ANEXOS

A.COMPARATIVA MSP430 Y MSP432

Como parte de este trabajo se adjunta este anexo que de forma breve explica las diferencias que son destacables entre ambos microcontroladores. De esta forma se podrá tener una perspectiva mayor de las ventajas que trae este nuevo microcontrolador [22].

CPU Y NÚCLEO

Aunque ambos microcontroladores se pueden percibir como dos dispositivos distintos como se puede comprobar en la tabla 1 siguiente ambos comparten similitudes respecto a su arquitectura.

Name	MSP430™	MSP432™
Data size	16 bit	32 bit
Program bus width	16-bit (CPU) or 20-bit (CPUX) address bus	32 bit
Bus type	16-bit MSP430 bus	AHB
Architecture	Von Neumann (Princeton): data op and instruction fetch share same bus	Harvard: separate data and instruction buses
Instruction Set	RISC, MSP430 proprietary	RISC, thumb and thumb2
Instruction size	16-bit (and 16-bit for each operand)	16-bit and 32-bit
Instruction Cycle (typical)	1-4 cycles	1-2 cycles
Pipeline	None	3-stage pipeline
Prefetch buffer	128 bit	128 bit
Power Modes	Active, LPM0-LPM4, LPMx.5	Active, Low Frequency, LPM0, LPM3, LPMx.5
Debug Interface	MSP430 4-wire JTAG, and 2-wire SBW	ARM JTAG in 4-wire and 2-wire modes
Math support	Hardware Multiplier (MPY)	Hardware Multiplier and Divider, DSP extension, and integrated FPU

Tabla 1 Comparativa de núcleos de MSP430 Y MSP432

Como ya se expuso previamente en la memoria, las arquitecturas son *Harvard* y *Von Neuman* respectivamente, lo que supone una mejora ya en el procesamiento para datos e instrucciones. La CPU del MSP432 es de 32 bits con respecto a los 16 bits del MSP430 aumentando así el tamaño de los registros de la memoria. El rendimiento del microprocesador se ve mejorado de los 1-4 ciclos por instrucción a los 1-2 del MSP432. Se amplían los modos de bajo consumo introduciendo los nuevos modos de Active y Low Frecuency ambos diseñados para trabajar a frecuencias reducidas. Por último y como se ha indicado en la memoria, la unidad de coma flotante que mejora sustancialmente la velocidad de cálculo del procesador con respecto al MSP430.

A continuación tabla 2 con los atributos del sistema de ambos microcontroladores.

Parameter	MSP430™ MCUs	MSP432™ MCUs
Supply voltage range	1.8 V to 3.6 V	1.62 V to 3.7 V
Analog supply voltage	1.8 V or 2.2 V to 3.6 V	1.8 V to 3.7 V
Maximum system frequency	8, 16, 20, or 25 MHz	48 MHz
Nonvolatile (flash or FRAM) memory	512 bytes to 512K bytes	Up to 256K bytes
RAM memory	128 bytes to 64K bytes	Up to 64K bytes

Tabla 2 Comparativa del Hardware entre MSP430 Y MSP432

Como se puede comprobar el MSP432 permite operar en un rango mayor de tensión de alimentación. Mientras que en el MPS430 el sistema de reloj tiene 4 frecuencias calibradas disponibles, en el MPS432 la frecuencia del reloj es modificable en el rango entre 1.5 MHz y 48 MHz, permitiendo así operar a frecuencias que se adapten mejor a los requisitos de trabajo. La familia del MSP432 incluye nuevas características para el sistema de alimentación entre ellas el doble regulador (LDO y DC-DC) que genera dos niveles de tensión suministrada a la CPU (Vcore). Tanto el regulador de frecuencia del Vcore como los reguladores seleccionables son algunas de las características que dan una mayor flexibilidad a la hora de optimizar el sistema de potencia y el consumo del sistema.

En cuanto a los modos de bajo consumo en los MSP430 los modos de bajo consumo van del LPM0 al LPM4 recientemente ampliados a LPM3.5 y LPM4.5. En donde cada nivel ofrece unos niveles de reloj distintos y de disponibilidad de los periféricos. El MSP432 mantiene algunos de los modos de bajo consumo del MSP430 y se extiende a dos más que serán útiles para trabajar a bajas velocidades. En la tabla 3 se ve la correlación entre los distintos modos.

