



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Facultat d'Informàtica de Barcelona

FIB

# DISSENY I DESENVOLUPAMENT D'UN CONTROLADOR DE VOL, PER A LA SEVA IMPLEMENTACIÓ EN UN COET QUE PARTICIPI A LA COMPETICIÓ EUROPEAN ROCKETRY CHALLENGE

ARNAU MARIEGAS BAROT

#### Director/a

ANTONIO CAMACHO SANTIAGO (Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial)

#### Titulació

Grau en Enginyeria Informàtica (Enginyeria de Computadors)

Memòria del treball de fi de grau

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

22/10/2024



---

## Resum

Aquest treball se centra en el disseny i desenvolupament d'un controlador de vol per a un coet de petita escala, destinat a competir a l'European Rocketry Challenge (EUROC). El projecte sorgeix com una millora del sistema d'altímetre actual utilitzat en els coets del programa UPC Space Program, solucionant problemes de mida i funcionalitat. L'objectiu principal és crear un altímetre capaç d'activar els paracaigudes i recopilar dades de vol, integrant sensors addicionals com el GPS i acceleròmetres, a un cost més assequible que les solucions comercials. A més, el projecte contribueix a documentar el procés de disseny, facilitant futures actualitzacions.

## Resumen

Este trabajo se centra en el diseño y desarrollo de un controlador de vuelo para un cohete de pequeña escala, destinado a competir en la European Rocketry Challenge (EUROC). El proyecto surge como una mejora del sistema de altímetro actual utilizado en los cohetes del programa UPC Space Program, solucionando problemas de tamaño y funcionalidad. El objetivo principal es crear un altímetro capaz de activar los paracaídas y recopilar datos de vuelo, integrando sensores adicionales como el GPS y acelerómetros, a un coste más accesible que las soluciones comerciales. Además, el proyecto documenta el proceso de diseño, facilitando futuras actualizaciones.

## Abstract

This project focuses on the design and development of a flight controller for a small-scale rocket, aimed at competing in the European Rocketry Challenge (EUROC). The project arises as an improvement of the current altimeter system used in rockets from the UPC Space Program, addressing issues of size and functionality. The main goal is to create an altimeter capable of deploying parachutes and collecting flight data, integrating additional sensors such as GPS and accelerometers, at a more affordable cost compared to commercial solutions. Furthermore, the project documents the design process, facilitating future updates.



# Índex

<b>1. Introducció</b>	<b>9</b>
1.1. Terminologia i definicions . . . . .	9
1.2. Identificació del problema . . . . .	10
1.3. Actors implicats . . . . .	11
<b>2. Justificació</b>	<b>12</b>
2.1. Solucions i alternatives existents . . . . .	12
2.2. Solució pressa . . . . .	12
<b>3. Abast del projecte</b>	<b>13</b>
3.1. Objectius . . . . .	13
3.2. Sub-objectius . . . . .	13
3.3. Requeriments . . . . .	13
3.4. Tecnologies utilitzades . . . . .	13
3.5. Possibles riscos i obstacles . . . . .	13
3.6. Competencies tècniques . . . . .	14
<b>4. Metodologia i rigor</b>	<b>15</b>
<b>5. Planificació temporal</b>	<b>16</b>
5.1. Tasques del projecte . . . . .	16
5.1.1. Gestió del projecte . . . . .	16
5.1.2. Desenvolupament del projecte . . . . .	16
5.2. Recursos . . . . .	18
5.3. Estimació de les tasques . . . . .	18
5.4. Diagrama de gantt . . . . .	19
5.5. Gestió de riscos . . . . .	20
<b>6. Gestió econòmica</b>	<b>21</b>
6.1. Pressupost . . . . .	21
6.1.1. Recursos de maquinari . . . . .	21
6.1.2. Recursos humans . . . . .	21
6.1.3. Recursos de programari . . . . .	22
6.1.4. Despeses generals . . . . .	23
6.1.5. Cost de contingència . . . . .	23
6.1.6. Cost d'imprevistos . . . . .	23
6.1.7. Estimació final . . . . .	24
6.2. Control de gestió . . . . .	24
<b>7. Sostenibilitat del projecte</b>	<b>25</b>
7.1. Dimensió ambiental . . . . .	25
7.2. Dimensió econòmica . . . . .	25
7.3. Dimensió social . . . . .	26
<b>8. Disseny del sistema</b>	<b>27</b>
8.1. Consideracions prèvies . . . . .	27
8.2. Decisions de disseny . . . . .	27
8.3. Selecció de components . . . . .	28
8.3.1. Baròmetre . . . . .	28

8.3.2. IMU . . . . .	29
8.3.3. Memòria flash . . . . .	29
8.3.4. LoRa . . . . .	29
8.3.5. GNSS . . . . .	29
8.3.6. Microcontrolador . . . . .	30
<b>9. Disseny de la placa</b>	<b>31</b>
9.1. Esquemàtics . . . . .	31
9.1.1. Baròmetre . . . . .	31
9.1.2. IMU . . . . .	32
9.1.3. Memòria flash . . . . .	33
9.1.4. LoRa . . . . .	33
9.1.5. GNSS . . . . .	35
9.1.6. Circuits d'ignició . . . . .	36
9.1.7. Regulador de tensió . . . . .	40
9.1.8. Buzzer . . . . .	41
9.1.9. Connexions externes . . . . .	42
9.1.10. Extres . . . . .	43
9.1.11. Microcontrolador . . . . .	43
9.2. PCB . . . . .	45
<b>10. Fabricació</b>	<b>49</b>
<b>11. Proves de components</b>	<b>52</b>
11.1. Proves del microcontrolador . . . . .	52
11.2. Proves de la memòria flash . . . . .	52
11.3. Llibreria del baròmetre . . . . .	58
11.4. Llibreria IMU . . . . .	59
11.5. Llibreria GNSS . . . . .	60
11.6. Llibreria LoRa . . . . .	61
<b>12. Segona versió</b>	<b>62</b>
12.1. Disseny . . . . .	62
12.2. Fabricació . . . . .	68
12.3. Proves . . . . .	69
<b>13. Conclusions</b>	<b>71</b>
<b>A. Taules de Components</b>	<b>73</b>
A.1. Primera versió . . . . .	73
A.2. Segona versió . . . . .	74
<b>B. Codis sencers</b>	<b>75</b>
B.1. Memòria flash . . . . .	75
B.2. Baròmetre . . . . .	102
B.3. IMU . . . . .	116
B.4. GNSS . . . . .	134
B.5. LoRa . . . . .	147
<b>Bibliografia</b>	<b>170</b>

## Llista de figures

1.	Altímetre actual de la missió Ares . . . . .	10
2.	Aplicació de la metodologia iterativa . . . . .	15
3.	Diagrama de Gantt . . . . .	19
4.	Diagrama de blocs del sistema. . . . .	27
5.	Mòdul MS560702BA03-50 . . . . .	28
6.	Mòdul ICM-20948 . . . . .	29
7.	Mòdul W25Q128JV . . . . .	29
8.	Mòdul RF-LORA-868-SO . . . . .	29
9.	Mòdul SIM68M . . . . .	30
10.	Mòdul STM32F401VET6 . . . . .	30
11.	Esquemàtic de referència del baròmetre. . . . .	31
12.	Esquemàtic del baròmetre. . . . .	31
13.	Esquemàtic de referència de la IMU. . . . .	32
14.	Esquemàtic de la IMU. . . . .	32
15.	Descripció dels pins de la memòria flash. . . . .	33
16.	Esquemàtic de la memòria flash. . . . .	33
17.	Exemple de funcionament del mòdul LoRa. . . . .	34
18.	Esquemàtic del mòdul LoRa. . . . .	35
19.	Esquemàtic de referència del GNSS. . . . .	36
20.	Esquemàtic del GNSS. . . . .	36
21.	Esquemàtic de la part esquerra del circuit d'ignició, amb l'Enable a zero . . . . .	37
22.	Esquemàtic de la part esquerra del circuit d'ignició, amb l'Enable a nivell alt . . . . .	38
23.	Esquemàtic de la part dreta del circuit d'ignició . . . . .	39
24.	Esquemàtic dels circuits d'ignició . . . . .	40
25.	Esquemàtic de referència del regulador de tensió . . . . .	40
26.	Esquemàtic del regulador de tensió . . . . .	41
27.	Esquemàtic recomanat del buzzer . . . . .	41
28.	Esquemàtic del buzzer . . . . .	42
29.	Esquemàtic de referència del USB. . . . .	42
30.	Esquemàtic del USB. . . . .	43
31.	Esquemàtic del VSTlink. . . . .	43
32.	Esquemàtic dels LEDs. . . . .	43
33.	Esquemàtic dels servos. . . . .	43
34.	Esquemàtic de referència del microcontrolador . . . . .	44
35.	Esquemàtic de referència del reset . . . . .	44
36.	Esquemàtic del microcontrolador . . . . .	45
37.	Disseny de disposició recomanat del regulador de tensió. . . . .	46
38.	Vista top de la PCB. . . . .	47
39.	Vista bottom de la PCB. . . . .	47
40.	Capa top de la PCB. . . . .	47
41.	Capa 3.3V de la PCB. . . . .	48
42.	Capa GND de la PCB. . . . .	48
43.	Capa bottom de la PCB. . . . .	48
44.	Plaques fabricades. . . . .	49
45.	Placa amb pasta de soldadura aplicada. . . . .	49
46.	Placa després de sortir del forn. . . . .	50
47.	Referència de mida del regulador de tensió. . . . .	50

---

48.	Stencil de la placa. . . . .	51
49.	Placa soldada vista per a baix. . . . .	51
50.	Placa soldada vista per a dalt. . . . .	51
51.	Circuit de reset erroni. . . . .	52
52.	Prova de la funció de lectura del baròmetre. . . . .	55
53.	Prova de la funció de lectura de la IMU. . . . .	56
54.	Prova de la funció de lectura del GPS. . . . .	57
55.	Prova del baròmetre. . . . .	59
56.	Error de connexions de la IMU. . . . .	60
57.	Esquema del circuit de reset de la segona versió. . . . .	62
58.	Esquema del circuit de la <i>switch key</i> de la segona versió. . . . .	63
59.	Esquema del circuit de la IMU de la segona versió. . . . .	63
60.	Esquema del circuit del GPS de la segona versió. . . . .	64
61.	Esquema del circuit de l'USB de la segona versió. . . . .	64
62.	Esquema del circuit dels ignitors de la segona versió. . . . .	65
63.	Vista superior de la PCB de la segona versió. . . . .	65
64.	Vista inferior de la PCB de la segona versió. . . . .	66
65.	Capa superior de la PCB de la segona versió. . . . .	66
66.	Capa inferior de la PCB de la segona versió. . . . .	67
67.	Visualització 3D superior de la segona versió. . . . .	67
68.	Visualització 3D inferior de la segona versió. . . . .	68
69.	Cara superior de la segona versió. . . . .	68
70.	Cara inferior de la segona versió. . . . .	69
71.	Marques del microcontrolador. . . . .	70

## Llista de taules

1.	Estimació de les Tasques . . . . .	18
2.	Identificació de riscos i plans alternatius. . . . .	20
3.	Maquinari utilitzat i la seva amortització. . . . .	21
4.	Salaris pels diferents rols del projecte . . . . .	21
5.	Costos estimats per tasca. . . . .	22
6.	Costs estimats per despeses generals. . . . .	23
7.	Costs estimats per contingències. . . . .	23
8.	Costos d'imprevistos. . . . .	23
9.	Pressupost final estimat. . . . .	24
10.	Característiques del mòdul MS560702BA03-50. . . . .	28
11.	Característiques del mòdul ICM-20948. . . . .	29
12.	Característiques del mòdul W25Q128JV. . . . .	29
13.	Característiques del mòdul RF-LORA-868-SO. . . . .	29
14.	Característiques del mòdul SIM68M. . . . .	30
15.	Característiques del microcontrolador STM32F401VET6. . . . .	30
16.	Organització de pagines. . . . .	53
17.	Organització de la memòria de la pàgina 0. . . . .	53
18.	Taula de components de la primera versió. . . . .	73
19.	Taula de components de la segona versió. . . . .	74

## 1. Introducció

El treball de fi de grau “Disseny i desenvolupament d'un controlador de vol, per a la seva implementació en un coet que participi a la competició European Rocketry Challenge” del Grau d'Enginyeria Informàtica especialitzat en Enginyeria de Computadors, té com a objectiu principal millorar el firmware i el hardware utilitzats actualment en un coet de petita escala i que es presentarà a una competició entre equips de diferents universitats europees. Aquest projecte es realitza a la Facultat d'Informàtica de Barcelona (FIB), de l'Universitat Politècnica de Catalunya (UPC), i dins del projecte UPC Space Program.

A l'estiu de 2023 vaig entrar al departament d'electrònica de la missió Ares del projecte UPC Space Program, ja que en Guillem Garcia, amic meu i actual coordinador de la missió em va convèncer a entrar al departament d'electrònica. A la missió Ares ens dediquem a dissenyar, construir i llençar coets de maqueta, amb l'objectiu d'entrar al European Rocketry Challenge (EUROC), i concretament al departament d'electrònica, principalment dissenyem l'altímetre que porta el coet, la “ground station” i recopilem les dades obtingudes, per posteriorment analitzar-les.

Aquest projecte va sorgir perquè, encara que teníem un disseny d'altímetre, tenia certs defectes com, un tamany massa gran per alguns coets, poques funcions addicionals, i errors menors en el disseny de la placa. Per tant, llançàvem els coets amb el nostre altímetre de ”passatger”, i un altímetre comercial, que s'assegurava que s'obrissin els paracaigudes. A vista de tot això, encara que la normativa de l'EUROC no requereix fer servir un altímetre propi, ajuda a entrar i a puntuar a dins de la competició, per tant, vaig proposar d'aprofitar que havia de fer el treball de fi de grau, per dissenyar de zero un altímetre nou, i que quedés ben documentat tot el procés. Aprofitant aquesta necessitat s'ha redissenyat tota l'electrònica del projecte.

### 1.1. Terminologia i definicions

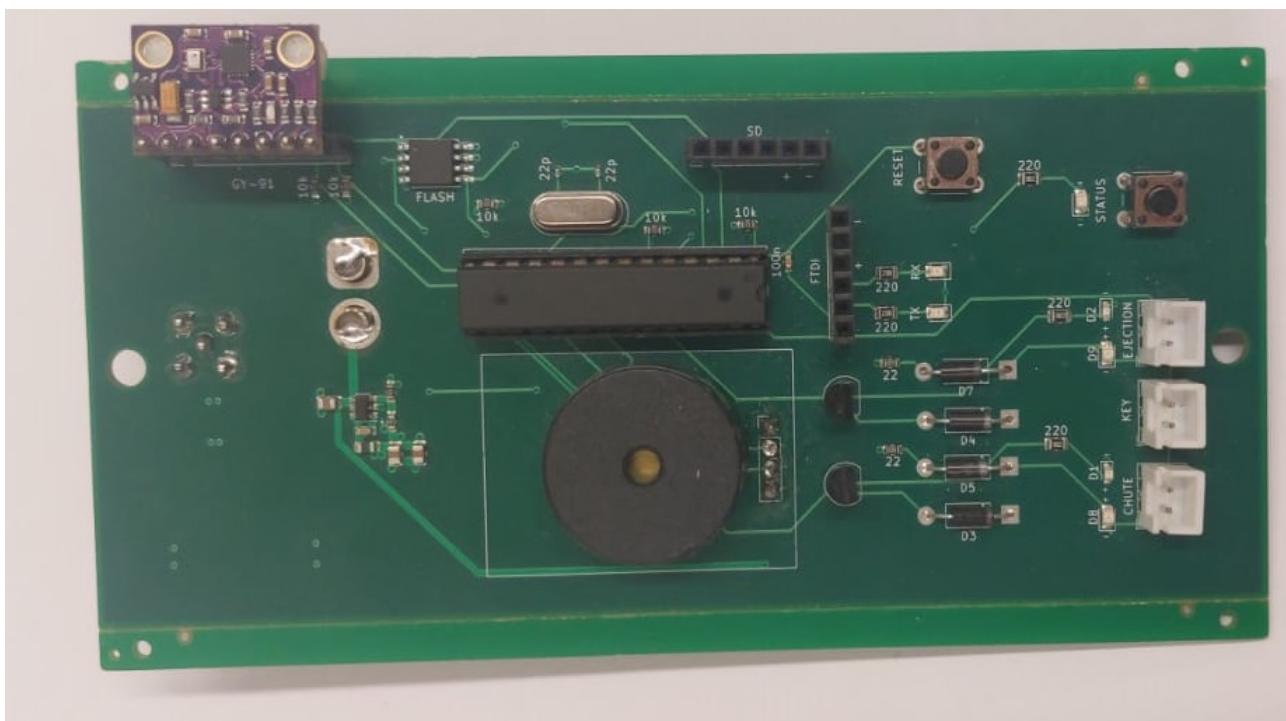
- **Altímetre:** En el context dels coets, un altímetre és un sensor utilitzat per determinar l'altura del coet durant el vol. Això és crucial per a la supervisió dels diferents segments de la missió i per a la implementació de diverses funcions com ara el desplegament dels paracaigudes en el moment adequat.
- **Ground station:** Una ground station és el centre de control des d'on s'observa i s'interactua amb el coet durant el seu vol. Aquesta estació pot estar equipada amb diverses tecnologies com ara ràdios, antenes i ordinadors per comunicar-se amb el coet i rebre dades en temps real sobre el seu estat i la seva ubicació.
- **Apogeu:** L'apogeu és el punt més alt o la màxima altitud assolida per un coet durant el seu vol. En altres paraules, és el punt on la velocitat vertical del coet s'atura abans que comenci el descens cap a la terra. El moment de l'apogeu és crític per a diverses funcions del coet, com ara el desplegament dels paracaigudes o l'execució de determinades maniobres planificades.
- **Drogue:** El ”drogue”és un tipus de paracaigudes petit per ajudar a frenar un coet durant el descens inicial després de l'apogeu. A diferència del paracaigudes principal (main), el drogue és més petit i està dissenyat per crear resistència aerodinàmica i estabilitzar el coet abans que es desplegi el paracaigudes principal per al descens final.
- **Main:** El terme ”main”, es refereix al paracaigudes principal utilitzat per frenar i estabilitzar el descens del coet després de l'apogeu. Aquest paracaigudes és normalment el de major mida i

està dissenyat per proporcionar una descens lent i controlat del coet fins a la seva arribada al terra.

## 1.2. Identificació del problema

Per començar, cal explicar breument com funcionen els models de coet que participen a l'EUROC, o competicions similars. Són coets que varien entre mides de 30 cm a més de 3 m, que s'impulsen amb motors de combustible sòlid, el que fa que puguin pujar fins a 10 km depenent del model. Com els coets més grans tenen un pes considerable, i estan dissenyats per tindre una forma aerodinàmica, tant com pugen molt ràpid, també cauen molt ràpid. Per això és molt important tindre un sistema de paracaigudes que s'obri en el moment indicat, i així no tindre el que essencialment seria un míssil, i aquí és on entren els altímetres, que a part d'obrir els paracaigudes, poden afegir més funcionalitats al coet.

Els altímetres comercials existents al mercat no ofereixen un gran rang de funcionalitats, per un preu raonable, i com a la nostra missió hi ha un pressupost molt ajustat comprar-ne un de nou no era una opció viable. L'altímetre que tenim ja dissenyat, com es pot veure a la Figura 1, no té un disseny gaire eficient, ja que és una placa innecessàriament gran, veient tot l'espai desocupat entre components, el que fa que no càrga en molts models de coets. A part, tampoc té gaires funcionalitats, i la funcionalitat essencial de l'encesa dels ignitors, no s'ha comprovat de manera correcta.



**Figura 1:** Altímetre actual de la missió Ares

A vista de tot això, aquest projecte de fi de grau arrenca de la necessitat de crear un altímetre propi amb les funcionalitats necessàries, que ajudi a la missió a entrar i competir a la competició EURO[1].

### 1.3. Actors implicats

- **Desenvolupador:** Aquest projecte té un únic desenvolupador, l'Arnau Mariegas Barot. Ell serà l'encarregat de conduir la investigació sobre quines tecnologies utilitzar i el desenvolupament de la solució proposada. Tot això, conjuntament amb tota la documentació del projecte.
- **Director del Treball de Fi de Grau:** L'encarregat de la supervisió d'aquest projecte és en Antonio Camacho Santiago, del Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial de l'UPC.
- **UPC Space Program:** La missió UPC Space Program es veu directament implicada, ja que proporciona els coneixements dels dissenys d'altímetres antics, materials i recursos per ajudar al desenvolupament del projecte, i és el destinatari final del resultat d'aquest projecte.
- **Organitzacions:** En el cas que es fessin públics els resultats del projecte, qualsevol altra associació o empresa que faci altímetres, podria inspirar-se en parts del treball, o inclús fer servir el disseny final.

## 2. Justificació

Com que el coet actual requereix una actualització de hardware i firmware, i aprofitant que la missió s'està preparant per ingressar a l'EUROC, s'ha decidit que la creació d'una nova placa és la millor opció.

### 2.1. Solucions i alternatives existents

Ara mateix a la missió Ares, fem servir un altímetre comercial, per assegurar-nos que el coet obrirà els seus paracaigudes, però aquesta opció té certs desavantatges, com la falta de funcionalitats addicionals, a part de mesurar l'altura del vol i obrir els paracaigudes, com podrien ser un GPS per a localitzar el coet un cop ha caigut o un acceleròmetre per poder veure l'inclinació del coet durant el vol. Encara que sí que és veritat que existeixen alguns models d'altímetre que tenen aquestes funcionalitats, solen ser més cars, i a la missió un dels problemes principals que tenim és el pressupost.

Un altre desavantatge seria que no permeten modificació o addició de funcionalitats noves, per exemple, si volem fer un coet amb doble propulsió, és a dir, que un motor del coet s'encengui quan el primer s'ha esgotat, s'hauria de modificar el programa del altímetre i hauria de tindre canals de sortida addicionals, entre altres modificacions necessàries, i com un altímetre comercial no proporciona els seus esquemàtics ni el seu codi font, aquestes modificacions costarien massa de fer.

L'altra opció que tenim és fer servir la placa ja existent, però com ja s'ha explicat en la secció Identificació del problema, realment no és la millor opció, ja que requeriria de massa modificacions del disseny original, i que aquestes modificacions, al no tenir una documentació del disseny, serien essencialment com començar de zero el projecte.

### 2.2. Solució pressa

Finalment, es va decidir dissenyar un altímetre nou, ja que el seu disseny està directament alineat amb els objectius de la missió, com abordar les limitacions actuals i proporcionar una solució més eficient i precisa, o trobar una solució més econòmica als altímetres comercials. A més, l'elaboració de documentació sobre el procés de desenvolupament d'aquest nou altímetre contribuirà a la transferència de coneixement a futurs integrants de la missió, fent les possibles modificacions, o noves versions del disseny més fàcils de fer.

### 3. Abast del projecte

#### 3.1. Objectius

L'objectiu principal d'aquest treball és crear un altímetre amb les funcions basiques de l'ignició dels paracaigudes i l'enregistrament de l'altura del vol, com fan la majoria altimetres comercials. També es vol dissenyar amb funcions adicionals, que no són essencials pel seu funcionament, però fan que l'altímetre sigui molt complet.

#### 3.2. Sub-objectius

Un cop s'hagin complert els objectius, s'ha de considerar dissenyar un programa que recopili les dades obtingudes, i les presenti en formats visuals com gràfiques. I també es vol considerar dissenyar una ground station nova.

#### 3.3. Requeriments

A continuació s'especifiquen els requisits dels quals necessita el treball:

- És necessari estudiar les funcionalitats, i com aconseguir-les, de l'altímetre a dissenyar.
- L'altímetre ha de ser capaç d'ignitar les càrregues dels paracaigudes en el moment adequat, i de manera confiable.
- L'altímetre ha de guardar les dades obtingudes dels sensors de manera precisa, pel seu anàlisi posterior.
- S'haurà de tindre una primera versió al 16 de març, que es farà un llançament per provar l'altímetre.

#### 3.4. Tecnologies utilitzades

Per poder completar aquests objectius i requisits es faran servir diferents tecnologies. Per començar a dissenyar l'altímetre es farà servir el programa Fusion 360[2], que permet fer els esquemes elèctrics, el disseny de la PCB (de l'anglès Printed Circuit Board), i fa un render automàtic de la placa. Posteriorment, potser es passarà el disseny a Altium Designer[3], ja que és possible que es pugui aconseguir una llicència del programa. Per la fabricació de la placa es farà servir l'empresa JLCPCB[4], que fabrica les plaques amb preus econòmics, i en terminis de temps acceptables. Finalment pel desenvolupament software es farà servir STM32 Cube[5], que és l'IDE del fabricant de microcontroladors ST, i que proporciona moltes facilitats per programar els seus dispositius.

#### 3.5. Possibles riscos i obstacles

Quan es comença el disseny d'una placa des de zero, poden sorgir molts problemes, especialment si es disposa de poc temps. Un problema del disseny podria ser tindre errors en la PCB, que faria que el procés s'endarrerís dues setmanes, que és el que triga a arribar una placa nova, i encara que el projecte s'ha d'entregar al juny, per poder fer proves reals caldrà ajustar-se als horaris dels llançaments, el primer dels quals està previst per al 16 de març. Un altre problema seria que arribés un component defectuós, o s'espantllés fent les proves, que també tindria un retràs d'unes dues setmanes.

Un altre possible problema seria el desenvolupament del software, també amb les deadlines tan ajustades, es pot arribar al dia del llançament amb un software poc depurat, i en aquest cas les proves de llançament no serien productives.

### 3.6. Competencies tècnicas

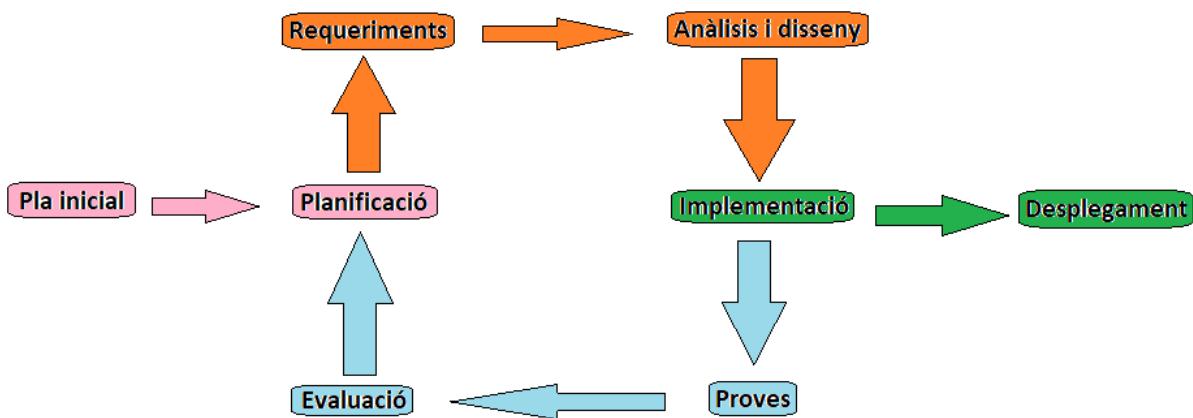
Considerant aquests aspectes, les competències tècnicas que s'han considerat adequades i que s'han escollit per aquest projecte són:

- CEC1.1: Dissenyar un sistema basat en microprocessador/microcontrolador.
- CEC2.3: Desenvolupar i analitzar software per a sistemes basats en microprocessadors i les seves interfícies amb usuaris i altres dispositius.
- CEC2.4: Dissenyar i implementar software de sistema i de comunicacions.
- CEC3.1: Analitzar, avaluar i seleccionar les plataformes hardware i software més adients per al suport d'aplicacions encastrades i de temps real.

## 4. Metodologia i rigor

Per la realització del projecte s'ha decidit utilitzar un mètode iteratiu. Per això cada cop que s'avanci en una tasca, es comunicarà al tutor, i es faran les reunions pertinents per poder analitzar el progrés i fer les modificacions que faci falta, per tindre el millor resultat possible. El mètode iteratiu està basat en 6 fases:

1. **Planificació:** es crea un pla pel projecte perquè s'alinei amb els objectius d'aquest.
2. **Requisits:** es defineixen els requisits bàsics per complir el projecte exitosament.
3. **Anàlisis i disseny:** es comencen a crear els primers dissenys del projecte.
4. **Implementació:** es crea la primera iteració del projecte, basat en el disseny inicial.
5. **Proves:** la implementació ha de passar un joc de proves per comprovar el seu correcte funcionament.
6. **Evaluació:** amb els resultats de les proves, s'analitza quines coses s'han de canviar, i quines es poden millorar, i si fa falta es pot començar una nova iteració.



**Figura 2:** Aplicació de la metodologia iterativa. Font: Elaboració pròpia

Per complir amb la metodologia, s'utilitzaran varíes eines de gestió de projectes, per poder tindre el projecte ordenat, i que sigui fàcil de seguir. La primera eina és Jira[6], que és una eina web que permet crear un projecte i administrar les seves tasques. També es farà servir Google Drive[7] per mantenir tota la documentació al núvol, i accessible des de tot arreu. Per la gestió del codi s'utilitzarà Github[8]. Finalment per la gestió del disseny hardware, es farà servir el mateix Fusion 360.

## 5. Planificació temporal

L'objectiu d'aquest apartat és fer una guia pel seguiment temporal del treball, per així tindre un marc delimitat per acabar el treball a temps.

La data d'inici del treball es fixa al 19 de febrer de 2024, i la data final a 18 de juny. S'estima una dedicació de 30 hores setmanals, que poden variar depenent de les fases del projecte, o altres compromisos en certes dates, i per això aquest número és una aproximació. Si tenim en compte que hi ha 17 setmanes, la dedicació total esperada serà d'unes 510 hores.

Cada tasca estarà dins d'un bloc, que definirà la seva finalitat. Els blocs són: Gestió del Projecte (GEP) i Desenvolupament del Projecte (DEP).

### 5.1. Tasques del projecte

#### 5.1.1. Gestió del projecte

En aquest bloc s'hi recullen totes les tasques que tenen a veure amb la gestió del projecte, com la seva documentació, o preparació prèvia. Aquestes tasques es faran en paral·lel amb les altres tasques, durant tota la durada del projecte, i es calcula que tingui una durada total de 120 hores.

- **GEP1 - Contextualització i Abast (20h):** Implica la creació del seu primer document, en el qual es presenta una introducció al projecte, es detalla la seva justificació, s'estableixen els objectius i es descriu la metodologia i els recursos que s'utilitzaran.
- **GEP2 - Planificació Temporal (10h):** Consisteix a establir una planificació temporal inicial que servirà com a guia al llarg de tot el projecte. **Dependències:** GEP1.
- **GEP3 - Pressupost i Sostenibilitat (10h):** Implica realitzar una autoavaluació i anàlisi de la sostenibilitat del projecte. **Dependències:** GEP2.
- **GEP4 - Documentació de la fase inicial (5h):** Es basa a aplicar les correccions dels apartats anteriors (GEP1, GEP2, GEP3). **Dependències:** GEP3.
- **GEP5 - Documentació de la fase de Recerca i Investigació (15h):** Consisteix a escriure i documentar el progrés en la fase de Recerca i Investigació. **Dependències:** GEP4.
- **GEP6 - Documentació del Desenvolupament del Projecte (25h):** Consisteix en escriure i documentar el progrés en la fase del Desenvolupament del Projecte **Dependències:** GEP5.
- **GEP7 - Memòria i Defensa del Projecte (20h):** Es farà la redacció de la documentació de la memòria. A més, es desenvoluparà divers material i es realitzaran assajos en preparació per a la presentació oral de la defensa **Dependències:** GEP6.
- **GEP8 - Reunions amb el Director (15h):** Es farà reunions amb el tutor per controlar que el desenvolupament del projecte sigui correcte.

#### 5.1.2. Desenvolupament del projecte

Aquest bloc es divideix en tasques repartides, i dues seccions importants: Desenvolupament Hardware (DH) i Desenvolupament Software (DS), i es calcula que tingui una durada de 390 hores.

- **DEP1 - Investigació dels Productes en el Mercat (50h):** Recerca sobre altímetres comercials actuals, mirant quines funcionalitats tenen i com les apliquen.

- **DEP2 - Instal·lació de Programari (10h):** Instal·lació i configuració del programari utilitzat per el desenvolupament del projecte.

### Desenvolupament Hardware (DH)

- **DH1 - Selecció de Sensors (50h):** Investigació sobre quins sensors seran necessaris, i quins models són els més adequats. **Dependències:** DEP1.
- **DH2 - Selecció de Microcontrolador (15h):** Investigació sobre quin microcontrolador encaixa més amb les necessitats dels sensors, i possibles futures aplicacions. **Dependències:** DH1.
- **DH3 - Planificació de Disseny (15h):** Començar a fer esquemes preliminars amb els sensors, per tindre una idea de quins detalls s'haurà de tindre en compte en fer el disseny complet. **Dependències:** DH2.
- **DH4 - Selecció de la Resta de Components (20h):** A partir de la planificació de disseny, buscar els components com resistències, condensadors o LED, per fer el disseny i encarregar els components com més aviat millor. **Dependències:** DH3.
- **DH5 - Disseny de la Placa (40h):** Fer els esquemàtics, i disseny de la PCB. **Dependències:** DH4.
- **DH6 - Encarregar Components (2 setmanes):** Temps d'espera habitual perquè arribin els components. **Dependències:** DH4.
- **DH7 - Encarregar la Placa (2 setmanes):** Temps d'espera usual perquè arribi la placa. **Dependències:** DH5.
- **DH8 - Fabricació de la Placa (10h):** Soldar els components a la placa. **Dependències:** DH6, DH7.
- **DH9 - Proves de la Placa (20h):** Provar que la placa funcioni correctament. **Dependències:** DH8.

### Desenvolupament Software (DS)

- **DS1 - Selecció i Creació de Llibreries (50h):** Investigar llibreries dels components utilitzats, i en cas que no hi hagi, crear llibreries pròpies. **Dependències:** DH4
- **DS2 - Creació del Programa (60h):** Desenvolupar el programa que utilitzarà l'altímetre. **Dependències:** DS1.
- **DS3 - Depuració del Programa (30h):** Corregir errors, i millorar el codi del programa. **Dependències:** DS2.
- **DS4 - Proves del Software (20h):** Comprovar que tot el programa funcioni correctament. **Dependències:** DH8, DS2.

## 5.2. Recursos

Els recursos humans d'aquest treball, serà només una persona, el mateix autor.

El programari utilitzat per la gestió del projecte serà Jira[6], Google Drive[7] i TeamGantt[9], per fer la documentació es farà servir Overleaf[10], pel disseny de la placa es farà servir Fusion360[2], i pel disseny software es farà servir STM32Cube[5].

El maquinari utilitzat serà un ordinador personal, un forn per soldadura facilitat per l'UPC, i material de soldadura proporcionat per UPC Space Program.

## 5.3. Estimació de les tasques

Després de detallar totes les tasques, a la Taula 1 s'hi resumeixen amb el seu identificador, nom, quantitat d'hores assignades i les seves dependències.

ID	Tasca	Dedicació	Depèndencies
	Gestió del Projecte	120 hores	
GEP1	Contextualització i Abast	20 hores	
GEP2	Planificació Temporal	10 hores	GEP1
GEP3	Pressupost i Sostenibilitat	10 hores	GEP2
GEP4	Documentació de la fase inicial	5 hores	GEP3
GEP5	Documentació de la fase de Recerca i Investigació	15 hores	GEP4
GEP6	Documentació del Desenvolupament del Projecte	25 hores	GEP5
GEP7	Memoria i Defensa del Projecte	20 hores	GEP6
GEP8	Reunions amb el Director	15 hores	
	Desenvolupament del Projecte	390 hores	
DEP1	Investigació dels Productes en el Mercat	50 hores	
DEP2	Instal·lació de Programari	10 hores	
DH	Desenvolupament Hardware	170 hores	
DH1	Selecció de Sensors	50 hores	DEP1
DH2	Selecció de Microcontrolador	15 hores	DH1
DH3	Planificació de Disseny	15 hores	DH2
DH4	Selecció de la Resta de Components	20 hores	DH3
DH5	Disseny de la Placa	40 hores	DH4
DH6	Encarregar Components	2 setmanes	DH4
DH7	Encarregar la Placa	2 setmanes	DH5
DH8	Fabricació de la Placa	10 hores	DH6, DH7
DH9	Proves de la Placa	20 hores	DH8
DS	Desenvolupament Software	160 hores	
DS1	Selecció i Creació de Llibreries	50 hores	DH4
DS2	Creació del Programa	60 hores	DS1
DS3	Depuració del Programa	30 hores	DS2
DS4	Proves del Software	20 hores	DH8, DS2
	Total	510 hores	

**Taula 1:** Estimació de les Tasques. Font: Elaboració pròpia

## 5.4. Diagrama de gant

A continuació es mostra el diagrama de Gantt de les tasques.

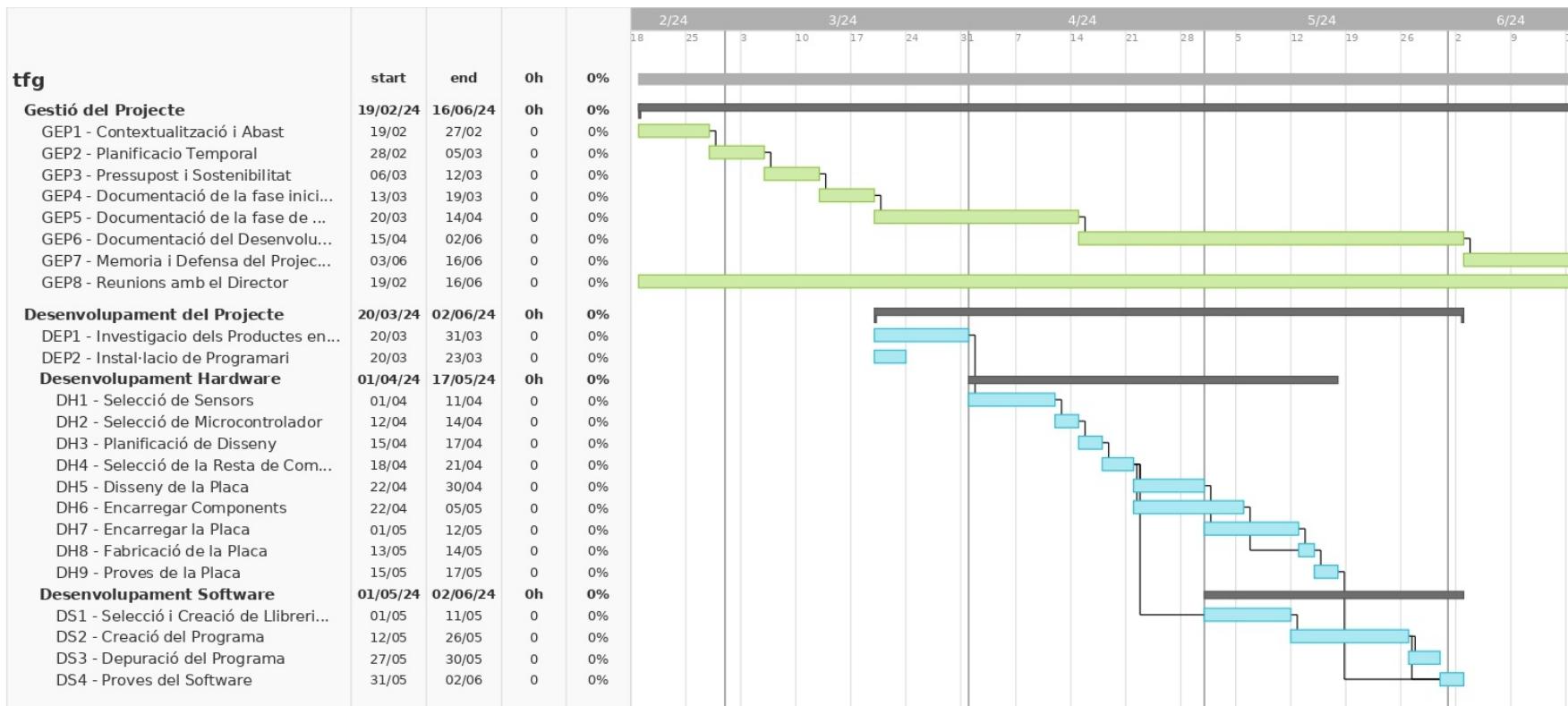


Figura 3: Diagrama de Gantt. Font: Elaboració pròpia mitjançant TeamGantt[9]

## 5.5. Gestió de riscos

A continuació, a la Taula 2, s'enumeren els possibles riscos i obstacles que podrien sorgir durant la implementació del projecte. S'assigna una probabilitat a cada risc, s'avalua l'impacte potencial en el progrés del treball i es proposa un pla alternatiu en cas que aquestes situacions es materialitzin.

Riscos	Probabilitat	Impacte	Pla alternatiu
Limitacions de temps	Mitjana (40 %)	Baix	L'aplicació d'una metodologia iterativa significa que cada fase comença una vegada finalitza la anterior. Per tant, si cal modificar la planificació, no representaria un gran problema, simplement s'ajustaria el calendari
Inexperiència en certes tecnologies	Mitjana (60 %)	Baix	En cas de requerir més temps per dominar una tecnologia específica, per manca d'experiència, hi ha dues opcions disponibles: dedicar més hores al projecte o redistribuir el temps assignat a altres tasques.
Errors lleus a la placa	Alta (70 %)	Baix	A les primeres versions d'una placa es comú que surtin errors, i poden ser de tot tipus, però mentre es compleixin les funcionalitats principals, es pot utilitzar, però en futures versions s'han de corregir aquests errors.
Errors greus a la placa	Mitjana (40 %)	Alt	Normalment, és difícil que hi hagi un error greu, però, s'ha de tindre en compte que si no es té gaire experiència creant esquemes electrònics o plaques, és més fàcil que surtin errors greus. L'única opció que hi ha per corregir aquest risc és arreglar els errors i tornar a encarregar la placa, i els components que no es puguin recuperar.
Errors en el codi	Alta (80 %)	Baix	És poc probable que els errors de codi tinguin un impacte notable en la gestió del temps. En general, la majoria d'aquests errors poden ser resolts sense dificultats mitjançant un ànalisi adequat del codi.

**Taula 2:** Identificació de riscos i plans alternatius. Font: Elaboració pròpia

## 6. Gestió econòmica

La planificació econòmica de qualsevol projecte implica una estimació detallada dels costos associats amb el seu desenvolupament. Això implica la identificació i valoració dels recursos humans, materials i tecnològics necessaris, així com la consideració d'impostos, contingències i possibles imprevistos que puguin sorgir durant el curs del projecte. Una vegada que aquests costos estan degudament calculats, s'estableixen mètriques per a monitorar i gestionar qualsevol desviació respecte al pressupost inicial, garantint així un control financer efectiu al llarg de tot el procés d'execució del projecte.

### 6.1. Pressupost

El primer pas de l'elaboració del pressupost és classificar, i llistar els pressupostos segons els recursos utilitzats. Poden ser humans, materials o generals.

#### 6.1.1. Recursos de maquinari

Aquests costos són tots els que estan associats al maquinari que és necessari utilitzar. A la Taula 3 hi ha els costos de maquinari d'aquest projecte, amb la seva amortització, que per calcular-la s'ha pres com a referència un any de 222 dies laborables de 8 hores. Com hisenda permet amortitzar el maquinari en 4 anys, abans de renovar-lo per obsolescència, ens queda aquesta fórmula:

$$\frac{\text{Cost del Dispositiu (\euro)} \times \text{Temps d'ús durant el TFG (hores)}}{\text{Vida útil (4 anys)} \times \text{Dies laborals anuals (222 dies)} \times \text{Dedicació diària (8 hores)}}$$

Maquinari	Vida útil	Hores	Cost	Amortització
Ordinador personal	4 anys	510 hores	1374,34 €	99,93 €
Kit de soldadura	4 anys	10 hores	79,99 €	0,11 €
Total			1454,33 €	100,04 €

**Taula 3:** Maquinari utilitzat i la seva amortització. Font: Elaboració pròpia

#### 6.1.2. Recursos humans

Els costos de recursos humans són aquells relacionats amb els salaris dels treballadors involucrats en el projecte. Encara que en aquest cas el projecte és desenvolupat únicament pel seu autor, es considerarà l'escenari amb diferents persones pels diferents rols. Per a calcular aquests costos, s'ha optat per utilitzar la plataforma Indeed[11], que permet comparar els salaris de treballs publicats en l'àrea especificada. En aquest cas, es buscarà el salari mitjà del rol introduït a Espanya. Els resultats obtinguts es mostren en la Taula 4, que té el salari per hora brut de cada rol, i a la columna de la dreta s'hi afegeix el 33 % que ha de tributar l'empresa a la seguretat social.

Rol	Salari/Hora	Salari/Hora + Seguretat Social
Cap de Projecte (CP)	23,48 €	31,23 €
Dissenyador de PCB (DP)	16,07 €	21,37 €
Programador (P)	17,09 €	22,73 €
Tester (T)	14,24 €	18,94 €

**Taula 4:** Salaris pels diferents rols del projecte. Font: Elaboració pròpia

Amb la informació de la Taula 4, i de les tasques té la Taula 1, podem fer una estimació dels costos dels recursos humans, que es veuen representats a la Taula 5.

ID	Tasca	Dedicació	Rols	Cost
	Gestió del Projecte	120 hores		3441,1 €
GEP1	Contextualització i Abast	20 hores	CP	624,6 €
GEP2	Planificació Temporal	10 hores	CP	312,3 €
GEP3	Pressupost i Sostenibilitat	10 hores	CP	312,3 €
GEP4	Documentació de la fase inicial	5 hores	CP	156,15 €
GEP5	Documentació de la fase de Recerca i Investigació	15 hores	CP	468,45 €
GEP6	Documentació del Desenvolupament del Projecte	25 hores	CP, DP, P, T	589,19 €
GEP7	Memoria i Defensa del Projecte	20 hores	CP	624,6 €
GEP8	Reunions amb el Director	15 hores	CP, DP, P, T	353,51 €
	Desenvolupament del Projecte	390 hores		8277,5 €
DEP1	Investigació dels Productes en el Mercat	50 hores	CP	1561,5 €
DEP2	Instal·lació de Programari	10 hores	DP, P	220,5 €
DH	Desenvolupament Hardware	170 hores		2991,8 €
DH1	Selecció de Sensors	50 hores	DP	1068,5 €
DH2	Selecció de Microcontrolador	15 hores	DP	320,55 €
DH3	Planificació de Disseny	15 hores	DP	320,55 €
DH4	Selecció de la Resta de Components	20 hores	DP	427,4 €
DH5	Disseny de la Placa	40 hores	DP	854,8 €
DH6	Encarregar Components	2 setmanes	-	-
DH7	Encarregar la Placa	2 setmanes	-	-
DH8	Fabricació de la Placa	10 hores	DP	213,7 €
DH9	Proves de la Placa	20 hores	T	378,8 €
DS	Desenvolupament Software	160 hores		3503,7 €
DS1	Selecció i Creació de Llibreries	50 hores	P	1136,5 €
DS2	Creació del Programa	60 hores	P	1363,8 €
DS3	Depuració del Programa	30 hores	P, T	624,6 €
DS4	Proves del Software	20 hores	T	378,8 €
	Total			11718,6 €

**Taula 5:** Costos estimats per tasca. Elaboració pròpria.

### 6.1.3. Recursos de programari

Tots els programaris utilitzats en aquest projecte són d'accés gratuït, o es té un espònsor amb l'empresa que el proporciona.

#### 6.1.4. Despeses generals

Dintre de les despeses generals hi poden entrar el local, la llum, l'internet o el mobiliari. A la taula 6 s'hi pot veure les despeses generals d'aquest projecte.

Recurs	Cost
Internet	60 €
Electricitat	15,06 €
Total	75,06 €

**Taula 6:** Costs estimats per despeses generals. Font: Elaboració pròpia

L'accés a Internet s'estima que costarà 27 € durant 5 mesos, però com també s'utilitzarà per altres activitats, es calcula que seria un cost mensual de 12 €. Per l'electricitat s'ha calculat que l'ordinador gasta 0,25 kWh, i la bombeta de l'habitació gasta 0,013 kWh, i s'ha tingut en compte el preu actual de la llum de 0,1123 €/kWh. Per al local no s'ha tingut res en compte ja que es farà tot desde el domicili familiar.

#### 6.1.5. Cost de contingència

La contingència és una reserva de recursos destinada a abordar imprevistos o desviacions no contemplades inicialment en el pressupost. En el context dels projectes informàtics, aquest cost sol estar entre el 10 % i el 20 %. Així que s'ha optat per assignar un 15 % com a cost de contingència, com es pot veure al resultat de la taula 7.

Recursos	Cost	Contingències
Recursos de maquinari	100,04 €	15,00 €
Recursos humans	11718,6 €	1757,79 €
Despeses generals	1454,33 €	218,15 €
Total	13272,97 €	1990,94 €

**Taula 7:** Costs estimats per contingències. Font: Elaboració pròpia

#### 6.1.6. Cost d'imprevistos

A la secció Gestió de riscos, ja s'expliquen quins són els riscos, i com es planeja afrontar-los. En aquesta secció es tindran en compte aquests riscos, per poder ajustar una part del pressupost pels costos que generaran.

A la Taula 8 es veuen els costos dels imprevistos, que es calculen amb el sou dels rols implicats, multiplicat per la probabilitat que passin.

Risc	Probabilitat	Hores	Rol	Cost
Limitacions de temps	40 %	10 hores	CP	124,92 €
Inexperiència en certes tecnologies	60 %	5 hores	DP, P	66,15 €
Errors lleus a la placa	70 %	5 hores	DP	74,79 €
Errors greus a la placa	40 %	20 hores	DP	170,96 €
Errors en el codi	80 %	5 hores	P	90,92 €
Total				527,74 €

**Taula 8:** Costos d'imprevistos. Font: Elaboració pròpia

### 6.1.7. Estimació final

La Taula 9 mostra el resum de l'estimació del pressupost.

Tipus de cost	Cost
Recursos de maquinari	100,04 €
Recursos humans	11718,6 €
Despeses generals	1454,33 €
Contingències	1990,94 €
Imprevistos	527,74 €
Total	15791,65 €

Taula 9: Pressupost final estimat. Font: Elaboració pròpia

## 6.2. Control de gestió

Per tal de garantir un control efectiu del compliment del pressupost, s'han establert diverses mètriques per monitorar i visualitzar qualsevol desviació que pugui sorgir durant el desenvolupament del projecte. Això inclou registrar les hores reals dedicades a cada tasca i qualsevol imprevist que pugui afectar-les. Es preveu realitzar un seguiment setmanal dels costos del projecte en reunions específiques, i al final de cada tasca es durà a terme una revisió del temps i dels recursos utilitzats. Aquests mecanismes de control són essencials per garantir que les estimacions i el pressupost inicial es compleixin, i es revisaran periòdicament per mantenir-se al corrent de les desviacions que puguin sorgir durant el desenvolupament del projecte. Les mètriques utilitzades quedaran d'aquesta manera:

- **Desviació total d'hores:** Hores totals estimades - hores totals reals
- **Desviació total de costos:** Cost total estimat - Cost total real.
- **Desviació total de costos humans:** Cost humà per activitat estimat - Cost humà per activitat real.
- **Desviació total de costos generals:** Cost general estimat - cost general real.
- **Desviació cost humà per tasca:** (Cost estimat - cost real) \* Hores reals.
- **Desviació cost d'hores per tasca:** (Hores estimades - hores reales) \* Cost real.

## 7. Sostenibilitat del projecte

Després de completar l'enquesta sobre sostenibilitat, he arribat a reconèixer l'amplitud i la complexitat d'aquesta qüestió en el context dels projectes actuals. Tot i que anteriorment ja tenia una noció general sobre la importància de la sostenibilitat en les nostres accions diàries, aquesta enquesta ha revelat diverses dimensions que abans no havia considerat amb profunditat.

Una de les primeres reflexions que em ve al cap és com la sostenibilitat abasta més que només el medi ambient. Si bé és cert que la preocupació per la conservació del nostre entorn natural és important, també és imprescindible comprendre els impactes econòmics i socials de les nostres decisions i accions. Mentre que en algunes àrees, com la gestió de residus tecnològics, havia pres consciència de la necessitat de canvis, altres aspectes, com la sostenibilitat social, han sortit a la llum com a temes que requereixen una anàlisi més aprofundida.

En definitiva, l'enquesta m'ha mostrat que, malgrat tenir una comprensió inicial de la sostenibilitat, hi ha molts matisos i aspectes tècnics que requereixen un aprenentatge continuat. Aquesta consciència em motiva a aprofundir en aquests coneixements i aplicar-los en els projectes futurs, amb l'objectiu de contribuir de manera més efectiva a la construcció d'un futur sostenible per a tots.

### 7.1. Dimensió ambiental

**PPP:** *Has estimat l'impacte ambiental que tindrà la realització del projecte? T'has plantejat minimitzar l'impacte, per exemple, reutilitzant recursos?*

Estimo que aquest projecte té poc impacte mediambiental, encara que no nul, ja que, sí que és veritat que tot el projecte es fa amb maquinari ja existent, també s'haurà d'utilitzar material nou per crear la placa. Aquest material no només té un impacte en la seva fabricació, sinó que també el té en el seu transport. Però considero que en ser components que requereixen una gran fiabilitat, és necessari que siguin nous.

**Vida Útil:** *Com es resol actualment el problema que vols abordar (estat de l'art)? En què millorarà ambientalment la teva solució a les existents?*

Actualment, els altímetres comercials no tenen un gran impacte, ja que es solen utilitzar fins que algun component de la placa deixa de funcionar. Malauradament, aquest projecte no té manera de millorar l'impacte ambiental respecte a les opcions existents.

### 7.2. Dimensió econòmica

**PPP:** *Has estimat el cost de la realització del projecte (recursos humans i materials)?*

El cost de la realització del projecte es troba a l'apartat Gestió econòmica, encara que la part de recursos humans serà una persona en lloc de quatre.

