

**ESCUELA POLITÉCNICA
SUPERIOR DE CÓRDOBA**
Universidad de Córdoba



Departamento de Arquitectura de Computadores, Electrónica y Tecnología Electrónica

TRABAJO FIN DE GRADO
Grado en Ingeniería Electrónica Industrial

DISEÑO DEL INTERFAZ DE UNA BALANZA ELECTRÓNICA BASADA EN UNA CÉLULA DE CARGA

Autor: José Antonio Olmedo Rivera

Directores: Francisco Javier Quiles Latorre y Carlos Diego Moreno Moreno

Septiembre de 2017



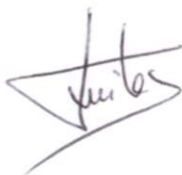
UNIVERSIDAD DE CÓRDOBA

Francisco Javier Quiles Latorre, del Área de Arquitectura y Tecnología de Computadores del Departamento de Arquitectura de Computadores, Electrónica y Tecnología Electrónica de la Universidad de Córdoba y **Carlos Diego Moreno Moreno**, del Área de Arquitectura y Tecnología de Computadores del Departamento de Arquitectura de Computadores, Electrónica y Tecnología Electrónica de la Universidad de Córdoba.

Informan:

Que el presente trabajo de fin de grado titulado *Diseño del interfaz de una balanza electrónica basada en una célula de carga*, que constituye la memoria presentada por **José Antonio Olmedo Rivera** para aspirar al Grado en Ingeniería Electrónica Industrial, ha sido realizado, bajo nuestra dirección, reuniendo, a nuestro juicio, las condiciones necesarias exigidas en este tipo de trabajos.

Y, para que así conste, expedimos y firmamos el presente informe en Córdoba, a 7 de Septiembre de 2017.



Fdo: Prof. Francisco Javier Quiles Latorre



Fdo: Prof. Carlos Diego Moreno Moreno

El autor:



Fdo: José Antonio Olmedo Rivera

Índice de contenidos

1. Introducción.....	3
2. Objetivos.....	5
3. Antecedentes.....	7
4. Tipos de balanzas digitales.....	9
5. Descripción de la solución adoptada.....	13
5.1. Recursos utilizados.....	14
5.1.1. Recursos hardware.....	14
5.1.2. Recursos software.....	14
5.1.2. Recursos humanos.....	14
6. Descripción del hardware de la balanza.....	15
6.1. Estructura de la balanza.....	15
6.2. Celda de carga.....	17
6.3. Filtrado del ruido proveniente de la celda de carga.....	21
6.4. Amplificación externa.....	22
6.5. Microcontrolador MSC1210.....	23
6.5.1. La placa UCOADP.....	24
6.5.2. ¿Qué es un convertidor analógico-digital Sigma-Delta ($\Delta\Sigma$)?.....	25
6.5.3. Convertidor analógico-digital $\Delta\Sigma$ del MCS1210.....	26
6.5.3.1. Frecuencias de muestreo de datos de entrada y de salida adoptadas.....	28
6.5.3.2. Resolución adoptada.....	28
6.5.3.3. Registros de control del ADC del MSC1210.....	31
6.5.4 Otros recursos del MSC1210 necesarios.....	35
6.5.4.1. Envío de los resultados.....	35
6.5.4.2. Espera de un determinado tiempo.....	37
6.5.4.3. Misceláneos.....	38
6.6. Visualizador LCD.....	40
6.7. Teclado.....	43
7. Ajustes de la balanza.....	45
7.1. Offset.....	45
7.2. Calibración.....	46
8. Descripción de la aplicación software.....	49
8.1. Herramienta software utilizada para la programación.....	49
8.2. Comentarios sobre el código.....	52
8.3. Descripción del enfoque seguido.....	53
9. Montaje y prueba de funcionamiento.....	59
10. Cambios efectuados en el diseño.....	63
11. Posibles mejoras futuras.....	65
12. Guía de usuario.....	67
13. Bibliografía.....	69
Anexo 1. Presupuesto.....	71
Anexo 2: Listado del código comentado.....	73
Anexo 3. Planos.....	95

Índice de figuras

Figura 1. Balanza de gancho.....	9
Figura 2. Balanza para cinta transportadora.	10
Figura 3. Balanza de plataforma.	10
Figura 4. Diagrama de bloques de la balanza.....	16
Figura 5. Puente de Wheatstone.....	17
Figura 6. Celda de carga de un único punto.....	18
Figura 7. Color de los cables de la celda de carga.....	19
Figura 8. Pines de la clavija.....	19
Figura 9. Celda carga con plataforma.....	20
Figura 10. Respuesta de un filtro paso bajo.....	21
Figura 11. Filtro paso bajo para eliminar ruido proveniente de la celda de carga.....	22
Figura 12. Diagrama de bloques del MSC1210.....	23
Figura 13. Esquema básico de un conversor analógico-digital Δ - Σ	26
Figura 14. Arquitectura del convertidor analógico-digital.....	27
Figura 15. Número efectivo de bits respecto al ratio de diezmo.....	30
Figura 16. Cronograma de un ciclo de lectura y escritura.....	41
Figura 17. Teclado tipo teléfono.....	43
Figura 18. Curva de calibración de la celda de carga.....	47
Figura 19. Editor de μ Vision.....	50
Figura 20. Gestor de proyectos y entorno de ejecución en tiempo real.....	51
Figura 21. Modo depuración.....	52
Figura 22. Diagrama del flujo de una transmisión.....	54
Figura 23. Diagrama de flujo de la función “gestion_adc()”.....	56
Figura 24. Diagrama de flujo del bucle infinito de sondeo.....	57
Figura 25. Montaje del prototipo.....	59
Figura 26. Balanza calibrada.....	61

1. Introducción

En la antigüedad ya existía la necesidad de tener una referencia exacta de los pesos de diferentes productos para poder comerciar con ellos. Esta necesidad fue satisfecha por la balanza, que se trataba de un instrumento mecánico compuesto de una palanca y dos platillos, que se utilizaba para determinar la masa de un objeto comparándolo con otro de masa conocida.

Con el avance tecnológico se ha conseguido desarrollar balanzas electrónicas, que se tratan de uno de los dispositivos más precisos jamás diseñados. Incluso los modelos de bajo coste nos permiten realizar una medida más precisa que la mayoría de balanzas mecánicas, sin embargo, las balanzas electrónicas son sensibles a algunos factores ambientales que pueden causar una lectura inexacta. Una de las principales diferencias respecto a las balanzas mecánicas, es que las balanzas electrónicas requieren de energía eléctrica suficiente para poder funcionar, algunos modelos se conectan a la red para obtener esta energía eléctrica, mientras que otras disponen de baterías para obtener la energía eléctrica necesaria.

Hoy en día las balanzas electrónicas son empleadas en multitud de aplicaciones, desde en cocinas, para determinar la cantidad de cada ingrediente de una receta, o en laboratorios, para determinar de forma exacta el peso de una pequeña cantidad de reactivos, hasta en la industria para determinar el peso de objetos con un peso considerable (incluso varias toneladas), mediante las denominadas balanzas electrónicas de suelo. Su uso está muy extendido por su precisión, exactitud y, sobre todo, por su facilidad de uso; para pesar un objeto solamente hay que depositarlo sobre el platillo de la balanza, y de ser necesaria alguna calibración, se puede realizar de forma automática pulsando un botón, y al instante el resultado aparece en un display.

Las balanzas electrónicas nos permiten determinar inmediatamente y de forma exacta el peso de un objeto, mediante la utilización de un sensor de fuerza. Dichas balanzas emplean una celda de carga.

La celda de carga se usa para convertir la fuerza peso, ejercida sobre el platillo de la balanza, en una señal eléctrica. El platillo es necesario para que la presión se ejerza de forma uniforme sobre toda la celda de carga. Si cambia la presión ejercida por el objeto que se encuentra sobre el platillo, cambiará el valor del voltaje que obtenemos de la celda de carga.

Esta señal eléctrica proveniente de la celda de carga será acondicionada para eliminar el ruido eléctrico y para conseguir una determinada resolución. A continuación la señal acondicionada será digitalizada, mediante el paso a través de un convertidor analógico digital. Una vez digitaliza la señal será procesada por un microcontrolador, que enviará el resultado del peso obtenido a un visualizador LCD, para ver el resultado con una determinada resolución. También al procesar la señal, dichas balanzas pueden realizar varias funciones que no podríamos realizar con una balanza mecánica, como son contar las piezas colocadas sobre el platillo, comunicarnos con un ordenador a través de un puerto serie, o almacenar los resultados en una memoria SD.

2. Objetivos

El objetivo de este trabajo es por un lado diseñar un circuito para la amplificación de la señal proveniente de la celda de carga, y por otro lado diseñar el programa para el control de la balanza, utilizando una placa de desarrollo basada en el microcontrolador MSC1210. Además la balanza deberá contar con las siguientes características:

- Rango de pesado comprendido entre 0 y 150kg.
- Resolución de al menos 100gr.
- Opción de calibración.
- Una interfaz serie para la comunicación.
- Dispondrá de una pantalla LCD, para mostrar los resultados, y de un pequeño teclado, para la configuración de la balanza.

3. Antecedentes

En la escuela politécnica de Córdoba no se ha desarrollado ningún trabajo relacionado con el diseño y control de una balanza, pero si se han realizado algunos en los que se utilizan celdas de carga, como son los siguientes:

- *Mejora de un prototipo de medida de erosión de suelo* (2015).
Autor: Óscar Pérez Martín.
Directores: Rafael Pérez Alcántara, Francisco José Bellido Outeiriño.
- *Sistema de alimentación automática para explotaciones ganaderas de caballos en semi-libertad* (2015).
Autor: Manuel Fajardo García.
Director: Antonio Moreno Fernández-Caparrós.

Y también se han realizado varios trabajos relacionados con el diseño y control de otros dispositivos a través de un microcontrolador, como son los siguientes:

- *Diseño de un sistema de control de temperatura para la carga de un camión frigorífico basado en microcontrolador y utilizando tecnología GSM* (2006).
Autor: Enrique Rael Fuster.
Directores: Manuel Agustín Ortiz López, Carlos Diego Moreno Moreno.
- *Sistema de control de iluminación mediante microcontrolador* (2008).
Autor: Jesús del Rey Olegario.
Director: José María Flores Arias.
- *Diseño del hardware de un módulo portátil de adquisición datos* (2009).
Autor: Rafael Jiménez Fernández.
Directores: Francisco Javier Quiles Latorre, Manuel Agustín Ortiz López.
- *Diseño de un sistema de control e iluminación de Led's de alta luminiscencia* (2009).
Autor: Jesús Herrera Beurnio.
Director: Antonio Moreno Fernández-Caparrós.
- *Diseño de una placa de control de una impresora 3D basada en Arduino* (2015).
Autor: Paula García-Moreno Estrella ;
Directores: Francisco Javier Quiles Latorre y Rafael Pérez Alcántara.
- *Control de actuador neumático/hidráulico a través de microcontrolador* (2015).
Autor: Juan Carlos Almoguera Torres.
Director: Juan Jesús Luna Rodríguez.
- *Placa de control de servomotores mediante PWM* (2015).
Autor: Gabriel González Páez.
Director: Antonio Moreno Fernández-Caparrós.

- *Diseño de control de una servoválvula con la plataforma de bajo coste Arduino (2016).*

Autor: Francisco Mohedano Bujalance.

Directores: Francisco Javier Vázquez Serrano y Mario Luis Ruz Ruiz.

4. Tipos de balanzas digitales

En el mercado existen diferentes tipos de balanzas dependiendo del uso que se le vaya a dar [5].

Dependiendo de la estructura, por un lado tenemos las balanzas de gancho grúa o dinamómetros digitales, que son balanzas para pesadas en vertical a través de un gancho que sujeta el objeto en el aire. Este tipo de balanzas resultada ideal para el comercio minorista, debido al gran efecto óptico que presenta en el escaparate; aunque también se puede usar en la industria debido a la amplia gama de balanzas con rangos de pesado que van desde 20 gramos a 10 toneladas.



Figura 1. Balanza de gancho.

Otro tipo de balanza dependiendo de su estructura son las cintas transportadoras pesadoras, que se tratan de sistemas de pesaje integrados en cintas transportadoras, que determinan el peso gracias a un sistema de pesaje dinámico. Este tipo de balanzas se usan en sistemas en los que se necesite pesar objetos en movimiento.



Figura 2. Balanza para cinta transportadora.

Por último tenemos las balanzas de plataforma, que se caracterizan por contar con una o más celdas de carga instaladas bajo la plataforma o plato donde se colocan los objetos. Su campo de aplicación es muy versátil, con frecuencia se emplean en la industria, en establecimientos y en otro tipo de locales.

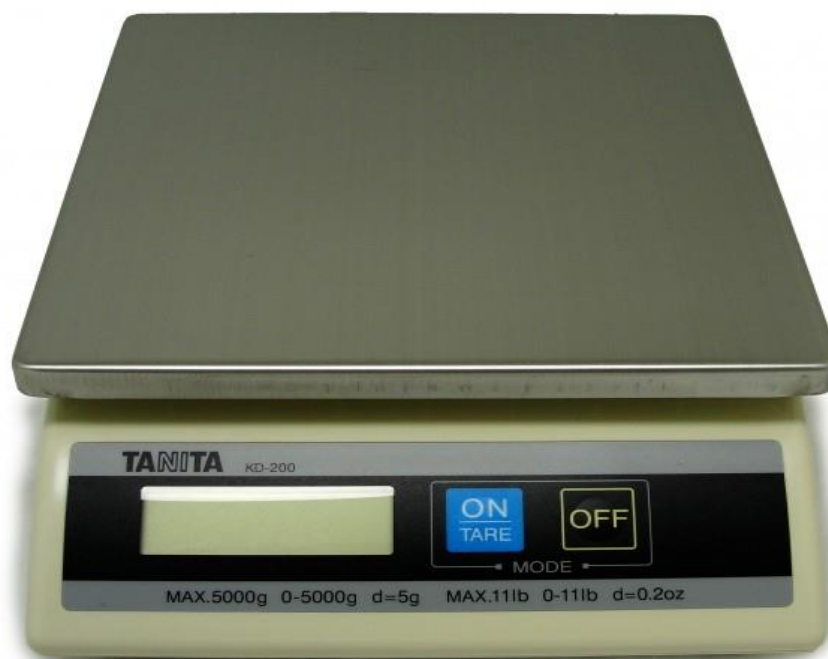


Figura 3. Balanza de plataforma.

Existen muchos tipos de diseños de este tipo de balanzas. Por ejemplo existen balanzas con un rango de pesaje pequeño y con una resolución muy pequeña (menor a 1 gramo), que se utilizan en los laboratorios, otras que cuentan con una función para contar las piezas que se depositan sobre ella, y que se utilizan para realizar inventario, incluso otras con amplios rangos de pesado para usos industriales, etc.

5. Descripción de la solución adoptada

La balanza desarrollada en este proyecto se ha diseñado pensando que se usará para pesar colmenas de abejas. Teniendo esto en mente, como pesar las colmenas es mucho más sencillo de forma horizontal, y no es necesario pesarlas en movimiento, hemos decidido que la balanza se diseñará como una balanza de plataforma.

La balanza estará compuesta por una celda de carga con una plataforma, sobre la que podemos poner los objetos que queramos pesar, a la que podemos añadir un platillo para depositar objetos más pequeños.

La señal de salida de la celda carga se trata de una señal de voltaje muy pequeña con ruido eléctrico presente en ella; por lo que hemos diseñado un prototipo que se encarga de acondicionar esta señal para llevarla al microcontrolador, que incluye un conversor analógico digital para convertir la señal analógica, proporcional al peso del objeto depositado sobre la plataforma, en una señal digital. Dicha señal digitalizada es procesada por el microcontrolador y convertida en un valor expresado en kilogramos.

El prototipo para acondicionar la señal de salida de la celda de carga dispondrá de un amplificador de instrumentación, para amplificar la señal; un circuito de referencia de tensión, que genera la tensión utilizada para alimentar la celda de carga con la mayor precisión; y un filtro paso bajo, a la salida de la señal de la celda de carga, para filtrar el ruido proveniente de ella, y evitar que este se amplifique.

El microcontrolador que hemos utilizado dispone de un amplificador de ganancia programable, que también hemos aprovechado para amplificar más la señal y obtener una mayor resolución. Una vez digitalizada la señal se convierte en un valor expresado en kilogramos aplicando la siguiente fórmula:

$$\text{Peso(kg)} = \frac{V_o * P_{\text{máx}}}{S_{cc} * V_{\text{exc}}} : G\text{-offset}$$

(Ecuación 5.1)

Siendo “Peso(kg)” el peso observado en kilogramos, “Vo” el voltaje de salida de la celda de carga una vez acondicionado, “Scc” la sensibilidad de la celda carga, que se mide en mV/V, “Vexc” la tensión de excitación con la que se alimenta la celda de carga, “G” el valor de la ganancia externa (la ganancia interna se incluye dentro de “Vo”), y “offset” el peso que presenta la balanza cuando no hay ningún objeto sobre ella.

La balanza dispone de un visualizador LCD alfanumérico para mostrar el peso resultante, y tiene la posibilidad de enviar por el puerto serie el peso para poder comunicarnos con un ordenador. También dispone de un teclado matricial para poder poner la balanza a cero, calibrarla, realizar una tara, y decidir si los resultados se envían o no por el puerto serie.

La balanza está compuesta de una parte hardware y de una parte software. La parte hardware está formada por la celda de carga, el prototipo para acondicionar la señal de salida proveniente de ella, el microcontrolador, el teclado y el visualizar LCD. La parte software se trata del programa de aplicación que se ejecuta en el microcontrolador para

controlar la balanza; el código de este programa de aplicación se ha diseñado usando una herramienta software que permite realizar el programa mediante el lenguaje C.

5.1. Recursos utilizados

5.1.1. Recursos hardware

- Ordenador portátil personal.
- Conexión a Internet.
- Equipo de soldadura *JBC-AM6500*.
- Multímetro *Agilent 3401A*.
- Fuente de alimentación *Agilent E3631A*.
- Osciloscopio *Tektronix MSO 4032*.

5.1.2. Recursos software

- Sistema operativo *Microsoft Windows 10*.
- Herramienta de edición y programación *Keil μ Vision 5*.
- Herramienta para la programación a través del puerto serie *TI Downloader*.
- Herramienta para la comunicación a través del puerto serie *Termite*.
- Herramienta de diseño *OrCad 16.6*.
- Paquete de ofimática *Microsoft Word 2003*.
- Herramienta de edición *SciTe*.

5.1.2. Recursos humanos

- Directores del trabajo: Francisco Javier Quiles Latorre y Carlos Diego Moreno Moreno.
- Autor del trabajo: José Antonio Olmedo Rivera.

6. Descripción del hardware de la balanza.

6.1. Estructura de la balanza

El diagrama de bloques de la figura 4 muestra la estructura del hardware de la balanza. El peso colocado sobre la celda de carga, hará que la celda de carga entregue una determinada señal eléctrica, de valor muy pequeño. A continuación, esta señal pasa por una etapa de filtrado, para eliminar el ruido proveniente de la celda de carga. Una vez hecho esto, la señal pasa por una etapa de amplificación externa al microcontrolador, para aumentar el valor de la señal y conseguir una mayor resolución.

Una vez que se amplifica externamente la señal entra en la placa UCOADP, más concretamente en el microcontrolador MSC1210, en el que volverá a ser amplificada, esta vez mediante un amplificador interno, pasa por el convertidor analógico digital Sigma-Delta ($\Delta\Sigma$), se procesa el valor digital para determinar el peso, mediante el núcleo 8051 del MSC1210, y se envía este a una pantalla LCD y, si queremos, también a través de la UART0 a un ordenador. El control del teclado y del visualizador LCD se realiza a través del CPLD de la placa UCOADP.

La alimentación de la placa UCOADP, y la conexión entre la UART0 y el ordenador, se realiza mediante el dispositivo *FTDI 232*.

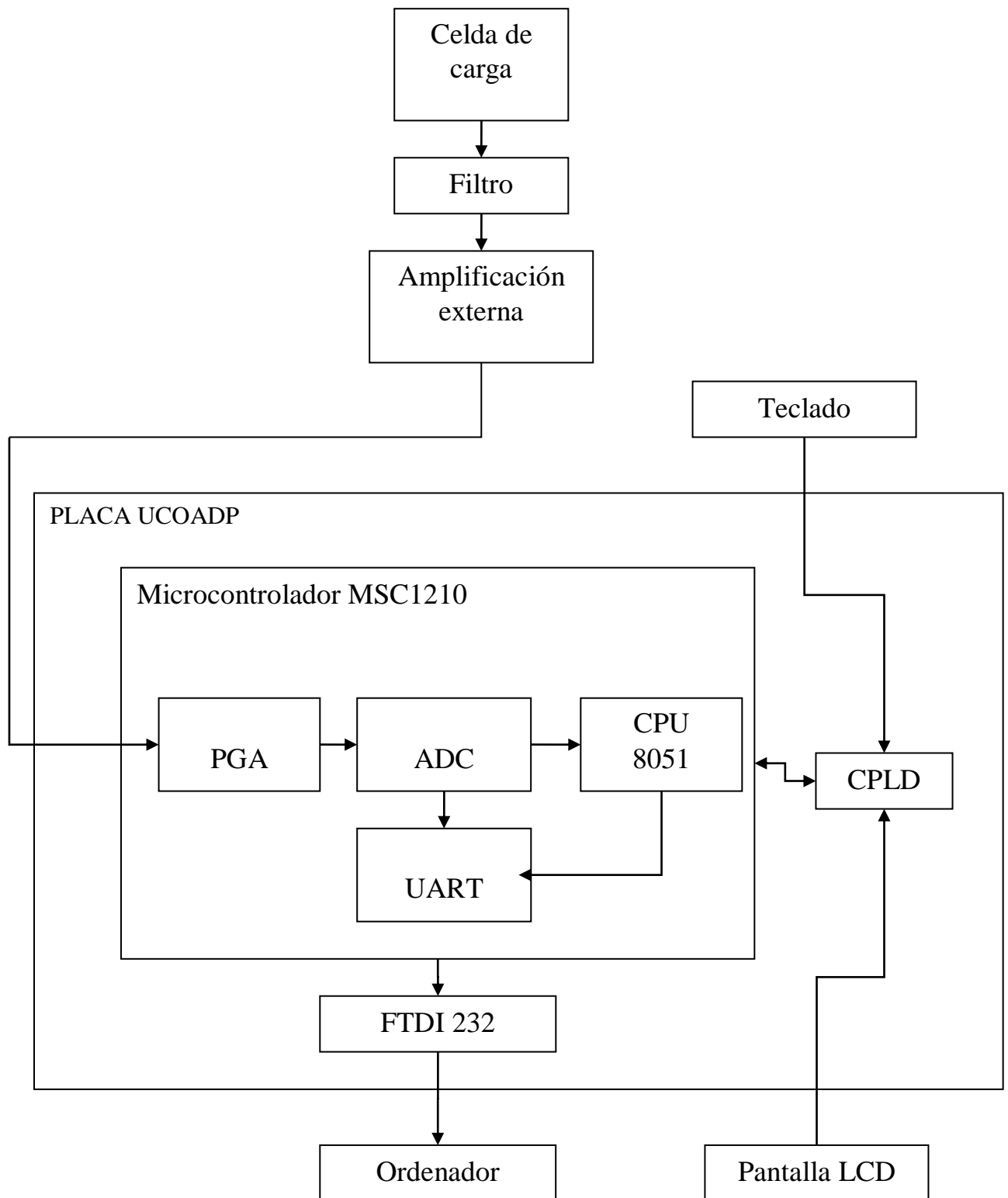


Figura 4. Diagrama de bloques de la balanza.

