CAPÍTULO 4.FASE II:DPL2. INCLINÓMETRO

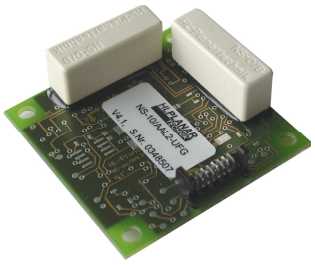
4.1 INTRODUCCIÓN.

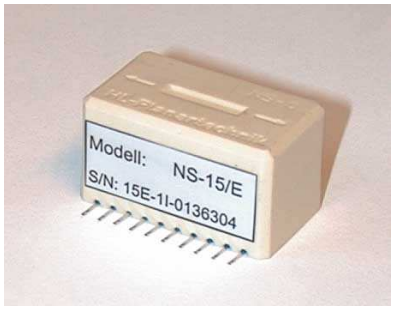
Después de haber concluido el nuevo sistema de medida de rotación con el nuevo micro, pasamos a la segunda parte, el inclinómetro. En este caso, como dijimos, el DPL2 integra ya el microprocesador MSP430F133. La idea de esta parte del proyecto es aunar en uno sólo los dos sistemas de medida independientes. Para ello hay que modificar el programa ya existente y adaptarlo al nuevo sistema. También hay que proveer al sistema final de capacidad de comunicación por SPI. En este caso, también se producen mejoras en el propio sistema inclinómetro, como nuevos algoritmos matemáticos o nuevas formas de almacenar datos en la memoria flash.

Lo que se trata, entonces, es de modificar y adaptar el programa existente del DPL2, que manejaba los datos obtenidos por los sensores NS-15 para mejorarlo a su propio sistema y conseguir adaptarlo al KMS32.

Las funciones a realizar por el micro son:

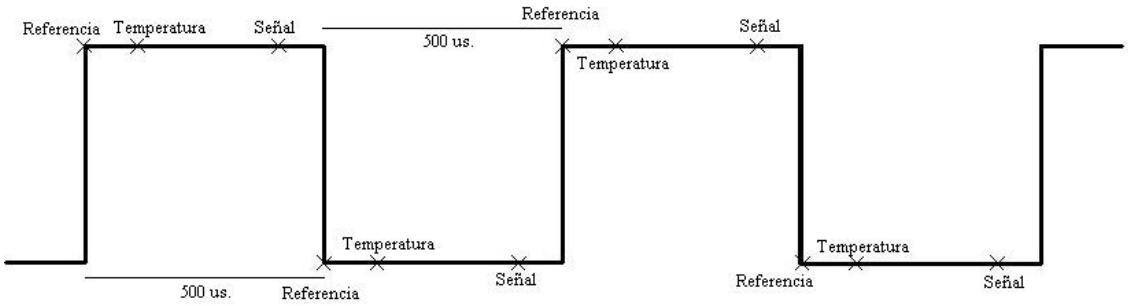
- controlar el Multiplexador que alterna las señales analógicas procedentes del sensor KMT32.
- convertir de analógico a digital las señales de entrada provenientes de los sensores
- calcular con esos resultados el ángulo final de rotación e inclinación.
- responder a las interrupciones producidas por el interfaz SPI que nos piden datos de los sensores.



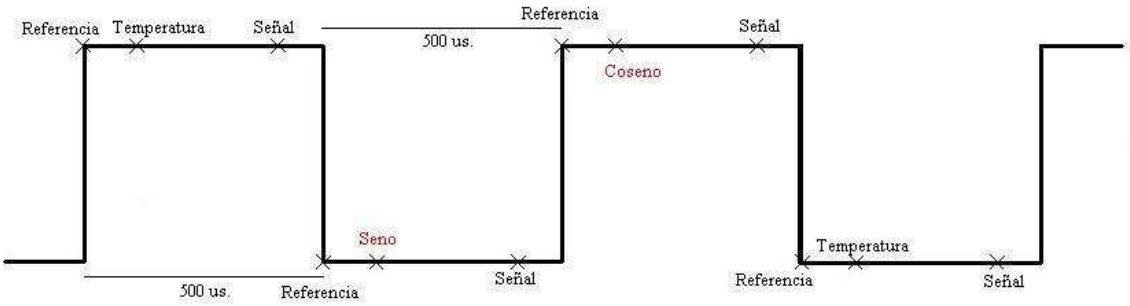


4.2 DESCRIPCIÓN DEL FUNCIONAMIENTO

El MSP430 toma las señales provenientes de los sensores NS-15 (uno para cada eje X e Y) y comparándolos con una tabla almacenada en memoria flash que se realiza en el proceso de calibración, da como resultado el ángulo de inclinación positivo o negativo de cada uno de los dos ejes. Para ello necesita hacer tres medidas, la temperatura (existen pequeñas desviaciones en la curva según la temperatura), la señal de referencia, y la señal que mide la inclinación. Además, ahora, añadimos una cuarta medida, la del sensor de ángulo de rotación. La forma de medir cada una de estas señales en el tiempo era la siguiente:



Como se puede observar, las medidas en este caso tienen una forma periódica, de 500 us. Cada 500 us, se medía la señal del eje X o del Y alternativamente, lo que da un periodo real del ciclo de 1 ms. Para este sistema, la velocidad no es crítica, y por tanto, no se persigue tomar medidas velozmente. Por esto se sigue la estructura temporal antes descrita. Al integrar la medida de ángulo de rotación, lo que hacemos es, en lugar de medir cada 500 us. la temperatura, alternamos esta medida con la del seno y el coseno. De esta manera cada 1,5 ms. hemos completado todas las medidas:



4.3 PROGRAMA IMPLEMENTADO

Pasemos a continuación a ver el programa implementado, para lo cual lo dividiremos en 3 bloques básicos, el programa principal, las interrupciones y las rutinas secundarias, que serán utilizadas por el programa principal de forma lógica y ordenada para conseguir que el funcionamiento del sistema sea el deseado.

