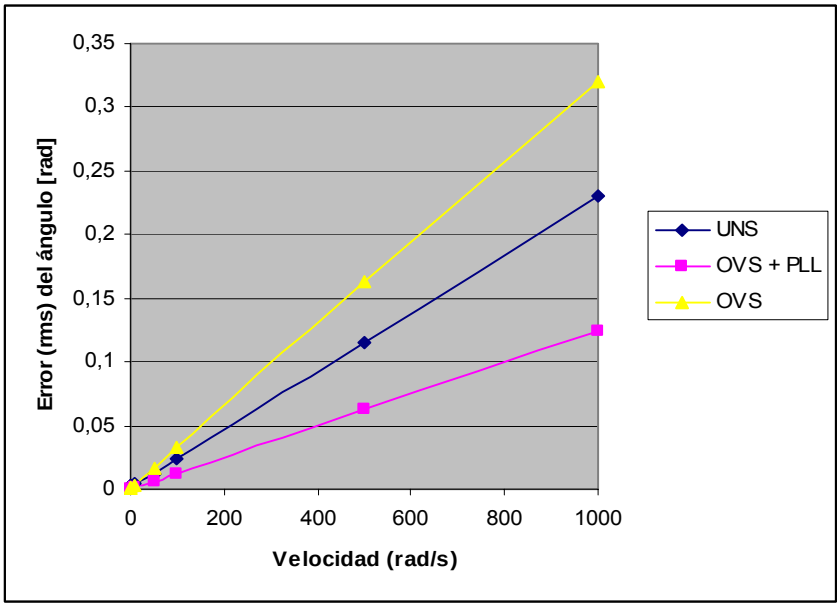

Ilustración 5.10: precisión del ángulo frente a la velocidad mecánica para diversos sistemas.

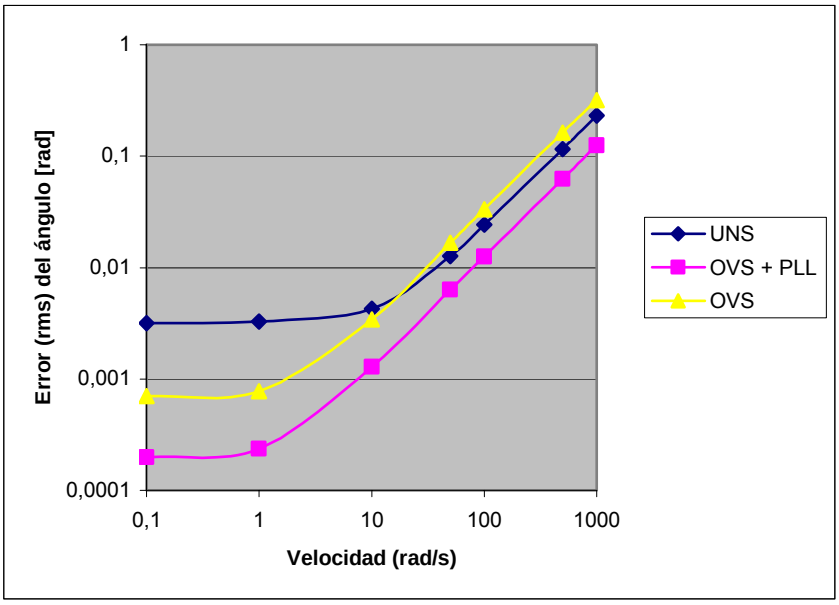

Ilustración 5.11: ilustración anterior en ejes logarítmicos.

En la Ilustración 5.10 puede parecer que el sistema que utiliza Oversampling y el arcotangente sea la de menor precisión, y que por lo tanto el oversampling no produzca ninguna mejora en la resolución, sino que únicamente la PLL produce mejora. Pero si se pones los ejes en escala logarítmica (Ilustración 5.11) se puede apreciar mejor la zona de bajas velocidades, y se observa que para velocidades hasta 15rad/s aproximadamente el Oversampling es claramente mejor que el Undersampling. En combinación con la PLL el oversampling es mejor en todo el rango de velocidades. La explicación reside en el retraso del filtro FIR; a bajas velocidades no es importante, pero a medida que la velocidad aumenta, el retraso (constante) es mayor relativamente y empeora la precisión del oversampling hasta el punto de ser peor que el Undersampling.

5.2.2 Resolución

En lo referente a la resolución, se puede ver la desviación estándar del error para las diversas simulaciones y sistemas en la Ilustración 5.12. Aquí también se aprecia que la desviación aumenta con la velocidad (por lo tanto disminuye la resolución). Y de nuevo se produce el hecho de que a partir de cierta velocidad (por encima de 140rad/s aproximadamente) es mejor el Undersampling que el Oversampling sin PLL.

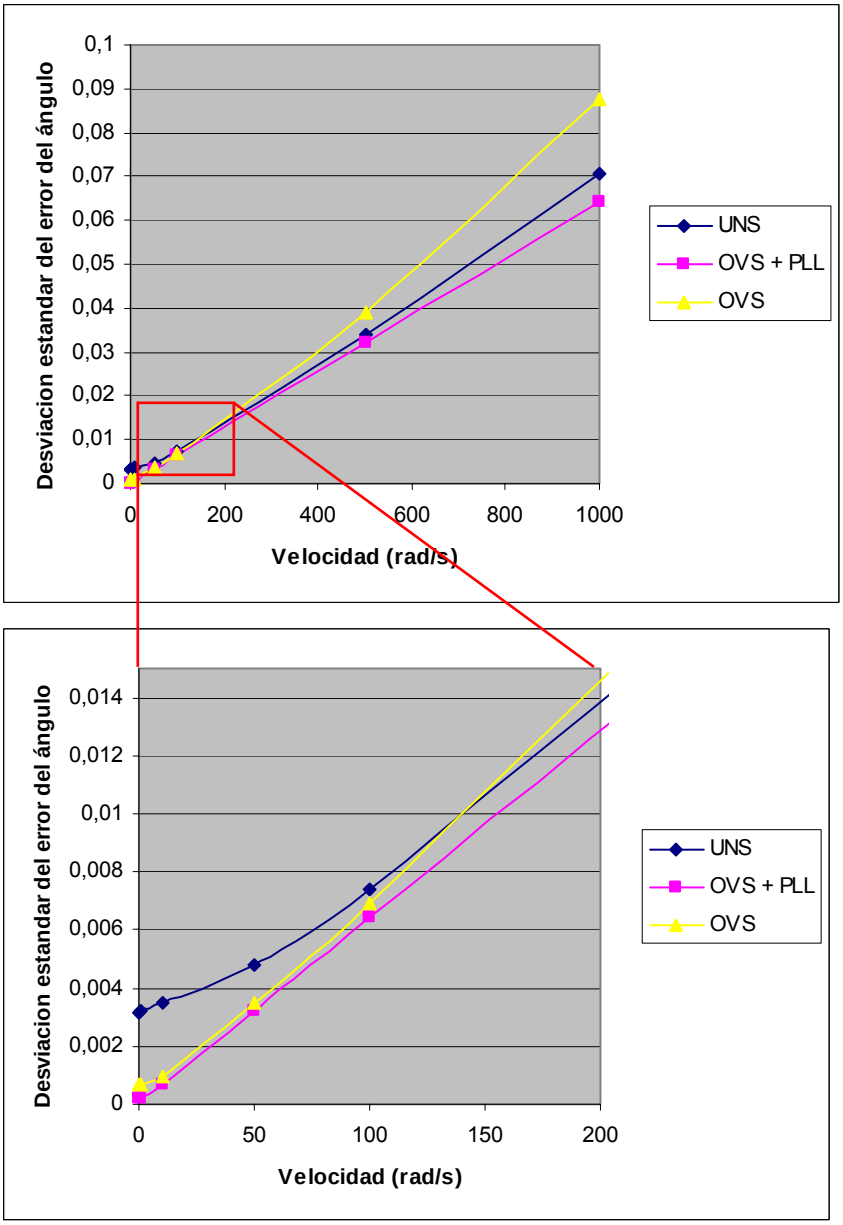

Ilustración 5.12: resolución del ángulo frente a la velocidad mecánica para diversos sistemas y ampliacion de la parte de bajas velocidades.

Para ver la resolución efectiva en número de bits se aplica la siguiente formula:

$$2^b = Número\ de\ posiciones = \frac{Rango}{Resolución\ real}$$

$$b = \frac{\ln\left(\frac{Rango}{Resolución\ real}\right)}{\ln(2)} \qquad \text{Ec. 5.1}$$

Aplicando esta fórmula resulta la gráfica de la Ilustración 5.13. En esta se puede ver que a velocidad prácticamente nula, y en condiciones teóricas, el binomio Oversampling-PLL puede llegar a ofrecer una resolución efectiva de más de 12bits. A partir de velocidades cercanas a los 100rad/s todos los sistemas ofrecen resoluciones muy parecidas que disminuyen todavía más con la velocidad. Esto en realidad no supone un problema importante, ya que a altas velocidades las resoluciones altas no tienen gran interés, ya que generalmente interesa tener una buena precisión en el momento de posicionar algo, es decir; cuando el motor va a bajas velocidades, con lo que el sistema ya es suficientemente válido.

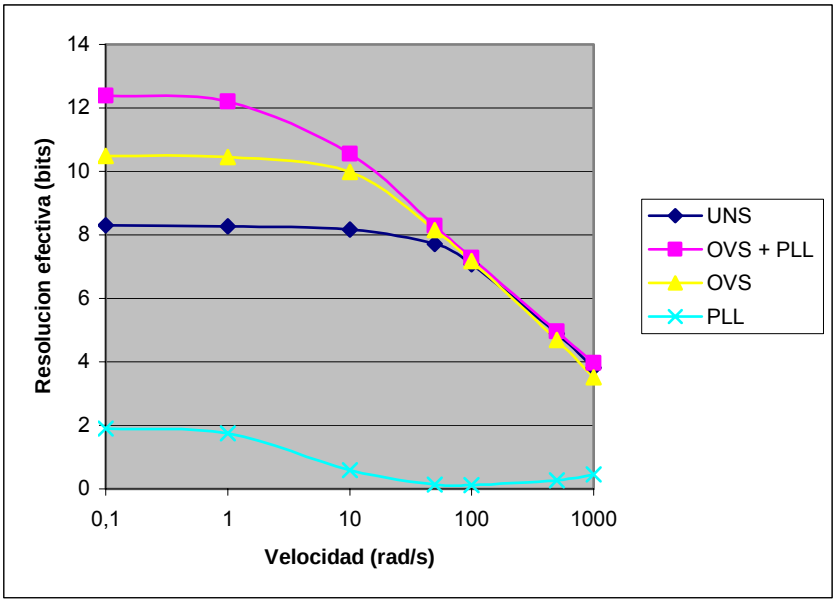

Ilustración 5.13: resolución en bits de los diversos sistemas. La curva azul (PLL) es la diferencia entre la resolución del sistema OVS + PLL frente al sistema OVS.

Por otro lado también se produce un error en la lectura de la velocidad con la PLL, este error parece constante hasta un valor entre 100rad/s y 200rad/s y creciente a partir de este punto. Seguramente el error constante a bajas velocidades, es debido a un ajuste impreciso del los parámetros del controlador PID que incorpora la PLL.

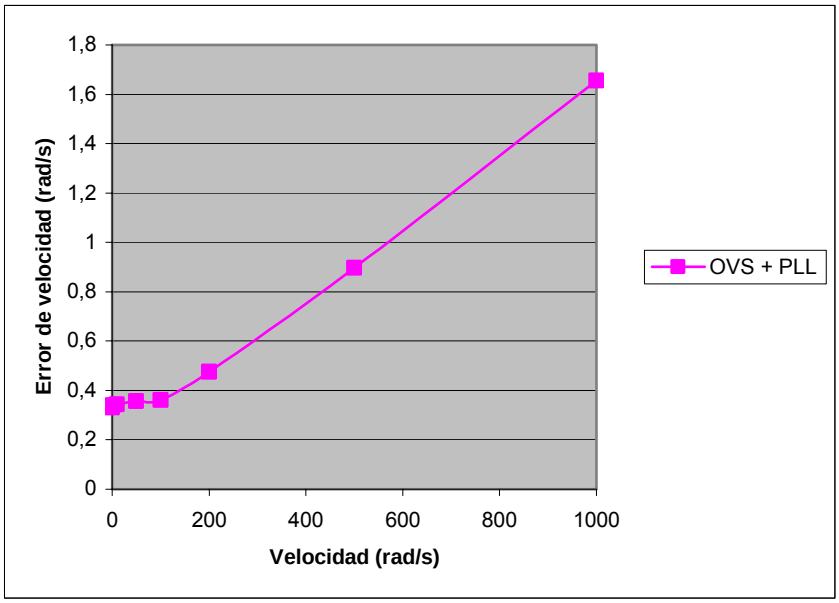

Ilustración 5.14: error de velocidad en función de la velocidad.

6 Algoritmos

La estructura del programa se basa en algoritmos que se ejecutan cuando se producen las interrupciones que están asignadas a ciertos eventos. Así las dos interrupciones principales en que se divide el programa son las siguientes:

- ADC_Int: esta interrupción se produce cada vez que el DSP ha acabado de muestrear los valores analógicos deseados: el seno y coseno provenientes del resolver, las intensidades de dos fases y la tensión de bus. Este muestreo es iniciado de manera automática por el temporizador T2 del DSP a 144kHz. En esta interrupción se calcula el seno y coseno de la posición con oversampling y a través de la PLL se calcula la posición y velocidad.

- SVPWM_Int: esta interrupción es disparada por el timer T1 del DSP a 9kHz. En esta interrupción se hace el control de velocidad, control de par (o corriente, que viene a ser equivalente) y se realiza el SVPWM.

Estas interrupciones se ejecutan con una frecuencia bastante alta. Es por esto que se han realizado enteramente en ensamblador, para ahorrar el máximo tiempo de cálculo posible. En los apartados siguientes se explicarán los algoritmos que se ejecutan en cada una de estas dos interrupciones y el ensamblaje en el programa principal.

6.1 Algoritmo de oversampling (ADC_Int)

6.1.1 Cálculos a realizar por el DSP

Teniendo en cuenta el esquema de la Ilustración 4.2 el DSP tiene que hacer los cálculos que se describen a continuación:

A $2k \cdot f_e = 2 \cdot 16 \cdot 4{,}5kHz = 144kHz$, es decir cada 1/(144kHz)=6,94μs se ha de hacer lo siguiente:

- Generar el valor de la señal de excitación del resolver (que tiene una frecuencia fundamental de fexc). Generar esta señal senoidal, consiste básicamente en acceder a un valor de una tabla según un índice.

