TÍTULO:
Sismómetro para el registro de la actividad volcánica. Diseño Electrónico.

AUTOR: Santiago M. Migliorelli Falcone
TITULACIÓN: Grado en Ingeniería Electrónica Industrial y Automática
DIRECTOR: Joaquín del Río Fernández
DEPARTAMENTO: 710, Ingeniería Electrónica
FECHA: 1 de Julio de 2013
El grupo de investigación de SARTI, en colaboración con el proyecto de investigación de “Peligro Volcánico y Evaluación del Riesgo en Tenerife (PEVERTE)”, precisa realizar el diseño electrónico de un dispositivo de adquisición de datos sísmicos en una zona de riesgo volcánico.

Para ello se realizará el estudio de los componentes a utilizar para el diseño del dispositivo, seleccionando los que se adapten mejor a los requisitos de bajo consumo y bajo precio. Los datos adquiridos han de poder enviarse remotamente, a la vez de que han de almacenarse en una memoria externa. Se adquirirán datos provenientes de tres canales diferentes.

A raíz de la selección de estos componentes se realizará el diseño del hardware del dispositivo, de las comunicaciones necesarias, y de los componentes extra que sean necesarios para el correcto funcionamiento. Se construirá un prototipo del dispositivo en el que realizar todas las pruebas.

Luego se diseñará el software encargado de gestionar el funcionamiento de la aplicación, programada en lenguaje C. Una vez configurados los registros del microprocesador de manera adecuada, deberá gestionar la lectura de los datos con dos velocidades diferentes de adquisición (50 y 100 SPS), según la decisión del usuario, y poder enviar y/o almacenarlos en la memoria externa de tipo SD. También se gestionarán los diferentes tipos de comunicaciones (SPI y RS232).

Finalmente, una vez la aplicación sea programada y probada, se testeará su funcionamiento, mediante aplicaciones hechas en LabView y Matlab. Para ello se comprobarán los resultados de diferentes adquisiciones hechas en el laboratorio a través de los instrumentos.

### Palabras clave

<table>
<thead>
<tr>
<th>Sismómetro</th>
<th>Microprocesador</th>
<th>Analógico-digital</th>
<th>Memoria SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bajo consumo</td>
<td>MSP430F5438</td>
<td>ADS1246</td>
<td></td>
</tr>
</tbody>
</table>
Índice

1 Introducción ........................................................................................................................................... 3
   1.1 Justificación del TFG .................................................................................................................. 3
   1.2 Objetivos ........................................................................................................................................ 4
   1.3 Arquitectura del sistema .............................................................................................................. 4
2 Selección de los componentes electrónicos del sistema y características ........................................... 6
   2.1 Microprocesador ......................................................................................................................... 6
   2.1.1 Características del MSP430F5438......................................................................................... 7
   2.2 Convertidor analógico – digital ................................................................................................. 9
   2.2.1 Características del ADS1246 ............................................................................................. 11
   2.3 Geófono SM6 ............................................................................................................................. 12
   2.4 Tarjeta SD ...................................................................................................................................... 12
3 Hardware ............................................................................................................................................ 14
   3.1 Configuración de las comunicaciones ......................................................................................... 14
   3.1.1 Comunicación serie ............................................................................................................... 14
   3.1.2 Comunicación SPI ............................................................................................................... 15
   3.1.3 Funcionamiento de los ADS1246 ........................................................................................ 17
   3.2 Fuentes de reloj ........................................................................................................................... 19
   3.2.1 Frecuencias del MCLK, SMCLK y ACLK ........................................................................... 21
   3.3 Real time clock ............................................................................................................................ 21
   3.4 Esquema del conexiónado general ............................................................................................. 21
4 Software .............................................................................................................................................. 23
   4.1 Entorno de programación .......................................................................................................... 23
   4.2 HyperTerminal ............................................................................................................................ 25
   4.3 LabView ......................................................................................................................................... 26
   4.4 Aplicación de control del sistema .............................................................................................. 26
   4.4.1 Diagrama general ............................................................................................................... 27
   4.4.2 Adquisición, compresión, envío y guardado de los datos .................................................... 28
   4.4.3 Formato del envío de datos ................................................................................................. 29
   4.4.4 Formato del almacenamiento de datos ............................................................................... 30
   4.4.5 Funciones de configuración ................................................................................................. 31
   4.4.6 Función principal y funciones propias del sistema .............................................................. 36
   4.4.7 Funciones e interrupciones de recepción, compresión y envío de datos ......................... 37
   4.4.8 Funciones de almacenamiento de los datos en memoria externa .................................... 40
4.4.9 Menú de configuración

5 Caracterización del sistema
5.1 Dispositivos de laboratorio utilizados
5.2 Pruebas realizadas
5.3 Consumo del sistema
5.3.1 Consumo teórico
5.3.2 Consumo real
5.4 Resultados
5.4.1 Envío y visualización de los datos
5.4.2 Lectura de los datos almacenados en memoria SD
5.5 Prototipo

6 Conclusiones

7 Bibliografía

8 Anexos
1 Introducción

1.1 Justificación del TFG
Este Trabajo de Final de Grado ha sido realizado en el laboratorio del “Centre Tecnològic de Vilanova i la Geltrú (CTVG)”, y en junto con el equipo del Centro de Desarrollo de Sistemas de Adquisición Remota y Tratamiento de la Información (SARTI), colaboradores del proyecto de investigación en riesgo volcánico, “Peligro Volcánico y Evaluación del Riesgo en Tenerife (PEVERTE)’’.

La ciencia siempre ha hecho grandes esfuerzos en el estudio de la prevención de las catástrofes naturales. Las erupciones volcánicas pueden resultar una catástrofe natural, pero gracias al trabajo de cientos de científicos, este fenómeno natural es predecible. Una predicción a tiempo de la actividad volcánica evitaría hablar de catástrofe natural. Es por este motivo que la actividad sísmica de las zonas volcánicas ha de ser recopilada continuamente.

Se puede poner como ejemplo, la detección de temblores hecha por los dispositivos del Instituto Geológico Nacional en la isla del Hierro, en Octubre del 2011. Gracias a esta detección de los temblores, que finalmente hicieron que el volcán submarino entra en erupción, se pudo poner el alerta la isla, pudiendo, si hubiera sido necesario, tomar las medidas adecuadas para la prevención de una catástrofe.

La instrumentación en las zonas de riesgo volcánico debe operar durante periodos de tiempo largos (meses o incluso años), sin interrupciones, manteniendo la compatibilidad entre los datos obtenidos a lo largo del tiempo.

En los momentos en los que se detecta algún tipo de cambio en el comportamiento de la actividad sísmica de la zona volcánica, será necesaria la instalación de más dispositivos para poder caracterizar mejor la actividad que se está desarrollando. La instalación de equipos redundantes mejorará la fiabilidad del sistema ante la posibilidad de fallos en el funcionamiento de algún equipo, o en la comunicación con estos [1].

Cuando la actividad medida por los instrumentos fijados alcanza niveles que indican la posibilidad de una erupción, se procede a la instalación de equipos en zonas amenazadas por el riesgo de la erupción (estos dispositivos reciben el nombre de “instrumentos perdidos”). Será necesario que el coste de estos sea el menor posible, ya que funcionarán durante un período limitado de tiempo, ante la amenaza de que sean destruidos, perdidos, o simplemente porque su fuente de energía se agote.

En el campo de la instrumentación, hay cuatro maneras de estudiar y predecir la actividad llevada a cabo por un volcán: la sísmica, la deformación, la geoquímica y la electromagnética [2].

En el mercado actual, encontrar un equipo de adquisición de datos sísmicos no supondría ningún problema, tanto si se quiere estudiar el movimiento terrestre como el marítimo. El inconveniente que presentan estos equipos, suele estar relacionado con su precio, sus dimensiones, autonomía y con su fin, ya que estos equipos no están diseñados para estudiar la actividad sísmica volcánica.

Los sistemas de adquisición sísmica comerciales suelen ser de gran tamaño, esto sería un problema a la hora de intentar disimular el instrumento en el medio. Además, el elevado precio de ellos imposibilitaría la instalación de equipos redundantes. La autonomía tampoco es suficientemente elevada como para poder adquirir información durante un tiempo significativo y poder enviarla inalámbricamente. Por último, el fin de estos equipos suele estar dirigido en su gran mayoría a los seísmos provocados por terremotos.
Para el seguimiento de la actividad volcánica tiene especial relevancia el estudio del ruido sísmico. Existen importantes diferencias entre los instrumentos habitualmente utilizados para el estudio de la sismicidad tectónica y los sismómetros para el estudio vulcanológico. Se podría decir que para los sismólogos la señal de interés son las que provienen de terremotos o movimientos de las placas tectónicas, y el resto es ruido, mientras que para los volcanólogos la señal de interés es el ruido sísmico y los terremotos se consideran ruido.

Por estos motivos, y enmarcado dentro del proyecto de investigación PEVerTE se propone el diseño de la electrónica necesaria para la construcción de un prototipo de sismómetro que cumpla con los requisitos necesarios para realizar estudios vulcanológicos.

El sistema a diseñar, capturará la actividad sísmica registrada y, empaqueta la información para poder enviarla inalámbricamente además de almacenarla en una memoria interna.

1.2 Objetivos

El objetivo principal del Trabajo de Final de Grado es el diseño de la electrónica de control, registro y transmisión de datos, de un instrumento para el seguimiento de la actividad volcánica.

El dispositivo a diseñar ha de cumplir una serie de requisitos. Por una parte, el consumo de los diferentes componentes ha de ser lo suficientemente bajo, para que funcione durante un año mediante baterías. Esto es debido, a que este tipo de equipos suelen ser instalados en zonas menos pobladas y sin vigilancia, donde pueda recoger datos con menores interferencias de las actividades urbanas. Por lo tanto también será necesario que tenga un bajo impacto visual, siendo compacto y reducido, con el objetivo de evitar hurtos y vandalismos. Este requisito, será el más restrictivo.

Por otra parte, el dispositivo ha de recoger la señal generada por los sensores y transformarla en una señal digital, con una resolución en bits lo suficientemente precisa para poder caracterizar los datos sin cometer errores, y poder escoger entre recoger cincuenta o cien muestras por segundo.

Finalmente el equipo ha de tener un coste reducido, tanto de realización como de mantenimiento.

Por lo tanto el objetivo del proyecto será seleccionar los elementos adecuados que puedan cumplir las funciones requeridas, y el desarrollo de una aplicación en C, para el correcto funcionamiento de toda la cadena.

1.3 Arquitectura del sistema

El sistema de adquisición de datos, seguirá la estructura típica de la cadena de medida de la instrumentación electrónica.

En primer lugar está la captación del medio a estudiar, en éste caso, las vibraciones del suelo. Estas vibraciones serán registradas mediante un sensor de tipo geófono. Este tipo de sensores ofrece una diferencia de potencial al discriminar un movimiento. El dispositivo tendrá tres sensores de este tipo, colocados según la posición de los ejes cartesianos. De esta manera, se podrá caracterizar mejor cualquier tipo de movimiento que realice la tierra.

Luego, está señal analógica registrada por los tres sensores, se tendrá que digitalizar a través de tres convertidores analógicos-digitales, para poder ser procesada por el microprocesador. Las señales son digitalizadas y la información es enviada desde los convertidores analógicos-digitales hacia el microprocesador por medio de una comunicación de tipo serie (SPI, Serial Protocol Interface).
La información será empaquetada para, finalmente, poder ser almacenada en una memoria SD y/o poder ser enviada a un módulo de comunicaciones, mediante puerto serie (RS232, *Recommended Standard 232*).

![Diagrama de la arquitectura del sistema de adquisición](image)

**Figura 1-1. Arquitectura del sistema de adquisición a diseñar.**

Por lo tanto, en la figura 1-1, se puede observar un diagrama de bloques general de la estructura del sistema de adquisición como se había descrito.
2 Selección de los componentes electrónicos del sistema y características

El primer paso que se ha de dar una vez fijados los objetivos del proyecto, es la correcta selección de los componentes que vayan a integrar el sistema de recogida de datos.

