Escuela Técnica Superior de Ingenieros de Telecomunicación

Universitat Politècnica de València

Proyecto Fin de Carrera

Diseño de un sistema embebido portátil de almacenamiento de registros electrofisiológicos intracerebrales en murino

Autor: **Borja Tarazona Raga**

Directores: **Dr. David Moratal Pérez**
Centro de Biomateriales e Ingeniería Tisular
Universitat Politècnica de València

Dr. Santiago Canals Gamoneda
Instituto de Neurociencias
Consejo Superior Investigaciones Científicas
Universidad Miguel Hernández

The brain is a wonderful organ; it starts working the moment you get up in the morning and does not stop until you get into the office.

Robert Frost

Agradecimientos

En estas líneas quería dedicarles unas palabras de agradecimiento a todas las personas que me han apoyado en estos años de carrera, en especial a mi familia más cercana. A mi madre que me lo ha dado todo sin que yo pidiera nada y a mi padre que, a pesar de su enfermedad, siguió adelante para que no nos faltara de nada. A mi novia, Marina, que me ha acompañado en estos últimos años tan difíciles, muchas gracias por aguantarme.

No podía acabar este trabajo sin agradecer a mi tutor, el *Dr. David Moratal Pérez*, la oportunidad de participar en este proyecto. No sólo he aprendido mucho, sino que me llevo una nueva amistad y unos muy buenos recuerdos. También quería agradecer a *Aarón Cuevas* su paciencia y sus consejos, fueron de gran ayuda para la finalización de este *PFC*.

Las personas no somos más que pedacitos de otras y yo, después de estos años, me llevo muchos vuestros. Muchas gracias a todos por vuestros pedacitos.

Índice

0. - Motivación y objetivos

Until the past few decades, neuroscientists had one way to plumb the human brain: wait for disaster to strike people and, if the victims pulled through, see how their minds worked differently afterward.

Sam Kean, The tale of the dueling neurosurgeons

El propósito principal de las Neurociencias es entender cómo el encéfalo produce la marcada individualidad de la acción humana. Es aportar explicaciones de la conducta en términos de actividades de éste, explicar cómo actúan millones de células nerviosas individuales en el encéfalo para producir la conducta y cómo, a su vez, estas células están influidas por el medio ambiente, incluyendo la conducta de otros individuos.

Los actuales estudiosos del cerebro, saben que para comprenderlo hay que derrumbar las barreras de las disciplinas tradicionales. Esta tendencia queda muy evidente en las obras científicas recientes las cuales tratan las funciones más complejas de este órgano, como las emociones y la consciencia, apoyándose en los principales conceptos provenientes de las diversas disciplinas.

Algunas de las esperanzas alimentadas por este avance del conocimiento tienen que ver con el aumento de nuestra comprensión de las funciones normales así como también de las disfunciones psicológicas y consecuentemente, de que surjan métodos más eficaces de tratamiento de las enfermedades mentales que en conjunto eleven la calidad de vida del ser humano.

Muchos de los experimentos científicos en los que se basa la neurociencia, están cimentados en el uso de animales de laboratorio. Estos animales normalmente se encuentran en condiciones poco naturales en comparación con sus congéneres que viven en libertad. Se trata de animales que se encuentran en condiciones de aislamiento social, son sedentarios, presentan obesidad y son propensos a muertes prematuras. Esto contribuye a la confusión y obtención de errores en los estudios. Por otra parte, los procesos neurológicos tales como aprendizaje, memoria y reconocimiento

son de naturaleza lenta y requieren de largos registros con el fin de realizar estudios estadísticos fiables que permitan obtener conclusiones certeras.

Con esta idea en mente, se pretende diseñar un entorno en el que se disponga de animales de laboratorio en libertad (dentro de un entorno controlado) y en los que se pueda registrar en todo momento su actividad electrofisiológica cerebral así como su posición y orientación dentro del entorno controlado. Estos animales deben poder realizar una vida lo más "real" posible y con la menor interacción por parte de los científicos, por lo que han de disponer de dispositivos inalámbricos que registren la actividad cerebral y la transmitan a una estación base para su registro y almacenamiento.

0.1. - *T-Rat Lab*

Partiendo de la premisa anterior nace el *T-Rat Lab*. El laboratorio que se propone está pensado para dotar al investigador de una serie de herramientas que permitan observar y recopilar información que no sería posible obtener con los métodos tradicionales.

Figura [1]. **Esquema 3D del T-RAT Lab**

Una serie de sensores, cámaras, micrófonos y sistemas de medida se encargarán de registrar todos los parámetros fisiológicos de interés para el científico. Se podrán diseñar un sinfín de experimentos científicos aprovechando las ventajas de estudiar animales en un entorno artificial en condiciones de semi-libertad con la tecnología e instrumentos de medida que podemos encontrar en un laboratorio convencional.

El sistema completo estará compuesto por el implante, el sistema de localización y seguimiento en tiempo real y el sistema de alimentación por inducción. Todos estos sistemas, han sido pensados de forma que puedan ser integrados en un entorno de las características mencionadas. [1]

El sistema de seguimiento se realizará mediante unas etiquetas *RFID* y un suelo de baldosas hexagonales que mediante triangulación y resonancia detectará la posición de la rata. Actualmente esto se encuentra en fase de prueba.

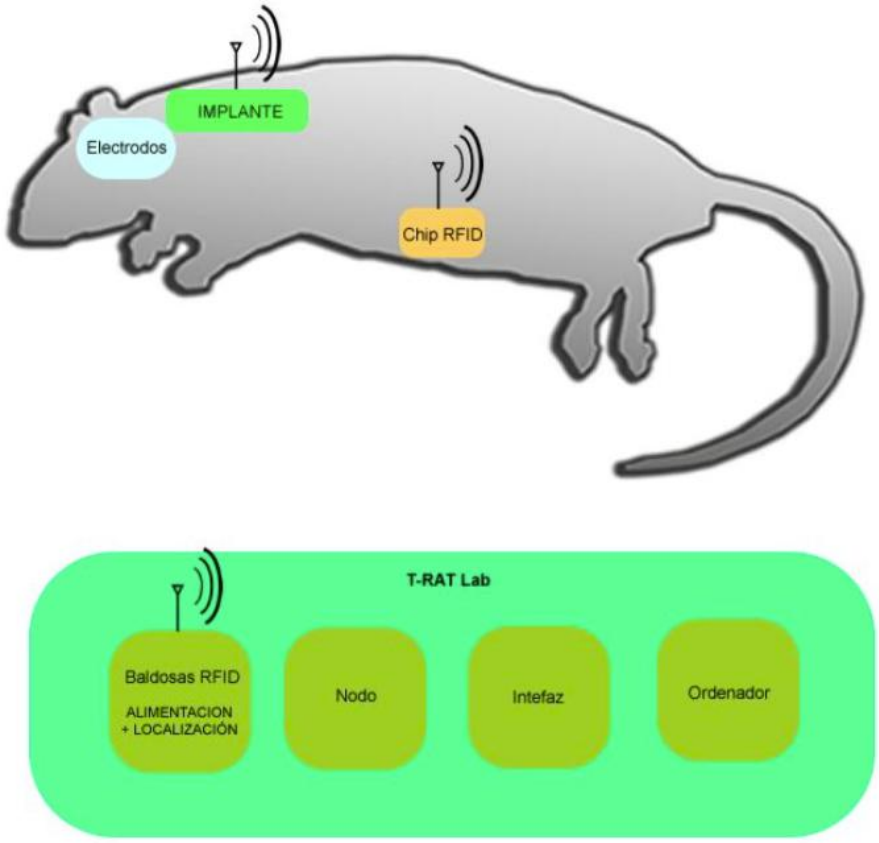


Figura [2]. Esquema del sistema completo de bloques utilizado

A parte, se instalarán cámaras que permitirán realizar un seguimiento adicional de los animales. Esto permitirá ver con mayor detalle comportamientos y asociarlos a los datos obtenidos por el implante y el sistema de seguimiento en tiempo real. Con el fin de grabar las vocalizaciones de los animales se instalarán una serie de micrófonos y sensores de ultrasonidos que captarán esta información.

Todos los elementos en conjunto, formarán un completo laboratorio que dotará al investigador de una versatilidad enorme a la hora de realizar sus experimentos.

0.2. - Objetivo

El laboratorio propuesto por el *T-RAT Lab* es muy ambicioso y complejo. Pretende, además de tener un seguimiento total de los animales a estudiar, transmitir las señales cerebrales en tiempo real. Es aquí, en medio de esta ambición, donde nace este proyecto final de carrera.

El objetivo de este proyecto es realizar el diseño de un sistema de almacenamiento portátil de registros electrofisiológicos intracerebrales. Con esto, se pretende realizar un sistema fiable que permita dotar al animal de total libertad sin tener que entrar en la problemática de la transmisión de señales en tiempo real.

El diseño final de este proyecto no tiene porqué ser sólo útil como paso intermedio hasta poder tener un sistema fiable que permita transmitir la transmisión de datos en tiempo real, sino como complemento que podrá utilizarse conjuntamente con ésta. Esto es debido a que en el laboratorio se situarán una serie de madrigueras en las cuales es posible que la señal *bluetooth* pueda perderse y de esta forma luego se podría observar las partes perdidas.

0.2.1. - Requerimientos del sistema a diseñar

Con este propósito se deberá diseñar un dispositivo con las siguientes características:

- Portátil
- Alta frecuencia de muestreo (en torno a 20*kHz*)
- Capaz de almacenar grandes cantidades de información (del orden de *GB*)
- Con una autonomía suficiente para poder realizar experimentos

La portabilidad en este proyecto es el principal atractivo, pues queremos recrear las condiciones reales de libertad. La frecuencia de muestreo vendrá acordada con el *Instituto de Neurociencias de Alicante*, ésta estará en torno a los 20-25*kHz*. Su capacidad de almacenaje se calculará más adelante en la memoria, pero en un primer momento establecemos el objetivo en varios *Gigabytes*, en torno a los 8 *GB*. El último parámetro a tener en cuenta es la autonomía, sabiendo que un experimento medio puede durar alrededor de dos horas, buscaremos alcanzar ese objetivo.

Hay un último punto que debemos marcar como objetivo en este diseño: el número de canales a almacenar. En el sistema implementado actualmente se adquieren 32 canales, pero para un primer diseño se ha decidido adquirir sólo cuatro canales. Estos canales serán los más relevantes para los estudios que se están realizando actualmente.

1. - Introducción

The human brain has 100 billion neurons, each neuron connected to 10 thousand other neurons. Sitting on your shoulders is the most complicated object in the known universe.

Michio Kaku

La neurociencia es el estudio científico del sistema nervioso. Tradicionalmente, ésta ha sido vista como una rama de la biología, aunque, actualmente se considera como una ciencia interdisciplinar que implica a otras disciplinas tales como la psicología, la computación científica, las matemáticas, la física, la filosofía y la medicina. Como consecuencia, el ámbito de la neurociencia se ha ampliado para incluir diferentes enfoques usados para estudiar los aspectos moleculares, de desarrollo, estructurales, funcionales, evolutivos, computacionales y médicos del sistema nervioso. [1]

La neurología, la psiquiatría y la neuropatología son las especialidades médicas que tratan específicamente las enfermedades del sistema nervioso. Estos términos se refieren también a las disciplinas de investigación que incluyen el diagnóstico y tratamiento de estas enfermedades. La neurología se ocupa de las enfermedades del sistema nervioso central y periférico, como la Esclerosis Lateral Amiotrófica (*ELA*) y los accidentes cerebro-vasculares, mientras que la psiquiatría se centra en los trastornos afectivos, conductuales y cognitivos. La neuropatología se centra en la clasificación y mecanismos patogénicos subyacentes del sistema nervioso central y periférico y las enfermedades del músculo. Los límites entre estas especialidades son cada vez más difusos, y todas ellas están influenciadas por la investigación neurocientífica. [2]

1.1. - El uso de animales

La experimentación en animales, o pruebas in vivo, es el uso de organismos modelo en sustitución de los humanos. Un organismo modelo en biología es una especie muy estudiada para entender fenómenos

biológicos particulares, que puedan darnos una idea de cómo funcionan esos procesos en otros organismos. En particular, los organismos modelo son frecuentemente usados para analizar las causas de enfermedades humanas y los posibles tratamientos de éstas, ya que la realización de estas pruebas en humanos es contraria a la bioética. [2]

En las investigaciones realizadas con vertebrados, los ratones y las ratas son los más comúnmente utilizados debido a su disponibilidad , su tamaño (fáciles de manejar), su bajo coste, y la tasa de reproducción rápida. Esta última característica se pincela como esencial debido a la necesidad de tener una amplia población a considerar en un experimento con el fin de obtener resultados coherentes. Las ratas de laboratorio son consideradas un animal modelo y su uso abarca desde estudios de fisiología a etología o neurobiología. La velocidad de reproducción , la facilidad de manejo y muchas de sus similitudes fisiológicas con el ser humano han favorecido la utilización de la rata desde hace muchos años como sujeto de experimentos en los laboratorios de ciencias biológicas. Con ellas suelen ponerse a prueba los medicamentos que luego se aplican como tratamiento de las enfermedades humanas o se intenta lograr un diagnóstico precoz de las mismas (enfermedades degenerativas, por ejemplo); de igual modo, se llevan a cabo numerosos experimentos relacionados con la genética, con el sueño y con muchos otros temas de la salud. También han resultado muy útiles en los estudios psicológicos acerca del aprendizaje y otros procesos mentales. [2]

1.1.1. - La rata *Sprague Dawley*

La rata *Sprague Dawley* es una raza consanguínea polivalente de rata albina se utiliza ampliamente en la investigación médica. Su principal ventaja es su tranquilidad y facilidad de manejo.

1.2. - Registro de señales

Conviene dedicar en esta memoria un pequeño apartado a la naturaleza de las señales a registrar, ya que el proyecto se basa en la adquisición y el almacenamiento de éstas.

1.2.1. - Componentes del sistema nervioso

El sistema nervioso está formado fundamentalmente por dos tipos de células: neuronas y células gliales.

El tejido nervioso está compuesto por unas células especializadas llamadas neuronas, de distintas formas y dimensiones pero todas con una estructura común. Cada neurona tiene un cuerpo celular del cual surgen dos tipos de prolongaciones: las dendritas, ramificaciones cortas y arborescentes a través de las cuales le llegan los impulsos procedentes de otras células nerviosas, y el axón, o cilindroeje, extensión única y larga a través de la cual transmite los impulsos a otras células nerviosas o a los tejidos del cuerpo.

- **Dendritas**: Son la estructura más ramificada de la célula, constituyendo los puntos de entrada de señales procedentes de otras neuronas. Las diferentes señales de entrada serán integradas y codificadas al comienzo del axón.

- **Axón**: Constituye la salida de señal, dirigiéndola hacia las dendritas de una o más neuronas. Apenas ramificado en comparación con las dendritas, suele presentar una estructura alargada con longitudes que van desde unos pocos micrómetros en los circuitos locales del sistema nervioso central hasta varios metros en los nervios periféricos de algunos mamíferos. Se observa pues que una neurona puede recibir información de varias neuronas distintas a través de sus dendritas pero integrará esta información y transmitirá una señal de salida a través del axón, de nuevo, a una o varias neuronas receptoras. Esta comunicación axón-dendrita, llamada sinapsis, no se realiza mediante una conexión física directa, sino que existe un espacio denominado hendidura sináptica a través de la cual se efectúa un intercambio químico de unas sustancias conocidas como neurotransmisores que son segregados por la terminación sináptica del axón en respuesta a la señal transmitida e inducen esta misma señal de nuevo en la célula de destino. El otro tipo de células presentes en el sistema nervioso, las células gliales, juegan un papel bastante distinto. Pese a existir en una proporción aproximadamente tres veces mayor que las neuronas, no forman sinapsis ni transmiten señales, sino que cumplen funciones de apoyo. Esto no significa que no jueguen ningún papel en la comunicación neuronal, ya que entre sus tareas está mantener y ajustar el balance electroquímico del medio y modular la presencia de neurotransmisores en las sinapsis.

1.2.2. - Características de las señales neuronales a estudiar

Las señales neuronales son de pequeña magnitud y de baja frecuencia. Las amplitudes típicas varían en el orden de las decenas y centenas de microvoltios y tienen la mayor parte de la energía centrada en la banda situada entre los 10 *Hz* y los 8 *KHz*.

La amplificación y el filtrado de estas señales queda determinada por estas características. Aunque el ancho de banda llega a los 10 *KHz*, la mayor parte de la potencia espectral se encuentra por debajo de los 8 *KHz*, esto requiere una frecuencia de muestreo mínima de 15 *KS/s* para evitar problemas de *aliasing*. [1]

Hay dos tipos de señales neuronales: Potenciales de Acción (*spikes*) y Potenciales de Campo Local (*LFP*).

Los Potenciales de Acción o spikes son impulsos eléctricos que se generan en determinadas células del cuerpo humano, normalmente las del sistema nervioso, y se transmiten a través de la membrana celular modificando su distribución de carga. Los potenciales de acción son la vía fundamental de transmisión de información en el sistema nervioso.

El potencial de campo local (*LFP*) es la actividad eléctrica integrada de un gran número de neuronas vecinas , y está compuesto por fluctuaciones de voltaje extracelular de baja frecuencia (entre 1 y 100*Hz*, habitualmente), reflejando una contribución principal de potenciales sinápticos así como contribuciones de señales relacionadas con potenciales de acción. El *LFP* es una medida electrofisiológica de interés creciente en la comunidad neurocientífica porque puede proporcionar un valioso vínculo entre los registros de neuronas individuales y medidas globales neurofisiológicas tales como el *EEG*, la imagen por resonancia magnética funcional (*fMRI*) y el *ECoG*. Los neurocientíficos lo han usado con mayor frecuencia en los últimos años para relacionar la actividad neuronal con la percepción y la cognición, incluyendo la codificación de los estímulos sensoriales, la atención y la memoria de trabajo . [3]

1.2.3. - Sistemas de registro de señales electrofisiológicas

Un sistema de registro de señales electrofisiológicas se compone de cuatro elementos principales: electrodo, *headstage*, módulo de adquisición

y *software* de registro. En este proyecto sólo se alcanza hasta el módulo de adquisición, pero a continuación se va a exponer todos los elementos para dar una visión más amplia:

- Electrodo: Es la parte que está en contacto con el cerebro y aunque los primeros micro electrodos únicamente podrían medir en una sola zona y tenían frecuentes problemas de biocompatibilidad, los avances en áreas como biomateriales o la miniaturización en dispositivos de silicio han permitido desarrollar nuevos dispositivos que solucionan los problemas de rechazo y proporcionan una alta densidad de canales de registro. A continuación se presentan algunos ejemplos de micro electrodos:

    - Agrupaciones planas: Son los más simples y menos invasivos. Consisten en una matriz de electrodos que se sitúa sobre la corteza cerebral. A pesar de que su precisión es hasta tres veces inferior a las de sondas de inserción, son útiles para los casos en los que se requiere poca precisión, al tener la colocación menos traumática.

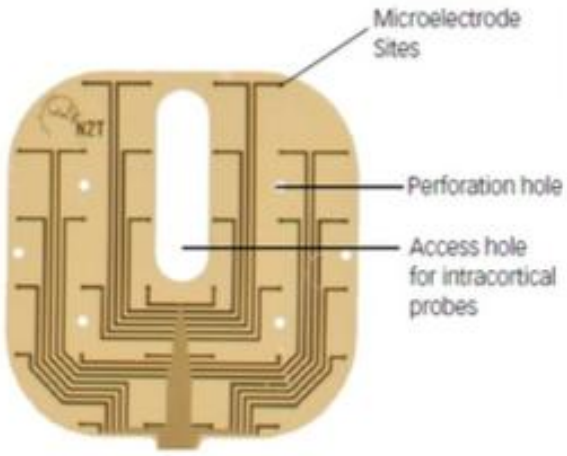


Figura [3]. Agrupación plana

    - Electrodos tipo *Utah*: Estos electrodos, con forma de parche de unos pocos mm de lado, constan de una matriz de micro-agujas de unos pocos milímetros de longitud que se sitúa en la superficie del cerebro. Cada una de estas agujas es capaz de registrar una única señal, situada en la punta de la misma. De

este modo se ajustan tanto la longitud de las agujas como la separación entre ellas para registrar en las zonas deseadas.

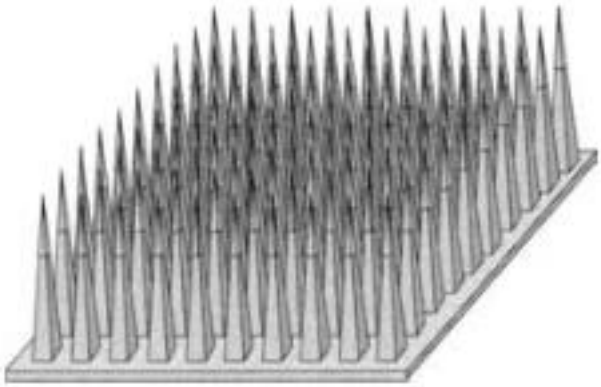


Figura [4]. Electrodo tipo Utah

- Sondas de varilla: Estas sondas, al estar basadas en sustratos de silicio, tienen un inconveniente en su fragilidad. No obstante, este inconveniente se ve altamente compensado por su versatilidad. En este tipo de sondas, cada varilla dispone de múltiples electrodos dispuestos bien uniformemente, bien en agrupaciones de interés. Además, cada electrodo puede fabricarse con diferentes configuraciones, adaptadas para las necesidades concretas de cada investigación.

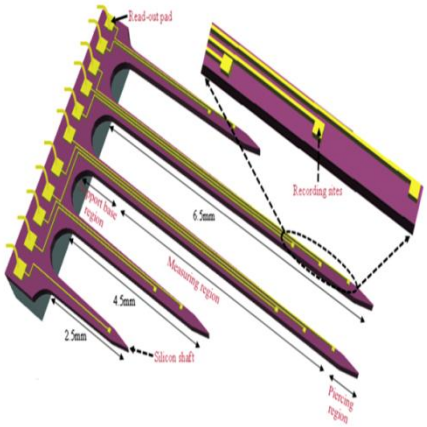


Figura [5]. Electrodo de varilla

A continuación se presentan los dos bloques que más relevancia van a tener en este proyecto: el *headstage* y el sistema de adquisición:

- *Headstage*: Debido a la pequeña amplitud de las señales a registrar y la elevada impedancia de los electrodos, la señal debe ser amplificada lo más cerca de estos posible. Por otro lado, dado que normalmente los experimentos se realizan con animales de pequeño tamaño, hay limitaciones en el equipo a colocar en la cabeza del animal. Lo habitual es que los electrodos posean un conector sobresaliendo del cráneo del sujeto al que se le conecta un pequeño circuito que realiza una pre amplificación de la señal. Este se tratará con más detalle en apartados posteriores.

- Sistema de adquisición: Las señales provenientes de uno o más *headstage* se conectan a través de un cable con el suficiente blindaje y buenas características conductoras, a estos equipos que realizan la amplificación total de la señal, pre tratado en algunos casos, y digitalización de la misma para su transmisión a un equipo informático o registro en un sistema de memoria. Éste es básicamente el objeto del proyecto y se verá con detalle a lo largo de toda la memoria.

- Software de registro: Este software nos permitirá realizar filtrados, observar las señales en tiempo real o detectar *spikes*. Algunos ejemplos son: *MATLAB* (Análisis matemático), *KlustaViewa* (detector de *spikes*)

2. - Material y métodos

Engineering is a great profession. There is the satisfaction of watching a figment of the imagination emerge through the aid of science to a plan on paper. Then it moves to realisation in Stone or metal or energy. Then it brings homes to men or women. Then it elevates the standard of living and adds to the conforts of life. This is the engineer's high privilege.

Herbert Hoover

Para la realización de este proyecto se ha seleccionado el micro controlador *NXP LPC1768*, utilizando la plataforma *mbed* para su programación. Este dispositivo y esta plataforma se proponían en la memoria del Proyecto Final de Carrera de Javier Cambra[1].

Como *headstage* se ha utilizado el *Intan RHD2132*, propiedad del *Instituto de Neurociencias de Alicante*, y como memoria de almacenamiento, tras un estudio previo, se ha decidido utilizar una *microSD* de 16*GB*.

Más adelante, debido a problemas en el cumplimiento de los requerimientos mínimos, se decidió utilizar una *FPGA*. El *Departamento de Ingeniería Electrónica* me prestó el kit de evaluación, disponible en el laboratorio de la *ETSIT*, *Altera DE2-115*. Para su programación se utilizó el *software* de *Altera Quartus II 13.0sp web edition*. Ésta es la versión gratuita del *software* en su página *web*.

2.1. - Plataforma de evaluación *mbed*

La plataforma de desarrollo *mbed* permite desarrollar rápidamente productos basados en la gama de micro controladores *ARM Cortex*. Incluye un entorno de desarrollo de *software* (*SDK* por sus siglas en inglés) para micro controlador *open source* basado en *C/C++* en conjunto con un compilador online que incluye un conveniente sistema de control de versiones. La plataforma está soportada por una comunidad de miles de

usuarios a nivel mundial que constantemente aportan nuevas librerías en forma gratuita y ejemplos para ser utilizados de referencia o desarrollar nuevas aplicaciones. El *SDK* se distribuye bajo la licencia *Apache 2.0*, de modo que se puede utilizar tanto en proyectos personales o con fines comerciales sin restricciones [4].

