UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE MÁSTER

DISEÑO DE UN SISTEMA EMPOTRADO PARA MEDIDA Y TRANSMISIÓN DE PROPIEDADES ELECTROFISIOLÓGICAS

Titulación: Máster Universitario de Ingeniería de Telecomunicación
Autor: María Luisa Barragán Pulido
Tutores: Tomás Bautista Delgado
Fecha: Julio 2016

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE MÁSTER

DISEÑO DE UN SISTEMA EMPOTRADO PARA MEDIDA Y TRANSMISIÓN DE PROPIEDADES ELECTROFISIOLÓGICAS

HOJA DE FIRMAS

Alumno/a

Fdo.: María Luisa Barragán Pulido

Tutor/a

Fdo.: Tomás Bautista Delgado

Fecha: Julio 2016

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE MÁSTER

DISEÑO DE UN SISTEMA EMPOTRADO PARA MEDIDA Y TRANSMISIÓN DE PROPIEDADES ELECTROFISIOLÓGICAS

HOJA DE EVALUACIÓN

Calificación: _____

Presidente

Fdo.:

Vocal

Fdo.:

Secretario/a

Fdo.:

Fecha: julio 2016

ESTE TRABAJO FIN DE MÁSTER CONTIENE LOS SIGUIENTES DOCUMENTOS:

DOCUMENTO Nº 1, MEMORIA

MEMORIA	62 páginas

DOCUMENTO Nº 2, PLIEGO DE CONDICIONES

PLIEGO DE CONDICIONES	6 páginas

DOCUMENTO Nº3, PRESUPUESTO

PRESUPUESTO GENERAL	11 páginas

DOCUMENTO Nº1 MEMORIA

MEMORIA DESCRIPTIVA

ÍNDICE DETALLADO

CAPÍTULO 1. INTRODUCCIÓN

La acuicultura se ha convertido en una industria millonaria, y más del 30% de los animales marinos que se consumen anualmente provienen hoy en día de estas granjas. Según la FAO (Organización de las Naciones Unidas para la Agricultura y la Alimentación), la acuicultura se está desarrollando tres veces más rápido que la producción agropecuaria, y lo más probable es que las granjas pesqueras pasen a un primer lugar a medida que nuestros caladeros naturales se agoten (1).

Según diversas fuentes, los peces criados en granjas pasan sus vidas en espacios insuficientes, y muchos sufren de infecciones parasitarias, heridas graves y otras enfermedades, estimando que el 40% de los peces puede morir antes de ser sacrificados para su comercialización.

En este marco, el proyecto "Sistema Subacuático RFID para acuicultura (SURF)", se está llevando a cabo actualmente en el IUMA (Instituto Universitario de Microelectrónica Aplicada) de la Universidad de Las Palmas de Gran Canaria. Está financiado por las ayudas de I+D+i "Retos investigación" del Programa Estatal de Investigación, Desarrollo e Innovación Orientada a los Retos de la Sociedad, y se estima una duración total de 36 meses.

Es un trabajo de investigación cuyo objetivo es diseñar un sistema subacuático RFID que ofrezca información sobre determinados signos vitales de los peces que se encuentran en piscifactorías y, además, que puedan tomarse durante un período relativamente prolongado. Este Trabajo Fin de Máster (en adelante TFM) se centra en uno de los elementos que deben diseñarse a tal fin: el sistema implantado en el pez. Éste deberá recoger la información de sus datos fisiológicos y presentarla al usuario en la forma que, como se verá más adelante, se convenga más adecuada.

1.1 MOTIVACIÓN

Para seguir evolucionando y mejorando al ritmo que ofrece la tecnología, conocer las condiciones en las que se desarrolla la actividad en las granjas pasa también por estudiar las situaciones a las que se pueden ver sometidos los peces y saber cómo afectan a distintos aspectos.

La sociedad en la que vivimos puede y tiene la obligación moral de garantizar que las condiciones en las que se desarrollen estas actividades sean dignas tanto para los propios seres vivos con los que trabajan como para las personas que se alimentarán de ellos. Reflejo de las condiciones de vida de los peces es la calidad de su carne, como se ha venido demostrando en diferentes investigaciones (2) (3) y que es consecuencia, entre otros, del estrés al que se someten.

Por tanto, la información que este sistema puede brindar es interesante y útil en cuanto que pudiera servir para garantizar una industria responsable y comprometida. Sin embargo, como todo, cómo utilizar la tecnología y con qué fin, es otro asunto.

1.2 OBJETIVOS

El objetivo es programar adecuadamente el microcontrolador (en adelante MCU) que, en el sistema implantado en el pez, se encargue de realizar las diferentes medidas de interés sobre los peces. Los objetivos detallados son:

- Leer datos sobre el electrocardiograma (en adelante ECG) y temperatura.
- Realizar un procesamiento de estos datos para obtener como resultado final algunos indicadores que sean de interés y representativos de estos datos.
- Guardar estos resultados en el sistema empotrado.
- Permitir ejecutar diferentes tareas, tareas simples o experimentos programados durante un período dado.
- Que el sistema sea lo más autónomo posible en términos de consumo de potencia.

1.3 ESTRUCTURA DE LA MEMORIA

Para facilitar la lectura del trabajo y explicar la organización del contenido, se incluye a continuación un breve resumen de cada capítulo:

ESTADO DEL ARTE

En este capítulo se comenta de manera resumida el contexto general, se plantea el problema general y concreto de este TFM y se hace un repaso de las tecnologías disponibles para el desarrollo de la solución.

DISEÑO DE LA SOLUCIÓN

Se definen dos tipos de diseño; el de la parte hardware y software. Cada una de estas partes incluye una descripción de cómo se ha ido dando forma a la solución para alcanzar los objetivos.

PRUEBAS Y RESULTADOS

Detalla pruebas básicas a las que se ha sometido el sistema diseñado y se presentan los resultados obtenidos junto con parámetros calculados a partir de ellos como el error cometido. Incluye, además, el análisis de potencia del sistema.

CONCLUSIONES

Consiste en un repaso global del desarrollo del trabajo y se especifica la consecución de los diferentes objetivos. También se aportan posibles líneas futuras de desarrollo interesantes para la continuación del trabajo.

CAPÍTULO 2. ESTADO DEL ARTE

2.1 INTRODUCCIÓN

En este capítulo se describe, grosso modo, en qué consiste el proyecto del que forma parte este TFM. Para comenzar se hace un repaso general de la situación actual de la acuicultura y, más concretamente, de la acuicultura inteligente. Una sección posterior se encargará de plantear y aclarar el problema que se presenta como motivo del TFM y, finalmente, se describe la filosofía de diseño que marcará las pautas para el desarrollo del mismo.

2.1 CONTEXTO GENERAL

El incremento anual de la producción global, concretamente de la acuicultura, se espera que continúe creciendo en el futuro. Una de las principales preocupaciones es la producción de alimentos seguros, lo cual se está viendo comprometido por el modo y la legislación que controlan cómo se desarrollan estas actividades en las piscifactorías. Existen numerosos factores a tener en cuenta y que deben regularse como por ejemplo el incremento del número de sustancias químicas que se utilizan y que contaminan el entorno acuático y los potenciales alimentos o la sobreproducción, masificación y los problemas de sanidad que pueden derivarse de ello. Todas estas circunstancias tienen efectos negativos sobre el bienestar de los peces, pudiendo resultar incluso contraproducente, económicamente hablando, para las propias fábricas (4).

En el gráfico 1 se muestra la evolución del suministro mundial de pescado comparando las producciones del sector de la acuicultura frente a las capturas en mar abierto desde 1970 hasta 2010. También refleja las previsiones de crecimiento a partir de 2010 hasta el año 2030 (5).

GRÁFICO 1.SUMINISTRO MUNDIAL DE PESCADO.

La idea de bienestar animal se abre paso en medio de una industria que lejos de mantenerse, aumenta día a día a pasos agigantados. Parece necesario, aunque esto ya viene planteándose desde mitad del siglo XX, mejorar y avanzar en el desarrollo de un sector que necesita cada vez más el apoyo de la tecnología, siempre en pro de una industria que debería ser responsable. La certeza de que los animales son sensibles plantea que su bienestar sea tenido en consideración, más aún cuando se encuentren bajo la supervisión humana.

El bienestar de los animales es una cuestión objeto de las interpretaciones sobre qué es moralmente aceptable para que los humanos posean y tengan animales para alimento, así como para experimentación, vestimenta y entretenimiento. Las conocidas como Cinco Libertades del Bienestar Animal, declaradas por el Gobierno de Inglaterra (Farm Animal Welfare Council, en 1979) se definen como garantías para una vida digna:

- Libres de temor y angustia.
- Libres de molestias físicas, frío o calor.
- Libres de expresar su comportamiento normal.
- Libres de dolor, lesión o enfermedad.

Los factores con un impacto negativo sobre el bienestar también tienen un impacto negativo sobre la calidad y salubridad de los alimentos producidos por las granjas. Por este motivo, es importante supervisar y cuantificar el bienestar de los animales bajo producción, para asegurar que éste lo más óptimo posible. Para los

productores este hecho supondría, además, mayores ingresos y reducción de los costes de producción. Los ingresos mayores vendrían no solo de la complacencia de los consumidores a la hora de pagar por una mayor calidad, sino porque también menos peces morirían en el proceso. Se estima que en los períodos de crecimiento del salmón en Noruega el 38% (4) de los peces muere por motivos que pueden atribuirse a un pobre planteamiento del sistema de producción y que podría mejorarse con técnicas de monitorización y control de determinados aspectos como pueden ser parámetros fisiológicos o químicos.

2.2 PLANTEAMIENTO DEL PROBLEMA

Para supervisar adecuadamente signos vitales en un pez se plantea una idea de sistema que, de una manera muy resumida, podemos caracterizar como se muestra en la siguiente figura (Figura 1):

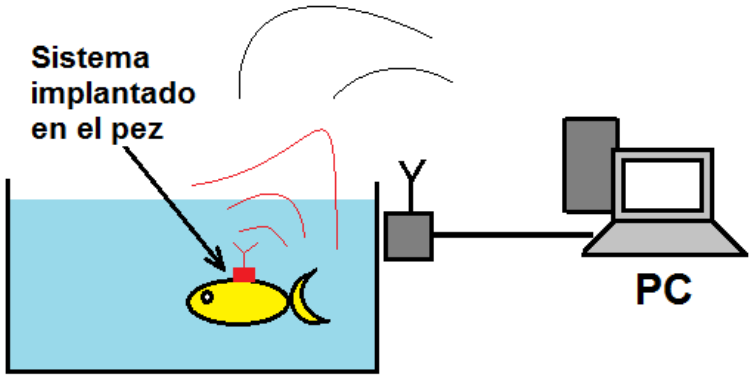

FIGURA 1. ESQUEMA GENERAL.

En esta Figura 1 encontramos, básicamente:

- Un PC desde el que se controlarán los experimentos a realizar con los peces y que permitirá recoger los resultados de dichos experimentos.
- Una antena que transmita las instrucciones que han de llegar al sistema implantado en el pez y que, a su vez se conectará al ordenador. Deberá estar capacitado también para recibir la información que le llegue en sentido inverso (sentido pez-PC).

- Sistema implantado en el pez: se encuentra en el cuerpo del pez y se encarga de recibir las instrucciones, ejecutarlas y devolver los datos solicitados o la respuesta pertinente a la estación base. A su vez está formado por:
    - Tag o etiqueta: incluye la antena (RFID) y la comunicación con el MCU y con la estación base.
    - MCU: se encarga del procesamiento de los datos. Comunica con el Tag y con el sistema de adquisición de datos.
    - Sistema de adquisición de datos: capturar las muestras a partir de las señales eléctricas producidas por el pez (ADC).

Existe una multitud de aspectos que deben estudiarse con detenimiento, desde los relacionados con el MCU (cómputo, algoritmo implementado y consumo de potencia) y el sistema de adquisición de datos (sensores y ADC), hasta los aspectos relacionados con la cobertura RFID (en medios diferentes de transmisión, como agua y aire) y la integración de los distintos componentes, entre otros.

En este TFM el problema que se plantea se reduce a cómo programar el MCU y a gestionar su relación con el sistema de adquisición de datos, un sensor específico para medidas del ECG y la temperatura. Asimismo, forma parte del problema que debe ser resuelto conseguir reducir los consumos de potencia al mínimo posible.

2.3 FILOSOFÍA DE DISEÑO

El diseño propuesto en este trabajo se ha estructurado en tres bloques principales:

- Adquisición de datos: para la medida de señales electropotenciales el sensor elegido es el ADS1298ECG-FE.
- Microcontrolador (MCU): para el control, gestión y procesamiento de tareas el MCU elegido es el KL25Z128VLK4.
- Interfaz básica de usuario: en principio, basándonos en los objetivos finales del proyecto completo, no es necesaria esta parte pero, dado que se trata de una solución inicial que aún no ha sido probada, resulta fundamental disponer de una interfaz básica para que el usuario pueda interactuar con el sistema.

A. MICROCONTROLADOR Y PLACA DE DESARROLLO

Un microcontrolador (Figura 2) es un circuito integrado que contiene todos los componentes de un ordenador muy básico y en dimensiones reducidas. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su pequeño tamaño, se le confiere la denominación de controlador empotrado.

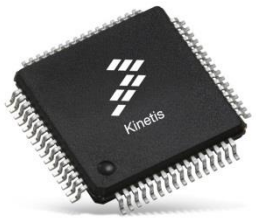

FIGURA 2. MICROCONTROLADOR KINETIS.

En su memoria sólo se encuentra el programa destinado a desarrollar una aplicación determinada; sus líneas de entrada/salida permiten la conexión de sensores y actuadores del dispositivo a controlar. Una vez programado y configurado el MCU sólo sirve para realizar la tarea que le ha sido asignada.

El MCU que se ha escogido para este trabajo es un microcontrolador fabricado por Freescale de la serie Kinetis L en un encapsulado de tipo 80 LQFP llamado KL25Z128VLK4 e integrado en la placa de desarrollo FRDM-KL25Z (Figura 3).

FIGURA 3. PLACA DE DESARROLLO FRDM-KL25Z.

La Placa Freedom de Freescale es una plataforma de desarrollo de bajo costo para MCUs de las series KL1x y KL2x, con núcleo ARM® Cortex™-M0+. La placa incluye acceso fácil a E/S del MCU, operación de bajo consumo de potencia e interfaz de depuración para programación de la memoria flash y control de ejecución. El KL25Z puede programarse usando entornos de desarrollo tales como IAR Workbench, Keil y Codewarrior de Freescale. También pueden utilizarse con entornos de desarrollo online como Mbed.

FRDM-KL25Z ha sido diseñada por Freescale para el prototipo de dispositivos que, especialmente, tienen limitaciones en cuanto a tamaño y precio. A continuación, en la figura 4 obtenida de la página oficial de Mbed se muestra un esquema con las diferentes señales accesibles desde los conectores al KL25Z.

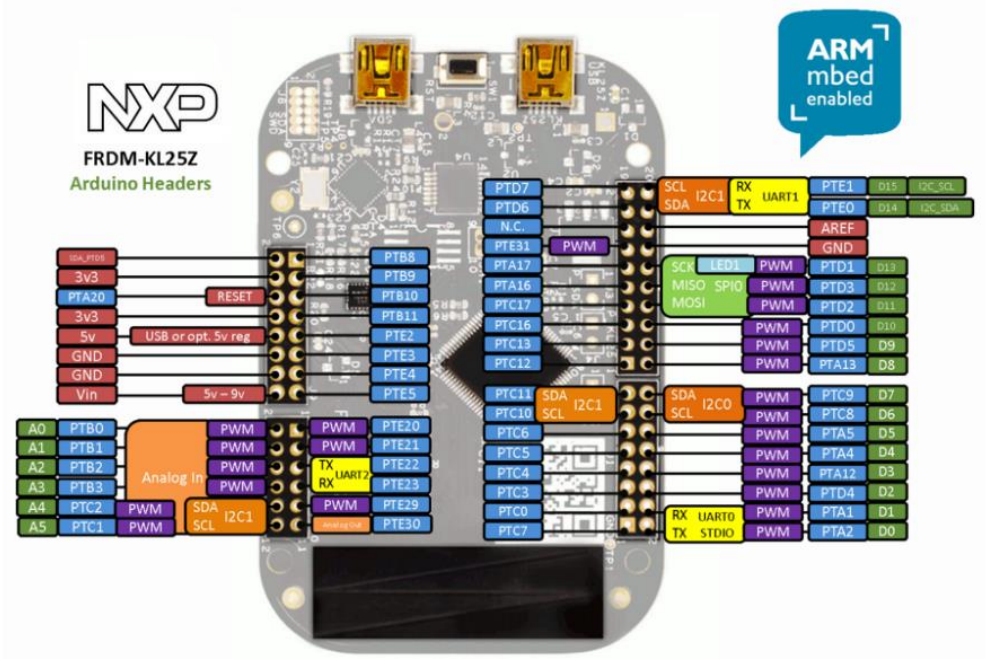

FIGURA 4. SEÑALES KL25Z.