- módulo principal (con las interrupciones para los Timers, el ADC y USART (SPI)) –**DPL2.c**
- módulo de inicio y control del ADC, además de controlar el Mux y el orden de las señales -**ADC.c**
- módulo para inicializar los temporizadores – **Misc.c**
- módulo para el manejo de la memoria flash (lectura, escritura y borrado) – **Flash.c**
- módulo para la inicialización y almacenamiento de datos para comunicación SPI – **SPI.c**
- módulo para sacar las señales que van al Multiplexador – **Mux_DPL2.c**
- módulo para calcular la compensación de temperatura, manejar la tabla de calibración, y ángulos finales de inclinación y giro -**Calc.c**
- módulo para el cálculo de la atan que nos dará el resultado final del ángulo de giro – **atan.c**

4.3.1 Módulo Principal

En el programa principal se hará tanto la correcta inicialización de todas las partes del micro que se utilizarán posteriormente, como de llamar a las rutinas necesarias para el cálculo de los ángulos y su transmisión serie. Se encarga de configurar de forma adecuada tanto la USART, como los temporizadores o el convertidor analógico/digital. Veamos cómo funciona el programa (haciendo uso de las diferentes subrutinas cuando sea necesario) separándolas en los mismos pasos que realizaría durante el funcionamiento normal del circuito:

- Inicialización de variables y pines de entrada/salida
- Apagado del Watchdog
- Configuración del cristal. El oscilador interno de 8 MHz. Se elige también como reloj máster
- Establecer orden de medidas (primero referencia)
- Inicializar el ADC
- Arrancar temporizadores
- Leer los Offset de la memoria flash (tanto de inclinación como de giro)
- Inicializar la comunicación SPI
- Habilitar interrupciones
- Comienzo del bucle infinito. Se repite continuamente lo siguiente:
    - Espera hasta que se active el flag para el cálculo, activado por el temporizador B.
    - La primera vez que entra, inicializa los valores de las tablas usadas posteriormente en el filtro del ADC.
    - Cálculo de la temperatura, y de los ángulos.
    - Si el flag para el offset de los ángulos de inclinación está activado, fija el offset. Lo que hace es dar un nuevo valor de 0º a partir de donde medir.
    - Si el flag para borrar el offset de los ángulos de inclinación está activado, borra el offset. Restaura el punto inicial de medida, el 0º.
    - Si el flag para escribir continuadamente en la memoria flash está activado, llama a la función que se encarga de eso.
    - Si las medidas del ángulo de inclinación del eje X han sido correctas, se guardan para su posible envío por SPI.

- Si las medidas del ángulo de inclinación del eje X están por debajo de los valores de la tabla de calibración, se trata de un desbordamiento por abajo (UNDERFLOW). En este caso el ángulo toma un valor fijado para señalarlo, el valor es 0800h.

- Si las medidas del ángulo de inclinación del eje X están por encima de los valores de la tabla de calibración, se trata de un desbordamiento por arriba (OVERFLOW). En este caso el ángulo toma un valor fijado para señalarlo, el valor es 7FFEh.

- Repetición de los tres pasos anteriores, pero para el eje Y.

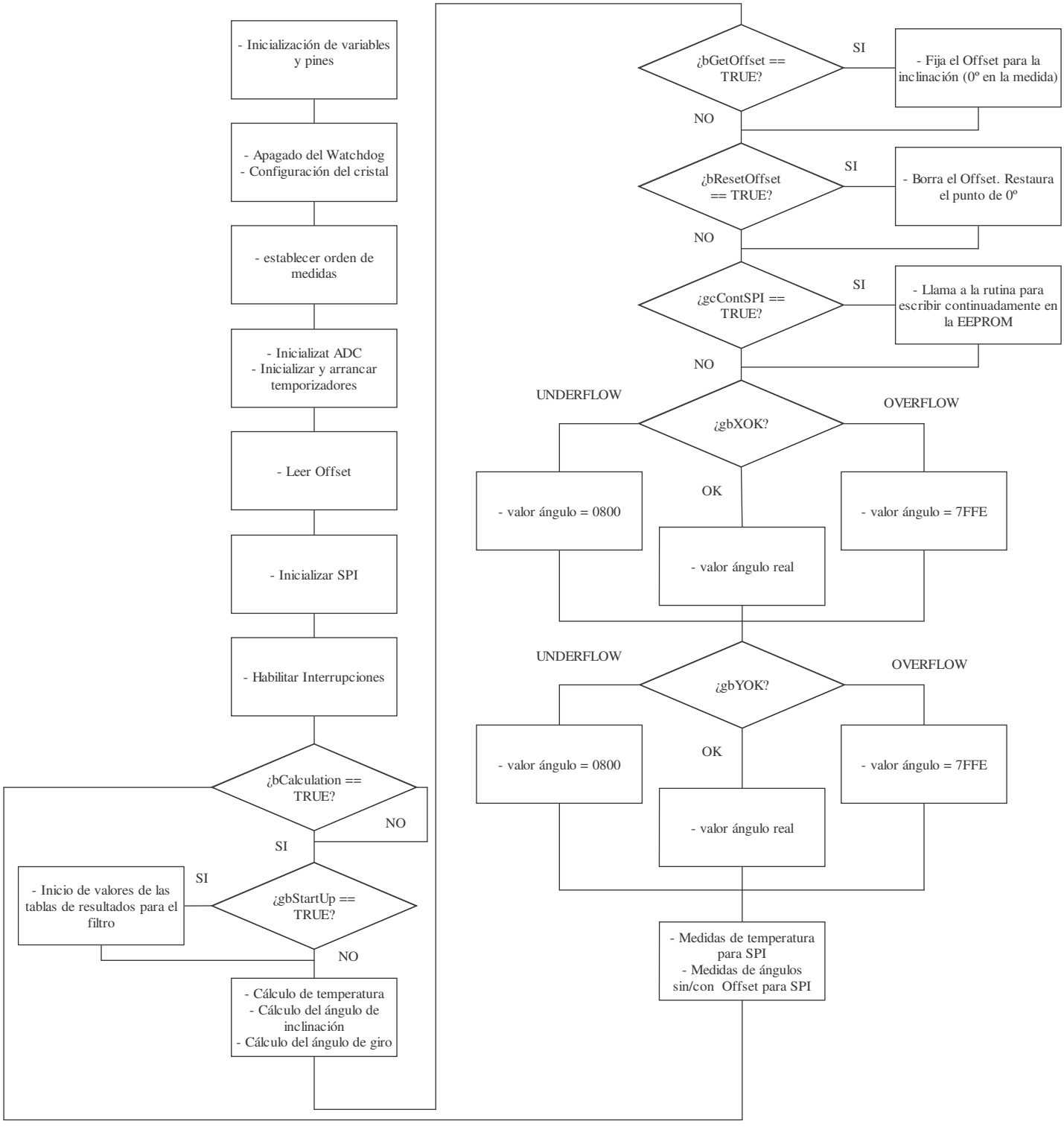
- Se guardan las medidas de la temperatura para su posible envío por SPI.