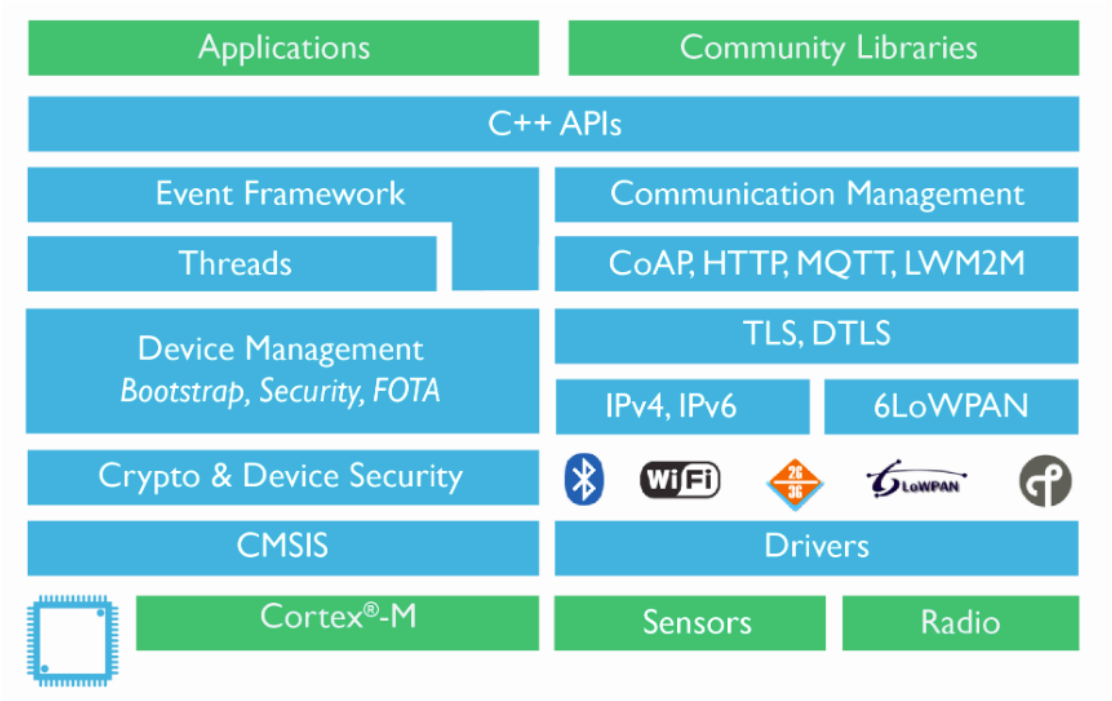


Figura [6]. Posibilidades de la plataforma mbed

2.2. - NXP LPC1768

La placa *LPC1768* permite evaluar prototipos que requieran alta capacidad de procesamiento. Cuenta con un micro controlador *ARM Cortex M3* de 32 *bits* con un reloj de 96 *MHz*. Incluye una memoria *RAM* de 32 *KBytes* y una memoria FLASH de 512*Kbytes* [5]. También posee múltiples interfaces de comunicación incluyendo *USB*, *SPI*, *UART*, etc. Todos los pines del micro controlador están expuestos para su interconexión simplificada. La placa se puede alimentar directamente mediante el puerto *USB* o utilizando una fuente externa de 5*V* a 9*V* ya que posee un regulador de potencia integrado. Cuenta con un programador integrado de memoria *FLASH* que permite subir fácilmente nuevos códigos a través del puerto *USB* [5].

En este proyecto se utiliza esta placa para realizar una simulación hardware de la interacción entre el sistema de adquisición y el de

almacenamiento de registros electrofisiológicos. Para ello se han usado las dos interfaces *SPI*.

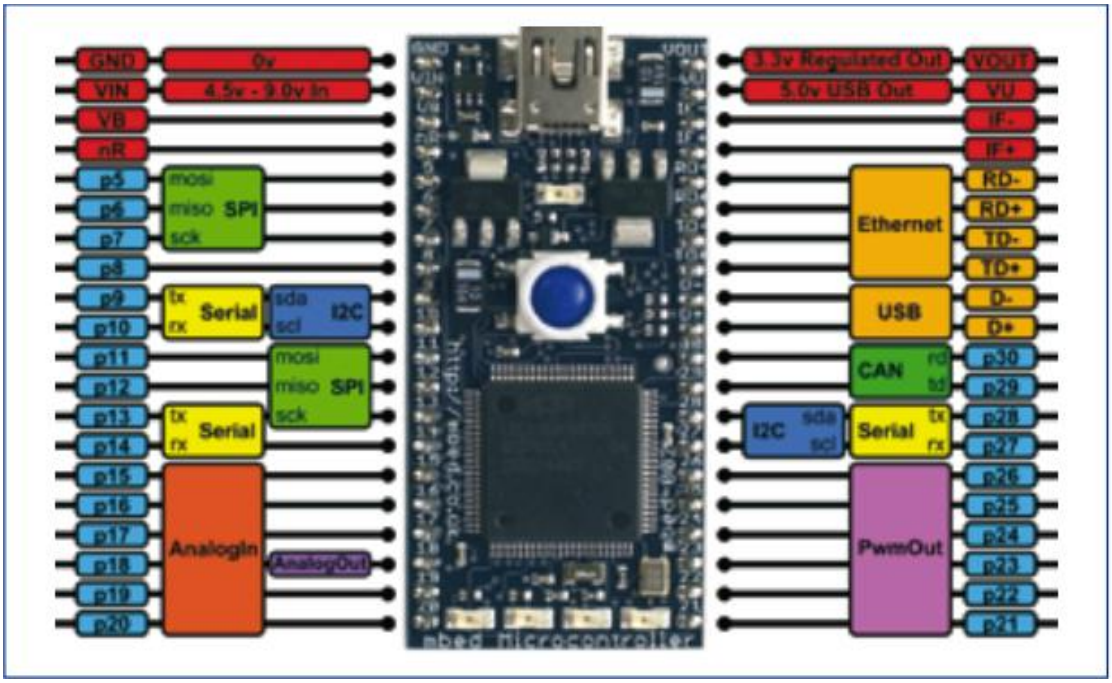


Figura [7]. Placa de prototipado *NXP LPC1768*

Esta placa de prototipado se ha utilizado en *PFC*'s anteriores al mío, también en colaboración con el Centro de Biomateriales e Ingeniería Tisular, y, por lo tanto, ya se encuentra disponible en el grupo de investigación.

2.3. - *Quartus II* web edition

Quartus II es una herramienta de software producida por Altera para el análisis y la síntesis de diseños realizados en *HDL*. Permite al desarrollador compilar sus diseños, realizar análisis temporales, examinar diagramas *RTL* y configurar el dispositivo de destino con el programador. [6]

La Edición *Web* es una versión gratuita de *Quartus II* que puede ser descargada o enviada gratuitamente por correo. Esta edición permite la compilación y la programación de un número limitado de dispositivos Altera. Se requiere un registro de licencia para utilizar la Edición *Web* de *Quartus II*, la cual es gratuita y puede ser renovada ilimitadamente o de

pago.

2.4. - *Kit* de desarrollo *Altera DE2-115*

La serie *DE2* está al frente de los *kits* de desarrollo debido a su multitud de interfaces que hacen que se pueda acomodarse a diferentes tipos de aplicación. Este *kit* ofrece una *Cyclone IV E*, una *FPGA* de bajo coste soportada por la edición *web*, por lo que los pequeños desarrolladores no tendrán problemas por el coste del desarrollo de software. Además, ésta posee entre sus características: 114.480 elementos lógicos, el mayor número ofrecido por una *Cyclone IV* de la serie E, hasta 3.9 *Mbits* de *RAM* y 266 multiplicadores. [7]

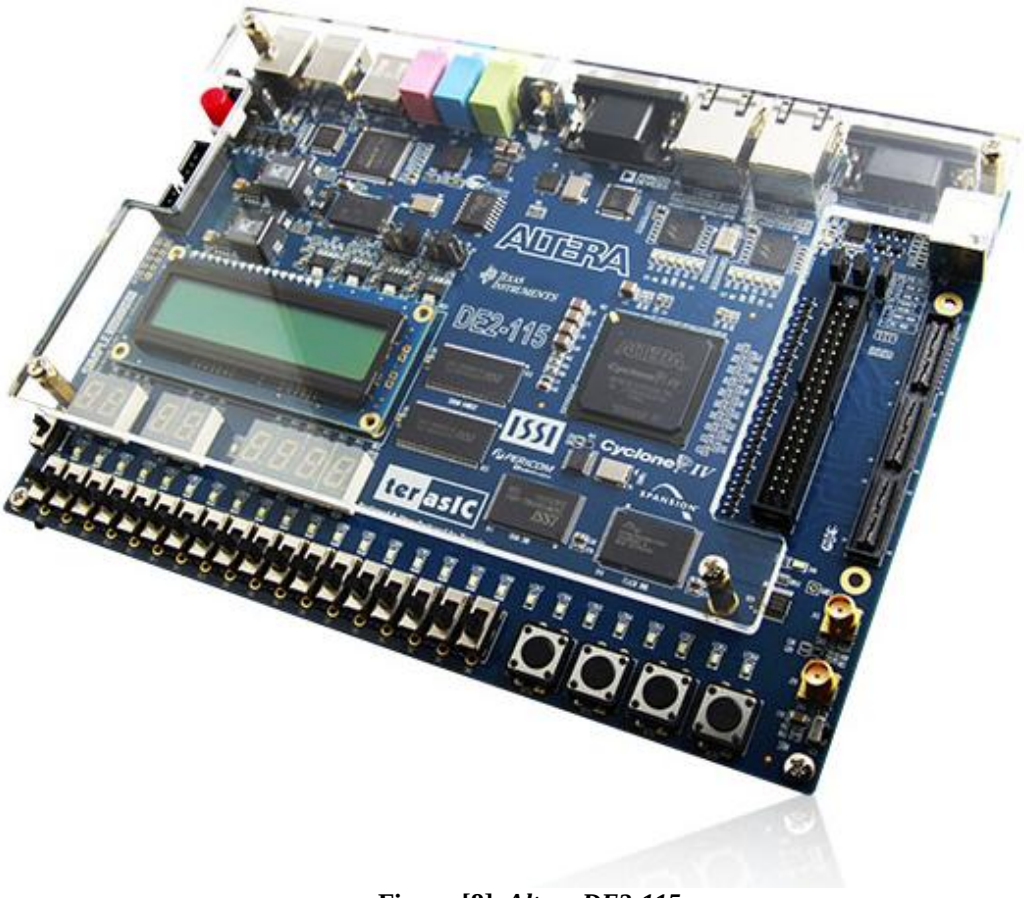


Figura [8]. ***Altera DE2-115***

2.5. - *Intan RHD2132*

Las señales adquiridas por los microelectrodos deben ser acondicionadas y convertidas a formato digital antes de que puedan ser procesadas electrónicamente. Este proceso generalmente se realiza acondicionando las señales mediante amplificadores de bajo ruido y posteriormente representándolas en formato digital mediante conversores de dominio analógico a digital (*ADC*, por sus siglas en inglés). Para este proyecto se plantea el uso de una solución integrada que incorpora todo este proceso, el *Intan RDH2000*. [8]

El microchip de la serie *RHD2000* es una solución de bajo consumo para la adquisición de registros electrofisiológicos desarrollado por la empresa *Intan Technologies*. Estos dispositivos contienen bancos de amplificadores de bajo ruido con anchos de banda programables y son apropiados para una amplia gama de aplicaciones de monitoreo de biopotenciales.

El chip incluye amplificadores, filtros analógicos y digitales configurables, un conversor analógico – digital de 16 *bits* y una interfaz de comunicación *SPI* que permite su integración con otros sistemas digitales. Se pueden muestrear hasta 16 canales simultáneamente y extraer la información en forma digital a través del bus *SPI*. [8]

En la figura siguiente vemos un esquema simplificado del interior del integrado.

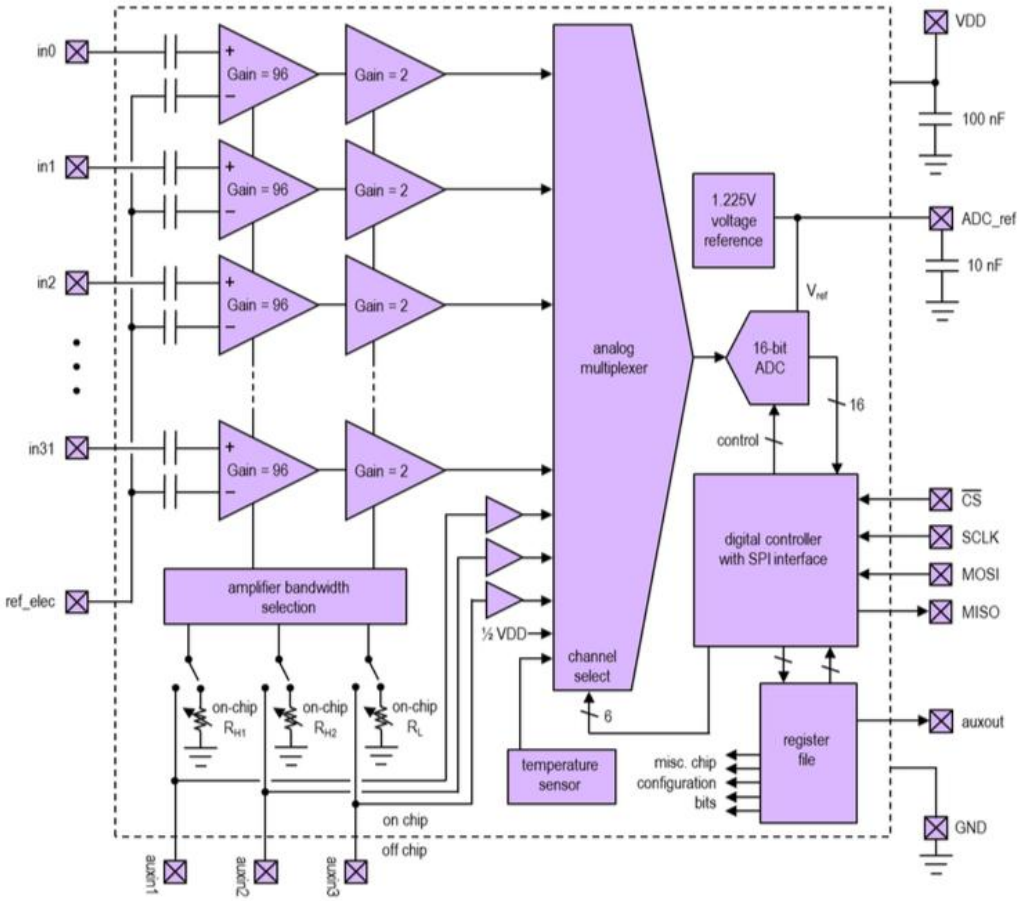


Figura [9]. Esquema de bloques del *Intan RHD2132*

Una de las características que saltan a primera vista es que el ancho de banda de los amplificadores se puede configurar de manera completamente interna mediante los registros digitales. Esto convierte al *RHD* en un circuito altamente integrado para adquisición de señal, requiriendo apenas componentes externos para su funcionamiento.

Más adelante se expondrá la librería realizada para controlar la adquisición de señales.

2.6. - Memoria utilizada

Las tarjetas *microSD (Micro Secure Data)* son memorias *flash* externas que, actualmente, están ganando más popularidad en dispositivos portátiles. Su pequeño tamaño, relativa simplicidad, gran capacidad y

pequeño coste convierten a esta memoria en la solución ideal para muchas aplicaciones. [9]

Figura [10]. Memoria *microSD*

2.7. - Software utilizado

A continuación se presenta la herramienta *software* utilizada: *MATLAB* .

2.7.1. - MATLAB

MATLAB es el lenguaje de alto nivel y el entorno interactivo utilizado por millones de ingenieros y científicos en todo el mundo. Le permite explorar y visualizar ideas, así como colaborar interdisciplinarmente en procesamiento de señales e imagen, comunicaciones, sistemas de control y finanzas computacionales. [10]

Figura [11]. Logo MATLAB

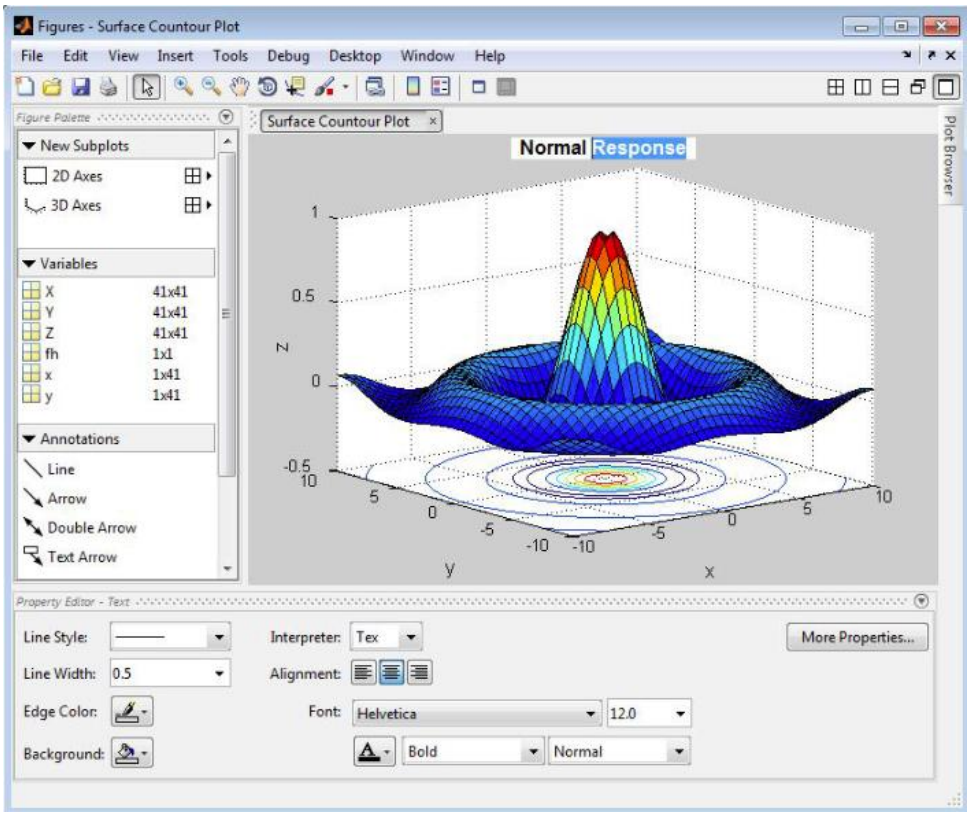


Figura [12]. Software MATLAB

En este proyecto se utilizará *MATLAB* para realizar un estudio frecuencial de las señales almacenadas y para comparar las señales almacenadas con las reales de una forma gráfica. También se utilizará *MATLAB* para comparar las señales mediante el *SSIM* (Índice de simetría estructural).

3. - Resultados

El apartado de resultados se ha dividido en dos grandes bloques: el bloque de adquisición y el de almacenamiento. Posteriormente se presentará el estudio de consumo y los resultados obtenidos.

Para cada uno de los bloques se presentará el diseño realizado con el micro controlador y el diseño realizado con la *FPGA*.

3.1. - Adquisición de registros electrofisiológicos

Las señales con las que se trabaja en este proyecto se adquieren a partir del integrado *Intan RHD2132* antes expuesto.

La serie de integrados *RHD2000*, está formada por un *array* de amplificadores de bajo ruido y baja potencia como su predecesora pero incluyen además un convertidor analógico-digital de 16 *bits* integrado junto a toda la circuitería digital necesaria para la configuración del *chip* y su comunicación con dispositivos externos. [12]

Cada chip de la serie *RHD2000* responde a cinco comandos básicos: conversión analógica-digital de una señal particular, borrar la calibración del *ADC*, escribir en un registro de la *RAM* o leer de un registro de la *ROM* o *RAM*. [8]

La serie *RHD2000* usa un protocolo de comunicación segmentado; cada comando enviado a través de la línea *MOSI* del *SPI* genera un resultado de 16 *bits* que es transmitido a través de la línea *MISO* dos comandos después[8]. La comunicación con el chip se ilustra en la diagrama siguiente:

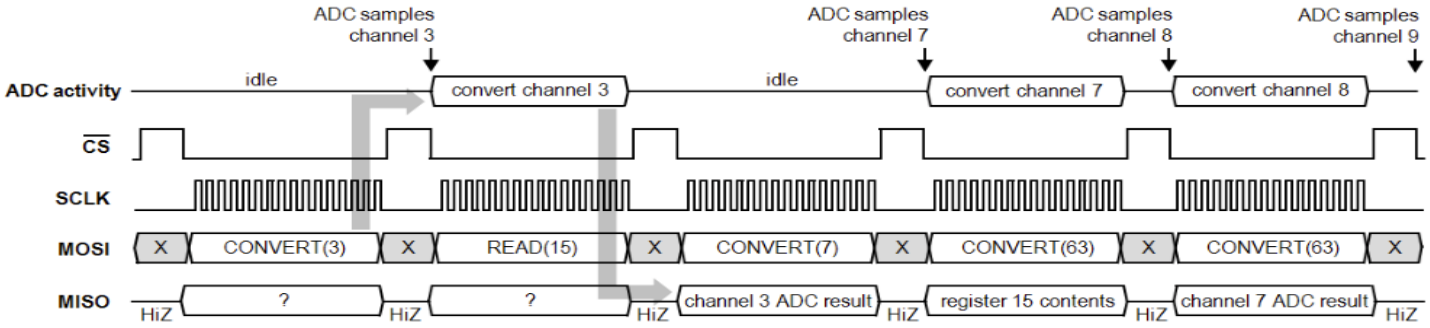


Figura [13]. Diagrama de comunicación Intan RHD2000

3.1.1. - Adquisición mediante microcontrolador

Para la comunicación con este integrado se realizó una librería en *C++* que seguía la siguiente rutina:

1. En primer lugar configuraremos los registros internos del Intan, esta configuración viene determinada por cuatro parámetros: frecuencia de corte del filtro paso-bajo, frecuencia de corte del filtro paso alto, frecuencia de muestreo y frecuencia de corte de la funcionalidad de *DSP* que incorpora el propio *Intan*. A continuación se va a presentar cómo se han configurado estos registros.

Registro 0: Configuración *ADC*

Register 0: ADC Configuration and Amplifier Fast Settle

bit	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Register 0	ADC reference BW [1:0]		amp fast settle	amp Vref enable	ADC comparator bias [1:0]		ADC comparator select [1:0]	

Figura [14]. Registro 0

Este registro configura los conversores analógico-digital. Es un registro de 16 bits, dónde los primeros 8 bits indican el número de registro.[8] Los demás bits se programan como se indica a continuación:

- ***ADC Reference BW [1:0]:*** Esta variable configura el ancho de banda del circuito de realimentación generador de referencia interno del ADC. Como nos indica el catálogo se ha puesto a 3. [8]

- ***Amp Vref enable:*** En una operación normal, este bit se pone a uno para encender los voltajes de referencia usados por los amplificadores de biopotenciales. [8]

- ***ADC comparator bias [1:0]:*** Este valor configura la corriente de bias del comparador del ADC. Se pone a uno para calibración y para uso normal. Si el ADC no se va a utilizar por un periodo largo de tiempo este valor puede colocarse a cero para reducir consumo. Esta variable debe estar a 3. [8]

- ***ADC comparator select [1:0]:*** Esta variable selecciona entre cuatro comparadores diferentes que pueden ser utilizados por el ADC. Debería estar siempre a 2. [8]

Registro 1: Sensor tensión alimentación and *ADC Buffer* corriente *bias*

Register 1: Supply Sensor and ADC Buffer Bias Current

bit	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Register 1	X	VDD sense enable	ADC buffer bias [5:0]					

Figura [15]. Registro 1

- ***VDD sense enable:*** Poniendo a uno este bit activamos el sensor de tension de alimentación. Se puede poner a cero para ahorrar en consumo. [8]

- ***ADC buffer bias current [5:0]:*** Este valor configura la corriente de bias de un buffer de referencia en el *ADC*. Este valor depende de la frecuencia de muestreo. En la figura siguiente se muestra el valor que debemos escoger según la tasa de muestreo para 36 canales. En el caso de tener, por ejemplo, un solo canal tendremos que dividir ese valor entre 36. [8]

ADC sampling rate	ADC buffer bias	MUX bias
≤ 120 kS/s	32	40
140 kS/s	16	40
175 kS/s	8	40
220 kS/s	8	32
280 kS/s	8	26
350 kS/s	4	18
440 kS/s	3	16
525 kS/s	3	7
≥ 700 kS/s	2	4

Figura [16]. Valor del *buffer bias current* en función de la tasa de muestreo

Registro 2: Corriente de bias del *MUX*

Register 2: MUX Bias Current

bit	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Register 2	X	X	MUX bias [5:0]					

Figura [17]. Registro 2

- ***MUX bias [5:0]:*** Esta variable configura la corriente de bias del *MUX* que conduce la señal analógica a la entrada del *ADC*. Esta variable la podemos obtener también en la figura anterior usada para el registro 1. [8]

Registro 4: Formato de salida del *ADC* y eliminación del *offset*

Register 4: ADC Output Format and DSP Offset Removal

bit	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Register 4	weak MISO	twoscomp	absmode	DSPen	DSP cutoff freq [3:0]			

Figura [18]. Registro 4

- ***Twoscomp:*** Este bit hace que el formato de salida del *ADC* sea complemento a 2 con signo. En este caso se pondrá a cero. [8]

- ***Absmode:*** Poniendo este *bit* a uno pasamos la salida del ADC por una función de valor absoluto. Esto equivale a una rectificación de onda completa. Este valor se pone a cero. [8]

- ***DSPen:*** Cuando este bit se pone a uno activamos la opción de usar las funcionalidades de *DSP* para eliminar el offset de los 32 amplificadores usando un filtro paso alto de primer orden *IIR*. [8]

- ***DSP cutoff freq [3:0]:*** Variable que definirá la frecuencia de corte para eliminar el *offset*. Para elegir este valor habrá que seguir la figura siguiente. [8]

DSP cutoff freq [3:0]	k_{freq} ($f_c = k_{freq} \cdot f_{sample}$)
0	differentiator; see below
1	0.1103
2	0.04579
3	0.02125
4	0.01027
5	0.005053
6	0.002506
7	0.001248
8	0.0006229
9	0.0003112
10	0.0001555
11	0.00007773
12	0.00003886
13	0.00001943
14	0.000009714
15	0.000004857

Figura [19]. Elección de la variable que define la frecuencia de corte

Los registros del 5-7 no se han utilizado y se han dejado a cero.

