

Integración del SoC nRF5340 en una plataforma de neuroestimulación implantable (NeuroSoC)

MEMORIA DE PROYECTO PRESENTADA A LA FACULTAD DE
INGENIERÍA DE LA UNIVERSIDAD DE LA REPÚBLICA POR

Mauricio Guerrero Martorell, Irina Magallanes Cohn,
Martina Montichelli Aplanalp

EN CUMPLIMIENTO PARCIAL DE LOS REQUERIMIENTOS
PARA LA FINALIZACIÓN DE LA CARRERA DE
INGENIERÍA ELÉCTRICA.

TUTOR

Santiago Martínez Universidad de la República
Leonardo Steinfeld Universidad de la República

TRIBUNAL

Federico Nin Integer Holdings Corporation
Francisco Veirano Universidad de la República
Julio Pérez Universidad de la República

Montevideo
martes 7 julio, 2026

Integración del SoC nRF5340 en una plataforma de neuroestimulación implantable (NeuroSoC), Mauricio Guerrero Martorell, Irina Magallanes Cohn, Martina Montichelli Aplanalp.

Esta tesis fue preparada en L^AT_EX usando la clase iietesis (v1.2).
Contiene un total de 159 páginas.
Compilada el martes 7 julio, 2026.
<http://iie.fing.edu.uy/>

Caminante, no hay camino, se hace camino al andar.

ANTONIO MACHADO

Esta página ha sido intencionalmente dejada en blanco.

Agradecimientos

Finalmente, llegó el momento de agradecer a todos los que formaron parte de este proceso y nos allanaron el camino. En primer lugar, a nuestras familias, no solo por acompañarnos y apoyarnos durante toda la carrera, sino por todo lo que han hecho por nosotros y nos ha permitido llegar hasta aquí. Lo que somos hoy es el reflejo de sus valores, sus enseñanzas y del amor con el que nos han acompañado siempre. De igual forma, agradecer a nuestros amigos, que en el día a día o a la distancia fueron contención y compañía a lo largo de estos años. Gracias por entender nuestras ausencias y celebrar nuestros logros.

A los profesores, que con vocación nos brindaron todas las herramientas que tuvieron a su disposición para nuestra formación. A los que, con cercanía y compromiso, nos transmitieron no solo conocimientos sino también tranquilidad y orientación en momentos de mayor dificultad. A todos nuestros compañeros que conocimos a lo largo de la carrera, entre los que siempre nos brindamos apoyo y aliento mutuo. A la *UdelaR*, por la formación de calidad que nos brindó, enseñándonos a aprender.

A nuestros compañeros de trabajo, por su disposición para ayudarnos técnica y emocionalmente. A *Impulse Dynamics*, por brindarnos el espacio e instrumentos para el desarrollo del proyecto. A *Integer Holdings Corporation* por la propuesta de proyecto y los materiales brindados.

Finalmente, queremos agradecer especialmente a nuestros tutores, Santi y Leo. Valoramos profundamente su constante dedicación y todas las contribuciones valiosas que aportaron al proyecto. El resultado obtenido es, en gran medida, fruto de sus aportes y colaboración.

Esta página ha sido intencionalmente dejada en blanco.

A nuestros padres.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

La neuroestimulación es una técnica ampliamente utilizada para el tratamiento de patologías como el dolor crónico y trastornos neurológicos, cuyo desarrollo reciente ha estado impulsado por la miniaturización y la personalización de las terapias. En este contexto, los dispositivos modernos requieren sistemas embebidos capaces de operar en tiempo real, cumpliendo estrictos requisitos de seguridad, confiabilidad y eficiencia. Motivado por estas necesidades, el proyecto NeuroSoC propone evaluar la viabilidad de migrar la arquitectura de un neuroestimulador comercial —originalmente basado en dos microcontroladores— hacia una solución integrada en un único *System on Chip* (SoC).

Para ello, se desarrolló una prueba de concepto utilizando el SoC nRF5340 y el sistema operativo de tiempo real Zephyr, incorporando además comunicación inalámbrica mediante Bluetooth Low Energy (BLE), con el objetivo de analizar las ventajas, limitaciones y desafíos de esta integración.

En cuanto al *hardware*, se partió de un neuroestimulador provisto por *Integer Holdings Corporation*, implementando una arquitectura simplificada que reemplaza ambos microcontroladores por un único SoC *dual core*. Se diseñó una nueva PCB y se introdujeron modificaciones como la incorporación de un DAC externo y la reducción de electrodos, eliminando bloques no esenciales. El diseño fue validado mediante la observación y medición de las formas de onda esperadas. En relación con la comunicación, se logró el emparejamiento mediante intercambio a través de NFC, así como la correcta recepción de los parámetros de estimulación. Respecto al *firmware*, se reemplazó la implementación *bare-metal* por una basada en Zephyr, logrando cumplir los requisitos temporales del sistema original. Se identifican como oportunidades de mejoras ajustar el *layout* del PCB, ya que las diferencias con el original generan un mayor ruido en la señal, y estudiar cómo reducir un leve *jitter* asociado al desarrollo con un RTOS y a la sincronización del ADC con los pulsos de estimulación.

Por otra parte, la comunicación BLE para la programación de la terapia se implementó de forma satisfactoria, incorporando mecanismos de seguridad como emparejamiento, autenticación y cifrado, mediante el uso de *Near Field Communication* (NFC).

En síntesis, el prototipo desarrollado demuestra la viabilidad de una arquitectura integrada basada en un único SoC *dual core*, validando el uso de un RTOS y comunicación BLE en aplicaciones de neuroestimulación. Los resultados evidencian ventajas en términos de simplificación y flexibilidad, así como oportunidades de mejora para optimizar el desempeño y la robustez del sistema.

Esta página ha sido intencionalmente dejada en blanco.

Glosario

ACV	Accidente cerebrovascular.
ADC	Analog-to-Digital Converter.
AD	Advertisement Data.
AIMD	Active Implantable Medical Device.
API	Application Programming Interface.
ATT	Attribute Protocol.
BLE	Bluetooth Low Energy.
CDMO	Contract Development and Manufacturing Organization.
CPU	Central Processing Unit.
CRC	Cyclic Redundancy Check.
DAC	Digital-to-Analog Converter.
DBS	Deep Brain Stimulation.
DHKey	Diffie–Hellman Key.
DIS	Device Information Service.
DRC	Design Rule Check.
EIR	Extended Inquiry Response.
FDA	Food and Drug Administration.
GAP	Generic Access Profile.
GATT	Generic Attribute Profile.
GCC	GNU Compiler Collection.
GPIO	General Purpose Input/Output.
HFCLK	High Frequency Clock.

IEEE	Institute of Electrical and Electronics Engineers.
IRK	Identity Resolving Key.
ISM	Industrial, Scientific and Medical (frequency band).
L2CAP	Logical Link Control and Adaptation Protocol.
LESC	LE Secure Connections.
LL	Link Layer.
LTK	Long Term Key.
MITM	Man-in-the-middle.
MTU	Maximum Transmission Unit.
NDEF	NFC Data Exchange Format.
NFC	Near Field Communication.
NSS	Neurostimulation Service.
OOB	Out-of-Band.
PCB	Printed Circuit Board.
PDU	Protocol Data Unit.
PHY	Physical Layer.
PNS	Phrenic Nerve Stimulation.
PPI	Programmable Peripheral Interconnect.
PRAND	Private Random Part.
PWM	Pulse Width Modulation.
RF	Radio Frequency.
RFU	Reserved for Future Use.
RNS	Responsive Neurostimulation.
RTOS	Real-Time Operating System.
SAADC	Successive Approximation Analog-to-Digital Converter.
SCS	Spinal Cord Stimulation.
SDK	Software Development Kit.
SMD	Surface Mount Device.

SoC	System on Chip.
SPI	Serial Peripheral Interface.
STK	Short Term Key.
TK	Temporary Key.
UART	Universal Asynchronous Receiver-Transmitter.
UUID	Universally Unique Identifier.
VNS	Vagus Nerve Stimulation.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Agradecimientos	5
Resumen	9
Glosario	11
1. Introducción	19
1.1. Motivación	19
1.2. Objetivos	21
1.2.1. Objetivo general	21
1.2.2. Objetivos específicos	22
1.2.3. Alcance	22
1.3. Organización del Documento	23
2. Descripción del sistema	25
2.1. Introducción	25
2.2. Requerimientos del sistema	28
2.3. Hardware	29
2.4. Firmware	30
3. Plataforma de desarrollo: nRF Connect SDK y Zephyr RTOS	33
3.1. nRF Connect SDK	33
3.2. Zephyr RTOS	34
3.2.1. Aplicaciones multihilo	35
3.3. Elementos de una aplicación nRF Connect SDK	36
3.3.1. Archivo de configuración de aplicación: <code>prj.conf</code>	36
3.3.2. Devicetree	37
3.3.3. CMake	39
3.3.4. Sysbuild	39
4. Desarrollo de Hardware	41
4.1. Descripción del Sistema	42
4.1.1. Bloque de estimulación	43
4.1.2. Bloque de control y comunicación	45
4.1.3. Bloque de alimentación	47
4.2. Componentes	48

Tabla de contenidos

4.2.1.	System on Chip nRF5340	48
4.2.2.	Elección del conversor digital-analógico (DAC)	52
4.3.	Configuración para pruebas iniciales	53
4.4.	Diseño del PCB	55
4.4.1.	Fabricación del PCB	55
5.	Diseño e implementación de Firmware	57
5.1.	Introducción	57
5.2.	Arquitectura del Firmware	57
5.2.1.	thread_application	60
5.2.2.	thread_ble	61
5.2.3.	thread_housekeeping	62
5.2.4.	thread_processing	62
5.3.	Subsistema BLE	63
5.3.1.	Proceso de conexión	63
5.3.2.	Advertising	63
5.3.3.	Capa GATT	66
5.3.4.	Conexión	67
5.3.5.	Configuración de los parámetros BLE	67
5.3.6.	Emparejamiento y autenticación	68
5.3.7.	Servicio Neurostimulation Service (NSS)	73
5.3.8.	Módulo user_type	78
5.4.	Subsistema de Estimulación	79
5.4.1.	Funcionamiento de la máquina de estados de estimulación	79
5.4.2.	Implementación de la máquina de estados: módulo stim_state_machine	81
5.4.3.	Módulo electrodes	83
5.4.4.	Módulo sink	84
5.4.5.	Módulo timer	86
5.4.6.	Módulo dac	86
5.4.7.	Módulo adc	88
5.4.8.	Módulo battery	91
5.4.9.	Módulo magnet	92
5.5.	Configuración y compilación del proyecto en Zephyr	92
5.5.1.	Devicetree	92
5.5.2.	prj.conf	93
5.5.3.	CMake	94
6.	Resultados	97
6.1.	Generación de pulsos de estimulación	97
6.1.1.	Análisis de datos relevados	106
6.2.	Validación de la comunicación BLE	106
6.2.1.	Validación del advertising	107
6.2.2.	Proceso de establecimiento de conexión	108
6.2.3.	Indicadores iniciales de conexión	108
6.2.4.	Establecimiento de conexión y descubrimiento de servicios GATT	108

6.2.5. Validación del emparejamiento y seguridad	109
6.2.6. Activación del cifrado	110
6.2.7. Rechazo de conexión no autenticada	111
6.2.8. Conclusión de la validación	111
6.3. Caracterización del PCB implementado	112
6.3.1. Variaciones en la caída de tensión en el tejido	112
6.3.2. Funcionamiento del boost converter	114
6.4. Perfil de Consumo	116
6.4.1. Configuración para las pruebas	116
6.4.2. Consumo de comunicación BLE y estimulación	117
6.4.3. Duración estimada del ciclo de batería	118
7. Conclusiones	121
7.1. Conclusiones	121
7.2. Trabajo a futuro	122
Apéndices	1
A. Fundamentos de Bluetooth Low Energy	1
A.1. Introducción	1
A.2. BLE Protocol Stack	1
A.2.1. Capa GAP	2
A.2.2. Capas GATT y ATT	3
A.2.3. Capa PHY	4
A.3. Advertising	4
A.3.1. Proceso de advertising	5
A.3.2. Tipos de advertising	7
A.3.3. Bluetooth address	7
A.3.4. Paquetes de advertising	9
A.4. Conexiones	10
A.4.1. Parámetros de conexión	12
A.5. Intercambio de datos	13
A.5.1. Operaciones de la capa GATT	14
A.5.2. Atributos	14
A.5.3. Servicios	15
A.5.4. Características	16
A.6. Seguridad en BLE	18
A.6.1. Proceso de emparejamiento	18
A.6.2. Niveles de seguridad	20
A.6.3. NFC	20
Índice de tablas	26
Índice de figuras	28

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Introducción

En este capítulo se presenta el proyecto NeuroSoC, junto con la motivación para su realización, así como sus objetivos y alcance. Asimismo, se describe el contexto en el cual se desarrolla el proyecto y del cual se parte. Finalmente, se describe la organización del documento.

1.1. Motivación

La estimulación eléctrica constituye una técnica ampliamente utilizada en la actualidad para el tratamiento de diversas patologías. En particular, los dispositivos de neuroestimulación han demostrado ser herramientas eficaces en el manejo de dolores crónicos, así como en trastornos neurológicos [1]. En los últimos años, estos dispositivos han experimentado una evolución significativa, orientada hacia la miniaturización y la personalización de las terapias, permitiendo tratamientos cada vez más específicos y adaptados a las necesidades del paciente [2].

Como se menciona en *Pain Management NYC*, “la neuroestimulación comprende un conjunto de técnicas que utilizan señales eléctricas, magnéticas o ultrasónicas para modular la actividad del sistema nervioso periférico” [1]. Estas técnicas se emplean en el tratamiento de diversas afecciones, como el dolor crónico, la enfermedad de Parkinson, la epilepsia y la apnea del sueño, entre otras, contribuyendo a mejorar la funcionalidad y la calidad de vida de los pacientes.

Dentro de los neuroestimuladores disponibles en el mercado, es posible distinguir tres categorías principales: monitores neuronales, sistemas *open-loop* y sistemas *closed-loop*. Los denominados monitores neuronales se limitan exclusivamente a la adquisición y registro de la actividad neuronal, con el objetivo de analizar el comportamiento del sistema nervioso sin intervenir mediante estimulación. Por otro lado, los sistemas *open-loop* administran la terapia mediante la entrega de estímulos definidos por un conjunto de parámetros previamente configurados por el profesional médico en función de la patología y el tratamiento requerido, sin considerar la respuesta en tiempo real del paciente. Finalmente, los sistemas *closed-loop* incorporan la capacidad de sensado de señales neurales, lo que permite ajustar dinámicamente los parámetros de estimulación en función de la respuesta fisiológica observada.

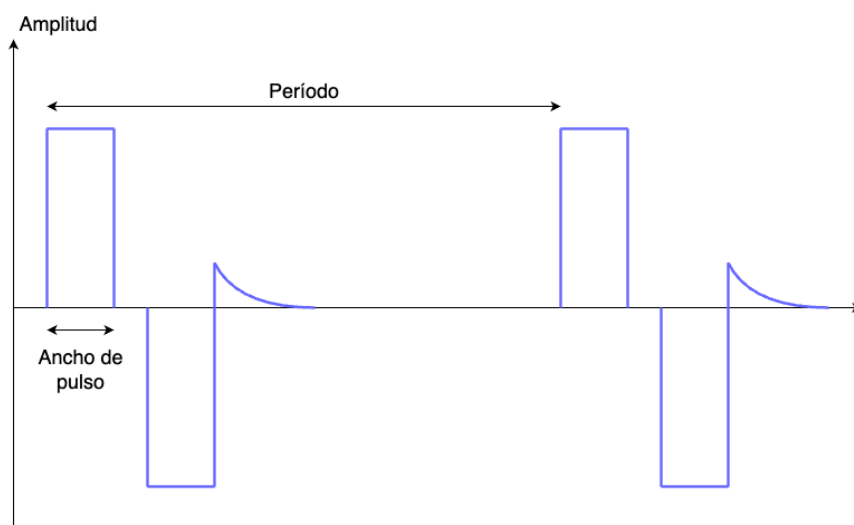


Figura 1.1: Forma de onda típica de estimulación. Se observa una secuencia periódica de pulsos rectangulares, definidos por su amplitud y ancho de pulso, seguida de una fase de balance activo y otra de balance pasivo. Esta secuencia se repite con una frecuencia $\frac{1}{T}$, siendo T el período.

En este tipo de sistemas, la estrategia de estimulación más ampliamente utilizada se basa en la inyección de corriente constante en el tejido. La forma de onda típica puede ser observada en la Figura 1.1. Consiste en una secuencia de pulsos rectangulares, definidos principalmente por su amplitud de corriente y su ancho de pulso. En muchos casos, estos pulsos son seguidos por una fase de balance activo, cuyo objetivo es evitar la inyección neta de corriente continua (DC) en el tejido, reduciendo así posibles efectos adversos. Además, de forma complementaria a esta etapa, se suele incorporar un mecanismo de balance pasivo. Esta secuencia de estimulación se repite de manera periódica mientras el sistema de estimulación permanece activo [3].

Desde el punto de vista tecnológico, los dispositivos médicos modernos se implementan como sistemas embebidos complejos, sujetos a estrictos requisitos de seguridad, confiabilidad y cumplimiento normativo. Organismos regulatorios como la Dirección General de la Comisión Europea y la Administración de Alimentos y Medicamentos de los Estados Unidos (FDA) establecen lineamientos rigurosos que estos sistemas deben cumplir. En este contexto, el firmware del sistema desempeña un rol central, ya que debe garantizar un funcionamiento seguro, confiable y predecible, especialmente en aplicaciones donde existen restricciones temporales estrictas.

En particular, la mayoría de los sistemas médicos operan bajo requisitos de tiempo real. Esto implica que no solo deben responder rápidamente, sino dentro de intervalos de tiempo definidos, asegurando un comportamiento determinista. Es decir, la respuesta a un evento debe ocurrir en el instante adecuado, evitando tanto retrasos como ejecuciones prematuras. Para cumplir con estas exigencias, resulta fundamental gestionar la concurrencia y los tiempos de ejecución de manera predecible [4].

En este marco, el proyecto NeuroSoC propone el desarrollo de una plataforma de neuroestimulación basada en circuitos de estimulación provistos por *Integer Holdings*

Corporation, con el objetivo de evaluar la factibilidad, así como las ventajas y desventajas, de implementar el sistema utilizando el *System on Chip* nRF5340, el cual cuenta con una arquitectura *dual-core*. Esta migración implicó la adopción de Zephyr RTOS como base del *firmware*, ya que para la familia nRF53 Nordic no brinda soporte oficial para un flujo de desarrollo *bare-metal* dentro de su ecosistema actual.

El presente proyecto se basa en un neuroestimulador provisto por el cliente, del cual se dispuso de los requerimientos del sistema, el código fuente, un *test board* y los esquemáticos. La arquitectura original del dispositivo está compuesta por dos microcontroladores: uno destinado a la gestión de la comunicación y otro encargado de la ejecución de la terapia de estimulación.

El objetivo principal del trabajo consiste en reemplazar dicha arquitectura por una solución integrada basada en un único *System on Chip* (SoC), específicamente el nRF5340. Con este fin, se desarrolla una prueba de concepto que implementa un subconjunto representativo de los requerimientos del sistema original, permitiendo evaluar la viabilidad técnica de la migración hacia una arquitectura unificada.

La implementación se lleva a cabo utilizando Zephyr, un sistema operativo de tiempo real de código abierto. Esto permite, a su vez, analizar su utilidad para aplicaciones biomédicas embebidas, particularmente en aspectos como la gestión de concurrencia, la operación en tiempo real y la escalabilidad del sistema.

En cuanto a la comunicación, se implementa una solución basada en BLE, aprovechando las capacidades inalámbricas integradas del SoC. De esta forma, se evalúa también el desempeño de este protocolo en el contexto de la aplicación propuesta.

Desde el punto de vista del *hardware*, se procura mantener el diseño original, introduciendo únicamente las modificaciones estrictamente necesarias. En particular, dado que el SoC seleccionado no dispone de convertidor digital-analógico (DAC) integrado, se incorpora un dispositivo externo para cumplir con los requerimientos de generación de señal del sistema de estimulación.

Finalmente, se busca reproducir un comportamiento funcional lo más cercano posible al sistema original, aunque con una reducción en la cantidad de funcionalidades y mecanismos de verificación implementados, priorizando aquellos aspectos necesarios para evaluar la factibilidad del cambio de arquitectura.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo de este proyecto es desarrollar un prototipo de plataforma de neuroestimulación basado en el SoC nRF5340, con el fin de evaluar la factibilidad de sustituir la implementación actual, basada en dos microcontroladores, por un único SoC de doble núcleo. Este SoC permite distribuir las tareas de estimulación y comunicación de manera equivalente a la arquitectura actual. A través de esto, se busca determinar si su uso puede reducir el tamaño y consumo del dispositivo sin comprometer su rendimiento ni funcionalidad.

Capítulo 1. Introducción

1.2.2. Objetivos específicos

El objetivo general se descompone en los siguientes objetivos específicos:

1. Analizar la arquitectura de un sistema médico implantable, con énfasis en la electrónica del dispositivo y su interacción con el resto del sistema, identificando los parámetros programables relevantes en un neuroestimulador, tales como amplitud, frecuencia, ancho de pulso, entre otros.
2. Adoptar el SoC nRF5340 y el sistema operativo Zephyr, implementando una arquitectura de *firmware* que permita la ejecución concurrente de tareas de estimulación y comunicación y el uso eficiente de sus funcionalidades.
3. Implementar la comunicación mediante BLE entre el SoC y un dispositivo externo, permitiendo la transmisión y recepción de parámetros de configuración, mediante un canal seguro.
4. Desarrollar una prueba de concepto para la generación de señales de estimulación utilizando los circuitos provistos (*test board*) y la placa de desarrollo del SoC nRF5340, a partir de parámetros configurables dentro de los rangos especificados.
5. Validar el funcionamiento completo del sistema, integrando la configuración de parámetros vía BLE con la generación de la señal de estimulación.
6. Diseñar una placa de circuito impreso basada en el SoC nRF5340, integrando y adaptando los esquemáticos del circuito de estimulación provisto por el cliente.
7. Evaluar el desempeño del sistema en términos de consumo energético, analizando posibles optimizaciones asociadas al uso de Zephyr y a la gestión de energía del SoC.
8. Evaluar la precisión en la temporización del sistema y su impacto sobre la forma de onda generada, verificando el cumplimiento de los requerimientos de tiempo y analizando el *jitter* y la latencia asociados al uso de Zephyr.

1.2.3. Alcance

El alcance del proyecto comprende, en primer lugar, el análisis de la documentación técnica del sistema existente, con énfasis en los documentos de diseño del estimulador. Asimismo, incluye la adopción y utilización de Zephyr como sistema operativo en tiempo real para el desarrollo del *firmware*, evaluando su adecuación para este tipo de aplicaciones de dispositivos médicos.

Como resultado del trabajo, se desarrolla una prueba de concepto capaz de recibir comandos mediante BLE, a través de un protocolo diseñado para este fin. Asimismo, el sistema es capaz de ejecutar la estimulación conforme a los parámetros configurados. El protocolo define distintos roles de usuario, con diferentes niveles de acceso a las funcionalidades del sistema. En este contexto, se implementan mecanismos de seguridad para el intercambio de información. Finalmente, se valida que la señal generada se corresponda

con la configuración establecida, garantizando el correcto funcionamiento del sistema de estimulación.

A nivel de *hardware*, se realizan las adaptaciones necesarias sobre los circuitos originales para permitir la implementación del neuroestimulador con el SoC. Para ello, se diseña una PCB que incorpora dichas modificaciones, incluyendo la integración del SoC en reemplazo de los microcontroladores originales.

Por otra parte, se analizan estrategias orientadas a la optimización del consumo energético provistas por el RTOS.

El diseño e implementación de una interfaz de usuario completa y orientada al uso final no formó parte del alcance de este proyecto.

Finalmente, se incorporan otros mecanismos de seguridad, incluyendo la validación de parámetros de entrada y la implementación de chequeos periódicos de integridad del sistema, con el objetivo de aumentar la robustez de la solución propuesta.

1.3. Organización del Documento

El presente documento comienza con una visión general del proyecto, avanzando progresivamente hacia los detalles de implementación y los resultados obtenidos. En este capítulo se introduce el proyecto, presentando su motivación, objetivos y alcance. En el Capítulo 2 se realiza una descripción general del sistema. Los Capítulos 3 y A abordan los fundamentos teóricos de la plataforma de desarrollo y de BLE, respectivamente. En el Capítulo 4 se describe el *hardware* del sistema, mientras que en el Capítulo 5 se detalla el *firmware* implementado. Finalmente, en el Capítulo 6 se presentan los resultados obtenidos y en el Capítulo 7 se exponen las conclusiones finales, junto con posibles líneas de trabajo futuro.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 2

Descripción del sistema

El objetivo de este capítulo es brindar una visión general del sistema completo. En primer lugar, se presenta una descripción del sistema diseñado. Luego, se expone una especificación global del *hardware* y del *firmware*, los cuales serán desarrollados en profundidad en los capítulos 4 y 5, respectivamente. Por último, se detallan los requerimientos del sistema.

2.1. Introducción

En esta Sección se describe la arquitectura de partida y la propuesta de migración hacia una solución basada en el SoC nRF5340. Esta transición no sólo implica el reemplazo de los dos microcontroladores originalmente utilizados por un SoC, sino también la implementación del *firmware* sobre el sistema operativo en tiempo real Zephyr, adoptando una nueva arquitectura de software. Asimismo, se presentan las principales decisiones de diseño que guían esta migración.

El desarrollo se basa en un neuroestimulador provisto por *Integer Holdings Corporation*, utilizado en una amplia variedad de terapias. *Integer* es una de las principales organizaciones a nivel mundial en el desarrollo y fabricación por contrato de dispositivos médicos (CDMO). La empresa participa en el diseño y producción de dispositivos en diversas áreas, entre las que se incluyen aplicaciones cardiovasculares, vasculares y de neuromodulación, así como sistemas de manejo del ritmo cardíaco [5].

Esta alternativa se consideró viable debido a que el SoC presenta una arquitectura de doble núcleo, compuesta por un núcleo de aplicación y un núcleo de comunicación, lo que permite asumir las funcionalidades previamente distribuidas entre ambos microcontroladores sin comprometer el desempeño del sistema. Asimismo, esta integración reduce la complejidad del *hardware*, disminuye potencialmente el consumo energético y permite reducir el tamaño del dispositivo al reemplazar dos circuitos integrados por un único chip. A su vez, simplifica el ruteo de la placa de circuito impreso, lo que facilita su diseño, mejora la integridad de señal y contribuye a una mayor confiabilidad del sistema reduciendo potenciales puntos de falla.

Con este fin, se planteó el desarrollo de una implementación equivalente a la original, pero basada en el sistema operativo en tiempo real Zephyr. Esto implicó evaluar si, bajo

Capítulo 2. Descripción del sistema

este nuevo enfoque, era posible cumplir con los requerimientos originales del sistema, en particular aquellos asociados a restricciones temporales.

La adopción de un RTOS, en lugar de una implementación *bare metal*, permite mejorar la portabilidad del sistema y simplificar significativamente el desarrollo del *firmware*, especialmente en lo relativo a la gestión de recursos del microcontrolador. Esto habilita enfocar el esfuerzo de desarrollo en la mejora de la tecnología y la incorporación de nuevas funcionalidades y mecanismos de control.

Bajo estas hipótesis, se planteó el diseño de una prueba de concepto de un neuroestimulador considerando las principales funcionalidades y restricciones características de un AIMD de este tipo.

Entre las funcionalidades principales se encuentran la generación de la señal de estimulación (forma de onda) y la programabilidad de los parámetros de la terapia, la cual se realiza de forma remota mediante comunicación inalámbrica con el programador. Por otro lado, las principales restricciones están asociadas al tamaño y al consumo energético del sistema.

Asimismo, se tuvieron en cuenta aspectos de seguridad, tanto desde el punto de vista de la protección de datos (*security*), como de la seguridad funcional (*safety*), incorporando distintos mecanismos de verificación con el objetivo de minimizar el riesgo de daño al paciente¹. En particular, se implementó un mecanismo de emparejamiento BLE asistido por NFC, que permite establecer una autenticación más segura. Este enfoque restringe la vinculación del dispositivo únicamente a usuarios autorizados, ya que requiere un intercambio de información mediante proximidad física inmediata, reduciendo así el riesgo de accesos no autorizados o conexiones accidentales.

Es importante destacar que el sistema desarrollado corresponde a una prueba de concepto. Por lo tanto, no se buscó implementar una solución completa que cumpla con todos los requerimientos de un dispositivo comercial, sino evaluar la factibilidad del enfoque propuesto, considerando algunos aspectos relevantes e implementando únicamente las funcionalidades necesarias para dicha evaluación.

En base a estos criterios, se diseñó el sistema que se presenta en la Figura 2.1. Este está compuesto por un dispositivo (PCB) que integra el SoC nRF5340 y los circuitos de estimulación, del cual se conectan los electrodos destinados al paciente. Adicionalmente, el sistema utiliza una aplicación móvil (*nRF Connect Mobile*) que, mediante una conexión BLE, permite configurar los parámetros de la terapia, así como iniciar y finalizar su ejecución. Cabe destacar que esta aplicación es utilizada únicamente con fines de depuración y validación del sistema.

En la Figura 2.2 se presenta un diagrama del flujo de operación del sistema. En primer lugar, se muestra el proceso de emparejamiento mediante NFC, utilizado para la autenticación del usuario y el establecimiento de la conexión BLE. El sistema define distintos roles de usuario, a partir de los cuales se establecen los permisos y las funcionalidades accesibles. Una vez conectado, mediante la aplicación *nRF Connect Mobile* el usuario puede configurar los parámetros de la terapia en caso de contar con el rol de

¹En español, los términos *security* y *safety* suelen traducirse ambos como “seguridad”. Sin embargo, en el ámbito de la ingeniería, *security* se refiere a la protección frente a accesos no autorizados, ataques o compromisos de la información, mientras que *safety* alude a la capacidad del sistema para operar sin causar daños a las personas, al entorno o al propio equipo.

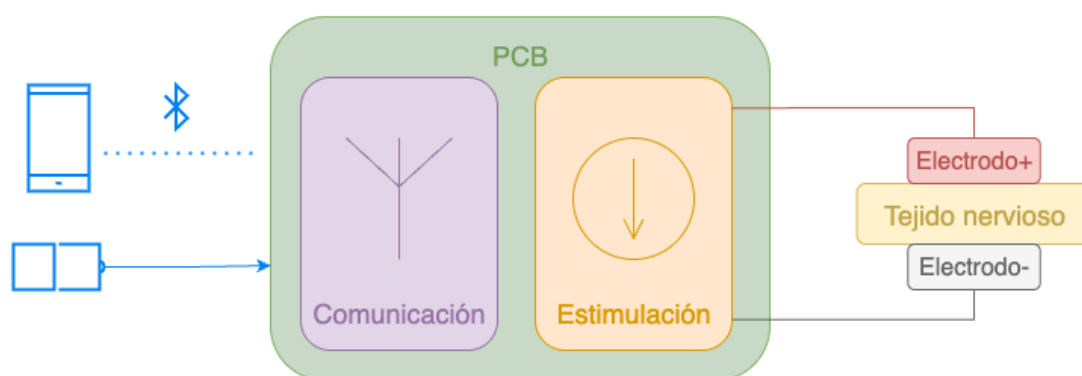


Figura 2.1: Descripción general del sistema



Figura 2.2: Diagrama del flujo de operación del sistema incluyendo las etapas de emparejamiento del dispositivo vía NFC para la conexión Bluetooth, la configuración de parámetros por parte del programador mediante una aplicación móvil y ejecución de la terapia por el paciente utilizando el dispositivo previamente configurado.

Programador. Posteriormente, la terapia puede ser iniciada tanto por el paciente como por el programador. Una vez iniciada la ejecución, el dispositivo genera la forma de onda de la Figura 1.1, cuyos parámetros fueron programados mediante BLE. Ésta es aplicada a través de los electrodos seleccionados, como se detallará en las secciones siguientes.

El PCB mostrado corresponde al diseño desarrollado en el proyecto, el cual se basa en la circuitería provista y permite generar la forma de onda de estimulación que sería aplicada al paciente.

En las siguientes secciones se presentan los requerimientos del sistema y, posteriormente, una descripción general del *hardware* y el *firmware* implementados.

2.2. Requerimientos del sistema

En esta Sección se establecen los requerimientos del sistema, que definen las condiciones que deben cumplirse para garantizar su correcto funcionamiento y sirven como base para el diseño e implementación del *hardware* y el *firmware*. Los requerimientos de los parámetros de estimulación corresponden a los definidos en la plataforma de estimulación provista por el cliente.

Dispositivo: Deberá implementarse sobre una placa de circuito impreso diseñada específicamente para la aplicación, la cual integrará el SoC nRF5340 de Nordic Semiconductor, los circuitos de estimulación provistos por el cliente. Asimismo, el dispositivo deberá ser alimentado con una fuente externa de 3.8 V. Además, se incorporará un botón en sustitución del *reed-switch* original, con el fin de permitir la detención de la terapia.

Plataforma de prueba: Durante las etapas de desarrollo, validación y depuración del *firmware*, se empleará una plataforma de pruebas compuesta por la *test board* suministrada por el cliente, la placa de desarrollo del SoC nRF5340 y eventuales componentes o circuitos auxiliares necesarios. Esta configuración permitirá interconectar las señales generadas por el microcontrolador con la *test board*, facilitando la verificación del *firmware* y la evaluación de las formas de onda producidas, a fin de corroborar que cumplan con las especificaciones de diseño.

Firmware: El desarrollo del *firmware* se realiza sobre el sistema operativo en tiempo real Zephyr.

Comunicación: El dispositivo debe ser capaz de establecer comunicación mediante BLE con un dispositivo móvil, permitiendo la configuración de los parámetros de estimulación. Esta comunicación debe garantizar la integridad y confiabilidad de los datos intercambiados. En este sentido, el sistema debe verificar la integridad de los datos recibidos mediante un código de redundancia cíclica (CRC), así como validar que los parámetros se encuentren dentro de los rangos operacionales definidos antes de ser almacenados o aplicados.

En caso de que los parámetros recibidos se encuentren fuera de los rangos permitidos, el dispositivo no los acepta. De igual forma, si se detecta corrupción en los datos, evidenciada por una verificación de integridad fallida, la acción solicitada tampoco es ejecutada. En cambio, cuando los parámetros son válidos tanto en términos de rango como de integridad, el dispositivo procede a ejecutar la acción correspondiente.

Control de la terapia: La terapia podrá ser iniciada y detenida por el programador y el paciente, quienes además deberán poder consultar el conjunto de parámetros de estimulación configurados.

Generación de la señal de estimulación: El dispositivo deberá generar pulsos bifásicos de estimulación en corriente, cuyos parámetros —amplitud, frecuencia y ancho de pulso— serán configurables por el usuario a través de una interfaz BLE.

Frecuencia de estimulación: Será configurable en un rango de 1 a 1500 Hz con una tolerancia en el período del máximo entre 16 μ s y 1 %. Los pasos serán configurados de la siguiente manera:

- Pasos de 1 Hz para frecuencias entre [1, 10] Hz.
- Pasos de 2 Hz para frecuencias entre (10, 20] Hz.
- Pasos de 5 Hz para frecuencias entre (20, 50] Hz.
- Pasos de 10 Hz para frecuencias entre (50, 100] Hz.
- Pasos de 20 Hz para frecuencias entre (100, 200] Hz.
- Pasos de 50 Hz para frecuencias entre (200, 500] Hz.
- Pasos de 100 Hz para frecuencias entre (500, 1500] Hz.

Ancho de pulso de estimulación: Será configurable entre 20 y 996 μ s, con incrementos de 30.5 μ s. La tolerancia asociada se establece como el máximo entre 1 μ s y el 1 % del valor seleccionado.

Amplitud de estimulación: Debe ajustarse en el rango de 0 a 15 mA, con incrementos de 0.5 mA y una tolerancia igual al máximo entre 0.1 mA y el 5 %.

2.3. Hardware

El *hardware* del sistema se compone de dos bloques principales: uno destinado a la comunicación y otro a la estimulación. En la parte central de la Figura 2.1 se muestra un diagrama general del *hardware* del sistema.

En lo que respecta a la estimulación, el *hardware* es capaz de entregar pulsos de corriente bifásicos a través del tejido, garantizando su correcto funcionamiento durante toda la sesión de terapia. Estos pulsos son de forma rectangular y están compuestos por dos fases: una en la que la corriente circula en un sentido y otra en sentido opuesto. Asimismo, presentan un intervalo interpulso fijo y una etapa de balance pasivo. La generación de todas las señales de temporización necesarias para controlar la estimulación se realiza mediante el SoC ubicado en el bloque de comunicación. Este determina el inicio y la duración de cada fase y el intervalo entre pulsos.

La señal de estimulación es periódica, y sus parámetros —amplitud, ancho de pulso y frecuencia— son configurables. En particular, la amplitud puede variar entre 0 y 15 mA, con incrementos de 0.5 mA. El ancho de pulso puede variar entre 20 y 996 s, con incrementos de 30.5 s. Asimismo, la frecuencia de estimulación puede variar entre 1 y 1500 Hz. La forma de los pulsos se puede observar en la Figura 1.1.

Las variaciones de impedancia del tejido modifican la tensión necesaria para que la fuente de corriente mantenga la corriente de estimulación programada. Si la tensión disponible es insuficiente, la fuente puede salir de su región de regulación, comprometiendo

Capítulo 2. Descripción del sistema

tanto la precisión de la estimulación como el consumo energético del sistema. Por este motivo, el bloque de estimulación ajusta automáticamente la tensión de alimentación de la fuente de corriente en función de la impedancia de carga, manteniéndola en el valor mínimo necesario para garantizar su operación dentro de la región de regulación y optimizar el consumo de energía.

En cuanto a la comunicación, los parámetros de estimulación se configuran mediante BLE, utilizando el transceptor de radio integrado en el SoC y el *stack* correspondiente. Para ello, el dispositivo emplea una antena de 2.4 GHz, que permite la transmisión y recepción de datos. Asimismo, para el proceso de autenticación se utiliza tecnología NFC, también soportada por el SoC, la cual requiere una antena específica para la comunicación de corto alcance. Para establecer la conexión, es necesario utilizar un dispositivo móvil que cuente con soporte para BLE y NFC. A través de este, el usuario puede configurar los parámetros de estimulación y controlar el inicio y la detención de la terapia.

Por último, la alimentación del sistema se realiza mediante una fuente de tensión externa.

2.4. Firmware

En esta Sección se describe el funcionamiento general del sistema desde el punto de vista de la interacción con el usuario, la gestión de la terapia y los mecanismos de seguridad implementados. Se abordan los procesos de configuración de los parámetros de estimulación, el control de la ejecución de la terapia y las restricciones impuestas para garantizar un funcionamiento seguro y confiable del dispositivo.

El dispositivo realiza la estimulación en función de un conjunto de parámetros configurables, tales como la amplitud, la frecuencia, el ancho de pulso, la selección de los electrodos con su respectiva polaridad y la duración de la terapia. Estos parámetros se obtienen a través de una conexión BLE y se almacenan para su utilización en las sesiones de terapia programadas. Para garantizar la seguridad del sistema, el acceso a dichos parámetros —tanto para su lectura como para su modificación— requiere el establecimiento de una conexión BLE que alcance un nivel de seguridad adecuado basado en autenticación.

Una vez establecida la conexión, el usuario debe identificarse según su rol, ya que las funcionalidades disponibles dependen del tipo de usuario. Se definen dos tipos de usuarios: *programador*, con permisos para modificar todos los parámetros de estimulación e iniciar o detener la terapia, y *paciente*, con permisos limitados al encendido y apagado de la terapia, sin posibilidad de modificar la configuración establecida.

Una vez iniciada la terapia, ya sea por el programador o el paciente, esta puede finalizar debido a la detección de un error, a la finalización del tiempo de terapia programado o a una solicitud explícita del usuario a través de la aplicación o del acercamiento de un imán. Durante la ejecución de la terapia, la detención de la misma sólo se efectiviza al finalizar un ciclo completo de estimulación, con el objetivo de evitar la inyección de corriente continua (DC) en el tejido, lo cual podría resultar perjudicial. No obstante, el sistema es capaz de recibir solicitudes de detención en cualquier instante, las cuales se procesan de forma segura respetando esta restricción.

Adicionalmente, el dispositivo realiza monitoreos de forma continua, tanto durante la terapia como en reposo, incluyendo el estado de la batería y el ajuste dinámico de la tensión de alimentación del circuito de estimulación, con el fin de garantizar un funcionamiento seguro y eficiente. Estos procesamientos se llevan a cabo de manera controlada, de modo de no introducir retardos en los pulsos programados, generando *jitter*.

Finalmente, con el objetivo de mejorar la experiencia de uso y evitar la repetición del proceso de autenticación en cada conexión, el dispositivo almacena en memoria no volátil la información asociada a los usuarios previamente emparejados, incluyendo su tipo. El número máximo de usuarios totales y la cantidad por cada tipo de usuario está acotado.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 3

Plataforma de desarrollo: nRF Connect SDK y Zephyr RTOS

En este capítulo se presentan los fundamentos del conjunto de herramientas de desarrollo de software *nRF Connect SDK* y el sistema operativo de tiempo real Zephyr. Se describen los archivos principales que intervienen en el proceso de compilación, así como su rol dentro de la configuración y construcción de la aplicación. Asimismo, se explica cómo el SDK integra distintas capas de software (como librerías, controladores y stacks de protocolos) sobre la base de Zephyr, facilitando el desarrollo sobre el hardware objetivo. Además, se introduce el modelo de ejecución de Zephyr, haciendo foco en su mecanismo de planificación (*scheduling*), el cual resulta clave para la gestión de tareas concurrentes y el cumplimiento de los requisitos temporales del sistema.

3.1. nRF Connect SDK

El *nRF Connect SDK* es un conjunto de herramientas de desarrollo de software diseñado para la creación de aplicaciones de bajo consumo basadas en dispositivos de Nordic Semiconductor. Incluye el sistema operativo en tiempo real (RTOS, por sus siglas en inglés) Zephyr junto con una extensa gama de aplicaciones, ejemplos y soporte para múltiples protocolos de comunicación, tales como BLE, Bluetooth Mesh, Wi-Fi, Matter, Thread/Zigbee, LTE-M/NB-IoT/GPS y TCP/IP [6].

En la Figura 3.1 se presenta un diagrama de la cadena de herramientas (*toolchain*) del *nRF Connect SDK*, que se basa en la *toolchain* de Zephyr. En este esquema se observa un conjunto de herramientas, cada una con un rol específico en el proceso de desarrollo de una aplicación, abarcando desde la configuración del sistema hasta su compilación.

Kconfig se encarga de definir la configuración del sistema, incluyendo la habilitación de protocolos de comunicación y la selección de bibliotecas a utilizar en la aplicación. Por su parte, *Devicetree* describe el hardware del sistema (ver Sección 3.3.2).

A partir de esta información, *CMake* genera los archivos de construcción (*build files*), los cuales son utilizados por *Ninja* (una herramienta similar a *make*) para compilar el programa. Finalmente, el compilador *GCC* se encarga de generar los ejecutables.

Esta separación entre el código fuente (.c), la configuración del sistema mediante

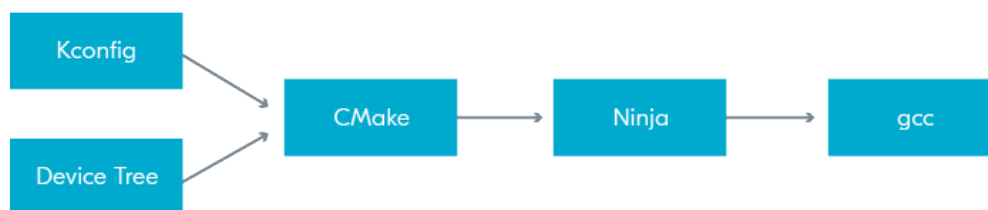


Figura 3.1: Herramientas del *nRF Connect SDK* y métodos de configuración. *Imagen extraída del curso nRF Connect SDK Fundamentals. [6].*

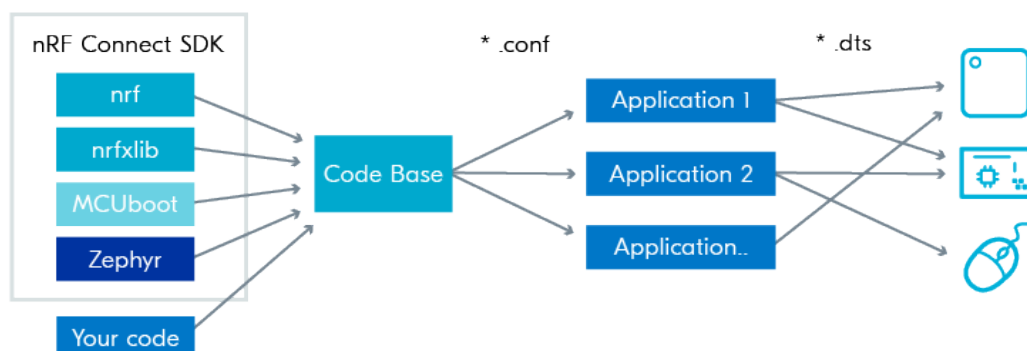


Figura 3.2: Proceso de compilación de una aplicación con *nRF Connect SDK*. *Imagen extraída del curso nRF Connect SDK Fundamentals. [6].*

Kconfig (*.conf*) y la descripción del hardware a través de *Devicetree* (*.dts*) permite reutilizar la misma aplicación en distintos dispositivos y configuraciones, requiriendo modificaciones mínimas [6].

3.2. Zephyr RTOS

Zephyr RTOS es un sistema operativo de tiempo real de código abierto, orientado a dispositivos embebidos conectados y con recursos limitados, impulsado por la *Linux Foundation*. Su objetivo es proporcionar un RTOS escalable, modular y optimizado. Su arquitectura modular y altamente configurable permite adaptarse a distintas plataformas y arquitecturas de hardware, facilitando el desarrollo de soluciones específicas según los requerimientos de cada aplicación [7].

Entre sus principales características se destacan el soporte para ejecución multihilo, con planificación tanto cooperativa como preemptiva, la gestión eficiente de memoria —generalmente estática— y la integración de controladores de dispositivos. Además, incluye soporte nativo para múltiples protocolos de comunicación, como BLE, Wi-Fi y redes basadas en IEEE 802.15.4.

Cuenta con un *scheduler* que se encarga de la planificación y sincronización temporal de las tareas (o hilos) del sistema. Asimismo, incorpora un conjunto de bibliotecas fun-

damentales y componentes de *middleware* que facilitan el desarrollo de aplicaciones [6].

3.2.1. Aplicaciones multihilo

Una aplicación multihilo permite ejecutar múltiples lógicas concurrentes mediante unidades de ejecución denominadas *hilos*. Un *hilo* es la unidad de ejecución más pequeña que compite por el tiempo de CPU.

Un *hilo* puede encontrarse en uno de los siguientes estados en cualquier momento:

- **Running:** corresponde al *hilo* que está siendo ejecutado en ese instante por la CPU. Esto implica que el *scheduler* lo ha seleccionado y ha cargado su contexto en los registros del procesador.
- **Runnable:** un *hilo* en este estado está listo para ejecutarse, es decir, no se encuentra bloqueado ni esperando recursos. Su única espera es por tiempo de CPU, por lo que puede ser seleccionado por el *scheduler*.
- **Non-runnable:** agrupa a los *hilos* que no pueden ejecutarse, ya sea porque están bloqueados esperando un recurso, sincronización con otros hilos o porque han sido suspendidos. Estos hilos no son considerados por el *scheduler*.

Además, existen *hilos del sistema* (*system threads*), que son creados automáticamente por Zephyr durante la inicialización:

- **main thread:** ejecuta la inicialización del sistema y luego llama a la función `main()` de la aplicación.
- **idle thread:** se ejecuta cuando no hay otros hilos listos, pudiendo permanecer en un bucle vacío o activar mecanismos de ahorro de energía (*power management*).

El encargado de decidir qué *hilo* se ejecuta en cada momento es el *scheduler*. Zephyr es un RTOS *tickless*, lo que significa que el *scheduler* no se ejecuta de forma periódica, sino en respuesta a eventos denominados *rescheduling points*. Un *rescheduling point* es el instante en el cual el sistema evalúa qué *hilo* debe ejecutarse a continuación.

Estos eventos ocurren cuando se producen cambios en el conjunto o en las condiciones de los hilos en estado **runnable**, de modo que podría modificarse cuál es el hilo de mayor prioridad listo para ejecutarse [8]. Algunos ejemplos relevantes, utilizados en esta aplicación, son los siguientes:

- Cuando un *hilo* invoca la función `k_yield()`, cediendo voluntariamente la CPU y pasando de estado **running** a **runnable**.
- Cuando un *hilo* se desbloquea debido a la liberación de un *semáforo*, *mutex* o *alert*, pasando de estado **non-runnable** a **runnable**.

