GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

APLICACIÓN PARA COMUNICACIÓN INALÁMBRICA CON SENSOR DE ELECTROMIOGRAFÍA

Autor: Juan Diego Díaz Hidalgo

Director: José Daniel Muñoz Frías

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Aplicación para comunicación inalámbrica con sensor de electromiografía

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2024/25 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Juan Diego Díaz Hidalgo Fecha: 29 / 06 / 2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Firmado por MUÑOZ FRIAS JOSE DANIEL - ***7384** el día 30/06/2025 con un certificado emitido por AC FNMT Usuarios

Fdo.: José Daniel Muñoz Frías Fecha: ……/ ……/ ……

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

APLICACIÓN PARA COMUNICACIÓN INALÁMBRICA CON SENSOR DE ELECTROMIOGRAFÍA

Autor: Juan Diego Díaz Hidalgo

Director: José Daniel Muñoz Frías

Madrid

A mis padres, Hima y Juan Pedro,
por confiar en mí cuando el esfuerzo no
daba resultados, y por forjar la persona
en la que me he convertido.

A Miguel y a Juan, porque este
camino nunca estuvo pensado
para recorrerse en solitario.

APLICACIÓN PARA COMUNICACIÓN INALÁMBRICA CON SENSOR DE ELECTROMIOGRAFÍA

Autor: Díaz Hidalgo, Juan Diego.
Director: Muñoz Frías, José Daniel.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Este proyecto presenta el desarrollo de una plataforma portátil para la adquisición y visualización en tiempo real de señales electromiográficas (EMG). Se implementó un sistema basado en un microcontrolador ESP32-C3 con comunicación Bluetooth Low Energy (BLE) y una aplicación de escritorio en Python para la recepción, visualización y almacenamiento de datos. La solución facilita el análisis dinámico de señales musculares, proporcionando una herramienta accesible y flexible para aplicaciones clínicas y de investigación.

Palabras clave: Electromiografía, Bluetooth Low Energy, Visualización en tiempo real, Python

1. Introducción

La electromiografía es una técnica fundamental en la evaluación de la actividad muscular, con aplicaciones clínicas y de investigación. Sin embargo, los sistemas tradicionales suelen requerir equipamiento voluminoso y costoso, limitando su accesibilidad. Este trabajo presenta el desarrollo de una solución integrada que combina hardware personalizado y software interactivo para facilitar el diagnóstico y análisis muscular en entornos menos especializados.

2. Definición del proyecto

El proyecto consiste en el diseño y construcción de un sistema portátil para la adquisición, transmisión y visualización de señales EMG. El objetivo principal es crear una herramienta de bajo coste, fácil de usar y con capacidades de análisis en tiempo real. Para ello, se integra un microcontrolador ESP32-C3 con capacidades BLE y una aplicación de escritorio desarrollada en Python que gestiona la conexión, visualización y almacenamiento de datos. La arquitectura general del sistema se ilustra en la Figura 1

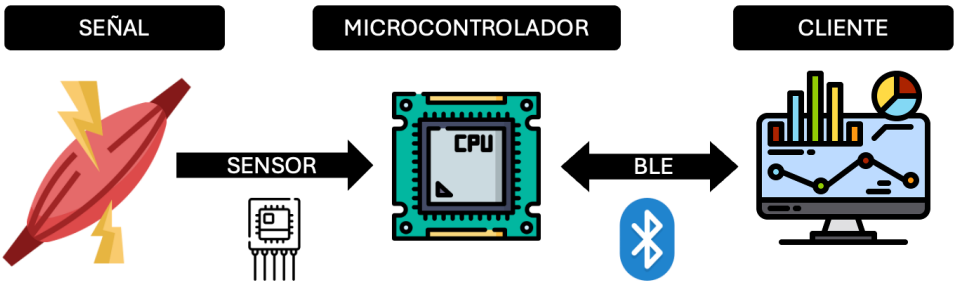

Figura 1 Diagrama de arquitectura general [1], [2], [3], [4], [5]

3. Descripción del modelo/sistema/herramienta

El sistema consta de dos componentes principales:

- Hardware: Está basado en un ESP32-C3, un microcontrolador que incluye conectividad BLE, ideal para enviar datos de forma inalámbrica con bajo consumo. El ESP32-C3 está conectado a un sensor de EMG de superficie, el cual capta la actividad eléctrica generada por los músculos. Estas señales se adquieren a través del ADC interno del microcontrolador, se procesan de forma básica (por ejemplo, cambio de canal o filtrado si es necesario), y luego se transmiten a la aplicación de escritorio por BLE. La ausencia de cables permite una mayor comodidad y libertad de movimiento para el usuario.

- Software: Es una aplicación de escritorio programada en Python. Utiliza PySide6 para construir una interfaz gráfica moderna y funcional, Bleak para gestionar la conexión BLE con el ESP32-C3, y pyqtgraph para mostrar en tiempo real los datos de los diferentes canales EMG. Además, permite guardar las sesiones en archivos Csv y .m para que puedan ser analizadas después. Desde la aplicación se puede controlar la conexión con el dispositivo, visualizar en directo la actividad muscular y registrar los datos de forma sencilla.

4. Resultados

La aplicación ha sido probada en un entorno controlado, logrando una transmisión de datos estable a través de BLE con latencias mínimas. La aplicación permite visualizar las señales EMG en tiempo real con un intervalo de refresco personalizable, y el almacenamiento de datos en formato CSV y .m facilita su análisis con herramientas externas. La Figura 2 presenta una captura de la visualización de señales durante una sesión típica.

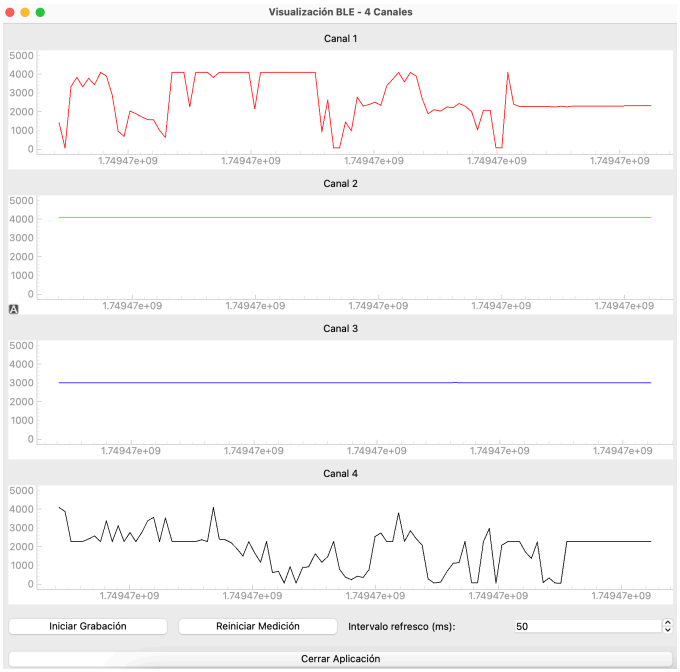

Figura 2 Interfaz de la aplicación de visualización

5. Conclusiones

El sistema desarrollado ha logrado cumplir con los objetivos principales del proyecto, proporcionando una plataforma portátil capaz de adquirir y visualizar señales EMG en tiempo real con gran estabilidad. La integración entre el microcontrolador ESP32-C3 y la aplicación de escritorio basada en Python se ha realizado de forma modular y escalable, permitiendo futuras extensiones. El diseño se apoya en estándares abiertos como BLE y CSV, lo que garantiza la compatibilidad con otras herramientas y entornos. Además, se ha validado el correcto funcionamiento en pruebas de uso continuo, lo que demuestra su aplicabilidad en contextos reales.

6. Referencias

[1] Flaticon. Accedido el 9 de junio de 2025. [En línea]. Disponible: https://www.flaticon.com/free-icons/sensor

[2] Flaticon. Accedido el 9 de junio de 2025. [En línea]. Disponible: https://www.flaticon.com/free-icons/bluetooth.

[3] Flaticon. Accedido el 9 de junio de 2025. [En línea]. Disponible: https://www.flaticon.com/free-icons/muscle

[4] Flaticon. Accedido el 9 de junio de 2025. [En línea]. Disponible: https://www.flaticon.com/free-icons/cpu

[5] Flaticon. Accedido el 9 de junio de 2025. [En línea]. Disponible: https://www.flaticon.com/free-icons/big-data

WIRELESS COMMUNICATION APPLICATION FOR ELECTROMYOGRAPHY SENSOR

Author: Díaz Hidalgo, Juan Diego.
Supervisor: Muñoz Frías, José Daniel.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

This project presents the development of a portable platform for real-time acquisition and visualization of electromyographic (EMG) signals. A system was implemented based on an ESP32-C3 microcontroller with Bluetooth Low Energy (BLE) communication and a desktop application developed in Python for data reception, visualization, and storage. The solution facilitates dynamic analysis of muscle signals, providing an accessible and flexible tool for both clinical and research applications.

Keywords: Electromyography, Bluetooth Low Energy, Real-time Visualization, Python

1. Introduction

EMG is a fundamental technique for evaluating muscle activity, with applications in both clinical and research environments. However, traditional systems often require bulky and expensive equipment, which limits accessibility. This work presents the development of an integrated solution that combines custom hardware and interactive software to support muscle diagnosis and analysis in less specialized settings.

2. Project definition

The project consists of designing and building a portable system for acquiring, transmitting, and visualizing EMG signals. The main goal is to create a low-cost, user-friendly tool with real-time analysis capabilities. To achieve this, an ESP32-C3 microcontroller with BLE capabilities is integrated with a Python-based desktop application that manages data connection, visualization, and storage. The overall system architecture is illustrated in Figure 1.

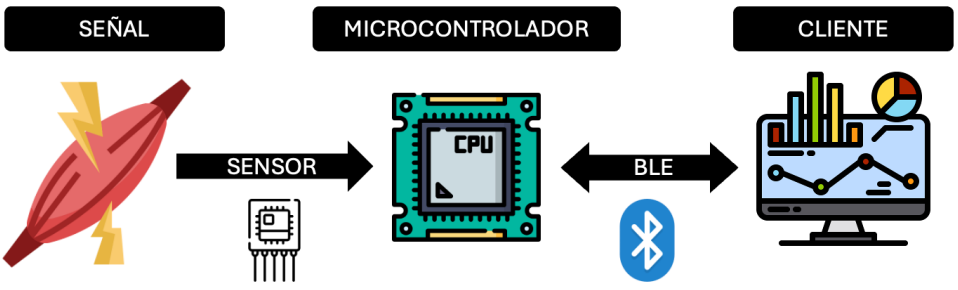

Figure 1 General system architecture diagram [1], [2], [3], [4], [5]

3. System description

The system is composed of two main components:

- Hardware: Based on an ESP32-C3 microcontroller, which includes BLE connectivity, making it ideal for wireless data transmission with low power consumption. The ESP32-C3 is connected to a surface EMG sensor that captures the electrical activity generated by muscles. These signals are acquired through the microcontroller's internal ADC, processed at a basic level (e.g., channel switching or filtering if necessary), and transmitted to the desktop application via BLE. The wireless design provides greater comfort and freedom of movement for the user.
- Software: A desktop application developed in Python. It uses PySide6 to build a modern and functional graphical interface, Bleak to manage the BLE connection with the ESP32-C3, and pyqtgraph for real-time visualization of EMG signals from multiple channels. It also allows saving sessions in CSV and .m files for later analysis. From the application, users can control the connection with the device, monitor muscle activity live, and store data easily.

4. Results

The application has been tested in a controlled environment, achieving stable data transmission over BLE with minimal latency. The application enables real-time visualization of EMG signals with a customizable refresh rate, and data storage in CSV and .m formats simplifies analysis using external tools. Figure 2 shows a screenshot of the signal visualization interface during a typical session.

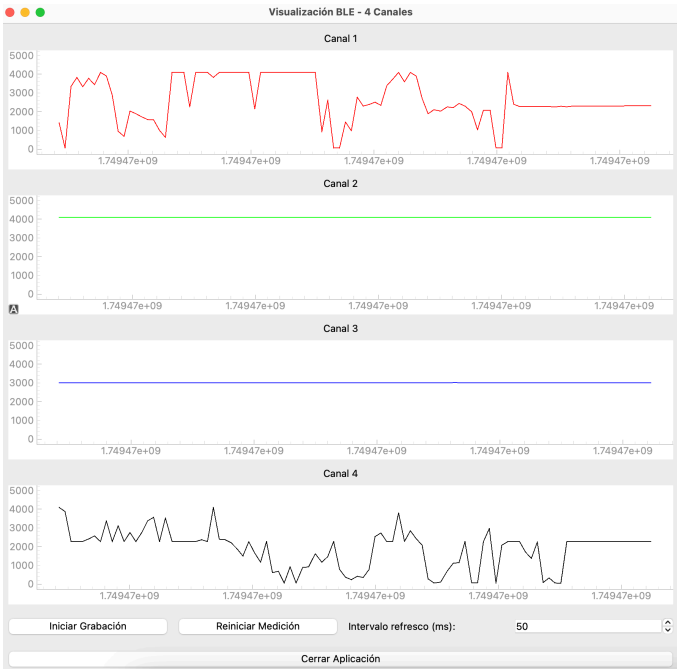

Figure 2 Application interface for signal visualization

5. Conclusions

The developed system has successfully met the main objectives of the project, providing a portable platform capable of acquiring and visualizing EMG signals in real time with high stability. The integration between the ESP32-C3 microcontroller and the Python-based desktop application was carried out in a modular and scalable way, enabling future extensions. The design relies on open standards such as BLE and CSV, ensuring compatibility with other tools and environments. Furthermore, the system was validated under continuous use tests, demonstrating its applicability in real-world contexts.