Registros 8-13: Selección ancho de banda del amplificador *on-chip*

Registers 8-13: On-Chip Amplifier Bandwidth Select

bit	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Register 8	offchip RH1	X	RH1 DAC1 [5:0]					
Register 9	ADC aux1 en	X	X	RH1 DAC2 [4:0]				
Register 10	offchip RH2	X	RH2 DAC1 [5:0]					
Register 11	ADC aux2 en	X	X	RH2 DAC2 [4:0]				
Register 12	offchip RL	RL DAC1 [6:0]						
Register 13	ADC aux3 en	RL DAC3	RL DAC2 [5:0]					

Figura [20]. Registros 8-13

- ***Offchip RH1, offchip RH2 y offchip RL:*** Poniendo estos bits a uno cambiamos de usar los resistencias internas programables del *Intan* a usar resistencias externas para definir el ancho de banda. [8]

- ***RH1 DAC1 [5:0], RH1 DAC2 [4:0], RH2 DAC1 [5:0] y RH2 DAC2 [4:0]:*** Estas variables establecen la frecuencia superior de corte de

los amplificadores biopotenciales. En la tabla siguiente podemos ver los valores adecuados para estos registros según la frecuencia de corte que queramos. [8]

UPPER BANDWIDTH f_H	RH1 DAC1	RH1 DAC2	RH2 DAC1	RH2 DAC2
20 kHz	8	0	4	0
15 kHz	11	0	8	0
10 kHz	17	0	16	0
7.5 kHz	22	0	23	0
5.0 kHz	33	0	37	0
3.0 kHz	3	1	13	1
2.5 kHz	13	1	25	1
2.0 kHz	27	1	44	1
1.5 kHz	1	2	23	2
1.0 kHz	46	2	30	3
750 Hz	41	3	36	4
500 Hz	30	5	43	6
300 Hz	6	9	2	11
250 Hz	42	10	5	13
200 Hz	24	13	7	16
150 Hz	44	17	8	21
100 Hz	38	26	5	31

Figura [21]. Valores *RH1 DAC1, RH1 DAC2, RH2 DAC1, RH2 DAC2*

- ***RL DAC1 [6:0], RL DAC2 [5:0], RL DAC3:*** Estas variables establecen la frecuencia de corte inferior de los amplificadores biopotenciales. [8]

LOWER BANDWIDTH f_L	RL DAC1	RL DAC2	RL DAC3
500 Hz	13	0	0
300 Hz	15	0	0
250 Hz	17	0	0
200 Hz	18	0	0
150 Hz	21	0	0
100 Hz	25	0	0
75 Hz	28	0	0
50 Hz	34	0	0
30 Hz	44	0	0
25 Hz	48	0	0
20 Hz	54	0	0
15 Hz	62	0	0
10 Hz	5	1	0
7.5 Hz	18	1	0
5.0 Hz	40	1	0
3.0 Hz	20	2	0
2.5 Hz	42	2	0
2.0 Hz	8	3	0
1.5 Hz	9	4	0
1.0 Hz	44	6	0
0.75 Hz	49	9	0
0.50 Hz	35	17	0
0.30 Hz	1	40	0
0.25 Hz	56	54	0
0.10 Hz	16	60	1

Figura [22]. Valores *RL DAC1, RL DAC2, RL DAC3*

- ***ADC aux1 en, ADC aux2 en, ADC aux3 en:*** Poniendo estos valores a uno cuando el uso de las resistencias internas está seleccionado, active buffers para permitir que los pines *auxin1*, *auxin2* y *auxin3* sean usados como entradas *ADC* auxiliares. [8]

Registros 14-17: Potencia Individual de Amplificadores

Registers 14-17: Individual Amplifier Power

bit	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Register 14	apwr[7]	apwr[6]	apwr[5]	apwr[4]	apwr[3]	apwr[2]	apwr[1]	apwr[0]
Register 15	apwr[15]	apwr[14]	apwr[13]	apwr[12]	apwr[11]	apwr[10]	apwr[9]	apwr[8]
Register 16	apwr[23]	apwr[22]	apwr[21]	apwr[20]	apwr[19]	apwr[18]	apwr[17]	apwr[16]
Register 17	apwr[31]	apwr[30]	apwr[29]	apwr[28]	apwr[27]	apwr[26]	apwr[25]	apwr[24]

Figura[23]. Registros 14-17

- ***apwr [31:0]:*** Poniendo estos bits a cero se apaga el amplificador del número seleccionado. [8]

2. A continuación, se realizará la calibración previa mediante el commando *CALIBRATE*.

Command: CALIBRATE – Initiate ADC self-calibration routine

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Result:

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
*	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Figura [24]. Estructura del comando *CALIBRATE*

El comando *CALIBRATE* inicia una rutina de autocalibración del *ADC*. Esta calibración tarda muchos ciclos de reloj en ejecutarse y, por lo tanto, habrá que enviar nueve comandos *dummy* para asegurarnos de que la rutina habrá terminado. Este comando *dummy* no puede ser el commando de calibración, ya que esta se resetearía. Los comandos enviandos mientras esperamos hasta que la calibración acabe serán ignorados por el Intan. [8]

3. Tras realizar la calibración pasaremos a comprobar que la comunicación *SPI* está activa y funciona correctamente. Para esto leeremos los registros 40-44. Estos registros nos tienen que devolver la cadena *INTAN*. Si no recibimos correctamente esta cadena la comunicación a través del *SPI* no se está produciendo correctamente. [8]

Registros 40-44: Nombre Compañía

Los registros del 40 al 44 contienen los caracteres *INTAN* en *ASCII*. [8]

Una vez la configuración se ha realizado correctamente utilizamos el comando *CONVERT(C)* para obtener una muestra, donde *C* es el canal del que queremos obtenerla.

Command: CONVERT(C) – Run analog-to-digital conversion on channel C

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	C[5]	C[4]	C[3]	C[2]	C[1]	C[0]	0	0	0	0	0	0	0	H

Result:

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
A[15]	A[14]	A[13]	A[12]	A[11]	A[10]	A[9]	A[8]	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]

Figura [25]. Estructura del comando *CONVERT*

3.1.2. - Adquisición mediante *FPGA*

La rutina de inicialización en la *FPGA* es, básicamente, la misma en líneas generales.

Figura [26]. Rutina inicialización

En esta implementación se realizará también el módulo *SPI master* que controlará el *Intan*. El lenguaje utilizado será *Verilog HDL* por ser un lenguaje un tanto más amigable que *VHDL* y la mayor soltura del diseñador con este lenguaje.

El cambio principal lo notamos a la hora de obtener los datos del *Intan*, en vez de simplemente enviar el comando, ahora tendremos una máquina de 80 estados. El comportamiento de esta máquina de estados viene definida en el catálogo del código realizado por *Intan Technologies* llamado *Rhythm* y que se describe a continuación [13].

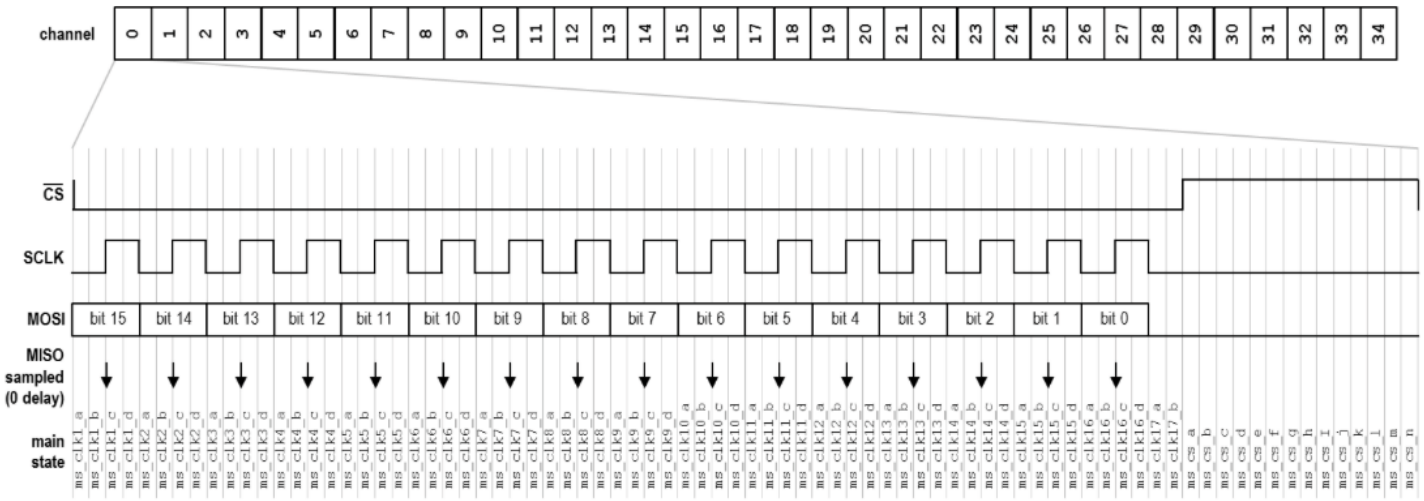


Figura [27]. Máquina de estados comando *CONVERT* para 35 canales

Cada periodo de reloj se compone de cuatro estados llamados *ms_clkX_Y* donde la *X* va de 1 a 16 y la *Y* va de *a* a *d.* Después de 16 ciclos completos hay dos estados adicionales llamados *ms_clk17_a* y *ms_clk17_b* para generar un retardo entre el flanco de bajada del último ciclo de reloj y el flanco de subida del *chip select.* Esta señal se mantiene a *1* por 14 estados que van de *ms_cs_a* a *ms_cs_n*. Después de completar los 80 estados cambiamos la variable del canal. Estas rutinas de 80 estados se repiten para cada canal del que queramos obtener una muestra. [13]

Pese a que se ha seguido la metodología indicada en el catálogo, algunos de los estados están vacíos, ya que muchas de las opciones que presenta este chip no nos interesan para el objeto de este proyecto, por lo que el tamaño de este bloque se podría reducir. En vez de esto, esta máquina de estados se aprovecha para enviar los comandos de inicialización de los registros, calibración y comprobación del enlace SPI. Para ello se ha añadido un estado al principio que comprueba que la inicialización de todos los registros y calibración se hayan realizado.

El modo *SPI* que se ha utilizado para esta implementación es el 2, es decir los datos son capturados en el flanco de bajada del reloj. En las figuras siguientes se ve esto ilustrado.

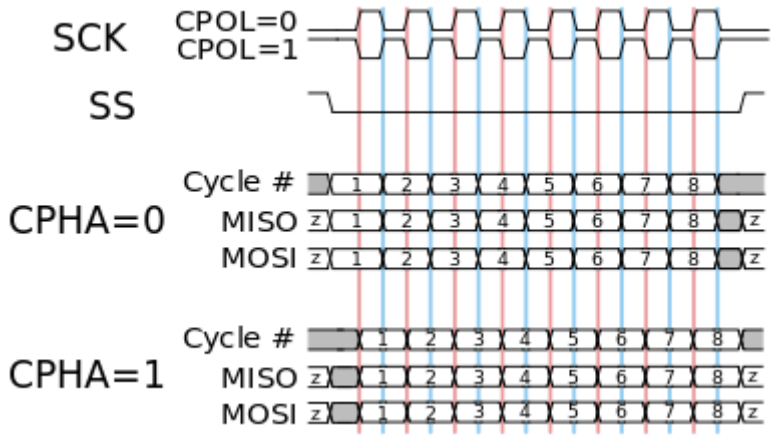


Figura [28]. Diagrama de tiempos enseñando la polaridad y la fase del reloj

SPI Mode	Clock Polarity (CPOL/CKP)	Clock Edge (CKE/NCPHA)
0	0	1
1	0	0
2	1	1
3	1	0

Figura [29]. Modo *SPI*

3.2. - Almacenamiento de registros electrofisiológicos

En este apartado primero se introducirá en el estudio de las posibles memorias a utilizar y, posteriormente, se expondrá la metodología utilizada para almacenar los registros electrofisiológicos.

3.2.1. - Elección de la memoria a utilizar

Para comenzar se va a realizar un pequeño estudio de la tecnología disponible en dispositivos de almacenamiento de datos y la mejor opción para este proyecto.

En ese trabajo se han estudiado las memorias *FeRAM*, *SRAM* y las tarjetas *microSD*. Por último se ha estudiado la posibilidad de utilizar nuevas tecnologías como las memorias *SSD (Solid State Disk)* que actualmente se están volviendo muy populares.

3.2.1.1. - Memorias SRAM

Las memorias *SRAM (Static Random Access Memory)*, memoria estática de acceso aleatorio son un tipo de memoria *RAM* basada en semiconductores capaz de mantener los datos, mientras siga alimentada, sin necesidad de circuito de refresco. Existen dos tipos: volátiles y no volátiles, cuya diferencia está en si los datos permanecen o se volatilizan al retirar la alimentación eléctrica.

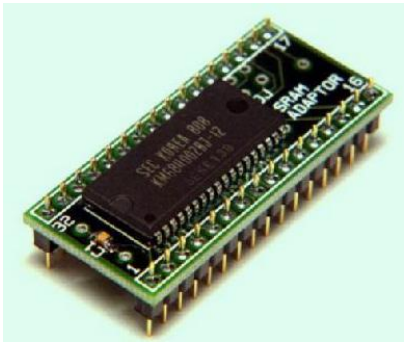


Figura [30]. Memoria SRAM

Las memorias de tipo *RAM* ofrecen buenas prestaciones y además son fáciles de usar.

Cada bit en una *SRAM* se almacena en cuatro transistores que forman un biestable. Este circuito biestable tiene dos estados estables, utilizados para almacenar un *1* o un *0*.

Las características de las memorias *SRAM* que pueden resultar atractivas para este proyecto son:

1. Muy bajo consumo, de pocos mili Amperios en modo escritura.

2. Tiempos de acceso muy bajos, decenas de nano segundos.

Por otro lado, las memorias *SRAM* actuales solo llegan hasta los 32*MB*, por lo que, para este proyecto es insuficiente y al ser algo que no se puede resolver (ya que necesitaríamos unas 100 memorias *SRAM*), esta memoria será descartada.

3.2.1.2. - MicroSD Card

Como se ha presentado anteriormente, las tarjetas *microSD (Micro Secure Data)* son memorias flash externas que, por su pequeño tamaño, relativa simplicidad, gran capacidad y pequeño coste convierten a esta memoria en la solución ideal para muchas aplicaciones [9].

3.2.1.2.1. - SD Standard

SD Standard es un estándar para dispositivos extraíbles de almacenamiento diseñado y con licencia de *SD Card Association* [9]. Este estándar surge de la colaboración de tres importantes fabricantes: *Toshiba*, *SanDisk* y *MEI* y proviene del protocolo anterior, *Multi Media Card (MMC)*.

SD Standard no se limita a dispositivos externos de almacenamiento y ha sido adaptado a diferentes clases de dispositivos, incluyendo tarjetas 802.11, dispositivos *bluetooth* y módems [9].

3.2.1.2.2. - Características

Las tarjetas *microSD* son uno de los muchos dispositivos extraíbles entre lo que podemos encontrar otros estándares como: *CF*, *CF+* y *USB*. Estos dispositivos ejecutan sus funciones de manera similar, pero difieren mucho en temas como factores de forma, complejidad y/o consumo.

Una tarjeta *microSD* tiene, frente a sus otros competidores extraíbles, las siguientes ventajas:

1. Tamaño: Su tamaño es muy reducido, lo cual es una condición casi obligatoria en nuestro sistema.

2. Interfaz eléctrica: Su interfaz eléctrica es relativamente sencilla, requiriendo como mucho seis conexiones para su comunicación y,

aún así, conseguir grandes velocidades.

3. Consumo: Una *microSD* puede consumir, como mucho, en modo escritura, 100mA. Este consumo es mucho más reducido que otras alternativas como memorias *CF+* o *USB*. Aunque este consumo es mucho mayor que en dispositivos no extraíbles.

Frente a otras memorias no extraíbles lo que convierte a la *microSD* la memoria idónea es su gran capacidad y/o el consumo.

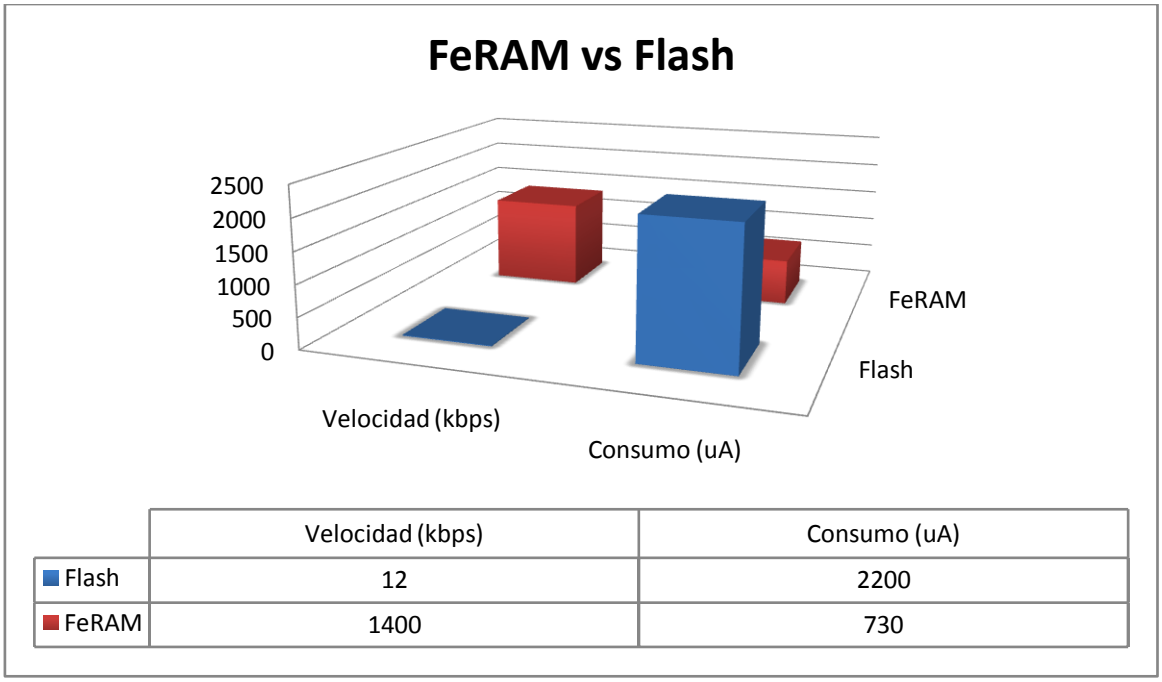
3.2.1.3. - Memorias *FeRAM*

Existe una nueva tecnología como alternativa a las memorias *FLASH* y *SRAM* que se caracteriza por unos consumos realmente reducidos. Se trata de las nuevas memorias *FRAM* [1].

Figura [31]. Memoria F-RAM

Los chips *F-RAM (Ferromagnetic Random Access Memory)* contienen una fina capa de un material ferromagnético llamado Zirconato de *Titanio Pb(Zr,Ti)O*3. Los átomos de *Zr/Ti* cambian de polaridad en contacto con un campo eléctrico, produciendo de este modo un interruptor binario. Al contrario que los dispositivos *RAM*, los *F-RAM* retienen los datos cuando la fuerte de energía es apagada o interrumpida debido a que la polaridad en el cristal del material ferromagnético se mantiene. Esta propiedad única convierte a las memorias *F-RAM* en no volátiles de bajo consumo [1].

Una *F-RAM* tiene una vida 10.000 mayor que una memoria convencional, y consume 3000 veces menos potencia que un dispositivo *EEPROM* y su velocidad de escritura es cerca de 500 veces superior. En la figura [32] podemos ver un gráfico comparativo.



	Velocidad (kbps)	Consumo (uA)
Flash	12	2200
FeRAM	1400	730

Figura [32]. Comparativa de velocidad y consumo entre memorias *FeRAM* y *Flash*

Pese a todas las ventajas que presentan estas memorias su capacidad actual las dejan fuera de ser una opción real para este proyecto específico. Pero es una memoria más que interesante para el proyecto final que pretende desarrollar el *T-RAT Lab*, donde, como se ha comentado anteriormente, se pretende transmitir los datos en tiempo real y por lo tanto sólo se necesita una memoria de capacidad suficiente para almacenar un búfer durante un tiempo.

3.2.1.4. - Memorias SSD (Solid State Drive)

SSD es el acrónimo de *Solid State Disk*, disco de estado sólido en castellano. Este es el nombre que se uso en principio para denominar a la nueva generación de dispositivos de almacenamiento para *PCs* aunque debido a que no llevan discos en su interior en la actualidad es más correcto usar *Solid State Drive*, es decir unidad de estado sólido. En estos al

contrario que ocurre con los discos duros convencionales se utiliza una memoria formada por semiconductores para almacenar la información.

Figura [33]. Memoria SSD

Su funcionamiento es el siguiente: poseen dos zonas de memoria, una en la que se guarda toda la información aunque deje de tener corriente eléctrica y otra, de mucho menor tamaño, que actúa de cache acelerando los accesos. Esta última es muy parecida a la memoria *RAM*.

Todo este sistema es gobernado por un controlador que actúa coordinando los distintos elementos. En realidad, nos encontramos con varios bloques de memoria que actúan como un *RAID* en miniatura el cual nos ofrece tanto aumentar la velocidad al poder realizar varias lecturas y escrituras al mismo tiempo como hacer que el dispositivo sea más resistente a fallos.

Las características de las memorias *SSD* que pueden resultar atractivas para este proyecto son:

1. Tiempos de accesos muy bajos.

2. Una gran capacidad: desde 32*GB* hasta 1*TB*.

La gran desventaja de estas memorias y la razón por la que se hace inviable es su consumo. Estas memorias consumen incluso más que un disco duro. En escritura esto es un consumo aproximado de 1*A*.

3.2.1.5. - Memoria elegida

De las memorias estudiadas la única que por consumo, tamaño, capacidad y velocidad encaja en este proyecto es la memoria *microSD*. Nos aporta la capacidad necesaria, su tamaño es muy reducido, su consumo es algo que nos podemos permitir, es un dispositivo con una interfaz relativamente sencilla, es extraíble y nos permite usar un sistema de archivos.

Como se verá más adelante en este trabajo, en la parte de almacenamiento, han surgido muchos más problemas de los que esperábamos con la memoria *microSD*, aún así sigue siendo la mejor opción para nuestro propósito.

3.2.2. - Estudio interfaz microSD

A continuación se va a presentar la interfaz eléctrica y las rutinas de inicialización y escritura en una memoria *microSD*. Estas rutinas sirven tanto para microcontrolador como para *FPGA*. Al igual que en la parte de adquisición, para microcontrolador se ha usado el driver *SPI* de la plataforma *mbed*, mientras que para el diseño *Verilog HDL* se ha implementado desde cero.