**Vida Útil:** *Com es resol actualment el problema que vols abordar (estat de l'art)? En què millorarà econòmicament la teva solució a les existents?*

Actualment, els altímetres comercials tenen el problema de no tenir moltes funcionalitats per un preu raonable, per això aquest treball pretén fer una solució més econòmica fent un disseny propi, que com-

pleixi les necessitats exactes de les quals es requereix, reduint el cost respecte als productes del mercat.

### 7.3. Dimensió social

**PPP:** *Què creus que t'aportarà a nivell personal la realització del projecte?*

Aquest projecte m'ajudarà a aprendre sobre el món dels altímetres. Tambe aprendré com fer dissenys electrònics de PCB, i a aprofunditzar el meu coneixement de fer programari ben documentat i organitzat.

A part també m'aporta l'experiència de fer un projecte extens i tècnic, que em pot ajudar a fer i planificar millor projectes futurs, evitant possibles problemes.

**Vida Útil:** *Com es resol actualment el problema que vols abordar (estat de l'art)? En què millorarà socialment la teva solució a les existents? Podria el projecte tenir un impacte social negatiu?*

Els altímetres realment no tenen gens d'impacte social, com a màxim en un cas molt exagerat, fan que algunes persones amb un hobby de coeteria s'ho passin bé, i l'únic que pot millorar és que al UPC Space Program, l'altímetre nou els agradi més.

Com ja he remarcat, els altímetres no tenen gens d'impacte social, així que és difícil trobar algun possible cas en què tingui un impacte negatiu.

## 8. Disseny del sistema

### 8.1. Consideracions prèvies

Donat el temps limitat per a les dates de llançament disponibles, s'ha decidit no realitzar la fase de prototipatge amb plaques de coure fresades. En lloc d'això, s'encarregarà una PCB amb el disseny inicial directament. Es provarà si funciona el disseny i, si les funcionalitats essencials del disseny inicial són funcionals, es podrà realitzar una prova en un llançament real abans de la data d'entrega d'aquest treball.

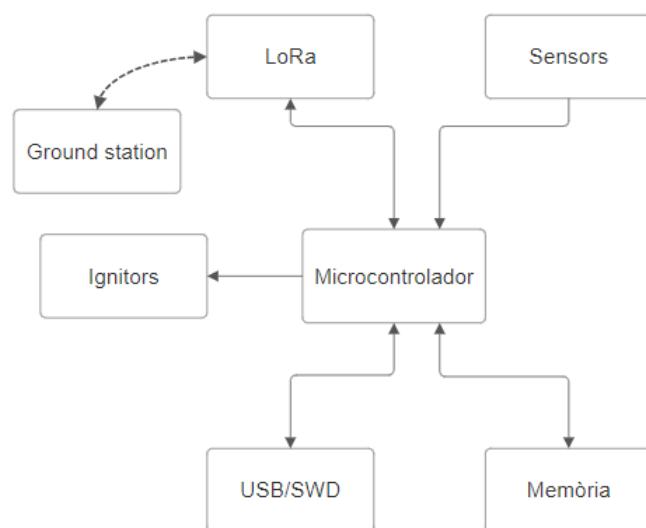
### 8.2. Decisions de disseny

L'altímetre ha de complir els requeriments i funcionalitats llistades a continuació:

- Ha de tenir una amplada inferior a 5 cm i una longitud inferior a 15 cm.
- Ha de disposar d'un circuit elèctric per verificar la continuitat dels ignitors abans d'encendre'ls.
- Ha d'obtenir les dades d'alçada, velocitat, acceleració, orientació i posició.
- Ha de desar les dades obtingudes.
- Ha de ser capaç d'enviar dades a 2 km.

Per això, s'ha decidit que l'altímetre estigui format pels següents components: un microcontrolador, un baròmetre, una IMU (de l'anglès Inertial Measurement Unit) de 9 eixos, un GNSS (de l'anglès global navigation satellite system), un mòdul de comunicació LoRa (Long Range) i una memòria flash.

La Figura 4 mostra el diagrama de blocs del sistema, que connecta tots els mòduls al microcontrolador. Aquest últim sol·licita les dades necessàries als sensors i, dependent de l'etapa de vol, les processa adequadament. També es representa la connexió entre la ground station i l'altímetre, que es realitza mitjançant LoRa, amb un enviament de dades unidireccional, de l'altímetre a la ground station.



**Figura 4:** Diagrama de blocs del sistema. Font: Elaboració pròpia.

L'altímetre funciona en diferents fases, segons l'estat del vol en què es trobi. A continuació s'expliquen les diferents etapes que seguirà l'altímetre:

- Setup: inicialitza el programa i verifica que els components funcionin correctament.
- Preparació: verifica la continuitat dels ignitors, indica la configuració actual amb senyals acústics i espera l'activació de la switch key per passar a la següent fase.
- Rampa: indica amb senyals acústics si es troba en un angle superior a  $30^\circ$ , comença a recollir lectures d'alçada i revisa constantment si el coet s'ha enlairat per passar a la següent fase.
- Vol: recull dades de la IMU, el baròmetre i el GNSS, les desa a la memòria i envia l'alçada a la ground station. La fase acaba quan es detecta l'apogeu.
- Apogeu: encén els ignitors del drogue i espera arribar a una alçada determinada, configurada abans del llançament, per obrir el main. Durant aquesta fase, també recull dades com a la fase anterior.
- Main: encén els ignitors del main i es manté en aquesta fase fins que toca terra. Continua recollint dades com en les dues fases anteriors.
- Terra: deixa de recollir dades de la IMU i el baròmetre, envia les coordenades del GNSS a la ground station i inicia una seqüència sonora que indica l'alçada de l'apogeu, fins que s'apaga la switch key.

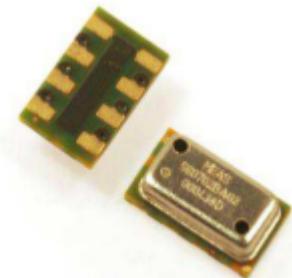
### 8.3. Selecció de components

Un cop establert com es vol dissenyar l'altímetre, cal seleccionar els components, prioritant que siguin petits, econòmics i que disposin de recursos en línia.

Per començar, s'ha decidit que l'últim component a seleccionar serà el microcontrolador, ja que així es podrà veure quins requeriments dels altres components té. A continuació, es presenta una llista dels components principals. A l'Annex: Taules de Components es troba un llistat amb tots els components seleccionats.

#### 8.3.1. Baròmetre

Per al baròmetre, s'ha seleccionat el model MS560702BA03-50 de TE-Connectivity[12], específicament dissenyat per a altímetres. A la Taula 10 es mostren les seves característiques.



**Figura 5:** Mòdul MS560702BA03-50[12]

Característiques	MS560702BA03-50
Corrent	1 $\mu$ A (en espera <0.15 $\mu$ A)
Voltatge	1.8 a 3.6 V
Mida	5.0 x 3.0 x 1.0 mm <sup>3</sup>
Comunicació	I <sup>2</sup> C o SPI
Rang	10 a 2000 mbar, -40 a +85 °C
Precisió	20 cm

**Taula 10:** Característiques del mòdul MS560702BA03-50. Font: Elaboració pròpia

### 8.3.2. IMU

Per a la IMU, s'ha optat pel ICM-20948 de TDK Invensense[13], per les seves dimensions reduïdes i baix consum. A la Taula 11 es mostren les seves característiques.



**Figura 6:** Mòdul ICM-20948[13]

Característiques	ICM-20948
Corrent	3,11 mA (en espera 8 $\mu$ A)
Voltatge	1.71V a 3.6V
Mida	3.0 x 3.0 x 1.0 mm <sup>3</sup>
Comunicació	I <sup>2</sup> C o SPI
Rang	$\pm$ 2000 dps, $\pm$ 16g

**Taula 11:** Característiques del mòdul ICM-20948.

Font: Elaboració pròpria

### 8.3.3. Memòria flash

Per a la memòria flash, s'ha seleccionat la W25Q128JV de Winbond[14]. A la Taula 12 es mostren les seves característiques.



**Figura 7:** Mòdul W25Q128JV[14]

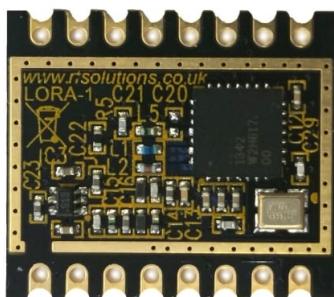
Característiques	ICM-20948
Corrent	25 mA (en espera 60 $\mu$ A)
Voltatge	2.7V a 3.6V
Mida	5.3 x 7.9 x 1.8 mm <sup>3</sup>
Comunicació	SPI, dual SPI o quad SPI
Capacitat	128 Mb

**Taula 12:** Característiques del mòdul W25Q128JV.

Font: Elaboració pròpria

### 8.3.4. LoRa

Per al mòdul LoRa, s'ha seleccionat el RF-LORA-868-SO de RF Solutions[15], ja que opera a 100 mW, dins del límit màxim de potència permès per la Unió Europea[16] de 500 mW. Aquesta potència teòricament arriba a fins a 16 km, complint així el requisit de 2 km. A la Taula 13 es mostren les seves característiques.



**Figura 8:** Mòdul RF-LORA-868-SO[15]

Característiques	RF-LORA-868-SO
Corrent	125 mA (en espera 1.5 $\mu$ A)
Voltatge	1.8V a 3.6V
Mida	23 x 20 x 2 mm <sup>3</sup>
Comunicació	SPI
Potència	100 mW

**Taula 13:** Característiques del mòdul RF-LORA-868-SO. Font: Elaboració pròpria

### 8.3.5. GNSS

Per al GNSS, s'ha optat pel SIM68M de SIMCOM[17]. A la Taula 14 es mostren les seves característiques.



**Figura 9:** Mòdul SIM68M[17]

Característiques	SIM68M
Corrent	25 mA (en espera 14 $\mu$ A)
Voltatge	2.4V a 4.3V
Mida	10.1 x 9.7 x 2.5 mm <sup>3</sup>
Comunicació	UART
Precisió	<2.5 m

**Taula 14:** Característiques del mòdul SIM68M. Font: Elaboració pròpria

### 8.3.6. Microcontrolador

Havent seleccionat els altres components, ara podem veure que el microcontrolador necessita tenir suport per UART, 2 SPI i 2 I<sup>2</sup>C, i ha de tenir suficients pins per poder afegir funcionalitats. Per això, s'ha seleccionat el STM32F401VET6 de ST-Microelectronics[18], que compleix aquests requisits, consumeix poca energia i, amb 100 pins, en disposa suficients per a futures ampliacions. A mes des de l'associació s'ha volgut fer servir microcontroladors de la gamma STM32 dades de fa temps, però no hi ha hagut ningú que s'atrevis a començar. Els motius principals per optar per STM32 són que compten amb un ampli suport tant de la comunitat com de la documentació oficial, i ofereixen eines de programació molt còmodes, com el STMCubeIDE i que utilitzen les instruccions estàndard d'ARM.



**Figura 10:** Mòdul STM32F401VET6[18]

Característiques	STM32F401VET6
Corrent	146 $\mu$ A (en espera 2.4 $\mu$ A)
Voltatge	1.7V a 3.6V
Mida	14 x 14 x 1.6 mm <sup>3</sup>
Comunicació	3xI <sup>2</sup> C, 3xUART, 4xSPI, SDIO, USB 2.0
Memòria	512 Kb

**Taula 15:** Característiques del microcontrolador STM32F401VET6. Font: Elaboració pròpria

## 9. Disseny de la placa

Com ja s'ha vist quines característiques i quins components ha de tenir l'altímetre, el següent pas és dissenyar la placa, que comença amb els esquemàtics, per poder dissenyar la PCB.

### 9.1. Esquemàtics

El disseny dels esquemàtics s'ha fet de forma similar a la selecció de components, deixant el microcontrolador per al final per escollir quines connexions són les adequades per a cada component.

#### 9.1.1. Baròmetre

El baròmetre utilitzarà el protocol I<sup>2</sup>C segons la documentació del fabricant[19], on hi ha un esquemàtic de referència del baròmetre, com es pot veure a la Figura 11. A la Figura 12 es pot veure l'esquemàtic del baròmetre implementat.

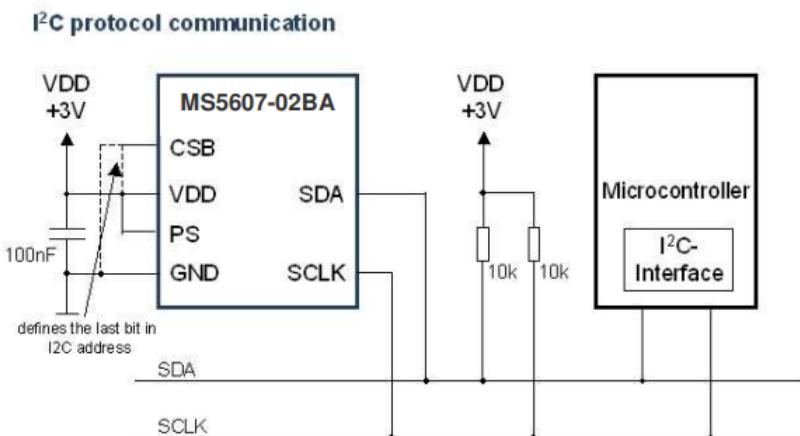


Figura 11: Esquemàtic de referència del baròmetre[19].

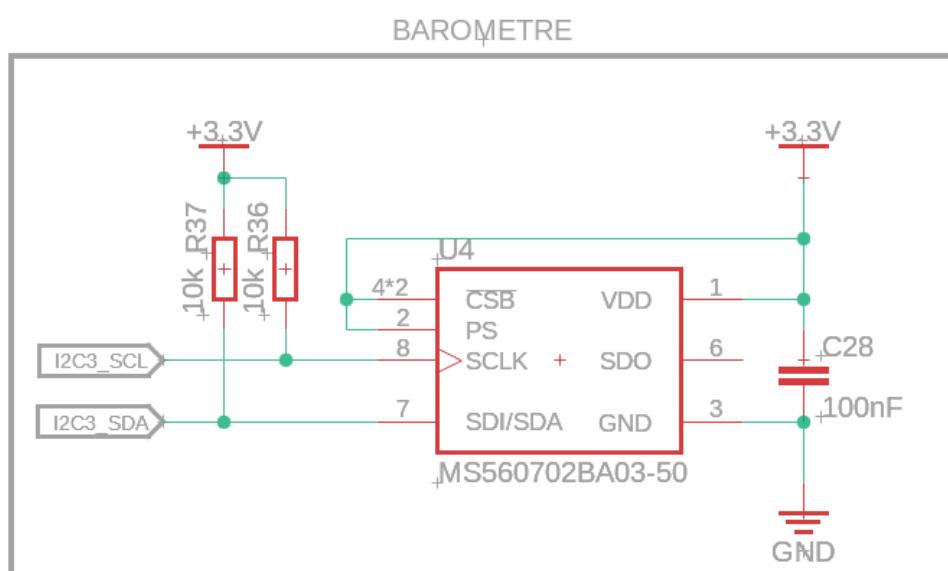
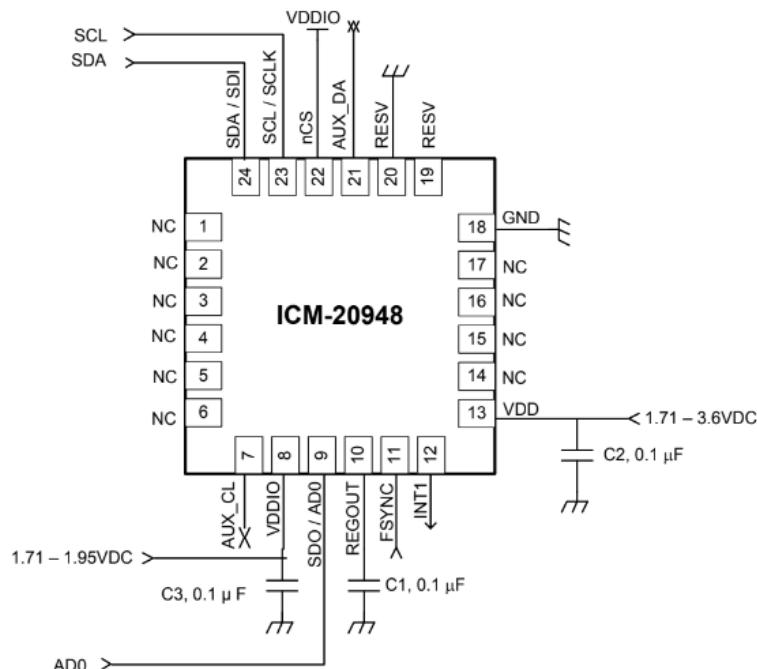


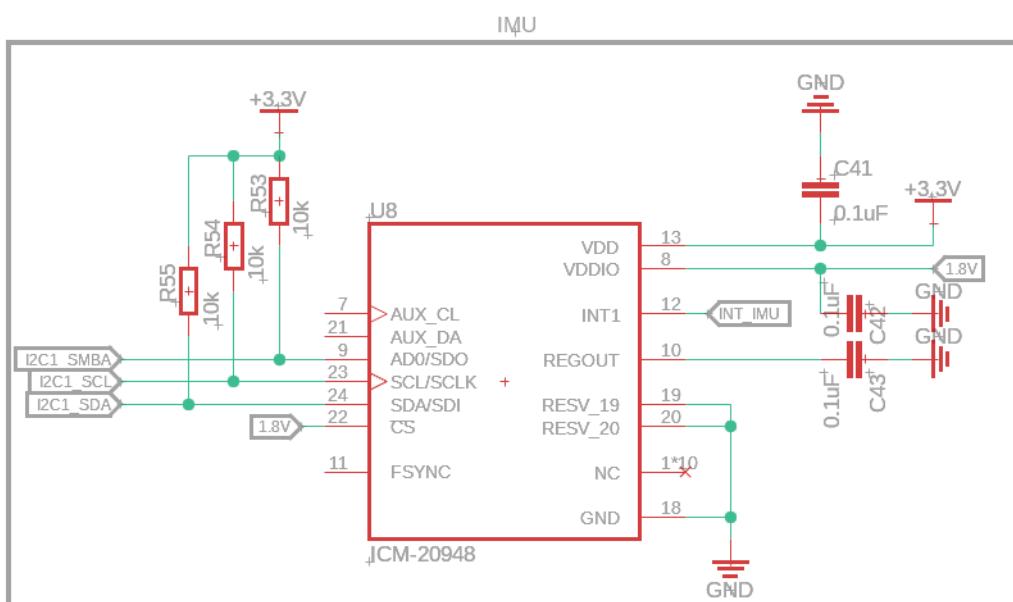
Figura 12: Esquemàtic del baròmetre. Font: Elaboració pròpia.

### 9.1.2. IMU

La IMU també utilitza el protocol I<sup>2</sup>C. Com en el cas del baròmetre, s'ha proporcionat un esquemàtic de referència per a la IMU[20], el qual es pot observar a la Figura 13. La Figura 14 mostra l'esquemàtic de la IMU que s'ha dissenyat.



**Figura 13:** Esquemàtic de referència de la IMU[20].



**Figura 14:** Esquemàtic de la IMU. Font: Elaboració pròpia.

### 9.1.3. Memòria flash

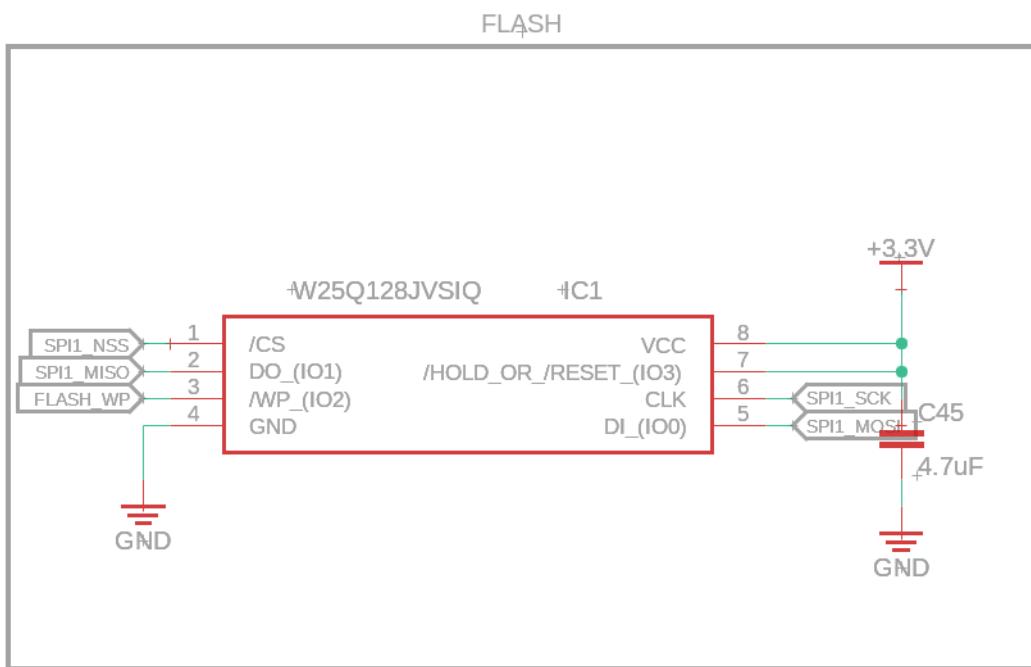
La memòria flash utilitza SPI, i ofereix opcions de dual i quad SPI. S'ha decidit utilitzar el SPI simple, ja que és el protocol més senzill i ofereix bones velocitats de transmissió. A diferència dels components anteriors, no s'ha proporcionat cap esquemàtic de referència. Per tant, a partir de la descripció dels pins de la memòria flash a la Figura 15, s'ha creat l'esquemàtic que es mostra a la Figura 16.

PAD NO.	PAD NAME	I/O	FUNCTION
1	/CS	I	Chip Select Input
2	DO (IO1)	I/O	Data Output (Data Input Output 1) <sup>(1)</sup>
3	/WP (IO2)	I/O	Write Protect Input ( Data Input Output 2) <sup>(2)</sup>
4	GND		Ground
5	DI (IO0)	I/O	Data Input (Data Input Output 0) <sup>(1)</sup>
6	CLK	I	Serial Clock Input
7	/HOLD or /RESET (IO3)	I/O	Hold or Reset Input (Data Input Output 3) <sup>(2)</sup>
8	VCC		Power Supply

**Notes:**

1. IO0 and IO1 are used for Standard and Dual SPI instructions
2. IO0 – IO3 are used for Quad SPI instructions, /HOLD (or /RESET) function is only available for Standard/Dual SPI.

**Figura 15:** Descripció dels pins de la memòria flash[21].



**Figura 16:** Esquemàtic de la memòria flash. Font: Elaboració pròpia.

### 9.1.4. LoRa

El mòdul LoRa també utilitza SPI, i al seu datasheet[22] hi ha un exemple de funcionament, amb un PIC, que es pot veure a la Figura 17, el qual s'ha utilitzat com a referència per dissenyar l'esquemàtic

de la Figura 18.

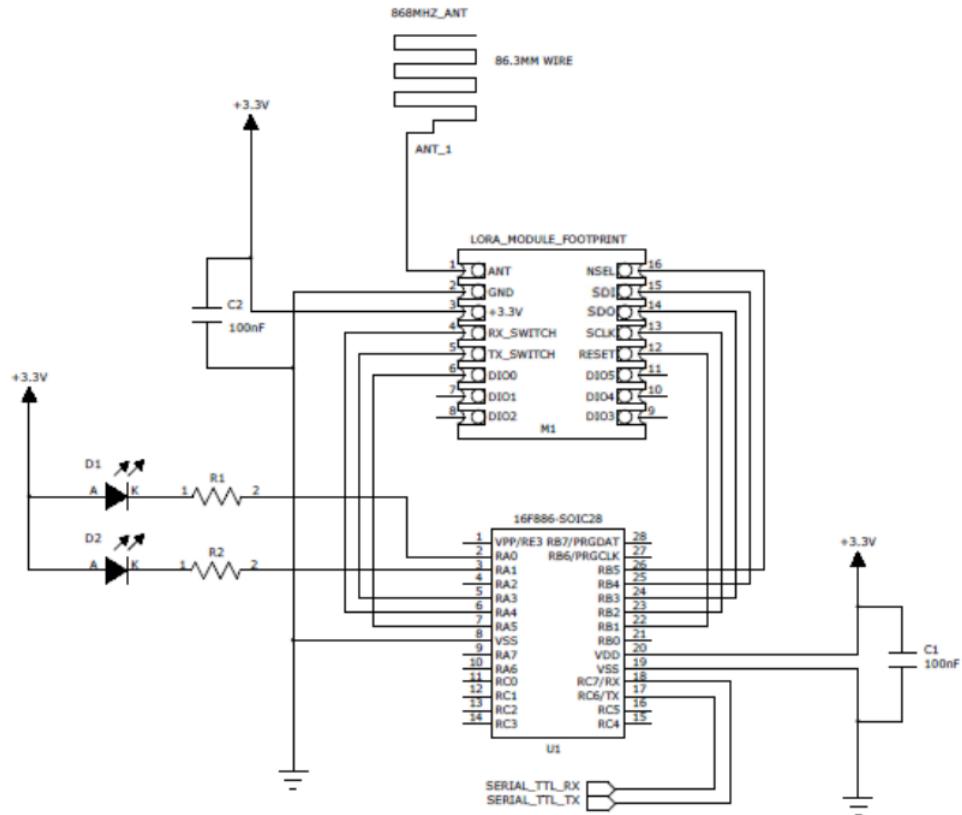
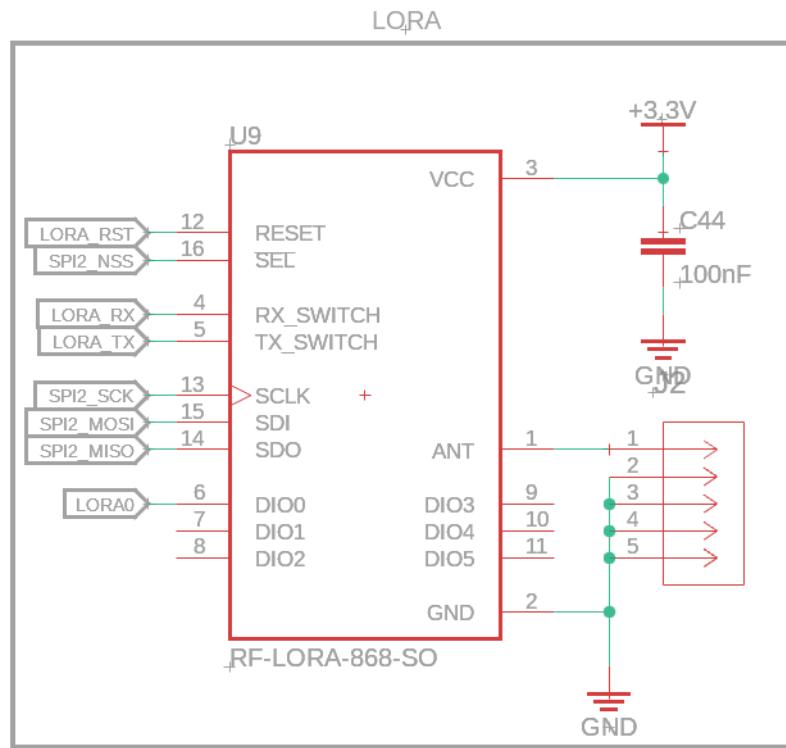


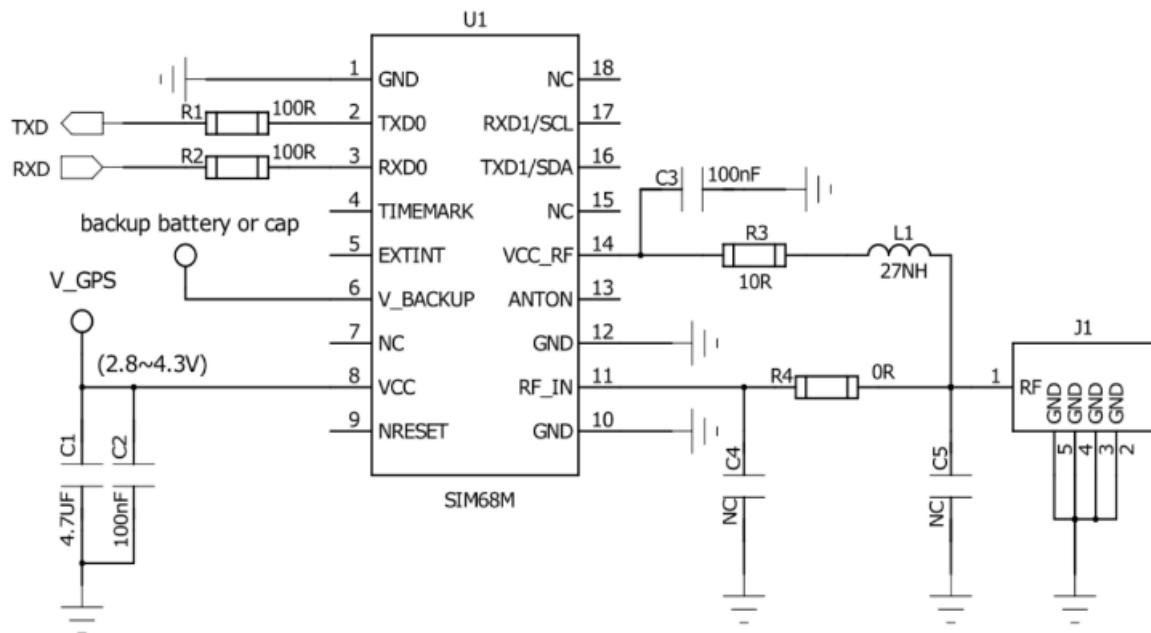
Figura 17: Exemple de funcionament del mòdul LoRa[22].



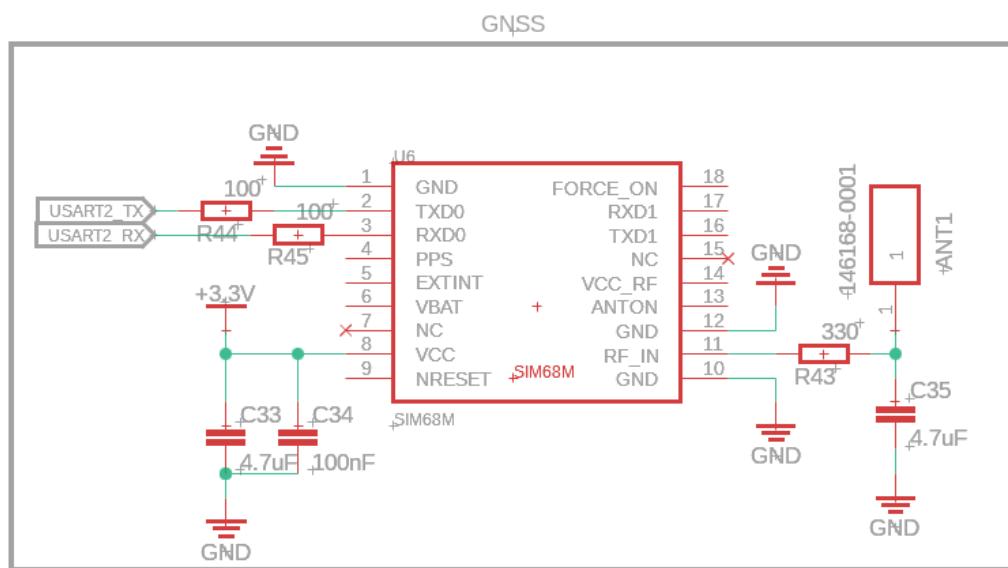
**Figura 18:** Esquemàtic del mòdul LoRa. Font: Elaboració pròpria.

#### 9.1.5. GNSS

El receptor GNSS utilitza comunicació UART per rebre les dades de posicionament. Pel GNSS també s'ha proporcionat un esquemàtic de referència[23]. A la Figura 19 es pot observar aquest esquemàtic de referència, el qual ha servit per al disseny del esquemàtic de la Figura 20.



**Figura 19:** Esquemàtic de referència del GNSS[23].



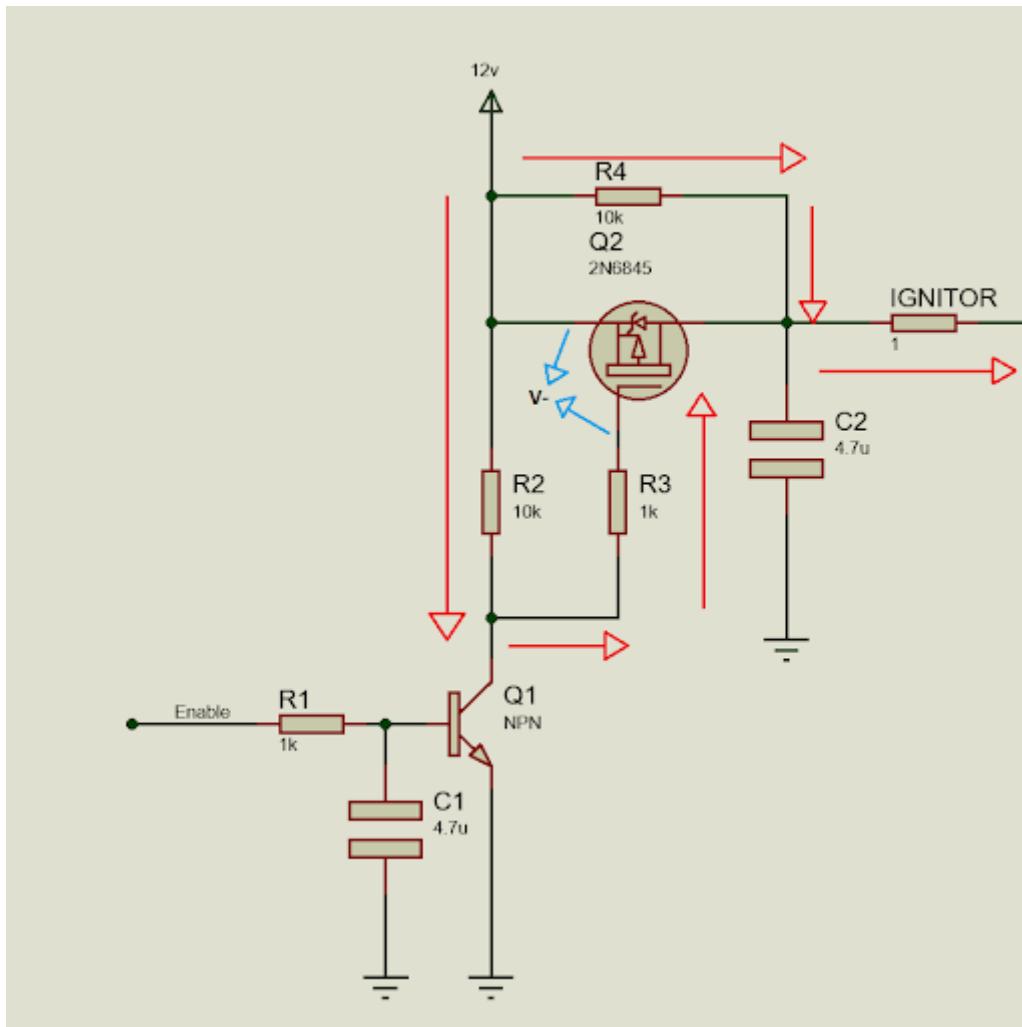
**Figura 20:** Esquemàtic del GNSS. Font: Elaboració pròpia.

### 9.1.6. Circuits d'ignició

Els ignitors per un coet poden fallar de diverses maneres, per tant, és crucial tenir un sistema per detectar la continuïtat i assegurar-se que funcionaran durant el llançament.

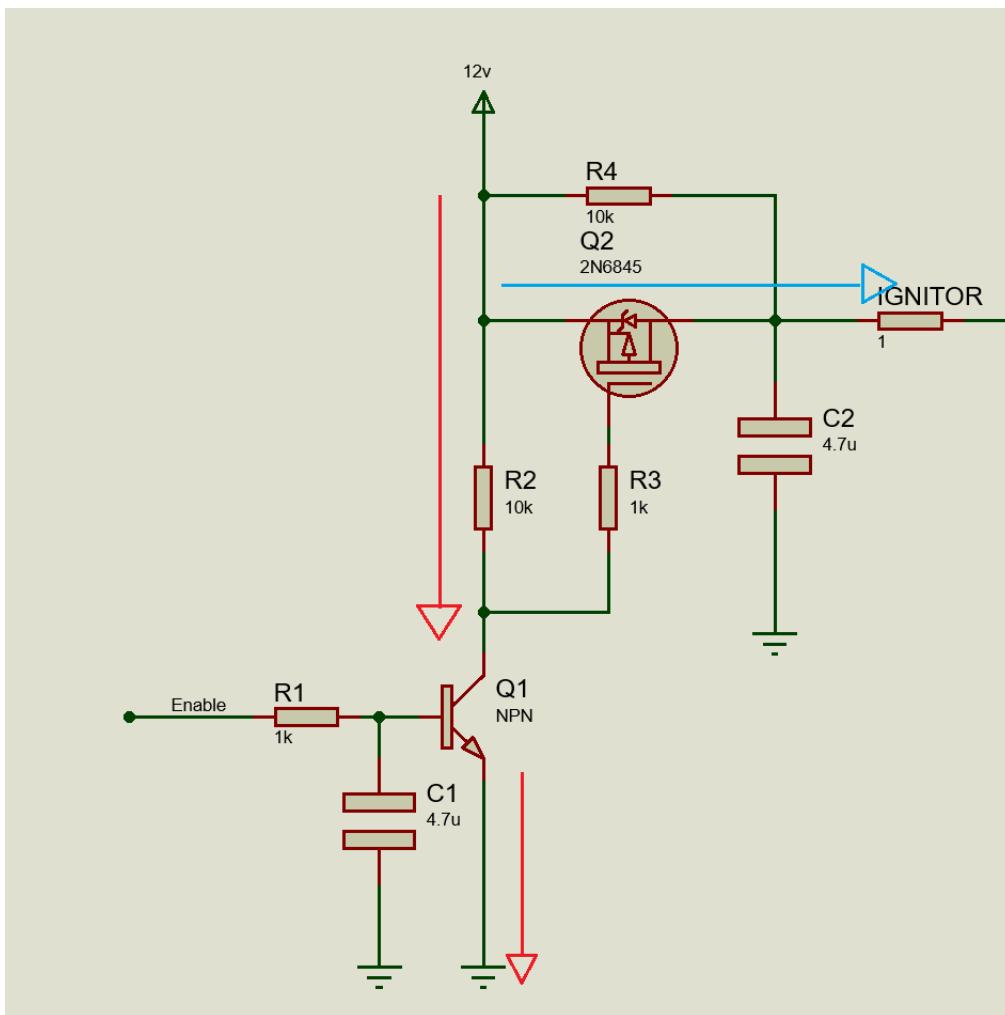
Per començar, es necessita dissenyar un circuit per controlar la quantitat de corrent que passa a través de l'ignitor. Aquest circuit consta d'un transistor NPN connectat a un pin del MCU, un MOSFET amb canal P, una sèrie de resistències i un parell de condensadors per filtrar el soroll dels senyals elèctrics.

Mirant l'esquemàtic de la Figura 21, quan el pin del MCU, anomenat Enable, emet un senyal digital baix, el transistor NPN està tancat, i per tant, el corrent arribarà a la base del MOSFET. Però, com passa a través de les resistències, el voltatge entre el source i la base serà negatiu, mantenint el MOSFET tancat. Això permet que el voltatge arribi a l'ignitor a través de la resistència R4, la qual, al ser alta, redueix suficientment el voltatge per evitar que l'ignitor s'encengui.



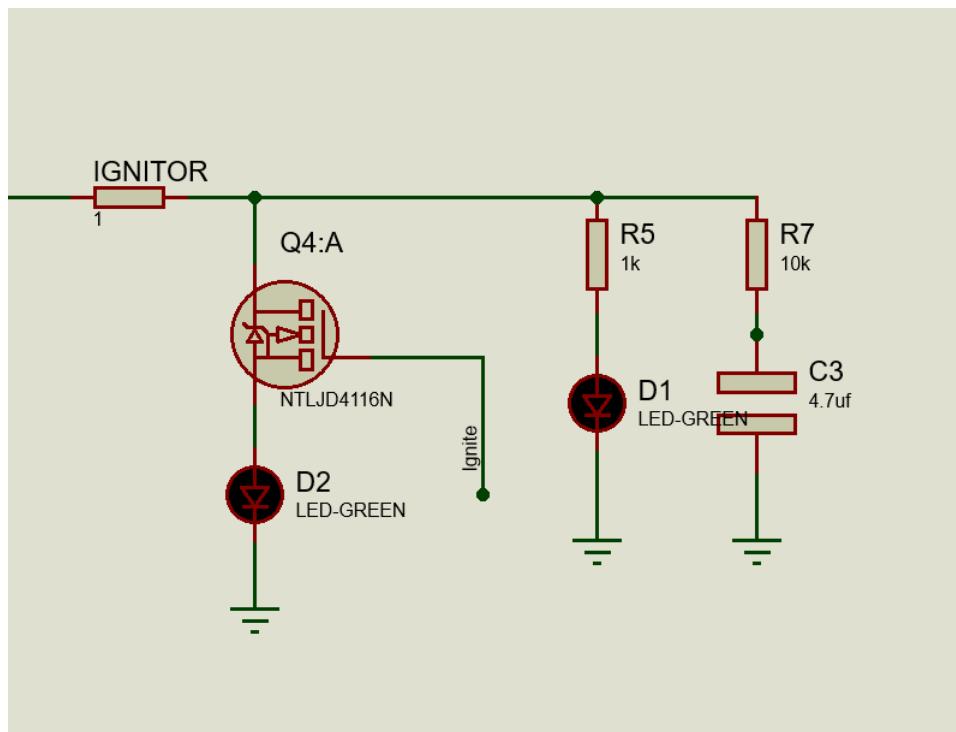
**Figura 21:** Esquemàtic de la part esquerra del circuit d'ignició, amb l'Enable a zero. Font: Elaboració pròpia.

Quan el pin Enable emet un senyal digital alt, el transistor NPN permet el pas del corrent directament a terra, evitant que arribi corrent a la base del MOSFET. Així, el MOSFET permet que passi corrent entre el source i el drain, permetent que gairebé tots els 12V arribin a l'ignitor. Aquest comportament està representat a la Figura 22.



**Figura 22:** Esquemàtic de la part esquerra del circuit d'ignició, amb l'Enable a nivell alt. Font: Elaboració pròpria.

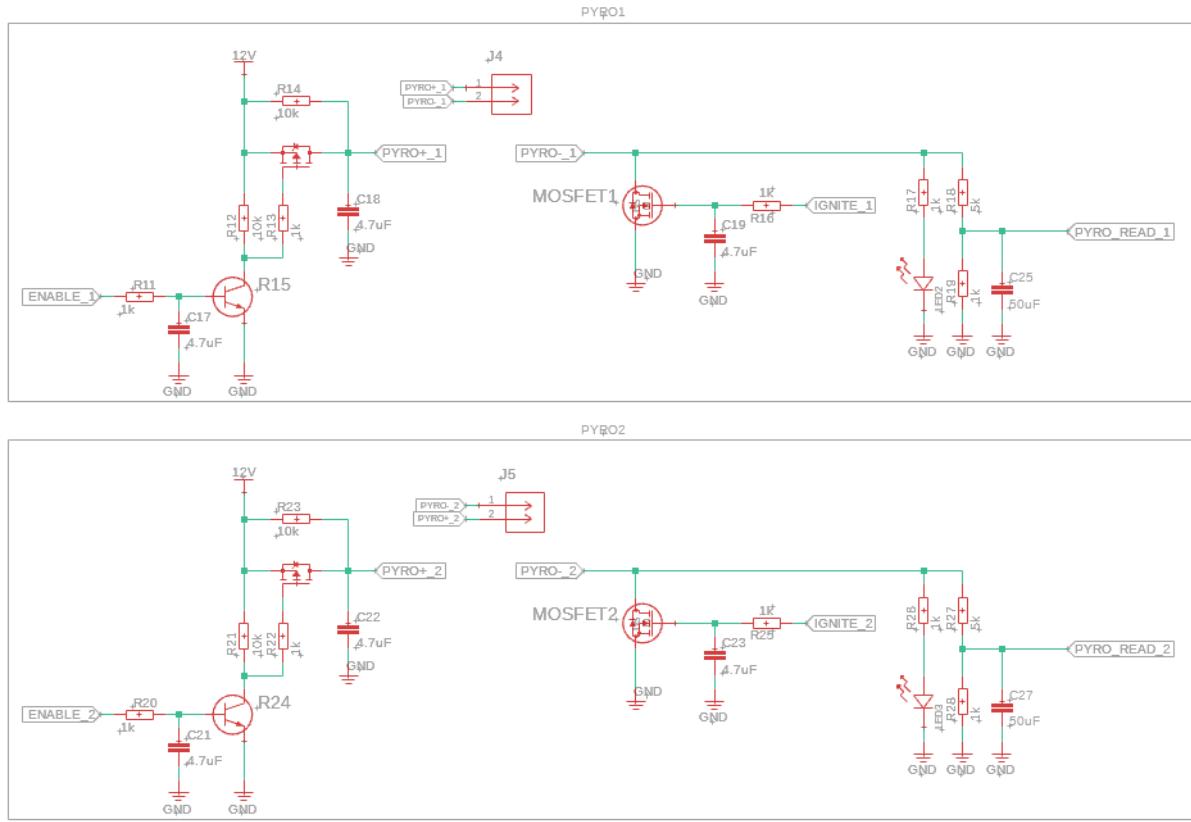
La Figura 23 mostra l'esquemàtic de l'altra banda de l'ignitor, on hi ha un altre pin del MCU que controla si deixa passar corrent o no a través d'un MOSFET de canal P. Quan aquest permet el pas del corrent i l'Enable està activat, s'encén l'ignitor.



**Figura 23:** Esquemàtic de la part dreta del circuit d'ignició. Font: Elaboració pròpia.

A la dreta del sistema d'ignició, es troba la detecció de continuïtat, que encén un LED quan hi ha continuïtat. També hi ha un altre pin del MCU entre R7 i R5 per mesurar el corrent que arriba mitjançant un ADC, i així poder detectar la continuïtat.

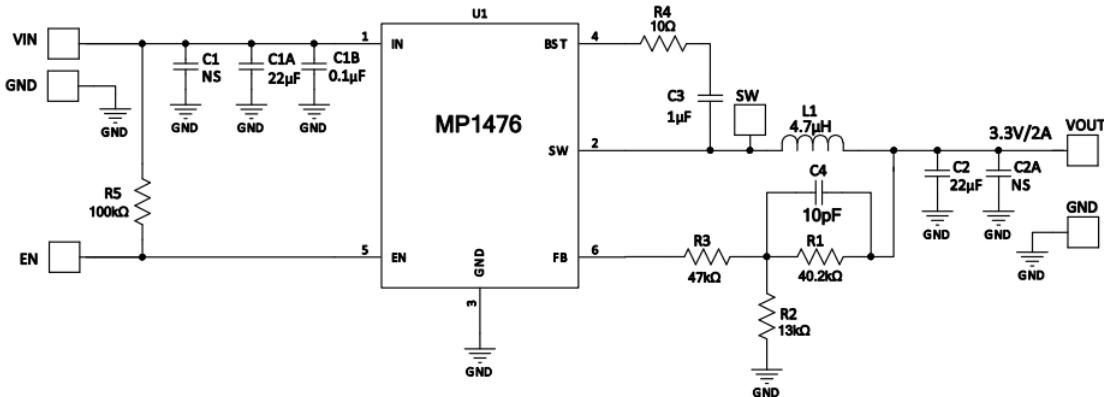
A la Figura 24 es mostren els esquemàtics dels dos sistemes d'ignició de l'altímetre.



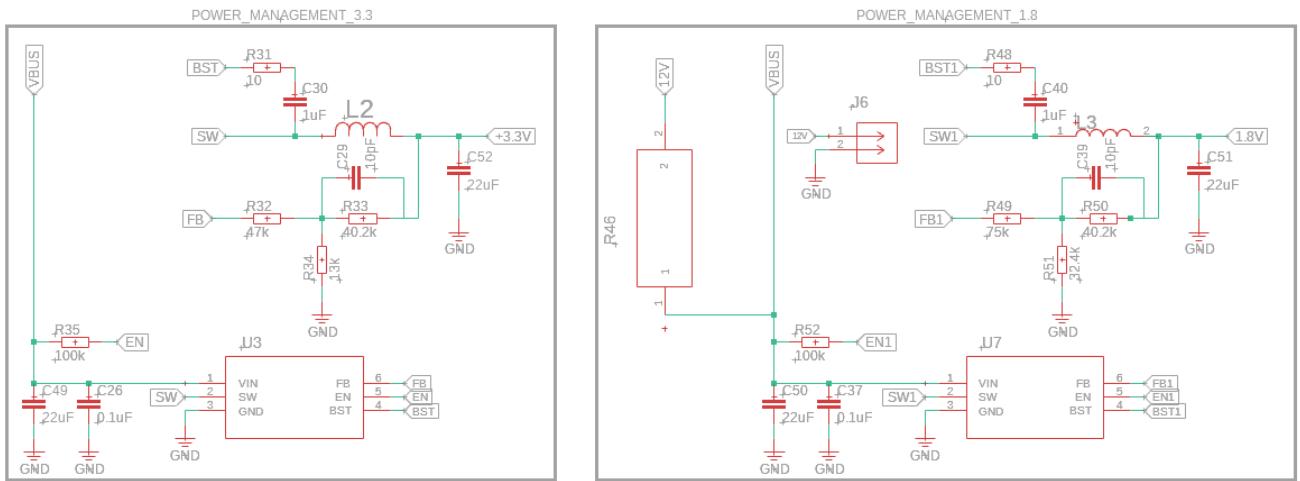
**Figura 24:** Esquemàtic dels circuits d'ignició. Font: Elaboració pròpria.

### 9.1.7. Regulador de tensió

El circuit de regulació de tensió està basat en l'integrat MP1476[24], consistent en un convertidor reductor síncron d'alta eficiència. Aquest dispositiu està molt ben documentat i conté esquemàtics de referència, com el mostrat a la Figura 25, per a diverses sortides de voltatge. A la Figura 26, es presenten els esquemàtics desenvolupats, dels quals en són necessaris dos: un per a 3.3V, que serà utilitzat per la majoria de la placa, i un altre de 1.8V requerit per la IMU.



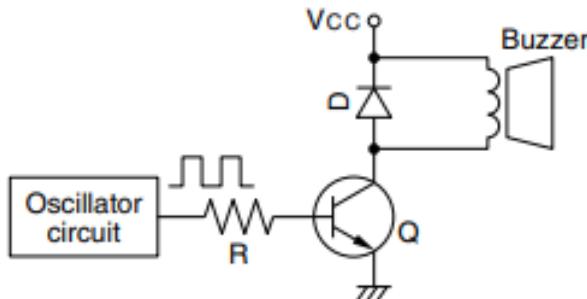
**Figura 25:** Esquemàtic de referència del regulador de tensió [24].



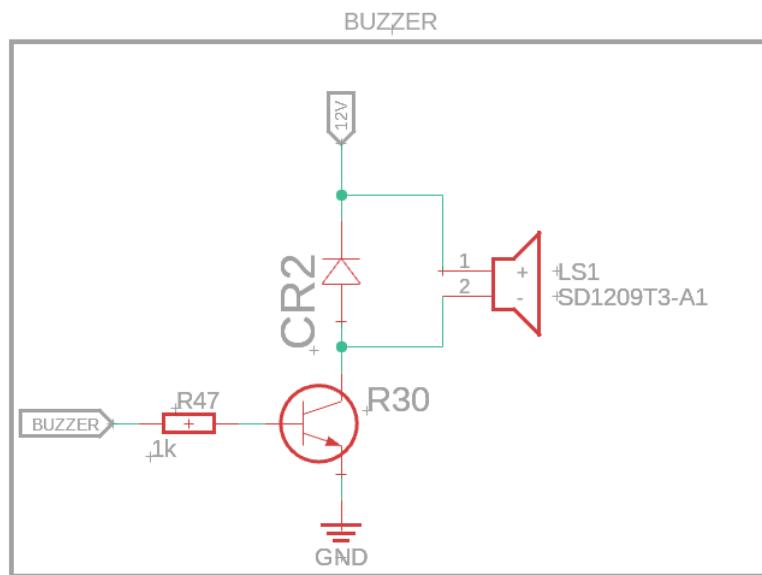
**Figura 26:** Esquemàtic del regulador de tensió. Font: Elaboració pròpria.

### 9.1.8. Buzzer

Per al buzzer s'ha utilitzat el circuit recomanat pel datasheet [25], que es mostra a la Figura 27. Aquest circuit controla la senyal amb un PWM que activa i desactiva un transistor per generar diferents sons. A la Figura 28 es pot observar l'esquemàtic desenvolupat.