Cada una de las diferentes partes ha de cumplir su función con la mayor precisión, además de consumir la mínima potencia posible y, a ser posible, tener un precio que no encarezca el dispositivo de manera desmesurada.

Por lo tanto es necesario hacer una investigación de los diferentes componentes que hay en el mercado, buscando y comparando aquellos que cumplan los requisitos mínimos, para hacer la mejor elección posible y, de esta manera, optimizar el sistema en este sentido.

Este estudio estará centrado básicamente, en el consumo. Es decir, entre dos componentes que cumplan las características básicas, la decisión de la elección estará basada en aquel componente que presente un consumo menor y un precio asequible dentro del presupuesto.

2.1 Microprocesador

El microprocesador es la parte fundamental de todo el sistema, ya que es el encargado de ejecutar la aplicación del empaquetamiento de los datos, además de gestionar la comunicación con los diferentes componentes de la cadena de medida.

A la hora de escoger entre un microprocesador y otro, se ha de observar la cantidad de canales de comunicación que tenga principalmente. Para el presente caso, serán necesarios un mínimo de cuatro buses de comunicación serie SPI, y tres puertos de comunicación serie RS232.

También será importante que disponga de RTC (Real Time Clock), para poder llevar la cuenta del tiempo real. Es importante para la aplicación que el momento de captación de los datos sea recogido también, de cara al tratamiento al que pueda ser sometida la información.

Otro punto a observar será la cantidad de temporizadores que disponga el dispositivo. Siempre es importante contar con una cantidad de temporizadores adecuada, ya que suelen ser de gran interés a la hora de programar todo tipo de aplicaciones.

Finalmente otro aspecto a tener en cuenta será la cantidad de pines de entrada-salida que disponga el microprocesador. Estos podrán ser usados para diferentes cuestiones que puedan surgir una vez empezada la programación de la aplicación. Por lo tanto si el número es elevado podría evitar un problema de falta de pines.

Al ser un sistema de adquisición, el procesado de los datos requerirá tenerlos almacenados en el propio microprocesador, antes de guardarlos en la memoria externa o enviarlos inalámbricamente. Por lo tanto será importante que el dispositivo a escoger tenga una capacidad de memoria interna suficientemente grande como para poder almacenar temporalmente algunos datos.

A continuación, en la tabla 2-1, se puede observar una comparación entre las características de algunos microprocesadores que podrían realizar las tareas necesarias en el sistema. Se compararán sus características para decidirse por aquel más adecuado, teniendo en cuenta, principalmente, la relación entre el precio y el consumo.
Finalmente, una vez analizados los diferentes tipos de microprocesadores disponibles, se escogerá el MSP430F5438 de Texas Instruments para el desarrollo de la aplicación, por su bajo consumo, su bajo coste y cumplir con las necesidades en cuanto a pines de entrada-salida, RTC y comunicaciones.

### 2.1.1 Características del MSP430F5438

El microprocesador de Texas Instruments MSP430F5438 tiene prestaciones suficientes para adaptarse y desarrollar la actividad necesaria para el sistema descrito en este proyecto.

Principalmente, como se ha comentado, el microprocesador tendrá que disponer de cuatro buses de comunicación SPI, tres puertos serie RS232, reloj en tiempo real (RTC) y una disponibilidad amplia de pines de entrada y salida.

La disponibilidad de los pines será especialmente importante de cara al conexionado de los diferentes periféricos. Estos, aparte de estar conectados al puerto de comunicación (RS232 o SPI), a veces, precisan de pines de selección, los cuales han de estar conectados al microprocesador.
En la figura 2-1 [3], se puede observar una vista desde arriba del microprocesador y la función de cada uno de sus pines. Como se puede observar, hay hasta diez puertos de ocho pines cada uno. Alguno de estos pines no podrá ser utilizado como pin de entrada-salida, al utilizarse su otra función, ya sea como comunicación SPI, serie, o fuente de reloj. Más adelante, cuando se precisen las funciones se explicarán.

Por lo tanto, las principales características de este microprocesador son las siguientes:

- Rango de Voltaje de suministro: de 2.2V a 3.6V
- Consumo de energía extremadamente bajo (en modo activo 312µA/MHz)
• Tres Temporizadores de 16 bits
• Convertidor analógico-digital de 12 bits
• Ocho interfaces de comunicación serie universal (cuatro SPI y cuatro RS232)
• 18 MHz de velocidad de operación
• Tres canales de DMA interna
• Temporizador básico con característica de Reloj en Tiempo Real (RTC)
• Cristal interno de 32 kHz
• Posibilidad de incorporar Cristales de alta frecuencia hasta 32 MHz
• 256kB de memoria Flash
• 16kB de memoria SRAM
• 82 pines de entrada-salida (GPIOs)

En la figura 2-2, se puede observar el diagrama de bloques funcional del MSP430F5438 extraído de la hoja de características del fabricante [4]. En él se puede observar las características mencionadas anteriormente, y los buses internos del microprocesador para la comunicación interna de cada uno de los bloques.

![Diagrama de bloques del microprocesador MSP430F5438](image)

Figura 2-2. Diagrama de bloques del microprocesador MSP430F5438

Más adelante se especificará cómo se configurará cada uno de los bloques del microprocesador para el correcto funcionamiento de la aplicación.

2.2 Convertidor analógico – digital

El convertidor analógico-digital será el encargado de transformar la señal analógica obtenida de los sensores, a una señal digital de manera que el microprocesador sea capaz de entender.

Como se ha dicho anteriormente, se dispondrán de tres sensores, uno para cada eje de coordenadas, por lo tanto esto supone indirectamente una característica esencial del convertidor. Se tendrá que optar por un convertidor analógico-digital de mínimo tres canales, o bien, por tres convertidores analógicos de un solo canal.
Siguiendo la metodología para la elección del microprocesador, primero se seleccionarán algunos dispositivos que cumplan con los requisitos de funcionamiento mínimos y luego, en base a su consumo y precio, se seleccionará uno de los dispositivos.

En la tabla 2-2, se observa la selección de cinco dispositivos que cumplirían con las funciones deseadas.

<table>
<thead>
<tr>
<th>Marca</th>
<th>Analog Devices</th>
<th>Analog Devices</th>
<th>Cirrus Logic</th>
<th>Texas Instruments</th>
<th>Texas Instruments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modelo</td>
<td>AD 7710</td>
<td>AD 7193</td>
<td>CS 5532</td>
<td>ADS1248</td>
<td>ADS1246</td>
</tr>
<tr>
<td>Número de canales</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Muestras por segundo máximas</td>
<td>1028</td>
<td>4800</td>
<td>3840</td>
<td>2000</td>
<td>2000</td>
</tr>
<tr>
<td>Ganancia</td>
<td>1 – 128</td>
<td>1 – 128</td>
<td>1 – 64</td>
<td>1 – 128</td>
<td>1 – 128</td>
</tr>
<tr>
<td>Voltaje de trabajo</td>
<td>3 V a 5.25 V</td>
<td>3 V a 5.25 V</td>
<td>±2.5V</td>
<td>2.7V a 5.25V</td>
<td>2.7V a 5.25V</td>
</tr>
<tr>
<td>Consumo</td>
<td>53mW</td>
<td>17.5mW</td>
<td>70mW</td>
<td>2.3mW</td>
<td>2.3mW</td>
</tr>
</tbody>
</table>

Tabla 2-2. Comparación de cinco diferentes convertidores analógicos-digitales

En este caso la elección podría haber sido, tanto la del ADS1248 de Texas Instruments, como la de tres ADS1246 de la misma marca. Finalmente se ha optado por la elección de tres ADS1246, ya que el ADS1248. En este caso, la elección se basará principalmente por el bajo consumo de este dispositivo.

Como se puede observar en la figura 2-3, extraída de la hora de características del fabricante [5], el ADS1248 pese a tener cuatro canales de adquisición diferencial, solo presenta un modulador para convertir la señal. Esto representa un problema a la hora de adquirir por más de un canal, ya que esta adquisición no se realizará en paralelo, sino que se realizará en serie, habiendo un desfase entre las muestras de un canal y otro.

Por lo tanto se optará por la instalación de tres módulos ADS1246.
2.2.1 Características del ADS1246

El ADS1246 es un convertidor analógico-digital de 24 bits de precisión y bajo consumo. A continuación se detallan las principales características de interés del dispositivo [5]:

- Conversión a datos de 24 bits
- Velocidades de conversión de 5, 10, 20, 40, 80, 160, 320, 640, 1000 y 2000 muestras por segundo
- Amplificador de ganancia de bajo ruido de x1, x2, x4, x8, x16, x32, x64, x128
- Detección de rotura de sensor
- Sensor interno de temperatura
- Sistema de calibración
- Comunicación SPI
- Alimentación analógica unipolar (+2.7V hasta +5.7V) o bipolar (+2.5V)
- Alimentación digital: +2.7V hasta +5.7V
- Temperatura de operación entre -40°C a 125°C
- Consumo (a 20SPS): 1.4mW con AVDD = DVDD = 3.3V y 2.3mWAVDD = DVDD = 5V

En lo que respecta a la estructura interna, el ADS1246 es una versión simplificada del ADS1248, cuyo diagrama se puede observar en la figura 2-3. A diferencia de este, solo presenta un canal. Se puede ver su diagrama en la figura 2-4. En ella es fácilmente reconocible que la estructura es la misma que se observa en el otro modelo, pero con la diferencia de que ahora, solo hay un canal disponible.

![Diagrama del ADS1246](image)

**Figura 2-4. Estructura del convertidor ADS1246**

Este dispositivo tiene diferentes modos de trabajo. A través de diferentes comandos SPI enviados desde el microprocesador, se puede configurar la manera de trabajar del convertidor. El microprocesador se comunicará con los tres convertidores a través de tres buses SPI diferenciados. Además, cada uno de los tres convertidores tendrá sus respectivos pines de START, RESET, CS y DRDY conectados a un pin de entrada-salida del microprocesador.
El dispositivo dispone de diferentes metodologías de funcionamiento. Más adelante se especifica cómo se configurará para su correcto funcionamiento.

### 2.3 Geófono SM6

Para registrar el movimiento de la tierra, se utilizará un sensor de tipo geófono. Un geófono es un transductor de desplazamiento, velocidad o aceleración que convierten el movimiento del suelo en una señal eléctrica.

![Imagen del geófono SM6](image1)

**Figura 2-5. Sensor de tipo geófono SM6**

En el sistema de medida a diseñar, se utilizarán tres geófonos, uno para cada eje de coordenadas. Por lo tanto se tendrán que situar de la manera correcta.

Los geófonos están formados por acelerómetros magnéticos. Las características principales de estos sensores son las siguientes [6]:

- Sensibilidad de 28.8V/ms⁻¹
- Ancho de banda de 4.5Hz a 100Hz

### 2.4 Tarjeta SD

Para guardar los datos en una memoria física externa se utilizará una tarjeta SD PRETEC SDY01G 60X, como la que se puede observar en la figura 2-6.

![Imagen de la tarjeta SD](image2)

**Figura 2-6. Tarjeta de memoria externa SD**
Según la hoja de características del fabricante, las características de esta tarjeta serán las siguientes:

- Capacidad de 1GB
- Velocidad de transmisión de datos de hasta 9MB por segundo
- Bajo consumo
- Estado sólido - No volátil: los datos no son perdidos cuando se desconecta la alimentación

El modo de comunicación con la tarjeta será mediante SPI. Más adelante se explica la configuración del software para la creación de ficheros y su escritura.
3 Hardware
Anteriormente se han explicado los diferentes dispositivos que formarán el equipo de adquisición. A continuación se especificará cómo se configurará cada uno de estos dispositivos y la manera de comunicación entre ellos.

3.1 Configuración de las comunicaciones
Una vez seleccionados todos los componentes del sistema, ya se puede especificar el tipo de conexión que habrá entre el microprocesador con los demás dispositivos. En la figura 3-1, se puede observar un diagrama de bloques del equipo, junto al tipo de comunicación que habrá entre convertidores, ordenador, comunicación y memoria externa, con el microprocesador.

