Universidad Zaragoza
1542

Trabajo Fin de Máster

Caracterización y mejora de un sistema de adquisición de señales electrofisiológicas basado en FPGA

Characterization and improvement of an FPGA-based electrophysiological signal acquisition system

Autor

Antonio Castel Santolaria

Directores

Antonio Velarte Iranzo

Isidro Urriza Parroqué

Máster Universitario en Ingeniería Electrónica

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2024

Este trabajo ha sido parcialmente financiado por el Programa de Becas y Ayudas del Instituto de Investigación en Ingeniería de Aragón (I3A).

Agradecimientos

Gracias a Antonio y a Isidro por su paciencia, orientación, implicación y por compartir sus conocimientos a lo largo de este trabajo.

Caracterización y mejora de un sistema de adquisición de señales electrofisiológicas basado en FPGA

Resumen

La captación y análisis de señales electrofisiológicas provenientes de tejidos cardíacos tiene una gran relevancia tanto en la detección de enfermedades cardiovasculares como en estudios farmacológicos. A pesar de los avances realizados, la obtención de las características más relevantes de estas señales sigue constituyendo un desafío significativo debido al alto coste o la dificultad de uso de los sistemas existentes. Por tanto, el objetivo de este trabajo es generar una plataforma de adquisición accesible, fácil de utilizar y adaptable a los requisitos de la investigación.

Como punto inicial del proyecto se utiliza un sistema de adquisición electrofisiológico sobre el que se realizan modificaciones y se añaden nuevos bloques funcionales. Las mejoras se centran en la captura, el manejo de los datos capturados y en su procesado. Para facilitar el uso por parte de los usuarios se proponen dos interfaces gráficas, una para controlar la plataforma y otra para realizar el procesado de los datos capturados.

Finalmente, se realiza una validación de la plataforma en la que se comparan las señales obtenidas con la plataforma propuesta frente a otros dos sistemas comerciales. Se comprueba que la funcionalidad de ambas interfaces es adecuada, registrando y procesando señales generadas de manera artificial. Este Trabajo Fin de Máster presenta una plataforma que solventa las dificultades detectadas en el estado del arte manteniendo la filosofía *open source*.

Índice general

1. Introducción

La captación y el análisis de señales electrofisiológicas provenientes de tejidos cardiacos suponen un desafío significativo en el ámbito de la investigación. La naturaleza dinámica de estas señales y las variaciones en la respuesta eléctrica de los tejidos presentan una complejidad inherente que requiere de un tratamiento meticuloso de la señal [1]. Para poder estudiar su comportamiento, es necesario un procesado preciso de la señal que permita eliminar los artefactos y señales espurias, así como extraer con exactitud los parámetros relevantes de la señal.

La adquisición de estas señales y su posterior procesado deben realizarse utilizando herramientas que se adapten a las necesidades de los experimentos realizados. Lo que muchas veces se traduce en herramientas de difícil acceso para el usuario, existiendo dos grandes barreras: el precio de los equipos y los conocimientos previos para su utilización. Con este proyecto se busca superar la segunda barrera, implementando una serie de mejoras en una plataforma de adquisición de señales que faciliten su uso por parte del usuario.

1.1. Alcance y Objetivos

El objetivo del trabajo es conseguir una plataforma de adquisición de señales electrofisiológicas, provenientes de tejido cardiaco, funcional y con un flujo de trabajo sencillo. Partiendo de la versión inicial de la plataforma, se valoran las limitaciones de la misma y una vez establecidas, se implementan una serie de mejoras en la captura de los datos, transmisión de los datos y control de la plataforma.

Se diseñan dos interfaces gráficas con el objetivo de facilitar el control de la plataforma, permitiendo la configuración y la captura de las señales desde un entorno amigable, y el procesado de los datos, para extraer y analizar las características de la señal de una forma rápida.

Finalmente se realiza una validación del funcionamiento de la plataforma, por medio de la comparación con dos equipos que forman parte del estado del arte, tomando medidas sobre una señal de prueba. Para la adquisición de la señal es necesaria la creación de un entorno de pruebas y la extracción de las características de la señal.

1.2. Cronograma

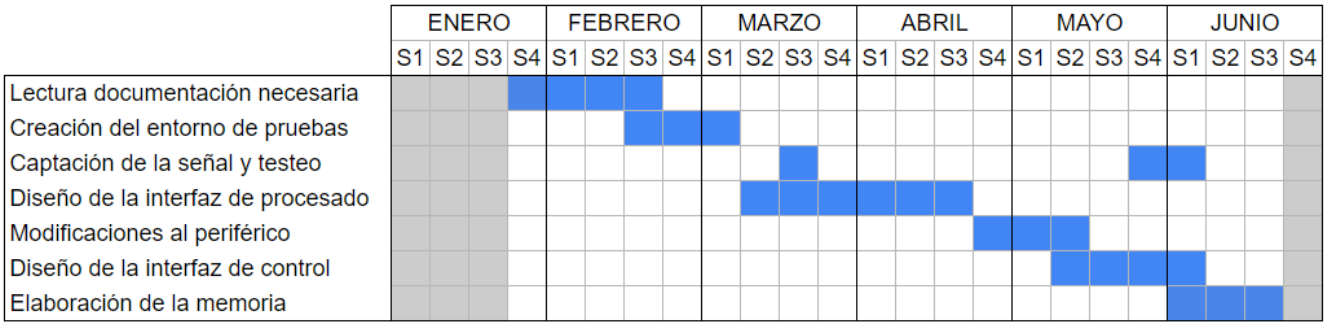


Figura 1.1: Cronograma del proyecto

1.3. Organización del documento

El presente documento se divide en 6 apartados, comenzando por una breve introducción a la memoria. En el capítulo 2 se ponen en contexto los conceptos básicos sobre la señal a capturar, así como las herramientas utilizadas a lo largo del proyecto. Los dos apartados siguientes detallan el diseño de las interfaces implementadas para facilitar el trabajo del usuario, separando la interfaz de procesado de la interfaz de control del periférico. El capítulo 5 se centra en la validación de la plataforma así como el entorno de pruebas implementado para poder realizar la adquisición de las señales. Finalmente, en el capítulo 6, se incluyen las conclusiones del proyecto y las posibles lineas futuras de este.

1.4. Herramientas utilizadas

Para la realización del trabajo, se han utilizado las siguientes plataformas:

- *Vivado 2019.2*: Software de síntesis y análisis de diseños de descripción de hardware (HDL) preparado para el desarrollo de sistemas en FPGAs.
- *Keil uVision 5*: Software para el desarrollo de código C y C++ preparado para el desarrollo en sistemas embebidos, permitiendo compilar el programa y depurar el código.
- *Matlab R2022a*: Plataforma de programación y cálculo numérico.

- *Matlab App Designer*: Software integrado en Matlab orientado a la creación de interfaces gráficas de usuario.
- *Altium Designer*: Software para el diseño de placas de circuito impreso (PCB).
- *FreeCad*: Software libre para el diseño asistido por ordenador para el modelado de objetos 3D.
- *Open Ephys GUI*: Interfaz gráfica para el control de la plataforma Open Ephys.
- *MultiChannel Experimenter*: Interfaz gráfica para el control de las plataformas diseñadas por MultiChanels Systems.
- *MultiChannel Analyzer*: Interfaz gráfica para el procesado de los datos capturados con las plataforma de la marca MultiChannels Systems.
- *MultiChannel Data Manager*: Interfaz gráfica para la exportación de los datos capturados con las plataformas de la marca MultiChannels Systems.

2. Contexto

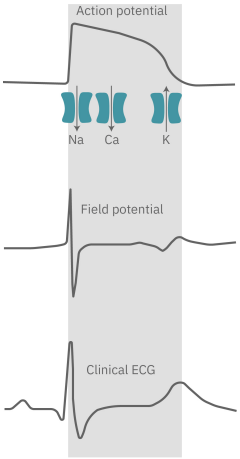
2.1. Señal de interés

La actividad eléctrica para la cual está orientado el sistema es generada por las células del músculo cardiaco, conocidas como cardiomiocitos. Esta actividad se origina a nivel celular, sin embargo, en este proyecto se registra la captura a nivel de tejido. De esta manera, la señal capturada no proviene de una única célula, sino que refleja la actividad combinada de varias células. A lo largo de la membrana celular se encuentran proteínas denominadas canales iónicos encargadas del paso de los iones entre el exterior y el interior de la célula. Este intercambio de iones produce variaciones en el potencial eléctrico que atraviesa la membrana celular, variaciones conocidas como potencial de acción *(AP)* presentando la forma mostrada en la figura 2.1.a.

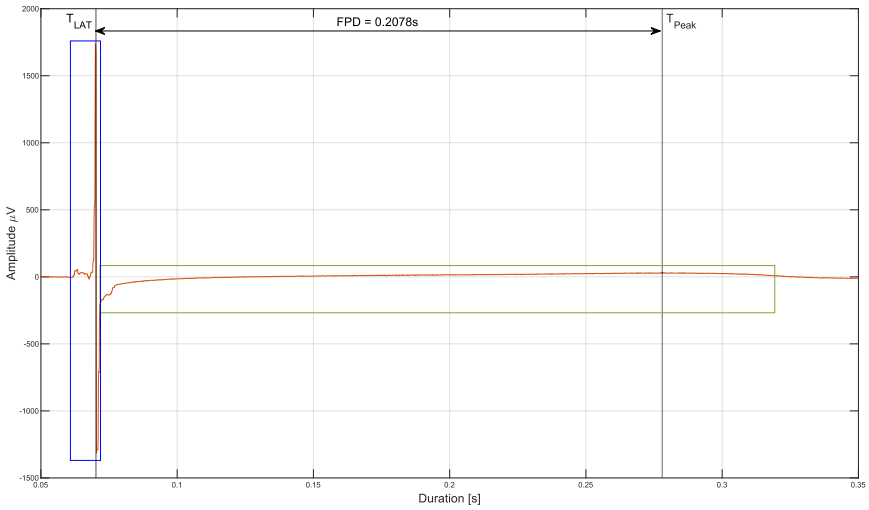
Cuando un cardiomiocito dispara un potencial de acción, que puede ocurrir de manera esporádica o ser provocado por un estímulo externo, la señal eléctrica se propaga a través del tejido o de la estructura celular. Esto provoca que todas las células por las que pasa la onda de activación disparen su potencial de acción. Los electrodos colocados a lo largo del tejido permiten captar la distribución de potenciales a lo largo del mismo. La superposición de los potenciales de acción de las células cubiertas por cada electrodo da lugar al potencial de campo (FP), siendo esta una señal capturada extracelularmente. La figura 2.1.b muestra la forma de las señales. La medida del potencial del campo se ha validado en una gran variedad de artículos, presentando beneficios en el ámbito de la investigación farmacológica [2] y la mejora de tratamientos médicos [3].

El potencial de campo presenta unas características específicas en cuanto a amplitud de la señal y características temporales. Es posible dividir la señal en dos fases de diferentes características: una fase inicial de despolarización y la fase de repolarización. Estas características se marcan en la figura 2.1.

La despolarización, marcada con un rectángulo azul en la figura 2.1.b, es la primera parte de la señal y se caracteriza por una rápida variación de la amplitud de la señal. Su comportamiento es similar al complejo QRS de un electrocardiograma [4], marcando el inicio de la actividad cardiaca. Esta sección está marcada por alcanzar el máximo y el mínimo absoluto de la señal, comprendidos entre $\pm 2mV$. Dentro de esta zona tomamos como tiempo de referencia T_{LAT}, el punto de mínima derivada [3, p. 2].



(a)



(b)

Figura 2.1: a) Comparación del potencial de acción, potencial de campo y ECG como referencia. Se indican los canales iónicos que actúan en cada momento junto con la dirección de los iones y el periodo de duración del potencial de campo, marcado como la zona gris. Extraída de Axion biosystems. b) Ejemplo de potencial de campo sobre el que se marcan las principales características temporales.

Por otro lado, la fase de repolarización, marcada con un rectángulo verde en la figura 2.1.b, continua tras la rápida caída de la señal. A lo largo de este periodo se produce una lenta subida hasta el punto máximo tras lo que se recupera el valor de reposo. Dentro de este periodo se destaca el punto T_{peak} que representa el punto máximo relativo en esta fase de la señal. Se calcula entonces la duración del potencial de campo (FPD) como resultado de la diferencia $T_{peak} - T_{LAT}$. Su duración media en tejido cardiaco se encuentra en torno a los 250ms [2].

2.2. Equipos utilizados

Para la adquisición de señales provenientes de células o tejidos *ex-vivo* existen distintas soluciones en el mercado. A lo largo de este proyecto se han utilizado dos soluciones comerciales: el sistema *MEA2100-Mini-systems* fabricado por la marca *MultiChannelSystems (MCS)* y la plataforma *Open-Ephys*. Ambas plataformas están destinadas a la adquisición de señales electrofisiológicas provenientes de matrices de multielectrodos (MEAs). A continuación se presentan los distintos equipos y la plataforma de adquisición de diseño propio.

Para establecer una comparación entre los distintos equipos se han tenido en cuenta parámetros como el número de canales que puede capturar simultáneamente, el rango de la señal de entrada, el ruido de entrada, la frecuencia de muestreo máxima y la resolución del conversor. El número de canales permite conocer las dimensiones máximas del MEA a utilizar, un mayor número de electrodos permite mayor resolución espacial a costa de un mayor gasto de recursos en el bloque encargado de la conversión.

2.2.1. MEA2100-Mini-system

La diversidad de los equipos del fabricante MCS lo convierten en la opción predominante en la investigación electrofisiológica, siendo utilizado en la gran mayoría de los artículos publicados en los último años. Dispone de distintas versiones de su producto para adaptarse a las necesidades de la investigación. En este proyecto se ha utilizado la versión *MEA2100-Mini*[5], una plataforma para adquisición de señales que permite la captura con MEAs de 60 o 120 electrodos. Como se observa en la figura 2.2 el sistema se divide en varios bloques para cumplir todas las funciones que implementa.

Figura 2.2: Diagrama de bloques de la plataforma MCS2100-Mini, Extraído de la web de MultiChannelSystem.

Este modelo presenta una resolución de 24 bits y un rango de la señal de entrada de $\pm 70mV$, permitiendo una resolución menor de $1nV$. Además presenta un nivel de ruido

a la entrada de $0.7\mu V_{RMS}$. En cuanto a la frecuencia de muestreo, permite capturar con velocidades de hasta 50ksamples/s.

El hardware se cumplimentan con una suite de aplicaciones informáticas que permiten al usuario configurar los experimentos y capturar los resultados (*Multi Channel Experimenter*), realizar un análisis de los resultados de una forma sencilla (*Muli Channel Analyzer*) y exportar los datos capturados en distintos formatos de archivo (*Multi Channel DataManager*). El flujo de trabajo comienza con la adquisición de muestras, que almacena los datos en varios archivos, para su posterior procesado con la herramienta *Multi Channel Analyzer*. Si se quiere exportar los archivos es necesario utilizar la tercera aplicación, que permite los siguientes formatos de salida: HDF5, NEX, CED/CED-64, ASCII (CSV), EDF+/BDF+.

2.2.2. Open-Ephys

La empresa *Open-Ephys* se dedica al diseño y fabricación de tecnología de código abierto para la adquisición de señales electrofisiológicas. Aunque la suite de herramientas que propone está principalmente ideada para la captura de señales neuronales, es posible su uso para la captura de otro tipo de señales como las provenientes de tejido cardiaco, en las que se centra este proyecto. Al igual que en el caso anterior, la plataforma se compone de varios bloques encargados de distintas funciones como se puede ver en la figura 2.3.

El núcleo de la plataforma es la placa de adquisición de señales, interfaz para la conexión entre el *headstage* encargado de la captura y el ordenador. Este bloque permite el trabajo con hasta 8 *headstages* simultáneos. El componente principal de la placa de adquisición es una FPGA *Lattice ECP5U85*[7] sobre la que se implementa toda la lógica. Además, todos sus puertos utilizan conexiones estandarizadas permitiendo una fácil interconexión con distintos equipos como los *hedastages* o salidas analógicas/digitales de otros equipos. La placa de adquisición permite una frecuencia de muestreo comprendida en el rango 1-30kHz. La comunicación con el ordenador se hace a través de una conexión USB 3.0, utilizando para ello el chip *FT600* de la marcar *FTDI*. El chip utiliza una conexión en paralelo como entrada de los datos a enviar por el puerto, permitiendo una mayor tasa de transmisión.

El *headstage* es el primer bloque de la cadena de adquisición, es el bloque encargado de la conversión analógico-digital de la señal. En el caso de esta plataforma, está preparada para trabajar con toda la gama de chips del fabricante *Intan*. Estos circuitos integrados

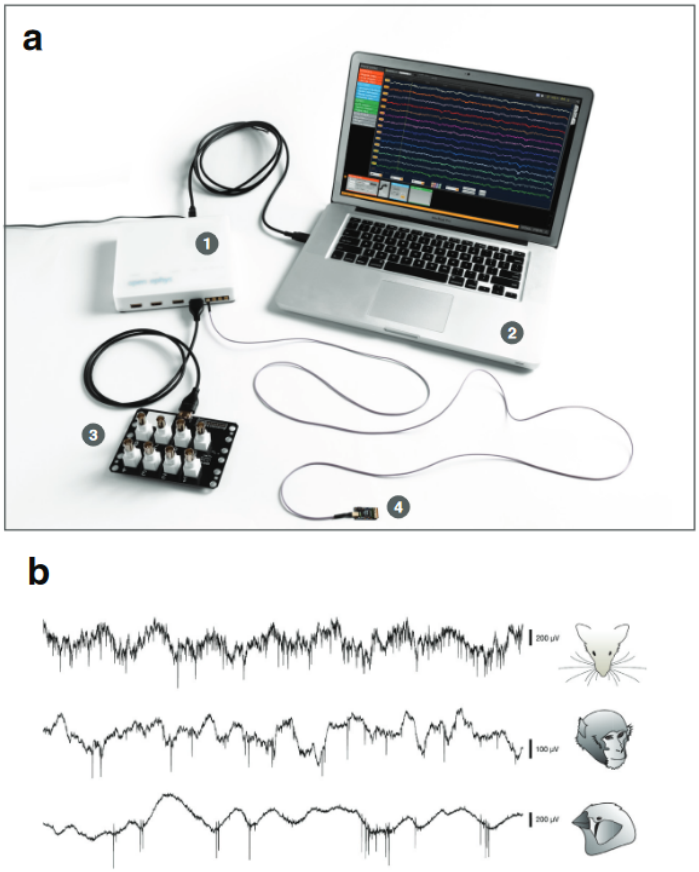


Figura 2.3: a) Configuración estandar del sistema. Consta de la placa de adquisición (1), ordenador con la aplicación (2), placa para entradas digitales (3) y headstage (4). b) Ejemplo de señales neuronales obtenidas con el bloque.[6]

son una de las opciones más utilizadas en esta campo, gracias a que incorporan toda la cadena analógica y de conversión en un único chip. En el presente trabajo se utiliza el *INTAN RHD2000*[8] tanto para trabajar con la plataforma Open Ephys como con la plataforma de desarrollo propio. Sin embargo, el *Open Ephys* no permite modificar la configuración inicial del *headstage* fijando los valores por defecto para todas las capturas realizadas.

