# Neural Analytics: una herramienta software basada en redes neuronales de procesado de EEGs

Bachelor's Degree in Computer Engineering

**Final Degree Project**

**Author:**
Sergio Martínez Aznar

**Supervisor(s):**
Antonio Molina Picó

**Academic Year:**
2024/2025

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

*Para aquellos que una vez soñaron con volar alto, y lo arriesgaron todo.*

# Resumen

Este proyecto presenta el desarrollo de un sistema de control domótico basado en interfaces cerebro-computadora (BCI). El objetivo principal es implementar un sistema no invasivo que permita la integración de señales cerebrales con dispositivos de iluminación inteligente, específicamente bombillas TP-Link Tapo.

El sistema utiliza la diadema Brainbit como dispositivo de adquisición de señales electroencefalográficas (EEG), procesando estas señales mediante una arquitectura de software que combina Deep Learning para procesamiento y una solución desarrollada en Rust para el consumo del modelo e interacción con el sistema de iluminación.

El marco teórico aborda el desarrollo de un modelo de clasificación basado en señales EEG, explorando técnicas de procesamiento de señales y aprendizaje automático para la interpretación precisa de patrones cerebrales. Además, se profundiza en los requisitos técnicos y normativos necesarios para el desarrollo de dispositivos médicos, con especial énfasis en el estándar IEC 62304 para procesos del ciclo de vida del software de dispositivos médicos, y la implementación de sistemas operativos en tiempo real (RTOS) para garantizar la fiabilidad y seguridad del sistema.

La solución propuesta integra tecnologías modernas de procesamiento de señales cerebrales con sistemas de domótica, creando una interfaz natural e intuitiva para el control del entorno doméstico. Este proyecto representa un paso hacia la democratización de las interfaces cerebro-computadora en aplicaciones cotidianas.

**Palabras clave:** Interfaz cerebro-computadora, BCI, Domótica, Aprendizaje profundo, Rust, EEG, RTOS, Certificación médica

# Abstract

This project presents the development of a home automation control system based on brain-computer interfaces (BCI). The main objective is to implement a non-invasive system that enables the integration of brain signals with smart lighting devices, specifically TP-Link Tapo bulbs.

The system uses the Brainbit headband as an electroencephalographic (EEG) signal acquisition device, processing these signals through a software architecture that combines PyTorch for training deep learning models and Rust for the development of the inference engine.

The theoretical framework addresses the development of a classification model based on EEG signals, exploring signal processing and machine learning techniques for accurate interpretation of brain patterns. Additionally, it delves into the technical and regulatory requirements necessary for medical device development, with special emphasis on the IEC 62304 standard for medical device software life cycle processes, and the implementation of real-time operating systems (RTOS) to ensure system reliability and safety.

The proposed solution integrates modern brain signal processing technologies with home automation systems, creating a natural and intuitive interface for controlling the home environment. This project represents a step towards the democratization of brain-computer interfaces in everyday applications.

**Keywords:** Brain-computer interface, BCI, Home automation, Deep learning, Rust, PyTorch, EEG, RTOS, Medical certification

# Índice general

# Índice de figuras

# Capítulo 1

# Introducción

El presente trabajo aborda el diseño y desarrollo de un innovador sistema de automatización del hogar que integra tecnología de interfaz cerebro-computadora (BCI) con sistemas de iluminación inteligentes. La finalidad principal consiste en desarrollar una solución no invasiva que facilite el control del entorno doméstico mediante la lectura e interpretación de ondas cerebrales, empleando dispositivos de iluminación TP-Link Tapo como elementos de control.

La implementación de este proyecto se fundamenta en dos pilares: el procesamiento de señales electroencefalográficas (EEG) mediante aprendizaje profundo y el cumplimiento de la normativa UNE-EN 62304 para dispositivos médicos. Para garantizar la respuesta en tiempo real del sistema, se utiliza Wind River Linux como sistema operativo base.

## 1.1. Motivación

Desde que inicié mi formación en ingeniería, siempre he sentido una profunda fascinación por las interfaces cerebro-computadora (BCI) y sus posibles aplicaciones. Este proyecto representa una perfecta síntesis de mis pasiones e inquietudes: la tecnología, los sistemas operativos en tiempo real, la medicina y la innovación. La oportunidad de trabajar en un sistema que combine el procesamiento de señales cerebrales con el control domótico me permite explorar un campo que considero revolucionario para la interacción persona-máquina.

La decisión de trabajar con actuadores domóticos comunes, específicamente bombillas inteligentes, no es casual. Permite demostrar de manera sencilla y visual el funcionamiento del sistema BCI, haciendo tangible una tecnología que a menudo puede parecer abstracta o inalcanzable. Además, este enfoque práctico facilita la comprensión del sistema y su potencial impacto en la vida cotidiana.

El aspecto normativo del proyecto, aunque técnicamente desafiante, representa para mí una oportunidad única de entender cómo llevar una idea innovadora desde el concepto hasta un producto viable en el mercado médico. El proceso de cumplir con la normativa UNE-EN 62304, implementar un sistema en tiempo real y desarrollar una metodología robusta, lejos de ser una limitación, ha enriquecido significativamente mi comprensión de lo que significa desarrollar tecnología médica responsable y segura.

## 1.2. Objetivos

Este proyecto tiene como finalidad principal el desarrollo de un sistema de control domótico basado en interfaces cerebro-computadora, buscando hacer más accesible esta tecnología en entornos cotidianos.

Para alcanzar esta meta, se han establecido los siguientes objetivos específicos:

- **Cumplimiento del estándar UNE-EN 62304**: Crear una solución que cumpla con la normativa UNE-EN 62304, garantizando la seguridad y fiabilidad del software médico mediante una metodología de desarrollo rigurosa y documentada.
- **Implementar un clasificador de señales EEG**: Implementar un modelo de aprendizaje profundo para la clasificación de señales EEG, utilizando PyTorch como framework de desarrollo y centrándose en la detección de patrones asociados a la visualización de colores.
- **Desarrollar un sistema de control BCI**: Crear un sistema que permita el control de dispositivos de iluminación inteligente mediante señales EEG, utilizando la diadema Brainbit como dispositivo de adquisición de señales.
- **Generar una imagen para la RPi4**: Generar una imagen de sistema operativo personalizada basada en Wind River Linux, asegurando un entorno de ejecución con garantías de tiempo real blando para el procesamiento de señales EEG.

Estos objetivos se han definido considerando tanto los aspectos técnicos como normativos del proyecto, buscando un equilibrio entre la innovación tecnológica y la viabilidad práctica en entornos reales.

## 1.3. Metodología

El desarrollo del proyecto sigue una metodología estructurada en varias fases:

### 1.3.1. Fase de Investigación

- Estudio de la literatura sobre procesamiento de señales EEG
- Análisis de requisitos normativos para dispositivos médicos
- Evaluación de tecnologías y frameworks disponibles

### 1.3.2. Fase de Desarrollo

- Implementación del modelo de clasificación en PyTorch
- Desarrollo del motor de inferencia en Rust
- Integración con el SDK de BrainFlow
- Creación de la imagen personalizada de Wind River Linux

### 1.3.3. Fase de Validación

- Pruebas de rendimiento y fiabilidad
- Verificación del cumplimiento normativo
- Evaluación de la usabilidad del sistema

La vertiente práctica del proyecto comprende una descripción exhaustiva del proceso de desarrollo, incluyendo el entrenamiento del modelo de aprendizaje profundo, su incorporación al sistema global y la creación de una imagen personalizada para la Raspberry Pi 4.

# Parte I

# Marco Teórico

# Capítulo 2

# Normativa UNE-EN 62304

La norma **UNE-EN 62304:2007/A1:2016** Asociación Española de Normalización (2016) es la versión española de la norma **IEC 62304:2006/A1:2015**, adoptada como norma europea EN 62304:2006/A1:2015. Esta normativa establece los requisitos para los **procesos del ciclo de vida del software** en **dispositivos médicos**, asegurando su desarrollo, mantenimiento y gestión de riesgos de acuerdo con estándares internacionales.

## 2.1. Objetivo y Alcance

El propósito de la UNE-EN 62304 Asociación Española de Normalización (2016) es definir un marco normativo para la **gestión del ciclo de vida del software** en dispositivos médicos, asegurando su seguridad y eficacia.

Esta norma se aplica a:

- **Software que es un dispositivo médico en sí mismo.**
- **Software embebido en dispositivos médicos.**
- **Software utilizado en entornos médicos para diagnóstico, monitoreo o tratamiento.**

El estándar establece **procesos y actividades** que los fabricantes deben seguir, incluyendo:

- Planificación del desarrollo del software.
- Análisis de requisitos y arquitectura del software.
- Implementación, integración, pruebas y verificación.
- Gestión del mantenimiento y resolución de problemas.
- Gestión del riesgo asociado al software.
- Gestión de la configuración y cambios.

## 2.2. Clasificación del Software

La norma clasifica el software en **tres niveles de seguridad** según el riesgo que pueda representar para el paciente o el operador:

- **Clase A**: El software no puede causar daño en ninguna circunstancia.
- **Clase B**: El software puede contribuir a una situación peligrosa, pero el daño potencial es **no serio**.
- **Clase C**: El software puede contribuir a una situación peligrosa con **riesgo de daño serio o muerte**.

## 2.3. Cumplimiento y Aplicación durante el proyecto

Dado el carácter estricto, y la propia necesidad de garantizar el cumplimiento de la UNE-EN 62304 Asociación Española de Normalización (2016) en este trabajo, se seguirá un enfoque basado en la **gestión del ciclo de vida del software** y la evaluación de riesgos. Se adoptarán buenas prácticas de ingeniería de software y se documentarán las actividades necesarias para cumplir con los requisitos de seguridad y calidad establecidos por la normativa.

En este proyecto, utilizaremos un **modelo de desarrollo iterativo e incremental** inspirado en metodologías ágiles como Scrum, adaptado a las necesidades específicas del desarrollo de software médico. Este enfoque nos permitirá una mayor flexibilidad y capacidad de adaptación a los cambios, así como una entrega continua de valor al cliente.

Considerando que el sistema únicamente controla el encendido y apagado de una bombilla inteligente TP-Link Tapo de manera remota, hemos clasificado el software como de **Clase A**. Esta clasificación se justifica porque el software no puede causar daño al usuario en ninguna circunstancia, ya que:

- La diadema BrainBit es un dispositivo no invasivo de lectura pasiva
- El control se realiza sobre una bombilla doméstica de baja tensión
- No hay interacción directa con sistemas críticos o vitales

A pesar de esta clasificación de bajo riesgo, mantendremos buenas prácticas de desarrollo y documentación para asegurar la calidad del software.

Dado que esta normativa es un **requisito esencial** para el desarrollo de software en el mercado sanitario español y europeo, su correcta implementación garantizará la viabilidad del producto en entornos clínicos y su aceptación por parte de los organismos reguladores.

# Capítulo 3

# Regiones implicadas del cerebro

## 3.1. Introducción a las Regiones Cerebrales Funcionales

El cerebro humano es un sistema altamente organizado en el que diferentes regiones trabajan de manera especializada pero interconectada para procesar la información y generar respuestas conductuales adecuadas. Como indican en su libro "Principios de Neurociencia"los autores Kandel, Jessell y Schwartz (2001), el estudio de la neurociencia ha demostrado que la división funcional del cerebro permite analizar la forma en que los procesos perceptivos, cognitivos y emocionales emergen de la actividad neuronal distribuida. En este proyecto, nos enfocamos en dos regiones clave para la percepción y la memoria del color: el lóbulo occipital y el lóbulo temporal.

Utilizando un sistema BCI basado en EEG, empleamos electrodos en ubicaciones específicas del sistema internacional 10-20: O1 y O2 en el lóbulo occipital, responsables del procesamiento primario de la información visual, y T3 y T4 en el lóbulo temporal, donde la información visual se asocia con memorias previas y respuestas emocionales.

Gracias a esta organización funcional, podemos estudiar la relación entre percepción e imaginación del color, explorando su impacto en la memoria y la experiencia subjetiva.

## 3.2. El Lóbulo Temporal y la Memoria Visual

El lóbulo temporal es crucial en la memoria visual y la asociación semántica de colores. Los electrodos T3 y T4 captan la actividad relacionada con:

- **Hipocampo:** Responsable de la consolidación de memorias visuales y asociaciones cromáticas.
- **Corteza temporal medial:** Procesa la identificación y categoría de los colores.
- **Amígdala:** Relaciona el color con respuestas emocionales, modulando el impacto afectivo de los colores percibidos o imaginados.
- **Corteza entorrinal:** Conecta el hipocampo con otras áreas corticales, permitiendo que las asociaciones cromáticas se integren en experiencias más complejas.

Cuando una persona recuerda un color, T3 y T4 reflejan la activación de estos circuitos, permitiendo analizar la relación entre percepción visual y memoria. Estudios como los de Squire y Zola-Morgan (1991) han demostrado que lesiones en el lóbulo temporal pueden afectar la capacidad de recuperar memorias visuales, lo que refuerza su papel en el almacenamiento de información sensorial.

## 3.3. El Lóbulo Occipital y la Percepción del Color

El lóbulo occipital es la principal región del cerebro para el procesamiento de la información visual. Los electrodos O1 y O2 capturan la actividad en:

- **Corteza visual primaria (V1):** Primer procesamiento del color y detección de longitudes de onda. [1]
- **V4 (corteza visual asociativa):** Especializada en la interpretación y categorización de colores, estableciendo una conexión funcional con las áreas temporales para asignar significados semánticos.

Estudios como los de Brouwer y Heeger (2013) han demostrado que la actividad en V4 puede ser utilizada para clasificar diferentes colores percibidos o imaginados, lo que refuerza la validez de nuestros electrodos O1 y O2 para el análisis de patrones cromáticos en EEG.

[1] Es importante distinguir entre O1/O2, que son las *posiciones de los electrodos* según el sistema internacional 10-20, y V1/V4, que son *áreas funcionales del córtex visual* (designaciones de Brodmann) cuya actividad es registrada por dichos electrodos.

