DESARROLLO DE UNA API EN FORMATO DLL PARA LA GESTIÓN DEL CIRCUITO INTEGRADO ADS1298R EN APLICACIONES BIOELÉCTRICAS

ROBERTO JOSÉ CALDERÓN BERMEJO

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO
FACULTAD DE INGENIERÍA
INGENIERÍA DE SOFTWARE
MEDELLÍN
2025

DESARROLLO DE UNA API EN FORMATO DLL PARA LA GESTIÓN DEL CIRCUITO INTEGRADO ADS1298R EN APLICACIONES BIOELÉCTRICAS

ROBERTO JOSÉ CALDERÓN BERMEJO

Trabajo de grado para optar al título de Ingeniero de Software

Asesor Técnico
Diego Hernando Orozco Gómez
Magister en Ingeniería – Automatización Industrial

Asesor Metodológico
Bayardo Emilio Cadavid Gómez
Magister en Automatización y Control Industrial

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO
FACULTAD DE INGENIERÍA
INGENIERÍA DE SOFTWARE
MEDELLÍN
2025

Dedicatoria

A Dios, mi familia, amigos y compañeros, cuyos ánimos han sido la fortaleza del autor a lo largo de este camino.

Agradecimientos

Al ingeniero Diego Orozco, cuya disciplina y constancia durante casi dos años fueron fundamentales para abordar la complejidad del circuito integrado ADS1298. Asimismo, al ingeniero Bayardo Cadavid, por su valiosa orientación y motivación desde la asignatura de Control Discreto.

Contenido

Pág.

Lista de figuras

Pág.

Lista de anexos

Pág.

Resumen

DESARROLLO DE UNA API EN FORMATO DLL PARA LA GESTIÓN DEL CIRCUITO INTEGRADO ADS1298R EN APLICACIONES BIOELÉCTRICAS

ROBERTO JOSÉ CALDERÓN BERMEJO

Las disciplinas dedicadas al estudio de la actividad bioeléctrica requieren sistemas de adquisición capaces de registrar múltiples bioseñales con la precisión y confiabilidad necesaria para llevar a cabo investigaciones rigurosas con impacto social. En el mercado se encuentran diferentes dispositivos electrónicos pensados para este propósito, como es el caso del circuito integrado ADS1298R, el cual posee características técnicas excepcionales. Sin embargo, el software oficial proporcionado por la empresa Texas Instruments presenta limitaciones debido su carácter restrictivo, ausencia de actualizaciones recientes e incompatibilidad con sistemas operativos modernos como Windows 10 y la falta de una interfaz de programación abierta. Estas barreras dificultan su integración en aplicaciones avanzadas que utilizan lenguajes de alto nivel como Python o MATLAB.

Este trabajo se centró en desarrollar una interfaz de programación de aplicaciones (API) en formato biblioteca de vínculos dinámicos (DLL) compatible con el sistema operativo Windows 10 y Windows 11 en versión de 64 bits. Esta biblioteca es capaz de gestionar los registros del circuito integrado ADS1298R y transmitir tramas de datos hacia entornos de programación de alto nivel. Para lograr esto se utilizó una combinación del método experimental, el método de análisis y síntesis, y la aplicación de algunas técnicas provenientes de las metodologías ágiles del desarrollo de software. Además, se implementó el hardware necesario para la comunicación entre el circuito de adquisición de datos y el computador. Finalmente, se realizaron pruebas unitarias y de integración para asegurar el correcto funcionamiento de la API.

Palabras claves: ADS1298, API, C#, DLL, Python

Abstract

DEVELOPMENT OF A DESKTOP APPLICATION FOR COMMUNICATION WITH ADS1298RECGFE-PDK MODULE

ROBERTO JOSÉ CALDERÓN BERMEJO

Disciplines dedicated to the study of bioelectrical activity require acquisition systems capable of recording multiple biosignals with the precision and reliability necessary to carry out rigorous research with social impact. There are various electronic devices on the market designed for this purpose, such as the ADS1298R integrated circuit, which has exceptional technical characteristics. However, the official software provided by Texas Instruments has limitations due to its restrictive nature, lack of recent updates, incompatibility with modern operating systems such as Windows 10, and lack of an open programming interface. These barriers hinder its integration into advanced applications that use high-level languages such as Python or MATLAB.

This work focused on developing an application programming interface (API) in dynamic link library (DLL) format compatible with the Windows 10 operating system and higher in 64-bit version. This library is capable of managing the registers of the ADS1298R integrated circuit and transmitting data frames to high-level programming environments. To achieve this, a combination of the experimental method, the analysis and synthesis method, and the application of some techniques from agile software development methodologies were used. In addition, the necessary hardware for communication between the data acquisition circuit and the computer was implemented. Finally, unit and integration tests were performed to ensure the correct functioning of the API.

Keywords: ADS1298, API, C#, DLL, Python

Glosario

ADS1298R: circuito integrado especializado de adquisición analógica, diseñado por Texas Instruments para registrar bioseñales con alta precisión y múltiples canales.

API: interfaz de programación de aplicaciones que proporciona un conjunto estructurado de funciones para facilitar la comunicación entre hardware y aplicaciones informáticas.

Bioseñales: señales eléctricas generadas por procesos fisiológicos en organismos vivos, tales como las señales musculares captadas mediante electromiografía.

Consumo: cantidad de recursos utilizados por un sistema durante su funcionamiento, tanto a nivel local (uso de CPU, memoria RAM, ancho de banda o energía eléctrica) como en el aprovechamiento de servicios externos, donde se mide en términos de llamadas a la API, volumen de datos transferidos o cuotas de uso asignadas al usuario o aplicación.

DLL: biblioteca de vínculos dinámicos que permite encapsular funciones reutilizables para integrar fácilmente software con hardware específico en entornos Windows.

DllImport: atributo del lenguaje C# que permite declarar funciones externas ubicadas en bibliotecas compiladas (DLL), de modo que puedan ser invocadas desde código administrado como si fuesen métodos nativos del propio programa.

Error de regresión: falla de software que reaparece o se introduce nuevamente en una versión posterior, como consecuencia indirecta de modificaciones, correcciones o nuevas funcionalidades añadidas al código previamente probado.

Fachada API: patrón de diseño que agrupa y simplifica el acceso a múltiples funciones internas en una única interfaz de alto nivel, ocultando la complejidad del sistema subyacente y ofreciendo métodos más intuitivos para el usuario o desarrollador.

Ganancia programable: característica que permite ajustar electrónicamente el nivel de amplificación de una señal analógica antes de su digitalización, optimizando así la calidad y precisión del registro.

Latencia: tiempo total que tarda un dato desde ser capturado por el hardware hasta estar disponible para su visualización o procesamiento en el software.

Parser: componente de software encargado de leer y descomponer una secuencia de caracteres o bytes (por ejemplo, tramas de comunicación o archivos de configuración) en estructuras internas significativas, validando su formato según reglas previamente definidas.

Runtime: entorno de ejecución que proporciona los servicios básicos necesarios para que un programa funcione, tales como administración de memoria, manejo de excepciones y carga de bibliotecas, y que suele estar asociado a una máquina virtual o framework específico.

SPI: protocolo de comunicación digital síncrono utilizado para la transferencia rápida y confiable de datos entre dispositivos electrónicos, como sensores y microcontroladores.

Stakeholders: conjunto de personas o entidades interesadas o afectadas por un proyecto, incluyendo usuarios finales, desarrolladores, docentes, directivos y cualquier otro actor que tenga expectativas o responsabilidades frente a los resultados obtenidos.

Tasa de muestreo: frecuencia con la que un sistema adquiere muestras discretas de una señal analógica para convertirla en digital, medida generalmente dada en muestras por segundo (SPS).

TLV: esquema de codificación de datos basado en campos del tipo “Type-Length-Value”, en el que cada elemento se representa por un identificador de tipo, la longitud del contenido y el valor propiamente dicho, facilitando la interpretación estructurada de tramas en protocolos de comunicación.

Introducción

Una interfaz de programación de aplicaciones (API) es un conjunto estructurado de definiciones y protocolos que habilita la comunicación entre diferentes sistemas computacionales, y en algunos casos con dispositivos físicos, favoreciendo la interoperabilidad y la reutilización de código en entonos heterogéneos (Prongnuch & Wiangtong, 2016). En los sistemas operativos de Microsoft Windows las API que se encargan de crear puentes entre el hardware y el software se implementan mediante bibliotecas de vínculos dinámicos (DLL), las cuales encapsulan rutinas de bajo nivel en componentes reutilizables sin necesidad de recompilación, característica que agiliza el desarrollo y mejora la mantenibilidad (Dick & Volmar, 2018).

Sobre esta base, el presente trabajo busca la creación de una API implementada como DLL para comunicar de forma asertiva aplicaciones de análisis de datos y un dispositivo de adquisición de señales bioeléctricas. Estas señales son perturbaciones eléctricas creadas por las células de los seres vivos, y su estudio requiere dispositivos de medida precisos y confiables (Plonsey & Barr, 2007). En este contexto, el circuito integrado ADS1298R de Texas Instruments integra múltiples canales de alta resolución con opciones avanzadas de configuración, lo que lo convierte en un candidato idóneo para entornos de investigación y aplicaciones de grado médico (Texas Instruments, 2021).

Sin embargo, el mencionado chip de adquisición está limitado por un software privado, sin actualizaciones, incompatible con sistemas operativos actuales y carente una API documentada, dificultando su integración en entornos de desarrollo modernos. En respuesta, se propone la creación de una biblioteca dinámica que exponga una API funcional que permita la integración del chip ADS1298R en marcos de investigación orientados a la analítica de datos bioeléctricos.

En esta misma línea, la revisión de la literatura no identifica desarrollos previos orientados al diseño de una API para la familia ADS129x, por tanto, la solución aquí planteada busca cubrir ese vacío con una herramienta vigente, replicable y alineada con las necesidades de los investigadores de esta área de estudio.

1. Planteamiento del problema

1.1 Descripción

La electricidad y el magnetismo son el resultado de la interacción estática y dinámica de las cargas eléctricas, sus manifestaciones se observan en la luz, las auroras boreales, los rayos y la bioluminiscencia de algunos animales (Sadiku, 2003; Elvey, 1957; Ramesh & Meyer-Rochow, 2021), siendo éstas el punto de partida de numerosos estudios realizados por importantes investigadores como Franklin, Galvani, Volta, Ampere, Oesterd, los cuales llevaron a Maxwel a formalizar matemáticamente una teoría que explica todos estos fenómenos a nivel macroscópico (Mitolo & Araneo, 2019; Seroglou, Koumaras & Tselfes, 1998).

De igual modo estos fenómenos se observan en los organismos vivos en forma de campos bioeléctricos, originados por las cargas de los iones que se mueven a través de las membranas celulares mediante canales selectivos de sodio y potasio (Enderle, 2005), actividad que sustenta funciones vitales como la transmisión de Señales nerviosas, la contracción muscular y la actividad cardíaca, dichos campos inducen diferencias de potencial también denominados biopotenciales, que pueden medirse y aprovecharse en múltiples aplicaciones (Park, 2023).

Uno de los primeros estudios formales de la actividad bioeléctrica Data de los experimentos de Luigi Galvani con ratas en el siglo XVIII, al cual se le conoce como la teoría de la electricidad animal, desde entonces, el análisis de estos fenómenos ha evolucionado hasta conformar la electrofisiología moderna, disciplina de Vital importancia para el estudio y la aplicación de los biopotenciales (Piccolino, 1998).

Los avances de la electrofisiología se fundamentan en técnicas como la electrocardiografía (ECG), la encefalografía (EEG) y la electromiografía (EMG), las cuales permiten estudiar la actividad eléctrica del corazón, cerebro y los músculos, respectivamente (Fu et al., 2024). Estas técnicas fueron utilizadas principalmente en el ámbito clínico, con el transcurso del tiempo se fueron popularizando en contextos industriales, deportivos y tecnológicos, expandiéndose hacia el desarrollo de interfaces cerebro-computadora, sistemas de monitoreo remoto y dispositivos

portátiles de salud. De esta manera el estudio de la señal es bioeléctricas ha abierto Las puertas hacia soluciones tecnológicas con impacto social (Ni et al., 2024; Rani et al., 2023; Singh & Krishnan, 2023).

El avance tecnológico actual exige el procesamiento de grandes volúmenes de datos, una necesidad que se extiende al campo de la electrofisiología, donde la caracterización de patrones bioeléctricos demanda un creciente número de sensores distribuidos estratégicamente en el área del cuerpo humano bajo estudio (Batko & Ślęzak, 2022). Esta condición implica la necesidad de emplear sistemas con múltiples canales de entrada, capaces de registrar simultáneamente diversas bioseñales sin comprometer la calidad de los datos. Asimismo, en sintonía con los principios de la cuarta revolución industrial, estos sistemas deben ser compactos, portátiles e integrables fácilmente en soluciones tecnológicas modernas como dispositivos vestibles o herramientas médicas móviles (Ng et al., 2024; Asheghabadi et al. 2021).

En consecuencia, los sistemas de adquisición de datos (DAQ) deben cumplir no solo con rigurosos estándares médicos e industriales, sino también con la flexibilidad necesaria para adaptarse a entornos computacionales diversos y a bibliotecas de procesamiento de uso creciente en investigación y analítica de datos (Young & Schmid, 2021; Subrahmanya et al, 2021; Kabra, et al., 2022; Nacitarhan & Semiz, 2022). En este escenario, el circuito integrado ADS1298R de Texas Instruments representa una solución apropiada, debido a que incorpora ocho canales de alta resolución y cumple con las normativas internacionales AAMI EC11, EC13, IEC 60601-1, IEC 60601-2-27 e IEC 60601-2-51, lo que le confiere la capacidad de satisfacer simultáneamente los requisitos técnicos y regulatorios asociados a la adquisición de bioseñales (Texas Instruments, 2021).

Sin embargo, disponer de un circuito integrado como el ADS1298R cuyas prestaciones son excepcionales no garantiza por sí solo el funcionamiento adecuado de un sistema completo, debido a que éste depende de un paquete de software que pueda integrarse con lenguajes de programación diversos como Python, MATLAB o R, y además cumplir con estándares médicos como la IEC 62304. Esta normativa exige trazabilidad y gestión de riesgos en el desarrollo de

software médico, aspectos importantes para evitar errores en las señales o vulnerabilidades que comprometan datos sensibles de los pacientes (Karthick et al., 2020; Bombarda et al., 2022).

Además, en aplicaciones emergentes como la telemedicina o el análisis con inteligencia artificial, el software debe ser escalable y compatible con entornos heterogéneos, incluyendo aplicaciones de escritorio, computación en la nube y sistemas embebidos como Raspberry Pi (Croatti et al., 2022; Bajwa et al., 2021).

Una técnica fundamental para abordar los requerimientos mencionados en el desarrollo de software es la implementación de interfaces de programación de aplicaciones (API), las cuales estandarizan la comunicación entre sistemas, incluso cuando están escritos en lenguajes distintos u operan sobre plataformas heterogéneas. Asimismo, estas interfaces bajo un conjunto de reglas y protocolos permiten abstraer las complejidades inherentes tanto del hardware como del software, facilitando el trabajo y acortando el tiempo empleado por los desarrolladores en la construcción de soluciones en entornos donde coexisten múltiples tecnologías (De, 2023; Prongnuch & Wiangtong, 2016; Huf & Siqueira, 2019, Javed et al., 2020).

En el ámbito de la integración entre hardware y software bajo el sistema operativo Windows, es común el uso de un tipo especializado de API conocidas como bibliotecas de enlace dinámico (DLL), diseñadas para encapsular funciones de bajo nivel en módulos reutilizables (Dick & Volmar, 2018, Morrison, 2000). A diferencia de las API basadas en protocolos web o servicios abstractos, una DLL permite interactuar directamente con componentes físicos, facilitando operaciones técnicas complejas como el manejo de registros mediante una interfaz simplificada (Xosifovich et al., 2017). De esta forma se pueden desarrollar aplicaciones de alto nivel con acceso a las funcionalidades del hardware sin necesidad de conocimientos detallados de su electrónica subyacente.

No obstante, el software oficial provisto por el fabricante para la operación de la tarjeta de evaluación del circuito integrado ADS1298R resulta insuficiente para satisfacer los requerimientos mencionados, situación causada por el carácter cerrado y la ausencia de actualizaciones en los últimos años. Este hecho restringe el software a entornos operativos

obsoletos como Windows XP, incompatible con arquitecturas modernas de 64 bits o sistemas basados en Unix (Texas Instruments, 2016; Ward, 2014).

Por otra parte, el fabricante tampoco provee una API o una DLL que permita acceder a las funciones de bajo nivel del hardware (Texas Instruments, 2016), lo que en términos prácticos imposibilita su integración con arquitecturas modernas como servicios web, plataformas cloud o entornos de desarrollo especializados como LabVIEW o MATLAB, la cuales son herramientas de interés para aplicaciones de procesamiento de señales biomédicas en tiempo real (Dilip et al., 2021, Raval et al., 2017; Balaju et al., 2021).

Estas limitaciones, y especialmente la inexistencia de una capa de abstracción que independice el hardware del software hace que los desarrolladores se vean forzados a implementar soluciones ad-hoc, ya sea mediante el desarrollo de sus propias API o simplemente hacer desarrollos monolíticos que replican funcionalidades básicas, situación evidenciada en el trabajo realizado por Calderón (2024).

Por tanto, la carencia de una API restringe el aprovechamiento del ADS1298R como dispositivo de adquisición de alta precisión, confinándolo a paradigmas de desarrollo obsoletos que se contradicen con las exigencias tecnológicas actuales, tal como señalan Jaleel et al. (2020) en su análisis sobre interoperabilidad en dispositivos biomédicos.

1.2 Formulación

¿Es posible que el desarrollo de la biblioteca de vínculos dinámicos (DLL) como interfaz de programación de aplicaciones (API) permita la gestión del circuito integrado ADS1298R, garantizando compatibilidad con el sistema operativo Windows 10 y Windows 11, integración con el lenguaje de programación Python y flexibilidad para adaptarse a futuras actualizaciones tecnológicas?

2. Justificación

El acceso universal a la salud y a la tecnología es uno de los pilares de los objetivos del desarrollo sostenible (ODS) propuestos por las Naciones Unidas, los cuales fomentan la innovación tecnológica, el acceso a servicio de salud de calidad y la reducción de las desigualdades como ejes transformadores (Gamez, 2022). Por su parte la República de Colombia se articula a estos objetivos mediante la Misión Soberanía Sanitaria y Bienestar Social 2024-2033 que hace parte de la Política Nacional de Ciencia, Tecnología e Innovación, los cuales están dentro de un marco estratégico que busca impulsar los avances científicos y tecnológicos, y los servicios de salud en el país (Ministerio de Ciencia, Tecnología e Innovación, 2023).

En este escenario el desarrollo de una interfaz de programación de aplicaciones (API) que permita gestionar las funcionalidades del circuito integrado ADS1298R, toma importancia al funcionar como puente tecnológico directo entre el hardware biomédico restrictivo y el software de código abierto. Esta herramienta facilitaría a las redes de investigación el acceso a tecnologías de punta en la adquisición de bioseñales, impulsando soluciones a problemas locales sin depender de plataformas comerciales, reduciendo brechas tecnológicas y aportando a la democratización del hardware biomédico en entornos con recursos económicos limitados, siendo coherente con el objetivo 10 de los ODS que corresponde a la reducción de las desigualdades.

Por otra parte, el desarrollo de esta API se alinea con iniciativas globales en el campo de las bioseñales que buscan estandarizar los protocolos de comunicación, los formatos de almacenamiento de los datos y los algoritmos empleados en el preprocesamiento. Este esfuerzo es importante para garantizar la replicabilidad de las investigaciones y se vincula al ODS 17 (Alianza para lograr los objetivos) al promover un lenguaje técnico común que facilite la colaboración transnacional (Vidaurre et al., 2011). Además, contribuirá en la reducción del tiempo de diseño de prótesis electromecánicas, sistemas de diagnóstico de patologías neuromusculares y dispositivos de monitoreo de la salud, alineándose así con el ODS 3 (Salud y bienestar).

Mas allá de lo técnico, este desarrollo representará un aporte hacia la soberanía tecnológica al proveer una herramienta configurable a entornos diversos, al contribuir a la formación de talento local, al promover el trabajo interdisciplinar y potenciar la capacidad de los investigadores para abordar procesos de innovación con pertinencia regional. Por las razones expuestas, resulta pertinente desarrollar una API para la gestión del circuito integrado ADS1298R con la confiabilidad para abordar investigaciones científicas y permitir al usuario adaptarla a las particularidades de su entorno, promoviendo el desarrollo ágil de soluciones tecnológicas en el campo de las bioseñales.