- Por otro lado, pero a la misma frecuencia se ha de adquirir dos señales (una para el seno y otra para el coseno) y multiplicar ambas por la señal de excitación desfasada.

A f_e=4,5kHz, es decir cada 1/(4,5kHz)=222μs se ha de hacer lo siguiente:

- Los valores obtenidos en el paso anterior se tienen que filtrar con un filtro de 4k+1=4·16+1=65 coeficientes para obtener un valor filtrado (promediado). Los filtros FIR consisten básicamente en hacer sumas de productos: se suman los 65 productos de valores anteriores en el tiempo (incluido el valor actual) por sus respectivos coeficientes (Ec. 6.1). Esto requiere guardar los valores de los 64 valores anteriores y el actual. Al aplicar un diezmado solo se tendrá que calcular una suma de productos cada 32 valores que muestreemos. En conclusión: cada 32 valores se inicia una nueva suma de 65 productos anteriores. Por lo tanto se "solaparán" siempre de dos a tres de estas sumas de productos, es decir que cada valor muestreado participará en dos o tres de estas sumas de productos. Con esto se obtendrá como resultado el seno y coseno del ángulo girado del motor (pero con un desfase de 222µs).

$$y(n) = \sum_{i=(n-4\cdot k)}^{n} x(i)\cdot b_i \qquad \text{Ec. 6.1}$$

- Por otro lado se tendrá la PLL que con este seno y coseno calcula el ángulo y la velocidad del motor según se explica en el capítulo 4.3. Para el cálculo del seno y coseno se recurre a una tabla de 360 valores en la cual se interpola para conseguir más precisión. Para la integración se utiliza el método de los trapecios, igual que para la parte integral del PID.

6.1.2 Optimizaciones

Teniendo en cuenta las posibilidades del DSP se han introducido algunas mejoras para disminuir el tiempo de cálculo requerido por el oversampling, ya que es la parte más critica de todo el programa y debido al hecho de que se ejecuta a una frecuencia realmente alta (144kHz).

- Primeramente se utiliza la posibilidad que da el DSP, para leer dos tandas de todos los valores analógicos de manera totalmente automática. Así la interrupción solamente se ejecutará la mitad de veces, es decir a 72Khz (13.9µs de periodo). Esto es especialmente útil, ya que así se evitan la mitad de interrupciones, con lo que el tiempo de cálculo se optimiza considerablemente. Esto se debe a que cada vez que salta una interrupción hay que guardar todo el contexto del programa, y cuando termina hay que restaurarlo, lo cual ya supone un tiempo de cálculo importante.

- Como el valor de los retrasos se ha mantenido relativamente constante, se ha optado por introducir la multiplicación de las señales muestreadas y_{s1} e y_{c1} por la señal de excitación retrasada y_{e1} directamente en los coeficientes del filtro. Con esto se ahorra tener que hacer el retraso de la señal de excitación y las dos multiplicaciones. Al no requerir recalcular la señal de excitación retrasada para la multiplicación, se puede generar el seno de excitación del resolver con la mitad de valores, es decir uno en cada interrupción (a 72kHz).

- El hecho de que varias de las sumas de productos (necesarias los filtros FIR) se "solapen" introduce una relativa complicación que pude llevar a tener que copiar muchos datos de una posición de memoria a otra. Para evitar esto se ha ideado un algoritmo para filtrar estos datos de una manera relativamente simple aprovechando comandos de ensamblador optimizados del DSP. Para ello cada vez que se han adquirido 32 valores nuevos se filtra alternativamente con uno u otro método. No se entrará en más detalle de estos métodos.

Se puede ver un esquema básico de funcionamiento del algoritmo optimizado en la Ilustración 6.1.

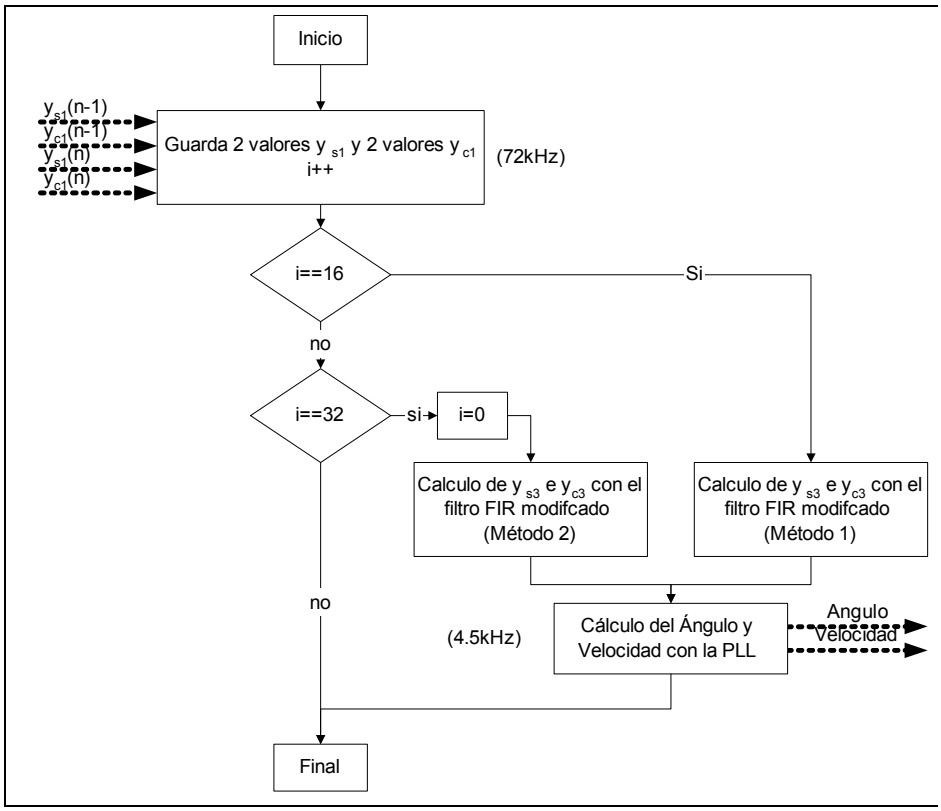

Ilustración 6.1: esquema básico de la interrupción ADC_Int que se realiza a 72kHz.

6.2 Algoritmo de control (SVPWM_Int)

La función básica del algoritmo de control integrado en esta interrupción, es dar la orden de apertura o cierre de cada uno de los 6 IGBTs a partir de un valor de consigna de velocidad o par (que equivale a dar consigna de corriente).

El esquema de control de velocidad completo se puede ver en la Ilustración 6.2.

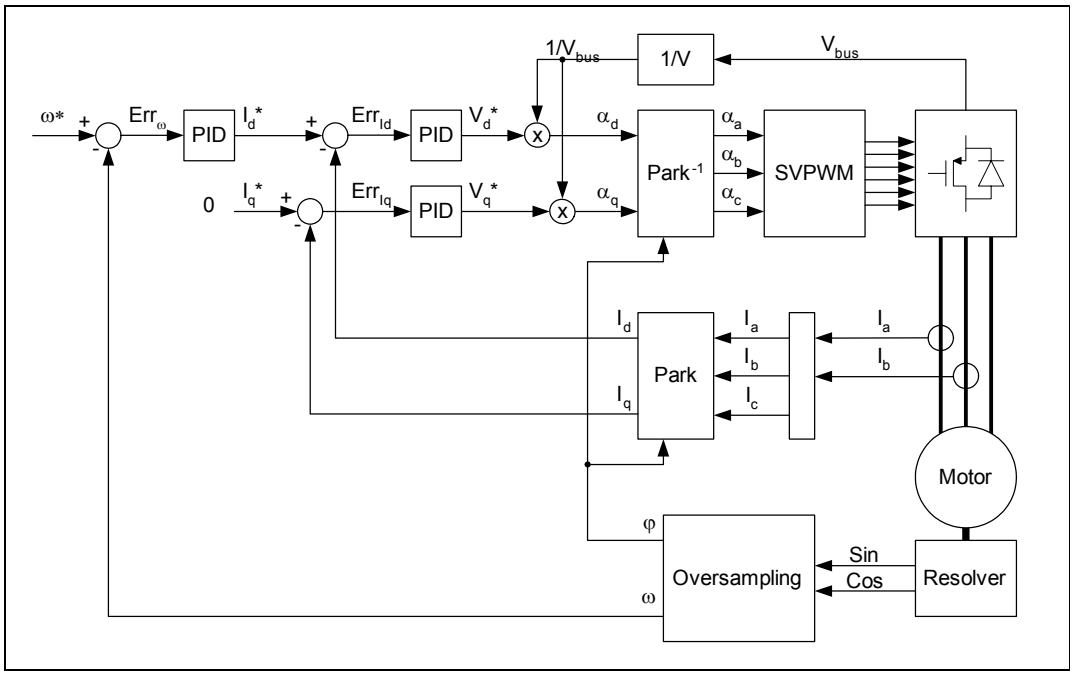

Ilustración 6.2: esquema completo del sistema de control.

Como se ve en la Ilustración 6.2, el sistema necesita la lectura de los siguientes parámetros:

- Las corrientes de las fases del motor. En realidad solo se miden dos de ellas y se calcula la tercera, ya que su suma tiene que dar cero. A estas corrientes se les aplica la trasformada de Park con lo que se obtienen la corriente directa y la corriente de cuadratura que nos sirven directamente para realimentar los dos bucles de corriente. Para una explicación más detallada de la transforma y antitransformada de Park consultar el apartado 6.2.1.

- La posición y velocidad del motor. Esto se consigue con las señales del resolver y a través del algoritmo del oversampling. En realidad se obtienen valores refrescados a una frecuencia de 4500Hz. Estos valores también sirven para realimentar el bucle de velocidad.

- Por otro lado se mide la tensión del bus, esto se hace para que los valores de las constantes de los PID de corriente sean independiente de la tensión de bus y para la seguridad del sistema frente a sobretensiones.

Existe la posibilidad de cambiar el tipo de control, así simplemente cambiando un parámetro del algoritmo se puede hacer los siguientes controles según la aplicación a que vaya destinada el motor:

- Control de velocidad, en este caso se dan consignas de velocidad (ω*) y se realizan todos los bucles.

- Control de corriente, en este caso se prescinde del bucle de velocidad y se dan consignas de corriente directa (I_d*) y corriente en cuadratura (I_q*).

- Control directo de tensión, en este caso se prescinden de todos los bucles y se dan como consignas la tensión directa (V_d*) y la tensión en cuadratura (V_q*). Cuando se hace este tipo de control no hay ningún tipo de realimentación para corregir errores. Esto se debe a que no hace falta, ya que la tensión se controla de forma directa variando el tiempo en que los interruptores de cada rama están en una posición o en otra. El control de estos tiempos se hace de forma muy precisa y sin prácticamente errores perceptibles por medio del DSP, sobretodo si se tiene en cuenta que el tiempo de conmutación es de 9000Hz.

Los bucles de corriente se ejecutan a 9000Hz. En el caso de que se haga control de velocidad, los bucles de corriente se realizan también a 9000Hz pero el bucle de velocidad se realiza a la mitad de frecuencia, a 4500Hz, para poder permitir adquirir las velocidades deseadas al motor ya que se tienen que tener en cuenta las constantes de tiempo eléctricas y mecánicas del motor.

Sea cual sea el bucle que se ejecute, al final se tiene un valor deseado de tensión directa (V_d*) y tensión en cuadratura (V_q*).

El algoritmo SVPWM (Space Vector Pulse Width Modulación) se encarga de dar los tiempos que tienen que estar abiertos y cerrados cada uno de los IGBT. De esta forma en cada una las tres fases de salida del convertidor se tiene unos valores de tensión relativos a la tensión de bus, que son función directa de los valores α_a, α_b y α_c que se de como consigna al algoritmo SVPWM. Por tanto estos valores son valores en tanto por uno sobre la tensión de bus. Así, si la tensión del bus fuese siempre continua, se podría convertir directamente los

valores V_d* y V_q* a sus componentes por fases a través de la antitransformada de Park. Estos valores (escalados convenientemente) son los valores que habría que dar al algoritmo SVPWM para que el convertidor diese como resultado las tensiones deseadas. Para una explicación más detallada del SVPWM consultar el apartado 6.2.2.

Pero la realidad es que la tensión de bus puede variar por diversos motivos:

- Fluctuaciones de tensión de la red, que repercuten de forma directa en el valor de tensión una vez rectificado.
- Consumos o aportaciones de corrientes grandes al bus de continua, de forma que los condensadores del bus no pueden mantener constantes la tensión.
- Utilización en ambientes y situaciones con valores de tensión de red diferentes, por lo que el valor rectificado varia consecuentemente.

Por todo esto es conveniente introducir el valor de tensión del bus de continua en el algoritmo, de manera que el valor correcto de las constantes de los controladores PID de las dos corrientes sean independientes de la tensión del bus. Así los valores deseados de tensión directa (V_d*) y tensión en cuadratura (V_q*), se multiplican por la inversa de la tensión de bus para obtener unos valores en tanto por uno respecto la tensión de bus (α_d y α_q respectivamente). Estos se pasan a valores se pasan a valores por fase (α_a, α_b y α_c) con la antitransformada de Park, y finalmente se introducen en el algoritmo SVPWM que da directamente las órdenes de apertura y cierre de los IGBT en los instantes adecuados.

6.2.1 Transformada de Park

La transformada de Park se basa en hacer un cambio de variables, que consiste en sustituir les variables (tensiones, intensidades y flujos concatenados) asociados a los devanados del estator de un motor síncrono (o como en nuestro caso, un motor brushless) por otras variables asociadas a unos devanados ficticios que van dando vueltas junto al rotor. Se puede considerar como el cambio de las variables a una referencia fija al rotor. La transformada de Park tiene la propiedad de eliminar las inductancias variables con la posición que aparecen en las ecuaciones de las tensiones de la máquina síncrona.