6. References

[1] Flaticon. Accessed on June 9, 2025. [Online]. Available: https://www.flaticon.com/free-icons/sensor

[2] Flaticon. Accessed on June 9, 2025. [Online]. Available: https://www.flaticon.com/free-icons/bluetooth

[3] Flaticon. Accessed on June 9, 2025. [Online]. Available: https://www.flaticon.com/free-icons/muscle

[4] Flaticon. Accessed on June 9, 2025. [Online]. Available: https://www.flaticon.com/free-icons/cpu

[5] Flaticon. Accessed on June 9, 2025. [Online]. Available: https://www.flaticon.com/free-icons/big-data

Índice de la memoria

Capítulo 1. Introducción ***5***

Capítulo 2. Descripción de las Tecnologías ***9***

2.1 Hardware 9

2.2 Programación del microcontrolador 10

2.2.1 FreeRTOS *10*

2.2.2 Manejo de periféricos: GPIO y SPI *11*

2.2.3 Lectura analógica continua (ADC) *11*

2.2.4 Stack BLE (NimBLE) *11*

2.3 Comunicación inalámbrica 12

2.4 Aplicación de escritorio 12

2.4.1 PySide6 ..... *13*

2.4.2 PyQTGraph *13*

2.4.3 Bleak *13*

2.4.4 Csv *13*

Capítulo 3. Estado de la Cuestión ***15***

3.1 ConsensysPRO Software 15

3.2 iMotions Lab 18

3.3 Cadwell Sierra Software 19

3.4 Alternativas Open Source 21

Capítulo 4. Definición del Trabajo ***23***

4.1 Justificación 23

4.1.1 Limitaciones técnicas de las herramientas disponibles *24*

4.1.2 Ventajas del enfoque propuesto *25*

4.1.3 Aplicaciones futuras potenciales *25*

4.2 Objetivos 26

4.3 Metodología 26

4.4 Planificación 27

Capítulo 5. Sistema/Modelo Desarrollado ***29***

5.1 Programación ESP32-C3 29
5.1.1 Adquisición y Acondicionamiento de la Señal *30*
5.1.2 Comunicación inalámbrica mediante BLE *31*
5.2 Aplicación de escritorio 36
5.2.1 Arquitectura General *36*
5.2.2 Interfaz gráfica de usuario *37*
5.2.3 Comunicación BLE *37*
5.2.4 Adquisición y procesamiento de datos *42*
5.2.5 Exportación de datos *44*
5.2.6 Grabación y reinicio de medición. *46*

Capítulo 6. Análisis de Resultados 49
6.1 Funcionamiento general del Sistema 49
6.2 Tasa de actualización gráfica 50
6.3 Exportación y carga de sesiones 52
6.4 Análisis crítico: Fortalezas y Limitaciones 55
6.4.1 Fortalezas técnicas y operativas *56*
6.4.2 Limitaciones identificadas y aspectos mejorables *56*

Capítulo 7. Conclusiones y Trabajos Futuros 59

Capítulo 8. Bibliografía 61

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS 65
8.1 ODS 3: Salud y bienestar [25] 65
8.2 ODS 4: Educación de calidad [26] 65
8.3 ODS 9: Industria, innovación e infraestructura [27] 66
8.4 ODS 12: Producción y consumo responsables [28] 66

Índice de figuras

Figura 1 Diagrama de arquitectura general [1], [2], [3], [4], [5] 8
Figura 2 Interfaz de la aplicación de visualización 9
Figura 3 Diagrama de una unidad motora [1] 5
Figura 4 Gráfica del potencial de acción de la unidad motora [2] 6
Figura 5 Logo de Espressif [3] 9
Figura 6 Logo de Shimmer [6] 16
Figura 7: Dispositivo Shimmer3 EMG Unit [7] 16
Figura 8 Interfaz de ConsensysPRO [8] 17
Figura 9 Logo de iMotions [9] 18
Figura 10 Interfaz de iMotionsLab [10] 19
Figura 11 Logo de Cadwell Industries, Inc [12] 20
Figura 12: Equipo Sierra Summit EMG/NCS/EP/NMUS [13] 20
Figura 13 Interfaz de Cadwell Sierra Software [14] 21
Figura 14 Sesión funcional 50
Figura 15 Configuración intervalo de refresco 51
Figura 16: Ventana para exportar .m. 52
Figura 17: Configuración del archivo .m. 53
Figura 18: Validación del guardado. 53
Figura 19: Señal analizada en MATLAB 54
Figura 20: Menú desplegable con opciones de sesión. 54
Figura 21: Ventana para selección de la sesión 55

Índice de tablas

Tabla 1 Planificación del proyecto 27
Tabla 2: UUID de los canales 33
Tabla 3 Flujo general de la aplicación 37
Tabla 4: Objetivos y resultados. 60

Capítulo 1. INTRODUCCIÓN

La electromiografía (EMG) es el estudio de la actividad eléctrica de los músculos del esqueleto [1]. Proporciona información muy útil sobre su estado fisiológico y el de los nervios que los activan. Las señales de electromiografía registran la actividad eléctrica generada por las unidades motoras, que son la unidad funcional del sistema neuromuscular. Cada unidad motora está compuesta por una neurona motora y todas las fibras musculares que inerva. Cuando el sistema nervioso central envía una señal eléctrica, esta viaja por la neurona motora hasta las fibras musculares, generando una despolarización que provoca su contracción.

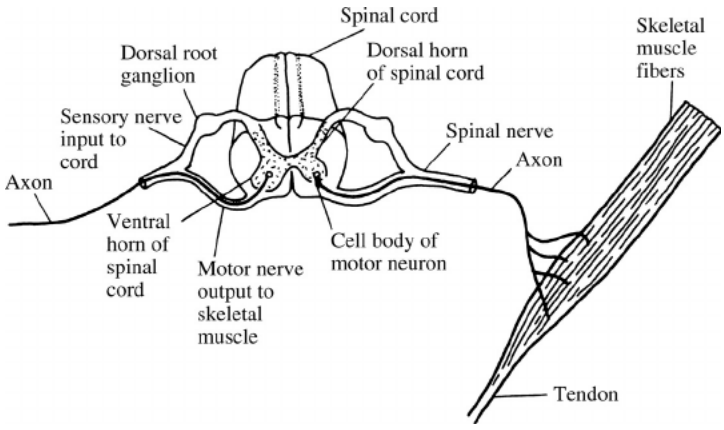

Figura 3 Diagrama de una unidad motora [1]

Este proceso bioeléctrico genera pequeños potenciales eléctricos que pueden ser detectados mediante electrodos colocados sobre la piel, permitiendo así la medición de la actividad muscular en tiempo real.

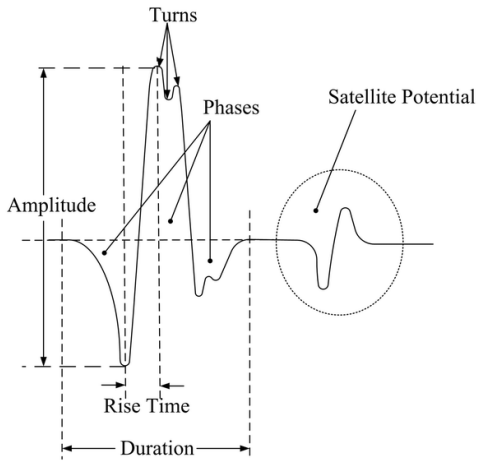

Figura 4 Gráfica del potencial de acción de la unidad motora [2]

Durante los últimos años, las tecnologías médicas no invasivas han experimentado un avance significativo, impulsadas por la necesidad de mejorar el diagnóstico, el monitoreo y la rehabilitación de los pacientes. En este contexto, los sensores de electromiografía se han consolidado como herramientas de gran utilidad para analizar la actividad muscular con alta precisión, ofreciendo una alternativa eficaz frente a técnicas que implican acceso físico al interior del cuerpo. Esta evolución ha sido posible gracias al desarrollo de sensores portátiles y a la incorporación de sistemas de comunicación inalámbrica, que han permitido ampliar el uso de la EMG más allá de los entornos clínicos tradicionales.

Uno de los avances más relevantes en este ámbito ha sido la implementación de la tecnología Bluetooth Low Energy (BLE), que facilita una comunicación eficiente y de bajo consumo entre los sensores y los dispositivos encargados de la visualización y el análisis de los datos. Esta integración tecnológica ha abierto la puerta a nuevas aplicaciones prácticas de la EMG, más accesibles, cómodas y adaptadas a distintos entornos.

En este contexto se enmarca el presente proyecto, que tiene como objetivo el desarrollo de una aplicación de escritorio para la lectura y visualización de registros provenientes de un

sensor de EMG desarrollado por la Escuela en colaboración con el Hospital Universitario de Getafe. La conexión entre el sensor y la aplicación se ha establecido mediante tecnología Bluetooth, lo que permite una experiencia de uso más flexible y libre de cables.

La herramienta permite establecer una conexión con el sensor mediante BLE y proporciona una interfaz amigable e intuitiva para su uso. Entre sus principales funcionalidades destacan el control remoto del dispositivo —incluyendo su encendido y apagado—, la selección del modo de funcionamiento, la visualización del estado de la batería y la representación gráfica en tiempo real de hasta cuatro canales de señal muscular. Además, la herramienta está equipada con funciones avanzadas como la activación o desactivación independiente de cada canal, la grabación de sesiones completas para su análisis posterior y la exportación de datos en formato compatible con MATLAB, permitiendo así una evaluación más profunda mediante herramientas científicas especializadas.

Estas soluciones, que combinan hardware personalizado con un software de visualización interactiva en tiempo real, representan un avance significativo hacia sistemas de diagnóstico más accesibles, eficientes y adaptables a diferentes entornos. El desarrollo de esta plataforma surge de la necesidad de contar con herramientas que ofrezcan tanto precisión técnica como facilidad de uso, sin depender de infraestructuras complejas. En este sentido, uno de los principales motores del proyecto ha sido la voluntad de democratizar el acceso a este tipo de tecnologías, tradicionalmente reservadas a entornos clínicos altamente especializados. Al ofrecer una alternativa portátil, intuitiva y de bajo coste, se abre la puerta a su integración en ámbitos como la investigación académica, la docencia universitaria o incluso aplicaciones domésticas avanzadas.

En definitiva, este proyecto busca demostrar cómo la ingeniería puede dar respuesta a retos médicos reales, integrando tecnologías emergentes en plataformas accesibles que amplían sus posibilidades de uso más allá del laboratorio o el hospital.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este capítulo, se presentarán los diferentes recursos empleados durante el desarrollo del trabajo. El proyecto se puede dividir en cuatro partes, donde cada una se enfoca en una funcionalidad específica del sistema:

- Hardware
- Programación del microcontrolador
- Comunicación inalámbrica
- Aplicación de escritorio

2.1 HARDWARE

El primer paso para llevar a cabo esta iniciativa fue seleccionar un microcontrolador capaz de procesar y enviar las muestras recogidas por el sensor de electromiografía. Para ello, se optó por el ESP32-C3, un chip de bajo consumo fabricado y comercializado por la empresa china de semiconductores Espressif. Este microcontrolador destaca por su conectividad inalámbrica integrada y su capacidad para ejecutar tareas en tiempo real, lo que lo convierte en una opción idónea para aplicaciones portátiles en el ámbito biomédico.

Figura 5 Logo de Espressif [3]

Adicionalmente, el dispositivo cuenta con varios canales ADC, lo que permite la lectura simultánea de múltiples señales. También dispone de un módulo Bluetooth Low Energy

(BLE) integrado, que facilita la comunicación con dispositivos externos. La arquitectura del núcleo del microcontrolador está basada en RISC-V, que junto con el soporte para entornos de desarrollo ESP-ID (Espressif IoT Development Framework), simplifica enormemente la implementación lógica de adquisición de datos y comunicación inalámbrica.

2.2 PROGRAMACIÓN DEL MICROCONTROLADOR

La configuración del ESP32-C3 se ha realizado en lenguaje C, utilizando el entorno de desarrollo ESP-IDF, que permite un control detallado de los periféricos del microcontrolador. Esta opción no solo ofrece una mayor optimización y eficiencia en la gestión de recursos, sino que también permite una programación más robusta frente a soluciones como Arduino, cuya capa de abstracción, aunque sencilla y accesible, limita el control sobre el hardware.

El lenguaje C es ampliamente utilizado en sistemas embebidos debido a su eficiencia, velocidad de ejecución y cercanía al hardware. A diferencia de entornos como Arduino, que están más orientados a principiantes y prototipado rápido, trabajar directamente con C y ESP-IDF permite explotar al máximo las capacidades del ESP32-C3, tanto en rendimiento como en versatilidad.

Durante el desarrollo, se emplearon múltiples bibliotecas específicas incluidas en ESP-IDF, las cuales veremos a continuación.

2.2.1 FREERTOS

La programación multitarea del ESP32-C3 se ha basado en el uso del sistema operativo en tiempo real FreeRTOS, incluido por defecto en ESP-IDF. A través de los includes freertos/FreeRTOS.h y freertos/task.h, se gestionan múltiples tareas concurrentes como la lectura del ADC, la actualización del estado del dispositivo o la transmisión por BLE. Esta arquitectura permite una ejecución eficiente y sincronizada, fundamental en sistemas biomédicos donde la latencia y la estabilidad son críticas.

2.2.2 MANEJO DE PERIFÉRICOS: GPIO Y SPI

El control de periféricos externos se ha realizado mediante los includes driver/gpio.h y driver/spi_master.h.

- GPIO: permite la configuración y control de los pines digitales del microcontrolador, esenciales para interactuar con componentes como el multiplexor o señales de control del sensor.
- SPI Master: habilita la comunicación síncrona con periféricos que requieren un intercambio de información más compleja y a más velocidad.

2.2.3 LECTURA ANALÓGICA CONTINUA (ADC)

Para capturar las señales del sensor EMG, se ha empleado el módulo ADC en modo continuo utilizando la librería esp_adc/adc_continuous.h. Esta configuración permite adquirir datos sin interrupciones, minimizando la pérdida de muestras y asegurando una señal estable. La elección del modo continuo frente al modo por interrupciones convencionales responde a la necesidad de adquirir información de forma rápida y precisa en tiempo real, con un periodo de muestreo constante.