![Figura 3-1. Diagrama de bloques de las comunicaciones entre los diferentes periféricos del sistema](image)

Como se puede apreciar, se tendrán que configurar cuatro puertos SPI y dos RS232. A continuación, se detallará la configuración y la metodología de comunicación entre los dispositivos.

3.1.1 Comunicación serie
La comunicación serie, como ya se ha mencionado se efectuará según el estándar RS232. Estará presente en el equipo, en el envío de los datos desde el micro al sistema de comunicación por radiofrecuencia, y en la comunicación con un ordenador, para proceder a la configuración del dispositivo.

El tipo de comunicación RS232 (*Recommended Standard 232*) designa una norma para el intercambio de una serie de datos binarios entre un equipo terminal de datos y un equipo de comunicación de datos.

En este caso, el microprocesador actuará como comunicador de datos, de cara al envío de información obtenida, y como receptor de datos, de cara a recibir la configuración desde un ordenador.

Las tramas de datos, seguirán la estructura de la figura 3-2. Se enviarán datos de ocho bits, sin paridad, y con un bit de Stop. Por lo tanto, el receptor recibirá diez bits, el de *Start*, el dato y, finalmente, el bit de *Stop*. 

---

**Sismómetro para el registro de la actividad volcánica. Diseño Electrónico**

**Grado en Ingeniería Electrónica Industrial y Automática**

**Santiago M. Migliorelli Falcone**

---

14
Esta comunicación será del tipo asincrónica, ya que la comunicación no seguirá ninguna fuente de reloj. La velocidad de envío de los datos será de 115200 bits por segundo.

El microprocesador trabaja con tensiones del tipo TTL (transistor-transistor logic). Esto quiere decir, que la información enviada o recibida por el microprocesador, tomará como cero lógico un valor de tensión de cero voltios, y como uno lógico, un valor de cinco voltios. En cambio las comunicaciones RS232, toman como cero lógico, un valor negativo de tensión, entre tres y quince voltios, mientras que como uno lógico, toman el mismo valor de tensión que para el cero, pero positivo. Esta problemática obliga a la instalación de un dispositivo de conversión de datos TTL a RS232.

El dispositivo que se instalará para dicho fin será un MAX232 de MAXIM [7], como el que se puede ver en la figura 3-3.

El dispositivo dispone de la capacidad para conectar dos entradas y dos salidas, por lo tanto se podrá aprovechar para las dos comunicaciones RS232.

3.1.2 Comunicación SPI
El otro tipo de comunicación serie que se presentará en el dispositivo es la comunicación SPI. Esta se producirá entre los convertidores analógicos-digitales y la tarjeta de memoria externa SD, con el microprocesador.
En este tipo de comunicación serie se produce siempre entre un dispositivo que actúa como maestro y el que actuará como esclavo. Al contrario que en RS232, la comunicación SPI es síncrona, por lo tanto sigue una fuente de reloj.

![Figura 3-4. Esquema de la comunicación SPI entre maestro y esclavo, y sus líneas de conexión](image)

En la figura 3-4 se puede observar un esquema general de cómo será comunicación SPI. Por un lado está el maestro, el cual será el microprocesador. Por el otro lado están los diferentes esclavos, los cuales serán los tres convertidores y la memoria SD.

Como se puede observar, cada esclavo tiene tres pines de entrada y uno de salida a conectar. El primero de ellos será la fuente de reloj, proporcionada por el maestro, el segundo la entrada de datos del maestro y el tercero, un pin de selección del dispositivo (chip select). Este pin se activará si el maestro quiere comunicarse con ese esclavo. Mientras este desactivado, no habrá comunicación entre maestro y esclavo. El pin de salida corresponde al enviado de datos desde el esclavo al maestro. Por otra parte, la conexión del maestro será antagónica.

En cuanto a las tramas que sean enviadas por los dispositivos, será configurada de la siguiente manera. Se enviarán datos de ocho bits, empezando por el bit de mayor peso MSB (Most significant bit), como se puede apreciar en la figura 3-5.
En este caso, todos los niveles de tensión serán TTL, por lo tanto no será necesario ningún tipo de ajuste de las tensiones de entrada-salida.

3.1.3 Funcionamiento de los ADS1246

El convertidor ADS1246 de Texas Instruments ofrece una gran cantidad de configuraciones para su funcionamiento. Como se ha especificado anteriormente, la comunicación entre el microprocesador y los convertidores será del tipo SPI, por lo tanto se conectará cada uno, con un puerto SPI del microprocesador.

Observando la figura 3-6, se puede ver que aparte de los pines correspondientes a la comunicación SPI, el dispositivo requiere de una alimentación analógica (AVDD y AVSS), una digital (DVDD y DVSS) y una señal de referencia (REFP y REFN). El dato de salida del dispositivo estará siempre dentro del rango de esta tensión de referencia. También, se tendrá que tener en cuenta, que la señal de entrada también ha de estar dentro del rango de la tensión de referencia. Los pines de AINP y AINN, corresponden a la señal de entrada.

Por otro lado, hay una serie de pines que tendrán mucha importancia en la aplicación que se explica más adelante, por lo tanto su conexión tendrá particularidades. Es el caso por ejemplo del pin de DRDY. Este pin es el encargado de avisar que el dispositivo ha convertido un dato y está listo para transmitirlo. Cuando se realice la explicación de la aplicación se podrá ver como se interactuará con él.

Asimismo, el pin de START también tiene un papel importante en cuanto al funcionamiento. Este pin controlará el estado de funcionamiento del dispositivo. Principalmente habrá dos tipos de funcionamiento de los convertidores, conversión continua o discontinua.
En la figura 3-7, extraída de la hoja de características del fabricante [5], se puede ver cómo actúa la conversión continua. Si se mantiene el pin de START en su valor alto, el dispositivo irá convirtiendo datos continuamente y avisando, mediante el pin de DRDY, de que ya están listos para ser leídos. Directamente sin previa instrucción, comenzará a convertir el siguiente dato. Sin embargo esta configuración, pese a ser más fácil de implementar, presenta un problema para el dispositivo a fabricar. El convertidos ADS1246 nos da la opción de trabajar a unas velocidades de conversión de datos que no son las deseadas, ya que no trabaja ni a cincuenta ni a cien muestras por segundo.

Sin embargo, hay otro método de funcionamiento que permitirá poder tener las muestras por segundo deseadas. En la figura 3-8 se puede observar éste método.

Aqui, en vez de mantener en calor alto el pin de START, lo que se realizará, es un pulso. De esta manera, se le estará pidiendo al convertidor, que inicie la conversión de un dato. Cuando este dato esté listo (observando el pin de DRDY), se leerá y se le pedirá que inicie la conversión de otro nuevo.

Con esta metodología, se puede pedir al convertidor la cantidad de datos que se quieren cada segundo, es por este motivo que será posible pedir las muestras por segundo que sean más apropiado para la aplicación.

El pin de RESET, servirá para poner los registros de configuración del convertidor a su valor por defecto. Más adelante se explicará la configuración de dichos registros.

Por último, relacionado con el hardware de los convertidores analógicos-digitales, está el reloj interno. Como se puede apreciar en la figura 3-9, el pin número tres corresponde al reloj de trabajo. Estos dispositivos disponen de un reloj interno, el cual si activaría si se conectara el pin de CLK a tierra. Sin embargo, si se deja que cada uno de los convertidores trabaje bajo su propia fuente de reloj, sería imposible poder controlar
el sincronismo entre ellos tres. Es por este motivo que se optará por conectar una fuente de reloj externa proveniente del microprocesador.

![Diagrama de conexión de los tres convertidores ADS1246](image)

Figura 3-9. Esquema del conexionado de los tres convertidores ADS1246

La velocidad máxima de trabajo del convertidor ADS1246 es de 4.5MHz. Por lo tanto, los tres pines de CLK irán conectados al mismo pin del microprocesador, que les proporcionará una señal de reloj de 4MHz. De esta manera, las conversiones, siempre y cuando se arranquen en el mismo momento, estarán sincronizadas.

### 3.2 Fuentes de reloj

Los relojes son parte fundamental del sistema, para el control de las velocidades de envío en las comunicaciones, de trabajo del microprocesador, por lo tanto es importante ver cómo funciona el sistema de relojes del MSP430F5438.

Como se puede observar en la figura 3-10, extraída de la hoja de características del fabricante [3], el microprocesador dispone de tres señales de salida de reloj diferentes, la maestra (MCLK), la maestra del subsistema (SMCLK) y la auxiliar (ACLK). La MCLK será la encargada de definir la velocidad de trabajo del microprocesador, por lo tanto, es interesante que se configure a su máxima capacidad. Luego, la salida auxiliar ACLK, será importante que esté configurada a 32768Hz, para el correcto funcionamiento de la RTC, como se explicará más adelante. Por último, la salida SMCK será la encargada de gestionar las velocidades de las comunicaciones, y también actuará como fuente de reloj para los convertidores analógicos-digitales.
Para una correcta sincronización de las señales de reloj, se escogerá la misma fuente para las tres. Se utilizará la fuente de referencia interna REFOCLK, la cual es un oscilador de baja frecuencia (32768Hz). De esta manera, se evitará el tener que gestionar un sincronismo entre las tres señales, además de evitar la colocación de cristales externos que podrían aumentar el consumo del sistema.

En los próximos apartados, se explicará la funcionalidad de cada fuente de reloj. Su configuración se detallará en la explicación del software.

Figura 3-10. Diagrama de bloques de las fuentes de reloj del microprocesador MSP430F5438
3.2.1 Frecuencias del MCLK, SMCLK y ACLK

Como se ha comentado anteriormente, la señal maestra de reloj es la encargada de gestionar la velocidad de trabajo del microprocesador. Es interesante que esta velocidad sea lo más alta posible, para que no suponga un problema a la hora de gestionar la aplicación. Observando la figura 3-11 extraída de la hoja de características del fabricante [4], la velocidad máxima de trabajo del microprocesador es de 18MHz, por lo tanto se configurará para que funcione a esta velocidad.

![Tabla de frecuencias](image)

**Figura 3-11. Máxima frecuencia de trabajo del MSP430F5438**

La señal del subsistema maestra de reloj será la encargada de gestionar la velocidad de las comunicaciones y como fuente de reloj de los convertidores. Por lo tanto, tanto la comunicación mediante RS232 y SPI, utilizarán como fuente de reloj la SMCLK.

El hecho de que los convertidores ADS1246 utilicen la señal SMCLK como fuente de reloj impone una limitación a esta señal. Como se ha comentado anteriormente, el reloj interno del ADS1246 puede tener un valor máximo de 4.5MHz. Por lo tanto esta señal de reloj se estipulará a esta velocidad.

Por último, como se ha comentado anteriormente, la señal auxiliar ACLK se mantendrá en el mismo valor que la fuente REFoCLK (32768Hz), ya que será una condición necesaria para el correcto funcionamiento de la RTC del sistema, que se explica en el próximo apartado.

3.3 Real time clock

Para poder tener un control del tiempo real en el sistema, será necesaria la utilización del Real Time Clock (RTC).

En el MSP430F5438 existe una RTC, la cual puede ser utilizada en modo contador o en modo calendario. Se utilizará el modo calendario para poder gestionar el sistema de ficheros, que se explicará más adelante.

En este modo, la RTC llevará una cuenta del tiempo, actualizando, cuando sea necesario, diferentes registros para poder consultar el año, mes, día del mes, día de la semana, hora, minuto y segundo. Será necesaria una configuración inicial que se explicará en el apartado de software.

La RTC puede trabajar en un rango de tiempos del año 1901 al 2099, por lo tanto será útil y suficiente para la aplicación desarrollada.

Es necesario que para un funcionamiento correcto, la fuente de reloj que utilice la RTC en modo calendario, trabaje a 32768Hz, ya que si no la cuenta no se hará correctamente. En este sentido, aparece una imposición de diseño en la señal ACLK, como se había comentado anteriormente, ya que en modo calendario la fuente de reloj es la ACLK, sin posibilidad de ser cambiada.