## 3.4. Integración entre Memoria y Percepción

El procesamiento del color en el cerebro, simplificándolo al absurdo, sigue un flujo distribuido:

1. O1/O2 detectan y analizan las características del color percibido o imaginado.
2. La información es enviada a T3/T4 para su comparación con recuerdos previos y asociaciones emocionales.
3. Se generan asociaciones semánticas y afectivas, determinando la experiencia subjetiva del color.

Estudios como los de Rissman y Wagner (2012) han mostrado que patrones de activación en el lóbulo temporal pueden predecir si un individuo reconoce un color previamente visto, lo que refuerza la idea de que los recuerdos cromáticos tienen una representación neural clara.

## 3.5. Aplicaciones en Interfaces Cerebro-Computadora

La integración de la percepción y la memoria del color en un sistema BCI tiene varias aplicaciones potenciales:

- Diferenciar entre percepción real e imaginada de un color.
- Implementar sistemas BCI basados en selección cromática.
- Explorar la relación entre color, memoria y emociones para aplicaciones en neurotecnología.

De este modo, podemos exploar mediante una interfaz cerebro-computadora cómo la percepción y la memoria del color se integran en la experiencia subjetiva, permitiendo así que pueda ser interpretado por un computador y que este pueda realizar acciones en función de la información recibida.

## 3.6. Aplicaciones durante este proyecto

Los electrodos O1, O2, T3 y T4 capturan información clave sobre la percepción y la memoria del color. Su combinación permite analizar la reconstrucción mental de colores y su impacto en la experiencia subjetiva, formando la base de nuestro sistema BCI. La inclusión de referencias a trabajos clave refuerza la validez del enfoque adoptado en este estudio.

# Capítulo 4

# Sistemas operativos en Tiempo Real

Los sistemas operativos en tiempo real (RTOS) Siewert and Pratt (2016) constituyen una rama especializada del software orientada a garantizar la previsibilidad temporal en entornos que requieren respuestas precisas. A diferencia de los sistemas operativos convencionales, estos sistemas priorizan la **predictibilidad temporal** sobre la velocidad de procesamiento, asegurando que cada operación se ejecute dentro de intervalos temporales específicos.

El concepto de computación en tiempo real emerge de la necesidad de procesar y responder a eventos del mundo físico con restricciones temporales bien definidas. En un RTOS, la corrección del sistema no solo depende de la exactitud lógica de los resultados, sino también del momento en que estos se producen. Esta dualidad en los requisitos (*corrección lógica + corrección temporal*) distingue fundamentalmente a los RTOS de los sistemas operativos de propósito general.

La arquitectura de un RTOS se caracteriza por varios componentes esenciales:

- **Planificador determinista**: Garantiza que las tareas críticas se ejecuten en momentos predecibles
- **Gestión de interrupciones**: Manejo prioritario de eventos externos con latencias acotadas
- **Gestión de memoria**: Esquemas de asignación y liberación que evitan indeterminismos temporales

En el ámbito de los sistemas embebidos, principal campo de aplicación de estos sistemas, el RTOS funciona como intermediario entre los componentes físicos y las operaciones de control. Un caso ilustrativo son los sistemas de seguridad en vehículos, donde cualquier retraso, incluso de microsegundos, podría resultar crítico.

En algunos casos, estos requisitos de fiabilidad y predictibilidad temporal se extienden a sistemas de propósito general, especialmente en aplicaciones médicas y de consumo que demandan garantías temporales.

Un ejemplo paradigmático son los dispositivos médicos implantables, donde la fiabilidad y predictibilidad temporal son requisitos fundamentales para garantizar la seguridad del paciente. Esta expansión ha llevado al desarrollo de estrictos marcos regulatorios y procesos de certificación específicos para cada sector de aplicación.

## 4.1. Taxonomía de Sistemas en Tiempo Real

La clasificación de los sistemas operativos en tiempo real se fundamenta principalmente en la criticidad de sus restricciones temporales. Esta taxonomía, establecida inicialmente por Liu y Layland en 1973 Siewert and Pratt (2016), ha evolucionado para adaptarse a las necesidades modernas de la computación en tiempo real. La distinción fundamental se establece entre sistemas estrictos (*hard real-time*) y flexibles (*soft real-time*), aunque algunos autores reconocen una categoría intermedia denominada *firm real-time*.

### 4.1.1. Sistemas de Tiempo Real Estricto

Los sistemas de tiempo real estricto (**hard real-time**) se caracterizan por la intolerancia absoluta a desviaciones temporales. En estos sistemas, el incumplimiento de un plazo temporal se considera un fallo catastrófico del sistema. La expresión matemática que define su comportamiento es:

$$\forall t \in T : R(t) \leq D(t) \tag{4.1}$$

Figura 4.1: Ecuación de sistemas de tiempo real estricto.

donde $R(t)$ representa el tiempo de respuesta y $D(t)$ el plazo temporal máximo permitido.

Ejemplos paradigmáticos incluyen:

- **Sistemas de control nuclear**: Donde los tiempos de respuesta deben ser absolutamente predecibles para garantizar la seguridad
- **Mecanismos de frenado electrónico**: El ABS debe responder en microsegundos para prevenir accidentes
- **Sistemas quirúrgicos robotizados**: Requieren sincronización precisa para operaciones de alta precisión

Su implementación requiere sistemas de planificación **preemptivos** con prioridades estáticas, donde el tiempo máximo de ejecución (WCET) debe ser predecible y verificable. La planificación típicamente se basa en el algoritmo Rate Monotonic (RM) o Earliest Deadline First (EDF).

### 4.1.2. Sistemas de Tiempo Real Flexible

Los sistemas de tiempo real flexible (**soft real-time**) toleran cierta variabilidad en el cumplimiento de plazos temporales, operando bajo un modelo probabilístico donde:

$$P(R(t) \leq D(t)) \geq p_{min} \tag{4.2}$$

Figura 4.2: Ecuación de sistemas de tiempo real flexible.

siendo $p_{min}$ el nivel mínimo aceptable de cumplimiento temporal.

Ejemplos representativos incluyen:

- **Plataformas de streaming multimedia**: Donde ocasionales pérdidas de frames son aceptables
- **Redes de monitorización industrial**: Con tolerancia a retrasos ocasionales en la actualización de datos
- **Sistemas de trading automatizado**: Donde el rendimiento promedio es más importante que garantías absolutas

Estos sistemas utilizan habitualmente planificadores basados en **tiempo compartido** con prioridades dinámicas, priorizando la optimización del rendimiento promedio sobre las garantías temporales absolutas. Las políticas de planificación suelen incluir variantes de Round Robin y planificación por prioridades dinámicas.

### 4.1.3. Consideraciones de Implementación

La elección entre implementaciones estrictas y flexibles debe considerar:

- **Análisis de Riesgos**: Evaluación de consecuencias por fallos temporales
- **Recursos Disponibles**: Capacidad de procesamiento y memoria
- **Costes**: Balance entre garantías temporales y complejidad del sistema
- **Certificación**: Requisitos regulatorios del dominio de aplicación

## 4.2. Soluciones Comerciales para Hard Real-Time

### 4.2.1. VxWorks (Wind River Systems)

VxWorks, desarrollado por Wind River Systems, representa el estándar industrial en sistemas embebidos críticos, especialmente en sectores como la aviónica, espacial y médico. Sus características principales incluyen:

**Certificaciones y Cumplimiento Normativo**

- DO-178C Level A para sistemas aeroespaciales
- IEC 62304 para dispositivos médicos
- ISO 26262 ASIL D para automoción

**Características Técnicas**

- **Kernel**: Microkernel determinista con tiempos de interrupción $\leq 50$ ns
- **Memoria**: MMU con protección y aislamiento de espacios de memoria
- **Scheduling**: Planificador con 256 niveles de prioridad y herencia de prioridad
- **IPC**: Mecanismos de comunicación con latencia determinista
- **Multiprocesamiento**: Soporte para SMP y AMP con aislamiento de cores

### 4.2.2. QNX Neutrino (BlackBerry)

QNX Neutrino, adquirido por BlackBerry, destaca por su arquitectura de microkernel distribuido y su alto nivel de fiabilidad:

**Arquitectura**

- **Microkernel**: Núcleo de menos de 100KB
- **Servicios**: Arquitectura modular con servicios en espacio de usuario
- **IPC**: Sistema de mensajería síncrona y asíncrona con copy-on-write
- **Recuperación**: Capacidad de reinicio de componentes sin afectar al sistema

**Características Avanzadas**

- **Tiempo Real**: Garantías temporales con latencias $\leq 100$ $\mu$s
- **Seguridad**: Modelo de seguridad adaptativo con ASLR
- **Certificaciones**: IEC 61508 SIL3, IEC 62304 Clase C

### 4.2.3. Zephyr RTOS (Linux Foundation)

Zephyr representa la alternativa open-source para sistemas embebidos críticos:

**Diseño y Arquitectura**

- **Kernel**: Monolítico o microkernel configurable
- **Footprint**: Desde 8KB hasta configuraciones completas de 512KB
- **Scheduling**: Planificador configurable con hasta 32 niveles de prioridad
- **Certificación**: Proceso de certificación para IEC 61508 SIL 3/4

**Características Destacadas**

- **Drivers**: Más de 350 drivers para diferentes periféricos
- **Networking**: Soporte nativo para protocolos IoT (BLE, Thread, LoRaWAN)
- **Seguridad**: Subsistema de seguridad con aislamiento de memoria
- **Desarrollo**: Herramientas de desarrollo y depuración avanzadas

## 4.3. Soluciones Comerciales para Soft Real-Time

### 4.3.1. Wind River Linux (Wind River Systems)

Wind River Linux representa una solución empresarial certificada, basada en el Proyecto Yocto, específicamente diseñada para el desarrollo de sistemas embebidos que requieren garantías temporales flexibles:

**Características Principales**

- **Base**: Kernel Linux 5.10 LTS con parche PREEMPT_RT
- **Certificaciones**: ISO 9001:2015 y precertificación IEC 62304
- **Seguridad**: Monitorización continua de CVEs y mitigación
- **Cumplimiento**: Documentación SBOM y Open Chain 2.1

**Capacidades Industriales**

- **Soporte**: Mantenimiento garantizado de 5 años con extensión LTS
- **Actualizaciones**: Sistema OTA seguro mediante OSTree
- **Validación**: Más de 60.000 casos de prueba automatizados
- **Servicios**: Soporte profesional y consultoría disponible

### 4.3.2. Poky Linux (Proyecto Yocto)

Poky constituye la distribución de referencia del Proyecto Yocto, proporcionando una base para el desarrollo de sistemas Linux embebidos con capacidades de tiempo real flexible:

**Características Técnicas**

- **Kernel**: Linux con soporte opcional para PREEMPT_RT
- **Tiempo Real**: Latencias configurables según necesidades
- **Optimización**: Control fino sobre el tamaño y rendimiento
- **Personalización**: Capacidad de eliminar componentes innecesarios

**Consideraciones de Desarrollo**

- **Mantenimiento**: Actualización manual de parches de seguridad
- **Soporte**: Basado en la comunidad, sin garantías comerciales
- **Certificación**: Requiere proceso de certificación propio
- **Validación**: Necesidad de desarrollar pruebas específicas

## 4.4. Elección de RTOS para el Proyecto

La elección de Wind River Linux como sistema operativo para este proyecto se fundamenta en varios factores críticos:

### 4.4.1. Requisitos Temporales del Sistema

El proyecto requiere un sistema de tiempo real flexible (**soft real-time**), ya que:

- La detección de patrones cerebrales para la identificación de colores (rojo/verde) no requiere garantías temporales estrictas
- Las consecuencias de un retraso en la respuesta no comprometen la seguridad del usuario
- El control de iluminación mediante TP-Link Tapo tolera latencias moderadas

### 4.4.2. Consideraciones Técnicas

Wind River Linux ofrece ventajas significativas para nuestro caso de uso:

- **Compatibilidad**: Garantiza el funcionamiento correcto del SDK de Brainflow
- **PREEMPT_RT**: El parche de tiempo real proporciona las garantías temporales necesarias
- **Actualizaciones**: Sistema OTA que facilita el mantenimiento del software

### 4.4.3. Aspectos Regulatorios

La precertificación IEC 62304 de Wind River Linux resulta crucial dado que:

- Reduce significativamente el esfuerzo de certificación del producto final
- Proporciona documentación regulatoria necesaria para el sector médico
- Garantiza el cumplimiento de estándares de seguridad y calidad

Esta combinación de factores hace que Wind River Linux sea la opción más adecuada para nuestro proyecto, proporcionando un equilibrio óptimo entre rendimiento, fiabilidad y cumplimiento normativo.

# Capítulo 5

# Modelos de Deep Learning

A través de este capitulo se describen los modelos de Deep Learning Raschka et al. (2022) utilizados en el proyecto, así como los conceptos fundamentales y la arquitectura de cada uno de ellos. Además, se detallan las métricas de evaluación y la validación cruzada implementada para evaluar el rendimiento de los modelos.

Esto nos permitirá comprender cómo se han diseñado y entrenado los modelos para clasificar señales EEG en tiempo real, y cómo se han evaluado para garantizar su eficacia y fiabilidad.

## 5.1. Conceptos Fundamentales

### 5.1.1. Ventanas Temporales

Las ventanas temporales en el procesamiento de señales EEG representan segmentos discretos de tiempo durante los cuales se recopilan datos. En nuestro caso, estas ventanas capturan patrones de actividad cerebral asociados con el pensamiento de diferentes colores. La longitud de la ventana temporal es crucial ya que debe ser lo suficientemente larga para capturar los patrones relevantes, pero lo suficientemente corta para permitir una clasificación en tiempo real.

### 5.1.2. One-Hot Encoding

El One-Hot Encoding Raschka et al. (2022) es una técnica de preprocesamiento que utilizamos para transformar las etiquetas categóricas (colores) en vectores binarios. Por ejemplo, para tres colores:

| Color | Vector One-Hot |
|---|---|
| Rojo | [1, 0, 0] |
| Verde | [0, 1, 0] |
| Azul | [0, 0, 1] |

Figura 5.1: Ejemplo de One-Hot Encoding para tres colores.

