Sergi Arjona Martí

DESENVOLUPAMENT D'UN BOOTLOADER PER AL COMPONENT ESP32

TREBALL DE FI DE GRAU

dirigit pel Dr. Pere Millán Marco

Grau d'Enginyeria Informàtica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2022

“De tot cor, vull expressar el meu agraïment a les següents persones:

A Albert Escala i Joaquín López, de Generation RFID S.L., per brindar-me l’oportunitat de realitzar aquest projecte.

Al Dr. Pere Millán, pels 6 anys que l’he tingut com a professor, per dirigir aquest projecte i orientar-me durant tot el procés.

Als meus companys i amics, per acompanyar-me i fer més curta aquesta llarga etapa.

Als meus pares, a la meva germana i a la meva parella, per tenir tanta paciència amb mi, pel seu suport incondicional i per no deixar-me caure en els pitjors moments. És gràcies a ells que avui sóc aquí.

A tothom qui no va deixar de creure en mi, per donar-me forces.

A tothom qui ho va deixar de fer, per fer-me fort.

I, per últim, a mi mateix, per no rendir-me.”

NOTA DE L'AUTOR

Aquest document és una versió adaptada de l'original per tal de satisfer el requisits de l'acord de confidencialitat establert entre l'empresa Generation RFID S.L., l'Escola Tècnica Superior d'Enginyeria (URV) i jo mateix, Sergi Arjona Martí, com autor d'aquest Treball de Fi de Grau.

En conseqüència, algunes parts del document original han sigut omeses o modificades. Per a més informació, si us plau contacti amb mi per correu electrònic o a través de LinkedIn.

NOTA DEL AUTOR

Este documento es una versión adaptada del original a fin de poder satisfacer los requisitos del acuerdo de confidencialidad establecido entre la empresa Generation RFID S.L., la Escola Tècnica Superior d'Enginyeria (URV) y yo mismo, Sergi Arjona Martí, como autor de este Trabajo de Fin de Grado.

En consecuencia, algunas partes del documento original han sido omitidas o modificadas. Para más información, por favor contacte conmigo por correo electrónico o a través de LinkedIn.

AUTHOR'S NOTE

This document is an adapted version of the original one to meet the requirements from the confidentiality agreement established between the enterprise Generation RFID S.L., the Escola Tècnica Superior d'Enginyeria (URV) and myself, Sergi Arjona Martí, as the author of this Final Degree Project.

Consequently, some parts of the original document have been omitted or modified. For more information, please contact me by email or via LinkedIn.

Resum.

Com a conseqüència de la crisi de subministrament de xips provocada per la pandèmia de la COVID-19, la producció d'un dispositiu mòbil de baix consum, de la mà de Generation RFID S.L., es veu completament paralitzada, obligant a l'empresa a buscar una solució alternativa. En aquest punt, l'empresa aprofita per introduir millores al dispositiu, entre les quals s'incorpora un component ESP32 que dota a l'aparell de connectivitat sense fil.

Gràcies a l'anterior, s'estableix l'objectiu de facilitar el desplegament de programari a distància. Per fer-ho possible, es proposa el desenvolupament d'un *bootloader* que permeti l'actualització remota del dispositiu a través de la tecnologia *Bluetooth Low Energy*.

Així doncs, atret per la idea del projecte, es produeix un acord entre autor i empresa, que resulta en la realització d'aquest Treball de Fi de Grau.

Resumen.

Como consecuencia de la crisis de suministro de chips provocada por la pandemia del COVID-19, la producción de un dispositivo móvil de bajo consumo, de la mano de Generation RFID S.L., se ve completamente paralizada, obligando a la empresa a buscar una solución alternativa. En este punto, la empresa aprovecha para introducir mejoras al dispositivo, entre las cuales se incorpora un componente ESP32 que dota al aparato de conectividad inalámbrica.

Gracias a lo anterior, se establece el objetivo de facilitar el despliegue de software a distancia. Para hacerlo posible, se propone el desarrollo de un *bootloader* que permita la actualización remota del dispositivo a través de la tecnología *Bluetooth Low Energy*.

Así pues, atraído por la idea del proyecto, se produce un acuerdo entre autor y empresa, que resulta en la realización de este Trabajo de Fin de Grado.

Abstract.

Because of the chip supply shortage caused by the COVID-19 pandemic, the production of a low consumption mobile device, by the hands of Generation RFID S.L., becomes completely paralyzed, forcing the enterprise to search for an alternative solution. At this point, the enterprise leverages to add some enhancements to the device, among which an ESP32 component is incorporated, providing the apparatus with wireless connectivity.

As a result, it is established the goal to ease the remote software deployment. To make it possible, it is proposed the development of a bootloader that allows to update the device remotely via the Bluetooth Low Energy technology.

So, attracted by the idea of the project, an agreement takes place between author and enterprise, which results in the development of this Final Degree Project.

Índex

Índex de taules

Índex de figures

Índex de fórmules

1 Introducció

1.1 Presentació de l'empresa

Generation RFID S.L. [3] és una empresa tecnològica situada a Reus que ofereix serveis i productes relacionats amb el desenvolupament i la validació de components electrònics encastats.

Figura 1. Logo de l'empresa Generation RFID S.L.

Especialitzada en el *testing*, i amb més d'onze anys d'experiència, Generation RFID desenvolupa la seva activitat principalment en els sectors de l'automoció i de l'energia, duent a terme projectes de tecnologia punta en col·laboració amb empreses OEM[1] i Tier-1[2] com LEAR Corporation, Jaguar, Volvo, BMW, Land Rover, Volkswagen Group o Hipra, entre d'altres.

1.2 Context previ al projecte

El projecte s'origina com a conseqüència directa de la crisi de xips ocasionada per la pandèmia de la COVID-19. Degut a la impossibilitat de seguir amb la producció, Generation RFID fa una nova versió d'un producte ja existent, redissenyant el PCB[3] i substituint l'anterior microcontrolador per un mòdul ESP32 [3], amb connectivitat Wi-Fi i Bluetooth integrada en el propi xip.

1.3 Descripció i objectius del projecte

Aquest projecte consisteix en el disseny i el desenvolupament d'un bootloader que permeti actualitzar el firmware d'un dispositiu que integra un component ESP32.

Aprofitant la connectivitat Bluetooth que ofereix el component ESP32, l'objectiu principal del bootloader és poder desplegar noves versions de firmware i que tots els dispositius existents es puguin actualitzar autònomament.

---

1 Un fabricant d'equipament original o OEM, de l'anglès *Original Equipment Manufacturer* és una empresa que manufactura productes que són comprats per una segona empresa i venuts sota la marca d'aquesta.

2 Les empreses Tier-1 són aquelles que subministren directament a una empresa OEM i, per tant, són les més importants de la cadena de subministrament.

3 Una placa de circuit imprès o PCB, de l'anglès *Printed Circuit Board*, consta de làmines de material no conductor sobre una làmina de material conductor que permet comunicar diferents components electrònics sobre una mateixa superfície.

Concretament, l'objectiu d'aquesta solució és satisfer les necessitats de Generation RFID. Seguint el procés d'enginyeria del software, aquestes necessitats venen especificades en forma de requisits (veure Secció 3.1).

D'altra banda, seguidament s'exposen els motius que m'han portat a formar part d'aquest projecte.

1.3.1 Objectius personals

Per un costat, busco retrobar-me amb un estudiant que durant el seu transcurs a la URV ha anat perdent l'interès per l'electrònica i el món *embedded*: jo mateix. Amb el pas del temps em vaig decantar cap al desenvolupament web fins aquest mateix any, en què he recuperat la motivació amb la que vaig començar.

Altrament, també busco guanyar confiança en l'àmbit laboral, on el fet de formar part d'un projecte real que tindrà una repercussió directa dins de la indústria, juga un paper clau de cara al meu desenvolupament professional.

En definitiva, tinc el repte personal de posar-me a prova amb el projecte més important que he fet fins el moment i intentar deixar la meva marca personal en aquest.

1.3.2 Objectius professionals

Canviant el focus a l'àmbit acadèmic, aquest projecte és literalment l'últim pas de la meva primera etapa acadèmica a la URV, amb el qual pretenc consolidar els coneixements que he adquirit durant els últims anys i donar lloc a la meva entrada al món laboral dins del sector.

Així mateix, també vull que aquest treball sigui indicatiu de l'experiència que he anat adquirint i que em permeti culminar els estudis amb un expedient acadèmic competent.

2 Marc teòric

2.1 Què és un bootloader?

Un bootloader és un programa encarregat de gestionar l'arrencada d'un dispositiu i carregar a la memòria interna el sistema operatiu o l'aplicació a executar. Com a tal, el bootloader és el primer que s'executa quan s'engega el dispositiu i la seva funció es divideix en dues etapes, que sovint se solen denominar *1st stage* o *2nd stage bootloader*, és a dir, de primera o segona etapa.

El bootloader de primera etapa es localitza a la memòria ROM i, degut als seus recursos limitats, només sol tenir dos propòsits: inicialitzar el hardware i carregar el bootloader de segona etapa a la memòria RAM.

Un cop carregat, es comença a executar el codi del bootloader de segona etapa. En dispositius mòbils, és comú que n'hi hagi dos modes de funcionament que depenen d'un paràmetre d'entrada.

Generalment, el bootloader se sol iniciar en mode normal i, després de fer les comprovacions corresponents, cedeix l'execució a la partició del disc on es troba l'aplicació principal. En cas contrari, el bootloader s'inicia en mode alternatiu i executa codi personalitzat abans de donar pas al mode normal. En dispositius mòbils, és comú que aquest codi consisteixi en actualitzar el dispositiu (veure Secció 2.3).

En el següent diagrama es pot observar el comportament típic d'un bootloader:

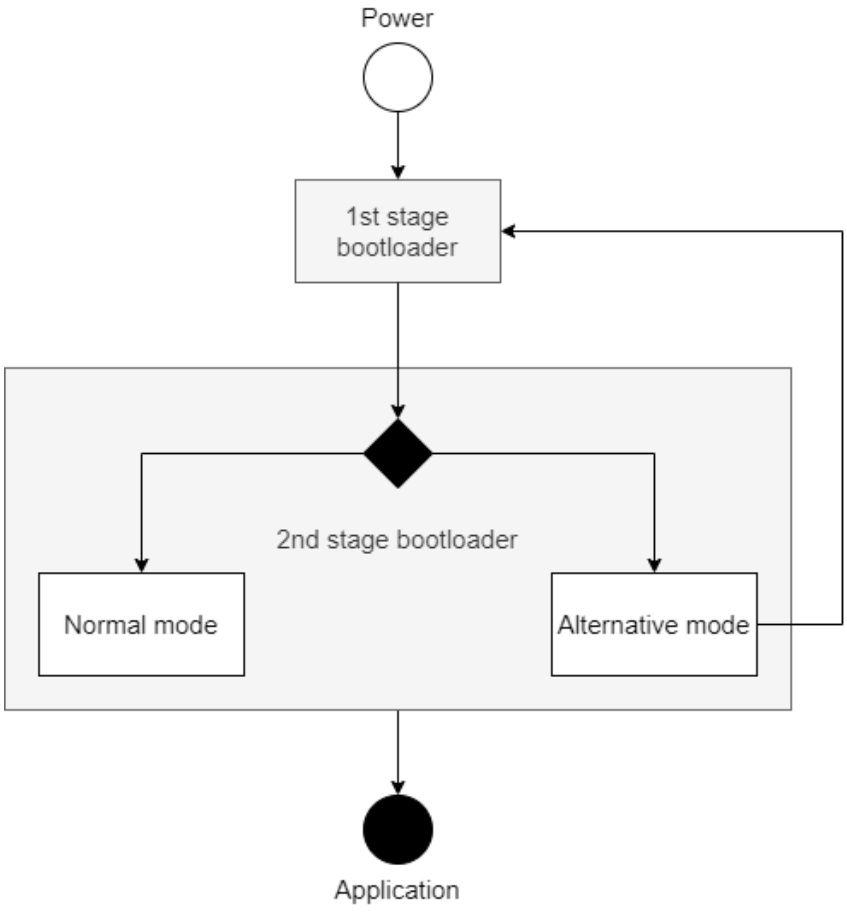

Figura 2. Diagrama de funcionament d'un bootloader de dues etapes

2.2 La tecnologia BLE

El Bluetooth de baixa energia o BLE, de l'anglès *Bluetooth Low Energy*, és una tecnologia de xarxa sense fil que neix l'any 2010 com un subconjunt de l'estàndard Bluetooth v4.0. En comparació amb el Bluetooth clàssic, BLE està enfocat a dispositius que necessiten tenir un consum energètic molt baix i que normalment s'alimenten amb piles o bateries.

Els tres principals blocs dins de l'arquitectura d'un dispositiu BLE [4] són tres: l'aplicació, l'amfitrió o *host* i el controlador.

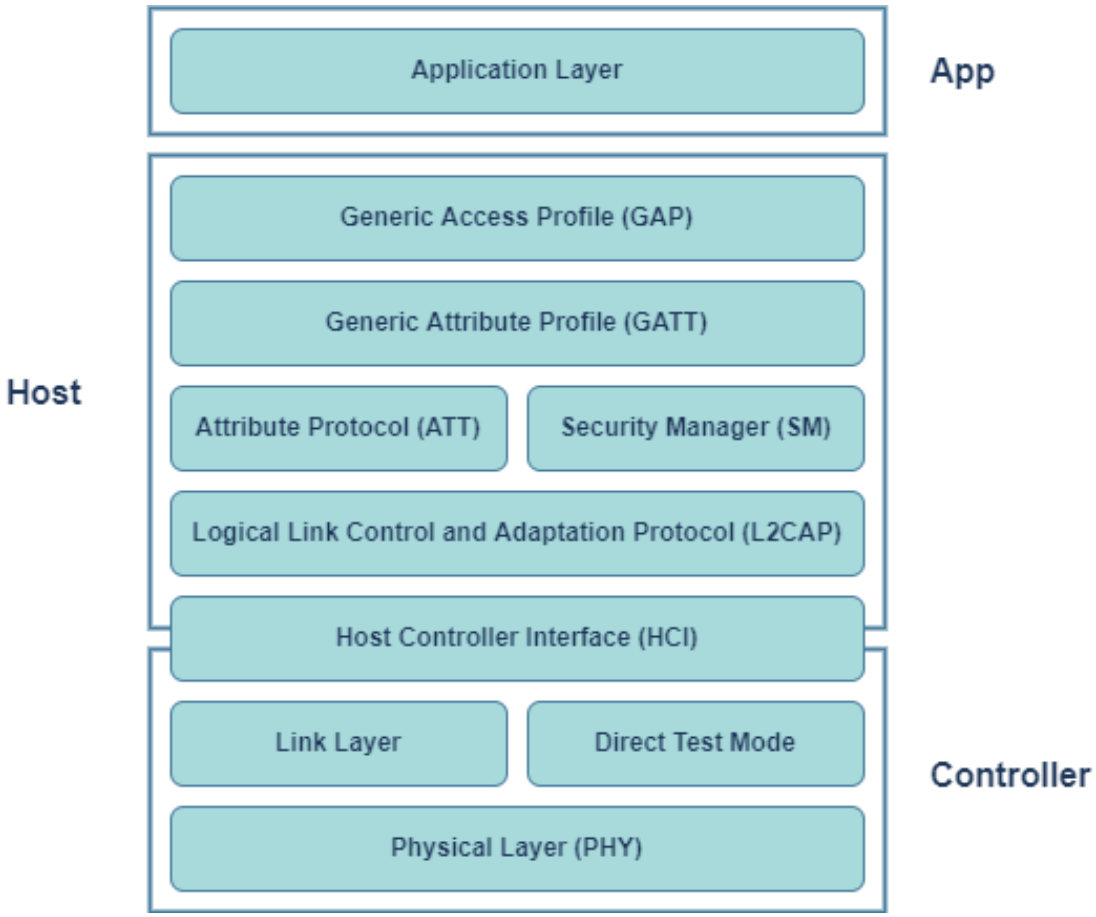

Figura 3. Arquitectura típica d'un dispositiu BLE

El bloc d'aplicació depèn directament de la implementació de les tres capes superiors de l'amfitrió: el perfil d'accés genèric (GAP), que defineix la comunicació i les connexions entre els dispositius, el perfil d'atributs genèrics (GATT), que s'encarrega d'estructurar la informació que mostra el dispositiu, i el protocol d'atributs (ATT), que especifica el rol de cada dispositiu.

Aquestes capes són les més importants i les que tindran el focus del present treball, ja que són responsables de l'enviament i la recepció de dades de l'aplicació.

Especialment, GATT juga un paper clau dins de l'especificació BLE [5] ja que totes les dades rellevants per a les aplicacions i els usuaris s'hauran de formatar, empaquetar i enviar segons les seves regles. Així doncs, GATT estructura les dades jeràrquicament mitjançant els conceptes d'atributs, característiques i serveis. Un atribut és una abstracció de

les dades d'aplicació que poden ser identificades de forma única. Un conjunt d'atributs forma una característica, que conté altres propietats com els permisos i les regles d'interacció per a un conjunt únic de dades. Per últim, un servei agrupa característiques que tenen propietats en comú.

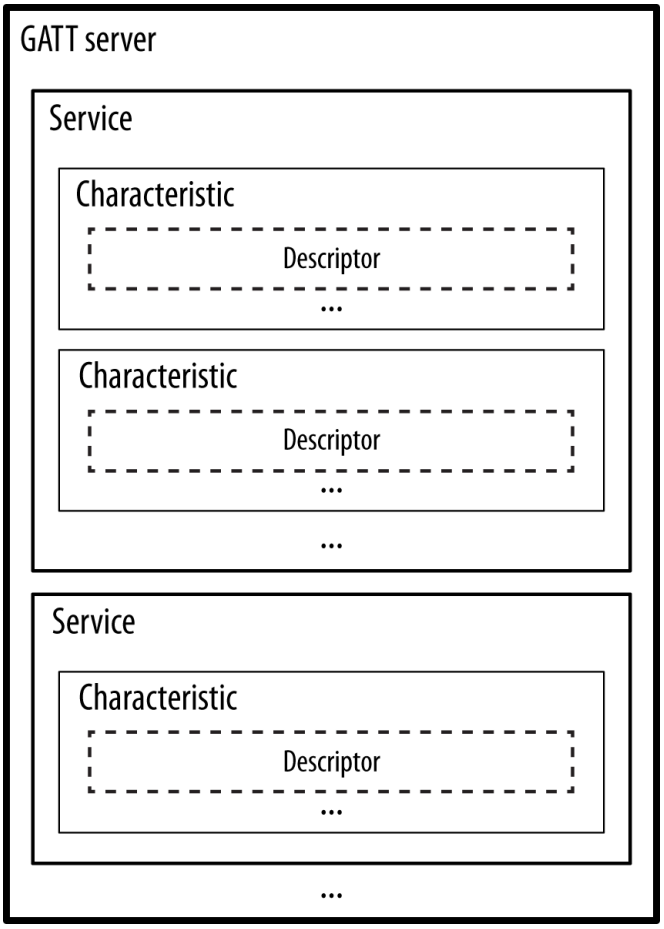

Figura 4. Diagrama simplificat de l'estructura i jerarquia d'un perfil GATT de BLE

En aquest cas, la configuració de les capes anteriors es converteix en un procés simplificat gràcies a l'API[4] que proporciona Espressif Systems[5] per al component ESP32 [6].

A més a més, cal tenir en compte que la versió de BLE sobre la qual es treballa és Bluetooth LE v5.0. Tot i així, gràcies a la retrocompatibilitat de Bluetooth, el component ESP32 també admet connexions amb versions anteriors, assumint els seus desavantatges.

4 Una interfície de programació d'aplicacions o API, de l'anglès Application Programming Interface, és un conjunt de definicions i protocols emprats per dissenyar i integrar el software de les aplicacions.

5 Espressif Systems és l'empresa creadora de la sèrie de xips ESP32, entre d'altres.

2.3 Les actualitzacions OTA

Les actualitzacions a través de l'aire o OTA, de l'anglès *over-the-air*, són un mecanisme per actualitzar el firmware de dispositius mòbils de manera remota i sense fils. Gràcies a aquesta tecnologia, els dispositius es poden actualitzar mentre executen el seu firmware normal, normalment a través de Wi-Fi o Bluetooth.

Les actualitzacions OTA comporten múltiples beneficis, com la millora contínua dels dispositius, la reducció de costos, la facilitat per arreglar vulnerabilitats de seguretat, la recuperació de versions anteriors, l'increment de l'escalabilitat, etc.

En el cas d'aquest projecte, l'actualització del firmware es realitza a través de la tecnologia BLE i es pretén aconseguir els anteriors beneficis.

2.4 El component ESP32

Els ESP32 són una família de xips de baix cost fabricats per Espressif Systems que integren connectivitat Wi-Fi i Bluetooth en un mateix SoC[6] amb un consum molt baix.

Concretament, la variant que integra el dispositiu es tracta d'un ESP32-WROOM-32 [7]. Aquest microcontrolador, juntament amb la resta de la seva família, ha guanyat popularitat en els últims anys degut a la seva àmplia varietat d'aplicacions, des de xarxes de sensors de baixa potència, fins a tasques més demandants com la codificació de veu, transmissió de música o descodificació MP3.

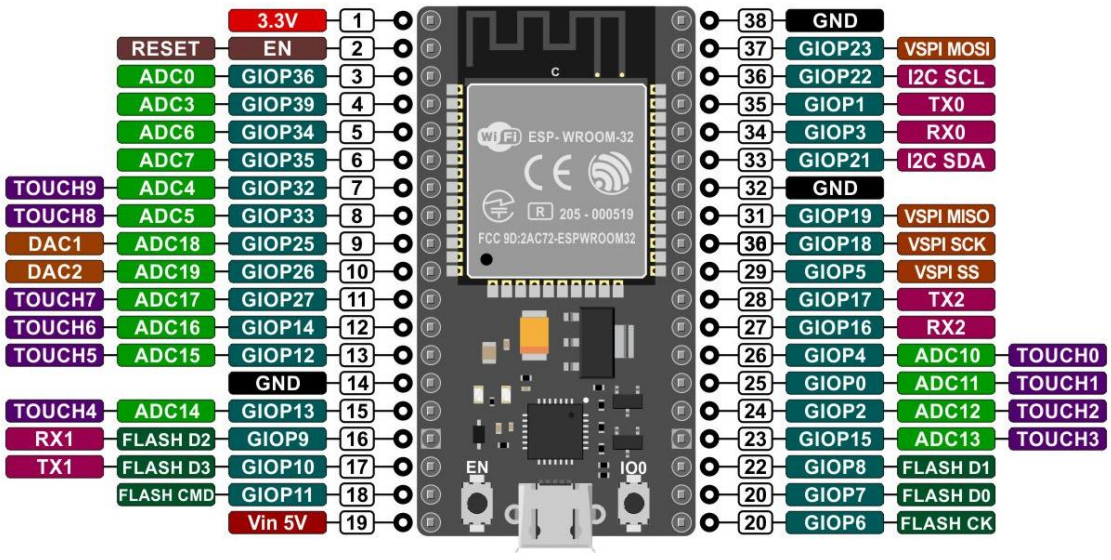

Figura 5. Diagrama de pins del mòdul ESP32-WROOM-32

6 Un sistema en xip o SoC, de l'anglès *system-on-a-chip*, fa referència al fet que tots els components d'un computador es troben integrats en un mateix xip.