El circuito integrado cuenta con bancos de amplificadores de bajo ruido multiplexados sobre un conversor analógico-digital de 16 bits y con una resolución de $0.195\mu V$. Puede trabajar hasta una frecuencia de muestreo de 30kSamples/s, superior a la utilizada normalmente para trabajar con señales cardiacas. Además presenta un bajo nivel de ruido, con valor de $2.4\mu V_{RMS}$. Dentro del propio integrado se dispone de bancos de filtros paso bajo y paso alto configurables. El RHD2000 incluye una serie de registros para la configuración del chip (regs 0-17) y otros con información de solo lectura (40-44 y 60-63). La comunicación con el headstage se realiza a través de una interfaz SPI en

una arquitectura *master-slave*. El chip trabaja como esclavo utilizando para su control desde la FPGA una serie de 5 comandos: WRITE y READ, dirigidos hacia los registros; CALIBRATE y CLEAR, dirigidos al control del ADC; y finalmente CONVERT para convertir el canal indicado.

Del mismo modo que la plataforma anterior, el sistema incluye una interfaz gráfica (*Open-Ephys GUI*) destinada a controlar el sistema, configurar los experimentos y la visualización de los canales en tiempo real. Al estar pensada para la captura de señales neuronales la mayoría de las funciones que implementa están enfocadas en este tipo de señales, sin embargo, mantiene la ideología de cógido abierto y permite el diseño de nuevas funcionalidades y la implementación de estas por medio de plugins creados por la comunidad. La interfaz permite exportar los datos en tres tipos de ficheros distintos con su propia organización de carpetas y ficheros: Binary (.bin), formato propio (open ephys format) y NWB.

2.2.3. Sistema de desarrollo propio

El trabajo desarrollado utiliza como punto de partida una plataforma de adquisición de señales propuesta en varios trabajos fin de estudios anteriores [9, 10]. El desarrollo completo de una plataforma de adquisición de señales permite tener total control sobre las prestaciones del sistema y adaptarla a los requisitos de la investigación en la que se aplica.

En la figura 2.4 se puede ver la estructura del sistema, en el que se mantiene un diseño por bloques. Los datos se adquieren a través de un MEA de diseño propio y se digitalizan en el *headstage*, que se comunica con un bloque encargado de su control. Del mismo modo, desde el ordenador se podría modificar la configuración del bloque de control y desde este, enviar las acciones a realizar al *headstage*. Como se aprecia en la figura 2.4, el flujo de información es bidireccional entre los bloques que componen el sistema.

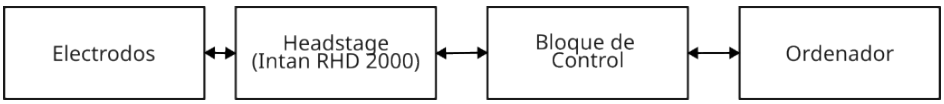


Figura 2.4: Diagrama de bloques del sistema completo.

Matriz de electrodos

Dado que este proyecto se centra en la validación y mejora de las etapas digitales de la plataforma, los electrodos no han sido utilizados durante la realización del mismo. En su lugar se utiliza un generador de señales de bajo voltaje que permite emular las señales capturadas por los electrodos, se detalla más a fondo en la sección 6.1.

En esta subsección únicamente se detallan ciertas especificaciones de los mismos y no se describe el proceso de fabricación llevado a cabo. La matriz de electrodos que utiliza la plataforma se diseña en una lámina de poliamida, permitiendo flexibilidad para adaptarse a la muestra de tejido [9]. La versión actual del MEA cuenta con un total de 16 electrodos, sin embargo la plataforma se diseña para trabajar con hasta 32 canales.

Headstage y bloque de control

La plataforma, al igual que el *Open Ephys*, utiliza como *heastage* el circuito integrado *Intan RHD2000*. La etapa de adquisición está conformada por el kit de desarrollo *Artix-7 100T* fabricado por *Digilent*, que se conecta al *headstage* por medio de un chip *SN65LVDT14* de *Texas instrument*, encargado de traducir la interfaz SPI de la FPGA a señales LVDS. Sobre la FPGA de la placa de desarrollo se implementa, con una arquitectura *softcore*, un microprocesador *Cortex-M1* junto a los periféricos necesarios para el funcionamiento de la plataforma. En la figura 2.5.a se pueden ver los distintos elementos que conforman el bloque de control. Para la conexión con el ordenador el kit incorpora un chip *FT2232HQ*[11] de la marca *FTDI* que permite la conexión por medio de USB 2.0. En el kit de desarrollo se utiliza una comunicación serie como entrada del chip de *FTDI* que junto al uso de un estándar USB con menores prestaciones, reduce la velocidad de transmisión frente al equipo comercial *Open Ephys*.

Todo el proceso de comunicación y control del *headstage* se realiza desde un periférico diseñado específicamente para cubrir esta función [10]. El periférico, representado en el diagrama de bloques de la figura 2.5.b, se encarga de realizar la configuración inicial del Intan, en función de unos parámetros predefinidos por el usuario en los registros de configuración. También permite el envío de los comandos descritos en la sección 2.2.2 directamente desde el microprocesador. El comando a enviar se escribe en uno de los registros del periférico (*DirectCMD*) y este devuelve la respuesta inmediata en otro de los registros (*DCResp*).

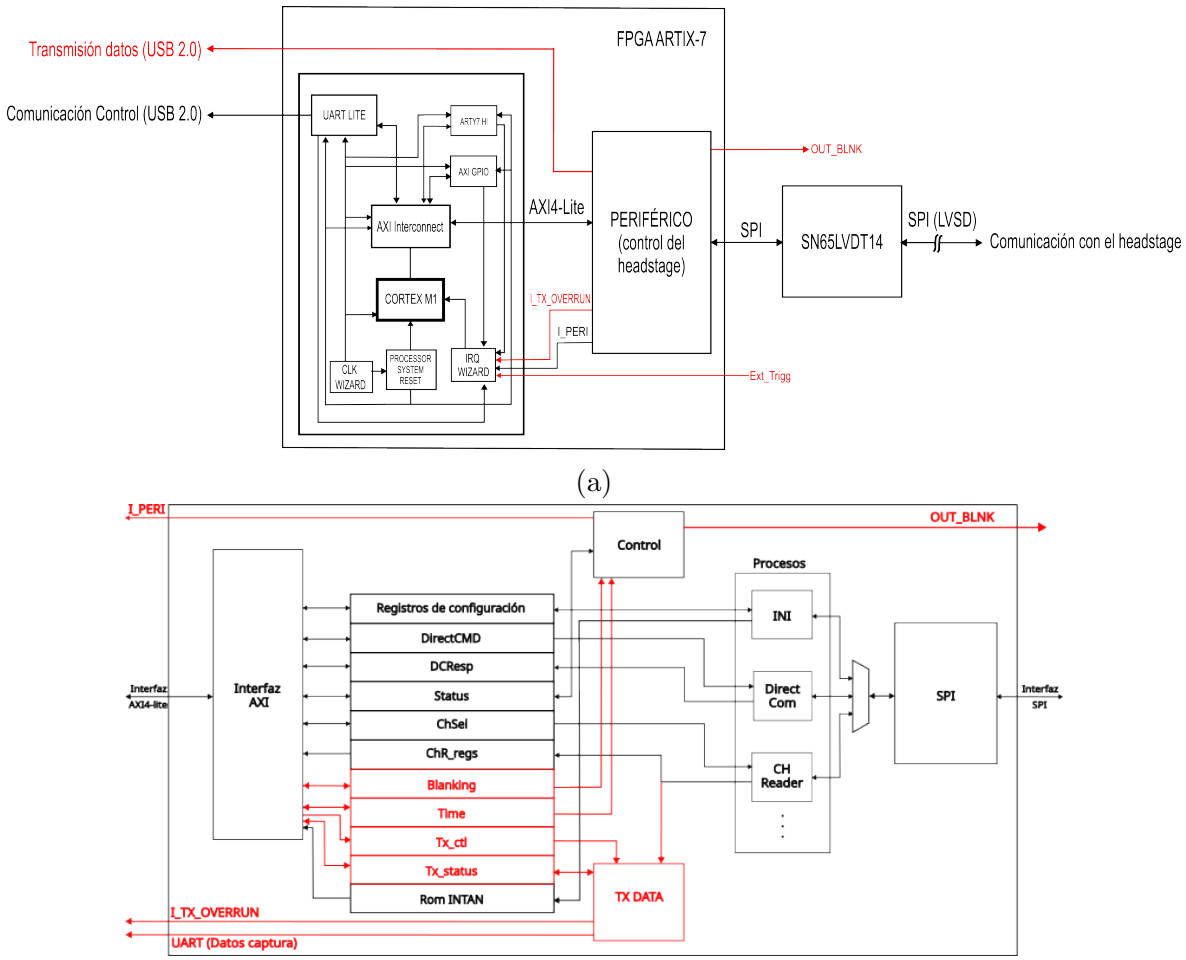


Figura 2.5: a) Diagrama de bloques de la etapa de adquisición. b) Diagrama de bloques del periférico, en rojo aparecen las modificaciones realizadas durante este proyecto sobre la plataforma inicial.

La función principal del periférico es el bucle de adquisición (*CHLoop*). En este modo de trabajo, el periférico está preparado para realizar capturas periódicas de una serie de canales, seleccionados por el usuario en el registro *ChSel*, a una frecuencia de muestreo también programada por el usuario. Los datos resultantes de estas capturas se almacenan en unos registros a los que se accede desde el microprocesador para su lectura. En la versión inicial, el periférico era controlado íntegramente desde el registro *Status*, desde donde se configuraba la frecuencia de muestreo de los canales y se daba inicio a los distintos modos de trabajo.

En esta versión se programaba en lenguaje C, sobre el microprocesador, las rutinas que ejecutaba el sistema. Lo que hace necesario modificar el *firmware* al querer modificar los parámetros de la captura. Los datos capturados durante el bucle de conversión de canales eran leídos por el microprocesador y se enviaban a través de una interfaz UART hacia el ordenador.

Esta versión inicial del periférico presenta una serie de limitaciones. La configuración del mismo es fija, siendo necesario modificar el *firmware* cada vez que se quiera realizar un experimento con unos parámetros distintos. Como resultado, es necesario que el usuario de la plataforma tenga conocimientos en programación o disponga de herramientas para poder cargar el nuevo firmware en el microprocesador.

Otro de los problemas del bloque es una limitación en el envío de los datos. La interfaz UART por la que se envían los datos está formada por la IP *UART-Lite* desarrollado por *Xilinx*. Este bloque se encarga del envío de los datos hacia el ordenador, sin embargo presenta una limitación en la tasa de transmisión. Esto afecta a la hora de configurar la captura de los canales poniendo un límite al número de muestras que es posible capturar, que debe ser igual o inferior al número de palabras que puede gestionar la interfaz UART sin llegar a llenar el *buffer* utilizado en la transmisión.

3. Mejoras en el bloque de control

Se han realizado tres modificaciones principales sobre la plataforma original. En primer lugar se realiza una modificación en el bucle de adquisición de las señales, *ChLoop*. A continuación, se separan las interfaces de transmisión de datos y control para finalmente modificar el firmware de la plataforma implementando una interfaz por linea de comandos, que permite la gestión de la plataforma desde el ordenador.

3.1. Captura de la señal (ChLoop)

En la versión inicial de la plataforma, como se comenta en la sección anterior, el bucle de adquisición realizaba una captura continua de los canales seleccionados a la frecuencia de muestreo introducida por el usuario. La duración de esta captura se controlaba a través del *firmware*. Este funcionamiento es similar al de los equipos comerciales, que realizan una captura continua hasta que el usuario la detiene. Si embargo, este tipo de captura desaprovecha recursos en el experimento hacia el que se enfoca esta plataforma.

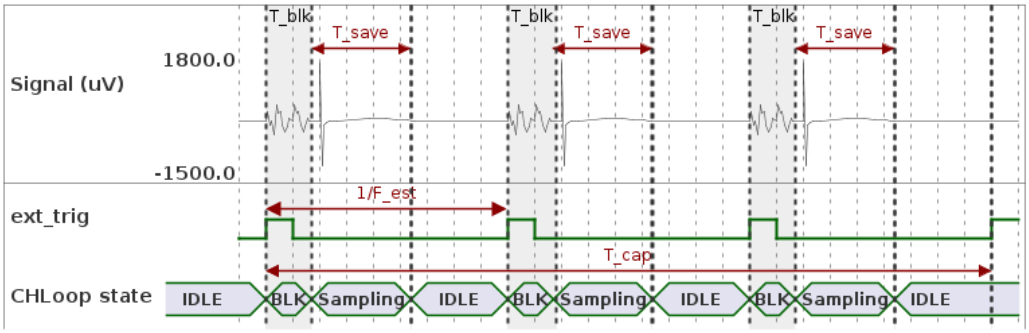


Figura 3.1: Representación temporal del resultado de dos estimulaciones. La señal *Signal* representa la señal registrada en el MEA, *ext_trig* representa el trigger que indica un nuevo estímulo y *ChLoop state* muestra el estado de la conversión. Se marcan los periodos temporales del bucle de adquisición.

Como se observa en el ejemplo de experimento representado en la figura 3.1, el potencial de campo (FP) se genera como resultado de una estimulación con un generador externo. La señal solo se genera tras la estimulación y su duración es mucho menor que el periodo de estimulación. Esto se traduce en que, al capturar de manera continua, gran parte de la señal que se almacene no contendrá información. El uso de un generador supone otro

problema para el sistema de adquisición, debido a los valores de corriente utilizados para la estimulación, genera artefactos sobre la señal capturada por los microelectrodos en los instantes siguientes al estímulo pudiendo llegar a saturar el *headstage*. Además, previo al experimento se calcula el número de estimulaciones en este, a partir de la duración del experimento, T_{cap} , y la frecuencia de estimulación, T_{est}.

Para mitigar el problema, se propone una mejora en la que el sistema solo realizará la adquisición de señal durante un tiempo determinado, T_{save}, introducido por el usuario. Se utiliza una señal TTL, proveniente del estimulador, que actúa como *trigger* indicando el comienzo de un estimulo. El flanco de subida de la señal, *ext_Trigg* en la figura 2.5.a, se utiliza como interrupción y pone en marcha el proceso de adquisición *ChLoop*.

La primera etapa del proceso consiste en un periodo de tiempo en el que el periférico se mantiene en reposo. Este periodo de *blanking*, T_{blk} se configura por el usuario en el registro *Blanking* del periférico. El valor por defecto propuesto es de $25\mu s$. Una vez terminado este periodo, se procede a la captura de la señal durante el periodo T_{save}. El muestreo de los canales genera una interrupción al finalizar la conversión de los canales. Se configura el *firmware* para calcular el periodo capturado por medio de un contador de esta interrupción. Al alcanzar el tiempo T_{save}, se desactiva el bucle de adquisición y el *headstage* queda en reposo hasta que se detecte una nueva señal de *trigger*. Al llegar al número de estimulaciones totales se desactiva el bucle de adquisición y el sistema queda en reposo. A modo de resumen del proceso, se presenta la figura 3.1 con una representación del funcionamiento del bucle de adquisición con 3 estímulos totales en la captura.

3.2. Comunicación ordenador-FPGA

La comunicación con el ordenador se modifica para hacer uso de dos interfaces UART, separando el envío de los comandos control del envío de los datos. Durante el periodo de captura se convierten y almacenan una gran cantidad de datos que es necesario enviar al ordenador para su almacenamiento. Inicialmente se utiliza el IP desarrollado por Xilinx para la transmisión UART, trabajando por encima de las especificaciones recomendadas. Esto generaba fallos en la comunicación.

Transmisión de datos

La interfaz encargada del envío de los datos se encuentra dentro del periférico de control del *headstage* y solo está preparada para la transmisión desde la FPGA. Permite configurar la tasa de transmisión, por defecto establecida en 1Mb/s, pudiendo llegar hasta un máximo de 12Mb/s limitados por el chip de *FTDI* del que dispone la placa [11]. Esta interfaz se controla desde los registros *Tx_ctl* y *Tx_status* del periférico, figura 2.5, y para su implementación se utilizan 2 registros de control y una memoria FIFO de 65.536 palabras de 16 bits en la que se almacenan directamente los datos a enviar.

La transmisión de los datos solo se realiza durante el bucle de adquisición donde los datos capturados durante los periodos T_{save} se almacenan en los registros del periférico y se envían simultáneamente esta interfaz para su transmisión, en caso de estar habilitada. La transmisión de los datos se realiza formando paquetes con las muestras de todos los canales correspondientes al mismo instante. Estos paquetes son precedidos por una cabecera que facilita el trabajo con los datos una vez llegan al ordenador, pudiendo detectar perdidas de bytes o de paquetes enteros. Debido a la resolución del *headstage* las muestras tienen un tamaño de 16 bits, por lo que cada muestra se transmite como dos palabras de 8 bits en formato *big-endian*. En el anexo A se detalla el funcionamiento de la transmisión.

La interfaz de control representa una limitación para el resto de la plataforma. Debido a la tasa de transmisión de la interfaz (R_{Tx}), el sistema tendrá que reducir el número de muestras que pueda generar. Para ello, el usuario deberá encontrar un balance entre el tiempo de captura (T_{save}), la frecuencia de estimulación (F_{est}) y el número de canales a capturar (N_{Ch}). Este balance viene dado por la inecuación 3.1, donde se compara el número de muestras que se envían por la interfaz a lo largo de una estimulación y el número de muestras que se generan en cada estimulación, en función de los parámetros que puede controlar el usuario. Este dimensionado de la comunicación permite que no se almacenen muestras en la FIFO al final de la estimulación, que en función de los parámetros podría llegar a llenar la memoria y resultar en la pérdida de muestras.

$$\frac{R_{Tx}}{20 \cdot F_{est}} \geq F_s \cdot (N_{ch} + 1) \cdot T_{save} \tag{3.1}$$

Estas modificaciones, implementadas sobre el periférico, se han añadido a la documentación del periférico, que se incluye en el anexo A.