**Figura 27:** Esquemàtic recomanat del buzzer [25].



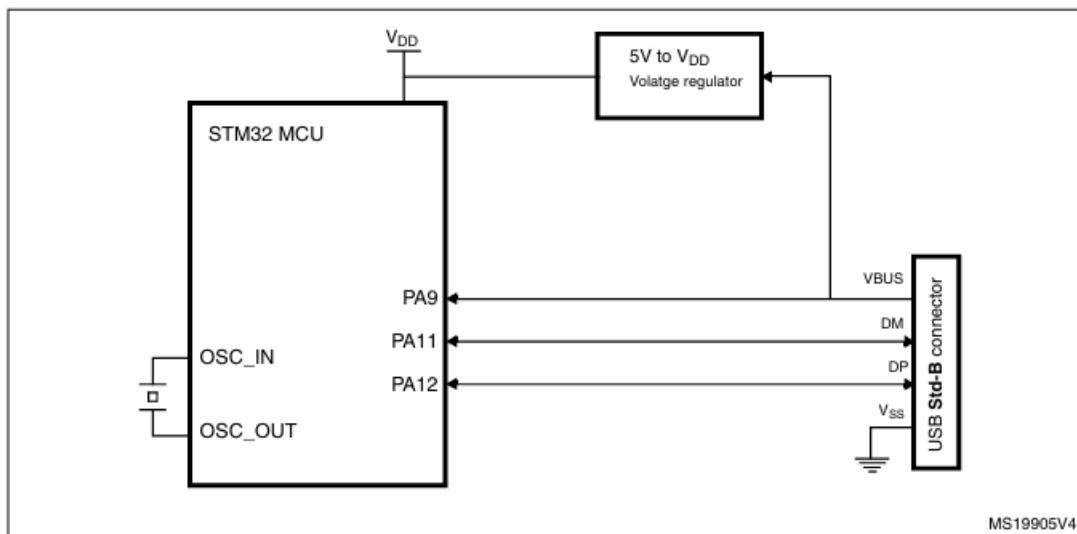
**Figura 28:** Esquemàtic del buzzer. Font: Elaboració pròpria.

### 9.1.9. Connexions externes

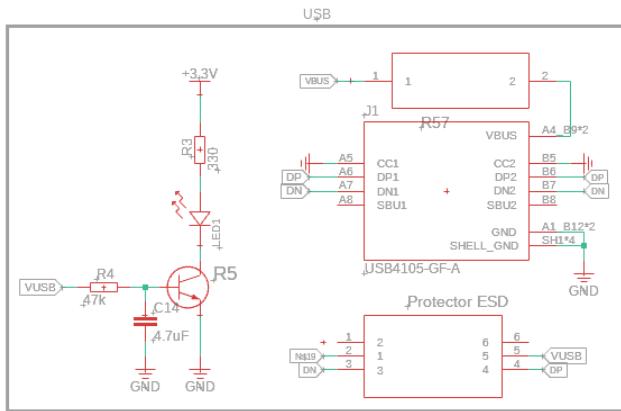
Per a facilitar la connectivitat externa, s'han incorporat un port USB, que servira per poder rebre dades del microcontrolador, i una interfície STLink[26], per programar i debuggar el microcontrolador.

Pel port USB, STM [27] proporciona documentació sobre com s'utilitza, així com esquemàtics per a cada funció. Per la funció que ha de fer l'USB a l'altímetre, l'esquemàtic més adequat és el que es mostra a la Figura 29. Per a l'esquemàtic final, representat a la Figura 30, s'ha afegit un protector de descàrregues electroestàtiques i un circuit per encendre un LED quan es connecta el USB.

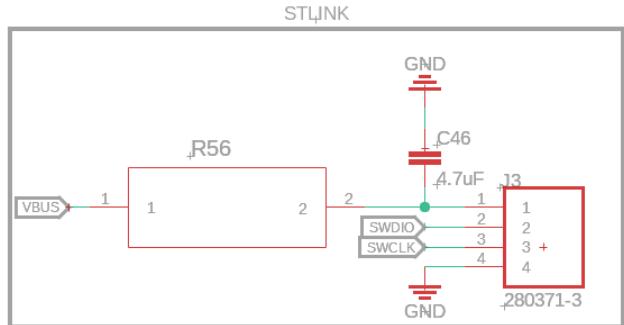
Pel que fa al VSTLink, com es pot veure a la Figura 31, l'esquemàtic és senzill, amb només dues connexions, un diode protector i una connexió a terra.



**Figura 29:** Esquemàtic de referència del USB [27].



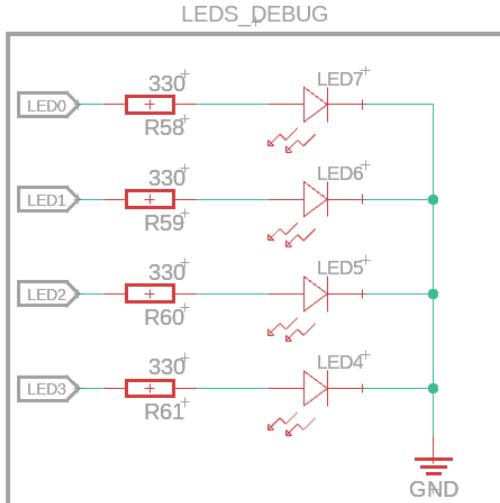
**Figura 30:** Esquemàtic del USB. Font: Elaboració pròpria.



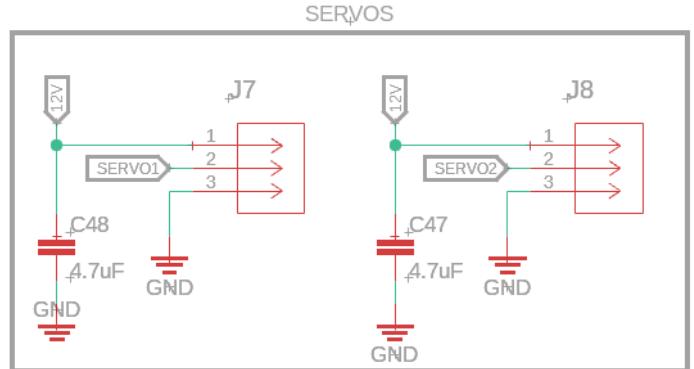
**Figura 31:** Esquemàtic del VSTlink. Font: Elaboració pròpria.

### 9.1.10. Extres

S'han afegit quatre LEDs per a la depuració, com es mostra a la Figura 32, i dos ports per a servos per a futures aplicacions possibles, com s'observa a la Figura 33.



**Figura 32:** Esquemàtic dels LEDs. Font: Elaboració pròpria.



**Figura 33:** Esquemàtic dels servos. Font: Elaboració pròpria.

### 9.1.11. Microcontrolador

El microcontrolador també té un esquemàtic de referència en el seu datasheet [28], com es pot veure a la Figura 34, que indica els circuits d'alimentació necessaris, així com l'esquemàtic de reset mostrat a la Figura 35. Tenint en compte aquestes referències i les connexions necessàries dels esquemàtics anteriors, s'ha elaborat l'esquemàtic representat a la Figura 36.

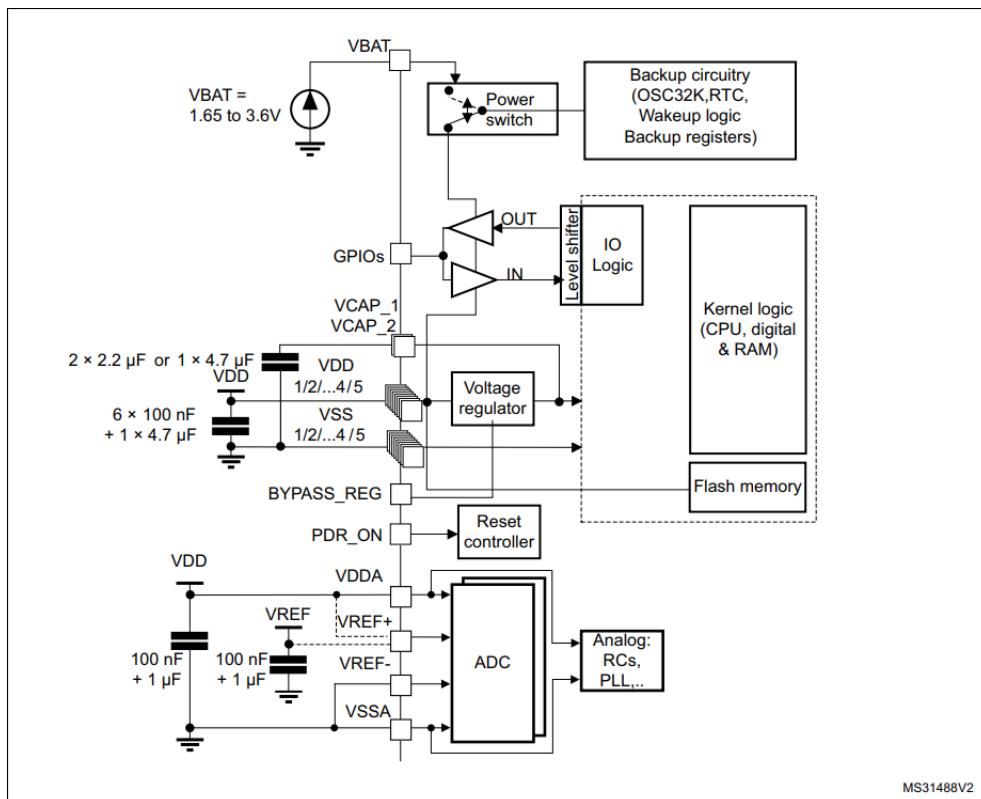


Figura 34: Esquemàtic de referència del microcontrolador [28].

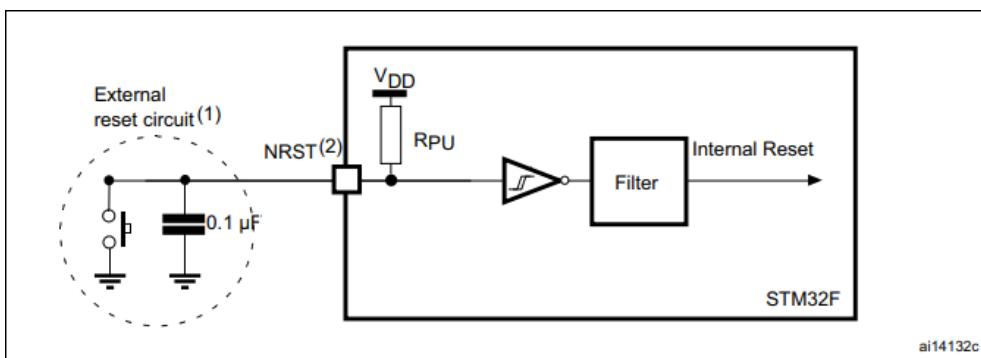
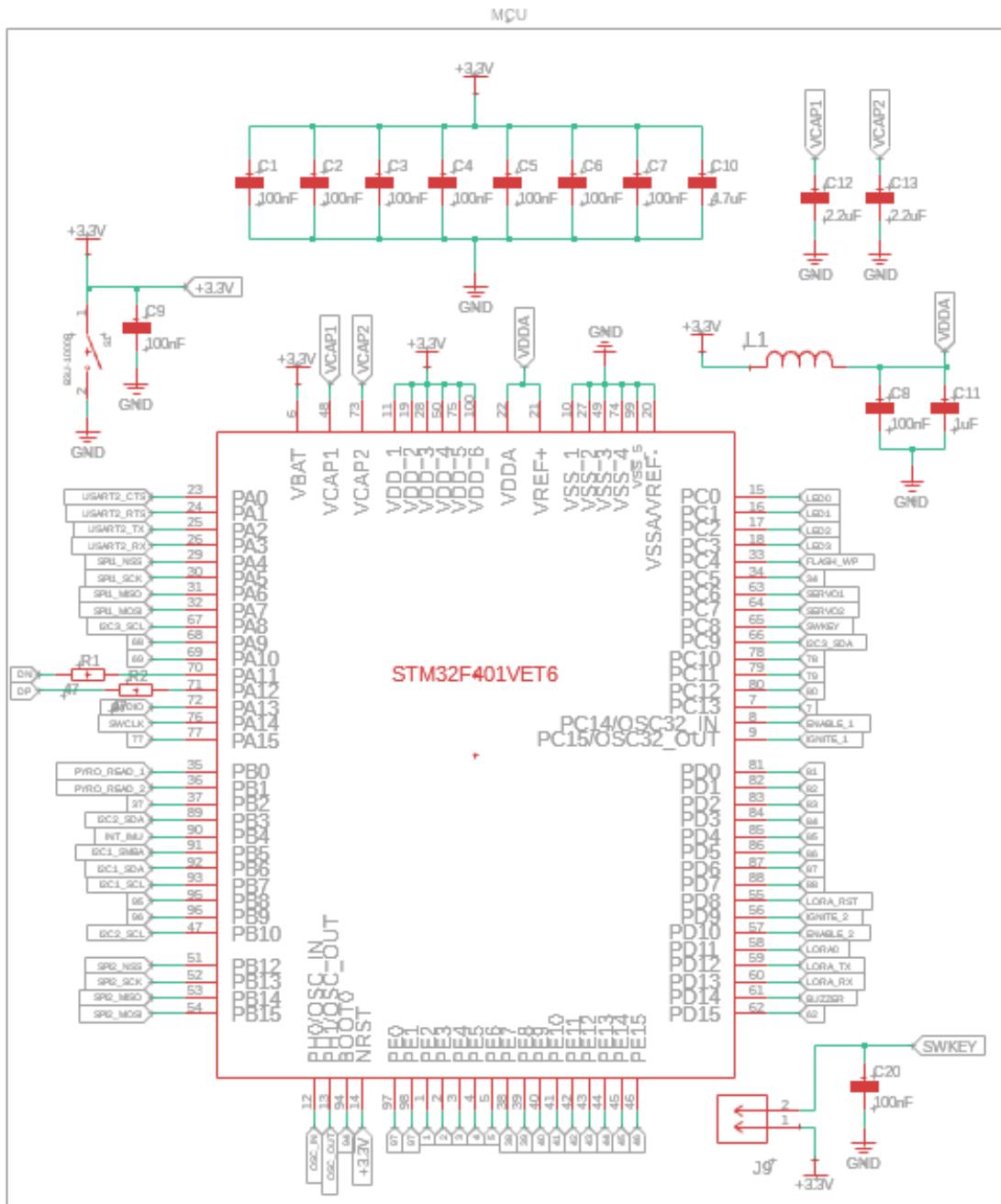


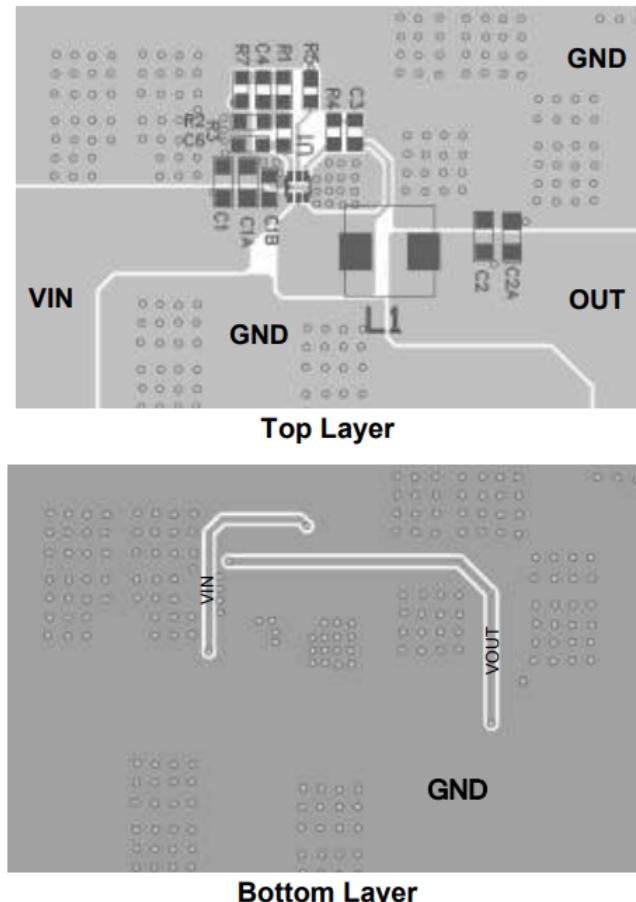
Figura 35: Esquemàtic de referència del reset [28].



**Figura 36:** Esquemàtic del microcontrolador. Font: Elaboració pròpria.

## 9.2. PCB

En general els components només donen direccions generals pel seu disseny a la PCB, a excepció del regulador de tensió (Figura 37).

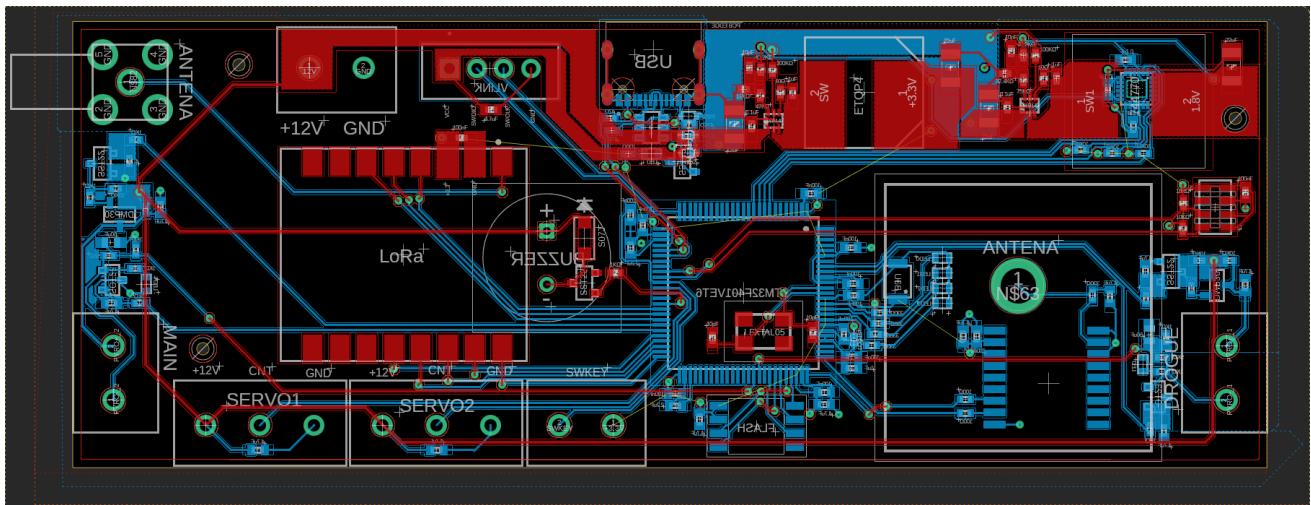


**Figura 37:** Disseny de disposició recomanat del regulador de tensió[24].

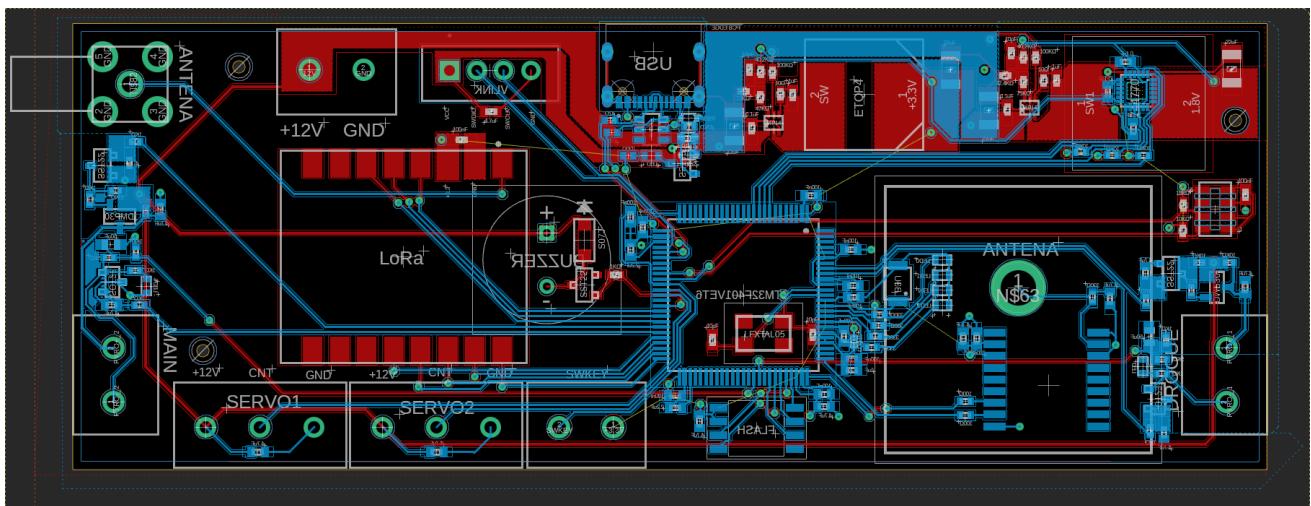
Per tant, l'objectiu principal ha estat minimitzar la mida de la placa, tenint en compte diversos factors:

- **Col · locació dels condensadors:** A prop dels pins d'alimentació per a un funcionament òptim i per reduir les interferències electromagnètiques.
- **Accés a l'aire del baròmetre:** S'ha de garantir que el baròmetre tingui accés a aire net sense tocar altres components, per la qual cosa la placa ha de mantenir una distància adequada.
- **PCB de quatre capes:** S'ha optat per una PCB de quatre capes per simplificar el disseny (més opcions de col · locació i connexions), reduir la mida de la placa (principalment en l'eix horitzontal) tot i que això augmenta el cost de producció.

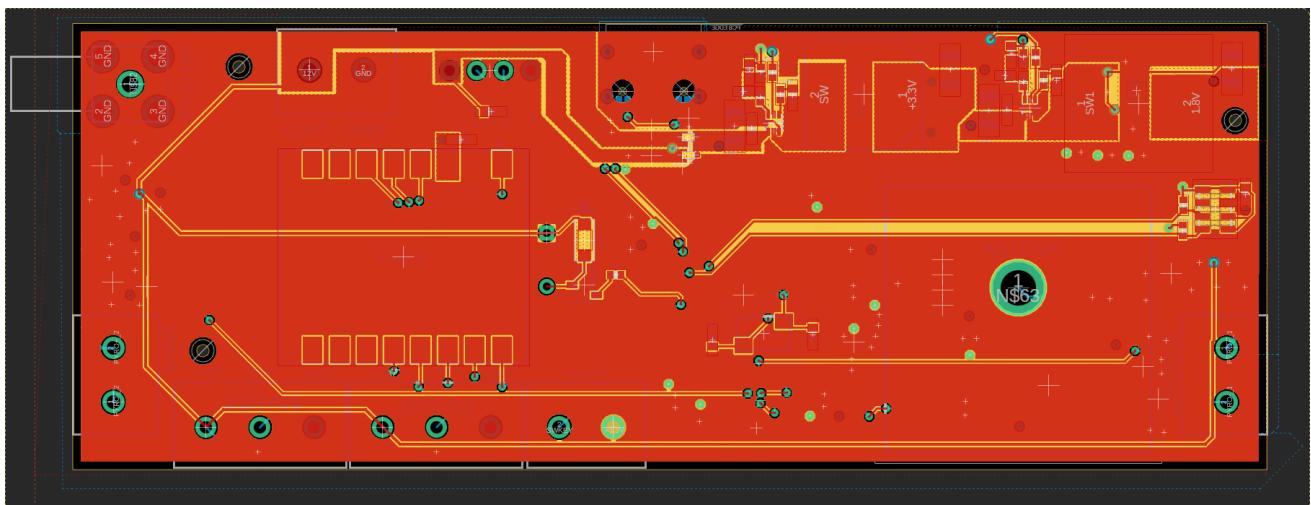
Les següents figures mostren el disseny de la PCB per capes, oferint una visió completa de l'organització dels components i les connexions.



**Figura 38:** Vista top de la PCB. Font: Elaboració pròpria.



**Figura 39:** Vista bottom de la PCB. Font: Elaboració pròpria.



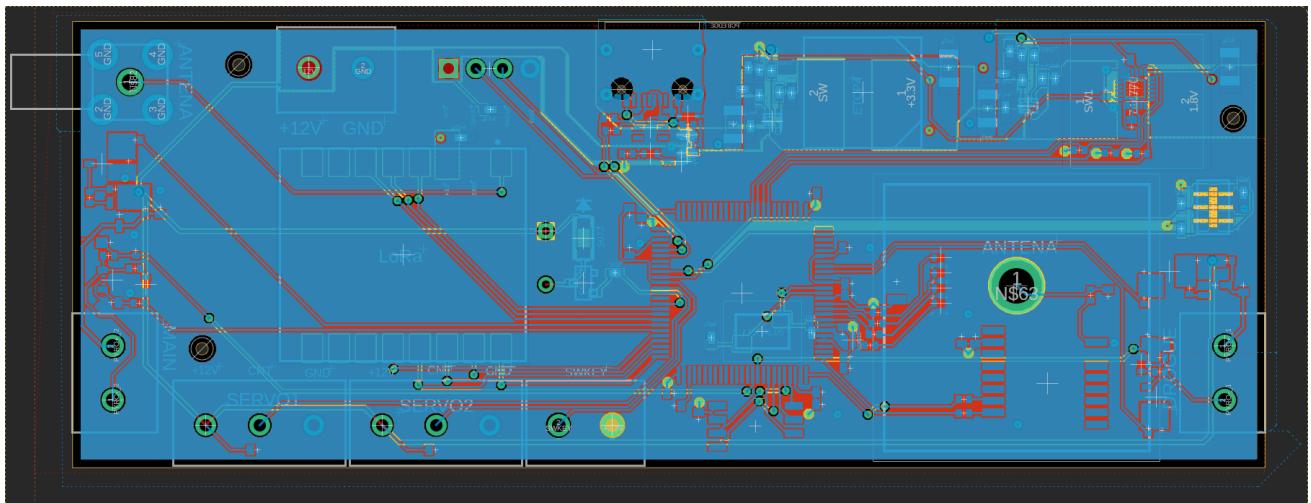
**Figura 40:** Capa top de la PCB. Font: Elaboració pròpria.



**Figura 41:** Capa 3.3V de la PCB. Font: Elaboració pròpria.



**Figura 42:** Capa GND de la PCB. Font: Elaboració pròpria.

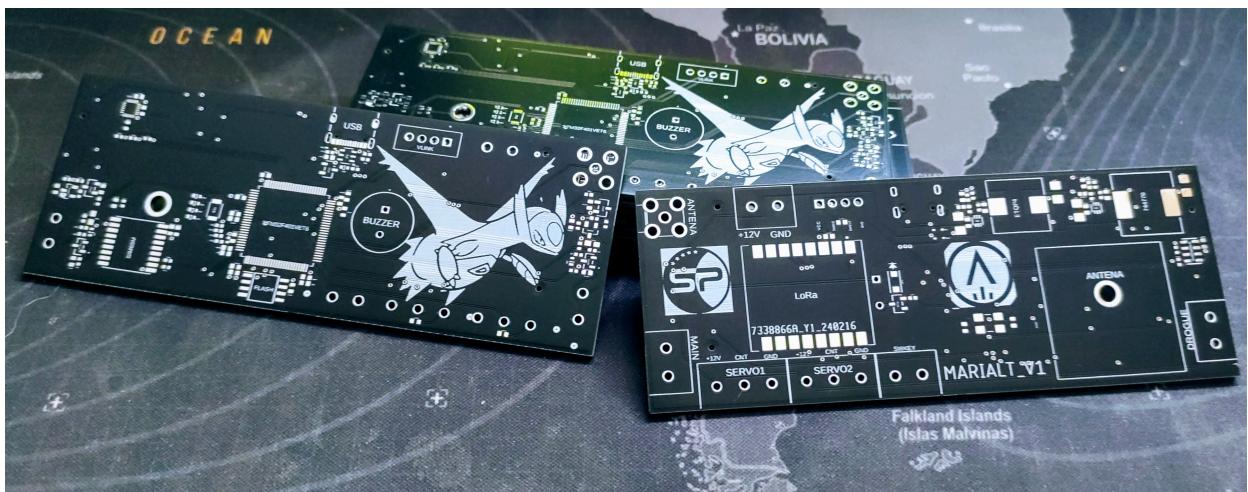


**Figura 43:** Capa bottom de la PCB. Font: Elaboració pròpria.

## 10. Fabricació

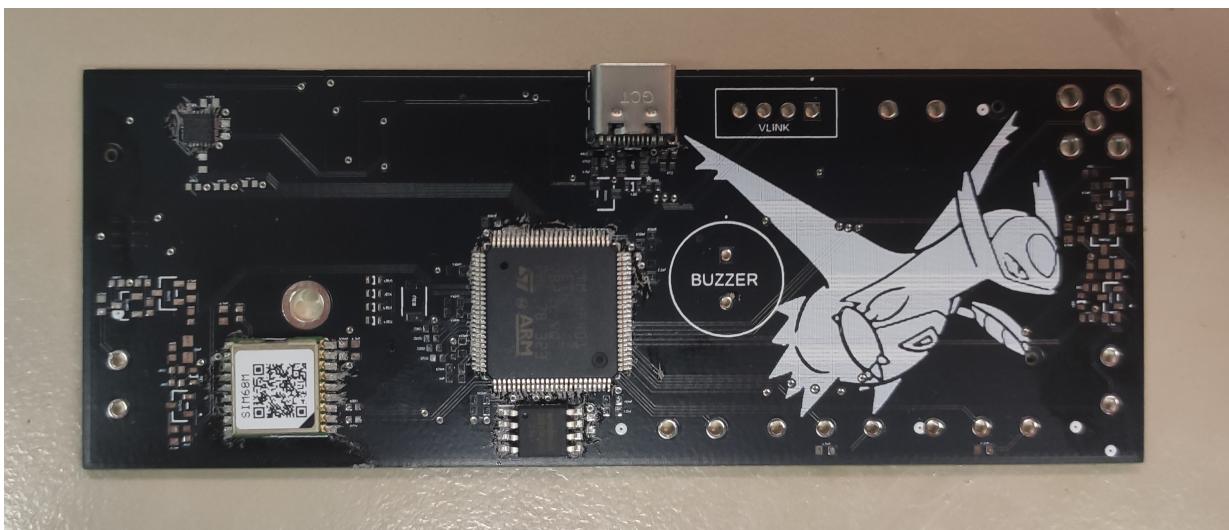
Per la fabricació, s'ha decidit no elaborar primer una placa fresada de coure per fer proves dels components, atès que hi ha restriccions temporals que impedeixen el prototipatge de la placa. A més, no es disposa dels recursos necessaris per gastar-los en plaques de prova. Per tant, s'ha encarregat la placa directament a JLCPCB [4], la qual triga només dues setmanes en arribar, que és el temps d'espera habitual per a l'arribada dels components.

Un cop transcorregudes les dues setmanes, ja es disposa de la placa, com es pot apreciar a la Figura 44, així com dels components. Això permet iniciar el procés de soldadura dels components a la placa i comprovar que tot funcioni correctament.

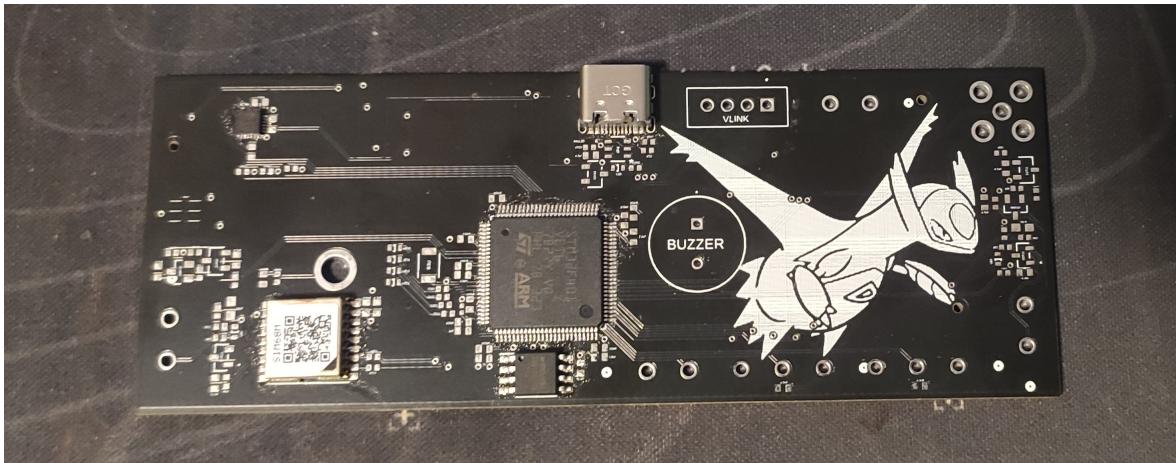


**Figura 44:** Plaques fabricades. Font: Elaboració pròpia.

Inicialment, s'ha decidit soldar la capa amb més components petits utilitzant un forn, mentre que la resta es soldaran manualment. Això implica una soldadura progressiva de les parts que s'han de provar, començant pel microcontrolador i el regulador de tensió. A la Figura 45, es pot observar la placa amb la pasta de soldadura aplicada, mentre que la Figura 46 mostra la placa després de sortir del forn amb els components soldats.

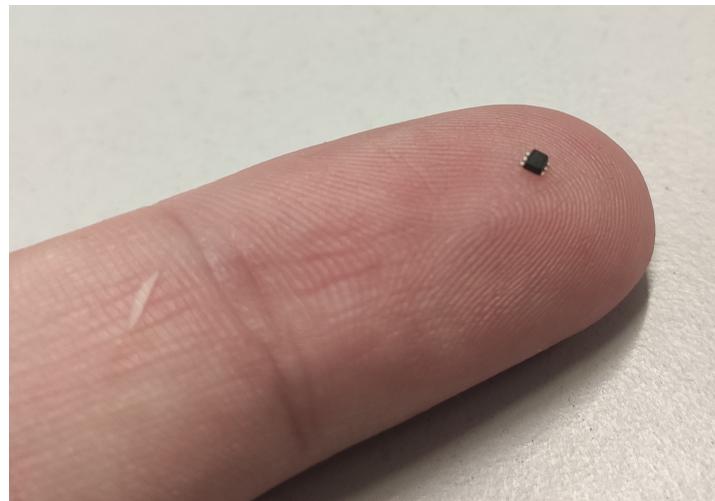


**Figura 45:** Placa amb pasta de soldadura aplicada. Font: Elaboració pròpia.



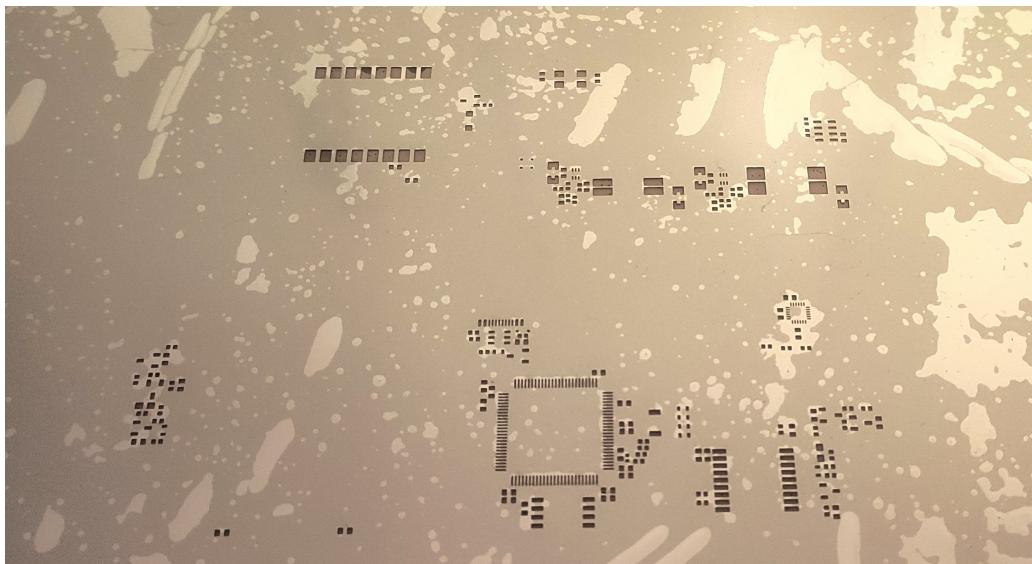
**Figura 46:** Placa després de sortir del forn. Font: Elaboració pròpria.

El següent pas consisteix en soldar manualment els components del regulador de tensió. Aquí és on s'ha trobat el primer problema de la placa. Ja que es troben a l'altra cara de la placa, els reguladors de tensió no es poden soldar amb el forn i, a més, com que són components molt petits, resulta difícil soldar-los manualment. Tot i així, s'ha intentat soldar-los manualment i, al comprovar el seu funcionament, s'ha produït fum al regulador de tensió, indicant que s'ha cremat. Això ha provocat la pèrdua d'un microcontrolador, un connector USB, un mòdul GNSS, un mòdul IMU, una memòria flash, un regulador de tensió i diversos components passius. En principi, això no hauria de ser un problema, ja que s'han adquirit tots els components per duplicat. No obstant això, si torna a fallar, el projecte haurà de sofrir un retard addicional de dues setmanes perquè arribin els nous components.



**Figura 47:** Referència de mida del regulador de tensió. Font: Elaboració pròpria.

Per sort, just quan s'estava soldant la placa, a l'associació es va aconseguir un patrocinador, que es dedica a la soldadura de plaques, i és Fadesa [29]. Fadesa necessita que se li proporcioni un stencil, que és una plantilla perforada que s'utilitza per aplicar soldadura en zones específiques d'una placa de circuit imprès, per a soldar plaques amb components de mida 0603 imperial. Com la nostra placa generalment té components de mida 0402, s'ha encarregat un stencil a JLCPLB, que ha arribat en 5 dies, i es pot veure a la Figura 50. Amb el stencil ja fet, s'ha enviat tot el material a Fadesa, i han realitzat la soldadura a la placa.



**Figura 48:** Stencil de la placa. Font: Elaboració pròpria.



**Figura 49:** Placa soldada vista per abaix. Font: Elaboració pròpria.



**Figura 50:** Placa soldada vista per adalt. Font: Elaboració pròpria.

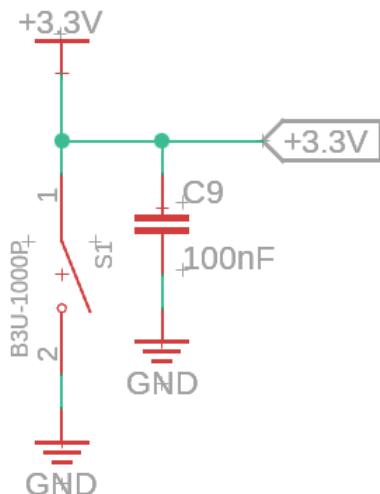
## 11. Proves de components

Per provar els components, s'ha decidit realitzar jocs de proves similars a la futura implementació prevista, utilitzant llibreries ja existents sempre que sigui possible i adaptant-les al microcontrolador utilitzat.

Per garantir el correcte funcionament dels components, és essencial començar comprovant que el microcontrolador es pot programar i depurar correctament. Si aquesta operació no es pot dur a terme, a més de resultar poc útil, la placa no podria executar les funcions més bàsiques, cosa que complicaria significativament la verificació del funcionament dels components.

### 11.1. Proves del microcontrolador

En primer lloc, cal verificar que el microcontrolador rebi correctament l'alimentació. Utilitzant un multímetre, s'ha comprovat que el corrent arriba de manera adequada. A continuació, s'ha d'assegurar que el microcontrolador es programa correctament. No obstant això, durant aquest procés s'ha detectat un problema: en intentar programar-lo amb un ST-Link, es produeix un error de reset. Després de diverses proves i d'investigar la causa, s'ha identificat un error en el disseny del circuit de reset. Tal com es mostra a la Figura 56, el circuit de reset es manté normalment en un estat alt, però, segons el full de dades, hauria d'estar flotant i connectar-se a terra quan calgui.



**Figura 51:** Circuit de reset erroni. Font: Elaboració pròpria.

Per solucionar aquest problema, s'ha decidit disconnectar el pin de reset. Tot i que aquesta solució no és ideal, ha permès superar l'obstacle i programar el microcontrolador.

Un cop programat, s'ha comprovat que la comunicació amb l'USB funciona correctament. S'ha utilitzat la llibreria proporcionada pel mateix IDE i s'ha confirmat el seu correcte funcionament.

### 11.2. Proves de la memòria flash

Per a la memòria flash, s'ha trobat una llibreria<sup>[30]</sup> creada per Lars Boegild Thomsen, un usuari de GitHub, dissenyada per a la sèrie de microcontroladors STM32. Aquesta llibreria està ben documentada i explicada, cosa que la fa ideal per al projecte. La llibreria està sota la llicència MIT, que permet el seu ús lliure.

Com que es volen dissenyar els tests de manera que serveixin per al programa final, s'ha dissenyat el sistema de memòria que s'utilitzarà. Aquest constarà d'una pàgina de control a la pàgina 0 i d'una

pàgina per a cada tipus de dada, intercalades en grups de tres, tal com es mostra a la Taula 16.

Pàgina	Dades
0	Control
1	Barometre
2	IMU
3	GPS
4	Barometre
5	IMU
6	GPS
7	Barometre
8	IMU

**Taula 16:** Organització de pagines. Font: Elaboració pròpia

La pàgina de control estarà organitzada tal com es detalla a la Taula 17.

Posició	0	4	8	12	16	20	24	28
0	Última pag bar utilitzada	Última pag IMU utilitzada	Última pag GPS utilitzada	Nombre de vols guardats	~	~	~	~
1	Pag ini bar v1	Pag ini IMU v1	Pag ini GPS v1	Pag fin bar v1	Pag fin IMU v1	Pag fin GPS v1	Apogeu v1	Main v1
2	Pag ini bar v2	Pag ini IMU v2	Pag ini GPS v2	Pag fin bar v2	Pag fin IMU v2	Pag fin GPS v2	Apogeu v2	Main v2
3	Pag ini bar v3	Pag ini IMU v3	Pag ini GPS v3	Pag fin bar v3	Pag fin IMU v3	Pag fin GPS v3	Apogeu v3	Main v3
4	Pag ini bar v4	Pag ini IMU v4	Pag ini GPS v4	Pag fin bar v4	Pag fin IMU v4	Pag fin GPS v4	Apogeu v4	Main v4
5	Pag ini bar v5	Pag ini IMU v5	Pag ini GPS v5	Pag fin bar v5	Pag fin IMU v5	Pag fin GPS v5	Apogeu v5	Main v5
6	Pag ini bar v6	Pag ini IMU v6	Pag ini GPS v6	Pag fin bar v6	Pag fin IMU v6	Pag fin GPS v6	Apogeu v6	Main v6
7	Pag ini bar v7	Pag ini IMU v7	Pag ini GPS v7	Pag fin bar v7	Pag fin IMU v7	Pag fin GPS v7	Apogeu v7	Main v7

**Taula 17:** Organització de la memoria de la pàgina 0. Font: Elaboració pròpia

Les pàgines de dades emmagatzemem el màxim nombre possible de paquets de dades. En el cas del baròmetre, són 32 paquets de 2 dades: el temps i l'alçada. Per a la IMU, són 8 paquets de 7 dades: el temps, l'acceleració i la inclinació de cada eix. Finalment, per al GPS, són 21 paquets de 3 dades: el temps, la longitud i la latitud.

Aquest sistema s'ha implementat tal com es mostra en el següent fragment de codi, que realitza els passos inicials de consulta de la pàgina 0 i emmagatzema diverses pàgines de dades del baròmetre:

```

1  uint8_t bufferBar[256];
2      uint8_t bufferlec[256];
3      //Llegeix la pagina 0
4      w25qxx_read(&w25qxx, 0, bufferlec, 256);
5      int vols;
6      //consulta la quantitat de vols, a la posicio 12 de la pagina, i li suma un.
7      memcpy(&vols, &bufferlec[12], sizeof(int));
8      vols += 1;
9      memcpy(&bufferlec[12], &vols, sizeof(int));
10
11     //copia l'adreca de la ultima pagina utilitzada a l'ultimo vol pel barometre, de
12     //la posicio 0 de la pagina 0, a la posicio 0 del nou vol.
13     memcpy(&bufferlec[vols * 8 * 4], &bufferlec[0], sizeof(int));
14     //copia l'adreca de la ultima pagina utilitzada a l'ultimo vol per la IMU, de la
15     //posicio 4 de la pagina 0, a la posicio 4 del nou vol.
16     memcpy(&bufferlec[vols * 8 * 4 + 4], &bufferlec[4], sizeof(int));
17     //copia l'adreca de la ultima pagina utilitzada a l'ultimo vol pel GPS, de la
18     //posicio 8 de la pagina 0, a la posicio 8 del nou vol.
19     memcpy(&bufferlec[vols * 8 * 4 + 8], &bufferlec[8], sizeof(int));

```

```

18 //copia un apogeu de 1000m a la posicio 24 del nou vol.
19 float flot = 1000.0;
20 memcpy(&bufferlec[vols * 8 * 4 + 24], &flot, sizeof(int));
21 //copia un main de 150m a la posicio 28 del nou vol.
22 flot = 150;
23 memcpy(&bufferlec[vols * 8 * 4 + 28], &flot, sizeof(int));
24
25 uint8_t *data = (uint8_t*) "Escrivint dades barometre\n";
26 CDC_Transmit_FS(data, strlen((char*) data));
27 int pag = 0;
28 //copia l'adreca de la ultima pagina utilitzada a l'ultim vol pel barometre, de
29 //la posicio 0 de la pagina 0,
30 //per començar a escriure dades del barometre
31 memcpy(&pag, &bufferlec[0], sizeof(int));
32 //escriu 4 pagines amb noms ascendents i temps ascendent.
33 for (int j = 0; j < 4; j += 1) {
34     memset(bufferBar, 0, 256);
35     for (int i = 0; i < 256; i += 8) {
36         float value_f = i * 4.0;
37         uint32_t value_u = (i + 4) * 4;
38         memcpy(&bufferBar[i], &value_u, sizeof(uint32_t));
39         memcpy(&bufferBar[i + 4], &value_f, sizeof(float));
40     }
41     //Escriu la pagina a la memoria.
42     w25qxx_write(&w25qxx, pag * 256, bufferBar, sizeof(bufferBar));
43     HAL_Delay(100);
44     pag += 3;
45 }
46 //guarda la ultima pagina utilitzada a la posicio 0 de la pagina 0.
47 memcpy(&bufferlec[0], &pag, sizeof(int));
48 //guarda la ultima pagina utilitzada a la posicio 12 del vol.
49 memcpy(&bufferlec[vols * 8 * 4 + 12], &pag, sizeof(int));

```

Després, hi ha les funcions que llegeixen les dades emmagatzemades a la memòria. La primera és la funció `llegirBar`, que llegeix les dades d'un rang de pàgines donat i les transmet via USB:

```

1 void llegirBar(int pagIni, int pagFin) {
2     uint8_t bufferlec[256]; // Buffer per emmagatzemar les dades llegides
3     uint32_t temps = 0; // Variable per emmagatzemar el temps
4     float alt = 0; // Variable per emmagatzemar l'alcada
5
6     // Enviar titol de seccio a traves de USB
7     CDC_Transmit_FS((uint8_t*)"-----Dades Barometre-----\nTemps Altura(m
8     )\n", 55);
9
10    while (pagIni < pagFin) {
11        // Llegir una pagina de memoria
12        w25qxx_read(&w25qxx, pagIni * 256, bufferlec, sizeof(bufferlec));
13
14        for (int i = 0; i < 256; i += 8) {
15            // Copiar el temps i l'alcada de la memoria al buffer
16            memcpy(&temps, &bufferlec[i], sizeof(uint32_t));
17            memcpy(&alt, &bufferlec[i + 4], sizeof(float));
18
19            // Format i transmissio de dades a traves de USB
20            uint8_t data[40];
21            sprintf((char*)data, "%lu: %f\n", temps, alt);
22            CDC_Transmit_FS(data, strlen((char*)data));
23
24        pagIni += 3; // Incrementar la pagina en sequencies de 3
25        HAL_Delay(100); // Esperar per evitar errors en la transmissio
26    }

```

```

26
27 // Final de la transmissio de dades
28 CDC_Transmit_FS((uint8_t*)"-----\n", 51);
29 }

-----Dades Barometre-----
Temps          Altura (m)
16: 0.00
48: 32.00
80: 64.00
112: 96.00
144: 128.00
176: 160.00
208: 192.00
240: 224.00
272: 256.00
304: 288.00
336: 320.00
368: 352.00
400: 384.00
432: 416.00
464: 448.00

```

**Figura 52:** Prova de la funció de lectura del baròmetre. Font: Elaboració pròpia.

La funció `llegirIMU` funciona de manera similar, però llegeix les dades d'acceleració i inclinació:

```

1 void llegirIMU(int pagIni, int pagFin) {
2     uint8_t bufferlec[256];
3     uint32_t temps = 0;
4     float datax, datay, dataz, incx, incy, incz;
5
6     CDC_Transmit_FS((uint8_t*)"-----Dades IMU-----\nTemps      Acceleracio(g)
7                           Inclinacio(dps)\n", strlen((char*)data));
8
9     while (pagIni < pagFin) {
10         // Llegir una pagina de memoria
11         w25qxx_read(&w25qxx, pagIni * 256, bufferlec, sizeof(bufferlec));
12
13         for (int i = 0; i < 252; i += 28) {
14             // Copiar temps i dades d'acceleracio i inclinacio de la memoria
15             memcpy(&temps, &bufferlec[i], sizeof(uint32_t));
16             memcpy(&datax, &bufferlec[i + 4], sizeof(float));
17             memcpy(&datay, &bufferlec[i + 8], sizeof(float));
18             memcpy(&dataz, &bufferlec[i + 12], sizeof(float));
19             memcpy(&incx, &bufferlec[i + 16], sizeof(float));
20             memcpy(&incy, &bufferlec[i + 20], sizeof(float));
21             memcpy(&incz, &bufferlec[i + 24], sizeof(float));
22
23             // Format i transmissio de les dades
24             uint8_t data[256];
25             sprintf((char*)data, "%lu: %f %f %f || %f %f %f \n", temps, datax, datay,
26                     dataz, incx, incy, incz);
27             CDC_Transmit_FS(data, strlen((char*)data));
28
29         }
30         pagIni += 3; // Incrementar la pagina en sequencies de 3

```

```

28         HAL_Delay(100);
29     }
30
31 // Final de la transmissio
32 CDC_Transmit_FS((uint8_t*)"-----\n-----\n", 51);
33 }

-----Dades IMU-----
Temps          Acceleracio(g)                                Inclinacio(dps)
16: 0.00 0.00 0.00 || 0.00 0.00 0.00
128: 112.00 112.00 112.00 || 112.00 112.00 112.00
240: 224.00 224.00 224.00 || 224.00 224.00 224.00
352: 336.00 336.00 336.00 || 336.00 336.00 336.00
464: 448.00 448.00 448.00 || 448.00 448.00 448.00
576: 560.00 560.00 560.00 || 560.00 560.00 560.00
688: 672.00 672.00 672.00 || 672.00 672.00 672.00
800: 784.00 784.00 784.00 || 784.00 784.00 784.00
16: 0.00 0.00 0.00 || 0.00 0.00 0.00
128: 112.00 112.00 112.00 || 112.00 112.00 112.00
240: 224.00 224.00 224.00 || 224.00 224.00 224.00
352: 336.00 336.00 336.00 || 336.00 336.00 336.00
464: 448.00 448.00 448.00 || 448.00 448.00 448.00
576: 560.00 560.00 560.00 || 560.00 560.00 560.00
688: 672.00 672.00 672.00 || 672.00 672.00 672.00
800: 784.00 784.00 784.00 || 784.00 784.00 784.00
-----
```

**Figura 53:** Prova de la funció de lectura de la IMU. Font: Elaboració pròpia.

La funció `llegirGPS` també funciona de manera similar, però llegeix les dades de posició:

```

1 void llegirGPS(int pagIni, int pagFin) {
2     uint8_t bufferlec[256];
3     uint32_t temps = 0;
4     float lat = 0;
5     float lon;
6
7     CDC_Transmit_FS((uint8_t*)"-----Dades GNSS-----\nTemps           Latitud
8                           Longitud\n", 51);
9
10    while (pagIni < pagFin) {
11        // Llegir una pagina de memoria
12        w25qxx_read(&w25qxx, pagIni * 256, bufferlec, sizeof(bufferlec));
13
14        for (int i = 0; i < 252; i += 12) {
15            // Copiar temps, latitud i longitud
16            memcpy(&temps, &bufferlec[i], sizeof(uint32_t));
17            memcpy(&lat, &bufferlec[i + 4], sizeof(float));
18            memcpy(&lon, &bufferlec[i + 8], sizeof(float));
19
20            // Format i transmissio de dades
21            uint8_t data[256];
22            sprintf((char*)data, "%lu: %f %f \n", temps, lat, lon);
23            CDC_Transmit_FS(data, strlen((char*)data));
24        }
25        pagIni += 3;
26        HAL_Delay(100);
27    }
28 }
```

```

26     }
27
28 // Final de la transmissio
29 CDC_Transmit_FS((uint8_t*)"-----\n-----\n", 51);
30 }

-----Dades GNSS-----
Temps           Latitud(°)           Longitud(°)
16: 0.00 0.00
64: 48.00 48.00
112: 96.00 96.00
160: 144.00 144.00
208: 192.00 192.00
256: 240.00 240.00
304: 288.00 288.00
352: 336.00 336.00
400: 384.00 384.00
448: 432.00 432.00
496: 480.00 480.00
544: 528.00 528.00
592: 576.00 576.00
640: 624.00 624.00
688: 672.00 672.00
736: 720.00 720.00
784: 768.00 768.00
832: 816.00 816.00
880: 864.00 864.00
928: 912.00 912.00
976: 960.00 960.00

```

**Figura 54:** Prova de la funció de lectura del GPS. Font: Elaboració pròpia.

Finalment, hi ha la funció per esborrar les dades de la memòria. Un cop esborrada, s'ha de configurar perquè estigui preparada per emmagatzemar un nou vol:

```

1 void borrar_memoria() {
2     uint8_t bufferBar[256];
3     buffer[0] = (uint8_t) '*';
4     uint8_t *data =
5         (uint8_t*) "Borraras tota la memoria, estas segur? \n1. => si,\n"
6         "2. => no\n";
7     CDC_Transmit_FS(data, strlen((char*) data));
8     //Pregunta per seguretat
9     while (buffer[0] == (uint8_t) '*');
10    if (buffer[0] == '1') {
11        data = (uint8_t*) "Borrant memoria\n";
12        CDC_Transmit_FS(data, strlen((char*) data));
13        //esborra la memoria
14        w25qxx_chip_erase(&w25qxx);
15        data = (uint8_t*) "Memoria borrada\n";
16        CDC_Transmit_FS(data, strlen((char*) data));
17        //Estableix els valors inicials de la pagina 0
18        memset(bufferBar, 0, 255);
19        int value = 21;
20        memcpy(&bufferBar[0], &value, sizeof(int));
21        value = 22;
22        memcpy(&bufferBar[4], &value, sizeof(int));
23        value = 23;

```

```

23         memcpy(&bufferBar[8], &value, sizeof(int));
24         value = 0;
25         memcpy(&bufferBar[12], &value, sizeof(int));
26         //escriu la pagina 0
27         w25qxx_write(&w25qxx, 0, bufferBar, sizeof(bufferBar));
28         CDC_Transmit_FS(
29             (uint8_t*) "\n"
30             -----\n",
31             51);
32     }

```

### 11.3. Llibreria del baròmetre

La llibreria del baròmetre s'ha adaptat d'una llibreria[31] creada per UravuLabs, un usuari de GitHub. Estava dissenyada per a Arduino, però només ha calgut modificar el mètode de transmissió de dades, canviant de Wire a I2C per STM32, i adaptant els *delay* al seu equivalent en STM32. La llibreria està sota la llicència GNU General Public License v2.0, que permet el seu ús lliure.

```

1 MS5607 barometre;
2
3 MX_USB_DEVICE_Init();
4 while (HAL_GPIO_ReadPin(SW_KEY_GPIO_Port, SW_KEY_Pin) != 1);
5 barometre.set(hi2c3);
6 while (!barometre.begin()) {
7     // Si no es pot inicialitzar, envia un missatge d'error per UART
8     const char *error_msg = "Error inicialitzant el barometre MS5607.\n";
9     CDC_Transmit_FS( (uint8_t*)error_msg, strlen(error_msg));
10 }
11 /* USER CODE END 2 */
12
13 /* Infinite loop */
14 /* USER CODE BEGIN WHILE */
15 while (1)
16 {
17     // Llegeix els valors de temperatura i pressio
18     if (barometre.readDigitalValue()) {
19         float temperature = barometre.getTemperature();
20         float pressure = barometre.getPressure();
21         float altitude = barometre.getAltitude();
22
23         // Envia els valors per USB
24         char buffer[150];
25         snprintf(buffer, sizeof(buffer), "Temperatura: %.2f C, Pressio: %.2f
26             hPa, Altura: %.2f m\n", temperature, pressure, altitude);
27         CDC_Transmit_FS( (uint8_t*)buffer, strlen(buffer));
28     }
29     else { CDC_Transmit_FS((uint8_t*)"fallada en lectura\n", 20);}
30     // Espera un segon abans de tornar a llegir
31     HAL_Delay(1000);
32     /* USER CODE END WHILE */
33     /* USER CODE BEGIN 3 */
34 }

```

En aquest codi, es fa servir la funció `readDigitalValue()` per llegir els valors de les últimes mesures del baròmetre. A continuació, es calculen els valors de la temperatura, la pressió i l'alçada amb les seves funcions "get". Aquestes dades es transmeten per USB per verificar que siguin correctes.

```

temperatura: 22.90C, Pressio: 999.69 hPa, Altura: 116.36 m
Temperatura: 22.71C, Pressio: 999.45 hPa, Altura: 116.12 m
Temperatura: 22.84C, Pressio: 999.64 hPa, Altura: 116.31 m
Temperatura: 22.69C, Pressio: 999.44 hPa, Altura: 116.11 m
Temperatura: 22.67C, Pressio: 999.37 hPa, Altura: 116.04 m
Temperatura: 22.67C, Pressio: 999.50 hPa, Altura: 116.17 m
Temperatura: 23.09C, Pressio: 999.40 hPa, Altura: 116.07 m
Temperatura: 22.53C, Pressio: 999.87 hPa, Altura: 116.54 m
Temperatura: 23.08C, Pressio: 999.80 hPa, Altura: 116.47 m
Temperatura: 22.82C, Pressio: 999.60 hPa, Altura: 116.27 m
Temperatura: 22.61C, Pressio: 999.58 hPa, Altura: 116.25 m
Temperatura: 22.73C, Pressio: 999.48 hPa, Altura: 116.15 m
Temperatura: 22.74C, Pressio: 999.90 hPa, Altura: 116.57 m
Temperatura: 22.87C, Pressio: 999.83 hPa, Altura: 116.50 m
Temperatura: 23.00C, Pressio: 999.55 hPa, Altura: 116.22 m
Temperatura: 23.05C, Pressio: 999.65 hPa, Altura: 116.32 m
Temperatura: 22.94C, Pressio: 999.82 hPa, Altura: 116.49 m
Temperatura: 22.72C, Pressio: 999.82 hPa, Altura: 116.49 m

```

**Figura 55:** Prova del baròmetre. Font: Elaboració pròpia.

## 11.4. Llibreria IMU

Per a la IMU, s'ha utilitzat la llibreria[32] creada per mokhwasomssi, un usuari de GitHub. Com que està dissenyada per a STM32, no ha calgut adaptar-la, i s'ha dissenyat el codi mostrat a continuació:

```

1  /* USER CODE BEGIN 2 */
2  icm20948_i2c(hi2c1);
3  icm20948_init();
4  ak09916_init();
5  /* USER CODE END 2 */

6
7  /* Infinite loop */
8  /* USER CODE BEGIN WHILE */
9  while (1)
10 {
11     /* USER CODE END WHILE */

12
13     /* USER CODE BEGIN 3 */
14     icm20948_accel_read_g(&accel_data);
15     icm20948_gyro_read_dps(&gyro_data);
16     if (ak09916_mag_read_uT(&mag_data)) {
17         angle = calculateAngle(&accel_data);

18         char buffer[150];
19         sprintf(buffer, sizeof(buffer), "Accel: X=%.2f, Y=%.2f, Z=%.2f\n", accel_data.x,
20                 accel_data.y, accel_data.z);
21         HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), HAL_MAX_DELAY);

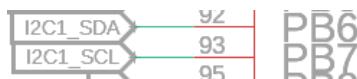
22         sprintf(buffer, sizeof(buffer), "Gyro: X=%.2f, Y=%.2f, Z=%.2f\n", gyro_data.x,
23                 gyro_data.y, gyro_data.z);
24         HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer),
25                           HAL_MAX_DELAY);

26         sprintf(buffer, sizeof(buffer), "Mag: X=%.2f, Y=%.2f, Z=%.2f\n",
27                 mag_data.x, mag_data.y, mag_data.z);
28         HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer),
29                           HAL_MAX_DELAY);

```

```
28         sprintf(buffer, sizeof(buffer), "Angle: %.2f\n\n", angle);
29         HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer),
30                             HAL_MAX_DELAY);
31     }
32
33     HAL_Delay(1000);
34 }
35 /* USER CODE END 3 */
36 }
```

En carregar i provar el codi, no ha funcionat. Després de buscar el problema, s'ha descobert que les connexions SDA i SCL estaven intercanviades. Com es pot veure a la Figura 56, s'indicava que el SCL era el pin PB7, però en realitat el PB7 hauria de ser el SDA.