D'entre les seves característiques, destaca el seu microprocessador Tensilica Xtensa LX6 de 32 bits [8], amb dos nuclis que es poden controlar individualment, amb una freqüència de rellotge ajustable entre 80 i 240 MHz.

Cadascun d'ells té un espai d'adreces de 4 GB (32 bits) i aquests són simètrics entre ells. El bus de dades i el bus d'instruccions són *little-endian* [9].

Les adreces inferiors a 0x4000_0000 s'accedeixen mitjançant el bus de dades. Les adreces entre 0x4000_0000 i 0x4FFF_FFFF s'accedeixen mitjançant el bus d'instruccions. Si més no, les adreces superiors a 0x5000_0000 s'accedeixen mitjançant els dos busos.

La memòria flash del component ESP32 té una mida de 4 MB i està mapejada a l'espai d'adreces. En concret, aquesta ocupa des de l'adreça 0x3F400000 fins a la 0x3F70000 i és accessible per ambdós microprocessadors a través del bus de dades.

A més a més, el component ESP32 integra diversos perifèrics com sensors tàctils capacitatius, Ethernet, SPI d'alta velocitat, UART, I2S, I2C, etc. De manera anàloga a la memòria flash, aquests també estan mapejats a l'espai d'adreces i, per tant, són accessibles a través del bus de dades.

El sistema operatiu del ESP32 és FreeRTOS amb lwIP[7], que admet actualitzacions OTA de manera que els usuaris puguin actualitzar els seus productes fàcilment fins i tot després del seu llançament.

2.5 L'entorn de treball ESP-IDF

L'ESP-IDF, de l'anglès *ESPressif IoT Development Framework*, és l'entorn de treball oficial per al component ESP32. Gràcies a aquest, els processos de configuració del dispositiu, la compilació del codi i la gravació d'aquest es simplifiquen de la següent manera.

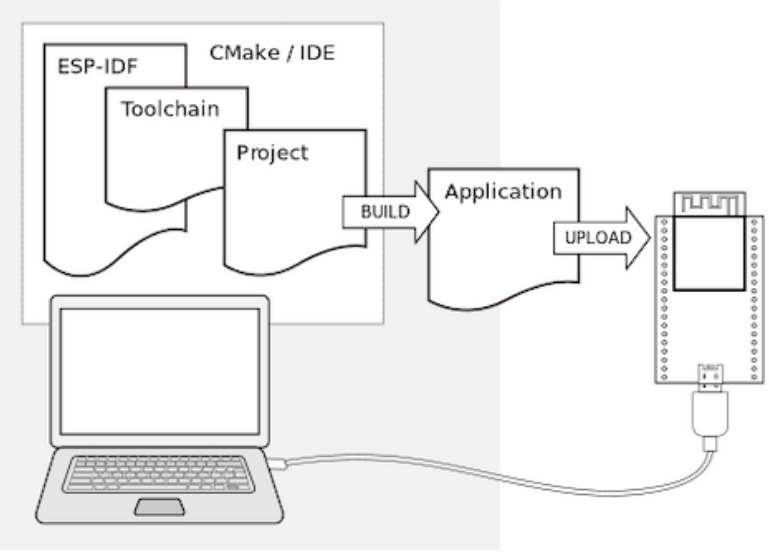

Figura 6. Diagrama de funcionament de l'entorn de treball ESP-IDF

[7] Lightweight IP o lwIP és una pila TCP/IP alternativa de codi obert, compatible amb FreeRTOS.

3 Requisits

3.1 Recollida de requisits

La recollida de requisits s'inicia realitzant diverses reunions telemàtiques amb el client, amb la finalitat d'entendre les seves necessitats i poder extreure les regles de negoci corresponents.

Després de discutir els detalls amb l'equip, es procedeix amb l'especificació dels requisits del sistema, dels quals els següents tenen relació amb el desenvolupament del bootloader.

Requisit	Descripció
#SYS_007	El dispositiu s'ha de mantenir encès mentre hi hagi una connexió de BLE activa. En cas contrari, s'ha d'apagar als 90 segons.
#SYS_008	El LED d'estat ha de parpellejar quan un hi hagi una connexió de BLE activa. En cas contrari, aquest s'ha de mantenir encès.
#SYS_078	El dispositiu ha de tenir servei de BLE per a rebre actualitzacions OTA.
#SYS_079	El dispositiu ha de tenir dues particions principals que s'alternen quan es realitza una actualització OTA.
#SYS_080	Les particions OTA han de tenir espai suficient per contenir una imatge del firmware (aproximadament 950 KB) amb un marge d'uns 100 KB davant de futurs desenvolupaments.
#SYS_081	L'aplicació ha de ser capaç de tractar amb possibles desconnexions durant el procés d'actualització.
#SYS_082	El bootloader només pot gravar imatges validades.
#SYS_083	El bootloader ha de facilitar la seva traçabilitat mitjançant registres o *logs*.
#SYS_084	El bootloader ha de processar les trames nul·les (longitud zero) com a final d'enviament d'una imatge.
#SYS_101	El dispositiu s'ha d'actualitzar mitjançant esdeveniments d'escriptura amb resposta de BLE.

Taula 1. Requisits del sistema

3.2 Casos d'ús

Donat que el focus del treball present és precisament el bootloader, a continuació es procedeix a detallar els casos d'ús que afecten exclusivament a aquest. Així doncs, queda exclosa tota funcionalitat que no tingui relació directa amb el bootloader.

3.2.1 Actors

L'únic actor present en aquest sistema és l'**Usuari**. Normalment, aquest serà un client que desitja actualitzar el firmware del dispositiu. En el seu defecte, aquest també pot ser qualsevol dispositiu que simuli l'enviament d'una imatge vàlida a través de BLE.

En conseqüència, qualsevol acció duta a terme per l'Usuari és processada pel sistema en forma d'esdeveniments, i d'ara en endavant s'interpretaran com a tal.

3.2.2 Diagrama de casos d'ús

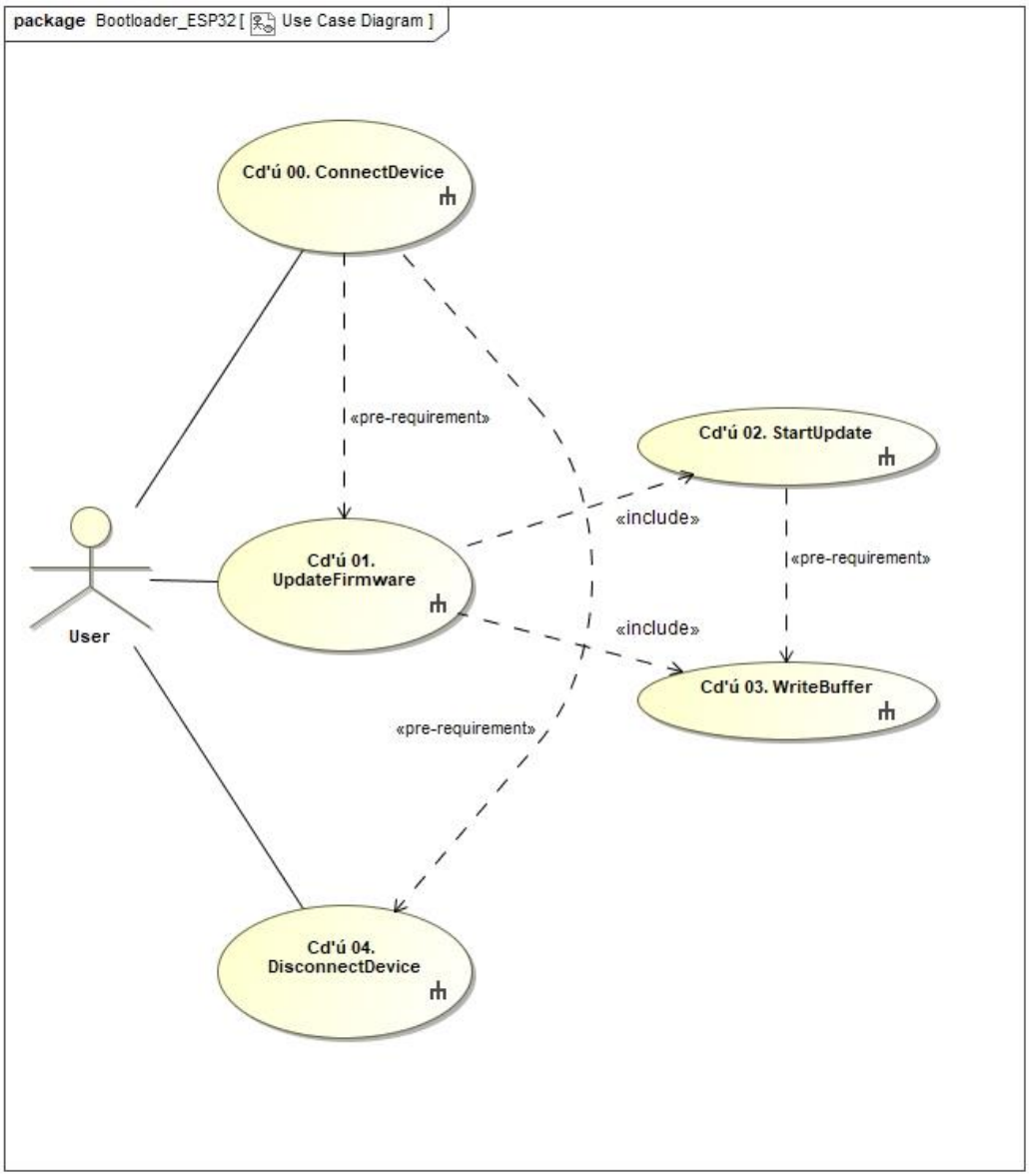

Figura 7. Diagrama de casos d'ús

3.2.3 Especificació textual dels casos d'ús

3.2.3.1 Cd'ú 00. ConnectDevice

Resum de la funcionalitat: estableix una connexió amb un dispositiu BLE.

Paràmetres d'entrada: adreça MAC del dispositiu.

Paràmetres de sortida: cap.

Actors: **Usuari**.

Precondició: no hi ha cap altre dispositiu connectat i el sistema s'està anunciant.

Postcondició: s'ha establert una connexió de BLE correctament.

Procés normal principal:

1. L'usuari envia un esdeveniment de connexió.
2. El sistema rep un esdeveniment de connexió.
3. El sistema mostra un missatge per consola informant que s'ha iniciat una connexió amb el dispositiu.
4. El sistema actualitza els paràmetres de la connexió.
5. El sistema inicia el parpelleig del LED d'estat.
6. El sistema activa una variable booleana de connexió activa.
7. El sistema dona per finalitzat l'esdeveniment de connexió.

Alternatives de procés i excepcions: cap.

3.2.3.2 Cd'ú 01. UpdateFirmware

Resum de la funcionalitat: actualitza el firmware del sistema a través de BLE.

Paràmetres d'entrada: cap.

Paràmetres de sortida: cap.

Actors: **Usuari**.

Precondició: s'ha establert una connexió amb un dispositiu BLE.

Postcondició: s'ha actualitzat el firmware del sistema.

Procés normal principal:

1. L'usuari envia un esdeveniment d'escriptura a la característica B del servei de transferència d'objectes (veure Secció 4.2).
2. El sistema executa el Cd'ú 02. StartUpdate.
3. L'usuari envia un esdeveniment d'escriptura a la característica A del servei de transferència d'objectes.
4. El sistema executa el Cd'ú 03. WriteBuffer.
5. El sistema torna al pas 3.
6. El sistema valida la imatge del firmware.

7. El sistema selecciona la partició d'actualització com a pròxima partició d'arrencada.

8. El sistema es reinicia.

Alternatives de procés i excepcions:

5a. S'ha completat l'enviament de la imatge del nou firmware:

5a1. El sistema salta al pas 6.

6a. La imatge no és vàlida:

6a1. El sistema mostra un missatge d'error per consola.

6a2. El sistema salta al pas 8.

3.2.3.3 Cd'ú 02. StartUpdate

Resum de la funcionalitat: inicia el procés d'actualització del firmware del sistema.

Paràmetres d'entrada: mida en bytes de la nova versió del firmware.

Paràmetres de sortida: cap.

Actors: cap**.**

Precondició: s'ha escrit la característica B del servei de transferència d'objectes.

Postcondició: s'ha iniciat el procés d'actualització.

Procés normal principal:

1. El sistema inicialitza els paràmetres d'actualització.

2. El sistema mostra un missatge per consola informant de la versió actual del firmware.

3. El sistema comprova l'estat de les particions.

4. El sistema selecciona la partició d'actualització.

Alternatives de procés i excepcions:

3a. S'ha corromput alguna de les particions:

3a1. El sistema mostra un missatge d'error per consola.

3a2. El sistema es reinicia.

3.2.3.4 Cd'ú 03. WriteBuffer

Resum de la funcionalitat: acumula missatges de BLE a un buffer i els grava al disc.

Paràmetres d'entrada: missatge de BLE.

Paràmetres de sortida: cap.

Actors: cap**.**

Precondició: s'ha escrit la característica A del servei de transferència d'objectes.

Postcondició: s'ha gravat el buffer a la memòria flash.

Procés normal principal:

1. El sistema rep un missatge de BLE.
2. El sistema afegeix el missatge al buffer.
3. El sistema torna al pas 2.
4. El sistema grava el buffer a la memòria flash.
5. El sistema mostra un missatge per consola informant del procés de gravació.

Alternatives de procés i excepcions:

3a. El buffer està ple:

3a1. El sistema salta al pas 4.

3b. El missatge està buit:

3b1. El sistema salta al pas 4.

4a. És la primera gravació del buffer:

4a1. El sistema llegeix i processa les capçaleres de la imatge.

4a2. El sistema mostra un missatge per consola informant de la versió del nou firmware.

4a3. El sistema inicialitza la nova partició OTA.

4a3a. El sistema falla en la inicialització:

4a3a1. El sistema mostra un missatge d'advertència per consola.

4a3a2. El sistema es reinicia.

4a4. El sistema torna al pas 4.

4b. El sistema falla durant el procés de gravació:

4b1. El sistema mostra un missatge d'error per consola.

4b2. El sistema es reinicia.

3.2.3.5 Cd'ú 04. DisconnectDevice

Resum de la funcionalitat: finalitza una connexió amb un dispositiu BLE.

Paràmetres d'entrada: cap.

Paràmetres de sortida: cap.

Actors: **Usuari**.

Precondició: hi ha una connexió de BLE activa.

Postcondició: s'ha finalitzat la connexió de BLE correctament.

Procés normal principal:

1. L'usuari envia un esdeveniment de desconnexió o es perd la connexió.
2. El sistema rep un esdeveniment de desconnexió.

3. El sistema mostra un missatge per consola informant que s'ha finalitzat la connexió amb el dispositiu.

4. El sistema comença a anunciar-se.

5. El sistema para el parpelleig del LED d'estat.

6. El sistema restableix la variable booleana de connexió.

7. El sistema dona per finalitzat l'esdeveniment de desconnexió.

Alternatives de procés i excepcions: cap.

3.3 Anàlisi dels requisits

3.3.1 Diagrama de classes

El següent diagrama de classes té l'única finalitat d'oferir una representació gràfica de les dades que es tracten al sistema, situant-se en un nivell d'abstracció més alt que el que es correspon al propi llenguatge. Així doncs, aquestes dades es mostren de manera simplificada i encapsulada, ja que el seu tractament a baix nivell s'exposa en seccions posteriors a la present (veure Secció 5 i Secció 6).

Per tant, tenint en compte l'anterior, es procedeix amb la classificació i explicació de les classes definides en funció de les seves característiques.

D'entre les classes definides es poden diferenciar tres tipus, que es defineixen com segueix:

Tipus	Definició
A	Encapsula dades corresponents a un component software principal, és a dir, que té una relació directa amb el procés d'actualització del firmware del sistema.
B	Encapsula dades corresponents a un component software secundari, és a dir, que té una relació indirecta amb el procés d'actualització del firmware del sistema.
C	Encapsula dades que s'utilitzen en un component software, principal o secundari, però que no en representen un en sí i que, a més a més, ja són encapsulades per l'API d'Espressif Systems.

Taula 2. Definició de tipus de classes

Així doncs, considerant els tres tipus definits anteriorment, les classes es classifiquen de la següent forma:

Nº	Nom de la classe	Tipus	Definició
1	Bootloader	A	Conté dades necessàries per a la gestió interna del procés d'actualització del firmware del sistema.
2	Bluetooth	A	Conté dades necessàries per a la gestió interna i externa del procés d'actualització del firmware del sistema.
3	*GATTSParams*	C	Generalització de les classes ConnectEvent, DisconnectEvent i WriteEvent. Conté dades necessàries per a la gestió de la comunicació BLE a través d'esdeveniments.
4	Power	B	Conté dades necessàries per la gestió de l'encès i l'apagat del dispositiu.
5	LED	B	Conté dades necessàries per a la gestió del LED d'estat del dispositiu.
6	ConnectEvent	C	Conté dades d'un esdeveniment de connexió de BLE.
7	DisconnectEvent	C	Conté dades d'un esdeveniment de desconnexió de BLE.
8	WriteEvent	C	Conté dades d'un esdeveniment d'escriptura de BLE.