6.2. Celda de carga

Las celdas de carga (o células de carga) [4] son los sensores de fuerza más comunes en el mercado. Se trata de un transductor que convierte la fuerza aplicada sobre la celda de carga en una señal eléctrica medible. Entre los distintos tipos de celda de carga que existen, la utilizada en este proyecto se trata de una celda de carga basada en galgas extensométricas resistivas.

Las galgas extensométricas se basan en la variación de resistencia cuando un conductor, o un semiconductor, es sometido a un esfuerzo mecánico. Las galgas que estén en tracción aumentarán su resistencia, mientras que las que están en compresión disminuirán su resistencia, como resultado de aplicar una serie de fuerzas.

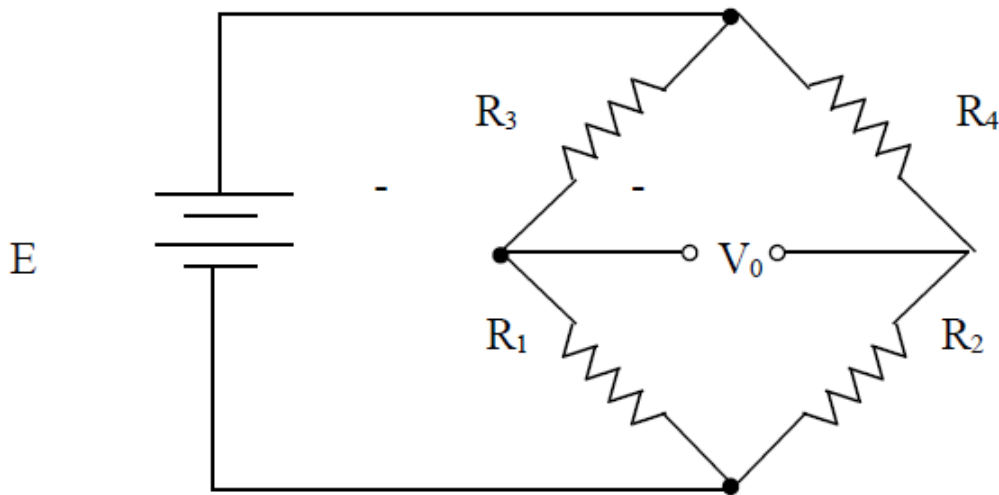


Figura 5. Puente de Wheatstone.

Para medir esta variación de resistencia, la celda de carga utiliza un puente de Wheatstone, que está constituido por cuatro resistencias pasivas o variables que forman un cuadrado. En una de las diagonales se aplica la tensión de alimentación del puente, mientras que en la otra diagonal se obtiene un valor de tensión que depende de las cuatro resistencias que forman el puente, permitiéndonos así medir la variación de resistencia. Existen diferentes tipos de puentes de Wheatstone dependiendo de cuáles de las cuatro resistencias sean sensores, es decir, galgas resistivas.

En este proyecto la celda de carga que se ha utilizado es una celda de carga de punto único fabricada por *HIWEIGH*, la cual se muestra junto con sus dimensiones en la figura 6.

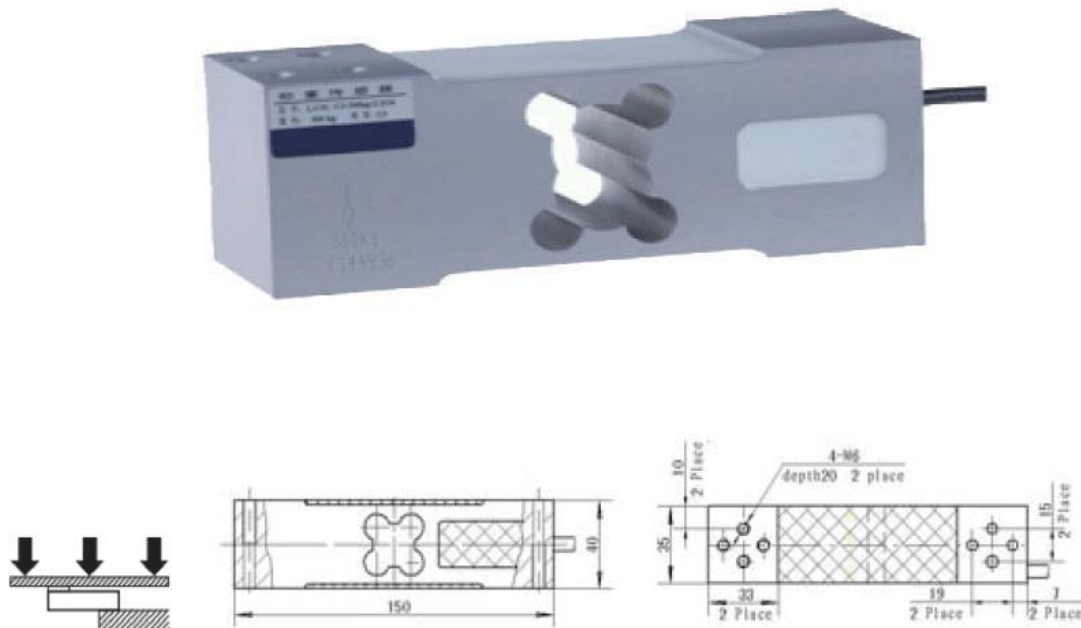


Figura 6. Celda de carga de un único punto.

Sus características se indican en la siguiente tabla:

Capacidad	50/100/150/200/300/400Kg
Sensibilidad	$2.0 \pm 0.2\text{mV/V}$
Variación a la salida con un peso constante (“Creep”) en 30 minutos	0.02%FS (<i>Full scale</i>)
Equilibrio de zero	$\pm 2\% E_{\text{max}}$
Resistencia de entrada	$406 \pm 6\Omega$
Resistencia de salida	$350 \pm 2\Omega$
Resistencia de aislamiento	$\geq 5000\text{M}\omega$
Rango de temperatura de operación	$-20\dots 50^{\circ}\text{C}$
Carga segura	120%FS
Límite de sobrecarga	150%FS
Excitación recomendada	5-10Vdc
Protección	IP65
Longitud de cables	1.6m
Diámetro de los cables	5mm

De la celda de carga salen 4 cables, cada uno de un color, que se corresponden con los cuatro puntos del puente de Wheatstone de acuerdo a la siguiente figura:

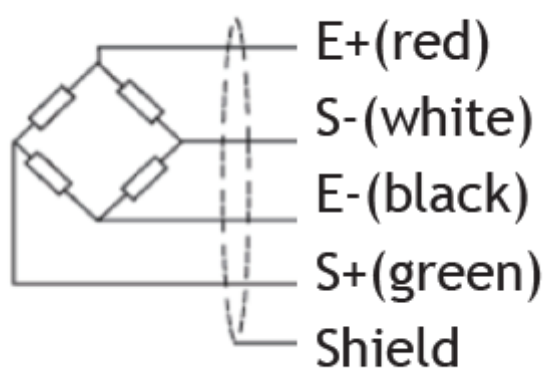


Figura 7. Color de los cables de la celda de carga.

Estos cuatro cables se conectan a una clavija de 7 pines, para facilitar su conexión, de la cual podemos ver sus pines en la siguiente figura:

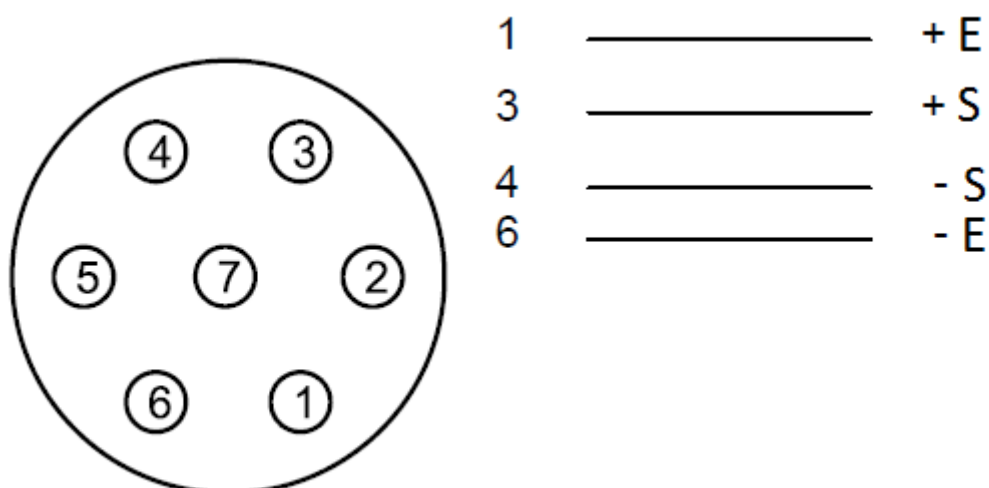


Figura 8. Pines de la clavija.

De entre los diferentes modelos de estas celdas de carga, de acuerdo a la capacidad, se ha usado el modelo de 300kg de capacidad junto con una plataforma. El conjunto se muestra en la figura 9.

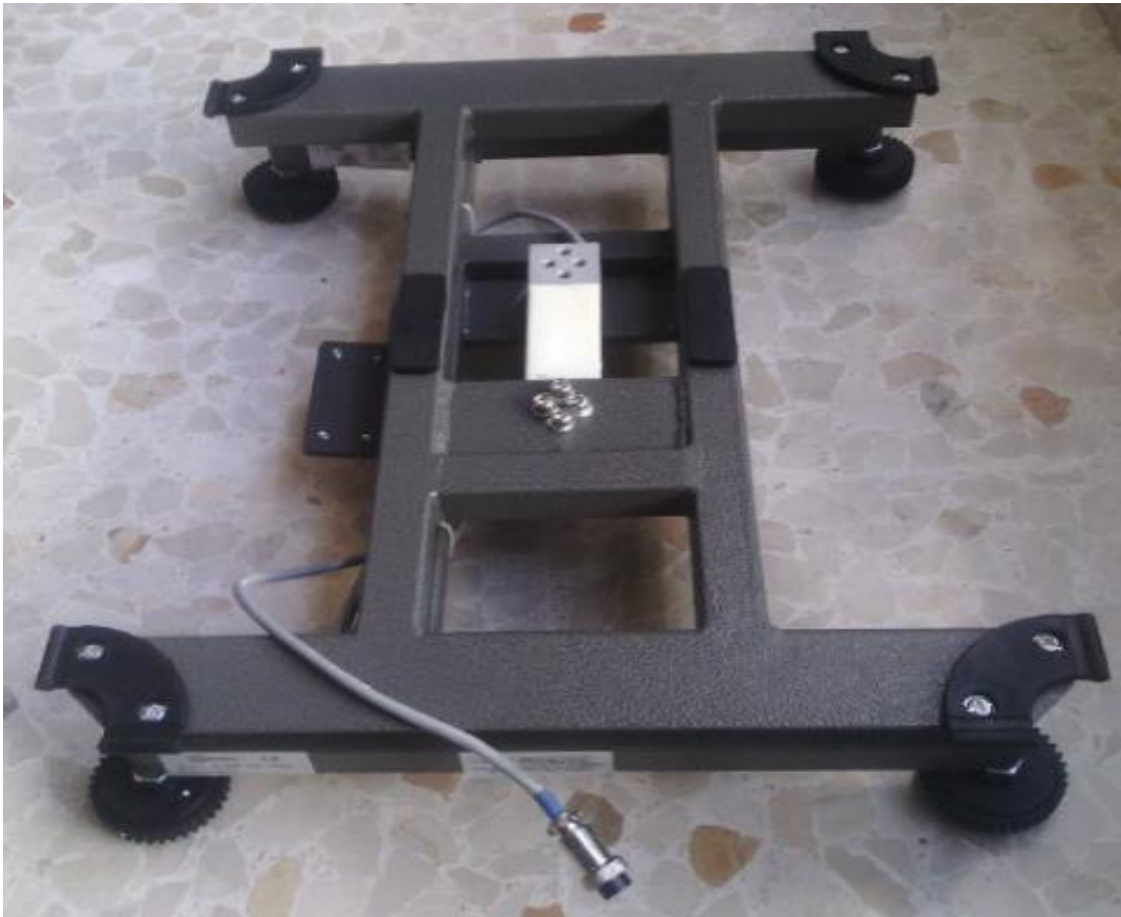


Figura 9. Celda carga con plataforma.

La tensión de alimentación que utilizamos tiene que tener una gran estabilidad, ya que no es posible medir variaciones de una magnitud con un sensor, si no podemos garantizar una estabilidad muy elevada de la tensión de alimentación. Por ello, para alimentar la celda de carga hemos utilizado un generador de tensión de referencia; en concreto hemos utilizado el integrado *LT1461* [15], que proporciona una tensión de salida (V_{ref}) de 4.096V con una precisión mínima del 0.04%.

Esta tensión de alimentación es menor a la recomendada (5-10Vdc), ya que para alimentar el generador de tensión de referencia es necesario utilizar una tensión 300mV mayor a la que proporcionará el generador, y como para alimentarlo se dispone de los 5V con los que se alimenta la placa UCOADP; para no tener que utilizar otra tensión de alimentación se ha utilizado estos 5V, por lo que el circuito generador de tensión de referencia utilizado es de 4.096V.

Otra característica que se debe tener en cuenta en la selección del generador de tensión de referencia es la intensidad de salida. En este caso, debe suministrar una intensidad de salida superior a la que consume el puente de la celda de carga. Como el puente tiene una resistencia de entrada de 406Ω , este consume 1mA aproximadamente. El *LT1461* tiene una intensidad de salida típica de 100mA, que es más que suficiente.

Dado que la celda de carga es de 300kg, su sensibilidad es de 2mV/V, y se alimenta con una tensión de 4.096V, el valor máximo que se tendrá a la salida será:

$$V_o(300\text{Kg}) = \text{Sensibilidad} \cdot V_{\text{ref}} = 2\text{mV/V} \cdot 4.096\text{V} = 8.192\text{mV} \quad (\text{Ecuación 6.1})$$

6.3. Filtrado del ruido proveniente de la celda de carga

Como la señal de voltaje de salida de la celda carga es muy pequeña (del orden de los milivoltios), y esta genera un ruido eléctrico debido a su vibración mecánica, se debe pasar por un filtro paso bajo, con el propósito de eliminar estas señales no deseadas, que están afectando a la magnitud que queremos medir, antes de amplificar dicha señal. Si no se utiliza este filtro, parte del ruido proveniente de la celda carga no puede ser eliminado por el amplificador, sino que se amplifica, provocando un error DC a la salida.

Para filtrar este ruido, se utiliza un filtro pasa bajo, que permite el paso de las señales inferiores a la frecuencia de corte (f_c), atenuando enormemente las señales de frecuencia mayor a la de corte.

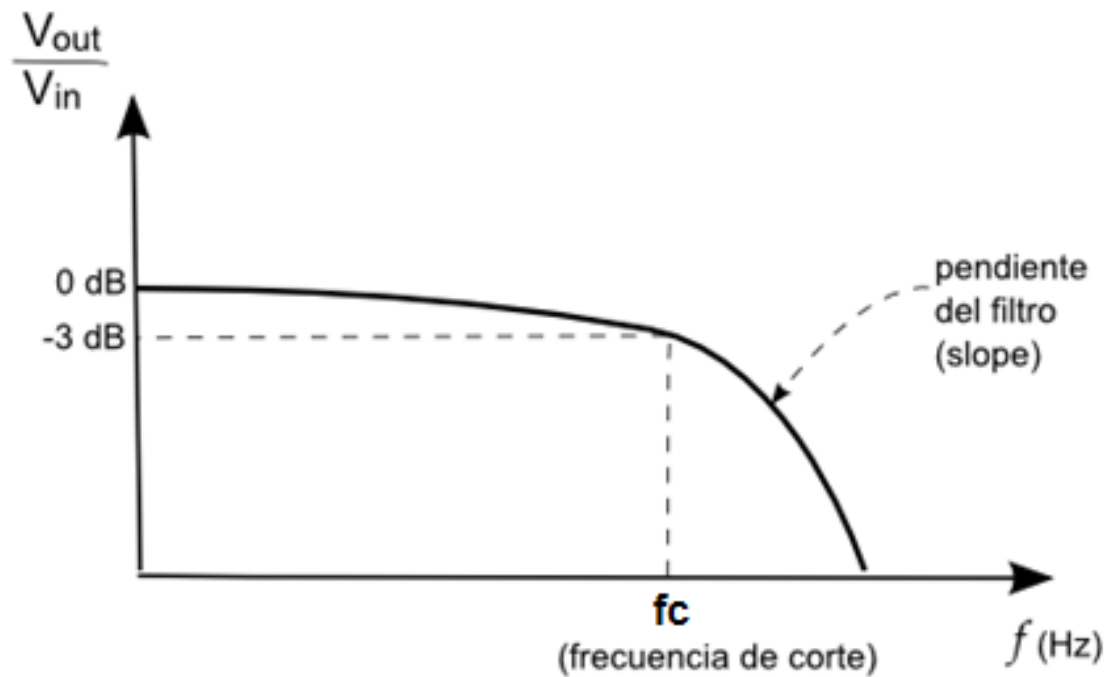


Figura 10. Respuesta de un filtro paso bajo.

Para este proyecto se ha utilizado un filtro pasivo para el modo común, que es el que se ha visto que es recomendado utilizar para diseños de balanzas con el microcontrolador MSC1210 [6], su configuración es la siguiente:

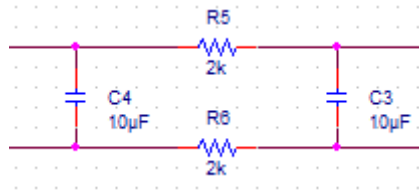


Figura 11. Filtro paso bajo para eliminar ruido proveniente de la celda de carga.

Con este filtro se tiene una frecuencia de corte de:

$$f_c = \frac{1}{2 * \pi * R * C} = \frac{1}{2 * \pi * 2000 * 0.00001} = 7.96\text{Hz} \text{ (Ecuación 6.2)}$$

6.4. Amplificación externa

Para obtener una mayor resolución es necesario amplificar la señal proveniente de la celda de carga; para ello se ha optado por amplificar de forma externa al microcontrolador y utilizar también la ganancia interna programable del microcontrolador MSC1210. No se ha usado solamente la ganancia interna programable porque a medida que se utiliza un mayor valor de la ganancia interna, aumenta el ruido en la señal del convertidor analógico digital, haciendo que haya menos bits libres de ruido en la conversión.

Para realizar la amplificación externa se ha utilizado un amplificador de instrumentación debido a su alta impedancia de entrada, su elevada relación de rechazo en modo común y, sobre todo, a su capacidad de tener una ganancia en voltaje dependiente de una sola resistencia. Para este proyecto se ha utilizado el amplificador de instrumentación de precisión *INA122P* [14], que resulta ideal para la adquisición de señales diferenciales de bajo ruido, por sus excelentes características:

- Baja corriente de reposo: 60µA.
- Amplio rango de alimentación: 2.2V a 36V (Single Supply) y -0.9/+1.3V a ±18V.
- Oscilación de salida *rail-to-rail*.
- Bajo voltaje de *offset*: máximo de 250µV.
- Bajo *offset drift*: máximo de 3µV/°C.
- Bajo ruido: 60nV/√Hz.
- Baja desviación de la corriente de entrada: máximo de 25nA.

Para fijar la ganancia se ha utilizado una resistencia de 2.01kΩ, por lo que la ganancia, calculada con la fórmula sacada del datasheet del componente, es:

$$G = 5 + \frac{200k}{R_g} = 5 + \frac{200k}{2.01k} \approx 104.50 \text{ (Ecuación 6.3)}$$

Por lo que para los 300kg tendremos $104.50 * 8.192\text{mV} \approx 856.08\text{mV}$.

6.5. Microcontrolador MSC1210

Para este proyecto hemos decidido utilizar el microcontrolador MSC1210 [1], el cual se trata de un microcontrolador de alta velocidad, basado en el 8051 y desarrollado por *Texas Instruments*. Está diseñado para mediciones de alta resolución, aplicaciones en transmisores inteligentes, control de procesos industriales, básculas de pesaje (como es nuestro caso) e instrumentación portátil.

Este microcontrolador, cuyo diagrama de bloques se ve en la figura 12, incorpora un conversor analógico-digital Sigma-Delta ($\Delta\Sigma$) con 8 canales de entrada, y 24 bits de resolución cada uno. Además tiene dos UART (*Universal Asynchronous Receiver-Transmitter*) y un SPI (*Serial Peripheral Interface*), para la comunicación serie; una memoria Flash interna de capacidad variable (entre 4k y 32kB, dependiendo del modelo), que se utiliza como memoria de programa o de datos; un módulo de memoria RAM adicional; funciones analógicas y periféricos de alto rendimiento.

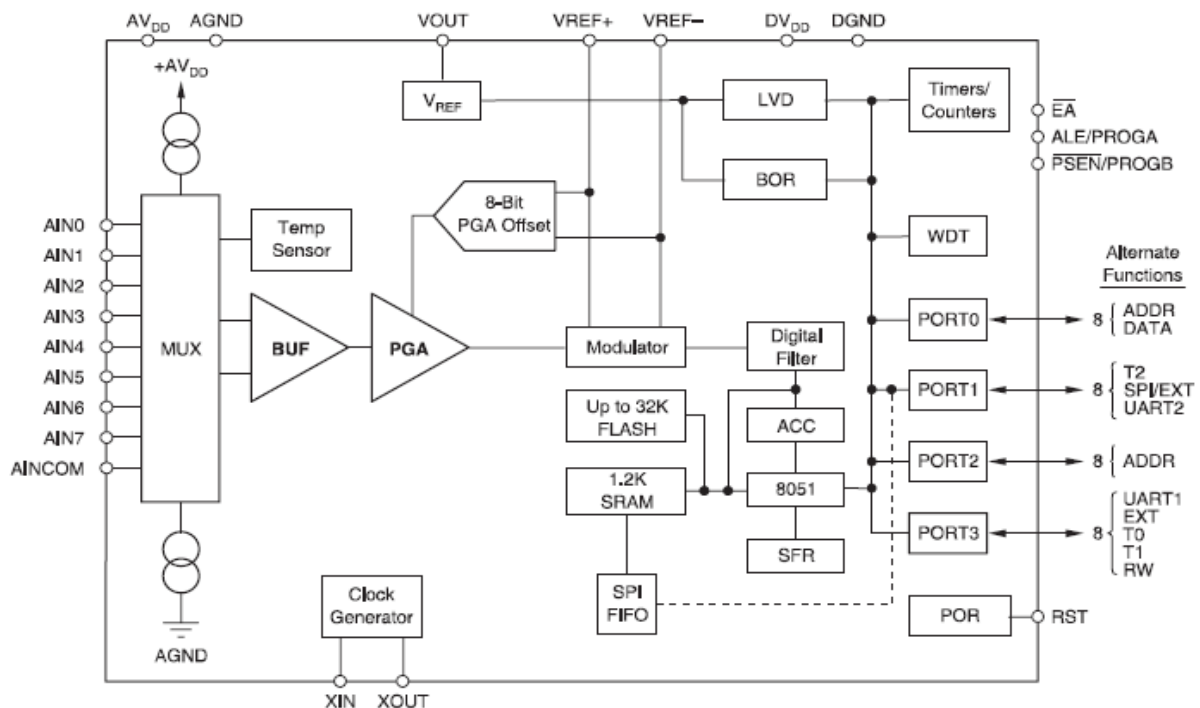


Figura 12. Diagrama de bloques del MSC1210.