En realidad se trata de la encadenación de dos transformadas, la primera (transformación de concordia) consiste en reducir las tres variables, expresadas en los tres ejes situados sobre un plano y separados 120º entre ellos, a dos nuevas variables expresadas en dos ejes también en un plano y perpendiculares entre ellas. La segunda transformación consiste en

una rotación, que pasa de la referencia ortogonal en que estaban, a una nueva referencia igualmente ortogonal pero girada un cierto ángulo φ respecto a la anterior. Es ángulo puede ser cualquiera, pudiendo variar con el tiempo, con lo que se consigue una nueva referencia que gira a cierta velocidad respecto a la anterior.

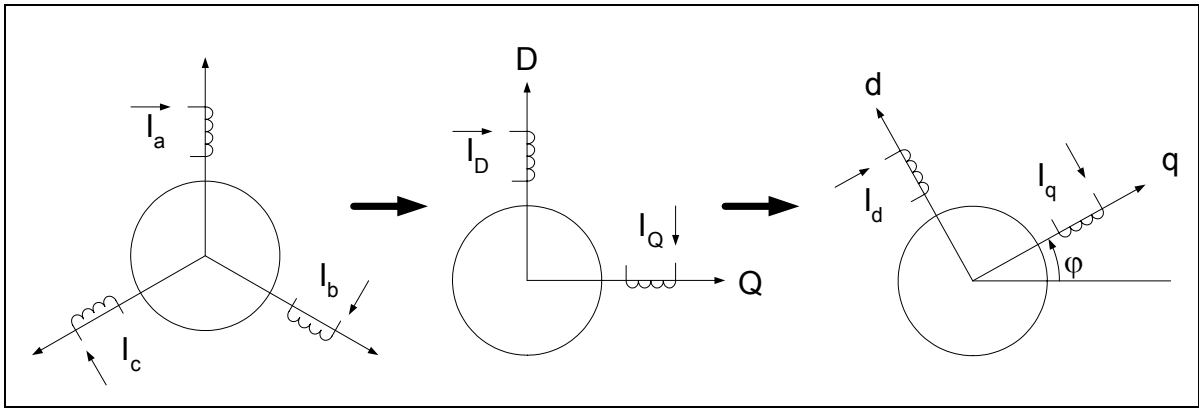

Ilustración 6.3: esquema de las dos transformaciones de las que se compone la transformada Park.

Matemáticamente la transformada se representa con las ecuaciones:

$$\begin{bmatrix} X_0 \\ X_D \\ X_Q \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \sqrt{\frac{3}{2}} & -\sqrt{\frac{3}{2}} \end{bmatrix} \cdot \begin{bmatrix} X_a \\ X_b \\ X_c \end{bmatrix}$$

$$\begin{bmatrix} X_0 \\ X_d \\ X_q \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_D \\ X_Q \end{bmatrix}$$

$$\begin{bmatrix} X_0 \\ X_d \\ X_q \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos\varphi & \cos\left(\varphi - \frac{2\pi}{3}\right) & \cos\left(\varphi + \frac{2\pi}{3}\right) \\ -\sin\varphi & -\sin\left(\varphi - \frac{2\pi}{3}\right) & -\sin\left(\varphi + \frac{2\pi}{3}\right) \end{bmatrix} \cdot \begin{bmatrix} X_a \\ X_b \\ X_c \end{bmatrix} \qquad \text{Ec. 6.2}$$

Donde X_a, X_b y X_c son tres variables cualquiera de las tres fases (tensión, corriente, flujo, etc.) y X_0, X_d y X_q son la componente homopolar, la componente directa y la componente en

cuadratura respectivamente, expresadas en la referencia de Park. Esta transformación es homopolar, por lo que la función inversa corresponde con la matriz transpuesta:

$$\begin{bmatrix} X_a \\ X_b \\ X_c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \cos\varphi & -\sin\varphi \\ \frac{1}{\sqrt{2}} & \cos\left(\varphi - \frac{2\pi}{3}\right) & -\sin\left(\varphi - \frac{2\pi}{3}\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\varphi + \frac{2\pi}{3}\right) & -\sin\left(\varphi + \frac{2\pi}{3}\right) \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_d \\ X_q \end{bmatrix} \qquad \text{Ec. 6.3}$$

Se demuestra que si $X_a+X_b+X_c=0$, la componente homopolar X_0 es en todo momento nula. Este es el caso que se tendría cuando las variables corresponden con las corrientes del estator del motor cuando no se tiene el neutro conectado.

6.2.2 SVPWM

Este método de control del ondulador trifásico se basa en que los ocho estados que puede adoptar según estén abiertos o cerrados cada uno de los interruptores que contiene.

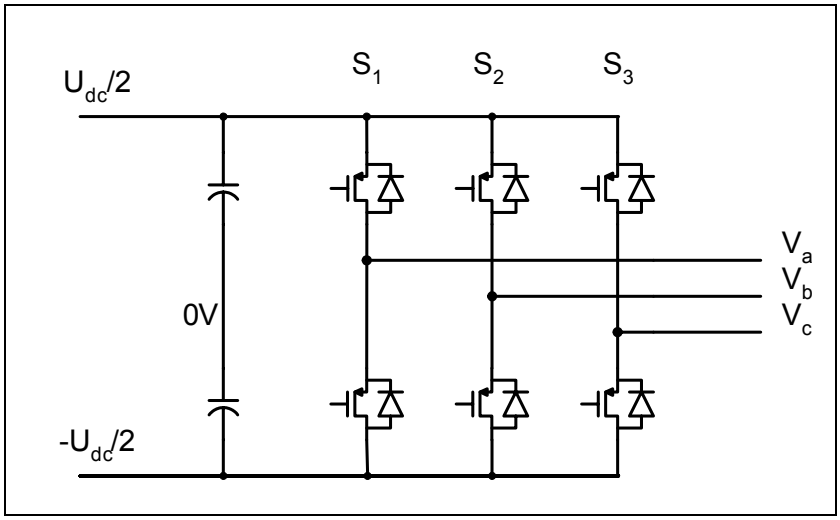

Ilustración 6.4: ondulador trifásico.

De los dos interruptores de cada una de las tres ramas (S_1, S_2, S_3) siempre habrá uno cerrado y uno abierto, correspondiendo el 1 cuando el de arriba está cerrado y el 0 cuando es al revés. Con esto, los 8 posibles estados que se pueden conseguir son los que muestran en la Tabla 6.1. En ella también se muestran las tensiones de cada una de las fases y las respectivas tensiones después de aplicarles la transformada de Park.

	S1	S2	S3	$V=\begin{pmatrix}V_a & V_b & V_c\end{pmatrix}$	$V_{Park}=\begin{pmatrix}V_d & V_q & V_0\end{pmatrix}$
$\underline{V}_0$	0	0	0	$\begin{pmatrix}\frac{-1}{2} & \frac{-1}{2} & \frac{-1}{2}\end{pmatrix}U_{dc}$	$\begin{pmatrix}0 & 0 & \frac{-\sqrt{3}}{2}\end{pmatrix}U_{dc}$
$\underline{V}_1$	1	0	0	$\begin{pmatrix}\frac{1}{2} & \frac{-1}{2} & \frac{-1}{2}\end{pmatrix}U_{dc}$	$\begin{pmatrix}\sqrt{\frac{2}{3}} & 0 & \frac{-\sqrt{3}}{6}\end{pmatrix}U_{dc}$
$\underline{V}_2$	1	1	0	$\begin{pmatrix}\frac{1}{2} & \frac{1}{2} & \frac{-1}{2}\end{pmatrix}U_{dc}$	$\begin{pmatrix}\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{6}\end{pmatrix}U_{dc}$
$\underline{V}_3$	0	1	0	$\begin{pmatrix}\frac{-1}{2} & \frac{1}{2} & \frac{-1}{2}\end{pmatrix}U_{dc}$	$\begin{pmatrix}\frac{-1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & \frac{-\sqrt{3}}{6}\end{pmatrix}U_{dc}$
$\underline{V}_4$	0	1	1	$\begin{pmatrix}\frac{-1}{2} & \frac{1}{2} & \frac{1}{2}\end{pmatrix}U_{dc}$	$\begin{pmatrix}-\sqrt{\frac{2}{3}} & 0 & \frac{\sqrt{3}}{6}\end{pmatrix}U_{dc}$
$\underline{V}_5$	0	0	1	$\begin{pmatrix}\frac{-1}{2} & \frac{-1}{2} & \frac{1}{2}\end{pmatrix}U_{dc}$	$\begin{pmatrix}\frac{-1}{\sqrt{6}} & \frac{-1}{\sqrt{2}} & \frac{-\sqrt{3}}{6}\end{pmatrix}U_{dc}$
$\underline{V}_6$	1	0	1	$\begin{pmatrix}\frac{1}{2} & \frac{-1}{2} & \frac{1}{2}\end{pmatrix}U_{dc}$	$\begin{pmatrix}\frac{1}{\sqrt{6}} & \frac{-1}{\sqrt{2}} & \frac{\sqrt{3}}{6}\end{pmatrix}U_{dc}$
$\underline{V}_7$	1	1	1	$\begin{pmatrix}\frac{1}{2} & \frac{1}{2} & \frac{1}{2}\end{pmatrix}U_{dc}$	$\begin{pmatrix}0 & 0 & \frac{\sqrt{3}}{2}\end{pmatrix}U_{dc}$

Tabla 6.1: 8 posibles estados del ondulador trifásico

Si se representan las tensiones en los ejes d-q se obtienen 6 vectores desfasados 60º y dos vectores de longitud cero ($\underline{V}_1$ y $\underline{V}_7$), tal como se puede ver en la Ilustración 6.5.

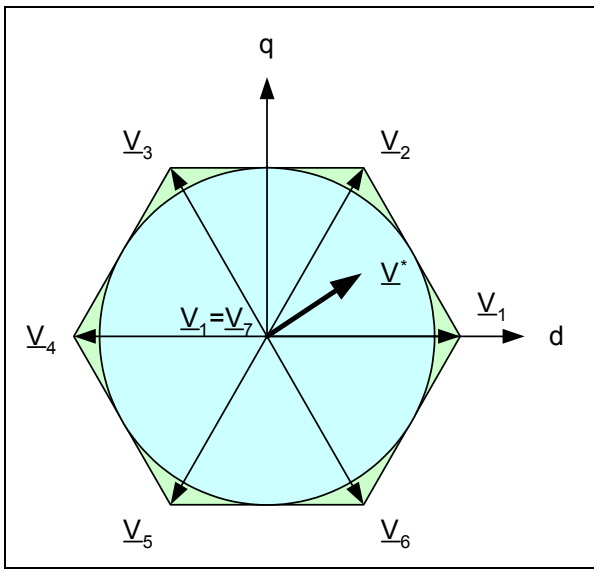

Ilustración 6.5: diagrama de modulación vectorial SVPWM y el vector consigna $\underline{V}$*.

La modulación SVPWM consiste en sintetizar el vector tensión ($\underline{V}^*$) mediante los 8 únicos vectores de que dispone el ondulador. Para conseguir esto se impone:

$$\underline{V}^* \frac{T}{2} = \sum_{i=0,\ldots,7} \underline{V}_i \cdot t_i \qquad \text{Ec. 6.4}$$

Donde $\underline{V}^*$ es el vector consigna en coordenadas d-q que se quiere aplicar, T es la inversa de la frecuencia de conmutación de los interruptores, $\underline{V}_i$ es cada uno de los 8 vectores del ondulador que se aplican durante un tiempo t_i, con la siguiente restricción:

$$\sum_{i=0,\ldots,7} t_i = \frac{T}{2} \qquad \text{Ec. 6.5}$$

El vector que se desea se puede sintetizar de múltiples maneras, en nuestro caso se utilizará el método propuesto por H. W. Van Der Broek [5], en el que únicamente se utilizan los dos vectores adyacentes al vector a sintetizar y los dos vectores nulos, por lo tanto 4 en total.

Por lo tanto, lo que hay que hacer para sintetizar el valor consigna, es imponer que el valor medio durante un periodo de los cuatro vectores sea el valor consigna, esto se consigue variando el tiempo que se activa cada uno de los cuatro vectores. Para el caso particular en que el vector consigna esté en la primera región se tiene:

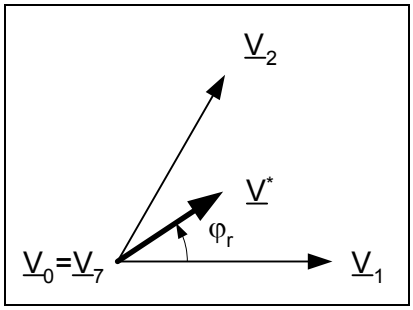

Ilustración 6.6: síntesis del vector consigna cuando está en la primera región.

Si se coge:

$$V = \sqrt{V_d^{\,2} + V_q^{\,2}}$$
$$\varphi_{dq} = \arctan\left(\frac{V_d}{V_q}\right) \qquad \text{Ec. 6.6}$$

Entonces el ángulo φ_{dq} determina una región de las seis posibles. Si se define φ_r como:

$$\varphi_r = \varphi_{dq} - n\pi/3 \qquad n = 1\ldots 6 \qquad \text{Ec. 6.7}$$

Este ángulo representa el ángulo dentro de una región concreta. Por lo tanto se cumple:

$$\underline{V}^* = \frac{T_\alpha}{T/2}\underline{V}_\alpha + \frac{T_\beta}{T/2}\underline{V}_\beta \qquad \text{Ec. 6.8}$$

Donde $\underline{V}_\alpha$ y $\underline{V}_\beta$ son los vectores adyacentes directamente sintetizables por el ondulador, y T_α y T_β son los tiempos de aplicación de estos vectores y T el periodo de conmutación. Aislando se tiene:

$$\begin{aligned} T_\alpha &= \frac{\sqrt{3}\cdot V}{V_{dc}}\frac{T}{2}\sin\left(\frac{\pi}{3} - \varphi_r\right) \\ T_\beta &= \frac{\sqrt{3}\cdot V}{V_{dc}}\frac{T}{2}\sin\left(\varphi_r\right) \\ T_0 &= \frac{T}{2} - T_\alpha - T_\beta \end{aligned} \qquad \text{Ec. 6.9}$$

Donde V_{dc} es la tensión de bus y T_0 es el tiempo total que se aplican los vectores nulos ($\underline{V}_1$ y $\underline{V}_7$). Habitualmente T_0 se reparte a partes iguales entre los dos vectores nulos. Con esto se consigue el SVPWM simétrico. Para conseguir el menor número de conmutaciones se sigue la estrategia de ir cambiando el orden de aplicación de los cuatro vectores. En el caso de estar en la primera región la secuencia de aplicación es V_0, V_1, V_2, V_7, V_2, V_1, V_0:

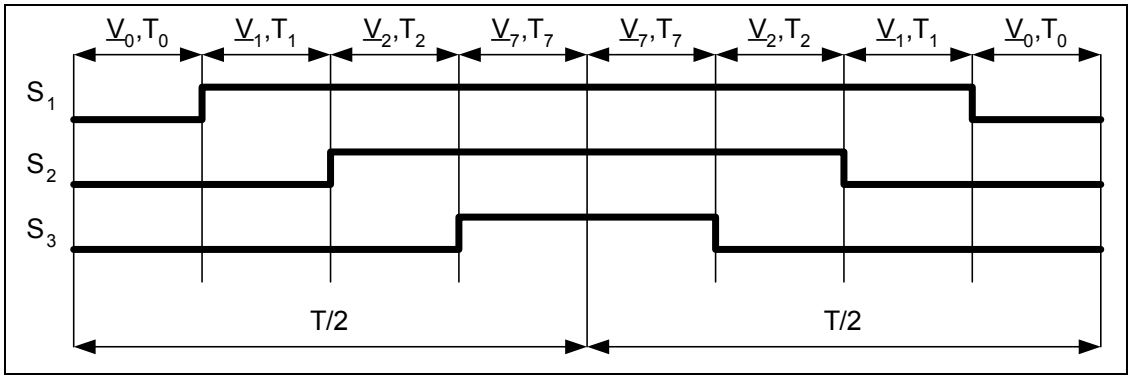

Ilustración 6.7: aplicación de los vectores adyacentes y vectores nulos con el SVPWM simetrico.