A cada *hilo* se le asigna una *prioridad*. En el caso de los *hilos preemptibles*, utilizados en esta aplicación, la prioridad se representa mediante un entero no negativo (por defecto

```
CONFIG_BT=y
CONFIG_BT_DEVICE_NAME="NEUROSOC"
```

Figura 3.3: Configuración de la pila Bluetooth Low Energy.

entre 0 y 15), donde los valores más bajos indican mayor prioridad. El `main thread` tiene prioridad 0, mientras que el `idle thread` tiene prioridad 15.

Para el *firmware* desarrollado se definieron cuatro hilos de ejecución. En primer lugar, un hilo de alta prioridad (1), denominado *thread_application*, encargado del funcionamiento general del sistema y, en particular, de la ejecución de la máquina de estados asociada a la terapia. Adicionalmente, se implementaron tres hilos de menor prioridad (8): *thread_ble*, *thread_housekeeping* y *thread_processing*, encargados respectivamente de la gestión del emparejamiento, el disparo de mediciones del sistema y el procesamiento de eventos. Esta asignación responde a que *thread_application* ejecuta las tareas críticas desde el punto de vista temporal, mientras que los restantes realizan funciones auxiliares sin requerimientos temporales tan estrictos. Además, estos últimos comparten la misma prioridad debido a que ninguno presenta una precedencia funcional sobre los demás y las tareas que ejecutan son acotadas, por lo que no resulta necesario establecer una jerarquía adicional entre ellos. El funcionamiento de estos hilos se explica en detalle en la Sección 5.2.

3.3. Elementos de una aplicación nRF Connect SDK

Como se mencionó previamente, una aplicación está compuesta por diversos elementos, que abarcan desde la configuración del sistema hasta su implementación, los cuales son utilizados por el sistema de compilación para generar el ejecutable final.

3.3.1. Archivo de configuración de aplicación: `prj.conf`

El archivo *prj.conf* se utiliza para definir la configuración de la aplicación, especificando los módulos de software y servicios del kernel a emplear, así como sus parámetros. Cada opción de configuración se expresa en la forma `CONFIG_<nombre>=<valor>`, donde **nombre** identifica la funcionalidad a configurar y **valor** puede ser, por ejemplo, *y* o *n*, indicando si la opción está habilitada o deshabilitada, respectivamente. De esta manera, se controla la inclusión de los distintos módulos en el proceso de compilación y su disponibilidad en la aplicación [9].

Por ejemplo, en el archivo *prj.conf* de la aplicación se incluyeron las configuraciones mostradas en la Figura 3.3, las cuales habilitan el *stack* de BLE y establecen el nombre con el que el dispositivo se anuncia durante el proceso de *advertising*. El resto de las configuraciones seleccionadas para el *firmware* desarrollado se describen con mayor detalle en la Sección 5.5.2.

Además del archivo de configuración de la aplicación, existen configuraciones por defecto asociadas a la placa objetivo. Estas configuraciones son heredadas automáticamente por la aplicación; sin embargo, los valores definidos en *prj.conf* tienen prioridad

```
/{
    electrodes {

        compatible = "gpio-leds";

        e1_neg: e1_neg {
            gpios = <&gpio0 19 (GPIO_PULL_DOWN | GPIO_ACTIVE_HIGH)>;
        }
    }
}
```

Figura 3.4: Fragmento del *Devicetree* de la aplicación que define el nodo correspondiente al electrodo *e1_neg*.

y pueden sobrescribir las configuraciones de la placa.

3.3.2. Devicetree

Devicetree es una estructura de datos jerárquica utilizada por Zephyr RTOS para describir el *hardware* de una aplicación, representada en archivos con extensión *.dts*. Está compuesta por un conjunto de nodos organizados jerárquicamente, donde cada nodo puede contener propiedades y nodos hijos.

En la Figura 3.4 se muestra un fragmento del archivo *.dts* generado para la aplicación desarrollada, donde se muestra un nodo raíz */*, un nodo hijo *electrodes* y, a su vez, un nodo hijo de este último, *e1_neg*. A los nodos se les pueden asignar etiquetas (*labels*) para facilitar su referencia desde otras partes del *Devicetree*. Asimismo, cada nodo posee una dirección que permite ubicarlo dentro de la estructura jerárquica [10].

En este fragmento también se observa que el nodo *e1_neg* contiene una propiedad denominada *gpios*. En *Devicetree*, los valores de tipo numérico se representan entre *<>*. Las propiedades pueden ser de distintos tipos, incluyendo valores enteros, cadenas de texto, arreglos y referencias a otros nodos.

En este caso, el valor de la propiedad *gpios* indica que se utiliza el pin 19 del puerto GPIO 0 para esa señal. Las banderas *GPIO_PULL_DOWN* y *GPIO_ACTIVE_HIGH* especifican, respectivamente, que el pin se configura con una resistencia interna de *pull-down* y que la señal asociada es activa en nivel lógico alto.

Las restantes configuraciones relevantes incorporadas al *Devicetree* para la aplicación desarrollada se describen con mayor detalle en la Sección 5.5.1.

Devicetree bindings

Los *Devicetree bindings* en Zephyr son archivos en formato YAML que especifican cómo deben interpretarse los nodos del *Devicetree*. En particular, definen las propiedades que cada nodo puede o debe contener, el tipo de dato asociado a cada una y si son obligatorias u opcionales. Estas definiciones se vinculan a los nodos mediante la propiedad

`compatible`, la cual indica el dispositivo o controlador asociado. En general, todos los nodos que representan dispositivos deben definir esta propiedad.

En el fragmento mostrado en la Figura 3.4, se utiliza la propiedad `compatible = "gpio-leds"`, correspondiente a un controlador genérico provisto por Zephyr para gestionar pines GPIO configurados como salidas digitales. Aunque originalmente está pensado para el control de LEDs, este *binding* puede emplearse para cualquier señal digital de salida, como en este caso, donde se utiliza para controlar uno de los electrodos del sistema.

Estas especificaciones permiten validar la estructura del *Devicetree* y garantizan que cada nodo incluya la información necesaria para su correcto funcionamiento dentro del sistema. De esta forma, se proporciona un mecanismo estructurado y verificable para extender el soporte de hardware en el sistema operativo Zephyr.

Los archivos de *bindings* se ubican típicamente en `<install_path>\zephyr\dts\bindings`. En el caso de definir un *binding* propio para un dispositivo no soportado, este debe colocarse en dicha ubicación o en un directorio equivalente dentro del proyecto, de forma que el sistema de compilación pueda detectarlo [10].

Device Driver Model

El *device driver model* de Zephyr tiene como objetivo principal abstraer el acceso al hardware mediante una interfaz genérica, permitiendo que las aplicaciones interactúen con los periféricos sin necesidad de conocer los detalles específicos de su implementación. En este modelo, los controladores (*drivers*) se encargan de configurar y gestionar los dispositivos físicos a bajo nivel, mientras que las aplicaciones acceden a ellos a través de APIs estandarizadas.

Para utilizar un periférico, la aplicación obtiene un puntero al dispositivo (`const struct device *`) mediante macros como `DEVICE_DT_GET()`, las cuales se basan en la información definida en el *Devicetree*. En el caso del periférico GPIO, existen macros como `GPIO_DT_SPEC_GET()`, que permiten obtener una estructura con la configuración del pin, y funciones como `gpio_is_ready_dt()`, que verifican si el dispositivo fue inicializado correctamente y está listo para su uso. De esta forma, se logra una interacción estructurada, portable y eficiente entre el software y el hardware, favoreciendo la reutilización de código y la escalabilidad del desarrollo embebido [10].

Por ejemplo, para controlar el GPIO `e1_neg` presentado en la Figura 3.4, durante la etapa de inicialización se obtiene su configuración mediante la siguiente declaración:

```
static struct gpio_dt_spec e1_neg_gpio =  
GPIO_DT_SPEC_GET(DT_NODELABEL(e1_neg), gpios);
```

La macro `DT_NODELABEL(e1_neg)` permite obtener una referencia en tiempo de compilación al nodo del *Devicetree* identificado con la etiqueta `e1_neg`. A su vez, `GPIO_DT_SPEC_GET()` genera una estructura de tipo `gpio_dt_spec` que contiene el dispositivo GPIO asociado, el número de pin y las banderas de configuración definidas en el *Devicetree*. A partir de esta estructura, es posible configurar y controlar el pin utilizando la API estándar de GPIO de Zephyr.

Devicetree overlays

No es recomendable modificar el *Devicetree* base directamente. En su lugar, se utilizan superposiciones de *Devicetree* (*overlays*). Estas consisten en archivos con extensión `.overlay`, cuyo nombre suele estar asociado a la placa objetivo, y que permiten modificar o extender la configuración definida originalmente.

Un archivo *overlay* debe incluir únicamente los nodos y propiedades que se desean agregar o modificar. Generalmente, se ubica en el directorio raíz de la aplicación, donde el sistema de compilación lo detecta automáticamente y lo incorpora durante el proceso de construcción [10].

Para la PCB desarrollada en este proyecto, se creó un archivo de superposición de *Devicetree* en el que se definieron las configuraciones y adaptaciones necesarias para la correcta utilización del hardware diseñado. Entre ellas se encuentra la definición del nodo mostrado en la Figura 3.4, correspondiente a uno de los electrodos controlados mediante GPIO.

3.3.3. CMake

En el SDK de *nRF Connect*, todas las aplicaciones se estructuran como proyectos *CMake*. Esto implica que la propia aplicación define la configuración y el proceso de compilación, incluyendo tanto el código de la aplicación como el sistema Zephyr y las bibliotecas asociadas.

En particular, *CMake* se encarga de interpretar las instrucciones del proyecto (qué archivos incluir, qué opciones de compilación utilizar) y, a partir de ello, generar los archivos de construcción (*build files*) que serán utilizados por una herramienta externa, como *Ninja*, para compilar el sistema completo.

El archivo `CMakeLists.txt` es el principal archivo de configuración de *CMake* en el proyecto, donde se especifican las directivas necesarias para la construcción de la aplicación [10].

3.3.4. Sysbuild

Sysbuild es un sistema de compilación de alto nivel que facilita la gestión de compilaciones multiimagen. Una compilación multiimagen se utiliza en sistemas que requieren generar más de un *firmware*, como en el caso de dispositivos con múltiples núcleos [11].

Un ejemplo de esto es el SoC nRF5340, que cuenta con dos núcleos: el *Application Core* y el *Network Core*. En este dispositivo, para habilitar funcionalidades como BLE es necesario incluir una imagen adicional que se ejecuta en el *Network Core*.

Sysbuild utiliza Kconfig de manera similar a `prj.conf`, pero en este caso las configuraciones se definen en archivos como `sysbuild.conf` o `Kconfig.sysbuild`. Para seleccionar una imagen para el *Network Core*, se deben habilitar las opciones correspondientes, como se muestra en la Figura 3.5.

```
source "${ZEPHYR_BASE}/share/sysbuild/Kconfig"

config NRF_DEFAULT_IPC_RADIO
    default y

config NETCORE_IPC_RADIO_BT_HCI_IPC
    default y
```

Figura 3.5: Configuraciones en `sysbuild.conf` para incluir una imagen para el *Network Core* [11].

Capítulo 4

Desarrollo de Hardware

Para el desarrollo del *hardware* del proyecto, *Integer Holdings Corporation* proporcionó los esquemáticos de un neuroestimulador, junto con los requerimientos del sistema, el *firmware* original y una *test board* que implementa dicho diseño. El sistema original incluye dos fuentes de corriente, ocho electrodos, comunicación RF y un circuito de carga de batería.

El neuroestimulador original se basa en una arquitectura con dos microcontroladores: uno destinado a la ejecución de la terapia, correspondiente a un microcontrolador RISC de 16 bits de ultra bajo consumo, optimizado para aplicaciones embebidas portátiles, y otro encargado de la comunicación.

A partir de esta plataforma, se propuso reemplazar ambos microcontroladores por el SoC nRF5340, evaluando su capacidad para asumir ambas funciones sin degradar el desempeño del sistema. Para ello, fue necesario realizar ajustes sobre el *hardware* original, con el fin de adaptarlo al uso del nuevo dispositivo.

Con el objetivo de facilitar la etapa de pruebas e integrar los nuevos componentes, se diseñó una placa de circuito impreso (PCB) que incluye el SoC y los elementos adicionales requeridos. Tanto el esquemático actualizado como el diseño del *layout* del PCB fueron desarrollados en el marco de este proyecto.

En comparación con el sistema original, se introdujeron las siguientes modificaciones:

- Sustitución de los dos microcontroladores por el nRF5340.
- Incorporación de un DAC y de una antena NFC.
- Implementación de una única fuente de corriente en lugar de dos.
- Reducción del número de electrodos de ocho a cuatro.
- Eliminación del circuito de carga de batería.
- Eliminación de la comunicación RF.
- Eliminación de los diodos de protección asociados a los terminales de estimulación.
- Remoción del acelerómetro.
- Sustitución del *reed-switch* por un botón que cumple su misma función.

4.1. Descripción del Sistema

En la siguiente Sección se presenta el *hardware* del sistema, describiendo las funciones de los distintos bloques que componen el diseño. Asimismo, se analizan las principales modificaciones introducidas respecto al diseño original, incluyendo los cambios de componentes y su impacto en la implementación.

El sistema es capaz de generar la corriente de estimulación requerida a través de los electrodos hacia el tejido en ambos sentidos, garantizando en todo momento un funcionamiento correcto. De esta forma, se evitan condiciones que puedan comprometer la seguridad (*safety*) del paciente, al cumplir con la amplitud y los tiempos programados. Asimismo, el sistema cuenta con los elementos necesarios para establecer la comunicación con el dispositivo móvil que se utiliza para programar.

En la Figura 4.1 se presenta un diagrama de bloques de alto nivel del *hardware* del sistema. Éste se compone de tres bloques principales: control y comunicación, alimentación y estimulación.

El bloque de alimentación es responsable de la generación y regulación de las distintas tensiones requeridas, incluyendo aquellas necesarias para la operación de la fuente de corriente y de las llaves analógicas. Este bloque garantiza condiciones de suministro adecuadas para el correcto funcionamiento de los demás módulos.

El bloque de control y comunicación está encargado de la generación de las señales de comando que gobiernan el comportamiento del sistema, así como de la implementación de la comunicación inalámbrica mediante BLE, permitiendo la configuración y programación del dispositivo de manera remota.

Finalmente, el bloque de estimulación está compuesto por la fuente de corriente y los electrodos. Estos últimos se conectan a través del tejido del cuerpo, estableciendo el camino de conducción de la corriente. La corriente es inyectada y absorbida por la fuente de corriente a través de los electrodos.

Aquellos elementos representados con borde de línea punteada corresponden a componentes provistos por el cliente. Dentro de esta categoría se incluyen los circuitos pertenecientes al diseño original, tales como la matriz de conexiones programable hacia los electrodos, la fuente de corriente y el convertidor *boost*, junto con sus componentes asociados. Asimismo, se incluye el SoC, el cual, si bien no formaba parte del diseño original, fue especificado por el cliente.

Por otro lado, los bloques representados con borde de línea continua corresponden a los componentes que fueron modificados o incorporados durante el desarrollo. Entre ellos se encuentra la fuente de alimentación, que en el diseño original estaba basado en una batería recargable con su correspondiente circuito de carga inalámbrica, mientras que en la implementación desarrollada se optó por una fuente de alimentación externa. Asimismo, se incorporó un convertidor digital-analógico (DAC) externo, cuya selección fue realizada en el marco del proyecto, dado que el nRF5340 no dispone de este periférico integrado, a diferencia del microcontrolador original.

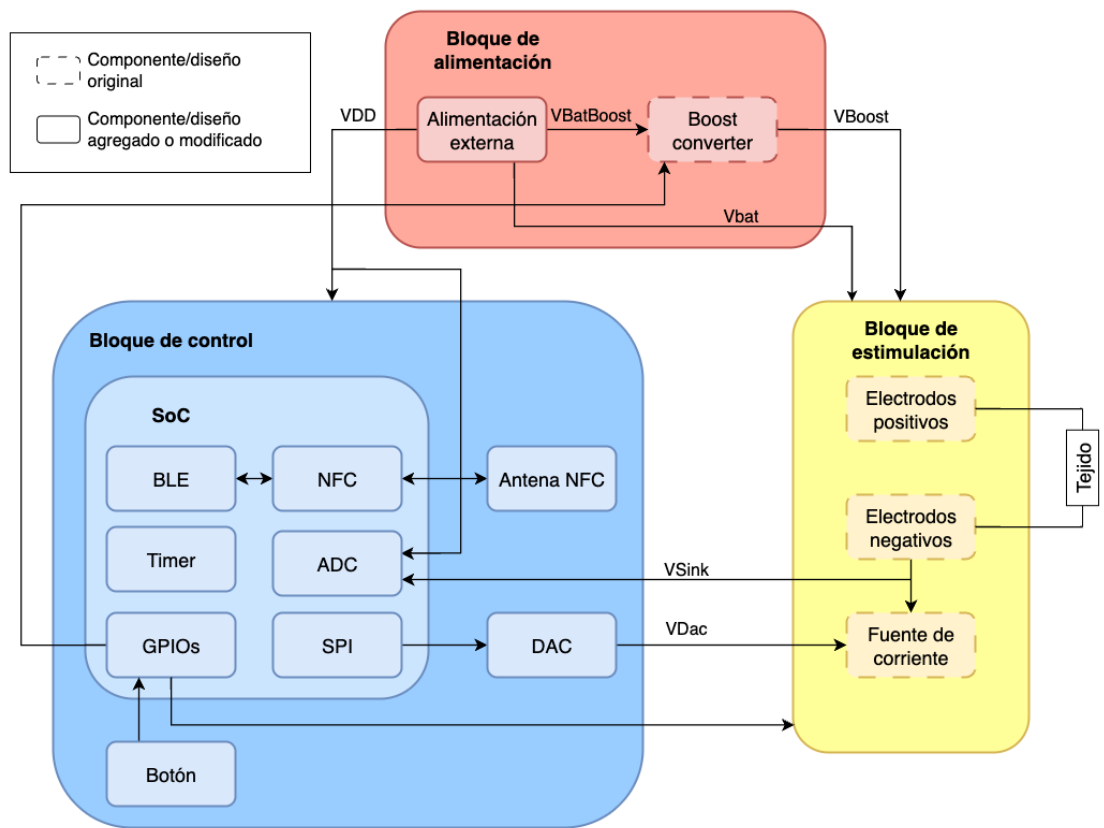


Figura 4.1: Diagrama de bloques del dispositivo. Se presentan los tres bloques principales que conforman el sistema, así como la interacción entre ellos. Los bloques con línea punteada corresponden al diseño original provisto por *Integer*, mientras que los de línea continua fueron agregados o modificados en el marco del proyecto. En particular, se eliminaron los dos microcontroladores, el *reed-switch* y la antena RF del diseño original, y se incorporaron el SoC, el DAC, un botón que sustituye el funcionamiento del *reed-switch* y la antena NFC.

4.1.1. Bloque de estimulación

La estimulación se realiza mediante la circulación de corriente a través del tejido, utilizando el circuito mostrado en la Figura 4.2. Como se observa en el diagrama, el circuito es alimentado a partir de la tensión V_{anode} , la cual puede provenir tanto de la batería como de la salida del convertidor *boost*, en función de la tensión requerida en V_{sink} para garantizar el correcto funcionamiento de la fuente de corriente.

Asimismo, los distintos electrodos pueden configurarse dinámicamente como ánodo o cátodo, permitiendo controlar el sentido de circulación de la corriente a través del tejido.

En las siguientes subsecciones se describen en detalle la interfaz hacia los electrodos y el ajuste de tensión en la fuente de corriente.

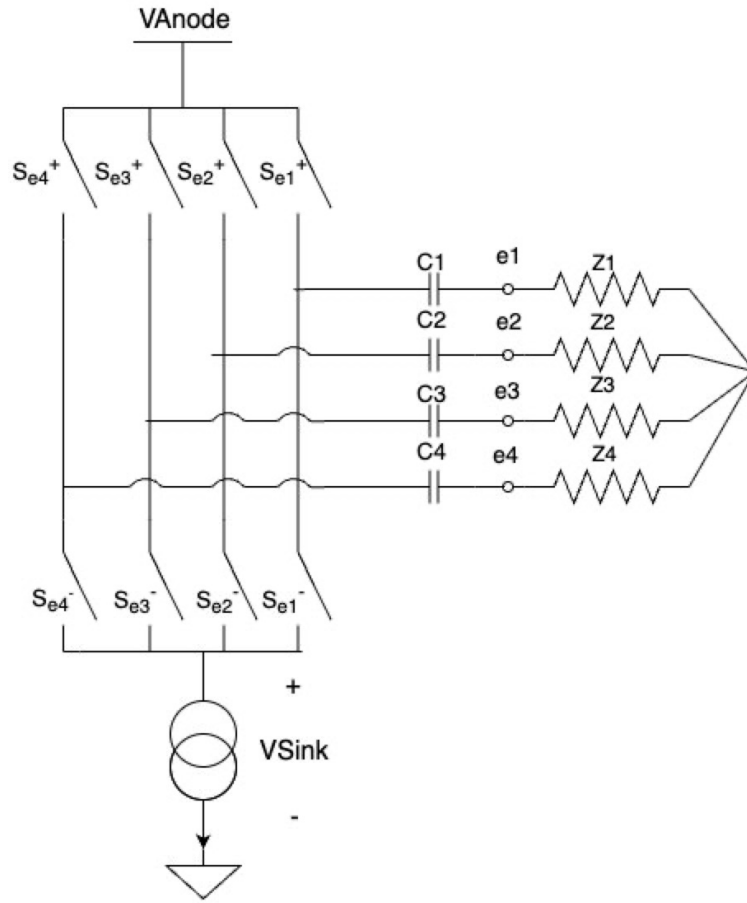


Figura 4.2: Esquema del circuito de estimulación. El sistema es alimentado a partir de la tensión V_{anode} , a la cual se conecta el electrodo configurado como positivo. La corriente circula a través del tejido hacia los electrodos configurados como negativos y luego es conducida por la fuente de corriente operando en modo *sink*. La tensión V_{sink} corresponde a la caída de tensión en la fuente de corriente. La alimentación V_{anode} puede provenir de V_{bat} o de V_{boost} , dependiendo del nivel de tensión requerido.

Interfaz hacia los electrodos

El electrodo configurado como ánodo se conecta a la tensión V_{anode} , la cual puede corresponder tanto a la tensión de la batería como a la salida del convertidor *boost*. Esta conexión se observa en la Figura 4.2, donde el ánodo corresponde al electrodo positivo.

Por su parte, el terminal configurado como cátodo actúa como colector de la fuente de corriente, fijando así la tensión V_{sink} .

Bajo esta configuración, la corriente circula desde el ánodo hacia el cátodo a través del tejido, el cual puede modelarse como una resistencia, permitiendo así la aplicación controlada de los pulsos de estimulación. La decisión de modelarlo como una resistencia parte de que es la forma estandarizada de modelar la carga de un neuroestimulador definida en el standard ISO 14708-3, sub-cláusula 6.101 [3].

Fuente de corriente

La fuente de corriente presenta una topología tipo *current sink* controlada por tensión. Su función es fijar la corriente que el circuito entrega al tejido durante la estimulación, para lograr la onda descrita en la Sección 5.4.1.

La activación de la fuente se realiza mediante una señal de control proveniente del SoC, mientras que el valor de corriente se establece a partir de una tensión analógica. Para esta implementación, se utilizó la tensión de salida del DAC, denotada como V_{DAC} , y un divisor resistivo con constante $\frac{1}{5}$. La relación entre la corriente de estimulación deseada y la tensión aplicada está dada por:

$$I_{\text{estimulación}} = \frac{V_{DAC}}{5} \cdot \frac{1}{20 \Omega} \quad (4.1)$$

La tensión de alimentación de la fuente de corriente depende del valor de V_{anode} y de la impedancia del tejido. En esta configuración, el terminal negativo ($V_{cathode}$) actúa como retorno de la fuente de corriente.

Para garantizar el correcto funcionamiento del circuito, es necesario asegurar una diferencia de tensión suficiente entre los bornes de la fuente de corriente (V_{sink} en la Figura 4.2), de modo que ésta opere en su región activa y pueda regular adecuadamente la corriente de salida. En la configuración implementada esta tensión mínima es de 600 mV.

Además, la eficiencia del circuito de estimulación se define como:

$$\eta = \frac{P_{stim}}{P_{stim} + P_{sink}} \quad (4.2)$$

donde

$$P_{stim} = RI^2 \quad (4.3)$$

$$P_{sink} = V_{sink}I \quad (4.4)$$

De esta forma, al disminuir V_{sink} se reduce P_{sink} y, en consecuencia, mejora la eficiencia del circuito.

Por lo tanto, resulta deseable minimizar V_{sink} en la medida de lo posible, sin dejar de cumplir la condición de que supere la tensión mínima necesaria para que la fuente de corriente opere correctamente en su región de regulación. En función de esto, se ajusta la tensión V_{anode} , como se detalla en secciones posteriores.

4.1.2. Bloque de control y comunicación

El bloque de control y comunicación está compuesto por cuatro elementos: el SoC nRF5340, un DAC, la antena NFC y un botón. Tal como se observa en el diagrama, la utilización del SoC nRF5340 no fue una decisión tomada en el marco del proyecto, sino un requerimiento inicial. En consecuencia, el sistema fue implementado sobre esta plataforma. Por otro lado, el DAC fue seleccionado e integrado durante el desarrollo del proyecto, debido a la ausencia de este periférico en el nRF5340.

Cabe destacar que, previo a la selección del DAC, se consideró la alternativa de utilizar el periférico PWM provisto por el SoC para generar la tensión de control de la

Capítulo 4. Desarrollo de Hardware

fuentes de corriente. Para ello, se planteó el uso de un filtro pasabajos a la salida del PWM, con el objetivo de obtener una señal analógica adecuada.

El diseño de dicho filtro debía cumplir con múltiples restricciones. Por un lado, la frecuencia de corte debía ser significativamente menor que la frecuencia del PWM, a fin de atenuar sus componentes de alta frecuencia. Por otro, debía ser suficientemente mayor que la máxima frecuencia de estimulación, asociada al menor ancho de pulso considerado (20 μ s), para evitar la distorsión de la señal útil. Adicionalmente, para cubrir el rango de tensión de control de la fuente de corriente (0 mV a 300 mV en pasos de 10 mV), se requería una resolución de al menos 30 niveles. Dado que la resolución del PWM está determinada por la relación entre la frecuencia de reloj (hasta 16 MHz) y la frecuencia del PWM, esto implicaba una frecuencia de operación del orden de 500 kHz.

Sin embargo, la proximidad entre estas frecuencias dificultó el diseño de un filtro que satisficiera simultáneamente los requisitos de atenuación y fidelidad de la señal, lo que motivó la evaluación de alternativas como el uso de un DAC externo.

nRF5340

Para su incorporación en el diseño, se optó por utilizar el módulo *BT40F* [12], el cual integra el nRF5340 junto con una antena para comunicación BLE. Adicionalmente, se incluyó un conector externo que permite la conexión de una antena NFC.

En cuanto a la adquisición de señales, el periférico ADC del SoC se utiliza para medir la tensión de alimentación de la fuente de corriente, permitiendo así monitorear su valor y ajustar dinámicamente la tensión de salida del convertidor *boost* para garantizar un funcionamiento adecuado con el menor consumo posible. Asimismo, la medición de la tensión de batería se realiza mediante un canal interno del propio SoC, sin requerir conexiones externas.

Por último, los pines de propósito general (GPIO) se emplean para el control de diversas señales del sistema, incluyendo las líneas de configuración del convertidor *boost*, la habilitación de la fuente de corriente, y el manejo de las llaves analógicas que determinan la configuración de los electrodos como ánodo o cátodo.

Convertidor digital-analógico (DAC)

Se incorporó al sistema un DAC MCP4821, encargado de controlar la amplitud de los pulsos de corriente generados. Este dispositivo se comunica con el SoC mediante una interfaz SPI, a través de la cual se configuran sus registros de operación y se establece el valor de salida analógica correspondiente.

El DAC forma parte de la etapa de generación de corriente, permitiendo ajustar la tensión de control de la fuente y, en consecuencia, definir con precisión la amplitud de los pulsos de estimulación generados.

Antena NFC

Para implementar el proceso de autenticación *Out-of-Band* (OOB) descrito en la Sección 5.3.6, fue necesaria la incorporación de una antena NFC que permite el intercambio de datos entre el dispositivo y el programador.

Botones

Se incorporaron tres botones en el diseño. Uno de ellos se utiliza para reiniciar el SoC, otro cumple la función de *reed-switch*, permitiendo la detención segura de la terapia, y el tercero se emplea como botón auxiliar para tareas de depuración (*debugging*).

4.1.3. Bloque de alimentación

Este bloque es capaz de suministrar los distintos niveles de tensión requeridos por el circuito, garantizando condiciones de alimentación adecuadas para el correcto funcionamiento de todo el sistema.

Alimentación externa

La alimentación del microcontrolador nRF5340 de Nordic Semiconductor y del DAC se realiza directamente desde la fuente de alimentación externa, la cual se configura con una tensión nominal de 3.8 V. Asimismo, esta misma fuente de alimentación se utiliza para energizar, dentro del bloque, un convertidor elevador de tensión (*boost converter*), encargado de generar niveles de tensión superiores a los provistos por la batería. Debido a la presencia de un diodo en la entrada de alimentación del SoC, se produce una caída de tensión asociada, por lo que se espera que la tensión V_{DD} del mismo sea aproximadamente de 3.1 V.

Boost converter

Por otro lado, el convertidor *boost* permite ajustar la tensión de salida V_{out} en función del valor de la resistencia conectada a la entrada de realimentación (*feedback*). La relación entre estos parámetros está dada por:

$$V_{out} = V_{REF} \frac{R_1 + R_2}{R_2} \quad (4.5)$$

donde:

- V_{REF} es la tensión de referencia interna del convertidor, en este caso 1.225 V.
- R_2 es la resistencia interna entre la entrada de *feedback* (FB) y tierra.
- R_1 es la resistencia conectada entre V_{out} y la entrada FB.

De esta forma, variando el divisor resistivo conformado por R_1 y R_2 , es posible modificar la tensión de salida del convertidor. En la implementación desarrollada, este ajuste se realiza mediante llaves controladas por señales digitales del SoC, las cuales permiten seleccionar diferentes configuraciones del divisor resistivo.

En la Tabla 4.1 se presentan los valores de V_{out} obtenidos en función de las combinaciones de dichas señales.

Tabla 4.1: Tensión de salida del convertidor *boost* en función de las señales de control.

Boost_ctrl_0	Boost_ctrl_1	Boost_ctrl_2	V_{out}
0	0	0	5.0 V
0	0	1	7.5 V
0	1	0	10.0 V
0	1	1	12.5 V
1	1	1	15.0 V

4.2. Componentes

4.2.1. System on Chip nRF5340

Como se mencionó al comienzo del capítulo, la elección del SoC respondió a un requerimiento inicial provisto por el cliente. En particular, se utilizó el nRF5340 de Nordic Semiconductor, un SoC de ultra bajo consumo que integra dos procesadores del tipo Arm Cortex-M33 completamente programables [13].

El dispositivo cuenta con una amplia variedad de periféricos, entre los que se incluyen temporizadores, GPIO, ADC, SPI y NFC, utilizados en la implementación de la aplicación desarrollada. Asimismo, incorpora un transceptor multiprotocolo de 2.4 GHz que soporta BLE, entre otros protocolos inalámbricos.

El núcleo de aplicación es de propósito general, lo que le permite ejecutar la lógica principal del sistema, mientras que el núcleo de comunicación presenta un conjunto de funcionalidades reducido y está optimizado para operar con un consumo de energía ultra bajo.

En la Figura 4.3 se presenta un diagrama de bloques simplificado del SoC, donde se muestran los dos núcleos y la interacción entre ellos.

Doble núcleo

Como se mencionó, el sistema cuenta con una arquitectura de doble núcleo, compuesta por un *Application Core* y un *Network Core*.

El *Application Core* se encarga de ejecutar la lógica principal del sistema, incluyendo el procesamiento de datos y la aplicación en general. En el caso de BLE, el stack se distribuye entre ambos núcleos, implementándose el Host en el *Application Core*.

Por su parte, el *Network Core* está dedicado a la comunicación de más bajo nivel. En el contexto de BLE, se encarga de implementar el Controller, incluyendo la capa de enlace (*Link Layer*) y el control de la radio.

De esta forma, se aíslan las tareas críticas de comunicación de la lógica de aplicación, evitando interferencias entre ambas y asegurando el cumplimiento de los requerimientos temporales del protocolo.

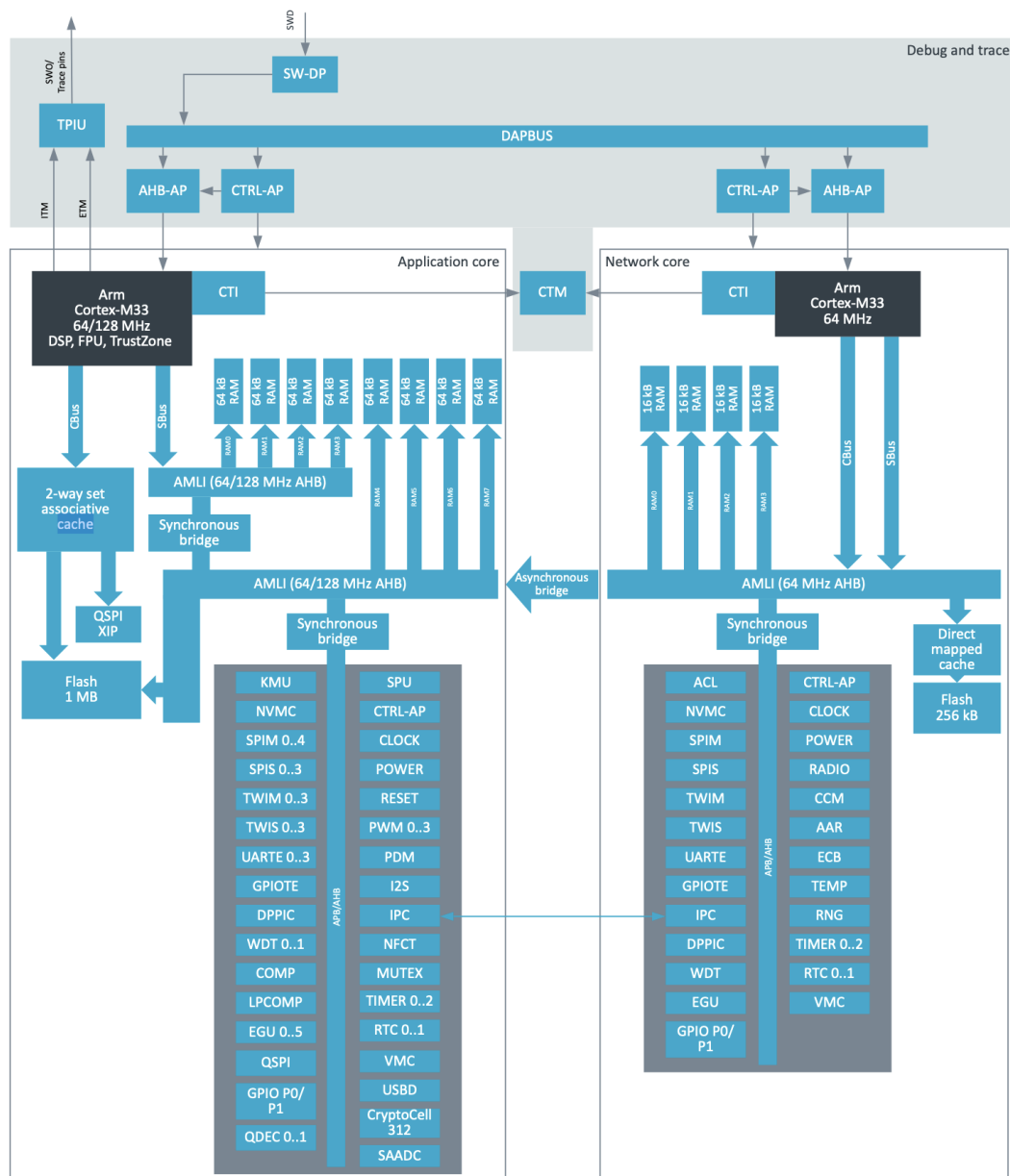


Figura 4.3: Diagrama de bloques simplificado del nRF5340. Imagen extraída de la Especificación del Producto del nRF340 [13]

Capítulo 4. Desarrollo de Hardware

NFCT

El periférico NFCT (*Near Field Communication Tag*) implementa un dispositivo del tipo *listening device* NFC-A, según lo definido por el NFC Forum [14]. Este opera a una frecuencia de entrada de 13.56 MHz y una velocidad de 106 kbps. Se encarga del ensamblado y desensamblado de tramas NFC-A conforme a dicha especificación, e incorpora funcionalidades como resolución automática de colisiones, verificación de redundancia cíclica (CRC) y control de paridad.

Durante la transmisión, los datos de las tramas se transfieren directamente desde la memoria RAM. El sistema es notificado mediante eventos cada vez que una trama completa es transmitida o recibida. Las tramas entrantes son desensambladas automáticamente y su contenido de datos es transferido a la memoria RAM mediante el mecanismo EasyDMA.

El periférico dispone de dos pines para la conexión de la antena, los cuales incorporan mecanismos de protección frente a campos de alta intensidad.

Timer

Se trata de un temporizador de propósito general que puede operar en dos modos: timer o contador. En este proyecto se utilizó exclusivamente el modo timer.

El periférico utiliza como fuente de reloj el HFCLK (High Frequency Clock) y dispone de un *prescaler* de 4 bits que permite dividir la frecuencia de entrada. A partir de esta configuración, la frecuencia base del temporizador queda determinada por:

$$f_{TIMER} = \frac{16 \text{ MHz}}{2^{PRESALER}} \quad (4.6)$$

Internamente, el temporizador puede operar con dos señales derivadas del HFCLK: PCLK16M (16 MHz) y PCLK1M (1 MHz). La selección entre estas fuentes se realiza automáticamente en función de la frecuencia configurada mediante el prescaler. En particular, cuando la frecuencia del temporizador es menor a 1 MHz, se utiliza PCLK1M con el objetivo de reducir el consumo de energía. La resolución temporal del timer está determinada por el período de un tick de f_{TIMER} , lo que define la precisión con la que pueden generarse eventos temporales.

Además, el sistema PPI (*Programmable Peripheral Interconnect*) permite que un evento generado por el temporizador desencadene automáticamente una acción en otro periférico, sin intervención de la CPU.

En la Figura 4.4 se muestra el sistema de control de relojes utilizado por el nRF5340.

ADC

El SoC utilizado cuenta con el periférico SAADC, cuyo diagrama de bloques se muestra en la Figura 4.5. Cuenta con ocho entradas provenientes de pines analógicos, así como la posibilidad de utilizar la alimentación (VDD) como entrada. El rango de tensión de entrada se extiende desde 0 V hasta VDD, y se pueden configurar hasta ocho canales en modo *single ended* o hasta cuatro en modo *differential*. La conversión analógica a digital puede ser disparada tanto desde software como utilizando un canal PPI.

4.2. Componentes

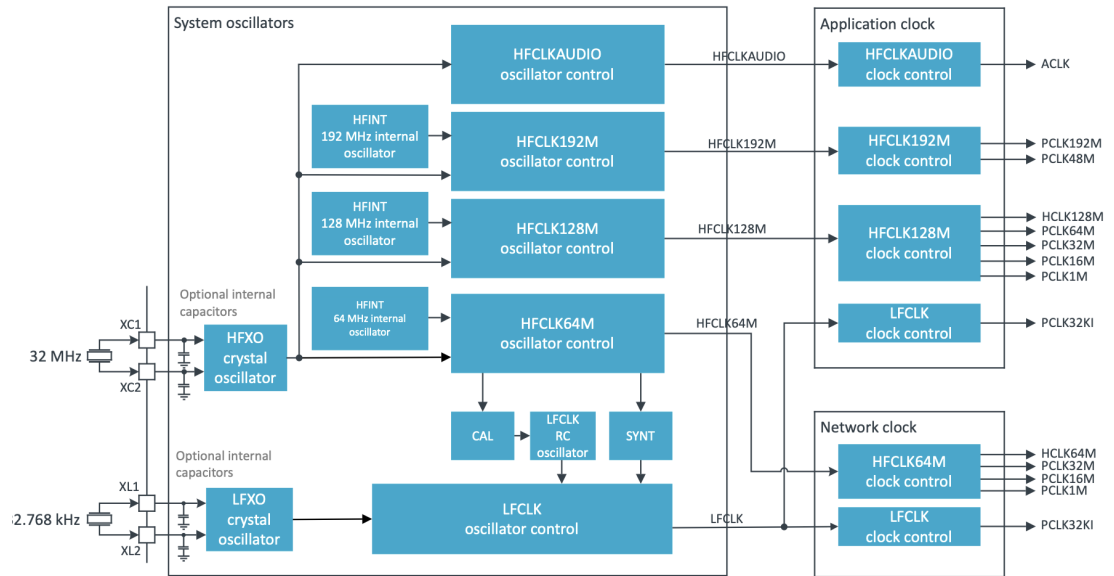


Figura 4.4: Sistema de control de relojes del SoC. El timer utiliza los relojes PCLK1M y PCLK16M provenientes del HFCLK. *Imagen extraída de la Especificación del Producto del nRF340 [13]*

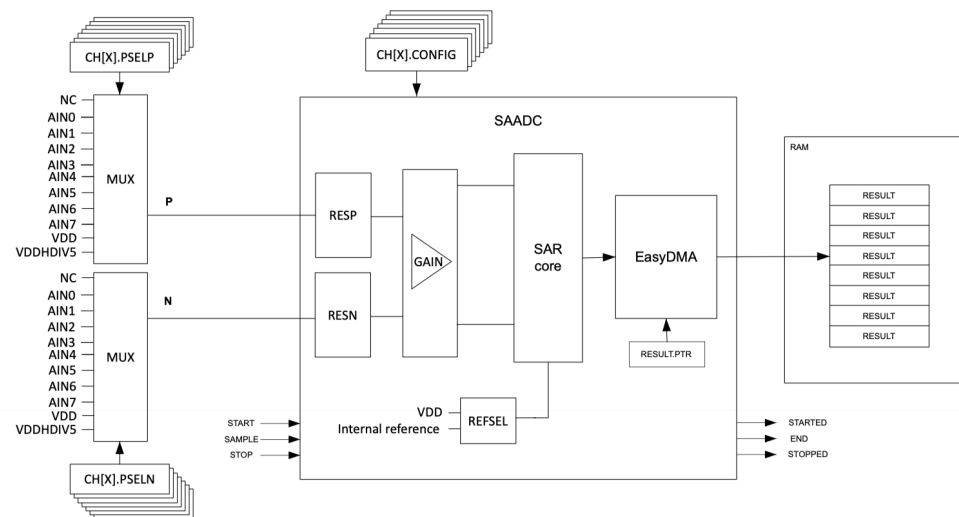


Figura 4.5: Diagrama de bloques simplificado del ADC. *Imagen extraída de la Especificación del Producto del nRF340 [13].*

SPI

El periférico SPI permite la comunicación con múltiples dispositivos esclavos mediante el uso de una señal CS (Chip Select) independiente para cada uno de ellos. Este periférico implementa EasyDMA, lo que permite acceder directamente a la memoria RAM sin intervención de la CPU, reduciendo la carga de procesamiento. Soporta los modos 0 a 3 de SPI, los cuales difieren en la polaridad (CPOL) y la fase (CPHA) de la señal de reloj. Además, soporta la funcionalidad D/CX, utilizada por ciertos dispositivos esclavos. Esta consiste en una línea adicional que permite diferenciar entre bytes de datos y de comandos. La configuración de los pines del SPI debe realizarse antes de habilitar el periférico, ya que, una vez habilitado, estos registros no pueden modificarse.

GPIOs

El SoC cuenta con pines de *General Purpose Input/Output* (GPIO) agrupados en puertos, donde cada puerto puede disponer de hasta 32 pines. El nivel lógico alto de un GPIO corresponde al voltaje presente en el pin VDD.

Los GPIO pueden ser asignados para ser utilizados por cualquiera de los dos núcleos del sistema. Una vez asignado un pin a un *core*, su configuración y estado —incluyendo si opera como entrada, salida o en modo analógico— solo pueden ser controlados y leídos por dicho *core*.

Los GPIO permiten configurar diversas características, tales como la dirección (entrada o salida) y la capacidad de manejo de corriente (*drive strength*). Además, es posible habilitar resistencias internas de *pull-up* y *pull-down*.

Asimismo, los pines permiten desconectar el buffer de entrada cuando no se utilizan como entradas, lo que contribuye a reducir el consumo de energía.

4.2.2. Elección del convertor digital-analógico (DAC)

Se seleccionó e integró un convertidor digital-analógico (DAC) externo. El proceso de selección se basó en una serie de requerimientos funcionales y de integración, definidos a partir de las características del sistema original y de las restricciones del diseño. A continuación, se detallan los principales requerimientos considerados, junto con su justificación:

- **Dos canales de salida (*dual channel*):** El sistema original cuenta con dos fuentes de corriente independientes, por lo que se requiere un DAC con al menos dos canales para mantener dicha funcionalidad, asignando un canal a cada fuente.
- **Bajo consumo de potencia:** Se priorizaron dispositivos con bajo consumo energético, en línea con los objetivos generales de eficiencia del sistema.
- **Encapsulado compatible con montaje superficial (SMD):** Se requirió un encapsulado del tipo *LFCSP*, adecuado para la integración en el PCB final. Se descartaron encapsulados *through-hole* por no ser compatibles con un diseño compacto.

4.3. Configuración para pruebas iniciales

- **Tamaño reducido:** El tamaño del encapsulado fue un factor determinante, con el objetivo de minimizar el área ocupada en el PCB y contribuir a la reducción de dimensiones del sistema.
- **Facilidad de prototipado:** Para la etapa de validación inicial con el *Development Kit* (DK) y la *test board*, se consideró necesario que el DAC estuviera disponible en al menos una de las siguientes variantes:
 - Encapsulado *TSSOP* o *through-hole*, que permitiera su uso en placas de desarrollo o protoboard.
 - Módulo *breakout*, que facilitara las pruebas preliminares sin requerir un diseño de PCB dedicado.

Teniendo en cuenta estas características, se optó por utilizar el DAC MCP4821 [15]. Este dispositivo presenta un tamaño reducido de aproximadamente $3\text{ mm} \times 3\text{ mm}$ en su versión SMD, comparable con el de otras alternativas evaluadas. En estado activo, su consumo de corriente es del orden de los microamperios, resultando inferior al de otros dispositivos similares.

La comunicación con el microcontrolador se realiza mediante interfaz SPI, lo que permite una transmisión de datos rápida, determinística y con bajo uso de recursos.

En estado apagado, el dispositivo presenta un consumo típico de $0.5\text{ }\mu\text{A}$, también menor que el de otras opciones consideradas, lo cual resulta favorable para esta aplicación.

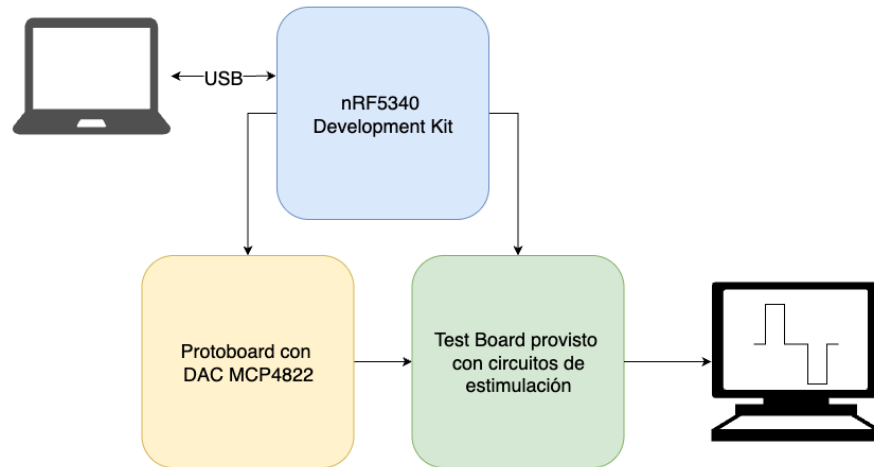
Asimismo, el componente se encuentra disponible en encapsulados PDIP y MSOP. Esto permitió, por un lado, su incorporación en el diseño final mediante encapsulado SMD, y por otro, su utilización en protoboard durante las etapas de validación y mediciones iniciales gracias a su versión *through-hole*.