2.2.4 STACK BLE (NIMBLE)

Para la comunicación inalámbrica, se utilizó el *stack* NimBLE, una alternativa ligera y eficiente al *stack* BLE clásico. Los principales ficheros de cabecera incluidos fueron:

- host/ble_hs.h: base del núcleo de NimBLE.
- services/gap/ble_svc_gap.h: permite la gestión de la publicidad, visibilidad y emparejamiento del dispositivo.
- services/gatt/ble_svc_gatt.h: se encarga de definir y manejar los servicios y características GATT, que permiten la transmisión de datos al dispositivo cliente (como el ordenador).

NimBLE ofrece mejor rendimiento y menor uso de memoria, siendo especialmente adecuado para microcontroladores con recursos limitados.

2.3 COMUNICACIÓN INALÁMBRICA

Bluetooth Low Energy (BLE) es un protocolo de comunicación inalámbrico pensado para transmitir datos en pequeñas cantidades de forma eficiente, consumiendo muy poca energía. A diferencia del Bluetooth clásico, BLE está diseñado para mantener conexiones rápidas y estables sin necesidad de mantener un enlace constante, lo que lo hace ideal para dispositivos como sensores portátiles alimentados con baterías , donde el consumo ha de ser pequeño.

En este proyecto, se ha utilizado BLE para establecer la conexión entre el sensor de electromiografía y la aplicación de escritorio. Esta tecnología permite enviar en tiempo real las señales musculares sin necesidad de cables, manteniendo el consumo energético bajo y facilitando la portabilidad del sistema.

El microcontrolador utilizado, un ESP32-C3, cuenta con un módulo BLE integrado que permite gestionar todas las funciones necesarias para la comunicación inalámbrica sin necesidad de hardware adicional. Esto ha hecho posible definir varias características BLE, cada una asociada a un canal del sensor, permitiendo enviar las señales de forma estructurada y sincronizada.

Gracias a esta integración, se ha logrado una solución compacta, eficiente y adecuada para la transmisión continua de señales fisiológicas, manteniendo un buen equilibrio entre rendimiento, consumo y simplicidad.

2.4 APLICACIÓN DE ESCRITORIO

Para el desarrollo de la aplicación de escritorio se eligió Python como lenguaje principal. La decisión se basa en su simplicidad, legibilidad y la enorme cantidad de librerías disponibles, lo que permite desarrollar prototipos funcionales en menos tiempo que con otros lenguajes como C++ o Java.

Además, Python ofrece una gran flexibilidad a la hora de integrar distintos componentes del sistema, desde interfaces gráficas hasta comunicaciones BLE, pasando por el procesamiento

y guardado de datos, gracias a su amplia gama de bibliotecas especializadas. A continuación, se detallan las principales librerías utilizadas y el papel que desempeñan dentro del proyecto.

2.4.1 PySide6

Se ha utilizado para crear la interfaz gráfica de la aplicación. Es una de las implementaciones oficiales de las librerías Qt para Python, y permite diseñar ventanas interactivas con menús, botones, formularios y cuadros de diálogo. Gracias a su integración con el resto del ecosistema de Python, ha sido posible construir una interfaz intuitiva y funcional sin sacrificar rendimiento ni estética.

2.4.2 PyQTGraph

Esta librería se ha empleado para la representación de las señales en tiempo real. Está optimizada para mostrar gráficos dinámicos con alta frecuencia de actualización [4], lo que la convierte en una herramienta ideal para representar las señales musculares. Su integración con PySide6 es fluida, permitiendo combinar visualización y controles de usuario en una sola ventana.

2.4.3 Bleak

Bleak ha sido la biblioteca utilizada para gestionar la comunicación con el microcontrolador a través de BLE. Al ser multiplataforma y funcionar de forma asíncrona, permite mantener una conexión estable sin bloquear el resto de la aplicación. Gracias a esta herramienta, se ha podido establecer el enlace con el dispositivo, recibir los datos de los distintos canales en tiempo real y controlar algunas de sus funciones de forma remota.

2.4.4 Csv

Esta librería se ha implementado para almacenar las sesiones grabadas en archivos con formato estructurado y legible, lo que facilita su análisis posterior con herramientas como Excel o MATLAB. Complementariamente, os y QFileDialog han permitido integrar funciones para explorar el sistema de archivos, brindando al usuario una forma intuitiva de seleccionar rutas para guardar o cargar las muestras recogidas durante la sesión.

Capítulo 3. ESTADO DE LA CUESTIÓN

La EMG comenzó a desarrollarse a principios del siglo XX como una herramienta para estudiar la actividad eléctrica de los músculos. Uno de los hitos clave tuvo lugar en 1922, cuando los fisiólogos Joseph Erlanger y Herbert Gasser utilizaron un osciloscopio para registrar por primera vez señales eléctricas provenientes del tejido muscular [5]. Este avance marcó un antes y un después en la comprensión de los procesos neuromusculares, sentando las bases de una disciplina que ha evolucionado de forma constante desde entonces.

Con el tiempo, el perfeccionamiento de los equipos de medición permitió realizar estudios más precisos, seguros y accesibles. La electromiografía se ha consolidado como una técnica fundamental en el ámbito clínico, especialmente en el diagnóstico de trastornos neuromusculares, la evaluación funcional del sistema motor y el desarrollo de prótesis o dispositivos de asistencia.

Durante las últimas décadas, los avances tecnológicos han permitido reducir el tamaño de los sensores y mejorar significativamente la precisión con la que se registran las señales musculares.

Actualmente, existen múltiples aplicaciones dedicadas a la lectura, análisis y visualización de las señales recogidas por sensores de EMG. Entre los softwares más destacados se encuentran los siguientes:

3.1 CONSENSYSPRO SOFTWARE

Este software está diseñado para trabajar con los sensores comercializados por la empresa Shimmer, misma empresa que desarrolla el programa en este caso.

Figura 6 Logo de Shimmer [6]

Uno de los dispositivos más representativos de este ecosistema es el Shimmer3 EMG Unit, una unidad portátil específicamente diseñada para la adquisición de señales electromiográficas. Este sensor cuenta con entradas analógicas de alta precisión, transmisión inalámbrica vía Bluetooth y almacenamiento local mediante tarjeta SD. Su funcionamiento está totalmente integrado con la interfaz de ConsensysPRO, lo que permite configurar los parámetros de adquisición, visualizar los datos en tiempo real y exportarlos para su análisis.

Figura 7: Dispositivo Shimmer3 EMG Unit [7]

El dispositivo está orientado a usos clínicos y de investigación avanzada, especialmente en áreas como biomecánica, ergonomía o neurorehabilitación. Su arquitectura modular también permite el acoplamiento de otros sensores fisiológicos, como ECG o acelerómetros, ampliando su campo de aplicación.

Las funcionalidades principales del software son:

- Configuración de los sensores Shimmer.

- Reproducción de información a tiempo real a través de tecnología Bluetooth.
- Gestión de la recolección de los datos. [8]

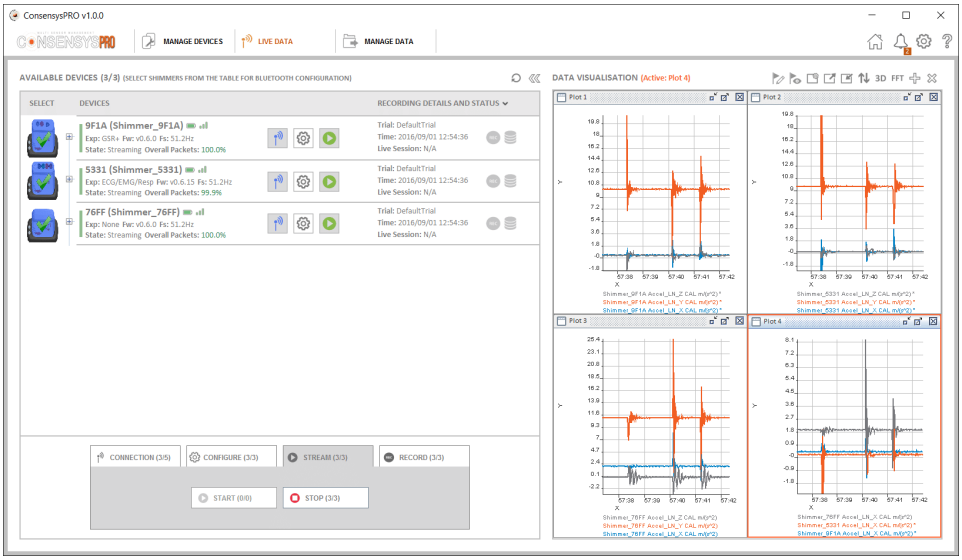

Figura 8 Interfaz de ConsensysPRO [8]

Cuenta con dos versiones para Windows (PRO y BASIC) y el precio de la licencia de uso se encuentra en el rango de 199€ - 1.791€ [8], variando según el número de dispositivos válidos para la licencia y la duración de la suscripción.

La versión ConsensysPRO cuenta con una prueba gratuita de 5 días, y después requerirá adquirir una licencia para continuar con su uso. Además, cuenta con diferentes ventajas frente a la versión básica:

- Grabación y visualización de datos desde varios dispositivos Shimmer.
- Configuración de varios dispositivos de manera simultánea.
- Soporte técnico para el software.
- Algoritmos y funcionalidades extra para el análisis de mediciones.

La versión ConsensysBASIC, sin embargo, no requiere licencia de uso, por lo que se podría disponer de esta versión de manera gratuita para acceder a las funciones más básicas del programa.

3.2 iMOTIONS LAB

Es una plataforma de software modular cuyo funcionamiento parte del módulo fundamental denominado CORE. Este componente básico sirve como pilar para la configuración y administración de laboratorios orientados a la investigación, ofreciendo herramientas clave para el diseño experimental, la recopilación de datos y su análisis posterior.

Figura 9 Logo de iMotions [9]

Entre los módulos adicionales que pueden integrarse a iMotions Lab se encuentran sistemas de medición fisiológica como electroencefalografía (EEG), electrocardiografía (ECG) y EMG.

Este último módulo presenta una serie de características destacadas, entre las que se incluyen [9]:

- Herramientas de análisis automatizadas, que adaptan los datos recibidos para una visualización más clara e intuitiva.
- Indicadores del estado del dispositivo, como la batería o la fuerza de la señal.

- Compatibilidad con la conexión simultánea de hasta seis dispositivos mediante una única licencia.
- Compatible con dispositivos como Shimmer3 EMG, BIOPAC100 EMG y Plux Electromyography (EMG) Sensor.

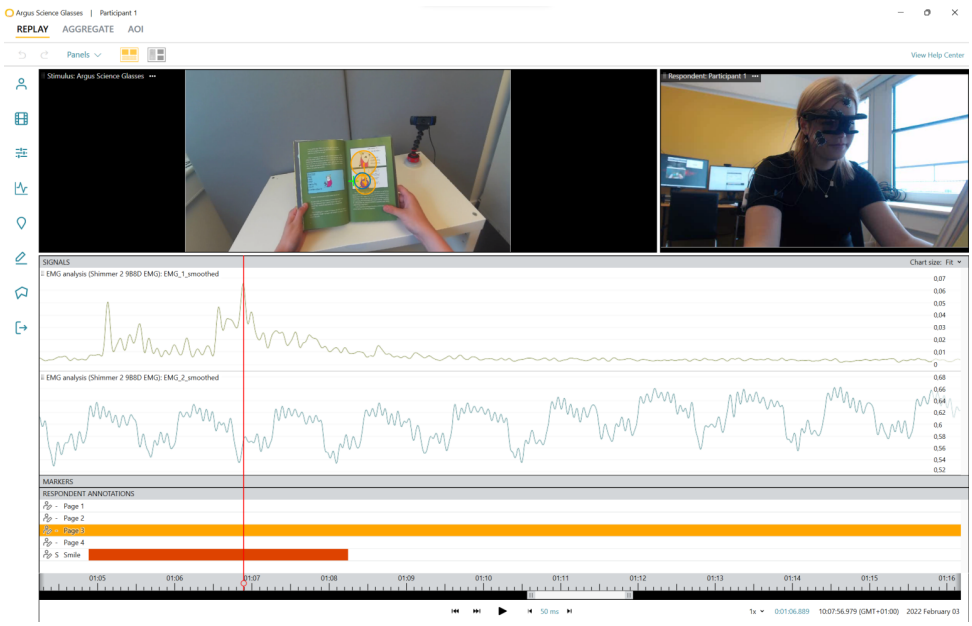

Figura 10 Interfaz de iMotionsLab [10]

Dependiendo de la finalidad de uso, el software ofrece licencias de uso académicas y comerciales, ambas con un precio anual por módulo. El precio de la licencia académica asciende a 3.400€, mientras que la comercial se sitúa en 3.000€. [11]

En el caso académico, la licencia permite extender el uso a estudiantes mediante licencias específicas para análisis de datos, siempre que estos hayan sido recogidos con una instalación completa de iMotions Lab

3.3 CADWELL SIERRA SOFTWARE

Es una de las herramientas más potentes del mercado. Ofrece una amplia gama de protocolos para el tratamiento de medidas del sensor EMG, así como una gran capacidad de personalización de interfaz. Proporciona un gran control del display de

datos y soporta el flujo de estos en tiempo real. Los datos en tiempo real se almacenan para su posterior editado o manipulación.

Figura 11 Logo de Cadwell Industries, Inc [12]

Este software está diseñado principalmente para su integración en la propia infraestructura de la línea Sierra, como por ejemplo el Sierra Summit EMG/NCS/EP/NMUS, que consiste en un sistema completo con amplificadores, estimuladores y software.