- Se guardan las medidas 'crudas' del sensor de inclinación, esto es, la señal convertida a digital pura, sin offset para su posible envío por SPI.

- Se guardan las medidas 'crudas' del seno y del coseno del sensor de giro, esto es, la señal convertida a digital pura, sin offset para su posible envío por SPI.

- Se guarda la medida del ángulo de giro para su posible envío por SPI.

A continuación se puede ver la estructura del programa principal en un diagrama de flujo para su mejor entendimiento. En asterisco señalo el bloque que se desarrollará posteriormente en diagrama de flujo también:



Módulo Principal

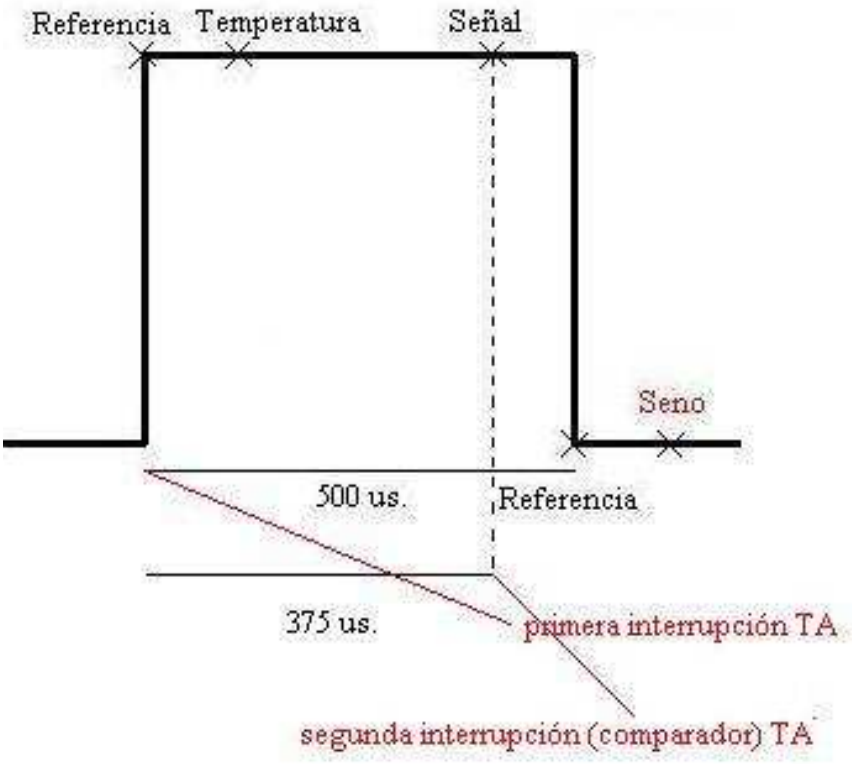
4.3.2 Rutinas

En este apartado se explican las rutinas llamadas por el módulo principal y cada una de las subrutinas llamadas a su vez por éstas. Se irán explicando según son llamadas por el programa principal y se procurará seguir una estructura jerarquizada. En algunos casos más complicados se incluye diagrama de flujo para facilitar su comprensión.

Ansteuerung:

- en esta función se inicializa el temporizador A (TA). Este temporizador tiene dos funciones, crear la "señal" rectángulo de 500 us. mostrada con anterioridad para crear la estructura temporal de medidas, y con el comparador hacer que cada 375 us. después de empezar el pulso, empiece a convertir la señal del inclinómetro. De esta forma cada 500 us. comienza la conversión de la referencia, y 375 us. después la de la señal.

- cuando se produce la interrupción del temporizador A salta a ala rutina de interrupción que se explicará luego, pero que básicamente lo único que hace es arrancar el convertidor.



Adc_Init:

- inicialización del registro ADC12CTL0 y ADC12CTL1 (tipo de conversión, duración del tiempo de sampling, tensión de referencia, etc.).

- se usan siempre (a diferencia del sistema anterior) 8 registros de memoria del ADC.

- habilitar la conversión. Se convierte el mismo canal 8 veces.

PWM_Init:

- conserva el nombre de una versión anterior donde se usaba modulación PWM.

- se trata de la inicialización del temporizador B (TB).

- al saltar el temporizador salta la rutina del temporizador B que. aunque se explicará posteriormente, provoca el inicio de los cálculos (de temperatura y ángulos).

ReadOffset:

- con esta función se lee, en caso que se haya introducido, el valor del offset de la inclinación, para cambiar el punto de referencia. El 0º.

- lo que hace es leer de dos direcciones concretas de la memoria flash el valor allí guardado, que no es otra cosa que el valor medido cuando se fijó el offset, de esta manera se resta ese valor al calcular el ángulo de inclinación cambiando así el punto de referencia.

- la forma de introducir el offset se explica en la parte de comandos que se pueden introducir a través de la comunicación SPI.

- esta función hace uso, a su vez, de la función **Flash_Read**.

  Flash_Read:

  - a esta función se le pasa como parámetro una dirección de memoria de la memoria flash.

- con esta dirección, distingue entre tres segmentos de la memoria flash, según sea offset, cheksum (no usada en este programa) o ninguna le las dos anteriores. Así según sea la dirección a leer se sabe a qué segmento pertenece y qué tipo de dato es.

- la forma de leer es cambiar el valor de un puntero de forma que señale a la dirección deseada. Después se lee el contenido de ese puntero y se devuelve como dato.

ReadOffset_SC:

- a la hora de medir el ángulo de giro, es necesario un offset que se introduce al calibrar.

- lo que hace esta función es leer de dos posiciones concretas de la memoria memoria flash el valor del offset del seno y del coseno, para así tener un valor de ángulo de giro válido.