3.4 Esquema del conexionado general

En la Figura 15 se puede observar cómo será el circuito general del sistema de adquisición de datos. Todo será montado alrededor del microprocesador.
Figura 3-12. Esquema general de conexionado del sistema de adquisición de datos diseñado

Más adelante se podrá ver una imagen del prototipo construido, en base a este esquema de conexiones.
4 Software
En este apartado se explicará todo lo relacionado con la aplicación programada, desde que tipo de software se utiliza para la programación y visualización de resultados, hasta la explicación de las diferentes funciones del programa y su integración.

4.1 Entorno de programación
El programa utilizado para el diseño de la aplicación en C, es el Code Composer Studio v5 de Texas Instruments. Este entorno de programación está preparado para el desarrollo de aplicaciones sobre los procesadores de Texas Instruments.

El programa ofrece un entorno de trabajo sencillo e intuitivo. En primer lugar tendremos el entorno de edición de las aplicaciones, el cual se puede apreciar en la figura 4-2. En él, se pueden ver a la izquierda los diferentes proyectos que se tengan en el entorno de trabajo, en medio los diferentes archivos que se estén editando y compilando, y finalmente abajo, la consola y el listado de errores o advertencias.

Figura 4-1. Entorno de trabajo del Code Composer Studio v5

La ventaja que ofrece este programa, es que al crear un nuevo proyecto, se puede especificar qué tipo de microprocesador de la casa Texas Instruments se utilizará, como se puede apreciar en la figura 4-2. El programa directamente incluirá las cabeceras del fabricante que sean necesarias para el correcto funcionamiento de la aplicación, facilitando al usuario su búsqueda e integración.
Por otra parte, está el entorno de depurado de la aplicación, donde se ejecutará. En la figura 4-3 se puede observar las diferentes partes de este entorno.

Figura 4-3. Entorno de depurado de la aplicación en Code Composer Studio

Principalmente, las dos ventanas de la parte superior brindarán la información más útil para el programador. A la izquierda se puede observar donde se ha detenido el programa, mostrando el árbol de ejecución. A la
derecha, se pueden observar los valores de las diferentes variables del programa, o de expresiones. Por otra parte, en la zona central se puede observar en que zona del programa se ha detenido la aplicación y el mapa de memoria del microprocesador. Por último, en la zona inferior, se puede observar la consola.

4.2 HyperTerminal
El programa HyperTerminal de Windows, será utilizado para la comunicación entre el dispositivo y el ordenador, para configurarlo. Esta comunicación será, como se ha explicado anteriormente, mediante el estándar RS232.

Para realizar esto, una vez abierto, se tendrá crear una nueva comunicación y, configurarla. Una vez esté el programa en ejecución, se podrá ver por pantalla lo que el microprocesador envíe hacia el HyperTerminal y, a su vez, escribir y enviar información. En la figura 4-4 se puede observar un esquema de lo anteriormente descrito.

![Imagen de HyperTerminal](image)

Figura 4-4. Esquema de funcionamiento de HyperTerminal

Se observa como siguiendo los pasos de la configuración, acaba habiendo una comunicación con el equipo.

Santiago M. Migliorelli Falcone
4.3 LabView
Para poder visualizar los datos enviados por el puerto serie para las comunicaciones inalámbricas, se utilizará el programa de National Instruments LabView. Es una plataforma y entorno de desarrollo para diseñar sistemas, con un lenguaje de programación visual gráfico.

El programa se divide en dos partes. En primer lugar, el diagrama de bloques, donde se creará la aplicación a ejecutarse. Esto se realizará mediante la incorporación de diferentes bloques funcionales conectados entre ellos. Y, por otro lado, está el Panel Frontal, una interface con la que interacciona el usuario y donde se puede visualizar información.

Más adelante se expondrá la aplicación que se realiza en este programa.

4.4 Aplicación de control del sistema
En este apartado se explicará la estructura de la aplicación programada en el microprocesador. Ésta estará programada en el lenguaje de programación C.

De forma general, se podría decir que la aplicación sigue el esquema de la figura 4-5. En ella se puede ver que en primer lugar, se hará una configuración general del equipo, para poder habilitar los diferentes periféricos y relojes del sistema según las necesidades del conjunto. Luego, dependiendo de las características seleccionadas por el usuario en el menú de configuración, el programa configurará lo que reste para poder comenzar su ejecución.

![Figura 4-5. Esquema general de la aplicación](image)

Más adelante se podrá observar con mayor claridad la funcionalidad de cada una de las partes de la aplicación.

Principalmente, inicialmente se programará la configuración de todos los dispositivos. Luego, una vez vayan llegando los datos, se almacenarán en unos buffers, que a medida que se vayan llenando, se vaciarán enviando y/o guardando la información. A su vez, se tendrá que ir controlando el paso del tiempo.
4.4.1 Diagrama general
La aplicación seguirá una ejecución según la estructura que se puede observar en la figura 4-6.

![Diagrama general](image)

**Figura 4-6. Esquema del funcionamiento de la aplicación**

Para poder llevar a cabo la ejecución de esta estructura, será necesario el diseño de una serie de funciones y de interrupciones, las cuales se explicarán en los siguientes apartados. Se dividirán principalmente en los siguientes tipos:

- Funciones de configuración
- Función principal
- Funciones e interrupciones de recepción, compresión y envío de datos
- Funciones de almacenamiento de los datos en memoria externa
- Menú de configuración

Antes de explicar el funcionamiento de estas funciones, conviene explicar cómo se tratarán los datos recibidos de los convertidores analógico-digitales.

**4.4.2 Adquisición, envío y almacenamiento de los datos**

A continuación se expondrá el funcionamiento de la aplicación en lo relacionado a la adquisición de los datos y su almacenamiento intermedio en el microprocesador, antes de ser enviados y/o guardados en la memoria externa.

Una vez la aplicación ha configurado todos los dispositivos, comienza la adquisición de los datos. Estos, mientras vayan siendo adquiridos, se tendrán que ir almacenando antes de ser enviados o guardados. Por lo tanto se creará una estructura de dos buffers por canal para realizar este almacenamiento intermedio.

Los datos se leerán dentro de la interrupción del pin 5 del puerto 1. Este pin estará configurado como pin de entrada, y estará conectado a la señal de DRDY proveniente del ADS1246 del canal uno, el cual siempre funcionará, tanto si se configura uno, dos, o tres canales. Cuando el convertidor avise que hay dato, mediante el DRDY, saltará la interrupción y se procederá a leer los datos de los puertos SPI. Una vez leídos y guardados en variables intermedias, se guardarán en uno de los buffers correspondientes. Esto quiere decir, que los datos leídos del canal uno, irán al primer buffer del canal uno, los datos del canal dos, al primer buffer del canal dos, y los datos del canal tres, al primero del canal tres.

Este procedimiento continuará así hasta que los buffers donde se estén guardando los datos estén llenos. Una vez pase esto, los próximos datos se comenzarán a guardar en los segundos buffers.

Mientras tanto, en el programa principal, una vez se haya llenado el buffer, se procederá a enviar y/o almacenar estos datos. Cuando se haya acabado de escribir los datos en la memoria externa y se hayan terminado de enviar por puerto serie, estos primeros buffers quedarán libres. Cuando los segundos buffers estén llenos de datos nuevos, se podrá repetir el proceso, guardando datos nuevos en los primeros buffers, y enviando o almacenando los datos de los segundos.

De esta manera, se conseguirá el poder ir leyendo los datos en el mismo momento en el que estén listos, y se podrán ir enviado y guardando al mismo tiempo.

El envío y almacenamiento, se deben realizar cada segundo. Por lo tanto, dependiendo de las muestras por segundo que se hayan configurado (50 o 100), los buffers tendrán una capacidad de 50 o 100 datos.

Los buffers tendrán la estructura que se ve en la figura 26. Cada dato ocupa tres bytes, por lo que cada posición del buffer constará de una estructura de tres posiciones, la primera con el byte de mayor peso, la segunda con el byte intermedio, y la tercera, con el de menor peso.
El buffer se guardará y hasta la posición media.

4.4.3 Formato del envío de datos

Los datos obtenidos de la conversión se enviarán por puerto serie RS232, como se ha comentado anteriormente. Este envío no será continuo, se enviarán paquetes de datos por segundo.

Estos paquetes de datos, seguirán una estructura como la que se puede apreciar en la figura 27. En ella se puede observar, en sistema hexadecimal, una trama de datos, correspondiente a una adquisición con tres canales.

Figura 4-7. Estructura de un buffer de datos

El buffer se creará de cien posiciones, pero los punteros que se utilicen en la aplicación para recorrer el buffer guardando y leyendo los datos, llegarán hasta la última posición, si se configura cien muestras por segundo, o hasta la posición cincuenta, si se configuran cincuenta muestras por segundo.

Figura 4-8. Ejemplo de trama recibida por puerto RS232

Cada uno de estos paquetes, contendrá el número indicado de muestras por segundo (cincuenta o cien), y una cabecera para la identificación de estos datos, como se observa en la figura anterior. En ella se puede observar en primer lugar la cabecera y luego los cien datos de 24 bits cada uno de cada canal.

La cabecera siempre ocupará dieciocho bytes, cada uno de estos bytes, solos o en conjunto, tendrán una información, explicada en la tabla:
Cuando la adquisición de datos se haga de más de un canal, siempre se enviará cada segundo, una trama para cada canal, con una cabecera para los tres canales al principio del envío.

La cabecera se tendrá que ir actualizando a medida que pase el tiempo. Por lo tanto en el programa principal, cada vez que se vayan a enviar tramas de datos, se comprobará si hay que actualizar los valores, y si es necesario se hará.

### 4.4.4 Formato del almacenamiento de datos

Los datos, cuando se almacenen en la memoria externa, seguirán un formato de ficheros, de manera que sea sencilla su clasificación.

En este caso, a diferencia que en el envío por puerto serie, los datos se guardarán en ficheros .SIS, pero no dispondrán de una cabecera para identificarlos. Los datos se identificarán a través de los nombres de los ficheros y las carpetas donde estén contenidos.

La estructura que se seguirá es la que se puede observar en la figura 4-9, donde se ve cómo se organiza la estructura por carpetas, luego de una simulación de adquisición.

Se crearán una, dos o tres carpetas principales, dependiendo del número de canales a utilizar. En el ejemplo visto en la figura 4-9, se utilizan tres canales, por lo tanto los datos del canal uno, estarán dentro de la carpeta SAR1, los del canal dos en SAR2 y finalmente los del canal tres, en SAR3. Luego, dentro de cada una de ellas, habrá una carpeta correspondiente al año en el que se esté realizando la captura. A su vez, dentro de la carpeta de años, se encontrará una carpeta por cada mes, y dentro de ellas, los ficheros de datos.

Los ficheros de datos serán nombrados de la siguiente manera. En primer lugar, los primero dos dígitos corresponderán al mes, luego, los siguientes dos, al día del mes, y finalmente, los últimos dos, corresponderán a la hora del día (de 0 a 23 horas). De esta manera, se evita tener que incluir cabeceras dentro de los ficheros.
Dentro de estos ficheros, con formato .SIS, estarán los datos obtenidos de la adquisición del canal correspondiente durante una hora. Por lo tanto, si cada fichero corresponde a una hora de adquisición, observando la tabla 4-2, se ven la cantidad de datos que habrá en cada fichero.

<table>
<thead>
<tr>
<th>Cantidad de datos en un fichero .SIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 muestras por segundo</td>
</tr>
<tr>
<td>100 muestras por segundo</td>
</tr>
</tbody>
</table>

Tabla 4-2. Tamaño de los ficheros según la cantidad de muestras por segundo

4.4.5 Funciones de configuración
Las funciones de configuración, son todas aquellas encargadas de inicializar los periféricos, relojes y buffers del sistema. A continuación se explican una por una todas ellas.