Taula 3. Classificació de les classes en funció del tipus

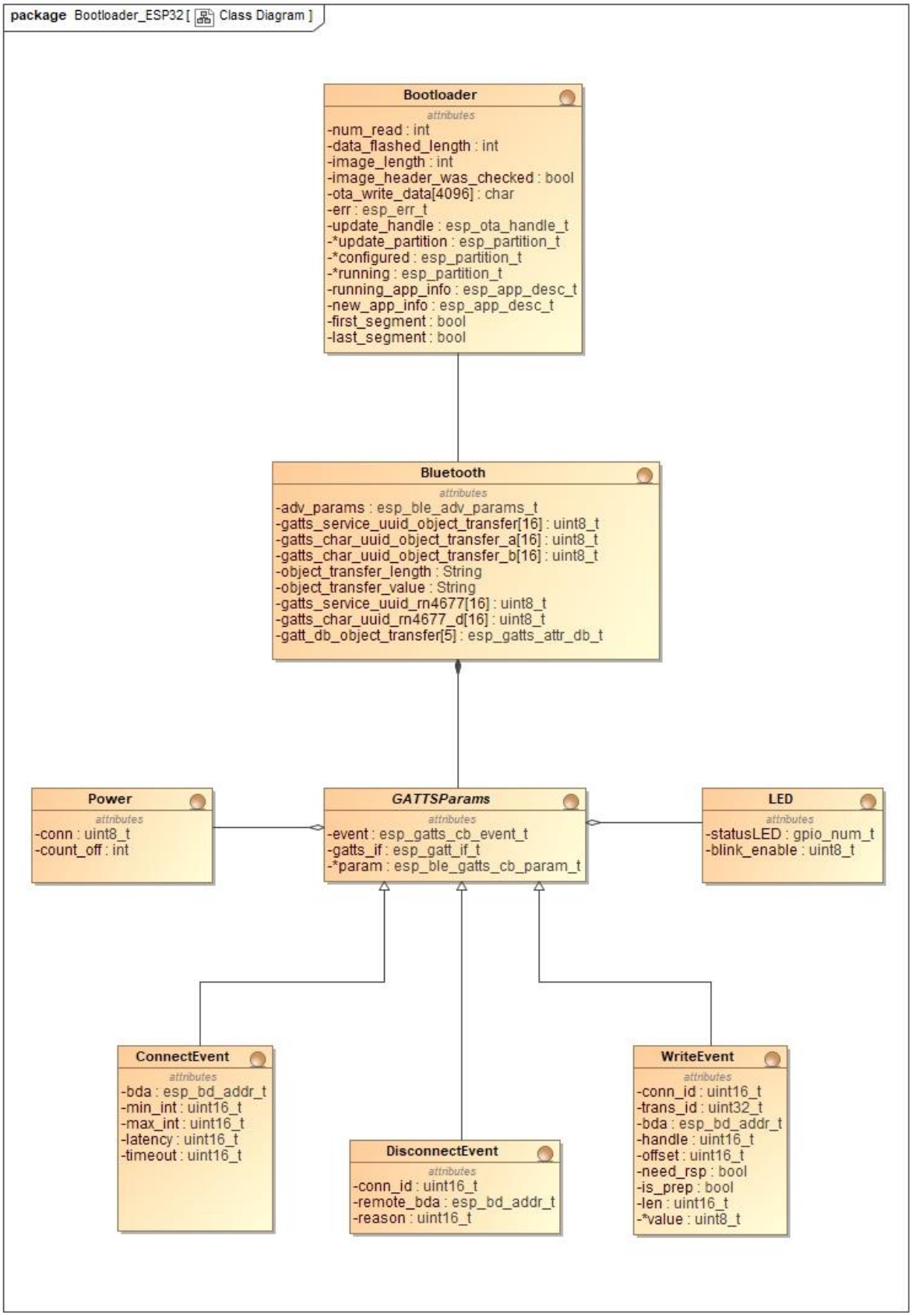

Figura 8. Diagrama de classes d'entitat sense operacions

3.3.2 Anàlisi dels casos d'ús

3.3.2.1 Cd'ú 00. ConnectDevice

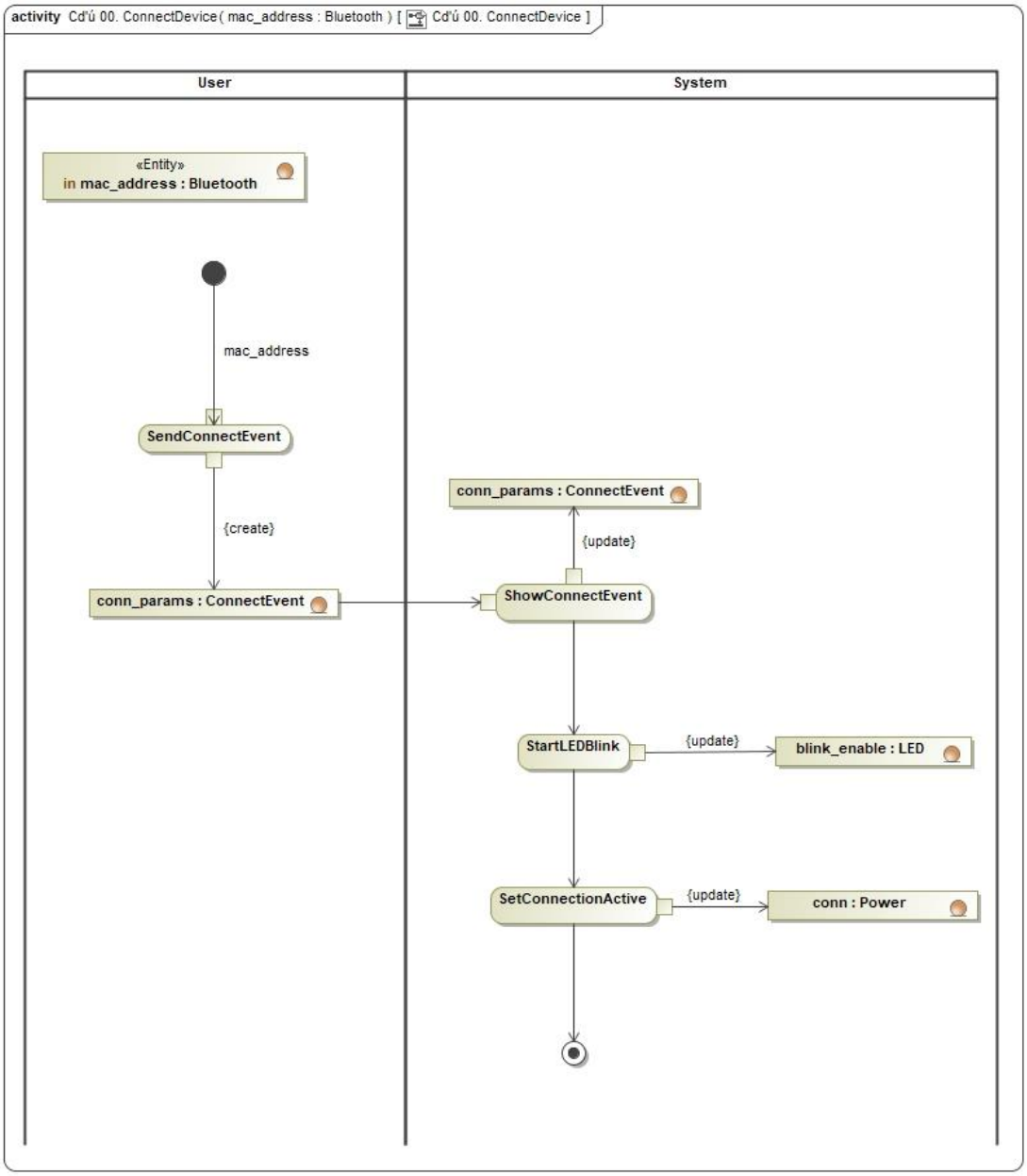

Figura 9. Diagrama d'activitats del Cd'ú 00. ConnectDevice

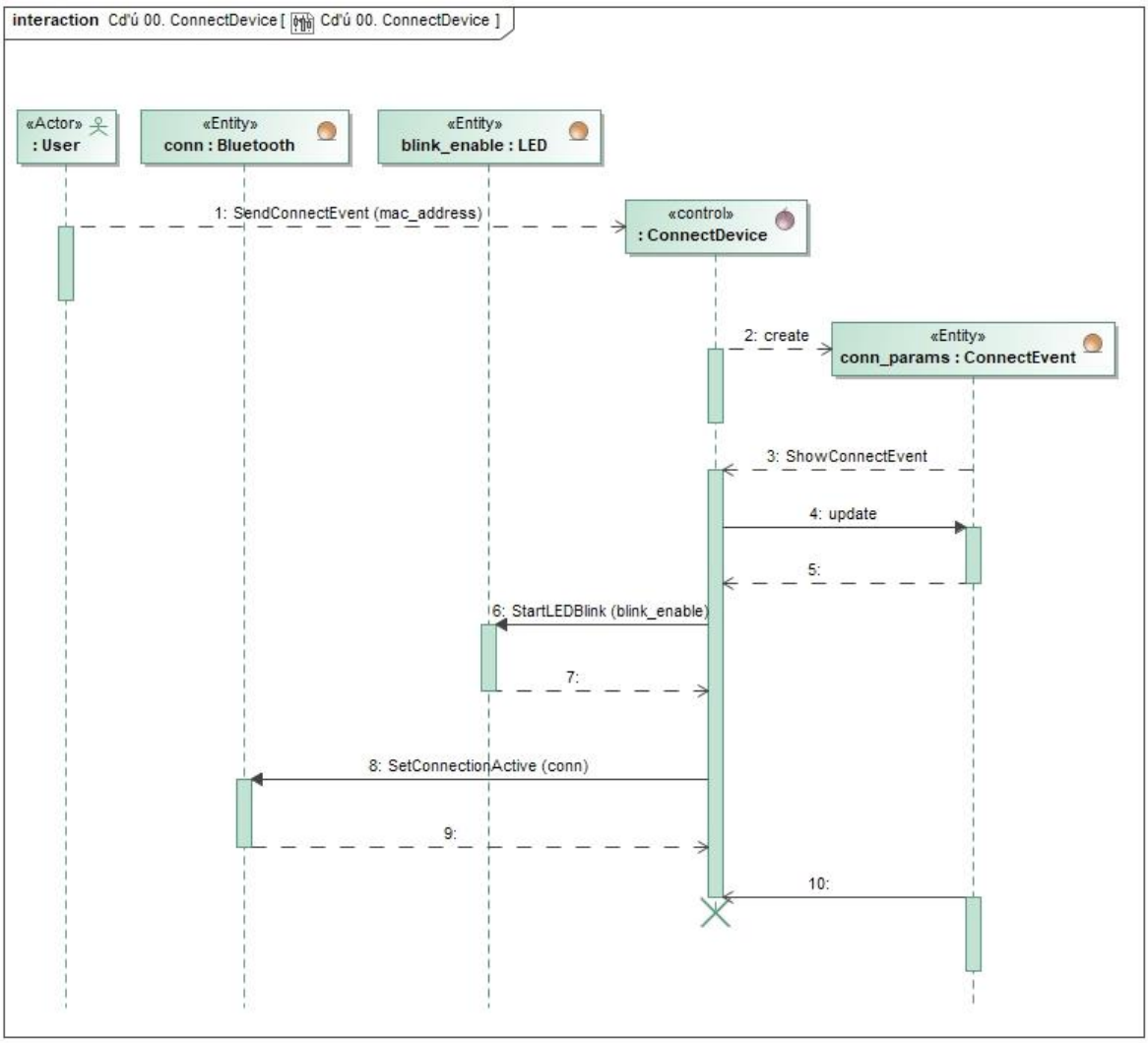

Figura 10. Diagrama de seqüències del Cd'ú 00. ConnectDevice

3.3.2.2 Cd'ú 01. UpdateFirmware

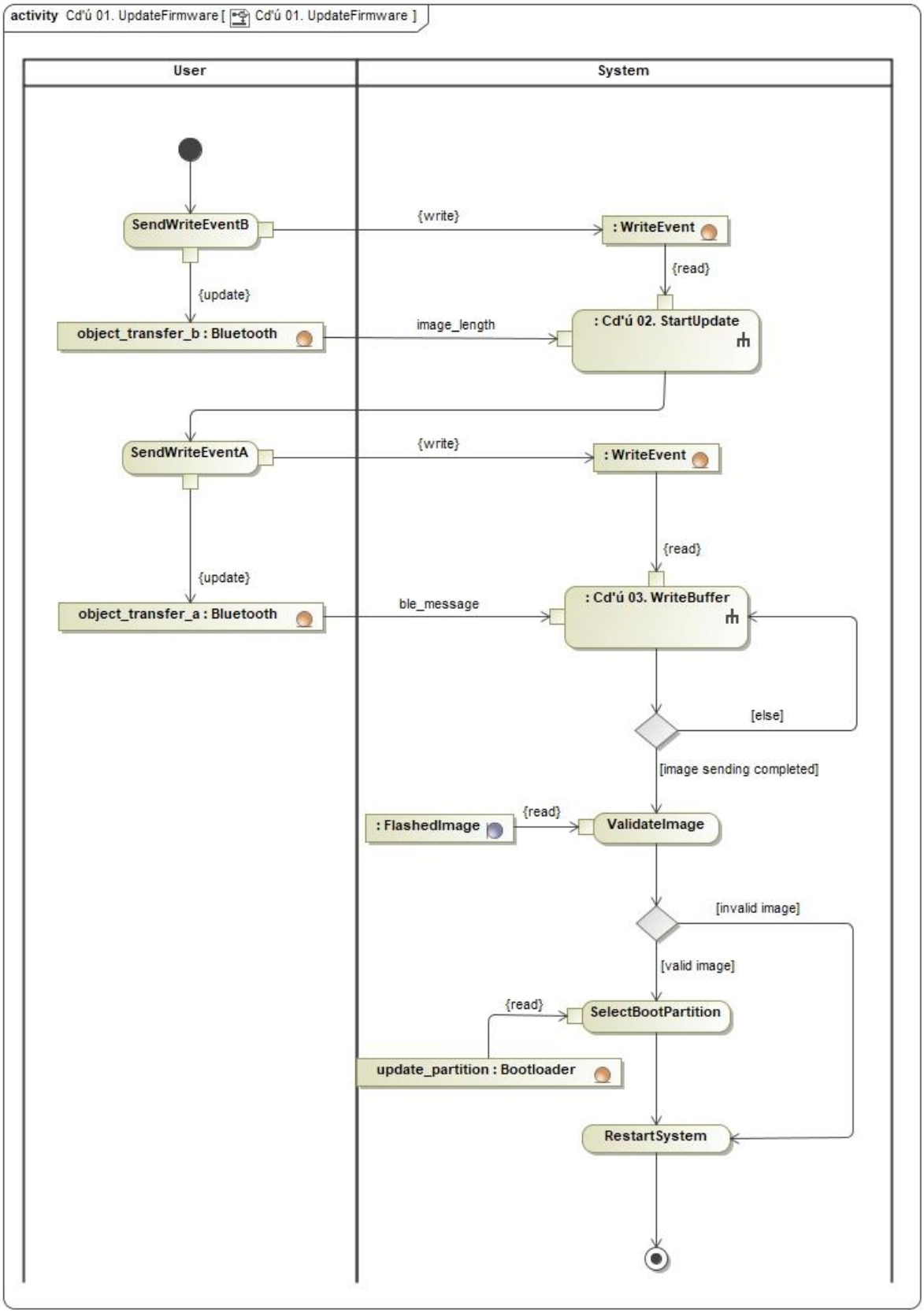

Figura 11. Diagrama d'activitats del Cd'ú 01. UpdateFirmware

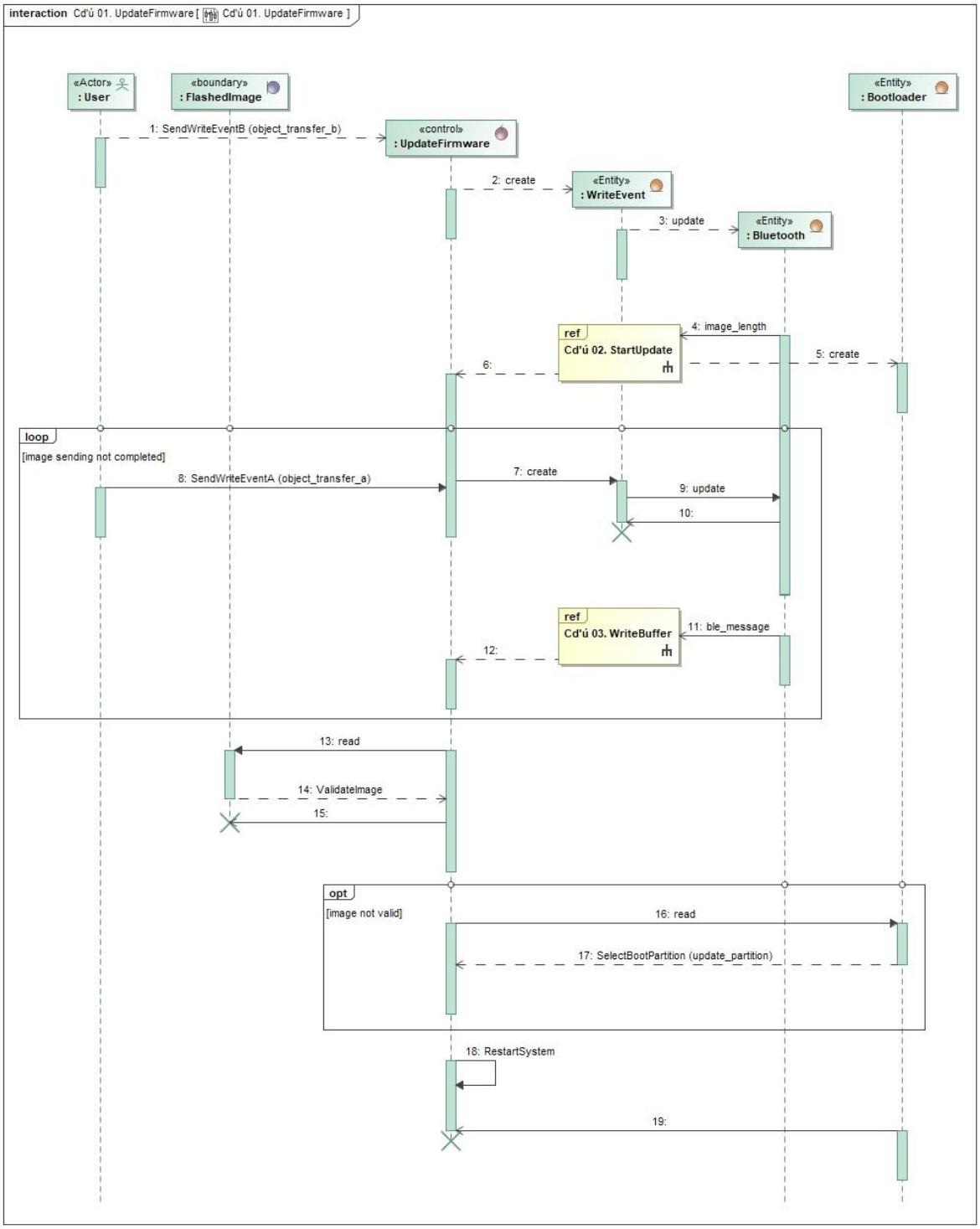

Figura 12. Diagrama de seqüències del Cd'ú 01. UpdateFirmware

3.3.2.3 Cd'ú 02. StartUpdate

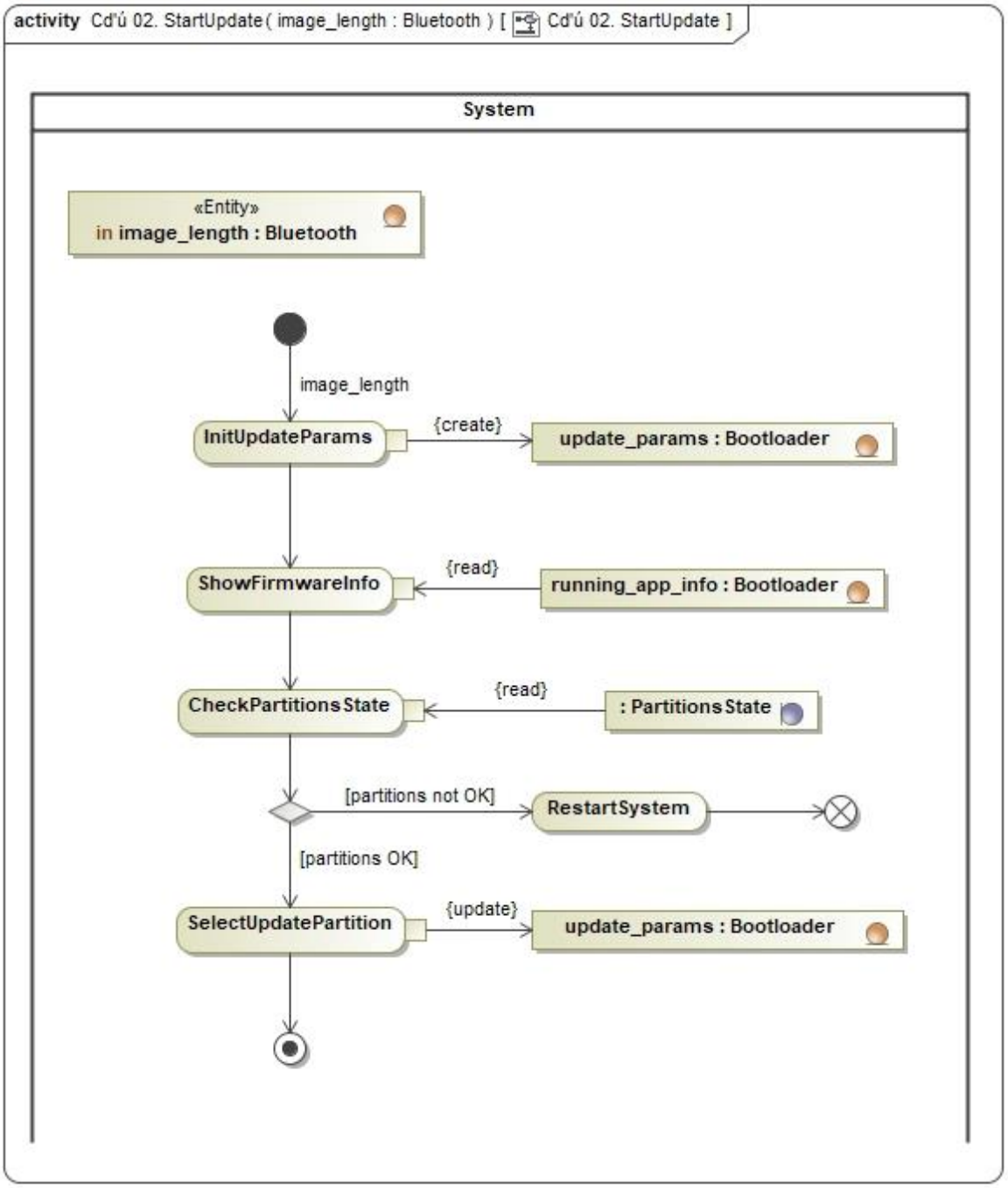

Figura 13. Diagrama d'activitats del Cd'ú 02. StartUpdate

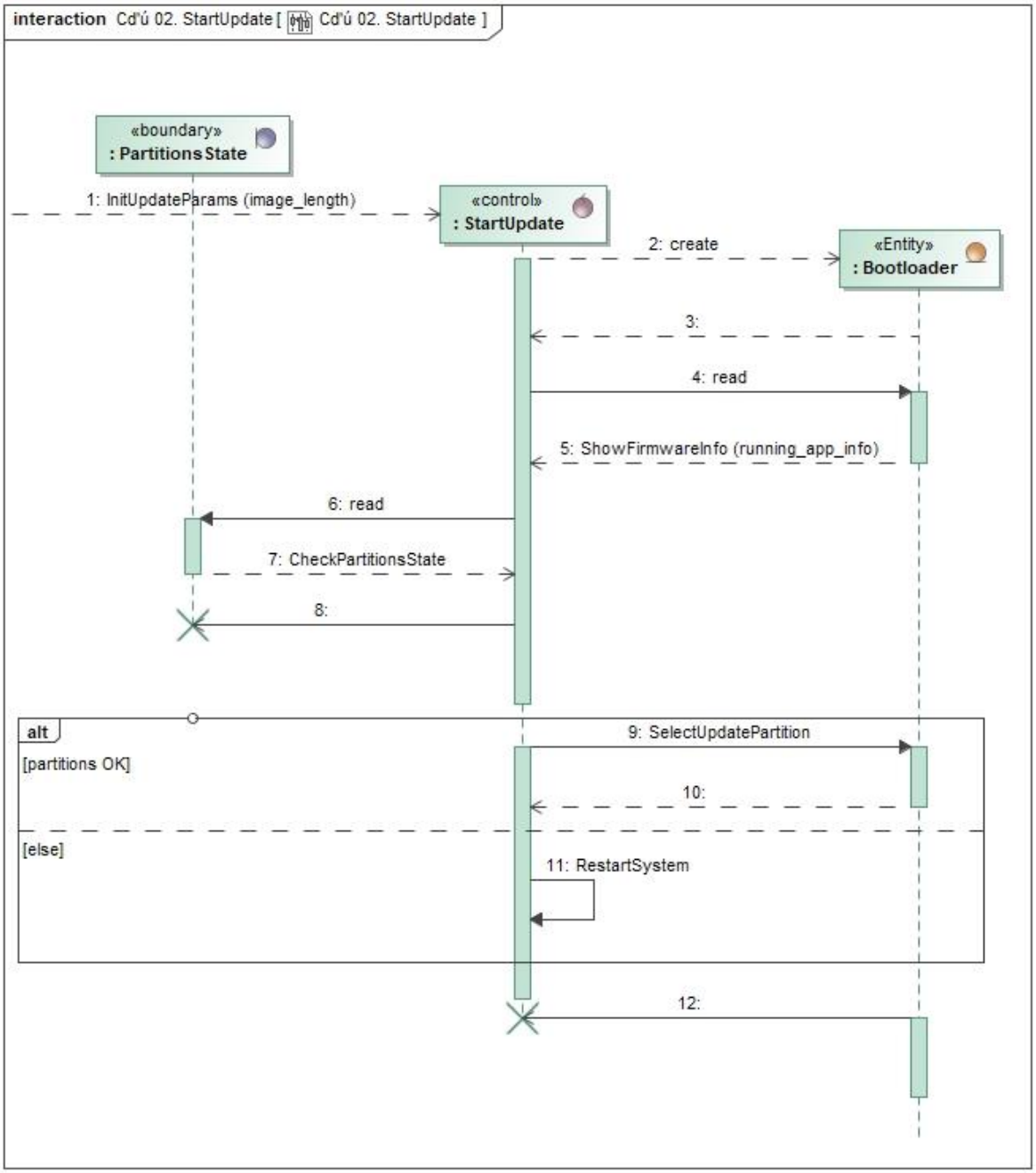

Figura 14. Diagrama de seqüències del Cd'ú 02. StartUpdate

3.3.2.4 Cd'ú 03. WriteBuffer

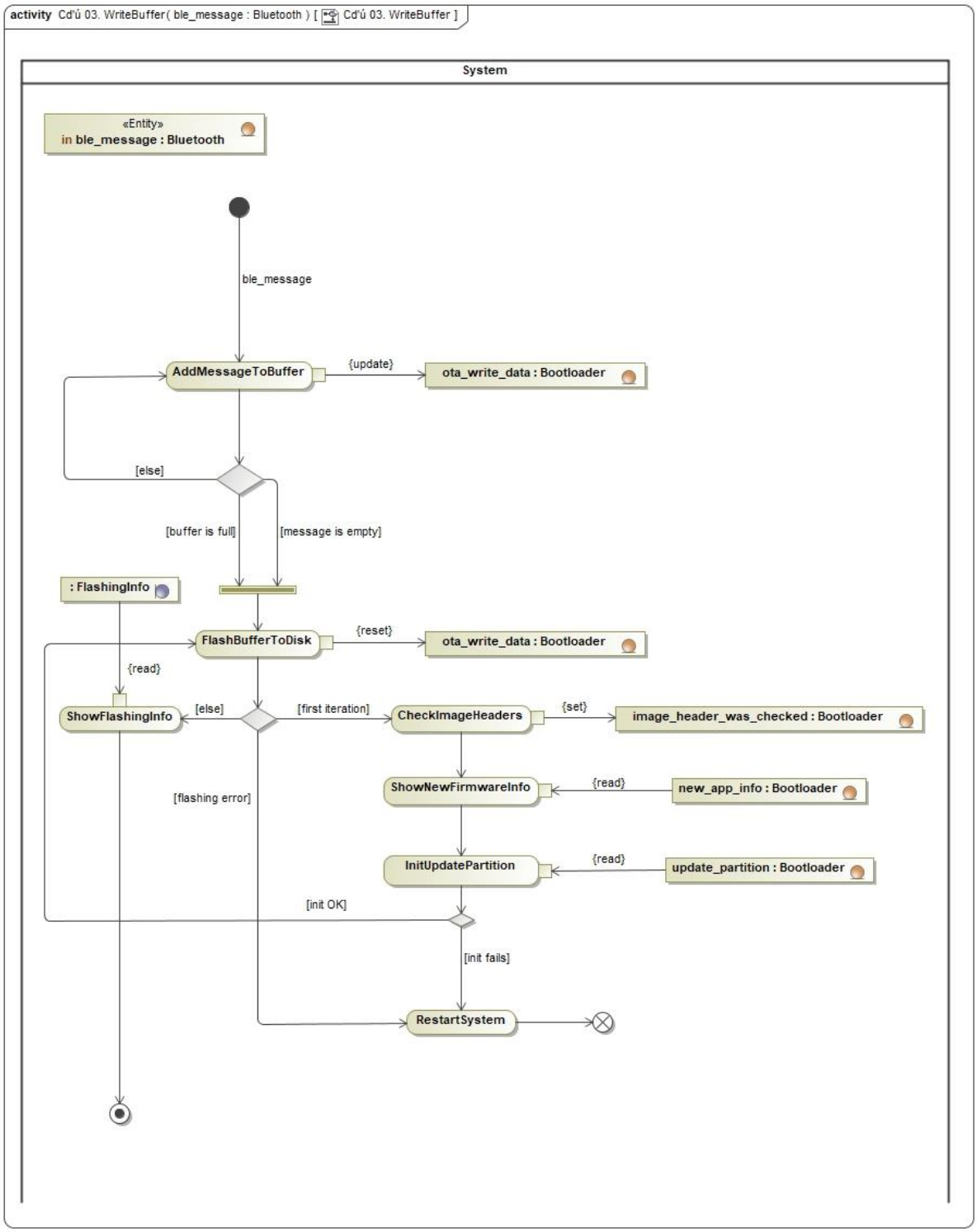

Figura 15. Diagrama d'activitats del Cd'ú 03. WriteBuffer

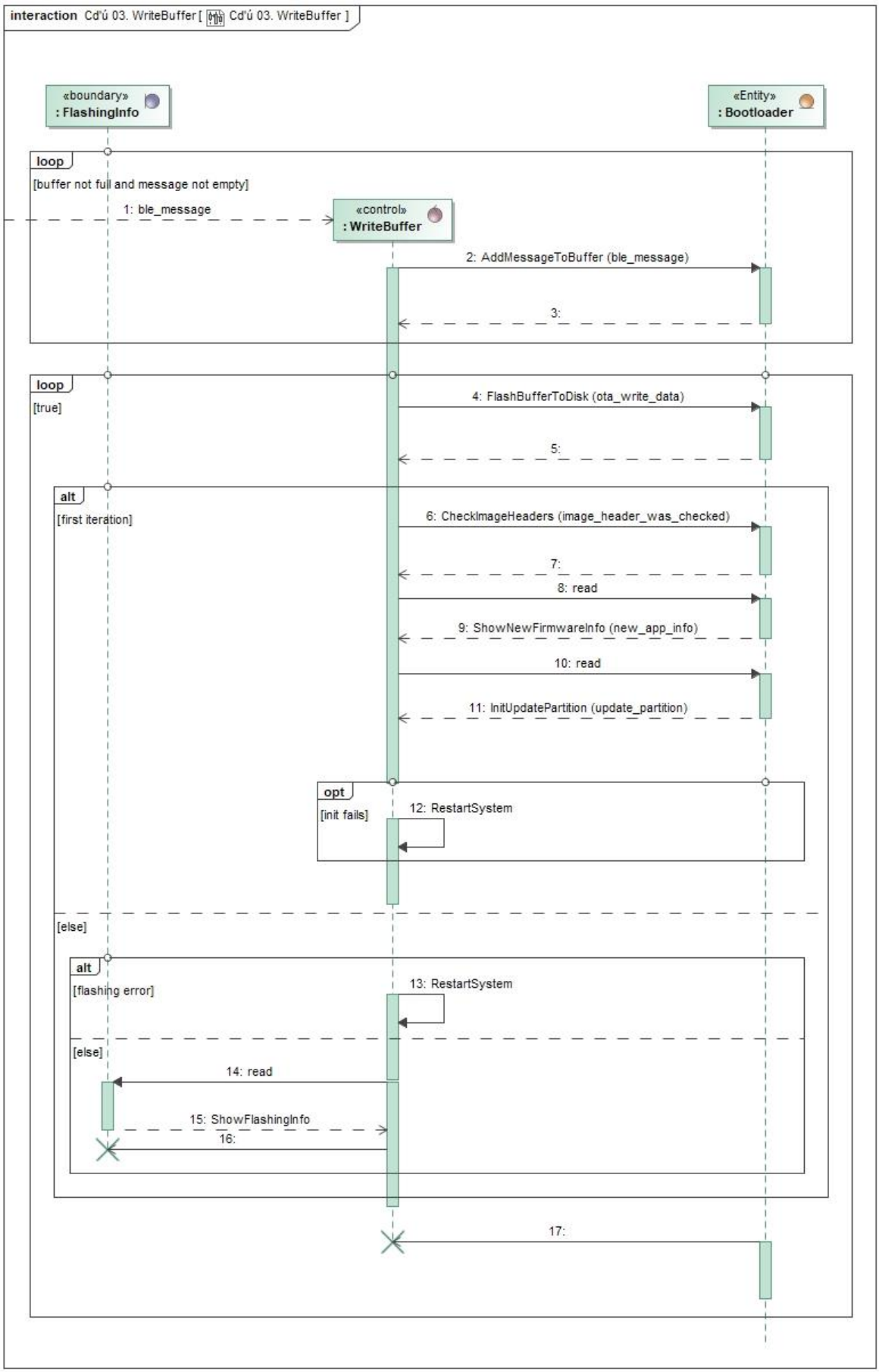

Figura 16. Diagrama de seqüències del Cd'ú 03. WriteBuffer

3.3.2.5 Cd'ú 04. DisconnectDevice

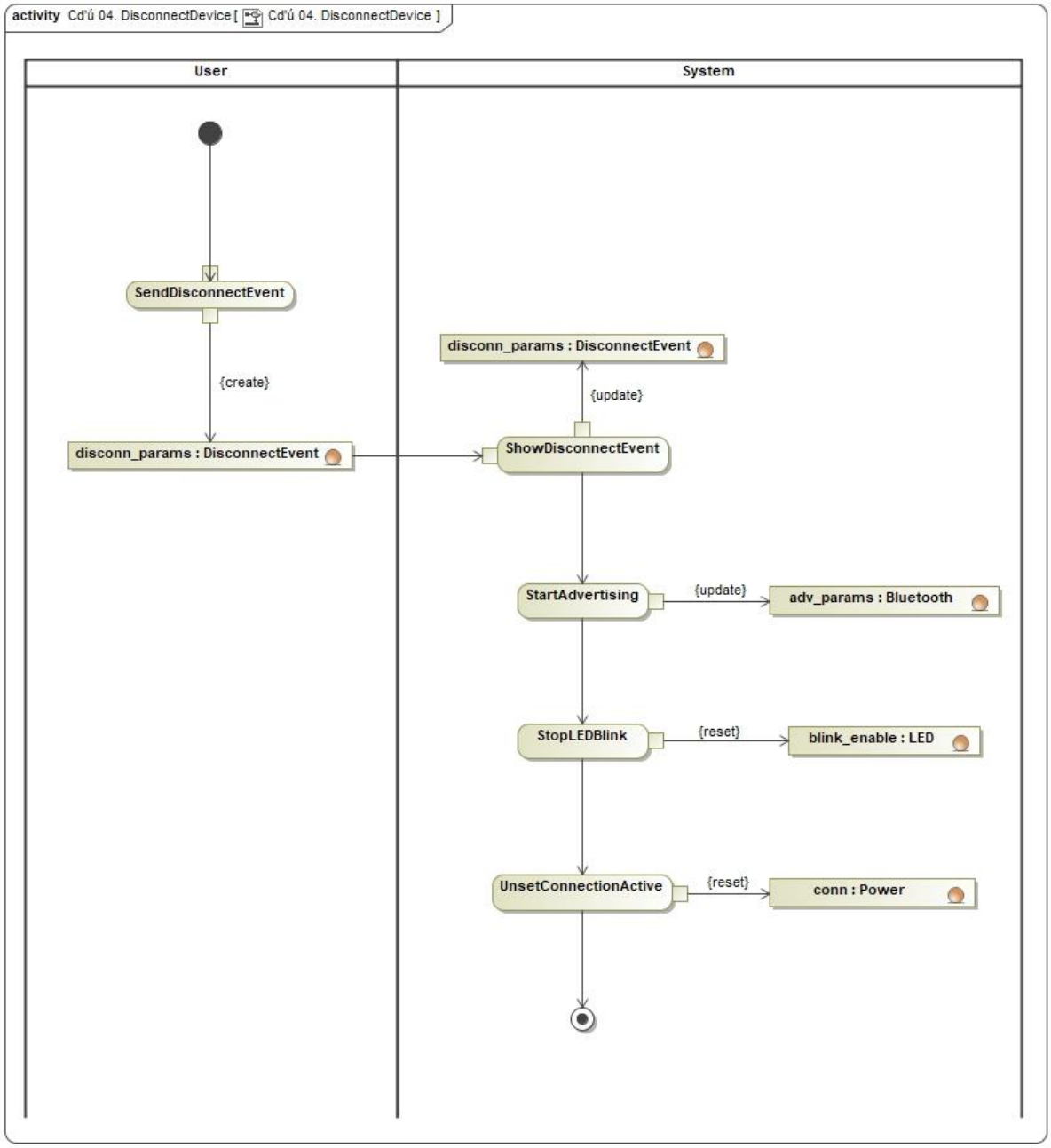

Figura 17. Diagrama d'activitats del Cd'ú 04. DisconnectDevice

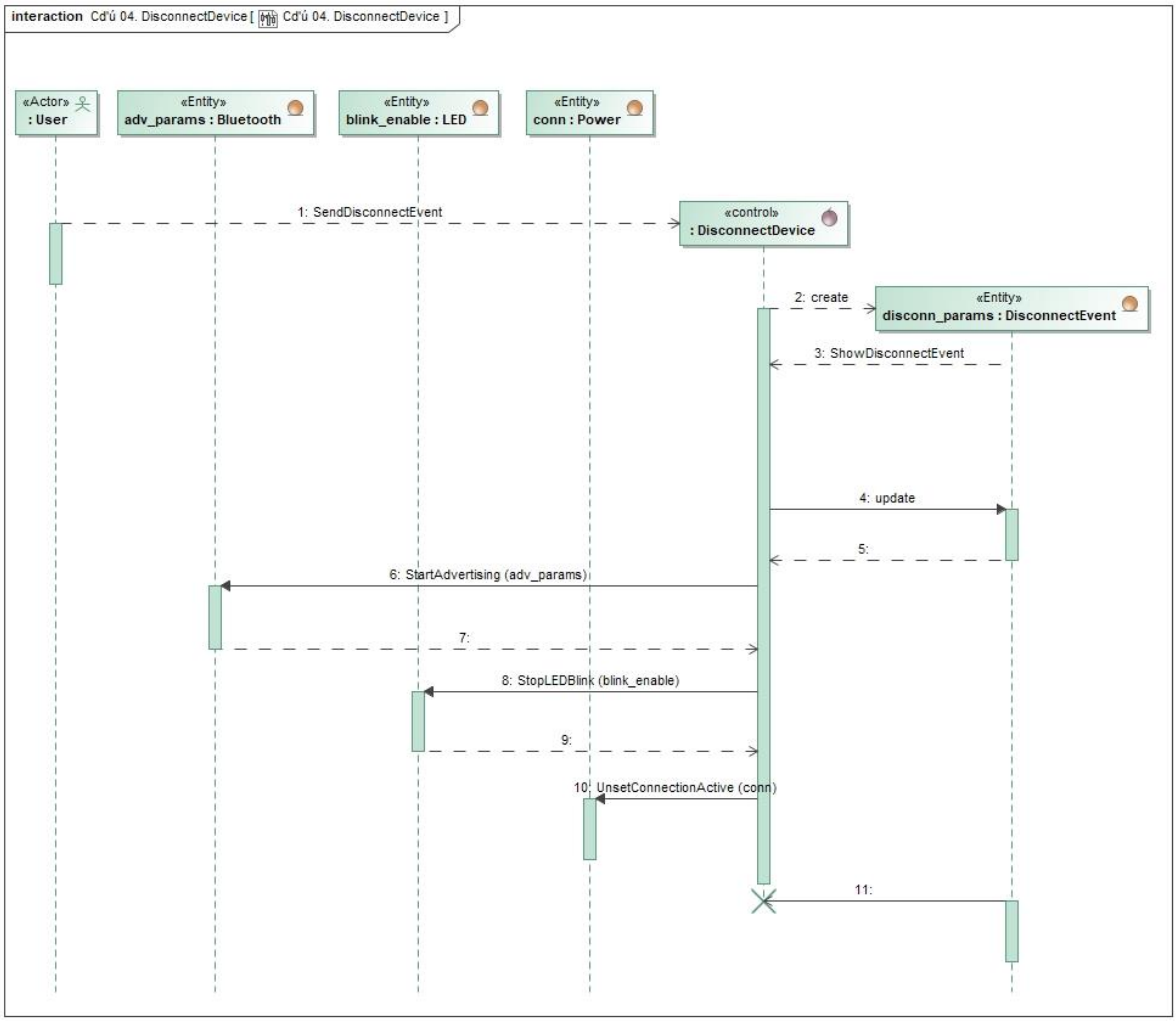

Figura 18. Diagrama de seqüències del Cd'ú 04. DisconnectDevice

4 Disseny

A l'hora de procedir amb el disseny del bootloader, es presenten tres reptes principals a resoldre.

En primer lloc, cal fer una anàlisi de la memòria interna del dispositiu. En tractar-se d'un dispositiu amb recursos molt limitats, és important entendre com funciona per tal de fer una gestió òptima dels mateixos, especialment de la memòria interna. Davant d'aquest panorama, cal prendre decisions de disseny en relació a quantes i quines particions són necessàries per obtenir el funcionament desitjat.

En segon lloc, cal decidir com estructurar la informació de cara a l'exterior, és a dir, aquella que incumbeix a altres dispositius. En aquest punt, entren en joc temes com l'arquitectura de BLE, quines capes té i com interaccionen entre elles. En el cas del component ESP32, aquesta gestió es pot fer mitjançant l'API que proporciona la pròpia Espressif Systems.

Per últim, i englobant els dos reptes anteriors, cal establir una jerarquia i un procediment per dur a terme les actualitzacions OTA. Aquí influeixen factors com el processament de les dades que es reben, control dels casos d'error, temps d'execució, eficiència, etc.

Tots els temes anteriors, entre d'altres, es tracten en els següents subapartats.

4.1 Gestió de la memòria interna

Durant el desenvolupament del codi del projecte, aquest ocupa un espai d'aproximadament 950 KB (veure Secció 3.1). Donat que el mòdul ESP32 té una memòria flash de 4 MB, cal fer una gestió òptima d'aquesta, per tal que hi hagi prou espai per tenir dues particions per fer les actualitzacions OTA i que tinguin marge per si la mida del codi augmenta en un futur.

Davant d'aquesta situació, cal identificar i assignar la mida adequada a les particions no prescindibles, que corresponen a la taula de particions, el bootloader de segona etapa i altres particions del sistema.

4.1.1 Descripció de les àrees de memòria

Les àrees de memòria del dispositiu tenen funcionalitats diverses i poden ser o no ser particions. La principal diferència entre elles és que les particions són visibles per l'usuari (p. ex. la partició ota_0), mentre que la resta són àrees de memòria reservades (p. ex. la taula de particions).

Tot seguit, es procedeix a enumerar les àrees de memòria presents al dispositiu, considerant si són particions o no, i mostrant la seva etiqueta en cas afirmatiu:

Nº	Àrea de memòria	Partició	Etiqueta
1	Espai reservat	No	–
2	Bootloader de segona etapa	No	–
3	Taula de particions	No	–
4	Emmagatzematge no volàtil (NVS)	Sí	`nvs`
5	Dades d'OTA	Sí	`otadata`
6	Inicialitzacions de la memòria física	Sí	`phy_init`
7	OTA 0	Sí	`ota_0`
8	OTA 1	Sí	`ota_1`
9	Espai lliure	Sí	–

Taula 4. Descripció de les àrees de memòria del dispositiu

Així doncs, es procedeix amb la descripció de les àrees de memòria. Tanmateix, cal esmentar que per tal de facilitar l'enteniment d'aquestes i ubicar-les dins de la memòria, es poden contrastar gràficament mitjançant el mapa de memòria (veure Secció 4.1.2).

4.1.1.1 Espai reservat

Ocupa 4 KB, des de l'adreça 0x000000 fins a la 0x001000 de la memòria interna del dispositiu.

Aquesta àrea de memòria es tracta d'un espai reservat pel sistema per arrencades segures o *secure boot*[8]. Quan aquest està activat, s'utilitzen els primers 192 bytes per emmagatzemar la sortida del bootloader, mentre que la resta d'espai queda inutilitzat. En cas contrari, l'àrea de memòria sencera queda inutilitzada.

Malgrat l'espai inutilitzat d'aquesta àrea de memòria, aquest és necessari, ja que la mida mínima d'una partició és d'un sector (veure Secció 4.1.2).