- esta función también hace uso de **Flash_Read**, ya descrita anteriormente, para leer el valor de los offsets.

init_SPI:

- aquí se habilita la USART en modo SPI. Datos de 8 bits. Modo esclavo. Al actuar en modo esclavo, el sistema sólo puede, y de hecho debe, responder cuando el Master le pregunta. De esta forma el sistema no puede comunicarse por sí sólo, sino que necesita una petición del Master. Al producirse esta petición del Master, entra en interrupción del SPI, donde se manejan los datos recibidos y se envían los datos que procedan.

- se fija también el 'baudrate' a mitad de velocidad del reloj (8/2 = 4MHz.)

- se habilita la interrupción para el SPI.

Calculation_Angle:

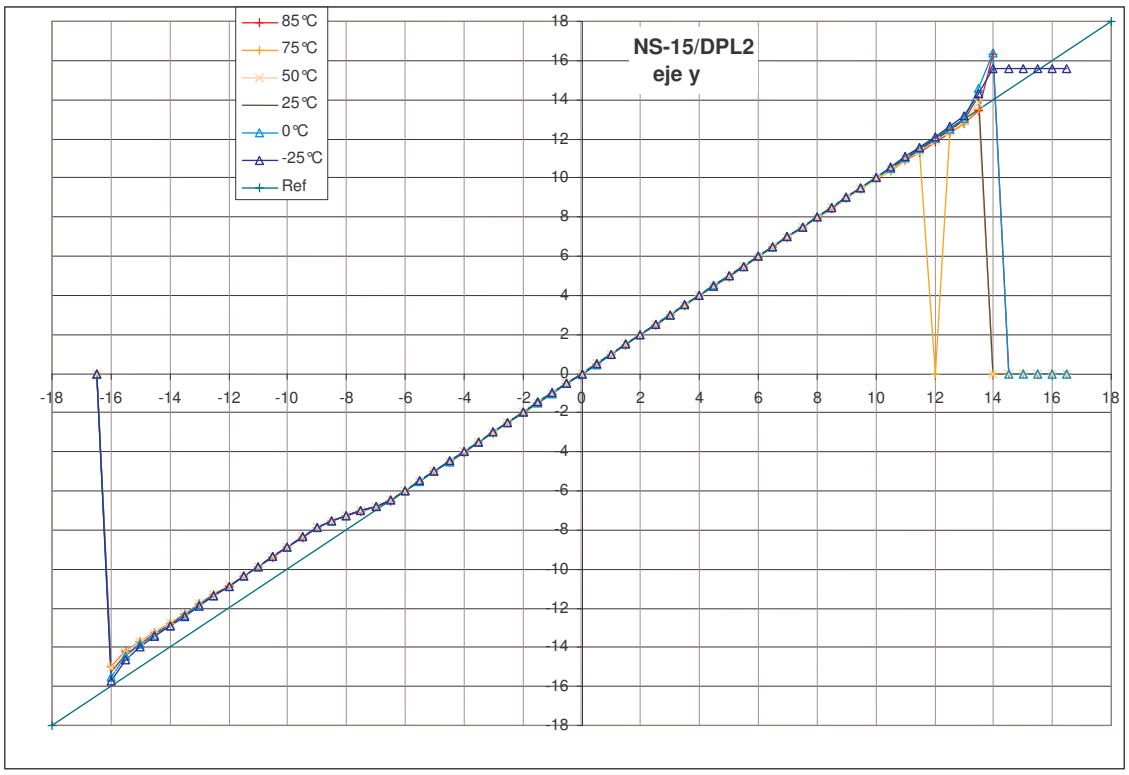
- llama a la ya conocida función **atangreen** para calcular el valor del ángulo de giro. Para ello usa los valores del seno y del coseno convertidos por el ADC.

- la función **atangreen** se describió en el capítulo 3. Se trata de una función alternativa más rápida, pero menos exacta de la función **atan** de la librería math.h

Calculation:

- en esta función se calcula el ángulo de inclinación de los ejes X e Y.

- la forma de tomar el valor final es interpolando el valor medido por el sensor con una tabla de calibración que lleva almacenado previamente en la memoria flash. Esa tabla de calibración crea una curva característica valor/ángulo. La temperatura crea pequeñas variaciones en esa curva que deben ser compensadas.



- para ello hace uso de dos funciones, **TemperatureCompensation** y **LUTSearch**.

TemperatureCompensation:

- en esta función se hace la compensación térmica del valor obtenido por el sensor para cada uno de los ejes, y obtiene el valor que se debe interpolar a continuación usando **LUTSearch**.

- primero obtiene al factor de compensación a través de la temperatura medida por el sensor integrado en el micro y un coeficiente grabado en memoria. Este valor lo guarda en la variable "f_Tcomp_Faktor".

- a continuación obtiene los valores de la señal de referencia y de los sensores y los almacena en otras tres variables: "f_delta_Ref", "f_delta_X" y "f_delta_Y".

- lo siguiente es comparar el valor de las señales de X e Y con la de referencia y corregirlo con el factor de compensación, todos ellos obtenidos en el paso anterior.

- los valores obtenidos son los valores que se compararán con la tabla de calibración o "Lookup Table" (LUT). Se almacenan en las variables: "gsiRawCompX" y "gsiRawCompY".

LUTSearch:

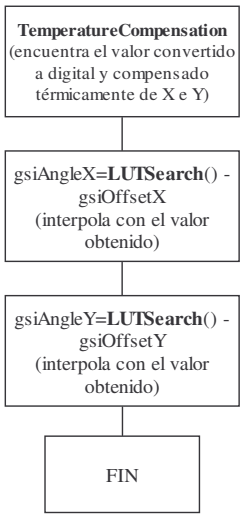
- esta función toma el valor obtenido anteriormente y lo busca en la tabla de calibración donde interpola para buscar el ángulo de inclinación final. La forma de interpolar que había previamente era usando una sencilla regla de tres. Era una interpolación lineal. Uno de los nuevos objetivos era encontrar una función que interpolara mejor. Aunque luego se explicará en detalle, la nueva función empleada fue una Spline cúbica natural. De esta manera aumentamos la precisión a la hora de interpolar. Lo que hacemos una vez tenemos el dato a interpolar es tomar cuatro puntos de la tabla. Los dos anteriores y los dos posteriores, y con esos puntos se calcula la Spline.