MSP430™ MCUs	MSP432™ MCUs	Industry Description	Comments
Active	Active	Active Mode	CPU and peripherals
	Low-Frequency Active	Low-Power Run	CPU and peripherals <128 kHz
LPM0	LPM0	ARM: Sleep	Peripherals on, CPU off
	Low-Frequency LPM0		Sleep + CLK < 128 kHz
LPM1	N/A	MSP430-specific mode	
LPM2	N/A	MSP430-specific mode	
LPM3	LPM3	ARM: Deep-sleep Standby with RAM and RTC	A/BCLK, <32 kHz, some peripherals available
LPM4	N/A	Standby with RAM	No clocks, some peripherals available
LPM3.5	LPM3.5	ARM: Shutdown	RTC without RAM
LPM4.5	LPM4.5	ARM: Shutdown	Shutdown

Tabla 3 Comparativa de modos de bajo consumo entre MSP430 Y MSP432

RELOJ

El módulo de reloj se extiende hasta las 4 señales de reloj para altas y bajas frecuencias. Se puede hacer uso de varias fuentes de reloj internas y externas con diferentes frecuencias y grados de precisión para suministrar las señales de reloj. Estas fuentes se pueden utilizar para dar soporte a la CPU (MCLK) y a los periféricos. De esta forma el MSP432 tiene un sistema de reloj similar al del MSP430. Para aplicaciones que requieren una fuente de reloj interno de alta frecuencia y precisión, se puede hacer uso del DCO mejorado que puede

operar hasta los 48 MHz, de baja fluctuación y capaz de aumentar su precisión con una resistencia externa.

PERIFÉRICOS

En cuanto a los periféricos comparando ambas familias se puede afirmar que funcionan del mismo modo, a excepción del NVIC para el procesado de interrupciones del MSP432 y de algún periférico que recibe alguna modificación y mejora como el ADC14 que es una versión del ADC12_B del MSP430 que disponía de resolución de 12 bits frente a los 14 bits del ADC14 y los 200 Ksps del MSP430 frente a los 1 Msps del ADC14. Además como derivados de la arquitectura ARM se extienden las funcionalidades con nuevos periféricos como el Timer32 que puede generar interrupciones cuando su contador llega a cero y que incluye:

- Dos temporizadores independientes, cada uno configurable como contador de 32 o 16 bits.
- Tres modos de operación para cada contador.
- La unidad iPrescale que divide el reloj de entrada entre 16 o 256.
- Interrupciones independientes para cada uno de los contadores así como una interrupción combinada de ambos. que puede producir interrupción cuando su contador descendente llega a 0.

Los tres modos de operación para este Timer son:

1. Free-reuning: El contador llega a cero y cuelve a contar desde el valor máximo. Este modo está configurado por defecto.
2. Periodic timer: El contador genera una interrupción en un intervalo constante de tiempo, una vez llega a cero recarga el valor inicial.
3. One-shot timer: El contador genera una única interrupción. Una vez el contador llega a cero se mantiene en espera hasta que se lo reprograma.

INTERRUPCIONES

Las interrupciones para ambas familias de microcontroladores tienen diferencias significativas. Es por ello que a la hora de portar una aplicación de una plataforma a otra se tendrán que tener en cuenta varias consideraciones a la hora del manejo de sistema de interrupciones. En el MSP430 las fuentes de interrupciones periféricas están unidas al sistema central de interrupción el cual se controla mediante el bit GIE del registro SR En cambio el MSP432 hace uso del vector de interrupciones anidadas NVIC que proporciona más flexibilidad a la hora de configurar prioridad, eficiencia al encadenar interrupciones y un control individual de las interrupciones en los periféricos.

En el MSP430 el compilador maneja la asignación de espacio de memoria del vector de interrupciones, reserva espacio para los vectores de interrupción no usados y registra los vectores de interrupción utilizados detectando la rutina de servicio de interrupción especial (ISR) mediante el código de la figura A.1.

```
#pragma vector = USCI_B0_VECTOR
__interrupt void USCI_B0_ISR(void)
{
  switch(__even_in_range(UCB0IV,12))
  {
  case  0: break;
  ......
}
```

Figura A.1 Código para identificación de ISRs en el MSP430

Para el MSP432 serán necesario los tres pasos que se observan en al figura A.2.

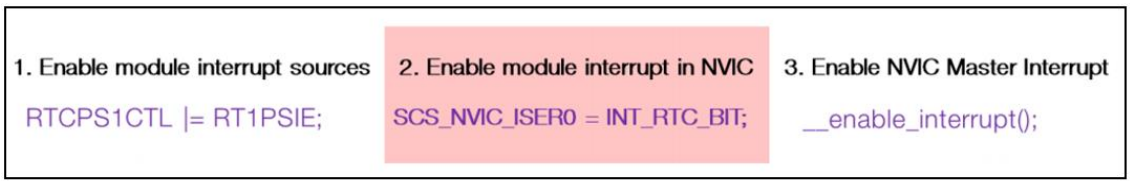

Figura A.2 Habilitación de las interrupciones en el MSP432

En primer lugar se habilitan las fuentes de interrupción, como paso extra se ponen a uno los bits del registro ISER0 o ISER1 que habilitan el módulo de interrupción del NVIC. Y finalmente se habilitan las interrupciones en el NVIC mediante *_enable_interrupt()*. El vector de interrupciones se define como un vector cuya dirección de inicio está fijada en 0x00000000. Como último paso se escribirán en el archivo *msp432_startup_ccs.c* generado por el compilador las ISR en sus posiciones correspondientes en el vector de interrupciones.