Figura 12: Equipo Sierra Summit EMG/NCS/EP/NMUS [13]

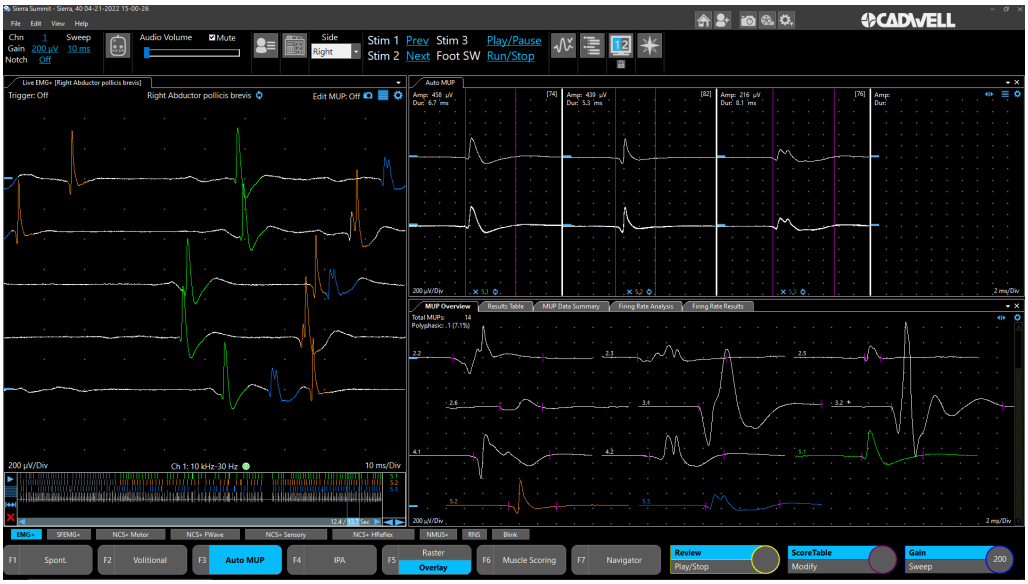

Figura 13 Interfaz de Cadwell Sierra Software [14]

3.4 ALTERNATIVAS OPEN SOURCE

Uno de los pocos desarrollos de código abierto orientados a la adquisición de señales EMG es OpenEMG, disponible públicamente en la plataforma GitHub [15]. El repositorio proporciona una interfaz de visualización y un firmware funcional, junto con una guía de montaje para construir el sistema desde cero. Su enfoque está basado en un diseño cerrado, utilizando una cadena de adquisición específica desarrollada por el propio autor.

A pesar de su valor como iniciativa libre, OpenEMG no emplea hardware estándar ni modular, sino que se apoya en una configuración personalizada que incluye componentes y conexiones propias. Esto implica que, para integrarlo en otros sistemas o reutilizar su firmware y visualizador, sería necesario modificar parte del proyecto o replicar físicamente su arquitectura. Además, no implementa protocolos modernos como Bluetooth Low Energy (BLE), lo que reduce su compatibilidad con dispositivos actuales o interfaces móviles.

En consecuencia, aunque es una referencia válida dentro del panorama open source, OpenEMG no resulta una alternativa óptima para proyectos que requieran portabilidad, flexibilidad o integración con arquitecturas preexistentes, como la planteada en este trabajo.

Capítulo 4. DEFINICIÓN DEL TRABAJO

En los últimos años, las señales biológicas, y en concreto las de EMG, han ganado peso en campos como la rehabilitación, el deporte o la robótica. Sin embargo, trabajar con este tipo de señales en tiempo real sigue sin ser tan accesible como debería. Las soluciones comerciales son caras y cerradas. Las opciones open source, por otro lado, suelen estar mal documentadas, ser poco intuitivas o demasiado genéricas.

Esto deja un hueco importante: no hay muchas herramientas que permitan a estudiantes, investigadores o desarrolladores visualizar señales EMG de forma sencilla, en tiempo real y desde su propio equipo, sin depender de la nube ni de configuraciones complicadas.

El objetivo de este proyecto es cubrir ese hueco con una aplicación de escritorio que se conecta por Bluetooth (BLE) a un ESP32-C3, el cual recibe datos de un sensor EMG. La app muestra en tiempo real la señal, permite guardar sesiones, cargarlas después y analizar lo registrado, todo con una interfaz clara y moderna. Se busca que sea funcional, ligera y fácil de modificar, para que cualquiera con conocimientos básicos pueda adaptarla a su propio caso.

4.1 JUSTIFICACIÓN

Los sensores EMG, actualmente, destacan por su accesibilidad en comparación con equipos clínicos tradicionales, lo que ha facilitado su incorporación en proyectos educativos y de investigación. Sin embargo, en cuanto tratamos de trabajar con los datos, surgen múltiples desafíos: ¿cómo visualizarlos?, ¿cómo analizar las señales? En muchos casos, las herramientas disponibles resultan complejas, costosas o, simplemente, no están pensadas para quienes se inician o desean realizar un prototipo funcional de forma rápida.

Este proyecto cubre precisamente esa necesidad. Se plantea una solución sencilla y directa: una aplicación de escritorio que permita visualizar en tiempo real las señales recogidas por un sensor EMG, sin complicaciones ni barreras técnicas. Todo el sistema está construido a partir de hardware de bajo coste, con un ESP32-C3 que transmite las medidas por Bluetooth, y software libre, para que cualquier persona interesada pueda utilizarlo, modificarlo o adaptarlo a sus propias necesidades.

Además, el proyecto no se limita a mostrar datos de forma superficial. La intención es que esta herramienta pueda servir como base para desarrollos posteriores: desde el análisis de patrones musculares hasta la implementación de sistemas de control mediante movimientos del brazo. En definitiva, se trata de acercar la tecnología a quienes desean aprender, experimentar o construir soluciones propias, sin depender de plataformas cerradas ni herramientas excesivamente técnicas.

4.1.1 LIMITACIONES TÉCNICAS DE LAS HERRAMIENTAS DISPONIBLES

La mayoría de las herramientas diseñadas para trabajar con estos sensores están enfocadas a contextos profesionales, siendo necesaria una infraestructura compleja y unos conocimientos técnicos elevados. Estas soluciones, aunque potentes, se alejan de las necesidades del proyecto, precisamente por estas barreras que imponen.

Si observamos las alternativas de software libre, encontramos herramientas poco documentadas, desarrolladas a medida de hardware concreto, o que precisan de modificaciones sustanciales para adaptarse a nuestro dispositivo.

Por último, es fundamental destacar la necesidad de disponer de herramientas accesibles no solo en términos económicos, sino también en cuanto a independencia tecnológica, evitando la dependencia de plataformas cerradas o servicios en la nube que pueden limitar el control sobre los datos y el desarrollo del proyecto.

4.1.2 VENTAJAS DEL ENFOQUE PROPUESTO

Por lo mencionado anteriormente, el sistema que se ha desarrollado tenía como misión cubrir las carencias que presentaban las alternativas del mercado. Teniendo en cuenta las limitaciones observadas, se planteó desde el inicio el desarrollo de un software que eliminase esas barreras sin renunciar a la funcionalidad.

El resultado es una herramienta que no requiere equipamiento especializado ni conocimientos avanzados para su uso, que evita cualquier inversión económica desorbitada y que, aun así, ofrece capacidades más que suficientes para la lectura y análisis de señales EMG en tiempo real. Esta solución ofrece, en definitiva, acercar el trabajo con señales biológicas a perfiles que tradicionalmente quedaban fuera de este tipo de tecnologías, como podría ser en este caso el proyecto de la Universidad.

4.1.3 APLICACIONES FUTURAS POTENCIALES

Aunque esta herramienta ha sido concebida como una solución básica para la visualización de señales EMG en tiempo real, su diseño modular y su carácter abierto permiten imaginar múltiples líneas de desarrollo a partir de la versión actual.

Entre las aplicaciones más inmediatas se encuentra su uso en el ámbito académico, tanto en el proyecto de la Escuela con el Hospital Universitario de Getafe, como su uso como recurso en asignaturas de la especialidad de Ingeniería Biomédica del grado. También puede servir como base para proyectos de investigación que requieran adquirir y analizar datos musculares sin recurrir a plataformas comerciales.

A medio plazo, la herramienta podría ampliarse con funciones más avanzadas, como el procesamiento automático de patrones musculares, la detección de gestos o la integración con sistemas de control por movimiento. Asimismo, su compatibilidad con hardware de bajo coste abre la puerta al desarrollo de dispositivos portátiles con capacidades personalizadas, adaptadas a distintas necesidades clínicas, deportivas o de rehabilitación.

En definitiva, se trata de una base sólida sobre la que construir soluciones más específicas, que contribuyan a democratizar el acceso a tecnologías de análisis muscular y a fomentar la innovación desde entornos donde tradicionalmente no se dispone de los recursos necesarios.

4.2 OBJETIVOS

Esta iniciativa nace con el objetivo de complementar el desarrollo del sensor de electromiografía llevado a cabo por ICAI en colaboración el Hospital Universitario de Getafe. La principal consigna del proyecto es proporcionar una herramienta funcional que permita visualizar, de forma inalámbrica y en tiempo real, los múltiples canales del sensor. Esta aplicación servirá como soporte durante todo el desarrollo del dispositivo, facilitando tanto la verificación de su funcionamiento como la captura y exportación de datos para su posterior análisis.

4.3 METODOLOGÍA

El proyecto se ha desarrollado en diferentes etapas a lo largo del curso lectivo, cada una de ellas orientada a cubrir las necesidades descritas en los apartados anteriores.

En primer lugar, se llevó a cabo la programación inicial del ESP32-C3. Esta fase se organizó en pequeños *sprints*, cada uno centrado en implementar y validar funcionalidades concretas. Se comenzaron desarrollando programas sencillos de prueba para comprobar el funcionamiento del dispositivo y familiarizarse con su arquitectura. A continuación, se abordó el desarrollo del código encargado de establecer la comunicación entre el microcontrolador y el ordenador utilizando BLE. En esta primera versión, el sistema enviaba números aleatorios que simulaban las medidas de un único canal ADC.

Con esta base funcional, se inició el desarrollo de la aplicación de escritorio, diseñada inicialmente para recibir y visualizar los datos enviados por BLE mediante una interfaz simple.

Una vez establecida la comunicación y la visualización básica, se avanzó hacia la integración de los resultados reales de los canales del ADC, ampliando el sistema para incorporar las mediciones de los cuatro canales disponibles en el dispositivo.

La fase final del proyecto consistió en dotar a la aplicación de las funcionalidades necesarias para convertirla en una herramienta efectiva para la lectura y análisis de las señales biológicas. Se implementaron opciones para el control individual de canales, la grabación de sesiones, la exportación de las muestras obtenidas y la posibilidad de reiniciar las mediciones, completando así el ciclo de desarrollo de la herramienta.

4.4 PLANIFICACIÓN

A continuación, se muestra la planificación que se ha seguido a lo largo del proyecto:

TAREAS	NOVIEMBRE				DICIEMBRE				ENERO				FEBRERO				MARZO				ABRIL				MAYO			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Investigación y planificación																												
Investigación previa																												
Establecer funcionalidades y requisitos																												
Diseño																												
Diseño UX/UI																												
Pruebas de diseño																												
Definición de estructura																												
Backend																												
Programación ESP32 - C3																												
Conexión inalámbrica																												
Frontend																												
Desarrollo de aplicación de escritorio																												
Cierre del proyecto																												
Integración Back - Front																												
Pruebas de funcionalidad																												

Tabla 1 Planificación del proyecto

Como se indica en la Tabla 1, el desarrollo del proyecto se ha estructurado en cinco fases fundamentales:

1. **Investigación y planificación:** En esta etapa inicial se definieron las directrices generales que guiarían el resto del desarrollo. Se investigaron las tecnologías disponibles y se establecieron la metodología de trabajo, los requisitos del sistema y las funcionalidades principales.
2. **Diseño:** Se abordó la definición de la arquitectura general del proyecto. Se esbozaron las estructuras principales tanto de la aplicación de escritorio como del sistema de

comunicación inalámbrica, dando lugar a una primera versión funcional de ambos componentes.

3. **Desarrollo del Backend:** Esta fase comprendió la programación y configuración del microcontrolador ESP32-C3, así como la implementación de una comunicación BLE estable con la primera versión de la interfaz de visualización.
4. **Desarrollo del Frontend:** Se continuó con la construcción de la aplicación de escritorio, integrando progresivamente todas las funcionalidades definidas durante la fase de planificación. Se puso especial énfasis en la experiencia de usuario y en la estabilidad de la visualización en tiempo real.
5. **Cierre del proyecto:** Esta última etapa consistió en la integración de todos los componentes del sistema, la realización de pruebas exhaustivas y la incorporación de mejoras iterativas hasta alcanzar una versión final funcional y robusta.

Capítulo 5. SISTEMA/MODELO DESARROLLADO

Este capítulo recoge con detalle el desarrollo completo del sistema, desde los primeros prototipos hasta la integración final entre el dispositivo de adquisición y la aplicación de escritorio. La solución se basa en un enfoque modular y funcional, donde el microcontrolador ESP32-C3 actúa como emisor inalámbrico de las señales EMG, y una aplicación desarrollada en Python permite su visualización, análisis y almacenamiento en tiempo real.

Para organizar adecuadamente la descripción técnica, este capítulo se divide en dos bloques principales. En primer lugar, se explica el diseño y programación del dispositivo, abordando cómo se realiza la lectura de señales mediante el ADC y la transmisión por BLE. A continuación, se detalla el desarrollo de la aplicación de escritorio, incluyendo su estructura, tecnologías utilizadas y las funcionalidades implementadas.

Esta estructura permite presentar con claridad tanto los retos técnicos encontrados como las soluciones adoptadas durante el proceso de desarrollo, con el objetivo de construir una herramienta útil, adaptable y accesible para distintos tipos de usuarios, desde estudiantes hasta investigadores o desarrolladores.

5.1 PROGRAMACIÓN ESP32-C3

El ESP32-C3 ha sido programado para actuar como interfaz entre el sensor de EMG y la aplicación de escritorio. Este microcontrolador se encarga tanto de adquirir los datos como de empaquetarlos y transmitirlos vía Bluetooth Low Energy (BLE). Todo el proceso se ha desarrollado en C, utilizando el entorno de ESP-IDF y el *stack* NimBLE para la comunicación inalámbrica.

5.1.1 ADQUISICIÓN Y ACONDICIONAMIENTO DE LA SEÑAL

Uno de los pilares del sistema es la captura precisa de señales EMG a través del microcontrolador ESP32-C3.

El ESP32-C3 cuenta con un subsistema de entrada analógica que incluye:

- Multiplexor interno para seleccionar entre múltiples canales de entrada.
- Conversor analógico-digital (ADC) con un controlador digital configurable, el cual permite definir tanto el patrón de conversión como la frecuencia de muestreo.