3.2.1. Control

La interfaz encargada del control está directamente gestionada por el microprocesador utilizando el IP *UART-Lite* de Xilinx. Esta interfaz transmite los comandos y sus respuestas a una tasa estándar de 115200b/s, al no necesitar transmitir una gran cantidad de datos. A través de esta comunicación se transmiten los comandos que permiten configurar todas las opciones de la plataforma. Para facilitar la configuración, se añade un sistema de comandos enviados desde el ordenador para cubrir todas las funcionalidades de la plataforma.

3.3. Interfaz por línea de comandos

Para controlar la plataforma desde el ordenador es necesario implementar primero un diccionario de comandos que permita la comunicación bidireccional entre ambas partes. Se analizan algunas opciones ya existentes como es el caso del *Standard Commands for Programmable Instruments (SCPI)*[12], normalmente utilizado para equipos de laboratorio y medida. Finalmente se opta por implementar un sistema de comandos propio que se envíen directamente como caracteres de texto.

Se define un total de cuatro comandos principales, que se envían por la interfaz de control. Dos de ellos permiten realizan diferentes procesos en función de los argumentos de entrada. En la tabla 3.1 se encuentra un resumen de los comandos implementados y una breve descripción de su función. El comando *test* permite verificar la comunicación, provocando que la plataforma devuelva una cadena de texto conocida que valida el correcto funcionamiento. El resto de los comandos están diseñados para interactuar directamente con el *headstage* de la plataforma, modificando la configuración de sus registros (comando de inicialización) o enviando las órdenes directamente. El bucle de adquisición consta de dos comandos para su configuración (*config*) e inicio (*start*). para el dimensionado de los parámetros del comando *config* es necesario tener en cuenta la inecuación 3.1 para evitar errores en el proceso. Para facilitar la configuración del bucle, en la interfaz desarrollada en el capítulo 5 encargada de la gestión de la plataforma, se incluye la comprobación automática de los parámetros.

A la hora de implementar la interfaz, los comandos se reciben como cadenas de caracteres, utilizando como terminadores el retorno de carro para el comando transmitido y el carácter de nueva linea para la respuesta de la plataforma. Se diseña también un eco desde la plataforma con el que cada carácter recibido por la plataforma se

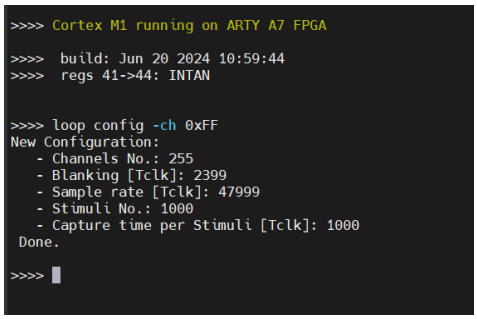


Figura 3.2: Extracto de la interfaz por linea de comandos donde se aprecia la respuesta al comando introducido.

devuelve al ordenador para que se represente en el terminal, consiguiendo así que el usuario pueda tener control sobre lo que se transmite. En la figura 3.2 se muestra un extracto del terminal en la que se pueden ver la respuesta de la plataforma al inicio de la conexión, primeras lineas, junto con el comando introducido por el usuario y su respuesta.

Nombre	**Comando**	**Modificador**	**Argumentos**	**Descripción**
Test	test	.	.	Comando de prueba, devuelve una cadena de texto
Inicialización	ini	.	.	Inicia el proceso de inicialización
		conf	-Rxx < reg value>	Envía la nueva configuración para la inicialización. Permite actualizar los 18 registros modificando el nombre del arg. (R00, R01,...)
Comando directo	cmd	read	<addr>	Envía un comando de lectura, con la dirección a leer
		write	<addr> <data>	Envía un comando de estructura sobre uno de los registros
		convert	< channel no.>	Envía un comando de conversión sobre un canal determinado
		calibrate	.	Envía un comando de calibración de los ADCs
		clear	.	Envía un comando para limpiar la calibración de los ADCs
Bucle de captura	loop	congif	-ch <channel sel. (hex)> -blnk <blanking time> -fs <sample freq> -nest <stim no.> -Tc <stim. save period>	Configura el bucle de captura modificando sus parámetros. No es necesario enviar todos los argumentos, es posible modificar solo los parámetros necesarios.
		start	.	Inicia el bucle de conversión

Tabla 3.1: Resumen de los comandos implementados para la comunicación con la plataforma.

4. MEA-GUI: Procesado de los datos

Tras la captura de los datos es necesario poder trabajar con ellos de una manera sencilla para extraer la información necesaria. Para esto, los equipos comerciales como los nombrados en la sección 2, incluyen sus propias *suites* de trabajo para el procesado de los datos. En estos casos, las aplicaciones suelen disponer de una variedad de herramientas muy completa, pero únicamente permiten su uso con los datos capturados con equipos de la misma marca y no están abiertas a modificaciones en el procesado.

A lo largo de los últimos años han aparecido algunas opciones *open source* que buscan eliminar la restricción entre el hardware utilizado y el software para el procesado de los datos. Estas opciones a menudo tienen una interfaz más simple que sus homólogas comerciales, solo permiten funciones muy básicas y están orientadas a las funciones que utiliza la investigación hacia la que va destinada. Algunos ejemplos de estas aplicaciones son *MEA-Tools*[13] o *ToolConnect*[14] (destinadas al análisis de actividad neuronal), *MultiElec*[15]. Las opciones anteriores están desactualizadas, sin embargo existen interfaces más actuales como es el caso de *MEAnalyzer*[16] o *CardioDMA*[17], que no es del todo libre pues requiere de un registro previo a la utilización de la aplicación.

Fomentado por intentar adaptar la interfaz a las necesidades de la señal de interés y las pocas opciones libres que existen para estas, se ha diseñado una interfaz de usuario que permita la representación y el trabajo con los datos capturados. Con el objetivo de conseguir que usuarios con pocos conocimientos técnicos puedan utilizar la interfaz de una manera intuitiva y que sea fácil incluir nuevas funcionalidades en forma de *scripts* se opta por utilizar como base *MATLAB*, ya que es una herramienta ampliamente extendida en el mundo de la investigación y permitirá una rápida familiarización con el software.

Uno de los problemas que presentan las herramientas comerciales es la poca libertad para realizar modificaciones, además de tener un sistema de archivos único para sus aplicaciones. Esto dificulta la aparición de herramientas estándares [13]. Para mitigar este problema se diseña la interfaz para trabajar distintos tipos de ficheros, correspondientes a los equipos de adquisición utilizados a lo largo del proyecto.

Para conseguir la interoperabilidad entre los distintos sistemas de adquisición y la interfaz de procesado se lleva a cabo una normalización de los datos capturados. Se implementa la clase *LabData* que contiene tanto la información de interés sobre la captura

y los datos correspondientes a las señales capturadas. En la figura 4.1 se muestra un diagrama de la clase que, además de incluir los datos provenientes de los ficheros. En el anexo B, se recoge en detalle todos las propiedades y funciones de la clase *LabData* que se muestran en la figura 4.1.

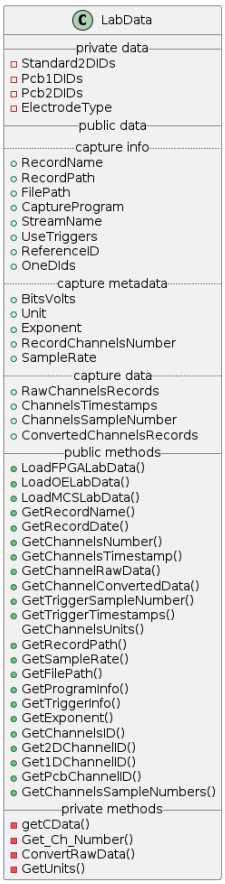


Figura 4.1: Diagrama de la clase *LabData*.

La clase dispone de tres métodos que permiten importar datos generados con los sistemas utilizados en el proyecto: La suite de Multi Channel System (*.h5), La plataforma Open-Ephys (*.oebin) y la plataforma de desarrollo propio (*.mat). Los tres métodos tienen un funcionamiento similar en los que se distinguen tres fases: extracción de metadatos, extracción de los datos y conversión de los datos. En la figura 4.1 se pueden observar los atributos generados en cada etapa del método.

Durante el la extracción de los metadatos se recuperan campos de las clases originales que incorporan información relevante para el usuario. Esto permite identificar la captura, ver la duración total o conocer el número de electrodos capturados. También se asignan los números de identificación de los electrodos capturados, los cuales varían dependiendo del equipo utilizado. Simultáneamente se almacenan también los atributos

necesarios para realizar la siguiente parte del método.

A continuación se realiza la recuperación de la señal. Para reducir el tamaño de los archivos, los distintos equipos comerciales almacenan los datos capturados como números enteros, con el valor a la salida del conversor A/D. Para conseguir el valor real de la señal se aplica un factor de conversión, que varía entre los distintos equipos y se normaliza para trabajar en todos los casos en microvoltios (μV). Este proceso se aplica para todos los canales que se han capturado. Finalmente se extrae también el eje de tiempos. Resulta de interés sincronizar los datos durante el procesado, para ello se utiliza una señal de *trigger* generada de manera síncrona con el estímulo aplicado al tejido y que posteriormente se almacena en la nueva clase por medio del método correspondiente.

En el caso de la plataforma de diseño propio es necesario un proceso que permita reconstruir la señal ya que, como se explica en el capítulo 3, solo se capturan fragmentos de esta. Para reconstruir la señal se añaden ceros entre cada evento, rellenando el tiempo entre estímulos para adaptar los datos capturados al eje de tiempos real.

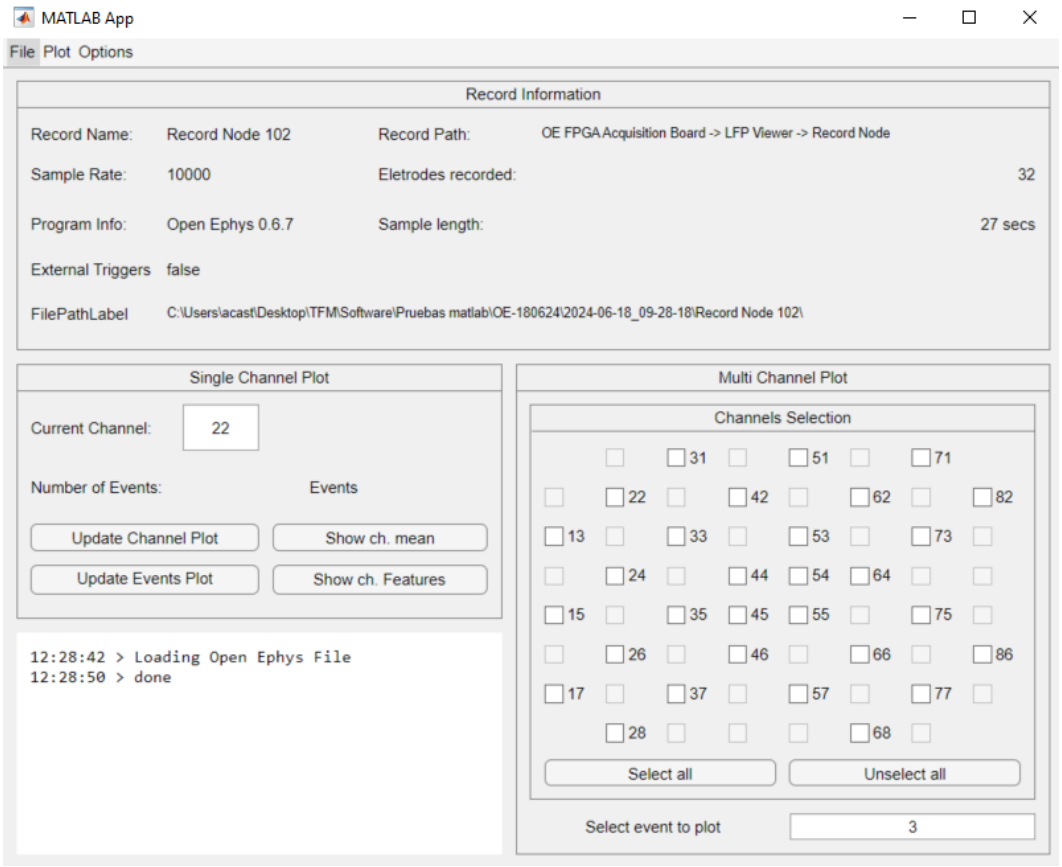


Figura 4.2: Ventana principal de la GUI. Encargada del control general de la aplicación, también muestra información básica sobre la captura.

4.1. Ventana de control

La aplicación inicia con una ventana principal (Figura 4.2) desde la que se controlará la mayor parte de las funciones de la aplicación. En la parte superior, hay un menú con tres opciones desplegables con la estructura de submenús que aparece en la figura 4.2: *File* engloba la importación y exportación de ficheros, *Plot* permite abrir o cerrar la pantalla de gráficas y *Options* incluye una variedad de las funciones a utilizar con las señales y para la configuración. Se muestra también un resumen de los metadatos de la señal, mostrando información útil para el usuario.

En la parte inferior, encontramos dos paneles que nos permiten controlar los distintos tipos de gráficas que genera la interfaz. Además se añade una pequeña ventana en la que se muestran las acciones realizadas por el usuario y su resultado, de esta manera el usuario es consciente de las acciones que se están llevando a cabo y mejora la interacción.

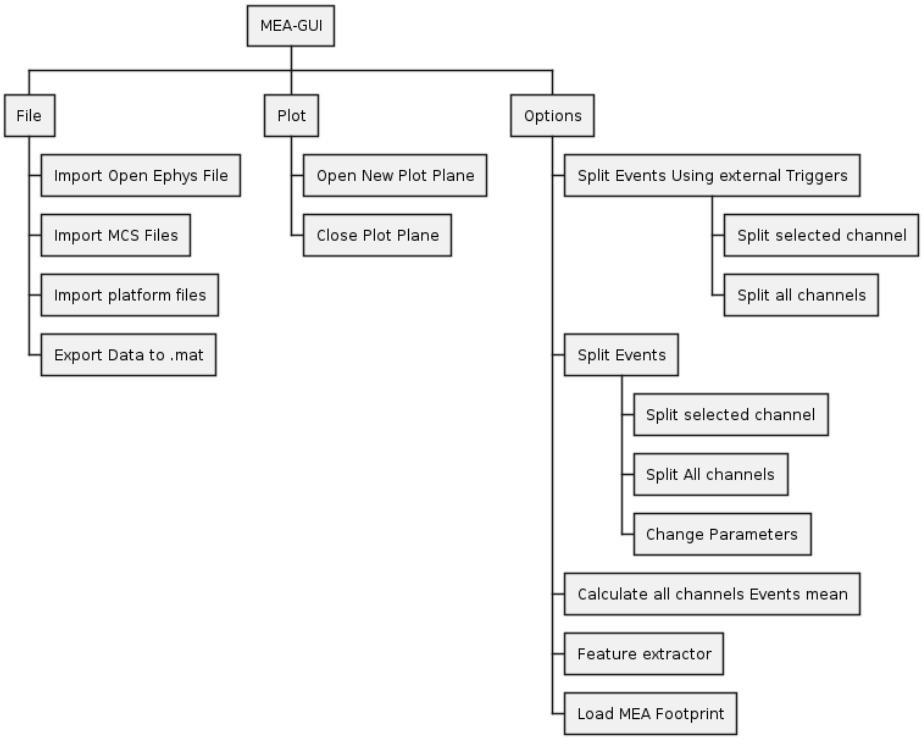


Figura 4.3: Diagrama de los 3 menús desplegables de la aplicación y las funciones que implementa cada uno.

Dentro del menú *Options* se implementa la opción de separar la captura en los eventos ocurridos. El tejido de la señal se estimula con un tren de pulsos eléctricos y se captura la respuesta para todos ellos, el análisis de los pulsos de manera individual

permite obtener información más detallada y descartar muestras en caso de que haya habido algún fallo en el proceso de captura. En caso de que la señal capturada no haya registrado el *trigger* de cada estimulación, se implementa un *script* para extraer los eventos en función de los parámetros de la señal capturada detectando los picos máximos de la señal. Los parámetros para la separación de eventos pueden modificarse desde una ventana emergente. Los eventos también sufren un proceso de normalización para conseguir la misma duración temporal entre todos y permitir así una mejor comparación.

Uno de los problemas principales de la señal es el efecto del ruido externo sobre ella. Se han estudiado distintas soluciones para un filtrado previo de la señal que permita suavizar los resultados: filtrado de media móvil, filtro de Savitzky-Golay y el cálculo de la media de los eventos en una misma captura. Estos métodos son ampliamente utilizado para el tratamiento de señales biológicas. En este caso se elige la opción de calcular la media de los eventos del canal ya que las características de la señal lo permiten, es periódica y la respuesta de los eventos mantiene las mismas magnitudes y tiempos en todos. Esto permite eliminar los posibles ruidos de alta frecuencia sin distorsionar la amplitud de la señal.