En el cas del present projecte, la funcionalitat d'arrencada segura no s'utilitza i, en conseqüència, l'àrea de memòria queda inutilitzada.

[8] L'arrencada segura o *secure boot* és una funcionalitat que garanteix que només es pot executar codi propi del fabricant al xip en qüestió.

$$MIDA_{ESPAI\ RESERVAT} = 4\ KB$$

Fórmula 1. Mida de l'espai reservat del disc

4.1.1.2 Bootloader de segona etapa

Ocupa 28 KB, des de l'adreça 0x001000 fins a la 0x008000 de la memòria interna del dispositiu.

Aquesta àrea de memòria és on s'ubica el bootloader de segona etapa, que és carregat a la memòria principal pel bootloader de primera etapa situat a la memòria ROM (veure Secció 2.1).

El bootloader de segona etapa s'encarrega de carregar la taula de particions i seleccionar la partició que s'executa.

En el cas del present projecte, el bootloader de segona etapa consultarà la partició `otadata` per triar quina partició OTA entra en execució.

$$MIDA_{BOOTLOADER} = 28\ KB$$

Fórmula 2. Mida del bootloader de segona etapa

4.1.1.3 Taula de particions

Ocupa 4 KB, des de l'adreça 0x008000 fins a la 0x009000 de la memòria interna del dispositiu.

La taula de particions conté l'especificació de totes les particions del sistema i es grava directament a la direcció de memòria 0x8000. Aquesta s'encarrega d'indicar l'etiqueta, el tipus, el subtipus, el desplaçament o *offset* i la mida de cada partició.

Aquesta especificació es defineix en un document en format CSV[9] i es carrega a la configuració del SDK[10] on es tria la ruta del fitxer i la seva direcció de memòria.

En el cas del present projecte, la taula de particions emprada per al dispositiu és la següent:

9 Els fitxers CSV, de l'anglès *Comma Separated Values*, són un tipus de document de format obert emprats per representar dades en forma de taula.

10 Un SDK, de l'anglès *Software Development Kit*, és un conjunt d'eines de desenvolupament de software que faciliten la creació d'aplicacions.

Etiqueta	Tipus	Subtipus	Desplaçament *(offset)*	Mida
nvs	data	nvs	0x009000	16 KB
otadata	data	ota	0x00D000	8 KB
phy_init	data	phy	0x00F000	4 KB
ota_0	app	ota_0	0x010000	1984 KB
ota_1	app	ota_1	0x200000	1984 KB

Taula 5. Especificació de la taula de particions del dispositiu

$$MIDA_{TAULA_{PARTICIONS}} = 4\ KB$$

Fórmula 3. Mida de la taula de particions

4.1.1.4 Emmagatzematge no volàtil (NVS)

Ocupa 16 KB, des de l'adreça 0x009000 fins a la 0x00D000 de la memòria interna del dispositiu.

La partició `nvs` s'encarrega d'emmagatzemar parelles clau-valor a la memòria flash. Aquestes parelles solen ser dades corresponents a la memòria física, al Wi-Fi o altres dades d'aplicació.

La mida d'aquesta partició varia en funció de la quantitat de dades que es desitgi emmagatzemar. Segons l'API d'Espressif Systems, tot i que no es faci un ús especial d'aquesta partició és recomanable que tingui una mida mínima de 12 KB.

En el cas del present projecte, la mida establerta s'ha heretat de la configuració alternativa per a dues particions OTA proporcionada per Espressif Systems.

4.1.1.5 Dades d'OTA

Ocupa 8 KB, des de l'adreça 0x00D000 fins a la 0x00F000 de la memòria interna del dispositiu.

La partició `otadata` s'encarrega d'emmagatzemar dades per a les actualitzacions OTA. El bootloader de segona etapa consulta aquesta partició amb la fi d'escollir quina partició s'executa. En cas que aquesta partició estigui buida, es passa a executar l'aplicació predeterminada o, en el seu defecte, la partició `ota_0`.

En el cas del present projecte, s'aprofita la lògica anterior per establir la partició `ota_0` com a predeterminada.

$$MIDA_{PARTICIÓ_{NVS}} = 16\ KB$$

Fórmula 4. Mida de la partició NVS

4.1.1.6 Inicialitzacions de la memòria física

Ocupa 4 KB, des de l'adreça 0x00F000 fins a la 0x010000 de la memòria interna del dispositiu.

La partició `phy_init` s'encarrega d'emmagatzemar dades d'inicialització de la memòria física. D'aquesta manera, aquesta es pot configurar de forma específica per a cada dispositiu, en comptes de fer-ho a través del firmware.

En el cas del present projecte, la partició `phy_init` no s'utilitza i les dades d'inicialització de la memòria física es compilen directament a l'aplicació. No obstant, aquesta partició s'ha mantingut davant la possibilitat de futures aplicacions.

$$MIDA_{PARTICIÓ_{PHY_{INIT}}} = 4\ KB$$

Fórmula 5. Mida de la partició d'inicialitzacions de la memòria física

4.1.1.7 OTA 0

Ocupa 1984 KB, des de l'adreça 0x010000 fins a la 0x200000 de la memòria interna del dispositiu.

La partició `ota_0` és una de les dues particions per a les actualitzacions OTA. Segons l'API d'Espressif Systems hi pot haver fins a 16 particions com aquesta. Ara bé, en cas del present projecte aquesta partició és també la predeterminada com a principal per al dispositiu, partint d'un inici (veure Secció 4.1.2).

4.1.1.8 OTA 1

Ocupa 1984 KB, des de l'adreça 0x200000 fins a la 0x3F0000 de la memòria interna del dispositiu.