3.2.2.1. - Interfaz Eléctrica

La interfaz eléctrica de una memoria *microSD* es la siguiente:

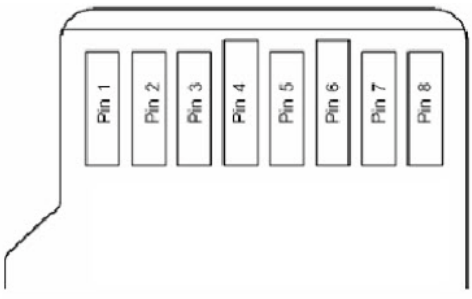


Figura [34]. Memoria *microSD*

Pin	Nombre	Función SD	Función SPI
1	DAT2	Línea de datos 2	Reservado
2	CD/DAT3	Card Detect/ Línea de datos	Chip Select (Nivel Bajo)
3	CMD	Comando/Respuesta	Data In
4	VDD	Alimentación	Alimentación
5	CLK	Reloj	Reloj
6	VSS	Tierra	Tierra
7	DAT0/DO	Línea de datos 0	Data Out
8	DAT1/IRQ	Línea de datos 1	Reservado

Tabla [1]. Asignación de pines

Las *microSD* soportan tres protocolos:

1. **Protocolo de *1-bit***: es un protocolo serie síncrono con una línea de datos, usada para la transferencia de datos en masa, una línea de reloj para sincronización y una línea de comandos. [9]

2. **Protocolo de *4-bits***: éste es muy similar al de 1-bit. La principal diferencia reside en el ancho del bus, transferencia de datos en masa sucede sobre un bus paralelo de 4 *bits* en vez de un solo bit. Ambos protocolos requieren de protección *CRC*, *Cyclic Redundancy Check*, un método para detectar errores en la transmisión de bloques de datos.

   En el protocolo de *4-bits* los datos entrantes son multiplexados en las cuatro líneas del bus y los 16 *bits* del *CRC* se calculan independientemente en cada línea. [9]

3. **Protocolo *SPI***: Se diferencia de los protocolos anteriores en que usa una interfaz de bus conocida y genérica, *SPI*. *SPI* es un protocolo comúnmente utilizado para el control de dispositivos mediante microcontroladores. La comunicación mediante el protocolo *SPI* no soporta algunas de las funciones que los otros dos protocolos sí que pueden realizar. Sin embargo, muchas de estas funciones no son necesarias. Para realizar lecturas y escrituras se puede realizar perfectamente usando el protocolo *SPI*. [9]

En este trabajo se ha seleccionado el protocolo *SPI* por ser el más extendido y no tener ninguna razón de peso, en principio, para seleccionar otro tipo de protocolo.

3.2.2.2. - Comandos

Los comandos presentan dos formatos *CMDXX* o *ACMDXX*. Éstos se refieren a comandos generales o específicos de aplicación respectivamente. *XX* representa el número del comando. [9]

Primer Byte	Bytes 2-5	Último Byte
0 1 Comando	Argumento	CRC 1

Tabla [2]. Formato de comando

La *microSD* responde a cada trama recibida con otra trama de respuesta. Cada comando espera un tipo de respuesta y ésta depende únicamente del número del comando y no del contenido cada trama. Para el protocolo *SPI* hay tres tipos de respuesta:

1. Respuesta tipo R1:

Byte	Bit	Significado
1	7	Bit Start (0)
	6	Parámetro Error
	5	Dirección Error
	4	Borrar Secuencia Error
	3	Error de CRC
	2	Comando Ilegal
	1	Borrar Reset
	0	Idle State

Tabla [3]. Respuesta tipo R1

2. Respuesta tipo R2:

Byte	Bit	Significado
1	7	Significado

	6	Parámetro Error
	5	Dirección Error
	4	Borrar Secuencia Error
	3	Error de CRC
	2	Comando Ilegal
	1	Borrar Reset
	0	Idle State
2	7	Fuera de rango
	6	Borrar parámetro
	5	Violación de protección de escritura
	4	Fallo ECC
	3	Error Controlador
	2	Error sin especificar
	1	Error Bloqueo/Desbloque
	0	Tarjeta Bloqueada

Tabla [4]. Respuesta tipo R2

3. Respuesta tipo R3:

Byte	Bit	Significado
1	7	Significado
	6	Parámetro Error
	5	Dirección Error
	4	Borrar Secuencia Error
	3	Error de CRC
	2	Comando Ilegal
	1	Borrar Reset
	0	Idle State
2-5	Todos	Registro Condición

Tabla [5]. Respuesta tipo R3

Para la escritura/lectura de un bloque, la transferencia es precedida por un bloque de un *byte 11111110*. Esto es seguido por el envío de un bloque de datos, 512 *bytes*, y por último los 16 *bits* del *CRC*.

Después de cada escritura la *microSD* devuelve un bloque de un *byte* con el estado de la operación. Este bloque viene definido por *XXX0AAA1* donde *XXX* no importa los bits que sean y *AAA* indica el estado. *010* señala

que los datos fueron aceptados. *101* se ha producido un error de *CRC*. *110* se ha producido un error en la escritura. [9]

Aunque existen muchos más comandos, en la tabla [6] se ha expuesto los comandos que se utilizarán en este proyecto. Estos comandos se utilizarán para implementar las funciones de inicialización y escritura.

Comando	Argumento	Respuesta	Descripción
CMD0	Ninguno	R1	Resetea y pone la tarjeta en reposo
CMD16	32 bits tamaño	R1	Selecciona el tamaño de bloque
CMD17	32 bits dirección	R1	Lee un bloque
CMD24	32 bits dirección	R1	Escribe un bloque
CMD55	Ninguno	R1	Próximo comando será un comando específico de la aplicación
CMD58	Ninguno	R3	Lee OCR
ACMD41	Ninguno	R1	Inicializa la tarjeta SD

Tabla [6]. Comandos más relevantes en este proyecto

A continuación se va a exponer las rutinas de inicialización y escritura en una *microSD*. Posteriormente se expondrá con detalle como se ha realizado en el microcontrolador y en la *FPGA*.

3.2.2.3. - Rutina de inicialización

Las memorias *microSD* requieren una secuencia específica de inicialización. Algunas partes de ésta son históricas y otras son requeridas para compatibilidad con versiones anteriores y posteriores. *MMC* y *SD* no son sustancialmente diferentes; la principal diferencia desde el punto de vista del software está en la inicialización.

La inicialización comienza estableciendo el reloj del bus *SPI* a 400*kHz*. Esto es requerido para compatibilidad en un amplio rango de tarjetas *MMC* y *SD*.

Después, enviamos al menos 74 ciclos de reloj antes de ningún intento de comunicación. Esto permite a la tarjeta inicializar cualquier tipo de registro interno antes de comenzar con la inicialización de la tarjeta.

Después, la tarjeta es reseteada enviando el comando cero (*CMD0*) con el chip select a cero. Esto resetea la tarjeta y además la fuerza a trabajar en modo *SPI*. Aunque, generalmente, el *CRC* es ignorado en el modo *SPI*, lo siguiente que se hace es enviar un *CRC* válido. El *CRC* para el comando cero con argumento cero es *0x95*. Por simplicidad este *CRC* es siempre enviado con cada comando.

Lo siguiente es enviar los comandos 55 (*CMD55*) y 41 (*ACMD41*) hasta que el bit de reposo se pone a cero, indicando que la tarjeta está completamente inicializada y preparada para responder.

A continuación, el comando 58 (*CMD58*) es usado para determinar si la tarjeta soporta el voltaje de operación del procesador. El comando 58 devuelve un campo de bits con el rango de voltaje aceptado. Generalmente, éste suele estar entre 2.7*V* y 3.6*V*, aunque puede llegar a los 5*V*. En esta aplicación usaremos 3.3*V* para alimentar nuestro sistema.

Por último, el reloj del bus *SPI* se colocará a la máxima frecuencia posible, en nuestro caso 25*MHz*.

A continuación, se ha querido representar esta secuencia en forma de flujograma para ilustrar este proceso en la figura [35].

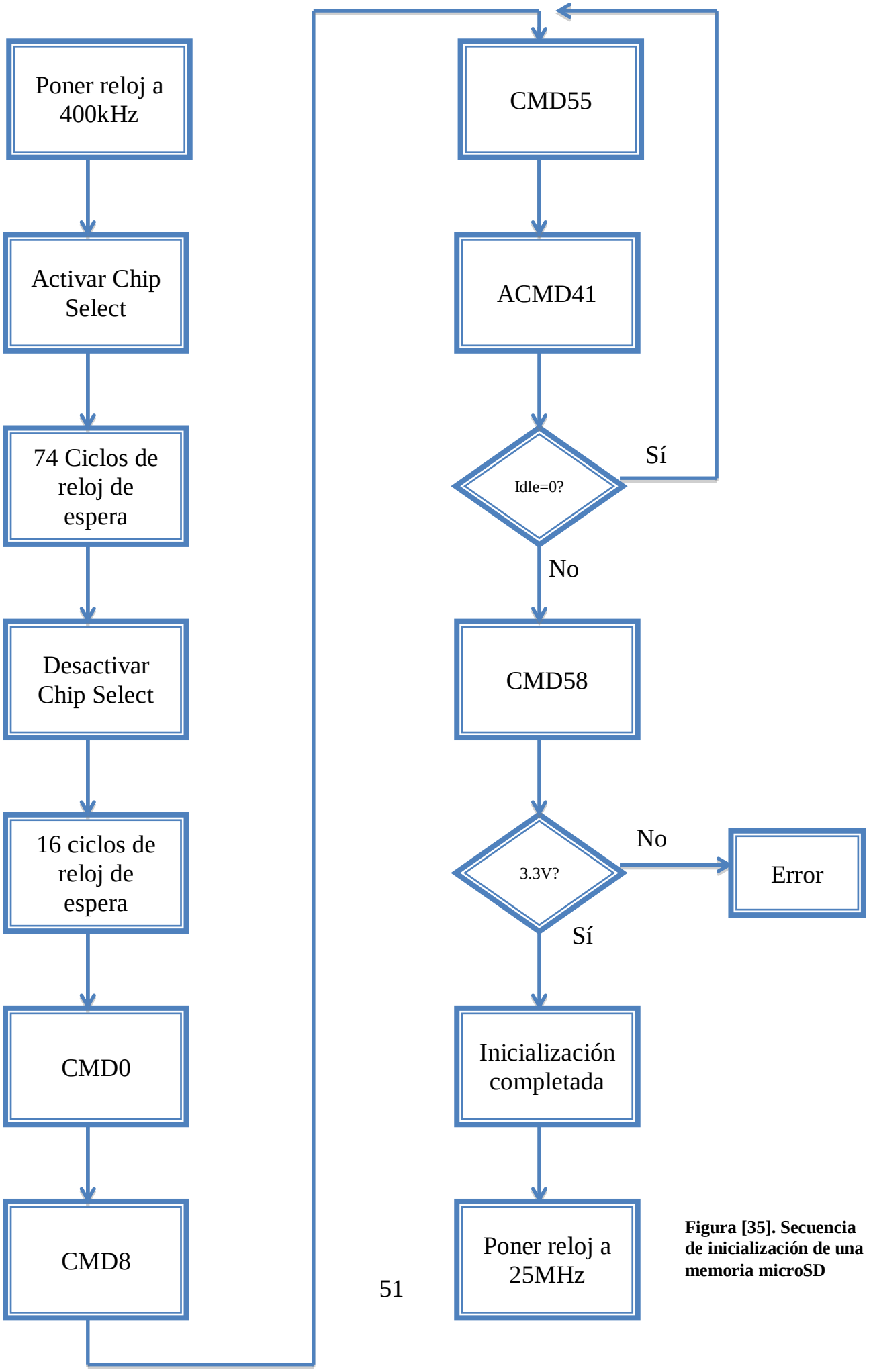


Figura [35]. Secuencia de inicialización de una memoria microSD

3.2.2.4.- Rutina de escritura

La implementación de la rutina de escritura que se utilizará en este PFC será una secuencia específica para la escritura "en crudo" en la tarjeta, aunque en primera instancia, para el microcontrolador, se intentó usar un sistema de archivos como se explicará más adelante.

Lo primero que se realiza es comprobar que la tarjeta está inicializada mediante los registros de estado y, posteriormente, averiguar si la tarjeta está protegida contra escritura para saber si podemos realizar la escritura o tan sólo acceder en modo lectura.

A continuación activamos el chip *select*, poniéndolo a cero, y enviamos 8 bits a uno para activar la recepción de datos.

Lo siguiente es enviar el comando 24 (*CMD24*), éste indica a la tarjeta que se va a realizar la escritura de un bloque (típicamente 512 *Bytes*). [9]

Tras saber que la tarjeta está lista para escribir en ella procedemos a enviar los datos. Posteriormente se realizará la comprobación del *token* de respuesta del comando 24 (*R1*) para saber si los datos han sido escritos.

Por último, desactivaremos el chip *select*, poniéndolo a uno, y enviaremos otro bloque de ocho bits a uno para deshabilitar la recepción de datos por parte de la *microSD*.

A continuación, como se ha realizado para la secuencia de inicialización, se presenta un flujograma con la secuencia de escritura en la figura. [36]

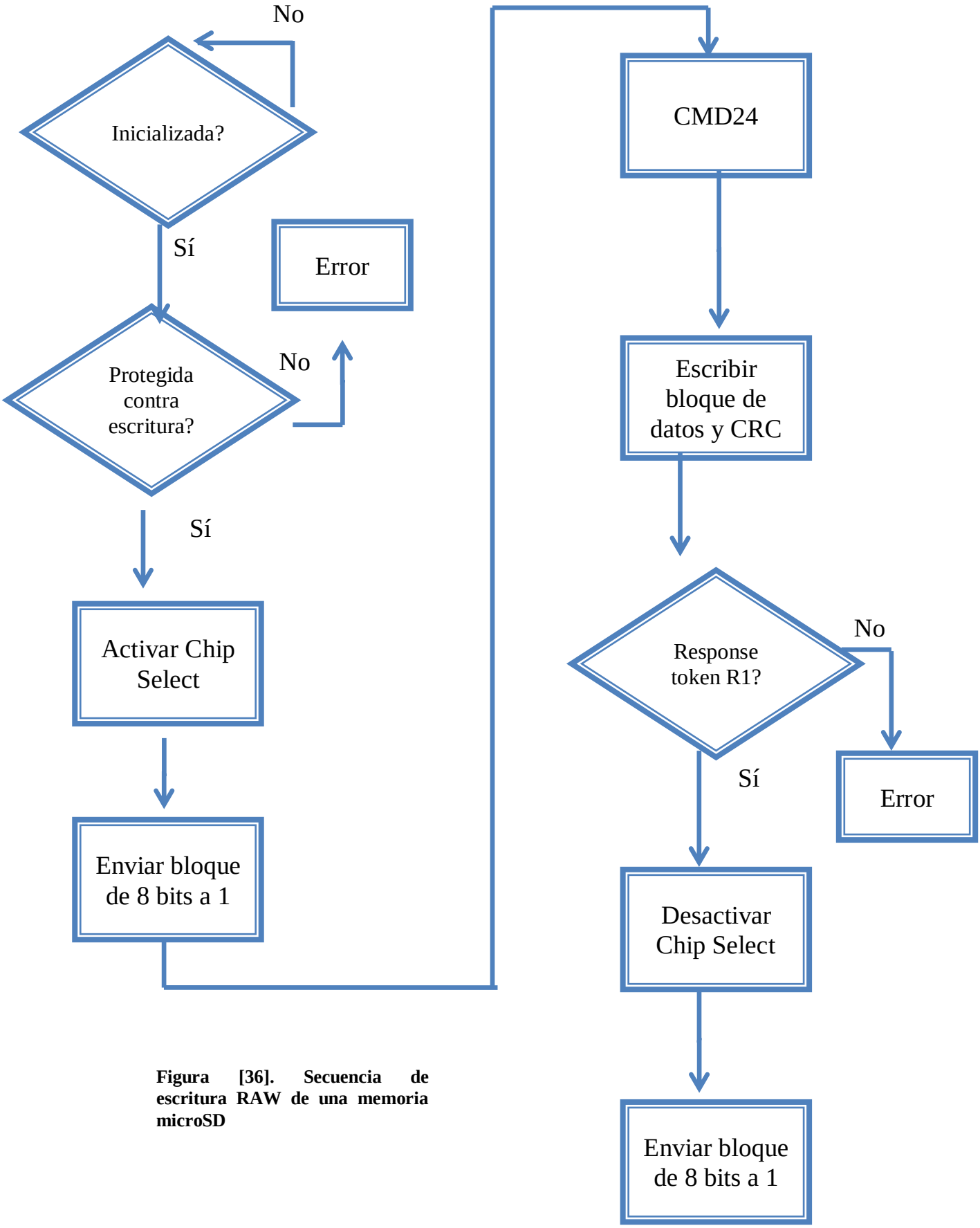


Figura [36]. Secuencia de escritura RAW de una memoria microSD

3.2.2.5.- Tamaño mínimo de la memoria

A la hora de seleccionar la memoria *microSD* a utilizar hemos de tener en cuenta su tamaño. Para ello tendremos que calcular el volumen de datos generado. Esto lo calcularemos a partir de la siguiente fórmula:

$$\boldsymbol{Tasa\ de\ datos}\left(\frac{\boldsymbol{MBytes}}{\boldsymbol{s}}\right) = \frac{f_s(Hz) \cdot Resolución\left(\frac{bits}{muestra}\right) \cdot Nº\ Canales}{1024 \cdot 1024}$$

La fórmula de tasa de datos calcula los *MBytes* generados por segundo, bastará con multiplicar este resultado por el tiempo, en segundos, que durará el experimento.

$$\boldsymbol{Datos\ generados}\ (\boldsymbol{MBytes}) = \frac{f_s\ (Hz) \cdot Resolución\ \left(\frac{bits}{muestra}\right) \cdot Nº\ Canales \cdot Tiempo\ (s)}{1024 \cdot 1024}$$

Para nuestro caso en particular los datos son:

- $\boldsymbol{f_s} = 20.000\ Hz$
- **Resolución** = 16 ($\frac{bits}{muestra}$)
- **Nº Canales** = 4
- **Tiempo** ≈ 2 horas = 7.200 segundos

$$\boldsymbol{Datos\ generados}\ (\boldsymbol{GBytes}) = \frac{20.000\ (Hz) \cdot 16\ \left(\frac{bits}{muestra}\right) \cdot 4 \cdot 7.200(s)}{1024 \cdot 1024 \cdot 1024} = \boldsymbol{8,58\ GBytes}$$

Por lo tanto la capacidad mínima de la memoria ha de ser de 16*GB*. Más adelante se realizará un estudio sobre las velocidades y consumos de diferentes memorias.

3.2.3. - Almacenamiento mediante microcontrolador

El principal problema al que nos enfrentamos en este proyecto es el tiempo de escritura en la memoria.

Antes de proceder con la exposición de los sistemas diseñados cabe tener en mente el modo de trabajo de un microcontrolador. Éstos trabajan muestra a muestra, es decir, obtenemos una muestra del *Intan RHD2132* y antes de obtener la siguiente hemos de realizar todas las operaciones que necesitemos con esa muestra. De lo contrario, estaremos perdiendo muestras.

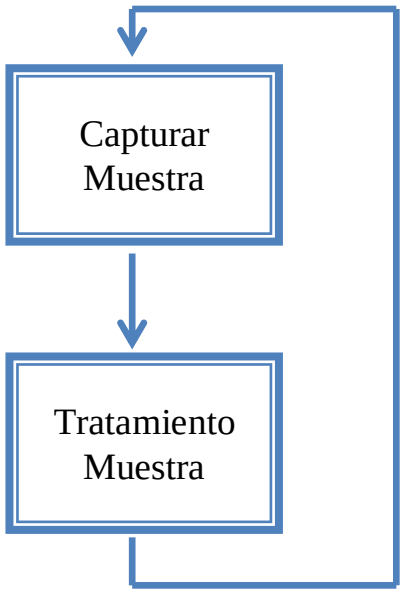


Figura [37]. Funcionamiento microcontrolador

En los apartados anteriores se ha expuesto cómo son las rutinas de almacenamiento y escritura. A continuación, se expondrán los problemas que se han tenido y las soluciones.

3.2.3.1. - Estructura del programa a realizar

La estructura del programa a realizar es la siguiente:

- Inicializar Intan *RHD*
- Inicializar *microSD*
- Crear fichero
- Capturar muestra
- Almacenar muestra

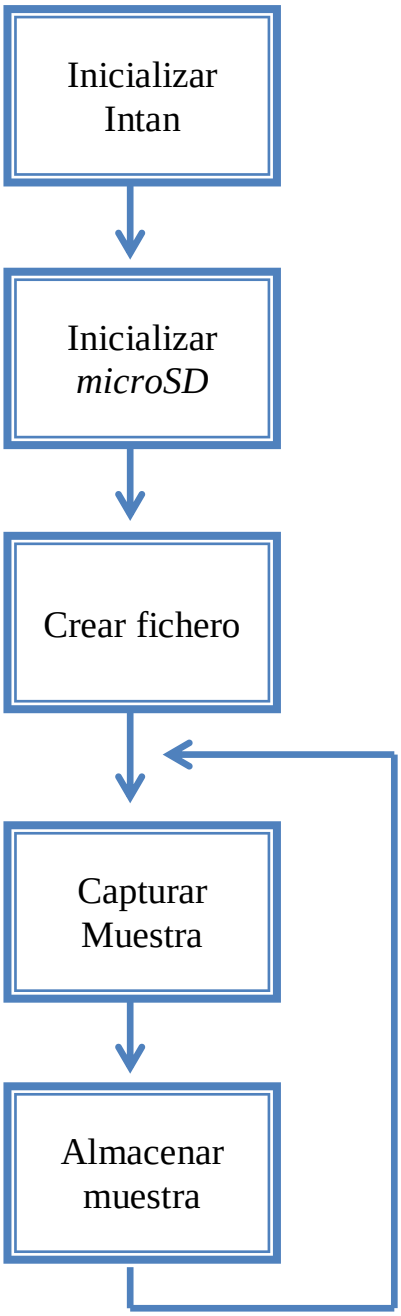


Figura [38]. Estructura del programa a realizar

El Intan se inicializará para una frecuencia de muestreo de 20*kHz* y un ancho de banda de 0.15*Hz* a 10*kHz*.

La *microSD* en un principio la intentaremos hacer funcionar al máximo de sus posibilidades a 25*MHz*.

En un primer intento, crearemos un archivo de texto *.txt* para almacenar las muestras separadas por comas como se expone a continuación.

3.2.3.2. - El uso de sistema de archivos

FAT es con diferencia el sistema de archivos más simple de aquellos compatibles con *Windows NT*. El sistema de archivos *FAT* se caracteriza por la tabla de asignación de archivos (*FAT*), que es realmente una tabla que reside en la parte más "superior" del volumen. Para proteger el volumen, se guardan dos copias de la *FAT* por si una resultara dañada. Además, las tablas *FAT* y el directorio raíz deben almacenarse en una ubicación fija para que los archivos de arranque del sistema se puedan ubicar correctamente. [14]

Un disco con formato *FAT* se asigna en clústeres, cuyo tamaño viene determinado por el tamaño del volumen. Cuando se crea un archivo, se crea una entrada en el directorio y se establece el primer número de clúster que contiene datos. Esta entrada de la tabla *FAT* indica que este es el último clúster del archivo o bien señala al clúster siguiente.

No hay ninguna organización en cuanto a la estructura de directorios de *FAT*, y se asigna a los archivos la primera ubicación libre de la unidad. Además, *FAT* solo es compatible con los atributos de los archivos de almacenamiento, del sistema, ocultos y de solo lectura.

Una de las razones para elegir la *microSD* era su sencillez a la hora de extraer los resultados. El uso de un sistema de archivos hacía que se pudiera escribir las muestras en un simple archivo *.txt*, de manera que, con solo insertarla en el ordenador podríamos tener cada uno de los canales en diferentes archivos *.txt* ya separados para su lectura.

La rutina que se utilizó en primera instancia era la siguiente:

1. Crear el directorio y los archivos en los que se guardará la información.

2. Escribir toda la información muestra a muestra.

3. Cerrar los archivos para que se guarde la información, si no se cierran la información se perdería.

En esta implementación las tramas enviadas a la *microSD* son de 16 *bits* y, además, se escribe muestra a muestra.

Para realizar esta implementación se utilizó la librería proporcionada en la plataforma *mbed SDFileSystem.h*. Esta librería nos permitió realizar un pequeño prototipo con sistema de archivos incluido de una manera muy rápida.

Tras realizar un estudio de tiempos para saber cuánto se tarda en escribir en la *mciroSD* se veía que escribir una muestra no tardaba más que un par de microsegundos y que cada 256 muestras se tardaba un tiempo en torno a los 2000 microsegundos. Esto se debía a que, como se ha comentado anteriormente, no se puede realizar una escritura muestra a muestra y es necesario realizar la escritura en bloques de 512*Bytes*.

Es decir, en esta implementación, lo que se conseguía era escribir 256 muestras en un tiempo alrededor de 2500 *microsegundos*. Muestreando a 20*kHz*, o lo que es lo mismo, tomando una muestra cada 50 *microsegundos*, esto supondría unas pérdidas de en torno a las 50 muestras.