4.3. Configuración para pruebas iniciales

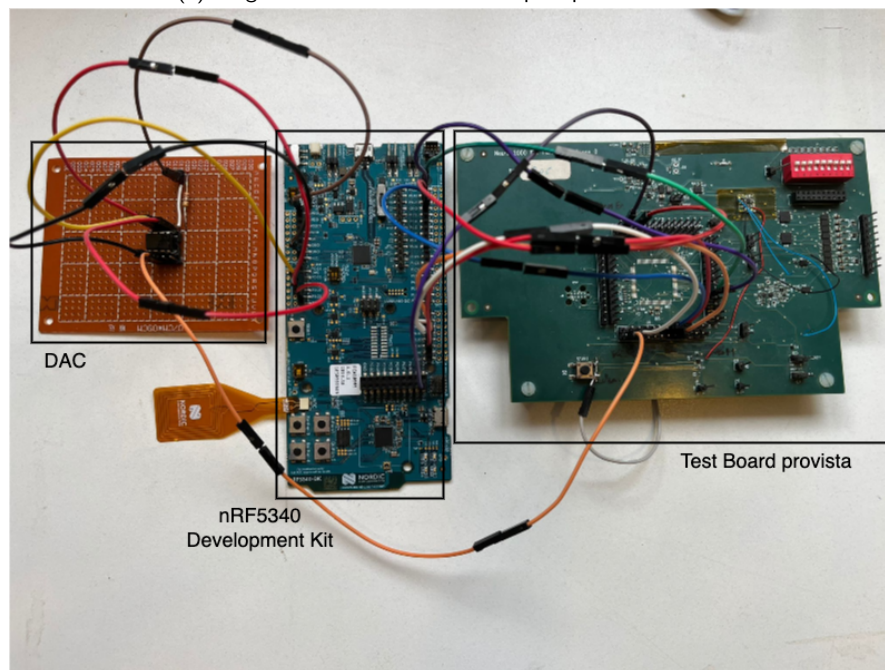
Como se mencionó anteriormente, *Integer Holdings* proporcionó una *test board* que incluye los circuitos de estimulación, pero no incorpora un microcontrolador. En su lugar, las señales que serían generadas por este se encuentran disponibles a través de conectores externos, lo que permite su control mediante un dispositivo externo.

Las primeras pruebas del *firmware* se realizaron utilizando un *setup* como el que se muestra en la Figura 4.6. Se empleó la placa de desarrollo nRF5340 DK [16] para validar el *firmware* diseñado, tanto mediante la interfaz UART (para la visualización de *logs*) como a través de sus GPIO, los cuales se utilizaron para controlar los circuitos de estimulación presentes en la *test board*. Las señales generadas fueron observadas mediante un osciloscopio.

Asimismo, el DAC fue conectado al DK mediante la interfaz SPI, y este a su vez a la fuente de corriente contenida en la *test board*.



(a) Diagrama de Hardware utilizado para pruebas iniciales.



(b) Fotografía del setup utilizado para las pruebas iniciales. Se observa el nRF5340dk, protoboard con DAC conectado y Test board provista.

Figura 4.6: Diagrama y foto del setup utilizado para las pruebas iniciales del firmware y hardware en conjunto.

4.4. Diseño del PCB

En el siguiente capítulo se describe el proceso para el desarrollo de la PCB utilizada para la validación del nuevo sistema diseñado.

Dado que la placa fue concebida con fines de prueba, se definieron dimensiones que permitieran un espaciado adecuado entre componentes y facilitaran el acceso a los distintos nodos del circuito para la realización de mediciones. En este sentido, se buscó un compromiso entre compacidad y accesibilidad. Las dimensiones finales de la placa fueron de $80\text{ mm} \times 70\text{ mm}$. Debido a estas dimensiones y a la cantidad de componentes y señales de control involucradas, se optó por un diseño multicapa de cuatro capas.

Con el objetivo de minimizar el ruido parásito generado por el propio circuito, se adoptó una estrategia de separación física entre los distintos bloques funcionales como se muestra en la Figura 4.8, considerando especialmente aquellos nodos más sensibles al ruido y aquellos subsistemas más propensos a generarlo. En particular, se decidió ubicar el *boost converter* alejado del DAC y de la fuente de corriente de los electrodos, dado que su principio de funcionamiento basado en conmutación introduce perturbaciones. La capa superior y la inferior se destinaron principalmente al ruteo de señales digitales de control, comunicaciones (por ejemplo, SPI) y señales analógicas. Las dos capas internas se utilizaron como planos: uno de alimentación, que distribuye las distintas tensiones generadas por el bloque de alimentación, y otro de tierra, que cubre la mayor parte del área de la PCB. Este último resulta útil para minimizar interferencias electromagnéticas y reducir el ruido en el sistema.

Finalmente, todos los componentes se ubicaron en la cara superior de la placa, facilitando tanto el ensamblado como las tareas de medición y depuración del sistema. En la Figura 4.8a se muestra la vista superior del modelo 3D de la PCB, mientras que en la Figura 4.8b se presenta la vista inferior de la misma.

Para el diseño de la PCB se utilizó el software de desarrollo *Altium Designer*, haciendo uso de la licencia de estudiante. Esta herramienta permitió, a partir de los esquemáticos del circuito del sistema, generar un *layout*, posibilitando la distribución física de los componentes, considerando sus encapsulados y pines, así como el ruteo de las interconexiones entre los mismos.

Otra ventaja de este tipo de entorno CAD es la disponibilidad de herramientas de verificación que permiten validar la correcta asignación de señales y el cumplimiento de las reglas de diseño (Design Rule Check, DRC) en el *layout*, previo a su fabricación. Para reducir la probabilidad de errores, una vez finalizado el diseño en *Altium Designer* se realizó una verificación adicional utilizando un software visor de archivos de fabricación (CAM) denominado *ViewMate*. Esta herramienta permite visualizar, inspeccionar y validar los archivos *Gerber* y de perforado (*drill*) antes de su envío al fabricante de PCBs, en este caso JLCPCB.

4.4.1. Fabricación del PCB

Para la construcción de la PCB se eligió mandar a fabricar a JLCPCB. De igual modo, el soldado de los componentes fue realizado por dicha empresa, ya que se contaban con algunos componentes de tipo QNF y LGA que dificultan una colocación y soldado

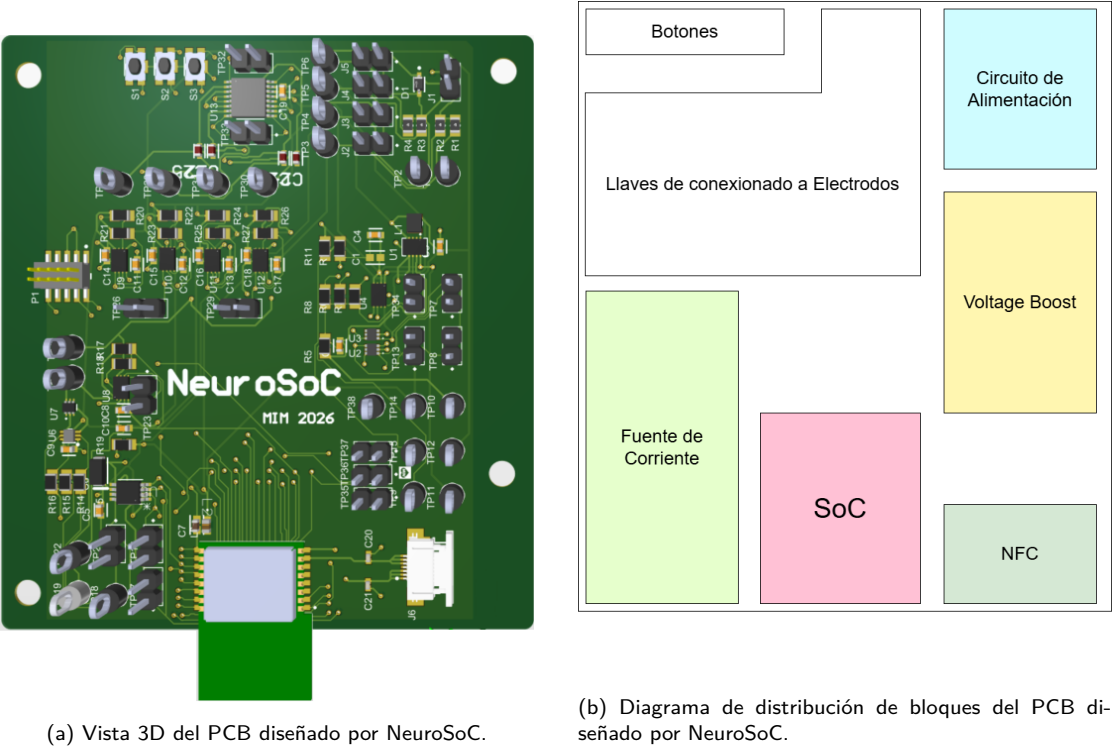


Figura 4.7: PCB Diseñado

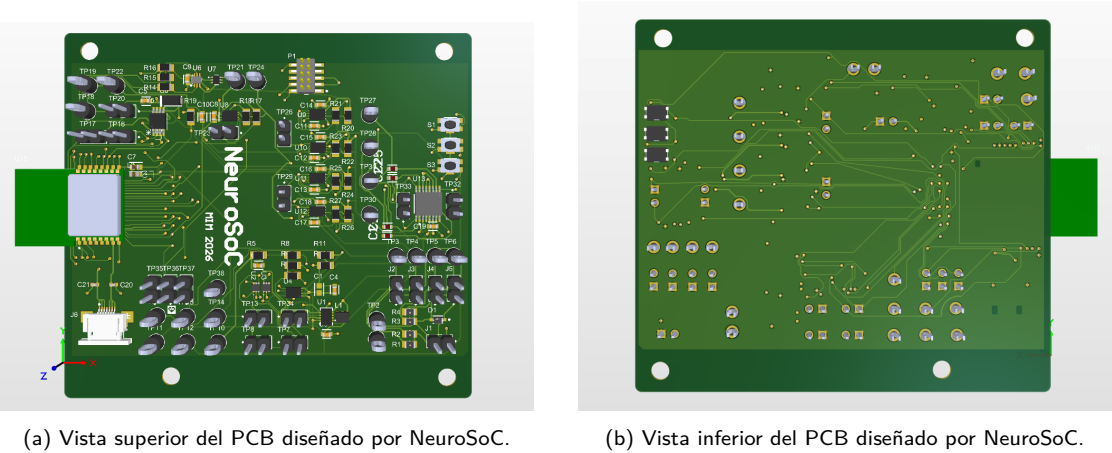


Figura 4.8: PCB Diseñado

manual. Para esto se proporcionaron a la empresa los archivos de diseño generados por el CAD.

Capítulo 5

Diseño e implementación de Firmware

5.1. Introducción

En este capítulo se presenta el desarrollo del *firmware* del proyecto. En primer lugar, se ofrece una visión general de la arquitectura implementada, incluyendo el diagrama de capas del sistema, el flujo general de ejecución y su organización en hilos. A continuación, se describen los dos subsistemas principales: el encargado de la comunicación mediante BLE y el responsable de la estimulación, junto con sus respectivos módulos asociados. Finalmente, se detallan los archivos de configuración y compilación empleados en la aplicación desarrollada sobre Zephyr.

5.2. Arquitectura del Firmware

La arquitectura del *firmware* se estructura en tres niveles como se muestra en la Figura 5.1: una capa independiente del *hardware* que contiene la lógica de aplicación, una capa dependiente del *hardware* que implementa los controladores específicos, y una capa de *hardware* que representa los periféricos utilizados. Esta separación permite la modularidad y portabilidad del sistema. La aplicación general integra los distintos módulos del *firmware* para proporcionar una solución completa que permite gestionar de manera adecuada el inicio y la finalización de la terapia, la comunicación vía BLE y el procesamiento de medidas relevantes.

En la Figura 5.2 se presenta el flujo general de ejecución del sistema. El mismo comienza con una etapa de inicialización, la cual una vez completada, el sistema queda a la espera de una solicitud de conexión mediante BLE. En caso de que el dispositivo no se encuentre previamente *bonded*, se realiza el proceso de emparejamiento a través de NFC. Luego de establecer la conexión, el sistema permanece en estado de espera hasta recibir un evento de inicio de terapia.

Ante la recepción de dicho evento, se verifica la validez de los parámetros de estimulación, que debieron ser configurados previamente vía BLE. Si estos no son correctos, el sistema retorna al estado de espera. En caso contrario, se da inicio a la ejecución de la terapia, representada en el diagrama como el bloque de estimulación.

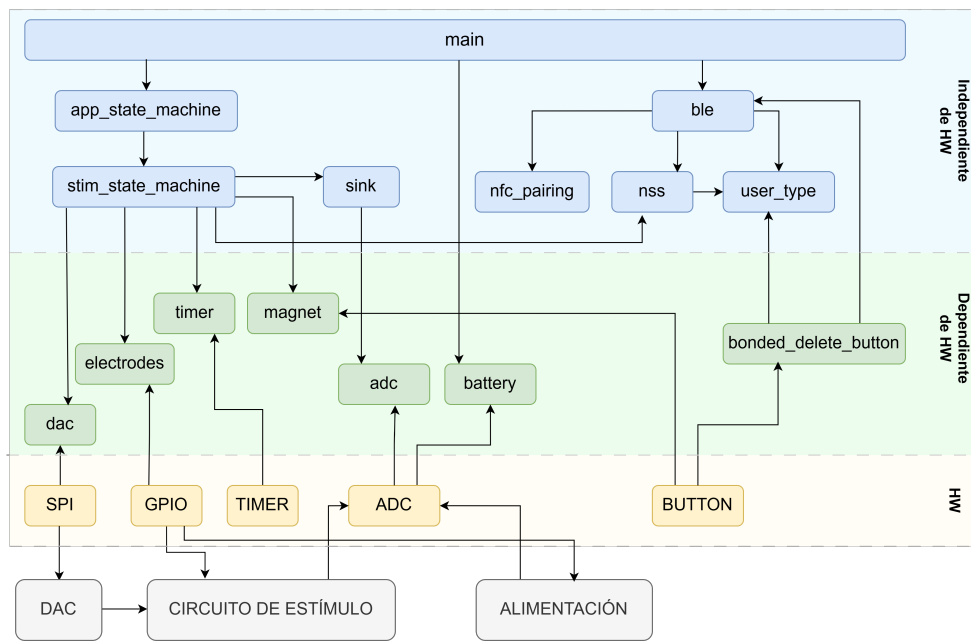


Figura 5.1: Diagrama de la estructura en capas del *firmware* desarrollado.

Durante la ejecución de la terapia, se distingue el estado de *Passive Balance*, en el cual se habilita el procesamiento de mediciones, tales como el nivel de batería y la tensión en la fuente de corriente.

Finalmente, se evalúan las condiciones de finalización de la terapia. En caso de que alguna de estas se cumpla, se detiene la estimulación y el sistema retorna al estado inicial de espera. La máquina de estados de estimulación se describe en detalle en la Sección 5.4.1.

Para la organización del *firmware*, se definieron cuatro hilos de ejecución cuya coordinación se realiza mediante el uso de semáforos. El uso de hilos con distintas prioridades permite un control adecuado de la ejecución de los diversos procesos, garantizando la precisión temporal requerida por la terapia y evitando interrupciones ocasionadas por la ejecución de otras tareas del sistema. En particular, esta estrategia permite reducir tanto la latencia de respuesta ante eventos críticos como la variabilidad en los tiempos de ejecución (*jitter*), asegurando que acciones como el inicio y la finalización de cada fase del pulso de estimulación ocurran en los instantes previstos. De este modo, se minimizan errores en parámetros temporales como el período, la frecuencia y el ancho de pulso, preservando el comportamiento determinista del sistema.

Los cuatro hilos definidos son: *thread_application*, *thread_ble*, *thread_processing* y *thread_housekeeping*, donde el primero posee mayor prioridad (1), mientras que los restantes operan con menor prioridad (8). Se asignó la misma prioridad a estos últimos debido a que ninguno presenta requisitos temporales más exigentes que los demás. Además, las tareas que realizan son acotadas por lo que no resulta necesario establecer una jerarquía adicional entre ellos. El funcionamiento de estos hilos se explica en las secciones a continuación.

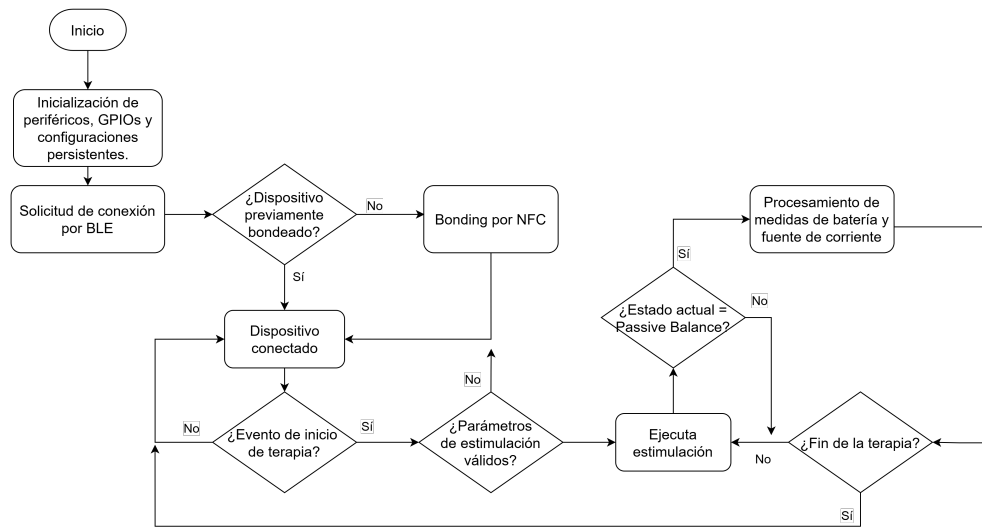


Figura 5.2: Diagrama de flujo del funcionamiento general del sistema, mostrando las etapas de inicialización, conexión mediante BLE, validación de parámetros y ejecución de la terapia de estimulación.

En la Figura 5.3a se muestra la jerarquía de hilos del sistema. Como se puede observar, la ejecución de la aplicación comienza con una etapa de inicialización que se realiza en el programa principal (*main*) y, hasta que esta no finaliza, no se permite la ejecución de ningún hilo. Esto se logra mediante el uso de semáforos: cada hilo requiere la liberación de un semáforo para comenzar su ejecución (*start_sem*), los cuales se habilitan una vez completada la etapa de inicialización. Una vez liberados los semáforos de inicio, los hilos se ejecutan de acuerdo con sus prioridades.

La etapa de inicialización incluye la configuración de todos los GPIOs y periféricos utilizados durante la ejecución del programa, tales como temporizador, ADC y SPI, así como la inicialización de las señales digitales y analógicas, la habilitación de la comunicación BLE y la configuración de las estructuras de eventos que serán gestionadas durante la ejecución del sistema. Asimismo, se realiza la carga de las configuraciones almacenadas en memoria no volátil (Flash), incluyendo la estructura de datos persistente que contiene la información de los dispositivos previamente emparejados y su correspondiente tipo de usuario, en caso de existir registros almacenados, la cual se describe en detalle en la Sección 5.3.8.

La ejecución de los hilos de prioridad 8 depende del estado del hilo de aplicación (*thread_application*), el cual presenta dos estados: *Idle* y *Stimulating*, cuyo funcionamiento se explica en detalle en la Sección 5.2.1. En el estado *Idle*, estos hilos pueden ejecutarse libremente. En cambio, durante el estado *Stimulating*, donde se ejecuta la máquina de estados de la terapia, permanecen bloqueados y únicamente se habilitan durante la fase de *Passive Balance*, mediante la liberación de un semáforo (*passive_balance_sem*). Este enfoque permite aprovechar el tiempo, ya que la duración de esta fase es mayor a las demás y no se requieren restricciones estrictas de temporización.

Por otra parte, el hilo de *housekeeping* presenta una condición adicional para su ejecución: además de depender de *passive_balance_sem*, permanece bloqueado a la es-

pera de *housekeeping_sem*, el cual es liberado periódicamente cada 2 segundos por un temporizador.

Adicionalmente, la Figura 5.3b ilustra la secuencia temporal de ejecución de los hilos y las transiciones entre los estados **runnable** y **non-runnable**. En ella se observa que, inicialmente, se ejecuta el hilo de mayor prioridad, *thread_application*. Cuando este invoca *k_event_wait()* para esperar la ocurrencia de un evento de inicio de terapia, pasa del estado **running** al estado bloqueado, cediendo así el procesador a los hilos de menor prioridad.

Posteriormente, al recibirse el evento que inicia la terapia, *thread_application* retoma la ejecución y entra en el estado *Stimulating*. Durante esta etapa, los hilos de menor prioridad permanecen en estado **non-runnable**, ya que se encuentran bloqueados a la espera de la liberación del semáforo *passive_balance_sem*. Una vez liberado dicho semáforo, estos hilos vuelven al estado **runnable** y pueden ser ejecutados nuevamente por el sistema, durante la duración de *Passive Balance*.

En las secciones siguientes se describe en detalle el funcionamiento de cada uno de los hilos que componen el sistema.

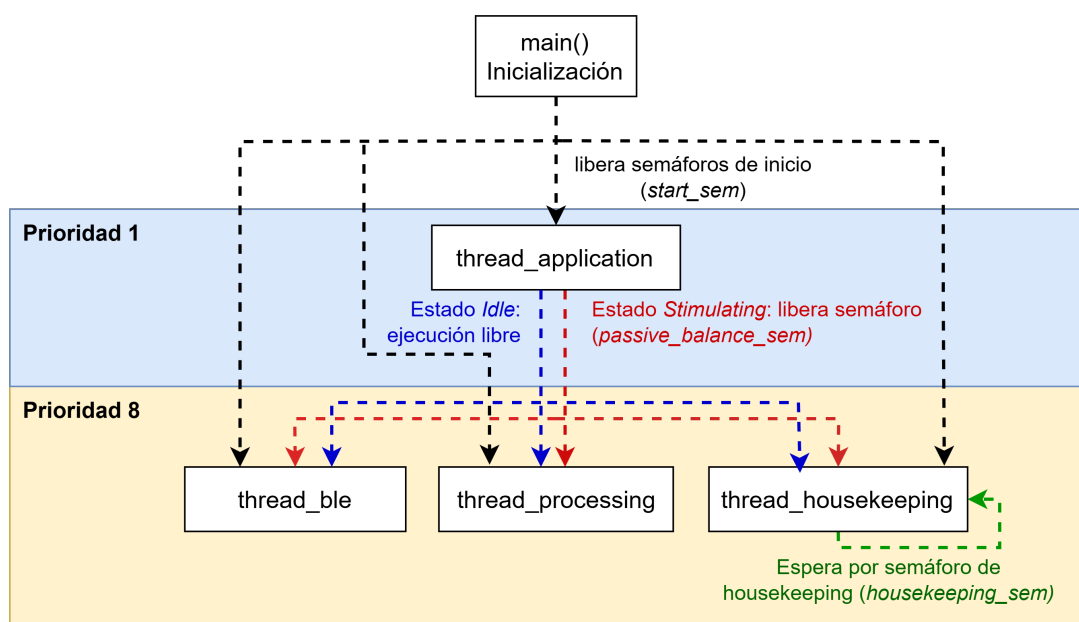
5.2.1. *thread_application*

Este hilo contiene la ejecución principal de la aplicación y se estructura como una máquina de estados compuesta por dos estados principales, *Idle* y *Stimulating*, cuya interacción se muestra en la Figura 5.4.

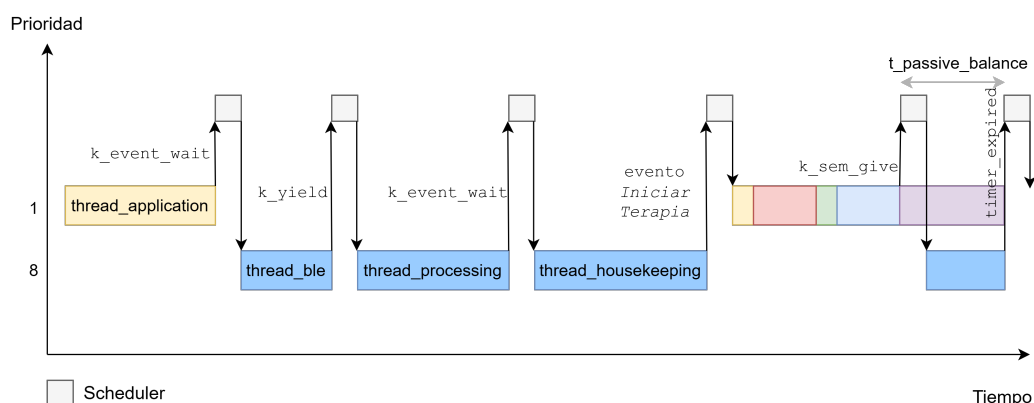
Para la implementación de la máquina de estados se utilizó el *State Machine Framework* de Zephyr, debido a su simplicidad y a que esta máquina no presenta requerimientos estrictos de temporización.

En la máquina de estados se distinguen los siguientes estados:

- **Idle:** En este estado el sistema permanece a la espera de la recepción de un evento de inicio de terapia enviado vía BLE. Una vez recibido dicho evento, se verifica la existencia de parámetros de estimulación válidos. En caso afirmativo, se realiza la transición al estado *Stimulating*; en caso contrario, el sistema permanece en el estado *Idle* hasta que se reciba un nuevo evento de terapia junto a parámetros válidos.
- **Stimulating:** En este estado se ejecuta la máquina de estados de estimulación, tal como se describe en la Sección 5.4.1. El sistema permanece en este estado hasta que se produce la detención de la terapia, la cual puede deberse a las siguientes causas: la recepción de un evento de detención vía BLE, la detección de un imán, la ocurrencia de un evento de batería baja o la expiración del temporizador de duración de la terapia, según lo descrito en la Sección 5.4.1. Al salir de este estado, se ejecuta una función de salida encargada de desactivar todos los periféricos y módulos y señales del circuito utilizados durante la terapia, tales como el *sink*, el *boost* y sus señales asociadas, los electrodos y el DAC.



(a) Diagrama de la jerarquía de hilos del sistema.



(b) Diagrama de tiempos de los hilos.

Figura 5.3

5.2.2. thread_ble

Este hilo realiza *polling* para detectar la presencia de una señal NFC asociada al proceso de *bondeado*.

La conexión BLE y el intercambio de datos pueden llevarse a cabo de manera concurrente con la ejecución de la aplicación, ya que estas tareas son gestionadas por el *Network Core*. Los eventos del stack BLE corresponden a notificaciones generadas por la pila de protocolos ante cambios en el estado de la comunicación, tales como conexiones, desconexiones, recepción de datos o procesos de autenticación. Estos eventos son propagados al *Application Core*, donde se ejecutan los *callbacks* definidos por la aplicación. Sin

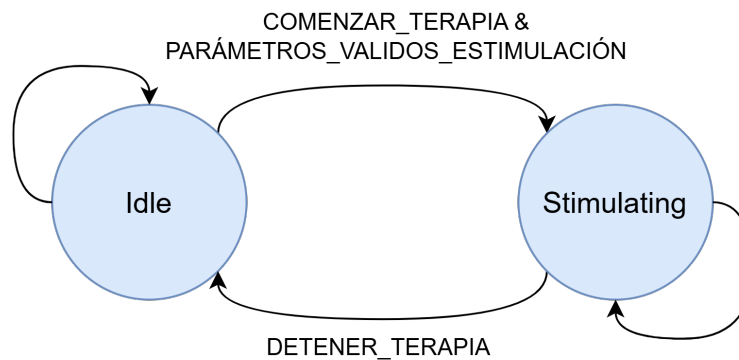


Figura 5.4: Diagrama de estados de la aplicación.

embargo, en el caso de requerir autenticación mediante NFC, es necesaria la ejecución de este hilo, que se encarga de procesar los eventos asociados a dicho mecanismo.

5.2.3. thread_housekeeping

Este hilo espera la liberación de un semáforo que se activa periódicamente cada dos segundos mediante un temporizador. Su objetivo es disparar las mediciones de batería realizadas por el ADC, así como indicar si se debe tomar una medida de la tensión de la fuente de corriente durante la primera fase del próximo pulso de estimulación.

5.2.4. thread_processing

El objetivo de este hilo es realizar el procesamiento de las distintas mediciones y eventos del sistema. Para ello, permanece en espera de tres tipos de eventos: medición de la tensión de la fuente de corriente, medición de la tensión de la batería y solicitud de eliminación de dispositivos previamente emparejados.

El procesamiento de la medición de la tensión de la fuente de corriente implica la actualización de los campos `IncreaseCounter`, `DecreaseCounter` y `VoltageStep` en función del valor medido, con el objetivo de ajustar la tensión del *boost* durante la estimulación, de acuerdo con el criterio detallado en la Sección 5.4.4.

El procesamiento de la medición de la tensión de la batería consiste en verificar si esta supera el umbral de batería baja y, en caso afirmativo, generar un evento de batería baja que provoque la finalización de la terapia.

Finalmente, ante la recepción de un evento asociado al botón que solicita la eliminación de dispositivos emparejados, el procesamiento consiste en borrar la información almacenada tanto en memoria volátil como en memoria no volátil (Flash), incluyendo la correspondencia entre dirección pública BLE y tipo de usuario.

5.3. Subsistema BLE

El neuroestimulador asume el rol de dispositivo periférico, siendo responsable de la transmisión de paquetes de *advertising* para anunciar su presencia. Por su parte, el dispositivo móvil actúa como central, iniciando la conexión mediante el envío de una solicitud correspondiente y continuando con el proceso de emparejamiento, tal como se describe en la Sección 5.3.6.

5.3.1. Proceso de conexión

En la Figura 5.5 se presenta el diagrama de estados del módulo de comunicación implementado. Tal como se detalla en las secciones siguientes, el dispositivo requiere autenticación mediante intercambio *Out-of-Band* (*OOB*) utilizando NFC, y alcanzar un nivel de seguridad 4 (*LE Secure Connections*) para poder mantener la conexión y habilitar el acceso a las características de los servicios implementados. Asimismo, el sistema permite la conexión de un único dispositivo a la vez y, en ausencia de conexiones activas, permanece en estado de *advertising*. Mantener el dispositivo en estado de *advertising*, si bien incrementa levemente el consumo, garantiza la disponibilidad del dispositivo.

Una vez inicializado, el dispositivo comienza en estado de *advertising*. Al recibir una solicitud de conexión por parte de un dispositivo central, la conexión se establece automáticamente, de acuerdo con el funcionamiento del protocolo BLE, sin posibilidad de rechazo explícito. A partir de este punto, el sistema transita al estado de conexión no autenticada.

Desde este estado, existen dos posibles transiciones: avanzar hacia la fase de autenticación o finalizar la conexión. Para iniciar la autenticación, es necesario que se haya realizado previamente un intercambio de datos OOB mediante NFC. Si no se realiza dicho intercambio, el dispositivo permanece conectado hasta que expira un temporizador interno que exige alcanzar el nivel de seguridad 4, tras lo cual se produce la desconexión.

En caso de haberse realizado el intercambio OOB, el sistema avanza a la fase de autenticación. Este proceso puede resultar fallido, ya sea por errores en la verificación de autenticación o por no alcanzar el nivel de seguridad requerido, lo que provoca el retorno al estado de *advertising*. En caso contrario, si la autenticación es exitosa, se establece una conexión autenticada con nivel de seguridad 4, habilitando el acceso a los servicios ofrecidos por el dispositivo.

La salida del estado de conexión puede producirse mediante una solicitud explícita de desconexión o por expiración del *supervision timeout*.

Finalmente, una vez completado exitosamente el proceso de autenticación, los dispositivos quedan vinculados (*bonded*). En conexiones posteriores, no es necesario repetir el intercambio OOB mediante NFC, ya que las claves previamente intercambiadas son almacenadas en ambos dispositivos. Esto permite que la conexión se establezca directamente con autenticación y nivel de seguridad 4.

5.3.2. Advertising

La selección del intervalo de *advertising* implica un compromiso entre la latencia de descubrimiento del dispositivo, asociada a la usabilidad, y el consumo energético. Inter-

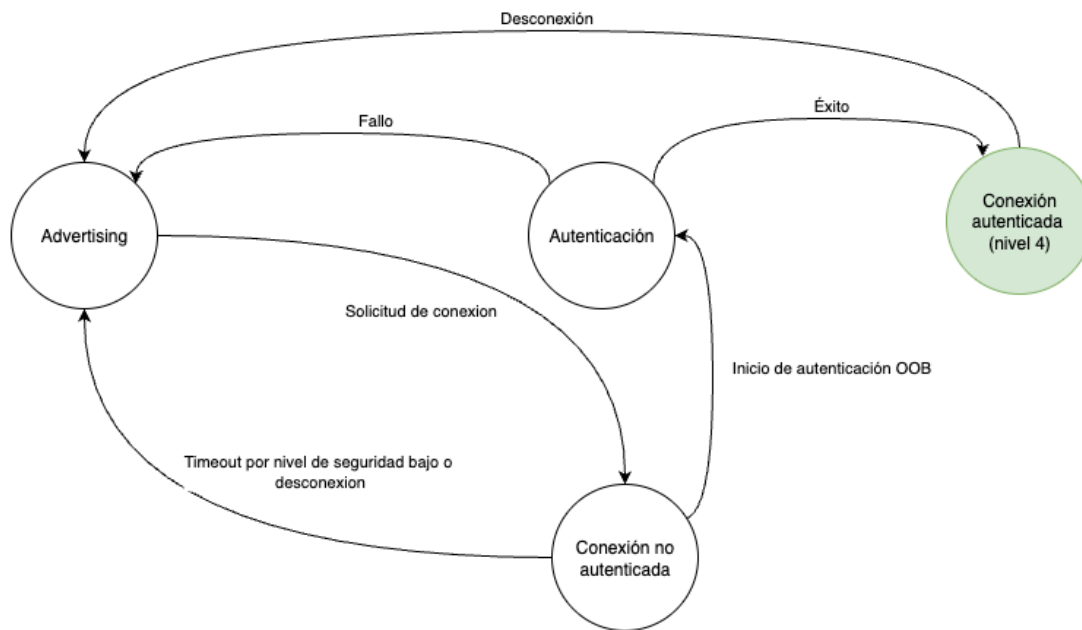


Figura 5.5: Diagrama de estados de la comunicación entre el dispositivo y el teléfono celular.

valos más cortos permiten una detección más rápida por parte del dispositivo central, a costa de un mayor consumo de energía.

Para determinar un valor adecuado, se utilizó la herramienta *Nordic Online Power Profiler for Bluetooth LE* [17], que permite estimar el consumo tanto en estado de *advertising* como durante la conexión, en función de distintos parámetros, tales como la tensión de alimentación, el intervalo de *advertising* y el tamaño del *payload*.

En este caso, el tamaño del *payload* está determinado por la cantidad de información a transmitir, la cual se busca minimizar en la medida de lo posible, sin quitar información de interés. Por otro lado, el intervalo de *advertising* puede ajustarse dentro de un rango que permita mantener una adecuada experiencia de usuario, sin comprometer significativamente el consumo energético.

Los datos configurados para su transmisión en los paquetes de *advertising* corresponden al nombre del dispositivo y a dos banderas: una que indica que el dispositivo es descubrible por cualquier otro dispositivo, y otra que especifica que no soporta Bluetooth clásico. A continuación, se detalla el cálculo del tamaño del *payload* asociado al nombre del dispositivo y a las *flags*:

- Flags:
 - Length: 1 byte
 - Type: 1 byte
 - Data: 1 byte

Total Flags = 3 bytes

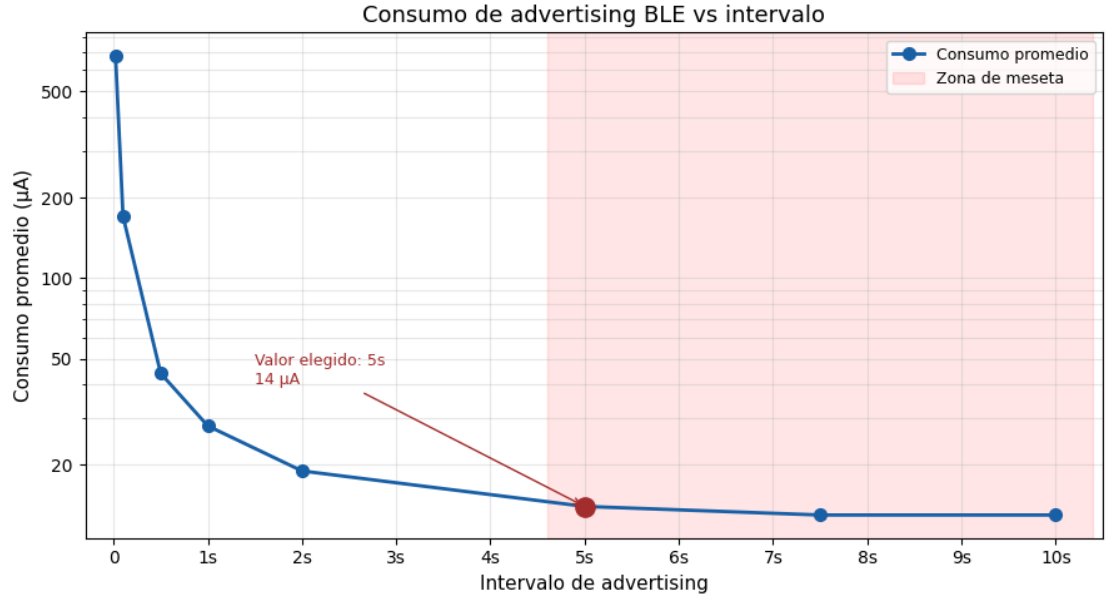


Figura 5.6: Consumo promedio del *advertising* BLE en función del intervalo de *advertising*. Datos extraídos de la herramienta Nordic Online Power Profiler for Bluetooth LE [17]

■ Nombre del dispositivo (NeuroSoC):

- Length: 1 byte
- Type: 1 byte
- Data: 8 bytes

Total Nombre = 10 bytes

Por lo tanto, el tamaño total de los datos de *advertising* es:

$$\text{TxPayload} = 3 + 10 = 13 \text{ bytes} \quad (5.1)$$

A partir del uso de la herramienta de estimación y de ensayos experimentales variando el intervalo de *advertising*, y considerando el *TxPayload* previamente calculado, se observó que el consumo promedio tiende a un valor cercano a $13 \mu A$ a medida que aumenta el intervalo de *advertising*, tal como se muestra en la Figura 5.6.

A partir de los resultados obtenidos mediante la herramienta, se realizaron ensayos experimentales variando el intervalo de *advertising*. Se observó que, para valores superiores a 5 segundos, el tiempo requerido para el establecimiento de la conexión aumentaba de manera significativa, afectando negativamente la experiencia de usuario.

En consecuencia, se seleccionó un intervalo de *advertising* de 5 segundos, logrando un compromiso adecuado entre consumo energético y tiempo de descubrimiento, sin degradar la usabilidad del sistema. Al utilizar este intervalo, se obtiene un consumo promedio de $14 \mu A$. En la Figura 5.7 se muestra el consumo de los distintos componentes del protocolo BLE durante un intervalo de *advertising* de 5 segundos.

El tipo de *advertising* elegido fue *ADV_IND*, es decir, conectable y escaneable.

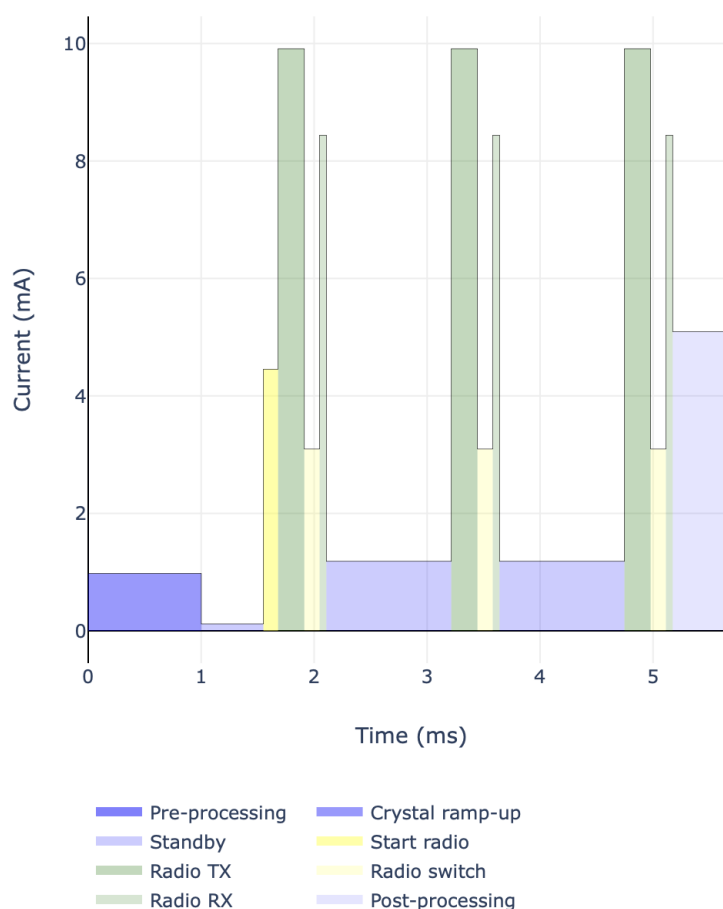


Figura 5.7: Consumo de los diferentes componentes de *advertising* durante un *advertising* interval. Imagen extraída de la herramienta *Nordic Online Power Profiler for Bluetooth LE* [17]

5.3.3. Capa GATT

La configuración de la capa GATT implementada incluye dos servicios: uno desarrollado específicamente para este proyecto y otro definido por la especificación de Bluetooth.

El servicio desarrollado corresponde al *NSS (Neurostimulation Service)*, el cual se encarga de la gestión de parámetros asociados a la terapia y del protocolo de comunicación utilizado para su configuración. El detalle de este servicio y de los protocolos asociados se presenta en la Sección 5.3.7.

Adicionalmente, se incorpora el servicio estándar *DIS (Device Information Service)* [18], provisto por la especificación BLE, que permite acceder a información general del dispositivo, como nombre, modelo y versión de firmware.

En el sistema desarrollado, el dispositivo implementado asume el rol de servidor GATT, almacenando los datos que pueden ser consultados y/o modificados por un cliente. En este caso, el cliente corresponde a un dispositivo móvil que utiliza la aplicación de Nordic *nRF Connect for Mobile*, la cual permite establecer la conexión y acceder a

los servicios expuestos por el dispositivo.

Para poder realizar operaciones de lectura o escritura sobre las características, es necesario que la conexión haya alcanzado el nivel de seguridad 4. De este modo, se evita que dispositivos no autenticados accedan a información sensible, considerando que las características pueden ser descubiertas inmediatamente después de establecer la conexión, incluso si esta aún no ha sido autenticada.

5.3.4. Conexión

Se realizó un análisis análogo al llevado a cabo para la configuración de los parámetros de *advertising*, utilizando la misma herramienta de estimación de consumo, pero considerando el estado de conexión. En este caso, se estimó el consumo para distintas configuraciones de *PHY* e intervalos de conexión, con el objetivo de minimizar el consumo energético sin afectar la experiencia de usuario, evitando la introducción de retardos o desconexiones no deseadas.

A diferencia del caso de *advertising*, en el estado de conexión los parámetros no son definidos exclusivamente por el dispositivo *peripheral*, sino que el dispositivo central es quien establece los valores finales. Si bien el *peripheral* puede solicitar actualizaciones de parámetros, estas no necesariamente son aceptadas, por lo que la configuración efectiva puede variar dependiendo del dispositivo central utilizado.

En el escenario de uso previsto, la cantidad de datos útiles intercambiados es reducida, predominando los paquetes de control propios del protocolo BLE que se transmiten en cada *connection interval*. Por esto se consideraron valores de *RX* y *TX payload* iguales a 0 bytes. En este contexto, los parámetros con mayor impacto sobre el consumo son la *PHY* utilizada y el intervalo de conexión.

Al analizar el consumo en función del intervalo de conexión, manteniendo la *PHY* fija en 1M, se observó que el consumo promedio decrece a un valor cercano a 13 μA para intervalos elevados. Este comportamiento se mantuvo al repetir el análisis utilizando la *PHY* de 2M, obteniéndose valores de consumo similares. Los resultados correspondientes se presentan en la Figura 5.8.

En base a estos resultados, se optó por utilizar la *PHY* de 2M, dado que presenta compatibilidad con la mayoría de dispositivos comerciales y permite un menor consumo en intervalos de conexión menores, los cuales posiblemente el dispositivo central prefiera.

Finalmente, se adoptó el menor intervalo de conexión que permite mantener un consumo promedio cercano al valor en régimen estacionario, sin degradar la usabilidad del sistema. En particular, se seleccionó un intervalo de conexión de 750 ms, para el cual se estima un consumo promedio de 18 μA , tal como se presenta en la Figura 5.9.

5.3.5. Configuración de los parámetros BLE

De acuerdo con los criterios de diseño presentados previamente, se seleccionaron los parámetros de *advertising* y conexión BLE que se resumen en la Tabla 5.1. Estos valores buscan satisfacer los requerimientos de consumo energético, tiempo de establecimiento de conexión y desempeño de la comunicación definidos para la aplicación.

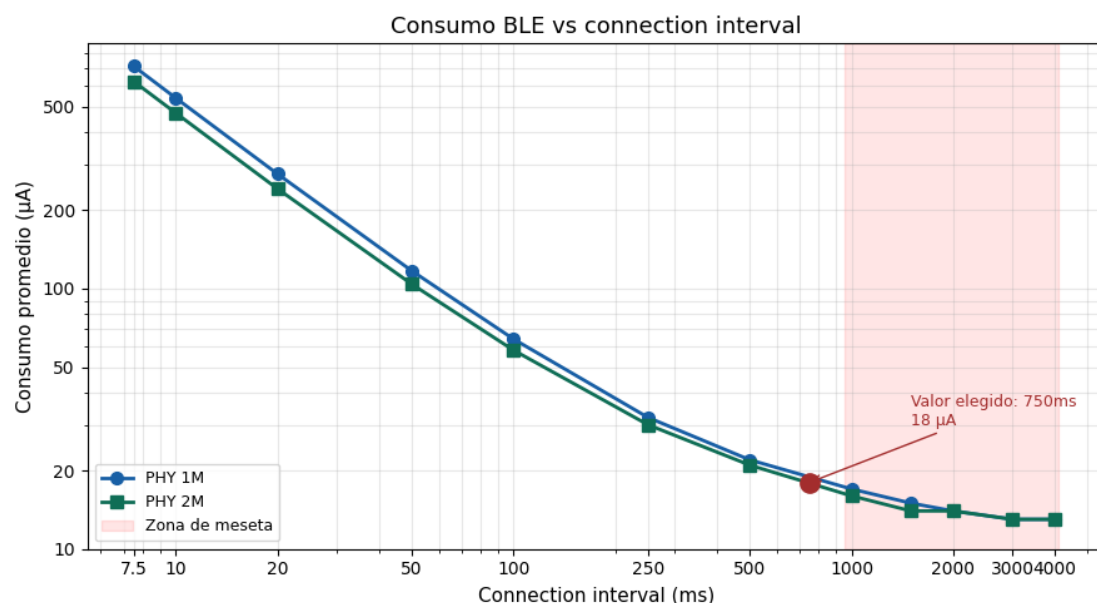


Figura 5.8: Consumo promedio de la conexión BLE en función de la *PHY* utilizada y del intervalo de conexión. Datos obtenidos a partir de la herramienta de [17].

Tabla 5.1: Parámetros de *advertising* y conexión BLE seleccionados.

Parámetro	Valor
Intervalo de advertising	5000 ms
Payload de advertising (TX)	13 bytes
Intervalo de conexión	750 ms
Payload de conexión (TX)	0 bytes
Payload de conexión (RX)	0 bytes
PHY	2M
Slave latency	0 (valor por defecto)
Potencia de transmisión	0 dBm

5.3.6. Emparejamiento y autenticación

Para la selección del método de emparejamiento, se consideraron las recomendaciones de seguridad de Bluetooth [19]. En primer lugar, se evaluó la necesidad de intercambiar claves que habiliten mecanismos de seguridad, como el cifrado y la privacidad. Se optó por garantizar el canal más seguro posible, requiriendo la distribución de claves que permitan aprovechar todas las funcionalidades de seguridad provistas por Bluetooth.

De igual forma, el sistema busca protegerse frente a interceptores pasivos que puedan intentar capturar dichas claves durante su distribución, evitando que terceros no autorizados logren autenticarse. Adicionalmente, con el objetivo de mitigar ataques comunes durante el proceso de emparejamiento, como el *Man-in-the-Middle (MITM)*, se requiere un mecanismo adicional de autenticación.