Como se ve en la figura 12, el MSC1210 tiene la alimentación analógica y digital separada, las cuales pueden ser independientemente alimentadas con tensiones que van desde +2.7V a +5.5V, teniendo el componente una disipación de potencia menor a 4mW, funcionando con una tensión de alimentación de +3V.

El MSC1210 es compatible con el conjunto de instrucciones del 8051. El núcleo de este microcontrolador es un núcleo 8051 optimizado con el que se consigue que las instrucciones se ejecuten 3 veces más rápido que con un núcleo del 8051 estándar (4 ciclos de reloj por ciclo máquina en vez de 12), lo que hace posible ejecutar un

programa logrando el mismo rendimiento, que un núcleo del 8051 estándar con una frecuencia de reloj externa menor.

Hemos elegido el MSC1210 ya que cuenta con un conversor analógico digital de gran resolución, además de tener un amplificador de ganancia programable, gracias al cual podemos aumentar la resolución de la medida. Por otra parte, permite enviar los datos por el puerto serie, y una velocidad de ejecución relativamente elevada, aunque esto no es determinante para una balanza, ya que no es necesario que las mediciones se realicen cada muy poco tiempo.

Para comprobar el funcionamiento de la balanza, se ha utilizado la placa de adquisición de datos, desarrollada en el proyecto final de carrera “*Diseño del hardware de un módulo portátil de adquisición de datos*” realizado por Rafael Jiménez Fernández, a la cual me referiré a partir de ahora como placa UCOADP [10].

6.5.1. La placa UCOADP

La placa UCOADP se trata de un registrador de datos desarrollado en la universidad de Córdoba. Un registrador de datos es un tipo especial de adquirente de datos que almacena datos según transcurre el tiempo o según la localización del dispositivo, mediante el uso de instrumentos y sensores propios o externos. Este tipo de dispositivos están pensados para aplicaciones donde se desean recoger gran cantidad de datos a lo largo del tiempo.

El bloque de control de dicha placa cuenta con dos circuitos VLSI; por un lado está el microcontrolador MSC1210, cuya operación es variable dependiendo de variables externas y de un programa cargado en memoria, por otro lado está un circuito CPLD (*Complex Programmable Logic Device*), que se trata de un dispositivo lógico programable que está formado múltiples bloques lógicos equivalentes a pequeñas estructuras PAL (*Programmable Array Logic*) que se comunican entre sí utilizando una matriz de conexiones. La programación del CPLD queda definida antes de la operación del módulo UCOADP, con el objetivo de descargar al microcontrolador de operaciones que puedan realizarse con lógica secuencial o combinacional sencilla.

Entre otras muchas características con las que cuenta la placa UCOADP, mencionaré las que nos servirán para este proyecto, como son los conectores de los que dispone para conectar un visualizador LCD alfanumérico de 2 filas con 16 caracteres cada una, y un teclado matricial de cuatro filas y cuatro columnas, ambos controlados a través del CPLD, simplificando así la programación de ambos.

También la placa cuenta con un circuito USB-UART que permite utilizar el USB del dispositivo como un puerto COM virtual, permitiendo la comunicación con el dispositivo mediante programas tipo *HyperTerminal*. Por lo tanto, se puede programar el microcontrolador a través del puerto serie; para ello será necesario entrar en el modo programación, al que se entra cuando se produce un reset y el bit PSEN está a nivel lógico bajo. Una vez programado el microcontrolador, para salir del modo programación, para que el microcontrolador pueda ejecutar la aplicación, habrá que poner a nivel lógico alto el bit PSEN y producir un reset. La placa UCOADP tiene un

Jumper diseñado para poner a nivel bajo el bit PSEN, el cual permanecerá en estado bajo cuando el *Jumper* esté pulsado.

La alimentación de toda la placa se realiza mediante conexión USB gracias al dispositivo *FTDI 232* instalado en ella.

También cabe mencionar que la placa UCOADP sólo se piensa utilizar para realizar la prueba de la balanza, la idea sería sustituirla por un microcontrolador MSC1210 solo cuando se monte de forma definitiva, ya que dicha placa dispone de muchos recursos que no son aprovechados por una balanza. En el montaje definitivo el visualizador LCD y el teclado pueden ser controlados a través de los puertos del microcontrolador, ya que en un principio no se prevé otro uso para ellos.

6.5.2. ¿Qué es un convertidor analógico-digital Sigma-Delta ($\Delta\Sigma$)?

Los convertidores Sigma-Delta [12] se utilizan en aplicaciones en las que se necesita medir una señal de tensión analógica con una precisión razonablemente alta. Los convertidores analógico-digitales $\Delta\Sigma$ son comunes en la digitalización de señales de audiofrecuencia, en sistemas de adquisición de datos y en aplicaciones de medición (como es nuestro caso).

Este convertidor está formado por dos bloques: un modulador $\Delta\Sigma$, que muestrea y convierte la señal, y un filtro-diezmadador, que elimina todas las componentes fuera de la banda de señal y reduce la frecuencia de muestreo, mediante un proceso de diezmo (*decimation*).

En el modulador $\Delta\Sigma$, cuyo esquema básico se ve en la figura 13, la señal de error (V_e), entre la entrada analógica y la salida de un conversor digital-analógico de un bit, entra en un integrador, cuya salida se compara con la tierra del sistema mediante un comparador, el cual actúa como un conversor analógico-digital de 1 bit generando una salida binaria, $+V$ o $-V$ dependiendo de si la salida del comparador es positiva o negativa. A continuación la señal pasa por un biestable tipo D, que marca la frecuencia de sobremuestreo ($k \cdot f_s$), y la salida del biestable se realimenta a la entrada, pasando previamente por un conversor digital-analógico, para calcular nuevamente el error V_e . La salida del modulador $\Delta\Sigma$ consiste en una cadena de pulsos de frecuencia $k \cdot f_s$, que pasan por el filtro diezmadador, para disminuir considerablemente el ruido y la frecuencia de muestreo de $k \cdot f_s$ a f_s .

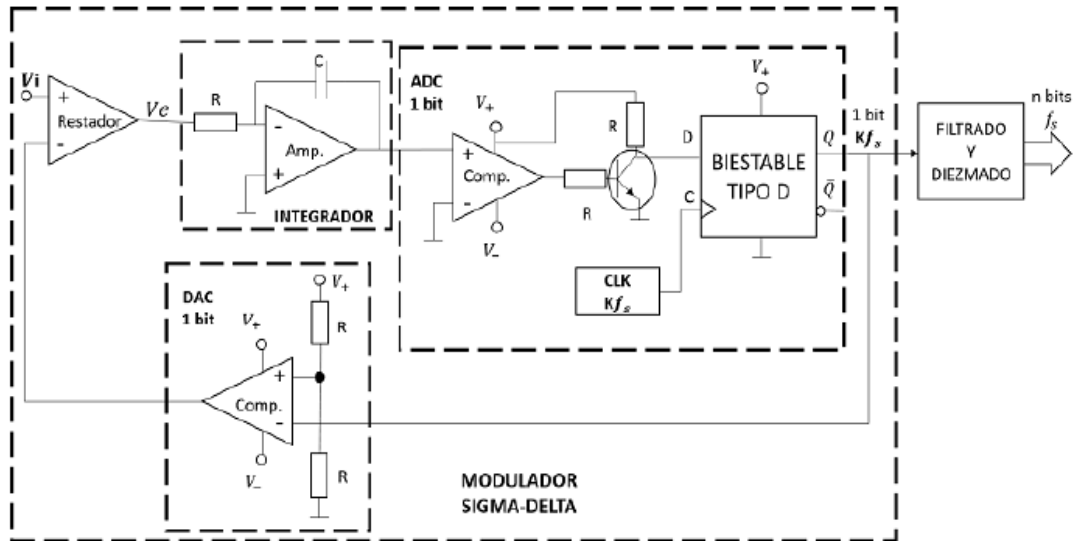


Figura 13. Esquema básico de un convertor analógico-digital Δ - Σ .

Como se ve en la figura 13, el convertidor analógico-digital dispone de dos frecuencia de muestreo, la frecuencia de muestreo de entrada ($k \cdot f_s$) y la frecuencia de muestreo de salida de datos (f_s). La frecuencia de muestreo de entrada es mucho mayor que la frecuencia de salida, por lo que esta técnica se denomina sobremuestreo. La relación entre la frecuencia de muestreo entrada y de salida se denomina ratio de diezmo (k).

6.5.3. Convertidor analógico-digital $\Delta\Sigma$ del MCS1210

El convertidor analógico-digital (ADC) $\Delta\Sigma$ del MCS1210 [1] está compuesto por un multiplexor de entrada, un buffer opcional, un amplificador de ganancia programable (PGA) y un filtro digital programable. Su arquitectura se ve en la siguiente figura:

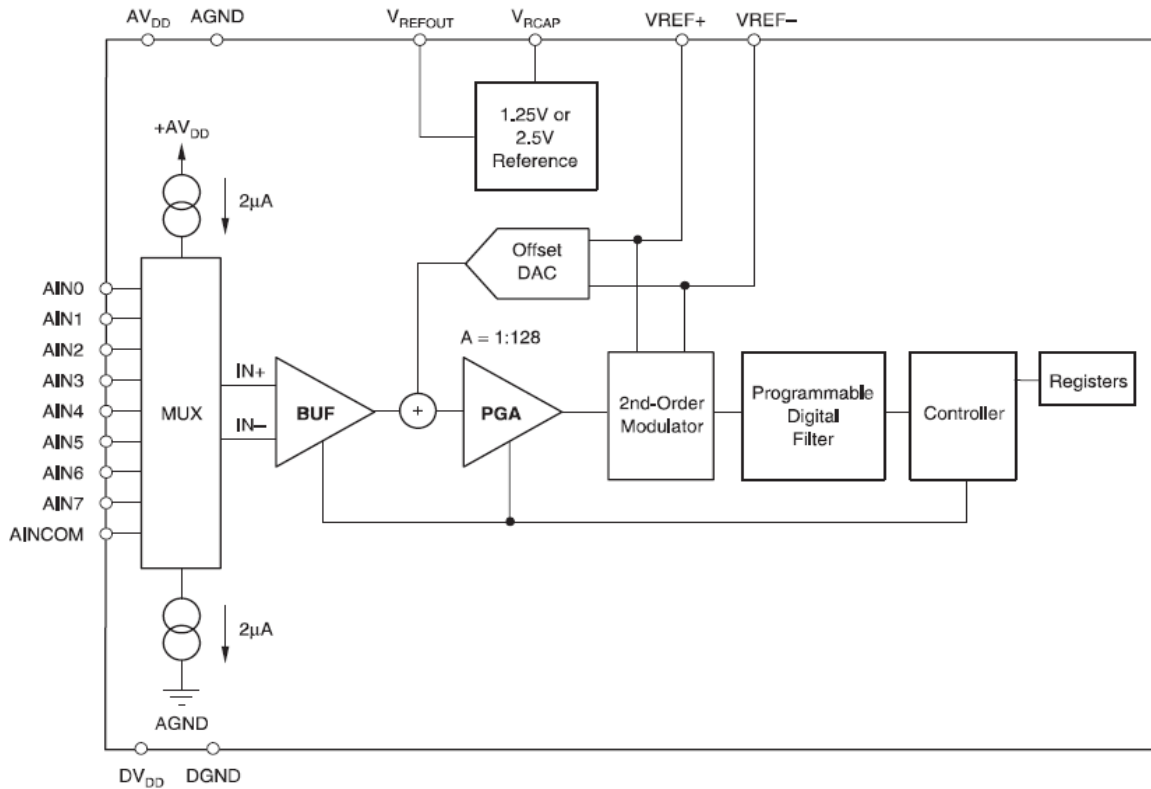


Figura 14. Arquitectura del convertidor analógico-digital.

La frecuencia del modulador de este ADC, el cual se trata de un modulador de segundo orden, es derivada de la frecuencia del oscilador patrón, dividida entre el registro ACLK más 1, y todo esto a su vez dividido entre 64. Por lo que la frecuencia analógica de muestreo de datos a la entrada del ADC (F_{ent}), responde a la siguiente formula:

$$F_{ent} = \frac{F_{osc}/(ACLK + 1)}{64}$$

(Ecuación 6.4)

La frecuencia de salida de datos del ADC (F_{sal}) se determina dividiendo la frecuencia analógica de muestreo de datos entre el ratio de diezmo (*decimation*). Por lo que responde a la siguiente fórmula:

$$F_{sal} = \frac{F_{ent}}{decimation}$$

(Ecuación 6.5)

La resolución de este ADC, es decir el valor del bit menos significativo (LSB), responde a las siguientes fórmulas, dependiendo de si hemos configurado el ADC como unipolar (datos digitales sin signo) o como bipolar (datos digitales con signo):

Unipolar:

$$LSB = \frac{V_{ref}/(PGA)}{2^{n^{\circ} \text{ de bits del ADC}}}$$

(Ecuación 6.6)

Bipolar:

$$\text{LSB} = \frac{2 * V_{\text{ref}} / (PGA)}{2^{\text{nº de bits del ADC}}}$$

(Ecuación 6.7)

6.5.3.1. Frecuencias de muestreo de datos de entrada y de salida adoptadas

La mayoría del ruido eléctrico es producido por interferencias de la señal de la red eléctrica, por lo que tiene una frecuencia de 50Hz o 60Hz. Además también existirá ruido proveniente de las vibraciones mecánicas de la célula de carga, que típicamente tienen una frecuencia de 20Hz [6].

Ya que el filtro digital del convertidor analógico digital $\Delta\Sigma$ del MSC1210 elimina el ruido eléctrico introducido por los múltiplos de la frecuencia de salida de datos, para eliminar el ruido introducido por las principales fuentes de ruido, es decir para eliminar el ruido eléctrico de 20Hz, 50Hz y 60Hz, se ha seleccionado como frecuencia de salida de datos 10Hz, ya que es múltiplo de 20, 50 y 60.

De acuerdo al manual del microcontrolador, podemos lograr un mayor número de bits efectivos del ADC (ENOB) conforme mayor sea el valor del ratio de diezmo (*decimation*), por lo que como ratio de diezmo he seleccionado el valor 1571.

Dado un ratio de diezmo de 1571 y una frecuencia de salida de datos de 10Hz, la frecuencia de muestreo de datos de entrada será:

$$F_{\text{sal}} = \frac{F_{\text{ent}}}{\text{decimation}} \rightarrow F_{\text{ent}} = F_{\text{sal}} * \text{decimation} = 10\text{Hz} * 1571 = 15710\text{Hz}$$

Y dada esa frecuencia de entrada de datos, y teniendo en cuenta que el oscilador patrón que se va a utilizar para que funcione como frecuencia de reloj será de 11.0592MHz, el valor del registro ACLK será:

$$F_{\text{ent}} = \frac{F_{\text{osc}} / (\text{ACLK} + 1)}{64} \rightarrow \text{ACLK} = \frac{F_{\text{osc}}}{64 * F_{\text{ent}}} - 1 = \frac{11.0592\text{MHz}}{64 * 15710\text{Hz}} - 1 = 9.9993 \approx 10$$

6.5.3.2. Resolución adoptada

Teniendo en cuenta que a la entrada del microcontrolador tendremos un valor comprendido entre 0 y 856.08mV, y que se va a configurar el conversor analógico digital en modo unipolar, para obtener el doble de resolución que se tendría en modo

bipolar, el rango dinámico de tensiones en el que trabaja el ADC queda determinado por la siguiente fórmula:

$$\text{Rango dinámico} = \frac{V_{\text{ref}}}{\text{PGA}}$$

(Ecuación 6.8)

Por lo que, los valores del rango dinámico para los distintos valores de la ganancia interna programable (que pueden ser 1, 2, 4, 8, 16, 32, 64 y 128), se recogen en la siguiente tabla, teniendo en cuenta que para el conversor analógico digital se utilizará como tensión de referencia 4.096V. Posteriormente se explicará por qué se ha seleccionado esta tensión de referencia.

PGA	Rango dinámico (V)
1	4.096
2	2.048
4	1.024
8	0.512
16	0.256
32	0.128
64	0.064
128	0.032

Como se ve en la tabla, a partir de una ganancia de 8 el rango dinámico del ADC es demasiado pequeño para que el conversor pueda reconocer todos los valores que pueden entrar al conversor sin saturar; pero también hay que tener en cuenta que los 856.08mV que proporciona como máximo la celda de carga (correspondientes a 300kg), no se llegarán a tener a la entrada del ADC, ya la balanza se ha diseñado para pesar como máximo 150kg, así que tendríamos a la entrada del ADC como máximo la mitad de esos 856.08mV, que son 428.04mV. Por lo que se podría usar la ganancia de 8, que es la se ha utilizado.

Configurando la ganancia programable a 8 se ha obtenido, usando la fórmula para calcular el valor en voltios al que equivale el bit menos significativo del ADC configurado en modo unipolar (Ecuación 6.6), el siguiente resultado:

$$\text{LSB} = \frac{V_{\text{ref}} / (\text{PGA})}{2^{\text{nº de bits del ADC}}} = \frac{4.096\text{V}/8}{2^{24}} = 3.052 \cdot 10^{(-8)}\text{V}$$

Sustituyendo este valor de tensión en la fórmula para calcular el peso equivalente a la tensión que se encuentra a la entrada del microcontrolador (Ecuación 5.1 teniendo en cuenta que el *offset* se elimina automáticamente por programa), se obtiene el siguiente resultado:

$$\text{Peso(kg)} = \frac{V_o \cdot P_{\text{máx}}}{S_{\text{cc}} \cdot V_{\text{exc}}} : G = \frac{3.052 \cdot 10^{(-8)}\text{V} \cdot 300\text{kg}}{2\text{mV/V} \cdot 4.096\text{V}} : 104.50 = 0.001118\text{kg} = 1,118\text{g}$$

Por lo que la resolución de la balanza sería de 1.118 gramos, pero esta resolución presenta un problema y es que según el manual del MSC1210 con el ratio de diezmo

utilizado (1571) el número efectivo de bits (ENOB) sería de 21 bits, como podemos ver en la figura 15 sacada del manual [7]. Si el buffer del ADC está habilitado habría más ruido eléctrico, ya que con él habilitado aumenta la relación señal/ruido, por lo que el número efectivo de bits sería algo menor.

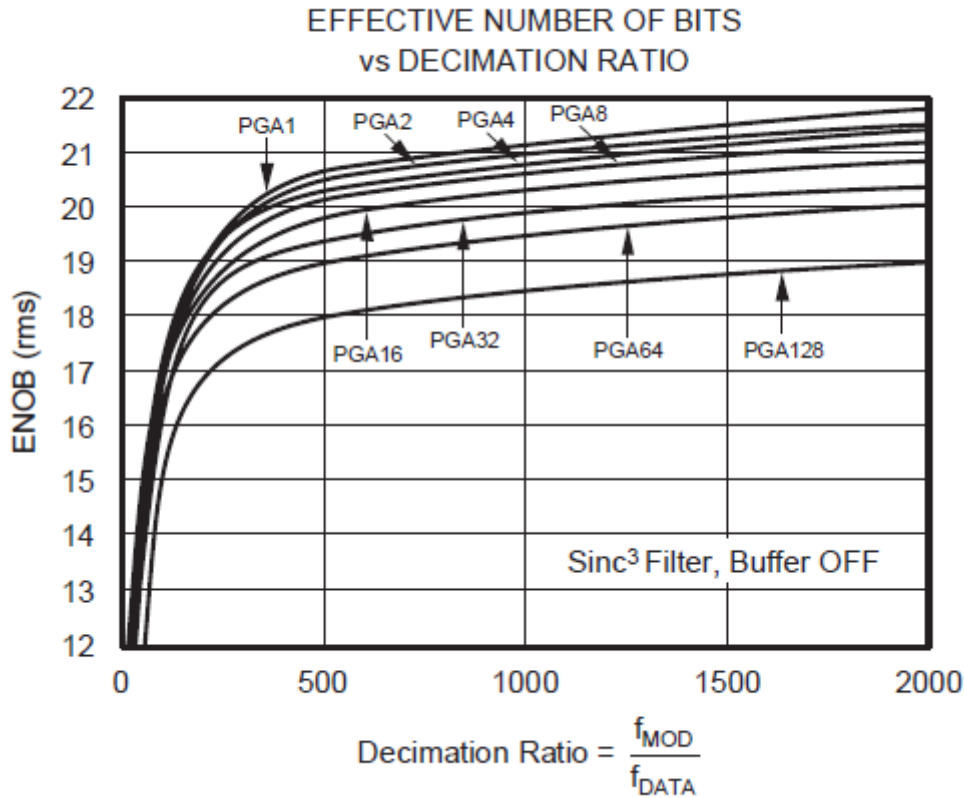


Figura 15. Número efectivo de bits respecto al ratio de diezmo.

Ya que hemos habilitado el buffer como se explicará más adelante, por seguridad, hemos tomado como número efectivo de bits 20. Esto significa que los 4 bits menos significativos se pierden por el ruido eléctrico; es decir, hay un ruido eléctrico de $2^4 = 32\text{LSB}$. Por tanto, sólo se usarán los 20 bits más significativos del convertidor, por lo que la resolución será:

$$\text{LSB} = \frac{4.096\text{V}/8}{2^{20}} = 4.883 \cdot 10^{(-7)}\text{V}$$

$$\text{Peso(kg)} = \frac{4.883 \cdot 10^{(-7)}\text{V} \cdot 300\text{kg}}{2\text{mV/V} \cdot 4.096\text{V}} : 104.50 = 0.017881\text{kg} = 17.881\text{g}$$

Por lo que al final se tendrá una resolución de 17.881 gramos, que cumple con el objetivo marcado al principio, que eran al menos 100 gramos de resolución.

6.5.3.3. Registros de control del ADC del MSC1210

El MSC1210 tiene los siguientes registros de control [1] [8] para la configuración del convertidor analógico digital $\Delta\Sigma$:

- **ADMUX. Registro del multiplexor del ADC.**

	7	6	5	4	3	2	1	0	Reset value
SFR D7h	INP3	INP2	INP1	INP0	INN3	INN2	INN1	INN0	01h

Los cuatro bits superiores (INP0, INP1, INP2 y INP3) seleccionan el pin del multiplexor que se utilizará como señal de entrada positiva.

INP3	INP2	INP1	INP0	Entrada positiva
0	0	0	0	AIN0
0	0	0	1	AIN1
0	0	1	0	AIN2
0	0	1	1	AIN3
0	1	0	0	AIN4
0	1	0	1	AIN5
0	1	1	0	AIN6
0	1	1	1	AIN7
1	0	0	0	AINCON
1	1	1	1	Sensor de Temp.

Los cuatro bits inferiores (INN0, INN1, INN2 y INN3) seleccionan el pin que se utilizará como señal de entrada negativa.

INN3	INN2	INN1	INN0	Entrada positiva
0	0	0	0	AIN0
0	0	0	1	AIN1
0	0	1	0	AIN2
0	0	1	1	AIN3
0	1	0	0	AIN4
0	1	0	1	AIN5
0	1	1	0	AIN6
0	1	1	1	AIN7
1	0	0	0	AINCON
1	1	1	1	Sensor de Temp.

Hemos seleccionado como entrada positiva el pin 1 del multiplexor (AIN1), y como entrada negativa el pin 0 (AIN0).

La configuración de los 8 bits del registro ADMUX a 1, harán que funcione como un sensor de temperatura, desconectando todas las entradas del multiplexor, y conectando las entradas del conversor a dos diodos de unión por los que circulan diferentes

corrientes. La diferencia de potencial entre ambos variará linealmente con la temperatura, proporcionando así un sensor lineal de temperatura.