En vista de estos resultados se averiguó que la orden *fprintf* que se usa para escribir en un archivo es muy lenta y por lo tanto se procederá a escribir en binario para ver si de esta forma ganamos tiempo.

Siguiendo con la utilización de un sistema de archivos se intentó escribir en binario en vez de en texto plano. Al escribir en binario se esperaba ganar algo de tiempo, y así fue, aunque seguimos teniendo el mismo problema. En este caso en vez de perder en torno a las 50 muestras se pierde casi 40 muestras cada 256. La orden *fwrite* es algo más rápida que

fprintf. En la figura siguiente podemos ver una muestra de las pérdidas. En esas pérdidas se incluyen varios ciclos de la señal.

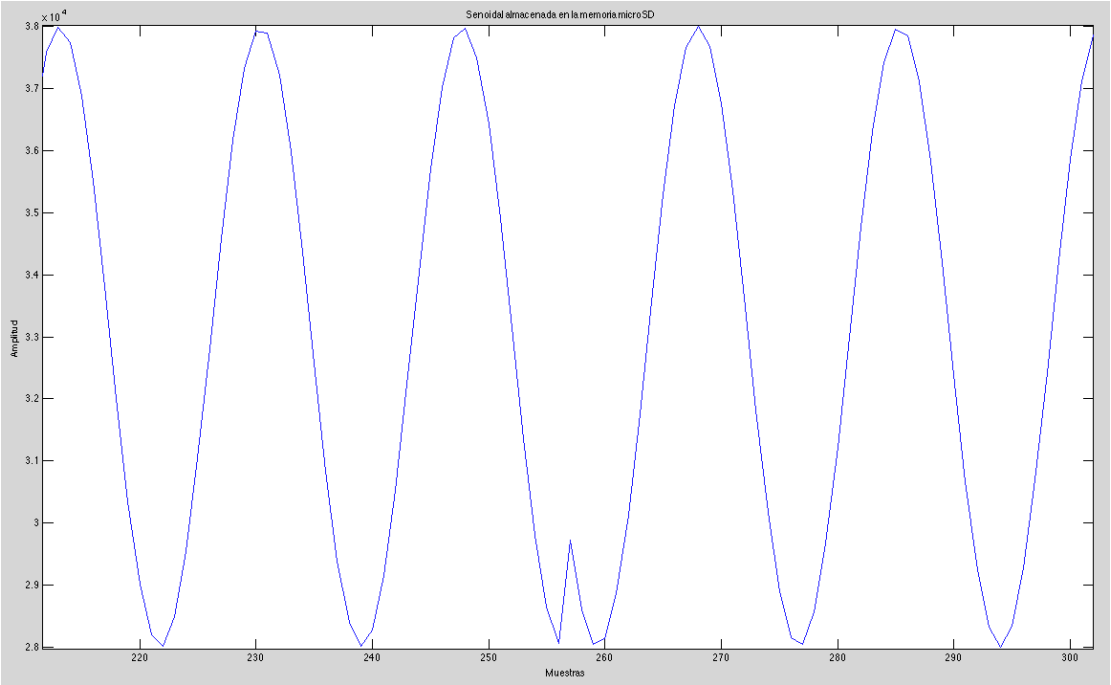


Figura [39]. Senoidal almacenada en la memoria microSD

Ante estos resultados la solución que se plantea es olvidarnos del sistema de archivos *FAT* y escribir en la memoria *en crudo*. De esta forma nos olvidamos de la lentitud de tener que escribir en un sistema de archivos y pasamos a tratar la memoria *microSD* como a una memoria *Eprom*. De esta forma perdemos la sencillez de usar archivos para guardar los datos y luego almacenarlos fácilmente en el ordenador.

3.2.3.4. - El uso de la *microSD* como un dispositivo *RAW*

En primer lugar antes de saber cómo escribir en la *microSD en crudo* lo primero que me planteé fue saber cómo leer una tarjeta sin formato con el ordenador. Para este propósito se realizó un script que usa la terminal *UNIX*. Este script detecta el dispositivo conectado sin formato y extrae todo su contenido a un archivo *.raw*. Este archivo se puede abrir fácilmente con *MATLAB* o cualquier editor hexadecimal. Los comandos utilizados en el script han sido los siguientes:

```
diskutil umountDisk /dev/disk1

sudo dd if=/dev/disk1 of=~/Desktop/output.raw
```

Si abrimos el archivo *output.raw* con un editor hexadecimal obtendremos lo siguiente:

0	01000000	01000000	F14C0000	42560000	53550000	82570000	87580000	4D550000	39590000	55560000	33570000	B1560000	50540000
52	B8550000	2F550000	4C540000	82540000	01570000	68560000	B0540000	DB550000	F6540000	50540000	6C550000	2B550000	EB540000
104	8A560000	63570000	15560000	D6570000	4B570000	F2560000	3F580000	28580000	2D580000	E6570000	6B590000	CE580000	755A0000
156	C7590000	F2580000	FA580000	B5570000	05570000	77560000	63580000	36570000	DA560000	8C580000	89580000	03570000	43580000
208	185A0000	0C580000	FF570000	D6590000	C35A0000	F4590000	F05A0000	A95A0000	6F5B0000	2A580000	1E580000	2E5B0000	81580000
260	46590000	CF5E0000	605A0000	84600000	E35F0000	615D0000	07620000	E7600000	AD610000	AB640000	1A660000	476A0000	60690000
312	F3690000	156A0000	7D690000	35690000	E36B0000	C56C0000	986E0000	306D0000	4F6F0000	6E6F0000	0B720000	8D700000	D9710000
364	81710000	70710000	E9730000	7A730000	C3740000	89750000	CC740000	AC760000	CD760000	4E770000	4A780000	E3770000	7C790000
416	2B790000	E8780000	3F7A0000	297B0000	447B0000	4B7C0000	537B0000	667A0000	237D0000	677B0000	6D7A0000	E97B0000	94790000
468	7B7B0000	467B0000	6B7A0000	317A0000	727C0000	3C7B0000	9B7C0000	5F7E0000	5E7E0000	4A800000	8A830000	F3830000	73830000
520	B2770000	5D7F0000	BF7B0000	F96D0000	20770000	1E850000	F2890000	76800000	C0860000	DF820000	60780000	68760000	35750000
572	B3750000	FC740000	E5740000	14790000	FD760000	267B0000	0A790000	097B0000	5A7E0000	577A0000	687F0000	3C7C0000	017E0000
624	417E0000	027E0000	017F0000	6B7E0000	02800000	057E0000	387F0000	E97E0000	2E800000	437F0000	FB7D0000	2B7F0000	5B7E0000
676	457E0000	427C0000	2B7D0000	937D0000	717E0000	D17D0000	F97E0000	A57E0000	C5800000	C4800000	E0810000	7D810000	ED810000
728	D0830000	1B820000	84850000	43840000	61850000	86860000	EC850000	B6880000	D7880000	65880000	7D920000	43940000	C3990000
780	159A0000	489D0000	8C9F0000	959D0000	6AA20000	25A10000	7CA10000	18A60000	20A50000	12A80000	0BA90000	D5A60000	42A90000
832	9AA90000	86A90000	E8A90000	67AA0000	B7AB0000	ADAA0000	DDAA0000	EFAB0000	78AA0000	67AA0000	CAAA0000	ECAA0000	7AA90000
884	A1AA0000	B0AB0000	62A90000	F2AA0000	D8A90000	A9A90000	3AA90000	90A80000	77A80000	88A70000	23A80000	7CA80000	B0A70000
936	6EA80000	8AA80000	88A70000	5FA80000	4BA80000	E6A70000	99A80000	71A90000	EBA90000	07A90000	29A90000	68AA0000	AFA80000
988	B1A60000	05A90000	FEA50000	3AA50000	2AA60000	DDA30000	55A80000	BBA50000	8BA30000	C6A60000	F2A90000	919A0000	A4A10000
1040	B38D0000	DF960000	9CAB0000	D19E0000	C89E0000	38A10000	E89E0000	DC9C0000	56990000	D5970000	4C990000	B5960000	D6960000
1092	999A0000	E6950000	19960000	C0930000	A4930000	86920000	43900000	BC8F0000	2B8F0000	A98E0000	3C8C0000	1C8E0000	8E8D0000
1144	178C0000	D58C0000	6A8B0000	0E8C0000	668A0000	85890000	84890000	53880000	A2880000	A1870000	CF860000	2F860000	84860000
1196	B3850000	1A850000	16830000	CC830000	DB840000	1C840000	C4830000	D8850000	D7840000	B0840000	82870000	E6840000	56860000
1248	20850000	D4840000	95830000	6D830000	CB830000	DD800000	79810000	08800000	3D7B0000	987D0000	0D7C0000	EE7C0000	617E0000
1300	E47E0000	277F0000	EA7F0000	C57F0000	3F820000	D77F0000	FB7E0000	E07C0000	457C0000	C57C0000	CC7C0000	8E820000	16890000
1352	F28A0000	728C0000	E88B0000	CF8D0000	EA8A0000	1C890000	A4880000	B1860000	54880000	53850000	6E840000	BF830000	3A830000
1404	7E820000	E0810000	B1810000	B67F0000	8B830000	5B810000	43810000	39820000	2F800000	0F810000	B9800000	8A800000	EF800000
1456	D9800000	44820000	6E820000	FF810000	87810000	42830000	AF810000	99820000	7A820000	687F0000	C27F0000	F87E0000	0F7E0000
1508	E37F0000	9C7E0000	A27C0000	E27E0000	A17D0000	157A0000	7A7C0000	EB770000	CA710000	195C0000	E2610000	21510000	29540000
1560	06690000	66620000	355B0000	E1610000	BB600000	D55B0000	5E5A0000	43590000	E7590000	01570000	17570000	B25A0000	53570000
1612	D6560000	F8550000	C6560000	C2550000	15560000	F0550000	4F540000	8E540000	EA540000	9A560000	25560000	12560000	6B560000
1664	C1560000	0A560000	05580000	13560000	EE560000	9B570000	E9570000	55580000	CF570000	66590000	CF580000	135A0000	51590000
1716	0E580000	BF580000	0C580000	88580000	71570000	D3570000	D7560000	52570000	93550000	C4570000	52580000	1E580000	42590000
1768	5C580000	475A0000	F1590000	B7580000	A95A0000	2C5A0000	D1590000	A85C0000	83570000	8C590000	5B5C0000	BB590000	E0590000
1820	7D5D0000	915B0000	415D0000	B4600000	BB5D0000	BA610000	5F620000	AB610000	8E660000	3D670000	2C690000	1A690000	206A0000

Figura [40]. Posiciones de memoria de la microSD

En la figura podemos observar las posiciones de memoria de la memoria *microSD*.

Para abrir este fichero en *MATLAB* se usará la orden *fread* y la precisión de lectura será de de 16 bits sin signo.

En el apartado anterior se ha expuesto la rutina seguida para escribir en la memoria en *RAW*. Para realizar esto se ha modificado la librería proporcionada en la plataforma mbed *SDFileSystem.h*. En esta modificación la librería funcionará exactamente como se ha expuesto en el apartado *Rutina de escritura*.

Aunque era de esperar una mejora en la velocidad, los resultados obtenidos fueron sorprendentes ya que pasamos de unos tiempos de 2000

microsegundos a unos tiempos de 80 microsegundos en escribir el bloque mínimo de 512*Bytes*. Estos resultados suponían unas pérdidas de 1 muestra de cada 257 para un canal y 1 muestra cada 65 para cuatro canales.

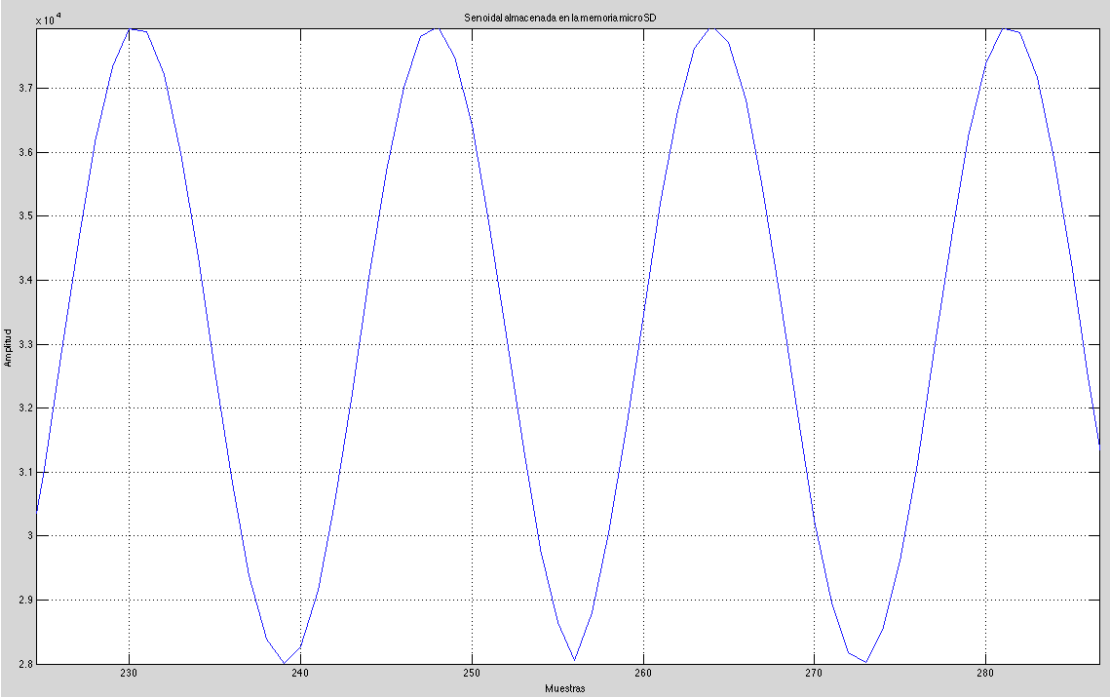


Figura [41]. Senoidal almacenada en la memoria microSD

Lo que estas pérdidas suponen para el proyecto se estudiará más adelante en el apartado *3.3.- Resultados obtenidos con el diseño en microcontrolador*.

3.2.4. - Almacenamiento mediante *FPGA*

Como se ha comentado anteriormente, la principal diferencia entre el microcontrolador y la *FPGA* a la hora de diseñar es que, mientras en el microcontrolador usamos el driver *SPI* proporcionado por la plataforma mbed, en la *FPGA* esta implementación se realiza de cero. En cuanto a funcionamiento la principal diferencia es que mientras el microcontrolador trabaja muestra a muestra, la *FPGA* puede adquirir y almacenar al mismo tiempo sin tener que detenerse cada vez que se almacena un bloque en la memoria *microSD*.

La implementación que se realiza sólo contemplará la rutina de inicialización y la rutina de escritura.

La máquina de estados posee los siguientes estados:

1. **Inicializar**: Deselecciona la tarjeta *SD* y con *MOSI* a *1* envía varios pulsos de reloj.

   Pone la dirección de escritura y el contador de pulsos de reloj a cero. Pasa al estado Deseleccionar y cambia la variable *DevolverEstado* a *Cmd_0*.

2. ***Cmd_0***: Pone la tarjeta en *Idle*.

   Activa el *CS* y envía el comando 0 a través del estado *Tx_bits*. Este estado también cambia el valor de la variable *DevolverEstado* a *Cmd_0_Respuesta*.

3. ***Cmd_0_Respuesta***: Comprueba el *token R1*.

   Comprueba que el *token R1* recibido es correcto y pasa al siguiente estado *Cmd_8*. En caso de no ser correcto se vuelve al estado *Cmd_0*.

4. ***Cmd_8***: Necesario para inicializar tarjetas *SDHC*.

   Activa el *CS* y envía el comando 8 a través del estado *Tx_bits*. Este estado también cambia el valor de la variable *DevolverEstado* a *Cmd_8_Respuesta*.

5. ***Cmd_8_Respuesta***: Comprueba el *token R7*.

   Activa el *CS* y recibe el *token R7* para saber si todo ha sido inicializado correctamente. Para esto vamos al estado Rx_bits. Por último pasamos al estado *Cmd_55*.

6. ***Cmd_55***: Envío del comando 55.

   Activa el *CS* y envía el comando 55 a través del estado *Tx_bits*. Este estado también cambia el valor de la variable *DevolverEstado* a *Cmd_41*.

7. ***Cmd_41***: Envía el comando de inicialización.

   Activa el *CS* y envía el comando 41 a través del estado *Tx_bits*. Este estado también cambia el valor de la variable *DevolverEstado* a A*cmd_41_Respuesta*.

8. ***Acmd_41_Respuesta***: Comprueba que se haya inicializado correctamente.

   Comprueba que el *token R1* recibido es correcto y pasa al siguiente estado *Espera*. En caso de no ser correcto se vuelve al estado *Cmd_55*.

9. ***Espera***: Espera a que se active la señal de escritura.

   En este estado se espera hasta que se activa la señal de escritura, en ese momento se pasa al estado *Escribir_bloque*.

10. ***Escribir_bloque***: Escribe un bloque de 512 *Bytes*.

Se activa el *CS* y se envía ocho bits a *1* pasando al estado *Tx_bits*, posteriormente enviamos el *start token* pasándolo también al estado *Tx_bits* y por último enviamos los datos y el *CRC* enviándolo al mismo estado. Una vez hayamos transmitido todo el estado *Tx_bits* volverá a *Escribir_bloque* y de aquí iremos al estado *Deseleccionar*.

11. ***Tx_bits***: Transmite el número de bits indicado a través de la salida *MOSI*.

    Si estamos en el flanco positivo del *reloj_spi* pasamos el *bit* más significativo del *buffer* y decrementamos tanto el *buffer* como

el contador *Cnt*. Una vez acabemos éste nos devolverá al estado que tengamos en la variable *DevolverEstado.*

12. ***Deseleccionar***: Desactiva *CS* y pone *MOSI* a *1*.

    Ponemos las salidas *cs_spi* y *MOSI* a *1*. Pasamos al siguiente estado *Pulsos_de_reloj*.

13. ***Rx_bits***: Recibe el número de bits indicado.

    Introduce el número de *bits* indicados de la entrada *MISO* al *buffer datos_rx*. Por último, vuelve al estado que tengamos en la variable *DevolverEstado*.

14. ***Pulsos_de_reloj***: Espera el número de ciclos de reloj que le indicamos.

    Cada flanco positivo de reloj resta 1 a la variable *Cnt* y una vez ésta es cero vuelve al estado que tengamos en la variable *DevolverEstado*.

Aunque antes se ha descrito de forma generalizada las rutinas de inicialización y escritura, a continuación se va a exponer con más detalle cómo funcionan estas rutinas en el código *Verilog HDL* implementado.

Como se puede comprobar en la figura siguiente hay pequeñas diferencias con respecto a la rutina de inicialización usada con el microcontrolador, ya que ésta es una inicialización más generalizada. En la implementación en *FPGA* se ha eliminado el comando 58. Este comando tan sólo se utiliza para comprobar que la tarjeta seleccionada soporta el voltaje de operación del procesador.

3.2.4.1. - Rutina de inicialización

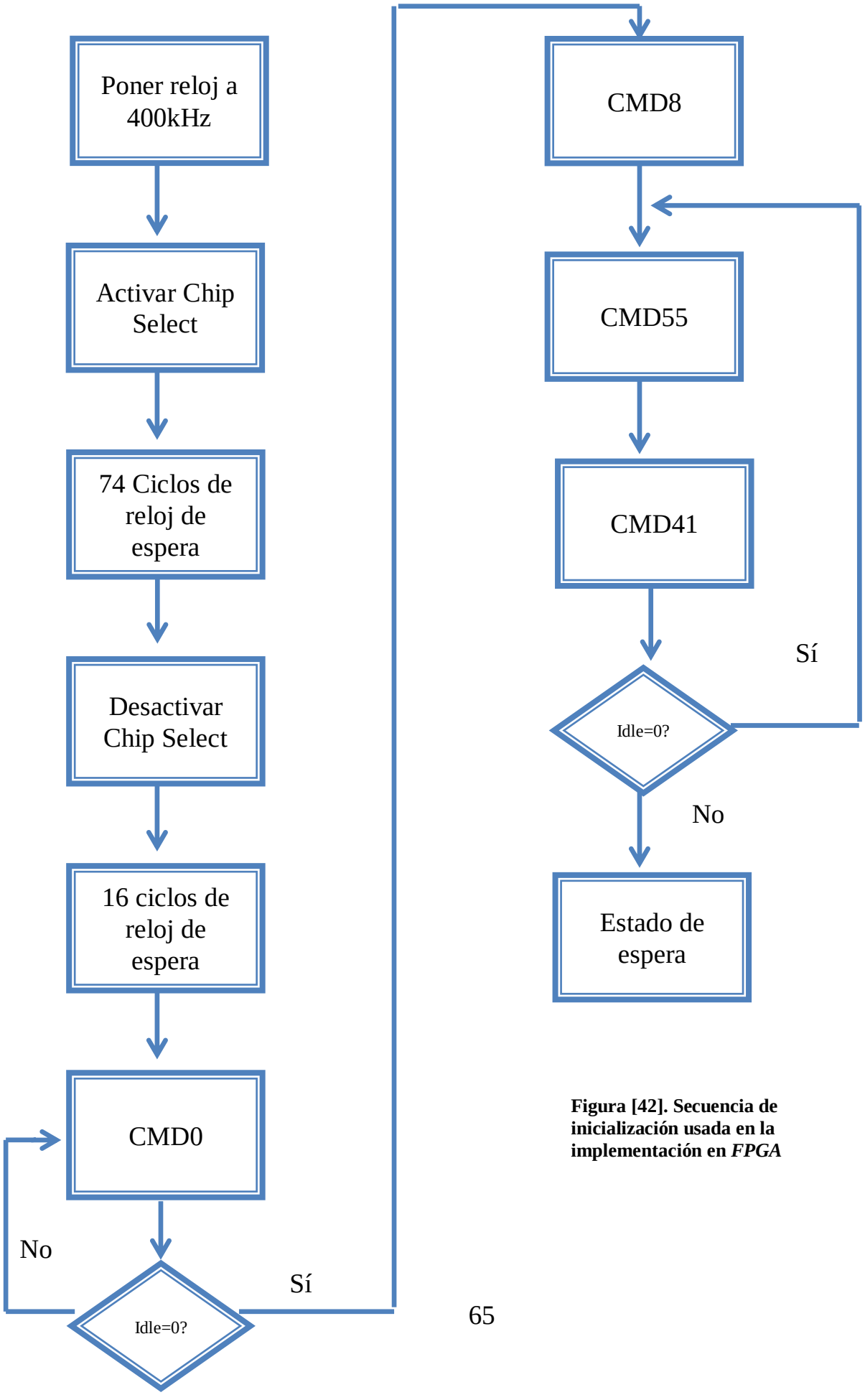


Figura [42]. Secuencia de inicialización usada en la implementación en *FPGA*

3.2.4.2.- Rutina de escritura

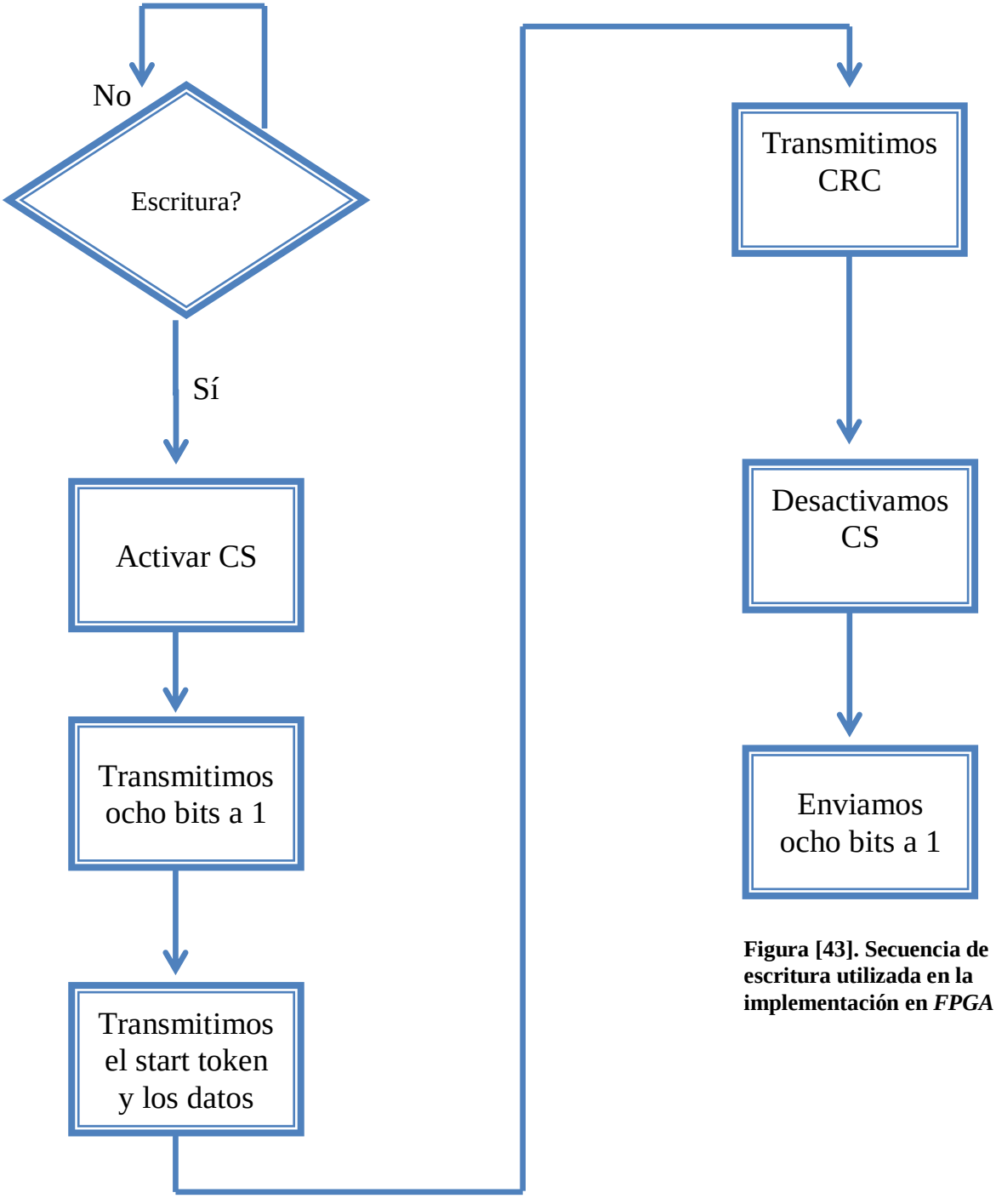


Figura [43]. Secuencia de escritura utilizada en la implementación en *FPGA*

Cómo se puede observar en la figura anterior, en este caso se ha prescindido del comando 24, al no tener implementada la opción de escribir uno o varios bloques en la *microSD*. Tan sólo escribimos uno. De esta forma lo que queremos conseguir es utilizar el menor número de elementos lógicos, ya que esto será importante en el diseño final.