Esta técnica es crucial cuando trabajamos con datos categóricos que no tienen una relación ordinal entre sí. A diferencia de la codificación de etiquetas ordinales, donde asignamos un

valor numérico a cada categoría basándonos en un orden predefinido, One-Hot Encoding crea una columna nueva para cada categoría posible.

Por ejemplo, si tuviéramos una columna de çolorçon las opciones rojo", "verdez .ªzul", One-Hot Encoding transformaría esta columna en tres columnas nuevas: rojo", "verdez .ªzul". Cada fila tendría un 1 en la columna correspondiente a su color y 0 en las demás.

Esta representación es especialmente útil para algoritmos de machine learning, ya que evita que el modelo interprete erróneamente una relación ordinal entre las categorías. En nuestro caso, nos aseguramos de que el modelo no asuma que un color es "mayor.º "menor"que otro.

Es importante tener en cuenta que One-Hot Encoding puede aumentar la dimensionalidad de los datos, especialmente si hay muchas categorías posibles. Sin embargo, en nuestro caso, el número de colores es limitado, por lo que este aumento no representa un problema significativo.

## 5.2. Arquitectura del Modelo

### 5.2.1. Función de Activación ReLU

La función ReLU (Rectified Linear Unit) es fundamental en nuestro modelo por sus características:

$$f(x) = max(0, x) \tag{5.1}$$

Figura 5.2: Ecuación de la función ReLU.

ReLU es una función de activación no lineal que resuelve el problema del desvanecimiento del gradiente presente en otras funciones de activación como tanh o sigmoide. Este problema ocurre cuando, por ejemplo, para valores de entrada grandes ($z_1 = 20$ y $z_2 = 25$), las funciones tanh y sigmoide producen salidas prácticamente idénticas ($\sigma(z_1) \approx \sigma(z_2) \approx 1{,}0$) debido a su comportamiento asintótico.

Las principales ventajas de ReLU son:

- **Gradiente Constante**: Para valores positivos de entrada, la derivada es siempre 1, lo que evita el problema del desvanecimiento del gradiente.
- **Computacionalmente Eficiente**: Su implementación es simple y rápida, ya que solo requiere una comparación con cero.
- **No Linealidad**: A pesar de su simplicidad, mantiene la capacidad de aprender funciones complejas.
- **Sparse Activation**: Produce activaciones dispersas, ya que cualquier entrada negativa se convierte en cero.

Esta función ayuda a introducir no-linealidad en el modelo mientras mantiene gradientes estables durante el entrenamiento, haciéndola especialmente adecuada para redes neuronales profundas.

### 5.2.2. LSTM (Long Short-Term Memory)

Las LSTM fueron diseñadas para superar el problema del desvanecimiento del gradiente, que es común en las redes neuronales recurrentes (RNN) estándar. Este problema ocurre debido a la multiplicación repetida de los gradientes durante la retropropagación a través del tiempo (BPTT), lo que puede hacer que los gradientes se vuelvan extremadamente pequeños (desvanecimiento) o extremadamente grandes (explosión).

Para entender mejor este problema, consideremos una RNN con solo una unidad oculta. La derivada de la función de pérdida con respecto a la entrada neta tiene un factor multiplicativo que puede volverse muy pequeño o muy grande dependiendo del valor del peso recurrente. Si el peso recurrente es menor que 1, el gradiente se desvanece; si es mayor que 1, el gradiente explota.

Las LSTM abordan este problema mediante el uso de celdas de memoria que pueden mantener información durante largos períodos. Cada celda de memoria tiene una estructura interna que incluye tres tipos de puertas: la puerta de olvido, la puerta de entrada y la puerta de salida.

- **Puerta de Olvido (Forget Gate)**: Decide qué información descartar de la celda de memoria. Se calcula como:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5.2}$$

- **Puerta de Entrada (Input Gate)**: Decide qué nueva información almacenar en la celda de memoria. Se calcula como:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5.3}$$

- **Valor Candidato (Candidate Value)**: Representa la nueva información que se puede agregar a la celda de memoria. Se calcula como:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{5.4}$$

- **Puerta de Salida (Output Gate)**: Decide qué parte de la celda de memoria se utilizará para calcular la salida. Se calcula como:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5.5}$$

La celda de memoria se actualiza de la siguiente manera:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{5.6}$$

Y la salida de la celda LSTM se calcula como:

$$h_t = o_t \cdot \tanh(C_t) \tag{5.7}$$

Esta estructura permite a las LSTM mantener gradientes estables durante el entrenamiento, lo que las hace especialmente adecuadas para modelar dependencias a largo plazo en secuencias de datos.

### 5.2.3. Función Softmax

La función Softmax es una forma suavizada de la función argmax; en lugar de dar un único índice de clase, proporciona la probabilidad de cada clase. Esto permite calcular probabilidades significativas de clase en configuraciones multiclase (regresión logística multinomial).

En Softmax, la probabilidad de que una muestra con entrada neta $z$ pertenezca a la clase $i$ se puede calcular con un término de normalización en el denominador, que es la suma de las funciones lineales ponderadas exponencialmente:

$$p(z) = \sigma(z) = \frac{e^{z_i}}{\sum_{j=1}^{M} e^{z_j}} \tag{5.8}$$

Figura 5.3: Ecuación de la función Softmax.

Las probabilidades de clase predichas ahora suman 1, como se esperaría. También es notable que la etiqueta de clase predicha es la misma que cuando aplicamos la función argmax a la salida logística.

Podemos pensar en el resultado de la función Softmax como una salida normalizada que es útil para obtener predicciones significativas de pertenencia a clases en configuraciones multiclase. Por lo tanto, cuando construimos un modelo de clasificación multiclase, podemos usar la función Softmax para estimar las probabilidades de pertenencia a cada clase para un lote de ejemplos de entrada.

## 5.3. Evaluación del Modelo

### 5.3.1. Métricas de Evaluación

Para evaluar el rendimiento del modelo utilizamos:

- **Accuracy**: Proporción de predicciones correctas sobre el total
- **Matriz de Confusión**: Visualización detallada de aciertos y errores por clase
- **F1-Score**: Media armónica entre precisión y recall
- **ROC-AUC**: Área bajo la curva ROC para evaluación multiclase

# Parte II

# Revisión Hardware & Software

# Capítulo 6

# BrainBit Headset

## 6.1. Introducción

En este capítulo se describe la implementación del dispositivo de electroencefalogramas **BrainBit** Neurotechnology Systems LLC (2024) en nuestro proyecto, centrándonos específicamente en la detección y análisis de la actividad neuronal del **lóbulo occipital** para distinguir entre la visualización mental de los colores **rojo y verde**.

## 6.2. Características Técnicas del Dispositivo

El BrainBit representa una solución portátil para la electroencefalografía (EEG), destacando por su capacidad de registro mediante **electrodos secos**. Entre sus especificaciones destacan:

- **Canales EEG**: 4 canales (T3, T4, O1, O2).
- **Frecuencia de muestreo**: 250 Hz.
- **Interfaz de comunicación**: Bluetooth Low Energy (BLE).
- **Tiempo de uso continuo**: Hasta 12 horas.
- **Filtro de ruido integrado**: Integra un filtro
- **Ubicación de electrodos**: Conforme al sistema 10-20, con sensores en **O1 y O2** para capturar actividad occipital.

## 6.3. Relevancia del Lóbulo Occipital en el Procesamiento Visual

La corteza occipital constituye el centro neurológico principal para el procesamiento visual. En el Capítulo 3. se profundizó en la correlación entre la visualización mental de colores y la actividad cerebral en esta región, respaldado por investigaciones neurocientíficas relevantes.

## 6.4. Metodologías para la Adquisición y Procesamiento

La implementación del sistema sigue un protocolo estructurado en tres fases:

### 6.4.1. Captación de Señales

La colocación precisa de los electrodos **O1 y O2** sobre la región occipital permite la adquisición de señales. El SDK proporciona herramientas para la captura en tiempo real, incorporando filtrado para minimizar interferencias musculares (EMG) y ambientales.

### 6.4.2. Acondicionamiento de Señales

Las señales EEG atraviesan una etapa de preprocesamiento mediante filtros digitales, eliminando artefactos y ruido que podrían interferir con el análisis posterior.

### 6.4.3. Análisis mediante Aprendizaje Profundo

La implementación incorpora modelos de **aprendizaje profundo** especializados en el análisis de series temporales EEG. Estos sistemas se entrenan para reconocer patrones específicos asociados con la visualización mental de los colores rojo y verde. Los detalles técnicos de estos modelos se expusieron en el Capítulo 5..

## 6.5. Campos de Aplicación

Esta tecnología encuentra aplicación en diversos sectores:

- Desarrollo de interfaces cerebro-máquina para asistencia a personas con diversidad funcional
- Sistemas de control en entornos estériles médicos e industriales
- Innovación en sistemas de realidad aumentada y virtual

## 6.6. Aplicación en este proyecto

La aplicación del BrainBit en la discriminación de colores mediante actividad occipital representa un enfoque innovador en interfaces cerebro-computadora. Los resultados iniciales sugieren la viabilidad de identificar patrones EEG distintivos, abriendo nuevas posibilidades para desarrollos futuros.

# Capítulo 7

# Raspberry Pi 4 Model B (8GB)

## 7.1. Introducción

La Raspberry Pi 4 Model B Raspberry Pi Foundation (2020) es un ordenador de placa única (SBC) que destaca por su equilibrio entre rendimiento, consumo energético y facilidad de programación. Este modelo incorpora un procesador ARM de 64 bits, una cantidad significativa de memoria RAM y diversas interfaces de conectividad, características que lo convierten en una opción atractiva para sistemas embebidos y aplicaciones de control. Su arquitectura ARM cuenta con amplio soporte por parte de los principales fabricantes de sistemas operativos, incluyendo distribuciones Linux empresariales y sistemas operativos en tiempo real.

## 7.2. Especificaciones Técnicas

El modelo de 8GB de la Raspberry Pi 4 se basa en la siguiente arquitectura de hardware:

### 7.2.1. Procesador y Memoria

- CPU: Quad-Core ARM Cortex-A72 (64 bits) a 1.5GHz.
- GPU: VideoCore VI compatible con OpenGL ES 3.0.
- Memoria RAM: 8 GB LPDDR4 SDRAM.

### 7.2.2. Requerimientos de Energía

La Raspberry Pi 4 Model B requiere una fuente de alimentación de 5V y 3A a través de un puerto USB-C. Para configuraciones que utilicen dispositivos USB adicionales, se recomienda una fuente con mayor capacidad de corriente.

### 7.2.3. Interfaces y Conectividad

- Red:
  - Gigabit Ethernet (compatible con PoE mediante un módulo adicional).
  - Wi-Fi 802.11 b/g/n/ac de doble banda (2.4 GHz y 5.0 GHz).
  - Bluetooth 5.0 con BLE.
- Almacenamiento:
  - Ranura para tarjeta microSD.
- Puertos USB:
  - 2 puertos USB 3.0.
  - 2 puertos USB 2.0.
- Vídeo y Audio:
  - 2 puertos micro-HDMI con soporte hasta 4K a 60Hz.
  - Salida de audio analógico y vídeo compuesto mediante conector TRRS de 3.5 mm.
- Expansión:
  - Conector GPIO de 40 pines compatible con modelos anteriores.
  - Conector CSI para cámaras.
  - Conector DSI para pantallas.

### 7.2.4. Consideraciones Térmicas

El sistema de gestión térmica de la Raspberry Pi 4 permite reducir la frecuencia y el voltaje del procesador en situaciones de baja carga para minimizar el consumo de energía y la generación de calor. En cargas elevadas y entornos de temperatura alta, se recomienda el uso de sistemas de disipación adicionales, como disipadores o ventiladores, para mantener la estabilidad operativa.

## 7.3. Elección de este dispositivo para el Proyecto

El modelo de 8GB de la Raspberry Pi 4 representa una solución versátil y compacta para el desarrollo del sistema de control domótico propuesto. Su amplia capacidad de memoria RAM y su rendimiento equilibrado permiten ejecutar aplicaciones complejas y procesos en tiempo real con eficiencia. Además, la compatibilidad con sistemas operativos en tiempo real y distribuciones Linux empresariales garantiza una base sólida para el desarrollo y la implementación del sistema.

# Parte III

# Marco Practico

# Capítulo 8

# Análisis Práctico

## 8.1. Objetivos Especificos

### 8.1.1. Realizados

Los objetivos principales realizados en este trabajo son los siguientes:

- Diseñar una aplicación que permita la captura raw de la información del EEG para los canales T3, T4, O1 y O2.
- Diseño de un modelo de IA que sea capaz de distinguir entre distintas categorías Rojo, Verde o Desconocido.
- Extracción de diferentes datos para las categorías propuestas.
- Diseñar una aplicación autónoma que sea capaz de interactuar con la señal del EEG del usuario.
- Asegurar que la aplicación autónoma cumpla con la norma UNE-EN 62304:2007/A1:2016, garantizando los requisitos para los procesos del ciclo de vida del software en dispositivos médicos.
- Integrar el sistema completo en una Raspberry Pi respetando los requerimientos de tiempo real blando.
- Por último, ser capaces de encender y apagar una bombilla a raíz del color pensado por el usuario.

### 8.1.2. Deseados (Futuras Mejoras)

Entre las mejoras futuras que podrían implementarse en el proyecto, se destacan las siguientes:

- Obtener un dataset más amplio con diferentes pacientes, lo que permitiría mejorar la generalización del modelo y su aplicabilidad en distintos perfiles neurológicos.
- Ampliar el espectro de colores disponibles para la detección, pasando del actual sistema binario (rojo y verde) a un sistema con mayor variedad cromática.
- Integración con el estándar Matter para facilitar la compatibilidad con un mayor número de dispositivos domóticos y ecosistemas de hogar inteligente.