B. MODOS DE BAJO CONSUMO

El siguiente apartado se explica con el objetivo de minimizar el consumo de energía y analizar los variados modos de bajo consumo de que puede hacer uso el MSP432. En primer lugar decir que el MSP432 cuenta con unos periféricos de bajo consumo de corriente, modo de bajo consumo cuando está en activo y modo de bajo consumo de corriente. A continuación se enumeran varias formas de reducir el consumo [23]:

- Disminuyendo la tensión de operación: Siempre que esta no afecte directamente a la tensión mínima de alimentación del MSP432 que es 1.62V o dependiendo de la frecuencia de trabajo de la CPU el voltaje mínimo requerido por los periféricos.
- Reduciendo la frecuencia de operación.
- Maximizando el tiempo de *Sleep*: Esto se hace minimizando el tiempo en activo y aumentando el tiempo en los modos de bajo consumo.

EL MSP432 utiliza una tensión secundaria (Vcore) que da soporte a las operaciones digitales internas acompañada por la tensión de alimentación propia del dispositivo (Vcc). Vcore da soporte a la CPU las memorias flash y RAM y otros módulos digitales. Esta tensión es programable, y se pueden utilizar dos niveles de tensión que están restringidos individualmente por la frecuencia de operación. Específicamente para el MSP432 si la frecuencia máxima de trabajo ≤24 MHz el VCORE0 con una tensión de 1.2V es recomendable en comparación con el VCORE1 de 1.4V.

Asimismo para generar esta tensión secundaria Vcore partiendo del primario DVCC, el MSP432 cuenta con dos reguladores el *low-dropout voltage regulator (LDO)* o regulador de voltaje de baja caída que está por defecto configurado, y el *DC-DC*. Si bien el *LDO* tiene ciertas ventajas como, prácticamente nulo ruido de salida o rápidas transiciones de entrada y salida a los modos de bajo consumo este tiene un elevado consumo de corriente, lo que lo diferencia del *DC-DC*. En la tabla 1 se muestran los consumos de corriente relativos a cada regulador según frecuencia del procesador.

Regulator	CPU Frequency = 24 MHz	CPU Frequency = 48 MHz
LDO	3950 µA	7600 µA
DC-DC	2200 µA	4600 µA

Tabla 1 Tabla de consumo de corriente de LDO y DCDC

De los modos de bajo consumo que ya tiene incorporados el MSP430 como son LPM0, LPM3, LPM4 y los recientemente incorporados LPM3.5 y LPM4.5 se extiende este grupo a dos modos más AM_LF, LPM0_LF, especiales para baja velocidad de ejecución.

Estos modos de baja frecuencia son modos especiales de baja frecuencia y consumo que se pueden utilizar con los modos de LPM0 y activo. El máximo de frecuencia al que pueden trabajar memoria y periféricos en estos modos se reduce a 128 Khz. Esto en conjunto con el ajuste del regulador puede llevar a que tengamos un consumo total del dispositivo por debajo de los 80µA. En estos modos todos los periféricos son plenamente funcionales y la única diferencia que existe entre ambos es que en el modo LPM0_LF la CPU estaría

desactivada. Para programar todos estos modos de trabajo se necesita acceder al registro de PCMCTL0.

Los procesadores de ARM Cortex tienen tres instrucciones para poner al procesador en los modos de bajo consumo y son las de *WFI, WFE* y una adicional que se llama *Sleep-on-exit.*

- WFI (Wait for interruption): Esta instrucción suspende inmediatamente la ejecución de código. Con el registro PRIMASK= 0 espera a que se produzca una interrupción, y si esta tiene mayor prioridad que la de prioridad actual la interrupción se produce, en caso de que sea de menor prioridad, el procesador permanece en *Sleep.* Si el registro PRIMASK = 1 la interrupción sacaría al procesador del estado *Sleep* pero la interrupción no sería atendida.
- WFE (Wait for event): Sacar al WFE del Sleep depende del bit SEVONPEND del registro SCR. Con este bit a 1 cualquier evento, interrupción incluso aquellas que no están habilitadas pueden despertar el procesador. En cambio si este bit está a 0 solo aquellos eventos e interrupciones habilitados lo harán.
- Sleep on Exit: El ARM tiene este modo que permite al procesador volver al modo Sleep después de que la ISR haya atendido una interrupción. Para activar esto, es necesario poner el bit SLEEPONEXIT = 1 en el sistema de control SCR. De esta forma el procesador solo está activo cuando se atiende a una interrupción.

Si se quisiera activar cualquiera de los modos LPM3, LPM3.5, LPM4, LPM4.5 haría falta activar un bit que se encuentra en el registro SCR llamado SLEEPDEEP, que indica al sistema que el reloj del Cortex-M4 se puede parar.