Este método presenta una serie de problemas a la hora de implementarlo en controles digitales tales como el DSP, ya que requieren un elevado tiempo de cálculo:

- Para saber en que región está el vector consigna hace falta hacer una división y calcular el arcotangente de esta división.
- Para calcular los tiempos de aplicación de de cada uno de los vectores hace falta calcular un seno.

Todas estas funciones no están implementadas de manera directa en el DSP y requieren un elevado tiempo de cálculo y espacio en memoria, si se aplican mediante la consulta en tablas de valores. Para corregir estos problemas se utilizarán soluciones alternativas que se presentan a continuación.

6.2.3 Cálculo de los tiempos de aplicación de cada vector

Para calcular los tiempos de aplicación de cada uno de los cuatro vectores se una solución alternativa propuesta por Joan Bergas [2] (Pág. 72-74). Se parte de la modulación del vector consigna que se hace en el SVPWM:

$$\begin{aligned} \underline{V}^* \frac{T}{2} &= \underline{V}_1 \cdot t_1 + \underline{V}_2 \cdot t_2 + \underline{V}_0 \cdot t_0 + \underline{V}_7 \cdot t_7 \\ \frac{T}{2} &= t_1 + t_2 + t_0 + t_7 \end{aligned} \qquad \text{Ec. 6.10}$$

La primera ecuación es una ecuación compleja que se puede dividir en dos:

$$\begin{aligned} V_d^* \frac{T}{2} &= V_{1d} \cdot t_1 + V_{2d} \cdot t_2 + V_{0d} \cdot t_0 + V_{7d} \cdot t_7 \\ V_q^* \frac{T}{2} &= V_{1q} \cdot t_1 + V_{2q} \cdot t_2 + V_{0q} \cdot t_0 + V_{7q} \cdot t_7 \\ \frac{T}{2} &= t_1 + t_2 + t_0 + t_7 \end{aligned} \qquad \text{Ec. 6.11}$$

En estas ecuaciones las únicas incógnitas son los tiempos t_1, t_2, t_0 y t_7. Además se tiene que las componentes directa y en cuadratura de los vectores nulos valen 0 (V_{0d}= V_{7d}= V_{0q}= V_{7q}=0). De manera que despejando queda:

$$t_1 = \frac{V_{2q}V_d - V_{2d}V_q}{V_{1d}V_{2q} - V_{2d}V_{1q}} \cdot \frac{T}{2}$$
$$t_2 = \frac{V_{1d}V_q - V_{1q}V_d}{V_{1d}V_{2q} - V_{2d}V_{1q}} \cdot \frac{T}{2} \qquad \text{Ec. 6.12}$$
$$t_0 = t_7 = \frac{T}{2} - T_1 - T_2$$

Para simplificar el cálculo con el DSP al máximo se puede juntar todos los términos constantes:

$$t_1 = \frac{V_{2q}}{V_{1d}V_{2q} - V_{2d}V_{1q}} \cdot \frac{T}{2} V_d - \frac{V_{2d}}{V_{1d}V_{2q} - V_{2d}V_{1q}} \cdot \frac{T}{2} V_q = K_{11}(V_{dc})\frac{T}{2}V_d - K_{12}(V_{dc})\frac{T}{2}V_q$$
$$t_2 = \frac{V_{1d}}{V_{1d}V_{2q} - V_{2d}V_{1q}} \cdot \frac{T}{2} V_q - \frac{V_{1q}}{V_{1d}V_{2q} - V_{2d}V_{1q}} \cdot \frac{T}{2} V_d = K_{21}(V_{dc})\frac{T}{2}V_q - K_{22}(V_{dc})\frac{T}{2}V_d \qquad \text{Ec. 6.13}$$

Así solamente se tendrá que hacer cuatro multiplicaciones y dos restas, que con el DSP se puede hacer de manera muy fácil y rápida. Se ha de tener en cuenta que los valores de estas constantes dependen de la tensión del bus V_{dc} y por tanto esto solo son válidos para una tensión de bus fijada. Como se ha comentado antes la tensión de bus es variable, por lo tanto lo que se hará es calcular estos valores para un caso en concreto (en que la tensión de bus sea máxima) y luego dar consignas en tanto por uno de respecto a este valor.

Región	K_{11}	K_{12}	K_{21}	K_{22}
1	$\sqrt{\frac{3}{2}}\frac{1}{V_{dc}}$	$\frac{1}{\sqrt{2}}\frac{1}{V_{dc}}$	$\sqrt{2}\frac{1}{V_{dc}}$	0
2	$\sqrt{\frac{3}{2}}\frac{1}{V_{dc}}$	$\frac{-1}{\sqrt{2}}\frac{1}{V_{dc}}$	$\frac{1}{\sqrt{2}}\frac{1}{V_{dc}}$	$\sqrt{\frac{3}{2}}\frac{1}{V_{dc}}$
3	0	$-\sqrt{2}\frac{1}{V_{dc}}$	$\frac{-1}{\sqrt{2}}\frac{1}{V_{dc}}$	$\sqrt{\frac{3}{2}}\frac{1}{V_{dc}}$
4	$-\sqrt{\frac{3}{2}}\frac{1}{V_{dc}}$	$\frac{-1}{\sqrt{2}}\frac{1}{V_{dc}}$	$-\sqrt{2}\frac{1}{V_{dc}}$	0
5	$-\sqrt{\frac{3}{2}}\frac{1}{V_{dc}}$	$\frac{1}{\sqrt{2}}\frac{1}{V_{dc}}$	$\frac{-1}{\sqrt{2}}\frac{1}{V_{dc}}$	$-\sqrt{\frac{3}{2}}\frac{1}{V_{dc}}$
6	0	$\sqrt{2}\frac{1}{V_{dc}}$	$\frac{1}{\sqrt{2}}\frac{1}{V_{dc}}$	$-\sqrt{\frac{3}{2}}\frac{1}{V_{dc}}$

Ilustración 6.8: valores de las constantes para el cálculo de los tiempos en cada región.

6.2.4 Cálculo de la región del vector tensión

Para solucionar el problema de determinar la región del vector tensión consigna que se tiene, se aplicará una solución propuesta también por Joan Bergas [2] (Pág. 98-100). Se recuerda que el problema consistía en hacer el cálculo del ángulo φ_{dq} para saber en cual de las 6 regiones está el vector consigna.

$$\varphi_{dq} = \arctan\left(\frac{V_d}{V_q}\right) \qquad \text{Ec. 6.14}$$

Para saber en que región estamos en realidad no hace falta saber el ángulo. Para empezar se pueden distinguir cuatro regiones mirando simplemente los signos de las componentes del vector consigna. Por lo tanto, hace falta una tercera relación que nos permita discernir entre las seis regiones. Para esto se aprovecha el hecho de que todas las regiones están divididas por vectores que forman 60º respecto la horizontal, y que por tanto la ecuación matemática que describe esta recta es (en el primer cuadrante):

$$\begin{aligned} V_q &= \tan(60)V_d = \sqrt{3}V_d \\ \sqrt{3}V_d - V_q &= 0 \end{aligned} \qquad \text{Ec. 6.15}$$

Por lo tanto un vector tensión que se encuentre en el primer cuadrante tendrá el signo de la expresión anterior positivo, mientras que si se encuentra en el segundo cuadrante será negativo.

Por lo tanto, aprovechando el signo de las dos componentes del vector tensión, y el signo de la expresión anterior se puede crear la siguiente tabla que nos permitirá calcular la región del vector tensión utilizando únicamente multiplicaciones, sumas y comparaciones.

Región	1	2	3	4	5	6
Signo Vd	+		-	-		+
Signo Vq	+	+	+	-	-	-
$\sqrt{3}\lVert V_d \rVert - \lVert V_q \rVert$	+	-	+	+	-	+

6.3 Programa completo

Como se ha comentado anteriormente el programa se basa en una serie algoritmos que se ejecutan cuando se producen las interrupciones correspondientes. Pero mientras estas interrupciones no se producen hay un programa principal que va haciendo otras tareas no críticas en un bucle infinito. Entre estas tareas pueden estar las siguientes:

- Control global de las consignas. Aquí se incluye la programación del algoritmo de aquello que realmente se quiere hacer con el motor, esto dependerá de la aplicación a que esté destinado o lo que se quiera hacer. Se puede controlar según los valores digitales o analógicos que pueda haber en las entradas, según tiempos preestablecidos, o incluso según consignas recibidas a través de algún puerto de comunicaciones o bus de comunicaciones.

- Control de parámetros de seguridad como la tensión de bus, corrientes, velocidad, etc. En caso de que estas magnitudes lleguen a valores críticos se puede detectar y hacer el tratamiento correspondiente a cada uno de los casos

- Comunicación a través del bus industrial CAN, es decir recepción y tratamientos de mensajes entrantes y también el envío de mensajes salientes. El DSP y la tarjeta están totalmente preparados para la comunicación por CAN con otros aparatos, como un autómata, un PC, etc. Así la tarjeta, y por tanto el motor puede ser controlado a distancia, se puede enviar por ejemplo el modo de trabajo (control de velocidad, control de corriente o control de tensión) y las magnitudes de las variables asociadas. En realidad se puede programar el DSP para ser controlado de formas muy diversas. Por otro lado el DSP puede enviar de forma periódica mensajes dando información sobre el estado del convertidor, del motor o del mismo DSP. Así por ejemplo se puede enviar información referente la tensión del bus, corrientes del DSP, temperatura del motor (si se conecta la sonda de temperatura del motor a una de las entradas analógicas sobrantes), errores producidos, tiempo de cálculo sobrante, etc.

- Extracción de variables por el puerto SPI hacia el DAC externo, Esto sirve sobretodo para depurar el programa, pudiendo ver la evolución de cualquier variable en tiempo real en la salida del DAC, con la ayuda de un osciloscopio, por ejemplo. Esto es de vital importancia cuando el programa, y sobretodo las interrupciones, se ejecutan a frecuencias tan elevadas, ya que se hace imposible seguirlas a través del programa de monitoreo que incluye la suite de programación del DSP.

En realidad, como el prototipo no se ha diseñado para ninguna aplicación concreta, solo se han programado ciertas funcionalidades básicas:

- Control de activación del control del motor. El control no se activa hasta que se activa el control con una variable booleana y hasta que la tensión del bus no supera un umbral mínimo de funcionamiento (unos 100V), por debajo del cual el control de

corriente se hace más impreciso y por tanto introducen inestabilidades en el convertidor y vibraciones en el motor. Para la desactivación del control la tensión tiene que caer por debajo de 60V. Con esta histéresis se quiere evitar que el control se comporte de forma incontrolada cuando la tensión del bus esté cerca del límite. Esto se produce ya que tanto la tensión del bus como la lectura de esta pueden variar entorno de este límite haciendo que el control se active y desactive a frecuencias altas, produciendo importantes ruidos y sobretodo situaciones peligrosas.

- Envío de señales a monitorear al DAC
- Envío y recepción de mensajes del CAN, aunque, no se hace ningún tratamiento de las señales entrantes ni se envía ninguna.

Por otro lado se ha de comentar que no todas las interrupciones tienen la misma prioridad. Así se ha asignado la interrupción de máxima prioridad al oversampling, ya que es la que se ejecuta con más frecuencia y por tanto es crítico que se que realice a tiempo. Cuando está activa esta interrupción no se permite la interrupción ninguna otra. La segunda interrupción mas importante es la del control, mientras se ejecuta esta solo se permite la interrupción del oversampling.

Se ha de comentar que en caso de que se produzca un fallo en el SKiiP este envía una señal de error que activa la misma interrupción del ADC y produce que la las salidas que controlan los IGBT del SKiiP se pongan en alta impedancia. Como estás salidas tienen unas resistencias de “pull-down” se abrirán de forma inmediata todos los IGBT para evitar daños materiales y humanos. Esta señal de error además hace saltar la misma interrupción que usa el oversampling (la de máxima prioridad), por lo tanto para detectar este caso se vigila la causa de la interrupción a la entrada de la misma.

7 El prototipo

Aparte de la programación del DSP, el montaje de la placa de control y otros elementos, uno de los objetivos fundamentales del proyecto era construir un prototipo que realmente pudiera hacer el control de los motores. El prototipo está formado básicamente por el rectificador con el bus de continua, el puente ondulador, el control y la realimentación del sistema.

Este prototipo aparte de servir para el control de motores puede servir para otros múltiples propósitos, como por ejemplo: filtros activos, onduladores,… Para ello simplemente se tendrá que quitar ciertos módulos y añadir otros, pero dejando los elementos fundamentales intactos. Estos elementos fundamentales son el bus de continua, el puente ondulador y el control. Después, y según el propósito se pueden aprovechar alguno o todos los módulos del prototipo para otros usos. Por ello se ha intentado hacer el diseño del prototipo lo más modular posible para permitir implementar otras funcionalidades en el mismo prototipo.