Las características principales del MCU KL25Z (6) (7) (8)son:

- Núcleo ARM Cortex-M0+ de 32-bits.
  - Máxima Frecuencia de CPU: 48 MHz
  - Acceso rápido a puerto I/O, ciclo único
- Memorias:
  - Flash de 128 KB
  - SRAM de 16 KB
  - 64 B Cache
- Integración de sistemas:
  - Control y gestión de diferentes modos de potencia
  - Controlador de acceso directo a memoria (DMA)
  - Watchdog timer, computer operating properly (COP)
- Relojes:
  - Módulo de generación de reloj con FLL y PLL, para generación de reloj de sistema y CPU
  - Reloj de referencia interna de 4 MHz y 32 KHz

    - Oscilador de sistema con cristal externo o resonador
    - Oscilador RC de 1 KHz y baja potencia para RTC y COP watchdog.
- Periféricos analógicos:
    - Soporte de 16-bit SAR ADC con DMA
    - Soporte de 12-bit DAC con DMA
    - Comparador de alta velocidad
- Periféricos de comunicación
    - Dos interfaces SPI de 8-bit
    - USB dual-role controller with built-in FS/LS transceiver
    - Regulador de tensión USB
    - Dos módulos I2C
    - Una UART de baja potencia y dos módulos UART estándar
- Temporizadores
    - Un módulo temporizador/PWM de 6 canales
    - Dos módulos temporizador /PWM de 2 canales
    - Un temporizador de interrupción periódica (PIT) de 2 canales
    - Reloj de tiempo real (RTC)
    - Temporizador de bajo consumo (LPTMR)
    - Temporizador del reloj del sistema (System tick timer)
- Interfaces humano-máquina (HMI)
    - Controlador I/O de propósito general
    - Módulo hardware con interfaz de entrada sensor capacitivo de toque.

MODOS DE CONSUMO

El controlador de gestión de potencia (PMC) ofrece múltiples opciones que permiten al usuario optimizar el consumo de potencia en función de las necesidades de la aplicación.

Cada modo RUN tiene su correspondiente modo WAIT y STOP. Los modos de espera son similares a los modos sleep de ARM. Los modos Stop (VLPS, STOP) son similares al sleep deep del ARM. El modo de operación de baja potencia (VLPR) puede reducir drásticamente la potencia trabajando con frecuencias menores, si la aplicación lo permite.

Los tres modos de potencia primaries son run, wait y stop, que derivan en otros de baja potencia para poder utilizarse con aplicaciones que lo requieran. En la tabla 1

obtenida del manual de referencia del dispositivo se comparan los distintos modos disponibles (7).

Chip mode	Description	Core mode	Normal recovery method
Normal run	Allows maximum performance of chip. Default mode out of reset; on-chip voltage regulator is on.	Run	—
Normal Wait - via WFI	Allows peripherals to function while the core is in sleep mode, reducing power. NVIC remains sensitive to interrupts; peripherals continue to be clocked.	Sleep	Interrupt
Normal Stop - via WFI	Places chip in static state. Lowest power mode that retains all registers while maintaining LVD protection. NVIC is disabled; AWIC is used to wake up from interrupt; peripheral clocks are stopped.	Sleep Deep	Interrupt
VLPR (Very Low Power Run)	On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency. Reduced frequency Flash access mode (1 MHz); LVD off; in BLPI clock mode, the fast internal reference oscillator is available to provide a low power nominal 4MHz source for the core with the nominal bus and flash clock required to be <800kHz; alternatively, BLPE clock mode can be used with an external clock or the crystal oscillator providing the clock source.	Run	—
VLPW (Very Low Power Wait) -via WFI	Same as VLPR but with the core in sleep mode to further reduce power; NVIC remains sensitive to interrupts (FCLK = ON). On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency.	Sleep	Interrupt
VLPS (Very Low Power Stop)-via WFI	Places chip in static state with LVD operation off. Lowest power mode with ADC and pin interrupts functional. Peripheral clocks are stopped, but OSC, LPTMR, RTC, CMP, TSI can be used. TPM and UART can optionally be enabled if their clock source is enabled. NVIC is disabled (FCLK = OFF); AWIC is used to wake up from interrupt. On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency. All SRAM is operating (content retained and I/O states held).	Sleep Deep	Interrupt
LLS (Low Leakage Stop)	State retention power mode. Most peripherals are in state retention mode (with clocks stopped), but OSC, LLWU, LPTMR, RTC, CMP,, TSI can be used. NVIC is disabled; LLWU is used to wake up. **NOTE:** The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully exit stop mode on an LLS recovery. All SRAM is operating (content retained and I/O states held).	Sleep Deep	Wakeup Interrupt[1]
VLLS3 (Very Low Leakage Stop3)	Most peripherals are disabled (with clocks stopped), but OSC, LLWU, LPTMR, RTC, CMP, TSI can be used. NVIC is disabled; LLWU is used to wake up. SRAM_U and SRAM_L remain powered on (content retained and I/O states held).	Sleep Deep	Wakeup Reset[2]
VLLS1 (Very Low Leakage Stop1)	Most peripherals are disabled (with clocks stopped), but OSC, LLWU, LPTMR, RTC, CMP, TSI can be used. NVIC is disabled; LLWU is used to wake up. All of SRAM_U and SRAM_L are powered off.	Sleep Deep	Wakeup Reset[2]
VLLS0 (Very Low Leakage Stop 0)	Most peripherals are disabled (with clocks stopped), but LLWU, LPTMR, RTC, TSI can be used. NVIC is disabled; LLWU is used to wake up. All of SRAM_U and SRAM_L are powered off. LPO disabled, optional POR brown-out detection	Sleep Deep	Wakeup Reset[2]

TABLA 1. MODOS DE POTENCIA.

A continuación, se muestra el diagrama de estados para los diferentes modos de potencia del KL25Z (Figura 5). Resulta muy interesante para realizar el estudio de potencia conocer el margen de movimiento que existe para cada estado.

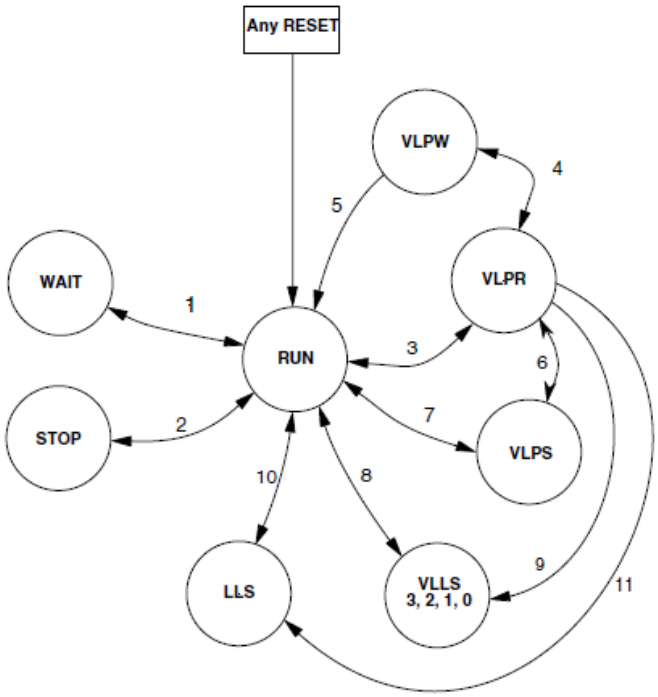

FIGURA 5. DIAGRAMA DE ESTADOS: MODOS DE POTENCIA.

B. SENSORES

Los sensores constituyen el principal medio de enlace entre el mundo físico y los circuitos electrónicos que permiten controlar las variables físicas. Un sensor es un dispositivo capaz de medir magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas.

En la práctica los sensores más empleados son aquéllos que dan una respuesta basada en la naturaleza eléctrica de la materia, en los que cualquier variación de un parámetro físico (temperatura, humedad, distancia, etc.) viene acompañada por la variación de un parámetro eléctrico (resistencia, capacitancia, inductancia, etc.).

Básicamente, con los sensores se convierte la magnitud física a medir en otra más fácil de manipular, si bien, una vez obtenida esta señal, se suele someter a un proceso de acondicionamiento y amplificación para ajustarla a las necesidades de la carga exterior o de la circuitería de control.

La resolución de un sensor es el menor cambio en la magnitud de entrada que se aprecia en la magnitud de salida y, la precisión, el máximo error esperado en la medida.

TIPOS DE SENSORES

Pueden ser de indicación directa, que no precisen una fuente de energía auxiliar (por ejemplo un termómetro de mercurio), o pueden estar conectados a un indicador (posiblemente a través de un convertidor analógico a digital, un ordenador y un display) de modo que los valores detectados puedan ser leídos por un usuario. Este último tipo se denomina sensores activos o moduladores y precisan, a parte del medio de visualización, una fuente de alimentación externa.

Los sensores más utilizados son los sensores cuya salida es una señal analógica de tensión o corriente que está continuamente variando. Los rangos más habituales de salida de tensión analógica son +10V, +5V, ±10V, ±5V y ±1V. Otro tipo de sensores con salida analógica cada vez más utilizados son los que proporcionan una salida no en tensión sino en corriente.

SENSOR PARA MEDIDAS BIOPOTENCIALES

El sensor elegido ha sido el ADS1298 de Texas Instruments (Figura 6). Se trata de un sensor multicanal, con muestreo simultáneo de 24 bits, que dispone de convertidor analógico-digital (ADC) y que lleva integrado un amplificador de ganancia programable (PGA), referencia interna, y también un oscilador. La familia de sensores a la que pertenecen los ADS1294/6/8 incorpora todas las características requeridas comúnmente en los electrocardiogramas médicos (ECG) y en las aplicaciones de electroencefalograma (EEG), aunque ésta última no sea, en principio, de nuestro interés.

FIGURA 6. SENSOR ADS1298.

Dispone, además, de un multiplexor de entrada flexible por canal que se puede conectar de forma independiente a señales generadas internamente para pruebas,

medidas de temperatura, o para detección *lead-off*. Además, para la derivación de la señal de *Right Leg Drive* (RLD) cualquier configuración de entrada de los canales puede ser seleccionada. El ADS1298 puede operar a velocidades de datos de hasta 32kSPS (muestras por segundo). En la Figura 7 se muestra el diagrama de bloques el ADS1298.

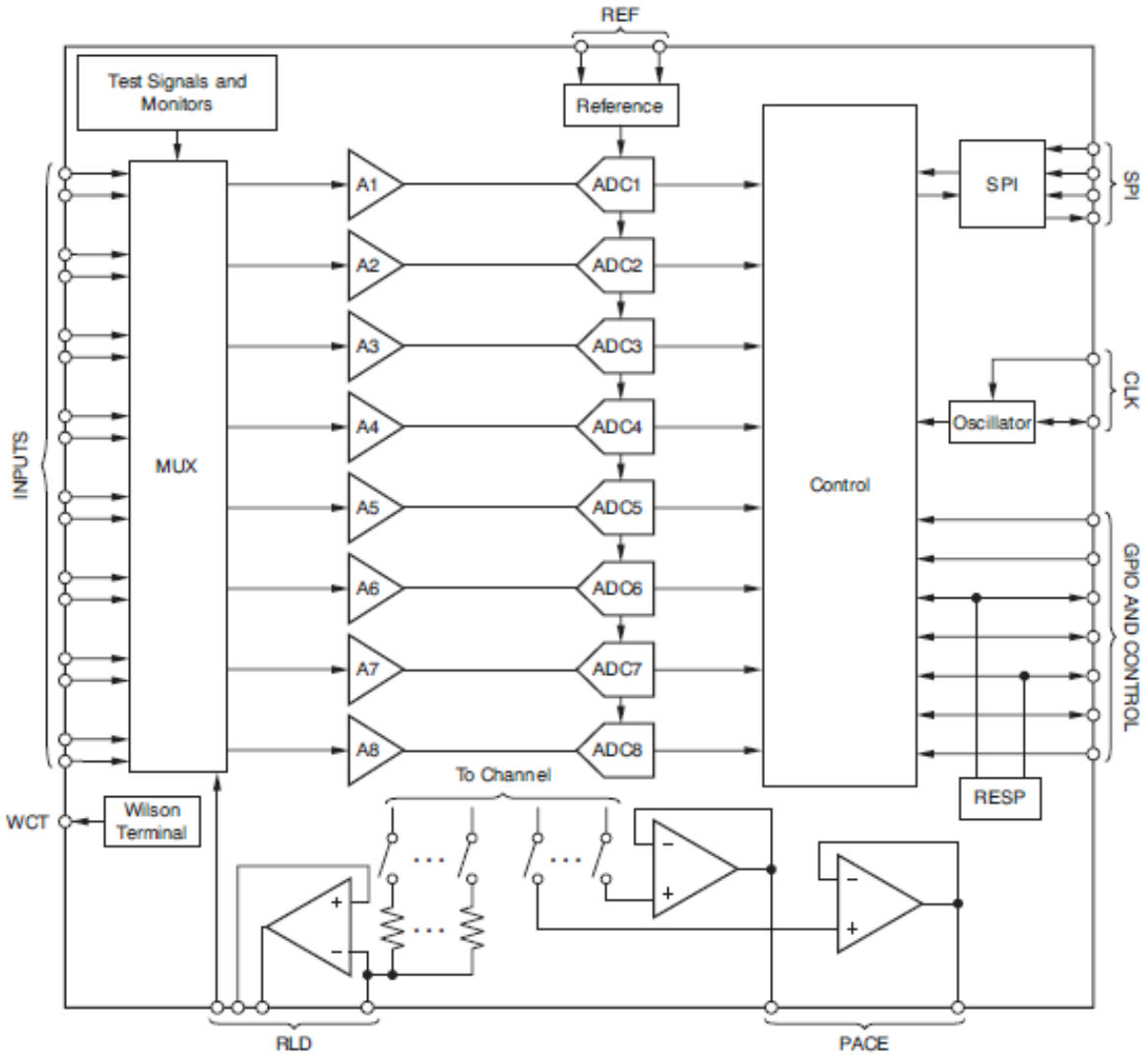

FIGURA 7. DIAGRAMA DE BLOQUES DEL ADS1298

Se trabajará con una placa denominada ADS1298ECG-FE EVM (Figura 8), que integra este sensor, y está configurada para su uso con la plataforma de evaluación TI MMB0. En nuestro caso, no utilizaremos esta segunda placa.

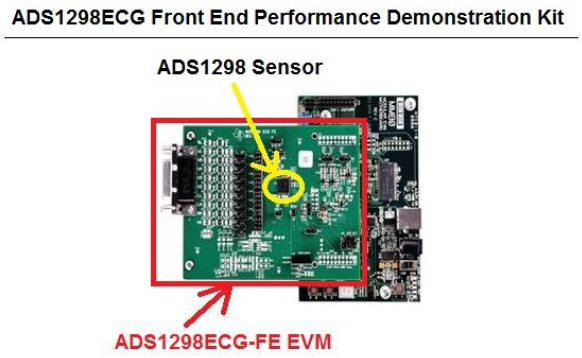

FIGURA 8. ADS1298ECG-FE PDK.

La ADS1298RECG-FE es una placa de circuito de cuatro capas que se puede utilizar como placa de demostración para aplicaciones de ECG de 12 derivaciones estándar con una configuración de entrada de 10 electrodos o bien se puede proporcionar cualquier tipo de señal directamente a través de diferentes configuraciones de los *jumpers* (9). Por otra parte, ofrece otros circuitos útiles para pruebas, tales como referencias externas, relojes y resistencias de lead-off. La Figura 9 muestra el diagrama de bloques funcional del *Front-End* que contiene el ADS1298.

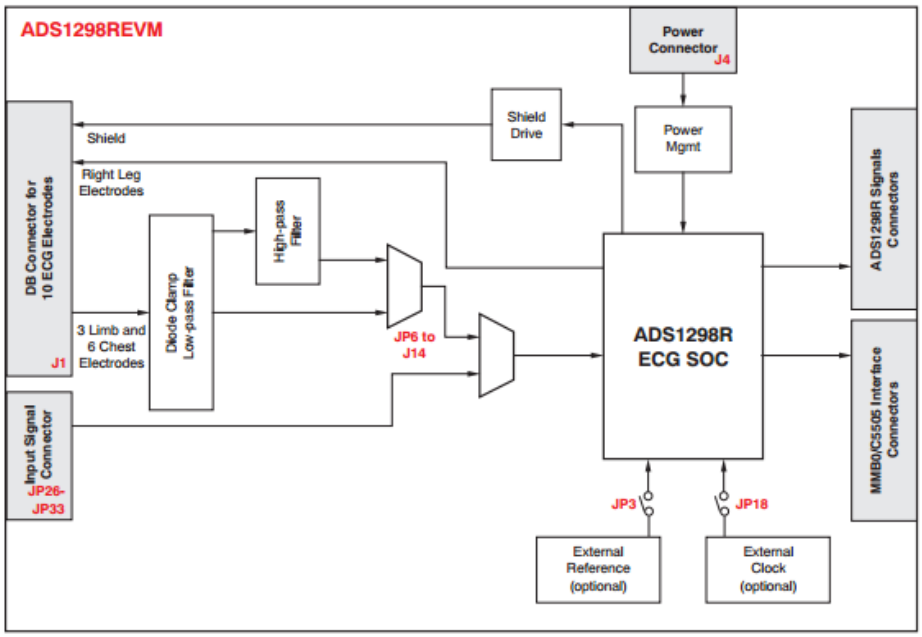

FIGURA 9. DIAGRAMA DE BLOQUES ADS1298ECG-FE EVM.