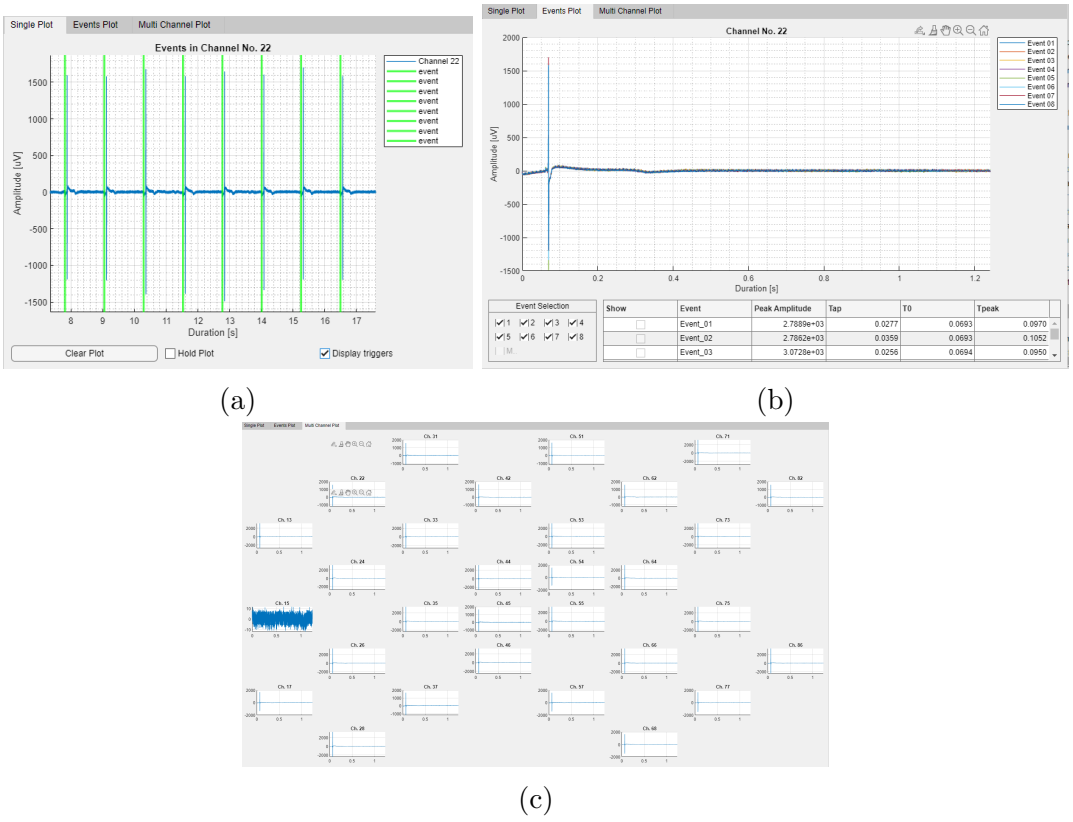
La interfaz automatiza la extracción de las características más importantes de la señal. Esto se realiza para todos los eventos de la captura y se muestran para que el usuario pueda analizar los datos. Las características extraídas son las explicadas en el apartado 2.1. Este proceso se implementa en un *script* externo a la aplicación, lo que permite al usuario modificarlo en caso de que requiera extraer algún parámetro en concreto. Además, el resultado de esta extracción se almacena en una estructura de Matlab, que se exporta junto a los datos de la señal de la clase *LabData* permitiendo su análisis fuera de la aplicación.

Finalmente el menú permite cargar un fichero *.mat* con una huella del MEA utilizado. Esta huella se utiliza para la representación espacial de los eventos. La huella por defecto corresponde a un electrodo de 60 canales con la distribución del fabricado por MCS.

Con los botones que aparecen en los paneles *Single Channel Plot* y *Multi Channel Plot* se controla una nueva ventana que se abre al cargar los datos a analizar. En esta nueva pantalla se mostrarán todas las representaciones temporales de la captura así como las características extraídas de los eventos.

4.2. Ventana de representación

En la pestaña principal, figura 4.4.a, se representa la señal capturada en un canal, con toda la duración. También permite mostrar la posición de los estímulos que ocurren durante la captura, y que generan la respuesta del tejido. El cambio del canal representado se realiza desde la aplicación principal. La aplicación ofrece la capacidad de representar distintos canales superpuestos de forma que sea posible analizar la propagación espacial del potencial de campo.



(a) (b)

(c)

Figura 4.4: Diferentes menús de la ventana de representación. a) Ventana representación de un canal, b) representación de los eventos y c) representación espacial de los canales

En la siguiente ventana se muestra una representación de todos los eventos del canal seleccionado (Figura 4.4.b). Se permite modificar los eventos representados, así como representar la media de todos los eventos. La representación de los eventos se superpone, facilitando el análisis de las diferencias entre ellos. También se muestran las características de la señal en todos los eventos en una tabla resumen.

Finalmente se utiliza otra ventana para representar todos los canales capturados manteniendo la estructura espacial (Figura 4.4.c). Dado que los recursos consumidos para la representación de todos los canales del MEA supone un largo tiempo de cómputo,

para agilizar este proceso se representa un único evento de cada canal y en caso de estar calculada es posible representar la media de cada canal. Es posible controlar los canales que se representan y el evento a representar con los botones del panel *Multi Channel Plot* de la ventana de control.

En la figura 4.5, se muestra un ejemplo del flujo de trabajo recomendado de la aplicación. Este flujo de trabajo permite extraer toda la información de la captura para finalmente exportar todos los datos obtenidos en un formato simple y que pueda utilizarse en Matlab para un análisis más profundo.

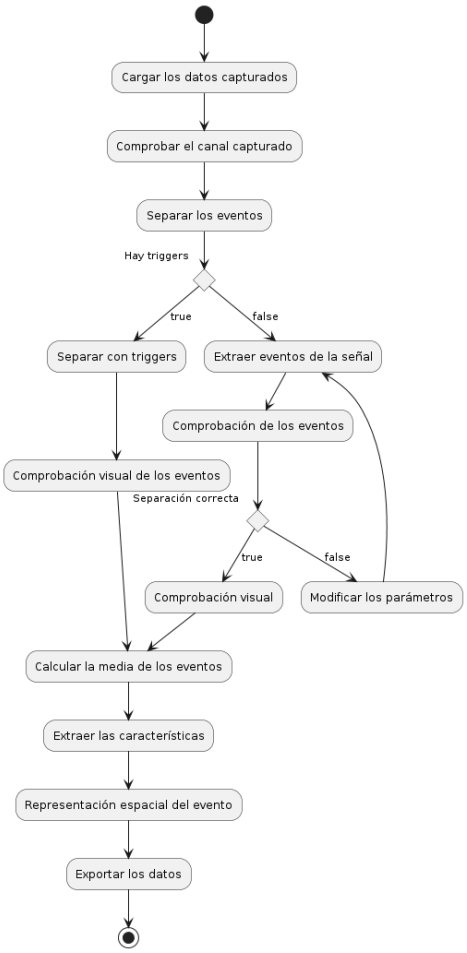


Figura 4.5: Ejemplo de diagrama de trabajo de la aplicación.

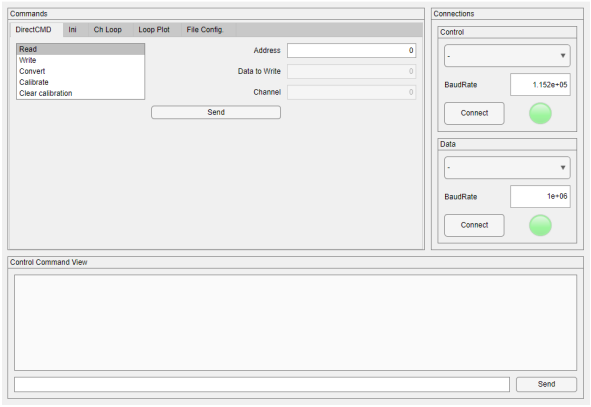
El flujo de trabajo comienza importando los datos capturados, independientemente de la plataforma utilizada para su captura. Tras esto se procede a una verificación visual de los canales, lo que permite encontrar fallos en el proceso de captura o conversión. A continuación se realiza la separación de los eventos que conforman la captura, haciendo uso de los *triggers* o extrayendo los eventos de la propia señal. En este último caso, es necesario revisar y modificar los parámetros para adaptarlos a la señal que se está analizando.

Una vez obtenidos los eventos, se realiza el cálculo de la media de estos, eliminando así posibles defectos en la señal, y se extraen las características de todos los eventos y de la media. Posteriormente se analiza la representación espacial de los eventos sobre la huella del MEA para estudiar posibles relaciones espaciales.

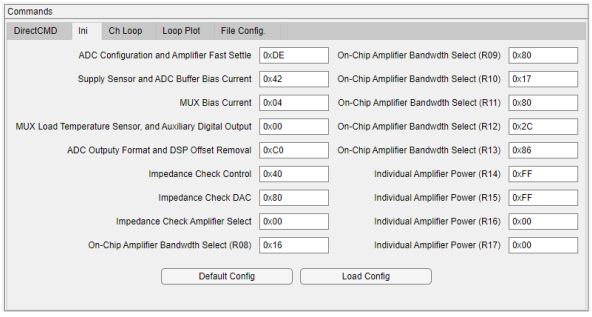
Finalmente, se exportan los datos en formato **.mat*. El fichero resultante contiene dos variables que almacenan los datos de la captura original, en formato *LabData*, y los datos generados a partir del procesado de la aplicación: Eventos individuales, características de la señal y media de los eventos.

5. MEA-CTL: Comunicación con el sistema

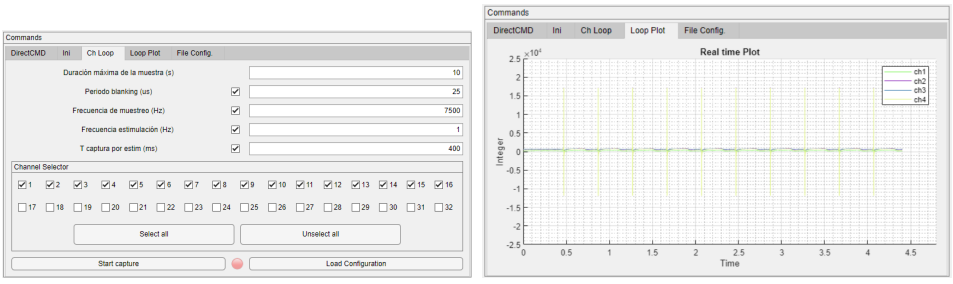
La interfaz por linea de comandos implementada permite verificar el sistema de comandos así como trabajar directamente con la plataforma. Sin embargo, requiere del conocimiento previo de la sintaxis de los comandos y el cálculo de los argumentos de estos. Para mitigar estas desventajas, se propone una interfaz gráfica que permita formar los comandos a enviar de una manera sencilla e interactiva.



(a)



(b)



(c) (d)

Figura 5.1: Pantalla principales de la aplicación de control: (a) Ventana inicial y envío de comando directo, (b) Inicialización, (c) Configuración bucle de conversión, (d)

La pantalla de la aplicación se divide en tres paneles desde donde se controlan todas las acciones de la aplicación y se monitorizan sus resultados: (i) El panel de conexiones es utilizado para realizar la conexión con las dos interfaces de la plataforma, tanto para la conexión de control como para la que transmite los datos; (ii) El panel de control de comandos implementa la interfaz por línea de comandos, muestra en un terminal toda la información recibida por el puerto de control y además dispone de un campo que permite introducir los comandos a enviar manualmente; y (iii) el panel de comandos.

El panel principal de la aplicación se encuentra en el panel de comandos (iii). Desde este panel es posible configurar los comandos a enviar de una forma intuitiva y rápida, además se implementa una vista preliminar de la señal capturada "en tiempo real" y la exportación de los datos capturados. La interfaz realiza la traducción de las opciones elegidas por el usuario y los envía hacia la plataforma.

Las dos primeras ventanas (*DirectCMD, Ini*) están ideadas para la configuración de los comandos homónimos de la plataforma. Al utilizar estos paneles, la aplicación solo permite modificar los valores necesarios para el comando y, en caso de encontrarse fuera de los rangos válidos o no ser introducido por el usuario, utilizar la configuración por defecto de la plataforma. Esto presenta una mejora frente a la interfaz por línea de comandos en la que no se realiza ninguna comprobación sobre los valores introducidos.

El control de la función principal de la plataforma se encuentra en las dos ventanas siguientes, *Ch Loop* y *Loop Plot*. Desde la primera ventana se realiza la configuración del bucle de captura. A la hora de cargar la configuración del bucle, es posible hacerlo variando todos los parámetros o, si no es necesario, es posible variar solo uno manteniendo en los otros la anterior configuración. Tras cargar la configuración, la aplicación calcula si los parámetros introducidos producirán errores en la transmisión, como se comenta en el capítulo 3, en caso de producirlos avisará al usuario y no dejará enviar el comando de inicio del bucle hasta que se soluciones.

Durante el tiempo en el que el bucle se encuentra activo, aparecen mensajes cada nuevo estímulo en el terminal y un mensaje al finalizar las estimulaciones marcadas al igual que el la interfaz por linea de comandos. Además, en esta ventana se mantiene encendido un indicador durante todo el proceso de captura y almacenamiento de los datos. El tiempo necesario para el almacenamiento de los datos es mayor que la transmisión de los mismos por lo que, reducir el trabajo de la aplicación, se recomienda no iniciar otra captura hasta que el proceso de almacenamiento de los datos haya finalizado. En la ventana *Loop plot* se representa una previsualización de los datos recibidos, esta se actualiza a medida que los datos se almacenan.

Los datos adquiridos se almacenan en una variable de matlab tipo *struct* que se guarda en un archivo binario propio de matlab, *.mat*. Como se puede ver en la figura 5.2 en el primer nivel se encuentra un campo con la información general sobre la captura, como la resolución del conversor o información del programa. También almacena un campo para cada captura realizada en la misma ejecución de la aplicación, denominadas *Records*. El campo de información general es común a todas las capturas del fichero.

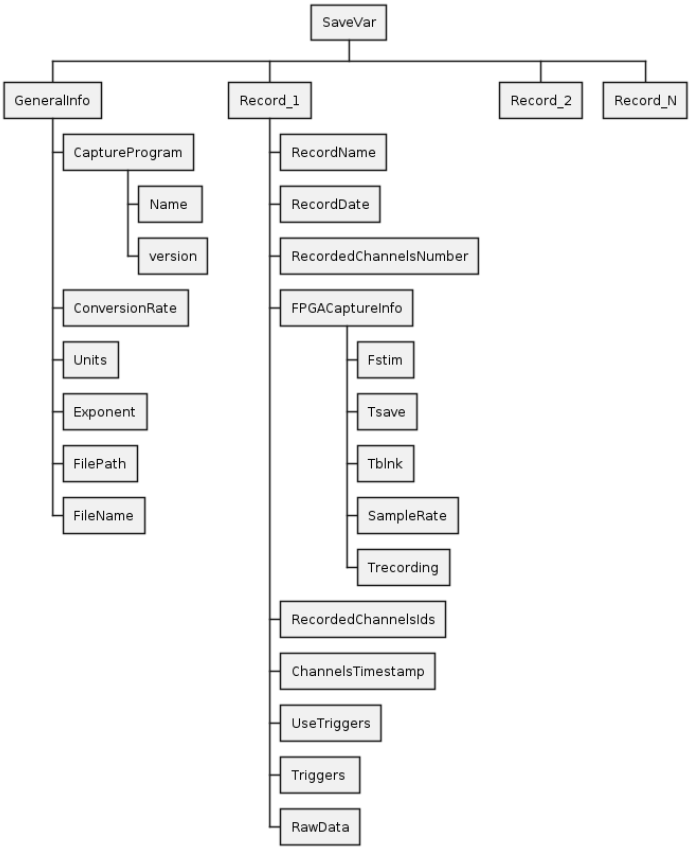


Figura 5.2: Estructura de la variable resultante del proceso de captura en bucle al usar la interfaz.

Cada *Record* alberga varios campos dedicados a información sobre la captura realizada, como puede ser el número de canales capturados y sus Ids, o los parámetros utilizados para la captura almacenados en una estructura. Finalmente se incluyen los campos de datos: *Rawdata*, que contiene los valores capturados; *ChannelsTimestamp* almacena el eje de tiempos final de la señal; y el valor temporal y posición de los eventos de trigger, se guarda en *Triggers*.

La plataforma solo envía al ordenador los datos adquiridos de cada canal y de manera continua, sin tener en cuenta que en estos datos existen saltos entre capturas. Esto implica que tanto el eje de tiempos como la posición de los eventos tengan que ser generados en el ordenador a partir de la configuración introducida por el usuario para

el bucle de conversión. La creación del eje de tiempos resulta trivial conociendo la duración total de la captura y la frecuencia de muestreo. En el caso de los eventos, conociendo la frecuencia de estimulación y la duración total de la captura se puede estimar el número de eventos que tendrán lugar, sumado a la estimación del número de muestras por estimulación, es posible calcular la posición de todos los eventos.

Como se explica en el capítulo 3, los datos se envían hacia el ordenador en bloques de muestras precedidas por una cabecera. Además, cada muestra está formada por dos palabras de 8 bits. Es trabajo de la interfaz interpretar estos datos y detectar la cabecera. La aplicación lee los datos del *buffer* en grupos del número de canales más la cabecera. Cuando detecta la cabecera en el primer canal significa que las siguientes muestras han llegado correctamente y se asignan al canal en cuestión. Si se detecta la cabecera en otra posición, ha ocurrido un error en la transmisión y se rellenan los canales restantes con *padding*. También se habilita una opción que nos permite eliminar el nivel de continua que aparece al capturar la señal.

6. Validación del sistema

Una vez diseñada la plataforma es necesaria su validación antes de poder realizar medidas experimentales sobre tejido vivo. A excepción de los electrodos utilizados por la plataforma, el resto del sistema no había sido validado sobre señales reales, por lo que se ha implementado un banco de pruebas para verificar el comportamiento de las etapas de conversión, procesado y las interfaces gráficas. Además, los resultados obtenidos se han comparado con otras plataformas comerciales para valorar las prestaciones del sistema.

6.1. Banco de pruebas

Un banco de pruebas permite realizar las validaciones en un entorno controlado y sin la dependencia de factores externos, como puede ser el estado de las muestras biológicas de tejido. El trabajo con tejido vivo *ex-vivo* presenta dificultades debido a su corta vida útil, la disponibilidad de las muestras y la diferente respuesta en función de su estado entre otras. Además, son necesarios conocimientos previos para poder preparar la muestras antes de la adquisición.

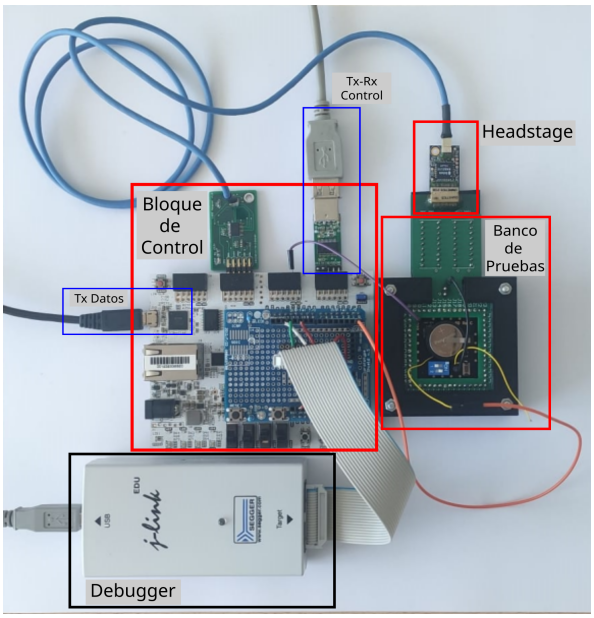


Figura 6.1: Montaje utilizado durante la validación donde se identifican los distintos bloques que formal la plataforma y el entorno de pruebas. Se marcan ambas interfaces de la plataforma (rectángulo azul).