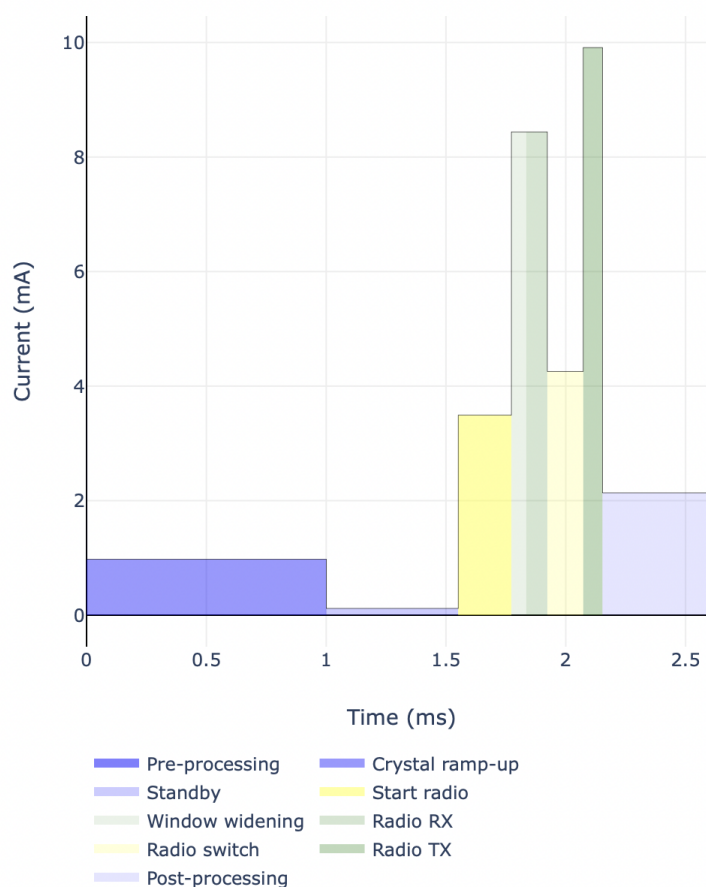


Figura 5.9: Consumo de los diferentes componentes de la conexión durante un intervalo de conexión. Imagen extraída de la herramienta *Nordic Online Power Profiler for Bluetooth LE* [17]

La elección del método de emparejamiento se realizó considerando las capacidades de entrada y salida de ambos dispositivos, como se mencionó en la Sección A.6. Dado que el dispositivo está diseñado para ser implantado, no dispone de interfaces de entrada ni salida. Por otro lado, el teléfono celular cuenta tanto con teclado como con pantalla. Sin embargo, los métodos *Passkey Entry* y *Numeric Comparison* requieren que ambos dispositivos posean capacidades de entrada y/o salida, condición que no se cumple en este caso. En consecuencia, los métodos de emparejamiento viables se reducen a *Just Works* y *Out of Band*.

En función de estos requerimientos y de las limitaciones de entrada/salida del dispositivo previamente mencionadas, se optó por utilizar el método *Out of Band (OOB)* como mecanismo de emparejamiento.

Luego, el proceso de emparejamiento puede dividirse en tres etapas. En la primera, que comienza con una solicitud de emparejamiento por parte del dispositivo iniciador (en este caso un teléfono celular), se define si se empleará *LE Legacy Pairing* o *LE Secure Connections (LESC)*. Para el dispositivo se optó por *LESC*, ya que ofrece un nivel de seguridad considerablemente superior, lo cual resulta prioritario en este proyecto. Como

desventaja, esta elección implica que dispositivos con versiones de Bluetooth anteriores a 4.2 no puedan conectarse, ya que no disponen de esta funcionalidad. No obstante, se decidió priorizar la seguridad del canal y la autenticación por sobre la compatibilidad con equipos más antiguos.

A continuación, se identifica la segunda etapa de emparejamiento. En el caso del proyecto esta etapa incluye la autenticación con el usuario, la generación de una clave para el cifrado del enlace y, posteriormente, el inicio de la encriptación del mismo. El paso de autenticación con el usuario se implementa para lograr la seguridad mencionada anteriormente.

La primera etapa del proceso de autenticación se puede observar en la Figura 5.10. Para utilizar el emparejamiento mediante OOB, y considerando que se emplea *LESC*, al menos uno de los dispositivos debe tener activada la OOB Flag. En este caso, el dispositivo cuenta con dicha *flag* habilitada.

Por otro lado, el teléfono celular no necesariamente inicia el emparejamiento utilizando OOB, por lo que, a efectos del diagrama, se asume que no genera su propia clave. Cabe destacar que este comportamiento no es configurable por el usuario.

Para el proceso de emparejamiento, el dispositivo requiere la generación de ciertas claves criptográficas. En primera instancia, y previo al inicio del proceso de emparejamiento, el neuroestimulador genera un número aleatorio *random*, denominado r_b , y establece el valor del número aleatorio del otro dispositivo (r_a) en cero. A continuación, calcula un valor de confirmación C_b mediante la función f_4 , utilizando como parámetros la clave pública del neuroestimulador (PK_b) y el número r_b . La función f_4 corresponde al mecanismo de generación de valores de confirmación, el cual emplea r_b como clave de 128 bits, junto con la concatenación de los demás parámetros, para generar el mensaje sobre el cual se realiza el cálculo. De esta manera, el valor C_b solo puede ser obtenido si se conoce el número aleatorio r_b . Cuando el teléfono celular accede a los datos OOB del dispositivo, obtiene su dirección B , el número r_b y la confirmación C_b , los cuales son transmitidos mediante NFC.

A continuación, se intercambian las claves públicas PK_a y PK_b . A partir de estas, junto con las respectivas claves privadas SK_a y SK_b , cada dispositivo calcula la clave *Diffie-Hellman* (*DHKey*). La *DHKey* es una clave simétrica, lo que implica que ambos dispositivos obtienen el mismo valor una vez finalizado el cálculo. Esta clave es utilizada en la segunda etapa del proceso de autenticación.

Para finalizar la primera etapa del proceso de autenticación, el teléfono celular calcula el valor C_b a partir de PK_b y r_b . Una vez obtenido, lo compara con el valor C_b recibido mediante NFC. En caso de que ambos valores difieran, el proceso de emparejamiento se cancela. En cambio, si coinciden, se procede al intercambio de los *nonces* N_a y N_b , generados por cada dispositivo.

La segunda etapa del proceso de autenticación consiste en la realización de verificaciones adicionales, con el objetivo de asegurar que todos los datos intercambiados hasta el momento sean correctos. En esta etapa se llevan a cabo cuatro pasos de verificación. En caso de que el segundo o el tercer paso (encargados de validar la consistencia de la información intercambiada) fallen, el proceso de emparejamiento se cancela. El flujo correspondiente a esta etapa se ilustra en la Figura 5.11.

El primer paso cada dispositivo calcula dos claves: la LTK y la clave MacKey, calcu-

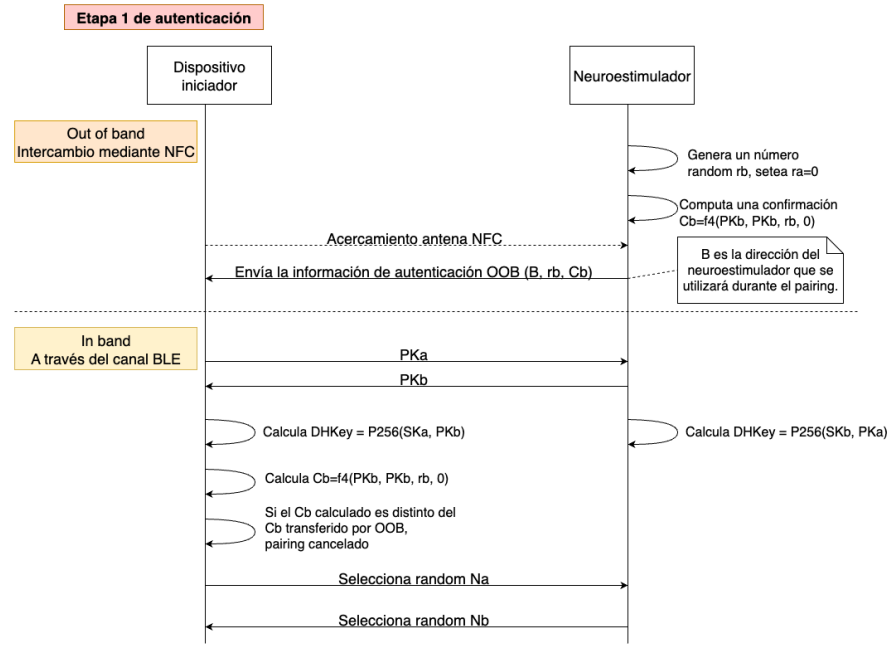


Figura 5.10: Etapa 1 del proceso de autenticación utilizando OOB.

ladas mediante la función $f5$:

$$MacKey || LTK = f5(DHKey, NaNb, BD_ADDR_C, BD_ADDR_P) \quad (5.2)$$

Los argumentos utilizados para el cálculo son los siguientes:

- El secreto compartido $DHKey$, cuyo cómputo involucra las claves privadas de cada dispositivo, las cuales nunca son compartidas.
- Los *nonces* N_a y N_b , intercambiados durante la primera etapa del proceso de autenticación.
- Las direcciones de ambos dispositivos.

A continuación, se lleva a cabo el segundo paso de la segunda etapa del proceso de autenticación, correspondiente al primer mecanismo de verificación. En este paso, el dispositivo iniciador calcula un valor de verificación, denominado E_a , mediante la función $f6$, la cual utiliza como clave la $MacKey$.

El valor E_a es enviado al otro dispositivo, que realiza el mismo cálculo de manera local y compara el resultado obtenido con el valor recibido. En caso de discrepancia entre ambos valores, el proceso de emparejamiento se cancela.

El tercer paso es análogo al segundo, pero en este caso se calcula el valor de verificación E_b . Si la verificación falla, el proceso de emparejamiento se cancela.

El último paso de la segunda etapa del proceso de autenticación consiste en la activación del cifrado de la comunicación entre ambos dispositivos. Para ello, se utiliza la clave de cifrado de largo plazo (LTK), previamente generada durante el primer paso de

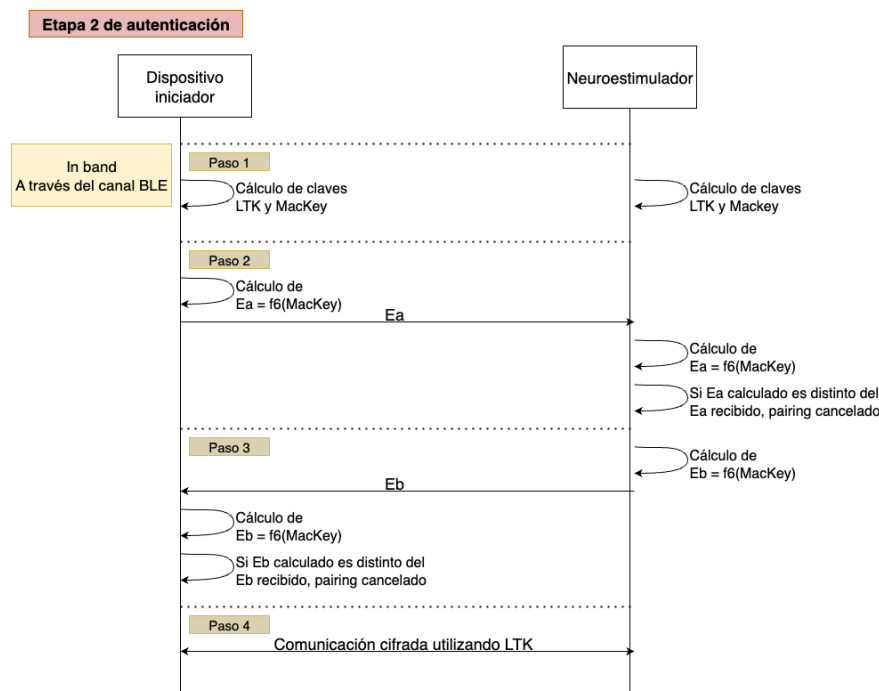


Figura 5.11: Etapa 2 del proceso de autenticación

esta etapa, a partir de la cual se deriva una clave de sesión. Esta última es la encargada de cifrar los datos intercambiados durante la conexión, garantizando la confidencialidad e integridad de la información transmitida.

Por último, se da una tercera y última etapa del proceso de emparejamiento, correspondiente a la distribución de claves. Durante esta etapa, se intercambian las claves *IRK* y *CSRK* (*Connection Signature Resolving Key*). Asimismo, la dirección Bluetooth del dispositivo se emplea como identificador único de la clave de largo plazo (*LTK*) dentro de la base de datos de seguridad.

Una vez completadas las tres etapas del emparejamiento, se considera que los dispositivos han sido emparejados. Dado que las claves *LTK*, *CSRK* e *IRK* se almacenan para su uso futuro, se establece una relación de *bonding*, lo que permite que, tras una desconexión, los dispositivos puedan reconectarse sin necesidad de repetir el proceso de emparejamiento.

En la implementación, la selección de LESC se realiza a nivel de configuración mediante el archivo `prj.conf`, donde se habilita su uso. Con esta configuración, Zephyr intentará utilizar LESC siempre que ambos dispositivos lo soporten; de lo contrario, permitirá el emparejamiento con Legacy Pairing. Para garantizar que las conexiones se establezcan únicamente con LESC, se agregó en el código una validación que rechaza cualquier intento de conexión que no utilice este método. Una vez realizada la conexión se inicializa un temporizador de 30 segundos. Si al expirar el temporizador no se alcanzó un nivel de seguridad 4, el dispositivo procede a desconectarse.

5.3.7. Servicio Neurostimulation Service (NSS)

Para la implementación del servicio mediante el cual se envían los parámetros programables utilizados en la estimulación, se evaluaron dos enfoques distintos. En primer lugar, se consideró diseñar un servicio con cinco características independientes, correspondientes a cada uno de los parámetros programables: amplitud, frecuencia, ancho de pulso, selección de electrodos y duración de la terapia.

Si bien este enfoque presenta la ventaja de brindar claridad al momento de enviar los valores de cada parámetro, introduce una limitación al momento de validar los parámetros como un conjunto. En efecto, al recibir los valores de manera independiente, es posible verificar que cada uno se encuentre dentro de su rango permitido y almacenarlo, pero se dificulta la validación de la coherencia entre ellos, en particular entre el ancho de pulso y la frecuencia.

Esto puede dar lugar a situaciones en las que un parámetro se configure correctamente de forma individual, pero al modificar otro parámetro posteriormente, la combinación de ambos resulte inválida. En dicho caso, sería necesario solicitar nuevamente la configuración de uno o más parámetros, lo que complica la lógica de validación y la interacción con el usuario.

Asimismo, ante la necesidad de incorporar nuevos parámetros o configuraciones, este esquema resultaría poco escalable, ya que implicaría agregar nuevas características por cada tipo de información a transmitir.

Es por esto que se consideró una segunda estrategia, la cual fue finalmente implementada. En este caso, se diseñó un servicio con una única característica, denominada *StimData Characteristic*, que permite enviar en un único mensaje la información correspondiente a múltiples parámetros.

El mensaje tiene una longitud de 64 bits y su interpretación depende del tipo de comando, definido en los 8 bits más significativos. Al recibir un nuevo valor en la característica *StimData*, se leen en primer lugar dichos bits y, en función del comando identificado, se interpreta el contenido del resto del buffer.

Se implementaron tres comandos distintos: el comando 0x01, destinado a la configuración de los parámetros programables de la terapia; el comando 0x02, utilizado para configurar el tipo de usuario conectado (*Patient* o *Programmer*) junto con su contraseña; y el comando 0x03, empleado para iniciar o finalizar la terapia.

Este protocolo, a diferencia del descrito previamente, presenta la desventaja de ser menos intuitivo. Sin embargo, permite validar de forma conjunta los parámetros en el momento de recibir un comando de configuración, simplificando la lógica de verificación.

Asimismo, mejora la portabilidad hacia otros tipos de interfaces de comunicación, dado que se basa en un único mensaje que concentra toda la información necesaria. Además, resulta más escalable, dado que permite incorporar nuevos tipos de comandos sin modificar la estructura general del protocolo.

Comando 0x01: Configuración de parámetros de estimulación

A cada parámetro se le asigna una posición específica dentro del buffer, así como una cantidad de bits acorde al tamaño de la información a representar. En la Figura 5.12 se detalla la organización del buffer y la asignación de bits correspondiente.

Capítulo 5. Diseño e implementación de Firmware

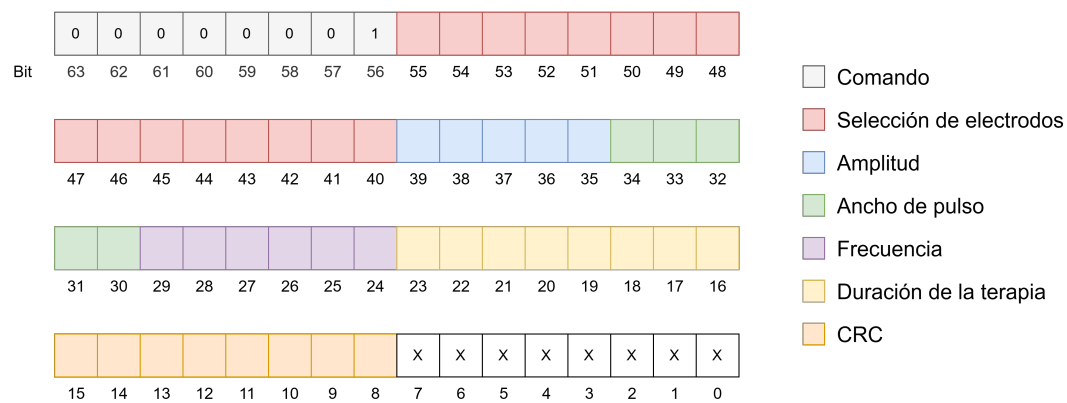


Figura 5.12: Formato del buffer de la característica *StimData* para el comando 0x01 de configuración de los parámetros programables de la terapia.

Tabla 5.2: Rango de valores posibles y pasos definidos para la amplitud, el ancho de pulso y la frecuencia de estimulación.

Parámetro	Paso	Valor mínimo	Valor máximo
Amplitud de corriente	0.5 mA	0.0 mA	15.0 mA
Ancho de pulso	30.5 μs	20.0 μs	996 μs
Frecuencia	1 Hz en [1, 10] Hz 2 Hz en (10, 20] Hz 5 Hz en (20, 50] Hz 10 Hz en (50, 100] Hz 20 Hz en (100, 200] Hz 50 Hz en (200, 500] Hz 100 Hz en (500, 1500] Hz	1 Hz	1500 Hz

Dado que cada parámetro posee un rango discreto de valores con incrementos fijos, fue posible determinar la cantidad de bits necesaria para su codificación. En la Tabla 5.2 se presentan los rangos y resoluciones definidos.

Por ejemplo, la amplitud de corriente del estímulo presenta un paso de 0.5 mA en el rango [0.0; 15.0] mA, lo que resulta en 30 valores posibles. En consecuencia, se optó por transmitir el índice del valor (número de paso), de modo que el sistema reconstruya la magnitud multiplicando dicho índice por el incremento correspondiente. De esta forma, se requieren únicamente 5 bits para representar todos los valores posibles. Este enfoque evita además la recepción de valores arbitrarios fuera del conjunto permitido.

La misma metodología se aplicó al ancho de pulso, con la diferencia de que en este caso existen 32 valores posibles y el incremento es de 30.5 μs .

En el caso de la frecuencia, se adoptó una estrategia diferente, ya que el incremento depende del rango de valores considerado. Dado que existen 47 valores posibles, estos fueron mapeados de forma ascendente en un arreglo. Así, la selección de la frecuencia se realiza mediante el envío de un índice entre 0 y 46, correspondiente a la posición en

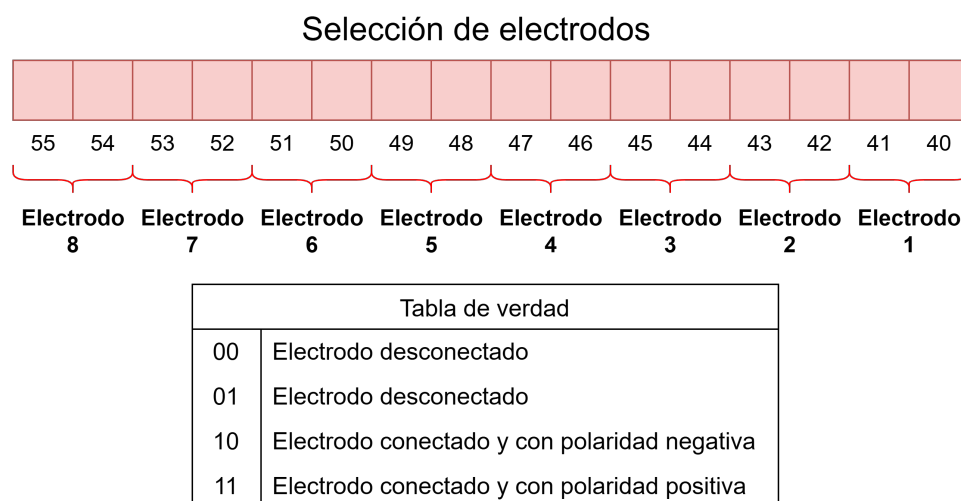


Figura 5.13: Asignación de bits para cada electrodo y tabla de verdad para el par de bits.

dicho arreglo. Por ejemplo, para configurar una frecuencia de 1500 Hz, se debe enviar el valor 46 en los 6 bits asignados a este parámetro.

Por otra parte, la selección de los electrodos y su polaridad se codifica utilizando 16 bits, asignando 2 bits a cada uno de los 8 electrodos. Para cada electrodo, el bit más significativo indica el estado de conexión (1 = conectado, 0 = desconectado), mientras que el bit menos significativo define la polaridad (1 = positivo, 0 = negativo). En la Figura 5.13 se muestra la tabla de verdad correspondiente y la asignación de bits. Cabe destacar que esta codificación se definió considerando que la plataforma de estimulación otorgada por *Integer Holdings Corporation* dispone de 8 electrodos.

En cuanto a la duración de la terapia, se asignaron 8 bits, lo que permite configurar sesiones de hasta 255 minutos. Finalmente, se destinaron otros 8 bits al CRC.

En caso de recibir este comando durante la ejecución de la máquina de estados de estimulación, los valores no son almacenados, dado que la modificación de los parámetros de estimulación se encuentra deshabilitada durante la terapia.

Comando 0x02: Configuración del tipo de usuario

El objetivo de este comando es configurar el tipo de usuario de un dispositivo una vez que ha sido emparejado, de modo de almacenar en memoria persistente la correspondencia entre la dirección BLE y el tipo de usuario, tal como se describe en la Sección 5.3.8.

Los tipos de usuario que pueden asignarse son: *Patient* y *Programmer*. Para su codificación se reserva 1 bit, donde 0 corresponde a *Patient* y 1 a *Programmer*. Adicionalmente, la configuración del usuario debe ir acompañada de una contraseña, para la cual se asignan 8 bits.

Una vez recibido este comando, se lee el bit asignado al tipo de usuario y, a continuación, se compara la contraseña correspondiente almacenada en el sistema con la recibida en el buffer. Para la prueba de concepto desarrollada en este proyecto, dicha contraseña

Capítulo 5. Diseño e implementación de Firmware

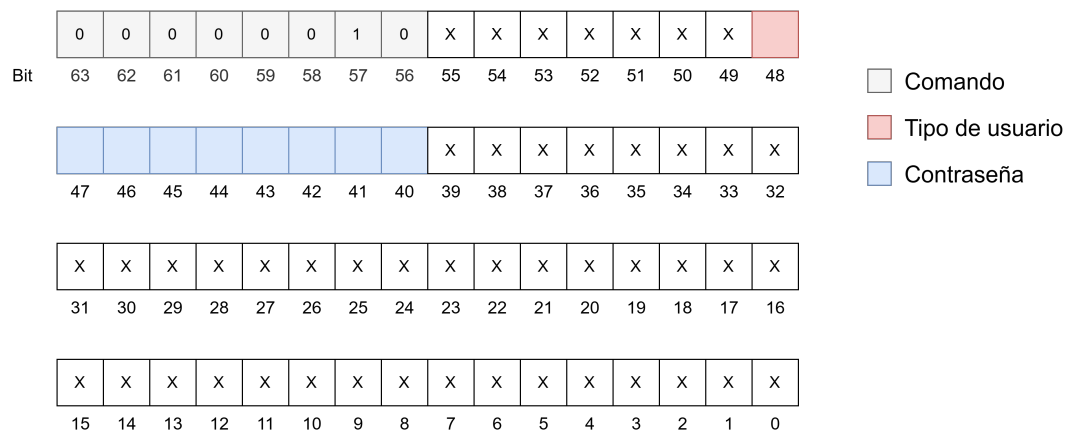


Figura 5.14: Formato del buffer de la característica *StimData* para el comando 0x02 de configuración del tipo de usuario.

se implementó como un valor fijo definido en el *firmware*, por lo que la validación consiste únicamente en verificar que la contraseña recibida coincida con dicho valor. En la Figura 5.14 se muestra la ubicación de estos campos dentro del mensaje.

En el caso de un dispositivo recientemente emparejado al cual no se le haya asignado un tipo de usuario, este no podrá ejecutar ningún comando hasta que dicha configuración sea realizada.

Comando 0x03: Inicio/detención de la terapia

Este comando se utiliza tanto para iniciar como para detener la terapia. Cada vez que el sistema recibe un comando de este tipo, se genera un evento cuya acción depende del estado actual del sistema.

Si el comando es recibido durante la ejecución de la terapia, se interpreta como una solicitud de detención, provocando su finalización. En cambio, si la terapia no se encuentra en ejecución, el comando se interpreta como una solicitud de inicio, siempre que los parámetros de estimulación previamente configurados sean válidos. En caso contrario, el comando es descartado.

Dado que este comando no requiere el envío de información adicional, los únicos bits relevantes dentro del buffer son aquellos correspondientes al identificador de comando. El formato resultante se muestra en la Figura 5.15.

Módulo nss.c

Para la implementación del servicio y su correspondiente característica fue necesario definir dos UUIDs. En este proyecto, los mismos fueron generados utilizando la herramienta web *UUID Generator* [20], la cual permite obtener identificadores a partir de valores aleatorios. Si bien esta herramienta no garantiza la unicidad absoluta de los identificadores generados, para el alcance de este proyecto dicha condición no resulta crítica, dado que se trata de una prueba de concepto y no de un producto destinado a su comercialización.

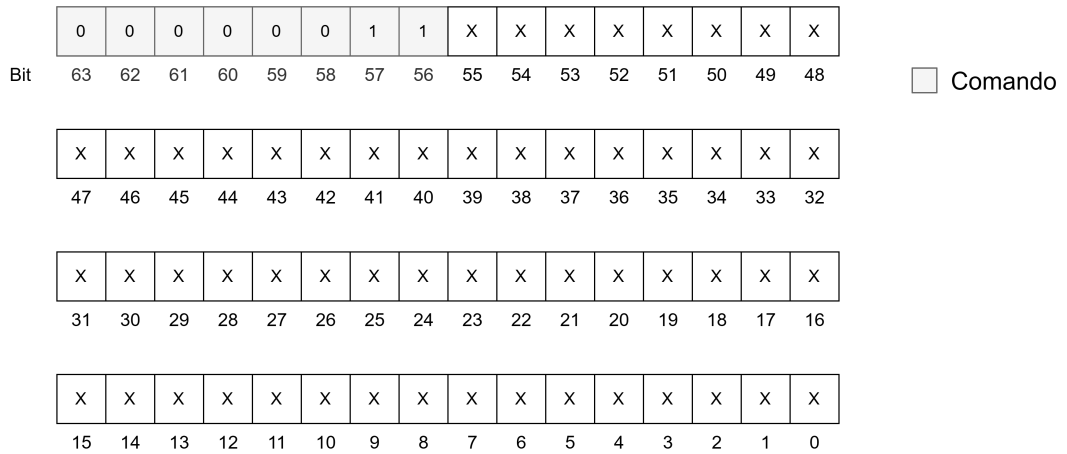


Figura 5.15: Formato del buffer de la característica *StimData* para el comando 0x03 de inicio/detención de la terapia.

Los UUID definidos son los siguientes:

- *Neurostimulation Service UUID:*
0x438c7dee-77ec-4a5b-8741-24c721eeb043
- *StimData Characteristic UUID:*
0x438c7def-77ec-4a5b-8741-24c721eeb043

Adicionalmente, se incorporó un descriptor de característica (véase Sección A.5.4) con el objetivo de permitir la consulta del nombre de la misma. Este descriptor utiliza un UUID predefinido por la especificación BLE, que lo identifica de manera unívoca como un descriptor de tipo *Characteristic User Description* (0x2901).

Para la definición de la característica *StimData* se establecieron tanto sus propiedades como sus permisos de acceso. Para ello, se utilizaron las siguientes banderas:

- BT_GATT_CHRC_WRITE: Permite que el cliente BLE escriba la característica.
- BT_GATT_CHRC_READ: Permite que el cliente BLE lea el valor de la característica.
- BT_GATT_PERM_WRITE_LESC: Requiere el uso de LE Secure Connections (LESC) para la escritura.
- BT_GATT_PERM_READ_LESC: Requiere el uso de LE Secure Connections (LESC) para la lectura.
- BT_GATT_PERM_WRITE_ENCRYPT: Permite la escritura únicamente si la conexión se encuentra encriptada.
- BT_GATT_PERM_READ_ENCRYPT: Permite la lectura únicamente si la conexión se encuentra encriptada.

Capítulo 5. Diseño e implementación de Firmware

- `BT_GATT_PERM_WRITE_AUTHEN`: Requiere autenticación (protección contra ataques *Man-In-The-Middle*) para la escritura.
- `BT_GATT_PERM_READ_AUTHEN`: Requiere autenticación (protección contra ataques *Man-In-The-Middle*) para la lectura.

Estas configuraciones permiten restringir el acceso a la característica, evitando que dispositivos no emparejados puedan leer o modificar los valores. En particular, aunque un dispositivo logre establecer una conexión sin completar el proceso de emparejamiento, no podrá ejecutar comandos ni acceder a la característica debido a las restricciones de seguridad impuestas.

Cabe destacar que, en caso de que un dispositivo se conecte sin emparejarse, la conexión se finaliza automáticamente luego de 30 segundos, tal como se describe en la Sección 5.3.6. Durante este intervalo, el dispositivo no dispone de los permisos necesarios para enviar comandos válidos, incluso en el caso de conocer credenciales de mayor privilegio (contraseña de *Programmer*).

En este módulo, además de la definición del servicio y su correspondiente característica, se implementaron las *callbacks* de lectura y escritura. En particular, en la *callback* de escritura se realiza la identificación de los comandos mencionados previamente, así como la verificación de los privilegios asociados al tipo de usuario.

Adicionalmente, se incorporó una verificación de integridad de los datos recibidos mediante CRC. Para ello, el mensaje enviado por BLE incluye un campo de 8 bits que contiene el CRC calculado sobre los campos de datos del comando. Al recibir el mensaje, el *firmware* extrae dicho valor y calcula nuevamente el CRC sobre el contenido recibido, excluyendo el propio campo de CRC. Si ambos valores no coinciden, el mensaje se considera corrupto y se descarta sin modificar los parámetros internos del neuroestimulador. Esta verificación agrega una capa de robustez a nivel de aplicación, ya que permite comprobar la integridad de los datos desde el momento en que el programador carga los parámetros, hasta el instante en que el *firmware* los recibe, recalcula el CRC y verifica su coincidencia antes de cargarlos internamente para configurar la terapia.

5.3.8. Módulo `user_type`

Se implementó un módulo con el objetivo de incorporar un nivel adicional de autenticación. En este esquema, una vez que un dispositivo es emparejado, se le debe asignar un tipo de usuario, el cual puede ser *Patient* o *Programmer*. En función de esta clasificación, se determinan los privilegios y permisos asociados.

Para configurar el tipo de usuario, se debe enviar un comando BLE, tal como se describe en la Sección 5.3.7, acompañado de una contraseña que es validada por el sistema. Esta información se almacena en memoria persistente, de modo que, cada vez que un dispositivo previamente emparejado se conecta, el sistema recupera automáticamente su tipo de usuario y los privilegios correspondientes.

Se estableció un límite en la cantidad de usuarios de cada tipo, permitiéndose hasta tres usuarios *Patient* y dos *Programmer*. Un usuario de tipo *Patient* únicamente puede iniciar o detener la terapia, sin posibilidad de modificar los parámetros de estimulación, tales como la amplitud, el ancho de pulso, la frecuencia, la selección de electrodos y

la duración de la terapia. Por el contrario, un usuario de tipo *Programmer* dispone de permisos tanto para configurar los parámetros como para iniciar o finalizar la terapia.

De esta manera, cada dispositivo emparejado debe ser asociado a un tipo de usuario, quedando esta información almacenada para futuras conexiones. En cada encendido del sistema, durante la etapa de inicialización, se carga la correspondencia entre la dirección pública BLE y el tipo de usuario.

Finalmente, para permitir la eliminación de esta información, se implementó un botón que, al ser presionado, borra los datos de los dispositivos emparejados junto con sus tipos de usuario almacenados en memoria persistente. Esta solución fue adoptada para la prueba de concepto desarrollada en el proyecto, ya que simplifica las tareas de validación y ensayo del sistema. En una implementación final, esta funcionalidad podría reemplazarse por un comando específico accesible únicamente para usuarios con permisos de tipo *Programmer*, evitando la necesidad de interacción física con el dispositivo.

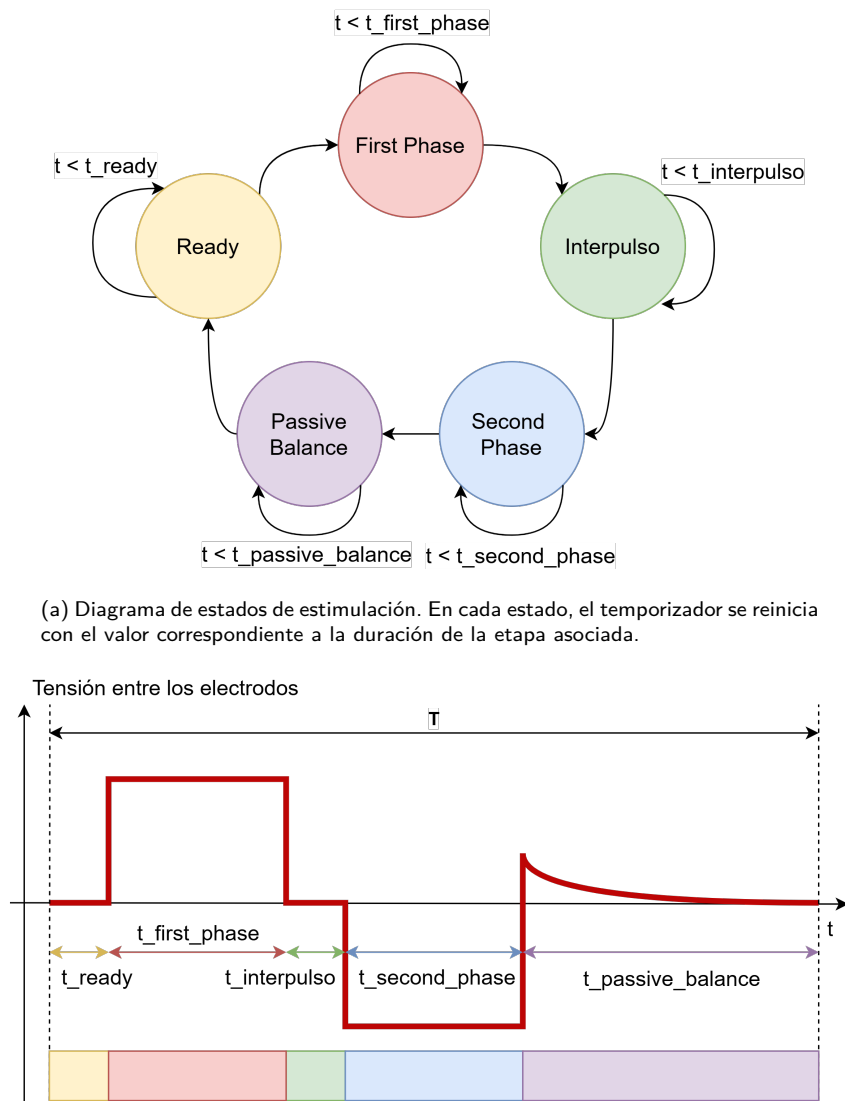
5.4. Subsistema de Estimulación

5.4.1. Funcionamiento de la máquina de estados de estimulación

La máquina de estados está compuesta por cinco estados, cada uno de los cuales representa una fase del pulso de estimulación. Las transiciones entre estados están controladas por eventos generados por un temporizador, cuya duración se ajusta en función del estado activo. En la Figura 5.16a se ilustra el funcionamiento de la máquina de estados, mientras que la Figura 5.16b presenta la evolución temporal de la tensión entre los electrodos durante un ciclo completo, indicando la duración de cada estado.

A continuación, se describe el objetivo de cada uno de los estados de la máquina:

- **Ready:** Este es el estado inicial, cuyo objetivo es verificar si existe alguna condición que requiera finalizar la terapia, dado que el pulso de estimulación no puede interrumpirse a mitad de su ejecución. Asimismo, en este estado se configura la tensión adecuada del *boost converter* en función de las mediciones de voltaje de la fuente de corriente, tal como se describe en la Sección 5.4.4.
- **First Phase:** En este estado se aplica el pulso de estimulación de acuerdo con los parámetros previamente configurados mediante BLE antes del inicio de la terapia, tales como la amplitud, el ancho de pulso, la frecuencia, los electrodos seleccionados junto con su polaridad y la duración de la terapia. Adicionalmente, en este estado se realiza la medición de la tensión de la fuente de corriente para su posterior procesamiento.
- **Interpulse:** Este es un estado de espera entre la primera y segunda fase del pulso de estimulación, durante el cual no circula corriente. Su duración no es configurable y se encuentra fijada en $20\ \mu\text{s}$.
- **Second Phase:** El objetivo de este estado es realizar un balance activo de carga, con el fin de lograr una corriente continua neta nula. Para ello, se aplica un pulso de estimulación de igual magnitud en sentido opuesto, es decir, con la polaridad de los electrodos invertida.



(a) Diagrama de estados de estimulación. En cada estado, el temporizador se reinicia con el valor correspondiente a la duración de la etapa asociada.

(b) Diagrama de la tensión entre los electrodos en función del tiempo. Se detalla la duración de cada estado.

Figura 5.16

- ***Passive Balance:*** El balance pasivo de carga se encarga de disipar la carga residual almacenada en los condensadores que no haya sido compensada por el balance activo. Durante este estado, además, se realiza el procesamiento de las mediciones correspondientes a la tensión de la fuente de corriente y de la batería. Su duración no es configurable y se encuentra fijada en 165 μ s.

Los motivos por los cuales la terapia puede finalizar son: la recepción de un evento de detención de la terapia vía BLE, la detección de un imán, la ocurrencia de un evento de batería baja y la expiración del temporizador de duración de la terapia. Como se mencionó previamente, la ejecución de un ciclo de la máquina de estados no puede interrumpirse, a fin de evitar la inyección de corriente continua al tejido. En consecuencia, las condiciones de finalización se evalúan únicamente al comienzo de cada ciclo de estimulación.

Previo al inicio de la terapia, se verifica la disponibilidad de parámetros válidos de configuración; en caso de no existir, la terapia no se inicia. Asimismo, los parámetros no pueden modificarse durante la estimulación: cualquier intento de reconfiguración durante esta etapa es ignorado y no se almacena. Durante la terapia, el único comando recibido por BLE que es atendido es el de detención.

Dado que se requiere alta precisión en el ancho de pulso durante la primera y la segunda fase de estimulación, el sistema se diseñó de manera tal que el procesamiento de las mediciones se realice durante el estado de balance pasivo, el cual presenta una mayor duración. De este modo, durante la primera fase únicamente se adquiere la medición de la tensión de la fuente de corriente.

5.4.2. Implementación de la máquina de estados: módulo `stim_state_machine`

Durante el desarrollo del proyecto se evaluó la utilización del *State Machine Framework* de Zephyr. No obstante, su implementación presentó diversas limitaciones en el contexto de la aplicación.

En primer lugar, se observó que la precisión temporal en la ejecución de los estados no resultaba adecuada para los requerimientos del sistema, ya que las transiciones no ocurrían exactamente en los instantes esperados, presentando desvíos temporales del orden de algunos milisegundos. Estos desvíos se deben a que la ejecución del *framework* está basada en el procesamiento de eventos y no en interrupciones temporizadas que desencadenen inmediatamente los cambios de estado. Como alternativa, se exploró la posibilidad de forzar los cambios de estado desde el *handler* de un temporizador. Sin embargo, este enfoque no se ajusta al modelo de uso previsto para el *framework*, además de introducir restricciones adicionales.

En particular, al realizar las transiciones de estado dentro de rutinas de interrupción, se imposibilita el uso de mecanismos de sincronización como semáforos, debido a que estos no pueden ser utilizados en dicho contexto. Este aspecto resulta crítico, ya que el sistema requiere la utilización de semáforos para la coordinación entre hilos y la ejecución de tareas de procesamiento durante la estimulación.

Adicionalmente, se determinó que el uso del *State Machine Framework* introduce una complejidad innecesaria para la lógica de control requerida. La secuencia de operación

del sistema es estrictamente determinística, con una progresión lineal entre estados y sin bifurcaciones ni transiciones condicionales. Asimismo, no se contempla la interrupción del ciclo de estimulación una vez iniciado.

Finalmente, la máquina de estados fue implementada mediante una estructura de control del tipo **switch-case**, como se muestra en el pseudocódigo 1. En esta, la transición entre estados se realiza de forma cíclica a partir de la expiración del temporizador asociado a cada estado. Al ingresar a cada estado, se inicializa el temporizador correspondiente y, a continuación, se realiza un *polling* de su estado. De este modo, una vez que el temporizador expira, se produce la transición inmediata al siguiente estado, garantizando la precisión temporal requerida.

Algorithm 1: Pseudocódigo de la máquina de estados de estimulación

```

current_state ← READY;
while flg_stimulating do
    switch current_state do
        case READY do
            stop ← verificar fin de estimulación;
            if no stop then
                ejecutar estado Ready;
                current_state ← FIRST_PHASE;
        case FIRST_PHASE do
            ejecutar primera fase;
            current_state ← INTERPULSE;
        case INTERPULSE do
            ejecutar interpulso;
            current_state ← SECOND_PHASE;
        case SECOND_PHASE do
            ejecutar segunda fase;
            current_state ← PASSIVE_BALANCE;
        case PASSIVE_BALANCE do
            ejecutar balance pasivo;
            current_state ← READY;

```

Para la implementación de los diferentes estados, este módulo interactúa con los siguientes:

- **electrodes.c**: módulo encargado del control de las llaves de los electrodos. (Véase Sección 5.4.3.)
- **sink.c**: módulo necesario para configurar la tensión adecuada del *boost converter* y medir la tensión de la fuente de corriente durante la primera fase. (Véase Sección 5.4.4.)

```
typedef struct {
    uint8_t number;
    uint8_t polarity;
    uint8_t activity;
    const struct gpio_dt_spec *gpio_pos;
    const struct gpio_dt_spec *gpio_neg;
} electrode;
```

Figura 5.17: Estructura de datos utilizada para representar un electrodo

- `timer.c`: módulo encargado de configurar y habilitar el temporizador en cada estado. (Véase Sección 5.4.5.)
- `dac.c`: módulo responsable de establecer en la salida del DAC la tensión equivalente a la amplitud de corriente programada mediante BLE, durante la primera y segunda fase de estimulación. (Véase Sección 5.4.6.)

5.4.3. Módulo electrodes

El módulo `electrodes.c` incluye las funciones encargadas del manejo de las llaves que conectan los electrodos a los circuitos de estimulación. Dependiendo del estado de las señales *ei_pos* y *ei_neg*, cada electrodo puede conectarse a *V_Anode*, a *V_Cathode* o quedar desconectado. El índice *i* indica el número de electrodo, que puede tomar valores de 1 a 4.

En primera instancia, se define una constante que indica la cantidad de electrodos presentes en el neuroestimulador. A partir de esta constante, se define un arreglo de tipo `gpio_dt_spec` compuesto por las estructuras que contienen la información de cada electrodo proveniente del *devicetree*. Posteriormente, utilizando este arreglo, se define una función encargada de la inicialización de los pines.

Para almacenar toda la información relativa a un electrodo, se definió una estructura llamada `electrode` como se detalla en la Figura 5.17.

El campo `number` indica el identificador del electrodo, mientras que `polarity` define su polaridad. El campo `activity` señala si el electrodo se encuentra activo. Por último, `gpio_pos` y `gpio_neg` almacenan referencias a los pines físicos que permiten conectar el electrodo a *V_{anode}* o *V_{cathode}* respectivamente.

El módulo contiene una función llamada `set_electrodes_by_state`, la cual toma como parámetros el estado actual de la máquina de estados de estimulación, la lista de electrodos activos de tipo `electrode` y el número de electrodos activos. Esta función se encarga de conectar los electrodos activos de forma adecuada según el estado de la terapia en el que se encuentre el sistema. Se detalla a continuación la conexión que se realiza de los electrodos en función del estado de la terapia:

- **Ready:** En caso de que el estado sea *Ready*, se desconectan todos los electrodos, ya que en este estado no se espera que circule corriente entre ellos.

- **First Phase:** Si la terapia se encuentra en la primera fase, los electrodos deben conectarse con la polaridad programada, de forma tal que la corriente circule desde el electrodo programado como positivo hacia el programado como negativo. Para lograr esto, se activa el pin *ei_Pos* en los electrodos positivos y el pin *ei_Neg* en los electrodos negativos, desactivando las señales complementarias. De esta forma, el electrodo positivo se conecta a *V_Anode* y el electrodo negativo a *V_Cathode*.
- **Interpulse:** Durante el interpulso, no debe circular corriente entre los electrodos, por lo tanto estos se desconectan.
- **Second Phase:** En la segunda fase, para el balance activo la corriente debe circular en el sentido contrario al de la primera fase. Para ello, se conectan los electrodos programados como positivos a *V_Cathode* y los programados como negativos a *V_Anode*, de forma tal que la corriente circule en el sentido opuesto al de la fase anterior.
- **Passive Balance:** Para realizar el balance pasivo, todas las señales *ei_Pos* de los electrodos activos se desactivan, mientras que todas las señales *ei_Neg* de los electrodos activos se activan. De esta forma, se descargan los condensadores presentes a la entrada de los electrodos, permitiendo que la corriente circule a través de la resistencia del cuerpo, sin involucrar a la fuente de corriente.

5.4.4. Módulo sink

El módulo `sink.c` consiste en la implementación de las funciones relacionadas al voltaje de la fuente de corriente. La fuente de corriente debe tener una tensión mínima en el colector para poder funcionar correctamente. Este valor está determinado por la ecuación

$$V_{sink} = V_{anode} - I * R_{tejido} \quad (5.3)$$

donde V_{sink} es la tensión en el colector, V_{anode} es la tensión a la salida de la llave del *boost* o de la batería, I es la corriente de estimulación y R_{tejido} es la resistencia entre los electrodos positivos y negativos. En la Figura 4.2 de la Sección 4.1.1 se identifican estos componentes y las señales involucradas.

Como se mencionó anteriormente, la tensión de alimentación de la fuente de corriente viene dada por el valor de V_{anode} , la amplitud de la corriente y la resistencia del tejido. La resistencia del tejido tiene un valor variable a lo largo del tiempo, y la amplitud de la corriente es programable, por lo tanto la tensión en la fuente se ve afectada por las variaciones de estos dos parámetros. Para lograr una fuente de corriente eficiente, con la mínima tensión necesaria para la correcta operación, se implementó un algoritmo para ajusta el valor de V_{anode} de forma tal que se garantice un correcto funcionamiento de la fuente.

El algoritmo consiste en medir la tensión de salida de la fuente de corriente mediante un divisor resistivo durante la primera fase del pulso de estimulación. En caso de que la tensión tenga margen para ser disminuida, se decrementa el valor de V_{anode} . En caso de que la tensión se encuentre en un valor bajo o muy próximo al límite, se aumenta la tensión en V_{anode} .

```
typedef struct {

    uint8_t VoltageStep;

    bool ShouldTakeMeasure;

    uint8_t IncreaseCounter;
    uint8_t DecreaseCounter;

} v_sink_state;
```

Figura 5.18: Estructura para controlar la tensión de alimentación de la fuente de corriente

Se implementó una estructura para almacenar las variables relativas al control de la tensión de alimentación de la fuente de corriente. La implementación se puede observar en la Figura 5.18.

El campo **VoltageStep** representa el valor actual de V_{anode} mediante un índice discreto comprendido entre 0 y 5. El valor 0 indica la conexión directa a la batería, activando el GPIO **stim_from_bat**, mientras que los valores entre 1 y 5 corresponden a los distintos niveles posibles de V_{Boost} , dentro del rango de 5 a 15 V, según se detalla en la Sección 4.1.

El campo **ShouldTakeMeasure** indica si en el siguiente pulso debe realizarse la medición de V_{sink} . Dado que el *boost converter* utilizado presenta un tiempo de respuesta del orden de $500\ \mu s$ para ajustar su tensión de salida, y considerando que los anchos de pulso programados son de duración comparable o incluso menor, no resulta factible adaptar el valor de en cada pulso. Por otro lado, las variaciones en la impedancia del tejido ocurren a frecuencias significativamente menores que la frecuencia de estimulación, por lo que realizar mediciones en cada pulso no aporta beneficios adicionales, así como tampoco implica una degradación del desempeño espaciar dichas mediciones.

En consecuencia, se optó por realizar la medición a intervalos periódicos, en este caso cada dos segundos. De esta manera, es posible ajustar la tensión de la fuente en función de los cambios de impedancia, respetando las limitaciones dinámicas del *boost converter*. La decisión de realizar la medición cada dos segundos se basó en la implementación de Integer.