Las entradas analógicas en el ADS1298 son diferenciales. Suponiendo una ganancia PGA = 1, la entrada (INP – INN) puede variar entre *–Vref* y *+Vref*. En la tabla 2 se explica la correlación entre la entrada analógica y los códigos digitales que ocaciona en la salida.

INPUT SIGNAL, V_{IN} (AINP – AINN)	IDEAL OUTPUT CODE (1)
≥ V_{REF}	7FFFFFh
$+V_{REF}/(2^{23} - 1)$	000001h
0	000000h
$-V_{REF}/(2^{23} - 1)$	FFFFFFh
≤ $-V_{REF}$ $(2^{23}/2^{23} - 1)$	800000h

TABLA 2. CORRELACIÓN: ENTRADA ANALÓGICA-CÓDIGOS DIGITALES.

La conducción de la entrada analógica puede realizarse con dos métodos diferentes: *single-ended* o diferencial (Figura 10 y Figura 11). Para un rendimiento óptimo, se recomienda utilizar el ADS1298 en una configuración diferencial.

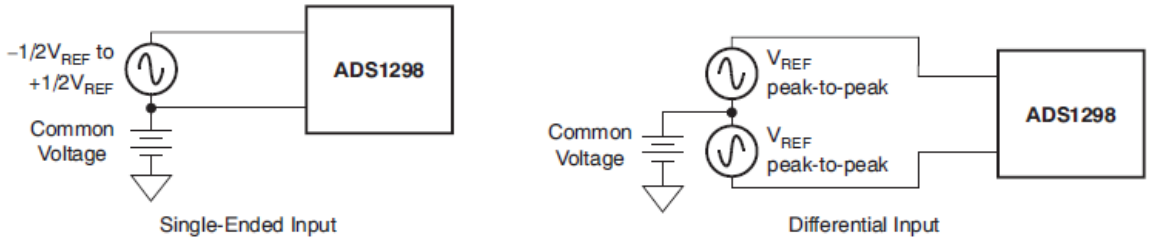

FIGURA 10. MÉTODOS PARA ENTRADA DE SEÑALES ANALÓGICAS DEL ADS1298: SINGLE ENDED O DIFERENCIAL

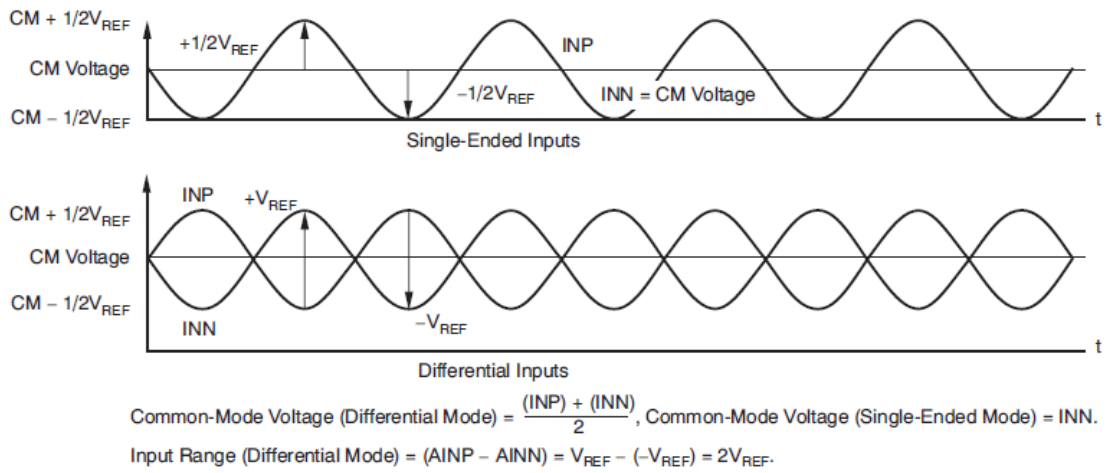

FIGURA 11. USO DEL ADS1298 PARA LOS DISTINTOS MODOS DE ENTRADA.

SEÑAL DE TEMPERATURA

La placa ADS1298ECG-FE contiene un sensor de temperatura integrado en el chip. Este sensor utiliza dos diodos internos siendo la diferencia entre sus tensiones proporcional a la temperatura absoluta.

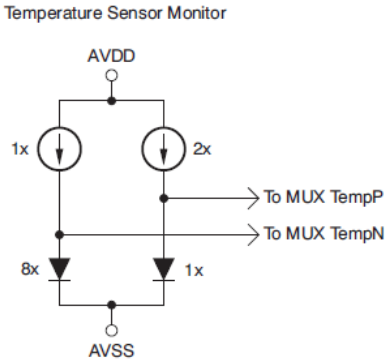

FIGURA 12. MEDIDA DEL SENSOR DE TEMPERATURA A LA ENTRADA.

Para obtener el valor de la temperatura en grados Celsius se aplica la Ecuación 1.

$$\text{Temperature (°C)} = \left(\frac{\text{Temperature Reading } (\mu V) - 145{,}300 \mu V}{490 \mu V/°C} \right) + 25°C$$

ECUACIÓN 1. TEMPERATURA.

C. INTERFAZ DE USUARIO

Para controlar el correcto funcionamiento y las tareas que realizará el sistema es necesario, en una fase preliminar, como en la que se centra este trabajo, y antes de avanzar con las siguientes partes del proyecto, que se pueda interactuar con el sistema por medio de una interfaz básica de usuario basado en el control del flujo de la aplicación mediante el terminal del ordenador.

Para ello será necesario hacer uso de la UART del KL25Z, así como del "puente" OpenSDA USB conectado al ordenador y, en éste, un software como el Hyperterminal de Windows o el software PuTTY que permita la comunicación con el MCU (básicamente para mandar instrucciones, comprobar resultados, acceder a memoria, etc).

En nuestro caso concreto el programa utilizado ha sido el PuTTy v.067.1010.0 (Figura 13). Éste es un emulador de terminal que permite conectar con máquinas remotas y ejecutar programas a distancia. PuTTY se conecta como cliente a múltiples protocolos, como SSH, Telnet o Rlogin. Será de gran ayuda para saber qué información maneja la placa y, en caso de que así se desee, almacenarla en un fichero para posteriormente representar los datos de las medidas en un programa como GNUPlot.

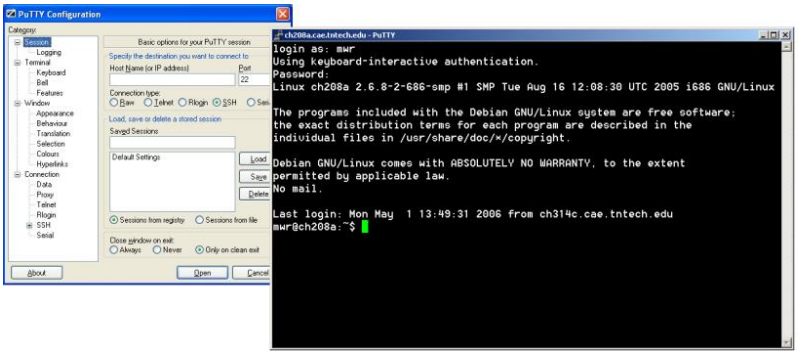

FIGURA 13. EMULADOR DE TERMINAL PUTTY.

CAPÍTULO 3. DISEÑO DE LA SOLUCIÓN

3.1 INTRODUCCIÓN

En este capítulo se detallan las decisiones tomadas para alcanzar los objetivos propuestos. Se pretende dar forma a una solución que establezca y gestione las relaciones microcontrolador-sensor, comunicaciones, procesamiento y almacenamiento de la información y gestión de la energía.

3.2 DISEÑO HARDWARE

Para explicar cómo se ha organizado este diseño se puede dividir en los siguientes bloques principales:

- Alimentación
- Procesamiento
- Visualización

A. ALIMENTACIÓN

Se centra en suministrar niveles de tensión estables que son necesarios para el correcto funcionamiento del resto de bloques del diseño. Distinguiremos:

- Alimentación de la placa
- Alimentación del sensor
- Conexionado

ALIMENTACIÓN DE LA PLACA FRDM-KL25Z

La placa KL25Z puede alimentarse de diferentes maneras (ver tabla 3) a saber: con conectores USB, a través del pin Vin (I/O header) de 4.3-9V, con una pila de botón en la propia placa o con alimentación de 1.71-3.6V externa de la placa usando el pin de 3.3V del I/O header.

En nuestro caso hemos optado por alimentarla a través de la conexión OpenSDA USB (5V) desde el ordenador. Este tipo de alimentación se regula en la propia placa por medio de un regulador lineal de 3.3V que produce o provee la alimentación de potencia principal. Como se indicó, en la tabla 3 se muestran los requisitos de alimentación de las diferentes opciones. En rojo se muestra la opción elegida.

Supply Source	Valid Range	OpenSDA Operational?	Regulated on-board?
OpenSDA USB (J7)	5V	Yes	Yes
KL25Z USB (J5)	5V	No	Yes
V_{IN} Pin	4.3-9V	No	Yes
3.3V Pin	1.71-3.6V	No	No
Coin Cell Battery	1.71-3.6V	No	No

TABLA 3. REQUISITOS ALIMENTACIÓN FRDM-KL25Z.

ALIMENTACIÓN DEL SENSOR DEL ADS1298ECG-FE

La alimentación de la placa de evaluación ADS1298ECG-FE en principio llega desde otra placa llamada MMB0 EVM, por el conector J4. Sin embargo, en nuestro caso, prescindiremos de esta segunda placa ya que nuestra finalidad es trabajar simplemente con el sensor ADS1298, sin necesitar las prestaciones que nos ofrece la MMB0 EVM. Por tanto, la alimentación que necesita el ADS1298 llegará desde la KL25Z a través del conector J4. En las hojas de características se especifica que la alimentación puede ser con señal unipolar (alimentación analógica de +3V a +5V, AVDD/AVSS) o bipolar (alimentación de +/-1.5V a +/-2.5V). Además es necesario para la operación del dispositivo alimentación digital (DVDD de +1.8V a 3.3V). La configuración se realiza mediante los siguientes jumpers (Tabla 4).

Jumper	Function	Settings
JP2	AVDD supply source	2-3: single supply operation (AVDD = +3V)
JP20	AVSS supply source	1-2: single supply operation (AVSS = 0V (AGND))
JP24	DVDD supply select	2-3: DVDD supply = 3.3V

TABLA 4. ALIMENTACIÓN: CONFIGURACIÓN DE LOS JUMPERS.

CONEXIONADO

Como se ha indicado, el ordenador alimentará a la FRDM-KL25Z a través del conector OpenSDA USB (J7) con +5V y, ésta, a su vez, alimentará al sensor por medio de conectores de salida específicos que se encuentran en la placa. Los conectores utilizados para alimentar al ADS1298ECG-FE se corresponden exactamente con los pines P3V3 (3v3), P5V_USB (5V) y GND, y se conectarán directamente al conector J4 del ADS1298, en sus entradas Vcc 5V, Vcc 3.3V y GND. En la Tabla 5 se muestran estas conexiones.

FRDM-KL25Z		ADS1298ECG-FE	
I/O header	**Pin**	**I/O header**	**Pin**
J9	P3V3	**J4**	Vcc 3.3V
J9	P5V_USB	**J4**	Vcc 5V
J2	GND	**J4**	GND

TABLA 5. RELACIÓN DE PINES DE ALIMENTACIÓN PARA ADS1298.

La figura 14 representa un esquema de estas conexiones.

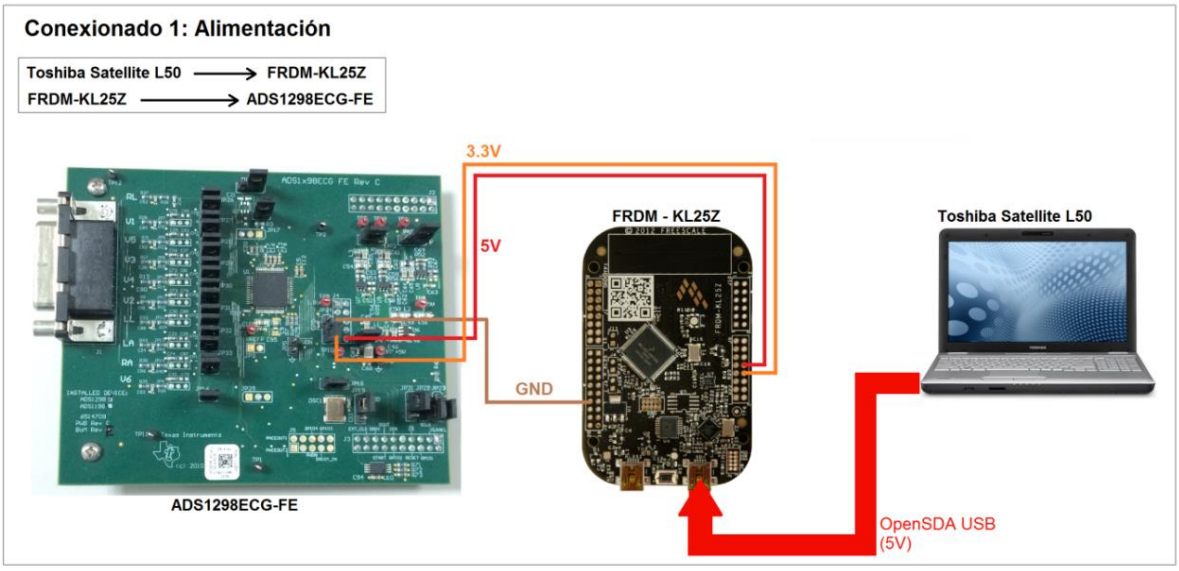

FIGURA 14. CONEXIONADO 1 (ALIMENTACIÓN).

B. PROCESAMIENTO

En esta sección el diseño hardware debe encargarse de que el MCU KL25Z128VLK4 pueda procesar adecuadamente toda la información procedente tanto del sensor como del ordenador, este último a través del terminal. Las principales funciones que ha de desarrollar el MCU son:

- Configurar ADS1298
- Calcular el valor del ritmo cardíaco y de la temperatura
- Almacenar resultados en la memoria flash
- Decidir el modo de operación más adecuado en cada etapa del experimento para ahorro de potencia

Para realizar todas estas funciones, es preciso configurar dos tipos de comunicaciones: la primera entre el KL25Z y el ADS1298 (por medio de SPI y otras señales I/O) y, la segunda, entre el usuario y el KL25Z (UART). En la tabla 6 se muestra la configuración de los pines para la comunicación SPI y señales I/O con el ADS1298.

FRDM-KL25Z			ADS1298ECG-FE		
I/O header	**Pin**	**Name**	**I/O header**	**Pin**	**Name**
J2	PTD7	MISO	**J3**	13	DOUT
J2	PTD6	MOSI	**J3**	11	DIN
J2	PTD5	SCLK	**J3**	3	CLK
J2	PTC12	CS	**J3**	2	CS
J2	PTC13	I/O	**J3**	8	RESET
J2	PTC16	I/O	**J3**	14	START
J2	PTD0	I/O	**J3**	15	DRDY

TABLA 6. DISTRIBUCIÓN DE PINES SPI.

En la figura 15 se muestran las conexiones SPI:

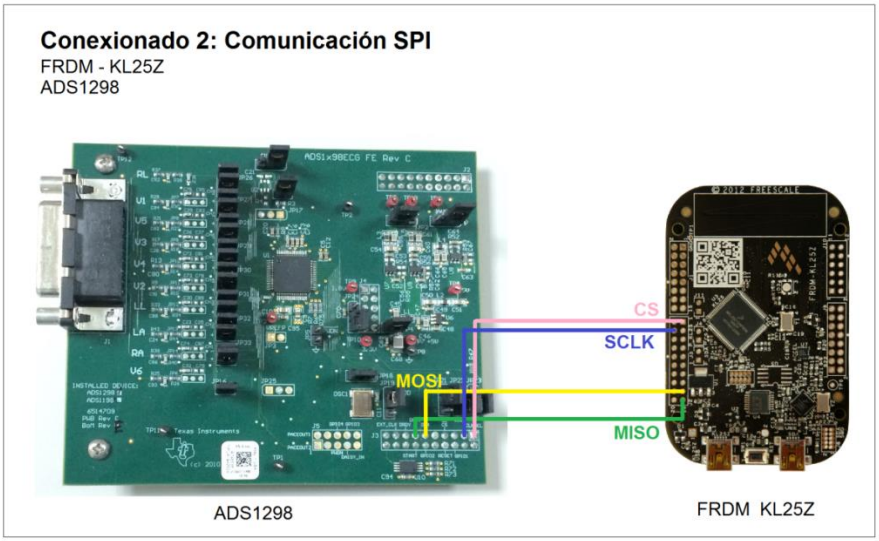

FIGURA 15. CONEXIONADO SPI.