- lo primero es determinar si se trata de un valor del eje X o del eje Y ya que ambos tienen tablas distintas. Una vez sabemos de qué se trata nos situamos en la tabla correspondiente.

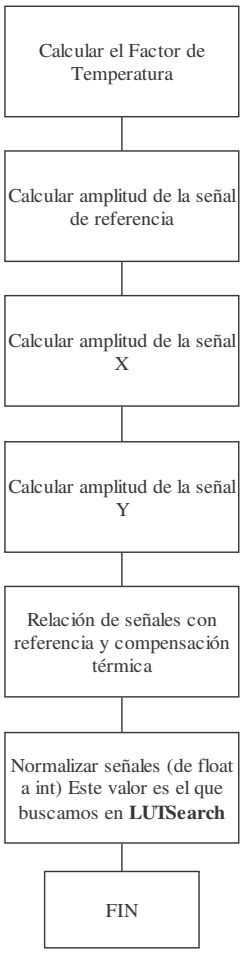
- si el valor es menor que el mínimo de la tabla, o mayor que el máximo, entonces estamos en un caso de desbordamiento por abajo o por arriba (UNDERFLOW o OVERFLOW). En alguno de esos casos, se indica y se sale de la función.

- si no es el caso anterior se recorre la tabla comparando el valor que tenemos con los datos de ésta. En el momento que superamos algún valor de la tabla nos paramos en esa posición. Existen dos excepciones al resto de los casos, el caso que nos encontremos en el primer intervalo o el caso que nos encontremos en el último intervalo. Normalmente se toman los dos valores anteriores y los dos valores posteriores para interpolar, pero en estos dos casos no tenemos dos valores anteriores en el primero ni dos posteriores en el segundo. Lo que se hace es seguir cogiendo cuatro puntos, pero si nos encontramos en el primer intervalo tomamos únicamente el primer valor de la tabla y los tres siguientes. Si estamos en el último intervalo los que se toman son el último y los tres anteriores.

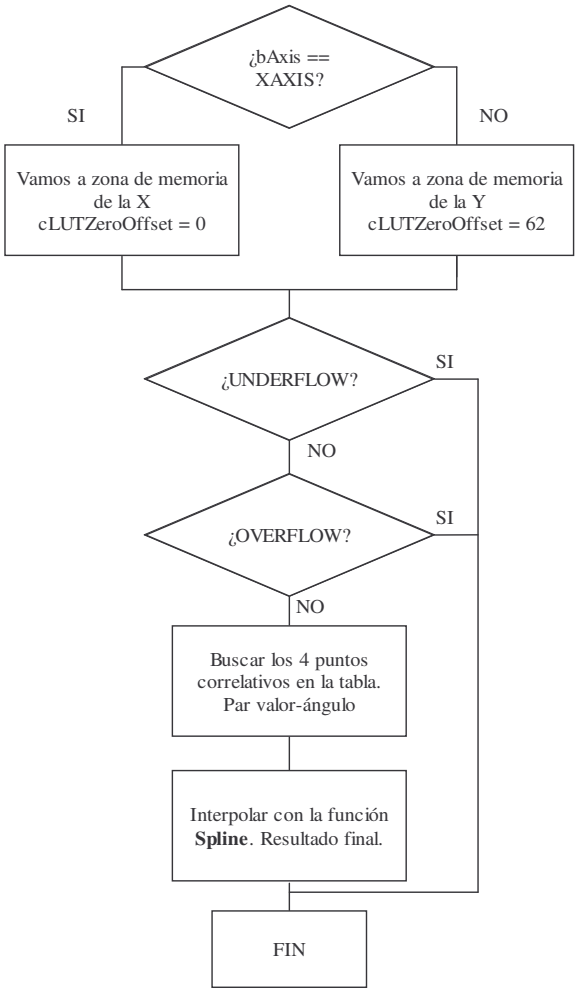
- para calcular el valor final se usa la función **Spline**, que se detallará luego.



Calculation



TemperatureCompensation



LTUSearch

Flash_Erase:

- se emplea para borrar la memoria memoria flash.
- la función esta diseñada para poder elegir el segmento de memoria que se desea borrar.
- la forma de hacerlo es tal y como viene en el manual del micro. Se trabaja con bits de seguridad y para borrar se hace una “dummy write” o escritura sin sentido (en este caso 255)

Flash_Write:

- a esta función se le pasa como parámetro una dirección de la memoria flash.

- con esta dirección, distingue entre tres segmentos de la memoria flash, según sea offset, cheksum (no usada en este programa) o ninguna le las dos anteriores. Así según sea la dirección donde escribir se sabe a qué segmento pertenece y qué tipo de dato es.

- la forma de escribir es cambiar el valor de un puntero de forma que señale a la dirección deseada. Después se escribe en el contenido de ese puntero.

Flash_Write_Continuous:

- esta función surge con la necesidad de introducir todos los datos en la memoria flash (tablas, constantes, etc.) La forma que teníamos para escribir es byte a byte, un proceso que aunque se automatice, no deja de ser lento. Aún siendo una función práctica, y que se utilizó infinidad de veces en la realización del proyecto, es un tanto polémica por la forma que tiene de finalizar.

- una vez que se introduce el comando adecuado a través de la consola, el programa entra en esta rutina, de la cual sólo saldrá a través de interrupción del Watchdog.

- una vez dentro de la rutina, habiendo escogido una dirección de inicio de escritura, cada carácter introducido se irá almacenando consecutivamente en la memoria. Al poderse guardar todo tipo de información en la memoria flash, no existe un comando válido para indicar que se quiere salir de la rutina en cuestión, que se ha terminado de escribir. No hay forma de diferenciar entre comando o información a guardar. Por ello, la solución implementada es a través del Watchdog. En el momento que se dejan de introducir datos para almacenar por más tiempo del establecido en el Watchdog, se produce una interrupción y se resetea el programa, quedando toda la información previa almacenada en memoria flash sin problemas.

- esta forma poco ortodoxa de salir de la función la convierte en algo inestable y difícil de hacer llegar al cliente a través de los manuales. Sin embargo, a la hora de introducir las tablas de

calibración manualmente para probar el sistema, ha sido de muchísima utilidad.