Por último, para introducir histéresis en el ajuste de la tensión, se utilizan los campos **IncreaseCounter** y **DecreaseCounter**. Estos se emplean para contabilizar la cantidad de mediciones de tensión por encima o por debajo de los umbrales definidos, permitiendo determinar si corresponde o no realizar un ajuste en la tensión.

Para inicializar una variable del tipo **v_sink_state**, se define la función **InitializeVSink**. Esta función se encarga de establecer **VoltageStep** en su valor máximo, reiniciar los contadores y asignar el valor verdadero a la variable booleana **ShouldTakeMeasure**, de modo que se realice una medición de V_{sink} durante el siguiente pulso. De esta forma, se garantiza que el primer pulso disponga de una tensión de alimentación lo suficientemente elevada como para permitir a la fuente de corriente entregar

cualquier corriente programada.

Se implementa la función `SetVAnode`, la cual recibe como parámetro un puntero a una estructura del tipo `v_sink_state`. En función del valor del campo `VoltageStep`, esta función controla las señales `stim_from_bat` y `boost_ctrl`, activándolas o desactivándolas según corresponda. Adicionalmente, activa la señal `boost_en` con el objetivo de mantener la tensión en V_{boost} .

A continuación, se implementa la función `v_sink_measure`, encargada de disparar una conversión del SAADC y de generar un evento del tipo `EVENT_SINK_MEASUREMENT`, definido en el mismo módulo, que indica que se dispone de una medición de V_{sink} lista para ser procesada.

Por último, se implementa la función `ProcessVSinkMeasurement`. Esta función recibe como parámetros un puntero a una instancia de tipo `v_sink_state` y el resultado de una conversión del SAADC, con el objetivo de determinar si corresponde ajustar el valor de V_{anode} . En primer lugar, evalúa si la medición se encuentra por encima o por debajo de los umbrales definidos, y verifica si existe margen para incrementar o disminuir la tensión.

Si la medición se encuentra por debajo del umbral y es posible aumentar la tensión, se reinicia el contador `DecreaseCounter` y se verifica si se han acumulado tres mediciones consecutivas en dicha condición. En caso contrario, se incrementa el contador `IncreaseCounter`, encargado de llevar esta cuenta. Si se alcanza la cantidad de mediciones requerida para efectuar un cambio, se actualiza el valor de `VoltageStep` en consecuencia.

De forma análoga, si la medición se encuentra por encima del umbral y es posible disminuir la tensión, se aplica el mismo criterio utilizando el contador `DecreaseCounter`. En caso de que no se cumpla ninguna de las condiciones para modificar el valor de V_{anode} , ambos contadores se reinician. Finalmente, se actualiza el campo `ShouldTakeMeasure` a falso.

5.4.5. Módulo timer

En este módulo se implementan las funciones de inicialización y configuración del temporizador. Para ello, se utilizó la API `nrfx_timer`. Es necesario habilitar el uso del driver correspondiente mediante la opción `CONFIG_NRFX_TIMERn=y` (siendo n la instancia del periférico *timer* a utilizar) en el archivo de configuración del proyecto (`prj.conf`), como se menciona en la Sección 3.3.1. De este modo, se incluye la compilación de la librería `nrfx_timer`.

Conforme a los requerimientos de tiempo de los anchos de pulso (del orden de microsegundos) y con el objetivo de garantizar precisión, se utilizó la señal `PCLK16M` derivada del `HFCLK`. Esta señal, con una frecuencia de 16 MHz, proporciona una resolución temporal de 62.5 ns.

5.4.6. Módulo dac

En este módulo se implementan las funciones de control del DAC. Para ello, se utiliza una instancia del periférico SPI del SoC, mediante la cual se envía, a través de la línea

5.4. Subsistema de Estimulación

Upper Half:							
W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x
$\overline{A/B}$	—	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8
bit 15				bit 8			

Lower Half:							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
D7	D6	D5	D4	D3	D2	D1	D0
bit 7				bit 0			

Figura 5.19: Registro para escribir un comando en el DAC MCP4821. *Imagen extraída de la hoja de datos del DAC [15].*

MOSI, la palabra correspondiente a la amplitud de corriente deseada hacia la entrada SDI del DAC. En la Figura 5.19 se muestra la estructura del registro necesario para escribir una palabra en el DAC MCP4821. Se tiene que:

- **Bit 15:** Selección del DAC, dado que el integrado cuenta con dos DACs independientes.
 - 1 = Escritura en el DAC B.
 - 0 = Escritura en el DAC A.
- **Bit 13:** Ganancia de salida del DAC.
 - 1 = 1x.
 - 0 = 2x.
- **Bit 12:** Bit de control de apagado de la salida.
 - 1 = Salida habilitada.
 - 0 = Buffer de salida deshabilitado; la salida queda en alta impedancia.
- **Bits 11–0:** Bits de datos del DAC. Corresponden a un valor de 12 bits “D” que determina el nivel de salida, comprendido entre 0 y 4095.

Por lo tanto, se implementó una función que construye la palabra de configuración, seleccionando el DAC A, ganancia 1x y con la salida habilitada. Los bits de datos se calculan de acuerdo con la ecuación 5.4, donde V_{DAC} es la tensión deseada a la salida del DAC y $V_{REF} = 2.048$ V es la tensión de referencia interna del dispositivo.

$$data_bytes = \frac{V_{DAC}}{V_{REF}} \cdot 4096 \quad (5.4)$$

La tensión deseada a la salida del DAC se calcula según la ecuación 5.5, donde $RES_CURRENT_SOURCE = 20 \Omega$ corresponde a la resistencia de la fuente de corriente y $CTE_DIVISOR_TENSION = 5$ es la constante asociada al divisor de tensión que se encuentra a la entrada de la fuente, como se explica en la Sección 4.1.1.

$$V_{DAC} = I \cdot RES_CURRENT_SOURCE \cdot CTE_DIVISOR_TENSION \quad (5.5)$$

Capítulo 5. Diseño e implementación de Firmware

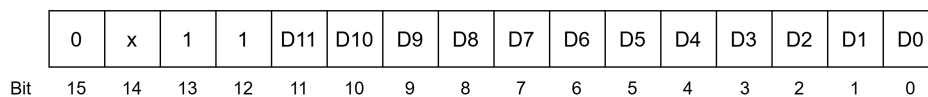


Figura 5.20: Registro del DAC que se escribe durante la primera y segunda fase de estimulación. D11-D0 corresponde al valor de *data_bytes* calculado en la ecuación 5.4.

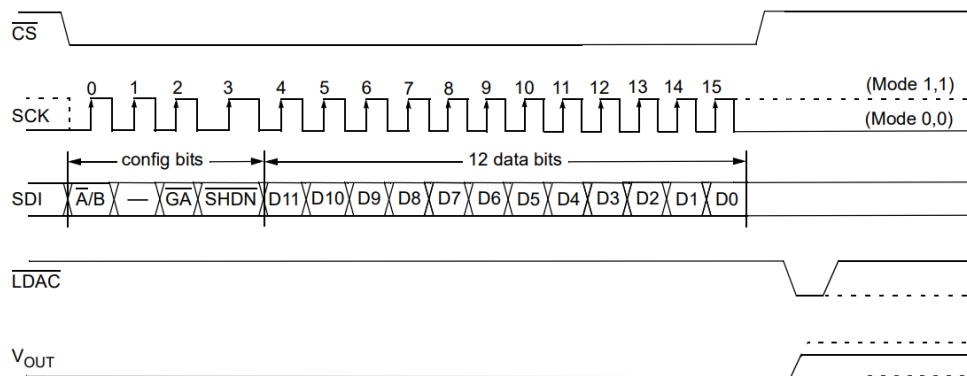


Figura 5.21: Diagrama de temporización de la comunicación SPI del DAC, mostrando la estructura de la trama de datos (bits de configuración y datos), así como las señales $\overline{CS}$, SCK, SDI y $\overline{LDAC}$. En la implementación realizada, la señal $\overline{LDAC}$ se encuentra fijada por hardware a nivel bajo, por lo que V_{OUT} se actualiza automáticamente al finalizar cada escritura, luego del flanco ascendente de $\overline{CS}$. Imagen extraída de la hoja de datos del DAC [15].

Una vez calculados los bits correspondientes a la tensión deseada, la trama de datos se construye una única vez al inicio de la terapia, ya que no se modifica durante su ejecución. El resultado se muestra en la Figura 5.20.

Adicionalmente, el integrado dispone del pin $\overline{LDAC}$, el cual, al llevarse a nivel bajo, permite transferir el valor almacenado en el registro de entrada hacia el registro de salida del DAC. Este comportamiento se ilustra en la Figura 5.21. En este trabajo, se optó por fijar por hardware la señal $\overline{LDAC}$ en nivel bajo, de modo que la actualización de la salida del DAC sea inmediata, minimizando posibles retardos. El control de la señal $\overline{CS}$ se realiza de acuerdo con el diagrama de temporización mostrado en la Figura 5.21.

5.4.7. Módulo adc

Se implementan en este módulo las funciones de inicialización, junto con la gestión de los canales del ADC y el control de los disparos de conversión del SAADC.

Para utilizar el periférico ADC del SoC se consideraron dos alternativas: emplear la API provista por Zephyr RTOS o utilizar directamente los drivers a través de la API `nrfx SAADC`.

El uso de la API de Zephyr presenta como principal ventaja la portabilidad del código, ya que permite su reutilización en cualquier plataforma compatible con Zephyr mediante la modificación del devicetree. Asimismo, facilita la integración con los mecanismos de bajo consumo que ofrece el sistema operativo.

No obstante, esta API no resulta adecuada en aplicaciones donde se requiere realizar muestreos en instantes de tiempo precisos, dado que el disparo de la conversión puede verse demorado por la ejecución de otras tareas o interrupciones de mayor prioridad. En estos casos, resulta más conveniente emplear la API `nrfx SAADC`, la cual permite un control más directo y determinista del periférico.

En este caso, la conversión debe realizarse en un instante temporal de alta precisión, ya que tiene lugar durante la primera fase de la estimulación, la cual puede presentar una duración mínima de $20 \mu s$. Por este motivo, se consideró más conveniente utilizar los drivers directamente, con el objetivo de lograr un mayor control sobre el momento en que se realiza la medición.

Como contrapartida, esta decisión implica una menor portabilidad hacia otras plataformas, así como la pérdida de las optimizaciones de consumo energético que ofrece Zephyr RTOS.

Se habilitó el uso del driver en el archivo de configuración del proyecto (`prj.conf`), mediante la opción `CONFIG_NRF_SAADC=y`. De este modo, se incluye la compilación de la librería `nrfx_saadc`.

Parámetros definidos

Dado que se utilizó la API `nrfx SAADC`, las muestras se obtienen en cuentas del ADC, por lo que se implementó la conversión a mV. Según se detalla en la especificación de producto del nRF5340 [13], la conversión se debe realizar según la ecuación:

$$Result = [V(P) - V(N)] * Gain / Reference * 2^{Resolution-m} \quad (5.6)$$

Donde:

- **V(P)** es la tensión de entrada en P.
- **V(N)** es la tensión de entrada en N.
- **Gain** es la ganancia seleccionada en el bloque SAADC.
- **m** es el modo, el cual toma el valor 0 si se configuró al bloque como *single ended*, y 1 si se configuró como *differential*.
- **Reference** es la referencia seleccionada en el bloque SAADC.

Cada uno de los parámetros de la ecuación 5.6 se corresponde con los elementos representados en la Figura 4.5, en la Sección 4.2.1.

En el sistema proporcionado se realizan dos medidas que utilizan el ADC: la medida de la batería y la medida de la tensión en V_{sink} . Cada canal cuenta con diferentes requerimientos, por lo que el bloque SAADC se configuró de acuerdo a las especificaciones de cada medida. En ambos casos, se optó por utilizar la referencia interna de 600 mV, así como el modo *single ended*, el cual como se muestra en el diagrama de bloques continúa siendo diferencial, pero se conecta internamente V(N) a GND. La escalera de resistencias está deshabilitada en ambos casos.

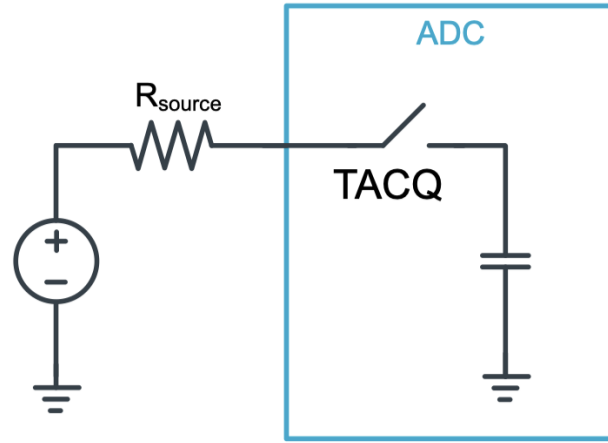


Figura 5.22: Circuito equivalente de la entrada a un pin analógico del nRF5340. *Imagen extraída de la Especificación del Producto del nRF340 conectado al ADC [13].*

En el caso de la medición de V_{sink} , la tensión a adquirir corresponde a $\frac{V_{sink}}{10}$, debido a que se implementó un divisor de tensión con un factor $\frac{1}{10}$ en la entrada analógica del nRF5340. Considerando que la tensión V_{sink} está dada por la ecuación 5.3, su valor máximo se alcanza cuando V_{anode} toma su valor máximo (15 V) y no circula corriente por la fuente, es decir, cuando $I = 0$. En dicha condición, V_{sink} es igual a 15 V, por lo que el valor máximo esperado en el canal analógico conectado a V_{sink_meas} es 1.5 V. Por lo tanto, para este canal se configuró una ganancia de $\frac{1}{3}$, logrando así un rango de entrada de 0 a 1.8 V de acuerdo a la ecuación 5.7 proveniente de la especificación de producto del nRF5340 [13].

$$Input\ range = (600\ mV)/(Gain) \quad (5.7)$$

Para la medición de V_{sink} , se configuró el tiempo de adquisición como el mínimo valor que permite una adquisición correcta, de acuerdo con las especificaciones del nRF5340.

La entrada del pin analógico puede modelarse como se muestra en la Figura 5.22. Debido a la presencia de un condensador de muestreo en la entrada, el tiempo durante el cual este debe permanecer conectado depende del valor de R_{source} . En este sentido, la especificación del SoC proporciona una tabla que establece el valor mínimo de T_{acq} en función de dicha resistencia.

Considerando el divisor resistivo implementado en la entrada, se obtiene un valor de $R_{source} = 27.1\ k\Omega$. Según la tabla de la especificación, para resistencias de hasta $40\ k\Omega$ se requiere un tiempo mínimo de adquisición de $T_{acq} = 5\ \mu s$, valor que se adopta en esta implementación.

Se selecciona el menor tiempo posible debido a que el tiempo total de adquisición de una muestra está dado por $T_{acq} + T_{conversion}$. En consecuencia, para anchos de pulso pequeños, utilizar tiempos de adquisición mayores podría introducir errores significativos en la medición.

Para la medición de VDD , no existe una restricción estricta sobre el tiempo de adquisición, por lo que se optó por utilizar el valor por defecto de $10\ \mu s$. En cuanto

a la ganancia, se seleccionó un valor de $\frac{1}{6}$, lo que permite alcanzar, de acuerdo con la ecuación 5.7, un rango de entrada máximo de 3.6 V, siendo que el nivel de batería en condiciones normales debe ser de aproximadamente 3.1 V.

Una diferencia relevante del sistema implementado con el nRF5340 respecto a la versión original es la posibilidad de realizar la medición de la tensión de batería de forma interna, sin necesidad de cablear la entrada del ADC a VDD .

Esta mejora presenta dos ventajas principales. En primer lugar, elimina la necesidad de utilizar un divisor resistivo externo y el correspondiente conexionado, reduciendo el área ocupada en el circuito. En segundo lugar, en la implementación original la medición se realizaba a partir de V_{anode} , cuyo valor durante la estimulación puede corresponder tanto a V_{BAT} como a la salida del convertidor *boost*. Esto implica que la medición de batería solo podía efectuarse en períodos en los que no se estuviera estimulando.

En contraste, la capacidad del nRF5340 de realizar esta medición de forma interna, sin interferir con el funcionamiento del sistema, permite monitorear la tensión de batería incluso durante la estimulación. Esto habilita la implementación de mecanismos de protección, como la detección de batería baja en tiempo real y la finalización de la terapia en caso de que la tensión descienda por debajo de un umbral definido.

Implementación de la medida

Como se mencionó previamente, la implementación del módulo de adquisición se realizó utilizando *nrfx_saadc.h*, la biblioteca provista por Nordic Semiconductor para el manejo del ADC. Esta decisión se fundamenta en los requerimientos temporales de la aplicación, ya que permite un control más preciso de los tiempos de conversión en comparación con otras abstracciones de mayor nivel.

Al inicio de la primera fase del pulso de estimulación, se dispara una conversión del ADC en todos los canales habilitados. Dado que la única variable de interés en este caso es la tensión V_{sink} , en el *handler* de fin de conversión se genera un evento que indica la disponibilidad de una nueva muestra de dicha señal.

El procesamiento de esta medida se difiere y se ejecuta únicamente cuando la CPU dispone de tiempo disponible, evitando interferir con las tareas críticas de temporización. En particular, este procesamiento se realiza durante la etapa de *Passive balance* que presenta una mayor duración y los requerimientos temporales son menos estrictos.

Durante la implementación, se observó que la adquisición de muestras de V_{sink} introduce un *jitter* apreciable en la temporización de las fases de estimulación, en comparación con aquellas en las que no se realiza la medición. Para mitigar este efecto, se implementó una calibración de los anchos temporales de los estados de estimulación, compensando el retardo introducido por la conversión ADC. Dado que esta variación resultó ser consistente, la corrección aplicada permitió mantener la temporización dentro de los márgenes requeridos.

5.4.8. Módulo battery

Este módulo se encarga de procesar la medición de la tensión de la batería proveniente del ADC. Cuenta con una única función responsable de convertir la medida a milivoltios,

utilizando la ecuación 5.6, y de evaluar si dicha tensión se encuentra por debajo del umbral definido para considerar la batería como baja.

En caso de que la medición se encuentre por debajo de este umbral, se incrementa un contador que registra la cantidad de mediciones consecutivas en condición de batería baja. Posteriormente, se verifica si dicho contador alcanza el número de muestras requerido para confirmar este estado. De ser así, se genera un evento de batería baja y se reinicia el contador. Este evento permite desencadenar la detención de la terapia.

Por otro lado, si la medición se encuentra por encima del umbral establecido, el contador se reinicia. De esta manera, se garantiza que las mediciones de batería baja sean consecutivas, implementando un mecanismo de histéresis.

5.4.9. Módulo magnet

Se incorpora este módulo con el objetivo de añadir una funcionalidad de seguridad para el paciente. Permite detener la terapia de forma inmediata sin necesidad de utilizar la interfaz BLE, ya sea por decisión del usuario o ante una eventual falla del sistema. Para ello, se contempla el uso de un imán como mecanismo de finalización de la terapia.

El módulo magnet se encarga de inicializar el periférico correspondiente y de implementar el *callback* asociado a la interrupción generada por la detección del imán. Ante este evento, se genera una señal interna que indica la presencia del imán y desencadena la detención de la terapia.

En la implementación actual, y debido a limitaciones en el proceso de fabricación, se utilizó un botón en lugar de un *reed-switch* para emular esta funcionalidad.

5.5. Configuración y compilación del proyecto en Zephyr

Como se mencionó en la Sección 3.3, una aplicación desarrollada sobre Zephyr se compone de distintos archivos de configuración y construcción que definen tanto el comportamiento del sistema como el proceso de compilación. En esta Sección se describen los principales archivos utilizados en el desarrollo de este proyecto, así como las configuraciones más relevantes realizadas.

5.5.1. Devicetree

Como se menciona en la Sección 3.3.2, nRF Connect SDK utiliza la estructura *Devicetree* para la descripción del *hardware* del sistema. Para el PCB diseñado, se implementó un archivo `.overlay`¹, en el cual se deshabilitaron todos los periféricos no utilizados, con el objetivo de minimizar el consumo energético, tales como I2C, UART, QSPI y PWM.

Esto se realiza como se muestra en la Figura 5.23, donde *nombre* representa cualquiera de los periféricos mencionados previamente, y *X* indica la instancia específica que se desea deshabilitar.

¹https://gitlab.fing.edu.uy/mauricio.guerrero/neurosoc/-/blob/sistema-final/neurosoc_integracion/boards/nrf5340dk_nrf5340_cpuapp.overlay?ref_type=heads

```
&nombreX {
    status = "disabled";
};
```

Figura 5.23: Estructura de datos utilizada para deshabilitar un periférico en el `.overlay`.

```
/ {
    electrodes {
        compatible = "gpio-leds";

        e1_neg: e1_neg {
            gpios = <&gpio0 19 (GPIO_PULL_DOWN | GPIO_ACTIVE_HIGH)>;
        };
    }
}
```

Figura 5.24: Estructura utilizada para definir la asignación de una señal en el archivo `.overlay`.

Adicionalmente, se debe definir la asignación de cada una de las señales utilizadas a los pines físicos del microcontrolador. En la Figura 5.24 se presenta como ejemplo la configuración de la señal *e1_neg*, donde *electrodes* corresponde a un nodo definido para agrupar este conjunto de señales y facilitar su acceso.

La propiedad `compatible = "gpio-leds"`; permite asociar dicho conjunto de señales a un *binding*, tal como se describe en la Sección 3.3.2. En particular, `gpio-leds` es un *driver* genérico de Zephyr que permite manejar pines como salidas digitales de forma sencilla.

En este ejemplo, la señal *e1_neg* se asigna al pin 19 del puerto 0 (`&gpio0 19`). Las banderas `GPIO_PULL_DOWN | GPIO_ACTIVE_HIGH` indican, respectivamente, que el pin se configura con una resistencia de *pull-down* interna y que la señal es activa en nivel alto.

Esta estructura se replica para todas las señales utilizadas en el PCB, modificando según corresponda el puerto, el número de pin y las banderas de configuración. En el caso de los botones, se emplean las banderas `GPIO_PULL_UP | GPIO_ACTIVE_LOW`, lo que implica el uso de una resistencia de *pull-up* interna y una lógica activa en nivel bajo.

5.5.2. prj.conf

Como se menciona en la Sección 3.3.1, el archivo `prj.conf`² define las opciones de configuración de la aplicación, permitiendo habilitar *drivers*, ajustar funcionalidades y establecer parámetros del sistema y sus distintos subsistemas.

Para habilitar los *drivers* de los periféricos utilizados, de modo que se incluyan en la compilación las bibliotecas correspondientes, se agregaron las configuraciones de la Figura 5.25 para el ADC, el temporizador y la interfaz SPI, respectivamente.

²https://gitlab.fing.edu.uy/mauricio.guerrero/neurosoc/-/blob/sistema-final/neurosoc_integracion/prj.conf?ref_type=heads

Capítulo 5. Diseño e implementación de Firmware

```
CONFIG_NRF52833_SAADC=y
CONFIG_NRF52833_TIMER1=y
CONFIG_NRF52833_SPIM1=y
```

Figura 5.25: Configuración de *drivers* de periféricos en `prj.conf`.

```
CONFIG_CLOCK_CONTROL=y
CONFIG_CLOCK_CONTROL_NRF=y
```

Figura 5.26: Configuración del sistema de relojes.

Asimismo, para permitir la utilización del reloj de alta frecuencia (HFCLK) en el temporizador, se habilitó el módulo de control de relojes mediante las opciones de la Figura 5.26.

Por otra parte, para la implementación de la comunicación inalámbrica, se habilitó un conjunto de configuraciones correspondientes a las pilas de BLE y NFC provistas por el nRF Connect SDK, como se detalla en la Figura 5.27.

En el caso de NFC, se incluyeron las bibliotecas necesarias para la generación y el procesamiento de mensajes NDEF (*NFC Data Exchange Format*), así como módulos adicionales para soportar distintos tipos de registros y mecanismos de intercambio de datos.

En cuanto a BLE, se habilitó la pila de Bluetooth en modo periférico, permitiendo que el dispositivo pueda ser detectado y conectado por otros dispositivos centrales. Asimismo, se activó el soporte para el protocolo de seguridad SMP (*Security Manager Protocol*), encargado de gestionar el emparejamiento, la autenticación y el cifrado de las comunicaciones. Adicionalmente, se configuró el nombre con el cual el dispositivo es identificado durante el proceso de descubrimiento.

Finalmente, para permitir el almacenamiento de datos en memoria persistente, como se detalla en secciones posteriores, se habilitaron las configuraciones de la Figura 5.28.

5.5.3. CMake

Se genera un archivo `CMakeLists.txt`³, el cual define cómo se construye el proyecto, según lo mencionado en la Sección 3.3.3. Para ello, se incorpora la instrucción `find_package(Zephyr REQUIRED HINTS $ENV{ZEPHYR_BASE})`, que permite localizar e integrar el entorno Zephyr en el sistema de compilación, asegurando la inclusión de sus módulos, sistema de configuración y herramientas de construcción.

Adicionalmente, en este archivo se especifican los archivos fuente que forman parte de la aplicación mediante la instrucción `target_sources(app PRIVATE src/main.c ...)`, donde se enumeran todos los archivos `.c` utilizados en el proyecto.

³https://gitlab.fing.edu.uy/mauricio.guerrero/neurosoc/-/blob/sistema-final/neurosoc_integracion/CMakeLists.txt?ref_type=heads

5.5. Configuración y compilación del proyecto en Zephyr

```
CONFIG_NFC_T4T_NRFXMLIB=y
CONFIG_NFC_NDEF=y
CONFIG_NFC_NDEF_MSG=y
CONFIG_NFC_NDEF_RECORD=y
CONFIG_NFC_NDEF_LE_OOB_REC=y
CONFIG_NFC_NDEF_CH_MSG=y
CONFIG_NFC_NDEF_TNEP_RECORD=y
CONFIG_NFC_TNEP_TAG=y
CONFIG_NFC_NDEF_PARSER=y
CONFIG_NFC_NDEF_CH_PARSER=y
CONFIG_NFC_NDEF_LE_OOB_REC_PARSER=y
CONFIG_NFC_TNEP_CH=y

CONFIG_BT=y
CONFIG_BT_SMP=y
CONFIG_BT_SMP_APP_PAIRING_ACCEPT=y
CONFIG_BT_PERIPHERAL=y
CONFIG_BT_DEVICE_NAME="NEUROSOC"
```

Figura 5.27: Configuración de las pilas de NFC y BLE.

```
CONFIG_BT_SETTINGS=y
CONFIG_FLASH=y
CONFIG_FLASH_PAGE_LAYOUT=y
CONFIG_FLASH_MAP=y
```

Figura 5.28: Configuración para almacenamiento en memoria persistente.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Resultados

En el presente capítulo se detalla el procedimiento de validación del funcionamiento de la plataforma de neuroestimulación, tanto a nivel de *firmware* como de *hardware*, y se verifica el cumplimiento de los requerimientos definidos en la Sección 2.2.

En primer lugar, se analiza la generación de los pulsos de estimulación en el circuito del sistema, relevando mediciones de amplitud y temporización. En este contexto, la capacidad del neuroestimulador de mantenerse dentro de los rangos especificados durante cada etapa de la terapia resulta crítica, ya que garantiza la seguridad del paciente. Asimismo, se estudian las formas de onda generadas y distintas señales de interés en los bloques de estimulación y de alimentación, así como el retardo observado.

Dado que tanto el diseño del *layout* del PCB como diversas modificaciones en los componentes electrónicos fueron realizados durante el proyecto para la adaptación del SoC nRF5340, se analizan específicamente los posibles impactos de estas decisiones de diseño en el comportamiento del sistema.

Posteriormente, se valida el proceso de comunicación, lo cual incluye la verificación del funcionamiento esperado en las etapas de autenticación y conexión mediante BLE, así como la correcta configuración de los parámetros de estimulación a través de dicha interfaz.

Adicionalmente, se consideran distintos escenarios de falla con el objetivo de evaluar la seguridad del diseño (*safety*) y su respuesta ante condiciones anómalas.

Por último, se estudia el perfil de consumo energético del sistema en diferentes escenarios de operación, dado que este constituye un aspecto fundamental en sistemas médicos implantables activos. Este análisis permite no solo verificar el desempeño energético del dispositivo, sino también establecer una base de comparación con otras implementaciones y servir como referencia para futuras optimizaciones.

6.1. Generación de pulsos de estimulación

Se relevaron y analizaron datos para distintos conjuntos de parámetros de estimulación asociados a terapias utilizadas en diversas aplicaciones clínicas, con el fin de obtener una primera métrica del desempeño del sistema. En esta subsección, se presentan mediciones experimentales de las formas de onda generadas por el sistema para distintos

Capítulo 6. Resultados

conjuntos de parámetros representativos de las terapias mencionadas, evaluando su correspondencia con los valores esperados y su cumplimiento con las especificaciones de diseño.

Metodología de medición

Para la realización de las mediciones, se utilizó la placa diseñada en el marco del proyecto, empleando como fuente de alimentación para la fuente de corriente el *boost* del *test board* provisto, debido a que el implementado en la placa desarrollada presenta niveles de ruido elevados.

Para modelar el tejido, se utilizó una resistencia de $500\ \Omega$, ya que es el valor empleado en la referencia utilizada para la definición de las terapias [3].

En una primera instancia, se midió el ancho de pulso correspondiente a los valores programados. Se observó que, en todos los casos, existía una diferencia sistemática respecto al valor configurado. Este desvío se atribuye a los retardos introducidos por las funciones de control encargadas de la generación de la señal de estimulación. A partir de este resultado, se implementó una calibración del sistema, consistente en programar el tiempo en cada estado con un valor menor al deseado, con el fin de obtener en la salida el valor objetivo. Una vez aplicada esta corrección, se procedió a la medición del resto de los parámetros de la terapia, verificándose que los mismos se encuentran dentro de los rangos de aceptación establecidos tomados de los requerimientos.

Al momento de realizar las mediciones se observó que los intervalos en los que se adquiere la señal V_{sink} afectan la temporización de los estados. Por este motivo, las mediciones y el proceso de calibración se llevaron a cabo evitando la adquisición de V_{sink} , mientras que el retardo introducido por dicha medición se analizó de forma independiente.

Por otro lado, se evaluó la latencia introducida por el proceso de adquisición del ADC durante la toma de mediciones. En particular, el tiempo de adquisición de la tensión de batería es de aproximadamente $10\ \mu s$, mientras que la medición de la tensión V_{sink} requiere alrededor de $5\ \mu s$, ya que cada disparo del ADC realiza la adquisición de ambos canales. A estos tiempos debe sumarse la latencia asociada a la ejecución del controlador del ADC y a la planificación del sistema operativo, así como la derivada de los cambios de contexto gestionados por el *scheduler*.

Se observó que esta sobrecarga temporal afecta de manera significativa a los pulsos de menor duración, en particular a aquellos con anchos de $20\ \mu s$ y $50.5\ \mu s$, ya que la adquisición puede representar una fracción considerable de la primera fase. Por este motivo, para garantizar el cumplimiento de los requerimientos temporales en terapias que emplean estos anchos de pulso, se decidió deshabilitar la medición de V_{sink} durante la estimulación. En estas condiciones, la tensión de salida del *boost* se mantiene fija en su valor máximo de 15 V. Esta decisión implica un compromiso entre eficiencia energética y precisión temporal, privilegiándose esta última para asegurar la correcta generación de la forma de onda de estimulación.

6.1. Generación de pulsos de estimulación

Tabla 6.1: Comparación entre valores nominales, programados y medidos de los parámetros de estimulación para terapia VNS.

Parámetro	Valor nominal	Valor programado	Valor medido	Rango de aceptación
Ancho de pulso [μs]	250.0	264.0	264,0	[261.4; 266.6]
Frecuencia [Hz]	20.00	20.00	20.04	[19.80; 20.20]
Amplitud [mA]	1.50	1.50	1.44	[1.40; 1.60]

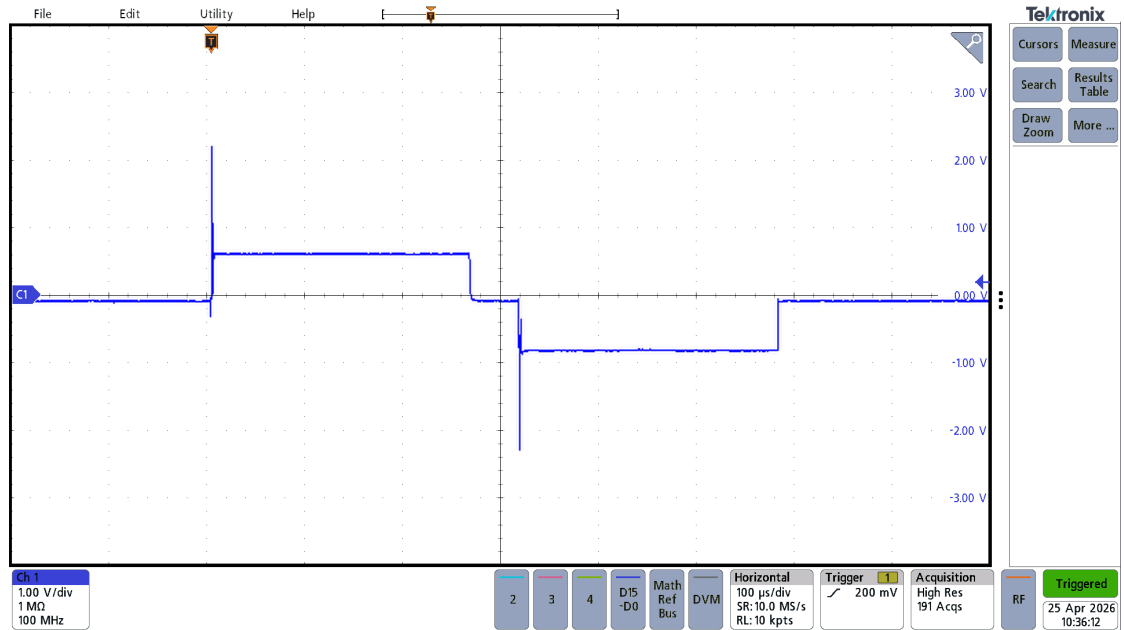


Figura 6.1: Forma de onda de estimulación para terapia VNS - Epilepsia, medida con osciloscopio.

Terapia VNS (Vagus Nerve Stimulation) – Epilepsia

El nervio vago es el principal componente del sistema nervioso parasimpático, encargado de regular funciones corporales involuntarias como la digestión, la frecuencia cardíaca y ciertos procesos del sistema inmunológico [21]. La estimulación de este nervio permite modular diversas respuestas fisiológicas, incluyendo la actividad del sistema inmune. En este sentido, se ha utilizado en el tratamiento de enfermedades como la artritis reumatoide, así como en aplicaciones clínicas tales como la rehabilitación posterior a accidentes cerebrovasculares (ACV) y la prevención de crisis epilépticas, entre otras [3].

En la Tabla 6.1 se presentan los parámetros de estimulación considerados para el tratamiento de la epilepsia, incluyendo sus valores nominales, los valores programados ajustados según los incrementos permitidos, los valores medidos y las tolerancias aceptadas. En la Figura 6.1 se muestra la forma de onda generada para esta terapia, medida con osciloscopio.

Capítulo 6. Resultados

Tabla 6.2: Comparación entre valores nominales, programados y medidos de los parámetros de estimulación para terapia SCS.

Parámetro	Valor nominal	Valor programado	Valor medido	Rango de aceptación
Ancho de pulso [μ s]	347.0	355.5	354.4	[351.9; 359.1]
Frecuencia [Hz]	39.20	40.00	40.19	[39.60; 40.40]
Amplitud [mA]	3.60	3.50	3.48	[3.30; 3.70]

Terapia SCS (Spinal Cord Stimulation) - Dolor crónico

La médula espinal es una estructura tubular del sistema nervioso central encargada de conducir señales nerviosas entre el cerebro y el resto del cuerpo. A través de estas vías se transmiten tanto impulsos motores, que permiten el movimiento, como señales sensitivas y funciones autónomas, como la respiración. En consecuencia, las lesiones en la médula espinal pueden provocar déficits motores, alteraciones sensoriales y compromiso de diversas funciones fisiológicas [22].

Los estimuladores de la médula espinal (*Spinal Cord Stimulators, SCS*) se utilizan comúnmente en el tratamiento del dolor crónico. Ante una lesión, las señales de dolor se transmiten hacia el cerebro a través de la médula espinal. Estos dispositivos actúan aplicando estímulos eléctricos que logran bloquear dicha transmisión, reduciendo así la percepción del dolor [3].

Como se detalla en la Tabla 6.2, la terapia SCS emplea una frecuencia de 39.2 Hz, un ancho de pulso de 347 μ s y una amplitud de 3.6 mA [3]. Para la validación del neuroestimulador desarrollado en este proyecto, se utilizaron los valores programables más cercanos a los especificados, los cuales también se presentan en la tabla.

En la Figura 6.2 se muestra la forma de onda correspondiente a esta terapia.

Terapia PNS (Phrenic Nerve Stimulation) - Apnea central del sueño

El nervio frénico desempeña un rol fundamental en la respiración, ya que es responsable de transmitir las señales motoras que controlan la contracción del diafragma. La activación de este músculo produce su descenso, lo que genera la expansión de la cavidad torácica y permite la entrada de aire en los pulmones. Por el contrario, cuando el diafragma se relaja, asciende, reduciendo el volumen torácico y facilitando la expulsión del aire [23].

La terapia de estimulación del nervio frénico se emplea en el tratamiento de la apnea central del sueño, un trastorno caracterizado por interrupciones intermitentes en la señal neural que regula la respiración durante el sueño. En este contexto, se aplican pulsos de estimulación sobre los nervios frénicos con el objetivo de inducir la contracción del diafragma y asistir el proceso respiratorio [3].

Tal como se indica en la Tabla 6.3, la terapia PNS se caracteriza por una frecuencia de 40 Hz, un ancho de pulso de 300 μ s y una amplitud de 1.0 mA [3]. Para la validación del neuroestimulador desarrollado, se emplearon los valores programables más próximos a estos parámetros, los cuales también se incluyen en la tabla.

6.1. Generación de pulsos de estimulación

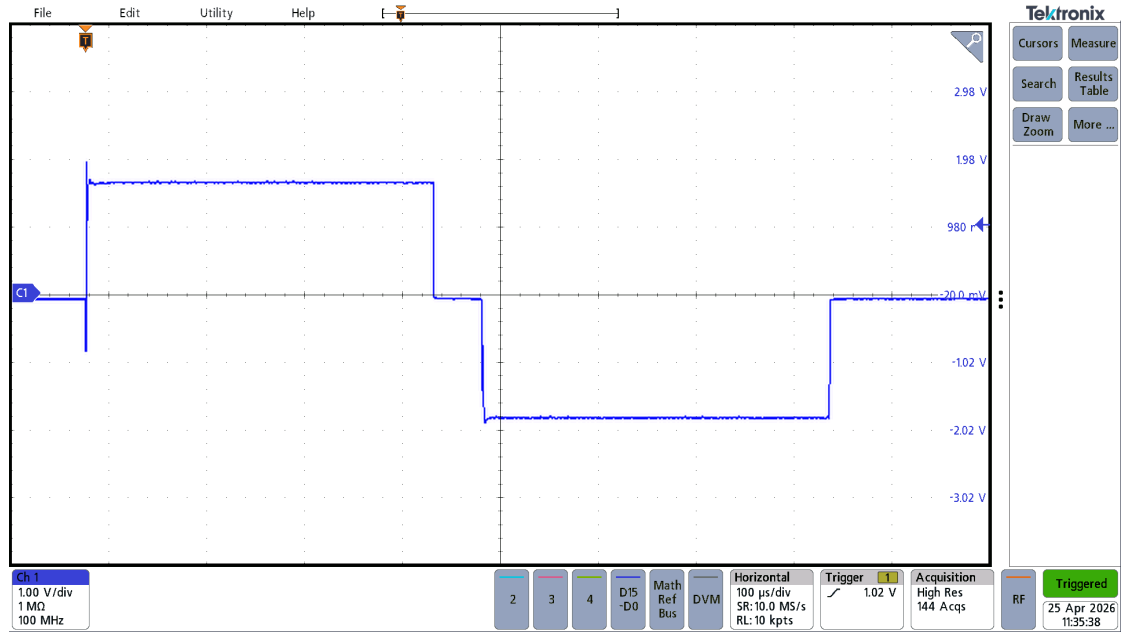


Figura 6.2: Forma de onda de estimulación para terapia SCS medida con osciloscopio.

Tabla 6.3: Comparación entre valores nominales, programados y medidos de los parámetros de estimulación para terapia PNS.

Parámetro	Valor nominal	Valor programado	Valor medido	Rango de aceptación
Ancho de pulso [μs]	300.0	294.5	294.4	[291.6; 297.4]
Frecuencia [Hz]	40.00	40.00	40.06	[39.60; 40.40]
Amplitud [mA]	1.00	1.00	0.94	[0.90; 1.10]

Al igual que en los resultados previamente presentados, en la Figura 6.3 se muestra la forma de onda correspondiente a este pulso.

Terapia RNS (Responsive Neurostimulation)

Los estimuladores DBS (*Deep Brain Stimulation*) se utilizan para el tratamiento de enfermedades como Parkinson, epilepsia y temblor esencial, entre otras. Un caso particular dentro de estas terapias es la estimulación responsiva (RNS), utilizada específicamente para el tratamiento de la epilepsia [3].

Para llevar a cabo la terapia, se implantan electrodos en distintas regiones del cerebro, los cuales generan impulsos eléctricos que modulan la actividad neuronal. Asimismo, estos estímulos pueden influir en los circuitos neuronales y en los procesos neuroquímicos asociados a estas patologías [24].

Tal como se presenta en la Tabla 6.4, la terapia RNS se define por una frecuencia de 200 Hz, un ancho de pulso de 200 μs y una amplitud de 12.0 mA [3]. Con el objetivo de validar el neuroestimulador desarrollado, se seleccionaron los valores configurables más

Capítulo 6. Resultados

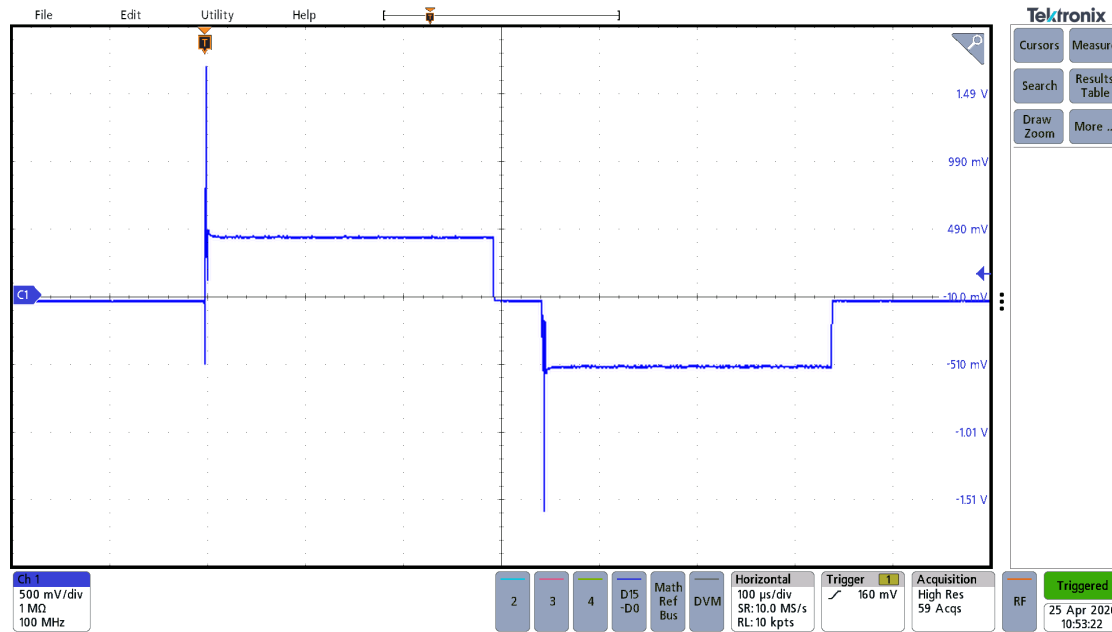


Figura 6.3: Forma de onda de estimulación para terapia PNS medida con osciloscopio.

Tabla 6.4: Comparación entre valores nominales, programados y medidos de los parámetros de estimulación para terapia RNS.

Parámetro	Valor nominal	Valor programado	Valor medido	Rango de aceptación
Ancho de pulso [μ s]	200.0	203.0	202.8	[201.0; 205.0]
Frecuencia [Hz]	200.0	200.0	200.8	[198.0; 202.0]
Amplitud [mA]	12.0	12.0	12.1	[11.4; 12.6]

cercanos a estos parámetros, los cuales también se detallan en la tabla.

Asimismo, en la Figura 6.4 se observa el pulso capturado en el osciloscopio correspondiente a la terapia RNS.

Prueba de desempeño en condiciones límite

Por último, con el objetivo de evaluar el desempeño del sistema en condiciones límite desde el punto de vista temporal —esto es, con el menor ancho de pulso y la mayor frecuencia— se programó el neuroestimulador utilizando el ancho de pulso mínimo y la frecuencia máxima disponibles.

En la Tabla 6.5 se presentan los valores medidos en esta condición, mientras que en la Figura 6.5 se muestra la forma de onda obtenida para dicha configuración. Además, en la Figura 6.6 se presenta una captura de la señal obtenida con el osciloscopio, donde se observa la frecuencia de los pulsos generados.

6.1. Generación de pulsos de estimulación

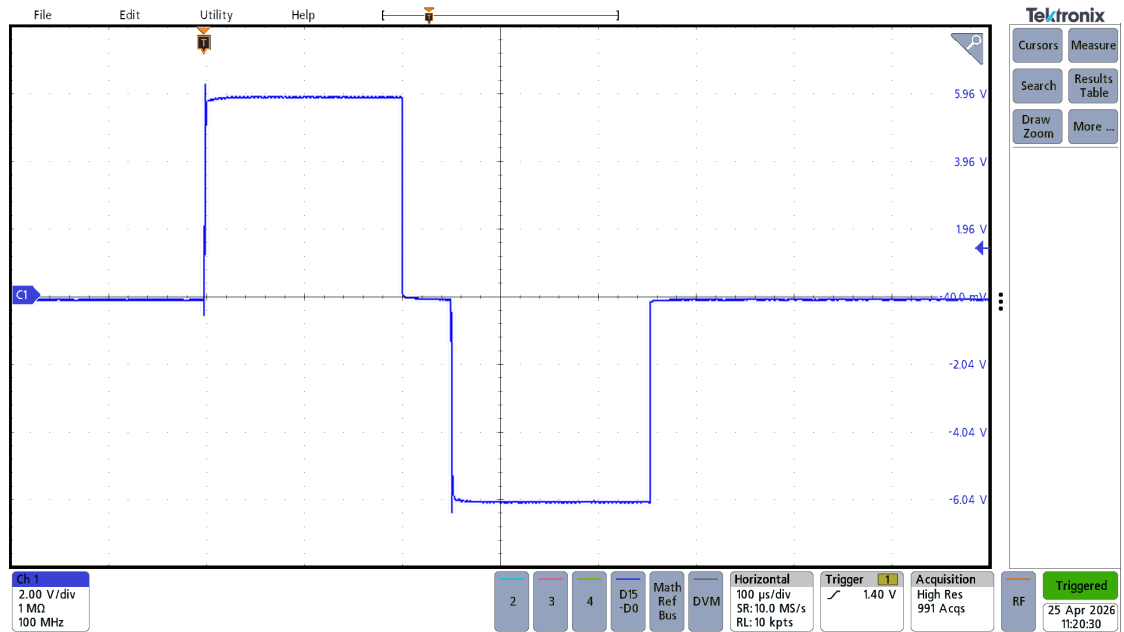


Figura 6.4: Forma de onda de estimulación para terapia RNS medida con osciloscopio.

Tabla 6.5: Comparación entre valores programados y medidos de los parámetros de estimulación para la prueba de desempeño en condiciones límite.

Parámetro	Valor programado	Valor medido	Rango de aceptación
Ancho de pulso [μs]	20.00	20.08	[19.00; 20.00]
Frecuencia [Hz]	1500	1490	[1463; 1536]
Amplitud [mA]	2.00	1.96	[1.90; 2.10]

Caracterización del desempeño temporal: *jitter*

Uno de los objetivos del proyecto fue evaluar el *jitter* presente en los pulsos de estimulación. Para ello, se distinguieron dos contribuciones principales: por un lado, el *jitter* asociado al uso de Zephyr, y por otro, el *jitter* introducido durante los intervalos en los que se realiza la adquisición de la señal V_{sink} en la primera fase del pulso.

En la Figura 6.7a se presenta el *jitter* medido en la primera fase, correspondiente al introducido por Zephyr. Por su parte, en la Figura 6.7b se muestra el *jitter* observado en el intervalo interpulso durante los períodos en los que se adquiere la señal V_{sink} .

En la Tabla 6.6 se resumen los valores máximos de *jitter* medidos, tanto en los casos en los que se realiza la adquisición de V_{sink} como en aquellos en los que no, para cada una de las terapias previamente descritas. Notar que, para el caso de un ancho de pulso de $20 \mu\text{s}$, no se realizaron mediciones de V_{sink} , ya que el tiempo requerido para la adquisición supera la duración del pulso.