Por tanto, el registro ADMUX se ha configurado con el valor 10h.

- Registro 0 de control del ADC.

	7	6	5	4	3	2	1	0	Reset value
SFR DCh		BOD	EVREF	VREFH	EBUF	PGA2	PGA1	PGA0	30h

El bit 6 del registro de control ADCON0 (BOD) sirve para habilitar un detector de circuito abierto en los canales de entrada del ADC.

BOD= 0 → Detector de circuito abierto deshabilitado.

BOD= 1 → Detector de circuito abierto habilitado.

Como en la balanza no se requiere detectar si se produce una situación de circuito abierto en los canales de entrada del ADC, lo hemos dejado deshabilitado.

El bit 5 (EVREF) sirve para indicar si la tensión de referencia interna es usada o no por el ADC. Se configura sin referencia, cuando esta se aplica mediante un circuito de referencia externo, porque se necesita un valor distinto a los que puede proporcionar la referencia interna, que son 1.25V y 2.5V.

EVREF=0 → Tensión de referencia interna activada.

EVREF=1 → Tensión de referencia interna desactivada.

Como tensión de referencia hemos utilizado la misma tensión que alimenta la celda carga (los 4.096V generados por el circuito de referencia de tensión), para que las oscilaciones que se puedan producir en la tensión de salida de la célula de carga, debidas a oscilaciones en la tensión de alimentación de la célula de carga, sean compensadas por el ADC. Por lo tanto, como se va a usar una tensión de referencia externa hemos dejado desactivada la tensión interna de referencia.

El bit 4 (VREFH) sirve para seleccionar la tensión interna de referencia como 1.25V o 2.5V.

VREFH=0 → Tensión interna de referencia es de 1.25V.

VREFH=1 → Tensión interna de referencia es de 2.5V.

Como vamos a utilizar una tensión de referencia externa nos es indiferente el valor de este bit.

El bit 3 (EBUF) sirve para habilitar un buffer de entrada que provee una mayor impedancia de entrada, pero limita el rango de tensión de entrada y disipa más calor.

EBUF=0 → Buffer deshabilitado.

EBUF=1 → Buffer habilitado.

El filtro R-C que usamos en la entrada del ADC, puede provocar una medida inesperada debida a un *offset* en la medición. El problema principal con este efecto es que el proceso de muestreo a la entrada tira una pequeña cantidad de la carga del filtro capacitivo de entrada, la cual debe ser repuesta antes de la siguiente muestra. Por esto, hemos dejado el buffer habilitado, aunque aumente la relación señal/ruido.

Los tres bits inferiores (PGA0, PGA1, PGA2) sirven para configurar la ganancia de la PGA desde 1 a 128.

PGA2	PGA1	PGA0	Ganancia
0	0	0	1 (por defecto)
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Hemos utilizado una ganancia de 8 para obtener la resolución calculada en el apartado 6.5.3.2.

Por tanto, el registro ADCON0 se ha configurado con el valor 0Bh.

- ADCON1. Registro 1 de configuración del ADC.

	7	6	5	4	3	2	1	0	Reset value
SFR DDh		POL	SM1	SM0		CAL2	CAL1	CAL0	00h

El bit 6 del registro de control ADCON1 (POL) sirve para configurar la polaridad del resultado del ADC como unipolar o bipolar.

POL=0 → Unipolar.

POL=1 → Bipolar.

Hemos seleccionado la polaridad del resultado del ADC como unipolar, ya que la resolución es el doble que si se configura como bipolar.

Los bits 4 y 5 (SM0 y SM1) se utilizan para configurar el modo del filtro empleado.

SM1	SM0	Modo
0	0	<i>Auto</i>
0	1	<i>Fast Settling Filter</i>
1	0	<i>Sinc² Filter</i>
1	1	<i>Sinc³ Filter</i>

Hemos seleccionado el modo *Sinc³ Filter*, ya que es con el que obtenemos un menor ruido a la salida, que es lo que buscamos para realizar mediciones del pesaje con la mayor precisión posible. Aunque este filtro tiene el inconveniente de ser el más lento,

pero como para la balanza no se requiere de una respuesta rápida, esto no es un problema.

Los tres bits inferiores (CAL0, CAL1 y CAL2) se utilizan para realizar una calibración del ADC interna (*Self-Calibration*) o externa (*System Calibration*), de la ganancia y/o del offset.

CAL2	CAL1	CAL0	Modo de calibración
0	0	0	Sin calibración (por defecto)
0	0	1	<i>Self-Calibration</i> de offset y ganancia
0	1	0	<i>Self-Calibration</i> de Offset
0	1	1	<i>Self-Calibration</i> de ganancia
1	0	0	<i>System Calibration</i> de offset
1	0	1	<i>System Calibration</i> de ganancia

Se ha utilizado la calibración *Self-Calibration* de offset y ganancia, para calibrar el offset y la ganancia del ADC, ya que para calibrar la celda de carga se ha usado la curva de calibración mencionada en el apartado 7.2.

Por tanto, el registro ADCON1 se ha configurado con el valor 31h.

- ADCON2. Registro 2 de configuración del ADC.

	7	6	5	4	3	2	1	0	Reset value
SFR DEh	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0	1Bh

El registro ADCON2 se utiliza para configurar los 8 bits menos significativos del ratio de diezmo (*decimation*). Dicho registro se cargará con los 8 bits menos significativos del valor 1571 (0623h en hexadecimal), seleccionado como ratio de diezmo.

Por tanto, el registro ADCON2 se ha configurado con el valor 23h.

- ADCON3. Registro 3 de configuración del ADC.

	7	6	5	4	3	2	1	0	Reset value
SFR DFh						DR10	DR9	DR8	1Bh

Los tres bits menos significativos de ADCON3 se utilizan para configurar los 3 bits más significativos del ratio de diezmo (*decimation*). Dicho registro se cargará con los 3 bits más significativos del valor 1571 (0623h en hexadecimal), seleccionado como ratio de diezmo.

Por tanto, el registro ADCON3 se ha configurado con el valor 06h.

6.5.4 Otros recursos del MSC1210 necesarios

6.5.4.1. Envío de los resultados

Para enviar los resultados del peso sobre la balanza, hemos optado por enviar por la UART0 del MSC1210 el resultado en código ASCII. Para configurar la UART0 se usa el registro SCON0, el cual tiene los siguientes bits:

	7	6	5	4	3	2	1	0	Reset value
SFR 98h	SM0_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TI_0	RI_0	00h

Los bits 5, 6 y 7 (SM0_0, SM1_0 y SM2_0) se utilizan para configurar el modo en el que funcionará el puerto serie 0.

Modo	SM0	SM1	SM2	Función	Longitud	Periodo
0	0	0	0	Síncrono	8 bits	12 p _{CLK}
0	0	0	1	Síncrono	8 bits	4 p _{CLK}
1	0	0	1	Asíncrono	10 bits	Ec. del Temp1 o 2
1	0	1	1	Bit de stop requerido	10 bits	Ec. Del Temp1
2	1	0	0	Asíncrono	11 bits	64 p _{CLK} (SMOD=0)
2	1	0	1	Asíncrono con comunicación multiprocesador	11 bits	32 p _{CLK} (SMOD=1) 64 p _{CLK} (SMOD=0) 32 p _{CLK} (SMOD=1)
3	1	1	0	Asíncrono	11 bits	Ec. Del Temp1 o 2
3	1	1	1	Asíncrono con comunicación multiprocesador	11 bits	Ec. Del Temp1 o 2

En nuestro caso hemos optado por usar el modo 1, que permite una comunicación asíncrona completamente bidireccional con 10 bits de longitud, de los cuales el primero indica el inicio de la comunicación, después se envían los 8 bits correspondientes al dato en código ASCII, empezando por el bit menos significativo, y por último se envía el bit de stop.

Para establecer la velocidad de comunicación hemos utilizado el temporizador 1, para el cual se tiene la siguiente fórmula para calcular el valor con el que se recargarlo para obtener una determinada velocidad de comunicación (Baudrate):

$$TH1 = 256 - \frac{2^{\text{SMOD}} * F_{\text{osc}}}{384 * \text{Baudrate}}$$

Siendo “SMOD” el bit octavo del registro PCON, el cual nos permite doblar la velocidad de comunicación y “Fosc” la frecuencia de oscilación del reloj patrón.

Para una velocidad de comunicación de 9600 bps (bits por segundo), poniendo SMOD a 1, para duplicar la velocidad de comunicación, y teniendo en cuenta que la frecuencia del oscilador patrón es 11.0592MHz, habrá que recargar el temporizador 1 con el siguiente valor:

$$TH1 = 256 - \frac{2^1 * 11.0592\text{MHz}}{384 * 9600\text{bps}} = 250 = \text{FAh}$$

El bit 4 (REN_0) habilita o deshabilita la recepción.

REN_0=0 → Recepción deshabilitada.

REN_0=1 → Recepción habilitada.

Como sólo necesitamos enviar el resultado del pesaje, este bit se ha configurado a 0, ya que no vamos a realizar ninguna recepción.

Los bits 2 y 3 (RB8_0 y TB8_0) definen el estado del noveno bit, para la recepción y la transmisión, respectivamente, en los modos 2 y 3. Como se va a usar el modo 1 es indiferente el valor de estos bits.

Los bits 0 y 1 (RI_0 y TI_0) indican cuando se pueden realizar una recepción y una transmisión, respectivamente. Se activarán automáticamente cuando termine la recepción o transmisión que está en proceso para impedir que se pueda realizar otra, antes de que termine la anterior.

RI_0=0 → No se puede realizar una recepción.

RI_0=1 → Se puede realizar una recepción

TI_0=0 → No se puede realizar una transmisión.

TI_0=1 → Se puede realizar una transmisión.

Como en nuestro caso no se va a recibir nada por el puerto serie, RI_0 se ha configurado a 0, y para que se puedan realizar las transmisiones, para enviar el valor del pesaje, TI_0 se configurado a 1.

Por tanto, el registro SCON0 se ha configurado con el valor 42h.

Para configurar el temporizador 1 que genera la velocidad de comunicación de la UART, como se ha comentado antes, hay que configurar el registro TMOD. Los bits de configuración de TMOD se muestran a continuación:

	7	6	5	4	3	2	1	0	Reset value
	Temporizador 1				Temporizador 0				
SFR 89h	GATE	$\overline{C/T}$	M1	M0	GATE	$\overline{C/T}$	M1	M0	00h

Los 4 bits superiores permiten configurar el temporizador 1, y los 4 bits inferiores el temporizador 0.

Los bits 3 y 7 (GATE) sirven para habilitar o deshabilitar que el temporizador se incremente dependiendo del valor de $\overline{INT1}$.

GATE=0 → El temporizador se incrementa cuando $TRX=1$, sin importar el valor de $\overline{INT1}$.

GATE=1 → El temporizador se incrementa cuando $TRX=1$ y $\overline{INT1}=1$.

Estos dos bits se han configurado a 0, ya que no es necesario poner la condición de que el temporizador debe incrementarse usando $\overline{INT1}$.

Los bits 2 y 6 (C/\overline{T}) determinan si el temporizador funcionará como contador de eventos o como contador de pulsos de reloj.

$C/\overline{T}=0$ → El temporizador se incrementa mediante los pulsos de reloj.

$C/\overline{T}=1$ → El temporizador se incrementa mediante los pulsos recibidos en T0 o T1, dependiendo de si usamos el temporizador 0 o 1.

Ya que el temporizador 1 debe funcionar como contador de pulsos la señal de reloj, por lo que dicho bit se ha puesto a 0.

Los bits 0, 1, 4 y 5, determinan el modo de funcionamiento de los temporizadores 0 y 1.

M1	M0	Modo
0	0	0: contador de 8 bits con 5 bis de preescala
0	1	1: contador de 16 bits
1	0	2: contador de 8 bits con autorecarga
1	1	2: contadores de 8 bits

En nuestro caso, el temporizador 1 se ha configurado en modo 2, como contador de 8 bits con autorecarga, ya que es necesario que el contador se autorecarge para generar la señal de reloj de la UART.

Según lo indicado, el registro TMOD se ha configurado con el valor 20h.

6.5.4.2. Espera de un determinado tiempo

Para inicializar el visualizador LCD y para filtrar los rebotes mecánicos en la pulsación de las teclas es necesario esperar un determinado tiempo, del orden de los milisegundos. Se ha optado por utilizar un recurso del MSC1210 que permite contar milisegundos, el *Miliseconds Timer*, el cual se habilita configurando el bit 1 (PDST) del registro PDCON.

Dicho temporizador producirá una interrupción cada $MSINT+1ms$, siendo MSINT un registro que se puede configurar, habiendo configurado previamente los registros MSECH (los 8 bits más significativos de MSEC) y MSECL (los 8 bits menos

significativos de MSEC) para contar un milisegundo, siendo:

$$MSEC = \frac{f_{osc}}{1000} = \frac{11.0592M}{1000} = 11059.2 \approx 11059$$

Dicha interrupción pondrá a 1 el bit 4 (EMSEC) del registro AIE.

6.5.4.3. Misceláneos

El registro de control CKCON permite configurar la duración de las operaciones de transferencia externa (MOVX) y el valor por el que se divide la frecuencia de reloj, para que se incremente el valor del temporizador cuando se utiliza como contador de tiempo. El registro CKCON tiene los siguientes bits:

	7	6	5	4	3	2	1	0	Reset value
SFR 8Eh			T2M	T1M	T0M	MD2	MD1	MD0	01h

Los bits 3, 4 y 5 (T0M, T1M y T2M) permiten configurar el valor por el que se divide la frecuencia de reloj. Si se configura con el valor 4, el incremento del temporizador corresponderá con los ciclos máquina del MSC1210, y si se configura con el valor 12 corresponderá a los ciclos máquina del 8051.

T0M=0 → El temporizador 0 utiliza la frecuencia de reloj dividida entre 12.

T0M=1 → El temporizador 0 utiliza la frecuencia de reloj dividida entre 4.

T1M=0 → El temporizador 1 utiliza la frecuencia de reloj dividida entre 12.

T1M=1 → El temporizador 1 utiliza la frecuencia de reloj dividida entre 4.

T2M=0 → El temporizador 2 utiliza la frecuencia de reloj dividida entre 12.

T2M=1 → El temporizador 2 utiliza la frecuencia de reloj dividida entre 4.

Como en esta aplicación no se va a realizar ninguna temporización usando los timers 0, 1 o 2, se han dejado con sus valores por defecto, que utilizan la frecuencia de reloj dividida entre 12.

Los bits 0, 1 y 2 (MD0, MD1 Y MD2) permiten configurar la duración de las operaciones de transferencia externa, de acuerdo a la siguiente tabla:

MD 2	MD 1	MD 0	STRETCH VALUE	MOVX DURATION	\overline{RD} or \overline{WR} STROBE WIDTH (SYS CLKs)	\overline{RD} or \overline{WR} STROBE WIDTH (μs) at 12MHz
0	0	0	0	2 ciclos de instrucciones	2	0.167
0	0	1	1	3 ciclos de instrucciones	4	0.333
0	1	0	2	4 ciclos de instrucciones	8	0.667
0	1	1	3	5 ciclos de instrucciones	12	1.000
1	0	0	4	6 ciclos de instrucciones	16	1.333
1	0	1	5	7 ciclos de instrucciones	20	1.667
1	1	0	6	8 ciclos de instrucciones	24	2.000
1	1	1	7	9 ciclos de instrucciones	28	2.333

Como la memoria es lo suficientemente rápida, no es necesario ensanchar los pulsos \overline{RD} o \overline{WR} , por lo que se ha configurado la duración de las operaciones de transferencia externa en 2 ciclos de instrucciones.

Por tanto, el registro CKCON se ha configurado con el valor 00h.

Para poder usar el ADC es necesario activarlo, poniendo a 0 el bit 3 (PDADC) del registro PDCON, y activar las interrupciones del ADC, poniendo a 1 el bit 5 (EADC) del registro AIE.

Por último también es necesario configurar el registro P3DDRL, el cual permite configurar los 4 bits inferiores del puerto 3, cuyos bits son:

	7	6	5	4	3	2	1	0	Reset value
SFR B3h	P33H	P33L	P32H	P32L	P31H	P31L	P30H	P30L	00h

Los bits 6 y 7 (P33L y P33H) permiten configurar el pin 3 del puerto 3, de acuerdo a la siguiente tabla:

P33H	P33L	
0	0	Standard 8051 (Pull-Up)
0	1	CMOS Output
1	0	Open Drain Output
1	1	Input

Estos bits los dejaremos con su valor por defecto, ya que no es necesario usar este pin.

Los bits 4 y 5 (P32L y P32H) permiten configurar el pin 2 del puerto 3, de acuerdo a la siguiente tabla:

P32H	P32L	
0	0	Standard 8051 (Pull-Up)
0	1	CMOS Output
1	0	Open Drain Output
1	1	Input

Estos bits los dejaremos con su valor por defecto, ya que no es necesario usar este pin.

Los bits 2 y 3 (P31L y P31H) permiten configurar el pin 1 del puerto 3, de acuerdo a la siguiente tabla:

P31H	P31L	
0	0	Standard 8051 (Pull-Up)
0	1	CMOS Output
1	0	Open Drain Output
1	1	Input

En este caso, el pin 1 del puerto 3 se utiliza para la transmisión de la UART0, por lo que se ha configurado como salida de tipo CMOS.

Los bits 6 y 7 (P30L y P30H) permiten configurar el pin 0 del puerto 3, de acuerdo a la siguiente tabla:

P30H	P30L	
0	0	Standard 8051 (Pull-Up)
0	1	CMOS Output
1	0	Open Drain Output
1	1	Input

Dicho pin se corresponde con la recepción de la UART0, pero como no se ha habilitado la recepción de datos, se ha dejado con su valor por defecto.

6.6. Visualizador LCD

Para mostrar los resultados del peso colocado sobre la celda de carga hemos utilizado un visualizador LCD alfanumérico de 16 columnas y 2 filas, ya que es suficiente para mostrar el peso y algún que otro mensaje necesario para configurar la balanza. El modelo de pantalla LCD que hemos utilizado es el *LCM-S01602DSF/A* [13], cuyas características son las siguientes:

Fabricante	Lumex Opto/Components Inc.
Número de caracteres	32
Formato de pantalla	16x2
Formato de caracteres	5x8 puntos
Tipo de pantalla	STN- Neumático súper trenzado
Modo de pantalla	Transreflectivo
Tamaño de caracteres	5.56x2.96mm
L x A x H de contorno	80.00x36.00x12.70mm
Área visible	66.00x16.00mm
Luz de fondo	LED- Amarillo/Verde
Voltaje de alimentación	5V
Tamaño de punto	0.56x0.66mm
Interfaz	Paralelo
Temperatura de operación	0 °C ~ 50 °C
Color de fondo	Verde

La función de los pines de este dispositivo se ven en la siguiente tabla:

Número del pin	Señal del LCD	Descripción
1	VSS	Masa de la lógica y del contraste
2	VDD	Alimentación de la lógica
3	V0	Ajuste del contraste
4	RS	Bit de selección del tipo de instrucción (dato o comando)
5	R/W	Bit de selección del tipo de ciclo (lectura o escritura)
6	E	Bit de habilitación de la instrucción

7	DB0	Bit 0 del bus de datos del LCD
8	DB1	Bit 1 del bus de datos del LCD
9	DB2	Bit 2 del bus de datos del LCD
10	DB3	Bit 3 del bus de datos del LCD
11	DB4	Bit 4 del bus de datos del LCD
12	DB5	Bit 5 del bus de datos del LCD
13	DB6	Bit 6 del bus de datos del LCD
14	DB7	Bit 7 del bus de datos del LCD
15	LED+	Ánodo del Backlight
16	LED-	Cátodo del Backlight

La placa UCOADP incorpora un conector IDE de 2x8 pines para conectar directamente un visualizador alfanumérico. Esta placa está diseñada para que el control del visualizador LCD se realice a través del CPLD, ya que se han implementado en él los registros de control, estado, escritura y lectura. Se ha diseñado así, para dejar olibres los puertos del microcontrolador, que se tendrían que utilizar para controlar el visualizador, y porque los visualizadores LCD son periféricos lentos, por lo que si se conectan a los buses del microcontrolador, que son más rápidos, existiría la necesidad de realizar las transmisiones de manera asíncrona, ralentizando el sistema.

La placa UCOADP está diseñada para que el visualizador LCD se alimente a través del driver *MOSFET MAX4420ESA*, el cual permite habilitar y deshabilitar la alimentación por programa modificando la señal PWRLCD, que sale del CPLD, actuando sobre un determinado registro del mismo.

Para realizar un ciclo de lectura o un ciclo de escritura se debe seguir el siguiente cronograma [3], en el que se muestra en rojo lo específico a un ciclo de lectura y en verde lo de uno de escritura.

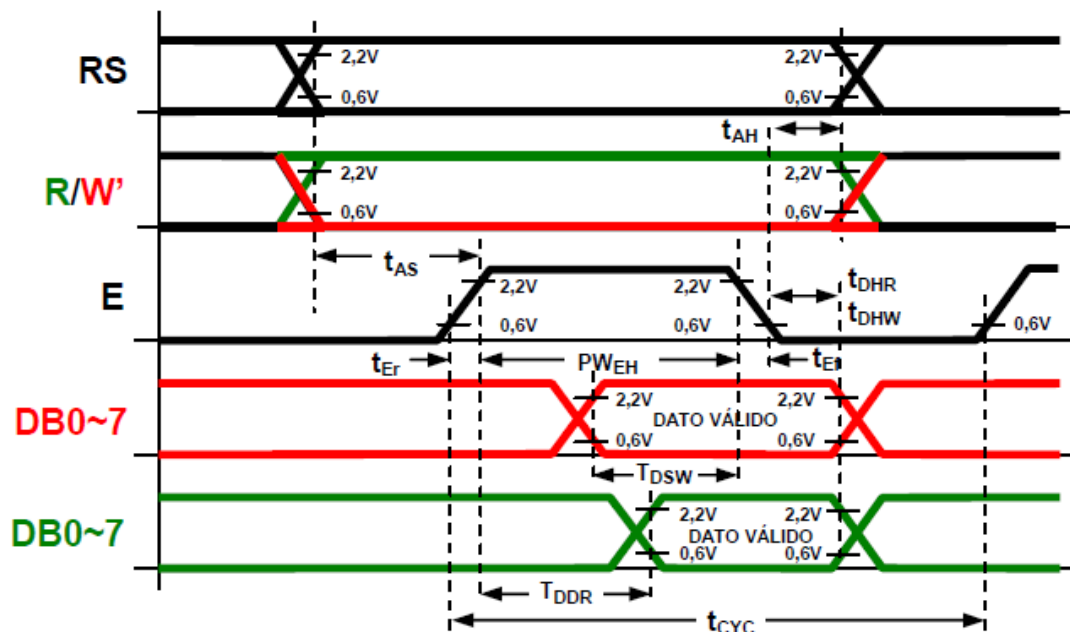


Figura 16. Cronograma de un ciclo de lectura y escritura.