La partició `ota_1` és la segona de les dues particions per a les actualitzacions OTA (veure Secció 4.1.2). En funció de les dades de la partició `otadata`, el bootloader triarà executar la partició `ota_0` o, en el seu defecte, aquesta.

4.1.1.9 Espai lliure

Ocupa 64 KB, des de l'adreça 0x3F0000 fins a la 0x400000 de la memòria interna del dispositiu.

Aquesta àrea de memòria es tracta d'un espai que ha quedat lliure per tal de mantenir la simetria del disc (veure Secció 4.1.2) o davant possibles futures millores (veure Secció 4.1.3).

4.1.2 Decisions de disseny

Aquest disseny sorgeix dels requisits especificats a l'API d'Espressif Systems, que estableixen que la mida mínima d'una partició és 4 KB i que totes les particions de tipus *app* han d'estar alineades a 64 KB.

Per tal de simplificar la gestió de la memòria interna del dispositiu, així com garantir l'espai suficient per a cada una de les particions mantenint els requisits funcionals i deixant marge de cara a possibles canvis en el futur, es proposa un disseny basat en un concepte de simetria.

Així doncs, s'ha decidit estructurar el disc mitjançant els conceptes de segment i sector. Es defineix com a segment a cada fragment de 64 KB que està alineat a 64 KB. Anàlogament, es defineix com a sector a cada fragment de 4 KB que està alineat a 4 KB.

Tenint en compte que la mida de la memòria flash del component ESP32 és de 4 MB, s'obtenen les següents fórmules:

$$MIDA_{DISC_{ESP32}} = 4\,MB = 4096\,KB$$

Fórmula 6. Mida total del disc del component ESP32

$$MIDA_{SECTOR} = 4\,KB$$

Fórmula 7. Mida d'un sector del disc

$$MIDA_{SEGMENT} = 64\,KB$$

Fórmula 8. Mida d'un segment del disc

$$N_{SECTORS} = \frac{MIDA_{DISC_{ESP32}}}{MIDA_{SECTOR}} = \frac{4096\,KB}{4\,KB} = 1024\,sectors$$

Fórmula 9. Nombre total de sectors del disc

$$N_{SEGMENTS} = \frac{MIDA_{DISC_{ESP32}}}{MIDA_{SEGMENT}} = \frac{4096\,KB}{64\,KB} = 64\,segments$$

Fórmula 10. Nombre total de segments del disc

$$\frac{N_{SECTORS}}{MIDA_{SEGMENT}} = \frac{1024\ sectors}{1\ segment} = \frac{1024\ sectors}{64\ KB} = 16\ \frac{sectors}{segment}$$

Fórmula 11. Nombre de sectors per segment del disc

$$MIDA_{DISC_{ESP32}} = 4\ MB = 4096\ KB = 64\ segments = 1024\ sectors$$

Fórmula 12. Equivalència entre les unitats de mesura de la mida del disc del component ESP32

Realitzant una síntesi de les fórmules anteriors, s'obté el següent diagrama, que representa una visió conjunta de la memòria interna del component ESP32:

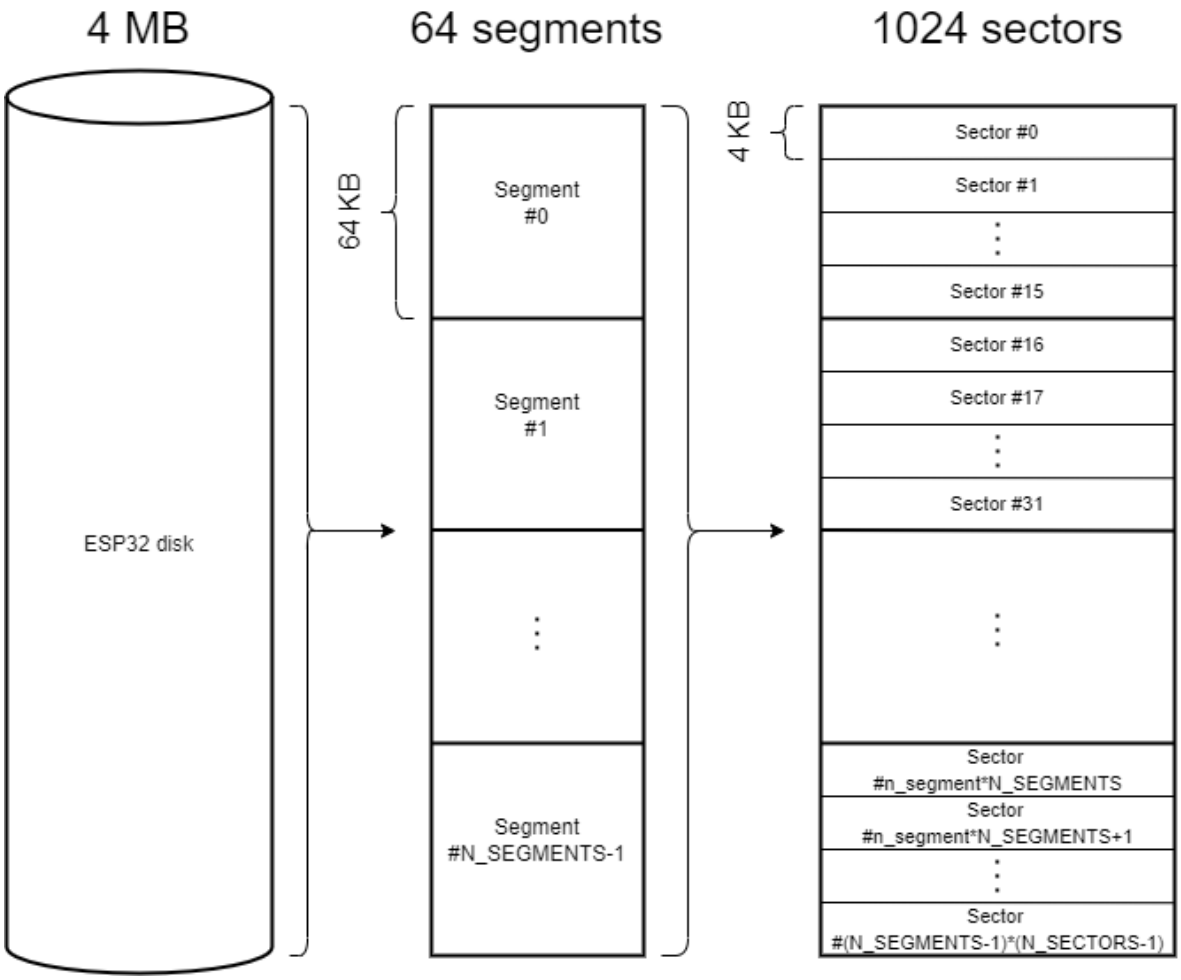

Figura 19. Diagrama de la memòria interna del dispositiu.

Per tant, el següent pas respecte al disseny de la memòria interna és especificar la mida de les particions OTA.

Això és:

$$MIDA_{PARTICIÓ_{OTADATA}} = 8\ KB$$

Fórmula 13. Mida de la partició de dades d'OTA

$$MIDA_{SEGMENT\#0} =$$

$$= MIDA_{ESPAI\ RESERVAT} + MIDA_{BOOTLOADER} + MIDA_{TAULA_{PARTICIONS}} +$$

$$+ MIDA_{PARTICIÓ_{NVS}} + MIDA_{PARTICIÓ_{OTADATA}} + MIDA_{PARTICIÓ_{PHY_{INIT}}} =$$

$$= 4\ KB + 28\ KB + 4\ KB + 16\ KB + 8\ KB + 4\ KB = 64\ KB$$

Fórmula 14. Mida desglossada del primer segment del disc

$$MIDA_{RESTANT} = MIDA_{DISC_{ESP32}} - MIDA_{SEGMENT\#0} =$$

$$= 64 - 1\ segments = 63\ segments = 4032\ KB$$

Fórmula 15. Espai del disc no ocupat per les particions del sistema

$$N_{PARTICIONS_{OTA}} = 2\ particions$$

Fórmula 16. Nombre de particions OTA necessàries per satisfer els requisits del sistema

$$MIDA_{IDEAL_{OTA}} = \frac{MIDA_{RESTANT}}{N_{PARTICIONS_{OTA}}} = \frac{63\ segments}{2\ particions} = 31{,}5\ \frac{segments}{partició} = 2016\ \frac{KB}{partició}$$

$$MIDA_{IDEAL_{OTA}} = \nexists$$

Fórmula 17. Mida màxima ideal de cada partició OTA

En cas que es volgués aprofitar tot l'espai del disc, inclòs l'últim segment, la mida restant s'hauria de repartir equitativament, implicant que cada partició OTA augmentaria mig segment, és a dir, 32 KB. Per tant, donat que són particions de tipus app, s'estaria incomplint el requisit de l'API d'Espressif Systems que indica que les particions han d'estar alineades a 64 KB.

Fent referència al concepte de simetria esmentat anteriorment, s'ha decidit tractar per separat el primer i l'últim segment del disc, deixant els 62 segments restants per a les dues particions principals necessàries per a les actualitzacions OTA.

D'aquesta manera, al primer segment s'hi troben les particions del sistema (veure Secció 4.1.2) mentre que l'últim segment és espai lliure.

Cal esmentar que aquesta decisió s'ha pres per tal de poder aprofitar el màxim espai possible del disc mantenint dues particions OTA simètriques.

Això és:

$$MIDA_{SEGMENT\#63} = MIDA_{ESPAI\ LLIURE} = 1\ segment = 64\ KB$$

Fórmula 18. Mida de l'últim segment del disc

$$MIDA_{PARTICIÓ_{OTA}} = \frac{MIDA_{RESTANT} - 1}{N_{PARTICIONS_{OTA}}} = \frac{62\ segments}{2\ particions} =$$

$$= 31\ \frac{segments}{partició} = 496\ \frac{sectors}{partició} = 1984\ \frac{KB}{partició}$$

Fórmula 19. Mida màxima real de cada partició OTA

$$MIDA_{DISC_{ESP32}} = \sum MIDA_{ÀREES_{MEMÒRIA}} =$$

$$= MIDA_{SEGMENT\#0} + 2 * MIDA_{PARTICIÓ_{OTA}} + MIDA_{SEGMENT\#63} =$$

$$= 64\ KB + 2 * 1984\ KB + 64\ KB = 4096\ KB = 4\ MB$$

Fórmula 20. Equivalència entre la mida del disc i el sumatori de les àrees de memòria

Així doncs, tenint en compte les anteriors fórmules, s'obtenen els següents diagrames:

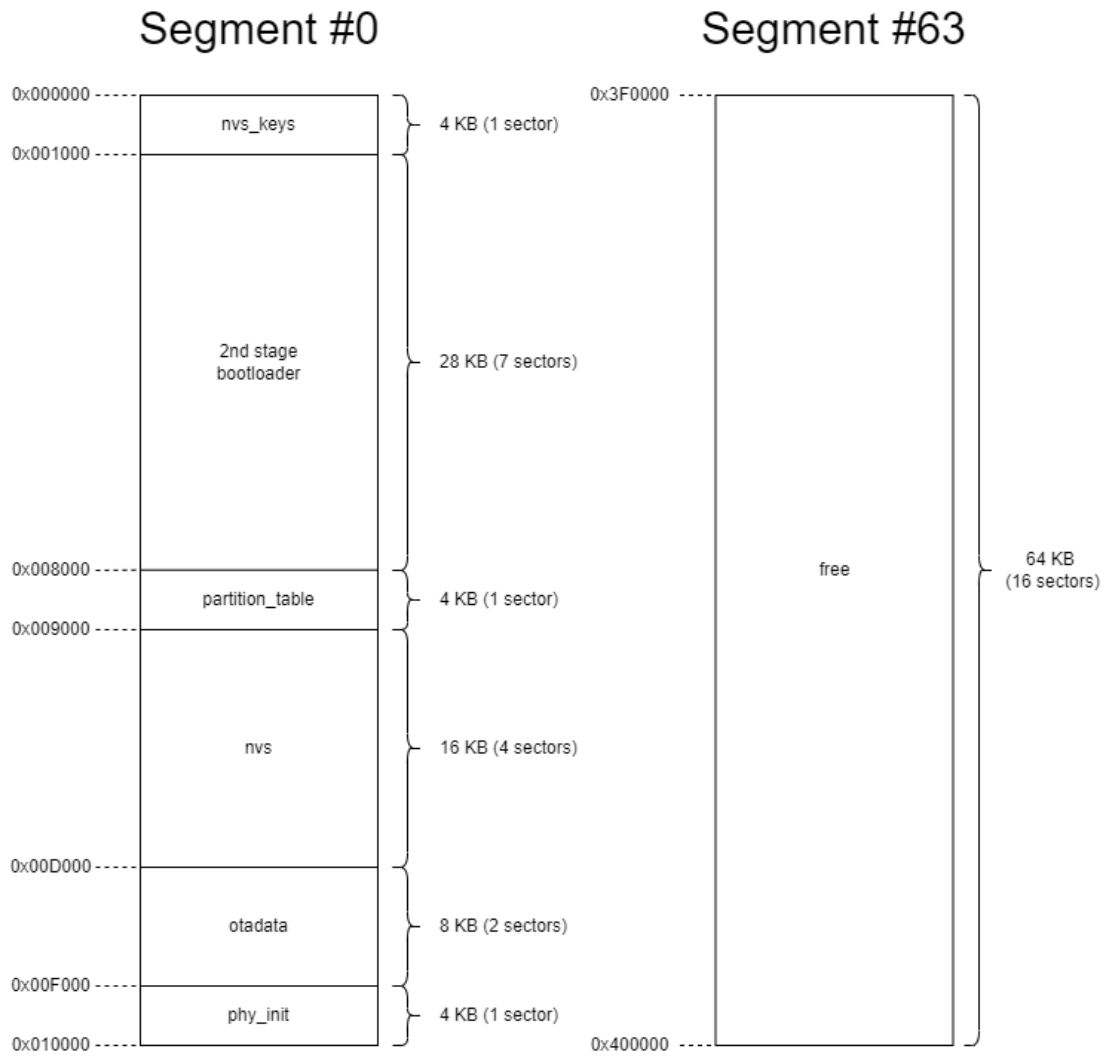

Figura 20. Diagrama del primer i últim segment de la memòria interna del dispositiu

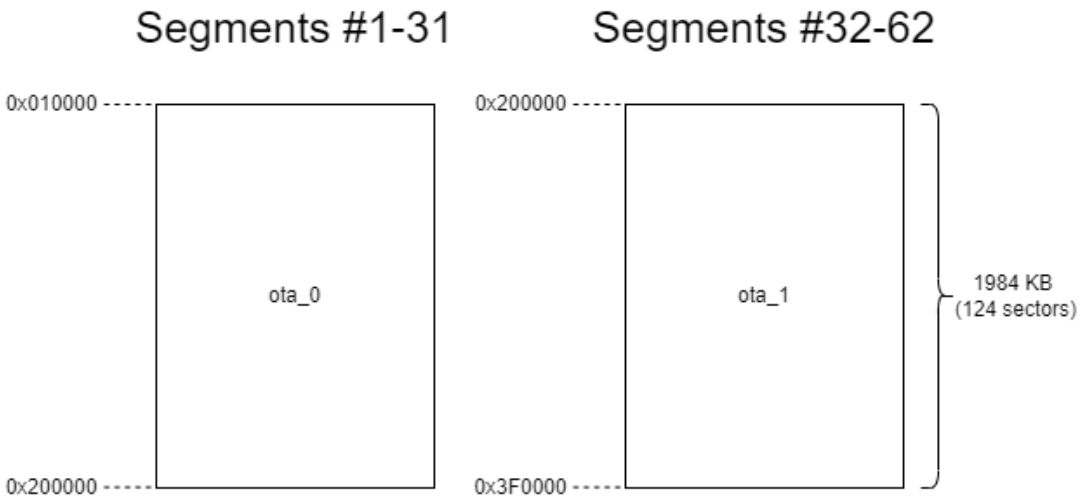

Figura 21. Diagrama dels segments restants de la memòria interna del dispositiu.

Per últim, s'especifica la mida de la imatge del firmware establerta segons els requisits del sistema, inclosos els marges d'increment davant possibles futurs desenvolupaments.

$$MIDA_{IMATGE_{FIRMWARE}} = 950\ KB$$

$$MIN\left(MIDA_{IMATGE_{FIRMWARE}}\right) = 950 - 100\ KB = 850\ KB$$

$$MAX\left(MIDA_{IMATGE_{FIRMWARE}}\right) = 950 + 100\ KB = 1050\ KB$$

Fórmula 21. Mida de la imatge del firmware del dispositiu

4.1.3 Marge de millora

El disseny proposat a l'anterior apartat (veure Secció 4.1.2) ha estat pensat per tenir certa flexibilitat davant la possibilitat que en el futur aparegui una necessitat d'incrementar l'espai de les particions. En concret, aquest disseny s'ha creat per poder suportar un increment del 200% tant de la mida de la imatge del firmware com de les particions del sistema, a excepció de les particions OTA que ja estan optimitzades per ocupar el màxim espai possible.

Per aquest motiu, a continuació es procedeix a analitzar el marge de millora que ofereix aquest disseny.

En primer lloc, s'analitza el marge de millora respecte a la mida de la imatge del firmware:

$$ESPAI_{OCUPAT_{OTA}} = \frac{MIDA_{IMATGE_{FIRMWARE}}}{MIDA_{PARTICIÓ_{OTA}}} = \frac{950\ KB}{1984\ KB} \cong 47{,}9\%$$

Fórmula 22. Espai d'una partició OTA ocupat per la imatge del firmware

Aplicant l'increment del 200% de l'anterior espai ocupat, es pot comprovar que seguiria cabent dins de la partició:

$$ESPAI_{INCREMENTAT_{OTA}} = 2 * ESPAI_{USAT_{OTA}} = \frac{1900\ KB}{1984\ KB}$$

$$ESPAI_{INCREMENTAT_{OTA}} \cong 95{,}8\% \leq 100\%$$

Fórmula 23. Espai d'una partició OTA ocupat per una imatge del firmware incrementada

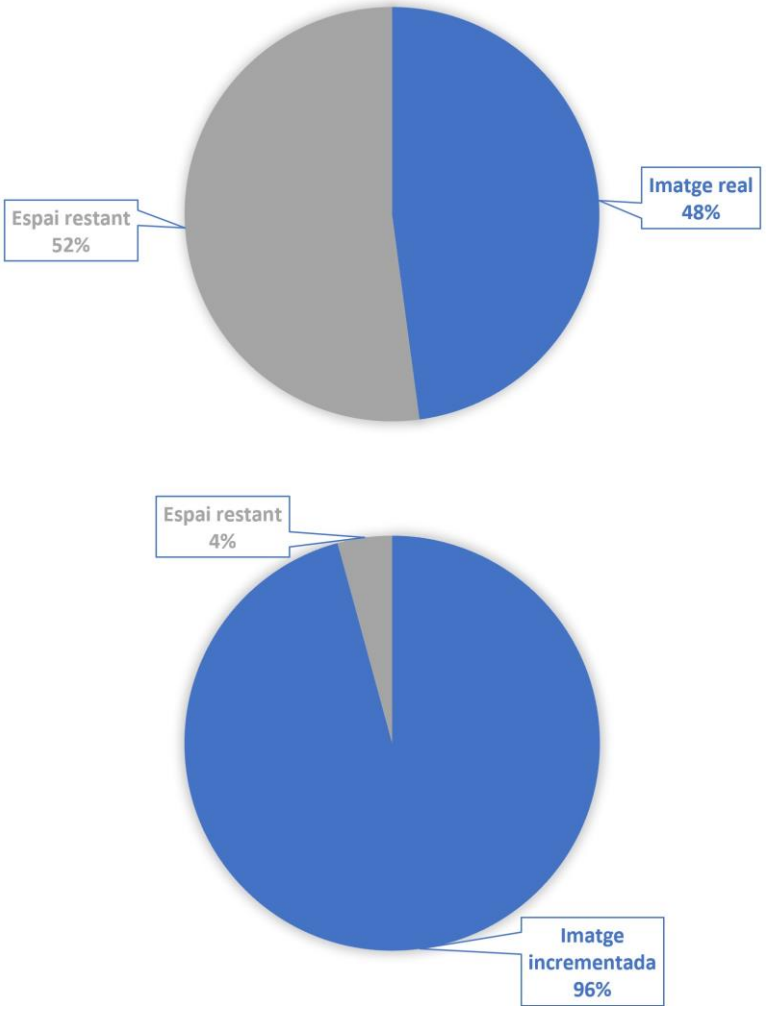

Figura 22. Gràfica comparativa de l'espai ocupat per la imatge real i incrementada

Aleshores, de les fórmules i gràfica comparativa anterior es dedueix que el disseny suportaria un increment del 200% respecte a les particions del sistema.

En segon lloc, s'analitza el marge de millora de la mida de les particions del sistema:

Concretament, aquest increment consistiria en un increment del 200% de la mida de la partició d'inicialitzacions físiques de la memòria, el bootloader de segona etapa, i la partició NVS. La taula de particions i l'espai reservat es mantindrien amb la mida inicial ja que no hi hauria necessitat d'incrementar-les, de manera que l'espai restant s'aplicaria a la partició `otadata` que, essent una partició clau pel sistema, s'incrementaria en un 300%.