**Figura 56:** Error de connexions de la IMU. Font: Elaboració pròpia.

## 11.5. Llibreria GNSS

Per al GPS, s'ha utilitzat la llibreria[33] creada per sztvka, un usuari de GitHub. La llibreria és completament compatible amb STM32, pel que no s'ha fet cap canvi. La llibreria simplement tradueix els missatges del format NMEA a un struct de dades comprensible, i està sota la llicència MIT, que permet el seu ús lliure. La comunicació amb el GNSS es realitza a través d'UART. Per començar, s'envien tres missatges de configuració: el primer comprova que està connectat, el segon configura el tipus de dades desitjades (en aquest cas, les més bàsiques de posició), i el tercer ajusta la freqüència de missatges a 10Hz. Un cop configurat, els missatges es reben a través de l'UART i es decodifiquen amb la llibreria, com es mostra al següent codi:

```

1  /* USER CODE BEGIN 2 */
2      uint8_t Rx_data[200];
3      HAL_UART_Transmit(&huart2, (uint8_t*)"$PMTK000*32\r\n\0", 15, 500);
4      HAL_UART_Receive(&huart2, Rx_data, 200, 100);
5      //HAL_UART_Transmit(&huart2, (uint8_t*)"$PMTK251,115200*1F<CR><LF>", 26, 500);
6      HAL_UART_Transmit(&huart2, (uint8_t*)"$PMTK314
7          ,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0*35\r\n\0", 55, 500);
8      HAL_UART_Receive(&huart2, Rx_data, 200, 100);
9      HAL_UART_Transmit(&huart2, (uint8_t*)"$PMTK220,100*2F\r\n\0", 21, 500);
10     HAL_UART_Receive(&huart2, Rx_data, 200, 100);
11     while (HAL_GPIO_ReadPin(SW_KEY_GPIO_Port, SW_KEY_Pin) != 1);
12
13 /* Infinite loop */
14 /* USER CODE BEGIN WHILE */
15 while (1)
16 {
17
18     GPS gnss;
19     uint8_t buffer_gnss[128];
20     HAL_UART_Receive(&huart2, buffer_gnss, 128, 100);
21     nmea_parse(&gnss, buffer_gnss);
22     uint8_t data[10];
23     sprintf((char*)data, "%f", gnss.latitude);
24     CDC_Transmit_FS(data, 10);
25     CDC_Transmit_FS((uint8_t *)"||", 2);
26     sprintf((char*)data, "%f", gnss.longitude);

```

```

27     CDC_Transmit_FS(data, 10);
28     CDC_Transmit_FS((uint8_t *)"||", 2);
29     sprintf((char*)data, "%f", gnss.altitude);
30     CDC_Transmit_FS(data, 10);
31     CDC_Transmit_FS((uint8_t *)"\n", 2);
32     /* USER CODE END WHILE */
33
34     /* USER CODE BEGIN 3 */
35 }
```

En provar aquest codi, s'ha vist que el codi en si funciona, però que les lectures no donen el resultat esperat. S'ha investigat i s'ha rebut el missatge: "GPRMC,000215.800,V,,,0.00,0.00,060180,,N\*4C", on la 'V' indica que el GNSS no detecta cap satèl·lit, cosa que impedeix proporcionar dades reals. Després de revisar possibles solucions, s'ha conclòs que seria millor utilitzar una antena activa en lloc d'una de passiva.

## 11.6. Llibreria LoRa

Per la comunicació LoRa, s'ha utilitzat la llibreria[34] creada per LoRaFi, un usuari de GitHub. La llibreria estava feta per a Arduino, i ha calgut adaptar les funcions d'inicialització i de transmissió de dades per STM32, i està sota la llicència GNU General Public License v3.0, que permet el seu ús. Un cop realitzats els canvis, s'ha provat la connexió, però s'ha descobert que per utilitzar un mòdul RF es necessita un emissor i un receptor, que no es disposava, fet que ha impedit realitzar les proves.

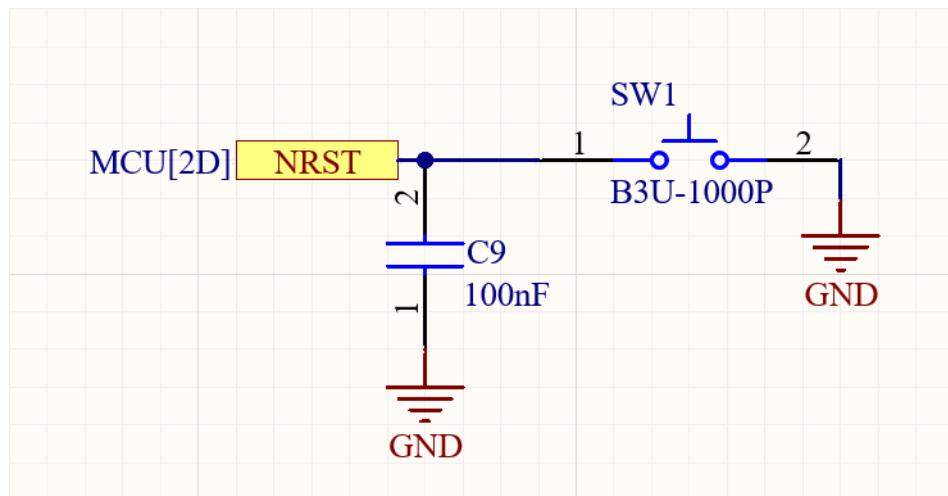
## 12. Segona versió

A causa dels problemes detectats en la primera versió de la placa, s'ha decidit crear una segona versió, corregint els errors i introduint alguns canvis menors en el disseny.

### 12.1. Disseny

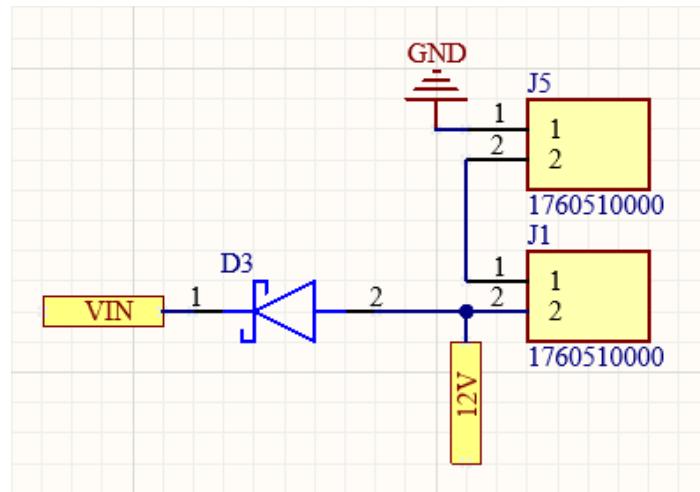
Per començar, s'ha redissenyat la placa des de zero amb Altium, ja que s'han rebut llicències per a aquest programa, i s'ha considerat més adequat per al desenvolupament de plaques entre els membres de l'associació. A més, s'ha optat per fer la placa de dues capes, ja que en la primera versió no es va trobar cap benefici significatiu en utilitzar quatre capes. Addicionalment, s'ha afegit un condensador d'1 F per prevenir possibles caigudes de tensió durant l'encesa dels ignitors.

El primer canvi en l'esquema ha estat el circuit de reset del microcontrolador, que s'ha dissenyat seguint exactament el que es recomana en el full de dades.



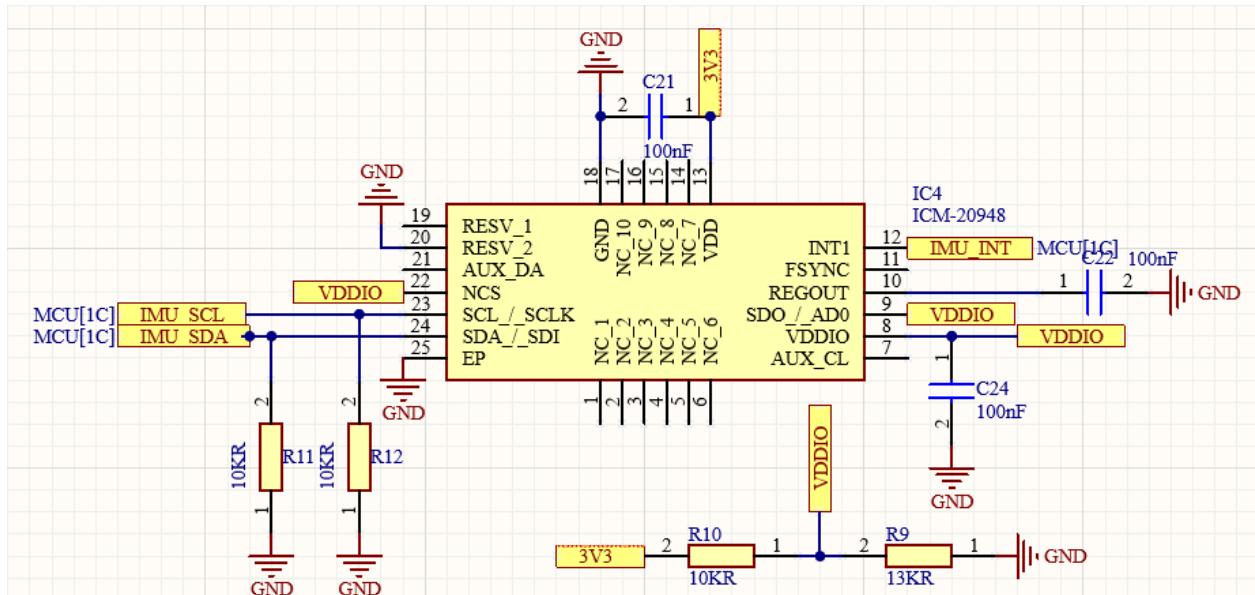
**Figura 57:** Esquema del circuit de reset de la segona versió. Font: Elaboració pròpia.

També s'ha modificat el funcionament de la *switch key*, que ara, en lloc de funcionar per software, talla el corrent de tota la placa. Aquest canvi respon al fet que, un cop muntat el coet, l'electrònica es torna inaccessible, i pot passar molta estona inactiva abans del llançament. Per això, és millor que no consumeixi energia fins que sigui necessari. A la Figura 58, es mostra el connector de la bateria a dalt i el connector de la *switch key* a baix.

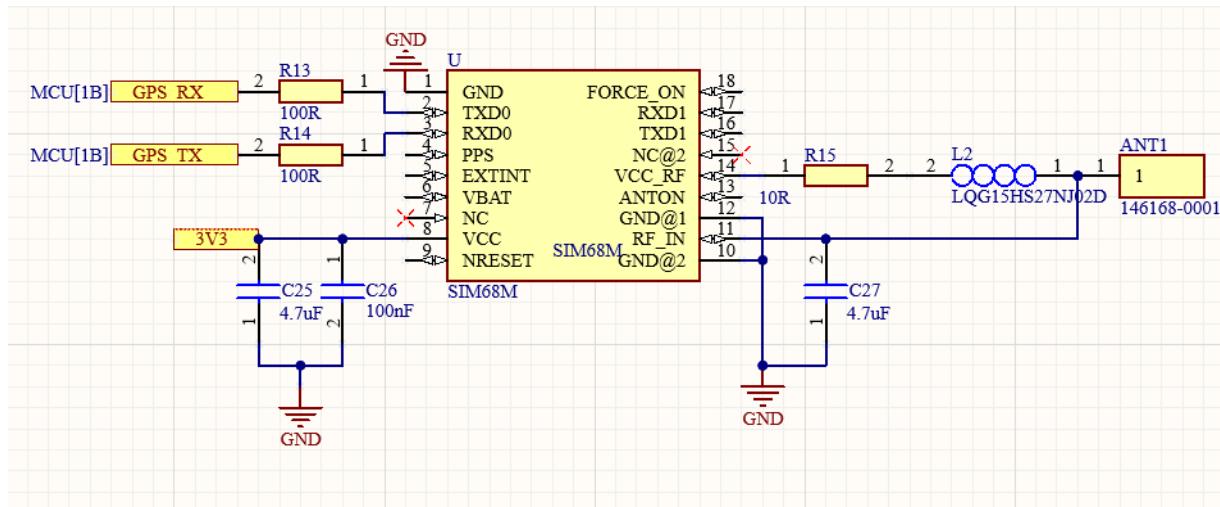


**Figura 58:** Esquema del circuit de la *switch key* de la segona versió. Font: Elaboració pròpria.

L'error en el disseny de la IMU també s'ha corregit, i pel GPS s'ha modificat l'esquema de l'antena, canviant-la per una antena activa en lloc d'una passiva.

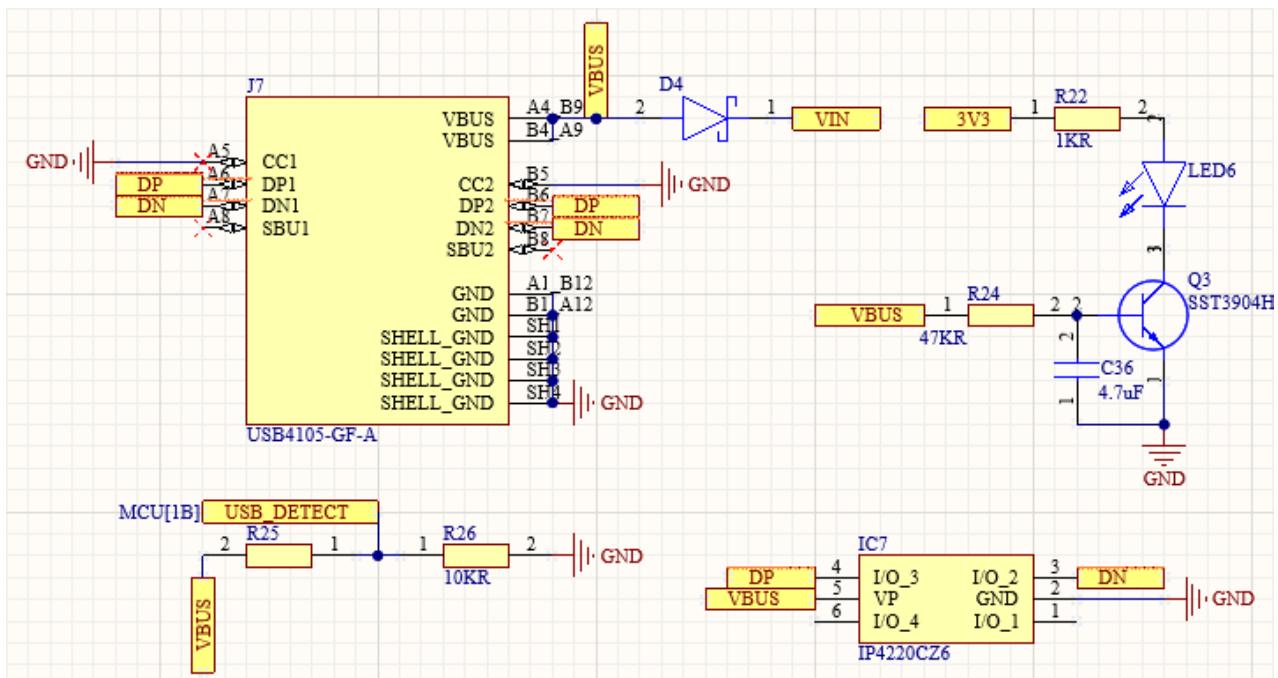


**Figura 59:** Esquema del circuit de la IMU de la segona versió. Font: Elaboració pròpria.



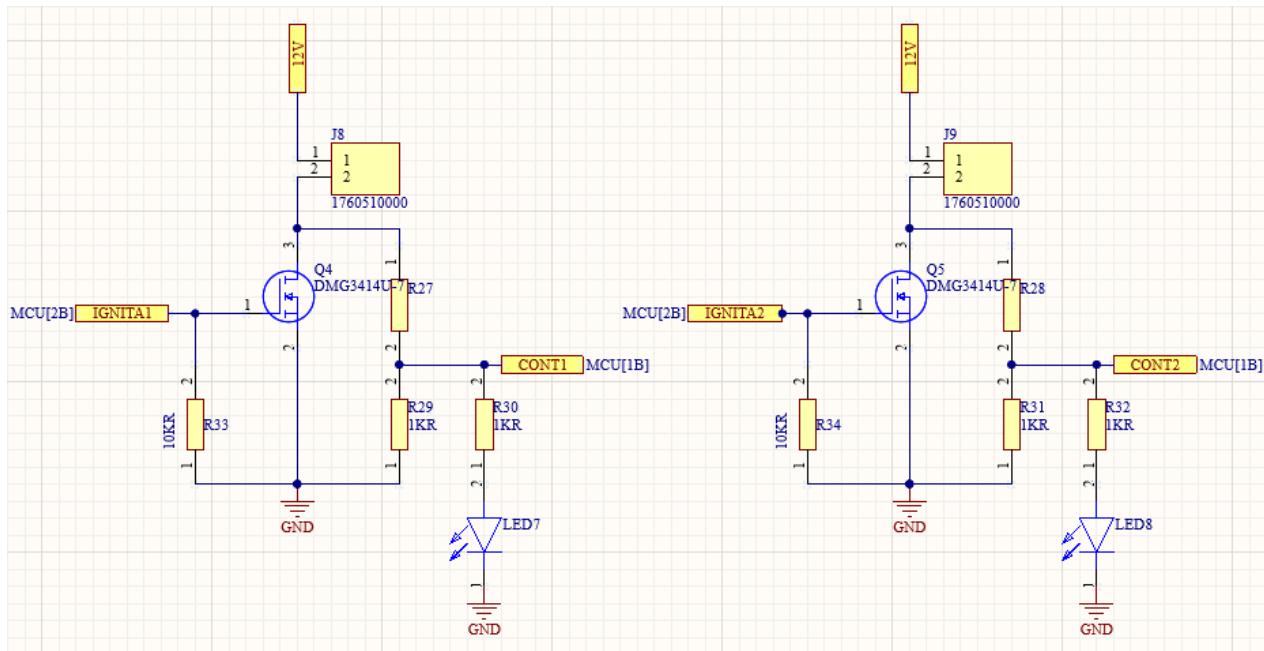
**Figura 60:** Esquema del circuit del GPS de la segona versió. Font: Elaboració pròpia.

Per a l'USB, s'ha afegit una connexió amb el microcontrolador per detectar quan està connectat, i així poder entrar en mode de comunicació automàticament.



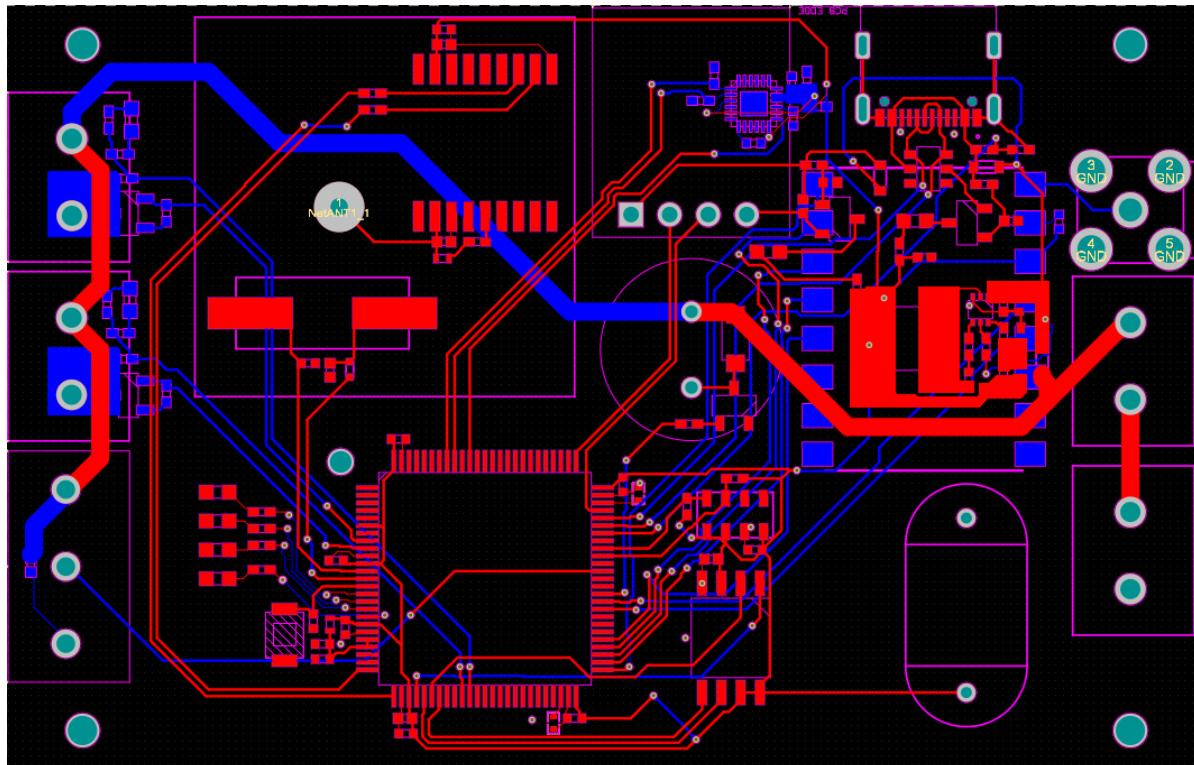
**Figura 61:** Esquema del circuit de l'USB de la segona versió. Font: Elaboració pròpria.

Finalment, s'han redissenyat els circuits d'ignició, fent un disseny més simple amb un únic activador. S'ha tornat a simular i, en aquest cas, s'ha realitzat una simulació amb una *protoboard*, comprovant que l'esquema funciona. Malauradament, amb l'emoció de veure el circuit funcionar, no es va pensar a enregistrar el funcionament.



**Figura 62:** Esquema del circuit dels ignitors de la segona versió. Font: Elaboració pròpria.

El disseny de la PCB és semblant a l'anterior, però s'ha intentat reduir la longitud de la placa col·locant els connectors en horitzontal en lloc de vertical.



**Figura 63:** Vista superior de la PCB de la segona versió. Font: Elaboració pròpria.

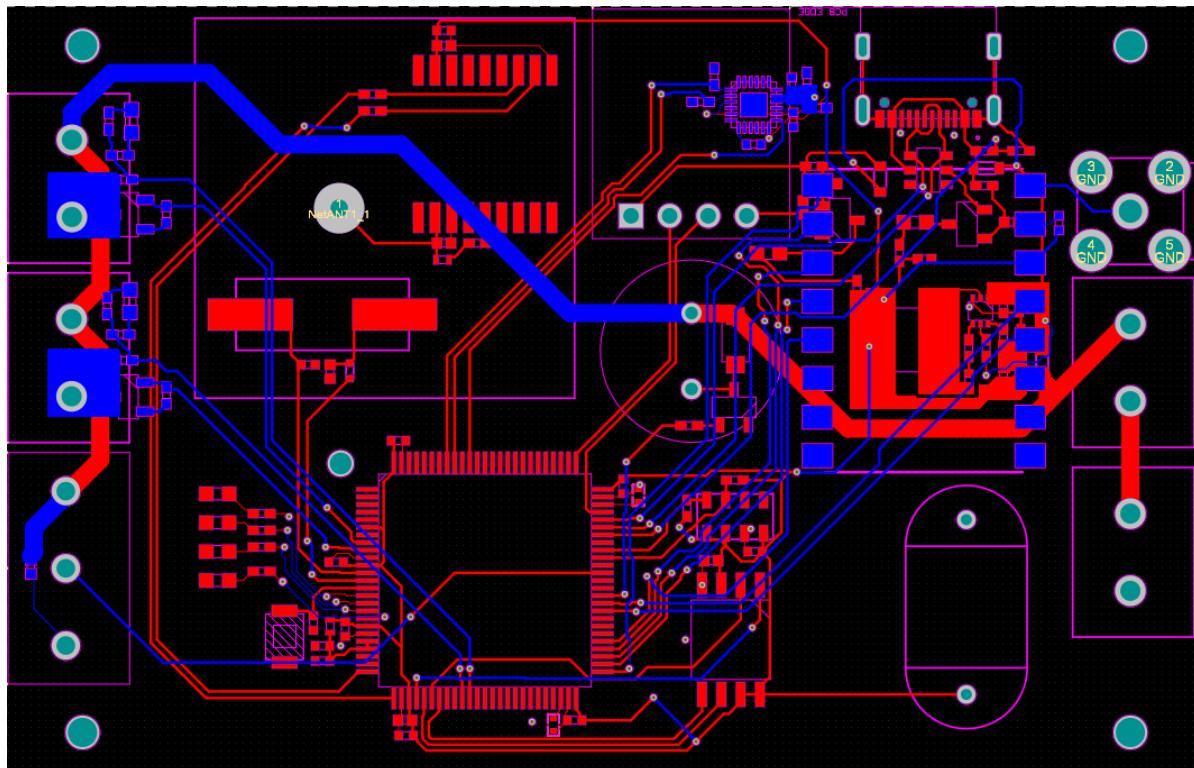


Figura 64: Vista inferior de la PCB de la segona versió. Font: Elaboració pròpria.

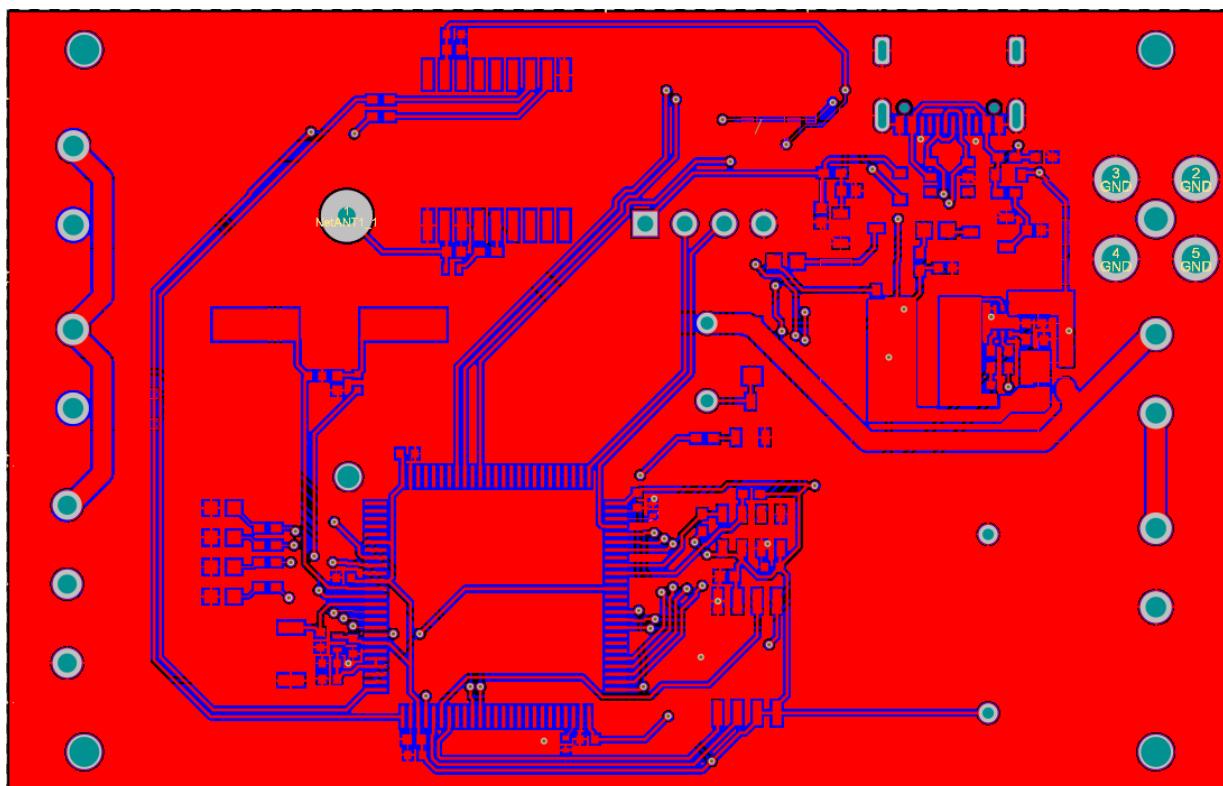


Figura 65: Capa superior de la PCB de la segona versió. Font: Elaboració pròpria.

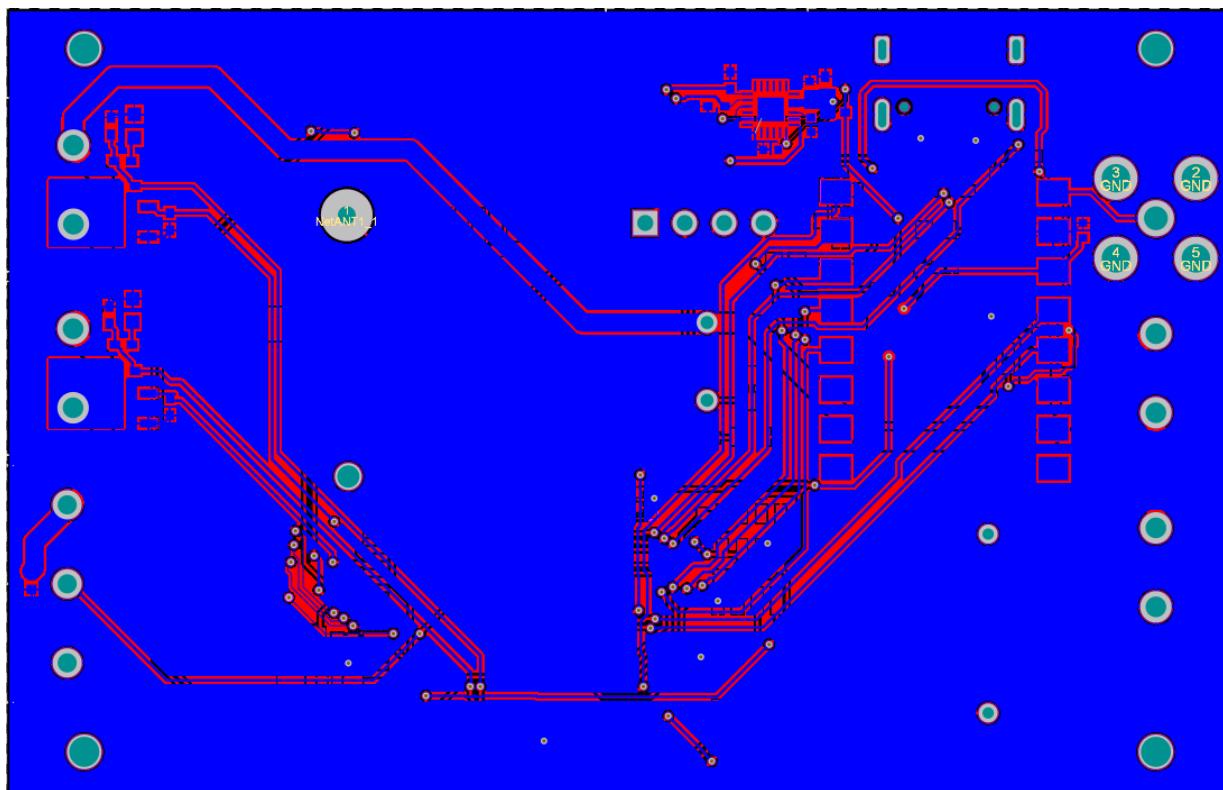


Figura 66: Capa inferior de la PCB de la segona versió. Font: Elaboració pròpria.

Altium també ofereix la possibilitat de fer una visualització en 3D de com quedarà la placa:

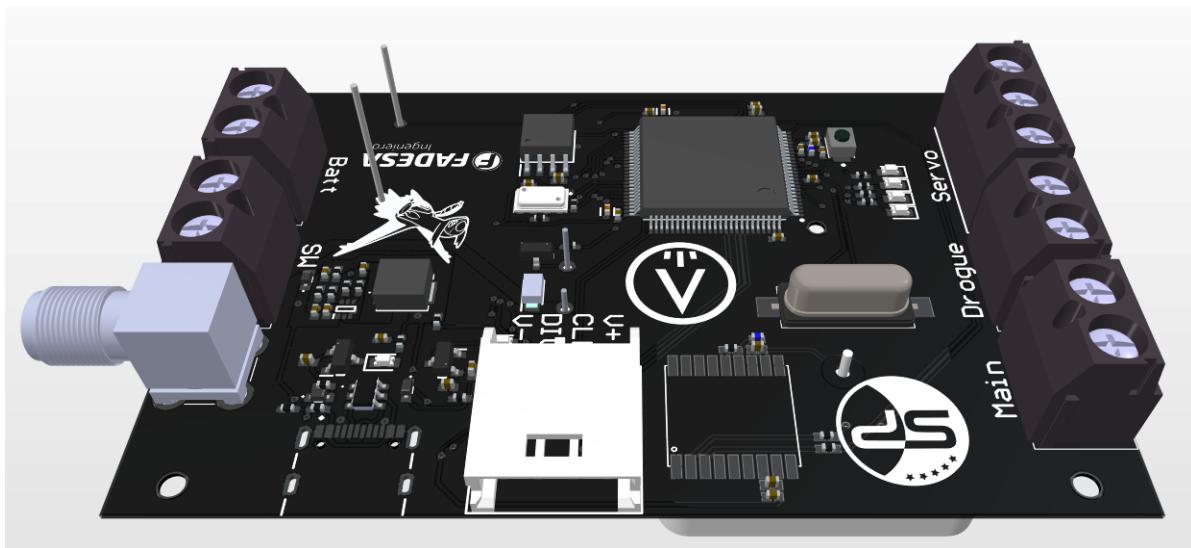


Figura 67: Visualització 3D superior de la segona versió. Font: Elaboració pròpria.

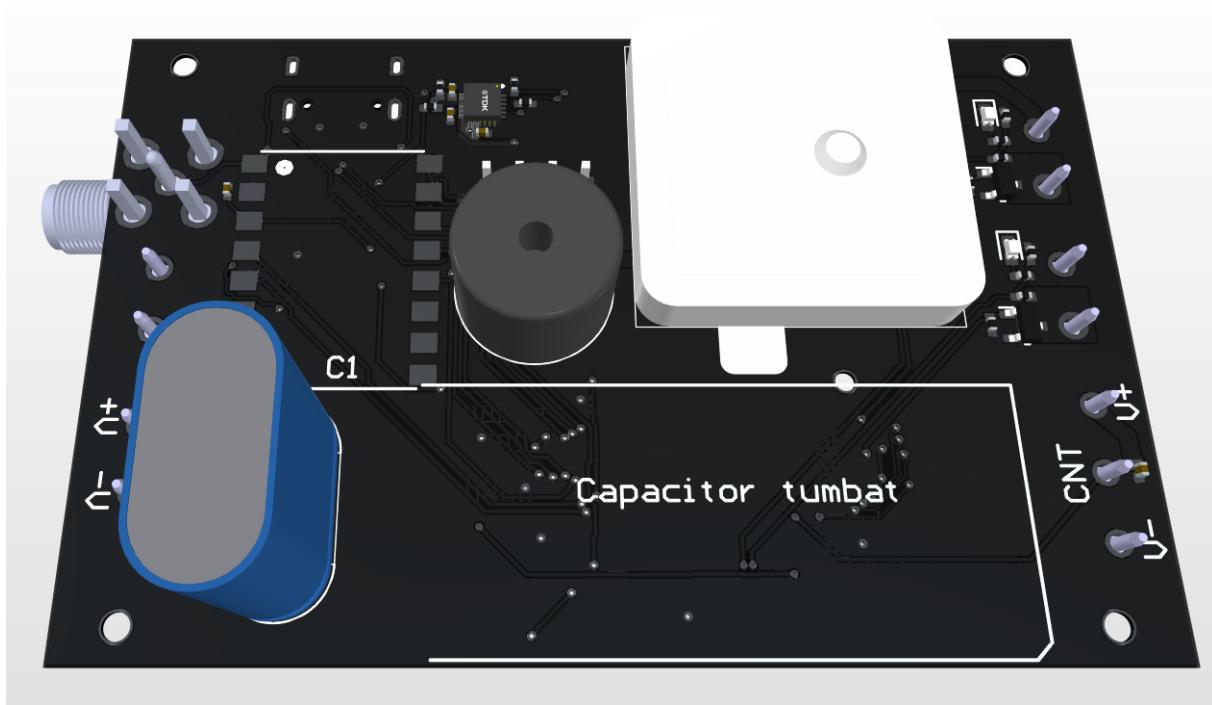


Figura 68: Visualització 3D inferior de la segona versió. Font: Elaboració pròpia.

## 12.2. Fabricació

Per fabricar la placa, s'ha decidit tornar a utilitzar els serveis de JLCPCB i Fadesa, obtenint els següents resultats:

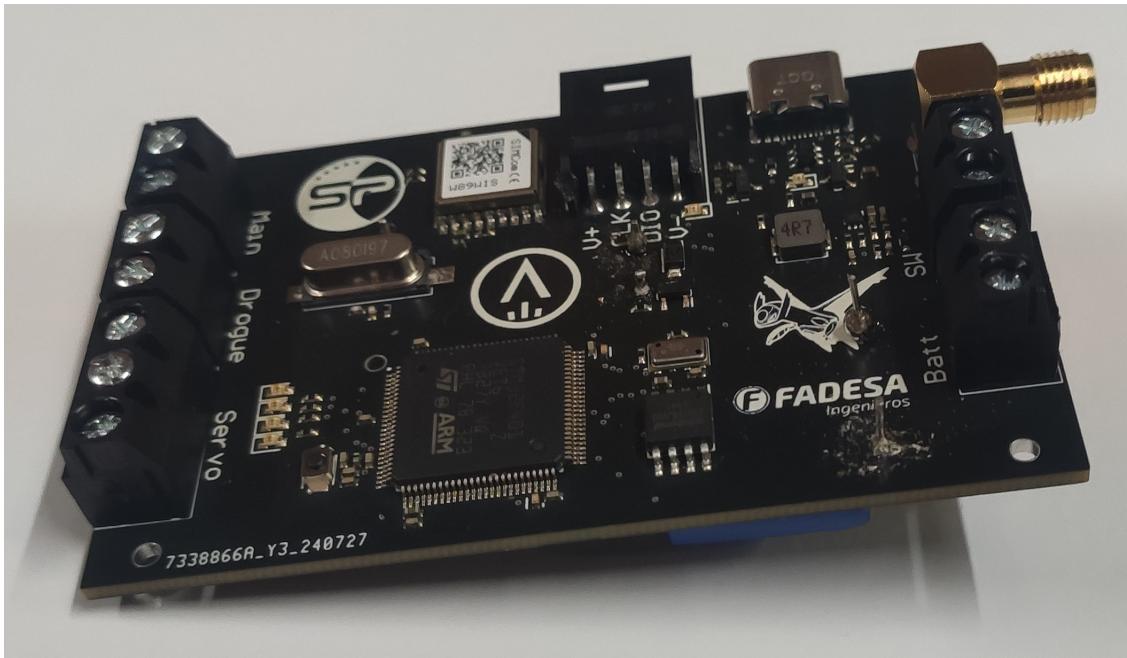
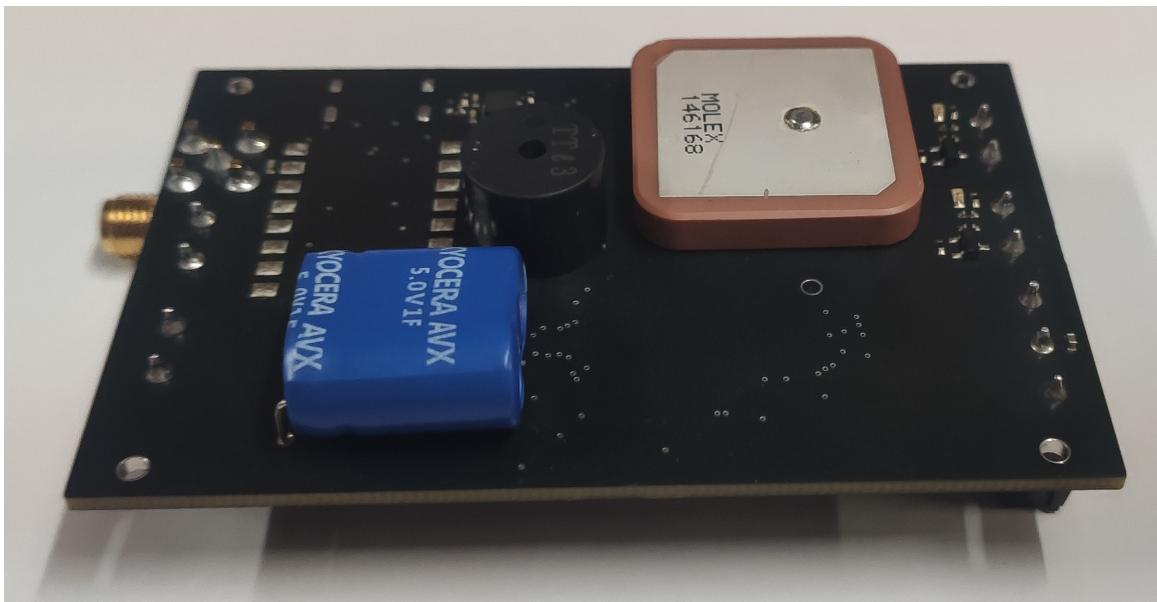


Figura 69: Cara superior de la segona versió. Font: Elaboració pròpia.

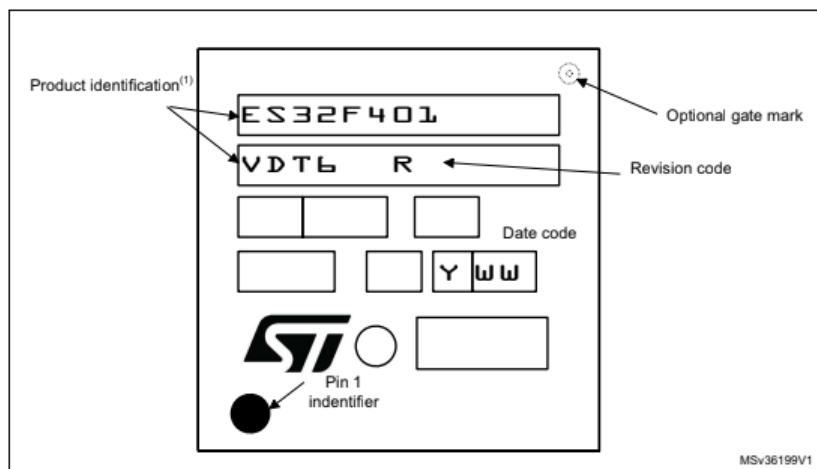


**Figura 70:** Cara inferior de la segona versió. Font: Elaboració pròpia.

### 12.3. Proves

Amb la placa ja soldada, s'han fet les proves seguint el mateix principi que en la versió anterior, començant per la programació del microcontrolador. En connectar la placa a la font d'alimentació, s'ha comprovat amb un multímetre que arribava el voltatge correcte a cada component. No obstant això, s'ha detectat que als components de 3,3V no els arribava res, motiu pel qual s'ha buscat algun punt de la placa on hi hagués un curtcircuit. Finalment, s'ha descobert que el microcontrolador estava girat 90°.

Com es pot observar a la Figura 69, el microcontrolador té una marca, en aquest cas la secundària, a la cantonada inferior dreta, quan hauria d'estar a la cantonada superior dreta. Aquest error, tot i que greu, és comprensible, ja que, una altra vegada, hi ha hagut complicacions logístiques per soldar la placa, i s'ha lliurat a Fadesa la mateixa setmana en què s'havia de fer un llançament. En aquesta situació, l'associació, en contra de la meva opinió, va decidir pressionar Fadesa per tenir la placa el més ràpidament possible, la qual cosa va provocar que passessin per alt que el microcontrolador, contràriament al que és habitual, té la marca a la part inferior esquerra en lloc de la superior dreta, com es pot veure a la Figura 71.



**Figura 71:** Marques del microcontrolador[28].

Que el microcontrolador estigui girat 90° significa que la placa és inutilitzable, ja que tots els components passen pel control del microcontrolador. Amb la manca de temps i de material de recanvi, s'ha intentat desoldar i girar el microcontrolador amb una pistola de calor, però malauradament alguns pads de la placa s'han cremat durant el procés, entre ells els del circuit de reset, l'oscil·lador i els activadors dels ignitors. Davant de tot això, s'ha considerat que la placa és inutilitzable i que caldrà tornar a fabricar aquesta versió. Aquesta decisió és especialment difícil, ja que no hi haurà temps de tenir-la llesta per a l'entrega d'aquest projecte.

## 13. Conclusions

Aquest projecte, tot i no estar totalment complet, ha permès assolir diversos objectius rellevants i ha proporcionat un aprenentatge profund tant a nivell tècnic com metodològic. A més, el treball fins al moment ha establert una base sòlida per a futures iteracions del projecte.

### Què es podria haver fet de manera diferent

Hi ha diversos aspectes que es podrien haver enfocat de manera diferent per optimitzar el desenvolupament de la placa. Per exemple, un dels canvis principals hauria estat començar utilitzant development boards, que, tot i ser més cares, haurien facilitat la identificació i resolució d'errors en les primeres fases del projecte. Això hauria estalviat temps i recursos dedicats a la creació de prototips.

Una altra millora possible seria implementar un disseny més modular, creant plaques que es poguessin separar en mòduls més petits segons les necessitats del projecte. Aquest enfocament hauria facilitat el procés de prova i hauria permès canviar components o afegir funcionalitats de manera més fàcil i ràpida.

També s'ha contemplat l'ús de components més grans en les primeres fases del projecte, ja que això podria haver ajudat a identificar problemes amb més facilitat, i hauria facilitat la soldadura i desoldadura de components, fent que solucionar problemes d'un component o part de la placa fos més fàcil.

### Encerts del projecte

Un dels aspectes més positius del projecte ha estat l'elecció d'utilitzar el microcontrolador STM32, que ha proporcionat un gran suport tant a nivell de programació, amb STM32 Cube, com en altres aspectes. Aquesta elecció ha estat fonamental per aconseguir un desenvolupament fluid i organitzat.

A més, el desenvolupament d'aquesta memòria deixa una documentació que no només ajuda a evitar repetir els mateixos errors en futures iteracions, sinó que també suposa una contribució valiosa per a l'associació. Fins ara, no s'havia establert un sistema de documentació adequat, i, encara que no és un document especialment didàctic, pot ajudar a entendre els raonaments i el procés a l'hora d'enfrontar-se a un projecte d'aquest tipus.

### Aprendentatges clau

Durant el projecte s'ha après a utilitzar eines especialitzades per a la creació de PCB, com Fusion 360, Altium Designer i STM32 Cube, que han permès dissenyar el sistema de manera estructurada i han ajudat a adquirir experiència amb aquest tipus d'eines.

També s'ha constatat que la selecció de components és un aspecte crític del projecte, ja que condiciona quines connexions s'utilitzaran, la forma de la placa, el rendiment o la facilitat d'utilitzar-los. En general, s'ha fet una bona selecció de components, però alguns, com s'ha comentat anteriorment, hauria estat millor triar-los més grans.