Para el cálculo de la energía consumida por el micro se puede hacer uso de la herramienta de *EnergyTrace+ Technology* disponible en el programa de *Code Composer Studio*, la cual está desactivada por defecto [24]. Existen dos modos de funcionamiento. El primero, el *EnergyTrace* que simplemente determina la energía consumida por el micro en una aplicación y el *EnergyTrace+* que además de esto proporciona información en tiempo real de los estados del micro.

En primer lugar se prueba el modo de bajo consumo *Sleep-on-exit* con las ISR *velocity* y *Port4_ISR.* Como se puede observar en la figura B.1 el *EnergyTrace* me da la información referente al porcentaje de tiempo que el micro a utilizado en cada interrupción así como el tiempo en bajo consumo. Que en este caso corresponde al 95% de tiempo en bajo consumo.

EnergyTrace+™ Profile (Relative Measurement)

Name	Runtime (%)	Energy (%)
System	100	100
◢ CPU		
◢ Low Power Mode	95,0	95,5
LPMx	95,0	95,5
◢ Active Mode	5,0	4,5
port4_ISR	4,4	3,9
velocity	0,5	0,6

Figura B.1 Energy Trace TM, tiempo de activación de las funciones

En la figura B.2 se observan las transiciones que hay de activo al modo de bajo consumo en el intervalo de 10 segundos.

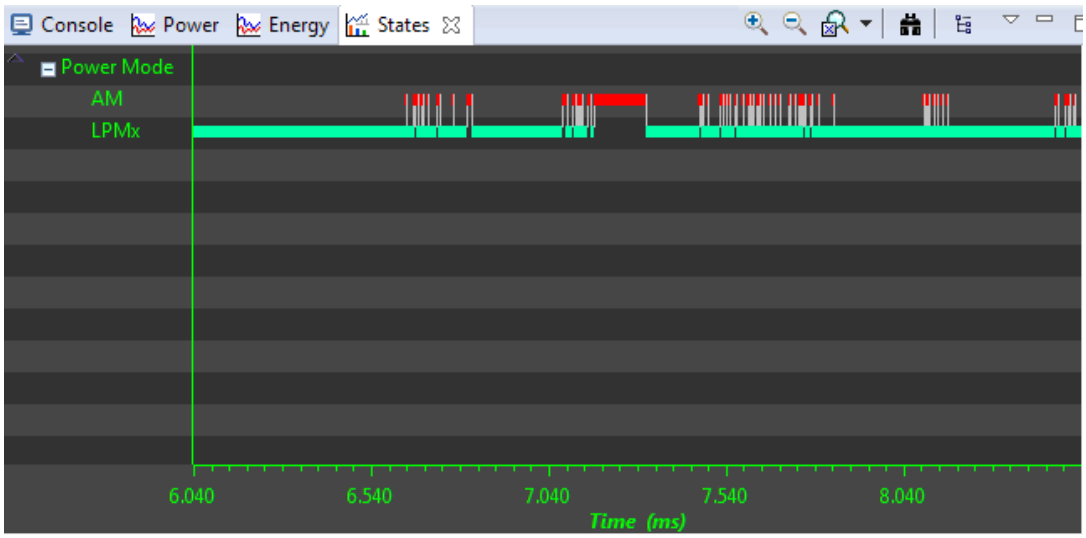

Figura B.2 Gráfica de transiciones de los modos de bajo consumo y activo

A modo de probar la resolución de estas transiciones he configurado el Timer2 como fuente de interrupción al desbordar el contador y de está forma ver que si es capaz de representar todas las transiciones. A continuación en la figura B.3 se ve el *EnergyTrace* de esta nueva configuración donde se puede observar que la mayor parte del tiempo el procesador permanece en estado activo.

EnergyTrace™ Technology

EnergyTrace+™ Profile (Relative Measurement)

Name	Runtime (%)	Energy (%)
System	100	100
◢ CPU		
◢ Active Mode	95,1	95,1
velocity	91,1	91,2
timer2	3,6	3,6
port4_ISR	0,3	0,3
◢ Low Power Mode	4,9	4,9
LPMx	4,9	4,9

Figura B.3 Energy Trace TM, tiempo de activación de las funciones con función timer2

Debido al elevado número de interrupciones que se producen durante estos 10 segundos, el tiempo que la CPU se encuentra en *Sleep* es muy bajo a esto hay que sumarle que la frecuencia a la que toma datos el *EnergyTrace+* es de 4Khz la cual no es suficiente para captar todas las transiciones entre los modos de activo y bajo consumo y que además suele funcionar de forma asíncrona con el depurador lo que puede hacer como se ve en la figura B.4 que se produzcan solapamientos o periodos donde parezca que no cambia de estado.