7.1 El rectificador y el bus de continua

Para dar energía a todo el sistema se utiliza la red eléctrica. Así el sistema recibe toda la potencia de una red trifásica de 230V a través de un autotransformador. El control en cambio se alimenta directamente de la red de 230V.

Para obtener la tensión continua que se necesita en el bus, hace falta rectificar la tensión alterna trifásica de la red. Para esto se ha utilizado un puente rectificador trifásico no controlado, es decir que solo integra diodos. El modelo usado es el modelo DF40AA160 de SANREX ya que se disponía de uno en el laboratorio. Este modelo es capaz de bloquear tensiones de 1600V y soportar corrientes de hasta 40A, lo cual es más que suficiente para el control de los motores brushless de los que se dispone en el laboratorio. Para poder disipar todo el calor que genera este dispositivo se ha montado en un disipador propio.

La entrada a este elemento está protegido por un interruptor magnetotérmico trifásico de 20A y la salida por otro bifásico de 30A. Esta protección es doble para conseguir la modularidad deseada: así, por ejemplo, se puede sustituir el rectificador por otro prototipo con un SKiiP que alimente el bus de forma controlada.

Para que la tensión a la salida del rectificador sea continua se tiene que poner un elemento que la regule, este elemento es el bus de tensión continua (o bus de continua). Este bus se encarga de eliminar el rizado de tensión y suministrar los picos de corriente que demande el puente ondulador. El bus de continua está formado básicamente por condensadores. En el

laboratorio se disponen de condensadores electrolíticos de 470μF y 450V, esta tensión no tiene un margen de seguridad suficiente para las altas tensiones que puedan surgir por ejemplo en el caso de que el motor frene y devuelva la energía al bus. Para aumentar la tensión soportable por el bus hasta 900V como máximo, se han puesto los condensadores en serie, pero con esto se reduce la capacidad a la mitad. Por ello se han puesto más grupos de dos condensadores en paralelo. Así se ha puesto un total de 12 condensadores de 470μF, que proporcionan una capacidad total de 1410μF y soportan hasta 900V.

Estos condensadores soportan tensiones altas y tienen una gran capacidad, pero tienen el inconveniente de tener una inductancia bastante elevada que puede perjudicar las conmutaciones de los IGBTs. Para solventar esto se ha puesto lo mas cerca posible de los IGBTs unos condensadores rápidos de 1.5μF, estos será los encargados de proporcionar las puntas de corriente que demanden los IGBTs.

En el bus de continua se han puesto unas resistencias de descarga. La misión principal de estas resistencias es descargar el bus de tensión una vez que se deje de alimentar con el rectificador. Estas resistencias son de 75kΩ y 0.25W, con lo que partiendo de un bus de 900V se llegará a niveles seguros (por debajo de 50V) en unos 3 minutos y medio aproximadamente. Por otra parte estas resistencias permiten equilibrar las tensiones entre los condensadores.

Si se quisiera conectar el prototipo directamente a la red se tendría que hacer una precarga de los condensadores, ya que sino se podrían destruir fácilmente cuando estuviesen descargados. En nuestro caso no es necesaria esta precarga, ya que se dispone del autotransformador que permite subir la tensión de forma gradual permitiendo una carga lenta de los condensadores y evitando así picos de corriente.

7.2 El puente ondulador

Hoy en día existe un gran número de interruptores de potencia: BJT, MOSFET, GTO e IGBT, en todos ellos se han hecho grandes avances en los últimos años. Las propiedades que se buscan en estos interruptores son:

- Capacidad de bloquear altas tensiones
- Capacidad de conducir elevadas corrientes cuando están en conducción y con una pequeña caída de tensión y resistencia para minimizar pérdidas
- Tiempos de apertura y cierre bajos para poder trabajar a altas frecuencias

- Buena disipación de la energía

Ninguno de los tipos de interruptor se desmarca de los demás en todos los campos, cada uno tiene su campo de aplicación específico. Para la utilización que se le va a dar en este proyecto, es decir el SVPWM (Space Vector PWM), la capacidad de trabajar a altas frecuencias es el requisito más importante. Así nos quedan dos posibles candidatos: el MOSFET y el IGBT.

Como los MOSFET tienen unas grandes pérdidas en la resistencia interna cuando están en conducción, se ha optado por los IGBT. Sin embargo se ha de decir que estos últimos han de trabajar frecuencias menores y la alimentación de la puerta necesita de tensión positiva y otra negativa para controlar la apertura.

En el laboratorio se disponía de equipos SKiiP 132 GLD 120-412CTVU(E), estos son puentes onduladores de tres ramas de dos IGBT y una tercera rama con un solo IGBT para el frenado. En esta se puede conectar una resistencia que puede disipar la energía del bus cuando la tensión de este sobrepase cierto nivel, de manera que no se destruya el bus ni se produzcan daños en el equipo.

Este equipo resulta idóneo para el prototipo, con la ventaja de que incorpora los drivers de los IGBT integrados. Este equipo incorpora protecciones contra sobretensiones, sobrecorrientes, cortocircuitos y sobretemperatura. Se ha de mencionar que los drivers ya realizan los tiempos muertos entre conmutaciones, de manera que no se han de realizar con el DSP, aunque esto no suponen ningún problema para el DSP, ya que lo puede realizar de forma totalmente automática. Estos equipos ya vienen con su propio disipador de calor incorporado tal y como se puede ver en la Ilustración 7.1.

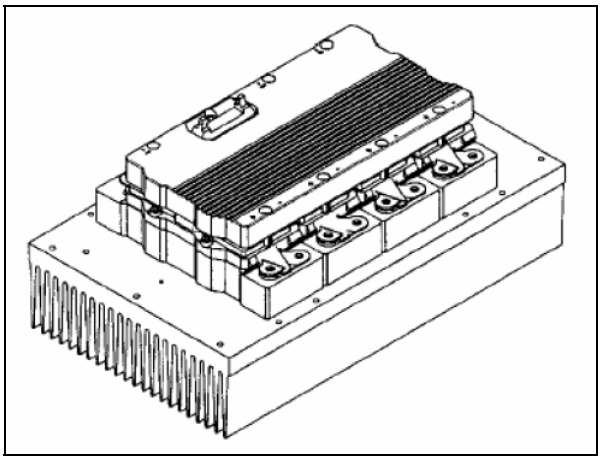

Ilustración 7.1: aspecto del ondulador SKiiP.

7.3 El control

El control es básicamente la placa de control con el DSP, esta se comenta ampliamente en el anejo A.1. Esta placa aloja el DSP y la electrónica necesaria para poder comunicar adecuadamente el microprocesador y el ondulador, adecuar las señales de las realimentaciones de corriente y del resolver, además de permitir la comunicación con elementos exteriores a través de RS232, SPI, CAN, JTAG,...

En este punto se podría contemplar también la fuente de alimentación, por que su función principal es alimentar esta placa, aunque también alimenta elementos de la realimentación del sistema.

7.4 La realimentación de sistema

Como realimentación del sistema básicamente se contemplan las corrientes de salida del SKiiP, la tensión del bus y las señales del resolver (una de excitación y las del seno y coseno que da como salida).

Para la lectura del bus de continua se utiliza la lectura que nos da el propio SKiiP, esta es lectura da de 0 a 10V cuando la tensión del bus varia entre 0 y 1000V.

7.4.1 Sondas de corriente

Para la lectura de corriente se tiene que tener en cuenta que se requiere un aislamiento galvánico entre la parte de potencia y la parte de control. Por otro lado se tendrá que adecuar la señal a niveles admisibles por el conversor analógico a digital (DAC) del DSP.

Para poder tener el asilamiento galvánico se podría pensar un transformador de corriente, pero dado que las corrientes que se quieren medir no son senoidales, no queda más remedio que usar sondas de efecto Hall. Estas proporcionan a su salida, cuando están convenientemente alimentadas, una corriente proporcional a la corriente que pasa a través del núcleo. Así al ser medida en corriente, y no en tensión, es menos sensible a ruidos. Para poderla medir finalmente con el DSP se tiene que convertir a señal de tensión mediante una resistencia de alta precisión que proporciona una caída de tensión proporcional a la corriente.

Las sondas utilizadas son el modelo LA55-P/SP1 de la marca LEM. Estas tienen un rango de hasta 50A que corresponden a 25mA a la salida. Estas sondas mantienen la forma de la onda de manera bastante perfecta. El error típico de estas sondas es del 0.5% de la intensidad nominal a 25ºC, esto son unos ±0.25A en el primario, lo cual es mínimo. En el

caso de que se quiera medir corrientes de menos intensidad con más precisión se pueden dar las vueltas necesarias en el primario. De manera que, para n vueltas, se tendrá que 50A/n en el primario, corresponden a 25mA en el secundario. Por otro lado estas sondas tienen un aislamiento galvánico que supera la prueba de los 2kVrms/50Hz/1min.

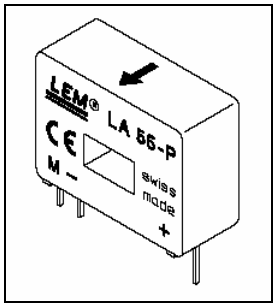

Ilustración 7.2: sondas de corriente.

Cada una de estas sondas van montadas en una placa separada que se tiene que alimentar con tensiones de -15V y +15V. Estas se han diseñado de manera que permitan un montaje modular, una al lado de otra. Así solo hace falta conectar cada sonda a una entrada analógica de la placa del DSP. Este montaje separado de la placa de control se ha hecho para evitar las perturbaciones electromagnéticas que pueden producir las altas corrientes a altas frecuencias que pueden pasar por los cables que atraviesan las sondas, y por que ocuparían demasiado sitio en la placa de control, cosa que haría encarecer mucho la fabricación de esta.

7.4.2 Filtro-amplificador del resolver

Para que el resolver pueda servir para leer el ángulo girado por el motor hace falta que este sea excitado en su primario por una señal senoidal de cierta amplitud. Esta señal se produce en el DSP mediante una salida PWM, por eso tiene que ser tratada antes de poder usarse como excitación del resolver. La salida PWM que genera el DSP está en el rango de 0 a 3.3V, y el resolver necesita una señal senoidal perfecta centrada en 0V y que tenga un valor efectivo del orden de los 10V.

Por eso se ha diseñado una placa aparte que filtra y amplifica la señal de manera que sirva para excitar el resolver. Este elemento no se ha montado en la placa de control por diversas razones; la primera es que no se han encontrado operacionales de gran corriente, como el que se necesita, para montaje superficial. Por otro lado, de esta manera se puede conservar una modularidad de todo el prototipo, ya que para otros montajes puede no hacer falta el filtro-amplificador del resolver, por lo que no interesa implementarlo.

7.5 Comunicación con el PC

El elemento encargado de intercomunicar el DSP y el PC es un emulador JTAG, este se conecta al puerto paralelo del PC por un lado y a la placa del DSP por otro. La interfaz que se utiliza desde PC para realizar todas las acciones necesarias sobre el DSP es el programa Code Composer Studio versión 2.20 de Texas Instruments. Entre otras las acciones que se realizan con este programa es realizar el programa para el DSP, depurarlo y grabarlo en la memoria flash del DSP.

Este emulador JTAG permite visualizar y realizar cambios sobre cualquier variable dentro del DSP, todo esto en tiempo real. Por otro lado permite programar la memoria flash del DSP, cosa que hace mucho más rápido que si se hace por el puerto serie. Todas estas cualidades son de una gran ayuda, ya que permiten la depuración de programas de manera rápida y eficiente. Si no se tuviese esta herramienta se tendría que reprogramar el DSP cada vez se quiera realizar cualquier cambio sobre una variable, aunque sean variables que se ubiquen en la memoria volátil (RAM). Además esto permite alargar la vida de la memoria flash de manera considerable. El fabricante garantiza un mínimo de 1000 grabaciones, pudiendo llegar a ser más de 10000. Esto es más que suficiente para el desarrollo de uno y varios programas. Por otro lado existe la solución de utilizar unas placas de evaluación de Texas Instruments de las que se disponen en el laboratorio. Estas placas de evaluación disponen de memoria RAM exterior, esto permite ubicar el código del programa en una memoria volátil en vez de la memoria flash. Así se puede rescribir el programa infinitas veces y de forma mucho más rápida, además de otras ventajas a la hora de depurar el programa.

El programa Code Composer Studio dispone de una herramienta, el Real Time que permite visualizar los valores de variables del DSP en tiempo real y de una forma continua. Esta herramienta es por tanto también muy útil, ya que permite visualizar gráficas de una o más variable que se actualizan de forma automática, aunque con la limitación de que no funciona adecuadamente a frecuencias de refresco superiores a los 20Hz.

Para la programación del DSP existen principalmente dos posibilidades reales, una es programar en C/C++ y la otra es programar directamente en lenguaje ensamblador del DSP. Programar en C/C++ tiene la ventaja de que es un lenguaje de alto nivel, relativamente fácil de programar y ampliamente difundido. A esto se ha de sumar el hecho de que Texas Instruments dispone de una amplia gama de librerías con muchas funciones y utilidades ya programadas específicamente para el DSP. El gran problema que tiene los programas

escritos en C/C++, es que cuando se traducen al código máquina, resultan códigos poco eficientes.

Esta falta de eficiencia se puede solucionar escribiendo el programa en ensamblador. Desde este lenguaje la traducción a código máquina es directa, resultando programas muy eficientes. El gran inconveniente de programar en ensamblador es su complejidad, sobretodo por el gran número de instrucciones de que se dispone, y que estas son exclusivas de cada DSP, pudiendo cambiar incluso entre DSPs de la misma familia.

Cabe la posibilidad de hacer partes del programa en ensamblador y otras en C/C++, esto es lo que se ha hecho en este caso, realizándose las funciones más criticas en ensamblador, y aquellas que no lo son tanto en C/C++.

7.6 Montaje del sistema

Lo primero que se tiene que tener en cuenta en el montaje es que el control tiene que ser fácilmente accesible por diversos motivos:

- Para acceder a las salidas del conversor digital a analógico.
- Por que lleva gran cantidad de elementos conectados y es necesario que estos se puedan conectar y desconectar de manera fácil.
- Para poder inspeccionar las señales de diversos elementos durante la depuración.
- Para poder cambiar fácilmente cualquier elemento de la placa de control en caso de avería.