D'altra banda, s'ha après, per la via dura, que la logística juga un paper fonamental en el desenvolupament de projectes d'enginyeria. Els retards ocasionats pel temps d'enviament, la fabricació, la soldadura i les gestions burocràtiques han mostrat la necessitat de planificar amb antelació i establir alternatives per evitar la interrupció del flux de treball. A més, s'ha après que és essencial fer diverses iteracions del disseny per assegurar la qualitat del producte final.

## Conclusió final

En resum, tot i que el projecte no està completament acabat, ha proporcionat una base sòlida per a futurs desenvolupaments. L'aprenentatge tècnic i metodològic obtingut, juntament amb les millores identificades per a futures iteracions, asseguren que es poden evitar alguns dels errors inicials i que es poden introduir canvis que millorin la qualitat del producte final. Els encerts, com l'elecció del microcontrolador STM32 i la documentació exhaustiva, són elements que asseguraran l'èxit futur del projecte, tant en competicions com en l'aplicació pràctica del sistema desenvolupat.

## A. Taules de Components

### A.1. Primera versió

Número de fabricant	Tipus	Designador	Quantitat	Fabricant	Preu unitari	Preu total
146168-0001	Antenna	ANT1	1	Molex	0,94449	0,94449
0402X104K160CT	Capacitor	C1, C2, C3, C4, C5, C6, C7, C8, C9, C20, C26, C28, C34, C37, C41, C42, C43	17	Multicomp	0,00363	0,06175
CL05A475MO5NUNC	Capacitor	C10, C14, C17, C18, C19, C21, C22, C23, C33, C35, C45, C46, C47, C48	14	Samsung	0,231	3,234
KGM05AR51A105KH	Capacitor	C11, C30, C40	3	Kyocera AVX	0,093	0,279
GRM155D80J225ME95D	Capacitor	C12, C13	2	Murata	0,09082	0,18163
AC0402JRNPO9BN100	Capacitor	C29, C39	2	Yageo	0,09082	0,18164
04026D226MAT2A	Capacitor	C49, C50, C51, C52	4	Kyocera AVX	0,29126	2,91
IP4220CZ6	Integrated Circuit	CR1	1	Nexperia	0,38143	0,38143
S07J-GS08	Diode	CR2	1	Vishay	0,401	2,01
W25Q128JVSIQ	Integrated Circuit	IC1	1	Winbond	1,55	1,55
USB4105-GF-A	Connector	J1	1	Global Connector Technology	0,72653	0,72653
PCB.SMAFRA.HT	Connector	J2	1	Taoglas	3,1	3,1
280378-2	Connector	J3	1	TE Connectivity	1,2	1,2
1760510000	Connector	J4, J5, J6, J9	4	Weidmuller	0,53219	2,13
1760520000	Connector	J7, J8	2	Weidmuller	0,901	1,802
LQG15HS27NJ02D	Inductor	L1	1	Murata	0,09082	0,09082
ETQ-P4M4R7YPF	Inductor	L2	1	Panasonic	1,22	1,22
74477002	Inductor	L3	1	Wurth Electronics	3,12	3,12
150060RS75000	LED	LED1, LED2, LED3, LED4, LED5, LED6, LED7	7	Wurth Electronics	0,13622	0,95354
SD1209TT-A1	Loudspeaker or Buzzer	LS1	1	TDK	2,09	2,09
DMG3414U-7	MOSFET	MOSFET1, MOSFET2	2	Diodes	0,29969	0,59939
DMP3098L-7	MOSFET	Q1, Q2	2	Diodes Incorporated	0,44	0,44
ERJ-PA2D47R0X	Resistor	R1,R2	2	Panasonic	0,288	0,576
RC0402JR-071KL	Resistor	R11, R13, R16, R17, R19, R20, R22, R25, R26, R28, R47	11	Yageo	0,00999	0,10989
RT0402FRE0710KL	Resistor	R12, R14, R21, R23, R36, R37, R53, R54, R55	9	Yageo	0,00908	0,08172
RC0402FR-075KL	Resistor	R18, R27	2	Yageo	0,00817	0,08173
560112110231	Resistor	R3, R43, R58, R59, R60, R61	6	Wurth Elektronik	0,093	0,558
RC0402FR-0710RL	Resistor	R31, R48	2	Yageo	0,00383	0,00766
MCMR04X4022FTL	Resistor	R33, R50	2	Multicomp	0,0064	0,128
MCMR04X1302FTL	Resistor	R34	1	Multicomp	0,0079	0,079
ERJ-S02J104X	Resistor	R35, R52	2	Panasonic	0,233	0,466
ERA-2AEB473X	Resistor	R4, R32	2	Panasonic	0,121	0,121
RC0402FR-07100RL	Resistor	R44, R45	2	Yageo	0,00182	0,00363
RB521SM-40FHT2R	Schottky Diode	R46, R56, R57	3	Rohm	0,35418	1,06
SFR01MZPJ753	Resistor	R49	1	ROHM Semiconductor	0,14	0,14
SST3904HZGT116	Transistor BJT NPN	R5, R15, R24, R30	4	Rohm	0,15439	0,61756
ERA-2AEB3242X	Resistor	R51	1	Panasonic	0,158	0,158
B3U-1000P	Switch	S1	1	Omron	0,63572	0,63572
STM32F401VET6	Integrated Circuit	U1	1	STMicroelectronics	4,1	4,1
MP1476GTF-P	Integrated Circuit	U3, U7	2	Monolithic Power Systems	1,94	3,88
MS560702BA03-50	Integrated Circuit	U4	1	TE Connectivity	9,26	9,26
SIM68M	GPS	U6	1	Simcom	15,41	15,41
ICM-20948	Integrated Circuit	U8	1	TDK	7,66	7,66
RF-LORA-868-SO	Integrated Circuit	U9	1	RF Solutions	18,6	18,6

**Taula 18:** Taula de components de la primera versió. Font: Elaboració pròpria

## A.2. Segona versió

Número de fabricant	Tipus	Designador	Quantitat	Fabricant	Preu unitari	Preu total
146168-0001	Antena	ANT1	1	Molex	0,94449	0,94449
SCMR18C105PRBA0	Supercondensador	C1	1	Kyocera AVX	1,52	1,52
0402X104K160CT	Condensador	C2	1	Walsin Technologies	0,09082	0,09082
MC0402X104K160CT	Condensador	C3, C4, C5, C6, C7, C9, C14, C15, C19, C20, C21, C22, C24, C26, C28, C32	16	Multicomp	0,00363	0,05812
CL05A475MO5NUNC	Condensador	C8, C17, C23, C25, C27, C34, C35, C36	8	Samsung	0,231	2,31
GRM155D80J225ME95D	Condensador	C10, C11	2	Murata	0,09082	0,18163
GJM1555C1H7R0BB01D	Condensador	C12	1	Murata	0,0439	0,439
CC0402JRNP09BN180	Condensador	C13	1	Yageo	0,012	0,12
AC0402JRNP09BN100	Condensador	C29	1	Yageo	0,09082	0,09082
04026D226MAT2A	Condensador	C31, C33	2	Kyocera AVX	0,29126	2,91
RS07J-GS08	Díode	D1	1	Vishay	0,401	2,01
RB521SM-40FHT2R	Díode Schottky	D3, D4, D5	3	Rohm	0,35418	1,06
STM32F401VET6	Circuit integrat	IC1	1	STMicroelectronics	4,1	4,1
RF-LORA-868-SO	Circuit integrat	IC2	1	RF Solutions	18,6	18,6
MS560702BA03-50	Circuit integrat	IC3	1	TE Connectivity	9,26	9,26
ICM-20948	Circuit integrat	IC4	1	TDK	7,66	7,66
W25Q128JVSIQ	Circuit integrat	IC5	1	Winbond	1,55	1,55
MP1476GTF-P	Circuit integrat	IC6	1	Monolithic Power Systems	1,94	1,94
IP4220CZ6F	Circuit integrat	IC7	1	Nexperia	0,38143	0,38143
1760510000	Connector	J1, J5, J8, J9	4	Weidmuller	0,53219	2,13
1760520000	Connector	J2	1	Weidmuller	0,901	0,901
PCB.SMAFRA.HT	Connector	J4	1	Taoglas	3,1	3,1
280378-2	Connector	J6	1	TE Connectivity	1,2	1,2
USB4105-GF-A	Connector	J7	1	Global Connector Technology	0,72653	0,72653
LQG15HS27NJ02D	Inductor	L1	1	Murata	0,09082	0,09082
LQG15HS27NJ02D	Inductor	L2	1	Murata	0,08248	0,08248
ASPI-4020HI-4R7M-T	Inductor	L3	1	Abraccon	0,553	2,77
150060RS75000	LED	LED1, LED2, LED3, LED4, LED5, LED6, LED7, LEDs	8	Wurth Electronics	0,13622	1,09
SD1209TT-A1	Altaveu o brunzidor	LS1	1	TDK	2,09	2,09
SST3904HZGT116	Transistor BJT NPN	Q1	1	Rohm	0,15439	0,15439
SST3904HZGT116	Transistor BJT NPN	Q2, Q3	2	Rohm	0,14021	0,28042
DMG3414U-7	MOSFET (N-canal)	Q4, Q5	2	Diodes	0,29969	0,59939
ERA-2AEB80R6X	Resistència	R1	1	Panasonic	0,15439	0,15439
RC0402JR-071KL	Resistència	R2, R3, R4, R5, R6, R21, R22, R29, R30, R31, R32	11	Yageo	0,00999	0,10989
RT0402FRE0710KL	Resistència	R7, R8, R10, R11, R12, R26, R33, R34	8	Yageo	0,00908	0,07265
MCMR04X1302FTL	Resistència	R9, R20	2	Multicomp	0,0079	0,079
RC0402FR-07100RL	Resistència	R13, R14	2	Yageo	0,00182	0,00363
RC0402FR-0710RL	Resistència	R15	1	Yageo	0,00383	0,00383
RC0402FR-0710RL	Resistència	R16	1	Yageo	0,00422	0,00422
RC0402FR-0710RL	Resistència	R17	1	Yageo	0,00399	0,00399
RC0402FR-0710RL	Resistència	R18, R19	2	Yageo	0,00264	0,00529
PMS2317	Interruptor	SW1	1	E-Switch	0,4339	0,4339
BC860A,215	Transistor PNP	T1	1	Nexperia	0,21209	0,21209

**Taula 19:** Taula de components de la segona versió. Font: Elaboració pròpria

## B. Codis sencers

### B.1. Memòria flash

#### Codi de la llibreria

```

1  /**
2   ****
3   * @file          : w25qxx.h
4   * @brief         : Minimal W25Qxx Library Source
5   ****
6   * @attention
7   *
8   * Copyright (c) 2022, 2023 Lars Boegild Thomsen <lbthomsen@gmail.com>
9   * All rights reserved.
10  *
11  * This software is licensed under terms that can be found in the LICENSE file
12  * in the root directory of this software component.
13  * If no LICENSE file comes with this software, it is provided AS-IS.
14  *
15  * Notice! The library does _not_ bother to check that sectors have been erased
16  * before writing.
17  *
18  ****
19  */
20
21 #include "main.h"
22 #include "w25qxx.h"
23 #include <string.h>
24
25 #ifdef DEBUG
26 #include <stdio.h>
27 #endif
28
29 /*
30  * Internal functions
31  */
32
33 /**
34  * @brief Enables CS (driving it low) of the W25Qxx
35  *
36  * @param W25Qxx handle
37  * @retval None
38  */
39 static inline void cs_on(W25QXX_HandleTypeDefDef *w25qxx) {
40     HAL_GPIO_WritePin(w25qxx->cs_port, w25qxx->cs_pin, GPIO_PIN_RESET);
41 }
42
43 /**
44  * @brief Disables CS (driving it high) of the W25Qxx
45  *
46  * @param W25Qxx handle
47  * @retval None
48  */
49 static inline void cs_off(W25QXX_HandleTypeDefDef *w25qxx) {
50     HAL_GPIO_WritePin(w25qxx->cs_port, w25qxx->cs_pin, GPIO_PIN_SET);
51 }
52
53 /**
54  * @brief Transmit data to w25qxx - ignore returned data
55  *

```

```

56     * @param W25Qxx handle
57     * @param Pointer to buffer with data to transmit
58     * @param Length (in bytes) of data to be transmitted
59     * @retval None
60 */
61 W25QXX_result_t w25qxx_transmit(W25QXX_HandleTypeDefDef *w25qxx, uint8_t *buf, uint32_t len)
62 {
63     W25QXX_result_t ret = W25QXX_Err;
64     if (HAL_SPI_Transmit(w25qxx->spiHandle, buf, len, HAL_MAX_DELAY) == HAL_OK) {
65         ret = W25QXX_Ok;
66     }
67     return ret;
68 }
69 /*
70  * Receive data from w25qxx
71 */
72 W25QXX_result_t w25qxx_receive(W25QXX_HandleTypeDefDef *w25qxx, uint8_t *buf, uint32_t len)
73 {
74     W25QXX_result_t ret = W25QXX_Err;
75     if (HAL_SPI_Receive(w25qxx->spiHandle, buf, len, HAL_MAX_DELAY) == HAL_OK) {
76         ret = W25QXX_Ok;
77     }
78     return ret;
79 }
80 uint32_t w25qxx_read_id(W25QXX_HandleTypeDefDef *w25qxx) {
81     uint32_t ret = 0;
82     uint8_t buf[3];
83     cs_on(w25qxx);
84     buf[0] = W25QXX_GET_ID;
85     if (w25qxx_transmit(w25qxx, buf, 1) == W25QXX_Ok) {
86         if (w25qxx_receive(w25qxx, buf, 3) == W25QXX_Ok) {
87             ret = (uint32_t) ((buf[0] << 16) | (buf[1] << 8) | (buf[2]));
88         }
89     }
90     cs_off(w25qxx);
91     return ret;
92 }
93 uint8_t w25qxx_get_status(W25QXX_HandleTypeDefDef *w25qxx) {
94     uint8_t ret = 0;
95     uint8_t buf = W25QXX_READ_REGISTER_1;
96     cs_on(w25qxx);
97     if (w25qxx_transmit(w25qxx, &buf, 1) == W25QXX_Ok) {
98         if (w25qxx_receive(w25qxx, &buf, 1) == W25QXX_Ok) {
99             ret = buf;
100        }
101    }
102 }
103 cs_off(w25qxx);
104 return ret;
105 }
106
107 W25QXX_result_t w25qxx_write_enable(W25QXX_HandleTypeDefDef *w25qxx) {
108     W25_DBG("w25qxx_write_enable");
109     W25QXX_result_t ret = W25QXX_Err;
110     uint8_t buf[1];
111     cs_on(w25qxx);
112     buf[0] = W25QXX_WRITE_ENABLE;
113     if (w25qxx_transmit(w25qxx, buf, 1) == W25QXX_Ok) {
114         ret = W25QXX_Ok;

```

```

115     }
116     cs_off(w25qxx);
117     return ret;
118 }
119
120 W25QXX_result_t w25qxx_wait_for_ready(W25QXX_HandleTypeDefDef *w25qxx, uint32_t timeout) {
121     W25QXX_result_t ret = W25QXX_0k;
122     uint32_t begin = HAL_GetTick();
123     uint32_t now = HAL_GetTick();
124     while ((now - begin <= timeout) && (w25qxx_get_status(w25qxx) && 0x01 == 0x01)) {
125         now = HAL_GetTick();
126     }
127     if (now - begin == timeout)
128         ret = W25QXX_Timeout;
129     return ret;
130 }
131
132 #ifdef W25QXX_QSPI
133 W25QXX_result_t w25qxx_init(W25QXX_HandleTypeDefDef *w25qxx, QSPI_HandleTypeDefDef *qhspi) {
134 }
135
136 #else
137 W25QXX_result_t w25qxx_init(W25QXX_HandleTypeDefDef *w25qxx, SPI_HandleTypeDefDef *hspi,
138     GPIO_TypeDef *cs_port, uint16_t cs_pin) {
139     W25QXX_result_t result = W25QXX_0k;
140
141     W25_DBG("w25qxx_init");
142
143     w25qxx->spiHandle = hspi;
144     w25qxx->cs_port = cs_port;
145     w25qxx->cs_pin = cs_pin;
146
147     cs_off(w25qxx);
148
149     uint32_t id = w25qxx_read_id(w25qxx);
150     if (id) {
151         w25qxx->manufacturer_id = (uint8_t) (id >> 16);
152         w25qxx->device_id = (uint16_t) (id & 0xFFFF);
153
154         switch (w25qxx->manufacturer_id) {
155             case W25QXX_MANUFACTURER_GIGADEVICE:
156
157                 w25qxx->block_size = 0x10000;
158                 w25qxx->sector_size = 0x1000;
159                 w25qxx->sectors_in_block = 0x10;
160                 w25qxx->page_size = 0x100;
161                 w25qxx->pages_in_sector = 0x10;
162
163                 switch (w25qxx->device_id) {
164                     case 0x6017:
165                         w25qxx->block_count = 0x80;
166                         break;
167                     default:
168                         W25_DBG("Unknown Giga Device device");
169                         result = W25QXX_Err;
170                     }
171
172                     break;
173             case W25QXX_MANUFACTURER_WINBOND:
174

```

```

175         w25qxx->block_size = 0x10000;
176         w25qxx->sector_size = 0x1000;
177         w25qxx->sectors_in_block = 0x10;
178         w25qxx->page_size = 0x100;
179         w25qxx->pages_in_sector = 0x10;
180
181     switch (w25qxx->device_id) {
182     case 0x4018:
183         w25qxx->block_count = 0x100;
184         break;
185     default:
186         W25_DBG("Unknown Winbond device");
187         result = W25QXX_Err;
188     }
189
190     break;
191 default:
192     W25_DBG("Unknown manufacturer");
193     result = W25QXX_Err;
194 }
195 } else {
196     result = W25QXX_Err;
197 }
198
199 if (result == W25QXX_Err) {
200     // Zero the handle so it is clear initialization failed!
201     memset(w25qxx, 0, sizeof(W25QXX_HandleTypeDef));
202 }
203
204 return result;
205
206 }
207 #endif
208
209 W25QXX_result_t w25qxx_read(W25QXX_HandleTypeDef *w25qxx, uint32_t address, uint8_t *buf,
210                               uint32_t len) {
211
212     W25_DBG("w25qxx_read - address: 0x%08lx, length: 0x%04lx", address, len);
213
214     // Transmit buffer holding command and address
215     uint8_t tx[4] = {
216         W25QXX_READ_DATA, (uint8_t) (address >> 16), (uint8_t) (address >> 8), (uint8_t) (
217             address), };
218
219     // First wait for device to get ready
220     if (w25qxx_wait_for_ready(w25qxx, HAL_MAX_DELAY) != W25QXX_0k) {
221         return W25QXX_Err;
222     }
223
224     cs_on(w25qxx);
225     if (w25qxx_transmit(w25qxx, tx, 4) == W25QXX_0k) { // size will always be fixed
226         if (w25qxx_receive(w25qxx, buf, len) != W25QXX_0k) {
227             cs_off(w25qxx);
228             return W25QXX_Err;
229         }
230     }
231     cs_off(w25qxx);
232
233     return W25QXX_0k;
}

```

```

234 W25QXX_result_t w25qxx_write(W25QXX_HandleTypeDefDef *w25qxx, uint32_t address, uint8_t *buf
235   , uint32_t len) {
236 
237   W25_DBG("w25qxx_write - address 0x%08lx len 0x%04lx", address, len);
238 
239   // Let's determine the pages
240   uint32_t first_page = address / w25qxx->page_size;
241   uint32_t last_page = (address + len - 1) / w25qxx->page_size;
242 
243   W25_DBG("w25qxx_write %lu pages from %lu to %lu", 1 + last_page - first_page,
244         first_page, last_page);
245 
246   uint32_t buffer_offset = 0;
247   uint32_t start_address = address;
248 
249   for (uint32_t page = first_page; page <= last_page; ++page) {
250 
251     uint32_t write_len = w25qxx->page_size - (start_address & (w25qxx->page_size - 1))
252       );
253     write_len = len > write_len ? write_len : len;
254 
255     W25_DBG("w25qxx_write: handling page %lu start_address = 0x%08lx buffer_offset =
256           0x%08lx len = %04lx", page, start_address, buffer_offset, write_len);
257 
258     // First wait for device to get ready
259     if (w25qxx_wait_for_ready(w25qxx, HAL_MAX_DELAY) != W25QXX_Ok) {
260       return W25QXX_Err;
261     }
262 
263     if (w25qxx_write_enable(w25qxx) == W25QXX_Ok) {
264 
265       uint8_t tx[4] = {
266         W25QXX_PAGE_PROGRAM, (uint8_t) (start_address >> 16), (uint8_t) (
267           start_address >> 8), (uint8_t) (start_address), };
268 
269       cs_on(w25qxx);
270       if (w25qxx_transmit(w25qxx, tx, 4) == W25QXX_Ok) { // size will always be
271         fixed
272         // Now write the buffer
273         if (w25qxx_transmit(w25qxx, buf + buffer_offset, write_len) != W25QXX_Ok)
274           {
275             cs_off(w25qxx);
276             return W25QXX_Err;
277           }
278         }
279       }
280     }
281 
282     W25QXX_result_t w25qxx_erase(W25QXX_HandleTypeDefDef *w25qxx, uint32_t address, uint32_t len
283   ) {
284 
285     W25_DBG("w25qxx_erase, address = 0x%08lx len = 0x%04lx", address, len);
286 
287     W25QXX_result_t ret = W25QXX_Ok;

```

```

287 // Let's determine the sector start
288 uint32_t first_sector = address / w25qxx->sector_size;
289 uint32_t last_sector = (address + len - 1) / w25qxx->sector_size;
290
291 W25_DBG("w25qxx_erase: first sector: 0x%04lx", first_sector);W25_DBG("w25qxx_erase:
292           last sector : 0x%04lx", last_sector);
293
294 for (uint32_t sector = first_sector; sector <= last_sector; ++sector) {
295
296     W25_DBG("Erasing sector %lu, starting at: 0x%08lx", sector, sector * w25qxx->
297             sector_size);
298
299     // First we have to ensure the device is not busy
300     if (w25qxx_wait_for_ready(w25qxx, HAL_MAX_DELAY) == W25QXX_0k) {
301         if (w25qxx_write_enable(w25qxx) == W25QXX_0k) {
302
303             uint32_t sector_start_address = sector * w25qxx->sector_size;
304
305             uint8_t tx[4] = {
306                 W25QXX_SECTOR_ERASE, (uint8_t) (sector_start_address >> 16), (uint8_t) (
307                     sector_start_address >> 8), (uint8_t) (sector_start_address), };
308
309             cs_on(w25qxx);
310             if (w25qxx_transmit(w25qxx, tx, 4) != W25QXX_0k) {
311                 ret = W25QXX_Err;
312             }
313             cs_off(w25qxx);
314         } else {
315             ret = W25QXX_Timeout;
316         }
317     }
318
319     return ret;
320 }
321
322 W25QXX_result_t w25qxx_chip_erase(W25QXX_HandleTypeDefDef *w25qxx) {
323     if (w25qxx_write_enable(w25qxx) == W25QXX_0k) {
324         uint8_t tx[1] = {
325             W25QXX_CHIP_ERASE };
326         cs_on(w25qxx);
327         if (w25qxx_transmit(w25qxx, tx, 1) != W25QXX_0k) {
328             return W25QXX_Err;
329         }
330         cs_off(w25qxx);
331         if (w25qxx_wait_for_ready(w25qxx, HAL_MAX_DELAY) != W25QXX_0k) {
332             return W25QXX_Err;
333         }
334     }
335     return W25QXX_0k;
336 }
337
338 /*
339  * vim: ts=4 et nowrap
340 */

```

### Codi de la prova

<sup>1</sup> /\* USER CODE BEGIN Header \*/

```

2 /**
3  ****
4 * @file          : main.c
5 * @brief         : Main program body
6 ****
7 * @attention
8 *
9 * Copyright (c) 2024 STMicroelectronics.
10 * All rights reserved.
11 *
12 * This software is licensed under terms that can be found in the LICENSE file
13 * in the root directory of this software component.
14 * If no LICENSE file comes with this software, it is provided AS-IS.
15 *
16 ****
17 */
18 /* USER CODE END Header */
19 /* Includes -----*/
20 #include "main.h"
21 #include "usb_device.h"
22
23 /* Private includes -----*/
24 /* USER CODE BEGIN Includes */
25 #include "w25qxx.h"
26 #include "usbd_cdc_if.h"
27 /* USER CODE END Includes */
28
29 /* Private typedef -----*/
30 /* USER CODE BEGIN PTD */
31
32 /* USER CODE END PTD */
33
34 /* Private define -----*/
35 /* USER CODE BEGIN PD */
36 W25QXX_HandleTypeDef w25qxx;
37 /* USER CODE END PD */
38
39 /* Private macro -----*/
40 /* USER CODE BEGIN PM */
41
42 /* USER CODE END PM */
43
44 /* Private variables -----*/
45 ADC_HandleTypeDef hadc1;
46
47 I2C_HandleTypeDef hi2c1;
48 I2C_HandleTypeDef hi2c3;
49
50 SPI_HandleTypeDef hspi1;
51 SPI_HandleTypeDef hspi2;
52
53 TIM_HandleTypeDef htim4;
54
55 UART_HandleTypeDef huart2;
56
57 /* USER CODE BEGIN PV */
58 //objecte de la memoria flash
59 W25QXX_HandleTypeDef w25qxx;
60 //buffer per el USB
61 uint8_t buffer[64];
62 /* USER CODE END PV */

```

```

63
64 /* Private function prototypes -----*/
65 void SystemClock_Config(void);
66 static void MX_GPIO_Init(void);
67 static void MX_ADC1_Init(void);
68 static void MX_I2C1_Init(void);
69 static void MX_I2C3_Init(void);
70 static void MX_SPI1_Init(void);
71 static void MX_SPI2_Init(void);
72 static void MX_USART2_UART_Init(void);
73 static void MX_TIM4_Init(void);
74 /* USER CODE BEGIN PFP */
75
76 /* USER CODE END PFP */
77
78 /* Private user code -----*/
79 /* USER CODE BEGIN 0 */
80
81 /*llegeix les pagines desde pagIni fins a pagFin llegint en la sequencia pagIni + n * 3,
82 * sent n el nombre d'iteracions necessaries per arribar a pagIni > pagFin.
83 * Per cada Pagina es llegiran, i representaran els valors de la següent manera:
84 * Primer El temps, en ms, en que es van recollir les dades, i despres l'alçada, en m,
85 * enregistrada.
86 */
87 void llegirBar(int pagIni, int pagFin) {
88     uint8_t bufferlec[256];
89     uint32_t temps = 0;
90     float alt = 0;
91     CDC_Transmit_FS(
92         (uint8_t*) "-----Dades Barometre-----\nTemps
93         Altura(m)\n",
94         55);
95     while (pagIni < pagFin) {
96         //llegeix la pagina pagIni i el guarda a bufferlec
97         w25qxx_read(&w25qxx, pagIni * 256, bufferlec, sizeof(bufferlec));
98         for (int i = 0; i < 256; i += 8) {
99             //copia la posicio de la memoria multiple de 8, que correspon al
100            temps.
101             memcpy(&temps, &bufferlec[i], sizeof(uint32_t));
102             //copia la posicio de la memoria multiple de 8 + 4, que correspon
103             a l'alçada.
104             memcpy(&alt, &bufferlec[i + 4], sizeof(float));
105             uint8_t data[40];
106             //Codifica el missatge en el format: Temps(ms): Alçada(m).
107             sprintf((char*) data, "%lu: %f\n", temps, alt);
108             CDC_Transmit_FS(data, strlen((char*) data));
109             HAL_Delay(100);
110         }
111         CDC_Transmit_FS(
112             (uint8_t*) "\n-----\n",
113             51);
114     }
115
116 /*llegeix les pagines desde pagIni fins a pagFin llegint en la seqüencia pagIni + n * 3,
117 * sent n el nombre d'iteracions necessaries per arribar a pagIni > pagFin.
118 * Per cada Pagina es llegiran, i representaran els valors de la següent manera:
119 * Primer El temps, en ms, en que es van recollir les dades, despres l'acceleracio de

```

```

        cada
120 * eix, (x, y i z) en g, i la inclinacio de cada eix, en dps, enregistrades.
121 */
122 void llegirIMU(int pagIni, int pagFin) {
123     uint8_t bufferlec[256];
124     uint32_t temps = 0;
125     float datax, datay, dataz, incx, incy, incz;
126     uint8_t *data =
127         (uint8_t*) "-----Dades IMU-----\nTemps
128             Acceleracio(g)
129             Inclinacio(dps)\n";
130     CDC_Transmit_FS(data, strlen((char*) data));
131     while (pagIni < pagFin) {
132         //llegeix la pagina pagIni i el guarda a bufferlec
133         w25qxx_read(&w25qxx, pagIni * 256, bufferlec, sizeof(bufferlec));
134         for (int i = 0; i < 252; i += 28) {
135             //copia la posicio de la memoria multiple de 28, que correspon al
136             //temps.
137             memcpy(&temps, &bufferlec[i], sizeof(uint32_t));
138             //copia la posicio de la memoria multiple de 28 + 4, que
139             //correspon a l'acceleracio del eix x.
140             memcpy(&datax, &bufferlec[i + 4], sizeof(float));
141             //copia la posicio de la memoria multiple de 28 + 8, que
142             //correspon a l'acceleracio del eix y.
143             memcpy(&datay, &bufferlec[i + 8], sizeof(float));
144             //copia la posicio de la memoria multiple de 28 + 12, que
145             //correspon a l'acceleracio del eix z.
146             memcpy(&dataz, &bufferlec[i + 12], sizeof(float));
147             //copia la posicio de la memoria multiple de 28 + 16, que
148             //correspon a la inclinacio del eix x.
149             memcpy(&incx, &bufferlec[i + 16], sizeof(float));
150             //copia la posicio de la memoria multiple de 28 + 20, que
151             //correspon a la inclinacio del eix y.
152             memcpy(&incy, &bufferlec[i + 20], sizeof(float));
153             //copia la posicio de la memoria multiple de 28 + 24, que
154             //correspon a la inclinacio del eix z.
155             memcpy(&incz, &bufferlec[i + 24], sizeof(float));
156             uint8_t data[256];
157             //Codifica el missatge en el format:
158             //Temps(ms): acceleracioX(g) acceleracioY(g) acceleracioZ(g) ||
159             //InclinacioX(dps) InclinacioY(dps) InclinacioZ(dps).
160             sprintf((char*) data, "%lu: %f %f %f || %f %f %f \n", temps,
161                     datax, datay, dataz, incx, incy, incz);
162             CDC_Transmit_FS(data, strlen((char*) data));
163         }
164         pagIni += 3;
165         //delay per evitar errors al rebre els missatges USB.
166         HAL_Delay(100);
167     }
168     CDC_Transmit_FS(
169         (uint8_t*) "\n-----\n",
170         51);
171 }
172 /*llegeix les pagines desde pagIni fins a pagFin llegint en la seqüencia pagIni + n * 3,
173 * sent n el nombre d'iteracions necessaries per arribar a pagIni > pagFin.
174 * Per cada Pagina es llegiran, i representaran els valors de la seguent manera:
175 * Primer El temps, en ms, en que es van recollir les dades, despues la latitud i
176 * longitud, en °, enregistrades.
177 */

```

```

169 void llegirGPS(int pagIni, int pagFin) {
170     uint8_t bufferlec[256];
171     uint32_t temps = 0;
172     float lat = 0;
173     float lon;
174     uint8_t *data =
175         (uint8_t*) "-----Dades GNSS-----\nTemps
176                             Latitud(°)           Longitud(°)\n";
177     CDC_Transmit_FS(data, strlen((char*) data));
178     while (pagIni < pagFin) {
179         //llegeix la pagina pagIni i el guarda a bufferlec
180         w25qxx_read(&w25qxx, pagIni * 256, bufferlec, sizeof(bufferlec));
181         for (int i = 0; i < 252; i += 12) {
182             //copia la posicio de la memoria multiple de 12, que correspon al
183             //temps.
184             memcpy(&temps, &bufferlec[i], sizeof(uint32_t));
185             //copia la posicio de la memoria multiple de 12 + 4, que
186             //correspon a la latitud.
187             memcpy(&lat, &bufferlec[i + 4], sizeof(float));
188             //copia la posicio de la memoria multiple de 12 + 8, que
189             //correspon a la longitud.
190             memcpy(&lon, &bufferlec[i + 8], sizeof(float));
191             uint8_t data[256];
192             //Codifica el missatge en el format: Temps(ms): latitud(°)
193             //longitud(°).
194             sprintf((char*) data, "%lu: %f %f \n", temps, lat, lon);
195             CDC_Transmit_FS(data, strlen((char*) data));
196         }
197         pagIni += 3;
198         //delay per evitar errors al rebre els missatges USB.
199         HAL_Delay(100);
200     }
201     CDC_Transmit_FS(
202         (uint8_t*) "\n-----\n",
203         51);
204 }
205 /*
206  * Dona opcions per llegir la memoria d'un vol, es pot llegir el barometre, el GPS i la
207  * IMU per separat o tot junt.
208 */
209 void llegir_vol(int vol, uint8_t *bufferlec) {
210     int pagIni, pagFin;
211     uint8_t data[256];
212     //mentre no s'esculli sortir es mante en la sel·leccio de lectures
213     while (1) {
214         buffer[0] = (uint8_t) '*';
215         sprintf((char*) data, "Vol %d\n\n", vol);
216         CDC_Transmit_FS(data, strlen((char*) data));
217         CDC_Transmit_FS(
218             (uint8_t*) "1: Dades Barometre \n2: Dades IMU \n3: Dades
219             GPS \n4: Totes les dades \n5: Tornar enrera \n
220             -----",
221             136);
222         //mentre no es rebi res pel USB es bloca
223         while (buffer[0] == (uint8_t) '*');
224         //comprova el missatge USB
225         switch (buffer[0]) {
226             //Lectura del barometre
227             case '1':

```

```

222     //Consulta a la pagina 0, la pagina inicial de les dades del
223     //barometre del vol.
224     memcpy(&pagIni, &bufferlec[vol * 4 * 8], sizeof(int));
225     //Consulta a la pagina 0, la pagina final de les dades del
226     //barometre del vol.
227     memcpy(&pagFin, &bufferlec[vol * 4 * 8 + 12], sizeof(int));
228     llegirBar(pagIni, pagFin);
229     break;
230
231     //Lectura de la IMU
232     case '2':
233         //Consulta a la pagina 0, la pagina inicial de les dades de la
234         //IMU del vol.
235         memcpy(&pagIni, &bufferlec[vol * 4 * 8 + 4], sizeof(int));
236         //Consulta a la pagina 0, la pagina final de les dades de la IMU
237         //del vol.
238         memcpy(&pagFin, &bufferlec[vol * 4 * 8 + 16], sizeof(int));
239         llegirIMU(pagIni, pagFin);
240         break;
241
242     //Lectura al GPS
243     case '3':
244         //Consulta a la pagina 0, la pagina inicial de les dades del GPS
245         //del vol.
246         memcpy(&pagIni, &bufferlec[vol * 4 * 8 + 8], sizeof(int));
247         //Consulta a la pagina 0, la pagina final de les dades del GPS
248         //del vol.
249         memcpy(&pagFin, &bufferlec[vol * 4 * 8 + 20], sizeof(int));
250         llegirGPS(pagIni, pagFin);
251         break;
252
253     //Lectura de tot el vol
254     case '4':
255         //Consulta a la pagina 0, la pagina inicial de les dades del
256         //barometre del vol.
257         memcpy(&pagIni, &bufferlec[vol * 4 * 8], sizeof(int));
258         //Consulta a la pagina 0, la pagina final de les dades del
259         //barometre del vol.
260         memcpy(&pagFin, &bufferlec[vol * 4 * 8 + 12], sizeof(int));
261         llegirBar(pagIni, pagFin);
262
263         //Consulta a la pagina 0, la pagina inicial de les dades de la
264         //IMU del vol.
265         memcpy(&pagIni, &bufferlec[vol * 4 * 8 + 4], sizeof(int));
266         //Consulta a la pagina 0, la pagina final de les dades de la IMU
267         //del vol.
268         memcpy(&pagFin, &bufferlec[vol * 4 * 8 + 16], sizeof(int));
269         llegirIMU(pagIni, pagFin);
270
271         float flot;
272         //Consulta a la pagina 0, l'apogeu del vol.
273         memcpy(&flot, &bufferlec[vol * 4 * 8 + 24], sizeof(float));
274         sprintf((char*) data, "Apogeu: %fm\n", flot);

```

```

271     CDC_Transmit_FS(data, strlen((char*) data));
272     //Consulta a la pagina 0, l'alçada de l'apertura del main del vol
273
274     memcpy(&flot, &bufferlec[vol * 4 * 8 + 28], sizeof(float));
275     sprintf((char*) data, "Main %fm\n", flot);
276     CDC_Transmit_FS(data, strlen((char*) data));
277
278     CDC_Transmit_FS(
279         (uint8_t*) "\n"
280             -----
281             n",
282             51);
283         break;
284
285         //Surt del menu
286         case '5':
287             return;
288             break;
289
290         //Entrada incorrecta
291         default:
292             CDC_Transmit_FS(
293                 (uint8_t*) "Entrada incorrecta\n"
294                     -----
295                     n",
296                     69);
297             break;
298         }
299     }
300
301     /*
302     * comprova les dades de la pagina 0 i permet llegir les dades del vol escollit, en cas
303     * que hi hagi.
304     */
305     void llegir_vols() {
306         uint8_t bufferlec[256];
307         uint8_t data[256];
308         //Llegeix la pagina 0
309         w25qxx_read(&w25qxx, 0, bufferlec, 255);
310         int i;
311         //Copia la posicio de memoria 12 de la pagina 0, que correspon al nombre de vols
312         //emmagatzemats.
313         memcpy(&i, &bufferlec[12], sizeof(int));
314         //Si no hi ha vols surt de la funcio.
315         if (i == 0) {
316             CDC_Transmit_FS(
317                 (uint8_t*) "No hi ha vols a la memoria\n"
318                     -----
319                     n",
320                     77);
321             return;
322         }
323         //mentre no s'esculli sortir es mante en la seleccio de vols.
324         while (1) {
325             //Dona les opcions de els vols.
326             for (int j = 1; j <= i; ++j) {
327                 sprintf((char*) data, "%d: Llegeix vol %d\n", j, j);
328                 CDC_Transmit_FS(data, strlen((char*) data));
329             }
330             buffer[0] = (uint8_t) '*';
331             sprintf((char*) data,

```

```

324             "%d: Tornar enrerrera\n"
325             -----\n",
326             i + 1);
327             CDC_Transmit_FS(data, strlen((char*) data));
328             //mentre no es rebi res pel USB es bloca
329             while (buffer[0] == (uint8_t) '*');

330             //Si hi ha una entrada incorrecta torna a començar
331             if ((buffer[0] - '0') > i + 1 || (buffer[0] - '0') < 1) {
332                 CDC_Transmit_FS(
333                     (uint8_t*) "Entrada incorrecta\n"
334                     -----\
335                     n",
336                     69);
337                     llegir_vols();
338             }
339             //Torna enrerrera.
340             else if ((buffer[0] - '0') == i + 1)
341                 return;
342             //Llegeix el vol seleccionat.
343             else {
344                 llegir_vol((buffer[0] - '0'), bufferlec);
345             }
346         }
347     /*
348     * Fa una rutina per fer una serie de proves predefinides.
349     */
350     void prova_mem() {
351         uint8_t bufferBar[255];
352         uint8_t bufferlec[255];
353         uint8_t *data1;
354         data1 =
355             (uint8_t*) "-----Proves Flash-----\n1.
356             Borra memoria:\nPagina 0:";
357             CDC_Transmit_FS(data1, strlen((char*) data1));

358             // Esborrar la memòria flash
359             w25qxx_chip_erase(&w25qxx);
360             w25qxx_read(&w25qxx, 0 * 256, bufferlec, 255);
361             CDC_Transmit_FS(bufferlec, 255);

362             // Verificar si la memòria està esborrada
363             data1 = (uint8_t*) "\nVerificant esborrat:\n";
364             CDC_Transmit_FS(data1, strlen((char*) data1));

365             for (int i = 0; i < 255; i++) {
366                 if (bufferlec[i] != 0xFF) {
367                     uint8_t error[50];
368                     sprintf((char*) error,
369                         "Error: Memòria no esborrada a la posició %d\n",
370                         i);
371                     CDC_Transmit_FS(error, strlen((char*) error));
372                     break;
373                 }
374             }

375             data1 = (uint8_t*) "\nPagina 1000:\n";
376             CDC_Transmit_FS(data1, strlen((char*) data1));
377
378
379

```

```

380     w25qxx_read(&w25qxx, 1000 * 256, bufferlec, 255);
381     CDC_Transmit_FS(bufferlec, 255);

382     // Verificar si la memòria està esborrada
383     data1 = (uint8_t*) "\nVerificant esborrat:\n";
384     CDC_Transmit_FS(data1, strlen((char*) data1));

385     for (int i = 0; i < 255; i++) {
386         if (bufferlec[i] != 0xFF) {
387             uint8_t error[50];
388             sprintf((char*) error,
389                     "Error: Memòria no esborrada a la posició %d\n",
390                     i);
391             CDC_Transmit_FS(error, strlen((char*) error));
392             break;
393         }
394     }

395     data1 =
396         (uint8_t*) "
397             -----\\n2.
398             Escriptura amb memoria buida:\n a Successio 1,2,3,4...255,
399             com a caracters:\n";
400     CDC_Transmit_FS(data1, strlen((char*) data1));

401     for (int i = 0; i <= 255; ++i)
402         bufferBar[i - 1] = (uint8_t) i;
403     w25qxx_write(&w25qxx, 0 * 256, bufferBar, 255);

404     w25qxx_read(&w25qxx, 0 * 256, bufferlec, 255);
405     CDC_Transmit_FS(bufferlec, 255);

406     // Verificar escriptura
407     data1 = (uint8_t*) "\nVerificant escriptura:\n";
408     CDC_Transmit_FS(data1, strlen((char*) data1));

409     for (int i = 0; i < 255; ++i) {
410         if (bufferlec[i] != bufferBar[i]) {
411             uint8_t error[50];
412             sprintf((char*) error,
413                     "Error: Escriptura fallida a la posició %d\n",
414                     i);
415             CDC_Transmit_FS(error, strlen((char*) error));
416             break;
417         }
418     }

419     data1 = (uint8_t*) "\\nb) Successio 0,4,8,12...255, com a floats:\n";
420     CDC_Transmit_FS(data1, strlen((char*) data1));

421     memset(bufferBar, 0, 255);
422     for (int i = 0; i < 64; i++) {
423         float value = i * 4.0;
424         memcpy(&bufferBar[i * 4], &value, sizeof(float)); // Copia la
425         // representació binària del float al buffer
426     }
427     w25qxx_write(&w25qxx, 1 * 256, bufferBar, sizeof(bufferBar));

428     // Llegir del dispositiu
429     w25qxx_read(&w25qxx, 1 * 256, bufferlec, sizeof(bufferlec));
430     for (int i = 0; i < 64; i++) {
431
432
433
434

```

```

435     float alt;
436     memcpy(&alt, &bufferlec[i * 4], sizeof(float)); // Copia la representació
437         binària del buffer al float
438     uint8_t data[20]; // Prou gran per contenir la cadena del float
439     sprintf((char*) data, "%f", alt);
440     CDC_Transmit_FS(data, strlen((char*) data));
441     CDC_Transmit_FS((uint8_t*) " ", 1);
442     HAL_Delay(5);
443 }
444
445 // Verificar escriptura
446 data1 = (uint8_t*) "\nVerificant escriptura:\n";
447 CDC_Transmit_FS(data1, strlen((char*) data1));
448
449 data1 = (uint8_t*) "\nc Successio 0,4,8,12...255, com a uint32:\n";
450 CDC_Transmit_FS(data1, strlen((char*) data1));
451
452 memset(bufferBar, 0, 255);
453 for (uint32_t i = 0; i < 255; i += 4) {
454     for (int j = 0; j < 4; j++)
455         bufferBar[i + j] = (i >> (8 * j)) & 0xFF;
456 }
457
458 w25qxx_write(&w25qxx, 2 * 256, bufferBar, sizeof(bufferBar));
459
460 // Llegir del dispositiu
461 w25qxx_read(&w25qxx, 2 * 256, bufferlec, sizeof(bufferlec));
462 for (int i = 0; i < 64; i++) {
463     uint32_t alt;
464     memcpy(&alt, &bufferlec[i * 4], sizeof(uint32_t)); // Copia la
465         representació binària del buffer al uint32_t
466     uint8_t data[20]; // Prou gran per contenir la cadena del uint32_t
467     sprintf((char*) data, "%lu", alt);
468     CDC_Transmit_FS(data, strlen((char*) data));
469     CDC_Transmit_FS((uint8_t*) " ", 1);
470     HAL_Delay(5);
471 }
472
473 // Verificar escriptura
474 data1 = (uint8_t*) "\nVerificant escriptura:\n";
475 CDC_Transmit_FS(data1, strlen((char*) data1));
476
477 for (int i = 0; i < 255; i += 4) {
478     uint32_t val = 0;
479     for (int j = 0; j < 4; j++)
480         val |= bufferlec[i + j] << (8 * j);
481     if (val != i) {
482         uint8_t error[50];
483         sprintf((char*) error,
484             "Error: Escriptura fallida a la posició %d\n", i)
485             ;
486         CDC_Transmit_FS(error, strlen((char*) error));
487         break;
488     }
489 }
490
491 data1 =
492     (uint8_t*) "\nd Successio 0,16,32,48...255, com a uint32 i float
493         intercalat:\n";
494 CDC_Transmit_FS(data1, strlen((char*) data1));

```

```

492     memset(bufferBar, 0, 255);
493     for (uint32_t i = 0; i < 64; i += 8) {
494         float value_f = i * 4.0;
495         uint32_t value_u = (i + 4) * 4;
496         memcpy(&bufferBar[i], &value_f, sizeof(float)); // Copia la representació
497             binària del float al buffer
498         memcpy(&bufferBar[i + 4], &value_u, sizeof(uint32_t)); // Copia la
499             representació binària del uint32_t al buffer
500     }
501     w25qxx_write(&w25qxx, 3 * 256, bufferBar, sizeof(bufferBar));
502
503     // Llegir del dispositiu
504     w25qxx_read(&w25qxx, 3 * 256, bufferlec, sizeof(bufferlec));
505     for (int i = 0; i < 64; i += 8) {
506         float alt_f;
507         uint32_t alt_u;
508         memcpy(&alt_f, &bufferlec[i], sizeof(float)); // Copia la representació
509             binària del buffer al float
510         memcpy(&alt_u, &bufferlec[i + 4], sizeof(uint32_t)); // Copia la
511             representació binària del buffer al uint32_t
512         uint8_t data[40]; // Prou gran per contenir la cadena del float i
513             uint32_t
514         sprintf((char*) data, "%f %lu", alt_f, alt_u);
515         CDC_Transmit_FS(data, strlen((char*) data));
516         CDC_Transmit_FS((uint8_t*) " ", 1);
517         HAL_Delay(5);
518     }
519
520     // Verificar escriptura
521     data1 = (uint8_t*) "\nVerificant escriptura:\n";
522     CDC_Transmit_FS(data1, strlen((char*) data1));
523
524     for (int i = 0; i < 255; i += 8) {
525         float val_f;
526         uint32_t val_u;
527         memcpy(&val_f, &bufferlec[i], sizeof(float)); // Copia la representació
528             binària del buffer al float
529         memcpy(&val_u, &bufferlec[i + 4], sizeof(uint32_t)); // Copia la
530             representació binària del buffer al uint32_t
531         if (val_f != i * 4.0 || val_u != i * 4) {
532             uint8_t error[50];
533             sprintf((char*) error,
534                 "Error: Escriptura fallida a la posició %d\n", i)
535             ;
536             CDC_Transmit_FS(error, strlen((char*) error));
537             break;
538         }
539     }
540
541     data1 =
542         (uint8_t*) "
543             -----\n";
544     CDC_Transmit_FS(data1, strlen((char*) data1));
545 }
546
547 /*
548 * Genera un vol per simular el funcionament de la memoria en un vol.
549 */
550 void vol_random() {
551     uint8_t bufferBar[256];
552     uint8_t bufferlec[256];

```

```

544 //Llegeix la pagina 0
545 w25qxx_read(&w25qxx, 0, bufferlec, 256);
546 int vols;
547 //consulta la quantitat de vols, a l posicio 12 de la pagina, i li suma un.
548 memcpy(&vols, &bufferlec[12], sizeof(int));
549 vols += 1;
550 memcpy(&bufferlec[12], &vols, sizeof(int));

551 //copia l'adreça de la ultima pagina utilitzada a l'ultim vol pel barometre, de
552 //la posicio 0 de la pagina 0, a la posicio 0 del nou vol.
553 memcpy(&bufferlec[vols * 8 * 4], &bufferlec[0], sizeof(int));
554 //copia l'adreça de la ultima pagina utilitzada a l'ultim vol per la IMU, de la
555 //posicio 4 de la pagina 0, a la posicio 4 del nou vol.
556 memcpy(&bufferlec[vols * 8 * 4 + 4], &bufferlec[4], sizeof(int));
557 //copia l'adreça de la ultima pagina utilitzada a l'ultim vol pel GPS, de la
558 //posicio 8 de la pagina 0, a la posicio 8 del nou vol.
559 memcpy(&bufferlec[vols * 8 * 4 + 8], &bufferlec[8], sizeof(int));

560 //copia un apogeu de 1000m a la posicio 24 del nou vol.
561 float flot = 1000.0;
562 memcpy(&bufferlec[vols * 8 * 4 + 24], &flot, sizeof(int));
563 //copia un main de 150m a la posicio 28 del nou vol.
564 flot = 150;
565 memcpy(&bufferlec[vols * 8 * 4 + 28], &flot, sizeof(int));

566 uint8_t *data = (uint8_t*) "Escrivint dades barometre\n";
567 CDC_Transmit_FS(data, strlen((char*) data));
568 int pag = 0;
569 //copia l'adreça de la ultima pagina utilitzada a l'ultim vol pel barometre, de
570 //la posicio 0 de la pagina 0,
571 //per començar a escriure dades del barometre
572 memcpy(&pag, &bufferlec[0], sizeof(int));
573 //escriu 4 pages amb noms ascendents i temps ascendent.
574 for (int j = 0; j < 4; j += 1) {
575     memset(bufferBar, 0, 256);
576     for (int i = 0; i < 256; i += 8) {
577         float value_f = i * 4.0;
578         uint32_t value_u = (i + 4) * 4;
579         memcpy(&bufferBar[i], &value_u, sizeof(uint32_t));
580         memcpy(&bufferBar[i + 4], &value_f, sizeof(float));
581     }
582     //Escriu la pagina a la memoria.
583     w25qxx_write(&w25qxx, pag * 256, bufferBar, sizeof(bufferBar));
584     HAL_Delay(100);
585     pag += 3;
586 }
587 //guarda la ultima pagina utilitzada a la posicio 0 de la pagina 0.
588 memcpy(&bufferlec[0], &pag, sizeof(int));
589 //guarda la ultima pagina utilitzada a la posicio 12 del vol.
590 memcpy(&bufferlec[vols * 8 * 4 + 12], &pag, sizeof(int));

591 data = (uint8_t*) "Escrivint dades IMU\n";
592 CDC_Transmit_FS(data, strlen((char*) data));
593 //copia l'adreça de la ultima pagina utilitzada a l'ultim vol pel barometre, de
594 //la posicio 4 de la pagina 0,
595 //per començar a escriure dades de la IMU.
596 memcpy(&pag, &bufferlec[4], sizeof(int));
597 for (int j = 0; j < 2; j += 1) {
598     memset(bufferBar, 0, 256);
599     for (int i = 0; i < 224; i += 28) {
600         float value_f = i * 4.0;
601     }
602 }
```

```

600     uint32_t value_u = (i + 4) * 4;
601     memcpy(&bufferBar[i], &value_u, sizeof(uint32_t));
602     memcpy(&bufferBar[i + 4], &value_f, sizeof(float));
603     memcpy(&bufferBar[i + 8], &value_f, sizeof(float));
604     memcpy(&bufferBar[i + 12], &value_f, sizeof(float));
605     memcpy(&bufferBar[i + 16], &value_f, sizeof(float));
606     memcpy(&bufferBar[i + 20], &value_f, sizeof(float));
607     memcpy(&bufferBar[i + 24], &value_f, sizeof(float));
608 }
609 //Escriu la pagina a la memoria.
610 w25qxx_write(&w25qxx, pag * 256, bufferBar, sizeof(bufferBar));
611 HAL_Delay(100);
612 pag += 3;
613 }
614 //guarda la ultima pagina utilitzada a la posicio 4 de la pagina 0.
615 memcpy(&bufferlec[4], &pag, sizeof(int));
616 //guarda la ultima pagina utilitzada a la posicio 16 del vol.
617 memcpy(&bufferlec[vols * 8 * 4 + 16], &pag, sizeof(int));
618
619
620 data = (uint8_t*) "Escrivint dades GPS\n";
621 CDC_Transmit_FS(data, strlen((char*) data));
622 //copia l'adreça de la ultima pagina utilitzada a l'ultim vol pel barometre, de
623 //la posicio 8 de la pagina 0,
624 //per començar a escriure dades del GPS.
625 memcpy(&pag, &bufferlec[8], sizeof(int));
626 for (int j = 0; j < 3; j += 1) {
627     memset(bufferBar, 0, 256);
628     for (int i = 0; i < 252; i += 12) {
629         float value_f = i * 4.0;
630         uint32_t value_u = (i + 4) * 4;
631         memcpy(&bufferBar[i], &value_u, sizeof(uint32_t));
632         memcpy(&bufferBar[i + 4], &value_f, sizeof(float));
633         memcpy(&bufferBar[i + 8], &value_f, sizeof(float));
634     }
635     //Escriu la pagina a la memoria.
636     w25qxx_write(&w25qxx, pag * 256, bufferBar, sizeof(bufferBar));
637     HAL_Delay(100);
638     pag += 3;
639 }
640 //guarda la ultima pagina utilitzada a la posicio 8 de la pagina 0.
641 memcpy(&bufferlec[8], &pag, sizeof(int));
642 //guarda la ultima pagina utilitzada a la posicio 20 del vol.
643 memcpy(&bufferlec[vols * 8 * 4 + 20], &pag, sizeof(int));
644 //Reescriu la pagina 0
645 w25qxx_erase(&w25qxx, 0, 256);
646 w25qxx_write(&w25qxx, 0, bufferlec, 256);
647 /*
648 * permet escollir quin tipus de prova es vol fer
649 */
650 void proves() {
651     while (1) {
652         buffer[0] = (uint8_t) '*';
653         CDC_Transmit_FS(
654             (uint8_t*) "1: Prova general \n2: Escriu vol random \n3:
655             Tornar enrera \n
656             -----\n",
657             108);
658         while (buffer[0] == (uint8_t) '*')
659             ;