El montaje utilizado se muestra en la figura 6.1, en la que se marcan todos los componentes utilizados. Durante la validación se ha hecho uso de un *debugger* de la marca *Segger* para realizar los cambios necesarios sobre el *firmware*. Se ha hecho uso de un ordenador para realizar el control, al que se conectan las interfaces de datos, control y el *debugger*.

6.1.1. Generación de señales

Por todas las complicaciones que suponen las muestras biológicas, se deciden sustituir por un generador de señales de muy bajo voltaje. El modelo elegido es el *60MEA-SG*[18] de la empresa MCS, preparado para validar sus sistemas de adquisición de señales.

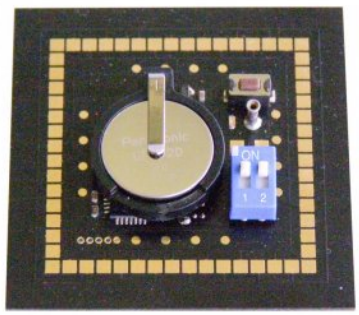


Figura 6.2: Generador de señales *60MEA-SG*. Extraída de MultiChannelSystems.

El generador ofrece 60 canales por los que, a través de los pads que se observan en la figura 6.2, simultáneamente se obtiene la señal generada en dos amplitudes distintas. Es capaz de generar una gran variedad de señales biológicas, como ECG y FP de cardiomiocitos. Para el segundo tipo de señal, muy similar a la señal objetivo de la plataforma, se tiene una amplitud en el pico de depolarización de $\pm 3{,}1mV$ en los canales con mayor amplitud y $\pm 1{,}7mV$ en el resto. Los estímulos se generan con una frecuencia de 1Hz.

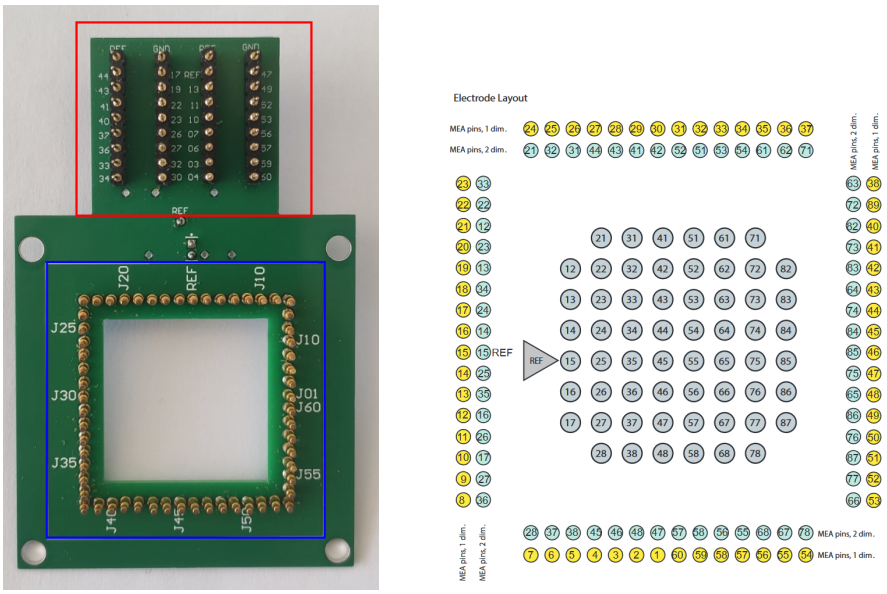
En los experimentos a realizar con la plataforma en el entorno del hospital se utilizará un estimulador para forzar la repuesta de los tejidos. Este estimulador genera una señal que se detecta en la plataforma como se ha explicado en la sección 2.2.3. Para emular el funcionamiento del estimulador, se ha estudiado el circuito del generador y se ha

encontrado una señal digital que activa el comienzo de cada muestra, que se conecta a la plataforma.

6.1.2. Conexión generador-plataforma

Al trabajar con señales de baja amplitud, uno de los mayores problemas presentados es el efecto del ruido. El ruido introducido sobre la señal debido a malas conexiones, longitud de cables u otras fuentes puede superar el voltaje de la señal y volverla irrecuperable.

Para mitigar este efecto se ha diseñado una placa de circuito impreso que permite la conexión entre el generador de señales y el headstage que contiene el chip del Intan y donde se realiza la conversión. Se han seleccionado 32 pads del generador que serán los que se conecten con los 32 canales del headstage del RHD2000. Para esta selección se ha utilizado como referencia la posición de los electrodos en un MEA de la marca MCS y, con el objetivo de cubrir la mayor área sensada se seleccionan los electrodos siguiendo un patrón ajedrezado con modificaciones. En la figura 6.3 se incluye la pcb diseñada y la distribución estándar de los electrodos que se utiliza como referencia.



(a) (b)

Figura 6.3: a) Fotografía de la pcb diseñada para interconectar el generador con el headstage. La conexión con el generador se marca con un cuadrado azul y la conexión con el *headstage* se marca con un rectángulo rojo. b) Disposición de los electrodos en un MEA estándar de 60 electrodos [19] fabricado por *MCS*.

Para el diseño de la pcb se han utilizado cuatro capas, donde las dos exteriores conforman planos de masa en su totalidad y a través de las dos interiores se conectan

todos los canales. Esta estructura de capas es robusta frente al posible ruido exterior [20, p .641]. Para el routeo de las pistas a lo largo de las capas interiores se aplica el criterio de maximizar la distancia entre pistas adyacentes, minimizando de esta forma el *crosstalk* entre canales. El resto de las capas interiores se rellenan con plano de masa.

Para garantizar una correcta conexión con el generador se hace uso de pines con resorte, que junto a un soporte modelado en *FreeCad* y fabricado mediante impresión 3D permite una conexión estable con una mayor tolerancia al ruido.

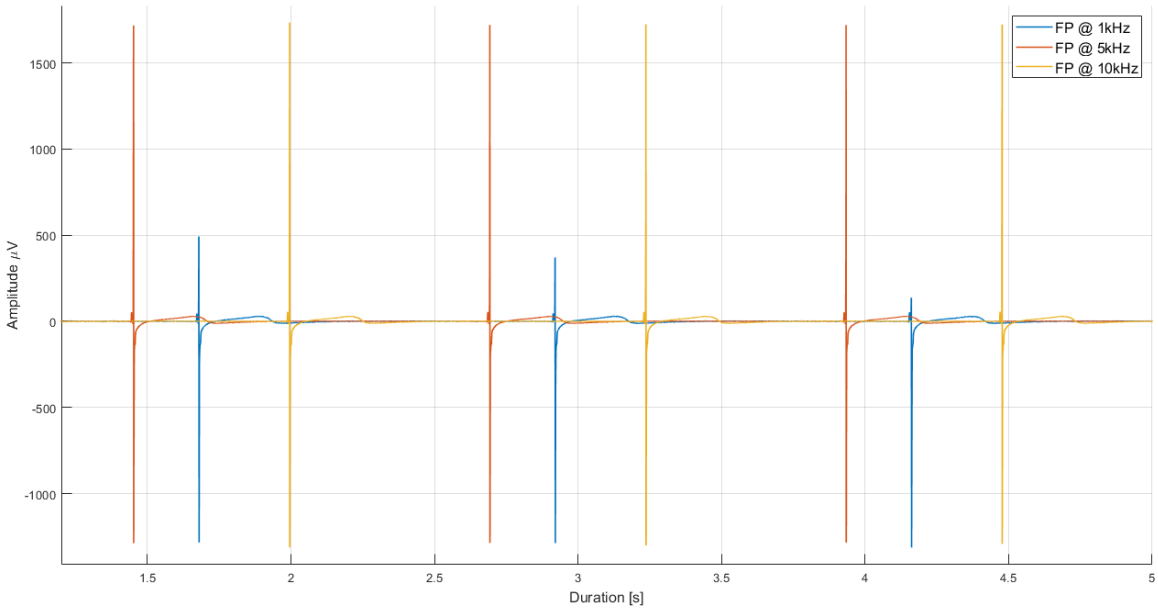
6.2. Resultados de la validación

En esta sección se evalúan los resultados obtenidos con el banco de pruebas desarrollado. Además, los datos adquiridos permitirán validar las decisiones de diseño del sistema, así como marcar las siguientes líneas de desarrollo de la plataforma. Como referencia, se utilizan las medidas obtenidas con el equipo de la marca MCS. Se comienza estudiando los parámetros de la plataforma a la hora de capturar la señal para posteriormente realizar la validación.

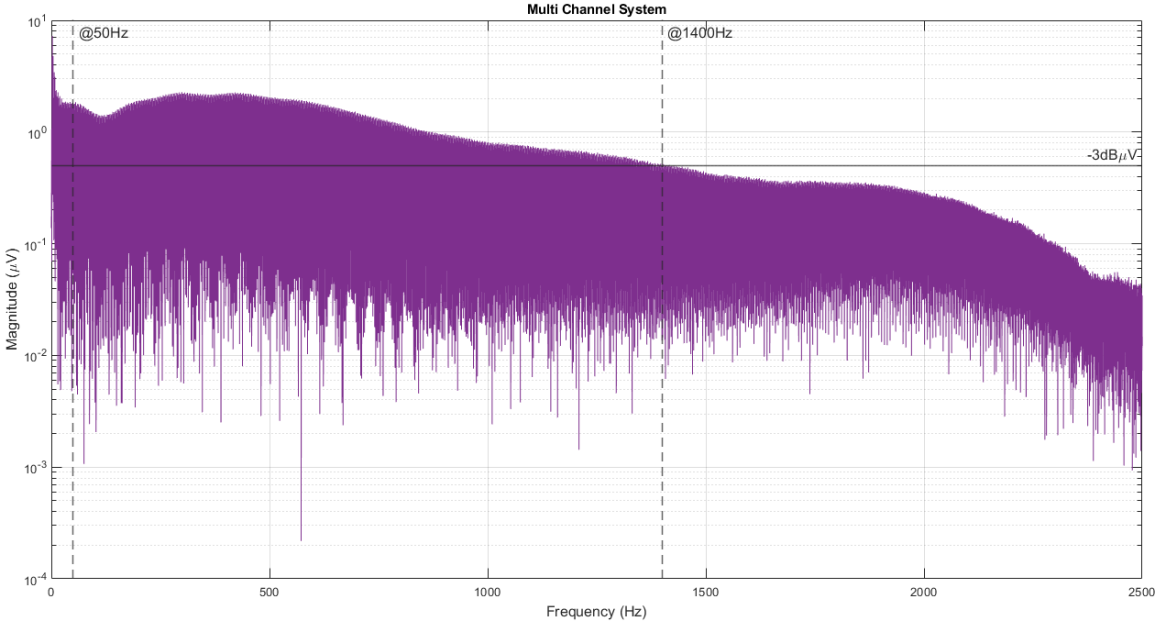
Uno de los parámetros a estimar antes de la validación de la plataforma es la frecuencia de muestreo mínima a la que el sistema captura la señal correctamente. Los equipos comerciales, como se detalla en el capítulo 2 permiten trabajar hasta los 30-50kSamples/s, muy por encima de los requisitos necesarios para la señal proveniente de tejidos cardiacos [21].

Utilizando el equipo de MCS, se realiza un barrido frecuencial entre 1kHz, 5kHz y 10kHz, figura 6.4, para estudiar su efecto. A una $F_s = 1kHz$ encontramos distorsión en la señal capturada. Al observar el barrido en el dominio frecuencial podemos encontrar que la señal capturada presenta contenido hasta los 1400Hz.

Al fijar la frecuencia máxima de la señal en torno a los 1400Hz, la plataforma deberá trabajar con frecuencias mayores de 2.8kHz para cumplir el criterio de Nyquist. Se decide fijar una frecuencia de muestreo por defecto de $F_s = 7{,}5kHz$ aunque se realizan pruebas para trabajar a $F_s = 10kHz$.



(a)



(b)

Figura 6.4: a) Representación temporal para distintas frecuencias de muestreo: 1kHz, 5kHz y 10kHz. b) representación frecuencial de la señal capturada a 10kHz.

Los parámetros F_{est} y T_{blk} vienen fijados por el estimulador utilizado en el experimento. En el caso del banco de pruebas, estimamos la frecuencia de estimulación en torno a 1Hz. En cuanto al periodo de *blanking* se toma un valor estimado de $25\mu s$. El máximo número de canales a capturar se decide fijar en 16, número de electrodos del MEA de

diseño propio [9].

Para la estimación del periodo de captura en un inicio se recurre a la literatura, al no disponer de muestras reales. La duración del potencial de campo (FPD) se estima en unos 250ms [2], con lo que T_{save} debe ser mayor que este valor. Al analizar los eventos de la señal capturada con el MCS se estima que la señal presenta información hasta los 400ms, tras los que la señal toma un valor de reposo.

Utilizando los valores anteriores, es posible dimensionar la tasa de transmisión por defecto de la plataforma, evitando así errores en la transmisión. Siguiendo la inecuación 3.1, $R_{Tx} \geq 1{,}02Mb/s$ en caso de capturar 16 canales. Se ha validado el funcionamiento de la interfaz de transmisión para distintas combinaciones de los parámetros, estableciendo el límite de funcionamiento de la interfaz en una tasa de 3Mb/s, $F_s = 10kHz$. Para cumplir con los requisitos, se decide establecer la tasa de la transmisión en $R_{Tx} = 2Mb/s$, que nos permitiría capturar 16 canales a frecuencias superiores a la frecuencia por defecto o periodos de captura superiores.

Al comparar el resultado capturado con la plataforma y la señal capturada con el equipo de MCS, figura 6.5, se observar un nivel de continua en la captura de la plataforma. Este nivel aparece en todos los canales, variando su valor. Al realizar la captura con el Open Ephys, que utiliza el mismo *headstage*, encontramos el mismo nivel de continua, por lo que puede deberse al diseño del banco de pruebas. En el sistema presentado, para la eliminación de la componente de continua se implementa la sustracción de la media de cada canal capturado en la interfaz de control, capítulo 5. Se opta por esta opción debido a su simplicidad frente a otras opciones como un filtrado paso alto con frecuencia de corte en torno a 0.5Hz.

Tras la eliminación de la componente de continua y volver a comparar la señal de la plataforma con la del equipo MCS, figura 6.5, es posible apreciar diferencias en la fase de despolarización, la parte con una gran variación, en la que se encuentra una menor amplitud que la captura del MCS. Sin embargo, en la fase de repolarización apenas se aprecian variaciones significativas. Esta diferencia en la zona de la señal que presenta rápidas variaciones puede deberse a efectos parásitos en la pcb encargada de la conexión con el generador, ya que aparece un efecto similar en las señales capturadas con el Open Ephys al utilizar el mismo banco de pruebas.

Se han realizado las validaciones de las interfaces de usuario, tanto de control como de procesado. La validación de la interfaz de control, MEA-CTL: Comunicación con el sistema, se ha realizado de manera simultanea a la validación de la transmi-

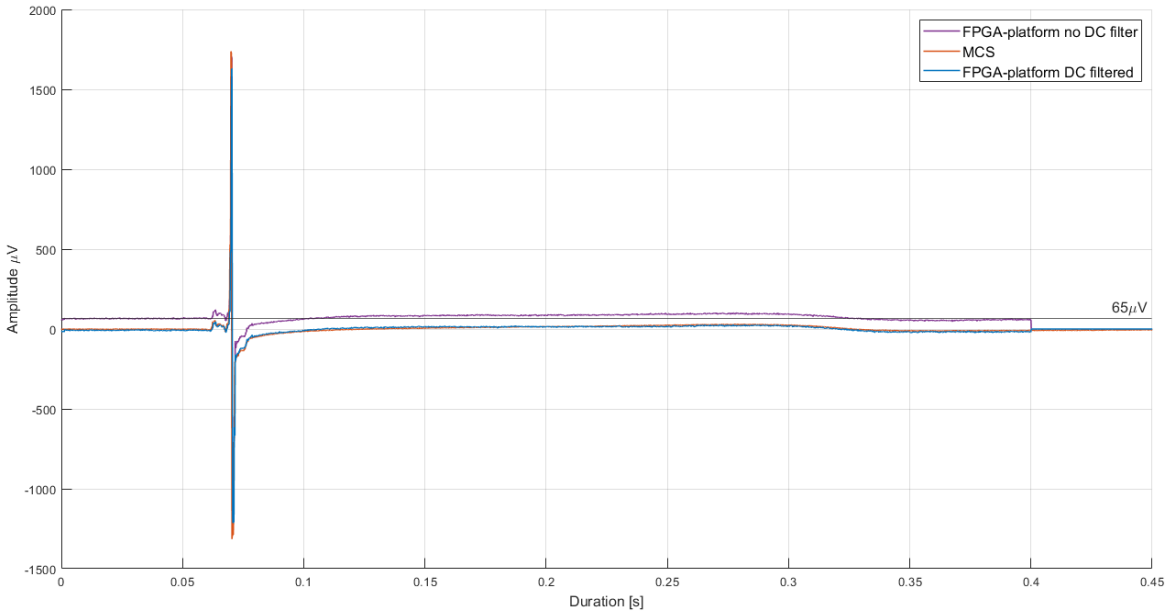


Figura 6.5: Comparación temporal de un evento capturado por el equipo de MCS y el mismo evento capturado por la plataforma antes y después de eliminar la componente de continua.

sión de los datos. Para la validación de la interfaz de procesado, MEA-GUI: Procesado de los datos, se realizan pruebas con archivos de todos los sistemas de aquisición de señales nombrados anteriormente. Durante estas pruebas se validan todas las funciones que ofrece la aplicación. En especial se validan de una forma más exhaustiva las funciones de separación de eventos, donde se evalúa utilizando eventos externos y extrayendo el evento de la propia señal, en la función de extracción de características y la representación espacial de los eventos.