3. Objetivos

3.1 Objetivo general

Desarrollar una interfaz de programación de aplicaciones en formato DLL mediante un conjunto de reglas y protocolos que permitan la gestión del circuito integrado ADS1298R haciendo uso de lenguajes de programación de alto nivel.

3.2 Objetivos específicos

Definir los requerimientos de la interfaz de programación de aplicaciones mediante el análisis de stakeholders y la revisión de la hoja de datos del circuito integrado ADS1298R para establecer la base del diseño del sistema.

Establecer la arquitectura, las clases y los métodos asociados al prototipo de la interfaz de programación de aplicaciones considerando el análisis de requerimientos de manera que se cumplan con las necesidades básicas de confiabilidad, seguridad y capacidad de escalamiento definidas.

Implementar la interfaz de programación aplicando metodologías ágiles y tradicionales, lenguajes computacionales adecuados, pruebas unitarias y el control de versiones, con el fin de asegurar que el código cumpla con los requerimientos funcionales y no funcionales establecidos.

Evaluar el desempeño de la interfaz de programación de aplicaciones desarrollada por medio de listas de comprobación, pruebas de integración sistémica y de estrés computacional, verificando de manera independiente cada característica y posteriormente su funcionamiento global.

4. Marco teórico

4.1 Estado del arte

La búsqueda se centró en artículos que brindan información relacionada al desarrollo de interfaces de programación de aplicaciones (API) y al uso de circuitos integrados de la familia ADS129x y del módulo ADS1298RECGFE-PDK en procesos investigativos. En la búsqueda realizada se utilizó la base de datos bibliográfica Scopus, los términos de búsqueda fueron:

API AND ADS1294, con esta búsqueda se obtuvieron cero resultados.

API AND ADS1296, con esta búsqueda se obtuvieron cero resultados.

API AND ADS1298, con esta búsqueda se obtuvo un resultado correspondiente a una investigación que hace uso del circuito integrado ADS1298 en la adquisición de señales electromiográficas, a fin de entrenar una red neuronal convolucional enfocada en el reconocimiento de patrones electromiográficos. Los resultados provistos por los autores indicaron una precisión alrededor de 95% en la clasificación de 12 movimientos funcionales y de agarre en un tiempo de entrenamiento de 100 segundos. Sin embargo, los autores no desarrollaron una interfaz de programación de aplicaciones, sino que utilizaron la interfaz Keras provista por la librería TensorFlow en el proceso de entrenamiento de la red neuronal (S. Pancholi & A. M. Joshi, 2020).

API AND ADS1298RECGFE-PDK, con esta búsqueda se obtuvieron cero resultados.

API AND electromyography, con esta búsqueda se obtuvieron siete resultados, de los cuales tres involucran en las investigaciones el uso de interfaces de programación de aplicaciones en procesos con señales electromiográficas (Pasqualin, B.R et al., 2024; S. Pancholi et al., 2022; S. Guo et al., 2016); en tanto que (P. Cardoso et al., 2017) desarrollaron una interfaz de programación de aplicaciones Web con el objeto de acceder a datos recolectados de señales

electromiográficas de personas con esclerosis lateral amiotrófica a fin de proveer una herramienta médica.

API AND EMG, con esta búsqueda se obtuvieron 15 resultados, entre los que se encuentran los previamente indicados con la búsqueda API AND electromyography. Adicionalmente, dos involucran en las investigaciones el uso de interfaces de programación de aplicaciones en procesos con señales electromiográficas (Y. Wang et al., 2022; A. Andrushevich et al., 2017), y uno corresponde al desarrollo de una interfaz cerebro computador que es controlada por dispositivos EEG/ERP y que utiliza señales electromiográficas de los músculos faciales a fin de obtener información eléctrica relacionada con las expresiones del rostro (E. Ertekin et al., 2022).

API, con esta búsqueda se obtuvieron 78505 resultados, entre estos se encuentran: interfaz con el objeto de proveer simulación y visualización de eventos relacionados a incendios forestales (Rui Wu et al., 2023). Interfaz que permite leer datos en modos online y offline relacionados a las propiedades de anisotropía elástica en materiales (Zheng Ran et al., 2023). Interfaz aplicada en la minería de procesos capaz de integrarse a sistemas de información que producen registros de eventos (Ilias Merkoureas et al., 2023). Plataforma de laboratorio virtual con interfaz de programación de aplicaciones basada en metadatos para la creación de componentes de software dinámicos (F. Pereira et al., 2023). Herramienta que tiene por objeto acortar el tiempo empleado en el proceso de diseño de enlaces punto a punto terrestres y satelitales de microondas utilizando una interfaz de programación de aplicaciones que posibilita el análisis y visualización de las características específicas de los sitios que pueden alterar la calidad de servicio del enlace (Ferreira, E et al., 2023).

A partir de lo anterior se establece que, de acuerdo con la búsqueda realizada, las interfaces de programación de aplicaciones que se han desarrollado no involucran el uso de circuitos integrados de la familia ADS129x o el módulo ADS1298RECGFE-PDK.

Diversos autores han reportado investigaciones en las cuales utilizan, ya sea algún circuito integrado de la familia ADS129x o el módulo ADS1298RECGFE-PDK en aplicaciones con señales electromiográficas. A continuación, se presenta la información de estos trabajos.

(M. Schoutenet et al., 2022) haciendo uso del módulo ADS1298RECGFE-PDK adquirieron de forma continua información de señales electromiográficas con el objeto de medir la impedancia de electrodos electromiográficos construidos a partir de impresión 3D tanto con trazos de tinta plateada como con trazos PI-ETPU. En su trabajo, realizaron mediciones de señales electromiográficas durante cinco contracciones isométricas y determinaron que la impedancia media absoluta para todas las mediciones con los electrodos de tinta plateada fue 72 KΩ y con los electrodos construidos con PI-ETPU fue 443 KΩ. De acuerdo con estos resultados los autores indicaron que la tinta de plata reduce considerable la impedancia de los electrodos.

(A. Toro et al., 2020) describieron el diseño e implementación de un brazalete electromiográfico de cuatro canales de código abierto para el reconocimiento de gestos manuales. Las etapas del proyecto incluyen acondicionamiento, filtrado y amplificación de las señales electromiográficas; conversión análoga digital haciendo uso de un microcontrolador PIC18F2550; y comunicación de datos a computador mediante el protocolo UART del mencionado microcontrolador. Los autores indicaron que a futuro el proyecto se realizará con el circuito integrado ADS1298 con el objeto de incrementar el número de canales y obtener una mejor precisión en el desempeño del brazalete, esto permitirá la adecuación de un clasificador de gestos basado en una red neuronal recurrente con aplicaciones en el control de prótesis de manos.

(S. Pancholi & A. M. Joshi, Oct. 2020) utilizaron el circuito integrado ADS1298 en la adquisición de señales electromiográficas con el fin de realizar extracción de características basadas en transformadas wavelets. A partir de la adquisición de señales electromiográficas de tres personas amputadas de miembro superior y utilizando los ocho canales del circuito integrado, el método propuesto suministró una precisión de hasta 98.32% en la clasificación de patrones electromiográficos.

Previo a la investigación anteriormente descrita, los autores (S. Pancholi & A. M. Joshi, 2019) desarrollaron una plataforma embebida basada en el circuito integrado ADS1298 en la adquisición de señales electromiográficas de cuatro personas, una de ellas amputada de miembro superior, con el propósito de ser utilizada en procesos de reconocimiento de gestos con las

manos. El método propuesto de extracción de características incluyó técnicas de raíz media cuadrática (RMS), longitud de forma de onda (WL), cruce por cero (ZC) y cambio de signo de la pendiente (SSC). Los resultados indicaron que haciendo extracción de características en el procesador y validando el método fuera de línea la precisión se ubicó en el 92.0% en la persona amputada y entre 94.5% y 97.6% en las personas sin esta condición; y realizando en el procesador tanto la extracción de características como la validación del método la precisión promedio en la persona amputada fue de 91.65% y en las personas sin esta condición se ubicó entre 92.20% y 97.75%.

(K. de Jager et al., 2019) desarrollaron un registro de actividad electromiográfica con multiplexación de canales a fin de seleccionar la configuración del biopotencial provisto por los electrodos. El sistema provee seis conversores análogo digital haciendo uso del circuito integrado ADS1298 y un arreglo de tres multiplexores 8x8 con dos canales por multiplexor. Los autores indicaron que si bien la calidad de la señal es menor (13.5 dB) en comparación con los dispositivos comerciales utilizados en el registro de señales electromiográficas (19.5 dB), el sistema propuesto puede ser apropiado en aplicaciones de control mioeléctrico.

(M. A. Favretto et al., 2018) desarrollaron un sistema de monitoreo electromiográfico de 32 canales basado en el circuito integrado ADS1298 con frecuencia de muestreo de hasta 2 kHz por canal y ganancia en voltaje configurable de 1 a 12. A partir de pruebas experimentales se obtuvo un valor de rectificación promedio de 4.58 ± 1.2 mV, raíz media cuadrática de 30.0 ± 16.9 μV y una frecuencia media de 110.0 ± 12.6 Hz. Los autores indicaron que estos resultados se encuentran dentro de los parámetros normales en relación con las consideraciones normativas y trabajos similares.

Adicionalmente se han reportado investigaciones en las cuales utilizan, ya sea algún circuito integrado de la familia ADS129x o el módulo ADS1298RECGFE-PDK en aplicaciones orientadas a herramientas de diagnóstico de enfermedades cardiovasculares (T. Soro et al., 2023; S. Belaid et al., 2022; Kumar N. et al., 2021), electrocardiografía (B. Nowak & K. Jędrzejewski, 2023; E. Supo et al., 2022; Dezső, D et al., 2021; R. Huamani R. et al., 2017; Fort C.M., et al., 2017; C. -X. Que et al., 2016; C. Cristea et al., 2015; Cook A.J. et al., 2015; Weyer S. et al.,

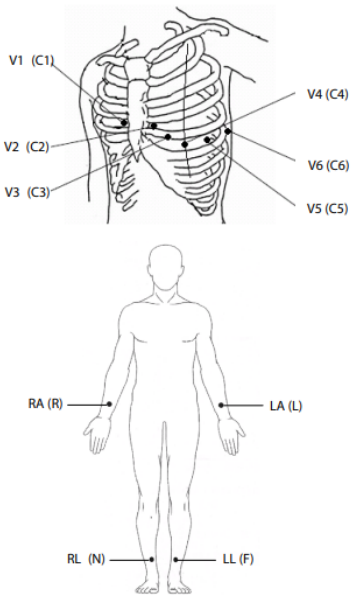
2015; G. Barabino et al., 2015), dispositivos enfocados en el estudio de la conducción nerviosa (Cossul S. et al., 2020), y electroencefalografía (Zhang Y. et al., 2019; Uktveris T. & Jusas V., 2018).

4.2 Señales Bioeléctricas

Las señales bioeléctricas son potenciales eléctricos generados en las células nerviosas y musculares como consecuencia de cambios electroquímicos que ocurren dentro y entre ellas, este flujo de iones a través de la membrana celular puede medirse utilizando electrodos intracelulares o extracelulares. Los potenciales generados por una célula excitada pueden transmitirse de una a otra por medio de su axón, cuando muchas células se activan, se genera un campo eléctrico que se propaga a través del tejido biológico. Estos cambios en el potencial extracelular pueden registrarse en la superficie del tejido u organismo mediante el uso de electrodos superficiales. Ejemplos de este fenómeno son el electrocardiograma (ECG), el electrogastrograma (EGG), el electroencefalograma (EEG) y el electromiograma (EMG) (Enderle, Blanchard, & Bronzino, 2005).

4.2.1 Electrocardiografía. Es el proceso por el cual se adquieren las señales provenientes de la actividad eléctrica del corazón, en la figura 2 se muestra un ejemplo de electrocardiograma donde la onda P indica la despolarización de las aurículas, mientras que el complejo QRS representa la despolarización de los ventrículos. La repolarización ventricular se manifiesta como la onda T, y la repolarización auricular queda enmascarada por la despolarización ventricular. Las variaciones en la amplitud y duración de las diferentes partes del ECG ofrecen información diagnóstica relevante para los médicos (Enderle, Blanchard, & Bronzino, 2005).

En la Figura 1 se presenta un ejemplo de la forma de conectar los sensores al cuerpo humano para el registro de las señales con un electrocardiógrafo BeneHeart R3/BeneHeart R3A.



AHA	IEC	Colocación de electrodos
V1	C1	Cuarto espacio intercostal, en el borde derecho del esternón
V2	C2	Cuarto espacio intercostal, en el borde izquierdo del esternón
V3	C3	Equidistante entre las posiciones V2 (C2) y V4 (C4) del electrodo
V4	C4	Quinto espacio intercostal, en la línea medioclavicular izquierda
V5	C5	Línea axilar anterior izquierda, horizontal con respecto a la posición V4 (C4) del electrodo
V6	C6	Línea media axilar izquierda, horizontal con respecto a la posición V4 (C4) del electrodo
RA	R	Sobre la muñeca derecha
LA	L	Sobre la muñeca izquierda
RL	N	Sobre el tobillo derecho
LL	F	Sobre el tobillo izquierdo

Figura 1. Conexión de electrodos para la adquisición de señales ECG
Fuente: Electrocardiógrafo BeneHeart R3/BeneHeart R3A Manual del operador" por Shenzhen Mindray Bio-Medical Electronics Co, 2013

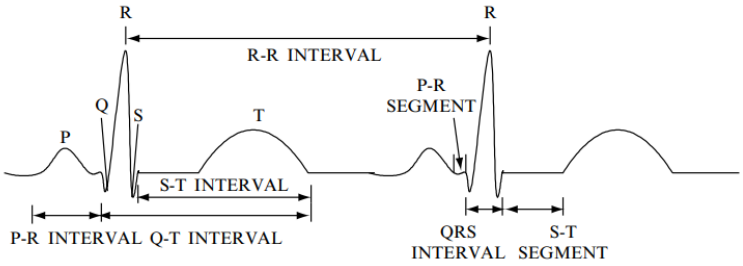


Figura 2. Forma de onda típica de un electrocardiograma
Fuente: Introduction to Biomedical. Engineering (2a ed.) por Enderle, J., Bronzino, J., & Blanchard, S. M., 2005, Elsevier Academic Press.

4.2.2 Electroencefalografía. La electroencefalografía (EEG) implica la medición no invasiva de los campos eléctricos en el cerebro, por medio de electrodos colocados en el cuero cabelludo se registran los potenciales de voltaje generados por las neuronas. El EEG es ampliamente utilizado en tratamientos de neurorrehabilitación, en la psicología experimental, en la captura de neuroimágenes, con extensiones más recientes en la neurociencia computacional. La versatilidad y accesibilidad de esta técnica, junto con los avances en el procesamiento de señales, permiten que siga ofreciendo nuevas capacidades e innovaciones (Biasiucci, Franceschiello, & Murray, 2019). En la figura 3 se presenta un ejemplo aplicado al estudio de la actividad eléctrica del cerebro en un infante.

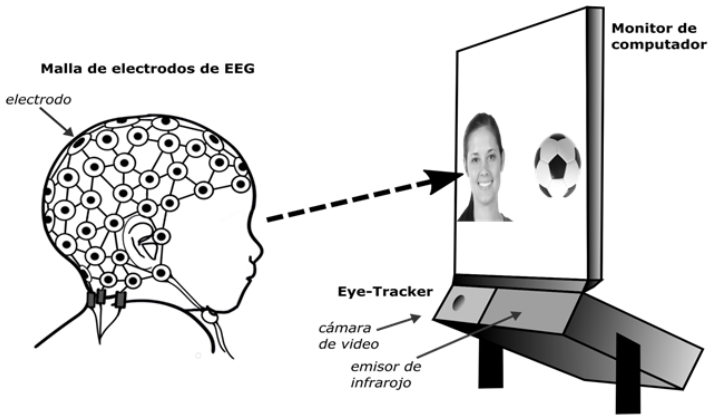


Figura 3. La electroencefalografía (EEG) aplicada en el seguimiento de la mirada
Fuente: https://www.scielo.cl/img/revistas/signos/v55n110//0718-0934-signos-55-110-902-gf1.png.

4.2.3 Electromiografía. Se puede definir como el registro de la actividad eléctrica de los músculos esqueléticos durante la contracción. Este proceso, esencial para el movimiento, inicia desde la toma de decisiones del cerebro, la transmisión del mensaje por medio de señales nerviosas, la activación eléctrica muscular y eventos bioquímicos complejos. En la figura 4 se muestra las formas de onda típicas de un electromiograma.

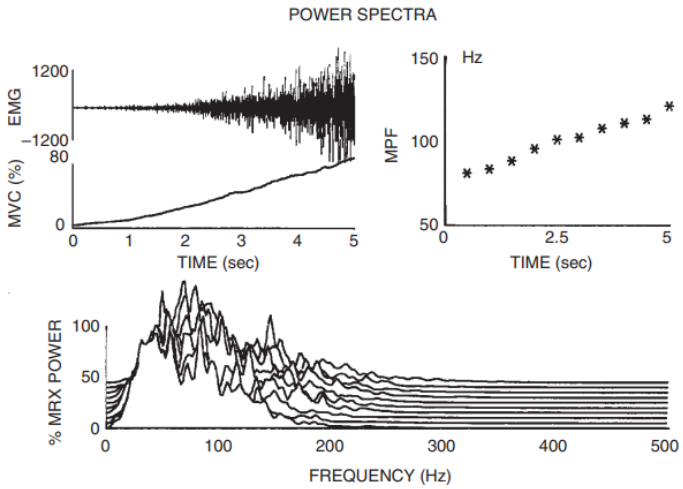


Figura 4. Formas de onda típicas obtenidas en un estudio de electromiografía (EMG)
Fuente: Electromyography: Physiology, Engineering, and Non-Invasive Applications por Merletti, R., & Parker, P., 2004, Nashville, TN, Estados Unidos de América: John Wiley & Sons.

El EMG proporciona información sobre la función neuromuscular y se utiliza en diversas aplicaciones, en el ámbito científico, permite avanzar en el entendimiento del sistema neuromuscular, mientras que en medicina facilita el diagnóstico de enfermedades asociadas a este sistema (Merletti & Parker, 2004).

4.3 Front-end analógico ADS1298R

El circuito integrado ADS1298R es un front end de ocho canales con conversión delta sigma de 24 bits y tasas de muestreo de hasta 32 kSPS, integra amplificadores de ganancia programable por canal, referencia interna y oscilador. Cada canal dispone de un multiplexor de entrada configurable que permite vincular señales internas para prueba, medición de temperatura y detección de desprendimiento de electrodos. Adicionalmente, posee un módulo denominado Wilson Central Terminal para aplicaciones en ECG de 12 derivaciones. La versión R añaden medición de impedancia respiratoria. Es posible encadenar múltiples dispositivos en daisy chain. Se ofrecen en BGA de 8 por 8 mm y en TQFP de 64 pines la versión BGA ADS129x opera de 0 a 70 °C y las versiones ADS129xR cubren de -40 a +85 °C (Texas Instruments, 2015). En la Figura 5 se puede observar una de las presentaciones comerciales del circuito integrado bajo estudio.

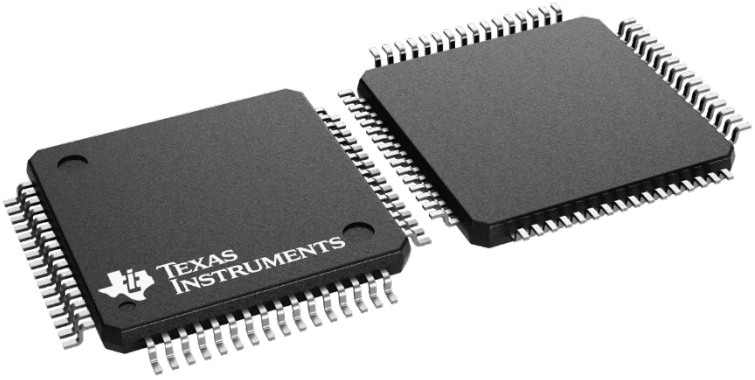


Figura 5. Circuito integrado ADS1298R en encapsulado TQFP de 64 pines
Fuente: https://www.ti.com/content/dam/ticom/images/products/package/p/pag0064a.png:large

4.3.1 Arquitectura. El diseño del ADS 1298R se puede modelar en 11 módulos principales que se encargan de la adquisición, el acondicionamiento, el muestreo, la cuantización, la codificación y la comunicación de los datos. En la Figura 6 se muestra la arquitectura del circuito integrado. A continuación, se presenta una descripción de los principales bloques constitutivos.