En este caso concreto, se emplea un ADC de 12 bits en modo continuo, operando bajo el subsistema ADC Continuous [16] del ESP-IDF. Esta modalidad permite la adquisición sin intervención directa del procesador para cada muestra, lo que reduce la carga de CPU y mejora la eficiencia energética.

La arquitectura del sistema de adquisición queda organizada de la siguiente forma:

1. Selección secuencial de canal mediante el PGA117: A pesar de que el ESP32-C3 puede seleccionar canales internos, en este diseño se utiliza un multiplexor externo (PGA117) que, en cada ciclo, conmuta entre cuatro entradas analógicas del sensor EMG. Este componente también permite fijar la ganancia para cada canal, lo cual es útil para adaptar la señal al rango óptimo del ADC.

2. Control digital del PGA117 vía SPI: La selección de canal y configuración de ganancia se realiza mediante el protocolo SPI, donde el microcontrolador envía comandos directamente al PGA117. Esto se realiza dentro de la rutina de interrupción del ADC, lo que permite una conmutación ágil e integrada con el flujo de adquisición.

3. Conversión y lectura de datos: Por cada cambio de canal, el ADC realiza dos conversiones consecutivas que posteriormente se promedian para obtener un valor más estable. Esta media simple ayuda a minimizar el ruido y las fluctuaciones propias de las señales biológicas.

4. Almacenamiento y sincronización: Una vez que los cuatro canales han sido leídos, los valores son almacenados en variables compartidas protegidas mediante mecanismos atómicos, lo cual garantiza la integridad de los datos frente a condiciones de carrera.

5. Notificación al sistema principal: Cuando se completa una ronda de muestreo de los cuatro canales, se genera una notificación mediante vTaskNotifyGiveFromISR, despertando el hilo principal encargado de procesar o transmitir los datos.

La frecuencia de muestreo final se fija en 1 kHz, lo que permite capturar con suficiente resolución la actividad muscular sin sobrecargar el sistema. Esta tasa se configura mediante el controlador digital del ADC y se adapta perfectamente al flujo de adquisición secuencial por canal.

En conjunto, este diseño aprovecha al máximo las capacidades internas del ESP32-C3 para realizar tareas que tradicionalmente habrían requerido circuitería adicional, integrando el acondicionamiento y la digitalización en una única placa, sin comprometer la calidad de la señal ni la flexibilidad del sistema.

5.1.2 COMUNICACIÓN INALÁMBRICA MEDIANTE BLE

La implementación concreta de la comunicación BLE se ha realizado mediante el *stack* NimBLE, integrado con FreeRTOS. Este permite definir servicios GATT personalizados y gestionar conexiones de forma eficiente en paralelo al resto del sistema. El ESP32-C3 actúa como servidor BLE, ofreciendo cuatro características de lectura independientes que representan las señales de los canales EMG. Esta configuración, junto a la gestión asíncrona mediante interrupciones, permite transmitir la información registrada con baja latencia y estabilidad en tiempo real.

5.1.2.1 Stack BLE NimBLE y su integración con FreeRTOS

Para gestionar la complejidad del protocolo BLE en un entorno embebido, se utiliza el *stack* NimBLE [17]. Esta implementación, desarrollada originalmente por Apache Mynewt [18] y

adaptada por Espressif, es una solución ligera y eficiente, diseñada para sistemas operativos en tiempo real como FreeRTOS, que es el sistema base del ESP32-C3.

NimBLE abstrae las capas de bajo nivel del protocolo BLE, permitiendo configurar:

- La publicidad del dispositivo, para anunciar su presencia y servicios disponibles.
- La gestión de conexiones GAP (Generic Access Profile).
- La definición y manejo de servicios y características GATT (Generic Attribute Profile).
- Los eventos y la interacción con clientes.

Gracias a su arquitectura modular, NimBLE corre en una tarea propia dentro de FreeRTOS (host_task), garantizando que la ejecución de las operaciones BLE se realice en paralelo y sin interferir con otras tareas críticas, como la adquisición de datos analógicos.

5.1.2.2 Publicidad y descubrimiento

El ESP32-C3 actúa como dispositivo periférico que emite periódicamente paquetes de publicidad. Estos contienen información clave para que un cliente (por ejemplo, la aplicación en el ordenador) pueda localizarlo y conectar:

- El nombre completo del dispositivo, configurado como "ESP32-ADC-PGA117".
- Información sobre los servicios que ofrece, a través de sus UUIDs.

El intervalo y la potencia de transmisión de estos anuncios se configuran para balancear la rapidez en la detección y el consumo energético, priorizando una conexión estable y rápida.

5.1.2.3 Servicios y características GATT personalizadas

Para facilitar la recepción estructurada de las muestras capturadas por el ESP32-C3 y permitir su representación en tiempo real desde la aplicación de escritorio, se ha diseñado un servicio GATT personalizado [18] con cuatro características de lectura independientes.

Cada una de estas características está asociada a uno de los canales del sensor de EMG, lo que permite una transmisión paralela y clara de la información.

El perfil GATT se define como un único servicio primario con UUID 0x1080, bajo el cual se han configurado cuatro características de tipo READ, con UUIDs comprendidos entre 0xFEF4 y 0xFEF7. Esta estructura sigue el siguiente esquema:

Servicio/Canal	**UUID**
Dirección del dispositivo	EEA0FD61-CCF5-067B-38EF-D10473D99B33
Canal 1	0000FEF4-0000-1000-8000-00805F9B34FB
Canal 2	0000FEF5-0000-1000-8000-00805F9B34FB
Canal 3	0000FEF6-0000-1000-8000-00805F9B34FB
Canal 4	0000FEF7-0000-1000-8000-00805F9B34FB

Tabla 2: UUID de los canales

Cada característica implementa una función de acceso (access_cb) que devuelve el último valor actualizado por el ADC en su canal correspondiente. Estos valores se almacenan en variables atómicas protegidas con semáforos (cuando procede), lo que permite garantizar la coherencia de lectura desde el hilo de BLE sin necesidad de bloquear la adquisición.

Internamente, las funciones device_read_chX() para cada canal actúan como interfaz entre el *stack* NimBLE y el valor numérico capturado por el conversor analógico-digital. Estas funciones empaquetan el valor como una cadena de caracteres (char buffer[20]) y lo anexan al buffer de salida de BLE (os_mbuf), lo que simplifica su decodificación en la aplicación cliente. Este diseño evita cualquier dependencia de codificación binaria, favoreciendo la interoperabilidad y la depuración durante el desarrollo.

Además, esta estructura tiene una ventaja importante: al emplear cuatro características independientes, la aplicación cliente puede leer cada canal de forma asíncrona y decidir su frecuencia de actualización de manera individual. Esto ofrece mayor flexibilidad a nivel de interfaz de usuario, optimización del consumo energético y tratamiento diferenciado de los canales, si fuese necesario aplicar filtrado, umbrales u otras operaciones específicas.

El diseño se mantiene deliberadamente simple y extensible. En caso de futuras ampliaciones, como añadir canales adicionales, comandos de control u opciones de configuración remota, sería posible agregar nuevas características o servicios secundarios sin romper la compatibilidad con las versiones actuales de la aplicación.

5.1.2.4 Sincronización y gestión de datos con FreeRTOS

El sistema utiliza FreeRTOS para organizar las tareas y sincronizar procesos críticos:

- La tarea principal de adquisición (en app_main) espera mediante ulTaskNotifyTake a que la interrupción del ADC le notifique que los cuatro canales han sido muestreados.
- Esta notificación permite procesar y actualizar los valores de las señales que luego se ofrecen a través de BLE.
- El *stack* NimBLE se ejecuta en paralelo en la tarea host_task, encargándose de gestionar las conexiones, solicitudes y eventos BLE.

Este esquema garantiza que la lectura de valores analógicos no se bloquee por operaciones BLE y viceversa, manteniendo un sistema fluido y estable.

5.1.2.5 Manejo de eventos BLE

El firmware implementa un *callback* para los eventos GAP, que incluye:

- Conexión: se registra el estado de conexión y se habilitan las funcionalidades necesarias para atender las solicitudes del cliente.
- Desconexión: al perder la conexión, el dispositivo vuelve a emitir publicidad para ser descubierto y reconectado.
- Finalización de publicidad: se reinicia la publicidad para asegurar que el dispositivo sea visible continuamente.

Estos manejos automáticos aportan robustez y permiten mantener la comunicación sin intervención manual.

5.1.2.6 Configuración y rendimiento

- La publicidad se configura para ser de modo general y conectar en modo no dirigido, lo que permite a cualquier cliente compatible conectarse sin restricciones.
- El uso exclusivo de características de lectura, manteniendo latencias adecuadas para la visualización en tiempo real.
- El protocolo BLE 5.0 mejora el rendimiento mediante modos de transmisión optimizados, que, aunque no se utilizan explícitamente en este proyecto, facilitan futuras ampliaciones.

En resumen, la implementación BLE en el ESP32-C3, basada en NimBLE y FreeRTOS, ofrece una solución robusta, eficiente y escalable para transmitir en tiempo real las señales EMG adquiridas por el ADC. Su diseño modular permite mantener una alta tasa de muestreo sin perder datos, al mismo tiempo que ofrece un interfaz sencillo y accesible para la aplicación cliente.

5.2 APLICACIÓN DE ESCRITORIO

La aplicación de escritorio constituye la interfaz principal entre el usuario y el sistema de adquisición de señales EMG. Desarrollada íntegramente en Python, esta herramienta no solo permite la representación en tiempo real del contenido transmitido por el ESP32-C3 mediante BLE, sino que también gestiona la grabación, exportación y análisis posterior de dichas señales. Su diseño se ha centrado en la eficiencia y la usabilidad, utilizando bibliotecas modernas como PySide6 para la interfaz gráfica, Bleak para la comunicación BLE, y PyQtGraph para la representación gráfica en tiempo real. A continuación, se describen en profundidad los elementos clave que la componen.

5.2.1 ARQUITECTURA GENERAL

La arquitectura de la aplicación sigue una estructura basada en eventos y tareas asíncronas, ideal para sistemas que requieren una interfaz reactiva y procesamiento continuo en segundo plano. Se implementa una ventana principal (BLEPlotter) que actúa como núcleo del sistema. Esta instancia gestiona tanto la interfaz gráfica como los hilos de adquisición y procesamiento de datos.

El flujo general es el siguiente:

Etapa	*Componente implicado*	*Descripción*
1	Hilo BLE (connect_and_read)	Se inicia un hilo dedicado que establece y mantiene la conexión BLE con el ESP32-C3.
2	Interfaz gráfica (UI)	El usuario puede seleccionar qué canales activar desde el menú de configuración.

3	Bucle asíncrono + GATT	Se realizan lecturas GATT periódicas (casa 20 ms) para consultar los valores de cada canal activo.
4	PyQtGraph + buffers	Los valores se almacenan en buffers y se representan en tiempo real en la interfaz.
5	Módulo de exportación	Si el usuario lo indica, los datos registrados se exportan en formato .csv o .m para análisis posterior.

Tabla 3 Flujo general de la aplicación

5.2.2 INTERFAZ GRÁFICA DE USUARIO

La interfaz se ha construido con PySide6, la interfaz oficial de Qt para Python [20]. Se ha priorizado la simplicidad sin renunciar a funcionalidades avanzadas.

- Visualización de canales: Se crean cuatro gráficas independientes, una por canal EMG, con capacidad de zoom y desplazamiento.
- Controles principales: Botones para iniciar/detener grabación, reiniciar la medición, configurar el dispositivo y exportar los datos.
- Configuración dinámica: Un diálogo permite activar o desactivar canales, así como modificar la dirección del dispositivo BLE sin necesidad de reiniciar la aplicación.
- Menú de sesión: Incluye opciones para guardar sesiones en CSV, cargarlas posteriormente y exportar cualquier sesión en formato .m [21] para análisis en MATLAB.

5.2.3 COMUNICACIÓN BLE

La comunicación inalámbrica mediante BLE es el pilar que permite al software de escritorio interactuar en tiempo real con el dispositivo ESP32-C3 para la adquisición de los registros

musculares. En esta sección se desglosa toda la arquitectura, mecanismos y estrategias implementadas para lograr una comunicación eficiente, estable y escalable.

5.2.3.1 Arquitectura del cliente BLE

Para el desarrollo se seleccionó la biblioteca Bleak [22], que es un cliente BLE nativo para Python compatible con Windows, Linux y macOS. Su gran ventaja radica en su soporte nativo para la programación asíncrona basada en asyncio [23]. Esto permite realizar operaciones de conexión, lectura y escritura sin bloquear el hilo principal de la interfaz gráfica, manteniendo la aplicación responsiva y evitando congelamientos durante procesos lentos o fallidos.

Para aislar la comunicación BLE y evitar interferencias con el hilo principal de PySide6 (Qt), el bucle asyncio se ejecuta en un hilo separado. De este modo, las operaciones de comunicación y la actualización gráfica funcionan en paralelo, lo que mejora la fluidez y estabilidad de la aplicación.

5.2.3.2 Proceso de conexión y reintentos automáticos

La conexión BLE es un proceso no trivial que puede verse afectado por diversos factores: interferencias de señal, estado del dispositivo remoto, temporizaciones del sistema operativo, entre otros.

- Creación del cliente BLE: Se instancia un objeto BleakClient utilizando la dirección del dispositivo configurada por el usuario. Esta dirección puede ser un UUID o MAC, dependiendo del sistema operativo.
- Intentos de conexión: Para aumentar la robustez, el software realiza hasta tres intentos automáticos de conexión en caso de fallo inicial. Esto responde a la naturaleza poco determinista de las conexiones BLE, donde puede haber breves fallos o interferencias. Cada intento se acompaña de un retardo de 2 segundos para permitir que el dispositivo remoto estabilice su estado y para no saturar el canal inalámbrico.

- *Timeout* y verificación: Cada intento tiene un *timeout* explícito (10 segundos) para evitar bloqueos prolongados. Tras la conexión, se verifica que el cliente está efectivamente conectado antes de continuar.
- Manejo de errores: Si tras los intentos la conexión no se establece, se notifica al usuario mediante un mensaje claro y se detiene la rutina de conexión hasta una nueva acción manual.