SPI_Temp, SPI_RawXY, SPI_RawSC, SPI_Angle:

- estas cuatro funciones son similares y se encargan de almacenar en distintos arrays del tipo char los valores de la temperatura, de la señal del sensor de cada uno de los ejes, de las señales seno y coseno sin offset y del ángulo final de giro respectivamente. Los valores se almacenan byte a byte con la finalidad de enviarlos de esa manera por el interfaz SPI posteriormente.

ADC_Read:

- se trata de la función más larga de nuestro programa. Se encarga de manejar los datos convertidos por el ADC (filtro), almacenarlos, preparar el convertidor para la siguiente conversión, etc. Establece también, el orden de las señales que tocan ser medidas en cada momento.

- primeramente averigua si lo medido es una señal de referencia, de temperatura/seno/coseno o del eje X o Y.

- a continuación realiza un filtro para aumentar la precisión de los datos medidos. Usa para ello, los datos obtenidos de la medida anterior y los nuevos datos. Básicamente el **filtro** funciona de la siguiente manera:

    - suma los 8 registros de memoria empleados en el ADC (todos deberían tener un valor aproximadamente igual)

    - multiplica ese valor por 4, con lo que tenemos aproximadamente 32 veces el mismo valor (aproximadamente porque los 8 registros nunca son exactamente iguales). Este valor será el nuevo valor medido

    - a continuación multiplica por 31 el último valor obtenido en la medida anterior del mismo valor

    - suma los dos valores obtenidos y además suma 16 para mejorar el redondeo

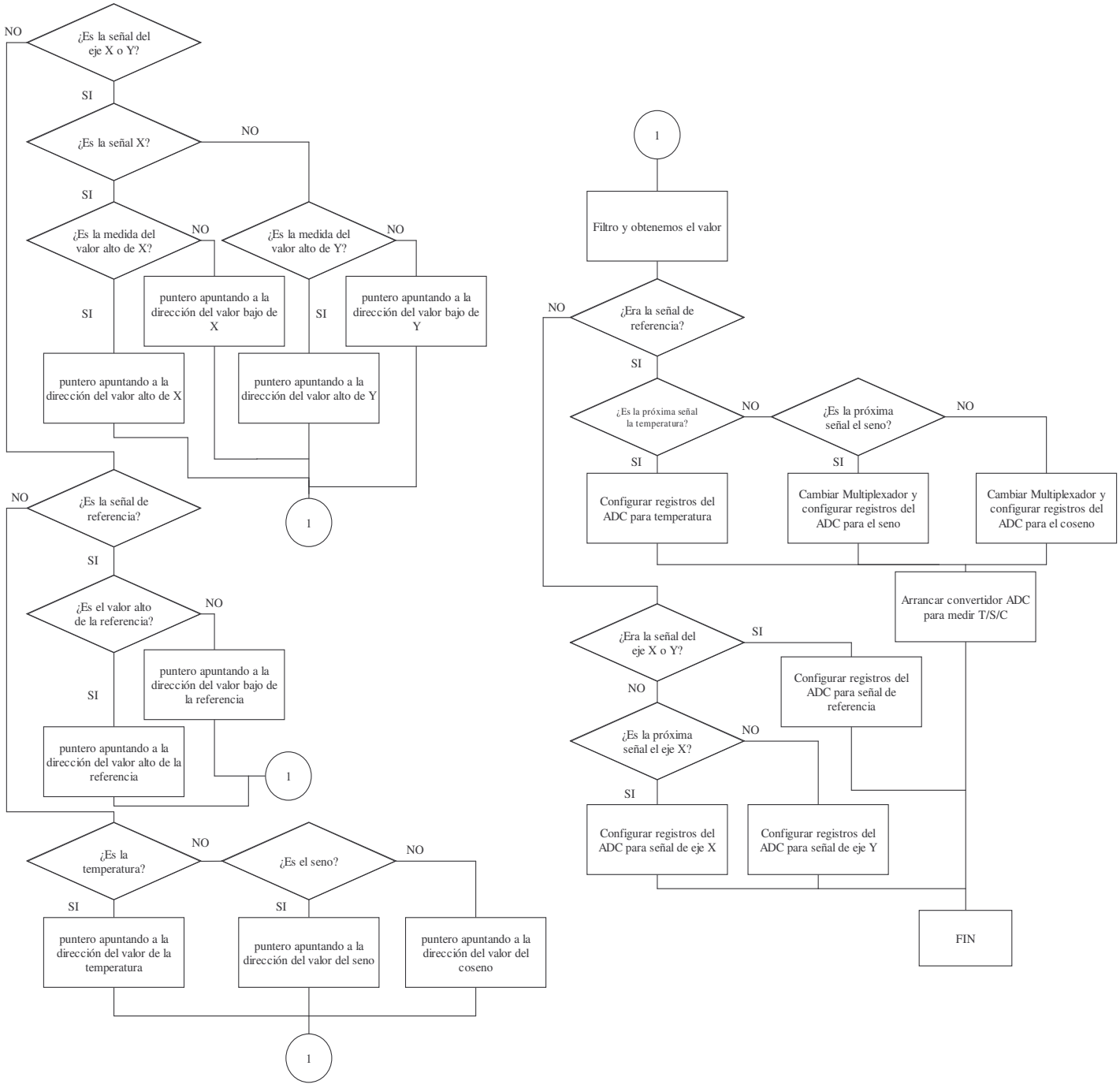
    - divide la suma total entre 32

- el resultado final será el valor empleado como nuevo valor. A la vez se almacena como valor antiguo para usarse en la próxima medida.

$$Nuevo = \frac{31 * Viejo + 4 * (\sum_{1}^{8} registrosADC) + 16}{32}$$

- una vez que tenemos el valor nuevo preparamos los registros del convertidor para la siguiente medida. En el caso de que la próxima medida sea el seno o el coseno, también se controla el Multiplexador por medio de la función **Mux_DPL2**, similar a la función Mux del KMS32 pero con pines diferentes en este caso.