Capítulo 6. Resultados

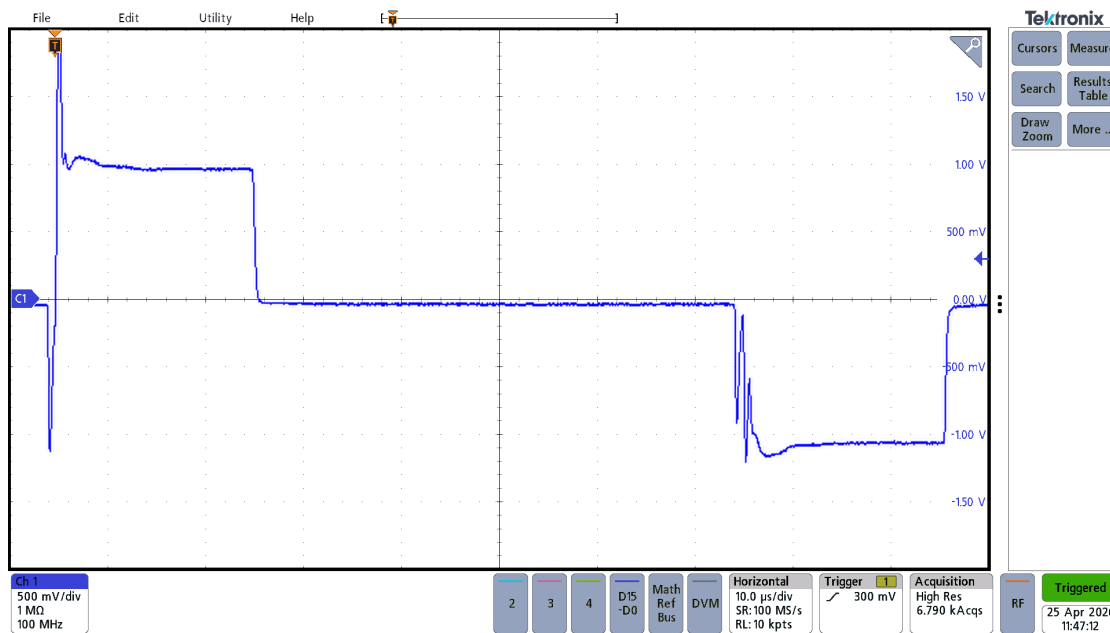


Figura 6.5: Forma de onda de estimulación para pulso con condiciones límite.

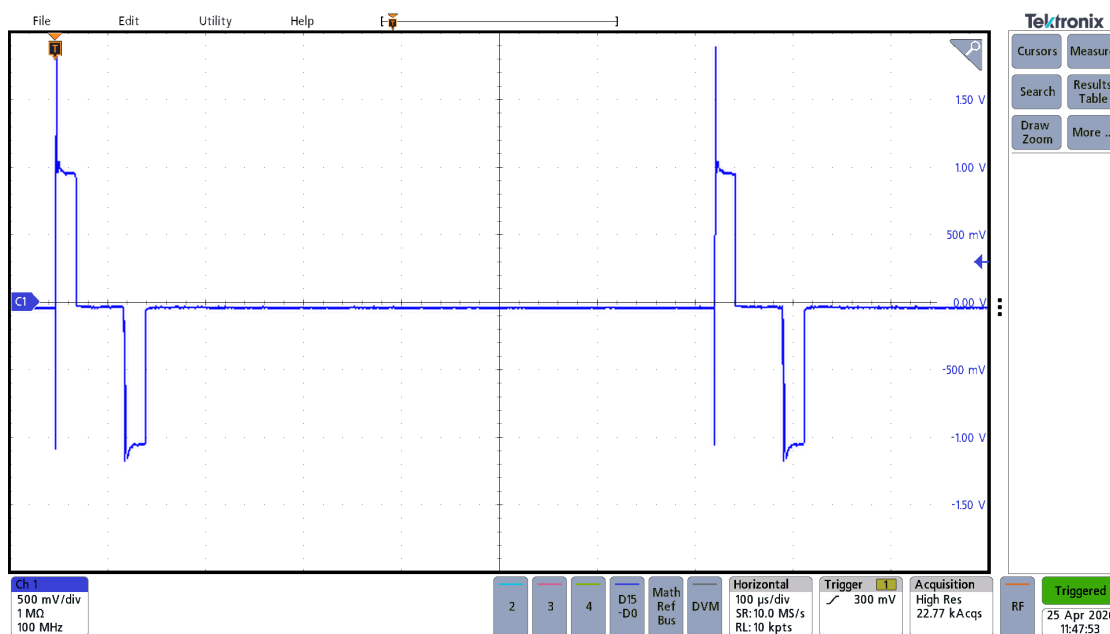
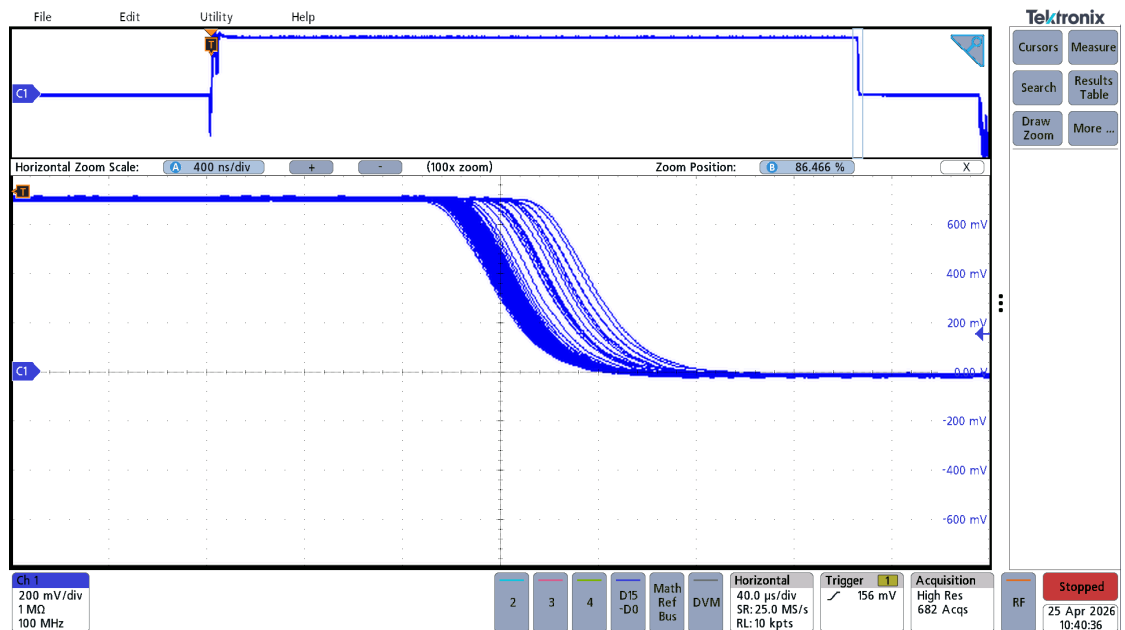
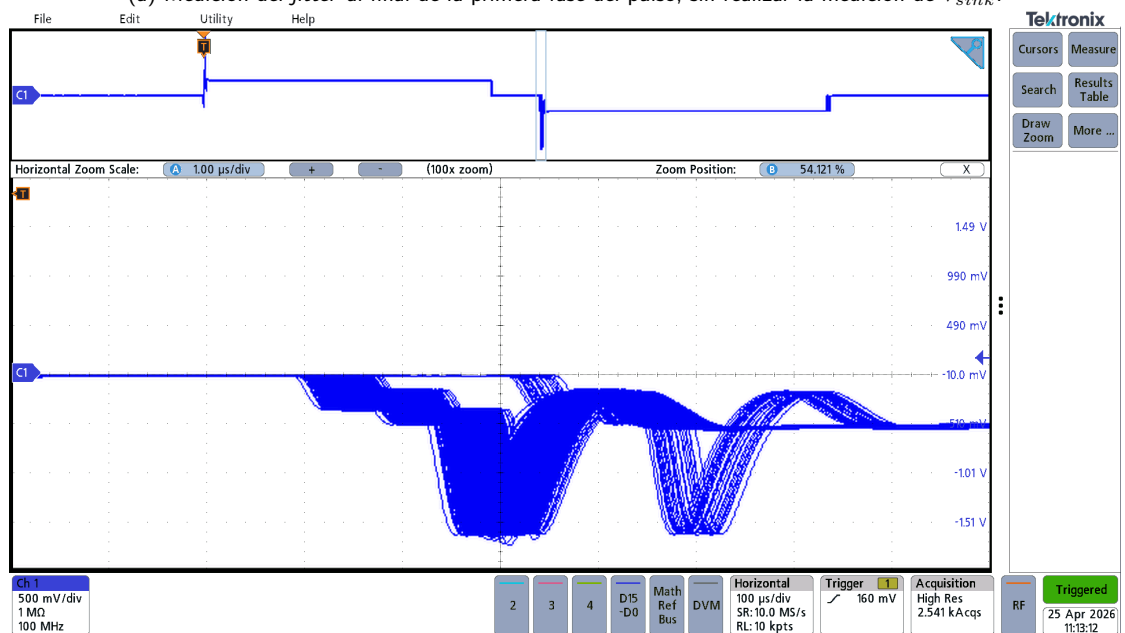


Figura 6.6: Frecuencia observada en el osciloscopio para la configuración de terapia con máxima frecuencia programable.

6.1. Generación de pulsos de estimulación



(a) Medición del *jitter* al final de la primera fase del pulso, sin realizar la medición de V_{sink} .



(b) Medición del *jitter* al final del intervalo interpulso, realizando la medición de V_{sink} .

Figura 6.7

Tabla 6.6: Jitter máximo medido para distintas terapias

Terapia	Jitter Zephyr (fase 1) [ns]	Jitter medición V_{sink} (fase 1) [ns]	Jitter medición V_{sink} (interpulso) [μ s]
VNS	420	836	2.00
SCS	840	1016	3.74
PNS	752	1020	2.15
RNS	500	668	2.52
Condiciones límite	1010	-	-

6.1.1. Análisis de datos relevados

A partir de las Tablas 6.1, 6.2, 6.3, 6.4 y 6.5, se observa que todos los valores medidos de los parámetros de estimulación se encuentran dentro de los márgenes de tolerancia especificados. Esto valida el correcto desempeño del dispositivo para las distintas terapias evaluadas.

Asimismo, las formas de onda presentadas en las Figuras 6.1, 6.2, 6.3, 6.4 y 6.5 presentan la morfología esperada para la señal de estimulación, identificándose claramente la primera fase del pulso, el intervalo interpulso y la segunda fase.

No obstante, se observan picos transitorios al inicio de la primera fase y durante el balance activo. La amplitud de estos transitorios varía según la terapia aplicada y se atribuye a los fenómenos de carga y descarga de las capacitancias parásitas asociadas a los elementos de conmutación del circuito. En particular, estos transitorios se producen durante la transición a nivel alto de la señal de control de la llave de la fuente de corriente, que conecta la fuente al cátodo. Como consecuencia, se genera una respuesta transitoria de aproximadamente 600 ns antes de que la señal alcance el régimen estacionario.

6.2. Validación de la comunicación BLE

Con el objetivo de validar el funcionamiento de la interfaz de comunicación inalámbrica implementada, se realizó un análisis basado en capturas de paquetes BLE. Esta validación permite contrastar, a nivel de tráfico inalámbrico, el establecimiento de una conexión segura y confiable entre el neuroestimulador y un dispositivo móvil.

Las capturas fueron realizadas utilizando el *nRF Sniffer for Bluetooth LE* de Nordic Semiconductor. Esta herramienta permite observar en *Wireshark* los paquetes BLE transmitidos en el aire entre un dispositivo central y un periférico. En el montaje experimental se utilizó un *nRF52840 Dongle* programado con el *firmware* del sniffer, conectado a una computadora donde se ejecutó *Wireshark* con el perfil correspondiente para BLE.

En este ensayo, el periférico BLE corresponde a la PCB del neuroestimulador, mientras que el dispositivo central corresponde al teléfono celular que ejecuta la aplicación de configuración *nRF Connect*.

6.2. Validación de la comunicación BLE

6	5.008	f1:9c:9f:03:48:cd	LE 1M	LE LL	19	292µs			ADV_IND
7	10.014	f1:9c:9f:03:48:cd	LE 1M	LE LL	19	50056µs			ADV_IND
8	10.015	47:e6:be:89:15:fe	LE 1M	LE LL	12	150µs			SCAN_REQ
9	10.015	f1:9c:9f:03:48:cd	LE 1M	LE LL	6	149µs			SCAN_RSP
10	10.015	f1:9c:9f:03:48:cd	LE 1M	LE LL	19	91µs			ADV_IND
11	10.016	f1:9c:9f:03:48:cd	LE 1M	LE LL	19	293µs			ADV_IND
12	15.015	f1:9c:9f:03:48:cd	LE 1M	LE LL	19	49988µs			ADV_IND
13	15.015	f1:9c:9f:03:48:cd	LE 1M	LE LL	19	294µs			ADV_IND
14	15.016	47:e6:be:89:15:fe	LE 1M	LE LL	34	150µs			CONNECT_IND
15	15.032	Central_0x44de1ebb	LE 1M	LE LL	9	16470µs	0	0	False Control Opcode: LL_FEATURE_REQ
16	15.033	Peripheral_0x44de1ebb	LE 1M	LE LL	0	150µs	0	1	False Empty PDU
17	15.077	Central_0x44de1ebb	LE 1M	LE LL	0	44619µs	1	1	False Empty PDU


```

> Frame 14: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface COM16-4.4, id 0
> nRF Sniffer for Bluetooth LE
v Bluetooth Low Energy Link Layer
  Access Address: 0x8e89bed6
  v Packet Header: 0x22e5 (PDU Type: CONNECT_IND, ChSel: #2, TxAdd: Random, RxAdd: Random)
    ... 0101 = PDU Type: 0x5 CONNECT_IND
    ...0 .... = Reserved: 0
    ..1. .... = Channel Selection Algorithm: #2
    .1.. .... = Tx Address: Random
    1... .... = Rx Address: Random
    Length: 34
    Initiator Address: 47:e6:be:89:15:fe (47:e6:be:89:15:fe)
    Advertising Address: f1:9c:9f:03:48:cd (f1:9c:9f:03:48:cd)
  v Link Layer Data
    Access Address: 0x44de1ebb
    CRC Init: 0x1dc04c
    Window Size: 2 (2,5 msec)
    Window Offset: 11 (13,75 msec)
    Interval: 36 (45 msec)
    Latency: 0
    Timeout: 500 (5000 msec)
  > Channel Map: ffff0f801f
    ...0 1101 = Hop: 13
    101. .... = Sleep Clock Accuracy: 31 ppm to 50 ppm (5)
    CRC: 0x7741d3

```

Figura 6.8: Captura de tráfico BLE, donde se observa la secuencia de establecimiento de conexión entre el dispositivo central y el periférico NeuroSoC. Inicialmente se registran paquetes de *advertising* del tipo ADV_IND, junto con el intercambio SCAN_REQ/SCAN_RSP. Posteriormente, el central transmite el paquete CONNECT_IND, que inicia la conexión BLE. En el detalle del paquete #14 se muestran parámetros de enlace relevantes, como el intervalo de conexión, latencia, timeout de supervisión.

6.2.1. Validación del advertising

La Figura 6.8 muestra los paquetes de *advertising* emitidos por el dispositivo antes de establecer la conexión. En la captura se observan paquetes del tipo ADV_IND, lo cual indica que el dispositivo se encuentra realizando *advertising* conectable e indirecto. Este comportamiento es consistente con la configuración utilizada en el *firmware*, donde se define el *advertising* mediante la opción BT_LE_ADV_OPT_CONNECTABLE. Además, el intervalo configurado es de aproximadamente 5 s, en concordancia con el valor establecido en el *firmware*.

En el contenido del paquete de *advertising* también se observa el nombre del dispositivo, NEUROSOC. Esto permite verificar que la información configurada en los datos de *advertising* es efectivamente transmitida por el periférico y puede ser detectada por el dispositivo central.

6.2.2. Proceso de establecimiento de conexión

Tal como se mencionó anteriormente, inicialmente NeuroSoC transmite paquetes ADV_IND en los canales de *advertising*. Luego, el central, en este caso un celular, envía un paquete SCAN_REQ, mediante el cual solicita información adicional al periférico. La PCB responde con un paquete SCAN_RSP. Este intercambio es el comportamiento esperado cuando un dispositivo central detecta un periférico BLE y solicita datos complementarios antes de iniciar una conexión.

Posteriormente, se observa un paquete CONNECT_IND, transmitido por el central. Este paquete marca el inicio formal de la conexión BLE. En él se definen parámetros que serán utilizados por ambos dispositivos para comunicarse en los canales de datos, tales como el intervalo de conexión, la latencia del esclavo y el *supervision timeout*. En este caso, estos parámetros resultan efectivos únicamente al inicio del proceso de conexión, ya que más adelante pueden ser renegociados por un nuevo conjunto de parámetros seleccionado específicamente para alcanzar un diseño de menor consumo energético.

En la captura analizada en la Figura 6.8 se observa un intervalo inicial de conexión de 45 ms, calculado a partir del valor transmitido en el paquete de conexión.

6.2.3. Indicadores iniciales de conexión

Luego del paquete CONNECT_IND, la comunicación deja de realizarse en los canales de *advertising* y pasa a los canales de datos. La Figura 6.9 muestra los primeros paquetes posteriores al establecimiento de la conexión. En esta etapa se observa tráfico de control de la capa *Link Layer*, incluyendo el intercambio de características soportadas y versiones del controlador.

Estos paquetes permiten verificar que, luego de iniciada la conexión, ambos dispositivos comienzan a negociar capacidades propias de la comunicación BLE. Esta etapa resulta relevante porque confirma que el vínculo ya no se limita al *advertising*, sino que pasa a operar como una conexión BLE establecida sobre los canales de datos.

6.2.4. Establecimiento de conexión y descubrimiento de servicios GATT

A continuación se presentan paquetes donde se puede ver la solicitud por parte del central para el descubrimiento de servicios y la respuesta por parte del periférico brindando la información requerida. Como se observa en la Figura 6.10, aparecen paquetes ATT del tipo *Read By Group Type Request* y *Read By Group Type Response* (#59 y #64, respectivamente). Estos paquetes corresponden al mecanismo utilizado por el cliente GATT para consultar los servicios primarios expuestos por el servidor GATT.

En este caso, la respuesta por parte del periférico brinda información de un servicio con UUID 0x438c7dee-77ec-4a5b-8741-24c721eeb043, correspondiente al servicio NSS implementado destinado al intercambio de parámetros de estimulación y comandos de control de terapia entre la aplicación y el dispositivo implantable, mencionado en la Sección 5.3.7. En la captura también se identifican servicios estándar BLE, como *Generic Access*, *Generic Attribute* y *Device Information*.

En el *firmware*, la inicialización del servicio BLE se realiza mediante la función `nss_init()`, a la cual se le entrega una estructura de *callbacks* de aplicación. En dicha

6.2. Validación de la comunicación BLE

12	15.017	f1:9c:9f:03:48:cd	LE 1M	LE LL	19	50044...	ADV_IND
13	15.018	f1:9c:9f:03:48:cd	LE 1M	LE LL	19	296µs	ADV_IND
14	15.018	f1:9c:9f:03:48:cd	LE 1M	LE LL	19	296µs	ADV_IND
15	15.019	4e:4e:c0:cd:16:00	LE 1M	LE LL	34	150µs	CONNECT_IND
16	15.022	Central_0x455659b6	LE 1M	LE LL	9	2797µs	0 0 False Control Opcode: LL_FEATURE_REQ
17	15.022	Peripheral_0x455659b6	LE 1M	LE LL	0	150µs	0 1 False Empty PDU
18	15.067	Central_0x455659b6	LE 1M	LE LL	0	44618...	1 1 False Empty PDU
19	15.067	Peripheral_0x455659b6	LE 1M	LE LL	9	150µs	1 0 False Control Opcode: LL_FEATURE_RSP
20	15.112	Central_0x455659b6	LE 1M	LE LL	6	44618...	0 0 False Control Opcode: LL_VERSION_IND
21	15.112	Peripheral_0x455659b6	LE 1M	LE LL	0	150µs	0 1 False Empty PDU
22	15.157	Central_0x455659b6	LE 1M	LE LL	0	44643...	1 1 False Empty PDU
23	15.157	Peripheral_0x455659b6	LE 1M	LE LL	6	150µs	1 0 False Control Opcode: LL_VERSION_IND
24	15.202	Central_0x455659b6	LE 1M	SMP	11	44642...	0 0 False Sent Pairing Request: AuthReq: Bonding, MITM,
25	15.202	Peripheral_0x455659b6	LE 1M	LE LL	0	150µs	0 1 False Empty PDU
26	15.247	Central_0x455659b6	LE 1M	LE LL	0	44602...	1 1 False Empty PDU
27	15.247	Peripheral_0x455659b6	LE 1M	LE LL	9	151µs	1 0 True Control Opcode: LL_PERIPHERAL_FEATURE_REQ
28	15.247	Central_0x455659b6	LE 1M	LE LL	0	150µs	0 0 False Empty PDU
29	15.292	Central_0x455659b6	LE 1M	LE LL	0	44388...	0 0 True Empty PDU
30	15.292	Peripheral_0x455659b6	LE 1M	SMP	11	150µs	0 1 False Rcvd Pairing Response: AuthReq: Bonding, Secur
31	15.292	Central_0x455659b6	LE 1M	LE LL	9	150µs	1 1 False Control Opcode: LL_FEATURE_RSP
32	15.337	Central_0x455659b6	LE 1M	LE LL	9	44300...	1 1 True Control Opcode: LL_FEATURE_RSP


```

> Frame 15: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface COM16-4.4, id 0
> nRF Sniffer for Bluetooth LE
  v Bluetooth Low Energy Link Layer
    Access Address: 0x8e89bed6
    > Packet Header: 0x22e5 (PDU Type: CONNECT_IND, ChSel: #2, TxAdd: Random, RxAdd: Random)
    Initiator Address: 4e:4e:c0:cd:16:00 (4e:4e:c0:cd:16:00)
    Advertising Address: f1:9c:9f:03:48:cd (f1:9c:9f:03:48:cd)
    v Link Layer Data
      Access Address: 0x455659b6
      CRC Init: 0xb3d841
      Window Size: 2 (2,5 msec)
      Window Offset: 0 (0 msec)
      Interval: 36 (45 msec)
      Latency: 0
      Timeout: 500 (5000 msec)
    > Channel Map: ffff0f801f
      ...0 0111 = Hop: 7
      101. .... = Sleep Clock Accuracy: 31 ppm to 50 ppm (5)
      CRC: 0x088c1e
  
```

Figura 6.9: Primeros paquetes posteriores al establecimiento de la conexión. Se observan paquetes de control de Link Layer asociados al intercambio de capacidades y parámetros iniciales.

estructura se definen funciones asociadas a la lectura y escritura de datos de estimulación y escritura del control de terapia. Por tanto, el descubrimiento GATT observado en *Wireshark* permite validar que el servicio implementado en el *firmware* es expuesto correctamente al central BLE.

6.2.5. Validación del emparejamiento y seguridad

La comunicación entre la aplicación móvil y la PCB requiere el establecimiento de una conexión segura, tal como se menciona en la Sección 5.3.6. En el *firmware* se registran *callbacks* de autenticación y *callbacks* de información de autenticación, incluyendo la solicitud de datos OOB, la aceptación del emparejamiento y la notificación de emparejamiento completo o fallido. Este diseño se utiliza para implementar un mecanismo de autenticación basado en NFC como canal fuera de banda (*Out of Band*, OOB).

En la captura de *Wireshark*, esta etapa se observa como tráfico SMP sobre el canal L2CAP destinado al emparejamiento. En particular, se identifican paquetes *Pairing Request*, *Pairing Response*, intercambio de claves públicas, paquetes *Pairing Random* y *DHKey Check*. Estos paquetes son consistentes con un procedimiento de emparejamiento LE Secure Connections. Además, en el *Pairing Request* se observa el indicador OOB activado, lo cual evidencia que el central declara disponer de datos de autenticación

Capítulo 6. Resultados

59	15.652	Central_0x455659b6	LE 1M	ATT	11 42567...	0	0	True	14 Sent Read By Group Type Request, Primary Service, Handles: 0x0017..0xffff
60	15.652	Peripheral_0x455659b6	LE 1M	LE LL	0 150µs	0	1	False	14 Empty PDU
61	15.652	Central_0x455659b6	LE 1M	SMP	21 150µs	1	1	False	14 Sent Pairing Random
62	15.653	Peripheral_0x455659b6	LE 1M	LE LL	0 150µs	1	0	False	14 Empty PDU
63	15.697	Central_0x455659b6	LE 1M	LE LL	0 43974...	0	0	False	15 Empty PDU
64	15.697	Peripheral_0x455659b6	LE 1M	ATT	26 150µs	0	1	True	15 Rcvd Read By Group Type Response, Attribute List Length: 1, Unknown
65	15.697	Central_0x455659b6	LE 1M	LE LL	0 150µs	1	1	False	15 Empty PDU

Bluetooth Low Energy Link Layer

Access Address: 0x455659b6
[Central Address: 4e:4e:c0:cd:16:00 (4e:4e:c0:cd:16:00)]
[Peripheral Address: f1:9c:9f:03:48:cd (f1:9c:9f:03:48:cd)]

Data Header

[L2CAP Index: 8]
[Connection Parameters in: 15]
CRC: 0xf17464

Bluetooth L2CAP Protocol

Length: 22
CID: Attribute Protocol (0x0004)

Bluetooth Attribute Protocol

Opcode: Read By Group Type Response (0x11)
0... = Authentication Signature: False
.0... = Command: False
..01 0001 = Method: Read By Group Type Response (0x11)
Length: 20
Attribute Data, Handle: 0x0017, Group End Handle: 0x001a, UUID128: Unknown
Handle: 0x0017 (Unknown)
[UUID: 438c7dee77ec4a5b874124c721eeb043]
Group End Handle: 0x001a
UUID: 43b0ee21c72441875b4aec77ee7d8c43
[UUID: Primary Service (0x2800)]
[Request in Frame: 59]

Figura 6.10: Proceso de descubrimiento GATT, donde el central solicita los servicios primarios mediante Read By Group Type Request y el periférico responde con Read By Group Type Response, identificándose el servicio NSS implementado mediante su UUID de 128 bits.

26	15.257	Central_0x44de1ebb	LE 1M	SMP	11 44642...	0	0	False	Sent Pairing Request: AuthReq: Bonding, MITM, SecureConnection
27	15.258	Peripheral_0x44de1ebb	LE 1M	LE LL	0 150µs	0	1	False	Empty PDU
28	15.302	Central_0x44de1ebb	LE 1M	LE LL	0 44603...	1	1	False	Empty PDU

> Frame 26: 37 bytes on wire (296 bits), 37 bytes captured (296 bits) on interface COM16-4.4, id 0

> nRF Sniffer for Bluetooth LE

Bluetooth Low Energy Link Layer

Access Address: 0x44de1ebb
[Central Address: 47:e6:be:89:15:fe (47:e6:be:89:15:fe)]
[Peripheral Address: f1:9c:9f:03:48:cd (f1:9c:9f:03:48:cd)]

Data Header

[L2CAP Index: 0]
[Connection Parameters in: 14]
CRC: 0x00df8a

Bluetooth L2CAP Protocol

Length: 7
CID: Security Manager Protocol (0x0006)

Bluetooth Security Manager Protocol

Opcode: Pairing Request (0x01)
IO Capability: Keyboard, Display (0x04)
OOB Data Flags: OOB Auth. Data From Remote Device Present (0x01)
AuthReq: 0x2d, CT2 Flag, Secure Connection Flag, MITM Flag, Bonding Flags: Bonding
Max Encryption Key Size: 16
Initiator Key Distribution: 0x0f, Link Key, Signature Key (CSRK), Id Key (IRK), Encryption Key (LTK)
Responder Key Distribution: 0x0f, Link Key, Signature Key (CSRK), Id Key (IRK), Encryption Key (LTK)

Figura 6.11: Captura del paquete de terminación de conexión y su contenido.

obtenidos por un canal externo a BLE, en este caso NFC.

Esta observación resulta especialmente importante para la validación del sistema, ya que permite relacionar el mecanismo de autenticación diseñado con evidencia directa en el tráfico capturado. Si bien *Wireshark* no indica explícitamente el nivel de seguridad interno alcanzado por el stack de BLE, sí permite observar las etapas del procedimiento de emparejamiento seguro y la posterior activación del cifrado de enlace.

6.2.6. Activación del cifrado

La Figura 6.13 muestra la etapa final del establecimiento seguro de la conexión. Luego del intercambio SMP, se observan paquetes de control de *Link Layer* asociados

26	15.257	Central_0x44de1ebb	LE 1M	SMP	11 44642...	0	0	False	Sent Pairing Request: AuthReq: Bonding, MITM, SecureConnection
27	15.258	Peripheral_0x44de1ebb	LE 1M	LE LL	0 150µs	0	1	False	Empty PDU
28	15.302	Central_0x44de1ebb	LE 1M	LE LL	0 44603...	1	1	False	Empty PDU


```

> Frame 26: 37 bytes on wire (296 bits), 37 bytes captured (296 bits) on interface COM16-4.4, id 0
> nRF Sniffer for Bluetooth LE
v Bluetooth Low Energy Link Layer
  Access Address: 0x44de1ebb
  [Central Address: 47:e6:be:89:15:fe (47:e6:be:89:15:fe)]
  [Peripheral Address: f1:9c:9f:03:48:cd (f1:9c:9f:03:48:cd)]
> Data Header
  [L2CAP Index: 0]
  [Connection Parameters in: 14]
  CRC: 0x00df8a
v Bluetooth L2CAP Protocol
  Length: 7
  CID: Security Manager Protocol (0x0006)
v Bluetooth Security Manager Protocol
  Opcode: Pairing Request (0x01)
  IO Capability: Keyboard, Display (0x04)
  OOB Data Flags: OOB Auth. Data From Remote Device Present (0x01)
  AuthReq: 0x2d, CT2 Flag, Secure Connection Flag, MITM Flag, Bonding Flags: Bonding
  Max Encryption Key Size: 16
  Initiator Key Distribution: 0x0f, Link Key, Signature Key (CSRK), Id Key (IRK), Encryption Key (LTK)
  Responder Key Distribution: 0x0f, Link Key, Signature Key (CSRK), Id Key (IRK), Encryption Key (LTK)

```

Figura 6.12: Paquete *Pairing Request* durante el procedimiento de emparejamiento BLE. Se observa el indicador OOB activado, asociado al uso de NFC como canal fuera de banda.

a la activación del cifrado, tales como `LL_ENC_REQ`, `LL_ENC_RSP` y `LL_START_ENC_REQ`. A partir de este punto, los paquetes de datos posteriores dejan de ser interpretables directamente por *Wireshark* sin disponer de las claves de sesión.

Este comportamiento es esperado y constituye una evidencia de que la comunicación posterior, incluyendo el envío de parámetros de estimulación y comandos de encendido o apagado de terapia, se realiza sobre un enlace cifrado. En la práctica, durante el ensayo se verificó que luego de enviar los parámetros desde la aplicación móvil, la terapia se iniciaba correctamente en la PCB, lo cual fue confirmado mediante medición con osciloscopio. Sin embargo, el contenido de dichos parámetros no aparece en claro en *Wireshark* debido a que ya se había activado el cifrado de la conexión.

6.2.7. Rechazo de conexión no autenticada

Por último, se evaluó el caso en que un dispositivo intenta establecer una conexión con el neuroestimulador sin haber realizado previamente el intercambio de claves OOB mediante NFC.

La Figura 6.14 muestra la captura correspondiente a este ensayo. En ella se observa que, si bien el dispositivo central logra iniciar el procedimiento de conexión, este no puede completarse exitosamente y finaliza con un paquete `LL_TERMINATE_IND`, en cuyo contenido se indica el motivo de terminación mediante el código `0x13`.

Este resultado muestra que el mecanismo de autenticación mediante NFC cumple efectivamente su función, ya que impide que dispositivos que no hayan realizado previamente el intercambio de claves OOB accedan a la configuración del neuroestimulador.

6.2.8. Conclusión de la validación

En consecuencia, el análisis de las capturas BLE permitió verificar que la comunicación inalámbrica implementada en el neuroestimulador sigue el comportamiento esperado

Capítulo 6. Resultados

→	73	15.794	Central_0x455659b6	LE 1M	LE LL	23 6491µs	0	0	False Control Opcode: LL_ENC_REQ
	74	15.795	Peripheral_0x455659b6	LE 1M	LE LL	0 149µs	0	1	False Empty PDU
	75	15.802	Central_0x455659b6	LE 1M	LE LL	0 7007µs	1	1	False Empty PDU
	76	15.802	Peripheral_0x455659b6	LE 1M	LE LL	13 150µs	1	0	False Control Opcode: LL_ENC_RSP
	77	15.809	Central_0x455659b6	LE 1M	LE LL	0 7086µs	0	0	False Empty PDU
	78	15.810	Peripheral_0x455659b6	LE 1M	LE LL	0 150µs	0	1	False Empty PDU
	79	15.817	Central_0x455659b6	LE 1M	LE LL	0 7190µs	1	1	False Empty PDU
←	80	15.817	Peripheral_0x455659b6	LE 1M	LE LL	1 150µs	1	0	False Control Opcode: LL_START_ENC_REQ
	81	15.824	Central_0x455659b6	LE 1M	LE LL	5 7182µs	0	0	False Encrypted packet decrypted incorrectly (bad MIC)
	82	15.825	Peripheral_0x455659b6	LE 1M	LE LL	0 150µs	0	1	False Empty PDU
	83	15.832	Central_0x455659b6	LE 1M	LE LL	0 7150µs	1	1	False Empty PDU
	84	15.832	Peripheral_0x455659b6	LE 1M	LE LL	5 150µs	1	0	False Encrypted packet decrypted incorrectly (bad MIC)
	85	15.839	Central_0x455659b6	LE 1M	LE LL	15 7150µs	0	0	True Encrypted packet decrypted incorrectly (bad MIC)
	86	15.840	Peripheral_0x455659b6	LE 1M	LE LL	0 150µs	0	1	False Empty PDU

> Frame 80: 27 bytes on wire (216 bits), 27 bytes captured (216 bits) on interface COM16-4.4, id 0

> nRF Sniffer for Bluetooth LE

> Bluetooth Low Energy Link Layer

Access Address: 0x455659b6

[Central Address: 4e:4e:c0:cd:16:00 (4e:4e:c0:cd:16:00)]

[Peripheral Address: f1:9c:9f:03:48:cd (f1:9c:9f:03:48:cd)]

> Data Header

....11 = LLID: Control PDU (0x3)

....0.. = Next Expected Sequence Number: 0 [ACK]

....1... = Sequence Number: 1 [OK]

...0.... = More Data: False

..0.... = CTE Info: Not Present

00.... = RFU: 0

Length: 1

Control Opcode: LL_START_ENC_REQ (0x05)

[Request in Frame: 73]

[Connection Parameters in: 38]

CRC: 0x8963d1

Figura 6.13: Activación del cifrado de la conexión BLE. Luego de los paquetes LL_ENC_REQ, LL_ENC_RSP y LL_START_ENC_REQ, los datos posteriores se transmiten cifrados.

468	140.159	Peripheral_0x70b59ede	LE 2M	LE LL	2 150µs	1	0	False Control Opcode: LL_TERMINATE_IND
469	140.909	Central_0x70b59ede	LE 2M	LE LL	0 749371µs	0	0	False Empty PDU

> Bluetooth Low Energy Link Layer

Access Address: 0x70b59ede

[Central Address: 74:ca:ea:d1:69:31 (74:ca:ea:d1:69:31)]

[Peripheral Address: e6:28:8c:d7:bf:64 (e6:28:8c:d7:bf:64)]

> Data Header

....11 = LLID: Control PDU (0x3)

....0.. = Next Expected Sequence Number: 0 [ACK]

....1... = Sequence Number: 1 [OK]

...0.... = More Data: False

..0.... = CTE Info: Not Present

00.... = RFU: 0

Length: 2

Control Opcode: LL_TERMINATE_IND (0x02)

Error Code: Remote User Terminated Connection (0x13)

[Connection Parameters in: 381]

CRC: 0x9e69f9

Figura 6.14: Captura del paquete de terminación de conexión y su contenido.

de acuerdo con la configuración del *firmware* y con los fundamentos teóricos presentados previamente. En particular, se validó la secuencia desde el *advertising* hasta la activación del cifrado para el posterior intercambio de datos.

6.3. Caracterización del PCB implementado

6.3.1. Variaciones en la caída de tensión en el tejido

A continuación se analiza el desempeño del sistema frente a variaciones en la caída de tensión en el tejido.

Como se estudió en la Sección 4.1.1, el circuito de estimulación puede modelarse

6.3. Caracterización del PCB implementado

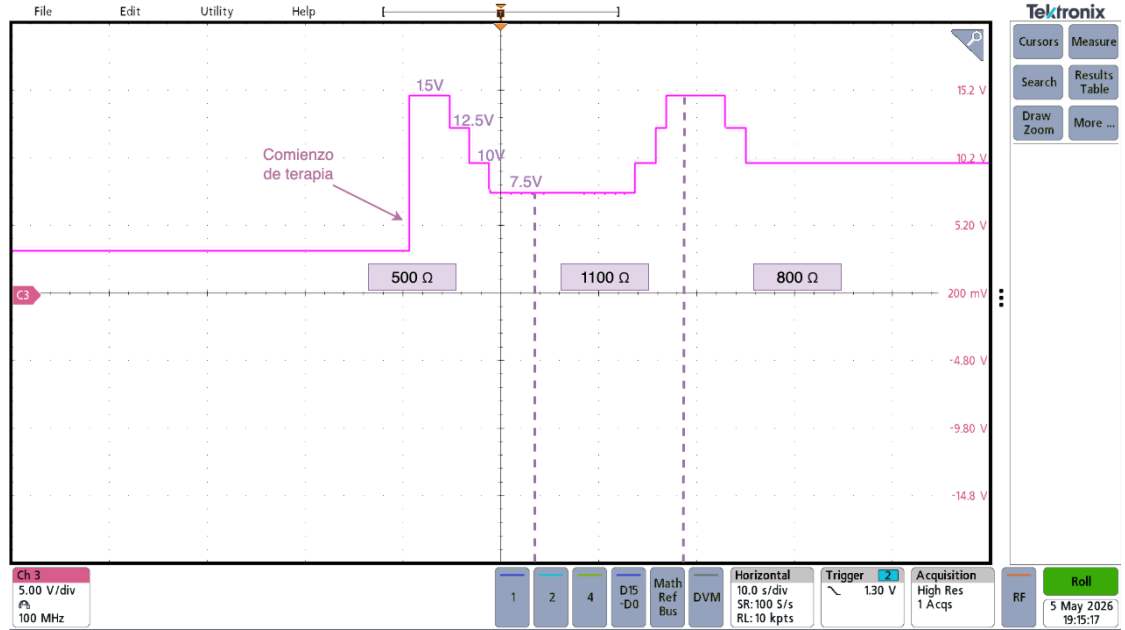
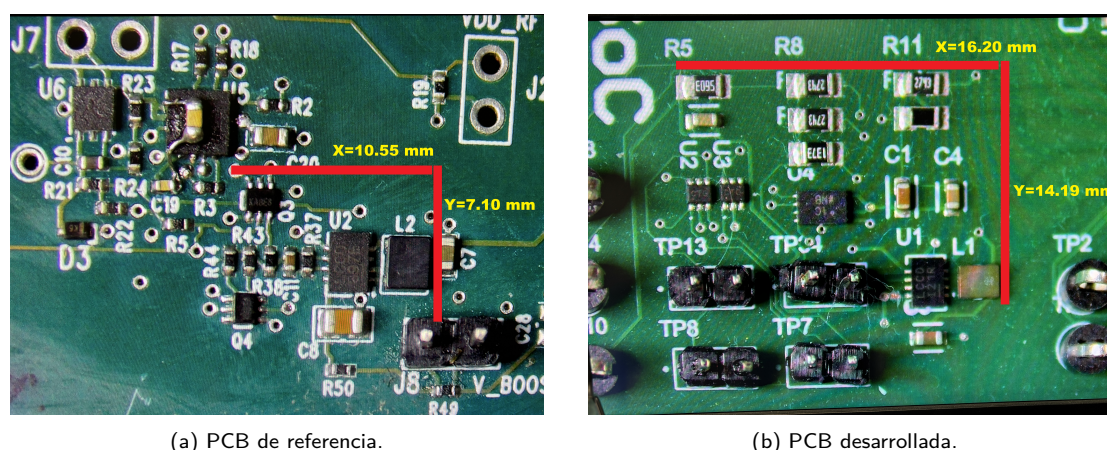


Figura 6.15: Tensión en V_{Boost} al estimular con una amplitud de corriente de 10 mA sobre una resistencia variable. Al comienzo de la terapia la tensión se fija en 15 V. Al realizar 3 medidas altas consecutivas de la tensión en V_{Sink} se ajusta la tensión V_{Boost} , hasta alcanzar el valor en el que se estabiliza. Luego, al aumentar la resistencia a 1100 Ω, la tensión de V_{Boost} se ajusta hasta alcanzar nuevamente el valor de 15 V. Al disminuir la resistencia a 800 Ω, la tensión vuelve a ajustarse al obtener 3 medidas altas consecutivas y se estabiliza en 10 V.

como se muestra en la Figura 4.2. Las variaciones en la caída de tensión sobre el tejido afectan la tensión de alimentación necesaria para que la fuente de corriente opere dentro de su región de regulación.

En la Figura 6.15 se presenta la evolución de la tensión V_{Boost} durante la estimulación, para una amplitud de corriente de 10 mA. Al inicio de la terapia, V_{Boost} se establece en su valor máximo (15 V) con el objetivo de garantizar la correcta generación de los primeros pulsos, independientemente de la amplitud de corriente y de la resistencia del tejido. Posteriormente, considerando una caída de tensión de 5 V en el tejido, es posible reducir V_{Boost} manteniendo el correcto funcionamiento, alcanzando un valor de 7.5 V. Luego, al aumentar la resistencia del tejido a 1100 Ω, la caída de tensión se incrementa a 11 V. En este caso, tras detectarse tres mediciones consecutivas de tensión baja en V_{Sink} , V_{Boost} comienza a incrementarse hasta alcanzar nuevamente su valor máximo de 15 V. Finalmente, al reducir la resistencia a un valor intermedio de 800 Ω, se obtiene una caída de tensión de 8 V en el tejido. Como consecuencia, al registrarse tres mediciones consecutivas de tensión alta en V_{Sink} , V_{Boost} puede disminuir hasta 10 V. Las variaciones de resistencia utilizadas durante el ensayo se realizaron mediante un equipo *Ohm Ranger*.



(a) PCB de referencia.

(b) PCB desarrollada.

Figura 6.16: Comparación del layout de la etapa de realimentación en ambas implementaciones. En la placa de test proporcionada, la distancia entre el nodo de salida y el pin FB es aproximadamente 0.5 cm , mientras que en la PCB desarrollada dicha distancia es de aproximadamente 1.8 cm .

6.3.2. Funcionamiento del boost converter

Si bien la configuración eléctrica implementada en la PCB desarrollada en este trabajo es, a nivel esquemático, equivalente a la utilizada en la *test board* provista por el diseño anterior del neuroestimulador, experimentalmente se observó que la señal V_{boost} presenta un nivel de ruido sensiblemente mayor.

La principal diferencia identificada entre ambas implementaciones se encuentra en el *layout* de la etapa de realimentación. En la PCB desarrollada, las resistencias que conectan el nodo de salida con el pin de realimentación FB se encuentran a una distancia aproximada de 2.5 cm del integrado, mientras que en la placa de referencia dicha distancia es del orden de 0.7 cm como se muestra en las Figuras 6.16a y 6.16b, respectivamente.

Esta diferencia resulta relevante, ya que el fabricante *boost* indica que la conexión de *feedback* debe realizarse directamente desde V_{OUT} hacia FB y mantenerse lo más corta posible, de forma de garantizar una señal limpia y libre de ruido. Asimismo, también recomienda minimizar la longitud y el área de las pistas asociadas al nodo de conmutación SW , debido a sus flancos rápidos y a su elevada capacidad de acoplar interferencia electromagnética hacia otras redes del circuito.

Desde el punto de vista físico, al aumentar la longitud de la interconexión del divisor resistivo de realimentación, la pista se vuelve más susceptible al acople capacitivo e inductivo proveniente de los nodos de alta variación temporal del convertidor, especialmente del nodo SW . Como consecuencia, la tensión sensada en FB deja de representar únicamente el valor medio de salida y pasa a contener una componente de alta frecuencia asociada al switching. Esto perturba el lazo de control e incrementa el rizado y el ruido observable en la salida del convertidor.

En el presente caso, este efecto se vuelve particularmente importante porque la aplicación requiere una tensión de alimentación de baja perturbación para la etapa de estimulación. Por lo tanto, aunque el valor medio de V_{boost} pueda mantenerse dentro de especificación, un aumento en el ruido de conmutación impacta negativamente sobre la calidad de la alimentación del bloque analógico y puede propagarse hasta los nodos

6.3. Caracterización del PCB implementado

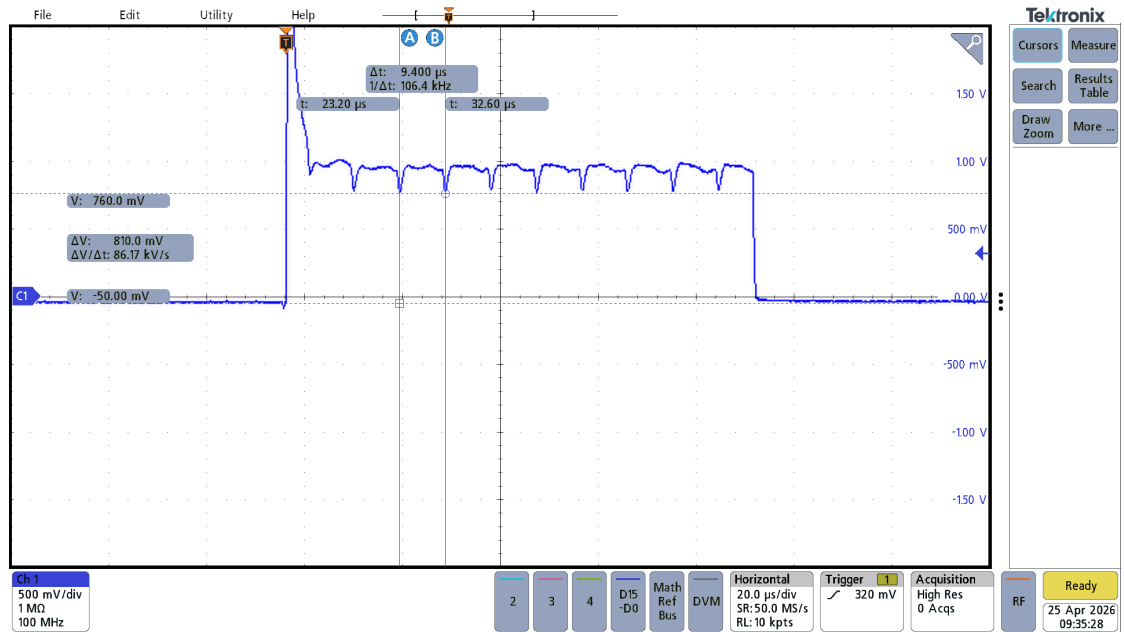


Figura 6.17: Frecuencia del ruido observado durante un pulso de estimulación al utilizar el *boost converter* de la PCB diseñada.

utilizados para excitar los electrodos.

Respecto a la frecuencia del ruido, la *datasheet* indica que la frecuencia de conmutación nunca cae por debajo de 50 kHz , y que aumenta con la corriente de carga. La curva “Switching Frequency vs Load Current” muestra que, para la configuración típica del fabricante, la frecuencia puede variar aproximadamente desde 50 kHz en carga muy baja hasta valores del orden de 1.3 MHz en corrientes elevadas. En la región intermedia, que suele ser la más representativa en operación normal, la frecuencia se ubica típicamente entre algunos cientos de kHz y alrededor de 700 kHz .

Por lo tanto, si en las capturas experimentales del nodo *CAP* se observa una componente dominante dentro del rango aproximado de 50 kHz a 1.3 MHz , y en particular en la zona de algunos cientos de kHz , resulta razonable atribuir dicho ruido al funcionamiento intrínseco del convertidor step-up. En ese caso, la mayor amplitud observada en la PCB desarrollada puede asociarse a un layout menos favorable del lazo de realimentación y del camino de conmutación, más que a una diferencia en el esquemático eléctrico.

En la Figura 6.17 se puede observar la frecuencia medida del ruido en el pulso alimentado del *boost converter* de la PCB diseñada. Además, en la Figura 6.18, se puede observar una comparación entre el ruido presente en el pulso de estimulación al utilizar cada uno de los *boost converters*.