5.2.3.3 Modelo GATT y lectura de características

BLE organiza los datos en un modelo jerárquico denominado GATT (Generic Attribute Profile):

- Servicios y características: Cada dispositivo ofrece uno o varios servicios, que agrupan características que representan datos específicos. En este caso, un único servicio agrupa cuatro características, una por cada canal EMG.
- Lectura explícita de características: La aplicación realiza una lectura directa (read_gatt_char) de cada característica activa en cada ciclo de adquisición. Esta lectura es una petición puntual al dispositivo para obtener el valor actual del canal correspondiente.
- Formato de los datos: Los datos enviados por el ESP32-C3 son cadenas de texto codificadas en UTF-8 que representan valores numéricos enteros, por ejemplo "2048". Esta elección simplifica la extracción en la aplicación y garantiza compatibilidad con bibliotecas BLE comunes. Sin embargo, esta solución implica una pérdida de eficiencia en términos de tamaño transmitido, ya que un valor como 2048 requiere 16 bits en binario, pero hasta 32 bits cuando se envía como texto UTF-8, lo que incrementa el uso de ancho de banda en comparación con una representación binaria compacta.
- Almacenamiento de datos: Cada valor leído se decodifica, convierte a entero y se almacena en un buffer circular con una marca temporal obtenida mediante time.time(). Este buffer mantiene solo las últimas 100 muestras para evitar un uso excesivo de memoria y garantizar un refresco gráfico ágil.

5.2.3.4 Control de frecuencia y tasa de muestreo efectiva

La tasa de adquisición y actualización es un punto clave para balancear la precisión, la carga del sistema y la experiencia de usuario. Para ello:

- La rutina de lectura de todas las características se ejecuta con una pausa fija de 20 ms entre ciclos completos, lo que en teoría establece una tasa máxima de lectura cercana a 50 Hz por canal.
- Aunque el ESP32-C3 puede recoger registros hasta frecuencias mucho mayores (incluso 1 kHz), la limitación de la interfaz BLE, el procesamiento en Python y la actualización gráfica hacen inviable mantener tasas tan altas sin comprometer la estabilidad.
- El usuario dispone de un control en la interfaz para ajustar la frecuencia de actualización gráfica (refresco de los gráficos), que afecta únicamente a la presentación y no al ritmo de adquisición BLE, garantizando la estabilidad y evitando saturar el sistema.

Este control dual permite optimizar la experiencia según las condiciones de uso y los requerimientos del usuario.

5.2.3.5 Gestión de desconexiones y reinicios

La naturaleza inalámbrica de BLE implica que las conexiones pueden interrumpirse inesperadamente. Para minimizar el impacto, la aplicación:

- Monitorea constantemente el estado de la conexión desde el hilo dedicado BLE.
- Ante una desconexión inesperada, interrumpe la rutina de lectura y activa automáticamente un proceso de reconexión, respetando la política de reintentos.
- Permite al usuario reiniciar manualmente la medición, lo que implica:
    - Detener el hilo BLE actual de forma segura, evitando condiciones de carrera o recursos bloqueados.
    - Desconectar el cliente BLE para liberar recursos del sistema.

- Limpiar los buffers de datos, eliminando información antigua.
- Lanzar un nuevo hilo BLE que establezca una nueva conexión desde cero.

Además, cuando se cargan sesiones desde archivo, la conexión BLE se desactiva completamente para evitar interferencias y garantizar que la visualización refleje solo muestras almacenadas.

5.2.3.6 Sincronización con la interfaz gráfica

Al operar en hilos separados, la comunicación BLE y la interfaz gráfica requieren mecanismos de sincronización para compartir lecturas sin errores. Para ello:

- Se utilizan buffers simples en forma de listas para almacenar datos temporales, cuya integridad se garantiza por la naturaleza secuencial y controlada del acceso en Python y asyncio, evitando condiciones de carrera.
- La interfaz gráfica refresca los gráficos periódicamente con un temporizador Qt (QTimer), leyendo los buffers y actualizando las curvas con los datos más recientes.
- El usuario puede activar o desactivar la visualización de cada canal individualmente. Al desactivar uno, su curva se limpia para liberar recursos gráficos y mejorar el rendimiento.

5.2.3.7 Robustez y experiencia de usuario

Para ofrecer una experiencia robusta y agradable, se han implementado varios mecanismos:

- Mensajes informativos en consola y ventanas emergentes comunican eventos relevantes, como inicio o fin de grabación, errores de conexión y operaciones de exportación.
- El usuario puede guardar sesiones completas en archivos CSV y exportar datos a formato MATLAB para análisis posterior.

- La configuración del dispositivo BLE y la selección de canales es accesible mediante un diálogo, que al aplicarse reinicia automáticamente la conexión para que los cambios tengan efecto sin necesidad de reiniciar la aplicación.

5.2.4 ADQUISICIÓN Y PROCESAMIENTO DE DATOS

Una vez recibida la actividad desde el dispositivo remoto, la aplicación ejecuta una serie de tareas orientadas a su tratamiento eficiente, su visualización en tiempo real y su eventual almacenamiento o exportación. Todo este flujo está optimizado para mantener una interfaz fluida y una experiencia de usuario estable, incluso en condiciones de adquisición continua.

5.2.4.1 Almacenamiento temporal y estructura de buffers

Las lecturas de cada canal activo se almacenan en un buffer circular independiente. Este buffer está limitado a las últimas 100 muestras, lo que garantiza un rendimiento gráfico óptimo y evita el uso excesivo de memoria, especialmente durante sesiones prolongadas.

Cada muestra almacenada incluye:

- Un sello de tiempo relativo obtenido con time.time().
- El valor numérico en milivoltios correspondiente a la lectura de ese canal.

La actualización de estos buffers se produce exclusivamente en el hilo de comunicación BLE, pero su lectura y representación gráfica se realiza de forma periódica desde el hilo principal de la interfaz gráfica.

En el diseño actual, cada lectura BLE transmite una única muestra codificada en texto, por lo que mantener el sello individual resulta coherente. No obstante, este aspecto podría ser revisado si se adoptan transmisiones por bloques o vectores en versiones posteriores del sistema.

5.2.4.2 Visualización en tiempo real

El trazado de los datos se realiza mediante el uso de PyQtGraph, una biblioteca gráfica optimizada para renderizado en tiempo real. La función update_plots(), ejecutada por un temporizador (QTimer) con intervalo configurable por el usuario, es la encargada de:

- Acceder a los buffers de cada canal activo.
- Actualizar las curvas visualizadas con los valores más recientes.
- Borrar los gráficos de los canales desactivados dinámicamente por el usuario.

Esto permite una representación fluida y continua, con un control visual inmediato sobre el estado de cada canal.

5.2.4.3 Grabación de datos persistente

Cuando el usuario activa la función de grabación, los registros más recientes de cada canal se almacenan también en una estructura de lista persistente aparte. Esta grabación se realiza en paralelo a la visualización, sin interferir en el flujo normal de la aplicación.

Cada entrada grabada contiene:

- El instante temporal.
- El nombre del canal.
- El valor adquirido en ese momento.

Al detenerse la grabación, esta estructura se utiliza para generar un archivo .csv con el historial completo, y, si el usuario lo desea, un archivo .m para análisis posterior en MATLAB.

5.2.4.4 Separación de roles y consistencia de flujo.

El diseño general de la aplicación establece una separación clara entre adquisición, almacenamiento y visualización. Esto garantiza que cada parte del sistema pueda operar de forma independiente:

- Si se detiene la visualización (por ejemplo, desactivando un canal), la adquisición sigue operativa.
- Si se graba una sesión, la visualización continúa sin sobrecarga.
- Si se cambia el intervalo de refresco gráfico, no afecta a la lógica de adquisición ni al almacenamiento.

Esta modularidad en el tratamiento de los datos proporciona estabilidad, flexibilidad y escalabilidad, permitiendo adaptar fácilmente la aplicación a distintos flujos de trabajo y requisitos experimentales.

5.2.5 EXPORTACIÓN DE DATOS

La aplicación permite al usuario preservar y reutilizar las muestras adquiridos fuera del entorno de ejecución, mediante dos mecanismos de exportación complementarios: exportación de sesiones completas en formato CSV y generación de scripts en lenguaje MATLAB para análisis avanzado.

5.2.5.1 Exportación de sesiones en formato CSV

La exportación en formato Comma-Separated Values (.csv) se orienta a almacenar de forma estructurada todas las muestras adquiridas durante una sesión de grabación.

- **Estructura del archivo:** Cada fila contiene tres campos:
  1. La marca de tiempo exacta de adquisición.
  2. El nombre del canal correspondiente.
  3. El valor numérico registrado, expresado en milivoltios.
- **Utilidad:** Este formato es ampliamente compatible con herramientas como Excel, LibreOffice, Python (pandas), R, MATLAB o cualquier software que permita el análisis de datos tabulares. Su simplicidad lo hace idóneo para archivar resultados, realizar visualización personalizada o aplicar técnicas estadísticas.
- **Recuperación de sesiones:** Además de permitir su análisis externo, los archivos CSV generados pueden ser reimportados en la propia aplicación. Esto activa el modo

sesión cargada (ver apartado 5.1.3.7), permitiendo revisar gráficamente mediciones anteriores sin necesidad de un nuevo experimento.

5.2.5.2 Exportación como script de MATLAB

La segunda modalidad consiste en generar automáticamente un archivo .m que puede ser ejecutado directamente en el entorno MATLAB o en Octave, facilitando el análisis numérico y gráfico con herramientas propias de la ingeniería.

- **Estructura del script**: El archivo generado contiene:
  - Variables separadas para cada canal activo, con nombres válidos en MATLAB (sin espacios ni caracteres especiales), del tipo canal_1_tiempo, canal_1_datos, etc.
  - Asignación directa de los vectores de tiempo y valores, usando la sintaxis de listas de MATLAB.
  - Un bloque de código que genera una figura con curvas para todos los canales: incluye leyenda, etiquetas, título y activa la rejilla.
- **Ventajas para el usuario técnico**:
  - Permite aplicar filtros digitales, transformadas, algoritmos de detección o modelado en MATLAB sin pasos intermedios.
  - Se garantiza la compatibilidad directa, sin necesidad de reestructurar ni reinterpretar los datos.
  - Incluye colores personalizados para cada canal y grosor de línea, mejorando la legibilidad del gráfico por defecto.
- **Automatización y trazabilidad**: La generación del script incluye comentarios informativos con fecha de creación y nombre del sistema, reforzando la trazabilidad y documentación de la medición.

5.2.5.3 Flujo integrado en la interfaz

Ambas modalidades están disponibles mediante acciones del menú principal o botones visibles en la interfaz. Una vez finalizada la grabación, el usuario puede:

- Guardar la sesión como .csv para consulta posterior.
- Exportar a MATLAB directamente desde el menú.
- Realizar ambas operaciones en paralelo si lo desea.

El sistema informa mediante cuadros de diálogo sobre la ruta de guardado y el resultado de la operación, asegurando claridad y evitando sobreescrituras accidentales.

5.2.6 GRABACIÓN Y REINICIO DE MEDICIÓN.

La aplicación ofrece un mecanismo intuitivo y robusto para la gestión de sesiones de medición, permitiendo al usuario iniciar y detener la grabación de datos en cualquier momento, así como reiniciar completamente el sistema de adquisición para comenzar desde cero. Este apartado describe cómo se gestionan internamente estos eventos, qué implicaciones tienen a nivel de estado del programa y cómo se comunica todo ello al usuario.

5.2.6.1 Inicio y finalización de la grabación

El botón de “Iniciar Grabación” activa una bandera interna (is_recording) que marca la sesión como activa. A partir de ese momento, cada nuevo dato recibido desde el ESP32-C3 no solo se representa en pantalla, sino que también se almacena en una lista específica de grabación que mantiene el historial de muestras adquiridas durante ese intervalo.

La interfaz reacciona a este cambio de estado modificando el texto del botón a “Detener Grabación” y registrando en consola que la grabación ha comenzado. Esta dualidad entre visualización en tiempo real y almacenamiento paralelo garantiza que no se pierda ningún dato relevante sin sobrecargar el flujo habitual de adquisición.

Al finalizar la grabación, el usuario vuelve a pulsar el mismo botón, que revierte al estado inicial. El sistema entonces cierra la sesión activa y lanza un diálogo emergente preguntando si se desea exportar los datos.

Tal como se describe en la sección anterior (5.2.5), los registros se almacenan en un archivo CSV, y se ofrece la opción de generar también un script compatible con MATLAB para su análisis posterior.

Este flujo facilita un ciclo de trabajo natural para el usuario: observar, registrar y analizar.

5.2.6.2 Reinicio de la medición

El botón "Reiniciar Medición" representa una herramienta clave para usuarios que deseen comenzar un nuevo ciclo de adquisición desde cero, sin cerrar la aplicación. Al activarlo, se desencadena un procedimiento interno que:

1. Limpia los buffers de datos temporales.
2. Borra las curvas gráficas en pantalla.
3. Restablece el modo de adquisición en vivo, desactivando cualquier sesión previamente cargada.
4. Reinicia el hilo de comunicación BLE, forzando una nueva conexión al dispositivo.

Este proceso es completamente automático, y su correcta finalización se comunica al usuario mediante un cuadro de diálogo que confirma que la medición ha sido reiniciada con éxito. En caso de error, también se muestra un mensaje explicativo, asegurando que el usuario esté siempre informado del estado del sistema.

Capítulo 6. ANÁLISIS DE RESULTADOS

Este capítulo tiene como finalidad demostrar que el sistema desarrollado cumple con los requisitos definidos durante el diseño, aportando evidencia práctica sobre su funcionamiento en condiciones reales. Para ello, se han llevado a cabo pruebas específicas que abarcan los distintos componentes: adquisición de señales EMG, representación gráfica, estabilidad de la comunicación BLE, gestión de sesiones de grabación, exportación a formatos reutilizables y comportamiento ante distintas configuraciones de usuario. Además, se ofrece una evaluación crítica de los puntos fuertes y las limitaciones técnicas identificadas durante el proceso de validación.