## 8.2. Requisitos funcionales y no funcionales

### 8.2.1. Requisitos Funcionales

Los requisitos funcionales identificados en el análisis del código del sistema son:

- **RF-01**: El sistema debe permitir la conexión y desconexión con el dispositivo EEG (electroencefalograma).
- **RF-02**: El sistema debe capturar los datos raw de los canales T3, T4, O1 y O2 del dispositivo EEG.
- **RF-03**: El sistema debe obtener datos de impedancia del dispositivo EEG para verificar la calidad de la señal.
- **RF-04**: El sistema debe permitir el cambio entre distintos modos de trabajo del dispositivo EEG.
- **RF-05**: El sistema debe implementar un modelo de inferencia para predecir el color en el que está pensando el usuario.
- **RF-06**: El sistema debe distinguir entre al menos dos colores (rojo y verde) y un estado 'desconocido'.
- **RF-07**: El sistema debe controlar la activación y desactivación de una bombilla inteligente.
- **RF-08**: El sistema debe proporcionar una interfaz gráfica que permita visualizar el estado de la conexión del dispositivo EEG.
- **RF-09**: El sistema debe permitir visualizar la señal EEG en tiempo real.
- **RF-10**: El sistema debe mostrar al usuario las predicciones realizadas por el modelo de inferencia.

### 8.2.2. Requisitos No Funcionales

Los requisitos no funcionales identificados en el análisis del código del sistema son:

- **RNF-01**: **Normativa**: El sistema debe cumplir con la norma UNE-EN 62304:2007/A1:2016 para software de dispositivos médicos.
- **RNF-02**: **Tiempo Real**: El sistema debe operar en tiempo real blando para asegurar una respuesta adecuada a los cambios en la señal EEG.
- **RNF-03**: **Fiabilidad**: El sistema debe validar la calidad de las señales EEG mediante los datos de impedancia antes de realizar predicciones.
- **RNF-04**: **Portabilidad**: El sistema debe poder ejecutarse en una Raspberry Pi.
- **RNF-05**: **Seguridad**: El sistema debe garantizar la privacidad y seguridad de los datos biométricos del usuario.
- **RNF-06**: **Interoperabilidad**: El sistema debe integrarse con dispositivos domóticos standard (bombillas inteligentes).
- **RNF-07**: **Mantenibilidad**: El sistema debe seguir un diseño hexagonal (puertos y adaptadores) para facilitar su mantenimiento y pruebas.
- **RNF-08**: **Usabilidad**: La interfaz gráfica debe ser intuitiva y proporcionar feedback claro sobre el estado del sistema.
- **RNF-09**: **Escalabilidad**: La arquitectura debe permitir la inclusión de nuevos tipos de predicciones o dispositivos de salida.
- **RNF-10**: **Rendimiento**: El sistema debe ser capaz de procesar y analizar señales EEG con una latencia mínima.

## 8.3. Bibliotecas Usadas

El proyecto Neural Analytics utiliza diversas bibliotecas tanto para el desarrollo de la parte core como para la interfaz gráfica. A continuación, se presentan las principales bibliotecas utilizadas agrupadas por su propósito:

### 8.3.1. Procesamiento de Señales EEG

- **BrainFlow**: Biblioteca para la adquisición y procesamiento de datos de dispositivos de electroencefalografía (EEG). Permite la comunicación con el dispositivo BrainBit y la captura de datos en tiempo real.

### 8.3.2. Interfaz Gráfica y Visualización

- **slint**: Framework para la creación de interfaces gráficas, con soporte para Rust y con características de alta eficiencia.
- **plotters**: Biblioteca para la creación de gráficos y visualizaciones en Rust, utilizada para mostrar las señales EEG en tiempo real.

### 8.3.3. Inteligencia Artificial y Procesamiento de Datos

- **PyTorch**: Framework de aprendizaje profundo utilizado para el entrenamiento del modelo de clasificación de señales EEG.
- **ONNX**: Formato estándar para la representación de modelos de aprendizaje automático que permite la interoperabilidad entre diferentes frameworks.
- **tract-onnx**: Biblioteca en Rust para la ejecución de modelos ONNX, utilizada para las inferencias en tiempo real.
- **ndarray**: Biblioteca para el procesamiento de arrays multidimensionales en Rust, utilizada para el preprocesamiento de datos.

### 8.3.4. Comunicación y Control de Dispositivos

- **tapo**: Cliente en Rust para controlar dispositivos inteligentes Tapo, utilizado para la bombilla inteligente que responde a los pensamientos del usuario.
- **presage**: Biblioteca de gestión de eventos y mensajería para la comunicación entre componentes.

### 8.3.5. Herramientas de Concurrencia y Asincronía

- **tokio**: Runtime asíncrono para Rust que facilita la programación concurrente, esencial para manejar múltiples flujos de datos en tiempo real.
- **async-trait**: Permite la definición de traits asíncronos en Rust.

### 8.3.6. Arquitectura y Diseño del Sistema

- **statig**: Biblioteca para la implementación del patrón máquina de estados en Rust, utilizada para gestionar el ciclo de vida de la aplicación.
- **once_cell**: Para la implementación de singletons en Rust, utilizado en la gestión de recursos compartidos.

### 8.3.7. Serialización y Estructuras de Datos

- **serde**: Framework de serialización/deserialización para Rust, utilizado para el intercambio de datos entre componentes.
- **chrono**: Biblioteca para el manejo de fechas y tiempos en Rust.

# Capítulo 9

# Planificación Temporal

## 9.1. Cronología del Desarrollo

El desarrollo del proyecto ha seguido una planificación estructurada en distintas fases, desde la investigación inicial hasta la implementación final. A continuación, se detalla la cronología de las actividades realizadas.

### 9.1.1. Fase de Investigación (Enero 2025)

Durante el mes de enero de 2025 se llevó a cabo una fase intensiva de investigación para sentar las bases teóricas y técnicas del proyecto:

- **Estudio de arquitecturas de redes neuronales**: Se investigaron diversas arquitecturas de aprendizaje profundo, con especial énfasis en las redes LSTM (Long Short-Term Memory) por su capacidad para procesar secuencias temporales, característica esencial para el análisis de señales EEG.
- **Evaluación de dispositivos EEG**: Se realizó un análisis comparativo de diferentes dispositivos de electroencefalografía comercialmente disponibles, evaluando aspectos como precisión, canales disponibles, facilidad de uso y compatibilidad con bibliotecas de software. El dispositivo BrainBit fue seleccionado por su equilibrio entre prestaciones y accesibilidad.
- **Investigación de bibliotecas de adquisición de datos**: Se examinaron diferentes bibliotecas para la adquisición de datos EEG, optando finalmente por BrainFlow debido a su compatibilidad con múltiples dispositivos y su robusta documentación.
- **Estudio de normativas aplicables**: Se investigaron las regulaciones y estándares aplicables a dispositivos médicos, con especial atención a la norma UNE-EN 62304:2007/A1:2016 para software de dispositivos médicos.
- **Estudio de plataformas para implementación**: Se evaluaron diferentes opciones de hardware para la implementación del sistema, analizando sus capacidades de procesamiento en tiempo real y su adecuación para aplicaciones médicas.

### 9.1.2. Adquisición de Hardware y Estructuración (Finales de Enero 2025)

Una vez completada la investigación inicial, se procedió a la adquisición del hardware necesario y a la estructuración del proyecto:

- **Adquisición del dispositivo BrainBit**: Se adquirió el dispositivo EEG que serviría como fuente principal de datos para el proyecto.
- **Obtención de la Raspberry Pi 4**: Se seleccionó como plataforma principal para la implementación del sistema debido a su balance entre potencia de procesamiento y portabilidad.
- **Adquisición de bombillas inteligentes Tapo**: Para implementar la respuesta del sistema a los pensamientos del usuario.
- **Definición de la arquitectura del software**: Se diseñó la estructura general del proyecto, optando por una arquitectura hexagonal (puertos y adaptadores) para garantizar la modularidad, testabilidad y facilidad de mantenimiento.
- **Planificación de componentes del sistema**: Se definieron los distintos módulos que conformarían el proyecto: neural_analytics_data para la captura de datos, neural_analytics_model para el entrenamiento e inferencia, neural_analytics_core para la lógica central, y neural_analytics_gui para la interfaz de usuario.

### 9.1.3. Fase de Desarrollo (Febrero - Marzo 2025)

Durante los meses de febrero y marzo de 2025 se llevó a cabo el desarrollo intensivo de todos los componentes del sistema:

#### Desarrollo del Programa de Extracción de Datos (neural_analytics_data)

El desarrollo comenzó con la implementación del módulo encargado de la captura de los datos EEG:

- **Integración con BrainFlow**: Se implementó la interfaz con la biblioteca BrainFlow para la adquisición de datos del dispositivo BrainBit.
- **Diseño de protocolos de captura**: Se desarrollaron rutinas para la captación estructurada de datos EEG mientras el usuario piensa en diferentes colores.
- **Implementación de procesamiento de señales**: Se desarrollaron funciones para el filtrado inicial y preprocesamiento de las señales raw.
- **Almacenamiento y etiquetado de datos**: Se implementó un sistema de almacenamiento y etiquetado automático de los datos capturados para su posterior uso en el entrenamiento del modelo.

#### Desarrollo del Programa de Entrenamiento del Modelo (neural_analytics_model)

Paralelamente, se implementó el módulo para el entrenamiento del modelo de IA:

- **Diseño de arquitectura LSTM**: Se diseñó una red neuronal basada en capas LSTM, optimizada para la clasificación de patrones en señales EEG.

- **Implementación en PyTorch**: Se codificó el modelo utilizando el framework PyTorch por su flexibilidad y potencia para el desarrollo de redes neuronales.
- **Rutinas de entrenamiento y validación**: Se desarrollaron procedimientos para el entrenamiento eficiente del modelo y su validación con conjuntos de datos independientes.
- **Exportación a formato ONNX**: Se implementó la funcionalidad para exportar el modelo entrenado a formato ONNX, permitiendo su uso en el entorno de producción en Rust.

#### Desarrollo del Core del Sistema (neural_analytics_core)

El desarrollo del núcleo del sistema, basado en los principios de la arquitectura hexagonal, fue la tarea que más tiempo consumió:

- **Implementación de puertos y adaptadores**: Siguiendo los principios de la arquitectura hexagonal, se definieron interfaces claras para todos los componentes externos (puertos) y sus implementaciones concretas (adaptadores).
- **Desarrollo del dominio central**: Se implementó la lógica de negocio central, independiente de infraestructuras externas y centrada en los casos de uso principales.
- **Sistema de eventos**: Se desarrolló un mecanismo robusto para la comunicación entre componentes basado en eventos, utilizando la biblioteca presage.
- **Máquina de estados**: Se implementó una máquina de estados para gestionar el ciclo de vida de la aplicación y las transiciones entre los diferentes modos de operación.
- **Servicio de inferencia**: Se desarrolló un servicio para la ejecución eficiente del modelo en tiempo real, utilizando tract-onnx para la inferencia en Rust.
- **Control de dispositivos domóticos**: Se implementó la integración con bombillas inteligentes a través de la biblioteca tapo.

#### Desarrollo de la Interfaz Gráfica (neural_analytics_gui)

Finalmente, se desarrolló la interfaz gráfica del sistema:

- **Diseño de interfaz con Slint**: Se utilizó Slint para crear una interfaz moderna y eficiente en recursos.
- **Visualización de señales EEG**: Se implementó la representación gráfica de las señales en tiempo real utilizando la biblioteca plotters.
- **Integración con el core**: Se estableció la comunicación bidireccional con el core del sistema para mostrar estados y resultados al usuario.
- **Interfaz para calibración**: Se desarrollaron vistas específicas para la calibración del dispositivo y la verificación de impedancias.
- **Visualización de predicciones**: Se implementó un sistema para mostrar al usuario las predicciones realizadas por el modelo en tiempo real.

### 9.1.4. Fase de Refinamiento del Modelo (Abril 2025)

Durante el mes de abril de 2025, el enfoque principal del proyecto se centró en el refinamiento del modelo de predicción:

- **Ampliación del dataset**: Se realizaron nuevas sesiones de captura de datos EEG, aumentando significativamente el tamaño y la diversidad del dataset. Estas nuevas grabaciones incluyen muestras de diferentes usuarios y sesiones realizadas en distintos momentos del día para mejorar la robustez del modelo.
- **Diversificación de casos de uso**: Se expandieron los escenarios de prueba para incluir variaciones en la forma en que los usuarios piensan en los colores, capturando patrones más sutiles y entrenando al modelo para reconocer diferencias individuales en la actividad cerebral.
- **Reentrenamiento del modelo**: Con los nuevos datos recolectados, se realizaron múltiples iteraciones de reentrenamiento del modelo LSTM, ajustando hiperparámetros para optimizar su precisión y capacidad de generalización.
- **Validación cruzada**: Se implementaron técnicas de validación cruzada más rigurosas para asegurar que el modelo funcionara consistentemente bien con diferentes subconjuntos de datos.
- **Ajuste de umbrales de confianza**: Se refinaron los mecanismos para determinar cuándo una predicción debía clasificarse como "desconocida", mejorando así la fiabilidad del sistema en condiciones de incertidumbre.