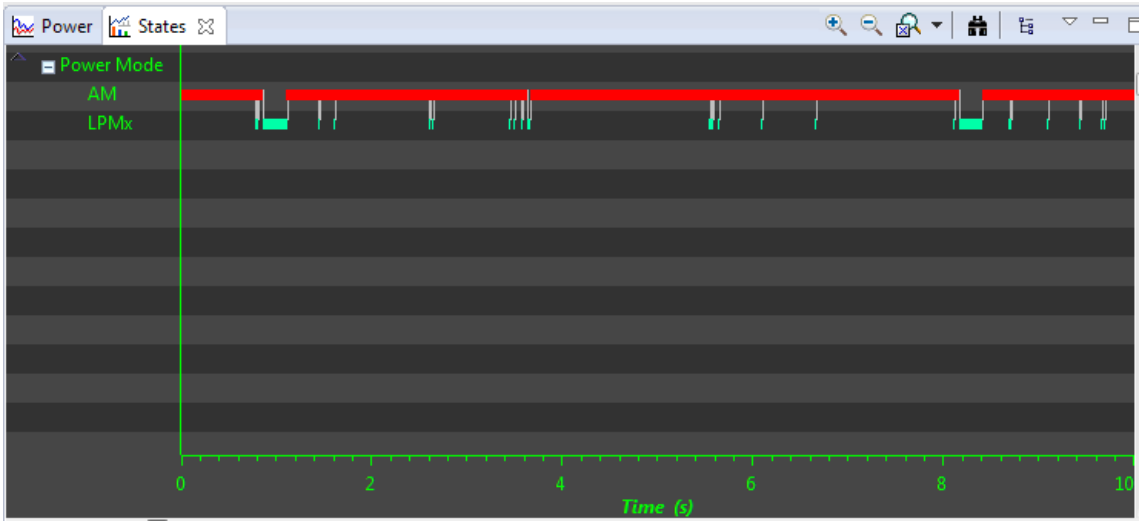

Figura B.4 Gráfico de las transiciones a modos de bajo consumo

Asimismo, se puede ver el consumo total durante el tiempo estipulado de toma de datos, como se muestra en la figura B.5 donde se puede observar el consumo tras una disminución de tensión de referencia, que afecta como es lógico al consumo total.

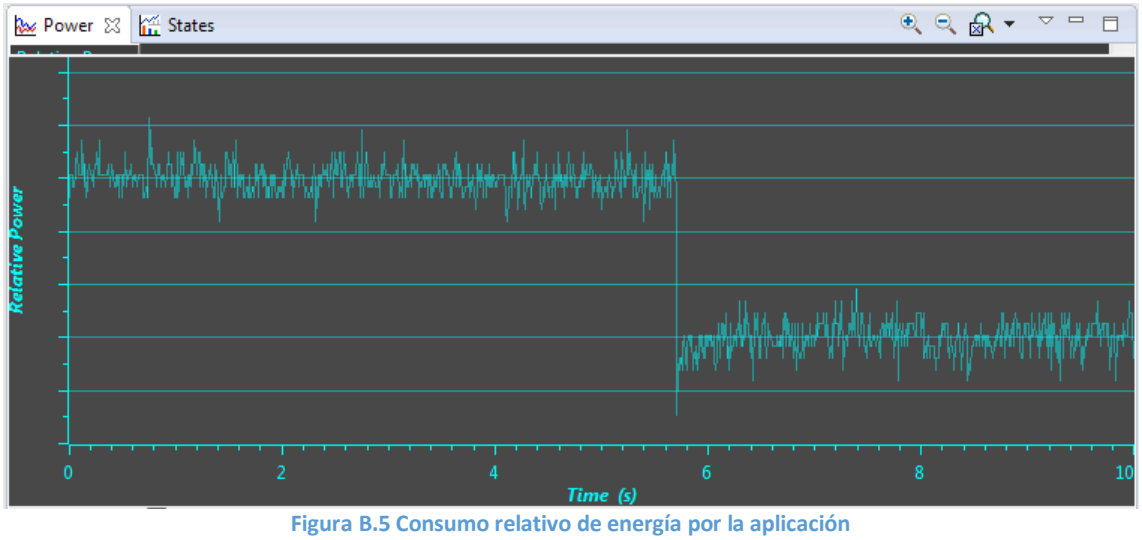

Figura B.5 Consumo relativo de energía por la aplicación

Los valores absolutos de consumo de energía en el eje vertical no se observan, ya que durante la captura de los estados internos se accede constantemente al micro por la lógica de depuración JTAG o Spy-Bi-Wire los cuales consumen energía

C. Controlador lineal-no lineal

A modo de estudio de otro método para el control del sistema se aporta este anexo. Como se observa en la memoria las especificaciones para el cálculo del controlador son en el espacio frecuencial de esta forma y para darle otro punto de estudio los requisitos de la respuesta del sistema de control serán en el tiempo.