```

```

658     switch (buffer[0]) {
659     case '1':
660         prova_mem();
661         break;
662     case '2':
663         vol_random();
664         break;
665     case '3':
666         return;
667         break;
668     default:
669         CDC_Transmit_FS(
670             (uint8_t*) "Entrada incorrecta\n"
671             -----
672             n",
673             69);
674         break;
675     }
676 }
677 /*
678 * Esborra tota la memoria
679 */
680 void borrar_memoria() {
681     uint8_t bufferBar[256];
682     buffer[0] = (uint8_t) '*';
683     uint8_t *data =
684         (uint8_t*) "Borraras tota la memoria, estás segur? \n1. => si,\n"
685         n2. => no\n";
686     CDC_Transmit_FS(data, strlen((char*) data));
687     //Pregunta per seguretat
688     while (buffer[0] == (uint8_t) '*');
689     if (buffer[0] == '1') {
690         data = (uint8_t*) "Borrant memoria\n";
691         CDC_Transmit_FS(data, strlen((char*) data));
692         //esborra la memoria
693         w25qxx_chip_erase(&w25qxx);
694         data = (uint8_t*) "Memoria borrada\n";
695         CDC_Transmit_FS(data, strlen((char*) data));
696         //Estableix els valors inicials de la pagina 0
697         memset(bufferBar, 0, 255);
698         int value = 21;
699         memcpy(&bufferBar[0], &value, sizeof(int));
700         value = 22;
701         memcpy(&bufferBar[4], &value, sizeof(int));
702         value = 23;
703         memcpy(&bufferBar[8], &value, sizeof(int));
704         value = 0;
705         memcpy(&bufferBar[12], &value, sizeof(int));
706         //escriu la pagina 0
707         w25qxx_write(&w25qxx, 0, bufferBar, sizeof(bufferBar));
708         CDC_Transmit_FS(
709             (uint8_t*) "\n"
710             -----
711             n",
712             51);
713     }
714 }
715 /* USER CODE END 0 */

```

```

715 /**
716 * @brief The application entry point.
717 * @retval int
718 */
719 int main(void) {
720     /* USER CODE BEGIN 1 */
721
722     /* USER CODE END 1 */
723
724     /* MCU Configuration-----*/
725
726     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
727     HAL_Init();
728
729     /* USER CODE BEGIN Init */
730
731     /* USER CODE END Init */
732
733     /* Configure the system clock */
734     SystemClock_Config();
735
736     /* USER CODE BEGIN SysInit */
737
738     /* USER CODE END SysInit */
739
740     /* Initialize all configured peripherals */
741     MX_GPIO_Init();
742     MX_ADC1_Init();
743     MX_I2C1_Init();
744     MX_I2C3_Init();
745     MX_SPI1_Init();
746     MX_SPI2_Init();
747     MX_USART2_UART_Init();
748     MX_TIM4_Init();
749     MX_USB_DEVICE_Init();
750     /* USER CODE BEGIN 2 */
751     /* USER CODE BEGIN 2 */
752
753     w25qxx_init(&w25qxx, &hspi1, FLASH_NSS_GPIO_Port, FLASH_NSS_Pin);
754     while (HAL_GPIO_ReadPin(SW_KEY_GPIO_Port, SW_KEY_Pin) != 1)
755         ;
756     buffer[0] = (uint8_t) '*';
757     uint8_t *data =
758         (uint8_t*) "Envia qualsevol resposta per accedir a la memoria
759         flash.\n-----\n";
760     CDC_Transmit_FS(data, strlen((char*) data));
761     while (buffer[0] == (uint8_t) '*')
762         ;
763     /* USER CODE END 2 */
764
765     /* Infinite loop */
766     /* USER CODE BEGIN WHILE */
767     while (1) {
768         buffer[0] = (uint8_t) '*';
769         data =
770             (uint8_t*) "1: Llegir memoria \n2: Borrar memoria \n3:
771             Proves de memoria \n
772             -----\n";
773         CDC_Transmit_FS(data, strlen((char*) data));
774         while (buffer[0] == (uint8_t) '*')
775             ;

```

```

773
774     switch (buffer[0]) {
775     case '1':
776         llegir_vols();
777         break;
778     case '2':
779         borrar_memoria();
780         break;
781     case '3':
782         proves();
783         break;
784     default:
785         CDC_Transmit_FS(
786             (uint8_t*) "Entrada incorrecta\n"
787             -----
788             n",
789             69);
790         break;
791     }
792
793     /* USER CODE END WHILE */
794
795 }
796 /* USER CODE END 3 */
797
798 /**
799 * @brief System Clock Configuration
800 * @retval None
801 */
802 void SystemClock_Config(void) {
803     RCC_OscInitTypeDef RCC_OscInitStruct = { 0 };
804     RCC_ClkInitTypeDef RCC_ClkInitStruct = { 0 };
805
806     /** Configure the main internal regulator output voltage
807     */
808     __HAL_RCC_PWR_CLK_ENABLE();
809     __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);
810
811     /** Initializes the RCC Oscillators according to the specified parameters
812     * in the RCC_OscInitTypeDef structure.
813     */
814     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI
815             | RCC_OSCILLATORTYPE_HSE;
816     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
817     RCC_OscInitStruct.HSISState = RCC_HSI_ON;
818     RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
819     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
820     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
821     RCC_OscInitStruct.PLL.PLLM = 4;
822     RCC_OscInitStruct.PLL.PLLN = 72;
823     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
824     RCC_OscInitStruct.PLL.PLLQ = 3;
825     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) {
826         Error_Handler();
827     }
828
829     /** Initializes the CPU, AHB and APB buses clocks
830     */
831     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK

```

```

832     | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
833     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
834     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
835     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
836     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
837
838     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK) {
839         Error_Handler();
840     }
841 }
842
843 /**
844 * @brief ADC1 Initialization Function
845 * @param None
846 * @retval None
847 */
848 static void MX_ADC1_Init(void) {
849
850     /* USER CODE BEGIN ADC1_Init_0 */
851
852     /* USER CODE END ADC1_Init_0 */
853
854     ADC_ChannelConfTypeDef sConfig = { 0 };
855
856     /* USER CODE BEGIN ADC1_Init_1 */
857
858     /* USER CODE END ADC1_Init_1 */
859
860     /** Configure the global features of the ADC (Clock, Resolution, Data Alignment
861      and number of conversion)
862      */
863     hadc1.Instance = ADC1;
864     hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
865     hadc1.Init.Resolution = ADC_RESOLUTION_12B;
866     hadc1.Init.ScanConvMode = DISABLE;
867     hadc1.Init.ContinuousConvMode = ENABLE;
868     hadc1.Init.DiscontinuousConvMode = DISABLE;
869     hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
870     hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
871     hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
872     hadc1.Init.NbrOfConversion = 1;
873     hadc1.Init.DMAContinuousRequests = DISABLE;
874     hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
875     if (HAL_ADC_Init(&hadc1) != HAL_OK) {
876         Error_Handler();
877     }
878
879     /** Configure for the selected ADC regular channel its corresponding rank in the
880      sequencer and its sample time.
881      */
882     sConfig.Channel = ADC_CHANNEL_8;
883     sConfig.Rank = 1;
884     sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
885     if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
886         Error_Handler();
887     }
888     /* USER CODE BEGIN ADC1_Init_2 */
889
890 }

```

```

891 /**
892 * @brief I2C1 Initialization Function
893 * @param None
894 * @retval None
895 */
896 static void MX_I2C1_Init(void) {
897
898     /* USER CODE BEGIN I2C1_Init_0 */
899
900     /* USER CODE END I2C1_Init_0 */
901
902     /* USER CODE BEGIN I2C1_Init_1 */
903
904     /* USER CODE END I2C1_Init_1 */
905     hi2c1.Instance = I2C1;
906     hi2c1.Init.ClockSpeed = 100000;
907     hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
908     hi2c1.Init.OwnAddress1 = 0;
909     hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
910     hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
911     hi2c1.Init.OwnAddress2 = 0;
912     hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
913     hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
914     if (HAL_I2C_Init(&hi2c1) != HAL_OK) {
915         Error_Handler();
916     }
917     /* USER CODE BEGIN I2C1_Init_2 */
918
919     /* USER CODE END I2C1_Init_2 */
920
921 }
922 /**
923 * @brief I2C3 Initialization Function
924 * @param None
925 * @retval None
926 */
927 static void MX_I2C3_Init(void) {
928
929     /* USER CODE BEGIN I2C3_Init_0 */
930
931     /* USER CODE END I2C3_Init_0 */
932
933     /* USER CODE BEGIN I2C3_Init_1 */
934
935     /* USER CODE END I2C3_Init_1 */
936     hi2c3.Instance = I2C3;
937     hi2c3.Init.ClockSpeed = 100000;
938     hi2c3.Init.DutyCycle = I2C_DUTYCYCLE_2;
939     hi2c3.Init.OwnAddress1 = 0;
940     hi2c3.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
941     hi2c3.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
942     hi2c3.Init.OwnAddress2 = 0;
943     hi2c3.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
944     hi2c3.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
945     if (HAL_I2C_Init(&hi2c3) != HAL_OK) {
946         Error_Handler();
947     }
948     /* USER CODE BEGIN I2C3_Init_2 */
949
950     /* USER CODE END I2C3_Init_2 */
951

```

```

952         /* USER CODE END I2C3_Init_2 */
953
954     }
955
956 /**
957 * @brief SPI1 Initialization Function
958 * @param None
959 * @retval None
960 */
961 static void MX_SPI1_Init(void) {
962
963     /* USER CODE BEGIN SPI1_Init_0 */
964
965     /* USER CODE END SPI1_Init_0 */
966
967     /* USER CODE BEGIN SPI1_Init_1 */
968
969     /* USER CODE END SPI1_Init_1 */
970     /* SPI1 parameter configuration*/
971     hspi1.Instance = SPI1;
972     hspi1.Init.Mode = SPI_MODE_MASTER;
973     hspi1.Init.Direction = SPI_DIRECTION_2LINES;
974     hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
975     hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
976     hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
977     hspi1.Init.NSS = SPI_NSS_SOFT;
978     hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
979     hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
980     hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
981     hspi1.Init.CRCCalculation = SPI_CRCALCULATION_DISABLE;
982     hspi1.Init.CRCPolynomial = 10;
983     if (HAL_SPI_Init(&hspi1) != HAL_OK) {
984         Error_Handler();
985     }
986     /* USER CODE BEGIN SPI1_Init_2 */
987
988     /* USER CODE END SPI1_Init_2 */
989 }
990
991 /**
992 * @brief SPI2 Initialization Function
993 * @param None
994 * @retval None
995 */
996
997 static void MX_SPI2_Init(void) {
998
999     /* USER CODE BEGIN SPI2_Init_0 */
1000
1001     /* USER CODE END SPI2_Init_0 */
1002
1003     /* USER CODE BEGIN SPI2_Init_1 */
1004
1005     /* USER CODE END SPI2_Init_1 */
1006     /* SPI2 parameter configuration*/
1007     hspi2.Instance = SPI2;
1008     hspi2.Init.Mode = SPI_MODE_MASTER;
1009     hspi2.Init.Direction = SPI_DIRECTION_2LINES;
1010     hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
1011     hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
1012     hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;

```

```

1013     hspi2.Init.NSS = SPI_NSS_SOFT;
1014     hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
1015     hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
1016     hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
1017     hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
1018     hspi2.Init.CRCPolynomial = 10;
1019     if (HAL_SPI_Init(&hspi2) != HAL_OK) {
1020         Error_Handler();
1021     }
1022     /* USER CODE BEGIN SPI2_Init_2 */
1023
1024     /* USER CODE END SPI2_Init_2 */
1025
1026 }
1027 /**
1028 * @brief TIM4 Initialization Function
1029 * @param None
1030 * @retval None
1031 */
1032 static void MX_TIM4_Init(void) {
1033
1034     /* USER CODE BEGIN TIM4_Init_0 */
1035
1036     /* USER CODE END TIM4_Init_0 */
1037
1038     TIM_ClockConfigTypeDef sClockSourceConfig = { 0 };
1039     TIM_MasterConfigTypeDef sMasterConfig = { 0 };
1040     TIM_OC_InitTypeDef sConfigOC = { 0 };
1041
1042     /* USER CODE BEGIN TIM4_Init_1 */
1043
1044     /* USER CODE END TIM4_Init_1 */
1045     htim4.Instance = TIM4;
1046     htim4.Init.Prescaler = 31;
1047     htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
1048     htim4.Init.Period = 254;
1049     htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
1050     htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
1051     if (HAL_TIM_Base_Init(&htim4) != HAL_OK) {
1052         Error_Handler();
1053     }
1054     sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
1055     if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK) {
1056         Error_Handler();
1057     }
1058     if (HAL_TIM_PWM_Init(&htim4) != HAL_OK) {
1059         Error_Handler();
1060     }
1061     sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
1062     sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
1063     if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig)
1064         != HAL_OK) {
1065         Error_Handler();
1066     }
1067     sConfigOC.OCMode = TIM_OCMODE_PWM1;
1068     sConfigOC.Pulse = 0;
1069     sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
1070     sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
1071     if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_3)
1072         != HAL_OK) {
1073

```

## B.1 Memòria flash

---

```

1074             Error_Handler();
1075     }
1076     /* USER CODE BEGIN TIM4_Init_2 */
1077
1078     /* USER CODE END TIM4_Init_2 */
1079     HAL_TIM_MspPostInit(&htim4);
1080
1081 }
1082
1083 /**
1084 * @brief USART2 Initialization Function
1085 * @param None
1086 * @retval None
1087 */
1088 static void MX_USART2_UART_Init(void) {
1089
1090     /* USER CODE BEGIN USART2_Init_0 */
1091
1092     /* USER CODE END USART2_Init_0 */
1093
1094     /* USER CODE BEGIN USART2_Init_1 */
1095
1096     /* USER CODE END USART2_Init_1 */
1097     huart2.Instance = USART2;
1098     huart2.Init.BaudRate = 9600;
1099     huart2.Init.WordLength = UART_WORDLENGTH_8B;
1100     huart2.Init.StopBits = UART_STOPBITS_1;
1101     huart2.Init.Parity = UART_PARITY_NONE;
1102     huart2.Init.Mode = UART_MODE_TX_RX;
1103     huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
1104     huart2.Init.OverSampling = UART_OVERSAMPLING_16;
1105     if (HAL_UART_Init(&huart2) != HAL_OK) {
1106         Error_Handler();
1107     }
1108     /* USER CODE BEGIN USART2_Init_2 */
1109
1110     /* USER CODE END USART2_Init_2 */
1111 }
1112
1113 /**
1114 * @brief GPIO Initialization Function
1115 * @param None
1116 * @retval None
1117 */
1118
1119 static void MX_GPIO_Init(void) {
1120     GPIO_InitTypeDef GPIO_InitStruct = { 0 };
1121     /* USER CODE BEGIN MX_GPIO_Init_1 */
1122     /* USER CODE END MX_GPIO_Init_1 */
1123
1124     /* GPIO Ports Clock Enable */
1125     __HAL_RCC_GPIOC_CLK_ENABLE();
1126     __HAL_RCC_GPIOH_CLK_ENABLE();
1127     __HAL_RCC_GPIOA_CLK_ENABLE();
1128     __HAL_RCC_GPIOB_CLK_ENABLE();
1129     __HAL_RCC_GPIOD_CLK_ENABLE();
1130
1131     /*Configure GPIO pin Output Level */
1132     HAL_GPIO_WritePin(GPIOC,
1133                         ENABLE_1_Pin | IGNITE_1_Pin | LED0_Pin | LED1_Pin | LED2_Pin
1134                                         | LED3_Pin | FLASH_WP_Pin | SERVO1_Pin |

```

```

1135     SERVO2_Pin,
1136     GPIO_PIN_RESET);
1137
1138     /*Configure GPIO pin Output Level */
1139     HAL_GPIO_WritePin(FLASH_NSS_GPIO_Port, FLASH_NSS_Pin, GPIO_PIN_RESET);
1140
1141     /*Configure GPIO pin Output Level */
1142     HAL_GPIO_WritePin(GPIOB, LORA_NSS_Pin | INT_IMU_Pin, GPIO_PIN_RESET);
1143
1144     /*Configure GPIO pin Output Level */
1145     HAL_GPIO_WritePin(GPIOD,
1146                     LORA_RST_Pin | IGNITE_2_Pin | ENABLE_2_Pin | LORA0_Pin |
1147                     LORA_RX_Pin, GPIO_PIN_RESET);
1148
1149     /*Configure GPIO pins : ENABLE_1_Pin IGNITE_1_Pin LED0_Pin LED1_Pin
1150     LED2_Pin LED3_Pin FLASH_WP_Pin SERVO1_Pin
1151     SERVO2_Pin */
1152     GPIO_InitStruct.Pin = ENABLE_1_Pin | IGNITE_1_Pin | LED0_Pin | LED1_Pin
1153         | LED2_Pin | LED3_Pin | FLASH_WP_Pin | SERVO1_Pin | SERVO2_Pin;
1154     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
1155     GPIO_InitStruct.Pull = GPIO_NOPULL;
1156     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
1157     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
1158
1159     /*Configure GPIO pin : FLASH_NSS_Pin */
1160     GPIO_InitStruct.Pin = FLASH_NSS_Pin;
1161     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
1162     GPIO_InitStruct.Pull = GPIO_NOPULL;
1163     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
1164     HAL_GPIO_Init(FLASH_NSS_GPIO_Port, &GPIO_InitStruct);
1165
1166     /*Configure GPIO pins : LORA_NSS_Pin INT_IMU_Pin */
1167     GPIO_InitStruct.Pin = LORA_NSS_Pin | INT_IMU_Pin;
1168     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
1169     GPIO_InitStruct.Pull = GPIO_NOPULL;
1170     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
1171     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
1172
1173     /*Configure GPIO pins : LORA_RST_Pin IGNITE_2_Pin ENABLE_2_Pin LORA0_Pin
1174     LORA_TX_Pin LORA_RX_Pin */
1175     GPIO_InitStruct.Pin = LORA_RST_Pin | IGNITE_2_Pin | ENABLE_2_Pin | LORA0_Pin
1176         | LORA_RX_Pin | LORA_TX_Pin;
1177     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
1178     GPIO_InitStruct.Pull = GPIO_NOPULL;
1179     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
1180     HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
1181
1182     /*Configure GPIO pin : SW_KEY_Pin */
1183     GPIO_InitStruct.Pin = SW_KEY_Pin;
1184     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
1185     GPIO_InitStruct.Pull = GPIO_PULLDOWN;
1186     HAL_GPIO_Init(SW_KEY_GPIO_Port, &GPIO_InitStruct);
1187
1188     /* USER CODE BEGIN MX_GPIO_Init_2 */
1189     /* USER CODE END MX_GPIO_Init_2 */
1190 }
1191
1192 /* USER CODE BEGIN 4 */
1193 /* USER CODE END 4 */

```

```

1194 /**
1195  * @brief Period elapsed callback in non blocking mode
1196  * @note This function is called when TIM1 interrupt took place, inside
1197  * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment
1198  * a global variable "uwTick" used as application time base.
1199  * @param htim : TIM handle
1200  * @retval None
1201 */
1202
1203 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
1204     /* USER CODE BEGIN Callback 0 */
1205
1206     /* USER CODE END Callback 0 */
1207     if (htim->Instance == TIM1) {
1208         HAL_IncTick();
1209     }
1210     /* USER CODE BEGIN Callback 1 */
1211
1212     /* USER CODE END Callback 1 */
1213 }
1214
1215 /**
1216  * @brief This function is executed in case of error occurrence.
1217  * @retval None
1218 */
1219 void Error_Handler(void) {
1220     /* USER CODE BEGIN Error_Handler_Debug */
1221     /* User can add his own implementation to report the HAL error return state */
1222     __disable_irq();
1223     while (1) {
1224     }
1225     /* USER CODE END Error_Handler_Debug */
1226 }
1227
1228 #ifdef USE_FULL_ASSERT
1229 /**
1230  * @brief Reports the name of the source file and the source line number
1231  * where the assert_param error has occurred.
1232  * @param file: pointer to the source file name
1233  * @param line: assert_param error line source number
1234  * @retval None
1235 */
1236 void assert_failed(uint8_t *file, uint32_t line)
1237 {
1238     /* USER CODE BEGIN 6 */
1239     /* User can add his own implementation to report the file name and line number,
1240      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
1241     /* USER CODE END 6 */
1242 }
1243 #endif /* USE_FULL_ASSERT */

```

## B.2. Baròmetre

### Codi de la llibreria

```

1 /*
2  * MS5607.cpp
3  *
4  * Created on: Feb 25, 2024
5  * Author: User
6 */

```

```

7
8 #include <math.h>
9 #include <MS5607.h>
10
11 MS5607::MS5607(){}
12
13 MS5607::MS5607(I2C_HandleTypeDef hi2c1){
14     hi2c = hi2c1;
15 }
16
17 MS5607::MS5607(I2C_HandleTypeDef hi2c1, short address){
18     this->MS5607_ADDR = address;
19     hi2c = hi2c1;
20 }
21
22 // Initialise coefficient by reading calibration data
23 char MS5607::begin(){
24     return(readCalibration());
25 }
26
27 void MS5607::set(I2C_HandleTypeDef hi2c1){
28     hi2c = hi2c1;
29 }
30
31
32 // Fa un reset al barometre, i llegeix els valors de calibracio de la PROM
33 char MS5607::readCalibration(){
34     if(resetDevice()){
35         HAL_Delay(1000);
36         read_PROM(PROM_READ+2, C1);
37         HAL_Delay(1000);
38         read_PROM(PROM_READ+4, C2);
39         HAL_Delay(1000);
40         read_PROM(PROM_READ+6, C3);
41         HAL_Delay(1000);
42         read_PROM(PROM_READ+8, C4);
43         HAL_Delay(1000);
44         read_PROM(PROM_READ+10, C5);
45         HAL_Delay(1000);
46         read_PROM(PROM_READ+12, C6);
47         return (1);
48
49     } else return(0);
50 }
51
52 // fa un reset del barometre
53 char MS5607::resetDevice(void){
54     HAL_StatusTypeDef error = HAL_I2C_Master_Transmit(&hi2c, (MS5607_ADDR << 1), RESET,
55     1,1000);
56     if(error == HAL_OK) return(1);
57     else return(0);
58 }
59
60 // llegeix el valor de l'adreça de la PROM seleccionada
61 char MS5607::read_PROM(unsigned char address, unsigned int &value){
62     unsigned char data[2];
63     data[0] = 0xA0;
64
65     HAL_StatusTypeDef error = HAL_I2C_Master_Transmit(&hi2c, (MS5607_ADDR << 1), &
66     address, 2, 50);
67     if (error == HAL_OK){

```

```

66         HAL_I2C_Master_Receive(&hi2c, (MS5607_ADDR << 1) | 0x1, data, 2 ,50);
67         value = (((unsigned int)data[0] << 8)) | (unsigned int)data[1]);
68         return(1);
69     }
70     value = 0;
71     return(0);
72 }
73
74 // envia la comanda per començar a mesurar
75 char MS5607::startMeasurment(void){
76     HAL_StatusTypeDef error = HAL_I2C_Master_Transmit(&hi2c, (MS5607_ADDR << 1),
77                                         R_ADC, 1,50);
78     if(error == HAL_OK) return(1);
79     else return(0);
80 }
81
82 // envia la comanda per començar la conversio
83 char MS5607::startConversion(unsigned char CMD){
84     HAL_StatusTypeDef error = HAL_I2C_Master_Transmit(&hi2c, (MS5607_ADDR << 1), &CMD,
85                                         1,50);
86     if(error == HAL_OK) {
87         HAL_Delay(Conv_Delay);
88         return(1);
89     }
90     else return(0);
91 }
92
93 // llegeix els valors demanats
94 char MS5607::getDigitalValue(unsigned long &value){
95     unsigned char data[3];
96     HAL_I2C_Master_Receive(&hi2c, (MS5607_ADDR << 1) | 0x1, data, 3 ,1000);
97     value = (((unsigned long)data[0] << 16) | ((unsigned long)data[1] << 8) | (unsigned
98     long)data[2]);
99     return(1);
100 }
101
102 // llegeix els valors digitals de temp i press del MS5607
103 char MS5607::readDigitalValue(void){
104     if(startConversion(CONV_D1)){
105         if(startMeasurment()){
106             if(getDigitalValue(DP));
107         }
108     }else{return 0;}
109     if(startConversion(CONV_D2)){
110         if(startMeasurment()){
111             if(getDigitalValue(DT));
112         }
113     }else{return 0;}
114     return 1;
115 }
116
117 // obte la temperatura
118 float MS5607::getTemperature(void){
119     DT = (float)DT - ((float)C5)*((int)1<<8);
120     TEMP = 2000.0 + DT * ((float)C6)/(float)((long)1<<23);
121     return TEMP/100 ;
122 }
123
124 // obte la pressio
125 float MS5607::getPressure(void){
126     DT = (float)DT - ((float)C5)*((int)1<<8);

```

```

124 TEMP = 2000.0 + dT * ((float)C6)/((float)((long)1<<23));
125 OFF = (((int64_t)C2)*((long)1<<17)) + dT * ((float)C4)/((int)1<<6);
126 SENS = ((float)C1)*((long)1<<16) + dT * ((float)C3)/((int)1<<7);
127 float pa = (float)((float)DP/((long)1<<15));
128 float pb = (float)(SENS/((float)((long)1<<21)));
129 float pc = pa*pb;
130 float pd = (float)(OFF/((float)((long)1<<15)));
131 P = pc - pd;
132 return P/100;
133 }
134
135 // set OSR and select corresponding values for conversion commands & delay
136 void MS5607::setOSR(short OSR_U){
137     this->OSR = OSR_U;
138     switch (OSR) {
139         case 256:
140             CONV_D1 = 0x40;
141             CONV_D2 = 0x50;
142             Conv_Delay = 1;
143             break;
144         case 512:
145             CONV_D1 = 0x42;
146             CONV_D2 = 0x52;
147             Conv_Delay = 2;
148             break;
149         case 1024:
150             CONV_D1 = 0x44;
151             CONV_D2 = 0x54;
152             Conv_Delay = 3;
153             break;
154         case 2048:
155             CONV_D1 = 0x46;
156             CONV_D2 = 0x56;
157             Conv_Delay = 5;
158             break;
159         case 4096:
160             CONV_D1 = 0x48;
161             CONV_D2 = 0x58;
162             Conv_Delay = 10;
163             break;
164         default:
165             CONV_D1 = 0x40;
166             CONV_D2 = 0x50;
167             Conv_Delay = 1;
168             break;
169     }
170 }
171
172 // fa les conversions necessaries per obtindre l'altura
173 float MS5607::getAltitude(void){
174     float h,t,p;
175     t = getTemperature();
176     p = getPressure();
177     p = P0/p;
178     h = 153.84615*(pow(p,0.19) - 1)*(t+273.15);
179     return h;
180 }
```

### Codi de la prova

1 /\* USER CODE BEGIN Header \*/

```

2 /**
3 ****
4 * @file          : main.c
5 * @brief         : Main program body
6 ****
7 * @attention
8 *
9 * Copyright (c) 2024 STMicroelectronics.
10 * All rights reserved.
11 *
12 * This software is licensed under terms that can be found in the LICENSE file
13 * in the root directory of this software component.
14 * If no LICENSE file comes with this software, it is provided AS-IS.
15 *
16 ****
17 */
18 /* USER CODE END Header */
19 /* Includes -----*/
20 #include "main.h"
21 #include "usb_device.h"
22
23 /* Private includes -----*/
24 /* USER CODE BEGIN Includes */
25 #include "MS5607.h"
26 #include "usbd_cdc_if.h"
27 /* USER CODE END Includes */
28
29 /* Private typedef -----*/
30 /* USER CODE BEGIN PTD */
31
32 /* USER CODE END PTD */
33
34 /* Private define -----*/
35 /* USER CODE BEGIN PD */
36
37 /* USER CODE END PD */
38
39 /* Private macro -----*/
40 /* USER CODE BEGIN PM */
41
42 /* USER CODE END PM */
43
44 /* Private variables -----*/
45 ADC_HandleTypeDef hadc1;
46
47 I2C_HandleTypeDef hi2c1;
48 I2C_HandleTypeDef hi2c3;
49
50 SPI_HandleTypeDef hspi1;
51 SPI_HandleTypeDef hspi2;
52
53 TIM_HandleTypeDef htim4;
54
55 UART_HandleTypeDef huart2;
56
57 /* USER CODE BEGIN PV */
58
59 /* USER CODE END PV */
60
61 /* Private function prototypes -----*/
62 void SystemClock_Config(void);

```

```

63 static void MX_GPIO_Init(void);
64 static void MX_ADC1_Init(void);
65 static void MX_I2C1_Init(void);
66 static void MX_I2C3_Init(void);
67 static void MX_SPI1_Init(void);
68 static void MX_SPI2_Init(void);
69 static void MX_USART2_UART_Init(void);
70 static void MX_TIM4_Init(void);
71 /* USER CODE BEGIN PFP */
72
73 /* USER CODE END PFP */
74
75 /* Private user code -----*/
76 /* USER CODE BEGIN 0 */
77
78 /* USER CODE END 0 */
79
80 /**
81 * @brief The application entry point.
82 * @retval int
83 */
84 int main(void)
85 {
86     /* USER CODE BEGIN 1 */
87
88     /* USER CODE END 1 */
89
90     /* MCU Configuration-----*/
91
92     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
93     HAL_Init();
94
95     /* USER CODE BEGIN Init */
96
97     /* USER CODE END Init */
98
99     /* Configure the system clock */
100    SystemClock_Config();
101
102    /* USER CODE BEGIN SysInit */
103
104    /* USER CODE END SysInit */
105
106    /* Initialize all configured peripherals */
107    MX_GPIO_Init();
108    MX_ADC1_Init();
109    MX_I2C1_Init();
110    MX_I2C3_Init();
111    MX_SPI1_Init();
112    MX_SPI2_Init();
113    MX_USART2_UART_Init();
114    MX_TIM4_Init();
115    MX_USB_DEVICE_Init();
116    /* USER CODE BEGIN 2 */
117    MS5607 barometre;
118
119    MX_USB_DEVICE_Init();
120    while (HAL_GPIO_ReadPin(SW_KEY_GPIO_Port, SW_KEY_Pin) != 1);
121    barometre.set(hi2c3);
122    while (!barometre.begin()) {
123        // Si no es pot inicialitzar, envia un missatge d'error per UART

```

```

124         const char *error_msg = "Error inicialitzant el baròmetre MS5607.\n";
125         CDC_Transmit_FS( (uint8_t*)error_msg, strlen(error_msg));
126     }
127     /* USER CODE END 2 */
128
129     /* Infinite loop */
130     /* USER CODE BEGIN WHILE */
131     while (1)
132     {
133         // Llegeix els valors de temperatura i pressió
134         if (barometre.readDigitalValue()) {
135             float temperature = barometre.getTemperature();
136             float pressure = barometre.getPressure();
137             float altitude = barometre.getAltitude();
138
139             // Envia els valors per UART
140             char buffer[150];
141             sprintf(buffer, sizeof(buffer), "Temperatura: %.2f C, Pressio: %.2f
142                                         hPa, Altura: %.2f m\n", temperature, pressure, altitude);
143             CDC_Transmit_FS( (uint8_t*)buffer, strlen(buffer));
144         }
145         else { CDC_Transmit_FS((uint8_t*)"fallada en lectura\n", 20);}
146         // Espera un segon abans de tornar a llegir
147         HAL_Delay(1000);
148     /* USER CODE END WHILE */
149     /* USER CODE BEGIN 3 */
150     }
151     /* USER CODE END 3 */
152 }
153 /**
154 * @brief System Clock Configuration
155 * @retval None
156 */
157 void SystemClock_Config(void)
158 {
159     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
160     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
161
162     /** Configure the main internal regulator output voltage
163     */
164     __HAL_RCC_PWR_CLK_ENABLE();
165     __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);
166
167     /** Initializes the RCC Oscillators according to the specified parameters
168     * in the RCC_OscInitTypeDef structure.
169     */
170     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_HSE;
171     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
172     RCC_OscInitStruct.HSISState = RCC_HSI_ON;
173     RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
174     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
175     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
176     RCC_OscInitStruct.PLL.PLLM = 4;
177     RCC_OscInitStruct.PLL.PLLN = 72;
178     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
179     RCC_OscInitStruct.PLL.PLLQ = 3;
180     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
181     {
182         Error_Handler();
183     }

```

```

184
185     /** Initializes the CPU, AHB and APB buses clocks
186     */
187     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
188             |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
189     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
190     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
191     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
192     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
193
194     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
195     {
196         Error_Handler();
197     }
198 }
199
200 /**
201 * @brief ADC1 Initialization Function
202 * @param None
203 * @retval None
204 */
205 static void MX_ADC1_Init(void)
206 {
207
208     /* USER CODE BEGIN ADC1_Init_0 */
209
210     /* USER CODE END ADC1_Init_0 */
211
212     ADC_ChannelConfTypeDef sConfig = {0};
213
214     /* USER CODE BEGIN ADC1_Init_1 */
215
216     /* USER CODE END ADC1_Init_1 */
217
218     /** Configure the global features of the ADC (Clock, Resolution, Data Alignment and
219      number of conversion)
220      */
221     hadc1.Instance = ADC1;
222     hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
223     hadc1.Init.Resolution = ADC_RESOLUTION_12B;
224     hadc1.Init.ScanConvMode = DISABLE;
225     hadc1.Init.ContinuousConvMode = ENABLE;
226     hadc1.Init.DiscontinuousConvMode = DISABLE;
227     hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
228     hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
229     hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
230     hadc1.Init.NbrOfConversion = 1;
231     hadc1.Init.DMAContinuousRequests = DISABLE;
232     hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
233     if (HAL_ADC_Init(&hadc1) != HAL_OK)
234     {
235         Error_Handler();
236     }
237
238     /** Configure for the selected ADC regular channel its corresponding rank in the
239      sequencer and its sample time.
240      */
241     sConfig.Channel = ADC_CHANNEL_8;
242     sConfig.Rank = 1;
243     sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
244     if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)

```

```

243 {
244     Error_Handler();
245 }
246 /* USER CODE BEGIN ADC1_Init_2 */
247
248 /* USER CODE END ADC1_Init_2 */
249
250 }
251
252 /**
253 * @brief I2C1 Initialization Function
254 * @param None
255 * @retval None
256 */
257 static void MX_I2C1_Init(void)
258 {
259
260     /* USER CODE BEGIN I2C1_Init_0 */
261
262     /* USER CODE END I2C1_Init_0 */
263
264     /* USER CODE BEGIN I2C1_Init_1 */
265
266     /* USER CODE END I2C1_Init_1 */
267     hi2c1.Instance = I2C1;
268     hi2c1.Init.ClockSpeed = 100000;
269     hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
270     hi2c1.Init.OwnAddress1 = 0;
271     hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
272     hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
273     hi2c1.Init.OwnAddress2 = 0;
274     hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
275     hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
276     if (HAL_I2C_Init(&hi2c1) != HAL_OK)
277     {
278         Error_Handler();
279     }
280     /* USER CODE BEGIN I2C1_Init_2 */
281
282     /* USER CODE END I2C1_Init_2 */
283
284 }
285
286 /**
287 * @brief I2C3 Initialization Function
288 * @param None
289 * @retval None
290 */
291 static void MX_I2C3_Init(void)
292 {
293
294     /* USER CODE BEGIN I2C3_Init_0 */
295
296     /* USER CODE END I2C3_Init_0 */
297
298     /* USER CODE BEGIN I2C3_Init_1 */
299
300     /* USER CODE END I2C3_Init_1 */
301     hi2c3.Instance = I2C3;
302     hi2c3.Init.ClockSpeed = 100000;
303     hi2c3.Init.DutyCycle = I2C_DUTYCYCLE_2;

```

## B.2 Baròmetre

---

```

304     hi2c3.Init.OwnAddress1 = 0;
305     hi2c3.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
306     hi2c3.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
307     hi2c3.Init.OwnAddress2 = 0;
308     hi2c3.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
309     hi2c3.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
310     if (HAL_I2C_Init(&hi2c3) != HAL_OK)
311     {
312         Error_Handler();
313     }
314     /* USER CODE BEGIN I2C3_Init 2 */
315
316     /* USER CODE END I2C3_Init 2 */
317
318 }
319
320 /**
321 * @brief SPI1 Initialization Function
322 * @param None
323 * @retval None
324 */
325 static void MX_SPI1_Init(void)
326 {
327
328     /* USER CODE BEGIN SPI1_Init 0 */
329
330     /* USER CODE END SPI1_Init 0 */
331
332     /* USER CODE BEGIN SPI1_Init 1 */
333
334     /* USER CODE END SPI1_Init 1 */
335     /* SPI1 parameter configuration*/
336     hspi1.Instance = SPI1;
337     hspi1.Init.Mode = SPI_MODE_MASTER;
338     hspi1.Init.Direction = SPI_DIRECTION_2LINES;
339     hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
340     hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
341     hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
342     hspi1.Init.NSS = SPI NSS_SOFT;
343     hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
344     hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
345     hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
346     hspi1.Init.CRCCalculation = SPI_CRCALCULATION_DISABLE;
347     hspi1.Init.CRCPolynomial = 10;
348     if (HAL_SPI_Init(&hspi1) != HAL_OK)
349     {
350         Error_Handler();
351     }
352     /* USER CODE BEGIN SPI1_Init 2 */
353
354     /* USER CODE END SPI1_Init 2 */
355
356 }
357
358 /**
359 * @brief SPI2 Initialization Function
360 * @param None
361 * @retval None
362 */
363 static void MX_SPI2_Init(void)
364 {

```

## B.2 Baròmetre

---

```

365
366     /* USER CODE BEGIN SPI2_Init_0 */
367
368     /* USER CODE END SPI2_Init_0 */
369
370     /* USER CODE BEGIN SPI2_Init_1 */
371
372     /* USER CODE END SPI2_Init_1 */
373     /* SPI2 parameter configuration*/
374     hspi2.Instance = SPI2;
375     hspi2.Init.Mode = SPI_MODE_MASTER;
376     hspi2.Init.Direction = SPI_DIRECTION_2LINES;
377     hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
378     hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
379     hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
380     hspi2.Init.NSS = SPI_NSS_SOFT;
381     hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
382     hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
383     hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
384     hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
385     hspi2.Init.CRCPolynomial = 10;
386     if (HAL_SPI_Init(&hspi2) != HAL_OK)
387     {
388         Error_Handler();
389     }
390     /* USER CODE BEGIN SPI2_Init_2 */
391
392     /* USER CODE END SPI2_Init_2 */
393
394 }
395
396 /**
397 * @brief TIM4 Initialization Function
398 * @param None
399 * @retval None
400 */
401 static void MX_TIM4_Init(void)
402 {
403
404     /* USER CODE BEGIN TIM4_Init_0 */
405
406     /* USER CODE END TIM4_Init_0 */
407
408     TIM_ClockConfigTypeDef sClockSourceConfig = {0};
409     TIM_MasterConfigTypeDef sMasterConfig = {0};
410     TIM_OC_InitTypeDef sConfigOC = {0};
411
412     /* USER CODE BEGIN TIM4_Init_1 */
413
414     /* USER CODE END TIM4_Init_1 */
415     htim4.Instance = TIM4;
416     htim4.Init.Prescaler = 31;
417     htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
418     htim4.Init.Period = 254;
419     htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
420     htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
421     if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
422     {
423         Error_Handler();
424     }
425     sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;

```

```

426     if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
427     {
428         Error_Handler();
429     }
430     if (HAL_TIM_PWM_Init(&htim4) != HAL_OK)
431     {
432         Error_Handler();
433     }
434     sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
435     sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
436     if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL_OK)
437     {
438         Error_Handler();
439     }
440     sConfigOC.OCMode = TIM_OCMODE_PWM1;
441     sConfigOC.Pulse = 0;
442     sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
443     sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
444     if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_3) != HAL_OK)
445     {
446         Error_Handler();
447     }
448     /* USER CODE BEGIN TIM4_Init_2 */
449
450     /* USER CODE END TIM4_Init_2 */
451     HAL_TIM_MspPostInit(&htim4);
452 }
453 /**
454 * @brief USART2 Initialization Function
455 * @param None
456 * @retval None
457 */
458 static void MX_USART2_UART_Init(void)
459 {
460
461     /* USER CODE BEGIN USART2_Init_0 */
462
463     /* USER CODE END USART2_Init_0 */
464
465     /* USER CODE BEGIN USART2_Init_1 */
466
467     /* USER CODE END USART2_Init_1 */
468
469     /* USER CODE BEGIN USART2_Init_2 */
470     huart2.Instance = USART2;
471     huart2.Init.BaudRate = 9600;
472     huart2.Init.WordLength = UART_WORDLENGTH_8B;
473     huart2.Init.StopBits = UART_STOPBITS_1;
474     huart2.Init.Parity = UART_PARITY_NONE;
475     huart2.Init.Mode = UART_MODE_TX_RX;
476     huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
477     huart2.Init.OverSampling = UART_OVERSAMPLING_16;
478     if (HAL_UART_Init(&huart2) != HAL_OK)
479     {
480         Error_Handler();
481     }
482     /* USER CODE BEGIN USART2_Init_3 */
483
484     /* USER CODE END USART2_Init_3 */
485 }

```

```

487 /**
488  * @brief GPIO Initialization Function
489  * @param None
490  * @retval None
491  */
492
493 static void MX_GPIO_Init(void)
494 {
495     GPIO_InitTypeDef GPIO_InitStruct = {0};
496 /* USER CODE BEGIN MX_GPIO_Init_1 */
497 /* USER CODE END MX_GPIO_Init_1 */
498
499     /* GPIO Ports Clock Enable */
500     __HAL_RCC_GPIOC_CLK_ENABLE();
501     __HAL_RCC_GPIOH_CLK_ENABLE();
502     __HAL_RCC_GPIOA_CLK_ENABLE();
503     __HAL_RCC_GPIOB_CLK_ENABLE();
504     __HAL_RCC_GPIOD_CLK_ENABLE();
505
506     /*Configure GPIO pin Output Level */
507     HAL_GPIO_WritePin(GPIOC, ENABLE_1_Pin|IGNITE_1_Pin|LEDO_Pin|LED1_Pin
508                             |LED2_Pin|LED3_Pin|FLASH_WP_Pin|SERVO1_Pin
509                             |SERVO2_Pin, GPIO_PIN_RESET);
510
511     /*Configure GPIO pin Output Level */
512     HAL_GPIO_WritePin(FLASH_NSS_GPIO_Port, FLASH_NSS_Pin, GPIO_PIN_RESET);
513
514     /*Configure GPIO pin Output Level */
515     HAL_GPIO_WritePin(GPIOB, LORA NSS_Pin|INT_IMU_Pin, GPIO_PIN_RESET);
516
517     /*Configure GPIO pin Output Level */
518     HAL_GPIO_WritePin(GPIOD, LORA_RST_Pin|IGNITE_2_Pin|ENABLE_2_Pin|LORA0_Pin
519                             |LORA_TX_Pin|LORA_RX_Pin, GPIO_PIN_RESET);
520
521     /*Configure GPIO pins : ENABLE_1_Pin IGNITE_1_Pin LEDO_Pin LED1_Pin
522                             LED2_Pin LED3_Pin FLASH_WP_Pin SERVO1_Pin
523                             SERVO2_Pin */
524     GPIO_InitStruct.Pin = ENABLE_1_Pin|IGNITE_1_Pin|LEDO_Pin|LED1_Pin
525                             |LED2_Pin|LED3_Pin|FLASH_WP_Pin|SERVO1_Pin
526                             |SERVO2_Pin;
527     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
528     GPIO_InitStruct.Pull = GPIO_NOPULL;
529     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
530     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
531
532     /*Configure GPIO pin : FLASH_NSS_Pin */
533     GPIO_InitStruct.Pin = FLASH_NSS_Pin;
534     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
535     GPIO_InitStruct.Pull = GPIO_NOPULL;
536     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
537     HAL_GPIO_Init(FLASH_NSS_GPIO_Port, &GPIO_InitStruct);
538
539     /*Configure GPIO pins : LORA_NSS_Pin INT_IMU_Pin */
540     GPIO_InitStruct.Pin = LORA_NSS_Pin|INT_IMU_Pin;
541     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
542     GPIO_InitStruct.Pull = GPIO_NOPULL;
543     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
544     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
545
546     /*Configure GPIO pins : LORA_RST_Pin IGNITE_2_Pin ENABLE_2_Pin LORA0_Pin
547                             LORA_TX_Pin LORA_RX_Pin */

```

```

548     GPIO_InitStruct.Pin = LORA_RST_Pin|IGNITE_2_Pin|ENABLE_2_Pin|LORAO_Pin
549                         |LORA_TX_Pin|LORA_RX_Pin;
550     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
551     GPIO_InitStruct.Pull = GPIO_NOPULL;
552     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
553     HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
554
555     /*Configure GPIO pin : SW_KEY_Pin */
556     GPIO_InitStruct.Pin = SW_KEY_Pin;
557     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
558     GPIO_InitStruct.Pull = GPIO_PULLDOWN;
559     HAL_GPIO_Init(SW_KEY_GPIO_Port, &GPIO_InitStruct);
560
561     /* USER CODE BEGIN MX_GPIO_Init_2 */
562     /* USER CODE END MX_GPIO_Init_2 */
563 }
564
565     /* USER CODE BEGIN 4 */
566
567     /* USER CODE END 4 */
568
569 /**
570 * @brief Period elapsed callback in non blocking mode
571 * @note This function is called when TIM1 interrupt took place, inside
572 * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment
573 * a global variable "uwTick" used as application time base.
574 * @param htim : TIM handle
575 * @retval None
576 */
577 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
578 {
579     /* USER CODE BEGIN Callback_0 */
580
581     /* USER CODE END Callback_0 */
582     if (htim->Instance == TIM1) {
583         HAL_IncTick();
584     }
585     /* USER CODE BEGIN Callback_1 */
586
587     /* USER CODE END Callback_1 */
588 }
589
590 /**
591 * @brief This function is executed in case of error occurrence.
592 * @retval None
593 */
594 void Error_Handler(void)
595 {
596     /* USER CODE BEGIN Error_Handler_Debug */
597     /* User can add his own implementation to report the HAL error return state */
598     __disable_irq();
599     while (1)
600     {
601     }
602     /* USER CODE END Error_Handler_Debug */
603 }
604
605 #ifdef USE_FULL_ASSERT
606 /**
607 * @brief Reports the name of the source file and the source line number
608 * where the assert_param error has occurred.