- void init_clocks(void);

La función init_clocks es la encargada de configurar las señales de reloj del sistema de la manera que se ha comentado en el hardware.

Se configurarán los registros necesarios para coger como fuente de reloj la señal REFO. Como se mencionó, esta señal es de 32768Hz, el cual es el mismo valor que necesita el ACLK. Para poder proveer al MCLK y al SMCLK, se necesitará el Frequency Locked Loop (FLL) para generar el valor deseado para estas señales.
Configura los registros de la figura 26, para que la señal siga el recorrido marcado en color rojo.

- **void initRS232(void);**

Esta función se encarga de habilitar los puertos USCI_A0 y USCI_A1 del microprocesador como puertos serie. En ella, se configuran los pines de transmisión y recepción de cada uno, y se configuran los registros.

Los registros principales son los que se encargan de seleccionar la fuente de reloj de la transmisión serie. Como se había comentado anteriormente, esta fuente de reloj será la SMCLK, la cual está a 4.5MHz.

El otro registro importante, será el que determine la velocidad de envío. Esta velocidad será de 115200 bits por segundo. El registro se ha de configurar poniendo el resultado de la división de la velocidad de la fuente de reloj entre la velocidad de transmisión.

También se habilitará la interrupción de dato recibido para el puerto USCI_A0, que será el que esté conectado al ordenador. La funcionalidad de esta interrupción se explicará más adelante.
void init_3_SPI_asd1246(void);

Esta función será la encargada de configurar todos los pines de los puertos SPI y los puertos de entrada-salida, conectados a los convertidores analógico-digitales. También configurará la comunicación SPI de los puertos UCB1, UCB2 y UCB3.

La configuración de los pines se hará puerto a puerto, mirando para cada puerto la cantidad de conexiones que tendrá y configurándolo según el propósito de cada pin. Se reservará el puerto 8 para la conexión de todos los pines de START y RESET de los ADS1246. De esta manera, cuando haya que activarlos, desactivarlos o restearlos, atacando el puerto se estará actuando sobre los tres convertidores. Esto, unido al hecho de que ambos tres convertidores están conectados a la misma fuente de reloj, asegurará la sincronización de los tres convertidores (cuando se utilicen tres canales).

La configuración del SPI se realizará configurando los registros de los puertos UCB1, UCB2 y UCB3 mencionados anteriormente. Principalmente, se configurará el microprocesador como maestro, que los datos sean de 8 bits, y que el primer bit en llegar sea el de mayor peso.

Por otro lado se configurará la fuente de reloj para la transmisión síncrona, y la velocidad de envío de los datos. Como se había comentado, la fuente será la SMCLK. La velocidad máxima de transmisión del SPI es de 2MHz, por lo tanto, se configurará un registro para dividir la fuente de reloj por cuatro, lo que proporcionará una velocidad algo superior a 1MHz, la cual es más que suficiente para el correcto funcionamiento de la aplicación.

void buf_setup(void);

Esta función será la encargada de configurar la cantidad y el tamaño de los buffers. Observando la configuración introducida por el usuario en el menú principal, decidirá si se configuran dos, cuatro o seis buffers (dos por canal), y su respectiva longitud (cincuenta posiciones o cien, según el número de muestras por segundo introducido).

A su vez, también será la encargada de posicionar correctamente los punteros de los buffers para su correcta lectura y escritura. Habrá cuatro punteros por buffer, dos constantes y dos dinámicos. Los constantes, apuntarán uno a la primera posición del buffer, y el otro a la última. Se utilizarán para marcar los límites a los punteros dinámicos. En cuanto a éstos últimos, uno se utilizará para recorrer el buffer escribiéndole datos, y el otro para ir leyendo los datos.

void ADS_Power_Up(void);

Los convertidores analógicos-digitales requieren una serie de instrucciones para su encendido. Esta función será la encargada de ejecutar esta serie de instrucciones para cada uno de los convertidores, dependiendo del número de canales a utilizar. En síntesis, se activarán los convertidores y luego se realizará un reseteo de los registros, mediante la modificación de puertos de entrada-salida. Finalmente se activará la comunicación SPI entre el microprocesador y los convertidores a utilizar.

void registers(void);

Los convertidores ADS1246 precisan de la configuración de sus registros para su funcionamiento. Esta configuración se realizará mediante comandos enviados por SPI. Esta función será la encargada de configurar los registros que se pueden ver la tabla 4-3, según el tipo de adquisición que desee el usuario.
Esto se realizará para unos, dos o tres convertidores, según la cantidad de canales a utilizar.

El método a seguir para la configuración de registros mediante comandos SPI es el siguiente. En primer lugar, se envía un comando que informa al convertidor que será configurado, junto con la dirección del primer registro a configurar. En segundo lugar, se envía la cantidad de registros a configurar partir del introducido menos uno, por ejemplo, si se modificaran todos, en este caso se enviaría un diez. Por último ya se procedería a enviar, registro por registro, su valor.

Principalmente, son dos los registros que tienen mayor relevancia en la configuración, el registro MUX y el SYS0, los cuales se pueden observar en las figuras 4-11 y 4-12.

En este registro, se configurará principalmente la utilización del reloj externo. Este reloj externo, como se ha comentado anteriormente, será proporcionado por el microprocesador, y será la fuente SMCLK, de 4.5MHz.
El registro SYS0 servirá para configurar dos aspectos muy importantes en un convertidor, la ganancia y las muestras por segundo. En este caso, la ganancia no tiene mayor relevancia, ya que se preestablecerá a 1, pero la cantidad de muestras por segundo sí que será importante.

- void initRTC(int year, char month, char day, char day_week, char hour, char minute, char second);

Como se ha comentado anteriormente, será imprescindible el uso de un reloj en tiempo real. Esta función tendrá el objetivo de configurar los registros de tiempo de la RTC, con los valores de la fecha y hora actual que configure el usuario.

También será la encargada de modificar el registro que configura la RTC en modo calendario.

- void StartRTC(void);

El objetivo de esta función es la puesta en marcha de la RTC. Simplemente se limitará al cambio de un bit en un registro para llevar su función a cabo.

- void ADS1246_1_AssertCS(int fAssert);

Anteriormente, se comentó que una de las líneas SPI importantes era la de chip select (CS). Ésta será la que habilite la comunicación entre el dispositivo maestro, en este caso el microprocesador, y el esclavo, el convertidor. Esta línea se activa a nivel bajo, es decir, poniendo el pin de salida del microprocesador conectado al CS a valor lógico 0.

El objetivo de esta función es activar la comunicación introduciendo un uno en la variable entera fAssert. Cuando se introduzca el uno, se pondrá un cero en el pin correspondiente, lo que habilitará la comunicación.
De manera contraria, introduciendo un cero en la variable fAssert, se deshabilitará la comunicación, poniendo a nivel alto el pin de CS.

Esta función realizará esta acción para el convertidor 1.

- void ADS1246_2_AssertCS(int fAssert);

Esta función funciona exactamente igual que la anterior, pero realizará su acción para el convertidor número 2.

- void ADS1246_3_AssertCS(int fAssert);

Esta función también funciona exactamente igual que la anterior, pero realizará su acción para el convertidor número 3.

4.4.6 Función principal y funciones propias del sistema

La función principal será la encargada de gestionar toda la aplicación, decidiendo el orden de ejecución de las de las demás. Además también se comentarán algunas funciones propias del sistema.

- void main(void)

La función main es la función principal del sistema. Ella será la encargada de ir llamando a las funciones anteriores para que se ejecuten y configuren el dispositivo.

Una vez configurado, cuando comienza la adquisición de los datos, la función principal será la encargada de gestionar la aplicación. Cuando haya buffers de datos llenos, será la encargada de dar la orden de enviarlos.

A su vez, cuando sea necesario, se encargará de ir actualizando la cabecera de datos del puerto serie, de crear nuevas carpetas o ficheros para los archivos de datos, y cerrarlos cuando el usuario decida detener la aplicación.

- void __delay_cycles(unsigned long n);

Ésta función, propia del sistema, se encarga de esperar la cantidad de ciclos de reloj que le sean introducidos en la variable n.

- unsigned char __bis_SR_register(unsigned char mask);

La siguiente función, también propia del sistema, es la encargada de modificar el Status Register (SR) del microprocesador. El SR es un registro de 16 bits que controla algunos funcionamientos básicos del microprocesador. En la siguiente figura se puede observar el registro, junto a la descripción de sus bits modificables.
Esta función se utilizará principalmente para la activación o desactivación de las interrupciones en la aplicación.

- `time_t mktime(struct tm *timeptr);`

Esta función, propia de la librería de tiempo del sistema, tiene la función de devolver una variable entera, que contenga el valor de tiempo en formato Epoch. Este formato de tiempo dice la cantidad de segundos transcurridos desde el año 1970.

Se introducirá una estructura, en la cual se tendrá que especificar los datos del año actual, el mes, el día del mes y el de la semana, la hora, el minuto y el segundo.

Se utilizará para el dato del tiempo en la cabecera del envío por puerto RS232.

### 4.4.7 Funciones e interrupciones de recepción, compresión y envío de datos

Las funciones e interrupciones que se describirán a continuación, son las encargadas de interactuar con el usuario y con los datos.

- `void enviarRS232(unsigned char Code, char canal);`

Para realizar el envío de un byte por puerto RS232 se utilizará esta función.

En primer lugar, se le pasará como parámetro, el byte a enviar. Luego habrá que especificar por cuál de los dos canales disponibles se enviará, dependiendo si es un dato de la adquisición, o si es información a transmitir al usuario mediante texto.
void enviar_texto(char dades[], char canal);

Cuando se quiera enviar texto, principalmente en el menú de configuración, habrá que utilizar esta función. Esta, a su vez, llamará a la función anterior para hacer el envío.

Habrá que pasarle como parámetros, en primer lugar, un vector con el texto a enviar. Cada posición del vector corresponderá a una letra diferente. Luego, igual que en la función anterior, el canal por el cual se enviará.

void buf_vacio(void);

Esta función será utilizada al principio y al final de la aplicación. Su objetivo es evitar que el primer y el segundo archivo de datos generado en la memoria SD tengan menos datos que los demás archivos.

Esto es posible ya que normalmente, los archivos tienen datos equivalentes a una hora de adquisición, a excepción del primero y el último. Esto es debido a que al existir la posibilidad de arrancar y/o finalizar el programa en cualquier instante, es más que probable que el archivo inicial no arranque a una hora determinada con cero minutos, ni que acabe justo cuando se haya cerrado el archivo por haber cambiado de hora.

Por lo tanto esta función en primer lugar mirará si entra en ella por ser el primer o el último archivo. Luego, dependiendo del minuto actual, pondrá tantos ceros como sean necesarios al principio del archivo (si se ha llamado por ser el primero), o lo completará de ceros (si es el último archivo).

void escribe_buffer1(unsigned char a, unsigned char b, unsigned char c);

Cuando se utilice solo un canal para la adquisición, esta función será la encargada de guardar el dato obtenido del convertidor analógico-digital en el buffer del canal uno correspondiente.

Esta función estará activa únicamente si se utiliza un solo canal. Los tres parámetros que se le pasan a la función, corresponden a los tres bytes del dato obtenido. Por lo tanto cada vez que se lea un dato, se entrará en esta función.

void escribe_buffer2(unsigned char a, unsigned char b, unsigned char c, unsigned char d, unsigned char e, unsigned char f);

Al igual que la función anterior, esta función se encargará de guardar los datos leídos en los buffers del canal uno y dos correspondientes, cuando se esté llevando a cabo una adquisición con dos canales.

También, igual que anteriormente, en el caso de no utilizar dos canales, la función estará inactiva. Los parámetros a pasarle, corresponden a los seis bytes de los dos datos de información, el del canal uno, y el del canal dos.

void escribe_buffer3(unsigned char a, unsigned char b, unsigned char c, unsigned char d, unsigned char e, unsigned char f, unsigned char g, unsigned char h, unsigned char i);

Por último, cuando se esté llevando a cabo una adquisición con tres canales, esta función se encargará de guardar los datos leídos en los buffers del canal uno, dos y tres correspondientes.