Además es necesario que el control esté apantallado electromagnéticamente de la parte de potencia, para evitar en lo posible las interferencias. Por esto y para compactar al máximo el sistema, se ha decidido poner el control encima del SKiiP montado sobre dos planchas, una de acero y otra de aluminio. Así, una apantalla el control frente campos magnéticos y otra frente a campos eléctricos.

Todo el prototipo se ha montado sobre una plancha de acero de 40cmx70cm con 4 patas, con una disposición como la que muestra el esquema siguiente:

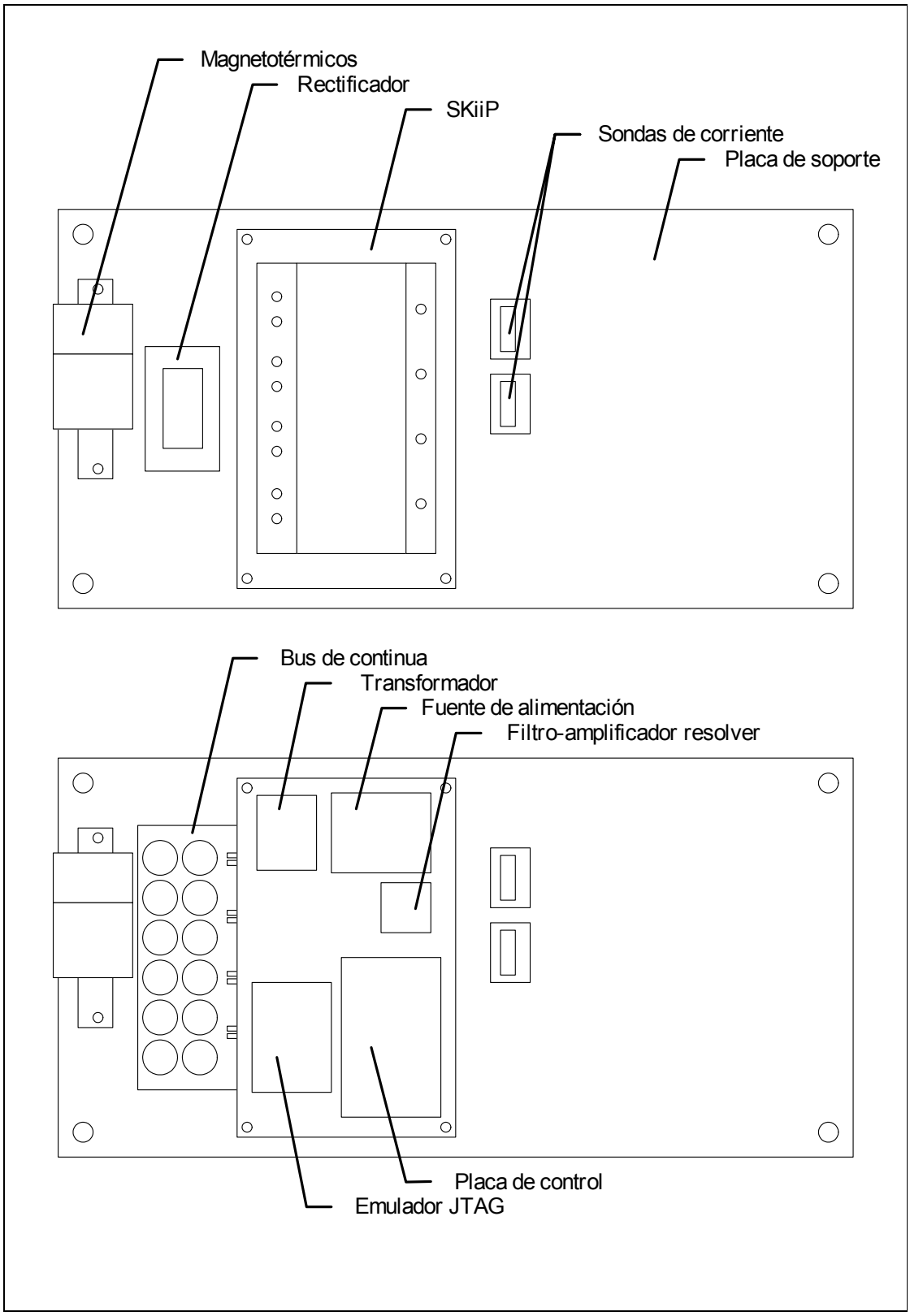

Ilustración 7.3: esquema de montaje del prototipo.

Para sujetar los interruptores magnetotérmicos se ha dispuesto un carril DIN, en el cual se pueden alojar otros elementos si fuese necesaria.

En el montaje se observa que sobre bastante sitio, este se puede utilizar en otros montajes para ubicar por ejemplo bobinas y condensadores de filtro a la salida del puente ondulador, o resistencias de frenado si estas fuesen necesarias.

7.7 Fotografías del montaje

A continuación se exponen algunas fotografías del conjunto para poder hacerse una idea de cómo ha quedado todo el montaje final:

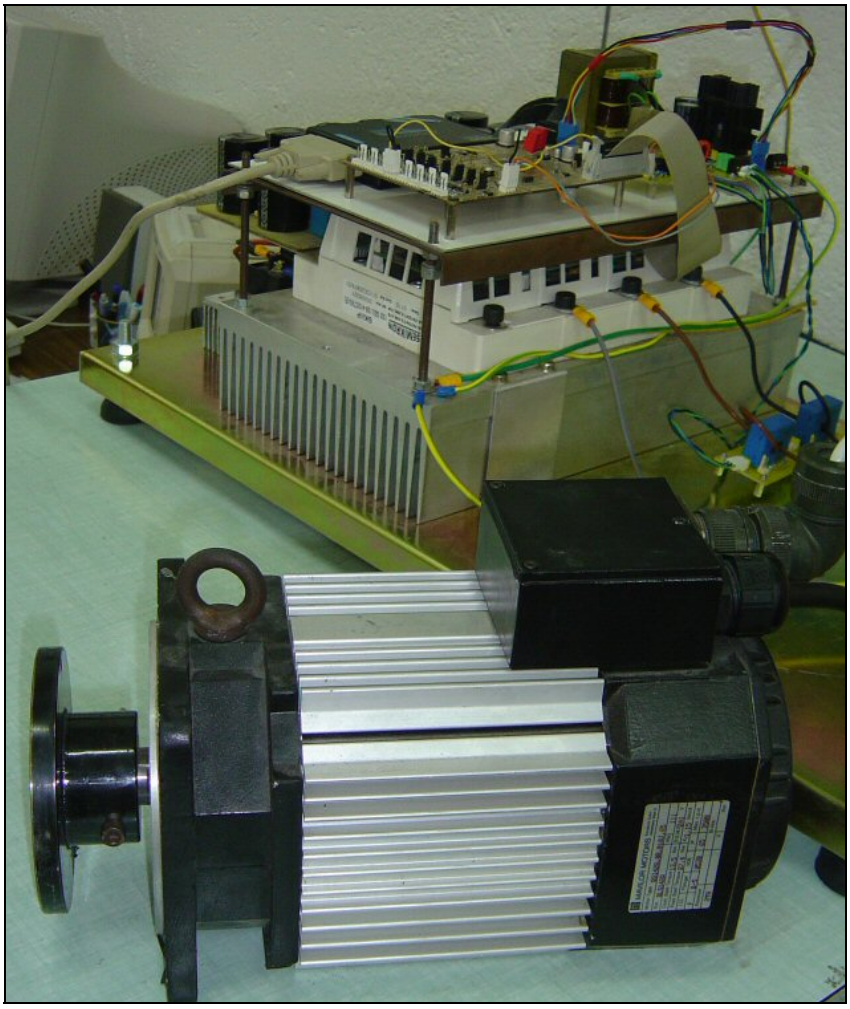

Ilustración 7.4: vista general del prototipo.

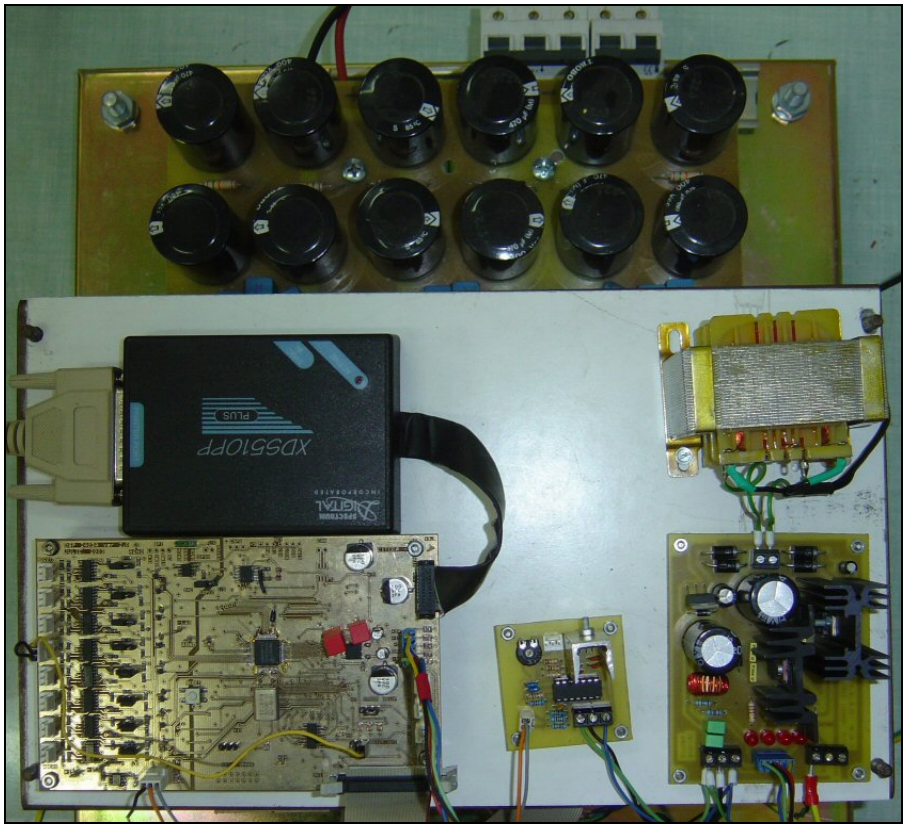

Ilustración 7.5: vista superior del prototipo.

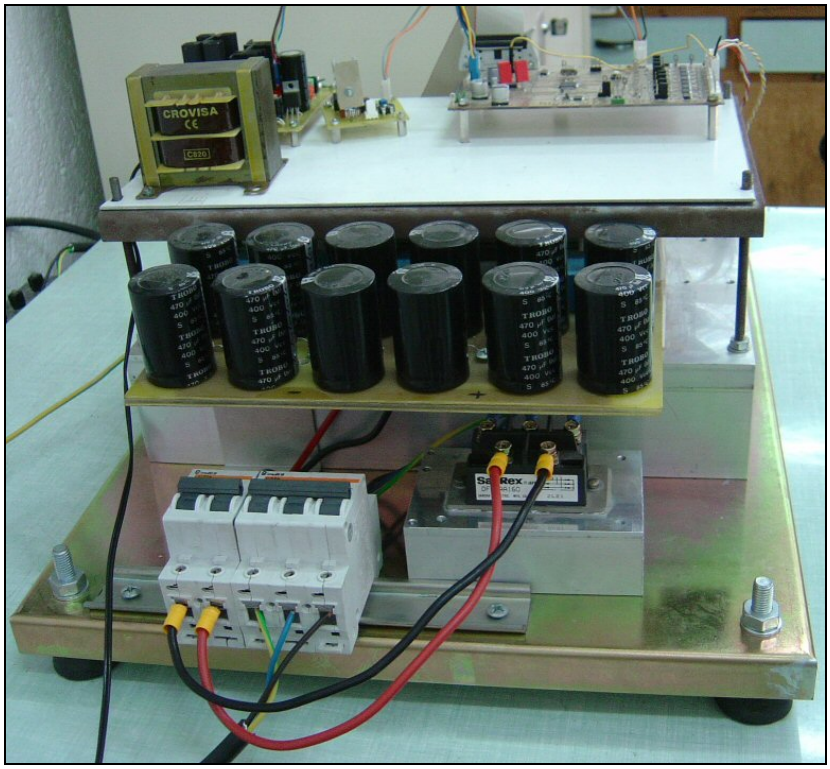

Ilustración 7.6: vista posterior del prototipo.

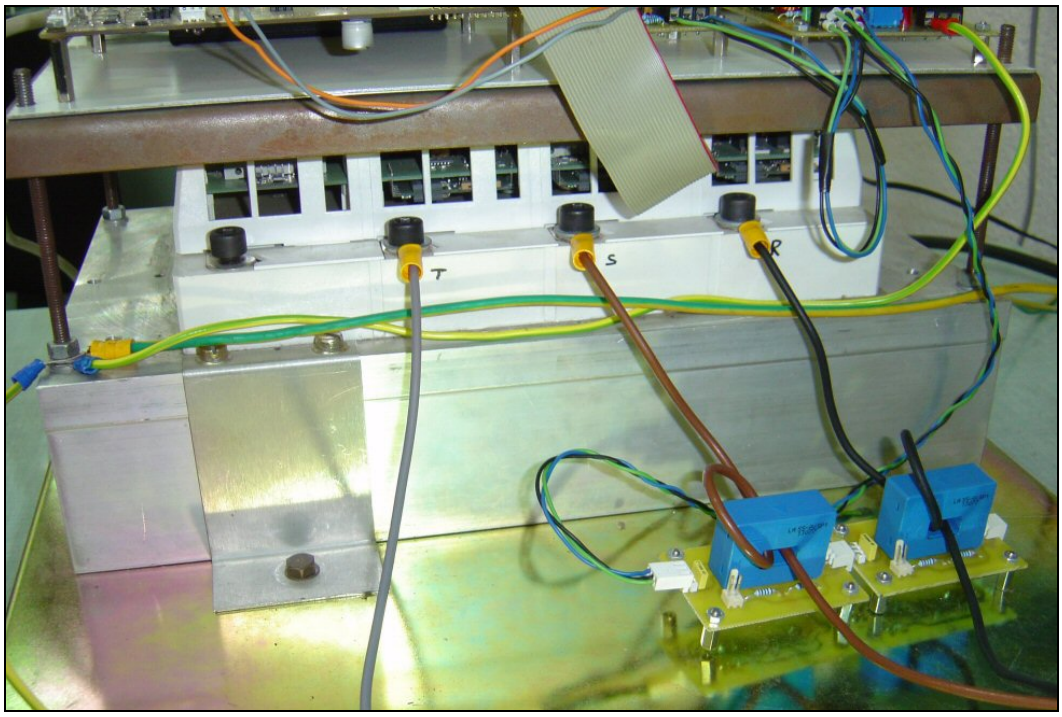

Ilustración 7.7: vista delantera del prototipo.