Las especificaciones serán:

- Sobreoscilación 5%
- Tp =0.06 seg
- Tm = 0.002 seg

Especificaciones menos agresivas que en el controlador calculado en la memoria. Mediante una de las metodologías estudiadas en la asignatura de sistemas automáticos y discretizando la ecuación 1 se obtiene el controlador discretizado. (Los cálculos del controlador sin embargo no están optimizados y es por ello que las especificaciones previas difieren del resultado, ya que la sobreoscilación es del 10% y el Tp= 0.064 seg). La ecuación 1 muestra el controlador discretizado con la aproximación de tustin.

$$C(z) = \frac{0.0491 \cdot z^2 + 0.0031 \cdot z - 0.0459}{z^2 - 1.908 \cdot z + 0.908} \quad (1)$$

Lo que se prueba con este controlador es el método de respuesta transitoria rápida combinada con control linear-no-linear [25]. Este método combina control lineal con no lineal. En este control la salida se comprueba con dos niveles de tensión, (para este trabajo de Rpm), que conformarán los umbrales superior e inferior. De esta forma si la respuesta del sistema se encuentra fuera de los umbrales superior e inferior, entra en modo no lineal y el duty de la señal PWM es máximo para el caso que la respuesta no alcance el umbral inferior, y mínimo si esta respuesta se encuentra por encima del umbral máximo. Una vez que la salida del sistema se encuentre dentro de este rango el controlador entra en modo lineal y la respuesta del sistema depende ya de la dinámica del controlador. En la figura C.1 se observa la respuesta a este sistema controlado a una entrada escalón de 2300 Rpm.

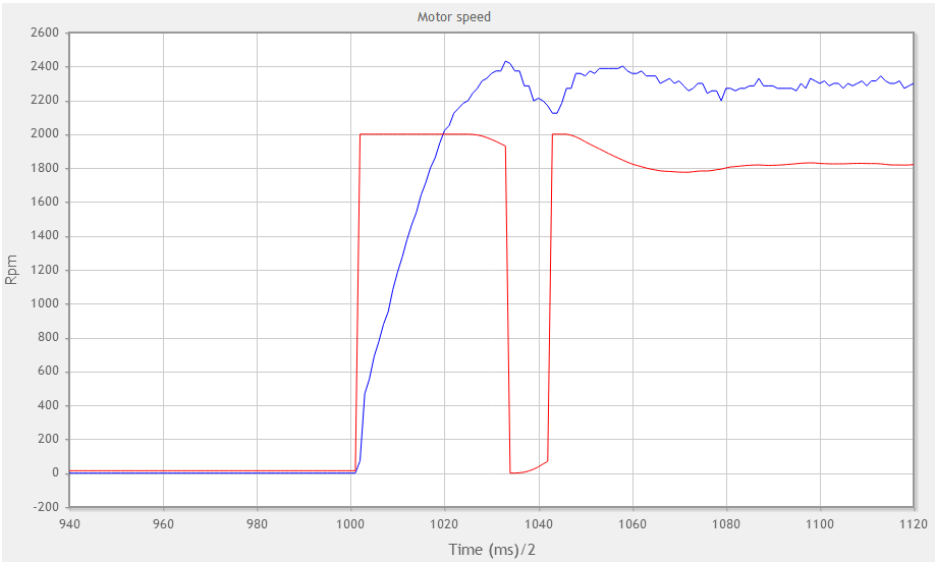

Figura C.1 Respuesta del sistema con control lineal-no lineal.

Para este controlador se establecen como umbrales aquella respuesta del sistema que no difiera más de un 5% de la referencia a alcanzar. Tal y como se observa en la figura D.1 la acción del controlador es máxima hasta que el sistema alcanza el 95% de la referencia, una vez llega hasta aquí se cambia al modo lineal y la acción se reduce respondiendo al modelo dinámico del controlador, sin embargo la dinámica del controlador no es suficientemente rápida y la respuesta supera el umbral superior llevando de nuevo el controlador a no lineal. Esto es así hasta que el controlador es capaz de estabilizar la respuesta dentro del rango establecido. Sin embargo para referencias menores como se ve en la figura C.2, que simula la respuesta del sistema para una referencia de 700 Rpm y con un rango del 10% donde el control es lineal, el sistema no es capaz de estabilizarse.

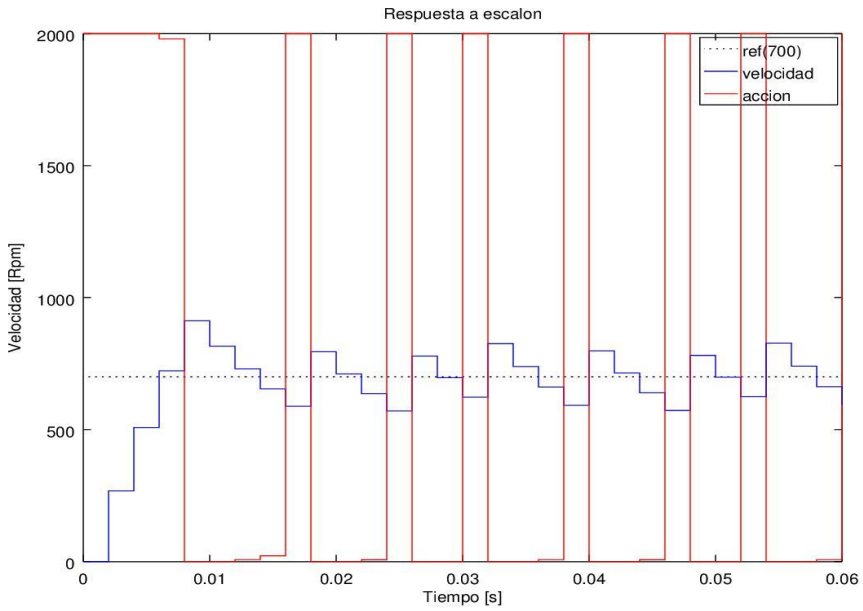

Figura C.2 Simulación del control lineal-no lineal ante escalón de 700 Rpm.