Se deben cumplir con los tiempos que aparecen en la siguiente tabla:

Parámetro	Mínimo	Máximo	Unidades
Tiempo del ciclo de habilitación (t_{CYC})	1000		ns
Anchura del pulso de habilitación (PW_{EH})	450		ns
Tiempo de subida/ bajada de la habilitación (t_{Er}/ t_{Ef})		25	Ns
Tiempo de asentamiento de la dirección (t_{AS})	140		ns
Tiempo de retención de la dirección (t_{AH})	10		ns
Tiempo previo del dato asentado en escritura (t_{DSW})	195		ns
Tiempo de asentamiento del dato en lectura (t_{DDR})		320	ns
Tiempo de retención del dato en escritura/lectura (t_{DHW}/ T_{DHR})	10		ns

Para realizar una escritura en el visualizador LCD hay que seguir los siguientes pasos:

1. Poner a 0 el bit R/W, para indicar que se va a realizar un ciclo de escritura, y poner el bit RS a 0 o a 1 dependiendo de si se va a escribir en un registro de control (para mandar una orden al visualizador) o en uno de datos (para enviar un carácter al visualizador).
2. Esperar el tiempo de establecimiento de la dirección (t_{AS}).
3. Poner a 1 la señal de habilitación (E).
4. Escribir el valor deseado en el canal de datos del visualizador.
5. Esperar el tiempo de establecimiento del dato en escritura (t_{DSW}).
6. Poner a 0 la señal de habilitación (E), para terminar el ciclo de escritura y escribir el dato en el LCD.

Mientras que para realizar un ciclo de lectura es necesario:

1. Poner a 1 el bit R/W, para indicar que se va a realizar un ciclo de lectura, y poner el bit RS a 0 o a 1 dependiendo de si se va a leer un registro de control (para leer el registro de control) o uno de datos (para leer un el registro de estado del visualizador).
2. Esperar el tiempo de establecimiento de la dirección (t_{AS}).
3. Poner a 1 la señal de habilitación (E).
4. Esperar el tiempo de establecimiento del dato en lectura (t_{DDR}).
5. Leer el dato del bus de datos del visualizador.
6. Poner a 0 la señal de habilitación (E), para terminar el ciclo de lectura.

Para realizar todo esto, en el CPLD se han implementado 5 registros. El primero de ellos se encarga de realizar una lectura del registro de control del visualizador, poniendo R/W a 1 y RS a 0, esperando t_{AS} , y poniendo E a 1. El segundo registro se encarga de realizar una lectura del registro de estado del visualizador, poniendo R/W y RS a 1, esperando t_{AS} , y poniendo E a 1. El tercer registro se encarga de escribir un dato en la memoria del LCD, poniendo R/W a 0 y RS a 1, esperando t_{AS} , y poniendo E a 1. El cuarto registro se encarga de escribir una orden, poniendo R/W y RS a 0, esperando t_{AS} , y poniendo E a 1. Por último el quinto registro se encarga de poner a 0 el bit E, que es cuando en realidad se ejecuta la instrucción.

Para leer el registro de estado del visualizador, y ver si el LCD está o no ocupado, realizamos un acceso al segundo registro, y a continuación realizamos una lectura del quinto registro.

Para escribir un dato en la memoria del visualizador, se escribirá dicho dato en el tercer registro, y a continuación se accederá al quinto registro. El dato se escribe en el primer acceso para cumplir con t_{DSW} .

Para enviar una orden al visualizador, se escribirá dicha orden en el cuarto registro, y a continuación se accederá al quinto registro. La orden se escribe en el primer acceso para cumplir con t_{DSW} .

6.7. Teclado

Para poder configurar la balanza se ha utilizado un teclado. La placa UCOADP tiene un conector tipo IDE de 2x5 pines para conectar un teclado matricial de 4 filas y 4 columnas, que, al igual que el visualizador LCD, está controlado completamente por el CPLD; se ha decidido aprovecharlo, aunque en vez de usar un teclado de 4 filas y 4 columnas se ha utilizado uno de 4 filas y 3 columnas tipo teléfono, que se ha adaptado al conector.

La forma del teclado y la función de sus pines, se ven en la siguiente figura:

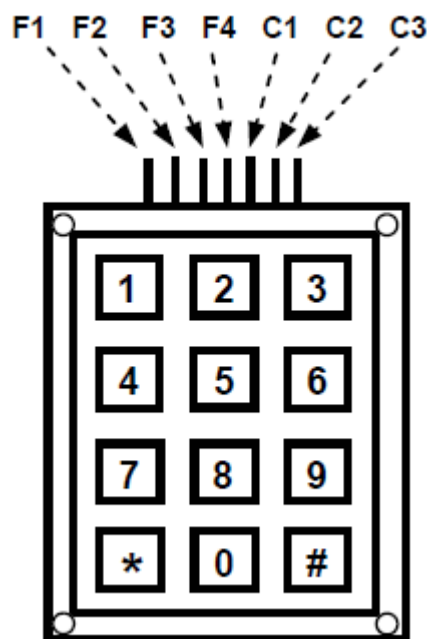


Figura 17. Teclado tipo teléfono.

El control de este teclado se realiza a través de un registro del CPLD implementado para ello; un ciclo de escritura en este registro supondrá escribir los 4 bits inferiores en los pines de las filas del teclado conectado al CPLD, mientras que un ciclo de lectura supondrá leer el estado de los 4 bits correspondientes a las columnas del teclado.

En el proyecto de la placa UCOADP, para filtrar los rebotes mecánicos producidos en la pulsación de las teclas, se incluyen unos filtros consistentes en un condensador cerámico por cada entrada asociada a una columna del teclado matricial y una resistencia de pull-up en serie con el condensador. La resistencia de pull-up era

necesaria previamente, ya que las filas no dan el nivel lógico alto porque están configuradas como drenador abierto. Para que en las columnas se tenga un nivel lógico alto, cuando no se pulsa la tecla correspondiente, debe incluirse esta resistencia de puesta en alto.

Además de este filtro, por si fuera insuficiente, hemos realizado un filtro software de 10ms después de pulsar y a la relajación de cada tecla, para asegurarnos de que no se trate de un espurio.

De las 12 teclas del teclado sólo hemos utilizado las 3 primeras, las cuales las hemos utilizado para lo siguiente:

- Tecla 1. Se utiliza para poner a 0 la balanza.
- Tecla 2. Se utiliza para realizar una tara de la balanza.
- Tecla 3. Se utiliza para habilitar y deshabilitar el envío por el puerto serie de los resultados. Por defecto la balanza está configurada para no enviar los resultados por el puerto serie; para enviar los resultados habría que pulsar esta tecla, y para volver a inhabilitar la transmisión por el puerto serie habría que volver a pulsarlo. En el LCD se muestra si el envío por el puerto serie está o no habilitado.

7. Ajustes de la balanza

7.1. Offset

El puente de la celda de carga, cuando no tiene ningún peso sobre ella, tiene en su salida una determinada diferencia de potencial, cuando teóricamente está diferencia de potencial debería ser cero. A esta diferencia de potencial cuando no hay peso sobre la celda de carga, es a lo que denominamos tensión inicial de *offset*.

Este problema es debido a las ligeras variaciones en las resistencias entre las ramas del puente y la resistencia de los cables. Durante las pruebas realizadas a la balanza, se observó que el *offset* de la celda de carga utilizada en este proyecto es de unos 0.152mV. Para compensar esta tensión inicial de *offset* existen diferentes alternativas [11]:

- 1.- Compensación por software. Consiste en compensar esta tensión inicial mediante software, es decir, a través del programa de la aplicación. Con este método primeramente se toma una medida del peso, cuando la balanza se encuentra vacía, y se guarda en la memoria, para que el resto de medidas se comparen con este peso inicial. A este método también se le denomina auto-cero. Se trata de un método simple, rápido y que no requiere de ajustes manuales. La desventaja de este método es que la tensión inicial de *offset* sigue estando a la salida de la celda carga; si esta tensión es lo suficientemente grande, limitará la ganancia que el amplificador puede aplicar a la tensión de salida, limitando por tanto el rango dinámico de la medida y con ello la resolución de la balanza.
- 2.- Circuito de anulación de *offset*. Consiste en utilizar un potenciómetro a la salida del puente de la celda carga, para poder ajustar manualmente la tensión inicial de *offset* a 0V.
- 3.- Anulación del *offset* mediante un buffer. Consiste en un circuito de anulación que añade una tensión ajustable a la salida del amplificador de instrumentación. Este método al igual que la compensación por software, presenta la ventaja de que no afecta al puente directamente.

La solución adoptada ha sido la compensación por software, ya que disponemos de un microcontrolador para el tratamiento de los datos. Al arrancar el programa se toma una medida, que será almacenada en una variable; posteriormente a las demás medidas se les restará el valor que tenga dicha variable. Además la balanza dispone de una tecla (la tecla 1), para que se vuelva a realizar una medición y volver a cargar la variable con un determinado valor, por si la balanza se desajusta durante su uso por motivos como pueden ser que la balanza se haya iniciado con algún peso sobre ella, un aumento de la temperatura, variación de resistencia de los cables, etc.

7.2. Calibración

La calibración es el conjunto de operaciones con las que se establece, en condiciones específicas, la correspondencia entre los valores indicados en un instrumento, equipo o sistema de medida, y los valores conocidos correspondientes a una magnitud de medida o patrón, asegurando así la trazabilidad de las medidas a las correspondientes unidades básicas y procediendo a su ajuste o expresando esta correspondencia por medio de tablas o curvas de corrección.

El envejecimiento de los componentes, los cambios de temperatura y el estrés mecánico que soportan los equipos deterioran poco a poco sus funciones. Cuando esto sucede, las medidas comienzan a perder confianza. Esta situación puede ser evitada, por medio del proceso de calibración.

La balanza está dentro de los sistemas de instrumentación que funcionan en régimen estático, dado que la masa del objeto pesado permanece constante o varía con una frecuencia no superior a 1 o 2Hz. Las características estáticas quedan determinadas por las características estáticas del sensor, es decir de la celda de carga.

Las características estáticas de la celda de carga se definen a través de una curva de calibración. Para construir esta curva se han usado dos objetos que se encontraban en el laboratorio: una maleta de 24.3kg y una fuente de alimentación de 8.5kg.

El peso de estos objetos ha sido determinado con una báscula de baño con 100 gramos de resolución. No hemos podido disponer de una báscula más precisa, o de pesos patrón, para realizar una mejor calibración, por motivos de presupuesto.

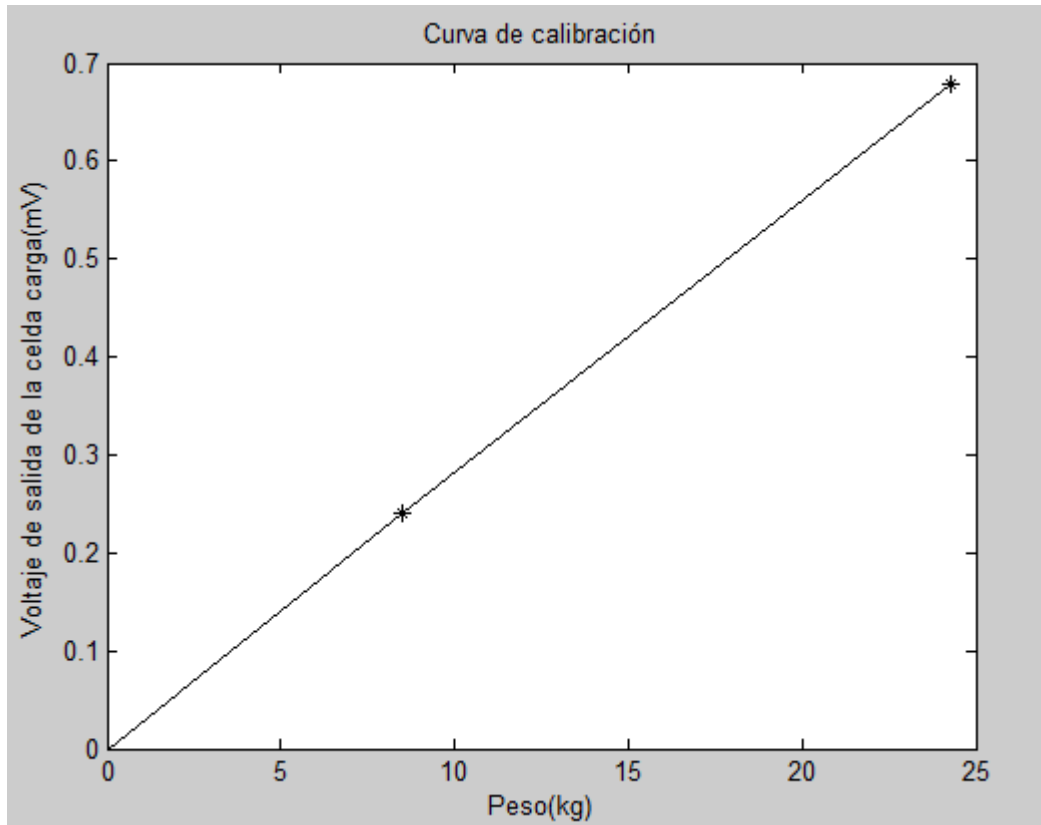


Figura 18. Curva de calibración de la celda de carga.

En la curva de calibración, en la figura 18, se ve como la celda de carga es completamente lineal como garantiza el fabricante. Otra cosa que se puede analizar es si la sensibilidad es la misma que el fabricante nos decía, es decir 2mV/V.

La sensibilidad en un punto de la curva de calibración es la pendiente en ese punto. La pendiente en el punto correspondiente al peso de 24.3kg sería:

$$\text{Sensibilidad} = \frac{0.678\text{mV}}{24.3\text{kg}} = 0.0279\text{mV/kg}$$

Este valor para pasarlo a mV/V es necesario multiplicarlo por el peso máximo de la celda de carga (300kg) y dividirlo entre la tensión de excitación (4.096V). Haciendo esto, dicho valor quedaría:

$$\text{Sensibilidad} = \frac{0.0279\text{mV/kg} * 300\text{kg}}{4.096\text{V}} = 2.0435\text{mV/V}$$

Como se ve, la sensibilidad calculada es distinta a la que el fabricante nos decía. Para calibrar la celda de carga se ha incluido el valor de la sensibilidad calculada, en vez de la que el fabricante nos decía, en el código diseñado, para realizar la conversión de voltios a kilogramos.

También se ha utilizado otro parámetro denominado “*cal*” para que nuestra balanza dé el mismo peso que la báscula que hemos usado para calibrarla. El cómo se ajusta este parámetro se explica en el apartado 9.

8. Descripción de la aplicación software

8.1. Herramienta software utilizada para la programación

Para diseñar el programa de la aplicación, que se ejecutará dentro del microcontrolador para controlar la balanza, se ha utilizado el entorno de desarrollo μ Vision de Keil, más concretamente la versión 5 de este programa.

El entorno de desarrollo μ Vision [9] combina gestión de proyectos, entorno de tiempo de ejecución, edición de código fuente y depuración de programas en un único y potente entorno. μ Vision es fácil de usar y acelera el desarrollo de software embebido. Este entorno presenta dos modos básicos de trabajo:

- Modo principal o de proyecto.
- Modo depuración.

El modo principal o de proyecto está formado por el editor texto, y por el gestor de proyectos y el entorno en tiempo de ejecución.

El editor de texto, nos permite diseñar el código, en lenguaje ensamblador o en C/C++. Dicho editor incluye todas las características estándar de un moderno editor de código fuente (sintaxis resaltada en color, sangrado del texto y fuente de esquematización optimizada para C/C++), y que también está disponible durante la depuración. Este editor tiene la siguiente apariencia:

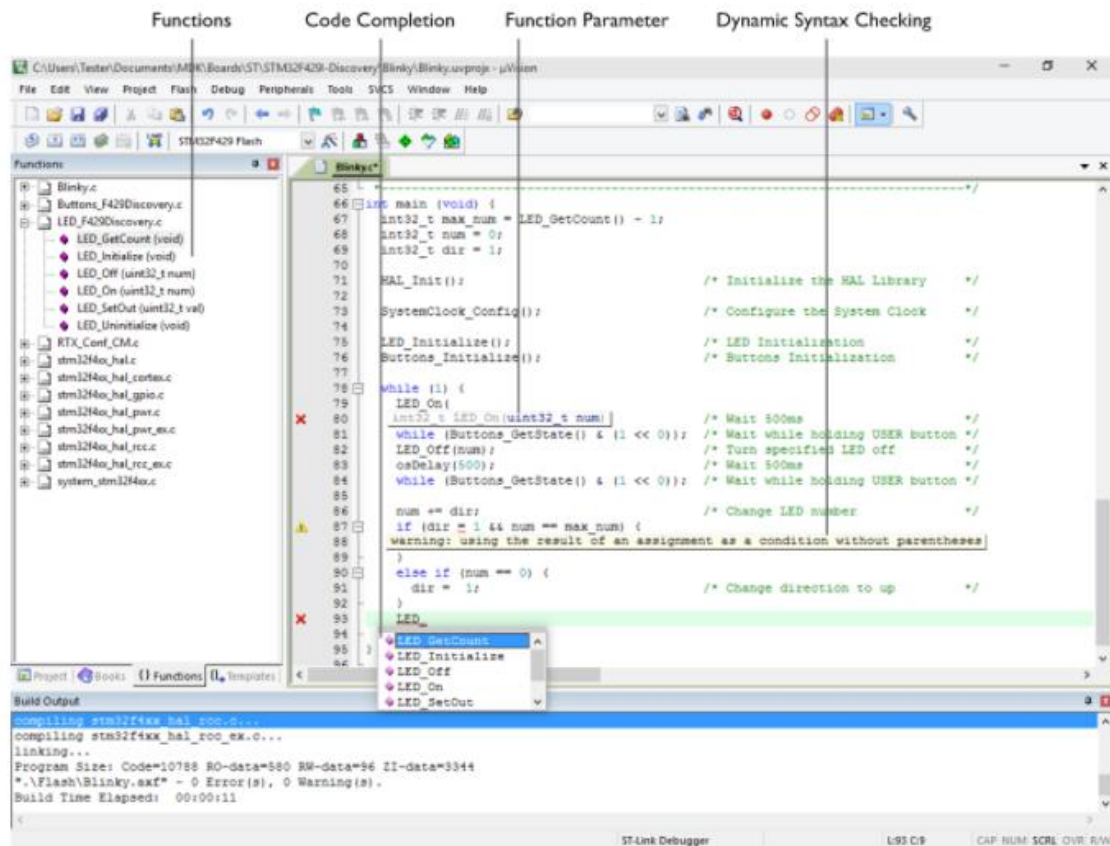


Figura 19. Editor de µVision.

- La ventana de funciones provee un rápido acceso a todas las funciones que hay en cada módulo del código fuente en ensamblador o en lenguaje C/C++.
- La lista de completación de código y la función de información de parámetros ayudan a buscar símbolos, funciones y parámetros.
- El comprobador dinámico de sintaxis valida la sintaxis del programa mientras se escribe, y provee alertas en tiempo real de potenciales violaciones de código antes de la compilación.

Con el gestor de proyectos y el entorno en tiempo de ejecución se crea una aplicación software usando componentes software de pre-compilación y dispositivos de soporte de paquetes software. Los componentes software contienen librerías, módulos fuente, ficheros de configuración, plantillas de código fuente y documentación. El gestor de proyectos y el entorno en tiempo de ejecución tienen la siguiente apariencia:

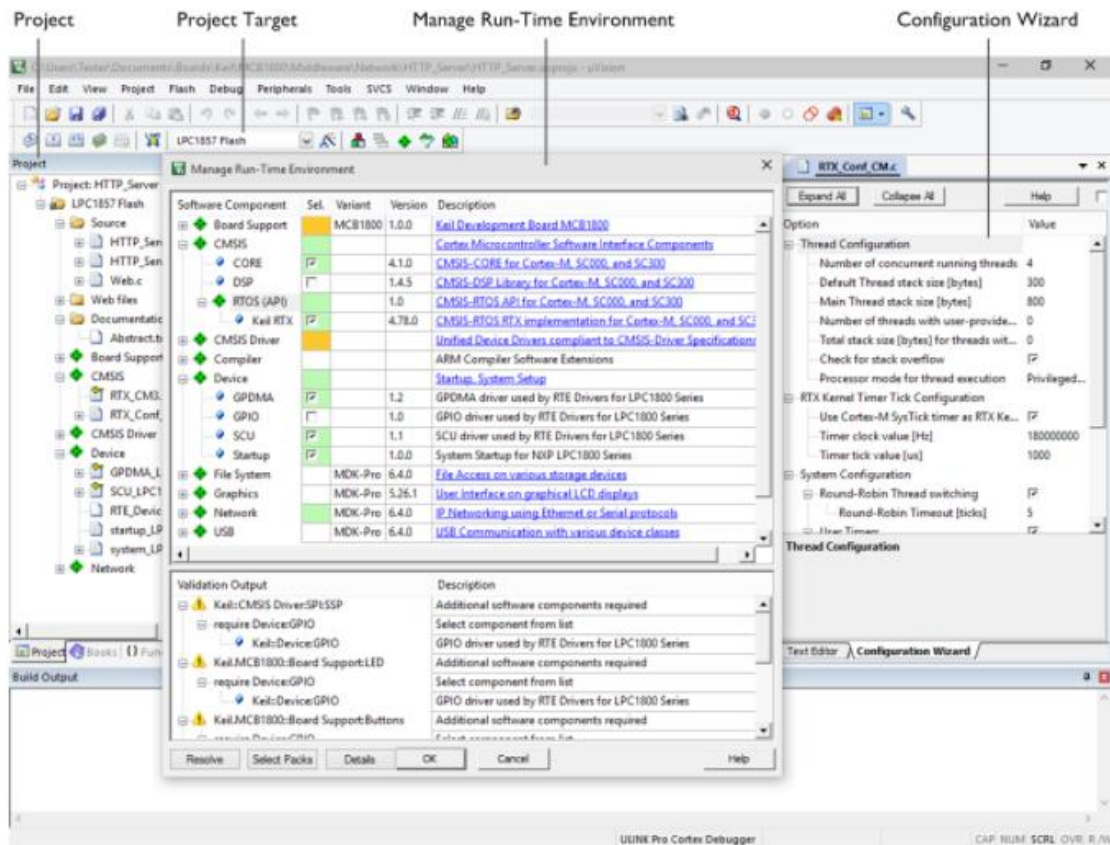


Figura 20. Gestor de proyectos y entorno de ejecución en tiempo real.

- La ventana de proyecto nos muestra los archivos fuentes y los componentes software seleccionados. Debajo de los componentes software se encuentran las librerías correspondientes y los archivos de configuración.
- Los proyectos soportan varias tarjetas, que facilitan la administración de la configuración, y que pueden usarse para generar compilaciones para diferentes plataformas hardware.
- La ventana de gestión del entorno en tiempo de ejecución nos muestra todo los componentes software que son compatibles con el dispositivo seleccionado.
- El asistente de configuración es un editor de utilidad para controles de configuración similares a GUI en ensamblador, en C/C++ o en archivos de inicialización.