7. Conclusiones y trabajo futuro

7.1. Conclusión

En este trabajo se han cumplido todos los objetivos propuestos. Se han implementado las mejoras propuestas en cuanto a la transmisión de los datos adquiridos por la plataforma. Se ha separado la comunicación utilizada para el control de la plataforma de la transmisión de los datos capturados. Esto permite transmitir datos y comandos de forma concurrente y con distintas especificaciones en cada comunicación. Estas mejoras permiten de manera directa mayores prestaciones a la hora de configurar el bloque de conversión de canales de la plataforma. También se ha implementado una interfaz por línea de comandos así como un diccionario de comandos que permiten configurar, de una forma fácil, todos los aspectos de la plataforma desde un terminal.

Se ha conseguido mejorar la interacción del usuario con la plataforma por medio de dos interfaces gráficas.

- La interfaz encargada del control permite la comunicación con ambos puertos de la plataforma, permitiendo configurar todos los parámetros de la plataforma y lanzar la captura. La conexión a través de la interfaz de datos permite recibir el resultado de la captura, almacenarlo y realizar una representación para verificar la correcta adquisición.
- Por otro lado, la interfaz desarrollada para el procesado permite trabajar con datos de distintos equipos comerciales y con la plataforma propia. Desde esta interfaz es posible realizar representaciones temporales de los canales individuales, de los eventos que tienen lugar durante una captura y permite la representación simultánea de los canales guardando la distribución espacial de los electrodos sobre el tejido. También permite la extracción de las características de la señal de una forma rápida y la exportación de los datos en un formato común para un futuro procesado más detallado en Matlab.

Finalmente se ha diseñado e implementado un pequeño entorno de pruebas para validar todas las configuraciones de la plataforma, realizando las modificaciones necesarias para su correcto funcionamiento.

A lo largo de todo el proyecto se ha mantenido la idea de facilitar al usuario final el

uso de la plataforma. Con el enfoque de la filosofía *open source* se favorece la interoperabilidad con distintos equipos. Esto deja la puerta abierta a futuras modificaciones o nuevas funcionalidades en la plataforma, expandiendo sus capacidades en función del experimento a llevar a cabo.

7.2. Trabajo futuro

A modo de trabajo futuro, es posible mantener la plataforma en constante mejora. La forma de desarrollo, siguiendo una estructura por bloques, facilita la implementación de nuevas mejoras y modificaciones. Sobre esta línea, dotar al sistema de una nueva forma de almacenamiento, haciendo uso de las memorias RAM de las que dispone la placa, agilizaría la transmisión de los datos. Otra línea de mejoras se centraría en la inclusión de etapas de filtrado implementadas directamente sobre el hardware de la FPGA y configurables por el usuario desde el ordenador.

En la aplicación de control sería posible aumentar la velocidad y las prestaciones de la comunicación, migrando la interfaz a otras tecnologías que permitan trabajar con esas características o utilizando una forma distinta para el almacenamiento de la información, permitiendo así reducir el tiempo de procesado de la aplicación. Sería posible también la mejora de la vista en tiempo real de la señal, acelerando la representación.

La aplicación de procesado puede ampliarse, permitiendo la extracción y análisis de otros parámetros de interés de la señal, y evitando así la necesidad de realizar un segundo procesado desde otra herramienta. Otra línea de mejoras buscaría implementar nuevas formas de representación haciendo uso de la distribución espacial de los electrodos que forman el MEA, como por ejemplo la representación de mapas de calor, generación de patrones de propagación del potencial de campo.

Finalmente, para validar completamente la plataforma, sería necesario sustituir el banco de pruebas por muestras de tejido cardiaco y realizar una serie de medidas que permitan medir el desempeño de la plataforma en un entorno real.

Bibliografía

[1] J. M. De Bakker, "Electrogram recording and analyzing techniques to optimize selection of target sites for ablation of cardiac arrhythmias," *Pacing and Clinical Electrophysiology*, vol. 42, no. 12, pp. 1503–1516, Dec. 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/pace.13817

[2] P. Camelliti, S. A. Al-Saud, R. T. Smolenski, S. Al-Ayoubi, A. Bussek, E. Wettwer, N. R. Banner, C. T. Bowles, M. H. Yacoub, and C. M. Terracciano, "Adult human heart slices are a multicellular system suitable for electrophysiological and pharmacological studies," *Journal of Molecular and Cellular Cardiology*, vol. 51, no. 3, pp. 390–398, Sep. 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0022282811002574

[3] S. Gaeta, T. D. Bahnson, and C. Henriquez, "High-Resolution Measurement of Local Activation Time Differences From Bipolar Electrogram Amplitude," *Frontiers in Physiology*, vol. 12, p. 653645, Apr. 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fphys.2021.653645/full

[4] S. Silbernagl and A. Despopoulos, *Color Atlas of Physiology*. Thieme Medical Publishers, 1 2015.

[5] "MEA2100-Mini-Systems | www.multichannelsystems.com." [Online]. Available: https://www.multichannelsystems.com/products/mea2100-mini-systems

[6] J. H. Siegle, A. C. López, Y. A. Patel, K. Abramov, S. Ohayon, and J. Voigts, "Open Ephys: an open-source, plugin-based platform for multichannel electrophysiology," *Journal of Neural Engineering*, vol. 14, no. 4, p. 045003, Aug. 2017. [Online]. Available: https://iopscience.iop.org/article/10.1088/1741-2552/aa5eea

[7] "Check out ECP5 / ECP5-5G on www.latticesemi.com! http://t.cn/RXUHWlF." [Online]. Available: https://www.latticesemi.com/Products/FPGAandCPLD/ECP5#_11D625E1D2C7406C96A5312C93FF0CBD

[8] "Intan Technologies." [Online]. Available: https://www.intantech.com

[9] A. Velarte Iranzo, A. Otín Acín, and I. Urriza Parroqué, "Diseño de una plataforma para la adquisición de potenciales de acción de muestras de tejido cardíaco con una matriz de electrodos," 2021.

[10] A. J. Castel Santolaria and I. Urriza Parroqué, "Diseño de un periférico AXI-4 Lite para la adquisición de señales electro-fisiológicas con el integrado INTAN RHD22xx," 2023.

[11] "FT2232HQ - FTDI," Aug. 2020. [Online]. Available: https://ftdichip.com/products/ft2232hq/,https://ftdichip.com/products/ft2232hq/

[12] SCPI Consortium, "SCPI Syntax and Sytle," 1999. [Online]. Available: https://www.ivifoundation.org/downloads/SCPI/scpi-99.pdf

[13] U. Egert, T. Knott, C. Schwarz, M. Nawrot, A. Brandt, S. Rotter, and M. Diesmann, "MEA-Tools: an open source toolbox for the analysis of multi-electrode data with matlab," *Journal of Neuroscience Methods*, vol. 117, no. 1, pp. 33–42, May 2002. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0165027002000456

[14] V. P. Pastore, D. Poli, A. Godjoski, S. Martinoia, and P. Massobrio, "ToolConnect: A Functional Connectivity Toolbox for In vitro Networks," *Frontiers in Neuroinformatics*, vol. 10, Mar. 2016, publisher: Frontiers. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fninf.2016.00013

[15] V. Georgiadis, A. Stephanou, P. A. Townsend, and T. R. Jackson, "MultiElec: A MATLAB Based Application for MEA Data Analysis," *PLOS ONE*, vol. 10, no. 6, p. e0129389, Jun. 2015. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0129389

[16] R. M. Dastgheyb, S.-W. Yoo, and N. J. Haughey, "MEAnalyzer – a Spike Train Analysis Tool for Multi Electrode Arrays," *Neuroinformatics*, vol. 18, no. 1, pp. 163–179, Jan. 2020. [Online]. Available: http://link.springer.com/10.1007/s12021-019-09431-0

[17] P. Pradhapan, J. Kuusela, J. Viik, K. Aalto-Setälä, and J. Hyttinen, "Cardiomyocyte MEA Data Analysis (CardioMDA) – A Novel Field Potential Data Analysis Software for Pluripotent Stem Cell Derived Cardiomyocytes," *PLoS ONE*, vol. 8, no. 9, p. e73637, Sep. 2013. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3777951/

[18] "60MEA-SG | www.multichannelsystems.com." [Online]. Available: https://www.multichannelsystems.com/products/60mea-sg

[19] "MCS_60standardmea_layout." [Online]. Available: https://www.multichannelsystems.com/sites/multichannelsystems.com/files/MCS_60StandardMEA_Layout.pdf

[20] H. W. Ott, *Electromagnetic Compatibility Engineering*.

[21] L. D. Garma, L. Matino, G. Melle, F. Moia, F. De Angelis, F. Santoro, and M. Dipalo, "Cost-effective and multifunctional acquisition system for in vitro electrophysiological investigations with multi-electrode arrays," *PLOS ONE*, vol. 14, no. 3, p. e0214017, Mar. 2019. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0214017

Índice de figuras

Índice de tablas

Lista de acrónimos

AP Action Potential

FP Field Potential

FPD Field Potential Duration)

MEA Micro Electrode Array

SPI Serial Peripheral Interface

LVDS Low Voltage Differential Signaling

FPGA Field Programable Gate Array

GPIO General Purpose Input/Output

ADC Analog-Digital Converter

USB Universal Serial Bus

UART Universal Asynchronous Receiver-Transmitter

IP Intelectual Property

TTL Transistor-Transistor Logic

FIFO First In First Out

SCPI Standard Commands for Programmable Instruments

ECG Electrocardiogram

PCB Printed Circuit Board

RAM Random Access Memory

Anexos

A continuación se incluyen como anexos al trabajo presentado los siguientes documentos:

A. Documentación del periférico de control del *headstage*, del Sistema de desarrollo propio, actualizado con las nuevas funcionalidades.

B. Documentación de la clase *LabData* utilizada en MEA-GUI: Procesado de los datos.

A. Periférico para comunicación AXI4-Lite - Intan RHD2000

Introducción

El periférico permite una comunicación con el circuito integrado *INTAN RHD 2000* por medio de la interfaz serie (SPI). Permite la comunicación para su control y acceso a sus registros por medio de la interfaz AXI4-Lite de la especificación AMBA (*Advance Microcontroller Bus Architecture*). Incluye también una interfaz UART para el envío de los datos capturados.

El periférico descrito en este documento está pensado para su utilización junto al circuito integrado *INTAN RHD 2000*[1]. El datasheet del circuito integrado es referenciado a lo largo del documento y amplía la información del funcionamiento de la comunicación y del mapa de memoria.

Características

- Interfaz AXI4-Lite para el acceso a los registros y control del periférico.
- Interfaz SPI para la comunicación con el integrado *INTAN RHD2000*.
- Interfaz UART para el envío de los datos capturados en el proceso de adquisición. Envío de palabras de 16 bits en formato big-endian.
- Proceso de inicialización del integrado predefinido.
- Proceso de conversión de los canales seleccionados, preparado para muestreo continuo.
- Permite la comunicación directa con el circuito integrado.
- Generación de interrupciones de control del periférico e interrupción por *Overrun* en la transmisión.

1. Descripción general

Los bloques del periférico se muestran en la siguiente figura y son descritos a continuación.

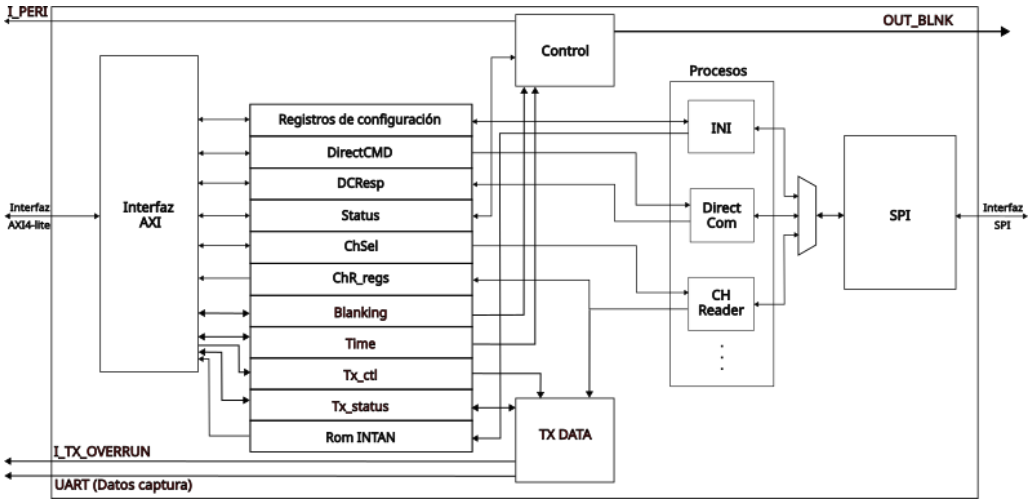


Figura 1-1: Diagrama de bloques del periférico

- **Interfaz AXI:** El bloque implementa la interfaz esclava de AXI4-Lite [2] [2]para el acceso a los registros.
- **Registros:** Este bloque está formado por 64 registros de 32 bits utilizados para el control del periférico y de los distintos procesos que implementa. Los registros siguen la distribución del mapa de memoria del *RHD2000* y almacenan una copia de la memoria de este. El mapa de los registros se detalla en la sección Descripción de los registros.
- **Control:** Este bloque se encarga del control y la gestión del periférico. Se controla por medio de la escritura en determinados bits del registro *Status*. Es el encargado de la generación de la interrupción *I_PERI*, control de los distintos procesos y de la gestión del bucle de adquisición.
- **Procesos:** Este bloque engloba los distintos procesos que ejecuta el periférico:
    - **INI** → Bloque encargado del proceso de inicialización.
    - **Direct com** → Bloque encargado de la transmisión y recepción de comandos y respuestas de manera directa a través de la interfaz SPI.
    - **CH Reader** → Bloque encargado del proceso de conversión de los canales seleccionados.
- **SPI:** El bloque implementa la interfaz SPI [3] maestra para la comunicación con el circuito integrado *RHD2000*.
- **TX DATA**: Implementa el transmisor de una interfaz UART y una FIFO de 65.536 palabras de 16 bits actuando como *buffer* de transmisión. Solo permite el envío de los datos. La comunicación utiliza un bit de stop, sin paridad. Además genera una interrupción en caso de *Overrun* en la transmisión.

Resumen de las características

El periférico presenta las siguientes características:

- Realiza el proceso de inicialización de manera independiente a partir de la configuración indicada en los registros por medio de la interfaz AXI4-Lite. Realiza también una calibración y copia de los registros del *RHD2000.*
- Presenta un proceso de conversión de canales seleccionados configurable y con capacidad para realizarlo de manera periódica con un periodo configurable.
- Genera 1 interrupción desde el control del periférico. para la notificación de la recepción de respuestas y para el final del bucle de conversión.
- Genera 1 interrupción en caso de *Overrun* en la transmisión.
- Genera Señal que marca el periodo de *blanking*.
- La distribución de los registros mantiene la estructura de la memoria del *RHD2000*
- El periférico trabaja con una frecuencia de reloj igual a $f_{clk} = 96\ MHz$.
- La interfaz serie trabaja a una frecuencia igual a $f_{sclk} = 24\ MHz$.
- La interfaz UART tiene tasa configurable, desde 1Mb/s hasta 12Mb/s. Transmisión de palabras de 16 bits en formato big-endian. 1 bit de Stop sin paridad.

2. Especificaciones

Descripción de los puertos

Los puertos del periférico se describen en la Tabla 2-1.

Tabla 2-1: Descripción de señales I/O

Nombre de la señal	Interfaz	I/O	Estado inicial	Descripción
Señales SPI				
MISO	SPI	I	-	*Master Input Slave Output*. Recepción de datos. (Esclavo → Maestro)
MOSI	SPI	O	0	*Master Output Slave input.* Transmisión de datos. (Maestro → Esclavo)
SCLK	SPI	O	0	Salida del reloj ($f_{sclk} = 24\ MHz$) de la interfaz Serie. (Maestro → Esclavo)
CS	SPI	O	1	Chip Select, SPI. (Maestro → Esclavo)
Señales de la interfaz AXI4-Lite				
S00_axi_*	S_AXI	-	-	Consultar el documento "AXI reference guide (UG761)" [3][4] para una descripción completa de las señales de la interfaz AXI4.
Señales de la interfaz UART				
TX	UART	O	1	Salida de la interfaz UART de envío de los datos del bucle de adquisición.
I_TX_OVERRUN	Interrupción	O	0	Señal de interrupción en caso de que la FIFO de transmisión se llene durante un bucle de adquisición.
Señales del sistema				
s_axi_aclk	Sistema	I	-	Reloj de la interfaz AXI. Utilizado como reloj del periférico. Frecuencia de trabajo: $f_{clk} = 96MHz$
s_axi_arestn	Sistema	I	-	Reset AXI, activo en bajo.
I_PERI	Interrupción	O	0	Señal de interrupción de control del periférico. Engloba la interrupción por nueva respuesta y la finalización del proceso *ChRead.*
OUT_BLNK	Control	O	0	Señal que se mantiene activa a 1 durante el periodo de *blanking* del proceso *ChRead.*

Descripción de los registros

La Tabla 2-2 muestra todos los registros del periférico, su dirección, tipo de acceso y la descripción de su función.

Tabla 2-2: Mapa de direcciones de los registros

Dirección	Nombre del registro	Tipo de acceso	Descripción
0x00 – 0x44	Registros de configuración	R/W	Registros de configuración del INTAN RHD22xx.
0x48	DirectCMD	W	Comando transmisión directa hacia el Integrado.
0x4C	DCResp	R	Respuesta directa del Integrado al comando enviado.
0x50	Status	R/W	Registro de estado y control del periférico.
0x54	ChSel	R/W	Selector de canales para la conversión en bucle.
0x58	ChR_reg0	R	Repuesta de la conversión de los canales 0 y 1.
0x5C	ChR_reg1	R	Repuesta de la conversión de los canales 2 y 3.
0x60	ChR_reg2	R	Repuesta de la conversión de los canales 4 y 5.
0x64	ChR_reg3	R	Repuesta de la conversión de los canales 6 y 7.
0x68	ChR_reg4	R	Repuesta de la conversión de los canales 8 y 9.
0x6C	ChR_reg5	R	Repuesta de la conversión de los canales 10 y 11.
0x70	ChR_reg6	R	Repuesta de la conversión de los canales 12 y 13.
0x74	ChR_reg7	R	Repuesta de la conversión de los canales 14 y 15.
0x78	ChR_reg8	R	Repuesta de la conversión de los canales 16 y 17.
0x7C	ChR_reg9	R	Repuesta de la conversión de los canales 18 y 19.
0x80	ChR_reg10	R	Repuesta de la conversión de los canales 20 y 21.
0x84	ChR_reg11	R	Repuesta de la conversión de los canales 22 y 23.
0x88	ChR_reg12	R	Repuesta de la conversión de los canales 24 y 25.
0x8C	ChR_reg13	R	Repuesta de la conversión de los canales 26 y 27.
0x90	ChR_reg14	R	Repuesta de la conversión de los canales 28 y 29.
0x94	ChR_reg15	R	Repuesta de la conversión de los canales 30 y 31.
0x98 – 0x9C	Reservado	N/A	
0xA0 – 0xB0	INTAN	R	Registro de solo lectura para datos del integrado.
0xB4	s_blnk	R/W	Tiempo de *blanking* en el bucle de adquisición.
0xB8	S_time	R/W	Periodo de muestreo del bucle de adquisición en No. De ciclos de reloj
0xBC	TX_CTL	W	Control del bloque de transmisión de los datos (UART)
0xC0	TX_STATUS	R/W	Estado y control del bloque de transmisión de los datos (UART)
0xC0 – 0xEC	Reservado	N/A	
0xF0 – 0xFC	INTAN ROM	R	Registros de solo lectura para datos del integrado.