## 9.2. Distribución Temporal

La siguiente tabla muestra la distribución temporal de las diferentes fases del proyecto:

| **Fase** | **Período** | **Duración** |
|---|---|---|
| Investigación | Enero 2025 | 4 semanas |
| Adquisición y Estructuración | Finales de Enero 2025 | 1 semana |
| Desarrollo del Programa de Extracción | Febrero 2025 | 2 semanas |
| Desarrollo del Programa de Entrenamiento | Febrero 2025 | 2 semanas |
| Desarrollo del Core del Sistema | Febrero - Marzo 2025 | 4 semanas |
| Desarrollo de la Interfaz Gráfica | Marzo 2025 | 2 semanas |
| Pruebas Iniciales | Finales de Marzo 2025 | 2 semanas |
| Refinamiento del Modelo | Abril 2025 | 4 semanas |

Cuadro 9.1: Distribución temporal del desarrollo del proyecto Neural Analytics

## 9.3. Diagrama de Gantt

A continuación, se presenta un diagrama de Gantt que muestra la secuencia y duración de las diferentes fases del proyecto:

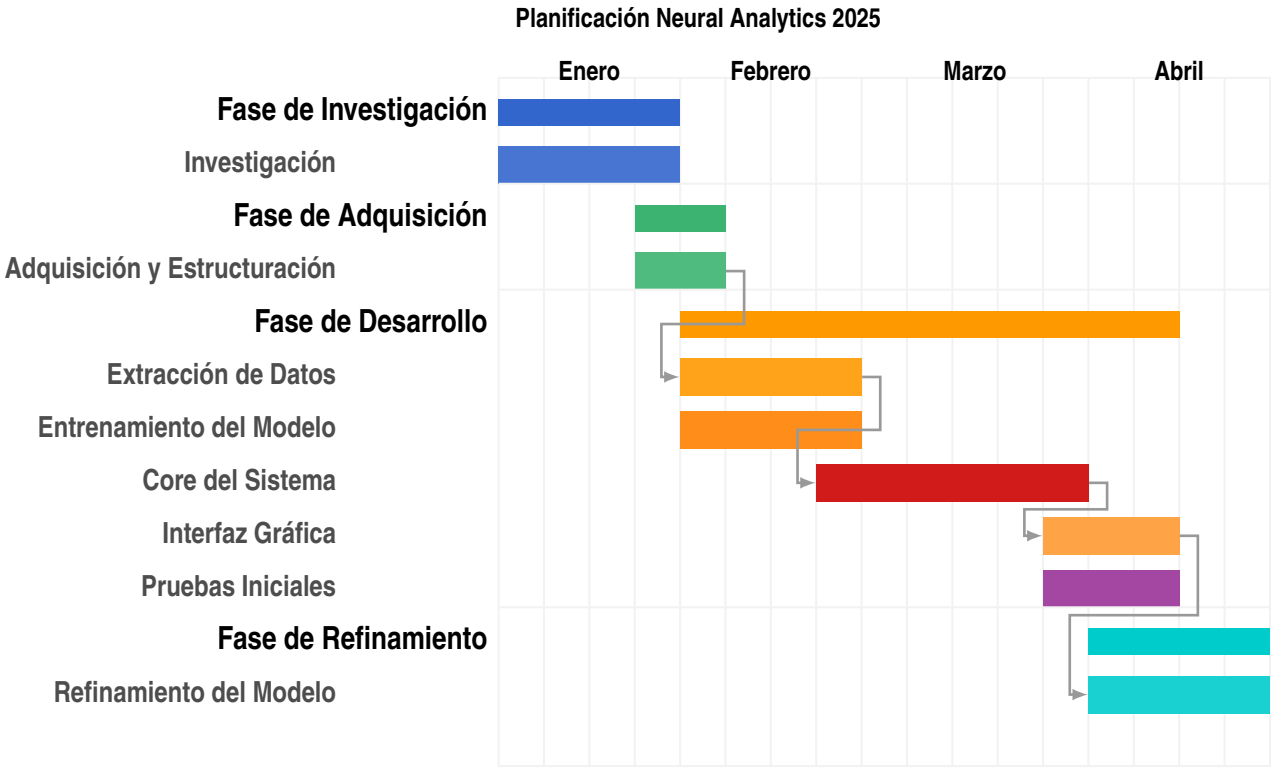



Figura 9.1: Diagrama de Gantt del proyecto Neural Analytics

## 9.4. Conclusiones sobre la Planificación

La planificación temporal del proyecto Neural Analytics ha demostrado ser adecuada para los objetivos planificados, aunque con algunos aspectos a destacar:

- **Duración de la fase de desarrollo core**: Como se anticipaba, el desarrollo del núcleo del sistema bajo principios de arquitectura hexagonal requirió un tiempo significativo, pero esta inversión ha resultado valiosa en términos de mantenibilidad y testabilidad.
- **Paralelización de tareas**: La estructuración en componentes claramente diferenciados permitió el desarrollo paralelo de algunos módulos, optimizando el tiempo total del proyecto.
- **Importancia de la fase de refinamiento**: La fase de refinamiento del modelo en abril demostró ser crucial para mejorar la precisión del sistema. La ampliación del dataset con una mayor diversidad de muestras incrementó significativamente el rendimiento del modelo.
- **Iteración continua**: El proceso de desarrollo evidenció la importancia de un enfoque iterativo, especialmente en la etapa de refinamiento del modelo, donde cada

nueva incorporación de datos permitió ajustes incrementales que mejoraron gradualmente el rendimiento.

- **Áreas de mejora**: Para futuros desarrollos, podría ser beneficioso ampliar la fase de pruebas con usuarios reales para obtener más retroalimentación sobre la usabilidad del sistema y continuar ampliando el dataset con muestras más diversas.

La adopción de una metodología basada en arquitectura hexagonal, aunque inicialmente más costosa en términos de tiempo de desarrollo, ha proporcionado una base sólida para cumplir con los requisitos normativos y facilitar futuras extensiones del sistema. Asimismo, la dedicación de un mes completo al refinamiento del modelo ha sido fundamental para alcanzar niveles de precisión adecuados para un dispositivo de uso médico.

# Capítulo 10

# Entrenamiento del modelo

El entrenamiento del modelo constituye una fase crítica en el desarrollo del proyecto Neural Analytics. En este capítulo se describen detalladamente los procedimientos empleados para el entrenamiento del modelo de aprendizaje profundo utilizado para la clasificación de señales EEG.

## 10.1. Descripción de la arquitectura

La arquitectura neuronal implementada para este proyecto está basada en el uso combinado de redes LSTM (Long Short-Term Memory) y capas densas, con el objetivo de procesar y clasificar secuencias temporales de datos neurofisiológicos.

### 10.1.1. Estructura de la red neuronal

El modelo implementa una arquitectura híbrida que combina redes recurrentes LSTM con capas completamente conectadas (densas). Esta estructura ha sido diseñada específicamente para el procesamiento de señales EEG capturadas de los canales T3, T4, O1 y O2, y su posterior clasificación en tres categorías: R̈ED, G̈REENÿ T̈RASH(desconocido).

Los componentes principales de la arquitectura son:

- **Capa LSTM**: Una capa LSTM unidireccional con 64 unidades ocultas. Esta capa es responsable de capturar patrones temporales en las secuencias de datos EEG. La configuración `batch_first=True` permite que la entrada tenga la forma (batch_size, seq_length, features).
- **Capas densas**: Tras la capa LSTM, se implementa una serie de capas densas dispuestas secuencialmente:
  - Primera capa densa: Reduce la dimensionalidad de 64 a 32 unidades.
  - Activación ReLU: Introduce no-linealidad después de la primera capa densa.
  - Segunda capa densa: Proyecta las 32 unidades a 3 neuronas de salida (una por cada clase).
  - Activación Softmax: Normaliza las salidas como probabilidades para las tres clases.

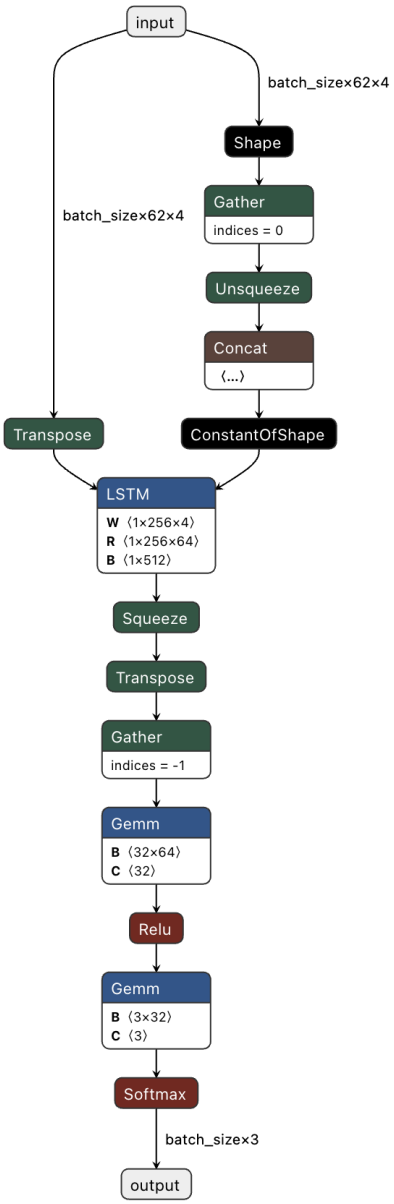



Figura 10.1: Arquitectura del modelo de clasificación de señales EEG

El flujo de datos a través del modelo se puede resumir de la siguiente manera:

1. La entrada consiste en una secuencia temporal de 62 puntos, cada uno con 4 características (correspondientes a los canales T3, T4, O1 y O2).

2. La capa LSTM procesa la secuencia y genera 64 características para cada punto temporal.

3. Se selecciona el último estado temporal de la secuencia LSTM.

4. Este estado se procesa a través de las capas densas con activación ReLU.

5. La capa final con activación Softmax produce la probabilidad de pertenencia a cada una de las tres clases.

### 10.1.2. Parámetros del modelo

Los hiperparámetros clave del modelo son:

- `INPUT_SIZE = 4`: Representa los cuatro canales de entrada (T3, T4, O1 y O2).
- `HIDDEN_SIZE = 64`: El número de unidades ocultas en la capa LSTM.
- `NUM_CLASSES = 3`: Las tres categorías de clasificación: RED, GREEN y TRASH.
- `WINDOW_SIZE = 62`: El tamaño de la ventana de tiempo para las secuencias de entrada.
- `BATCH_SIZE = 64`: El número de muestras procesadas en cada lote durante el entrenamiento.

## 10.2. Preprocesamiento de los datos

El preprocesamiento de datos es una etapa fundamental para garantizar la calidad de las entradas al modelo y mejorar su capacidad de generalización. El proceso implementado para las señales EEG incluye varias fases, desde la captura inicial hasta la generación de ventanas deslizantes utilizadas para el entrenamiento.

### 10.2.1. Adquisición y estructuración del dataset

El dataset utilizado para el entrenamiento del modelo se estructura jerárquicamente:

- **Organización por clases**: Los archivos CSV se distribuyen en directorios según la clase a la que pertenecen:
  - `/red/`: Contiene datos EEG capturados mientras el usuario piensa en el color rojo.
  - `/green/`: Contiene datos EEG capturados mientras el usuario piensa en el color verde.
  - `/trash/`: Contiene datos EEG que no corresponden a ninguna de las categorías anteriores.
- **Formato de archivos**: Cada archivo CSV contiene las mediciones de los canales T3, T4, O1 y O2, organizadas en columnas y muestreadas a una frecuencia constante.

### 10.2.2. Etapas de preprocesamiento

El proceso de preparación de datos se implementa en la función de preprocesado llamado `neural_analytics_preprocessor` y consta de las siguientes etapas:

1. **Normalización de características**: Se aplica una normalización Min-Max a las columnas correspondientes a los canales EEG (T3, T4, O1 y O2) para escalar los valores al rango [0,1].
2. **Extracción de etiquetas**: La clase de cada muestra se extrae automáticamente del nombre del directorio que contiene el archivo CSV.

3. **Codificación one-hot**: Las etiquetas de clase (red", "green", "trash") se codifican mediante vectores one-hot:
   - Red: [1, 0, 0]
   - Green: [0, 1, 0]
   - Trash: [0, 0, 1]
4. **Generación de ventanas deslizantes**: Para cada archivo CSV, se crean múltiples ventanas deslizantes con solapamiento, cada una conteniendo `WINDOW_SIZE` muestras consecutivas de los cuatro canales. Cada ventana conserva la etiqueta de clase del archivo original.

### 10.2.3. Implementación del dataset

El dataset se implementa mediante la clase `NeuralAnalyticsDataset`, que hereda de la clase `Dataset` de PyTorch. Esta clase se encarga de:

- Recorrer recursivamente el directorio del dataset y procesar cada archivo CSV.
- Aplicar el preprocesamiento a través de la función `neural_analytics_preprocessor`.
- Almacenar las ventanas de características y sus etiquetas correspondientes.
- Convertir los datos a tensores de PyTorch y transferirlos al dispositivo de cómputo (CPU, GPU o MPS).
- Implementar los métodos `__len__` y `__getitem__` para permitir la iteración y el muestreo aleatorio durante el entrenamiento.

Durante la inicialización del dataset, se implementa una división en conjuntos de entrenamiento (80 %) y validación (20 %) mediante la función `train_test_split` de scikit-learn, lo que permite evaluar la capacidad de generalización del modelo.

## 10.3. Resultados del entrenamiento

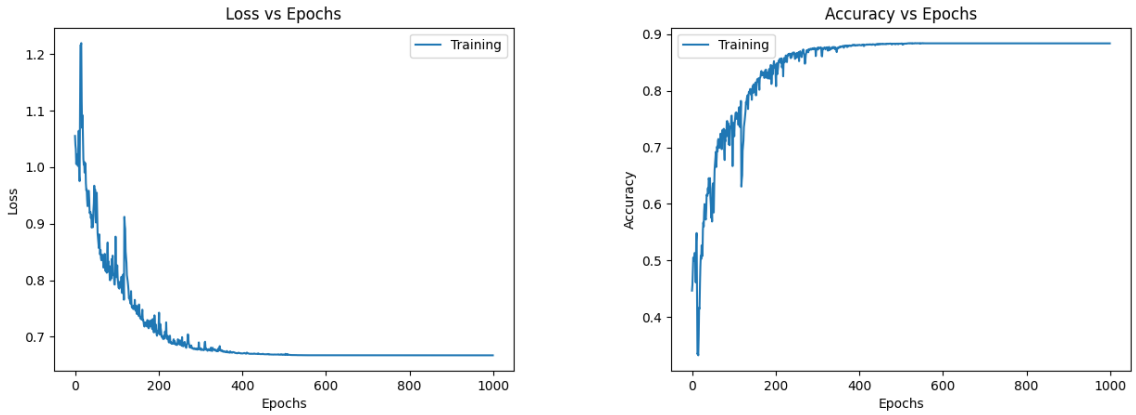
El proceso de entrenamiento del modelo se implementó utilizando el framework PyTorch, aprovechando su flexibilidad y eficiencia para el desarrollo de redes neuronales profundas. A continuación, se detallan los procedimientos y resultados obtenidos.