```

```

609     * @param file: pointer to the source file name
610     * @param line: assert_param error line source number
611     * @retval None
612 */
613 void assert_failed(uint8_t *file, uint32_t line)
614 {
615     /* USER CODE BEGIN 6 */
616     /* User can add his own implementation to report the file name and line number,
617      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
618     /* USER CODE END 6 */
619 }
620 #endif /* USE_FULL_ASSERT */

```

### B.3. IMU

#### Codi de la llibreria

```

1  /*
2  * icm20948.c
3  *
4  * Created on: Dec 26, 2020
5  * Author: mokhwasomssi
6  */
7
8
9 #include "icm20948.h"
10 #include <math.h>
11
12 static float gyro_scale_factor;
13 static float accel_scale_factor;
14 static I2C_HandleTypeDef hi2c1;
15
16 /* Static Functions */
17 /*static void      cs_high();
18 static void      cs_low();*/
19
20 static void      select_user_bank(userbank ub);
21
22 static uint8_t   read_single_icm20948_reg(userbank ub, uint8_t reg);
23 static void      write_single_icm20948_reg(userbank ub, uint8_t reg, uint8_t val);
24 static uint8_t*  read_multiple_icm20948_reg(userbank ub, uint8_t reg, uint8_t len);
25 static void      write_multiple_icm20948_reg(userbank ub, uint8_t reg, uint8_t* val,
26                                               uint8_t len);
27
28 static uint8_t   read_single_ak09916_reg(uint8_t reg);
29 static void      write_single_ak09916_reg(uint8_t reg, uint8_t val);
30 static uint8_t*  read_multiple_ak09916_reg(uint8_t reg, uint8_t len);
31
32 /* Main Functions */
33 float calculateAngle(axeses* acceleration) {
34     float magnitude = sqrt(acceleration->x * acceleration->x +
35                           acceleration->y * acceleration->y +
36                           acceleration->z * acceleration->z);
37
38     // Calculate the angle
39     float angle = asin(acceleration->z / magnitude);
40
41     // Convert angle from radians to degrees
42     angle = angle * 180.0 / M_PI;
43

```

```

44         return angle;
45     }
46
47     void icm20948_i2c(I2C_HandleTypeDef hi2c) {
48         hi2c1 = hi2c;
49     }
50     void icm20948_init()
51     {
52         while(!icm20948_who_am_i());
53
54         icm20948_device_reset();
55         icm20948_wakeup();
56
57         icm20948_clock_source(1);
58         icm20948_odr_align_enable();
59
60         //icm20948_spi_slave_enable();
61
62         icm20948_gyro_low_pass_filter(0);
63         icm20948_accel_low_pass_filter(0);
64
65         icm20948_gyro_sample_rate_divider(0);
66         icm20948_accel_sample_rate_divider(0);
67
68         icm20948_gyro_calibration();
69         icm20948_accel_calibration();
70
71         icm20948_gyro_full_scale_select(_2000dps);
72         icm20948_accel_full_scale_select(_16g);
73     }
74
75     void ak09916_init()
76     {
77         icm20948_i2c_master_reset();
78         icm20948_i2c_master_enable();
79         icm20948_i2c_master_clk_fraq(7);
80
81         while(!ak09916_who_am_i());
82
83         ak09916_soft_reset();
84         ak09916_operation_mode_setting(continuous_measurement_100hz);
85     }
86
87     void icm20948_gyro_read(axeses* data)
88     {
89         uint8_t* temp = read_multiple_icm20948_reg(ub_0, B0_GYRO_XOUT_H, 6);
90
91         data->x = (int16_t)(temp[0] << 8 | temp[1]);
92         data->y = (int16_t)(temp[2] << 8 | temp[3]);
93         data->z = (int16_t)(temp[4] << 8 | temp[5]);
94     }
95
96     void icm20948_accel_read(axeses* data)
97     {
98         uint8_t* temp = read_multiple_icm20948_reg(ub_0, B0_ACCEL_XOUT_H, 6);
99
100        data->x = (int16_t)(temp[0] << 8 | temp[1]);
101        data->y = (int16_t)(temp[2] << 8 | temp[3]);
102        data->z = (int16_t)(temp[4] << 8 | temp[5]) + accel_scale_factor;
103        // Add scale factor because calibraiton function offset gravity acceleration.
104    }

```

```

105
106 bool ak09916_mag_read(axeses* data)
107 {
108     uint8_t* temp;
109     uint8_t drdy, hofl;      // data ready, overflow
110
111     drdy = read_single_ak09916_reg(MAG_ST1) & 0x01;
112     if(!drdy)      return false;
113
114     temp = read_multiple_ak09916_reg(MAG_HXL, 6);
115
116     hofl = read_single_ak09916_reg(MAG_ST2) & 0x08;
117     if(hofl)      return false;
118
119     data->x = (int16_t)(temp[1] << 8 | temp[0]);
120     data->y = (int16_t)(temp[3] << 8 | temp[2]);
121     data->z = (int16_t)(temp[5] << 8 | temp[4]);
122
123     return true;
124 }
125
126 void icm20948_gyro_read_dps(axeses* data)
127 {
128     icm20948_gyro_read(data);
129
130     data->x /= gyro_scale_factor;
131     data->y /= gyro_scale_factor;
132     data->z /= gyro_scale_factor;
133 }
134
135 void icm20948_accel_read_g(axeses* data)
136 {
137     icm20948_accel_read(data);
138
139     data->x /= accel_scale_factor;
140     data->y /= accel_scale_factor;
141     data->z /= accel_scale_factor;
142 }
143
144 bool ak09916_mag_read_uT(axeses* data)
145 {
146     axeses temp;
147     bool new_data = ak09916_mag_read(&temp);
148     if(!new_data)      return false;
149
150     data->x = (float)(temp.x * 0.15);
151     data->y = (float)(temp.y * 0.15);
152     data->z = (float)(temp.z * 0.15);
153
154     return true;
155 }
156
157 /* Sub Functions */
158
159 bool icm20948_who_am_i()
160 {
161     uint8_t icm20948_id = read_single_icm20948_reg(ub_0, B0_WHO_AM_I);
162
163     if(icm20948_id == ICM20948_ID)
164         return true;
165     else

```

```

166             return false;
167     }
168
169     bool ak09916_who_am_i()
170     {
171         uint8_t ak09916_id = read_single_ak09916_reg(MAG_WIA2);
172
173         if(ak09916_id == AK09916_ID)
174             return true;
175         else
176             return false;
177     }
178
179     void icm20948_device_reset()
180     {
181         write_single_icm20948_reg(ub_0, B0_PWR_MGMT_1, 0x80 | 0x41);
182         HAL_Delay(100);
183     }
184
185     void ak09916_soft_reset()
186     {
187         write_single_ak09916_reg(MAG_CNTL3, 0x01);
188         HAL_Delay(100);
189     }
190
191     void icm20948_wakeup()
192     {
193         uint8_t new_val = read_single_icm20948_reg(ub_0, B0_PWR_MGMT_1);
194         new_val &= 0xBF;
195
196         write_single_icm20948_reg(ub_0, B0_PWR_MGMT_1, new_val);
197         HAL_Delay(100);
198     }
199
200     void icm20948_sleep()
201     {
202         uint8_t new_val = read_single_icm20948_reg(ub_0, B0_PWR_MGMT_1);
203         new_val |= 0x40;
204
205         write_single_icm20948_reg(ub_0, B0_PWR_MGMT_1, new_val);
206         HAL_Delay(100);
207     }
208
209     void icm20948_spi_slave_enable()
210     {
211         uint8_t new_val = read_single_icm20948_reg(ub_0, B0_USER_CTRL);
212         new_val |= 0x10;
213
214         write_single_icm20948_reg(ub_0, B0_USER_CTRL, new_val);
215     }
216
217     void icm20948_i2c_master_reset()
218     {
219         uint8_t new_val = read_single_icm20948_reg(ub_0, B0_USER_CTRL);
220         new_val |= 0x02;
221
222         write_single_icm20948_reg(ub_0, B0_USER_CTRL, new_val);
223     }
224
225     void icm20948_i2c_master_enable()
226     {

```

```

227     uint8_t new_val = read_single_icm20948_reg(ub_0, B0_USER_CTRL);
228     new_val |= 0x20;
229
230     write_single_icm20948_reg(ub_0, B0_USER_CTRL, new_val);
231     HAL_Delay(100);
232 }
233
234 void icm20948_i2c_master_clk_fraq(uint8_t config)
235 {
236     uint8_t new_val = read_single_icm20948_reg(ub_3, B3_I2C_MST_CTRL);
237     new_val |= config;
238
239     write_single_icm20948_reg(ub_3, B3_I2C_MST_CTRL, new_val);
240 }
241
242 void icm20948_clock_source(uint8_t source)
243 {
244     uint8_t new_val = read_single_icm20948_reg(ub_0, B0_PWR_MGMT_1);
245     new_val |= source;
246
247     write_single_icm20948_reg(ub_0, B0_PWR_MGMT_1, new_val);
248 }
249
250 void icm20948_odr_align_enable()
251 {
252     write_single_icm20948_reg(ub_2, B2_ODR_ALIGN_EN, 0x01);
253 }
254
255 void icm20948_gyro_low_pass_filter(uint8_t config)
256 {
257     uint8_t new_val = read_single_icm20948_reg(ub_2, B2_GYRO_CONFIG_1);
258     new_val |= config << 3;
259
260     write_single_icm20948_reg(ub_2, B2_GYRO_CONFIG_1, new_val);
261 }
262
263 void icm20948_accel_low_pass_filter(uint8_t config)
264 {
265     uint8_t new_val = read_single_icm20948_reg(ub_2, B2_ACCEL_CONFIG);
266     new_val |= config << 3;
267
268     write_single_icm20948_reg(ub_2, B2_GYRO_CONFIG_1, new_val);
269 }
270
271 void icm20948_gyro_sample_rate_divider(uint8_t divider)
272 {
273     write_single_icm20948_reg(ub_2, B2_GYRO_SMPLRT_DIV, divider);
274 }
275
276 void icm20948_accel_sample_rate_divider(uint16_t divider)
277 {
278     uint8_t divider_1 = (uint8_t)(divider >> 8);
279     uint8_t divider_2 = (uint8_t)(0x0F & divider);
280
281     write_single_icm20948_reg(ub_2, B2_ACCEL_SMPLRT_DIV_1, divider_1);
282     write_single_icm20948_reg(ub_2, B2_ACCEL_SMPLRT_DIV_2, divider_2);
283 }
284
285 void ak09916_operation_mode_setting(operation_mode mode)
286 {
287     write_single_ak09916_reg(MAG_CNTL2, mode);

```

```

288         HAL_Delay(100);
289     }
290
291     void icm20948_gyro_calibration()
292     {
293         axeses temp;
294         int32_t gyro_bias[3] = {0};
295         uint8_t gyro_offset[6] = {0};
296
297         for(int i = 0; i < 100; i++)
298         {
299             icm20948_gyro_read(&temp);
300             gyro_bias[0] += temp.x;
301             gyro_bias[1] += temp.y;
302             gyro_bias[2] += temp.z;
303         }
304
305         gyro_bias[0] /= 100;
306         gyro_bias[1] /= 100;
307         gyro_bias[2] /= 100;
308
309         // Construct the gyro biases for push to the hardware gyro bias registers,
310         // which are reset to zero upon device startup.
311         // Divide by 4 to get 32.9 LSB per deg/s to conform to expected bias input format
312
313         // Biases are additive, so change sign on calculated average gyro biases
314         gyro_offset[0] = (-gyro_bias[0] / 4 >> 8) & 0xFF;
315         gyro_offset[1] = (-gyro_bias[0] / 4) & 0xFF;
316         gyro_offset[2] = (-gyro_bias[1] / 4 >> 8) & 0xFF;
317         gyro_offset[3] = (-gyro_bias[1] / 4) & 0xFF;
318         gyro_offset[4] = (-gyro_bias[2] / 4 >> 8) & 0xFF;
319         gyro_offset[5] = (-gyro_bias[2] / 4) & 0xFF;
320
321         write_multiple_icm20948_reg(ub_2, B2_XG_OFFSETS_USRH, gyro_offset, 6);
322     }
323
324     void icm20948_accel_calibration()
325     {
326         axeses temp;
327         uint8_t* temp2;
328         uint8_t* temp3;
329         uint8_t* temp4;
330
331         int32_t accel_bias[3] = {0};
332         int32_t accel_bias_reg[3] = {0};
333         uint8_t accel_offset[6] = {0};
334
335         for(int i = 0; i < 100; i++)
336         {
337             icm20948_accel_read(&temp);
338             accel_bias[0] += temp.x;
339             accel_bias[1] += temp.y;
340             accel_bias[2] += temp.z;
341         }
342
343         accel_bias[0] /= 100;
344         accel_bias[1] /= 100;
345         accel_bias[2] /= 100;
346
347         uint8_t mask_bit[3] = {0, 0, 0};

```

```

348     temp2 = read_multiple_icm20948_reg(ub_1, B1_XA_OFFSET_H, 2);
349     accel_bias_reg[0] = (int32_t)(temp2[0] << 8 | temp2[1]);
350     mask_bit[0] = temp2[1] & 0x01;
351
352     temp3 = read_multiple_icm20948_reg(ub_1, B1_YA_OFFSET_H, 2);
353     accel_bias_reg[1] = (int32_t)(temp3[0] << 8 | temp3[1]);
354     mask_bit[1] = temp3[1] & 0x01;
355
356     temp4 = read_multiple_icm20948_reg(ub_1, B1_ZA_OFFSET_H, 2);
357     accel_bias_reg[2] = (int32_t)(temp4[0] << 8 | temp4[1]);
358     mask_bit[2] = temp4[1] & 0x01;
359
360     accel_bias_reg[0] -= (accel_bias[0] / 8);
361     accel_bias_reg[1] -= (accel_bias[1] / 8);
362     accel_bias_reg[2] -= (accel_bias[2] / 8);
363
364     accel_offset[0] = (accel_bias_reg[0] >> 8) & 0xFF;
365     accel_offset[1] = (accel_bias_reg[0]) & 0xFE;
366     accel_offset[1] = accel_offset[1] | mask_bit[0];
367
368     accel_offset[2] = (accel_bias_reg[1] >> 8) & 0xFF;
369     accel_offset[3] = (accel_bias_reg[1]) & 0xFE;
370     accel_offset[3] = accel_offset[3] | mask_bit[1];
371
372     accel_offset[4] = (accel_bias_reg[2] >> 8) & 0xFF;
373     accel_offset[5] = (accel_bias_reg[2]) & 0xFE;
374     accel_offset[5] = accel_offset[5] | mask_bit[2];
375
376     write_multiple_icm20948_reg(ub_1, B1_XA_OFFSET_H, &accel_offset[0], 2);
377     write_multiple_icm20948_reg(ub_1, B1_YA_OFFSET_H, &accel_offset[2], 2);
378     write_multiple_icm20948_reg(ub_1, B1_ZA_OFFSET_H, &accel_offset[4], 2);
379 }
380
381 void icm20948_gyro_full_scale_select(gyro_full_scale full_scale)
382 {
383     uint8_t new_val = read_single_icm20948_reg(ub_2, B2_GYRO_CONFIG_1);
384
385     switch(full_scale)
386     {
387         case _250dps :
388             new_val |= 0x00;
389             gyro_scale_factor = 131.0;
390             break;
391         case _500dps :
392             new_val |= 0x02;
393             gyro_scale_factor = 65.5;
394             break;
395         case _1000dps :
396             new_val |= 0x04;
397             gyro_scale_factor = 32.8;
398             break;
399         case _2000dps :
400             new_val |= 0x06;
401             gyro_scale_factor = 16.4;
402             break;
403     }
404
405     write_single_icm20948_reg(ub_2, B2_GYRO_CONFIG_1, new_val);
406 }
407
408 void icm20948_accel_full_scale_select(accel_full_scale full_scale)

```

```

409 {
410     uint8_t new_val = read_single_icm20948_reg(ub_2, B2_ACCEL_CONFIG);
411
412     switch(full_scale)
413     {
414         case _2g :
415             new_val |= 0x00;
416             accel_scale_factor = 16384;
417             break;
418         case _4g :
419             new_val |= 0x02;
420             accel_scale_factor = 8192;
421             break;
422         case _8g :
423             new_val |= 0x04;
424             accel_scale_factor = 4096;
425             break;
426         case _16g :
427             new_val |= 0x06;
428             accel_scale_factor = 2048;
429             break;
430     }
431
432     write_single_icm20948_reg(ub_2, B2_ACCEL_CONFIG, new_val);
433 }
434
435 /* Static Functions */
436 /*
437 static void cs_high()
438 {
439     HAL_GPIO_WritePin(ICM20948_SPI_CS_PIN_PORT, ICM20948_SPI_CS_PIN_NUMBER, SET);
440 }
441
442 static void cs_low()
443 {
444     HAL_GPIO_WritePin(ICM20948_SPI_CS_PIN_PORT, ICM20948_SPI_CS_PIN_NUMBER, RESET);
445 }
446 */
447
448 static void select_user_bank(userbank ub)
449 {
450     uint8_t write_reg[2];
451     write_reg[0] = REG_BANK_SEL;
452     write_reg[1] = ub;
453
454     HAL_I2C_Master_Transmit(&hi2c1, WRITE, write_reg, 2, 10);
455 }
456
457 static uint8_t read_single_icm20948_reg(userbank ub, uint8_t reg)
458 {
459     uint8_t reg_val;
460     select_user_bank(ub);
461
462     HAL_I2C_Master_Transmit(&hi2c1, WRITE, &reg, 1, 1000);
463     HAL_I2C_Master_Receive(&hi2c1, READ, &reg_val, 1, 1000);
464     return reg_val;
465 }
466
467 static void write_single_icm20948_reg(userbank ub, uint8_t reg, uint8_t val)
468 {
469     uint8_t write_reg[2];

```

```

470     write_reg[0] = reg;
471     write_reg[1] = val;
472
473     select_user_bank(ub);
474
475     HAL_I2C_Master_Transmit(&hi2c1, WRITE, write_reg, 2, 1000);
476 }
477
478 static uint8_t* read_multiple_icm20948_reg(userbank ub, uint8_t reg, uint8_t len)
479 {
480     static uint8_t reg_val[6];
481     select_user_bank(ub);
482
483     HAL_I2C_Master_Transmit(&hi2c1, WRITE, &reg, 1, 1000);
484     HAL_I2C_Master_Receive(&hi2c1, READ, reg_val, len, 1000);
485
486     return reg_val;
487 }
488
489 static void write_multiple_icm20948_reg(userbank ub, uint8_t reg, uint8_t* val, uint8_t
490 len)
491 {
492     select_user_bank(ub);
493
494     HAL_I2C_Master_Transmit(&hi2c1, WRITE, &reg, 1, 1000);
495     HAL_I2C_Master_Transmit(&hi2c1, WRITE, val, len, 1000);
496 }
497
498 static uint8_t read_single_ak09916_reg(uint8_t reg)
499 {
500     write_single_icm20948_reg(ub_3, B3_I2C_SLVO_ADDR, READ | MAG_SLAVE_ADDR);
501     write_single_icm20948_reg(ub_3, B3_I2C_SLVO_REG, reg);
502     write_single_icm20948_reg(ub_3, B3_I2C_SLVO_CTRL, 0x81);
503
504     HAL_Delay(1);
505     return read_single_icm20948_reg(ub_0, B0_EXT_SLV_SENS_DATA_00);
506 }
507
508 static void write_single_ak09916_reg(uint8_t reg, uint8_t val)
509 {
510     write_single_icm20948_reg(ub_3, B3_I2C_SLVO_ADDR, WRITE | MAG_SLAVE_ADDR);
511     write_single_icm20948_reg(ub_3, B3_I2C_SLVO_REG, reg);
512     write_single_icm20948_reg(ub_3, B3_I2C_SLVO_DO, val);
513     write_single_icm20948_reg(ub_3, B3_I2C_SLVO_CTRL, 0x81);
514 }
515
516 static uint8_t* read_multiple_ak09916_reg(uint8_t reg, uint8_t len)
517 {
518     write_single_icm20948_reg(ub_3, B3_I2C_SLVO_ADDR, READ | MAG_SLAVE_ADDR);
519     write_single_icm20948_reg(ub_3, B3_I2C_SLVO_REG, reg);
520     write_single_icm20948_reg(ub_3, B3_I2C_SLVO_CTRL, 0x80 | len);
521
522     HAL_Delay(1);
523     return read_multiple_icm20948_reg(ub_0, B0_EXT_SLV_SENS_DATA_00, len);
524 }
```

### Codi de la prova

```

1  /* USER CODE BEGIN Header */
2  /**
3  ****

```

```

4   * @file          : main.c
5   * @brief         : Main program body
6   ****
7   * @attention
8   *
9   * Copyright (c) 2024 STMicroelectronics.
10  * All rights reserved.
11  *
12  * This software is licensed under terms that can be found in the LICENSE file
13  * in the root directory of this software component.
14  * If no LICENSE file comes with this software, it is provided AS-IS.
15  *
16  ****
17  */
18 /* USER CODE END Header */
19 /* Includes -----*/
20 #include "main.h"
21 #include "usb_device.h"
22
23 /* Private includes -----*/
24 /* USER CODE BEGIN Includes */
25 #include "icm20948.h"
26 #include <stdio.h>
27 /* USER CODE END Includes */
28
29 /* Private typedef -----*/
30 /* USER CODE BEGIN PTD */
31
32 /* USER CODE END PTD */
33
34 /* Private define -----*/
35 /* USER CODE BEGIN PD */
36
37 /* USER CODE END PD */
38
39 /* Private macro -----*/
40 /* USER CODE BEGIN PM */
41
42 /* USER CODE END PM */
43
44 /* Private variables -----*/
45 ADC_HandleTypeDef hadc1;
46
47 I2C_HandleTypeDef hi2c1;
48 I2C_HandleTypeDef hi2c3;
49
50 SPI_HandleTypeDef hspi1;
51 SPI_HandleTypeDef hspi2;
52
53 TIM_HandleTypeDef htim4;
54
55 UART_HandleTypeDef huart2;
56
57 /* USER CODE BEGIN PV */
58 axes accel_data;
59 axes gyro_data;
60 axes mag_data;
61 float angle;
62 /* USER CODE END PV */
63
64 /* Private function prototypes -----*/

```

```

65 void SystemClock_Config(void);
66 static void MX_GPIO_Init(void);
67 static void MX_ADC1_Init(void);
68 static void MX_I2C1_Init(void);
69 static void MX_I2C3_Init(void);
70 static void MX_SPI1_Init(void);
71 static void MX_SPI2_Init(void);
72 static void MX_USART2_UART_Init(void);
73 static void MX_TIM4_Init(void);
74 /* USER CODE BEGIN PFP */
75
76 /* USER CODE END PFP */
77
78 /* Private user code -----*/
79 /* USER CODE BEGIN 0 */
80
81 /* USER CODE END 0 */
82
83 /**
84 * @brief The application entry point.
85 * @retval int
86 */
87 int main(void)
88 {
89 /* USER CODE BEGIN 1 */
90
91 /* USER CODE END 1 */
92
93 /* MCU Configuration-----*/
94
95 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
96 HAL_Init();
97
98 /* USER CODE BEGIN Init */
99
100 /* USER CODE END Init */
101
102 /* Configure the system clock */
103 SystemClock_Config();
104
105 /* USER CODE BEGIN SysInit */
106
107 /* USER CODE END SysInit */
108
109 /* Initialize all configured peripherals */
110 MX_GPIO_Init();
111 MX_ADC1_Init();
112 MX_I2C1_Init();
113 MX_I2C3_Init();
114 MX_SPI1_Init();
115 MX_SPI2_Init();
116 MX_USART2_UART_Init();
117 MX_TIM4_Init();
118 MX_USB_DEVICE_Init();
119 /* USER CODE BEGIN 2 */
120 icm20948_i2c(hi2c1);
121 icm20948_init();
122 ak09916_init();
123 /* USER CODE END 2 */
124
125 /* Infinite loop */

```

```

126  /* USER CODE BEGIN WHILE */
127  while (1)
128  {
129      /* USER CODE END WHILE */
130
131      /* USER CODE BEGIN 3 */
132      icm20948_accel_read_g(&accel_data);
133      icm20948_gyro_read_dps(&gyro_data);
134      if (ak09916_mag_read_uT(&mag_data)) {
135          angle = calculateAngle(&accel_data);
136
137          char buffer[150];
138          snprintf(buffer, sizeof(buffer), "Accel: X=%.2f, Y=%.2f, Z=%.2f\n", accel_data.x,
139                      accel_data.y, accel_data.z);
140          HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), HAL_MAX_DELAY);
141
142          snprintf(buffer, sizeof(buffer), "Gyro: X=%.2f, Y=%.2f, Z=%.2f\n", gyro_data.x,
143                      gyro_data.y, gyro_data.z);
144          HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer),
145                          HAL_MAX_DELAY);
146
147          snprintf(buffer, sizeof(buffer), "Mag: X=%.2f, Y=%.2f, Z=%.2f\n",
148                      mag_data.x, mag_data.y, mag_data.z);
149          HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer),
150                          HAL_MAX_DELAY);
151          HAL_Delay(1000);
152      }
153      /* USER CODE END 3 */
154  }
155
156 /**
157 * @brief System Clock Configuration
158 * @retval None
159 */
160 void SystemClock_Config(void)
161 {
162     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
163     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
164
165     /** Configure the main internal regulator output voltage
166     */
167     __HAL_RCC_PWR_CLK_ENABLE();
168     __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);
169
170     /** Initializes the RCC Oscillators according to the specified parameters
171     * in the RCC_OscInitTypeDef structure.
172     */
173     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_HSE;
174     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
175     RCC_OscInitStruct.HSISState = RCC_HSI_ON;
176     RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
177     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
178     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
179     RCC_OscInitStruct.PLL.PLLM = 8;
180     RCC_OscInitStruct.PLL.PLLN = 168;

```

```

181     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
182     RCC_OscInitStruct.PLL.PLLQ = 4;
183     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
184     {
185         Error_Handler();
186     }
187
188     /* Initializes the CPU, AHB and APB buses clocks
189 */
190     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
191             |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
192     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
193     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
194     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
195     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
196
197     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
198     {
199         Error_Handler();
200     }
201 }
202
203 /* ADC1 init function */
204 static void MX_ADC1_Init(void)
205 {
206
207     /* USER CODE BEGIN ADC1_Init_0 */
208
209     /* USER CODE END ADC1_Init_0 */
210
211     ADC_ChannelConfTypeDef sConfig = {0};
212
213     /* USER CODE BEGIN ADC1_Init_1 */
214
215     /* USER CODE END ADC1_Init_1 */
216
217     /* Configure the global features of the ADC (Clock, Resolution, Data Alignment and
218      number of conversion)
219 */
220     hadc1.Instance = ADC1;
221     hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
222     hadc1.Init.Resolution = ADC_RESOLUTION_12B;
223     hadc1.Init.ScanConvMode = DISABLE;
224     hadc1.Init.ContinuousConvMode = DISABLE;
225     hadc1.Init.DiscontinuousConvMode = DISABLE;
226     hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
227     hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
228     hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
229     hadc1.Init.NbrOfConversion = 1;
230     hadc1.Init.DMAContinuousRequests = DISABLE;
231     hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
232     if (HAL_ADC_Init(&hadc1) != HAL_OK)
233     {
234         Error_Handler();
235     }
236
237     /* Configure for the selected ADC regular channel its corresponding rank in the
238      sequencer and its sample time.
239 */
240     sConfig.Channel = ADC_CHANNEL_0;
241     sConfig.Rank = 1;

```

```

240     sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
241     if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
242     {
243         Error_Handler();
244     }
245
246     /* USER CODE BEGIN ADC1_Init 2 */
247
248     /* USER CODE END ADC1_Init 2 */
249
250 }
251
252 /* I2C1 init function */
253 static void MX_I2C1_Init(void)
254 {
255
256     /* USER CODE BEGIN I2C1_Init 0 */
257
258     /* USER CODE END I2C1_Init 0 */
259
260     /* USER CODE BEGIN I2C1_Init 1 */
261
262     /* USER CODE END I2C1_Init 1 */
263     hi2c1.Instance = I2C1;
264     hi2c1.Init.ClockSpeed = 100000;
265     hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
266     hi2c1.Init.OwnAddress1 = 0;
267     hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
268     hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
269     hi2c1.Init.OwnAddress2 = 0;
270     hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
271     hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
272     if (HAL_I2C_Init(&hi2c1) != HAL_OK)
273     {
274         Error_Handler();
275     }
276     /* USER CODE BEGIN I2C1_Init 2 */
277
278     /* USER CODE END I2C1_Init 2 */
279
280 }
281
282 /* I2C3 init function */
283 static void MX_I2C3_Init(void)
284 {
285
286     /* USER CODE BEGIN I2C3_Init 0 */
287
288     /* USER CODE END I2C3_Init 0 */
289
290     /* USER CODE BEGIN I2C3_Init 1 */
291
292     /* USER CODE END I2C3_Init 1 */
293     hi2c3.Instance = I2C3;
294     hi2c3.Init.ClockSpeed = 100000;
295     hi2c3.Init.DutyCycle = I2C_DUTYCYCLE_2;
296     hi2c3.Init.OwnAddress1 = 0;
297     hi2c3.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
298     hi2c3.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
299     hi2c3.Init.OwnAddress2 = 0;
300     hi2c3.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;

```

```

301     hi2c3.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
302     if (HAL_I2C_Init(&hi2c3) != HAL_OK)
303     {
304         Error_Handler();
305     }
306     /* USER CODE BEGIN I2C3_Init 2 */
307
308     /* USER CODE END I2C3_Init 2 */
309
310 }
311
312 /* SPI1 init function */
313 static void MX_SPI1_Init(void)
314 {
315
316     /* USER CODE BEGIN SPI1_Init 0 */
317
318     /* USER CODE END SPI1_Init 0 */
319
320     /* USER CODE BEGIN SPI1_Init 1 */
321
322     /* USER CODE END SPI1_Init 1 */
323     hspi1.Instance = SPI1;
324     hspi1.Init.Mode = SPI_MODE_MASTER;
325     hspi1.Init.Direction = SPI_DIRECTION_2LINES;
326     hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
327     hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
328     hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
329     hspi1.Init.NSS = SPI NSS_SOFT;
330     hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_16;
331     hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
332     hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
333     hspi1.Init.CRCCalculation = SPI_CRCALCULATION_DISABLE;
334     hspi1.Init.CRCPolynomial = 10;
335     if (HAL_SPI_Init(&hspi1) != HAL_OK)
336     {
337         Error_Handler();
338     }
339     /* USER CODE BEGIN SPI1_Init 2 */
340
341     /* USER CODE END SPI1_Init 2 */
342 }
343
344 /* SPI2 init function */
345 static void MX_SPI2_Init(void)
346 {
347
348     /* USER CODE BEGIN SPI2_Init 0 */
349
350     /* USER CODE END SPI2_Init 0 */
351
352     /* USER CODE BEGIN SPI2_Init 1 */
353
354     /* USER CODE END SPI2_Init 1 */
355     hspi2.Instance = SPI2;
356     hspi2.Init.Mode = SPI_MODE_MASTER;
357     hspi2.Init.Direction = SPI_DIRECTION_2LINES;
358     hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
359     hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
360     hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
361

```

```

362     hspi2.Init.NSS = SPI_NSS_SOFT;
363     hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_16;
364     hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
365     hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
366     hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
367     hspi2.Init.CRCPolynomial = 10;
368     if (HAL_SPI_Init(&hspi2) != HAL_OK)
369     {
370         Error_Handler();
371     }
372     /* USER CODE BEGIN SPI2_Init 2 */
373
374     /* USER CODE END SPI2_Init 2 */
375
376 }
377
378 /* TIM4 init function */
379 static void MX_TIM4_Init(void)
380 {
381
382     /* USER CODE BEGIN TIM4_Init 0 */
383
384     /* USER CODE END TIM4_Init 0 */
385
386     TIM_ClockConfigTypeDef sClockSourceConfig = {0};
387     TIM_MasterConfigTypeDef sMasterConfig = {0};
388
389     htim4.Instance = TIM4;
390     htim4.Init.Prescaler = 0;
391     htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
392     htim4.Init.Period = 0;
393     htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
394     htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
395     if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
396     {
397         Error_Handler();
398     }
399     sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
400     if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
401     {
402         Error_Handler();
403     }
404     sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
405     sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
406     if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL_OK)
407     {
408         Error_Handler();
409     }
410
411     /* USER CODE BEGIN TIM4_Init 2 */
412
413     /* USER CODE END TIM4_Init 2 */
414
415 }
416
417 /* USART2 init function */
418 static void MX_USART2_UART_Init(void)
419 {
420
421     /* USER CODE BEGIN USART2_Init 0 */
422

```

```

423  /* USER CODE END USART2_Init_0 */
424
425  /* USER CODE BEGIN USART2_Init_1 */
426
427  /* USER CODE END USART2_Init_1 */
428  huart2.Instance = USART2;
429  huart2.Init.BaudRate = 115200;
430  huart2.Init.WordLength = UART_WORDLENGTH_8B;
431  huart2.Init.StopBits = UART_STOPBITS_1;
432  huart2.Init.Parity = UART_PARITY_NONE;
433  huart2.Init.Mode = UART_MODE_TX_RX;
434  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
435  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
436  if (HAL_UART_Init(&huart2) != HAL_OK)
437  {
438      Error_Handler();
439  }
440  /* USER CODE BEGIN USART2_Init_2 */
441
442  /* USER CODE END USART2_Init_2 */
443
444 }
445
446 /* GPIO init function */
447 static void MX_GPIO_Init(void)
448 {
449     GPIO_InitTypeDef GPIO_InitStruct = {0};
450  /* USER CODE BEGIN MX_GPIO_Init_1 */
451  /* USER CODE END MX_GPIO_Init_1 */
452
453  /* GPIO Ports Clock Enable */
454  __HAL_RCC_GPIOC_CLK_ENABLE();
455  __HAL_RCC_GPIOH_CLK_ENABLE();
456  __HAL_RCC_GPIOA_CLK_ENABLE();
457  __HAL_RCC_GPIOB_CLK_ENABLE();
458  __HAL_RCC_GPIOD_CLK_ENABLE();
459
460  /*Configure GPIO pin Output Level */
461  HAL_GPIO_WritePin(GPIOC, ENABLE_1_Pin|IGNITE_1_Pin|LEDO_Pin|LED1_Pin
462                      |LED2_Pin|LED3_Pin|FLASH_WP_Pin|SERVO1_Pin
463                      |SERVO2_Pin, GPIO_PIN_RESET);
464
465  /*Configure GPIO pin Output Level */
466  HAL_GPIO_WritePin(FLASH_NSS_GPIO_Port, FLASH_NSS_Pin, GPIO_PIN_RESET);
467
468  /*Configure GPIO pin Output Level */
469  HAL_GPIO_WritePin(GPIOB, LORA_NSS_Pin|INT_IMU_Pin, GPIO_PIN_RESET);
470
471  /*Configure GPIO pin Output Level */
472  HAL_GPIO_WritePin(GPIOD, LORA_RST_Pin|IGNITE_2_Pin|ENABLE_2_Pin|LORA0_Pin
473                      |LORA_TX_Pin|LORA_RX_Pin, GPIO_PIN_RESET);
474
475  /*Configure GPIO pins : ENABLE_1_Pin IGNITE_1_Pin LED0_Pin LED1_Pin
476  LED2_Pin LED3_Pin FLASH_WP_Pin SERVO1_Pin
477  SERVO2_Pin */
478  GPIO_InitStruct.Pin = ENABLE_1_Pin|IGNITE_1_Pin|LEDO_Pin|LED1_Pin
479                      |LED2_Pin|LED3_Pin|FLASH_WP_Pin|SERVO1_Pin
480                      |SERVO2_Pin;
481  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
482  GPIO_InitStruct.Pull = GPIO_NOPULL;
483  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;

```

```

484     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
485
486     /*Configure GPIO pin : FLASH_NSS_Pin */
487     GPIO_InitStruct.Pin = FLASH_NSS_Pin;
488     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
489     GPIO_InitStruct.Pull = GPIO_NOPULL;
490     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
491     HAL_GPIO_Init(FLASH_NSS_GPIO_Port, &GPIO_InitStruct);
492
493     /*Configure GPIO pins : LORA_NSS_Pin INT_IMU_Pin */
494     GPIO_InitStruct.Pin = LORA_NSS_Pin|INT_IMU_Pin;
495     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
496     GPIO_InitStruct.Pull = GPIO_NOPULL;
497     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
498     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
499
500     /*Configure GPIO pins : LORA_RST_Pin IGNITE_2_Pin ENABLE_2_Pin LORA0_Pin
501          LORA_TX_Pin LORA_RX_Pin */
502     GPIO_InitStruct.Pin = LORA_RST_Pin|IGNITE_2_Pin|ENABLE_2_Pin|LORA0_Pin
503         |LORA_TX_Pin|LORA_RX_Pin;
504     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
505     GPIO_InitStruct.Pull = GPIO_NOPULL;
506     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
507     HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
508
509     /*Configure GPIO pin : SW_KEY_Pin */
510     GPIO_InitStruct.Pin = SW_KEY_Pin;
511     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
512     GPIO_InitStruct.Pull = GPIO_PULLDOWN;
513     HAL_GPIO_Init(SW_KEY_GPIO_Port, &GPIO_InitStruct);
514
515     /* USER CODE BEGIN MX_GPIO_Init_2 */
516     /* USER CODE END MX_GPIO_Init_2 */
517 }
518
519
520     /* USER CODE BEGIN 4 */
521
522     /* USER CODE END 4 */
523
524 /**
525     * @brief This function is executed in case of error occurrence.
526     * @retval None
527 */
528 void Error_Handler(void)
529 {
530     /* USER CODE BEGIN Error_Handler_Debug */
531     /* User can add his own implementation to report the HAL error return state */
532
533     /* USER CODE END Error_Handler_Debug */
534 }
535
536 #ifdef USE_FULL_ASSERT
537 /**
538     * @brief Reports the name of the source file and the source line number
539     *        where the assert_param error has occurred.
540     * @param file: pointer to the source file name
541     * @param line: assert_param error line source number
542     * @retval None
543 */
544 void assert_failed(uint8_t *file, uint32_t line)

```

```

545 {
546     /* USER CODE BEGIN 6 */
547     /* User can add his own implementation to report the file name and line number,
548        ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
549     /* USER CODE END 6 */
550 }
551 #endif /* USE_FULL_ASSERT */

```

## B.4. GNSS

### Codi de la llibreria

```

1 #include "nmea_parse.h"
2
3
4 char *data[15];
5
6 int gps_checksum(char *nmea_data) {
7     if(strlen(nmea_data) < 5) return 0;
8     char recv_crc[2];
9     recv_crc[0] = nmea_data[strlen(nmea_data) - 4];
10    recv_crc[1] = nmea_data[strlen(nmea_data) - 3];
11    int crc = 0;
12    int i;
13
14    for (i = 0; i < strlen(nmea_data) - 5; i++) {
15        crc ^= nmea_data[i];
16    }
17    int receivedHash = strtol(recv_crc, NULL, 16);
18    if (crc == receivedHash) {
19        return 1;
20    }
21    else {
22        return 0;
23    }
24}
25
26 int nmea_GPGGA(GPS *gps_data, char* inputString) {
27     char *values[25];
28     int counter = 0;
29     memset(values, 0, sizeof(values));
30     char *marker = strtok(inputString, ",");
31     while (marker != NULL) {
32         values[counter++] = strdup(marker); //free later!!!!!
33         marker = strtok(NULL, ",");
34     }
35     char lonSide = values[5][0];
36     char latSide = values[3][0];
37     strcpy(gps_data->lastMeasure, values[1]);
38     if (latSide == 'S' || latSide == 'N') {
39         char lat_d[3];
40         char lat_m[8];
41         strncpy(lat_d, values[2], 2);
42         strncpy(lat_m, values[2] + 2, 6);
43         lat_d[2] = '\0';
44         lat_m[6] = '\0';
45
46         int lat_deg_strtol = strtol(lat_d, NULL, 10);
47         float lat_min_strtof = strtof(lat_m, NULL);
48         double lat_deg = lat_deg_strtol + lat_min_strtof / 60;
49

```

```

50     char lon_d[4];
51     char lon_m[8];
52     strncpy(lon_d, values[4], 3);
53     strncpy(lon_m, values[4] + 3, 6);
54     lon_d[3] = '\0';
55     lon_m[6] = '\0';
56
57     int lon_deg_strtol = strtol(lon_d, NULL, 10);
58     float lon_min_strtof = strtof(lon_m, NULL);
59     double lon_deg = lon_deg_strtol + lon_min_strtof / 60;
60
61     if (lat_deg != 0 && lon_deg != 0 && lat_deg < 90 && lon_deg < 180) {
62         gps_data->latitude = lat_deg;
63         gps_data->latSide = latSide;
64         gps_data->longitude = lon_deg;
65         gps_data->lonSide = lonSide;
66         float altitude = strtof(values[9], NULL);
67         gps_data->altitude = altitude != 0 ? altitude : gps_data->altitude;
68         gps_data->satelliteCount = strtol(values[7], NULL, 10);
69
70         int fixQuality = strtol(values[6], NULL, 10);
71         gps_data->fix = fixQuality > 0 ? 1 : 0;
72
73         float hdop = strtof(values[8], NULL);
74         gps_data->hdop = hdop != 0 ? hdop : gps_data->hdop;
75     }
76     else {
77         for (int i = 0; i < counter; i++) free(values[i]);
78         return 0;
79     }
80 }
81
82 for (int i = 0; i < counter; i++) free(values[i]);
83 return 1;
84 }
85
86 int nmea_GPGSA(GPS *gps_data, char* inputString) {
87     char *values[25];
88     int counter = 0;
89     memset(values, 0, sizeof(values));
90     char *marker = strtok(inputString, ",");
91     while (marker != NULL) {
92         values[counter++] = strdup(marker); //free later!!!!!
93         marker = strtok(NULL, ",");
94     }
95     int fix = strtol(values[2], NULL, 10);
96     gps_data->fix = fix > 1 ? 1 : 0;
97     int satelliteCount = 0;
98     for (int i = 3; i < 15; i++) {
99         if (values[i][0] != '\0') {
100             satelliteCount++;
101         }
102     }
103     gps_data->satelliteCount = satelliteCount;
104     for (int i = 0; i < counter; i++) free(values[i]);
105     return 1;
106 }
107
108 int nmea_GPGLL(GPS *gps_data, char* inputString) {
109     char *values[25];
110     int counter = 0;

```

```

111     memset(values, 0, sizeof(values));
112     char *marker = strtok(inputString, ",");
113     while (marker != NULL) {
114         values[counter++] = strdup(marker); //free later!!!!!
115         marker = strtok(NULL, ",");
116     }
117     char latSide = values[2][0];
118     if (latSide == 'S' || latSide == 'N') {
119         char lat_d[3];
120         char lat_m[8];
121         strncpy(lat_d, values[1], 2);
122         strncpy(lat_m, values[1] + 2, 6);
123         lat_d[2] = '\0';
124         lat_m[6] = '\0';
125
126         int lat_deg_strtol = strtol(lat_d, NULL, 10);
127         float lat_min_strtof = strtof(lat_m, NULL);
128         double lat_deg = lat_deg_strtol + lat_min_strtof / 60;
129
130         char lon_d[4];
131         char lon_m[8];
132         char lonSide = values[4][0];
133         strncpy(lon_d, values[3], 3);
134         strncpy(lon_m, values[3] + 3, 6);
135         lon_d[3] = '\0';
136         lon_m[6] = '\0';
137
138         int lon_deg_strtol = strtol(lon_d, NULL, 10);
139         float lon_min_strtof = strtof(lon_m, NULL);
140         double lon_deg = lon_deg_strtol + lon_min_strtof / 60;
141
142         if (lon_deg_strtol == 0 || lon_min_strtof == 0 || lat_deg_strtol == 0 ||
143             lat_min_strtof == 0) {
144             for (int i = 0; i < counter; i++) free(values[i]);
145             return 0;
146         }
147         else {
148             gps_data->latitude = lat_deg;
149             gps_data->longitude = lon_deg;
150             gps_data->latSide = latSide;
151             gps_data->lonSide = lonSide;
152             for (int i = 0; i < counter; i++) free(values[i]);
153             return 1;
154         }
155     }
156 }
157
158 void nmea_parse(GPS *gps_data, uint8_t *buffer) {
159     memset(data, 0, sizeof(data));
160     char *token = strtok(reinterpret_cast<char *>(buffer), "$");
161     int cnt = 0;
162     while (token != NULL) {
163         data[cnt++] = strdup(token); //free later!!!!
164         token = strtok(NULL, "$");
165     }
166     for (int i = 0; i < cnt; i++) {
167         if (strstr(data[i], "\r\n") != NULL && gps_checksum(data[i])) {
168             if (strstr(data[i], "GPGLL") != NULL) {
169                 nmea_GPGLL(gps_data, data[i]);
170             }
171         }
172     }
173 }
```

```

171         else if (strstr(data[i], "GPGSA") != NULL) {
172             nmea_GPGSA(gps_data, data[i]);
173         }
174         else if (strstr(data[i], "GPGGA") != NULL) {
175             nmea_GPGGA(gps_data, data[i]);
176         }
177     }
178 }
179 for (int i = 0; i < cnt; i++) free(data[i]);
180 }
```

### Codi de la prova

```

1  /* USER CODE BEGIN Header */
2 /**
3  ****
4  * @file          : main.c
5  * @brief         : Main program body
6  ****
7  * @attention
8  *
9  * Copyright (c) 2024 STMicroelectronics.
10 * All rights reserved.
11 *
12 * This software is licensed under terms that can be found in the LICENSE file
13 * in the root directory of this software component.
14 * If no LICENSE file comes with this software, it is provided AS-IS.
15 *
16 ****
17 */
18 /* USER CODE END Header */
19 /* Includes -----*/
20 #include "main.h"
21 #include "usb_device.h"
22
23 /* Private includes -----*/
24 /* USER CODE BEGIN Includes */
25 #include "nmea_parse.h"
26 #include "usbd_cdc_if.h"
27 /* USER CODE END Includes */
28
29 /* Private typedef -----*/
30 /* USER CODE BEGIN PTD */
31
32 /* USER CODE END PTD */
33
34 /* Private define -----*/
35 /* USER CODE BEGIN PD */
36
37 /* USER CODE END PD */
38
39 /* Private macro -----*/
40 /* USER CODE BEGIN PM */
41
42 /* USER CODE END PM */
43
44 /* Private variables -----*/
45 ADC_HandleTypeDef hadc1;
46
47 I2C_HandleTypeDef hi2c1;
48 I2C_HandleTypeDef hi2c3;
```

```

49
50 SPI_HandleTypeDef hspi1;
51 SPI_HandleTypeDef hspi2;
52
53 TIM_HandleTypeDef htim4;
54
55 UART_HandleTypeDef huart2;
56
57 /* USER CODE BEGIN PV */
58
59 /* USER CODE END PV */
60
61 /* Private function prototypes -----*/
62 void SystemClock_Config(void);
63 static void MX_GPIO_Init(void);
64 static void MX_ADC1_Init(void);
65 static void MX_I2C1_Init(void);
66 static void MX_I2C3_Init(void);
67 static void MX_SPI1_Init(void);
68 static void MX_SPI2_Init(void);
69 static void MX_USART2_UART_Init(void);
70 static void MX_TIM4_Init(void);
71 /* USER CODE BEGIN PFP */
72
73 /* USER CODE END PFP */
74
75 /* Private user code -----*/
76 /* USER CODE BEGIN 0 */
77
78 /* USER CODE END 0 */
79
80 /**
81 * @brief The application entry point.
82 * @retval int
83 */
84 int main(void)
85 {
86     /* USER CODE BEGIN 1 */
87
88     /* USER CODE END 1 */
89
90     /* MCU Configuration-----*/
91
92     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
93     HAL_Init();
94
95     /* USER CODE BEGIN Init */
96
97     /* USER CODE END Init */
98
99     /* Configure the system clock */
100    SystemClock_Config();
101
102    /* USER CODE BEGIN SysInit */
103
104    /* USER CODE END SysInit */
105
106    /* Initialize all configured peripherals */
107    MX_GPIO_Init();
108    MX_ADC1_Init();
109    MX_I2C1_Init();

```

```

110     MX_I2C3_Init();
111     MX_SPI1_Init();
112     MX_SPI2_Init();
113     MX_USART2_UART_Init();
114     MX_TIM4_Init();
115     MX_USB_DEVICE_Init();
116     /* USER CODE BEGIN 2 */
117     uint8_t Rx_data[200];
118     HAL_UART_Transmit(&huart2, (uint8_t*)"$PMTK000*32\r\n\0", 15, 500);
119     HAL_UART_Receive(&huart2, Rx_data, 200, 100);
120     //HAL_UART_Transmit(&huart2, (uint8_t*)"$PMTK251,115200*1F<CR><LF>", 26, 500);
121     HAL_UART_Transmit(&huart2, (uint8_t*)"$PMTK314
122         ,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0*35\r\n\0", 55, 500);
123     HAL_UART_Receive(&huart2, Rx_data, 200, 100);
124     HAL_UART_Transmit(&huart2, (uint8_t*)"$PMTK220,100*2F\r\n\0", 21, 500);
125     HAL_UART_Receive(&huart2, Rx_data, 200, 100);
126     while (HAL_GPIO_ReadPin(SW_KEY_GPIO_Port, SW_KEY_Pin) != 1);
127     /* USER CODE END 2 */
128
129     /* Infinite loop */
130     /* USER CODE BEGIN WHILE */
131     while (1)
132     {
133
134         GPS gnss;
135         uint8_t buffer_gnss[128];
136         HAL_UART_Receive(&huart2, buffer_gnss, 128, 100);
137         nmea_parse(&gnss, buffer_gnss);
138         uint8_t data[10];
139         sprintf((char*)data, "%f", gnss.latitude);
140         CDC_Transmit_FS(data, 10);
141         CDC_Transmit_FS((uint8_t *)"||", 2);
142         sprintf((char*)data, "%f", gnss.longitude);
143         CDC_Transmit_FS(data, 10);
144         CDC_Transmit_FS((uint8_t *)"||", 2);
145         sprintf((char*)data, "%f", gnss.altitude);
146         CDC_Transmit_FS(data, 10);
147         CDC_Transmit_FS((uint8_t *)"\n", 2);
148         /* USER CODE END WHILE */
149
150     /* USER CODE BEGIN 3 */
151 }
152 /* USER CODE END 3 */
153 }
154 /**
155 * @brief System Clock Configuration
156 * @retval None
157 */
158 void SystemClock_Config(void)
159 {
160     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
161     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
162
163     /** Configure the main internal regulator output voltage
164     */
165     __HAL_RCC_PWR_CLK_ENABLE();
166     __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);
167
168     /** Initializes the RCC Oscillators according to the specified parameters
169     * in the RCC_OscInitTypeDef structure.

```

```

170  /*
171  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_HSE;
172  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
173  RCC_OscInitStruct.HSISState = RCC_HSI_ON;
174  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
175  RCC_OscInitStruct.PLL.PLLstate = RCC_PLL_ON;
176  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
177  RCC_OscInitStruct.PLL.PLLM = 4;
178  RCC_OscInitStruct.PLL.PLLN = 72;
179  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
180  RCC_OscInitStruct.PLL.PLLQ = 3;
181  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
182  {
183      Error_Handler();
184  }
185
186  /** Initializes the CPU, AHB and APB buses clocks
187  */
188  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
189          |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
190  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
191  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
192  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
193  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
194
195  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
196  {
197      Error_Handler();
198  }
199 }
200
201 /**
202 * @brief ADC1 Initialization Function
203 * @param None
204 * @retval None
205 */
206 static void MX_ADC1_Init(void)
207 {
208
209     /* USER CODE BEGIN ADC1_Init_0 */
210
211     /* USER CODE END ADC1_Init_0 */
212
213     ADC_ChannelConfTypeDef sConfig = {0};
214
215     /* USER CODE BEGIN ADC1_Init_1 */
216
217     /* USER CODE END ADC1_Init_1 */
218
219     /** Configure the global features of the ADC (Clock, Resolution, Data Alignment and
220         number of conversion)
221     */
222     hadc1.Instance = ADC1;
223     hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
224     hadc1.Init.Resolution = ADC_RESOLUTION_12B;
225     hadc1.Init.ScanConvMode = DISABLE;
226     hadc1.Init.ContinuousConvMode = ENABLE;
227     hadc1.Init.DiscontinuousConvMode = DISABLE;
228     hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
229     hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
230     hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;

```

```

230     hadc1.Init.NbrOfConversion = 1;
231     hadc1.Init.DMAContinuousRequests = DISABLE;
232     hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
233     if (HAL_ADC_Init(&hadc1) != HAL_OK)
234     {
235         Error_Handler();
236     }
237
238     /** Configure for the selected ADC regular channel its corresponding rank in the
239      sequencer and its sample time.
240 */
241     sConfig.Channel = ADC_CHANNEL_8;
242     sConfig.Rank = 1;
243     sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
244     if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
245     {
246         Error_Handler();
247     }
248     /* USER CODE BEGIN ADC1_Init 2 */
249
250     /* USER CODE END ADC1_Init 2 */
251 }
252
253 /**
254 * @brief I2C1 Initialization Function
255 * @param None
256 * @retval None
257 */
258 static void MX_I2C1_Init(void)
259 {
260
261     /* USER CODE BEGIN I2C1_Init 0 */
262
263     /* USER CODE END I2C1_Init 0 */
264
265     /* USER CODE BEGIN I2C1_Init 1 */
266
267     /* USER CODE END I2C1_Init 1 */
268     hi2c1.Instance = I2C1;
269     hi2c1.Init.ClockSpeed = 100000;
270     hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
271     hi2c1.Init.OwnAddress1 = 0;
272     hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
273     hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
274     hi2c1.Init.OwnAddress2 = 0;
275     hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
276     hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
277     if (HAL_I2C_Init(&hi2c1) != HAL_OK)
278     {
279         Error_Handler();
280     }
281     /* USER CODE BEGIN I2C1_Init 2 */
282
283     /* USER CODE END I2C1_Init 2 */
284 }
285
286 /**
287 * @brief I2C3 Initialization Function
288 * @param None
289 */

```

```

290     * @retval None
291     */
292 static void MX_I2C3_Init(void)
293 {
294     /* USER CODE BEGIN I2C3_Init 0 */
295
296     /* USER CODE END I2C3_Init 0 */
297
298     /* USER CODE BEGIN I2C3_Init 1 */
299
300     /* USER CODE END I2C3_Init 1 */
301     hi2c3.Instance = I2C3;
302     hi2c3.Init.ClockSpeed = 100000;
303     hi2c3.Init.DutyCycle = I2C_DUTYCYCLE_2;
304     hi2c3.Init.OwnAddress1 = 0;
305     hi2c3.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
306     hi2c3.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
307     hi2c3.Init.OwnAddress2 = 0;
308     hi2c3.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
309     hi2c3.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
310     if (HAL_I2C_Init(&hi2c3) != HAL_OK)
311     {
312         Error_Handler();
313     }
314     /* USER CODE BEGIN I2C3_Init 2 */
315
316     /* USER CODE END I2C3_Init 2 */
317
318 }
319
320 /**
321  * @brief SPI1 Initialization Function
322  * @param None
323  * @retval None
324  */
325
326 static void MX_SPI1_Init(void)
327 {
328     /* USER CODE BEGIN SPI1_Init 0 */
329
330     /* USER CODE END SPI1_Init 0 */
331
332     /* USER CODE BEGIN SPI1_Init 1 */
333
334     /* USER CODE END SPI1_Init 1 */
335     /* SPI1 parameter configuration*/
336     hspi1.Instance = SPI1;
337     hspi1.Init.Mode = SPI_MODE_MASTER;
338     hspi1.Init.Direction = SPI_DIRECTION_2LINES;
339     hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
340     hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
341     hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
342     hspi1.Init.NSS = SPI NSS_SOFT;
343     hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
344     hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
345     hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
346     hspi1.Init.CRCCalculation = SPI_CRCALCULATION_DISABLE;
347     hspi1.Init.CRCPolynomial = 10;
348     if (HAL_SPI_Init(&hspi1) != HAL_OK)
349     {

```

```

351     Error_Handler();
352 }
353 /* USER CODE BEGIN SPI1_Init_2 */
354
355 /* USER CODE END SPI1_Init_2 */
356
357 }
358
359 /**
360 * @brief SPI2 Initialization Function
361 * @param None
362 * @retval None
363 */
364 static void MX_SPI2_Init(void)
365 {
366
367     /* USER CODE BEGIN SPI2_Init_0 */
368
369     /* USER CODE END SPI2_Init_0 */
370
371     /* USER CODE BEGIN SPI2_Init_1 */
372
373     /* USER CODE END SPI2_Init_1 */
374     /* SPI2 parameter configuration*/
375     hspi2.Instance = SPI2;
376     hspi2.Init.Mode = SPI_MODE_MASTER;
377     hspi2.Init.Direction = SPI_DIRECTION_2LINES;
378     hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
379     hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
380     hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
381     hspi2.Init.NSS = SPI_NSS_SOFT;
382     hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
383     hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
384     hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
385     hspi2.Init.CRCCalculation = SPI_CRCALCULATION_DISABLE;
386     hspi2.Init.CRCPolynomial = 10;
387     if (HAL_SPI_Init(&hspi2) != HAL_OK)
388     {
389         Error_Handler();
390     }
391     /* USER CODE BEGIN SPI2_Init_2 */
392
393     /* USER CODE END SPI2_Init_2 */
394 }
395
396 /**
397 * @brief TIM4 Initialization Function
398 * @param None
399 * @retval None
400 */
401 static void MX_TIM4_Init(void)
402 {
403
404     /* USER CODE BEGIN TIM4_Init_0 */
405
406     /* USER CODE END TIM4_Init_0 */
407
408     TIM_ClockConfigTypeDef sClockSourceConfig = {0};
409     TIM_MasterConfigTypeDef sMasterConfig = {0};
410     TIM_OC_InitTypeDef sConfigOC = {0};
411

```

```

412
413 /* USER CODE BEGIN TIM4_Init_1 */
414
415 /* USER CODE END TIM4_Init_1 */
416 htim4.Instance = TIM4;
417 htim4.Init.Prescaler = 31;
418 htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
419 htim4.Init.Period = 254;
420 htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
421 htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
422 if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
423 {
424     Error_Handler();
425 }
426 sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
427 if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
428 {
429     Error_Handler();
430 }
431 if (HAL_TIM_PWM_Init(&htim4) != HAL_OK)
432 {
433     Error_Handler();
434 }
435 sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
436 sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
437 if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL_OK)
438 {
439     Error_Handler();
440 }
441 sConfigOC.OCMode = TIM_OCMODE_PWM1;
442 sConfigOC.Pulse = 0;
443 sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
444 sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
445 if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_3) != HAL_OK)
446 {
447     Error_Handler();
448 }
449 /* USER CODE BEGIN TIM4_Init_2 */
450
451 /* USER CODE END TIM4_Init_2 */
452 HAL_TIM_MspPostInit(&htim4);
453 }
454 /**
455 * @brief USART2 Initialization Function
456 * @param None
457 * @retval None
458 */
459 static void MX_USART2_UART_Init(void)
460 {
461
462     /* USER CODE BEGIN USART2_Init_0 */
463
464     /* USER CODE END USART2_Init_0 */
465
466     /* USER CODE BEGIN USART2_Init_1 */
467
468     /* USER CODE END USART2_Init_1 */
469
470     /* USER CODE END USART2_Init_1 */
471     huart2.Instance = USART2;
472     huart2.Init.BaudRate = 9600;