También, igual que anteriormente, en el caso de no utilizar tres canales, la función estará inactiva. Los parámetros a pasarle, corresponden a los nueve bytes de los tres datos de información, el del canal uno, dos y el tres.
- void lee_buffer(void);

Una vez algunos de los buffers estén llenos, se ejecutará esta función. Su objetivo será leer estos datos adquiridos anteriormente.

Esta lectura se hará de dos maneras. Por un lado, se guardarán los datos en los distintos ficheros de la tarjeta SD, siempre y cuando esté configurada para utilizarse. Por otro lado, se enviarán mediante puerto serie, empezando por la cabecera, y luego los datos del canal uno, los del dos y los del tres.

- __interruptvoid Port_1(void)

Esta interrupción tendrá un funcionamiento muy importante en la aplicación.

Se configurará que cada vez que la señal conectada al puerto 1.5, corresponedente al DRDY del convertidor analógico-digital del canal 1, pase de estado alto a bajo, entre en la interrupción. Esto significará que se ha convertido un dato y está listo para ser leído.

Por lo tanto, dentro de la rutina de la interrupción, se configurará la lectura de este o estos datos, según el número de canales a utilizar.

En primer lugar se enviarán los correspondientes comandos SPI para la lectura de los tres bytes de cada dato y luego, en segundo lugar, se ejecutarán las funciones explicadas anteriormente de escribe_bufferX. Una vez acabado este proceso, se saldrá de la interrupción.

- __interrupt void TIMER1_A0_ISR(void)

El temporizador uno se encargará simplemente de aumentar el valor de una variable cada vez que se entre en él. Sus tiempos de ejecución dependerán de la zona donde esté el programa.

Para la programación de los temporizadores, se configurarán dos registros. En el primero, se especificará la fuente de reloj sobre la que haga la cuenta, y la manera de ejecutarse. En el caso de esta aplicación, cada vez que el temporizador cuente la cantidad de ciclos de reloj, especificados en el segundo registro a configurar, entraría en la interrupción y volverá a contar desde cero.

Por lo tanto esta interrupción se utilizará para llevar cuentas, sobretodo en el menú de configuración, que se explicará más adelante.

- __interrupt void TIMER0_A0_ISR(void)

El temporizador cero tendrá un funcionamiento importante. Será el encargado de mandar los pulsos a los pines de START de los convertidores, para informarles de que comiencen con la conversión de otro dato.

Dependiendo de la cantidad de muestras por segundo configuradas, este temporizador saltará cien o cincuenta veces por segundo.

- __interrupt void USCI_A0_ISR(void)

Esta interrupción corresponde al puerto USCI_A0, configurado como uno de los dos puertos RS232.
Se ejecutará una vez se reciba un dato desde el ordenador y, si no es el símbolo de fin de programa, se limitará a guardar el dato recibido en una variable. En cambio, si símbolo recibido corresponde al de fin de programa, se procederá a el cierre de la aplicación, desactivando la recopilación de datos.

### 4.4.8 Funciones de almacenamiento de los datos en memoria externa

Para el volcado de la información adquirida a la memoria externa de tipo SD, se utilizará un sistema de tabla de asignación de archivos, comúnmente conocido como FAT, una arquitectura de sistema de archivos de ordenador.

El sistema de archivos FAT es un sistema de archivos heredado que es simple y robusto. Ofrece un buen rendimiento incluso en las implementaciones de poco peso, pero no puede ofrecer el mismo rendimiento, fiabilidad y escalabilidad, ya que algunos sistemas de archivos modernos. Sin embargo, se admite por razones de compatibilidad con prácticamente todos los sistemas operativos existentes para los ordenadores personales, y por lo tanto es un formato muy adecuado para el intercambio de datos entre ordenadores y dispositivos de casi cualquier tipo [8]. Para la implementación de las funciones FAT, se utilizará un código de libre acceso, llamado FATFS.

![Figura 4-14. Esquema de organización de los niveles en las memorias](image)

FATFS es un módulo de sistema de archivos FAT genérico para pequeños sistemas embebidos. El FATFS está escrito en cumplimiento con la norma ANSI C y completamente separados de la capa de entradas-salida de la memoria. Por lo tanto, es independiente de la arquitectura de hardware. Puede ser incorporado en los microprocesadores de bajo costo sin ningún cambio [9]. En la figura 4-14 se puede observar un pequeño esquema de la estructura de los niveles dentro de la memoria externa.

A continuación, se explicarán las funciones utilizadas del sistema FATFS en la aplicación.

- **DSTATUS disk_initialize (BYTE pdrv);**

Esta función inicializa una la unidad física poniéndola lista para su lectura y/o escritura de datos genéricos.

Devolverá un cero en caso de que se execute con éxito y un número con el valor correspondiente al número de error., en caso de que no lo haga.

- **FRESULT f_mount (BYTE vol, FATFS*fs);**
Esta función se utilizará al principio y al final de la aplicación para montar, al principio, y desmontar, al final, el sistema de ficheros de la memoria externa.

La función devolverá un OK, si se ha ejecutado con éxito, o un INVALID_DRIVE, si el número de discos especificados no corresponde con el de volúmenes.

Los parámetros a pasarle serán, por un lado el número de dispositivo físico a acceder (por si hubiera más de uno), y por otro lado, el puntero a la estructura de datos del sistema de ficheros.

- FRESULT f_open (FIL*fp, const TCHAR*path, BYTE mode);

Esta función se utilizará para la creación de los archivos de datos.

En primer lugar, se especificará la estructura de archivo a crear, luego la ruda y el nombre del archivo, y por último se le comunicará que serán archivos de escritura de datos.

Se comprobará que la función devuelve un OK, correspondiente a que el archivo se ha creado con éxito. De otro modo, se informará al usuario con el valor correspondiente al número de error.

- FRESULT f_close (FIL*fp);

Al contrario que la función anterior, esta cierra el fichero de datos al cual se estaba accediendo. Cerrar el fichero es fundamental para que no se pierdan los datos escritos.

Igual que en el caso anterior, se comprobará que la función devuelve un OK, correspondiente a que el archivo se ha creado con éxito. De otro modo, se informará al usuario con el valor correspondiente al número de error.

En este caso solo habrá que pasarle el puntero de la estructura de ficheros que se quiere cerrar. Esta estructura ha de estar creada anteriormente por la función f_open.

- FRESULT f_write(FIL*fp, const void*buff, UINT btw, UINT*bw);

Esta función se utilizará para poder escribir los datos dentro del fichero abierto. En caso de que se ejecute correctamente, devolverá un cero, y en caso de que no lo haga, devolverá un valor correspondiente al número de error.

Los parámetros a pasarle serán, por un lado el puntero de la estructura de datos donde se escribirán, luego el puntero de la variable que contiene los datos a escribir (en el caso de la aplicación, cada uno de los buffers de datos generados). Luego se le informará de la cantidad de bytes a escribir. En el caso de que se utilicen cien muestras por segundo, este valor será de trescientos, ya que se escribirá cada segundo, y habrá cien datos de tres bytes cada uno. Por último se enviará el puntero de la variable que indicará el número de bytes escritos.

- FRESULT f_mkdir (const TCHAR*path);

Esta función se utilizará para generar una carpeta dentro del fichero de datos. Su funcionamiento será muy sencillo, simplemente se le pasará el puntero que contenga el nombre del nuevo directorio, y la función se encargará de generarla.

Devolverá un cero en caso de haberse ejecutado correctamente, y en caso de no hacerlo devolverá un valor correspondiente al número de error.
• **DWORD get_fattime**(void);

Esta función es necesaria para que el sistema pueda obtener la hora del sistema y poder utilizarla en el protocolo de FAT. Esta función no requiere ningún parámetro de entrada debido a que lee la información directamente de la RTC del sistema.

Devuelve una trama de 32 bits, con la información completa de la fecha y hora actual, la cual se puede observar en la tabla 4-4.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Información</th>
<th>Rango de valores</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-4</td>
<td>Segundo</td>
<td>0-29</td>
</tr>
<tr>
<td>5-10</td>
<td>Minuto</td>
<td>0-59</td>
</tr>
<tr>
<td>11-15</td>
<td>Hora</td>
<td>0-23</td>
</tr>
<tr>
<td>16-20</td>
<td>Día</td>
<td>1-31</td>
</tr>
<tr>
<td>21-24</td>
<td>Mes</td>
<td>1-12</td>
</tr>
<tr>
<td>25-32</td>
<td>Año</td>
<td>0-127 (se cuenta desde 1980)</td>
</tr>
</tbody>
</table>

Tabla 4-4. Descripción de la trama de tiempo de la función get_fattime

### 4.4.9 Menú de configuración

En este apartado se podrán observar las funciones encargadas de comunicarse directamente con el usuario a través de los menús de configuración. Principalmente se configurará, como se puedo observar en la figura 4-6, las muestras por segundo de los convertidores analógicos-digitales, el número de canales a utilizar, la forma de envío y de almacenamiento de los datos o arrancar con una configuración por defecto.

• **void menu**(void);

En este primer menú de configuración el usuario deberá introducir los datos de la fecha y la hora actuales para la configuración de la RTC.

El menú enviará por puerto serie la información solicitada y recibirá, también por puerto serie, la información introducida por el usuario.

Se pedirá la siguiente información:

- **“Enter current date (ddmmyyyy)”**

El usuario deberá introducir el día, el mes y el año en el formato pedido.

- **“Enter current hour and minute (hhmm)”**

El usuario deberá introducir la hora y el minuto actual en el formato pedido.

- **“Enter current day (1 for Monday, 2 for Tuesday, ...)”**

El usuario deberá introducir el día de la semana correspondiente, siendo el 1 el lunes, 2 el martes, 3 el miércoles, 4 el jueves, 5 el viernes, 6 el sábado y 7 el domingo.

Una vez se introduzcan los datos se comprobará que sean correctos. Si no lo son, se volverán a pedir hasta que lo sean. Cuando lo sean, se arrancará la RTC y se irá al menú principal, el cual se explicará en la siguiente función.
• void main_menu(void);

El menú principal se encargará de pedir al usuario que configure las diferentes partes de los periféricos o de la aplicación. La información que enviará será la siguiente:

– “A/D CONFIG”

Se entrará en la función config_placa_adc para la configuración de los diferentes convertidores analógicos-digitales.

– “STORAGE AND COMMUNICATION CONFIG”

Se entrará en el menú de config_SD_SERIE, donde se configurará el uso del envío de datos por puerto serie y de la tarjeta de memoria externa.

– “DEFAULT CONFIG”

Aquí se determinará una configuración por defecto, la cual tendrá las siguientes características:

  o 3 canales de adquisición
  o 100 muestras por segundo
  o Envío por puerto serie activo
  o Almacenamiento en memoria externa SD activo

Si el usuario no seleccionará ninguna opción, cuando haya transcurrido un minuto, el programa arrancará con la configuración por defecto recién descrita.

– “START ACQUISITION”

Si se ha introducido una configuración válida, comenzará a ejecutar la aplicación. Si la configuración no es válida, volverá a ejecutar el menú principal.

• void config_placa_adc(void);

Este menú se encargará de la configuración de los convertidores analógicos-digitales. El menú desplegará las siguientes opciones:

– “Enter sampling rate”

En este apartado se deberá escoger entre hacer una adquisición a cincuenta o a cien muestras por segundo.

– “Enter board active channels”

En este apartado se deberá escoger el número de canales a utilizar en la adquisición de los datos. Como ya se ha comentado, las posibilidades serán de uno, dos o tres canales.

• void config_SD_SERIE(void);

Finalmente en este menú, se deberá seleccionar de qué manera se gestionarán los datos obtenidos. El menú desplegará la siguiente información, de la cual se tendrá que seleccionar una de ellas:
- “SD Storage and RS232 Communication”

Seleccionando esta opción, el usuario activará tanto el envío de datos por puerto RS232, como el almacenamiento de los datos en la memoria externa SD.