Las señales I/O de control, conectadas como indica la Figura 16, vienen definidas en las hojas de características del ADS1298ECG-FE (9). Son tres:

- START: señal de entrada (al sensor) que permitirá comenzar la captura de datos.
- RESET: señal de entrada (al sensor) para realizar una inicialización del dispositivo.
- DRDY: señal negada de salida (del sensor) que indicará al KL25Z que hay datos disponibles. Cabe mencionar que esta señal, a diferencia de las anteriores, es imprescindible para el funcionamiento del sistema.

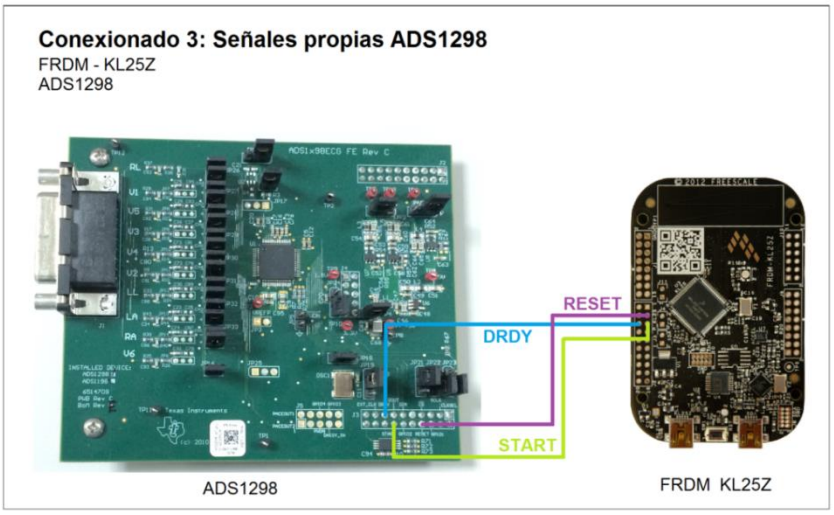

FIGURA 16. CONEXIONADO 3 (CONTROL ADS1298)

Sobre el segundo tipo de comunicación a establecer cabe mencionar que existe ya una comunicación habilitada entre el OpenSDA del KL25Z y los pines PTA1 y PTA2 (correspondientes a la UART0). Varias de las aplicaciones OpenSDA proporcionadas por Freescale, por omisión, proporcionan una interfaz de USB CDC (Communications Device Class) que sirve de puente para comunicaciones serie entre el host USB y esta interfaz serie en el KL25Z.

El resultado final de todas las estas conexiones se representa en la figura 17 que se corresponde con una fotografía real del sistema con el que se ha trabajado.

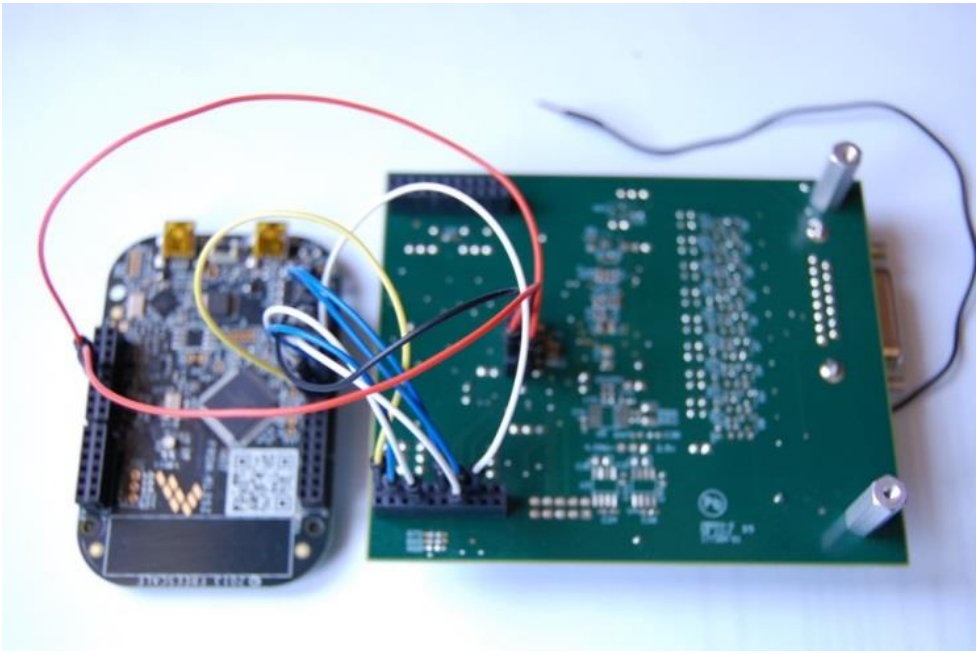

FIGURA 17. FOTOGRAFÍA DEL SISTEMA.

D. VISUALIZACIÓN

La visualización de todo el proceso puede verse en pantalla gracias a, como se ha especificado más arriba, la conexión establecida entre la placa del KL25Z y el PC. Recordamos que se establecía mediante la conexión ofrecida por el OpenSDA USB a través de la UART0. Las características de esta comunicación son:

- Velocidad de comunicación: 9600 baudios.
- Trama: 8 bits de datos.
- Bit de parada: 1 bit de parada.
- Paridad: Sin paridad.

En la comunicación sólo se emplean caracteres ASCII imprimibles, lo que permite el control del KL25Z desde cualquier software de comunicación para PC como, por ejemplo, el Hyperterminal de Windows o PuTTY.

Su funcionamiento se basa en las siguientes pautas:

- El KL25Z envía al terminal el menú imprimible en pantalla para que el usuario pueda elegir la opción que quiere ejecutar (Figura 18).

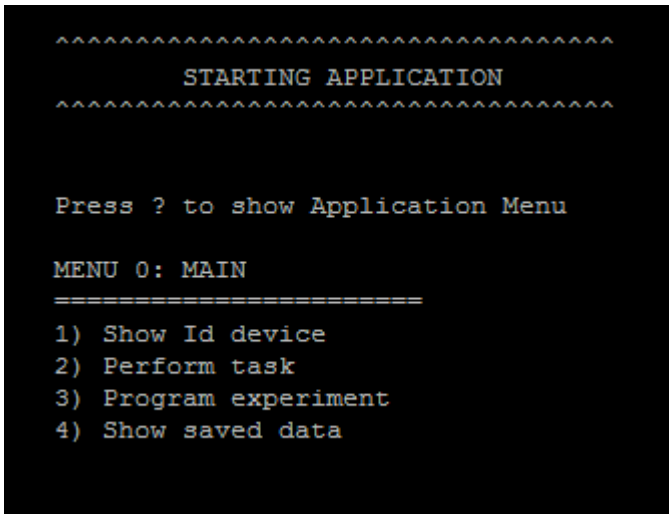

FIGURA 18. MENÚ PRINCIPAL.

- A través del teclado del PC se envía la opción elegida al KL25Z, que se traducirá en una orden o tarea.
- En pantalla, y según el tipo de tarea elegida para ejecutar, aparecerán o no los valores de las medidas y, como regla general para todos los casos, siempre el resultado final obtenido y un informe codificado. Este informe contendrá información relevante sobre la prueba realizada y se guardará en memoria flash, en caso de que el usuario seleccione dicha opción.

3.3 DISEÑO DEL SOTFWARE

Como ya se ha visto a lo largo de todo el trabajo, este diseño se basa en el microcontrolador KL25Z como elemento central para la integración y toma de decisiones. Gracias a su programabilidad como microcontrolador que es, ofrece una gran versatilidad para trabajar en multitud de aplicaciones. En este proyecto se ha diseñado el software que se va a ejecutar en lenguaje C/C++ utilizando el software de desarrollo µVision v.5 de Keil.

El diseño que se ha seguido puede estructurarse como un bloque principal que gestiona una serie de bloques secundarios con funciones específicas. Estos bloques secundarios pueden dividirse en:

- Funciones de inicialización del KL25Z.
- Funciones para la inicialización y uso del ADS1298.

- Configuración SPI1: ADS1298-KL25Z.
- Configuración de la UART0: KL25Z-PC.
- Configuración del RTC: para controlar tiempos.
- Configuración del LPTMR: temporizador de bajo consumo que puede mantenerse activo cuando el MCU se encuentra en modo sleep.
- Configuración del *Multipurpose Clock Generator*(MCG): gestión de los distintos modos de reloj.
- Configuración de los modos de consumo (SMC): que dependerán del momento o la tarea en la que nos encontremos durante la ejecución del programa.
- Configuración de la memoria flash: para almacenamiento de información cuando corresponda.
- Funciones para el procesamiento de los datos tanto de ECG como de temperatura recibidos del sensor.

A. ENTORNO DE TRABAJO

El entorno de trabajo elegido es el Kit de Desarrollo de Microcontroladores (MDK) de Keil. Proporciona un entorno de desarrollo software para un amplio rango de dispositivos basados en MCU de ARM, Cortex-M y Cortex-R, siendo la última versión la número 5 y la que hemos utilizado. En el paquete se incluye un Entorno de Desarrollo Integrado y Depurador µVision IDE/Debugger (Figura 19), Compilador ARM C/C++, y componentes middleware esenciales.

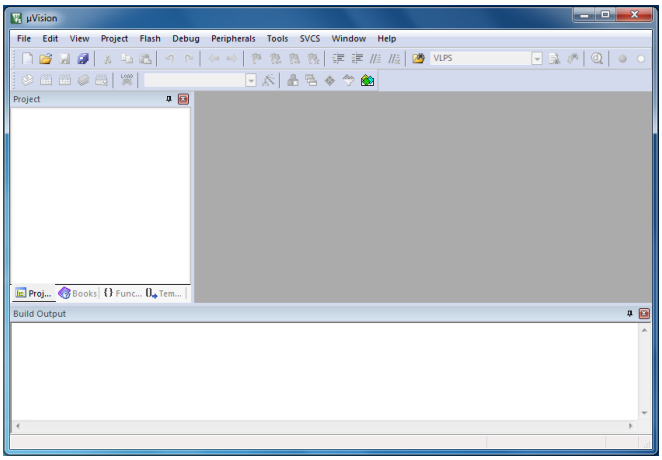

FIGURA 19. ENTORNO DE PROGRAMACIÓN µVISION V.5.

El entorno de desarrollo µVision 5 engloba una serie de componentes mediante los que se puede llevar a cabo los procesos inherentes al desarrollo y depuración de aplicaciones. Los componentes son los siguientes:

- IDE de µVision: gestión de proyectos, edición y ajuste de opciones.
- Compilador y Macroensamblador: C51 y A51. Generan los ficheros objeto reubicables que serán procesados por el modulo montador.
- Gestor de librerías: LIB51.
- Montador/ubicador: BL51. Crea un archivo o fichero tipo ELF/DWARF a partir de los creados por el compilador o macroensamblador. Este archivo se usa para:
    - Programar la ROM tipo Flash de un MCU, u otros tipos de memoria (.hex)
    - Simulación o depuración en placa
    - Verificación real del programa mediante un emulador integrado en el circuito
- Depurador µVision: depurador simbólico a nivel del código fuente, para seguimiento de la ejecución.

B. CÓDIGO

El trabajo planteado consiste en un sistema que permita de manera autónoma almacenar información sobre el ritmo cardíaco de un pez y su temperatura en diferentes momentos a lo largo del tiempo.

Para cada una de las tareas que deberá ejecutar el MCU ha sido preciso implementar distintas funciones. Todas ellas se integran en el programa principal, directa o indirectamente, y se relacionan entre sí atendiendo a la siguiente pseudomáquina de estados (Figura 20) donde, más allá de simples estados, se pretende mostrar la idea de funcionamiento.

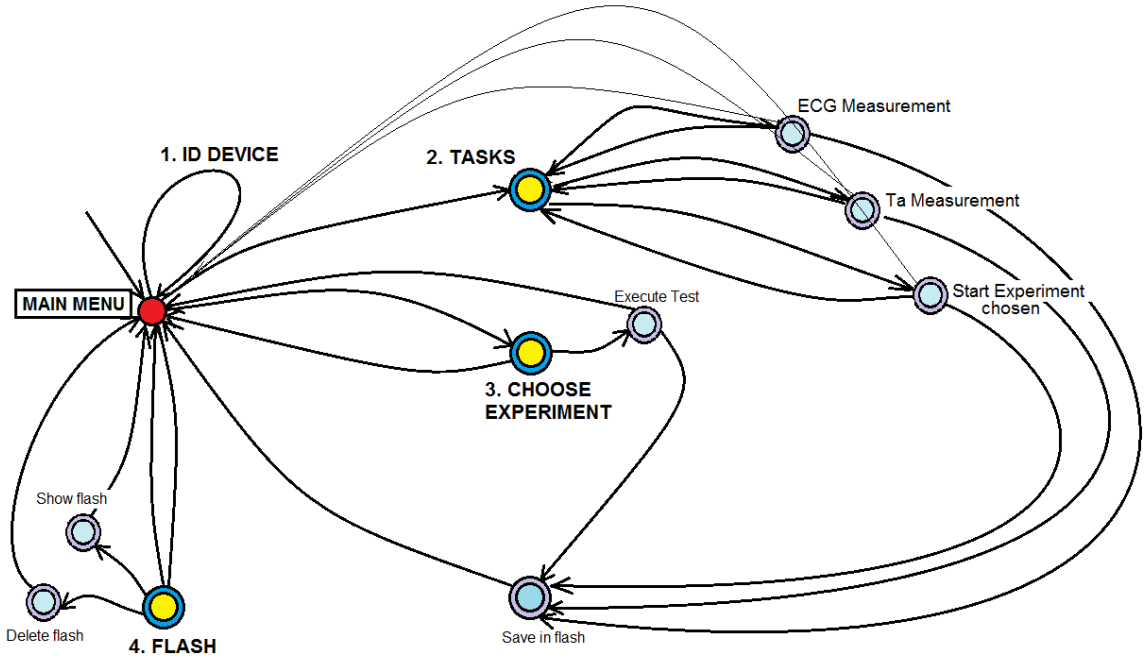

FIGURA 20. IDEA DE FUNCIONAMIENTO: PSEUDOMÁQUINA DE ESTADOS.

PROGRAMA PRINCIPAL

El programa principal se basa en un bucle infinito que está continuamente ejecutándose, esperando, cuando proceda, respuesta por parte del usuario. Dependiendo de la opción elegida por éste se ejecutará una parte del código u otra. Sin embargo, antes de llegar a este bucle principal es necesario hacer las inicializaciones y configuraciones adecuadas.

La primera información que le llega al usuario es a través de la pantalla y se trata de la información sobre el estado en el que se encuentra el MCU, el modo de reloj y el modo de consumo actuales (Figura 21), ya que precisamente queremos que trabaje en modo VLPR; de bajo consumo cuando así se lo pueda permitir.

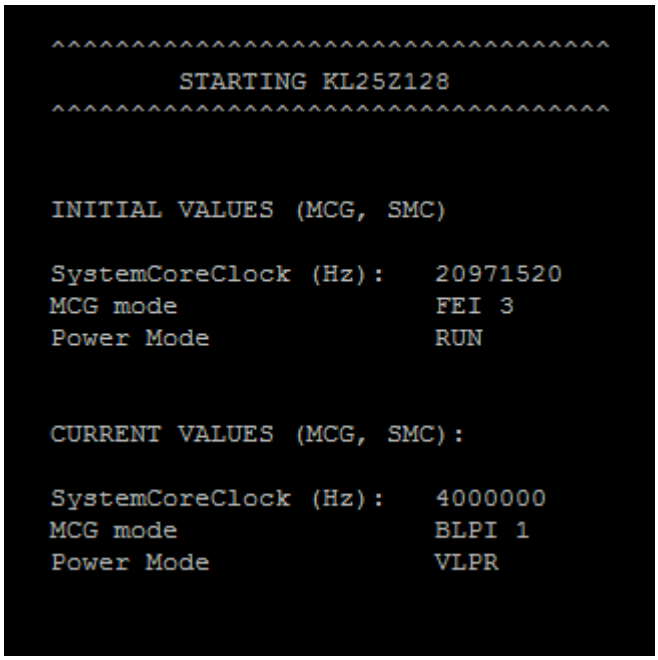

FIGURA 21. INICIO PROGRAMA.

Las funciones necesarias para ello suponen la primera parte del programa principal. El código utilizado para ello es el que se muestra a continuación y ha sido dividido en tres partes:

I. INIT MLK, UART0 and RTC

Averigua el reloj inicial del sistema, inicializa el KL25Z, la UART0 y el RTC configurando los registros que corresponden.

```
SystemCoreClockUpdate();                    // Check system core clock
SystemCoreClock_pre = SystemCoreClock;      // and MCG mode
mcgmode_pre = what_mcg_mode();              // save in variables

mkl_init();                                 // Initialize KL25Z
uart0 init();                               // Initialize UART0
rtc_config();                               // Config Real Time Clock
```

II. MCG and SMC

El reloj, que por defecto se inicia en modo FBI, se cambia a BLPI. Y el modo de potencia que inicialmente es RUN se cambia a VLPR.

```
Fbi2blpi();                                 // Pass from FBI mode
mcgmode_pos = what_mcg_mode();              // clock to BLPI
run2vlpr();                                 // Pass from RUN mode
```

III. Init SPI1, PORTD Interrupt and ADS1298

Se inicializa el SPI1, se habilita la interrupción que corresponde a la disponibilidad de datos nuevos y se inicializa el ADS1298 con las configuraciones que corresponden.

```
Spi1b_init();                               // Initialize SPI1
init_PortD_interrupt();                     // PORTD interrupt enable
ads1298_init();                             // Initialize ADS1298
```

A continuación lo que sigue es el bucle infinito, el en ocasiones llamado *while(1)*. Una vez alcanzado se esperará a que el usuario introduzca el símbolo "?" para mostrar un primer menú donde habrá diferentes opciones (puede verse en la Figura 18, sección "Diseño hardware"). El MCU esperará, entonces, a que se seleccione una de ellas.