```

```

473     huart2.Init.WordLength = UART_WORDLENGTH_8B;
474     huart2.Init.StopBits = UART_STOPBITS_1;
475     huart2.Init.Parity = UART_PARITY_NONE;
476     huart2.Init.Mode = UART_MODE_TX_RX;
477     huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
478     huart2.Init.OverSampling = UART_OVERSAMPLING_16;
479     if (HAL_UART_Init(&huart2) != HAL_OK)
480     {
481         Error_Handler();
482     }
483     /* USER CODE BEGIN USART2_Init_2 */
484
485     /* USER CODE END USART2_Init_2 */
486
487 }
488
489 /**
490 * @brief GPIO Initialization Function
491 * @param None
492 * @retval None
493 */
494 static void MX_GPIO_Init(void)
495 {
496     GPIO_InitTypeDef GPIO_InitStruct = {0};
497     /* USER CODE BEGIN MX_GPIO_Init_1 */
498     /* USER CODE END MX_GPIO_Init_1 */
499
500     /* GPIO Ports Clock Enable */
501     __HAL_RCC_GPIOC_CLK_ENABLE();
502     __HAL_RCC_GPIOH_CLK_ENABLE();
503     __HAL_RCC_GPIOA_CLK_ENABLE();
504     __HAL_RCC_GPIOB_CLK_ENABLE();
505     __HAL_RCC_GPIOD_CLK_ENABLE();
506
507     /*Configure GPIO pin Output Level */
508     HAL_GPIO_WritePin(GPIOC, ENABLE_1_Pin|IGNITE_1_Pin|LEDO_Pin|LED1_Pin
509                         |LED2_Pin|LED3_Pin|FLASH_WP_Pin|SERVO1_Pin
510                         |SERVO2_Pin, GPIO_PIN_RESET);
511
512     /*Configure GPIO pin Output Level */
513     HAL_GPIO_WritePin(FLASH_NSS_GPIO_Port, FLASH_NSS_Pin, GPIO_PIN_RESET);
514
515     /*Configure GPIO pin Output Level */
516     HAL_GPIO_WritePin(GPIOB, LORA_NSS_Pin|INT_IMU_Pin, GPIO_PIN_RESET);
517
518     /*Configure GPIO pin Output Level */
519     HAL_GPIO_WritePin(GPIOD, LORA_RST_Pin|IGNITE_2_Pin|ENABLE_2_Pin|LORA0_Pin
520                         |LORA_TX_Pin|LORA_RX_Pin, GPIO_PIN_RESET);
521
522     /*Configure GPIO pins : ENABLE_1_Pin IGNITE_1_Pin LEDO_Pin LED1_Pin
523                             LED2_Pin LED3_Pin FLASH_WP_Pin SERVO1_Pin
524                             SERVO2_Pin */
525     GPIO_InitStruct.Pin = ENABLE_1_Pin|IGNITE_1_Pin|LEDO_Pin|LED1_Pin
526                         |LED2_Pin|LED3_Pin|FLASH_WP_Pin|SERVO1_Pin
527                         |SERVO2_Pin;
528     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
529     GPIO_InitStruct.Pull = GPIO_NOPULL;
530     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
531     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
532
533     /*Configure GPIO pin : FLASH_NSS_Pin */

```

```

534     GPIO_InitStruct.Pin = FLASH_NSS_Pin;
535     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
536     GPIO_InitStruct.Pull = GPIO_NOPULL;
537     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
538     HAL_GPIO_Init(FLASH_NSS_GPIO_Port, &GPIO_InitStruct);
539
540     /*Configure GPIO pins : LORA_NSS_Pin INT_IMU_Pin */
541     GPIO_InitStruct.Pin = LORA_NSS_Pin|INT_IMU_Pin;
542     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
543     GPIO_InitStruct.Pull = GPIO_NOPULL;
544     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
545     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
546
547     /*Configure GPIO pins : LORA_RST_Pin IGNITE_2_Pin ENABLE_2_Pin LORA0_Pin
548      LORA_TX_Pin LORA_RX_Pin */
549     GPIO_InitStruct.Pin = LORA_RST_Pin|IGNITE_2_Pin|ENABLE_2_Pin|LORA0_Pin
550             |LORA_TX_Pin|LORA_RX_Pin;
551     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
552     GPIO_InitStruct.Pull = GPIO_NOPULL;
553     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
554     HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
555
556     /*Configure GPIO pin : SW_KEY_Pin */
557     GPIO_InitStruct.Pin = SW_KEY_Pin;
558     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
559     GPIO_InitStruct.Pull = GPIO_PULLDOWN;
560     HAL_GPIO_Init(SW_KEY_GPIO_Port, &GPIO_InitStruct);
561
562     /* USER CODE BEGIN MX_GPIO_Init_2 */
563     /* USER CODE END MX_GPIO_Init_2 */
564 }
565
566     /* USER CODE BEGIN 4 */
567
568     /* USER CODE END 4 */
569
570 /**
571 * @brief Period elapsed callback in non blocking mode
572 * @note This function is called when TIM1 interrupt took place, inside
573 * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment
574 * a global variable "uwTick" used as application time base.
575 * @param htim : TIM handle
576 * @retval None
577 */
578 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
579 {
580     /* USER CODE BEGIN Callback 0 */
581
582     /* USER CODE END Callback 0 */
583     if (htim->Instance == TIM1) {
584         HAL_IncTick();
585     }
586     /* USER CODE BEGIN Callback 1 */
587
588     /* USER CODE END Callback 1 */
589 }
590
591 /**
592 * @brief This function is executed in case of error occurrence.
593 * @retval None
594 */

```

```

595 void Error_Handler(void)
596 {
597     /* USER CODE BEGIN Error_Handler_Debug */
598     /* User can add his own implementation to report the HAL error return state */
599     __disable_irq();
600     while (1)
601     {
602     }
603     /* USER CODE END Error_Handler_Debug */
604 }
605
606 #ifdef USE_FULL_ASSERT
607 /**
608     * @brief Reports the name of the source file and the source line number
609     *        where the assert_param error has occurred.
610     * @param file: pointer to the source file name
611     * @param line: assert_param error line source number
612     * @retval None
613 */
614 void assert_failed(uint8_t *file, uint32_t line)
615 {
616     /* USER CODE BEGIN 6 */
617     /* User can add his own implementation to report the file name and line number,
618      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
619     /* USER CODE END 6 */
620 }
621 #endif /* USE_FULL_ASSERT */

```

## B.5. LoRa

### Codi de la llibreria

```

1 /*
2  * LoRaFi.cpp version 1.3.0
3  * https://github.com/LoRaFi/LoRaFi/tree/master/src
4  * Created on: May 13, 2017
5  * Authors: KHUDHUR ABDULLAH ALFARHAN and Dr. Ammar Zakaria
6  * Email: Qudoren@gmail.com , ammarzakaria@unimap.edu.my
7 *
8  * This library licensed under GNU General Public License (GPL)
9  * and supported by Centre of Excellence for Advanced Sensor Technology (CEASTech)
10 * https://ceastech.com/
11 * Universiti Malaysia Perlis (UniMAP)
12 * For more information about LoRaFi please visit the following links
13 * www.lorafi.tk
14 * www.loraficeastech.wixsite.com/lorafi
15 * LoRaFi forum:
16 * www.loraficeastech.wixsite.com/lorafi/forum
17 *
18 * This library is free software; you can redistribute it and/or
19 * modify it under the terms of the GNU Lesser General Public
20 * License as published by the Free Software Foundation; either
21 * version 2.1 of the License, or (at your option) any later version.
22 *
23 * This library is distributed in the hope that it will be useful,
24 * but WITHOUT ANY WARRANTY; without even the implied warranty of
25 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
26 * Lesser General Public License for more details.
27 */
28
29 #include "LoRaFi.h"

```

```

30 #include <stdio.h>
31 #include <stdlib.h>
32 #include "stm32f4xx_hal.h"
33
34
35 LoRaFi::LoRaFi()
36 {
37 }
38
39 //initialize LoRa module
40 void LoRaFi::begin(SPI_HandleTypeDef *hspi1, GPIO_TypeDef *cs_port, uint16_t cs_pin,
41                     GPIO_TypeDef *rst_port, uint16_t rst_pin, GPIO_TypeDef *rx_sw_port, uint16_t rx_sw_pin
42                     , GPIO_TypeDef *tx_sw_port, uint16_t tx_sw_pin, GPIO_TypeDef *dio0_port, uint16_t
43                     dio0_pin) {
44
45     hspi = hspi1;
46     //SPI.beginTransaction(SPISettings(1000000, MSBFIRST, SPI_MODE0));
47     //Store pins numbers in array to be used in other functions
48     LoRaPins[0] = cs_pin;
49     LoRaPins[1] = rst_pin;
50     LoRaPins[2] = rx_sw_pin;
51     LoRaPins[3] = tx_sw_pin;
52     LoRaPins[4] = dio0_pin;
53
54     LoRaPorts[0] = cs_port;
55     LoRaPorts[1] = rst_port;
56     LoRaPorts[2] = rx_sw_port;
57     LoRaPorts[3] = tx_sw_port;
58     LoRaPorts[4] = dio0_port;
59
60     //Reset LoRa chip
61     HAL_GPIO_WritePin(rst_port, rst_pin, GPIO_PIN_SET);
62     HAL_Delay(1);
63     HAL_GPIO_WritePin(rst_port, rst_pin, GPIO_PIN_RESET);
64     HAL_Delay(10);
65
66     // Set Chip select pin high to stop communication
67     HAL_GPIO_WritePin(cs_port, cs_pin, GPIO_PIN_SET);
68     HAL_Delay(10);
69
70     Mode(LORA);
71     HAL_Delay(10);
72     CRC_lora();
73     FrequencyHopping();
74     SpreadingFactor(12);
75 }
76
77
78
79
80 //Set LoRa Mode (SLEEP, LORA, STANDBY, TX, RX, RX_CONTINUOUS), FSK/OOK not available in
81 //this library
82 void LoRaFi::Mode(uint8_t mode){
83
84     if(mode == LORA)
85     {
86         Write_Register(0x01, SLEEP);                                // enter sleep mood to
87         select other modes

```

```

86             Write_Register(0x01, LORA);                                //enter user selected
87             mode
88         }
89     else { Write_Register(0x01, LORA | STANDBY);    //Wake up LoRa module
90 }
91 }
92
93
94
95
96 //Sleep Mode
97 void LoRaFi::sleep(void)
98 {
99     Mode();
100 }
101
102
103
104 // Idle mode
105 void LoRaFi::idle(void)
106 {
107     Mode(STANDBY);
108 }
109
110
111
112 //Stop LoRa module
113 void LoRaFi::end(void)
114 {
115     Mode(SLEEP);
116
117
118 // Set Chip select pin high to stop communication
119 HAL_GPIO_WritePin(LoRaPorts[0], LoRaPins[0], GPIO_PIN_SET);
120 HAL_GPIO_WritePin(LoRaPorts[2], LoRaPins[2], GPIO_PIN_SET);
121 HAL_GPIO_WritePin(LoRaPorts[3], LoRaPins[3], GPIO_PIN_RESET);
122
123 }
124
125
126
127
128 //Reset LoRa module, after reset, the setting will return to default and it should be
129 //configured
130 void LoRaFi::reset(void)
131 {
132     //Reset LoRa chip
133     HAL_GPIO_WritePin(LoRaPorts[1], LoRaPins[1], GPIO_PIN_SET);
134     HAL_Delay(1);
135     HAL_GPIO_WritePin(LoRaPorts[1], LoRaPins[1], GPIO_PIN_RESET);
136     HAL_Delay(10);
137
138
139
140 //Get RSSI value in dBm
141 int16_t LoRaFi::Rssi(void)
142 {
143     return -139 + Read_Register(0x1b);
144 }
```

```

145
146
147
148 // Get the RSSI of the last received packet
149 int16_t LoRaFi::PacketRssi(void)
150 {
151     if((SNR()) >= 0)
152     {
153         return -139 + ((int8_t)Read_Register(0x1A));
154     }
155     else
156     {
157         return -139 + ((int8_t)Read_Register(0x1A)) + SNR();
158     }
159 }
160
161
162
163 //Get SNR value
164 float LoRaFi::SNR(void)
165 {
166     return ((int8_t)Read_Register(0x19)) * 0.25;
167 }
168
169
170
171
172 // Set transmitting power ==> NORMAL_POWER = 13dBm ==> MAX_POWER = 20dBm
173 void LoRaFi::TXpower(uint8_t pow)
174 {
175     //Enter standby mode to set transmitting power
176     Mode(STANDBY);
177
178     Write_Register(POWER_REGISTER, pow);
179 }
180
181
182
183
184 // set channel frequency
185 void LoRaFi::ChannelFrequency(long frequency)
186 {
187     Mode(SLEEP);
188
189     uint64_t freq = ((uint64_t)frequency << 19) / 32000000;
190
191     Write_Register(0x06, (uint8_t)(freq >> 16));
192     Write_Register(0x07, (uint8_t)(freq >> 8));
193     Write_Register(0x08, (uint8_t)(freq >> 0));
194
195     Mode(STANDBY);
196 }
197
198
199
200
201 //Set Spreading Factor, the supported Spreading Factor values from 6 to 12, default to 7
202 void LoRaFi::SpreadingFactor(uint8_t SF)
203 {
204     Mode(STANDBY);
205 }
```

```

206 //To make the value of SF from 6 to 12
207     if(SF < 6) {SF = 6;}
208     if(SF > 12) {SF = 12;}
209
210     if (SF == 6)
211     {
212         Write_Register(0x31, 0xc5);
213         Write_Register(0x37, 0x0c);
214     }
215     else {
216         Write_Register(0x31, 0xc3);
217         Write_Register(0x37, 0xa0);
218     }
219
220     Write_Register(0x1E, (Read_Register(0x1E) & 0x0f) | ((SF << 4) & 0xf0));
221 }
222
223
224
225
226
227 //Set the signal bandwidth of the radio "supported values are 125E3, 250E3, 500E3"
228 //default value is 125E3
229 void LoRaFi::Bandwidth(long bw)
230 {
231     uint8_t set_bw;
232     Mode(STANDBY);
233
234     if (bw <= 125E3)
235     { set_bw = 0; }
236
237     else if (bw <= 250E3)
238     { set_bw = 1; }
239
240     else { set_bw = 2; }
241
242     Write_Register(0x1d, (Read_Register(0x1d) & 0x0f) | (set_bw << 6));
243 }
244
245
246
247 //Set the coding rate of LoRa radio "supported values are from 5 to 8" Default value is
248 //5
249 void LoRaFi::CodingRate(uint8_t C_Rate)
250 {
251     Mode(STANDBY);
252
253     if (C_Rate < 5) { C_Rate = 5; }
254
255     else if (C_Rate > 8) { C_Rate = 8; }
256
257     uint8_t CR = C_Rate - 4;
258
259     Write_Register(0x1d, (Read_Register(0x1d) & 0xf1) | (CR << 3));
260 }
261
262
263 //Set the preamble length of LoRa radio " supported values from 6 to 65535 default value
264 //is 8

```

```

264 void LoRaFi::PreambleLength(uint16_t length)
265 {
266     Mode(STANDBY);
267
268     if (length < 6) {length = 6;}
269
270     Write_Register(0x20, (uint8_t)(length >> 8));
271     Write_Register(0x21, (uint8_t)(length >> 0));
272
273 }
274
275
276
277 //Set the sync word of LoRa radio it is one-byte "Default is 0x34"
278 void LoRaFi::SyncWord(uint8_t sw)
279 {
280     Mode(STANDBY);
281
282     Write_Register(0x39, sw);
283 }
284
285
286
287 // Set the frequency hopping ON/OFF
288 void LoRaFi::FrequencyHopping(uint8_t FH)
289 {
290     if(FH == ON)
291     {
292         Write_Register(0x24, 0x11);
293     }
294     if(FH == OFF)
295     {
296         Write_Register(0x24, 0x00);
297     }
298 }
299
300
301 // Set CRC checksum ON/OFF
302 void LoRaFi::CRC_lora(uint8_t crc)
303 {
304     if(crc == ON)
305     {
306         Write_Register(0x1d, (Read_Register(0x1d) & 0xff) | (0x1 << 1));
307     }
308     if(crc == OFF)
309     {
310         Write_Register(0x1d, (Read_Register(0x1d) & 0xff) | (0x0 << 1));
311     }
312 }
313
314
315
316
317 // Set Low Data Rate Optimize ON/OFF
318 void LoRaFi::LDRoptimize(uint8_t ldr)
319 {
320     if(ldr == ON)
321     {
322         Write_Register(0x1d, (Read_Register(0x1d) & 0xff) | (0x1));
323     }
324     if(ldr == OFF)

```

```

325         {
326             Write_Register(0x1d, (Read_Register(0x1d) & 0xff) | (0x0));
327         }
328     }
329
330
331
332
333 //Set Receiving Interrupt
334 void LoRaFi::ReceivingInterrupt(void (*userFunction)(void))
335 {
336     if(!interrupt)
337     {
338         interrupt = true;
339         SetInterrupt();
340         //attachInterrupt(digitalPinToInterruption(LoRaPins[4]), userFunction,
341         //RISING);
342     }
343 }
344 }
345
346
347
348
349 //Cancel receiving interrupt
350 void LoRaFi::CancelInterrupt(void)
351 {
352     if(interrupt)
353     {
354         //detachInterrupt(LoRaPins[4]);
355         interrupt = false;
356     }
357 }
358
359
360
361
362 //set receiving interrupt registers
363 void LoRaFi::SetInterrupt()
364 {
365     HAL_GPIO_WritePin(LoRaPorts[0], LoRaPins[0], GPIO_PIN_SET);
366     //Switch LoRa to standby
367     Mode(RX_CONTINUOUS);
368
369     //Load FIFO tx pointer
370     uint8_t value = Read_Register(0x0E);
371     Write_Register(0x0D, value);
372
373     // Receive Mode
374     HAL_GPIO_WritePin(LoRaPorts[3], LoRaPins[3], GPIO_PIN_RESET);
375     HAL_GPIO_WritePin(LoRaPorts[2], LoRaPins[2], GPIO_PIN_SET);
376
377     //IRD mask
378     Write_Register(0x11, 0x87);
379
380     //clear IRQ
381     Write_Register(0x12, 0xFF);
382 }
383
384

```

```

385
386
387 //Write data to specific register
388 void LoRaFi::Write_Register(uint8_t address, uint8_t data)
389 {
390     // Set Chip select pin low to start communication with LoRa module
391     HAL_GPIO_WritePin(LoRaPorts[0], LoRaPins[0], GPIO_PIN_RESET);
392
393     uint8_t buf[2] = { (address | 0x80), data, };
394     HAL_SPI_Transmit(hspi, buf, 2, HAL_MAX_DELAY);
395
396     // Set CS pin high to stop communication
397     HAL_GPIO_WritePin(LoRaPorts[0], LoRaPins[0], GPIO_PIN_SET);
398 }
399
400
401
402 //Read data from specific register
403 uint8_t LoRaFi::Read_Register(uint8_t address)
404 {
405     uint8_t RegData;
406
407     // Set Chip select pin low to start communication with LoRa module
408     HAL_GPIO_WritePin(LoRaPorts[0], LoRaPins[0], GPIO_PIN_RESET);
409
410     // Send register address
411
412     HAL_SPI_Transmit(hspi, &address, 1, HAL_MAX_DELAY);
413     HAL_SPI_Receive(hspi, &RegData, 1, HAL_MAX_DELAY);
414
415     // Set Chip select pin high to stop communication with LoRa module
416     HAL_GPIO_WritePin(LoRaPorts[0], LoRaPins[0], GPIO_PIN_SET);
417
418     return RegData;
419 }
420
421
422
423
424
425 //send package
426 void LoRaFi::SendPackage(char *Package, uint8_t packageLength)
427 {
428     uint16_t i;
429     uint8_t Value;
430
431     //Switch LoRa to standby mode
432     Mode(STANDBY);
433
434     //Set payload length
435     Write_Register(0x22, packageLength);
436
437     //Load FIFO tx pointer
438     Value = Read_Register(0x0E);      //get FIFO address pointer value
439     Write_Register(0x0D, Value);      //set SPI interface address pointer in FIFO data
440     buffer.
441
442     //Transmit Mode
443     HAL_GPIO_WritePin(LoRaPorts[3], LoRaPins[3], GPIO_PIN_SET);
444     HAL_GPIO_WritePin(LoRaPorts[2], LoRaPins[2], GPIO_PIN_RESET);
445     //IRQ mask

```

```

445     Write_Register(0x11, 0x87);
446
447     //IRQ flag
448     Write_Register(0x12, 0xFF);
449
450     //Write payload in to the FIFO
451     for (i = 0; i < packageLength; i++)
452     {
453         Write_Register(0x80, *Package);
454         HAL_Delay(100);
455         Package++;
456     }
457
458     //Close TX LoRa
459     Write_Register(0x01, 0x82);
460     Write_Register(0x01, 0x83);
461
462     //Wait until tx done
463     do
464     {
465         Value = Read_Register(0x12);
466
467         Value = Value & 0x08;
468
469     } while (Value == 0);
470
471     HAL_Delay(1);
472
473     //Check for interrupt
474     if (!interrupt)
475     {
476         //stop sending and receiving
477         HAL_GPIO_WritePin(LoRaPorts[3], LoRaPins[3], GPIO_PIN_RESET);
478         HAL_GPIO_WritePin(LoRaPorts[2], LoRaPins[2], GPIO_PIN_RESET);
479     }
480     else
481     {
482         // Receive Mode
483         HAL_GPIO_WritePin(LoRaPorts[3], LoRaPins[3], GPIO_PIN_RESET);
484         HAL_GPIO_WritePin(LoRaPorts[2], LoRaPins[2], GPIO_PIN_SET);
485     }
486
487 }
488
489
490
491
492 //Receive package
493 void LoRaFi::ReceivePackage(char *Package, uint8_t packageLength)
494 {
495     uint16_t i;
496     uint8_t LoRa_Interrupt;
497     uint8_t PackageLocation;
498
499     HAL_GPIO_WritePin(LoRaPorts[0], LoRaPins[0], GPIO_PIN_SET);
500
501     //check for interrupt
502     if (!interrupt)
503     {
504         SetInterrupt();
505     }

```

```

506
507
508     while (HAL_GPIO_ReadPin(LoRaPorts[4], LoRaPins[4]) == 0)
509     {
510         HAL_Delay(1);
511     }
512
513     //Get interrupt register
514     LoRa_Interrupt = Read_Register(0x12);
515
516     //Clear interrupt register
517     Write_Register(0x12, 0x60);
518
519     //Get package location
520     PackageLocation = Read_Register(0x10);
521
522     //Set SPI pointer to PackageLocation
523     Write_Register(0x0D, PackageLocation);
524
525     //Get message
526     for (i = 0; i < packageLength; i++)
527     {
528         *Package = Read_Register(0x00);
529         Package++;
530     }
531
532
533
534
535     //stop sending and receiving
536     HAL_GPIO_WritePin(LoRaPorts[3], LoRaPins[3], GPIO_PIN_RESET);
537     HAL_GPIO_WritePin(LoRaPorts[2], LoRaPins[2], GPIO_PIN_RESET);
538
539     //Check for interrupt
540     if(interrupt)
541     { SetInterrupt();}
542 }
543
544
545
546
547
548
549 //handling data for send
550
551
552 //handling constant char
553 void LoRaFi::Send(const char package[])
554 {
555     SendPackage((char *) package, strlen(package));
556 }
557
558
559 //handling char
560 void LoRaFi::Send(char package)
561 {
562     uint8_t len = sizeof(package);
563     SendPackage( &package, len);
564 }
565
566 //handling unsigned char

```

```

567 void LoRaFi::Send(unsigned char package)
568 {
569     uint8_t len = sizeof(package);
570     SendPackage((char*) &package, len);
571 }
572
573
574
575
576
577 //handling int
578 void LoRaFi::Send(int package)
579 {
580     char buffer[10];
581     itoa(package, buffer, 10);
582     SendPackage(buffer, 10);
583 }
584
585
586 //Handling unsigned int
587 void LoRaFi::Send(unsigned int package)
588 {
589     uint8_t len = sizeof(package);
590     char buffer[10];
591     utoa(package, buffer, 10);
592     SendPackage(buffer, 10);
593 }
594
595
596
597
598 //Handling long
599 void LoRaFi::Send(long l)
600 {
601     long_conv.union_long = l;
602     SendPackage(long_conv.union_char, 4);
603 }
604
605
606
607 //Handling unsigned long
608 void LoRaFi::Send(unsigned long l)
609 {
610     Ulong_conv.union_Ulong = l;
611     SendPackage(Ulong_conv.union_char, 4);
612 }
613
614
615
616
617
618
619
620 //Handling received data
621
622
623 //Handling received int
624 int LoRaFi::ReceiveInt(void)
625 {
626     char receivedChar[10];
627     //Get received data in char type

```

```

628     ReceivePackage(receivedChar,10);
629
630     //convert char data to int
631     return atoi(receivedChar);
632 }
633
634
635 //Handling received unsigned int
636 unsigned int LoRaFi::ReceiveUint(void)
637 {
638     char receivedUint[10];
639     //Get received data in char type
640     ReceivePackage(receivedUint,10);
641
642     //convert char data to unsigned int and return it
643     return strtoul(receivedUint,NULL,10);
644 }
645
646
647 //Handling received long
648 long LoRaFi::ReceiveLong(void)
649 {
650     char receivedLong[4];
651     //Get received data in char type
652     ReceivePackage(receivedLong,4);
653
654     long_conv.union_char[0] = receivedLong[0];
655     long_conv.union_char[1] = receivedLong[1];
656     long_conv.union_char[2] = receivedLong[2];
657     long_conv.union_char[3] = receivedLong[3];
658
659     return long_conv.union_long;
660 }
661
662
663
664 //Handling received unsigned long
665 unsigned long LoRaFi::ReceiveUlong(void)
666 {
667     char receivedUlong[4];
668     //Get received data in char type
669     ReceivePackage(receivedUlong,4);
670
671     Ulong_conv.union_char[0] = receivedUlong[0];
672     Ulong_conv.union_char[1] = receivedUlong[1];
673     Ulong_conv.union_char[2] = receivedUlong[2];
674     Ulong_conv.union_char[3] = receivedUlong[3];
675
676     return Ulong_conv.union_Ulong;
677 }
678
679
680 //Handling received double
681 double LoRaFi::ReceiveDouble(void)
682 {
683     char receivedDouble[20];
684     //Get received data in char type
685     ReceivePackage(receivedDouble,20);
686
687     return atof(receivedDouble);
688 }
```

```

689
690
691
692 //Receive char
693 char LoRaFi::ReceiveChar(void)
694 {
695     char receivedchar[1];
696
697     ReceivePackage(receivedchar,1);
698
699     return receivedchar[0];
700 }
701
702
703
704 //Receive unsigned char
705 unsigned char LoRaFi::ReceiveUchar(void)
706 {
707     char receivedUchar[1];
708
709     ReceivePackage(receivedUchar,1);
710
711     return (unsigned char) receivedUchar[0];
712 }
```

### Codi de la prova

```

1  /* USER CODE BEGIN Header */
2 /**
3  ****
4  * @file           : main.c
5  * @brief          : Main program body
6  ****
7  * @attention
8  *
9  * Copyright (c) 2024 STMicroelectronics.
10 * All rights reserved.
11 *
12 * This software is licensed under terms that can be found in the LICENSE file
13 * in the root directory of this software component.
14 * If no LICENSE file comes with this software, it is provided AS-IS.
15 *
16 ****
17 */
18 /* USER CODE END Header */
19 /* Includes -----*/
20 #include "main.h"
21 #include "usb_device.h"
22
23 /* Private includes -----*/
24 /* USER CODE BEGIN Includes */
25 #include "LoRaFi.cpp"
26 /* USER CODE END Includes */
27
28 /* Private typedef -----*/
29 /* USER CODE BEGIN PTD */
30
31 /* USER CODE END PTD */
32
33 /* Private define -----*/
34 /* USER CODE BEGIN PD */
```

```

35  /* USER CODE END PD */
36
37  /* Private macro -----
38  /* USER CODE BEGIN PM */
39
40  /* USER CODE END PM */
41
42  /* Private variables -----
43  ADC_HandleTypeDef hadc1;
44
45  I2C_HandleTypeDef hi2c1;
46  I2C_HandleTypeDef hi2c3;
47
48  SPI_HandleTypeDef hspi1;
49  SPI_HandleTypeDef hspi2;
50
51  TIM_HandleTypeDef htim4;
52
53  UART_HandleTypeDef huart2;
54
55  /* USER CODE BEGIN PV */
56
57  /* USER CODE END PV */
58
59  /* Private function prototypes -----
60  void SystemClock_Config(void);
61  static void MX_GPIO_Init(void);
62  static void MX_ADC1_Init(void);
63  static void MX_I2C1_Init(void);
64  static void MX_I2C3_Init(void);
65  static void MX_SPI1_Init(void);
66  static void MX_SPI2_Init(void);
67  static void MX_USART2_UART_Init(void);
68  static void MX_TIM4_Init(void);
69
70  /* USER CODE BEGIN PFP */
71
72  /* USER CODE END PFP */
73
74  /* Private user code -----
75  /* USER CODE BEGIN 0 */
76
77  /* USER CODE END 0 */
78
79  /**
80   * @brief  The application entry point.
81   * @retval int
82   */
83  int main(void)
84  {
85
86  /* USER CODE BEGIN 1 */
87
88  /* USER CODE END 1 */
89
90  /* MCU Configuration -----
91
92  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
93  HAL_Init();
94
95  /* USER CODE BEGIN Init */

```

```

96
97     /* USER CODE END Init */
98
99     /* Configure the system clock */
100    SystemClock_Config();
101
102   /* USER CODE BEGIN SysInit */
103
104   /* USER CODE END SysInit */
105
106  /* Initialize all configured peripherals */
107  MX_GPIO_Init();
108  MX_ADC1_Init();
109  MX_I2C1_Init();
110  MX_I2C3_Init();
111  MX_SPI1_Init();
112  MX_SPI2_Init();
113  MX_USART2_UART_Init();
114  MX_TIM4_Init();
115  MX_USB_DEVICE_Init();
116  /* USER CODE BEGIN 2 */
117  LoRaFi LoRaFi;
118  LoRaFi.begin(&hspi2, LORA_NSS_GPIO_Port, LORA_NSS_Pin, LORA_RST_GPIO_Port, LORA_RST_Pin
119      , LORA_RX_GPIO_Port, LORA_RX_Pin, LORA_TX_GPIO_Port, LORA_TX_Pin, LORA_O_GPIO_Port,
120      LORA_O_Pin);
121
122  /* USER CODE END 2 */
123
124  /* Infinite loop */
125  /* USER CODE BEGIN WHILE */
126  while (1)
127  {
128
129      LoRaFi.Send("Hello World");
130
131      int len = 11;
132
133      //Receive message
134      uint8_t data[40];
135      LoRaFi.ReceivePackage(data,len);
136
137      //Print received message
138
139      CDC_Transmit_FS(data, strlen((char*) data));
140      CDC_Transmit_FS((uint8_t*) "\n",2);
141
142      delay(1000);
143
144  /* USER CODE END WHILE */
145
146  /* USER CODE BEGIN 3 */
147  }
148
149 /**
150 * @brief System Clock Configuration
151 * @retval None
152 */
153 void SystemClock_Config(void)
154 {

```

```

155     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
156     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
157
158     /* Configure the main internal regulator output voltage
159     */
160     __HAL_RCC_PWR_CLK_ENABLE();
161     __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);
162
163     /* Initializes the RCC Oscillators according to the specified parameters
164     * in the RCC_OscInitTypeDef structure.
165     */
166     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_HSE;
167     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
168     RCC_OscInitStruct.HSISite = RCC_HSI_ON;
169     RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
170     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
171     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
172     RCC_OscInitStruct.PLL.PLLM = 4;
173     RCC_OscInitStruct.PLL.PLLN = 72;
174     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
175     RCC_OscInitStruct.PLL.PLLQ = 3;
176     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
177     {
178         Error_Handler();
179     }
180
181     /* Initializes the CPU, AHB and APB buses clocks
182     */
183     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
184                     |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
185     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
186     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
187     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
188     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
189
190     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
191     {
192         Error_Handler();
193     }
194 }
195
196 /**
197 * @brief ADC1 Initialization Function
198 * @param None
199 * @retval None
200 */
201 static void MX_ADC1_Init(void)
202 {
203
204     /* USER CODE BEGIN ADC1_Init_0 */
205
206     /* USER CODE END ADC1_Init_0 */
207
208     ADC_ChannelConfTypeDef sConfig = {0};
209
210     /* USER CODE BEGIN ADC1_Init_1 */
211
212     /* USER CODE END ADC1_Init_1 */
213
214     /* Configure the global features of the ADC (Clock, Resolution, Data Alignment and
215     * number of conversion)

```

```

215  /*
216   hadc1.Instance = ADC1;
217   hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
218   hadc1.Init.Resolution = ADC_RESOLUTION_12B;
219   hadc1.Init.ScanConvMode = DISABLE;
220   hadc1.Init.ContinuousConvMode = ENABLE;
221   hadc1.Init.DiscontinuousConvMode = DISABLE;
222   hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
223   hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
224   hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
225   hadc1.Init.NbrOfConversion = 1;
226   hadc1.Init.DMAContinuousRequests = DISABLE;
227   hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
228   if (HAL_ADC_Init(&hadc1) != HAL_OK)
229   {
230     Error_Handler();
231   }
232
233   /** Configure for the selected ADC regular channel its corresponding rank in the
234   sequencer and its sample time.
235   */
236   sConfig.Channel = ADC_CHANNEL_8;
237   sConfig.Rank = 1;
238   sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
239   if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
240   {
241     Error_Handler();
242   }
243   /* USER CODE BEGIN ADC1_Init_2 */
244   /* USER CODE END ADC1_Init_2 */
245
246 }
247
248 /**
249 * @brief I2C1 Initialization Function
250 * @param None
251 * @retval None
252 */
253 static void MX_I2C1_Init(void)
254 {
255
256   /* USER CODE BEGIN I2C1_Init_0 */
257
258   /* USER CODE END I2C1_Init_0 */
259
260   /* USER CODE BEGIN I2C1_Init_1 */
261
262   /* USER CODE END I2C1_Init_1 */
263   hi2c1.Instance = I2C1;
264   hi2c1.Init.ClockSpeed = 100000;
265   hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
266   hi2c1.Init.OwnAddress1 = 0;
267   hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
268   hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
269   hi2c1.Init.OwnAddress2 = 0;
270   hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
271   hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
272   if (HAL_I2C_Init(&hi2c1) != HAL_OK)
273   {
274     Error_Handler();

```

```

275     }
276     /* USER CODE BEGIN I2C1_Init_2 */
277
278     /* USER CODE END I2C1_Init_2 */
279
280 }
281
282 /**
283 * @brief I2C3 Initialization Function
284 * @param None
285 * @retval None
286 */
287 static void MX_I2C3_Init(void)
288 {
289
290     /* USER CODE BEGIN I2C3_Init_0 */
291
292     /* USER CODE END I2C3_Init_0 */
293
294     /* USER CODE BEGIN I2C3_Init_1 */
295
296     /* USER CODE END I2C3_Init_1 */
297     hi2c3.Instance = I2C3;
298     hi2c3.Init.ClockSpeed = 100000;
299     hi2c3.Init.DutyCycle = I2C_DUTYCYCLE_2;
300     hi2c3.Init.OwnAddress1 = 0;
301     hi2c3.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
302     hi2c3.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
303     hi2c3.Init.OwnAddress2 = 0;
304     hi2c3.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
305     hi2c3.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
306     if (HAL_I2C_Init(&hi2c3) != HAL_OK)
307     {
308         Error_Handler();
309     }
310     /* USER CODE BEGIN I2C3_Init_2 */
311
312     /* USER CODE END I2C3_Init_2 */
313
314 }
315
316 /**
317 * @brief SPI1 Initialization Function
318 * @param None
319 * @retval None
320 */
321 static void MX_SPI1_Init(void)
322 {
323
324     /* USER CODE BEGIN SPI1_Init_0 */
325
326     /* USER CODE END SPI1_Init_0 */
327
328     /* USER CODE BEGIN SPI1_Init_1 */
329
330     /* USER CODE END SPI1_Init_1 */
331     /* SPI1 parameter configuration*/
332     hspi1.Instance = SPI1;
333     hspi1.Init.Mode = SPI_MODE_MASTER;
334     hspi1.Init.Direction = SPI_DIRECTION_2LINES;
335     hspi1.Init.DataSize = SPI_DATASIZE_8BIT;

```

```

336     hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
337     hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
338     hspi1.Init.NSS = SPI_NSS_SOFT;
339     hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
340     hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
341     hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
342     hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
343     hspi1.Init.CRCPolynomial = 10;
344     if (HAL_SPI_Init(&hspi1) != HAL_OK)
345     {
346         Error_Handler();
347     }
348     /* USER CODE BEGIN SPI1_Init 2 */
349
350     /* USER CODE END SPI1_Init 2 */
351 }
352 /**
353 * @brief SPI2 Initialization Function
354 * @param None
355 * @retval None
356 */
357 static void MX_SPI2_Init(void)
358 {
359     /* USER CODE BEGIN SPI2_Init 0 */
360
361     /* USER CODE END SPI2_Init 0 */
362
363     /* USER CODE BEGIN SPI2_Init 1 */
364
365     /* USER CODE END SPI2_Init 1 */
366     /* SPI2 parameter configuration*/
367     hspi2.Instance = SPI2;
368     hspi2.Init.Mode = SPI_MODE_MASTER;
369     hspi2.Init.Direction = SPI_DIRECTION_2LINES;
370     hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
371     hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
372     hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
373     hspi2.Init.NSS = SPI_NSS_SOFT;
374     hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
375     hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
376     hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
377     hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
378     hspi2.Init.CRCPolynomial = 10;
379     if (HAL_SPI_Init(&hspi2) != HAL_OK)
380     {
381         Error_Handler();
382     }
383     /* USER CODE BEGIN SPI2_Init 2 */
384
385     /* USER CODE END SPI2_Init 2 */
386 }
387 /**
388 * @brief TIM4 Initialization Function
389 * @param None
390 * @retval None
391 */
392 /**
393 * @brief TIM4 Initialization Function
394 * @param None
395 * @retval None
396 */

```

```

397 static void MX_TIM4_Init(void)
398 {
399     /* USER CODE BEGIN TIM4_Init_0 */
400
401     /* USER CODE END TIM4_Init_0 */
402
403     TIM_ClockConfigTypeDef sClockSourceConfig = {0};
404     TIM_MasterConfigTypeDef sMasterConfig = {0};
405     TIM_OC_InitTypeDef sConfigOC = {0};
406
407     /* USER CODE BEGIN TIM4_Init_1 */
408
409     /* USER CODE END TIM4_Init_1 */
410     htim4.Instance = TIM4;
411     htim4.Init.Prescaler = 31;
412     htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
413     htim4.Init.Period = 254;
414     htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
415     htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
416     if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
417     {
418         Error_Handler();
419     }
420
421     sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
422     if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
423     {
424         Error_Handler();
425     }
426     if (HAL_TIM_PWM_Init(&htim4) != HAL_OK)
427     {
428         Error_Handler();
429     }
430     sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
431     sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
432     if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL_OK)
433     {
434         Error_Handler();
435     }
436     sConfigOC.OCMode = TIM_OCMODE_PWM1;
437     sConfigOC.Pulse = 0;
438     sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
439     sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
440     if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_3) != HAL_OK)
441     {
442         Error_Handler();
443     }
444     /* USER CODE BEGIN TIM4_Init_2 */
445
446     /* USER CODE END TIM4_Init_2 */
447     HAL_TIM_MspPostInit(&htim4);
448
449 }
450
451 /**
452 * @brief USART2 Initialization Function
453 * @param None
454 * @retval None
455 */
456 static void MX_USART2_UART_Init(void)
457 {

```

```

458
459 /* USER CODE BEGIN USART2_Init_0 */
460
461 /* USER CODE END USART2_Init_0 */
462
463 /* USER CODE BEGIN USART2_Init_1 */
464
465 /* USER CODE END USART2_Init_1 */
466 huart2.Instance = USART2;
467 huart2.Init.BaudRate = 9600;
468 huart2.Init.WordLength = UART_WORDLENGTH_8B;
469 huart2.Init.StopBits = UART_STOPBITS_1;
470 huart2.Init.Parity = UART_PARITY_NONE;
471 huart2.Init.Mode = UART_MODE_TX_RX;
472 huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
473 huart2.Init.OverSampling = UART_OVERSAMPLING_16;
474 if (HAL_UART_Init(&huart2) != HAL_OK)
475 {
476     Error_Handler();
477 }
478 /* USER CODE BEGIN USART2_Init_2 */
479
480 /* USER CODE END USART2_Init_2 */
481 }
482 }
483 /**
484 * @brief GPIO Initialization Function
485 * @param None
486 * @retval None
487 */
488
489 static void MX_GPIO_Init(void)
490 {
491     GPIO_InitTypeDef GPIO_InitStruct = {0};
492 /* USER CODE BEGIN MX_GPIO_Init_1 */
493 /* USER CODE END MX_GPIO_Init_1 */
494
495     /* GPIO Ports Clock Enable */
496     __HAL_RCC_GPIOC_CLK_ENABLE();
497     __HAL_RCC_GPIOH_CLK_ENABLE();
498     __HAL_RCC_GPIOA_CLK_ENABLE();
499     __HAL_RCC_GPIOB_CLK_ENABLE();
500     __HAL_RCC_GPIOD_CLK_ENABLE();
501
502     /*Configure GPIO pin Output Level */
503     HAL_GPIO_WritePin(GPIOC, ENABLE_1_Pin|IGNITE_1_Pin|LEDO_Pin|LED1_Pin
504                                         |LED2_Pin|LED3_Pin|FLASH_WP_Pin|SERVO1_Pin
505                                         |SERVO2_Pin, GPIO_PIN_RESET);
506
507     /*Configure GPIO pin Output Level */
508     HAL_GPIO_WritePin(FLASH_NSS_GPIO_Port, FLASH_NSS_Pin, GPIO_PIN_RESET);
509
510     /*Configure GPIO pin Output Level */
511     HAL_GPIO_WritePin(GPIOB, LORA_NSS_Pin|INT_IMU_Pin, GPIO_PIN_RESET);
512
513     /*Configure GPIO pin Output Level */
514     HAL_GPIO_WritePin(GPIOD, LORA_RST_Pin|IGNITE_2_Pin|ENABLE_2_Pin|LORA_TX_Pin
515                                         |LORA_RX_Pin, GPIO_PIN_RESET);
516
517     /*Configure GPIO pins : ENABLE_1_Pin IGNITE_1_Pin LEDO_Pin LED1_Pin
518                                         LED2_Pin LED3_Pin FLASH_WP_Pin SERVO1_Pin

```

```

519             SERVO2_Pin */
520     GPIO_InitStruct.Pin = ENABLE_1_Pin|IGNITE_1_Pin|LEDO_Pin|LED1_Pin
521                 |LED2_Pin|LED3_Pin|FLASH_WP_Pin|SERVO1_Pin
522                 |SERVO2_Pin;
523     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
524     GPIO_InitStruct.Pull = GPIO_NOPULL;
525     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
526     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
527
528 /*Configure GPIO pin : FLASH_NSS_Pin */
529     GPIO_InitStruct.Pin = FLASH_NSS_Pin;
530     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
531     GPIO_InitStruct.Pull = GPIO_NOPULL;
532     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
533     HAL_GPIO_Init(FLASH_NSS_GPIO_Port, &GPIO_InitStruct);
534
535 /*Configure GPIO pins : LORA_NSS_Pin INT_IMU_Pin */
536     GPIO_InitStruct.Pin = LORA_NSS_Pin|INT_IMU_Pin;
537     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
538     GPIO_InitStruct.Pull = GPIO_NOPULL;
539     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
540     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
541
542 /*Configure GPIO pins : LORA_RST_Pin IGNITE_2_Pin ENABLE_2_Pin LORA_TX_Pin
543                 LORA_RX_Pin */
544     GPIO_InitStruct.Pin = LORA_RST_Pin|IGNITE_2_Pin|ENABLE_2_Pin|LORA_TX_Pin
545                 |LORA_RX_Pin;
546     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
547     GPIO_InitStruct.Pull = GPIO_NOPULL;
548     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
549     HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
550
551 /*Configure GPIO pin : LORA_O_Pin */
552     GPIO_InitStruct.Pin = LORA_O_Pin;
553     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
554     GPIO_InitStruct.Pull = GPIO_NOPULL;
555     HAL_GPIO_Init(LORA_O_GPIO_Port, &GPIO_InitStruct);
556
557 /*Configure GPIO pin : SW_KEY_Pin */
558     GPIO_InitStruct.Pin = SW_KEY_Pin;
559     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
560     GPIO_InitStruct.Pull = GPIO_PULLDOWN;
561     HAL_GPIO_Init(SW_KEY_GPIO_Port, &GPIO_InitStruct);
562
563 /* USER CODE BEGIN MX_GPIO_Init_2 */
564 /* USER CODE END MX_GPIO_Init_2 */
565 }
566
567 /* USER CODE BEGIN 4 */
568
569 /* USER CODE END 4 */
570
571 /**
572 * @brief Period elapsed callback in non blocking mode
573 * @note This function is called when TIM1 interrupt took place, inside
574 * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment
575 * a global variable "uwTick" used as application time base.
576 * @param htim : TIM handle
577 * @retval None
578 */
579 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)

```

```

580 {
581     /* USER CODE BEGIN Callback_0 */
582
583     /* USER CODE END Callback_0 */
584     if (htim->Instance == TIM1) {
585         HAL_IncTick();
586     }
587     /* USER CODE BEGIN Callback_1 */
588
589     /* USER CODE END Callback_1 */
590 }
591
592 /**
593 * @brief This function is executed in case of error occurrence.
594 * @retval None
595 */
596 void Error_Handler(void)
597 {
598     /* USER CODE BEGIN Error_Handler_Debug */
599     /* User can add his own implementation to report the HAL error return state */
600     __disable_irq();
601     while (1)
602     {
603     }
604     /* USER CODE END Error_Handler_Debug */
605 }
606
607 #ifdef USE_FULL_ASSERT
608 /**
609 * @brief Reports the name of the source file and the source line number
610 * where the assert_param error has occurred.
611 * @param file: pointer to the source file name
612 * @param line: assert_param error line source number
613 * @retval None
614 */
615 void assert_failed(uint8_t *file, uint32_t line)
616 {
617     /* USER CODE BEGIN 6 */
618     /* User can add his own implementation to report the file name and line number,
619      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
620     /* USER CODE END 6 */
621 }
622 #endif /* USE_FULL_ASSERT */

```

## Bibliografía

- [1] [EUROC | EUROPEAN ROCKETRY CHALLENGE](https://euroc.pt/). URL: <https://euroc.pt/> (visitado 04-05-2024).
- [2] Autodesk Fusion | Software de CAD 3D, CAM, CAE y PCB basado en la nube | Autodesk. URL: <https://www.autodesk.es/products/fusion-360/overview> (visitado 23-02-2024).
- [3] Software y herramientas de diseño de PCB para construir la próxima generación en electrónica | Altium. URL: <https://www.altium.com/es> (visitado 23-02-2024).
- [4] PCB Prototype & PCB Fabrication Manufacturer - JLCPCB. URL: [https://jlcpcb.com/?from=VGP2&gad\\_source=1&gclid=Cj0KCQiAoeGuBhCBARIzAGfKY7zIMKDapxa60Dw3\\_YhTCSHHOFQOF8n-Krf65gstwUmihFLp4IaVoCMaAns2EALw\\_wcB](https://jlcpcb.com/?from=VGP2&gad_source=1&gclid=Cj0KCQiAoeGuBhCBARIzAGfKY7zIMKDapxa60Dw3_YhTCSHHOFQOF8n-Krf65gstwUmihFLp4IaVoCMaAns2EALw_wcB) (visitado 23-02-2024).
- [5] STM32CubeIDE - Integrated Development Environment for STM32 - STMicroelectronics. URL: <https://www.st.com/en/development-tools/stm32cubeide.html> (visitado 23-02-2024).
- [6] Tablero KAN: tablero ágil - Jira. URL: <https://www.atlassian.com/software/jira> (visitado 23-02-2024).
- [7] Página principal - Google Drive. URL: <https://www.google.es/intl/ca/drive/> (visitado 23-02-2024).
- [8] GitHub: Let's build from here. GitHub. 2024. URL: <https://github.com/> (visitado 08-10-2024).
- [9] TeamGantt. URL: <https://app.teamgantt.com/> (visitado 02-03-2024).
- [10] Overleaf, Online LaTeX Editor. URL: <https://www.overleaf.com> (visitado 02-03-2024).
- [11] Sueldos | Indeed.com. URL: <https://es.indeed.com/career/salaries?from=gnav-title-webapp> (visitado 09-03-2024).
- [12] MS560702BA03-50 : MEAS Board Mount Pressure Sensors. TE Connectivity. URL: <https://www.te.com/usa-en/product-MS560702BA03-50.html> (visitado 07-04-2024).
- [13] ICM-20948. TDK InvenSense. URL: <https://invensense.tdk.com/products/motion-tracking/9-axis/icm-20948/> (visitado 07-04-2024).
- [14] W25Q128JV - Serial NOR Flash - Code Storage Flash Memory - Winbond. URL: [https://www.winbond.com/hq/product/code-storage-flash-memory/serial-nor-flash/?\\_\\_locale=en&partNo=W25Q128JV](https://www.winbond.com/hq/product/code-storage-flash-memory/serial-nor-flash/?__locale=en&partNo=W25Q128JV) (visitado 08-04-2024).
- [15] FM LORA TRANSCEIVER MODULE, PRE SET TO 868MHZ. URL: <https://www.rfsolutions.co.uk/radio-modules-c10/frequency-c57/fm-lora-transceiver-module-pre-set-to-868mhz-p468> (visitado 08-04-2024).
- [16] LoRa — LoRa documentation. URL: <https://lora.readthedocs.io/en/latest/> (visitado 08-04-2024).
- [17] SIM68M. URL: <https://www.simcom.com/product/SIM68M.html> (visitado 08-04-2024).
- [18] STM32F401VE - High-performance access line, Arm Cortex-M4 core. STMicroelectronics. URL: <https://www.st.com/en/microcontrollers-microprocessors/stm32f401ve.html> (visitado 08-04-2024).
- [19] TE Connectivity. Datasheet de MS5607-02BA03 Barometric Pressure Sensor, with stainless steel cap . 2024. URL: <https://www.farnell.com/datasheets/2917207.pdf>.
- [20] TDK InvenSense. Datasheet de ICM-20948 World's Lowest Power 9-Axis MEMS MotionTracking™ Device 2024. URL: <https://docs.rs-online.com/637f/A700000007834295.pdf>.
- [21] Winbond. Datasheet de W25Q128JV 3V 128M-BIT SERIAL FLASH MEMORY WITH DUAL/QUAD SP 2024. URL: <https://docs.rs-online.com/cdee/0900766b81622f85.pdf>.

- [22] RF Solutions. Datasheet de RF-LORA-868-SO LongRange Transceiver. 2024. URL: <https://4donline.ihs.com/images/VipMasterIC/IC/RFSO/RFSO-S-A0009067532/RFSO-S-A0009067532-1.pdf?hkey=6D3A4C79FDBF58556ACFDE234799DDFO>.
- [23] SIMCom. Datasheet de SIM68M Hardware Design. 2024. URL: [https://drive.google.com/file/d/16DcqSZpx00Y\\_eTA23IL3Wg06BYqADSZn/view?usp=sharing](https://drive.google.com/file/d/16DcqSZpx00Y_eTA23IL3Wg06BYqADSZn/view?usp=sharing).
- [24] MPS. Datasheet de MP1476 18V, 2A, 800kHz, High-Efficiency, Synchronous, Step-Down Converter in SOT. 2024. URL: [https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document\\_type/Datasheet/lang/en/sku/MP1476/document\\_id/3726/](https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP1476/document_id/3726/).
- [25] TDK. Datasheet de SD1209TT-A1 Electromagnetic buzzers Pin terminal(Without oscillator circuit). 2024. URL: <https://www.farnell.com/datasheets/2720059.pdf>.
- [26] ST-LINK/V2 - ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32 - STMicroelectronics. URL: <https://www.st.com/en/development-tools/st-link-v2.html> (visitado 08-10-2024).
- [27] Introduction to USB with STM32 - stm32mcu. URL: [https://wiki.st.com/stm32mcu/wiki/Introduction\\_to\\_USB\\_with\\_STM32](https://wiki.st.com/stm32mcu/wiki/Introduction_to_USB_with_STM32) (visitado 04-05-2024).
- [28] ST Microelectronics. Datasheet de STM32F401VET6 ARM® Cortex®-M4 32b MCU+FPU, 105 DMIPS, 5. 2024. URL: <https://www.st.com/content/ccc/resource/technical/document/datasheet/30/91/86/2d/db/94/4a/d6/DM00102166.pdf/files/DM00102166.pdf/jcr:content/translations/en.DM00102166.pdf>.
- [29] Diseny i muntatge de circuits electrònics. FADESA INGENIEROS. URL: <https://fadesaing.com/ca/inici-2/> (visitado 15-09-2024).
- [30] Lars Boegild Thomsen. lbthomson/stm32-w25qxx. original-date: 2022-02-26T03:59:38Z. 15 de sep. de 2024. URL: <https://github.com/lbthomson/stm32-w25qxx> (visitado 15-09-2024).
- [31] GitHub - UravuLabs/MS5607: Arduino Library for TE Connectivity Pressure Sensor MS5607. URL: <https://github.com/UravuLabs/MS5607/tree/master> (visitado 15-09-2024).
- [32] Eunhye Seok. mokhwasomssi/stm32\_hal\_icm20948. original-date: 2020-12-02T13:38:55Z. 19 de sep. de 2024. URL: [https://github.com/mokhwasomssi/stm32\\_hal\\_icm20948](https://github.com/mokhwasomssi/stm32_hal_icm20948) (visitado 03-10-2024).
- [33] GitHub - sztvka/stm32-nmea-gps-hal: My take on the STM32 NMEA GPS Library. URL: <https://github.com/sztvka/stm32-nmea-gps-hal/tree/master> (visitado 03-10-2024).
- [34] CEASTech. LoRaFi/LoRaFi. original-date: 2017-01-28T10:34:24Z. 10 de feb. de 2022. URL: <https://github.com/LoRaFi/LoRaFi> (visitado 03-10-2024).