La verificación experimental de este comportamiento se puede observar en la figura C.3

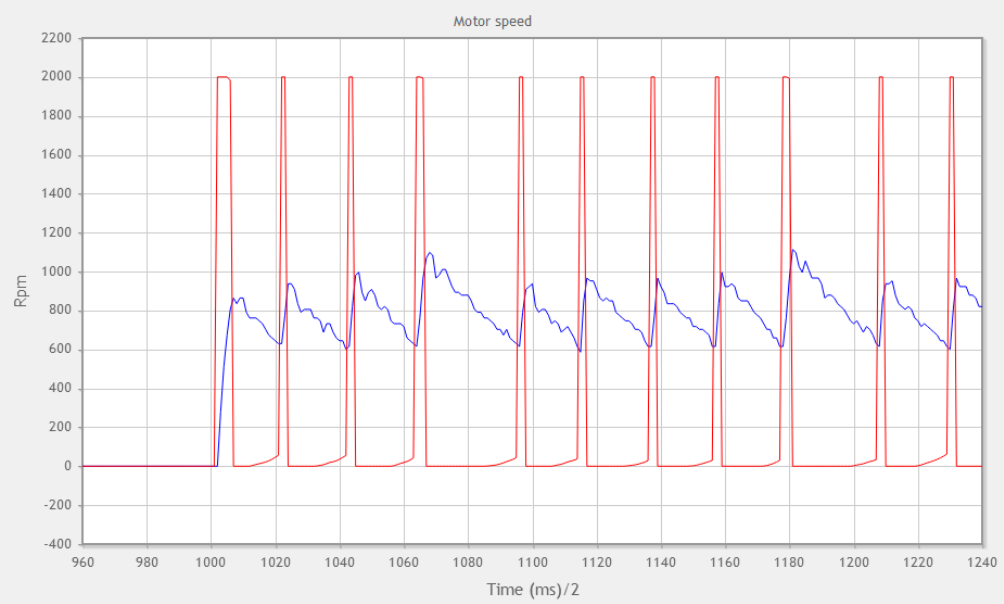

Figura C.3 Respuesta del sistema a entada escalón de 700 Rpm.

Esto es así porque la dinámica del controlador no es lo suficientemente para revertir con rapidez los valores de duty del PWM máximo y mínimo y de ajustar su valor al valor que le corresponde dentro del rango lineal. De esta forma este tipo de controlador solo funcionará para valores muy próximos a los máximos y mínimos de respuesta del sistema, o bien aumentando el rango donde el control es lineal y se da tiempo suficiente al controlador a ajustar el duty y a estabilizar la respuesta del sistema.