IV. WHILE(1)

```
Comienzo del bucle infinito. Espera que se introduzca carácter.

While(1){
  receiveBuff = uart0_getchar();              //save user option from PC
  if(receiveBuff == '?')                      //first wait a ? symbol
      menu1(submenu);
```

El código de esta parte se basa en una variable que almacena el valor corriente del menú en el que nos encontramos. En este caso, se trata del menú raíz, el principal, y tiene asignado el valor 0 de manera que una vez mostrado espere la respuesta del usuario para cambiar a un nuevo menú (como puede ser el de tareas o el de elegir experimento).

V. Options from Main Menu

```
El menú principal ofrece cuatro opciones diferentes. La primera se
encarga de devolver el identificador del dispositivo directamente. Las
otras tres se diferencian en que deben mostrar al usuario otros menús que
corresponden, según el caso. De manera que utilizan funciones menux()
para imprimir por pantalla las opciones de los nuevos menús y esperan la
respuesta del usuario nuevamente pero ya en la siguiente repetición del
bucle infinito. Al repetir el bucle la variable submenú será igual al
menú que corresponda y las funciones que se apliquen serán en función de
la siguiente elección del usuario.

If (submenu == 0){                            // Main menu
      switch(receiveBuff){                    // User choice
            case '1':                         // ID Device
                  spi1b_clear_CS();           // SPI: cs = 0
                  rtc_ms();                   // delay 1 ms
                  ads1298_id();               // show ID device
                  submenu = 0;                // remain in this menu
                  menu1(submenu);             // show options of menu
                  break;                      // again
            case '2':
                  menu1(2);                   // show menu of tasks
                  submenu = -2;               // and wait next choice
                  break;
            case '3':
                  submenu = -3;               // show menu of experiment
                  menu4();                    // and wait next choice
                  break;
            case '4':
                  submenu = -4;               // show menu of FLASH
                  menu5();                    // and wait next choice
```

```
                break;
            default:
                break;
            }
    }
```

MENU 1: Identificación del dispositivo

De las hojas de características del ADS1298 (9) sabemos que la respuesta que debemos recibir por parte del sensor es, en hexadecimal: c0 00 00 92. En la figura 22 se muestra la respuesta que obtenemos.

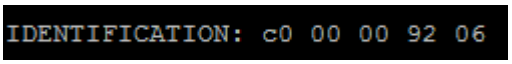

FIGURA 22. RESPUESTA ID ADS1298.

MENU 2: Realizar tarea

Si por el contrario el usuario pulsa la opción 2 del menú principal pasa al menú de tareas (Figura 23), que permitirá al ejecutar: una medida puntual del valor del ritmo cardíaco (opción 1), una medida puntual del valor de la temperatura corporal (opción 2), comenzar a ejecutar un experimento programado previamente (opción 3) o por último volver al menú principal (opción x). En caso de no haberse programado previamente el experimento a ejecutar en la opción 3, se ejecutará por defecto el experimento número 2 que consiste en lo que llamamos un *full* (ID + ECG + Ta) realizado cada 3 horas durante 1 mes.

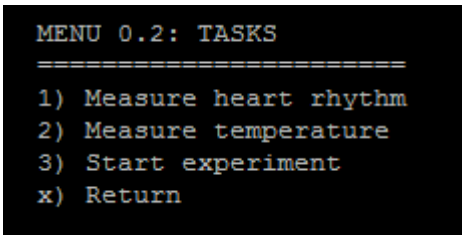

FIGURA 23. MENU 2: TAREAS.

Si el usuario ha llegado a este punto, se espera una nueva elección: ¿Cuál de las posibles tareas quiere realizar? Para ello se dispone de una función llamada *task(int "opción elegida")* que se encargará de realizar una u otra y que se estudiará en detalle más adelante, a fin de conseguir una mejor comprensión del texto.

VI. Options from Task Menu

```
if (submenu == 2){
      switch(receiveBuff){
            case '1':                           //punctual ecg measurement
                  task(1);                      //call task() function
                  menu6();                      //menu6 is to save/discard
                  submenu = -5;                 //submenu=5 for flash opt
                  break;
            case '2':                           //punctual ta measument
                  task(2);                      //call task() function
                  menu6();
                  submenu = -5;
                  break;
            case '3':                           //start experiment exect
                  task(3);
                  if(num_exp == 1){             //must distinguish between
                        menu6();                //2 options:
                        submenu = -5;           //experiment1 is a test to
                        }else{                  //execute immediately and
                              submenu = 0;      //it's possible to save it
                              menu1(submenu);   //or not.
                              }                 //other  experiments  must
                  break;                        //return to main menu when
                                                //finished     and     are
                                                //automatically saved
            case 'x':
                  submenu = 0;                  //return to main menu
                  menu1(submenu);               //and show options again
                  break;
            default:
                  break;
                  }
            }
```

MENU 3: Identificación del dispositivo

Si la opción pulsada por el usuario es la número tres, en ese caso, el menú mostrado será el que se muestra en la figura 24.

```
MENU 0.3: PROGRAM
=======================
1) EXP1 - TEST
2) EXP2 - FULL 3h       (each 15s for testing)
3) EXP3 - FULL 6h
4) EXP4 - FULL 12h
5) EXP5 - FULL 24h
x) Return
```

FIGURA 24. MENU 3: SELECCIONAR EXPERIMENTO.

Hay que hacer una aclaración sobre el EXP2: inicialmente debería estar programado para ejecutarse cada tres horas, sin embargo ha sido reducido a 15 segundos para hacer viable

la comprobación de su funcionamiento. El código que gestiona las diferentes opciones es el que se muestra a continuación.

VII. Options from Program Experiments Menu

La primera opción es que realiza un test inmediato del funcionamiento del dispositivo utilizando la función Test(). Esta función revisará que la comunicación SPI funciona, que recibe señales a la entrada del multiplexor y por último que los valores de temperatura respetan unos límites coherentes. El resto de opciones, relacionadas con diferentes experimentos, se encarga de actualizar la variable *num_exp* y presentar de nuevo el menú principal. La variable *num_exp* es una variable global que se utilizar dentro de la función *task(int)* para saber cuál de los experimentos realizar. Para ejecutar los experimentos automáticos (EXP2-5) es necesario volver al menú de tareas y elegir la opción "Comenzar experimento".

```
If (submenu == 3){
      switch(receiveBuff){
            case '1':                           //Test experiment
                  num_exp = 1;
                  rtc_ms();
                  ads1298_testInput();          //config ADS in test mode
                  Test();                       //call to Test() function
                  menu6();                      //when    finished    show
                  submenu = -5;                 //option of save/discard
                  break;
            case '2':
                  num_exp = 2;                  //save   selected   num_exp
                  submenu = 0;                  //and return to main menu
                  menu1(submenu);
                  break;
            case '3':
                  num_exp = 3;                  //save   selected   num_exp
                  submenu = 0;                  //and return to main menu
                  menu1(submenu);
                  break;
            case '4':
                  num_exp = 4;                  //save   selected   num_exp
                  submenu = 0;                  //and return to main menu
                  menu1(submenu);
                  break;
            case '5':
                  num_exp = 5;                  //save   selected   num_exp
                  submenu = 0;                  //and return to main menu
                  menu1(submenu);
                  break;
            case 'x':
                  submenu = 0;                  //save   selected   num_exp
                  menu1(submenu);               //and return to main menu
                  break;
            default:
                  break;
                  }
```

MENU 4: Memoria Flash

La cuarta opción es la del menú de la memoria flash (Figura 25). Permite mostrar los datos que hayan sido guardados (opción 1), borrar todo (opción 2) y volver al menú inicial (una vez más, opción x).

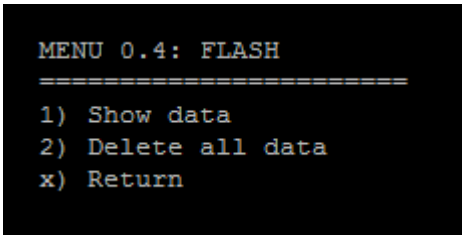

FIGURA 25. MENU 4: FLASH.

VIII. Options from FLASH menu

Hace uso de funciones específicas de la flash. Otras funciones utilizadas son las que permiten cambiar el modo de consumo del MCU. Es necesario realizar estos cambios dado que el uso de la flash está restringido o limitado en modo VLPR.

```
If (submenu == 4){
      switch(receiveBuff){
            case '1':                          //show data
                  flash_show(address);         //call flash_show(address)
                  submenu = 0;                 //function
                  menu1(submenu);              //return to main menu
                  break;
            case '2':
                  vlpr2run();                  //to clear first change
                  flash_eraseSector(address);  //power mode to RUN mode
                  run2vlpr();                  //next it's possible to
                  submenu = 0;                 //delete. Finally return
                  menu1(submenu);              //to VLPR mode and main
                  break;                       //menu
            case 'x':
                  submenu = 0;
                  menu1(submenu);
                  break;
            default:
                  break;
                  }
            }
```

MENU x: Guardar/Descartar

El último de los menús es el que aparece después de ejecutar cada tarea y que permiten guardar o descartar los resultados obtenidos (Figura 26). Cabe mencionar que en los

experimentos automáticos de larga duración los resultados se guardan de manera automática.

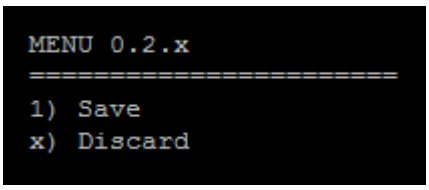

FIGURA 26. MENU GUARDAR.

IX. Save/Discard Menu

De nuevo aquí es necesario cambiar el modo de consumo de VLPR a RUN para poder guardar en flash.

```
If (submenu == 5){
      switch(receiveBuff){
            case '1':
                  vlpr2run();                  //change to RUN mode
                  Ram2Flash(flash);            //call Ram2flash to save
                  run2vlpr();                  //return to VLPR mode
                  submenu = 0;                 //return to main menu
                  menu1(submenu);
                  break;
            case 'x':                          //return to main menu
                  submenu = 0;
                  menu1(submenu);
                  break;
            default:
                  break;
                  }
            }
```

Para terminar esta sección hacemos referencia a los informes que se crean automáticamente después de cada tarea y que, salvo en los experimentos de larga duración, el usuario puede decidir guardar o descartar. Se trata de informes que guardan información sobre qué tarea se realizó, los resultados que se obtuvieron y si se encontraron errores o no durante la ejecución. Para llevar a cabo esta idea se ha establecido la palabra de 32 bits de la memoria flash como unidad de almacenamiento de información, asociando a cada una de las tareas un código, como el que aparece en la tabla 7.

Id	Código	Medida
1	000	ECG
2	001	Temperatura
3	010	Test
4	011	Full 3h
5	100	Full 6h
6	101	Full 12h

7	110	Full 24h
8	111	x

TABLA 7. CÓDIGO DE TAREAS.

La estructura establecida para el almacenamiento de la información se ve en la figura 27, donde los tres primeros bits se corresponden con la identificación de la tarea o medida, los tres siguientes informan de posibles errores que se hayan encontrado y desde el bit número 23 al 8 se utilizan para guardar los resultados obtenidos tanto del ritmo cardíaco como de la temperatura.

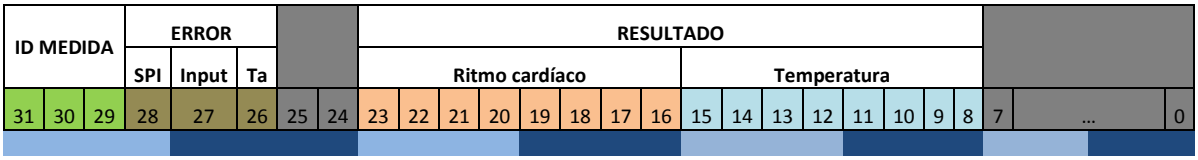

FIGURA 27. ESTRUCTURA DE ALMACENAMIENTO DE PALABRAS EN FLASH.

FUNCIONES ESPECÍFICAS

En este apartado se comenta el funcionamiento de las siguientes funciones específicas que aparecen en la siguiente tabla (Tabla 8).

Nombre	Descripción
void task(int)	Gestiona la tarea concreta que se elige ejecutar. Es una función muy genérica que se encarga de organizar de manera lógica y coherente las llamadas a funciones que serán necesarias para alcanzar los objetivos propuestos. Todas las funciones que se describen a continuación son usadas por task().
void ECG_10Hz()	Gestiona exclusivamente todo lo que tiene que ver con la medida del ritmo cardíaco, independientemente de la duración, el experimento o tipo de tarea que se haya seleccionado. Incluye la función proc_data_ECG-10Hz.
float proc_data_ECG_10Hz(uint_8 *data)	Devuelve el período de la señal para una medida. Se encarga de "limpiar" el vector inicial donde se guardan los datos, prepararlo, y hacer el propio recuento de pulsos para el posterior cálculo del ritmo cardíaco.
void Ta()	Configura el ADS1298 para medir temperatura y gestiona, los valores calculados por proc_data_Ta(). Es el equivalente al ECG_10Hz().
void proc_data_Ta()	Calcula propiamente el valor de la temperatura.

TABLA 8. FUNCIONES ESPECÍFICAS.

A continuación se presenta el código comentado de las funciones mencionadas en la tabla 8.

I. TASKs

Al tratarse de una función con fines "organizativos" lo que se presenta para facilitar la lectura es el pseudocódigo empleado. Puede verse que en función de *num* se realizará una u otra de las opciones presentadas en el menú 2.La opción 3 "Comenzar experimento", además, incluye la variable *num_exp* que, como se mencionó, guardaba el experimento que se había programado previamente en el menú 3.

```
Void task(int num){
  int I;
  flash = 0;
  ads1298_init();
  switch(num){
       case 1:
         MEDIDA ECG
         break;
       case 2:
         MEDIDA TEMPERATURA
         break;
       case 3:
         switch(num_exp){
              case 1:
                EXPERIMENTO TEST
                break;
              case 2:
                for(I = 0; i<240; i++){
                     EXPERIMENTO FULL 3h
                     CONSTRUCCION DEL INFORME
                     GUARDAR EN FLASH
                     DORMIR
                     }
                     break;
              case 3:                                    //120 rounds/month
                     ...                                 //and 21600sec
                break;                                   sleeping
              case 4: //se repite 60 y duerme 43200
                     ...                                 //60 rounds/month
                break;                                   //and sleeping for
              case 5: //se repite 30 y duerme 86400       //43200sec
                     ...
                break;                                   //30 rounds/month and
              default:                                   //sleeping for
                break;                                   86400sec
                }
              break;
         default:
           break;
           }
         }
```

Pasando a las medidas de ECG, por diversos motivos que se explican en este párrafo se estimó oportuno realizar seis medidas de 6.004 segundos cada una, que en total suman 36.024. Esto se debe limitaciones de capacidad del dispositivo que se

encontraron a la hora de programar la aplicación. En otros términos, se detectaron errores cuando se definían variables de mayor longitud limitándonos, según pudimos comprobar mediante ensayo-error, a crear un vector 52ufa donde queríamos guardar la información que se recibía del ADS, con una longitud máxima de 6000 variables de tipo uint8_t. Con esta importante limitación sobre la mesa se decidió plantear varios métodos diferentes que permitieran reconstruir una señal que fuera equivalente a 36 segundos aproximadamente de duración.

¿Por qué 36 segundos?

De las hojas de características del ADS1298 (9) puede obtenerse el formato en que transmite la información este dispositivo. En la figura 28 se puede ver de manera más intuitiva esta idea mediante valores a modo de ejemplo, para el caso de una transmisión que se efectuara correctamente:

- STATUS: 1100 + LOFF_STATP + LOFF_STATN + bits[4:7] del registro GPIO
- DATA: 24 bits por canal, en complemento a2 y MSB.

Status	C0	00	00	00	00	00
Data	2	e	f	0	3	1

FIGURA 28. FORMATO DE DATOS.

Puede deducirse que entonces para almacenar una sola muestra son necesarios 6 bytes (3 de estado + 3 de información). El cálculo de cuántas muestras podemos tomar con un vector de longitud 6000 es inmediato:

$$Muestras = \frac{6000B}{6B/muestra} = 1000muestras$$

Si además la tasa de muestreo del ADS1298 está configurada para 250 ksps tendremos medidas de:

$$Tmedida = \frac{1000muestras}{250kmuestras/segundos} = 4segundos$$

En el código que se muestra a continuación se ha incluido un retardo de 6ms para que, intentando preservar la calidad de la señal, se alargue el tiempo de captura de cada medida. Al incluir este retardo podemos calcular:

$$Tiempo\ de\ cada\ muestra = \frac{1s}{250ksps} = 4\mu s$$

$$4\mu s + 6ms(retardo) = 6.004ms$$

Por tanto, el tiempo que tardará en realizar cada medida de 1000 muestras será 6.004s. A continuación, se presenta el código de la función que gestiona las medidas de ritmo cardíaco.