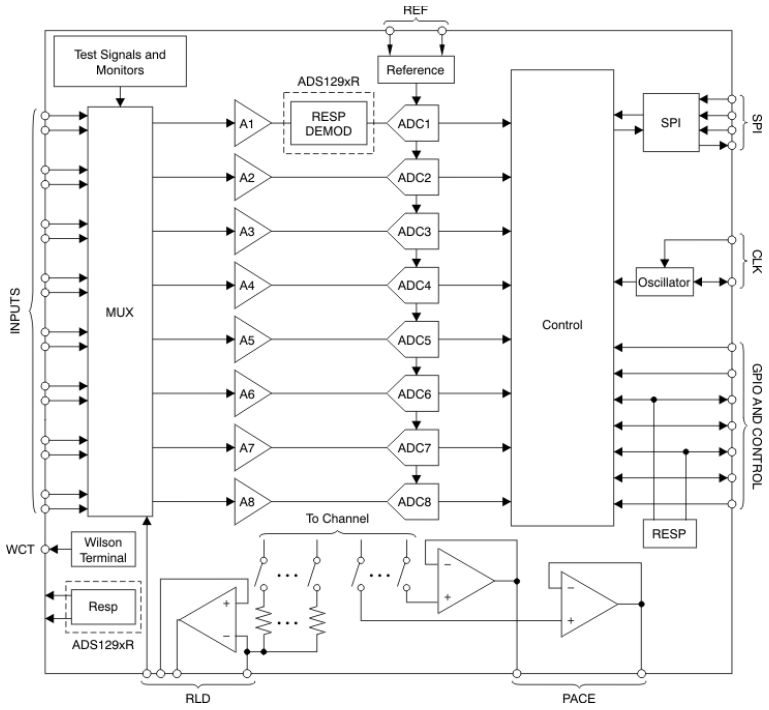


Figura 6. Arquitectura del circuito integrado ADS1298R
Fuente: https://www.ti.com/ds_dgm/images/alt_sbas459k.giflarge

El bloque de multiplexación constituye el punto de entrada. Por medio de la configuración de los registros se elige entre electrodos externos, cortocircuito de prueba, señales internas de verificación y rutas para variables auxiliares. Esta conmutación local a cada canal permite aislar fallas, validar el cableado y verificar la red de referencia del paciente sin hardware adicional. Seguidamente, el bloque que abarca desde A1 hasta A8 son los amplificadores de ganancia programable que acondicionan cada una de las señales seleccionadas. Su objetivo es que el usuario pueda adaptar el rango dinámico de cada derivación antes de la conversión, preservando el rechazo de modo común.

A continuación, el bloque comprendido desde ADC1 hasta ADC8 se encargan de la conversión analógico a digital de 24 bits utilizando modulación sigma-delta. Se dispone de

modos continuo y bajo demanda, así como funciones digitales para atenuar contenido en modo común y ajustar el nivel de continua asociado a los electrodos (Texas Instruments, 2015). Por otra parte, el bloque de referencia da apoyo a los bloques de amplificadores y conversores. Este integra una fuente con búfer seleccionable a 2,4 o 4 voltios y admite deshabilitación cuando el diseño requiere una referencia externa. El desacoplo local recomendado limita el ruido dentro de la banda de interés, lo que se refleja en menor error de ganancia.

El bloque de Control gobierna la lectura y escritura de los registros, accediendo así a todas las funcionalidades del circuito integrado. Junto a este se encuentra el bloque de comunicación SPI el cual se encarga de la comunicación con el usuario. Emplea líneas de selección, reloj, datos de entrada y datos de salida, con un decodificador de comandos para lectura y escritura de registros. La línea DRDY anuncia la disponibilidad de nuevas muestras y la frecuencia de reloj debe dimensionarse según el número de canales, resolución y tasa de datos.

El Oscilador interno aporta un reloj de 2,048 MHz con tiempo de arranque breve, suficiente para sostener la cadena de conversión y la comunicación SPI. Cuando se requiere una referencia común para varios circuitos, el dispositivo acepta reloj externo y mantiene el enrutamiento necesario para conservar la relación de fase con las señales de disparo de adquisición.

La Respiración, presente en la versión R, mide impedancia torácica por inyección de una portadora y demodulación en el canal uno. La frecuencia y la fase se ajustan para evitar interferencia con el ECG y para compensar retardos de la cadena analógica. Al activarse, el canal uno se dedica a esta función y libera a los demás canales para la captura de biopotenciales cardiacos (Texas Instruments, 2015).

4.4 Fundamentos de UART

Universal Asynchronous Receiver/Transmitter se utiliza para la transferencia serial de datos, un bit a la vez, entre dos dispositivos electrónicos. Esta comunicación se describe como asincrónica porque el transmisor no necesita enviar una señal de reloj al receptor para sincronizar los datos, por tanto, se debe establecer la velocidad de transmisión antes de iniciar este proceso.

Debido a que no se requiere un reloj, los datos se envían típicamente utilizando solo dos líneas de señal TX y RX. Al igual que una línea telefónica regular, la conexión de transmisión de datos (TX) de un extremo se conecta a la conexión de recepción de datos (RX) en el otro extremo de la conexión, y viceversa.

De forma tradicional el UART se combina con conversores de nivel y controladores de línea para implementar interfaces como RS-232 o RS-485, ampliamente utilizados en la industria. Cuando la distancia es corta se puede omitir el acondicionamiento y enlazar dos UART directamente, práctica viable, aunque no estandarizada. La cantidad de símbolos por segundo se denomina velocidad en baudios o tasa de modulación, y en ciertos esquemas un símbolo puede representar más de un bit; por ejemplo, en QPSK un símbolo codifica dos bits y la tasa de bits resulta el doble de la de baudios. En un enlace UART binario simple, en cambio, la tasa de bits coincide con la de baudios.

Antes de iniciar el envío, el emisor y el receptor pactan la tasa de bits y la transmisión comienza cuando el emisor coloca un bit de inicio en nivel bajo, tras lo cual el receptor detecta el flanco descendente y espera 1,5 periodos de bit para muestrear el primer dato. A partir de allí toma muestras cada periodo de bit hasta completar el número acordado de bits útiles, típicamente siete u ocho. La paridad es opcional y requiere configuración coincidente en ambos extremos; sirve para detectar errores y adopta nivel alto o bajo según se elija paridad par o impar. La trama concluye con uno o dos bits de parada en nivel alto, siendo común el formato 8N1: ocho bits de datos, sin paridad y con un bit de parada (Molloy, 2016).

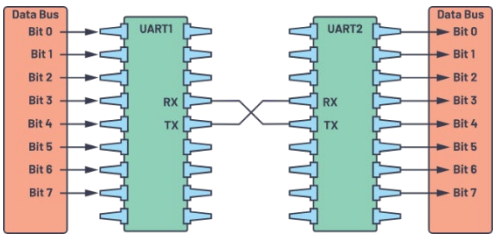


Figura 7. Conexión de dos dispositivos UART
Fuente: https://www.analog.com/en/_/media/images/analog-dialogue/en/volume-54/number-4/articles/uart-a-hardware-communication-protocol/335962-fig-02.svg?h=270&hash=822F6D4489C5A7B5EA8209D4139D5955&rev=7d55981f85ba4f1fb8f2c41635303994

5. Metodología

5.1 Tipo de proyecto

La presente investigación se clasifica como un estudio de tipo aplicado, dado que su objetivo es llevar a la práctica los conceptos teóricos y técnicos de las áreas de la electrónica y del software con el fin de desarrollar una interfaz de programación de aplicaciones haciendo uso del circuito integrado ADS1298R. El enfoque adoptado busca obtener resultados replicables por medio de técnicas que orientan el desarrollo del software, además de utilizar herramientas especializadas de testing que garanticen una evaluación imparcial sin influencias subjetivas.

5.2 Método

La metodología propuesta para el presente estudio es el análisis y síntesis la cual implica el proceso de descomponer un fenómeno en sus elementos constitutivos para comprender su estructura y funcionamiento. Por otro lado, la síntesis se refiere a la integración de elementos separados en una unidad coherente, esta etapa sin el análisis se convierte en una construcción imaginativa alejada de la realidad, por tanto, las dos partes son componentes fundamentales del método científico (Lopera Echavarria, Ramírez Gómez, & Zuluaga Aristazábal, 2010).

Por tanto, se busca realizar un análisis del chip ADS1298R, desglosando cada componente para comprender sus funciones y características específicas, es esta etapa se identificarán los requisitos funcionales y no funcionales, los cuales serán el punto de partida para llevar a cabo la síntesis, que se verá reflejada en el diseño de la API. Este enfoque combinado de análisis y síntesis asegura que cada aspecto del proyecto sea abordado por el camino óptimo para alcanzar la meta planteada en los objetivos, resultando en una API que cumpla con los requisitos establecidos.

5.3 Instrumentos de recolección de información

5.3.1 Fuentes primarias. En el proceso de rastreo de información, se utilizan los libros como fuente principal, datasheet del circuito integrado ADS1298R y el software del proveedor debido a que suelen ser elaborados por expertos y académicos en sus respectivos campos.

5.3.2 Fuentes secundarias. Se emplean diversos artículos científicos y tesis académicas. Estos son fundamentales, ya que ofrecen antecedentes y experiencias relevantes que orientan las estrategias investigativas desarrolladas en este trabajo.

6. Resultados del proyecto

6.1 Definición de los requerimientos de la interfaz de programación de aplicaciones

En esta fase del proyecto se realizaron varias reuniones con el asesor técnico y el asesor metodológico, los cuales tomaron el rol de stakeholders con el objeto de discutir de forma detallada las necesidades funcionales y no funcionales de la API. En estas sesiones se abordaron las necesidades particulares de los investigadores involucrados en el proyecto y se contrastaron con las restricciones inherentes impuestas por el fabricante del circuito integrado ADS1298R, de este análisis surgió el siguiente conjunto de requerimientos.

6.1.1 Requerimientos no funcionales. Desde esta perspectiva la solución propuesta debe funcionar en sistemas operativos con arquitectura de 64 bits pertenecientes a la familia Microsoft Windows 10 y superiores. Por otra parte, la API debe ser escalable y poder integrarse en diferentes lenguajes de programación de alto nivel.

6.1.2 Requerimientos funcionales. La biblioteca de vínculos dinámicos debe ofrecer rutinas que posibiliten la configuración de las principales características del ADS1298R, incluyendo la selección de ganancia programable, el ajuste de la frecuencia de muestreo, la habilitación individual de canales y el modo de entrada de estos. Del mismo modo, debe contar con métodos que inicien o detengan el proceso de adquisición y que transmitan datos crudos a aplicaciones desarrolladas en lenguajes de alto nivel, preservando la coherencia temporal mediante marcas de tiempo y mecanismos de verificación.

6.2 Establecimiento de la arquitectura y diseño conceptual de la API BioAdqRt.dll

La arquitectura de la BioAdqRt.dll se concibe como una interfaz que separa con claridad el mundo del análisis de datos de las particularidades del enlace físico y del hardware de adquisición. En esta primera versión el objetivo es la operatividad, minimizando dentro de lo posible todas aquellas características que no inciden en el funcionamiento de la API. Por ejemplo, la comunicación se resuelve mediante bloques binarios sin mecanismos propios de gestión de errores, delegando la detección y corrección a la capa física (UART).

La anterior decisión agiliza cada una de las etapas de desarrollo, dejando explícita la ruta de evolución hacia controles de integridad cuando el sistema supere la fase de pruebas. Sobre esta idea la DLL se organiza en tres grandes responsabilidades (ver Figura 8) que no deben mezclarse: el transporte de los datos (codificación y decodificación con versión y direccionamiento), el traductor de los datos (semántica de canal, unidades, escala y metadatos de sesión) y la interfaz pública que expone los servicios abstrayendo completamente las características de bajo nivel. A continuación, se describen cada uno de estos bloques.

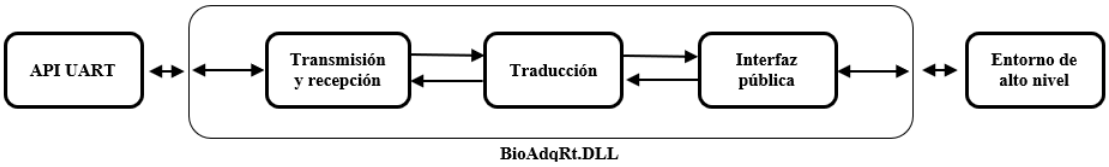


Figura 8. Los tres bloques principales de la API BioAdqRt.dll
Fuente: Autor.

6.2.1 Bloque de transmisión y recepción. Esta etapa actúa como un canal entre la API que gestiona la capa física y los demás bloques de la API ADS1298.dll. Para ello se adopta un formato binario basado en el estándar Type–Length–Value (TLV) que permite distinguir la naturaleza de cada unidad y al mismo tiempo minimizar la introducción de latencias y sobrecargas innecesarias. En la Figura 9 se puede observar la estructura del formato TLV.

Esta decisión es coherente con el alcance de esta primera versión de la interfaz, en la que toda forma de gestión de errores se delega explícitamente a la capa física del enlace, y el énfasis se sitúa en alcanzar una cadena funcional y estable que permita validar el hardware, verificar la instrumentación y obtener registros útiles con rapidez.

Tipo	Longitud	Valor

Figura 9. Formato propuesto para las tramas de datos.
Fuente: Autor.

Con el formato TLV definido, el siguiente paso es ajustarlo a la comunicación entre el PC y el circuito integrado ADS1298R, estableciendo la estructura de los paquetes que circularán en cada

sentido y la lectura básica de sus campos. A continuación, se describe cada sección de las tramas que se emplearán en esta etapa.

El campo de tipo constituye la puerta de entrada de cada paquete de información y cumple una función doble: por un lado, identifica de manera inmediata si el contenido corresponde a datos de adquisición, a órdenes de control o a las diferentes respuestas del chip de adquisición, por otro habilita el despacho dentro de la biblioteca sin explorar el resto del contenido. En términos prácticos, el lector de la API examina el primer byte que llega desde el búfer de entrada y según su valor, envía el paquete al decodificador de datos o al intérprete de comandos; de este modo la complejidad de las rutas internas se mantiene contenida y la lógica de la API no queda expuesta a decisiones de bajo nivel.

El campo de longitud declara de manera explícita cuántos bytes componen el valor que sigue, de modo que el parser pueda saber con precisión dónde termina una unidad y dónde comienza la siguiente, aun cuando la secuencia se lea en ráfagas asimétricas o llegue intercalada con otras tareas del sistema. La biblioteca coteja la longitud anunciada con la cantidad efectivamente disponible en el búfer y, solo si el número de bytes es suficiente, extrae la unidad completa y la entrega a la capa superior; en caso contrario espera el arribo de más datos sin bloquear al resto de la aplicación.

El campo de valor corresponde al contenido de interés de cada paquete TLV y por diseño es polimórfico: su forma concreta depende del tipo declarado. Cuando el tipo corresponde a datos de adquisición, el valor transporta una ventana de muestras consecutivas organizada en forma contigua y sin rellenos, siguiendo un orden fijo de canales. Cuando el tipo representa un comando, el valor se interpreta como una orden compacta del host hacia el dispositivo de adquisición, por ejemplo, la lectura o escritura de un registro. Finalmente, cuando el tipo corresponde a una respuesta del dispositivo, el valor porta la confirmación o el resultado de la operación solicitada.

A partir del formato generalizado para las tramas, se precisan ahora las estructuras de los paquetes específicos que emite el front-end analógico y los que emite el PC (API).

Tramas del Front-End analógico: para esta versión se contemplan dos tipos: stream de adquisición y respuestas a lecturas de registro. A continuación, se presenta el diseño lógico de los paquetes de datos entregados por el sistema de adquisición.

El stream de adquisición adopta la forma tipo + longitud + valor, donde el valor inicia con una marca temporal y continúa con una secuencia contigua de muestras tomadas directamente del bus SPI del ADS1298. La marca temporal es un entero little-endian que se incrementa por trama y desborda de manera natural, siendo su función marcar la posición temporal de cada muestra sin introducir cómputos adicionales. Luego el resto de la trama replica el formato nativo del ADS1298: primero los 24 bits de status y enseguida los 8x24 bits de los canales en el mismo orden de transmisión del bus SPI, sin rellenos ni reempaquetado intermedio. En la Figura 10 se presenta la estructura de la trama que contendrá las muestras tomadas en cada canal del ADS1298.

Tipo	Longitud	Time	Status	CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8

Figura 10. Formato propuesto para la trama de adquisición de datos
Fuente: Autor

El segundo tipo de trama corresponde a las respuestas del ADS1298R ante operaciones de lectura de sus registros. Este formato se mantiene en una forma mínima del estándar TLV: tipo (1 byte) + longitud (1 byte) + valor. El valor contiene sin modificación los bytes devueltos por el bus SPI en el orden en que fueron leídos. La longitud indica cuántos bytes contiene el valor y debe coincidir con lo solicitado en la orden previa. La API correlaciona la respuesta con la petición; si la longitud no coincide, descarta la trama y solicita nuevamente el dato. En la Figura 11 se muestra la forma de la trama para las respuestas de lectura de los registros del circuito integrado.

Tipo	Longitud	REG DATA 1	REG DATA 2	REG DATA N

Figura 11. Trama para la respuesta de lectura de registros del circuito integrado ADS1298R
Fuente: Autor

Tramas del PC: para esta versión se contemplan dos tipos: lectura y escritura. A continuación, se muestra la estructura de estos paquetes de datos.

Para la trama de orden de lectura, el campo tipo identifica la operación, la longitud es fija e igual a 2 y el valor replica el formato del bus SPI del ADS1298R con dos octetos: OPCODE1 (dirección inicial de registro) y OPCODE2 (número de registros a leer). El valor llega sin modificación por SPI y el ADS1298R devuelve en la respuesta los bytes solicitados. La Figura 12 muestra la estructura de este paquete de datos.

Tipo	Longitud	OPCODE 1	OPCODE 2

Figura 12. Formato propuesto para la orden de lectura de registros del circuito integrado ADS1298R
Fuente: Autor

En el caso de la solicitud de escritura, el campo tipo identifica la operación, la longitud expresa el tamaño total del valor y este se compone de OPCODE1 (dirección inicial del registro), OPCODE2 (número de registros a escribir, según convención del SPI del ADS1298R) y REG DATA que corresponden a los datos a escribir en el mismo orden en que serán almacenados en los registros consecutivos a partir de la dirección base. En la Figura 13 se presenta la estructura del paquete de escritura de registros.

Tipo	Longitud	OPCODE 1	OPCODE 2	REG DATA 1	REG DATA 2	REG DATA N

Figura 13. Formato propuesto para la orden de escritura de registros del circuito integrado ADS1298R
Fuente: Autor

6.2.2 Interfaz pública. Esta interfaz expone un conjunto mínimo de operaciones, todas las llamadas o instrucciones se basan en el lenguaje natural de los humanos y no revelan detalles del enlace ni las diferentes instrucciones de bajo nivel que se deben ejecutar. Un ejemplo de esto es el inicio del dispositivo de adquisición: con una sola instrucción de alto nivel se establecen las condiciones de adquisición, se inicializa el bus SPI, se reserva espacio de memoria y se ejecuta la tarea interna de comunicación que recibe y envía tramas sin intervención del usuario.

La gestión del dispositivo por medio de la interfaz se expresa en términos de intención y no de la manipulación de registros basada en las reglas del fabricante. Una sola instrucción permite fijar la frecuencia de muestreo, otra activar o desactivar un canal, otra ajustar su ganancia en términos de voltaje y otra seleccionar el modo de entrada de los canales, siendo suficiente para que la biblioteca traduzca la intención a las órdenes necesarias que ejecutará el bloque de control del circuito integrado ADS1298R. La Figura 14 muestra el ejemplo de un conjunto de instrucciones que pueden ser encapsuladas por una sola función de alto nivel.