- por último, establecemos la próxima señal a medir. En el siguiente diagrama de flujo se puede ver el funcionamiento de la función entera:


¿Es la señal del eje X o Y?
NO
SI
¿Es la señal X?
NO
SI
¿Es la medida del valor alto de X?
NO
¿Es la medida del valor alto de Y?
NO
SI
SI
puntero apuntando a la dirección del valor bajo de X
puntero apuntando a la dirección del valor bajo de Y
puntero apuntando a la dirección del valor alto de X
puntero apuntando a la dirección del valor alto de Y
1
¿Es la señal de referencia?
NO
SI
¿Es el valor alto de la referencia?
NO
SI
puntero apuntando a la dirección del valor bajo de la referencia
puntero apuntando a la dirección del valor alto de la referencia
1
¿Es la temperatura?
NO
¿Es el seno?
NO
SI
SI
puntero apuntando a la dirección del valor de la temperatura
puntero apuntando a la dirección del valor del seno
puntero apuntando a la dirección del valor del coseno
1
1
Filtro y obtenemos el valor
¿Era la señal de referencia?
NO
SI
¿Es la próxima señal la temperatura?
NO
¿Es la próxima señal el seno?
NO
SI
SI
Configurar registros del ADC para temperatura
Cambiar Multiplexador y configurar registros del ADC para el seno
Cambiar Multiplexador y configurar registros del ADC para el coseno
Arrancar convertidor ADC para medir T/S/C
¿Era la señal del eje X o Y?
SI
NO
Configurar registros del ADC para señal de referencia
¿Es la próxima señal el eje X?
NO
SI
Configurar registros del ADC para señal de eje X
Configurar registros del ADC para señal de eje Y
FIN

ADC_Read

4.3.3 Interrupciones

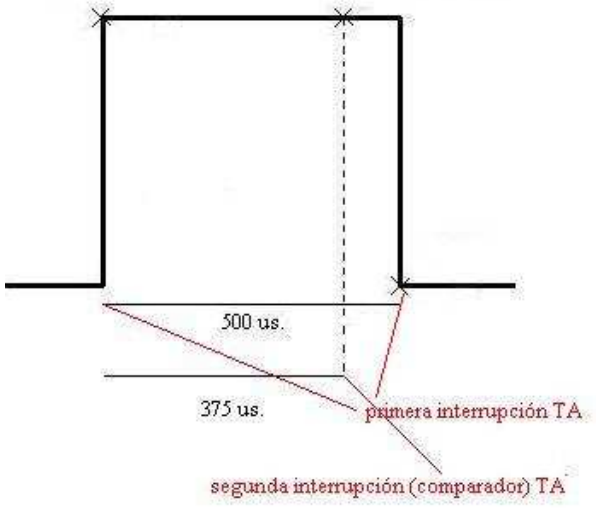
En este caso disponemos de cuatro interrupciones distintas cuyas rutinas han sido programadas. Tenemos las rutinas de interrupción de los temporizadores A y B, la rutina de interrupción del convertidor analógico/digital y la rutina de interrupción de la USART en su modo SPI.

A través de estas interrupciones controlamos los tiempos de ejecución del programa (con los temporizadores) y manejamos los datos de los sensores (convertidor) a la vez que se habilita de comunicación externa al sistema (USART).

Pasamos a explicar cada una de ellas:

Interrupción Temporizador A (timer0)

- en esta interrupción se entra por dos motivos, o bien porque han pasado 500 us periódicamente, o bien porque el comparador ha llegado a los 375 us.



- en cualquiera de los dos casos el resultado es el mismo, se arranca el convertidor analógico/digital. De esta manera, sabemos seguro que la conversión de la señal referencia y de la señal de inclinación de los ejes se harán en esos tiempos. La medida de la temperatura, seno o coseno se arranca inmediatamente después de acabar de tratar la señal de referencia. Para iniciar esa conversión no se usa ningún temporizador ya que no se disponen de más registros capturadores,

Interrupción Temporizador B (timerb2)

- al saltar esta interrupción se habilita el flag para empezar a medir las distintas señales. El tiempo establecido es de 10 ms. de modo que se hacen cien medidas por segundo.

Interrupción ADC (ADC12ISR)

- al producirse la interrupción por el convertidor analógico digital, lo que hacemos es llamar a la función **ADC_Read**. Esta función se explicó ya en el apartado anterior.

Interrupción SPI (SPInterruptRx)

- esta interrupción se produce al llenarse el buffer de recepción del SPI. Recordemos que estamos funcionando en modo esclavo, de modo que, cuando llega un dato al buffer de recepción inmediatamente mandamos lo que tenemos en nuestro buffer de transmisión.

- se emplea para poder comunicarme con el micro y pedirle los datos que ha obtenido gracias a los sensores. Existe todo un juego de comandos para obtener distintos datos y actuar sobre el micro y la memoria flash. En capítulos posteriores se detallarán estos comandos y para qué se usan.

- la forma de actuar es con un juego de funciones if y switch y usando varios flags de manera que según vayan llegando caracteres responde de una manera o de otra.

- al estar correctamente inicializado, para enviar los caracteres pedidos únicamente es necesario meterlos en el buffer de salida.

4.4 MEJORAS EN ALGORITMO MATEMÁTICO

La forma que hay para calcular el ángulo de inclinación medido es usando una tabla de valores (Lookup Table) e interpolando el valor medido con valores conocidos de ángulos. Hasta ahora la forma de hacerse esto era tomando el valor obtenido y buscando el valor tabulado anterior y posterior se interpolaba linealmente (por comparación de triángulos). Si vemos la curva formada por los valores tabulados (medida/ángulo) vemos que no se trata de una función lineal, por lo que al hacer la aproximación se está perdiendo bastante precisión. Sobre todo pierde la linealidad en los extremos.