II. ECG Measurement

```
void ECG_10Hz(void){
  int I, aux, j, flag = 0, o =0;                  //Declarations
  float ppm =0, period, frec, periodomedio = 0;   //Initilizations
  float vect[6];
                                                  //to save the result of
  cont2=0;                                        //each measurement
  cont = 0;                                       //number of written B
                                                  //number of round
  spi1b_clear_CS();
  rtc_ms();                                       //cs=0 SPI
  rtc_ms();

  ads1298_startcapt();
  rtc_reset();                                    //START command to ADS
  tmp = RTC_TPR;
  while(flag<6){                                  //save current moment
      aux = cont2;                                //6 measurem. Of 6.004s
while(dataavailable != 1){}
for (I = aux; i< (aux + 6); i++) {                //when there's data to
*(bufA+i)=spi1b_write_read(0x00);                 //save
        cont2 ++;                                 //read with SPI
}                                                 //save in bufA
      if(cont2 == 6000){
        rtc_reset();                              //if bufA is completed
        vect[flag] = proc_data_ECG_10Hz(bufA);    //init RTC
        cont2 = 0;                                //save result of round
        cont = 0;                                 //in vect[]
        flag ++;
        }
for(j= 0; j<6; j++)                               //delay of 6ms
  rtc_ms();                                       //increment round
      cont++;
      }
      rtc_ms();
      ads1298_stopcapt();
      spi1b_set_CS();                             //STOP cmd to ADS
      rtc_reset();                                //cs=1 No SPI
                                                  //init RTC
```

```
      for(i=0; i<6;i++){
        periodomedio = periodomedio + vect[i];    //mean of 6 results
        }
      periodomedio = periodomedio/6;
      for (I = 0; i<6; i++){
        if(vect[i]<(periodomedio   +   150)   &&  //filter  to  dispersed
vect[i]>(periodomedio - 150)){                    //value, +/-15ms
      period = period + vect[i];                  //for doing a new mean
      o++;
      }
  }
  if(o == 0) period = periodomedio/10000;         //if  not  possible  new
      else period = (period/o)/10000;             //mean do old method
                                                  //if possible, do mean
                                                  //again

  frec = 1/period;                                //calculate frequency
  ritmo = (int)(60/period);                       //calculate       heart
                                                  rhythm
  if(ritmo<=0) flash = (1<<27);
    else flash = 0 | (ritmo);                     //built word to save

      }
```

La siguiente función es, quizá, la pieza clave del cálculo del ritmo cardíaco, el cual dependerá completamente del algoritmo utilizado. La idea que se desarrolla en el código es la que se representa en la figura 29:

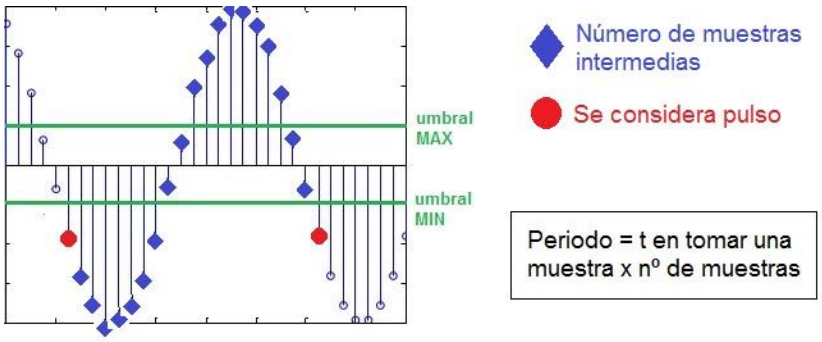

FIGURA 29. CÁLCULO DEL PERÍODO.

De esta manera, una vez que se detecta un pulso se guarda la posición donde ha sido localizado. Para obtener el valor del período de la señal se calculará la media de las distancias que se han encontrado entre pulsos. Posteriormente se multiplicarán el valor medio de muestras entre pulsos por el tiempo que tarda en tomar cada muestra.

III. ECG periodo

```
float proc_data_ECG_10Hz(uint8_t *data){
  int I = 0, u =0, cnt = 0, j=0, c=0;                    // Declarations and
  pulsos = 0, pos = 0, tiempopulsos = 0, o =0;           // Initializations
  int underumbralmin = 1;
```

```
  float media = 0, max = 0, min = 0;
  float umbralmax, umbralmin, periodo;
  float muestrasporpulso, frecuencia;
  float voltios[1000];                                  //value of samples mV
  int posiciones[32];                                   //save pulse position
  int pos1, pos2, pos3;

  while (i<cont2){                                      //Clear bufa to save
       if (bufA[i] == 0xc0 &&(bufA[i+1] == 0            //true information
       || bufA[i+2] == 0))
         i=i+3;
         else{
              buf12[j] = bufA[i];
              j++;
              i++;
              cnt++;
              }
       for (i=0; i<(cont2/6); i++ ){                    //built a vector with
         pos1 = (int)(buf12[j]<<16);                    //three    bufA    bytes
         pos2 = (int)(buf12[j+1]<<8);                   //each position
         pos3 = (int)(buf12[j+2]);                      //save in muestras[]
         muestras[i] = pos1|pos2|pos3;
         j = j+3;
         }
       for(i=0; i<1000; i++){                           //Calculate   value   in
         if((muestras[i]&0x800000)==0)                  //voltios
              voltios[i] = (float)muestras[i]*LSBw*1000;
         if((muestras[i]&0x800000) != 0){
              voltios[i] = (( (float)muestras[i]
              - 16777216) * LSBw)*1000;
              }
       for(i=10; i<700; i++){                           //Calculate   arithmetic
         media = (voltios[i] + media);                  mean, max and min values
         if(max < voltios[i])
              max = voltios[i];
         if(min > voltios[i])
              min = voltios[i];
              }
       media = media/690;                               //Calculate   max   and
       if(media>=0){                                    //min thresholds
         umbralmax = max/5 + media;
         umbralmin = media+(min/5);
         }
       for (I = 0; i<700; i++){                         //when   pass   the   min
         if((voltios[i] <= umbralmin) &&                //threshold   it   will
         !underumbralmin && ((voltios[i-1]<0)           //active
         || (voltios[i+1]<0))){                         //underumbralmin     and
              pulsos++;                                 //then,   wait   to   pass
              underumbralmin = 1;                       //max   threshold.   At
              posiciones[pos] = i;                      //this     moment     we
              pos++;                                    //assume   that   ther   is
              }                                         //a pulse
         if (voltios[i] >= umbralmax)
              underumbralmin = 0;
              }
       for (i=0; i<(pulsos-1); i++)                     //Calculate   mean   of
         tiempopulsos = tiempopulsos +                  //number   of   samples
              (posiciones[i+1] - posiciones[i]);        //between two pulses
       muestrasporpulso =  tiempopulsos /(pulsos-1);
       periodo = 10000*0.006*(muestrasporpulso);        //Calculate period
       frecuencia = 10000/ periodo;                     //and frequency

       return periodo;
}
```

Otra de las funciones principales destaca la que realiza la gestión de la medida de temperatura. Concretamente establece el tiempo que estará tomando medidas, guardará su valor en el vector *buf11* para que posteriormente la función *proc_data_Ta()* se encargue de calcular el valor final, media de todas las medidas realizadas.

IV. Ta Measurement

```
void Ta(void){
  int I;
  ads1298_Ta();                                  //Init    with    Ta
                                                 //sensor active

  ads1298_startcapt();                           //START capture
  cont = 0;
  i=0;                                           //Set to 0 RTC values
  cont22 = 0;
  rtc_reset();                                   //Measure temperature
  tmp = RTC_TPR;                                 //for 1s
  while(RTC_TPR<(tmp+0x3E8)){                    //(7 measurements)
       dataReady_Ta();
       cont++;                                   //Set to 0 RTC values
       }                                         //call to calculation
  rtc_reset();                                   //function
  proc_data_Ta();
  ads1298_stopcapt();                            //STOP capture
  }
```

Para terminar, se comenta el código referente a cómo se calcula el valor final de temperatura que se muestra al usuario.

V. Temperature Value

La función comienza con las típicas inicializaciones y declaraciones. A continuación de ello se construye a partir de los bytes recibidos del ADS1298 (guardados en *ta_vector*) otro vector llamado *pos[]*, que guardará los valores preparados para ser convertidos ya a grados centígrados y cuya longitud es de 7 enteros.

Seguidamente se busca los dos valores mayores, el valor menor y se calcula la media sin estos tres.

Finalmente, para calcular el valor en grados centígrados multiplicamos por el peso en voltios del bit menos significativo, y aplicamos la fórmula facilitada para la temperatura en las hojas de características del ADS1298 (9) y que se muestra en el apartado 2.3 del Capítulo 2 de este texto.

Las últimas dos líneas corresponden a la generación de los informes mencionados en secciones anteriores.

```
Void proc_data_Ta(void){
  int value = 0, p = 0, I;
  int pos[6];
  float r;
  float LSBw = 2.86e-7;
```

```
ta = 0;
pos[0]=(ta_vector[9]<<16)|(ta_vector[10]<<8)|(ta_vector[11]);
pos[1]=(ta_vector[15]<<16)|(ta_vector[16]<<8)|(ta_vector[17]);
pos[2]=(ta_vector[21]<<16)|(ta_vector[22]<<8)| ta_vector[23]);
pos[3]=(ta_vector[27]<<16)|(ta_vector[28]<<8)|(ta_vector[29]);
pos[4]=(ta_vector[33]<<16)|(ta_vector[34]<<8)|(ta_vector[35]);
pos[5]=(ta_vector[39]<<16)|(ta_vector[40]<<8)|(ta_vector[41]);


min = pos[0];
for(i=0; i<6 ; i++)
  if(pos[i]<min) min = pos[i];

max=pos[0];
for(i=0; i<6 ; i++)
  if(pos[i]>max) max = pos[i];

max2 = pos[0];
for(i=0; i<6 ; i++)
  if(pos[i]>max2 && pos[i]!=max) max2 = pos[i];

for(i=0; i<6 ; i++)
  value = value + pos[i];
value = (value-min-max-max2) / 3;

r = (float)value * LSBw;
r = ((r/6)*1000000);
ta = ((((int)r - 145300)/490)+25);

if(ta<3 || ta>100) flash = (1<<29) | (1<<26);
   else flash = (1<<29) | (ta);
}
```

OTROS BLOQUES FUNCIONALES NECESARIOS

De una manera menos detallada que en la sección anterior se relacionan, a continuación, las funciones empleadas para poder dar funcionalidad real al conjunto. En la siguiente tabla 9 se indica la cabecera de las funciones utilizadas en cada bloque. Su estructura interna puede consultarse en la versión digital de la documentación, en el código del programa. Entendemos que se trata de funciones con un cierto carácter general y que disponen de una estructura que, básicamente, es común a todos los sistemas que los utilizan, centrándose sobre todo en la configuración de los registros pertinentes para cada caso, que viene debidamente especificada en las hojas de características de los dispositivos (9) (7).

Bloque	Funciones
Universal Asynchronous Receiver Transmitter (UART)	`void uart0_init(){`

Serial Peripheral Interface (SPI) (10)	`void spi1b_init(void);`
	`uint8_t spi1b_write_read(uint8_t txdata);`
	`void spi1b_set_CS(void);`
	`void spi1b_clear_CS(void);`

	void spi1b_set_RESET(void);
	void spi1b_clear_RESET(void);
	void spi1b_set_START(void);
	void spi1b_clear_START(void);

ADS1298 (11)	void ads1298_init(void){
	void ads1298 id(void);
	void ads1298_STOP(void);
	void ads1298_START(void);
	void ads1298_reset_pin(void);
	void ads1298_RESET(void);
	uint8_t ads1298_WREG(char addr, char value);
	uint8_t ads1298_RREG(char addr);
	void ads1298_readData(uint8_t *data);
	void ads1298_testInput(void);
	void ads1298_RLD(void);
	void ads1298_Ta(void);

Real Time Clock (RTC) (12)	void rtc_config(void);
	void rtc_ms (void);
	void rtc_s (void);
	void rtc_5s(void);

Memoria Flash (13)	void flash_eraseSector(int address);
	void flash_writeLongWord(int address,char *dat);
	void flash_program(int address, char *data, unsigned int length);
	void flash_save(int address, int newdata);
	void flash_show(int address);
	void Ram2Flash(int newdata);

Low Power Timer (LPTMR)	void LPTMR0_IRQHandler(void);
	void lptmr_dummy_call_back(void);
	void wait_vlps_s(uint32_t seg);

Multipurpose Clock Generator (MCG)	void fbi2blpi(void); void clock_FBI_Init(void);

Power Modes	void go2vlps();
	void vlpr2run();
	void run2vlpr();

TABLA 9. FUNCIONES DE OTROS BLOQUES FUNCIONALES NECESARIOS.

CAPÍTULO 4. PRUEBAS Y RESULTADOS

4.1 INTRODUCCIÓN

El objetivo de este capítulo es describir las pruebas realizadas para comprobar el funcionamiento del sistema, su nivel de fiabilidad y limitaciones. Los resultados que se obtengan serán analizados y evaluados con el fin de poder emitir, en capítulos posteriores, las conclusiones del trabajo. Las pruebas realizadas han sido las que se muestran en la tabla 10.

PRUEBAS REALIZADAS			
ECG		**Temperatura**	
Diferentes frecuencias	Diferentes amplitudes	Temperatura ambiente	Calentamiento con flexo
Comprobación de resultados			
Frec VS Frec Generador de funciones		Temperatura VS Temperatura termómetro de ambiente	

TABLA 10. PRUEBAS REALIZADAS.

4.2 PRUEBAS REALIZADAS

A. MEDIDAS DE ECG

Para llevar a cabo estas pruebas se utiliza como elemento auxiliar el generador de funciones PROMAX HM8030. Se configura para obtener de él una salida de tipo sinusoidal, sobre la que se cambiará tanto frecuencia como amplitud. Su salida ha sido conectada al canal 1 del ADS1298 (CH- , CH+), simulando la señal que podría llegar desde el corazón del pez.

Pero, ¿cómo es el ECG de un pez? En primer se debe considerar las diferencias entre peces como la carpa, dorada, lubina (con las cuales trabajaremos) y otros peces de mayor tamaño, para los que, de manera prácticamente rotunda, puede afirmarse que no serán válidas las premisas utilizadas para el cálculo del ritmo cardíaco en este texto. En la

figura 30 se muestra una imagen de una carpa común (*Cyprinus carpio L.*), la cual ronda los 60-80 cm de longitud y un peso medio de 8-9 kg.

FIGURA 30. CARPA COMÚN.

FIGURA 31. PEZ DORADO.

En segundo lugar, hay que señalar que el ECG dependerá totalmente de la derivación que se utilice a la hora de colocar los electrodos, lo que quiere decir que si se colocan alejados del corazón tendremos una forma de señal diferente, para el mismo pez, en el mismo momento, que si lo colocamos más cerca (14). En este caso asumiremos que esta señal será la que se obtenga en las mejores condiciones de amplitud (la cual se corresponde en el ECG humano con la derivación DII). Se trata de una derivación cercana al corazón, y situada en un punto que corta la trayectoria del vector eléctrico de mayor amplitud. Un ejemplo de señal de ECG de una carpa común es la de la figura 32.

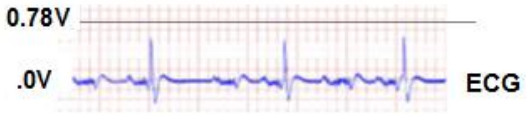

FIGURA 32. ECG CARPA COMÚN.

Un dato interesante que se puede obtener de la figura 32 es la amplitud de la señal, aproximadamente de unos 0.8Vpp. Otro parámetro es la frecuencia que, aunque no se especifique el tiempo, puede obtenerse fácilmente sabiendo que el tamaño de cuadrícula de los ECG está estandarizado y vale exactamente 0.04s. Por tanto, la frecuencia del ECG anterior será igual a 0.6Hz.

Otro ejemplo que puede comentarse es el del pez dorado o *Carassius auratus*, en inglés *goldfish* (figura 31). Su longitud media es de aproximadamente 30cm y su peso puede alcanzar los 2.5Kg en algunos casos. Observando su ECG (figura 33), tenemos que la amplitud es menor que en el caso anterior llegando a los 0.5Vpp. Este hecho puede

deberse a su tamaño que es menor. Por otra parte se puede deducir que que la derivación utilizada para la medida no es la misma que en el caso de la carpa, ya que tiene una forma de onda diferente. Su frecuencia rondará los 0.9Hz.

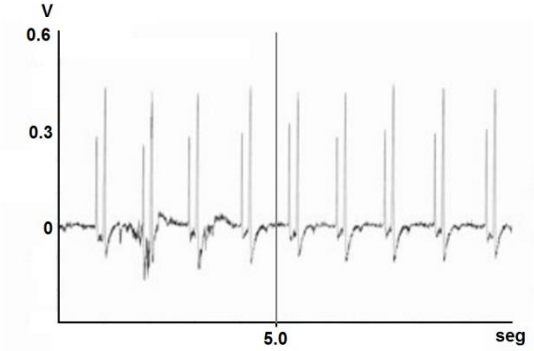

FIGURA 33. ECG PEZ DORADO.