3.3. - Resultados obtenidos con el diseño en microcontrolador

A continuación, se van a presentar los resultados obtenidos durante la experimentación. Los primeros experimentos se realizaron con el generador de funciones y se realizó un estudio en frecuencia. Posteriormente, se realizó un estudio con registros electrofisiológicos reales.

3.3.1. - Análisis frecuencial

Para poder realizar una simulación en la que se pueda comparar la señal original con la señal adquirida lo que se ideó fue el siguiente sistema:

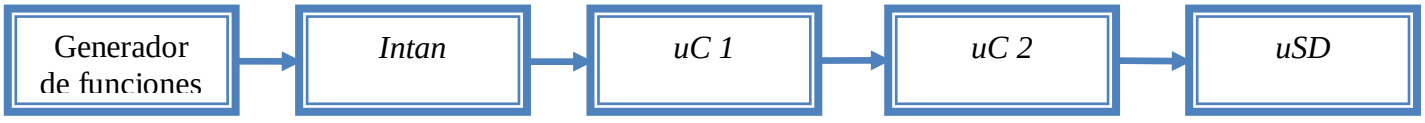


Figura [44]. Montaje realizado para poder hacer el estudio en frecuencia con microcontrolador

En este montaje lo que se quiere conseguir es almacenar por un lado la señal original que viene del generador y, por otro, la señal almacenada en la *microSD*.

El primer microcontrolador (*uC* 1) almacenará todas las muestras en un *buffer* que luego volcará en pantalla del ordenador, mientras que el segundo microcontrolador (*uC* 2) recibirá las muestras del primero y, si está ocupado, es decir, si está escribiendo en la *microSD*, descartará las muestras que reciba.

Para una entrada senoidal y la lectura de un solo canal muestreado a 20*kHz* obtenemos lo siguiente:

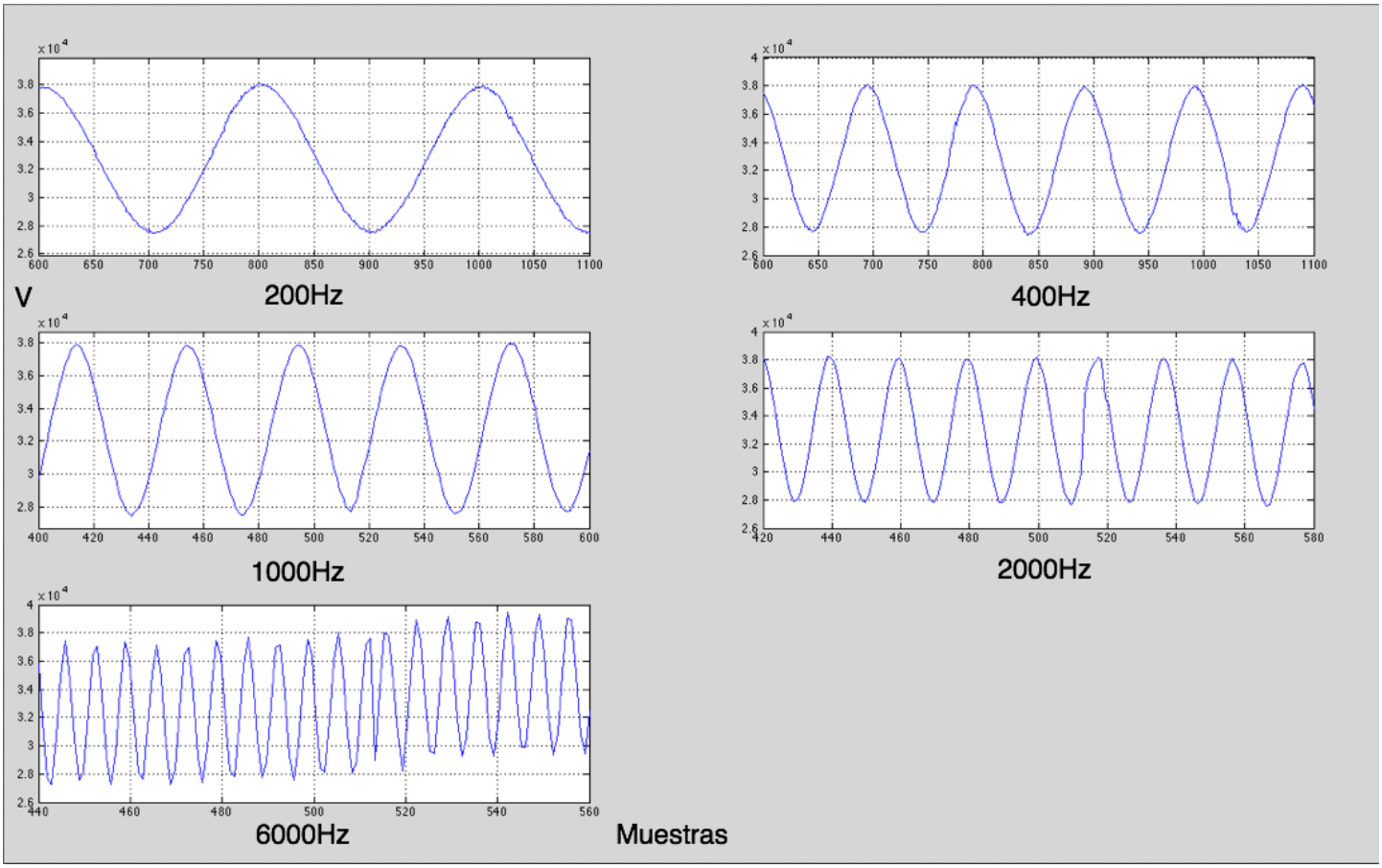


Figura [45]. Estudio en frecuencia de las señales obtenidas de un canal muestreadas a 20*kHz*

Para cuantificar lo que esto representa se ha utilizado el índice de simetría estructural (*SSIM*, por sus siglas en inglés). El *SSIM* se utiliza generalmente para medir la calidad de reconstrucción de algoritmos de compresión de imágenes en referencia a sus versiones originales pero también ha sido probado que es un indicador significativo para señales unidimensionales. Un *SSIM* mayor indica una mejor reconstrucción y cuando la señal reconstruida es exactamente igual a la original el *SSIM*=1.

Para que este índice nos dé una medida real hubo que insertar un valor correspondiente a la media entre los dos valores de alrededor cada 256 muestras de manera que la señal almacenada en la *microSD* no se vaya "adelantando" con respecto a la original, debido a esa muestra que se pierde cada 256 almacenadas, y el *SSIM* nos dé una medida más real.

Para un canal obtenemos unos valores de *SSIM* de:

- Para 200*Hz* e interpolando una estimación de los valores de alrededor: *SSIM*=0.999999999999932

- Para 1000*Hz* e interpolando una estimación de los valores de alrededor: *SSIM*=0.99999999928374
- Para 3000*Hz* e interpolando una estimación de los valores de alrededor: *SSIM*=0.999999988723842

Para ver esto de una forma más gráfica, se ha realizado una representación del *SSIM* en función de la frecuencia en la siguiente figura:

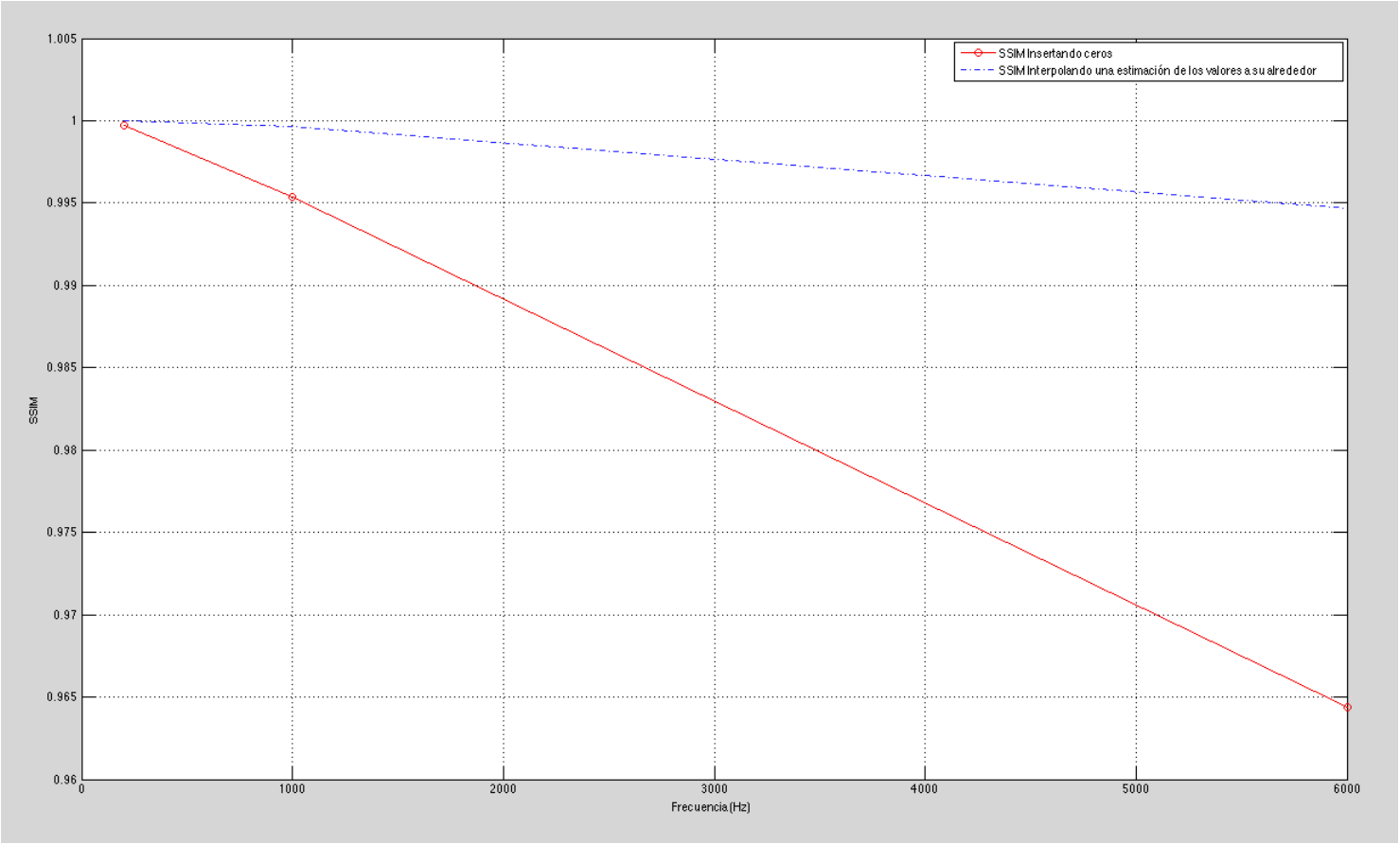


Figura [46]. *SSIM* en función de la frecuencia para un canal

En la figura 46 el eje de abscisas representa la frecuencia y el eje de ordenadas el *SSIM*. La línea azul discontinua es la interpolación realizando la media de los valores contiguos, mientras que la línea roja se refiere a la inserción de ceros.

Realizando el mismo estudio para los cuatro canales que se propusieron al principio de este trabajo obtenemos lo siguiente:

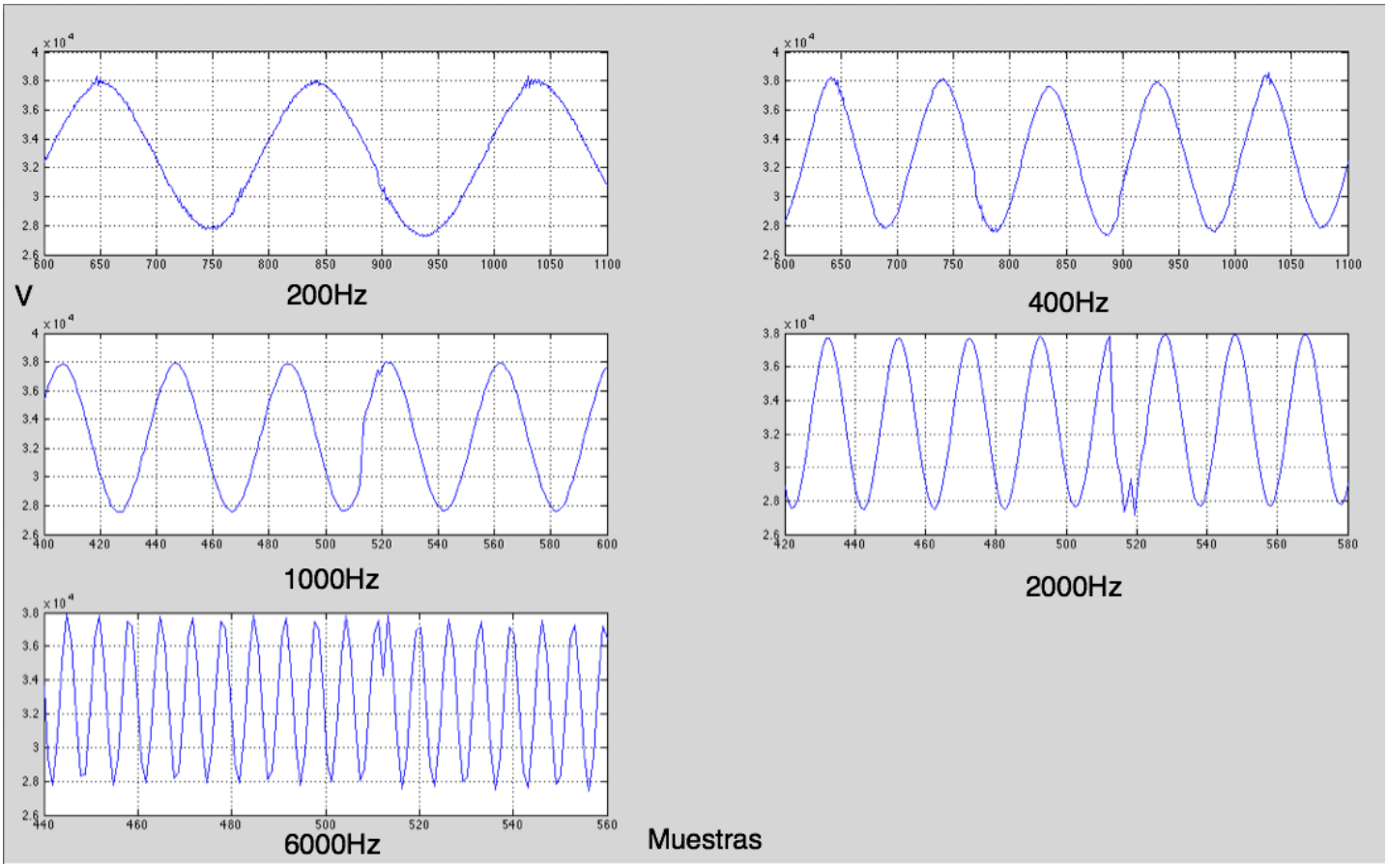


Figura [47]. Estudio en frecuencia de las señales obtenidas de cuatro canales muestreadas a 20*kHz*

Para cuatro canales las pérdidas son cuatro veces superior, es decir, perdemos 1 muestra cada 64 y, por lo tanto se realizará la inserción de un valor después de cada 64 muestras adquiridas.

Para cuatro canales obtenemos unos valores de *SSIM* de:

- Para 200*Hz* e interpolando una estimación de los valores de alrededor: *SSIM*=0.999973837283849
- Para 1000*Hz* e interpolando una estimación de los valores de alrededor: *SSIM*=0.999647223849492

- Para 6000*Hz* e interpolando una estimación de los valores de alrededor: *SSIM*=0.964738273898474

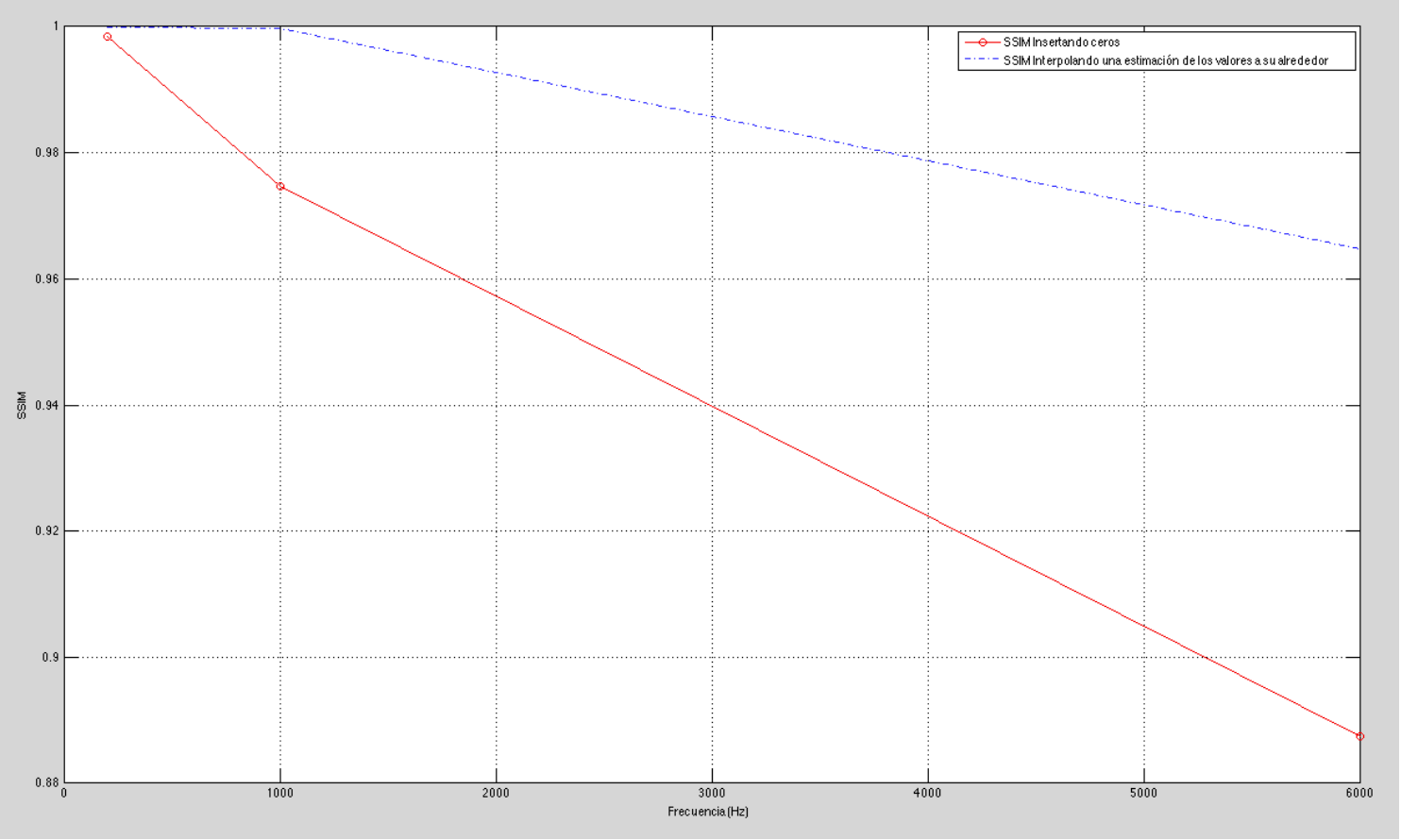


Figura [48]. *SSIM* en función de la frecuencia para 4 canales

A pesar de que este análisis en frecuencia nos puede dar una pequeña visión de lo que sucede, lo interesante es poder saber que representa esto con registros reales.

3.3.2. - Análisis con registros electrofisiológicos reales

Para realizar esta simulación el montaje realizado ha sido algo diferente al no poder pasar a través del *Intan* los registros reales proporcionados por el *Instituto de Neurociencias de Alicante*. En este caso lo que se ha hecho es enviar los registros a través de MATLAB. El montaje realizado ha sido el siguiente:

Figura [49]. Montaje realizado para poder hacer el estudio con registros reales

En este montaje MATLAB se encarga de simular la parte de adquisición y enviarla al microcontrolador cada 50 microsegundos (20*kHz*). De nuevo, en caso de estar ocupado el *uC* escribiendo en la memoria *microSD* se descartará la muestra (para un canal) o muestras (para cuatro canales).

Para realizar este análisis se ha tenido que realizar una simulación interpolando a 20*kHz* y enviando los datos desde el ordenador al *microcontrolador*, esto es así porque los registros con los que se trabaja ya están muestreados a 2.5*kHz*.

Los resultados que obtenemos para un canal son los siguientes:

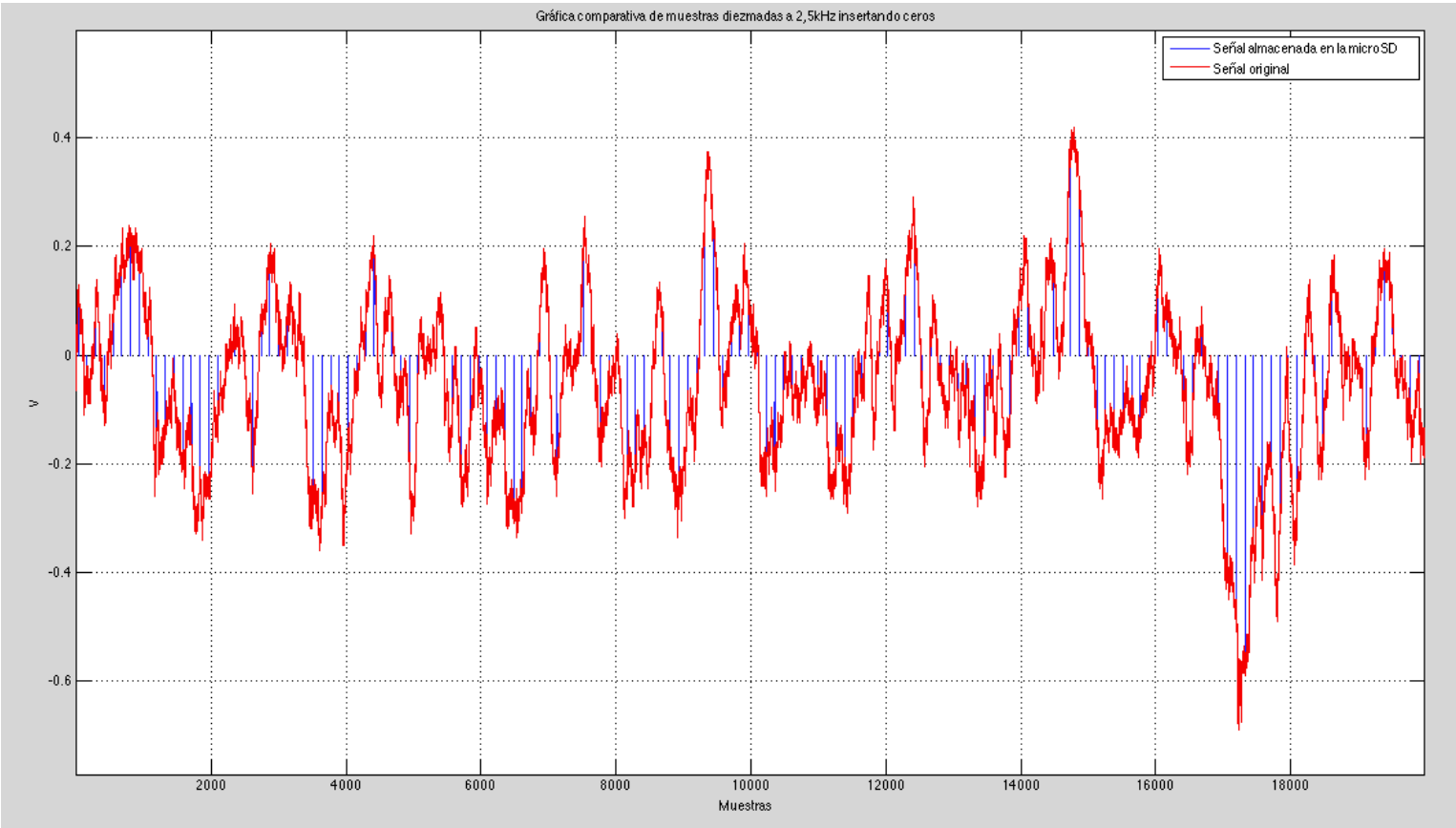


Figura [50]. Señal original frente a señal almacenada en la *microSD*

En la figura 50 la señal roja es la señal original y la señal azul representa la señal almacenada en la *microSD* habiendo insertado ceros en el lugar de las muestras que se pierden. Su *SSIM* es de 0.91465897519357.