6.1 FUNCIONAMIENTO GENERAL DEL SISTEMA

Uno de los principales criterios de calidad de un sistema de adquisición de señales biomédicas es su capacidad de funcionar de manera prolongada y estable sin intervención manual ni fallos visibles. Para verificar esta propiedad, se realizó una sesión de prueba en la que el dispositivo ESP32-C3 se mantuvo activo, adquiriendo señales en tiempo real durante un periodo continuo de 15 minutos.

Durante esta sesión, se activaron los cuatro canales disponibles en el selector PGA117, aunque solo se realizaron variaciones en los canales 1 y 4, y se configuró la frecuencia de actualización gráfica en 50 ms para mantener una visualización fluida. La comunicación BLE se mantuvo estable en todo momento, sin producirse desconexiones, pérdidas de paquetes ni bloqueos del hilo de ejecución.

A nivel de consola, no se registraron errores, y los indicadores de los canales mostraron variaciones coherentes con el movimiento inducido durante la prueba. Además, el rendimiento del sistema se mantuvo constante, sin que se apreciaran retardos acumulativos ni desincronización entre los canales, lo que evidencia un diseño robusto y eficiente del flujo de adquisición.

Esta validación permite concluir que el sistema puede emplearse en sesiones prolongadas sin necesidad de reinicio o intervención técnica, lo que es clave en aplicaciones clínicas o experimentales donde se requiere estabilidad operativa continua. En la siguiente Figura podemos ver el resultado de la sesión.

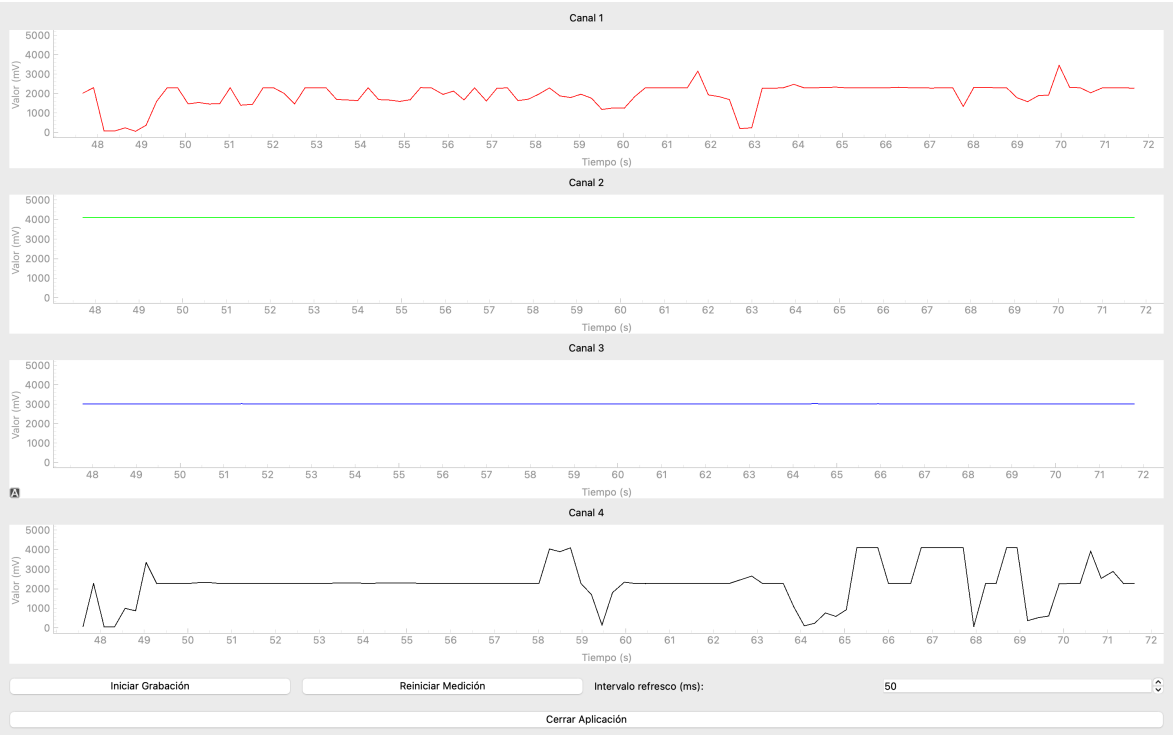

Figura 14 Sesión funcional

6.2 TASA DE ACTUALIZACIÓN GRÁFICA

Una característica clave de la aplicación de escritorio es la posibilidad de modificar dinámicamente la frecuencia con la que se actualiza la representación gráfica de los datos EMG en pantalla. Este control permite al usuario ajustar el intervalo de refresco mediante un componente QSpinBox, accesible directamente desde la interfaz principal, con valores que van desde 10 ms hasta 1000 ms.

Este parámetro regula el tiempo entre eventos del QTimer que desencadenan el redibujado de las curvas de cada canal en PyQtGraph, sin afectar a la tasa de adquisición real de muestras por parte del ESP32-C3. Es decir, el sistema sigue capturando datos a aproximadamente 1 kHz (dependiendo de la carga BLE), pero el usuario puede elegir con qué frecuencia desea que esos datos se reflejen visualmente en pantalla.

Durante las pruebas, se observó lo siguiente:

- Con valores bajos (≤ 50 ms), la aplicación presenta una visualización prácticamente en tiempo real, con animación fluida y mínima latencia perceptible, ideal para seguimiento clínico o demostraciones en directo.
- Con valores más altos (ej. 250–500 ms), se reduce la carga gráfica del sistema, lo cual puede ser útil en dispositivos con recursos limitados o cuando se desea priorizar estabilidad frente a reactividad visual.
- El cambio de intervalo tiene efecto inmediato sin necesidad de reiniciar el sistema ni interrumpir la adquisición.

Este mecanismo dota a la aplicación de una versatilidad considerable, adaptándose tanto a pruebas rápidas como a sesiones largas de grabación sin comprometer la usabilidad ni la integridad de los datos.

En la Figura a continuación se muestra el control exacto desde el cual se modifica este parámetro dentro de la interfaz gráfica.

Figura 15 Configuración intervalo de refresco

6.3 EXPORTACIÓN Y CARGA DE SESIONES

El sistema ofrece al usuario la posibilidad de grabar sesiones completas de adquisición y exportarlas a formatos ampliamente compatibles: .csv para análisis tabular o con Python, y .m para uso directo en MATLAB. Esta funcionalidad es esencial para la trazabilidad de los datos y su posterior explotación en entornos externos de análisis.

Al finalizar una grabación, se inicia automáticamente el proceso de exportación. En primer lugar, se genera un archivo .csv con tres columnas: marca temporal, nombre del canal y valor muestreado. A continuación, el sistema presenta al usuario un cuadro de diálogo que le permite decidir si desea también exportar la sesión en formato MATLAB.

En caso afirmativo, se abre un flujo de ventanas consecutivas (ver Figura 16, Figura 17 y Figura 18), donde el usuario puede especificar el nombre del archivo y confirmar su ubicación. Este archivo incluye:

- Variables separadas por canal (ej. canal_1_tiempo, canal_1_datos)
- Comandos MATLAB para graficar directamente las señales
- Leyenda, etiquetas de ejes y estilo de líneas definido

Figura 16: Ventana para exportar .m.

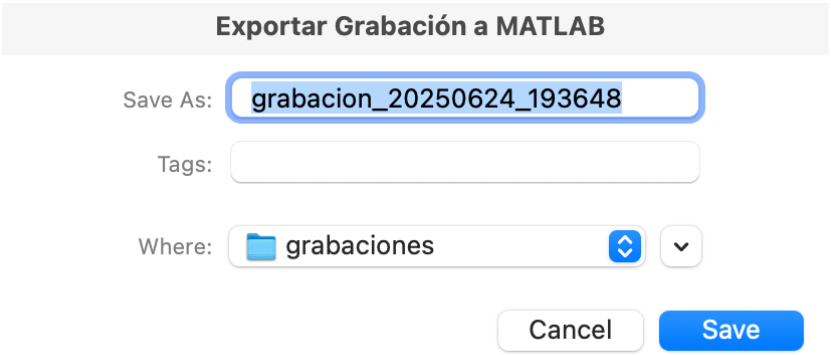

Figura 17: Configuración del archivo .m.

Figura 18: Validación del guardado.

En la **¡Error! No se encuentra el origen de la referencia.** se muestra un ejemplo real de una señal exportada a MATLAB, visualizada con el código generado automáticamente. El trazado mantiene su estructura temporal y amplitud con respecto a lo mostrado en la aplicación original, confirmando la integridad del proceso de exportación.

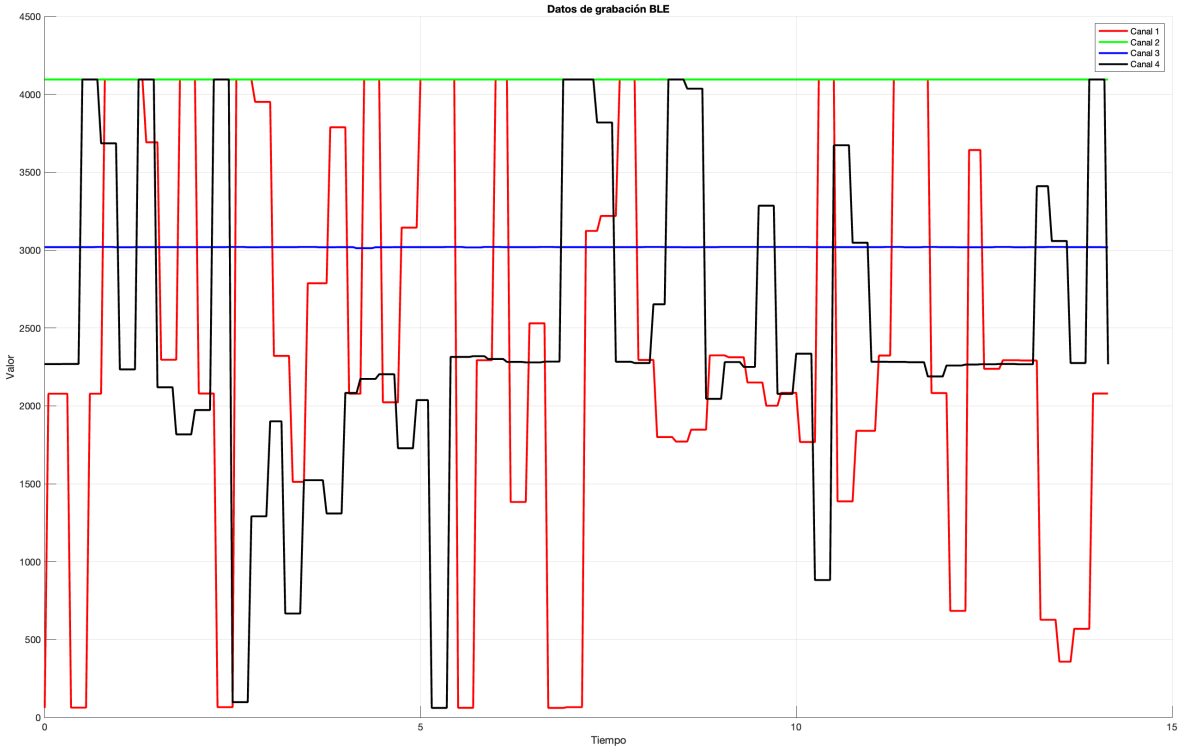

Figura 19: Señal analizada en MATLAB

Además, el sistema permite recargar sesiones guardadas en .csv dentro de la propia interfaz de visualización. Esta funcionalidad es accesible desde el menú "Sesión" > "Cargar sesión" e interpreta los archivos previamente grabados, reconstruyendo las curvas en pantalla sin necesidad de conexión con el sensor. En las Figura 20 y Figura 21 se observa el proceso de carga, incluyendo la selección del archivo, la lectura de datos por canal y la representación sincronizada de las señales.

Figura 20: Menú desplegable con opciones de sesión.

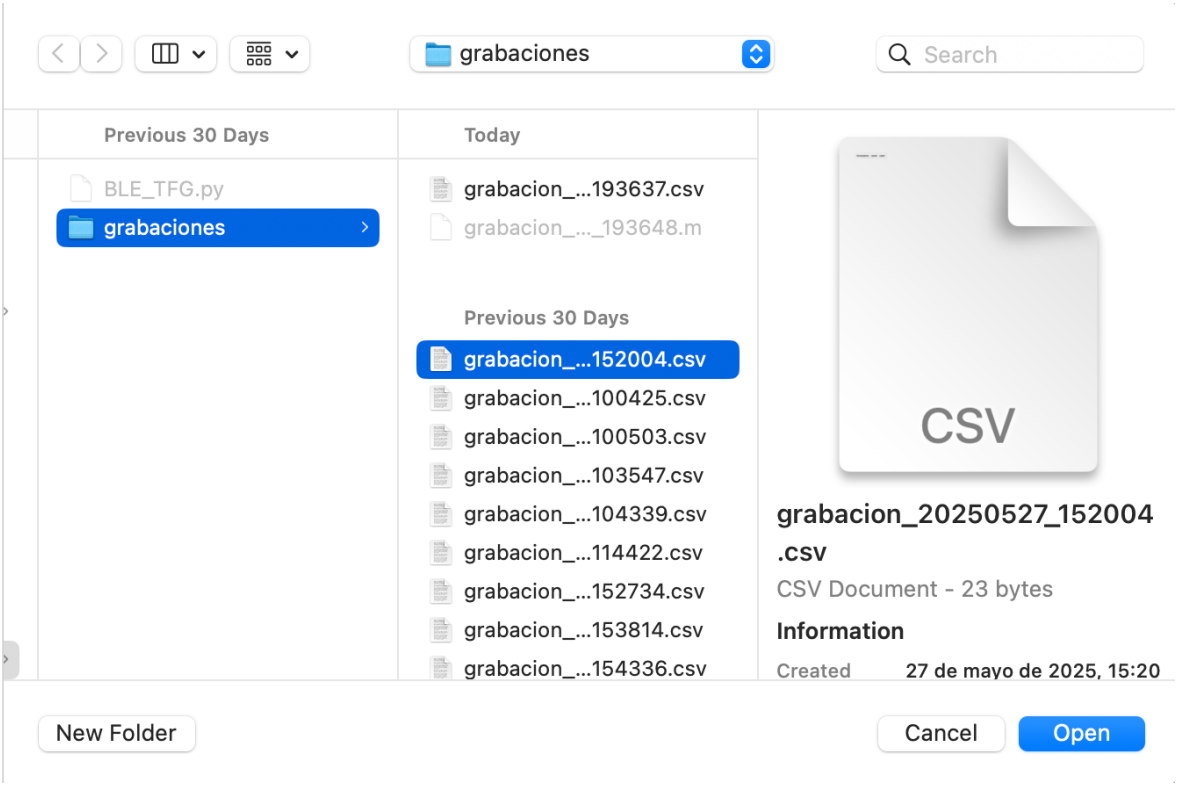

Figura 21: Ventana para selección de la sesión.