Se ha estimado conveniente probar el generador de funciones para un amplio rango de frecuencias: desde 0.5Hz hasta 15Hz. Teniendo en cuenta que el ancho de un pulso puede ser aproximadamente de 0.1s, y que eso equivale a 10Hz, siguiendo el criterio de Nyquist la frecuencia de muestreo debería ser, al menos de 20Hz (lo cual se corresponde con 0.05s ó 50ms para el tiempo entre muestras). Como ya se mencionó en el capítulo anterior, el algoritmo tomará muestras cada 0.006s, reduciendo la probabilidad de que algún pulso pudiera resultar invisible.

B. MEDIDAS DE TEMPERATURA

Las pruebas se realizan en dos ambientes distintos: a temperatura ambiente y colocando el sensor cerca de un flexo de estudio (Figura 34). En la Figura 35 se muestra la uno de los resultados obtenidos del cálculo de la temperatura, en grados centígrados, junto con el código de su informe de tarea.

FIGURA 34. CALENTAMIENTO DEL SENSOR CON FLEXO.

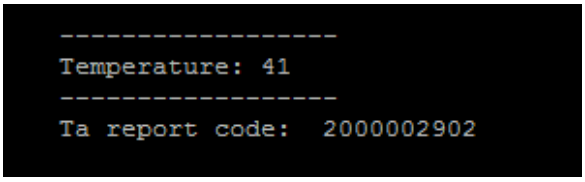

FIGURA 35. RESULTADO DE TEMPERATURA.

4.3 RESULTADOS

A. MEDIDAS DE ECG

Como se comentó, para realizar las pruebas se jugará con dos parámetros, la amplitud y la frecuencia. En la tabla 11 se muestran los resultados para la medida de ritmo cardíaco.

Ritmo	Amplitud (Vpp)						
	Ritmo medio	1,5	1	0,8	0,6	0,4	0,3
30	24,2	19	19	20	39	24	39
60	61,6	67	60	62	65	54	87
90	88,6	97	95	98	87	66	88
180	178,6	182	179	182	176	174	123
300	289,4	293	300	300	296	258	60
420	407,6	423	423	423	423	346	97
540	518,2	537	547	532	532	443	221
600	595,4	598	598	604	595	582	118
900	891,6	882	893	897	893	893	195
Frec.	**Amplitud (Vpp)**						
	Frec media	**1,5**	**1**	**0,8**	**0,6**	**0,4**	**0,3**
0,5	0,44	0,32	0,32	0,33	0,65	0,40	0,65
1	1,10	1,12	1,00	1,03	1,08	0,90	1,45
1,5	1,48	1,62	1,58	1,63	1,45	1,10	1,47
3	2,98	3,03	2,98	3,03	2,93	2,90	2,05
5	4,82	4,88	5,00	5,00	4,93	4,30	1,00
7	6,79	7,05	7,05	7,05	7,05	5,77	1,62
9	8,64	8,95	9,12	8,87	8,87	7,38	3,68
10	9,92	9,97	9,97	10,07	9,92	9,70	1,97
15	14,86	14,70	14,88	14,95	14,88	14,88	3,25

TABLA 11. RESULTADOS PRUEBA ECG.

Puede verse que para la columna de correspondiente a 0.3Vpp de amplitud los datos no son correctos, limitando el rango de aplicación del algoritmo.

Un análisis más exhaustivo del error que se ha cometido en cada medida se muestra en la tabla 12, mediante el cálculo del error relativo.

Ritmo	Amplitude (Vpp)						
	Error medio	1,5	1	0,8	0,6	0,4	0,3
30	19%	37%	37%	33%	-30%	20%	-30%
60	-3%	-12%	0%	-3%	-8%	10%	-45%
90	2%	-8%	-6%	-9%	3%	27%	2%
180	1%	-1%	1%	-1%	2%	3%	32%
300	4%	2%	0%	0%	1%	14%	80%
420	3%	-1%	-1%	-1%	-1%	18%	77%
540	4%	1%	-1%	1%	1%	18%	59%
600	1%	0%	0%	-1%	1%	3%	80%
900	1%	2%	1%	0%	1%	1%	78%

TABLA 12. ECG: ERROR RELATIVO.

El rango que consideramos de trabajo del sistema puede considerarse de amplitud 0.8-0.6Vpp y en un margen de frecuencias de 0.5 a 1.5Hz. En las dos tablas anteriores puede verse que se realizan pruebas para frecuencias mucho mayores, sin embargo, hay que aclarar que no se trata de que esperemos que la señal del ECG del pez pueda ir a 600 p.p.m. (10Hz), sino que es interesante utilizarlas ya que el ancho del pulso sí que puede ser parecido al ancho del pulso de la señal generada por el corazón de un pez (aproximadamente 0.1s).

Respecto al error relativo medio, sin contar con las medidas para amplitudes de 0.3Vpp y menores, es igual al 4%.

B. Medidas de Temperatura

Los valores de temperatura obtenidos se muestran en la tabla 13, donde aparecen 20 medidas diferentes ordenadas por hora de realización y por situación en la que han sido tomadas. Comprobando en la página web de Aemet comprobarse que la temperatura media al aire libre en el intervalo que va desde las 18 a las 24 horas del día 13 de julio de 2016 (día en que se realizaron las pruebas) es de 23°C.

Ta ambiente			Con flexo		
Nº	Hora	Temperatura	Nº	Hora	Temperatura
1	20:59	23	11	21:12	42
2	21:01	25	12	21:13	41
3	21:01	26	13	21:13	31
4	21:01	26	14	21:14	32
5	21:02	24	15	21:14	35
6	21:03	25	16	21:14	38
7	21:04	23	17	21:15	39
8	21:04	25	18	21:15	41
9	21:04	25	19	21:16	38
10	21:05	26	20	21:16	37

TABLA 13. RESULTADO PRUEBA DE TEMPERATURA.

La desviación estándar de estos resultados es igual a 1.07 para la prueba realizada a temperatura ambiente y de 3.56 para la que se realiza calentando el sensor con el flexo. Sobre la calidad o fiabilidad de estos datos, de las hojas de características podemos deducir que mientras la tensión de alimentación del dispositivo se encuentre entre AVDD y AVSS, siendo ésta de -0.3V a 5.5V respectivamente, el funcionamiento será adecuado obteniéndose valores comprendidos entre los 0°C y los 70°C.

4.4 ESTUDIO DE POTENCIA

El sistema está implementado de manera que en función del instante de ejecución en el que se encuentre entrará en un modo de consumo de potencia u otro, ya que es uno de los propósitos limitar el consumo, un aspecto clave en este trabajo.

El modo normal de operación es el modo RUN, activado siempre después de un reset. Para reducir el consumo este modo parte con los relojes de los módulos que no se utilizan desactivados (por medio de los registros SIM).

Cuando entra en modo VLPR, el regulador de tensión se pone en estado de regulación en modo stop, y se encargará de alimentar al MCU lo suficiente para que trabaje a una frecuencia reducida. Al igual que en el modo RUN, desactiva los relojes a los módulos que no se utilicen. Para poder entrar en el modo VLPR es necesario que se cumplan las siguientes condiciones:

- El MCG deberá configurarse en un modo soportado en el modo VLPR.
- Todos los supervisores de reloj en modo MCG deben deshabilitarse.
- Las frecuencias máximas están limitadas a 4MHz la del sistema, y a 0.8MHz las del bus y la flash.
- Debe configurarse el modo de protección para permitir los modos de bajo consumo VLP (PMPROT[AVLP] = 1).
- PMCTRL[RUNM] = 10b para entrar en VLPR.
- No se permite borrar ni programar la flash.

Por último, el siguiente modo que se utiliza en el código es el VLPS al que se puede entrar de dos maneras:

- Sleep now o Sleep-on-reset: vía activación del bit SLEEPDEEP en el Registro de Control de Sistema en el núcleo del ARM mientras el MCU se encuentra en modo VLPR y además STOPM=010 ó 000 en el registro PMCTRL.
- Entrar de la misma manera pero cuando el MCU está en modo RUN y STOPM=010 en el registro PMCTRL.

En modo VLPS, el regulador de tensión permanece en su estado de regulación stop como en el modo VLPR. Existe un módulo capaz de generar una interrupción que provoca que el dispositivo salga del VLPS y vuelva al VLPR. Un reset del sistema siempre supondrá la salida del modo VLPS, devolviendo al dispositivo al modo RUN normal.

Un Power-on-Reset establece el modo de reloj FEI por omisión, con una frecuencia de sistema de 21MHZ y de bus y flash de 10.5MHz. En este caso, el sistema parte de este modo de operación cuando se inicia el dispositivo.

Para realizar el análisis de potencia comenzamos por identificar los modos que se utilizan en los distintos momentos de la ejecución del experimento, el consumo de cada uno de ellos y, posteriormente, estudiaremos el tiempo que se invierte en cada modo. Del manual de referencia del KL25Z (15) puede sacarse la información que aparece en la tabla 14.

Símbolo	Descripción	Típica	Máx	Unidad
IDD_RUN	Modo: Run Núcleo: 48 MHz Bus/Flash: 24 MHz Relojes de periféricos: deshabilitados. Código ejecución: flash a 3V a 3.0 V	5	5.9	mA
IDD_VLPR	Modo: Very Low Power Run Núcleo: 4 MHz Bus/Flash: 0.8 MHz Relojes de periféricos: habilitados. Código ejecución: flash, a 3.0 V	262	509	µA
IDD_VLPS	Modo: Very Low Power Stop, a 3.0V Temperatura: 25 °C	3.75	8.46	µA

TABLA 14. CONSUMO DE LOS MODOS DE OPERACIÓN KL25Z.

Es necesario conocer, además, cuánto tiempo invierte el MCU en cada modo y para cada experimento. El interés de estos cálculos radica en los experimentos automáticos de larga duración que, en el caso de este código, es de un mes. Se ha utilizado el módulo RTC que estaba configurado previamente para conocer los tiempos de ejecución. Así, en la tabla 15, se muestra el planteamiento de los cálculos para obtener el resultado final de consumo.

		Ejecución experimento			Iteraciones/mes	Energía (J) en un mes	mA-h
		Cálcular	Guardar	Dormir			
Modo de potencia		VLPR	RUN	VLPS			
Corriente (mA)		0,262	5	0,00375			
Potencia (W)		0,786	15	0,01125			
Tiempo invertido (s)	EXP1	40,345	0,032	10800	240	36885,88	3,42
	EXP2	40,345	0,032	21600	120	33022,94	3,06
	EXP3	40,345	0,032	43200	60	31091,47	2,88
	EXP4	40,345	0,032	86400	30	30125,74	2,79

TABLA 15. CÁLCULO DEL CONSUMO DE POTENCIA.

CAPÍTULO 5. CONCLUSIONES

5.1 CONCLUSIONES

El trabajo realizado se centra en un entorno importante para la sociedad, como es la acuicultura. Es una importante actividad económica de producción de alimentos, que permite obtener materias primas de uso industrial y farmacéutico, e incluso en ocasiones los organismos vivos se emplean para repoblación y ornamentación.

La principal función del sistema desarrollado consiste en medir el ritmo cardíaco de un pez. El algoritmo que realiza esta función se basa en métodos matemáticos como la media y la desviación estándar para calcular el valor final, y para el cálculo se establecen umbrales dependientes de la amplitud de entrada en cada medida con lo que se consigue mayor flexibilidad y robustez.

Por otra parte, al tratarse de un sistema inicial en el conjunto del proyecto mayor, realiza tareas basándose en las órdenes que envía un usuario al MCU de forma directa e inmediata. Otras opciones que ofrece son la de identificación del dispositivo, valor de temperatura corporal, ejecución de experimentos autónomos y de larga duración y almacenamiento de informes de monitorización para cada medida.

En cuanto al análisis de potencia el sistema se calcula que consumirá en un mes que dura cada experimento una energía igual a 36.89kJ en un mes para el experimento programado para ejecutarse cada 3 horas, 33.02kJ para el que se realiza cada 6 horas, 31.09kJ para el que se realiza cada 12 horas y finalmente 30.13kJ para el que se realiza una vez al día; cada 24 horas. Se trata de valores buenos que entran dentro del rango que se estima coherente y deseable para este tipo de sistemas que deberán pasar bastante tiempo sin recibir energía de ningún tipo. En el peor de los casos, que es en el experimento 1, los valores que se manejan son del orden de los 3mAh. Con estos valores tan reducidos se deja abierta la posibilidad de estudiar qué forma de alimentación

emplearse para el sistema; si una simple pila de botón o, si quizá es posible realizarlo mediante RFID.

Sobre de los resultados obtenidos para el ECG puede decirse que su funcionamiento es fiable en un 98% para señales que se encuentran entre los 1.5V y los 0.4V y para frecuencias comprendidas entre 1Hz y 15Hz, y del aumentado el error al 4% si incluimos cálculos para la frecuencia de 0.5Hz. A continuación se muestra una gráfica comparativa de estos resultados, donde aparecen todas las frecuencias distintas para las que se han realizado medidas y que se expusieron en el capítulo 4.

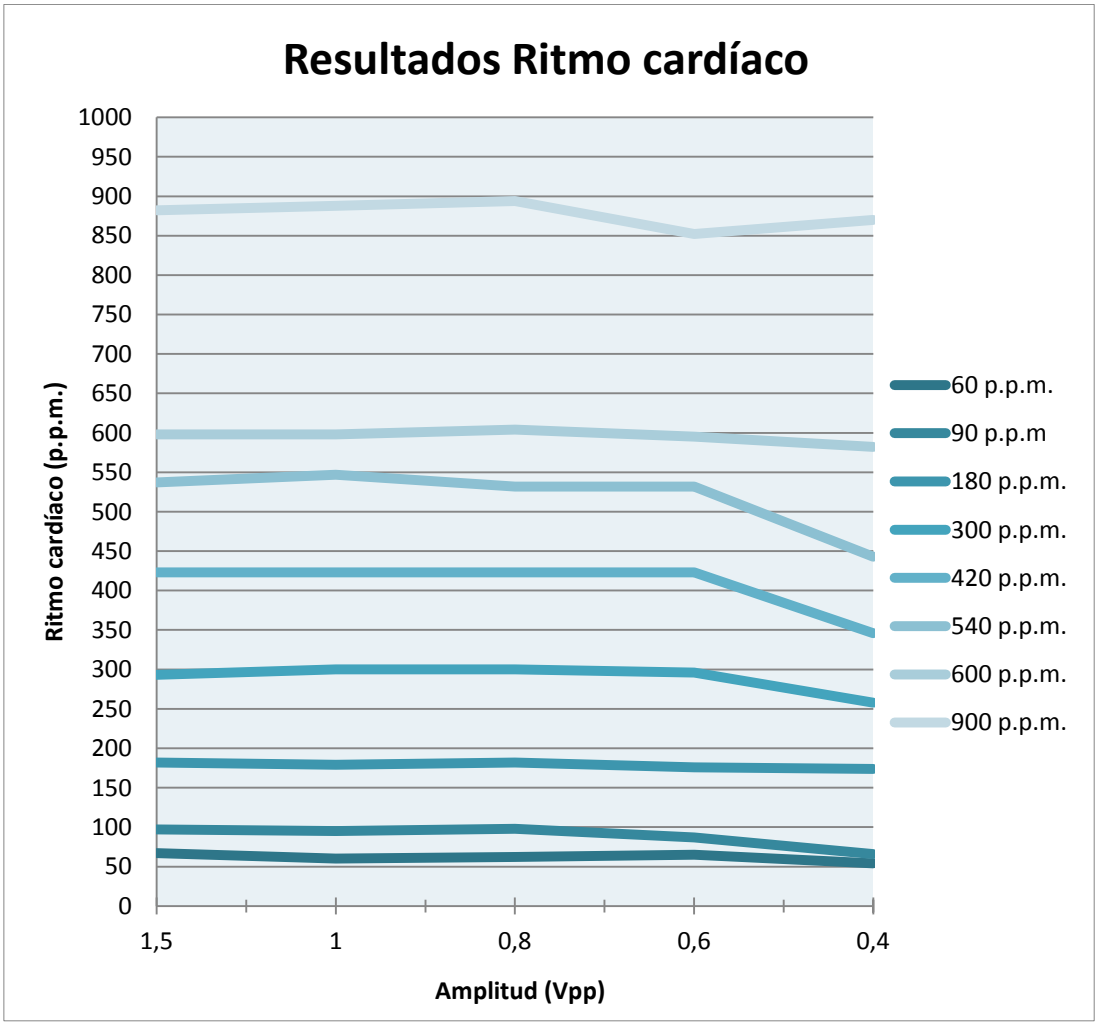

GRÁFICO 2. RESULTADOS DE LA MEDIDA DEL RITMO CARDÍACO.

Sobre esta gráfica podemos concluir, por tanto, que el rango de operación óptimo del sistema, en cuanto a amplitud, se encuentra entre los 1.5Vpp-0.6Vpp, y en cuanto a

frecuencia desde 1Hz-10Hz. Trabajando en ese margen los valores esperados serán correctos.