Això és:

$$MIDA'_{ESPAI_{RESERVAT}} = MIDA_{ESPAI_{RESERVAT}} = 4\,KB$$

Fórmula 24. Equivalència entre la mida incrementada i la mida real de l'espai reservat

$$MIDA'_{BOOTLOADER} = 2 * MIDA_{BOOTLOADER} = 2 * 28\,KB = 56\,KB$$

Fórmula 25. Equivalència entre la mida incrementada i la mida real del bootloader de segona etapa

$$MIDA'_{TAULA_{PARTICIONS}} = MIDA_{TAULA_{PARTICIONS}} = 4\,KB$$

Fórmula 26. Equivalència entre la mida incrementada i la mida real de la taula de particions

$$MIDA'_{PARTICIÓ_{NVS}} = 2 * MIDA_{PARTICIÓ_{NVS}} = 2 * 16\,KB = 32\,KB$$

Fórmula 27. Equivalència entre la mida incrementada i la mida real de la partició NVS

$$MIDA'_{PARTICIÓ_{OTADATA}} = 3 * MIDA_{PARTICIÓ_{OTADATA}} = 3 * 8\,KB = 24\,KB$$

Fórmula 28. Equivalència entre la mida incrementada i la mida real de la partició de dades d'OTA

$$MIDA'_{PARTICIÓ_{PHY\_INIT}} = MIDA_{PARTICIÓ_{PHY\_INIT}} = 4\,KB$$

Fórmula 29. Equivalència entre la mida incrementada i la mida real de la partició d'inicialitzacions de la memòria física

$$MIDA'_{SEGMENT\#0} + MIDA'_{SEGMENT\#63} =$$

$$= MIDA'_{ESPAI\,RESERVAT} + MIDA'_{BOOTLOADER} + MIDA'_{TAULA_{PARTICIONS}} +$$

$$+ MIDA'_{PARTICIÓ_{NVS}} + MIDA'_{PARTICIÓ_{OTADATA}} + MIDA'_{PARTICIÓ_{PHY_{INIT}}} =$$

$$= 4\,KB + 56\,KB + 4\,KB + 32\,KB + 24\,KB + 4\,KB = 128\,KB$$

Fórmula 30. Mida incrementada i desglossada del primer i últim segment del disc

Tal com s'ha esmentat amb anterioritat, les particions OTA es mantindrien amb la mateixa mida. Si més no, un cop aplicades les millores anteriors, el primer i l'últim segment de la memòria interna quedarien de la manera següent:

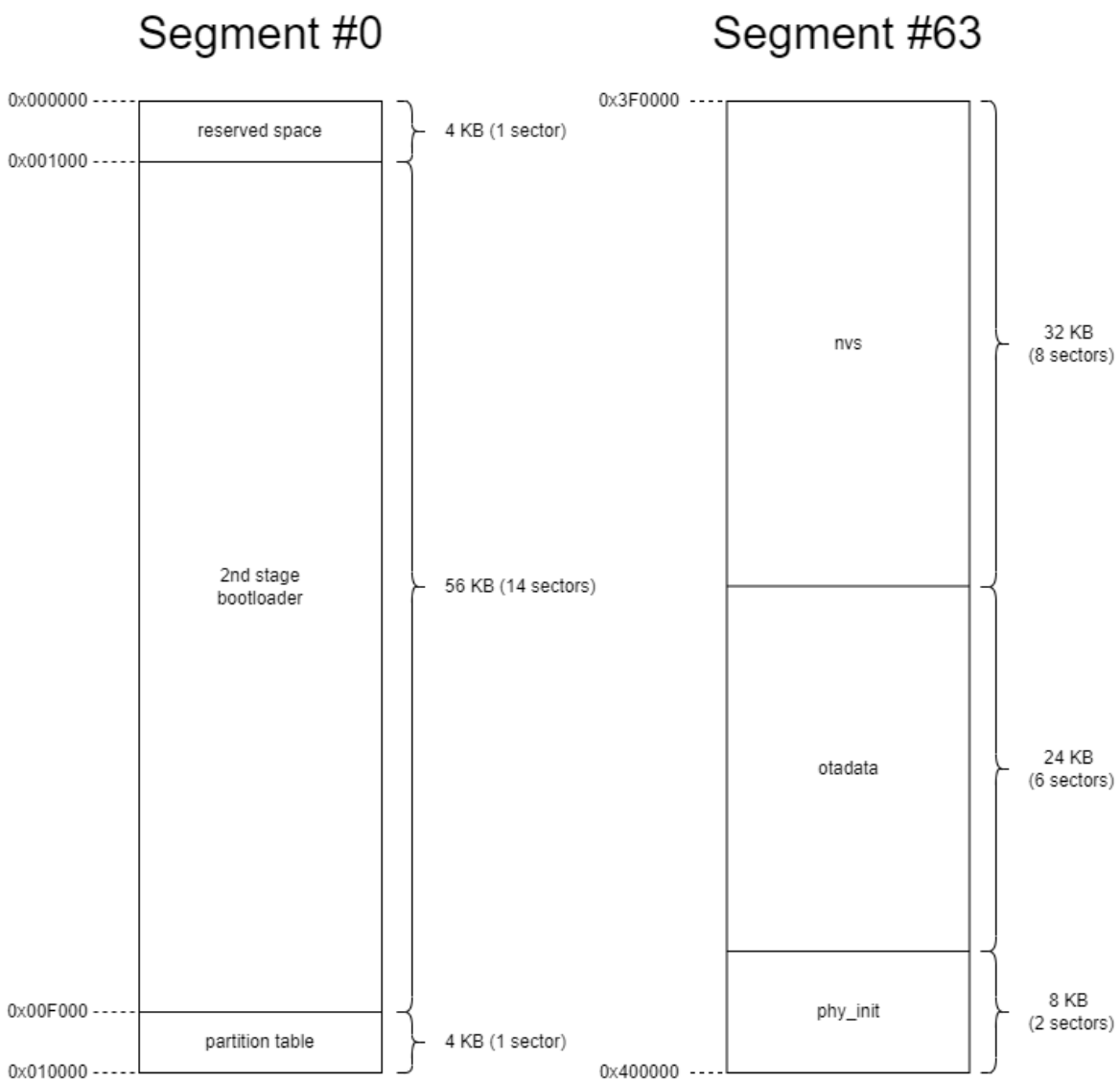

Figura 23. Diagrama del primer i últim segment amb les particions incrementades

4.2 Gestió de la comunicació

De la fase de recollida de requisits del sistema s'obté el següent diagrama d'estats, que mostra una visió general del funcionament del bootloader, on cada estat representa un mode de funcionament del dispositiu:

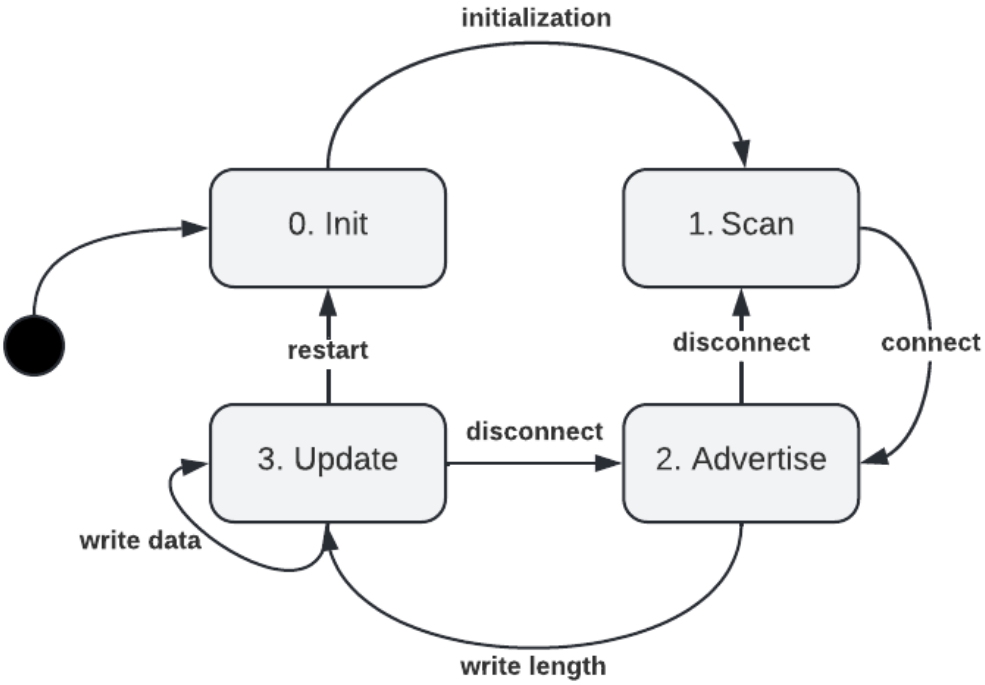

Figura 24. Diagrama d'estats del funcionament del bootloader

4.2.1 Descripció dels modes de funcionament

Així doncs, prenent l'anterior diagrama com a referència, es procedeix amb la descripció dels modes de funcionament:

Nº	Estat	Mode de funcionament	Descripció
0	Init	Mode d'inicialització	El sistema inicialitza el hardware i la interfície Bluetooth.
1	Scan	Mode d'escaneig	El sistema cerca dispositius propers amb els quals establir una connexió.
2	Advertise	Mode de publicació	El sistema està fent pública la informació especificada per GATT.
3	Update	Mode d'actualització	El sistema continua publicant però es troba en mig del procés d'actualització del firmware.

Taula 6. Descripció dels modes de funcionament del dispositiu

4.2.2 Interacció entre les capes

Tal com s'ha introduït al marc teòric, les principals capes de l'arquitectura de BLE implicades en el desenvolupament del bootloader són el perfil d'accés genèric (GAP), el perfil d'atributs genèrics (GATT) i el protocol d'atributs (ATT). A més a més, a les capes anteriors s'hi afegeix la capa d'aplicació (APP), corresponent al propi firmware del dispositiu.

A continuació, es procedeix amb l'explicació detallada d'aquestes capes [10] i com interactuen entre elles per complir l'objectiu d'actualitzar el firmware del dispositiu.

4.2.2.1 Protocol d'atributs (ATT)

ATT és la capa encarregada d'exposar les dades del servidor per al client en forma de paquets. En concret, ATT s'enfoca en gestionar els paquets entrants i sortints en funció de l'especificació de GATT.

En aquest punt, entra en joc un factor clau: la unitat de transferència màxima o MTU, de l'anglès *Maximum Transfer Unit*. La MTU determina la longitud màxima d'un paquet d'ATT i té per defecte un valor de 23 bytes. Per tal d'entendre la seva funció, a continuació es pot observar el següent diagrama:

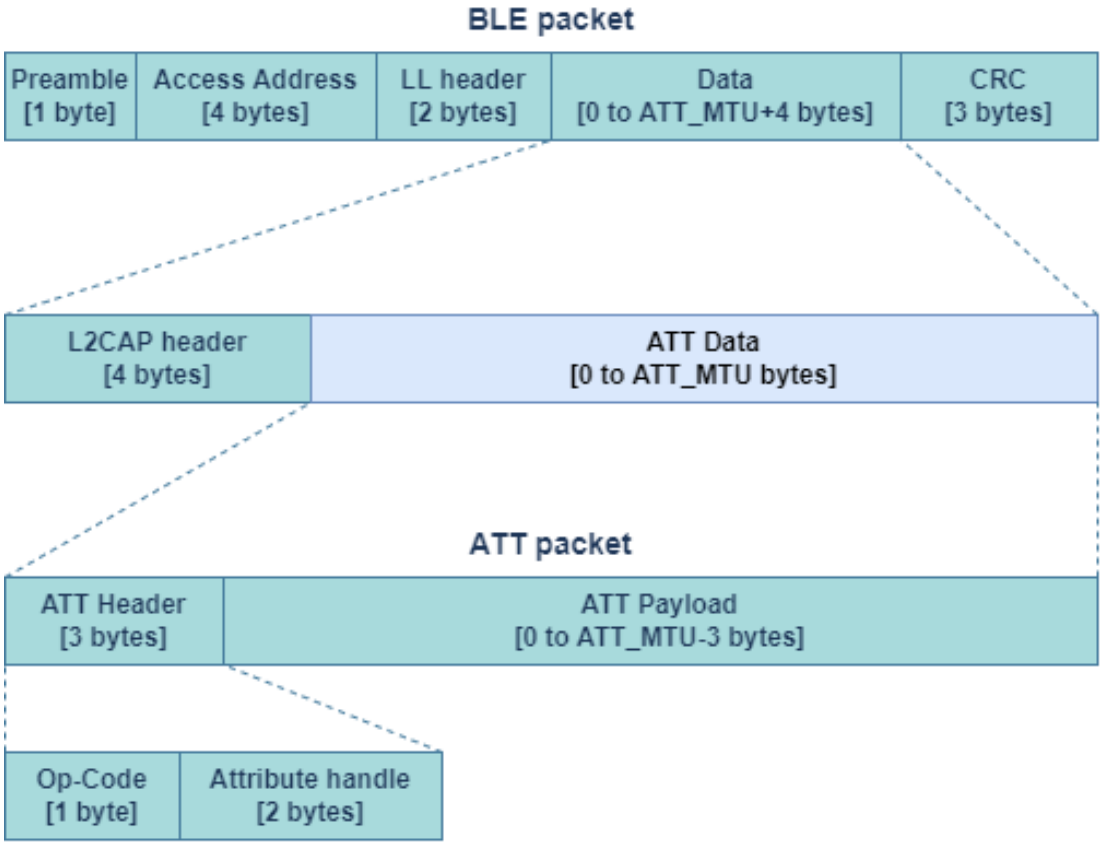

Figura 25. Estructura d'un paquet d'ATT dins d'un paquet de BLE

Davant d'aquesta situació, es troba que cal determinar una mida adequada per tal de reduir l'*overhead* de cada paquet de BLE al mínim. A més a més, cal tenir en compte que interessa que la càrrega útil o *payload* de l'ATT sigui el més gran possible [11].

Així doncs, abans de procedir amb aquesta determinació, es consideren les següents fórmules:

$$MIDA_{CAPÇALERA_{ATT}} = 3\ bytes$$

Fórmula 31. Mida de la capçalera d'un paquet d'ATT

$$MIN\left(MIDA_{ATT_{MTU}}\right) = MTU_{MIN} = 23\ bytes$$

$$MAX\left(MIDA_{ATT_{MTU}}\right) = MTU_{MAX} = 517\ bytes$$

Fórmula 32. Mida mínima i mida màxima de la MTU

Donat que el component ESP32 no limita la mida màxima dels atributs, es presenten dues opcions: establir una mida màxima d'atribut inferior a la mida de la MTU o establir una mida màxima d'atribut superior a la mida de la MTU.

En cas que un atribut fos més gran que la MTU, els missatges es fragmentarien de manera que s'executarien diversos esdeveniments d'escriptura (*write event*) seguits d'un esdeveniment d'execució d'escriptura (*execute write event*), el qual resultaria en un increment del tràfic de la comunicació.

Per tant, donat que l'anterior s'estén més enllà del focus d'aquest projecte i que l'objectiu és tenir una comunicació eficient, s'estableix una mida màxima d'atribut menor que la MTU.

En concret, de cara a la gestió de la memòria interna (veure Secció 4.1) convé que la mida dels atributs sigui potència de 2, ja que representa la mida dels missatges de BLE que encapsulen un fragment d'una imatge del nou firmware del dispositiu.

Això és:

$$MAX(MIDA_{ATRIBUT}) = \{\max(x) \mid (x \in \{MTU_{MIN}, MTU_{MIN}+1, \ldots, MTU_{MAX}\}) \wedge$$

$$\wedge (n \in \mathbb{N}) \wedge (\mathrm{x} \bmod 2^n = 0)\} = 512\ bytes$$

Fórmula 33. Mida màxima d'un atribut de BLE

De l'anterior, s'obté que la mida adequada de la MTU és la següent:

$$MIDA_{ATT_{MTU}} = MTU = MAX(MIDA_{ATRIBUT}) + MIDA_{CAPÇALERA_{ATT}} =$$

$$= 512\ bytes + 3\ bytes = 515\ bytes$$

Fórmula 33. Mida de la MTU

Aleshores, un cop definides les mides anteriors, es procedeix amb els càlculs dels *overheads* dels diferents paquets d'ATT per comprovar la seva reducció:

$$OVERHEAD_{INICIAL} = \frac{MIDA_{CAPÇALERA_{ATT}}}{MTU_{MIN}} = \frac{3\ bytes}{23\ bytes} \cong 13{,}0\%$$

Fórmula 34. *Overhead* d'un paquet d'ATT amb la MTU predeterminada

$$OVERHEAD_{FINAL} = \frac{MIDA_{CAPÇALERA_{ATT}}}{MTU} = \frac{3\ bytes}{515\ bytes} \cong 0{,}6\%$$

Fórmula 35. *Overhead* d'un paquet d'ATT amb la MTU incrementada

$$REDUCCIÓ_{OVERHEAD} = |OVERHEAD_{FINAL} - OVERHEAD_{INICIAL}| =$$

$$= |0{,}6\% - 13{,}0\%| \cong 12{,}4\%$$

Fórmula 36. Reducció de l'*overhead* d'un paquet d'ATT

4.2.2.2 Perfil d'atributs genèrics (GATT)

GATT és la capa encarregada d'estructurar les dades que s'exposen del servidor al client en forma d'atributs, i aquests s'agrupen en característiques, que al mateix temps s'agrupen en serveis.

Així doncs, seguint aquest model i per tal de complir amb el requisit del sistema #SYS_078 (veure Secció 3.1) es crea un servei genèric de transferència d'objectes. Aquest servei genèric s'identifica amb un UUID[11] de 16 bits [12], a diferència dels serveis personalitzats que s'identifiquen amb un UUID de 128 bits.

Realment, a efectes pràctics tots els UUID són de 128 bits, dels quals 112 són iguals per a tots els serveis genèrics. El mateix raonament s'aplica també per a les característiques.

11 Un identificador únic universal, de l'anglès *Universally Unique IDentifier*, és un codi identificador estàndard emprat en el desenvolupament de software per poder distingir un objecte dins d'un o diversos sistemes.

Més específicament, un UUID de 16 bits està integrat dins d'un UUID de 128 bits com segueix, on XXXX és l'UUID de 16 bits:

```
0000XXXX-0000-1000-8000-00805F9B34FB
```

Al mateix temps, el servei de transferència d'objectes agrupa dues característiques de tipus text. Tot l'anterior queda especificat a continuació:

Tipus	UUID	Nom oficial	Etiqueta	Descripció
Servei	0x1825	Object Transfer	OBJECT_TRANSFER	Encapsula les dues característiques del servei.
Característica	0x2AF7	Fixed String 36	OBJECT_TRANSFER_A	Encapsula un missatge de 512 bytes que representa un fragment de la imatge del nou firmware.
Característica	0x2AF8	Fixed String 8	OBJECT_TRANSFER_B	Encapsula la mida en bytes de la imatge del nou firmware.

Taula 7. Descripció dels serveis i característiques de GATT

4.2.2.3 Perfil d'accés genèric (GAP)

GAP és la capa encarregada de gestionar la connexió i la publicació de missatges de BLE. No obstant, degut a què s'estén més enllà del focus d'aquest projecte, es considera innecessari fer èmfasi en aquesta capa a nivell de disseny.

La gestió de GAP es fa a través de l'API d'Espressif Systems i, per tant, és un tema relacionat amb la implementació de l'aplicació (veure Secció 5.4).

4.2.2.4 Capa d'aplicació (APP)

La capa d'aplicació representa una abstracció de les dades d'aplicació, les quals es gestionen principalment a través de l'API d'Espressif Systems o directament des del propi codi font (veure Secció 5.5).

Malgrat això, a continuació es detallen les parts més rellevants a nivell de disseny en el procés d'actualització del firmware.

En primer lloc, s'estableix la mida dels missatges de BLE entrants a la característica OBJECT_TRANSFER_B, corresponents al valor en format de text de la mida de la imatge del nou firmware.

$$MIDA_{OBJECT\_TRANSFER\_B} = 8\ bytes$$

Fórmula 37. Mida de la característica OBJECT_TRANSFER_B

En segon lloc, s'estableix la mida dels missatges de BLE entrants a la característica OBJECT_TRANSFER_A, corresponents a un fragment de 512 bytes de la imatge del nou firmware.

$$MIDA_{OBJECT\_TRANSFER\_A} = MAX(MIDA_{ATRIBUT}) = 512\ bytes = 1\ missatge$$

Fórmula 38. Mida de la característica OBJECT_TRANSFER_A

Per últim, s'estableix la mida del buffer on s'emmagatzemaran els missatges per ser posteriorment gravats a la memòria interna del dispositiu.

$$\exists\ MIDA_{BUFFER} \Leftrightarrow \forall\ 2^n \mid n\ \in \{2,3,\dots,6\}\ KB$$

$$MIDA_{BUFFER} = 1\ buffer = 4096\ bytes = 4\ KB$$

Fórmula 39. Mida del buffer d'escriptura a la memòria interna

La mida escollida per al buffer d'escriptura és 4 KB perquè és la mida d'un sector i aquest és la divisió mínima de la memòria interna (veure Secció 4.1).

Per tant, el procés d'actualització consisteix en l'acumulació de 8 missatges de 512 bytes en un buffer de 4 KB que quan s'omple es grava a l'adreça de memòria corresponent.

$$\frac{N_{MISSATGES}}{1\ buffer} = \frac{MIDA_{BUFFER}}{MIDA_{OBJECT_{TRANSFER_A}}} = \frac{4096\ bytes}{512\ bytes} = 8\frac{missatges}{buffer}$$

Fórmula 40. Nombre de missatges per buffer

Així doncs, a continuació es representa gràficament el procés d'actualització des de l'origen dels missatges de BLE fins que es fa una gravació al disc:

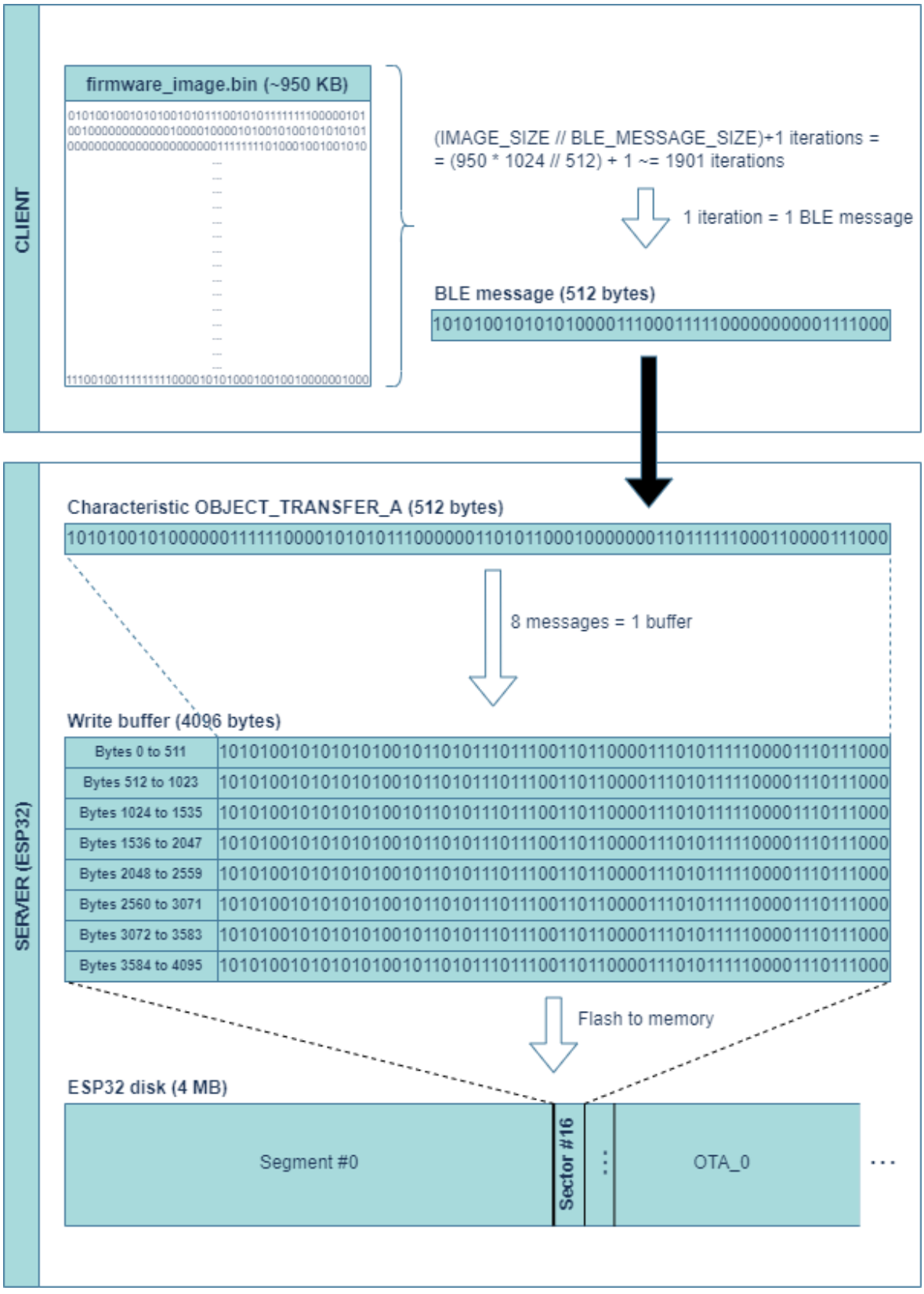

Figura 26. Diagrama del procés d'actualització del firmware del dispositiu

5 Desenvolupament

5.1 Mapa de ruta

Aquest projecte, com qualsevol altre de l'enginyeria del software, ha passat per diverses fases per poder-se realitzar. Cal tenir en compte que la durada d'aquestes fases està directament condicionada per la influència de la càrrega de treball de tasques alienes al projecte present.