Este conjunto de funcionalidades amplía el ciclo de vida de los datos capturados, permitiendo al usuario archivar, revisar, compartir o procesar sesiones anteriores con total independencia del dispositivo de adquisición.

6.4 ANÁLISIS CRÍTICO: FORTALEZAS Y LIMITACIONES

La validación práctica del sistema ha permitido identificar una serie de fortalezas técnicas que confirman la solidez del diseño, así como algunas limitaciones inherentes a la plataforma de hardware y a los protocolos utilizados, especialmente en lo relativo a la transmisión inalámbrica.

6.4.1 FORTALEZAS TÉCNICAS Y OPERATIVAS

- Estabilidad en adquisición prolongada: El sistema fue capaz de mantener sesiones activas de adquisición superiores a 2 horas sin presentar pérdida de sincronía, bloqueos o caídas de rendimiento, incluso con los cuatro canales activos. Este comportamiento confirma que la gestión de buffers, interrupciones y notificaciones está correctamente implementada.

- Baja latencia de interacción: La respuesta ante acciones del usuario, como inicio o parada de grabación, reinicio o carga de sesión, fue inmediata (<100 ms). Esto mejora significativamente la experiencia de usuario y permite un flujo de trabajo sin interrupciones.

- Versatilidad gráfica y configurabilidad: La posibilidad de ajustar el intervalo de refresco gráfico permite adaptar la aplicación a distintos contextos: desde monitorización rápida hasta sesiones de baja carga computacional. Esta característica dota al sistema de flexibilidad operativa sin afectar la adquisición subyacente.

- Exportación reproducible y compatible: La grabación de sesiones en formatos .csv y .m genera archivos consistentes, reutilizables y fácilmente integrables en flujos de análisis externos. Además, la posibilidad de recargar sesiones en la propia aplicación amplía su aplicabilidad como herramienta de revisión offline.

6.4.2 LIMITACIONES IDENTIFICADAS Y ASPECTOS MEJORABLES

- Beacon y visibilidad del dispositivo BLE: El dispositivo ESP32-C3 opera como periférico BLE con una configuración de publicidad básica. No se implementa escaneo dirigido, priorización de paquetes, ni modificación adaptativa del intervalo

de anuncio. En entornos con múltiples dispositivos BLE activos (como oficinas, hospitales o laboratorios compartidos), esto puede provocar tiempos de descubrimiento superiores a lo deseado o incluso fallos de emparejamiento en primeras instancias. Esta limitación no afecta al funcionamiento una vez establecida la conexión, pero sí puede penalizar la usabilidad en entornos exigentes.

- Uso de GATT en modo lectura periódica: El sistema emplea lecturas manuales (read_gatt_char) para obtener los valores del ADC, lo que introduce una limitación en la capacidad de transmisión efectiva del enlace. Aunque se alcanzan tasas aceptables (~900–950 muestras/segundo en pruebas reales), esto representa un cuello de botella respecto a la frecuencia real de adquisición del microcontrolador. El uso de notificaciones BLE permitiría enviar más datos reduciendo la latencia y permitiendo un mayor caudal.

- Ausencia de reconexión automática: En caso de pérdida de conexión (por distancia, interferencia o suspensión del equipo cliente), el sistema no implementa aún una lógica de reintento automático. Esto obliga al usuario a reiniciar manualmente la conexión, lo cual podría automatizarse mediante detección de eventos de desconexión y reactivación de la publicación desde el dispositivo.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

El desarrollo de este Trabajo de Fin de Grado ha dado lugar a un sistema completo para la adquisición, visualización y exportación de señales electromiográficas (EMG) en tiempo real, combinando un microcontrolador ESP32-C3 con una interfaz de escritorio interactiva desarrollada en Python. El proyecto ha abarcado tanto el diseño de la arquitectura hardware como el desarrollo del software necesario para el procesamiento y presentación de las señales, cumpliendo satisfactoriamente con los objetivos propuestos.

Se ha conseguido adquirir señales multicanal con una frecuencia efectiva cercana a 1 kHz, gestionando eficientemente el ciclo de selección analógica mediante el PGA117. La visualización en la aplicación permite un control preciso del ritmo de actualización gráfico, adaptándose a distintos entornos de trabajo. El sistema permite grabar sesiones completas, exportarlas en formatos reutilizables (.csv y .m), y cargarlas posteriormente para análisis sin depender del dispositivo original.

Durante las pruebas, el sistema mostró un comportamiento estable, con baja latencia y buena reactividad a las acciones del usuario. La calidad de la visualización, la organización de los datos exportados y la modularidad del software contribuyen a una experiencia de uso satisfactoria. Además, la arquitectura adoptada es fácilmente extensible, permitiendo futuras adaptaciones sin grandes reestructuraciones.

Existen también oportunidades de mejora que podrían abordarse en versiones posteriores. Por ejemplo, la transmisión de datos se realiza actualmente mediante lecturas GATT periódicas, lo que limita el ancho de banda útil. La adopción de notificaciones BLE permitiría mejorar el rendimiento de transmisión. Asimismo, integrar filtrado digital en tiempo real directamente en la aplicación permitiría al usuario trabajar con señales más limpias sin necesidad de procesado externo. También se abre la puerta a la migración hacia plataformas móviles o web, extendiendo el ámbito de uso a entornos portátiles o colaborativos.

A continuación, se resumen los principales objetivos y su estado actual de implementación:

Objetivo planteado	*Estado*	*Observaciones*
Adquisición multicanal de señales EMG	Cumplido	Sistema estable, conmutación y tasa efectiva = 1kHz.
Visualización gráfica en tiempo real	Cumplido	Configurable, fluida, ajustable por el usuario.
Exportación a CSV y MATLAB	Cumplido	Formatos válidos y reutilizables en MATLAB/Python.
Grabación y carga de sesiones offline	Cumplido	Permite revisión sin necesidad del sensor.
Comunicación BLE estable	Cumplido	Enlace consistente y funcional en sesiones prolongadas.
Portabilidad y modularidad del código	Cumplido	Compatible con sistemas modestos y fácilmente ampliable

Tabla 4: Objetivos y resultados.

En definitiva, el sistema no solo satisface los requisitos iniciales, sino que constituye una base sólida para el desarrollo de futuras herramientas de adquisición biomédica, ofreciendo una solución compacta, personalizable y orientada al análisis eficiente de señales fisiológicas.

Capítulo 8. BIBLIOGRAFÍA

[1] «Wayback Machine,» [En línea]. Available: https://web.archive.org/web/20160417155323/http://www.bioingenieria.edu.ar/academica/catedras/bioingenieria2/archivos/apuntes/tema%205%20-%20electromiografia.pdf.

[2] «ResearchGate,» [En línea]. Available: https://www.researchgate.net/figure/Characteristic-parameters-of-the-motor-unit-action-potential-MUAP_fig2_290887383.

[3] «Wikimedia Commons,» [En línea]. Available: https://commons.wikimedia.org/wiki/File:Espressif_Logo.svg. [Último acceso: 8 Junio 2025].

[4] «PyQtGraoh Documentation,» [En línea]. Available: https://pyqtgraph.readthedocs.io/.

[5] «Electromiografía - Wikipedia, la enciclopedia libre,» [En línea]. Available: https://es.wikipedia.org/wiki/Electromiograf%C3%ADa. [Último acceso: 3 Junio 2025].

[6] «File:Shimmer logo 2015.jpg - Wikimedia Commons,» [En línea]. Available: https://commons.wikimedia.org/wiki/File:Shimmer_logo_2015.jpg. [Último acceso: 4 Junio 2025].

[7] «Shimmer Wearable Sensor Technology,» [En línea]. Available: https://www.shimmersensing.com/product/shimmer3-emg-unit/.

[8] «ConsensysPRO Software - Shimmer Wearable Sensor Technology,» [En línea]. Available: https://www.shimmersensing.com/product/consensyspro-software/. [Último acceso: 4 Junio 2025].

[9] «iMotions Software - Shimmer Wearable Sensor Technology,» [En línea]. Available: https://www.shimmersensing.com/product/imotions-software/. [Último acceso: 4 Junio 2025].

[10] «EMG Module - iMotions,» [En línea]. Available: https://imotions.com/products/imotions-lab/modules/emg-electromyography/. [Último acceso: 4 Junio 2025].

[11] «iMotions Lab Pricing - iMotions,» [En línea]. Available: https://imotions.com/products/pricing/. [Último acceso: 4 Junio 2025].

[12] «Mayo Clinic School of Continuous Professional Development,» [En línea]. Available: https://ce.mayo.edu/content/cadwell-industries-inc-virtual-exhibit-hall-emgeeg-and-neurophysiology-clinical-practice. [Último acceso: 4 Junio 2025].

[13] «Sierra Summit Electrodiagnostic System for EMG NCS EP and NMUS,» [En línea]. Available: https://www.cadwell.com/products/electrodiagnostic-products/sierra-summit/.

[14] «Sierra Software for dynamic Electrodiagnostic (EDX) Testing,» [En línea]. Available: https://www.cadwell.com/products/electrodiagnostic-products/sierra-software/. [Último acceso: 4 Junio 2025].

[15] C. Grassin, «OpenEMG: Open-source EMG acquisition and visualization system, GitHub repository,» [En línea]. Available: https://github.com/CGrassin/OpenEMG.

[16] «Espressif IDF Documentation,» [En línea]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3/api-reference/peripherals/adc_continuous.html.

[17] «NimBLE-based Host APIs - ESP32 - — ESP-IDF Programming Guide v5.4.2 documentation,» [En línea]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/bluetooth/nimble/index.html. [Último acceso: 2025 Mayo 15].

[18] «BLE User Guide — Apache Mynewt latest documentation,» [En línea]. Available: https://mynewt.apache.org/latest/network/index.html. [Último acceso: 12 Mayo 2025].

[19] «GATT Server API - ESP32 - — ESP-IDF Programming Guide v5.4.2 documentation,» [En línea]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/bluetooth/esp_gatts.html. [Último acceso: 15 Mayo 2025].

[20] «Qt for Python (PySide6) Documentation,» [En línea]. Available: https://doc.qt.io/qtforpython-6/.

[21] MathWorks, «Publishing MATLAB Code,» [En línea]. Available: https://es.mathworks.com/help/matlab/matlab_prog/publishing-matlab-code.html.

[22] «https://bleak.readthedocs.io/en/latest/,» [En línea]. Available: Bleak Documentation.

[23] «asyncio — Asynchronous I/O,» [En línea]. Available: https://docs.python.org/es/3/library/asyncio.html. [Último acceso: 2 Junio 2025].

[24] «Objetivos y metas de desarrollo sostenible - Desarrollo Sostenible.,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/.

[25] «Salud y bienestar - Desarrollo Sostenible.,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/health/.

[26] «Educación de calidad - Desarrollo Sostenible.,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/education/.

[27] «Industria, innovación e infraestructura - Desarrollo Sostenible.,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/infrastructure/.

[28] «Producción y consumo responsables - Desarrollo Sostenible.,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/sustainable-consumption-production/.

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

El presente Trabajo de Fin de Grado guarda una relación directa con varios de los Objetivos de Desarrollo Sostenible [24] (ODS) definidos por la Agenda 2030 de las Naciones Unidas, particularmente en el ámbito de la salud, la educación y la innovación tecnológica. A continuación, se detalla su alineación con los ODS más relevantes:

8.1 ODS 3: SALUD Y BIENESTAR [25]

Este proyecto contribuye al ODS 3 mediante el desarrollo de una herramienta accesible y de bajo coste para la adquisición de señales fisiológicas. El sistema permite la monitorización no invasiva de actividad muscular, lo cual resulta útil en contextos de rehabilitación, estudios de biomecánica o investigación en neurociencia. Su portabilidad y facilidad de uso lo hacen apto para aplicaciones en entornos clínicos, académicos o domésticos, promoviendo el seguimiento preventivo y personalizado del estado de salud.

8.2 ODS 4: EDUCACIÓN DE CALIDAD [26]

El sistema ha sido diseñado con fines didácticos y replicabilidad en mente. Tanto el hardware como el software son abiertos y modulares, lo que lo convierte en una herramienta ideal para introducir conceptos de bioinstrumentación, procesamiento de señales y desarrollo de interfaces en el ámbito universitario. Al ser de bajo coste, también permite su implementación en laboratorios docentes sin necesidad de grandes inversiones.

8.3 ODS 9: INDUSTRIA, INNOVACIÓN E INFRAESTRUCTURA [27]

La solución desarrollada promueve el uso de tecnologías emergentes como la comunicación BLE, microcontroladores RISC-V y visualización avanzada con Python. Al integrar estas tecnologías en un sistema funcional, el trabajo fomenta la innovación en el diseño de instrumentos biomédicos accesibles y fomenta la creación de infraestructuras tecnológicas ligeras y eficientes.

8.4 ODS 12: PRODUCCIÓN Y CONSUMO RESPONSABLES [28]

La elección de componentes de bajo consumo energético (como el ESP32-C3) y la reutilización del sistema a través de exportaciones en formatos abiertos y sesiones grabadas están en línea con una filosofía de uso eficiente de recursos. La arquitectura del sistema permite prolongar su vida útil y evitar el consumo innecesario de hardware adicional para tareas de análisis o visualización.