Para un canal interpolando valores a partir de la media de los valores a su alrededor tenemos:

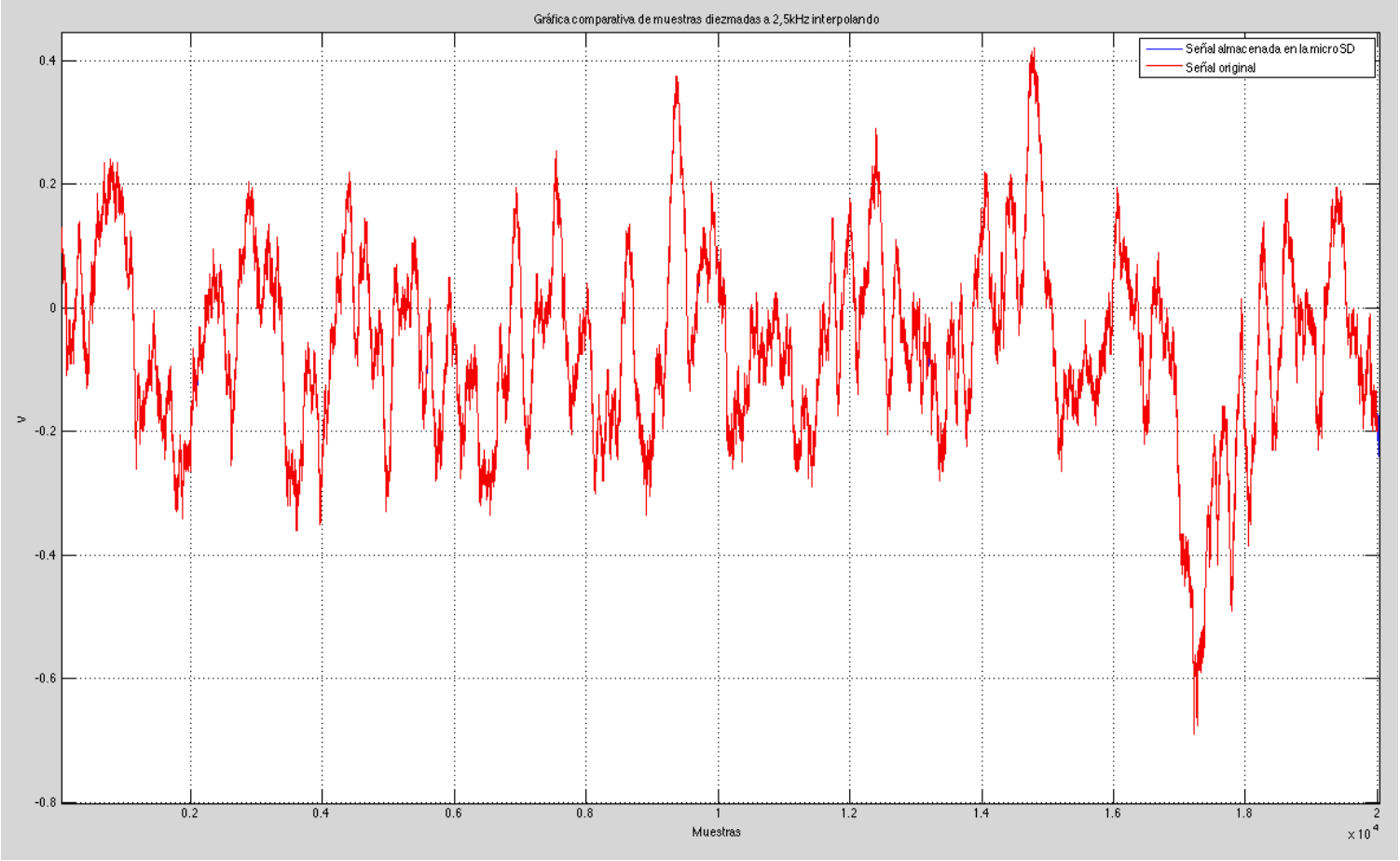


Figura [51]. Señal original frente a señal almacenada en la *microSD* para valores interpolados

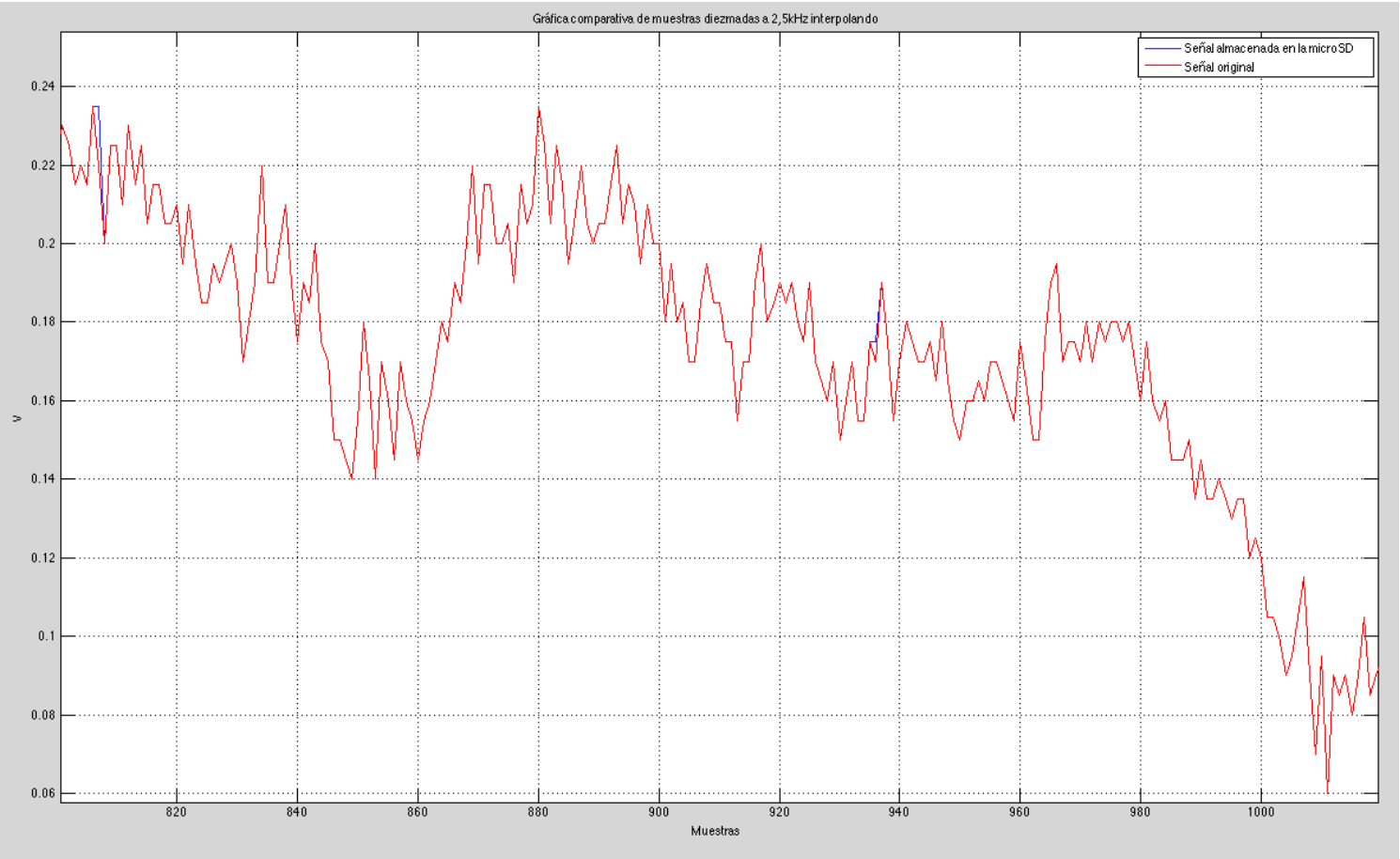


Figura [52]. Ampliación de la figura 51

En las figuras 51 y 52 podemos ver que interpolando valores se consigue una gran similitud entre las señales y su SSIM es de 0.999922614233569.

Para cuatro canales insertando ceros los resultados son los siguientes:

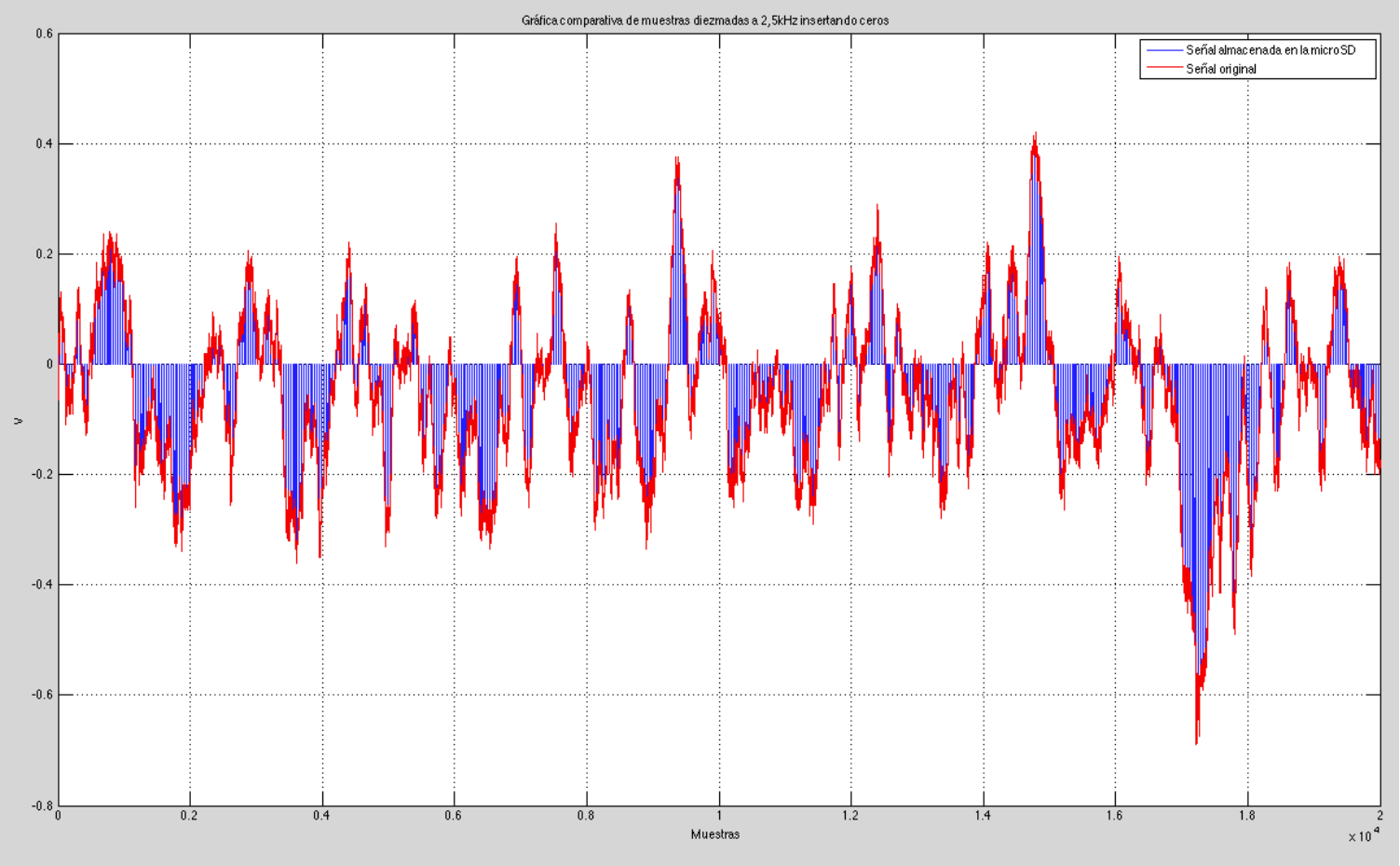


Figura [53]. Señal original frente a señal almacenada en la microSD para inserción de ceros y cuatro canales

De nuevo, siguiendo el mismo procedimiento pero para cuatro canales obtenemos la figura 53. En este caso el SSIM es de 0.889145896572315.

Para cuatro canales interpolando valores a partir de la media de los valores a su alrededor tenemos:

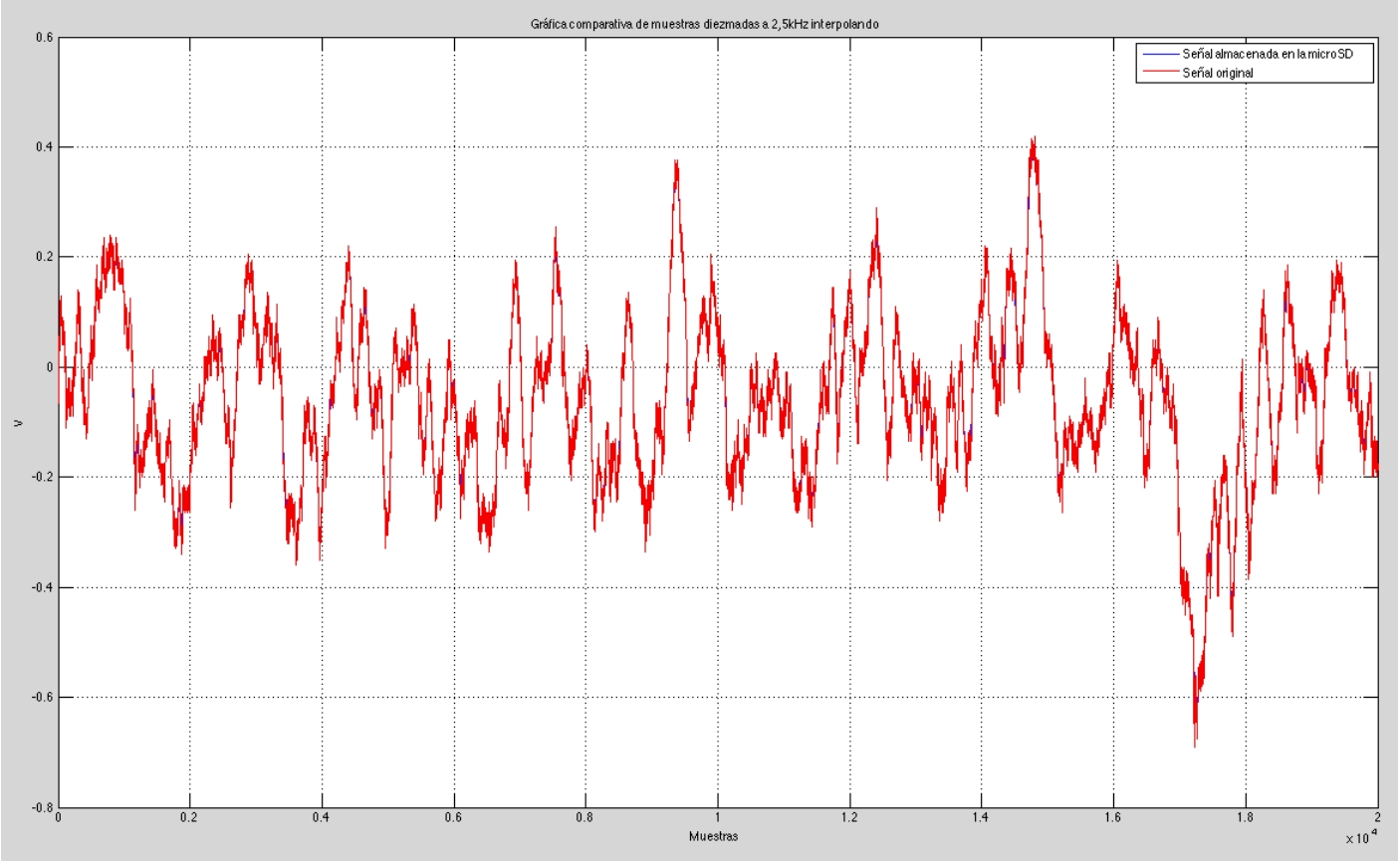


Figura [54]. Señal original frente a señal almacenada en la microSD para valores interpolados

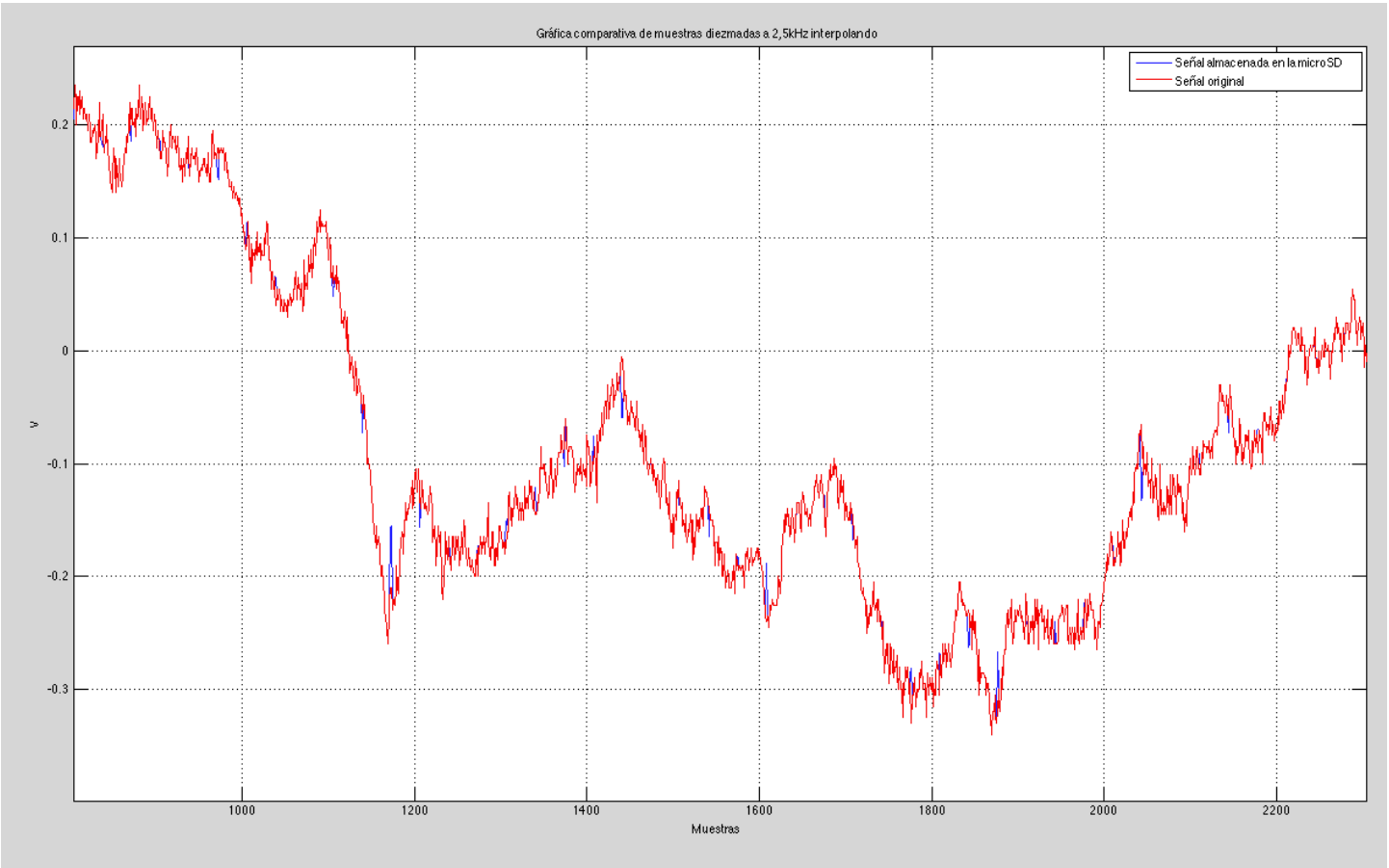


Figura [55]. Ampliación de la figura

En las figuras 54 y 55 podemos ver que interpolando valores se consigue, de nuevo, una gran similitud entre las señales y su *SSIM* es de 0.999423590627898.

3.4. - Resultados obtenidos con el diseño en *FPGA*

A continuación vamos a ver el mismo estudio para el diseño en *FPGA*. En primer lugar con el estudio en frecuencia y posteriormente con registros electrofisiológicos reales.

En este caso para ambos estudios se ha utilizado únicamente la *FPGA* e, internamente, se ha instanciado una *ROM* de manera que podamos simular la entrada de todo aquello que nos interesa (por un lado el estudio en frecuencia y por otro la entrada de registros electrofisiológicos reales). Internamente en la *FPGA* se ha realizado el siguiente diseño:

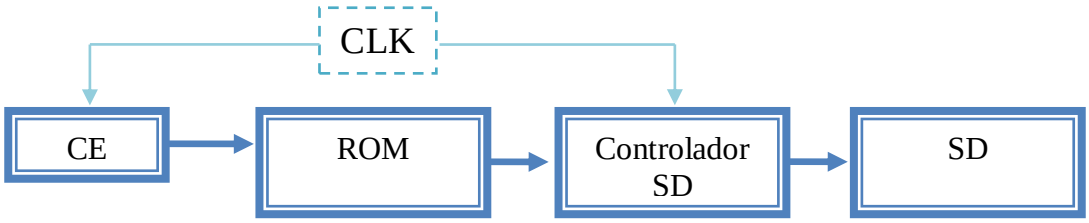


Figura [56]. Implementación interna *FPGA* para la simulación

3.4.1. - Análisis frecuencial

En este análisis frecuencial se ha pasado directamente a la implementación de 4 canales, ya que no se ha implementado un diseño para un solo canal.

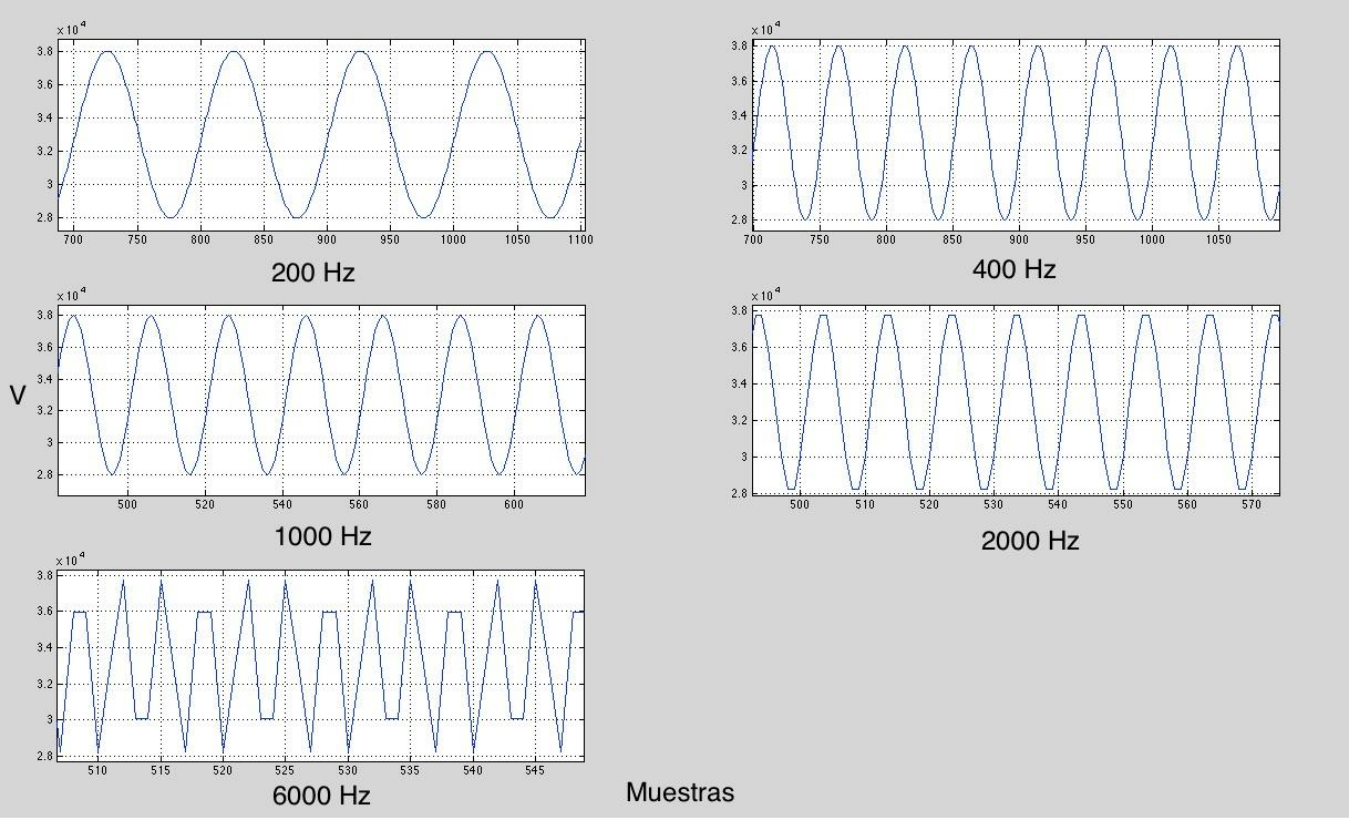


Figura [57]. Muestras obtenidas con la *FPGA* para el análisis frecuencial

En este análisis se observa que no hay pérdida alguna. Para ello se ha comprobado, de nuevo, utilizando el *SSIM* que, como era de esperar es de 1.