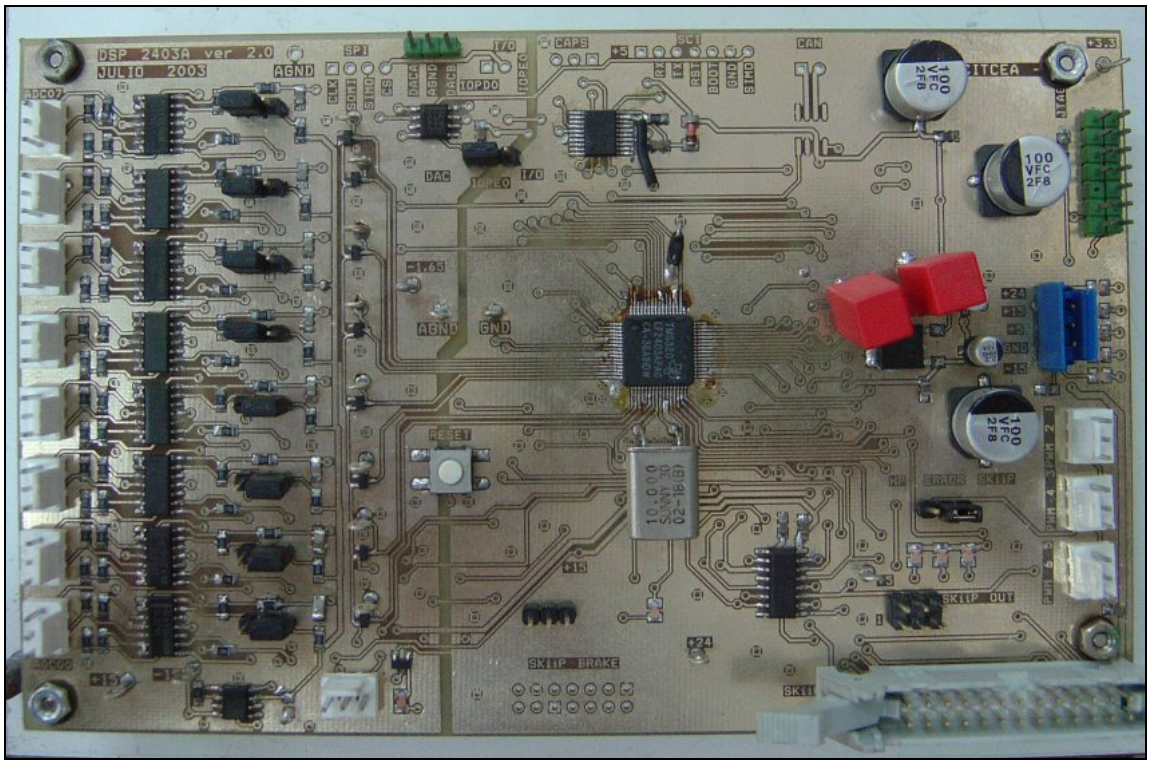

Ilustración 7.8: detalle de la placa de control.

8 Resultados experimentales

Una vez montado todo el prototipo y programado todo el control se ha pasado a evaluar los resultados experimentales. Estos resultados se pueden clasificar en dos apartados, por un lado las resoluciones conseguidas en cuanto al ángulo y en cuanto a velocidad, y por otro lado la respuesta de los bucles de control: tanto el de par (o intensidad) como el de velocidad.

8.1 Resoluciones conseguidas

Para ver la resolución que se ha conseguido se ha introducido una pequeña porción de código en el programa que se encarga de almacenar la variable deseada en un tabla cada cierto tiempo. Con esto se puede evaluar a altas frecuencias, y de forma exacta el valor de las variables. En principio se podría usar el DAC, pero este saca valores a una frecuencia relativamente baja y de forma no perfectamente síncrona debido a limitaciones del DSP. Además el DAC siempre introduce errores, esto se debe a que no está pensado para evaluar variables con precisión, sino de forma más bien cualitativa.

Así, se ha guardado valores del ángulo calculado usando el oversampling y la PLL a una frecuencia de 4500Hz, estando el motor parado en un cierto ángulo. Los errores respecto al valor medio son los que indica la Ilustración 8.1.

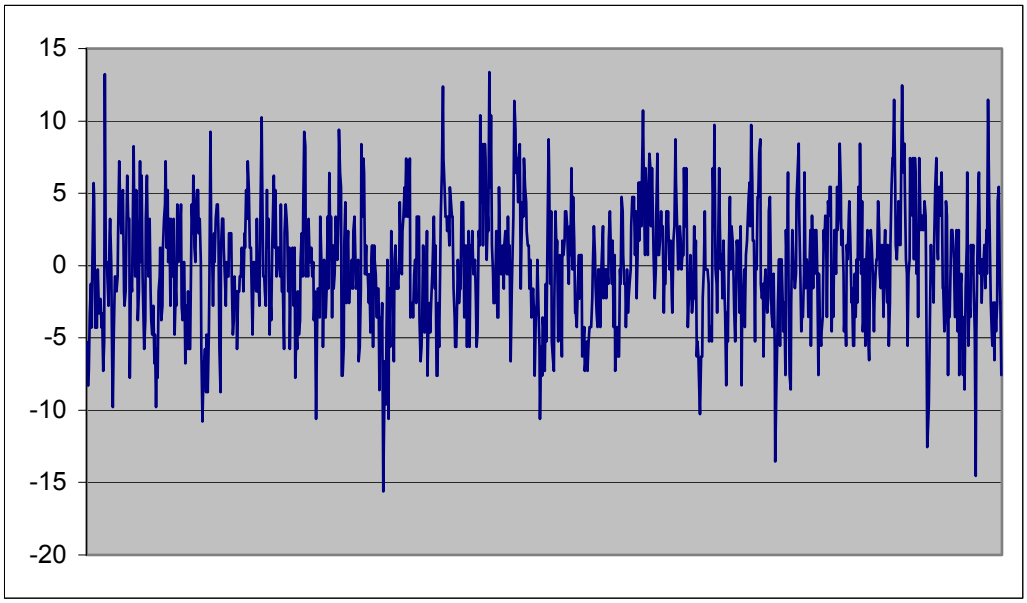

Ilustración 8.1: error respecto al valor medio del ángulo a velocidad 0.

Para evaluar los bits de resolución que represente este error se calculado la desviación estándar de este error y se ha procedido según:

$$2^b = \frac{Rango}{2 \cdot Desv} = \frac{65535}{2 \cdot Desv}$$
$$b = \frac{\ln\left(\frac{65535}{2 \cdot Desv}\right)}{\ln(2)} \qquad \text{Ec. 8.1}$$

La desviación estándar del error es de 6.48 sobre 65535, lo que representa unos 12.5bits de resolución. Esto es prácticamente lo que se ha previsto con las simulaciones hechas (12.4bits). Este valor tiene una fiabilidad limitada ya que se leído cuatro tandas de 256 valores. Esto se debe a limitaciones del DSP, sobretodo en cuanto a memoria se refiere, ya que no se pueden guardar muchos valores en una tanda. Para obtener una mayor fiabilidad se tendría que leer muchos más valores, pero debido a lo tedioso de este proceso se ha decidido hacer pocas tandas de lectura y obtener valores aproximados.

Por otro lado se ha hecho la lectura del ángulo calculado, usando oversampling y la función arcotangente. Con este método se consigue 11.3bits frente a los 10.5bits que se obtuvo con la simulación.

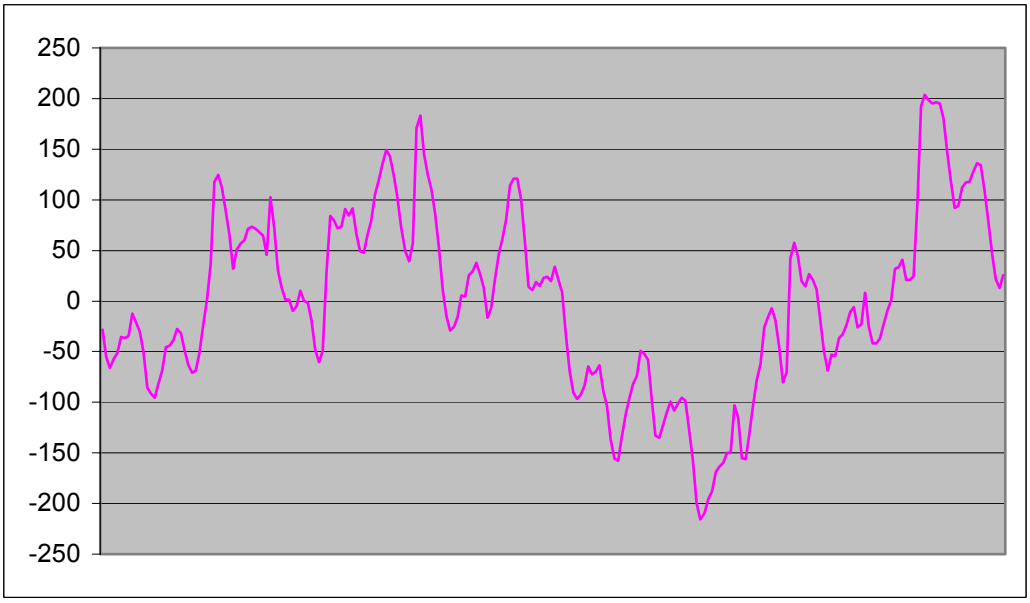

Ilustración 8.2: error respecto al valor medio del ángulo a 3000rpm.

Para una velocidad mecánica de 3000rpm se ha vuelto a hacer lecturas de la posición, observando una resolución de 8bits en el ángulo, frente a los aproximadamente 6bits que se

obtuvo en la simulación. Este valor todavía tiene menos fiabilidad ya que como se está en movimiento, el ángulo varía, y lo que se ha hecho es calcular el error respecto a la tendencia. Se observa que este error tiene una componente importante de aproximadamente 50Hz, provocada por la influencia de la red eléctrica.

Con todo esto se confirma que estas lecturas son de una fiabilidad limitada, pero aún así se entrevé que la resolución ha mejorado considerablemente con el oversampling, ajustándose de forma cualitativa a las simulaciones. Siendo la resolución en ángulo, del orden de los 12bits, frente a los 8.3bits que se obtendría con la tradicional técnica del undersampling.

8.2 Repuesta de de los bucles de control

Para evaluar la respuesta de los bucles de control también se ha introducido una pequeña porción de código, esta se dedica a cambiar la consigna cada cierto tiempo, con ello se puede ajustar los PID de los bucles y también evaluar la respuesta.

Cuando se activa solo el bucle de corriente, las consignas que se dan son las corrientes en coordenadas d-q (ver capítulo 6.2.1). Concretamente la consigna de I_q es 0 y la consigna de I_d se ha hecho variar entre -39A y 39A. Tanto la consigna como el valor real de I_d se han leído a través del DAC de DSP. Se observa que la corriente llega a su valor final en aproximadamente 7ms.

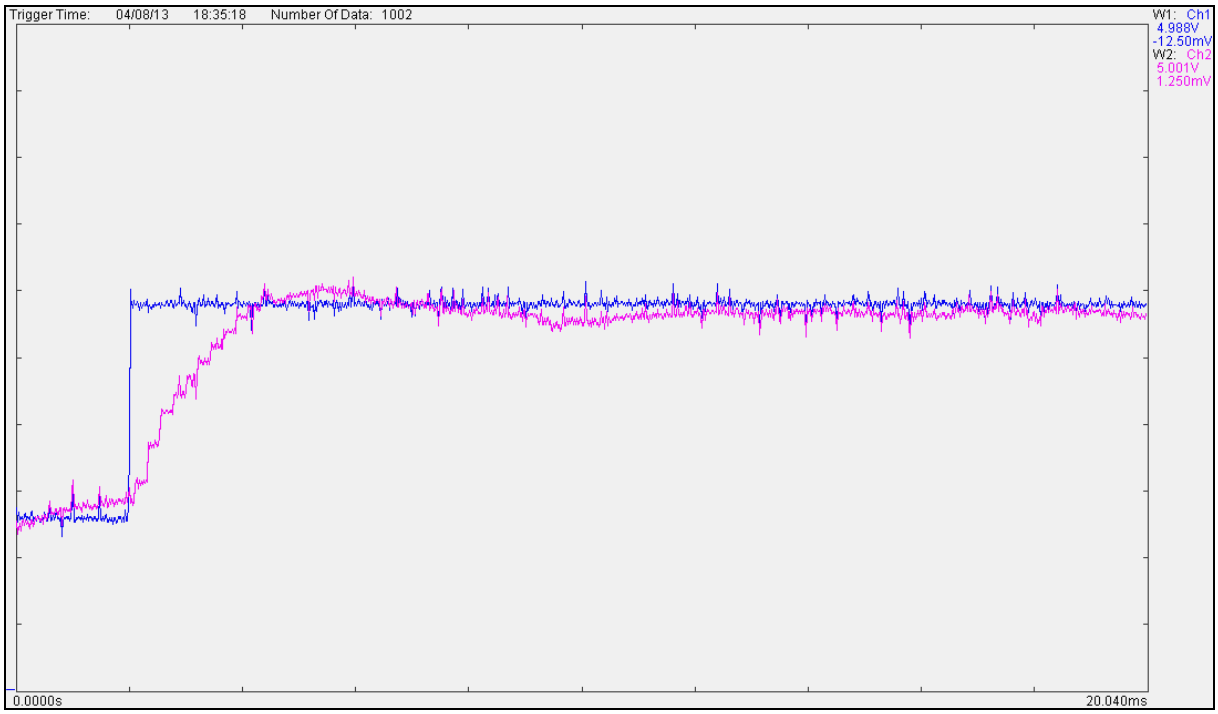

Ilustración 8.3: consigna (azul) y respuesta (lila) del control de corriente.

Para evaluar el control de velocidad se ha procedido de forma similar, aplicando una consigna que varia entre -3000rpm y 3000rpm.