Registros de configuración

Los registros de configuración mantienen el mismo orden que los registros de configuración del INTAN RHD22xx. Ver "INTAN RHD2000 series datasheet"[1] para una descripción completa de estos registros. La información de estos registros se copia al registro correspondiente durante el proceso de inicialización y, al finalizar, se sobrescriben con el contenido de los registros del *RHD22xx.*

Registro de comando con prioridad (DirectCMD)

Sobre este registro se escribe el comando recibido desde la interfaz AXI4-Lite, a la espera de enviarlo hacia el circuito integrado por la interfaz SPI, sin ser procesado en el periférico. Es un registro de escritura y lectura.

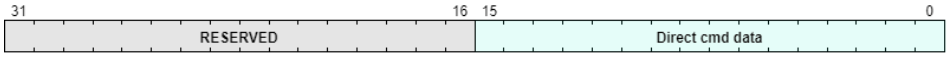


Figura 2-1: DirectCMD (@0x48, bits de datos = 16).

Registro de respuesta al comando con prioridad (DCResp)

Este registro de sólo lectura contiene la respuesta inmediata del *INTAN RHD22xx* obtenida tras el envío del comando escrito en el registro *DirectCMD*. Tiene como valor por defecto es 0x00.

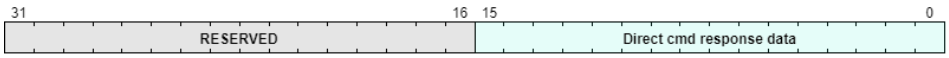


Figura 2-2: DCResp (@0x4C, bits de datos = 16).

Registro de estado y control (Status)

Este registro contiene la información del estado del periférico y la escritura en algunos bits permite configurar e iniciar los distintos procesos: inicialización, comando directo y bucle de conversión. Registro de Escritura y lectura.

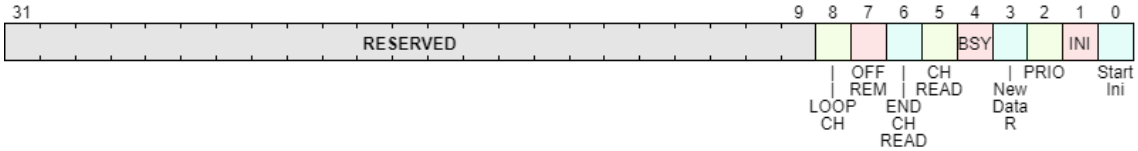


Figura 2-3: Status (@0x50)

En la Tabla 2-3 se describen los bits del registro.

Tabla 2-3: Definición de los bits del registro de estado y control

Bits	Nombre	Tipo de acceso	Valor inicial	Descripción
0	START_INI	Write	0h	Escribir 1 da inicio al proceso de inicialización. Este bit se limpia de manera automática tras su escritura. 0 = Reposo 1 = comenzar inicialización
1	INI	Read	0h	Indica que el proceso de inicialización está corriendo. 0 = Proceso de inicialización desactivado 1 = Proceso de inicialización corriendo
2	PRIORITY (PRIO)	Write	0h	Escribir 1 comienza el proceso de comunicación directa, transmite el contenido del registro *DIRECTCMD*. Se limpia de manera automática al comenzar el proceso. 0 = Reposo 1 = Enviar comando con prioridad

3	NEW_DATA_R	Read / Write	0h	Indica la recepción de una respuesta al comando enviado con prioridad [1] . Es necesario poner a 0 el campo una vez se ha leído el registro *DCResp*. Conectado a la interrupción *I_NPD* 0 = No hay respuesta nueva 1 = Nueva respuesta recibida
4	BSY	Read	0h	Indica si el periférico se encuentra procesando o puede aceptar nuevos procesos. 0 = Reposo 1 = Periférico ocupado
5	CH_READ	Write	0h	Escribir 1 comienza el proceso de conversión de los canales seleccionados en el registro ChSel. Este bit se limpia automáticamente al comenzar el proceso. 0 = Reposo 1 = Comienzo proceso de conversión de canales
6	END_CH_READ	Read / Write	0h	Indica la finalización del proceso de conversión de canales y la disposición de nuevos datos. Es necesario poner a 0 el bit una vez se hayan leído los nuevos datos de los registros. Conectado a la interrupción I_CHL. 0 = no hay nuevos datos disponibles 1 = fin de la conversión y nuevos datos disponibles
7	OFF_REM	Write	0h	Indica si el proceso CH READ va a realizarse aplicando Offset Removal[2]. Esto se aplicará en todos los canales selecionados en ChSel. 0 = Offest Removal desactivado 1 = Offest Removal activado
8	LOOP_CH	Write	0h	Escribir 1 indica el comienzo del bucle de conversión de canales periódico. Escribir 0 detiene el bucle. 0 = Bucle detenido 1 = Bucle de conversión de canales iniciado
15-31	RESERVED	N/A	-	Reservado

[1] Se entiende por respuesta inmediata los bits recibidos por la entrada MISO del SPI durante la transmisión del comando con prioridad. Los datos recibidos no contienen la respuesta al comando enviado de forma inmediata. Para más información ir al *"INTAN RHD2000 series datasheet"*[1].

[2] Para más información sobre la conversión de canales y sobre el Offset removal consultar la "INTAN RHD2000 series datasheet"[1].

Registro de selección de canales (ChSel)

El registro de selección de canales, *ChSel*, es un registro de escritura que permite seleccionar los canales que se quieren convertir durante el bucle de conversión de canales. Escribir a 1 un bit en el registro implica que el canal que coincide con la posición del bit se convertirá (i.e: bit 0 corresponde al canal 0).

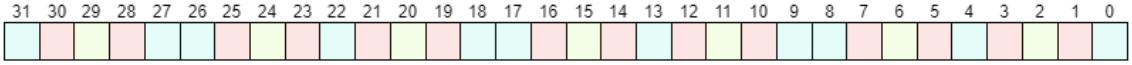


Figura 2-4: ChSel (@0x54, cada bit corresponde al canal con el mismo número).

Registros de conversión de canales (ChR_reg0 – ChR_reg 15)

Los registros de conversión de canales contienen el resultado de realizar la conversión de los canales indicados en el registro *ChSel*. Son registros de sólo lectura, la escritura en estos registros no tiene efecto. Todos los registros denominados *ChR_regN*, con N de 0 a 15 siguen la misma estructura: Contienen el contenido de 2 canales, los bits [31:16] corresponden al canal impar y los bits [15:0] al canal par.

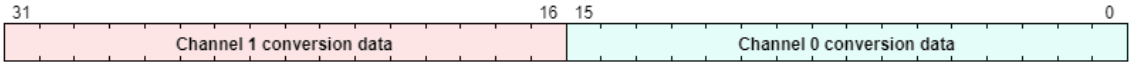


Figura 2-5: ChR_regN (@0x58-@0x94)

Registro de tiempo de blanking

Este registro permite configurar el tiempo inicial del bucle de adquisición. Durante este periodo el periférico se mantiene en reposo esperando para comenzar la captura.

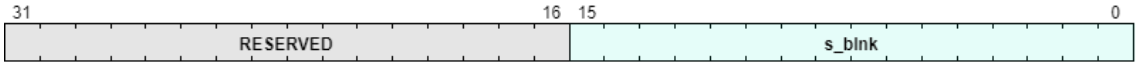


Figura 2-6: s_blnk(@0xB4)

Registros de tiempo de captura

Este registro permite la configuración del periodo de captura del bucle de adquisición. Este periodo indica el tiempo entre activaciones de la conversión de canales, *ChRead*, cuando el bucle de adquisición, *ChLoop* se encuentra activo.

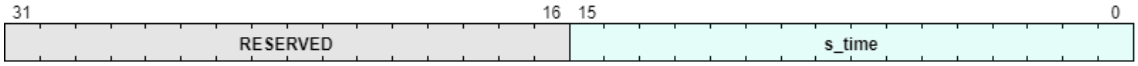


Figura 2-7: s_time(@0xB8)

Registro de control del bloque Tx DATA

Registro encargado de controlar el bloque de transmisión de los datos. Los datos se generan durante el bucle de adquisición. Es un registro de lectura y escritura, en el que los bits habilitados vuelven a su valor por defecto en el siguiente ciclo de reloj.

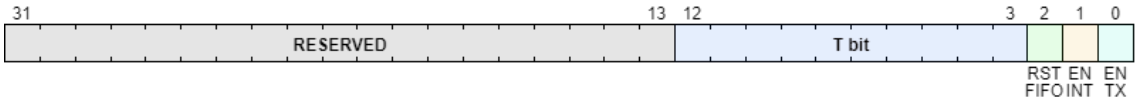


Figura 2-8: TX_CTL(@0xBC)

En la siguiente tabla, Tabla 2-4, se describen las acciones de los bits.

Tabla 2-4: Descripción de los bits del registro TX_CTL

Bits	Nombre	Tipo de acceso	Valor inicial	Descripción
0	EN_TX	W	0h	Bit que habilita o deshabilita el bloque de transmisión de los datos. Para cambiar el estado se escribe un '1' en el bit. Este vuelve a '0' en el siguiente ciclo de reloj. 0 = Reposo 1 = Habilitar / deshabilitar el bloque
1	EN_INT	W	0h	Bit que permite habilitar/deshabilitar generación de interrupciones por el bloque TX_DATA. Para cambiar el estado se escribe un '1' en el bit. Este vuelve a '0' en el siguiente ciclo de reloj. 0 = Reposo 1 = Habilitar / deshabilitar la generación de interrupciones.
2	RST FIFO	W	0h	Bit que permite resetear la FIFO que actúa como *buffer* de salida en el bloque TX DATA. Para activar el RESET se escribe un '1' en el bit. Este vuelve a '0' en el siguiente ciclo de reloj. 0 = Reposo 1 = Habilitar / deshabilitar el bloque
3-12	T BIT	W/R	0x60	Periodo de un bit transmitido por el bloque TX DATA en ciclos de reloj. Toma como valor por defecto 0x60, que representa una tasa de transmisión de 1Mb/s
13-31	RESERVED	NA/A		Reservado

Registro de estado del bloque Tx Data

Este registro de solo lectura permite ver comprobar el estado del bloque de transmisión, TX DATA. Los cambios realizados en el registro Tx_CTL se traducen en cambios sobre este registro. La interrupción I_TX_OVERRUN se conecta a un bit de este registro.

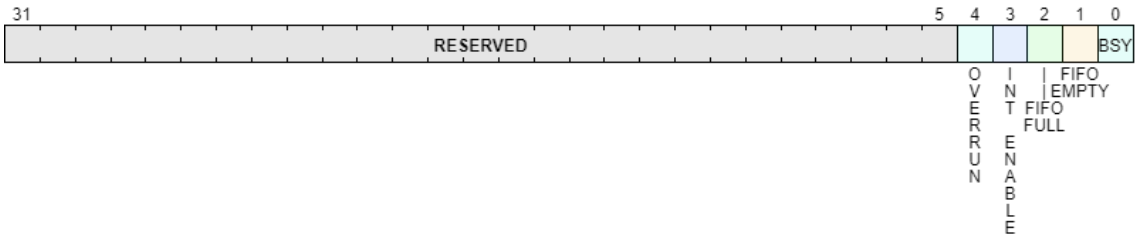


Figura 2-9: TX_Status(@0xC0)

En la siguiente tabla, Tabla 2-5, se describe cada bit.

Tabla 2-5: Descripción de los bits del registro TX_Status

Bits	Nombre	Tipo de acceso	Valor inicial	Descripción
0	BSY	R	0h	Bit indica si el bloque de Transmisión se encuentra ocupado transmitiendo datos o en reposo. 0 = Reposo 1 = Ocupado
1	FIFO EMPTY	R	0h	Indica si la FIFO de transmisión se encuentra vacía. 0 = FIFO contiene datos 1 = FIFO vacía
2	FIFO FULL	R	0h	Indica si la FIFO de transmisión se ha llenado. 0 = Espacio disponible en la FIFO 1 = FIFO Llena
3	INT_ENABLE	R	0h	Muestra si se ha habilitado la generación de interrupciones por parte del bloque de transmisión. El bit cambia de valor al escribir un 1 en el bit correspondiente del registro TX_CTL. 0 = Generación de interrupciones deshabilitada 1 = Generación de interrupciones habilitada
4-31	RESERVED	N/A		Reservado

Registros de solo lectura (INTAN, INTAN ROM)

Los registros de sólo lectura, *INTAN* e *INTAN ROM*, son registros que mantienen la misma dirección que los registros en el circuito integrado. Durante el proceso de inicialización copian el contenido de los registros del *INTAN RHD22xx* y no se modifican. Para más información sobre el contenido de los registros ver "INTAN RHD2000 series datasheet"[1].

Procesos

En los siguientes apartados se detallan los procesos que contiene el periférico.

Comunicación Directa

Este proceso permite el envío de comandos directamente al *RHD22xx*, sin ser modificados por el periférico. El comando a enviar se almacena en el registro *DirectCMD (@0x48)* para su envío.

La respuesta inmediata, que entendemos por la respuesta que se recibe por el puerto *MISO* durante la transmisión del comando, se almacena en el registro *DCResp (@0x4C)* para su lectura.

El proceso genera una interrupción para avisar de la llegada de la nueva respuesta, también activa el bit *NEW_DATA_R* en el registro *Status (@0x50)*.

Inicialización

Este proceso se encarga de la inicialización del *RHD2000*. Esta basado en el proceso descrito en la hoja de características del *RHD2000*[1]. Solo debe ejecutarse una vez al comienzo de la comunicación.

Antes de lanzar el proceso, es necesario escribir en los registros de configuración *(@0x00 – @0x44)* la configuración deseada para el *RHD22xx*.

Este proceso realiza las siguientes funciones:

1. Inicio de la comunicación.
2. Escritura de los registros de configuración.
3. Calibración del ADC del *RHD22xx*.
4. Copia del contenido de todos los registros del *RHD22xx* a los registros del periférico.

Durante el tiempo que dura el proceso, el bit *INI* del registro *Status (@0x50)* se mantiene a nivel alto.

Conversión de selección de canales

El proceso se encarga de la conversión de una serie de canales definidos por el usuario en el registro *CHSel (@0x54)*.

El resultado del proceso se almacena en los registros indicados para ello.

El proceso ofrece la posibilidad del eliminar el offset asociado a los amplificadores analógicos de los canales muestreados en todos los canales seleccionados por el usuario. Para más información sobre el "offset removal" ver "INTAN RHD2000 series datasheet"[1].

Al terminar cada proceso de conversión se genera una interrupción y se activa el bit *END_CH_LOOP* en el registro *Status (@0x50)*.

Bucle de adquisición (*CH Loop*)

Es posible realizar la captura de los canales seleccionados de manera periódica. Para configurar esta opción es necesario indicar en el registro *s_time (@0xB8)* el periodo de muestreo de la señal, es decir, el periodo con el que se activará un nuevo proceso *ChRead*.

Es necesario definir también el tiempo de *blanking* del bucle de conversión, periodo inicial del bucle de adquisición durante el que no se realiza captura. Este periodo se define, en ciclos de reloj, en el registro *s_blnk (@0xB4)* y es utilizado para evitar la adquisición de señales espurias. A modo de ejemplo, se incluye la figura x, en la que se detalla el periodo de blanking, T_blk.

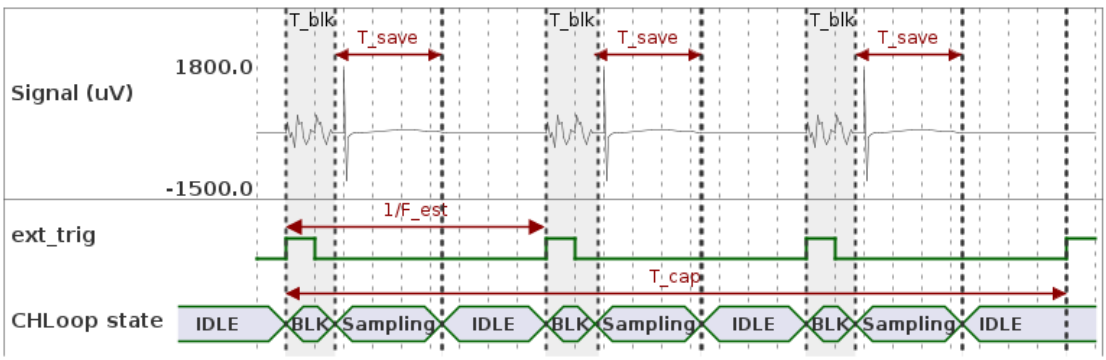


Figura 2-10: Ejemplo en el que se observa el funcionamiento del tiempo de blanking

Simultáneamente, los datos se envían a través de la interfaz UART del bloque TX DATA. con la frecuencia de trabajo definida en los bits asignados para ello en el registro *TX_CTL(@0xBC).* Los datos se envían en paquetes formados por los resultados del proceso *ChRead* ejecutado. El paquete va precedido por la cabecera de 16 bits: “0110011011001100”. Los datos se organizan en el paquete según el número del canal al que pertenecen, siendo los de menor numeración los más cercanos a la cabecera. En la siguiente figura se muestra un ejemplo de los paquetes transmitidos.

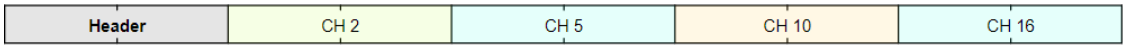


Figura 2-11: Ejemplo de paquete transmitido por el bloque TX Data

Recursos utilizados

El periférico utiliza los siguientes recursos medidos en una FPGA Artix-7.