El modo depuración provee un único entorno en el cual podemos testar, verificar y optimizar nuestras aplicaciones de código. El depurador incluye características típicas como puntos de ruptura simples y complejos, ventanas de visión de variables, y una ejecución proveyendo una visión completa de los dispositivos periféricos. Este modo presenta la siguiente apariencia:

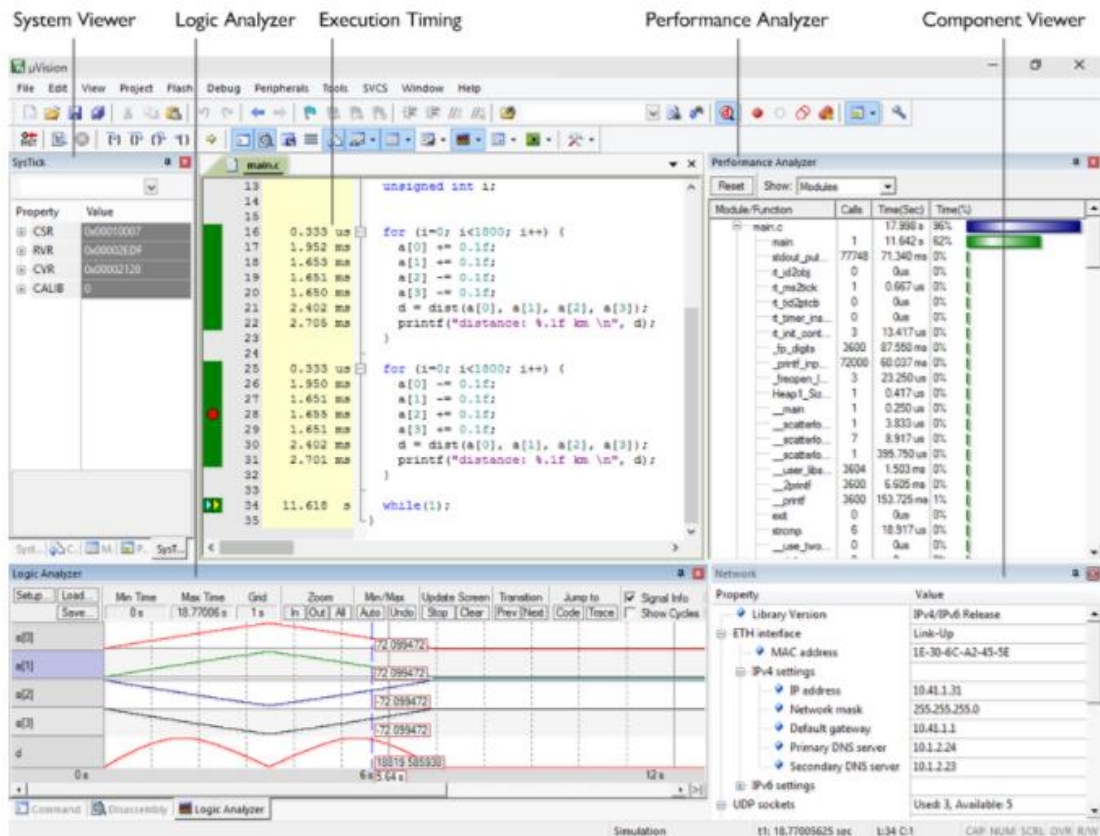


Figura 21. Modo depuración.

- El visualizador del sistema provee información detallada de cada periférico del microcontrolador.
- El tiempo de ejecución es resumido en el analizador de rendimiento y está disponible para las sentencias de código.
- El analizar lógico nos muestra de forma gráfica como las variables y señales cambian de estado.
- El visualizador de componentes nos muestra las variables y las estructuras de los componentes software.

8.2. Comentarios sobre el código

La programación del código de control de la balanza se ha realizado en el lenguaje de alto nivel CX-51 [2], el cual es una evolución del lenguaje C que permite utilizar la misma sintaxis para programar los microcontroladores derivados del 8051. Para usar los registros de control del MSC1210 hemos tenido que incluir el fichero “REG1210.H”, que asigna a las direcciones de memoria de cada registro del microcontrolador su nombre.

En el proyecto hemos incluido un fichero de inicialización (“STARTUP.A51”), que se ejecuta una única vez después de que se produzca un reset en el microcontrolador. Este

fichero está escrito en código ensamblador porque el compilador de C no está configurado para crear código fuera de la zona de memoria de código, y el microcontrolador MSC1210 tiene una zona de la memoria FLASH, que no es memoria de código, sino donde están implementados los registros de configuración hardware, que nos permiten, entre otras cosas, establecer la partición de la memoria FLASH entre memoria de programa y memoria de datos. Se ha configurado toda la memoria FLASH como memoria de programa, ya que no se necesita memoria de datos.

Los registros de configuración hardware sólo son accesibles cuando el microcontrolador se encuentra en modo programación, es decir, cuando PSEN se encuentra a 0; por lo que la modificación de estos registros se producirá mientras se está programando el microcontrolador a través de puerto serie, y no siempre que se produzca un reset.

Se ha llevado a cabo una programación modular para diseñar el código. Para probar la balanza se ha utilizado la placa UCOADP, que cuenta con un CPLD a través del que se realiza el control del visualizador LCD y del teclado. Cuando la balanza se monte de forma definitiva se utilizará un microcontrolador MSC1210 en vez de la placa UCOADP, por lo que se podría optar por usar los puertos del microcontrolador para el control del teclado y del visualizador LCD, y sólo habría que modificar las funciones utilizadas para el control de ellos, en vez de tener que modificar todo el código del programa.

8.3. Descripción del enfoque seguido

Primeramente se han definido una serie de constantes: “*A_CLK*” y “*DECIMATION*”, para configurar la frecuencia de salida de datos del conversor analógico digital; “*LSB*”, para indicar el valor equivalente al bit menos significativo; “*sensibilidad*”, para indicar la sensibilidad de la celda de carga; “*excitación*”, para indicar el valor con el que se excita la celda de carga; “*Amp_ext*”, para indicar el valor de la ganancia del amplificador de instrumentación externo; “*bit_elim*”, para indicar el número de bit a eliminar de la conversión, porque hay ruido presente en ellos; y “*cal*”, para que la balanza muestre el mismo resultado que la balanza con la que la hemos calibrado.

Cuando el programa empieza a ejecutarse, primero se realiza la configuración de la duración de las operaciones de transferencia externa, de los registros del contador de milisegundos, del puerto serie y del temporizador para generar la señal de reloj que usa el puerto serie, del pin 1 del puerto 3 como salida CMOS, y de todos los registros de control del ADC. Una vez realizadas todas estas configuraciones, que están detalladas en el apartado 6.5.3.3, se habilita las interrupciones del ADC, y, antes de entrar en un bucle infinito, se realiza una calibración de los parámetros internos del ADC, y se calcula el valor del *offset*, el cual se introduce en la variable “*offset*”, que se restará a las siguientes conversiones que se realicen. También antes de entrar en el bucle infinito, configuramos el teclado para que se sondeen sólo la primera fila, ya que sólo utilizamos las 3 primeras teclas de dicha fila.

Para controlar el teclado se ha definido dos funciones, la función “*filas*”, que escribe en el registro del teclado el valor que se haya pasado como argumento; al escribir en este registro se escribe el valor de las filas del teclado. Y la función “*pulsada*”, que lee el

registro del teclado para ver si se ha pulsado una tecla pulsada. Al leer este registro se lee el valor de las columnas.

Para controlar el visualizador LCD se han definido seis funciones: “*esperar_lcd_no_ocupado*”, “*envia_orden_lcd*” y “*envia_caracter_lcd*”, que hacen lo que su nombre indica, y cuyo funcionamiento ha sido explicado en el apartado 6.6; “*saca_texto_lcd*”, para sacar una cadena de caracteres a partir de la dirección especificada; “*inicio_lcd*”, para realizar el proceso necesario para la inicialización del LCD; y “*alimentar_lcd*”, para habilitar, a través del registro correspondiente, la alimentación del visualizador LCD.

Para el control de la transmisión de los caracteres tipo ASCII a través del puerto serie se han definido dos funciones: “*transmitir*”, para transmitir un carácter; “*sacar_texto_ps*”, para sacar por el puerto serie una cadena de caracteres. Cabe mencionar la particularidad de la familia 8051 respecto a las transmisiones [3]: el bit TI nos indica cuando podemos realizar una transmisión (cuando este está en estado alto), pero cuando comenzamos la transmisión, escribiendo en el registro SBUF un determinado valor, el bit TI no se pone automáticamente a estado bajo, sino que hay que ponerlo por programa. El diagrama de flujo en el que se muestra como habría que realizar una transmisión por el puerto 0, es el siguiente:

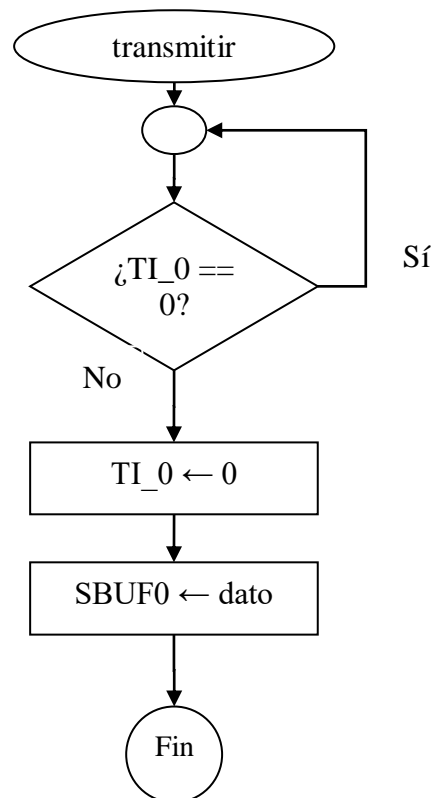


Figura 22. Diagrama del flujo de una transmisión.

Una vez que el programa entra en el bucle infinito se escanean las tres primeras columnas del teclado, y a continuación se comprueba si está lista la conversión del ADC, leyendo el bit 5 del registro AISTAT que nos indica cuando ha terminado una conversión. En el caso de que esté lista una conversión se llama a la función “*gestion_adc*”. Para escanear las columnas primeramente, mediante la función

”*pulsada*”, se guarda el valor de las columnas en la variable “*col*”, y a continuación se compara esta variable con el valor que debería tener si alguna tecla estuviese pulsada. Si se detecta que una tecla está pulsada, para que comience a realizarse la función correspondiente a la tecla pulsada (explicadas en el apartado 6.7), previamente se esperarán 10ms tras detectar que la tecla está pulsada y 10ms tras detectar la relajación de la tecla, para filtrar los rebotes mecánicos. El diagrama de flujo de este bucle infinito de escaneo se indica en la figura 24.

La función “*gestión_adc*” es la encargada de realizar las gestiones correspondientes a las conversiones del ADC. En ella, primero se leen los registros en los que aparece la conversión del ADC, ya que de lo contrario no empezaría una nueva conversión. A continuación se eliminan los bits que hayamos determinado, redondeando el resultado con el último bit eliminado, se ve si se ha descartado ya tres muestras (para que el filtro se estabilice); en el caso de que no se haya descartado las tres muestras, se decrementa la variable que nos indica cuantas muestras quedan por descartar, la cual es “*datos_a_descartar*”, que se inicializa con el valor 3 al arrancar el programa, y salimos de la función. En el caso de que ya se haya descartado los datos necesarios, se convierte el valor digital de nuevo en voltios mediante la constante “*LSB*”; luego se convierte el valor de voltaje en kilogramos mediante las constantes “*sensibilidad*”, “*excitación*”, “*Amp_ext*” y “*cal*”. Por último se envía el resultado al LCD y a través del puerto serie, si la variable “*hab_t*” nos lo indica. El diagrama de flujo correspondiente a esta función se indica en la figura 23.

Por último mencionar que también se ha utilizado una función, denominada “*num_a_car3d*”, que convierte un valor tipo *int*, de tres dígitos como máximo, en su valor correspondiente en código ASCII. Los valores que se pasan como atributo a esta función son la parte entera y la parte decimal del peso, para enviarlas en código ASCII al LCD y por el puerto serie.

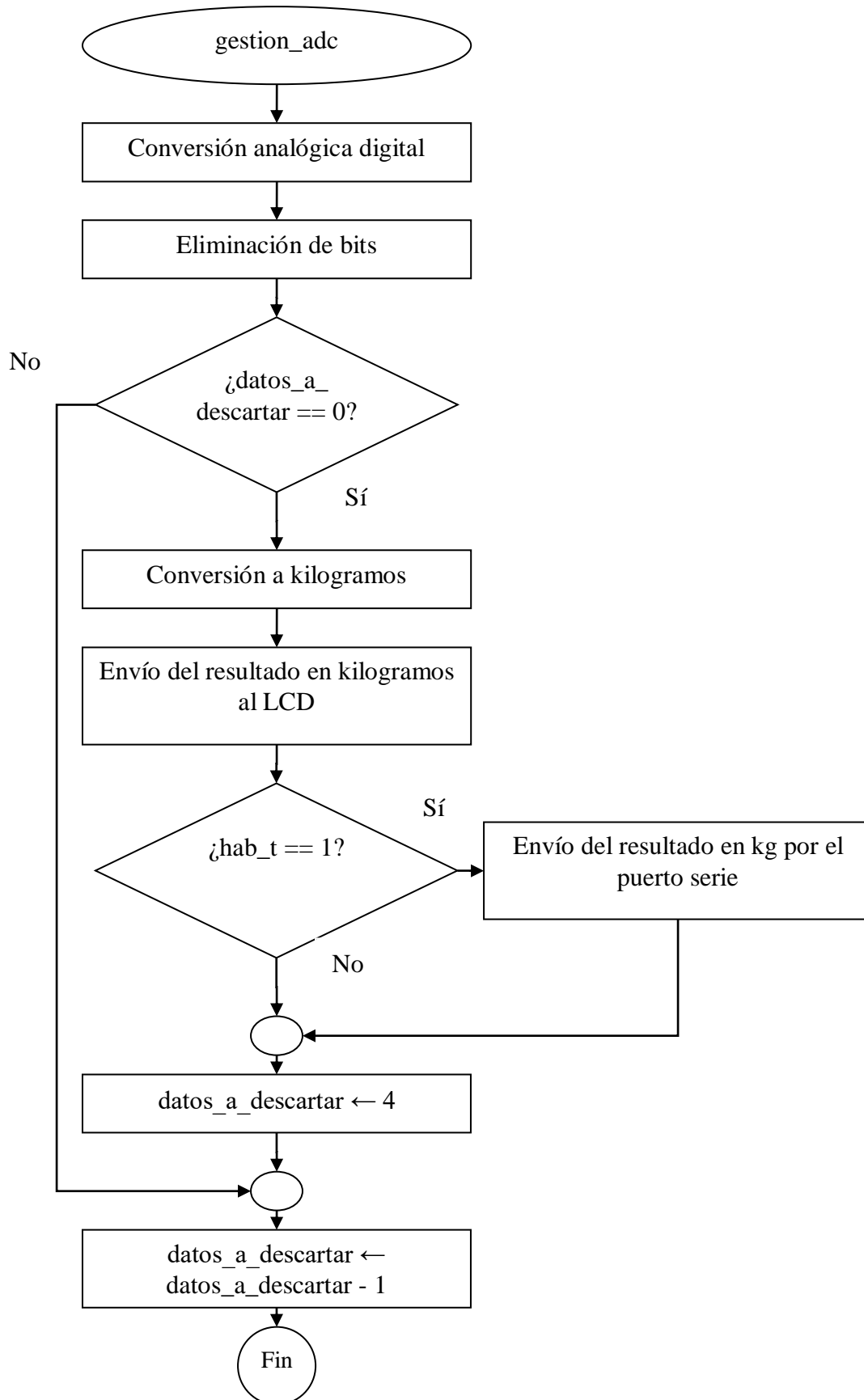


Figura 23. Diagrama de flujo de la función “gestion_adc()”

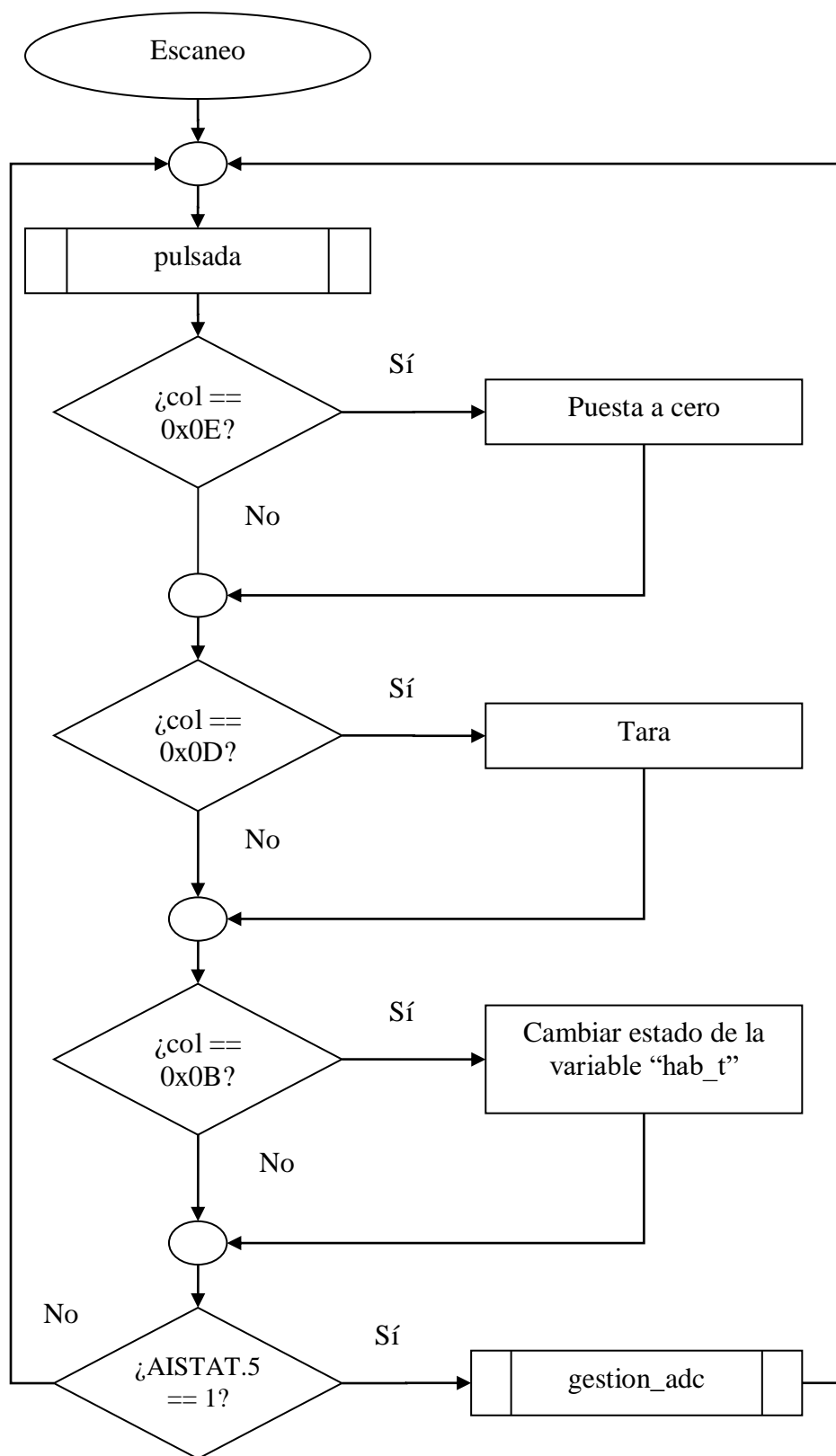


Figura 24. Diagrama de flujo del bucle infinito de sondeo.

9. Montaje y prueba de funcionamiento

Para programar el microcontrolador MSC1210, con la aplicación software diseñada, a través del puerto serie, mediante conexión USB con un ordenador, gracias al dispositivo *FTDI 232* con el que cuenta la placa *UCOADP*, se ha usado el programa software *TI downloader*.

Una vez programado el MS1210, con el programa *TI downloader*, que también actúa como terminal en la que se muestran todos los mensajes enviados por el puerto serie, se vio que había mucho ruido en las mediciones, y que las medidas estaban saliendo distintas a lo que se esperaba.

Al hacer algunas mediciones en el prototipo, se vio que la desviación en las medidas era debida a que el amplificador externo no estaba funcionando correctamente, ya que su ganancia no se estaba manteniendo constante, sino que variaba en función del voltaje. Seguramente esto se debiera a que el amplificador estaba mal o que había algún problema en las conexiones.

Dado que no se disponía de tiempo como para pedir otro amplificador, para sustituir el otro, se optó por hacer algunos cambios en el diseño, los cuales se explican en el apartado 10.

El montaje definitivo del prototipo se ve en la figura 25.

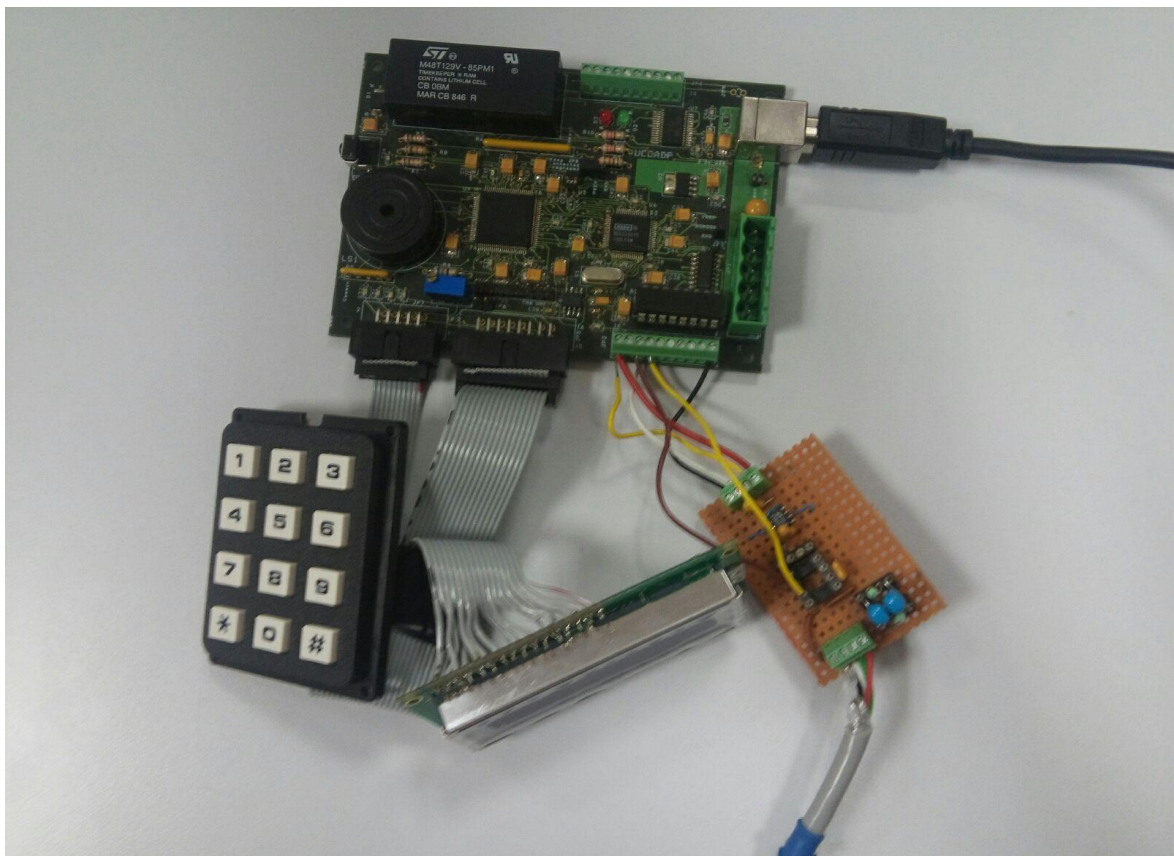


Figura 25. Montaje del prototipo.

También en la prueba de funcionamiento, se vio que había más ruido eléctrico del que se esperaba. En el resultado cambiaban los 11 bits menos significativos de los 24 bits del conversor analógico digital. Esto también obligó a realizar algunos cambios en el diseño, que se recogen en el apartado 10.

Una vez hechos todos los cambios necesarios, se empezó a ver como la balanza iba dando medidas cercanas a las esperadas, sin que oscilaran entre dos o más valores.