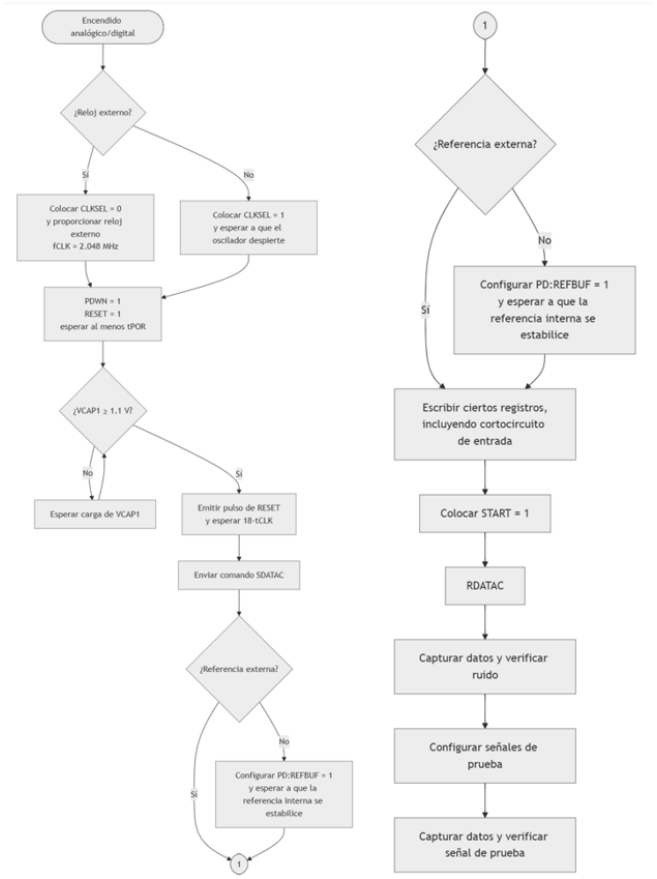


Figura 14. Diagrama de flujo de la secuencia de arranque abstraída en la función powerUp()
Fuente: Adaptada por el autor de: https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/73/Flow_5F00_chart_5F00_initilization.jpg

En este sentido la interfaz pública es el resultado de un conjunto de instrucciones de alto nivel que tiene como objeto desbloquear funciones que requieren conocimientos especializados en hardware. Estas instrucciones o métodos en lenguaje humanizado se derivan directamente de los requerimientos funcionales definidos en la primera etapa de resultados de este trabajo.

Adicionalmente, los requerimientos funcionales son contrastados con los requerimientos impuestos por el fabricante del circuito integrado, dando lugar a un flujo de trabajo que corresponde a parametrización del puerto, configuración funcional, inicialización del dispositivo y operación continua. Cada una de estas etapas se expresa con una instrucción de alto nivel.

La parametrización comienza designando el puerto de comunicación en el que estará conectado el circuito integrado ADS1298R. Una sola instrucción valida la disponibilidad, crea el contexto de sesión y deja la comunicación establecida. En esta versión las pruebas se realizan utilizando puertos COM virtualizados. La configuración se expresa en términos típicos del contexto de análisis de señales: una instrucción define qué canales estarán encendidos y en qué modo operarán; otra fija la frecuencia de muestreo de la sesión y otra ajusta la ganancia de voltaje de los canales de interés.

La inicialización se encarga de la puesta a punto del bus de datos, verificación del estado del circuito integrado ADS1298R y reserva de memoria para el búfer de comunicación. Posterior a la ejecución de esta fase el dispositivo queda en condiciones de adquirir los datos. Por medio de una instrucción de alto nivel se inicia el proceso de adquisición y a partir de ese momento el flujo de datos está disponible en tiempo real. Esta operación habilita la entrega de cada bloque de datos para ser graficados de forma continua o para ser almacenados. La finalización se realiza con una instrucción de parada que detiene la adquisición sin modificar la parametrización establecida en el contexto de la sesión, permitiendo reanudar con las mismas condiciones al momento que sea necesario. El cierre de la sesión libera el puerto y los recursos asociados y deja el sistema en estado limpio para un nuevo ciclo.

La interfaz pública puede modelarse mediante un diagrama de clases, entendido como una representación gráfica que define tipos, atributos y operaciones, además de las asociaciones que existen entre ellos. En el diagrama mostrado en la Figura 14, la clase ADS1298R actúa como fachada de la API, conserva el puerto de comunicación, la tasa global de muestreo y la cantidad de canales, además, gobierna el ciclo de vida de la sesión con operaciones de alto nivel para parametrizar el puerto, inicializar la comunicación, iniciar y finalizar la adquisición, obtener bloques de datos y cerrar la sesión.

La clase UART abstrae la comunicación física y provee apertura y cierre del puerto, lectura, escritura y consulta de disponibilidad, apoyada en una clase tipo enumerado BaudRate que documenta las tasas probadas sin impedir el uso de valores enteros cuando sea necesario. La clase Channel concentra la configuración por canal, mantiene el estado de habilitación, la ganancia en voltaje del PGA (Amplificador de ganancia programable) y el modo de entrada ya sea común o diferencial. La tasa de muestreo es global y se representa con una clase de tipo enumerada SampleRate asociada a ADS1298R, lo que preserva la sincronía entre canales. La clase Block es el contenedor de datos que consume la aplicación, incorpora la marca temporal de cada trama, el vector de estado de 24 bits por muestra y la matriz de enteros de 32 bits.

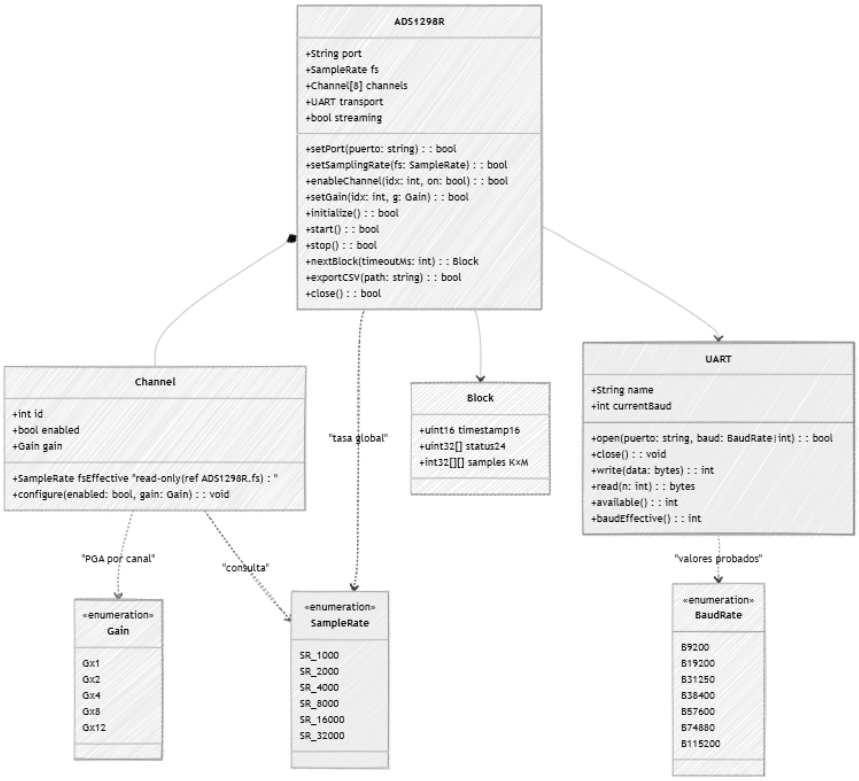


Figura 15. Diagrama de clases de la interfaz de programación de aplicaciones ADS1298R.dll
Fuente: diseño propio haciendo uso del software online Mermaid Diagramming and charting tool

6.2.3 Concurrencia y buffers. La adquisición de datos de forma continua es un proceso que exige separar con claridad quién produce datos y quién los consume. La API resuelve este requerimiento mediante el bloque de comunicación que actúa como productor y una interfaz

pública que opera como consumidor. La etapa de comunicación lee el puerto físico, recibe las tramas de datos y los almacena en un buffer diseñado para adquisición en tiempo real. La interfaz pública, por su parte, extrae bloques completos cuando la aplicación los solicita o los recibe mediante un mecanismo de entrega inmediata, según el modo elegido por el usuario.

El almacenamiento intermedio se organiza en un buffer preasignado en memoria, dimensionado en función de la tasa de muestreo y el tamaño de cada muestra. Este buffer evita asignaciones dinámicas durante la captura y reduce el estrés del sistema de memoria del equipo de cómputo. Para reducir la latencia y evitar bloqueos, la preparación del buffer contempla dos niveles, en primer lugar, el enlace, en el cual se respeta el formato con el que el dispositivo emite las tramas TLV, así mismo, a nivel de entrega se agrupan segmentos en ventanas o intervalos de tiempo adecuados para que los procesos de graficar y almacenar en disco trabajen con estabilidad sin penalizar la sensación de fluidez. Esta estrategia se fija al inicio de la sesión, lo que permite calcular de antemano la capacidad del buffer y verificar que la aplicación disponga de margen suficiente para funcionar dentro de condiciones adecuadas.

Por otra parte, el proceso de coordinación entre productor y consumidor evita esperas innecesarias. El bloque de comunicación se activa solo cuando llega nueva información desde el puerto, la interfaz pública ofrece una operación de lectura con tiempo de espera configurable que retorna de inmediato si no hay bloques listos, preservando la capacidad de la aplicación para realizar otras tareas mientras la comunicación está en pausa. En el modo de entrega inmediata, el usuario registra un receptor y la biblioteca almacena en éste la información en cuanto un bloque queda disponible; este modelo es apropiado para graficar continuamente y canalizar el flujo de datos hacia una cola de escritura en segundo plano. Finalmente, si la aplicación no alcanza a consumir al ritmo de producción, el buffer podría colapsar, en este punto la API debe priorizar la continuidad de la sesión y descartar los bloques de datos más antiguos que aún no han sido consumidos.

6.2.4 Interoperabilidad con lenguajes de alto nivel. Esta característica permite que la API pueda funcionar de forma transparente en diversos lenguajes de programación de alto nivel en entornos Windows 10 y 11. En primer lugar, la interoperabilidad de la API ADS1298.dll se

establece por medio de las funcionalidades que ofrece .NET Framework bajo el lenguaje de programación C#, esta elección permite importar las funciones mediante atributos de DllImport y garantiza que el runtime de .NET pueda enlazar en tiempo de carga sin paquetes de software adicionales como librerías, drivers, frameworks, entre otras. Además, la forma de invocar las funciones, la tipificación de las variables y el tamaño exacto de cada estructura quedan fijados en la DLL, lo que evita ambigüedades del traductor y permite que los lenguajes de alto nivel utilicen la API sin necesidad de compilar en cada ejecución, asegurando así la compatibilidad requerida.

6.3 Implementación de la API

Antes de abordar el desarrollo de la API, resulta necesario adecuar el hardware de adquisición y comprobar que éste funciona adecuadamente, en particular, se requiere asegurar la continuidad del enlace SPI y la integridad de las señales. Esta verificación reduce la incertidumbre asociada a fallos de la capa física, aislar correctamente los eventos de comunicación, establecer una base confiable sobre la cual implementar y validar las rutinas de la biblioteca dinámica. En la sección 6.3.1 se describe la preparación del hardware y su validación como prerrequisito para avanzar con el desarrollo de la API.

6.3.1 Adecuación del hardware de adquisición de señales bioeléctricas. Durante esta etapa se decide retirar la tarjeta madre del ADS1298ECGFE-PDK y sustituir su función de interfaz por un módulo ESP32-S3-N16R8 que actúa como maestro SPI frente a la tarjeta hija. El objeto de esta decisión es preservar el acondicionamiento y el ruteo definidos por el fabricante hasta el circuito integrado ADS1298R, evitando rediseñar etapas sensibles y minimizando incertidumbres en el desempeño del front-end analógico, además, se concentra el esfuerzo en la interfaz digital y la captura de las señales. En la Figura 16 se muestra la placa de evaluación utilizada como base, cuyo uso permite mantener la cadena analógica de adquisición sin modificaciones sustanciales.

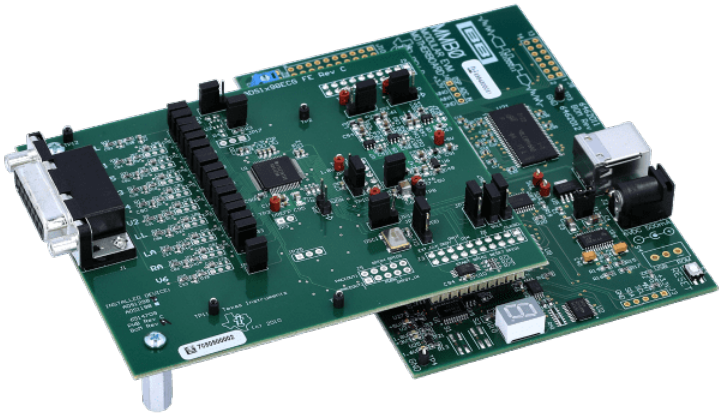


Figura 16. Placa de evaluación ADS1298ECGFE-PDK
Fuente: https://www.ti.com/content/dam/ticom/images/products/ic/dataconverters/evm-board/ads1298ecgfe-pdkangled.png

Elegir la tarjeta hija frente a un PCB (circuito impreso) desde cero implica reducir riesgo técnico y ciclos de iteración en el diseño e implementación, aprovechando los puertos de entrada-salida y puntos de prueba establecidos en la tarjeta hija. Esta elección reduce el problema al desarrollo de un PCB de enlace entre la tarjeta hija y el ESP32-S3-N16R8 con el propósito de eliminar cables tipo Dupont, preservando la integridad del bus digital y de las demás señales al acortar trayectos, controlar impedancias y evitar bucles de masa. Finalmente, surge la necesidad de una base que cumpla la función de fijación mecánica, evitando esfuerzos sobre conectores y estabilizando el conjunto durante las pruebas, lo que facilita montaje, desmontaje y acceso a los puntos de verificación.

El primer paso de esta etapa fue desarrollar el PCB de acoplamiento entre la tarjeta hija y el módulo del ESP32-S3-N16R8, este diseño tuvo como base el proceso de identificación de los puertos de la placa de evaluación realizado por Calderón (2024). Como resultado se obtuvo el diagrama esquemático mostrado en la Figura 17 y el PCB correspondiente observado en la Figura 18, elaborado en la versión de prueba del software Sprint Layout 6.

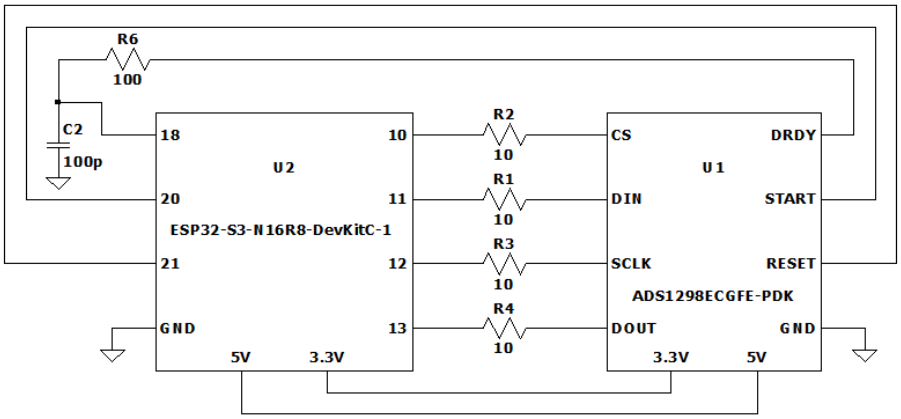


Figura 17. Diagrama esquemático del hardware de adquisición implementado
Fuente: diseño propio haciendo uso del software LTSpice de Analog Devices

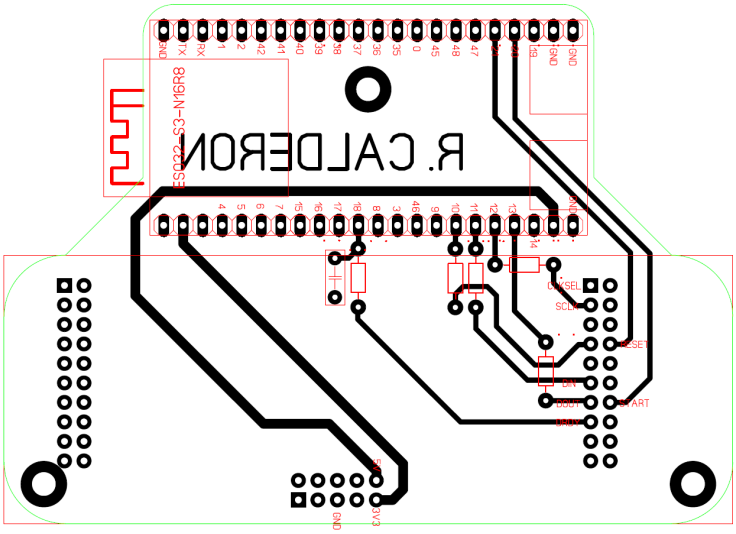


Figura 18. PCB de acoplamiento entre el ESP32-S3-N16R8 y la tarjeta hija ADS1298ECGFE-PDK
Fuente: diseño propio haciendo uso la versión de prueba del software Sprint Layout 6

Una vez finalizado el diseño del circuito impreso, se procede a la fabricación del PCB mediante fresado en máquina CNC de tres ejes. Este proceso permite obtener el trazado de las pistas y la perforación de los orificios de montaje con la precisión requerida para garantizar la correcta alineación entre la tarjeta hija del kit de evaluación y el módulo ESP32-S3-N16R8. La placa se elaboró sobre sustrato FR4 de 1.6 mm de espesor y cobre de 1 oz. La Figura 19 muestra el proceso de fabricación del PCB mediante fresado CNC.

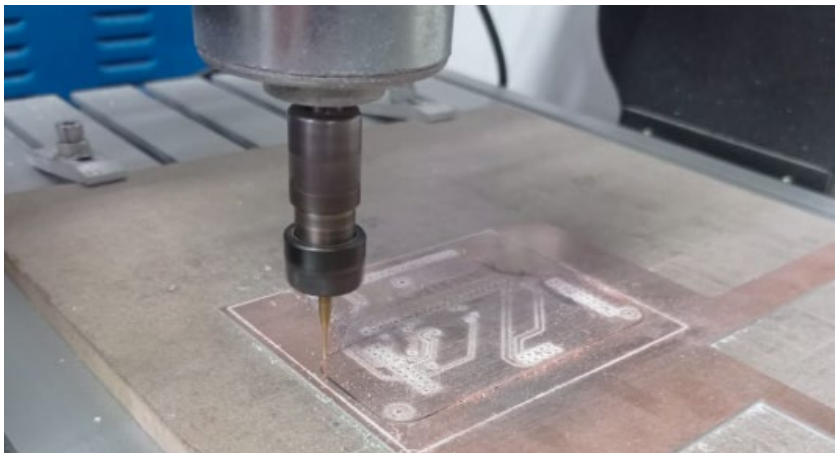


Figura 19. Fabricación del PCB de acoplamiento mediante fresado CNC
Fuente: autor

Con el propósito de asegurar la fijación mecánica y proteger la placa de acoplamiento durante su operación, se diseña una base de soporte utilizando el software Autodesk Fusion, la cual es fabricada en una impresora 3D Creality Ender 6 empleando filamento PLA de 1.75 mm. La estructura incorpora alojamientos específicos para la tarjeta hija, el módulo ESP32-S3-N16R8 y el PCB de enlace, con el objetivo de facilitar el ensamblaje y evitar tensiones sobre los conectores. La Figura 20 muestra el proceso de impresión 3D de la base de soporte.

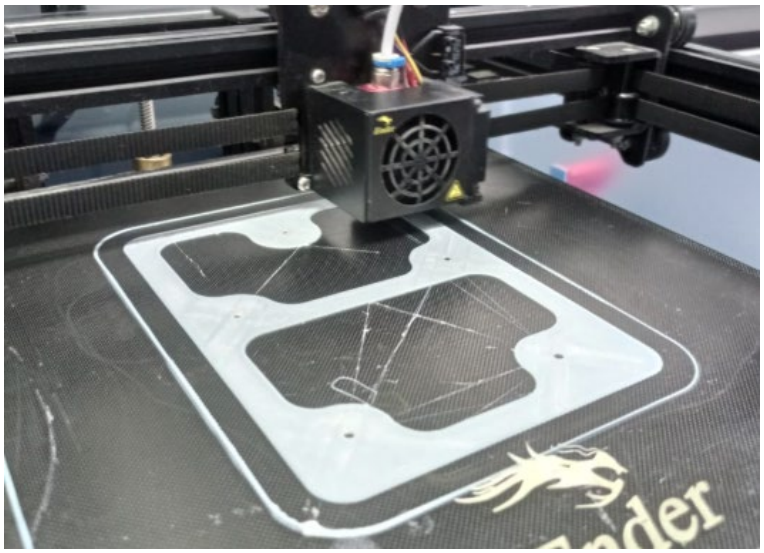


Figura 20. Fabricación de la base mediante impresión 3D
Fuente: autor

Seguidamente, se lleva a cabo el ensamblaje final del hardware, integrando la tarjeta hija del módulo ADS1298ECGFE-PDK, el PCB de acoplamiento y el módulo ESP32-S3-N16R8. La Figura 21 presenta el conjunto final del sistema de adquisición ensamblado y listo para las pruebas de validación.