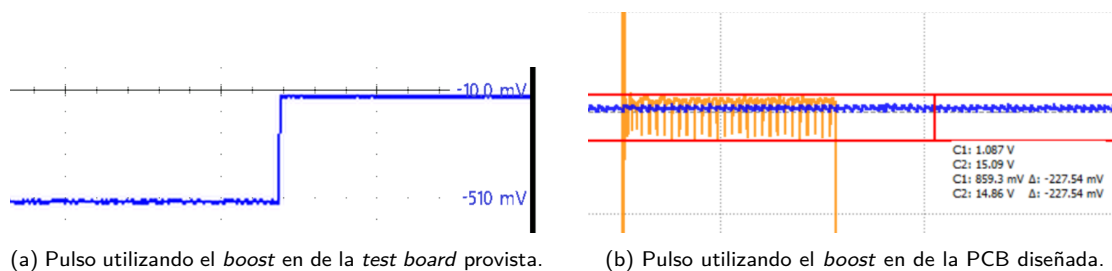


Figura 6.18: Comparación del ruido observado en el pulso de estimulación al utilizar el *boost* de la *test board* provista y la PCB diseñada, con ruidos de amplitud 227mV y 10mV respectivamente.

6.4. Perfil de Consumo

En esta Sección se analiza el consumo energético del sistema desarrollado, considerando distintos modos de operación y bloques funcionales. El objetivo es obtener una caracterización inicial del consumo, identificar los principales contribuyentes y evaluar el desempeño en relación con los valores esperados según la documentación del fabricante.

La metodología adoptada consistió en realizar mediciones progresivas, comenzando por escenarios controlados y simples, para luego incorporar gradualmente funcionalidades del sistema. Esto permitió aislar el impacto de cada subsistema en el consumo total y orientar posibles optimizaciones.

Finalmente, se presentan mediciones representativas para distintos escenarios de uso, incluyendo la operación en diferentes configuraciones de estimulación.

6.4.1. Configuración para las pruebas

Las mediciones de consumo se realizaron utilizando una fuente externa Otii Arc, empleada para alimentar el sistema y registrar el perfil temporal de corriente durante su funcionamiento. Este instrumento presenta una alta precisión de medición, del orden de $\pm(0.1\% + 50\text{ nA})$ para corrientes de hasta 19 mA , lo que permite una caracterización confiable del consumo del sistema [25]. Para la caracterización se evaluaron distintos escenarios de operación, considerando tanto estados de comunicación como de estimulación, y utilizando parámetros representativos de terapia.

Además de medir el consumo total de la placa, se realizaron mediciones específicas sobre la alimentación del SoC, desacoplándolo del bloque de estimulación como se muestra en la Figura 6.19. Esto permitió estimar la contribución relativa del microcontrolador y del circuito de estimulación en cada estado operativo, así como comparar los resultados con sistemas de naturaleza similar.

Finalmente, se evaluó el consumo en función de la frecuencia de estimulación, variando dicho parámetro para analizar si el aumento de la frecuencia introducía cambios apreciables en la corriente promedio del sistema.

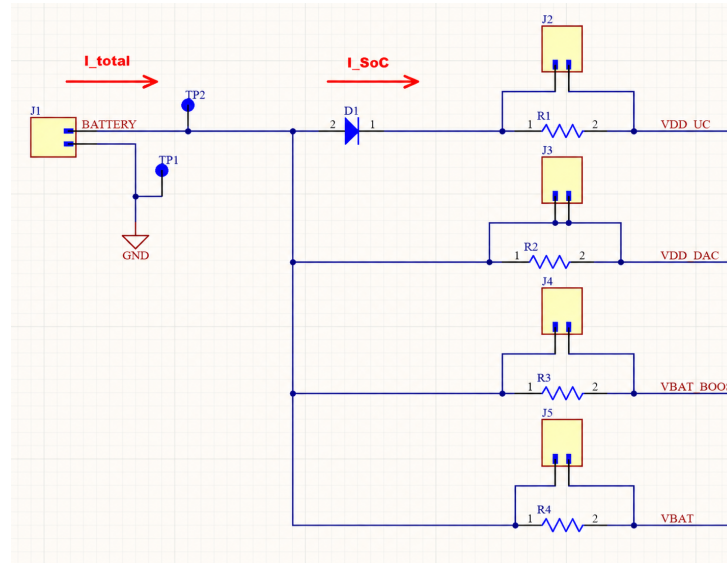


Figura 6.19: Esquema de alimentación del PCB utilizado. Las mediciones se realizaron tanto sobre el nodo *BATTERY*, correspondiente al consumo total de la placa, como sobre el nodo *VDD_UC*, asociado a la alimentación del SoC. Para esta última medición se retiró la resistencia R1, permitiendo aislar el consumo del microcontrolador respecto del resto del circuito.

6.4.2. Consumo de comunicación BLE y estimulación

Se analizaron los escenarios en los cuales el sistema establece comunicación mediante BLE, incluyendo *advertising*, autenticación por NFC, inicio de conexión y conexión establecida. La Figura 6.20 muestra la corriente consumida durante estas etapas, permitiendo identificar el impacto de cada una sobre el consumo del sistema.

En el estado de *advertising*, el neuroestimulador permanece desconectado y transmite periódicamente paquetes para anunciar su presencia. Esto genera picos breves de corriente asociados a la activación de la radio, manteniendo un consumo promedio reducido.

Luego, durante la interacción NFC, se observa un aumento temporal del consumo debido al intercambio de información OOB utilizado para la autenticación. Finalizado este proceso, el sistema retorna al estado de *advertising*.

El mayor incremento de consumo ocurre durante el inicio de la conexión BLE, donde se concentran los paquetes de control, el emparejamiento, la autenticación y la negociación inicial del enlace. Una vez establecida la conexión, el consumo disminuye y adopta un patrón más estable, determinado principalmente por los eventos periódicos del enlace BLE.

En conjunto, la medición permite distinguir claramente las etapas de comunicación inalámbrica y muestra que el consumo más elevado se produce durante el establecimiento de la conexión, mientras que *advertising* y conexión establecida presentan un régimen de menor actividad promedio.

A partir de las mediciones realizadas en la entrada del sistema completo y en la alimentación del SoC, se estimó la contribución del circuito de estimulación como la

Capítulo 6. Resultados

Tabla 6.7: Comparación del consumo entre el SoC, el circuito de estimulación y el total del sistema para una terapia PNS caracterizada por una frecuencia de 40 Hz, un ancho de pulso de 300 μs y una amplitud de 1.0 mA. La corriente I_{total} corresponde al consumo definido en la Figura 6.19, mientras que I_{SoC} representa el consumo del SoC según la misma figura. El consumo del circuito de estimulación se obtiene como la diferencia $I_{total} - I_{SoC}$, correspondiente al resto del sistema.

Estado	I_{total}	I_{SoC}	$I_{total} - SoC$	SoC	Estim
NFC	1.39 mA	1.29 mA	0.10 mA	92.8 %	7.2 %
Advertising	360 μA	284 μA	76 μA	78.9 %	21.1 %
Conexión establecida	364 μA	288 μA	76 μA	79.1 %	20.9 %
Terapia activa	7.21 mA	2.25 mA	4.96 mA	31.2 %	68.8 %

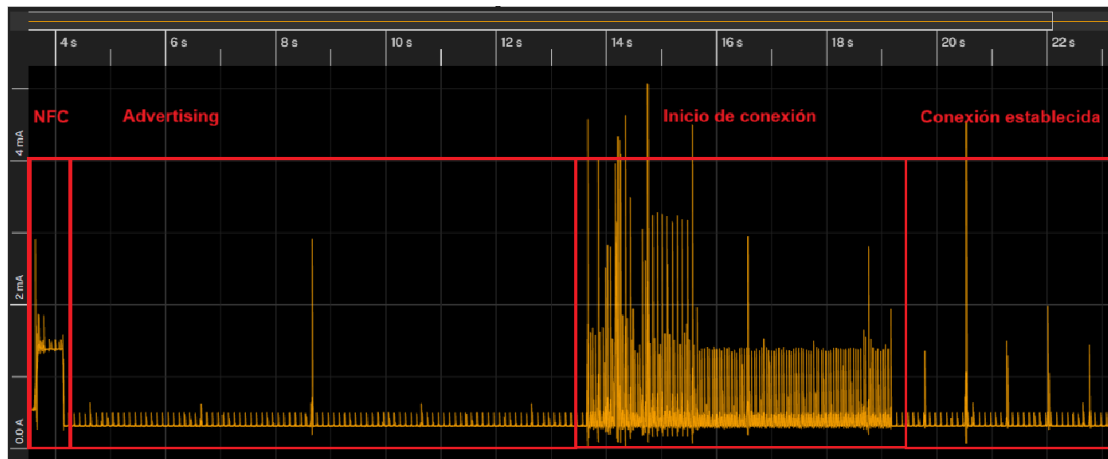


Figura 6.20: Captura del ciclo de consumo del sistema obtenida mediante Otii Arc. Se distinguen las etapas de *advertising*, activación del módulo NFC, inicio de conexión BLE y conexión establecida.

diferencia entre ambos valores. Como se observa en la Tabla 6.7, en los estados asociados principalmente a comunicación inalámbrica, como *advertising*, NFC y conexión establecida, el consumo se encuentra dominado por el SoC. En estas condiciones, el circuito de estimulación permanece mayormente inactivo y aporta una fracción reducida del consumo total. Esto se estimó considerando que el consumo obtenido como la resta de $I_{total} - I_{SoC}$ corresponde al consumo del circuito de estimulación.

En cambio, durante la terapia activa se observa un cambio significativo en la distribución del consumo. Esta diferencia está directamente asociada a la corriente de estimulación configurada, la cual puede variar entre 0.5 mA y 15 mA. En consecuencia, el consumo del circuito de estimulación puede representar hasta aproximadamente un 90 % del consumo total en el caso de máxima corriente, y descender hasta valores cercanos al 50 % para la corriente mínima.

6.4.3. Duración estimada del ciclo de batería

La autonomía del prototipo se estimó a partir de los consumos medidos experimentalmente en los distintos estados de operación del sistema. Para ello, se tomó como

referencia una batería recargable de ion-litio de 180 mAh, valor compatible con capacidades reportadas para generadores de pulsos implantables recargables comerciales [26]. Sin embargo, para obtener una estimación más conservadora, se consideró una capacidad útil del 90 % de la capacidad nominal, resultando en una capacidad efectiva de 162 mAh. Esta capacidad se utilizó únicamente como referencia para evaluar el tiempo de operación entre ciclos de carga, no como una especificación final del dispositivo.

La estimación considera dos estados principales de funcionamiento: un estado de reposo, asociado al sistema sin entrega activa de terapia, con un consumo de 364 μ A; y un estado de terapia activa, con un consumo de 7.21 mA. A partir de estos valores, se calculó la corriente promedio para distintos perfiles de uso mediante el duty-cycle de estimulación, según:

$$I_{promedio} = I_{reposo}(1 - D) + I_{terapia}D \quad (6.1)$$

donde D representa la fracción de tiempo en la cual el sistema se encuentra entregando terapia. La autonomía se calculó luego como:

$$T = \frac{C_{efectiva}}{I_{promedio}} \quad (6.2)$$

En particular, se evaluaron perfiles de estimulación intermitente compatibles con terapias PNS de bajo duty-cycle, considerando sesiones de 30 minutos aplicadas cada 3 o 4 días. Bajo estas condiciones, la autonomía estimada del sistema se encuentra en el entorno de 18 días por ciclo de carga.

Estos resultados indican que, para perfiles de estimulación intermitente de baja corriente, el consumo dominante no corresponde necesariamente a la etapa de estimulación, sino al consumo en reposo acumulado durante los períodos prolongados sin entrega de terapia. Por lo tanto, la reducción del consumo basal del SoC y de los periféricos asociados resulta un aspecto crítico para extender la autonomía del sistema.

En la Tabla 6.9 se analiza la variación del consumo del SoC al modificar la frecuencia de estimulación. Dado que un aumento de frecuencia implica una mayor cantidad de eventos de procesamiento por unidad de tiempo, podría esperarse un incremento en la corriente consumida. Sin embargo, en el rango evaluado no se observan variaciones apreciables. Esto sugiere que el consumo del SoC se encuentra dominado principalmente por componentes de consumo estático, asociadas a periféricos activos y al funcionamiento general del sistema, más que por el costo adicional introducido por el aumento de frecuencias y no tanto por el propio consumo de procesamiento.

Es importante destacar que el dispositivo fue concebido como una plataforma genérica de estimulación, capaz de entregar corrientes de hasta 15 mA. Esto exige dimensionar la etapa de salida y la alimentación para el peor caso de operación, con tensiones y márgenes superiores a los necesarios para terapias de baja corriente. Por lo tanto, en perfiles de menor amplitud, el consumo de la electrónica de estimulación es mayor que el de una arquitectura optimizada específicamente para esa terapia.

Al comparar estos resultados con dispositivos comerciales implantables, se observa que la autonomía estimada del prototipo es inferior a la de sistemas optimizados para una terapia determinada. Si bien el prototipo valida funcionalmente la operación del sistema y permite implementar distintos perfiles de estimulación, los resultados muestran que el

Capítulo 6. Resultados

consumo asociado al bloque de control continúa siendo uno de los principales aspectos a optimizar en futuras versiones.

Tabla 6.8: Estimación de autonomía para distintos perfiles de uso, considerando una batería de referencia de 180 mAh y una capacidad útil del 90 %.

Perfil de uso	Duty-cycle	I_{promedio}	Autonomía
Sin terapia	0 %	0.364 mA	18.5 días
PNS: 30 min cada 3 días	0.694 %	0.411 mA	16.4 días
PNS: 30 min cada 4 días	0.521 %	0.400 mA	16.9 días
PNS: 30 min/día	2.08 %	0.506 mA	13.3 días
PNS: 1 h/día	4.17 %	0.650 mA	10.4 días

Un resultado relevante es que la autonomía máxima queda fuertemente limitada por el consumo en reposo. Con una capacidad útil de 162 mAh y un consumo basal de 364 μA , el límite superior de autonomía, aun sin entrega de terapia, es de aproximadamente 19 días. Por lo tanto, en perfiles de bajo duty-cycle, la mejora de eficiencia de la etapa de estimulación tendría un impacto limitado si no se reduce en paralelo el consumo del bloque de control.

Tabla 6.9: Variación del consumo del SoC en función de la frecuencia de estimulación.

Frecuencia de estimulación	I_{SoC}
40 Hz	2.25 mA
100 Hz	2.25 mA
250 Hz	2.25 mA
600 Hz	2.25 mA
1500 Hz	2.25 mA

Capítulo 7

Conclusiones

En este capítulo se presenta una evaluación global del trabajo realizado. Se comparan los resultados obtenidos con los objetivos definidos en la Sección 1.2, analizando el nivel de cumplimiento alcanzado. Por último, se plantean posibles mejoras al sistema desarrollado, así como líneas de trabajo futuro identificadas durante el desarrollo del proyecto.

7.1. Conclusiones

En este trabajo se presentó la implementación de un sistema de neuroestimulación que reutiliza los circuitos de estimulación provistos por el cliente y reemplaza los dos microcontroladores originales por el SoC nRF5340. La solución desarrollada abarcó tanto el diseño del *hardware* como el desarrollo del *firmware*. En particular, se diseñó una nueva placa de circuito impreso a partir de los esquemáticos suministrados, incorporando las modificaciones necesarias para la integración del nuevo SoC y de los periféricos requeridos.

Desde el punto de vista del *hardware*, se logró sustituir exitosamente los microcontroladores originales por el SoC *dual-core* nRF5340 de Nordic Semiconductor. Para ello, fue necesario incorporar un conversor digital-analógico externo, dado que el microcontrolador original disponía de este periférico, así como una antena NFC para habilitar un mecanismo de emparejamiento más seguro. Asimismo, se simplificó el diseño original eliminando aquellos bloques que no resultaban necesarios para la prueba de concepto, tales como una de las fuentes de corriente, parte de la matriz de electrodos y el circuito de carga de la batería.

En una primera etapa, se validó y depuró el *firmware* mediante una plataforma de pruebas compuesta por la *test board* suministrada por el cliente, la placa de desarrollo del nRF5340 y el DAC seleccionado. Esta etapa permitió verificar el correcto funcionamiento del sistema y avanzar en el diseño del PCB con un mayor grado de confianza.

Posteriormente, el PCB desarrollado demostró ser funcional y confirmó la compatibilidad del SoC con el circuito de estimulación original. No obstante, el diseño del *layout* introdujo mayores niveles de ruido en el *boost converter* respecto de la *test board*. Las mejores prestaciones en la calidad de los pulsos se obtuvieron al estimular directamente

Capítulo 7. Conclusiones

desde la batería utilizando el PCB diseñado, o bien empleando el circuito desarrollado en conjunto con el *boost converter* de la plataforma de pruebas original.

En relación con el desarrollo del *firmware*, la utilización de Zephyr RTOS permitió cumplir con los requerimientos temporales establecidos para el sistema original, a pesar del *jitter* introducido por la utilización del sistema operativo en tiempo real. La única excepción se presentó en los pulsos de menor duración, para los cuales la incorporación de la medición de la tensión (V_{sink}) impidió satisfacer dichos requerimientos. Asimismo, se comprobó que el SoC *dual-core* permite replicar la arquitectura funcional del sistema original, en la que un microcontrolador se dedicaba a la comunicación y otro a la estimulación. La ejecución concurrente de ambas funciones se realizó sin comprometer el desempeño del sistema.

En síntesis, se considera que los objetivos planteados al inicio del proyecto fueron alcanzados satisfactoriamente. Los resultados obtenidos aportan valor académico y tecnológico al estudio del uso de sistemas operativos en tiempo real en aplicaciones médicas con requerimientos temporales estrictos, así como al empleo de SoC *dual-core* en dispositivos de neuroestimulación.

Finalmente, desde el punto de vista formativo, el proyecto involucró actividades de diseño de *hardware*, desarrollo de *firmware*, integración y validación experimental. Los desafíos asociados a cada una de estas áreas permitieron aplicar de manera integral los conocimientos adquiridos a lo largo de la carrera, constituyendo una experiencia formativa completa y multidisciplinaria. Asimismo, el trabajo brindó una valiosa aproximación al desarrollo de tecnologías orientadas al tratamiento de diversas enfermedades neurológicas.

7.2. Trabajo a futuro

La principal limitación identificada en esta implementación se relaciona con la adquisición de señales mediante el ADC durante la estimulación. En particular, la ejecución de estas mediciones introduce retardos adicionales que afectan el comportamiento temporal del sistema. Si bien no se determinó con certeza la causa exacta de esta latencia, se considera que una posible solución consistiría en emplear el mecanismo DPPI (*Distributed Programmable Peripheral Interconnect*), que permitiría una interacción más eficiente entre periféricos y reduciría la intervención del procesador. No obstante, debido a las restricciones de tiempo del proyecto y a la limitada disponibilidad de ejemplos de aplicación en configuraciones no sincronizadas mediante temporizadores, esta alternativa no pudo ser implementada en la presente versión.

En cuanto a los transitorios observados en los pulsos de estimulación, una posible línea de mejora consiste en optimizar la secuencia de conmutación de las señales que controlan las llaves del circuito. Sin embargo, dado el elevado número de elementos de conmutación, las capacitancias parásitas asociadas y la gran cantidad de combinaciones posibles de activación, la estrategia implementada representa una solución adecuada para los objetivos de esta prueba de concepto.

Respecto al diseño del PCB, futuras iteraciones podrían beneficiarse de una optimización del *placement* y del *routing*, especialmente en la etapa del *boost converter*, con el objetivo de reducir el nivel de ruido observado y mejorar la calidad de la señal generada.

En relación con el consumo energético, una línea de trabajo futuro de especial interés consiste en profundizar la optimización del *firmware* con el objetivo de maximizar la autonomía del dispositivo. Si bien en esta implementación se incorporaron mecanismos básicos de gestión de energía, futuras iteraciones podrían aprovechar de manera más exhaustiva las capacidades de bajo consumo del SoC. En particular, sería posible reducir aún más el consumo mediante el uso más intensivo del hilo de inactividad (*idle_thread*), así como a través del encendido y apagado selectivo de periféricos durante intervalos de inactividad del procesador durante la terapia, especialmente en aquellas de baja frecuencia o con anchos de pulso elevados. Estas estrategias deberían evaluarse experimentalmente, considerando las latencias asociadas al encendido y apagado de los periféricos, así como a la transición entre estados activos e inactivos del procesador.

Finalmente, como línea de trabajo futuro fuera del alcance de este proyecto, resultaría de gran interés el desarrollo de una interfaz gráfica de usuario que facilite la configuración del dispositivo, mejore la experiencia de uso y permita la visualización del estado del sistema, así como la recepción y gestión de mensajes de error.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice A

Fundamentos de Bluetooth Low Energy

El objetivo de este capítulo es introducir los fundamentos de Bluetooth Low Energy. Se presentan los conceptos del protocolo relevantes para la implementación, tales como el *BLE Protocol Stack*, los parámetros de conexión y los mecanismos de emparejamiento.

A.1. Introducción

El nRF5340 utiliza comunicación mediante BLE, un protocolo de comunicación inalámbrica diseñado para aplicaciones de bajo consumo energético [27]. En comparación con Bluetooth clásico, BLE reduce la tasa de transmisión efectiva de datos (*data rate*) con el objetivo de minimizar el consumo de energía. Esto se logra mediante el uso de paquetes de datos de menor tamaño y el incremento en los intervalos entre transmisiones, lo que permite mantener la radio apagada durante períodos prolongados.

Este protocolo es ampliamente soportado por la mayoría de los teléfonos inteligentes, lo que lo convierte en una solución práctica tanto para la implementación como para el desarrollo y validación de aplicaciones. Asimismo, presenta un costo reducido en comparación con otras tecnologías de comunicación inalámbrica de bajo consumo.

Algunas características relevantes de BLE son:

- Opera en la banda ISM de 2.4 GHz, específicamente en el rango de 2400 a 2483.5 MHz, con un ancho de banda de 2 MHz por canal.
- Dispone de 40 canales de radiofrecuencia (RF) y admite una potencia máxima de transmisión de hasta 20 dBm.
- La tasa máxima de transmisión de datos alcanza aproximadamente 1.4 Mbps.
- El alcance máximo, para tasas de datos reducidas (125 y 500 kbps), puede llegar a aproximadamente 1000 m en condiciones favorables.

A.2. BLE Protocol Stack

La arquitectura del protocolo BLE se organiza en una estructura en capas, donde cada una cumple una función específica dentro del sistema.

Apéndice A. Fundamentos de Bluetooth Low Energy

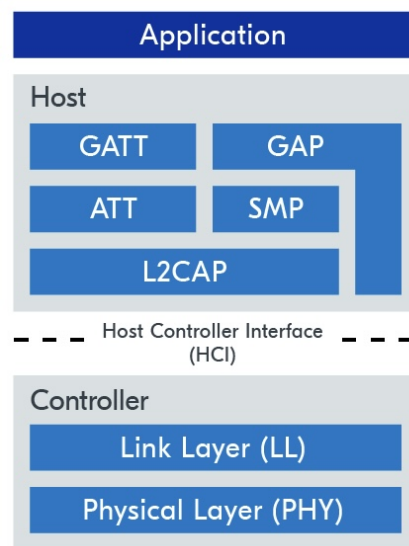


Figura A.1: BLE Protocol Stack Application *Imagen extraída del curso Bluetooth Low Energy Fundamentals de Nordic Semiconductor* [29].

En el nivel superior se encuentra la capa de aplicación, que constituye la interfaz con la que el usuario interactúa a través de distintas APIs para hacer uso del módulo BLE. Por debajo, el *Host* es responsable de la gestión de los datos, definiendo cómo los dispositivos BLE almacenan e intercambian información. En este trabajo, el *Host* se encuentra implementado mediante *Zephyr Bluetooth Host* [28].

En los niveles inferiores se encuentra el *Controller*, que abarca las capas más cercanas al hardware, incluyendo la radio física encargada de la generación, modulación y transmisión de las señales. En este caso, el *Controller* es provisto por el *SoftDevice Controller*.

La comunicación entre el *Host* y el *Controller* se realiza a través de la *Host Controller Interface* (HCI). La arquitectura en capas del protocolo BLE se muestra en la Figura A.1.

El *Controller* se compone de las siguientes capas:

- *Physical Layer (PHY)*: define cómo la información es modulada en señales de radiofrecuencia, así como los mecanismos de transmisión y recepción.
- *Link Layer (LL)*: gestiona el estado de la radio, el cual puede ser *standby*, *advertising*, *scanning*, *initiating* o *connection*, además de encargarse del control del enlace.

A continuación se presentan algunas capas del *stack*, y en las secciones siguientes se profundiza en aquellas que resultan relevantes para la aplicación.

A.2.1. Capa GAP

En BLE existen dos modos principales de comunicación:

- *Connection-oriented communication*: se establece una conexión dedicada entre dispositivos, permitiendo una comunicación bidireccional.
- *Broadcast communication*: los dispositivos transmiten información sin haber establecido una conexión previa, mediante el envío de paquetes a todos los dispositivos dentro del rango de cobertura.

La capa *GAP* (*Generic Access Profile*) define los roles de los dispositivos dentro de una red BLE, así como los mecanismos mediante los cuales estos anuncian su presencia y establecen conexiones. Esto se lleva a cabo a través de los procesos de *advertising* y *scanning*. El *advertising* consiste en la transmisión de paquetes de anuncio, ya sea para difundir información o para permitir que otros dispositivos detecten su presencia. Por su parte, el *scanning* corresponde al proceso de escucha de dichos paquetes de anuncio.

Dentro de la capa *GAP* se definen dos roles principales:

- *Central*: dispositivo que realiza el *scanning* e inicia conexiones con dispositivos periféricos una vez detectados sus paquetes de *advertising*.
- *Peripheral*: dispositivo que transmite paquetes de *advertising* y acepta solicitudes de conexión provenientes de dispositivos centrales.

Un dispositivo periférico puede aceptar múltiples conexiones si reinicia el proceso de *advertising* luego de establecer una conexión previa.

Adicionalmente, BLE soporta distintas topologías de red:

- *Broadcast topology*: la transmisión de datos se realiza mediante paquetes de *advertising*, sin establecer conexión. En este esquema, el *broadcaster* transmite información y el *observer* la recibe mediante *scanning*. Sus principales ventajas son el bajo consumo energético y la capacidad de transmitir a un número ilimitado de dispositivos. Como desventajas, la cantidad de datos está limitada por el tamaño del paquete de *advertising* y no existe confirmación de recepción.
- *Connected topology*: se establece una conexión entre un dispositivo central y uno periférico antes de la transmisión de datos. Esto permite una comunicación bidireccional y mayores tasas de transferencia. Si bien la especificación de BLE no define un límite estricto en la cantidad de conexiones, este se encuentra acotado por el ancho de banda disponible y las capacidades del hardware.
- *Multi-role topology*: un mismo dispositivo puede operar simultáneamente como central y periférico en distintas conexiones. Esta configuración es común en dispositivos tipo *hub*, que reciben datos de múltiples sensores y los retransmiten a otro dispositivo.

A.2.2. Capas GATT y ATT

Las capas *GATT* (*Generic Attribute Profile*) y *ATT* (*Attribute Protocol*) definen el modo en que los datos son representados e intercambiados entre dispositivos BLE durante una conexión.

Apéndice A. Fundamentos de Bluetooth Low Energy

La capa *ATT* constituye la base del intercambio de datos, proporcionando los mecanismos para su transmisión, recepción y gestión, bajo un modelo de arquitectura cliente-servidor. Estos roles son independientes de los definidos en la capa *GAP* (central y periférico). En este esquema, el servidor *GATT* es el dispositivo que almacena la información y ofrece los métodos necesarios para acceder a ella, mientras que el cliente *GATT* es el encargado de realizar las operaciones de acceso sobre dichos datos.

Adicionalmente, la capa *ATT* define una estructura de datos denominada *atributo*, que es utilizada por la capa *GATT* para organizar la información de manera jerárquica. En este contexto, la capa *GATT* clasifica los atributos en perfiles, servicios y características, cuya descripción se detalla en la Sección A.5.

Al establecerse una conexión, el cliente inicia un proceso denominado *service discovery*, mediante el cual interroga al servidor con el objetivo de obtener información acerca de los atributos disponibles.

A.2.3. Capa PHY

A diferencia de las capas previamente descritas, la capa *PHY* (*Physical Layer*) forma parte del *Controller*. En esta capa se definen los esquemas de modulación y codificación utilizados para la transmisión de datos a través de la radio BLE. La elección de la *PHY* implica un compromiso entre tasa de transferencia de datos, consumo energético y alcance de comunicación.

BLE define tres modos principales de *PHY*:

- *1M PHY*: es la *PHY* básica soportada por todos los dispositivos BLE y la utilizada por defecto al iniciar una conexión. Opera con una tasa de transmisión de 1 Mbps.
- *2M PHY*: introducida en versiones más recientes de BLE, permite una tasa de transmisión de 2 Mbps. Esto reduce el tiempo durante el cual la radio permanece activa, disminuyendo el consumo energético. Como desventaja, presenta un menor alcance debido a la reducción en la sensibilidad del receptor.
- *Coded PHY*: diseñada para aplicaciones que requieren mayor alcance, a costa de una menor tasa de transmisión. Emplea esquemas de codificación que mejoran la robustez frente a errores. Existen dos configuraciones principales:
 - $S=2$: cada bit se representa mediante 2 símbolos, resultando en una tasa de transmisión de 500 kbps.
 - $S=8$: cada bit se representa mediante 8 símbolos, resultando en una tasa de transmisión de 125 kbps.

Al aumentar el número de símbolos por bit, se mejora la capacidad de detección y corrección de errores, lo que incrementa el alcance efectivo de la comunicación.

A.3. Advertising

El *advertising* en BLE se utiliza principalmente para la transmisión de información a dispositivos cercanos y para anunciar la presencia de un dispositivo, permitiendo que

otros dispositivos dentro del rango puedan detectarlo e iniciar una conexión.

A.3.1. Proceso de advertising

La información transmitida en los paquetes de *advertising*, junto con el intervalo de transmisión, impacta directamente en el consumo energético. Considerando que el dispositivo permanece en este estado siempre que no se encuentra conectado, se busca optimizar dicho consumo.

Durante el estado de *advertising*, el dispositivo periférico transmite paquetes denominados *advertising packets*, los cuales permiten anunciar su presencia y, eventualmente, transmitir información a dispositivos cercanos. Estos paquetes se envían de manera periódica.

El intervalo entre transmisiones se denomina *advertising interval*, y puede tomar valores entre 20 ms y 10.24 s, con una resolución de 0.625 ms. La elección de este intervalo implica un compromiso entre consumo energético y capacidad de descubrimiento (*discoverability*): intervalos más cortos aumentan la frecuencia de transmisión de paquetes, facilitando una detección más rápida por parte de dispositivos centrales, pero incrementando el consumo de energía.

Adicionalmente, se introduce un retardo aleatorio (*advertising delay*) de entre 0 y 10 ms entre eventos de *advertising*, con el objetivo de reducir la probabilidad de colisiones entre dispositivos que operan con intervalos similares.

La comunicación en BLE se realiza sobre 40 canales de radiofrecuencia: 3 canales primarios utilizados para *advertising* y 37 canales secundarios destinados principalmente a la transmisión de datos durante una conexión.

En la Figura A.2 se muestra la distribución espectral de los canales de BLE dentro de la banda ISM de 2.4 GHz, donde cada canal presenta un ancho de banda de 2 MHz. Los tres canales de *advertising* se seleccionan estratégicamente en regiones del espectro con menor interferencia, minimizando el impacto de otras tecnologías que operan en la misma banda, como Wi-Fi.

Los paquetes de *advertising* se transmiten secuencialmente en estos tres canales para incrementar la robustez de la comunicación, mientras que los dispositivos en modo *scanning* escuchan en dichos canales para detectar la presencia de dispositivos periféricos.

Como se mencionó previamente, los paquetes de *advertising* son transmitidos por el dispositivo periférico para ser detectados por el dispositivo central mediante el proceso de *scanning*. Este proceso se realiza de manera periódica, con un período denominado *scan interval*. Por su parte, la *scan window* define el intervalo de tiempo dentro de cada período en el cual el dispositivo permanece activamente escuchando en busca de paquetes de *advertising*. Ambos parámetros pueden configurarse en el rango de 2.5 ms a 10.24 s, con una resolución de 0.625 ms.

En la Figura A.3 se ilustran los intervalos de *scanning* y *advertising* para un dispositivo central y uno periférico, respectivamente. En el ejemplo mostrado, el *scan interval* es de 50 ms y la *scan window* de 25 ms, lo que corresponde a un *duty cycle* del 50%. En la parte inferior de la figura se observan los eventos de *advertising*, con un intervalo de 20 ms, donde la transmisión se realiza secuencialmente a través de los tres canales primarios (indicados en verde, celeste y azul).

Apéndice A. Fundamentos de Bluetooth Low Energy

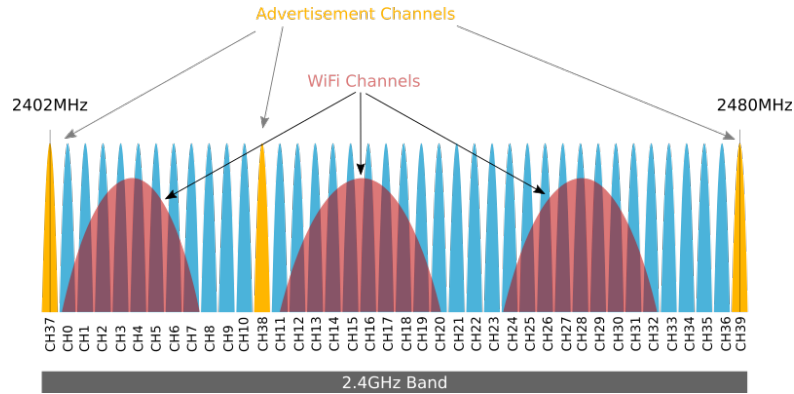


Figura A.2: Canales de Advertisement de BLE. Imagen extraída de *BLE Advertisement Channels* [30]

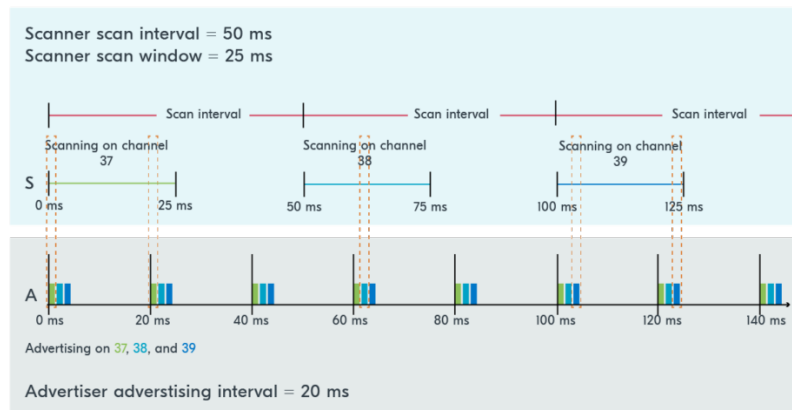


Figura A.3: Intervalos de escaneo y de advertising. Imagen extraída del curso *Bluetooth Low Energy Fundamentals* de Nordic Semiconductor [29]

De manera análoga, el proceso de *scanning* también se lleva a cabo de forma secuencial sobre estos tres canales. Al igual que en el caso del *advertising*, existe un compromiso entre consumo energético y desempeño: una mayor *scan window* dentro de cada *scan interval* incrementa la probabilidad de detección rápida de dispositivos, pero a costa de un mayor consumo de energía. Por lo tanto, la configuración de estos parámetros debe balancear la velocidad de descubrimiento y establecimiento de conexión frente a las restricciones de consumo.

La estrategia de *advertising* adoptada en el sistema se describe en detalle en la Sección 5.3.2, donde se presentan los parámetros configurados y los criterios considerados para su selección en función de los requerimientos de la aplicación.

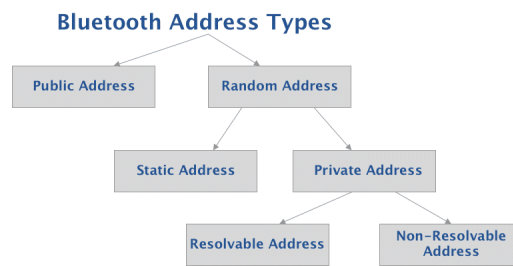


Figura A.4: Clasificación de direcciones BLE. *Imagen extraída de Bluetooth Addresses & Privacy in Bluetooth Low Energy* [31]

A.3.2. Tipos de advertising

Existen distintos tipos de *advertising* definidos en BLE, los cuales se especifican dentro de los paquetes de *advertising* transmitidos por el dispositivo periférico. Estos tipos pueden agruparse en tres categorías principales según su finalidad: conectables, escaneables y dirigidos.

El *advertising* conectable permite que un dispositivo central establezca una conexión con el periférico. El *advertising* escaneable habilita al periférico a responder solicitudes de escaneo (*scan requests*) provenientes de un central. Por su parte, el *advertising* dirigido está orientado a un dispositivo específico, identificado previamente.

A partir de estas categorías, se definen cuatro tipos principales de *advertising*:

- *ADV_IND*: es el tipo de *advertising* más común, siendo tanto conectable como escaneable. Permite que el dispositivo central envíe solicitudes de escaneo, las cuales son respondidas por el periférico mediante *scan responses*. Posteriormente, puede establecerse una conexión.
- *ADV_DIRECT_IND*: corresponde a un *advertising* dirigido y conectable, pero no escaneable. Se utiliza principalmente en escenarios donde el periférico ya conoce al dispositivo central y se desea establecer una reconexión rápida, por ejemplo, tras una desconexión previa.
- *ADV_SCAN_IND*: es un tipo de *advertising* escaneable pero no conectable. Permite la transmisión de información adicional mediante respuestas de escaneo, sin establecer una conexión.
- *ADV_NONCONN_IND*: corresponde a un *advertising* no conectable ni escaneable. Se emplea en aplicaciones donde únicamente se requiere difundir información, priorizando un consumo energético reducido.

A.3.3. Bluetooth address

Los dispositivos BLE se identifican mediante una dirección única de 48 bits. Existen distintos tipos de direcciones, como se muestra en la Figura A.4.



Figura A.5: Formato de una dirección random private resolvable (en formato little-endian)

Una dirección pública es asignada al dispositivo durante su proceso de fabricación y no puede ser modificada posteriormente. Su utilización requiere el registro ante la IEEE, lo que implica un costo asociado.

Las direcciones que no son públicas corresponden a direcciones *random*, las cuales se clasifican en estáticas y privadas.

Las direcciones *random estáticas* constituyen una alternativa a las direcciones públicas, sin requerir registro ante la IEEE. Estas pueden asignarse de forma permanente al dispositivo o bien establecerse en cada arranque, pero no pueden modificarse durante la ejecución.

En cuanto a su estructura, las direcciones *random estáticas* se caracterizan por tener los dos bits más significativos fijados en ‘1’, mientras que los 46 bits restantes son generados de manera aleatoria por el fabricante.

Es importante destacar que todo dispositivo BLE debe disponer de una dirección pública o de una dirección *random estática*.

Las direcciones *random privadas*, ya sean resolubles o no resolubles, se utilizan de manera adicional a la dirección pública o estática, con el objetivo de mejorar aspectos de seguridad y privacidad, tal como se detalla a continuación.

Las direcciones privadas resolubles (*resolvable private addresses*) se utilizan para evitar que dispositivos no autorizados puedan rastrear un dispositivo, permitiendo al mismo tiempo que dispositivos confiables puedan reconocerlo. Este tipo de dirección puede ser “resuelta” por un dispositivo autorizado mediante el uso de una clave secreta compartida denominada *IRK* (*Identity Resolving Key*).

La dirección se genera a partir de la combinación de esta clave *IRK* con un valor aleatorio. Un dispositivo es considerado confiable cuando se encuentra vinculado (*bonded*). El *bonding* es un proceso opcional posterior al emparejamiento (*pairing*), que consiste en el almacenamiento de claves en ambos dispositivos, permitiendo futuras reconexiones seguras sin necesidad de repetir el proceso de emparejamiento. Entre las claves intercambiadas durante este proceso se encuentra la *IRK*, que habilita la resolución de direcciones privadas resolubles.

Este tipo de dirección se actualiza periódicamente, con un período recomendado de aproximadamente 15 minutos, con el objetivo de dificultar su rastreo. Su estructura se muestra en la Figura A.5. Los dos bits más significativos toman el valor ‘01’, seguidos por 22 bits generados aleatoriamente. Estos 24 bits conforman el campo denominado *prand* (*private random part*), el cual varía en el tiempo. Los 24 bits menos significativos corresponden al *hash*, el cual se calcula a partir del *prand* y la clave *IRK*.

El otro tipo de direcciones privadas *random* son las no resolubles (*non-resolvable private addresses*). Estas direcciones se actualizan periódicamente, pero no pueden ser

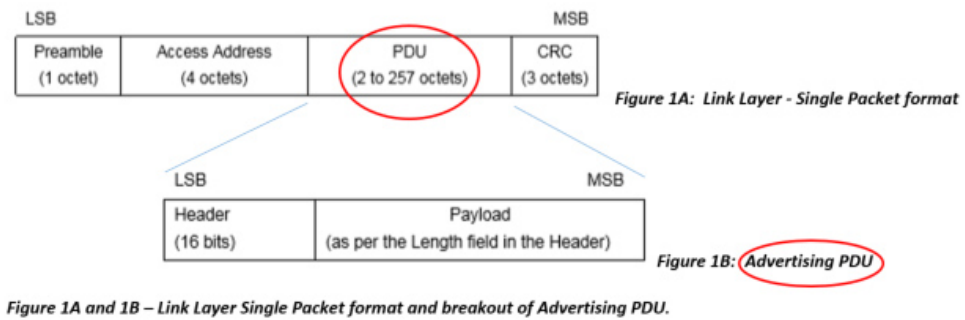


Figura A.6: Formato de un paquete de advertising. Imagen extraída de *Bluetooth® Low Energy: It starts with advertising* [32]

resueltas por ningún dispositivo, lo que impide completamente la identificación del dispositivo y previene su rastreo. Debido a esta característica, no son ampliamente utilizadas en aplicaciones donde se requiere reconexión o identificación del dispositivo.

A.3.4. Paquetes de advertising

En la Figura A.6 se muestra el formato de un paquete de *advertising*, el cual, como se mencionó anteriormente, puede utilizarse tanto para anunciar la presencia de un dispositivo como para la transmisión de datos, por ejemplo en aplicaciones tipo *beacon*.

El paquete se compone de cuatro campos principales: un byte de preámbulo, cuatro bytes de dirección de acceso (*access address*), entre 2 y 257 bytes correspondientes a la *Protocol Data Unit* (PDU), y tres bytes de código de redundancia cíclica (*CRC*).

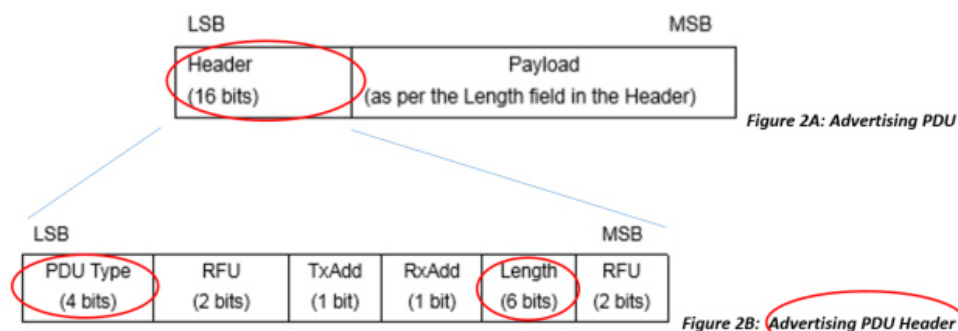
Dentro de la PDU se especifica el tipo de paquete, permitiendo distinguir entre paquetes de *advertising* y de datos. Este desarrollo se centra exclusivamente en los paquetes de *advertising*.

El PDU de *advertising* está compuesto por un encabezado (*Header*) de 2 bytes y un campo de datos (*Payload*) de longitud variable, entre 0 y 37 bytes, como se muestra en la Figura A.7.

El *Header* comienza con el campo *PDU Type*, que indica el tipo de *advertising* utilizado (según lo definido en la Sección A.3.2). A continuación, se incluye un bit reservado para uso futuro (*RFU*) y un bit de selección de canal, utilizado en dispositivos que soportan el algoritmo de selección de canal #2 introducido en Bluetooth 5.0, el cual permite una asignación más eficiente de los canales de transmisión.

Posteriormente, se encuentran los bits *TxAdd* (*Transmit Address*) y *RxAdd* (*Receive Address*), que indican si las direcciones del transmisor y del receptor, respectivamente, son públicas o *random*. Finalmente, el encabezado incluye un campo de 8 bits que especifica la longitud del *Payload*.

Luego, el *payload* del PDU de un paquete de *advertising* presenta el formato mostrado en la Figura A.8. Este comienza con un campo de 6 bytes que corresponde a la dirección del dispositivo que realiza el *advertising* (denominado *AdvA* en la figura), seguido por el campo *AdvData*, que contiene la información de *advertising* y cuyo tamaño varía entre



Figures 2A and 2B – Advertising PDU header, specifically PDU type and Length.

Figura A.7: PDU del paquete de advertising. Imagen extraída de *Bluetooth® Low Energy: It starts with advertising* [32]

0 y 31 bytes.

El formato de la *Advertisement Data* también se ilustra en la Figura A.8. Esta se compone de múltiples estructuras denominadas *AD structures*, cada una de las cuales incluye un campo de longitud de un byte, un campo de tipo (*AD type*) de un byte, y un campo de datos (*AD data*) de tamaño variable. El campo de tipo indica el contenido de la información transmitida, pudiendo representar, por ejemplo, el nombre del dispositivo, la potencia de transmisión, un identificador o banderas de configuración, entre otros. Estos tipos están definidos en la especificación de Bluetooth y se conocen como *EIR/AD data type definitions*.

En ciertos casos, como en el *advertising* dirigido, no se incluye el campo *AdvData*. En su lugar, este es reemplazado por la dirección del dispositivo destinatario, también de 6 bytes.

A.4. Conexiones

Una conexión entre dispositivos BLE se establece cuando un dispositivo central detecta, mediante *scanning*, un paquete de *advertising* de un periférico y envía una solicitud de conexión. A partir de este momento, se crea un canal de comunicación bidireccional. El periférico únicamente puede rechazar esta solicitud si utiliza mecanismos de filtrado, como listas de dispositivos permitidos. Adicionalmente, una vez establecida la conexión, el periférico puede finalizarla si así lo requiere.

Para poder recibir solicitudes de conexión, el periférico habilita una breve ventana de recepción inmediatamente después de cada evento de *advertising*.

Una vez establecida la conexión, la comunicación deja de utilizar los canales de *advertising* (37, 38 y 39) y pasa a emplear los canales de datos (0 a 36). Dado que estos canales operan en la banda ISM de 2.4 GHz, están sujetos a interferencias provenientes

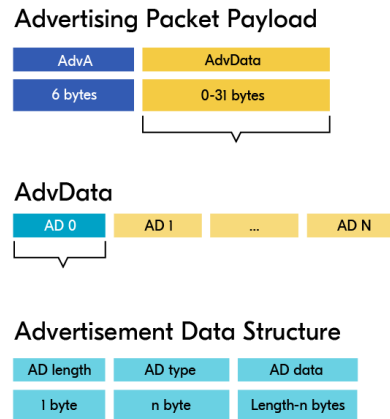


Figura A.8: Formato del *payload* del PDU del paquete de *advertising*. Imagen extraída de *Bluetooth Low Energy Fundamentals* [29]

de otras tecnologías. Para mitigar este efecto y mejorar la robustez del enlace, BLE implementa un mecanismo de *frequency hopping*, mediante el cual el canal de transmisión cambia periódicamente. De este modo, si un canal presenta interferencia, la información puede retransmitirse en otro canal durante el siguiente intervalo de conexión. Este proceso continúa hasta que se recibe una confirmación por parte del receptor o se finaliza la conexión.

La Figura A.9 ilustra el funcionamiento de la comunicación BLE durante la etapa de conexión. En intervalos de tiempo periódicos, denominados *connection intervals*, definidos inicialmente por el dispositivo central al establecer la conexión, ambos dispositivos activan sus radios para intercambiar datos. Este intervalo puede ser modificado dinámicamente durante la conexión.

En cada *connection interval*, los dispositivos deben intercambiar al menos un paquete para mantener la sincronización temporal. En ausencia de datos útiles, se transmiten paquetes vacíos. En caso de que la cantidad de información a transmitir exceda la capacidad de un intervalo, esta se distribuye en múltiples intervalos consecutivos. Si bien es posible transmitir más de un paquete dentro de un mismo intervalo, cualquier transmisión adicional deberá esperar hasta el siguiente intervalo si no se completa dentro del actual.

Por último, la finalización de una conexión puede producirse por dos mecanismos: desconexión iniciada por la aplicación o por *supervision timeout*.