3.4.2. - Análisis con registros electrofisiológicos reales

Como en el caso anterior, sólo se ha tenido en cuenta el estudio para cuatro canales.

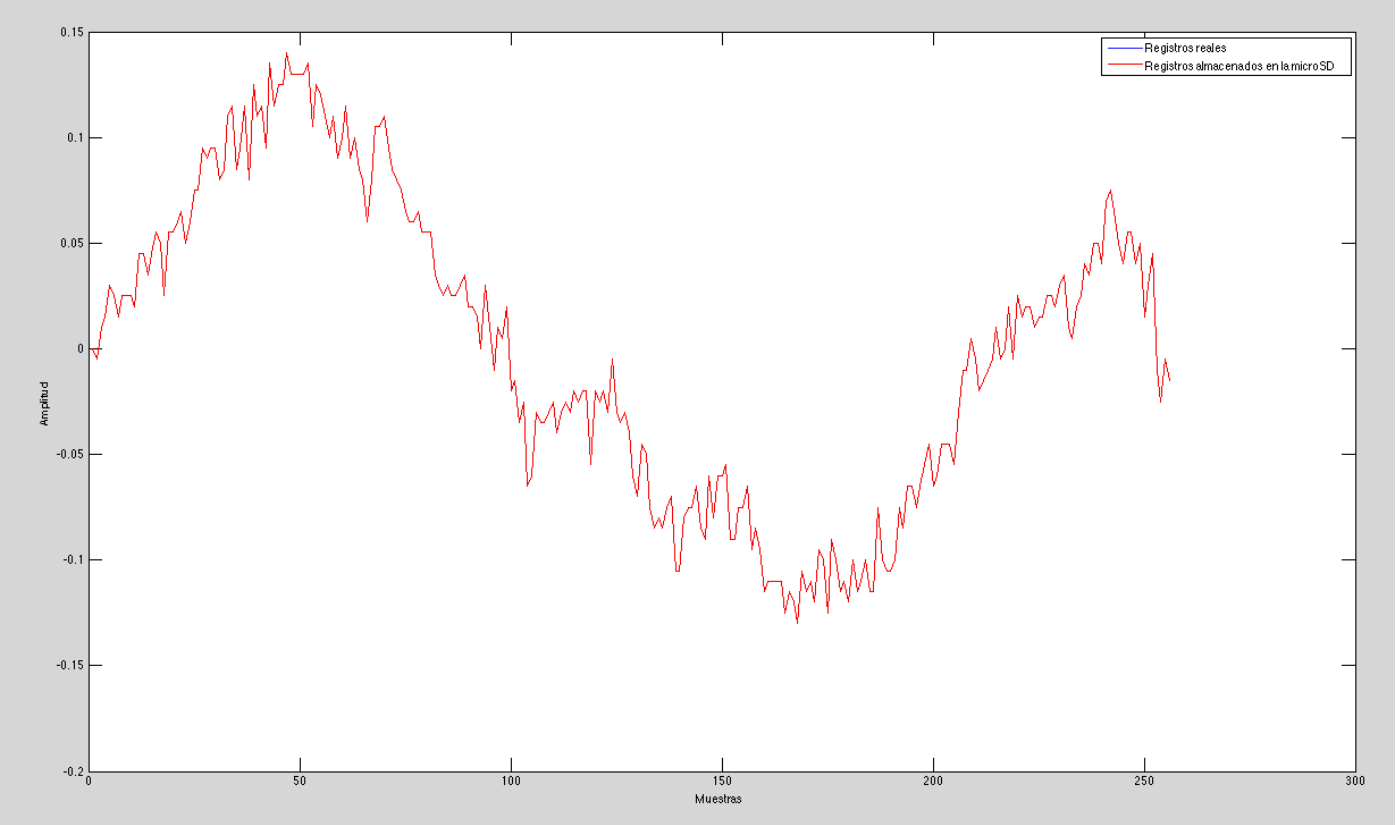


Figura [58]. Análisis con registros electrofisiológicos reales

Como se puede observar, las señales están totalmente superpuestas y si volvemos a utilizar el *SSIM* como referencia, obtenemos otra vez un valor de 1. Estos resultados hablan de la utilidad de una *FPGA* cuando trabajamos con frecuencias "altas".

3.5. - Diseño final

Para presentar el diseño final se va a exponer el conjunto de todos los componentes seleccionados, el consumo total del sistema y su coste.

3.5.1. - Sistema

En este apartado se va a hablar de las partes del diseño que han sido seleccionadas y no de las partes que ya venían fijadas anteriormente por el *Instituto de Neurociencias de Alicante* (Intan RHD, cable SPI del propio Intan…) que sólo se tendrán en cuenta para realizar una estimación de los costes del sistema completo.

3.5.1.1. - Memoria

La memoria elegida ha sido la *Lexar 16GB*, esta memoria es una de las de menor consumo del mercado y además es muy barata, como veremos en apartados posteriores.

Figura [59]. Memoria *microSD* elegida

3.5.1.2. - *FPGA*

La *FPGA* elegida ha sido la gama *IGLOO Nano* de *microSemi*.

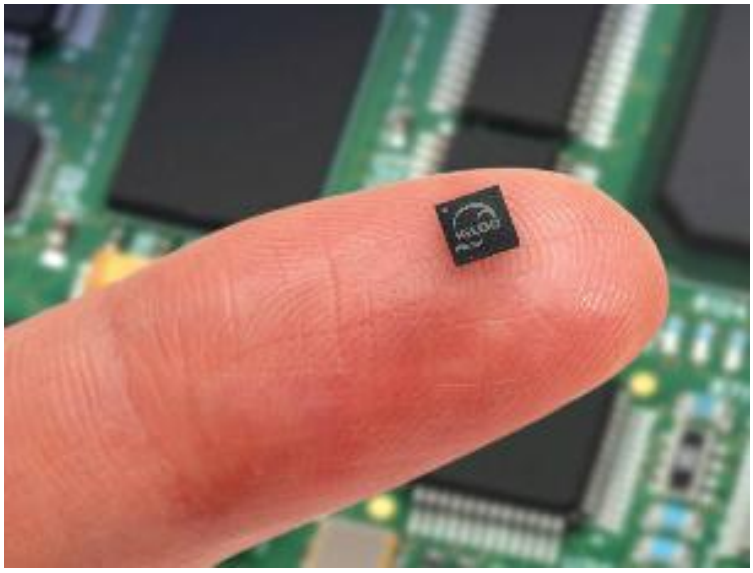


Figura [60]. *FPGA* de la serie *IGLOO Nano*

Como se aprecia en la imagen su reducido tamaño y su ultra bajo consumo hace que se ajusten a la perfección al propósito de este trabajo.

Para esta elección se ha utilizado el *Low Power Calculator* disponible en la *web* de *microSemi* para seleccionar la *FPGA* de menor consumo que nos permita alcanzar las prestaciones que queremos.[15]

Dentro de la gama *IGLOO Nano* de *microSemi*, la de menor consumo que mejor se ajusta es la *AGLN60*.

3.5.1.3. - Batería

Teniendo en cuenta que la batería ha de tener el menor tamaño posible, se propone utilizar una pila de botón, en concreto el modelo *Renata CR2477*. Esta pila posee una capacidad de 950 *mAh*, una altura de 8.5 *mm* y una anchura de 24.7 *mm*. Su capacidad es más que suficiente como para varias horas de experimento.

El tamaño puede parecer grande, pero es un tamaño muy aceptable comparado con el tamaño del murino.

Para calcular el tamaño de esta batería se ha realizado el siguiente estudio:

3.5.2. - Consumo

En este apartado se va a presentar la medición de consumos hecha y, en el caso de la *FPGA*, la correspondiente a su *datasheet*.

3.5.2.1. - Consumo *microSD*

Para realizar un estudio del consumo se ha dispuesto de diferentes *microSD* y se ha medido la corriente de entrada para saber cuánta corriente requiere una escritura en cada una de ellas.

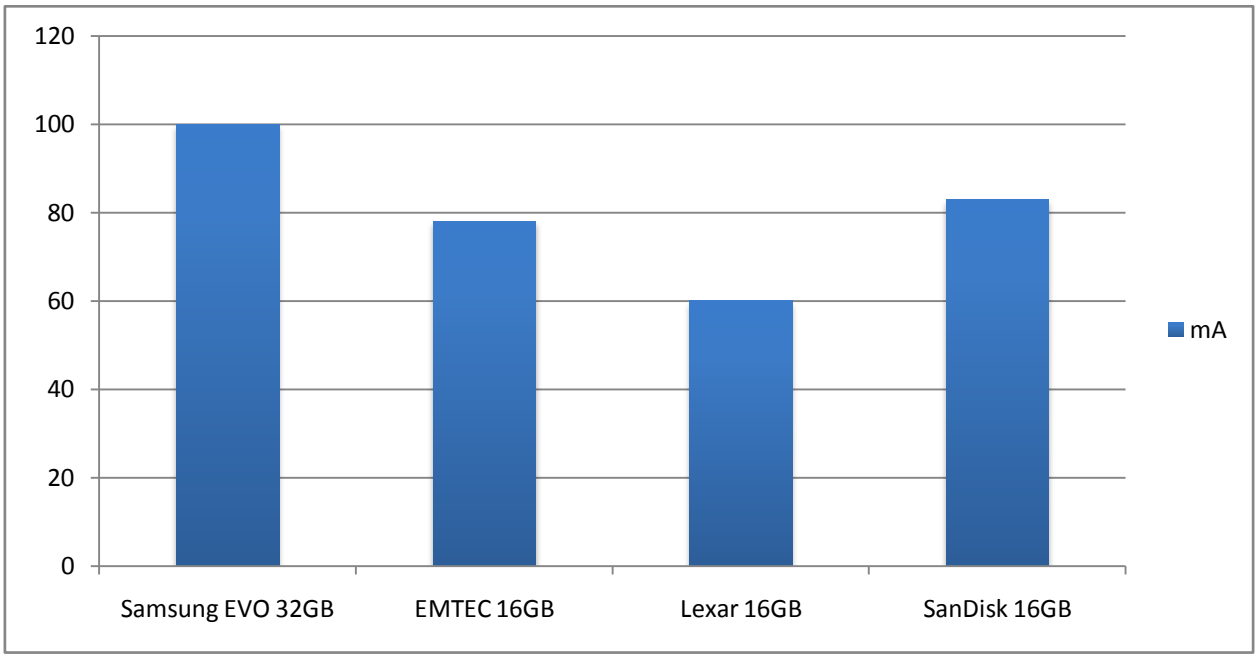


Figura [61]. Comparación de consumos de diferentes memorias *microSD*

Estos consumos son significativos, incluso equiparables al consumo de un microprocesador o una *FPGA* de bajo consumo, aún así, existen pilas de botón de hasta 1000*mAh* (Esto quiere decir que puede suministrar 1000*mA* durante una hora).

Como podemos observar la *Lexar* es la que menor consumo tiene de lejos, aunque también es más lenta. Al haber cambiado de diseño y estar

trabajando con una *FPGA*, ahora lo que nos interesa es que tenga un menor consumo, ya que hemos evitado la problemática temporal.

3.5.2.2. - Consumo FPGA

Mirando en las especificaciones de la *FPGA* elegida y a partir del *Low Power Calculator* de *microSemi,* obtenemos un consumo de entorno a los 10*mAh.*

3.5.2.3. - Consumo Total

A los consumos de los elementos seleccionados para nuestro diseño debemos sumarle el consumo del *Intan RHD2000,* si observamos el datasheet vemos que el consumo típico está en torno a los 18*mAh.*

El **consumo total** de nuestro sistema estará en torno a los **88 *mAh***.

3.5.3. - Coste

El coste total del sistema es el siguiente:

Elemento	Precio (euros)
Intan RHD 2132	890,81
Cable Interfaz SPI	192,49
IGLOO Nano	11,50
microSD Lexar 16GB	14,39
Batería	6,83
Total	**1116,02**

Tabla [7]. Tabla de costes del sistema

A este coste habría que sumarle los gastos de envío correspondiente y los costes de la realización del diseño de manera que esté listo para poder realizar el experimento correspondiente.

Los precios del *Intan RHD2132* se han sacado del apartado de pedidos de la *web* de *Intan Technologies* [16]. Los precios de la *Igloo Nano* y batería se han mirado en las *webs* de *mouser* [17] y *farnell* [18], mientras que la tarjeta *microSD* se ha mirado en *Amazon*[19].

4. - Conclusiones

Reasoning draws a conclusion, but does not make the conclusion certain, unless the mind discovers it by the path of experience.

Roger Bacon

Durante la duración de este proyecto se ha realizado el estudio y prototipado de un sistema embebido portátil, muy pequeño y de ultra bajo consumo.

Para ello se ha seleccionado el sistema óptimo para poder adquirir las señales a una frecuencia de muestreo de 20*kHz* y almacenarlas en una memoria lo suficientemente grande como para poder realizar experimentos de varias horas.

Para el almacenamiento de señales se eligió, tras un estudio previo, una memoria *microSD*, ya que es la única que cumplía las especificaciones necesarias: Gran capacidad de almacenaje y un consumo asumible, aunque mayor del que nos gustaría.

Se ha realizado un diseño hardware para un sistema portátil y de bajo consumo y como era de esperar se han marcado dos puntos clave para su consecución: el consumo y la frecuencia de adquisición.

El punto más limitante ha sido la frecuencia de adquisición, 20*kHz*. En un primer momento se optó por utilizar un microcontrolador por su rápido y sencillo uso. Tras una primera implementación nos encontramos con la problemática que caracteriza a estos sistemas: trabajan muestra a muestra. Esto quiere decir que entre cada adquisición hemos de ser capaces de realizar todo el tratamiento que necesitemos.

Los resultados para cuatro canales eran buenos, pero hemos de recordar que este proyecto no es más que un paso intermedio hacia algo más grande y, en vez de buscar una solución fácil (por ejemplo, un microcontrolador que permita hacer funcionar el puerto *SPI* a más frecuencia, lo que para este PFC valdría, pero en el momento que queramos adquirir más canales tendremos que buscar otra solución), se ha querido

buscar una solución que abra el camino para el diseño final que se propone este grupo de investigación.

El siguiente diseño propuesto fue un diseño *hardware* de más bajo nivel con una *FPGA* para evitar la problemática de la frecuencia de muestreo y poder trabajar en paralelo. Con este diseño se obtuvieron los resultados esperados: No hay pérdida de muestras. Esta solución le sirve también al grupo como punto de partida para poder continuar hacia el diseño final antes descrito debido a su flexibilidad para incorporar nuevos elementos al diseño.

En el consumo, como se ha visto, el limitante es la memoria. Las memorias con más capacidad consumen mucho y las que consumen poco no cumple nuestro criterio de un mínimo de 16 *GB*. Por estos motivos se eligió la tarjeta *microSD* que, a pesar de tener un consumo elevado, es algo asumible para la finalidad de este proyecto.

La batería elegida para alimentar a todo el sistema es una pila de botón de 950 *mAh*. Capaz de poder alimentar el sistema durante varias horas. En un futuro se pretende recargar inalámbricamente esta batería de manera que el experimento pueda durar más.

En este *PFC* se ha podido ver la eterna lucha *FPGA vs microcontrolador* y cuándo es mejor la elección de uno u otro. Esta decisión dependerá de las operaciones que tengamos que realizar entre la adquisición de dos muestras. Actualmente las *FPGA's* se utilizan en sistemas de tratamiento de señal a muy alta frecuencia y sistemas que necesitan realizar funciones en paralelo. El diseño *Hardware* es bastante más complejo que el *software* y también requiere un cambio en la mentalidad del diseñador, pues estamos mucho más acostumbrados al pensamiento desde el punto de vista del *software*. Al realizar una implementación en *FPGA* debemos dejar eso a un lado y empezar a pensar en una interconexión hardware mediante flip-flops, puertas lógicas, sumadores, multiplicadores… Este es uno de los puntos clave por los que se suele preferir usar un microcontrolador o DSP.

5. - Líneas futuras

Para entender las líneas futuras que se proponen a continuación hay que tener en mente el contexto, ya explicado, en el que se enmarca este *PFC*. Este contexto se ha explicado con anterioridad y se encuentra dentro del *T-RAT Lab*.

Para conseguir los objetivos que el grupo se propone, se sugiere continuar con el diseño en *FPGA* y la siguiente hoja de ruta:

5.1. - Transmisión de señales en tiempo real

A partir de la adquisición de los cuatro canales de este *PFC* se propone que el siguiente paso sea la transmisión de estos antes de adquirir más canales. Para ello se utilizará el algoritmo de compresión, diseñado por *Yoel Wigoda* para este grupo en su *TFM*, para reducir la cantidad de datos a transmitir. La tecnología a utilizar será *bluetooth*, a falta de un estudio sobre esto que nos pueda indicar que haya otra tecnología mejor.

5.2. - Recarga de baterías

Es de esperar que al añadir la transmisión de señales en tiempo real el consumo aumente sustancialmente y, a la vista de esto, se recomienda hacer un buen estudio de consumo que nos indique si es posible realizar experimentos de varias horas aumentando la capacidad de la batería o, por el contrario, tendremos que realizar un sistema de recarga de baterías sin tener que parar el experimento. En caso de de poder realizar experimentos con un simple aumento de batería se recomienda dejar este punto para más adelante, ya que dejará de ser un tema prioritario.

Actualmente existen tecnologías de recarga inalámbrica en las que por inducción se consigue recargar dispositivos móviles incluso a varios metros de distancia. En un principio, recomendaría crear una *"estación de recarga"* en la que colocar al murino unos minutos, pero esto podría cambiar si el estudio previo demuestra que se podría recargar con el animal en movimiento y a varios metros de distancia.

5.3. - Aumento del número de canales

El siguiente paso en mi hoja de ruta es aumentar el número de canales. Una vez comprobado que somos capaces de transmitir un número de canales determinado pasaríamos a transmitir más canales. Se va a tener que transmitir una gran cantidad de datos y se tendrá que comprobar si somos lo suficientemente rápidos para transmitirlos sin pérdidas.

5.4. - Aumento de la frecuencia de muestreo

Como último punto de mi hoja de ruta estaría el aumento de la frecuencia de muestreo. En los próximos años se espera seguir estudiando las señales *LFP* que están en baja frecuencia 1-200 *Hz* y, por lo tanto, con 20*kHz* tenemos de sobra, pero en un futuro sí que se espera estudiar señales que pueden llegar incluso a unos 8*kHz* y para tener una muy buena resolución se podría subir la frecuencia de muestreo a unos 30*kHz* como quieren en el *Instituto de Neurociencias de Alicante*.

6. - Bibliografía

[1]. Cambra Enguix, J. Directores: David Moratal Pérez, Santiago Canals Gamoneda. "Estudio teórico y de viabilidad de un sistema hardware de posisionamiento y seguimiento de murinos y de transmición inalámbrica de sus señales de electrofisiología cerebral". Proyecto Final de Carrera en la E.T.S.I. de Telecomunicación, Universitat Politècnica de València, 2012.

[2]. Gabaldón López, V. Directores: David Moratal Pérez, Santiago Canals Gamoneda. "Estudio de la estructura dinámica de las interacciones funcionales en el hipocampo mediante procesado de registros electrofisiológicos cerebrales en rata". Tésis Doctoral, Universitat Politècnica de València, 2013.

[3]. Gabaldón López, V. Directores: David Moratal Pérez, Santiago Canals Gamoneda. "Estudio de la memoria y del aprendizaje a partir del análisis de la correlación y la coherencia de las señales de LFP en el hipocampo de rata adquiridas mediante registros electrofisiológicos de alta densidad". Proyecto Final de Carrera en la E.T.S.I. de Telecomunicación, Universitat Politècnica de València, 2010.

[4]. Página *web* oficial de *mbed*: http://mbed.org [Internet]. *ARM mbed*. [Último acceso 7/4/2015]. Información disponible en el apartado de *"About mbed"*: http://mbed.org/about/

[5]. Página *web* oficial de NXP: http://www.nxp.com [Internet]. NXP. [Último acceso 7/4/2015]. *Datasheet* del microcontrolador *NXP LPC1768*: http://www.nxp.com/documents/data_sheet/LPC1768_66_65_64.pdf

[6]. Página *web* oficial de *Altera*: https://www.altera.com [*Internet*]. *Altera*. [Último acceso 7/4/2015]. Página oficial del software quartus II: https://www.altera.com/products/software/quartus-ii/web-edition/qts-we-index.html

[7]. Página *web* oficial de: http://www.terasic.com.tw/en/ [*Internet*]. *Altera*. [Último acceso 7/4/2015]. Información disponible en: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=502

[8]. Página *web* oficial de: http://www.intantech.com [*Internet*]. Intan technologies. [Último acceso 7/4/2015]. Información disponible en: http://www.intantech.com/files/Intan_RHD2000_eval_system.pdf

[9]. Página *web* oficial de: https://www.sdcard.org [*Internet*]. *SD Association*. [Último acceso 7/4/2015]. Información disponible en: https://www.sdcard.org/developers/index.html

[10]. Página *web* oficial de: http://es.mathworks.com/index.html?s_tid=gn_logo [*Internet*]. *MathWorks*. [Último acceso 7/4/2015]. Información disponible en: http://es.mathworks.com/products/matlab/

[11]. Página *web* oficial de: http://klusta-team.github.io [*Internet*]. *Klusta-Team*. [Último acceso 7/4/2015]. Información disponible en: https://github.com/klusta-team/klustaviewa/

[12]. Cuevas López, A. Directores: David Moratal Pérez, Santiago Canals Gamoneda. "Sistemas de adquisición de bajo coste de registros electrofisiológicos en murino". Proyecto Final de Carrera en la E.T.S.I. de Telecomunicación, Universitat Politècnica de València, 2013.

[13]. Página *web* oficial de: http://www.intantech.com [*Internet*]. *Intan technologies*. [Último acceso 7/4/2015]. Información disponible en: http://www.intantech.com/files/Intan_RHD2000_USB_FPGA_interface.pdf

[14]. Página *web* oficial de: http://www.ntfs.com/fat-systems.htm [Internet]. *NTFS*. [Último acceso 7/4/2015]. Información disponible en: http://www.ntfs.com/fat-systems.htm

[15]. Página *web* oficial de: http://www.microsemi.com/products/fpga-soc/fpga/igloo-nano [*Internet*]. [Último acceso 23/6/2015]. Información disponible en: http://www.microsemi.com/products/fpga-soc/fpga/igloo-nano

[16]. Página *web* oficial de: http://www.intantech.com [*Internet*]. [Último acceso 23/6/2015]. Información disponible en: http://www.intantech.com/pricing.html

[17]. Página *web* oficial de: http://www.mouser.es [*Internet*]. [Último acceso 23/6/2015]. Información disponible en: http://www.mouser.es/ProductDetail/Microsemi/AGLN060V5-VQG100/?qs=sGAEpiMZZMvoScKlWpK8TNeNT1Y5lux5vvgtPJoRVwg%3d

[18]. Página *web* oficial de: http://www.farnell.com [*Internet*]. [Último acceso 23/6/2015]. Información disponible en: http://es.farnell.com/renata/cr2477-nfh-lf/pila-litio-boton-2-pines-3v/dp/1319732

[19]. Página *web* oficial de: http://www.amazon.es [*Internet*]. [Último acceso 23/6/2015]. Información disponible en: http://www.amazon.es/dp/B00I8NBXOS/ref=asc_df_B00I8NBXOS27152368/?tag=googshopes-21&creative=24538&creativeASIN=B00I8NBXOS&linkCode=df0