Figura 21. Ensamble del hardware de adquisición de señales bioeléctricas
Fuente: autor

Durante la validación inicial se decide reproducir el procedimiento de identificación del dispositivo empleado en la investigación de Calderón (2024), solicitando el ID del circuito integrado ADS1298R mediante el comando de lectura del registro 0x00. Para ello, el ESP32-S3-N16R8 actúa como maestro SPI y envía la secuencia de bytes correspondiente a la operación de lectura de registros sobre el bus. Esta prueba se orienta a confirmar la continuidad del enlace, la temporización básica del bus y la respuesta del integrado ante un código de registro conocido, sin introducir todavía capas de abstracción adicionales. La Figura 22 muestra el diagrama de tiempos visualizado en el osciloscopio de las señales SCLK y DIN durante el envío de la trama de comando.

A continuación, se registra la respuesta del circuito integrado observando las señales SCLK y DOUT, verificando que el byte recibido corresponde al esperado para el ADS1298R. La

coincidencia entre el patrón de bits reportados por el circuito integrado y el valor de referencia confirma que la ruta física, el mapeo de pines y el PCB de enlace funcionan dentro de los parámetros esperados, con esta evidencia se cierra la prueba de la capa física y se habilita la siguiente etapa de desarrollo. La Figura 23 presenta el diagrama de tiempos de las señales SCLK y DOUT con el byte de ID recibido.

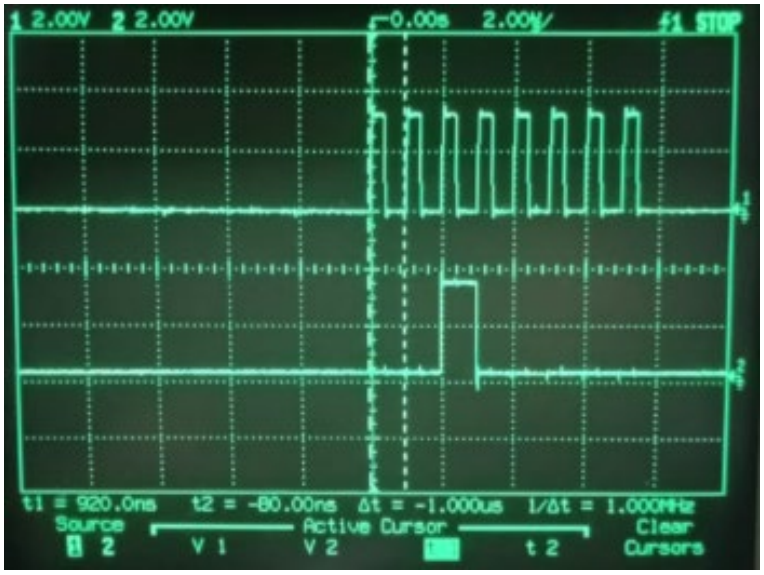


Figura 22. Lectura del registro 0x00 visualizado en osciloscopio Hewlett Packard 54603B
Fuente: autor

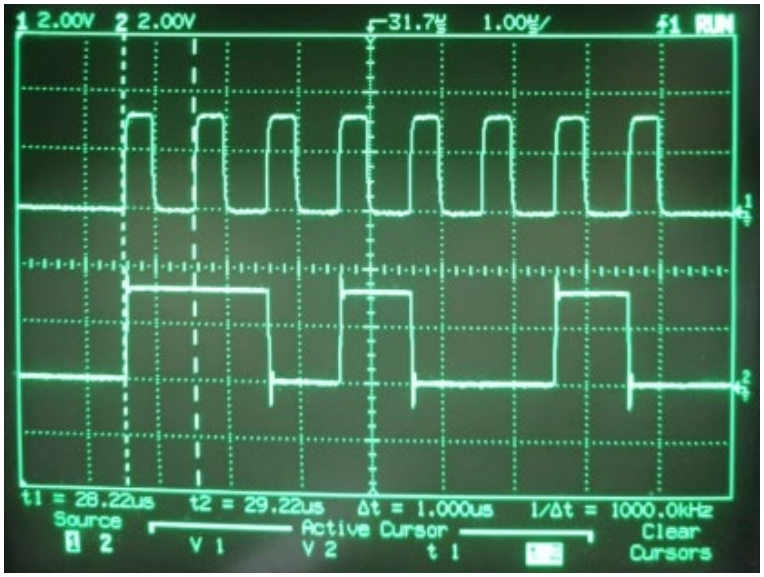


Figura 23. Respuesta del sistema a la lectura del registro 0x00 correspondiente a 0xD2
Fuente: autor

6.3.2 Lectura del ID desde un entorno de alto nivel. Una vez verificada la continuidad eléctrica del enlace entre la tarjeta hija y el módulo de control, se procede a desarrollar el algoritmo para solicitar desde un entorno de programación de alto nivel el ID (identificador) del circuito integrado ADS1298R. Esta prueba tiene como propósito validar la interoperabilidad de los tres componentes fundamentales del sistema: el firmware, la biblioteca de vínculos dinámicos y el script implementado en lenguaje de alto nivel.

El firmware desarrollado cumple la función de puente entre el puerto USB del computador y el conversor USB-UART del módulo ESP32-S3-N16R8, el módulo toma los datos y los envía al circuito integrado ADS1298R por medio del bus SPI, esto se puede observar en la Figura 24. Al iniciar la ejecución, el programa configura el puerto serie del módulo ESP32-S3-N16R8 utilizando los pines físicos GPIO43 y GPIO44 en modo UART. Seguidamente, se definen los pines de control asociados al circuito integrado ADS1298R y se inicializa la interfaz SPI. Posteriormente, el algoritmo realiza un reinicio físico del ADS1298R con el fin de detener cualquier transmisión continua que pudiera interferir con las operaciones de lectura de los registros.

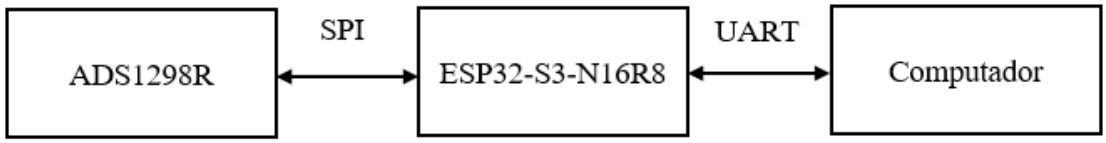


Figura 24. Módulo ESP32-S3-N116R8 como puente entre dos sistemas
Fuente: autor

Una vez establecidas las condiciones iniciales, el microcontrolador entra en un bucle permanente en el cual analiza cada byte recibido por UART, si el comando recibido corresponde a una lectura de registro, el ESP32-S3-N16R8 ejecuta una transacción SPI compuesta por el envío del código de lectura junto con la dirección del registro y la cantidad de bytes solicitados. Luego, el microcontrolador lee el dato del bus SPI y transmite la respuesta de regreso al computador. En la Figura 25 se ilustra el proceso descrito, desde la recepción del comando hasta el envío de la respuesta hacia el host.

Sobre esta base se desarrolla la biblioteca dinámica BioAdqRT.dll, cuyo propósito es concentrar toda la lógica asociada a la comunicación con el microcontrolador y proporcionar una interfaz de programación de alto nivel libre de detalles propios de la capa de transporte. La biblioteca se implementa en lenguaje C# bajo .NET Framework 4.8, seleccionando la clase SerialPort como sistema nativo para el manejo de la comunicación serial.

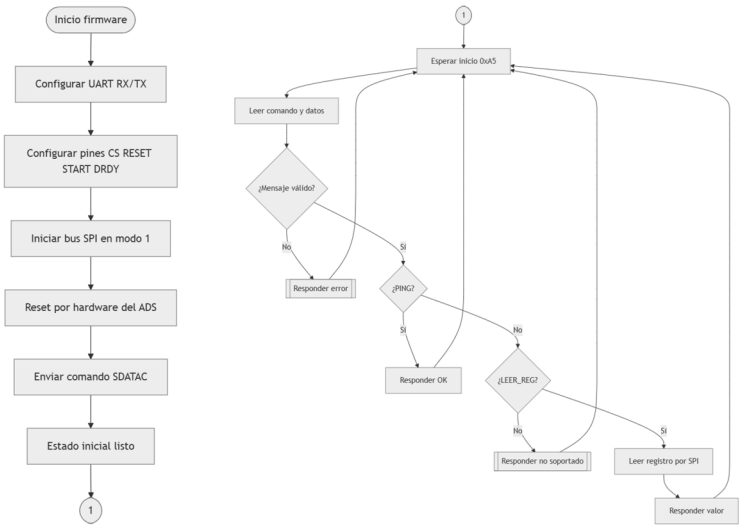


Figura 25. Diagrama de flujo del firmware implementado en C++
Fuente: diseño propio haciendo uso del software online Mermaid Diagramming and charting tool

Al ser invocada por un entorno externo la DLL ejecuta internamente una rutina denominada AutoConnect() que recorre los puertos COM disponibles en el sistema operativo e intenta establecer conexión con cada uno de ellos de forma secuencial. Durante este proceso, la biblioteca envía una trama de prueba tipo PING a cada puerto y espera una respuesta válida proveniente del ESP32-53-N16R8. En el momento en que recibe un encabezado correcto, la DLL identifica el puerto activo, almacena su nombre en memoria y lo mantiene abierto para el resto de la sesión. Esta operación permite que el usuario no tenga que especificar manualmente el puerto ni la velocidad de transmisión, lo que contribuye a mantener el diseño transparente y libre de configuraciones externas.

Una vez establecida la conexión, la biblioteca expone métodos públicos que encapsulan las funciones de lectura y verificación del dispositivo, como Ping(), ReadRegister() y ReadIdRaw(). Cuando se solicita la lectura del identificador, la función ReadIdRaw() construye una trama

binaria con los campos requeridos por el protocolo, envía la información al ESP32-53-N16R8 y espera la respuesta. Al recibirla, la biblioteca verifica el encabezado, el código de estado, la longitud de los datos y la suma de comprobación; si todos los valores son coherentes, el método devuelve el byte leído desde el registro 0x00. En caso contrario, se genera una excepción que evita la propagación de errores al entorno de usuario.

Finalmente, la DLL incluye una función adicional denominada Decodeld(), encargada de traducir el valor hexadecimal recibido a una etiqueta legible que identifica la versión del circuito integrado. En la Figura 26 se presenta el diagrama de flujo de ejecución de la biblioteca BioAdqRT.dll, este representa la secuencia de operaciones realizadas por la biblioteca desde la autodetección del puerto hasta la validación de la respuesta enviada por el módulo de control.

El nivel superior se implementa mediante un script en lenguaje Python, concebido para validar el correcto funcionamiento de la DLL desde un entorno de análisis de datos de uso común. El programa inicia verificando la existencia del archivo BioAdqRT.dll en el directorio correspondiente, seguidamente, se importa la biblioteca pythonnet que permite interactuar directamente con archivos DLL desarrollados en .NET, y se carga la clase ADS1298 desde la biblioteca BioAdqRT.

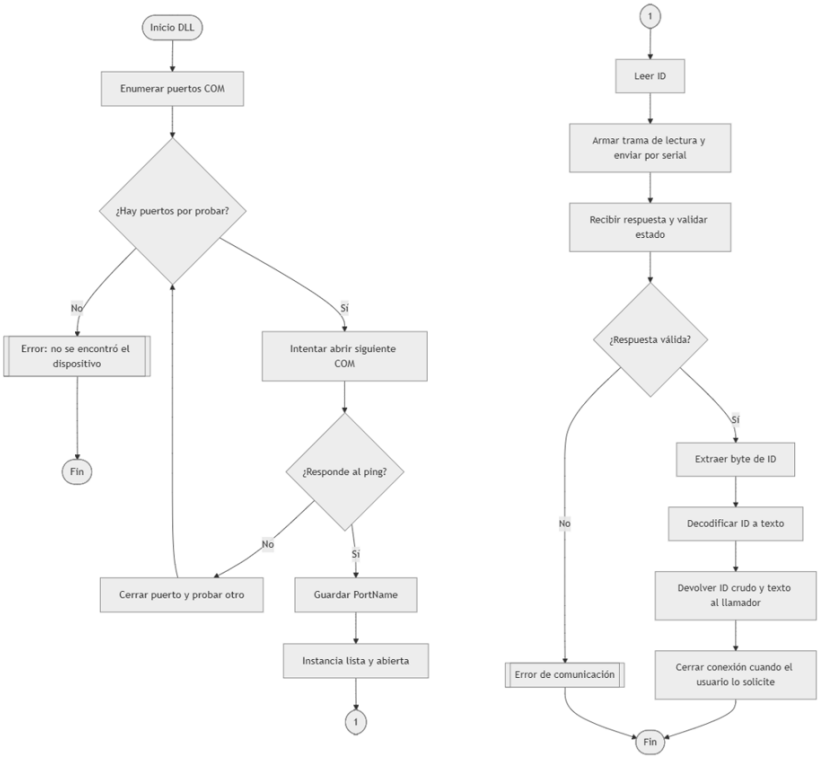


Figura 26. Diagrama de flujo de la biblioteca de vínculos dinámicos implementada en C#
Fuente: diseño propio haciendo uso del software online Mermaid Diagramming and charting tool

En este punto el script desarrollado en Python invoca el método AutoConnect() de la DLL, el cual realiza la detección automática del puerto y retorna una instancia abierta y lista para ser utilizada. A partir de ese momento, el usuario puede ejecutar las funciones Ping() y ReadIdRaw() con el fin de comprobar la conexión y leer el identificador del circuito integrado ADS1298R, el resultado de estas funciones se muestra en la consola en forma hexadecimal junto con su interpretación textual generada por la función Decodeld(). Para garantizar el cierre correcto del puerto, el programa ejecuta la función Dispose() antes de finalizar, liberando los recursos asociados a la sesión. En la Figura 27 se muestra el diagrama de flujo general del script en Python para la lectura del identificador desde un entorno de alto nivel.

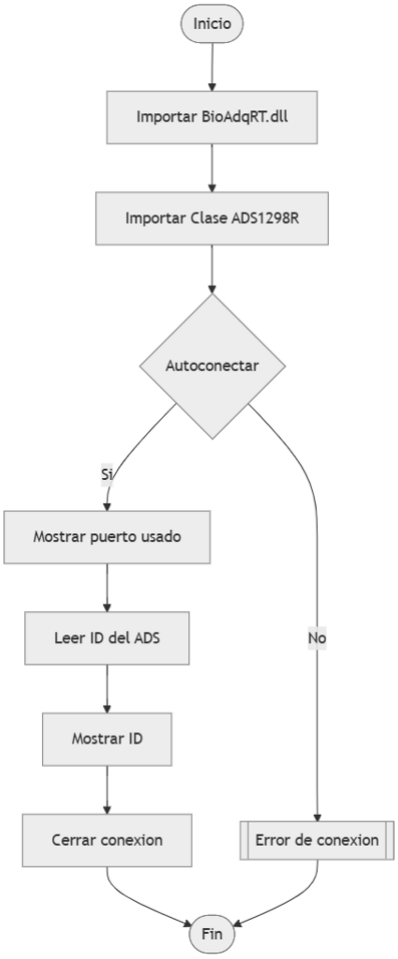


Figura 27. Diagrama de flujo del script implementado en Python
Fuente: diseño propio haciendo uso del software online Mermaid Diagramming and charting tool

El resultado mostrado en la consola representa la evidencia funcional de la comunicación entre las tres capas del sistema. En la salida se observa la respuesta emitida por la biblioteca BioAdqRT.dll tras ejecutar la lectura del registro 0x00, donde el valor reportado corresponde al identificador 0xD2, propio del circuito integrado ADS1298R, este resultado confirma que la trama transmitida desde Python fue correctamente interpretada por la DLL, enviada al módulo ESP32-S3-N16R8, ejecutada en el bus SPI y devuelta de forma íntegra hasta el entorno de alto nivel. La coincidencia entre el valor obtenido y el esperado valida la correcta implementación del protocolo, la sincronización de los niveles de software y la funcionalidad de la capa física del sistema. En la Figura 28 se presenta la salida obtenida en la consola, la cual muestra la respuesta final del proceso de identificación.

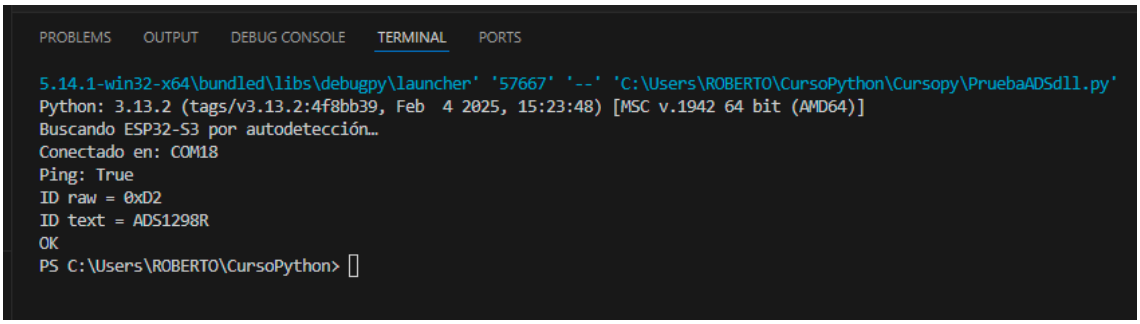


Figura 28. Respuesta del sistema a la lectura del ID mediante Python
Fuente: Autor

6.4 Evaluación de desempeño

Todas las pruebas descritas en esta sección se realizaron sobre un equipo de escritorio de características modestas, con sistema operativo Windows 10 Pro de 64 bits, procesador Intel Core i3-5005U a 2,0 GHz y 8 GB de memoria RAM instalada.

Previo a la evaluación de desempeño se llevó a cabo una validación funcional orientada a comprobar que la cadena completa ESP32-S3-N16R8, el circuito integrado ADS1298R, la API DLL y el script en Python operara de manera coherente. En esta fase se verificó que el puente de firmware respondiera a los comandos de lectura y escritura de registros, que la biblioteca dinámica fuera capaz de encontrar automáticamente el puerto serie activo y que, desde el entorno de alto nivel, el usuario pudiera inicializar el sistema, activar la adquisición continua y cerrar la sesión sin recurrir a instrucciones de bajo nivel. Alcanzado este punto de estabilidad funcional se procedió a evaluar el comportamiento de la API en condiciones de operación durante una hora y 50 minutos, prestando atención a la escritura correcta de los registros de configuración, al mecanismo de streaming, al uso de recursos de cómputo y a la capacidad del sistema para sostener una sesión sin bloqueos aparentes.

6.4.1 Escritura de CONFIG1. El primer aspecto evaluado fue la correcta transmisión de los datos a fin de realizar cambios en la configuración desde el entorno de alto nivel hasta el circuito integrado de adquisición. Para ello se empleó el registro CONFIG1 del circuito integrado ADS1298R, que agrupa parámetros importantes como la frecuencia global de muestreo. La prueba se ejecutó completamente desde el script en Python, invocando un método de alto nivel

de la DLL encargado de fijar la tasa de muestreo requerida para las pruebas, en este caso 250 muestras por segundo, sin exponer al usuario los detalles del comando WREG ni la dirección física del registro.

6.4.2 Streaming. Una vez verificada la escritura de CONFIG1 se evaluó el comportamiento del sistema en modo de adquisición continua. La lógica del firmware del ESP32-S3-N16R8 se mantuvo lo más simple posible y se limitó a leer en cada transición a nivel bajo de la línea DRDY una trama completa de 27 bytes correspondiente a una conversión del ADS1298R. Estos datos se enviaron al computador a través del enlace serie configurado a 115200 baudios, sin aplicar procesamiento adicional en el microcontrolador.