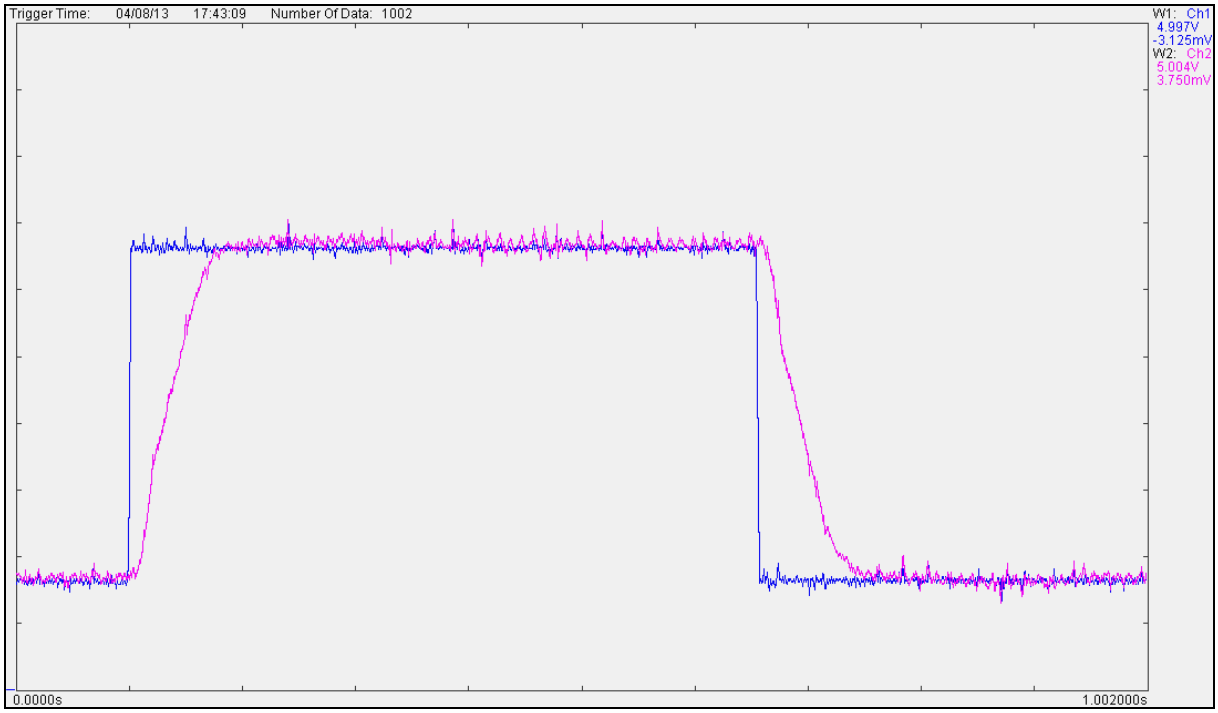

Ilustración 8.4: consigna (azul) y respuesta (lila) del control de velocidad.

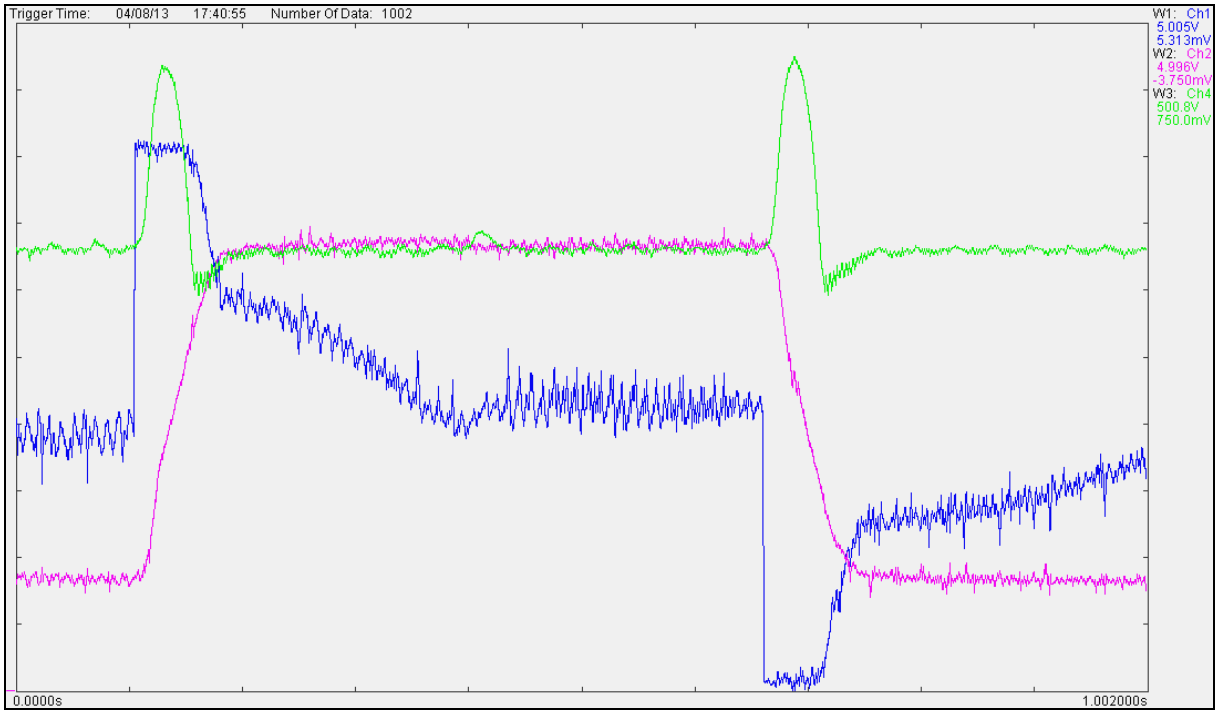

Ilustración 8.5: respuesta del control de la velocidad (lila), con detalle de la corriente (azul) y la tensión de bus (verde).

Se puede observar en la Ilustración 8.4, que la velocidad no responde todo lo rápido que se cabía esperar, esto se debe a que la corriente está limitada; no se deja aplicar más corriente de la que se puede leer, y por lo tanto controlar, esto se puede observar en la Ilustración 8.5. En el caso de dar una sola vuelta a las sondas esta corriente máxima es de 50A. Con este límite, la respuesta es bastante rápida, se hace un salto de 6000rpm en aproximadamente 75ms, esto puede ser bastante más rápido si se puede disponer de más corriente, es decir, sondas de mas corriente y un bus de más tensión.

También se puede observar en la Ilustración 8.5 una crecida de la tensión de bus bastante importante en los momentos de cambio de sentido (varia de 330V a 460V en el máximo), es por ello que es tan importante haber independizado el control de la tensión del bus.

8.3 Otros resultados

En este apartado se de cabida a diversos otros resultados que no se engloban en los dos apartados anteriores.

Por una parte están los tiempos que tarda cada parte del algoritmo en ejecutarse. Este puede parecer un dato poco relevante, pero en realidad lo es mucho, ya que si los algoritmos son demasiado largos, quedará poco tiempo para ejecutar los algoritmos de control principales, esta es además, la razón por la que los dos módulos principales del programa (la interrupciones ADC_int y SVPWM_Int) se han escrito en ensamblador. Para evaluar esto se ha contado las instrucciones de cada tipo y los ciclos necesarios para ejecutar los módulos principales. Esto no es una tarea trivial, ya que las instrucciones pueden necesitar un número diferente de ciclos según como se usen o en que memoria se encuentran las variables que intervienen en la instrucción. Además el programa no se ejecuta de una manera lineal, de manera que siempre tarde lo mismo, sino que según las variables se ejecuta de una manea u otra. Por eso se ha hecho un recuento aproximado, ya que tampoco es el objetivo saberlo con total exactitud.

Teniendo en cuenta todo esto se ha recontado que el algoritmo de adquisición del ángulo y la velocidad (ADC_Int) necesita aproximadamente un 24.3% del tiempo de cálculo disponible por la CPU. En cambio el algoritmo encargado de hacer el control de motor propiamente dicho (SVPWM_Int) ocupa una 8.3% si se ejecuta un control de velocidad, un 8.0% si se ejecuta un control de par y un 6.5% si se ejecuta un control de tensión. Por consiguiente toda la parte básica del control ocupa menos de una tercera parte del tiempo de cálculo disponible. Se ha de tener en cuenta que los dos tercios de tiempo de cálculo sobrantes no son consecutivos, ya que ambas interrupciones interrumpen a una frecuencia

bastante alta; el ADC_Int a 72kHz y el SVPWM_Int a 9kHz y ambos con la prioridad máxima sobre el resto de código, por lo que si se quiere ejecutar código entremedio siempre tendrá que tener en cuenta se puede ejecutar de forma no perfectamente síncrona.

Conclusiones

Como conclusión se puede decir que los objetivos del proyecto se han cumplido con éxito. Se ha montado un prototipo de un ondulador trifásico con un control para motores brushless que funciona adecuadamente. Además se ha conseguido una mejora en la resolución únicamente con el software, sin la necesidad de un costoso chip resolver-to-digital. Ahora la resolución del ángulo es del orden de los 12bits, lo cual es más que suficiente para el control de motores. De paso la velocidad también ha sufrido una mejora en resolución.

Por otro lado, tanto la placa de control como el prototipo entero se han montado lo suficientemente modular para permitir transformarla fácilmente para su uso con otros fines, por ejemplo como ondulador o como rectificador controlado.

En lo que respecta la placa de control, se ha conseguido una placa de control de muy bajo coste y gran potencia. Esta ha demostrado su robustez durante todo el periodo de pruebas. El DSP que lleva montado también ha demostrado un comportamiento excepcional, pudiendo llegar ejecutar bastantes más algoritmos en paralelo de los que ejecuta solo con el control, ya que todo el algoritmo creado ocupa solamente un 33% del tiempo de cálculo disponible. Con esto se puede crear controles de motores con algoritmos relativamente complicados, cosa que no se puede llegar a hacer con variadores comerciales típicos.

Quedan sin embargo algunas mejoras que se pueden hacer en un futuro. Una de ellas es programar el control de posición del motor. En principio se pensó hacer, pero por falta de tiempo y por estar fuera de los objetivos iniciales, finalmente se ha dejado. Otra mejora posible es una interfaz, que permita a una persona o un PLC controlar el motor de forma remota, esto se podría conseguir fácilmente a través del bus CAN, o también a través del puerto RS232 (el puerto serie del PC).

Como dato final, se ha de comentar que el control creado se está implementando en un proyecto real, se trata de prototipo de una boya para generar energía eléctrica con el movimiento de las olas. Esta aplicación requiere el control de la velocidad y la corriente del motor de una forma relativamente complicada para poder aprovechar correctamente la energía de las olas.

Agradecimientos

Para finalizar quiero dar las gracias a todas aquellas personas que me han ayudado en este proyecto, ya que sin su ayuda este proyecto no hubiera sido posible.

A Joan Bergas, director del proyecto, por su gran ayuda desde el principio, por la aportación de múltiples buenas ideas, y sobretodo por todas las horas y el esfuerzo que ha dedicado a este proyecto.

A Jordi Motgé por la enorme ayuda que me ofrecido en el desarrollo de la placa del DSP y también a la hora de programar.

A Antoni Sudriá por su gran apoyo en todo momento y sus valiosos consejos y orientaciones que me ha dado durante todo el proyecto.

A Miquel Teixidor por la ayuda continua que me ha prestado y sobretodo por estar ahí siempre que lo he necesitado.

Al resto de profesores, investigadores y becarios del Departamento de Ingeniería Eléctrica de la ETSEIB por las múltiples ayudas prestadas.

A mis padres por el soporte incondicional que me han prestado, gracias al cual este proyecto finalmente ha llegado a buen puerto.

Bibliografía

Referencias bibliográficas

[1] Coughlin & Driscoll, “Operacional Amplifiers and Linear Circuits”.

[2] Joan Bergas, “Control del motor d’inducció considerant els límits del convertidor i del motor”, mayo 2000.

[3] Faulkenberry L. M., “Introducción a los amplificadores Operacionales con aplicaciones a CI lineales”, Noriega Editores.

[4] Luis Sainz Sapera, Felipe Córcoles López, Joaquin Pedra Durán, “Fundamentos de máquinas eléctricas”, Ed. CPDA, 2001.

[5] H. W. Van Der Broek, H. Skudenlny, G. V. Stanke, “Analysis an Realization of a Pulsewidth Modulator Base on Voltage Space Vectors”, IEEE Transactions on Industry Applications, Vol. 24, No. 1, 142-150, enero/febrero 1988.

[6] John G. Proakis, Dimitris G. Manolakis, “Tratamiento digital de señales” 3ª edición, Prentice Hall, 2000.

[7] Martin Staebler, “TMS320F240 DSP Solution for Obtaining Resolver Angular Position and Speed”, Texas Instruments Application Report, febrero 2000.

[8] Muhamed H. Rashid, “Electrónica de potencia, circuitos, dispositivos y aplicaciones” 2ª edición, Prentice Hall, 1995.

[9] Peter Moretor, “Industrial Brushless Servomotors”, Newnes, 2000.

[10] Richard M. Crowder, “Electric Drives and their controls”, Oxford Science Publications, 1995.

[11] Sudriá, Antoni, “IGBT: más cerca del interruptor ideal”, MEI, NO11, mayo 1992.

[12] Texas Instruments, “Oversampling Techniques using the TMS320C24x Family”, junio 1998.

[13] Texas Instruments, “3.3 DSP for Digital Motor Controls”, junio 1999.

[14] Texas Instruments, “PCB Design Guidelines form reduced EMI”, Noviembre 1999.

[15] Texas Instruments, “TMS320F/C DSP Controllers Reference Guide: CPU and Instruction Set”, junio 1999.

[16] Texas Instruments, “TMS320F/C DSP Controllers Reference Guide: System and Peripherials”, diciembre 2001.

[17] T.J.E. Miller, “Brushless Permanent-Magnet an Reluctance Motor drives”, Oxford Science Publications, 1989.

[18] Yasuhiko Date, Sakon Kinoshita, “Brushless Servomotors, fundamentals and applications”, oxford science publications, 1990.

Otras referencias bibliográficas

Onda Radio. Catálogo on-line. http://www.ondaradio.es

RS Amidata: Catálogo on-line. http://www.amidata.es

Texas Instruments. Fabricante. http://www.ti.com