Finalmente se llevó a cabo la medición necesaria para realizar el último paso de la calibración, de forma que nuestra balanza dé el mismo valor que la báscula con la que la se ha calibrado. Este último paso de la calibración consistía en comparar el peso de una maleta en nuestra balanza y en la báscula que usamos para calibrarla, y encontrar el factor necesario para que nuestra balanza muestre el mismo peso que la báscula con la que la hemos calibrado. Este factor denominado “*cal*” que se utilizará en el programa de la aplicación, responde a la siguiente fórmula:

$$cal = \frac{\text{peso de la maleta en la báscula de baño}}{\text{peso de la maleta en nuestra balanza}}$$

(Ecuación 9.1)

Se obtuvo que la medida que nos daba nuestra balanza era 23.606, y dado que la báscula de baño daba una medida de 24.3kg, aplicando la fórmula 9.1 se obtiene:

$$cal = \frac{24.3}{23.606} = 1.029399305$$

Configurando la aplicación software con este parámetro, podemos ver en la figura 26, como nuestra balanza nos daba el mismo valor, sin oscilar, que la báscula de baño, una vez que se estabiliza.

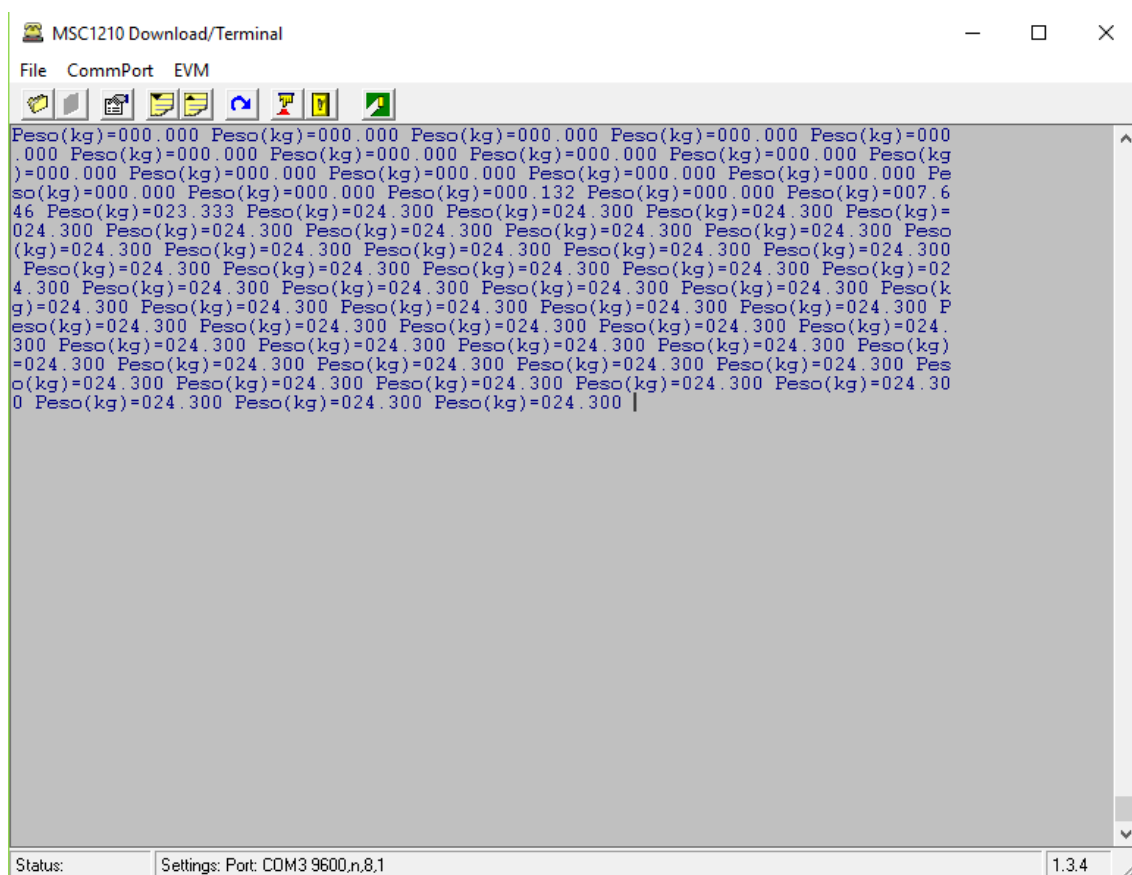


Figura 26. Balanza calibrada.

10. Cambios efectuados en el diseño

En este apartado se recopilan todos los cambios que hubo que hacer en el diseño que se preveía.

Como se vio que había más ruido eléctrico del que se esperaba y al no poder usar el amplificador, hubo que aumentar el valor de la ganancia interna que se usaba. Si en la ecuación 5.1 se elimina la ganancia externa y el *offset*, ya que se corrige automáticamente por programa, quedaría:

$$\text{Peso(kg)} = \frac{V_o * P_{\text{máx}}}{S_{cc} * V_{\text{exc}}}$$

(Ecuación 10.1)

Usando la ganancia máxima (128) y aplicando las ecuaciones 6.6 y 10.1, y teniendo en cuenta que se vio que hay ruido en los 11 bits menos significativos, por lo que hay que eliminarlos, se obtiene una resolución:

$$\text{LSB} = \frac{4.096\text{V}/128}{2^{13}} = 3.906 * 10^{(-6)}\text{V}$$

$$\text{Peso(kg)} = \frac{3.906 * 10^{(-6)}\text{V} * 300\text{kg}}{2\text{mV/V} * 4.096\text{V}} = 0.143051\text{kg} = 143.05\text{g}$$

Esta resolución no es suficiente, por lo que se optó por usar la tensión de referencia interna de 1.25V, para el ADC en vez de la externa de 4.096V, e implementar mediante la aplicación software una media de las 4 últimas mediciones, haciendo así un filtro software con el que consiguió un bits más de resolución. Aplicando las ecuaciones 6.6 y 10.1, ahora la resolución es:

$$\text{LSB} = \frac{1.25\text{V}/128}{2^{14}} = 5.960 * 10^{(-7)}\text{V}$$

$$\text{Peso(kg)} = \frac{5.960 * 10^{(-7)}\text{V} * 300\text{kg}}{2\text{mV/V} * 4.096\text{V}} = 0.021827\text{kg} = 21.83\text{g}$$

Dicho valor de resolución cumple con los objetivos.

Para seleccionar la tensión de referencia interna de 1.25V y cambiar la ganancia programable a 128, hubo que cambiar el valor con el que se cargaba el registro ADCON0 por el valor 2Fh.

Al usar la tensión de referencia interna para el ADC, en vez de la referencia externa, que también alimenta la celda de carga, hace que las oscilaciones en la tensión de salida de la celda de carga, provocadas por oscilaciones en la tensión de referencia, no se compensen, pero se vio que esto no ocasionaba problemas.

El realizar la media de los 4 últimos valores hace que ahora salga una nueva medida en el visualizador LCD cada 1.6s. Esto se debe a que hay que descartar 3 mediciones para que el filtro se estabilice, por lo que nos quedamos con la cuarta, y dado que se adquiere una muestra del ADC cada 0.1s, obtendríamos un tiempo de 1.6s entre cada medida ($0.1s \times 4 = 0.4s$).

Dado que la balanza se ha calibrado con una báscula de 100 gramos de resolución, y el objetivo eran 100 gramos de resolución, se ha optado por mostrar 1 dígito decimal del resultado, o 2 dígitos, por si queremos ver un resultado más preciso. Para cambiar entre mostrar 1 o 2 dígitos decimales, se ha empleado otra tecla, la tecla 4, correspondiente a la segunda fila, por lo que ahora en el bucle infinito de escaneo del teclado, es necesario llamar a la función “*filas*” para cambiar entre sondear la primera y la segunda fila.

Para conseguir mostrar 1 o 2 dígitos de la parte decimal, redondeando el resultado, se han definido otras dos funciones: “*num_a_car1d*” y “*num_a_car2d*”, para 1 y 2 dígitos decimales respectivamente. Al redondear la parte decimal al dígito que hayamos elegido, puede que, al trabajar con caracteres ASCII, el carácter resulte no ser un número, por lo que se ha diseñado la función “*ajuste_ASCII*”, que se ocupa de solucionar este problema, y a la que se le pasa como argumento un valor, que indica si se ha redondeado a 1 o 2 dígitos decimales.

11. Posibles mejoras futuras

En este apartado se recopilan las posibles mejoras que se podrían realizar en el diseño del prototipo de la balanza, y que no se han podido llevar a cabo por motivos de presupuesto, y en algunos casos por falta de tiempo.

- Diseñar una placa de circuito impreso para el prototipo con plano de masa, para que el prototipo sea más inmune al ruido eléctrico.
- Seleccionar un amplificador externo, para obtener una mayor resolución.
- Diseñar un filtro paso bajo Butterworth a la salida del amplificador, para eliminar aún más ruido eléctrico.
- Apantallar los cables procedentes de la celda, para hacerla más inmune al ruido eléctrico.
- Realizar una calibración utilizando una balanza más precisa que la nuestra, para que la calibración sea buena.

12. Guía de usuario

La balanza diseñada en este proyecto sirve para pesar objetos de 0 a 150kg de peso con una resolución de 21.83 gramos, mostrando el resultado, redondeado a 1 o 2 dígitos decimales, en un visualizador LCD, y, si se ha configurado, enviando a través del puerto serie dicho resultado a un ordenador.

Las instrucciones para el uso de la balanza son las siguientes:

1. Preparación

- Colocar balanza sobre una superficie sólida y plana.
- Alimentar la placa UCOADP mediante el conector USB de la misma, ya sea conectándolo a la red, mediante un adaptador, o a un ordenador, a través de un puerto USB del mismo. Sobre la plataforma de la balanza no debe haber ningún peso a la hora de alimentarla.

2. Pesaje

- Si se coloca un objeto sobre la plataforma de la balanza, su peso aparecerá en el visualizador LCD mostrando 1 o 2 dígitos decimales del resultado, según se haya elegido, y, si se ha elegido, este resultado también se enviará por el puerto serie.
- En la esquina inferior derecha del visualizador LCD aparecerá un mensaje que nos indica si el resultado se está mandando por el puerto serie. En el caso de que se esté mandando aparecerá: "PS=SI", y en el caso de que no se esté enviando aparecerá: "PS=NO".
- Para recibir, y ver, lo enviado a través del puerto serie será necesario tener instalado algún programa que actúe como terminal. Por ejemplo, el que se ha usado para probar la balanza se denomina *termite*, y puede ser descargado de la siguiente dirección Web: https://www.compuphase.com/software_termite.htm#
- Mediante el teclado numérico es posible hacer algunas configuraciones en la balanza.

3. Teclas para la configuración de la balanza.

- *Tecla 1.* Se utiliza para poner a cero la balanza, si la balanza se ha arrancado con un peso sobre ella, o se ha producido un cambio en la balanza significativo. Al pulsar esta tecla se muestra en la pantalla LCD un mensaje que nos indica que debemos vaciar la balanza, y pulsar de nuevo esta tecla para que comience la puesta a 0.
- *Tecla 2.* Se utiliza para realizar una TARA del resultado.
- *Tecla 3.* Se utiliza para alternar entre la habilitación y la deshabilitación del envío del resultado, a través del puerto serie. Por defecto, la balanza está configurada para no enviar el resultado por el puerto serie.
- *Tecla 4.* Se utiliza para alternar entre mostrar el resultado con 1 o 2 dígitos decimales del peso. Por defecto se muestra 1 dígito decimal.

13. Bibliografía

La bibliografía que se ha utilizado para realizar este proyecto es la siguiente:

[1]. TEXAS INSTRUMENTS. *MCS1210. Analog-to-Digital Converter with 8051 Microcontroller and Flash Memory. User's Guide* [en línea]. Dalas, Texas: 2002. Disponible en:
<http://www.ti.com/lit/ug/sbau077/sbau077.pdf>

[2]. KEIL SOFTWARE. *Cx51 Compiler. Optimizing C Compiler and Library Reference for Classic and Extended 8051 Microcontrollers. User's Guide* [en línea]. 2001. Disponible en:
www.neuromorphs.net/nm/raw-attachment/wiki/2010/usb10/C51.PDF

[3]. MORENO FERNÁNDEZ- CAPARRÓS, Antonio. Apuntes de las asignatura: *microcontroladores*. Universidad de Córdoba, 2016.

[4]. PALLARÉS LÓPEZ, Víctor. Apuntes de la asignatura: *instrumentación electrónica*. Universidad de Córdoba, 2016.

[5]. Página Web:
<http://www.pce-iberica.es/instrumentos-de-medida/balanzas-vision-general.htm>

[6]. TEXAS INSTRUMENTS. *Incorporating the MSC1210 into Electronic Weight Scale Systems* [en línea]. Dalas, Texas: 2004. Disponible en:
[ftp://ftp.ti.com/pub/data_acquisition/MSC_CD-ROM/Application_Notes/sbaa092a_MSC1210_in_Weight_Scale_Systems.pdf](http://ftp.ti.com/pub/data_acquisition/MSC_CD-ROM/Application_Notes/sbaa092a_MSC1210_in_Weight_Scale_Systems.pdf)

[7]. TEXAS INSTRUMENTS. *Precision Analog-to-Digital Converter (ADC) with 8151 Microcontroller and Flash Memory* [en línea]. Dalas, Texas: 2008. Disponible en:
<http://www.ti.com/lit/ds/symlink/msc1210y3.pdf>

[8]. TEXAS INSTRUMENTS. *Using the Delta-Sigma ADC on the MSC12xx* [en línea]. Dalas, Texas: 2003. Disponible en:
<http://www.icbase.com/file/pdf/add/ti/AN-101-00018en.pdf>

[9]. Página Web:
<http://www2.keil.com/mdk5/uvision/>

[10]. JIMÉNEZ FERNÁNDEZ, Rafael. Proyecto final de grado: *Diseño del hardware de un módulo portátil de adquisición datos*. Universidad de Córdoba, 2009.

[11]. LERGA OLCOZ, Josué. Proyecto final de grado: *Control de pesaje industrial* [en línea]. Universidad de Pamplona, 2013. Disponible en:
<http://academica-e.unavarra.es/bitstream/handle/2454/10345/629114.pdf?sequence=1>

[12]. CASTRO RIOS, Ramiro Saito. Proyecto final de grado: *Estudio e implementación de un convertidor analógico digital delta sigma* [en línea]. Universidad de Barcelona, 2015. Disponible en:

<http://upcommons.upc.edu/handle/2117/89044>

[13]. LUMEX. Datasheet: *LCM-S01602DSF/A* [en línea]. Palatine, Illinois: 2002. Disponible en:

<http://www.mouser.com/ds/2/244/LCM-S01602DSF%20A-108827.pdf>

[14]. TEXAS INSTRUMENTS. Datasheet: *INA22* [en línea]. Dalas, Texas: 2016. Disponible en:

<http://www.ti.com/lit/ds/symlink/ina122.pdf>

[15]. LINEAR TECHNOLOGY. Datasheet: *LT1461* [en línea]. Milpitas, California. Disponible en:

<http://cds.linear.com/docs/en/datasheet/1461fb.pdf>

Anexo 1. Presupuesto

Aclarar antes de nada, que para realizar presupuesto del prototipo diseñado en vez de poner la placa UCOADP se ha puesto un microcontrolador MSC1210 únicamente, pensando en cuando el prototipo se monte definitivamente, y como la placa UCOADP fue desarrollada en un proyecto de fin de grado es difícil estimar su precio.

A continuación se muestran las tablas con las mediciones, precios unitarios y con el presupuesto parcial.

1. Mediciones

UNIDAD DE MEDIDA	DENOMINACIÓN	DESCRIPCIÓN	NÚMERO DE UNIDADES
UD	Microcontrolador	Microcontrolador MSC1210Y5 de <i>TI</i>	1
UD	Visualizador LCD	Visualizador LCD LCM-S01602DSF/A de <i>Lumex</i>	1
UD	Teclado matricial	Teclado matricial de 4 filas y tres columnas	1
UD	Celda de carga con plataforma	Celda de carga de 300kg fabricada por <i>HIWEIGH</i>	1
UD	Generador de tensión de referencia	Generador de tensión de referencia <i>LT1461</i>	1
UD	Condensador	Condensador tántalo de 10 μ F, $\pm 20\%$	4
UD	Resistencia	Resistencia fija de 2k Ω $\pm 20\%$	2
UD	Conector	Conector macho de 4 pines	2
UD	Cristal de cuarzo	Cristal de cuarzo de 11.0592MHz	1

2. Precios unitarios

UNIDAD DE MEDIDA	DENOMINACIÓN	DESCRIPCIÓN	PRECIO UNITARIO (€)
UD	Microcontrolador	Microcontrolador MSC1210Y5 de <i>TI</i>	22.25
UD	Visualizador LCD	Visualizador LCD LCM-S01602DSF/A de <i>Lumex</i>	11.76
UD	Teclado matricial	Teclado matricial de 4 filas y tres columnas	40.46
UD	Celda de carga con plataforma	Celda de carga de 300kg fabricada por <i>HIWEIGH</i>	105
UD	Generador de tensión de referencia	Generador de tensión de referencia <i>LT1461</i>	3.12

UD	Condensador	Condensador tántalo de 10 μ F $\pm 20\%$	0.934
UD	Resistencia	Resistencia fija de 2k Ω $\pm 20\%$	0.278
UD	Conector	Conector macho de 4 pines	0.196
UD	Cristal de cuarzo	Cristal de cuarzo de 11.0592MHz	1.88

3. Presupuesto parcial

UNIDAD DE MEDIDA	DENOMINACIÓN	NÚMERO DE UNIDADES	PRECIO UNITARIO (€)	PRECIO (€)
UD	Microcontrolador	1	22.25	22.25
UD	Visualizador LCD	1	11.76	11.76
UD	Teclado matricial	1	40.46	40.46
UD	Celda de carga con plataforma	1	105	105
UD	Generador de tensión de referencia	1	3.12	3.12
UD	Condensador	4	0.934	3.736
UD	Resistencia	2	0.278	0.556
UD	Conector	2	0.196	0.392
UD	Cristal de cuarzo	1	1.88	1.88
Precio total				189.16

Anexo 2: Listado del código comentado

Se adjunta a partir de la siguiente hoja el listado del código en formato horizontal de la página.


```
#include <REG1210.H>
#include <MATH.H>

//Prototipado de funciones
void esperar_lcd_no_ocupado(void);
void envia_orden_lcd(unsigned char data octeto);
void inicio_lcd(void);
void envia_caracter_lcd(unsigned char data dato);
void sacar_texto_lcd(unsigned char data texto[], unsigned char cursor);
void espera_xms(unsigned int data tiempo);
void transmitir(unsigned char data dato);
void num_a_car3d(unsigned int data num);
void num_a_car2d(unsigned int data num);
void num_a_car1d(unsigned int data num);
void ajuste_ASCII(unsigned char data dig);
void sacar_texto_ps(unsigned char data texto[]);
unsigned char pulsada(void);
void filas(unsigned char data fil);
void alimentar_lcd(void);
void gestion_adc(void);

/* Direcciones de los registros de control, estado, escritura
y lectura (en su caso) de los periféricos mapeados en memoria por
decodificación del CPLD*/
#define REGTECLADO 0xFFEA //Registro del teclado, al escribir en él se escribe en
                          //filas y al leer de él se leen columnas (todo ello
                          //en sus LSBs)

#define REGCNTLCD 0xFFE0 //Dirección en la que comienzan los registros
                          //de control del LCD

#define REGCONFTONE 0xFFEB //Registro en el que hay que escribir para habilitar la salida
                          //de tono (TONEOUT*), la salida del PWM (PWM) y
                          //la alimentación del LCD (PWRLCD) escribiendo un 1 en los bits
                          //de este registro 0, 1 y 2, respectivamente
```

```
// Definiciones de constantes para configurar el ADC
#define A_CLK    10
#define DECIMATION 1571    // Con estos ACLK y DECIMATION se consigue una frecuencia de salida de datos de 10 Hz
#define LSB 5.960464478e-7    // LSB=(1.25/128)/2^14
#define sensibilidad 0.002043547454
#define excitacion 4.096
#define bit_elim 10
#define cal 1.029399305    // Valor necesario calibrar la balanza

//Definición de variables
unsigned char datos_a_descartar=3;    // Variable para descartar 3 conversiones del ADC
unsigned long data res;    // Variable que almacena el resultado de la conversión a digital
unsigned int data parte_entera;    // Variable en la que se almacena la parte entera del peso
unsigned int data parte_decimal;    // Variable en la que se almacena la parte decimal del peso
signed long data peso_gramos_anterior=0;
bit data hab_t=0;    // Variable que habilita y deshabilita la transmisión por el puerto serie
/* hab_t=0 -> No se transmite el dato
   hab_t=1 -> Se transmite el dato por el PS */
bit data digitos=0;    // Variable que indica si se mostrarán 1 o 2 dígitos decimales
/* digitos=0 -> 1 Se muestra un dígito
   digitos=1 -> 2 Se muestra un dígito */
unsigned long pesos[4];
signed char indice_pesos=0;
unsigned char code mensaje1[]="Vacie la báscula";
unsigned char code mensaje2[]="Y pulse 1";
unsigned char code mensaje3[]="Poniendo a 0...";
unsigned char code mensaje4[]="Peso(kg)=";
unsigned char code mensaje5=".";
unsigned char code mensaje6=" ";
unsigned char code mensaje7[]="Peso(kg)=000.0";
unsigned char code mensaje8[]="PS=SI";
unsigned char code mensaje9[]="PS=NO";
unsigned char code mensaje10[]="Peso(kg)=000.00";
```



```
unsigned long data offset=0;           // Variable para modificar el peso que aparece por pantalla
signed long data conversion;          // Variable para guardar el valor de la conversión digital
unsigned char data decimales_3d[3];   // Variable en la que se guarda el número convertido por la función num_a_car3d
unsigned char data decimales_2d[2];   // Variable en la que se guarda el número convertido por la función num_a_car2d
unsigned char data decimales_1d;      // Variable en la que se guarda el número convertido por la función num_a_car1d
unsigned char code secuencia_de_iniciacion_lcd[5]={0x38,0x38,0x06,0x0C,0x01};
/* 38h = comunicación con bus de 8 bits, LCD de 2 líneas
o6h = incremento del cursor, pantalla estática (se mueve el cursor)
oCh = enciende pantalla, cursor no visible, cursor no parpadeante
o1h = borra pantalla y ubica cursor al inicio */

void main(void){
    // Declaración de variables locales
    char data i;
    unsigned char data col=0;
    bit bit_aux;
    // Fin de la declaración
    // Configuración inicial
    CKCON=0;           // Los contadores usan fosc/12 y 0 MOVX cycle stretch
    MSECH=(11058>>8);
    MSECL=11058 & 0xFF;
    // Configuración del puerto serie y de los temporizadores
    SCON0 = 0x42;      // Se usa la uarto en modo 1 (full-duplex asíncrono) con la recepción deshabilitada
    TMOD |= 0x20;      // Se configura el temporizador 1 en modo 2 (contador de 8 bits con autorecarga)
    PCON=0x80|PCON;    // Se pone SMOD a 1 para duplicar la velocidad de comunicación
    TH1=0xFA;         // Se configura la velocidad de comunicación a 9600 bps
    TR1=1;            // Se arranca el Temp1
    P3DDRL&=0xF3;     // Se configura el puerto TxDo de la UART0
    P3DDRL|=0x04;      // como salida para la transmisión
    // Configuración del ADC
    ADMUX = 0x10;      // (AIN+ = AIN1), (AIN- = AIN0)
    ACLK = A_CLK;
    ADCON0 = 0x2F;     // Vref on=1.25V, Buff off, BOD on, PGA=128
    ADCON1 = 0x71;     // Unipolar, filtro en modo sin3, self calibration (offset, gain)
    ADCON2 = DECIMATION & 0xFF; // Parte baja de DECIMATION
```

```
ADCON3 =(DECIMATION>>8) & 0x07; // 3 bits más significativos de DECIMATION

alimentar_lcd();           // Se habilita la alimentación al LCD
inicio_lcd();             // Se inicia la pantalla LCD

PDCON=0x23;               // Se activa el ADC del microcontrolador
AIE=AIE | 0x20;           // Se habilita la interrupción del ADC

for(i=0;i<4;i++){         // Se realiza una medida para la puesta a cero de la balanza
while(!(AISTAT & 0x20));
    res=ADRESH;
    res=res<<8;
    res=res+ADRESM;
    res=res<<8;
    res=res+ADRESL;
    res=res>>(bit_elim-1);
    bit_aux=res;
    res=res>>1;
    if (bit_aux==1) res=res+0x01;
}
offset=res;
sacar_texto_lcd(mensaje7,0x80);
if (hab_t==1) sacar_texto_ps(mensaje7);

while(1){
    // Sondeo del teclado
    filas(0x0E);           // Se sondea la primera fila
    col=pulsada();
    if(col==0x0E){          // Primera tecla -> Puesta a 0
        espera_xms(10);    // Se espera 10 ms para filtrar los rebotes mecánicos
        while(pulsada()!=0x0F); // Se espera a que se deje de pulsar la tecla
        espera_xms(10);    // Se espera 10 ms para filtrar los rebotes mecánicos
        sacar_texto_lcd(mensaje1,0x80);
        sacar_texto_lcd(mensaje2,0xC3);
    }
}
```

```
if(hab_t==1){
    sacar_texto_ps(mensaje1);
    sacar_texto_ps(mensaje2);
}
while(pulsada()!=0x0E); // Se espera a que se vuelva a pulsar la primera tecla
espera_xms(10); // Se espera 10 ms para filtrar los rebotes mecánicos
while(pulsada()!=0x0F); // Se espera a que se deje de pulsar la tecla
espera_xms(10); // Se espera 10 ms para filtrar los rebotes mecánicos
envia_orden_lcd(0x01); // Se borra lo que hay en la pantalla
sacar_texto_lcd(mensaje3,0x80);
if(hab_t==1){
    sacar_texto_ps(mensaje3);
}
for(i=0;i<4;i++){ // Se realiza una medida para la puesta a cero de la balanza
    while(!(AISTAT & 0x20));
    res=ADRESH;
    res=res<<8;
    res=res+ADRESM;
    res=res<<8;
    res=res+ADRESL;
    res=res>>(bit_elim-1);
    bit_aux=res;
    res=res>>1;
    if (bit_aux==1) res=res+0x01;
}
offset=res;
envia_orden_lcd(0x01); // Se borra lo que hay en la pantalla
sacar_texto_lcd(mensaje7,0x80);
if (hab_t==1){
    if (digitos==1) sacar_texto_ps(mensaje10);
    else sacar_texto_ps(mensaje7);
}
}
if(col==0x0D){ // Segunda tecla -> TARA
```

```

    espera_xms(10);           // Se espera 10 ms para filtrar los rebotes mecánicos
    while(pulsada()!=0x0F);    // Se espera a que se deje de pulsar la tecla
    espera_xms(10);           // Se espera 10 ms para filtrar los rebotes mecánicos
    offset=conversion+offset;
}
if(col==0x0B){                // Tercera tecla -> Habilita o deshabilita la transmisión por el ps
    espera_xms(10);           // Se espera 10 ms para filtrar los rebotes mecánicos
    while(pulsada()!=0x0F);    // Se espera a que se deje de pulsar la tecla
    espera_xms(10);           // Se espera 10 ms para filtrar los rebotes mecánicos
    if (hab_t==1) hab_t=0;
    else hab_t=1;
}
filas(0x0D);                 // Se sondea la segunda fila
col=pulsada();
if(col==0x0E){                // Cuarta tecla -> Se cambia de mostrar 1 o 2 dígitos
    espera_xms(10);           // Se espera 10 ms para filtrar los rebotes mecánicos
    while(pulsada()!=0x0F);    // Se espera a que se deje de pulsar la tecla
    espera_xms(10);           // Se espera 10 ms para filtrar los rebotes mecánicos
    if (digitos==1) digitos=0;
    else digitos=1;
}
if((AISTAT & 0x20)!=0){       // Se comprueba si se ha producido una interrupción producida por el ADC
    gestion_adc();
}
}
}

void esperar_lcd_no_ocupado(void){
// Función que espera hasta que el LCD deje de estar ocupado
// Declaración de variables locales
bit data ocupado=0;
unsigned char xdata *reglcd;   // Se define el puntero que leerá el registro de control del LCD
// Fin de la declaración de variables locales
reglcd=REGCNTLCD+1;           // Se posiciona el puntero en el registro correspondiente a la lectura del registro de estado