Així doncs, a continuació es pot observar l'evolució de les diverses fases del projecte en funció del temps:

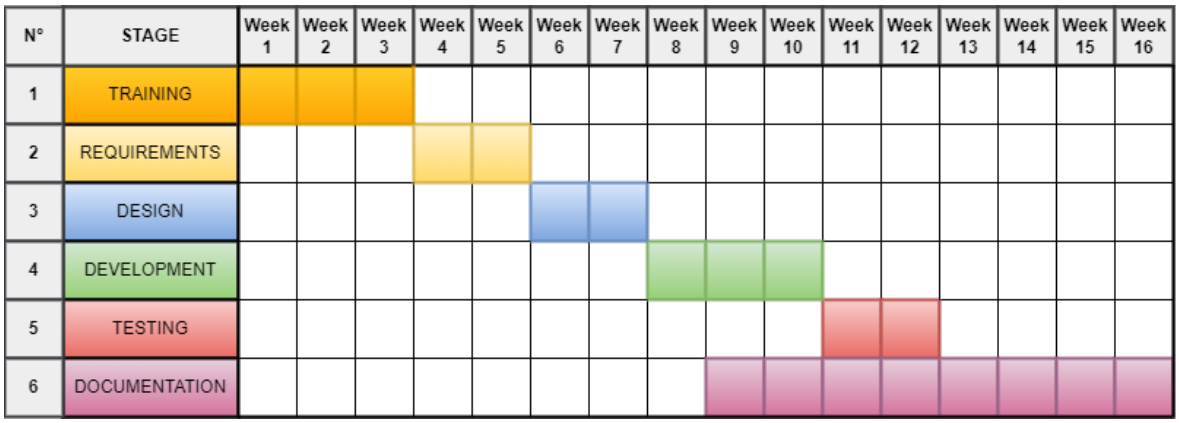

Figura 27. Diagrama de Gantt del projecte

5.2 Arquitectura de software i disseny d'alt nivell

L'arquitectura de software està basada en l'arquitectura AUTOSAR [13], la qual distingeix en el nivell més alt d'abstracció entre tres capes de software que s'executa en un microcontrolador: aplicació, entorn en temps d'execució o RTE[12], i software bàsic o BSW[13].

Al mateix temps, l'arquitectura AUTOSAR també divideix la capa de BSW en tres subcapes, a banda dels controladors de dispositius complexos (CDD): abstracció de serveis (SAL), abstracció de l'ECU (ECUAL) i abstracció del microcontrolador (MCAL).

[12] Un entorn en temps d'execució o RTE, de l'anglès *RunTime Environment*, és un software que proveeix serveis per a un programa en temps d'execució, sense formar part del sistema operatiu.

[13] La capa de software bàsic o BSW, de l'anglès *Basic SoftWare*, és una capa d'abstracció de dades inferior al RTE que dona suport a la capa d'aplicació en la comunicació amb els perifèrics del microcontrolador.

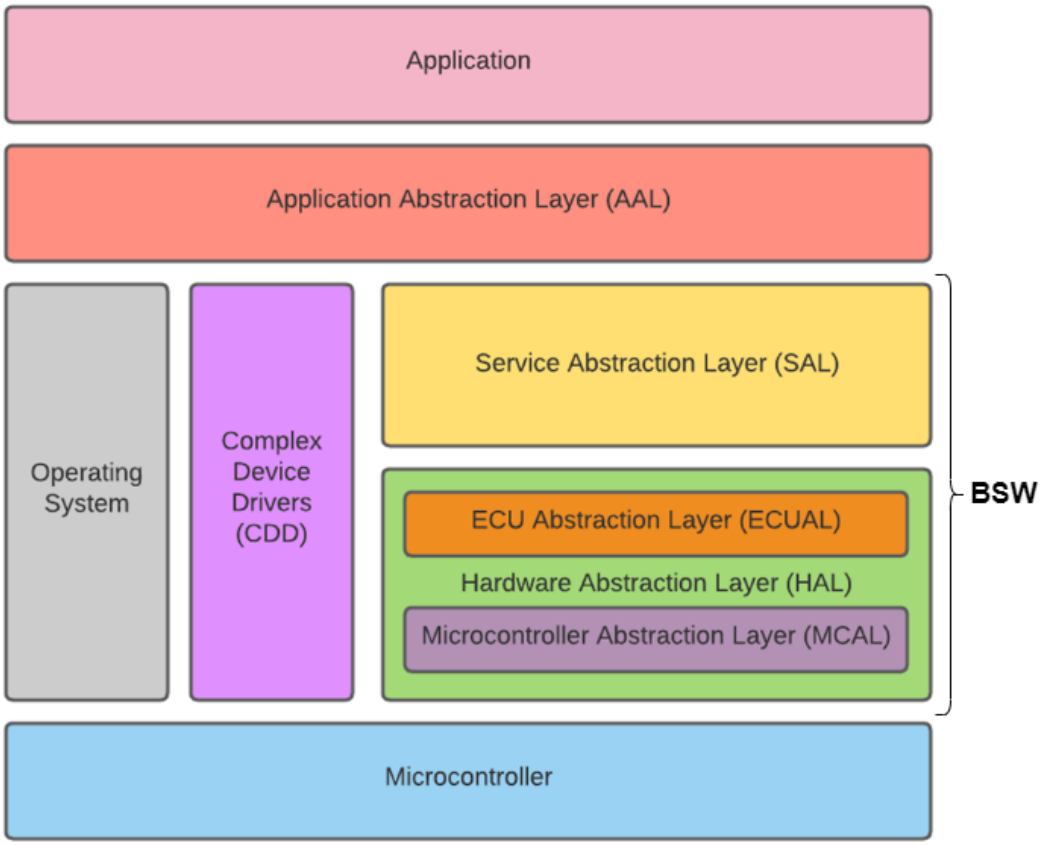

Figura 28. Diagrama de l'arquitectura de software AUTOSAR

No obstant, l'arquitectura AUTOSAR està orientada a sistemes molt més complexes que el cas del projecte present. Per tant, amb l'objectiu d'adaptar aquesta arquitectura per a un dispositiu més simple, algunes parts d'aquestes queden omeses.

En concret, la capa de RTE queda omesa, dividint el software en les dues capes d'abstracció restants, les quals queden reflectides en l'estructura de fitxers del projecte: aplicació i BSW (veure Secció 5.3). Així mateix, les subcapes SAL i ECUAL també queden omeses.

Aquesta arquitectura simplificada té l'objectiu de satisfer dos principis de disseny de software molt importants en els sistemes encastats: el principi de modularitat i el principi d'encapsulació.

Per un costat, la capa d'aplicació és pràcticament independent del hardware i, per tant, satisfà el principi de modularitat, ja que les dades queden separades en subsistemes.

Per l'altre, la capa BSW actua com a interfície de comunicació entre la capa d'aplicació i el propi microcontrolador, ja que la MCAL conté el *framework* ESP-IDF (veure Secció 2.5) i els CDD s'encarreguen de la comunicació amb els perifèrics. Per tant, aquesta satisfà el principi d'encapsulació.

Així doncs, després d'adaptar l'arquitectura de software al context del projecte, es procedeix amb una visió més específica del mateix, tal com es pot observar en el següent diagrama:

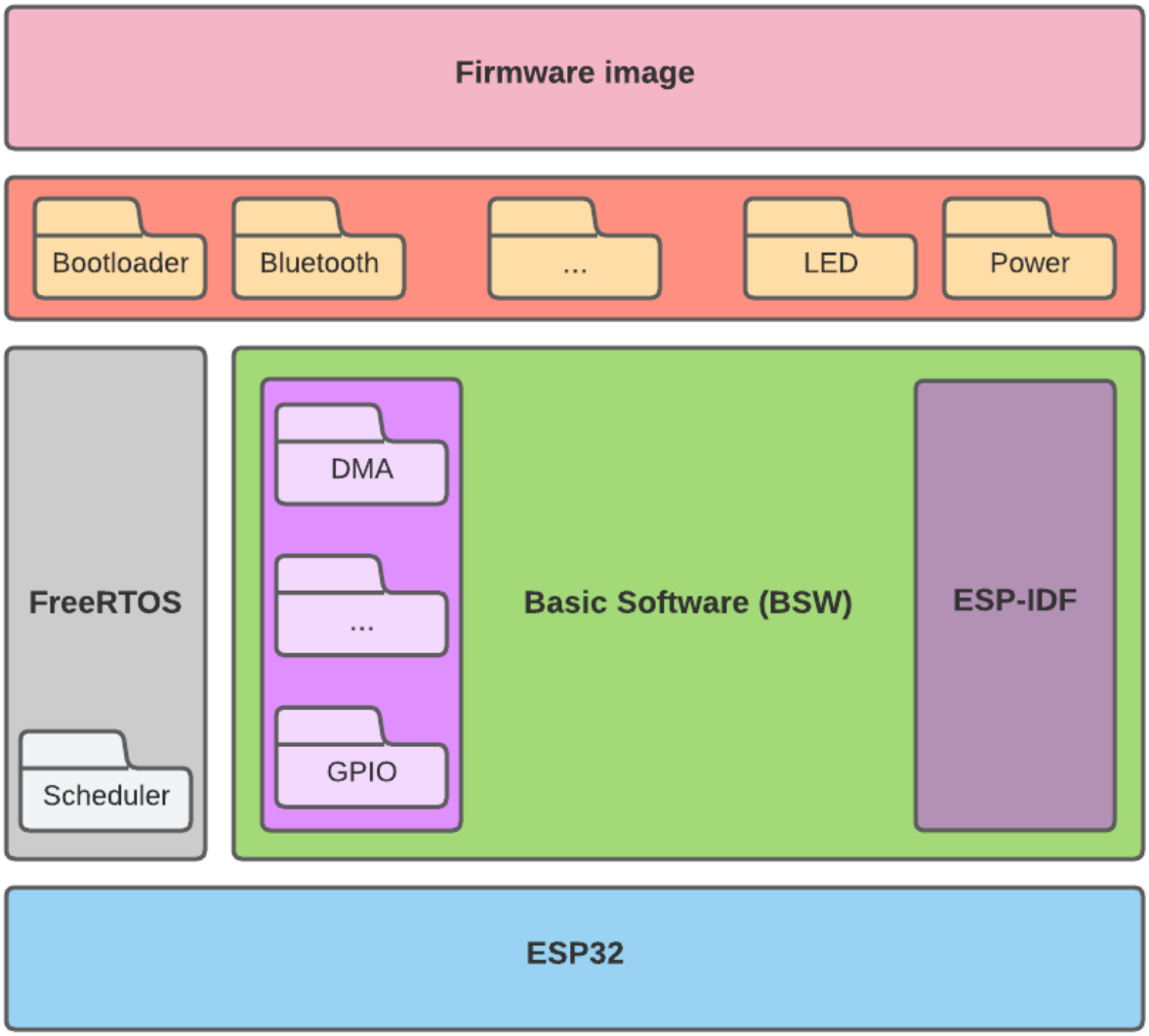

Figura 29. Diagrama d'arquitectura de software del dispositiu

5.3 Estructura de fitxers del projecte

L'estructura de fitxers del projecte està pensada per respectar el màxim possible l'organització per capes proposada per l'arquitectura de software.

A continuació, en la següent figura es pot apreciar detalladament l'estructura, amb els fitxers corresponents als components de software principals (veure Secció 5.4) desplegats:

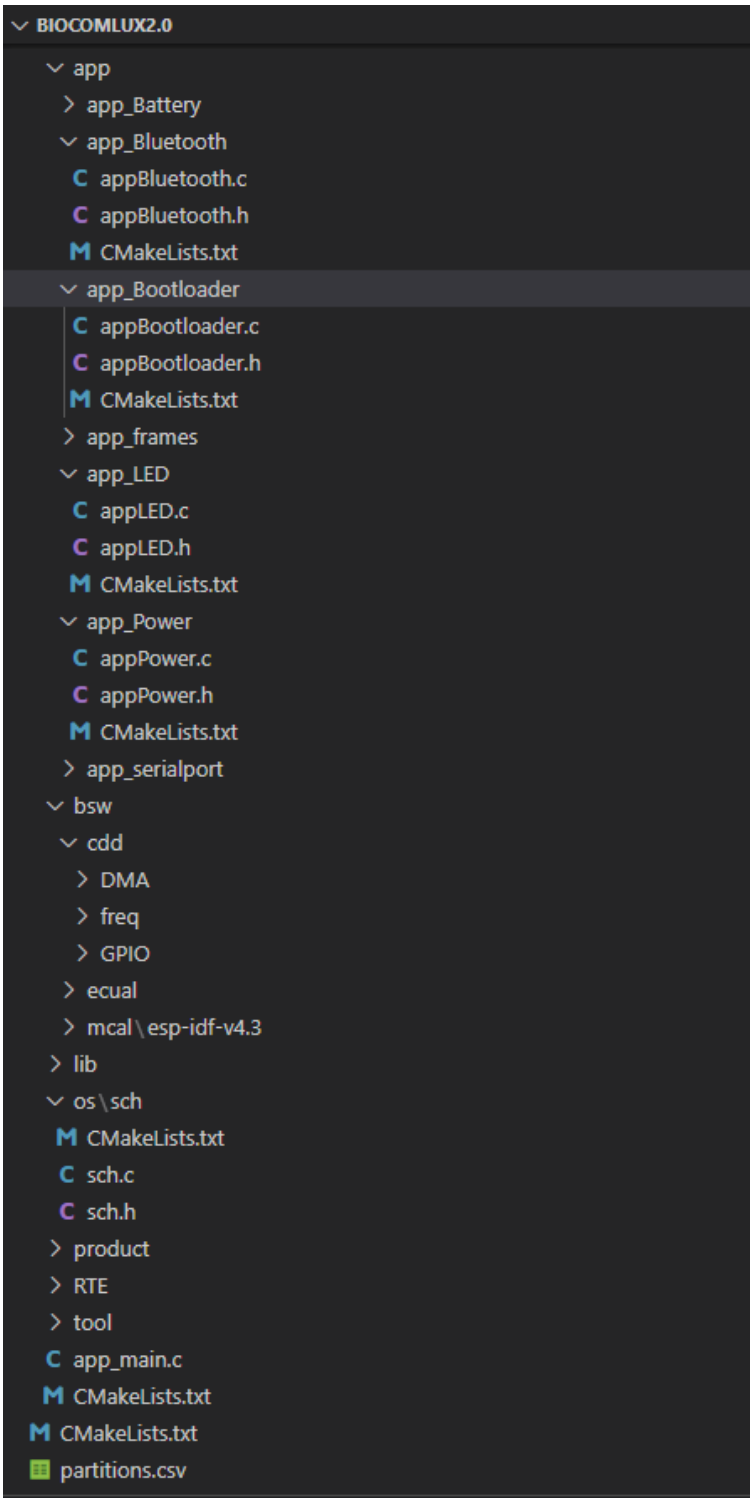

Figura 30. Estructura de fitxers del projecte

5.4 Components de software

Seguint l'estructura dels fitxers del projecte, aquests queden encapsulats en forma de components de software. En concret, a continuació es mostren aquells components que tenen una implicació directa amb el bootloader, organitzats per capes:

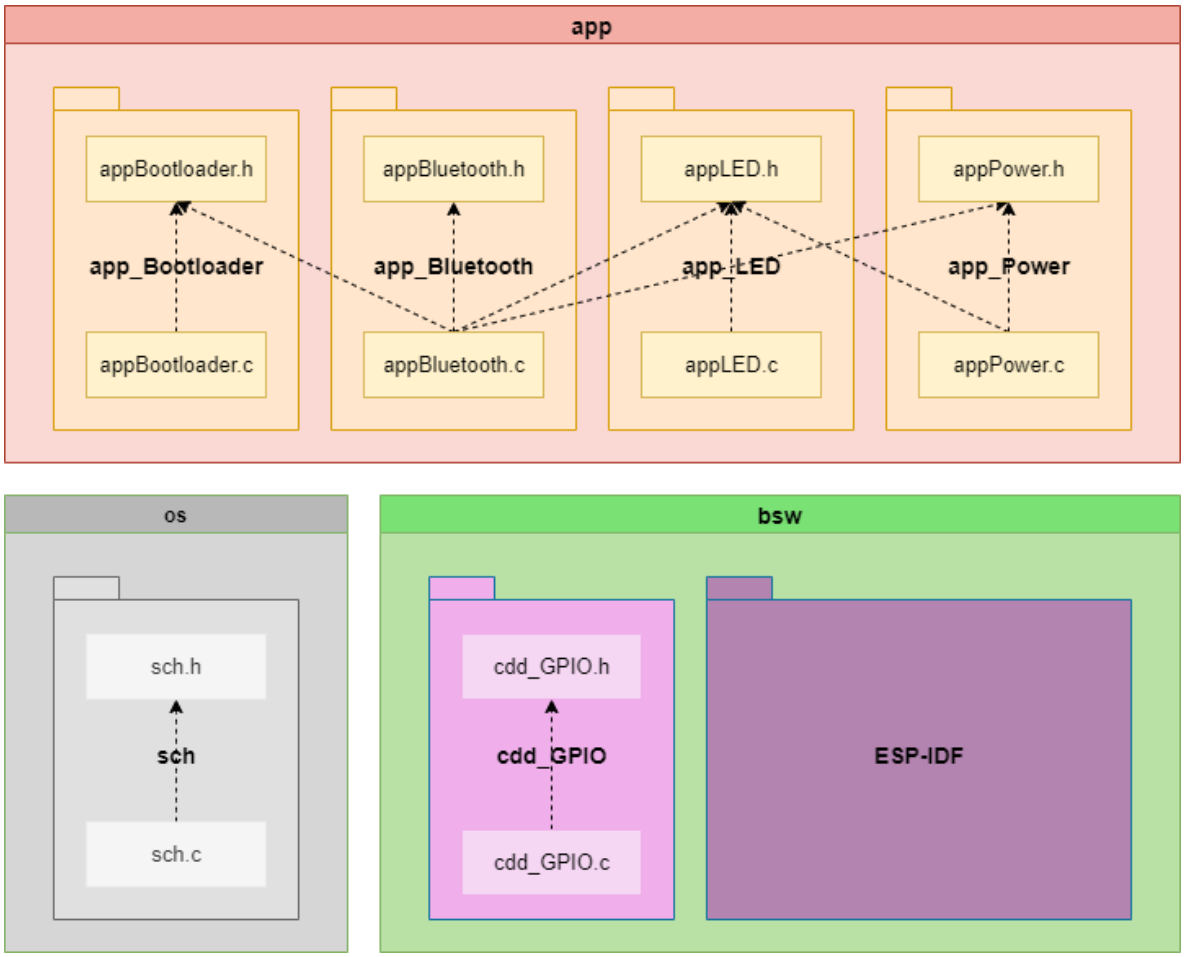

Figura 31. Diagrama dels components de software

Els components pertanyents a la capa d'aplicació corresponen als especificats al diagrama de classes (veure Secció 3.3). Així mateix, n'hi ha tres components més sobre els quals cal emfatitzar la seva funció: el planificador de tasques, el controlador dels pins GPIO[14] i el propi *framework* ESP-IDF.

En total, es consideren 7 components de software, que queden organitzats com segueix:

5.4.1 Components del sistema operatiu

5.4.1.1 Planificador de tasques

El planificador de tasques s'encarrega de gestionar els recursos per a l'execució de tasques periòdiques que, generalment, s'executen cada 10 o cada 100 ms. Principalment, aquestes tasques periòdiques estan relacionades amb l'encès del dispositiu, la bateria o la il·luminació del LED. Si bé aquestes no tenen una relació directa amb el bootloader, ja que

[14] Una entrada/sortida de propòsit general o GPIO, de l'anglès *General Purpose Input/Output*, és un pin configurable en temps d'execució.

aquest és dirigit per esdeveniments (*event driven*), cal tenir constància d'aquestes per a una gestió òptima del dispositiu.

5.4.2 Components de la capa BSW

5.4.2.1 Controlador dels pins GPIO

El controlador dels pins GPIO s'encarrega de proporcionar una interfície per a la configuració dels pins GPIO que s'utilitza principalment en els components LED i Power.

5.4.2.2 ESP-IDF

El *framework* ESP-IDF també s'encarrega de proporcionar una interfície per a la gran majoria de funcionalitats del projecte. Així doncs, aquest conté llibreries especificades a l'API d'Espressif Systems que s'inclouen en tots els components.

5.4.3 Components de la capa d'aplicació

5.4.3.1 Power

El component Power s'encarrega de la gestió de l'encès i l'apagat del dispositiu. Si bé no té una relació directa amb el bootloader, aquest component fa possible el compliment del requisit `#SYS_007`.

5.4.3.2 LED

El component LED s'encarrega de la gestió del LED d'estat del dispositiu. Si bé no té una relació directa amb el bootloader, aquest component fa possible el compliment del requisit `#SYS_008`.

5.4.3.3 Bluetooth

El component Bluetooth s'encarrega de la gestió de totes les funcionalitats implicades en la tecnologia BLE. Aquest component sí que té una relació directa amb el bootloader (veure Secció 5.5) i fa possible el compliment dels requisits, `#SYS_078`, `#SYS_081` i `#SYS_101`.

5.4.3.4 Bootloader

El component Bootloader s'encarrega de la gestió del procés d'actualització OTA, inclosa la gestió de la memòria interna. Aquest component, tal com el seu nom indica, conté la lògica del bootloader en sí (veure Secció 5.4) i fa possible el compliment dels requisits, `#SYS_079`, `#SYS_080`, `#SYS_082`, `#SYS_083` i `#SYS_084`.

5.5 Especificació de les funcions

Seguint una metodologia de disseny descendent[15], s'han implementat diverses funcions per a gestionar adequadament el procés d'actualització OTA. Davant d'aquesta situació, aquestes funcions es distingeixen entre dos tipus: funcions principals i funcions auxiliars.

S'entén per funció principal aquella que té un paper imprescindible en el procés d'actualització i que, a més a més, es crida des d'una part del codi aliena al component del bootloader.

En canvi, s'entén per funció auxiliar aquella que té un paper secundari en el procés d'actualització i que, a més a més, es crida des d'una funció principal.