- “Only SD Storage”

Seleccionando esta opción, el usuario desactivará el envío de datos por puerto RS232, y activará el almacenamiento de los datos en la memoria externa SD.

- “Only RS232 Communication”

Seleccionando esta opción, el usuario activará el envío de datos por puerto RS232, y desactivará el almacenamiento de los datos en la memoria externa SD.
5 Caracterización del sistema
A continuación se expondrá la metodología aplicada para el testeo de la aplicación finalizada, los diferentes instrumentos utilizados, los pasos para el entendimiento del funcionamiento de las diferentes partes del microprocesador y finalmente los resultados obtenidos.

5.1 Dispositivos de laboratorio utilizados
Para realizar el trabajo hecho en el laboratorio, se han utilizado una serie de dispositivos encargados de realizar diferentes funciones. A continuación se detallará uno por uno cada uno de estos dispositivos.

   – Fuente de alimentación: PROMAX FAC-662B

La fuente de alimentación es la encargada de dar la alimentación digital y analógica al sistema. Dispone de dos salidas independientes. Para las diferentes pruebas, se utiliza una salida a 5V como alimentación analógica, y la otra a 3.3V como alimentación digital.

   – Generador de funciones: HP 33120A

Este instrumento será el encargado de simular la señal analógica a adquirir por el sistema de adquisición. Se conectará directamente en lugar del sensor.

Se configurará según la necesidad de la prueba a realizar, pero siempre utilizando valores similares al tipo de señal que se querrá adquirir (0V-1V de amplitud, 0.5V de offset y señales con frecuencias entre 1Hz-10Hz).

   – Osciloscopio: PROMAX ON-590

El osciloscopio se utilizará para hacer diferentes pruebas de testeo de diferentes señales.
Figura 5-1. Esquema de la conexión de los instrumentos con las diferentes partes del sistema de adquisición

En la figura 5-1, se puede observar un pequeño esquema de la conexión de los diferentes instrumentos utilizados con el sistema de adquisición.

5.2 Pruebas realizadas

Antes de la realización de la programación de la aplicación, para el aprendizaje del funcionamiento del microprocesador, se han realizado diferentes pequeñas aplicaciones que luego, sean extrapolables a la aplicación principal. Estas se explicarán a continuación:

- Configuración de los puertos de entrada-salida del sistema

Para la comunicación con los convertidores analógicos-digitales, será importante conocer el funcionamiento de los puertos de entrada-salida del microprocesador. Los puertos suelen tener diferentes funciones, como se puede observar en la tabla 5-1.
En este caso, se observa cómo cada uno de los pines del puerto uno puede tener diferentes funciones. Para la configuración como puerto de entrada-salida, se deberá especificar en el registro PXDIR, el valor cero si se quiere que sea pin de entrada, o el valor uno si se quiere que sea de salida. Luego se tendrá que especificar el registro PXSEL. Para que sea de entrada-salida, deberá tener un valor de cero.

Esta aplicación consiste en la activación de alguno de los pines del puerto como salida, poniéndolo periódicamente su valor a nivel alto y bajo, y otros pines como entrada, aplicándoles una tensión externa. Se comprobará su funcionamiento de los pines de salida observando la el pin de salida del microprocesador con el osciloscopio. En cuanto a los pines de entrada, se observará en la aplicación como su valor ha cambiado.

- Configuración de los relojes del sistema

Se realizará la programación de los relojes, ya que será necesaria en todos los diferentes programas. Esta aplicación, simplemente consistirá en hacer la configuración de los registros necesarios, para configurar los relojes como se explicó en el apartado 3.2.

Para la comprobación de la correcta configuración de las tres señales de reloj, se utilizarán tres pines como salidas de las señales de reloj ACLK, SMCLK y MCLK. Se observará las diferentes señales en el osciloscopio y se comprobará su valor.

- Configuración y envío de datos por puerto serie

En esta aplicación, se configurará alguno de los puertos que pueda actuar como puerto serie RS232. Para ello, solo se deberá configurar el puerto como especifica la hoja de características, igual que se ha hecho anteriormente, y los registros propios del puerto serie. Para probar la aplicación se utilizará el HiperTerminal de Windows, tanto para enviar datos, como para observar los recibidos.
La aplicación consistirá, en primer lugar, la configuración del puerto y los registros. En los registros, se especificará la velocidad y la característica de la trama a enviar.

En segundo lugar, la aplicación esperará a que el buffer de datos recibidos reciba un cierto carácter del ordenador. Una vez sea recibido este carácter, enviará un valor específico al ordenador. Este valor se podrá visualizar mediante el HiperTerminal.

- Puerto SPI

Para la realización de esta prueba, se conecta a uno de los puertos SPI del microprocesador, uno de los convertidores. Esto será necesario para que actúe como esclavo en la comunicación. Por lo tanto se seguirá la estructura que se observa en el figura 5-2.

![Figura 5-2. Esquema de la aplicación](image)

Como se puede observar, se convertirá una señal sinusoidal proveniente de un generador de funciones, luego se enviará por puerto SPI al microprocesador, el cual a su vez, mediante puerto serie, enviará los datos a una aplicación hecha en LabView, para la visualización de la señal sinusoidal.

El procedimiento será muy similar al de la aplicación anterior. En primer lugar se configurará uno de los puertos y los registros SPI, especificando la velocidad del reloj de la comunicación y el formato de la trama. También se configurará uno de los puertos serie, como se había hecho anteriormente.

Luego, siguiendo las especificaciones de la hoja de características del convertidor analógico-digital, mediante los comandos SPI especificados, se configurará el dispositivo, y se pedirá que convierta datos periódicamente. A medida que se vayan convirtiendo los datos, el microprocesador los leerá del buffer de recepción del puerto SPI, y los enviará al ordenador mediante puerto serie. La aplicación de LabView se encargará de representar estos datos en un gráfico.

En la figura 5-3 se puede observar el panel frontal de la aplicación de LabView diseñada. En ella se puede ver la representación de la onda sinusoidal obtenida a partir de la adquisición hecha por el ADS1246.
Figura 5-3. Panel frontal de LabView con la representación de los datos de un canal

- Real Time Clock (RTC)

Para la configuración de la RTC, será necesario modificar sus registros como indique la hora de características del microprocesador.

Como se explicó anteriormente, se configurará en modo calendario, y los la información se escribirá y se recibirá en formato BCD. Una vez configurados estos registros, se tendrán que introducir los valores de fecha y hora actuales. Esto se realizará desde la propia aplicación.

Luego para poder comprobar el correcto funcionamiento, se enviarán todos los valores de los registros de tiempo por puerto seria al HiperTerminal de Windows, cada un segundo. De esta manera se debería observar cómo van cambiando los valores a medida que se van enviando.

- Interrupciones

El correcto funcionamiento de las interrupciones será fundamental para el funcionamiento de la aplicación.

Se utilizarán un total de cuatro interrupciones, la del puerto 1, la de recepción de uno de los puertos serie y dos temporizadores. Para poder testear su funcionamiento se realizará un pequeño programa que las configure y, una vez vaya entrando en ellas, se envíe un símbolo al ordenador para saber que ha entrado.

En primer lugar, como siempre, se hacen las respectivas configuraciones. En primer lugar se configuran los registros del puerto 1 para que la interrupción salte en el momento que en el pin 1.5 haya un flanco de bajada. Luego se configuran los temporizadores, uno de ellos para que se ejecute cada treinta segundos y el otro para que cada cuarenta y cinco segundos se ejecute. Por último, la interrupción del puerto serie deberá saltar cada vez que el usuario tecleó algo.

Cada vez que salten estas interrupciones se enviará por puerto serie al HiperTerminal un cierto carácter, dependiendo de cuál de ellas se haya ejecutado. Por lo tanto la aplicación consistirá en la configuración, la activación de las interrupciones, y la espera de sus ejecuciones. Mientras tanto, al saber cuándo se han de ir
executando, mediante la observación del HiperTerminal, se puede ir observando si el funcionamiento es correcto o no.

Tanto en un temporizador como en otro, se puede saber cuándo se ha de ejecutar llevando un simple control del tiempo. Luego, se sabrá que la interrupción del puerto serie se deberá ejecutar cada vez que se presione una tecla. Finalmente, para poder comprobar la interrupción del puerto 1, se deberá conectar una señal a nivel alto al pin 1.5 y luego, bajar ese nivel de tensión, para provocar un flanco de bajada.

- Buffers de datos

De cara a la prueba del funcionamiento del buffer de datos, se generó una pequeña aplicación encargada de generar dos buffers y hacer el recorrido de estos.

En primer lugar, se generaban estáticamente los buffers, a través de la estructura descrita en el punto 4.4.2 de esta memoria. Esta generación de buffers consistía, básicamente, en la creación de un vector de estructuras, dónde a su vez cada estructura estuviera formada por un vector de tres posiciones. Cada una de estas estructuras guardaría un dato de la adquisición.

La aplicación generada para el testeo, consiste en la generación de dos buffers. Una vez generados, se irá llenando uno de ellos, mientras el contenido del otro se envía por puerto serie. Esto se realizará en un bucle continuo, para poder observar si en algún momento se produce el posible conflicto de que un buffer se tenga que enviar antes de que esté escrito.

Finalmente adaptando velocidades de envío y optimizando el código, se consigue realizar con éxito la tarea encomendada.

- Memoria externa SD

Para realizar la prueba del funcionamiento de la memoria externa SD, se siguen los diferentes ejemplos proporcionados por los programadores de código libre del sistema FatFs.

A partir de estos ejemplos, se realizan aplicaciones que se centran en la generación de directorios, dónde estén contenidos los ficheros. Una vez generados, la aplicación se encargará de guardar datos conocidos, anteriormente guardados en los buffers de datos. Una vez se crea que la cantidad de datos sea suficiente, mediante una sencilla aplicación de MatLab, se abrirá el fichero y se comprobará que los datos guardados corresponden a lo que deberían ser.

- Aplicación final

A raíz de todas las pruebas realizadas, se procede a la unificación de las diferentes aplicaciones. Esta unificación se realiza de manera escalada, testando el funcionamiento de las diferentes partes acopladas.

Una vez que se van consiguiendo los objetivos planteados en cuanto al funcionamiento, se consigue realizar la aplicación descrita en el punto 4 de esta memoria.

5.3 Consumo del sistema

Para realizar la evaluación del consumo del sistema, se estudiará cada una de las partes del sistema de adquisición, evaluando la potencia consumida por cada uno de ellos.
En primer lugar se calculará el valor teórico del consumo. Luego mediante los instrumentos, se calculará el valor práctico a diferentes modos de funcionamiento.

### 5.3.1 Consumo teórico

En la tabla 5-2, se puede observar cada uno de los componentes del dispositivo y su consumo teórico. Como se puede apreciar, el consumo total sería de unos 58.6mW, bastante menor al de otros dispositivos de adquisición de este tipo.

<table>
<thead>
<tr>
<th>Fabricante</th>
<th>Modelo</th>
<th>Unidades</th>
<th>Consumo unitario [mW]</th>
<th>Consumo [mW]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microprocesador</strong></td>
<td><strong>Texas Instruments</strong></td>
<td>MSP430F5438</td>
<td>1</td>
<td>26.7</td>
</tr>
<tr>
<td><strong>Convertidor A/D</strong></td>
<td><strong>Texas Instruments</strong></td>
<td>ADS1246</td>
<td>3</td>
<td>2.3</td>
</tr>
<tr>
<td><strong>Comunicación RF</strong></td>
<td><strong>LYNX</strong></td>
<td>YXL-TRM8053-025-05</td>
<td>1</td>
<td>198</td>
</tr>
<tr>
<td><strong>Convertidor TTL/RS232</strong></td>
<td><strong>MAXIM</strong></td>
<td>MAX232CPE</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td><strong>Sensor</strong></td>
<td><strong>Input/Output</strong></td>
<td>SM6</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

**Tabla 5-2. Consumos de los diferentes dispositivos del sistema de adquisición de datos**

### 5.3.2 Consumo real

Para el testeo del consumo real, no se utilizará la radiofrecuencia. Por otro lado, solo se analizará el consumo de los dispositivos alimentados a 3.3V, por lo tanto el convertidor TTL/RS232 no se tendrá en cuenta. Esto es debido a que este convertidor será cambiado por otro de 3.3V en un futuro diseño.