La lógica de negocio reside en la DLL, que interpreta las tramas, extiende el signo de los datos de 24 bits, asigna cada muestra a su canal correspondiente y expone métodos de lectura de alto nivel. El script de Python consume estos métodos y construye las señales en forma de vectores de tiempo, permitiendo visualizar en una misma ventana las trazas asociadas a los ocho canales habilitados. En las pruebas realizadas con la señal de prueba interna del circuito integrado ADS1298R se constató que el flujo de datos era estable, que no se producían pérdidas evidentes de muestras y que la forma de onda observada en cada canal (Figura 30) coincidía con la esperada según la configuración establecida. A continuación, se realiza la descripción del proceso de evaluación de desempeño.

6.4.3 Latencia. En esta fase del proyecto no se realizaron mediciones cuantitativas de latencia de extremo a extremo para estimar retrasos entre la conversión analógica digital y la representación gráfica en pantalla. El análisis de latencia se abordó de manera cualitativa, observando el comportamiento del trazado en tiempo real durante las sesiones de streaming. En las condiciones de prueba empleadas, con ocho canales activos a 250 muestras por segundo utilizando la señal de prueba interna, no se apreciaron acumulaciones visibles de muestras ni retardos crecientes en la visualización. Para los fines exploratorios de este trabajo la latencia percibida puede considerarse aceptable; no obstante, una caracterización precisa de este parámetro se identifica como trabajo futuro para etapas en las que se evalúen aplicaciones que requieran sincronización estricta con otros sistemas.

6.4.4 Uso de CPU. El consumo de CPU se evaluó de forma práctica mediante la observación directa del Administrador de tareas de Windows mientras se ejecutaba una sesión de adquisición continua. En la configuración de prueba, con los ocho canales activos a 250 muestras por segundo y la graficación en tiempo real habilitada desde Python, el proceso asociado al intérprete de Python se mantuvo alrededor del 30% de utilización de CPU, mientras que la carga total del sistema osciló cerca del 37%. En la captura de pantalla correspondiente a la Figura 29 se aprecia que la columna de consumo de energía clasifica el proceso de Python como de consumo muy alto, en contraste con el resto de las aplicaciones que permanecen en niveles muy bajos.

Nombre	Estado	37% CPU	72% Memoria	2% Disco	0% Red	0% GPU	Motor de GPU	Consumo de energía	Tendencia de consumo de e
Aplicaciones (7)									
Administrador de tareas		1,1%	31,6 MB	0 MB/s	0 Mbps	0%		Muy baja	Muy baja
Explorador de Windows		0,4%	32,6 MB	0 MB/s	0 Mbps	0%		Muy baja	Muy baja
Google Chrome (10)		2,4%	1.131,1 MB	0,1 MB/s	0 Mbps	0%	GPU 0 - 3D	Muy baja	Muy baja
Microsoft Visual Studio 2022		0%	131,0 MB	0 MB/s	0 Mbps	0%		Muy baja	Muy baja
Microsoft Word (2)		0%	64,2 MB	0 MB/s	0 Mbps	0%		Muy baja	Muy baja
Python		29,9%	33,7 MB	0 MB/s	0 Mbps	0%		Muy alta	Bajo
Visual Studio Code (4)		0%	201,2 MB	0 MB/s	0 Mbps	0%		Muy baja	Muy baja

Figura 29. Consumo de recursos durante la ejecución de la prueba
Fuente: Autor

Este resultado debe interpretarse a la luz del hardware disponible. El procesador Intel Core i3-5005U es un modelo de bajo consumo pensado para equipos portátiles de entrada, por lo que un 30% de utilización implica que una fracción importante de la capacidad de cómputo disponible se dedica a la graficación en tiempo real y al manejo del entorno interactivo. Aun así, durante las pruebas el sistema se mantuvo plenamente operativo y fue posible utilizar otras aplicaciones como el navegador, el editor de código y el procesador de texto sin bloqueos ni retardos críticos. Esto sugiere que la carga añadida por la API y por el flujo de datos de ocho canales es compatible con un equipo de escritorio convencional, aun cuando no se trate de una plataforma de altas prestaciones.

6.4.5 Uso de RAM. El mismo registro del Administrador de tareas permitió estimar el uso de memoria principal durante las pruebas. En la captura asociada (Figura 29) puede observarse que, aunque la memoria total del sistema se sitúa alrededor del 70% de ocupación, el proceso de

Python responsable de la graficación en tiempo real utiliza del orden de 30 a 40 MB de RAM. Considerando que el equipo dispone de 8 GB instalados, esta cifra representa una fracción reducida de la memoria total y resulta comparable o incluso inferior al consumo de otras aplicaciones abiertas, como el navegador web o el entorno de desarrollo.

Desde el punto de vista de la API, este comportamiento es coherente con la estrategia adoptada en el diseño, en la que se privilegia el manejo de buffers de tamaño acotado (27 Bytes por trama) y la reutilización de estructuras de datos, evitando acumulaciones de muestras que puedan conducir a un crecimiento indefinido del consumo de memoria. En consecuencia, en las condiciones de prueba empleadas, la API no introduce un impacto crítico sobre la memoria disponible y deja un margen razonable para que se ejecuten de manera simultánea otras herramientas de análisis o documentación en un equipo con 8 GB de RAM.

6.4.6 Pruebas de estrés. Finalmente se realizó una prueba de estrés orientada a comprobar la estabilidad del sistema en un escenario de adquisición prolongada sobre el equipo descrito. Para ello se configuró el circuito integrado ADS1298R en modo de prueba interna, habilitando los ocho canales en configuración diferencial y fijando la frecuencia de muestreo en 250 muestras por segundo mediante el registro CONFIG1. A continuación se inició el streaming continuo y se mantuvo la adquisición durante aproximadamente una hora y 50 minutos, lo que supone del orden de 1650000 muestras por canal y más de 13000000 de muestras en total a través de la interfaz UART.

Durante toda la sesión no se observaron bloqueos del ESP32-S3-N16R8 ni desconexiones del puerto serie, y la DLL no reportó excepciones asociadas a fallos de protocolo o a tramas incompletas. El script de Python continuó recibiendo y graficando los datos sin interrupciones aparentes, a pesar de ejecutarse sobre un procesador Intel Core i3 con 8 GB de RAM y con otras aplicaciones de uso cotidiano abiertas en segundo plano. Como evidencia de esta estabilidad, la Figura 30 muestra una imagen de las ocho señales en un instante avanzado de la transmisión, donde se aprecia la señal de prueba interna reproducida en todos los canales y sin discontinuidades atribuibles a pérdidas de muestras.

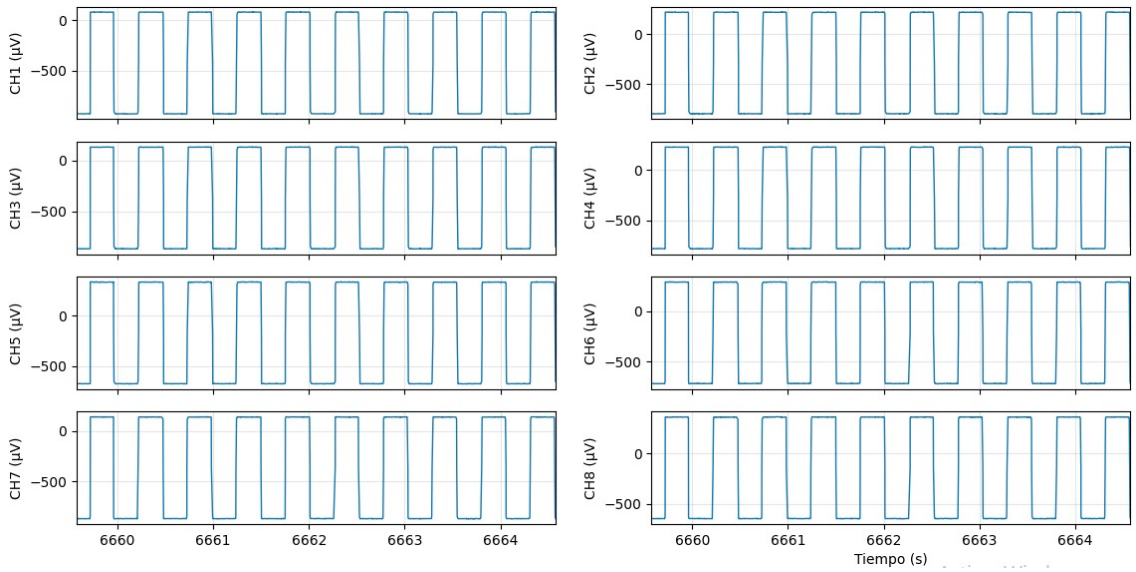


Figura 30. Visualización de la señal de prueba en los ocho canales del sistema de adquisición
Fuente: Autor

Por otra parte, se realizó una prueba específica con señales externas en la que se emplearon cinco generadores de funciones GW Instek SFG-2110 conectados de manera independiente a cinco canales de entrada del circuito integrado ADS1298R, manteniendo la misma cadena de adquisición basada en el ESP32-S3-N16R8, la biblioteca de vínculos dinámicos y el script de Python desarrollado en el proyecto. En la Figura 31 se presenta el montaje físico del sistema de adquisición junto con los generadores, donde se observa la disposición de la tarjeta de adquisición ADS1298ECGFE-PDK, el módulo ESP32-S3-N16R8 y el cableado correspondiente. En la Figura 32 se muestra la traza registrada por el osciloscopio HP 54603B para una de las señales de prueba. Las salidas de los generadores se configuraron en el orden de los milivoltios, lo que hace que estas señales sean especialmente vulnerables al ruido eléctrico del entorno y a los acoplamientos indeseados propios de un laboratorio académico, condición que resulta similar a la que se encuentra habitualmente en la adquisición de bioseñales reales.

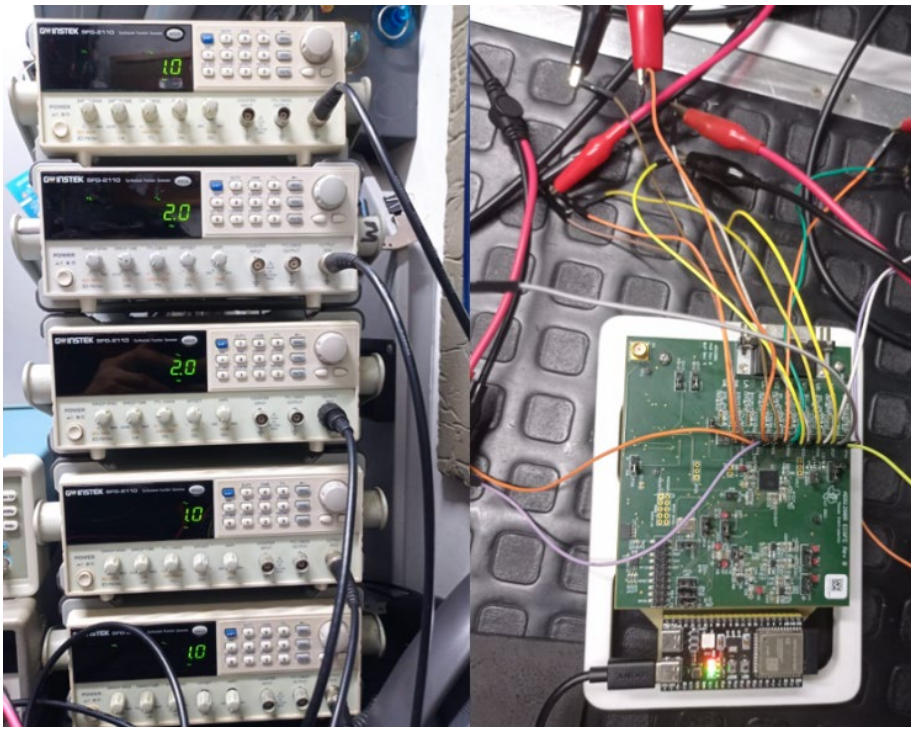


Figura 31. Prueba de adquisición con cinco señales externas
Fuente: Autor

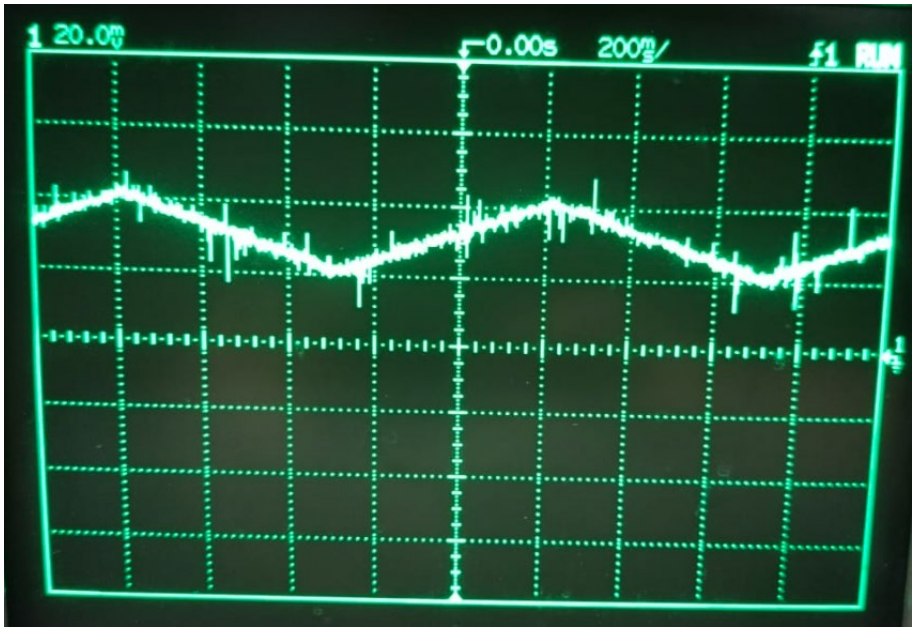


Figura 32. Señal de prueba externa visualizada en osciloscopio HP 54603B
Fuente: Autor

La comparación entre la forma de onda observada en el osciloscopio HP 54603B y las señales reconstruidas a través de la API en el entorno de Python puso en evidencia una ventaja importante del sistema de adquisición basado en el circuito integrado ADS1298R, ya que en el instrumento de medida convencional la señal se observa contaminada por componentes de ruido de alta frecuencia. En la Figura 33 se muestran las cinco trazas asociadas a los generadores de funciones GW Instek SFG-2110 visualizadas en Visual Studio Code haciendo uso de la librería Matplotlib de Python. La gráfica superior derecha correspondiente al canal dos (CH2) es la misma a la mostrada en la Figura 32. Estas señales se presentan más limpias y estables gracias al rechazo en modo común que proveen los amplificadores de instrumentación que hacen parte de la arquitectura de entrada diferencial del circuito integrado ADS1298R.

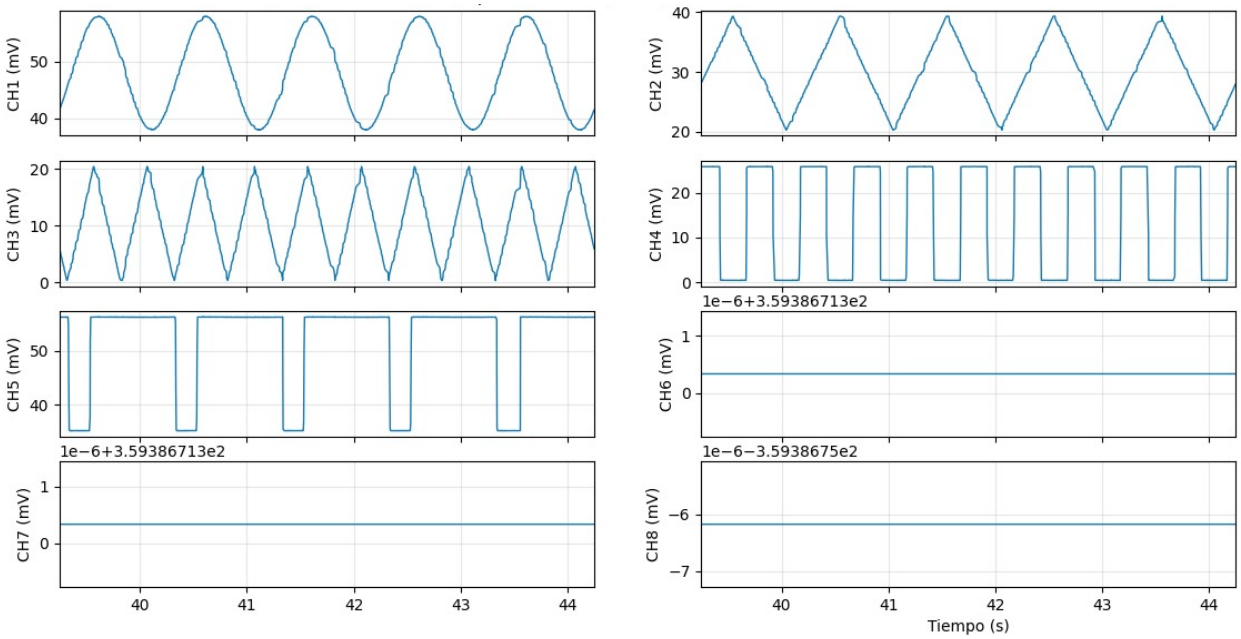


Figura 33. Visualización de cinco señales de prueba externas utilizando la librería Matplotlib
Fuente: Autor

En la Figura 33 se distinguen cinco canales con señales distintas, coherentes con la configuración aplicada en cada generador, y tres canales adicionales sin contenido útil debido a la ausencia de más equipos de prueba, lo que confirma simultáneamente la correcta asignación de canales, la robustez frente al ruido del entorno y la capacidad del sistema para manejar varias fuentes externas de baja amplitud.

Con esta prueba se considera que, dentro del alcance definido para el presente trabajo, la interfaz de programación desarrollada es capaz de sostener un régimen de adquisición continuo de ocho canales en un equipo de escritorio convencional y sin errores visibles en la transmisión de datos. Los detalles lógicos de la implementación en firmware, en la biblioteca de vínculos dinámicos y en el script de Python se resumen en los pseudocódigos incluidos en los Anexos A, B y C, que sirven como referencia para futuras extensiones o migraciones a otros entornos de ejecución.

7. Conclusiones

El presente trabajo tuvo como propósito desarrollar una interfaz de programación de aplicaciones en formato DLL para la gestión del circuito integrado ADS1298R, orientada a aplicaciones bioeléctricas. Los resultados obtenidos permiten afirmar que es posible desacoplar la complejidad del front end analógico del fabricante de la lógica de control que requiere el usuario final, proporcionando un conjunto de métodos que abstraen los detalles de los registros, del protocolo de comunicación y que pueden ser invocados desde entornos como Python. El sistema implementado demuestra que, aun partiendo de un único módulo de adquisición disponible, es viable construir una capa de software reutilizable que extienda la vida útil del hardware y lo integre con plataformas modernas de análisis de datos.

Durante el desarrollo de este trabajo, se logró definir un conjunto claro de requerimientos funcionales y no funcionales para la API, a partir del análisis de la hoja de datos del ADS1298R, del estado del arte y de las necesidades concretas expuestas por los asesores del proyecto. Este ejercicio permitió identificar restricciones asociadas a la tasa de muestreo, al número de canales, al formato de las tramas de datos y a la capa física de comunicación, así como aspectos de mantenibilidad y escalabilidad que debían quedar reflejados en el diseño. La formalización de estos requerimientos se convirtió en la guía principal para las decisiones posteriores de arquitectura y para la delimitación del alcance del proyecto, evitando desviaciones hacia funcionalidades que no aportan valor directo al propósito de investigación.

Seguidamente, se estableció una arquitectura por capas que separa con claridad las responsabilidades del firmware, de la biblioteca dinámica y del entorno de alto nivel. El microcontrolador ESP32-S3-N16R8 se limita a actuar como puente entre el bus SPI del circuito integrado ADS1298R y el puerto serie del computador, con una lógica orientada a la gestión de comandos básicos y al envío de tramas de adquisición. La DLL concentra la lógica de negocio, incluyendo la configuración de registros como CONFIG1, CONFIG2 y CONFIG3, la selección de modos de prueba interna, la conversión de las palabras de 24 bits a valores enteros con signo y la exposición de métodos de alto nivel que ocultan los detalles del protocolo.

En este mismo contexto, se desarrolló un script en lenguaje Python que se encarga de ejecutar casos de prueba y visualizar los datos en tiempo real. Esta organización no solo cumple con los criterios de confiabilidad y seguridad planteados, sino que deja preparado el camino para futuras extensiones hacia otros lenguajes y hacia interfaces gráficas más elaboradas.