5.5.1 Funcions principals

5.5.1.1 Funció ota_init()

La funció `ota_init()` inicialitza les variables necessàries i comprova el correcte estat de les particions per al procés d'actualització OTA. Aquesta funció es crida només en el primer esdeveniment d'escriptura de la característica OBJECT_TRANSFER_A.

Paràmetre	Direcció	Tipus	Rang	Descripció
-	-	`void`	-	No es rep cap paràmetre.

Taula 8. Definició de paràmetres de la funció ota_init()

Tipus	Rang	Descripció
`void`	-	No es retorna cap valor.

Taula 9. Definició del retorn de la funció ota_init()

5.5.1.2 Funció ota_flash()

La funció `ota_flash()` acumula un missatge de BLE en forma de vector de bytes a un buffer de dades que, quan s'omple, es grava el seu contingut a la memòria interna del dispositiu. Aquesta funció es crida en tots els esdeveniments d'escriptura de la característica OBJECT_TRANSFER_A.

[15] El disseny descendent o top-down és una metodologia de la programació que consisteix en la divisió d'un problema en subproblemes més petits

Paràmetre	Direcció	Tipus	Rang	Descripció
`data`	Entrada	`uint8_t *`	-	Missatge de BLE en forma de vector de bytes.
`data_length`	Entrada	`uint16_t`	512	Longitud del missatge.

Taula 10. Definició de paràmetres de la funció ota_flash()

Tipus	Rang	Descripció
void	-	No es retorna cap valor.

Taula 11. Definició del retorn de la funció ota_flash()

5.5.1.3 Funció ota_finish()

La funció `ota_finish()` finalitza el procés d'actualització OTA, validant la imatge del firmware i seleccionant la partició d'arrancada. Aquesta funció es crida només en l'últim esdeveniment d'escriptura de la característica OBJECT_TRANSFER_A.

Paràmetre	Direcció	Tipus	Rang	Descripció
-	-	`void`	-	No es rep cap paràmetre.

Taula 12. Definició de paràmetres de la funció ota_finish()

Tipus	Rang	Descripció
`void`	-	No es retorna cap valor.

Taula 13. Definició del retorn de la funció ota_finish()

5.5.1.4 Funció ota_cancel()

La funció `ota_cancel()` reinicialitza les variables necessàries en cas d'interrupció del procés d'actualització OTA, mantenint l'estat inicial del dispositiu. Aquesta funció es crida només en el cas que hi hagi un esdeveniment de desconnexió.

Paràmetre	Direcció	Tipus	Rang	Descripció
-	-	`void`	-	No es rep cap paràmetre.

Taula 14. Definició de paràmetres de la funció ota_cancel()

Tipus	Rang	Descripció
`void`	-	No es retorna cap valor.

Taula 15. Definició del retorn de la funció ota_cancel()

5.5.2 Funcions auxiliars

5.5.2.1 Funció ota_restart()

La funció `ota_restart()` mostra un missatge per consola i reinicia el dispositiu. Aquesta funció es crida des de les funcions principals.

Paràmetre	Direcció	Tipus	Rang	Descripció
-	-	`void`	-	No es rep cap paràmetre.

Taula 16. Definició de paràmetres de la funció ota_restart()

Tipus	Rang	Descripció
`void`	-	No es retorna cap valor.

Taula 17. Definició del retorn de la funció ota_restart()

6 Avaluació

Un cop finalitzades les fases de requisits, disseny i desenvolupament, cal realitzar una avaluació de la solució proposada. Així doncs, una vegada verificat el funcionament correcte del dispositiu, n'hi ha dos temes addicionals per tractar: l'avaluació del temps d'execució i l'avaluació del tractament d'errors.

6.1 Temps d'execució

En aquest punt, n'hi ha dos factors crucials en l'avaluació dels resultats, ja que són potencials colls d'ampolla[16] en l'execució del procés d'actualització: la mida de missatge de BLE i la mida del buffer d'escriptura.

Tot i això, ambdós factors són independents de la correctesa de l'execució (veure Secció 9.2), és a dir, tot i que les mides dels missatges i del buffer no fossin òptimes, l'actualització del firmware s'hauria de completar, si més no, en un període de temps major que el desitjat.

De fet, tot i que no s'especifica com a requisit del sistema, es determina que s'entén per solució òptima aquella que garanteix la correctesa de l'execució en un temps proper als 5 minuts, per a l'enviament d'una imatge de la mida establerta als requisits del sistema (veure Secció 3.1), és a dir, aproximadament 950 KB.

Si bé d'entrada l'enviament de 950 KB en 5 minuts pot semblar lent, cal tenir present que aquesta transferència de dades es realitza a través de Bluetooth de baixa energia i, per tant, a fi de poder tenir consums de bateria molt baixos, les velocitats de transferència també són baixes.

6.1.1 Mida de missatge

Lògicament, cal prioritzar els processos que s'executen més vegades, ja que en cas d'errors, l'execució es podria alentir precisament per la repetició d'aquests. Així doncs, el punt crític del procés d'actualització és la mida de missatge de BLE.

Per tant, amb la fi de corroborar que les decisions de disseny són les més òptimes per a aquesta solució, es procedeix a estudiar la influència de la mida dels missatges de BLE que s'escriuen a la característica OBJECT_TRANSFER_A en el temps d'execució del procés d'actualització.

Així mateix, cal tenir present que l'estudi que s'exposa a continuació es tracta d'una solució provisional, degut a que no ha estat possible realitzar l'avaluació de la solució proposada amb l'aplicació mòbil oficial del client, ja que aquest implementarà la funcionalitat d'actualitzar el dispositiu en una data posterior a la del desenvolupament d'aquest Treball de Fi de Grau.

16 Un coll d'ampolla o *bottleneck* és un concepte emprat per referir-se a una part d'un procés que frena o alenteix l'execució d'aquest.

En conseqüència, davant la impossibilitat de dur a terme les proves corresponents mitjançant l'aplicació del client, s'ha optat per implementar un script en Python per simular l'enviament d'una imatge del firmware a través de BLE (veure Secció 9.3).

Per tal de realitzar aquesta simulació, s'executa el script des d'una línia de comandes de Windows PowerShell, mesurant el temps d'execució amb la funció `Measure-Command`.

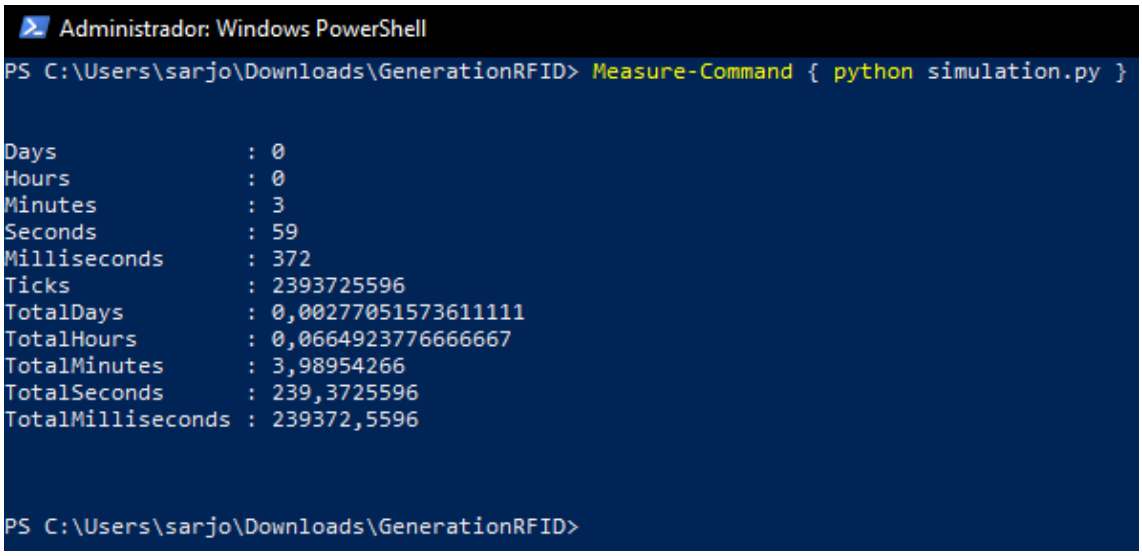

Figura 32. Execució del script de simulació de l'enviament d'una imatge del firmware

Després de realitzar diverses iteracions del script de simulació amb les diferents mides de missatge, s'observa fàcilment que les mides més baixes no són una solució òptima. No obstant, a fi d'analitzar la tendència del temps d'actualització en funció de la mida de missatge, aquestes també s'inclouen en l'estudi.

Tanmateix, després de realitzar més de 10 iteracions amb la mida de missatge establerta pel disseny, s'observa que els resultats no varien més de 5 segons entre totes les iteracions. Per tant, donat que l'objectiu a complir a nivell temporal és de l'ordre dels minuts, es pren el valor de les mitjanes de les iteracions, aproximades a les dècimes.

Així doncs, els resultats obtinguts són els següents:

Mida de missatge (bytes)	Temps (min)
16	82,3
32	41,3
64	26,0
128	13,4
256	7,0
512	4,0

Taula 18. Durada de l'actualització OTA del firmware en funció de la mida de missatge

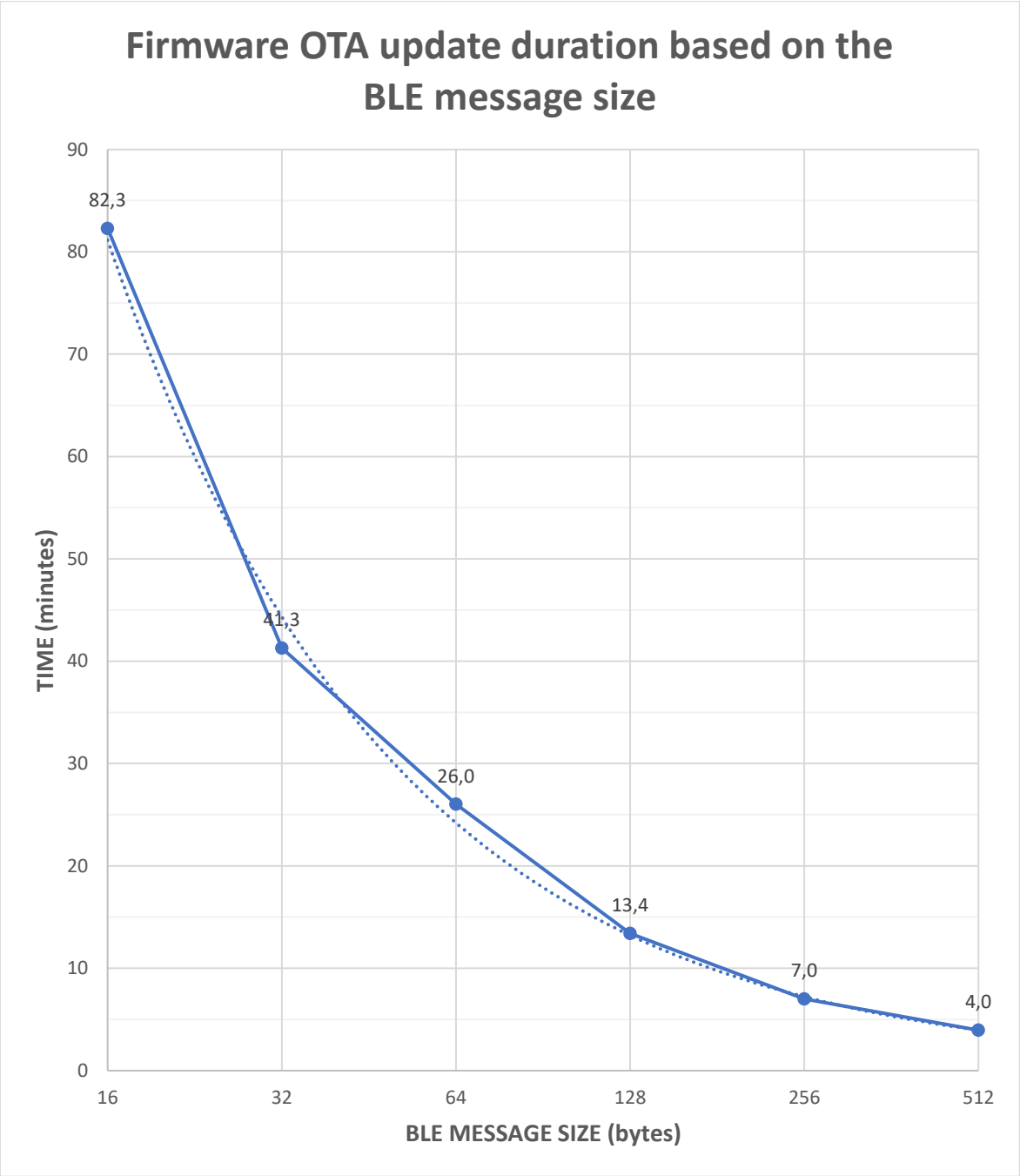

Figura 33. Gràfica comparativa de la durada de l'actualització OTA del firmware en funció de la mida de missatge, en escala logarítmica

Tal com es pot observar en la gràfica anterior, el temps d'execució disminueix de forma inversament proporcional a la mida de missatge, seguint una línia de tendència potencial i amb un temps d'execució resultant d'aproximadament 4 minuts.

6.1.2 Mida del buffer d'escriptura

El segon punt més important del procés d'actualització és la mida del buffer d'escriptura, ja que té una influència directa en el temps de gravació a memòria interna. Per tant, en un intent de buscar possibles millores quant a la durada de l'actualització, es procedeix a realitzar el següent estudi, assumint una mida de missatge de 512 bytes.

Així doncs, els resultats obtinguts són els següents:

Mida del buffer (KB)	Temps (min)
4	4,0
8	4,0
16	4,0
32	4,0
64	4,0

Taula 19. Durada de l'actualització OTA del firmware en funció de la mida del buffer

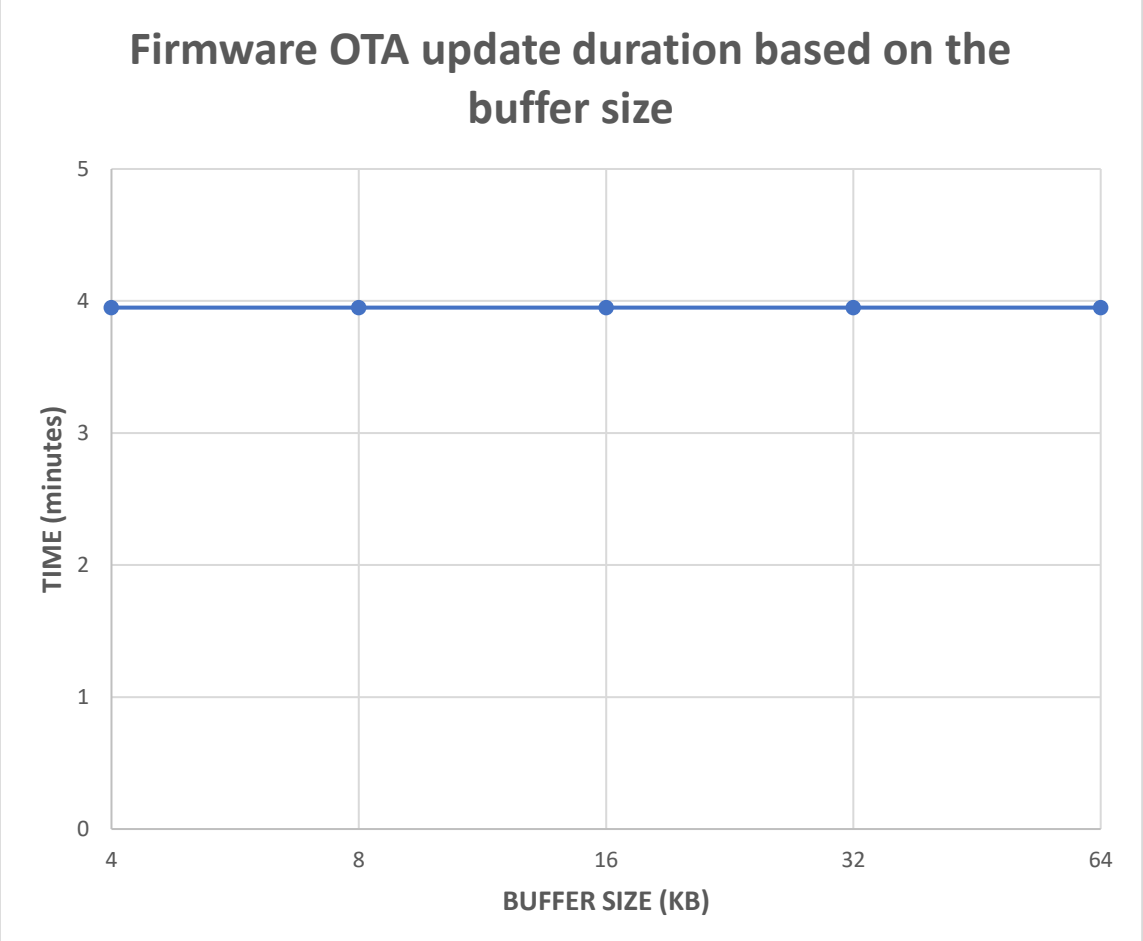

Figura 34. Gràfica comparativa de la durada de l'actualització OTA del firmware en funció de la mida del buffer, en escala logarítmica

Per tant, tal com es pot observar en la taula anterior, es determina que la mida del buffer no té un impacte significatiu en el temps d'actualització.

6.2 Tractament d'errors

El tractament dels possibles errors del procés d'actualització del firmware és bastant simple, en el sentit que generalment aquest consisteix en mostrar un missatge d'error i reiniciar el sistema.

En particular, hi ha tres principals casos d'error a tractar, els quals s'exposen a continuació i es detallen als annexos del present document (veure Secció 9.2):

Nº	Cas d'error	Descripció	Tractament
1	Error d'inicialització	No n'hi ha una partició OTA disponible.	Mostrar un missatge d'error i reiniciar el sistema.
2	Error de desconnexió	Es perd la connexió de BLE durant l'actualització del sistema.	Mostrar un missatge d'error i cancel·lar l'actualització, sense reiniciar el sistema.
3	Error de corrupció	La imatge del firmware és corrupta.	Mostrar un missatge d'error i reiniciar el sistema.

Taula 20. Tractament d'errors en el l'actualització del firmware

7 Conclusions

Per concloure la realització d'aquest projecte, m'agradaria destacar el gran avantatge que suposa aquest per al client final. Gràcies a la implementació del bootloader i el fet d'afegir la funcionalitat de dur a terme actualitzacions del firmware a través de BLE, l'usuari ja no requereix de cap coneixement tècnic per actualitzar el dispositiu.

De fet, en tractar-se d'un dispositiu mòbil, composat de hardware i software, la forma viable de realitzar un manteniment d'aquest és via software. Per tant, aquesta solució també facilita el procés de manteniment, ja que de cara al futur es podran fer desplegaments de noves versions a distància, de manera que l'empresa client podrà afegir aquestes a la seva aplicació mòbil oficial.

D'altra banda, cal mencionar que un dels principals reptes a afrontar ha estat trobar una solució que permeti realitzar actualitzacions en un temps raonable sense que es disparés el consum del dispositiu. Davant d'aquest problema, es va optar per realitzar l'actualització a través de BLE, en comptes de Wi-Fi, amb l'objectiu d'assegurar inicialment un baix consum i, a partir d'aquest punt, intentar optimitzar la velocitat de transferència de dades per tal d'arribar a un temps d'actualització acceptable.

Per tant, considero que aquest projecte ha estat un èxit per tres raons. La primera és l'assoliment dels objectius proposats, fent possible una actualització en menys de 5 minuts i mantenint un baix consum. La segona és l'aprenentatge que m'ha suposat realitzar aquest projecte en els darrers mesos. I la tercera és l'oportunitat d'haver pogut aplicar els coneixements adquirits durant els darrers anys per a la resolució de problemes reals.

A més a més, m'agradaria destacar la importància d'haver realitzat aquest projecte en col·laboració amb Generation RFID. Durant la meva estada a l'empresa he pogut viure i veure com és un projecte real dins del món laboral, participant des de les fases inicials fins a l'enviament del producte final al client.

Gràcies a això, he pogut consolidar coneixements que sentia que no tenia del tot agafats per la mà, com la recollida de requisits i l'especificació d'aquests, seguint els estàndards de l'enginyeria del software, la utilització d'eines d'administració de tasques en forma de tiquets i de sistemes de control de versions, el seguiment de metodologies concretes durant la fase de desenvolupament i, especialment, la fase de *testing*, ja que és l'especialitat de Generation RFID.

Per últim, m'agradaria concloure que aquest projecte ha estat molt important per a mi ja que no només em suposa posar punt i final a una etapa universitària que per motius personals ha estat plena de dificultats, sinó que també implica el compliment d'un repte d'autosuperació que ha estat la meva única font d'energia durant alguns dels darrers anys. Així doncs, espero haver aconseguit plasmar els meus coneixements i haver deixat d'alguna manera la meva marca personal en aquest document.

8 Referències

[3] Generation RFID. (05/2022). *Test & Embedded Electronics Generationrfid Technological Company Reus*. https://generationrfid.com/ [consulta: 29/05/2022]ms. (05/2022). *ESP32 Wi-Fi & Bluetooth MCU | Espressif Systems*. https://www.espressif.com/en/products/socs/esp32 [consulta: 29/05/2022]

[4] Afaneh, M. (2018). *Intro to Bluetooth low energy: The fastest way to learn BLE*. Independently Published.

[5] Townsend, Sawyer, B., Loukides, M. K., Montgomery, K., Futato, D., & Demarest, R. (2014). *Getting started with bluetooth low energy* (First edition.). O'Reilly.

[6] Espressif Systems. (05/2022). *ESP-IDF programming guide - ESP32 - — ESP-IDF programming guide latest documentation*. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html [consulta: 29/05/2022]

[7] Espressif Systems. (2022). *ESP32-WROOM-32 Datasheet* (Versió 3.3) [fitxer PDF]. https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf [consulta: 29/05/2022]

[8] Espressif Systems. (2022). *ESP32 Series Datasheet* (Versió 3.9) [fitxer PDF]. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf [consulta: 29/05/2022]

[9] Espressif Systems. (2021). *ESP32 Tecnhical Reference Manual* (Versió 4.6) [fitxer PDF]. https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf [consulta: 29/05/2022]

[10] Derhgawen, A. (16/11/2020). *Maximizing BLE Throughput Part 4: Everything You Need to Know*. Punch Through. https://punchthrough.com/ble-throughput-part-4/ [consulta: 29/05/2022]

[11] Afaneh, M. (17/06/2020). *Bluetooth 5 speed: How to achieve maximum throughput for your BLE application*. Novel Bits. https://www.novelbits.io/bluetooth-5-speed-maximum-throughput/ [consulta: 29/05/2022]

[12] Bluetooth SIG Propietary. (19/05/2022). 16-bit UUID Numbers Document [fitxer PDF]. https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%20UUID%20Numbers%20Document.pdf [consulta: 29/05/2022]

[13] AUTOSAR Development Cooperation. (2021, December 6). *Classic platform - AUTOSAR*. https://www.autosar.org/standards/classic-platform/ [consulta: 29/05/2022]