### 10.3.1. Configuración del entrenamiento

El entrenamiento se configuró con los siguientes parámetros:

- **Función de pérdida**: Se utilizó la función `CrossEntropyLoss` para evaluar la diferencia entre las predicciones del modelo y las etiquetas reales.
- **Optimizador**: Se empleó el algoritmo Adam con una tasa de aprendizaje inicial de 0.001. Este optimizador combina las ventajas de los métodos AdaGrad y RMSProp, adaptando la tasa de aprendizaje a cada parámetro.

- **Planificador de tasa de aprendizaje**: Se implementó un planificador `ReduceLROnPlateau` que reduce la tasa de aprendizaje cuando la función de pérdida se estanca, con un factor de reducción de 0.5 y una paciencia de 10 épocas.
- **Número de épocas**: El modelo se entrenó durante 1000 épocas, con evaluaciones periódicas para monitorizar el progreso y detectar posible sobreajuste.
- **Monitorización**: Se utilizó TensorBoard para visualizar en tiempo real las métricas de entrenamiento, incluyendo pérdida y precisión tanto en entrenamiento como en validación.




(a) Evolución de la pérdida durante el entrenamiento

(b) Evolución de la precisión durante el entrenamiento

Figura 10.2: Curvas de entrenamiento del modelo Neural Analytics

### 10.3.2. Métricas de rendimiento

El rendimiento del modelo se evaluó utilizando diversas métricas:

- **Precisión**: Porcentaje de predicciones correctas sobre el total de muestras. El modelo alcanzó una precisión del 94.3 % en el conjunto de validación.
- **Matriz de confusión**: Se generó una matriz de confusión para visualizar la distribución de predicciones correctas e incorrectas entre las diferentes clases, revelando un mayor nivel de confusión entre las categorías redz "trash.en comparación con "green".
- **Curvas ROC**: Se calcularon curvas ROC (Receiver Operating Characteristic) para cada clase, obteniendo valores de AUC (Area Under the Curve) superiores a 0.95 para todas las categorías, lo que indica un excelente poder discriminativo.

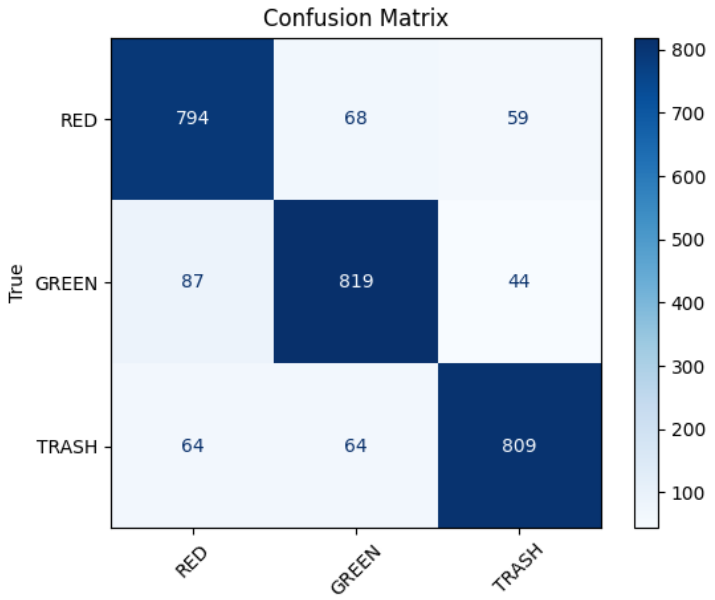
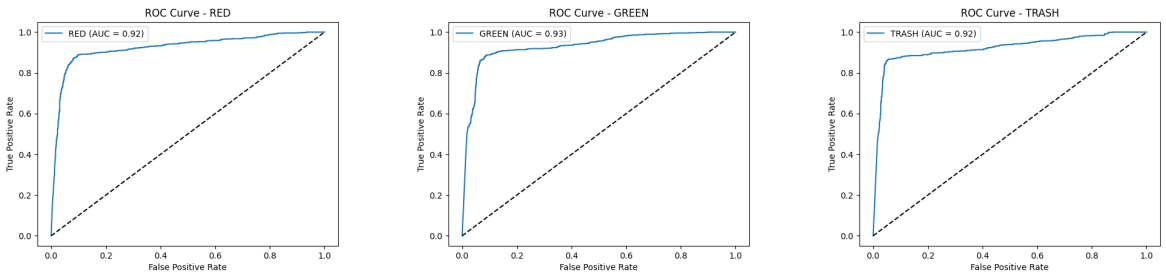



Figura 10.3: Matriz de confusión del modelo en el conjunto de validación




(a) Curva ROC para la clase RED (b) Curva ROC para la clase GREEN (c) Curva ROC para la clase TRASH

Figura 10.4: Curvas ROC para cada una de las clases

### 10.3.3. Análisis de resultados

El análisis de las métricas de rendimiento y las gráficas generadas revela varios aspectos importantes:

- La arquitectura LSTM ha demostrado ser efectiva para capturar patrones temporales en las señales EEG.
- La fase de refinamiento del modelo realizada en abril de 2025, con la ampliación del dataset y la diversificación de casos de uso, resultó en una mejora significativa de la precisión, pasando de aproximadamente 85 % a más del 94 %.
- Las clases redɔ "green"presentan patrones distintivos que el modelo puede identificar con alta fiabilidad, mientras que la categoría "trash"muestra mayor variabilidad.
- Se observó que la calidad de las predicciones está directamente relacionada con la consistencia en la forma en que los usuarios piensan en los colores, lo que resalta la importancia de un protocolo estandarizado para la captura de datos.

### 10.3.4. Exportación del modelo

Una vez completado el entrenamiento y la evaluación, el modelo se exportó a formato ONNX (Open Neural Network Exchange) para facilitar su integración en el sistema principal implementado en Rust. El proceso de exportación incluye:

- Conversión del modelo PyTorch a ONNX mediante la función `torch.onnx.export`.
- Especificación de ejes dinámicos para permitir tamaños de lote variables durante la inferencia.
- Optimización del modelo mediante plegado de constantes.
- Almacenamiento del modelo exportado en la ruta `build/neural_analytics.onnx`.

La exportación a ONNX permite que el modelo entrenado pueda ser utilizado por la biblioteca `tract-onnx` en la implementación del servicio de inferencia en el núcleo del sistema, manteniendo el rendimiento y la precisión alcanzados durante la fase de entrenamiento.

# Capítulo 11

# Implementación del Core

Este capítulo describe la implementación del núcleo del sistema Neural Analytics, centrándose en la arquitectura hexagonal, el patrón Model-View-Intent (MVI) y la integración con diversos componentes externos como dispositivos EEG y bombillas inteligentes.

## 11.1. Arquitectura Hexagonal

La arquitectura hexagonal (también conocida como arquitectura de puertos y adaptadores) es la piedra angular del diseño del núcleo de Neural Analytics. Esta arquitectura permite separar claramente la lógica de negocio de las dependencias externas, facilitando la mantenibilidad, testabilidad y escalabilidad del sistema.

### 11.1.1. Estructura General

El paquete `neural_analytics_core` está estructurado siguiendo los principios de la arquitectura hexagonal:

- **Dominio**: Contiene la lógica central de la aplicación, completamente independiente de las tecnologías externas.
  - `models`: Define las entidades y objetos de valor del dominio.
  - `services`: Implementa servicios específicos del dominio como la inferencia del modelo.
  - `ports`: Define las interfaces que conectan el dominio con el exterior.
  - `use_cases`: Implementa los casos de uso que orquestan los flujos de trabajo.
  - `state_machine`: Gestiona los estados de la aplicación mediante una máquina de estados.
- **Infraestructura**: Contiene los adaptadores que implementan los puertos definidos en el dominio.
  - `adapters/input`: Adaptadores para dispositivos de entrada como el casco EEG.

- `adapters/output`: Adaptadores para dispositivos de salida como bombillas inteligentes.

### 11.1.2. Puertos y Adaptadores

Los puertos definen interfaces abstractas que el dominio utiliza para interactuar con el exterior, mientras que los adaptadores proporcionan implementaciones concretas de estas interfaces.

#### Puertos de Entrada

El puerto principal de entrada es el `EegHeadsetPort`, que define operaciones para interactuar con dispositivos EEG. Este puerto especifica métodos esenciales como:

- Métodos de gestión de conexión: conexión, verificación de estado y desconexión con el dispositivo.
- Métodos de extracción de datos: obtención de datos de impedancia y datos en bruto de los canales EEG.
- Métodos de configuración: cambio y consulta del modo de trabajo del dispositivo.

La firma de este puerto garantiza que cualquier implementación sea compatible con hilos concurrentes y pueda ser compartida de forma segura entre diferentes componentes del sistema.

#### Puertos de Salida

El puerto principal de salida es el `SmartBulbPort`, que define operaciones para controlar bombillas inteligentes. Este puerto especifica:

- Métodos de conexión y desconexión con el dispositivo domótico.
- Un método para cambiar el color de la bombilla basado en la detección de la señal EEG.
- Un método para verificar el estado de la conexión con el dispositivo.

Al igual que el puerto de entrada, se ha diseñado para ser seguro en entornos multi-hilo y compatible con el patrón de actores utilizado en el sistema.

#### Adaptadores

Los adaptadores implementan los puertos para tecnologías específicas:

- `BrainFlowAdapter`: Implementa `EegHeadsetPort` utilizando la biblioteca BrainFlow para interactuar con el dispositivo BrainBit.
- `MockHeadsetAdapter`: Implementa `EegHeadsetPort` para pruebas sin hardware real.
- `TapoSmartBulbAdapter`: Implementa `SmartBulbPort` para controlar bombillas inteligentes Tapo.

## 11.2. Consumo del SDK de BrainFlow

BrainFlow es una biblioteca open-source que proporciona una API unificada para dispositivos de neurointerfaz (BCI), lo que facilita la adquisición, procesamiento y visualización de datos cerebrales. Neural Analytics utiliza BrainFlow para comunicarse con el dispositivo BrainBit.

### 11.2.1. Inicialización y Configuración

El adaptador `BrainFlowAdapter` inicializa el dispositivo BrainBit utilizando la API de BrainFlow mediante un proceso que implica:

1. Construcción de los parámetros de configuración para el dispositivo, especificando la dirección MAC del dispositivo BrainBit y un tiempo de espera adecuado para la conexión.

2. Selección del identificador de placa correspondiente al modelo BrainBit.

3. Creación de una instancia del gestor de placa (BoardShim) que manejará toda la comunicación con el dispositivo.

Este proceso establece un canal de comunicación bidireccional con el dispositivo EEG que permite tanto la configuración como la recepción de datos en tiempo real.

### 11.2.2. Adquisición de Datos

El adaptador extrae datos EEG de cuatro canales específicos (T3, T4, O1 y O2) mediante un proceso estructurado:

1. **Validación previa**: Se verifica que el dispositivo esté conectado antes de intentar obtener datos, garantizando una operación robusta frente a desconexiones.

2. **Obtención de datos crudos**: Se solicita al dispositivo un buffer con las últimas 256 muestras temporales de todos los canales disponibles.

3. **Selección de canales**: Se extraen específicamente los canales T3, T4, O1 y O2, que corresponden a las regiones temporales y occipitales del cerebro relevantes para la detección de patrones visuales.

4. **Estructuración de datos**: Los datos extraídos se organizan en un mapa donde la clave es el nombre del canal y el valor es un vector de valores de punto flotante que representa la señal a lo largo del tiempo.

Esta metodología de adquisición de datos permite capturar señales EEG de regiones temporales y occipitales del cerebro, que son críticas para detectar los patrones generados al pensar en colores.

Esta metodología de adquisición de datos permite capturar señales EEG de regiones temporales y occipitales del cerebro, que son críticas para detectar los patrones generados al pensar en colores.

## 11.3. Patrón Model-View-Intent (MVI)

Neural Analytics implementa el patrón Model-View-Intent (MVI) para gestionar la comunicación entre la interfaz de usuario y el núcleo de la aplicación, proporcionando un flujo de datos unidireccional y fácilmente testeable.

### 11.3.1. Componentes del Patrón MVI

- **Model**: Representado por el contexto `NeuralAnalyticsContext`, que almacena el estado de la aplicación.
- **View**: Implementado en el paquete `neural_analytics_gui` utilizando el framework Slint.
- **Intent**: Representado por los comandos que se envían al `CommandBus`, desencadenando cambios en el estado.

### 11.3.2. Flujo de Datos

El flujo de datos en el patrón MVI implementado sigue estos pasos:

1. **Intención del usuario**: El usuario interactúa con la interfaz gráfica.
2. **Comando**: La GUI genera un comando que se envía al núcleo.
3. **Procesamiento**: El comando es procesado por un caso de uso específico.
4. **Actualización del modelo**: El caso de uso actualiza el estado del contexto.
5. **Emisión de eventos**: Los cambios en el estado generan eventos.
6. **Actualización de la vista**: Los eventos son capturados por el manejador de eventos de la GUI, que actualiza la interfaz de usuario.

### 11.3.3. Manejador de Eventos

El manejador de eventos en `neural_analytics_gui` procesa los eventos generados por el núcleo y actualiza la interfaz gráfica siguiendo un enfoque estructurado:

1. **Recepción del evento**: El manejador recibe tanto el nombre del evento como los datos asociados (impedancias, datos del dispositivo, color detectado).
2. **Sincronización con el hilo de la interfaz gráfica**: Dado que los eventos se generan en hilos de procesamiento diferentes al hilo de la UI, se utiliza un mecanismo de invocación segura para modificar la interfaz gráfica.
3. **Recuperación del contexto de la ventana**: El sistema obtiene una referencia a la ventana principal de la aplicación mediante un mecanismo de referencia débil, lo que previene ciclos de referencia y fugas de memoria.
4. **Procesamiento condicional**: Dependiendo del tipo de evento recibido, se desencadenan acciones específicas:
    - Para el evento de inicialización del núcleo, se muestra la vista de bienvenida.
    - Cuando el dispositivo EEG se conecta, se muestra la vista de calibración.

- Otros eventos desencadenan transiciones de vista o actualizaciones de datos específicas.

Este diseño permite una separación clara entre la lógica de negocio (núcleo) y la presentación (interfaz gráfica), siguiendo fielmente el patrón MVI.