Finalmente, la evaluación de desempeño mostró que la solución propuesta es capaz de sostener un régimen continuo (una hora y 50 minutos) de adquisición de ocho canales a 250 muestras por segundo utilizando la señal de prueba interna del circuito integrado ADS1298R, sin evidencias de bloqueos ni pérdidas de datos. A pesar de ejecutarse sobre un equipo con procesador Intel Core i3 y 8GB de memoria RAM, el sistema mantuvo un consumo de recursos compatible con el trabajo en un entorno de laboratorio, concentrando la mayor carga en el proceso de Python encargado de la graficación en tiempo real. Si bien la latencia no se caracterizó de forma cuantitativa, la observación de las gráficas no reveló retrasos crecientes ni acumulación visible de muestras, lo que resulta adecuado para aplicaciones de exploración, investigación y docencia.

8. Recomendaciones

A partir de los resultados obtenidos en el desarrollo de la API, se plantean varias recomendaciones orientadas a consolidar y ampliar el alcance de este trabajo. En primer lugar, se sugiere evolucionar el canal de comunicación hacia interfaces de mayor ancho de banda, como USB nativo de alta velocidad o Ethernet, manteniendo la misma filosofía de diseño por capas. Esto permitiría incrementar la frecuencia de muestreo y facilitar la integración de varios circuitos integrados ADS1298R en paralelo, sin que el enlace de comunicaciones se convierta en el principal cuello de botella del sistema. Cualquier migración de este tipo debería preservar la idea de un firmware mínimo, encargado únicamente de transportar tramas y concentrar la lógica de negocio en la biblioteca de alto nivel.

En cuanto al software, es recomendable fortalecer los mecanismos de prueba automatizada. Si bien en este proyecto se realizaron verificaciones manuales apoyadas en listas de comprobación internas, una siguiente etapa debería incorporar pruebas unitarias para los métodos de la DLL y pruebas de integración que simulen sesiones de adquisición completas. La inclusión de un conjunto de casos de prueba reproducibles, que cubran configuración de registros, lectura de canales y manejo de errores de comunicación, facilitaría el mantenimiento del código y reduciría el riesgo de introducir regresiones al añadir nuevas funcionalidades, como filtros digitales, cálculo de métricas en línea o soporte para otros miembros de la familia ADS129x.

Desde la perspectiva de la aplicación a bioseñales, se recomienda complementar las pruebas de desempeño con ensayos utilizando señales electromiográficas reales adquiridas sobre sujetos humanos, siguiendo las normas éticas vigentes y las buenas prácticas en instrumentación biomédica. Estas pruebas deberían incluir la evaluación de la calidad de la señal, la respuesta ante movimientos musculares y la interacción con etapas de acondicionamiento analógico externas. De manera paralela, resultaría conveniente documentar y, en lo posible, verificar formalmente el cumplimiento de requisitos de seguridad eléctrica y de aislamiento que se requieren cuando el sistema se conecta a seres humanos, incluso si en esta fase el uso se ha limitado a entornos de laboratorio y a señales de prueba internas.

Es importante mencionar que el proyecto se centró en una primera versión funcional de la API, capaz de configurar el circuito integrado ADS1298R, habilitar el modo de prueba interna, adquirir ocho canales a baja velocidad y exponer estos datos a través de una DLL invocable desde Python en un entorno Windows. No se desarrolló una interfaz biomédica completa ni se pretendió validar el sistema para uso diagnóstico, tampoco se exploraron todas las combinaciones posibles de frecuencias de muestreo, ganancias y modos de operación del circuito integrado. Este trabajo describió una plataforma de adquisición y ensayo orientada a la investigación y a la docencia, que sirvió como base sobre la cual construir soluciones más completas, aunque por sí sola no sustituye a un equipo médico certificado ni cubre todo el espacio de funcionamiento del circuito integrado ADS1298R.

A partir de este alcance, la recomendación es continuar enriqueciendo la API hasta cubrir de manera progresiva el conjunto de funcionalidades necesarias para aplicaciones de electromiografía, electrocardiografía, electroencefalografía y otras modalidades afines. Esto implica incorporar métodos de alto nivel para gestionar diferentes rangos de ganancia, modos de entrada monopolar y diferencial, detección de desprendimiento de electrodos, manejo de señales de marcapasos, configuración de filtros internos y externos, marcado de eventos y sincronización con otros dispositivos. Manteniendo la misma estructura por capas, estas ampliaciones permitirán que la API se convierta en una herramienta genérica de adquisición de bioseñales basada en la familia ADS129x, reutilizable en futuros trabajos de grado y proyectos de investigación.

9. Referencias bibliográficas

Aly, H., & Youssef, S. M. (2023). Bio-signal based motion control system using deep learning models: A deep learning approach for motion classification using EEG and EMG signal fusion. Journal of Ambient Intelligence and Humanized Computing, 14, 991–1002. https://doi.org/10.1007/s12652-021-03351-1

American Thoracic Society/European Respiratory Society. (2002). ATS/ERS statement on respiratory muscle testing. American Journal of Respiratory and Critical Care Medicine, 166(4), 518–624. https://doi.org/10.1164/rccm.166.4.518

Analog Devices. (2024). Instrumentation amplifiers. Recuperado el 27 de abril de 2024, de https://www.analog.com/en/product-category/instrumentation-amplifiers.html

Andrushevich, A., Biallas, M., Kistler, R., Rumiński, J., Bujnowski, A., & Wtorek, J. (2017). Open smart glasses development platform for AAL applications. En 2017 Global Internet of Things Summit (GIoTS) (pp. 1–6). https://doi.org/10.1109/GIOTS.2017.8016262

Asheghabadi, A. S., Moqadam, S. B., & Xu, J. (2021). Multichannel sensor system based on flexible architecture for IoT applications. IEEE Sensors Journal, 21(6), 8184–8193. https://doi.org/10.1109/JSEN.2021.3051070

Axelson, J. (2007). Serial port complete: The developer's guide (2nd ed.). Lakeview Research.

Balaji, M. S. P., Sivaraju, S. S., Britto, K. A., Fairooz, S., Elwin Charly, A., & Sivanantham, B. (2021). Cloud-based healthcare monitoring system using IoT. En 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV) (pp. 1728–1731). https://doi.org/10.1109/ICIRCA51532.2021.9544843

Bajwa, J., Munir, U., Nori, A., & Williams, B. (2021). Artificial intelligence in healthcare: Transforming the practice of medicine. Future Healthcare Journal, 8(2), e188–e194. https://doi.org/10.7861/fhj.2021-0095

Barabino, G., Pani, D., Dessì, A., & Raffo, L. (2015). A configurable biopotentials acquisition module suitable for fetal electrocardiography studies. En 2015 IEEE International Symposium on Medical Measurements and Applications (MeMeA) (pp. 479–483). https://doi.org/10.1109/MeMeA.2015.7145251

Barreto, P., & Prieto, M. (2022). Prototipo de adquisición y procesamiento paralelo para señales de electromiografía. Trabajo de grado, Universidad El Bosque.

Batko, K., & Ślęzak, A. (2022). The use of big data analytics in healthcare. Journal of Big Data, 9(1). https://doi.org/10.1186/s40537-021-00553-4

Belaid, S., Hattay, J., Mohamed, H. H., & Rezgui, R. (2022). Deep cardiac telemonitoring for clinical cloud healthcare applications. Procedia Computer Science, 207, 2843–2852. https://doi.org/10.1016/j.procs.2022.09.342

Bernal Sánchez, J. J., & Wilches Luna, E. C. (2021). Electromiografía de superficie (EMGs) en pacientes adultos en cuidados intensivos: Revisión exploratoria. Revista Colombiana de Medicina Física y Rehabilitación, 31(1), 11–28. https://doi.org/10.28957/rcmfr.v31n1a1

Biasiucci, A., Franceschiello, B., & Murray, M. M. (2019). Electroencephalography. Current Biology, 29(3), R80–R85. https://doi.org/10.1016/j.cub.2018.11.060

Bombarda, A., Bonfanti, S., Galbiati, C., Gargantini, A., Pelliccione, P., & Tichy, M. (2022). Software analytics for medical devices: A systematic approach. Information and Software Technology, 152, 107061. https://doi.org/10.1016/j.infsof.2022.107061

Bronzino, J. D., & Peterson, D. R. (2014). Biomedical signals, imaging, and informatics. CRC Press.

Calderón Bermejo, R. J. (2024). Desarrollo de una aplicación de escritorio para comunicación con el módulo ADS1298RECGFE-PDK. Institución Universitaria Pascual Bravo.

Carr, J. J., & Brown, J. M. (2001). Introduction to biomedical equipment technology (4th ed.). Pearson.

Cardoso, P., Datia, N., & Pato, M. P. M. (2017). Integrated electromyography visualization with multi-temporal resolution. En 2017 11th International Symposium on Medical Information and Communication Technology (ISMICT) (pp. 91–95). https://doi.org/10.1109/ISMICT.2017.7891775

Cook, A. J., Gargiulo, G. D., Lehmann, T., & Hamilton, T. J. (2015). Open platform, eight-channel, portable biopotential and activity data logger for wearable medical device development. Electronics Letters, 51(20), 1641–1643. https://doi.org/10.1049/el.2015.2764

Córdova Martínez, A., Nuin, I., Fernández-Lázaro, D., Latasa, I., & Rodríguez-Falces, J. (2017). Actividad electromiográfica (EMG) durante el pedaleo y su utilidad en el diagnóstico de la fatiga en ciclistas. Archivos de Medicina del Deporte, 34(4), 217–223.

Cossul, S., Andreis, F. R., Favretto, M. A., de Castro Antonio, A., & Marques, J. L. B. (2020). Portable microcontroller-based electrostimulation system for nerve conduction studies. IET Science, Measurement and Technology, 14(7), 695–703. https://doi.org/10.1049/iet-smt.2019.0174

Crispin, L., & Gregory, J. (2009). Agile testing: A practical guide for testers and agile teams. Pearson Education.

Criswell, E. (2010). Cram's introduction to surface electromyography (2nd ed.). Jones & Bartlett Learning.

Cromwell, L., Weibell, F. J., & Pfeiffer, E. A. (1980). Biomedical instrumentation and measurements (2nd ed.). Prentice Hall.

Croatti, A., Longoni, M., & Montagna, S. (2022). Applying telemedicine and digital twins to chronic patient care. Procedia Computer Science, 198, 164–170. https://doi.org/10.1016/j.procs.2021.12.224

Cristea, C., Pasarica, A., Andruseac, G., Dionisie, B., & Rotariu, C. (2015). A wireless ECG acquisition device for remote monitoring of heart rate and arrhythmia detection. En 2015 E-Health and Bioengineering Conference (EHB) (pp. 1–4). https://doi.org/10.1109/EHB.2015.7391543

Dawoud, D. S., & Dawoud, P. M. (2020). Serial communication protocols and standards. En Data acquisition and processing for embedded systems (capítulo de libro).

De, B. (2023). API management. En API-Driven Enterprise (pp. 27–47). Apress. https://doi.org/10.1007/979-8-8688-0054-2_2

De Jager, K., Mentink, M., Lancashire, H., Al-Ajam, Y., Taylor, S., & Vanhoestenberghe, A. (2019). Characterisation of a multi-channel multiplexed EMG recording system: Towards variable electrode configurations. En 2019 IEEE Biomedical Circuits and Systems Conference (BioCAS) (pp. 1–4). https://doi.org/10.1109/BIOCAS.2019.8918710

Dezső, D., Győri, G., & Husi, G. (2021). Development of ECG measurement and processing system in LabVIEW environment. International Review of Applied Sciences and Engineering, 12(1), 76–85. https://doi.org/10.1556/1848.2020.00152

Dick, S., & Volmar, D. (2018). DLL Hell: Software dependencies, failure, and the maintenance of Windows. IEEE Annals of the History of Computing, 40(4), 28–51. https://doi.org/10.1109/MAHC.2018.2877913

Dilip, R., Borole, Y. D., Sumalatha, S., & Nethravathi, H. (2021). Design of IoT-based health monitoring system. En 2021 7th International Conference on Computational Intelligence and Communication Technology (CICT) (pp. 1–7). https://doi.org/10.1109/CITSM52892.2021.9588853

Di Paolo Emilio, M. (2013). Data acquisition systems: From fundamentals to applied design. Springer.

Elvey, C. (1957). Aurora borealis. En Advances in Electronics and Electron Physics (pp. 1–42). https://doi.org/10.1016/S0065-2539(08)60162-6

Enderle, J. (2005). Bioelectric phenomena. En Introduction to biomedical engineering (pp. 627–691). Elsevier. https://doi.org/10.1016/B978-0-12-238662-6.50013-6

Ertekin, E., Günden, B. B., Yilmaz, Y., Sayar, A., Çakar, T., & Arslan, S. Ş. (2022). EMG-based BCI for PiCar mobilization. En 2022 7th International Conference on Computer Science and Engineering (UBMK) (pp. 496–500). https://doi.org/10.1109/UBMK55850.2022.9919502

Favretto, M. A., Cossul, S., Andreis, F. R., Balotin, A. F., & Marques, J. L. B. (2018). High-density surface EMG system based on ADS1298 front-end. IEEE Latin America Transactions, 16(6), 1616–1622. https://doi.org/10.1109/TLA.2018.8444157

Ferreira, E., Sebastião, P., Cercas, F., Sá Costa, C., & Correia, A. (2023). An optimized planning tool for microwave terrestrial and satellite link design. Future Internet, 15(2), 58. https://doi.org/10.3390/fi15020058

Fort, C. M., Ciupe, A. M., & Vlad, S. (2017). An ECG front-end device based on ADS1298 converter. En IFMBE Proceedings (Vol. 59, pp. 99–102). https://doi.org/10.1007/978-3-319-52875-5_22

Fortune, B. C., Pretty, C. G., Chatfield, L. T., McKenzie, L. R., & Hayes, M. P. (2019). Low-cost active electromyography. HardwareX, 6, e00085. https://doi.org/10.1016/j.ohx.2019.e00085

Fu, J., Huang, S., Cao, J., Huang, J., Xu, D., Jiang, N., … Zhang, W. (2024). Towards robust EMG decoding for prosthetic control. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 32, 2460–2469. https://doi.org/10.1109/TNSRE.2024.3422489

Gámez, M. J. (2022, 24 mayo). Objetivos y metas de desarrollo sostenible. Naciones Unidas. https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

Guo, S., Liu, Y., Zhang, Y., Zhang, S., & Yamamoto, K. (2016). A VR-based self-rehabilitation system for upper limb. En 2016 IEEE International Conference on Mechatronics and Automation (ICMA) (pp. 1173–1178). https://doi.org/10.1109/ICMA.2016.7558728

Huamani, R. R., Talavera, J. R., Mendoza, E. A. S., Dávila, N. M., & Supo, E. (2017). Implementation of a real-time 60 Hz interference cancellation algorithm for ECG signals based on ARM Cortex-M4 and ADS1298. En 2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON) (pp. 1–4). https://doi.org/10.1109/INTERCON.2017.8079725

Huf, A., & Siqueira, F. (2019). Composition of heterogeneous services in IoT healthcare platforms. Journal of Network and Computer Applications, 143, 89–110. https://doi.org/10.1016/j.jnca.2019.06.008

Institución Universitaria Pascual Bravo. (2022, diciembre). Plan de desarrollo Tradición – Transformación – Innovación 2023–2026. Recuperado el 27 de abril de 2024, de https://pascualbravo.edu.co

Instituto Nacional de Salud. (2020). Boletín epidemiológico semanal. Semana epidemiológica 15. Instituto Nacional de Salud, Colombia.

Jaleel, A., Mahmood, T., Hassan, M. A., Bano, G., & Khurshid, H. (2020). AI-powered clinical decision support systems: A review. IEEE Access, 8, 132302–132319. https://doi.org/10.1109/ACCESS.2020.3009783

Jara, C., Moënne-Loccoz, C., & Peña, M. (2022). Electroencefalografía y seguimiento de mirada: Dos técnicas para evaluar la adquisición de la lengua materna en niños pequeños. Revista Signos, 55(110), 902–927. https://doi.org/10.4067/S0718-09342022000300902

Javed, A., Malhi, A., Kinnunen, T., & Främling, K. (2020). An industrial IoT-based framework for real-time monitoring and anomaly detection. IEEE Access, 8, 211973–211985. https://doi.org/10.1109/ACCESS.2020.3039368

Jurado, C. A. (2022). Rediseño de un sistema de adquisición embebido de señal sEMG con electrodos flexibles y acondicionamiento digital. Trabajo de grado, Universidad de Antioquia.

Kabra, R., Israni, S., Vijay, B., Baru, C., Mendu, R., Felix, J., & Kohli, P. (2022). AI-enabled cardiovascular diagnostic services. Cardiovascular Diagnosis and Therapy Journal, 3(6), 263–275. https://doi.org/10.1016/j.cvdhj.2022.09.001

Karthick, R., Ramkumar, R., Akram, M., & Kumar, M. V. (2020). Smart health monitoring using IoT. Materials Today: Proceedings, 45, 1614–1619. https://doi.org/10.1016/j.matpr.2020.08.420

Kularatna, N. (2008). Electronic circuit design: From concept to implementation. CRC Press.

Kumar, N., Kumar, A., Gupta, P., Srivastava, R., Tewari, R. P., Sahai, N., & Kumar, B. (2021). Development of cloud-based multi-modal m-cardiac management system. En Lecture Notes in Electrical Engineering (pp. 429–437). https://doi.org/10.1007/978-981-15-6840-4_34

Lopera Echavarría, J. D., Ramírez Gómez, C. A., & Zuluaga Aristizábal, M. U. (2010). El método analítico como método natural. Nómadas. Revista Crítica de Ciencias Sociales y Jurídicas, 25, 331–347.

Ma, S., Lv, B., Lin, C., Sheng, X., & Zhu, X. (2021). EMG signal filtering based on variational mode decomposition and sub-band thresholding. IEEE Journal of Biomedical and Health Informatics, 25(1), 47–58. https://doi.org/10.1109/JBHI.2020.2987528

Margolis, M. (2020). Arduino cookbook: Recipes to begin, expand, and enhance your projects (3rd ed.). O'Reilly Media.

Martin, R. C. (2009). Clean code: A handbook of agile software craftsmanship. Pearson.

Merkoureas, I., Kaouni, A., Theodoropoulou, G., Bousdekis, A., Voulodimos, A., & Miaoulis, G. (2023). Smyrida: A web application for process mining and interactive visualization. SoftwareX, 22, 101327. https://doi.org/10.1016/j.softx.2023.101327

Merletti, R., & Parker, P. A. (2004). Electromyography: Physiology, engineering, and non-invasive applications. Wiley-IEEE Press.

Mican, C. A. (2020). Desarrollo de un sistema de parametrización de la conductividad nerviosa de músculos de la zona lumbar ante una flexión o extensión, basado en electromiografía no invasiva. Tesis de maestría.

Ministerio de Ciencia, Tecnología e Innovación. (2023, 13 de diciembre). Políticas de investigación e innovación orientadas por misiones – Misión soberanía sanitaria y bienestar social. https://minciencias.gov.co

Mitolo, M., & Araneo, R. (2019). A brief history of electrical safety in medical technology. IEEE Industry Applications Magazine, 25(2), 7–11. https://doi.org/10.1109/MIAS.2018.2884753

Molloy, D. (2016). Exploring Raspberry Pi: Interfacing to the real world with embedded Linux. John Wiley & Sons.

Morrison, J. (2000). What's the DLL, man? En Programming Windows with C# (pp. 169–182). Apress. https://doi.org/10.1007/978-1-4302-0858-7_9

Nagel, C. (2021). Professional C# and .NET (8th ed.). Wrox.

Nacitarhan, O. O., & Semiz, B. (2022). PySio: A new Python framework for biomedical signal I/O. En 2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC) (pp. 3686–3689). https://doi.org/10.1109/EMBC48229.2022.9871174

Ni, S., Al-Qaness, M. A., Hawbani, A., Al-Alimi, D., Elaziz, M. A., & Kim, T. (2024). EMG-driven pattern recognition for prosthetic applications using deep learning. Applied Soft Computing, 166, 112235. https://doi.org/10.1016/j.asoc.2024.112235

Nowak, B., & Jędrzejewski, K. (2023). Application of Bluetooth Low Energy 5 technology in wireless 12-lead ECG signal transmission. En 2023 Signal Processing Symposium (SPSympo) (pp. 128–132). https://doi.org/10.23919/SPSympo57300.2023.10302694

Oppenheim, A. V., Willsky, A. S., & Nawab, S. H. (1998). Señales y sistemas (2.ª ed.). Pearson Educación.