Recursos del dispositivo	Slices	Slice Flip-Flops	LUTs
Recursos utilizados	802	2169	1174

3. Guía de diseño

Este capítulo incluye información adicional y guías para facilitar el diseño y trabajo con el periférico.

Relojes

El periférico opera con el reloj en *s_axi_aclk*. La frecuencia de trabajo del periférico es igual a $f_{clk} = 96\ MHz$. Como la frecuencia difiere de la del reloj interno de las FPGA, se recomienda el uso de un MMCM [4] [5]para su generación.

El INTAN RHD22xx opera con el reloj del SPI, *SCLK*. La frecuencia de este reloj es igual a $f_{sclk} = 24\ MHz$.

Reset

El periférico utiliza como reset la señal *s_axi_aresetn*. Es una señal activa en bajo síncrona al reloj, *s_axi_aclk*. El reset se realiza de manera síncrona.

Flujo de trabajo

Para controlar el periférico se programan los bits deseados en el registro *Status*.

Para realizar el proceso de inicialización es necesario escribir primero la configuración deseada en los registros de configuración[3]. Durante este proceso el periférico envía el contenido de estos registros al circuito integrado y realiza la calibración. Además, se copia el contenido de los registros del circuito integrado a los registros del periférico. Al finalizar este proceso, se recomienda la lectura y verificación de los registros del periférico, para comprobar la correcta inicialización del *RHD22xx*. Durante este proceso se sobrescriben los registros de configuración (@0x00 - @0x44), la comparación entre el contenido de los registros y la configuración inicial permite comprobar si hay errores en el proceso. En la Tabla 3-1 se muestra un ejemplo de configuración del proceso de inicialización.

Tabla 3-1:Ejemplo de configuración y uso del proceso de inicialización

READ(@0x50)	Leer registro *Status* y comprobar que el bit *BSY = 0*
WRITE(@0x00, confReg0)	Escribir la configuración en el registro de configuración 0
WRITE(@0x04, confReg1)	Escribir la configuración en el registro de configuración 1
...	
WRITE(@0x44, confReg17)	Escribir la configuración en el registro de configuración 17
WRITE(@0x5C, START_INI = 1)	Escribir el bit *START_INI* del registro *Status* para comenzar el proceso
READ(@0x50)	Leer registro *Status* y comprobar que el bit *INI = 1* para indicar que el proceso ha comenzado

Puede escribir los comandos con prioridad en el registro *DirectCMD (@0x48)* y enviarlos modificando el registro de control. Los datos con prioridad recibidos se pueden leer en el registro *DCResp (@0x4C)* cuando se activa la señal correspondiente. En la siguiente tabla, Tabla 3-2, se muestra el proceso a seguir para el envío de este tipo de comandos.

[3] El proceso de inicialización implementado sigue la estructura propuesta en "INTAN RHD2000 series datasheet"[1].

Tabla 3-2: Ejemplo de uso del proceso de comandos con prioridad

READ(@0x50)	Leer registro *Status* y comprobar que el bit *BSY = 0.*
WRITE(@0x48, cmd)	Escribir el comando a enviar en el registro *DirectCMD.*
WRITE(@0x50, PRIORITY = 1)	Escribir el bit PRIORITY en el registro *Status* para comenzar el proceso.
READ(@0x50)	Leer el registro *Status* hasta que el bit *NEW_DATA_R* = 1.
READ(@0x4C)	Leer la respuesta al comando en el registro *DCResp.*
WRITE(@0x50, NEW_DATA_R = 0)	Escribir el bit *NEW_DATA_R* = 0 en el registro *Status* para indicar haber leído el nuevo dato.

Puede realizar la conversión de varios canales eligiéndolos en el registro *ChSel* e iniciando el proceso. Al finalizar la conversión de los canales, el resultado se almacena en los registros *ChR_reg* y son transmitidos a través de la interfaz UART por el bloque *TX_DATA.* Para realizar esta conversión de manera periódica, es necesario cargar la frecuencia de muestro y el tiempo de *blanking* en sus registros correspondientes, *s_time (@0xB8)* y *s_blnk (@0xB4)* respectivamente. En la siguiente tabla, Tabla 3-3, se muestra un ejemplo de uso del comando, asi como el proceso para ejecutarlo correctamente.

Tabla 3-3: Ejemplo de uso y configuración del proceso de conversión de canales seleccionados

READ(@0x50)	Leer registro *Status* y comprobar que el bit *BSY = 0*
WRITE(@0x54, chsel)	Escribir en el registro *ChSel* la selección de canales a convertir
WRITE(@0x5C, OFF_REM)[4]	Escribir el bit *OFF_REM* del registro *Status* con la opción deseada.
WRITE(@0xB8, LOOP_TIMER)	Escribir en el registro *s_time* la frecuencia de muestreo.
WRITE(@0xB4, BLANKING)	Escribir en el registro *s_blnk* el tiempo de blanking.
WRITE(@0x5C, LOOP_CH = 1)	Escribir el bit *LOOP_CH* = 1 del registro *Status* para iniciar la conversión de manera periódica.
WRITE(@0x5C, CH_READ = 1)[5]	Escribir el bit CH_READ = 1 del registro *Status* para iniciar la conversión de canales una sola vez.
READ(@0x5C)	Leer el registro *Status* hasta que el bit *END_CH = 1*, indicando que la conversión ha finalizado.
READ(ChR_reg)	Leer los registros que contienen la respuesta a la conversión de los canales deseados
WRITE(@0x5C, END_CH_READ = 0)	Escribir el bit *END_CH_READ = 0* en el registro *Status* para indicar que ya se han leído los resultados de la conversión.

[4] En función de si se desea utilizar o no está opción, la instrucción se deberá ejecutar o no.

[5] Este comando **solo** ha de ejecutarse cuando la conversión de canales no se realice de manera periódica, utilizando el temporizador del periférico. En ese caso los dos comandos anteriores **no hay que ejecutarlos**.

4. Referencias

Los siguientes documentos contienen material complementario al presentado en este documento y permiten profundizar en el funcionamiento del periférico.

[1] Intan Technologies, «Intan_RHD2000_series_datasheet». 2022.

[2] «AMBA AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite», 2003.

[3] «AN-1248: SPI Interface | Analog Devices». Accedido: 25 de junio de 2024. [En línea]. Disponible en: https://www.analog.com/en/resources/app-notes/an-1248.html

[4] Xilinx, «AXI Reference Guide». 2012. Accedido: 25 de junio de 2024. [En línea]. Disponible en: https://docs.amd.com/api/khub/documents/1N~rJgeEMU28Fyv7kWcW6Q/content?Ft-Calling-App=ft%2Fturnkey-portal&Ft-Calling-App-Version=4.3.48#

[5] Xilinx, «Mixed-Mode Clock Module (MMC) Manager». 2009. Accedido: 25 de junio de 2024. [En línea]. Disponible en: https://docs.amd.com/api/khub/documents/QprfWqQW~ClXP3i918KEVA/content?Ft-Calling-App=ft%2Fturnkey-portal&Ft-Calling-App-Version=4.3.48#

B. Documentación de clase *LabData*

La clase *LabData* se diseña como un formato estándar para trabajar con los datos provenientes de capturas de señales electrofisiológicas. La clase puede trabajar con capturas con equipos de adquisición de señales electrofisiológicas de la marca *MCS* y *Open Ephys*, así como de la plataforma de diseño propio. Realiza un proceso de normalización, en el que se convierte la información de los formatos originales de las plataformas de adquisición a la misma clase, independiente de la plataforma utilizada para la captura.

B.1. Propiedades de la clase

B.1.1. Propiedades privadas

- *Standard2DIDs*: Matriz de 10x6 tipo double. Almacena los números de canal asociados a las ID de cada electrodo dentro del array ([columna fila]). Utilizado al capturar con equipos de *MCS*
- *Pcb1DIDs*: Matriz de 8x4 tipo double. Almacena la ID de cada canal, en numeración de 1 dimensión, y utilizando la pcb diseñada para la captura.
- *Pcb2DIDs*: Matriz de 8x4 tipo double. Almacena la ID de cada electrodo dentro del array ([columna fila]) al capturar con la pcb diseñada.
- *ElectrodeType*: String. Tipo de MEA que se utiliza en la captura. "half" si utiliza 16 canales, "ful" en caso de utilizar los 60.

B.1.2. Propiedades públicas

- *RawChannelsRecords*: Matriz de n Canales x m Muestras de tipo integer. Almacena los datos de la captura sin realizar la conversión.
- *ChannelsTimestamps*: Vector de 1xm Muestras, tipo double. Almacena el eje de tiempos, común para todos los canales.
- *ChannelsSampleNumbers*: Double. Muestra inicial del Record.
- *UseTriggers*: Boolean. Indica si la captura ha registrado los *triggers*.

- *TriggerSampleNumber*: Vector de 1xm Muestras, tipo double. Cada valor almacenado indica la muestra en la que se registra un *trigger*. En caso de no registrar trigger toma valor *empty*.
- *TriggerTimestamps*: Vector de 1xm Muestras, tipo double. Cada valor almacenado indica el valor del eje de tiempos en el que se registra un *trigger*. En caso de no registrar *trigger* toma valor *empty*.
- *ConvertedChannelsRecords*: Matriz de n canales x m muestras, tipo double. contiene los valores de las muestras de los canales capturados convertidos a la unidad de referencia.
- *BitsVolts*: Double. Resolución del conversor AD de la plataforma utilizada para la captura.
- *Unit*: String. Caracteres que indican la unidad de los datos convertidos.
- *Exponent*: Double. Indica el exponente asociado a la unidad de los datos convertidos.
- *ReferenceID*: Double. ID del canal en el que se registra la referencia.
- *ChannelsID*: Vector 1xn canales, tipo double. Almacena las IDs en orden, asociadas a los canales capturados.
- *RecordName*: String. Nombre de la captura que se ha convertido a *LabData*.
- *RecordDate*: String. Fecha de la captura realizada.
- *RecordPath*: String. Camino de la señal hasta la captura. Solo cuando se captura con el Open Ephys toma un valor, de lo contrario toma valor *empty*
- *RecordChannelsNumber*: Double. Número de canales capturados.
- *SampleRate*: Double. Frecuencia de muestreo utilizada en la captura.
- *FilePath*: String. Ubicación de la captura.
- *CaptureProgram*: Struct. Estructura que almacena dos campos: *ProgramName* (String), con el nombre del programa; y *ProgramVersion* (String), que almacena la versión.
- *StreamName*: String. Nombre de la captura. Solo cuando se captura con el Open Ephys toma un valor, de lo contrario toma valor *empty*.

B.2. Métodos

B.2.1. Métodos públicos

- *LoadFPGALabData(obj, FileStruct, recordNumber)*
  - Método que realiza la conversión de la captura de la plataforma a un objeto de tipo LabData.
  - Argumentos:
    - obj: LabData. Objeto de tipo LabData.
    - FileStruct: Struct. Estructura con los datos de la captura realizada con la plataforma.
    - recordNumber: Double. Número del record de la captura a traducir en la clase LabData.
- *LoadOELabData(obj, RecordNode)*
  - Método que realiza la conversión de la captura del Open Ephys a un objeto de tipo LabData.
  - Argumentos:
    - obj: LabData. Objeto de tipo LabData.
    - RecordNode: RecordNode. Objeto que contiene toda la información de la captura a convertir a LabData.
- *LoadMCSLabData(obj, Data)*
  - Método que realiza la conversión de la captura de equipos MCS a un objeto de tipo LabData.
  - Argumentos:
    - obj: LabData. Objeto de tipo LabData.
    - Data: DataMCS. Objeto que contiene toda la información de la captura a convertir a LabData.
- *GetRecordName(obj)*
  - Devuelve el nombre del Record.

- Argumentos:
  - obj: LabData. Objeto de tipo LabData.
- Return: String. Nombre del Record.

– *GetRecordDate(obj)*

- Devuelve la fecha de captura del Record.
- Argumentos:
  - obj: LabData. Objeto de tipo LabData.
- Return: String. Fecha de captura del Record.

– *GetChannelsNumber(obj)*

- Devuelve el número de canales capturados.
- Argumentos:
  - obj: LabData. Objeto de tipo LabData.
- Return: Double. Número de canales capturados.

– *GetChannelTimestamp(obj)*

- Devuelve el eje de tiempos de la captura.
- Argumentos:
  - obj: LabData. Objeto de tipo LabData.
- Return: Vector 1xn muestras. Eje de tiempos de la captura.

– *GetChannelRawData(obj, channelNumber)*

- Devuelve los datos capturados en el canal seleccionado sin conversión.
- Argumentos:
  - obj: LabData. Objeto de tipo LabData.
  - channelNumber: Double. Número del canal seleccionado.
- Return: Vector 1xn muestras, tipo integer. *RawData* del canal seleccionado.

– *GetChannelConvertedData(obj, channelNumber)*

- Devuelve los datos capturados en el canal seleccionado convertidos a la unidad.

- Argumentos:
  - obj: LabData. Objeto de tipo LabData.
  - channelNumber: Double. Número del canal seleccionado.
- Return: Vector 1xn muestras, tipo integer. *ConvertedData* del canal seleccionado.

– *GetTriggerTimestamps(obj)*

- Devuelve el eje de tiempos del *trigger*.
- Argumentos:
  - obj: LabData. Objeto de tipo LabData.
- Return: Vector 1xj triggers, tipo double. Eje de tiempos de los *triggers*.

– *GetTriggerSampleNumbers(obj)*

- Devuelve las muestras en las que aparecen los *triggers*.
- Argumentos:
  - obj: LabData. Objeto de tipo LabData.
- Return: Vector 1xj triggers, tipo double. Eje de tiempos de los *triggers*.

– *GetChannelsUnits(obj)*

- Devuelve las unidades de los datos convertidos.
- Argumentos:
  - obj: LabData. Objeto de tipo LabData.
- Return: String. Unidades de los datos capturados.

– *GetRecordPath(obj)*

- Devuelve el camino de la señal de la captura.
- Argumentos:
  - obj: LabData. Objeto de tipo LabData.
- Return: String. RecordPath de la captura.

– *GetSampleRate(obj)*

- Devuelve la frecuencia de muestreo.

- Argumentos:
    - obj: LabData. Objeto de tipo LabData.
- Return: Double. Frecuencia de muestreo.

– *GetFilePath(obj)*

- Devuelve la ubicación del Record.
- Argumentos:
    - obj: LabData. Objeto de tipo LabData.
- Return: String. Ubicación del archivo.

– *GetProgramInfo(obj)*

- Devuelve información sobre la plataforma utilizada para la captura.
- Argumentos:
    - obj: LabData. Objeto de tipo LabData.
- Return: Struct. Devuelve la estructura *ProgramInfo* de obj.

– *GetExponent(obj)*

- Devuelve el exponente asociado a la unidad de los datos convertidos.
- Argumentos:
    - obj: LabData. Objeto de tipo LabData.
- Return: Double. Exponente.

– *GetTriggerInfo(obj)*

- Devuelve si se utilizan triggers.
- Argumentos:
    - obj: LabData. Objeto de tipo LabData.
- Return: Boolean.

– *GetChannelsID(obj)*

- Devuelve las IDs de los canales capturados.
- Argumentos:

    - obj: LabData. Objeto de tipo LabData.
  - Return: Vector 1xn canales, tipo double.

- *Get2DChannelID(obj, OneDChannelID)*
  - Devuelve la ID del electrodo capturado en función de la del canal. Sobre el MEA completo.
  - Argumentos:
    - obj: LabData. Objeto de tipo LabData.
    - OneDChannelID. Double. ID del canal.
  - Return: Double. ID del electrodo.

- *Get1DChannelID(obj, TwoDChannelID)*
  - Devuelve la ID del canal capturado en función de la del electrodo. Sobre el MEA completo.
  - Argumentos:
    - obj: LabData. Objeto de tipo LabData.
    - TwoDChannelID. Double. ID del electrodo.
  - Return: Double. ID del canal.

- *GetPcbChannelID(obj, ID)*
  - Devuelve las ID del canal asociada al electrodo utilizando el MEA partido (*half*)
  - Argumentos:
    - obj: LabData. Objeto de tipo LabData.
    - ID: Double. ID del electrodo.
  - Return: Double. ID del canal.

- *GetChannelsSampleNumbers(obj)*
  - Devuelve el número de la muestra inicial.
  - Argumentos:
    - obj: LabData. Objeto de tipo LabData.

- Return: Double. Valor de la muestra inicial

B.2.2. Métodos privados

– *getCData(obj, Record)*

- Función utilizada para la extracción de los datos de una captura del Open Ephys.
- Argumentos:
  - obj: LabData. Objeto de tipo LabData.
  - Record: BinaryRecordings. Clase interna de la captura del Open Ephys
- Return: Struct. Estructura con formato del Open Ephys que contiene la información de la captura.

– *Get_Ch_Number(obj, CData)*

- Devuelve el número de canales que se capturan. Se utiliza con capturas del Open Ephys.
- Argumentos:
  - obj: LabData. Objeto de la clase LabData.
  - CData: Struct. Estructura con la información de la captura.
- Return: Double. Número de canales.

– *ConvertRawData(obj)*

- Devuelve el los datos de la captura convertidos. Se utiliza con datos capturados con el Open Ephys o con la plataforma de desarrollo propio.
- Argumentos:
  - obj: LabData. Objeto de tipo LabData.
- Return: Matriz n canales x m muestras, tipo Double. Datos convertidos

– *GetUnits(obj, units)*

- Devuelve las unidades de los canales.
- Argumentos:
  - obj: LabData. Objeto de tipo LabData.

- Return: Double. Valor de la muestra inicial

– *RecoverData(obj,originalData, Timestamp, EventsSampleNumber, Nblnk, Nsave)*

  - Realiza la reconstrucción de la señal capturada con la plataforma de diseño propio. Se añade relleno de ceros en los periodos donde la señal no se captura.
  - Argumentos:
    - obj: LabData. Objeto de la clase LabData.
    - originalData: Matriz n muestras x m canales, integer. Datos capturados.
    - Timestamp: Vector 1 x j muestras, Double. El vector tiene la longitud de la señal final.
    - EventsSampleNumber: Vector 1xi *triggers*, double. Posiciones de los *triggers* en la señal recuperada.
    - Nblnk: Double. Número de muestras de *blanking* antes de cada evento capturado.
    - Nsave: Double. Número de muestras capturadas en cada evento
  - Return: Matriz n canales x m muestras, integer. Señal reconstruida en todos los canales