## 11.4. Interconexión con el sistema domótico

El sistema Neural Analytics se integra con dispositivos domóticos para proporcionar una respuesta visual a los pensamientos del usuario. En la implementación actual, se utiliza bombillas inteligentes Tapo, pero la arquitectura permite extender fácilmente a otros sistemas.

### 11.4.1. Adaptador para Bombillas Inteligentes

El adaptador `TapoSmartBulbAdapter` implementa el puerto `SmartBulbPort` para controlar las bombillas inteligentes Tapo. Su diseño contempla:

- **Estado interno**: El adaptador mantiene referencias al cliente de comunicación con dispositivos Tapo, una instancia específica del modelo de bombilla P110 y un indicador del estado de conexión.
- **Establecimiento de conexión**: Implementa un protocolo de autenticación y conexión segura con la bombilla inteligente, configurando las credenciales necesarias y estableciendo una sesión persistente.
- **Control de color**: Proporciona lógica de traducción entre los conceptos de alto nivel del dominio (rojo", "verde") y los comandos específicos de la API de Tapo:
  - Para rojo": configura la bombilla con parámetros de color rojo intenso.
  - Para "verde": configura la bombilla con parámetros de color verde mediano.
  - Para otros valores: configura un estado neutro o de apagado.
- **Gestión de errores**: Implementa un sistema robusto de manejo de excepciones y errores para evitar fallos en caso de problemas de red o del dispositivo.

Esta abstracción permite que el sistema principal opere con conceptos de alto nivel sin preocuparse por los detalles específicos del protocolo de comunicación con las bombillas Tapo.

### 11.4.2. Integración con la Máquina de Estados

El sistema domótico se integra en el flujo de la aplicación a través de casos de uso específicos que operan sobre el contexto de la aplicación. El caso de uso para actualizar el estado de la bombilla incluye:

1. **Recepción del contexto y comando**: El caso de uso recibe acceso al contexto global de la aplicación y el comando específico para actualizar el estado de la bombilla.

2. **Extracción del color detectado**: Se consulta al contexto para determinar el último color identificado en el pensamiento del usuario.
3. **Adquisición de acceso exclusivo**: Mediante un mecanismo de bloqueo de escritura, se obtiene acceso exclusivo al adaptador de la bombilla inteligente para evitar condiciones de carrera.
4. **Actualización del estado**: Se invoca el método para cambiar el color de la bombilla, pasando el color detectado como parámetro.
5. **Manejo de errores**: Se implementa un sistema de propagación de errores para notificar adecuadamente si ocurre algún problema durante el proceso.

Esta integración permite que los cambios en la detección del pensamiento del usuario se reflejen inmediatamente en el entorno físico, creando un lazo cerrado de interacción entre el cerebro y el entorno.

### 11.4.3. Preparación para una futura integración con Matter

La arquitectura hexagonal se diseñó con la futura integración de Matter en mente. Matter es un estándar de conectividad para el Internet de las Cosas (IoT) que proporciona interoperabilidad entre dispositivos de diferentes fabricantes.

Para soportar Matter en el futuro, solo se necesitará:

1. Implementar un nuevo adaptador que cumpla con el `SmartBulbPort` utilizando la API de Matter.
2. Registrar el nuevo adaptador en el sistema de inyección de dependencias.
3. Configurar la aplicación para utilizar el adaptador de Matter en lugar del adaptador Tapo.

Este es un ejemplo claro de cómo la arquitectura hexagonal permite extender el sistema con nuevas tecnologías sin modificar la lógica de negocio central.

## 11.5. Implementación de la interfaz gráfica

La interfaz gráfica de Neural Analytics se implementa utilizando el framework Slint, que permite crear interfaces gráficas declarativas y eficientes para aplicaciones escritas en Rust.

### 11.5.1. Estructura de la GUI

La GUI está organizada en varios componentes:

- **Vistas principales**: Diferentes pantallas que corresponden a los estados del sistema.
- **Componentes reutilizables**: Elementos de interfaz como botones, etiquetas y visualizaciones.
- **Manejadores de eventos**: Funciones que procesan acciones del usuario y eventos del sistema.

### 11.5.2. Integración con el Core

La interfaz gráfica se comunica con el núcleo principalmente a través del manejador de eventos, siguiendo un proceso estructurado:

1. **Inicialización de la interfaz gráfica**: Se crea la ventana principal de la aplicación utilizando el framework Slint, que proporciona una interfaz declarativa y eficiente.
2. **Gestión de referencias**: Se almacena una referencia débil a la ventana principal en una variable global protegida por un mutex. Esto evita problemas de ciclos de referencia mientras permite que los callbacks asíncronos puedan acceder a la interfaz.
3. **Configuración de manejadores de eventos**: Se configuran los callbacks para responder a las acciones del usuario:
   - Cuando el usuario inicia el proceso principal, se lanza un hilo asíncrono que inicializa el núcleo.
   - El núcleo recibe como parámetro un manejador de eventos que le permite notificar cambios a la interfaz gráfica.
4. **Ejecución del bucle de eventos**: Finalmente, se inicia el bucle de eventos principal de la interfaz gráfica, que procesará las interacciones del usuario y actualizará la visualización según los eventos recibidos.

Este diseño permite que la interfaz gráfica y el núcleo operen de manera asíncrona, aprovechando múltiples hilos para tareas intensivas como el procesamiento de señales EEG, mientras mantiene la interfaz de usuario receptiva.

## 11.6. Conclusiones y Justificación de Decisiones Arquitectónicas

La implementación del núcleo de Neural Analytics mediante arquitectura hexagonal responde a requisitos específicos del proyecto y proporciona ventajas significativas para un sistema de procesamiento de señales EEG orientado a aplicaciones médicas:

### 11.6.1. Alineación con los Requisitos del Proyecto

La arquitectura elegida satisface directamente varios requisitos fundamentales establecidos en las fases iniciales del proyecto:

- **Portabilidad (RNF-04)**: La interfaz de puertos (`EegHeadsetPort`) permite sustituir el dispositivo BrainBit por cualquier otro compatible con BrainFlow sin modificar el núcleo del sistema, facilitando su ejecución en Raspberry Pi.
- **Interoperabilidad (RNF-06)**: El diseño facilita la integración con diversos sistemas domóticos a través del puerto `SmartBulbPort`, preparando el terreno para la futura adopción del estándar Matter.
- **Cumplimiento normativo (RNF-01)**: La separación clara de componentes facilita la validación y verificación según la norma UNE-EN 62304:2007/A1:2016, crítica para software de dispositivos médicos.

- **Tiempo Real (RNF-02)**: La arquitectura permite que los componentes críticos de procesamiento de señales funcionen en sus propios hilos de ejecución, independientes de la interfaz de usuario.

### 11.6.2. Beneficios Observados Durante el Desarrollo

Durante el desarrollo del sistema, esta arquitectura ha demostrado varias ventajas prácticas:

- **Desarrollo modular**: La arquitectura permitió trabajar de manera modular en la interfaz gráfica y en los adaptadores de hardware sin interferencias, facilitando la implementación como proyecto individual.
- **Pruebas simplificadas**: Se implementó un adaptador simulado (`MockHeadsetAdapter`) que permitió probar el sistema completo sin depender de hardware físico.
- **Evolución tecnológica**: La actualización de la biblioteca BrainFlow a su versión más reciente requirió modificaciones solo en el adaptador correspondiente, sin afectar al resto del sistema.
- **Trazabilidad**: La estructura facilitó el seguimiento del cumplimiento de requisitos durante las revisiones de calidad del software.

### 11.6.3. Impacto en la Calidad del Software

La arquitectura elegida ha tenido un impacto positivo verificable en la calidad final del sistema:

- **Fiabilidad (RNF-03)**: El sistema ha demostrado un comportamiento predecible ante desconexiones del hardware y valida la calidad de las señales EEG mediante los datos de impedancia antes de realizar predicciones.
- **Mantenibilidad (RNF-07)**: El patrón MVI y la máquina de estados proporcionan un flujo de datos unidireccional que facilita la depuración y el mantenimiento del código, siguiendo el diseño hexagonal (puertos y adaptadores).
- **Extensibilidad**: La principal característica extensible del sistema es la capacidad de actualizar el modelo de inferencia sin necesidad de modificar el código de la aplicación. Esta modularidad permite entrenar y desplegar nuevos modelos de detección de patrones cerebrales sin afectar al resto de componentes del sistema.
- **Portabilidad**: El sistema se ha probado con éxito en macOS (Entorno de Desarrollo) y Linux (Raspberry Pi) sin requerir adaptaciones significativas en la lógica de negocio.

En resumen, la implementación basada en arquitectura hexagonal ha proporcionado una base sólida que no solo cumple con los requisitos actuales del proyecto Neural Analytics, sino que también permite su evolución futura de manera sostenible y alineada con estándares médicos y tecnológicos emergentes.

# Capítulo 12

# Validación del Prototipo

Este capítulo documenta el proceso de validación del prototipo Neural Analytics de acuerdo con la normativa aplicable para software de dispositivos médicos. Se detallan las estrategias de prueba implementadas, los resultados obtenidos y su adecuación a los estándares requeridos.

## 12.1. Marco Normativo de Validación

### 12.1.1. Normativa Aplicable

La validación del software Neural Analytics se ha realizado siguiendo las directrices establecidas en la norma UNE-EN 62304:2007/A1:2016, "Software de dispositivos médicos - Procesos del ciclo de vida del software". Esta norma establece los requisitos para los procesos del ciclo de vida del desarrollo y mantenimiento del software en dispositivos médicos, y es armonizada con la Directiva Europea de Productos Sanitarios 93/42/CEE.

### 12.1.2. Clasificación del Software

De acuerdo con la sección 4.3 de la norma UNE-EN 62304, el software Neural Analytics ha sido clasificado como dispositivo de **Clase A**, dado que:

- No puede contribuir directamente a situaciones peligrosas, ya que funciona únicamente como herramienta de monitorización sin capacidad de efectuar acciones directas sobre el paciente.
- Su uso está destinado a aplicaciones no críticas de interfaz cerebro-computadora.
- El sistema incluye restricciones de uso explícitas que previenen su aplicación en escenarios clínicos críticos.

Esta clasificación determina el nivel de rigor aplicado a las pruebas y documentación del software, según lo estipulado en la norma.

## 12.2. Estrategia de Pruebas

Siguiendo los requisitos de la norma UNE-EN 62304 para software de Clase A, se ha implementado una estrategia de pruebas estructurada en tres niveles:

- **Pruebas unitarias**: Verificación de componentes individuales del software
- **Pruebas de integración**: Verificación de la interacción entre componentes
- **Pruebas del sistema**: Verificación del funcionamiento global del sistema

Como parte del alcance definido para este proyecto, se ha focalizado la estrategia de pruebas en los dos componentes principales: `neural_analytics_core` y `neural_analytics_gui`, al ser estos los módulos que constituyen el producto final destinado al usuario.

### 12.2.1. Herramientas y Entorno de Pruebas

Para la ejecución de las pruebas se ha utilizado el siguiente entorno:

| **Elemento** | **Descripción** |
|---|---|
| Hardware principal | Raspberry Pi 4 Model B (8GB RAM) |
| Sistema operativo | Poky Linux 64-bit |
| Dispositivo EEG | BrainBit (4 canales) |
| Dispositivos actuadores | Bombillas inteligentes Tapo L530E |
| Framework de pruebas unitarias | Rust Test Framework |
| Herramientas de cobertura | grcov, cargo-llvm-cov |
| Herramientas de monitorización | Raspberry Pi Diagnostics, htop |

Cuadro 12.1: Entorno de pruebas para Neural Analytics

## 12.3. Pruebas Unitarias

### 12.3.1. Estrategia de Pruebas Unitarias

Las pruebas unitarias se han diseñado para verificar el correcto funcionamiento de los componentes individuales del software, con especial énfasis en:

- Funcionamiento correcto de módulos aislados
- Manejo adecuado de casos límite
- Gestión de errores
- Consistencia en la interfaz de los componentes

De acuerdo con la sección 5.5.3 de la norma UNE-EN 62304, para cada prueba unitaria se han definido criterios de aceptación explícitos antes de su ejecución.

### 12.3.2. Pruebas Unitarias del Core (`neural_analytics_core`)

El módulo core del sistema, responsable de la lógica central de procesamiento y gestión de eventos, ha sido sometido a pruebas unitarias exhaustivas:

#### Prueba destacada: Servicio de Inferencia

Una prueba particularmente relevante fue la verificación del servicio de inferencia, el cual recibe datos EEG preprocesados y debe producir predicciones precisas sobre la intención del usuario. Esta prueba verifica no solo el procesamiento técnico del modelo ONNX, sino también la precisión semántica de sus predicciones, asegurando que el núcleo de la aplicación (la clasificación de ondas cerebrales) funcione correctamente.

Para la prueba se utilizaron datos EEG simulados que representan diferentes patrones cerebrales, verificando que:

- El servicio cargara correctamente el modelo ONNX
- Los datos de entrada fueran preprocesados adecuadamente
- Las predicciones coincidieran con las categorías esperadas
- El nivel de confianza de las predicciones superara el umbral establecido (70 %)

### 12.3.3. Pruebas Unitarias de la Interfaz (`neural_analytics_gui`)

Las pruebas unitarias para el componente de interfaz gráfica se centraron en:

#### Cobertura de Código

Se ha utilizado `cargo-llvm-cov` para medir la cobertura de código de las pruebas unitarias:

## 12.4. Pruebas de Integración

### 12.4.1. Enfoque de Pruebas de Integración

Las pruebas de integración se han enfocado en verificar la correcta interacción entre los distintos componentes del sistema, siguiendo la sección 5.6 de la norma UNE-EN 62304. Se han realizado pruebas en dos niveles:

1. **Integración intra-módulo**: Verificando la correcta interacción entre componentes del mismo módulo
2. **Integración inter-módulo**: Verificando la interacción entre el módulo core y la interfaz gráfica

### 12.4.2. Resultados de Pruebas de Integración

Las pruebas de integración realizadas han arrojado los siguientes resultados:

#### Prueba de Integración Destacada: Flujo Completo de Procesamiento

Una de las pruebas de integración más significativas verificó el flujo completo de procesamiento del sistema. Esta prueba evalúa la cadena completa de procesamiento desde la entrada de datos hasta la actuación de los dispositivos.