Por lo tanto se comparará con un consumo teórico de 33.6mW, referente al microprocesador y a los convertidores. Se realizará una primera prueba a cincuenta muestras por segundo y, luego, una segunda a cien muestras por segundo. Esto simplemente se realizará para poder observar si existe diferencia teniendo en cuenta las muestras por segundo.

Para el estudio de este consumo, se conectará un multímetro AGILENT 34401 en serie con la alimentación de 3.3V.

- **50 muestras por segundo**

A cincuenta muestras por segundo el consumo de corriente indicado por la fuente de alimentación será de 2.4mA, como se puede apreciar en la figura 5-4.

![Figura 5-4. Corriente consumida a 50 muestras por segundo](image-url)
Por lo tanto el consumo a 50 muestras por segundo será de 7.92mW (2.4mA · 3.3V).

La diferencia con respecto al consumo teórico será de 25.68mW, muy por debajo de lo esperado según los consumos teóricos. Sin embargo hay que tener en cuenta la aparición de picos instantáneos (1 por segundo) de 40mA al hacer la escritura en la SD. De todas maneras, el resultado es muy positivo.

- **100 muestras por segundo**

A cien muestras por segundo el consumo de corriente indicado por la fuente de alimentación será de 2.6mA, como se puede apreciar en la figura 5-5.

![Figura 5-5. Corriente consumida a 100 muestras por segundo](image)

Por lo tanto el consumo a 100 muestras por segundo será de: 8.58mW.

La diferencia con respecto al consumo teórico será de 25.02mW.

Cómo se puede observar, existe una diferencia de 0.66mW entre el consumo a 50 muestras por segundo, respecto con el del 100, por lo tanto se puede afirmar que no resultará problemático el hecho de utilizar una velocidad de adquisición u otra.

**5.4 Resultados**

Para la comprobación del funcionamiento del sistema, se utilizará, por un lado, una aplicación de LabView, para la comprobación del envío de los datos por puerto serie. Por otro lado, para poder verificar el almacenamiento de los datos en la memoria externa SD, se utilizará un pequeño código de MatLab.

Ambos casos, la visualización y el almacenamiento, se realizarán en el modo más crítico de trabajo, el cual será cuando la adquisición se realice con tres canales. Ésta manera será la considerada crítica, al estar mayor tiempo enviando y guardando los datos.

**5.4.1 Envío y visualización de los datos**

Para realizar la visualización de los datos, se utilizará una aplicación de LabView. Esta aplicación se encargará de conectarse al puerto serie que se encargue de hacer este envío.

La aplicación se ejecutará paralelamente a la visualización del menú de configuración, ya que a diferencia del almacenamiento externo, el envío de datos se visualizará a tiempo real.
En la figura 5-6 se puede observar el diagrama de bloques de la aplicación. En él, en primer lugar se observa la configuración de la comunicación del puerto serie. Luego, se leerán los 18 bytes correspondientes a la cabecera de datos, los cuales se guardarán en un vector que se irá representando periódicamente.

A continuación comenzará la lectura de cada canal. En primer lugar se obtendrán los datos del canal uno, leyendo byte a byte, e invirtiendo su orden antes de ser representados, ya que la aplicación enviará en primer lugar el byte de menor peso. Los datos llegarán con el formato de String, el cual se convertirá a datos de 4 bytes, para poder representarlos visualmente.

Una vez se hayan leído los cincuenta o cien bytes (dependiendo de la cantidad de datos a adquirir por segundo) del canal uno, se repetirá el mismo proceso para los canales dos y tres. La aplicación acaba cuando el usuario lo indique.

- **50 muestras por segundo**

La señal que se adquirirá tendrá las siguientes características: 5Hz, 1Vpp y 0.5V de offset.

En la figura 5-7 se puede observar una captura de pantalla de un instante de la adquisición de datos. Es apreciable como en 50 puntos representados se observan 5 ciclos de la señal adquirida, pudiendo representar con una precisión de 10 muestras por periodo la señal.
Como se puede observar, la adquisición de los tres canales es correcta, al tener representada la misma figura, la cual corresponde a la misma señal sinusoidal que adquieren, proveniente del generador de funciones.
En la figura 5-8 se puede observar la cabecera de datos obtenida en la adquisición, a la izquierda, en formato ASCII, y a la derecha, su valor hexadecimal. Alguno de los valores ASCII no tienen codificación, estos son aquellos que siguen la estructura “.XX”.

Si se observan los datos de la cabecera, se puede ver como los primeros cuatro bytes corresponden al identificador $DAT. Luego, los siguientes cuatro, al formato de tiempo Epoch [10]. Esta prueba se realizó un 17 de junio a las 10:30 horas de la mañana. Teniendo en cuenta que los bytes se reciben del de menos peso al de mayor, en la tabla 5-3 se puede ver las diferentes equivalencias entre los formatos.

<table>
<thead>
<tr>
<th>Fecha y hora de la prueba</th>
<th>Hora en formato Epoch Hexadecimal</th>
<th>Hora en formato Epoch decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 de junio del 2013 a las 10:30:07 horas</td>
<td>51BEE52F</td>
<td>1371465007</td>
</tr>
</tbody>
</table>

Luego se puede observar, el nombre de la estación, el cual es SARTI. A continuación, el valor del contador de trama. Finalmente, los últimos tres bytes, se observa el valor 0x18, correspondiente a los 24 bits de cada dato, el valor 3, que corresponde a los tres canales de adquisición, y finalmente, el valor 0x32, que corresponde a las 50 muestras por segundo utilizadas.

Otro dato a evaluar, es el desfase entre los canales. En la figura 5-9, haciendo una aplicación sobre uno de los puntos, se podrá comprobar si el valor entre los diferentes canales varía.

Utilizando como ejemplo el punto 47, se puede observar que la aplicación representa el mismo valor para los tres canales. Por lo tanto con este resultado se puede decir que no hay un desfase apreciable, gracias a la manera en la que se ha implementado la adquisición, y a la congruencia entre las diferentes fuentes de reloj.
Figura 5-9. Visualización del punto 47 de la adquisición y su valor para cada canal

- 100 muestras por segundo

Para comprobar la adquisición a cien muestras por segundo, se recogerá una señal con las siguientes características: 10Hz, 1Vpp y 0.5V de offset.
Figura 5-10. Panel frontal de LabView durante la adquisición de la señal a 100 SPS

Observando la figura 5-10, se obtiene un resultado parecido al anterior. Se puede apreciar como ha hecho correctamente la adquisición de la señal descrita.

Igual que se ha realizado con la adquisición a cincuenta muestras por segundo, se tomará punto 74 como punto de comparación del desfase entre las señales. Por lo que se puede apreciar en la figura 5-11, no existe un desfase apreciable entre los diferentes canales, ya que al estar los tres canales adquiriendo la misma señal, no se observa diferencia entre los valores de la adquisición.
5.4.2 Lectura de los datos almacenados en memoria SD

Para la visualización de los datos almacenados en la tarjeta de memoria SD se utilizará una aplicación de MatLab. Esta aplicación será muy sencilla, ya que simplemente se encargará de abrir los ficheros de cada canal y representarlos.

En la figura 5-12, se puede observar el gráfico de tres ficheros de los datos obtenidos durante una hora de adquisición determinada.
La figura 5-12 muestra una adquisición de cada uno de los canales que, claramente, ha comenzado sobre los 45 minutos de la hora. Es por ese motivo, como se había explicado anteriormente, que antes de la información hay constantes ceros.

A simple vista puede parecer que los datos contienen una información ruidosa, pero es evidente que hay que aumentar la resolución para poder observarlos con claridad.

Figura 5-12. Datos correspondientes a una hora de adquisición en cada canal

Figura 5-13. Aumento de la señal adquirida vista en la figura 5-12
Como se comentaba, en la figura 5-13 se puede ver la señal sinusoidal adquirida por el sistema.

A continuación, se puede observar, en la figura5-14, la comparación de tres valores en un mismo instante de tiempo.

![Diagrama de comparación de tres valores](image)

Figura 5-14. Comparación de tres valores en un mismo instante de tiempo

Cómo se puede apreciar, prácticamente el valor es idéntico, salvo en el primer canal dónde el valor difiera en la cuarta cifra. Por lo tanto se puede afirmar que existe una buena sincronización de los datos almacenados en la memoria externa SD.

### 5.5 Prototipo

Finalmente, en la figura 5-15, se puede observar el prototipo del sistema de adquisición sobre el que se han realizado las diferentes pruebas.
Se pueden observar las diferentes partes mencionadas a lo largo de esta memora. Comparando el tamaño del dispositivo con el conector mini-USB, se puede apreciar que su tamaño no será robusto.
6 Conclusiones

El grupo de investigación SARTI-UPC tenía que diseñar un sistema de adquisición de datos sísmicos, aplicados a un volcán, en colaboración con el proyecto “Peligro Volcánico y Evaluación del Riesgo en Tenerife (PEVERTE)”. Principalmente este dispositivo tendría que cumplir los requisitos de bajo consumo y precio.

El primer paso fue la selección del microprocesador, el convertidor analógico-digital, el sensor y la memoria externa. Para ello se realizó un estudio de los componentes que cumplieran mejor los objetivos pautados. Por ello se escogieron el microprocesador MSP430F5428 de Texas Instruments, tres convertidores analógicos-digitales ADS1246, también de Texas Instruments y el sensor SM6 de Input/Output. Luego se fueron añadiendo los dispositivos necesarios para facilitar las comunicaciones, tales como el convertidor de TTL/RS232 MAX232, y la tarjeta SD.

Una vez obtenidos todos los componentes se procedió al conexionado de ellos. Teniendo en cuenta que tipo de comunicaciones se tenían que llevar a cabo, se diseñó, paso a paso, un esquema de la conexión del hardware, para poder diseñar y fabricar un primer prototipo sobre el que realizar diferentes pruebas. También, se planteó cómo se tendrían que configurar las diferentes comunicaciones y los convertidores, en relación a sus prestaciones y a las fuentes de reloj provenientes del microprocesador.

Una vez llevado a cabo el diseño del hardware, se empezó a poner en práctica el software. Para ello, en primer lugar se decidió que programas serían necesarios para la implementación y comprobación de la aplicación a diseñar. Una vez hecho esto, se diseñó una aplicación en lenguaje C. Esta aplicación explica de qué manera se adquieren, envían y recopilan los datos adquiridos. Se ha intentado hacer una descripción de la aplicación lo más detallada posible, teniendo en cuenta la dificultad que supone explicar un código tan extenso. Sin lugar a dudas, la programación de este código ha sido la parte más dificultosa y extensa de la realización de este proyecto. El mínimo error suponía buscar una nueva solución, la cual a veces no era la deseada. O el hecho de tener que tomar un camino diferente una vez se había avanzado por uno, como por ejemplo en el cambio de la utilización del convertidor A/D de tres canales a la utilización de tres de un solo canal.

Para poder implementar esta aplicación, se tuvo que realizar un proceso de aprendizaje y familiarización con el microprocesador. Antes de poder realizar todo el conjunto de la aplicación se tuvo que programar diferentes partes, para progresivamente poder acoplarlas. Finalmente se pudieron realizar diferentes pruebas con el dispositivo, comprobando su buen funcionamiento, y pudiendo caracterizarlo correctamente.
7 Bibliografía


8 Anexos

8.1 Manual de usuario

A continuación se explicará mediante un diagrama de bloques cómo arrancar la aplicación mediante HyperTerminal, una vez el programa esté cargado en el dispositivo. Se puede observar en la figura 8-1.

Figura 8-1. Manual de usuario de la aplicación
8.2 Código de Matlab

![Código de Matlab](image)

Figura 8-2. Código de ejemplo de aplicación en Matlab para abrir ficheros de tres canales