Ángulo / ºC	Valor medido X	Valor medido Y
-16,5	-20360	-20542
-15,4	-19138	-19386
-14,3	-17005	-17474
-13,2	-14137	-14465
-12,1	-11833	-12061
-11,0	-10014	-10232
-9,9	-8764	-8928
-8,8	-7634	-7763
-7,7	-6582	-6681
-6,6	-5588	-5665
-5,5	-4661	-4704
-4,4	-3745	-3764
-3,3	-2878	-2883
-2,2	-2019	-2001
-1,1	-1185	-1149
0,0	-351	-300
1,1	475	536
2,2	1318	1398
3,3	2170	2255
4,4	3026	3123
5,5	3907	4036
6,6	4812	4972
7,7	5762	5948
8,8	6760	6978
9,9	7822	8094
11,0	8956	9268
12,1	10542	10669
13,2	12596	12446
14,3	15065	14843
15,4	17924	17836
16,5	19405	19383

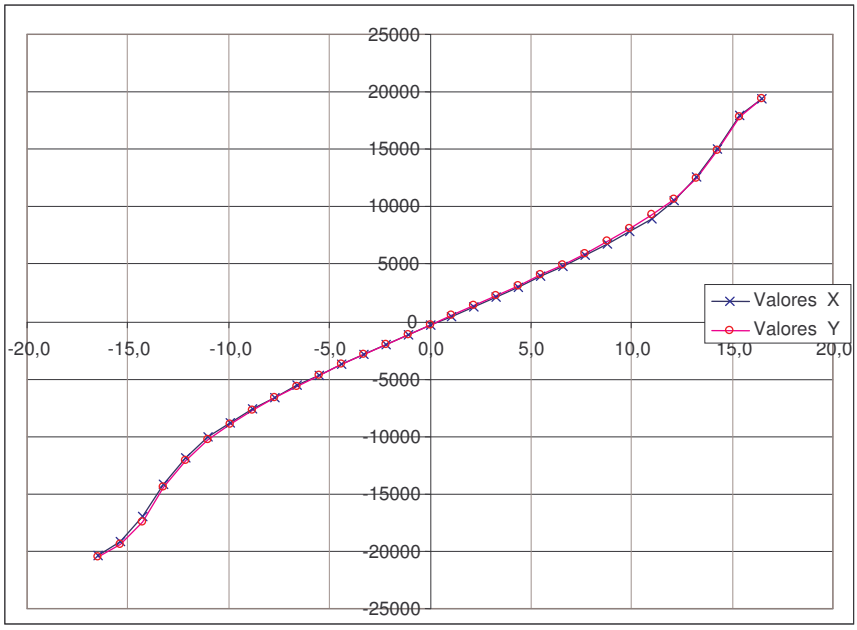
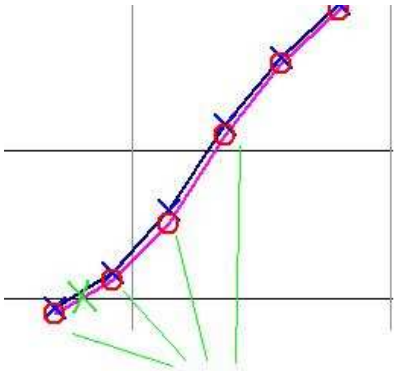


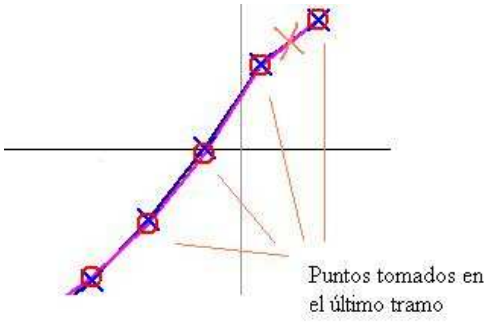
Tabla para el eje X e Y

Con objeto de mejorar esa interpolación se buscaron otras alternativas que permitiesen unas medidas más ajustadas a la función real. Se buscó una función de interpolación de mayor grado, y al final la opción tomada fue una Spline cúbica natural. Una Spline cúbica es un polinomio de interpolación por tramos (exactamente, un polinomio de grado 3 en cada tramo) que interpola n+1 puntos de forma que la función Spline resultante sea continua, su derivada sea continua, y su segunda derivada también. Se denomina natural porque se le impone que su segunda derivada en los extremos sea nula.

Para interpolar usamos cuatro puntos de la tabla. Los dos puntos anteriores al medido y los dos posteriores. Como ya se comentó antes, se hace una excepción en el caso del primer y último tramo donde se cogen cuatro puntos pero son el primero y los tres siguientes en el primer tramo y el último y los tres anteriores en el tramo final.



Puntos tomados en el primer tramo



Puntos tomados en el último tramo

Sin entrar en detalles matemáticos, la rutina de cálculo lo que hace es con esos cuatro puntos y el valor medido calcular una serie de coeficientes que nos permiten averiguar el polinomio Spline. En el caso de nuestra tabla, al tener valores muy grandes los coeficientes de tercer y segundo grado son muy pequeños, lo que hace que sea aproximadamente lineal. Aún así, los cálculos obtenidos fueron satisfactorios y esta solución, además de en este sistema se implementará en otros que trabajen con tabla de valores (Lookup Table). En el anexo final se adjuntan las hojas de cálculo donde se simula el algoritmo del Spline y se comparan algunos segmentos con la rutina anterior y la nueva.

input $n,(t_i\ ),(y_i\ )$
for $i = 0, 1, \ldots, n-1$ **do**
  $h_i \leftarrow t_{i+1} - t_i$
  $b_i \leftarrow 6(y_{i+1} - y_i)/h_i$
end

$u_1 \leftarrow 2(h_0 + h_1)$
$v_1 \leftarrow b_1 - b_0$
for $i = 2, 3, \ldots, n-1$ **do**
  $u_i \leftarrow 2(h_i + h_{i-1}) - h_{i-1}^2/u_{i-1}$
  $v_i \leftarrow b_i - b_{i-1} - h_{i-1}v_{i-1}/u_{i-1}$
end

$z_n \leftarrow 0$
for $i = n-1, n-2, \ldots, 1$ **do**
  $z_i \leftarrow (v_i - h_i z_{i+1})/u_i$
end
$z_0 \leftarrow 0$
output $(z_i\ )$

4.5 NUEVA PLACA PARA EL KMS32 (Add-On)

Con objeto de integrar los dos sistemas anteriores (el KMS32 y el DPL2) se fabricaron nuevas placas para el KMS32 mucho más pequeñas, cambiando totalmente su forma.

Las nuevas placas no están diseñadas para llevar microcontrolador, son extensiones (Add-On) para el DPL2. Están pensadas para llevar hasta dos sensores KMT, ya que algunos sistemas (KMS360) son capaces de medir hasta 360º usando para ello dos sensores.

Durante la realización de esta segunda parte llegaron las nuevas placas y una vez montados todos los componentes se pudieron probar junto al DPL2.