La desconexión por aplicación ocurre cuando alguno de los dispositivos envía un paquete de terminación (*termination packet*). Este paquete incluye un campo denominado *reason*, que indica la causa de la desconexión, la cual puede corresponder tanto a una decisión explícita de uno de los dispositivos como a un error ocurrido durante la conexión.

Por otro lado, el *supervision timeout* es un temporizador que provoca la desconexión

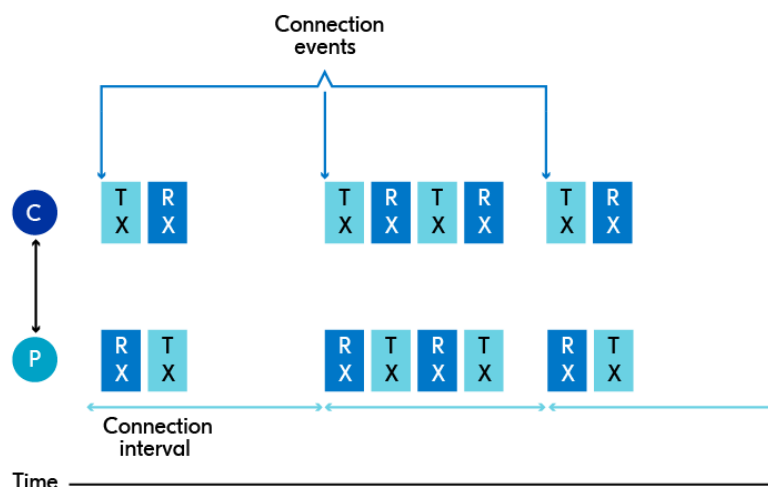


Figura A.9: Comunicación BLE durante etapa de conexión. *Imagen extraída de Bluetooth Low Energy Fundamentals* [29]

cuando uno de los dispositivos deja de responder dentro de un período determinado. Este parámetro es configurable y forma parte de los parámetros de conexión de BLE, los cuales se describen a continuación.

A.4.1. Parámetros de conexión

Los parámetros de conexión se intercambian entre los dispositivos al momento de establecer la conexión, siendo enviados por el dispositivo central en la solicitud de conexión.

El intervalo de conexión (*connection interval*) es definido inicialmente por el central en dicha solicitud. Si bien su gestión se realiza a nivel del *BLE stack*, la aplicación es la encargada de configurar su valor. El *supervision timeout* también se establece en la solicitud de conexión y determina el tiempo máximo durante el cual pueden dejar de recibirse paquetes válidos antes de considerar que la conexión se ha perdido. Por último, la latencia del periférico (*peripheral latency*) define la cantidad de eventos de conexión consecutivos en los que el periférico puede permanecer inactivo, es decir, sin intercambiar datos. Este parámetro permite reducir el consumo energético del dispositivo sin necesidad de aumentar el intervalo de conexión.

Otros parámetros adoptan valores por defecto al inicio de la conexión, con el objetivo de garantizar la compatibilidad con versiones anteriores de Bluetooth. Uno de estos parámetros es la configuración de la capa PHY. Por defecto, BLE utiliza la *1M PHY*; sin embargo, a partir de Bluetooth 5.0, es posible emplear también *2M PHY* y *Coded PHY*. La configuración de la PHY puede actualizarse dinámicamente una vez establecida

la conexión.

El periférico puede solicitar cambios en estos parámetros, los cuales pueden ser aceptados o rechazados por el central. En el proyecto se optó por solicitar estos cambios para, en caso de ser posible, mejorar el consumo energético. Los parámetros que se solicitaron modificar son el intervalo de conexión y la capa PHY utilizada.

Otros parámetros inicializados con valores estándar son el tamaño de datos (*data length*) y la *Maximum Transmission Unit* (MTU). La diferencia entre ambos radica en que la MTU define la cantidad máxima de bytes que pueden transferirse en una operación de la capa GATT, mientras que el *data length* determina el número máximo de bytes que pueden incluirse en un paquete a nivel de enlace.

Desde el punto de vista de eficiencia energética y de uso del canal, resulta conveniente transmitir la mayor cantidad de información posible en un único paquete, reduciendo así el tiempo de actividad de la radio y la sobrecarga asociada a múltiples encabezados. Cada paquete introduce una sobrecarga adicional, por lo que minimizar su cantidad contribuye a optimizar el rendimiento.

El *data length* puede alcanzar hasta 251 bytes. No obstante, el tamaño efectivo del *payload* es de 244 bytes, debido a la sobrecarga introducida por los encabezados de las capas superiores: 4 bytes correspondientes a L2CAP y 3 bytes a ATT.

Durante la conexión, cualquiera de los dispositivos puede solicitar un cambio en estos parámetros, y el otro dispositivo responde si soporta o no estos nuevos valores.

La configuración de los parámetros de conexión BLE adoptada en este trabajo se describe en detalle en la Sección 5.3.4, donde se presentan los valores seleccionados y los criterios considerados para su elección.

A.5. Intercambio de datos

El intercambio de datos entre dispositivos conectados se realiza a través del *Generic Attribute Profile* (GATT), el cual se apoya en el *Attribute Protocol* (ATT). Estos componentes definen la forma en que la información se organiza, representa y transfiere durante una conexión. Mientras que ATT establece el formato de los atributos y las reglas de acceso a los mismos, GATT proporciona una estructura jerárquica estandarizada que permite organizarlos en servicios y características, facilitando la interoperabilidad entre dispositivos de distintos fabricantes.

El modelo de comunicación en BLE sigue una arquitectura cliente-servidor. En este esquema, el servidor GATT mantiene una tabla de atributos que contiene los servicios y características disponibles, mientras que el cliente accede a dichos atributos mediante operaciones específicas.

Para poder interactuar con la información, el cliente debe realizar previamente un proceso de *service discovery*, mediante el cual identifica los servicios y características expuestos por el servidor, antes de iniciar operaciones de lectura, escritura o actualización de valores.

En el marco del proyecto, esto puede observarse en la aplicación móvil, donde, al establecer una conexión, se visualizan los servicios y características implementados. A través de esta interfaz es posible realizar tanto la escritura como la lectura de dichas características.

A.5.1. Operaciones de la capa GATT

Las operaciones en la capa GATT pueden clasificarse en función del dispositivo que las inicia. Por un lado, se encuentran las operaciones iniciadas por el cliente, en las cuales este solicita acceso a los datos almacenados en el servidor GATT. En este contexto, el cliente puede realizar operaciones de lectura o escritura sobre atributos.

Cuando un cliente desea leer el valor de un atributo, envía una solicitud de lectura al servidor, que responde devolviendo el valor correspondiente. De manera análoga, para escribir un valor, el cliente envía una solicitud de escritura junto con los datos, los cuales deben ajustarse al formato del atributo de destino. Si el servidor acepta la operación, responde con un acuse de recibo. Alternativamente, si está habilitada la operación de *write without response*, el cliente puede escribir en un atributo sin esperar confirmación.

Por otro lado, existen operaciones iniciadas por el servidor, en las cuales este envía información al cliente sin una solicitud previa. En este caso, el servidor puede utilizar mecanismos de *notification* o *indication*. Ambos permiten enviar automáticamente el valor de un atributo al cliente; sin embargo, la *notification* no requiere confirmación, mientras que la *indication* sí implica el envío de una respuesta por parte del cliente.

A.5.2. Atributos

La capa ATT define la forma en que los datos se almacenan y acceden dentro de la base de datos de un servidor. La información se organiza en estructuras denominadas *atributos*, que constituyen la unidad fundamental sobre la cual se construyen tanto las capas ATT como GATT.

Un atributo contiene tanto los datos de usuario como metadatos que describen sus características, tales como su tipo, permisos de acceso y requisitos de seguridad. El intercambio de información entre dispositivos se realiza, en última instancia, mediante la lectura y escritura de estos atributos.

Cuando se consideran únicamente atributos individuales, se hace referencia a un servidor ATT. Sin embargo, al organizar estos atributos en servicios y características bajo una estructura jerárquica, se habla de un servidor GATT.

Un atributo está compuesto por cuatro campos principales, como se muestra en la Figura A.10:

Cada uno de estos campos se define de la siguiente manera:

- **Identificador (Handle):** índice único de 16 bits que identifica a cada atributo dentro de la tabla de atributos. Es asignado por el *stack* y se utiliza para direccionar el atributo.
- **Tipo (UUID):** identificador único universal (*UUID*) que especifica el tipo de atributo.
- **Permisos:** definen el nivel de seguridad requerido (por ejemplo, cifrado y/o autorización), así como las operaciones permitidas sobre el atributo (lectura y/o escritura).
- **Valor:** contiene los datos propiamente dichos almacenados en el atributo.

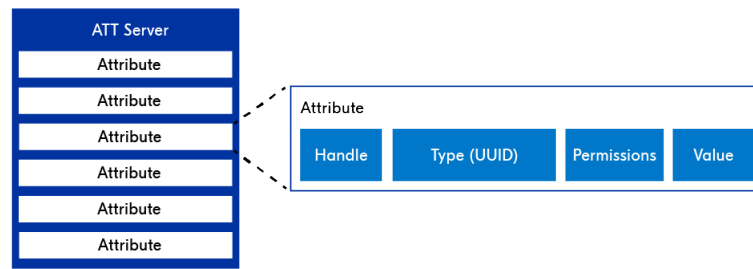


Figura A.10: Bloques de datos de un atributo. *Imagen extraída de Bluetooth Low Energy Fundamentals* [29]

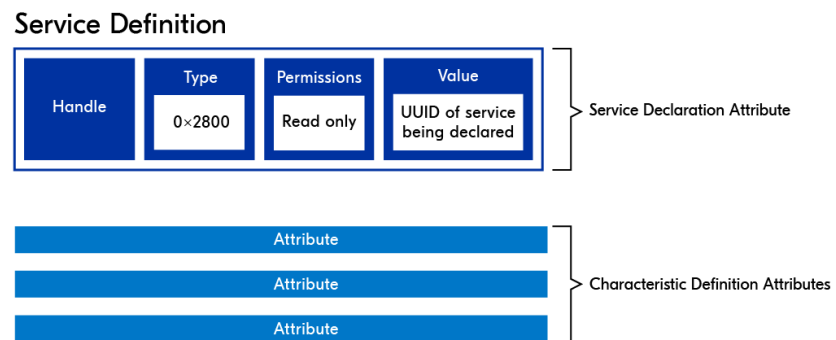


Figura A.11: Formato de un atributo de servicio *Imagen extraída de Bluetooth Low Energy Fundamentals* [29]

A.5.3. Servicios

Un servicio se compone de múltiples atributos organizados según un formato definido por la capa GATT.

La definición de un servicio comienza siempre con un atributo de servicio, el cual contiene los metadatos asociados al mismo e indica el inicio de un servicio dentro de la secuencia de atributos almacenados en el servidor GATT. En la Figura A.11 se observa el formato que debe tener este atributo.

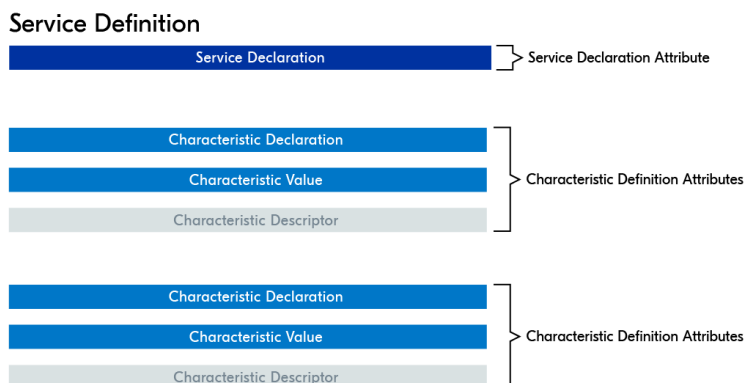


Figura A.12: Conjunto de atributos que forman una característica. *Imagen extraída de Bluetooth Low Energy Fundamentals* [29]

En el proyecto se definió un servicio denominado *Neurostimulation Service* (NSS), cuyo detalle se presenta en la Sección 5.3.7.

A.5.4. Características

A continuación, tal como se ilustra en la Figura A.12, un servicio puede incluir ninguna o varias características. Cada característica está formada, como mínimo, por dos atributos, pudiendo incorporar atributos adicionales según la funcionalidad requerida.

La definición de una característica comienza con un atributo de declaración de característica, el cual señala el inicio de una característica dentro de la secuencia de atributos que conforman un servicio.

El campo *Tipo* de este atributo contiene un UUID específico utilizado exclusivamente para la declaración de características. Asimismo, el atributo de declaración posee permisos de solo lectura, identificados por el valor 0x2803, lo que garantiza que los clientes puedan acceder a su contenido, pero no modificarlo.

El campo *Valor* del atributo de declaración incluye información relevante sobre la característica definida, y se encuentra compuesto por tres elementos principales:

- **Propiedades de la característica:** especifican las operaciones GATT que pueden realizarse sobre la característica, tales como lectura, escritura o notificación.
- **Identificador del valor de la característica:** corresponde al identificador (o dirección) del atributo que almacena los datos, es decir, el atributo de valor asociado a la característica.
- **UUID de la característica:** indica el identificador único universal que define el tipo de característica declarada.

Service Definition

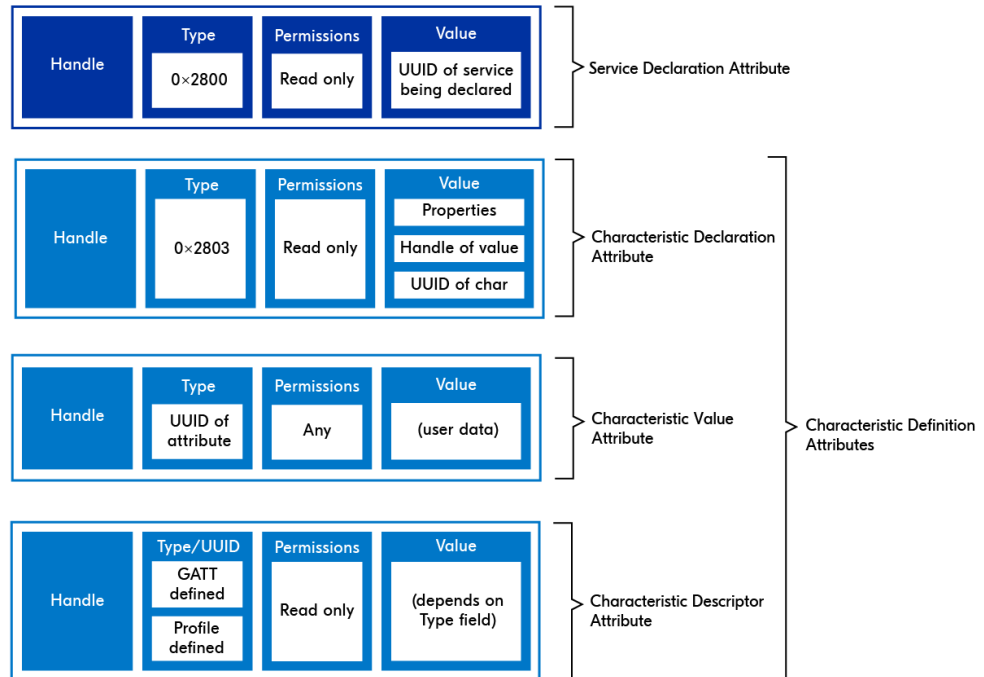


Figura A.13: Formato de un servicio conformado por una característica. *Imagen extraída de Bluetooth Low Energy Fundamentals* [29]

A continuación del atributo de declaración se encuentra el atributo de valor de la característica, el cual contiene los datos de usuario. Su identificador y su tipo coinciden con los especificados en el campo *Valor* del atributo de declaración. En este atributo, el campo *Valor* almacena la información de interés, mientras que el campo *Permisos* determina si el cliente puede acceder a estos datos mediante operaciones de lectura y/o escritura.

Por último, los descriptores de característica son atributos opcionales que aportan información adicional sobre la característica, proporcionando al cliente un mayor contexto sobre su uso y comportamiento. Existen distintos tipos de descriptores, los cuales pueden clasificarse en dos categorías: aquellos definidos por la especificación GATT y los definidos de manera personalizada.

En la Figura A.13 se observa el formato final de un servicio conformado por una característica.

A.6. Seguridad en BLE

El protocolo BLE provee diferentes funciones de seguridad, como autenticación, integridad, confidencialidad y privacidad.

A.6.1. Proceso de emparejamiento

Una de las prácticas fundamentales para garantizar la seguridad en comunicaciones inalámbricas consiste en el cifrado del enlace, mediante el cual la información transmitida se transforma en un formato accesible únicamente para dispositivos autorizados, proporcionando así confidencialidad e integridad de los datos.

Para establecer un canal cifrado, ambos dispositivos deben compartir un conjunto común de claves. El proceso mediante el cual estas claves son generadas, distribuidas y autenticadas se denomina proceso de *pairing* (emparejamiento).

El emparejamiento implica que los dispositivos ejecuten este procedimiento cada vez que se desee establecer un enlace cifrado, salvo que se haya realizado un proceso de *bonding*. El *bonding* consiste en el almacenamiento persistente de las claves de cifrado en ambos dispositivos, permitiendo restablecer conexiones seguras en el futuro sin necesidad de repetir el proceso de emparejamiento. Además, este proceso contribuye a garantizar la autenticidad de los dispositivos, al permitir verificar su identidad en conexiones posteriores.

Adicionalmente, durante el *bonding* pueden intercambiarse y almacenarse claves de identidad, lo que posibilita el reconocimiento mutuo entre dispositivos en conexiones posteriores.

Los mecanismos de autenticación, emparejamiento y *bonding* adoptados en el sistema se describen en detalle en la Sección 5.3.6, donde se justifican las decisiones de diseño tomadas para la gestión segura de las conexiones BLE.

BLE define tres fases en el proceso de cifrado, explicadas a continuación.

Fase 1: Inicio del emparejamiento

El proceso de emparejamiento se inicia cuando el dispositivo central envía una solicitud de emparejamiento (*pairing request*), la cual es respondida por el dispositivo periférico mediante una *pairing response*. Esta fase puede, en ciertos casos, dar inicio también al proceso de *bonding*.

Durante esta etapa, ambos dispositivos intercambian sus capacidades de emparejamiento, lo que permite determinar el método que será utilizado en la fase 2, así como las claves que se distribuirán en la fase 3.

Uno de los aspectos más relevantes de este intercambio es la definición de las capacidades de entrada/salida (*I/O capabilities*) de cada dispositivo, las cuales condicionan el nivel de seguridad que puede alcanzarse durante el emparejamiento. Estas capacidades se clasifican en las siguientes categorías:

- *DisplayOnly*: el dispositivo dispone únicamente de una pantalla.
- *DisplayYesNo*: el dispositivo cuenta con una pantalla y la capacidad de confirmar acciones (por ejemplo, selección de “sí” o “no”).

- *KeyboardOnly*: el dispositivo dispone únicamente de un teclado.
- *NoInputNoOutput*: el dispositivo no posee capacidades de entrada ni de salida.
- *KeyboardDisplay*: el dispositivo dispone tanto de teclado como de pantalla.

El neuroestimulador se encuentra en la categoría *NoInputNoOutput* teniendo en cuenta que es diseñado para ser implantado.

Fase 2: Realización del emparejamiento

En la fase 2 se lleva a cabo la generación de las claves utilizadas para el cifrado del enlace. El método de emparejamiento empleado depende de la información intercambiada durante la fase 1, particularmente de las capacidades de entrada/salida de los dispositivos.

En el esquema *LE Legacy Pairing*, los dispositivos intercambian una clave temporal (*Temporary Key*, TK), a partir de la cual se deriva una clave de corto plazo (*Short Term Key*, STK), utilizada para cifrar la conexión. Debido a las limitaciones de seguridad de este mecanismo, la versión Bluetooth 4.2 introdujo el modelo de *LE Secure Connections*. En este esquema, los dispositivos generan e intercambian claves basadas en criptografía de curva elíptica, a partir de las cuales se deriva una clave de largo plazo (*Long Term Key*, LTK). Durante el proceso de *bonding*, esta clave puede almacenarse en ambos dispositivos y utilizarse posteriormente para autenticar futuras conexiones y derivar las claves de sesión que protegen el enlace. Este mecanismo ofrece una mejora sustancial en términos de seguridad respecto a *LE Legacy Pairing*.

El proceso de emparejamiento contempla distintos métodos para el intercambio de información necesaria para la generación de claves. El nivel de seguridad alcanzado depende del método seleccionado. Los principales métodos son:

- *Just Works*: ambos dispositivos generan las claves a partir de la información intercambiada sin intervención significativa del usuario, requiriendo únicamente la aceptación de la conexión. Este método no proporciona protección frente a ataques de tipo *Man-in-the-Middle* (MITM).
- *Passkey Entry*: se muestra un código numérico de seis dígitos en uno de los dispositivos, el cual debe ser ingresado en el otro. Las capacidades de entrada/salida determinan qué dispositivo muestra el código y cuál lo introduce.
- *Out of Band (OOB)*: el intercambio de información crítica para la generación de claves se realiza a través de un canal alternativo distinto de BLE, como por ejemplo NFC, aumentando así la seguridad del proceso.
- *Numeric Comparison*: ambos dispositivos muestran un código numérico de seis dígitos, y el usuario debe confirmar que coinciden, seleccionando una opción afirmativa o negativa.

El método de emparejamiento a utilizar se determina en función de los parámetros intercambiados durante la fase 1, en particular la bandera *OOB* (que indica si alguno de los dispositivos soporta el intercambio de información a través de un canal externo a

Apéndice A. Fundamentos de Bluetooth Low Energy

BLE), la bandera *Man-In-The-Middle* (MITM), que especifica si se requiere protección frente a ataques de un tercero, y las capacidades de entrada/salida de los dispositivos.

La clave generada en esta etapa será empleada para el cifrado del enlace una vez finalizada la fase 2. En el caso de que únicamente se realice el proceso de *pairing*, sin *bonding*, el procedimiento concluye en esta fase, omitiéndose la fase 3 correspondiente a la distribución de claves.

Fase 3: Distribución de Claves

En esta fase se utiliza la clave de largo plazo (*Long Term Key*, LTK) como base para la distribución del resto de las claves de seguridad. En el esquema de emparejamiento tradicional (*LE Legacy Pairing*), la LTK se genera en esta etapa, mientras que en *LE Secure Connections* dicha clave es derivada durante la fase 2. En el proyecto se opta por *LE Secure Connections*, por lo que la LTK se deriva de la información intercambiada en la fase 2.

Adicionalmente, en esta fase se generan e intercambian otras claves destinadas a la identificación de los dispositivos en futuras conexiones, así como a permitir el restablecimiento del cifrado del enlace utilizando la misma LTK sin necesidad de repetir el proceso de emparejamiento.

A.6.2. Niveles de seguridad

BLE define cuatro niveles de seguridad dentro del modo de seguridad 1:

- Nivel 1: sin seguridad (sin cifrado ni autenticación)
- Nivel 2: cifrado sin autenticación
- Nivel 3: cifrado con autenticación
- Nivel 4: cifrado con autenticación utilizando *LE Secure Connections*

Toda conexión comienza en nivel 1 y puede escalar a niveles superiores según el método de emparejamiento utilizado. Los métodos como *Just Works* o *Passkey Entry* en *Legacy Pairing*, así como *LE Secure Connections* con *Just Works*, alcanzan el nivel 2, ya que no brindan protección contra ataques *Man-in-the-Middle* (MITM). Para mitigar este tipo de ataques, es necesario utilizar métodos autenticados (nivel 3 o superior), como *Out of Band (OOB)* o mecanismos de autenticación en *LE Secure Connections*, que permiten alcanzar el nivel 4.

Además, los permisos definidos en las características GATT pueden establecer el nivel de seguridad requerido para acceder a él. De esta manera, es posible restringir el acceso a ciertos datos únicamente cuando la conexión cumple con un nivel de seguridad determinado.

A.6.3. NFC

Los datos *NDEF* (*NFC Data Exchange Format*) se estructuran en mensajes. Cada mensaje consta de uno o más registros, que a su vez se componen de un encabezado y la

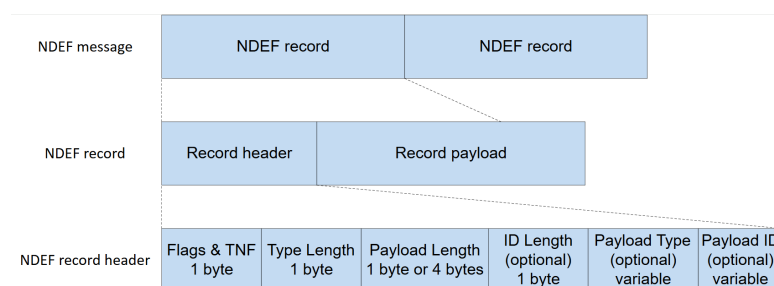


Figura A.14: Formato de los mensajes NDEF. *Imagen extraída del manual de nRF Connect SDK [33]*

carga útil del registro. El encabezado del registro contiene metadatos sobre, entre otros, el tipo y la longitud de la carga útil. La carga útil del registro constituye el contenido real del registro. En la Figura A.14 se observa el formato de estos mensajes.

El encabezado de un registro NDEF está compuesto por diferentes campos que definen su estructura y contenido. En primer lugar, se encuentra el campo de banderas y *TNF* (*Type Name Format*), de un byte de longitud, que es obligatorio en todo registro. A continuación, aparece el campo *Tipo Longitud*, también de un byte y de carácter obligatorio, el cual especifica la extensión del campo de *tipo de carga útil*; este puede tener valor cero si el registro no define un tipo específico.

El siguiente campo es la *Longitud de la carga útil*, cuya extensión puede ser de uno o de cuatro bytes, dependiendo de si se utiliza el indicador de registro corto (*Short Record, SR*). Este campo es obligatorio, aunque su valor puede ser cero si no se incluye contenido en la carga útil. Opcionalmente, puede añadirse el campo Longitud de identificación, de un byte, el cual solo está presente cuando se activa la bandera *IL*. Este especifica el tamaño del campo de identificación de la carga útil.

Posteriormente, se incorpora el campo *Tipo de carga útil*, de longitud variable y no obligatorio, que se incluye únicamente si la longitud del tipo definida anteriormente es mayor que cero. Este campo determina la naturaleza de los datos contenidos en el registro. Finalmente, el encabezado puede incluir un campo de *Identificación de la carga útil*, también de longitud variable, que resulta obligatorio únicamente si la bandera *IL* está configurada y la longitud de identificación es distinta de cero. Este campo se utiliza para asignar un identificador único a la carga útil del registro.

El campo *Banderas y TNF* corresponde a un byte dentro del encabezado del registro *NDEF* que concentra distintos indicadores de control y define el formato del nombre de tipo. En primer lugar, se encuentran las banderas *MB* (*Message Begin*) y *ME* (*Message End*), que señalan la posición del registro dentro del mensaje *NDEF*. La bandera *MB* se activa cuando el registro corresponde al primero del mensaje, mientras que *ME* se activa en el último. En el caso de que el mensaje contenga un único registro, ambas banderas se establecen simultáneamente.

La bandera *CF* (*Chunk Flag*) se utiliza cuando la carga útil de un registro se encuentra fragmentada en múltiples partes. En este caso, *CF* se activa en todos los fragmentos excepto en el último. No obstante, cabe señalar que la biblioteca *NFC* del *nRF Connect SDK* no soporta actualmente la fragmentación de registros.

La bandera *SR* (*Short Record*) determina el tamaño del campo de longitud de la carga

Apéndice A. Fundamentos de Bluetooth Low Energy

útil. Si esta bandera está activada, dicho campo ocupa un solo byte; en caso contrario, se reserva un espacio de cuatro bytes. Actualmente, el generador de *NDEF* del *SDK* únicamente admite longitudes codificadas en cuatro bytes.

Por su parte, la bandera *IL* (*ID Length Present*) indica si el campo *Longitud de Identificación* forma parte del encabezado. Cuando se encuentra activada, dicho campo se incluye para especificar el tamaño de la identificación de la carga útil. Finalmente, el campo *TNF* define la estructura y la semántica del campo Tipo de Carga Útil, determinando la forma en que los datos deben ser interpretados por el lector.

Referencias

- [1] D. L. Reyfman, “Técnicas de neuroestimulación. pain management - nyc.” 2026, accessed: 2026-04-16. [Online]. Available: <https://www.painmanagementnyc.com/es/tecnicas-de-neuroestimulacion/>
- [2] N. Arandia, J. I. Garate, and J. Mabe, “Embedded sensor systems in medical devices: Requisites and challenges ahead,” *Sensors*, vol. 22, no. 24, 2022. [Online]. Available: <https://pmc.ncbi.nlm.nih.gov/articles/PMC9781231/>
- [3] S. Martínez, T. G. Constandinou, F. Veirano, and F. Silveira, “Trends in volumetric-energy efficiency of implantable neurostimulators: A review from a circuits and systems perspective,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 17, no. 1, February 2023.
- [4] I. AspenCore, “Buy or build an rtos: Does it matter for medical devices?” 2009, accessed: 2026-04-16. [Online]. Available: <https://www.embedded.com/buy-or-build-an-rtos-does-it-matter-for-medical-devices/>
- [5] Integer Holdings Corporation. About integer. [Online]. Available: <https://www.integer.net/company/about/default.aspx>
- [6] *nRF Connect SDK Fundamentals. Lesson 1 - nRF Connect SDK structure and content.*, Nordic Semiconductor. [Online]. Available: <https://academy.nordicsemi.com/courses/nrf-connect-sdk-fundamentals/lessons/lesson-1-nrf-connect-sdk-introduction/topic/nrf-connect-sdk-structure-and-content/>
- [7] *About the Zephyr Project*, Linux Foundation. [Online]. Available: <https://www.zephyrproject.org/learn-about/>
- [8] *nRF Connect SDK Intermediate. Lesson 7 - Multithreaded applications.*, Nordic Semiconductor. [Online]. Available: <https://academy.nordicsemi.com/courses/nrf-connect-sdk-fundamentals/lessons/lesson-7-multithreaded-applications/topic/scheduler/>
- [9] *nRF Connect SDK Fundamentals. Lesson 3 - Elements of an nRF Connect SDK application.*, Nordic Semiconductor. [Online]. Available: <https://academy.nordicsemi.com/courses/nrf-connect-sdk-fundamentals/lessons/lesson-1-nrf-connect-sdk-introduction/topic/nrf-connect-sdk-structure-and-content/>

Referencias

- [10] *nRF Connect SDK Fundamentals. Lesson 2 - Reading buttons and controlling LEDs.*, Nordic Semiconductor. [Online]. Available: <https://academy.nordicsemi.com/courses/nrf-connect-sdk-fundamentals/lessons/lesson-2-reading-buttons-and-controlling-leds/topic/devicetree/>
- [11] *nRF Connect SDK Intermediate. Lesson 8 - Sysbuild.*, Nordic Semiconductor. [Online]. Available: <https://academy.nordicsemi.com/courses/nrf-connect-sdk-intermediate/lessons/lesson-8-sysbuild/topic/sysbuild-explained/>
- [12] Fanstel Corporation, *BT40F, BT40, BT40E, BT40N, BT40NE Product Specifications*, 2024, version 1.09. [Online]. Available: https://static1.squarespace.com/static/561459a2e4b0b39f5cefa12e/t/66a40fc5570fa15b6261bb3b/1722027977442/BT40F_BT40N+Product+Specifications.pdf
- [13] *nRF5340 Product Specification v1.6*, Nordic Semiconductor, Febrero 2025. [Online]. Available: https://docs-be.nordicsemi.com/bundle/ps_nrf5340/page/nRF5340_PS_v1.6.pdf?LANG=enus
- [14] NFC Forum. Nfc technology. [Online]. Available: <https://nfc-forum.org/learn/nfc-technology/>
- [15] *MCP4821/MCP4822. 12-Bit DACs with Internal VREF and SPI™ Interface*, Microchip Technology Inc., 2005. [Online]. Available: <https://www.alldatasheet.com/datasheet-pdf/pdf/195326/MICROCHIP/MCP4821.html>
- [16] Nordic Semiconductor, *nRF5340 DK User Guide*, 2024. [Online]. Available: https://docs.nordicsemi.com/bundle/ug_nrf5340_dk/page/UG/dk/intro.html
- [17] ——. Online power profiler for bluetooth le. [Online]. Available: <https://devzone.nordicsemi.com/power/w/opp/2/online-power-profiler-for-bluetooth-le>
- [18] Bluetooth SIG. (2023) Device information service (dis) specification. [Online]. Available: <https://www.bluetooth.com/specifications/specs/dis-1-2/>
- [19] *Developer Study Guide: Bluetooth® Low Energy Security*, Bluetooth, Marzo 2023. [Online]. Available: <https://www.bluetooth.com/bluetooth-resources/le-security-study-guide/>
- [20] UUIDGenerator.net, “Online uuid generator,” 2025, accessed: 2025-06-14. [Online]. Available: <https://www.uuidgenerator.net/>
- [21] C. Clinic, “Vagus nerve,” 2022, accessed: 2026-04-19. [Online]. Available: <https://my.clevelandclinic.org/health/body/22279-vagus-nerve>
- [22] Cleveland Clinic, “Spinal cord,” 2024. [Online]. Available: <https://my.clevelandclinic.org/health/body/21946-spinal-cord>
- [23] —, “Phrenic nerve: Function, anatomy & damage,” 2025. [Online]. Available: <https://my.clevelandclinic.org/health/body/22270-phrenic-nerve>

- [24] Mayo Clinic Staff. (2026) Deep brain stimulation. [Online]. Available: <https://www.mayoclinic.org/tests-procedures/deep-brain-stimulation/about/pac-20384562>
- [25] Qoitech, *Otii Arc Pro User Manual*, accedido: 2026-05-06. [Online]. Available: <https://docs.qoitech.com/en/user-manual/arc#arc-tech-spec>
- [26] U.S. Food and Drug Administration, *Advanced Bionics Precision Spinal Cord Stimulator System: Summary of Safety and Effectiveness Data*, 2004. [Online]. Available: https://www.accessdata.fda.gov/cdrh_docs/pdf3/P030017b.pdf
- [27] Bluetooth SIG, *Bluetooth Core Specification Version 6.2*, 2025. [Online]. Available: <https://www.bluetooth.com/specifications/specs/core-specification/>
- [28] Zephyr Project. (2026) Bluetooth low energy host. [Online]. Available: <https://docs.zephyrproject.org/latest/services/connectivity/bluetooth/bluetooth-le-host.html>
- [29] *Bluetooth Low Energy Fundamentals*, Nordic Semiconductor. [Online]. Available: <https://academy.nordicsemi.com/courses/bluetooth-low-energy-fundamentals/>
- [30] S. Networks, “Ble advertisement channels,” 2023. [Online]. Available: <https://semfionetworks.com/blog/ble-advertisement-channels/>
- [31] NovelBits, “Bluetooth addresses privacy in bluetooth low energy,” 2020. [Online]. Available: <https://novelbits.io/bluetooth-address-privacy-ble/>
- [32] Bluetooth, “Bluetooth® low energy: It starts with advertising,” 2017. [Online]. Available: <https://www.bluetooth.com/blog/bluetooth-low-energy-it-starts-with-advertising/>
- [33] *Near Field Communication (NFC)*, Nordic Semiconductor, 2025. [Online]. Available: <https://docs.nordicsemi.com/bundle/ncs-3.2.0/page/nrf/protocols/nfc/index.html#ug-nfc>

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

4.1.	Tensión de salida del convertidor <i>boost</i> en función de las señales de control.	48
5.1.	Parámetros de <i>advertising</i> y conexión BLE seleccionados.	68
5.2.	Rango de valores posibles y pasos definidos para la amplitud, el ancho de pulso y la frecuencia de estimulación.	74
6.1.	Comparación entre valores nominales, programados y medidos de los parámetros de estimulación para terapia VNS.	99
6.2.	Comparación entre valores nominales, programados y medidos de los parámetros de estimulación para terapia SCS.	100
6.3.	Comparación entre valores nominales, programados y medidos de los parámetros de estimulación para terapia PNS.	101
6.4.	Comparación entre valores nominales, programados y medidos de los parámetros de estimulación para terapia RNS.	102
6.5.	Comparación entre valores programados y medidos de los parámetros de estimulación para la prueba de desempeño en condiciones límite.	103
6.6.	Jitter máximo medido para distintas terapias	106
6.7.	Comparación del consumo entre el SoC, el circuito de estimulación y el total del sistema para una terapia PNS caracterizada por una frecuencia de 40 Hz, un ancho de pulso de 300 μ s y una amplitud de 1.0 mA. La corriente I_{total} corresponde al consumo definido en la Figura 6.19, mientras que I_{SoC} representa el consumo del SoC según la misma figura. El consumo del circuito de estimulación se obtiene como la diferencia $I_{total} - I_{SoC}$, correspondiente al resto del sistema.	118
6.8.	Estimación de autonomía para distintos perfiles de uso, considerando una batería de referencia de 180 mAh y una capacidad útil del 90 %.	120
6.9.	Variación del consumo del SoC en función de la frecuencia de estimulación.	120

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

1.1.	Forma de onda típica de estimulación. Se observa una secuencia periódica de pulsos rectangulares, definidos por su amplitud y ancho de pulso, seguida de una fase de balance activo y otra de balance pasivo. Esta secuencia se repite con una frecuencia $\frac{1}{T}$, siendo T el período.	20
2.1.	Descripción general del sistema	27
2.2.	Diagrama del flujo de operación del sistema incluyendo las etapas de emparejamiento del dispositivo vía NFC para la conexión Bluetooth, la configuración de parámetros por parte del programador mediante una aplicación móvil y ejecución de la terapia por el paciente utilizando el dispositivo previamente configurado.	27
3.1.	Herramientas del <i>nRF Connect SDK</i> y métodos de configuración. <i>Imagen extraída del curso nRF Connect SDK Fundamentals. [6].</i>	34
3.2.	Proceso de compilación de una aplicación con nRF Connect SDK. <i>Imagen extraída del curso nRF Connect SDK Fundamentals. [6].</i>	34
3.3.	Configuración de la pila Bluetooth Low Energy.	36
3.4.	Fragmento del <i>Devicetree</i> de la aplicación que define el nodo correspondiente al electrodo <code>e1_neg</code>	37
3.5.	Configuraciones en <code>sysbuild.conf</code> para incluir una imagen para el <i>Network Core</i> [11].	40
4.1.	Diagrama de bloques del dispositivo. Se presentan los tres bloques principales que conforman el sistema, así como la interacción entre ellos. Los bloques con línea punteada corresponden al diseño original provisto por <i>Integer</i> , mientras que los de línea continua fueron agregados o modificados en el marco del proyecto. En particular, se eliminaron los dos microcontroladores, el <i>reed-switch</i> y la antena RF del diseño original, y se incorporaron el SoC, el DAC, un botón que sustituye el funcionamiento del <i>reed-switch</i> y la antena NFC.	43

Índice de figuras

4.2. Esquema del circuito de estimulación. El sistema es alimentado a partir de la tensión V_{anode} , a la cual se conecta el electrodo configurado como positivo. La corriente circula a través del tejido hacia los electrodos configurados como negativos y luego es conducida por la fuente de corriente operando en modo <i>sink</i> . La tensión V_{sink} corresponde a la caída de tensión en la fuente de corriente. La alimentación V_{anode} puede provenir de V_{bat} o de V_{boost} , dependiendo del nivel de tensión requerido.	44
4.3. Diagrama de bloques simplificado del nRF5340. <i>Imagen extraída de la Especificación del Producto del nRF340</i> [13]	49
4.4. Sistema de control de relojes del SoC. El timer utiliza los relojes PCLK1M y PCLK16M provenientes del HFCLK. <i>Imagen extraída de la Especificación del Producto del nRF340</i> [13]	51
4.5. Diagrama de bloques simplificado del ADC. <i>Imagen extraída de la Especificación del Producto del nRF340</i> [13].	51
4.6. Diagrama y foto del setup utilizado para las pruebas iniciales del firmware y hardware en conjunto.	54
4.7. PCB Diseñado	56
4.8. PCB Diseñado	56
5.1. Diagrama de la estructura en capas del <i>firmware</i> desarrollado.	58
5.2. Diagrama de flujo del funcionamiento general del sistema, mostrando las etapas de inicialización, conexión mediante BLE, validación de parámetros y ejecución de la terapia de estimulación.	59
5.3.	61
5.4. Diagrama de estados de la aplicación.	62
5.5. Diagrama de estados de la comunicación entre el dispositivo y el teléfono celular.	64
5.6. Consumo promedio del <i>advertising</i> BLE en función del intervalo de <i>advertising</i> . Datos extraídos de la herramienta Nordic Online Power Profiler for Bluetooth LE [17].	65
5.7. Consumo de los diferentes componentes de <i>advertising</i> durante un <i>advertising</i> interval. <i>Imagen extraída de la herramienta Nordic Online Power Profiler for Bluetooth LE</i> [17]	66
5.8. Consumo promedio de la conexión BLE en función de la <i>PHY</i> utilizada y del intervalo de conexión. Datos obtenidos a partir de la herramienta de [17].	68
5.9. Consumo de los diferentes componentes de la conexión durante un intervalo de conexión. <i>Imagen extraída de la herramienta Nordic Online Power Profiler for Bluetooth LE</i> [17]	69
5.10. Etapa 1 del proceso de autenticación utilizando OOB.	71
5.11. Etapa 2 del proceso de autenticación	72
5.12. Formato del buffer de la característica <i>StimData</i> para el comando 0x01 de configuración de los parámetros programables de la terapia.	74
5.13. Asignación de bits para cada electrodo y tabla de verdad para el par de bits.	75

5.14. Formato del buffer de la característica <i>StimData</i> para el comando 0x02 de configuración del tipo de usuario.	76
5.15. Formato del buffer de la característica <i>StimData</i> para el comando 0x03 de inicio/detención de la terapia.	77
5.16.	80
5.17. Estructura de datos utilizada para representar un electrodo	83
5.18. Estructura para controlar la tensión de alimentación de la fuente de corriente	85
5.19. Registro para escribir un comando en el DAC MCP4821. <i>Imagen extraída de la hoja de datos del DAC</i> [15].	87
5.20. Registro del DAC que se escribe durante la primera y segunda fase de estimulación. D11-D0 corresponde al valor de <i>data_bytes</i> calculado en la ecuación 5.4.	88
5.21. Diagrama de temporización de la comunicación SPI del DAC, mostrando la estructura de la trama de datos (bits de configuración y datos), así como las señales $\overline{CS}$, SCK, SDI y $\overline{LDAC}$. En la implementación realizada, la señal $\overline{LDAC}$ se encuentra fijada por hardware a nivel bajo, por lo que V_{OUT} se actualiza automáticamente al finalizar cada escritura, luego del flanco ascendente de $\overline{CS}$. <i>Imagen extraída de la hoja de datos del DAC</i> [15].	88
5.22. Circuito equivalente de la entrada a un pin analógico del nRF5340. <i>Imagen extraída de la Especificación del Producto del nRF340 conectado al ADC</i> [13].	90
5.23. Estructura de datos utilizada para deshabilitar un periférico en el <code>.overlay</code>	93
5.24. Estructura utilizada para definir la asignación de una señal en el archivo <code>.overlay</code>	93
5.25. Configuración de <i>drivers</i> de periféricos en <code>prj.conf</code>	94
5.26. Configuración del sistema de relojes.	94
5.27. Configuración de las pilas de NFC y BLE.	95
5.28. Configuración para almacenamiento en memoria persistente.	95
6.1. Forma de onda de estimulación para terapia VNS - Epilepsia, medida con osciloscopio.	99
6.2. Forma de onda de estimulación para terapia SCS medida con osciloscopio.	101
6.3. Forma de onda de estimulación para terapia PNS medida con osciloscopio.	102
6.4. Forma de onda de estimulación para terapia RNS medida con osciloscopio.	103
6.5. Forma de onda de estimulación para pulso con condiciones límite.	104
6.6. Frecuencia observada en el osciloscopio para la configuración de terapia con máxima frecuencia programable.	104
6.7.	105

Índice de figuras

6.8. Captura de tráfico BLE, donde se observa la secuencia de establecimiento de conexión entre el dispositivo central y el periférico NeuroSoC. Inicialmente se registran paquetes de <i>advertising</i> del tipo ADV_IND, junto con el intercambio SCAN_REQ/SCAN_RSP. Posteriormente, el central transmite el paquete CONNECT_IND, que inicia la conexión BLE. En el detalle del paquete #14 se muestran parámetros de enlace relevantes, como el intervalo de conexión, latencia, timeout de supervisión.	107
6.9. Primeros paquetes posteriores al establecimiento de la conexión. Se observan paquetes de control de Link Layer asociados al intercambio de capacidades y parámetros iniciales.	109
6.10. Proceso de descubrimiento GATT, donde el central solicita los servicios primarios mediante Read By Group Type Request y el periférico responde con Read By Group Type Response, identificándose el servicio NSS implementado mediante su UUID de 128 bits.	110
6.11. Captura del paquete de terminación de conexión y su contenido.	110
6.12. Paquete <i>Pairing Request</i> durante el procedimiento de emparejamiento BLE. Se observa el indicador OOB activado, asociado al uso de NFC como canal fuera de banda.	111
6.13. Activación del cifrado de la conexión BLE. Luego de los paquetes LL_ENC_REQ, LL_ENC_RSP y LL_START_ENC_REQ, los datos posteriores se transmiten cifrados.	112
6.14. Captura del paquete de terminación de conexión y su contenido.	112
6.15. Tensión en V_{Boost} al estimular con una amplitud de corriente de 10 mA sobre una resistencia variable. Al comienzo de la terapia la tensión se fija en 15 V. Al realizar 3 medidas altas consecutivas de la tensión en V_{sink} se ajusta la tensión V_{Boost} , hasta alcanzar el valor en el que se estabiliza. Luego, al aumentar la resistencia a 1100 Ω , la tensión de V_{Boost} se ajusta hasta alcanzar nuevamente el valor de 15 V. Al disminuir la resistencia a 800 Ω , la tensión vuelve a ajustarse al obtener 3 medidas altas consecutivas y se estabiliza en 10 V.	113
6.16. Comparación del layout de la etapa de realimentación en ambas implementaciones. En la placa de test proporcionada, la distancia entre el nodo de salida y el pin FB es aproximadamente 0.5 cm, mientras que en la PCB desarrollada dicha distancia es de aproximadamente 1.8 cm.	114
6.17. Frecuencia del ruido observado durante un pulso de estimulación al utilizar el <i>boost converter</i> de la PCB diseñada.	115
6.18. Comparación del ruido observado en el pulso de estimulación al utilizar el <i>boost</i> de la <i>test board</i> provista y la PCB diseñada, con ruidos de amplitud 227mV y 10mV respectivamente.	116
6.19. Esquema de alimentación del PCB utilizado. Las mediciones se realizaron tanto sobre el nodo <i>BATTERY</i> , correspondiente al consumo total de la placa, como sobre el nodo <i>VDD_UC</i> , asociado a la alimentación del SoC. Para esta última medición se retiró la resistencia R1, permitiendo aislar el consumo del microcontrolador respecto del resto del circuito.	117

6.20. Captura del ciclo de consumo del sistema obtenida mediante Otii Arc. Se distinguen las etapas de <i>advertising</i> , activación del módulo NFC, inicio de conexión BLE y conexión establecida.	118
A.1. BLE Protocol Stack Application <i>Imagen extraída del curso Bluetooth Low Energy Fundamentals de Nordic Semiconductor</i> [29].	2
A.2. Canales de Advertisement de BLE. <i>Imagen extraída de BLE Advertisement Channels</i> [30]	6
A.3. Intervalos de escaneo y de advertising. <i>Imagen extraída del curso Bluetooth Low Energy Fundamentals de Nordic Semiconductor</i> [29]	6
A.4. Clasificación de direcciones BLE. <i>Imagen extraída de Bluetooth Addresses & Privacy in Bluetooth Low Energy</i> [31]	7
A.5. Formato de una dirección random private resolvable (en formato little-endian)	8
A.6. Formato de un paquete de advertising. <i>Imagen extraída de Bluetooth® Low Energy: It starts with advertising</i> [32]	9
A.7. PDU del paquete de advertising. <i>Imagen extraída de Bluetooth® Low Energy: It starts with advertising</i> [32]	10
A.8. Formato del <i>payload</i> del PDU del paquete de <i>advertising</i> . <i>Imagen extraída de Bluetooth Low Energy Fundamentals</i> [29]	11
A.9. Comunicación BLE durante etapa de conexión. <i>Imagen extraída de Bluetooth Low Energy Fundamentals</i> [29]	12
A.10. Bloques de datos de un atributo. <i>Imagen extraída de Bluetooth Low Energy Fundamentals</i> [29]	15
A.11. Formato de un atributo de servicio <i>Imagen extraída de Bluetooth Low Energy Fundamentals</i> [29]	15
A.12. Conjunto de atributos que forman una característica. <i>Imagen extraída de Bluetooth Low Energy Fundamentals</i> [29]	16
A.13. Formato de un servicio conformado por una característica. <i>Imagen extraída de Bluetooth Low Energy Fundamentals</i> [29]	17
A.14. Formato de los mensajes NDEF. <i>Imagen extraída del manual de nRF Connect SDK</i> [33]	21

Esta es la última página.
Compilado el martes 7 julio, 2026.
<http://iie.fing.edu.uy/>