```

```
do{
    reglcd=REGCNTLCD+1;    // Se escribe cualquier dato, y con ello se hace RS=0, RW=1 y E=1
    *reglcd=1;
    reglcd+=3;            // Se posiciona el puntero en el registro correspondiente a la desactivación del enable,
    ocupado=(*reglcd)&0x80; // y con ello se pone el enable a 0
}
while(ocupado);
}

void envia_orden_lcd(unsigned char data octeto){
    // Función que envia una orden al LCD
    // Declaración de variables locales
    unsigned char xdata *reglcd;    // Se define el puntero que escribirá en el registro de control del LCD
    // Fin de la declaración de variables locales
    esperar_lcd_no_ocupado();        // Se espera hasta que el LCD deje de estar ocupado
    reglcd=REGCNTLCD;               // Se posiciona el puntero en el registro de configuración del LCD

    *reglcd=octeto;                 // Se escribe el dato, y con ello se hace RS=0, RW=0 y E=1
    reglcd+=4;                      // Se posicio el puntero en el registro correspondiente a la desactivación del enable,
    *reglcd=1;                      // y con este acceso ponemos a 0 el enable
}

void inicio_lcd(void){
    // Función que inicia la pantalla LCD
    // Declaración de variables locales
    unsigned char data i;
    // Fin de la declaración
    for (i=0;i<5;i++){
        espera_xms(15);            // Se espera 15 ms antes de enviar cada orden
        envia_orden_lcd(secuencia_de_iniciacion_lcd[i]);
    }
    espera_xms(15);                // Se espera 15 ms al final de la última orden
}

void envia_caracter_lcd(unsigned char data dato){
```

```
// Función que escribe un carácter en la pantalla LCD
// Declaración de variables locales
unsigned char xdata *reglcd; // Se define el puntero que escribirá en el registro de DATOS del LCD
// Fin de la declaración
esperar_lcd_no_ocupado(); // Se espera a que el lcd de no esté ocupado
reglcd=REGCNTLCD+2; // Se posiciona el puntero en el registro de escritura de datos del LCD

*reglcd=dato; // Se escribe el dato, y con ello se hace RS=1, RW=0 y E=1
reglcd+=2; // Se posiciona el puntero en el registro correspondiente a la desactivación del enable,
*reglcd=dato; // y con este acceso ponemos a 0 el enable
}

void sacar_texto_lcd(unsigned char data texto[],unsigned char cursor){
// Función que saca un texto por pantalla desde la posición del cursor indicada
// Declaración de variables locales
unsigned char data i=0;
// Fin de la declaración
envia_orden_lcd(cursor);
while(texto[i]!=0x00){
    envia_caracter_lcd(texto[i]);
    i=i+1;
}
}

void espera_xms(unsigned int data tiempo){
/* Está función espera tantos milisegundos como se le pasen
como argumento, que puede ir desde 1ms a 65535ms */
// Declaración de variables locales
unsigned char data lectura; // Variable para leer el registro MSINT
// Fin de la declaración
PDCON=PDCON&0xFD; // Se activa el System time control para que
MSINT=0; // la bandera de interrupción aparezca cada
// 1ms+MSINT=1ms

while(tiempo){
while ((AIE&0x10)==0); // Se espera hasta que se produzca la interrupción
```

```
lectura=MSINT;      // La lectura del registro borra la petición de interrupción
AI=0;              // Se borra la bandera de interrupción auxiliar generada
tiempo--;          // Se decrementa los milisegundos que quedan por contar
}
PDCON=PDCON|0x02;   // Se desactiva el System time control
}
```

```
void transmitir(unsigned char data dato){
// Función para transmitir por la UART0
while(TI_0==0);      // Se espera hasta poder transmitir
TI_0=0;
SBUF0=dato;
}
```

```
void num_a_car3d(unsigned int data num){
/* Función que convierte un número de entrada, de 0 a 999 tipo int,
en una cadena de caracteres de 3 dígitos en formato ASCII */
if (num>899){
    decimales_3d[0]='9';
    num=num-900;
}
else if (num>799){
    decimales_3d[0]='8';
    num=num-800;
}
else if (num>699){
    decimales_3d[0]='7';
    num=num-700;
}
else if (num>599){
    decimales_3d[0]='6';
    num=num-600;
}
else if (num>499){
    decimales_3d[0]='5';
```

```
        num=num-500;
    }
    else if (num>399){
        decimales_3d[0]='4';
        num=num-400;
    }
    else if (num>299){
        decimales_3d[0]='3';
        num=num-300;
    }
    else if (num>199){
        decimales_3d[0]='2';
        num=num-200;
    }
    else if (num>99){
        decimales_3d[0]='1';
        num=num-100;
    }
    else decimales_3d[0]='0';

    if (num>89){
        decimales_3d[1]='9';
        num=num-90;
    }
    else if (num>79){
        decimales_3d[1]='8';
        num=num-80;
    }
    else if (num>69){
        decimales_3d[1]='7';
        num=num-70;
    }
    else if (num>59){
        decimales_3d[1]='6';
```



```
        num=num-60;
    }
    else if(num>49){
        decimales_3d[1]='5';
        num=num-50;
    }
    else if(num>39){
        decimales_3d[1]='4';
        num=num-40;
    }
    else if(num>29){
        decimales_3d[1]='3';
        num=num-30;
    }
    else if(num>19){
        decimales_3d[1]='2';
        num=num-20;
    }
    else if(num>9){
        decimales_3d[1]='1';
        num=num-10;
    }
    else decimales_3d[1]='0';

    decimales_3d[2]=num+48;
}

void num_a_car2d(unsigned int data num){
/* Función que convierte un número de entrada, de 0 a 999 tipo int,
en una cadena de caracteres de 2 dígitos en formato ASCII */
    if (num>899){
        decimales_2d[0]='9';
        num=num-900;
    }
    else if (num>799){
```

```
    decimales_2d[0]='8';  
    num=num-800;  
}  
else if (num>699){  
    decimales_2d[0]='7';  
    num=num-700;  
}  
else if (num>599){  
    decimales_2d[0]='6';  
    num=num-600;  
}  
else if (num>499){  
    decimales_2d[0]='5';  
    num=num-500;  
}  
else if (num>399){  
    decimales_2d[0]='4';  
    num=num-400;  
}  
else if (num>299){  
    decimales_2d[0]='3';  
    num=num-300;  
}  
else if (num>199){  
    decimales_2d[0]='2';  
    num=num-200;  
}  
else if (num>99){  
    decimales_2d[0]='1';  
    num=num-100;  
}  
else decimales_2d[0]='0';  
  
if (num>89){
```

```
    decimales_2d[1]='9';  
    num=num-90;  
}  
else if(num>79){  
    decimales_2d[1]='8';  
    num=num-80;  
}  
else if(num>69){  
    decimales_2d[1]='7';  
    num=num-70;  
}  
else if(num>59){  
    decimales_2d[1]='6';  
    num=num-60;  
}  
else if(num>49){  
    decimales_2d[1]='5';  
    num=num-50;  
}  
else if(num>39){  
    decimales_2d[1]='4';  
    num=num-40;  
}  
else if(num>29){  
    decimales_2d[1]='3';  
    num=num-30;  
}  
else if(num>19){  
    decimales_2d[1]='2';  
    num=num-20;  
}  
else if(num>9){  
    decimales_2d[1]='1';  
    num=num-10;
```

```
    }  
    else decimales_2d[1]='0';  
  
    if (num>=50) decimales_2d[1]=decimales_2d[1]+1;  
}  
  
void num_a_car1d(unsigned int data num){  
/* Función que convierte un número de entrada, de 0 a 999 tipo int,  
   en una cadena de caracteres de 3 dígitos en formato ASCII */  
    if (num>899){  
        decimales_1d='9';  
        num=num-900;  
    }  
    else if (num>799){  
        decimales_1d='8';  
        num=num-800;  
    }  
    else if (num>699){  
        decimales_1d='7';  
        num=num-700;  
    }  
    else if (num>599){  
        decimales_1d='6';  
        num=num-600;  
    }  
    else if (num>499){  
        decimales_1d='5';  
        num=num-500;  
    }  
    else if (num>399){  
        decimales_1d='4';  
        num=num-400;  
    }  
    else if (num>299){  
        decimales_1d='3';
```

```
        num=num-300;
    }
    else if (num>199){
        decimales_1d='2';
        num=num-200;
    }
    else if (num>99){
        decimales_1d='1';
        num=num-100;
    }
    else decimales_1d='0';

    if (num>=50) decimales_1d=+decimales_1d+1;
}

void ajuste_ASCII(unsigned char data dig){
/* Función que ajusta la parte decimal y entera, en código ASCII,
si un valor se sale de los límites
dig=1 -> Ajuste para 1 dígito decimal
dig=2 -> Ajuste para 2 dígito decimal */

    if (dig==1){
        if (decimales_1d==':'){
            decimales_3d[2]=decimales_3d[2]+1;
            decimales_1d='0';
            if (decimales_3d[2]==':'){
                decimales_3d[1]=decimales_3d[1]+1;
                decimales_3d[2]='0';
                if (decimales_3d[1]==':'){
                    decimales_3d[0]=decimales_3d[0]+1;
                    decimales_3d[1]='0';
                }
            }
        }
    }
}
```

```

}
if (dig==2){
    if (decimales_2d[1]==':'){
        decimales_2d[0]=decimales_2d[0]+1;
        decimales_2d[1]='0';
        if (decimales_2d[0]==':'){
            decimales_3d[2]=decimales_3d[2]+1;
            decimales_2d[0]='0';
            if (decimales_3d[2]==':'){
                decimales_3d[1]=decimales_3d[1]+1;
                decimales_3d[2]='0';
                if (decimales_3d[1]==':'){
                    decimales_3d[0]=decimales_3d[0]+1;
                    decimales_3d[1]='0';
                }
            }
        }
    }
}
}
}
}

void sacar_texto_ps(unsigned char data texto[]){
// Función que saca un texto por el puerto serie
//Declaración de variables locales
unsigned char data i=0;
//Fin declaración de variables locales
while(texto[i]!=0x00){
    transmitir(texto[i]);
    i=i+1;
}
}

unsigned char pulsada(void){
// Función que devuelve el registro del teclado, cuyos 4 LSB se tratan del valor de las columnas
// Declaración de variables locales

```

```
    unsigned char xdata *regtec;
    unsigned char data pulsad;
    // Fin de la delcaración
    regtec=REGTECLADO; // Se asigna al puntero la dirección del registro del CPLD correspondiente al teclado
    pulsad=*regtec;
    return pulsad;
}

void filas(unsigned char data fil){
    // Función que escribe el valor que se le pasa en las filas del teclado
    // Declaración de variables locales
    unsigned char xdata *regtec;
    // Fin de la declaración
    regtec=REGTECLADO; // Se asigna al puntero la dirección del registro del CPLD correspondiente al teclado
    *regtec=fil;
    espera_xms(5);    // Se espera 5 ms para que al CPLD le de tiempo a hacer lo que debe
}

void alimentar_lcd(void){
    // Función para alimentar la pantalla LCD
    // Declaración de variables locales
    unsigned char xdata *regtone;
    // Fin de la declaración de variables locales
    regtone=REGCONFTONE; // Se asigna al puntero la dirección del registro para la alimentación del LCD
    *regtone=0x04;    // Con la escritura de este valor en la dirección a la que apunta el puntero
}
    // se habilita la alimentación del LCD

void gestion_adc(void){
    /* Función para gestionar las conversiones realizadas por el conversor analógico digital
    eliminando 3 muestras para esperar a que el filtro se establezca, si se produce un cambio
    de peso, enviando por el resultado en kg al visualizador LCD y, si se quiere, por el puerto
    serie */
    // Declaración de variables locales
    double data peso_aux,V;
    long data peso_gramos;
```

```
bit bit_aux; // Bit donde se guarda el valor del último bits a eliminar
// Fin de la declaración
if(hab_t==1) sacar_texto_lcd(mensaje8,0xCB); // Se indica si se envía o no el resultado por el puerto serie
else sacar_texto_lcd(mensaje9,0xCB);
/* Se leen los registros de la conversión del ADC y se eliminan los bits que hagan falta,
usando el último bit que se elimina para redondear el resultado */
res=ADRESH;
res=res<<8;
res=res+ADRESM;
res=res<<8;
res=res+ADRESL;
res=res>>(bit_elim-1);
bit_aux=res;
res=res>>1;
if (bit_aux==1) res=res+0x01;
if(datos_a_descartar==0){
    pesos[indice_pesos]=res;
    if(indice_pesos==3){
        conversion=(pesos[0]+pesos[1]+pesos[2]+pesos[3])/4;
        conversion=conversion-offset;
        V=conversion*LSB;
        peso_aux=((V*300000)/(sensibilidad*excitacion))*cal;
        if (peso_aux>150000.0) peso_aux=150000.0; // Se limita el peso a 150kg
        peso_gramos=floor(peso_aux+0.5); // Se redondea el resultado
        if(peso_gramos>(peso_gramos_anterior+44) || peso_gramos<(peso_gramos_anterior-44)){
            parte_entera=peso_gramos/1000; // Parte entera del peso en kg
            parte_decimal=floor(((float)peso_gramos/1000-parte_entera)*1000+0.5); // Parte decimal del peso en kg
            envia_orden_lcd(0x01); // Se borra la pantalla, antes de mandar el nuevo dato
            sacar_texto_lcd(mensaje4,0x80);
            num_a_car3d(parte_entera);
            if (digitos==1){
                num_a_car2d(parte_decimal);
                ajuste_ASCII(2);
            }
        }
    }
}
```



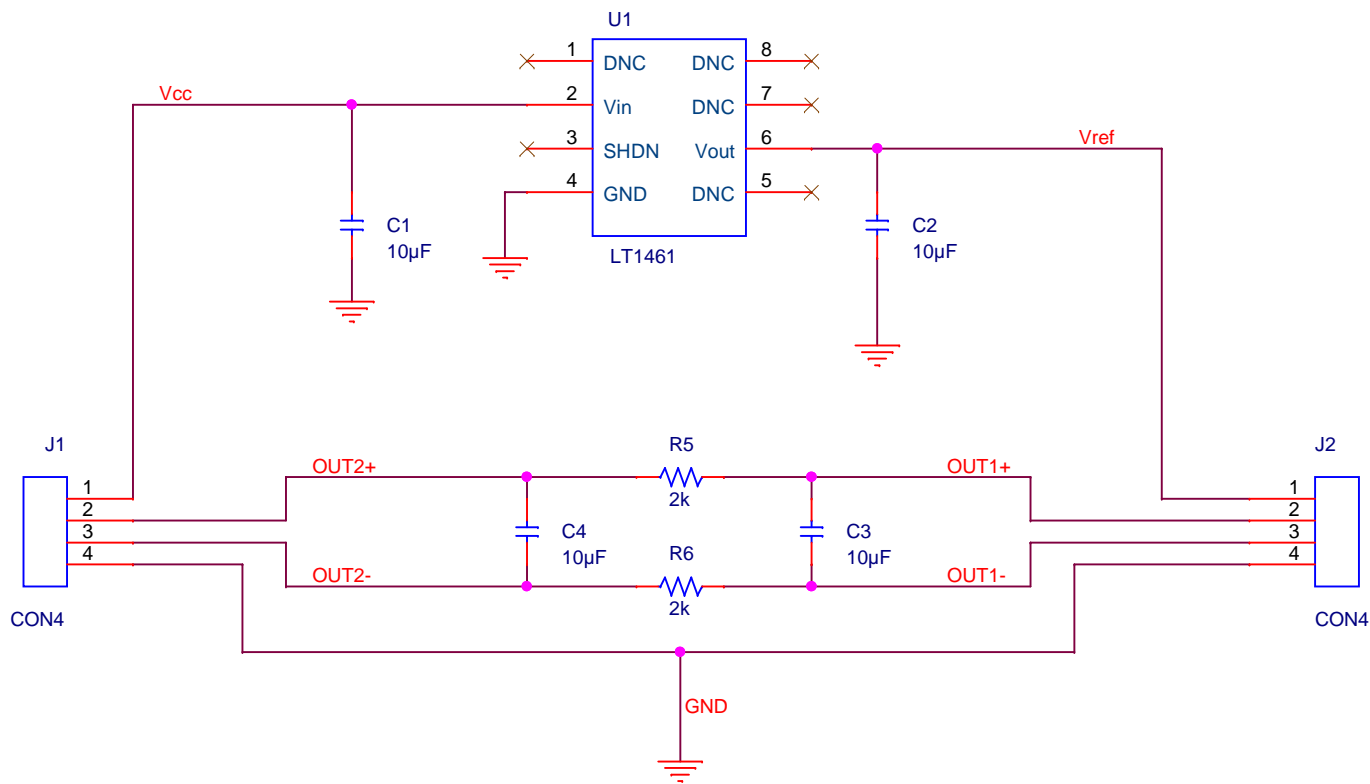
```
        num_a_car1d(parte_decimal);
        ajuste_ASCII(1);
    }
    sacar_texto_lcd(decimales_3d,0x89);
    if(hab_t==1){
        sacar_texto_ps(mensaje4);
        transmitir(decimales_3d[0]);
        transmitir(decimales_3d[1]);
        transmitir(decimales_3d[2]);
        transmitir(mensaje5);
        if (digitos==1){
            transmitir(decimales_2d[0]);
            transmitir(decimales_2d[1]);
        }
        else transmitir(decimales_1d);
        transmitir(mensaje6);
    }
    envia_orden_lcd(0x8C);
    envia_caracter_lcd(mensaje5);
    if (digitos==1){
        envia_caracter_lcd(decimales_2d[0]);
        envia_caracter_lcd(decimales_2d[1]);
    }
    else envia_caracter_lcd(decimales_1d);

    peso_gramos_anterior=peso_gramos;    // Se actualiza el valor de la variable peso_gramos_anterior
}
indice_pesos=-1;    // Se vuelve a cargar esta variable para que al salir de la función valga 0
}
indice_pesos=indice_pesos+1;
datos_a_descartar=4;    // Se vuelve a cargar esta variable para que se descargen 3 muestras
}
datos_a_descartar=datos_a_descartar-1;
```


Anexo 3. Planos

Los planos consisten en dos esquemas, un esquema eléctrico de la placa de ecualización, utilizada para acondicionar la señal de salida de la celda de carga, y para alimentar a la misma. El otro esquema consiste en un diagrama de bloques de todo el prototipo de control de la balanza.

De la placa UCOADP se han representado únicamente las conexiones necesarias, para más información sobre las conexiones se insta a consultar los planos del proyecto *“Diseño del hardware de un módulo portátil de adquisición de datos”* [10].



Fecha:	Autor	Firma	ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA
24/07/2017	José Antonio Olmedo Rivera		
Escala:	Diseño del interfaz de una balanza electrónica basada en una celda de carga		ESQUEMA 1
SE	PLACA DE ECUALIZACIÓN		