Orts Blázquez, J. (2019). Diseño de un sistema de adquisición de señales por electromiografía superficial para la interpretación del movimiento de la mano. Trabajo de grado, Universitat Politècnica de València. http://hdl.handle.net/10251/126202

Osherove, R. (2013). The art of unit testing: With examples in C# (2nd ed.). Manning Publications.

Pancholi, S., & Joshi, A. M. (2019). Electromyography-based hand gesture recognition system for upper limb amputees. IEEE Sensors Letters, 3(3), 1–4. https://doi.org/10.1109/LSENS.2019.2898257

Pancholi, S., & Joshi, A. M. (2020a). A fast and accurate deep learning framework for EMG-PR based upper-limb prosthesis control. En 2020 IEEE International Symposium on Smart Electronic Systems (iSES) (pp. 206–207). https://doi.org/10.1109/iSES50453.2020.00053

Pancholi, S., & Joshi, A. M. (2020b). Improved classification scheme using fused wavelet packet transform–based features for intelligent myoelectric prostheses. IEEE Transactions on Industrial Electronics, 67(10), 8517–8525. https://doi.org/10.1109/TIE.2019.2946536

Pancholi, S., Joshi, A. M., & Joshi, D. (2022). DLPR: Deep learning-based enhanced pattern recognition framework for improved myoelectric prosthesis control. IEEE Transactions on Medical Robotics and Bionics, 4(4), 991–999. https://doi.org/10.1109/TMRB.2022.3216957

Park, K. S. (2023). Humans and electricity: Understanding body electricity and applications. Springer Nature.

Pasqualin, B. R., Silva, V. A., Orselli, M. I. V., & Manffra, E. F. (2024). Computational modeling of the pendulum test to simulate spasticity in the elbow joint. En IX Latin American Congress on Biomedical Engineering and XXVIII Brazilian Congress on

Biomedical Engineering (CLAIB–CBEB 2022) (IFMBE Proceedings, Vol. 100). Springer. https://doi.org/10.1007/978-3-031-49407-9_7

Pereira, F., França, D., Paschoal, V., Nardes, M., Rosa, R. R., & Guerra, E. (2023). Esfinge Virtual Lab: A virtual laboratory platform with a metadata-based API and dynamic components. IEEE Access, 11, 143167–143181. https://doi.org/10.1109/ACCESS.2023.3342911

Piccolino, M. (1998). Animal electricity and the birth of electrophysiology. Brain Research Bulletin, 46(5), 381–407. https://doi.org/10.1016/S0361-9230(98)00026-4

Plonsey, R., & Barr, R. C. (2007). Bioelectricity: A quantitative approach (3rd ed.). Springer.

Proakis, J. G., & Manolakis, D. G. (1992). Digital signal processing: Principles, algorithms, and applications (3rd ed.). Macmillan.

Prongnuch, S., & Wiangtong, T. (2016). Heterogeneous computing architecture for real-time biomedical signal processing. En 2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS) (pp. 1–4). https://doi.org/10.1109/ISPACS.2016.7824762

Ramesh, C., & Meyer-Rochow, V. B. (2021). Bioluminescence for biomedical applications: A review. Aquatic Ecology, 55(3), 737–764. https://doi.org/10.1007/s10452-021-09875-0

Rani, G. J., Hashmi, M. F., & Gupta, A. (2023). Surface EMG signal–based prosthetic hand control using machine learning. IEEE Access, 11, 105140–105169. https://doi.org/10.1109/ACCESS.2023.3316509

Ran, Z., Zou, C., Wei, Z., & Wang, H. (2023). VELAS: An open-source toolbox for visualization and analysis of elastic anisotropy. Computer Physics Communications, 283, 108540. https://doi.org/10.1016/j.cpc.2022.108540

Raval, J. A., Sakinala, V. V., Jadhav, N. R., & Karia, D. (2017). ECG signal compression using wavelet transform. En 2017 International Conference on Communication and Signal Processing (ICCSP) (pp. 2153–2156). https://doi.org/10.1109/ICCSP.2017.8286788

Sadiku, M. N. O. (2003). Elementos de electromagnetismo. Oxford University Press México.

Schmitt, R. (2002). Electromagnetics explained: A handbook for wireless/RF, EMC, and high-speed electronics. Newnes.

Schouten, M., van de Maat, P., Nizamis, K., & Krijnen, G. (2022). Evaluating 3D-printed sEMG electrodes with silver ink traces using in-situ impedance measurements. En 2022 IEEE Sensors (pp. 1–4). https://doi.org/10.1109/SENSORS52175.2022.9967266

Seroglou, F., Koumaras, P., & Tselfes, V. (1998). History of science and scientific culture: Biographies in science education. Science & Education, 7, 261–280. https://doi.org/10.1023/A:1008649319416

Simao, M., Mendes, N., Gibaru, O., & Neto, P. (2019). A review on electromyography decoding and pattern recognition for human–machine interaction. IEEE Access, 7, 39564–39582. https://doi.org/10.1109/ACCESS.2019.2906584

Singh, A. K., & Krishnan, S. (2023). ECG signal feature extraction and classification using deep learning. BioMedical Engineering Online, 22(1). https://doi.org/10.1186/s12938-023-01075-1

Soro, T., Belaid, S., Hattay, J., & Thabet, H. (2023). Deep cardiovascular clinical decision support and control system. En 2023 IEEE International Conference on Artificial Intelligence & Green Energy (ICAIGE) (pp. 1–5). https://doi.org/10.1109/ICAIGE58321.2023.10346453

Subrahmanya, S. V. G., Shetty, D. K., Patil, V., Hameed, B. M. Z., & Shetty, N. (2021). Clinical validation of AI-based ECG interpretation. Irish Journal of Medical Science, 191(4), 1473–1483. https://doi.org/10.1007/s11845-021-02730-z

Supo, E., Huamani, R., Galdos, J., Rendulich, J., & Sulla, E. (2022). PRD as an indicator proposal in the evaluation of ECG signal acquisition prototypes in real patients. En 2022 IEEE ANDESCON (pp. 1–4). https://doi.org/10.1109/ANDESCON56260.2022.9989674

Texas Instruments. (2015, agosto). ADS129x low-power, 8-channel, 24-bit analog front-end for biopotential measurements (Hoja de datos). https://www.ti.com/lit/ds/symlink/ads1298.pdf

Texas Instruments. (2016). ADS1298ECGFE-PDK ADS1298 performance demonstration kit [Hoja de datos y guía de usuario]. Texas Instruments.

Texas Instruments. (2021). ADS1298R low-power, 8-channel, 24-bit analog front-end (Hoja de datos). Texas Instruments.

Texas Instruments. (2024a). ADS1298RECGFE-PDK ADS1298R performance demonstration kit [Hoja de datos y guía de usuario]. Texas Instruments.

Texas Instruments. (2024b). Instrumentation amplifiers. Recuperado el 27 de abril de 2024, de https://www.ti.com/amplifier-circuit/instrumentation/overview.html

Toro Ossaba, A., Jaramillo Tigreros, J. J., & Tejada Orjuela, J. C. (2020). Open source multichannel EMG armband design. En IX International Congress of Mechatronics Engineering and Automation (CIIMA) (pp. 1–6). https://doi.org/10.1109/CIIMA50553.2020.9290291

Toro-Ossaba, A., Jaramillo-Tigreros, J., Tejada, J. C., Peña, A., López-González, A., & Castanho, R. A. (2022). LSTM recurrent neural network for hand gesture recognition using EMG signals. Applied Sciences, 12(19), 9700. https://doi.org/10.3390/app12199700

Uktveris, T., & Jušas, V. (2018). Development of a modular board for EEG signal acquisition. Sensors, 18(7), 2140. https://doi.org/10.3390/s18072140

Vidaurre, C., Sander, T. H., & Schlögl, A. (2011). BioSig: The free and open source software library for biomedical signal processing. Computational Intelligence and Neuroscience, 2011, 1–12. https://doi.org/10.1155/2011/935364

Wang, J., Wang, L., Xi, X., Miran, S. M., & Xue, A. (2020). Estimation and correlation analysis of lower limb joint angles based on surface electromyography. Electronics, 9(4), 556. https://doi.org/10.3390/electronics9040556

Wang, Y., et al. (2022). A deep-learning-based method for gaining more biomechanical parameters with fewer sensors in fast and complex movements. En 2022 IEEE International Conference on Real-time Computing and Robotics (RCAR) (pp. 704–709). https://doi.org/10.1109/RCAR54675.2022.9872240

Weiss, L. D., Weiss, J. M., & Silver, J. K. (2022). Easy EMG: A guide to performing nerve conduction studies and electromyography (3rd ed.). Elsevier.

Weyer, S., Menden, T., Leicht, L., Leonhardt, S., & Wartzek, T. (2015). Development of a wearable multi-frequency impedance cardiography device. Journal of Medical Engineering & Technology, 39(2), 131–137. https://doi.org/10.3109/03091902.2014.990161

Wu, R., Scully-Allison, C., Carthen, C., Garcia, A., Hoang, R., Lewis, C., … Harris, F. C. (2023). vFirelib: A GPU-based fire simulation and visualization tool. SoftwareX, 23, 101411. https://doi.org/10.1016/j.softx.2023.101411

Yosifovich, P., Ionescu, A., Russinovich, M., & Solomon, D. (2017). Windows internals, part 1: System architecture, processes, threads, memory management, and more (7th ed.). Microsoft Press.

Young, B., & Schmid, J. (2021). The new ISO/IEC standard for automated cardiac defibrillators. Hearts, 2(3), 410–419. https://doi.org/10.3390/hearts2030032

Zhang, Y., Zhang, X., Sun, H., Fan, Z., & Zhong, X. (2019). Portable brain–computer interface based on novel convolutional neural network. Computers in Biology and Medicine, 107, 248–256. https://doi.org/10.1016/j.compbiomed.2019.02.023

Zheng, R., et al. (2023). Open-source toolbox for visualization of elastic anisotropy. Computer Physics Communications, 283, 108540.

Zhu, M., Guan, X., Li, Z., et al. (2023). sEMG-based lower limb motion prediction using CNN–LSTM with improved PCA optimization algorithm. Journal of Bionic Engineering, 20, 612–627. https://doi.org/10.1007/s42235-022-00280-3

10. Bibliografía

Burbank, D. P., & Webster, J. G. (1978). Reducing skin potential motion artifact by skin abrasion. Medical and Biological Engineering and Computing, 16(1), 31–38. https://doi.org/10.1007/BF02442929

Chowdhury, R. H., Reaz, M. B. I., Ali, M. A. B. M., Bakar, A. A. A., Chellappan, K., & Chang, T. G. (2013). Surface electromyography signal processing and classification techniques. Sensors, 13(9), 12431–12466. https://doi.org/10.3390/s130912431

Clancy, E. A., Morin, E. L., & Merletti, R. (2002). Sampling, noise-reduction and amplitude estimation issues in surface electromyography. Journal of Electromyography and Kinesiology, 12(1), 1–16. https://doi.org/10.1016/S1050-6411(01)00033-5

De Luca, C. J. (1997). The use of surface electromyography in biomechanics. Journal of Applied Biomechanics, 13(2), 135–163. https://doi.org/10.1123/jab.13.2.135

Farina, D. (2006). Surface electromyography (EMG) signal processing. En M. Akay (Ed.), Wiley encyclopedia of biomedical engineering. John Wiley & Sons. https://doi.org/10.1002/9780471740360.ebs1378

Geng, W., Du, Y., Jin, W., Wei, W., Hu, Y., & Li, J. (2016). Gesture recognition by instantaneous surface EMG images. Scientific Reports, 6, 36571. https://doi.org/10.1038/srep36571

Gijsberts, A., Atzori, M., Castellini, C., Müller, H., & Caputo, B. (2014). Movement error rate for evaluation of machine learning methods for sEMG-based hand movement classification. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 22(4), 735–744. https://doi.org/10.1109/TNSRE.2014.2303394

Holobar, A., Minetto, M. A., & Farina, D. (2014). Accurate identification of motor unit discharge patterns from high-density surface EMG and validation with a novel signal-based performance metric. Journal of Neural Engineering, 11(1), 016008. https://doi.org/10.1088/1741-2560/11/1/016008

Kyranou, I., Vijayakumar, S., & Erden, M. S. (2018). Causes of performance degradation in non-invasive electromyographic pattern recognition in upper limb prostheses. Frontiers in Neurorobotics, 12, 58. https://doi.org/10.3389/fnbot.2018.00058

McGill, K. C. (2004). Surface electromyogram signal modelling. Medical and Biological Engineering and Computing, 42(4), 446–454. https://doi.org/10.1007/BF02350985

Merletti, R., & Cerone, G. L. (2020). Tutorial: Surface EMG detection, conditioning and pre-processing: Best practices. Journal of Electromyography and Kinesiology, 54, 102440. https://doi.org/10.1016/j.jelekin.2020.102440

Merletti, R., & Farina, D. (Eds.). (2016). Surface electromyography: Physiology, engineering, and applications. John Wiley & Sons. https://doi.org/10.1002/9781119082934

Navarro, X., Krueger, T. B., Lago, N., Micera, S., Stieglitz, T., & Dario, P. (2005). A critical review of interfaces with the peripheral nervous system for the control of neuroprostheses and hybrid bionic systems. Journal of the Peripheral Nervous System, 10(3), 229–258. https://doi.org/10.1111/j.1085-9489.2005.10303.x

Scheme, E., & Englehart, K. (2011). Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use. Journal of Rehabilitation Research and Development, 48(6), 643–659. https://doi.org/10.1682/JRRD.2010.09.0177

Subasi, A. (2013). Classification of EMG signals using PSO optimized SVM for diagnosis of neuromuscular disorders. Computers in Biology and Medicine, 43(5), 576–586. https://doi.org/10.1016/j.compbiomed.2013.01.020

Sul, J.-H., Piyathilaka, L., Moratuwage, D., Dunu Arachchige, S., Jayawardena, A., Kahandawa, G., & Preethichandra, D. M. G. (2025). Electromyography signal acquisition, filtering, and data analysis for exoskeleton development. Sensors, 25(13), 4004. https://doi.org/10.3390/s25134004

Tam, H. W., & Webster, J. G. (1977). Minimizing electrode motion artefact by skin abrasion. IEEE Transactions on Biomedical Engineering, 24(2), 134–139. https://doi.org/10.1109/TBME.1977.326117

Vigotsky, A. D., Halperin, I., Lehman, G. J., Trajano, G. S., & Vieira, T. M. (2018). Interpreting signal amplitudes in surface electromyography studies in sport and rehabilitation sciences. Frontiers in Physiology, 8, 985. https://doi.org/10.3389/fphys.2017.00985

Winter, D. A. (2009). Biomechanics and motor control of human movement (4th ed.). John Wiley & Sons. https://doi.org/10.1002/9780470549148

Zhai, X., Jelfs, B., Chan, R. H. M., & Tin, C. (2017). Self-recalibrating surface EMG pattern recognition for neuroprosthesis control based on convolutional neural network. Frontiers in Neuroscience, 11, 379. https://doi.org/10.3389/fnins.2017.00379

11. Anexos

Anexo A. Pseudocódigo simplificado de la API

```
INICIO
  BAUD := 115200
  AutoConnect():
    para cada COM visible:
      abrir puerto a BAUD
      escribir "PING"            // texto
      si lee "OK": devolver instancia abierta
    si ninguno responde: error

  SafeInit():
    Spi(0x11)                  // SDATAC
    Spi(0x06)                  // RESET
    Spi(0x02)                  // WAKEUP
    Spi(0x11)                  // SDATAC

  ConfigureOdrNearest(250):
    WREG(0x01, 0x06)           // CONFIG1: 250 sps

  UseInternalTestSignal(freq=0x01, amp=0x00):
    WREG(0x02, 0x10 | (amp<<2) | freq)  // CONFIG2: INT_TEST=1, fCLK/2^20

  SetConfig3(0xC0):
    WREG(0x03, 0xC0)           // referencias internas ON

  para ch=1..8:
    ConfigureChannel(ch, MUX=TEST(0b101), GAIN=0b001):
      WREG(CHnSET, (GAIN<<4) | 0b101)

  RdAtac():  Spi(0x10)
  Start():   Spi(0x08)

  StartStream8(onFrame):
    escribir "STREAM8ON"       // texto
    lanzar lector binario:
      bucle:
        sincronizar 0xA5 0x5A
        LEN := leer byte (esperar 0x1B)
        payload := leer 27 bytes (STATUS[3] + 8×CH[3])
        chk := leer byte
        si (sum(payload)&0xFF) != chk: continuar
        descartar STATUS[3]
        para c=1..8:
```

```
        v := sign_extend_24(payload[3+3*(c-1) : +3])
      onFrame( int[8] := {v1..v8} )

  StopStream8():
    escribir "STREAM8OFF"
    detener lector

RUN_INTERNAL_TEST_8 (llamado por Python)
  SafeInit()
  ConfigureOdrNearest(250)
  UseInternalTestSignal(0x01, 0x00)
  SetConfig3(0xC0)
  configurar 8 canales en MUX=TEST, GAIN=0b001
  RdAtac(); Start(); StartStream8(onFrame)
```

Anexo B. Pseudocódigo simplificado del firmware

```
CONFIG FIJA
  UART: 115200 8N1 en GPIO44 (RX0), GPIO43 (TX0)
  SPI : MODE1 @ 1 MHz
  Pines ADS1298R: CS=10, SCLK=12, MOSI=11, MISO=13, DRDY=18, START=20,
RESET=21

ARRANQUE
  iniciar UART(115200), SPI(MODE1,1MHz)
  CS=HIGH, START=LOW, RESET=HIGH
  hard reset: RESET=LOW 5 ms → HIGH

COMANDOS (texto)
  "PING"        → responder "OK"
  "SPI <hex>"   → transferir y responder "RX: <hex>"
  "DIAG"        → esperar DRDY=LOW, leer 27 bytes y mostrarlos en texto
  "SCOPEON n"   → flag_scope=true; canal=n
  "SCOPEOFF"    → flag_scope=false
  "STREAM8ON"   → flag_stream8=true
  "STREAM8OFF"  → flag_stream8=false

LOOP
  si hay comando UART → atender
  si flag_scope y DRDY=LOW:
    leer 27 bytes (STATUS[3]+CH1..CH8)
    decodificar CH[n] (24b con signo) → imprimir CH[canal] como entero ASCII
  si flag_stream8 y DRDY=LOW:
    leer 27 bytes (STATUS[3]+CH1..CH8)
    enviar binario:
      0xA5 0x5A 0x1B   // cabecera y longitud fija
      payload(27B)
      checksum := sum(payload)&0xFF
      enviar checksum
```

Anexo C. Pseudocódigo simplificado del script de alto nivel

```
CONFIG
  DLL := "BioadqRt.dll"
  SPS := 250
  GAIN_CODE := 0b001      // PGA=2
  VREF := 2.42 V
  UNITS := µV
  ventana := 5.0 s
INICIO
  cargar pythonnet y referenciar DLL
  dev := ADS1298.AutoConnect()
  dev.SdAtac(); id := dev.Rreg(0x00,1)
ESCALADO
  PGA := 2
  LSB_volts := VREF / (PGA * 2^23)
  scale := LSB_volts * 1e6     // µV por cuenta
BUFFERS
  crear deques para CH1..CH8 y tiempo (longitud ≈ SPS*ventana)
CALLBACK onFrame(int[8])
  t := tiempo relativo
  para c=1..8:
    volts_c := int[c] * scale   // µV
    append a buffer[c]
  append t a buffer tiempo
```

```
ARRANQUE
  dev.RunInternalTest8(
    sps=250, gainCode=0b001,
    onFrame=callback,
    testFreqBits=0x01, testAmpBit=0x00, config3=0xC0
  )
  // DLL: SafeInit→ODR 250→INT_TEST(f=2^20, amp x1)→CONFIG3 0xC0→8ch TEST→
RDATAC→START→STREAM8ON
PLOTEO
  crear ventana con 8 subgráficas (4×2)
  cada ~33 ms:
    actualizar datos de cada canal (µV) en rango [t-5s, t]
    autoescalar eje Y con margen
SALIDA
  al cerrar:
    dev.StopStream8()
    dev.Stop()
    dev.Dispose()
```