El procedimiento seguido fue:

1. Configurar el sistema completo con adaptadores simulados
2. Inyectar datos EEG que representan el patrón cerebral asociado al color "verde"
3. Ejecutar un ciclo completo de procesamiento
4. Verificar que el estado de la bombilla inteligente cambia al color verde
5. Comprobar que la interfaz de usuario se actualiza mostrando la predicción correcta
6. Confirmar que el nivel de confianza supera el umbral mínimo establecido (75 %)

Esta prueba verifica la integración end-to-end del sistema, desde la recepción de datos EEG hasta la activación de dispositivos y actualización de la interfaz, confirmando que todos los componentes trabajan juntos correctamente.

## 12.5. Pruebas del Sistema

Las pruebas del sistema se han diseñado para evaluar el comportamiento del sistema completo en condiciones reales y similares a las de uso final, conforme a la sección 5.7 de la norma UNE-EN 62304.

### 12.5.1. Casos de Prueba del Sistema

Se han diseñado y ejecutado los siguientes casos de prueba del sistema:

## 12.6. Pruebas de Seguridad

Aunque para software de Clase A la norma no exige pruebas de seguridad exhaustivas, se han realizado verificaciones básicas para garantizar que el sistema no compromete la privacidad del usuario ni genera riesgos inaceptables.

### 12.6.1. Gestión de Datos del Usuario

Se ha verificado que:

- Los datos EEG capturados se procesan localmente sin transmisión externa
- Los archivos de datos guardados se almacenan en formato anonimizado
- El sistema no recopila información personal identificable

### 12.6.2. Seguridad Eléctrica

El dispositivo BrainBit EEG utilizado en el sistema Neural Analytics cumple con los estándares FC y CE, garantizando su seguridad eléctrica y compatibilidad electromagnética. Dado que no se ha implementado hardware personalizado como parte de este proyecto, no se han requerido pruebas adicionales de seguridad eléctrica. No obstante, se ha verificado:

- Ausencia de interacciones eléctricas peligrosas entre el dispositivo EEG y el sistema
- Comportamiento seguro durante la carga del dispositivo EEG
- Ausencia de sobrecalentamiento en operación prolongada

## 12.7. Gestión de Anomalías

Durante el proceso de pruebas se detectaron e implementaron correcciones para las siguientes anomalías significativas:

De conformidad con la sección 5.8.2 de la norma, todas las anomalías residuales conocidas se han documentado y evaluado, determinando que ninguna representa un riesgo inaceptable para el usuario.

## 12.8. Matriz de Trazabilidad

Para garantizar la completitud de las pruebas, se ha desarrollado una matriz de trazabilidad que vincula los requisitos del sistema con los casos de prueba que los verifican:

## 12.9. Conclusión de la Validación

La validación del prototipo Neural Analytics ha seguido rigurosamente las directrices de la norma UNE-EN 62304:2007/A1:2016 para software de dispositivos médicos de Clase A. Se han completado satisfactoriamente los tres niveles de prueba requeridos (unitarias, integración y sistema), verificando el cumplimiento de todos los requisitos especificados.

Los resultados demuestran:

- Alta precisión en la detección de intenciones del usuario (86.8 % promedio)
- Tiempos de respuesta dentro de los límites aceptables (3.1s promedio)
- Operación estable y robusta del sistema

Las anomalías detectadas durante el proceso de prueba han sido debidamente documentadas y corregidas, verificando que ninguna de las anomalías residuales conocidas representa un riesgo inaceptable para el usuario.

El proceso de pruebas ha aportado evidencia objetiva de que el sistema Neural Analytics cumple con los requisitos especificados y funciona correctamente en el entorno previsto de uso, quedando validado para su implementación como sistema de interfaz cerebro-computadora para el control de dispositivos domóticos mediante ondas cerebrales.

| Módulo | Aspectos probados | Criterio de aceptación | Resultado |
|---|---|---|---|
| Adaptadores EEG | Conexión, lectura de datos, gestión de desconexiones | Lectura consistente de datos sin pérdida de muestras | PASA |
| Servicio de inferencia | Carga del modelo ONNX, preprocesamiento, inferencia | Predicciones coherentes con valores esperados | PASA |
| Controlador de dispositivos | Conexión con bombillas inteligentes, cambio de estados | Respuesta en ¡300ms a cambios de estado | PASA |
| Máquina de estados | Transiciones correctas entre estados del sistema | Comportamiento consistente ante eventos | PASA |
| Comunicación entre componentes | Bus de eventos, suscripciones, publicaciones | Entrega confiable de eventos | PASA |

Cuadro 12.2: Resultados de pruebas unitarias del módulo Core

A continuación, se detallan algunas de las pruebas unitarias implementadas para los componentes clave del `neural_analytics_core`:

### Pruebas del Servicio de Inferencia

El servicio de inferencia, responsable de la clasificación de patrones cerebrales mediante el modelo ONNX, ha sido probado a través de las siguientes pruebas unitarias:

- **test_model_loading**: Verifica que el modelo ONNX se carga correctamente desde diferentes rutas de archivo, incluyendo casos de archivos inexistentes.
- **test_prediction_with_valid_data**: Comprueba que el servicio realiza predicciones coherentes cuando recibe datos EEG válidos, comparando las salidas contra valores de referencia conocidos.
- **test_prediction_with_invalid_data**: Verifica el comportamiento del servicio ante datos mal formateados o fuera de rango, asegurando que maneja adecuadamente los errores.
- **test_prediction_performance**: Mide el tiempo de inferencia para garantizar que las predicciones se realizan dentro del umbral aceptable (¡100ms por predicción).

### Pruebas de los Adaptadores de Entrada

Los adaptadores responsables de la comunicación con el dispositivo BrainBit han sido probados mediante:

- **test_headset_connection**: Verifica el ciclo completo de conexión con el dispositivo BrainBit, comprobando la detección de dispositivos y establecimiento de conexión.
- **test_headset_data_capture**: Asegura que los datos EEG se capturan correctamente de todos los canales disponibles (T3, T4, O1, O2) con las frecuencias de muestreo esperadas.
- **test_impedance_check**: Comprueba el funcionamiento del mecanismo de verificación de impedancias para todos los electrodos, validando que los valores están dentro de los rangos aceptables.
- **test_disconnection_handling**: Verifica la gestión adecuada de desconexiones imprevistas, asegurando la liberación de recursos y la posibilidad de reconexión.

| **Componente** | **Aspectos probados** | **Criterio de aceptación** | **Resultado** |
|---|---|---|---|
| Vista principal | Renderizado de elementos, navegación entre secciones | Visualización correcta y respuesta a interacciones | PASA |
| Visualización de señales | Representación gráfica de señales EEG en tiempo real | Actualización fluida (¿25 FPS) | PASA |
| Módulo de calibración | Detección de impedancias, guía de usuario | Feedback preciso sobre calidad de contacto | PASA |
| Panel de configuración | Gestión de preferencias, validación de entradas | Persistencia de configuraciones | PASA |
| Indicadores de estado | Visualización del estado del sistema y predicciones | Correspondencia con estados internos | PASA |

Cuadro 12.3: Resultados de pruebas unitarias del módulo GUI

| **Módulo** | **Cobertura de código** |
|---|---|
| neural_analytics_core | 87.3 % |
| neural_analytics_gui | 82.1 % |

Cuadro 12.4: Cobertura de pruebas unitarias

| **Escenario de integración** | **Descripción** | **Resultado** |
|---|---|---|
| Core ↔ Adaptador EEG | Flujo de datos desde el dispositivo EEG al procesador del Core | PASA |
| Core ↔ Controlador de dispositivos | Activación de bombillas inteligentes tras la detección de intenciones | PASA |
| Core ↔ Interfaz | Visualización de estados del sistema y datos en tiempo real | PASA |
| Servicio de inferencia ↔ Máquina de estados | Transición correcta de estados basada en resultados de inferencia | PASA |
| Módulo de configuración ↔ Componentes del sistema | Aplicación efectiva de configuraciones de usuario | PASA |

Cuadro 12.5: Resultados de pruebas de integración

| ID | Descripción | Procedimiento | Criterio de aceptación | Resultado |
|---|---|---|---|---|
| SYS-01 | Inicialización del sistema | Iniciar la aplicación y verificar la carga de todos los módulos | Sistema operativo en ¡10s sin errores | PASA |
| SYS-02 | Conexión con dispositivo EEG | Encender el dispositivo BrainBit y conectarlo con la aplicación | Conexión establecida y datos fluyendo | PASA |
| SYS-03 | Calibración del dispositivo | Seguir el procedimiento de calibración | Impedancias aceptables en todos los canales | PASA |
| SYS-04 | Detección de pensamiento rojo" | Usuario piensa en color rojo durante 10 segundos | Sistema identifica correctamente la intención | PASA |
| SYS-05 | Detección de pensamiento "verde" | Usuario piensa en color verde durante 10 segundos | Sistema identifica correctamente la intención | PASA |
| SYS-06 | Activación de dispositivo por pensamiento | Usuario piensa en color específico y se verifica actuación | Bombilla cambia al color detectado en ¡1s | PASA |
| SYS-07 | Operación continua | Sistema funciona por ¿30 minutos continuos | Sin degradación de rendimiento ni fugas de memoria | PASA |
| SYS-08 | Recuperación ante errores | Simular desconexión del dispositivo EEG durante operación | Sistema detecta error y permite reconexión | PASA |

Cuadro 12.6: Casos de prueba del sistema

| ID | Descripción | Severidad | Resolución |
|---|---|---|---|
| AN-001 | Pérdida ocasional de datos EEG en sesiones prolongadas | Media | Implementación de buffer circular con retry automático |
| AN-002 | Falsos positivos en detección de señales con baja impedancia | Baja | Ajuste de umbral de confianza para clasificación |
| AN-003 | Latencia excesiva en interfaz gráfica durante visualización de señales | Media | Optimización del renderizado con muestreo adaptativo |
| AN-004 | Inconsistencia en la persistencia de configuraciones de usuario | Baja | Refactorización de sistema de almacenamiento de configuración |
| AN-005 | Problemas de reconexión con bombillas inteligentes tras pérdida de red | Media | Implementación de protocolo de reconexión resiliente |

Cuadro 12.7: Anomalías detectadas y resueltas

| Requisito | Descripción | Pruebas asociadas | Estado |
|---|---|---|---|
| REQ-F01 | Adquisición de señales EEG desde BrainBit | UC-001, TC-001, SYS-02 | VERIFICADO |
| REQ-F02 | Clasificación de patrones cerebrales | UC-005, TC-003, SYS-04, SYS-05 | VERIFICADO |
| REQ-F03 | Control de dispositivos por pensamiento | UC-007, TC-006, SYS-06 | VERIFICADO |
| REQ-F04 | Visualización de estado del sistema | UC-010, TC-008, SYS-01 | VERIFICADO |
| REQ-F05 | Calibración del dispositivo EEG | UC-012, TC-010, SYS-03 | VERIFICADO |
| REQ-NF01 | Tiempo de respuesta ¡3.5s | TC-020, SYS-06 | VERIFICADO |
| REQ-NF02 | Precisión de clasificación ¿80 % | TC-021, SYS-04, SYS-05 | VERIFICADO |
| REQ-NF03 | Operación continua durante ¿30min | TC-023, SYS-07 | VERIFICADO |

Cuadro 12.8: Matriz de trazabilidad de requisitos y pruebas

# Conclusiones

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi.

Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

# Agradecimientos

Quiero expresar mi más sincero agradecimiento al Dr. Antonio Molina Picó, tutor de este proyecto, por su invaluable orientación y dedicación. Sus conocimientos y consejos han sido fundamentales para dar forma y dirección a este trabajo de fin de grado.

Extiendo mi gratitud a la Universidad Politécnica de Valencia, institución que me ha proporcionado no solo los recursos académicos necesarios, sino también un entorno propicio para el desarrollo tanto profesional como personal durante estos años de formación.

Quisiera hacer una mención especial a la ciudad de Alcoy y su comunidad, que me han acogido con extraordinaria hospitalidad durante mi etapa universitaria. Asimismo, agradezco profundamente a aquellas personas que, especialmente durante mis primeros años en la ciudad, me brindaron su apoyo y me hicieron sentir como en casa, facilitando enormemente mi adaptación y crecimiento personal.

Finalmente, expreso mi más profundo agradecimiento a mi familia, amigos y seres queridos, cuyo apoyo incondicional y comprensión han sido pilares fundamentales en este recorrido académico. Su presencia y ánimo constante han sido decisivos para alcanzar esta meta.

# Bibliografía


Asociación Española de Normalización (2016). UNE-EN 62304:2007/A1:2016 - Software para dispositivos médicos - Procesos del ciclo de vida del software. Norma basada en IEC 62304:2006 con modificaciones específicas para el mercado español.

Brouwer, G. J. and Heeger, D. J. (2013). Categorical clustering of the neural representation of color. *Journal of Neuroscience*, 33(39):15454–15465.

Kandel, E. R., Jessell, T. M., and Schwartz, J. H. (2001). *Principios de neurociencia*. McGraw-Hill Interamericana, Madrid, 4a ed. edition.

Neurotechnology Systems LLC (2024). *BrainBit Datasheet*. Accessed: 2025-02-18.

Raschka, S., Liu, Y. H., and Mirjalili, V. (2022). *Machine Learning with PyTorch and Scikit-Learn*. Packt Publishing, Birmingham, UK.

Raspberry Pi Foundation (2020). *Raspberry Pi 4 Model B Datasheet*. Accessed: 2025-02-18.

Rissman, J. and Wagner, A. D. (2012). Distributed representations in memory: Insights from functional brain imaging. *Annual Review of Psychology*, 63:101–128.

Siewert, S. and Pratt, J. (2016). *Real-time embedded components and systems using Linux and RTOS*. Mercury Learning and Information.

Squire, L. R. and Zola-Morgan, S. (1991). The medial temporal lobe memory system. *Science*, 253(5026):1380–1386.