D.Programa en lenguaje C

```
#include "msp.h"
void Iniclock(void);
void IniTimer(void);
void Inipins (void);
volatile float x=0;
volatile int o,ref,s;
volatile float i[5000]; //vector de la velocidad
volatile float l[5000]; //vector de la acción
volatile unsigned char t,b,p,m;
volatile float y,yk_1,uk,uk_1,uk_2,ek,ek_1,ek_2,ref1,ref2,ref3,ref4;
void main(void)

{


    WDTCTL = WDTPW | WDTHOLD;           // Stop watchdog timer
    Iniclock();
    IniTimer();
    Inipins();


    s=1000;
    y=0;
    o=0; // variable que indica nº para los vectores de velocidad y acción
    t=1;
    m=0;
    ref=0;
    uk,uk_1,yk_1,ek,ek_1,ref1,ref2,ref3,ref4=0;

    SCB_SCR |= SCB_SCR_SLEEPONEXIT;// La CPU está en modo de bajo consumo
hasta que se produce una interrupción.

    NVIC_ISER0 = 1 << (9);//Habilitación de las interrupciones (TA0_N)
    NVIC_ISER1 = 1 << (6);//Habilitación interrupciones Puerto 4
    NVIC_IPR2 = 0x4000; //Prioridad 2 para la interrupción 9
    __enable_interrupt(); //Habilitación de las interrupciones

    while(1)
    {
    }

}
void port4_ISR ( void )
{
      x= x +1;
      P4IFG &= ~BIT4;
}
void velocity (void){ //función de velocidad y controlador

      switch (TA0IV)
      {
            case 0x0E:
            {
                  P1OUT ^= BIT7; //Pin que conmuta al principio y al final de
la función para saber tiempo de cálculo
                  y=(float)(x/2)*1875/128;
```

```
                    x=0;
                    if (((b)||(t==1))&&(o<5001)){ //La variable b se activa
desde la GUI para dar comienzo a la toma de datos
                          o=o+1;
                          i[o]=y;
                          l[o]=uk;
                          t=1;}
                    else{
                          o=0;
                          t=0;
                          m=0;}
                    if ((p)&&(o>(s-1))){ //la variable p activa el controlador
una vez que el valor del dato o supera el valor de s proporcionado por la GUI
menos 1.
                          // de esta forma se puede concretar el tiempo a
partir del cual se habilita el controlador
                          m=m+1;
                          if (m>2){
                                      ref4=ref3;
                                      ref3=ref2;
                                      ref2=ref1;
                                      m=0;}
                          else{
                                      ref4=ref4;
                                      ref3=ref3;
                                      ref2=ref2;
                                      ref1=ref1;
                                      m=m;}
                          ref=0.25*(ref1+ref2+ref3+ref4);
                          ek= ref-y;
                          uk = uk_1 +1.236*ek -1.01*ek_1;

                          if (uk>2000)
                                uk= 2000;
                          else if (uk<0)
                                uk=0;
                          else
                                uk= uk;}
                          TA2CCR1 = uk;
                          ek_1=ek;
                          uk_1=uk;
                          P1OUT &= ~BIT7;
                          break;
             }

      }
}

void Iniclock (void){ //configuración del reloj
      CSKEY = 0x695A; // desactivar registros CS
      CSCTL0 = 0; // reset de la configuración del DCO
      CSCTL0 = DCORSEL_4; // selección de DCO 4 (24MHz)
      CSCTL1 |= SELS_3| SELM_3;
      CSKEY = 0; // lock CS registers

}
void IniTimer (void){ //configuración de los timer
      TA0CTL |= TASSEL_2+ MC_1 +TAIE;
      TA0CCR0 = 47999;
```

```
        TA2CTL |= TASSEL_2+ MC_1;
        TA2CCTL1 = OUTMOD_7;
        TA2CCR0 = 2000;
        TA2CCR1 = 0;


}
void Inipins (void){ //configuración de los puertos
        P1OUT &= + ~BIT7;
        P1DIR=0;
        P1DIR |= BIT7;
        P4IFG =0;// P4 clear flag
        P4DIR &= ~BIT4;// P4.4 in
        P4IES &= ~BIT4;// P4IFG se activa en la transición de bajo a alto.
        P4IE |= BIT4;// Habilitación de la interrupción P4.4
        P5OUT =0;
        P5IFG =0;// P5 clear flag
        P5DIR |= BIT6;
        P5SEL0 |=BIT6;
        P3OUT =0; // Se inicializan los puertos sin utilizar para eliminar
perdidas de corriente.
        P3DIR |=1;
        P6OUT =0;
        P6DIR |=1;
        P7OUT =0;
        P7DIR |=1;
        P9OUT =0;
        P9DIR |=1;
        P8OUT =0;
        P8DIR |=1;
        P10OUT =0;
        P10DIR |=1;
}
```

D.1 Diagrama de flujo

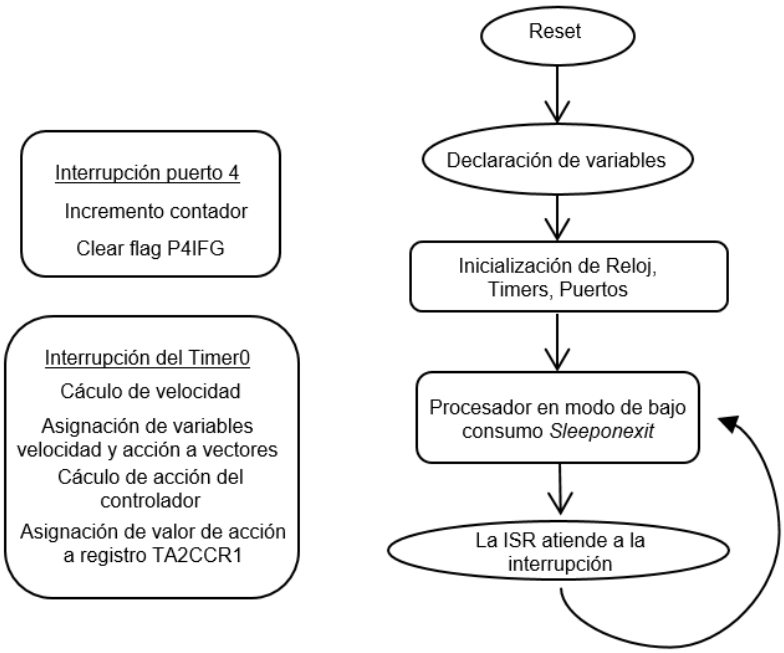

Figura D.1 Flow chart