5.2 LÍNEAS FUTURAS

Existe una gran cantidad de ideas que podrían plantearse como continuación del trabajo, o como mejora. Desde métodos que mejoren el funcionamiento del algoritmo que se encarga de los cálculos hasta el procesado de la señal mediante filtros (teniendo siempre presente mantener un buen compromiso con el bajo consumo de energía que se requiere), o el ajuste necesario para que sea de aplicación en peces de mayor tamaño. Por otra parte, conociendo la morfología y las características de la evolución de la señal la señal se podría diseñar una solución específica que eventualmente mejore el error cometido en los resultados.

Una importante línea futura que se plantea es la de realizar una interfaz con herramientas como LabView que ofrezca una monitorización gráfica de las medidas.

El dispositivo escogido como sensor, el ADS1298, ofrece otras opciones interesantes para el fin que aquí se plantea por ejemplo realizar medidas de encefalograma (EEG). Éstas se basan en el registro de la actividad bioeléctrica cerebral en condiciones basales de reposo, en vigilia o sueño, y durante diversas activaciones. Esta posibilidad podría resultar digna de ser considerada desde el punto de vista de profundizar más en las condiciones en las que se desarrolla la actividad de los peces en las piscifactorías.

Sobre el MCU podrían plantearse nuevos retos como, por ejemplo, la disminución del consumo utilizando otros modos de potencia que ofrece más sofisticados, como LLS (Low Leakage Stop), VLLS3 (Very Low Leakage Stop3), VLLS1 (Very Low Leakage Stop1) y el modo VLLS0 (Very Low Leakage Stop 0), y que, por las restricciones de tiempo de los TFM no se han considerado en este trabajo. Con ello se conseguiría reducir aún más el consumo del sistema y mejorar su autonomía.

CAPÍTULO 6. BIBLIOGRAFÍA

1. **Animaturalis.** Animaturalis Internacional. [En línea] [Citado el: 15 de julio de 2016.] http://www.animanaturalis.org/p/1051/granjas_pesqueras_o_piscifactori_as_fabricas_bajo_el_agua.

2. *Evaluation of electrical stunning of sea bass (Dicentrarchus labrax) in seawater and killing by chilling: welfare aspects, product quality and possibilities for implementation.* **Bert Lambooij.** The Netherlands : Blackwell Publishing Ltd, 2008.

3. *Electrical and percussive stunning of the common carp (Cyprinus carpio L.): Neurological and behavioural assessment.* **Lambooij, E.** The Netherlands : Aquacultural Engineering, 2007.

4. **Martínez, Harkaitz Eguiraum.** *Toward intelligent aquaculture.* Plentzia : Universidad del País Vasco, 2015.

5. **Hurtado, Nicolás.** SlideShare. [En línea] 2016. [Citado el: 19 de julio de 2016.] http://es.slideshare.net/nhurtado2000/estado-actual-y-perspectivas-de-la-acuicultura-al-2030.

6. **Freescale.** *FRDM-KL25Z User's Manual.* s.l. : Freescale Semiconductor, Inc., 2013.

7. **NXP.** KL25 Sub-Family Reference Manual. *Datasheet.* s.l. : Freescale Semiconductor, Inc., September de 2012. KL25P80M48SF0RM.

8. **ARM.** *Cortex M0+ Technical Reference Manual.* 2012.

9. **Texas Instruments.** ADS1294/96/98. *Low-Power, 8-Channel, 24-Bit Analog Front-End for Biopotential Measurements.* 2010.

10. GitHub. *FRDM-SPI.* [En línea] 30 de septiempbre de 2013. [Citado el: 10 de mayo de 2016.] https://github.com/burberger/FRDM-SPI.

11. **Charbiwala, Zainul.** ECFAFE_copy. [En línea] 1 de octurbre de 2015. [Citado el: 25 de junio de 2016.] https://developer.mbed.org/users/zainulcharbiwala/code/ECGAFE_copy/file/ee0649a9025a/ADS1298.cpp.

12. NXP. *KL25 - RTC and system oscillator.* [En línea] 6 de junio de 2013. [Citado el: 22 de mayo de 2016.] https://community.nxp.com/thread/308261.

13. **Olieman, Erik.** ARMmbed. *FreescaleIAP.* [En línea] 10 de mayo de 2014. [Citado el: 6 de junio de 2016.] https://developer.mbed.org/users/Sissors/code/FreescaleIAP/file/ab8a833a25eb/FreescaleIAP.cpp.

14. *Cardiac conduction times in Sparus auratus at different heart rates. Influence of body weight.* **Aissaoui, A.** March de 2005, Journal of fish biology.

15. **Freescale.** *Datasheet: Technical Data Kinetis KL25 Sub-Family.* Rev. 5 08/2014. KL25P80M48SF0.

16. **Toulson, Rob.** *Fast and effective embedded systems design - Applying the ARM mbed.* Oxford : Elsevier Ltd, 2012. pág. fag.

17. **Gómez, Óscar.** *Sistema de Seguridad en vehículos (SiSVe).* Madrid : Universidad Politécnica de Madrid, 2013.

DOCUMENTO Nº2 PLIEGO DE CONDICIONES

PLIEGO DE CONDICIONES

ÍNDICE DETALLADO

PLIEGO DE CONDICIONES

En esta sección se especifican los requisitos necesarios hardware y software para llevar a cabo el trabajo.

1. ESPECIFICACIONES HARDWARE

El hardware mínimo para realizar el trabajo es el que se muestra en la tabla 1.

Material	Características	Descripción
Ordenador personal	**Procesador**	Tipo : Procesador Intel® Pentium® T4200 Velocidad de reloj : 2.00 GHz Front Side Bus : 800 MHz caché de 2º nivel : 1 MB
	Memoria principal	Estándar : 4,096 (2,048 + 2,048) MB Máxima expansión : 8,192 MB Tecnología : DDR2 RAM (800 MHz)
	Disco duro	Capacidad formateado : 250 GB Velocidad de rotación : 5,400 rpm
	Adaptador Gráfico	Tipo : ATI Mobility Radeon™ HD 4570 con tecnología HyperMemory™ Memoria : 512 MB dedicados VRAM (memoria total disponible para gráficos usando la tecnología HyperMemory™ hasta 1.791 MB con un sistema operativo de 32-bit y 3 GB de RAM o 3.323 MB con un sistema operativo de 64-bit y 6 GB de RAM) Tipo de memoria : Video RAM DDR3 Conexión del bus : PCI Express®
	Interfaces	1 × Ranura multitarjetas 4-en-1 (soporte Tarjetas SD™ hasta 16 GB, Memory Stick® hasta 256 MB, Memory Stick Pro™ hasta 2 GB y MultiMedia Card™ hasta 2 GB) 2 × USB 2.0
	Comunic. sin cables	Compatibilidad : Wi-Fi® Soporte de red : 802.11b/g/n Wireless Technology : Wireless LAN

	Comunic. con cables	Tipología : Fast Ethernet LAN Velocidad : 10BASE-T/100BASE-TX
Plataforma de desarrollo	**FRDM-KL25Z**	
Kit de demostración	**ADS1298ECG-FE EVM PDK**	
Cableado	**Sin especificar**	
Ratón	**Sin especificar**	
Impresora comercial	**Sin especificar**	
Multímetro	**Sin especificar**	
Generador de funciones	**Sin especificar**	

TABLA 1. ESPECIFICACIONES HARDWARE.

2. ESPECIFICACIONES SOFTWARE

El software mínimo para realizar el trabajo es el que se muestra en la tabla 2.

Material	Características	Descripción
Sistema operativo	Sistema Operativo Windows 7 Home Premium	
Keil uVision	Version 5	
PuTTy		
Software de oficina	Microsoft Office 2007	Word:redacción del TFM. PowerPoint: presentación TFM. Excel: tratamiento de datos.
Paint	Versión de W7	
Recortes	Versión de W7	

TABLA 2. ESPECIFICACIONES SOFTWARE.

DOCUMENTO Nº3 PRESUPUESTO

PRESUPUESTO

ÍNDICE DETALLADO

PRESUPUESTO

Mª Luisa Barragán Pulido, autora del presente Trabajo Fin de Grado, declara que:

El Trabajo Fin de Máster con título: "Diseño de un sistema empotrado para medida y transmisión de propiedades electrofisiológicas", desarrollado en la Escuela de Ingeniería de Telecomunicaciones y Electrónica de la Universidad de Las Palmas de Gran Canaria, tiene un coste de desarrollo total de 16.276,34 € correspondiente a la suma de las cantidades consignadas a los apartados considerados a continuación.

Fdo.: Mª Luisa Barragán Pulido.

Las Palmas de Gran Canaria, a 19 de julio de 2016.

1. DESGLOSE

Este presupuesto ha sido realizado siguiendo las recomendaciones del Colegio Oficial de Ingenieros de Telecomunicación (COIT) sobre baremos orientativos mínimos para trabajos profesionales en 2014. Los distintos costes asociados al Trabajo Fin de Máster, cuya duración ha sido fijada en 4 meses, se han desglosado en varias secciones como sigue:

- Recursos materiales.
- Trabajo tarifado por tiempo empleado.
- Costes de redacción del TFM.
- Material fungible.
- Derechos del visado del COIT.
- Gastos de tramitación y envío.
- Aplicación de impuestos.

2. RECURSOS MATERIALES

Entre los recursos materiales utilizados se incluyen las herramientas software para el desarrollo del estudio, los paquetes utilizados para la redacción de la memoria y el sistema operativo. Asimismo, se incluyen los equipos hardware utilizados para dar soporte a estas herramientas.

Se estipula el coste de amortización para un período de tres años, empleando un sistema de amortización lineal, en el que se supone que el inmovilizado material se deprecia constantemente a lo largo de su vida útil. La cuota de amortización lineal viene dada por la siguiente expresión:

$$Cuota\ anual = \frac{Valor\ de\ adquisición - Valor\ residual}{Número\ de\ años\ de\ vida\ útil}$$

Donde:

Valor residual = valor teórico que se supone que tendrá el elemento después de su vida útil.

A. RECURSOS HARDWARE

Las herramientas hardware que se han utilizado han sido:

LISTA HARDWARE			
Nombre	**Descipción**	**Carácterísticas**	
Toshiba Satellite L500	Ordenador	Procesador	Intel Pentium Dual Core T4200
		RAM	4GB
		Disco duro	250 GB Serial ATA. (5.400 rpm)
		Tarjeta gráfica	ATI HD4570 512MB dedicados
FRDM-KL25Z	Plataforma de desarrollo	MCU	KL25Z128VLK4 Cortex-M0+ MCU
ADS1298ECG-FE PDK	Kit de evaluación	Sensor	ADS1298
HAMEG HM8030-6	Generador de funciones		
PROMAX PD-690	Multímetro		
Cableado		Cables con espadines en los extremos	

COSTES HARDWARE			
Nombre	**Coste total (€)**	**Valor residual a 3 años (€)**	**Valor de amortización a 4 meses (€)**
Toshiba Satellite L500	599,00	200,00	44,33
FRDM-KL25Z	13,58	4,53	1,01
ADS1298ECG-FE PDK	189,60	63,20	14,04
HAMEG HM8030-6	518,00	172,67	38,37
PROMAX PD-690	76,22	25,41	5,65
Cableado	10	3,33	0,74
TOTAL DE COSTES (€)			**104,14**

TABLA 1. MATERIAL HARDWARE.

B. RECURSOS SOFTWARE

Las herramientas software utilizadas fueron:

LISTA SOFTWARE		
Nombre	**Descipción**	**Carácterísticas**
Windows® 7	Sistema Operativo	
uVision v.5 de Keil	Paquete MDK	Herramientas de ARM
Microsoft Office® 2007	Paquete de oficina	
PuTTY v.067.1010.0	Emulador de terminal	

COSTES SOFTWARE			
Nombre	**Coste total (€)**	**Valor residual a 3 años (€)**	**Valor de amortización a 4 meses (€)**
Windows® 7	200	0	22,22
uVision v.5 de Keil	4.947,60	0	549,73
Microsoft Office® 2007	250	0	27,77
PuTTY v.067.1010.0	0	0	0
TOTAL DE COSTES (€)			**599,72**

TABLA 2. COSTES DE RECURSOS SOFTWARE.

3. TRABAJO TARIFADO POR TIEMPO EMPLEADO

Se siguen las recomendaciones del COIT para calcular el importe de las horas de trabajo empleadas. Fijando la duración de las tareas de especificación, desarrollo y documentación del TFM en 300 horas, se aplica:

$$H = C_t \cdot 74{,}88 \cdot H_n + C_t \cdot 96{,}72 \cdot H_e$$

Donde:

H: honorarios totales por el tiempo dedicado.

H_n: horas trabajadas dentro de la jornada laboral.

H_e: horas especiales.

C_t: factor de corrección, función del número de horas trabajadas.

Todas las horas invertidas se consideran dentro del horario laboral normal.

Según el COIT el factor C_t tiene un valor variable en función del número de horas empleadas dado por la Tabla 3.

Horas empleadas	Factor de corrección (Ct)
Hasta 36 h 1,00	1,00
36 a 72 h 0,90	0,90
72 a 108 h 0,80	0,80
108 a 144 h 0,70	0,70
144 a 180 h 0,65	0,65
180 a 360 h 0,60	0,60
360 a 540 h	0,55

TABLA 3. VALOR DEL FACTOR DE CORRECCIÓN.

Para este TFM, el valor del factor de corrección es C_t =0,60 y, por tanto, la expresión anterior queda:

$$H = 0{,}6 \cdot 74{,}88 \cdot 300 = 13.478{,}40\,€$$

Los honorarios totales por tiempo dedicado libres de impuestos ascienden a *trece mil cuatrocientos setenta y ocho euros con cuarenta céntimos* **(13.478,40€)**.

4. COSTES DE REDACCIÓN DEL TFM

El importe de redacción se calcula siguiendo la expresión:

$$R = 0{,}07 \cdot P \cdot C_n$$

Donde:

P: presupuesto.

C_n: coeficiente de ponderación en función del presupuesto.

En la Tabla 4 se muestra el presupuesto calculado hasta el momento.

Descripción	Costes (€)
Recursos hardware	104,14
Recursos software	**599,72**
Trabajo tarifado por tiempo empleado	**13.478,40**
Total de costes (€)	**14.182,26**

TABLA 4. COSTES DE EJECUCIÓN MATERIAL.

Como el coeficiente de ponderación para presupuestos menores de 30.050,00 € viene definido por el COIT con un valor de 1, el coste derivado de la redacción del TFM es de:

$$R = 0{,}07 \cdot 14.182{,}26 \cdot 1{,}00 = 992{,}758\ €$$

Por tanto, el coste libre de impuestos derivado de la redacción del TFM es de *novecientos noventa y dos euros con sesenta y seis céntimos* **(992,76 €).**

5. MATERIAL FUNGIBLE

En la Tabla 5, se relaciona el material fungible empleado en la realización del TFM.

Descripción	Costes (€)
Folios	10,00
Encuadernación	2,50
Impresión	18,00
Total de costes (€)	**30,50**

TABLA 5. COSTES DEL MATERIAL FUNGIBLE.

6. DERECHOS DEL VISADO DEL COIT

Todos los cálculos son orientativos y se ha añadido, también, los derechos de un supuesto visado por el COIT. Los gastos de visado del COIT se tarifan mediante la siguiente expresión:

$$V = 0{,}006 \cdot P \cdot C_V$$

Donde:

P: presupuesto.

C_V: coeficiente reductor en función del presupuesto del trabajo.

El presupuesto P calculado hasta el momento es la suma de la ejecución material, de redacción y de material fungible:

$$P = 14.182{,}26 + 992{,}76 + 30{,}50 = 15.205{,}52\ €$$

Como el coeficiente de ponderación para presupuestos menores de 30.050,00 € es igual a 1,00, según el COIT, el coste de los derechos de visado del trabajo asciende a:

$$V = 0{,}006 \cdot 15.205{,}52 \cdot 1{,}00 = 91{,}23\ €$$

El coste de los derechos de visado es asciende a *noventa y un euros con veintitrés céntimos* **(91,23€)**.

7. GASTOS DE TRAMITACIÓN Y ENVÍO

Los gastos de tramitación y envío están fijados en 6,01 €.

8. APLICACIÓN DE IMPUESTOS

El coste total de la realización del TFM, antes de aplicarle los correspondientes impuestos, es de 15.502,52 €, a lo que hay que sumarle el 7% de I.G.I.C. quedando el coste definitivo como se indica en la Tabla 6.

Costes totales del TFM		
Descripción	**Coste parcial (€)**	**Total (€)**
Recursos materiales		703,86
Hardware	104,14	
Software	599,72	
Coste de ingeniería		13.478,40
Coste de redacción		992,76
Material fungible		30,50
Derechos de visado		
Tramitación y envío		6,01
Subtotal		**15.211,53**
Aplicación de impuestos (7% I.G.I.C.)		1.064,80
Total costes		**16.276,34**

TABLA 6. COSTES TOTALES DEL TFM.

El presupuesto total asciende a la cantidad de *dieciséis mil doscientos setenta y seis euros con treinta y cuatro céntimos* **(16.276,34 €)**.

Fdo.: Mª Luisa Barragán Pulido.

Las Palmas de Gran Canaria, a 19 de julio de 2016.